IIT Indore Internship

Mridul Sharma

July 2024

1 Introduction

Ensuring the reliable and efficient operation of power systems is crucial for maintaining a stable and secure electricity supply. In today's power grid, Phasor Measurement Units (PMUs) are key players. They provide real-time data on electrical quantities like voltage, current, and frequency from various locations within the grid. This data is essential for detecting and classifying faults, which, if not addressed promptly, can lead to serious disruptions or even blackouts. Fault classification is critical for protecting power systems. It involves pinpointing the type and location of a fault so that the right corrective actions can be taken. While traditional methods like Support Vector Machines (SVM) and Kernel Density Estimation (KDE) have been widely used, they often face challenges with the complexity and high dimensionality of PMU data, leading to potential inaccuracies. In this project, we introduce a new approach to fault classification using a Convolutional Neural Network (CNN) with 3D Convolutional layers (Conv3D). By taking advantage of both the spatial and temporal aspects of PMU data, our model aims to enhance the accuracy and reliability of fault classification, ultimately contributing to a more dependable power system.

2 Motivation

The rapid evolution of power systems, driven by the integration of renewable energy sources and the increasing demand for reliable electricity, has introduced significant challenges in fault detection and classification. As power grids become more complex, the ability to accurately identify and address faults is crucial for preventing disruptions and ensuring system stability. Existing fault classification techniques face several limitations as they often struggle with high-dimensional and complex data. The detailed, real-time measurements provided by Phasor Measurement Units (PMUs) generate vast amounts of data, which can overwhelm conventional models. These methods may also require extensive manual feature engineering and can be computationally intensive, making them less suitable for real-time applications. Furthermore, many existing models are not well-equipped to handle the dynamic nature of modern power systems, which are characterized by frequent load changes, fluctuating power generation, and evolving grid conditions. This can lead to models that are either too rigid or require constant updates to remain effective. Moreover, traditional techniques often fall short of capturing the intricate temporal and spatial relationships within PMU data. The ability to automatically learn and adapt to these complex patterns is essential for accurate fault classification, yet many existing models lack this capability. Given these challenges, there is a clear need for more advanced approaches that can better manage the complexities of PMU data and adapt to the evolving nature of power systems. Convolutional Neural Networks (CNNs), particularly those utilizing 3D convolutional layers, offer a promising solution. By automatically learning both spatial and temporal features from the data, CNNs can enhance classification accuracy and robustness, making them well-suited for real-time fault detection and contributing to the overall reliability of modern power systems.

3 Report Organisation

This report is organized into several sections to provide a comprehensive overview of the project:

1. Introduction: Outlines the importance of fault detection in power systems and the role of PMUs in monitoring these systems.

2. Motivation: Discusses the limitations of traditional fault classification methods and the need for advanced approaches like CNNs.
4. Proposed Methodology: Describes the dataset, data processing steps, and the proposed CNN architecture used for fault classification.
5. CNN Architecture: Details the specific layers and configurations of the CNN model, explaining how it processes PMU data.
6. Results and Discussion: Present the performance metrics of the CNN model, compare it with traditional methods, and discuss the findings.
7. Conclusion: Summarizes the key outcomes of the project and suggests potential future work.
8. Acknowledgments: Acknowledges the support and guidance received during the internship.

My contributions to the project are that I played a key role in developing a Convolutional Neural Network (CNN) for fault classification. I crafted the model's architecture, fine-tuned its parameters, and trained it with PMU data to ensure it performed well. I also took charge of data preprocessing—organizing and normalizing data from various CSV files, extracting essential features, and carefully splitting the dataset for training and testing. To make the whole process smoother, I automated the data handling and model training steps and documented everything thoroughly so others could understand and replicate the work easily.

