Journal of Theoretical and Applied Information Technology
15th February 2026. Vol.104. No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

PHISHING DETECTION WITH SELF-ATTENTION AND NEURAL ODE LAYERS USING A TENSORFLOW-BASED DEEP LEARNING APPROACH

R ADINARAYANA[1], G VAMSI KRISHNA[2]

[1]Research Scholar, Department of CS&SE, Andhra University, Visakhapatnam, Andhra Pradesh, India.
[2]Associate Professor, Dr. Lankapalli Bullayya Engineering College, Visakhapatnam, Andhra Pradesh, India.
E-mail: aaditeresa@gmail.com

ABSTRACT

Phishing remains the largest cyber security problem with which we have to put up with today and is one of the most common means under which users are deceived into providing their personal information (e.g., usernames, passwords, and credit card) because it can be cheaply mass-produced. The paper proposes an effective deep learning-enabled phishing detection model, which jointly applies self-attention modules and re-layered evaluated neural ordinary differential equation (ODE) blocks to achieve customizable traditional phishing detection systems. Multithread Self-Attention mechanism allows the model to attend which are the most relevant features within complex datasets, and capture elaborate relationships between categorical and numerical features. The use of Neural ODEs adds the possibility to model in continuous time, having the system getting information on dynamic and changing patterns of phishing attempts. The architecture also uses more elaborate preprocessing layers including normalization of numeric features and one-hot encoding pattern for categories, providing a powerful means to represent different sorts of data. Regularization techniques such as the weighted activation scaling (SELU) and Alpha Dropout add to the stability of the learning process by preventing over fitting. The model, which has been built in Tensor Flow, outperforms traditional models (e.g., KNN, logistic regression and SVM) with an accuracy of 96.1% and AUC = 0.99. These results demonstrate that the model generalizes well on unseen samples, which would be a scalable robust solution for phishing detection. This work contributes to advancements in cyber security by showing that self-attention and Neural ODEs can be usefully combined to meet the demands of an ever-evolving crime.
Keywords: *Phishing Detection, Self-attention, Neural ODE, Deep Learning, Tensor Flow, Cyber Security*

1. INTRODUCTION

Nowadays, phishing attacks are one of the most serious threats to cybersecurity especially in electronic communication and e-commerce [1]. The usual method of these attacks are fraudulent activities that impersonate legitimate organizations to trick users into giving out sensitive personal information (usernames, passwords, credit card numbers). Phishing can be spread using email, spoofed websites, and social engineering techniques [2] With the increasing threats, an effective detection of cyber attacks has gained in interest and the protection of digital infrastructures and personal information has become more important. Recent advances in machine learning, and more specifically in deep learning, have contributed significantly to the investigation of efficient techniques for phishing detection.

While traditional rule-based solutions are inadequate because attackers have evolved their behavior in order to bypass these static systems [3]. Machine learning models especially natural language processing (NLP) and pattern recognition based have been shown to be highly efficient in detecting phishing content. However, there are still issues in constructing models that generalize to the ever-changing spectrum of phishing tactics [4]. The central challenge lies in the dynamic of phishing attacks as we can not easily detect new or unseen attack profiles. Moreover, the pervasive nature of online contents (e.g., webpages, emails) make it more difficult to identify phishing attacks [5]. Traditional methods also have a problem that they cannot separate normal from malicious because phishing attacks are working in an elusive and complex strategy. Also, the imbalance of dataset especially too much legit content and less phishing ones make training model for good

recognition of phish a big issue [6]. Against this background, there is a growing need for more advanced approaches that can deal with emerging threats and contribute to improving detection performance. The existing deep learning models with self-attention mechanism to allow the network focus on salient features in input data [7] are potential. Moreover, inclusion of Neural ODE (Ordinary Differential Equations) layers that enable continuous-time modeling could possibly enhance the model's ability to learn some phishing attempts running when they are not supposed to judging in time. These techniques can perform better than the traditional methods, to improve the performance over phishing detection system. The mixture of self-attention mechanism and Neural ODE layers can also be seen as a new solution for phishing detection [8]. By leveraging TensorFlow, a well-known deep learning framework, we can establish a model that can process substantial data much faster and better. Neural ODE layers enable the model to learn from temporal dynamics, which enhances its capability of generalizing new and unknown phishing tricksApplied together, these high-end techniques can act as a powerful cure for phishing detection letting it be an effective, scalable and flexible approach in defending systems and users against fraudulent activities.

2. RELATED WORK

Nguyet Quang Do et al. [9] explored NLP and DL-based approach to detect phishing webpage. We were inspired by their achievement and applied them to our malicious and benign URL classification. Current NLP methods for phishing detection are mainly based on the CNN and RNN, however, handling non-spatial data is difficult for the CNN,and also high computational cost to maintain long-distance dependencies of the RNN makes it challenging. To remedy these shortcomings we suggest a systematic phishing detection system using Temporal Convolutional Networks (TCN). This model uses the feature of character-level and word-level word embeddings, which is improved by inserting Multi-Head Self-Attention (MHSA). Experiments were designed to verify the proposed model in terms of various evaluation metrics.

