A System Framework for Non-Intrusive Monitoring of HMI States for Detecting Human-in-the-Loop Error Precursors

by

Harshvardhan P. Singh

A thesis submitted to the
School of Graduate and Postdoctoral Studies in partial
fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical and Computer Engineering

Department of Electrical, Computer and Software Engineering
University of Ontario Institute of Technology (Ontario Tech University)
Oshawa, Ontario, Canada
April 2020

THESIS EXAMINATION INFORMATION

Submitted by: **Harshvardhan P. Singh**

Doctor of Philosophy in Electrical and Computer Engineering

Thesis title:

A System Framework for Non-Intrusive Monitoring of HMI States for Detecting Human-in-the-Loop Error Precursors

An oral defence of this thesis took place on April 7, 2020 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee	Dr. Ying Wang
Research Supervisor	Dr. Qusay H. Mahmoud
Examining Committee Member	Dr. Lixuan Lu
Examining Committee Member	Dr. Jing Ren
University Examiner	Dr. Faisal Qureshi, Comp. Science
External Examiner	Dr. Medhat Moussa, Univ. of Guelph

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

ABSTRACT

A System Framework for Non-Intrusive Monitoring of HMI States for Detecting Human-in-the-Loop Error Precursors

Harshvardhan P. Singh
Ontario Tech University, 2020

Advisor:
Dr. Qusay H. Mahmoud

Past industrial accidents suggest that increased automation has had an adverse effect on operator situational awareness (SA) and Human-in-the-Loop (HITL) errors. More so, any current automation is not yet capable of providing the level of situational awareness that a human operator can provide and close the ethical responsibility-gap.

The objective of this research is to design a system for detecting HITL error precursors to ascertain operator SA (as a function of operator activity index) in real-time via non-intrusive monitoring. The proposed system frameworks are ViDAQ and *EYE-on-HMI*. ViDAQ is a visual data acquisition system that uses computer vision techniques to capture dynamic visual feedback from the industrial human-machine interface (HMI) (e.g., control panels) states. For example, ViDAQ results demonstrate approximately 90% accuracy at 1 meter acquisition distance when reading a multi-dial rotary-style meter. Positive results, coupled with real-world control room settings that offer constant lighting and a vibration-free environment, support the future efficacy of ViDAQ.

The *EYE-on-HMI* is a novel expert supervisory system that models HMI state patterns (as captured from ViDAQ) under normal and abnormal conditions, to detect HITL error precursors in real-time. It is developed using a variety of modeling techniques: linear regression (ARIMA), recurrent, and convolutional neural network (RNN and CNN) for HMI time-series modeling; and *Seq2Seq* deep learning natural language processing (NLP) models for HMI discrete event system model. As an example, relative root-mean-square-error in forecast accuracy is $RMSE \approx [30\%, 80\%, 100\%]$ for $N - ahead > 10$ time-step forecast window. This suggests regression-based models are closely followed by RNN, CNN, and NLP models in order of forecast accuracy achieved using synthetic HMI state datasets. Nevertheless, RNN and CNN are more versatile and scalable than the regression models during the training and evaluation phases, which is also anticipated for large scale multi-variate industrial HMI datasets. Moreover, the NLP based models embed contextual dependencies as semantic relations between HMI states for complex event patterns to improve HITL error precursor detection accuracy.

Lastly, the proof-of-concept HITL error precursor detection using CANDU^{TM} Nuclear Control Room Operator training simulator is demonstrated.

Keywords: Auto Regression Integrated Moving Average (ARIMA), Computer Vision; Human Machine Interface (HMI); Human-in-the-loop (HITL) error; Long-Short Term Memory (LSTM); Nuclear Power Plant (NPP); Natural Language Processing (NLP); Recurrent Neural Networks (RNN), Time series.

AUTHOR'S DECLARATION

I hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners

I authorize the University of Ontario Institute of Technology (Ontario Tech University) to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize University of Ontario Institute of Technology (Ontario Tech University) to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

_____ Harshvardhan P. Singh

ACKNOWLEDGEMENTS

This research would not have been possible if it weren't for the support of the following individuals.

My supervisor, **Prof. Qusay Mahmoud**, whose patience, guidance, and unwavering confidence in me helped me to grow as a researcher and as an independent thinker.

My wife, **Priya**, whose unparalleled patience, dedication, support and understanding in allowing me to accomplish this journey is remarkable.

Dr. Ashraf Z. Sadek, who has always been a mentor, a guiding light in my low times, and a good friend. I shall always remain indebted to.

Naweed Tajuddin, has always been there for me whenever I needed to bounce any ideas of life.

Abraham Varghese, who selflessly always found the time to guide me. Especially, for providing technical assistance with implementing a few ideas for this research work.

My mother, **Girija**, who has always inspired me to contribute to the research community and to undertake this endeavor.

To my little daughter, **Gourangi** from whom I have borrowed those precious tender years as a loan during the course of this research. I now expect to return this time to her with interest.

Grateful for all the contributions from my professional colleagues and friends:

Kim Iwasa-Madge and **Scott Malcolm**, for providing invaluable human factor engineering references, key concepts, and guidance.

To my work supervisor, **Tim Collier**, for being always supportive and offering the flexibility to make alternate arrangements to make up the time and conduct academic research with full-time work.

Gérald Parent, retired OPG licensed nuclear operations supervisor and currently an instructor for OPG AOOM (Advanced Operations Overview for Managers) nuclear systems course delivered in Ontario Tech University (OTU) and **Khalid Rizik**, for providing key help in setting up actual scenarios/scripts on the full-scope CANDU^{TM} OPG Darlington Nuclear control room operator training simulator located in OTU.

Lastly, I would like to offer this work as a dedication to my father, **Uday Pratap Singh**, whose constant and many a time under-appreciated support and strength, I am eternally grateful for.

STATEMENT OF CONTRIBUTIONS

I hereby certify that I am the sole author of this thesis. Results from this thesis research have been disseminated in the following publications:

- H.V.P. Singh and Q.H. Mahmoud. (2019). *ViDAQ: A Computer Vision Based Remote Data Acquisition System for Reading Multi-Dial Gauges.*" Elsevier Journal of Industrial Information Integration. 15(Sep 2019): 29-41.
- H.V.P. Singh and Q.H. Mahmoud. (2019). "*LSTM-based Approach to Monitor Operator Situation Awareness via HMI State Prediction.*" IEEE Proceedings. IEEE International Conference on Industrial Internet, Orlando, United States (pp. 1-10)
- H.V.P. Singh and Q.H. Mahmoud. (2018). "*Non-Intrusive Monitoring of Operator Situational Awareness via Human-Machine Interface States. Proceedings of Canadian Nuclear Society Conference.*", 8th Canadian Nuclear Society International Conference on Simulation Methods in Nuclear Science and Engineering, Ottawa, ON, Canada (pp. 1-13)
- H.V.P. Singh and Q.H. Mahmoud. (2017). "*HMI-Guard: A Platform for Detecting Errors in Human-Machine Interfaces.*", IEEE Proceedings. IEEE International Conference on Systems, Man, and Cybernetics, Banff, AB, Canada (pp. 2861-2866)
- H.V.P. Singh and Q.H. Mahmoud. (2017). "*ViDAQ: A Framework for Monitoring Human Machine Interfaces.*" IEEE Proceedings. 19th IEEE International Symposium on Real-Time Computing, Toronto, Canada (pp. 141-149)
- H.V.P. Singh and Q.H. Mahmoud. (2017). *EYE-on-HMI: A Framework for monitoring human machine interfaces in control rooms.*" IEEE Proceedings. 30th IEEE Canadian Conference on Electrical and Computer Engineering, Windsor, ON, Canada (pp. 1-5).

I have used standard referencing practices to acknowledge ideas, research techniques, or other materials that belong to others. Furthermore, I hereby certify that I am the sole source of the creative works and/or inventive knowledge described in this thesis.

Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

ANN Artificial Neural Network.

ACF Auto-correlation Function plot. Shows how much a signal is correlated with itself at various time-steps.

ADF Augmented Dickey-Fuller Test (null-hypothesis testing for unit roots).

ARIMA Autoregressive Integrated Moving Average model.

CANDU™ CANada Deuterium Uranium. A Canadian pressurized heavy water class of reactor technology. (Registered trademark of Candu Energy Inc., a subsidiary of SNC-Lavalin Inc.).

CHT Circular Hough Transform.

CNN Convolutional Neural Network.

CPS Cyber Physical System.

CRO Control Room Operator.

CV Computer Vision.

CvE Convex-Hull Edge List; Each edge is defined by a start and end vertex tuple.

DCS Distributed Control system.

DSP Digital Signal Processing.

EYE **E**xpert Supervisor**Y** Syst**E**m (EYE). *EYE-on-HMI* is a EYE for monitoring HMI systems.

EEG Electro Encephalogram (brain electrical activity) recorder.

HFE Human Factors Engineering.

HITL Human in the loop.

HMI Human Machine Interface.

IAEA International Atomic Energy Association.

INES International Nuclear Event Scale.

InS	In-Sample values. When a feature sample value for testing is taken from the same training set that the model was trained on previously.
LSTM	Long Short-Term Memory (RNN) model.
MT	Machine Translation.
NLP	Natural Language Processing
NPP	Nuclear Power generating Plant.
NN	Neural Network. a general term for all types of layered trainable networks.
OuS	Out-of-Sample values. When a feature sample value for testing is taken from a any set other than the training set that the model was trained on previously.
PAP	Perimeter Approximated Polygon based CvE list enumeration.
PCAF	Partial Auto-correlation function plot. Shows how much a signal is correlated with itself at various time-steps with the past corellational effects removed.
RDP	RamerDouglasPeucker based CvE list enumeration.
ROI	Region Of Interest - sub images processing.
$RMSE_p$	Root-mean-square Persistence score. Used as a base line score for $p-lagged$ persistence model.
RNN	Recurrent Neural Network
SCADA	Supervisory Control and Data Acquisition system.
ViDAQ	Visual Data Acquisition.

Chapter 1

Introduction

Several severe industrial accidents in the Nuclear Power Plant (NPP) and the aviation industries have brought about significant improvements in reducing human performance errors and identification of human factors engineering (HFE) deficiencies with legacy HMI designs. However, current legacy HMI systems design philosophy does not incorporate the detection of operator errors and validation of operator actions in real-time and non-intrusively.

This chapter briefly reviews past few significant industrial accidents in the nuclear power and aviation industry that were pivotal in bringing about significant improvements in human error reduction measures to set the context for drawing motivation for this research.

1.1 Industrial and Aviation Accidents

A historical review of significant accidents in the NPP industry that have been rated high on the severity scale (ranging between 5 to 7) of the IAEA International Nuclear Event Scale (INES) [1] underscore a common theme. This theme includes confounding operator performance issues combined with inherent HFE design flaws in legacy control room HMIs and poor equipment status monitoring practices. Combined effect of which, led to catastrophic failure of stand-by safety-critical systems essential to remove reactor decay heat during post SCRAM (emergency reactor trip) event.

Namely, with the NRX (National Research Experimental) reactor at Chalk River Labs, Canada (INES-5). The accident initiated owing to multiple failures involving misleading control rod status indicator lights in the control room, mechanical failures, and miscommunication between the control room and field operator personnel. This led to the accidental withdrawal of the safeguard bank of shut-off rods which caused an uncontrolled reactor power excursion over 4 times its design limit in a matter of 5 seconds, resulting in a severe core damage on December 12, 1952.

Three Mile Island, USA (INES-5), accident initiated owing to poorly designed ambiguous control room indicators, which introduced operator error to override the emergency cooling water supply, causing a partial meltdown of the TMI-2 reactor core containment on March 28, 1979.

In the Chernobyl disaster, USSR (INES-7), where confounding human factors and inherent design flaws led to a catastrophic reactor Unit 4 explosion and release of radioactivity on April 26, 1986.

Aviation industry accidents such as Tenerife Airport Disaster, March, 1977, involving two passenger planes one taking off and another just having landed on tarmac collide head-on. It was by far one of the worst on ground accidents causing fatalities of over 500 lives. Contributing factors included a combination of poor visibility due to fog and unclear communication between air traffic control and the two pilots involved, resulting in a loss of situational awareness in knowing precisely if the run-way was cleared for take-off. Unfortunately, within the fog, by the time the plane taking off saw the other plane landing, it was too late.

Asiana Flight 214 in July, 2013, a South Korean airliner (Boeing 777) outbound to San Francisco pilot misjudged final approach leading to the plane clipping a sea wall before crashing and bursting into flames. Pilot fatigue was one of the contributing human factors.

Similarly, recent accidents in the aviation industry, such as Lion Air Flight 610 outbound from Jakarta, Indonesia, tragically crashed on October 29, 2018. Within five months, Ethiopian Airlines Flight 302 outbound to Nairobi, Kenya, crashed on March 10, 2019. These accidents caused fatalities of over 300 lives. Both these accidents were traced to a single point vulnerability in the last design update, associated with a single malfunctioning sensor on the Boeing 737 Max 10, which falsely triggered the Maneuvering Characteristics Augmentation System (MCAS). A failure mode unbeknownst to its pilots due to improper training provided by the aircraft manufacturer and lack of proper indication in the cockpit repeatedly pushed the aircraft's nose down, causing it to dive uncontrollably despite pilot interventions.

Hence, Nuclear and Aviation industry accidents indicate a common theme of confounding factors that challenge human operator performance owing to human factor engineering-related design flaws in legacy HMI system designs [2]. Key accident precursors as evident from post-accident reports [3, 4, 5] reveal : (1) reduction in situational awareness owing to human factors related deficiencies in legacy HMI design; (2) normalization to deviance to lax engineering design control measures and reduced safety culture due to production pressures; (3) information overload (looking-but-not-seeing effects [6]) owing to large volume and rate at which information was presented to human operators via the control room/cockpit HMIs (panel indications, annunciations, etc); and (4) incorrect mental model of highly dynamic unit evolutions resulting in cognitive errors, owing to conflicting plant information supplied by

failed or faulty sensors, as some of the root-causes of such accidents.

In efforts to minimize above accident precursors, both NPP and commercial aviation industries have incorporated frequent operator and pilot training programs covering several accident scenarios in simulation to prepare human operators to: (1) mentally filter non-essential information and prioritize responses; (2) validate assumptions; (3) constantly keep-up their situational awareness to align mental models of actual plant process state; and (4) practice strict adherence to operating and alarm response procedures to control an evolving transient.

Therefore, an independent system that can continuously monitor the HMI states from an operators viewpoint can help determine the information rate, operators' workload pattern, and gauge SA trend.

1.2 Research Space

Industrial control rooms such as in a Nuclear Power Plant (NPP) (Fig. 1.1) contain densely populated system control panels.

Figure 1.1: In a typical NPP control room, there are densely populated panels with indication devices for various system specific processes. These are the primary source of plant information status as presented via visual feedback to operators

Each control panel is a Human Machine Interface consisting of various devices that visually convey the plant process status. A group of highly trained control room operators interact with and observe control panels as a primary source of information that is conveyed via visual feedback.

While the the concepts and proposed system frameworks presented in this thesis are generalized, they are first developed with reference to its envisioned application in a typical NPP industrial control room environment, as shown in Fig. 1.1 [1].

Below are key characteristics that are unique to NPP control room HMIs and nuclear safety culture based on the author's professional familiarity with this area. These experiential observations are essential considerations that are embedded in the formulation of the proposed solution for this research space, as presented in this thesis.

1. **Ambient Lighting**: Control room ambient lighting is always maintained at a fixed level as per human factors standards.

2. **Panel View**: Panel view obstructions are always kept to a minimum. Operators maintain a safe distance and limit crowding near the control panel to allow a clear obstruction-free view of the control panels for everyone at all times. There are very few sources of optical disturbances, including vibrations, reflections, glare, etc.

3. **Lamp Indications**: Control panel lamp indications are of fixed colours, and each corresponds to a particular field device state.

4. **Lamp Burn-out**: Lamp indicators, if burnt-out, fail to light up and, in some cases, go undetected for a long time.

5. **Lamp Test Routine**: Operators follow a routine to check all lamp states once per shift by doing all lamp test.

6. **Push Buttons**: Most panel push-buttons have backlights to offer positive visual confirmation once depressed, and the state has been toggled.

7. **Hand Switch**: Most panel hand switches have position indicator notches to indicate its state visually. Rocker style switches have a lever that either flips horizontally or vertically with respect to its base.

8. **Fail Safe design**: Most control panel devices are designed with fail-safe state (lamps are an exception), which is made visually apparent for quick identification of failure. For example, dials are spring-loaded to return to zero reading if field signal drops or goes irrational.

9. **Panel dial gauges**: There are few general dial gauges of different shapes. The majority of dials are rotary single needle type, while few are multi-dial gauges.

[1] Darlington Nuclear Generating Station in Clarington, Ontario, Canada. Source: "Like it or not, Toronto is a nuclear city" circa Feb. 19, 2016. The Globe and Mail

10. **Limited Digital Reading**: In legacy control rooms, not all control panel HMI devices are being logged digitally by a data acquisition or SCADA system. Therefore, it is only by Operator manual reading that this data can be obtained.

11. **Nature of Patterns**: It may be useful to note specific panel indications and operator input devices normally remain in steady-state for long durations compared to others that are frequently changing. In either case, operators need to know about any state changes, however brief or frequently it may occur. Currently, this task is often left to the operator vigilance and frequent panel checks.

12. **Operator Procedures**: Nuclear operators are trained to follow procedures while operating these control panels. Therefore, the sequence of interactions is normally fixed per procedure. Procedural sequence involves doing an action and observing expected field process response as seen via the control panel state. Control panel state comprises of all process indication device states, value readouts, and operator action feedback (E.g. hand switch positions, push-button illumination, etc.).

13. **Panel Checks**: Nuclear operators are trained to rely on constant situational awareness to cross-validate control panel indication states against field conditions as reported via other computerized data acquisition HMIs.

14. **Independent Validation**: For certain critical steps, two operators are engaged. One operator executes a procedure while the other observes and follows the same procedure to validate the steps and expected panel State independently.

15. **Situational Awareness**: Nuclear operators are trained to keep their situational awareness (SA) in check all the time. There are several personal cognitive reinforcement techniques they are trained to use. One of which is they verbalize what they are visualizing on the control panels. They are also requested to self-report after every shift if they missed any steps. The control room supervisor also monitors operator behaviors for fatigue and distractions to reduce changes in situational awareness from dropping.

16. **Human Error**: Human cognitive error rate is most probable to increase as operator situational awareness drops. Numerous factors: multiple process distractions during a unit evolution, non-routine maintenance activities, increase in operator workload, *looking-but-not-seeing* effects. These may be reflected in *quality* of operators control panel actions. Where *quality* of action, here refers to execution accuracy, which is affected by one or more missed, out-of-sequence, and mistimed actions. Operator sit-

uational awareness is also ascertained objectively by shift supervisors by reviewing aggregation of such self-reported metrics.

17. **Nuclear Security**: Nuclear power generation premises and control rooms require the highest security measures. Cameras and any personal recording devices are not permitted here. Therefore, placing surveillance cameras in control rooms is not allowed to protect personnel identity and details of critical safety systems design.

Based on the above observations, situational awareness (SA) is a paramount metric of human performance in the Nuclear Industry. However, SA is difficult to track in real-time systematically. Panel instrumentation states may be used to determine operator action.

Panel indication faults are manually checked but not monitored in real-time. Nuclear power plants rely on *defence-in-depth* philosophy in all aspects, E.g., systems design, operational procedures, operator training and safety systems or safeguards, to allow multiple designed barriers to fail before an accident can occur. Therefore, the uniqueness of the Nuclear power generation industry compared to other industrial operations is the urgency of early detection and expedient resolution of system faults. Consequently, it's essential panel indicator faults (even if its a small burn-out lamp) are identified and rectified as soon as possible.

Cameras may not be allowed in control rooms. However, there is an opportunity to gradually introduce computer vision technology in control room operator training simulators to collect valuable operator training and human performance modeling data.

1.3 Research Statement

The current state of automation in nuclear power generation and aviation industries still rely on highly trained NPP CRO (Control Room Operators) and airline pilots for their acquired cognitive skills to overcome the fundamental limitation inherent in the conventional operator based command - control - feedback architecture (Fig. 1.2-A), vis-á-vis errors injected by human command inputs via HMIs.

HITL errors are likely to manifest due to reduction in operator situational awareness or misinformation being displayed on the HMIs due to some malfunction. While rigorous operator training does minimize human command input errors, in reality, latent HMI system flaws continue to fatigue the human brain owing to sensory overload. This ultimately increases the chances of human-in-the-loop cognitive errors.

Hence, a research hypothesis is if the pattern of HMI states, as visually apparent to operators, can be modeled, then it may be possible to detect HITL error precursors and monitor operator situational awareness in real-time.

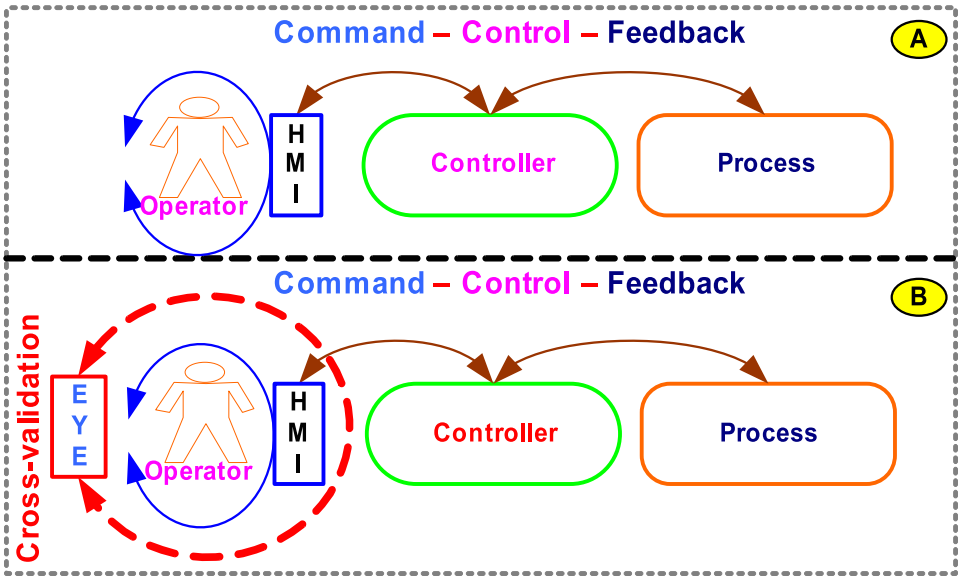


Figure 1.2: *EYE-on-HMI* Conceptual Framework. Architecture of: (A) Operator based command-control-feedback process; (B) *EYE-on-HMI* framework adds cross-validation to operator command-control-feedback process [7].

Therefore, the problem statement is as formulated: *Given an HMI Cyber-Physical System, design a computer system framework to detect HITL errors via real-time non-intrusive and indirect monitoring of operator actions.*

In this thesis, a broader term used to refer to the general approach of indirect and non-intrusive monitoring of operator and HMI interactions is also referred to as *Inverse Oculism*, or looking-inward on the HMI to infer operator situational awareness rather than directly monitoring the operator actions. The operator actions are intended to be captured as operator-caused state changes on the HMI, e.g. switches, push buttons, setpoint displays.

The proposed conceptual framework as depicted in Figure 1.2-B captures the above concept and addresses the following identified industry gaps:

Lack of Real-Time Supervisory System Currently, there is a lack of a formal real-time supervisory system framework (EYE) to cross-validate observed HMI states against operator actions, as represented in Fig. 1.2-B. Traditional human supervision may not be practical at all times in modern control rooms. Typically, control room supervision is limited to only providing a high level oversight over the actions of trained CROs in response to specific plant conditions, E.g. addressing critical plant transients. Hence, a system framework that combines continuous real-time non-intrusive monitoring of operator interactions with HMIs and situational awareness trending would be an added safety enhancement in the nuclear power generation and aviation industry.

Limitation of Risk-Based Modification Despite the advances in HMI designs and real-time process control technologies, several industrial complexes are significantly lagging in utilizing the potential benefits of advancement in high-performance computing platforms and deep machine learning techniques for pattern recognition and data analytics [8]. The slower adoption is owing to a conservative risk-based engineering design modification philosophy, which limits any changes to the existing design basis in order to prevent inadvertent flaws being introduced in the legacy design.

Therefore, from a systems engineering perspective, there would be a significant redesign and retrofit cost for a modification that collects all legacy control room panel indications and panel switch states to track operator actions fully. Therefore, modifications that involve the least change to physical systems are preferred. Hence, a remote monitoring solution that uses cameras would be a preferred option to consider.

Existence of Cognitive Errors Man-machine interactions are ubiquitous to every aspect of our lives but are also susceptible to human errors. More so, industries relying on human operators, such as in power generation, robotic manufacturing plants, industrial controls etc., are concerned with preventing human errors during manual and semi-autonomous operation. Cognitive Systems Engineering recognizes these interactions as not always efficient or easy and at times, has turned hazardous (Hollnagel, 2005) [9]. Therefore, monitoring and detecting precursory patterns of human cognitive errors is at the core for this research.

1.4 Research Motivation

The motivation for this research is to develop a system framework to address real-time monitoring of operator situational awareness and early human-in-the-loop error detection.

Industrial control room panels and aviation cockpit are densely populated with devices designed to visually convey the real-time state of the plant process to a dedicated team of operator(s) and pilots. Visual information forms a large part of the cognitive feedback and context required for overall operator situational awareness state. In response, the operators interact with the plant process via the control panel HMIs E.g. push buttons, hand switches, setpoint changes, etc.

The man-machine interactions conveyed by HMI state transition patterns convey both the plant process state and operator actions. These patterns may then be learned by deep learning models to discern subtle pattern variations from previously learned patterns as the basis for detecting specific error precursors.

1.5 Research Objectives

Majority of current human factors engineering research and design principles for man-machine interface design focus on operator task analysis, information presentation, and data visualizations to improve operator situational awareness (SA). However, there is no system framework available that addresses non-intrusive monitoring of human operator situational awareness metrics and performance measurements for real-time monitoring.

The primary objective of the proposed system frameworks is to investigate the feasibility of doing non-intrusive monitoring using visual data acquisition (ViDAQ), as a means of acquiring data from legacy HMI devices (E.g. analog meters, rotary dials, gauges, alarm indicators, etc.). Using ViDAQ for real-time monitoring of legacy industrial HMI devices has the added advantage of being readily deployed to industrial control rooms. The goal is to provide an alternative means that neither requires physical interfacing with target legacy indicator device(s) nor incur expensive retrofits owing to instrument design changes causing production downtime.

The secondary objective is to investigate appropriate HMI data modeling and time-series forecast techniques. These forecast models may be utilized to forecast HMI State patterns for future time windows. It is envisioned that with the ability to accurately forecast expected HMI states, given the past HMI state sequence, it shall allow the trending of operator situational awareness and detection of specific human-in-the-loop error precursors. Moreover, it's expected that the severity of accidents caused due to operator errors can be further reduced, if HITL errors are promptly detected, trended and intervened in real-time.

In summary, despite *"the devil is in the details"* - practical reality will offer several nuances and design challenges with proposed ViDAQ and *EYE-on-HMI*, nevertheless this research addresses the following objectives:

(I) **Non-Intrusive Monitoring**: Investigate the versatility and feasibility of visual data acquisition (ViDAQ) for reading legacy and standard HMI devices used in industrial control rooms (Sec. 3.2).

(II) **Detection of HITL errors**: Investigate methods that utilizes the real-time data collected by ViDAQ to build HMI state and operator response pattern prediction models (Sec. 3.3, Sec. 3.4).

(III) **Evaluate Detection Models**: Demonstrate prediction models can aid in detection of onset of a HITL error precursor. This underpins the requirement of the proposed **E**xpert supervisor**Y** syst**E**m framework for industrial control room HMIs.(Sec. 5.10)

1.6 Thesis Contributions

In this thesis, the idea of real-time non-intrusive monitoring of operator situational awareness and detection of HITL error is approached as a Cyber-Physical systems engineering design challenge[2].

***EYE-on-HMI* Framework:** A novel framework - *EYE-on-HMI* is developed to capture the notion of *Inverse Oculism* - an indirect monitoring approach for capturing human-machine interactions and learning to detect human-in-the-loop precursors in real-time.

Visual Data Acquisition (ViDAQ): Visual Data Acquisition (ViDAQ) as a system framework concept in the context of the NPP control room application is the first such publication on this specific concept that is known to date by the author. The use of image processing and computer vision (CV) techniques to read instrumentation dial gauges is not unique to this thesis; however, components implemented using standard CV techniques for the two use cases presented herein are novel. These include (a) demonstration of remote reading of circular single and multi-dial meters such as clock faces in real-time using a camera and associated multi-dial needle resolution algorithm. (b) demonstration of remote reading of indication lamp states in real-time as displayed on full-scope CANDU™ control room operator training simulator located in OTU[3] (Faculty of Energy Systems and Nuclear Science).

Time-series Modeling of HMI States Using ARIMA: HMI data time-series modeling using Autoregressive Integrated Moving Average models (ARIMA) is evaluated for HMI state forecast using a synthetic HMI state data set.

Time-series Modeling of HMI States Using RNN: Despite the simplicity of ARIMA models, they pose poor scalability over larger feature parameter spaces. It also requires additional initial effort for data preparation and hyper-parameter tunning than other machine learning models using the same size HMI state synthetic data set. RNN models such as LSTMs and CNN models were also implemented and evaluated against ARIMA model performance.

Natural Language Processing Modelling of HMI States: In succession with the above HMI data time-series model, an alternate modeling approach involving Discrete Event System (DES) automata is explored. DES HMI data model allows applying deep Natural

[2]Relevant publications associated with following contributions have been listed in **Statement of Contributions**

[3]Ontario Tech University, Ontario, Canada

Language Translation or NLP modeling techniques for forecasting HMI states. Moreover, the relative effectiveness and scalability for *EYE-on-HMI* industrial applications are also discussed with respect to the former HMI time-series data model.

Custom NLP CNN Encoder-Decoder Model: A standard RNN LSTM NLP Seq2Seq model is first implemented based on current research work. Its evaluated in comparison to a custom-designed, CNN encoder-decoder Seq2Seq model with attention layers (dubbed *Trident*). In addition, two different NLP model training regimes such as: *Teacher Forcing* and *Curriculum Learning* based on recent research works are tried and results compared against RNN NLP models using the HMI state synthetic data set.

Evaluation with NPP Simulator Data Set: Two real-world scenarios for a nuclear power plant are modeled using the full-scope CANDU^{TM} control room operator training simulator (available in OTU, Energy Systems and Nuclear Science dept.). Simulator data collected from these scenarios are used to train the custom NLP model demonstrate that HITL errors are possible to detect before the actual accident(s) event occurring.

1.7 Thesis Organization

The thesis organization outline is as follows :

Chapter 1: Introduction. This chapter includes the motivation for undertaking this research topic as mainly influenced by a few pivotal industrial disasters in the Nuclear Power Generation and Aviation industry. Discussion about the research hypothesis and statement which set the research direction is discussed here. This is followed by describing the specific research problem, its goals, and outlining the main contributions of this thesis.

Chapter 2: Background and Related Works. Presents the scope of the multidisciplinary survey that is required to relevant existing works. This includes a brief background on key enabling state-of-the-art works that can potentially benefit the design of the proposed solution for both non-intrusive monitoring and detection of HITL errors via HMI state forecast modeling.

Chapter 3: Proposed Solution. Presents the two proposed frameworks, system architectures, and theoretical models. Namely, *EYE-on-HMI* and its conceptual testbed framework, ViDAQ system framework. In addition HMI state-space or data models based on time-series and natural language translation concepts are discussed.

Chapter 4: Implementation Detail. Implementation details and algorithms of various core components of the frameworks discussed in the previous chapter are provided under corresponding subsections in this chapter.

Chapter 5: Experiments and Results. Includes an overview of the experimental methodology for evaluating the prototype core components based on the proposed frameworks. Analysis of results is discussed in this chapter.

Chapter 6: Conclusion and Future Work. This chapter concludes by summarizing the current research progress and outlines the road-ahead for this research.

Chapter 2

Background and Related Work

This chapter begins with outlining the multidisciplinary literature space that is explored within the scope of this thesis to identify the gaps. The sections covered in this chapter provide an overview of **Literature Survey Space**, which covers previous relevant works on **Non-Intrusive Monitoring**, **Cyber-Physical System**, **Computer Vision**, **HITL-Error Detection** covering time-series modeling and natural language processing machine learning models.

2.1 Literature Survey Space

This research draws on a multidisciplinary survey of works from various engineering domains. Such as image processing, computer vision, machine learning, and expert systems, as depicted in Fig. 2.1. Each domain has its key enabling technologies that can assist in the realization of the proposed framework. The literature survey space explored in this thesis (Fig. 2.1) allows us to build on the vital knowledge base and to identify existing gaps in current relevant state-of-the-art areas.

2.2 Cyber-Physical Systems

Proposed research adopts Cyber-Physical Systems as a systematic philosophy that accurately envelopes the industrial HMI systems domain. A Cyber-Physical System (CPS) is an integrative perspective of the real-world physical processes augmented with cyber (computation) systems. It captures the heterogeneity, concurrency, distributiveness, and timing requirements naturally intrinsic to such systems.

CPS, as a theoretical perspective, also aligns with the philosophy of the proposed (EYE-

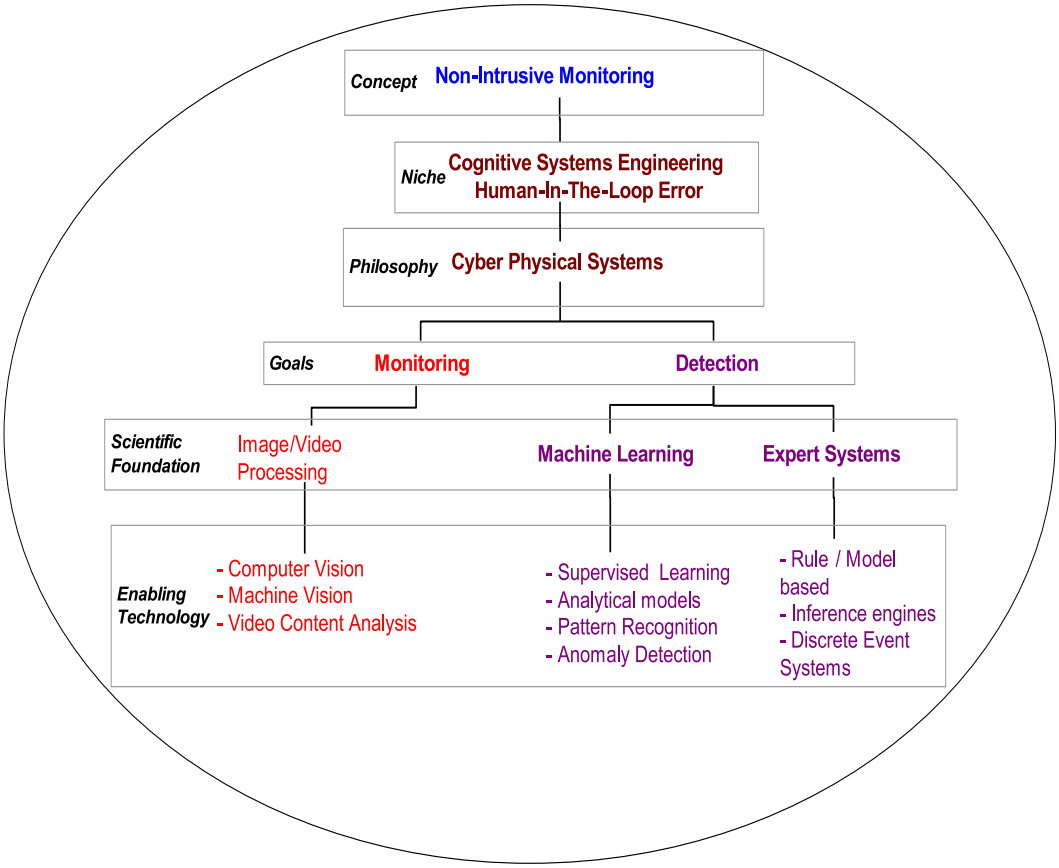


Figure 2.1: Research Space applicable to Research Goals and Key Enabling Technologies required to realize the Proposed Frameworks and Implementation of Evaluation Platforms

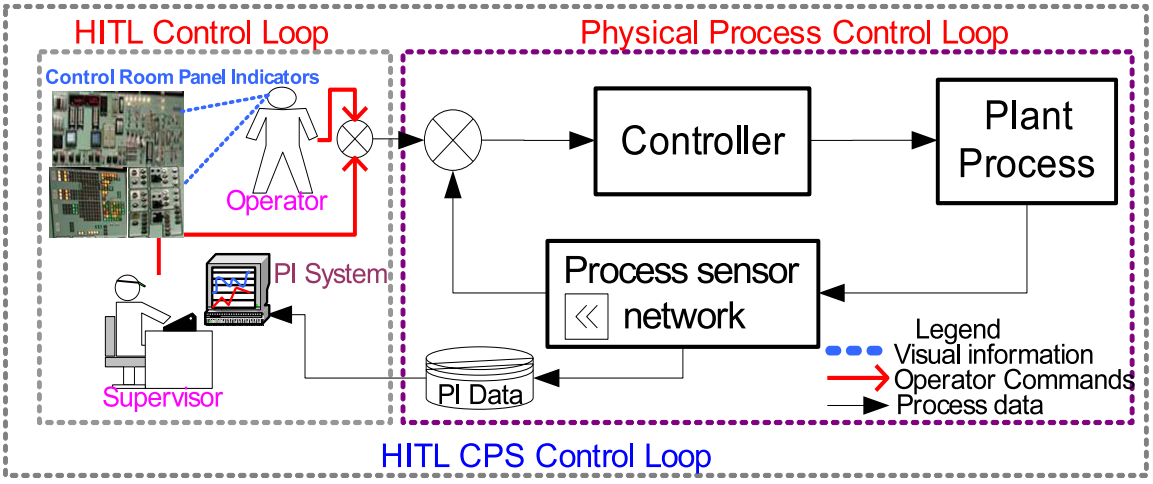


Figure 2.2: Human-in-the-loop (HITL) - A Cyber-Physical System view of a control room environment

on-HMI) framework (Fig. 2.1). It involves a diverse array of sensors, actuators, data networks, and computational algorithms (for control or analysis) to accomplish the safe operation of a

large process or machine, E.g. nuclear power plant, aircraft, etc. Common examples of CPS being ubiquitous to our daily lives include Internet-of-Things (IoT), smart grid, smart homes, smartphones, wearable devices, autonomous vehicles, etc.

CPS abstraction of feedback control systems captures the interaction of sub-components representing technologies from various engineering disciplines such as embedded systems, mechatronics, biomedical systems, computer networks, cloud computing, electric power plants, etc.

The aim is to develop a systems engineering philosophy for building reliable systems in which cyber and physical designs are compatible, synergistic, and integrated at all scales [10].

Without the stretch of the imagination, it is reasonable to extend CPS perspective to existing nuclear power plant (NPP) control rooms (Fig. 2.2) where heterogeneous control loops with real-time computation is required to control complex physical processes. Such as nuclear reactions (neutronic control system), steam generation (boiler control system), electricity generation (turbine governor & generator excitation control system), etc. Furthermore, in NPP control rooms, human operators (CROs) are an integral part of the overall CPS feedback loop. This notion is referred to as human-in-the-loop (HITL) cyber-physical system (CPS) [11].

One of the challenges in HITL CPS, is the lack of a definitive human cognition model, which makes validation and reliability assessment of public safety, critical safety systems difficult [12]. Aptly, this challenge is the focus of human factors engineering (HFE) to design reliable and least error-prone HMIs. HFE takes into account industry best practices and research data on human cognition, information perception models, decision context layers, human performance consequences due to automated decision aids etc. [13, 6].

2.2.1 Cognitive Errors

Human cognitive errors have been broadly classified into two categories based on Rasmussen (1986) [14] skill-based levels of performance, as shown in Fig. 2.3. Firstly, this model indicates Execution Errors are due to skill-based levels of performance owing to Slips and Lapses [15, 16]. The error manifests as a failure to execute the correct intended action under known or anticipated scenarios when the operator's performance behavior is typically skill-driven owing to their training or experience. Specifically, Slips relate to observable flawed actions commonly caused as a result of the attention gap or perceptual failures. These include but not limited to, looking-but-not-seeing effects [6]), interference error (cause distracted response), mistiming error (cause a jerky reaction), sequence error (cause incorrect response), fixation error (cause inadequate response), etc. Lapses, are owing to memory gaps, which manifest as missed steps, repetition errors, delayed response, forgotten steps (recall error), reduced intent

to execute the correct action, etc.

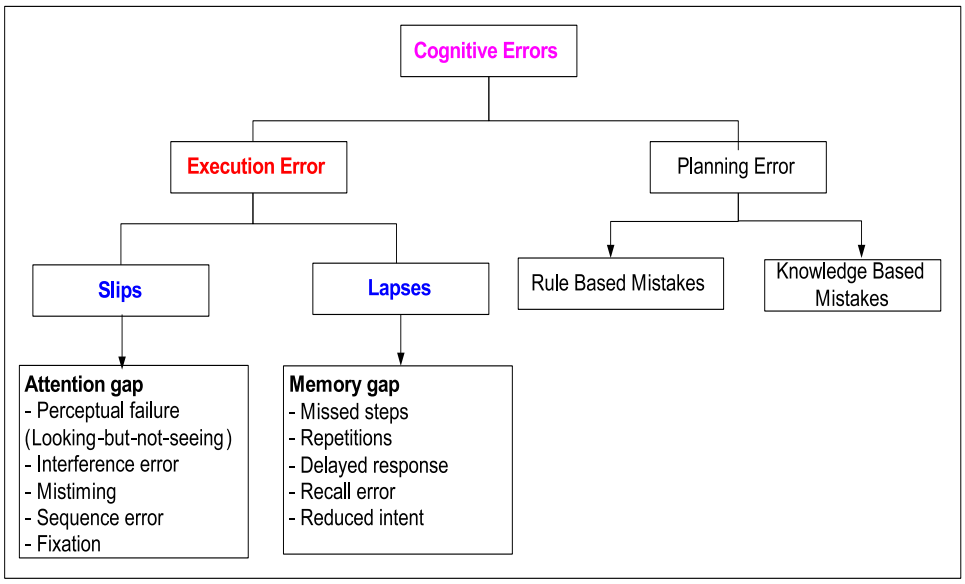


Figure 2.3: Theoretical Framework: Cognitive Errors in Execution and Planning failures based on Rasmussen's (1986) Cognitive Engineering Information processing and Human-Machine Interaction model.

Secondly, Planning Errors or mistakes may be due to an intended action being executed correctly, but the desired outcome is not achieved. These types of errors are often also called knowledge-based mistakes.

Hollnagel (1993) [2], similar to Rasmussen, identified operator actions tend to fail either due to execution errors, which correspond to operator situational awareness and planning failures, which correspond to the rule and knowledge-based levels of performance. The latter omission is introduced initially by knowledge-based workers who plan operational tasks or design systems, which is usually overcome by exercising rigor in the review and independent verification stages of planning.

Although cognitive system engineering [9] primarily focuses on recognizing sources and minimizing occurrences of cognitive errors, there is no current work to the author's knowledge that specifically addresses tracking execution errors, which can only be detected in-situ and in real-time. Therefore, the proposed non-intrusive monitoring approach will aid in the real-time trending of operator actions.

2.2.2 Situational Awareness

Situational awareness is formally defined as *"the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of*

their status in the near future" [17]

Generally, Situational awareness (SA) refers to the level of alignment of the human operator cognitive state with actual process states. Modeling situational awareness is an active area of research in cognitive sciences. Operator mental awareness of the current process state and being able to anticipate future process states based on past experience and training is the highest level of SA an operator can possibly maintain at all times.

Numerous SA cognitive models have been suggested from the widely accepted three-level *Endsley* model [18] (perception, comprehension, and projection stages of SA), to more recent Holistic framework [19] and Causal model [20]. However, as per *Endsley* model, SA is related to the task workload, information rate, system design, and complexity of tasks, requiring multivariate analysis.

Related to SA, modeling Human-in-the-loop (HITL) errors have certainly been the focus of previous research works. Initially, it was primarily motivated by quantifying and estimating Human Error Probabilities (HEP). For instance, notably the legacy human reliability analysis (HRA) [21, 12] approach has been re-hauled (E.g. first, second-generation models [22]) since its first introduction in 1950s [23]. HRA is concerned with qualitatively and quantitatively estimating the human contribution to the risk of error and has also been adopted in probabilistic risk assessment for critical nuclear power plant systems.

Previous works, such as [23], used Bayesian belief networks and incorporated performance shaping factors (PSF) [24, 25] to improve predicting human error probabilities (HEP) relevant to the particular industrial task. PSF, annotates various aspects of the context (or environment) (E.g. HMI design parameters, etc.) and human behavior (E.g. situational awareness) that can impact human performance. For instance, PSF is shown to set the operator context, which ultimately affects HEP. In [26], showed a linear regression model built using a long history of work condition survey database that can then be used to predict certain human error omens (precursors) based on routine personnel surveys done in NPPs.

The particular underlying challenge in previous approaches has remained with adequately capturing a causal relationship between HEP and PSF. However, since HRA approach incorporates behavioral sciences, cognitive psychology, with plant process states and various first and second-generation HRA methodologies [27], there is less emphasis on non-intrusive monitoring and prediction of HITL errors in real-time.

Therefore, a monitoring system that independently monitors the interaction between human operators and HMIs would be a definite improvement in flagging human errors in complex automated HMI systems.

Notably, the level of automation in HMIs also affects operator SA in a systemic fashion. For instance, in the "Automation and situational awareness report" [17], listed are several

aviation accidents that were caused due to accidental failure of legacy automated systems which pilots had come to rely on. These events highlight the concern with an inevitable increase in the level of automation in complex HMI systems.

Furthermore, above conjecture can conservatively be assumed relevant to modern automation design approaches, where human operators are increasingly being placed out of the loop (such as in autonomous or unmanned operations) - report [17] states: "In examining these failures, it becomes apparent that the coupling of human and machine in the form of observer and performer is far from perfect in terms of optimizing the overall functioning of the joint human-machine system.".

Previous techniques for monitoring SA, as addressed by various works [28], recognize it to be a multivariate data analysis challenge. As notably indicated by *Endsley* model [18], SA may be measured using objective measures, often requiring intrusive monitoring of operator physiological parameters (e.g. eye-movement tracking, frequent real-time queries, self-rating, etc.).

SA may also be ascertained (after the fact) by subjective measures, such as operator questionnaires, when the event scenario is suddenly frozen during training. In addition, former objective techniques (e.g. SAGAT, SART [29, 30] scores) that offer a real-time measure of SA by comparing the operators current cognitive state to an expected normal state. But its accomplished via intrusive monitoring techniques, which adds operator burden.

Therefore, using the above principle of objective monitoring as the basis of monitoring SA, the proposed approach of EYE-on-HMI [7] framework relies on a non-intrusive and indirect monitoring of operator pattern of actions, by monitoring HMI indication patterns encompassing operator response sequences.

2.2.3 Expert Systems

Expert systems are synonymous with the recommender, advisory, or decision support systems. Under the strict definition, these are rule-based computer systems that may employ data analytics, machine learning, optimization (fuzzy, genetic algorithms), pattern matching (inference) techniques, to emulate expert human judgment. Logical inference is derived from knowledge rules, decision trees, and historical data in order to assist in forming intelligent decisions with lower error rates. Other classes of expert systems replace rule-based expert knowledge with machine-learned models that are extract knowledge (useful patterns) from online and/or offline raw data.

Legacy expert systems with target applications in the nuclear power plants industry were used for fault diagnosis, operator support, alarm processing, etc. Examples of these include

REACTOR [31] and RiTSE [32] as the earliest examples of rule-based designs. These systems also utilized interpreter based language engines for expressing logical constructs, for example, QES-Shell [33] used turbo-Prolog to process upwards of thousands of rules in real-time to predict malfunctions from the observed symptoms in NPP.

The TPDES (Thermal Performance Diagnostic Expert System) [34] was earliest attempts to combine per component fault tree rules for each steam feedtrain equipment (Heat exchangers, boiler, pumps etc.) in conjunction with real-time sensor data to diagnose thermal losses, which is a critical performance parameter to NPP.

Similarly, APACS (Advanced Process Analysis and Control System) [8] was a working prototype of a real-time monitoring and diagnosis system for feedwater system in CANDU™[1] NPP. It relied on several numerical reference models to reduce the processing of large rule sets by constantly comparing simulation results.

More recent works of expert systems such as Flight Guardian [35], Industrial process failure predictions [36] using SCADA [2] with integrated domain-specific knowledge coupled with real-time process monitoring to augment human decision making by providing intelligent recommendations. Other class of expert systems such as the Alarm Forecasting with SCADA systems [37], show integration of a knowledge generation model which is trained using online raw data.

Specifically, in *Flight Guardian* [35], various nominal range of instrument readings with respect to different flight phases (E.g., take-off, cruise, landing, etc.) are obtained for a given type of aircraft operation. These ranges are then mapped to semi-overlapping membership functions to interpret these in categorical ranges (E.g., low, medium, high) with a degree of probability belief distribution. In addition, the real-time data that is acquired using cameras from the cockpit instrument dials is done independently without imposing any other interfacing requirements with existing aircraft avionics.

A slightly different type of operator decision support system, is *Alarm forecasting for SCADA systems* [37] that uses an online machine learning agent module for forecasting of alarms in the plant process. The framework is designed to use an agent-oriented environment for a distributed SCADA system for large industrial applications. The machine learning agent's purpose is to monitor the current process data and predict specific alarms in predefined periods. Where this approach differs from previous expert systems is that it does not require offline integration of expert knowledge or rules. It instead learns and adapts from the plant process raw data and with corresponding alarm states as labels. Therefore, it gener-

[1] CANDU-Canada Deuterium Uranium is a class of Canadian pressurized heavy-water reactor design used to generate electric power.

[2] Supervisory Control and Data Acquisition system commonly prevalent in industrial process systems for data gathering, HMI and control.

ates knowledge in the form of distinctive predictive patterns learned from raw data, which otherwise may not be very apparent to operators.

The latter described, knowledge generation type of expert system is also being adopted for the proposed EYE-on-HMI framework. As such, the last two approaches are useful to compare and contrast with the proposed EYE-on-HMI frameworks as follows:

1. **Not Rule-Based**: Unlike the rule-based expert systems (E.g. Flight Guardian [35]) that combine existing knowledge base to generate outputs as expert recommendations, the EYE-on-HMI learns offline from the raw patterns of the operator and HMI interactions. This is similar in approach to the *Alarm forecasting for SCADA systems* [37] using a knowledge generation module. However, the goal is the detection of human-in-the-loop error precursors.

2. **Validation vs. Qualification Tradeoff**: Expert systems are deterministic software systems and require extensive validation test effort to test all knowledge rule conditions. However, it may be easier to acquire qualifications for a deterministic type of expert system for industrial applications due to confirmed evidence of adequate test coverage, but also makes such products very expensive. However, knowledge generator based expert systems (E.g. EYE-on-HMI) model validation may be cheaper in terms of evaluating the trained model against the validation data set comprising of select error or accident scenarios, but very expensive to gain qualifications for field deployment in safety-critical applications. This is precisely the concern of the responsibility-gap with AI or machine-learned applications in public safety or risk critical domains.

Therefore, rule-based deterministic expert systems, for example, TPDES [34], APACS [8], Flight Guardian [35] may be expensive to validate but more accessible to obtain qualification for use. Whereas the opposite: cheaper validation and expensive qualification, is expected for knowledge generator (prediction) based expert systems E.g. EYE-on-HMI. Cost-benefit and risk analysis for end-use applications can determine the feasibility of the adoption of a particular type of expert system appropriately.

2.3 Non-Intrusive Monitoring

The primary goal of this research (Sec. 1.5) is to explore the feasibility of Visual Data Acquisition for achieving non-intrusive monitoring of HMI states. The below discussion further clarifies the meaning of non-intrusive monitoring in the context of this research.

Based on the literature survey, human or personnel state monitoring solutions can be mapped on a 2-dimensional space, as shown in Fig. 2.4. The vertical axis shows the degree

of intrusiveness of the monitoring method for various monitoring methods falling on the horizontal axis of how direct vs. in-direct they are.

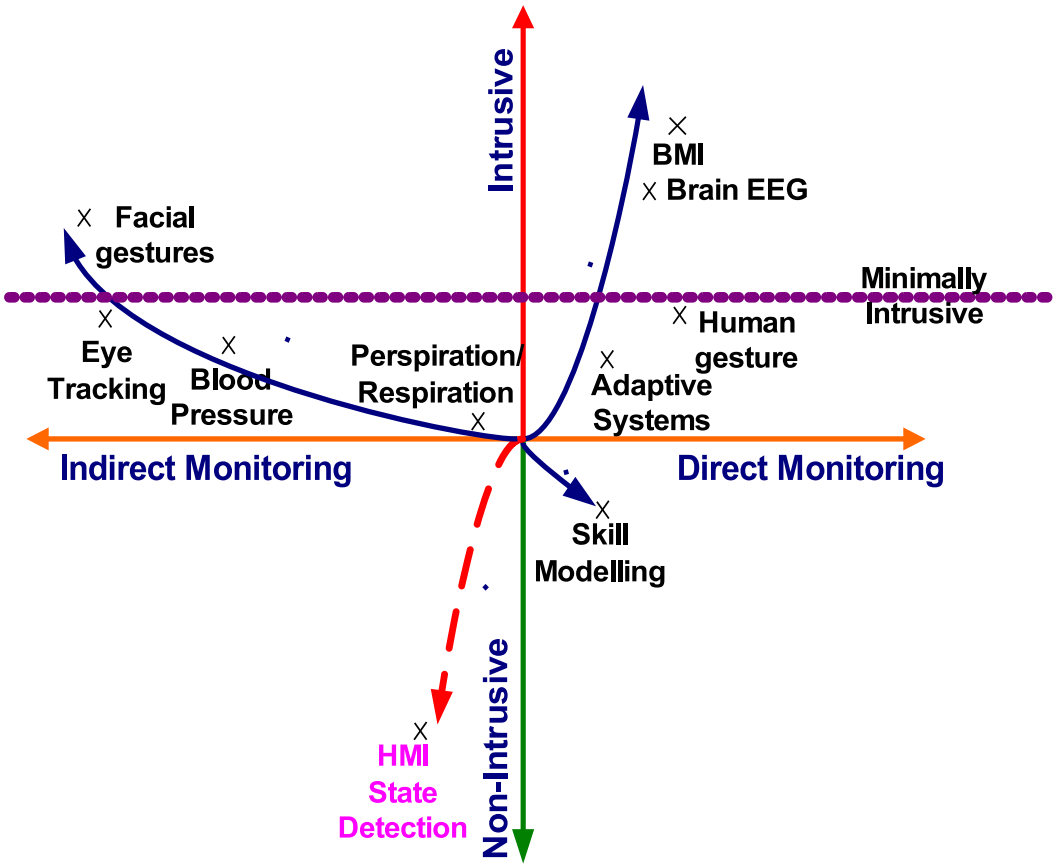


Figure 2.4: Vertical axis is for showing relative subjective degree of intrusive vs non-intrusive nature of the monitoring method. Horizontal axis shows whether the monitoring method is Direct or an Indirect type. The propose ViDAQ framework uses computer vision to remotely acquire HMI state - this is an example of monitoring human operator actions indirect and non-intrusive manner.

The vertical intrusiveness scale ranges from bottom: least *Non-intrusive* to top: most *Intrusive*. Where *Intrusive* can be defined as a monitoring method where a human subject or system is physically equipped with sensors, and the data being collected may potentially raise personal privacy concerns. While, *Non-intrusive* monitoring method that does not affect the human or system physically and does not compromise personal privacy. The majority of the monitoring solutions fall in top half quadrants as having some degree of intrusiveness, e.g. brain EEG [38, 39], human gesture [40], blood pressure monitoring, eye tracking [41], etc.

On the horizontal scale is the monitoring methodology, which ranges from left: most *Indirect* to right: most *Direct* methods. Where, *Indirect* monitoring methods may be defined as those that require only capturing the object state or other correlated parameters that are

affected by human actions. While, *Direct* monitoring method captures the human or subject parameters causing the action directly.

The list of available monitoring solutions are categorized or distributed over the range of direct vs. indirect methods, e.g. eye-tracking methods indirectly infer human operator state such as sleep or distractions. While capturing human gestures is a more direct means of monitoring operator actions using cameras. Current brain-machine interfaces can be considered quite intrusive due to the elaborate dry/wet sensor headset harness/apparatus that needs to be worn by the human subjects. The later is an example of direct monitoring via measuring brain signals, which may be correlated to human actions and thoughts.

2.3.1 Tracking Human Operator Performance

Many other substantial research work exist in the area of directly tracking operator performance and/or modeling operator skills of manipulating an HMI. Such works are of particular relevance to this research (EYE-on-HMI system) from the perspective of exploring existing practical techniques to capture operator actions and the related error from visually apparent HMI states.

The earliest attempt demonstrated by *Tustin, 1947* [42], was where the relationship between the operator action (i.e. manual actuator handle position) and the error (the difference between the gun position and a moving target), as perceived by the operator vision was sought. It was demonstrated that the existence of a non-linear relationship between operator control action and the magnitude and rate of change of perceived error. However, as the non-linear human skill model has little practical use and so it was approximated by a "linear-law" (linear servo control model). Linear law can also be used as a basis for the improvement of the controller or HMI design while taking into account human-in-the-loop errors. In this work, the operator action and response data were directly captured by recording the deflection of the actuator control stick in response to operator visual feedback of the tracking error.

Similarly, for HAM (Human Adaptive Mechatronics) by *Suzuki et al, 2005* [38], a novel intelligent human-machine mechatronic system that can adapt to the level of operator skills under various environments to further assist and improve the user's skill. Their goal was to achieve a catholic method for user skill evaluation for any man-machine manipulation task. Evidently, among other identified requirements in the realization of the HAM, was the need for definition and quantification of human skills.

The next challenge of skill quantification has been researched extensively by modeling human skills. Like the linear servo control model by *Tustin* [42], *Ragazzini* [43] recognized human action as a random time-variant system which they later approximated as a PID (Pro-

portional Integrative Derivative) controller. Other studies done by *Cabrera, 2002* [44], also concluded other interesting facts about human cognition or skills expressed while balancing a stick on a finger (classic control problem of an inverted pendulum stabilization), which is highly non-linear in nature.

With *Cabrera, 2002* [44] findings on inherent characteristics of skills expressed as human behavior, *Suzuki et al, 2005* [38] built a system to acquire data using a pressure sensor to detect the force of grip applied to a physical slider by the user while trying to stabilize a virtual, inverted pendulum fixed on a cart. The visual graphics and haptic slider feedback of virtual cart dynamics were computer-simulated. In addition to capturing the force of grip, human user's brain activation data was also measured using near infra-red spectroscopy (NIRS). This technique is used to detect cortical changes in oxygenated vs. deoxygenated hemoglobin subcutaneously. For this a 48 channel probe head harness was used to collect NIRS signals from the primary motor and somatosensory cortex based on motor homunculus locations (important for voluntary motion) on the user's scalp.

Brain NIRS data were analyzed using principal component analysis (PCA) to determine which parameter information of the pendulum system (E.g. angular velocity, angle position etc.) showed a strong correlation with the user cortical data. This result can then be used to distinguish between various levels (low, medium, high) of skilled users.

Alternatively, several works [45, 46, 40], focused on using minimally intrusive means of capturing human actions. Such as in *Kapoor & Picard, 2005* [45] used a camera and a pressure-sensitive chair to capture head gestures and posture to infer user emotional states. While in *Kapoor & Picard, 2001* [46] focused on tracking pupil movements in real-time to determine particular body language poses indicative of interest or frustration emotions.

In *Behoora & Tucker, 2015* [40], demonstrated use of non-wearable sensors such as off the shelf infra-red cameras to capture human poses based on skeletal joint positions, as shown initially by [47]. Furthermore, commercial devices such as Microsoft Kinect, has shown [48] to be used to detect human gestures and action recognition using joint-skeletal data as classified by use of support-vector-machine (SVM) based classifiers.

This data was used by [40] to build a real-time automated system to monitor participants emotional states to understand the overall team dynamics better.

Recent advancement on HITL (human-in-the-loop) system designs using direct brain-computer interfaces is increasingly becoming an active area of research. In [49], a self-adaptive stock trading application solution that utilizes variables from the Opportunity-Willingness-Capability (OWC) ontology [50] developed for the framework modeling task planning of cyber-human systems [51]. A pre-trained neurobehavioural reference module was used to monitor mental states via brain EEG along with stock buy/sell decisions made by a human

trader. A stock performance comparator in retrospect decides whether future software decision of buy/sell stocks should use favorably human inputs or not to enhance performance adaptively.

In contrast with the indirect monitoring works discussed above, the proposed EYE-on-HMI framework instead utilizes HMI state patterns apparent from the viewpoint of the operator, which may be captured using computer vision (ViDAQ) techniques. This approach exemplifies an indirect and non-intrusive (inverse-oculism) monitoring option, where the human operator actions with respect to an HMI or control panel(s) are detected as visual feedback changes registered on the HMI.

Therefore, previous research works (reviewed above) suggest that the majority of the current state-of-the-art direct and indirect methods of human performance monitoring solutions involve a level of intrusiveness that is considered higher than what the proposed *EYE-on-HMI* framework is advocating.

2.3.2 Computer Vision

Computer Vision technology is currently advancing at a high pace. It includes image, video processing (Fig. 2.1), object recognition algorithms and techniques to enable machines to process visual information efficiently.

Machine Vision

Notably, Machine Vision (MV) is a subset of several technologies under computer vision that has seen applications in robotics [52], intelligent transportation systems [53], industrial manufacturing, etc. Therefore, MV is a specific application domain of the much border research topic, that is computer vision.

Collaborative robotics [52] explores the current encouraging advancement in industrial robotics that have adaptive perception. In [52] machine vision techniques, E.g., point cloud processing, 3D stereo reconstruction for CAD model shape matching under arbitrary orientations and illumination conditions are evaluated, E.g., a car door assembly or picking correct screw heads, etc.

Machine vision is increasingly being applied to aid modern vehicles towards higher awareness of the situation on the road and reduce the chances of human driver error. For example, real-time pedestrian detection in [53], presents the use of support vector machine algorithm (SVM) to train image classifiers based on unique HAAR like features in order to detect certain basic elements of a pedestrian shape in real-time. They used Ada-Boost algorithm that can use a large number of basic classifiers to combine into a strong model.

Video Content Analysis

As stated previously, Computer Vision (CV) is at the core of the proposed ViDAQ and EYE-on-HMI (*inverse oculism*) class of human performance monitoring systems, to visually acquire HMI state in real-time and perform video content analysis (VCA) [54].

VCA generally [55, 56] involves computational steps such as: (1) image pre-processing or filtering to remove optical noise; (2) segmentation for image size reduction to a format that's more relevant and compact for computation; (3) feature extraction [57]; and (4) supervised training of machine learning classifier models (E.g. convolutional neural networks [58]) to detect target changes in panel indication states and to accurately interpret temporal events.

Application of VCA to proposed *EYE-on-HMI* framework stands to extend the next generation of monitoring systems for control room applications, E.g., reading instruments mounted in control rooms or even remote field mounted gauges. The feasibility and reliability of this use case is evident from the current research trend in applying CV for the traffic light, road sign, pedestrian recognition [59, 60] and driver fatigue detection [61]. Moreover, use of CV based devices across the automotive industry, for example, Toyota's lane departure warning system (LDA) [62] and Volvo, Nissan's driver fatigue alert systems [41] etc is also an indication of expanding CV's feasibility and reliability in public safety domain by making vehicles safer on roads.

Automatic Instrument Reading

Acquiring instrument measurement reading using image processing was first explored by previous works as the need to automate the reading of water and electric utility meters. For example AMRS (Automatic Meter reading System) [63, 64] has been shown by previous works such as [65, 66, 67] and few patents [68, 69] as well. The majority of such works can broadly be categorized in two bins: (1) research works that attempt to do numerical digit recognition to acquire meter values directly; (2) others that attempt to interpret the analog dial needle position to read the meter value indirectly. Both categories of approaches explore unique image processing challenges and present innovative use of image processing algorithms.

Dial Reading - Digit Recognition

Works such as [66, 70] used a 3-layer artificial neural network (ANN), with an input layer of 315 nodes and 10 nodes in the hidden and output layer, respectively. The ANN is trained off-line in Matlab using the back-propagation algorithm to recognize digits in the image training set. Image training set includes cropped binary images for each category of numerical digits $[0-9]$ with resolution $21 \times 15 = 315$ pixels (each pixel maps to an ANN node in the input

layer). While it would be possible to use currently existing OCR (optical character recognition) algorithms to achieve the same functionality, their notable contribution was in demonstrating the use of an embedded DSP (digital signal processing) system that uses the pre-trained ANN classifier to do in-situ processing and classification of images being captured via a camera. The extracted value is then relayed to a remote concentrator in real-time wirelessly, thereby making the system scalable and readily deployable.

In contrast, [71] relied on dual-threshold (whole and partial) constants to generate binary images from grayscale that offer better contrast results. Binary image conversion and segmentation is critical to visual data acquisition to reduce false positives and improve data acquisition precision. They separate individual digit by locating the digit boundaries using a vertical projection histogram (i.e. counting the number of white pixels along the vertical lines). Digit recognition involves using a heuristic method based on locating closed curve shapes on a grid, E.g. if a closed curve shape is on the top quadrant, the shape is likely to be digits categorized as $9, 8$, etc.

