

Model-based and Model-free Approaches for Power System Security Assessment

A Dissertation Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Mariana Magdy Mounir Kamel
August 2021

Acknowledgment

I would like to acknowledge the University of Tennessee Knoxville (UTK), Department of Electrical Engineering and Computer Science, for providing the opportunity to conduct my Ph.D. research through their excellent facilities and financial support. This research was made possible by the support of the sponsors of my research projects, the Center for Ultra-Wide-Area Resilient Electric Energy Transmission Networks (CURENT) and GEIRI North America.

I would like to express my deepest appreciation to my supervisor Dr. Fangxing (Fran) Li for his care and continuous support of my Ph.D. work here at the University of Tennessee. His mentoring, guidance, and encouragement have enabled me to be successful.

I am also very grateful to Dr. Abdelrahman Karrar for his remarkable guidance and for serving on my Ph.D. committee. His expertise and continuous support have encouraged me to achieve more than I could have imagined. Also, I would like to thank my committee members Dr. Yilu Liu and Dr. Hector Pulgar for their time and efforts through serving on this committee.

I would also like to thank all my friends who experienced the ups and downs of my research. Thank you for always being there for me.

Last but not the least; I would like to express my deepest gratitude to my family. This journey would not have been possible if not for their unconditional love and support.

Abstract

Continuous security assessment of a power system is necessary to insure a reliable, stable, and continuous supply of electrical power to customers. To this end, this dissertation identifies and explores some of the various challenges encountered in the field of power system security assessment. Accordingly, several model-based and/or model-free approaches were developed to overcome these challenges.

First, a voltage stability index, named TAVSI, is proposed. This index has three important features: TAVSI applies to general load models including ZIP, exponential, and induction motor loads; TAVSI can be used for both measurement-based and model-based voltage stability assessment; and finally, TAVSI is calculated based on normalized sensitivities which enables identification of weak buses and the definition of a global instability threshold. TAVSI was tested on both the IEEE 14-bus and the 181-bus WECC systems. Results show that TAVSI gives a reliable assessment of system stability.

Second, a data-driven and model-based hybrid reinforcement learning approach is proposed for training a control agent to re-dispatch generators' output power in order to relieve stressed branches. For large power systems, the agent's action space is highly dimensioned which challenges the successful training of data-driven agents. Therefore, we propose a hybrid approach where model-based actions are utilized to help the agent learn an optimal control policy. The proposed approach was tested and compared to the generic data-driven DDPG-based approach on the IEEE 118-bus system and a larger 2749-bus real-world system. Results show that the hybrid approach performs well for large power systems and that it is superior to the DDPG-based approach.

Finally, a Convolutional Neural Network (CNN) based approach is proposed as a faster alternative to the classical AC power flow-based contingency screening. The proposed approach is investigated on both the IEEE 118-bus system and the Texas 2000-bus synthetic system. For such large systems, the implementation of the proposed approach came with several challenges, such as computational burden, learning from imbalanced datasets, and performance evaluation of trained models. Accordingly, this work contributes a set of novel techniques and best practices that enables both efficient and successful implementation of CNN-based multi-contingency classifiers for large power systems.

Table of Contents

Chapter 1. Introduction.....	1
1.1 Overview	1
1.2 Problem Statement.....	1
1.3 Dissertation Outline	2
1.4 Contributions	4
Chapter 2. A Generalized Voltage Stability Indicator based on the Tangential Angles of PV and Load Curves considering Voltage-Dependent Load Models.....	5
2.1 Introduction	5
2.2 The Tangential Angles based Voltage Stability Index (TAVSI).....	8
2.3 TAVSI Calculation Considering Both Real and Reactive Power Increments.....	12
2.4 Generalization of TAVSI to Composite Loads	15
2.5 Generalization of TAVSI to an N-bus System	20
2.6 Simulation Studies	25
2.7 Conclusion	33
Chapter 3. Data-driven and Model-based Hybrid Reinforcement Learning to Reduce Stress on Power Systems Branches	36
3.1 Introduction	36
3.2 Reinforcement Learning and the Deep Deterministic Policy Gradient Algorithm	40
3.3 Data-Driven Reinforcement Learning and the Control Problem of Branch Over Load Relief.	43
3.4 Proposed Data-Driven and Model-based Hybrid Reinforcement Learning Approach ...	48
3.5 Simulation Studies.....	54
3.6 On the Training of 2749-bus HRL Agent.....	65
3.7 Conclusion	71
Chapter 4. Power Systems Contingency Screening Using Convolutional Neural Networks....	73
4.1 Introduction	73
4.2 CNN-based Contingency Screening.....	74
4.3 Training and Test Datasets	83
4.4 Implementation in Tensorflow	91

4.5	Proposed CNN Structure	97
4.6	Training and Performance Evaluation: Approaches.....	102
4.7	Training and Performance Evaluation: Results	108
4.8	Conclusion	148
Chapter 5.	Conclusion	150
5.1	Conclusion and Contributions	150
5.2	Future Work.....	151
References		153
Appendix		158
A.1	Standard Convolution	158
A.2	Sparse Convolution: Proposed Method	163
Vita		170

List of Tables

Table 2-1 Induction motor parameters in p.u. (100 MVA base)	19
Table 2-2 IEEE 14-bus system rankings, taken at TAVSI = 40.	24
Table 2-3 Induction motor parameters in p.u. (100 MVA base) [17].....	26
Table 3-1 Hybrid data-driven and model-based training algorithm	52
Table 3-2 IEEE 118-bus system: analysis of agents' performance when failed to secure the system.....	63
Table 3-3 2749-bus system: analysis of agents' performance when failed to secure the system..	64
Table 4-1 IEEE 118-bus system: statistics of the generated dataset.....	88
Table 4-2 Texas 2000-bus systems: statistics of the generated dataset	92
Table 4-3 CNN Structure for IEEE 118-bus system multi-contingency classifier.	100
Table 4-4 CNN Structure for 2000-bus synthetic system multi-contingency classifier.	101
Table 4-5 IEEE 118-bus system: macro-recall and macro-precision	111
Table 4-6 IEEE 118-bus system: # of contingencies with undefined recall and/or undefined precision.	111
Table 4-7 Texas 2000-bus synthetic system: macro-recall and macro-precision	131
Table 4-8 Texas 2000-bus synthetic system: # of contingencies with undefined recall and/or undefined precision.	131

List of Figures

Figure 2-1 Two-bus system	9
Figure 2-2 P-V nose curve and load characteristics' curves of a two-bus system under three load models.	11
Figure 2-3 Two-bus system with composite load at bus 2.....	17
Figure 2-4 Two-bus system with composite load at bus 2: power flow model	17
Figure 2-5 Bus 2 P-V curve: static CPF vs time-domain simulation.....	19
Figure 2-6 IEEE 14-bus system.	26
Figure 2-7 IEEE 14-bus system: voltage of load buses	27
Figure 2-8 IEEE 14-bus system: TAVSI of load buses	27
Figure 2-9 IEEE 14-bus system: motor loads' active and reactive power demand	28
Figure 2-10 IEEE 14-bus system: voltage of load buses	31
Figure 2-11 IEEE 14-bus system: TAVSI of load buses	31
Figure 2-12 IEEE 14-bus system: motor loads' active and reactive power demand	32
Figure 2-13 An illustrative diagram of the CURENT WECC test system.	32
Figure 2-14 181-bus WECC system: voltage of the 9 weakest load buses	34
Figure 2-15 181-bus WECC system: TAVSI of the 9 weakest load buses	34
Figure 3-1 IEEE 118-bus system: agents' training performance.....	58
Figure 3-2 2749-bus system: agents' training performance.	60
Figure 3-3 IEEE 118-bus system: DDPG agent's performance testing results.	60
Figure 3-4 IEEE 118-bus system: HRL agent's performance testing results.	61
Figure 3-5 2749-bus system: HRL agent's performance testing results.....	63
Figure 3-6 2749-bus system: HRL agent's training performance.	70
Figure 3-7 2749-bus system: HRL agent's training performance.	70
Figure 4-1 General structure of a CNN-based multi-contingency classifier.	77
Figure 4-2 IEEE 118-bus system: imbalance of label dataset.	88
Figure 4-3 IEEE 118-bus system: imbalance of label training dataset.	89
Figure 4-4 IEEE 118-bus system: imbalance of label test dataset.....	89
Figure 4-5 Texas 2000-bus synthetic system: imbalance of label dataset.	92
Figure 4-6 Texas 2000-bus synthetic system: imbalance of training dataset.	93
Figure 4-7 Texas 2000-bus synthetic system: imbalance of test dataset.	93
Figure 4-8 Basic implementation of sparse convolution using full arrays.	94
Figure 4-9 IEEE 118-bus system: Adjacency matrices.	97
Figure 4-10 Texas 2000-bus synthetic system: Adjacency matrices.	97
Figure 4-11 IEEE 118-bus system: recall-precision graph for CNN-MSE model, class insecure.	112
Figure 4-12 IEEE 118-bus system: recall-precision graph for CNN-WMSE model, class insecure.....	113

Figure 4-13 IEEE 118-bus system: recall-precision graph for CNN-MOL model, class insecure.	114
Figure 4-14 IEEE 118-bus system: recall performance comparison graph for class insecure. ..	115
Figure 4-15 IEEE 118-bus system: precision performance comparison graph for class insecure.	116
Figure 4-16 IEEE 118-bus system: recall-precision graph for CNN-MSE model, class marginal-insecure.....	117
Figure 4-17 IEEE 118-bus system: recall-precision graph for CNN-WMSE model, class marginal-insecure.	118
Figure 4-18 IEEE 118-bus system: recall-precision graph for CNN-MOL model, class marginal-insecure.....	119
Figure 4-19 IEEE 118-bus system: recall performance comparison graph for class marginal-insecure.....	120
Figure 4-20 IEEE 118-bus system: precision performance comparison graph for class marginal-insecure.....	121
Figure 4-21 IEEE 118-bus system: recall-precision graph for CNN-MSE model, class marginal-insecure.....	122
Figure 4-22 IEEE 118-bus system: recall-precision graph for CNN-WMSE model, class marginal-insecure.	123
Figure 4-23 IEEE 118-bus system: recall-precision graph for CNN-MOL model, class marginal-insecure.....	124
Figure 4-24 IEEE 118-bus system: recall performance comparison graph for class secure.	125
Figure 4-25 IEEE 118-bus system: precision performance comparison graph for class secure.	126
Figure 4-26 Texas 2000-bus synthetic system: recall-precision graph for CNN-MSE model, class insecure.....	132
Figure 4-27 Texas 2000-bus synthetic system: recall-precision graph for CNN-WMSE model, class insecure.....	133
Figure 4-28 Texas 2000-bus synthetic system: recall-precision graph for CNN-MOL model, class insecure.....	134
Figure 4-29 Texas 2000-bus synthetic system: recall performance comparison graph for class insecure.....	135
Figure 4-30 Texas 2000-bus synthetic system: precision performance comparison graph for class insecure. CNN-MSE is chosen as the base-model for comparison.....	136
Figure 4-31 Texas 2000-bus synthetic system: recall-precision graph for CNN-MSE model, class marginal-insecure.	137
Figure 4-32 Texas 2000-bus synthetic system: recall-precision graph for CNN-WMSE model, class marginal-insecure.	138
Figure 4-33 Texas 2000-bus synthetic system: recall-precision graph for CNN-MOL model, class marginal-insecure.	139

Figure 4-34 Texas 2000-bus synthetic system: recall performance comparison graph for class marginal-insecure.	140
Figure 4-35 Texas 2000-bus synthetic system: precision performance comparison graph for class marginal-insecure.	141
Figure 4-36 Texas 2000-bus synthetic system: recall-precision graph for CNN-MSE model, class marginal-insecure.	142
Figure 4-37 Texas 2000-bus synthetic system: recall-precision graph for CNN-WMSE model, class marginal-insecure.	143
Figure 4-38 Texas 2000-bus synthetic system: recall-precision graph for CNN-MOL model, class marginal-insecure.	144
Figure 4-39 Texas 2000-bus synthetic system: recall performance comparison graph for class secure.	145
Figure 4-40 Texas 2000-bus synthetic system: precision performance comparison graph for class secure.	146

Chapter 1. Introduction

1.1 Overview

Power systems are designed and operated with the objective of having a reliable, stable, and continuous supply of electrical power to customers. However, there always exists a possibility of unpredictable faults which might lead to degrading the voltage security of the system and/or initiating a cascade of branch outages. Both can easily evolve and end up causing interruption of energy supply across all or major parts of the system i.e. a blackout. Therefore, continuous assessment of system security is necessary to predict and accordingly, prevent the system from collapsing.

1.2 Problem Statement

This work considers three different problems yet, all are related to the security of a power system. Those are:

- **Voltage stability assessment of power systems:** voltage instability and voltage collapse are identified as the main reason behind several major blackouts worldwide. Several voltage stability assessment methods have been proposed in the literature in an attempt to assess proximity to voltage collapse and accordingly, define a threshold for taking corrective measures. Load characteristics are a major factor that greatly influences the accuracy and reliability of a proposed assessment scheme. Typically, a constant PQ model is assumed for the system load. This is acceptable within the power system community because it leads to more conservative margins when compared to other voltage-dependent loads, i.e., ZIP and exponential load models. However, if the system actual load includes a

large percentage of induction motors then such an assumption will lead to highly pessimistic and unreliable margins.

- **Overload relief of power systems branches:** overloaded branches, i.e. transmission lines and transformers, have been responsible (in part or fully) for several major blackouts worldwide: the US-Canadian grid 2003, Italy 2003, and India 2012 [1]. An overloaded branch might trip out on overload or zone-3 distance protection. The overloaded branch might also sag into a tree and trip out on fault. This, in turn, might cause additional branches to become overloaded and also trip out. To prevent such a cascade of outages, the initially overload branch(es) must be secured as quickly as possible.
- **N-1 contingency screening:** static N-1 contingency screening is a vital tool for assessing the security of power systems. It identifies the set of component outages, i.e., contingencies which would result in an insecure operation of the system, and therefore, power system planners and operators can prepare and plan ahead of time against such potential risks. Performing static contingency analysis is typically computationally expensive. In addition, with the increasing levels of penetration of intermittent renewables, e.g., wind and solar it becomes necessary to run N-1 contingency screening for a number of different generation scenarios. Such computational burden might prevent the application of static contingency analysis in an online type frame of work.

1.3 Dissertation Outline

Chapter 2 presents a generalized voltage stability index, named TAVSI, which accounts for the voltage dependencies of ZIP, exponential, and induction motor loads. First, the chapter presents the concepts and mathematical formulation of the TAVSI and how it can be applied using either a

model-based or a measurement-based approach. The chapter proceeds with simulation studies along with a discussion on the performance of this index.

Chapter 3 presents a hybrid data-driven and model-based reinforcement learning (RL) approach to tackle the control problem of branch overload relief in large power systems. First, the chapter presents a series of investigations on the degraded behavior of data-driven RL agents when trained for such a control task on “large” power systems. Then, the chapter proceeds with a proposal of a hybrid data-driven and model-based approach which improves the training of the RL agent by combining data-driven RL agent actions with generator shifting factor-driven actions. Next, the chapter presents a set of simulation studies and discussions which show a promising performance of the proposed method.

Chapter 4 proposes to utilize convolutional neural networks (CNN) for performing the task of power systems contingency screening. First, the chapter explores the potential for utilizing CNN for applications related to power systems. Then, it proposes a CNN-based multi-label classification approach for the simultaneous evaluation of a pre-selected set of N-1 contingencies. The chapter then explores the various elements involved in the implementation of the proposed approach. Finally, the chapter presents a set of simulation studies and investigations on the performance of the proposed approach.

Chapter 5 concludes the dissertation.

1.4 Contributions

The contributions of this work are listed as follow:

- This work proposes a generalized voltage stability indicator based on the tangential angles of PV and load curves considering voltage-dependent load models
- This work proposes a data-driven and model-based hybrid reinforcement learning approach to reduce stress on power systems branches.
- This work proposes a convolutional neural network-based approach as a faster alternative to the regular AC power flow-based contingency screening.

Chapter 2. A Generalized Voltage Stability Indicator based on the Tangential Angles of PV and Load Curves considering Voltage-Dependent Load Models

This chapter presents a generalized voltage stability index which has two important features. First, it is applicable to general load models including ZIP, exponential, and induction motor loads. Second, it can be used for both measurement-based and model-based voltage stability assessment. The key idea of this index is that at the collapsing point, the system P-V curve and the load P-V curve share the same tangent line. Thus, the tangential angle between the two curves is an appropriate indicator of proximity to voltage collapse when voltage-dependent loads like ZIP and motor are considered. The performance of the proposed index is tested on both the IEEE 14-bus system and the 181-bus WECC system. The results show that the proposed index gives a good and reliable assessment of system stability. In addition, it demonstrated a consistent behavior, always reaching a value of 0.0 at the stability limit which enables the definition of a threshold for taking corrective measures.

2.1 Introduction

Voltage instability and voltage collapse are identified as the main reason behind several major blackouts worldwide. The severity of these blackout events has prompted significant research efforts in the area of monitoring and control of system voltage stability. Several voltage stability assessment methods have been proposed in the literature in an attempt to assess proximity to voltage collapse. Different concepts and analysis techniques have been utilized in the literature of voltage stability assessment. Those include repeated power flows [2]; numerical continuation [3, 4]; modal analysis [5, 6]; optimization [7, 8]; Thevenin equivalent and the concept of maximum

power transfer [9, 10]; and sensitivities of system states to changes in system loading [11-13]. Some of the major issues and technical challenges related to the proposed voltage stability assessment techniques are theoretical soundness; computational complexity; the ability to quantify “proximity” to collapse and therefore, define a threshold for taking corrective measures; the accuracy of predicted performance and stability margins. In regards to the latter issue, the load model is a major factor that greatly influences the accuracy and reliability of a proposed assessment scheme. The load model and its dependency on bus voltage are typically ignored and a constant PQ model is assumed for the system load. Assuming a constant PQ load is acceptable within the power system community because it leads to more conservative margins when compared to other voltage-dependent loads, i.e., ZIP and exponential load models [14]. However, if the system actual load includes a large percentage of induction motors then such an assumption will lead to highly pessimistic and unreliable margins [15].

In the literature, several works have extended and/or modified some of the existing voltage stability assessment methods to account for induction motor loads [16-18]. Likewise, driven by the need for a more reliable voltage stability assessment, this work explores the application of a voltage stability index, recently proposed in [19], to systems with induction motor loads. This index (hereinafter referred to as the Tangential Angle based Voltage Stability Index (TAVSI)) is calculated based on the tangential angles of the system P-V curve and the load P-V curve. In [19], the authors considered the load to have either ZIP or exponential load characteristics. In this work, the formulation of the TAVSI is modified and generalized to account not only for ZIP or exponential loads but for induction motor loads as well.

In addition, the authors in [19] provided the calculation of the tangential angles based on the incremental change of only the load's real power. In this work, the formula for calculating the tangential angles, and therefore the TAVSI, is modified to include the impact from reactive power load increments as well. Moreover, in [19], the authors adopted a measurement-based approach in which the angle of the load bus P-V curve is approximated based on a two-bus Thevenin equivalent (TE) of the system at that load bus. Typically, the parameters of the TE are tracked based on two (or more) consecutive and synchronized measurement sets of voltage and current phasors [19-22]. However, in practice, there exist three major issues that might render a given TE identification method inapplicable. Those are measurement noise; changes in the operating conditions on the system side during a given measurement period, e.g., the operation of generators' over-excitation limiters; and phase angle drift caused by changes in system frequency. Due to these limitations, an alternative model-based approach is proposed for calculating the value of TAVSI. Whether a measurement-based or a model-based approach is chosen depends on the applicability of the method in the considered system.

The remainder of this chapter is organized as follows: Section 2.2 presents the concepts and mathematical formulation of the TAVSI as proposed in [19]. In Section 2.3, the formula for calculating the TAVSI is modified to include the impact of the load's reactive power increment. In Section 2.4, the application of the TAVSI is generalized for composite load models. In Section 0, the TAVSI is generalized for any given N bus system. Section 2.6 presents simulation results along with a discussion on the performance of TAVSI. Section 2.7 concludes the chapter.

2.2 The Tangential Angles based Voltage Stability Index (TAVSI)

This section presents the concepts behind the TAVSI and its mathematical formulation as proposed in [19] is presented.

2.2.1 Review of TAVSI

A simple two-bus system, shown in Figure 2-1, is utilized to present the concept of the TAVSI. The system consists of a slack bus connected to a load bus through a transmission line. E is the voltage magnitude of the slack bus (bus 1); R and X are the transmission line resistance and reactance respectively; V and δ are the voltage magnitude and angle of the load bus (bus 2); P_L and Q_L are the active and reactive power consumption of the load.

The load at bus 2 is assumed to consist of either a ZIP or an exponential load. The power-voltage relations of those two load models are detailed in (1) and (2).

$$P_{ZIP} = P_0(aV^2 + bV + c) \quad (1a)$$

$$Q_{ZIP} = Q_0(dV^2 + eV + f) \quad (1b)$$

$$P_{EXP} = P_0V^{\alpha_P} \quad (2a)$$

$$Q_{EXP} = Q_0V^{\alpha_Q} \quad (2b)$$

P_0 and Q_0 are the load real and reactive power at $V = 1$ p.u. a , b , c , d , e , and f are the ZIP load parameters where $a + b + c = 1$ and $d + e + f = 1$. α_P and α_Q are the exponential load parameters. The system P-V curve, also known in the literature as the nose curve, represents the sequence of solutions of the nonlinear power flow equations when the system load is uniformly increased. This P-V curve is typically obtained using continuation power flow (CPF) [1]. In CPF, a parameter λ , known as the loading parameter, is added to the power flow problem formulation. This λ controls

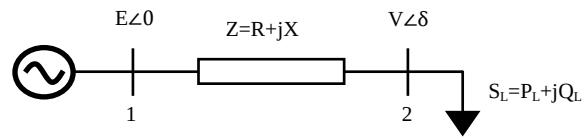


Figure 2-1 Two-bus system

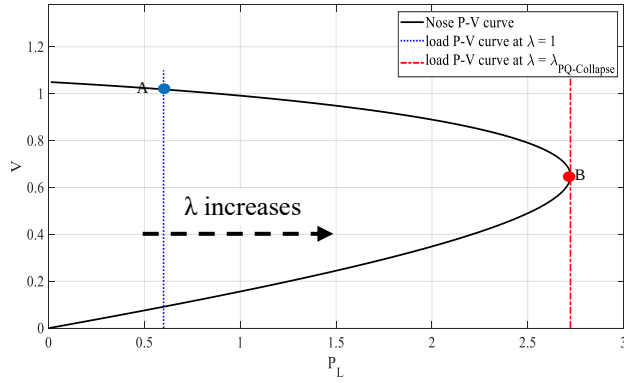
the increase in system loading. That is, load power becomes a function of both V and λ . For instance, if the load at bus 2 has an exponential load model, then $P_L = \lambda P_0 V^{\alpha_P}$ and $Q_L = \lambda Q_0 V^{\alpha_Q}$. At each increment of λ , the system equations are solved and the next point on the curve is obtained.

It should be noted that the three nose curves in Figure 2-2 are not of the same shape because the corresponding Q will be subject to different load models, which will lead to the PQV 3-D surface [20] projected to different PV curves. On the other hand, the load P-V curves represent the projection of the relation $P_L = \lambda P_0 V^{\alpha_P}$ on the P-V plane where each dashed curve corresponds to a distinct loading level, $\lambda = [\lambda_1, \lambda_2, \dots]$.

Now, looking at the P-V plane for load bus 2, at a given loading level $\lambda = \lambda_k$, the solution for the bus voltage V and load power P_L is the point where the two curves intersect (i.e., the P-V nose curve and the load P-V curve). If there exists no intersection between the two curves, then the system collapses. This is shown in Figure 2-2 a, b, and c for constant PQ, ZIP, and exponential load models respectively. Point A represents a stable operating point while point B represents the point of system collapse. It is interesting to note that at the collapse point, the load characteristic curve becomes tangential to the system P-V curve. Therefore, the collapse point can be characterized as the point where the system P-V curve and the load curve share the same tangent line. In mathematical terms:

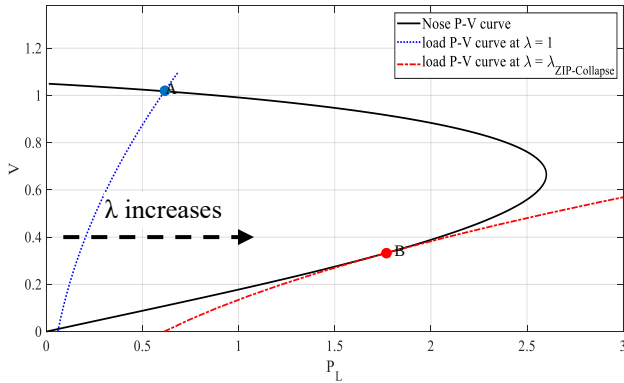
$$\left. \frac{dP_L}{dV} \right|_{Nose\ Curve} = \left. \frac{dP_L}{dV} \right|_{Load\ Curve} \quad (3)$$

Based on the above concept, the authors in [19] proposed to utilize the angle between: 1) the tangent to the P-V nose curve and 2) the tangent to the load P-V characteristic curve as an indicator of system stability. Equation (4), which is the concluding equation in [19], details the mathematical formulation of the proposed index.



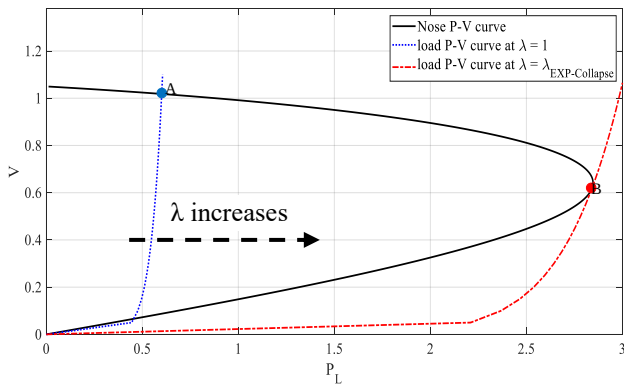
(a) P-V nose curve and PQ load

P-V curves



(b) P-V nose curve and ZIP load

P-V curves



(c) P-V nose curve and
exponential load P-V Curves

Figure 2-2 P-V nose curve and load characteristics' curves of a two-bus system under three load models.

$$TAVSI = \tan^{-1} \left(\frac{dP_L}{dV} \Big|_{Load\ Curve} \right) - \tan^{-1} \left(\frac{dP_L}{dV} \Big|_{Nose\ Curve} \right) \quad (4)$$

The value of the proposed TAVSI varies from 180° (i.e., π) at zero loading to zero at the point of system collapse. Therefore, values “close” to zero would indicate voltage instability. A threshold for raising an alarm or for taking corrective actions can be set to a value within the range of 40° to 20° . However, it is better recommended to first perform off-line studies and to examine the performance of the TAVSI under different operating conditions and scenarios (e.g., network topology, base-case loading, limits on controls ... etc.) Then, one may decide on the appropriate value for the alarm threshold for the system under consideration.

2.3 TAVSI Calculation Considering Both Real and Reactive Power

Increments

To evaluate the value of TAVSI for load bus 2 at any given operating point, the slopes of both the P-V nose curve and the load P-V curve need to be calculated. In [19], while deriving the formula for calculating the slope of the P-V nose curve, the authors assumed an incremental change of only the load’s real power. This work further extends the calculation of the P-V nose curve slope by considering the incremental change of both the real and reactive power of the load. In other words, the impact of reactive power load increments will be included. The following details the derivation of the formulas used for calculating the slope of the P-V curve, the slope of the load P-V curve, and accordingly the TAVSI.

2.3.1 Slope of the P-V Nose Curve: $dP_L/dV|_{Nose\ Curve}$

The formula for calculating the slope of the P-V nose curve at a given operating point is derived as follows: given the two-bus system of Figure 2-2, then, at any loading level λ , the following relations hold true:

$$V^4 + V^2(2RP_L + 2XQ_L - E^2) + Z^2S_L^2 = 0 \quad (5)$$

Where: if the load at bus 2 had a ZIP load model then,

$$P_L = \lambda P_0(aV^2 + bV + c) \quad (6a)$$

$$Q_L = \lambda Q_0(dV^2 + eV + f) \quad (6b)$$

Or, if the load at bus 2 had an exponential model then,

$$P_L = \lambda P_0 V^{\alpha_P} \quad (7a)$$

$$Q_L = \lambda Q_0 V^{\alpha_Q} \quad (7b)$$

Now, going back to how the system's P-V curves are obtained, according to [3], the system load is increased uniformly from a given base-case loading profile using the loading parameter λ . At each incremental change of λ , the system power flow equations are solved and a new point on the curve is obtained. For a given incremental change in λ , the changes of P_L , Q_L and V at load bus 2 can be related to one another according to the load's characteristic equations ((6) or (7)) as follows:

$$\Delta P_L = \frac{\partial P_L}{\partial V} \Delta V + \frac{\partial P_L}{\partial \lambda} \Delta \lambda \quad (8)$$

$$\Delta Q_L = \frac{\partial Q_L}{\partial V} \Delta V + \frac{\partial Q_L}{\partial \lambda} \Delta \lambda \quad (9)$$

Where $\partial P_L / \partial V$, $\partial P_L / \partial \lambda$, $\partial Q_L / \partial V$, and $\partial Q_L / \partial \lambda$ are the partial derivatives of the load real and reactive power functions ((6) or (7)) with respect to V and λ . In the limiting case where $\Delta \lambda \rightarrow 0$, then:

$$\frac{dP_L}{d\lambda} = \frac{\partial P_L}{\partial V} \times \frac{dV}{d\lambda} + \frac{\partial P_L}{\partial \lambda} \quad (10)$$

$$\frac{dQ_L}{d\lambda} = \frac{\partial Q_L}{\partial V} \times \frac{dV}{d\lambda} + \frac{\partial Q_L}{\partial \lambda} \quad (11)$$

Now, since both P_L and V are functions of λ , then, the slope of the P-V curve ($dP_L/dV|_{\text{Nose Curve}}$) can be expressed in terms of $dP_L/d\lambda$ and $dV/d\lambda$ as follows:

$$\left. \frac{dP_L}{dV} \right|_{Nose Curve} = \frac{dP_L}{d\lambda} \times \left(\frac{dV}{d\lambda} \right)^{-1} \quad (12)$$

Substituting for $dP_L/d\lambda$ from (10) into (12) results in:

$$\left. \frac{dP_L}{dV} \right|_{Nose Curve} = \frac{\partial P_L}{\partial V} + \frac{\partial P_L}{\partial \lambda} \times \left(\frac{dV}{d\lambda} \right)^{-1} \quad (13)$$

The slope term $dV/d\lambda$ in the above equation can be evaluated based on the circuit equation of (5). It should be noted that P_L , Q_L , and V are all functions of λ then, differentiating (5) with respect to λ results in:

$$\frac{dV}{d\lambda} = \frac{(P_L Z^2 + R V^2) \frac{dP_L}{d\lambda} + (Q_L Z^2 + X V^2) \frac{dQ_L}{d\lambda}}{V E^2 - 2V(R P_L + X Q_L + V^2)} \quad (14)$$

Substituting for $dP_L/d\lambda$ and $dQ_L/d\lambda$ from (10) and (11) into (14) results in:

$$\frac{dV}{d\lambda} = \frac{(P_L Z^2 + R V^2) \frac{\partial P_L}{\partial \lambda} + (Q_L Z^2 + X V^2) \frac{\partial Q_L}{\partial \lambda}}{V E^2 - 2V(R P_L + X Q_L + V^2) - (P_L Z^2 + R V^2) \frac{\partial P_L}{\partial V} - (Q_L Z^2 + X V^2) \frac{\partial Q_L}{\partial V}} \quad (15)$$

Substituting for $dV/d\lambda$ from (15) into (13) results in the following formula for calculating the slope of the nose curve:

$$\left. \frac{dP_L}{dV} \right|_{Nose Curve} = \frac{\partial P_L}{\partial V} + \frac{\partial P_L}{\partial \lambda} \left(\frac{V E^2 - 2V(R P_L + X Q_L + V^2) - (P_L Z^2 + R V^2) \frac{\partial P_L}{\partial V} - (Q_L Z^2 + X V^2) \frac{\partial Q_L}{\partial V}}{(P_L Z^2 + R V^2) \frac{\partial P_L}{\partial \lambda} + (Q_L Z^2 + X V^2) \frac{\partial Q_L}{\partial \lambda}} \right) \quad (16)$$

2.3.2 Slope of the Load P-V Curve: $dP_L/dV|_{Load Curve}$

As mentioned before, the load P-V curve represents the relation of P_L and V at a fixed loading level (λ is constant). Therefore, based on (6) and (7), $dP_L/dV|_{Load Curve}$ is calculated as given in (17) for ZIP loads or (18) for exponential ones.

$$\left. \frac{dP_L}{dV} \right|_{Load\ Curve} = \left(\frac{2aV + b}{aV^2 + bV + c} \right) P_L \quad (17)$$

$$\left. \frac{dP_L}{dV} \right|_{Load\ Curve} = \left(\frac{\alpha_P}{V} \right) P_L \quad (18)$$

Therefore, for a two-bus system, in the case of a ZIP or an exponential load is connected to bus 2, the TAVSI is calculated using (4); (16); and (17) or (18).

2.4 Generalization of TAVSI to Composite Loads

A composite load model consists of a ZIP (or exponential) load and an induction motor load. According to the studies in [21, 22], a composite load model proves to provide a more accurate representation of real-world system load behavior. In this work, induction motor dynamics are ignored and a static model is assumed for representing the motor operation. This assumption is valid when assessing the long-term voltage stability of the system [23]. Another assumption made in this work is that the mechanical load torque of the induction motor is independent of motor speed, i.e., constant torque load. This assumption leads to a more conservative stability assessment when compared to speed-dependent load torque characteristics [16]. Figure 2-3 shows the two-bus system when a composite load is connected to bus 2. Here, P_L and Q_L represent the power consumption of the ZIP (or exponential) part of the composite load. The induction motor load is represented by its static model, i.e., its steady-state equivalent circuit model where R_s and X_s represent the stator winding resistance and reactance; R_r and X_r represent rotor winding resistance and reactance; and X_m represents the magnetizing reactance. P_{sh} is the power transferred to the motor shaft [24]. The system of Figure 2-3 can be modified into the one shown in Figure 2-4 by transforming the wye impedance configuration between nodes 2, 2', and ground into its equivalent delta representation where:

$$Z_{22'} = \frac{Z_s Z_r + Z_s Z_m + Z_r Z_m}{Z_m} \quad (19a)$$

$$Z_{20} = \frac{Z_s Z_r + Z_s Z_m + Z_r Z_m}{Z_r} \quad (19b)$$

$$Z_{2'0} = \frac{Z_s Z_r + Z_s Z_m + Z_r Z_m}{Z_s} \quad (19c)$$

Assuming a constant torque load model then, in steady-state, P_{sh} can be calculated in terms of slip s and mechanical torque T_m as follows [24]:

$$P_{sh} = \frac{2\omega_s(1-s)T_m}{3p} \quad (20)$$

Where $\omega_s = 2\pi f_s$ is the synchronous speed, p is the number of poles. Now, given a base case where $P_{sh} = P_{sh0}$, $s = s_0$, and $T_m = T_{m0}$; then, from (20), the shaft power P_{sh} at any given mechanical load $T_m = \lambda T_{m0}$ can be calculated in terms of base-case P_{sh0} as follows:

$$P_{sh} = \lambda P_{sh0} \left(\frac{1-s}{1-s_0} \right) \quad (21)$$

The ratio $(1-s)/(1-s_0)$ is typically close to 1. Therefore, P_{sh} can be assumed to be equal to λP_{sh0} . Then, the motor load can be represented as a constant P load behind a transmission line. This model representation can easily be included in the power flow problem formulation. This makes it possible to utilize CPF technique for proper and reliable voltage stability assessment of systems with induction motor loads. In other words, the obtained P-V curve at bus 2 will reflect a true and accurate behavior of the system when the load at bus 2 contains an induction motor load,

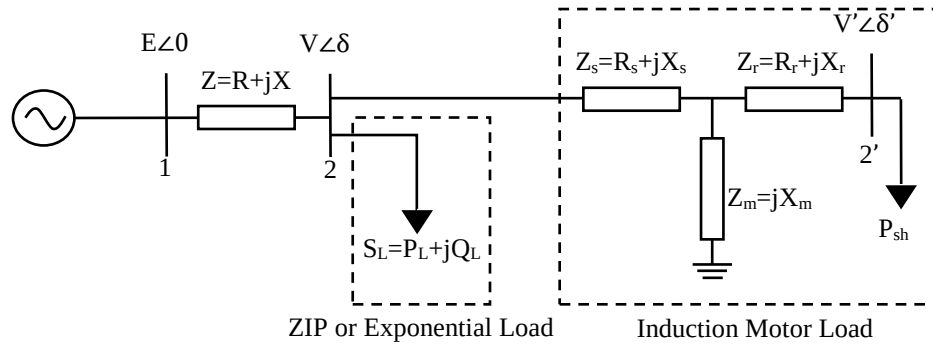


Figure 2-3 Two-bus system with composite load at bus 2

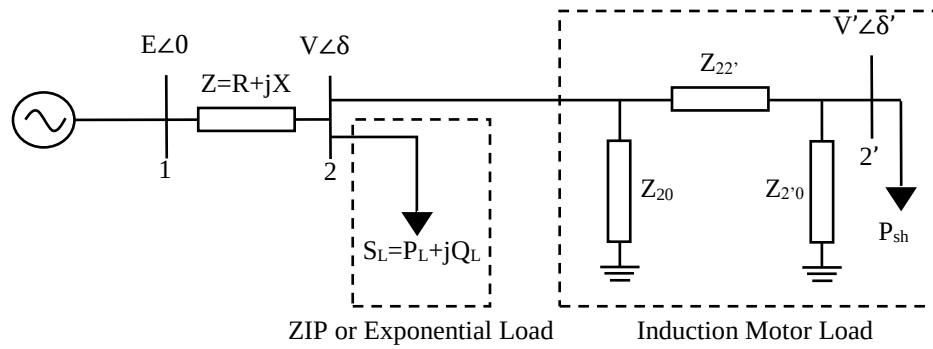


Figure 2-4 Two-bus system with composite load at bus 2: power flow model

which is usually a large, aggregated motor load model. To verify this conclusion, the P-V curve obtained for bus 2 using static CPF (performed on the equivalent three-bus system of Figure 2-4) is compared to that obtained using dynamic simulation based on Simulink. In Simulink, the induction motor is represented by a set of differential equations that models both the steady-state and transient responses of the motor. The studied system has the following parameters: $E = 1.05$ p.u.; $Z = 0.01 + j0.15$ p.u.; the load at bus 2 consists of a constant PQ load and an induction motor. For the constant PQ load, $P_{L2} = 0.25$ p.u. and $Q_{L2} = 0.15$ p.u. The induction motor parameters are given in Table 2-1.

Figure 2-5 shows two P-V curves for bus 2. The red curve is obtained through static CPF analysis. The blue curve is obtained through dynamic simulations using Simulink. In Simulink, the loading parameter is used on the base-case load torque (i.e., $T_m = \lambda T_{m0}$). For small load torque increments, the transient response quickly dies out and the motor stabilizes at a new operating point (steady-state response). At any steady-state operating point, the shaft power, motor speed, and load torque are governed by the relation of (20). On the other hand, in CPF the loading parameter is used on the base-case shaft power (i.e., $P_{2'} = \lambda P_{sh0}$). As seen from the figure, the two curves are almost an exact match to one another. They slightly differ near the point of system collapse. The static method of CPF yielded a lower point of collapse at $\lambda = 1.58$, i.e., a more conservative assessment. This is because in CPF, P_{sh} is assumed to be equal to λP_{sh0} . However, at heavy loading (near system collapse), the change in motor speed becomes more pronounced and the actual power delivered to the motor shaft P_{sh} would be less than its assumed value of $P_{sh} = \lambda P_{sh0}$. In other words, for the same mechanical load, the value of P_{sh} considered in CPF analysis is higher than its true value. Therefore, using CPF would lead to a more conservative assessment, i.e., the system is predicted to collapse at lower values of λ , i.e., lower mechanical load.

Table 2-1 Induction motor parameters in p.u. (100 MVA base)

Rs	Xs	Rr	Xr	Xm	H (sec)	P (base-case)
0.01	0.145	0.008	0.145	3.3	0.6	0.6

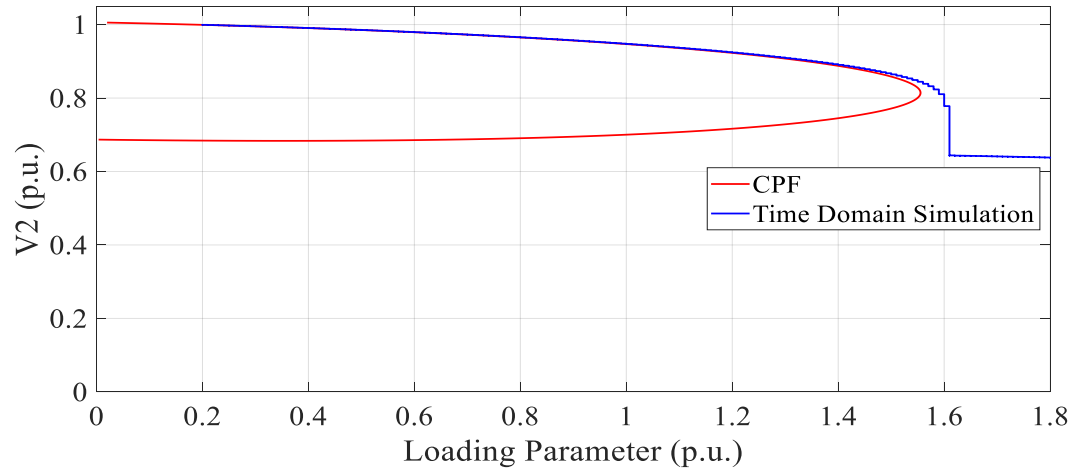


Figure 2-5 Bus 2 P-V curve: static CPF vs time-domain simulation

Coming back to TAVSI calculations, the index is calculated using Equations (4), (13), and (17) or (18). Here, the slope term $dV/d\lambda$ in (13) would be evaluated based on the circuit of Figure 2-4. The formula for calculating this $dV/d\lambda$ can be derived in a manner similar to that shown in the previous section. However, we present a more general method for calculating this $dV/d\lambda$ term in the next section.