4 Proposed Methodology

- Our dataset is derived from eight Phasor Measurement Units (PMUs), each capturing data across different fault conditions. To analyze these conditions, we focus on several key features: the Frequency (Freq), which measures the rate of the system's oscillation; Zero Sequence Voltage (V_0) magnitude, indicating imbalances in the system; Positive Sequence Voltage (V_1) magnitude, reflecting the normal operating state; Negative Sequence Voltage (V_2) magnitude, which helps in identifying asymmetries; and the Phase Voltage magnitudes (V_A, V_B, V_C), representing the voltages across the different phases. These features together offer a detailed snapshot of the electrical system's performance and its response to various faults. We use these parameters as our features to train our proposed model because for fault classification, if we use only frequency to classify different types of faults, then we might miss out on the effect caused by the voltage on the system, and the fault might be classified incorrectly and thus we also include the magnitudes of phase voltages so that such error doesn't occur from our side, and not all faults show changes in zero sequence, positive sequence, negative sequence voltages when they occur, but the phase voltages show changes in all cases. Hence, We use the above given parameters as our features.

- All the cases used in our dataset are generated using an IEEE 14 bus System. They are given in table 1:

Table 1: Cases Generated.

Cases	No. Cases
LL	100
LLG	131
LLLG	119
SLG	135
L-Off	162
L-On	137
LR	135
LT	135

- Only data processing that happens on the dataset is while executing the model, as given below:

  Normalization: For normalization of our dataset we use "StandardScaler". This method standardizes

the data so that each feature has a mean of 0 and a standard deviation of 1, this ensures that all features have similar scales. This also speeds up the training process and prevents gradient issues.

The standardization formula is given by (1):

$$X_{scaled} = \frac{X - \mu}{\sigma} \tag{1}$$

where: X is the original feature value, μ is the mean of the feature, and σ is the standard deviation of the feature.

4.1 Input Data Preparation for CNN

- Our Dataset comprises of multiple CSV files containing data extracted from PMUs and organised into folders representing different fault classes Each CSV file corresponds to a distinct instance of a fault and includes measurements of electric currents (I), voltages (V), frequency (freq), rate of change of frequency (rocof), and fractional measurements (frac). For example, a "line trip" fault is characterised by significant changes in currents and voltages, with potential frequency fluctuations at the moment of the trip, a "load change" event demonstrates changes in currents and voltages due to the reduction or increase in load, with initial frequency fluctuations stabilising over time, An "LLG" fault (Line-to-Line-to-Ground) shows a combination of significant changes in currents and voltages between two phases and the ground, with distinctive patterns in frequency and rate of change of frequency reflecting the fault's impact on the system, etc, as shown in figure 1.