Dial Reading - Position Detection

The latter category of works [72, 67, 63, 73, 74] attempt to interpret dial positions. These are of particular interest to the ViDAQ framework. The general motivation of previous research works is to develop an advanced metering infrastructure (AMI) [72] by using cameras to read legacy electromechanical power utility meter dials and transmitting this data for internet cloud-based tracking of power consumption per household.

Image data extraction presented in [72] involves using a static threshold constant, which is experimentally derived to generate a suitable contrast quality binary image for processing and segmentation. The pointer extraction algorithm involves the use of morphological operators (dilation and erosion) to fill any holes or deficiencies in the dial pointer area. Followed by, superimposition of a fixed size box on the cropped binary images of each sub-dials. The intersection of the box and dial shape edges allows locating the center of the dial and the angle the dial makes with respect to the square edges, which is finally used to extract the meter reading. This approach, while claimed to work well, is quite specific to the type of meter model.

In contrast, the methodology adopted in [67] is towards a more general approach, where SIFT (Scale-Invariant Feature Transform) is used. SIFT robustly matches features of a reference image of the sub-dials to the actual sub-dials in the input image irrespective of orientation, resolution, illumination, and other substantial range of an affine distortion that may be present in the source. As a result, image segmentation is improved as target sub-dials region-of-interest (ROI) can easily be validated and cropped out for downstream processing.

Binary image conversion in [67] is done by applying Otsu's Thresholding [75] method. This method dynamically selects an optimal threshold based on a discriminant criterion to improve contrast quality, which also improves the effectiveness of morphological operators. Such operators include dilation (emphasizing ROI shape boundaries) and erosion (de-emphasizing of ROI shape boundaries) to remove any pixelation noise and edge discontinuities. It also emphasizes the separation of neighboring shape boundaries in ROI.

Dial tip resolution technique used by [67] involves using a structuring element (kernel) to perform a morphological opening operation - this works by applying erosion followed by dilation using a common structuring kernel. This step has the effect of slightly eroding the dial shape boundaries to make them merge toward larger dial lobe while allowing thin boundaries, connecting unwanted neighboring shapes to un-merge. Since, thresholding operation can create unwanted neighboring shape boundaries to merge and distort the original dial shape and dial tip. Next, the centroid of the dial ROI shape is calculated using first and second-order sequence of image moments $(\overline{x}, \overline{y}) = (M_{10}/M_{00}, M_{01}/M_{00})$. Once the center is located, the tip of the dial is found by finding the pixel in dial ROI that maximizes the Euclidean distance from the centroid. Finally, the angle is computed using the coordinates of the centroid and tip point using trigonometric functions.

The results presented in [67] includes a comparison of two approaches. One approach uses a combination of HARRIS (corner detection) algorithm for detecting key points followed by, using BRIEF (Binary Robust Independent Elementary Features) algorithm for extracting the descriptors. Another approach uses using just the SIFT (Scale-invariant feature transform) algorithm that accomplishes both the former tasks in unison. They [67] showed, SIFT runs faster in detecting and segmenting the dial ROIs overall compared to HARRIS/BRIEF algorithms. In addition, the distance at which the dial image is captured also affects the run time of HARRIS/BRIEF more than SIFT algorithm.

Merit of the techniques presented in [67, 76, 73], addresses the image orientation issue and robust dial ROI selection using advanced image processing algorithms such as SIFT, BRIEF [67], ORB [76](Oriented FAST and rotated BRIEF) as shown in [73].

Another technique of a dynamic sliding window for dial needle detection and image segmentation is presented by [63]. Post thresholding (conversion to a binary image), a (rectangular) window of fixed width is used to systematically search for the location of the dial needle, which terminates once the dial arm is inside this window. They claim this may not be suitable for real-time processing and propose the Angular Detection Algorithm (ADA) as the alternative to the sliding window. ADA detects the dials in the following manner: it tries to find vertical growing lines in the image (details of which are not adequately covered in [63]) and selects the longest one, as that would correspond to the dial needle of interest. Upon

successfully locating the dial needle, the image is segmented with ROI containing only the dial needle. The needle angle is then computed using the start and endpoints of the needle. Even though ADA is an improvement over the former approach, the vertical growing lines use case applies to only dials where the needle pivot is at the bottom. In cases where the needle has a full 360 degree of motion on the meter face, the notion of vertical growth lines may not be adequate for image segmentation.

Works such as Automated Dial Reading (ADR) in *The Flight Guardian* [35] and computer vision-based approach for reading analog multimeters [77] also demonstrate dial reading systems for practical applications.

In ADR, two analog instruments (airspeed and engine RPM) dial gauges, found in the cockpit of a smaller aircraft (E.g., Cessna), are read using independent camera video streams. Dial needle position detection uses a 1-D (dimension) convolution operation with a 1-D unity kernel vector, which is convolved with various linear horizontal or vertical sliced sub-images (row or column pixel value vector) of the ROI. The binary image ROI that is selected for image processing is cropped from a smaller circular area around the dial gauge center. This effectively crops-out the unnecessary artifacts (e.g., digits, scale markings, reflections, etc.) on the gauge face leaving only the white base of the needle with a black background. The output of each convolution index, corresponding to the sub-image of the dial gauge ROI that yields the maximum value, is then used to estimate the needle position. Image orientation requires that cameras remain fixed and focused on dial center such that it appears on the pre-selected center coordinate of the image. Thus, allowing the ADR algorithm to determine the dial center correctly.

In [77] needle position on the analog multimeter is found by using the Edge-based geometric matching (EGM) and Pyramidal Gradient Matching (PGM) which currently is available in the NI (National Instruments) vision software suite for machine vision applications. EGM and PGM have been widely used as image feature extraction algorithms utilized for template matching. Unlike SIFT [67] EGM [78] uses edge gradient-based template matching. While PGM [79] uses Gaussian pyramids to obtain down-sampled multi-spatial resolution maps of the template image for matching using other scale and rotation invariant features. In [77] these template matching modules aid in extracting the horizontal alignment parameters of the meter scale as captured from the camera image. These alignment values are then used as the input to calculate the dial center and dial needle angle, which in turn helps to determine the dial reading.

Even though the above approaches for dial needle tip resolution are diverse and well demonstrated for single-dial per ROI, it does not address meters with multiple dials in the same ROI, such as in a clock. ViDAQ directly addresses this gap in the literature by develop-

ing a multi-dial detection algorithm using a convex hull as described later in Section 4.1.2.

Lastly, another widely used technique relies on Hough Transform (HT) [80] to detect straight lines in binary images, as demonstrated in [74]. The standard Hough Transform algorithm detects and interpolates straight edges in binary images. It does so by transforming image coordinate space into polar coordinate space. A line in cartesian space $y = mx + b$, can be transformed to polar coordinate space by $\rho = xcos\theta + ysin\theta$, where ρ is the perpendicular distance between the line and origin and θ is the angle between x-axis and ρ vector). Computational benefits of this transformation is that it limits the values of θ to $0 - 2\pi$ radians, thereby making it possible to represent any line in the ROI with tuple (ρ, θ). The algorithm checks for each white $(value = 255)$ pixel/point in the ROI, and computes and records tuple (ρ, θ) of the line that passes through the point. It continues to tally votes for each tuple (ρ, θ) in an accumulator matrix. Once all white points in ROI are visited, it checks which tuple (ρ, θ) has the highest votes, which corresponds to the straight line segment where the majority of points line on. Moreover, [74] also demonstrated an improvement of the HT by limiting ρ to only those ranges the dial needle is expected to travel.

In the automatic calibration system for analog dial meters [81], HT is used to determine both the dial center and dial needle angle. The main idea here is image thresholding is achieved by subtracting two grayscale images that are obtained after applying an edge detection algorithm. The resultant image ideally shall leave only the two segments corresponding to the needles in two locations and also helps eliminate any static noise. HT is then applied to the difference image to find the intersection points of the sinusoids in $\theta)$ space, which corresponds to (ρ, θ) values of the two-needle line segments, which are then extended to determine the dial center.

The later discussed shows a promising approach and can be extended to reading multi-dial meters, as shown by the ViDAQ framework components implemented in this thesis (Sec. 3.2).

2.4 HITL-Error Detection via Time-Series Modeling

The secondary goal of this research (Sec. 1.5) is to consider the modeling of HMI state transitions in order to build HMI state, forecast models. These models are considered to aid in the **detection** of HITL errors, by way of flagging deviation or anomalies between current HMI state patterns and the expected or learned HMI state patterns (generated from these forecast models).

Nuclear Power Plant (NPP) and transportation systems industry are currently challenged with the niche area of real-time non-intrusive monitoring of operator situational awareness, in order to predict adverse operator performance trends. This study initially started with

treating HMI states sequence as time series (TS) data to realize a non-intrusive means of estimating HITL errors based on HMI state pattern upsets. Numerous related works [82] have also addressed time-series forecast employing various techniques ranging from classic regression models (ARIMA) [83] to artificial neural networks: LSTM (Long Short Term Memory networks), CNN (Convolutional Neural Networks), SVMs (support vector machines), etc.

Reinforcement learning using human-in-the-loop machine learning is a burgeoning field that combines human agent inputs to aid in accelerating the learning and iterative improvement of the prediction accuracy of deep learning models. For example, in the survey [84] active learning is introduced to use human inputs to prune and annotate, as feedback inputs, the predictions generated by the deep trained model. Such as in case of automatic medical image analysis. These models evolve their predictive intelligence with human-in-the-loop agents to constantly correct their errors. Interactive machine learning [85] also proposes using human intervention in place of conventional automatic machine learning as means of solving hard optimization problems such as demonstrated on case studies of using Ant colony optimization algorithms. While in [86] a new machine learning workflow is proposed that uses human-in-the-loop inputs to intelligently track changes over time and rapid interventions to accelerate machine learning task.

The above works suggest capturing human-in-the-loop patterns is useful for incorporating human intuition, which in this research is extended to learn and infer something about the expected operator situational awareness towards detection of HITL error precursors.

As an initial approach and since LSTMs and CNN are the current state-of-the-art for time-series modeling, only relevant works for time-series pattern prediction using recurrent and convolutional neural networks (RNN) are reviewed here.

In this thesis, three distinct modeling approaches are analyzed in detail. While considering HMI data model as a time-series random process: (1) ARIMA (Auto-Regressive Integrated Moving Average) (2) RNN LSTMs and CNN are evaluated. While considering HMI data model as a language generator process: (3) Natural Language Processing (NLP) model is evaluated.

Background and related works in above areas are discussed below.

2.4.1 ARIMA Models

ARIMA[3] (Auto-Regressive Integrated Moving Average) or *Box-Jenkins* method was first introduced in 1970s [87] as a statistical means to capture standard temporal structures in time series data such as seasonal trends, the contribution of random noise, etc. It has seen numer-

[3]For brevity, this section does not review the basics of Auto-Regressive Moving Average models but only discusses the relevant effectiveness and application use cases. As such the required technical details have been covered in later chapters: Proposed Solution and Implementation Details.

ous applications in various areas for forecasting processes such as stock prices [88], electricity market [89], computer hardware resource [90, 91], natural phenomena [92], etc, that yield time series data. In [91] ARIMA and Long Short-Term Memory (LSTM) recurrent network for forecasting performance of CPU usage of server machines in Data Centers is compared. They showed that ARIMA does not quite model non-linear temporal relationship where the CPU data is unstable and volatile, requiring high effort in data transformation to build a functional model. A hybrid approach combining ARIMA and wavelet decomposition approach is proposed in [90], to accurately predict the number of arriving tasks to a cloud data center at the next time interval. The statistical test most commonly used to test stationariness of a TS is the Augmented Dickey-Fuller Test (ADF) [93], which has also been utilized while evaluating the stationary and unit-roots for the custom generated synthetic data. Other useful statistical hypothesis test that was utilized was the *granger-causality* [94] test, which checks whether exogenous predictor variables satisfy unidirectional causality to ensure the selected feature in the multivariate time series is useful in forecasting the target feature being modeled.

ARIMA time-series data forecast models are utilized and evaluated in this thesis to ascertain its feasibility in predicting human-machine interface (HMI) state transitions for n-steps ahead window HMI states generally include changes in its visually displayed information brought about due to both dynamic process parameters and user actions (E.g., a driver operating a vehicle as seen from the perspective of information on dashboard and steering wheel position). This approach has wide applications in industrial controls, such as nuclear power plant control rooms, and the transportation industry, such as aircraft cockpits, etc.

But in this thesis is the first known attempt to authors knowledge to use ARIMA for model HMI state sequences in order to detect Human-in-the-Loop error trend precursors.

Although TS data forecast modeling is a vast area of research employing varied techniques ranging from linear statistical models to non-linear neural networks, current *data analytics* best practices recommend it is prudent to start with a basic regression modeling technique before using more complex models, to get a good understanding of the nature of the dataset to begin modeling. Therefore, ARIMA models were evaluated in this research initiative to establish a baseline performance.

2.4.2 RNN and CNN Models

RNNs[4] differ from other classic feed-forward neural network by relying on feed-back of the hidden layer states, accumulated from previous $(t - n)$ time-steps, along with new inputs.

[4]For brevity, this section does not review the basics of the neural networks, recurrent neural, and convolutional neural networks, but only discusses their relevant effectiveness and application use cases. As such, the required technical details have been covered in later chapters: Proposed Solution and Implementation Details.

Hence, RNN are apt in capturing temporal structures in time-series data better. LSTMs (long short term memory) further utilize memory cells and forget gates to overcome the exploding and vanishing gradient challenges with RNNs. Therefore, LSTMs are more suitable at learning longer temporal dependencies (casualties) in complex sequences than regular RNNs. LSTMs have shown to be versatile for both pattern regression and classification type problems as well.

Time-series forecasting is relevant to several domains such as predicting traffic flow [95], energy demand [96], stock prices [97, 98], etc. The design challenge predominantly is with model architecture selection and feature engineering. Model architecture pertains to determining a suitable ensemble of basic machine learning models such as LSTM, in various arrangements (vanilla, encoder-decoder, stacked, bidirectional, etc) to do one-to-one, one-to-many, many-to-many mode of *n-step* ahead sample forecasts.

Whereas, feature engineering uses domain-specific knowledge to dimensionally reduce the learning space and pre-select the features of interest for the model to learn signal patterns. Besides, addressing stationary vs. non-stationary real-world time-series data has its own challenges and proposed approaches. Current feature engineering techniques include signal decomposition using various signal processing [97, 99] tools (E.g. Fourier, wavelet, chirplet transforms, etc). Hybrid models may use LSTMs in conjunction with other models such as Convolutional neural networks (CNN) as the encoder layer for LSTMs to exploit any spatial correlations. This has shown significant success in deep-learning applications, where both temporal and spacial features such as in images and video frames is required.

2.5 HITL-Error Detection via NLP Modeling

Nuclear Power Plant (NPP), transportation systems and other manufacturing industries alike are currently facing or eminently going to face information explosion. With the large volume of diverse (featureful) real-time sensor data being generated as industrial solutions transition to Industry 4.0 [5]. This massive amount of data is likely to create information overload and may lead to more tragic accidents. Therefore, a more versatile and scalable deep pattern learning techniques must be evaluated to utilize this data effectively.

Therefore, an alternative approach to HITL error detection via Time-Series modeling (Sec 2.4), is to consider the modeling of HMI as Discrete Event System automata to which natural language processing (NLP) techniques may be applied for next state prediction. In this approach, we transition the underlying problem of operator situational awareness mon-

[5]The fourth industrial revolution encompasses areas which are not normally classified as an industry, such as smart cities, robotic manufacturing network, driverless vehicles [100, 101]

itoring and human-in-the-loop error detection from a fundamentally time-series forecast to a natural language processing (language translation) problem, without loss of any generality. The core idea is to transform the HMI state forecast problem as a language translation. That is to translate from the source language of the previous sequence of HMI states to a target language of the expected future sequence of HMI states. The **detection** of HITL errors will then be flagged as a deviation or anomaly between current HMI state patterns and the expected or learned HMI state patterns generated from these language forecast models. The following sub-sections cover state-of-the-art concepts in deep machine aided NLP to highlight its strengths in addressing the scalability concerns.

Natural Language Processing (NLP) using Machine Translation (MT) is a multidisciplinary and an active area of research under computational linguistics having several applications spanning engineering and business domains such as gene mapping, video/image analysis (image captioning), sentiment analysis, conversational agents (chat bots), market intelligence surveys, etc. NLP MT algorithms generally depend on key supportive technologies: word embedding, attention mechanism, and deep learning models [102].

2.5.1 Word Embedding

A leap forward from traditional statistical NLP tasks to the current use of deep learning models was made possible by efficient word embedding techniques. Overcoming the *curse of dimensionality* in NLP while learning joint probability models for modeling complex language models was essential. Word embedding map each word or token (represented by a unique ID) from a dictionary (bag of words) to a lower-dimensional vector space. The most basic embedding type is one-hot encoding, where each token is represented by a multi-dimensional sparse zero column vector containing mostly 0's. Its embedding size is equal to the number of tokens in the dictionary and contains only 1 at the index corresponding to the token ID (often yields long word embedding vectors). However, other word embedding architectures such as *CBOW*(continuous bag of words) and *skip-gram* have been used by *word2vec* [103], *GloVe*(Global Vectors for word representation) [104] algorithms, etc. to learn to generate lower-dimensional denser vector representation of words using shallow feed-forward neural network. These embeddings take advantage of distributional semantic [105, 106, 107] similarity between the words in a given set of corpus space to produce a vector space, where words with similar meanings are located in closer proximity to yield a lower cosine (dot product) similarity score.

For example, *CBOW* uses neighboring words, appearing in a context window, to predict the target word, without capturing any dependencies on the order of appearance of the con-

text words (treated as a bag of words). Whereas *skip-gram* does the opposite by predicting the neighboring words in the context window for a given word. *word2vec* [103] model can utilize either *CBOW* and *skip-gram*. On the other hand, *GloVe* [104] does not only depend on the local context window of neighboring words (unlike *COBW*) but also incorporates a global context using word co-occurrence statistics. Currently, pre-trained word embedding is available for over 150 languages (English, German, French, Hindi, etc.) that were learned using unsupervised training of neural networks.

In addition to distributional semantics in word embeddings, further enrichment is achieved by incorporating additional context-specific information, which has been demonstrated by other works [108, 109] to be useful in further improving NLP predictions. Reference [108] shows the use of positional encoding in conjunction with the input embedding layer while describing their *Transformer* model. The intent was to inject a periodic seasonality trend as positional information of each input token relative to other in a given sequence (sentence), by taking advantage of sum of angles identities for the trigonometric function (*sine* and *cosine*) (closure under linear transformation) which allows the model to better identify tokens by their relative position with any fixed offset.

The limitations of the word-level embeddings (E.g. word2Vec) is that these do not quite produce vector representations of special word sequences or phrases which mean something more than just the words put together (*polysemy*) E.g. "brown out" , "phase in", etc. Implications of which is more evident in natural language translation tasks but also affects modeling more complex context dependencies in system state patterns, (E.g. system state sequence $S1 \rightarrow S2$ could invoke different user response in the presence of context state $S3$)

In the current scope of this work, due to a relatively smaller vocabulary size of HMI state space as compared to that encountered in language translation tasks, existing word embedding was not utilized in order to keep the initial model design more straightforward. Nevertheless, distributional encoding shall be a vital consideration when EYE-on-HMI system needs to scale up for more complex industrial HMI designs. Moreover, positional encoding in the proposed model is implicit by constructing the training data set using a shifted sliding window (described in detail in below Section 5.3.2).

2.5.2 Fine Grained Embedding

The word-level embedding vectors in NLP are useful to capture semantics and syntax - meanings based on the context and arrangement of words in sentences. However, fine-grained embedding can further capture morphological meaning [110] - meaning in the structure of words (E.g. "bad" vs "badly"). Which implies for HMI systems, each state can further be used

to find relationship with other exogenous (co-related) parameters to incorporate their context at state level embedding favorably.

2.5.3 Convolutional Models

Convolutional neural networks have shown to be an effective tool in extracting features for NLP tasks in sentiment analysis, chatbots, etc. The very first CNN NLP application appeared in [111], which presented a unified model that could be jointly trained to do a host of NLP tasks such as part-of-speech tags, named entity recognition (NER) and, predict if the input sentence makes grammatical and semantic sense. The fundamental approach presented in [111] was entire sentence sequence was tokenized and converted to an embedding matrix in which each row corresponds to a word embedding vectors that were generated using a look-up table. Several convolutional filters were convolved with each sentence matrix (like an image) to extract feature maps. Several *max-polling* operations then further reduce the dimensionality of each feature maps to obtain fixed-length column vectors, which are finally concatenated together into a feature vector. Finally, a densely connected output neural layer with output softmax normalization is trained to produce the desired class labels or sequence tokens.

Unlike the RNNs, classic CNNs do not learn sequential order or long-distance dependencies [112] between sequences well enough, which is essential for many NLP tasks in conversational agents, sentiment analysis, machine translation, etc. However, CNNs have been combined with time-delayed neural networks (TDNN) [113] with window sampling. This extends the temporal scope of CNNs to capture longer distance contexts. Additionally, the dynamic CNN (DCNN) [114], which uses an approach where the *k-max* pooling filters dynamically spans variable sentence length ranges has shown to be adept for hierarchically learning to extract low-level lexical features into high-level semantic concepts for generating text summaries.

2.5.4 Recurrent Models

Recurrent neural works (RNN) have been the workhorse of sequence prediction tasks, as they are fundamentally sequential in nature. Each RNN cell feeds back what was predicted previously in conjunction with new input token, which conditions the prediction of the next token. Tokens are fed and predicted one at a time (unlike in CNN). Particular types of RNN, such as the long-short term memory (LSTM), Gated Recurrent Unit (GRU) and Residual Networks (ResNet) have been designed to overcome the problem of vanishing gradients encountered with learning longer sequences by incorporating coefficient control gates (input, forget and

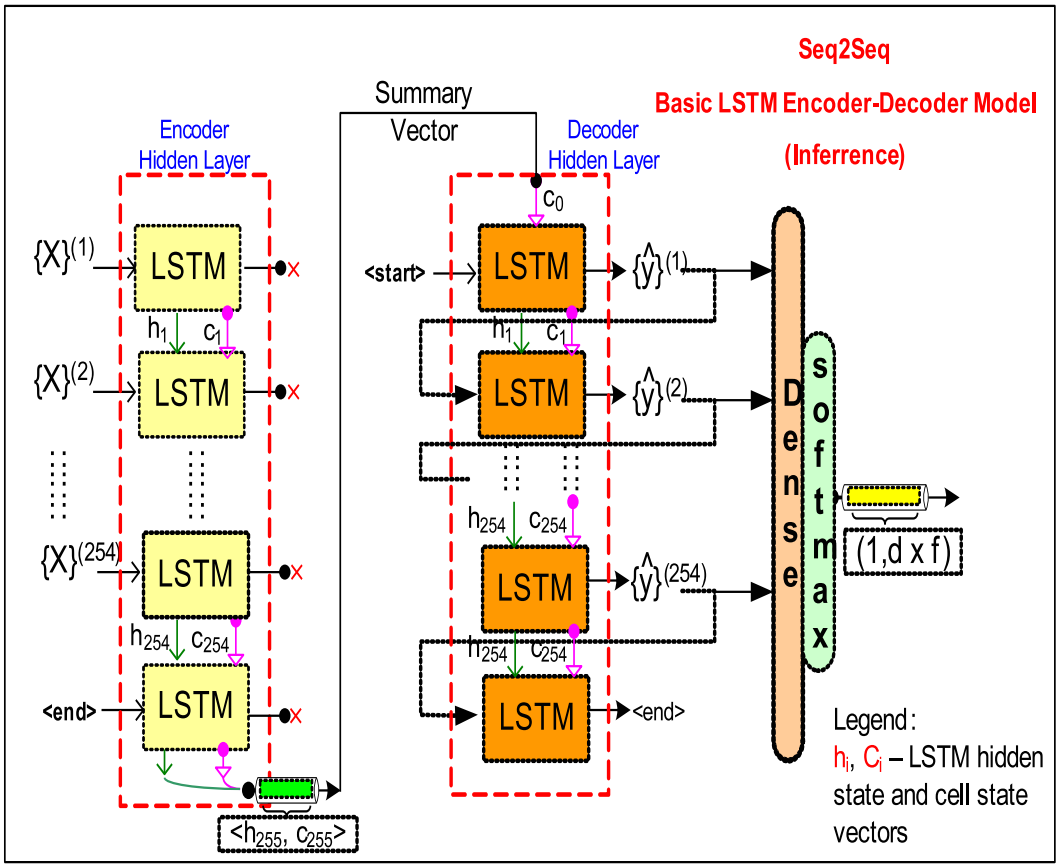


Figure 2.5: Basic LSTM (RNN) based encoder-decoder model for *Seq2Seq* applications. Sequence of input tokens, $X^{(K)}$ are fed to the encoder LSTM stack, which pass intermediate hidden (h_k) and cell state (c_k) vectors in between. The last layer hidden state vector (h, c) is the *summary* vector that captures the essence of entire input sequence which initializes the first decoder LSTM cell including a *<start>* token as input. The output or each LSTM cell is fed back as input to subsequent LSTM cells to generate target tokens ($\hat{y}^{(N)}$) one time step at a time until *<end>* token is generated (during inference).

output) that act as memories to hold previous states when required. RNNs have also been extensively utilized for NLP tasks such as in image captioning, machine translation, text summarization etc. LSTMs have shown to be suitable [115] for certain NLP modeling tasks such as machine translation, but they are not significantly any better than CNNs in general.

A prevalent RNN or LSTM-based model that is generally used for NLP machine translation or sequence to sequence prediction tasks, is the *encoder-decoder* architecture as shown in Fig. 2.5. On a basic level, the *encoder* summarizes a given input sequence of word vectors as a final hidden state vector, which the *decoder* uses as the context to generate output word vectors. Both *encoder* and *decoder* are trained together using *teacher forcing* method where, the *encoder* is given the input sequence (x_0,...,x_{t-1},x_t) alongside the *decoder*. The *decoder* is exposed to the output ground truth target sequence of tokens arranged as two separate

sequences offset by one time-step with respect to each other by inserting special *start* and *end* sequence markers - E.g. *decoder* input: [<$start$>, y_0,...,y_{t-1}] and *decoder* output: [y_1,...,y_t, <end>]. Where as during inference, each previously predicted token is fed directly forward to generate next predicted tokens.

The two main issues with *encoder-decoder* model (Fig. 2.5) is the difference between training and inference phases, that causes *exposure* bias which limits the ability of the encoder-decoder model to be resilient to previous incorrectly predicted tokens. Secondly, there is a bottleneck in the information transfer from *encoder* to *decoder*, since only the one hidden vector from the encoder is used to capture the context of the entire input sequence which rather prevents *decoder* to selectively focus on parts of the input sequence that carry relatively more influence in predicting the next target token.

2.5.5 Evolution of Attention Mechanism

Attention mechanism (AM) in Fig. 2.6 has been used to over come the information bottleneck in classic *encoder-decoder* models for NLP *seq2seq* applications. AM has significantly improved NLP performance for *alignment* and *translation* tasks: *alignment* refers to learning which parts of the input sequence are relevant to each output target token and *translation* is learning to use relevant contextual information between source tokens to select the appropriate output token. Mechanisms of attention allows the *decoder* to *learn* through training to selectively *attend* or focus on various positions in the input sequence based on the tokens predicted so far [116] in order to accurately predict the next token in target language.

Attention Mechanism

$$\begin{aligned} e_{t,i} &= f(Q_t, K_i) \text{ Similarity Score} \\ a_i &= softmax(e_{t,i}) \text{ Attn. Weight Vec.} \\ Attention(Q_t, K, V) = c_t &= \sum_i a_i V_i \text{ Attn. Context Vec.} \end{aligned} \tag{2.1}$$

Global vs. Local AM

Attention mechanism (Fig. 2.6) was initially introduced in 2014 as a technique mimicking selective concentration on parts of a larger scene in human vision to derive efficiencies in computer vision [117] algorithm using RNN based image classification. However, the first application of AM in machine translation tasks didn't come about until early 2015 [116, 118], where *global* (Fig. 2.7a) and *local* (Fig. 2.7b) AM types were first introduced. The *global* AM approach uses contribution from all source tokens (*encoder* hidden states) in jointly learning

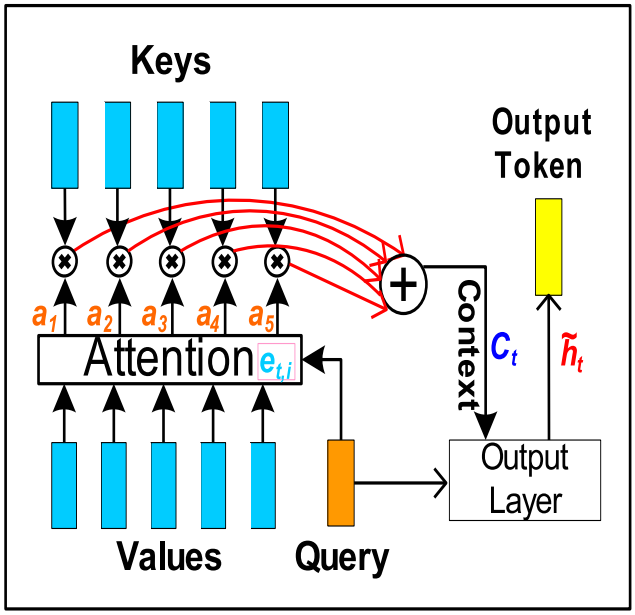
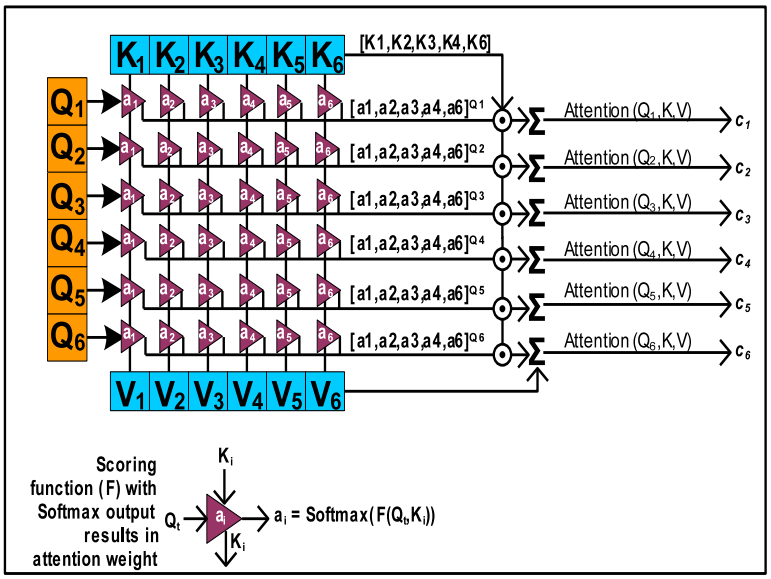


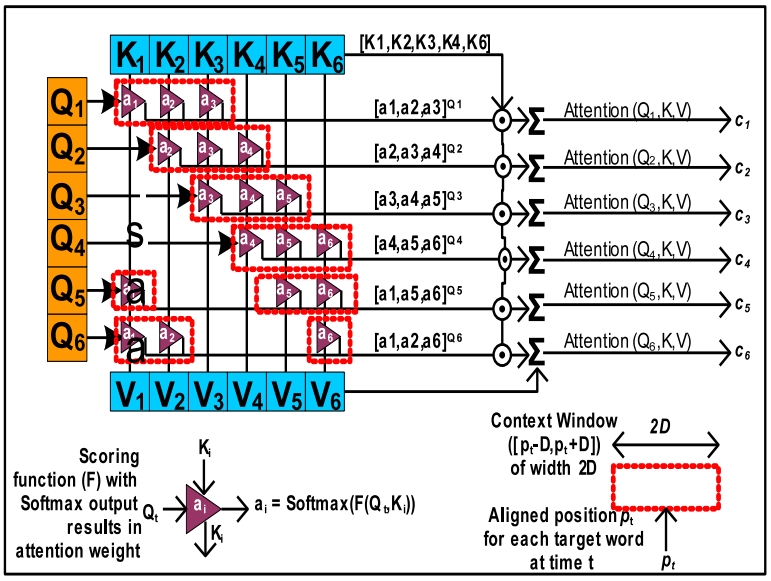
Figure 2.6: Attention Mechanism (conceptual). Allows the decoder to learn how much influence each **Key-Value** (input tokens) has on a given **Query** (target token) dependence which allows accurately predicting next output token.

to align the *decoder* (hidden state) to produce an output token. On the other hand, the *local* AM uses limited source tokens in order to reduce training computational load.

Besides *global* AM designs, the other prevalent AM designs fall under those using *local* [118] approach, which was actually developed by taking into account the comparative trade-offs between *hard* (similar to *local*) and *soft* (similar to *global*) attention types used in computer vision area as showed by [119] for automatic image captioning tasks. This technique was also applied to use a subset of source tokens at a time to generate attention weights. For instance, two *local* attention mechanisms: monotonic (**local-m**) and predictive (**local-p**) were introduced in [118] which uses a smaller context window of an empirically chosen fixed width D, for each target token at time step t (in comparison to *global* AM, the context window is of the size spanning entire *encoder* input sequence). The context window ($[p_t - D, p_t + D]$) is centred at an alignment position p_t generated by the *local* AM model, thereby only utilizing the encoder hidden states within the context window to calculate the context weights. The difference between monotonic (**local-m**) and predictive (**local-p**) *local* AM is how the alignment position p_t is generated: **local-m** - p_t is same as target token index t, **predictive-m** - p_t is predicted by a model ($S.sigmoid(v_p^T tanh(W_p h_t))$ which is trained along with entire *seq2seq* model.



(a) Global(soft) Attention



(b) Local (hard) Attention

Figure 2.7: (a) Generates variable length (same as input sequence) context vectors $[c_1, c_2.., c_6]$ which capture attention weights ($[a_1, a_2.., a_6]$) from all input tokens (**K**); (b) Generates a fixed length context vector $[c_1, c_2.., c_6]$ which capture fixed number of attention weights ($[a_1, a_2, a_3]$, etc) from select input tokens selected by a context window which is centred at token index position p_t (determined by the model separately).

Similarity Scoring Functions

Underlying attention mechanism is a similarity scoring functions as listed in (2.2). A generalized representation (Fig. 2.6) of AM as a function (2.1) of *Query*(**Q**), *Key*(**K**) and *Value*(**V**)

tuples: *Attention(Q,K,V)* terminlogy was introduced by [108] (normally for *seq2seq* models *Key* is taken to be same as *Value*: $K = V$ are just some internal representations of input language tokens). The similarity scoring function (*f*(Q, K_i)) (2.2) can differ in implementation, but in general compute the similarity between a *decoder* hidden state at index t (*Query* Q_t) and set of $(1, 2, 4..i)$ *encoder* hidden states (K_i or V_i), which in turn is used to obtain an alignment score vector (e_t) associated with each Q_t. A *softmax* operation is applied to each score vector (e_t) (2.1) in order to obtain a attention weight vector (a_t) (2.1). The (a_t) (2.1) is the probability distribution representing the degree of similarity or relevance each *encoder* source token (K_i) or position has to a given *decoder* target output Q_t token. This is followed by generating a weighted average of *encoder* K_i's, with attention weight vector a_t to generate a context vector (c_t) (2.1). Finally, a hidden attention vector ($\tilde{h_t}$) (2.1) is generated from the context vector to help generate the next output token. In [116] the *global* alignment used additive (perceptron with $\tanh$ as output non-linearity) type funtion which learns the attention weight matrices (W_a, U_a), while [118] evaluated other similarity functions such as, *Concat.*, *Location* based, *Dot* product, *General*, *Content-based* by [120] and *Scaled dot* product first being proposed by [108].

Attention Similarity Scoring Functions

$$f(Q_t, K_i) = \begin{cases} \cos(Q_t, K_i), & \text{Content-based} \\ softmax(W_a Q_t), & \text{Location-based} \\ v_a^\top \tanh(W_a[Q_t, K_i]), & \text{Concat.} \\ v_a^\top \tanh(W_a Q_t + U_a K_i), & \text{Additive} \\ Q_t^\top W_a K_i, & \text{General} \\ Q_t^\top K_i, & \text{Dot} \\ \frac{Q_t^\top K_i}{\sqrt{n}}, & \text{Scaled Dot} \end{cases} \quad Score(Q, K) = \tag{2.2}$$

Where W_a, U_a are tranable weight matrices,

n dimensionality of source hidden stateK_i

Other variants of AM that combine elements of both *hard* and *soft* attention mechanism have also been proposed. Such as [121] introduced a reinforced self-attention (ReSA) model that parallelizes hard attention using reinforced sequence sampling, which does not use any RNNs, such that it captures contextual dependencies while trimming a longer input sequence into a shorter one for soft attention to process.

Self-Attention

Self-attention or intra-attention is a specific case of AM which finds relationships between positions of a single sequence to compute a compressed feature representation of the same sequence. In essence self-attention is *Self-Attention(Q,K,V)* where, $Q = K = V$ and means attention is applied to each token of the sequence [122] with other tokens of the same sequence. Self-attention has been used in image description and text sentiment analysis applicaitons [122]. However, self-attention layers can also be stacked together in both *encoder-decoder* models to aid in machine translation tasks as demonstrated by the *transformer* [108] model which is further discussed below. Self-attention is also unique in resulting only a constant ($O(1)$) longest path dependency between source tokens for any length of sequence, compared to other AM (E.g. linear $O(n)$ for RNN *seq2seq* models). This effectively allows the model to focus on the semantic dependency relationships between the tokens rather than based on encoding distances between them. Resulting in the learning of the semantic structure of the sequence being processed. Furthermore, layer complexity is favourable for self-attention when using scaled-dot product similarity function is $O(n^2, d)$, where n is sequence length, d is encoding dimensionality and normally $d > n$, compared to that for RNN ($O(n, d^2)$) *seq2seq* models [108].

Single vs. Multi-Headed

Single head attention mechanism is the basic case of AM, where only one hidden state representation from *decoder* and *encoder* (*Q,K,V*) may be utilized in the attention layer.

In contrast, a multi-headed AM uses multiple parallel iteration of *attention* calculations using different values of *Q,K,V* resulting from parametrized linear transformations (LT) learned through training. Each head corresponds to using a different LT which lets the model to capture different features from each LT space.

Multi-headed attention was first demonstrated as a key component of the transformer [108] model for *seq2seq* applications. Work such as [123] has performed systematic analysis of the role and contribution of using multi-heads in the *encoder* of *transformer* model (E.g. seen as h heads in Fiq. 2.8). Their observation characterized heads as acting as positional (heads attending to a neighboring token), syntactic (heads attending to syntactic dependencies between tokens), and attention to rare tokens (heads pointing to the least frequent tokens in the sentence). Furthermore, [123] also identified the optimal number of attention heads required in the *encoder* by experimentally pruning the model incrementally by fine-tuning a trained model with a regularizer objective, thus showing a majority of the self-attention heads in *encoder* can be removed without significantly affecting performance. Other works such as

[124] showed the use of multi-headed self-attention to ascertain the authenticity of a news content based on several news feeds.

Transformer Architecture

Even though the *transformer* NLP *seq2seq* model has been analyzed in numerous works [125, 123] previously, it is worth reviewing the architectural highlights in brevity due to its contribution of establishing a new dominant modeling paradigm in machine translation applications since its introduction in 2017. The model (Fig. 2.8) demonstrated a feasible alternative to existing RNN and CNN based *seq2seq* models to models in which "Attention is all you need", where just a few stacked multi-headed self-attention layers with fully connected dense layers was all that was required to achieve state-of-the-art performance in NMT [126, 125].

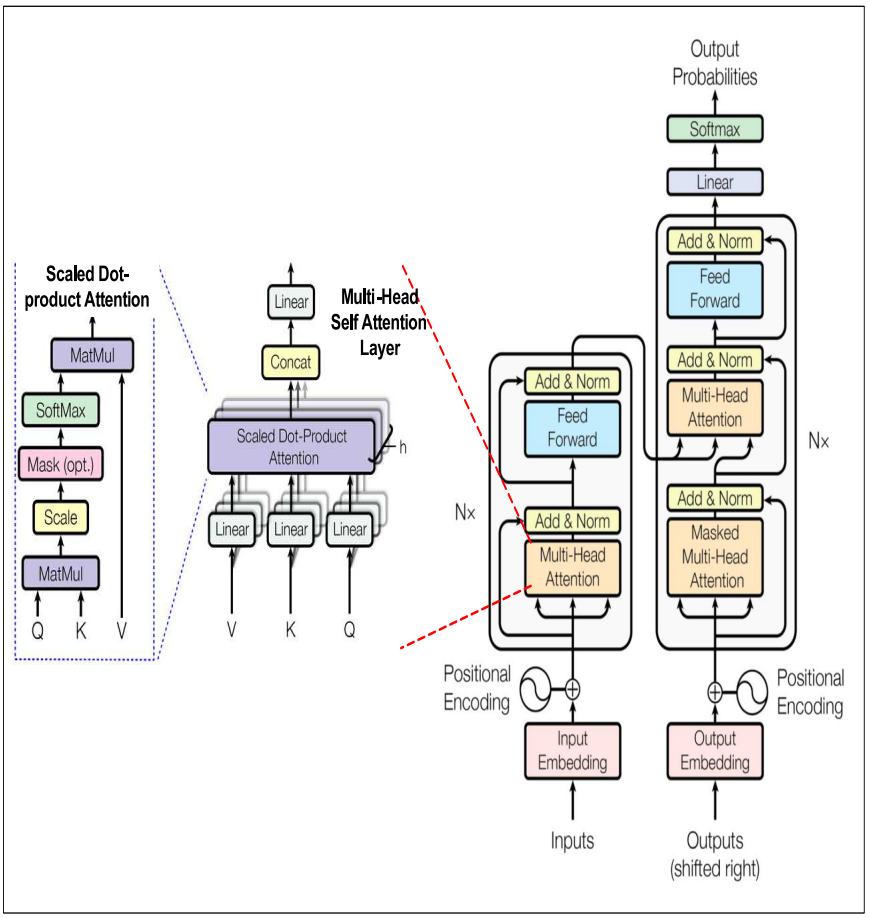


Figure 2.8: Transformer Architecture [108] is parallelized for *seq2seq* applications which is neither CNN nor RNN based. Encoder-decoder branches consists of stacked multi-headed self-attention layers with scaled-dot product function.

Key architectural highlights include, firstly a parallelized self-attention similarity function: the Scaled-dot product (Fig. 2.8). It first calculates similarity score using inner product

between each (either *encoder* or *decoder*) hidden state vectors Q ($[q_1, q_2, ..., q_n]$) against all other hidden state **Key** or **Value** pairs ($[(k_1, v_1), (k_2, v_2), ..., (k_m, v_m)]$) of the same sequence ($K = V$): $K^\top$. The resulting matrix is then scaled with $\sqrt{d_k}$, where d_k is the dimensionality of the **Key** (source or target token) word embedding, that prevents the inner product to become too large (2.3). Followed, by application of *softmax* operation to normalize the 2d ($n \times m$) similarity score matrix resulting in the attention weight matrix of same shape Lastly, an inner product of the attention weight matrix is performed with the Value vector ($m \times d_k$). This effectively results in a 2d matrix ($n \times d_k$) containing weighted sum of all **Values** for each **Query** (Q_t), where the weight assigned to each **Value** (V_i) is the attention weight.

Scaled-Dot Product

$$\begin{aligned} Attention(Q, K, V) &= softmax(\frac{QK^\top}{\sqrt{d_k}})V \\ \text{Where } Q \in \mathbb{R}^{n \times d_k}, K \in \mathbb{R}^{m \times d_k}, V \in \mathbb{R}^{m \times d_k}, \\ m, n \text{ are src., targ. sequence lenghts resp.} \\ d_k \text{ word embedding dim.} \end{aligned} \tag{2.3}$$

Secondly, it uses $N = 6$ parallel stacks of multi-headed self-attention (2.4) sub-layers for both *encoder* and *decoder* branches of the model (Fig. 2.8). Each layer computes h different linear transformation of (*Q,K,V*) controlled by a learned parameter W. Each representation of *Q,K,V* go through h scaled-dot attention computations, each of which is referred to as an attention *head*, thus its called *h-headed* self attention. Finally, all h iterations of the Scaled-dot attention *heads* are concatenated and fed to another parametrized (W^O) linear transformation to output the final context vector (Z).

Transformer Multihead Self-Attention

$$\begin{aligned} MultiHead(Q, K, V) &= [head_1; ...; head_h]W^O \\ \text{Where } head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \\ & W_i^Q, W_i^K, W_i^V, \&, W_i^O \\ & \text{are parameter matrices to be learned.} \end{aligned} \tag{2.4}$$

Each *transformer encoder* and *decoder* attention layer combines by addition of self-attention output Z with the original positional encoded input sequence (Q) and normalizes the results. The addition operation is similar to adding residual connections within each layer to reduce over-fitting and learning saturation. The output of *add and norm* operation is then fed to a feedforward (FF) neural network (auto-encoder), which re-shapes the output in the desired dimension for a yet another final *add and norm* operation.

The only difference between *transformer encoder* and *decoder* branches is the latter includes an additional multi-headed self-attention sub-layer, which merges the *decoder* input and *encoder* input attention based representations together as input to the subsequent multi-headed self-attention sub-layer in *decoder*.

Final *transformer* output is simply a linear dense layer followed by *softmax* operation to emit the target token class probabilities.

BERT and SNAIL

Other notable model architectures that have extended the basic *transformer* architectures are: BERT and SNAIL.

BERT [127] (Bidirectional Encoder Representations from Transformers) has shown to achieve (2018) state-of-the-art performance for various NLP tasks such as in question answering (SQuAD v1.1 database) [128], natural language inference (NLI) [129]. This architecture utilizes the *transformer* encoder to allow transfer learning. In that, it demonstrated developing a pre-trained deep NLP model, which can be fine-tuned with the addition of just one output layer to customize to a various range of applications without any added training overhead compared to other existing transfer learned models [130]. Furthermore, BERT adopts using a non-directional language learning model, unlike other NLP models that learn sequences sequentially (either left-to-right or right-to-left), thereby proposing two learning regimes: Masked Language Model (MLM) and Next Sentence Prediction(NSP). This allows BERT to easily be fine-tuned as its pre-trained with the learning goal to not just predict the next token in sequence, but learning the token-level context from both direction crucial to question answering tasks.

Even though *transformer* model has shifted away from using RNN and CNN for capturing inter-sequence dependencies, despite it using positional encoding to retain the sequence information. *Transformer* model may sometimes fall short for applications where a very long sequence of tokens exist, such as in meta-token sequence learning are required.

The SNAIL (Simple Neural Attentive Meta-Learner) [131] architecture it prioritizes capturing the sequential information in sequences by using temporal convolutions with causal attention layers. This is in contrast to*transformer* architecture that can draw context relationships from infinitely long sequences due to self-attention in which *Query* and *Key-Value* pair, are treated as unordered tuples lacking positional dependence. This can be undesirable, especially for reinforcement learning, where the observations, actions, and rewards are intrinsically sequential [131].

2.6 Summary

As depicted in Fig. 2.1, this research draws on concepts from multidisciplinary domains spanning human factors, cognitive science, computer vision, and pattern modeling techniques.

The literature survey space map outlines the most relevant concepts, system philosophy, goals, scientific foundation and key enabling technologies upon which to base the proposed *EYE-on-HMI* framework. Majority of the related works discussed in **Non-Intrusive Monitoring** section rely on intrusively monitoring human operator actions either indirectly by tapping into the HMI controller or measuring other physiological parameters (e.g. blood pressure, eye tracking, etc) or by recording brain EEG (electroencephalographic neuro markers signals) - all of which do not address non-intrusive monitoring. Nevertheless, various techniques and metrics of analyzing human performance data to model human errors are quite valid and applicable to this research. However, there is a lack of a framework to collect operator situational awareness data. Furthermore, current state-of-the-art technologies in computer vision and machine learning suggest the realization of the proposed ViDAQ systems and HMI forecast modeling solution is encouraging and will only improve with time.

Chapter 3

Proposed Solution

This chapter provides an in-depth background of the proposed solution components. The sections covered in this chapter are as follows:

***EYE-on-HMI* Framework** section provides an overview description of the proposed framework and its limitations.

ViDAQ Framework section discusses image processing pipeline in ViDAQ for its two component use cases for reading a rotary multi-dial gauge and array of lamp indications.

HMI State Space - Time-Series Modelling section presents the HMI state space model abstraction. This model is sufficiently generalized to be applied to any arbitrary HMI system that shall be monitored within the *EYE-on-HMI* framework. It aids in identifying the feature set and nature of information that would yield from a given HMI to be modelled. In addition key assumptions are discussed that allows modelling of HMI state transition sequences as a time-series stochastic process possible.

HMI State Space - NLP Modelling section presents alternative HMI state space model abstraction based on natural language processing. In comparison with the former HMI state space model based on time-series data, the NLP based model is useful in transforming the underlying time-series forecasting problem into a language translation problem without loss any generality in predicting HMI states in n-step ahead time windows.

3.1 *EYE-on-HMI* Framework

The proposed *EYE-on-HMI* conceptual framework (Fig. 3.1) is poised to provide an independent closed-loop validation of human-in-the-loop CPS by using HMI states. Validation of

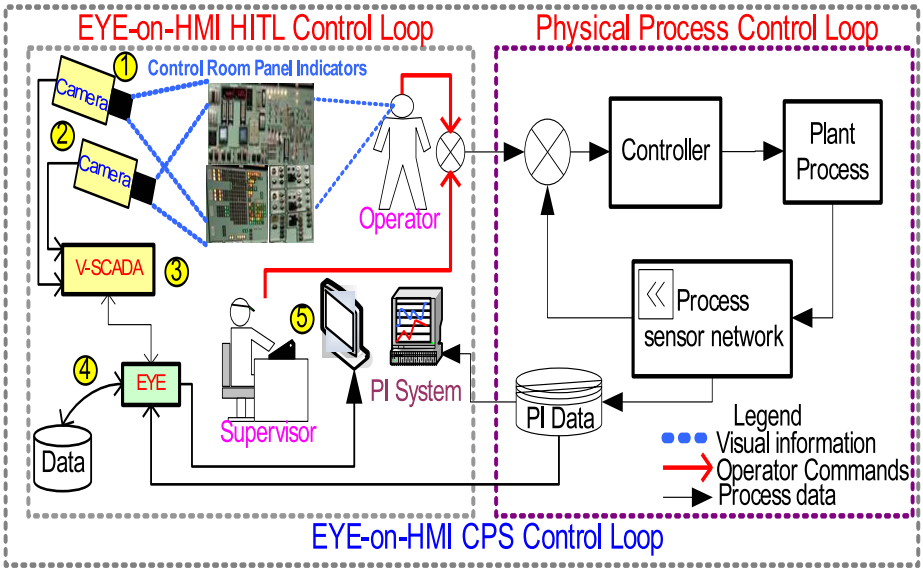


Figure 3.1: Expert Supervisory System (EYE) on HMI CPS integrated with in an NPP control room environment application. Showing all fundamental core components.

plant process state will aid operators with early detection of human-in-the-loop errors. The framework can conceptually be realized as a step-wise data flow as presented below.

Following describes core framework components in a conceptual control room application: Operator view (Fig. 3.1 step-①) of the control room panel HMIs can be captured using an array of cameras pointed from multiple angles to ensure an unobstructed stereoscopic view at all times. Camera video stream feeds (Fig. 3.1 step-②) can be captured by specialised data gathering hardware platforms synonymous with industrial automation such as DCS[1] and SCADA[2].

Current advacement in high bandwidth network switches and use of IP camera for intelligent security surveillance combined with VCA features (E.g. unauthorized track crossing detection on public transit stations, suspecious activity alerting at airports, etc) has brought forward several commercial off-the-shelf solutions [132, 133]. These VCA based solutions, commonly referred to here as V-SCADA (VCA enabled SCADA)(Fig. 3.1 step-③), can inspire further development of specialized intelligent platforms for industrial control room surveillance to do visual data acquisition (ViDAQ) for typical industrial HMI devices (E.g. analogue meters, digital bar displays, indicator lights, process controller display, etc)
Successful data logging of temporal HMI events can be used by a EYE (expert supervisory

[1]Distributed Control System - a computer based control system usually targeted for monitoring and controlling several processes in a plant from a control room.

[2]Supervisory Control and Data Acquisition - A term more common to industrial controls. SCADA refers to a system involving network of programmable controller(s) usually dedicated for controlling or monitoring a set of plant processes.

system) (Fig. 3.1 step-④) to correlate real-time plant process data obtained from the plant information (PI) system. Finally, EYE can generate cross-validation overview displays and reports for human supervisor to monitor both Operator command response in relation to live control room HMI state (Fig. 3.1 step-⑤).

The benefits of the *EYE-on-HMI* framework include: (a) capturing and trending operator situational awareness in real-time (as demonstrated in Section 5.1.1); (b) extending supervisory oversight on any missed procedural step for post event analysis and lastly; (c) holding a long term contextual memory of all actual or spurious HMI indication events which can help to correlate events using pattern recognition and yield useful hypothesis; (d) HMI state models trained on several datasets collected during operator training exercises during in Nuclear Operator Training Simulators can used to used as benchmark input into HMI reliability and safety analysis using probabilistic risk assessments.

3.2 ViDAQ Framework

The proposed ViDAQ framework using camera-based non-intrusive monitoring of Control room panels is to allow cross-validation of what the operator sees on the control panel and their actions.

Industrial control panels, especially in Nuclear power plants, are generally densely populated with various instrumentations such as dials (both digital and analog), lamps, hand switches (Sec. 1.2).

The advantage of capturing these panel instrumentation using a camera is to:

(I) **Capture Operator View**: Truly capture what the operator is viewing in-real time - to over-come *looking-but-not-seeing* effect due to poor situational awareness.

(II) **Minimal Retrofit Design Change**: Since not all control room HMI instrumentation are captured digitally, an independent data acquisition system such as proposed ViDAQ can allow digital reading. ViDAQ also requires the least amount of retrofit design change to the existing control room instrumentation.

(III) **Early Fault Detection**: Few HMI instrumentation may be indicating a faulty indication in the control room and may not be caught by the operators – ViDAQ monitoring system may be able to cross-check their indication with process signals in real-time. Burnt-out indication lamps may be detected sooner as well.

ViDAQ is only meant to monitor the HMI states, which also includes any operator initiated changes on the HMI in the form of button presses, hand switch state changes. Generally,

human factors engineering requirements require HMI designs to include some visual feedback that confirms operator actions have been registered E.g appropriately. In the control room, most buttons are back-lit, which indicates if the button has been depressed and hand switch states have position notches (Sec. 1.2.

Therefore, the design of an independent visual data capturing system for the control room panel states, as "seen" by the operator, resulted in the concept of ViDAQ and EYE-on-HMI frameworks. These two system frameworks shall provide cross-validation of operator actions with respect to information as presented on the HMI and detect error precursors.

There are obvious challenges with vision based systems such as:

- Acquisition Errors due to optical issues e.g. lighting conditions
- Obstructions
- Scalability

The ViDAQ concept envisions a system employing a distributed array of cameras to overcome few above listed challenges with obstructions. Moreover, there is an opportunity that control room environments may slightly be favourable for ViDAQ and application of computer vision than in other domains owing to:

- Control room ambient lighting conditions are maintained at constant levels – so once ViDAQ is calibrated it will not require re-adjustments.
- There is hardly any vibration – so cameras can be positioned at constant distance away from panel.
- Operators are trained to maintain a constant physical distance from the Panels – so HMI obstruction may be minimized.

Computer vision technology is also continuously improving and has found successful application in several industrial manufacturing, self-driving cars, robotic domains. It is envisioned some of these advancements may be applied to the ViDAQ to improve control room HMI panel state monitoring.

Conceptually, the ViDAQ (Visual Data Acquisition) framework [134] can involve confluence of several pre-selected algorithms required for non-intrusively (using computer vision) to acquire HMI states. In scope of current research typical HMI information output devices (e.g. rotary dials and alarm indication lamps) that are encountered in any industrial (e.g.

NPP) control room are considered as examples for proof-of-concept demonstration of ViDAQ concept.

This section discusses the proposed ViDAQ framework which, currently includes custom algorithm (Fig. 4.6a) to visually acquire readings from analogue rotary dial gauges as discussed below. Further extension of ViDAQ algorithm to read alarm states from coloured indicator lights (Fig. 4.4a) is also discussed below.

Extrapolating the commonality revealed in the above related works (Sec. 2.3.2), following are the pre-processing steps required to prepare images as inputs to the ViDAQ. These include: (1) image thresholding for conversion to binary image using Otsu's thresholding, which adaptively generates binary images that are clearly segmented with respect to foreground and background; (2) smart application of fundamental image morphological operators (dilation, erosion); (3) image segmentation to isolate the ROI containing only the dials. While all such proven and general steps have been considered in the ViDAQ framework as deemed required, our focus is mainly to improve dial tip localization in the ROI to extract the measurement value of interest with sufficient precision. ViDAQ uses a new approach for reading multi-dial meters using image contours and convex hull edges as discussed in following section.

3.3 HMI State Space - Time-Series Modelling

A state space model is developed below to aid in generalizing time-series patterns generated by a typical HMI system. Such pattern inclusively are intended to capture both the indications and operator actions. Moreover, there are two key assumptions that aids in adopting the proposed general HMI model as discussed below.

Conceptually, HMI states (Fig. 3.2) for any typical plant process may be captured by two categories of state features: process output (*PROCESS*: X_1) and human input (*HMI_USER*: X_2) vector. Each feature vector variable can be tuple of binary valued states (E.g. indication lamp states, push buttons states, etc) and/or a finite range of analogue valued states (E.g. rotary dial indicators, digital setpoint displays, etc).

For example, in Fig. 3.2, Process outputs: $q1, q2, q3$ are analogue variables associated with each rotary dial value and $q4$, a *4-bit* binary word, can capture indication patterns of all four lamp string. Similarly, Human operator inputs: $i1, i2, i3, i4$ as digital variables, can be associated with each push buttons and $i5$ unsigned integer variable can be associated with a setpoint indicator. An ensemble of such multi-type variables (or features) can sufficiently capture all states of the HMI. Trending the HMI state features yields multi-variate time series data.

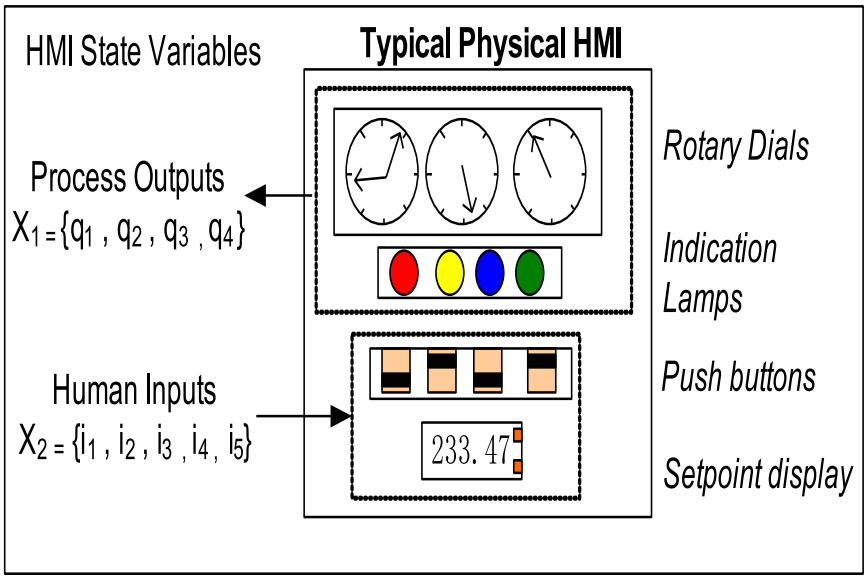


Figure 3.2: A Typical Physical HMI model. HMI State variables can be classified as representing Process Output indication (X_1) and Human Input (X_2) vectors. Each state variable can either be digital or fixed point real numbers [135]

.

3.3.1 HMI Model Time-Series

A series of data points that are indexed at equal time intervals is referred to as time-series data. Univariate or multi-variate time-series (TS) data captures the sequences with-in processes such that the next time step state data is dependent on the process state at previous time steps. Time-series analysis differs from classical classification and regression predictive modelling, in that the temporal structure envelopes underlying patterns. For example, weather data, stock prices, industrial processes, etc. Analysis of time-series data attempts to extract and model such dependencies which can then be used to do n-steps ahead forecast of the process state variables.

Typically (as show in Fig. 3.3) there are three classes of TS that are statistically possible, with the combination of which other more complex time-series may be decomposed into or analysed using.

In consideration of the above proposed HMI time-series feature model (Sec. 3.3) for practical applications, I anticipate a multi-mode multi-variate time-series may also be encountered, i.e. a case where an HMI being modelled may have to allow for its data features to have combination of all or few TS types shown in Fig. 3.3. Such, models may have to utilize advanced feature engineering and modelling techniques.

However, for the scope of this research, the above HMI state space time-series model assumes all its features are of the same type of TS, desirably of weakly-stationary (Fig. 3.3 c)

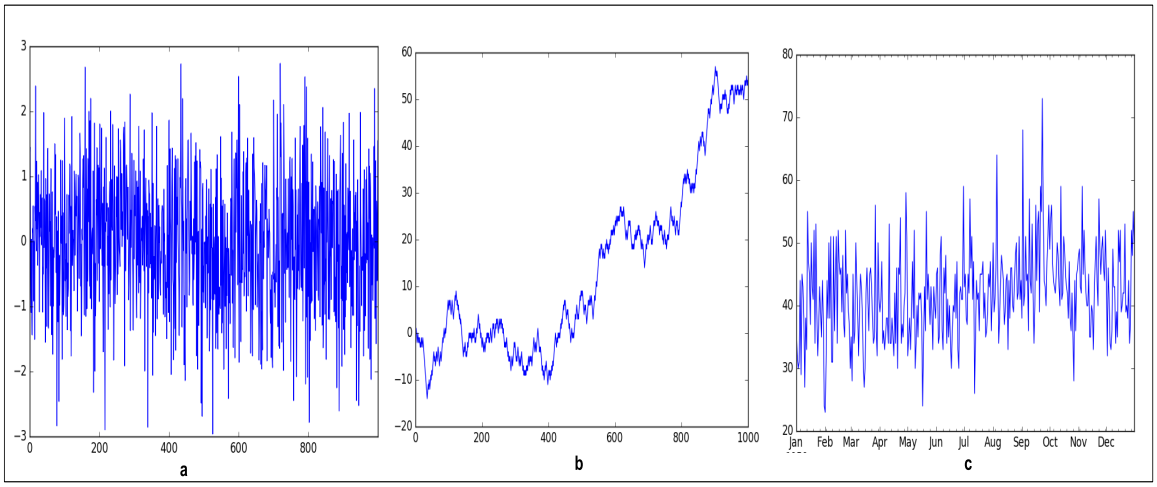


Figure 3.3: Example of Time Series (TS): (a) Gaussian White noise (Strictly Stationary) [$\mu = 0$, $\sigma^2 = k$ (constant), $\gamma[\tau] = 0$ for all τ]; (b) Random Walk (non-stationary); (c) Weakly Stationary [$\mu \neq 0$, $\sigma^2 = k$ (constant), $\gamma(t, s) = \gamma[\tau] = X$ auto-covariance between any two points (t,s) is constant for fixed $\tau = t - s$ distance (lag)] as there is no obvious trend and repeating seasonality effects [83]

type such that they may be modelled for HMI state forecasts in order to detect HITL errors.

3.3.2 HMI Model Weakly Stationary Assumptions

Two plausible and helpful assumptions for HMI state patterns that allow using standard time-series forecasting techniques are rationalized below:

First assumption, is that a HMI process may not yield a data series that is *white noise* (ε) - a series that is generated by random variables that are independent and identically distributed, i.e. having zero mean $(\mu = 0)$, with identical finite variance $(\sigma^2 < \infty)$ that are serially uncorrelated $E[\varepsilon_t \varepsilon_k]$ for all $t \neq k$ (3.1).

White noise TS (Fig. 3.3 a) is statistically random, unpredictable and, thus cannot be modelled for practical purposes. However, we ideally expect the HMI state time-series forecast errors or residuals to be white noise, which implies there are no recoverable patterns left to be learned from the forecast errors (difference between actual and predicted value output by the forecast model).

Otherwise, a HMI displaying white noise state patterns would suggest an error or malfunction in the external process displayed by the HMI system.

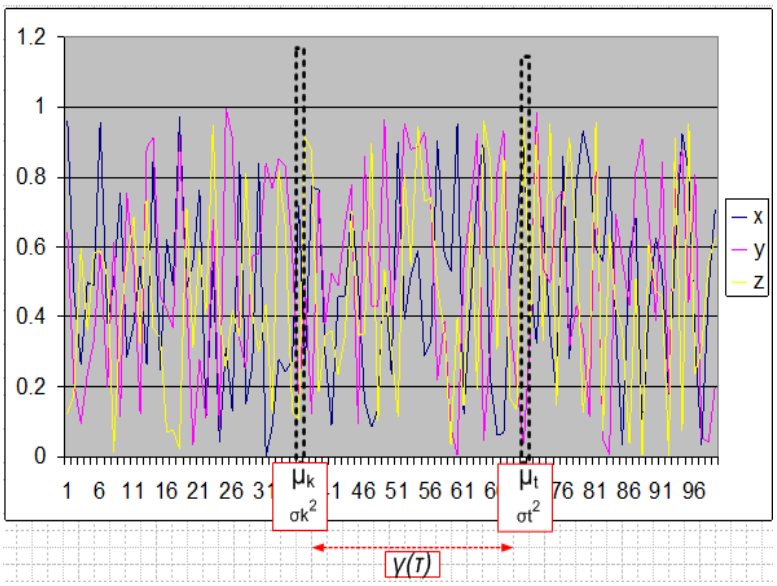


Figure 3.4: Depiction of an arbitrary Weakly stationary time-series traces (*X,Y,Z*) all generated from a stochastic process whose statistical properties are time invariant between various runs: implying the means (μ_k, μ_t) and variances (σ_k^2,σ_t^2) are finite and relatively constant over various time slices. Moreover, auto-covariance ($\gamma(\tau)$) between various time slices is finite and is only a function of temporal distance $\tau = |t - k|$.