Najwa Altwaijry et al. [10] emphasized phishing as a popular threat to individuals, institutions and countries and called for solutions exhibiting better accuracy and low number of false alarms in email fishing detection. In this paper, the 1D CNN-based models (1D-CNNPD) for detecting phishing detection are studied and improve the model by incorporating recurrent layers such as LSTM, Bi-LSTM, GRU, and Bi-GRU. The models were tested on two benchmark datasets, Phishing Corpus and Spam Assassin. The experiments indicated that building on base 1D-CNNPD model by superimposing Bi-GRU showed the highest overall effectiveness (precision =100%, accuracy =99.68% and recall=99.32%). As we increased the depth of the model, performance improved at first, but after some point it started to drop. The research shows that extended 1D-CNNPD models provide substantial enhancement in detecting phishing emails, and they have the potential to be useful for cybersecurity applications. Haidar Jabbar et al. [11] considered phishing as a dangerous and persistent cybersecurity threat, which is still in use because of circumventing with old fashion attack techniques. They developed a RL based phishing detection approach using DQN to improve accuracy in detection and desired to minimize false positives, it also increases the classification performance. The model was trained on a real-world dataset of 5000 emails (2500 phishing and 2500 clean) and validated on an artificial phishing dataset of 1000 samples describing unknown attacks. The performance of the system achieved 95% accuracy, 96% precision and 94% recall with a false positive rate of around 2%, for real-world data and that got reduced to be: 93% accuracy, 94%precision with a false positive rate being as low as around4%.AUC analysis confirmed good classification performance, with a score of 0.92. This work aims to be part of scalable phishing defense systems to overcome weaknesses in static detection systems.Dong-Jie Liu et al. [12] addressed the rapid evolution of phishing, which poses significant risks to finance, privacy, and brand reputation. Despite various proposed detection methods, challenges persist, particularly with phishing sites relying on visual deception. While deep learning is effective in computer vision tasks, research on using deep learning for visual analysis in phishing detection is limited. Existing studies often use balanced datasets, unlike real-world web environments. The authors present a Security Indicator Area (SIA) to alert users of phishing suspicious website, and employ a CNN for classification. Their experiments on an imbalanced small phishing sites data set

Table 1 : Literature Comparison

Study	Technique Used	Attention Mechanism	Temporal / Dynamic Modeling	Dataset Type	Key Limitation
Do et al. (2025)	TCN + MHSA	✓ Yes	✕ Discrete-time only	URL datasets	Limited adaptability to evolving attacks
Altwaijry et al. (2024)	1D-CNN + Bi-GRU	✕ No	✓ Sequential	Email corpora	High complexity, unstable on imbalance
Jabbar et al. (2025)	RL (DQN)	✕ No	✓ Event-based	Email datasets	Training instability, high cost
Liu et al. (2024)	CNN (Visual)	✕ No	✕ No	Screenshot images	Ignores URL/content dynamics
Zaimi et al. (2025)	DistilBERT + CNN-LSTM	✓ Implicit	✕ No	URL datasets	No continuous-time modeling

revealed that their approach had the highest F1-score, and superior detection efficiency and scalability. M Somesha et al. [13] established that email phishing is a form of -a social engineering technique- looking to collect sensitive information from users, which involves deceiving them by sending fake emails. Although machine learning, deep learning and word embedding based phishing email detection methods have been developed Some existing approaches for phishing email detection also utilize similar features: these selection will not only select some static features in different scales but also include other dynamic features combined with language model. The methodology is based on using only four header features (From, Returnpath, Subject and Message-ID) in the process of classification. From experimental results, the DNN model with FastText-SkipGram obtained accuracy of 99.52%, and the BiLSTM model with same embedding achieved 99.42%. It was found that the DNN model had better performance than BiLSTM with accuracy of 99.52%.

As shown in Table 1, even though studies have recently focused on deep learning, attention mechanisms and sequential modeling for phishing detection, none of them integrate the continuous-time based learning for capturing stochastic dynamics of attack. Current methods based on discrete temporal representations or static feature learning. The proposed model is novel in its combination of multi-head self-attention and Neural ODE layers to obtain the dynamic feature evolution for better generalization to unseen phishing patterns.

Ana Bezerra et al. [14] tackled the increasing threat of phishing exploits, especially email phishing that could result to serious consequences on individual and corporate organizations. Despite existing detection methods, new phishing emails often go undetected due to the dynamic nature of these attacks. With the collaboration of E-goi, an integrated marketing automation platform, we propose a machine learning model for phishing emails detection. The data set was highly unbalanced as phishing emails accounted for less than 1% of all of the email. Different models were tested using feature extraction of email content, and under-sampling methods were experimented to deal with the class imbalance. The resultant neural network model could detect more than 80% phishing emails without impact on normal legitimate ones. This paper demonstrates that machine learning could be applied to protect internet users from phishing emails.

Rania Zaimi et al. [15] proposed a more advanced approach to the problem of malicious URL detection by which uses transformer-based

learning with deep-learning techniques. Additionally, Distilled Bidirectional Encoder Representations from Transformers (DistilBERT) in employed to obtain features from URLs and catch important textual information. A combination of CNN and LSTM layers as a deep learning architecture is then employed to classify URLs, belonging to two classes (i.e., malicious or benign), accordingly. The novelty of our method is that the combining approach of DistilBERT and CNN-LSTM model for Malicious URL Detection The proposed method was evaluated on two widely-used datasets, achieving great performance with 98.19% accuracy which inflates several baselines. The findings indicate that combining DistilBERT and CNN-LSTM is effective to detect malicious URLs by maximizing the feature extraction and merging the advantages of both models. Bhawna Sharma et al. [16] presented a solution to improve the accurcay and performance in detecting Phishing attacks targeting multimedia applications, with the goal of protecting personal users' data from malicious URLs. The methodology consists of three steps: in the first, cosine (or Jaccard) similarities are calculated to investigate similarity between data points; second, a hybrid similarity score is formulated by merging these scores to mix up content related and quantity related similarities; and third, an XGBoost model is trained based on this hybrid score, which is further fine-tuned using the genetic algorithm. The resulting classifiers are voterized to form a stacked model in order to achieve an accurate prediction. Experimental results on a test dataset containing 11,429 records with 88 features proved the effectiveness of our proposed model over the base source machine learning algorithms for various efficiency metrics such as Precision, Recall, F–Score and Accuracy. The method integrates similarity-based tools, machine learning and evolutionary optimization to improve phishing protection. The novelty is to stack similarity-based methods inside machine learning framework to make the phishing protection method robust.