2.5 Generalization of TAVSI to an N-bus System

Given an N-bus system where $\mathcal{G}$ is the set of generator buses, $\mathcal{L}$ is the set of load buses, then; to evaluate the TAVSI at a given load bus i , all induction motor loads are first replaced with their equivalent power flow model i.e. a constant P load behind a transmission line. Then, the calculation of dP_L/dV at load bus i can be evaluated as follows:

Given the general vector form of the system power flow equations (with the consideration of both load characteristics and the loading parameter):

$$P_{INJ}(V, \delta) = P_G(\lambda) - P_L(V, \lambda) \quad (22)$$

$$Q_{INJ}(V, \delta) = -Q_L(V, \lambda) \quad (23)$$

Here, V and δ represent the vectors of buses' voltage magnitudes and voltage angles; P_G , P_L , and P_{INJ} are the vectors of real power generation, consumption, and injection at both generation and load buses. Q_G , Q_L , and Q_{INJ} Represent the vectors of reactive power generation, consumption, and injection at only load buses. From (22) and (23), the change in system states (V and δ) and the control variable (λ) are related according to the following set of equations:

$$\frac{\partial P_{INJ}}{\partial \delta} \Delta \delta + \left(\frac{\partial P_{INJ}}{\partial V} + \frac{\partial P_L}{\partial V} \right) \Delta V = \left(\frac{\partial P_G}{\partial \lambda} - \frac{\partial P_L}{\partial \lambda} \right) \Delta \lambda \quad (24)$$

$$\frac{\partial Q_{INJ}}{\partial \delta} \Delta \delta + \left(\frac{\partial Q_{INJ}}{\partial V} + \frac{\partial Q_L}{\partial V} \right) \Delta V = -\frac{\partial Q_L}{\partial \lambda} \Delta \lambda \quad (25)$$

Therefore, the vector of system states' slopes can be calculated as follows:

$$\begin{bmatrix} \frac{d\delta}{d\lambda} \\ \frac{dV}{d\lambda} \end{bmatrix} = \begin{bmatrix} \frac{\partial P_{INJ}}{\partial \delta} & \left(\frac{\partial P_{INJ}}{\partial V} + \frac{\partial P_L}{\partial V} \right) \\ \frac{\partial Q_{INJ}}{\partial \delta} & \left(\frac{\partial Q_{INJ}}{\partial V} + \frac{\partial Q_L}{\partial V} \right) \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial P_G}{\partial \lambda} - \frac{\partial P_L}{\partial \lambda} \\ -\frac{\partial Q_L}{\partial \lambda} \end{bmatrix} \quad (26)$$

After $dV_i/d\lambda$ at load bus i is obtained, then the TAVSI at that bus is calculated using (4); (13); and (17) or (18). It should be noted that TAVSI is not applicable to tie buses since they supply no load.

The index as defined in (4) cannot be used as an indicator of the “degree of weakness” of a given load bus when compared to other load buses. This is because the index is calculated based on raw slopes, i.e., dimensioned slopes. As stated earlier, the index represents the angular difference between the tangent to the P-V nose curve and the tangent to the load P-V characteristic curve. Using dimensioned slopes result in a relative rather than an absolute measure of distance to collapse (relative to the bus's voltage and power consumption). This does not allow for fair comparison among different load buses with different voltages and load power consumptions. Therefore, in order to improve on the value of TAVSI, it is proposed here to use normalized slopes rather than raw slopes. This would enable fair comparison among the different load buses which in turn allows for the definition of a global threshold for deciding on system stability, and the identification of weak buses where taking corrective actions should be most effective. At a given load bus i , the slopes of P-V curve and load curve are normalized as follows:

$$\left. \frac{dP_{Li}}{dV_i} \right|_{Nose/Load Curve} = \frac{V_i}{P_{Li}} \left. \frac{dP_{Li}}{dV_i} \right|_{Nose/Load Curve} \quad (27)$$

The modified TAVSI is then calculated as follows:

$$TAVSI_i = \tan^{-1} \left(\frac{dP_{Li}}{dV_i} \Big|_{Load\ Curve} \right) - \tan^{-1} \left(\frac{dP_{Li}}{dV_i} \Big|_{Nose\ Curve} \right) \quad (28)$$

In what follows, the performance of the modified TAVSI in terms of ranking and identifying weak buses is compared to that of the original TAVSI [19]. To draw a valid conclusion, a third index was included in the study: the P-index [11]. In [11], the authors proposed a new method for characterizing the point of maximum power transfer i.e., the nose point of the P-V nose curves. According to [11], at the system loadability limit, for any given load bus, the power gained due to the connection of any additional load would all be lost due to the accompanying voltage drop i.e. the net power increase being zero. Accordingly, they defined the P-index as the ratio of those two power components: gained and lost. The authors were able to arrive at an expression for such a ratio in terms of voltage sensitivities with respect to load power. Equation (29) details the formula for calculating the P-index for a given load bus i .

$$P_{index,i} = \frac{-2 \frac{P_{Li}}{V_i} \frac{dV_i}{dP_{Li}}}{1 - 2 \frac{P_{Li}}{V_i} \frac{dV_i}{dP_{Li}}} \quad (29)$$

The value of the P-index varies from “0” at no load to “1” at the point of maximum power transfer. It should be noted that, as explained previously in Section 2.2, the system point of collapse does not necessarily coincide with the point of maximum power transfer (i.e., the nose point of the P-V nose curve). Depending on the load P-V characteristics, the system might be able to accommodate the connection of additional load without collapsing. Therefore, to enable a fair comparison between the TAVSIs (original and modified) and the P-index, a constant PQ load model is assumed for all the loads of the system.

The ranking study is carried out on the IEEE 14-bus system. Table 2-2 lists the load bus rankings according to the modified TAVSI, original TAVSI, and the P-index. These rankings are taken at a modified TAVSI ≈ 40 for bus 14 (a candidate threshold for raising an alarm). As seen from Table 2-2, both the modified TAVSI and the P-index yielded the same ranking results with bus 14 ranked as the weakest bus (i.e. lowest TAVSI and highest P-index). On the other hand, the original TAVSI yielded a completely different ranking for the system buses. These results confirm that the original TAVSI may not be a good choice for ranking purposes.

A final note on the calculation of the TAVSI: as it was originally proposed in [19], the index at a given load bus i can be calculated based on the Thevenin equivalent of the system at that bus. A Thevenin based approach would eliminate the need of having full knowledge of system topology, states, and especially the load characteristics at each load bus in the system. Therefore, monitoring could be conformed to a specific area of the system where information on the load would only be needed for the pre-selected set of load buses. If a Thevenin-based approach is adopted, and if the load at the monitored bus is composite, then the induction motor part of the load needs to be included in the Thevenin equivalent. As shown previously, the induction motor load is equivalent to a constant P load behind a transmission line. Therefore, it can be thought of as part of the system i.e. another load bus connected to load bus i through a transmission line. Hence, it should be included in the Thevenin equivalent of the system at load bus i . After obtaining the Thevenin equivalent at load bus i , the TAVSI can be evaluated according to (4), (16), and (17) or (18).

Table 2-2 IEEE 14-bus system rankings, taken at TAVSI = 40.

	Rankings based on original TAVSI		Rankings based on modified TAVSI		Rankings based on P-index	
	Bus No.	TAVSI	Bus No.	TAVSI	Bus No.	P-index
Weakest  Strongest	11	3.5	14	39.5	14	0.708
	12	6.0	13	42.7	13	0.685
	10	8.8	10	42.7	10	0.684
	5	11.9	9	43.6	9	0.677
	13	12.8	12	43.7	12	0.677
	14	13.3	11	44.0	11	0.675
	9	27.5	4	54.8	4	0.585
	4	50.2	5	57.8	5	0.557

2.6 Simulation Studies

The performance of the proposed TAVSI is investigated on both: (1) the IEEE 14-bus system [25] and (2) a 181-bus reduced equivalent of the Western Electricity Coordinated Council (WECC) system [26]. Voltage collapse simulations were carried out in time-domain using PSAT toolbox. For each system, each load bus is assumed to have either a ZIP, an exponential, or a composite load connected to it. In order to drive the test system into voltage collapse, the system load (or parts of it) is increased at a uniform rate until the system collapses. For each load bus, the value of TAVSI is evaluated across the entire simulation period. A TAVSI value of 30 is assumed as a threshold for raising an alarm (average of the recommended range of threshold values). The following subsections present the simulation results along with a discussion on the performance of TAVSI.

2.6.1 IEEE 14-Bus System Test Case

For the IEEE 14-bus system shown in Figure 2-6, loads at buses 9, 10, 11, 12, 13, and 14 are modeled as ZIP loads with 30% constant Z, 40% constant I, and 30% constant P. On the other hand, loads at buses 4 and 5 are modeled as exponential loads with both α_P and α_Q being set equal to 0.5. In addition, induction motor loads are added to the existing loads of both buses 9 and 14. Accordingly, the loads at buses 9 and 14 are of composite nature (a ZIP load plus an induction motor load). The parameters of the induction motor load are given in Table 2-3.

The system load (except for induction motor loads) and real power generation were uniformly increased at a 0.2% per second increase rate. Figure 2-7 and Figure 2-8 show the voltage and calculated TAVSI for all load buses, i.e., buses 4, 5, and 9 to 14. As seen from Figure 2-7, the system collapsed at time $t = 121.6$. Now, looking at Figure 2-8, as the system was approaching

Table 2-3 Induction motor parameters in p.u. (100 MVA base) [17]

Rs	Xs	Rr	Xr	Xm	H (sec)	P (base-case)
0.0783	0.832	0.1055	0.832	16.48	0.1836	0.1

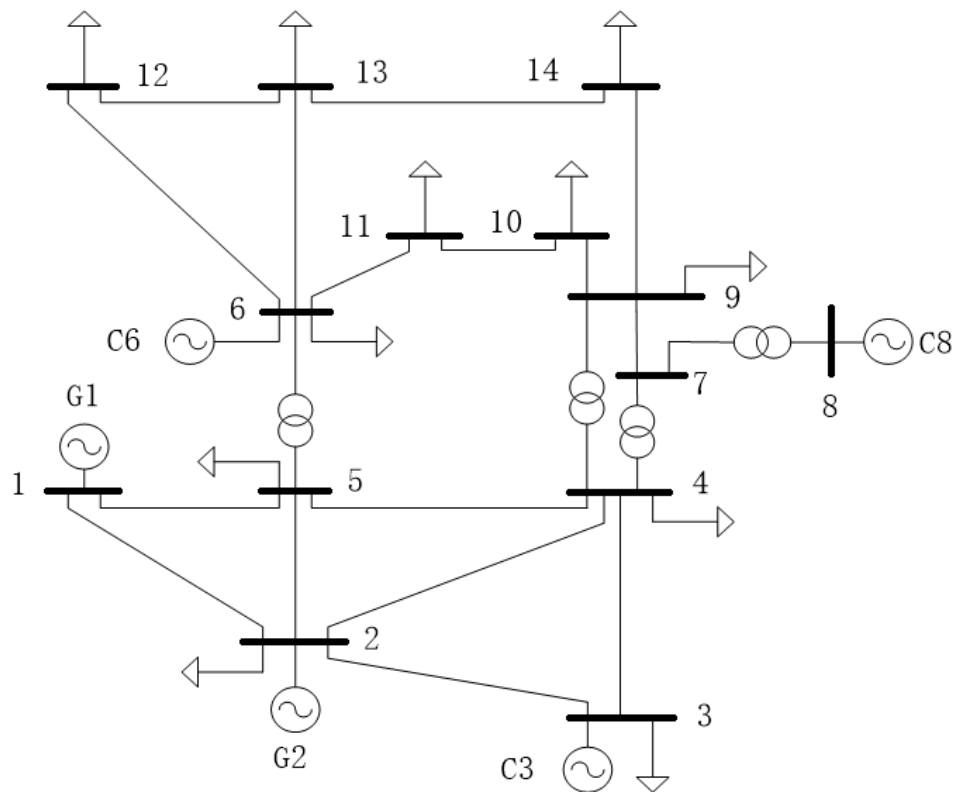


Figure 2-6 IEEE 14-bus system.

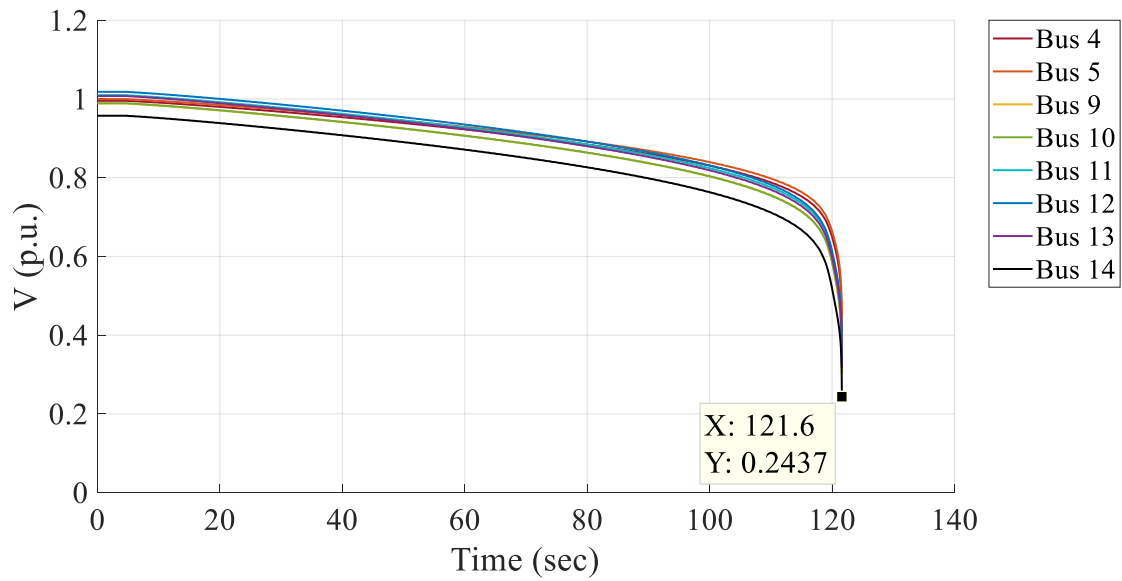


Figure 2-7 IEEE 14-bus system: voltage of load buses

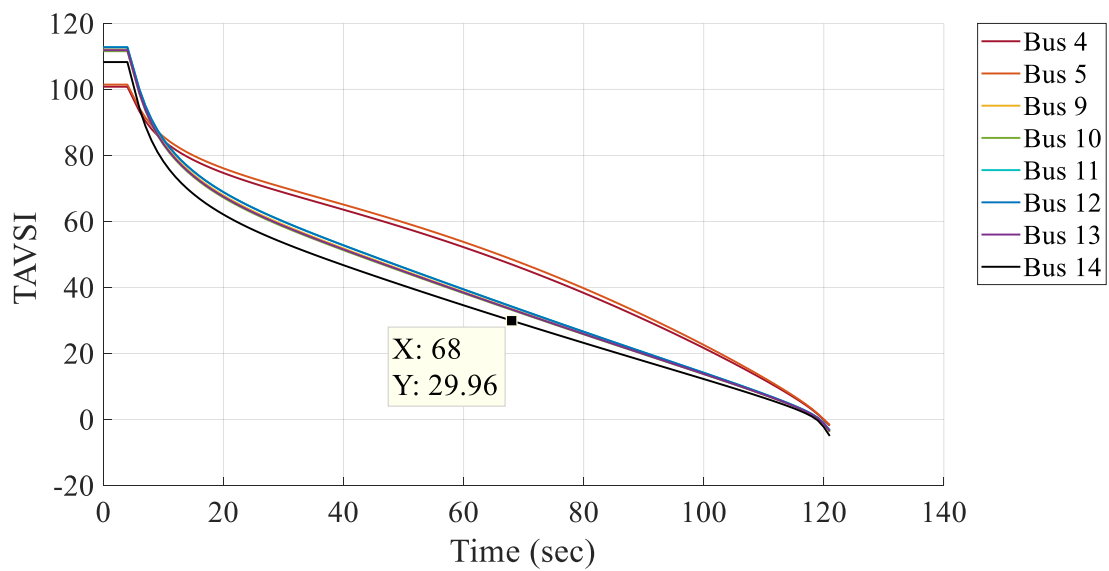


Figure 2-8 IEEE 14-bus system: TAVSI of load buses

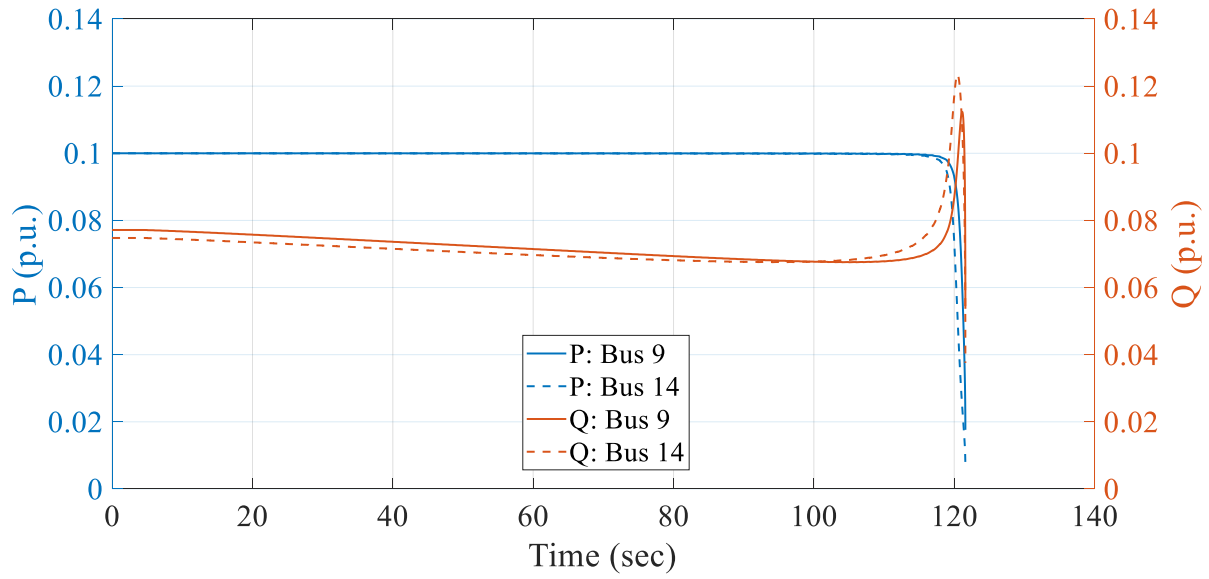


Figure 2-9 IEEE 14-bus system: motor loads' active and reactive power demand

collapse, the TAVSI started to attain small values. By time $t = 68$ seconds the TAVSI of bus 14 had reached a value less than 30° . This gives a clear indication of system instability at that point. By time $t \sim 119$ seconds, the TAVSI had reached a value of zero. It should be noted that a TAVSI of zero represents the point beyond which the equivalent static model has no solution. Now, even though the TAVSI has reached a value of zero, the dynamic simulation was able to advance for ~ 3 more seconds before the system went into a complete collapse at $t = 121.6$ seconds. In a nutshell, the system did not collapse immediately after the TAVSI has reached a value of zero due to motors' inertia and their power-slip dynamics. To further explain, the reader is referred to Figure 2-9. By the time $t \sim 119$ seconds, both induction motors have started to stall. When a motor stalls, its speed drops to zero (slip = 1). Looking at the circuit of Figure 2-3 along with Equation (20), with a slip of 1, the power delivered to the motor shaft, P_{sh} , becomes zero and the induction motor becomes a sink for reactive power. These power-slip dynamics have enabled the system to retain a solution for a few more seconds before the system went into a complete collapse at $t = 121.6$ seconds (i.e., the dynamic system equations were no longer solvable). During these 3 seconds, the calculated TAVSIs had negative values. Now, since the TAVSI is calculated based on measurements from a dynamic simulation rather than a static load flow solution, then, it is possible (but not necessary) for the TAVSI to have a negative value during the dynamic collapse event.

A final note on the results of this case study, as explained previously in Section 0, using normalized slopes would enable a fair comparison of proximity to collapse among system load buses. As seen from Figure 2-8, load buses 9, 10, 11, 12, 13, and 14 are “weak” buses when compared to load buses 4 and 5 with bus 14 being the weakest bus.

Another voltage collapse scenario was simulated for the IEEE 14-bus system. This time, the base-case load induction motors were chosen as 0.17 and 0.05 for motors at buses 9 and 14,

respectively. Figure 2-10 and Figure 2-11 show the voltage and the calculated TAVSIs. Figure 2-12 shows the active and reactive power consumption of the motor loads at buses 9 and 14. As can be seen from Figure 2-11, at time $t = 47$ seconds the TAVSI of bus 14 had reached a value less than 30° . This gives a clear indication of system instability at that point. By time $t \approx 82$ seconds, the calculated TAVSIs had near-zero values which indicate that the system is on the verge of collapse. By that time, as can be seen from Figure 2-12, the system was no longer able to supply the mechanical load of the motor connected to bus 9 and therefore, the motor started to stall. Similar to the previous case study, the stalling of the motor load at bus 9 has enabled the system to override the predicted collapse and transition through a set of feasible (but not stable) operating points till it eventually collapsed at $t = 99.1$ seconds.

The results of both case studies support the earlier discussion of Section 2.4 in that the static-based TAVSI would hit its theoretical limit of zero slightly before the actual dynamics collapse the system. Indeed, in some cases, the motor dynamics might enable a momentary recovery of the TAVSI indicator showing a brief feasible operation. However, this recovery has been observed to be unstable leading to ultimate collapse. Therefore, prevention actions should be taken as early as when the TAVSI hits its alarm threshold.

2.6.1 181-Bus WECC System Case

The second test system, as shown in Figure 2-13, is a 181-bus reduced equivalent of the WECC system developed at the CURENT research center, [26]. The system is driven into collapse through a uniform increase of part of its load (the load at the Los Angeles area) at a 0.2% per second increase rate. Loads of this area are modeled as either a ZIP, exponential, a composite ZIP + induction motor, or a composite exponential + induction motor load. The rest of the system load is modeled as a constant Z load.

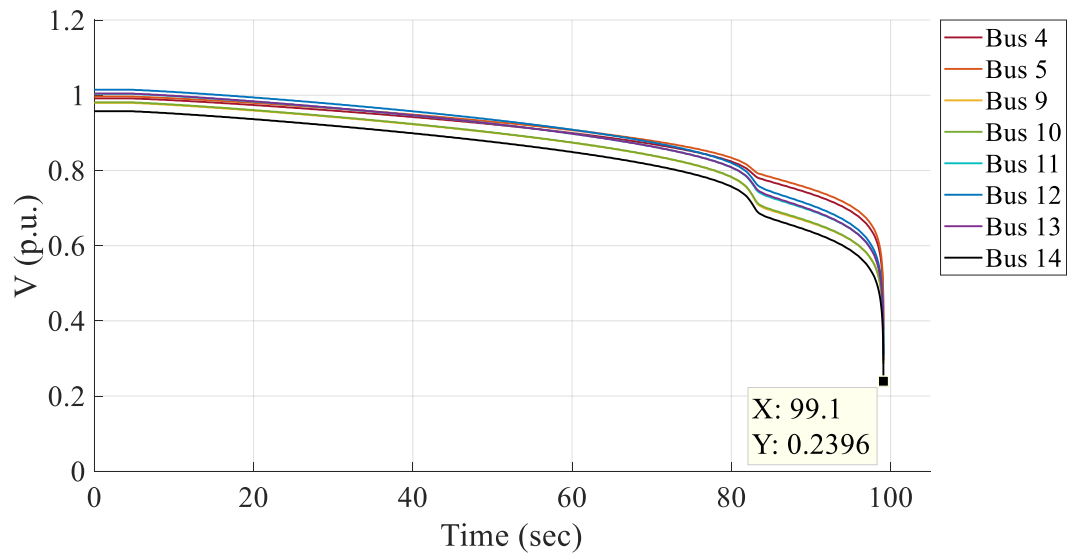


Figure 2-10 IEEE 14-bus system: voltage of load buses

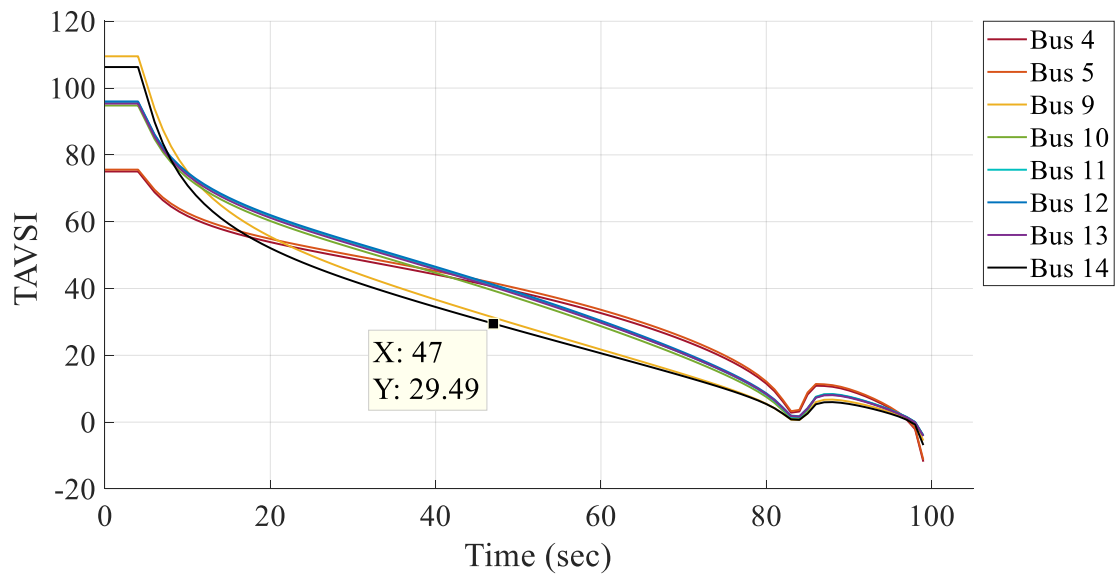


Figure 2-11 IEEE 14-bus system: TAVSI of load buses

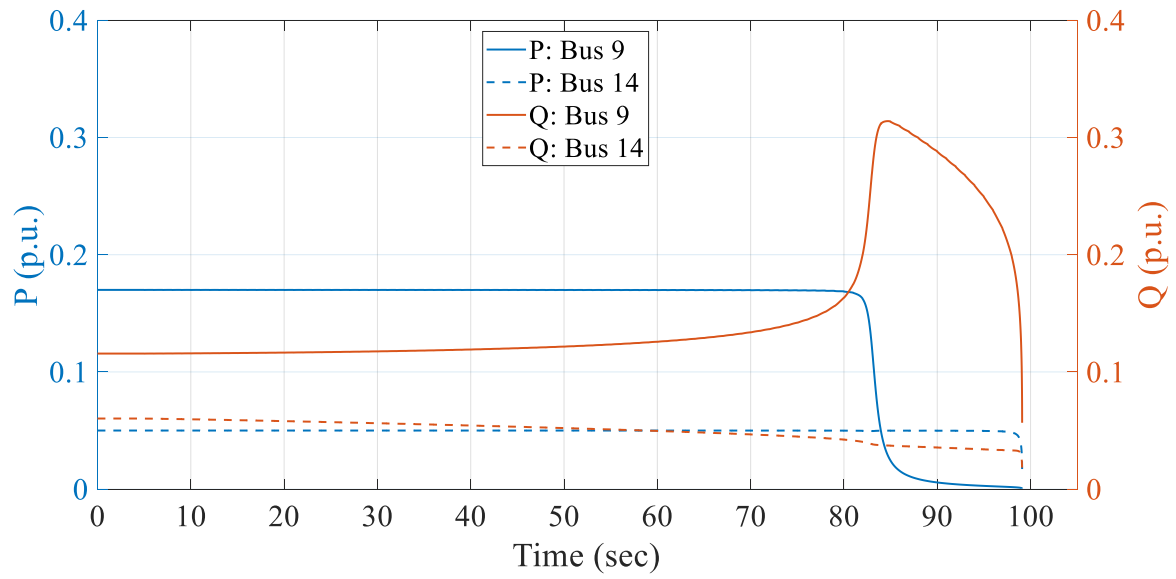


Figure 2-12 IEEE 14-bus system: motor loads' active and reactive power demand

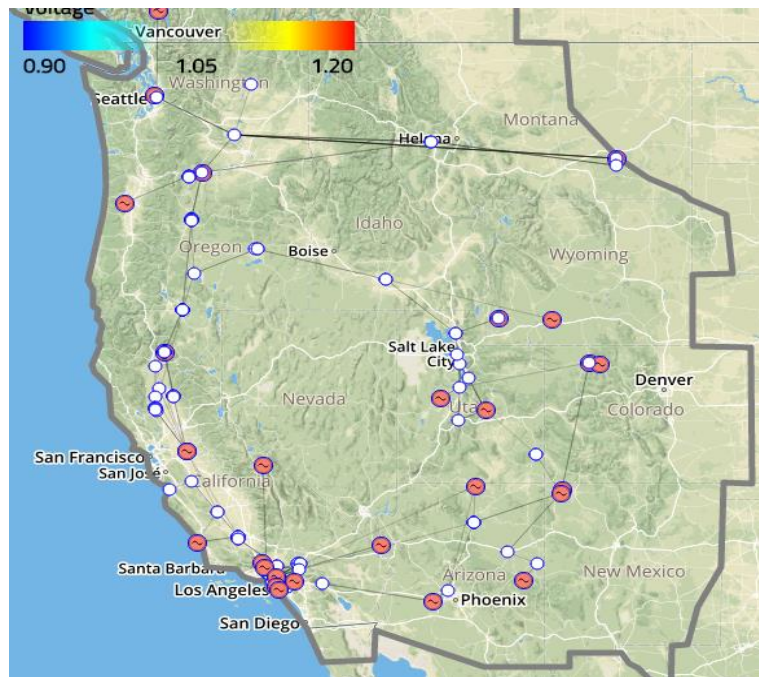


Figure 2-13 An illustrative diagram of the CURENT WECC test system.

Due to limits in space, the plots presented and shown in Figure 2-14 and Figure 2-15 are only for the weakest 9 load buses. According to the calculated TAVSIs, the weakest 9 buses of the system were identified to be buses 41, 50, 51, 54, 55, 57, 58, 62, and 63. From Figure 2-13, it can be seen that the system voltage collapsed at $t = 264$ seconds. The TAVSI at buses 51, 54, 57, and 58 gave an early indication of system instability (around 20 seconds before system collapse) as it hit a value less than 30 at $t = 240$ seconds.

2.6.1 Conclusions and Final Remarks

In all study cases, the TAVSI demonstrated a consistent behavior as it always reached a value of zero at the system point of collapse. It also gave an early indication of system instability which allows for enough time to take corrective actions. In addition, it successfully identified weak buses where taking corrective actions would be most effective.

Note, the above case studies work for both measurement-based and model-based approaches. If the measurement is precisely accurate, both produce identical results. Thus, the proposed TAVSI can be applied with both approaches.

It should also be noted that the advantage of TAVSI in comparison with traditional VSIs has been analyzed and demonstrated in the previous work [19] when ZIP or exponential loads are studied. Also, there is no previous work in VSI incorporating motor loads. Thus, here we only demonstrate the effectiveness of TAVSI with composite ZIP and induction motor loads included.

2.7 Conclusion

This work extended and generalized the newly proposed voltage stability indicator, TAVSI. Its calculation was extended to include ZIP and induction motor load. TAVSI measures the tangential angle difference between the system P-V curve and the load characteristic curve, so it provides a

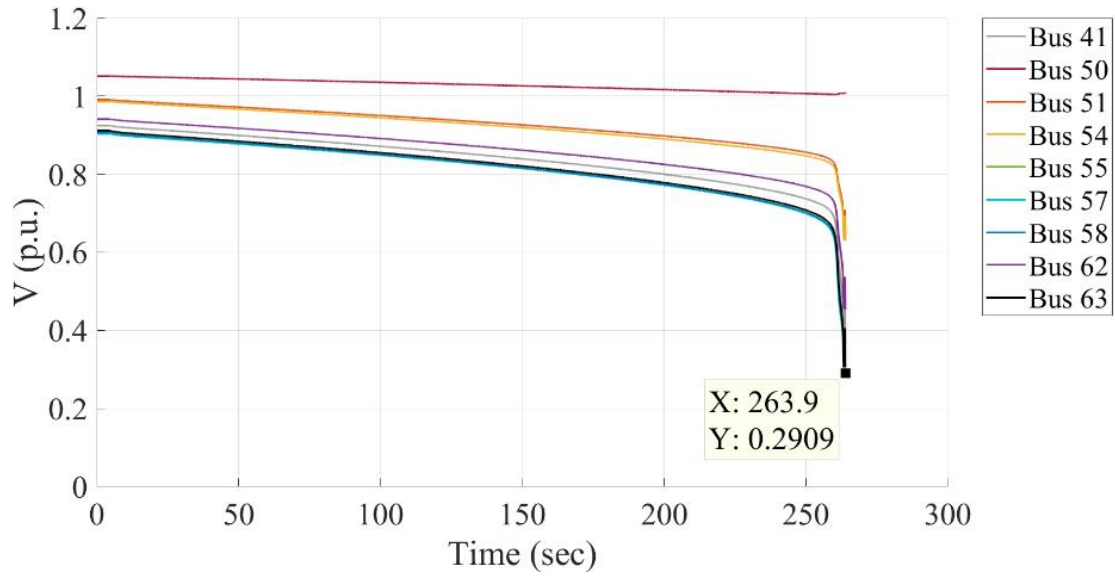


Figure 2-14 181-bus WECC system: voltage of the 9 weakest load buses

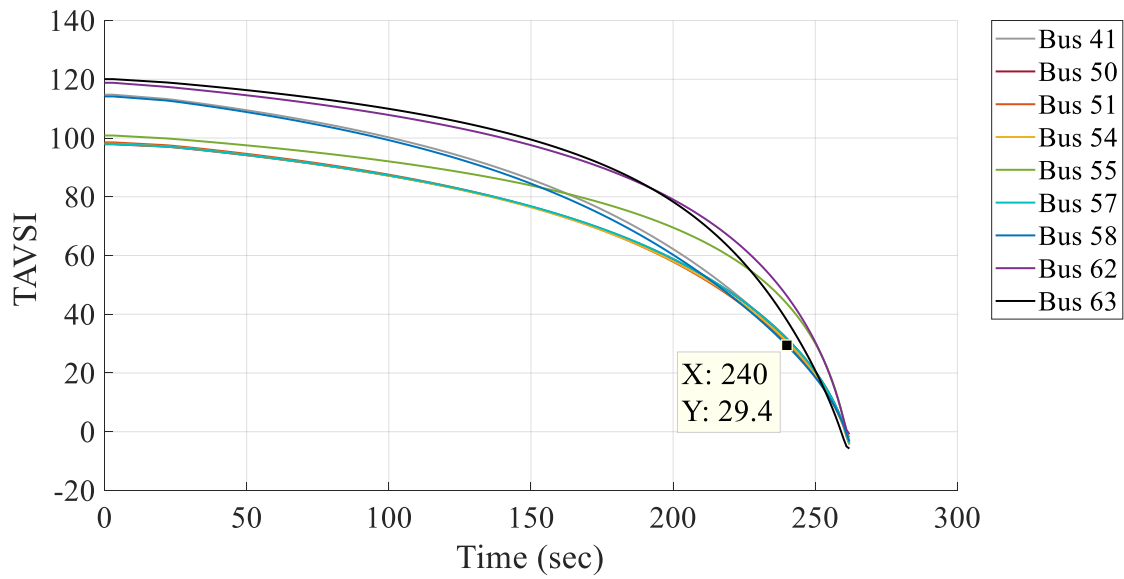


Figure 2-15 181-bus WECC system: TAVSI of the 9 weakest load buses

more reliable assessment of system voltage stability as opposed to traditional load models. The value of the TAVSI varies from π at zero loading to zero at the system point of collapse (i.e., maximum loading).

The performance of this index was tested on both the IEEE 14-bus system and the CURENT's version of the 181-bus WECC system. The results show that TAVSI gives a good and reliable assessment of system stability.

The TAVSI is calculated based on normalized sensitivities which allows for comparison among different load buses. This allows for: 1) the definition of a global threshold for deciding on when to take corrective actions; and 2) the identification of weak buses where corrective actions would be the most effective.

The TAVSI at a given load bus can be calculated based on either the complete system model (a model-based approach) or the Thevenin equivalent of the system at that bus (a measurement-based approach). Whether a model-based or a measurement-based approach is chosen depends on the method's applicability in the studied system.

Chapter 3. Data-driven and Model-based Hybrid Reinforcement

Learning to Reduce Stress on Power Systems Branches

In this chapter, we propose a reinforcement learning (RL) approach to tackle the control problem of branch overload relief in large power systems. Here, a control agent is trained to change generators' real power output in order to relieve the stressed branches. For large power systems, this control problem becomes one whose decision space (i.e., the action space) is both highly dimensioned and continuous. This makes it extremely difficult to have successful training for RL-based agents. Therefore, we propose a hybrid approach where optimal power flow (OPF) based actions (model-based actions) are utilized to help the agent learn an optimal control policy (data-driven actions). The proposed approach was tested and compared to the generic data-driven, deep deterministic policy gradient (DDPG)-based approach on both the IEEE 118-bus system and a larger 2749-bus real-world system. The obtained results show that the hybrid approach performs well for large power systems and that it is superior to the DDPG-based approach.

3.1 Introduction

Overloaded branches, i.e. transmission lines and transformers, have been responsible (in part or fully) for several major blackouts worldwide: the US-Canadian grid 2003, Italy 2003, and India 2012 [1]. An overloaded branch might trip out on overload or zone-3 distance protection. The overloaded branch might also sag into a tree and trip out on fault. This, in turn, might cause additional branches to become overloaded and also trip out. To prevent such a cascade of outages, the initially overload branch(es) must be secured as quickly as possible. One way to secure an overloaded branch is through generation re-dispatch. In general, generation re-dispatch is formulated as an optimal power flow (OPF) problem [27-29]. However, in the context of branch

over-load relief, OPF might not be a suitable tool. This is because generators have limits on their ramping rates. Therefore, the optimal solution might need to be applied in more than one time-step. The OPF does not inherently offer any guidance on the correct sequence of steps to achieve the desired dispatch, and an unguided approach can conceivably lead to overloading some of the healthy lines and/or increasing the overload on some of the already over-loaded. If the ramp rate constraints were included in the formulation of the OPF, and the problem is then solved as a scheduling optimization over the time needed to effect the schedule, then this would greatly increase the mathematical complexity of the OPF model. In addition, power systems are typically large and highly complex systems. This makes the OPF problem computationally expensive to solve and apply in a real-time frame of work. Such high computational requirements, not only for OPF but for other model-based assessment and control approaches as well have started to draw researchers more towards considering the adoption of model-free approaches in the field of power systems. Within this context, reinforcement learning (RL) offers a panel of methods that allow agents to learn a control law from interactions with a system without the need for the system model.

In general, reinforcement learning is a suitable tool for any problem that can be formulated as a sequential decision problem. However, applying RL to large-scale control problems remains a great challenge. Indeed, successful implementation of RL has been reported in the literature of power systems operation and control. However, it lacks works that implement RL to problems with a similar nature to ours, i.e., large-scale operational problems whose control variables are both: continuous in nature and great in number. References [30, 31] provide a comprehensive review of the applications of RL in the field of power systems operation. Some of the surveyed works apply basic RL algorithms to small-scale control problems. For example, in [32], tabular Q-learning is utilized for training an agent to control transformer taps and reactive power compensators to

regulate substations' voltage. Basic RL methods are known to have limited applicability to tasks with highly-dimensional state-action spaces. Accordingly, some of the surveyed works were tailored around the theory of reinforcement learning to tackle this “curse of dimensionality”. In [33], a multi-agent, distributed control framework is adopted to enable the application of the tabular Q-learning to the problem of reactive power control when considering a larger set of controlled devices. On the other hand, some of the surveyed works have utilized deep reinforcement learning (DRL) to enable the application of RL to large-scale operational problems (large mainly in terms of their state-spaces). In the field of RL, the use of deep learning in conjunction with reinforcement learning has led to significant breakthroughs in various tasks. DRL has enabled learning and significantly improved on the generalization of the control policy to unseen states. However, as pointed by the authors of [34], for tasks that have a large number of continuous control variables, the reported success of DRL is not of a great scale,

This work draws on and extends the one previously presented in [35]. In that work, we proposed an RL-based generation re-dispatch scheme whose objective is to relieve the overload on system branches. The deep deterministic policy gradient (DDPG) algorithm [36] was adopted for training the re-dispatch agents. The proposed approach was successfully tested on both the IEEE 14-bus and 39-bus systems. The trained agents of both systems demonstrated a better performance when compared to the classical interior-point-based OPF approach. The application of the proposed RL-based control on the IEEE 118-bus larger system, however, was not as successful. Therefore, in this work, we explore and identify the various challenges when extending the application of the proposed RL-based control scheme to larger power systems at the scale of over 1000 buses, and investigate solutions to overcome these challenges. As a result of our investigations, we proposed

a hybrid data-driven and model-based reinforcement learning approach (HRL) which enables the utilization of RL for the control problem of branch over-load relief in large power systems.

To summarize, the main contributions of this study include: (1) we propose to tackle the control problem of branch overload relief in power systems through generation re-dispatch as a sequential-decision control problem. This is because generators need to be dispatched to secure the system without overloading any healthy lines and/or increasing the overload on any of the already overloaded ones. Accordingly, we propose to adopt reinforcement learning as “the” tool to train our control agent. (2) we present a detailed investigation into the various challenges involved in the application of RL to the control problem of branch overload relief in “large” power systems. (3) as a result of our investigations, we propose a hybrid data-driven and model-based approach that utilizes actions obtained based on generator shifting factors to improve the training of the RL-agents. (4) finally, we present studies that demonstrate the effectiveness of our proposed approach when compared to the DDPG-based approach.

The remainder of this study is organized as follows: Section 3.2 presents a brief overview of reinforcement learning and the DDPG algorithm. In Section 3.3, the application of reinforcement learning to the control problem of branch over-load relief is investigated. Section 3.4 presents the proposed hybrid approach. Simulation studies are presented in Section 3.5. Section 3.6 presents a discussion on training HRL agents for large power systems. Finally, Section 3.7 concludes this work.

3.2 Reinforcement Learning and the Deep Deterministic Policy Gradient

Algorithm

In reinforcement learning, the agent learns a control policy based on its interactions with the system environment. In general, the agent interacts with the environment in discrete time-steps. At each time-step t , the agent observes the system states s_t , decides on an action a_t and receives a reward r_t . The agent learns progressively with time as it keeps accumulating more and more experience until it finally converges on an optimal control policy. The literature on reinforcement learning offers a wide range of learning algorithms. This work is based on the deep deterministic policy gradient (DDPG) learning algorithm [36]. This section presents a brief overview of this DDPG algorithm. The first subsection presents some key definitions and concepts used in the field of reinforcement learning. The second subsection presents a brief description of the DDPG algorithm.

3.2.1 Definitions

- 1. Policy:** A policy π , is a function that defines the behavior of the agent. A policy can either be deterministic or stochastic. A deterministic policy, $\pi: S \rightarrow A$, maps states to specific actions. On the other hand, a stochastic policy outputs a probability distribution over a pre-defined set of actions $\pi: S \rightarrow \mathcal{P}(A)$.
- 2. Return:** Return is defined as the sum of discounted future rewards $R_t = \sum_{i=t}^T \gamma^{(i-t)} r_i(s_i, a_i)$ where γ is a discounting factor, $\gamma \in [0, 1]$.
- 3. Value function:** Value functions represent the expected value of a given state. Two types of value functions are defined for a given state: 1) state-value function, $V(s_t)$, which is defined as the expected return starting from state s_t and following policy π . 2) action-value function,

$Q(s_t, a_t)$, which is defined as the expected return after taking action a_t in state s_t and following policy π thereafter.