White Noise

$$x_t = \varepsilon_t$$
$$\text{where } \varepsilon_t \sim (0, \sigma^2) \text{ with } \sigma^2 < \infty \text{ and } E[\varepsilon_t \varepsilon_k] = 0 \quad (3.1)$$
$$\text{for all } t \neq k$$

Second assumption, is that a majority of HMI state transition can be modelled as stochastic processes that yield a weak stationary (Fig. 3.4) multi-variate time-series in most practical scenarios. A weakly stationary process must satisfy three conditions as listed in (3.2) and yields a time-series (Fig. 3.4, Fig. 3.3 c) where (1) mean ($\mu < \infty$) (2) variance ($\sigma^2 < \infty$) are approximately finite and constants for all t windows (i.e. these do not vary with time), while (3) the auto-covariance ($\gamma(t, k)$) between any observed values at two time slices of a stochastic process is finite and constant for all τ, that is, the auto-covariance of a weakly stationary TS only depends on the temporal distance ($\tau = |t-k|$) between any two time points (t and k). Auto-covariance when normalized by standard deviation of each observation of TS, results in auto-correlation function (ACF), which makes the measure unit less. ACF (3.3) is calculated for various lagged versions of the TS, which show the degree of similarity of TS with lagged version of itself indicating presence of various patterns that can be modelled linearly.

Weakly Stationary Requirements

$$\begin{aligned} \mu &< \infty \text{ for all t} \\ \sigma^2 &< \infty \text{ for all t} \\ \gamma(t,k) = \gamma[\tau] &< \infty \text{ where } \tau = |t-k| \end{aligned} \tag{3.2}$$

Auto-Covarriance and Auto-Correlation Funtions

$$\begin{aligned} \gamma(t,k) = Cov(X_t, X_k) &= E[(X_t - \mu_t)(X_k - \mu_k)] \\ ACF(\tau) = \rho(t,k) = Corr(X_t, X_k) &= \frac{\gamma(t,k)}{\sigma_t \sigma_k} \\ \text{with} \quad -1 < \rho(t,k) &< +1 \end{aligned} \tag{3.3}$$

The rationale supporting above two assumptions is based on the fact that human machine interfaces primarily display the process values and accept operator inputs as commands. The process parameter values are ultimately governed by underlying process control laws modelled by a system of differential equations that vary in a tight allowable band (range bounded). Moreover, the operator inputs also change in some correlation to the process values. Therefore, process values and operator inputs ought to display causality effects (i.e. either the process information having influence on operator actions or vice-versa). For most practical scenarios the range of operator inputs are found not to vary indefinitely. Otherwise, those scenarios would require operator actions outside their normal range of trained behaviour (E.g. driving a vehicle on a freeway has set of rules every human driver normally adheres to). Hence, chances of any HMI state transition resembling that of a *random-walk* (Fig. 3.3 b) stochastic process is minimal, and therefore is not currently addressed in the scope of this work. Above, weakly stationary assumptions for the HMI generated time-series patterns make it possible to develop either linear regression based forecast models based on ARIMA [83] or using non-linear recurrent networks such as LSTM [135].

3.4 HMI State Space - NLP Modelling

Previously stated weakly stationary time-series assumption also implies finite system state transitions which, allows one to model the HMI system using framework of a finite-automaton (FA) (either deterministic or non-deterministic), broadly identified as a Discrete Event System (DES). DES by definition is an event-driven system, where its state transition occurs with discrete events and there is also no restriction placed on the nature of state space $(\mathcal{Q})$ of a DES to be either discrete or continuous or mixed. Such basic qualities of DES aligns with the previously stated HMI model assumptions (Sec 3.3).

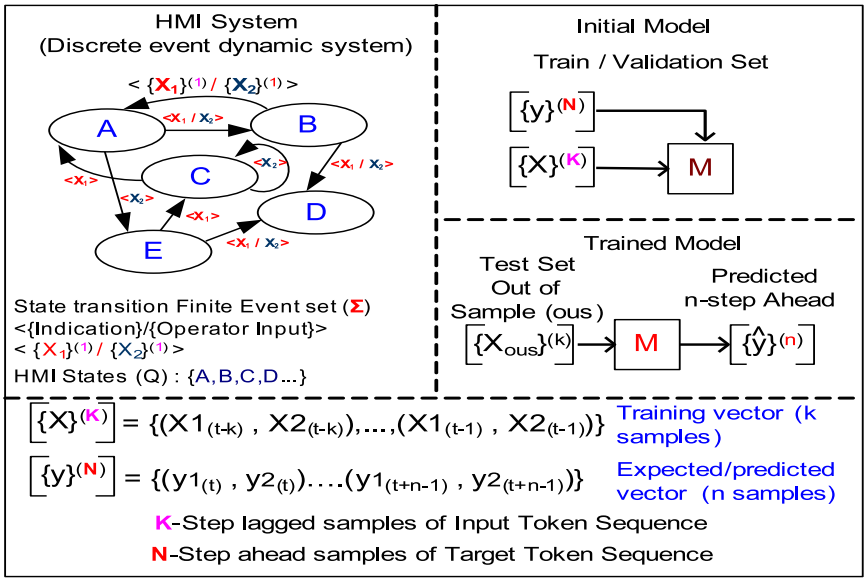


Figure 3.5: An arbitrary HMI System modelled as a Discrete Event System (DES) suitable for Natural Language Processing (NLP) applications; (left) Non-finite set of DES states (A,B,C... ϵ $\mathcal{Q}$) or HMI state map to sequence of events or string. HMI events as directed edges in HMI DES diagram map to words from a Finite dictionary ($\sum$); (right) Shows model training using K past and N step ahead sequence of HMI events as input and expected languages respectively (bottom).

Under the Ramadge and Wonham (RW) framework [136], a HMI DES plant model P can be obtained by a formal language generated by a "generator" FA ($\mathcal{G}$), whose alphabet consists of the (finite) set of events ($\sum$). The **generator** is defined as a 5-tuple (3.4) which can be depicted as a HMI DES directed graph (Fig. 3.5) with its the nodes as DES states from $\mathcal{Q}$ set and edge set defined by pairs (q, q'), such that $\delta(q, \sigma) = q'$ for some $\sigma \epsilon \sum$. That is an edge between states q and q' can be labelled with event σ that transitions the HMI DES from state q to q'.

The $\sum$ (3.4), is interpreted as the *alphabet* set corresponding to a finite set of **events** or directed edges in Fig. 3.5, which maps to a particular value of a HMI feature vector $X =< X_1, X_2 >$ (Fig. 3.2). The HMI DES state transition sequence is specified by δ, a partial function $\delta(q, \sigma)$ that is not required to be specified for all q (states) in $\mathcal{Q}$ and all σ (events) in $\sum$. In fact δ (3.4) is the function that must be learned and approximated by natural language processing (NLP) algorithm.

It is noteworthy to restate that RW framework only expects a finite set of HMI state transitions ($\sum$) but not necessarily a finite HMI state ($\mathcal{Q}$) set. This implies that the NLP model can be trained with a finite set of HMI events (*alphabet*) or dictionary of events each corresponding to a point in HMI feature space X. A non-finite HMI state set implies HMI DES can

have several sequence of events (**s**) or *strings*. This is the key realization that enabled us to consider evaluating NLP deep learning algorithm to model HMI DES.

Furthermore, HMI DES language model can be formally specified (3.5) under RW framework as a **language** $\mathcal{L}(\mathcal{G})$ over an event set $\sum$, as any subset $\sum^*$ which captures all (finite) strings s built using elements in $\sum$.

$$\begin{aligned} \mathcal{G} &= (\mathcal{Q}, \sum, \delta, q_0, \mathcal{Q}_m) \\ \mathcal{Q} &: \text{ States set} \\ \sum &: \text{ Events set} \\ \delta(q, \sigma) &= \sum \times \mathcal{Q} \rightarrow \mathcal{Q} \\ &\text{Where } q \,\epsilon\, \mathcal{Q}, \sigma \,\epsilon\, \sum \\ (q_0, \mathcal{Q}_m) &: \text{ (initial state, final (marker) states)} \end{aligned} \tag{3.4}$$

$$\mathcal{L}(\mathcal{G}) = \{s : s \,\epsilon\, \sum^* \subseteq \sum \,\&\, \delta(s, \sigma) \text{ is defined}\} \tag{3.5}$$

Lastly, HMI DES model under RW framework also inherently addresses scalability as DES can be expanded by incorporating subsystems, or sub-processes $\mathcal{G}_1, ..., \mathcal{G}_n$ that are asynchronous and independent as long as each $\mathcal{G}_i$ alphabet set $\sum_i$ are disjoint. That is the complete model of a HMI DES plant can be specified by a **shuffling** the languages of HMI subsystems $\mathcal{L}_1 ... \mathcal{L}_n$ which is denoted by $\mathcal{L}_1 \parallel \mathcal{L}_2 \parallel ... \parallel \mathcal{L}_n$ and defined by (3.6). Where, $s \uparrow i$ is the projection of **s** on $\sum_i$ that only keeps *alphabets* or events belonging to $\sum_i$. Which also implies the complete HMI DES language model can be obtained by appending new event symbols and patterns from other HMI subsystems.

$$\mathcal{L}_1 \parallel ... \parallel \mathcal{L}_n = \{s : s \,\epsilon\, \sum^* : s \uparrow i = s_i \,\epsilon\, \mathcal{L}_i, i = 1, ..., n\} \tag{3.6}$$

In summary, above RW framework for HMI DES allows application of NLP algorithms by specifying a generator automaton that can specify a DES by control language. The generator language dictionary is a finite event set containing alphabets or words equivalent to the finite range of discrete unique values various HMI features can take on. Patterns or sequences of events make the sentences or states of the HMI DES language and cause HMI to transition from one state to another. NLP algorithm are used to create deep learning models to estimate state mapping mechanisms (δ) through supervised training. Once trained such model can

translate multi-length sentences from one controller language into another.

3.5 Summary

The *EYE-on-HMI* core component functionalities are shown as an application in an industrial control room environment in this chapter. A discussion highlighting the benefits of this solution was provided. ViDAQ framework was discussed with its potential benefits to industrial control room applications. Despite the challenges with computer vision, there are opportunities in the industrial control room environment that will make ViDAQ feasible.

Lastly, the HMI state time-series model, which is based on a few key initial assumptions, is developed. Lastly, a discrete event system automata-based HMI model is developed as a way to utilize modern NLP machine translation multi-classifier machine learning models.

Chapter 4

Implementation Details

This chapter provides implementation details of proposed solutions. Notably, contributions of this research include the ViDAQ implementation for two test cases, implementation of various ARIMA models, various RNN based LSTM and CNN time-series models based on the specifications developed for time-series HMI state model (Sec. 3.3). Lastly, implementation of a custom designed NLP LSTM based *seq2seq* model based on the specification of the alternate discrete event system based HMI state model (Sec. 3.4) is discussed herein.

4.1 ViDAQ Design Details

Visual Data Acquisition in *EYE-on-HMI* framework is poised to provide non-intrusive remote monitoring of HMI states which also indirectly captures operator actions - a concept that is referred to as looking-inward approach.

As introduced in Sec. 3.2 in previous Chapter, this research implemented two instances of ViDAQ test cases for industrial control room applications: Multi-dial gauge reading and Alarm lamp indication reading.

The image processing pipelines of ViDAQ is implemented using python OpenCV library version 3.4, OpenCV contrib. modules for ARUCO, Ubuntu 18.04, Jetpack 4.2, gstreamer (for camera). The camera devices: USB camera Logoitech C720 HD Webcam.

To leverage the high performance computation advantage of a dedicated video processing and machine learning platform the ViDAQ code was evaluated on the **NVIDIA Jetson TX2** single board computer. This platform features: NVIDIA Pascal GPU with 256 CUDA capable cores. The CPU complex consists of two ARM $v8$ 64 bit CPU clusters which are connected by a high-performance coherent interconnect fabric. The memory subsystem incorporates a 128-bit memory controller, which provides high bandwidth LPDDR4 support. 8GB LPDDR4 Main Memory and 32 GB eMMC Flash memory are integrated on the Module.

Figure 4.1: ViDAQ Evaluated on NVIDIA Jetson TX2 Board.

For development and iterative testing all ViDAQ algorithms were primarily ran on either Jetson TX2 or laptop computer equipped with commercial web camera devices. OpenCV $ArUco$ module generates binary image square markers that are used here to detect the edges of the simulator mimic panel or the actual full-scope simulator panel as shown in Fig 4.2 and Fig 4.3 respectively.

4.1.1 ViDAQ - Indicator Lamp States Detection

Following is an overview of the stages of the ViDAQ framework that are unique for detecting the alarm indicator lamp state detection component, as shown in Fig. 4.4a.

Image Processing Pipeline - Feature Reduction

Indicator lamps usually indicate binary states using two colors (E.g. Red and Green). Therefore, input images are first converted to Hue, $Saturation$ and $Value$ (HSV) color space; this is referred to as image feature reduction (Fig. 4.4a) in this thesis. Transformation of input RGB image to HSV entails converting to a 3 dimensional cylindrical coordinate space that allows intuitive selection of Hue and $Saturation$ ranges for detecting only the desired color captured at various $luminance\,Value$ levels. This otherwise is practically not feasible to achieve in RGB space owing to a non-linear combination of Red, $Green$, $Blue$ values required for filtering similar shades (gamut) of a color.

The resulting benefit is a robust color detection methodology that is independent of color intensity (or object illumination) and display device or sensor dependent RGB response curve. Whereas, in RGB space, it is challenging to determine the luminosity of various color shades (gamut) owing to a non-linear combination of Red, $Green$, $Blue$ value required to achieve

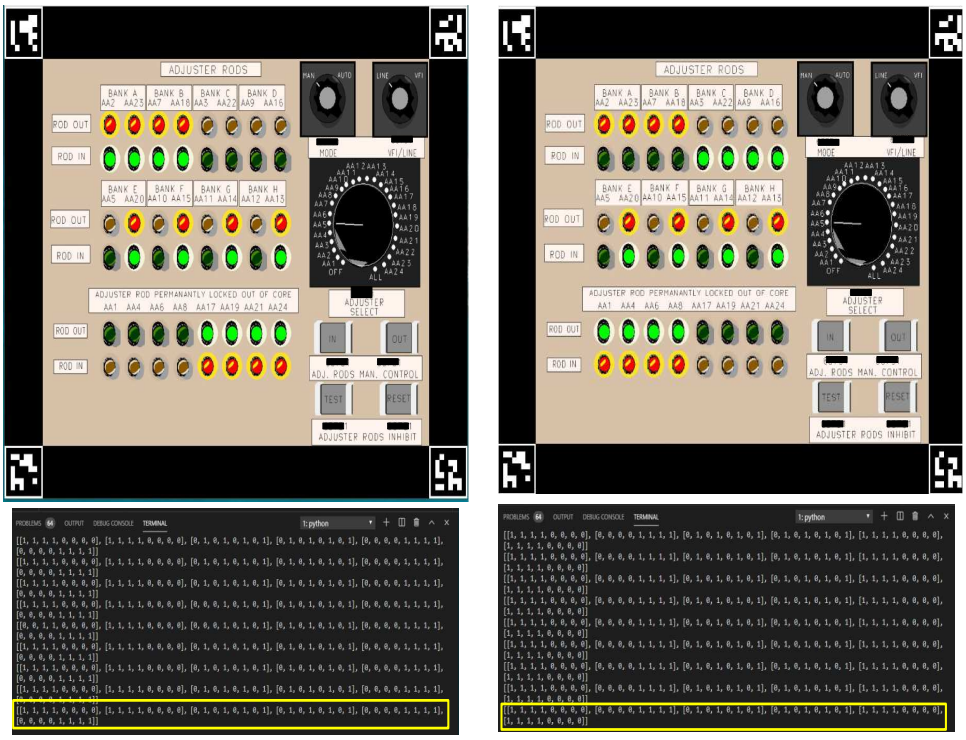


Figure 4.2: ViDAQ reading simulator software mimic control panel lamp states using Jetson TX2 board. $ArUco$ binary image markers are placed on panel corners by the panel mimic software, to aid in detection of panel shape boundaries when acquired using a camera. Top row of figures show simulator control panel lamps for two patterns. Bottom row shows bit patterns as read by ViDAQ in real-time. Highlighted in yellow is a set of 6 bit arrays for each row of red and green lamps.

the required color. HSV further assists in thresholding the image in HSV color space in order to create a binary mask that is of the same size as the original $3-$channel RGB image.

Once the HSV mask is applied (using bit-wise AND) to the original RGB image, it sets all R,G,B channels to zero value (black) for those pixels where the HSV mask has a 0, effectively suppressing pixels at locations where RGB values are non-zero for color shades that are required to be masked. In Fig. 4.4b, a green and a red mask is computed by thresholding the original HSV image using pre-determined upper and lower Hue, $Saturation$ and luminance $Value$ as thresholds for suppressing each red and green colored indicators. For example, the green HSV mask is applied to the original input RBG image to only detect indicators that are illuminated red and vice-versa.

Feature Extraction - Indicator Profile Detection

Feature extraction for indicator lamp entails identifying its shape profile to recognize the indicator type and functionality accurately. For instance, an HMI panel may have indicator

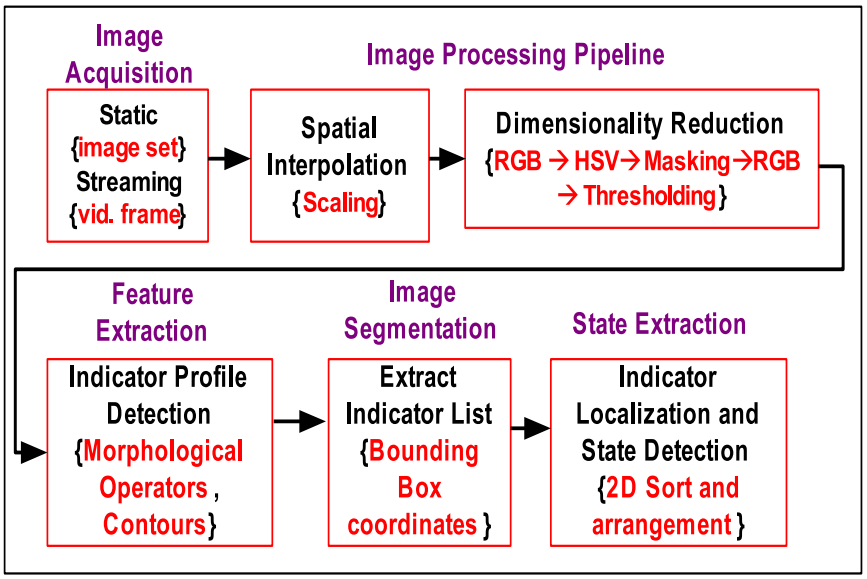
Figure 4.3: ViDAQ reading actual full-scope simulator control panel lamp states. $ArUco$ binary image markers are stuck by tape to aid in the detection of panel shape boundaries. Shows camera tripod to view the live panel lamp states.

lamps that are circular, square, etc. each displaying different colors.

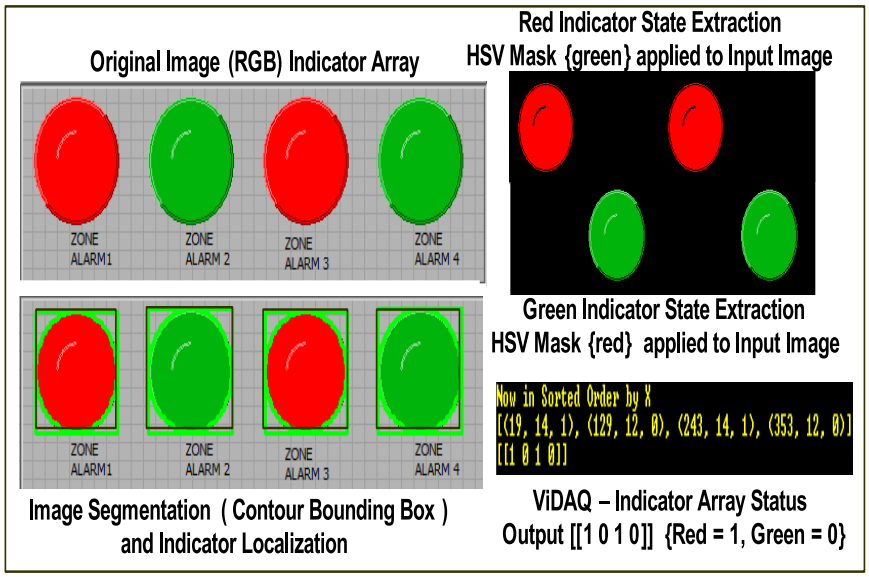
Morphological operators such as dilation and erosion [67] are applied to repair the indicator profile boundaries of any holes or irregularities which, assist in accurate contour set generation [137]. Contour set closely encloses the indicator profile outline, shown as bright green boundaries around each indicator in Fig. 4.4b. Contour sets are generated individually on each masked RGB image output, after it has been converted into a binary image using Otsu's Thresholding [75]. For example, in Fig. 4.4b the red and $green$ masked RGB image outputs are each converted to binary images using Otsu's Thresholding [75] separately and then used as inputs to the feature extraction step using contour generation.

Image Segmentation - Extracting Indicator State List

Once the indicator profile has been identified using contour sets, image segmentation can be achieved by means of determining bounding boxes enclosing the indicator lamps. Contour bounding boxes enclose the entire contour by reducing the storage to just two tuples: top-left corner coordinates and box width. This serves to create an enumerated list of tokens representing indicators in each masked image. Specifically, image segmentation is performed separately on each masked output image representing the two or more indicator states (colors). As shown in Fig. 4.4b, the list of 4 bounding boxes with black boundaries are merged into one list that is used to locate each indicator in the image in the correct order as discussed below.



(a)



(b)

Figure 4.4: ViDAQ Processing Stages (a) Dial Gauge Reading (*Note: dimensionality reduction in this context refers to reducing the image features E.g., size, color channels, ROI selection, required for subsequent processing stages*); (b) Alarm Indicator Detection and Localization. Red/Green HSV masks help to detect active lamp states Green/Red respectively. Localization is done by sorting the Cbb (contour bounding box) list as per Algorithm 1. It identifies lamp indication by its position on the panel.

Indicator Localization and State Detection

Prior to indicator state detection, indicators must be identified by localization based on their relative position in the acquired image (implemented by Algorithm 1). Indicator localization

for accurate indicator identification is based on prior knowledge of the HMI layout (Fig. 4.5) being captured using ViDAQ. For example, in Fig. 4.4b, it was known all indicators on the HMI panel are fitted in a 1×4 layout and numbered left to right. Hence, the list of bounding boxes (found previously) can be arranged into an equivalent data structure - a $2d$ matrix, which is used to hold the indicator states in the same sequence as matching the physical layout.

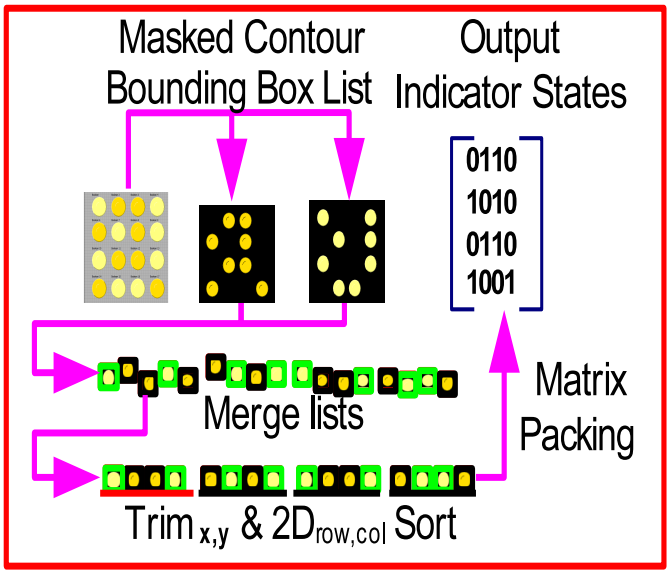


Figure 4.5: Indicator Localization and Final Output Indicator State Matrix using 2D (row and column wise) sort

Algorithm 1: Lamp state Localization and Identification Routine

procedure EXTRACTINDICATORSTATES(Cbl_0,Cbl_1,Row,Col) ▷ Parameters $Cbl_0[(x,y)_0 \dots (x,y)_n]$,$Cbl_1[(x,y)_0 \dots (x,y)_n]$: masked contour bounding box (Cbb) lists for each indicator state $\{0,1\}$, where tuple $(x,y)_n$ is the top-left hand corner coordinate of each Cbb; Row,Col : Row and Column counts defining indicator lamp grid layout on HMI
Initialization:
    $finalState[Row][Col] \leftarrow \{0\}\,\{0\}$ ▷ init. boolean output indicator state matrix
    $indList[Row * Col] \leftarrow \{\}$ ▷ init. flattened merged list of Cbb lists
Merge Cbb lists
    $indList[] \leftarrow \{Cbl_1 \cup Cbl_0 | c_1 \dots c_n\}$▷ Merge Cbb lists and include indicator state for each Cbb type; Where c_i is tuple (x,y,k) and $k \in \{0,1\}$ for indicator state (Fig. 4.5)
Coordinate Trimming ▷ Trim Cbb coordinates (x,y) to fit in a $Row \times Col$ grid
    $indList[] \leftarrow Trim_x(indList[], Col)$ ▷ Make all x values equal, for Column wise trim (Fig. 4.5)
    $indList[] \leftarrow Trim_y(indList[], Row)$ ▷ Make y values equal, for Row wise trim (Fig. 4.5)
2D Sort
    $2DSort_{x,y}(indList[])$ ▷ Ascending order sorting using x then sorting using y all sub-arrays containing Col elements at a time (Fig. 4.5)
Final Matrix Packing
$i = 0$
    for all $(m = 0 to Col)$ **do**
        for all $(n = 0 to Row)$ **do**
            $finalState[m][n] = k \in c_i$ of $indList[i++]$ ▷ Copy only indicator state k from $indList[i]$ (Fig. 4.5)
Output: Final Alarm State Matrix
    $finalState$

Indicator state data structure packing order (Fig. 4.5) is derived naturally by merging all

per indicator state bounding box lists into an array and then running a $2D$ (row $\rightarrow$ column wise) sort using top-left corner coordinates, which arranges all indicator states (Algorithm 1).

Individual indicator state is determined by iterating over each contour bounding box (Cbb) list for each indicator state (On or Off) (Fig. 4.5) as shown in Algorithm 1. For example, two Cbb list are obtained after the feature extraction step using contours set generation on each red and $green$ masked images, as shown in Fig. 4.4b.

4.1.2 ViDAQ - Multi-Dial Guage Processing

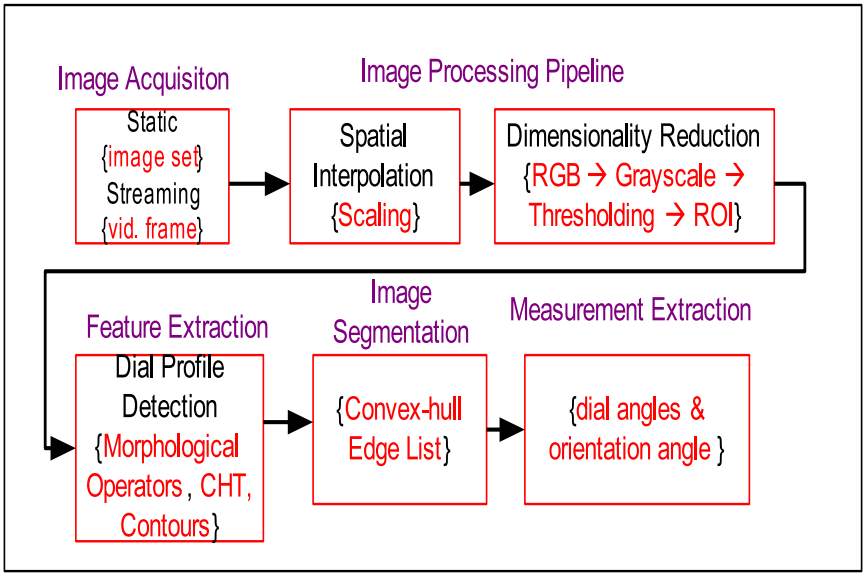
This section provides details of the ViDAQ processing framework (Fig. 4.6a) that is used to optically acquire temporal events indicated by an analog dial gauge (E.g. a clock). The goal is to develop a general ViDAQ algorithm to accurately and precisely acquire values from a typical multi-dial meter, such as a clock with at least two (hours and minutes) dials. Moreover, in a typical industrial control room, there are usually rotary dials for indicating various process parameters. Currently, rotary dials with circular faces are addressed with future extension to recognize a variety of dial form factors. ViDAQ (Fig. 4.6a) essentially entails image acquisition followed by an image processing pipeline, the output of which facilitates the required feature extraction and measurement extraction logic. The following description provides details of each stage.

Image Acquisition

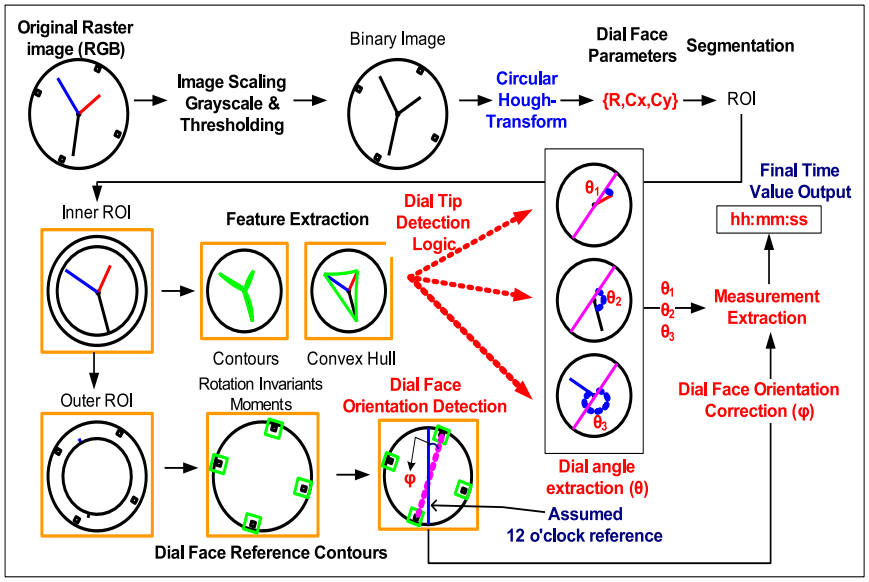
Image source for ViDAQ is selectable between a fixed source (E.g. directory archive) or a streaming source (E.g. live camera video stream). When a streaming source is selected, only frames that vary by a certain threshold compared to the previously processed frame, are forwarded to the image processing pipeline. Image delta is determined by performing an absolute (pixel) color (RGB) difference (using image histograms), with the pixel values in the region of interest (ROI) being assigned higher weights than those outside the ROI area. This is used to optimize the dial movement detection routine and effectively avoid unnecessary processing of similar content frames.

Image Processing Pipeline

This stage of the pipeline performs image pre-processing steps required to prepare the acquired raster image as data for inputs to subsequent feature extraction stages. The pipeline starts with dynamic image scaling, followed by conversion to grayscale and binary image formats.



(a)



(b)

Figure 4.6: (a) ViDAQ Multi-dial Reading Framework (*Note: dimensionality reduction in this context refers to reducing the image features E.g., size, color channels, ROI selection, required for subsequent processing stages*); (b) ViDAQ - Multi-Dial Processing Stages

Spatial Interpolation

It is essential to address the requirement of spatial interpolation (scaling) of input raster images to a required specific resolution $(width \times height)$ or pixel count, prior to processing. Without this step, the input image source may be assumed to have various quality levels (E.g. lossy type JPEG and non-lossy PNG) and captured at high camera sensor resolutions (E.g.

1.5, 2, 12 megapixels). That is, image scaling is required to bound both the run-time and error rate of the feature and measurement extraction algorithms. While, the run-time directly depends on image resolution (workload) size, the error (false feature detection) rate is inversely dependent on image resolution. For example, certain edge detection algorithms (Laplacian and Sobel) perform worst [138] (higher false-positive errors) when edges in the input image are blurry, as in lower resolution images, compared to sharper edges in a higher resolution source image. Naive geometric scaling causes pixelation or artifacts (high-frequency noise). Either Bicubic or Lanczos [139] based scaling for image downs-sampling showed acceptable results and has been utilized in ViDAQ processing framework (Fig. 4.6b).

Image Feature Reduction

Image processing pipeline begins with image feature reduction. This typically is achieved by converting a standard 3 channel (R,G,B) 24-bit per pixel image $(Width \times Height \times 3)$, into a single channel 8-bit per pixel grayscale image $(Width \times Height)$, which results in an image with only shades of gray. This sufficiently preserves the required visual features (edges and contours) with required luminance levels minus the chromatic noise. Colour channels carry the same image feature information with varying chrominance intensities, which may be useful in a situation where the subject of ViDAQ involves acquiring measurements from physical indicators (dials) that convey color encoded information. In such cases, conversion to HSB (Hue, Saturation and Brightness) space would be more useful than grayscale. The ViDAQ prototype currently assumes dial indicators are of different shapes and lengths. Therefore grayscale reduction is suitable.

There are several RGB to grayscale conversion techniques. However, for this prototype, using the standard linear transformation produced acceptable results. Linear transformation uses a weighted sum of normalized (between 0 and 1) values of R,G,B intensities per pixel to yield a normalized gray (Y) value: $Y = k * R + l * G + m * B$ (where $k + l + m = 1$ and $k, l, m \neq 0$), that is finally scaled up to [0,255] range . The channel-specific non-zero weighing constants (k,l,m) are usually available in image processing literatures for example, $Y = 0.2989 * R + 0.5870 * G + 0.1140 * B$

Conversion to (black/white) binary image is also required to suppress unwanted information from the image. Binary images are usually obtained by applying thresholding to a grayscale image, which essentially replaces all pixel values greater than the threshold to ON (white [value = 255]) and others to OFF (black [value = 0]. Significance of Otsu's thresholding [75] for generating binary images is briefly discussed in previous section (2.3.2) and is also adopted in ViDAQ processing framework (Fig. 4.6a).

Feature Extraction

ViDAQ processing framework [134] (Fig. 4.6a) relies on the following feature extraction. Firstly, the profile of a circular dial face must be detected to allow the selection of the inner ROI for processing. Followed by which is contour and convex hull detection to detect the extremities of the envelope subtended by dial hands.

Dial Profile Detection

In order to detect a dial face in the image, Circle Hough-Transform (CHT) [80] is utilized (Fig. 4.6a,4.6b) to detect areas in the image that resemble circular dial boundaries. CHT is a specialized algorithm based on general linear Hough-Transform algorithm. Latter is a well known image feature extraction technique for detecting straight line edges in images and is also briefly discussed in previous section (2.3.2). CHT also relies on a voting procedure to bin points that accumulate high votes when they fall along the profile of a standard analytical shape (lines/circles/ellipse), that is being detected. For CHT[80], the transform parameter space is three dimensional owing to the three parameters involved: radius (r) and center coordinates (cx, cy) describing a circle $(x - cx)^2 + (x - cy)^2 = r^2$. The output of the CHT is the list of parameter tuples for each detected circle: $\{(r_1, cx_1, cy_1), ..., (r_n, cx_n, cy_n)\}$.

Once the dial face radius R and center coordinates (C_x, C_y) are known, rectangular ROI just enclosing the dials arms is found and used for further processing (Fig. 4.6b). ROIs, allows partitioning the image into smaller computational workloads to take advantage of parallelization and allow simultaneous reading of a variety of HMI devices in real-time. Specifically, Inner and Outer ROIs are found (Fig. 4.6b). While Inner ROI only includes the central dial face region containing only the dial needles, outer ROI only contains the shapes located along the periphery of the dial, such as digits and other markings. Outer ROI is essential to detect dial face orientation (Fig. 4.6b) by identifying a reference mark (E.g. locating $'12'$ on a clock), the orientation correction angle (ϕ) is found. Reference mark identification is accomplished by comparing the expected values of rotation invariant image moments (I_1) [140].

ViDAQ's [134] terminal feature extraction step is contour generation [141]. A standard implementation of Canny edge detector based contours generator [137] has been utilized in ViDAQ framework. Canny edge detector is a multi-stage adaptive algorithm that produces a set of points representing the edges of the dial needle shape present in the inner ROI (Fig 4.6b). It offers excellent signal to noise ratio, as it uses Gaussian smoothing to remove high-frequency noise with hysteresis thresholding to reduce false-positive edges. Several closed graphs [137] may be generated using the set of edge points, and contour is one such largest graph that covers the entire shape.

Image Segmentation

A convex-hull is the smallest polygon that encloses a group of objects. Image segmentation using convex-hull entails finding a set of straight-edge approximation of a closed graph (contour) covering the shapes with similar color intensity. Convex-hull is usually found using binary images or output of edge detected images [137]. The Canny edge detection algorithm used is used to generate contour curves that are then used to find the convex-hull using Ramer–Douglas–Peucker (RDP) algorithm [142]. RDP finds a minimal cover set of points that fully describes the contour (cartography). The convex-hull is essential in retaining only the extremities of the contour curve bounding the shape in the image. In ViDAQ prototype (Fig 4.6b) convex-hull edge (CvE) list facilitates resolving the dial needle endpoints (tips).

Measurement Extraction

Measurement extraction entails three steps: (1) identify the dial arm tips (endpoints) accurately; (2) determine the clockwise angle each dial makes with a reference segment (such as the 12 o'clock diameter segment) (3) convert the measured angle to required measurement quantity - for example, we assume that the longest dial arm represents seconds dial and shortest dial as hour.
ViDAQ algorithm (Algorithm 2) takes the list of endpoints of the outer edges of convex hull bounding the dial shape and then sorts it in descending order of the CvE lengths (line segment connecting a pair of convex hull points). Sorting ensures longer edges that bound the dial extremity points (tips) are selected first while ignoring other extremities, that often occur in watches where the dials have tails. The sorted list of CvE typically contains two types of edges as illustrated in Fig 4.7 : radial type - edges that may extend from tip of a dial to a point closer to the center of the dial (cx, cy); secant type - edges that extend between tips of one dial to the other dial. In order to accurately recognize and capture the dial tips, the algorithm discerns the given edge as either a radial or secant type (Fig 4.7). Followed by, creating a unique list of dial tip candidate coordinates based on the rule(κ).

Rule κ**.** *Given a point of reference $C_{x,y}$, If Convex-hull Edge (CvE) $E_i(s, e)$ is radial type, then only include the far point (either s or e farthest from $C_{x,y}$) (if not included previously), Else If E_i is secant type, then add both start and end points (if not added previously).*

Finally, time value measurement for each dial (hour, minute and second) is done by extracting the clockwise angle the dial needle makes with respect to the 12 o'clock reference segment. The angle is computed using the dot product rule 4.1 while the sign of the determinant 4.2 between two 2-D vectors helps determine either the angle being measured

is clockwise or anti-clockwise angle (specifically, if $det(\overrightarrow{v}, \overrightarrow{w}) > 0$ then $\overrightarrow{v}$ is immediately clockwise of $\overrightarrow{w}$)

$$\Theta = cos^{-1}(\frac{\overrightarrow{v} \cdot \overrightarrow{w}}{\| v \| \| w \|}) \tag{4.1}$$

$$det(\overrightarrow{v}, \overrightarrow{w}) = v_x.w_y - v_y.w_x \tag{4.2}$$

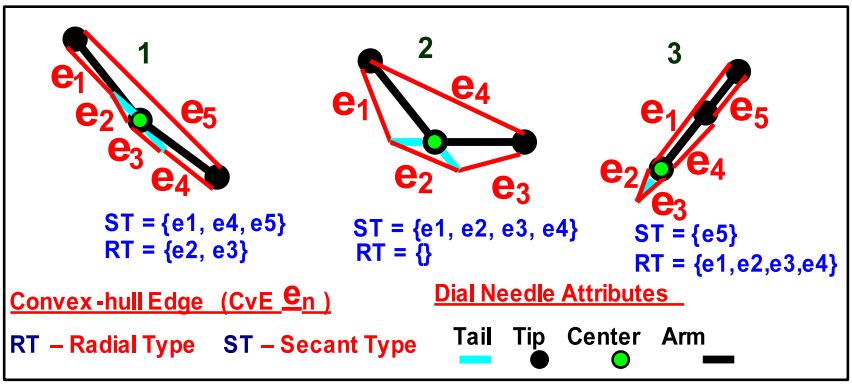


Figure 4.7: Radial and Secant Type CvE (convex-hull edges), under 3 typical orientations: 1 - dials maximally open, 2 - semi open dial tips & tails, 3 - minimally open (dial overlap)

Algorithm 2: Dial Tip Detection Routine

procedure FINDDIALTIPS(τ,R_d,$C_{x,y}$) ▷ τ: Convex-hull Edge(CvE) selection parameter, Dial face Radius and Center: R_d, $C_{x,y}$
Initialisation:
  $dial[] \leftarrow \{\}$ ▷ init. list of selected CvE as dial arm candidates
  $DT[] \leftarrow \{\}$ ▷ init. list of DialTip coordinates.
  $hull[] \leftarrow \{E_1, E_2, \ldots, E_n\}$ ▷ get list of CvE(s), where $E_i\{s, e\}$ has start/end cord. tuple.

Convex-Hull Edge Selection:
  for all $(E_i \in hull[])$ **do**
    $d \leftarrow Lenght(E_i\{s\}, E_i\{e\})$ ▷ Compute the length of CvE E_i
    if $(\tau * R_d \leqslant d < R_d)$ **then** ▷ criteria for good dial candidates
      $dial[] \leftarrow \{d, E_i\{s, e\}\}$ ▷ Select this CvE E_i and its length d

Sort dial[] List:
  $ReverseSort(dial[], \{d\})$ ▷ sort hull[] in descending order of each tuple $\{d, E_i\{s, e\}\}$ using length d

Dial Tip Selection Logic as per rule κ:
  for all $(D_i\{d, E_i\} \in dial[])$ **do**
    if $(RadialEdge(E_i\{s, e\}, C_{x,y}))$ **then** ▷ Checks if E_i is Radial CvE given $C_{x,y}$
      $DT[] \leftarrow UniqueFarPoint(DT[], E_i\{s, e\}, C_{x,y})$ ▷ returns farthest point of CvE ($E_i\{s\}$ or $E_i\{e\}$) w.r.t given center $C_{x,y}$, that is $\notin hull[]$
    else if $(SecantEdge(E_i\{s, e\}, C_{x,y}))$ **then** ▷ Checks if E_i is Secant CvE given $C_{x,y}$
      $DT[] \leftarrow SelectTips(E_i\{s, e\})$ ▷ Select points of CvE ($E_i\{s\}$ and/or $E_i\{e\}$) that are $\notin hull[]$
    else
      $DT[] \leftarrow null$ ▷ Reject E_i, as its neither Radial nor Secant type
  return $DT[]$ ▷ Final list of Dial Tips

4.2 Supervised Learning - InS Vs. OuS Dataset

All RNN (LSTM and CNN) time-series and NLP Discrete Event HMI state forecast models trained and evaluated herein, use the supervised training data framing as shown in Fig. 4.8. This supervised data sequence presents the raw column data as an input sequence of feature vectors ($\{X\}^k$) and a corresponding output (y^n) sequence. The original raw dataset (Fig. 4.9) collected from HMI state model is formatted in a *k-lagged* / *n-step* ahead format to obtain a sliding window training set consisting of sequences of raw samples arranged in a row of samples. Where each sequence row consists of a set of raw input feature values from *k-lagged* past time steps and corresponding raw output feature values from future *n-step* ahead samples as arranged in the supervised data frame (more detail frame shown in Fig. 5.11 in Sec. 5.3).

Transformation of raw dataset (Fig. 4.9) to supervised data frame is discussed in detail in Sec. 5.3.2 in **Experiments and Results** chapter.

Initially a HMI State model (M) is trained and validated sequence using k lagged HMI state vector $\{X\}^k$ samples with corresponding expected future n-step ahead target y^n samples of output HMI state feature vector.

During model testing or final evaluation the trained model (M) is expected to output *n-step* ahead values of the output feature ($\hat{y}^n$) samples, when it is given out-of-sample (values from outside initial training set) $\{X_{OuS}\}^k$ as input.

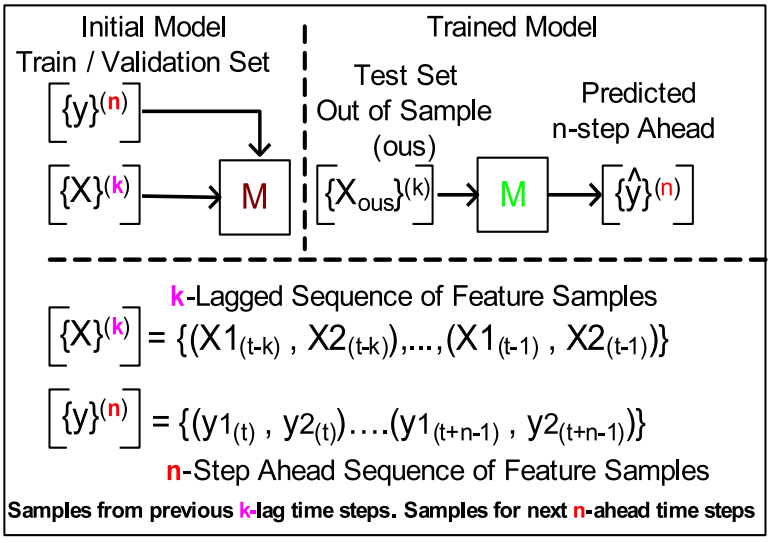


Figure 4.8: Raw HMI state feature time-series framed into k lagged and n-steps ahead sequence of samples. HMI state vector $\{X\}^k$ captures both process indications and user/operator input values. Initial model is trained with k lagged HMI state vector $\{X\}^k$ samples with corresponding expected future *n-step* ahead y^n samples of HMI state feature vector. The trained model can produce forecast $\hat{y}^n$ samples.

Time,	PROCESS,	HMI_USER
1-Jan-2000,	97,	182
2-Jan-2000,	97,	159
3-Jan-2000,	97,	110
4-Jan-2000,	97,	107
5-Jan-2000,	135,	113
6-Jan-2000,	135,	134
7-Jan-2000,	135,	135
8-Jan-2000,	135,	135
9-Jan-2000,	157,	147
10-Jan-2000,	157,	152
11-Jan-2000,	157,	155
....	....	...
....	....	...

Figure 4.9: Example snippet of a raw HMI time-series dataset. Currently restricted to contain only two features: *PROCESS* and *HMI_USER* with their values restricted to 8-bit unsigned integers in the scope of this research experiment(s). *PROCESS* is used to capture any one analogue process parameter from a HMI model, and *HMI_USER* can be used to capture maximum of 8 digital input/outputs: HMI lamp or switch status combinations.

InS prediction refers to generating next step forecast values when current input to the predictor model is drawn from the training set that was initially used to fit (train) the model. It is generally not necessary to re-train the model for doing *InS* prediction.

OuS refers to generating next step forecast values when current input sequences ($\{X\}^k$) to the predictor model is drawn from a new data set that the model was not trained on initially. The model may be re-trained progressively using past *OuS* samples to accomplish *OuS* prediction more accurately.

Both *InS* and *OuS* forecast capability have their own utility and limited accuracy trade-offs in TS modeling for *EYE-on-HMI* framework. In that, *InS* forecast may be used when current input data to predictor model is not expected to vary from the original training set (E.g. a manufacturing process parameters that tend to vary within a tight band). *OuS* may be used when current data is expected to vary within a broader range of values, for example, in traffic flow, road conditions, weather patterns, etc. In this study, LSTM and CNN model testing is done with an *OuS* test data set. ARIMA (regression-based) models were also tested with *InS* test data set to compare performance using synthetic HMI state data set.

In the context of this research regarding the modeling of control room HMI States. It is useful to exploit the trade-off between model generalization cost vs. available training dataset size. The observation regarding specific HMI parameters that do not vary over a large window of time (Sec 1.2), for example, parameters like temperature and pressure setpoint alarm lamps, valve position hand-switch, etc. will generally stay in their normal state. Such HMI parameters may be modeled using *InS* data sets, and don't need to be generalized, thus require smaller training and validation datasets.

While other fast-moving HMI parameters such as boiler level, feed-water flow fluctuate over a broader band. For such parameters, it may be prudent to train the HMI state model using

an *OuS* test data as the model needs to be generalized, thus require a more extensive training data set.

4.3 HMI Time-Series Modeling using ARIMA(p,d,q)

This work envisions an Expert Supervisory System (EYE) [7] that will allow non-intrusive monitoring of human-machine interfaces of any conventional operator based command - control - feedback based architecture (Fig. 1.2-B). The proposed approach entails using the data collected by monitoring HMI state patterns (Fig. 3.2), which can be used to create time-series (TS) forecast models for predicting HMI states few steps ahead. The idea is to use near future predicted HMI states, which include both process and human inputs to validate the user inputs in response to current process states and detect if any deviation occurs. Such deviations can then be trended and alarmed as a possibility of specific human errors before a human error is committed.

As a first-order approach to model HMI state time-series patterns, Box-Jenkins [87] or ARMA(p,q): Auto-Regressive (AR) and Moving Average (MA) model (4.3) is evaluated. These have been widely used and help to understand the nature of TS patterns. The AR(p) tries to model the hidden Markov nature in the data (i.e. current values depend on previous values), and MA(q) tries to model the effect of a moving window of past noise contributions.

AR(p)and MA(q) Models

$$x_t = \mu + \sum_{i=1}^{p} \alpha_p x_{t-p} + \epsilon_t \text{ where p is AR lag parameter}$$

$$x_t = \epsilon_t + \sum_{i=1}^{q} \beta_q \epsilon_{t-q} \tag{4.3}$$

where q is MA window parameter and MA(q) is always stationary for all $q < \infty$

Time-series analysis using the application of *ARMA* models first require the given raw TS to be converted to a stationary TS. In reality TS data may include well known characteristics such as *Trend*, *Seasonality* and *Noise* which violate the required stationary conditions (3.2). Classic TS analysis fundamentally attempts to identify and decompose the original TS into these various components, model these separately then subtract them from the original TS to isolate the resulting stationary pattern (signal). It is the residual signal that ultimately must be sought and modeled for forecasting. *Trend* refers to the steady direction of the series

movement i.e., either upwards or downwards. The *Trend* component can be removed by either using differencing, windowed mean, or logarithm in case TS exhibits non-constant variance, etc. *Seasonality* refers to repeated cycles which may also be removed by doing differencing between same time points within each cycle. *Noise* can often be removed by various low pass filtering techniques. A TS can be differenced with its several lagged versions of itself to do de-trend (or make trend stationary). Therefore, sometimes *ARMA(p,q)* model also includes the a differencing or Integration parameter (d), which collectively then is referred to *ARIMA(p,d,q)* model.

4.3.1 ARIMA - Tools for Checking Stationarity

The standard tools utilized for checking, whether the given time-series data is stationary, includes visual methods of assessing TS nature and few statical value tests.

The visual methods include generating plots such as Histograms, Density plots, Auto-correlation Function (ACF), and Partial Auto-correlation function (PACF) and *Q-Q* plot. The *Histogram* and *Density* plots show the distribution of data around the mean and how much variance (spread) there is in the data from its mean. For a weakly stationary TS, a normal Gaussian distribution (bell shape) curve is expected. As this shape indicates, a majority of samples are concentrated near the mean and, its narrowness indicates variance is low, which rather indicates the mean and variance to have little dependence on time as required by equation 3.2.

The *ACF* plots the auto-correlation coefficient values as a function of lags ($s > 1$) between the time-series (X_{t-s}) and itself (X_t), which is used to visually determine the appropriate order of the moving average parameter *q* of the *MA(q)* (4.3) process that will closely model the given TS. Moreover, *ACF* also helps to distinguish a purely *AR(p)* from a *MA(q)* process based on the rate of decay (gradual vs. abrupt) of the *ACF* values with respect to lags. That is, if the rate of decay is gradual, it's indicative of a *AR* process rather than a *MA*, which is when ACF decays abruptly. Lastly, periodic spikes in *ACF* is indicative of seasonal component.

Whereas, *PACF* shows the conditional auto-correlation function that controls the effect of all intermediate lags to be removed

$$PACF = Correlation[(X_t, X_{t-s})|(X_{t-s+1}, X_{t-s+2}...X_{t-1})]$$

This aids in detecting order of the parameter *p* of the *AR(p)* auto-regression process. The statistical test most commonly used to test stationariness of a TS is the Augmented Dickey-Fuller Test (ADF) [93]. This test is based on evaluating the null hypothesis that a unit root is

present in the given TS data, which implies the TS is not stationary.

Using the terminology of TS analysis, the current scope of experiment setup models a classical causal system. Where *HMI_USER* is considered as the *endogenous* variable, which is influenced by other factors in the test system, i.e., it may depend on process values that the user must track. For instance Fig. 5.11, feature *PROCESS* variable is considered an *exogenous* variable that is (independent) not affected by other variables in the system, i.e. user input has no influence over the process values. ARIMA models can use exogenous variables to improve the long-range out-sample forecasts.

4.3.2 ARIMA - Model Parameters

For the scope of this study a *ARIMA(p,d,q)* and seasonal *ARIMA(p,d,q)(P,D,Q,S)* statistical models were evaluated. The hyper-parameters of these models is determined using an exhaustive multi-variate grid search algorithm. The algorithm iterates over all combinations of each parameter (within a specified range) while re-fitting the ARIMA model every iteration over the training set. Finally saving the set of hyper-parameter values for AR(p) (auto-regression), MA(q) (moving average) and differencing (d) which yields the lowest mean square error (MSE) for predictions generated by the model on the validation data set. Model over-fitting is reduced by doing cross-validation checks that are performed by running the grid-search with a sliding window method, i.e., the training and validation data sets are segmented using a *500-sample* sliding window. That was advanced to generate different training and validation data subsets over the complete available data set for use with the grid search routine.

All ARIMA models were developed in Python 3.5.4 (Anaconda distribution) and using the $Statsmodels.Tsa.ArimaModel$ library. Time-series analysis auto-correlation plots were generated using the $Pandas.tools.plotting$ library.

4.3.3 ARIMA - Prediction Mode and Models

The goal of this study is to evaluate the performance of ARIMA models to do n-step ahead in-sample (*InS*), and out-of-sample (*OuS*) forecasts modes. *InS* prediction refers to generating the next step forecast values when current input to the predictor model is drawn from the training set that was initially used to fit (train) the model. While *OuS* refers to generating next step forecast values when current input value to the predictor model is drawn from a new data set that the model was not trained on initially.

As stated previously (Sec. 5.3.2) both *InS* and *OuS* forecast capability has its own utility and accuracy outcome in TS modeling for *EYE-on-HMI* framework. In that, *InS* may be used, when expected variation between the evaluation data set patterns and the original training

set is marginally low (E.g. few manufacturing process parameters such as temperature and pressures may vary little within a tight band in reality). *OuS* may be used for the case when actual current data is expected to vary within a broader band compared to the initial training data set (E.g. certain HMI feature parameters may vary cycle through various different patterns).

the *InS* vs. *OuS* forecast models also yield a generalization vs. available training data size trade-off. In that, the forecast models appropriate for using *InS* evaluation impose less restriction on being generalized and may be trained using smaller datasets in a shorter time. In contrast, models that require higher generalization skills across various patterns will need *OnS* evaluation data sets with k-fold cross-validation during training, requiring longer training times.

Few variations of forecasting tests that were evaluated using *ARIMA(p,d,q)* model and *ARIMA-X(p,d,q)* (with exogenous variable) model that were fitted on the test HMI *auto-pilot* data set. The following lists various flavours of prediction tests

1. *Persistence model*: This is also referred to as the *naive* forecast that is used to ascertain the baseline forecast error estimate for the given time series data. The Persistence algorithm uses the lagged version of the data series to simply output the actual data value at the next time step ($t + 1$) as the expected outcome given the current (t) actual value. Therefore, if subsequent samples in time series data are closely correlated, it will yield a lower root-mean-square (RMSE) forecast error using the persistence forecast ($RMSE_p$). The $RMSE_p$ value can then be used as a upper bound for forecast error or selection criteria for any candidate forecast model to be considered useful, i.e. the model must yield a lower value of forecast RMSE $< RMSE_p$. (More detail about the dataset and related persistence score is provided later in Chapter 5 Experiments and Results.)

2. *Static model*: When *ARIMA(p,d,q)* is fitted (trained) once on a training data set and then given an input sample ($t = k$) of the endogenous variable that is modeled and to be forecasted. The static model in *Standard mode* can be used to make n-step ($t = k + n$) ahead predictions based on only the given sample at $t = k$, i.e. the in-sample observed lagged values are used for prediction. *Static model* is generally faster at forecast generation and fitting. However, the downside is that the accuracy of the forecast is an inverse function of the variance of the actual input data sample in reality compared to the initial training set (i.e. larger the variance, lower the forecast accuracy).

3. *Dynamic mode*: Similar to above *Standard mode*, the *Static model* in the *Dynamic mode* (*ARIMA(p,d,q)*) is initially fitted on a training set. However, during forecast generation,

previous In-sample forecasted values fed back in place of actual InS values to predict the next step ahead forecast. The model is dynamic, i.e. it adjusts to the previous forecast values and is used to make n-step ($t = k + n$) ahead prediction values. In other words, Dynamic mode tries to overcome the forecast bias towards initial training data that is seen in the *Standard Mode*. Where the later mode uses lagged values of the original training set to predict the next step forecast. Forecast errors tend to propagate and amplify for every n-step ahead forecast run.

4. *Adaptive model*: When *ARIMA(p,d,q)* is re-fitted (re-trained) on new data set obtained by appending the previous ($t = k - n$) actually observed sample value(s) to the original training set. The retrained model has the ability to adapt to on-line using actual observed values while forecasting n-step ($t = k + n$) ahead predictions. However, the downside of this approach is model needs to be re-fitted before generating every forecast, and the training set will grow in size over time, making the training time successively longer.

4.4 HMI Time-Series Modeling using RNN and CNN

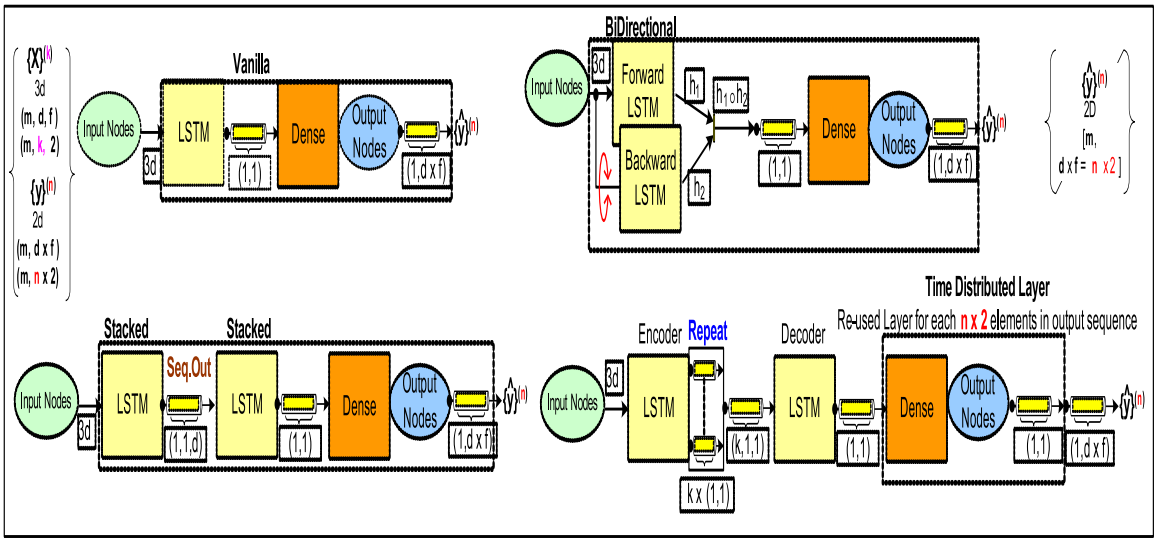
The goal of this study is to evaluate performance of out-of-the-box (unaltered) LSTM [143] and CNN models to do, *n-step* ahead in-sample (*InS*) and out-of-sample (*OuS*) forecast modes for predicting *HMI_USER* (operator response) given *k-lagged PROCESS* indication state vectors as the input (Fig. 5.11).

(**InS** and **OuS** in this context is common to the terminology used previously for ARIMA model testing in Sec. 4.3.3)

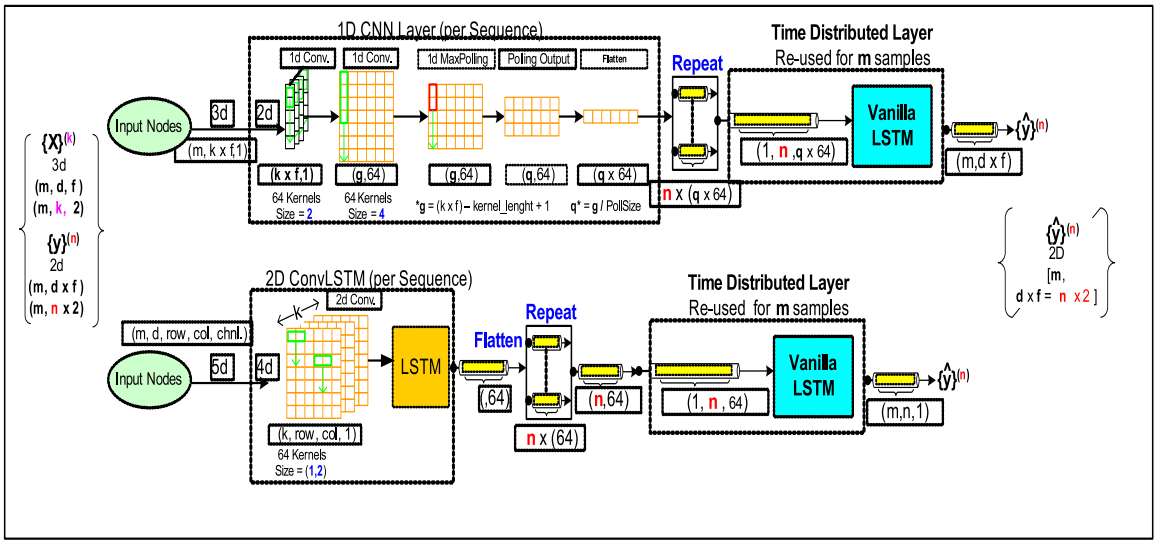
4.4.1 RNN and CNN Model Designs

This section briefly summarizes design highlights of various out-of-the-box (unaltered) RNN LSTM model architectures that are evaluated here-in on the synthetic HIM data set(s). All LSTM model archetypes were inspired by various architectures introduced previously by other works [143, 144, 145, 146, 147] and were implemented using *Python* 3 *Keras 2.1.3* API libraries with *TensorFlow 1.5* as its back end implementation.

RNNs differ from other classic feed-forward neural networks by relying on feed-back of the hidden layer states, accumulated from previous ($t - n$) time-steps, along with new inputs. In addition with current input to generate $t + 1$ output. Hence, RNN are apt in capturing temporal structures in time-series data better. LSTMs further utilize memory cells and forget gates to overcome the exploding and vanishing gradient problems with RNNs. Therefore,



(a)



(b)

Figure 4.10: (a) Out-of-the-box (unaltered) LSTM Network architectures: Vanilla, Stacked, Bidirectional and Encoder-Decoder. (b)Out-of-the-box 1d-CNN Encoder LSTM and Convl.LSTM Network. Input features HMI state vector $\{X\}^k$ samples data set is transformed into 3d shape $(Samples = m, Time - Steps = d, Features = f)$. Expected output (Ground truth) is y^n HMI state sequence vector, while predicted forecast is $\hat{y}^n$ samples.
[currently, m corresponds to number of rows in data set, $d = k$ or $d = n$ is sequence length for either input or output sequence respectively, $f = 2$ is number of features].

LSTMs are even more apt at learning longer temporal dependencies (casualties) in complex sequences.

All input k lagged sample vectors $\{X\}^k$ to LSTM models, must be reshaped as 3d matrix consisting of m sample rows of sequences. Where each sequence, dimensionally specifies

d time-steps and f features i.e. $(Samples = m, Time - Steps = d, Features = f)$. In current experiment, Time-Steps $d = k$ lag and features $f = 2$, since we have *PROCESS* and *HMI_USER* as two feature vectors in current data set, therefore the training and testing input $\{X\}^k$ vectors are presented as $(m, k, 2)$ dimension matrix. The corresponding expected output n time-steps ahead vector $\{y\}^n$ samples must be reshaped as 2d matrix consisting of m samples of output sequences $(m, d \times f)$ or $(m, n \times 2)$. Data set sample values are normalized to $0 - 1$ range prior to model training, validation and testing, then transformed back to original range for calculating the $RMSE$.

(i) **Vanilla:** This is a basic sequential LSTM network architecture consisting of the input layer, one layer of LSTM cell(s) followed by a dense output layer. The output of each LSTM cell is a single valued 2d matrix $(1, 1)$ for each input sequence of time-steps (each sample row). The Dense or output layer is a fully connected network of neural nodes that applies the final activation function and reshapes the model to desired output dimension, $\{\hat{y}\}^n$ as 2d vector $(1, TimeSteps(n) \times 2)$ for each sample row in input $\{X\}^k$ matrix.

