INTEGRATING ARTIFICIAL INTELLIGENCE FOR ENHANCED FAULT DETECTION INPOWER TRANSMISSION SYSTEMS: A SMART GRID APPROACH

Nijat Taghizad, Pietro Lio
Cambridge University, UK

How to cite: Taghizad, N. & Lio, P. (2026). Integrating Artificial intelligence for enhanced fault detection in power transmission systems: A smart grid approach. *Journal of IT & AI, 9*(10), 1-6.

ABSTRACT

The primary objective was to integrate artificial intelligence for enhanced fault detection inpower transmission systems: a smart grid approach; which was tested by establishing a new way of detecting faults within electrical systems; filling in the gaps of current research and work done within the field; classifying the type of faults according to their nature (gate-codes from the dataset), and evaluate the model's accuracy in classifying these faults; and comparing different types of machine learning models, including traditional ones that utilize established algorithms and a custom deep learning model. The study used experimental research designs. The methodology and procedure involved using a deep learning model aimed at enhancing fault prediction capabilities. The study utilized a simulated dataset representing a three-phase transmission line to enrich training data for a neural network. The main outcomes indicated that the neural network achieved higher accuracy in fault detection compared to conventional methods. Furthermore, the integration of machine learning algorithms into electrical systems promises a more robust and reliable fault detection mechanism, reducing false alarms and improving operational efficiency. In conclusion, machine learning transforms fault diagnostics in electrical power systems, paving the way for more advanced monitoring solutions.

Keywords: Artificial intelligence, fault detection, integrating, power transmission systems, and smart grid approach.

INTRODUCTION

Power grids are a crucial part of people's daily lives, as they are the foundation behind the reason why people have working electricity in homes, offices, and other industrial locations. These systems efficiently distribute electricityand energy to thousands of places. However, due to their sheer complexity and the components involved within these systems, faults are very common and occur frequently.

There are two main types of faults within such systems and components; these include Short-Circuit Faults and Symmetrical Faults. The Short-Circuit Faults happen when an unintended low-resistance path gets established, leading to excessive current flow. On the other hand, the Symmetrical Faults include balanced three-phase faults, in which all three phases experience the same disruptions simultaneously, thereby causing equal fault currents to occur in each phase. Both faults significantly damage the electrical systems when not properly addressed.

These faults occur due to insulation failure, natural disasters, harmonic disturbance, and many other reasons. In traditional three-phase transmission systems, the distance between the phases (cables) is set in a specific way to make sure the air and space between them act as the insulation device, preventing any extra current from generating, causing current flow imbalances, which mayresult in the faults disclosed above.

Errors may also occur due to a failure within the components of the power transmission system, such as a transformer. This research will also go over distribution transformers and investigate the parameters that influence the proper functioning of the transformer.

As presented in Fig. 1, the main components of a transformer are its core and its insulatingoil, where its core is responsible for doing the main work and fulfillingthe actual purpose of the transformer, which is either to increase or decrease the voltage and change the current accordingly, as energycannot be destroyed.

These faults occur due to insulation failure, natural disasters, harmonic disturbance, and many other reasons. In traditional three-phase transmission systems, the distance between the phases (cables) is set in a specific way to make sure the air and space between them act as the insulation device, preventing any extra current from generating, causing current flow imbalances, which may result in the faults disclosed above.

Errors may also occur due to a failure within the components of the power transmission system, such as a transformer. This research will also go over distribution transformers and investigate the parameters that have influenced the proper functioning of the transformer.

As presented in Fig. 1, the main components of a transformer are its core and its insulatingoil, where its core is responsible for doing the main work and fulfillingthe actual purpose of the transformer, which is either to increase or decrease the voltage and change the current accordingly, as energycannot be destroyed.