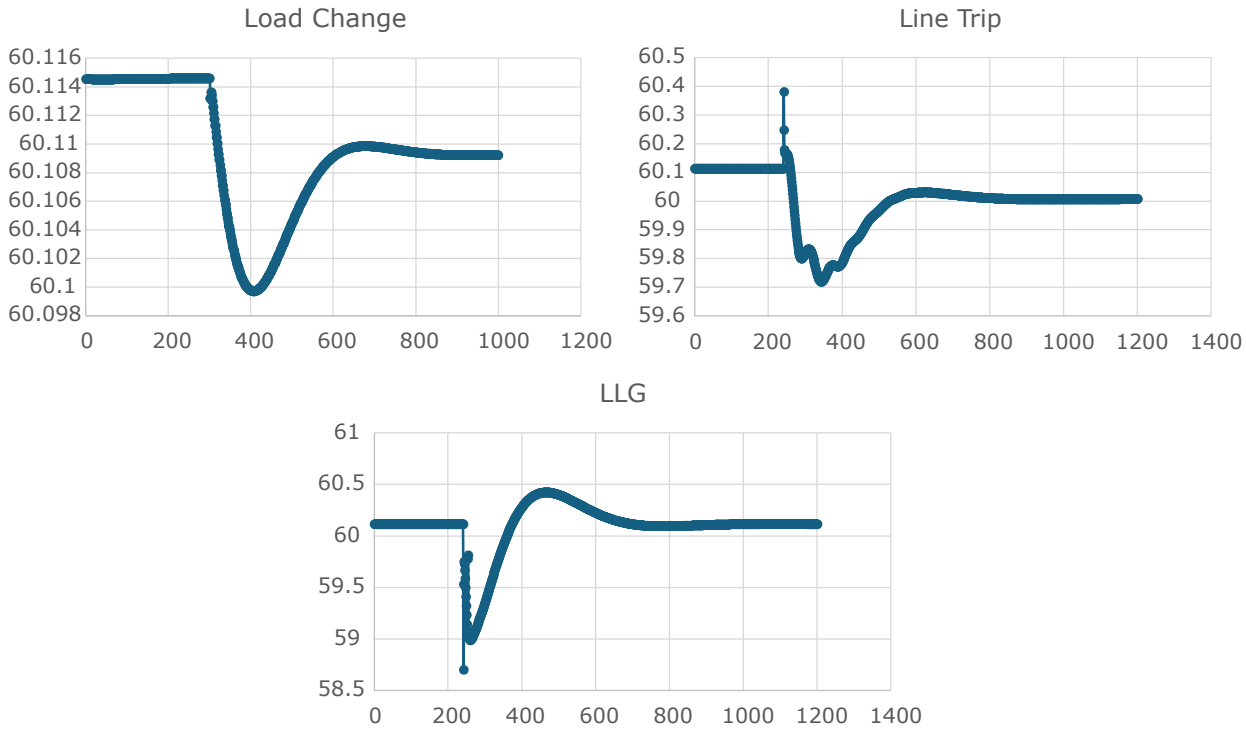

Figure 1: Graphs for frequency for Load Change, Line Trip and LLG Fault.

.

- The input data for the CNN was prepared by extracting all the necessary features from each CSV file belonging to the different classes of faults available to us. Then, the extracted data was normalized using StandardScaler as explained above, and then it was split into training and testing datasets, with

the training dataset containing 80% of our total data as input and the remaining being used as training data.

4.2 Proposed CNN Architecture

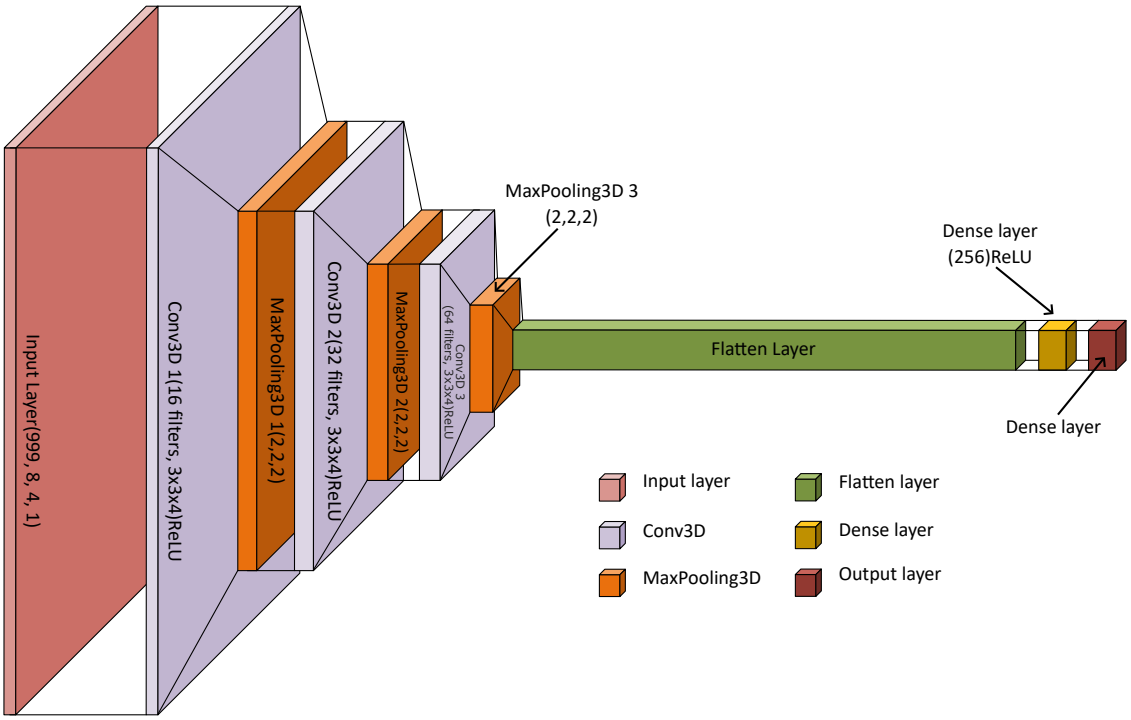

Figure 2: Proposed CNN architecture.