(ii) **Stacked:** A standard sequential network consisting of the input layer followed by one or more LSTM layers followed by the (dense) output layer. The first LSTM layer is configured to output a sequence of values for each time-step (d). Thus, a 3-d vector is generated as required by subsequent LSTM layers as input. Only two stacked-layer LSTM network was evaluated in the current experiment.

(iii) **Bidirectional [144]:** Is a standard sequential network consisting of an input layer followed by two parallel LSTM layers: forward and backward. The forward LSTM layer is normally trained as in Vanilla model. However, the backward LSTM layer is trained by feeding it each input sequence of time-steps in reversed order. The outputs of both these LSTMs is concatenated before inputting it into the output layer.

(iv) **Encoder-Decoder [145]:** Is a standard sequential network consisting of an input layer followed by an encoder LSTM layer, which encodes the relationships between the k time-steps for each f features in the input sequences as a fixed-length internal representation (hidden state) vector. One or several LSTM cells can be used as an encoder layer to extract salient features of the input sequence samples. The 2d output matrix $(1, 1)$ of the encoder is configured to repeat for each k time-steps to build the 3d matrix of dimension $(k, 1, 1)$ to be fed to the next decoder LSTM layer. The decoder LSTM maps per sequence representation to a single-valued 2-d matrix, which gets fed to the

$Keras$ time distributed (TD) dense layer. The TD common dense layer is reused for all $d \times f$ or $d \times 2$ number of elements required in the output vector $\{\hat{y}\}^n$.

(v) **1d-CNN [146]:** Like the Encoder-Decoder LSTM, a CNN-LSTM uses the CNN layer as an encoder that produces a subjective interpretation, as a result of performing various convolution operation on the input samples.

Typically Convolutional Neural Network layer uses convolution operation on 2d data sets with spacial information such as pixel values for each color channel in image and video input to extract salient features. The 1d CNN layer is trained per sequence, and the input to it is reshaped to a 3d matrix of $(m, k \times f, 1)$, and fed as 2d matrix $(k \times f, 1)$ for every sequence sample.

The k-lagged/n-ahead sample sequences in each supervised training batch have been generated in a rolling window fashion (Fig. 4.8) containing $k + n - 2$ duplicate tokens between each consecutive sample sequence as discussed in following **Experiments and Results** chapter Section. 5.3. Repetition of token values is likely to generate some spatial order within the batch of sequences, among other local patterns between HMI event tokens in this format. Therefore, convolution operators are also evaluated to see if any such spatial features may be existing in the dataset that can be useful for learning.

For time series prediction, a sequential network is built of the input layer followed by 1-d convolution network to extract features in the input sample sequences. In this experiment, two consecutive 1d convolution layers with 64 kernel filters (feature detectors) of sizes 2 and 4 respectively were utilized. Since, the kernel window slides one step at a time, the output dimension of 64 filters combined is $(g, 64)$ where $g = (k \times f) - kernelSize + 1$.

A max polling of size two was utilized to reduce the over-fitting of training data by CNN output. The max polling window steps (rather than sliding) through the final kernel output matrix to output a matrix with dimension $(q, 64)$ where $q = \frac{q}{PollSize}$.

The output of the max polling is flattened into a row vector, which is repeated n times, for each n-time steps ahead as required by expected predicted output $\{\hat{y}\}^n$.

Finally, a Vanilla LSTM network wrapped inside the $Keras$ time distributed (TD) dense layer is utilized to learn a sequence of 1d CNN output vectors. The TD makes the same Vanilla LSTM network to be reused for all m number of samples in the input vector $\{\hat{y}\}^n$

(vi) **Convl.LSTM [147]:** Convl.LSTM or convolutional LSTM is slightly different from CNN-LSTM, in that convolution results over input data is directly fed to the LSTM cell

rather than an internal representation of a neural network layer. convl.LSTM performs 2-d convolution over the input data.

A 2-d convolution expects the initial training and test input data $\{X\}^k$ vector matrix $(m, k, 2)$ to be presented as a k time-step sequence of image (shaped array) of size (row, col, $channels$). In this experiment, following were set as, $row = 1$, $col = 2$ and $channels = 1$ to yield a 5d input data matrix $(m, k, 1, 2, 1)$. The corresponding expected output n time-steps ahead vector $\{y\}^n$ samples is reshaped as 3d matrix consisting of m samples of output sequences with dimension (m, d, f), but containing only one output feature (*HMI_USER*) to be predicted, in-order to reduce training time ($\because$ final output dimension is $(m, n, 1)$).

The convl.LSTM output is a flattened row vector as an internal representation for each input sequence of 3d arrays, which is repeated n time-ahead output vector $\{\hat{y}\}^n$. The Vanilla LSTM network that is wrapped inside the $Keras$ time distributed (TD) dense layer finally learns the output of convl.LSTM layer every sequence sample and outputs m samples for every input vector $\{\hat{y}\}^n$ sequence sample.

4.5 HMI NLP Modelling using *Seq2Seq* RNN Model

Sequence-to-Sequence (*Seq2Seq*) represents a class of machine learning problems that entail model training to generate a fixed-length output sequence of symbols when given a fixed-length input sequence of symbols. There are no restrictions placed on any particular length of either input or output sequences. An *Encoder-Decoder(EncDec)* architecture is adept for *Seq2Seq* class of problems E.g. NLP (machine translation), image captioning, sentiment analysis, etc.

In the general case of machine translation using *Encoder-Decoder(EncDec)* architecture (Fig. 4.11a), during the model **training** mode, entire input sequence of one language ($\mathcal{L}_1$) is required along with the output sequence of the target language ($\mathcal{L}_2$). However, during translation (**inference** mode) each previously predicted symbol is required to be fed forward to *decoder* input to produce subsequent symbols until a sequence end token is produced.

In case of a *LSTM-EncDec* model (Fig 2.5) the *encoder* LSTM layer functions to produce a fixed size internal representation (summary vector) of the given input string of symbol sequence of any length truncated by a special $< End >$ token (it triggers decoder to start translating). The summary vector is obtained as a result of accumulated hidden states from each LSTM cell in the *encoder* hidden layer as a result of processing the entire input sequence of symbols. Therefore, the final summary vector consists of hidden state and cell output

$< (h, c) >$ from the last *encoder* LSTM cell. The LSTM *decoder* uses the summary vector to initialize its first cell state, with the desired effect of incorporating contextual information representing the entire input sequence that aids in predicting the next translated symbols.

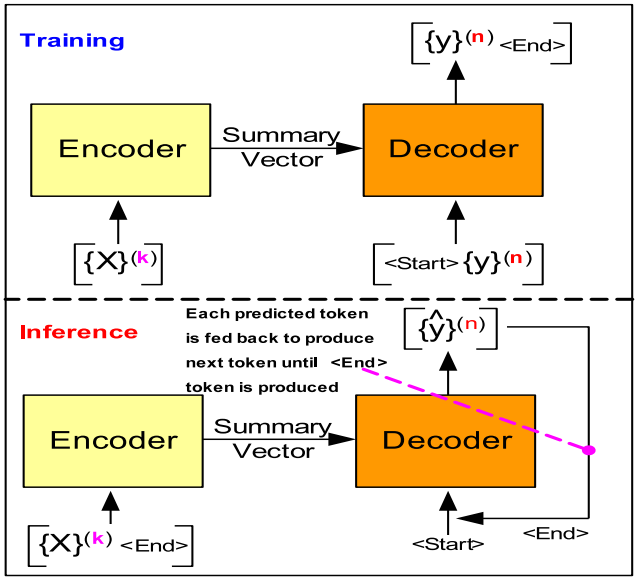
Table 4.1: Parameter Nomenclatures for NLP LSTM and CNN Models

Parameter	Description
(k) or (K)	Fixed time-series window of size k, containing k-lagged samples from past time steps $T = t - k...t - 1$.
(n) or (N)	Fixed time-series window of size n, containing n-ahead samples expected in future time steps $T = t + 0...t + n - 1$.
$[X]$, $[y]$	$[X] =< X1, X2 >$ represents an HMI DES event, containing two one-hot encoded feature vectors: $X1$ - HMI indication value vector, $X2$ - User input value vector. Where, each $X2, X2 \in \mathcal{R}^D$, D is the fixed dimensionality (dictionary size) of HMI DES event space (currently set to 255) including the <$start$> token $[y] =< y1, y2 >$ is same in construction as $[X]$, but is used to denote the expected HMI DES event.
$\{X\}^{(k)}$	k-lagged sequence of HMI indication ($X1$) and user input ($X2$) events containing samples of feature vectors used as model training input pattern. $\{X\}^{(k)}$ = $\{(X1_{t-k}, X2_{t-k})$,...,$(X1_{t-1}, X2_{t-1})\}$ is a DES HMI state vector of $\mathcal{L}_1$.
$\{y\}^{(n)}$ $\{\hat{y}\}^{(n)}$	n-step ahead sequence of HMI indication ($y1$) and user input ($y2$) events containing samples of feature vector values used as model training target output pattern (ground truth). $\{y\}^{(n)}$ = $\{(y1_t, y2_t)$,...,$(y1_{t+n-1}, y2_{t+n-1})\}$. $\{\hat{y}\}^{(n)}$ is the expected output event sequence or translated HMI DES state vector of $\mathcal{L}_2$.

4.5.1 LSTM-EncDec - Training Phase

Supervised training of *LSTM-EncDec* is to maximize the probability ($logP(\{y\}^{(n)}|\{X\}^{(k)})$ (4.4) of generating the target *n*-ahead samples of target HMI event sequence ($\{y\}^{(n)}$) given the entire previous *k*-lagged samples of input sequence ($\{X\}^{(k)}$) context. This is done by the learning (or optimizing) the RNN parameters (ϕ) of the *decoder* in *LSTM-EncDec* model to maximize the *log* probability of each target HMI event token (y_t) given a fixed summary vector (h_k) or the hidden state of final *encoder* cell. Where, h_k (4.6) is a non-linear function of each input event (token) ($\{X\}^{(k)}$) and corresponding previous hidden states (h_{t-1}) from other *encoder* RNN/LSTM cells.

The *LSTM-EncDec* model for current HMI DES *Seq2Seq* application is trained by using a method referred to as *Teacher forcing* (4.5) as shown in Fig. 4.11a, which is a dynamic supervised training task with input/output sequence pairs being: ($\{X\}^{(k)}$/$\{y\}^{(n)}$) from source and target HMI DES languages ($\mathcal{L}_1$, $\mathcal{L}_2$) respectively, to jointly train the *encoder-decoder* system.



(a) Training Vs. Inference of *Seq2Seq* Encoder-Decoder Models

INPUT1 : PROCESS[T-4], PROCESS[T-3] PROCESS[T-2] PROCESS[T-1]....
INPUT2 : 255 HMI_USER[T-0] HMI_USER[T+1] HMI_USER[T+2].....
OUTPUT : HMI_USER[T-0], HMI_USER[T+1] HMI_USER[T+2] HMI_USER[T+3]....

$INPUT1 : \{X\}^{(k)} = \{(X1_{(t-k)}, X2_{(t-k)}), (X1_{(t-k-1)}, X2_{(t-k-1)}), (X1_{(t-k-2)}, X2_{(t-k-2)}) \ldots, (X1_{(t-1)}, X2_{(t-1)})\}$

$INPUT2 : \{y\}^{(n-1)} = \{255, (y1_{(t)}, y2_{(t)}), (y1_{(t+1)}, y2_{(t+1)}) \ldots\ldots, (y1_{(t+n-1)}, y2_{(t+n-1)})\}$

$OUTPUT : \{y\}^{(n)} = \{(y1_{(t)}, y2_{(t)}), (y1_{(t+1)}, y2_{(t+1)}), (y1_{(t+2)}, y2_{(t+2)}) \ldots\ldots, (y1_{(t+n)}, y2_{(t+n)})\}$

(b) Teacher Forcing - Training Data Stream

Figure 4.11: (a) Encoder-Decoder Model are trained using *Teacher forcing* - decoder input is replaced with ground truth tokens (results in exposure bias). During inference previous decoder output is fed forward as input to output next token; (b) $INPUT1$ is the encoder training input which is fed as the source sequence. *Teacher forcing* training data stream requires decoder input ($INPUT2$) and expected ground truth output ($OUTPUT$) be offset by 1 time-step by inserting a start of sequence ($< Start >$) marker token (255). Optionally a $< End >$ marker may be inserted in training output sequence ($OUTPUT$).

Where, $\{X\}^{(k)}$ is framed as a sequence of HMI events from k time steps in past and $\{y\}^{(n)}$ as target HMI sequence of events *n* time steps-ahead in future. Under *Teacher forcing*, a RNN based *seq2seq* models is trained by replacing the previous predicted output of a *decoder* cell y_{t-1} by the actual (ground truth) or *teacher* supplied value as input to subsequent cells for

learning. That is, the feed-forward links between *encoder* RNN cells is bypassed by injecting *teacher* supplied signals. This is analogous to a *teacher* who corrects the student at every step of a task sequence; instead of allowing the student to complete the entire task sequence fully and then learn from her mistake.

Specifically in this design, sequence of one-hot encoded vectors are constructed for *encoder*: INPUT1- $\{X\}^{(k)}$, *decoder*: INPUT2- $\{y\}^{(n-1)}$ (suffixed with a $< Start >$ token) and *decoder*: OUTPUT- $\{y\}^{(n)}$ (as expected target output) as depicted in Fig. 4.11b. The sequence of one-hot encoded vector corresponds to a sequence of events (a.k.a forms *sentences* from $\mathcal{L}_1$ and $\mathcal{L}_2$ languages) that are HMI DES states. Each one-hot encoded vector has a fixed dimension (255 bits) that is same as the dictionary event size and represents a particular value of the HMI indication feature(s) $(X1, X2)$. The output of *decoder* is the target translated *sentence* in $\mathcal{L}_2$ that corresponds to the predicted next state of HMI DES, given the previous HMI state presented as INPUT1/INPUT2 to *encoder* and *decoder* respectively. One caveat with INPUT2 is that it is offset by one time step compared to expected OUTPUT sequence owing to inclusion of a $< Start >$ suffix token, which triggers the *decoder* to begin translation. Note, that the required $< End >$ token is not included as it is implicit in this design by having a fixed length input and output sequence(s) (However, this is not a hard restriction as the model is able to support variable length sequences).

LSTM-EncDec Model

$$\begin{aligned} & logP(\{y\}^{(n)}|\{X\}^{(k)}) \\ &= \sum_{t=0}^{N} logP(y_{t+1}|y_t, X; \phi) \end{aligned} \tag{4.4}$$

Where $N = n - 1$, $y_0 =$ <Start> token

Teacher Forcing Training

$$logP(y_{t+1}|y_t, X; \phi) = logP(y_{t+1}|h_t; \phi) \tag{4.5}$$

Hidden State or Context Vector

$$h_t = \begin{cases} f(X; \phi), & \text{if } t = 0 \\ f(h_{t-1}, y_{t-1}; \phi), & \text{otherwise} \end{cases} \tag{4.6}$$

4.5.2 LSTM-EncDec - Inference Phase

The *LSTM-EncDec* model once trained using *Teacher forcing* may be used to infer (Fig. 4.11a) the expected HMI event(s) of the target HMI state (translated language pattern) $\hat{y}^n$ one step at

a time. Translation begins once the $< Start >$ is fed to the first *decoder* cell whose internal cell state has been initialized by the summary vector from the *encoder*. Subsequent, target event tokens are inferred by feeding as input, the partial target HMI event pattern that is built by appending previously predicted event tokens to it. The downside of this is if any previous inferred token is incorrect, subsequent predictions will be off.

Moreover, the basic *LSTM-EncDec* model has also shown to be limited [148] in its translation skill for longer length sequences, owing to a bottleneck associated with the final fixed-length summary vector that only serves as a coarser context of the source pattern for the *decoder*. *Attention* mechanism has been introduced to overcome this problem of coarse context.

4.5.3 RNN Attention Mechanism

The encoder-decoder *Seq2Seq* RNN based models can be made more versatile in doing machine translation of longer source patterns by using *Attention* mechanism. Attention in general was proposed to provide more feature-full encoding of the source HMI event pattern from which finer grained context vectors can be obtained. This allows the *decoder* to apply varying degree of *Attention* to every input event token with its corresponding *encoder* hidden states - referred to as *Keys*(*K*) or *Values*(*V*), in the source sequence for predicting each HMI event token in target sequence, given previous decoder hidden state - referred to as *Query*(*Q*).

RNN Attention Mechanism

$$
\begin{aligned}
&\texttt{Similarity Score:}\\
&e_{ti} = a(s_{t-1}, h_i)\\
&\texttt{Attention Weight:}\\
&\alpha_{ti} = \frac{exp(e_{ti})}{\sum_{i=1}^{K} exp(e_{ti})}\\
&\texttt{Additive Alignment Model:}\\
&a(s,h) = v_a^T tanh(W_a s_t + U_a h_i)\\
&\text{Where } W_a, U_a, v_a\\
&\text{are trainable input, hidden and output}\\
&\text{layer weight matrices resptively.}\\
&\texttt{Context Vector:}\\
&c_t = \sum_{i=1}^{K} \alpha_{ti} h_i
\end{aligned}
\tag{4.7}
$$

The *Attention* mechanism [108, 116] (Fig. 2.6) results in constructing a attention context vector, *Attention*$(Q, K, V) = c_t$ (2.1), as a sum of encoder hidden states weighted with normalized attention weights ($softmax(f(Q, K_i)) = \alpha_{ti}$) used for predicting each output HMI event token. It starts with an alignment model ($f(Q, K_i)$ to learn a similarity score (e_{ti}) between all hidden states (h_i) of the *encoder* at time-steps $i = 1, ..., k$ of the source sequence with respect to the decoder's output hidden state s_{t-1}, at previous time step $t - 1$. The alignment score e_{ti} essentially captures how relevant (or similar) each encoded state h_i (h_1, h_2 ,...,h_k) (*Key* and *Value*) is to the decoder hidden state from previous time step (s_{t-1}) (*Query*), which can be used to infer decoder output at next time step ($T = t$). Each score e_{ti} is then normalized using *softmax* function in order to be used as probability value to indicate how likely each encoded input HMI token is relevant to the current decoded output HMI event token. The normalized scores are also referred to as attention weights (α_{ti}). The alignment model is implemented as a feed forward single layer perceptron that is jointly trained (W_a, U_a, v_a trainable matrices) (4.7) with the rest of the translation encoder-decoder *Seq2Seq* model to learn various attention weights for every encoded tokens with respect to an output token. The context vector (c_t) is calculated for every decoder output step (t), it represents the sum effect of all encoder hidden states ($i = 1, ..., k$) weighted with *expected* attention weights (α_{ti}) - this captures the relative influence of each HMI event token in input sequence $X^{(k)}$ on an output HMI event token in target sequence ($\hat{y}^{(N)}$). The above attention mechanism is based on global attention mechanism using additive (feed-forward) similarity function as initially proposed by [116] and described in (4.7).

4.6 HMI NLP Modelling using *Seq2Seq* CNN Model

Recent trend in departure [149, 150] from using RNN based encoder-decoder (E.g. *LSTM-EncDec*) models for *Seq2Seq* prediction to the approach that combines *Attention* with convolutional neural networks (CNN), has shown to outperform RNN based models in area of machine translation. RNN based models are sequential in nature and generally require more memory bandwidth resources than computational units. In contrast CNN *Attention* models offers themselves as better suited for parallel and in-memory processing computational architectures [151].

The proposed model, here referred to as *Trident*, draws inspiration from the design recipe "Embed, encode, attend, predict" by [152] in 2016, for building NLP CNN based models. It also utilizes design concepts from transformer [108] and residual network (ResNet) [153] models to build a scalable network that provides comparable or better performance than sequential *LSTM-EncDec* model with single-headed attention layers evaluated herein.

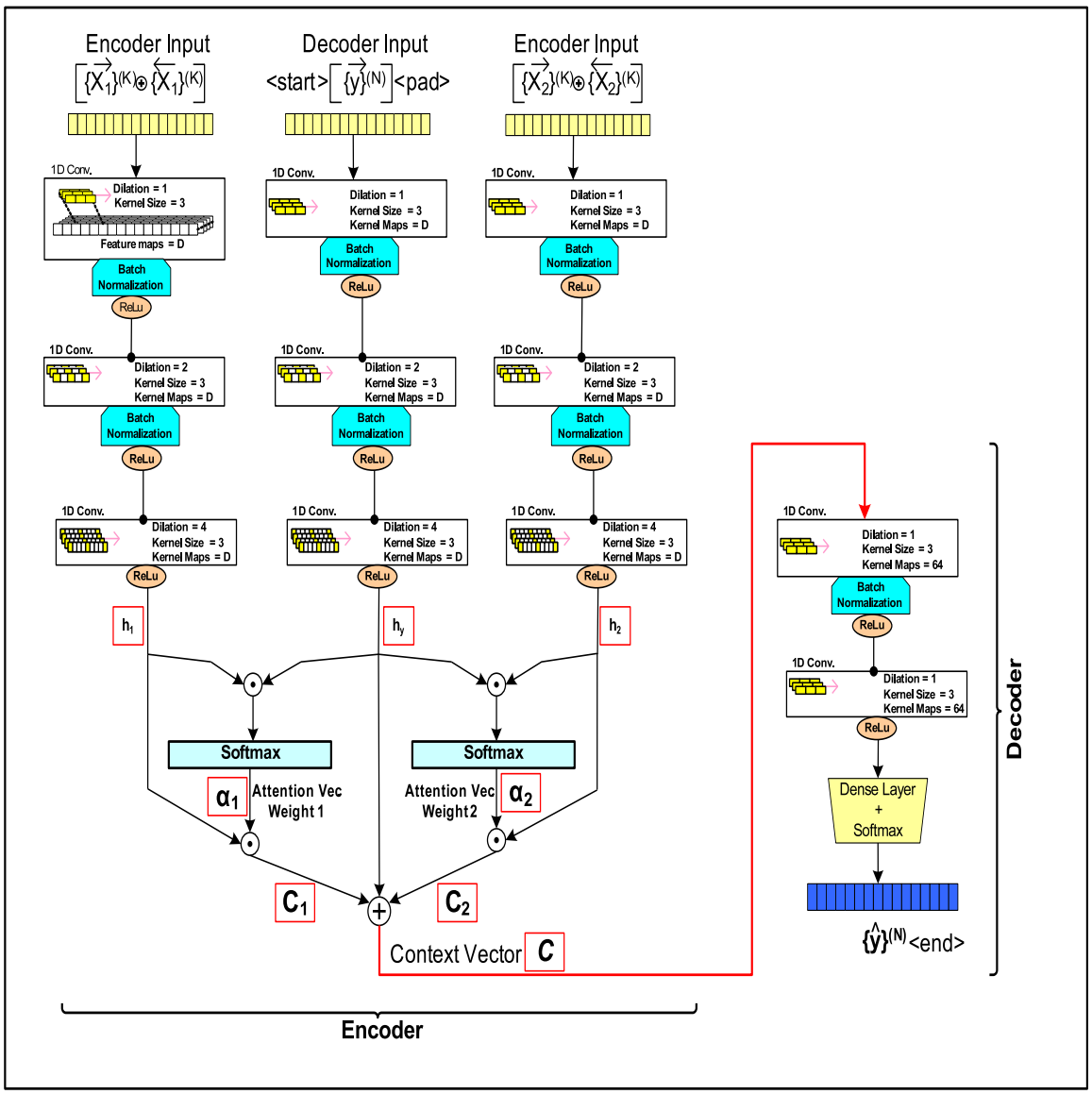


Figure 4.12: Trident Model: Contains stacked 1D CNN layers to build a parallel architecture that uses dot product attention layers for its encoder-decoder network. Intermediate deeper CNN layer have larger $(1, 2, 4)$ dilated kernel filters to increase the receptive filed of the convolution maps to capture longer dependencies.

4.6.1 Trident - Encoder

Supervised training of the *Trident* requires the same type of supervised framed inputs (encoder: INPUT1, decoder: INPUT2) and target reference output (OUTPUT) as required by *Seq2Seq LSTM-EncDec* model, discussed previously. However, the encoder inputs (INPUT1) is framed slightly different, where k-lag samples of each HMI indication feature (PROCESS and HMI_USER) sequence of $\{X\}^{(k)}$ $(X1, X2)$ are made bi-directional by concatenating the corresponding sequence in reverse order (E.g. $\overrightarrow{\{X_1\}}^{(k)}$ concatenated with $\overleftarrow{\{X_1\}}^{(k)}$). This doubles

the length of the resulting training input vector of each HMI indication feature vector.

In addition the target HMI event sequence ($\{y\}^{(n)}$) is introduced as input to the *Trident decoder* in a *progressive* way, during training, that is one new target token every few epochs (each target token from the training set is appended to intermediate training sets). The overall $\{y\}^{(n)}$ target sequence is padded to maintain a uniform sequence length. Therefore, the total number of training epochs is equally split between all (n) tokens of the target HMI event sequence ($\{y\}^{(n)}$). Progressive sequence training is done to allow the model to train in a way, mimicking its functionality during translation or performing inferences (Sec. 4.6.5). This technique is to overcome exposure bias [154] (training-inference discrepancy) in training a non-sequential CNN based model using *Teacher forcing*. Results show improvement while predicting the output sequence one token at a time during the inference phase, as the model *encoder* over-fitting is reduced. The model is also more resilient to begin translation using a partial sequence consisting of only a few previously output tokens generated by the *decoder*. A $< Start >$ token is appended to each *decoder* input and an $< End >$ token to each target HMI event sequence similar to training the *LSTM-EncDec* models. Again, the rationale for these tokens is to offset (lead) the *decoder* input sequence during training by one-time step ($t - 1$) compared to the target HMI indication sequence.

Trident's parallel structure is not a variant of the *Siamese* [155] or a *Triplet* [156, 157] network as much as its structural representation may resemble those latter works. The differences are:

1. *Siamese* Network requires two paired inputs: Positive and Negative training sets for learning the similarity between the two using a distance-based loss function. The model learns to distinguish between similar and dissimilar pairs of examples. This is called *One-Shot* learning. Even though *Siamese* networks do improve classification when the class sample size is small, they are sensitive to input data set calibration capturing the notion of similarity vs. dis-similarity in a given context [156]. For example, a pair of image 'A' and 'B' for a given multi-class data set may belong in separate classes. However, the same images 'A' and 'B' may be part of the same class in another multi-class data set, which will require the previously trained *Siamese* model to be retrained.

2. *Triplet* network [156, 157] is an improvement over the *Siamese* network and requires an *Anchor* reference sequence in addition to the Positive, Negative sequence – this is specially used for multi-classification and NLP domain for detecting text similarity problems. It uses a *triplet* loss function, which is self comparative with respect to the *Anchor* sequence combining both the dissimilarity and similarity measures.

3. In *Trident*, there are two K-lagged distinct $X1$, $X2$ bi-directional vectors, and the de-

coder is expected output sequence (y) vector based on the context from the two input vectors. CNN layers extract the hidden features from each of these input sequence vectors, which is then combined using Dot product operation to extract attention weights. The attention weights extract the context between Encoder and Decoder input sequences to aid the decoder layer to learn.

Therefore, unlike the above *Siamese* and *Triplet* networks which are maily used for multi-classification based on comparetive input sequences, the *Trident* network is a language translation multi-classification model which is inspired by Transformer model [108].

4.6.2 Trident - CNN Layers

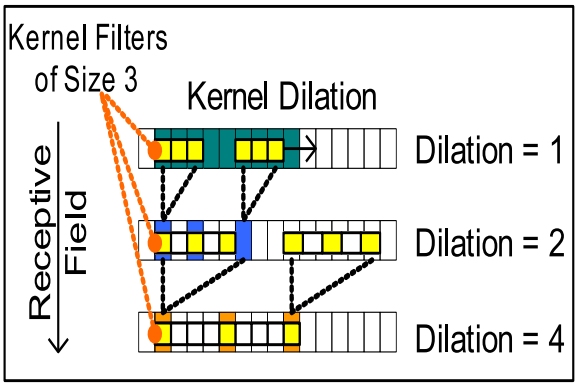


Figure 4.13: Dilation of kernel filters increases the receptive field of the higher subsequent convolution maps.

The proposed *Trident* model design as depicted in Fig. 4.12, includes a parallel pathway consisting of several convolutional neural network (CNN) layers to extract features from each input and the target HMI indication sequences. A 1D-convolution (CNN) layer is utilized since it is more conducive for processing sequence-based data sets, rather than images that have spacial information in two-dimensional space. The size of the convolution kernel (filter) is has been chosen ($KernelSize = 3$) to be a value that is commonly used most CNN based models for language translation as per literature review.

Number of feature (filter) maps generated in the encoder for each convolution layer is set to the dictionary size ($Featuremaps = D$) of the HMI model input and output indication languages ($\mathcal{L}_1$, $\mathcal{L}_2$) including the special $< End >$ and $< Start >$ tokens. Therefore, D feature map vectors are generated, followed by the application of a non-linear activation function (rectified linear unit (ReLU) is used currently). In order to minimize model over-fitting, regularization is required. Regularization in CNN can be achieved by *dropout*. Dropout causes the

outputs of intermediate convolution layers going into the final output dense (fully connected) layer to be randomly dropped, in order to avoid saturation of gradients during backpropagation. Nevertheless, *dropout* does lead to "un-learning" or decimating the previously learned weights, especially when the training data set is smaller. Instead, *batch normalization* [158] was used between each convolution layers as a way to regularize each convolution input pathway of *Trident* model while it yields other benefits such as reduction of covariate shift by normalizing the activations of each layer and speed up learning of CNNs.

Unlike the RNN *encoder-decoder* (*LSTM-EncDec*) models, basic CNN encoders are not sequential and do not learn sequence dependencies in long patterns by default as efficiently as LSTM layers can. This limitation can be overcome by arranging the longer input sequences in spatially close proximity such that the resulting convolution feature maps may capture sequential patterns and related dependencies. In order to avail this, each input sequence has been made bi-directional by concatenating the previous sequence in reverse order (as stated previously). Another design feature is to use dilated kernel filters in intermediate convolution layers, which further allows the *Trident* to increase its *receptive* field, as depicted in Fig. 4.13, to capture longer sequence dependencies.

4.6.3 Trident - Attention Layer

Attention mechanism allows the CNN decoder to use various sub-regions of the input to draw the output token and the mechanism fundamentally follows as described previously (Sec.2.5.5). In the proposed model CNN *Trident* model (Fig. 4.12), firstly the similarity of the CNN layer internal representation vectors (h_1, h_2) for bi-directional *encoder* input $\{X\}^{(k)}$ sequence(s), against the target *decoder* input sequence ($\{y\}^{(n)}$) internal representation (h_y) is obtained by performing vector dot products. The result of dot product generates similarity scores for each sub-segment of the input HMI event sequence against the target HMI event sequence. The following *softmax* activation is used to normalize the similarity scores to a probability distribution over all segments of the input sequence, which can be used as the attention weight vector (α_1,α_2). Secondly, the attention weight vectors are combined with the *encoder* input $\{X\}^{(k)}$ CNN layer internal representation vectors (h_1, h_2), which ultimately applies the attention (focus) on various segments of the HMI sequence $\{X\}^{(k)}$ given the target HMI sequence $\{y\}^{(n)}$. Lastly, all context vectors (C_1,C_2) are concatenated along with the internally represented *decoder* input h_y to obtain the final context vector $\mathcal{C}$.

4.6.4 Trident - Decoder

Training of the *Trident decoder* is done along with the *encoder* using progressive target sequence as per *Teacher forcing* technique. The *decoder* learns to use the context vector ($\mathcal{C}$) to output the next token ($t = T$) of the target sequence, given the partial sequence containing target tokens $t = 0$ to $t = T-1$ (where, T is the sequence index of token in target sequence of length $T = N$) as the input to the *decoder*. The context vector $\mathcal{C}$ captures the *encoder attention* or influence of the input sequences ($\{X\}^{(k)}$) on each partial target sequence $\{y\}^{(t-1)}$, which are introduced to the *decoder* progressively during training beginning with a $< Start >$ token marker. The ground truth target sequence is also required and is same as the *decoder* input sequence but is offset by one time step ahead ending with $< End >$ token.

Inference is carried out similar to model training, where various HMI indication parameter sequences of length K ($\{X\}^{(K)}$) are provided as input to the *encoder* as bi-directional sequences. The *decoder* input initially starts off with just the $< Start >$ token padded to the target HMI sequence of length (N). Subsequently, as the *decoder* outputs a new token ($\{\hat{y}\}^{(n-1)}$), it is appended to the sequence which is fed back as input to the *decoder*. The context vector ($\mathcal{C}$) evolves with the previously output target token and is used to generate a new target token ($\{\hat{y}\}^{(n)}$) by the *decoder* until the $< End >$ marker is output.

Trident decoder is implemented with two back to back 1D CNN layers with the first layer having batch normalization. Both CNN layers have fixed number of kernel filters (64). A dense layer with *softmax* activation is used to reshape to a target sequence of length N tokens, which are one-hot encoded to dictionary size $D = 255$ (including sequence de-marker tokens).

4.6.5 Trident - Training and Inference Phase

Training Phase

Trident training phase is done in progressive manner as described previously in Sec. 4.6.1 for both training techniques *Teacher forcing* (Fig.4.11a) and *curriculum learning* (Sec. 4.7).

Currently, the encoder training inputs (INPUT1 and INPUT2 $\{X\}^{(k)}$) contain bi-direction k-lagged sample values of PROCESS and HMI_USER ($X1$, $X2$) features. Another required training input to *Trident* is the target output (OUTPUT) sequence that is fed to the decoder and contains n-ahead samples of HMI_USER feature.

Inference Phase

The *Trident* model once trained using either *Teacher forcing* or *curriculum* training may be used to infer (Fig. 4.11a) the expected HMI event(s) of the target HMI state (translated lan-

guage pattern) $\hat{y}^n$ one step at a time. Translation begins once the $< Start >$ token is fed to the *Trident decoder* for all the input sample sequences at once. This is done as a 2-d zero vector containing only encoded $< Start >$ tokens in the first column. In addition, as required during training, inputs (INPUT1 and INPUT2 for $\{X\}^{(k)}$) containing bi-direction k-lagged sample values of PROCESS and HMI_USER ($X1$, $X2$) features is given.

Subsequent, target event tokens are inferred by feeding as input, partial target HMI event pattern that is built by appending (previously predicted) event tokens to the encoder's 2-d input vector (initialized by $< Start >$ token initially). The downside of this is if any previous inferred token is incorrect, subsequent predictions will be off. Therefore, having trained the model using progressive training compensates for this issue since it mimics the teacher forcing during the inference phase.

It is worth clarifying even though *Trident* can infer the output HMI states one step at a time, it is iteratively can be made to so for n-ahead future time steps, for which it has been trained for. It does so with only having supplied past k-lagged samples of both input features at a time. Hence, only k past samples of HMI states directly influence the forecast of HMI state for the next n-ahead time-step window. This applies to both NLP *Seq2Seq* models evaluated in this research and may be generalized to all NLP machine translation models as they are able to translate the entire length of source sentence from one language to another on the word at a time while previous translated word influences the next translated word.

4.7 Curriculum Training

Curriculum training [159] overcomes the limitations of *Teacher forcing* and makes the model more versatile (generalized) during training and accurate during inference. Training the *Seq2Seq* models discussed above using *Teacher forcing* inherently prevents the models to robustly learn to predict during inference phase. The issue is referred to as *exposure* bias (training-inference discrepancy [154]), where the models are trained using actual ground truth target tokens ($\{y\}^{(n)}$) each time step, while during inference the model *decoder* is fed back previously predicted token ($\{\hat{y}\}^{(n)}$). Thus, limiting the skill of the *Seq2Seq* models to correctly predict target tokens during inference over longer forecast windows. As, once an incorrect token is output, it may throw all prediction of subsequent target token sequence off completely.

Curriculum training [159] entails *Seq2Seq* model *decoder* to be trained in mini-batches where both actual ground truth and predicted tokens ($\{y\}$, $\{\hat{y}\}$ receptively) are utilized. A sampling schedule is followed to systematically control the probability (per epoch) in training mini-batch of how many of either target or formerly predicted tokens are to be fed to the

decoder during training. Namely, at each training epoch based on the selected probability schedule (linear, exponential or inverse sigmoid decay) [159] either (actual) $\{y\}^{(n-1)}$ or last predicted $\{\hat{y}\}^{(n-1)}$ is used. The sampling schedule is setup to decay the probability (ϵ_i) of selecting ground truth ($\{y\}$), vs. previous predicted token ($\{\hat{y}\}$) decreases as training epoch count increase in a mini-batch (e.g. exponential $\epsilon_i = k^i$ decay, where $k < 1$ and i is epoch index). This allows the model to get trained by *Teacher forcing* in the beginning and slowly transition to using predicted target token output from the trained *decoder* trained thus far, hence improve generalization and reduce over-fitting.

Trident is trained with using both *Teacher forcing* and *curriculum* training with linear decay sampling schedule. It is noteworthy to state here both training techniques are used in conjunction with the progressive sequence training, which rather controls how the *decoder* input is built progressively as previously described while paying attention to the entire source sentence structure as context.

4.8 Summary

This chapter covered the following implementation details:

ViDAQ Design Detail. This section discussed the design environment and platform for the ViDAQ prototype implementation. E.g., Jetson board platform, python /opencv version etc.

Supervised Learning - Data Framing. This section covered the detail of the supervised data framing technique that has been utilized for creating the training, evaluation, and test datasets for all the HMI state sequence forecasting models.

HMI Time-Series Modelling using ARIMA(p,d,q). This section covered various ARIMA: *Static*, *Dynamic*, *Adaptive*, models that were implemented and how these were trained and evaluated against the *Persistence* baseline model using the synthetic data set. Also covered in this section are the statistical pre-checks that are required to be conducted on the training data set prior to ensuring the effectiveness of the ARIMA model on the given data set. These include *stationarity* tests using *ACF* and *PCAF* plots in addition to *ADF* null-hypothesis test for unit roots.

HMI Time-Series Modelling using LSTM and CNN Models. This section include design details of the recurrent neural network LSTM based models in various configurations: *Vanilla*, *BiDirectional*, *Stacked* and *Stacked* with *Time distributed* output layer. More-

over, design details of two covolutional neural network (CNN) models: *1D-CNN* and *2D-CNN* that were implemented were also discussed herein.

HMI Modelling using NLP *Seq2Seq* RNN and CNN Models. This section include design details of the *Seq2Seq* recurrent neural network LSTM based *Encoder-Decoder* model with attention layers that is implemented based on other research works. In addition, design detail of a custom model (*Trident*) - a CNN based *Encoder-Decoder* with dot product attention mechanism was discussed above.

Chapter 5

Experiments and Results

This chapter provides experiment setup details and evaluation result analysis for previously proposed solutions in Chapter 4. Experiments included herein are for: ViDAQ implementation test cases; evaluation of ARIMA models: In-sample (*InS*), Out-of-Sample (*OuS*) testing of static, dynamic and adaptive models); evaluation of various standard LSTM (Vanilla, bidirectional, stacked, convl.LSTM) and CNN (1-d and 2-d) time-series models; evaluation of a NLP LSTM based *seq2seq* model; evaluation of custom designed NLP CNN based model.

The section also discusses the custom utility used to generate controlled synthetic data for studying forecast capabilities of various forecast models mentioned above. Lastly, data collected from real-world Nuclear Power Plant simulated scenarios that capture sequence of actual process and control room panel information was used to demonstrate capability of HMI state forecast models to detect HITL error precursors.

5.1 ViDAQ Test and Results

ViDAQ components are evaluated under two experiment setups: HMI integrated test with screen-captured images as a source and test with live camera stream as a source.

5.1.1 HMI Integrated Test Setup

The purpose of the HMI Integrated test is to validate the concept of *EYE-on-HMI* (Sec. 3.1) and ViDAQ frameworks to non-intrusively detect and trend operator activity by only using the visual feedback information from a given operator HMI. Situational Awareness is closely related to the level of operator task workload; therefore, monitoring operator activity index may be useful in ascertaining the level of operator situational awareness where the Activity Index is calculated below as the operator's alarm servicing rate. This metric is a function of

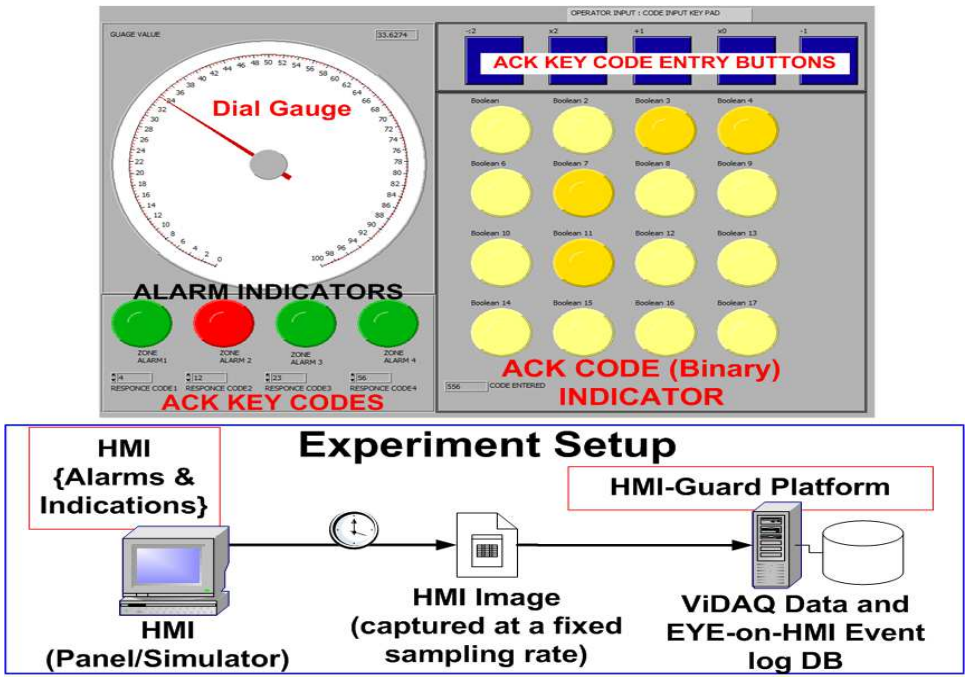


Figure 5.1: ViDAQ Integrated Test with Prototype *EYE-on-HMI* Platform called HMI-Guard Setup. Prototype HMI (top); Integrated test data flow (bottom) - HMI states as images captured are saved to disk at regular intervals and fed to HMI-Guard platform for processing

a number of HMI events demanding operator actions (E.g. alarms) divided by the cumulative missteps in addressing those alarms.

Experiment setup for evaluating the functionality and performance of the HMI-Guard, which is a simple *EYE-on-HMI* platform that combines ViDAQ capability only, as shown in Fig. 5.1. A soft HMI panel is used to emulate a typical industrial control panel with legacy indicator devices (E.g. a dial gauge and indicator lamps), including a basic operational rule to engage a human operator to respond to active alarms.

The prototype soft HMI panel (Fig. 5.1) (developed using NI Labview 8) includes: (1) one constantly moving rotary dial gauge (scale: 0 to 100 with 50 ticks); (2) 4 multi-state $Red(active)/Green(inactive)$ alarm state indicators; (3) a 4×4 grid to display a 16-bit binary word using $Orange(1)/Yellow(0)$ lamps; (4) 5 code entry pushbuttons for user. The soft HMI panel activates a corresponding zone alarm (Red) once the gauge reading is within the zone (high/low) limits.

Associated with each Zone alarm, is a pre-set alarm acknowledgment (Ack) code. The user must input Ack code by using available keypad push buttons. As the user inputs an Ack code its the 16-bit representation is displayed on a 4×4 grid (with 0^{th} to 15^{th} bit positions in order of top-left corner to bottom-right, as shown in Fig. 5.1).

The operational goal is to manually acknowledge the zone alarm as soon as possible, prior

to next zone alarm activating, by inputting the correct pre-set Ack *key* code. In other words, the user or operator must ensure Zone alarms are cleared as soon as possible by manually inputting the correct pre-set Zone Ack code value that is constructed by using five math function keypad buttons ($\times 0$, $+1$, -1, $\times 2$ and $\div 2$) (Fig. 5.1). The math function keys are included to simulate cognitive workload.

As mentioned previously, operator action via the input code values displayed dynamically on the 4×4 grid provides visual feedback for the operator. In addition, other visual feedback such as dial gauge reading, alarm indication lamp states are saved as screen-captured images of the prototype software HMI panel to be processed by the ViDAQ.

Experiment data is gathered by capturing screen-shots of the soft HMI panel to capture the HMI state at (an adjustable) $900mS$ sampling interval. The images are then streamed as input to the HMI-Guard platform for HMI event logging and evaluation. The experiments are conducted in simulation (i.e. live video camera feed is not used to capture HMI states), to only test ViDAQ processing precluding any data acquisition errors.

5.1.2 Gauge Reading Single dial Error

ViDAQ's absolute acquisition errors associated with reading gauge values are shown in Fig 5.2. Source of these errors typically can be attributed to 2 main reasons: Malformed contours and Quantization noise [160] where, the latter is more prevalent during reading single-dial meters. Quantization or round-off errors (shown enclosed by green rectangular box in Fig 5.2) lead to approximately $\pm\{0.1, 0.7\}$ in value reading error, while malformed contours shown enclosed by grey rectangular box in Fig 5.2) attribute approximately $\pm\{0.8, 0.9\}$ in reading error. The occurrence of quantization error inversely depends on the distance (quantization step) between two adjacent graduation marks (ticks) on the dial - smaller the distance higher is the occurrence of error.

Malformed contours emerge as artifacts of (Dilation mask of Feature Extraction step in Fig 4.6b) morphological operator parameters currently not being able to adapt to dial shape and orientation. Since, each dial orientation represents a unique contour shape, it is both a dial position and shape-dependent operation and adds noise to dial angle measurement.

5.1.3 Indicator State Detection Error

ViDAQ performance, while detecting the state of indicators, has proven to be quite robust based on the HSV colorspace masking technique, as demonstrated in Fig 4.4a combined with the indicator localization and state detection outlined by Algorithm 1 in Section 4.1.1. Simulation result yield negligible acquisition error while detecting 4×4 grid of lamp indicator

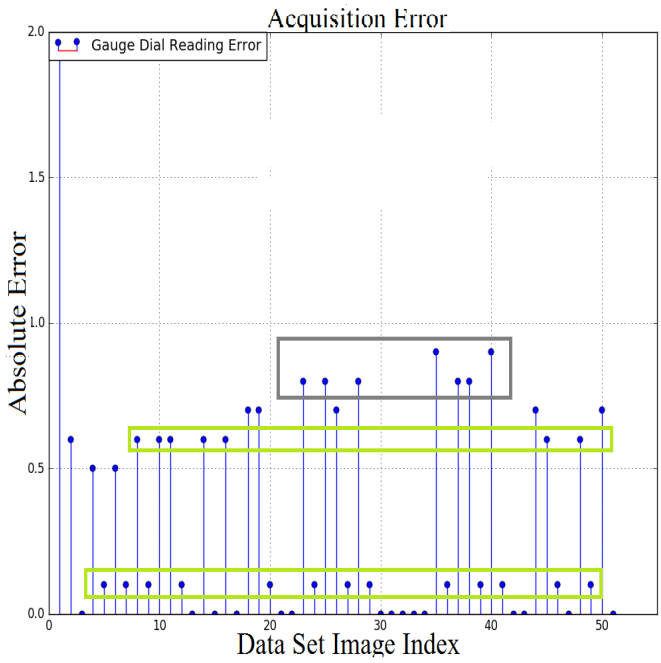


Figure 5.2: ViDAQ HMI Integrated - ViDAQ Dial Gauge reading Performance Error.

states as shown in HMI-Guard application (Fig. 5.1) and hence, has not been included herein for brevity.

5.1.4 HMI Integrated Test Result

The HMI-guard visually captures using ViDAQ [160] the readings of the dial gauge that sweeps values from 0 to 100. The acquired gauge value reading is validated as a monotonically increasing linear curve labelled *Gauge Reading* (★ symbol) in each top row of plots in Fig. 5.3 - 5.5. When the gauge dial value enters a Zone (*Low*$-$*Hi* limits of *Zone1*:$10-22$, *Zone2*:$23-50$, *Zone3*:$51-78$, *Zone4*:$79-100$) corresponding Zone alarm ($Alm.Zone$) is activated (ON) and is deactivated (OFF) once the dial exceeds that Zone's *Hi* limit. This behaviour is accurately captured by HMI-Guard platform output, as seen in each top row of plots in Fig. 5.3 - 5.5 as 4 non-overlapping step functions for each $Alm.Zone1, 2, 3, 4$ that activate and deactivate in sequence with respect to the *Gauge Reading* value. In addition, HMI-Guard also captures the $Ack.$ code values input by the operator via the keypad in response to Zone alarms. The $Ack.$ code vales are seen as scattered values (● symbol) in top row of Fig. 5.3 - 5.5. Evidently, each $Alm.Zone$ deactivates when user input $Ack.$ code value matches the pre-set $Ack.$ *key* code (shown as dashed horizontal lines with same colour as $Alm.Zone$), thus validating the intended operational goal for operator action.

Bottom row of plots in Fig. 5.3- 5.5 capture the human operator performance statistics

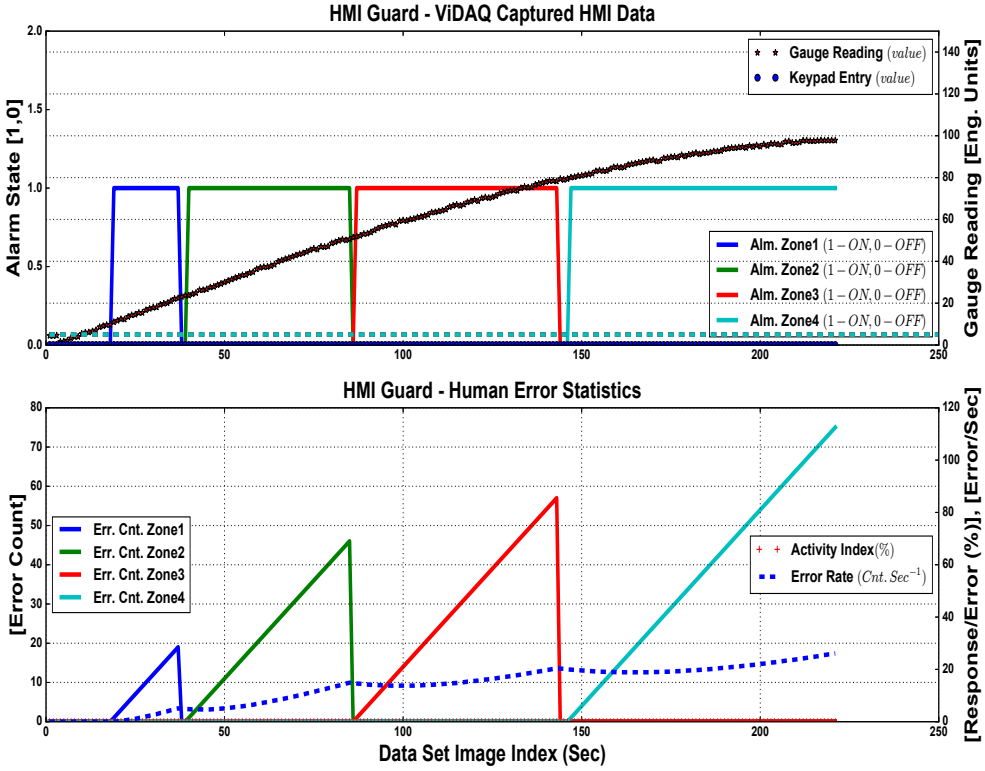


Figure 5.3: HMI Integrated ViDAQ - Constant HITL (Human-in-the-loop) Error Run. This is obtained when there no user input of $ACK.$ codes and HMI is in a constant HITL error state. **Top plot** shows all four zone alarms come ON/OFF as dial value constantly changes. There is no keypad entry events logged by ViDAQ; **Bottom plot** shows constant Error Rate trend and zero Activity Index as expected.

updated every sample interval (t_{sample}). Two error counters are utilized, firstly, a Zone Error Count (*Err.Cnt.Zone*), associated with each alarm event (Note: *Zone Error* in this context refers to sampled HMI states where, HMI is in trouble or alarm state). It keeps count of number of sample intervals that have elapsed since a $Alm.Zone$ was triggered and only resets to 0 when the associated zone alarm deactivates. Secondly, a Cumulative error count (*Cuml.ErrorCount* 5.1) is a running tally of individual *Err.Cnt.Zone* values and only resets at end of experiment.

Activity Index(%) (equation 5.2) is the ratio of total number of operator initiated updates (activity) on the HMI with respect to the current value of *Cuml.Error Count*. Where, each α_i keeps count of instances operator action is detected in response to a particular event (Zone $alarm_i$) - in this case operator action is detected once 4×4 binary indicator status changes (Fig. 5.1) as the operator inputs new $Ack.$ code; α_i resets once the associated Zone alarm

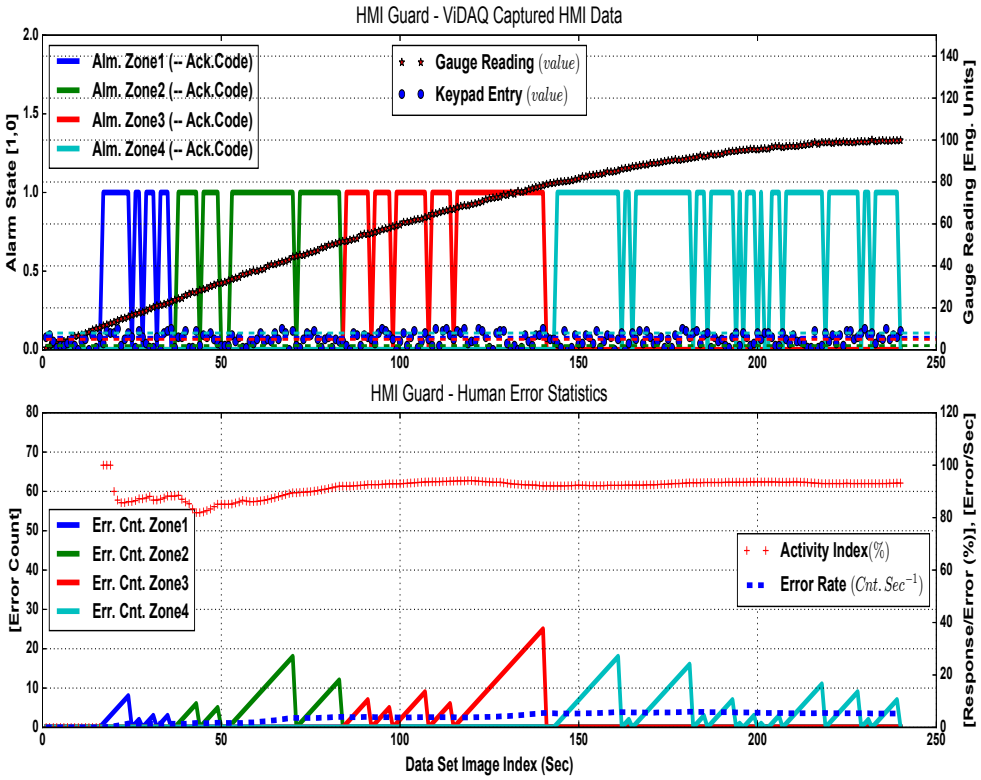


Figure 5.4: HMI Integrated ViDAQ - Random HITL Error Run Result. This is obtained under random user input of $ACK.$ codes and HMI shows random HITL error state. **Top plot** shows all four zone alarms come ON/OFF as dial value constantly changes. There are several keypad entry events logged by ViDAQ; **Bottom plot** shows constant but lower Error Rate trend due to user inputs are uniformly distributed. Activity Index is higher but constant as expected also owing to user inputs being uniformly distributed.

deactivates. This metric captures any operator response with respect to existing alarms events on HMI. Whereas, *Error Rate* 5.3 (■ symbol) in bottom row Fig. 5.3 - 5.5, tracks in real-time overall error in operator response to correctly address HMI alarms, which effectively aids in capturing operator situational awareness (Note: t_{sample} is the elapsed time obtained from current sample index $\times$ $900mS$ - ViDAQ sampling interval).

$$Cuml.ErrorCount = +\Sigma_{i=0}^{MaxAlm} Err.Cnt.Zone_i \tag{5.1}$$

$$ActivityIndex(\%) = \frac{\Sigma_{0}^{t_{sample}} \Sigma_{i=0}^{MaxAlm} \alpha_i}{Cuml.ErrorCount} \times 100 \tag{5.2}$$

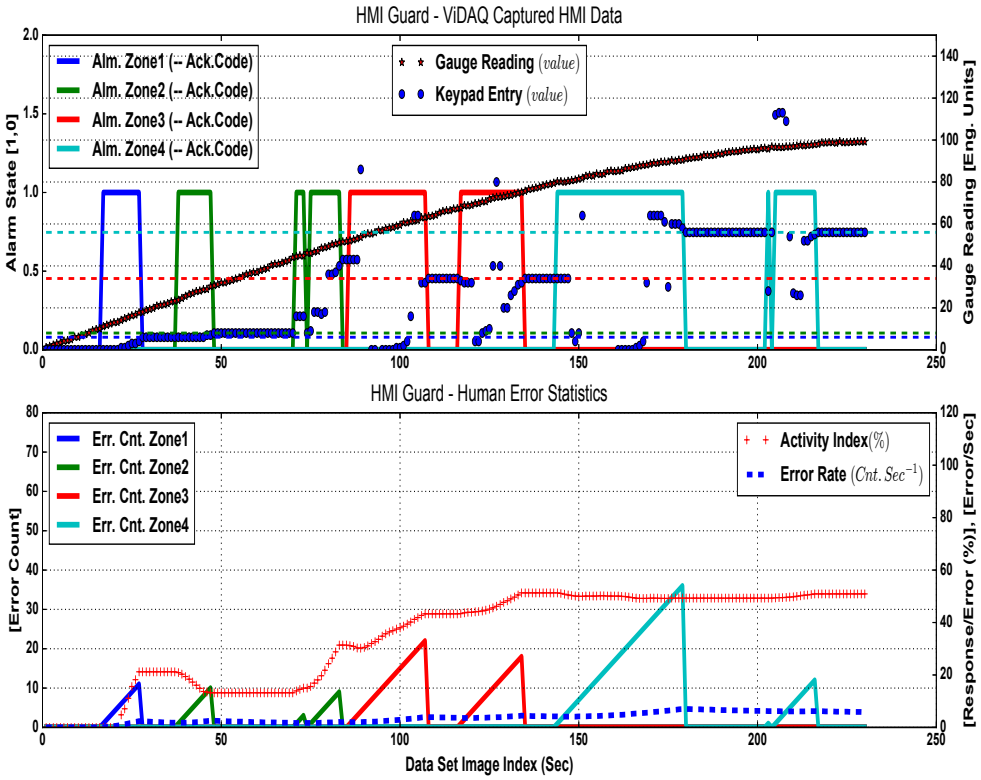


Figure 5.5: HMI Integrated ViDAQ - User Error Run Result. This is obtained by authors input of $ACK.$ codes and HMI shows real user input HITL error state. **Top plot** shows all four zone alarms come ON/OFF as dial value constantly changes. There are keypad entry events logged by ViDAQ as author's manual inputs; **Bottom plot** shows Error Rate on a slightly upward trend as user inputs become more error-prone with time possible due to manual fatigue. Activity Index steps up during the latter half (Zone 3, 4 alarms) and is correlated with increasing Error Rate during the same period. This is indicative of higher activity index that does not necessarily translate in accurate operator action but may be due to repeated trials owing to lower situational awareness.

$$ErrorRate = \frac{Cuml.ErrorCount}{t_{sample}} \tag{5.3}$$

Result in Fig. 5.3 shows experiment data collected under the constant error condition, where the operator never responds to any Zone alarms (Keypad Entry value is null) on the HMI. In such a scenario the *Error Rate* trends high to $28 Counts/Sec$ with 0% *Activity Index*.

In Fig. 5.4, data for the random operator error experiment is shown. In this scenario, keypad entry is automated to input random $Ack.$ code values close to the pre-set (alarm acknowledgement) $Ack.$ *key* code value. This simulates random acknowledgement of Zone alarms (seen as random sawtooth pattern of $Err.Cnt.Zone$). As expected, due to normal probability

distribution of randomized key inputs, both the *Activity Index* and *Error Rate* converge towards constant values. Moreover, due to frequent (every t_{sample}) automated but randomized $Ack.$ code entry, the operator *Activity Index* is higher (90%) but a lower operator *Error Rate* (5%) - both are indicative of an agitated operator having lower situational awareness.

In Fig. 5.5, data is captured when a real human user uses the soft HMI. In this scenario, keypad entry is done manually via the HMI code entry buttons (Fig. 5.1) to acknowledge Zone alarms. The values entered (• symbol), as seen in top row plot of Fig. 5.4, tends to converge closer to pre-set $Ack.\ key$ limits as the user attempts to clear the alarm. Two observation regarding the operator response rate and required effort can be made from the operator *Activity Index* in Fig. 5.5: (1) a steep increase is indicative of higher operator response rate (since $\alpha_i >$ *Cuml.Error Count*) to a HMI event (2) a gradual ramp-up (since $\alpha_i <$ *Cuml.Error Count*) is indicative of possible higher operator effort as operator is responding but its taking longer to reset the alarm.

5.2 ViDAQ Dial Reading and Results

Experiments were conducted to evaluate the functionality and performance of the ViDAQ component algorithms while extracting time value from rotary dial wall clocks. These experiments included profiling the algorithm performance and data acquisition error while processing clock face images sourced from a static image dataset. Secondly, ViDAQ accuracy and precision performance was measured while using a live camera feed to acquire time from a real rotary dial wall clock positioned at various distances from the camera. Finally, in order to test the functionality of ViDAQ on an embedded system, a Raspberry PI (ver. 2 model B) was used to run the ViDAQ code to read a dial watch from a fixed ($80cm$) distance under static lighting conditions over an extended period of time. The acquired time values were communicated via Ethernet for remote monitoring and trending.

In addition, lamp indication states from an actual control panel as displayed on the full-scope CANDU nuclear operator training simulator was also conducted using the **Nvidia Jetson TX2** board.

5.2.1 Static Multi-dial Image Test Setup

A static image dataset of dial clock faces were generated using a script. Specifically, two distinct datasets: 2-dial and 3-dial image datasets were generated, each consisting of 720 and $43,200$ images respectively showing all possible (seconds, minute, hour) dial positions for a 12 hour span. The ViDAQ algorithm was developed using *python* 2.7 *OpenCV* $2.4.8$ library

and tested on a Windows 7 machine {Intel i5-480M ($2.66Mhz$, $3MB$ $L3$ cache), $4GB$ DDR3 RAM}.

A test script was written to sequentially feed each image from the above two image datasets to the ViDAQ algorithm. ViDAQ output consisting of time and dial angle value was then compared to the expected time and dial angle values for each input image. The resulting absolute difference values were then captured in a delta-log file for further analysis.

5.2.2 Streaming Video Test Setup

In this test, a USB camera {Logoitech C720 HD Webcam} was used to capture clock dial face at 1280×720 resolution. The video stream was fed to the ViDAQ algorithm at $1/10$ the frame rate to avoid operating system resources from being overwhelmed.

The live camera streaming test entailed manually recording both the time values output by ViDAQ algorithm and the actual time indicated by the clock. A few designs of clock dials were also experimented to ascertain if the shape of dial needles and face design had any impact on visual data acquisition. Finally, a distance-based test was conducted to ascertain the accuracy and precision performance of ViDAQ with respect to the distance between the watch and the camera.

Results obtained under the experiments described above are discussed in detail below.

5.2.3 Static Multi-dial Image Test Result

Data Acquisition Errors

ViDAQ data acquisition errors are tallied as each dataset image is processed. Algorithm outputs captured at four critical processing stages of ViDAQ for both 2-dial and 3-dial clock images are shown in Fig 5.6.

In summary, corresponding image processing stages inlcude: (1) input image scaling and application of circular Hough-Transform to detect the rotary dial (meter) face; (2) Binarization which includes conversion to gray scale followed by thresholding; (3) application of morphological operators (dilation/erosion) to accentuate dial arm edges for contour detection; (4) finally convex-hull edge list (CvE) feature extraction is done to localize dial arm tip positions, which is required to determine respective dial angles.