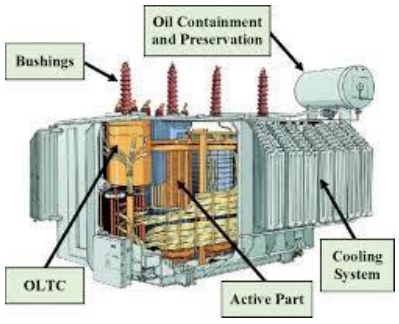


Fig. 1. Structure of the distribution transformer

Modern electrical systems utilize relays and traditional sensors to detect these faults, and despite their relatively complex nature, they have trouble when dealing with complex errors, such as harmonic distortion. Additionally, current fault detection technology tends to confuse between false alarms and actual faults within the system, resulting in a lower transmission efficiency as well as lower fault detection accuracy. Due to current power transmission systems having imperfect detection systems, this research will establish a new ground for error

detection and prediction within transmission lines and its components, and create a new tool for this very purpose. The model should include a localization system, helping the potential maintenance crew to determine the precise location whenever a fault occurs within such systems, saving valuable time and resources.

Research Objectives

The objectives of the study were: to;

1. Establish a new way of detecting faults within electrical systems, specifically in power transmission lines and distribution transformers.
2. Fill in the gaps of current research and work done within the field.
3. Classify the type of the faults according to their nature (gate codes from the dataset), and evaluate the model'saccuracy in classifying these faults.
4. Compare different types of machine learning models, including traditional ones that utilize established algorithms and a customdeep learning model.

Boundaries

Unfortunately, this research utilized a dataset that has been simulated for electrical systems that tend to operate in 50-55 Hertz, meaning the operating frequency of these power transmission systems does not match with those of the North American transmission lines. In North America, the operating frequency for the components and the transmission lines is 60 hertz.

This research included a comprehensive literature review that allowed the researcher to understand the current position of such technology and their developments within this specific application area. Additionally, the researcher identified the gaps in current research about this field, what kind of techniques have not been utilized, as well as what functionalities haven't been thought of yet.

METHODOLOGY

Information about the Dataset

In this study, the researcher utilized two different datasets, one for the distribution transformer fault prediction model and one for the general transmission line error classification. These datasets have various parameters that were present within such electrical systems and were found to be fundamental to the model during the training and optimization process to ensure its accuracy and overall performance in assessing new and unseen data. The dataset was obtained from a simulation of a classic transmission line, under various situations. These included when a fault is present, when the current flow is ideal e.t.c (Fahim, et al., 2021). Before working with the data and commencing with the coding portion of this methodology let the researcher identify the types of faults that may occur in power transmission lines as Short-Circuit Faults and Symmetrical Faults.

Short-Circuit Faults:

- LG Fault (Line to Ground)
- LL Fault (Line to Line)

Symmetrical Faults:

- LLL Fault (Between all phases, including A, B and C)
- LLG Fault (Between two phases and the ground)
- LLLG Fault (Three phase symmetrical fault)

From the dataset obtained from the simulation, the two different models were trained on these parameters:

VL1 - Voltage value in Phase A
VL2 - Voltage value in Phase B
VL3 - Voltage Value in Phase C
IL1 - Current Value in Phase A
IL2 - Current Value in Phase B
IL3 - Current Value in Phase C
OTI_A - Oil temperature Alarm (Distribution transformer)
OTI - Oil Temperature Indicator (Distribution transformer)
ATI - Ambient Temperature Indicator (Distribution transformer)
WTI - Winding Temperature Indicator (Distribution transformer)
MOG_A - Magnetic Oil Gauge Indicator (Distribution transformer)

Data Preprocessing

According to Ali Judge and Javaid, after obtaining the data from the simulation and having two final datasets, preprocessing these data is crucial, as it allows for a more stable and efficient training and optimization process, ensuring there are no false and inaccurate values within the dataset. During the simulation, occurrences of 0 and null values when switching the situations are a very common thing, which is why the first step was to clear these values with a simple conditional statement (Ali Judge & Javaid, 2017).