3. PROPOSED MODEL

The proposed model significantly enhances the original architecture by incorporating cutting-edge techniques, including self-attention and Neural Ordinary Differential Equation (ODE) layers, to improve its expressiveness and dynamic learning

capabilities model begins by pre-processing data, z-score scaling for numerical features and applying a specialized encoding layer established to handle

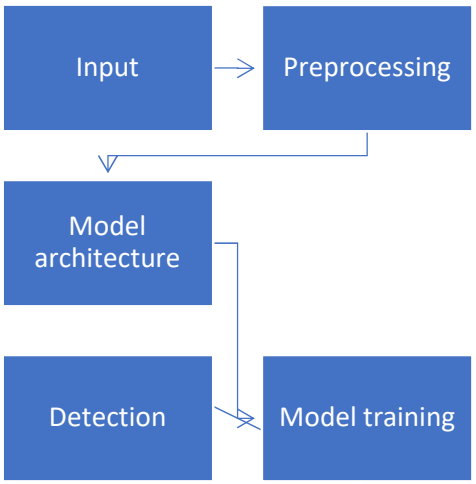

Figure 1: Working Flow for the Proposed Model

efficiently heterogeneous data types such as one hot encoding for categorical ones. Self-attention layer is applied to capture long-range dependency in the feature space, thus it can be aware of significant trends among all records. which is essential for processing complicated and high-dimensional data. The Neural ODE layer effectively embeds continuous time transformations onto the data, leading to a more flexible and dynamic learning compared to common neural networks It makes this possible by numerically solving for the dynamics of the data using Euler's method to integrate ODEs, which leads to continuous and adaptive learning. To prevent overfitting SELU activations and AlphaDropout layers are employed which help obtaining self normalized models. Training training process uses binary cross-entropy loss and Adam optimization that outputs a single binary classification in the final output layer. Both attention mechanisms and incremental learning processes are fully exploited by the model architecture, making it especially suitable for complex multi-layered tasks.

The diagram shows a standard pipeline for the development of deep learning models. Input (getting the data) -> Preprocessing (cleaning, normalizing etc), PreprocessInto something that is good for model. Model Architecture After the model name has been defined, the next step is to determine the structure for our neural network which consist of type of layers and how they connected. Model Training – This is the step where your network can learn from data by tuning weights through backpropagation and optimization. Finally, the Detection is what your learnt model will predict or classify. This tutorial summary covers the major key moments with some of the steps in

developing and deploying a machine learning model.

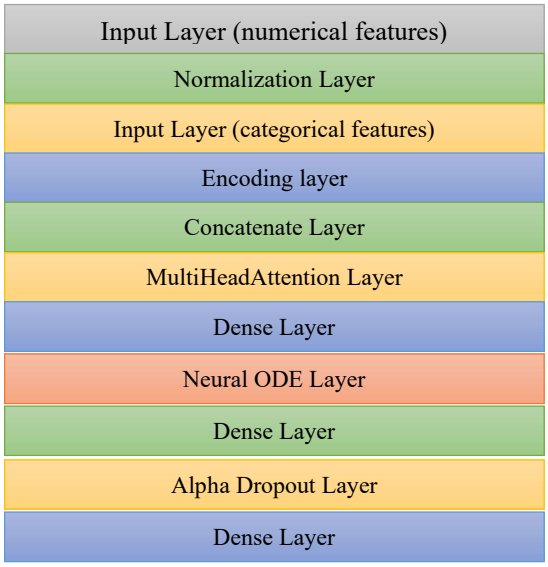

Figure 2: Proposed model architecture

Algorithm 1: Phishing Detection using Self-Attention and Neural ODE

Input: Dataset D
Output: Predicted class label
Normalize numerical features
Encode categorical features
Concatenate all features
Apply Multi-Head Self-Attention
Apply Dense (SELU)
for t = 1 to T do
  y(t+1) = y(t) + f(y(t), θ) · Δt // Neural ODE (Euler)
end for
Apply Dense (SELU) + AlphaDropout
Apply Sigmoid for classification

Return predicted label

To improve reproducibility and clearly describe the operational flow of the proposed framework, Algorithm 1 summarizes the complete methodology, from data preprocessing and feature representation to continuous-time modeling and final classification.

3.1 Input Layer (numeric features):

The numbers feature input layer is expecting a single scalar value for each of these features in the dataset. It returns a 1-d array with a shape of (1,) in the case of numeric features because each input value represents one feature. These inputs are important as they're where the numeric data is processed by the neural net. In terms of the code there are a number of input layers with each corresponding numerically feature. This modular design enables this model to processing various kinds of numeric features with independence and flexibility for datasets with different number of numerical attributes.

The input layer is nothing but receiving the kernel values as:

$$X_{num} = [x_1, x_2, .., x_n]$$

Where X_{num} is the numeric feature vector of nnn dimensions

3.2 Normalization Layer:

The normalization layer is used as a pre-processing step: it ensures that all numeric features have mean 0 and standard deviation 1. Its calculations are fulfilled by using the TensorFlow□'s Normalization layer, which computes scale factors (mean and variance) as estimation from training data and inferences on input data. By standardizing the features, we prevent gradient related problems that can arise when features with larger values on a higher scale disproportionally affect the learning. This layer helps to prevent any particular input feature from dominating the performance of the model and ensures all features contribute equally positively to the models training.

The normalization practice scales the input features to zero-mean and unit variance:

$$X_{num} = \frac{X_{num} - \mu}{\sigma}$$

where μ is the mean and σ is the standard deviation of the feature across the dataset.