ViDAQ absolute acquisition errors is shown in Fig 5.7. As previously mentioned, the source of errors typically can be attributed to 2 main reasons: Quantization and Malformed contours. Quantization or round-off errors (shown enclosed by green and grey rectangular box in Fig 5.7), lead to ± 1 value difference more commonly in acquired values of minute

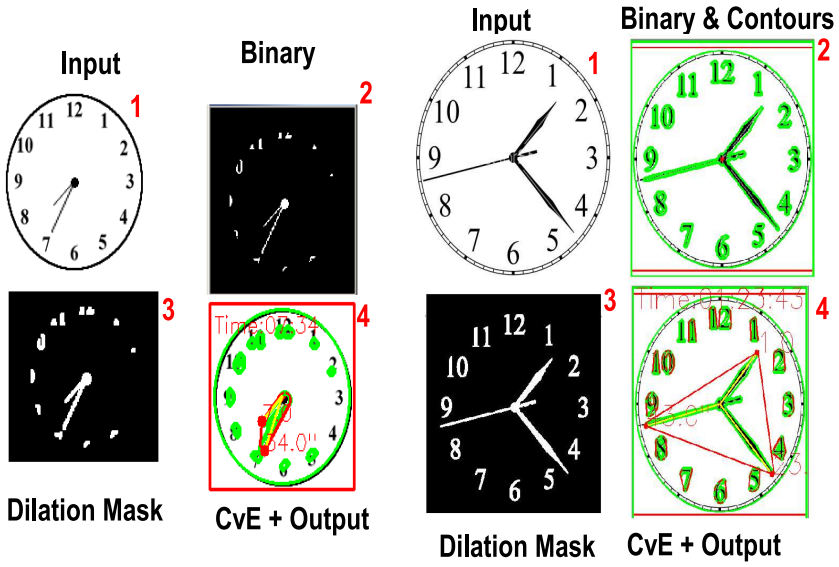


Figure 5.6: ViDAQ output - (left) 2-Dial: Input (600×600) Output:07:34, (right) 3-Dial: Input (380×400) Output:01:23:43

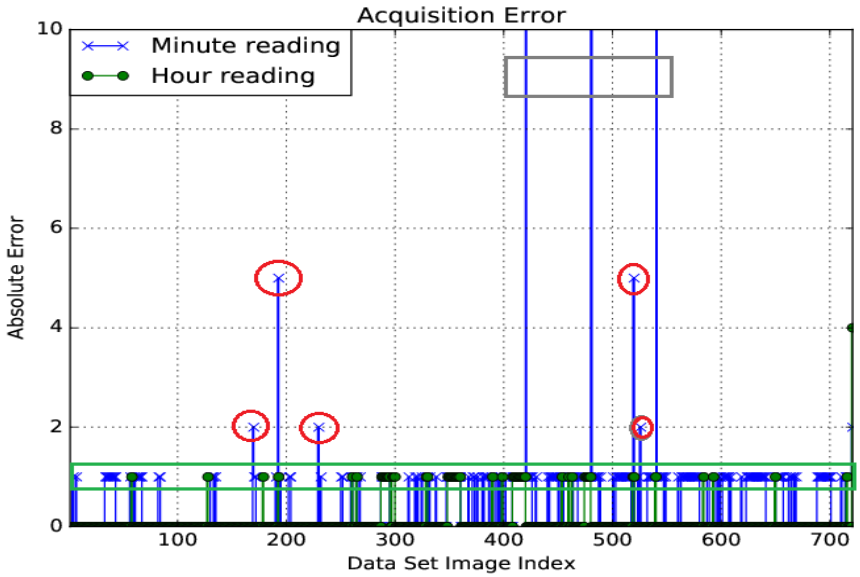


Figure 5.7: ViDAQ Acquisition Error over Image Data Sets: red circle - are due to malformed CvE; Grey rectangle - are due to round off error; Green rectangle - are due to quantization errors

readings than observed for hour reading. This is owing to the higher required accuracy ($6°/minute$ deflection - smaller quantization step) in minute dial angle measurement, to discern between adjoining minute values. Conversely, a lower accuracy ($30°/hour$ deflection - larger quantization step) is required in hour dial angle measurement, to discern between adjoining hour values. Also, apparent extreme ± 59 error overshoot noticeable in Fig 5.7, is

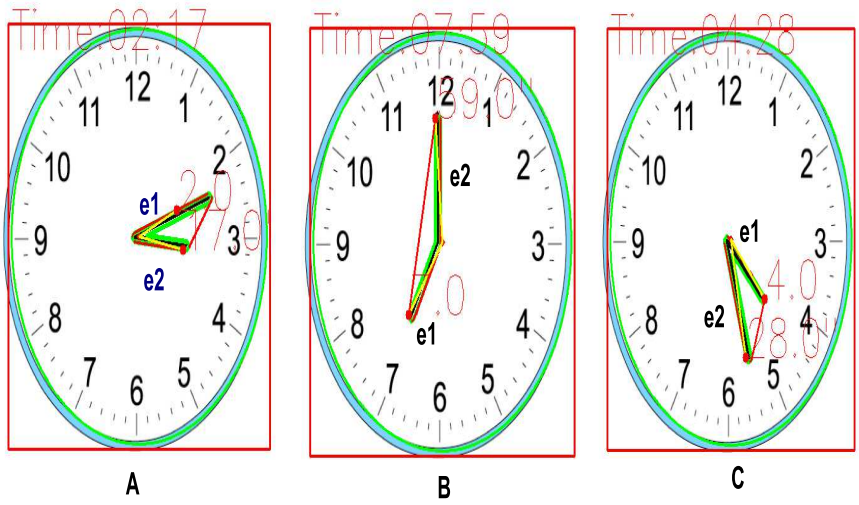


Figure 5.8: Common Errors: (A) Malformed Contour causes dials edges (CvE) to be incorrectly detected - Output:02:17 vs. 03:12; Quantization Errors: (B) Wrap Around Error - Output:07:59 vs. 07:00; and (C) Round-off Error - output 04:28 vs. 04:27.

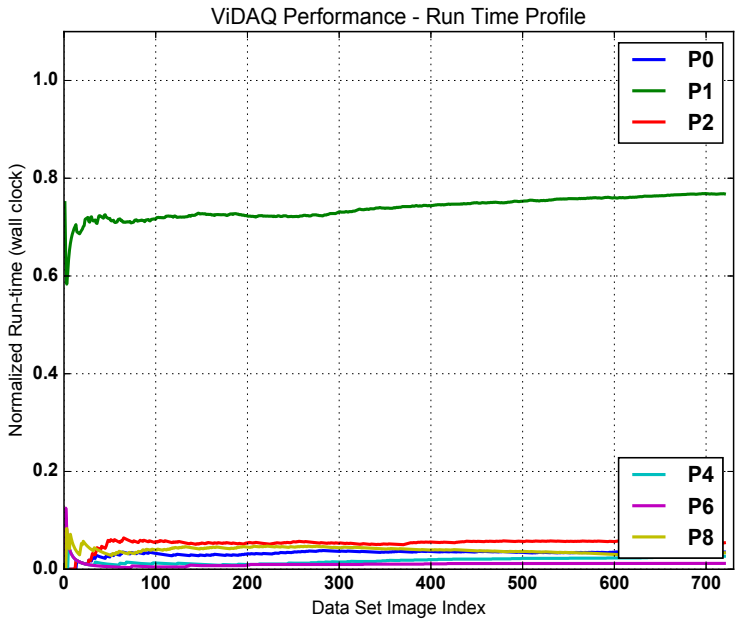


Figure 5.9: ViDAQ Normalized runtime profile (P0 - P8 are code segments)

caused when the difference in acquired value is between 0 versus 59 minutes, which can also be attributed to previous quantization error.

This error can be compensated within the ViDAQ value extraction logic by taking into account previously known states of minute and hour dials to smartly correct any minor quantization errors. Examples of two typical instances of quantization errors as discussed above

are shown in Fig 5.8: (B) - where 0 versus 59 minutes error occurs owing to conversion wrap around from 59 to 0 minute; (C) - is the case where ± 1 value difference occurs due to round-off error, which is more common in minute reading than in hour reading due to relative difference in quantization steps (as discussed previously).

The second source of error is caused due to malformed contours (step 4: feature extraction in Fig 5.6). Malformed contours emerge as artefacts of (step 3: Dilation mask in Fig 5.6) morphological operator parameters currently not being able to adapt to dial shape and orientation. Since, each dial orientation represents a unique contour shape, it is both dial position and shape dependent.

Moreover, malformed contour also causes instances of acquisition errors in ViDAQ, as has been observed when using various clock dial designs and lighting conditions when rotary dial faces are captured in reality via a camera. Currently, this is compensated by manually adjusting the dilation parameters tuned to work with a specific style of dial and under a given lighting condition. Malformed contour error can be seen in Fig 5.8 (A), where both dial tips have been incorrectly detected as being of equal lengths due to the merging effect noticed between the dials (CvE e1, e2) when they make smaller acute angles.

Malformed contour errors are less frequently occurring and can be compensated either by using a simple look-up or fuzzy logic technique to select appropriate morphological parameters based on dial angle dynamically.

Run-Time Performance

Run time performance of the ViDAQ code was determined by capturing wall clock time profile of various code segments while processing images from the image dataset. In Fig 5.9 wall clock times for each code segment has been normalized with respect to total cycle time spent processing per image.

Clearly, code segments $P1$ (Hough transform - dial face detection) and $P4$ (CvE contour feature extraction) each consume approximately 30% to 40% of overall processing time. This is expected as both these steps are computationally intensive and are currently using standard $OpenCV$ library routines.
Next noticeable time-consuming segments $P8$ (Angle measurement and time value extraction routine) utilizing approximately 5% and $P6$ (Dial arm and tip selection) utilizing approximately 3% of overall processing time. Both $P6$ and $P8$ are custom code segments implemented based on Algorithm 2. Other minor code segments are $P0$ (image scaling/grayscale/thresholding) and $P2$ (Inner/outter ROI and Erosion/Dilation), each consuming approximately 2% to 3% of overall processing time and are also currently using standard $OpenCV$ library routines.

Maximum ViDAQ image processing run-time latency is a function of the input image

Table 5.1: ViDAQ Live Video Streaming Test

Actual Time	**40cm**	**80cm**	**1m**	**1.5m**	**2m**	**3m**
6:21	6:21	6:20	6:21	6:21	6:25	1:09
6:09	6:08	6:09	6:09	6:10	6:11	3:23
3:29	3:30	3:29	3:32	3:29	3:26	9:20
2:38	2:38	2:37	2:38	**3:36**	**3:36**	n/a
12:20	12:19	12:18	12:20	12:19	12:19	n/a
1:55	1:54	1:55	1:54	**11:09**	**1:56**	n/a
2:37	2:39	2:40	2:40	**3:28**	**3:29**	n/a
3:00	3:02	3:02	3:03	3:03	**12:03**	n/a
2:21	2:22	2:22	2:22	2:20	2:20	n/a
7:15	7:15	7:16	7:17	7:17	7:17	4:15
Accuracy:	$\pm 1min.$	$\pm 1min.$	$\pm 2min.$	$\pm 3min.$	$\pm 1hr.$	$\pm > 1hr.$

resolution and number of dials in the first clock face to be processed in the image - as subsequent processing clock faces existing in the same input image exploits output of image pre-processing available from the previous run of processing the first clock face. For example, testing with an image resolution of 1280×720 containing one clock (with 3-dials), the maximum latency is still lower ($130\mu s$) than the rate at which the subject dials are expected to update. Consequently, ViDAQ sampling rate is adequate to offer real-time performance under the specific application scenario of reading analog dial faces.

5.2.4 Live Streaming Video Test Result

Table 5.1 tabulates manually captured data while ViDAQ is tested using a live video stream. The data is used to ascertain the dependence of ViDAQ data acquisition accuracy and precision when the subject rotary dial face (clock) is located at distances varying between $40cm$ (centimeter) to $3m$ (meter) from the camera. Accuracy here represents mean deviation in acquired time value versus actual time reading at a given acquisition distance (shown column-wise). Precision here represents the mean spread between the acquired time values captured at various acquisition distances when the clock time remains unchanged (shown row-wise).

The data (Table 5.1) shows an average accuracy of $\pm 1min.$ to $\pm 3min.$ (in minute values) when a reading is taken at a distance of $1.5m$, where the source of error is predominantly caused due to quantization effects (Sec. 5.2.3). Moreover, Table 5.1 data also reveals ViDAQ acquisition precision (smaller variance between contiguous readings) has a weaker dependence, compared to accuracy, on the ViDAQ acquisition distance when other factors such as target stability and lighting conditions remain unchanged. At distances greater than $2m$ the accuracy suffers drastically due to increased instances of malformed contours (Sec. 5.2.3)

resulting in misinterpretation of dial tips by the ViDAQ algorithm. However, as indicated previously (Sec. 5.2.3) malformed contours is addressed in our future work by incorporating a smart auto tuning of morphological parameters.

Though data acquisition accuracy and precision of ViDAQ's prototype algorithm is satisfactory for distances lower than $2m$, the current design has further room for improvement to increase the acquisition distance using higher resolution images for practical applications. Furthermore, future work should also extend ViDAQ performance to address external factors such as target illumination and image stability.

The main take away results are summarized as below in Table 5.2.

Table 5.2: ViDAQ Result Summary.

Test	**Result Highlights**	**Main Take Away**
ViDAQ HMI Integrated	Single-dial gauge sources of reading error: • Quantization error yields +/-{0.1,0.7} • Malformed features +/-{0.8,0.9}	Test confirmed ViDAQ can obtain information from an integrated EYE-on-HMI application using an image stream, which includes simultaneously reading states of: • A single dial gauge • 4 Alarm Lamps • 4 x 4 Lamp grid Real-time trending of user performance is demonstrated using custom **Activity Index** and **Error Rate** metrics.
	Lamp indication reading: 100% accuracy	
ViDAQ Live Camera test	Multi-dial gauge (Clock with hours, minutes dial): • Quantization error yields +/-{3} minutes at 1.5 meter distance and +/-{1} hour accuracy at 2m reading distance	Live camera test demonstrates successfully proof-of-concept means of acquiring values from HMI devices. **Acquisition accuracy depends on:** • Quality of camera, • Field of view (focal length), Image stability • Lighting conditions
	Lamp indication reading show 100% accuracy for **1 meter distance**.	**Acquisition accuracy depends on:** • Quality of camera

5.3 Synthetic Data Generation

For the scope of this experiment a hypothetical HMI (*ht*-MI) application (Fig. 5.10a, Fig.5.10b) was built using *National Instruments Labview 8.0* standard function blocks. The *ht*-MI displays values of an arbitrary simulated process (5.4) as a pattern of 8 indication lamp states (*8-bit*), which the HMI user is required to visually track and manually set an array of 8 rocker style toggle switches either ON or OFF corresponding to each indicator lamp state being either ON or OFF. The process state evolves using equation 5.4 - a first-order linear autoregressive (AR

or α), moving average (MA or β) process, with Gaussian random noise (ϵ_t) and an adjustable period of sinusoidal seasonal component. In addition, modulus of natural logarithm is used to introduce an auto-resetting trend component to TS. The parameters of the equation 5.4 are listed below in (5.5).

$$Y_i = \mu + \alpha Y_{i-1} + \beta \epsilon_{t-1} + \epsilon_t + \sin \frac{2\pi t}{\kappa} + ln[(t - \kappa) \left\lfloor \frac{t}{\kappa} \right\rfloor] \tag{5.4}$$

Process Parameters

$$\begin{aligned} \text{Gaussian random noise}(\epsilon_t) &: (\text{seed} = -1, \mu = 0, \sigma = 0.5) \\ \text{Seasonal component period}[\frac{t}{\kappa}] &= \frac{\text{time-step}}{200} \end{aligned} \tag{5.5}$$

This setup allows for generating two types of raw data sets under: manual entry and auto-pilot modes. In the manual entry mode (Fig. 5.10a) HMI user manually sets the switch states in response to the process indicator (lamp) values. Each time step sample (row) consists of a 2-tuple vector as shown in Fig. 5.11. In the auto-pilot mode (Fig. 5.10b) the user response is modeled using a PI (proportional-integral) controller with its proportional gain set each time step randomly within a fixed range. The range was determined by trail-error to closely match the author response rate observed in manual entry mode. Each HMI state value is restricted to *8-bit* value.

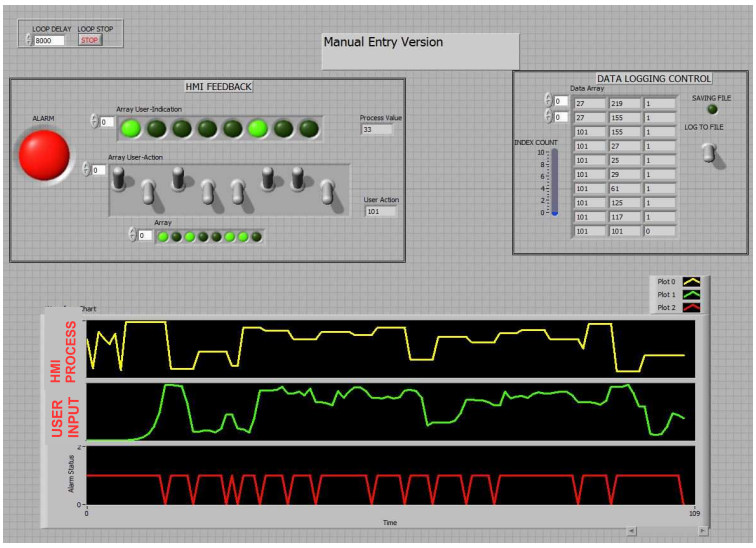
5.3.1 Raw Data Sets

The raw HMI data sets are generated from either two sources: manual and auto-pilot modes. Each data set (Fig. 5.11) contains approximately (adjusted as desired) $4K$ samples with columns: *Time*, *PROCESS*, *HMI_USER*. Time index is currently arbitrary stored as date strings while the data columns hold time sample tuples.

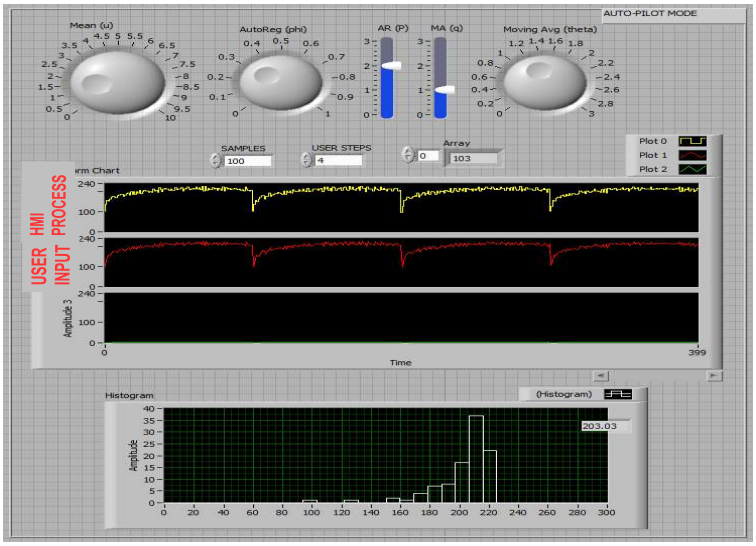
A pair of (training and test) HMI raw data sets are generated separately using auto-pilot mode with slightly different ARMA (α, β, μ) parameters (listed in (5.6) for the process generator (5.4) along with random proportional gain parameter for the (PI) operator response generator. These raw data sets are used to construct various supervised datasets as required by the various HMI forecast models implemented in this thesis.

For evaluating ARIMA models, the synthetic data generator ARMA parameters were kept consistent between training and test datasets (5.6), which effectively resulted in two datasets. These two have similar time-series mean, yet slight variance in the sequence of values owing to moving average gaussian noise term in the data generator process (5.4).

For evaluation of RNN/CNN time-series models the train and test pair datasets were gen-



(a)



(b)

Figure 5.10: *ht*-MI application in Manual and Auto-Pilot mode. Process values are generated using AR(p), MA(q) and gaussian noise process with trend and seasonal components. (a) Manual mode: Process values displayed via array of lamp indicators. User tracks indication patterns by manually setting toggle switches in same pattern until alarm indication (red lamp) goes off; (b) Auto-Pilot mode: User tracking response to process values is modelled using a PI controller with random proportional gain, which draws inspiration from the linear servo control model by *Tustin* [42]. HMI process and user response is captured as time series data

erated using slightly different synthetic data generator ARMA parameters (5.4) for training and test datasets (5.6). This effectively resulted in two datasets with different overall time-series mean, but similar patterns owing to the same PI (integral action constant).

For the NLP model evaluation the resulting two raw data sets were framed (arranged)

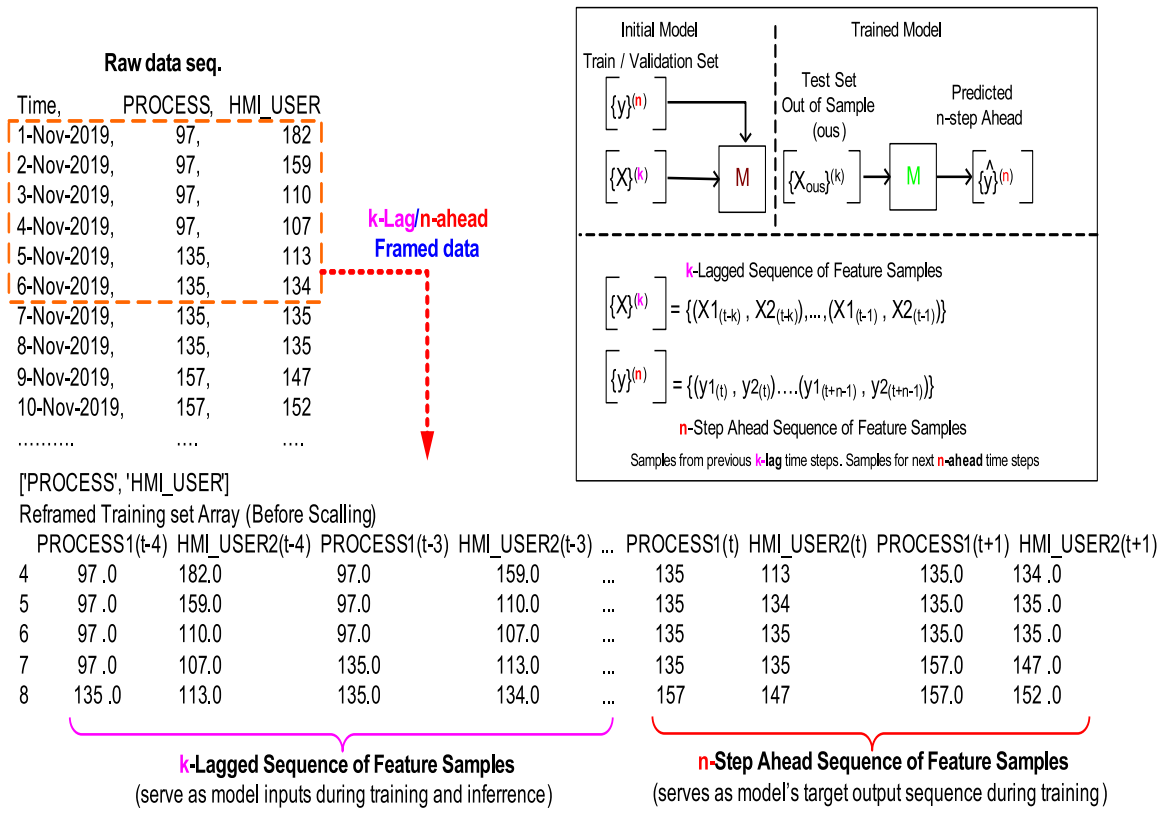


Figure 5.11: Snippet of raw time-series data set generated from *ht*-MI application. *PROCESS* and *HMI_USER* values are restricted to 8-bit integers (arbitrary time stamps generated). The raw data series are framed into supervisory training sample sequences as desired for *k-lag* and *n-ahead* pair of values. Each consecutive row or sample sequence is produced as a rolling window containing atleast $k + n - 2$ previous duplicate token values.

into (k lag and *n-step* ahead) supervised training and test time series sample data sets with the effective series mean constrained to range 198 to 255 as shown in Fig. 5.12. The rationale for keeping ARMA process generator parameters the same for both training and test data sets is to ensure the synthetic data sets are consistent and do not fall outside the fixed vocabulary size (HMI states: 0 to 255) pre-selected for this experiment. The test data set contains similar HMI process patterns but slightly different HMI user patterns as obtained by changing the PI integral action reset constant, from 4 to 6 steps (5.6). This effectively lowers the auto-pilot operator response rate, which is conducive for evaluating out of sequence prediction performance of the trained NLP models.

HMI Synthetic data generation AR-MA parameters

$$\begin{aligned}
&\texttt{For ARIMA Models}\\
&\text{Training/Test Set:}(\alpha,\beta,\mu)=(0.6,-0.1,4),\text{mean=198}\\
&\texttt{For RNN/CNN Models}\\
&\text{Training Set:}(\alpha,\beta,\mu)=(0.3,-0.8,4),\text{mean=201}\\
&\text{Test Set:}(\alpha,\beta,\mu)=(0.6,-0.1,10),\text{mean=350}\\
&\texttt{For NLP Models}\\
&\text{Training/Test Set:}(\alpha,\beta,\mu)=(0.6,-0.1,4),\text{mean=198}\\
&\text{Traning/Test:}(PI\ Iintegral\ reset)=4,6\ \text{Steps}
\end{aligned} \tag{5.6}$$

5.3.2 Supervised Learning - Data Framing

Supervised learning often requires the raw data sets to be re-framed as input and target output data sets in the representation of the underlying forecast problem to be modeled. For this experiment, the raw time-series HMI state feature data is framed as a short sequence of values arranged from previous k time steps (Fig. 5.11) - referred here with notation $\{X\}^k$ as the training sequence (containing both process X_1 and user input X_2 features). This lagged sequence of patterns serves as the training, validation and test data sets for the required prediction model.

Similarly, the ground truth or target model output (or labels) is framed as a short sequence of values taken from next *n-step* ahead times - referred to here with notation $\{y\}^n$ (containing both process and user input features). The *n-step* ahead sequence of patterns also needs to accompany the training and validation sets as expected output patterns for the required supervised training.

Finally, trained prediction model shall be able to predict *n-step* ahead samples of HMI state feature vector (Fig. 5.11) - referred to here with notation $\{\hat{y}\}^n$, when k-lagged samples ($\{X\}^k$) are provided as input.

5.3.3 Baseline Model - Persistence Score and Rolling Window RMSE

Persistence model (also referred to as naive or walk-forward forecast) is used to ascertain the baseline forecast error estimate for a given time series data. Persistence algorithm shifts (lags) a given time-series data by p steps in time, and uses it as the input to an ideal prediction model (Fig. 5.13). An ideal prediction model outputs the original data series exactly p steps out of phase.

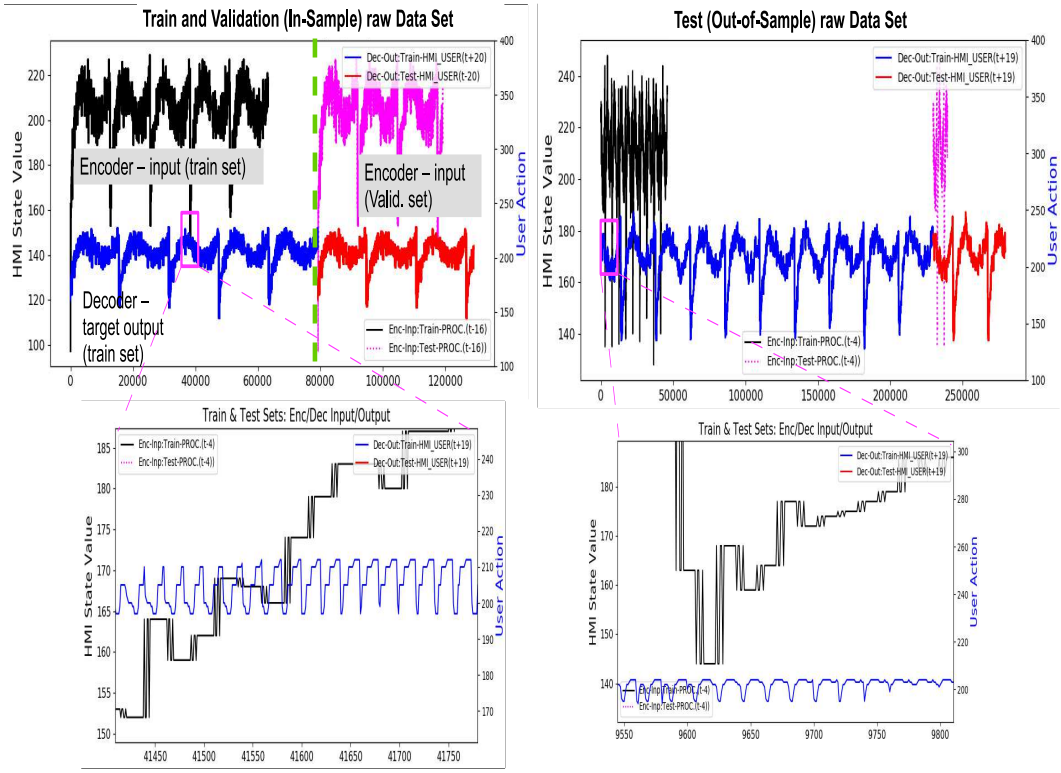


Figure 5.12: Raw Training/Validation and Test data sets for doing In-Sample and Out-of-Sample model performance evaluation, respectively. Lower row of plots are zoomed-in sections of the over-all sequence patterns to show the underlying patterns as an example. Test data set uses a slightly slower HMI operator response: The PI controller's integral constant is changed to 6 steps compared to 4 steps used for the original training set.

Persistence forecast root-mean-square ($RMSEp$) error which, is dependent on the correlation between the lagged samples, is calculated, taking into account all predicted and observed samples.

For example, when $p = 0$ the $RMSEp = 0$ is as expected, i.e. the persistence model will output the next step sample value when no shift or lag. *RMSEp* value is used as an upper bound for the forecast error or selection criteria for any candidate forecast model to be considered skilful, i.e. the models must yield a lower forecast error ($RMSE < RMSEp$) than the persistence error for same *n-step* ahead prediction.

$$RMSErw = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \frac{1}{W}\sum_{k=0}^{W}\hat{y}_k)^2}{N}} \tag{5.7}$$

*Where **W** - temporal slice size, **N** - total number of samples*

Rolling window persistence *rollWin.RMSEp* score is also calculated (5.7) taking into account previous W predicted sample values that fall in the same temporal slice (Fig. 5.14) as

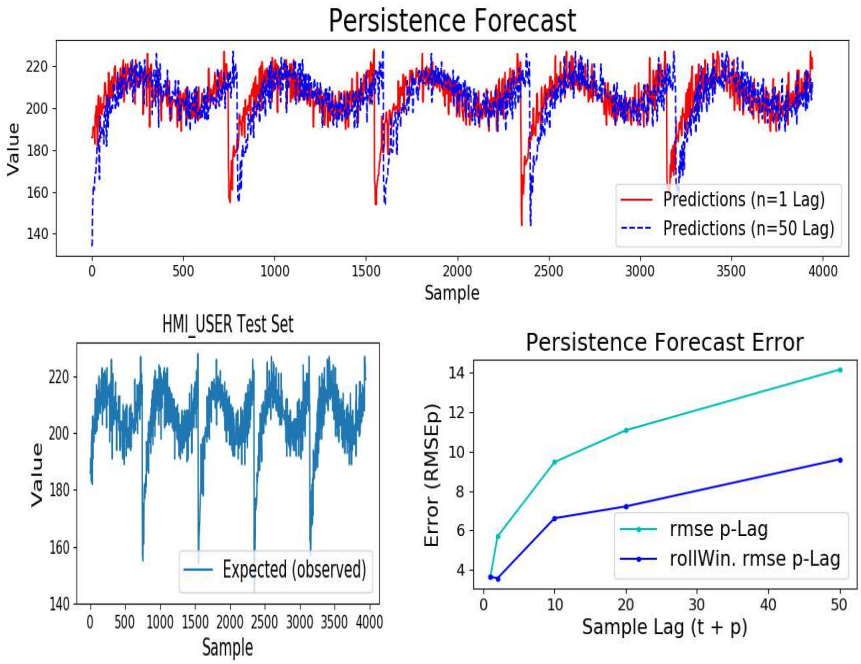


Figure 5.13: Persistence (RMSEp) score for a training/test data set for p-lag Persistence model forecast samples.

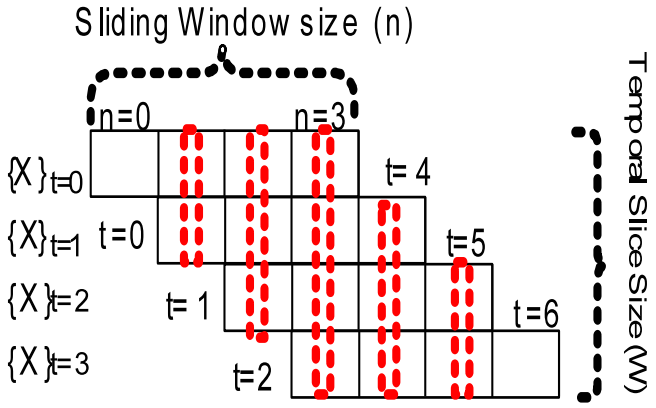


Figure 5.14: Rolling Window Forecast for n-ahead (E.g. n=3) Forecast for each input sequence $\{X\}_t$. Previous W predicted samples from each temporal (red vertical stripes) slice in sliding windows can be combined appropriately (E.g. averaged, max or min polled, etc) together for calculating rolling window forecast errors. Note: Sliding window size n equals Temporal slice size W while running n-ahead forecast.

the *n-step* window advances one time-step ahead over the next input feature $\{X\}^k$ sequence. Rolling or sliding window forecast generally tends to yield a lower *rollWin.RMSEp* score (Table 5.3), as *RMSE* is calculated for each window of observed samples and then all such inter-

Table 5.3: Persistence Score *p-Lag* RMSEp

n-Lag	*RMSEp*	*RMSEp (roll.Win.)*
1	3.75	3.75
2	5.69	3.56
10	9.48	6.62
20	11.08	7.21
50	14.16	9.61

mediate rolling window *RMSE* values are averaged to obtain one final *rollWin.RMSEp* value.

For instance, the persistence forecast model for single step (t + 1) ahead samples with respect to original (t) HMI data series, yields RMSE_p = 3.75 score, which can be used as an upper bound forecast error for any (t+1) 1-step ahead forecast model that is to be selected. Similarly, RMSE_p for other (t + 2; t + 10; t + 20; t + 50) were generated RMSEp values shown in Table. 5.3. For example, Fig. 5.13 compares both t + 1 and t + 50 forecast.

$RMSEp$ scores for lags (t + 2; t + 10; t + 20; t + 50) are as shown in Table 5.3. Fig. 5.13 compares both t + 1 and t + 50 lag persistence forecast and also depicts both $RMSEp$ and rolling window $RMSEp$ as it varies with *p-lag* over the test data set that has been used.

5.4 ARIMA Model Test Setup

Its envisioned that industrial operations can draw potential benefits from non-intrusive automatic monitoring of operator situational awareness using advances such as ViDAQ [160] and EYE-on-HMI framework [7]. This study further evaluates the feasibility of using existing stochastic techniques to model required operator response with industrial HMI states (Fig. 3.2) as time-series data. Furthermore, the assumptions outlined in section (Sec. 3.3.2) allows HMI state TS data to promise of being weakly stationary in nature, thus allowing the application of ARIMA forecast models. Ultimately, the goal of the HMI state forecast models is to monitor and predict deviations in operator actions. This section further presents the results of all tests, as mentioned in the previous section (Sec. 4.3.3).

5.4.1 Preliminary Training Data Analysis

Firstly, the *HMI_USER* data set generated from the test HMI in auto-pilot mode (Fig. 5.10b) is qualitatively assessed for stationariness. In Fig. 5.15, the first plot shows the entire TS data set, which is overlaid with its rolling window mean and standard deviation. As expected the (5.4) represents a TS with a seasonal trend (as mean changes with time), but the rolling window standard deviation does not appear to change with time which, suggests this TS is

not a random walk. Thus, the resulting dataset using (5.4) can be modeled for forecasting.

Autocorrelation (ACF) computes the effect of previous (lag) observations on current values, whereas Partial ACF (PACF) eliminates the effect of intermediate lag terms. Here, the sample autocorrelation (ACF) plot decays slowly, which also confirms a seasonal trend and that the TS is definitely non-stationary. Otherwise, stationary TS exhibits a rapidly decaying ACF. Here, PACF decays quickly to a lag value ($q = 5$) and trails off to zero, indicative of a moving-average *MA(q)* process of lag $q = 5$. The Histogram, Kernel density (KD), and QQ plots visually convey how much the sample distribution deviates in comparison to a Gaussian spread which, an ideal stationary TS exhibits. Here, the long tail in KD and histogram also indicates there is quite a bit of predictable information in trend component which must be removed i.e., in order to transform this raw data TS to a stationary TS.

The second set of plots in Fig. 5.16 are similar to Fig. 5.15, but are generated by taking the *log* of the difference of the original TS data with its one step lagged version. This is done to remove the seasonal trend component as apparent in the original TS data set (Fig. 5.10b). The first order differenced TS has a mean and standard deviation that does not vary with time, indicating the resulting TS data is closer to being a stationary. This is further validated by looking at the Histogram, QQ and Kernel density plots, which show a strong comparison to the ideal Gaussian (bell shape) distribution. Both ACF and PACF decay rapidly and show a hard cut-off after $lag = 4$. That aids in determining respective lag parameter (*p* and *q*) values for AR($p = 4$) and MA($q = 4$) processes from ACF and PCAF plots respectively.

Therefore, from above analysis a first order difference is sufficient to make the *HMI_USER* data set generated from the test HMI in auto-pilot mode (Fig. 5.10b) stationary and qualified to be modelled by *ARIMA(p,d,q)*, where *(p,d,q)* have been estimated as per plots in Fig. 5.16 (E.g. $(p, d, q) = (4, 1, 4)$. These parameter values were also confirmed programatically using the exhaustive grid search optimization.

5.4.2 In Sample (*InS*) Forecast

In-Sample (*InS*) forecast is done using a time series prediction model (e.g., ARIMA or SARIMA) to generate predictions for the time range that it has already seen during its training or fitting phase.

Static 1-Step Ahead

The basic *InS* 1-step ahead forecast was done using a static ARIMA model (Fig. 5.17), where the model is not re-trained prior to predicting next step prediction. Root-mean-square (*RMSE*) error (or square root of variance) is used to compare prediction error and is calculated with

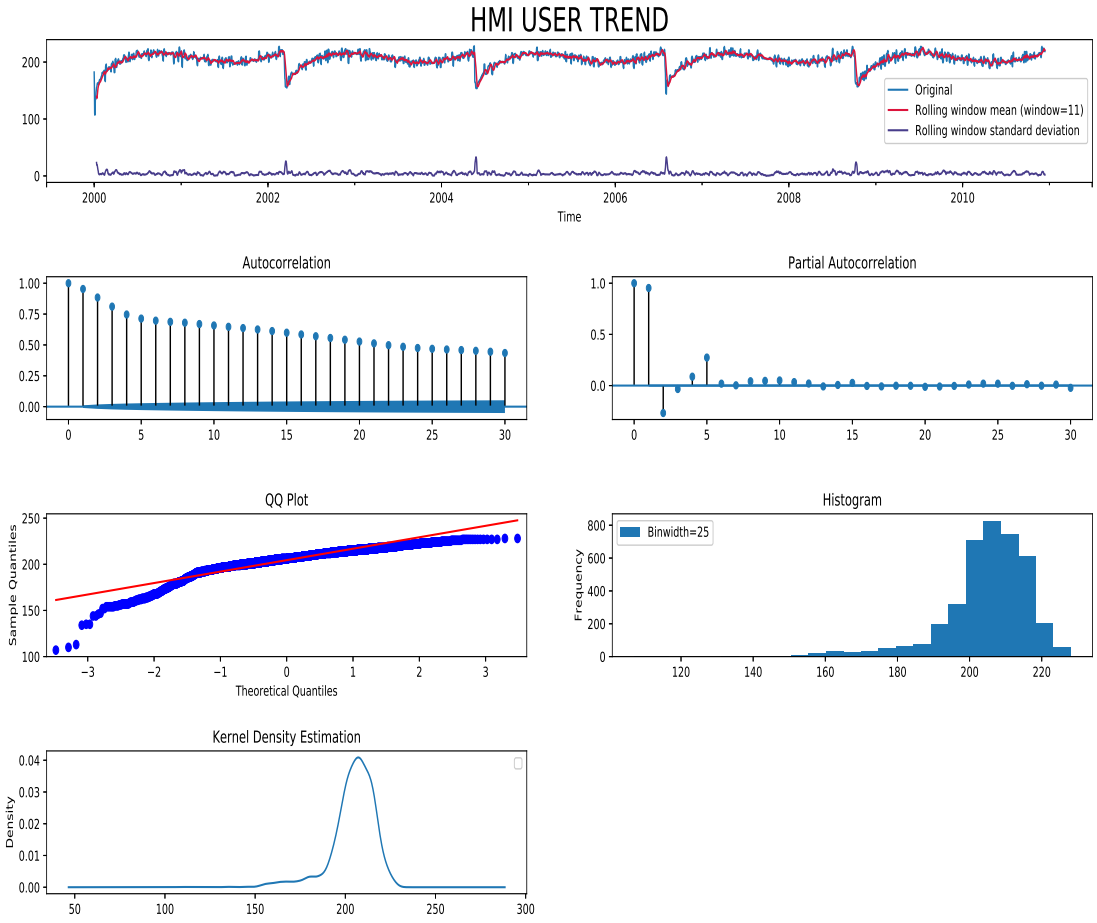


Figure 5.15: Diagnostic Plots of Original HMI User dataset. Moving average mean shows seasonal trend therefore TS is non-stationary. TS is not *Random-walk*, since rolling standard deviation (variance) does not vary with time. ACF decays slowly indicative of seasonal trend. PACF indicates a MA process of finite lag. Quantile (QQ), Histogram and Kernel density plots indicate a Gaussian like distribution with a long tail suggesting presence of a periodic trend component in TS.

respect to the actual observed values. Simulation showed a $RMSE = 3.5$ for *InS* 1-step ahead forecast using static model (zoomed trend shown in Fig. 5.18) tracked the actual observed values closely. As expected, this model showed a slight performance improvement over the above reported persistence model for *1-lag* i.e. $< RMSE_{p1} = 3.75$.

Static N-Step Ahead

In this test, the static ARIMA model is used to make InS n-step ahead predictions for every observed sample in the training set (Fig. 5.19). Evaluating any n-step ahead forecast requires using metrics calculated using a rolling (sliding) window metrics: E.g., rolling window aver-

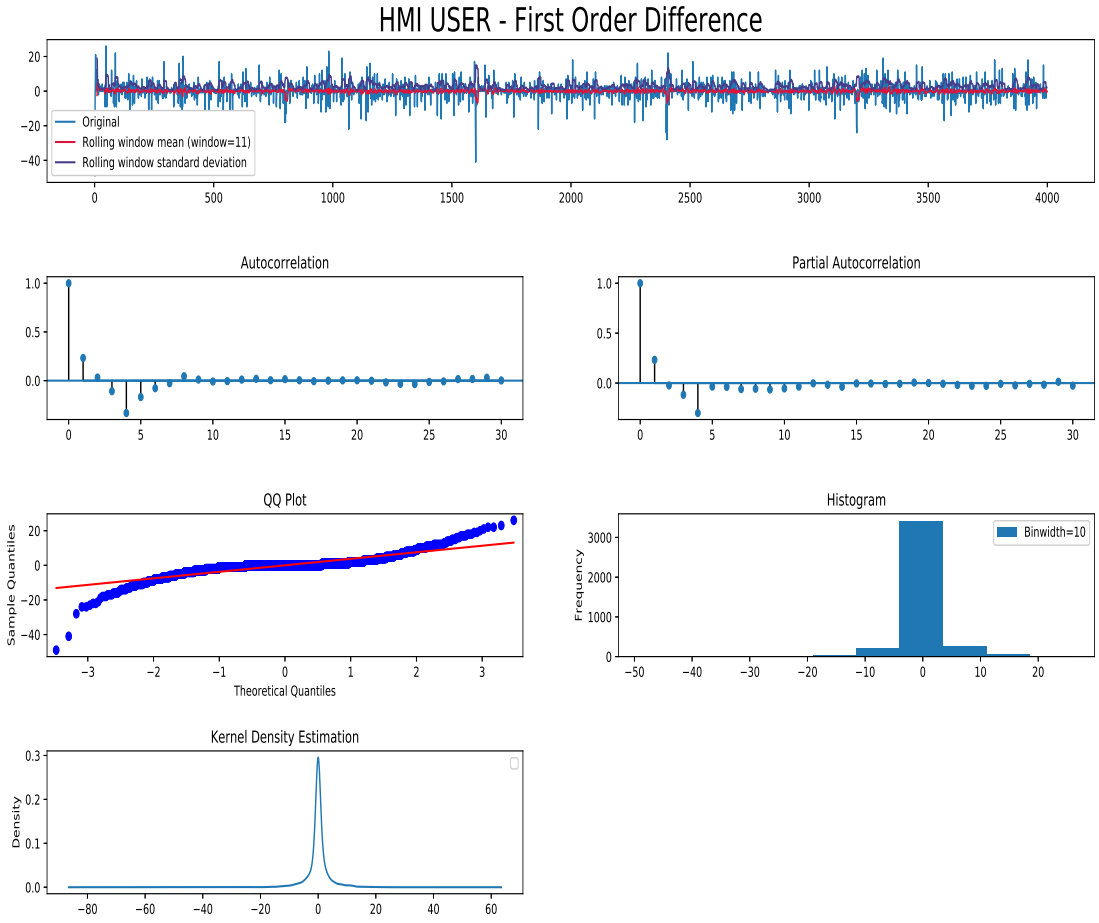


Figure 5.16: Diagnostic Plots of *Log* First Order difference of HMI User dataset. Moving average mean no longer shows seasonal trend therefore TS is closer to begin stationary. Moreover, both rolling mean and standard deviation (variance) does not vary with time. ACF decays rapidly and hard cut off indicative of *AR(p)* process. PACF also shows rapid decay after $lag = 4$ indicating *MA(q)* process. Quantile (QQ), Histogram and Kernel density plots indicate a more Gaussian like distribution with negligible tail suggesting trend component has been minimized significantly.

age and rolling window RMSE. Since an n-size rolling window, when advanced one time-step ahead, should take into account or combine previous $(n-1)$ duplicate (intermediate) predicted samples that were generated from each n-step ahead prediction.

The performance of the static n-step ahead model is evaluated by directly comparing the mean of the rolling window RMSE trend to the corresponding n-lag persistence *RMSE* value. Table. 5.4 shows a consistent ($RMSE = 3.4$) forecast error value that is independent of n-step *InS* ahead forecast using the static ARIMA model. The rationale for this performance is owing to the use of *InS* lagged values from the training set to generate next n-step ahead predictions; this prevents the forecast error from accumulating, i.e., or be independent of n-steps. In

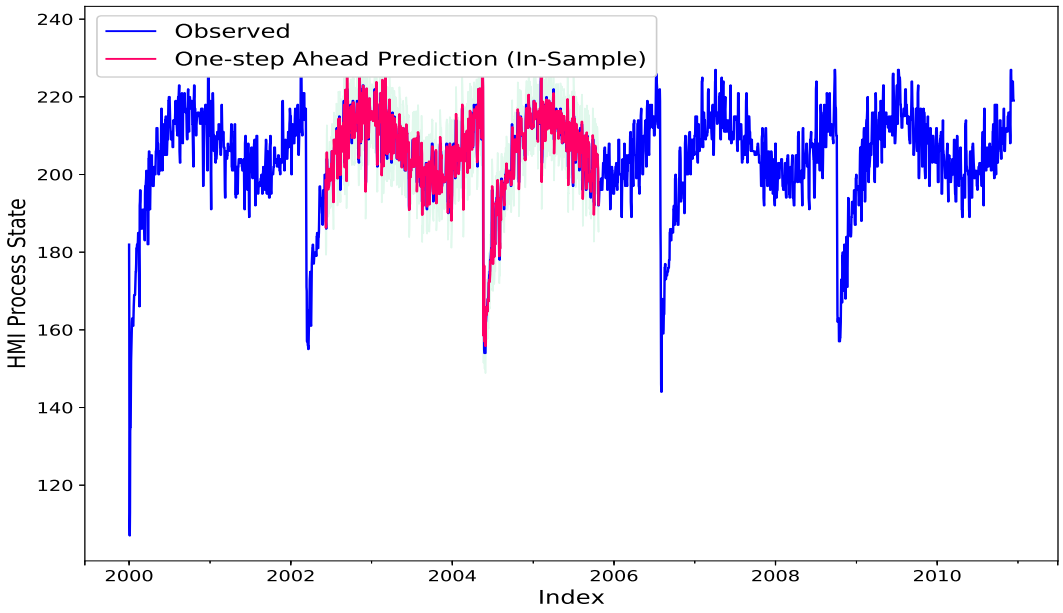


Figure 5.17: In-Sample (InS), Static 1-Step Ahead Forecast TS. TS (blue) Shows observed actual values of the training set. TS (red) shows the InS forecasted values which closely track the training set. Light blue background shows confidence bounds for the forecast TS.

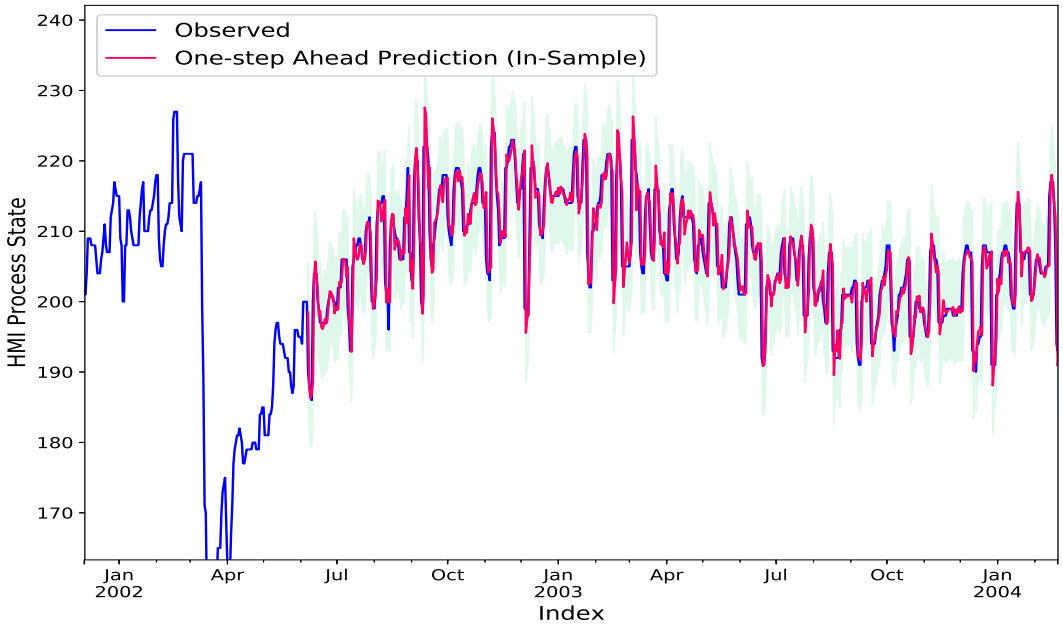


Figure 5.18: In Sample, Static 1-Step Ahead Forecast (Zoomed) with prediction confidence bands (light green background).

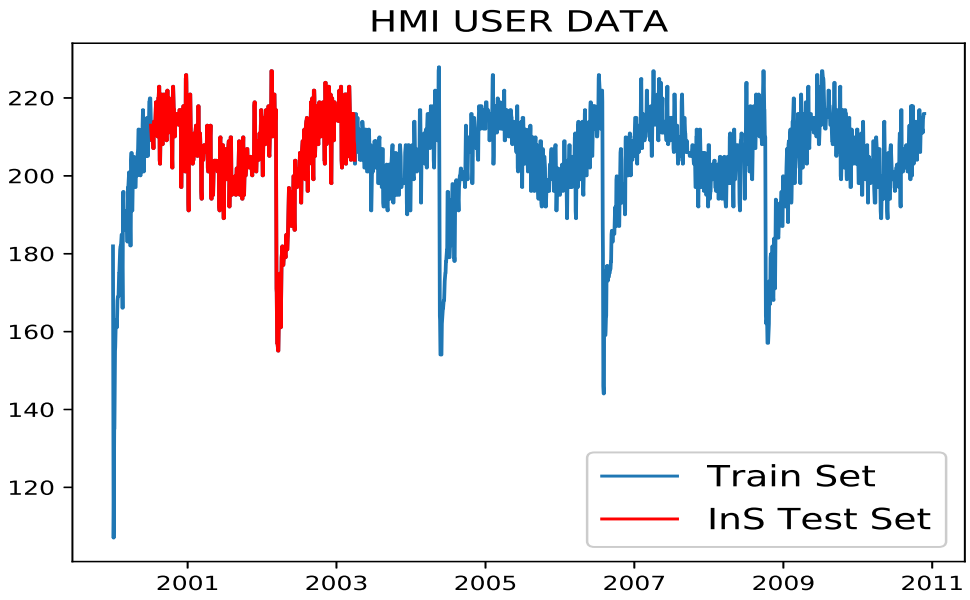


Figure 5.19: HMI States Time Series Training Set from which the In-Sample Test subset is obtained.

Table 5.4: InS Static n-Step RMSE

N-Step	RMSE
1	3.41
2	3.39
10	3.42
20	3.40
50	3.41

addition, the performance is quite similar to that obtained as above for *InS* 1-step ahead static ARIMA forecast. Fig. 5.20, shows an example of InS 50-step ahead static ARIMA forecast. The output predicted TS data shows closely tracking the *InS* training set. The confidence error bound, too, shows a consistent variance over the full range of the forecast. Rolling window average trend shows the predicted samples match temporal and scale variations with respect to the original test set. Rolling window RMSE trend (bottom left of Fig. 5.20) shows sharp spikes proportionate to the corresponding size of each HMI state transitions. This is an artifact of using rolling window computation, which combines errors of previous duplicate samples.

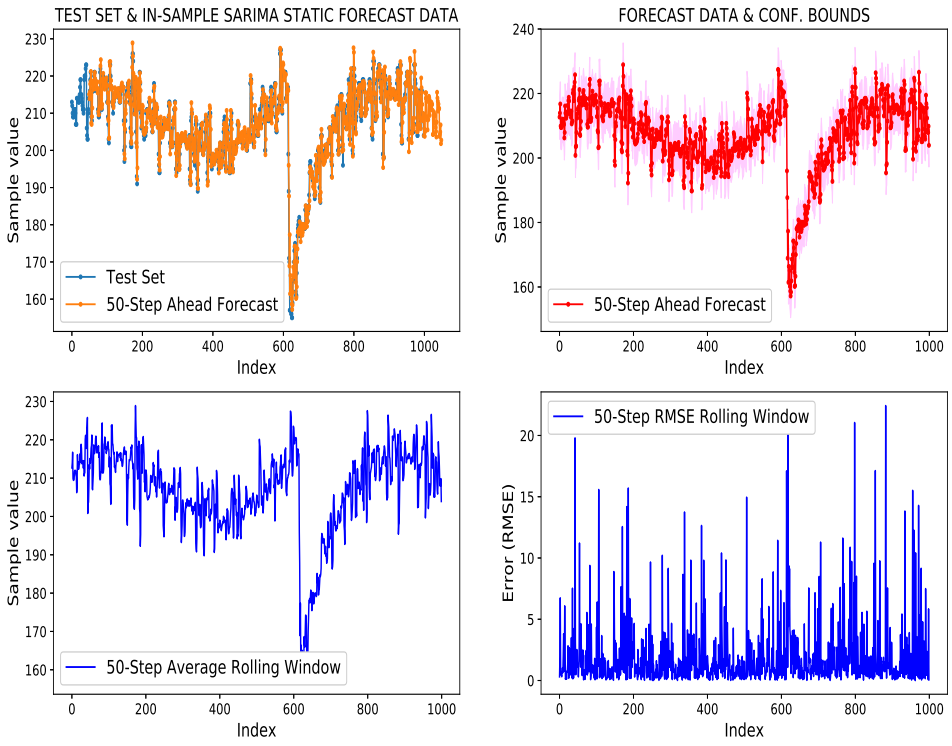


Figure 5.20: In-Sample (InS) Static 50-Step Ahead Forecast. Top left plot shows 50-step ahead predicted samples overlaid on the actual *InS* test set. Top right plot shows the forecast TS with per sample confidence bounds. Bottom (right and left) plots show rolling window metrics: Average and RMSE respectively for the 50-Step ahead forecast.

Static Dynamic N-Step Ahead

Using the static ARIMA model with dynamic mode prediction recursively uses previously predicted samples instead of the lagged actual training set values to make the next step *InS* predictions. As seen in Fig. 5.21, the prediction shows a deviation trend compared to the actual training set, even though only a 1-step ahead prediction is being used repeatedly stepwise over the full training set. The rationale for this performance is due to each $(t-1)$ predicted input values being fed to the static model contain prediction errors from when these were generated. The error tends to accumulate and amplify in each next time step predicted sample. Hence, the prediction confidence bound (light yellow background) grows linearly with n samples, which suggests a naive dynamic forecast is not feasible for making long-range predictions. Simulation result for 1-step ahead dynamic forecast results in $RMSE = 21.72$ over the full range of training set.

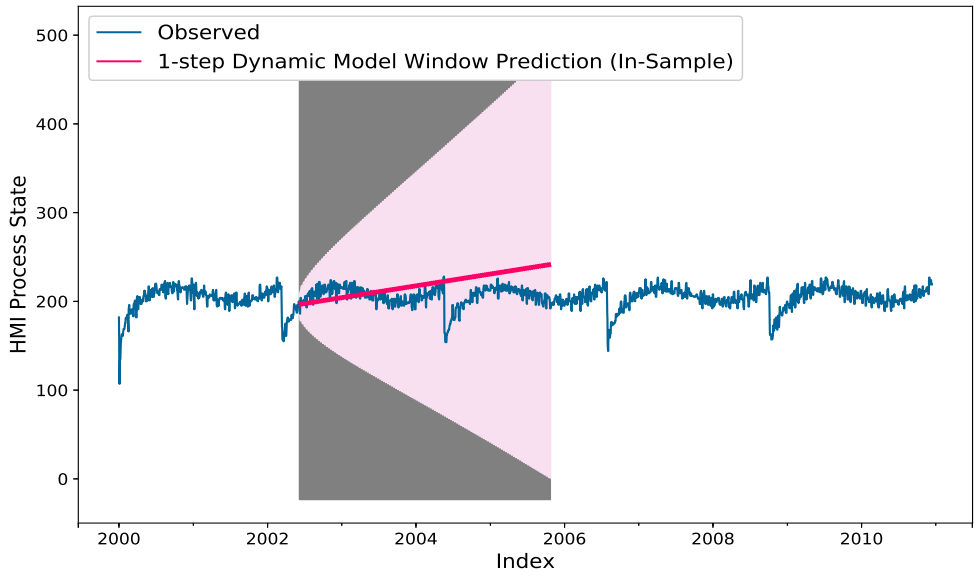


Figure 5.21: In-Sample 1-Step Static Model and Dynamic Mode Forecast with confidence bands (off white background). Forecast deviates with a constant up trend and is not even scaled appropriately.

Static Dynamic with Exogenous Input

Multivariate time-series Auto-regressive (ARIMA) prediction models can also be trained with exogenous (external) time-varying predictor variables. However, exogenous predictor variables must satisfy unidirectional causality, i.e., only the exogenous variables (X_i) should affect a change in Y (variable to be predicted), and Y should not affect any change in X_i. This condition is usually tested by the granger-causality [94] test, which tests a statistical hypothesis for determining whether one-time series is useful in forecasting another. In this study, causality is ensured as per the synthetic data generation setup (Sec. 5.3). The soft HMI (Fig. 5.10b) in autopilot mode generates $PROCESS$ TS data independently of the HMI_USER TS data. Thus, only HMI_USER data or simulated user response is affected by the current HMI $PROCESS$ state value.

In this test, a static ARIMA model is fitted once on the training set consisting of both HMI_USER and its companion exogenous (Exog.) predictor, PROCESS (example shown in Fig. 5.22). Then, it is evaluated by generating in-sample 1-step ahead predictions using both the lagged values from actual training set (containing both HMI_USER and PROCESS (exog.) variable) and using past predicted values (dynamic mode). Figure 5.23, compares the predicted 1-step ahead series using static model in standard and dynamic modes. The static

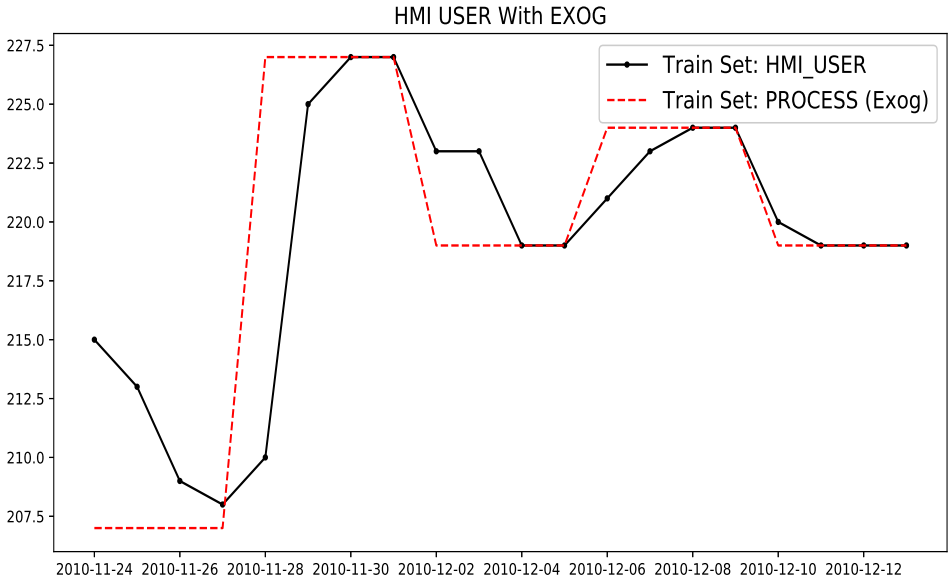


Figure 5.22: Example showing HMI_USER (variable to be modelled) and its Exogenous predictor variable: PROCESS state.

model in standard mode seems to show a close tracking with the training set with resulting $RMSE = 4.46$ as the forecast error. The static model under dynamic mode attempts to follow the training set; however, the predicted series is significantly scale attenuated, resulting in $RMSE = 6.73$ as the forecast error. Using the Exog. predictor has shown significant improvement in the dynamic mode forecast when compared to previous Fig. 5.21 (Sec. 5.4.2) 1-step ahead dynamic mode forecast without using a Exog. predictor.

5.4.3 Out-of-Sample (*OuS*) Forecast

Out-of-Sample (*OuS*) forecast is done using a time series prediction model (e.g. ARIMA or SARIMA) to generate predictions for the time samples that the model has not seen during its initial training or fitting phase.

Adaptive 1-Step Ahead

As an extension of using a static ARIMA model, where the model is not re-fitted, the adaptive model is re-fitted prior to generating the next step forecast. Re-fitting is done on the new data set that has been seeded with the recent actual observed value(s). Due to the computational overhead (owing to re-fitting), the adaptive model is feasibly used for making *OuS* predictions. The adaptive 1-step ahead forecast was evaluated for generating a few number of out-of-

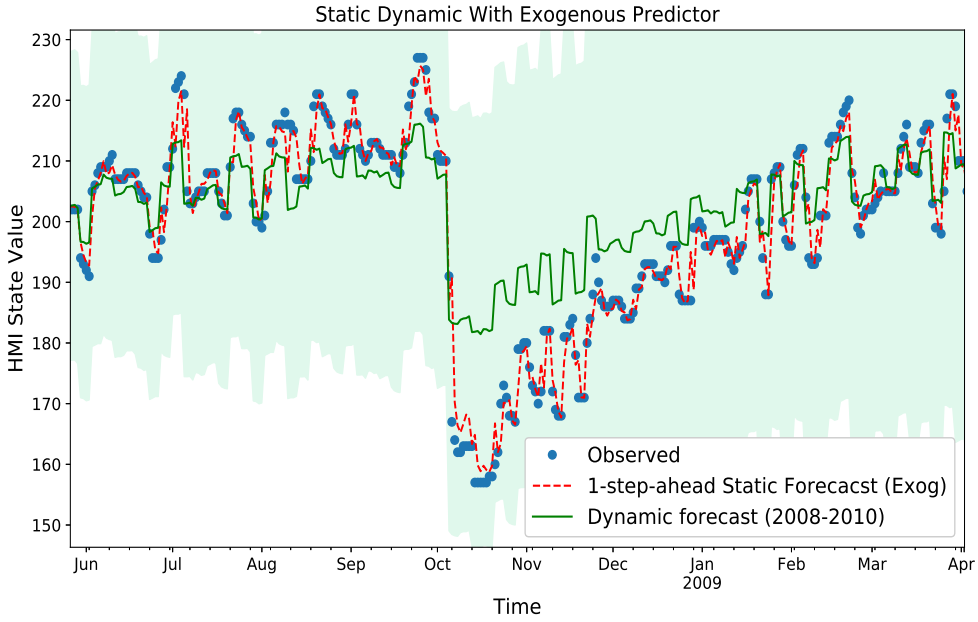


Figure 5.23: In Sample 1-Step ahead Static with normal and Dynamic mode Forecast With Exogenous Predictor (PROCESS) variable. Forecast sample confidence bounds shown in light blue envelope.

Table 5.5: OuS Adaptive 1-Step RMSE

Sample Run	RMSE
10	1.82
20	3.79
50	2.93
200	3.18

sample durations: $10, 50, 200$. In Fig. 5.24 only result of 200 sample duration forecast is shown and the RMSE values for various sample run durations evaluated are listed in Table .5.5.

The forecast samples show a close tracking with respect to the evaluation test data set. Owing to re-fitting, the model adapts to the new samples appropriately, and the rolling window RMSE trend shows spikes due to sudden value transitions in the evaluation test set. Since, all tests runs are done using a 1-step ahead forecast, the resulting RMSE values (Table .5.5) are only compared to required 1-lag persistence RMSE value ($< RMSE_{p1} = 3.75$). Lower RMSE values suggest acceptable performance of the adaptive model for doing out-of-sample 1-step ahead predictions.