4. **Bellman's equation:** In general, the agent attempts to learn a policy that maximizes the expected return of every state. The majority of reinforcement learning algorithms utilize the action-value function to infer the optimal policy. Those algorithms are all based on Bellman's principle of optimality [37]. This principle leads to the following necessary condition on the optimality of the action-value function:

$$Q^*(s_t, a_t) = \mathbb{E} \left[r_t(s_t, a_t) + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \right] \quad (1)$$

where $\mathbb{E}[\cdot]$ indicates the expected value. Equation (1), also known in the literature as Bellman's equation, is a recursive relationship which allows for the use of iterative approaches to solve for the optimal action-value function $Q^*(s, a)$.

3.2.2 Deep Deterministic Policy Gradient (DDPG) Algorithm

DDPG is a learning algorithm designed for control problems where the agent policy is deterministic and the action space is continuous [36]. DDPG utilizes an actor-critic frame of work which allows for concurrent learning of a policy function (the actor) and a Q-function (the critic). In [11], the authors proposed the use of neural networks as function approximators for both actor and critic functions. The actor network is parameterized by θ^π while the critic network is parameterized by θ^Q . The algorithm uses the Bellman equation to learn the Q-function, i.e. θ^Q , and uses the Q-function to learn the policy, i.e. θ^π .

1. **Q-learning:** As mentioned above, Bellman's equation provides the basis for learning the optimal action-value function $Q^*(s, a)$. Now, given the parameterized critic function $Q(s, a|\theta^Q)$, then, its parameters, θ^Q , can be optimized by minimizing the mean-squared

Bellman error (MSBE) given below in (2). This error indicates how close the value of Q comes to satisfying the optimality condition of (1):

$$L(\theta^Q) = E[(Q(s_t, a_t | \theta^Q) - y_t)^2] \quad (2)$$

Where

$$y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \pi(s_{t+1}) | \theta^Q) \quad (3)$$

y_t is known as the target. As seen from (3), the target value depends on the same parameters being optimized for, i.e. θ^Q . This fact is known to make learning through MSBE minimization unstable for neural networks. In order to stabilize the learning process, two techniques are adopted: a replay buffer, and a separate critic target network for calculating y_t . Readers are referred to [36] for more details on those two techniques.

2. Policy learning: The optimal policy is one that satisfies:

$$\theta^{\pi^*} = \arg \max_{\theta^\pi} (J = E[Q(s, \pi(s | \theta^\pi))]) \quad (4)$$

Since the action space is continuous and the Q-function is differentiable with respect to the action then, gradient ascent can be used to solve the optimization problem of (4). Accordingly, the parameters of the actor function are updated using the following gradient:

$$\nabla_{\theta^\pi} J = \frac{1}{N} \sum_i \nabla_a Q(s, a | \theta^Q) |_{s=s_i, a=\pi(s_i)} \nabla_{\theta^\pi} \pi(s | \theta^\pi) |_{s=s_i} \quad (5)$$

In [38], the authors proved that this is equivalent to the policy gradient, hence the name. Since the actor network is also used in (3), then, in order to have stable learning, a separate actor target network is used when calculating y_t .

3.3 Data-Driven Reinforcement Learning and the Control Problem of

Branch Over Load Relief

This section presents a discussion on the various elements (i.e., states, actions, reward function ... etc.) involved in the design of the data-driven DDPG-based controller. It also identifies, explores, and analyzes the various challenges and limitations one might face when training the DDPG-agent for larger power systems.

3.3.1 States

Given an N-bus power system where $\mathcal{G}$ is the set of system generators, $\mathcal{G}^c$ is the set of generators available for control, $\mathcal{B}$ is the set of system branches, $\mathcal{B}^{OL,t}$ is the set of overloaded branches at time-instant t , then; the system states are defined as:

$$S = \{s | s_t = \{S_{i,t} \forall i \in \mathcal{B}\} \cup \{P_{gj,t} \forall j \in \mathcal{G}\} \quad (7)$$

where $S_{i,t}$ is the magnitude of the apparent power (MVA) flow of the i^{th} branch, $P_{gj,t}$ is the real output power of the j^{th} generator at step t .

3.3.2 Actions

The action space is defined as follows:

$$A = \{a | a_{j,t} = \Delta P_{gj,t} \forall j \in \mathcal{G}^c\} \quad (8)$$

where $\Delta P_{gj,t}$ is the amount of change in the real output power of the j^{th} generator at step t . $\Delta P_{gj,t}$ is continuous in the range $[\Delta P_{gj-\min}, \Delta P_{gj-\max}]$ where $\Delta P_{gj-\min}$ and $\Delta P_{gj-\max}$ are the minimum and maximum allowable change for generator j between two consecutive time steps. It should be noted that if $\Delta P_{gj,t}$ results in violating the output limits of generator j , then, $P_{gj,t}$ is set at the violated limit.

3.3.3 Reward Function Design

The success of an RL algorithm relies heavily on the design of the reward function. Reward functions can either be sparse or dense. A sparse reward function is one that rewards the agent with a fixed value (typically a zero) most of the time. On the other hand, a dense reward is one that gives variant values to most of the transitions, thus the agent gets informative feedback at almost every time step. According to the DDPG algorithm, in the beginning, the actor policy is randomly initialized. For this particular control problem, the actor should be initialized in a way that makes it take small actions (small changes in generator outputs). This is because randomly changing generator outputs in large amounts could easily drive the system into collapse. Now that the taken actions are small, the agent might need a large number of steps before it can reach a winning terminal state. This issue gets more pronounced for larger systems where the number of controlled generators is relatively large.

If the reward function does not provide informative feedback on the taken actions, the reward signal would be sparse. Sparse rewards make it extremely difficult for the learning algorithms to connect such a long series of actions to a distant future reward. On the other hand, a dense reward helps the agent learn from the accumulated experience even when terminal states are not reached quite often. Optimizing performance on dense reward signals is expected to guide the agent towards its end goal of winning the game. However, due to the discounted nature of the action-value function (Q-function), dense rewards might encourage the agent to aim for short-term gains over long-term benefits. In addition, dense rewards might be misleading and, if not carefully designed, the agent might end up learning the wrong policies [39].

As mentioned above, the control problem in hand (over-load relief of power system branches) is one whose action space is continuous, and for large power systems, it would be highly-

dimensioned as well. Therefore, a dense reward might be a good option for helping the agent learn a control policy. In the literature, several works have proposed the use of the distance-to-goal as a measure of success for the actions taken by the agent [40-42]. For the control problem in hand, a dense distance-to-goal based reward can be formulated as follows:

$$r_t = -r_{IM,t} + r_{TML}$$

where:

For all $i \in \mathcal{B}^{OL,t}$ (overloaded branches at time t) and $j \in \mathcal{B}^{OL,t-1}$ (overloaded branches at time $t - 1$):

$$r_{IM,t} = \sum_{i \in \mathcal{B}^{OL,t}} \frac{S_{i,t}}{S_{i-max}} + \sum_{j \in \mathcal{B}^{OL,t-1}} \frac{(S_{j,t} - S_{j,t-1})}{S_{j-max}} \quad (9)$$

and

$$r_{TML} = \begin{cases} +R & \text{If all branches are within limits} \\ -R & \text{If power flow diverges} \end{cases}$$

The above reward function is composed of two parts:

1. An intermediate reward $r_{IM,t}$: this reward is also composed of two terms: The first term reads as follows: at each control step, the agent is penalized for being in a state where one (or more) of the system's lines is overloaded. The magnitude of the penalty depends on: the number of overloaded lines; and the amount of overload on each of those lines. The second term reflects the value of the taken action; the penalty on each line is either: (a) increased if the line had a better condition on the previous state (action resulted in more loading of the line), or (b) decreased if the action resulted in relieving some of the line loading. The higher the intermediate penalty, the further is the agent from its goal hence, $r_{IM,t}$ resembles a measure of distance to the goal state.

2. A terminal reward r_{TML} : the agent is rewarded with r_{TML} if the taken action results in either: (a) securing the system or (b) collapsing the system. If the taken action has secured the system, then the agent is given a high reward, i.e., $r_t = +R$ and the episode is terminated. On the other hand, if the taken action collapses the system, then, the agent is penalized, i.e., $r_t = -R$ and the episode is terminated. The terminal reward should be set to a relatively large value. This is due to the following: at the beginning of the training period, where the taken actions are far from being optimal, the intermediate penalty could attain high negative values. High intermediate penalties might over-shadow the terminal reward and therefore, actions leading to “early” termination on collapse would have a higher Q-value than those leading to a “later” termination on a win. Therefore, in order to overcome this issue, the value of r_{TML} needs to be set to a “relatively” large value e.g. ± 50 .

This reward design is “expected” to lead the agent to avoid being in states where it is highly penalized (states with overloaded branches) and therefore, try to reach a terminal state where it gets a positive reward. However, three major factors might render the agent from learning such optimal behavior. Those are:

- First, the intermediate penalty is defined as a sum over all overloaded lines. This might lead to a deceptive comparison of state-action pairs that differ in the number of over-loaded lines and therefore, the agent might end up learning sub-optimal policies. To further explain, consider the following scenario: an outage result in overloading 4 lines at 110%, 150%, 150%, and 150%. Then, the agent takes an action that results in increasing the loading of the first line up to 200% while decreasing the loading of the rest of the lines down to 95%, 100%, and 105%. For this state-action pair, the reward according to (9) is equal to -2.45. Now, assume that the agent is exploring the environment (or maybe had its policy updated). Then, following the new policy,

the agent takes an action that results in decreasing the loading of all lines to 105%, 110%, 110%, and 110%. In this case, the calculated reward is -3.10. Comparing the two state-action pairs, based on the obtained rewards, the agent sees the first action as the better one. However, in real-world operation, the first action is a worse one as it results in unacceptable over-loading of “line-1” (200%). This scenario shows that actions which result in a fewer number of highly loaded lines could be seen as better actions than those which result in a larger number of lightly over-loaded lines.

- Second, even though bad actions are penalized (i.e., actions that result in overloading healthy lines and/or increasing the flow on an already overloaded line), the main objective remains to reach a state where all lines are secured. In other words, winning the game is more important than how the game is won. Therefore, even if the agent learns a policy that wins the game, the trajectory of the taken actions might have states where healthy lines became overloaded.
- Finally, with large target values ($y = r + \gamma Q'$), the training of both critic and actor networks becomes prone to the problem of exploding gradients [43].

In order to overcome the above issues, another termination condition is added to the formulation of the reward function. So, in addition to terminating on securing the system (i.e., winning the game) or collapsing the system (i.e., losing the game), an episode is to be terminated if the taken action results in increasing the flow on one (or more) of the over-loaded lines, and/or if it results in over-loading one (or more) of the healthy lines. With this new termination condition, the agent will try to learn a policy that secures the system by taking “good” actions only, i.e., actions that result in decreasing the flow on all overloaded. In other words, while progressing towards its end goal, the agent will not overload healthy lines and/or increase the flow on some of the already over-loaded lines. Accordingly, the reward function becomes:

$$r_t = -r_{IM,t} + r_{TML}$$

where:

For all $i \in \mathcal{B}^{OL,t}$ (overloaded branches at time t) and $j \in \mathcal{B}^{OL,t-1}$ (overloaded branches at time $t - 1$):

$$r_{IM,t} = \sum_{i \in \mathcal{B}^{OL,t}} \frac{S_{i,t}}{S_{i-max}} + \sum_{j \in \mathcal{B}^{OL,t-1}} \frac{(S_{j,t} - S_{j,t-1})}{S_{j-max}} \quad (10)$$

and

$$r_{TML} = \begin{cases} +R & \text{If all branches are within limits} \\ -R & \begin{cases} \text{If power flow diverges} \\ \text{If healthy branches become overloaded} \\ \text{If flow on overloaded branches increase} \end{cases} \end{cases}$$

3.3.4 Exploration

Another factor that has a major impact on the training outcome of the RL agent is the quality of the accumulated experience, especially at the early stages of the training process [44]. This issue is often tackled with exploration. Exploration aims at helping the agent accumulates “useful” experience by efficiently exploring the environment to reach terminal states (a win in particular). Typically, the agent uses a noise signal to explore its environment to find more efficient trajectories [36]. However, exploration with random noise signals does not achieve good results in high-dimensional continuous-action spaces [41].

3.4 Proposed Data-Driven and Model-based Hybrid Reinforcement Learning Approach

The proposed hybrid approach mainly deals with the shortcomings of random exploration which proves to be inefficient when used on high-dimensioned and/or continuous action spaces. We build

our hybrid design based on the work proposed in [41]. In that work, the authors proposed a method for training RL agents for tasks where exploration is difficult. This method incorporates what is known as “demonstrations” within the learning process. A demonstration is a pre-determined (by other control algorithms e.g. humans, simulations ... etc.) sequence of actions that is known to solve the task. The coming subsections present the details of our proposed data-driven, model-based approach.

3.4.1 Agent Training

The authors in [41] introduced a new loss component to the policy objective of (4). This loss, detailed in the following equation, is known in the literature as behavior cloning (BC) loss. Behavior cloning is a training process in which the agent learns to perform the intended task from pre-demonstrated behaviors.

$$L_{BC} = \sum_{i=1}^{N_D} \|\pi(s_i|\theta_\pi) - a_i\|^2 \quad (11)$$

Where N_D is the number of demonstrated samples; (s_i, a_i) is a pre-demonstrated state-action pair; and $\pi(s_i|\theta_\pi)$ is the action that the agent would take given the system state s_i and the actor’s current policy π . The gradient of this loss is computed using only demonstration examples. This gradient is then added to the policy gradient of (5) and accordingly, the actor parameters are updated as follows:

$$\theta_\pi^{k+1} = \theta_\pi^k + \lambda(\nabla_{\theta_\pi} J + \beta \nabla_{\theta_\pi} L_{BC}) \quad (12)$$

Where λ is the learning rate, β is a hyper-parameter that controls the contribution of behavior cloning to updating the actor network parameters.

3.4.2 Reward Function

Investigating (12), training the actor network with the aid of demonstrations can be seen as a multi-objective optimization problem. Therefore, like (12), one might use the following equation to update the actor parameters:

$$\theta_{\pi}^{k+1} = \theta_{\pi}^k + \lambda(\beta_J \nabla_{\theta_{\pi}} J + \beta_{BC} \nabla_{\theta_{\pi}} L_{BC}) \quad (13)$$

Where β_J and β_{BC} are weights that reflect the importance of each objective to the training of the RL agent. At the beginning of training, behavior cloning should be of more importance to help the agent explore and accumulate useful experience. As the agent starts to learn, the Q-maximization objective becomes of more importance (to allow the agent, if possible, to find a better policy than that used to obtain the demonstrations). This can be achieved by setting β_{BC} at a “relatively” high value (compared to β_J) at the beginning of training and then anneal its value as the training progress. As simple as it might seem, for the control problem in hand, tuning the β_J and β_{BC} parameters is not a simple task. This is because the Q-maximization and the behavior cloning objectives have different scales. If the maximum and minimum possible values of the objective functions are known beforehand (i.e., upper and lower bounds), then one can use these values to re-scale the gradients of the two objective functions. This would enable a fair comparison between the two objectives and thus, easier tuning of β_J and β_{BC} .

Since actions are bounded between 1 and -1, the behavior cloning loss is also bounded. However, maximum and minimum bounds on the Q-maximization objective depends on the reward function design. The reward function of (10) is unbounded as its value depends on both the number of overloaded lines and, the amount of overload on each of those lines. Therefore, one cannot determine the maximum and minimum bounds on the corresponding Q-function.

The above discussion shows that, for the control problem in hand, using the reward function of (10) might render the applicability of the training approach proposed in [41]. If that is the case, then one can drop the intermediate penalty term (i.e., $r_{IM,t} = 0$). In this way, the corresponding Q-function becomes bounded. This alternative formulation of reward function is detailed in the following equation:

$$r_t = r_{TML}$$

where:

$$r_{TML} = \begin{cases} +R & \text{If all branches are within limits} \\ -R & \begin{cases} \text{If power flow diverges} \\ \text{If healthy branches become overloaded} \\ \text{If flow on overloaded branches increase} \end{cases} \\ 0 & \text{Otherwise} \end{cases} \quad (14)$$

A summary of the hybrid training algorithm is given next in Table 3-1.

3.4.3 Demonstrations for the Control Problem of Branch Overload Relief

One way to obtain demonstrations for the over-load relief control problem is through the use of generation distribution factors. These factors describe how the current flow changes in system branches if the power injection of generation units is changed. The following details how such factors can be obtained and accordingly, how one could use these factors to obtain the demonstrations.

Given a branch that is connected between bus i and bus j : it is known that the magnitude of the MVA flow S_{ij} is related to the real and reactive power flows P_{ij} and Q_{ij} as follows:

$$S_{ij}^2 = P_{ij}^2 + Q_{ij}^2 \quad (15)$$

Table 3-1 Hybrid data-driven and model-based training algorithm

Randomly initialize critic Q and actor π with weights θ^Q and θ^π .
Initialize target network Q' and π' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\pi'} \leftarrow \theta^\pi$
Initialize replay buffer R
for episode = 1, E **do**
 Initialize a random process $\mathcal{N}$ for action exploration
 Receive initial observation state s_1
 for $t = 1, T$ **do**
 Select action $a_t = \pi(s_t | \theta^\pi) + \mathcal{N}_t$ according to the current policy and exploration noise
 Execute action a_t and observe reward r_t and new state s_{t+1}
 Store transition (s_t, a_t, r_t, s_{t+1}) in R
 Sample a random mini-batch of M transitions from R :
 $\mathcal{M} = \{(s_i, a_i, r_i, s_{i+1}): i = [1, 2, \dots, M]\}$
 Sample a random mini-batch of K transitions from D :
 $\mathcal{K} = \{(s_i, a_i, r_i, s_{i+1}): i = [1, 2, \dots, K]\}$
 Update critic as follows:
 with $(s_i, a_i, r_i, s_{i+1}) \in \mathcal{M} \cup \mathcal{K}$:
 Calculate $y_i = r_i + \gamma Q'(s_{i+1}, \pi'(s_{i+1} | \theta^{\pi'})) | \theta^{Q'}$
 Update critic by minimizing loss $L = \frac{1}{(N+K)} \sum_i (y_i - Q(s_i, a_i | \theta^Q))^2$
 Update actor as follows:
 with $(s_i, a_i, r_i, s_{i+1}) \in \mathcal{M} \cup \mathcal{K}$:
 Calculate policy gradient

$$\nabla_{\theta^\pi} J = \frac{1}{(M+K)} \sum_i \nabla_a Q(s, a | \theta^Q) |_{s=s_i, a=\pi(s_i)} \nabla_{\theta^\pi} \pi(s | \theta^\pi) |_{s=s_i}$$

 with $(s_i, a_i, r_i, s_{i+1}) \in \mathcal{K}$:
 Calculate behavior cloning gradient

$$\nabla_{\theta^\pi} L_{BC} = \frac{1}{K} \sum_i \nabla_{\theta^\pi} \|\pi(s_i | \theta^\pi) - a_i\|^2$$

 Update actor policy using gradient: $\beta_J \nabla_{\theta^\pi} J + \beta_{BC} \nabla_{\theta^\pi} L_{BC}$
 Update target networks:

$$\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}$$

$$\theta^{\pi'} \leftarrow \tau \theta^\pi + (1 - \tau) \theta^{\pi'}$$

 end for
end for

Accordingly, small changes in S_{ij} , P_{ij} , and Q_{ij} are related to one another as follows:

$$S_{ij}\Delta S_{ij} = P_{ij}\Delta P_{ij} + Q_{ij}\Delta Q_{ij} \quad (16)$$

P_{ij} and Q_{ij} are both functions of buses' i and j voltage magnitudes (V_i , and V_j) and voltage angles (δ_i , and δ_j). Accordingly, changes in real and reactive power flows can be related to changes in voltage magnitudes and angles as follows:

$$\begin{bmatrix} \Delta P_{ij} \\ \Delta Q_{ij} \end{bmatrix} = \begin{bmatrix} \frac{\partial P_{ij}}{\partial \delta_i} & \frac{\partial P_{ij}}{\partial \delta_j} \\ \frac{\partial Q_{ij}}{\partial \delta_i} & \frac{\partial Q_{ij}}{\partial \delta_j} \end{bmatrix} \begin{bmatrix} \Delta \delta_i \\ \Delta \delta_j \end{bmatrix} + \begin{bmatrix} \frac{\partial P_{ij}}{\partial V_i} & \frac{\partial P_{ij}}{\partial V_j} \\ \frac{\partial Q_{ij}}{\partial V_i} & \frac{\partial Q_{ij}}{\partial V_j} \end{bmatrix} \begin{bmatrix} \Delta V_i \\ \Delta V_j \end{bmatrix} \quad (17)$$

Changes in voltage magnitudes and angles $\Delta \delta_i, \Delta \delta_j, \Delta V_i$, and ΔV_j are related to changes in real power generation $\Delta P_{g1}, \Delta P_{g2} \dots \Delta P_{gG}$ through the inverse of the system Jacobian matrix. The inverse Jacobian matrix equations can be written as follows:

$$\begin{bmatrix} \Delta \delta \\ \Delta V \end{bmatrix} = \begin{bmatrix} F & M \\ H & N \end{bmatrix} \begin{bmatrix} \Delta P_{inj} \\ \Delta Q_{inj} \end{bmatrix} \quad (18)$$

Where ΔP_{inj} and ΔQ_{inj} are the vectors of the change in real and reactive power injections. Now, allowing only the real power generation to change, then, one may write, for any bus k :

$$\begin{aligned} \Delta \delta_k &= \sum_{n \in G} f_{kn} \Delta P_{gk} \\ \Delta V_k &= \sum_{n \in G} h_{kn} \Delta P_{gk} \end{aligned} \quad (19)$$

From the above relations, one can express the sensitivity of MVA flows with respect to changes in real power generation as follows:

$$\Delta S_{ij} = \frac{1}{S_{ij}} \left(\begin{bmatrix} P_{ij} & Q_{ij} \end{bmatrix} \begin{bmatrix} \frac{\partial P_{ij}}{\partial \delta_i} & \frac{\partial P_{ij}}{\partial \delta_j} \\ \frac{\partial Q_{ij}}{\partial \delta_i} & \frac{\partial Q_{ij}}{\partial \delta_j} \end{bmatrix} \begin{bmatrix} F_i \\ F_j \end{bmatrix} + \begin{bmatrix} \frac{\partial P_{ij}}{\partial V_i} & \frac{\partial P_{ij}}{\partial V_j} \\ \frac{\partial Q_{ij}}{\partial V_i} & \frac{\partial Q_{ij}}{\partial V_j} \end{bmatrix} \begin{bmatrix} H_i \\ H_j \end{bmatrix} \right) [\Delta P_G] \quad (20)$$

Starting from a given initial state (i.e., a state where one (or more) of the system lines is overloaded), one can obtain a demonstration as follows: 1) solve the quadratic optimization problem detailed in (21) to obtain an estimate on the total required change in real power for the system to become secure. 2) If a solution ΔP_g^* to (21) exists, then, add ΔP_g^* to the initial generation and solve an AC power flow to verify that ΔP_g^* is indeed a valid solution. 3) Determine an appropriate sequence of actions $\Delta P_{G-t1}, \Delta P_{G-t2} \dots$ such that at each control step t , the taken action ΔP_{G-t} does not result in overloading healthy lines and/or increasing the flow on an already overloaded line.

$$\min \left(f = \sum_j^G (\Delta P_{gj})^2 \right) \quad (21)$$

s.t.

$$S_{ij} + \Delta S_{ij} \leq S_{ij-limit}$$

The above optimization problem has a convex quadratic objective function and a linear set of constraints. This makes (21) a convex quadratic program that can easily be solved by several methods such as those detailed in [45].

3.5 Simulation Studies

This section presents a set of simulation studies that investigates and compares the performance of the proposed hybrid method to that of the data-driven DDPG method. The investigation studies were carried out on both the IEEE 118-bus (18 generators and 186 branches) [25] and a larger 2749-bus real system (377 generators and 3125 branches). For each system, the MAV flow limit of each

branch is taken as 150% of the maximum flow recorded among 5000 different operating scenarios. For limits on ΔP_g , i.e., ΔP_{g-min} and ΔP_{g-max} , those are assumed to be $\pm 10\%$ of each unit's maximum capacity. The experimental setup and obtained results are presented and discussed next.

3.5.1 Experimental Setup

a) Actor and critic function design

The actor neural network is configured to have an input layer, 2 hidden layers, and an output layer. For the two hidden layers, a leaky rectified non-linearity activation function is used. On the other hand, a tanh activation function is used for the output layer. The critic network is also configured to have a similar structure: an input layer, 2 hidden layers, and an output layer. However, no activation function is used for the critic's output layer. Batch normalization is applied to the input of all layers with non-linear activation functions. The number of neurons of the hidden layers is set differently for the agents of each test system. For the 118-bus system agents, the 2 hidden layers of the actor and critic networks have 600 and 400 neurons respectively. For the 2749-bus system agents, the 2 hidden layers had 1000 and 600 neurons.

b) Training parameters

The following settings are common to both the data-driven DDPG method and the hybrid method: a learning rate of 0.0001 and 0.001 for the actor and critic neural networks respectively; a discount factor of $\gamma = 0.99$; a target update factor of $\tau = 0.001$; a replay buffer size of 10^5 ; and a mini-batch size of 512. A Gaussian noise with a standard deviation of 0.3 was used for exploring the environment. Exploration is performed with a probability of 0.1 at every step.

For the hybrid method, an additional buffer is used to hold the set of demonstrated transitions. $N_D = 64$, i.e., out of the 512 samples of the mini-batch, 64 are to be sampled from the demonstration

buffer. β_J and β_{BC} are set differently for each test system. For the IEEE 118-bus system, β_J and β_{BC} are initially set to 1 and 10 respectively. The value of β_{BC} was exponentially decayed by a factor of 10^{-4} at each time step ($\beta_{BC,t} = 10^{-4} \times \beta_{BC,t-1}$). Once β_{BC} has reached a near-zero value, then, behavior cloning is disabled and the agent continuous training based on the objective of maximizing the Q-value. For the 2749-bus system β_J and β_{BC} are set to 1 and 10 respectively. However, the annealing of behavior cloning has led to a failure in training. Best training performance was achieved by keeping β_{BC} fixed at 10. In other words, the objective of behavior cloning is maintained throughout the entire training period. By doing so, the agent is bounded to remember and maintain a policy that is close to the one implied in the observations.

Finally, DDPG uses the reward function given in (10) with $r_{TML} = \pm 50$. The hybrid method uses the reward function of (14) with $r_{TML} = \pm 1$.

c) Training setup

In this work, the power system environment is built in MATLAB. The RL agent is built in Tensorflow. Python is used to facilitate the exchange of signals (system states, agent actions, and rewards) between the power system environment and the RL.

Starting from a given initial state, the RL agent is allowed to interact with the system environment for a maximum number of 20 time-steps. The set of initial states is generated using Matpower. To generate these initial states the following steps are performed: 1) set the load at each bus at a random value between 70% and 130% of its base-case value. Set the power factor at each load bus randomly between 95% and 105% of its base-case value. Set the output power of each generator at a random value within 70% to 130% of its base-case value. 2) Take one of the system branches out (randomly) and attempt to solve for the system states. If the created case had no

solution, then, discard it. Otherwise, 3) calculate the MVA flow through all branches of the system. If any of these flows exceed their corresponding limit, proceed to the next step. Otherwise, discard the created case. 4) Verify that the flow on overloaded branches can be decreased down to a secure level. This can be done using the proposed method of Subsection 3.4.3. If all steps were carried out successfully, then, the created case is saved and used as an initial state for the training of the RL agent.

3.5.2 Agents' Training

a) IEEE 118-bus system

The IEEE 118-bus system has 54 units and 186 branches. The 54 units are composed of: a slack unit, 18 generators, and 35 synchronous condensers. Accordingly, the total number of states for its RL agent becomes 205 (186 measurements of branch MVA flows; 18 measurements of generators' output power; and a single measurement of the slack unit power).

The number of initial cases used for training the RL-agents is 10,000 initial cases. The training of the RL agents was carried out over 80,000 episodes on a personal desktop (Intel^R CoreTM i7 2.5 GHz, 16 GB RAM). The time taken to complete the training was around 12 hours per agent. The training performance of the RL-agents is shown in Figure 3-1. The displayed success rate is calculated as the moving average of the number of succeeded cases over a window of 1000 episodes.

As can be seen from the results of Figure 3-1, when using the data-driven DDPG method, the RL-agent was able to achieve an averaged ~96% success rate towards the end of the training period. On the other hand, when using the proposed hybrid method, the RL-agent was able to achieve a higher success rate of ~98%.

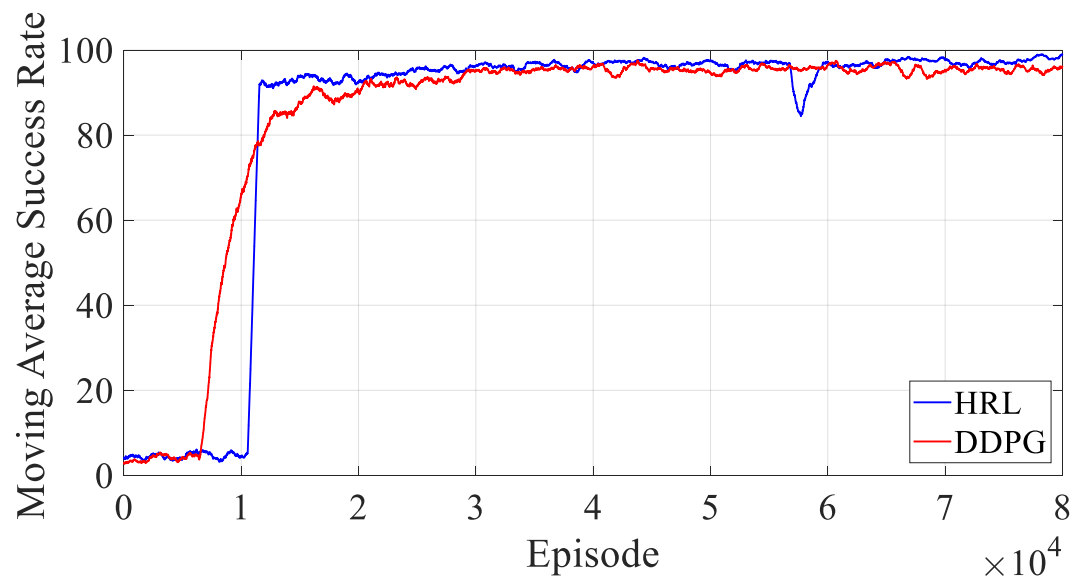


Figure 3-1 IEEE 118-bus system: agents' training performance.

b) 2749-bus system

The 2749-bus system has 377 units and 3125 branches. Out of the 377 units, 342 are available for control. The training of the RL-agents of the 2749-bus system was also carried out using a pool of 10,000 initial cases all of which had a feasible solution. The training of the RL agents was carried out over 100,000 episodes. The time taken to complete the training was around 13 hours and 58 hours for the data-driven and hybrid methods respectively. The training results are presented in Figure 3-2. As can be seen from this figure, the training of the RL-agent was successful only when using the hybrid method.

Even though the hybrid method uses an uninformative intermediate reward ($r_{IM,t} = 0$), it was still able to achieve a better performance when compared to the data-driven DDPG method. This clearly shows the advantage of using model-driven actions (i.e., actions obtained based on generator shifting factors) as an aid to the data-driven training process of RL-agents.

3.5.3 Agents' Performance Testing

The performance of the trained RL-agents was tested using a different set of 5,000 initial cases. Each one of those initial cases is ensured to have at least one valid sequence of actions that can safely transition the system into a secured state.

a) IEEE 118-bus system

The results of this test are shown in Figure 3-3 and Figure 3-4 and are summarized as follows: the agent trained with the proposed hybrid approach achieved a higher success rate of 97.6% compared to 95.5% for the agent trained with DDPG.

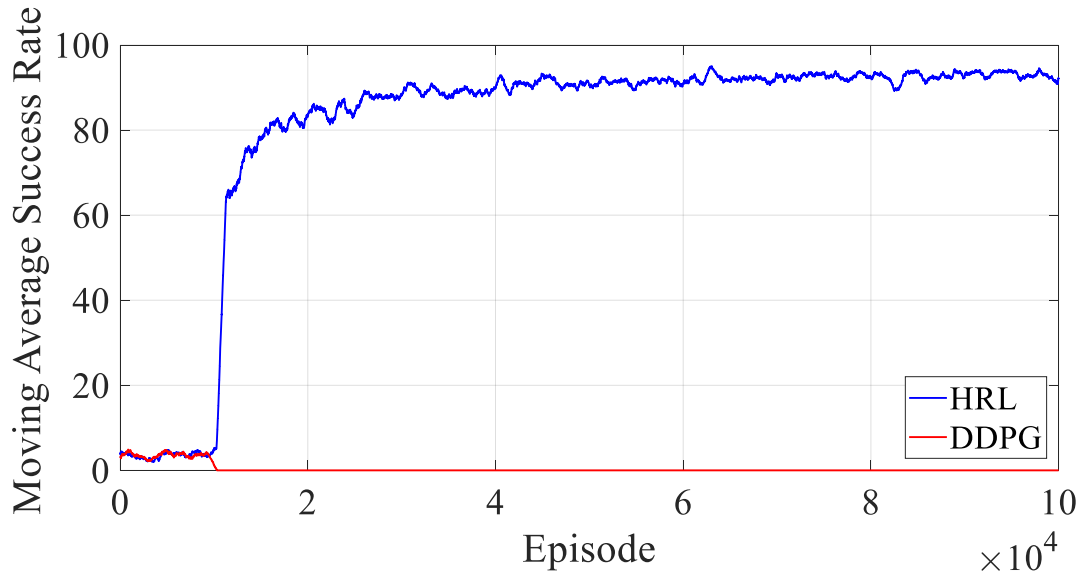


Figure 3-2 2749-bus system: agents' training performance.

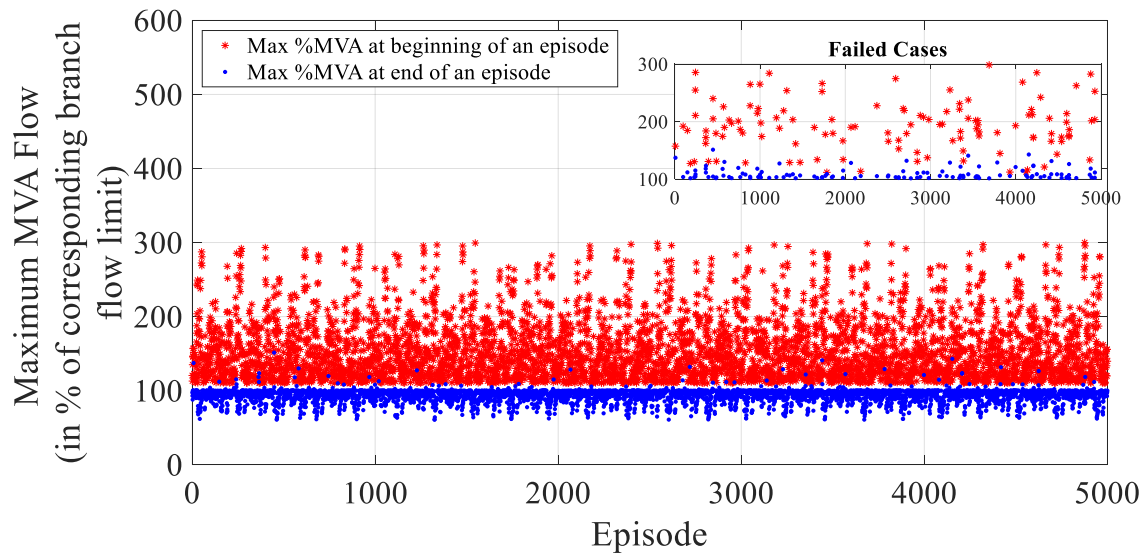


Figure 3-3 IEEE 118-bus system: DDPG agent's performance testing results. Maximum percentage MVA flow at the beginning and end of each tested episode.

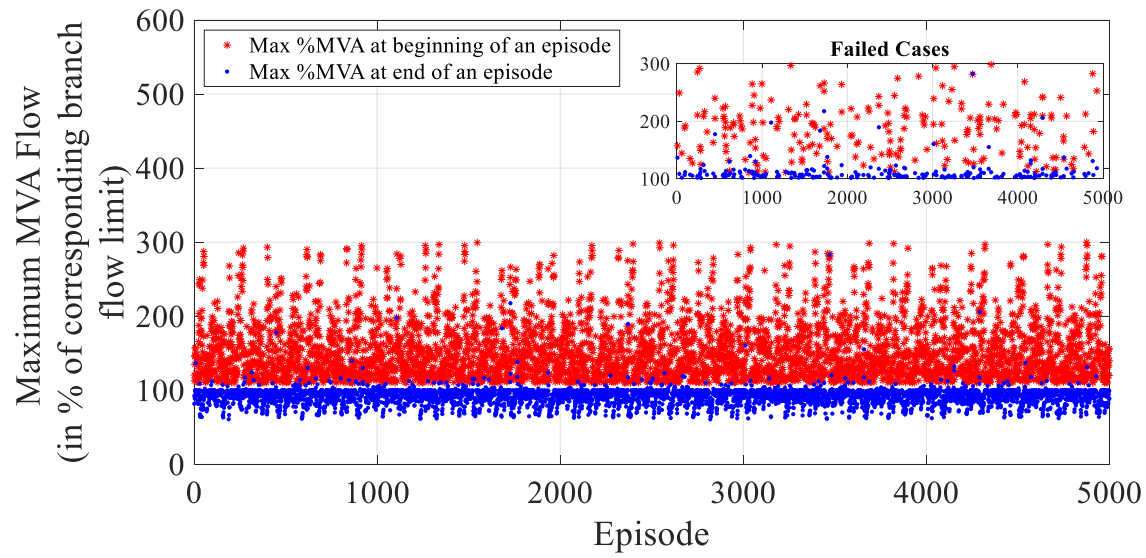


Figure 3-4 IEEE 118-bus system: HRL agent's performance testing results. Maximum percentage MVA flow at the beginning and end of each tested episode.

With RL-based control, one would not know the end results of the agent actions beforehand, i.e., whether the agent would succeed or fail in securing the system. Now, an episode is labeled failed if the agent takes an action that would either overload some of the healthy lines, increase the flow on some of the already overloaded lines, or most importantly collapse the system. Therefore, it is of great importance to investigate the performance of the agents for those cases where an agent has failed. Table 3-2 displays the number of failed cases where the agent has driven the system towards a better loading condition, a worse loading condition, or a collapse.

Here, we define a “better” loading condition as one where: a) the maximum percentage MVA flow at the end of the episode is less than that at the beginning of the episode; and b) the number of overloaded lines at the end of the episode is less or equal to that at the beginning of the episode. A “worse” loading condition would be one where either a) or b) is not true.

From the results shown in Figure 3-3 and Figure 3-4, and those displayed in Table 3-2, it is evident that the agents have learned a policy that drives the system towards a better operating condition.

b) 2749-bus system

For the 2749-bus system, the DDPG agent has failed to solve any of the 5,000 test cases. On the other hand, the agent trained with the proposed hybrid approach was able to solve 91.3% of them. Table 3-3 displays the number of failed cases where the HRL agent has driven the system towards a better or worse loading condition, or a collapse. Based on the results of Figure 3-5 and Table 3-3, it can be concluded that the learned policy is one that secures or at least transitions the system into a better operating condition.

Table 3-2 IEEE 118-bus system: analysis of agents' performance when failed to secure the system

RL-agent	The number of failed cases where the agent has driven the system towards:		
	better loading condition	worse loading condition	system collapse
DDPG-based agent	202	25	0
HRL-based agent	98	24	0

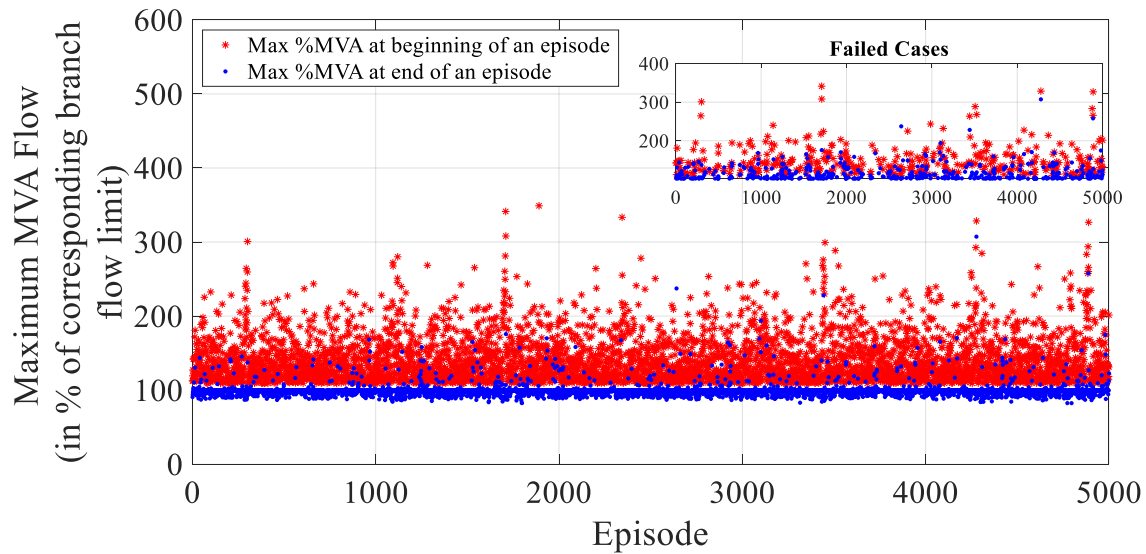


Figure 3-5 2749-bus system: HRL agent's performance testing results. Maximum percentage MVA flow at the beginning and end of each tested episode.

Table 3-3 2749-bus system: analysis of agents' performance when failed to secure the system

RL-agent	The number of failed cases where the agent has driven the system towards:		
	better loading condition	worse loading condition	system collapse
HRL-based agent	287	148	0

In conclusion, based on the results of both test systems, and especially the 2749-bus system, it can be asserted that our proposed HRL approach achieves good results and significantly outperforms the data-driven DDPG-based control.

3.6 On the Training of 2749-bus HRL Agent

The implemented DDPG-based hybrid learning algorithm has many design aspects and hyper-parameters to be defined, set, and tuned. Those are:

- Structure of RL agent: the number of hidden layers and the number of neurons in each hidden layer.
- Tunable parameters: actor and critic learning rates; gamma; replay buffer size; mini-batch size; β_J and β_{BC} .
- Training techniques: optimizer, batch normalization, weights regularization, etc.

As mentioned before, we adopted a manual tuning (trial and error) approach to arrive at a “suitable” (but not necessarily optimal) set of hyper-parameters. We started with a reference design and set of parameters that are presented in the initial DDPG paper [36]. Then, we further tuned the training parameters following some general guidelines and findings of several works in the literature.

In Section 3.5, we demonstrated the training performance for the HRL agents for both the IEEE 118-bus system and the larger 2749-bus system. There, we stated that for the 2749-bus system annealing of behavior cloning has led to a failure in training. Best training performance was achieved by keeping β_{BC} fixed. In other words, the objective of behavior cloning is maintained throughout the entire training period. A similar approach was implemented in [46]. However, by doing so, the agent is bounded to remember and maintain a policy that is close to the one implied

in the observations. This might have prevented the agent from finding a more “optimal” policy, had one existed.