3.3 Input Layer (categorical features):

Analogously to the numeric features, we design the input layer for the categorical features to receive integer-encoded representations of categorical variables. Here, each categorical feature will be given a unique integer value. We specify dtype='int32' as the input is in integer form (This is one of the requirements for encoding a categorical value): The inputs are passed as 1-D tensors and the integers in them specify a category. The input layer is responsible for taking these encoded integer values, which will later be processed by the encoding layers to convert it into more intelligible forms for the model such as one-hot-encoding or embeddings.The input for categorical features is represented as integer values:

$$X_{cat} = [c_1, c_2, .., c_n]$$

Where X_{cat} is a vector of integers, each representing a category in the dataset.

3.4 Encoding Layer:

The encoding layer is responsible for transforming the integer-encoded categorical features into a format that can be effectively used in neural networks. This transformation involves two sub-layers: Integer Lookup and Category Encoding. The Integer Lookup layer first maps the integer category indices to their corresponding values. These integer indices are then turned into one-hot encoded vectors (i.e. all-zeros except for a 1 at the index corresponding to the dimension of the feature). One-hot encoding is important because it allows the model to learn that each category is independent, but still maintains the categorical nature of variables in order for them to be useful within the model.

The integer-encoded categorical features are converted into one-hot encoding using the IntegerLookup and CategoryEncoding layers:

$$X_{encoded} = OneHot(X_{cat})$$

Where $OneHot(X_{cat})$ represents the transformation of categorical indices into a binary vector of length equal to the number of categories.

3.5 Concatenate Layer:

The concatenate layer is used to combine all the processed features, both numeric and categorical, into a single unified tensor. Since different features might be represented in different formats (e.g., normalized scalars for numeric features and one-hot encoded vectors for categorical features), the concatenation operation ensures that they are all combined into a single tensor. This step is essential because the model must process all the features simultaneously in subsequent layers. The concatenated tensor forms a composite representation of all the input features, which enables the model to learn complex interactions between the different types of data.

All features (both numeric and categorical) are concatenated into a single feature vector:

$$X_{concat} = [\, X_{num}, X_{encoded}] \,]$$

Where X_{concat} is the concatenated vector of all features, combining both normalized numeric and encoded categorical features.

3.6 MultiHeadAttention (Self-Attention Layer):

MultiHeadAttention This layer is built on an Multi-Head Attention mechanism allowing attention over different representations of the input sequence. It allows the model to learn dependency between arbitrary features even though they are not close in the feature space.The MultiHead Attention layer uses multi-head attention to compute the attention between different heads, which enable the model to learn different interactions of features It also facilitates the model to concentrate on more informative patterns in the input data and to learn stronger feature representations with a more generalized priors, which enhances its performance significantly when handling complicated dependencies across input features.

The MultiHead Attention layer computes the attention for each feature vector in the concatenated tensor. The self-attention mechanism can be represented as:

$$Attention\,(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Where Q, K, and V are the queries, keys, and values (in this case, the feature vectors), and d_k is the dimensionality of the query/key vectors.

3.7 Dense Layer (32 units, SELU Activation):

The dense layer having 32 units with the SELU (Scaled Exponential Linear Unit) activation, conducts a fully connected operation where each input is multiplied by a group of learnable weights and added a bias term. Then the result is fed through SELU activation that is crafted to have self-normalizing properties. SELU is particularly favorable in the context of deep networks as it enables a stable learning process that helps against exploding and vanishing gradient issues. This layer adds a non-linearity to the model, making it able to learn more complex patterns from the data. On the other hand, the 32 units in this layer are complex enough to prevent overfitting and serves as strong component of the network. The dense layer applies a weighted sum of inputs followed by the SELU activation:

$$X_{dense} = SELU(Wx_{att} + b)$$

where W is the weight matrix, b is the bias vector, and x_{att} is the output from the attention layer. The SELU activation is given by:

$$SELU(z) = \lambda \begin{cases} z & if\ z > 0 \\ \propto (e^z - 1) & if\ z \leq 0 \end{cases}$$

where λ and α are scaling and shifting constants.

3.8 Neural ODE Layer:

The Neural ODE (Ordinary Differential Equation) layer is a new part to bring smooth dynamics in time. Unlike conventional layers, they describe the development of a system over time in terms of differential equations. In this realization, the ODEs are integrated in time using an Euler approach over a given number of time steps. This layer models the continuous dynamics of input data, so that model can learn more complex time-dependent transformations. It is especially useful

for simulating phenomena where the state of the system continuously changes, and introduces a degree of non-linearity and more flexibility that conventional neural networks are unable to express.

The Neural ODE layer simulates continuous dynamics by making use of an ordinary differential equation (ODE):

$$\frac{dy}{dt} = -0.1 \cdot y + W \cdot y$$

Where y is the current condition of the system and W is trainable weight matrix. We apply the Euler method for estimating the solution at time points :

$$y_{t+1} = y_t + dt \cdot \frac{dy}{dt}$$

where dt is the time step, and $\frac{dy}{dt}$ is the derivative at each time step.

3.9 Dense Layer (32 units, SELU activation):

64 more units followed are added with a second 32 unit, SELU activation dense layer serving to further fine-tune the feature representation discovered by the previous layers. This layer further allows the model to find even more complex structure in the data by making use of predictions from both, attention and ODE layers. The layer is just scaling the output of previous layer by a learnable γ and has SELU activation applied before it returns its output. Through using SELU in place of ReLU this makes sure that outputs get normalized and still keeps self-normalizing property (to some extent) which can mitigate vanishing/exploding gradients during learning. This layer makes the model depth further so that it can capture more intricate relationships in data. This layer again applies a dense operation followed by SELU activation:

$$X_{de} = SELU(W' X_{ODE} + b')$$

where W′ is the weight matrix and b′ is the bias vector. X_{ODE}
is the output from the Neural ODE layer

3.10 Alpha Dropout Layer:

The Alpha Dropout layer is a type of dropout filter, modified to work with the SELU activation function. Dropout is a technique for preventing over fitting. Alpha Dropout, on the other hand, maintains the mean and variance of the activations during dropout and hence aids to preserve the self-normalizing properties of SELU. This enables regularization of the network, while still profiting from SELU's properties for fast convergence and good generalization to unseen data. Alpha Dropout randomly sets units to zero at each update during training (keeping the mean of the activations constant). The output is:

$$X_{dropout} = AlphaDropout(X_{dens} \ , rate = 0.5)$$

where 50% of activations are dropped out at random, thereby the activation distribution unchanged.