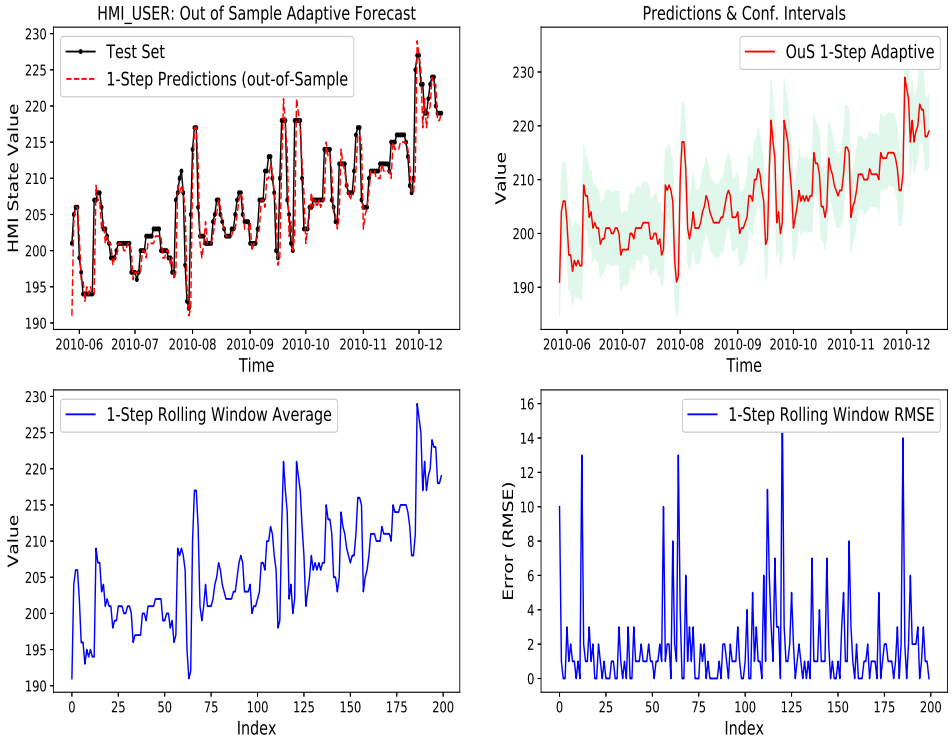


Figure 5.24: Out-of-Sample Adaptive 1-Step Ahead for 200 Sample Run.

Adaptive N-Step Ahead

The next evaluation is done using an adaptive model for n-step ahead predictions for various sample run durations as listed in Table. 5.6. Similar to 1-step ahead, the adaptive model is re-fitted with current (t) observed sample being appended to the original training data set added. In adaptive prediction (Sec. 5.4.3), n-step ahead predicted samples are generated with each new Out-of-sample from test set. In addition, as stated previously, all duplicate (having same time indexes) predicted samples are averaged using the n-size rolling window. In OuS adaptive model RMSE results show a desirable scaling in performance with respect to n-step ahead parameter. For example, ($n = 10$) 10-step ahead $RMSE = 4.97$ compared to ($n = 50$) 50-step ahead $RMSE = 5.2$ for 500 sample run is only a 4% increase in prediction error. In addition each n-step adaptive OuS resulting RMSE value is approximately less compared to the equivalent lag persistence $RMSE_p$ scores as listed in Table. 5.3, therefore indicating this to be an acceptable prediction model.

Additionally the RMSE value of n-step ahead prediction using an adaptive model is not conclusively related to the number of sample run (Table. 5.6) as it would have to depend on the

Table 5.6: OuS Adaptive N-Step RMSE

N-Step Ahead	Sample Run	RMSE
1	20	3.79
2	20	4.24
2	200	3.63
4	20	4.81
10	10	2.20
10	200	4.47
10	500	4.97
20	500	5.20
50	500	5.20

Table 5.7: OuS Adaptive Dynamic N-Step RMSE

N-Step Ahead	Sample Run	RMSE
4	10	3.01
10	200	5.09
1	500	5.02
4	500	5.45
20	500	5.42
50	500	5.67

nature of the training set data set being predicted. In general, longer n-step Adaptive forecast (Fig. 5.25 vs. Fig. 5.24) yield higher RMSE performance broader confidence bounds (shown as light blue background in top right plots). This is owing to the issue of scale attenuation in sample values generated in longer n-step ahead forecasts, which is apparent by similar amplitude range of the Test set trend with its corresponding forecast sample rolling window average trend in 1-step (Fig. 5.24) than in 50-step (Fig. 5.25) ahead forecast.

Adaptive Dynamic N-Step Ahead

Out-of-sample adaptive model using the dynamic forecast method is re-fit on the initial training set, to which previous $(n-1)$ predicted samples have also been appended. The RMSE values of this test is listed in Table. 5.7. Prediction performance is quite similar to what was observed for the standalone adaptive (Table. 5.6) forecast model, with the dynamic method generally yielding an overall higher RMSE, since the error tend do accumulate and amplify (as is the general case usually with dynamic predictions) over time. The dependence of performance error (RMSE) on n-step ahead parameter does not show any conclusive correlation to required prediction sample run length, E.g. for n-step = $4, 20, 50$, all show approximately $RMSE = 5.4$ for 500 sample run length.

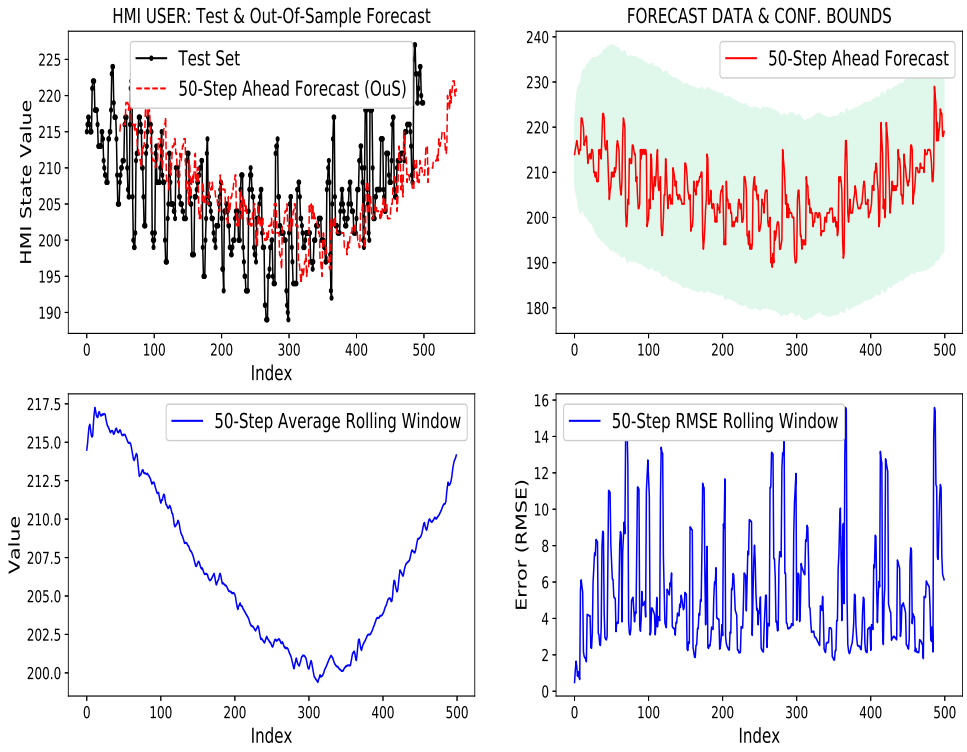


Figure 5.25: Out-of-Sample Adaptive n-Step Ahead for 500 sample run.

Adaptive N-Step with Exogenous Input

Out-of-Sample prediction performance with exogenous predictor variable (PROCESS) is evaluated with the adaptive model. Forecast error RMSE values of each n-step ahead forecast (shown in Table. 5.8) are smaller compared to RMSE for corresponding persistence lag values (Table. 5.3). However, the performance of the standalone adaptive model for OuS n-step forecast without using the exogenous input (Table. 5.6) is overall better than when exogenous input is used.

Adaptive Dynamic N-Step with Exogenous Input

Adaptive model using the dynamic mode of forecast with exogenous input, forecast error results is shown in Table. 5.9. Results indicate no marked improvement in performance by using the exogenous inputs in dynamic mode for adaptive models.

Table 5.8: OuS Adaptive Exog. N-Step RMSE

N-Step Ahead	Sample Run	RMSE
1	200	3.12
2	200	3.70
4	200	4.56
10	200	4.78
20	200	5.40
50	200	9.42

Table 5.9: OuS Adaptive Dynamic Exog. N-Step RMSE

N-Step Ahead	Sample Run	RMSE
1	200	5.40
2	200	5.96
4	200	6.48
10	200	5.92
20	200	6.34
50	200	9.62

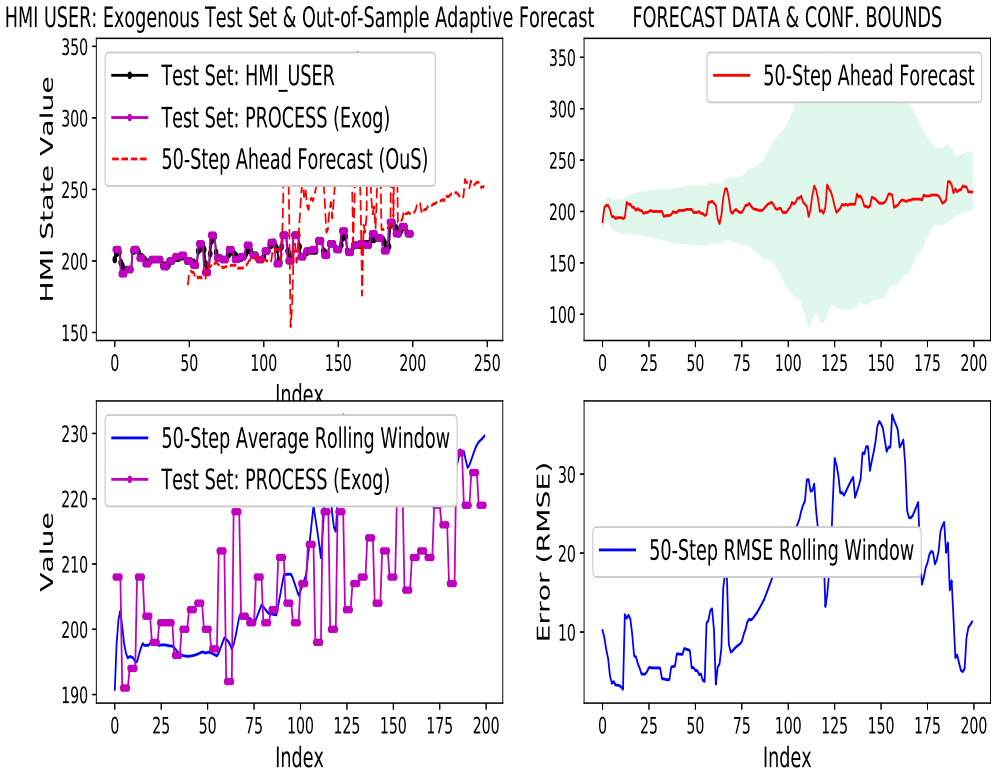


Figure 5.26: Out-of-Sample Adaptive 50-Step ahead 500 sample run with Exogenous Input.

5.5 ARIMA Summary of Results

The following observations can be made based on the results collected for ARIMA models under various tests, as depicted in Fig. 5.27.

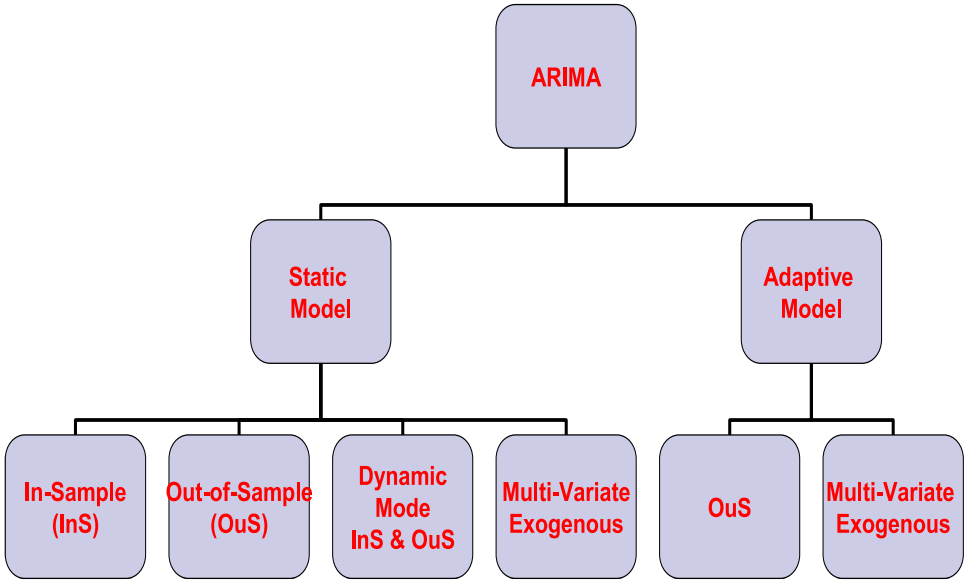


Figure 5.27: ARIMA Static model is only fitted once on the *InS* dataset, while Adaptive model is re-fitted with actual evaluation data as its doing forecasts. Static models were evaluated using *InS* and *OuS* data sets containing only the target feature (*HMI_USER*) being modeled. Exogenous data set contains multi-variate series containing two features: target *HMI_USER* and supporting *PROCESS*. Static and Adaptive models were also tested under Exogenous data sets. Dynamic mode is where model is recursively fed its own predicted output to generate forecast for *InS* forecast.

5.5.1 Static ARIMA Model

1. *In-Sample (InS) Forecast*: InS 1-Step ahead prediction using a static model shows comparable performance to 1-lag persistence model. Therefore it is acceptable to use this technique when available TS data is expected to be very similar to the initial training set that was used to fit the ARIMA static (where model fitted once) model.

2. *InS Uniform Performance*: Static models show a uniform prediction performance for InS with respect to the length of the forecast sample run. This is owing to the fact since the actual InS sample is used to make the next step prediction. In this case, the ARIMA model parameters are sufficiently optimal to make accurate 1-step ahead predictions for any length of required sequence run (as long as input samples are coming from previous training set).

3. *Out-of-Sample (OuS) N-Step Ahead Performance is Poor*: Unlike previous observation, the static model expectedly does not perform well, especially when demanded to do OuS N-step ahead predictions.

4. *Dynamic Mode for InS and OuS Performance is Poor*: Static model in dynamic mode where prediction recursively uses previously predicted samples instead of the lagged actual training set values to make next step *InS* predictions. This mode, when used with a static model for InS prediction, yields poor performance compared to n-lag persistence RMSE scores. OuS performance using the static model was not evaluated with this study. However, based on preliminary results, the poor performance of ARIMA for any OuS N-Step ahead predictions using dynamic mode is anticipated. This is owing to model not being re-fitted (as in case of the adaptive model), which causes forecast errors to accumulate and amplify as subsequent past forecast samples are used to make next step predictions.

5. *Static with Exogenous (Exog) Input Shows Improvement*: The performance of static models with exogenous input did show some performance improvement for InS prediction for both standard mode (where actual lagged values are used) and in dynamic mode when compared to simply using a standalone static model in dynamic mode.

5.5.2 Adaptive ARIMA Model

1. *Out-of-Sample (OuS) N-Step Ahead Performance is Good*: Adaptive model for N-Step ahead OuS prediction performs comparably to corresponding n-lag persistence RMSE scores. This is expected since the model is re-fitted each time with new OuS samples being appended to the original training set history.

2. *Dynamic Mode Performance is Stable*: Prediction performance using the adaptive model in dynamic mode shows consistent performance compared to just a standalone adaptive model for OuS n-step ahead predictions.

3. *Exogenous Inputs made Consistent Performance*: The performance of the adaptive model in dynamic mode with exogenous input did not show much improvement in overall performance. However, it yields a more consistent prediction error (RMSE) with respect to N-step ahead forecast even though the error is generally higher compared to when no exogenous input is used in adaptive model with dynamic mode.

Main take away from above results are briefly summarized in Table 5.10.

Table 5.10: ARIMA Model Result Summary.

Test	Result Highlights	Main Take Away
In-Sample (**InS**) Forecast using Static Model	• Static model (i.e. ARIMA model not re-trained) for next forecast • 1-Lag is acceptable (RMSE < RMSEp = 3.75). • N-Step gives uniform constant performance (RMSE ~ 3.41).	• Model inference skill is limited to initial training dataset only.
In-Sample (**InS**) Forecast using Static Model in Dynamic Mode	• Static model that recursively uses previous predicted samples to forecast next. • 1-step ahead dynamic forecast is not acceptable (RMSE > 21). • Static Dynamic with Exogenous variable shows marginal improvement.	• Model is very sensitive to bad forecasts. • Forecast error tends to accumulate and amplify over time.
Out-of-Sample (**OuS**) Forecast using Adaptive Model	• Model is re-fitted prior to generating next step forecast. • 1-Step ahead adaptive shows acceptable accuracy (RMSE < 3.2) for 200 sample run window. • N-Step ahead adaptive forecast shows higher RMSE.	• Model re-training incurs computational overhead for each forecast. • Doing longer (N-step > 50) the forecast are increasingly scale attenuated.

5.6 RNN and CNN Model Results

In this experiment, time-series: RNN - LSTM based and CNN based models were trained on supervised (k-lag/n-ahead) framed data sets and tested with separate (out-of-sample) similarly framed dataset.

5.6.1 Prediction Accuracy

Currently, the problem has been modeled as a regression time-series prediction type (Sec. 3.3.2), therefore rolling window Root-Mean-Square Error (RMSE) has been used as a metric to measure relative prediction accuracy. Whereas, Persistence rolling window $RMSEp$ scores for the desired n-ahead predictions (Sec. 5.3.3) is used to compare the relative prediction performance of various models.

In Table 5.11 prediction accuracy (RMSE) of all the LSTM models is listed for each combination of k lag and *n-step* values with corresponding $RMSEp$ scores also included for ease of comparison. The values in bold indicate lowest (best performing) and values highlighted in red are highest (worst performing) RMSE values for each row in Table 5.11. As evident, out-of-the-box performance of *Vanilla* (Fig. 5.28) and *Bidirectional* LSTM is better for all *n-step* ahead predictions in general, while *Convl.LSTM* showing potential performance gain (lower RMSE) for longer *n-step*$= 50$ predictions. In comparison, *CNN-Encoder* LSTM showed the worst performance, which may be attributed to using it in 1d CNN mode in contrast to *Convl.LSTM*

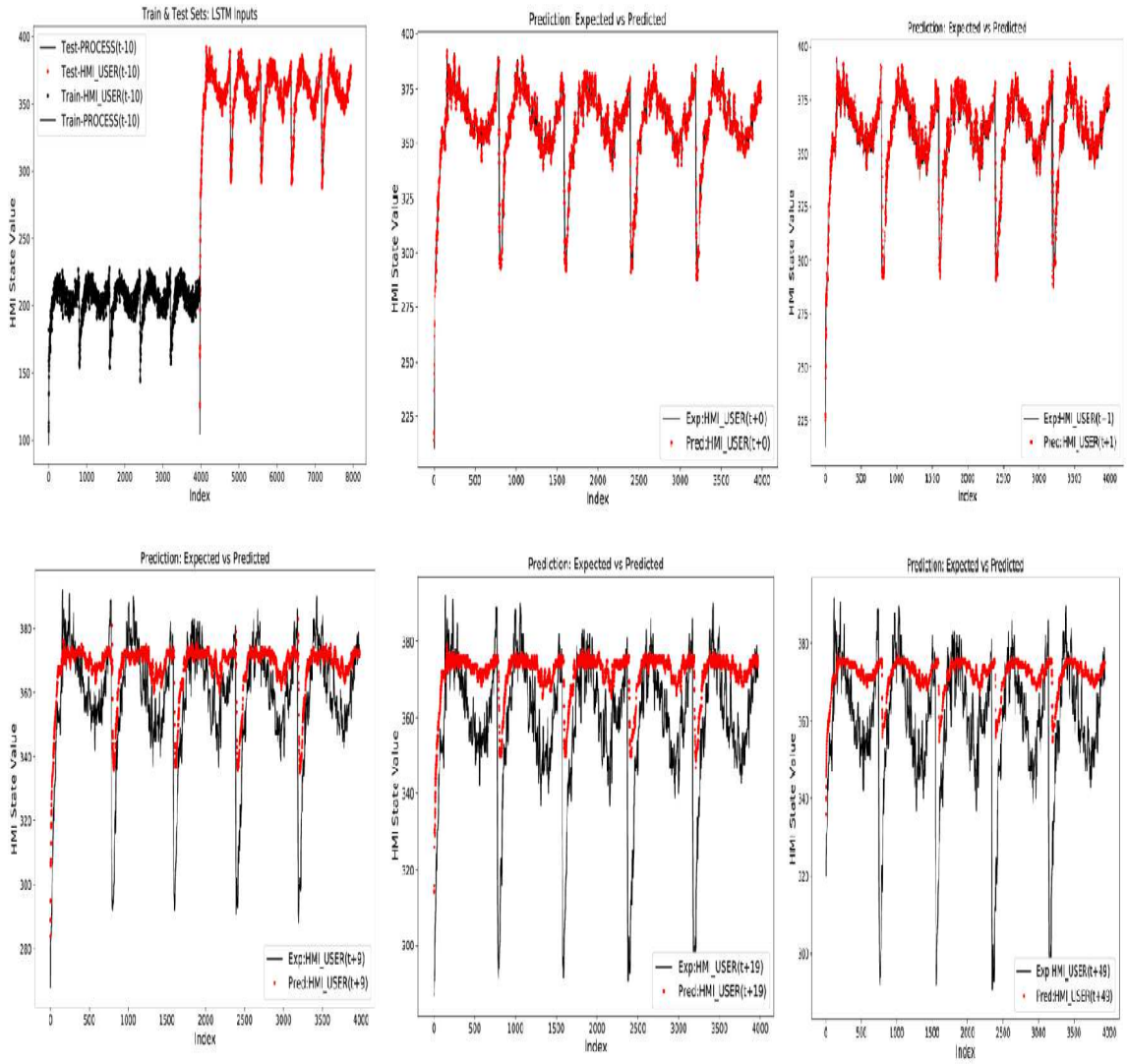


Figure 5.28: *Vanilla LSTM* Performance for $k = 10$ Lag and *n-step* (1,2,10,20,50) ahead forecast (y-axis are unit less HMI states). Prediction shown starting from second plot in top row left to right order.

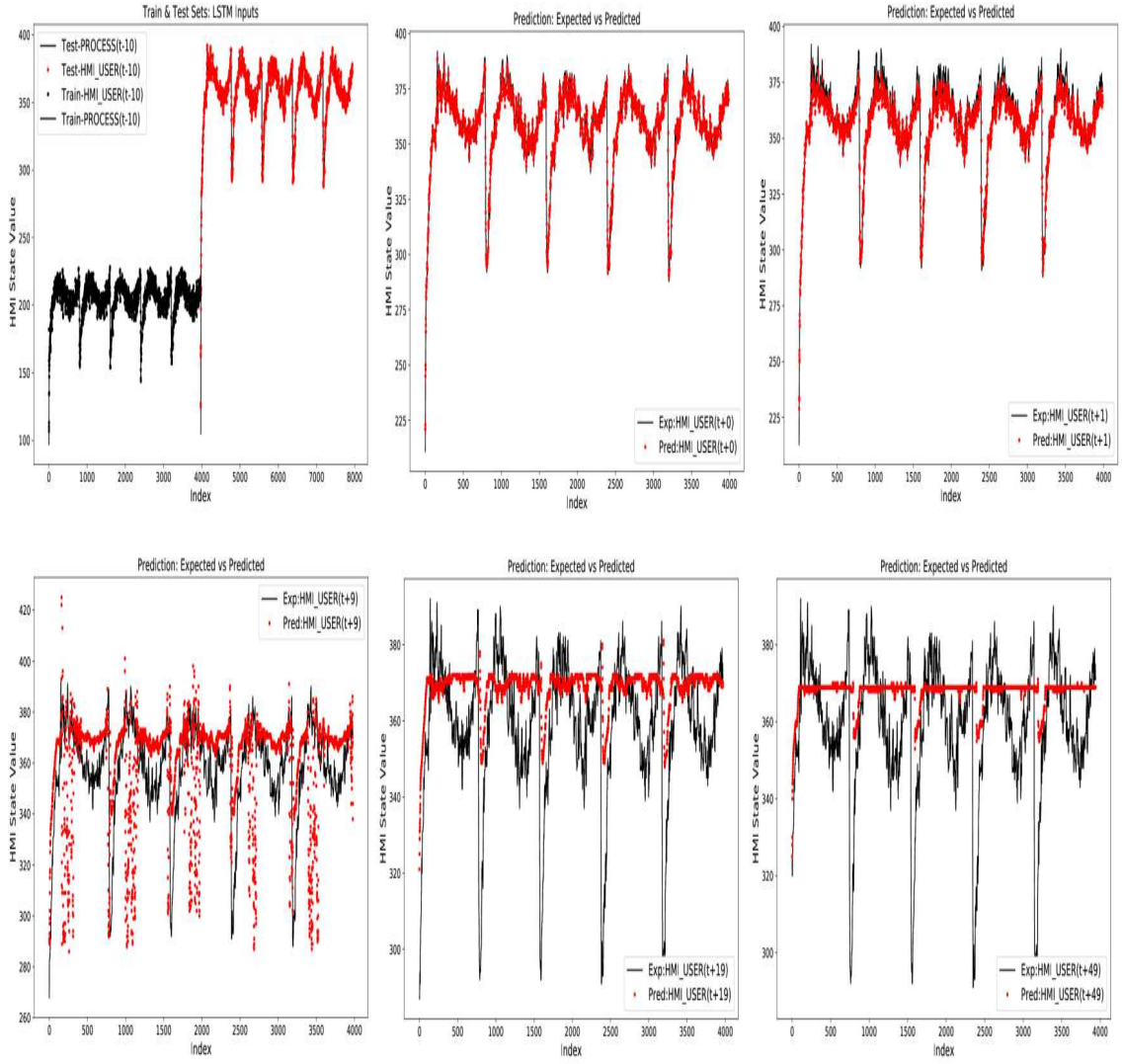


Figure 5.29: *Conv.LSTM* Performance for $k = 10$ Lag and *n-step* (1,2,10,20,50) ahead forecast (y-axis are unit less HMI states). Prediction shown starting from second plot in top row left to right order.

Table 5.11: Time-Series RNN (LSTM) and CNN Models Forecast Accuracy Results

Rolling Window RMSE								
k-Lag	**n-Step**	**Vanilla**	**BiDirect.**	**Stacked**	**Encdr-Decdr.**	**1d CNN-Encdr.**	**2d Convl.LSTM**	**Roll.RMSEp**
1	1	1.35	**1.15**	3.17	3.11	3.09	2.23	**3.75**
2	1	**1.43**	2.29	2.04	3.84	2.93	2.56	**3.75**
5	1	2.92	3.71	2.99	2.06	**1.68**	2.09	**3.75**
10	1	**1.11**	1.17	1.23	2.07	1.97	1.97	**3.75**
1	2	**1.84**	1.93	2.18	3.32	3.88	3.87	**3.56**
2	2	**2.24**	3.24	2.73	3.61	4.23	3.99	**3.56**
5	2	2.26	2.67	**2.25**	4.63	4.30	3.27	**3.56**
10	2	**1.89**	2.04	1.99	4.02	3.97	3.35	**3.56**
1	10	**7.10**	7.15	8.50	11.05	11.02	11.34	**6.62**
2	10	7.85	**7.72**	7.82	10.72	11.03	10.80	**6.62**
5	10	**6.89**	7.17	7.92	9.25	11.06	9.41	**6.62**
10	10	10.28	**9.63**	11.12	13.22	12.57	15.44	**6.62**
1	20	**9.30**	9.54	10.16	11.63	12.54	12.44	**7.21**
2	20	10.37	**10.29**	10.48	12.68	11.81	12.41	**7.21**
5	20	**10.88**	11.02	11.93	12.31	13.37	12.37	**7.21**
10	20	12.68	12.68	13.14	12.68	12.45	**12.05**	**7.21**
1	50	**12.79**	12.82	12.90	13.67	16.94	13.82	**9.61**
2	50	13.04	**12.97**	13.29	14.21	16.79	14.83	**9.61**
5	50	**13.29**	13.40	13.85	14.46	15.16	15.07	**9.61**
10	50	13.96	14.13	14.25	13.48	13.52	**13.42**	**9.61**

(Fig. 5.29) which was used in 2d mode.

5.6.2 Training Effort - Epochs

The number of required training epochs vary significantly among the LSMT models to achieve approximately the same training and validation loss accuracy. Generally, there is an optimal number of training epochs for the given model that can either result in *under* or *over-fitting* the final model. A heuristic method was used to determine the optimal epoch as the first inflection point of the validation loss curve when it starts to cross and go above the training loss curve (Fig. 5.30).

The training effort translates to the amount of time required to train an optimal model, which depends on the required number of training epochs. Results showed *Vanilla* and *Bidirectional* LSTMs required relatively lower number of epochs while, *Convl.LSTM* and *CNN-Encoder* required longer training time (i.e. larger number of epochs) to yield approximately steady similar or lower final validation loss.

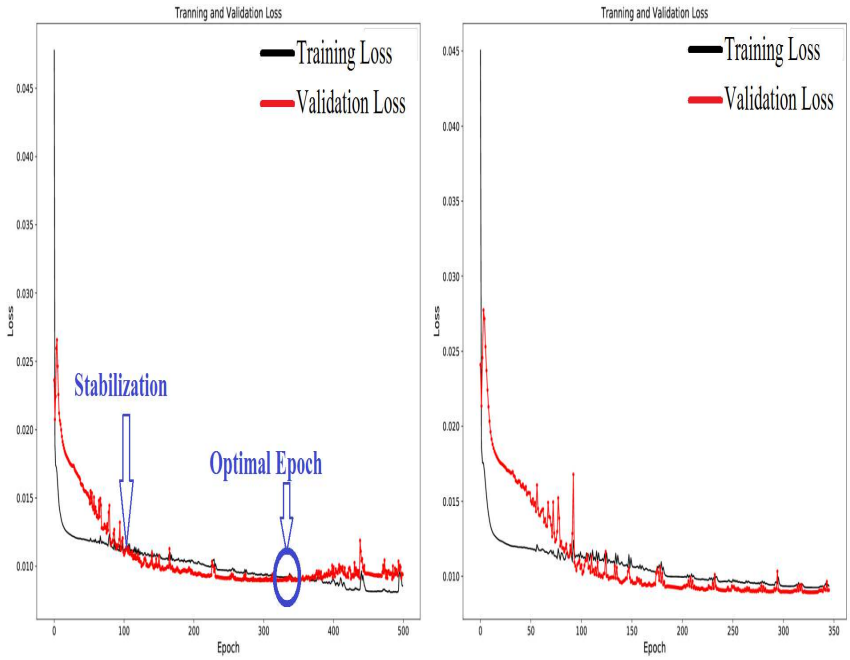


Figure 5.30: Optimal Training epoch is determined as the first inflection after stabilization of validation loss curve (*left*). Final model is then trained up to optimal epochs to avoid over-fitting (*right*).

5.7 RNN and CNN Model Summary of Results

Above results evaluates the feasibility of using existing out-of-the-box (un-optimized) RNN LSTM models (Sec. 4.4 and Fig. 4.10a) to predict operator actions embedded in HMI states (Fig. 3.2) as time-series data. Furthermore, the assumptions outlined in Sec. 3.3.2, allows HMI state patterns to be treated as weakly stationary time-series thus, making a case for employing deep learning RNN models as versatile and data-agnostic tools compared to legacy ARIMA models. Ultimately, the goal of the HMI state forecast models is to monitor and predict deviations in operator actions and achieve desired cross-validation.

Moreover, utilizing deep learning LSTM models with front-end multi-dimension convolutional feature extraction layers is favourable towards addressing scalability concerns [161] by overcoming the performance limiting curse of dimensionality [162] in learning more complex multi-variate HMI system patterns (Sec. 4.4 and Fig. 4.10b). Since, model training imposes significant computational burden rather than during model evaluation, access to more powerful graphics processor (GPU) based platforms are increasingly accessible as cost-effective cloud-based services for larger data sets.

The results indicate out-of-the-box LSTM models are accurate for shorter time *n-step*< 50 ahead predictions. However, in general, *Vanilla* LSTM performs better for all range of *n-step*

ahead forecasts with *2d Convl.LSTM* showing potentially slight advantage for doing longer time *n-step*$= 50$ ahead predictions.

The main take away from the above results are briefly summarized in Table 5.12.

Table 5.12: Time-Series RNN and CNN Model Result Summary

Test	**Result Highlights**	**Main Take Away**
Time-series forecast: K-Lag/N-ahead **out-of-sample** (**OuS**) dataset test for RNN (LSTM) and CNN Model	• Using standard LSTM models were evaluated using Out-of-sample training data set. Comparing below model performance for *K-lag = 10* past samples to generate *N-ahead = 50* future time steps: • Vanilla (basic) single layer LSTM model : RMSE < 13.9 • BiDirectional LSTM model: RMSE < 14.1 • Stacked LSTM model: RMSE < 14.2 • Encdr-Decdr LSTM model: RMSE < 13.48 • 1d CNN-Encder LSTM model: RMSE < 13.52 • 2d CNN-Ender LSTM model: RMSE < 13.42	• Both RNN and CNN based models generate acceptable results for N-ahead forecasts. • LSTM models are very useful to model longer sequences but show signs of over-fitting with smaller datasets. • Takes more memory resources to train as dataset size grows. • CNN based models show lower signs of over-fitting and train faster. • CNN models show larger capacity to scale up to use more multi-variate time-series features.

5.8 NLP Models Test Cases

Two models are evaluated in this thesis. First is a *seq2seq* LSTM based *encoder-decoder* model (referred to here as *LSTM E-D*) with a custom attention layer based on additive attention [116] mechanism as described previously in Section 4.5.3. The second model is the 1D CNN based custom model *encoder-decoder* design (referred to here as *Trident*) that uses dot product style attention mechanism as described previously in section 4.6.3.

The test cases include evaluating performance of above models under two training styles: (1) *Teacher forcing* (2) *Curriculum* (scheduled sample) Training. Teacher forcing for *LSTM E-D* model was previously described in Section 4.5.1 while teacher forcing, used for the *Trident* model was slightly modified, in that it incorporates *progressive* sample injection to overcome the non-sequential nature of the CNN networks as described previously in Section 4.6.1. Curriculum training was previously described in Section 4.7.

Test Cases:

1) *LSTM E-D* Teacher Forcing (LSTM-TF)

2) *LSTM E-D* Curriculum Learning (LSTM-CL)

3) *Trident* Teacher Forcing (Trident-TF)

4) *Trident* Curriculum Learning (Trident-CL)

Each above four test cases were swept over $K - Lag = [4, 8, 16, 32]$ and $n - Step = [1, 2, 4, 10, 20]$ parameters to train and evaluate the models using raw data set containing $4K$ samples with train vs. validation split of $60\% - 30\%$ for the resulting framed dataset (as described in above Section 5.3.2). The performance values obtained from the train data set is called the *in-sample* result test samples were selected from the original training set. Another data set (Section 5.3.1) containing $11K$ samples was generated for only re-evaluating the above trained models under above test cases independently - which is the out-of-sample(OuS) performance since all models have never seen these test samples during training.

All models have been custom built using Python 3.6.7 Keras 2.2.4 API libraries with open source TensorFlow 1.13.1 as its back end implementation.

5.9 NLP Model Results

The core modeling challenge is to accurately learn the supplied training dataset, which includes HMI process and corresponding Human response patterns. A rolling window Root-Mean-Square Error (RMSE) is used as a metric for comparing relative prediction accuracy, as done previously with regression time-series prediction. Persistence rolling window $RMSEp$ scores for the desired n-ahead predictions (Section 5.3.3) is used to compare the relative prediction performance of various models.

Lower $RMSErw$ (rolling window RMSE) values are demonstrated for longer $K - Lag$ values. Since, that provides the models longer history per sequence to base its prediction on and improve its forecast skill than avilable at lower $K - Lag$ values. Results in Table 5.13 show *LSTM E-D* generally yields lower $RMSErw$ (E.g. *LSTM E-D*: $RMSErw = 0$ while *Trident*: $RMSErw = 90.3$ for $K - Lag = 32$,$n - ahead = 20$ case) compared to *Trident* model under same test case. This is owing to non-sequential nature of convolutional networks to readily learn long sequences with extra CNN layers capture patterns over longer windows. However, current Trident model (Fig. 4.12) only utlizes two 1D CNN layers in this custom design. Nevertheless, current state-of-the-art research reveals CNNs can easily overcome the limitation by using more deeper layers, as demonstrated by the Transformer model [108] - which was discussed previously in Section 2.5.5.

Table 5.13: Seq2Sec Encoder-Decoder Model In-Sample Training Data Set Perf. Results

Test Cases/Metrics	k-Lag	n-Step	RMSEp (roll.Win)	Epochs	RMSE (rollWin)	Accuracy (hard)	Accuracy (Tolerance)	BLEU 1	BLEU 2	BLEU 3	BLEU 4	TRN (acc)	VAL(acc)
	4	1	3.75	158	1.153	79.76%	99.88%	1	0	0	0	69.10%	32.10%
LSTM E-D	4	20	7.21	164	3.529	72.25%	99.36%	1	0.98	0.92	0.77	85.90%	25.40%
Teacher Forcing	8	1	3.75	188	4.98	67.04%	98.90%	1	0	0	0	94.20%	18.20%
	8	20	7.21	135	1.925	93.81%	99.74%	1	1	1	0.99	99.40%	11.00%
	16	20	7.21	149	0.019	99.88%	100.00%	1	1	1	1	99.90%	10.60%
	32	20	7.21	148	**0**	**100.00%**	**100.00%**	1	1	1	1	**100.00%**	9.50%
LSTM E-D	4	1	3.75	81	1.262	79.24%	99.76%	1	0	0	0	70.30%	31.40%
Curriculum Learning	4	20	7.21	81	7.224	38.69%	97.53%	0.99	0.93	0.89	0.79	64.50%	9.00%
	8	1	3.75	81	0.314	95.48%	**100.00%**	1	0	0	0	92.90%	26.10%
	8	20	7.21	81	4.401	56.03%	98.99%	0.99	0.97	0.94	0.87	93.30%	7.30%
	16	20	7.21	81	2.242	67.44%	99.62%	0.99	0.98	0.96	0.92	99.90%	**6.70%**
	32	20	7.21	81	2.5	65.65%	99.58%	0.99	0.98	0.96	0.91	99.40%	7.10%
Trident	4	1	3.75	41	1.608	67.08%	99.76%	1	0	0	0	99.20%	**83.40%**
Teacher Forcing	4	20	7.21	421	7.165	46.72%	97.71%	1	0.96	0.94	0.89	95.20%	20.30%
	8	1	3.75	41	1.711	66.64%	99.52%	1	0	0	0	99.50%	80.80%
	8	20	7.21	421	3.559	79.24%	99.37%	1	0.98	0.97	0.94	95.20%	15.70%
	16	20	7.21	421	70.872	41.11%	84.50%	0.85	0.74	0.71	0.62	98.60%	16.00%
	32	20	7.21	421	**90.354**	**28.79%**	**75.43%**	0.75	0.62	0.59	0.49	99.10%	18.50%
Trident	4	1	3.75	81	2.238	54.80%	99.84%	1	0	0	0	85.70%	71.90%
Curriculum Learning	4	20	7.21	81	18.128	28.75%	95.18%	0.95	0.84	0.78	0.64	**49.70%**	15.20%
	8	1	3.75	81	1.902	56.92%	99.88%	1	0	0	0	86.90%	67.10%
	8	20	7.21	81	13.634	68.87%	98.36%	0.98	0.91	0.86	0.76	86.10%	12.50%
	16	20	7.21	81	16.424	74.36%	97.95%	0.97	0.92	0.88	0.8	91.10%	10.00%
	32	20	7.21	81	15.051	65.22%	98.14%	0.97	0.91	0.87	0.78	92.80%	10.40%

Table 5.14: Seq2Sec Encoder-Decoder Model Out-of-Sample Test Data Set Perf. Results

Test Cases/Metrics	k-Lag	n-Step	RMSEp (roll.Win)	Epochs	RMSE (rollWin)	Accuracy (hard)	Accuracy (Tolerance)	BLEU 1	BLEU 2	BLEU 3	BLEU 4	TRN (acc)	VAL(acc)
	16	4	4.32	0	**10.72**	16.05%	90.14%	0.99	0.75	0.51	0.19	0	0
	16	10	6.62	0	11.544	12.44%	89.27%	0.99	0.78	0.7	0.44	0	0
LSTM E-D	16	20	7.21	0	12.236	11.47%	90.38%	0.98	0.79	0.73	0.54	0	0
Teacher Forcing	32	4	4.32	0	11.101	14.59%	90.11%	0.99	0.75	0.51	0.18	0	0
	32	10	6.62	0	12.103	12.20%	89.18%	0.99	0.77	0.68	0.41	0	0
	32	20	7.21	0	12.127	11.40%	90.63%	0.98	0.79	0.73	0.54	0	0
	16	4	4.32	0	10.826	**17.47%**	90.66%	1	0.77	0.58	0.26	0	0
	16	10	6.62	0	11.422	14.36%	89.89%	0.98	0.83	0.79	0.64	0	0
LSTM E-D	16	20	7.21	0	12.044	11.57%	89.68%	0.98	0.82	0.78	0.63	0	0
Curriculum Learning	32	4	4.32	0	11.156	14.35%	89.78%	0.99	0.81	0.62	0.31	0	0
	32	10	6.62	0	11.359	13.20%	**91.27%**	0.99	0.81	0.76	0.56	0	0
	32	20	7.21	0	12.426	11.55%	90.11%	0.98	0.81	0.77	0.62	0	0
Trident	16	4	4.32	0	11.693	11.66%	88.68%	0.99	0.66	0.39	0.12	0	0
Teacher Forcing	16	10	6.62	0	17.119	11.25%	88.31%	0.96	0.68	0.58	0.31	0	0
	16	20	7.21	0	105.867	8.96%	63.64%	0.68	0.45	0.41	0.27	0	0
	32	4	4.32	0	16.101	10.37%	86.86%	0.97	0.7	0.46	0.18	0	0
	32	10	6.62	0	12.948	11.40%	89.60%	0.98	0.73	0.65	0.41	0	0
	32	20	7.21	0	**112.988**	**6.18%**	**55.60%**	0.63	0.43	0.4	0.29	0	0
	16	4	4.32	0	11.068	10.00%	89.91%	0.99	0.84	0.68	0.37	0	0
Trident	16	10	6.62	0	15.302	10.29%	84.66%	0.9	0.67	0.6	0.38	0	0
Curriculum Learning	16	20	7.21	0	34.656	9.91%	84.31%	0.9	0.67	0.62	0.45	0	0
	32	4	4.32	0	15.779	10.32%	84.65%	0.92	0.75	0.61	0.39	0	0
	32	10	6.62	0	14.224	10.64%	87.36%	0.95	0.72	0.66	0.47	0	0
	32	20	7.21	0	33.522	10.40%	87.50%	0.91	0.69	0.65	0.49	0	0

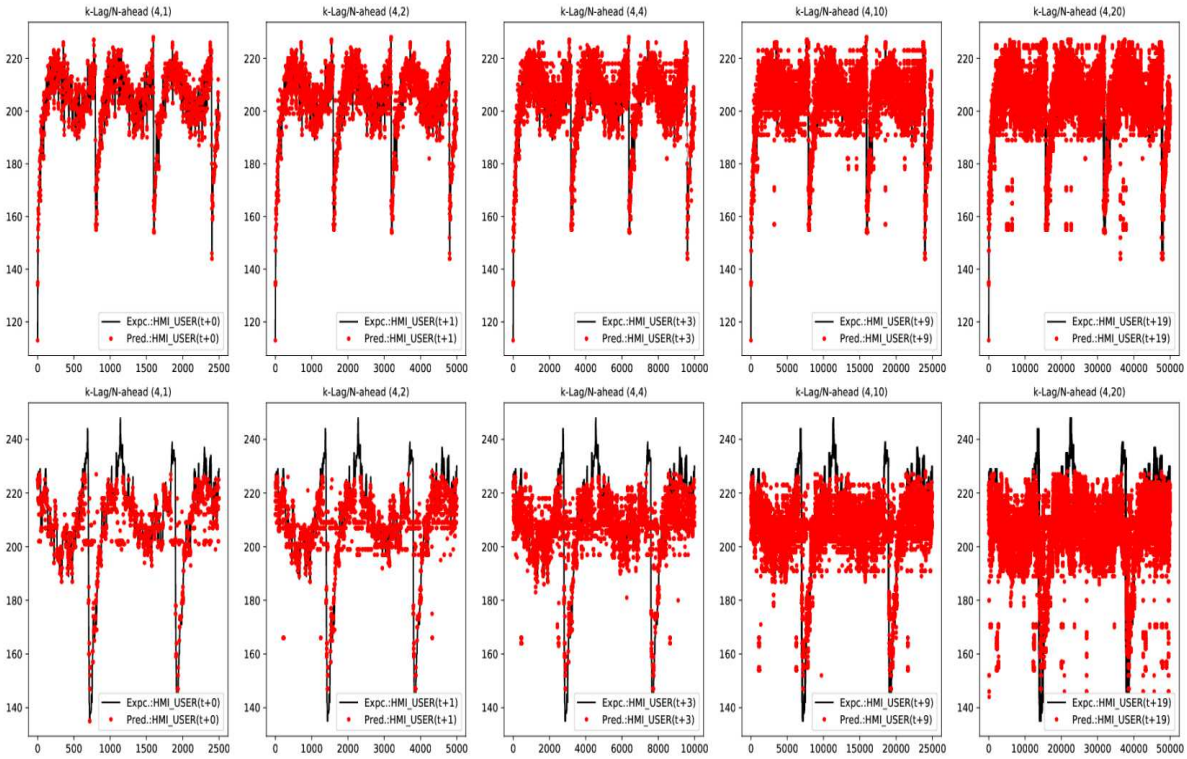


Figure 5.31: *LSTM E-D* Model - Expected(solid line) vs. Predicted(red dots) Operator response traces for Teacher Forcing test case from In-Sample (top row) & Out-of-Sample (bottom row) data set test under $K - Lag = (4)$ and $N - Step = (1, 2, 4, 10, 20)$ runs (in left to right order.)

Time-series Prediction Summary:

- All in-sample data test cases for *LSTM E-D* show lower rolling window RMSE ($RMSErw$) values than $RMSEp$ (Table 5.13). *Trident* shows lower rolling window RMSE ($RMSErw$) values than $RMSEp$ for shorter $N - Step < 20$ cases.
- *LSTM E-D* show lower $RMSErw$ values for longer $K - Lag$ values.
- *LSTM E-D* generally yields lower $RMSErw$ than *Trident* model.
- Curriculum learning benefits more *Trident* mode than it does to *LSTM E-D* to yield lower $RMSErw$ values for longer $N - Step >= 20$ cases.
- Out-of-sample $RMSErw$ scores are higher than $RMSEp$ (Table 5.14) for all models, but show consistent values for $K - Lag/N - Step$ values test cases.

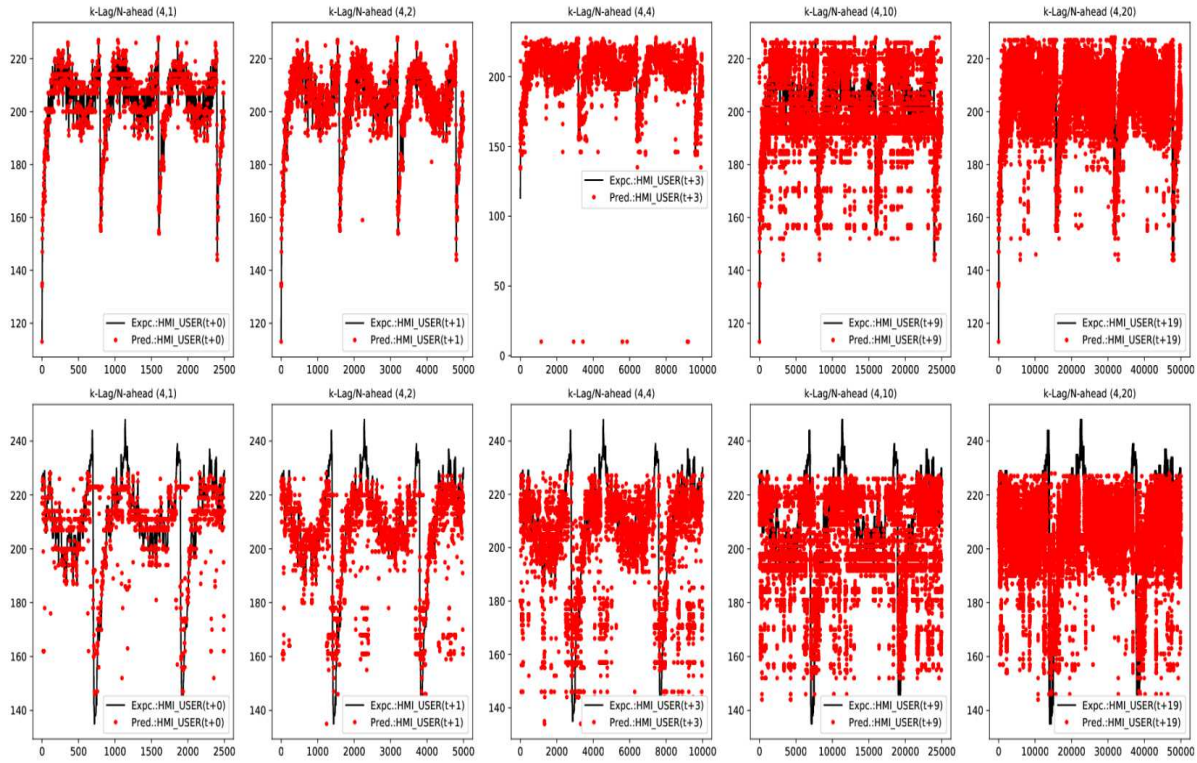


Figure 5.32: *Trident* Model - Expected(solid line) vs. Predicted(red dots) Operator response traces for Teacher Forcing test case from In-Sample (top row) & Out-of-Sample (bottom row) data sets test under $K - Lag = (4)$ and $N - Step = (1, 2, 4, 10, 20)$ runs (in left to right order.)

5.9.1 Prediction Accuracy

In addition, the proposed approach is to transform the time-series modeling as a NLP machine translation problem, which falls under the multi-classification domain of machine learning. Hence, label prediction accuracy measures and bilingual evaluation understudy (BLEU) scores have been utilized.

For the accuracy, *Accuracy(hard)* metric looks for a one-to-one match, each time step, between the class labels of expected and predicted output sequences. Secondly, *Accuracy(tolerance)* is a custom metric that is a relaxed accuracy measure that, too, makes one-to-one class label comparison. However, as long as class labels only do not vary more than a specified tolerance threshold ($tol. = 0.01\%$) they are accepted, otherwise rejected as not matching.

Results (Table 5.13), show the advantage of Curriculum learning vs. Teacher forcing for both *LSTM E-D* and $Trident$ models, as the models acquire prediction skills faster (requiring less epochs). Example, for *LSTM E-D* under teacher forcing $Accuracy(hard) = 100\%$ for $(32, 20)$ is reached at $Epoch = 148$, while $Accuracy = 65.65\%$ is reached at $Epoch = 81$. Similarly, for *Trident* model $Accuracy(hard) = 28.79\%$ for $(32, 20)$ at $Epoch = 421$, while

$Accuracy = 65.22\%$ is reached at at $Epoch = 81$.

Note, Epoch columns is all 0, since for doing out-of-sample testing, the model is not re-trained. The previously trained model using an in-sample dataset is tested with out-of-sample test dataset.

BLEU *1-gram* to *4-gram* overlap scores were calculated to ascertain each models translation performance while generating using previous *K-Lag* samples of HMI process to $N - ahead$ samples of HMI operator action sequence. BLEU algorithm requires two sets of reference and candidate sentence corpus. In this case, the reference sentence set contains the expected $N - ahead$ samples of HMI operator sequence, and the candidate set contains the actual predicted $N - ahead$ samples of HMI operator sentence.

In Table 5.13, BLEU scores have been averaged over all sequences in the evaluation data set for each listed test case. The perfect BLEU score is a value 1. We BLEU-1 as an individual 1-gram score to simply to account for the presence of all expected tokens in candidate sentences and that the expected sequence length matches the reference sequences. In-sample(Table 5.13) results indicate all models show a higher (> 0.95) BLEU-1 score for all test cases, indicating all models are skillful enough to generate the expected sequence length with all required tokens (irrespective order).

Similar to BLEU-1, BLEU-2 individual score looks at 2-gram (token) overlap (applicable for $N - Step > 1$ ahead sequences), i.e. for expected token pairs must appear in the same pair order as in reference sequences. For in-sample results (Table 5.13) BLEU-2 follows similar trend as demonstrated by BLEU-1 for all test cases with slightly lower values reported for *Trident* model for longer $N - Step > 10$) ahead sequence.

BLEU-3 and BLEU-4 scores are configured to yield a cumulative score, which is a weighted geometric mean of individual n-gram scores. For instance cumulative BLEU-3 uses $(0.33, 0.33, 0.33, 0)$ and BLEU-4 uses $(0.25, 0.25, 0.25, 0.25)$ as scale weight for 1-gram to 4-gram individual scores. This allows for meaningful scores to capture longer in order token sequences. The in-sample results in Table 5.13 show *LSTM E-D* models in general yield higher scores than *Trident* models. Even though curriculum learning results lower (approx.$< 3\%$) BLEU scores than Teacher forcing for both the models, but it is able to achieve this with a lower number of Epochs. Therefore, curriculum learning does offer learning at a faster rate, and further improvement is possible. Curriculum learning in particular does seem to help (i.e. yield higher accuracy) for *Trident* model more than it does *LSTM E-D* (approximately by factor of 1.06).

In brevity only a few expected vs. predicted output traces for in-sample and out-of-sample data sets for *LSTM E-D* and *Trident* models under Teacher Forcing case have been included in Fig. 5.31 and Fig. 5.32 respectively. These traces are representative of a general trend of a progressive drop in prediction accuracy as $N - Step$ ahead forecast range increases for a

given model trained at a fixed $K - Lag$ value ($K - Lag = 4$). However, as $K - Lag$ is increased, accuracy for longer forecast does improve as evident from results in Table 5.13 and Table 5.14.

Machine Translation Accuracy Summary:

- Accuracy(hard) measure for *LSTM E-D* is higher approximately (25%) compared to custom design $Trident$ model.
- Accuracy(hard) measure for *LSTM E-D* is lower (approx. $< 20\%$) for curriculum learning than for Teacher Forcing, but former is achieved via faster learning ($Epochs = 164$ vs $Epochs = 81$).
- BLEU scores generally validate Accuracy(hard) measure, i.e. higher Accuracy(hard) value will yield higher BLEU n-gram scores.
- Curriculum learning seems to help *Trident* model more (approximately by factor of 1.06) than it does to *LSTM E-D* model.

Lastly, looking at the training and validation accuracy values obtained during model training (Table 5.13) shows that - *Trident* model achieves higher validation accuracy than LSTM E-D model. This indicates that *Trident* is less prone to over-fitting and can further be improved by adding more convolution layers.

The main take away from the above results are briefly summarized in Table 5.15.

5.10 Trident Model- HITL Error Detection With Real NPP Scenarios

The objective here is to demonstrate the capability of a machine trained HMI state prediction model(s) to detect HITL errors before an accident event.

Two scenarios: LCV Swap (Sec. 5.12) and MBFB Duty Swap (Sec. 5.13) have been modeled based on actual Nuclear Power Plant (NPP) systems and control room panel indications which yield the required dataset covering both the normal and abnormal cases. Each scenario represents a specific operational goal that has been scripted to procedurally follow the same steps as a licensed control room operator (CRO) under normal case would. However, an instance of HITL error by way of missing a procedural step is intentionally introduced to simulate an abnormal case for the given operation scenario. The HITL error models an operator execution error or lapse as per the human cognitive error framework (Sec. 2.2.1), which is

Table 5.15: NLP RNN and CNN Model Result Summary.

Test	Result Highlights	Main Take Away
- State Translation/Forecast: K-Lag to N-ahead HMI states using **in-sample (InS)** and **out-of-sample (OuS)** dataset test for NLP RNN Seq2Seq. Encoder-Decoder Model.	- Using **NLP based RNN LSTM Encoder-Decoder** model with Attention mechanism Comparing below model performance for *K-lag = 32* past samples to generate *N-ahead = 20* future time steps: • **RNN In-Sample** *Teacher Forcing* yields 100% Accuracy (*hard*) • **RNN Out-of-Sample** *Teacher Forcing* yields 11.4% Accuracy (hard) • **RNN In-Sample** *curriculum training* yields 65% Accuracy (*hard*) • **RNN Out-of-Sample** *curriculum training* yields 11.5% Accuracy (hard)	• NLP RNN machine translation model for HMI state forecast is a class of multi-classifier models. • NLP RNN Seq2Seq models are superior for **in-sample** dataset HMI state forecast. • Accuracy improves with using more K-lag samples, as NLP RNN model can utilize stronger context for inference. (E.g. For LSTM-CL (8/20) vs. (32/20) yields 56% vs. 65% accuracy respectively.) • Approximately 80%, 60% accuracy drop seen for **out-of-sample** dataset forecast for both *Teacher forcing* and *Curriculum training* respectively. Suspected due to insufficient training samples and model over fitting – Seen as disparity between Training (TRN) and Validation (VAL) accuracy. • Lower scope for accuracy improvement unless model parameters are further tuned. • Lower scalability. As higher training resource cost associated with including more features.
- State Translation/Forecast: K-Lag to N-ahead HMI states using **in-sample (InS)** and **out-of-sample (OuS)** dataset test for NLP CNN (Trident) Seq2Seq. Encoder-Decoder Model	Using **CNN based Trident Encoder-Decoder** model with Attention mechanism Comparing below model performance for *K-lag = 32* past samples to generate *N-ahead = 20* future time steps: • **Trident In-Sample** *Teacher Forcing* yields 28.8% Accuracy (*hard*) • **Trident Out-of-Sample** *Teacher Forcing* yields 6.2% Accuracy (hard) • **Trident In-Sample** *curriculum training* yields 65.2% Accuracy (*hard*) • **Trident Out-of-Sample** *curriculum training* yields 10.4% Accuracy (hard)	• NLP CNN (Trident) machine translation model for HMI state forecast is a class of multi-classifier models. • Accuracy does not seem to be dependent on previous K-lag samples, but rather depends on number of layers of convolutions used. Trident utilizes two layers. • NLP Trident Seq2Seq models is superior for **out-of-sample** dataset HMI state forecast. • Consistent accuracy seen for **out-of-sample** dataset forecast for *Teacher forcing* and *Curriculum training* respectively. As model is learning well with lower over-fitting fitting – Seen as low disparity between Training (TRN) and Validation (VAL) accuracy. • Available room for accuracy improvement with longer training cycles. • Higher scalability. As parallelized structure incurs lower training resource cost associated with including more features.

indicative of reduced operator **situational awareness** in considering the current plant state before executing the next state.

The HMI state prediction model selected for these demonstration is the custom developed Trident (NLP Seq2Seq CNN based) model as described in Sec. 4.6 and evaluated in Sec. 5.9.

The methodology involved for training the models is a manually iterative process to select the best combination of supervised training parameter i.e., *K-lag*, *N-ahead*, number of training *Epochs* and training algorithm used (teacher forcing vs. Curriculum learning). This was required to select the best performing model which did not overfit and yielded acceptable validation accuracy for the given scenario (normal and abnormal) dataset(s). As such, the demonstration results included below are using the best performing trained Trident model

only.

5.11 Disclaimer for CANDU^{TM} Nuclear CRO Training Simulator Use

All NPP scenarios and accident(s) highlighting human errors as an initiating event, as described below in this thesis, even though plausible, are entirely hypothetical.

Therefore, it is not the goal of these scenarios to identify any particular past nuclear accident(s), any current operational or procedural deficiency nor identify any system specific detail for field or control room equipment(s) in the CANDU^{TM} OPG Darlington nuclear power generating station(s).

All information and data collected are in general, non-specific, and only intent is for research purposes.

Permission to use the Full Scope OPG CANDU^{TM} Darlington Nuclear Control Room Operator Training Simulator for obtaining the panel graphics and simulation data for the sole purpose of data analysis and research publication(s) has been obtained from **Ontario Power Generation** and **Ontario Tech University** training simulator responsible department/officials with concurrence from the research supervisor.

5.12 Scenario 1 - LCV Swap

Scenario 1 is about a power plant boiler or steam generator (BO2) LCVs (Level Control Valve) operation swap between an operating valve to a standby valve. This is a standard practice in a power plant to allow routine maintenance to occur on a valve by switching functionality to a standby valve during full-power operation. Human error risk is higher when returning the valve to service post-maintenance, as all isolations put in place may not have been removed completely and overlooked. This scenario captures one such instance of human error.

Scenario 1 description has obfuscated all surrounding support instrumentation detail from the real plant process (Fig. 5.33) , while only restricting to partial equipment identification codes to closely match the full scope CANDU^{TM} NPP training simulator control room operator panel graphics.

5.12.1 Background

In any nuclear power generating station, the boiler or steam generator is a pressurized vessel that converts the heat generated from the reactor to supersaturated steam which is used to

drive the turbine-generator set to produce electric power.

The level of water in the steam generator must be controlled in a very tight band due to several engineering and operational reasons that are outside the scope of this research. The main operational goal is to prevent the level of water in the steam generator from falling below or going above a setpoint.

In this scenario, we can consider a steam generator (Boiler-2 or BO2), which is supplied feedwater via three large Level Control Valves (LCVs): LCV201, LCV202, LCV203. Due to the large physical size of the valves, these have a large pneumatic actuators which makes them slow acting. The output of each LCV can be isolated for maintenance purposes by operating corresponding normally open motorized isolation valves: MV51, MV47, MV55 to closed, respectively.

LCVs (Level Control Valve) normal configuration is as indicated in Figure 5.33: LCV201 is 62% open while LCV202 and LCV203 are fully closed and remaining on standby with all isolation valves kept fully open (100%) normally. The $\%$-open position of LCV201 and LCV203 is automatically controlled by a boiler level control program (BLCP). The CRO via a three-position hand switch can manually switch the control of BLCP of LCV201 and LCV203 from either LCV201 only, BOTH and LCV203 only (Fig. 5.34).

The entire BO2 feedwater system is a closed-loop water supply. Therefore, the level of water in the BO2 is controlled indirectly by the input and output flow rate of the feed water into and going out of the BO2. Therefore, by modulating the LCVs positions, the BO2 water level is controlled.

Due to the non-linear physical phenomena: Nonminimum phase dynamics [163, 164] or more commonly referred to as *swell* and *shrink* effects exhibited by steam generators, as a result of the presence of high-pressure steam mixed with water (multi-phase dynamics) gives rise to reverse dynamics behavior of boiler level with respect to feed water and steam flow parameters. In general, any combination of three LCV position states that results in a BO2 input flow rate being restricted lower than $< 62\%$ ($308Kg/s$), results in BO2 water level to increase and vice-versa.

The reverse dynamics of this scenario can be seen in Figure 5.35 - as an increase of flow rate results in instantaneous dropping of the boiler level, and a restriction in flow rate instantaneously raises the boiler level.

5.12.2 Normal Case

The normal case for Scenario 1: Level Control Valve swap LCV201 $\rightarrow$ LCV203 and LCV203 $\rightarrow$ LCV201 entails following these sequence of steps in order as shown in Fig. 5.35: (Considering

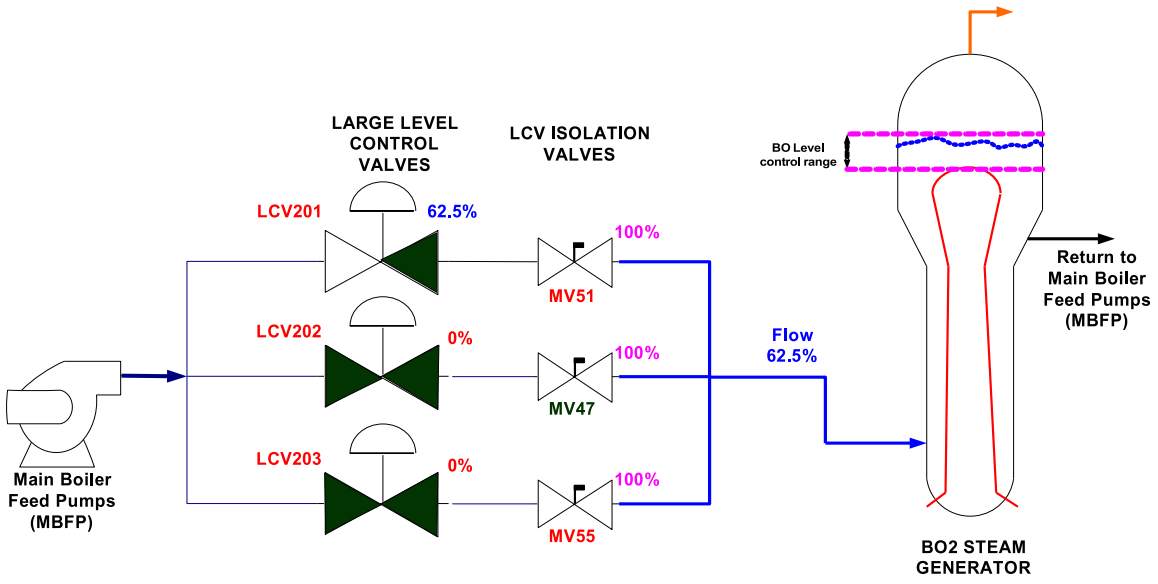


Figure 5.33: Scenario 1. Boiler2 (BO2) feed water level control valve normal configuration. LCV201 is normally at 62%, while LCV202, LCV203 are on standby at 0%. The feed water flow to BO2 is normally at 62%.

starting from LCV HS in LCV201 position selected)

1) **LCV HS Set to BOTH**: Set the LCV HS from **LCV201 to BOTH** position and wait until both Lamps are lit up. Both LCV201 and LCV203 will move to approximately 33% OPEN in the field. The feedwater flow increases instantaneously with boiler level decreasing in reverse, as LCVs transition: LCV201: $66\% \rightarrow 33\%$ and LCV203: $0\% \rightarrow 33\%$. Feedwater Flow and BO2 level stabilize after some settling period of the boiler level controller.