Now, why did the HRL agent fail once behavior cloning is annealed? Is it a matter of tuning training parameters (learning rate, buffer size, mini-batch size, and/or gamma)? Is it batch normalization? Weight regularization? Maybe it is a result of unintended interactions between some of the utilized training techniques. Or maybe it is just a bug in implementation? Indeed, many works in the literature attempt to investigate the limitations of the various techniques involved in training a neural network, however, most of these works are presented for supervised learning. In addition, these investigations are carried out on different benchmarks and problems of different sizes and nature. Therefore, the observed behaviors might well be dependent on the problem definition rather than being a general truth. Before we continue with our discussion, let’s first introduce the concepts behind batch normalization and weight regularization.

Batch normalization is a technique used to stabilize and speed up the training of deep neural networks [47]. Before batch normalization, the convergence of deep NNs was highly sensitive to parameter initialization and usually required the use of low learning rates. The authors of [47] have identified a potential reason for that: *“Training Deep Neural Networks is complicated by the fact that the distribution of each layer’s inputs changes during training, as the parameters of the previous layers change”*. Accordingly, they proposed batch normalization, a technique that stabilizes the distribution of internal activations during training. As the name suggests, batch normalization achieves this by using the mean μ and standard deviation σ of the mini-batch to standardize (normalize) the input to a given layer. This normalization makes the model invariant to the scale of its weights. This invariance reduces the sensitivity to initialization and makes it possible to use higher learning rates.

On the other hand, regularization is a technique used to prevent overfitting and therefore, provide for a better generalization performance of neural networks. There are several types of regularization. The most common ones are weight decay and L2-regularization. Here, we utilize L2-regularization in which the L2-norm of model weights is added as a penalty to the learning objective of the network.

Since batch normalization makes the model invariant to the scale of weights, then, for networks with batch normalization, one would NOT expect an L2-penalty to constrain the representation of the neural network. However, in [48], the authors observed that L2-regularization still exerts a regularization effect on the neural network, however, due to a different mechanism. In a nutshell, their investigations revealed the following: *“reducing the norm of the weights would increase the effective learning rate. Higher learning rates typically lead to larger gradient noise, which has been shown to act as a stochastic regularizer”* Therefore, on networks with batch normalization, L2-regularization exerts a regularizing effect “indirectly” through the effective learning rate. This was also observed in [49].

In addition, for networks with batch normalization, if no regularization is used, then the norm of the weights tends to increase over time, and so the effective learning rate decreases. In supervised learning, decreasing the effective learning rate is often desirable. However, for reinforcement learning, since learning is continued over long periods of time, such an effect on the learning rate is not desired.

Now, back to our question: why did the RL agent of the 2749-bus system fail once demonstrations were taken out? After investigating the agent training performance for a large number of different training scenarios, we can hypothesize about the reason why the RL agent has

failed. In a nutshell, it was due to unintended effects of batch normalization which were made worse by behavior cloning. To further explain we refer readers back to the mean squared Bellman error (MSBE) of (2) and (3). Now, the critic network is updated according to this MSBE. There are a number of factors that affect the behavior of this estimation error. First, since we are using function approximators (i.e. neural networks) to model the critic, then, as stated by the authors of [50], “*the Bellman equation is never exactly satisfied and each update leaves some amount of residual error*”. Second, using a target critic network which is a delayed copy of the critic network Q . This would create some bias in the critic’s estimations. Finally, using batch normalization. Now, an update of the critic function requires the calculation of $Q(s_t, a_t)$ and $Q'(s_{t+1}, a'_{t+1} = \pi'(s_{t+1}))$. The actions a_t and a'_{t+1} would have different distributions: a_t is either a demonstrated action or an action that was produced by previous policy iterations (i.e., sampled from the reply buffer). On the other hand, a'_{t+1} is produced by the current target policy $\pi'(s_{t+1})$. When batch normalization is used, then, $Q(s_t, a_t)$ and $Q'(s_{t+1}, a'_{t+1} = \pi'(s_{t+1}))$ would be calculated using different statistical features and therefore, their subtraction would result in a larger estimation error [51]. With behavior cloning, $\pi'(s_{t+1})$ changes fast. In addition, $\pi'(s_{t+1})$ is calculated using mini-batch statistics (BN set to training mode) while a_t was produced using moving statistics (BN set to evaluation mode). These discrepancies further worsen the difference in statistical features of Q and Q' . Now, according to [50], “*if the estimation error is not tamed, then, the variance in Q can grow rapidly with each update*”. This might lead to suboptimal policies or even a divergent Q .

In addition, updating the actor network requires an inquiry of the critic network ($\nabla_a Q(s, a | \theta^Q)|_{s=s_t, a=\pi(s_t)}$). With batch normalization, the set $a = \pi(s_t)$ would have different statistics that those used to update the critic network, i.e., a_t and $a'_{t+1} = \pi'(s_{t+1})$. These statistical differences, exacerbated by behavior cloning, might lead to unstable learning.

In Figure 3-6 and Figure 3-7 we present the training performance of the HRL-agent for a set of 4 training scenarios: Those are:

1. BN is used for both actor and critic networks, a mini-batch size of 512, and a demonstration buffer size of 64. BN is set to training mode (i.e., mini-batch statistics are used to normalize inputs to activation layers).
2. BN is used for both actor and critic networks, a mini-batch size of 512, and a demonstration buffer size of 64. BN is set to evaluation mode (i.e., moving statistics are used to normalize inputs to activation layers).
3. BN is used for both actor and critic networks, a mini-batch size of 64, and a demonstration buffer size of 32. BN is set to evaluation mode.
4. BN is NOT used, a mini-batch size of 512, and a demonstration buffer size of 64.

The following parameter settings were kept the same for all studied scenarios: 600 and 400 neurons for the 2 hidden layers of the actor and critic networks; a learning rate of 0.0001 and 0.001 for the actor and critic networks respectively; a discount factor of $\gamma = 0.95$; a target update factor of $\tau = 0.001$; a replay buffer size of 105; 0.001 L2-regularization for both actor and critic networks; $\beta_J = 1$; $\beta_{BC} = 100$; β_{BC} is exponentially decayed by a factor of 10^{-3} . As for the exploration noise, a normal (Gaussian) distribution with a standard deviation of 0.3 was utilized. Exploration is performed with a probability of 0.1 at every step.

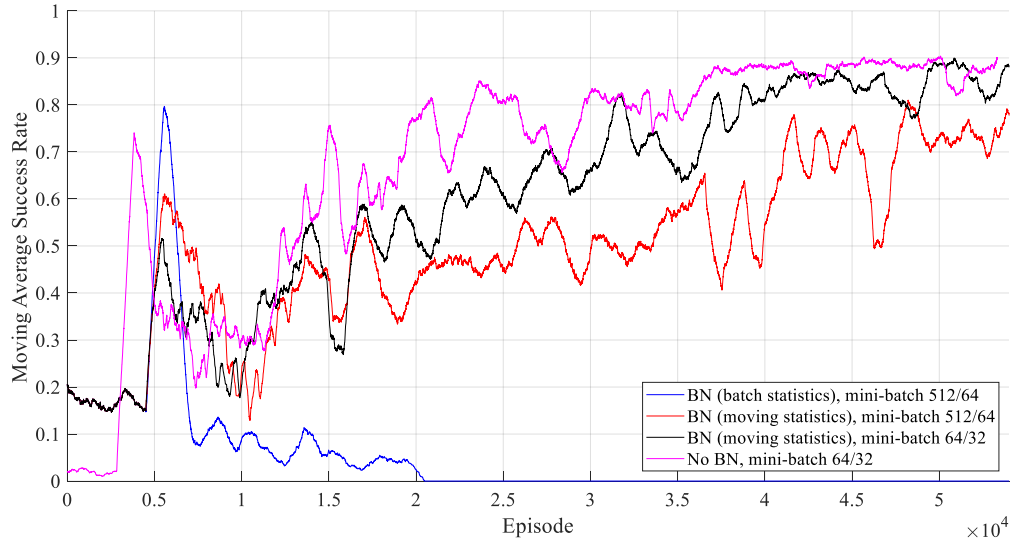


Figure 3-6 2749-bus system: HRL agent's training performance.Success rate.

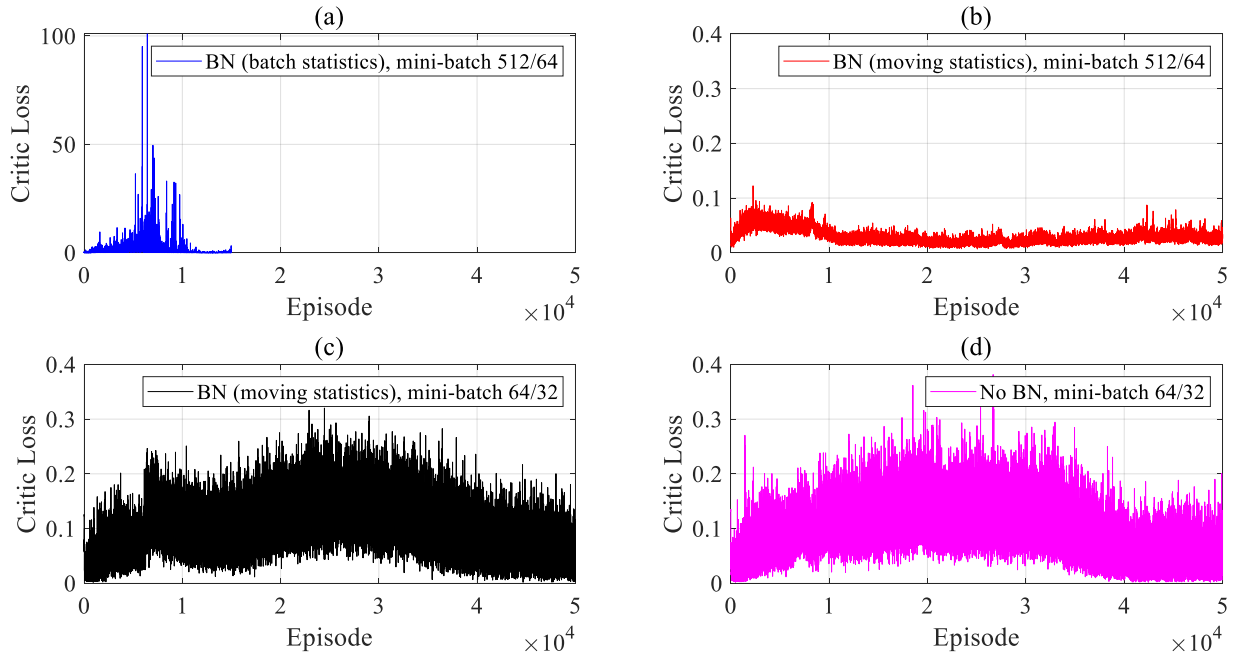


Figure 3-7 2749-bus system: HRL agent's training performance.Critic loss.

Figure 3-6 shows the success rate of 2749-bus system HRL-agent during training. This success rate is calculated as the moving average of the number of succeeded cases over a window of 1000 episodes. In addition, we show the critic loss during training in Figure 3-7. From Figure 3-7 (a) (critic loss shown by blue curve), it can clearly be seen that using batch normalization in training mode (i.e., activations are normalized using mini-batch statistics) has a negative impact on the training of the critic. As hypothesized above, the difference in statistical features of Q and Q' (made worse by behavior cloning) has led to a large critic loss and therefore, the agent failed. On the other hand, using moving statistics helped minimize the statistical discrepancies between sampled actions (a_t) and those produced by the target actor policy ($\pi'(s_{t+1})$). This hypothesis is further confirmed by the results of training the agent without batch normalization (scenario 4). As seen from Figure 3-6, and Figure 3-7 (d) (curves colored in magenta). When batch normalization is not used, then, once demonstrations were annealed out, the agent was successful in learning and picking up a policy on its own.

3.7 Conclusion

This work presented a data-driven and model-based hybrid reinforcement learning approach to the problem of branch overload relief in large power systems. A control agent is trained to change generators' real power output in order to adjust the power flow through the network so that branch overload is mitigated or minimized. The proposed approach was developed based on a series of investigations on why the data-driven DDPG-based approach fails when training the agent for large power systems. This hybrid approach utilizes generator shifting factor-driven actions (model-based actions) to help the agent learn an optimal control policy (data-driven actions). The performance of the proposed approach was tested and compared to the original DDPG-based approach on both the IEEE 118-bus and a larger 2749-bus real-world system. With the proposed approach, the agents

were able to achieve 97.6% and 91.3% success rates compared to 95.5% and 0% when trained according to the original data-driven DDPG algorithm. These results show that the proposed hybrid approach performs well for large power systems and that it is superior to the data-driven DDPG-based approach.

For the proposed HRL approach, demonstrations (model-based actions) might or might not be annealed while the agent is training. If not annealed, then the agent is bounded to remember and maintain a policy that is close to the one implied in the demonstrations. If annealed, then the agent is given the chance to converge onto a different, and maybe a better policy. Investigating the training performance of the HRL-agent on the larger 2749-bus system revealed that using batch normalization (a technique used for speeding up the training of neural networks) within the context of our proposed hybrid approach, might lead to unstable learning and a failure of HRL agent if and when demonstrations are annealed out.

Chapter 4. Power Systems Contingency Screening Using Convolutional Neural Networks

This chapter presents a Convolutional Neural Network (CNN) based approach that we proposed as a faster alternative to the classical AC power flow-based contingency screening. The proposed approach is investigated on both the IEEE 118-bus system and Texas 2000-bus synthetic system. For such large systems, the implementation of the proposed approach came with several challenges, such as computational burden, learning from imbalanced datasets, and performance evaluation of the trained models. Accordingly, this work contributes a set of novel techniques and best practices that enables both efficient and successful implementation of CNN-based multi-contingency classifiers for large power systems.

4.1 Introduction

Static contingency screening is a vital tool for assessing the security of power systems. It identifies the set of component outages, i.e., contingencies which would result in an insecure operation of the system, and therefore, power system planners and operators can prepare and plan ahead of time against such potential risks. Performing static contingency analysis on a power grid with M different components requires an M number of AC power flow runs. Each time, a single component is taken out and the system power flow equations are solved. The system is then evaluated to see whether or not any of its operational or equipment limits are violated. However, running an AC power flow for each contingency makes the overall contingency analysis process computationally expensive. Another factor that increases the computational burden of contingency analysis is the presence of intermittent renewable generation, e.g., wind and solar. In that case,

each contingency needs to be assessed for a number of different generation scenarios. Such computational burden might prevent the application of static contingency analysis in an online type frame of work. Several works have approached the problem of contingency screening (whether N-1 and/or N-2) using neural networks [52-55]. These works propose to utilize neural networks in place of the AC power flow solver. In other words, for a given contingency, instead of running generic AC power flow algorithms (e.g. Newton-Raphson power flow), one can use neural networks to quickly estimate system states (voltages, angles, and/or line flows) [52-54], or the ranking index itself [55]. Different from those works, we propose to utilize convolutional neural networks (CNN) as a faster alternative to the entire screening process, i.e., to obtain, in a single run, the security status of the system for all contingencies. Note that only branch-outage contingencies are considered in this work, i.e., transmission lines and transformers.

4.2 CNN-based Contingency Screening

4.2.1 CNNs for Power Systems

In recent years, neural networks, with all their different types but especially the feedforward neural networks (FNNs), have been extensively applied to many problems in the field of power systems including fault diagnosis [56, 57], stability analysis [58, 59], controllers' design [60], etc. In general, an FNN maps a number of input features to a given set of outputs. The selection of features depends on the application that the FNN is utilized for. Typically, domain experts would decide on what can be used as an input feature for the problem under consideration (e.g., wavelet energy and wavelet entropy of voltage and current signals [59]). In other words, the input features are hand-engineered.

On the other hand, in convolutional neural networks (CNNs), input features are not hand-engineered but rather extracted using convolutional filters on a set of input data (e.g. an image, a sequence of words, recordings, etc.) The extracted features are then fed to a fully connected neural network which is to learn the relation between the input (extracted features) and the output.

In general, convolutional neural networks are known to work well on structured data. This is because useful features can be identified based on local connectivity/correlations within the input dataset. There are three different types of data structures. Those are: 1) spatial structure (e.g. images [61], text [62], speech spectrum [63]); 2) temporal structure (e.g. musical compositions [64]); and 3) spatiotemporal structure (e.g. videos [65]).

In the literature of power systems, CNNs are typically used on data with temporal structure (e.g. load and wind power generation daily profiles [66]). To use CNNs on time series, first, one needs to transform the time series data into a 2D image. In [66], the authors achieve this by simply stacking the set of one-dimensional (1D) time series (one series per feature) side-by-side. Other works have utilized advanced time series encoding techniques such as wavelet transforms [54, 67] and Gramian Angular Field [67] to create a 2D image representation of the input data streams.

Few works have reported the use of CNNs on non-temporal power system data [54, 68]. In [68], the authors represent the information of a given N -bus power system as a 3-channeled $N \times N$ image (the channels carry information on real power, reactive power, and voltage magnitudes). The data of each channel is structured according to the adjacency matrix of the system. However, because the size of the input image depends on the number of system buses, the application of such an approach would be limited to relatively small power systems. In this work, we utilize this same approach and accordingly represent a given operating point as a 2-channeled $N \times N$ image. Here,

the channels carry information on voltage magnitudes and voltage angles. Different from [68], in this work, we go the extra mile of overcoming the implementation challenges and see the application of our CNN-based classifier to large power systems (systems as large as a 2000-bus system).

4.2.2 CNN-based Multi-contingency Classifiers

In this work, we propose to utilize a CNN-based classification model as a faster alternative to the classical AC power flow-based contingency analysis. The output of the CNN-based classifier is a vector whose elements correspond to the set (or sub-set) of all possible contingencies. In the literature, problems of this type are known as multi-label classification problems. In general, a multi-label classifier assigns an input instance with multiple labels simultaneously. For example, a given text document (input instance) can be associated with a range of topics (labels), such as entertainment, financial, political, international, and domestic [62]. Another example would be a piece of music that can be labeled as blues, sad, and guitar all at the same time [69]. Similarly, for our problem, a given operating point (input instance) can have a number of contingencies (labels) that would result in an insecure operation of the system. In this subsection, we present and briefly describe the different elements of a CNN-based multi-contingency classifier. Those are: CNN input, structure, and output. In addition, we provide a brief introduction to the various challenges encountered in training such classifiers.

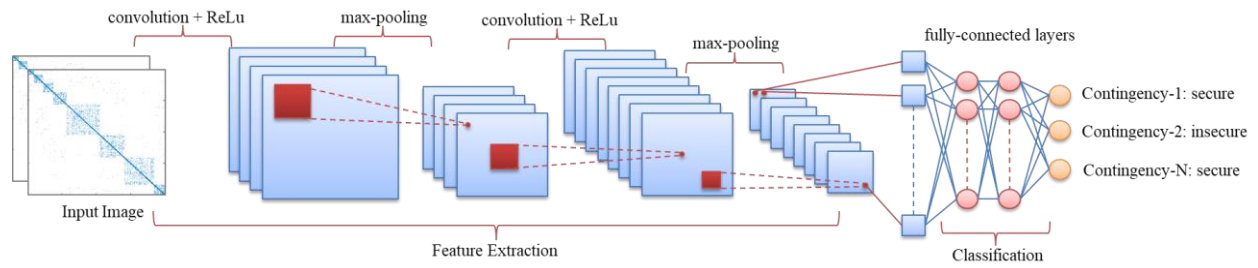


Figure 4-1 General structure of a CNN-based multi-contingency classifier.

a) CNN input

Given a general N-bus power system where $\mathcal{N}$ is the set of system buses, $\mathcal{L}$ is the set of transmission lines. Let V_i and δ_i represent the voltage magnitude and angle of bus i . V_{ij} and δ_{ij} represent the difference in voltage magnitude and voltage angle between buses i and j where $(i, j) \in \mathcal{L}$. Then, the information of a given operating point can be represented as a two-channel image of $N \times N$ pixels. The first channel contains voltage magnitudes (diagonal elements) and differences in voltage magnitudes between connected buses (off-diagonal elements). Similarly, the second channel contains bus voltage angles (diagonal elements) and differences in voltage angles between connected buses (off-diagonal elements). In general, when all lines are in service, the power system is expected to operate within its limits. Therefore, for each bus, one can use typical operational limits V_{i-min} , V_{i-max} , δ_{i-min} , and δ_{i-max} to normalize V_i and δ_i within the range [0, 1] as follows:

$$\bar{V}_i = \frac{V_i - V_{i-min}}{V_{i-max} - V_{i-min}} \quad \forall i \in \mathcal{N} \quad (1a)$$

$$\bar{\delta}_i = \frac{\delta_i - \delta_{i-min}}{\delta_{i-max} - \delta_{i-min}} \quad \forall i \in \mathcal{N} \quad (1b)$$

Where $\bar{(\cdot)}$ denotes a normalized quantity. Accordingly, the elements of X are defined as follows:

$$X(i, j, 1) = \begin{cases} \bar{V}_i & i = j \\ \bar{V}_i - \bar{V}_j & i \neq j, (i, j) \in \mathcal{L} \end{cases} \quad (2a)$$

$$X(i, j, 2) = \begin{cases} \bar{\delta}_i & i = j \\ \bar{\delta}_i - \bar{\delta}_j & i \neq j, (i, j) \in \mathcal{L} \end{cases} \quad (2b)$$

b) CNN output

The output of the CNN-based classifier is a vector whose elements correspond to the set (or sub-set) of all possible contingencies, i.e., output k represents the system status under contingency

k. Here, we utilize the ranking index (RI) detailed below in (3) to classify a given contingency as either secure, marginally-insecure, or insecure.

$$RI = \sum_{(i,j) \in \mathcal{L}} \frac{S_{ij} - S_{ij-alarm}}{S_{ij-max}} \quad (3)$$

Where: S_{ij} is the apparent power flow through line (i, j) , S_{ij-max} is the limit on apparent power flow for line (i, j) . $S_{ij-alarm}$ is the value of S_{ij} at which an alarm is raised for operators. Here, we set $S_{ij-alarm}$ equal to 120% of S_{ij-max} . Based on RI, a given contingency is classified as: 1) secure if $RI = 0$ which means all lines are loaded within their limits; 2) marginally-insecure if $RI < 1$ which is equivalent to a single line loaded in the range of 100 -to- 120% of its limit; or 3) insecure if $RI > 1$ which is equivalent to a single line loaded at more than 120% of its limit.

c) CNN structure

As shown in Figure 4-1, for a CNN-based classifier, first, features are extracted from the input image using a set of consecutive convolutional and max-pooling layers. Then, the extracted features are flattened and fed into a set of fully-connected layers.

1. Convolutional Layer: The convolution layer uses filters to extract patterns in the input image.

The convolution operation is defined as follows: given X , an input feature map of shape [input-height, input-width, number of input channels]; W , a convolutional filter of shape [filter-height, filter-width, filter-depth (= number of input channels), number of filters], and a layer bias b . Then, let convolving X with W result in a feature map Y of shape [output-height, output-width, number of output channels (= number of filters)]. Also, let X_h , X_w , and c represent input-height, input-width, and number of input channels. W_h , W_w and M represent filter-height, filter-width, and number of convolutional filters respectively. Y_h and Y_w represent output-height and output-width. Then, the elements of Y are calculated as follows:

$$Y_{(u, v, k)} = \sum_{i=1}^{W_h} \sum_{j=1}^{W_w} \sum_{d=1}^c X_{(u*s+i, v*s+j, d)} * W_{(i, j, d, k)} + b \quad (4)$$

2. **Max-pooling:** Pooling is an operation used to reduce the size of feature maps generated by convolutional layers. Pooling layers do not have trainable parameters (i.e. weights and biases). In general, there are two types of pooling operations: max, and average pooling. Here, we utilize max-pooling in which a max-filter is applied to sub-regions (usually non-overlapping) of the input feature map. The output is the maximum value from every sub-region covered by the max-filter.
3. **Flattening Layer:** Flattening layers connect convolutional or pooling layers to fully-connected layers. This type of layer reshapes the output of the last max-pooling layer into a 1D array per snapshot in order to be able to feed the first fully-connected
4. **Fully-Connected layers:** Fully-connected layers are the part of CNN-based classifier that learns the relation between the extracted features and the data labels. Typically, all fully-connected layers, except for the last one, use ReLU activation. The activation function of the output layer depends on the application of CNN (classification or regression). Here, we have three classes and therefore, we utilize a tanh activation function at the output layer of the fully-connected neural network. A tanh activation outputs a continuous value in the range of [-1, 1]. The model is trained to generate an output as close as possible to the actual label (-1, 0, or 1). The output is then rounded to the nearest integer. This approach has the advantage of creating a “distance” between secure (-1) and insecure classes (+1).

d) Expected challenges

As mentioned previously, the CNN-based contingency screening model is a multi-label classifier. In general, the majority of challenges encountered in training a multi-label classifier can be grouped under the umbrella of large-scale datasets. A dataset is considered large if it has a large number of training instances; a highly dimensioned input space; and/or a highly dimensioned output space. Large-scale datasets are inevitable in many real-world applications. Examples of large-scale datasets from the literature include the Ads-430K and Ads-1M. These datasets have a feature set size of 87,890 and 164,592; and a label set size of 434,594 and 1,082,898 respectively [70]. Large-scale datasets pose a number of challenges to multi-label learning. The three major ones are:

- 1. Implementation:** In general, the larger the input and/or output dimension, the larger and more complex the classifier model would be. This reflects on the computational burden of training the classifier which makes the search for “an” optimal set of model parameters (i.e., tuning) an expensive process.
- 2. Learning:** Large-scale data sets are often imbalanced. In the literature of single-label binary (or multi-class) classification, a dataset is considered imbalanced if a given class is encountered in less than 20% of the dataset. In reality, datasets can get far more imbalanced than this. In multi-label learning, data imbalance is encountered: 1) label-wise i.e., some labels are rarely encountered in the data set. For example, with reference to the problem of labeling a music piece, an electric guitar is rarely used in a music piece. 2) Instance-wise i.e., some labels’ compositions are rarely encountered. For example, a sad blues guitar music piece is rarely encountered in the dataset. Some examples of multi-label datasets with extreme imbalance ratios can be found in [71] and [72] which reports a maximum imbalance ratio (in terms of

samples per label combination) of 1:733 and 1:4,162 respectively. Learning from imbalanced datasets is hard and challenging. This is mainly due to:

- a. Standard loss functions used for training classifiers do not consider class distribution. Therefore, with highly imbalanced datasets, the classifier might simply learn to classify every label as its majority class.
- b. In general, relations governing classes of minorities are complex and typically hard to learn. The opposite is often true for classes of majorities.
- c. If classes of minorities have a small count in the training dataset, then, the classifier might simply consider them noise and ignore them.

Learning from imbalanced datasets has, and continues to be an active research topic. However, to this day, there has been no clear and definite answer to the question of how one can efficiently train a model on imbalanced datasets. Several approaches have been proposed to improve on the learning from imbalanced datasets, however, these are mainly proposed for single-label classification (binary or multi-class) [73]. Only a few are applicable to multi-label classification problems. Some of those are: adjusting the class weight (misclassification costs) [74] and adjusting the decision threshold [75, 76].

3. **Performance Evaluation:** Typically, the performance of a learning system is evaluated by a single-averaged metric. For multi-label multi-class classifiers, model performance can be averaged on the entire data-set (micro-averaged); or on each label (macro-averaged); or on each instance (instance-averaged). In general, for multi-label classification problems, micro-averaged metrics are of less meaning when it comes to evaluating the model performance. This is mainly due to:

- a. Dataset imbalance: when the dataset is imbalanced, then, the classifier simply learns to classify every contingency as its majority class. Now, if the dataset is dominated by a particular class, then, a micro-averaged metric would be high and thus, misleading.
- b. The classifier has multiple outputs: even if the dataset is perfectly balanced, having to classify more than one label makes performance evaluation a complicated task. To further explain, consider our CNN-based multi-contingency classifier and the following two cases: one instance (operating point) with three misclassified contingencies vs. three instances each with one misclassified contingency. Assuming misclassifications of the same type, then, the micro-averaged measure is the same for both cases even though they reflect different performances of the classifier.

Whether macro-averaged or instance-averaged is the better choice for evaluating model performance, depends on the nature of the classification problem

4.3 Training and Test Datasets

4.3.1 Data Generation Method

An instance corresponds to an operating point. To generate different operating points, one can randomly change the base-case: load real power, load power factor, generation real power, and generators voltage set-points. Given an N -bus system where $\mathcal{B}$ is the set of system buses; $\mathcal{G}$ is the set of generator buses; $\mathcal{L}$ is the set of load buses; P_L , and Q_L , are the vectors of load real and reactive power consumption; P_G , and V_G are the vectors of generation real power and voltage set-points. To create an operating point, each element of P_L and P_G is randomly set between 70% and 130% of its base-case value, each element of $P.F._L$ is randomly set between 95% and 105% of its base-case value. The elements of V_G are set randomly between V_{G-min} and V_{G-max} . Now, for large power systems, randomly setting each element of P_L , Q_L , P_G , and V_G means an infinite number of possible

operating points (i.e., a huge state-space). Indeed, with such flexibility, one can generate as many data points as needed. However, this opens the door for the following questions: “how to explore this huge input space and generate a dataset that offers useful information with respect to the learning objectives”. Typically, power systems are designed to be N-1 secure. Therefore, one would expect contingencies to “rarely” result in marginally-insecure and/or insecure status of the system. Now, since we are interested more in operating points with “insecure” and “marginally-insecure” classes, it is thought to devise a method that efficiently explores this huge input state-space and generates a dataset with minimal imbalance. Accordingly, we devise the following procedure for generating the dataset.

Step-1: In power systems community it is typical to assume: 1) a uniform change of system load, and 2) generation follows load. Accordingly, we propose to generate an “initial” set of operating points using only 2 control parameters λ_p and λ_{pf} . λ_p is a control parameter used to change both load and generation. λ_{pf} is a control parameter used to change the load power factor. To generate this initial set, we discretize λ_p and λ_{pf} and then generate operating points representing all possible combinations of λ_p and λ_{pf} . For each operating point, the system AC power flow is solved. If the solved case had any limit violations, then the created loading case is discarded. Otherwise, a full AC power flow-based contingency analysis is performed. Once the initial set (instances, labels) is generated, then go to Step-2.

Step-2: Identify contingencies with minority classes. For a given contingency C_j , a class k is considered a minority if its percentage count (denoted cnt_j^k) in the dataset is less than a threshold (e.g. $\text{cnt}_{\text{Th}}=10\%$). It should be noted that a contingency C_j might have one or two classes of minority. Accordingly,

$$C = \left\{ C_j : \min_{\forall k \in \{+1, 0, -1\}} \left(\frac{\text{nnz}(Y_j^k)}{|Y_j|} \right) \leq \text{cnt}_{\text{Th}} \right\} \quad (5)$$

Step-3: For each contingency C_j in C , determine its minority classes

$$k_j^{\text{minority}} = \left\{ k : \frac{\text{nnz}(Y_j^k)}{|Y_j|} \leq \text{cnt}_{\text{Th}}, \forall k \in \{+1, 0, -1\} \right\} \quad (6)$$

Step-4: For each contingency C_j in C , collect the samples (instances) where C_j was classified as a minority.

$$S_j = \{x_i : y_{ij} \in k_j^{\text{minority}}\} \quad (7)$$

Step-5: For each contingency C_j in C , randomly select a sample x_i from S_j , create a synthesized operating point x'_i around x_i in an attempt to generate another, yet different, operating point where contingency C_j is likely to be classified as a minority $y'_{ij} \in c_j^{\text{minority}}$. If the created operating point results in C_j (or at least one other contingency from C) being classified into its minority class(s), then keep x'_i otherwise, discard. Perform Step-6 and then proceed to the next contingency.

To synthesize around a given operating point, then:

$$\begin{aligned} P'_{L,k} &= P_{L,k} \pm n_k \Delta \lambda_P P_{L,k}^0 & \forall k \in \mathcal{L} \\ P \cdot F'_{L,k} &= P \cdot F_{L,k} \pm n_k \Delta \lambda_{pf} P \cdot F_{L,k}^0 & \forall k \in \mathcal{L} \\ P'_{G,k} &= P_{G,k} \pm n_k \Delta \lambda_P P_{G,k}^0 & \forall k \in \mathcal{G} \\ V'_{G,k} &= V_{G,k} \pm n_k \Delta \lambda_V V_{G,k}^0 & \forall k \in \mathcal{G} \end{aligned} \quad (8)$$

Where: $(.)^0$ denotes a base-case value, $(.)'$ denotes a synthesized value, n is “a” number of steps up (or down) the scale of the control parameter $\lambda_{(.)}$. The value of n is randomly selected out of the set $[n_{\min}, n_{\min} + 1, \dots, n_{\max}]$. $n_{\min}$ and $n_{\max}$ are parameters set by the user.

Step-6: To enable random exploration of input-space, then, with a probability of 20%, generate a data instance in a completely random manner, i.e., randomly set each element of PL and PG between [70% -130%] of its base-case value. Randomly set each element of P.F.L between [95% and 105%] of its base-case value. Randomly set each element of VG between $V_{G-\min}$ - $V_{G-\max}$.

Step-7: Repeat Steps 2-5 until the desired number of data points is reached.

4.3.2 Application to Test-Systems

In this work, two test systems were utilized, those are the IEEE 118-bus system and the Texas 2000-bus synthetic system. For each system, a dataset of 12500 instances is generated. These datasets are split into training and testing datasets using a 0.8:0.2 splitting ratio.

a) 118-bus system [25]:

The IEEE 118-bus system has 19 generation units, 35 synchronous condensers, 186 branches (177 transmission lines and 9 transformers), and 91 load buses. The published data-case does not provide MVA limits on branch flows. Therefore, in this work, the flow limit for each branch is taken as 150% of the maximum flow recorded among 5000 different loading scenarios for the intact system (i.e. all branches are in-service).

1. Data Generation: The 118-bus system has a set of 175 possible branch outage contingencies (i.e., branch outages NOT leading to islanding). Each contingency can be classified as either secure, marginally-insecure, or insecure. This means a set of 3^{175} possible outcomes. In general, datasets for problems with large output dimensions are highly imbalanced. Indeed, due to the characteristics and real-world operation of a power system, one would encounter a much less number of outcomes, however, the “actual” output-space remains high. Figure 4-2 visualize the imbalance of the 118-bus system dataset. This figure displays, for each

contingency, the number of samples where the given contingency is secure, marginally-insecure, or insecure. As seen from Figure 4-2, the dataset is highly imbalanced. Table 4-1 gives a summary of statistics for the generated dataset.

2. **Train/Test split:** In the literature of classification, the method of “stratification” is often utilized for splitting datasets that are highly imbalanced [77]. For single-label, binary (or multi-class) classification, stratification would split the dataset into a number of subsets (i.e., training, testing, and/or validation) so that the proportion of samples of **each class** in each subset is approximately equal to that of the complete dataset. For multi-label classification, stratification is typically utilized to split the dataset so that the proportion of samples of **each label combination** in each subset is approximately the same across all datasets. As seen from Table 4-1, the number of unique label combinations is 8335 which is high and very close to the number of samples of the dataset. This makes the application of stratification (as defined above) not feasible since many label combinations have only one sample. Therefore, we follow the approach of [77], which considers a more relaxed interpretation of stratification and split the dataset with the objective of maintaining an equal distribution of each class of each label in each subset. In Figure 4-3 and Figure 4-4, we show the distribution of each class for each contingency in the training and test datasets. Compared to Figure 4-2, one can see that the training and testing sub-sets have similar distributions to one another and to that of the whole dataset.

Table 4-1 IEEE 118-bus system: statistics of the generated dataset

# of contingencies	# of samples	# of unique label combinations	Samples per label combination			Samples per label's class				
			min	average	max			min	average	max
175	12500	8335	1	45	202	Class	-1	425	10,440	12,499
							0	2	8,231	9,749
							+1	1	1,699	12,499

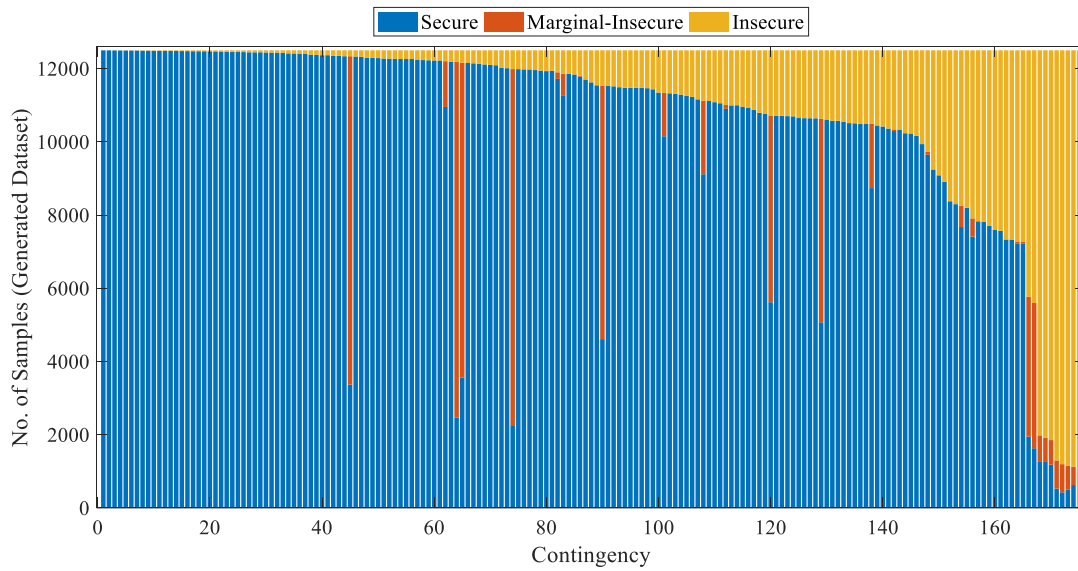


Figure 4-2 IEEE 118-bus system: imbalance of label dataset. The total number of contingencies is 175.

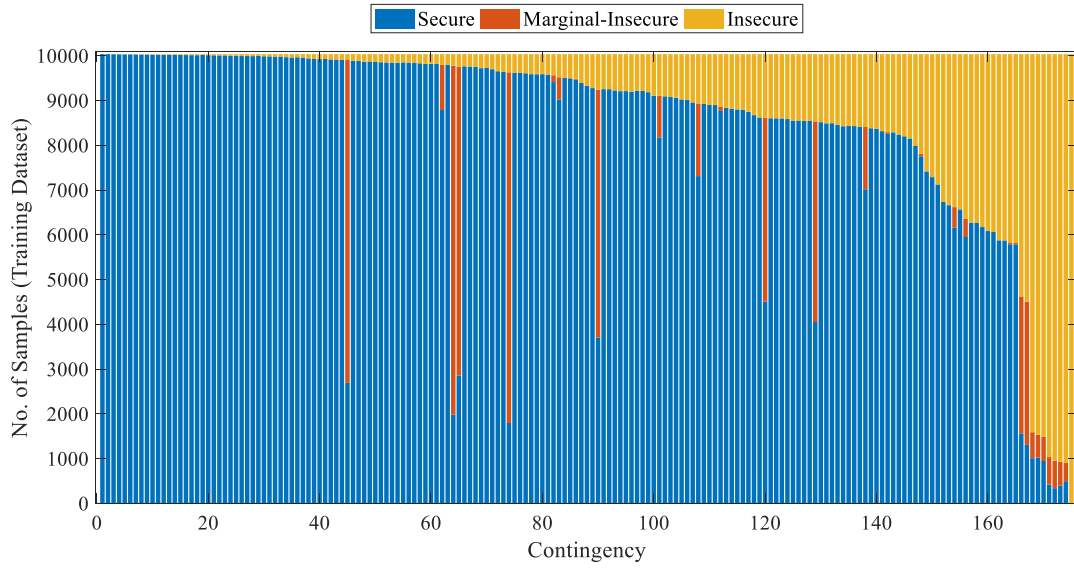


Figure 4-3 IEEE 118-bus system: imbalance of label training dataset. The total number of contingencies is 175.

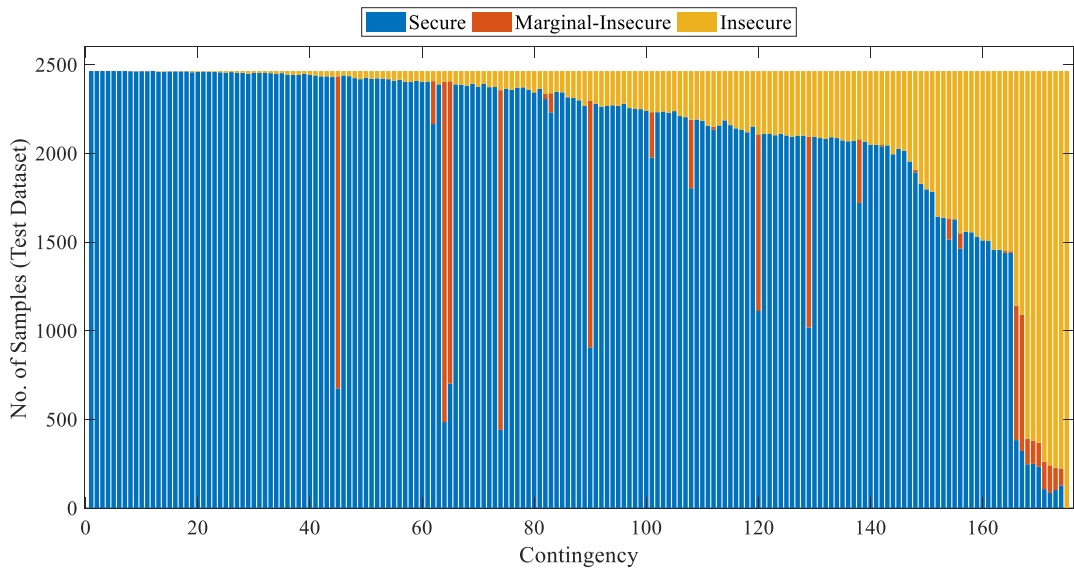


Figure 4-4 IEEE 118-bus system: imbalance of label test dataset. The total number of contingencies is 175.

b) Texas 2000-bus synthetic power system [78]

This is an entirely synthetic 2000-bus case. This 2000-bus case was designed by algorithms described in [78] to be statistically similar to the part of the U.S. state of Texas served by the Electric Reliability Council of Texas (ERCOT). The case is designed with a 115, 161, 230, and 500 kV transmission network to serve a load that roughly mimics the actual population of its geographic footprint. The 2000-bus system has 545 generation units, 1455 load buses, and 3202 branches (grouped into 1927 transmission line circuits and 741 transformer circuits). The data-case provides MVA limits for each transmission line. Those limits were determined based on voltage level and physical characteristics of realistic conductors and line configurations. The base-case has no lines loaded more than 85%.