Following this, normalizing the data (bringing it to a specific interval for better training) was step number two. In Fig. 2, the researcher utilized min-max scaling to bring the training and testing datasets into a smaller range of values, reducing the dimensionality and complexity of the obtained dataset.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \quad \textbf{(1)}$$

Fig. 2 Min-Max Scaling

Principal Component Analysis

On the dataset made up of temperature, voltage, and current readings from distribution transformers and electrical systems, the researcher used the Principal Component Analysis (PCA). PCA is a dimensionality-reducing and machine-learning approach that simplifies a large data set into a smaller set without significantly changing the patterns and trends. To make sure every variable contributed equally to the analysis, the researcher first normalized the data by giving each feature a zero mean and a unit variance. The researcher then computed the covariance matrix to assess the correlation between each variable and all other variables. The researcher used Eigen decomposition to extract the eigenvalues and accompanying eigenvectors from this matrix.

The primary directions of variation in the data are represented by the eigenvectors, and the quantity of variance in those directions is shown by the eigenvalues. The eigenvectors with the biggest eigenvalues were then used to identify the principal components since they best captured the patterns and considerable variance in the data. Lastly, the researcher projected the original data onto these primary components, which successfully decreased the dataset's dimensionality while preserving the critical elements required to anticipate possible system faults.

Model Development

Dai, et al discussed the model's development. According to the author, the very first model that was constructed for detecting and classifyingthese faults within electrical systems and distribution transformers was perceptron-based neural networks (deep learning models), which are very efficient in capturing intricate relationships between labeled parameters within a particular dataset. The type of learning that was

employed in this case for the deep learning model was Supervised Learning (Dai, et al., 2021).

Supervised Learning

A model is trained using labeled data—that is, input data that has been associated with the appropriate output—in supervised learning, a sort of machine learning. To predict the output from the input data, the model must learn to do so. When the model is trained, it can be applied to additional, unobserved data to provide predictions.

Model Architecture:

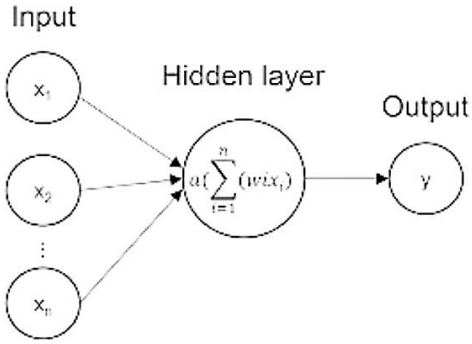


Fig. 3 Model Architecture

As presented in Fig. 3, the deep learning model is a neural network that has 3 different layers. These perceptrons carry a value that represents its activation value and are affected by weights and biases to minimize a defined cost function. To capture complex relationships, ensuring a non-linear relationship by utilizing activation functions for each of these layers is crucial. For the hidden layer, the researcher utilized a hyperbolic tangent activation function, which gets rid of the null gradient vector problem.

Learning and Optimization

For this custom deep learning model, the researcher utilized the stochastic gradient descent optimizer and the backpropagation learning algorithm. Stochastic gradient descent refers to an iterative approach for optimizing objective functions with appropriate smoothness properties. Stochastic gradient descent and backpropagation come together during this fundamental procedure to minimize the value of the cost function, and because this cannot be done directly, manipulating the values of the weights and biases so that the final output is closer to the actual value is essentially backpropagation.

Because the activation values of the neurons within each layer also affect the final output, the backpropagation algorithm also changes the values of the weights and biases that connect the first and second layers. To calculate these specific nudge values that the weights and biases have to be changed by, gradient descent (stochastic) is the key to achieving this as presented in Fig. 4. Gradient descent involves finding the partial derivative of the cost function concerning the weights and biases and using these, it creates a gradient vector that the weights and biases should be changed to improve the accuracy and F1-score of the model.

$$w := w - \eta \nabla Q_i(w) \quad \textbf{(2)}$$

Fig. 4. Stochastic Gradient Descent

Traditional Machine Learning Algorithms

According to Freire, et al, to make a point of comparison, the researcher also utilized popular and efficient machine learning algorithms to compare their accuracies, F1-Scores, and a few other performance metrics to decideand choose the final model that is most suitable to be used in real power transmission systems to detect faults within them. Because this model's main purpose and goal is binary classification, this filters out a few machine-learning models that are not so appropriate to employ (Freire, et al., 2019).