2) **HS Select to LCV203**: Set the LCV Hand switch from **BOTH to LCV203** position and wait until the corresponding Lamp is lit up. The feedwater flow decreases instantaneously with boiler level increasing in reverse as LCVs transition: LCV201: $33\% \rightarrow 0\%$ and LCV203: $33\% \rightarrow 66\%$. Feedwater Flow and BO2 level stabilize after some settling period of the boiler level controller.

3) **LCV HS Set to BOTH**: Return the LCV HS from **LCV203 to BOTH** position and wait until both Lamps are lit up. Both LCV201 and LCV203 will move to approximately 33% OPEN in the field. The feedwater flow increases instantaneously with boiler level decreasing in reverse, as LCVs transition: LCV201: $0\% \rightarrow 33\%$ and LCV203: $66\% \rightarrow 33\%$. Feedwater Flow and BO2 level stabilize after some settling period of the boiler level controller.

4) **HS Select to LCV201**: Return the LCV Hand switch from **BOTH to LCV201** position

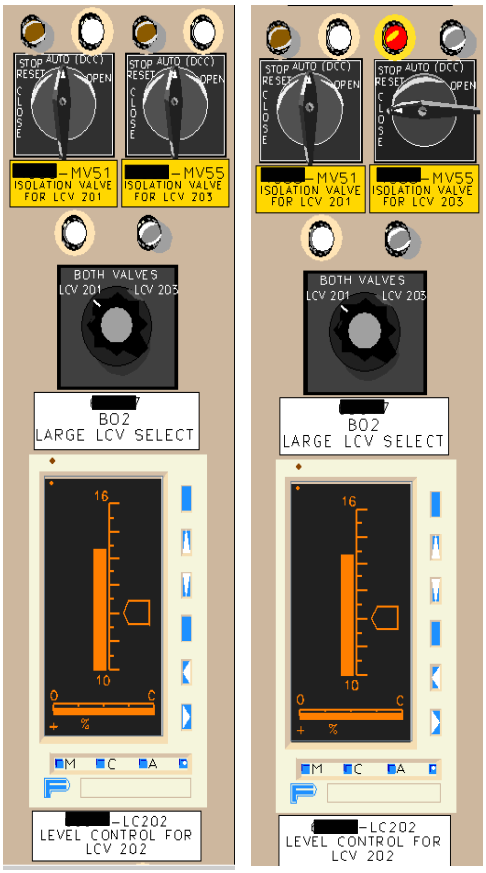


Figure 5.34: The LCV control panel obtained from training simulator, Panel indication, and switch in normal configurations; (left) shows LCV SELECT hand switch (LCV HS) with white indicator lights above it (when both lit, both LCV201, LCV203 are OPEN in field). The Hand switches for LCV isolation valves MV51 and M55 are for LCV201, LCV203 respectively. The Isolation valve status is indicated by Red (CLOSED), White (OPEN) lamps, when both lit valve is in transit;
(right) shows the panel when LCV203 is isolated for maintenance as MV55 has been closed by the operator.

and wait until the corresponding Lamp is lit up. The feedwater flow decreases instantaneously with boiler level increasing in reverse as LCVs transition: LCV201: $33\% \rightarrow 66\%$ and LCV203: $33\% \rightarrow 0\%$. Feedwater Flow and BO2 level stabilize after some settling period of the boiler level controller.

Note: During the above normal LCV swap sequence, the operator may isolate either LCV201 or LCV203 once it has been set to a fully closed (0%) position by setting the hand switch for isolation MV51 or MV55 to CLOSE position respectively. Once isolation MVs are fully closed AMBER Lamp will be lit on the panel (Fig. 5.34). Once the maintenance is completed, the isolation valve must be fully OPENED up.

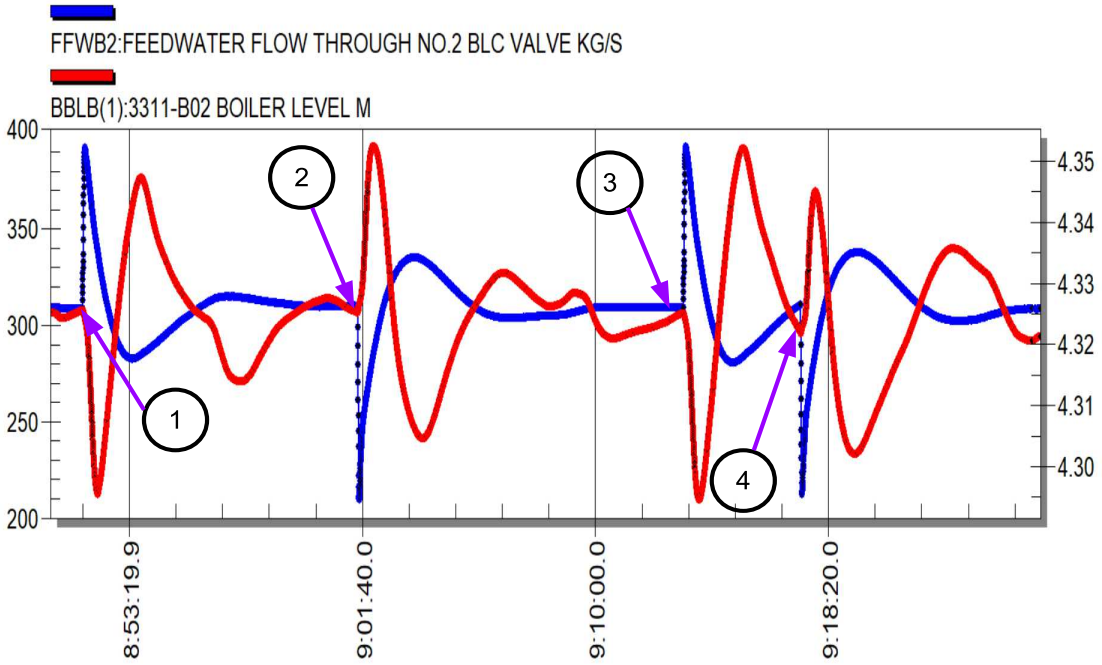


Figure 5.35: Feedwater Flow and Boiler Level under Normal LCV swap LCV 201 → LCV203 sequence. The normal sequence of switching between LCV201 and LCV203 for maintenance purpose is described above in List 5.12.2, is LCV201 → BOTH → LCV203; LCV203 → BOTH → LCV201.

5.12.3 Abnormal Case

The abnormal case for Scenario 1 LCV swap entails leaving an isolation valve CLOSED for the LCV that has been selected back to service by the operator as shown in Fiq. 5.36 after its maintenance is completed, which is indicated by control panel configurations shown in Fig. 5.38.

Considering, LCV203 is under maintenance (CLOSED), with MV55 isolation valve also CLOSED, and LCV HS is at LCV201 (62%), which currently provides the feedwater flow. Under human error, LCV203 will be put into service.

1) **LCV HS at LCV201**: Initially, LCV HS is at LCV201 position, which is 62% OPEN. LCV203 is isolated by closing MV55, Wait until MV55 AMBER Lamp is lit up. No change to feed water occurs, as it's flowing through LCV201.

2) **LCV HS to BOTH**: Set the LCV HS from **LCV201 to BOTH** position and wait until both Lamps are lit up. Only LCV203 will move to approximately 62% OPEN in the field,

and LCV201 will maintain at 62% OPEN. No change to feed water occurs, as it's flowing through LCV201.

3) **LCV HS to LCV203**: Set the LCV Hand switch from **BOTH to LCV203** position and wait until the only corresponding Lamp (LCV203) is lit up. The feedwater flow decreases instantaneously with boiler level increasing sharply in reverse as LCVs transition: LCV201: $62\% \rightarrow 0\%$ and LCV203: $33\% \rightarrow 100\%$ but there is no feedwater flow to BO2.

4) **LCV HS ATUO to BOTH**: Under automatic control of the BLC computer to prevent unit trip, the LCV select HS position at LCV203 is overridden and internally is automatically set to BOTH position. The feedwater flow increases sharply with boiler level decreasing sharply in reverse as LCVs transition: LCV201: $0\% \rightarrow 98\%$ and LCV203: $100\% \rightarrow 98\%$.

5) **LCV HS ATUO**: Under automatic control of the BLC computer continues to control the position of the LCV201 and LCV203 until the BO2 feed water and level have been stabilized in normal control range.

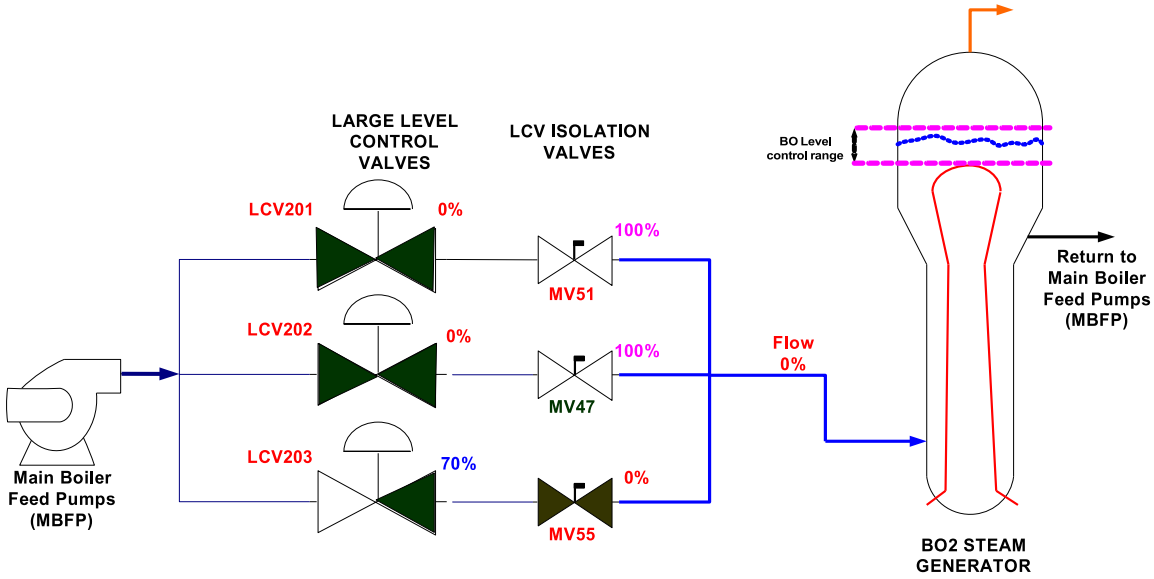


Figure 5.36: Scenario 1. Boiler2 (BO2) feed water level control valve abnormal configuration. LCV203 post maintenance returned to service 70% OPEN, while LCV202, LCV203 are on standby at 0% but, MV55 has been left CLOSED in error. The feed water flow to BO2 is now reduced to 0%, which may cause reactor to trip on Boiler level high event.

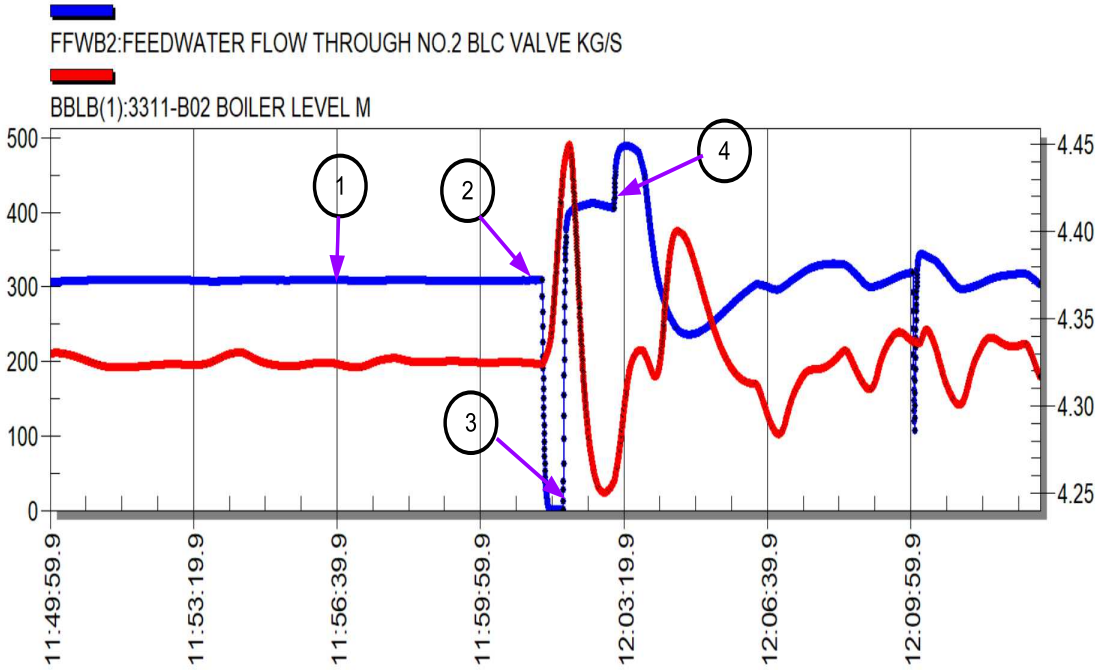


Figure 5.37: Feedwater Flow and Boiler Level under Abnormal LCV swap LCV201 → LCV203 sequence. The abnormal sequence of swapping from LCV201 to LCV203, after the maintenance on LCV203 has been completed: LCV201 → BOTH → LCV203 but MV55 was left CLOSED after maintenance in error. The entire sequence is describe in above List 5.12.3.

5.12.4 Simulator Data as Model Training Input

In order to transform the simulator data so it could be captured in the two features variables, which currently the Trident model expects as raw training data. The following data packing and transformation was done:

- Boiler Flow floating point value was scaled by $\times 10$ and converted to a 8 bit value (0 to 250) (the remaining tokens are reserved for other START and STOP markers). This transformed value can be represented by the PROCESS feature used by the model.

- The LCV HS positions $(LCV201, BOTH, LCV203)$.
  Lamp indication statues of: MV51 $(MV51CLOSE, MV51OPEN)$
  and MV55 $(MV55CLOSE, MV55OPEN)$ These 7 digital flags were packed in a 8 bit vector in this order
  (LCV201, BOTH, LCV203,0,MV51CLOSE, MV51OPEN, MV55CLOSE, MV55OPEN) for HMI_USER feature that is used by the model. Therefore, HMI_USER feature effectively

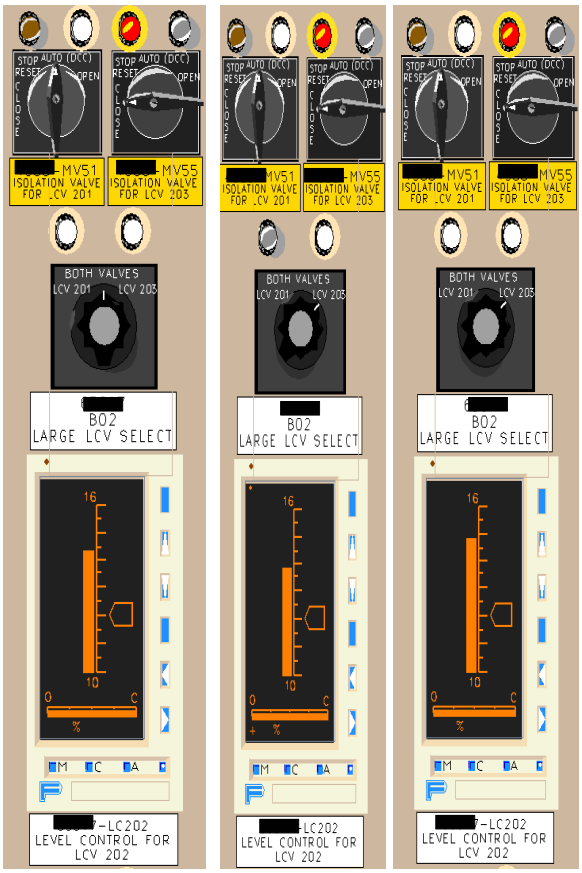


Figure 5.38: The LCV control panel in abnormal configuration. On Left, shows LCV SELECT hand switch (LCV HS) to BOTH while LCV203 isolation valve MV55 is CLOSED; In Middle, LCV HS is set to LCV203, while MV55 is till CLOSED; On Right, Boiler Level Controller takes control of LCV HS overrides LCV203 selection and internally sets it to BOTH, trying to OPEN LCV201 to allow feed water to start flowing again.

captures all HMI panel events, both process indications via Lamp status and operator actions via hand switch positions.

The following state table (Fig. 5.39) is based on various bit combination values of above HMI_USER feature variable. This is created for easier interpretation of the LCV control panel indication and hand switch states, which have been transformed into training dataset values and the trained model output.

5.12.5 Model Demonstration Result

The Trident model was selected and trained using curriculum learning (Sec. 4.7) using a raw dataset containing approximately $43K$ samples and the test dataset containing approximately $30K$ samples. The raw dataset contains 5 instances of normal LCV swaps and 1 instance of

BIT-PATTERN	HMI_STATE VALUE	OPERATOR ACTION - STATE LABEL	CONDITION
100 0 0101	133	SEL LCV201 – LCV HS: LCV201 SELECTED, MV51 OPN, MV55 OPN	NORMAL
010 0 0101	69	SEL BOTH – LCV HS: BOTH SELECTED, MV51 OPN, MV55 OPN	NORMAL
001 0 0101	37	SEL LCV203 – LCV HS: LCV203 SELECTED, MV51 OPN, MV55 OPN	NORMAL
100 0 0110	134	ISOL LCV203 – LCV HS: LCV201 SELECTED, MV51 OPN, MV55 CLOSE	NORMAL
010 0 1001	73	ISOL LCV201 – LCV HS: BOTH SELECTED, MV51 CLOSE, MV55 OPN	ABNORMAL
001 0 1001	41	ISOL LCV201 – LCV HS: LCV203 SELECTED, MV51 CLOSE, MV55 OPN	NORMAL
010 0 0110	70	ISOL LCV203 – LCV HS: BOTH SELECTED, MV51 OPN, MV55 CLOSE	ABNORMAL
001 0 0110	38	ISOL LCV203 – LCV HS: LCV203 SELECTED, MV51 OPN, MV55 CLOSE	BAD
001 0 1101	45	ISOL LCV201 – LCV HS: LCV203 SELECTED, MV51 MOVING, MV55 OPN	NORMAL
100 0 0111	135	ISOL LCV203 – LCV HS: LCV201 SELECTED, MV51 OPN, MV55 MOVING	NORMAL
010 1 0110	86	ISOL LCV203 – LCV HS: BOTH SELECTED, MV51 OPN, MV55 CLOSE – (ALRM BIT SET)	ABNORMAL
001 1 0110	54	ISOL LCV203 – LCV HS: LCV203 SELECTED, MV51 OPN, MV55 CLOSE – (ALRM BIT SET)	BAD

Figure 5.39: LCV control room panel state reference table. This may be used to interpret the HMI state and trained model output.

abnormal LCV swap sequence of Scenario 1. The second training dataset only contains a normal LCV swap sequence of Scenario 1.

Training and Testing Methodology

There are two variation of supervised training and validation data sets with: $K - LAG/N - Ahead$: $(32, 10)$ and $(32, 20)$ window samples that were generated as shown in Figure 5.11 (Sec. 5.3).

Training with each above supervised data sets (containing both normal/abnormal sequences) yielded two separate trained Trident models for Scenario 1 each with validation accuracy of 80% and 90% for the $K - LAG/N - Ahead$: $(32, 10)$ and $(32, 20)$ dataset cases respectively (these will be referred to as *Tr_K32_N10_Sc1* and *Tr_K32_N20_Sc1* models). As described previously in section 4.7, the Trident NLP model (Fig. 4.12) is trained using *curriculum* training which includes initially feeding it both K past samples of both PROCESS and HMI_USER features as encoder/decoder input and corresponding N ahead samples as decoder target reference output sequence (which it must learn).

A third model was also trained on a dataset containing only normal sequences. This dataset was $K-LAG/N-Ahead$: $(32, 10)$ and the trained model referred to as *Tr_K32_N10_Sc1Nor* resulted in validation accuracy of 98% (Fig. 5.40).

Testing the trained models is done with a completely new out-of-sample raw dataset containing 3 instances of normal and 1 instance of abnormal LCV transfer sequence. During testing Trident NLP model (Fig. 4.12) is made to forecast in *teacher forcing* mode, in which: both K past samples of both PROCESS and HMI_USER features as decoder input is provided, however for decoder input a *zero* valued input vector initialized with $< Start >$ token only is provided. Subsequently, as the model predicts more tokens for a given input sequence of

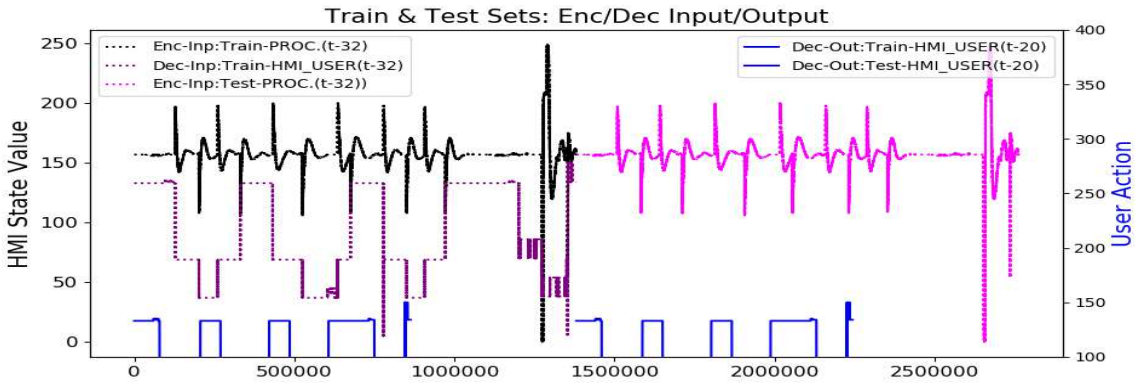


Figure 5.40: Example of combined normal and abnormal raw dataset supervised training and validation dataset for Scenario 2: $K - LAG/N - Ahead$: $(32, 20)$. Shows Decoder/Encoder PROCESS and HMI_USER input sequences and Decoder target train output sequences. (Validation dataset was set to 90% of training dataset).

$K - LAG$ samples, its predicted tokens are added to the decoder input vector until all the $N - Ahead$ tokens have been produced.

Model Output Analysis

The Trident model is designed and trained to forecast tokens at $N - Ahead$ time steps for the HMI_USER feature, which is a composite vector containing both control panel indications and operator switch status (as per Sec. 5.12.4).

Model output shows the predicted output closely tracks the expected HMI_USER feature state sequence. However, model forecast samples are generated prior (seen at lower sample indexes) to the decoder input $K - LAG$ HMI_USER sequence (Fig. 5.41), that are presented as inputs to the decoder during inference mode.

That is, the model is able to output HMI State forecasts, as designed, only based on past $K - LAG$ input sequences of both process variable (PROCESS feature): Boiler feedwater flow, and HMI LCV panel and hand switch states (HMI_USER feature).

More interestingly, above designed behavior of the model translates into actual time steps the model is able to forecast ahead in time. This is effectively at least 10 time steps prior to actual HMI states changing (Fig. 5.42) as recorded in the raw dataset (direct output recorded from the simulator). The rolling window average of predicted forecast (Fig. 5.42) is an effective visualization tool that averages the forecast samples belonging in the same temporal slice (as per Fig. 5.14 in Sec. 5.3.3)

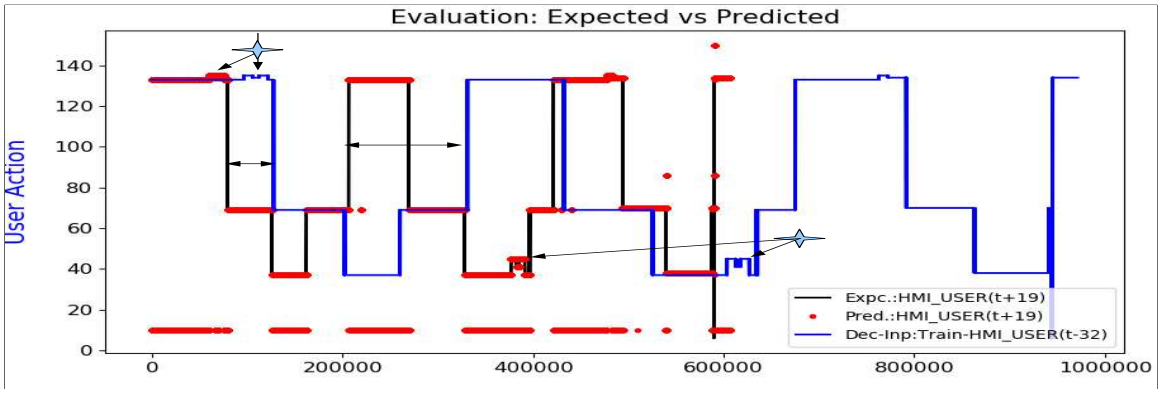


Figure 5.41: Example of Model Predicted vs. Expected Output from *Tr_K32_N20_Sc1*. Model predicted output (red dots) closely tracks the expected HMI_USER feature (black) state sequence. The $K - LAG$ HMI_USER (blue) HMI events appear as inputs to the model after the model has forecast those same events (marked by stars) previously. This because the model is trained to do $N - Ahead$ time step forecast, as long as the input sequence is similar to what it was trained on.

Attempt to detect HITL Error

Following provides interpretation of the trained model forecast output that may aid in the potential detection of target HITL error precursors for LCV Swap sequence scenario.

Described above in Sec. 5.12.3, the HITL error is introduced when LCV203 isolation valve MV55 is not OPENED up.

Normal expected sequence as seen in terms of HMI_USER feature state transitions:

LCV203 Maintenance isolation:
MV55 TO CLOSE $\rightarrow$ MV55 MOVING $\rightarrow$ MVV55 CLOSED

LCV203 Remove isolation:
MV55 TO OPEN $\rightarrow$ MV55 MOVING $\rightarrow$ MVV55 OPENED $\rightarrow$ LCV HS TO BOTH

Corresponding Normal HMI_USER State transition:
$133 \rightarrow 135 \rightarrow 134 \rightarrow 135 \rightarrow 133 \rightarrow 70$

Corresponding Abnormal HITL Error Precursor sequence is:
$133 \rightarrow 135 \rightarrow 134 \rightarrow 70$

The main mode of HITL error detection proposed is by looking for instances of sustained deviation between the forecast output and the HMI indication sequence. Such as demonstrated below in Figure 5.43 and Figure 5.44:

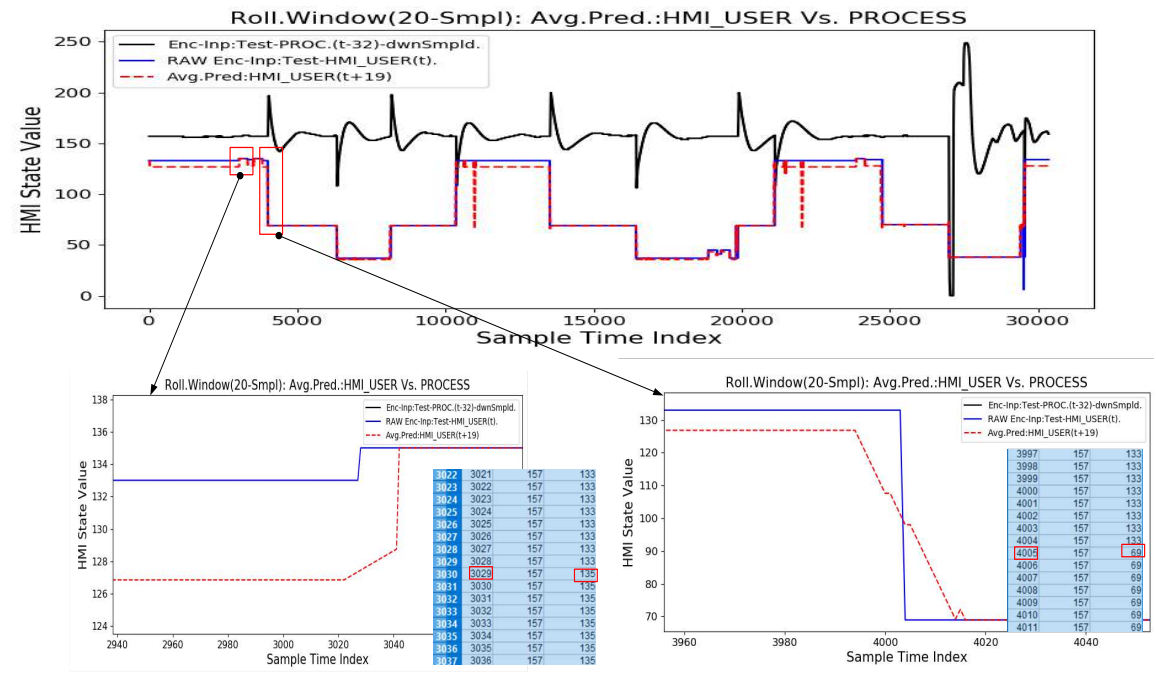


Figure 5.42: Rolling window average of Predicted vs. Raw Expected Model Output from *Tr_K32_N20_Sc1*. Rolling window average Model predicted output (red dashes) closely tracks the expected raw HMI_USER feature (blue) state sequence. The model forecast sequence is approximately atleast 10 time index prior;
The zoomed-in boxes, show two particular HMI state transition events and compares the time indexes of the predicted and raw sequences;
As per forecast, the first predicted rise (lower left plot) occurs at time index 3020, while as per the raw dataset, this transition occurs at index 3029.
Similarly, for second transition (lower right plot) forecast event occurs at index 3990 while, raw data transition occurs at 4004.

In, Figure 5.43 *Tr_K32_N10_Sc1* model which is trained on combined normal and abnormal sequence is used. During training, the abnormal sequence (decoder/encoder) input states were manually modified to include a marker state, while the rest of the normal sequence states were left unaltered. This was done to intentionally allow the model to learn to differentiate when normal and abnormal sequences are encountered. This, however, did not produce the intended behavior in the forecast, mainly owing to insufficient instances of abnormal sequences in the dataset. Nevertheless, the model did learn to emphasize the normal sequence more than the abnormal sequence owing to the presence of more former sequences in the training dataset. The outcome can be seen, when the model sees the start of the HITL error precursor - a sequence that partially matches the normal one in its beginning, its output forecast tries to match its learned behavior (lower right plot in Fig. 5.43) and infers how the sequence should have continued. This causes a deviation in the actual HMI sequence vs.

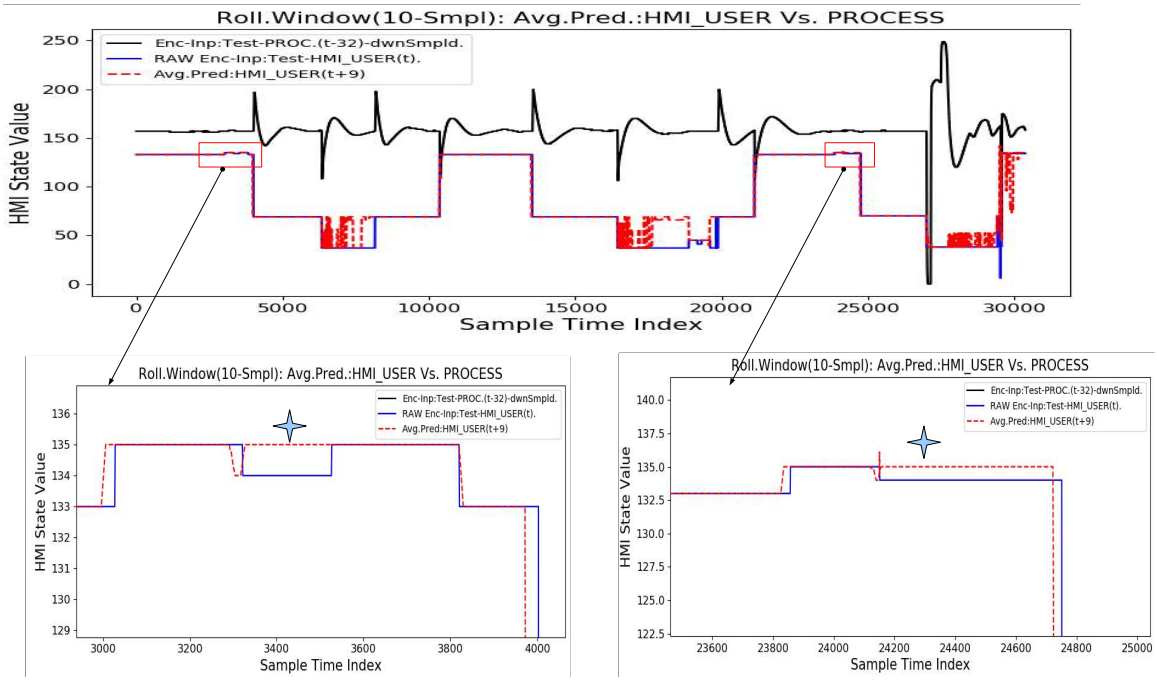


Figure 5.43: HITL Error detection with Rolling window average of forecast output vs. Raw actual HMI states using *Tr_K32_N10_Sc1* model (trained on combined normal and abnormal sequence dataset). The zoomed-in boxes, show two particular HMI state transition of normal and abnormal events;

As per the forecast, the normal sequence of MV55 (OPEN, MOVING, CLOSE, MOVING, OPEN) (lower left plot) (blue) is moderately tracked by the model output forecast (red dashed) showing deviation between the two for a shorter span (shown by star);

Compared, to the case of HITL error precursor initiates when MV55 (OPEN, MOVING, CLOSE) sequence occurs, the deviation between actual model forecast (red dashed) and actual HMI state (blue) persists for a longer duration span (shown by star).

forecast sequence, indicative of HITL error onset.

In, Figure 5.44 *Tr_K32_N10_Sc1Nor* model which is trained only on normal sequences is used. This model seems to be more closely tracking the actual HMI sequence. It automatically behaves very differently once a partial sequence (HITL error precursor) is encountered (lower right plot in Fig. 5.44), where the deviation between actual and forecast sequence persists for a longer time span.

In summary, above results demonstrate using forecast models that are trained on combined abnormal and normal sequence dataset (e.g. *Tr_K32_N10_Sc1* Fig. 5.43) or only trained on normal sequence dataset(e.g. *Tr_K32_N10_Sc1Nor* Fig. 5.44) can potentially be used to detect HITL error precursors in scenario 1 for LCV swap. The caveat being the model, must be trained on several normal and abnormal sequences to be able to accurately forecast the HMI

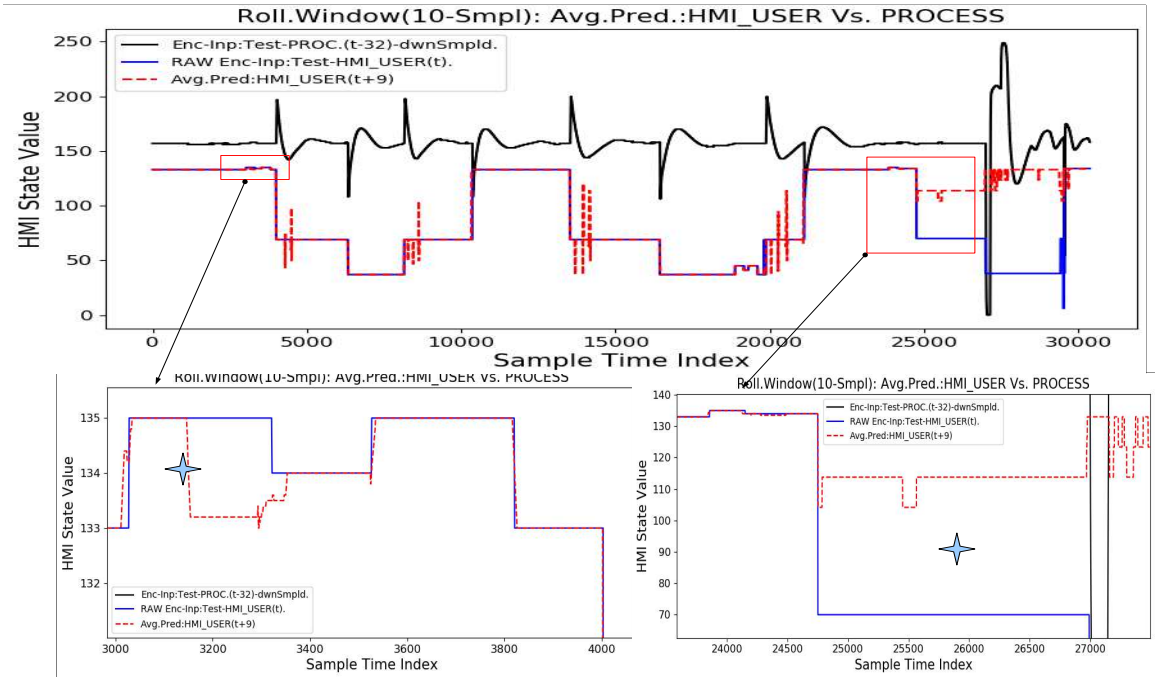


Figure 5.44: HITL Error detection with Rolling window average of Forecast vs. Raw actual HMI sequence using *Tr_K32_N10_Sc1Nor* model (trained on normal sequence dataset only). The zoomed-in boxes, show two particular HMI state transition of normal and abnormal events;

As per the forecast, the normal sequence of MV55 (OPEN, MOVING, CLOSE, MOVING, OPEN) (lower left plot) (blue) is moderately tracked by the model output forecast (red dashed) showing deviation between the two for a shorter span (shown by star);

Compared, to the case of HITL error precursor initiates when MV55 (OPEN, MOVING, CLOSE) sequence occurs, the deviation between actual model forecast (red dashed) and actual HMI state (blue) persists for a longer duration span (shown by star).

states for future time steps (in general to reduce over-fitting and obtain a more generalized model).

The tradeoff between the two types, normal only vs. combined abnormal and normal dataset trained, of models is the former is more sensitive to detecting any sequence change whereas the latter model is more generalized but requires many careful filters to detect pattern deviation anomalies to detect HITL error precursors.

5.13 Scenario 2 - MBFP Duty Swap

Scenario 2 is about the main boiler feed pump (MBFP) duty swap from an operating pump P1 to standby P4 (P1 $\rightarrow$ P4). This is a standard practice in a power plant to allow routine maintenance to occur on a pump equipment by switching functionality to a standby pump during full power unit operation (does not require unit outage). Human error risk is higher when returning pump P1 back to service post-maintenance, as all isolations put in place may not have been removed completely and overlooked. This scenario captures one such instance of human error.

Below, Scenario 2 description has obfuscated all surrounding support instrumentation detail from the real plant process (Fig. 5.33) and, only restricting to partial equipment identification codes to closely match the full scope CANDU^{TM} NPP training simulator control room operator panel graphics.

5.13.1 Background

As stated previously (Sec. 5.12.1), the level of water in the steam generator must be controlled in a very tight band due to several engineering and operational reasons that are outside the scope of this research. The main operational goal is to prevent the level of water in the steam generator from falling below or going above a setpoint.

In this scenario, we consider a steam generator (BO2), which is fed feed water by four large main boiler feedwater pumps (MBFP) P1, P2, P3, P4. P4 usually is on standby (Auto-Start) (Fig. 5.48) while all other three pumps operate at 80% load to keep the feedwater flow at approximately 100%. Each MBFP is associated with its discharge isolation motorized valve (MV).

Due to the large physical size of the MBFP discharge valves for each pump, these have a large pneumatic actuators which makes them slow acting. The output of each MBFP can be isolated for maintenance purposes by operating corresponding normally open motorized discharge valves: MV15, MV16, MV17, MV18 to a closed state commanded via control room panel hand switches.

Each MBFP output load is throttled automatically to maintain the desired flow rate to the boiler.

The operator via the control room panel (Fig. 5.46) hand switch commands the Pumps to turn ON/OFF and discharge valves to OPEN/CLOSE as required for doing the MBFP duty swap for equipment maintenance purposes.

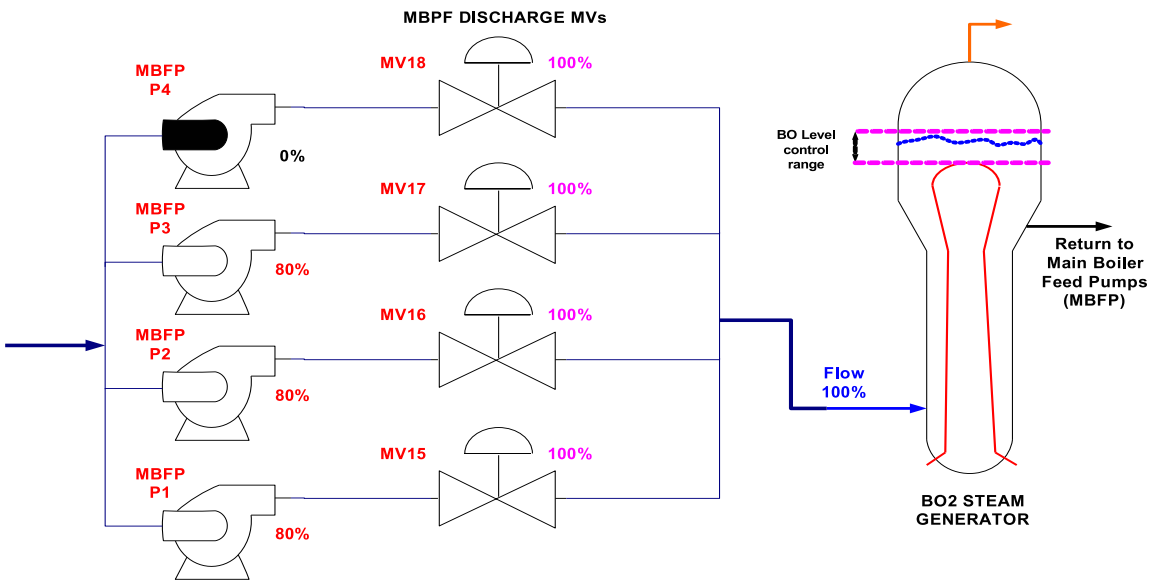


Figure 5.45: Scenario 2. Main Boiler Feed Pumps (MBFP) normal configuration. MBFP P4 is normally on standby 0%, while BMFP P1, P2, P3 are running at 80% load. All MBFP discharge valves MV15 to MV18 are left OPEN. The feed water flow to BO2 is normally at 100%.

5.13.2 Normal Case

The normal case for Scenario 2: Main Boiler Feed Pump duty swap from P1 $\rightarrow$ P4 entails following these sequence of steps in order as shown in Fig. 5.47a:

Considering the initial normal starting condition: P4 HS on standby and P4 is not running. P1 to P3 are all running and MBFP discharge MVs MV15 to MV18 are all OPEN with WHITE indicator LAMPs ON.

1) **P4 HS to ON, MV18 HS to CLOSE**: P4 hand switch is set to ON by the operator, P4 has started successfully in the field (WHITE Lamp is ON). Discharge MV18 hand switch is set to CLOSE by the operator, and it has been slowly closed fully (RED Lamp is ON) (Fig. 5.46(2)).

2) **MV18 HS to OPEN/STOP**: Gradually, P4 discharge MV18 is opened up, but switching between OPEN/STOP hand switch positions as per procedure is performed (Fig. 5.47b). The boiler feed flow increases at a fast rate (Fig. 5.47a) as P4 is now adding to the feedwater flow with all other MBFP P1,P2, P3 running.

3) **MV18 HS to OPEN**: P4 discharge MV18 is fully OPEN in the field (WHITE Lamp) is ON. P4 is automatically throttled back to adjust the feed flow rate. Therefore the flow drops to normal range (Fig. 5.47a)

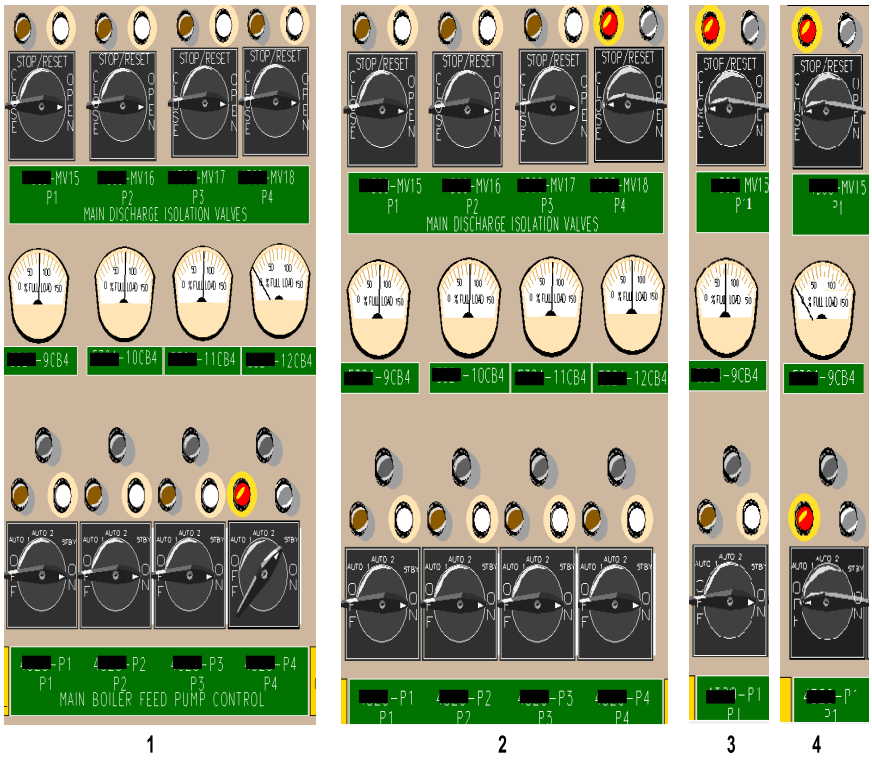


Figure 5.46: The MBFP control panel obtained from training simulator, Panel indication and switch in normal configurations.
(1) Shows P4 hand switch on standby position while the RED Lamp showing it is not running in the field. In addition, its load current meter (12CB4) is reading zero. On top MV15 to MV18 MBFP discharge valves, hand switch is OPEN and WHITE lamps are confirming the OPEN state of the valves in the field. P1, P2 and P3 are running at approximately 80%, indicated by their individual WHITE Lamps ON and load current meters;
(2) The only difference is P4 hand switch is changed to ON position, WHITE Lamp indicator shows P4 is now running in the field (further confirmed from meter reading 12CB4) at 75%;
(3) Shows P1 is still running in the field (meter reading 9CB4) at 75%. P1 discharge MV15 is CLOSED in field confirmed by the RED Lamp in ON state;
(4) Shows P1 HS is set to OFF, P1 is stopped running in the field (meter reading 9CB4) at 0% and RED Lamp is ON. P1 discharge MV15 is CLOSED in the field, RED Lamp is ON. (Note: RED or AMBER color Lamps are used interchangeably in this thesis to mean the same state.)

4) **MV15 HS to CLOSE/STOP**: MBFP P1 discharge MV15 is slowly closed by the operator by switching between CLOSE/STOP hand switch positions as per procedure. The boiler feed flow slow starts to increase as MBFP P2,P3,P4 are automatically throttled up to compensate for the loss of flow from MBFP P1. (Fig. 5.47a)

5) **MV15 HS to CLOSE**: MBFP P1 discharge MV15 is fully closed in field. MV15 RED Lamp is ON (Fig. 5.46). Feed water flow drops sharply (Fig. 5.47a) as P1 flow discharge is fully isolated.

6) **P1 HS to OFF, MV15 HS to CLOSE**: MBFP P1 HS is set to OFF by operator, P1 stops running in field (RED Lamp is ON) 5.46). Feed water flow starts to increase to normal level as other MBFP pumps P2, P3, P4 are throttled up automatically (Fig. 5.47a).

7) **P1 HS to OFF, P4 to ON**: MBFP duty swap from P1 to P4 is complete. Feedwater levels return to normal. Transient is over (Fig. 5.47a).

5.13.3 Abnormal Case

The abnormal case for Scenario 2 MBFP P1 $\rightarrow$ P4 duty swap entails leaving the P4 discharge valve MV18 CLOSED in human error after the P4 has been primed up and running ready for doing the swap with P1, which needs to be placed under maintenance as shown in process diagram Figure 5.48.

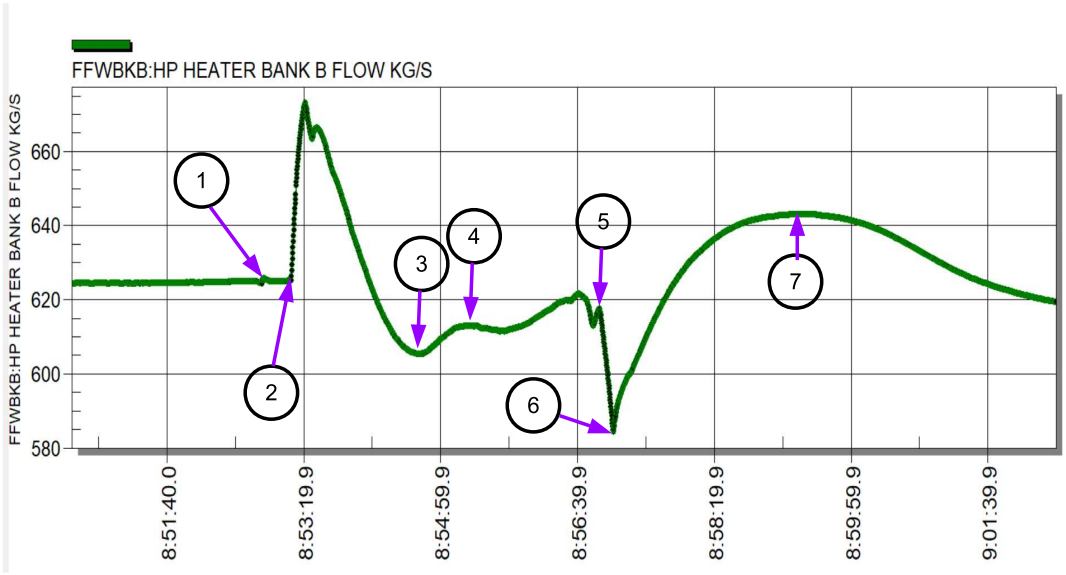
1) **P4 HS to ON, MV18 HS to CLOSE**: P4 hand switch is set to ON by the operator, P4 has started successfully in the field (WHITE Lamp is ON). Discharge MV18 hand switch is set to CLOSE by the operator, and it has been slowly closed fully (RED Lamp is ON) (Fig. 5.50(1)). This is the beginning of the HITL error precursor.

2) **MV15 HS to CLOSE/STOP**: MBFP P1 discharge MV15 is slowly closed by the operator by switching between CLOSE/STOP hand switch positions as per procedure. The boiler feed flow slow starts to decrease as only MBFP P2,P3 are supplying feed water, while MV18 CLOSED blocks P4 output.

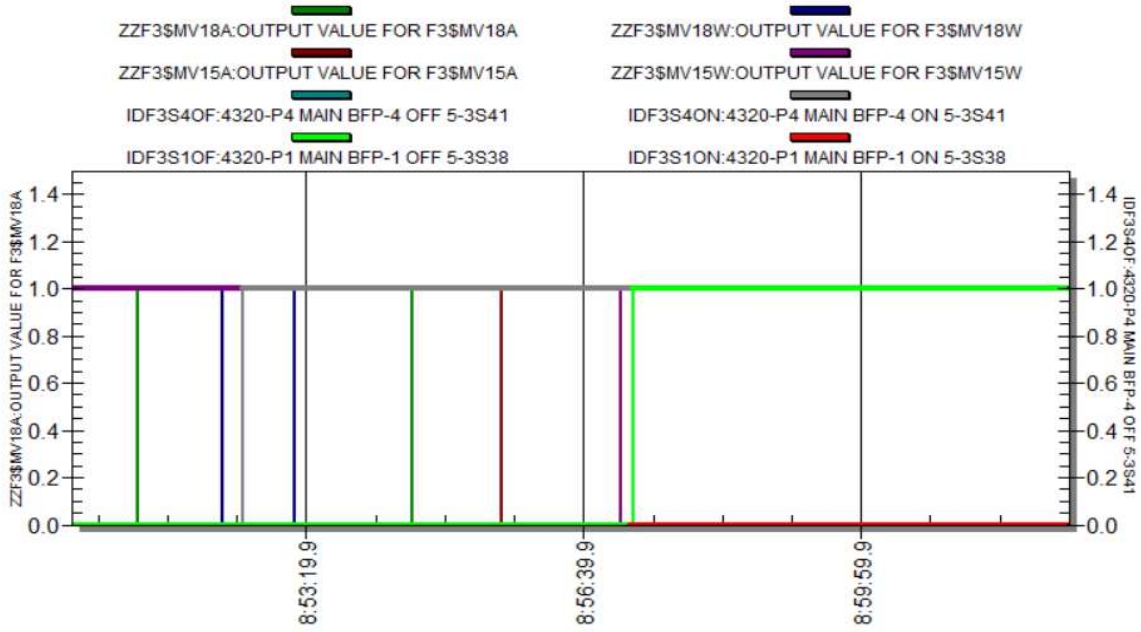
3) **MV15 HS to CLOSE**: MBFP P1 discharge MV15 is fully closed in field. MV15 RED Lamp is ON (Fig. 5.50(3)). Feed water flow drops sharply (Fig. 5.49a) as P1 flow discharge is fully isolated and P4 is isolated via MV18 CLOSED.

4) **P1 HS to OFF, MV15 HS to CLOSE**: MBFP P1 HS is set to OFF by operator, P1 stops running in field (RED Lamp is ON) (Fig. 5.50(3)). Feed water flow starts to increase to normal level as other MBFP pumps P2, P3 are throttled up automatically to approximately 90% (Fig. 5.49a).

5) **P1 HS OFF, P4 HS ON**: MBFP abnormal duty swap from P1 to P4 is complete. Feedwater levels stabilize to a new lower than normal flow rate that is 85%. Error has been committed and Transient is over (Fig. 5.49a).



(a)



(b)

Figure 5.47: (a) MBFP P1 → P4 duty swap showing impact on boiler feed flow process; (b) MBFP P1 → P4 duty swap showing MBFP control room panel indication and hand switch digital status evolution.

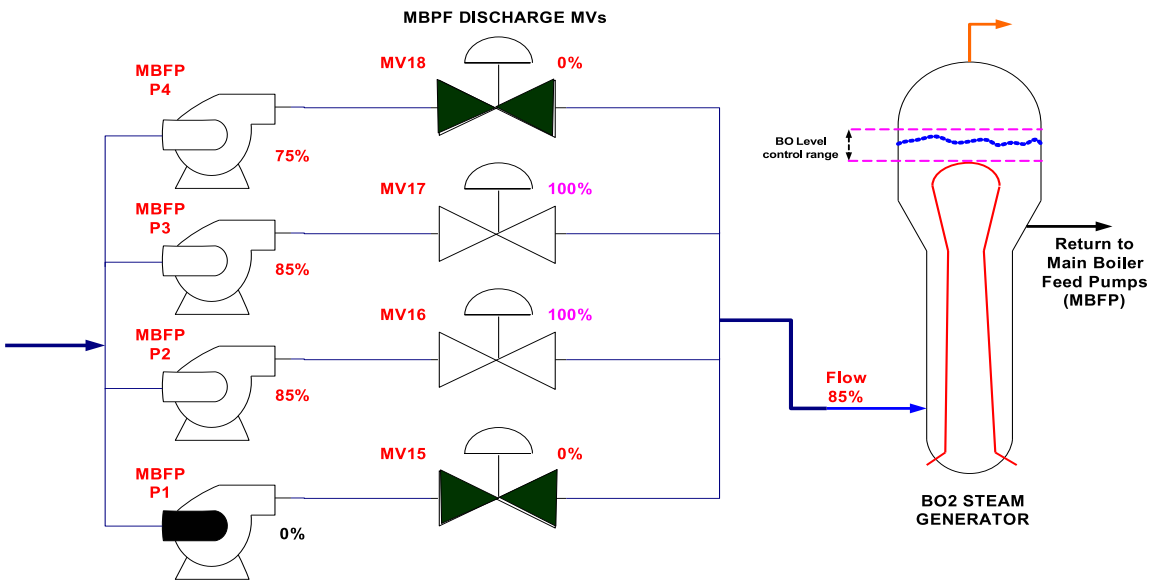


Figure 5.48: Scenario 2. Main Boiler Feed Pumps (MBFP) abnormal configuration. MBFP P4 ready and running at 75%, while MBFP P2, P3 are running at 85% load. But MBFP discharge valve MV18 has been left CLOSED in error while MBFP P1 has been placed under maintenance with CLOSED.

5.13.4 Simulator Data as Model Training Input

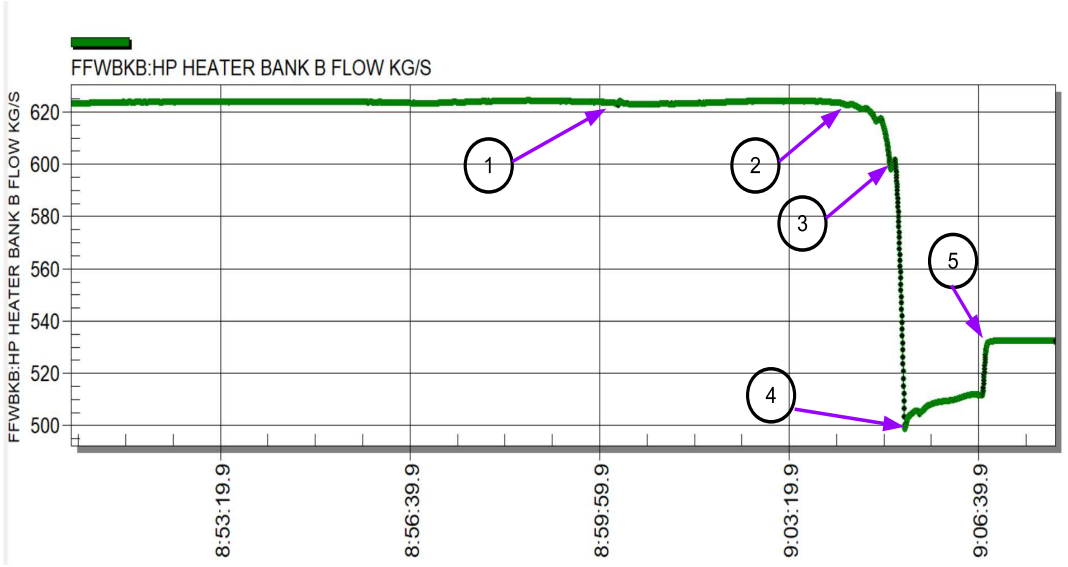
In order to transform the simulator data so it could be captured in the two features variables the Trident model expects as raw training data, the following data packing and transformation was done:

- Boiler Flow floating-point analog value was scaled by $\times 10$ and converted to an 8 bit value (0 to 250) (the remaining values are reserved for other START and STOP markers) for creating NLP supervised training dataset. The PROCESS feature used by the model represents this transformed boiler flow value.

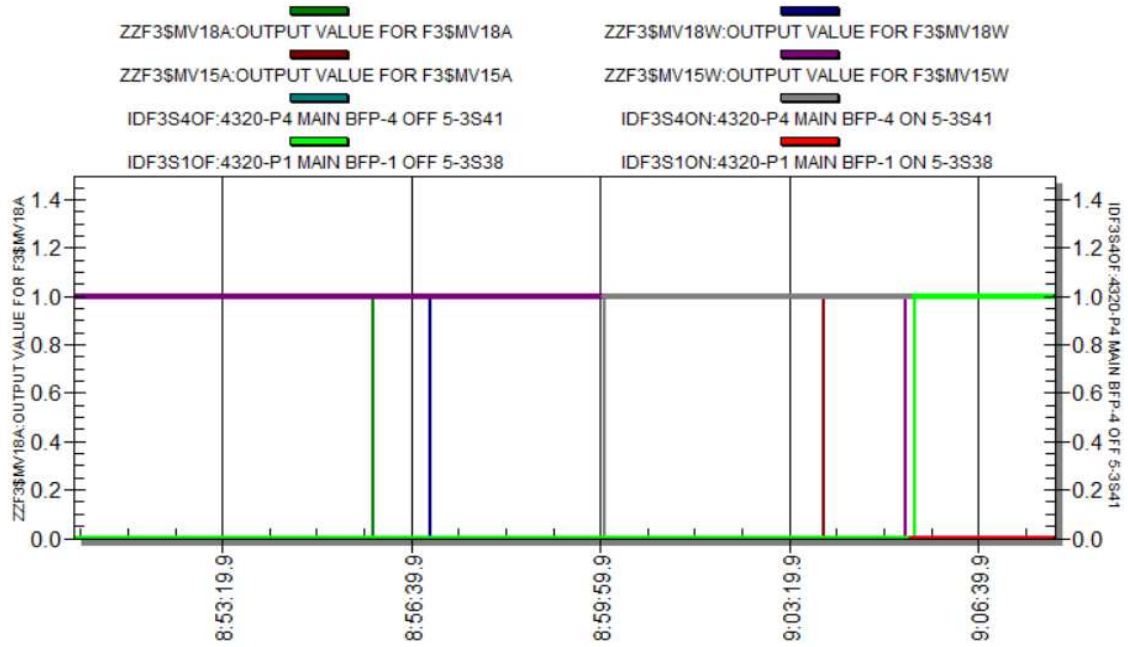
- The MBFP discharge MV field status indication lamps occupy half-word: AMBER(CLOSE), WHITE(OPEN) $(MV18A, MV18W, MV15A, MV15W)$.

  MBFP Pump Hand Switch position status ON, OFF (half-word):
  $(P4OFF, P4ON, P1OFF, P1ON)$

  These 8 digital values were packed in a 8 bit word vector in this bit order: (MV18A, MV18W, MV15A, MV15W,P4OFF, P4ON, P1OFF, P1ON) for HMI_USER feature that is used by the model. Therefore, HMI_USER feature effectively captures all HMI panel events, both process indications via Lamp status and operator actions via hand switch



(a)



(b)

Figure 5.49: (a) MBFP P1 → P4 abnormal duty swap showing impact on boiler feed flow process; (b) MBFP P1 → P4 abnormal duty swap showing MBFP control room panel indication and hand switch digital status evolution (there is a lot less activity compared to normal case Fig. 5.47b).

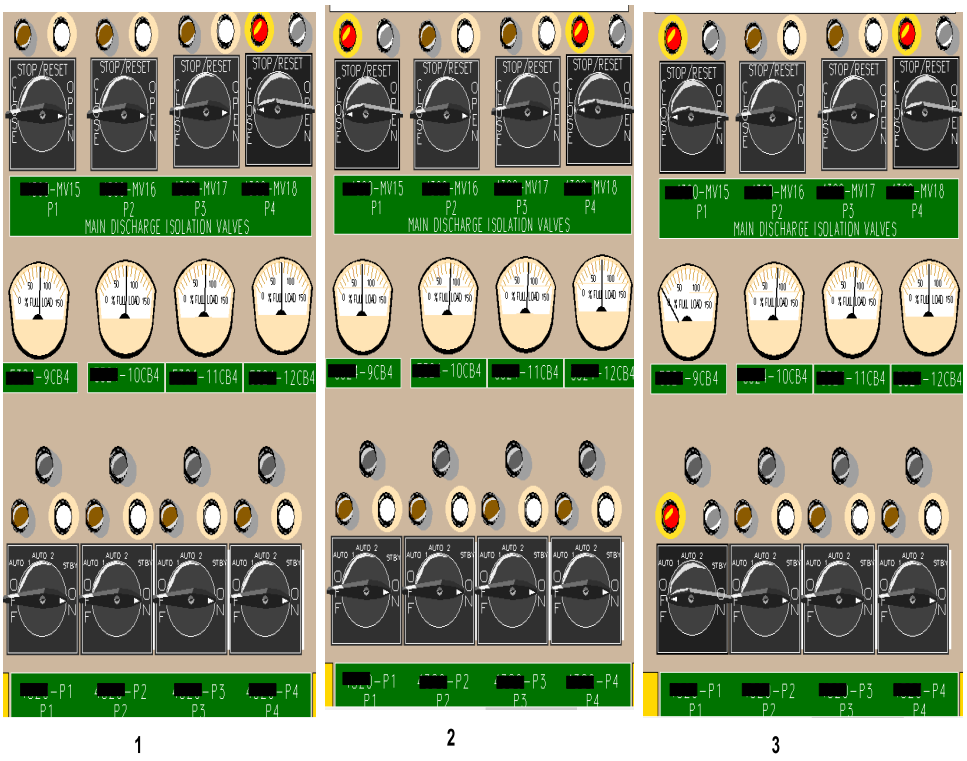


Figure 5.50: The MBFP control panel obtained from training simulator, Panel indication and switch shown in abnormal configurations.