- We have 3 Conv3D layers which are used to extract the spatial and temporal features from our multi-dimensional data. We increase the number of filters with each layer in order to capture more complex patterns.
- Each Conv3D layer is followed by a Maxpooling3D layer to reduce the dimensionality and the computational load, while retaining the important features. This makes the model more efficient.
- After this comes the Flatten layer which converts 3D maps to 1D feature vectors, further preparing them for the Dense layer.
- Dense layer is used to learn high level representations from the extracted features. In this layer we use L2 regularisation to prevent over-fitting of our data.
- Finally comes the Output layer, here Softmax activation is used to convert the logits (raw, unnormalized predictions) to probabilities that are suitable for multi-class classification.

5 Results and Discussion

- Our Proposed model is made using CNN and is trained on 843 cases, taking 63.265 seconds to train and giving an accuracy of 100% and tested on 211 cases, taking 0.486 seconds to test and giving an accuracy of 99.05% with the testing time per case being 2.3 ms.

Table 2: Comparison of Accuracy, Precision, Recall and F1-Score.

Model	Accuracy	Precision	Recall	F1-Score
SVM	0.6821	0.6798	0.6821	0.6385
KDE	0.8409	0.8657	0.8409	0.8365
Proposed	0.9905	0.9908	0.9905	0.9905

Table 2 shows the high values of accuracy (99.05%), precision (99.08%), recall (99.05%) and F1 score (99.05%) over the test data demonstrate our model's ability to generalize well on unseen data thus capturing the intricate patterns in the PMU signals, then the significant reduction in the number of false positives and false negatives which then ensures that non-fault conditions are not mistakenly classified as faults and vice-versa. Also, it confirms the model's overall effectiveness and robustness in handling our dataset, making better and more accurate predictions.

- We have compared our proposed model with two existing models, namely the KDE model from the paper "Real-Time Event Classification in Power System With Renewables Using Kernel Density Estimation and Deep Neural Network" [2] and the SVM model from the paper "A Methodology for Fault Detection and Classification using PMU Measurements" [1]. Figures 3 and 4 shows the model loss and model accuracy for the proposed method and KDE based method.