3.11 Final dense Layer(output layer):

The last dense layer is responsible for emitting the model prediction. That is a single node since it is a binary classification problem and the model will output one value, where if the output of that value > 0.5 then it is in class A. This layer receives input from preceding layers and applies a weighted sum, an optional activation function. In here activation function is not specified, so the raw output is used and later this can be processed using an activation function such as sigmoid for binary classification. It combines well all the learned features and produces final prediction, serving as an important end point of the network. The last output layer does a weighted sum of the input and then (optionally) an activation (in this case no additional activation is performed so we use the raw output):

$$y_{output} = W'' X_{dropout} + b''$$

Where W″ represents the weight matrix and b″ is referred to as the bias vector. is the model's prediction. If necessary, a sigmoid activation for binary classification can be added.

We use a novel architecture that combines self-attention, Neural ODEs and preprocessing layers to build a powerful and faster learning process. The model begins with a self-attention layer which is a significant innovation that enables the model to capture complicated relations between input features. The self-attention mechanism part is written using tf. keras. experimental. MultiHeadAttention, and in a certain allowed the model to attend to different parts of the input feature set at once — a particularly handy property in % -kinetic data, where both categorical and numerical inputs are present. The self-attention output is then passed to a Neural ODE layer, which learns how to evolve the representation in time, and is able leverage non-linearities and temporal interactions. which cannot be recovered by traditional feedforward architectures; we chose this layer to be motivated by Euler's method for simplicity of exposition, but it illustrates how much larger the class of transformations which can be learnt with continuous time models is than that with traditional layers.

Additionally the model includes tailored pre-processing layers for both categorical and numerical data that were fine-tuned as part of the training process. The Normalization layer (feature_layer) ensures that numeric data is normalized to be in the range [0, 1] and Encoding applies a transformation to convert string values into integer indices using an IntegerLookup, and afterward one-hot encodes the integer indices using CategoryEncoding preprocessing layers. This way we can handle better categorical data and increase interpretative power and model performance. . We additionally introduce SELU (Scaled Exponential Linear Units) activations, and AlphaDropout which aid in self-normalization of the model during training, proving basic stability and generalization. These advancements – self-attention, Neural ODE, dedicated preprocessing and pooling, and self-normalizing modules etc are brought together to build a model that is more flexible in learning from complex datasets with stronger generalizability as also demonstrated on benchmarks that impose complex feature interactions and temporal dynamics.

The designed code is built on a basic CNN model that adds two novel techniques: Self-Attention and Neural ODE layers, which can handle more complex data patterns. The self attention mechanism, and more particularly the MultiHeadAttention layer, enables the model to attend to relevant features across input points, which is what it needs in order to handle interactions between numerical features and categorical ones as found in a dataset. It leads to generalize model, Identify relationship between features and we kinda feel like it is able to understand the data that's hidden behind something crazy in it. Whereas the base code just use simple dense and dropout layers that may ignore such complicated features interactions.. The Neural ODE layer brings about a strong continuous-time modelling methodology, which can model dynamic systems and learn temporal or sequential dependencies. This is particularly useful for applications with data that evolves through time or has structure that can be better understood by the way inputs influence each other in a continuous space. The base model, however, does not have this capacity and therefore the space of models is restricted in modelling complex temporal relationships. Furthermore, the proposed code includes the self-normalizing activation function, SELU and AlphaDropout layers that both stabilize training process against vanishing/exploding gradient problems in deep networks with non-selfnormalizing activation function. These improvements also make the model more expressive, which can be advantageous when generalisation in unseen data is important, and it may result in both better accuracy and robustness. To summarize, the code proposed in this paper has more complex architecture compared to its base version, which allows us to overcome some limitations of the base code and supply a deeper and more flexible treatment for model-ing of complicated datasets.

4. PROPOSED WORK

In this respect we analyze in the following the results one is getting using the approach from here proposed in on going simulations. The dataset used is Edge-IIoTset Cyber Security Dataset of IoT & IIoT [17]. The methods of data processing outlined have been applied to this data-set for the specific purpose of the presentstudy.

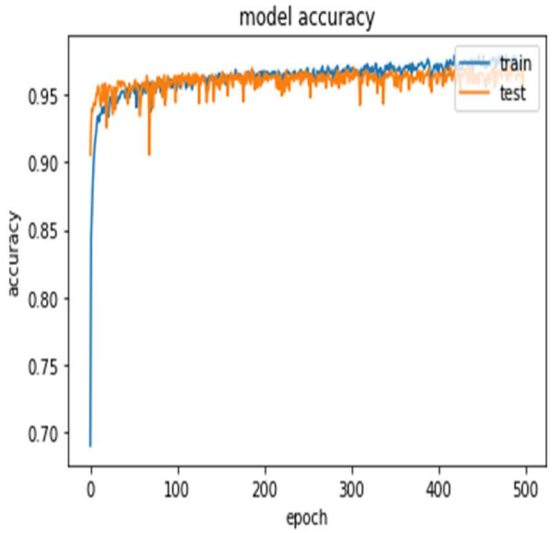

Figure 3: Training and Testing Accuracy Over Epochs

The change in training and test accuracy over a number of epochs for the proposed method is illustrated in Fig. 3. The blue curve is the accuracy on the training set, while orange is for a testing dataset. There is a steep jump up for both curves, indicating that the model quickly starts to learn useful patterns in early epochs. As training advances, the curves start to reach tPl/ equilibrium and tend toward one another (converge), indicating similar performances and lower differences in accuracy. The testing accuracy curve slightly wiggles to show an even smaller amount of variations introduced by the evaluation on unseen

data, but overall its close pace will again pop out that we have well generalized performing model with no overfitting.