Logistic Regression:

A common statistical modeling method for binary classification tasks—where the objective is to classify data into one of two groups—is logistic regression. It forecasts the likelihood that an input falls into a specific category, which is expressed as a number between 0 and 1.

In Fig. 5, applying a logistic (sigmoid) function to a linear equation of the input features is how the model works. The linear equation's result is converted into a probability score via this function. The logistic function's S-shaped curve makes it a good fit for models in which probability is the output.

Through training, the coefficients of the linear equation are modified, usually with the use of a technique similar to maximum likelihood estimation as presented in Fig. 6. Finding coefficients that minimize the discrepancy between the model's predictions and the observed results is the goal of determining which model best fits the data.

Because of its ease of use and easily interpreted results, logistic regression is a popular statistical technique. With the use of methodslike multinomial logistic regression and one-vs-rest (OvR), it may also be expanded to handle situations involving several classes in classification.

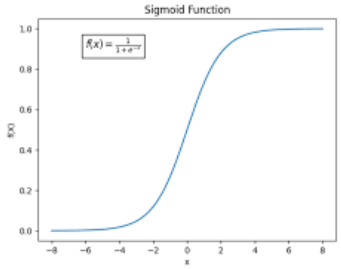


Fig. 5. (Sigmoid Function)

$$\mathcal{L}_n(Y_1, Y_2, \cdots, Y_n, \theta) = \prod_{i=1}^{\cdot\cdot} p_\theta(y_i)$$

$$\Rightarrow \mathcal{L}_n(Y_1, Y_2, \cdots, Y_n, p) = \prod_{i=1}^{n} p_p(y_i) = \prod_{i=1}^{n} p^{y_i}(1-p)^{(1-y_i)}$$

$$\Rightarrow \mathcal{L}_n(Y_1, Y_2, \cdots, Y_n, p) = p^{\Sigma y_i}(1-p)^{(n-\Sigma y_i)} \quad \textbf{(3)}$$

Fig. 6. Likelihood function

Decision Tree Classifier:

For classification problems, a machine-learning model called a decision tree classifier is also a viable choice that can be employed. To create a decision tree as presented in Fig. 7, it divides the data into subsets according to the values of the input features. Every node in the tree symbolizes a characteristic, and every branch denotes a decision rule that can result in more splits or other results. The final choices or classes are represented by the leaves of the tree. Decision trees are often used because of their simplicity and ease of interpretation and visualization.

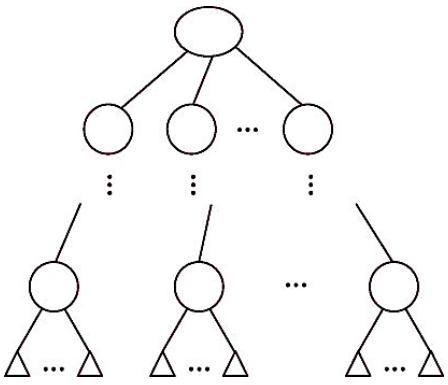


Fig. 7. Structure of a Decision Tree

Adaboost Classifier:

The machine learning model AdaBoost Classifier enhances classification accuracy by applying the boosting strategy. It sequentially combines several weak classifiers into one powerful classifier. Every classifier in the series corrects the weights of the cases based on concentrating on those that the preceding ones incorrectly classified. The performance of the model in challenging situations is enhanced by this procedure. AdaBoost works well because it adjusts to the ensemble errors, improving accuracy in a variety of situations.

Hyperparameter Tuning

Abakar, et al explain that hyperparameter tuning is modifying the training process's governing parameters, which are predetermined and remain constant during the training phase. Let's identify which parameters are key to the training and optimization process during gradient descent and backpropagation (Abakar, et al., 2020).