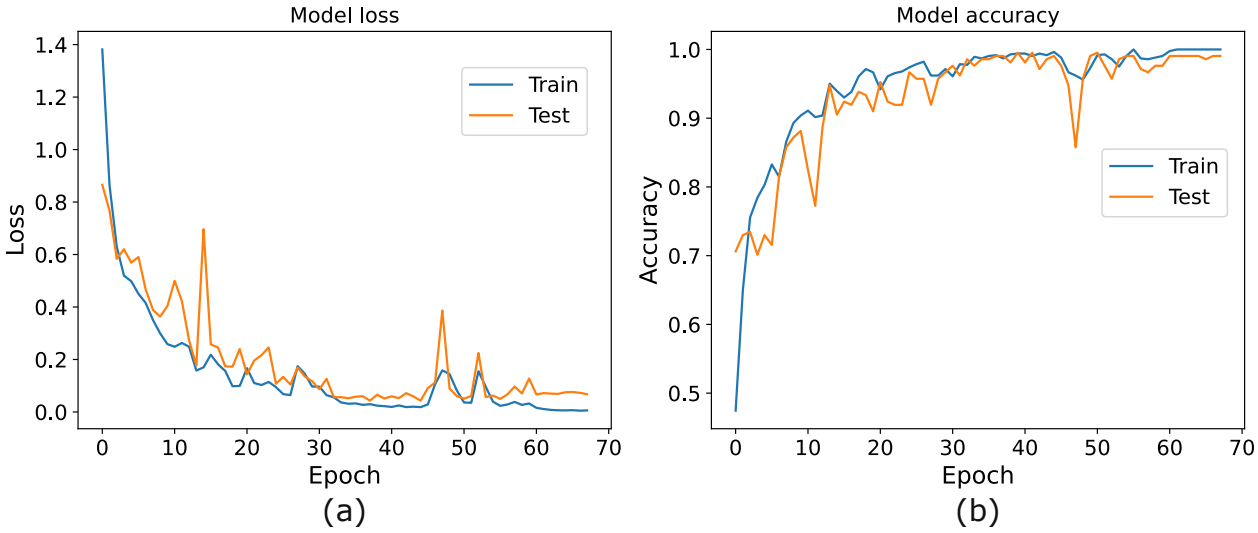

Figure 3: (a) Loss and (b) Accuracy graphs for training and testing for proposed model.

- In the SVM model, we have simply used SVM classification using the 'RBF' kernel, and according to the paper, we have to take the features as RMS value of voltage and zero sequence current, and thus we have taken the 'Fault' classes (e.g. LL, LLG, SLG, etc.) leaving our load change classes. We can also say that the SVM model is incapable of identifying Load Change.

- In the KDE model we take the strongest voltage and frequency signal from our dataset of 8 PMUs and generate a 3D-Distribution matrix from it and then use KDE to estimate the 3D-Probability Density Function and then further extract our required features to train our data from the 3D-PDFs (Mean, Kurtosis, Skewness, Standard deviation, and Variance) and then perform DNN training. We have followed their method to generate the 3DDMs as much as we can in order to get similar plots. First, we generate new CSV files containing only the strongest signals of voltage (V_A, V_A, V_C) and frequency from all the PMU, and we do this for all the classes available to us individually. Then, we use linear grid interpolation instead of the NNGI method to get the 3DDM for each case we have in comparison to the given 3DDM, which is a combination of around 50 examples. This whole process is very tedious and lengthy compared to our model, for which we only have to run a single code. Also, they have used

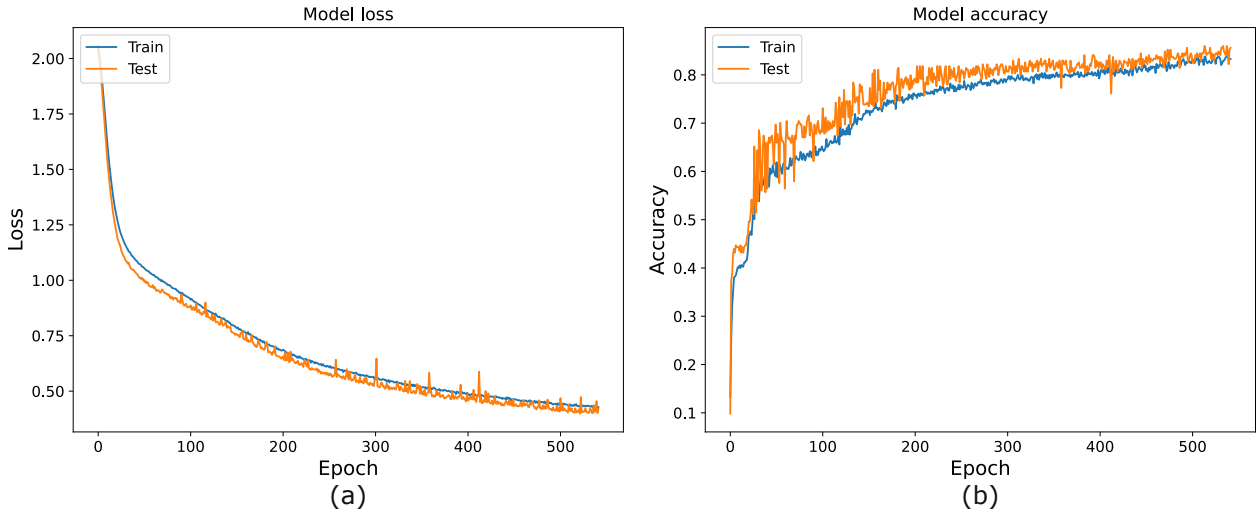

Figure 4: (a)Loss and (b)Accuracy graphs for training and testing for KDE model.