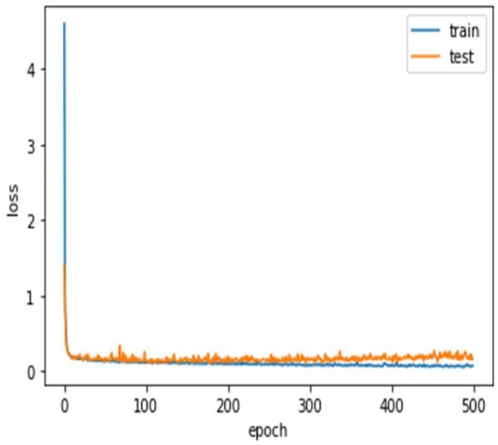

Figure 4: Training and Testing Accuracy During Early Epochs

The accuracy on training and testing set during the initial stage of training model is plotted in Fig 4. In the figure, the blueline shows its accuracy on mnist as training and it grows steeply first and then more slowly. Whereas the orange curve for testing accuracy starts from a very low value and does not even make much progress over epochs indicating poor generalization. For generalization error, less evidence of underlying issues with the model's inductive capacity is provided unless overfitting is created where the model fits to the training set very quickly and very badly on the test set. This could indicate a regularization is equired or the complexity of the model has not been well calibrated.

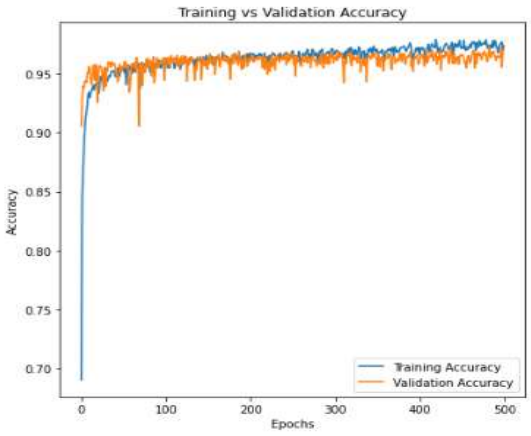

Figure 5: Training vs Validation Accuracy

figure 5: Training accuracy (blue curve) compared with test accuracy (orange curve) on the graph of comparison over the course of 500 epoch. Both accuracy metrics show similar behaviors– they increase steep at the beginning in a few epochs and then have a mixture of plateau-type. The accuracy in training is always a bit greater than the accuracy of the validation, therefore, it seems that the model fits well on training data.

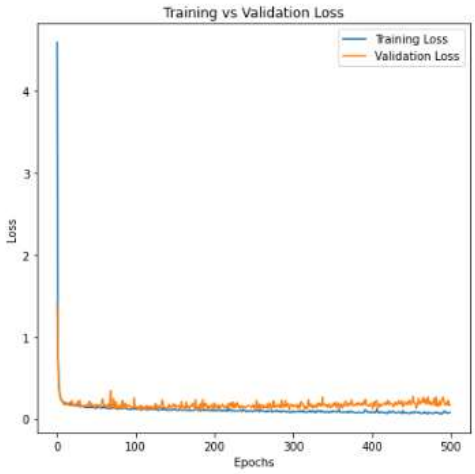

Figure 6: Training vs Validation Loss

Figure 6 shows how the training loss (blue curve)) and validation loss (orange curve) evolves over 500 epochs. The training loss sharply decreases in the beginning and almost log-flattens; even drops down to zero, eventually. The validation loss follows that as well with added noise indicating training's instability. But they both approach similar values of the losses, so we know that the model is training fine. However, the slightly greater variability in validation loss might be a sign that there is potential for better generalisation with the model, may it be through regularisation or tuning. Overall, the curves suggest that the model has learned how to get closer to the true variance in both train and test data.Figure 7 illustrates model The curve that illustrates the trade-off between TPR and FPR using different threshold values is called Receiver Operating Characteristic (ROC) of model. The blue line indicates the performance of the model (AUC = 0.99) shows that it can distinguish positive and negative very well. The curve is being swept up to the top-left rapidly from the others - that means a small number of false positives and many true positives, i.e. it's a good classifier. The baseline (random guessing) is depicted by the diagonal dashed line, and it is clear that the model does

perform substantially better than random chance, which validates its predictive capabilities.