(1) shows P4 hand switch on ON position while the WHITE Lamp is showing its running in the field. In addition, its load current meter (12CB4) is reading 70%. On top MV18 MBFP discharge valve hand switch is CLOSED, and RED lamps is confirming the same (CLOSED) field condition. All MBFP P1 to P4 pumps are running at approximately 70% - HITL Error precursor;

(2) The only difference is P1 discharge MV15 hand switch is changed to the CLOSED position, RED Lamp indicator shows MV15 is fully CLOSED. Effectively, MBFP P2 and P3 only are the only pumps providing feedwater. Therefore these are throttled at approximately 90% under automatic control;

(3) Shows P1 HS is set to OFF, P1 is stopped running in the field (meter reading 9CB4) at 0%, and RED Lamp is ON. P1 discharge MV15 is CLOSED in the field, RED Lamp is ON. P4 is ON, and MV18 is CLOSED - Accident Event.

positions.

The following state table (Fig. 5.51) is based on various bit combination values of above HMI_USER feature variable. This is created for easier interpretation of the MBFP control panel indication and hand switch states, which have been transformed into training dataset

values and the trained model output.

MV18A, MV18W, MV15A, MV15W, P4OFF, P4ON, P1OFF, P1ON			
BIT-PATTERN	HMI_STATE VALUE	OPERATOR ACTION - STATE LABEL	CONDITION
01010001	81	MV18 OPN, MV15 OPN, P4 OFF , P1 ON	NORMAL
11010001	209	MV18 MOVING, MV15 OPN, P4 OFF , P1 ON	NORMAL
10010001	145	MV18 CLOSE, MV15 OPN, P4 OFF , P1 ON	NORMAL
10010101	149	MV18 CLOSE, MV15 OPN, P4 ON, P1 ON	ABNORMAL
11010101	213	MV18 MOVING, MV15 OPN, P4 ON, P1 ON	NORMAL
01010101	85	MV18 OPEN, MV15 OPN, P4 ON, P1 ON	NORMAL
01110101	117	MV18 OPEN, MV15 MOVING, P4 ON, P1 ON	NORMAL
01100101	101	MV18 OPEN, MV15 CLOSE, P4 ON, P1 ON	NORMAL
01100100	100	MV18 OPEN, MV15 CLOSE, P4 ON, P1 SHUTTING	NORMAL
01100110	102	MV18 OPEN, MV15 CLOSE, P4 ON, P1 OFF	NORMAL
10110101	181	MV18 CLOSE, MV15 MOVING, P4 OFF, P1 ON	ABNORMAL
11010001	209	MV18 MOVING, MV15 OPN, P4 ON, P1 ON	NORMAL
10100101	165	MV18 CLOSE, MV15 CLOSE, P4 ON, P1 ON	BAD

Figure 5.51: MBFP control room panel state reference table. This may be used to interpret the HMI state and trained model output.

5.13.5 Model Demonstration Result

As done for previous scenario 1 results (Sec. 5.12.5), Trident model was selected and trained using curriculum learning (Sec. 4.7) using a raw dataset#1 containing approximately $14K$ samples and a separate test dataset containing approximately $14K$ samples as well. Both the raw training dataset#1 and the test dataset contain 2 instances of normal MBFP duty swap (P1 to P4) and 1 instance of abnormal LCV swap sequence of Scenario 2.

A second training dataset#2 that only contains a normal MBFP duty swap sequence of Scenario 2 contains $8.8K$ raw samples.

Training and Testing Methodology

Currently only one type of supervised training and validation data sets with: $K-LAG/N-Ahead$: $(32, 4)$ window samples is generated as per Figure 5.11 (Sec. 5.3) from the two raw dataset#1 and dataset#2.

Training with above two supervised datasets#1,#2 (containing combined normal/abnormal and normal sequences respectively) yielded two separate trained Trident models for Scenario 2, each with validation accuracy of 90% and 92% for the $K-LAG/N-Ahead$: $(32, 4)$ dataset case respectively.

The models will be referred to as *Tr_K32_N4_Sc2* (trained using datasets#1) and *Tr_K32_N4_Sc2Nor* (trained using normal datasets#2). As described previously in section 4.7, the Trident NLP model (Fig. 4.12) is trained using *curriculum* training which includes initially feeding it both K past samples of both PROCESS and HMI_USER features as encoder/decoder input and cor-

responding N ahead samples as decoder target reference output sequence (which it needs to learn).

Testing the trained models is done with a completely new out-of-sample raw test dataset containing $14K$ raw samples, 2 instances of normal, and 1 instance of abnormal MBFP duty swap transfer sequence. During testing Trident NLP model (Fig. 4.12) is made to forecast in *teacher forcing* mode, in which: both K past samples of both PROCESS and HMI_USER features as decoder input is provided. However, for decoder input, a *zero* valued input vector initialized with $< Start >$ token only is provided initially. Subsequently, as the model predicts more tokens for a given input sequence of $K - LAG$ samples, its predicted tokens are added to the decoder input vector until all the $N - Ahead$ tokens have been produced.

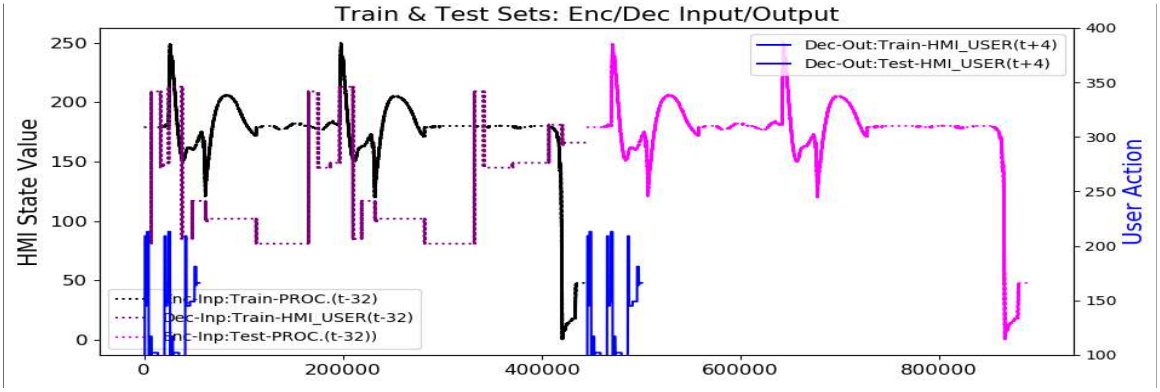


Figure 5.52: Example of combined normal and abnormal raw dataset supervised training and validation dataset for Scenario 2: $K - LAG/N - Ahead$: $(32, 4)$. Shows Decoder/Encoder PROCESS and HMI_USER input sequences and Decoder target train output sequences. (Validation dataset was set to 90% of training dataset).

Model Output Analysis

The Trident model is designed and trained to forecast tokens at $N - Ahead$ time steps for the HMI_USER feature, which is a composite vector containing both control panel indications and operator switch status (as per Sec. 5.13.4).

Model output shows the predicted output closely tracks the expected HMI_USER feature state sequence. However, model forecast samples are generated prior (seen at lower sample indexes) to the decoder input $K - LAG$ HMI_USER sequence (Fig. 5.53) that are presented as inputs to the decoder during inference mode.

Similar to scenario 1 (Sec. 5.13.5), scenario 2 supervised data sets contain the HMI states feature (PROCESS and HMI_USER) based on past $K - LAG$ input sequence. The process variable (PROCESS feature) consists of the scaled value of Boiler feedwater flow, and HMI_USER

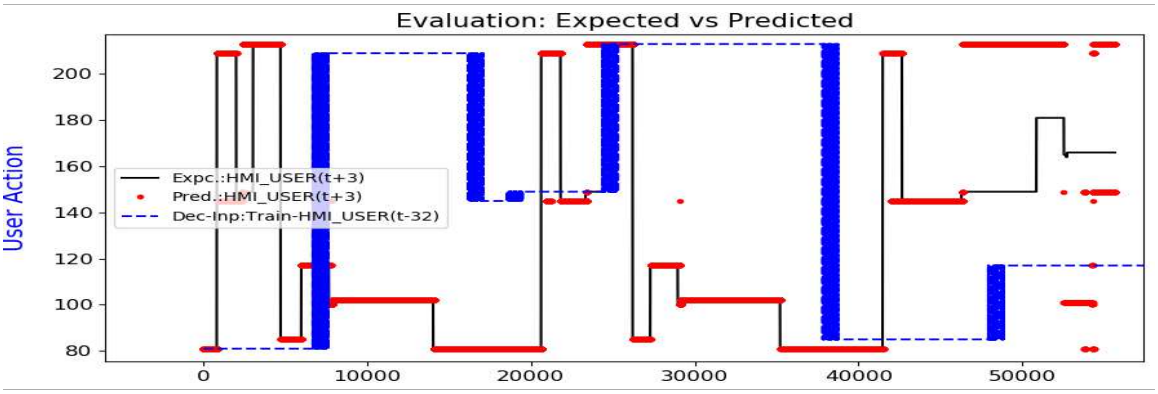


Figure 5.53: Example of Model Predicted vs. Expected Output from *Tr_K32_N4_Sc2Nor*. Model predicted output (red dots) closely tracks the expected HMI_USER feature (black) state sequence. The $K - LAG$ HMI_USER (blue) HMI events appear as inputs to the model after the model has forecast those same events (marked by stars) previously. This because the model is trained to do $N - Ahead$ time step forecast, as long as the input sequence is similar to what it was trained on.

feature consists of HMI MBFP panel indications (MV18, MV15 status Lamps) and (MBFP P1,P4) hand switch states).

More interestingly, the model forecast approximately at least 30 time steps prior to actual HMI state-changing (Fig. 5.54) as recorded in the raw dataset (direct output recorded from the simulator). This is due to the lower $N - Ahead = 4$ trained model, the prediction accuracy remains higher for longer time index forecast with high confidence level (compared to only 10 seen previously in Fig. 5.42 for *Tr_K32_N20_Sc1* model).

The rolling window average of predicted forecast (Fig. 5.54) is an effective visualization tool that averages the forecast samples belonging in the same temporal slice (as per Fig. 5.14 in Sec. 5.3.3)

Attempt to detect HITL Error

The following discussion provides an interpretation of the trained model forecast output that may aid in the potential detection of target HITL error precursors for MBFP Swap sequence scenario.

As described above in Sec. 5.13.3, the HITL error is introduced when P4 discharge valve MV18 is left CLOSED and not OPENED up after P4 has been started up.

Normal expected sequence as seen in terms of HMI_USER feature state transitions:

Corresponding Normal HMI_USER State transition:

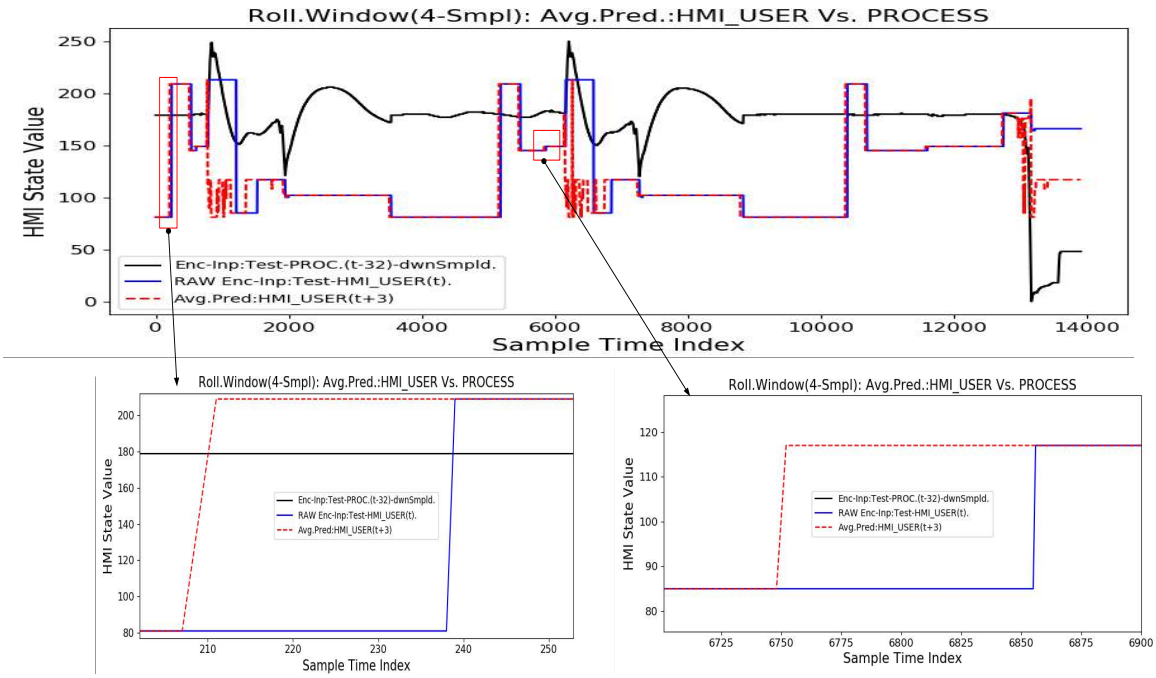


Figure 5.54: Rolling window average of Predicted vs. Raw Expected Model Output from *Tr_K32_N4_Sc2*. Rolling window average Model predicted output (red dashes) closely tracks the expected raw HMI_USER feature (blue) state sequence. The model forecast sequence is approximately atleast 30 time index prior;
The bottom zoomed-in boxes, show two particular HMI state transition events and compares the time indexes of the predicted and raw sequences;
As per the forecast, the first predicted rise (lower left plot) occurs at time index 207, while as per the raw dataset, this same transition occurs actually at index 237 in the test data set;
Similarly, for the second transition (lower right plot) forecast event occurs at index 6748 while the actual transition occurs much later at index 6855 in the test data set.

$81 \rightarrow 209 \rightarrow 145 \rightarrow 149 \rightarrow 213 \rightarrow 85 \rightarrow 117 \rightarrow 101 \rightarrow 100 \rightarrow 102 \rightarrow 81$

Corresponding Abnormal HITL Error Precursor sequence is:

$81 \rightarrow 209 \rightarrow 145 \rightarrow 149 \rightarrow 181$

The main mode of HITL error detection proposed is by looking for instances of sustained deviation between the forecast output and the actual HMI indication sequence. Such as demonstrated below in Figure 5.55 and Figure 5.56:

In, Figure 5.55 *Tr_K32_N4_Sc2* model which is trained on combined normal and abnormal sequence is used. During training, the abnormal sequence (decoder/encoder) input states were manually modified to include few marker states, while the rest of the normal sequence states were left unaltered. This was done to intentionally allow the model to learn to differentiate when normal and abnormal sequences are encountered. This, however, did not produce the intended behavior in the forecast, mainly owing to insufficient instances of abnormal se-

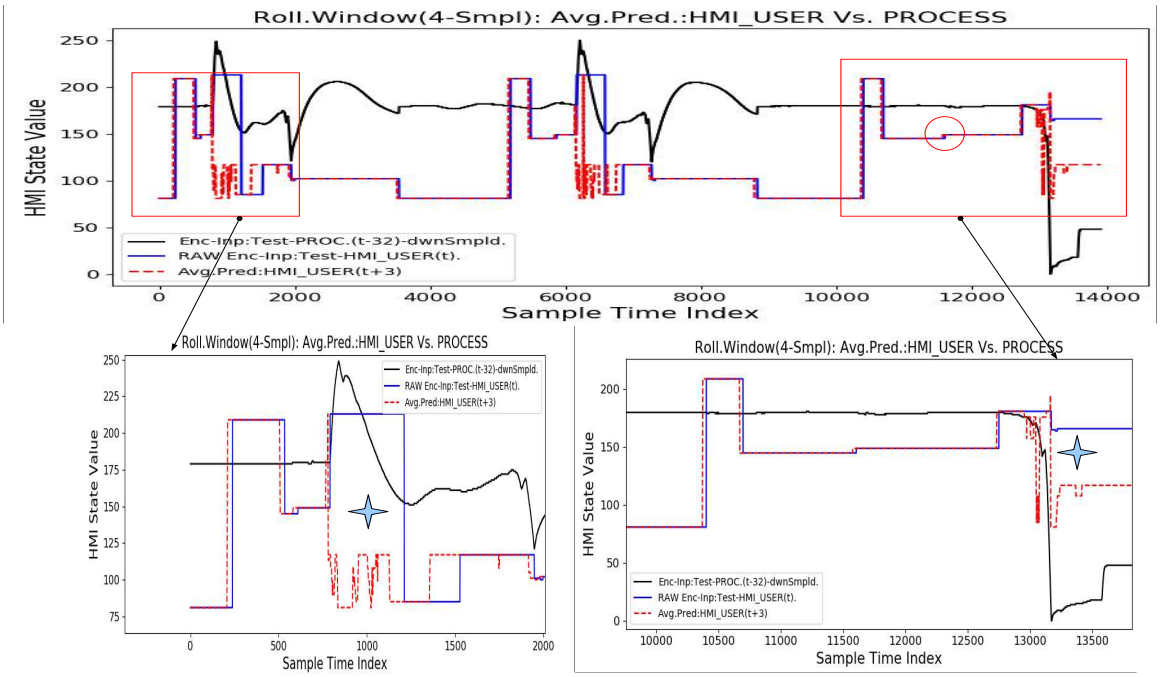


Figure 5.55: HITL Error detection with Rolling window average of forecast output vs. Raw actual HMI states using *Tr_K32_N4_Sc2* model (trained on combined normal and abnormal sequence dataset#1) did not detect the precursor prior to accident event;

The zoomed-in boxes show two particular HMI state transition of normal and abnormal events;

As per the forecast, the normal sequence of MV18 (OPEN, MOVING, CLOSE, MOVING, OPEN) (lower left plot) (blue) is moderately tracked by the model output forecast (red dashed) showing deviation between the two for a shorter span (shown by star);

Compared, to the case of HITL error precursor initiates when MV18 (OPEN, MOVING, CLOSE) sequence occurs, the deviation between actual model forecast (red dashed) and actual HMI state (blue) only occurs and persists for a longer duration span (shown by star) after the HITL error precursor has passed (shown by red circle).

quences in the dataset. Moreover, the modified states also negatively affected modeling the normal sequence with sufficient accuracy, as seen with deviations in the lower left plot in Fig. 5.55, shown by the star symbol. HITL error precursor was also not sufficiently identified, as seen in the lower right plot in Fig. 5.55), since the model is suspected to be overfitting. This may be due to due to sufficient instances of abnormal sequences in the training dataset.

Despite being unable to detect HITL error precursor, the model manages to detect the actual accident event state, as seen in the lower right plot in Fig. 5.55) as large deviations between forecast and actual HMI state sequence (shown by star symbol).

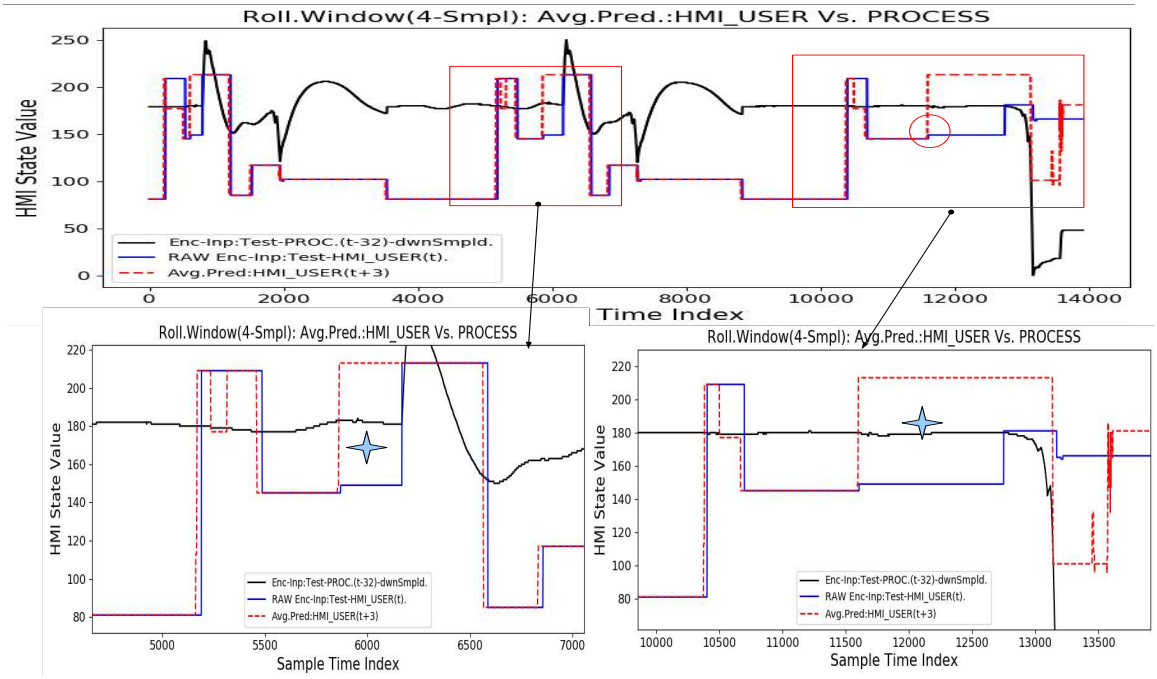


Figure 5.56: HITL Error detection with Rolling window average of Forecast vs. Raw actual HMI sequence using *Tr_K32_N4_Sc2Nor* model (trained on normal sequence dataset#2 only). This model detects the HITL error precursor more effectively;

The zoomed-in boxes, show two particular HMI state transition of normal and abnormal events;

As per forecast, the normal sequence of MV18 (OPEN, MOVING, CLOSE, MOVING, OPEN) (lower left plot) (blue) is moderately tracked by the model output forecast (red dashed) showing deviation between the two for a shorter span (shown by star);

Compared, to the case of HITL error precursor initiates when MV55 (OPEN, MOVING, CLOSE) sequence occurs, the deviation between actual model forecast (red dashed) and actual HMI state (blue) increases abruptly and persists for a longer duration span (shown by star) right after the HITL error precursor has occurred (shown by red circle).

In, Figure 5.56 *Tr_K32_N4_Sc2Nor* model which is trained only on normal sequences (dataset#2) is used. This model tracks more closely the actual HMI sequence. It automatically behaves very differently once a partial sequence (HITL error precursor) is encountered (lower right plot in Fig. 5.56), where the deviation between actual and forecast sequence persists for a longer time span. This abrupt increase in the model ouput is similar to the learned behavior that is exhibited earlier from the model trained on normal sequences, as shown in the lower left plot in Fig. 5.56.

In summary, above results demonstrate using forecast models that are trained on com-

bined abnormal and normal sequence dataset (e.g. *Tr_K32_N4_Sc2* Fig. 5.55) or only trained on normal sequence dataset (e.g. *Tr_K32_N4_Sc2Nor* Fig. 5.56) can potentially be used to detect HITL error precursors in scenario 2 for MBFP swap. The caveat being the model, must be trained on several normal and abnormal sequences to be able to accurately forecast the HMI states for future time steps (in general to reduce over-fitting and obtain a more generalized model). However, in the above results, the model that is only trained on normal sequences does perform well in detecting the HITL error given the limited amount of training samples it had during training.

The tradeoff between normal only vs. combined abnormal and normal dataset trained models, is that the former is more sensitive to detecting any sequence change. In contrast, the latter model is more generalized but requires a larger dataset and careful special filters tuned to detect pattern deviation anomalies to detect particular HITL error precursors.

The main take away from the above scenario test results are briefly summarized in Table 5.16.

Table 5.16: NPP Simulator Scenario Test Result Summary

Test	**Result Highlights**	**Main Take Away**
Scenario 1 Test; LCV Swap	• NLP Trident model utilized. • Two models trained using a combined normal and abnormal training dataset for K-lag/N-Ahead: (32/20) and (32/10) (*Tr_K32_N20_Sc1*, *Tr_K32_N10_Sc1* respectively) • One model trained on normal training data set for K-lag/N-Ahead: (32/10) (*Tr_K32_N10_Sc1Nor*) • Model trained on combined normal and abnormal sequences can approximately forecast 10 time steps ahead the actual HMI state occurring in test dataset. • HITL error precursors are detected as deviation between forecast and actual test HMI state sequence.	• Model trained on combined normal and abnormal sequence training set detects HITL error precursor with narrow margin of deviation between forecast and actual HMI sequence. • Model trained on only normal sequence training set is more sensitive and is able to detect HITL error precursor with wider margin of deviation between forecast and actual HMI sequence.
Scenario 2 Test; MBFP Swap	• NLP Trident model utilized. • One model trained using a combined normal and abnormal training dataset for K-lag/N-Ahead: (32/4) (*Tr_K32_N4_Sc2).* • Second model trained using a only normal training dataset for K-lag/N-Ahead: (32/4) (*Tr_K32_N4_Sc2).*	• Model trained on combined normal and abnormal sequence training is not able to detect HITL error precursor. • Model trained on only normal sequence training set is more sensitive and is able to detect HITL error precursor with wider margin of deviation between forecast and actual HMI sequence.

5.14 Summary

This chapter covered the experiment setup and detailed discussion of results. The results may be summarized as the following:

ViDAQ Test and Results. Two test setup: one evaluated ViDAQ functionality as an integrated component of a prototype *EYE-on-HMI* system (HMI-Guard) which demonstrated the capability to track operator activity level in real-time. The second set of tests evaluate multi-dial gauge and control room lamp indications reading independently. Both static image datasets and live camera feed were utilized as a source of images. Multi-dial reading from a distance upto $1.5m$ showed 90% accuracy. Lamp indications could be read without any errors from a distance upto $2m$. Although the coverage of tests and results is limited, these tests were conducted for the research objective of confirming the feasibility of ViDAQ in control room applications.

Synthetic Data Generation. A utility that is utilized to generate synthetic data for studying forecast capabilities of various regression (ARIMA), RNN (LSTM and CNN), and NLP (RNN and CNN *Seq2Seq*) models. Also provided herein are synthetic data generation parameters for training and test data sets used for all model evaluation. In addition, the baseline persistence model for the synthetic dataset is described, and its results are included. A custom forecast error test metric: Rolling window Root Mean Square Error (*rollWinRMSE*) that is used in addition to standard RMSE is discussed as well.

ARIMA Model Test Setup. This section discussed the evaluation methodology for ARIMA models, which involves doing stationarity checks, followed by doing In-sample (*InS*), Out-of-Sample (*OuS*) testing of static, dynamic and adaptive models. For each tests, model forecast performance is compared against the persistence score and RMSE for a given *n-ahead* window forecast.

RNN and CNN Model Results. The section covered a discussion of results of all time-series RNN and CNN models. All the models implemented are standard architectures based on existing research publications. The application of these out of the box (unoptimized) models on the custom synthetic HMI state datasets generated previously was compared.

NLP Models Results. This section provided a discussion of results for both Seq2Seq NLP LSTM and CNN Encoder-Decoder models. These models approach the HMI state forecast as a language translation task.

Trident Model - HITL Error Detection With Real NPP Scenarios. This section provided a description of two actual plant scenarios. The scenarios were set up on full scope

CANDU^{TM} Darlington Nuclear Operator Training Simulator (NOTS) available in **Ontario Tech University** that accurately simulated the sequence of process and control room panel HMI states. The data collected from the was used to train and demonstrate the capability of the Trident model (custom NLP *Seq2Seq* Encoder-Decoder CNN based model) in the detection of HITL error prior to the actual accident event.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Current technological advancements in deep learning and computer vision can be applied to improve human performance of control room operators and reduce industrial accidents. Industrial control room panels, such as in Nuclear Power Plants, are laden with HMI devices that visually convey information to operators. Tapping into this visual information stream and using it to infer operator actions poses an entirely new challenge.

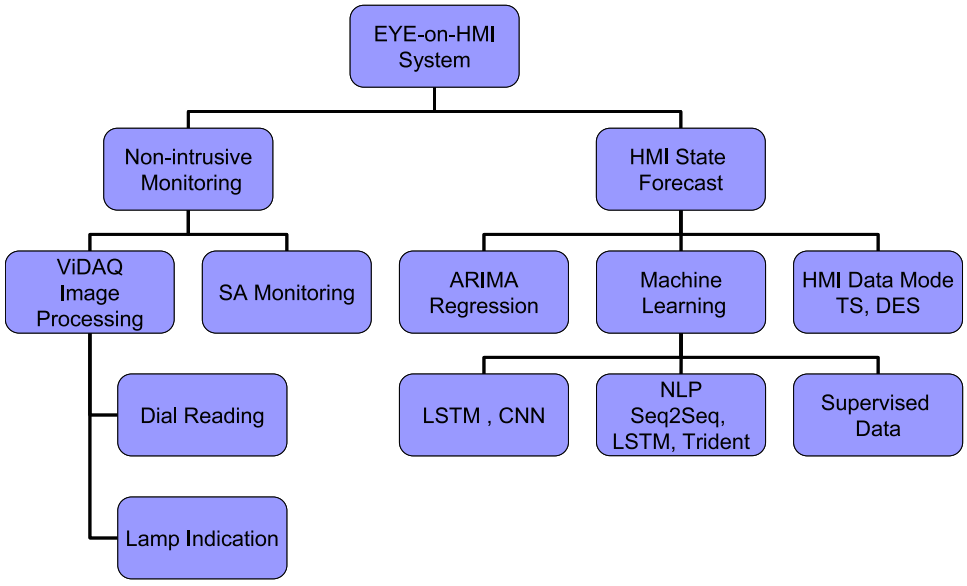


Figure 6.1: Research solution space shows span of solutions areas implemented and evaluated in this thesis.

In this dissertation, the concept of non-intrusive real-time monitoring of Human Machine Interface (HMI) state patterns is explored as a means of inferring operator situational aware-

ness in real-time. Which ultimately shall allow early detection of human-in-the-loop (HITL) error precursors in complex industrial control room environments.
Addressing the above research goals and based on Cyber-physical system philosophy as a design guide, two frameworks are proposed: Supervisory System on HMI (*EYE-on-HMI*) and Visual Data Acquisition (ViDAQ). *EYE-on-HMI* framework embodies non-intrusive monitoring with the aid of ViDAQ (Visual Data Acquisition) to achieve Human-in-the-loop (HITL) error detection using HMI state pattern forecasting models. These two conceptual system frameworks have been published in [7, 160] thus far.

The following lists the key conclusions for each solution (Fig. 6.1) presented in this thesis:

(I) **ViDAQ Components**: This study highlights several technical challenges and scalability concerns. The results and observations with ViDAQ have been published in [160, 165] thus far. The main contribution of ViDAQ is, the concept of non-intrusive monitoring of HMI states is demonstrated as achievable using computer vision in a control room environment.

(II) **ARIMA HMI State Modeling**: The results of the ARIMA models were published in [83]. This concludes with ARIMA models as being less suitable for future practical *EYE-on-HMI* system implementation due to the large volume of data to be modeled. They also require extensive pre-training data sanitization effort and data series stationary checks. Moreover, ARIMA does not offer the versatility and scalability required to include more extensive features in multi-variate time-series models, which are computationally less intensive to train compared to current deep learning models.

(III) **LSTM and CNN HMI State Modeling**: The results of this comparative study are published in [135]. This concludes that both LSTMs and CNNs were found to be apt in modeling HMI time-series data. However, due to the limited amount of data available, the RNN and CNN models show signs of over-fitting.

(IV) **DES HMI State Modeling using NLP**: NLP models offer far superior potential with the use of attention layers and word embeddings to capture complex dependencies among various HMI states while taking into account the context of various. This is consistent with how NLP models translate natural human languages.

(V) **HITL Error Precursor Detection**: The forecast models showed useful skills in learning the normal and abnormal control room HMI state sequences obtained form the NPP control room simulator. NLP models demonstrated the capability to detect target HITL

error precursors within the expected time window prior to the actual accident event occurring.

In summary, this thesis presents a novel system framework for non-intrusive monitoring of HMI states for detecting HITL error precursors. Successful demonstration of the proof-of-concept approach showed the research goals are practically achievable within the limitations and assumptions as identified in this thesis.

6.2 Limitations

It is essential to note that the *EYE-on-HMI* does not rely on direct monitoring of operator actions. Instead, *EYE-on-HMI* relies on the HMI state changes owing to direct operator actions, that is available as visual feedback from the HMI operator input device, E.g., push-buttons, set-point displays, etc. Therefore, HMI input devices are assumed to have some visual feedback for *EYE-on-HMI* to function as intended. In the absence of any visual feedback from a particular HMI input device, it may be challenging for *EYE-on-HMI* to recognize operator actions and, in that case, will require this information to be input via an alternative interface.

If the HMI is faulty, two cases expose certain limitations:

1. **False-Negative Case**: In this case, the operator action is correct, but it is not registered positively in *EYE-on-HMI* as it was expecting to receive. Consequently, this case will raise the alarm suggesting correct visual feedback corresponding to an affirmative operator action is not captured. A higher False-Negative rate is expected to raise nuisance alarms or erroneous events being logged in the *EYE-on-HMI* system.

2. **False-Positive Case**: In this case, the operator action is actually not performed, but it is registered positively as an operator action in *EYE-on-HMI* despite it not expecting to receive any. This situation will also be alarmed. However, False-Positive rate is expected to be low since most control room devices are normally fail-safe.

If *EYE-on-HMI* system is monitoring, an alarm for either False-Positive and False-Negative are expected to be captured in real-time, leading to early diagnosis of the fault on the HMI. However, the scope of addressing nuisance alarms, is an identified delimitation in this study and maybe the scope of future research.

Secondly, approval to use *EYE-on-HMI* in NPP control rooms may be challenging initially. Since *EYE-on-HMI* is not a deterministic expert system, which increases the difficulty to obtain necessary software qualifications (Sec. 2.2.3) required for an approved operator aid tool.

Closely related to software qualification is the issue of the responsibility-gap associated with the use of deep machine trained and AI models in safety-critical applications. However, in the case of *EYE-on-HMI*, it is intended as a monitoring system and does not issue any operational decisions; therefore, the responsibility-gap may not be as critical. Moreover, camera-based non-intrusive monitoring in NPP control room may also not be permitted due to nuclear security reasons (Sec. 1.2). Nevertheless, *EYE-on-HMI* may be useful in non-safety critical applications such as in full-scope NPP training simulators.

Lastly, an operator usability trial of *EYE-on-HMI* in a full-scope NPP training simulators is not conducted in the current scope. This will be required to validate the results obtained from the system to be considered an acceptable measure for monitoring situational awareness in real-time. This is also an identified delimitation in this study and can be the scope of future research.

6.3 Future Work

The novel concept of non-intrusive monitoring and *EYE-on-HMI* opens several related areas of research and development.

Key design challenges that ultimately need to be addressed to realize a viable production grade solution for industry application are listed below along with potential opportunities to overcome them.

6.3.1 Accuracy and Reliability

How accurately can the information from HMI panel indicators be acquired visually in real-time using computer vision? How can system reliability be ensured such that *EYE-on-HMI* operates as intended in certain performance-limiting scenarios such as obstructed view, damaged video camera, poor lighting condition, image vibration, etc.?

The following opportunities may be explored to address above challenges.

1. **Distributed Camera Networks**: Using an a distributed camera network can improve the accuracy limiting scenarios by observing the control panels from different viewpoints to avoid obstructions.

2. **Cross-Validation**: Applying current control room operator practices to address the above challenges is a viable approach. Currently, operators compare the visually acquired information against the real-time process data available through the plant information system. Any cross-validation abnormalities are flagged. Similarly, a designed

self (automated) cross-validation routines in *EYE-on-HMI* system can improve system reliability.

3. **System Redundancy** Duplicating validation processes on separate hardware and software systems running in tandem can also improve system availability during performance-limiting scenarios and load-balancing.

6.3.2 Scalability

Multi-video stream processing can pose heavy computational demand on system memory and storage capacity. But the ability to store the HMI data can be vital for enabling human performance trending, predictive equipment maintenance, and post transient reviews, which presents a Big Data management challenge.

Fortunately, owing to Big Data research, several potential solutions exist. Open source and enterprise real-time data processing frameworks with proven industry track records, such as Microsoft DFS, Apache Spark, Storm and Samza offer distributed, in-memory large-scale data and stream processing for graph and machine learning algorithms based on Hadoop YARN (MapReduce) architecture which can access data stored on HDFS.[1] Other notable enterprise class solutions for data stream processing include Amazon Web Services Keinesis - works with Amazon cloud IaaS EC3 elastic computing and S3, Redshift storage services; Microsoft StreamInsight - a high throughput complex event processing platform that can run queries on event live streams.

6.3.3 Extensibility

HMI's are seemingly ubiquitous to any CPS requiring a human command interface. Therefore EYE-on-HMI framework must be extensible to monitor a variety of HMI devices found in a control room environment. This requirement may take a twofold practical approach: (1) build a digital library that captures characteristics of standard HMI devices found in the end-of-use industry such as gauge dials, bar charts, alarm lights, etc., and, (2) develop a standard format to digitize HMI designs referencing HMI objects from the former library.

6.3.4 Immediate Future Goals

In addition to the above, the following immediate future works are proposed:

[1]HDFS - Hadoop Distributed File System is a java based file system that provides scalable and reliable data storage on server clusters.

Advanced ViDAQ ViDAQ system development specific to an area of industrial automation systems is possible. For example, developing a library of camera-based remote monitoring solutions to read typical control room devices and doing field validation for each. A scalability study could be conducted on the robustness and practicality of doing full control room monitoring using ViDAQ. Such a study could also be carried out on a smaller scale such as monitoring car instrument panel and road conditions

High Fidelity HMI State Sequence Data *EYE-on-HMI* framework requires the initial training data set to be able to forecast HMI states for future time steps. Therefore, one future work can address collecting domain-specific data collected over a considerable period in high fidelity (multi-feature parameters logged at appropriate sampling frequency) and using several users.

Human-in-the-Loop Training A new burgeoning field of human-in-the-loop or human-assisted training of deep learning models [85, 86] is used to improve the forecasting skill of these models and to accelerate learning. This new technique should be evaluated in *EYE-on-HMI* framework as it may allow the HMI state forecast models to learn on-line from actual operators under the training environment as they make specific HILT errors. Such trained models can be standardized for licensing exams of qualified professionals such as nuclear operators and aviation pilots.

Bibliography

[1] IAEA. International nuclear event scale (ines). Accessed on: Jan. 10, 2020. [Online]. Available: http://www-ns.iaea.org/tech-areas/emergency/ines.asp

[2] E. Hollnagel, *Human reliability analysis: Context and control*. Academic press, 1993.

[3] IAEA. (1992) Insag-7 safety report the chernobyl accident. Accessed on: Jan. 10, 2020. [Online]. Available: https://www-pub.iaea.org/MTCD/publications/PDF/Pub913e_web.pdf

[4] Nuclear Energy Institute. (2014) Lessons from the 1979 accident at three mile island. Accessed on: Jan. 10, 2020. [Online]. Available: https://www.nei.org/resources/fact-sheets/lessons-from-1979-accident-at-three-mile-island

[5] Atomic Energy of Canada Limited, Chalk River Laboratories. (1992) The chalk river accident in 1952. Accessed on: Jan. 10, 2020. [Online]. Available: http://www.nuclearfaq.ca/The_CR_Accident_in_1952_WG_Cross1980.pdf

[6] D. Manzey, J. Reichenbach, and L. Onnasch, "Human performance consequences of automated decision aids: The impact of degree of automation and system experience," *Journal of Cognitive Engineering and Decision Making*, vol. 6, no. 1, pp. 57–87, 2012.

[7] H. V. Singh and Q. H. Mahmoud, "Eye-on-hmi: A framework for monitoring human machine interfaces in control rooms," in *Electrical and Computer Engineering (CCECE), 2017 IEEE 30th Canadian Conference on*. IEEE, 2017, pp. 1–5.

[8] Q. Chou, Kramer *et al.*, "Experience with an expert system technology for real-time monitoring and diagnosis of industrial processes," in *IAEA specialists meeting on" Monitoring and diagnosis systems to improve nuclear power plant reliability and safety", Gloucester, GB*, 1996.

[9] E. Hollnagel and D. D. Woods, *Joint cognitive systems: Foundations of cognitive systems engineering*. CRC Press, 2005.

[10] T. Samad and A. Annaswamy, "The impact of control technology," *IEEE Control Systems Society*, 2011.

[11] G. Schirner *et al.*, "The future of human-in-the-loop cyber-physical systems," *Computer*, no. 1, pp. 36–45, 2013.

[12] H. Ye *et al.*, "A human reliability analysis method based on cognitive process model for risk assessment," in *Intelligent Rail Transportation (ICIRT), 2016 IEEE International Conference on.* IEEE, 2016, pp. 418–424.

[13] K. L. Mosier *et al.*, "Judgment and decision making by individuals and teams: issues, models, and applications," *Reviews of Human factors and Ergonomics*, vol. 6, no. 1, pp. 198–256, 2010.

[14] J. Rasmussen, "Information processing and human-machine interaction," *An approach to cognitive engineering*, 1986.

[15] P. D'Addario and B. Donmez, "The effect of cognitive distraction on perception-response time to unexpected abrupt and gradually onset roadway hazards," *Accident Analysis & Prevention*, vol. 127, pp. 177–185, 2019.

[16] L. Qin, Z. R. Li, Z. Chen, M. A. Bill, and D. A. Noyce, "Understanding driver distractions in fatal crashes: An exploratory empirical analysis," *Journal of Safety Research*, vol. 69, pp. 23–31, 2019.

[17] M. R. Endsley, "Automation and situation awareness," *Automation and human performance: Theory and applications*, vol. 20, pp. 163–181, 1996.

[18] C. D. Wickens, "Situation awareness: Review of mica endsley's 1995 articles on situation awareness theory and measurement," *Human factors*, vol. 50, pp. 397–403, 2008.

[19] J. Lundberg, "Situation awareness systems, states and processes: a holistic framework," *Theoretical Issues in Ergonomics Science*, vol. 16, pp. 447–473, 2015.

[20] K. McAnally, C. Davey, D. White, M. Stimson, S. Mascaro, and K. Korb, "Inference in the wild: A framework for human situation assessment and a case study of air combat," *Cognitive science*, vol. 42, pp. 2181–2204, 2018.

[21] A. D. Swain and H. E. Guttmann, "Handbook of human-reliability analysis with emphasis on nuclear power plant applications. final report," Sandia National Labs., Albuquerque, NM (USA), Tech. Rep., 1983.

[22] W. Yao and S. Zupei, "Cream—-a second generation human reliability analysis method," *Industrial Engineering and Management*, vol. 10, no. 3, pp. 17–21, 2005.

[23] L. Mu, B. Xiao, W. Xue, and Z. Yuan, "The prediction of human error probability based on bayesian networks in the process of task," in *Industrial Engineering and Engineering Management (IEEM), 2015 IEEE International Conference on.* IEEE, 2015, pp. 145–149.

[24] A. Mackieh and C. Cilingir, "Effects of performance shaping factors on human error," *International journal of industrial ergonomics*, vol. 22, no. 4-5, pp. 285–292, 1998.

[25] H. S. Blackman, D. I. Gertman, and R. L. Boring, "Human error quantification using performance shaping factors in the spar-h method," in *Proceedings of the human factors and ergonomics society annual meeting*, vol. 52, no. 21. SAGE Publications Sage CA: Los Angeles, CA, 2008, pp. 1733–1737.

[26] H.-C. Lee, T.-I. Jang, and K. Moon, "Anticipating human errors from periodic big survey data in nuclear power plants," in *Big Data (Big Data), 2017 IEEE International Conference on.* IEEE, 2017, pp. 4777–4778.

[27] Y. Chang and A. Mosleh, "Cognitive modeling and dynamic probabilistic simulation of operating crew response to complex system accidents: Part 1: Overview of the idac model," *Reliability Engineering & System Safety*, vol. 92, no. 8, pp. 997–1013, 2007.

[28] P. M. Salmon, N. A. Stanton, G. H. Walker, D. Jenkins, D. Ladva, L. Rafferty, and M. Young, "Measuring situation awareness in complex systems: Comparison of measures study," *International Journal of Industrial Ergonomics*, vol. 39, pp. 490–500, 2009.

[29] D. G. Jones and M. R. Endsley, "Examining the validity of real-time probes as a metric of situation awareness," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 1, 2000, pp. 278–278.

[30] M. R. Endsley, S. J. Selcon, T. D. Hardiman, and D. G. Croft, "A comparative analysis of sagat and sart for evaluations of situation awareness," in *Proceedings of the human factors and ergonomics society annual meeting*, vol. 42. SAGE Publications Sage CA: Los Angeles, CA, 1998, pp. 82–86.

[31] W. R. Nelson *et al.*, "Reactor: An expert system for diagnosis and treatment of nuclear reactor accidents." in *AAAI*, 1982, pp. 296–301.

[32] ——, "Response trees and expert systems for nuclear reactor operations," EG and G Idaho, Inc., Idaho Falls (USA), Tech. Rep., 1984.

[33] H. Qudrat-Ullah, "Qes-shell: An expert system for nuclear power plant operator's training," in *2012 Third International Conference on Intelligent Systems Modelling and Simulation*. IEEE, 2012, pp. 151–156.

[34] G.-H. Chou *et al.*, "The development of a thermal performance diagnostics expert system for nuclear power plant," *IEEE transactions on nuclear science*, vol. 41, no. 5, pp. 1729–1735, 1994.

[35] W. Khan, D. Ansell, K. Kuru, and M. Bilal, "Flight guardian: autonomous flight safety improvement by monitoring aircraft cockpit instruments," *Journal of Aerospace Information Systems*, vol. 15, no. 4, pp. 203–214, 2018.

[36] I. Zolotová and L. Landryová, "Knowledge model integrated in scada/hmi system for failureprocess prediction." *WSEAS Transactions on Circuits and Systems*, vol. 4, no. 4, pp. 309–318, 2005.

[37] T. Skripcak and P. Tanuska, "Utilisation of on-line machine learning for scada system alarms forecasting," in *2013 Science and Information Conference*. IEEE, 2013, pp. 477–484.

[38] S. Suzuki, F. Harashima, Y. Pan, and K. Furuta, "Skill analysis of human in machine operation," in *Neural Networks and Brain, 2005. ICNN&B'05. International Conference on*, vol. 3. IEEE, 2005, pp. 1556–1561.

[39] K.-M. Cha and H.-C. Lee, "A novel qeeg measure of teamwork for human error analysis: An eeg hyperscanning study," *Nuclear Engineering and Technology*, vol. 51, no. 3, pp. 683–691, 2019.

[40] I. Behoora and C. S. Tucker, "Machine learning classification of design team members' body language patterns for real time emotional state detection," *Design Studies*, vol. 39, pp. 100–127, 2015.

[41] T. Danisman *et al.*, "Drowsy driver detection system using eye blink patterns," in *Machine and Web Intelligence (ICMWI), 2010 International Conference on*. IEEE, 2010, pp. 230–233.

[42] A. Tustin, "The nature of the operator's response in manual control, and its implications for controller design," *Journal of the Institution of Electrical Engineers-Part IIA: Automatic Regulators and Servo Mechanisms*, vol. 94, no. 2, pp. 190–206, 1947.

[43] J. Ragazzini, "Engineering aspects of the human being as a servo-mechanism," in *Meeting of the American Psychological Association*, 1948.

[44] J. L. Cabrera and J. G. Milton, "On-off intermittency in a human balancing task," *Physical Review Letters*, vol. 89, no. 15, p. 158702, 2002.

[45] A. Kapoor and R. W. Picard, "Multimodal affect recognition in learning environments," in *Proceedings of the 13th annual ACM international conference on Multimedia.* ACM, 2005, pp. 677–682.

[46] ——, "A real-time head nod and shake detector," in *Proceedings of the 2001 workshop on Perceptive user interfaces.* ACM, 2001, pp. 1–5.

[47] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013.

[48] L. Xia, C.-C. Chen, and J. Aggarwal, "View invariant human action recognition using histograms of 3d joints," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on.* IEEE, 2012, pp. 20–27.

[49] E. Lloyd, S. Huang, and E. Tognoli, "Improving human-in-the-loop adaptive systems using brain-computer interaction," in *Proceedings of the 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems.* IEEE Press, 2017, pp. 163–174.

[50] D. Eskins and W. H. Sanders, "The multiple-asymmetric-utility system model: A framework for modeling cyber-human systems," in *Quantitative Evaluation of Systems (QEST), 2011 Eighth International Conference on.* IEEE, 2011, pp. 233–242.

[51] R. Sukkerd, D. Garlan, and R. Simmons, "Task planning of cyber-human systems," in *Software Engineering and Formal Methods.* Springer, 2015, pp. 293–309.

[52] A. Tellaeche *et al.*, "Use of machine vision in collaborative robotics: An industrial case." in *Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on.* IEEE, 2016, pp. 1–6.

[53] R. A. Kharjul, Tungar *et al.*, "Real-time pedestrian detection using svm and adaboost," in *Energy Systems and Applications, 2015 International Conference on.* IEEE, 2015, pp. 740–743.

[54] The Electric Power Research Institute (EPRI), "Video content analytics for nuclear process monitoring: A technology survey," *2015 Program 41.05.03 Instrumentation and Control Program*, p. 47, September 2015, accessed on: Jan. 10, 2020. [Online]. Available: http://www.epri.com

[55] P. F. Whelan *et al.*, *Machine vision algorithms in Java: techniques and implementation*. Springer Science & Business Media, 2012.

[56] E. R. Davies, *Machine vision: theory, algorithms, practicalities*. Elsevier, 2004.

[57] Y. Lin, Lv *et al.*, "Large-scale image classification: fast feature extraction and svm training," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 1689–1696.

[58] Y. LeCun, K. Kavukcuoglu, C. Farabet *et al.*, "Convolutional networks and applications in vision." in *ISCAS*, 2010, pp. 253–256.

[59] F. Lin, Lai *et al.*, "A traffic sign recognition method based on deep visual feature," in *Progress in Electromagnetic Research Symposium (PIERS)*. IEEE, 2016, pp. 2247–2250.

[60] M. Salarian *et al.*, "A vision based system for traffic lights recognition," in *SAI Intelligent Systems Conference (IntelliSys), 2015*. IEEE, 2015, pp. 747–753.

[61] R. Manoharan *et al.*, "Android opencv based effective driver fatigue and distraction monitoring system," in *Computing and Communications Technologies (ICCCT), 2015 International Conference on*. IEEE, 2015, pp. 262–266.

[62] Toyota. Lane departure alert (lda). Accessed on: Jan. 10, 2020. [Online]. Available: http://www.toyota-global.com/

[63] S. Sownthar and C. Manikandababu, "Visual monitoring and alerting system for arc type meters," *International Journal of Research in Computer Applications and Robotics*, vol. ISSN:2320-7345, pp. 29–41, 2014.

[64] D. S. Bhushan, "A review paper on automatic meter reading and instant billing," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 1, 2015.

[65] Y. Fujita and Y. Hamamoto, "Automatic reading of an analogue meter using image processing techniques," *IEEJ Transactions on Electronics, Information and Systems*, vol. 129, pp. 901–908, 2009.

[66] L. Zhao, Y. Zhang, Q. Bai *et al.*, "Design and research of digital meter identifier based on image and wireless communication," in *Industrial Mechatronics and Automation, 2009. ICIMA 2009. International Conference on.* IEEE, 2009, pp. 101–104.

[67] R. Ocampo-Vega, G. Sanchez-Ante *et al.*, "Image processing for automatic reading of electro-mechanical utility meters," in *Artificial Intelligence (MICAI), 2013 12th Mexican International Conference on.* IEEE, 2013, pp. 164–170.

[68] M. Gellaboina, G. Swaminathan, and V. Venkoparao, "Image based dial gauge reading," Mar. 21 2013, uS Patent App. 13/237,147. Accessed on: Jan. 10, 2020. [Online]. Available: http://www.google.com/patents/US20130070099

[69] J. Kennedy and T. Baird, "Analog utility meter reading," Sep. 25 2014, uS Patent App. 14/295,669. Accessed on: Jan. 10, 2020. [Online]. Available: http://www.google.com/patents/US20140286580

[70] L. Zhao, Y. Zhang, Q. Bai, Z. Qi, and X. Zhang, "Research of digital meter identifier based on dsp and neural network," in *Imaging Systems and Techniques, 2009. IST'09. IEEE International Workshop on.* IEEE, 2009, pp. 402–406.

[71] D. Shu, S. Ma, and C. Jing, "Study of the automatic reading of watt meter based on image processing technology," in *Industrial Electronics and Applications, 2007. ICIEA 2007. 2nd IEEE Conference on.* IEEE, 2007, pp. 2214–2217.

[72] Y. Tang, C.-W. Ten *et al.*, "Extraction of energy information from analog meters using image processing," *IEEE Transactions on Smart Grid*, vol. 6, no. 4, pp. 2032–2040, 2015.

[73] Z. Yang, W. Niu *et al.*, "An image-based intelligent system for pointer instrument reading," in *Information Science and Technology (ICIST), 2014 4th IEEE International Conference on.* IEEE, 2014, pp. 780–783.

[74] L. Xu *et al.*, "An automatic recognition method of pointer instrument based on improved hough transform," in *Applied Optics and Photonics China (AOPC2015)*, vol. 9675. International Society for Optics and Photonics, 2015, p. 96752T.

[75] D. M. Eler and R. E. Garcia, "Using otsu's threshold selection method for eliminating terms in vector space model computation," in *Information Visualisation (IV), 2013 17th International Conference.* IEEE, 2013, pp. 220–226.

[76] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on.* IEEE, 2011, pp. 2564–2571.

[77] Y.-S. Chen and J.-Y. Wang, "Computer vision-based approach for reading analog multimeter," *Applied Sciences*, vol. 8, no. 8, p. 1268, 2018.

[78] D. Mohr and G. Zachmann, "Continuous edge gradient-based template matching for articulated objects." in *VISAPP (2)*, 2009, pp. 519–524.

[79] Y. Li, "Pyramidal gradient matching for optical flow estimation," *arXiv preprint arXiv:1704.03217*, 2017.

[80] D. H. Widyantoro and K. I. Saputra, "Traffic lights detection and recognition based on color segmentation and circle hough transform," in *Data and Software Engineering (ICoDSE), 2015 International Conference on.* IEEE, 2015, pp. 237–240.

[81] E. C. Alegria and A. C. Serra, "Automatic calibration of analog and digital measuring instruments using computer vision," *IEEE transactions on instrumentation and measurement*, vol. 49, no. 1, pp. 94–99, 2000.

[82] G. Mahalakshmi, S. Sridevi, and S. Rajaram, "A survey on forecasting of time series data," in *2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16).* IEEE, 2016, pp. 1–8.

[83] H. V. Singh and Q. H. Mahmoud, "Evaluation of arima models for human–machine interface state sequence prediction," *Machine Learning and Knowledge Extraction*, vol. 1, no. 1, pp. 287–311, 2019.

[84] S. Budd, E. C. Robinson, and B. Kainz, "A survey on active learning and human-in-the-loop deep learning for medical image analysis," *arXiv preprint arXiv:1910.02923*, 2019.

[85] A. Holzinger, M. Plass, M. Kickmeier-Rust, K. Holzinger, G. C. Crişan, C.-M. Pintea, and V. Palade, "Interactive machine learning: experimental evidence for the human in the algorithmic loop," *Applied Intelligence*, vol. 49, no. 7, pp. 2401–2414, 2019.

[86] D. Xin, L. Ma, J. Liu, S. Macke, S. Song, and A. Parameswaran, "Accelerating human-in-the-loop machine learning: Challenges and opportunities," in *Proceedings of the Second Workshop on Data Management for End-To-End Machine Learning.* ACM, 2018, p. 9.

[87] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control.* John Wiley & Sons, 2015.

[88] Y. B. Wijaya, S. Kom, and T. A. Napitupulu, "Stock price prediction: Comparison of arima and artificial neural network methods-an indonesia stock's case," in *Advances*

in Computing, Control and Telecommunication Technologies (ACT), 2010 Second International Conference on. IEEE, 2010, pp. 176–179.

[89] Z. Tan, J. Zhang, J. Wang, and J. Xu, "Day-ahead electricity price forecasting using wavelet transform combined with arima and garch models," *Applied Energy*, vol. 87, no. 11, pp. 3606–3610, 2010.

[90] J. Bi, L. Zhang, H. Yuan, and M. Zhou, "Hybrid task prediction based on wavelet decomposition and arima model in cloud data center," in *Networking, Sensing and Control (ICNSC), 2018 IEEE 15th International Conference on.* IEEE, 2018, pp. 1–6.

[91] D. Janardhanan and E. Barrett, "Cpu workload forecasting of machines in data centers using lstm recurrent neural networks and arima models," in *Internet Technology and Secured Transactions (ICITST), 2017 12th International Conference for.* IEEE, 2017, pp. 55–60.

[92] A. Rath, S. Samantaray, K. S. Bhoi, and P. C. Swain, "Flow forecasting of hirakud reservoir with arima model," in *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS).* IEEE, 2017, pp. 2952–2960.

[93] Y. Cheung and K. Lai, "Lag order and critical values of the augmented dickey–fuller test," *Journal of Business & Economic Statistics*, vol. 13, no. 3, pp. 277–280, 1995.

[94] D. Yang, H. Chen, Y. Song, and Z. Gong, "Granger causality for multivariate time series classification," in *Big Knowledge (ICBK), 2017 IEEE International Conference on.* IEEE, 2017, pp. 103–110.

[95] Y. Tian, K. Zhang, J. Li, X. Lin, and B. Yang, "Lstm-based traffic flow prediction with missing data," *Neurocomputing*, vol. 318, pp. 297–305, 2018.

[96] S. Muzaffar and A. Afshari, "Short-term load forecasts using lstm networks," *Energy Procedia*, vol. 158, pp. 2922–2927, 2019.

[97] J. Cao, Z. Li, and J. Li, "Financial time series forecasting model based on ceemdan and lstm," *Physica A: Statistical Mechanics and its Applications*, vol. 519, pp. 127–139, 2019.

[98] S. Selvin, R. Vinayakumar, E. Gopalakrishnan, V. K. Menon, and K. Soman, "Stock price prediction using lstm, rnn and cnn-sliding window model," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI).* IEEE, 2017, pp. 1643–1647.

[99] H. Lu and F. Yang, "A network traffic prediction model based on wavelet transformation and lstm network," in *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, 2018, pp. 1–4.

[100] B. Sniderman, M. Mahto, and M. J. Cotteleer, "Industry 4.0 and manufacturing ecosystems: Exploring the world of connected enterprises," *Deloitte Consulting*, 2016.

[101] G. Erboz, "How to define industry 4.0: main pillars of industry 4.0," *Szent Istvan University, Gödöllő*, pp. 1–9, 2017.

[102] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.

[103] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[104] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[105] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[106] J. L. Elman, "Distributed representations, simple recurrent networks, and grammatical structure," *Machine learning*, vol. 7, no. 2-3, pp. 195–225, 1991.

[107] O. Levy, Y. Goldberg, and I. Dagan, "Improving distributional similarity with lessons learned from word embeddings," *Transactions of the Association for Computational Linguistics*, vol. 3, pp. 211–225, 2015.

[108] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[109] S. Takase and N. Okazaki, "Positional encoding to control output sequence length," *arXiv preprint arXiv:1904.07418*, 2019.

[110] X. Chen, L. Xu, Z. Liu, M. Sun, and H. Luan, "Joint learning of character and word embeddings," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[111] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 160–167.

[112] R. M. Iyer and M. Ostendorf, "Modeling long distance dependence in language: Topic mixtures versus dynamic cache models," *IEEE Transactions on speech and audio processing*, vol. 7, no. 1, pp. 30–39, 1999.

[113] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE transactions on acoustics, speech, and signal processing*, vol. 37, no. 3, pp. 328–339, 1989.

[114] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014.

[115] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[116] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[117] V. Mnih, N. Heess, A. Graves *et al.*, "Recurrent models of visual attention," in *Advances in neural information processing systems*, 2014, pp. 2204–2212.

[118] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[119] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, 2015, pp. 2048–2057.

[120] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[121] T. Shen, T. Zhou, G. Long, J. Jiang, S. Wang, and C. Zhang, "Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling," *arXiv preprint arXiv:1801.10296*, 2018.

[122] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," *arXiv preprint arXiv:1601.06733*, 2016.

[123] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, "Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned," *arXiv preprint arXiv:1905.09418*, 2019.

[124] Y. Fang, J. Gao, C. Huang, H. Peng, and R. Wu, "Self multi-head attention-based convolutional neural networks for fake news detection," *PloS one*, vol. 14, no. 9, 2019.

[125] A. Raganato, J. Tiedemann *et al.*, "An analysis of encoder representations in transformer-based machine translation," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics, 2018.

[126] L. Barrault, O. Bojar, M. R. Costa-jussà, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, P. Koehn, S. Malmasi *et al.*, "Findings of the 2019 conference on machine translation (wmt19)," in *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, 2019, pp. 1–61.

[127] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[128] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for squad," *arXiv preprint arXiv:1806.03822*, 2018.

[129] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from natural language inference data," *arXiv preprint arXiv:1705.02364*, 2017.

[130] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding with unsupervised learning," Technical report, OpenAI, Tech. Rep., 2018.

[131] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," *arXiv preprint arXiv:1707.03141*, 2017.

[132] Honneywell Video Systems. Intelligent analytics solutions(ias). Accessed on: Jan. 10, 2020. [Online]. Available: https://www.security.honeywell.com/me/product-repository/smart-impressions---honeywell-video-analytics

[133] ObjectVideo and Vivoteck. Vivotek releases video content analysis solution. Accessed on: Jan. 10, 2020. [Online]. Available: https://www.vivotek.com/en/news/press-releases/171

[134] H. V. Singh and Q. H. Mahmoud, "Vidaq: A computer vision based remote data acquisition system for reading multi-dial gauges," *Journal of Industrial Information Integration*, vol. 15, pp. 29–41, 2019.

[135] ——, "A lstm-based approach to monitor operator situation awareness via hmi," in *2019 IEEE International Conference on Industrial Internet (ICII)*. IEEE, 2019, pp. xx – yy.

[136] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM journal on control and optimization*, vol. 25, no. 1, pp. 206–230, 1987.

[137] X. Sun and Q. Chen, "Defects detecting of gloves based on machine vision," in *Real-time Computing and Robotics (RCAR), IEEE International Conference on*. IEEE, 2016, pp. 169–173.

[138] O. Vincent and O. Folorunso, "A descriptive algorithm for sobel image edge detection," in *Proceedings of Informing Science & IT Education Conference (InSITE)*, vol. 40, 2009, pp. 97–107.

[139] A. Amanatiadis and I. Andreadis, "Performance evaluation techniques for image scaling algorithms," in *Imaging Systems and Techniques, 2008. IST 2008. IEEE International Workshop on*. IEEE, 2008, pp. 114–118.

[140] J. Flusser and T. Suk, "Pattern recognition by affine moment invariants," *Pattern recognition*, vol. 26, no. 1, pp. 167–174, 1993.

[141] B. Catanzaro, B.-Y. Su *et al.*, "Efficient, high-quality image contour detection," in *Computer vision, 2009 IEEE 12th international conference*. IEEE, 2009, pp. 2381–2388.

[142] J. E. Hershberger and J. Snoeyink, *Speeding up the Douglas-Peucker line-simplification algorithm*. University of British Columbia, Department of Computer Science, 1992.

[143] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[144] B. Kosko, "Bidirectional associative memories," *IEEE Transactions on Systems, man, and Cybernetics*, vol. 18, no. 1, pp. 49–60, 1988.

[145] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[146] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2625–2634.

[147] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[148] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[149] Y. Liu, L. Ji, R. Huang, T. Ming, C. Gao, and J. Zhang, "An attention-gated convolutional neural network for sentence classification," *Intelligent Data Analysis*, vol. 23, no. 5, pp. 1091–1107, 2019.

[150] W. Yin, H. Schütze, B. Xiang, and B. Zhou, "Abcnn: Attention-based convolutional neural network for modeling sentence pairs," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 259–272, 2016.

[151] E. Culurciello, A. Zaidy, and V. Gokhale, "Computation and memory bandwidth in deep neural networks," *Medium*, vol. 5, pp. 1–4, 2017, accessed on: Jan. 10, 2020. [Online]. Available: https://medium.com/@culurciello/computation-and-memory-bandwidth-in-deep-neural-networks-16cbac63ebd5

[152] M. Honnibal, "Embed, encode, attend, predict: The new deep learning formula for state-ofthe-art nlp models," 2016, accessed on: Jan. 10, 2020. [Online]. Available: https://explosion.ai/blog/deep-learning-formula-nlp

[153] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[154] S. Wiseman and A. M. Rush, "Sequence-to-sequence learning as beam-search optimization," *arXiv preprint arXiv:1606.02960*, 2016.

[155] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2. Lille, 2015.

[156] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International Workshop on Similarity-Based Pattern Recognition*. Springer, 2015, pp. 84–92.

[157] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," in *Advances in neural information processing systems*, 2016, pp. 3630–3638.

[158] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[159] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 1171–1179.

[160] H. V. Singh and Q. H. Mahmoud, "Vidaq: A framework for monitoring human machine interfaces," in *Real-Time Distributed Computing (ISORC), 2017 IEEE 20th International Symposium on*. IEEE, 2017, pp. 141–149.

[161] K. Yeressian, "On overcoming the curse of dimensionality in neural networks," *arXiv preprint arXiv:1809.00368*, 2018.

[162] A. Zimek, E. Schubert, and H.-P. Kriegel, "A survey on unsupervised outlier detection in high-dimensional numerical data," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 5, no. 5, pp. 363–387, 2012.

[163] E. Irving, C. Miossec, and J. Tassart, "32a. towards efficient full automatic operation of the pwr steam generator with water level adaptive control," in *Boiler dynamics and control in nuclear power stations 2*. Thomas Telford Publishing, 1980, pp. 309–329.

[164] M. G. Na, "Auto-tuned pid controller using a model predictive control method for the steam generator water level," *IEEE Transactions on Nuclear Science*, vol. 48, no. 5, pp. 1664–1671, 2001.

[165] H. V. Singh and Q. H. Mahmoud, "Hmi-guard: A platform for detecting errors in human-machine interfaces," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2017, pp. 2861–2866.