Learning Rate:

As described before, the backpropagation and gradient descent come together to optimize and teach the model according to the provided data. When calculating the cost function, and the values of the gradient vectors (the amount by which the weights and biases shouldchange) the learning rate is a crucial hyperparameter that should be adjusted when considering this, as it is the value that ensures the model does not experience overfitting, which means it would show very high and promising results in the training accuracy, but would dopoorly during cross-validation. This phenomenon occurs when instead of learning, the model "memorizes" the patterns that it has created between these data of labeled parameters, which would cause it to have trouble classifying and predicting whether there is a fault within the system or not during the testing procedure.

Batch Size:

Another hyperparameter that is worth paying attention to is the batch size since it affects the model's estimation of the error gradient during training. A bigger batch size offers more reliable but slower updates, whereas a smaller batch size can produce noise that helps avoid local minima. To strike a compromise between model performance and training speed the researcher experimented with various sizes.

Number of Epochs:

Epochs are defined to be complete iterations over the given dataset,whether it is the training, testing, or any custom-sized dataset for any particular purpose. Logically, the higher the number of epochs the better is a natural human thought that comes to mind when finding out about this specific hyperparameter. However, as mentioned before, overtraining during the optimization and learning process may not only occur because of the inefficient learning rate but also because of the total iterations it has over the provided dataset, which may lead to the model memorizing the data.

L2 Regularization

Another technique that the researcher employed for minimizing the cost function and preventing overfitting is L2 Regularization as presented in Fig. 8. This technique works by adding a variable to the cost function during the optimization process and nudging procedure, where the weights and biases are changed according to the gradient vector values. To prevent the absolute values of the weights and biases connecting the different layers from being too high, L2 Regularization is a very useful technique, preventing overfitting and reducing the cost function.

$$Loss_{original} + \lambda \Sigma_{j=1}^{n} (w_j)^2 \quad (4)$$

Fig. 8. L2 Regularization

Evaluation Metrics

Damaschke revealed that several statistical methods can be used in evaluating the results of the deep learning model as well as the models that employ traditional machine learning algorithms. In this case, this research utilized the Confusion matrices (2000).

Confusion Matrices:

A confusion matrix is a specific table layout used in statistics and machine learning to visualize the performance of an algorithm, typically a classification model. It helps to easily see when a model is confusing one class for another. The matrix is structured like a square, and it most commonly shows the frequencies of the predicted variables to the actual labels, allowing for a clear representation of the model's performance. It consists of four different entries:

- True Positives (TP), are the cases in which the model correctlypredicts the positive outcome.
- False Positives (FP), are the cases in which the model incorrectlypredicts the positive outcome.
- True Negatives (TN), are the cases in which the model correctlypredicts the negative outcome.
- False Negatives (FN), are the cases in which the model incorrectlypredicts the negative outcome.

In addition to being a square-shaped performance-displaying tool, confusion matrices also allow us to calculate some important evaluation metrics, which include but are not limited to:

- **Accuracy** - How often the model would be correct?
- **Precision** - When the model predicts a positive outcome,when is it correct?
- **Recall** - How often does the model correctly predict the positiveclass?
- **F1-Score** - The average precision and recall, give us a good idea of the model's overall performance (Fig. 10).
- **Specificity** - How often does the model correctly classifynegative classes?

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (5)$$

Fig. 9. The formula for the F1 Score

Receiver Operating Characteristic Curves:

ROC curves are graphical plots and a contemporary statistical technique used to illustrate the predictive capabilities of a machine-learning model when the decision threshold is changed. Simply, it involves combining several confusion matrices under different decision thresholds and evaluating the result according to the areaunder the graph, which gives us an unbiased point of view of the performance of this model, even if the dataset is imbalanced.