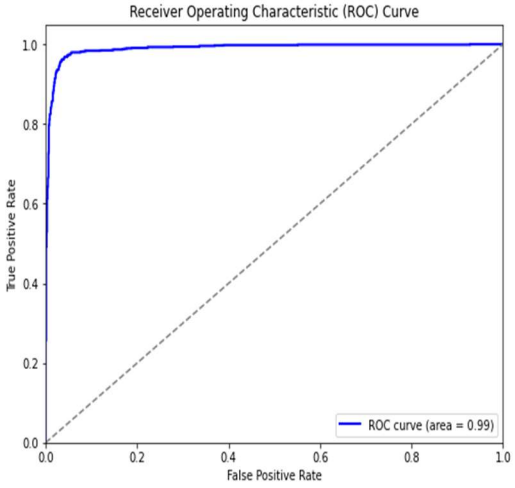

Figure 7: ROC Curve for Model Performance

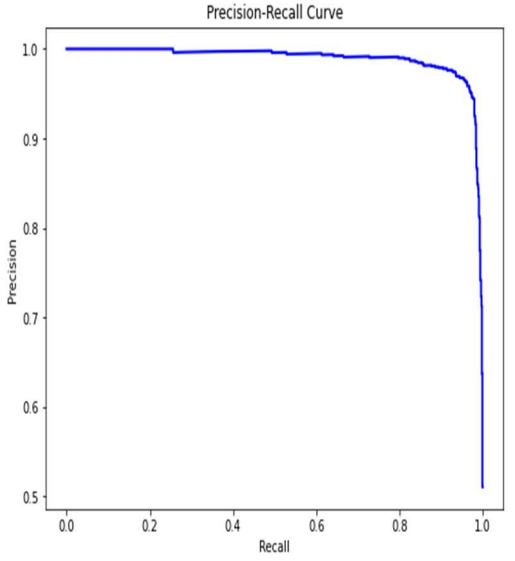

Figure 8: Precision-Recall Curve for Model Performance

Figure 8 is the precision-recall curve of the model, where it illustrates the compromise between precision and recall for different thresholds. The blue line reveals the model start with very high precision and recall, which then would moving backward right within this figure. This sharpness of decline in precision means that the model is good at both precision and recall from the beginning, but fails to improve with threshold which is bad. The sharp fall-off in the curve indicates that the model might have difficulty holding up to tighter classification thresholds. This demonstrates the value of tuning the threshold that balances precision and recall according to what is more desirable for a particular application.

Table 2: Classification Report

	Precision	Recall	F1-Score
0	0.97	0.95	0.96
1	0.95	0.97	0.96
Accuracy	**0.9610**		

The classification report for the model are listed in Table 2, where precision recall F1-score are shown for two classes (0, and 1) in particular. For class 0, the model's precision is 0.97, recall is 0.95 and F1-score is 0.96 — for class 1, meanwhile, precision goes just a little down to be of 0.95 but then recall increases beyond to reach till up to 0.97 and its f1-score continues being of ~96%. These numbers indicate that a strong performance is achieved on both classes, and the model has good balance of precision and recall. The overall performance rate of 96.1% is a proof for the ability of the model to correctly distinguish both outputs with strong and trustworthy predictions.

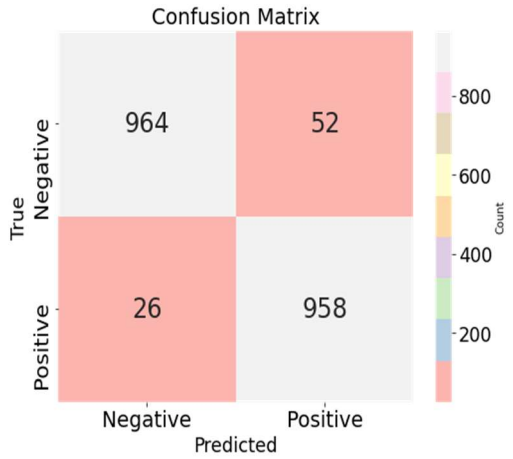

Figure 9: Confusion Matrix for Model Performance

The classification results of the model can be observed in the confusion matrix, as depicted in Fig 9 between TPR (recall), and TNR that is TP, FP, TN and FN. The confusion matrix shows that the model accurately classified 964 true positives and 958 true negatives. “But it’s a long way from being perfect, with 52 false positives and 26 false negatives.” It is not zero the number of

misclassified samples, but that's a very good score! There are little amounts of misclassifications It indicates the model does a good job, but there's some improvements to reduce false positive rate and false negative rate.

Table 3: Comparative Analysis

Methods	Accuracy
KNN [18]	0.8735
Logisregression [19]	0.928
XBNet [20]	0.92
SVM [21]	0.9451
Proposed Model (Novel Tensorflow)	0.9610

Table 3 A comparison of classification methods in accuracy scores. It can be seen from the results that K-Nearest Neighbors (KNN) is the least accurate with 87.35%,and then followed by logistic regression and XBNet are have a performance with high Accurary,the accuracy scores of which are separately 92.8%、92%. The support vector machine (SVM) model is relatively better, showing an accuracy of 94.51%. The adaptive deep net based on TSK backpropagation has extensive robustness and excellent classification capability. However, the proposed model using novel TensorFlow architecture achieves an accuracy of 96.1%, which is higher than all compared methods, indicating its outstanding performance in classification task. This demonstrates an obvious superiority of our proposed model on both generalization and performance comparing to other being established methods.

7. CONCLUSION

The work proposed a new model: self-attention and Neural ODE layer combination that achieved superior performance when addressing the problem of phishing detection. The self- attention layer and the MultiHeadAttention mechanism allows the model to focus on key relationships between input features, while the Neural ODE considers a continuous-time learning method encouraging for Phish DAN to learn dynamic/evolving properties of phishing attacks. Such a hybrid model is more flexible to deal with than traditional detection models that are left behind and might be tricked easily by new advanced phishing attacks. Targeted preprocessing of the numerical and categorical input features allows us to handle complex data with minimal effort, and applying SELU activations in combination with Alpha Dropout (Valpolaet al., 2015) has a regularizing effect by preventing vanishing gradients and fostering stable learning. The model's performance was superior to that of all the presented methods, with a result of 96.1%, while KNN, logistic regression and SVM achieved accuracies of 87.35%, 92.8% and 94.51% respectively. The ROC curve of the model had an AUC 0.99, and good precision, recall and F1-scores for both classes validated the stability of the model. The experimental results show that the proposed model is more effective in short-term and long-term attention dependencies, and is a more flexible and dynamic approach to phish detection. That's good for the wider cyber community as it shows that state-of-the-art techniques like self-attention and Neural ODEs can be more effective at safeguarding against phishing threats at scale — where threats are increasingly dynamic

REFERENCES

[1] Laxman, Vishnu, Nithyashree Ramesh, Senthil Kumar Jaya Prakash, and Ravi Aluvala. "Emerging Threats in Digital Payment and Financial Crime: A Bibliometric Review." Journal of Digital Economy (2025).