1. Data generation: The 2000-bus system has a set of 2216 possible branch outage contingencies (at transmission voltage levels). This means a set of 3^{2216} possible outcomes. In addition, as mentioned before, the generation of each data instance (input, output pairs) requires running a full AC contingency analysis. For the large 2000-bus system, it takes around 0.1 seconds to run a power flow and accordingly, around three and a half minutes to run a full AC contingency analysis. Therefore, if one wishes to generate a data-set of 12500 data instances then, it will take (at least) 32 days. Therefore, for the 2000-bus system, we selected a subset of 272 contingencies. All 272 contingencies represent branch outages at the 500kV transmission level. In Figure 4-5 we show the imbalance of the Texas 2000-bus synthetic system dataset. As seen in this figure, the dataset is highly imbalanced. Table 4-2 gives the summary of statistics for the generated dataset.

2. Train/Test split: As seen from Table 4-2, the number of unique label combinations is 11086 which is almost equal to the number of samples of the dataset. Accordingly, we followed the

same dataset splitting approach utilized for the IEEE 118-bus system (i.e., split data with the goal of maintaining an equal distribution of each class for each contingency in each data subset). In Figure 4-6 and Figure 4-7, we show the distribution of each class for each contingency in the training and test datasets. Compared to Figure 4-5, one can see that the training and testing sub-sets have similar distributions to one another and to that of the whole dataset.

4.4 Implementation in Tensorflow

The input to the CNN-based classifier is a highly sparse $N \times N \times 2$ image where N is the number of system buses. Information is carried on a small percentage of the image pixels. Now, due to the bias term in (4), non-useful information is allowed to “propagate” through the network. This means that the CNN must learn to be invariant to a wide range of non-useful patterns which could be challenging and therefore, might lead to sub-optimal performance. In general, there are two approaches for dealing with the sparsity of input feature maps, either: 1) Use the adjacency matrix as an additional input to the standard convolution [79, 80]. The basic implementation of this approach is shown in Figure 4-8. 2) Use a “compressed” representation of input along with a hash table (or a pointer matrix) which defines locations of input elements needed to calculate the element of a given output location [81, 82]. As seen in Figure 4-8, the implementation of the convolution uses “full” input feature maps. This introduces two challenges when implementing CNNs for large power systems. Those are:

1. Memory requirements: The input data set (training, validation, and testing data sets) is a 4D tensor with size $[n, N, N, 2]$, where n is the total number of instances (operating points) contained in the set. For large power systems, the required memory for storing such large data

Table 4-2 Texas 2000-bus systems: statistics of the generated dataset

# of contingencies	# of samples	# of unique label combinations	Samples per label combination			Samples per label's class				
			min	average	max			min	average	max
272	12500	11086	1	44	461	Class	-1	483	9,380	12,493
							0	1	1,674	10,141
							+1	6	1,446	10,747

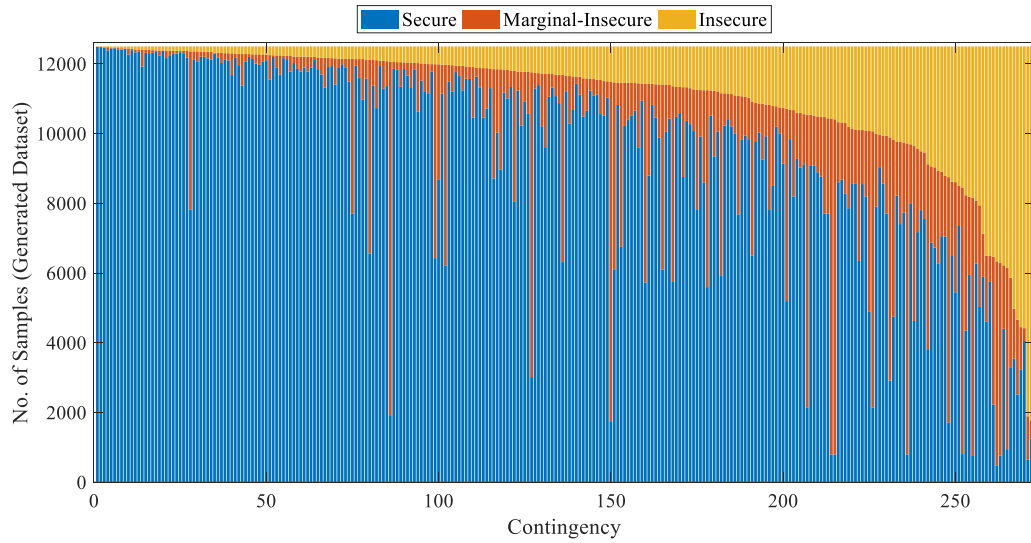


Figure 4-5 Texas 2000-bus synthetic system: imbalance of label dataset. The total number of contingencies is 272.

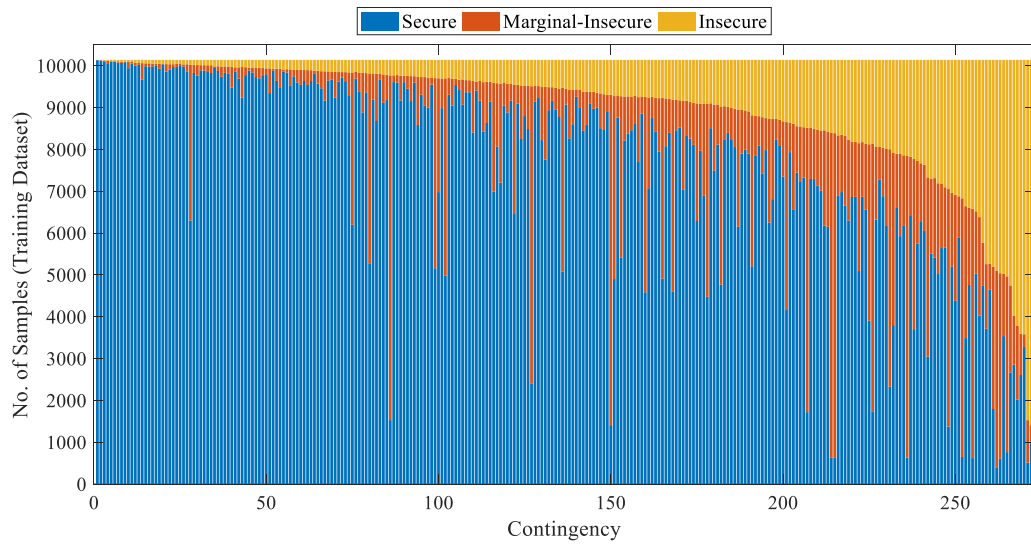


Figure 4-6 Texas 2000-bus synthetic system: imbalance of training dataset. The total number of contingencies is 272.

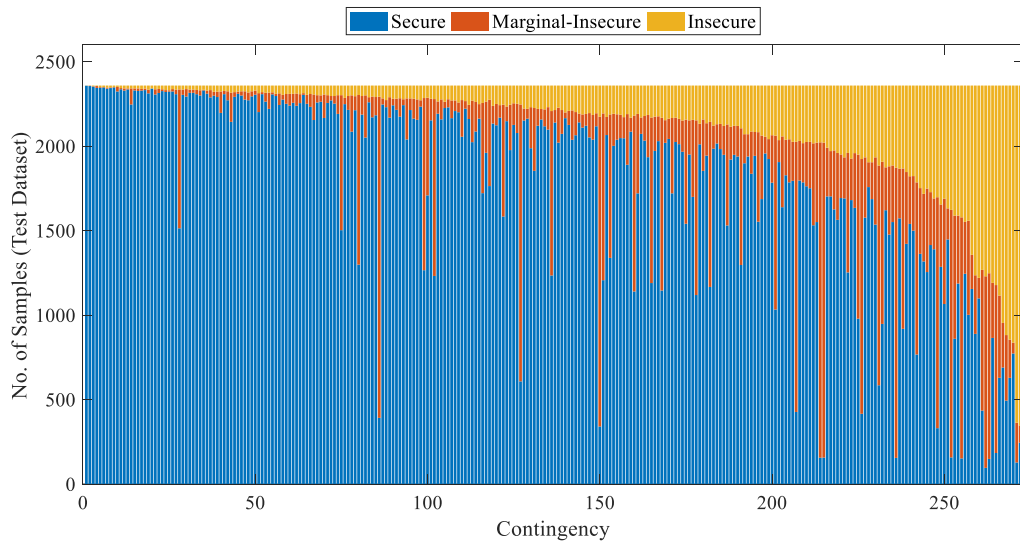


Figure 4-7 Texas 2000-bus synthetic system: imbalance of test dataset. The total number of contingencies is 272.

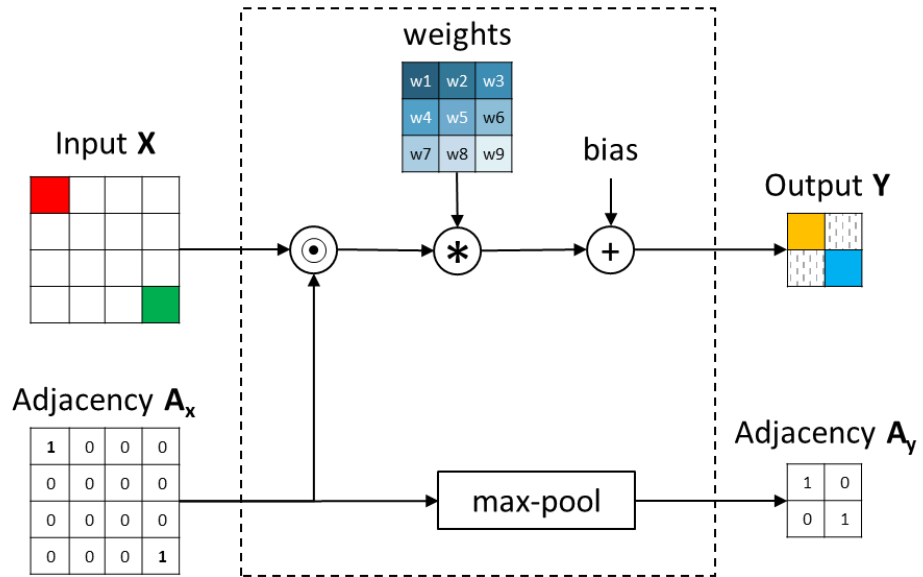


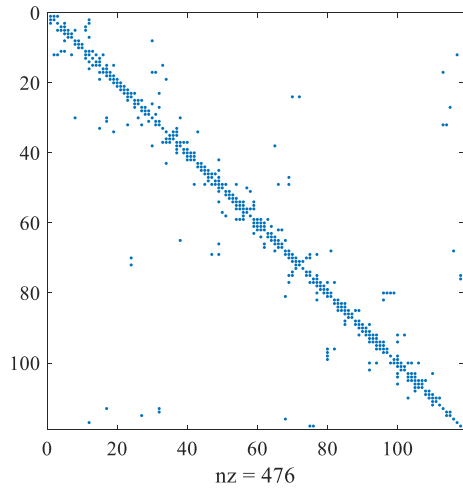
Figure 4-8 Basic implementation of sparse convolution using full arrays.

sets might prevent the application of the proposed method. Now, since training is carried out using smaller batches of the input dataset, one can create and save the training dataset using sparse operations. Now, since training is carried out using smaller batches of the input dataset, one can create and save the training dataset using sparse operations. Then, when training the model, we extract a batch from the sparse input dataset, transform it into a 4D full [batch-size, N, N, 2] tensor and then feed it to the training function [68]. Indeed, this would provide a workaround for saving and loading training datasets of large power systems. However, the next challenge remains.

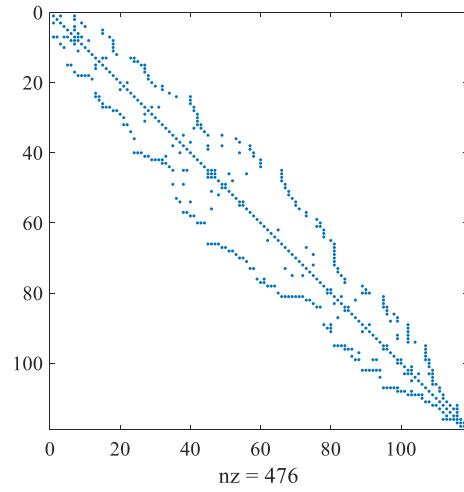
2. Computational burden: Training is carried out using batches of training datasets. Accordingly, convolution is carried out over full 4D arrays of shape [b, Xh, Xw, c]. Accordingly, the computational burden of the training process is increased as the system size is increased (sometimes to the point where performing convolution becomes infeasible). In addition, the time taken to train the network would also increase, and therefore, tuning and the search of optimal parameters becomes expensive.

The methods of approach-2 attempt to address both challenges as they use a “compressed” representation of the input feature map. The method as proposed in [81] cannot be implemented using standard tensor operations. Accordingly, the authors in [82] proposed a modified version of [81] which enables the use of standard tensor operations. Here, we introduce our own modification to [81] which enables the use of standard tensor operations and, when compared to [82], requires less number of operations. In Appendix A, we present our method for implementing sparse convolution in Tensorflow. First, a brief overview is presented on the convolutional function and how it can be implemented using matrix multiplication. Then, a description of the proposed method is presented.

Another factor that contributes to the computational burden of training is the number of trainable parameters. As seen from Figure 4-1, the output of convolutional and max-pooling layers is flattened and used as the input of a fully-connected neural net. Accordingly, the number of trainable parameters (weights and biases) of the fully-connected neural net becomes dependent on the “size” of extracted features map. Now, let M represent the corresponding mask of the extracted feature map (M is the same whether convolution and max-pooling are implemented using “full” or “compressed” representation of feature maps). If “full” representation is used, then, the number of trainable parameters becomes dependent on the size of M . On the other hand, if a “compressed” representation of feature maps is used, then the number of trainable parameters becomes dependent on the number of non-zero elements of M . Therefore, for large power systems, using compressed representation will greatly reduce the number of trainable parameters of the model. Further reduction can be achieved if compressed representation is combined with the re-ordering of buses in the adjacency matrix of the power system. Here, we propose to utilize Reverse Cuthill-McKee (RCM) ordering algorithm [83]. When RCM algorithm is applied to a matrix S , it gives a permutation $\mathbf{r}$ such that $S(\mathbf{r}, \mathbf{r})$ tends to have its nonzero elements closer to the diagonal. Applying the RCM re-ordering algorithm on a power system’s adjacency matrix would result in a less number of non-zero elements in the output image. The original and re-ordered adjacency matrices of the IEEE 118-bus system and Texas 2000-bus system are visualized in Figure 4-9 and Figure 4-10.

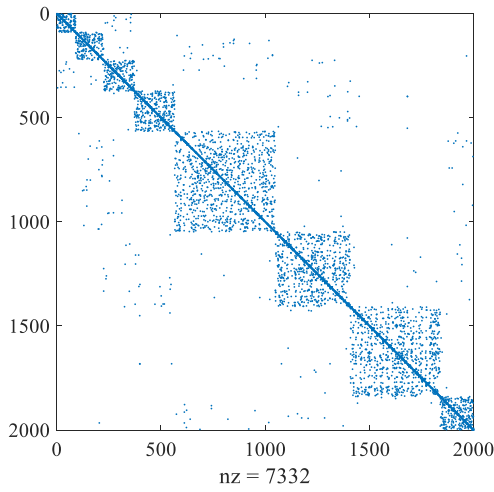


(a)

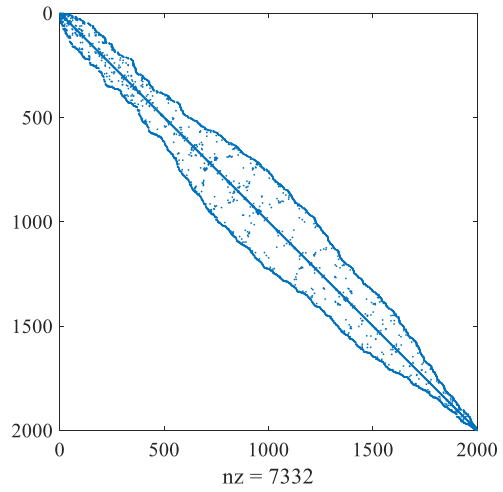


(b)

Figure 4-9 IEEE 118-bus system: Adjacency matrices.(a) the original order of buses, (b) RCM re-ordering of buses.



(a)



(b)

Figure 4-10 Texas 2000-bus synthetic system: Adjacency matrices.(a) the original order of buses, (b) RCM re-ordering of buses.

4.5 Proposed CNN Structure

CNNs have a large design space, with numerous options for their architecture: number of convolutional and max-pooling layers; number, shape, and strides of convolutional filters; shape and strides of max-pooling kernels; the number of hidden layers in the fully connected NN and the number of neurons in each of those hidden layers. With that said, the literature on CNN does offer a number of strategies and common practices for designing a CNN. Here are some which we found useful when deciding on a design for our CNN-based multi-contingency classifier.

1. In general, the larger a CNN (e.g. deeper layers, smaller strides ...etc), the better it is expected to perform. However, going large on the size of a CNN is rendered by:
 - a. The computational burden of the training process (a critical factor especially for large power systems).
 - b. The amount of labeled data available for training. In general, a CNN with a large number of trainable parameters is prone to overfitting, especially if the number of training data points is limited.
2. Typically, the number of convolutional filters is increased as the network goes deeper. This follows the intuition that initial layers hunt for local/low-level features which can be captured with a relatively small number of convolutional filters. On the other hand, deeper layers care for global/high-level (more complex and specialized) features which would require more parameters to learn.

In general, strides $s > 1$ are used within the convolutional layer and/or max-pooling layer to reduce the size of output feature maps, which in turn, reduces the number of trainable parameters. Typically, a stride $s = 2$ is recommended, especially for the lower (initial) layers to minimize the loss of local information.

Based on the above strategies we propose a CNN architecture that has 3 sets of convolutional and max-pooling layers. The size of convolutional filters is 7×7 , 5×5 , and 3×3 . The number of convolutional filters is 16, 32, and 64 respectively. All convolutional filters have 2 strides. All max-pooling layers have a kernel size of 3×3 . For the 118-bus system, max-pooling layers have a stride of 1. On the other hand, for the larger 2000-bus system, max-pooling layers have 2 strides. The fully-connected NN has 2 hidden layers with 1024 and 512 neurons each.

Table 4-3 and Table 4-4 detail the configuration of each layer of the proposed CNN model. It also presents the shape and number of trainable parameters (weights and biases) when the input feature map is structured according to the original ordering of system buses, and when RCM re-ordering is used on system buses. For the 2000-bus system, using compressed representation along with the RCM re-ordering scheme resulted in a significant reduction in the number of trainable parameters (71.2% reduction). This, in turn, will result in: 1) reducing the computation burden, and 2) improving the ratio of trainable parameters to the number of training data points, thus making the network less prone to overfitting. Note that for the IEEE 118-bus system, the shape of the flattening layer of the model that uses compressed image and original bus numbering is 9280 which is larger than that for the model using dense input image 9216. These additional 64 inputs ($9280 - 9216 = 64$) correspond to a “dummy” vector that we utilize in our proposed approach for sparse convolution. See Appendix.

Table 4-3 CNN Structure for IEEE 118-bus system multi-contingency classifier.

layer name	filter size / stride	# of filters	weights shape	bias shape	output shape			# of parameters		
					Dense input image	Compressed image, original bus numbering	Compressed image, RCM re-ordered buses	Dense input image	Compressed image, original bus numbering	Compressed image, RCM re-ordered buses
Input image					[118, 118, 2]	[476, 2]	[476, 2]			
Conv-1	7×7 / 2	16	[7, 7, 2, 16]	[16]	[59, 59, 16]	[920, 16]	[988, 16]	1,568		
Maxpool-1	3×3 / 1				[57, 57, 16]	[1316, 16]	[1188, 16]			
Conv-2	5×5 / 2	32	[5, 5, 16, 32]	[32]	[29, 29, 32]	[530, 32]	[398, 32]	12,800		
Maxpool-2	3×3 / 1				[27, 27, 32]	[586, 32]	[444, 32]			
Conv-3	3×3 / 2	64	[3, 3, 32, 64]	[64]	[14, 14, 64]	[195, 64]	[137, 64]	18,432		
Maxpool-3	3×3 / 1				[12, 12, 64]	[145, 64]	[133, 64]			
Flatten					9216	9280	8512			
Hidden-1			[Flatten, 1024]	[1024]				9,437,184	9,502,720	8,716,288
Hidden-2			[1024, 512]	[512]				524,288		
Total								9,994,272	10,059,808	9,273,376

Table 4-4 CNN Structure for 2000-bus synthetic system multi-contingency classifier.

layer name	filter size / stride	# of filters	weights shape	bias shape	output shape			# of parameters		
					Dense input image	Compressed image, original bus numbering	Compressed image, RCM re-ordered buses	Dense input image	Compressed image, original bus numbering	Compressed image, RCM re-ordered buses
Input image					[2000, 2000, 2]	[7332, 2]	[7332, 2]			
Conv-1	7×7 / 2	16	[7, 7, 2, 16]	[16]	[1000, 1000, 16]	[45553, 16]	[27919, 16]	1,568		
Maxpool-1	3×3 / 2				[499, 499, 16]	[22014, 16]	[11396, 16]			
Conv-2	5×5 / 2	32	[5, 5, 16, 32]	[32]	[250, 250, 32]	[13089, 32]	[6855, 32]	12,800		
Maxpool-2	3×3 / 2				[124, 124, 32]	[4269, 32]	[2411, 32]			
Conv-3	3×3 / 2	64	[3, 3, 32, 64]	[64]	[62, 62, 64]	[1593, 64]	[801, 64]	18,432		
Maxpool-3	3×3 / 2				[30, 30, 64]	[587, 64]	[253, 64]			
Flatten					57600	37568	16192			
Hidden-1			[Flatten, 1024]	[1024]				58,982,400	38,469,632	16,580,608
Hidden-2			[1024, 512]	[512]				524,288		
Total								59,539,488	39,026,720	17,137,696

4.6 Training and Performance Evaluation: Approaches

4.6.1 Training: Loss

As mentioned previously, the proposed CNN-based multi-contingency classifier has three classes, $[+1, 0, -1]$, per contingency. Therefore, a tanh activation function is utilized for the output layer of the fully-connected NN. In this case, the CNN-based classifier can be trained using the mean squared error (MSE) function.

$$Loss_{MSE} = \frac{1}{nm} \sum_{i=1}^n \sum_{j=1}^m (f_j(x_i) - y_{ij})^2 \quad (9)$$

Here, n denotes the number of training samples, m denotes the number of labels (i.e., contingencies). $x_i \in \mathbb{R}^{1 \times d}$ is the instance vector where d denotes the number of input features, $y_i \in \{-1, 0, +1\}^{1 \times m}$ is the label vector corresponding to x_i . Accordingly, $y_{ij} (i \in \{1, \dots, n\}, j \in \{1, \dots, m\})$ represents the class of contingency j for instance x_i . $f_j(x_i)$ denotes the model prediction of y_{ij} (a continuous value in the closed range $[-1, 1]$).

As shown previously in Subsection 4.3.2, the datasets for both IEEE 118-bus and Texas 2000-bus systems are highly imbalanced. Now, $Loss_{MSE}$ does not take the data distribution into consideration. Therefore, the learning algorithm would yield a classifier that simply classifies every contingency as its majority class. This behavior might have been acceptable if the minority class of ALL (or most of) contingencies is “secure”, i.e., it is better to classify a “secure” contingency as “insecure” or “marginal-insecure” rather than the other way around. However, for several contingencies, the minority class is either “insecure”, “marginal-insecure” or both. Typically, when the class of interest is of a minority in the dataset, a cost-sensitive learning approach can be adopted to force the model to care for that class. Here, we utilize the inverse of

the class distributions as misclassification costs, and accordingly, we modify $Loss_{MSE}$ to include these costs as follows:

$$Loss_{WMSE} = \frac{1}{nm} \sum_{i=1}^n \sum_{j=1}^m [w_j^{+1} y_{ij}^2 (1 + y_{ij}) + 2w_j^0 (1 - y_{ij}^2) + w_j^{-1} y_{ij}^2 (1 - y_{ij})] \times (f_j(x_i) - y_{ij})^2 \quad (10)$$

Where w_j^{+1} , w_j^0 , and w_j^{-1} are weights representing the costs of misclassifying “insecure”, “marginally-secure”, and “secure” for the j^{th} contingency respectively. These weights are calculated as follows:

$$w_j^{+1} = \frac{\text{Total number of samples in the training dataset}}{(\text{number of samples where } y_{ij} = +1) \times (\text{number of classes of contingency } j)} \quad (11a)$$

$$w_j^0 = \frac{\text{Total number of samples in the training dataset}}{(\text{number of samples where } y_{ij} = 0) \times (\text{number of classes of contingency } j)} \quad (11b)$$

$$w_j^{-1} = \frac{\text{Total number of samples in the training dataset}}{(\text{number of samples where } y_{ij} = -1) \times (\text{number of classes of contingency } j)} \quad (11c)$$

It should be noted that some contingencies might only have two out of the three classes. In such a case, the weight corresponding to the missing class is set to zero.

Using class weights as defined in (10) might not yield an optimal classification performance. To further explain, we refer readers back to Figure 4-3 and Figure 4-6. As seen in these figures, some contingencies are severely imbalanced with their minority class having a % count less than 0.1%. This would lead to extremely high weights. Such high weight values will tip the scale in the

other wrong direction. Indeed, minorities will be detected however, majorities will have a higher count of misclassifications.

Now, for some contingencies, insecure and/or marginally-insecure classes (i.e., classes of interest) might be of majority in the training dataset while for other contingencies, insecure and/or marginally-insecure are classes of minority. It's interesting to note that $Loss_{MSE}$ yields a model which performs well for majorities. On the other hand, $Loss_{WMSE}$ yields a model which cares more for minorities. Accordingly, we propose to train the CNN-based classifier using both functions within a multi-objective learning framework. In general, one can combine multiple objectives using the linear weighted sum approach. However, this naïve approach requires manual tuning of objectives' weights which could be an expensive process (might take several days for large power systems). Accordingly, we devise a loss function, as detailed below in (12), which enables adaptive learning of objectives' weights. This loss function is a trivial extension to the work proposed in [84]. Originally, the authors of [84] derived this loss function for multi-task learning however, by analogy we can extend its application to the multi-objective learning problem at hand.

$$Loss = e^{-\alpha_1} * Loss_{MSE} + e^{-\alpha_2} * Loss_{WMSE} + \alpha_1 + \alpha_2 \quad (12)$$

Where α_1 and α_2 are added to the set of trainable parameters. As α_1 increases, the weight of $Loss_{MSE}$ decreases. Similarly, as α_2 increases, the weight of $Loss_{WMSE}$ decreases. The term $\alpha_1 + \alpha_2$ in the objective function prevents the optimizer from increasing α_1 and α_2 in an attempt to drive the term $(e^{-\alpha_1} * Loss_{MSE} + e^{-\alpha_2} * Loss_{WMSE})$ to zero. In other words, the term $\alpha_1 + \alpha_2$ prevents the learning algorithm from ignoring the loss functions and converging on a minimum by simply driving the terms $e^{-\alpha_1}$ and $e^{-\alpha_2}$ to zero.

4.6.2 Performance Evaluation

In this work, we utilize two performance measures that are commonly used in the literature of classification, especially multi-label classification problems. Those are recall and precision. In a nutshell, recall measures how much of the relevant data is correctly predicted. On the other hand, precision measures how much of the predicted data is relevant. Typically, the performance of a learning system is evaluated by a single-averaged metric. For multi-label multi-class classifiers, model performance can be averaged on the entire data-set (micro-averaged); or on each label (macro-averaged); or on each instance (instance-averaged). The mathematical formulas for calculating each metric are summarized as follows:

Let $y_{ij}^k \in \{0,1\}$, $h_{ij}^k \in \{0,1\}$ such that:

$$y_{ij}^k = \begin{cases} 0 & \text{if } y_{ij} \neq k \\ 1 & \text{if } y_{ij} = k \end{cases} \quad (13a)$$

$$h_{ij}^k = \begin{cases} 0 & \text{if } h_{ij} \neq k \\ 1 & \text{if } h_{ij} = k \end{cases} \quad (13b)$$

Then,

$$\text{Micro} - \text{Recall} = \frac{1}{ml} \left(\frac{\sum_{i=1}^m \sum_{j=1}^l y_{ij}^k h_{ij}^k}{\sum_{i=1}^m \sum_{j=1}^l y_{ij}^k} \right) \quad (14)$$

$$\text{Micro} - \text{Precision} = \frac{1}{ml} \left(\frac{\sum_{i=1}^m \sum_{j=1}^l y_{ij}^k h_{ij}^k}{\sum_{i=1}^m \sum_{j=1}^l h_{ij}^k} \right) \quad (15)$$

Let R_j^k denotes model recall of class k for contingency. R_j^k is defined as the ratio of the number of data-instances where contingency j is correctly predicted as class k to the number of data-instances where contingency j is actually labeled class k . In mathematical terms:

$$R_j^k = \frac{\sum_{i=1}^m y_{ij}^k h_{ij}^k}{\sum_{i=1}^m y_{ij}^k} \quad (16)$$

Then,

$$\text{Macro} - \text{Recall} = \frac{1}{l} \sum_{j=1}^l R_j^k \quad (17)$$

Similarly, Let P_j^k denotes model precision of class k for contingency. P_j^k is defined as the proportion of data-instances where contingency j is correctly predicted as class k in the set of data-instances where contingency j is predicted as class k . In mathematical terms:

$$P_j^k = \frac{\sum_{i=1}^m y_{ij}^k h_{ij}^k}{\sum_{i=1}^m h_{ij}^k} \quad (18)$$

Then,

$$\text{Macro} - \text{Precision} = \frac{1}{l} \sum_{j=1}^l P_j^k \quad (19)$$

Let R_i^k denotes model recall of class k for a given instance i . R_i^k is the proportion of contingencies correctly predicted as class k in the set of contingencies actually labeled class k . In mathematical terms:

$$R_i^k = \frac{\sum_{j=1}^l y_{ij}^k h_{ij}^k}{\sum_{j=1}^l y_{ij}^k} \quad (20)$$

Then,

$$\text{Instance} - \text{Recall} = \frac{1}{m} \sum_{i=1}^m R_i^k \quad (21)$$

Similarly, let P_i^k denotes model precision of class k for a given instance i . P_i^k is the proportion of contingencies correctly predicted as class k in the set of contingencies predicted as class k . In mathematical terms:

$$P_i^k = \frac{\sum_{j=1}^l y_{ij}^k h_{ij}^k}{\sum_{j=1}^l h_{ij}^k} \quad (22)$$

Then,

$$\text{Instance – Precision} = \frac{1}{m} \sum_{i=1}^m P_i^k \quad (23)$$

In general, for multi-label classification problems, micro-averaged metrics are of less meaning when it comes to evaluating the model performance. This is mainly due to:

Dataset imbalance: when the dataset is imbalanced, then, the classifier simply learns to classify every contingency as its majority class. Now, if the dataset is dominated by a particular class, then, a micro-averaged metric would be high and thus, misleading.

The classifier has multiple outputs: even if the dataset is perfectly balanced, having to classify more than one label makes performance evaluation a complicated task. To further explain, consider our CNN-based multi-contingency classifier and the following two cases: one instance (operating point) with three misclassified contingencies vs. three instances each with one misclassified contingency. Assuming misclassifications of the same type, then, the micro-averaged measure is the same for both cases even though they reflect different performances of the classifier.

Whether macro-averaged or instance-averaged is the better choice for evaluating model performance, depends on the nature of the classification problem. Here, for CNN-based multi-contingency classification, label-wise evaluation is of more practical merit to power system operators. In addition, since contingencies might be of different importance, then, providing operators with performance metrics for each contingency would enable them to make informed decisions when presented with model predictions. Therefore, in addition to macro-averaged metrics, it is advised to investigate the performance of the model by looking at how it performs on each contingency separately.

4.7 Training and Performance Evaluation: Results

This section presents a set of performance evaluation results for the CNN-based classifiers proposed for both the IEEE 118-bus system and the 2000-bus synthesized Texas system. For each test system, we trained three CNN models. The first model is trained using the standard MSE loss given in (9). The second model is trained using the weighted MSE (WMSE) loss given in (10)-(11). The third model is trained using the multi-objective loss function (MOL) given in (12). The training setup and obtained results are presented and discussed next.

4.7.1 Training Setup

The CNN-based classifier models were implemented in Tensorflow. The following settings were used to train all classifiers: a learning rate of 0.001; a batch-size of 32; and a maximum number of 50 training epochs. The training was carried out on a personal desktop (IntelR CoreTM i7 2.5 GHz, 16 GB RAM). The time taken to complete a training cycle of 50 epochs was around 3 and 10 hours for 118- and 2000-bus systems respectively.

4.7.2 Performance Evaluation

As mentioned previously in sub-section (4.5.2), the performance of multi-label classifiers can be evaluated label-wise and/or instance-wise. For a CNN-based multi-contingency classifier, label-wise evaluation is of more practical merit to power system operators. This is because, for a given class, model performance differs greatly from one contingency to another (as shown next). Therefore, providing operators with performance metrics for each contingency would enable them to make informed decisions when presented with model predictions for them. For example, if the model had a 98% precision on class “insecure” for a given contingency, then the operator is highly encouraged to plan for this particular contingency. On the other hand, if the model had a low

precision, say 10%, on class “secure” for a given contingency, then the operator should re-evaluate cases where this contingency is predicted to be “secure” because; it is highly likely that these cases are actually insecure but the model has falsely predicted them to be secure i.e., wolves in sheep’s clothing. Therefore, we evaluate the performance of the trained CNN multi-contingency classifiers by looking at how they perform for each contingency separately. To ease the evaluation process and to enable a meaningful read on the obtained results, we utilize the following graphs:

1. **Recall-Precision graphs:** These are 2-D scatter graphs displaying recall and precision for each contingency. A recall-precision graph has precision on the x-axis, recall on the y-axis. Each contingency is represented on the graph as a circle. The (x, y) coordinates of each circle indicate precision and recall of class k for a given contingency, i.e. (P_j^k, R_j^k) . The size and color of the circle marker are proportional to the % count of class k in the training dataset. It should be noted that some contingencies might have “undefined” recall and/or “undefined” precision. For a given contingency j , recall is undefined when class k has “zero” counts in the test dataset. On the other hand, precision is undefined when the model makes “no” predictions of contingency j as a class k . Contingencies with either undefined recall and/or undefined precision are NOT displayed on the recall-precision graph.
2. **Model Performance Comparison graphs:** These graphs are used to compare the performance of different models with respect to a base-model. In these graphs, we plot the absolute difference in a performance metric for each contingency between a given model (CNN-WMSE or CNN-MOL) and a base-model (CNN-MSE). A positive difference indicates CNN-WMSE (or CNN-MOL) had a better performance compared to CNN- MSE. The opposite is true for negative differences.

4.7.3 Results and Discussion

a) IEEE 118-bus System

As mentioned above, we trained three CNN multi-contingency classifiers for the IEEE 118-bus system. Each classifier was trained using a different loss function: MSE, WMSE, and MOL. The models' performance is evaluated for each class separately. In Table 4-5, the macro-averaged metrics (calculated for each class) are presented. In Figure 4-11 to Figure 4-13, Figure 4-16 to Figure 4-18, Figure 4-21 to Figure 4-23 the performance of CNN-MSE, CNN-WMSE, and CNN-MOL models is visualized using recall-precision graphs (Figure 4-11 to Figure 4-13 for class “insecure”, Figure 4-16 to Figure 4-18 for class “marginal-insecure”, and Figure 4-21 to Figure 4-23 for class “secure”). Finally, in Figure 4-14, Figure 4-15, Figure 4-19, Figure 4-20, Figure 4-24, Figure 4-25, the performance of CNN-MSE, CNN-WMSE, and CNN-MOL models are compared against one another in terms of recall and precision. In what follows, we use class “insecure” to present our observations and analysis of models' performance. Similar observations can be made for the other two classes.

In general, it is preferred if a contingency lies in the upper right corner of the recall-precision graph i.e., high recall and high precision. For the CNN-MSE model, the recall-precision graph for class “insecure” (Figure 4-11) shows that this model performs well for contingencies where class “insecure” had a higher % count in the training dataset (larger reddish circles located at the top half of the graph). The opposite is true for contingencies which rarely result in an insecure operation of the system (smaller dark bluish circles located at the bottom half of the graph). This is because, for a given contingency, the MSE loss does not take its class distribution into account and therefore, the learning algorithm is guided in favor of the contingency's majority class.

Table 4-5 IEEE 118-bus system: macro-recall and macro-precision

	Macro-Recall			Macro-Precision		
	Insecure	Marginal-Insecure	Secure	Insecure	Marginal-Insecure	Secure
CNN-MSE	38.5%	56.6%	94.9%	61.4%	10.2%	98.1%
CNN-WMSE	82.0%	76.7%	81.7%	45.6%	7.0%	98.1%
CNN-MOL	65.4%	74.3%	92.5%	57.8%	8.7%	98.0%

Table 4-6 IEEE 118-bus system: # of contingencies with undefined recall and/or undefined precision.

	Insecure			Marginal-Insecure			Secure		
	# of contingencies with:			# of contingencies with:			# of contingencies with:		
	$R_j^{+1} = \text{NaN}$ $P_j^{+1} = 0$	$R_j^{+1} = 0$ $P_j^{+1} = \text{NaN}$	$R_j^{+1} = \text{NaN}$ $P_j^{+1} = \text{NaN}$	$R_j^{+1} = \text{NaN}$ $P_j^{+1} = 0$	$R_j^{+1} = 0$ $P_j^{+1} = \text{NaN}$	$R_j^{+1} = \text{NaN}$ $P_j^{+1} = \text{NaN}$	$R_j^{-1} = \text{NaN}$ $P_j^{-1} = 0$	$R_j^{-1} = 0$ $P_j^{-1} = \text{NaN}$	$R_j^{-1} = \text{NaN}$ $P_j^{-1} = \text{NaN}$
CNN-MSE	0	52	7	106	0	37	0	0	1
CNN-WMSE	7	0	0	142	0	1	0	0	1
CNN-MOL	7	0	0	142	0	1	0	0	1

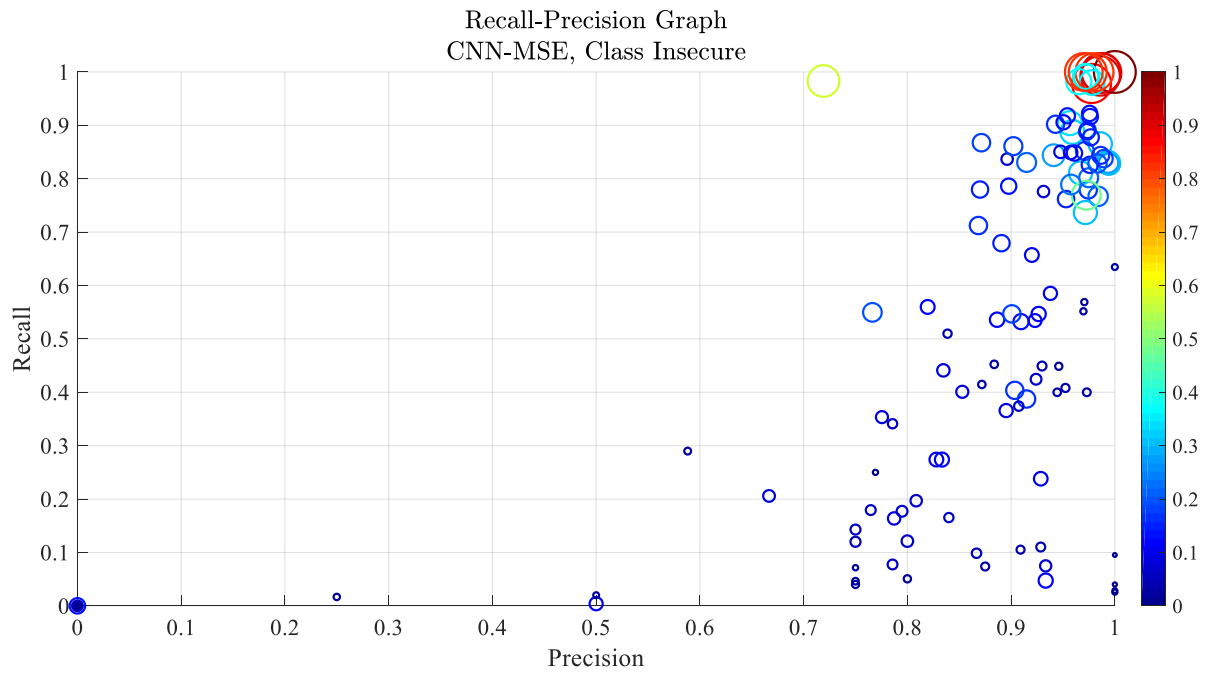


Figure 4-11 IEEE 118-bus system: recall-precision graph for CNN-MSE model, class insecure. For this class, the CNN-MSE model scored a macro-recall and a macro-precision of 38.5% and 61.4% respectively.

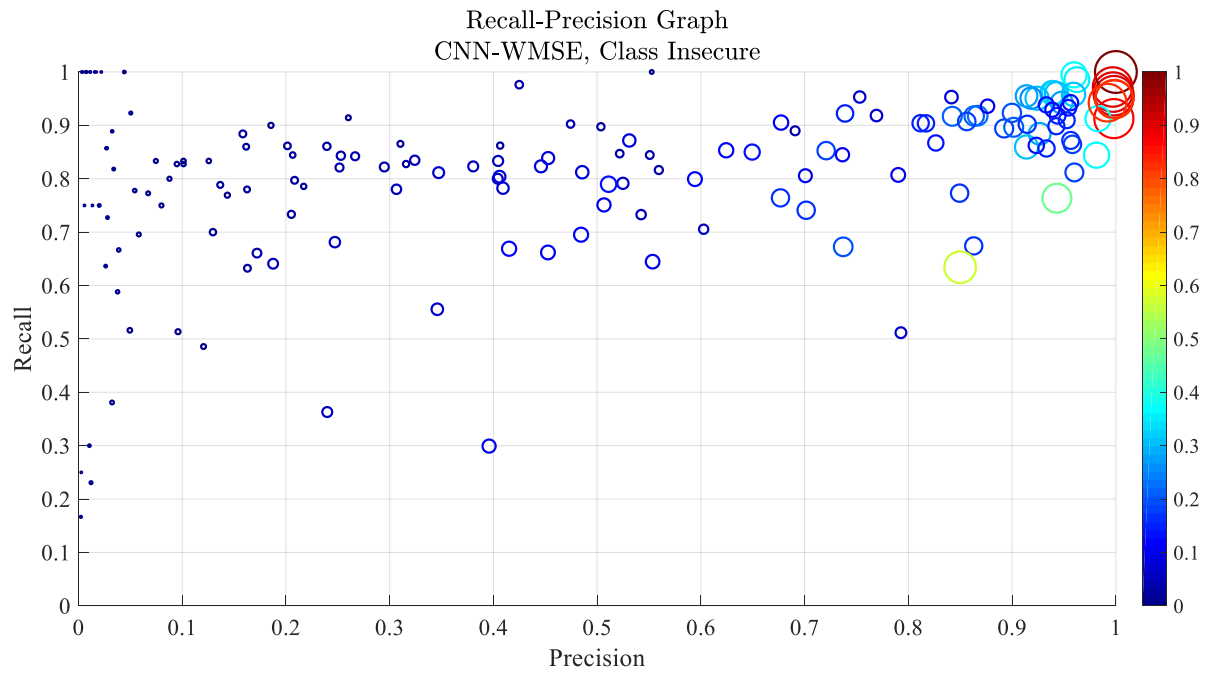


Figure 4-12 IEEE 118-bus system: recall-precision graph for CNN-WMSE model, class insecure. For this class, the CNN-WMSE model scored a macro-recall and a macro-precision of 82.0% and 45.6% respectively.