RESULTS

```
Accuracy for No Fault: 100.00%
Accuracy for LG Fault: 97.18%
Accuracy for LL Fault: 96.41%
Accuracy for LLG Fault: 86.22%
```

Fig. 10. Accuracy of the deep learning model in classifying thedifferent types of faults within power transmission systems

```
Logistic Regression
Training Accuracy:  99.82 %
Model Accuracy Score:  99.71 %
--------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      4077
         1.0       1.00      0.25      0.40        16

    accuracy                           1.00      4093
   macro avg       1.00      0.62      0.70      4093
weighted avg       1.00      1.00      1.00      4093

--------------------------------------------------------
```

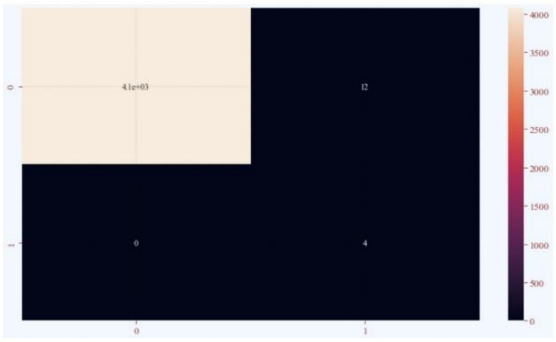


Fig. 11. Confusion Matrix for the Logistic Regression algorithm indetecting faults within the distribution transformer

```
Neural Network Training Accuracy: 99.64573662350354 %
Neural Network Model Accuracy: 99.60908868800391 %
```

Fig. 12. Accuracy of the custom deep learning model in classifyingfaults within distribution transformers

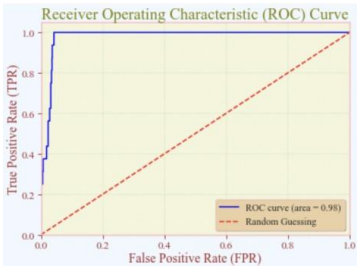


Fig. 13. ROC curve for the deep learning model

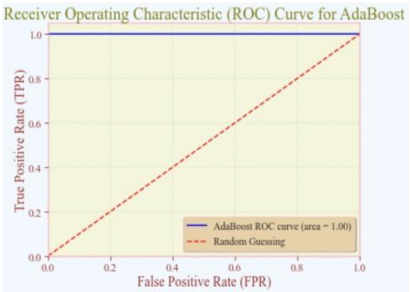


Fig. 14. ROC curve for the Adaboost Classifier Algorithm

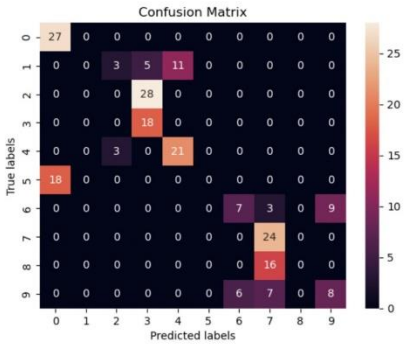


Fig. 15 The confusion matrix for the localization System

In Fig. 12, overall the accuracy of both the traditional machine learning algorithms and the deep learning model was very high, averaging 99%. In Fig. 13, when comparing the two, the deep learning model has higher accuracy than the logistic regression algorithm in both predicting errors within the power transmission systems and the distribution transformers. Additionally, in Fig.s 13 and 14, the neural network has an ROC curve with an AUC of 0.98, only less than the ROC curve of the Adaboost classifier, with an ideal AUC of 1.00. In Fig. 15, the localization system, unfortunately, has a relatively low accuracy as it can be observed from its confusion matrix.

This suggests that the simulated data obtained from very vague assumptions is not enough for its proper implementation and execution. The ideal model for predicting faults within distribution transformers and power transmission systems and effectively classifying them is either the Adaboostclassifer algorithm or the custom deep learning algorithm developed during this study. However, it is important to note that the deep learning model is much more flexible and has more hyperparameters available to tune.