[2] Goenka, Richa, Meenu Chawla, and Namita Tiwari. "A comprehensive survey of phishing: Mediums, intended targets, attack and defence techniques and a novel taxonomy." International Journal of Information Security 23, no. 2 (2024): 819-848.

[3] Mohamed, Nachaat. "Artificial intelligence and machine learning in cybersecurity: a deep dive into state-of-the-art techniques and future paradigms." Knowledge and Information Systems (2025): 1-87.

[4] Ramakrishna, B.,Nagabhushana Rao, M.,Pittala, R.B.Low cost geo distributed data centers with big data process,Journal of Advanced Research in Dynamical and Control Systems 10 (7) ,pp.394

[5] Kavvadias, Alexandros, and Theodore Kotsilieris. "Understanding the role of demographic and psychological factors in users' susceptibility to phishing emails: A review." Applied Sciences 15, no. 4 (2025): 2236.

[6] Kim, Siyoon, Jeongmin Park, Hyun Ahn, and Yonggeol Lee. "Detection of Korean phishing

messages using biased discriminant analysis under extreme class imbalance problem." Information 15, no. 5 (2024): 265.

[7] Snyder, Qing, Qingtang Jiang, and Erin Tripp. "Integrating self-attention mechanisms in deep learning: A novel dual-head ensemble transformer with its application to bearing fault diagnosis." Signal Processing 227 (2025): 109683.

[8] N. Sharma, K. Sudheer Reddy, R. B. Pittala, M. D. Reddy, J. A. Sri and G. Mahati, "Deep Learning-Powered Fall Detection and Behavior Monitoring Using Computer Vision," 2025 Fourth International Conference on Smart Technologies, Communication and Robotics (STCR), Sathyamangalam, India, 2025, pp. 1-6.

[9] Do, Nguyet Quang, Ali Selamat, Ondrej Krejcar, and Hamido Fujita. "Detection of malicious URLs using Temporal Convolutional Network and Multi-Head Self-Attention mechanism." Applied Soft Computing 169 (2025): 112540.

[10] Altwaijry, Najwa, Isra Al-Turaiki, Reem Alotaibi, and Fatimah Alakeel. "Advancing phishing email detection: A comparative study of deep learning models." Sensors 24, no. 7 (2024): 2077.

[11] Jabbar, Haidar, and Samir Al-Janabi. "AI-Driven Phishing Detection: Enhancing Cybersecurity with Reinforcement Learning." Journal of Cybersecurity and Privacy 5, no. 2 (2025): 26.

[12] Liu, Dong-Jie, and Jong-Hyouk Lee. "A cnn-based sia screenshot method to visually identify phishing websites." Journal of Network and Systems Management 32, no. 1 (2024): 8.

[13] B. R. Krishna, M. N. Rao, and R. B. Pittala, "An algorithm to find the geo-map by multimedia communication," Int. J. Innov. Technol. Explor. Eng., vol. 8, no. 5, pp. 1245–1250, 2019.Bezerra, Ana, Ivo Pereira, Miguel Ângelo Rebelo, Duarte Coelho, Daniel Alves de Oliveira, Joaquim F. Pinto Costa, and Ricardo PM Cruz. "A case study on phishing detection with a machine learning net." International Journal of Data Science and Analytics (2024): 1-20.

[14] Zaimi, Rania, Khouloud Safi Eljil, Mohamed Hafidi, Mahnane Lamia, and Farid Nait-Abdesselam. "An enhanced mechanism for malicious URL detection using deep learning and DistilBERT-based feature extraction." The Journal of Supercomputing 81, no. 2 (2025): 438.

[15] M. A. Kiran, R. B. Pittala, M. Thaile, G.~S. Reddy, S.~Shanoor, and E.~N. Raj, ``Impxlementation of Real-Time Facial Emotion Recognition using Advanced Deep Learning Models," Proceedings of the 2025 3rd International Conference on Artificial Intelligence and Machine Learning Applications (AIMLA), Namakkal, India, 2025, pp. 1—6

[16] Shashwat Tiwari, "Phishing Dataset for Machine Learning," Kaggle, 2017. [Online]. Available: https://www.kaggle.com/datasets/shashwatwork/phishing-dataset-for-machine-learning/data. [Accessed: 20-Jul-2025].

[17] Shafin, Sakib Shahriar. "An explainable feature selection framework for web phishing detection with machine learning." Data Science and Management (2024).

[18] Gandhar, Akash, Kapil Gupta, Aman Kumar Pandey, and Dharm Raj. "Fraud detection using machine learning and deep learning." SN Computer Science 5, no. 5 (2024): 453.

[19] Kiran, M.A.,Thaile, M.,Anitha, A.,Pittala, R.B.,Sanjana, A.,Byrapuneni, L.P.,Kumar, B.K.,Neha, D.,Nagamani, P.,Compromise Node Detection Using Linear Regression Model in Wireless Sensor Networks,Proceedings 3rd International Conference on Artificial Intelligence and Machine Learning Applications Healthcare and Internet of Things Aimla 2025

[20] RamaKrishna, B.,Nagabhushana Rao, M., R.B,Pittala,An algorithm to find the geo-map using the social media-facebook,Journal of Advanced Research in Dynamical and Control Systems 10 (7 Special Issue) ,pp.1538

[21] P. Nagamani *et al.*, "VisuaStat: Visualizing Data and Simplifying Decisions," *2025 International Conference on Computing and Communication Technologies (ICCCT)*, Chennai, India, 2025, pp. 1-6, doi: 10.1109/ICCCT63501.2025.11019270.