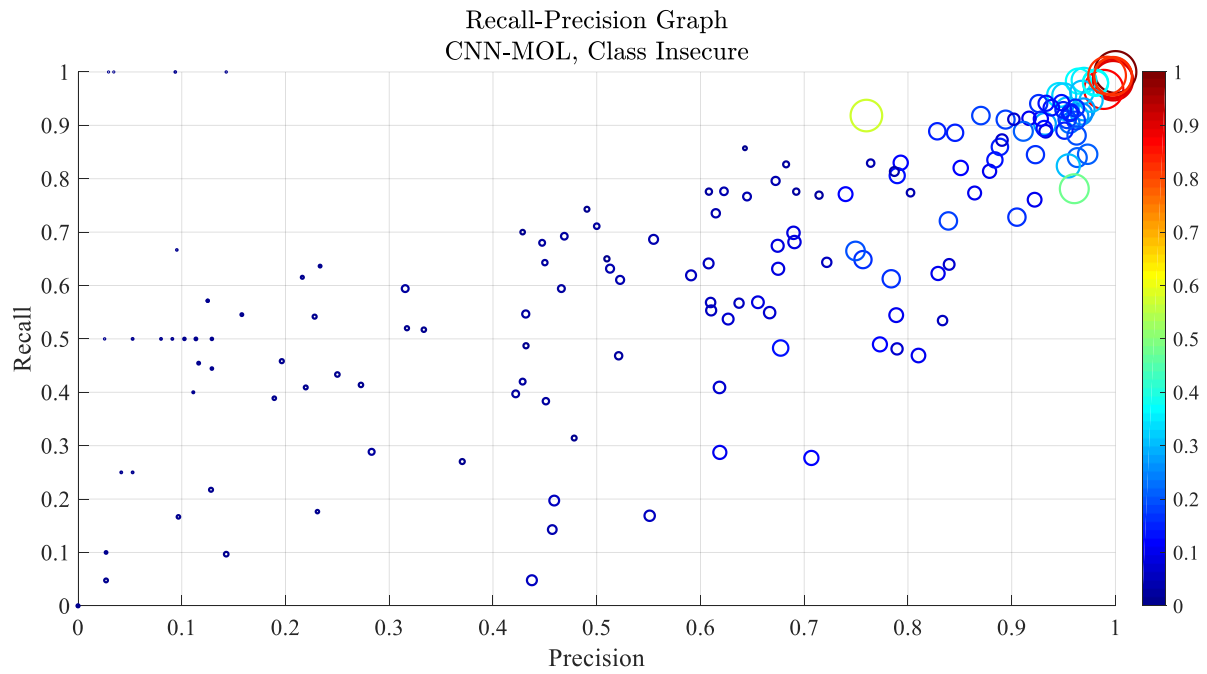


Figure 4-13 IEEE 118-bus system: recall-precision graph for CNN-MOL model, class insecure. For this class, the CNN-MOL model scored a macro-recall and a macro-precision of 65.4% and 57.8% respectively.

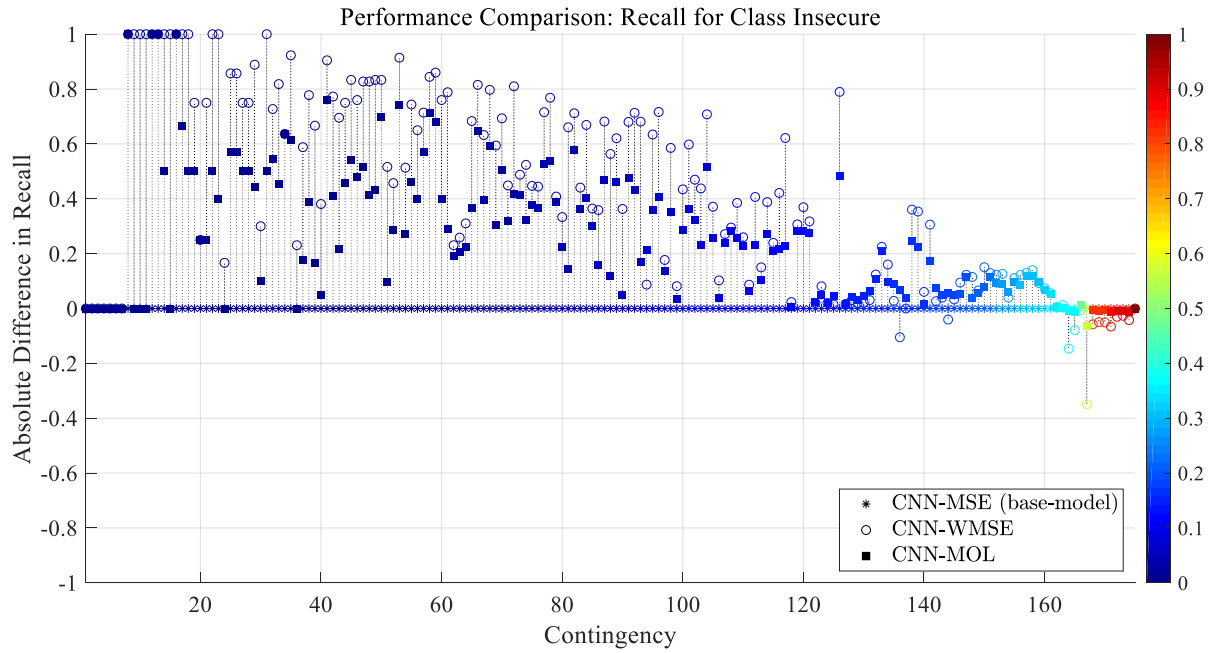


Figure 4-14 IEEE 118-bus system: recall performance comparison graph for class insecure. CNN-MSE is chosen as the base-model for comparison.

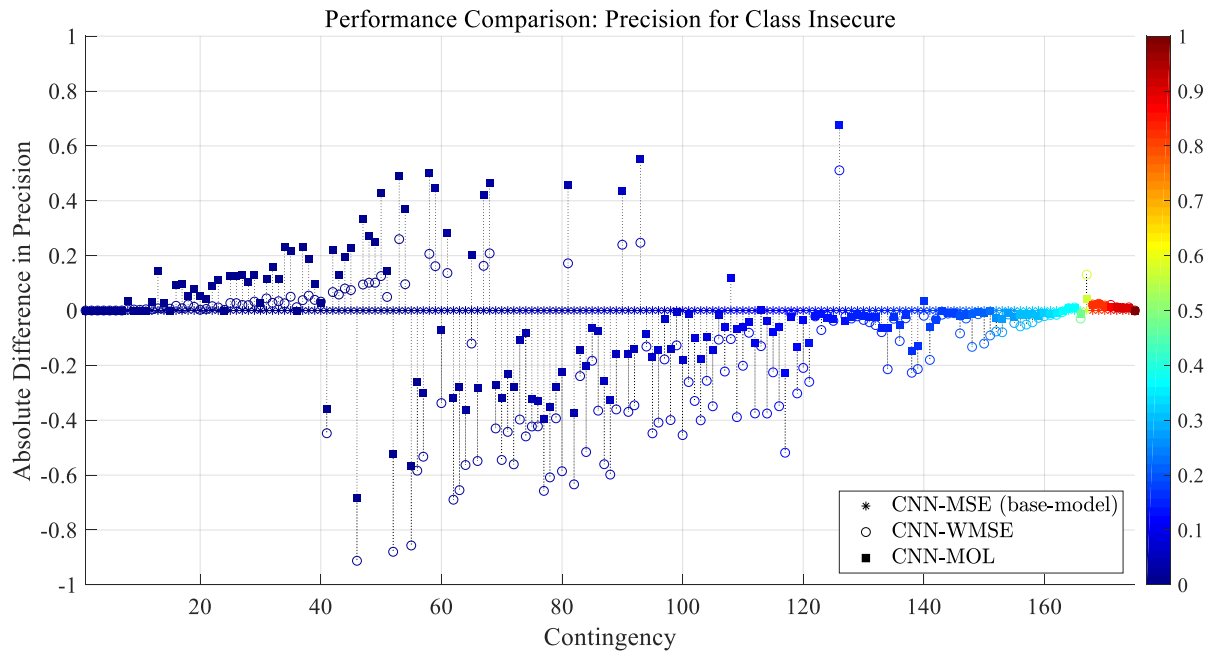


Figure 4-15 IEEE 118-bus system: precision performance comparison graph for class insecure. CNN-MSE is chosen as the base-model for comparison.

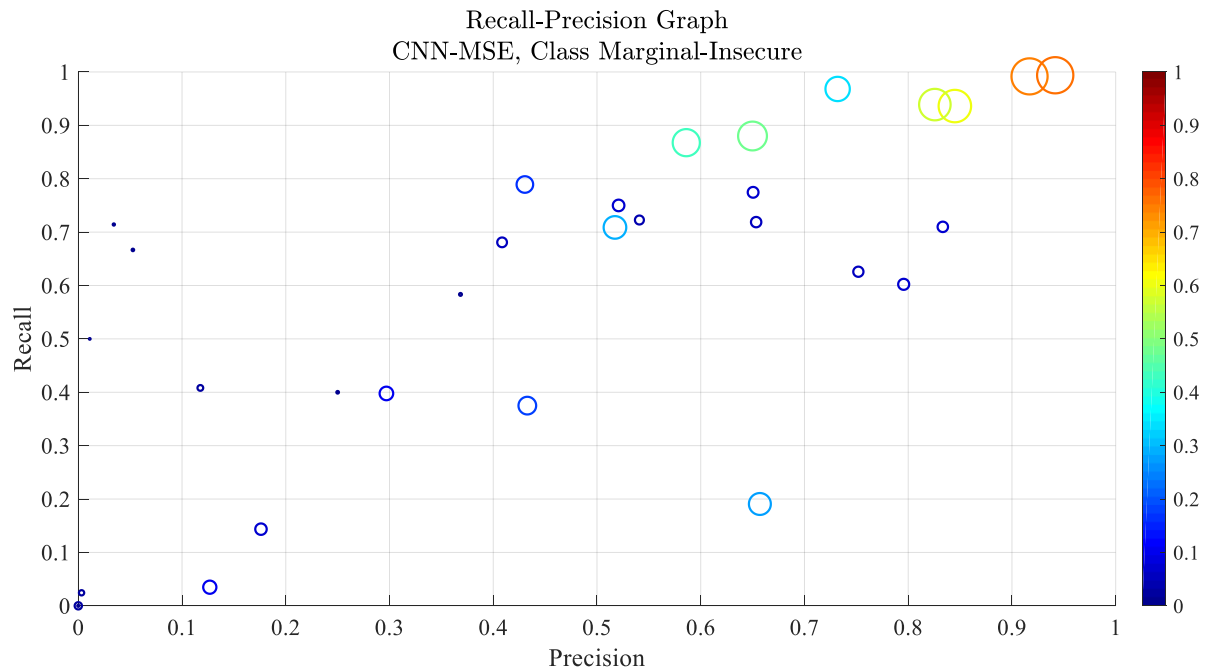


Figure 4-16 IEEE 118-bus system: recall-precision graph for CNN-MSE model, class marginal-insecure. For this class, the CNN-MSE model scored a macro-recall and a macro-precision of 56.6% and 10.2% respectively.

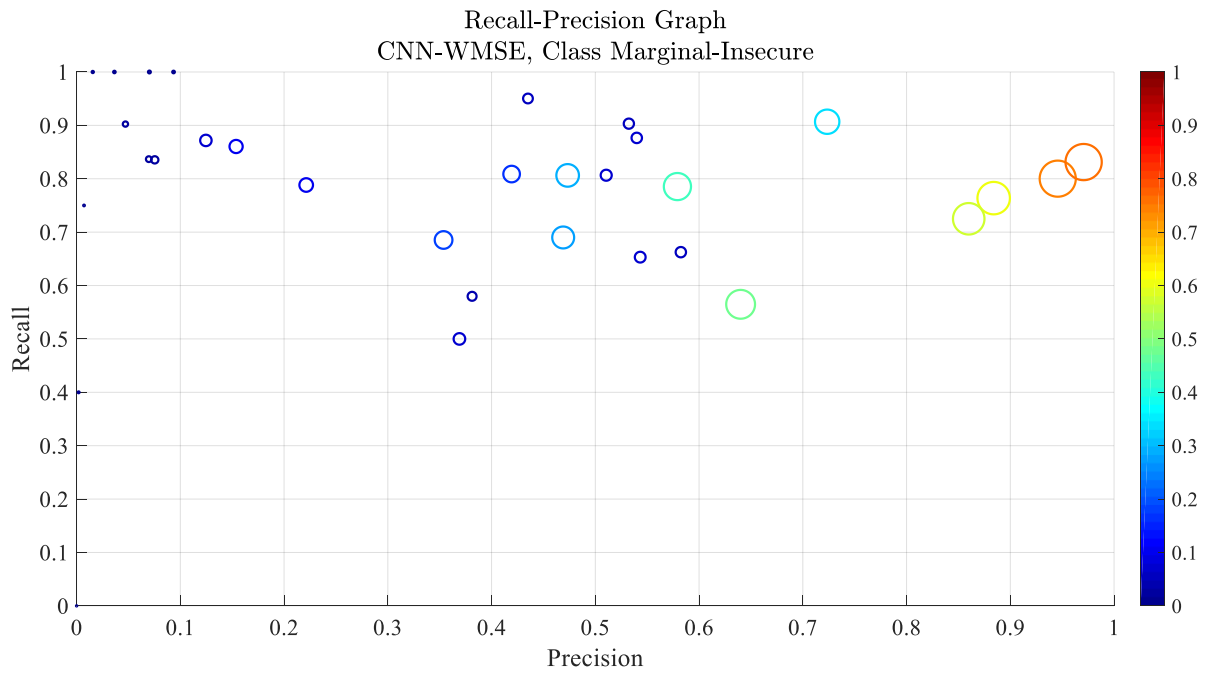


Figure 4-17 IEEE 118-bus system: recall-precision graph for CNN-WMSE model, class marginal-insecure. For this class, the CNN-WMSE model scored a macro-recall and a macro-precision of 76.7% and 7.0% respectively.

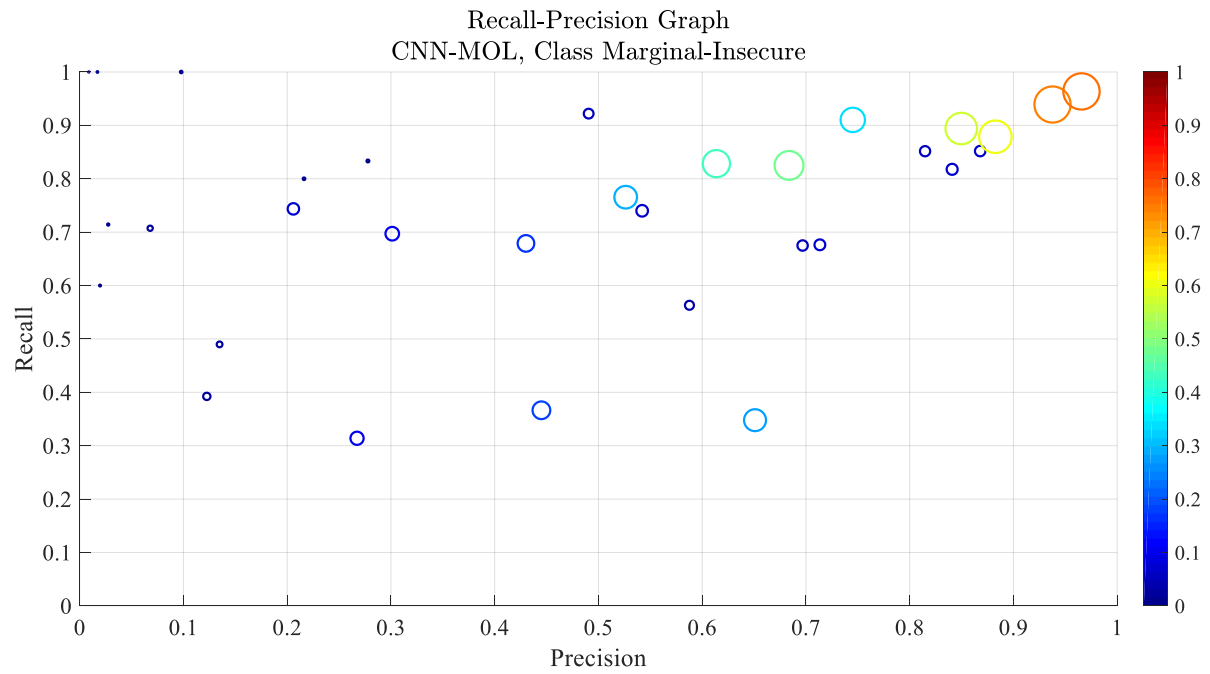


Figure 4-18 IEEE 118-bus system: recall-precision graph for CNN-MOL model, class marginal-insecure. For this class, the CNN-MOL model scored a macro-recall and a macro-precision of 74.3% and 8.7% respectively.

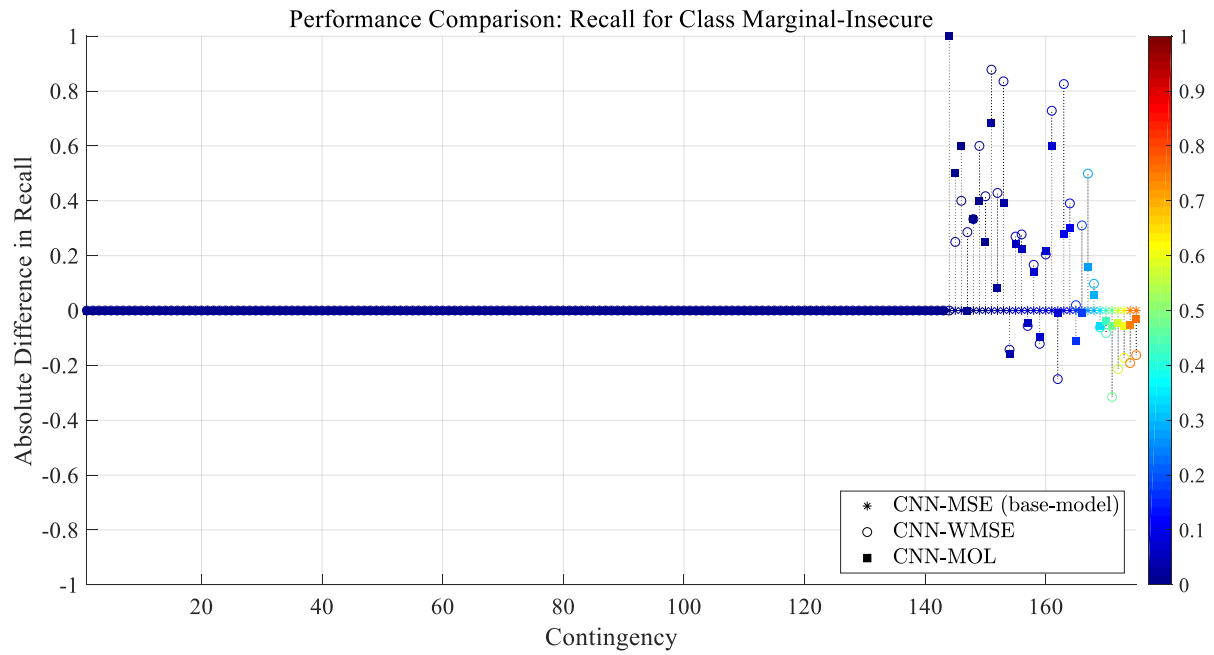


Figure 4-19 IEEE 118-bus system: recall performance comparison graph for class marginal-insecure. CNN-MSE is chosen as the base-model for comparison.

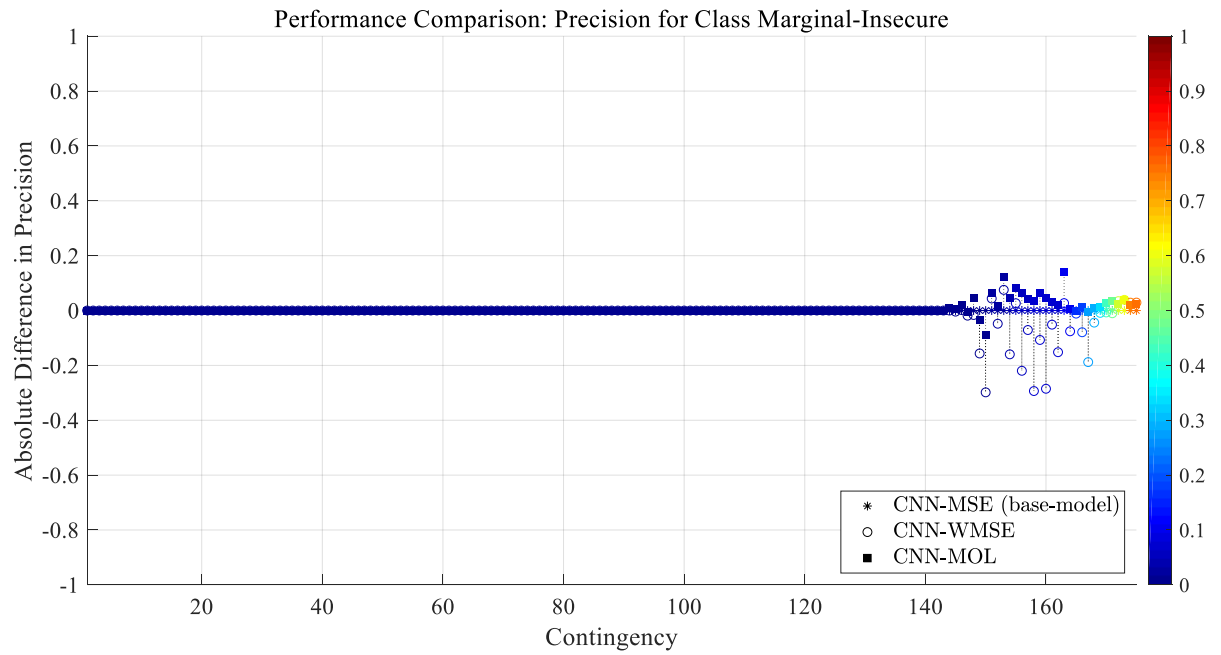


Figure 4-20 IEEE 118-bus system: precision performance comparison graph for class marginal-insecure. CNN-MSE is chosen as the base-model for comparison.

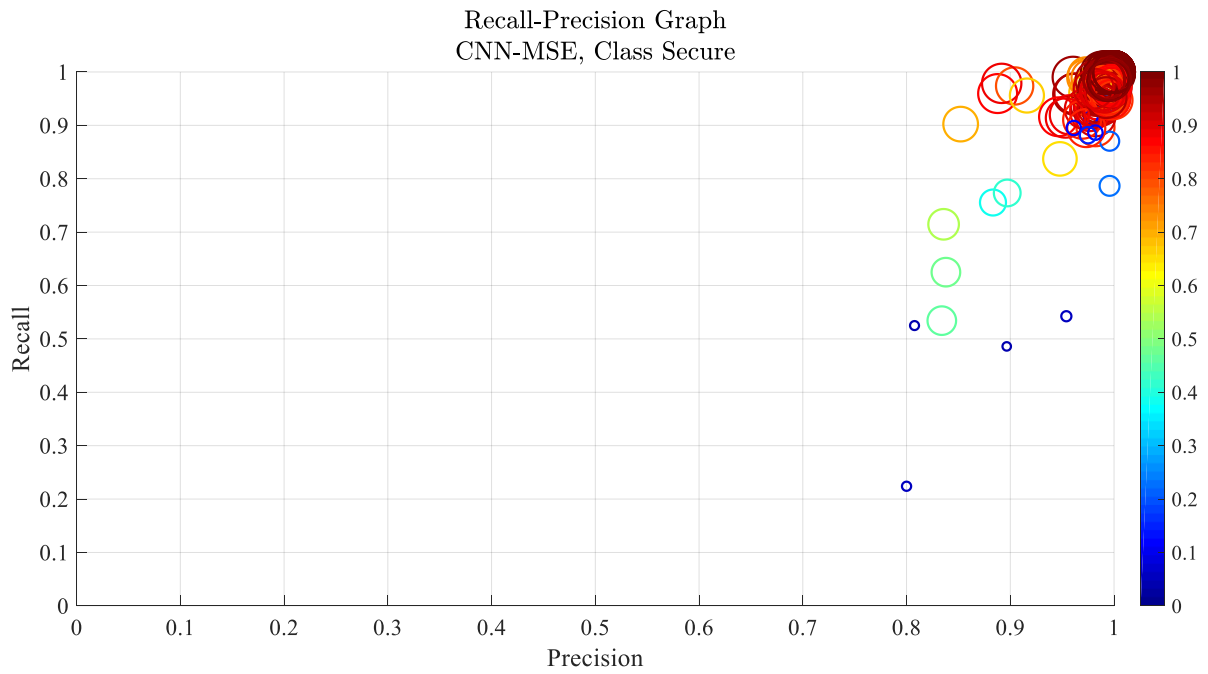


Figure 4-21 IEEE 118-bus system: recall-precision graph for CNN-MSE model, class secure. For this class, the CNN-MSE model scored a macro-recall and a macro-precision of 94.9% and 98.1% respectively.

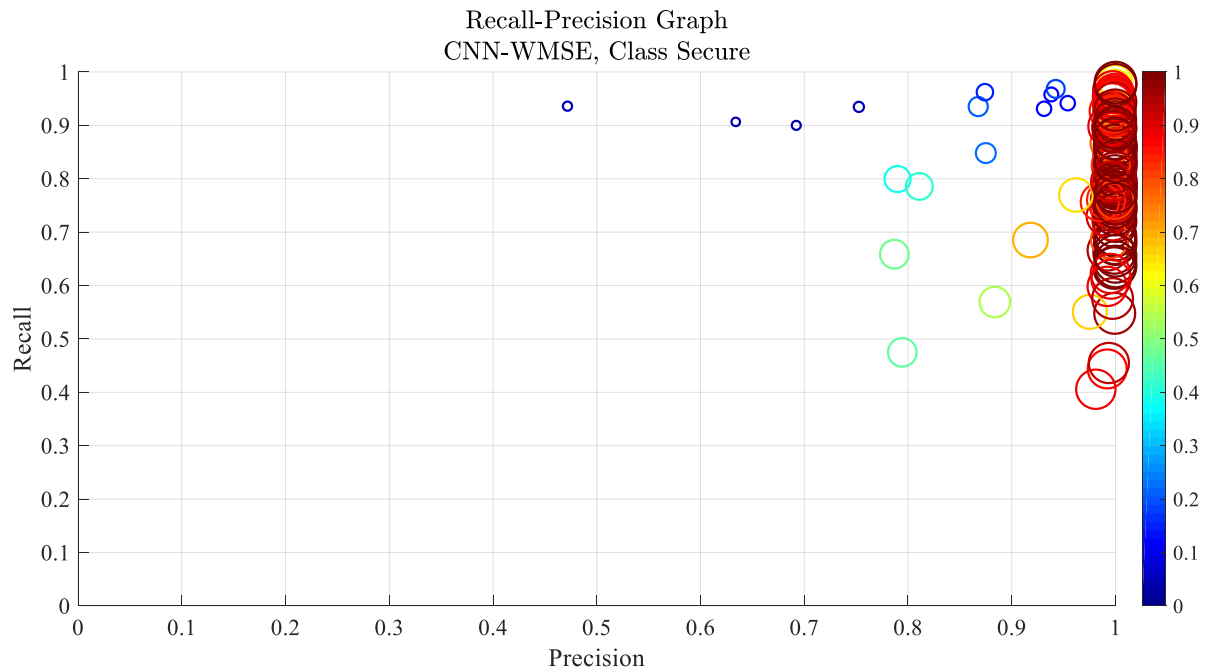


Figure 4-22 IEEE 118-bus system: recall-precision graph for CNN-WMSE model, class secure. For this class, the CNN-WMSE model scored a macro-recall and a macro-precision of 81.7% and 98.1% respectively.

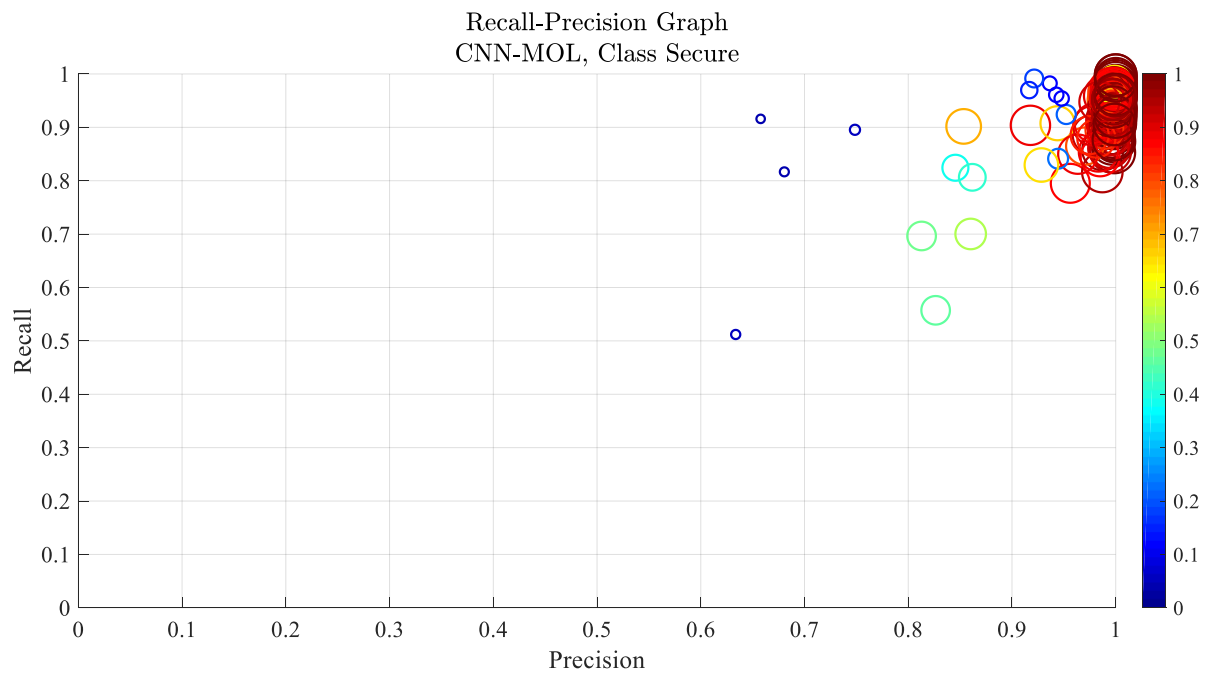


Figure 4-23 IEEE 118-bus system: recall-precision graph for CNN-MOL model, class secure. For this class, the CNN-MOL model scored a macro-recall and a macro-precision of 92.5% and 98.0% respectively.

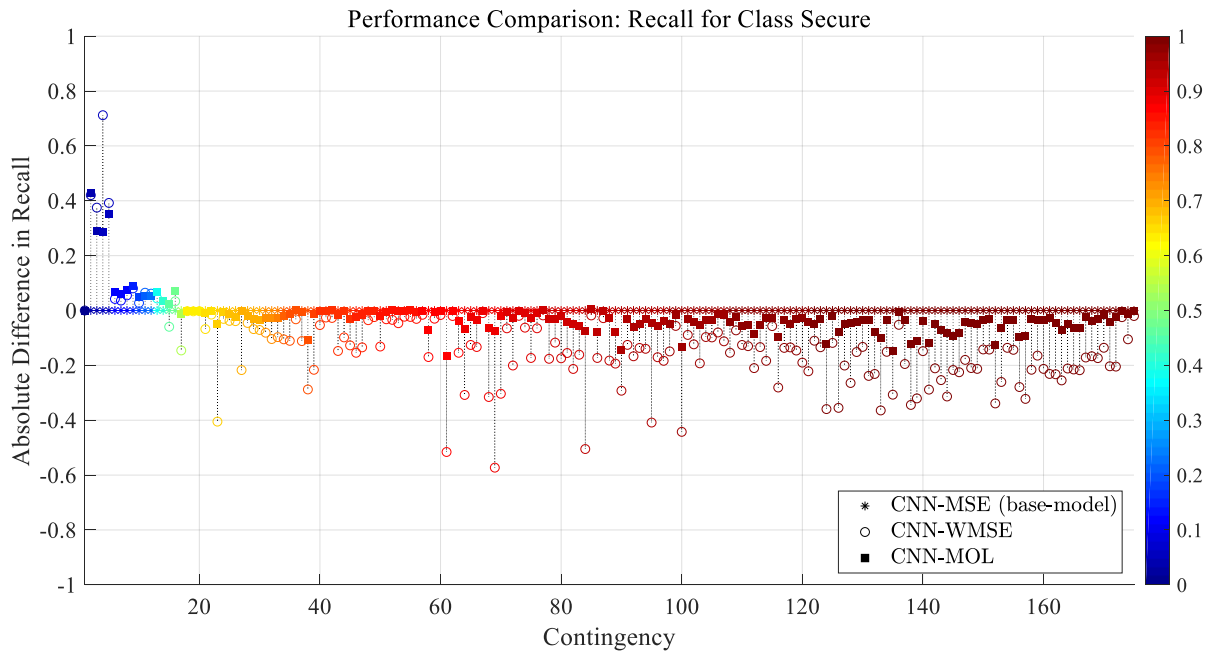


Figure 4-24 IEEE 118-bus system: recall performance comparison graph for class secure. CNN-MSE is chosen as the base-model for comparison.

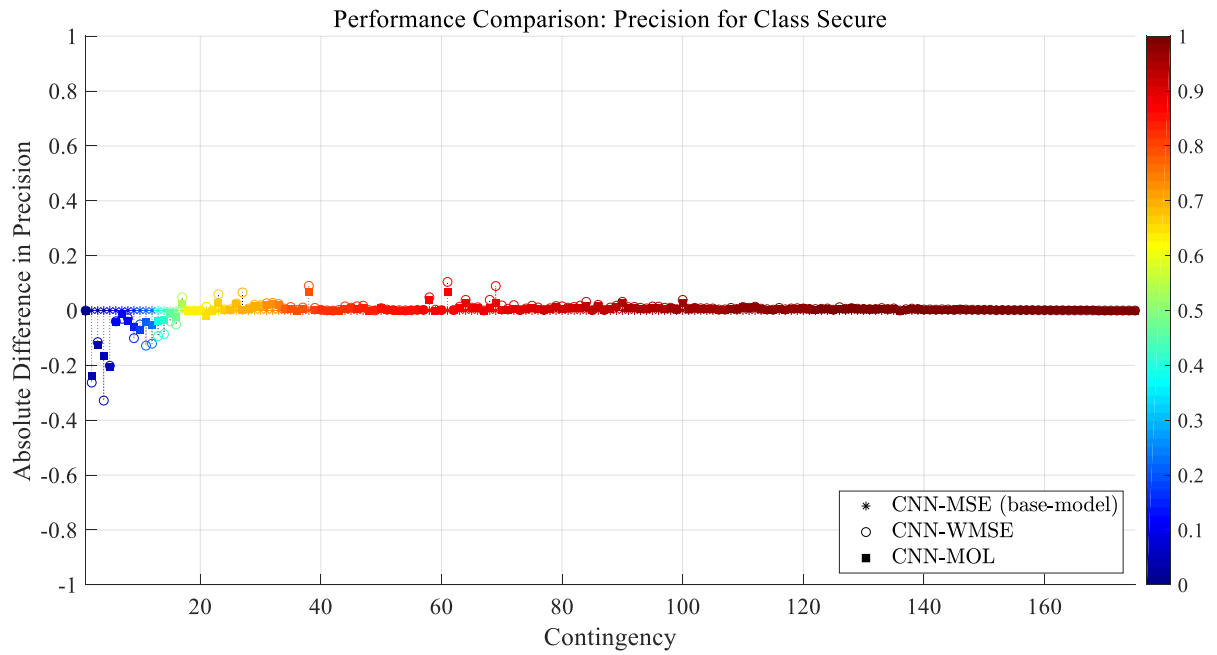


Figure 4-25 IEEE 118-bus system: precision performance comparison graph for class secure. CNN-MSE is chosen as the base-model for comparison.

On the other hand, as seen in Figure 4-12, CNN-WMSE has a better recall performance for minorities (Compared to Figure 4-11, one can see that smaller dark bluish circles have shifted towards the upper half of the graph). This is because WMSE loss penalizes classification errors of minority classes with larger weights which force the learning algorithm to care for them. However, with WMSE, classes of majorities will have a higher count of misclassifications (i.e., a lower recall). This can be seen in Figure 4-12 for contingencies where “insecure” is the class of majorities (Compared to Figure 4-11, one can see that larger reddish circles have shifted towards the lower half of the graph). This tradeoff in model performance is tilted back in favor of contingencies where “insecure” is of majority when MOL is used for training the CNN classifier, see Figure 4-13

In terms of precision, to provide a read on model performance, let’s first break down the definition of precision. Precision is defined as follows:

$$\text{Precision} = \frac{\text{correct predictions of “insecure”}}{\text{total predictions of “insecure”}}$$

where:

total predictions of “insecure”

= correct predictions of “insecure” + false predictions of “insecure”

Now, if “insecure” is a majority, then, the number of correct predictions of “insecure” would be much greater compared to the number of false predictions of “insecure”. Therefore, precision is high (highest for CNN-MSE). This can be seen from Figure 4-11, Figure 4-12, and Figure 4-13 where larger reddish circles are mainly located at the right half of the graph). On the other hand, if “insecure” is a minority, then, the number of correct predictions of “insecure” becomes comparable to the number of false predictions of “insecure”. Now, false predictions of “insecure” are “marginal-insecure” and/or “secure” cases misclassified as “insecure”. Since WMSE and MOL

result in a higher count (compared to MSE) of misclassifications for classes of majority, then, precision tends to be lower with CNN-WMSE and CNN-MOL for contingencies where “insecure” is a class of minority (lowest for CNN-WMSE). This can be seen from Figure 4-12 and Figure 4-13 where smaller dark bluish circles have shifted towards the left half of the graph.

In terms of averaged performance, for class “insecure”, we observe the following: as seen from Table 4-5, the CNN-WMSE scored the highest macro-recall, 68.7% compared to 46.6% and 18.5% for the CNN-MOL and CNN-MSE respectively. This is because, for most contingencies, class “insecure” is a class of minority. Therefore, averaging recall over ALL contingencies has shown CNN-WMSE to be the best model. In terms of precision, CNN-MSE scored the highest macro-precision with 50.8% compared to 43.9% and 35.8% for the CNN-MOL and CNN-WMSE models respectively.

To provide a better comparison of classification performance across the three models, we utilize the performance comparison graphs described above. For class “insecure”, the performance comparison graphs are shown in Figure 4-14 and Figure 4-15. In Figure 4-14, the performance of the three models is compared in terms of recall while in Figure 4-15 the comparison is carried out in terms of precision. As mentioned above, CNN-MSE is taken as the base-model. Accordingly, a positive difference indicates that CNN-WMSE (or CNN-MOL) had a better recall/precision compared to CNN-MSE. The opposite is true for negative differences. From Figure 4-14, one can clearly see that there exists a tradeoff in model performance between contingencies where “insecure” is a minority vs those where “insecure” is of majority. With WMSE loss, this tradeoff in recall performance between minorities and majorities is tilted in favor of contingencies where “insecure” is a minority (because of the extremely large weights used to penalize their classification error). On the other hand, using MOL loss tends to balance this tradeoff.

We also compare CNN-MSE, CNN-WMSE, and CNN-MOL models in terms of precision on class “insecure”. This comparison is presented in Figure 4-15. Again, a positive difference indicates that CNN-WMSE (or CNN-MOL) had a better precision compared to CNN-MSE. Now, if “insecure” is of the dominant majority, then, the number of correct predictions of “insecure” would be much greater than the number of false predictions of “insecure”. Therefore the difference in precision performance between the three models is minimal. This is better seen in Figure 4-25 which compares the precision performance of all models for class “secure” (a class of dominant majority). On the other hand, if “insecure” is a minority, then, the number of correct predictions of “insecure” becomes comparable to the number of false predictions of “insecure”. Now, the number of false predictions of “insecure” depends on the distribution of the other two classes (“marginal-insecure” and “secure”) AND the loss function used (MSE, WMSE, or MOL). Accordingly, the change in precision could be positive or negative, high or low.

The previously stated observations generalize to the other two classes. However, we note an extremely low macro-precision for class “marginal-insecure” for all models (10.2%, 7.0%, and 8.7% for the CNN-MSE, CNN-WMSE, and CNN-MOL models respectively). This can be explained in the light of the results given in Table 4-6. From this table, one can see that a total of 143 contingencies had an undefined recall for class “marginal-insecure” (i.e., in the test dataset, none of the 143 contingencies was ever classified as “marginal-insecure”). With the CNN-MSE model, 37 of these contingencies were never predicted to be marginal-insecure (precision is also undefined) while 106 of these contingencies were sometimes predicted to drive the system into a marginal-insecure state (precision is zero). Accordingly, averaging precision over ALL contingencies, with precision equal to zero for 106 out of the total 175 contingencies, has led to this low macro-precision of 10.2%. The macro-precision was even lower for the other two models,

7.0% and 8.7% for CNN-WMSE and CNN-MOL models respectively as both had 142 contingencies with undefined recall but zero precision.

b) Texas 2000-bus synthetic system

For the 2000-bus Texas system, we also trained three CNN multi-contingency classifiers: CNN-MSE, CNN-WMSE, and CNN-MOL models. In Table 4-7, the macro-averaged metrics (calculated for each class) are presented. In Figure 4-26 to Figure 4-28, Figure 4-31 to Figure 4-33, Figure 4-36 to Figure 4-38, the performance of CNN-MSE, CNN-WMSE, and CNN-MOL models is visualized using recall-precision graphs (Figure 4-26 to Figure 4-28 for class “insecure”, Figure 4-31 to Figure 4-33 for class “marginal-insecure”, and Figure 4-36 to Figure 4-38 for class “secure”). Finally, in Figure 4-29 and Figure 4-30, Figure 4-34 and Figure 4-35, Figure 4-39 and Figure 4-40, the performance of CNN-MSE, CNN-WMSE, and CNN-MOL models are compared against one another in terms of recall and precision. This set of results reveal a similar behavior to the CNN-based classifiers of the IEEE 118-bus system. Next, we provide a summary read on the obtained results.

For a given class k , the CNN-MSE model performs well for contingencies where class k had a higher % count in the training dataset (larger reddish circles located at the top half, see Figure 4-26, Figure 4-31, and Figure 4-36). The opposite is true for contingencies where class k is a minority (smaller dark bluish circles located at the bottom half). Again, this is because, for a given contingency, the MSE loss does not take its class distribution into account and therefore, the learning algorithm is guided in favor of the contingency’s majority class.

Table 4-7 Texas 2000-bus synthetic system: macro-recall and macro-precision

	Macro-Recall			Macro-Precision		
	Insecure	Marginal-Insecure	Secure	Insecure	Marginal-Insecure	Secure
CNN-MSE	18.5%	42.0%	89.0%	50.8%	22.8%	94.3%
CNN-WMSE	68.7%	86.8%	55.9%	35.8%	21.4%	97.6%
CNN-MOL	46.6%	67.5%	79.6%	43.9%	26.0%	96.2%

Table 4-8 Texas 2000-bus synthetic system: # of contingencies with undefined recall and/or undefined precision.