6 different optimisers (or methods) to train their model, namely ADAM, SGD, RMSP, ADAGRAD, ADAD, and ADAMAX.

- As we compare the proposed model with the KDE model, we start by comparing their loss and accuracy curves for training and testing. We observe that for both processes, in the case of KDE, the loss is over 2, whereas, for our proposed model, the training loss is less than 1.4, and the testing loss is less than 1. This shows that the proposed model has a lower number of instances where the predictions are wrong as compared to KDE. This is also apparent if we look at their test accuracy, which is 99.05% for our proposed model and 84.09% for the KDE model.

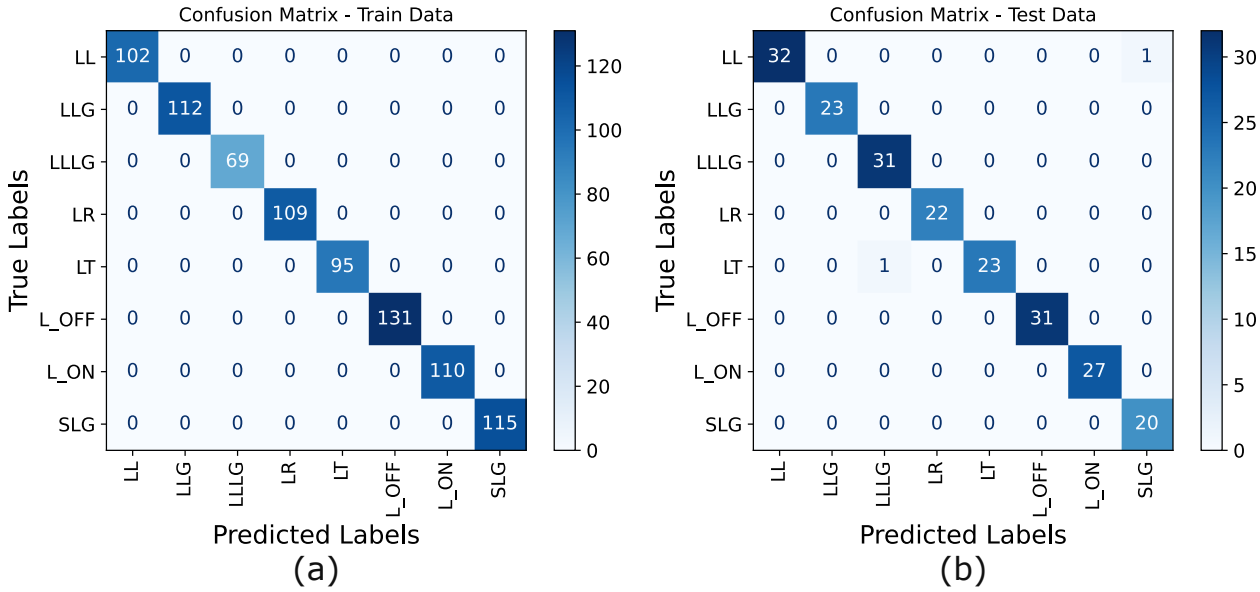

Figure 5: (a) Confusion matrix for training dataset, (b) Confusion matrix for testing dataset.

- The proposed model is advantageous over KDE and SVM as for SVM it require handcrafted features and struggle with high-dimensional data and it doesn't consider the relation between the different features, while in KDE, it may not be as effective in capturing spatial relationships without specialized architectures whereas these problems don't occur in the proposed model. In the proposed model the

only processing of data that is required is normalsation whereas in KDE we need to perform few steps as explained above.

- Also, in the proposed model all events (i.e. every timestamp from every case) from our datasets are considered for training and testing, whereas in KDE only the event of the strongest signals from each case is considered which might not be beneficial all the time.
- Comparison of test time:

  The Proposed model takes more time to test and train compared to other 2 models but given better results than them in return as given in table 3.

Table 3: Comparison of Training and Testing Time.