DISCUSSION

Zerguit et al studied how integrating AI-enhanced fault detection industrial systems supported the findings of this study. The author revealed that advanced neural networks and hybrid models had a higher accuracy detection compared to traditional models. Traditional models include SVM (Gamma 0.01), DT (Depth 64), KNN (8 Neighbors), and ANN (256 Units). In terms of performance in the accuracy of detecting faults, the SVM model's configuration presented a moderate accuracy. The DT model presented a relatively lower accuracy. The KNN model showed moderate accuracy. Finally, the ANN model displayed high accuracy. Advanced Neural Networks included CNN, RNN, LSTM, and GRU. The CNN model showed high accuracy. The RNN presented superior accuracy. The LSTM model had high accuracy. Finally, the GRU had a high accuracy. Hybrid Models included CNN_RNN, CNN_LSTM, and CNN_GRU. The CNN_RNN model was very high in accuracy. The CNN_LSTM model achieved top marks in accuracy. Finally, the CNN_GRU had the highest accuracy. In conclusion, the advanced and hybrid AI-driven machine learning models showed superiority in the accuracy of detection of defects as opposed to traditional models/approaches; thus agreeing with this study's findings (Zerguit, et al., 2024).

CONCLUSION

By utilizing artificial intelligence, this study sought to address several challenging goals in the area of fault detection in distribution transformers and power transmission networks. The major objectives were to create a new fault detection technique, closeresearch gaps, categorize defects according to their attributes, assess several machine-learning models, and create a model with a localization system to improve maintenance effectiveness.

The results show that in terms of fault detection and classification, both proprietary deep learning models and conventional machine learning techniques achieved reasonably high accuracy. This accomplishment satisfies the first three study objectives by showingthat AI-driven solutions may effectively improve the fault detection processes in electrical power systems. In particular, the models' efficacy in comprehending and classifying intricate fault dynamics—as shown by gate codes present in the dataset—is demonstrated

by their capacity to identify faults based on their type, thus fulfilling the second goal of addressing research gaps.

The particular deep learning model outperformed the other models inthe test, which was explained by its adaptable design and fault localization system integration. In addition to achieving the goal of comparing various machine learning model types, this comparison demonstrates the potential of specially constructed neural networks in managing specialized power system management tasks.

Nevertheless, despite these successes, the study ran into a serious problem with the localization mechanism. Finding that the accuracy of fault localization was quite low is concerning because one of the goals of the research was to help maintenance teams locate faults quickly. This constraint implies that either the model needs to be further refined to increase its forecast accuracy in localization tasks, or the present dataset may not be robust enough.As a result, the study makes evident the need for more data to improve the model's performance in real-world circumstances, whether through extensive simulations or real-world testing.

In conclusion, even though this research has effectively demonstrated the use of AI in fault detection and classification, a number of its objectives remain unfulfilled, mainly related to the development of an efficient localization system, which calls for additional research and data enrichment. To overcome these constraints, future research must concentrate on obtaining larger amounts of data and improving the model, which should ultimately result in a more reliable and workable fault management solution for power transmission networks.

REFERENCES

Abakar, D., Abouelsoud, A. A., Saulo, M. J., & Sitati, S. (2020). Decentralized Constrained Optimal Control of the Multimachine Power System Stability Improvement. *Indonesian Journal of Electrical Engineering. 17*. 1158-1169. https://doi.org/10.11591/ijeecs.v17.i3.pp1158-1169.

Ali Judge, M., & Javaid, N. (2017). *Monitoring of Power Transmission Lines through Wireless Sensor Network in Smart Grid. Research Gate. 1*, 1-42. https://www.researchgate.net/publication/319143129_Monitoring_of_Power_Transmission_Lines_through_Wireless_Sensor_Network_in_Smart_Grid

Dai, X., Hao, J., Jian, Z., Liao, R., Zheng, X., & Zhang, Y. (2021). High-voltage frequency domain spectroscopy analysis of a thermally aged XLPE submarine cable under continuous and cyclic voltage based on carrier transport and polarization characteristics. *High Voltage*, *7* (3), 473–482. https://doi.org/10.1049/hve2.12139

Damaschke, P. (2000). Adaptive Versus Nonadaptive Attribute-EfficientLearning. *Machine Learning, 41* (2), 197–215. https://doi.org/ 10.1023/a:1007616604496

Fahim, S. R., Sarker, S., Muyeen, S. M. & Das, S. & Kamwa, I. (2021). A deep learning-based intelligent approach in the detection and classification of transmission line faults. *International Journal of Electrical Power & Energy Systems. 133*. 107102. 10.1016/j.ijepes.2021.107102.