	Insecure			Marginal-Insecure			Secure		
	# of contingencies with:			# of contingencies with:			# of contingencies with:		
	$R_j^{+1} = \text{NaN}$ $P_j^{+1} = 0$	$R_j^{+1} = 0$ $P_j^{+1} = \text{NaN}$	$R_j^{+1} = \text{NaN}$ $P_j^{+1} = \text{NaN}$	$R_j^{+1} = \text{NaN}$ $P_j^{+1} = 0$	$R_j^{+1} = 0$ $P_j^{+1} = \text{NaN}$	$R_j^{+1} = \text{NaN}$ $P_j^{+1} = \text{NaN}$	$R_j^{-1} = \text{NaN}$ $P_j^{-1} = 0$	$R_j^{-1} = 0$ $P_j^{-1} = \text{NaN}$	$R_j^{-1} = \text{NaN}$ $P_j^{-1} = \text{NaN}$
CNN-MSE	0	79	0	0	16	1	0	0	0
CNN-WMSE	0	0	0	1	0	0	0	0	0
CNN-MOL	0	0	0	1	0	0	0	0	0

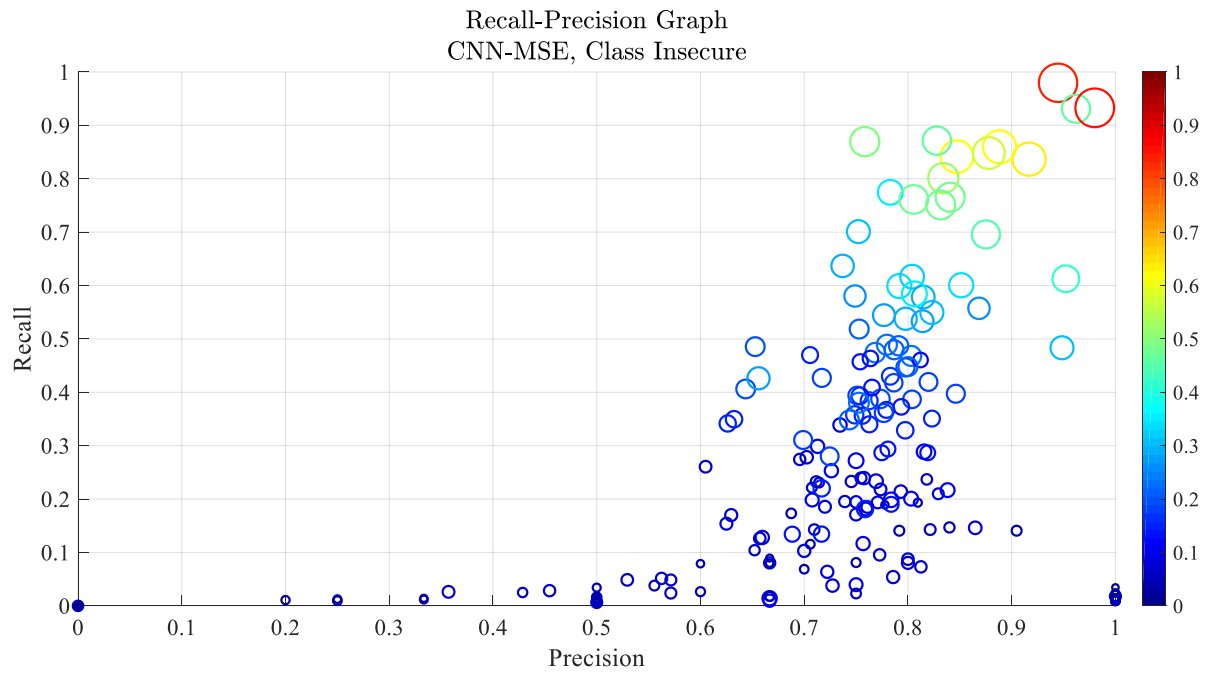


Figure 4-26 Texas 2000-bus synthetic system: recall-precision graph for CNN-MSE model, class insecure. For this class, the CNN-MSE model scored a macro-recall and a macro-precision of 18.5% and 50.8% respectively.

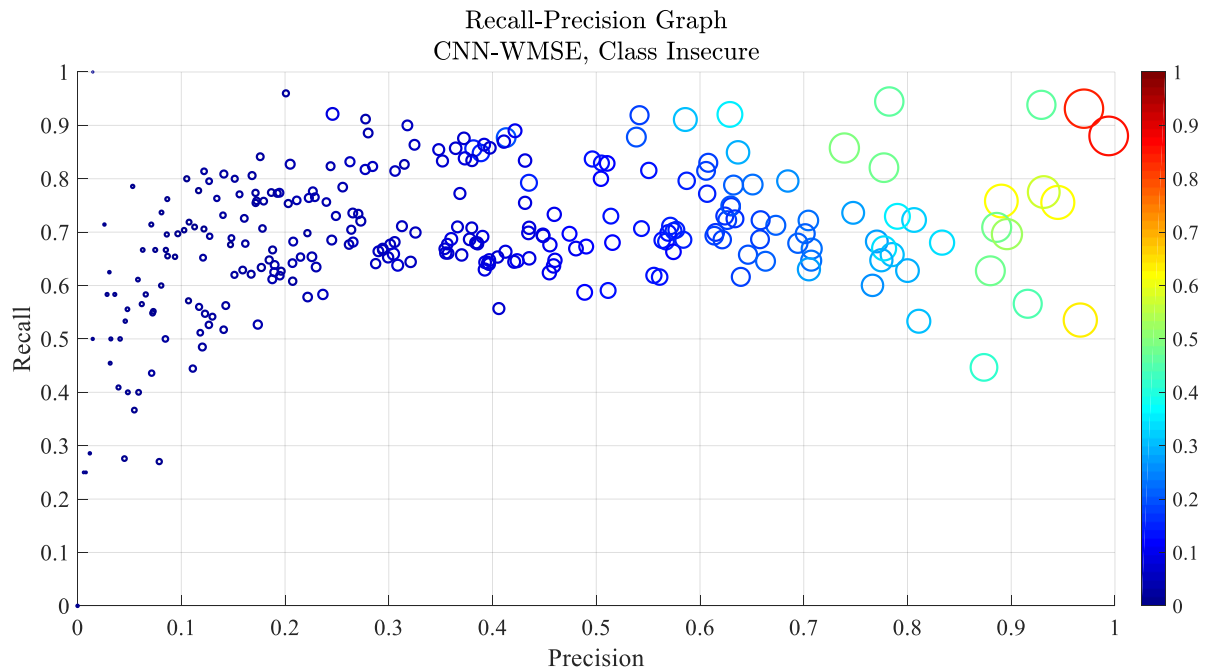


Figure 4-27 Texas 2000-bus synthetic system: recall-precision graph for CNN-WMSE model, class insecure. For this class, the CNN-WMSE model scored a macro-recall and a macro-precision of 68.7% and 35.8% respectively.

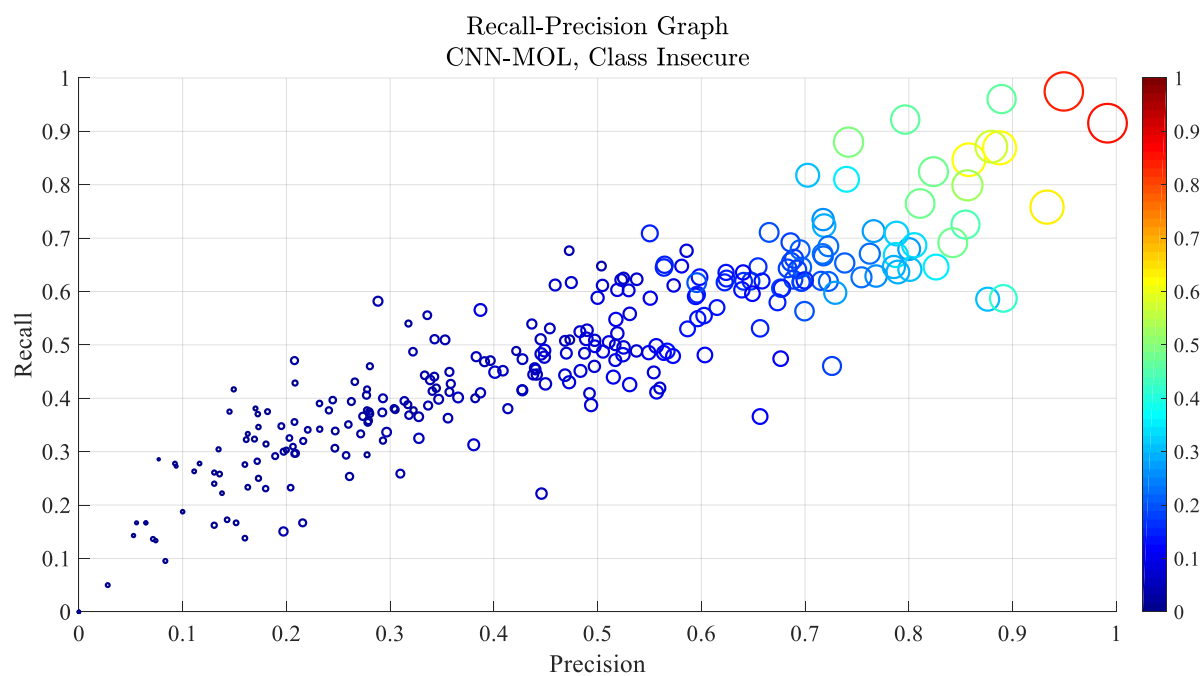


Figure 4-28 Texas 2000-bus synthetic system: recall-precision graph for CNN-MOL model, class insecure. For this class, the CNN-MOL model scored a macro-recall and a macro-precision of 46.6% and 43.9% respectively.

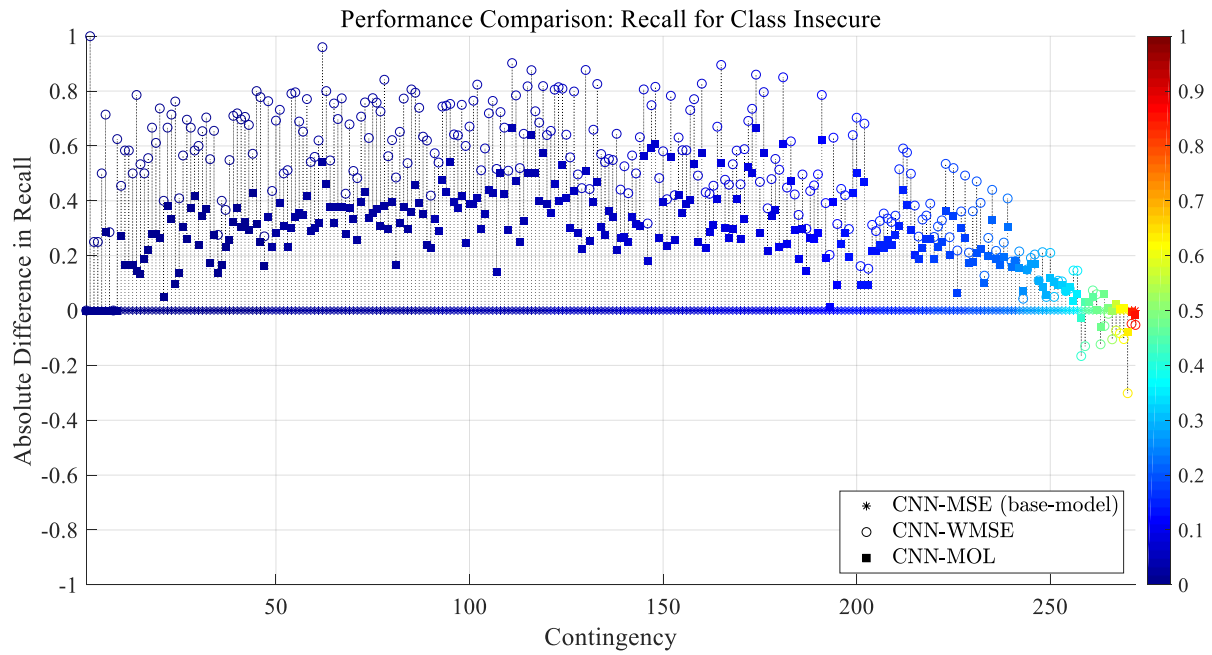


Figure 4-29 Texas 2000-bus synthetic system: recall performance comparison graph for class insecure. CNN-MSE is chosen as the base-model for comparison.

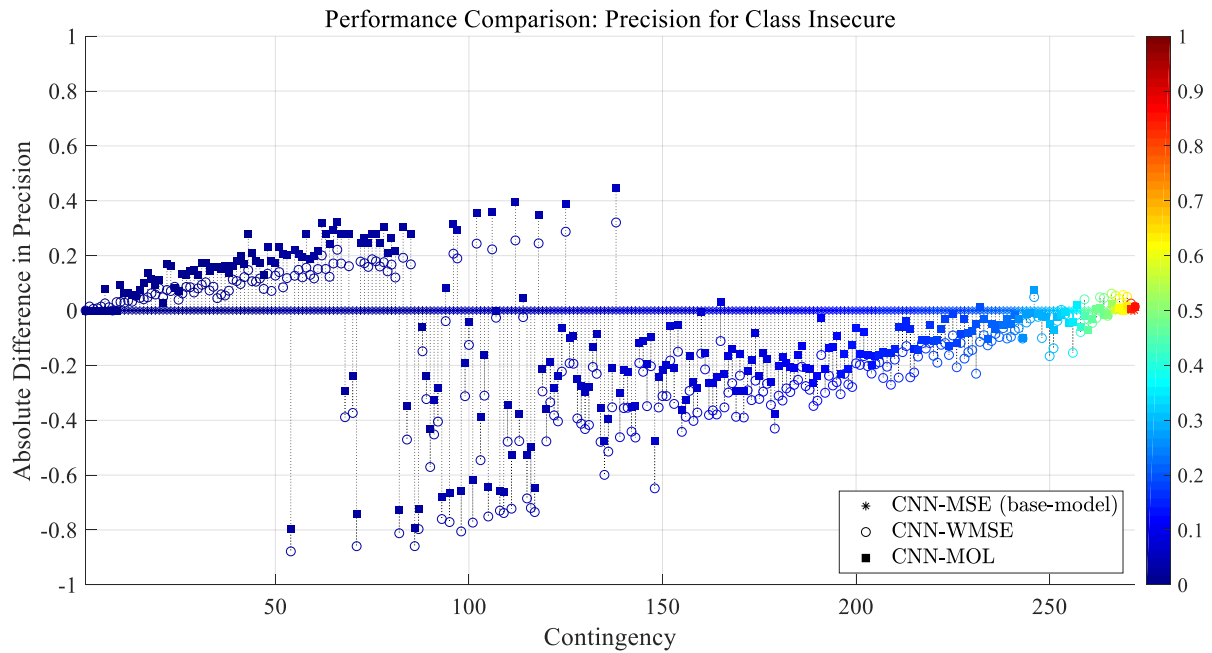


Figure 4-30 Texas 2000-bus synthetic system: precision performance comparison graph for class insecure. CNN-MSE is chosen as the base-model for comparison.

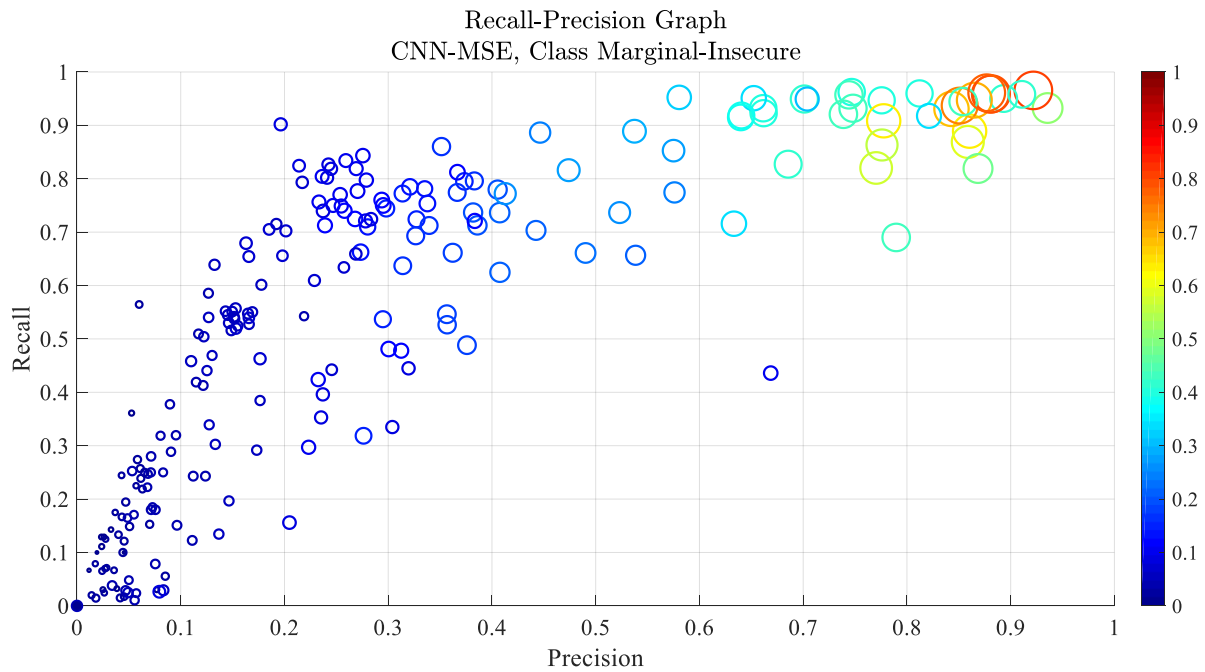


Figure 4-31 Texas 2000-bus synthetic system: recall-precision graph for CNN-MSE model, class marginal-insecure. For this class, the CNN-MSE model scored a macro-recall and a macro-precision of 42.0% and 22.8% respectively.

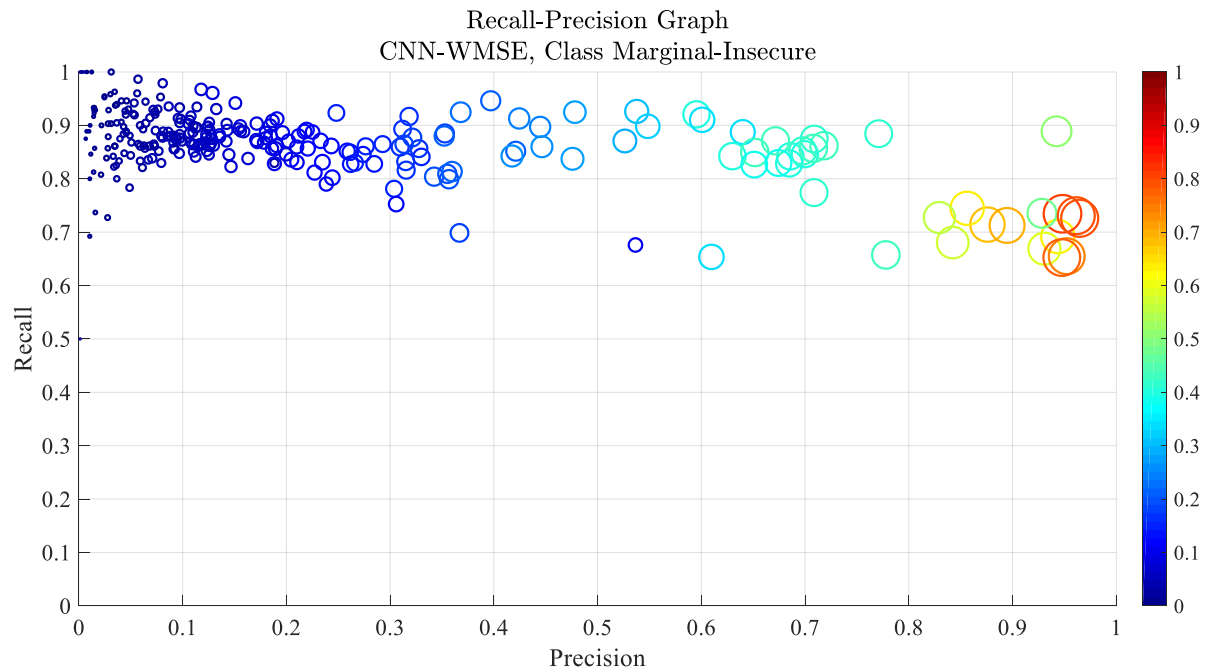


Figure 4-32 Texas 2000-bus synthetic system: recall-precision graph for CNN-WMSE model, class marginal-insecure. For this class, the CNN-WMSE model scored a macro-recall and a macro-precision of 86.8% and 21.4% respectively.

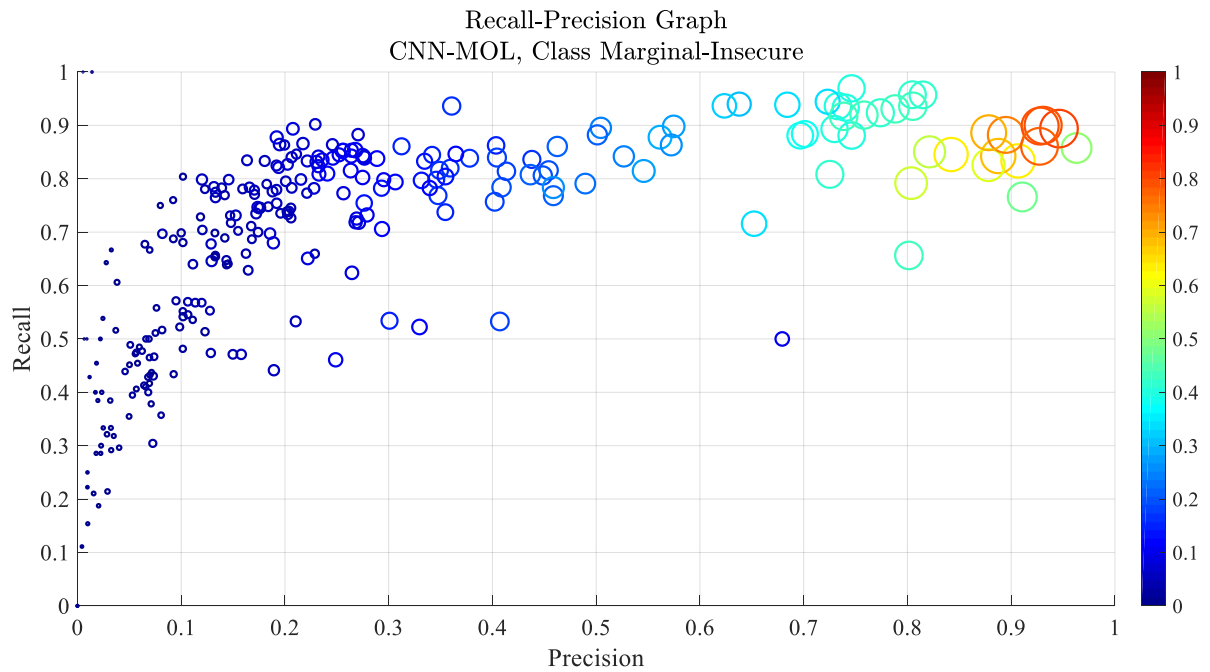


Figure 4-33 Texas 2000-bus synthetic system: recall-precision graph for CNN-MOL model, class marginal-insecure. For this class, the CNN-MOL model scored a macro-recall and a macro-precision of 67.5% and 26.0% respectively.

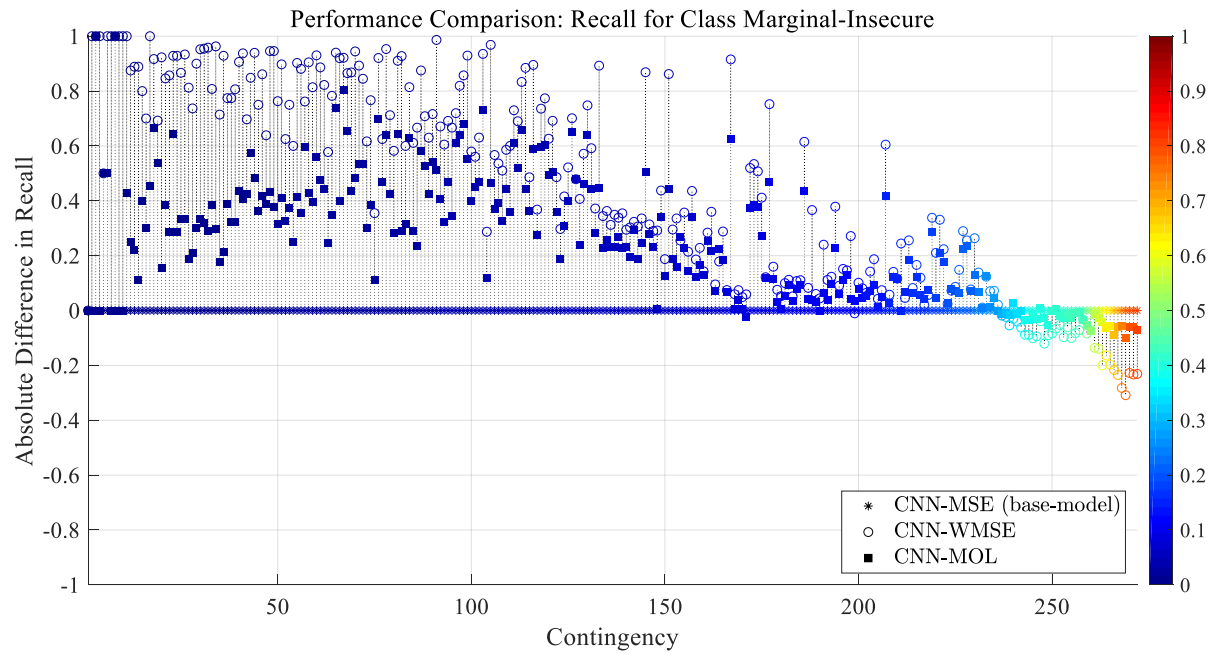


Figure 4-34 Texas 2000-bus synthetic system: recall performance comparison graph for class marginal-insecure. CNN-MSE is chosen as the base-model for comparison.

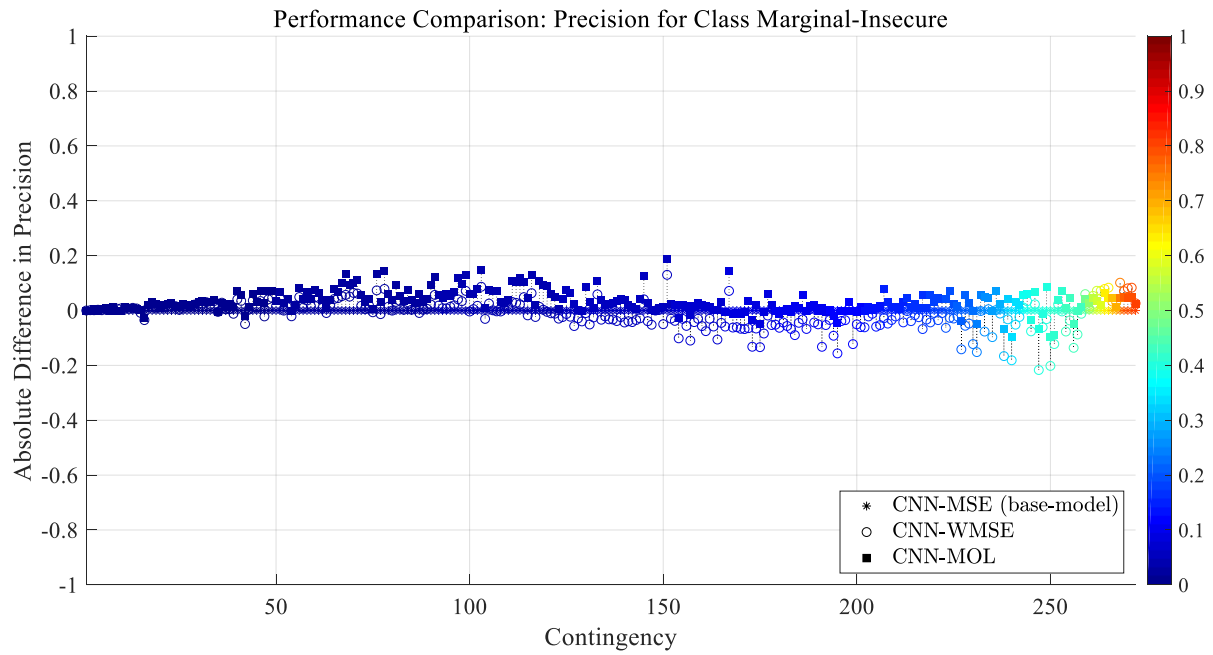


Figure 4-35 Texas 2000-bus synthetic system: precision performance comparison graph for class marginal-insecure. CNN-MSE is chosen as the base-model for comparison.

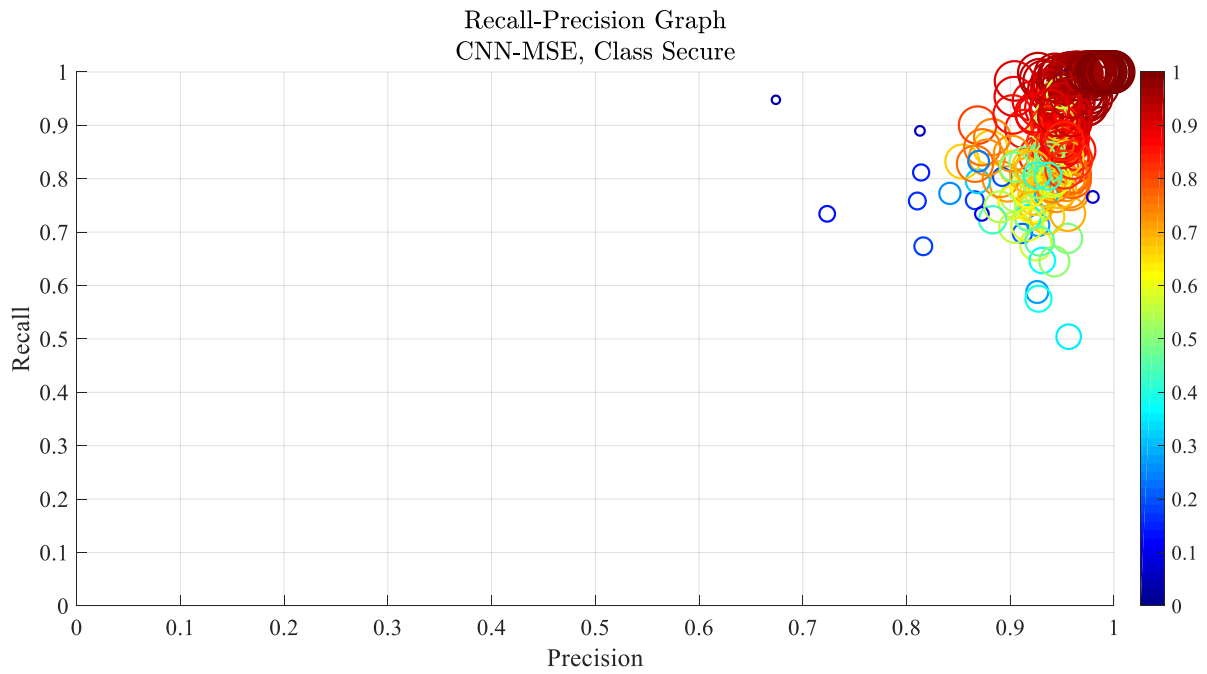


Figure 4-36 Texas 2000-bus synthetic system: recall-precision graph for CNN-MSE model, class secure. For this class, the CNN-MSE model scored a macro-recall and a macro-precision of 89.0% and 94.3% respectively.

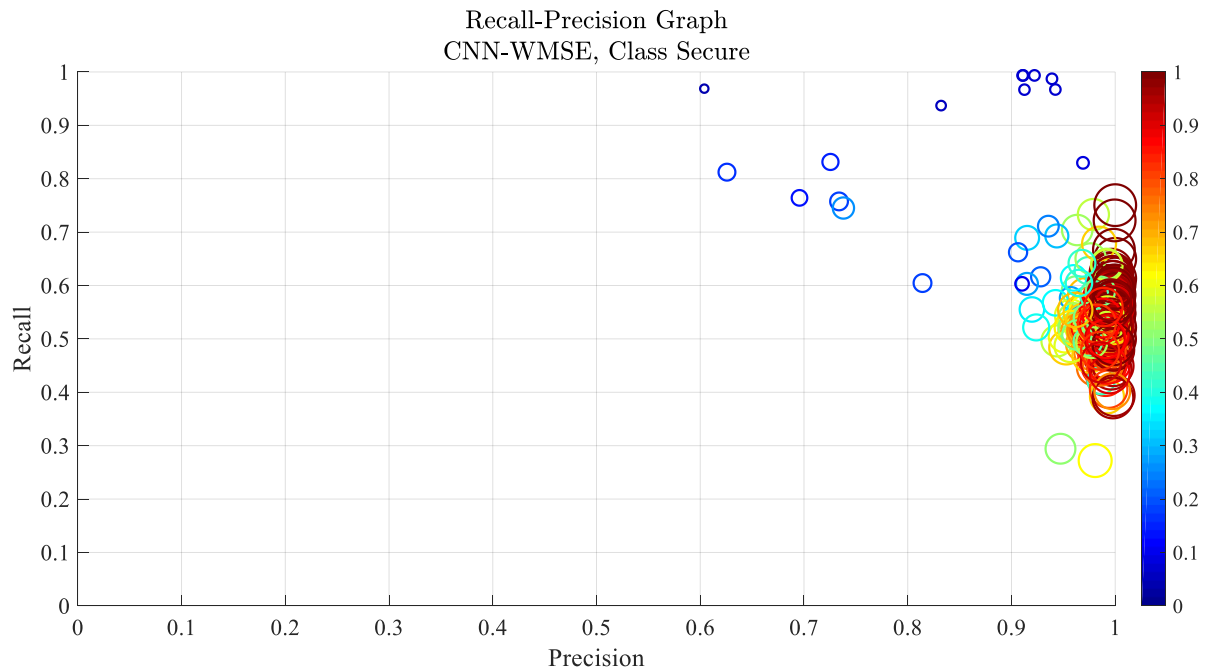


Figure 4-37 Texas 2000-bus synthetic system: recall-precision graph for CNN-WMSE model, class secure. For this class, the CNN-WMSE model scored a macro-recall and a macro-precision of 55.9% and 97.6% respectively.

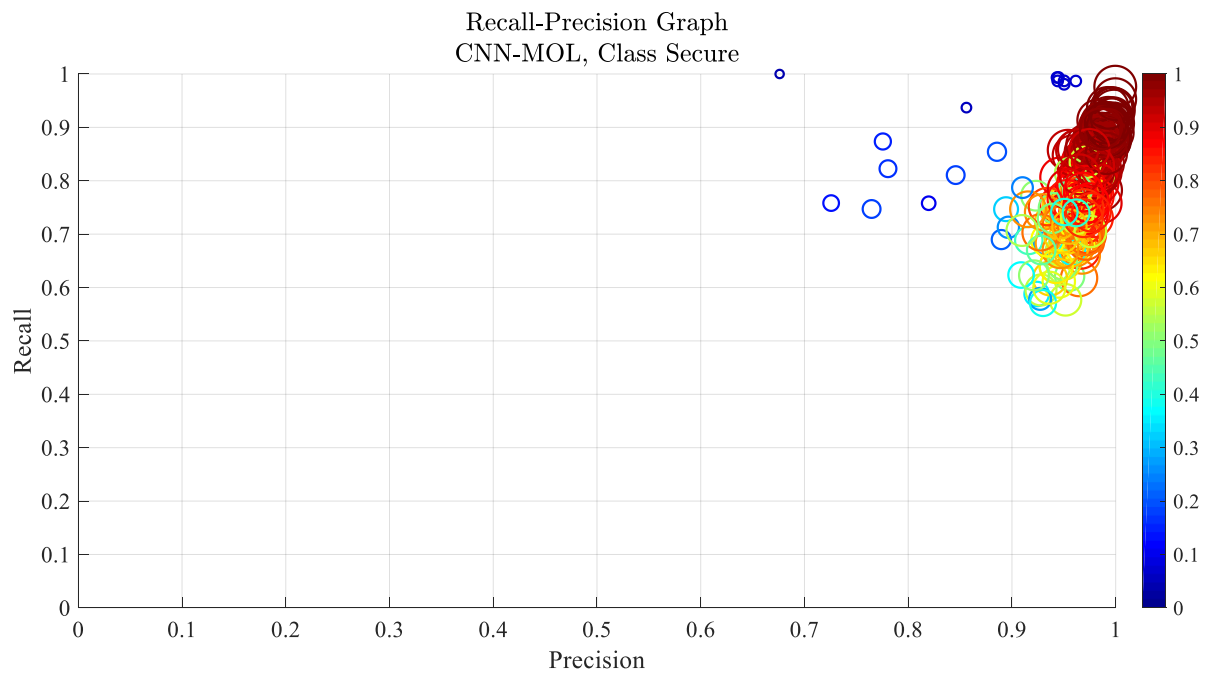


Figure 4-38 Texas 2000-bus synthetic system: recall-precision graph for CNN-MOL model, class secure. For this class, the CNN-MOL model scored a macro-recall and a macro-precision of 79.6% and 96.2% respectively.

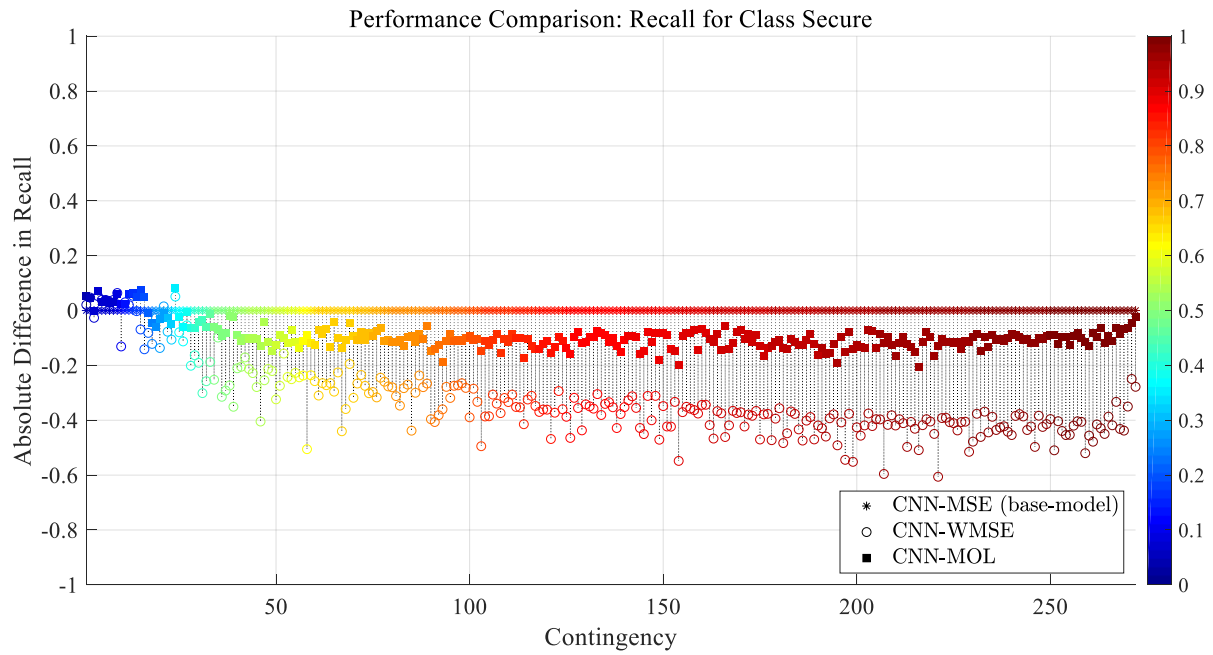


Figure 4-39 Texas 2000-bus synthetic system: recall performance comparison graph for class secure. CNN-MSE is chosen as the base-model for comparison.

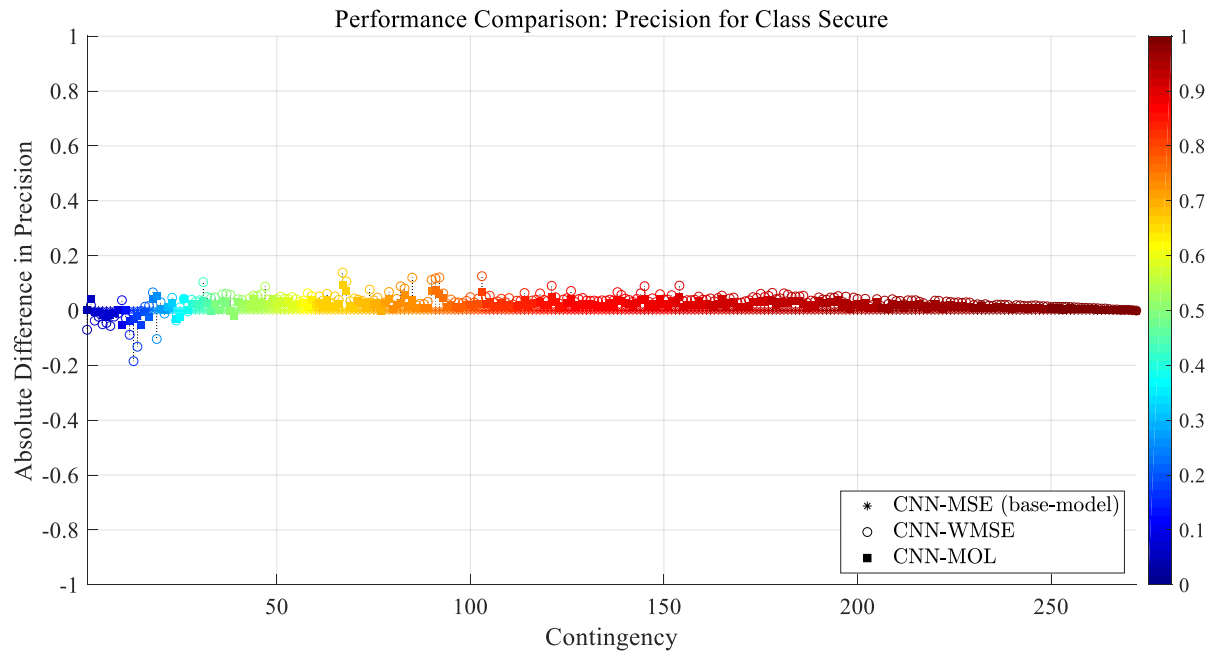


Figure 4-40 Texas 2000-bus synthetic system: precision performance comparison graph for class secure.CNN-MSE is chosen as the base-model for comparison.

Using WMSE loss improves recall for contingencies where class k is a minority. However, recall becomes worse for contingencies where class k is of majority. This can be seen from recall-precision graphs shown in Figure 4-27, Figure 4-32, and Figure 4-37. Compared to CNN-MSE recall-precision graphs, one can see that smaller circles (minorities) have shifted up (i.e., better recall) while larger circles (majorities) have shifted down. This comparison is better illustrated in performance comparison graphs shown in Figure 4-29, Figure 4-34, and Figure 4-39. From these figures, one can clearly see that CNN-WMSE has a better recall performance for minorities and vice-versa.