Model	Training Time (sec)	Test Time (sec)
SVM	0.008	0.022
KDE	22.782	0.101
Proposed	63.265	0.486

- The accuracy, precision, recall and F1 score for the 3 models are given in table 2:
- Train and Test confusion matrices for all the models for comparison:

  SVM model: Shown in figure 6

  The confusion matrix highlights that the SVM fault classification method has few limitations. It shows that the model has difficulty distinguishing between different fault types, often confusing one type with another. As a result, many faults are incorrectly classified, such as line trips with LLG, SLG, and so on, which significantly impacts the accuracy of the classification process. This misclassification issue not only reduces the reliability of the results but also suggests that the model's ability to differentiate between fault types is inadequate. The problem is caused by the model's inability to properly process large amounts of data, which increases the complexity of the training.

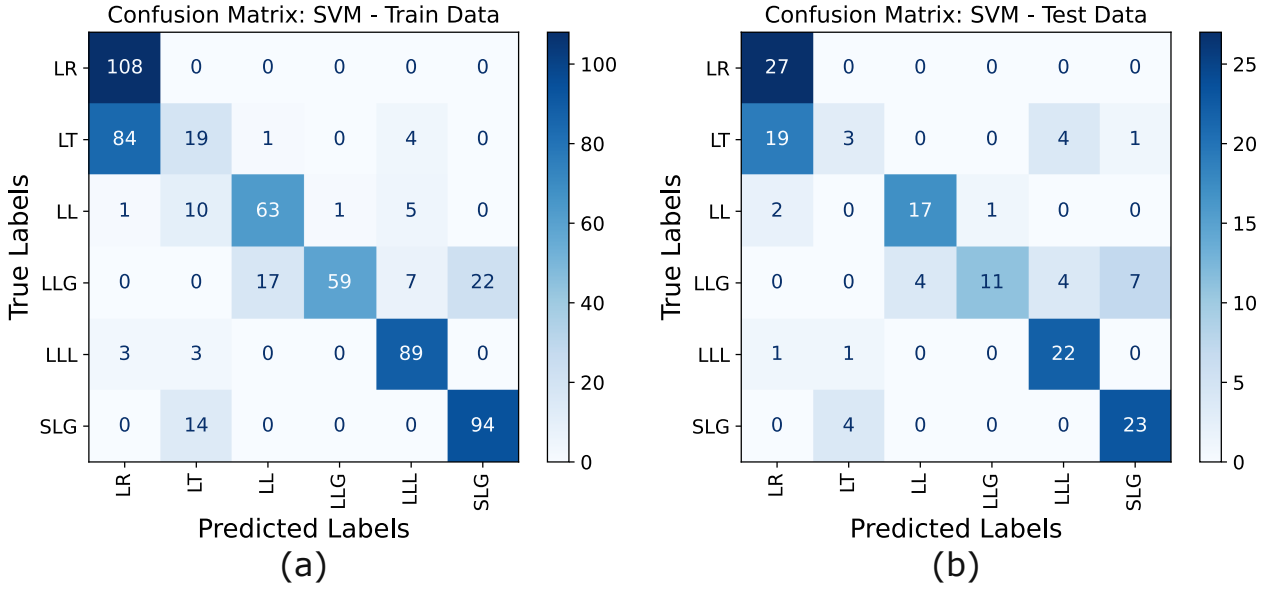

Figure 6: Confusion Matrices for SVM.

KDE model: Shown in figure 7

The confusion matrix indicates that the KDE model performs better than the SVM model in fault classification. While the SVM model struggled significantly with distinguishing between fault types,

this shows a marked improvement. It successfully identifies most faults but encounters challenges when differentiating between similar fault types. Specifically, it has difficulty distinguishing between faults such as line trip and line reclosure, load change faults (like load off and load on), and among various fault categories like LL (Line-to-Line), LLG (Line-to-Line-to-Ground), LLLG (Line-to-Line-to-Line-to-Ground), and SLG (Single-Line-to-Ground). This is because in the KDE model, all these different types of faults are taken as the same class, and thus, it confuses the model when taken separately. Now, these challenges are something that doesn't occur with our proposed model, making it better than both the SVM and KDE models.