Freire, Jean & Castro, Adriana & Homci, Marcia & Meiguins, Bianchi & Morais, Jefferson. (2019). Transmission Line Fault Classification Using Hidden Markov Models. *IEEE Access 1* (1), 1 - 1. https://doi.org/10.1109/ACCESS.2019.2934938

Zerguit, Y., Youness, H. & Derrhi, M. (2024). Integrating AI for enhanced fault detection in industrial systems: Evaluating machine and deep learning approaches in the industry 4.0 and IIOT era. *Journal of Theoretical and Applied Information Technology. 102.* 3795-3816.

APPENDIX

Appendix I: Detailed Dataset

	DeviceTimeStamp	VL1	VL2	VL3	IL1	IL2	IL3	VL12	VL23	VL31	INUT
0	2019-06-25T13:06	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	2019-06-27T10:49	238.7	238.7	238.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	2019-06-27T10:51	238.4	238.5	238.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	2019-06-27T10:52	239.9	240.0	240.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	2019-06-27T10:52	239.9	240.0	240.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...
19347	2020-04-13T23:30	246.4	244.7	244.3	79.0	45.4	86.6	426.0	422.3	426.1	38.0
19348	2020-04-13T23:45	246.8	245.1	244.8	77.0	42.0	84.2	426.8	423.1	427.1	39.1
19349	2020-04-14T00:00	247.8	246.0	246.0	77.4	37.1	75.4	428.5	425.0	428.7	38.8
19350	2020-04-14T00:15	247.7	245.9	245.5	70.1	36.7	88.1	428.2	424.3	428.4	45.2
19351	2020-04-14T00:30	248.4	246.6	246.3	59.2	33.1	86.8	429.4	425.4	430.0	46.5

Fig. 16. (Detailed Dataset)

Appendix II: Code for the Custom Deep Learning Model

```
class NeuralNetwork:
    def __init__(self, input_size, hidden_size, output_size):
        self.input_size = input_size
        self.hidden_size = hidden_size
        self.output_size = output_size

        self.weights_input_hidden = num.random.randn(self.input_size, self.hidden_size)
        self.bias_input_hidden = num.random.randn(1, self.hidden_size)
        self.weights_hidden_output = num.random.randn(self.hidden_size, self.output_size)
        self.bias_hidden_output = num.random.randn(1, self.output_size)
        self.mse_history = []

    def feedforward(self, X):
        self.hidden_input = num.dot(X, self.weights_input_hidden) + self.bias_input_hidden
        self.hidden_output = sigmoid(self.hidden_input)

        self.output_input = num.dot(self.hidden_output, self.weights_hidden_output) + self.bias_hidden_output
        self.output = sigmoid(self.output_input)

        return self.output

    def backward(self, X, y, output):
        error = y - output

        delta_output = error * sigmoid_derivative(output)
        error_hidden = delta_output.dot(self.weights_hidden_output.T)
        delta_hidden = error_hidden * sigmoid_derivative(self.hidden_output)

        self.weights_hidden_output += self.hidden_output.T.dot(delta_output)
        self.bias_hidden_output += num.sum(delta_output, axis=0, keepdims=True)
        self.weights_input_hidden += X.T.dot(delta_hidden)
        self.bias_input_hidden += num.sum(delta_hidden, axis=0, keepdims=True)

    def train(self, X, y, epochs):
        for epoch in range(epochs):
            output = self.feedforward(X)
            self.backward(X, y, output)

    def predict(self, X):
        predictions = self.feedforward(X)
        return predictions
```

Fig. 17. (Code for the Custom Deep Learning Model)