In terms of precision, if class k is of the dominant majority, then, the difference in precision performance between the three models is minimal. Again, this is because the number of correct predictions would be much greater than the number of false predictions. On the other hand, if class k is a minority, then, the number of correct predictions of class k becomes comparable to the number of its false predictions. Accordingly, precision tends to be low for contingencies where class k is a class of minority (typically lowest for CNN-WMSE).

As stated previously, macro-recall is recall averaged across contingencies. Now, because “insecure” is a class of minority for most contingencies, the CNN-WMSE scored the highest macro-recall, 68.7% compared to 46.6% and 18.5% for the CNN-MOL and CNN-MSE respectively. Class “marginal-insecure” is also a minority for many contingencies. Therefore, CNN-WMSE has again scored the heights macro-recall with 86.8% compared to 67.5% and 42.0% for the CNN-MOL and CNN-MSE respectively. The opposite is true for class “secure” (the dominating class for most contingencies) with CNN-WMSE scoring the lowest macro-recall, 55.9%, compared to 79.6% and 89.0% for the CNN-MOL and CNN-MSE respectively.

4.8 Conclusion

In this chapter, we proposed a Convolutional Neural Network (CNN) –based approach to the problem of static power systems contingency screening. A CNN-based multi-label classifier is designed to obtain, in a single run, the security status of the system (i.e. secure, marginally-secure, or insecure) for a pre-selected set of $N-1$ contingencies.

The IEEE 118-bus system and the larger Texas 2000-bus synthetic system were both utilized to investigate the proposed approach. In order to implement the CNN-based classifier for both test systems, we had to overcome a number of challenges. These include but are not limited to: computational requirements, learning from an imbalanced dataset, and evaluating the performance of trained CNN-based classifiers. In terms of computational challenges, we devised a new method for performing convolutional and max-pooling operations using “compressed” representations of input images which enabled the application of CNNs for the larger 2000-bus test system. As for learning, we utilized three different loss functions for training the CNN-based classifier. Those were: mean-squared error (MSE), weighted mean-squared error (WMSE), and a multi-objective loss (MOL) function combining both MSE and WMSE. Finally, for performance evaluation, we proposed two new visualization graphs, named here as recall-precision graphs and performance comparison graphs, which we found to help examine and compare the performance of the different CNN-classifiers.

Based on the obtained results, we were able to identify several tradeoffs in model performance for each class “secure”, “marginal-insecure”, and “insecure” and accordingly, we conclude the following: if most of the targeted contingencies had “insecure” and/or “marginal-insecure” as a minority class then, it is best to train the CNN-based classifier using the WMSE loss. On the other

hand, if many contingencies had a relatively high count of “insecure” and/or “marginal-insecure” cases in the dataset, then it is best to train the model with MOL. Finally, if most of the contingencies had “insecure” and/or “marginal-insecure” as a class of majority, then, MSE is the preferred choice for training the CNN-based classifier.

Chapter 5. Conclusion

5.1 Conclusion and Contributions

Chapter 2 proposed a voltage stability indicator, named TAVSI, which accounts for ZIP, exponential, and induction motor loads. The TAVSI measures the tangential angle difference between the system P-V curve and the load characteristic curve, so it provides a more reliable assessment of system voltage stability as opposed to traditional load models. The value of the TAVSI varies from π at zero loading to zero at the system point of collapse (i.e., maximum loading). The performance of this indicator was tested on both the IEEE 14-bus system and the CURENT's version of the 181-bus WECC system. The results show that TAVSI gives a good and reliable assessment of system stability. In addition, the TAVSI is calculated based on normalized sensitivities which allows for comparison among different load buses. This in turn allows for: 1) the definition of a global threshold for deciding on when to take corrective actions; and 2) the identification of weak buses where corrective actions would be the most effective. Finally, the TAVSI at a given load bus can be calculated based on either the complete system model (a model-based approach) or the Thevenin equivalent of the system at that bus (a measurement-based approach). Whether a model-based or a measurement-based approach is chosen depends on the method's applicability in the studied system.

Chapter 3 presented a data-driven and model-based hybrid reinforcement learning approach to the problem of branch overload relief in large power systems. A control agent is trained to change generators' real power output in order to adjust the power flow through the network so that branch overload is mitigated or minimized. The proposed approach was developed based on a series of investigations on why the data-driven Deep Deterministic Policy Gradient (DDPG) -based approach

fails when training the agent for large power systems. This hybrid approach utilizes generator shifting factor-driven actions (model-based actions) to help the agent learn an optimal control policy (data-driven actions). The performance of the proposed approach was tested and compared to the original DDPG-based approach on both the IEEE 118-bus and a larger 2749-bus real-world system. With the proposed approach, the agents were able to achieve 97.6% and 91.3% success rates compared to 95.5% and 0% when trained according to the original data-driven DDPG algorithm. These results show that the proposed hybrid approach performs well for large power systems and that it is superior to the data-driven DDPG-based approach.

Chapter 4 proposed a Convolutional Neural Network (CNN) –based approach to the problem of static power systems contingency screening. A CNN-based multi-label classifier is designed to obtain, in a single run, the security status of the system (i.e. secure, marginally-insecure, or insecure) for a pre-selected set of N-1 contingencies. The proposed approach was tested on IEEE 118-bus and the larger 2000-bus synthetic Texas system. For each system, three CNN-based classifiers were trained each using a different loss function. Those were: mean-squared error (MSE), weighted mean-squared error (WMSE), and a multi-objective loss (MOL) function combining both MSE and WMSE. Obtained results showed that deciding on the “best” model depends on the characteristics of the considered power system and the distribution of the class of interest (typically class “insecure”) in the dataset.

5.2 Future Work

This dissertation can be improved and extended as follows:

- As stated previously, the voltage stability index TAVSI proposed in this work is calculated based on normalized sensitivities. This allows for comparison among different load buses. This

allows for: 1) the definition of a global threshold for deciding on when to take corrective actions; and 2) the identification of weak buses where corrective actions would be the most effective. Load shedding is considered a very cost-effective solution for preventing widespread system collapse especially since voltage collapse is a low probability-high impact phenomenon. Load-shedding schemes can be classified into two categories. In the first one, the amount of load to be shed is fixed a priori. When, where, and how much to shed is usually pre-determined through extensive off-line investigations using dynamic time simulation analysis or static analysis such as V-P and Q-V curves. In the second category, the amount of load to be shed is determined using optimal power flow techniques. Now, if one intends to optimize shedding on a load-bus basis, i.e., different buses do not shed the same amount of load, then, the dimensions and accordingly the computational burden of the formulated OPF problem would be of high magnitude. Therefore, one can consider combining TAVSI with RL as an alternative approach and train an RL agent to shed load in a way that is optimal to both system operation and security.

References

- [1] H. Haes Alhelou, M. E. Hamedani-Golshan, T. C. Njenda, and P. Siano, "A survey on power system blackout and cascading events: Research motivations and challenges," *Energies*, vol. 12, no. 4, p. 682, 2019.
- [2] S. G. Ghiocel and J. H. Chow, "A power flow method using a new bus type for computing steady-state voltage stability margins," *IEEE Transactions on Power Systems*, vol. 29, no. 2, pp. 958-965, 2013.
- [3] V. Ajjarapu and C. Christy, "The continuation power flow: a tool for steady state voltage stability analysis," *IEEE transactions on Power Systems*, vol. 7, no. 1, pp. 416-423, 1992.
- [4] E. d. M. Magalhaes, A. Bonini Neto, and D. A. Alves, "A parameterization technique for the continuation power flow developed from the analysis of power flow curves," *Mathematical Problems in Engineering*, vol. 2012, 2012.
- [5] B. Gao, G. Morison, and P. Kundur, "Voltage stability evaluation using modal analysis," *IEEE transactions on power systems*, vol. 7, no. 4, pp. 1529-1542, 1992.
- [6] Y. Song, D. J. Hill, and T. Liu, "State-in-mode analysis of the power flow Jacobian for static voltage stability," *International Journal of Electrical Power & Energy Systems*, vol. 105, pp. 671-678, 2019.
- [7] W. Elballa, A. Karrar, and A. Eltom, "A novel optimization formulation for the direct computation of the voltage collapse point," in *2018 IEEE Power & Energy Society General Meeting (PESGM)*, 2018, pp. 1-5: IEEE.
- [8] A. A. Mohamed and B. Venkatesh, "Linearized voltage stability incorporation with line-wise optimal power flow," *International Journal of Electrical Power & Energy Systems*, vol. 108, pp. 232-239, 2019.
- [9] Y. Wang *et al.*, "Voltage stability monitoring based on the concept of coupled single-port circuit," *IEEE Transactions on Power Systems*, vol. 26, no. 4, pp. 2154-2163, 2011.
- [10] A. Chandra and A. K. Pradhan, "Online voltage stability and load margin assessment using wide area measurements," *International Journal of Electrical Power & Energy Systems*, vol. 108, pp. 392-401, 2019.
- [11] M. Kamel, A. A. Karrar, and A. H. Eltom, "Development and application of a new voltage stability index for on-line monitoring and shedding," *IEEE Transactions on Power Systems*, vol. 33, no. 2, pp. 1231-1241, 2018.
- [12] A. R. R. Matavalam and V. Ajjarapu, "Sensitivity Based Thevenin Index With Systematic Inclusion of Reactive Power Limits," *IEEE Transactions on Power Systems*, vol. 33, no. 1, pp. 932-942, 2018.
- [13] H. Chen, T. Jiang, H. Yuan, H. Jia, L. Bai, and F. Li, "Wide-area measurement-based voltage stability sensitivity and its application in voltage control," *International Journal of Electrical Power & Energy Systems*, vol. 88, pp. 87-98, 2017.
- [14] P.-H. Huang and T.-H. Tseng, "Analysis for effects of load characteristics on power system voltage stability," *AASRI Procedia*, vol. 2, pp. 229-234, 2012.
- [15] I. Smon, G. Verbic, and F. Gubina, "Local voltage-stability index using Tellegen's theorem," *IEEE Transactions on Power Systems*, vol. 21, no. 3, pp. 1267-1275, 2006.
- [16] D. Popović, I. Hiskens, and D. Hill, "Stability analysis of induction motor networks," *International Journal of Electrical Power & Energy Systems*, vol. 20, no. 7, pp. 475-487, 1998.
- [17] C. A. Cañizares and W. Rosehart, "Bifurcation analysis of induction motor loads for voltage collapse studies," in *Proc. North American Power Symposium (NAPS)*, MIT, 1996, pp. 559-565.
- [18] R. Yao, K. Sun, D. Shi, and X. Zhang, "Voltage stability analysis of power systems with induction motors based on holomorphic embedding," *IEEE Transactions on Power Systems*, vol. 34, no. 2, pp. 1278-1288, 2018.
- [19] H. Yuan, F. Li, H. Cui, X. Lu, D. Shi, and Z. Wang, "A measurement-based VSI for voltage dependent loads using angle difference between tangent lines of load and PV curves," *Electric Power Systems Research*, vol. 160, pp. 13-16, 2018.
- [20] T. Van Cutsem and C. Vournas, *Voltage stability of electric power systems*. Springer Science & Business Media, 2007.
- [21] W.-S. Kao, C.-J. Lin, C.-T. Huang, Y.-T. Chen, and C.-Y. Chiou, "Comparison of simulated power system dynamics applying various load models with actual recorded data," *IEEE Transactions on Power Systems*, vol. 9, no. 1, pp. 248-254, 1994.
- [22] W.-S. Kao, "The effect of load models on unstable low-frequency oscillation damping in Taipower system

- experience w/wo power system stabilizers," *IEEE Transactions on Power Systems*, vol. 16, no. 3, pp. 463-472, 2001.
- [23] I. Dobson, "The irrelevance of load dynamics for the loading margin to voltage collapse and its sensitivities," in *Bulk power system voltage phenomena III, voltage stability, security & control, Proceedings of the ECC/NSF workshop, Davos, Switzerland*, 1994.
 - [24] P. Kundur, N. J. Balu, and M. G. Lauby, *Power system stability and control*. McGraw-hill New York, 1994.
 - [25] R. D. Christie. (2000). *Power systems test case archive, University of Washington*. Available: <https://www.ee.washington.edu/research/pstca/>
 - [26] Q. Shi, F. F. Li, and H. Cui, "Analytical Method to Aggregate Multi-Machine SFR Model with Applications in Power System Dynamic Studies," *IEEE Transactions on Power Systems*, 2018.
 - [27] J. Lavei, A. Rantzer, and S. Low, "Power flow optimization using positive quadratic programming," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 10481-10486, 2011.
 - [28] X. Bai, H. Wei, K. Fujisawa, and Y. Wang, "Semidefinite programming for optimal power flow problems," *International Journal of Electrical Power & Energy Systems*, vol. 30, no. 6-7, pp. 383-392, 2008.
 - [29] B. Kocuk, S. S. Dey, and X. A. Sun, "Strong SOCP relaxations for the optimal power flow problem," *Operations Research*, vol. 64, no. 6, pp. 1177-1196, 2016.
 - [30] M. Glavic, "(Deep) Reinforcement learning for electric power system control and related problems: A short review and perspectives," *Annual Reviews in Control*, vol. 48, pp. 22-35, 2019.
 - [31] Z. Zhang, D. Zhang, and R. C. Qiu, "Deep reinforcement learning for power system applications: An overview," *CSEE Journal of Power and Energy Systems*, vol. 6, no. 1, pp. 213-225, 2019.
 - [32] J. G. Vlachogiannis and N. D. Hatziargyriou, "Reinforcement learning for reactive power control," *IEEE transactions on power systems*, vol. 19, no. 3, pp. 1317-1325, 2004.
 - [33] X. Dong, J. Liang, X. Zhang, and H. Sun, "Computation of closest steady state voltage stability bifurcation using PSO approach," in *IEEE PES Innovative Smart Grid Technologies*, 2012, pp. 1-4: IEEE.
 - [34] F. Agostinelli, G. Hocquet, S. Singh, and P. Baldi, "From reinforcement learning to deep reinforcement learning: An overview," in *Braverman Readings in Machine Learning. Key Ideas From Inception to Current State*: Springer, 2018, pp. 298-328.
 - [35] M. Kamel *et al.*, "A Reinforcement Learning Approach for Branch Overload Relief in Power Systems," in *2020 IEEE Power and Energy Society General Meeting (PESGM)*, 2020.
 - [36] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.
 - [37] R. Bellman, "The theory of dynamic programming," *Bulletin of the American Mathematical Society*, vol. 60, no. 6, pp. 503-515, 1954.
 - [38] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," 2014.
 - [39] M. Schneckentreiter, "Average Reward Adjusted Discounted Reinforcement Learning: Near-Blackwell-Optimal Policies for Real-World Applications," *arXiv preprint arXiv:2004.00857*, 2020.
 - [40] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," in *Advances in Neural Information Processing Systems*, 2018, pp. 9839-9849.
 - [41] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 6292-6299: IEEE.
 - [42] A. Trott, S. Zheng, C. Xiong, and R. Socher, "Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards," in *Advances in Neural Information Processing Systems*, 2019, pp. 10376-10386.
 - [43] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," *arXiv preprint arXiv:1709.06560*, 2017.
 - [44] G. Matheron, N. Perrin, and O. Sigaud, "The problem with DDPG: understanding failures in deterministic environments with sparse rewards," *arXiv preprint arXiv:1911.11679*, 2019.
 - [45] P. E. Gill and E. Wong, "Methods for convex and general quadratic programming," *Mathematical*

- programming computation*, vol. 7, no. 1, pp. 71-112, 2015.
- [46] T. Hester *et al.*, "Deep q-learning from demonstrations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018, vol. 32, no. 1.
 - [47] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015, pp. 448-456: PMLR.
 - [48] T. Van Laarhoven, "L2 regularization versus batch and weight normalization," *arXiv preprint arXiv:1706.05350*, 2017.
 - [49] G. Zhang, C. Wang, B. Xu, and R. Grosse, "Three mechanisms of weight decay regularization," *arXiv preprint arXiv:1810.12281*, 2018.
 - [50] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*, 2018, pp. 1587-1596: PMLR.
 - [51] A. Bhatt, M. Argus, A. Amiranashvili, and T. Brox, "Crossnorm: Normalization for off-policy td reinforcement learning," *arXiv preprint arXiv:1902.05605*, 2019.
 - [52] F. Schäfer, J.-H. Menke, and M. Braun, "Contingency analysis of power systems with artificial neural networks," in *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, 2018, pp. 1-6: IEEE.
 - [53] B. Donnot, I. Guyon, M. Schoenauer, A. Marot, and P. Panciatici, "Anticipating contingencies in power grids using fast neural net screening," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1-8: IEEE.
 - [54] S. Wang, P. Dehghanian, and L. Li, "Power grid online surveillance through PMU-embedded convolutional neural networks," *IEEE Transactions on Industry Applications*, vol. 56, no. 2, pp. 1146-1155, 2019.
 - [55] K. S. Swarup and G. Sudhakar, "Neural network approach to contingency screening and ranking in power systems," *Neurocomputing*, vol. 70, no. 1-3, pp. 105-118, 2006.
 - [56] P. Bhowmik, P. Purkait, and K. Bhattacharya, "A novel wavelet transform aided neural network based transmission line fault analysis method," *International Journal of Electrical Power & Energy Systems*, vol. 31, no. 5, pp. 213-219, 2009.
 - [57] S. Seyedtabaai, "Improvement in the performance of neural network-based power transmission line fault classifiers," *IET generation, transmission & distribution*, vol. 6, no. 8, pp. 731-737, 2012.
 - [58] A. Bahmanyar and A. Karami, "Power system voltage stability monitoring using artificial neural networks with a reduced set of inputs," *International Journal of Electrical Power & Energy Systems*, vol. 58, pp. 246-256, 2014.
 - [59] H. Zhengyou, G. Shibin, C. Xiaoqin, Z. Jun, B. Zhiqian, and Q. Qingquan, "Study of a new method for power system transients classification based on wavelet entropy and neural network," *International Journal of Electrical Power & Energy Systems*, vol. 33, no. 3, pp. 402-410, 2011.
 - [60] R. Segal, A. Sharma, and M. Kothari, "A self-tuning power system stabilizer based on artificial neural network," *International journal of electrical power & energy systems*, vol. 26, no. 6, pp. 423-430, 2004.
 - [61] O. Russakovsky *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211-252, 2015.
 - [62] R. E. Schapire and Y. Singer, "BoosTexter: A boosting-based system for text categorization," *Machine learning*, vol. 39, no. 2, pp. 135-168, 2000.
 - [63] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition," in *2012 IEEE international conference on Acoustics, speech and signal processing (ICASSP)*, 2012, pp. 4277-4280: IEEE.
 - [64] R. Mittelman, "Time-series modeling with undecimated fully convolutional neural networks," *arXiv preprint arXiv:1508.00317*, 2015.
 - [65] D. Tran, J. Ray, Z. Shou, S.-F. Chang, and M. Paluri, "Convnet architecture search for spatiotemporal feature learning," *arXiv preprint arXiv:1708.05038*, 2017.
 - [66] S. J. Plathottam, H. Salehfar, and P. Ranganathan, "Convolutional Neural Networks (CNNs) for power system big data analysis," in *2017 North American Power Symposium (NAPS)*, 2017, pp. 1-6: IEEE.

- [67] H. Ren, Z. J. Hou, B. Vyakaranam, H. Wang, and P. Etingov, "Power System Event Classification and Localization Using a Convolutional Neural Network," *Frontiers in Energy Research*, vol. 8, p. 327, 2020.
- [68] J.-M. H. Arteaga, F. Hancharou, F. Thams, and S. Chatzivasileiadis, "Deep learning for power system security assessment," in *2019 IEEE Milan PowerTech*, 2019, pp. 1-6: IEEE.
- [69] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet, "Semantic annotation and retrieval of music and sound effects," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 467-476, 2008.
- [70] Y. Prabhu and M. Varma, "Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 263-272.
- [71] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang, "Deepfashion: Powering robust clothes recognition and retrieval with rich annotations," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1096-1104.
- [72] Q. Chen, J. Huang, R. Feris, L. M. Brown, J. Dong, and S. Yan, "Deep domain adaptation for describing people based on fine-grained clothing attributes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5315-5324.
- [73] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information sciences*, vol. 250, pp. 113-141, 2013.
- [74] R. Alotaibi and P. Flach, "Multi-label thresholding for cost-sensitive classification," *Neurocomputing*, vol. 436, pp. 232-247, 2021.
- [75] R.-E. Fan and C.-J. Lin, "A study on threshold selection for multi-label classification," *Department of Computer Science, National Taiwan University*, pp. 1-23, 2007.
- [76] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine learning*, vol. 85, no. 3, p. 333, 2011.
- [77] K. Sechidis, G. Tsoumakas, and I. Vlahavas, "On the stratification of multi-label data," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2011, pp. 145-158: Springer.
- [78] A. B. Birchfield, T. Xu, K. M. Gegner, K. S. Shetye, and T. J. Overbye, "Grid structural characteristics as validation criteria for synthetic networks," *IEEE Transactions on power systems*, vol. 32, no. 4, pp. 3258-3265, 2016.
- [79] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant cnns," in *2017 international conference on 3D Vision (3DV)*, 2017, pp. 11-20: IEEE.
- [80] J. Hua and X. Gong, "A Normalized Convolutional Neural Network for Guided Sparse Depth Upsampling," in *IJCAI*, 2018, pp. 2283-2290: Stockholm, Sweden.
- [81] B. Graham, "Sparse 3D convolutional neural networks," *arXiv preprint arXiv:1505.02890*, 2015.
- [82] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.
- [83] E. Cuthill and J. McKee, "Reducing the bandwidth of sparse symmetric matrices," in *Proceedings of the 1969 24th national conference*, 1969, pp. 157-172.
- [84] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7482-7491.

Appendix

A.1 Standard Convolution

Given X , an input feature map of shape [input-height, input-width, number of input channels]; W , a convolutional filter of shape [filter-height, filter-width, filter-depth (= number of input channels), number of filters], and a layer bias b . Then, let convolving X with W result in a feature map Y of shape [output-height, output-width, number of output channels (= number of filters)]. Also, let X_h , X_w , and C represent input-height, input-width, and number of input channels. W_h , W_w and M represent filter-height, filter-width, and number of convolutional filters respectively. Y_h and Y_w represent output-height and output-width. Then, an element of Y is calculated as follows:

$$Y_{(u, v, k)} = \sum_{i=0}^{(W_h-1)} \sum_{j=0}^{(W_w-1)} \sum_{d=0}^{(C-1)} X_{(u*s+i, v*s+j, d)} * W_{(i, j, d, k)} \quad 1$$

Here, we follow a zero-based indexing scheme (as in python) where the first position of an axis is assigned the index “0”. Now, the element $Y_{(u, v, k)}$ can be calculated using matrix multiplication as follows:

$$Y_{(u, v, k)} = P^{uv} * W^k \quad 2$$

Where P^{uv} is a row vector of shape $[1, W_h \times W_w \times C]$. P^{uv} represents a 3D Patch of shape $[W_h, W_w, C]$ extracted from the input feature map X starting at $X_{(u*s, v*s, 0)}$ and then flattened into a row vector of shape $[1, W_h \times W_w \times c]$. The content of P^{uv} is detailed in (3).

$$P^{uv} = [P^{uv^0}, P^{uv^1}, \dots, P^{uv^i}, \dots, P^{uv^{(W_h-1)}}] \quad 3$$

Where $P^{uv^0}, P^{uv^1}, \dots, P^{uv^i}, \dots, P^{uv^{(W_h-1)}}$ are row vectors each of shape $[1, W_w \times C]$. The content of a given vector P^{uv^i} is detailed next in (4).

$$P^{uv^i} = [P^{uv^{i0}}, P^{uv^{i1}}, \dots, P^{uv^{ij}}, \dots, P^{uv^{i(W_w-1)}}] \quad 4$$

Where $P^{uv^{i0}}, P^{uv^{i1}}, \dots, P^{uv^{ij}}, \dots, P^{uv^{i(W_w-1)}}$ are row vectors each of shape $[1, C]$. The content of a given vector $P^{uv^{ij}}$ is detailed next in (5).

$$P^{uv^{ij}} = [X_{(u*s+i, v*s+j, 0)}, X_{(u*s+i, v*s+j, 1)}, \dots, X_{(u*s+i, v*s+j, d)}, \dots, X_{(u*s+i, v*s+j, C-1)}] \quad (5)$$

In Figure A-1, we visualize the extract, reshape process used to obtain P^{uv}

Similarly, W^k is a column vector of size $[W_h \times W_w \times C, 1]$. W^k represents filter k flattened into a column vector of size $[W_h \times W_w \times C, 1]$. The structure of W^k is detailed in (6).

$$W^k = \begin{bmatrix} W^{k^0} \\ W^{k^1} \\ \vdots \\ W^{k^i} \\ \vdots \\ W^{k^{(W_h-1)}} \end{bmatrix} \quad (6)$$

Where $W^{k^0}, W^{k^1}, \dots, W^{k^i}, \dots, W^{k^{(W_h-1)}}$ are column vectors each of shape $[W_w \times C, 1]$. The content of a given vector W^{k^i} is detailed next in (7).

$$W^{k^i} = \begin{bmatrix} W^{k^{i0}} \\ W^{k^{i1}} \\ \vdots \\ W^{k^{ij}} \\ \vdots \\ W^{k^{i(W_w-1)}} \end{bmatrix} \quad (7)$$

Where $W^{k^{i0}}, W^{k^{i1}}, \dots, W^{k^{ij}}, \dots, W^{k^{i(W_w-1)}}$ are column vectors each of shape $[C, 1]$. The content of a given vector $W^{k^{ij}}$ is detailed next in (8).

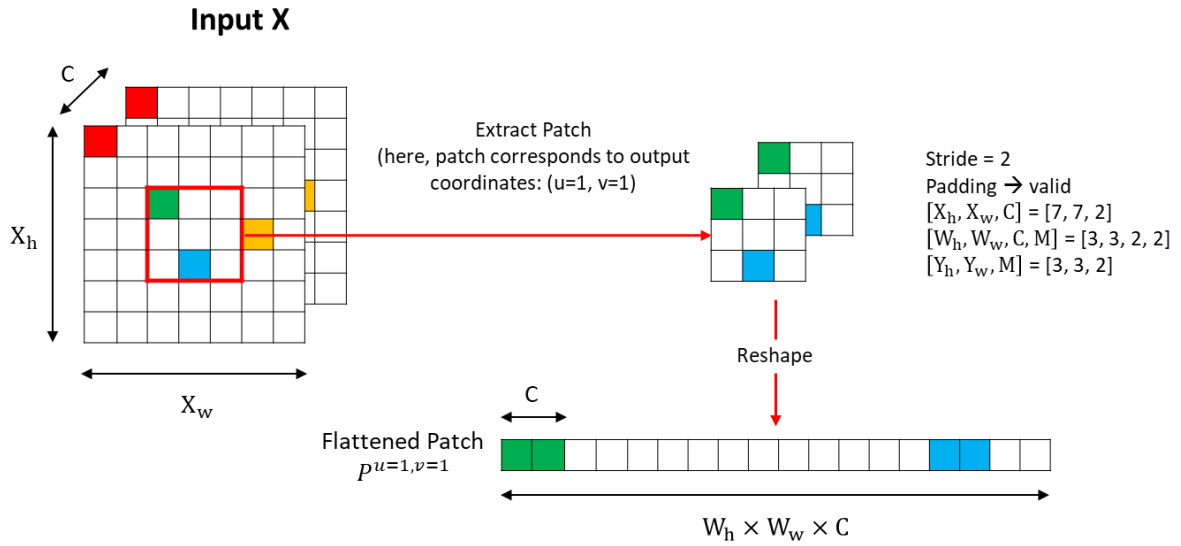


Figure A-1 Extract and reshape a 3D patch into a $[1, W_h \times W_w \times C]$ row vector. Here, X is a sparse $[7, 7, 2]$ input image. A white pixel represents no information while a colored pixel represents information.

$$W^{k^{ij}} = \begin{bmatrix} W_{(i,j,0)} \\ W_{(i,j,1)} \\ \vdots \\ W_{(i,j,d)} \\ \vdots \\ W_{(i,j,C-1)} \end{bmatrix} \quad (8)$$

Based on (1)-(8), one can obtain all elements of Y using matrix multiplication as follows:

$$Y^{flat} = P \times W^{flat} \quad (9)$$

Where: Y^{flat} is a 2D matrix of shape $[Y_h \times Y_w, M]$ representing a “flattened” version of the output image Y. Each row of Y^{flat} represents a vector slice of Y taken across its depth dimension.

$$Y^{flat} = \begin{bmatrix} Y^{00} \\ Y^{01} \\ \vdots \\ Y^{uv} \\ \vdots \\ Y^{(Y_h-1)(Y_w-1)} \end{bmatrix} \quad (10)$$

Where:

$$Y^{uv} = [Y_{(u,v,0)} \quad Y_{(u,v,1)} \quad \cdots \quad Y_{(u,v,k)} \quad \cdots \quad Y_{(u,v,M-1)}] \quad (11)$$

P is a 2D matrix of shape $[Y_h \times Y_w, W_h \times W_w \times C]$. P represents the stack of 3D patches extracted from X and flattened into row vectors each of shape $[1, W_h \times W_w \times C]$.

$$P = \begin{bmatrix} P^{00} \\ P^{01} \\ \vdots \\ P^{uv} \\ \vdots \\ P^{(Y_h-1)(Y_w-1)} \end{bmatrix} \quad (12)$$

W^{flat} is a 2D matrix of shape $[W_h \times W_w \times C, M]$. Each column of W^{flat} represents a flattened filter.

$$W^{flat} = [W^0 \quad W^1 \quad \cdots \quad W^k \quad \cdots \quad W^{(M-1)}] \quad (13)$$

In Figure A-2 we show a visual example of how to perform convolution using matrix multiplication

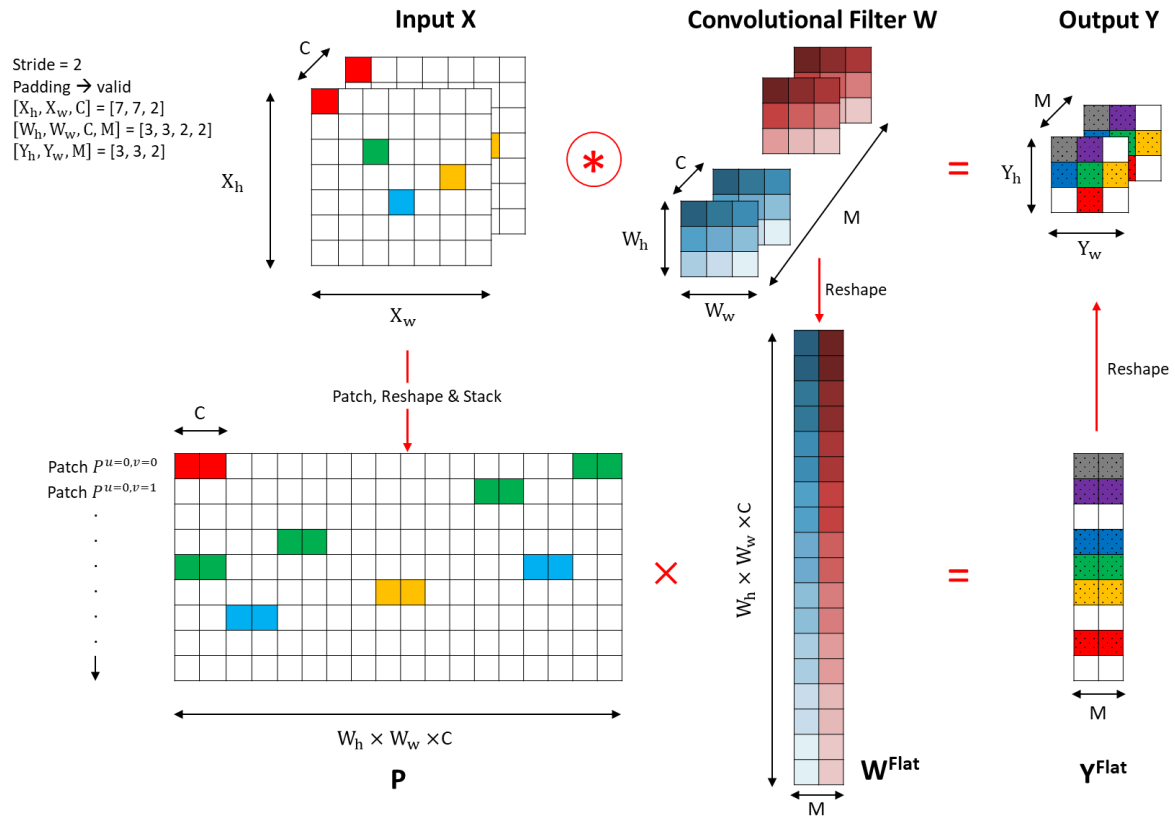


Figure A-2 Standard convolution performed using a set of patch, reshape, and matrix multiplication operations.

A.2 Sparse Convolution: Proposed Method

In this section, we show how standard convolution can be performed using a “compressed” representation of sparse feature maps.

Given Z , a 3D sparse matrix of shape $[Z_h, Z_w, C_z]$. Here, we assume all channels of Z have the same 2D adjacency matrix A_z . Accordingly, we define a compressed representation of sparse Z , csZ to be a 2D array of shape $[nnz(A_z), C_z]$. Each row of csZ represents a non-zero vector slice taken from Z at a given (height, width) coordinates across the depth dimension of Z . The (height, width) coordinates of each vector slice are stored in a separate array, named K . csZ can also be seen as a stack of column vectors each containing the non-zero elements of a given input channel of Z .

Based on the above general definitions, let’s introduce the following notations: Let csX denote the compressed representation of input feature map X , K_X denote the key-map that contains (height, width) coordinates of non-zero vector slices of X . Now, let p denotes a row index of K_X . Accordingly, let $m = K_X(p, 0)$ and $n = K_X(p, 1)$. Then,

$$csX^p = X^{mn} \quad (14)$$

where X^{mn} is the non-zero vector slice taken from X across its depth dimension starting at $X_{(m,n,0)}$

$$X^{mn} = [X_{(m,n,0)} \quad X_{(m,n,1)} \quad \cdots \quad X_{(m,n,d)} \quad \cdots \quad X_{(m,n,C-1)}] \quad (15)$$

Similarly, Let csY denote the compressed representation of input feature map Y , K_Y denote the key-map that contains (height, width) coordinates of non-zero vector slices of Y . Now, let q denotes a row index of K_Y . Accordingly, let $u = K_Y(q, 0)$ and $v = K_Y(q, 1)$. Then,

$$csY^q = Y^{uv} \quad (16)$$

where Y^{uv} is the non-zero vector slice taken from Y across its depth dimension starting at $Y_{(u,v,0)}$

$$Y^{uv} = [Y_{(u,v,0)} \quad Y_{(u,v,1)} \quad \cdots \quad Y_{(u,v,k)} \quad \cdots \quad Y_{(u,v,M-1)}] \quad (17)$$

It is interesting to note that Y^{flat} has the same format as csY . In fact, csY is the set of Y^{flat} rows that have at least one non-zero element. Now, in a manner similar to (9), one can write sparse convolution using matrix multiplication as follows:

$$csY = Q \times W^{flat} \quad (18)$$

Where:

$$Q = \begin{bmatrix} Q^0 \\ Q^1 \\ \vdots \\ Q^q \\ \vdots \\ Q^{(N_Y-1)} \end{bmatrix} \quad (19)$$

Here, Q^q represent the image patch P^{uv} taken at $u = K_Y(q, 0)$, and $v = K_Y(q, 1)$ coordinates.

As shown previously in (3)-(5), P^{uv} is made up of smaller P^{uvij} vectors. Now, comparing (5), (14), and (15), one can see that P^{uvij} is either: a row vector of csX or a row vector of zeros (let's call it ground state vector g). Now, let H be a hash table that relates “rows” of csX to “rows” of csY in accordance with (1). In theory, one can use H , csX , and g to build the Q matrix. However, in practice, this cannot be implemented using standard tensor-flow operations. The works of [81, 82] refrain from building a Q matrix. Instead, they go for calculating csY using H and csX . In [81], the authors developed their own code to enable looping over csY tensor and calculate the elements of csY one by one using H and csX . On the other hand, the authors of [82] propose to calculate csY as the sum of W_h tensors. Each tensor is calculated using a sequence of gather, matrix-multiplication, and scatter operations. This method requires two hash tables: one for gather operation, the other for scatter operation.

In this work, we develop a procedure for building the Q matrix using standard TensorFlow operations. First, we redefine the “compressed” representation of X and Y to include a row vector of zeros representing the ground state vector g . Let $\widetilde{csX}$ and $\widetilde{csY}$ denote this new definition of X and Y . Accordingly,

$$\widetilde{csY} = \tilde{Q} \times W^{flat} \quad (20)$$

Where $\tilde{Q} \equiv \begin{bmatrix} G \\ Q \end{bmatrix}$, G is a row vector of zeros.

Now that g is included in $\widetilde{csX}$ one can easily use standard tensor-flow operations to build $\tilde{Q}$. Let $H^0, H^1, \dots, H^{(N_Y-1)}$ be the sets of row indices of $\widetilde{csX}$ needed as in (19) to calculate row vectors $\widetilde{csY}^0, \widetilde{csY}^1, \dots, \widetilde{csY}^{(N_Y-1)}$. Let H be the hash table that combines all $H^0, H^1, \dots, H^{(N_Y-1)}$ sets. Using H , one can create $\tilde{Q}$ using standard gather and reshape operations of TensorFlow as follows:

$$\tilde{Q} = \text{reshape}(\widetilde{csX}(H, :), \text{shape} = [N_Y, W_h \times W_w \times C]) \quad (21)$$

It should be noted that a layer bias will be added to $\widetilde{csY}$. Accordingly, the elements of its first row (i.e. g) would no longer be zeros. In order to correct for this, one needs to multiply the output of the convolutional layer by a column vector whose elements are all ones except for the first element which is set to zero. In Figure A-3, we visualize an example of how to perform sparse convolution using our proposed approach.

As mentioned above, the hash table H relates “rows” of $\widetilde{csX}$ to “rows” of $\widetilde{csY}$. Now, each row q of $\widetilde{csY}$, (except row 0) is associated with a (u, v) coordinates pair where $Y^{uv} \neq 0$. Accordingly, based on (2)-(5), (14), and (15); each $q \in \{0, 1, \dots, (N_Y - 1)\}$ becomes associated with a set of $W_h \times W_w$ row indices of csX . Assume K_Y is known then, based on (1) one can identify $H = [H^0, H^1, \dots, H^{(N_Y-1)}]$ as given in Table A.1.

Table A- 1 Pseudo-code to build hash table H

Data: $X_h, X_w, Y_h, Y_w, W_h, W_w, K_X$ and K_Y

Results: H

$l = 0$

$\mathcal{R}_X = \text{zeros}(X_h \times X_w)$

$\mathcal{A}_Y = \text{zeros}(Y_h \times Y_w)$

$N_X = \text{number of rows of } K_X$

For $\mathbf{p} = \mathbf{0}$ to $\mathbf{N}_X$ do

$m = K_X[p, 0]$

$n = K_X[p, 1]$

$l = l + 1$

$index = m \times X_h + n$

$\mathcal{R}_X[index] = l$

$N_Y = \text{number of rows of } K_Y$

$H = \text{zeros}(N_Y \times W_h \times W_w)$

For $\mathbf{q} = \mathbf{0}$ to $\mathbf{N}_Y$ do

$u = K_Y[q, 0]$

$v = K_Y[q, 1]$

 For $\mathbf{i} = \mathbf{0}$ to $\mathbf{W}_h$ do

$m = u * s + i$

 For $\mathbf{j} = \mathbf{0}$ to $\mathbf{W}_w$ do

$n = v * s + j$

$index_1 = m \times X_h + n$

$index_2 = q \times W_h \times W_w + i \times W_w + j$

$H[index_2] = \mathcal{R}_X[index_1]$

Now, to obtain K_Y we also utilize the indexing relations of (1). Given p, m and n , then, based on (1) one can identify the set of all “valid” (u, v) combinations corresponding to m and n and therefore, p as follows: let IJ be the set of all possible (i, j) pairs i.e. $IJ = \{(i, j): i \in \{0, 1, \dots, (W_h - 1)\}, j \in \{0, 1, \dots, (W_w - 1)\}\}$, then:

$$UV^r = \left\{ (u, v): u = \frac{m-i}{s}, v = \frac{n-j}{s}, (i, j) \in IJ, 0 \leq u < W_h, 0 \leq v < W_w, u, v: \text{integer} \right\} \quad (22)$$

Now, by definition, K_Y is the set of all “valid” (u, v) combinations. Therefore, one can obtain K_Y as follows:

Step-1: for all row indices of K_X ($r = 0, 2, \dots, N_X - 1$); use (22) to obtain $UV^0, UV^1, \dots, UV^{(N_X-1)}$

Step-2: $K_Y = \text{unique}([UV^0, UV^1, \dots, UV^{(N_X-1)}])$

The pseudo-code for obtaining H is given in Table A.2

Table A- 2 Pseudo-code to build output key-map K_y and hash table H

Data: $X_h, X_w, Y_h, Y_w, W_h, W_w$, and K_X

Results: K_Y and H

$l = 0$

$\mathcal{R}_X = \text{zeros}(X_h \times X_w)$

$\mathcal{A}_y = \text{zeros}(Y_h \times Y_w)$

$N_X = \text{number of rows of } K_X$

For $\mathbf{p} = \mathbf{0}$ to N_X do

$m = K_X[p, 0]$

$n = K_X[p, 1]$

$l = l + 1$

$index = m \times X_h + n$

$\mathcal{R}_X[index] = l$

 For $\mathbf{i} = \mathbf{0}$ to W_h do

$u = \frac{m - i}{s}$

 If $0 \leq u < W_h$ and u : integer

 For $\mathbf{j} = \mathbf{0}$ to W_w do

$v = \frac{n - j}{s}$

 If $0 \leq v < W_w$ and v : integer

$index = u \times Y_h + v$

$\mathcal{A}_y[index] = 1$

$K_Y = \text{find}(\mathcal{A}_y == 1)$

$N_Y = \text{number of rows of } K_Y$

$H = \text{zeros}(N_Y \times W_h \times W_w)$

For $\mathbf{q} = \mathbf{0}$ to N_Y do

$u = K_Y[q, 0]$

$v = K_Y[q, 1]$

 For $\mathbf{i} = \mathbf{0}$ to W_h do

$m = u * s + i$

 For $\mathbf{j} = \mathbf{0}$ to W_w do

$n = v * s + j$

$index_1 = m \times X_h + n$

$index_2 = q \times W_h \times W_w + i \times W_h + j$

$H[index_2] = \mathcal{R}_X[index_1]$

Vita

Mariana Kamel (GS'17) received the B.Sc. in Electrical Engineering from the University of Khartoum, Sudan, in 2012 and the M.S in Electrical Engineering from the University of Tennessee at Chattanooga in 2016. She joined Professor Fangxing Li's group at the University of Tennessee at Knoxville in 2017 to pursue a Ph.D. degree in Electrical Engineering. Her research interests include power systems modeling, power systems protection, power systems stability and control, and artificial intelligence.