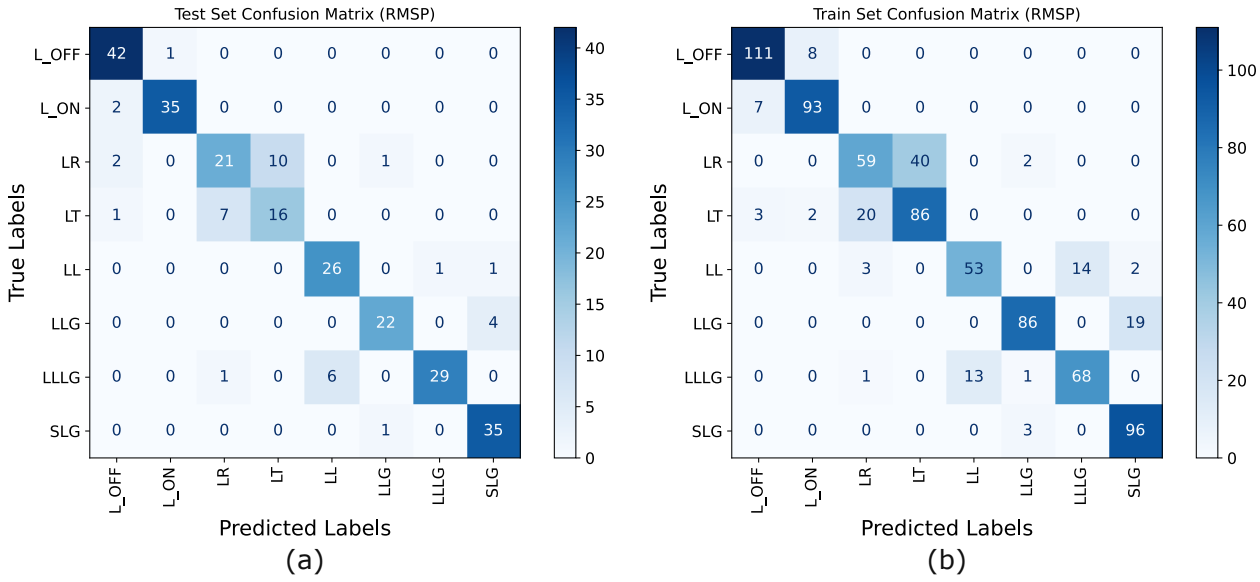

Figure 7: Confusion Matrices for KDE.

6 Conclusion

In this project, we aimed to improve fault classification in power systems by developing a Conv3D-based CNN model using PMU data. Our model achieved a high accuracy of 99.05% on the test data, with strong precision, recall, and F1 scores, proving its effectiveness in handling the complexities of modern power grids. Compared to traditional methods like SVM and KDE based models, our CNN significantly reduced errors, especially in distinguishing different fault types. While the training process was more time-consuming, the improved accuracy and reliability make it a valuable tool for real-time fault detection, ultimately contributing to the stability and security of power systems. This work highlights the potential of deep learning in solving complex challenges in power systems, setting the stage for future advancements in the field.

7 Acknowledgment

I would like to express my heartfelt gratitude to the Indian Institute of Technology (IIT) Indore for allowing me to undertake this internship. I sincerely thank Professor Trapti Jain for their invaluable guidance and support throughout the project. I also appreciate the technical support and excellent facilities provided, which were crucial for the successful completion of my work. This experience has been incredibly enriching, and I am grateful for all the resources and assistance extended to me.

References

[1] Amit Jain, T C Archana, and M B K Sahoo. A methodology for fault detection and classification using pmu measurements. In *2018 20th National Power Systems Conference (NPSC)*, pages 1–6, 2018.

[2] Ravi Yadav, Shristi Raj, and Ashok Kumar Pradhan. Real-time event classification in power system with renewables using kernel density estimation and deep neural network. *IEEE Transactions on Smart Grid*, 10(6):6849–6859, 2019.