Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Network Traffic Prediction: Optuna-Driven Model configuration Parameter Tuning for Quality Management

[1]Poonam Lohiya, [2]Gajendra Bamnote
[1]pnmlohiya127@gmail.com
Research Scholar, PRMIT&R, Badnera
[2]grbamnote@rediffmail.com
Principal, PRMIT&R, Badnera

Abstract: Protocol prediction in network traffic allows for smarter, more effective network management by anticipating the types of data that will flow through the network and regulating resources consequently.The prominence is on using Optuna, a framework for fine-tuning model configuration parameters that maximize model performance. A deep learning-based model that was tuned for tasks involving the classification of network traffic is created using TensorFlow/Keras. In order to increase classification accuracy, a variety of model setup parameters by employing Optuna's effective search methods and pruning processes is implemented. This AI-based network traffic classification model attained high performance, with 90.48% accuracy, 90.70% precision, 90.47% recall, and a 90.52% F1-score. The final model showed that optimizing model configuration parameters improve network traffic prediction and categorization, as seen by the distinguished gains in accuracy and other evaluation metrics that it attained.

1. Introduction:

Network traffic has increased significantly due to the exponential expansion in internet usage.That is the reason, effective traffic classification and prediction are crucial for maintaining network security and quality of service (QoS). Network managers prioritize, secure, and improve network performance by using the classification of network traffic, which is essential for recognizing and controlling various types of traffic. However, standard classification techniques have difficulties due to the dynamic nature of network traffic, which is considered by a variety of applications and protocols.

Today's developments in artificial intelligence [1][2] have brought increasingly complex models, such as machine learning (ML) and deep learning (DL), to solve these issues. These models have demonstrated potential in managing difficult traffic patterns and attaining increased precision in classification. The selection of model configuration parameter commands the behavior of the learning algorithm and it has a significant influence on these models' performance. It is often time-consuming and ineffective to manually adjust the model configuration settings, particularly for deep learning models with lots of parameters.

Within this framework, optimizing the model setup parameter [3] has become avital first step toward improving model performance. In order to optimize model performance through model configuration parameter adjustment, this research presents a novel way to AI-based network traffic classification and prediction. To enhance a deep learning model's classification accuracy, the tuning process using Optuna [17] [18], a state-of-the-art framework for optimizing model configuration parameters, is automated.

Optuna[19] [20] has a number of benefits over conventional optimization techniques, such as itdefines by methodology that enables flexible and dynamic optimization and its strong pruning algorithms that prematurely end unproductive trials and conserve computing capacity. The efficiency of the model for classifying network traffic is tried to improve as its use of adjusting parameters and show how fine-tuning model configuration parameters increase prediction accuracy.

The paper is divided in following sections: The past work in network traffic classification and model configuration parameter optimization is reviewed in Section 2. The approach, which includes the Optuna-based optimization process, model design, and data pretreatment, is explained in Section 3. The optimized model's results are shown in Section 4, and the paper's conclusion and future research prospects are outlined in Section 5.

By offering a thorough framework for classifying network traffic using neural network techniques and inevitably fine-tuning model configuration parameters within optuna framework, this research advances the area. The results is used to enhance network traffic management systems' accuracy and efficiency, which will ultimately improve network security and QoS.

2. Related Work:

With the goal of identifying the kind or purpose of Internet traffic, network traffic classification is an essential part of network administration and security. This review examines the range of approaches used in the classification of network traffic, emphasizing the progression from conventional methods to sophisticated machine learning and deep learning techniques.

2.1. Traditional Traffic Classification Methods

Conventional approaches for classifying network traffic are divided into payload based, port based, and statisticalbased methods [1]. These techniques are commonly used, still the growing usage of encryption and dynamic port allocation presents thoughtful difficulties.

1. Port-based Classification: To identify apps, this method uses well-known port numbers. But as more programs service or use dynamic or non-standard ports, its efficacy has decreased [2].
2. Payload-based Classification: This method distinguishes apps by examining the payload of packets. Method is accurate but it is inefficient for encrypted traffic and computationally challenging [2].
3. Statistical-based Classification: This technique classifies traffic based on statistical characteristics of traffic flows, such as packet size and inter-arrival intervals. Encryption has less of an impact on it, but accuracy perhaps be an issue [2].

2.2. Machine Learning Approaches

More advanced techniques for classifying protocol for network traffic quality management have been used with the introduction of ML techniques. It makes use of data to learn and get better over time.

1. Supervised Learning: Using labeled training data, algorithms have been used to classify traffic. These techniques do not work well with zero-day applications and necessitate big labeled datasets [3].
2. Unsupervised Learning: Similar traffic patterns are grouped using clustering techniques like k-means without any prior label. These techniques recognize novel forms of traffic, but challenging due to enormous use of internet in today's era [3].
3. Hybrid Approaches: Identifying zero-day applications is one area where combining supervised and unsupervised learningimprove robustness. To increase classification accuracy, the robust statistical traffic classification (RTC) scheme, for example, syndicates the two methods. [3][4][5].

2.3. Deep Learning Techniques

Deep Convolutional neural networks (CNN) and network-in-network (NIN) models are examples of deep learning models. They establishedpotential in managing encrypted and complex traffic patterns.

1. CNN-based Models: These models achieve good classification accuracy by extracting features from traffic data using convolutional layers. They however computationally costly [6].
2. NIN Models: By adding micro-networks after every convolution layer, NIN models improve local feature modeling. In order to reduce model complexity and achieve a balance between efficiency and accuracy, they additionally employ global average pooling [6] [7].
3. Hybrid Neural Networks: By applying dual-mode feature extraction, hybrid neural networks enhance classification performance by combining flow-level and packet-level characteristics [10].

Despite progress, there are still a number of issues in classifying network traffic:

1. Traditional and machine learning-based techniques face serious hurdles from the growing usage of encryption. To solve this problem, methods like hybrid neural networks and deep NIN models are being investigated [6] [10].
2. Biased models may result from imbalanced datasets. To lessen this issue, approaches including cost-sensitive learning and ensemble methods have been suggested [8].
3. The performance of DL-based classifiers is severely harmed by adversarial attacks. In order to create resilient models that able to survive such attacks, research is still being done [9].

Sophisticated ML and DL techniques have replaced more conventional port and payload-based approaches in the classification of network traffic. Even though there has been a lot of progress, issues still motivate study in this area.

3. Methodology:

This section explores the specific methods used to create the network traffic categorization for protocol prediction model with an emphasis on the mathematical underpinnings of the model, data preprocessing, model design, and model configuration parameter optimization using Optuna [20]. 87 features that were taken from IP flows will be used to classify network traffic. A deep learning framework optimized by supervised optuna [21] based model configuration parameters will be used to do this. Proposed methodology is as follows figure 3.1.

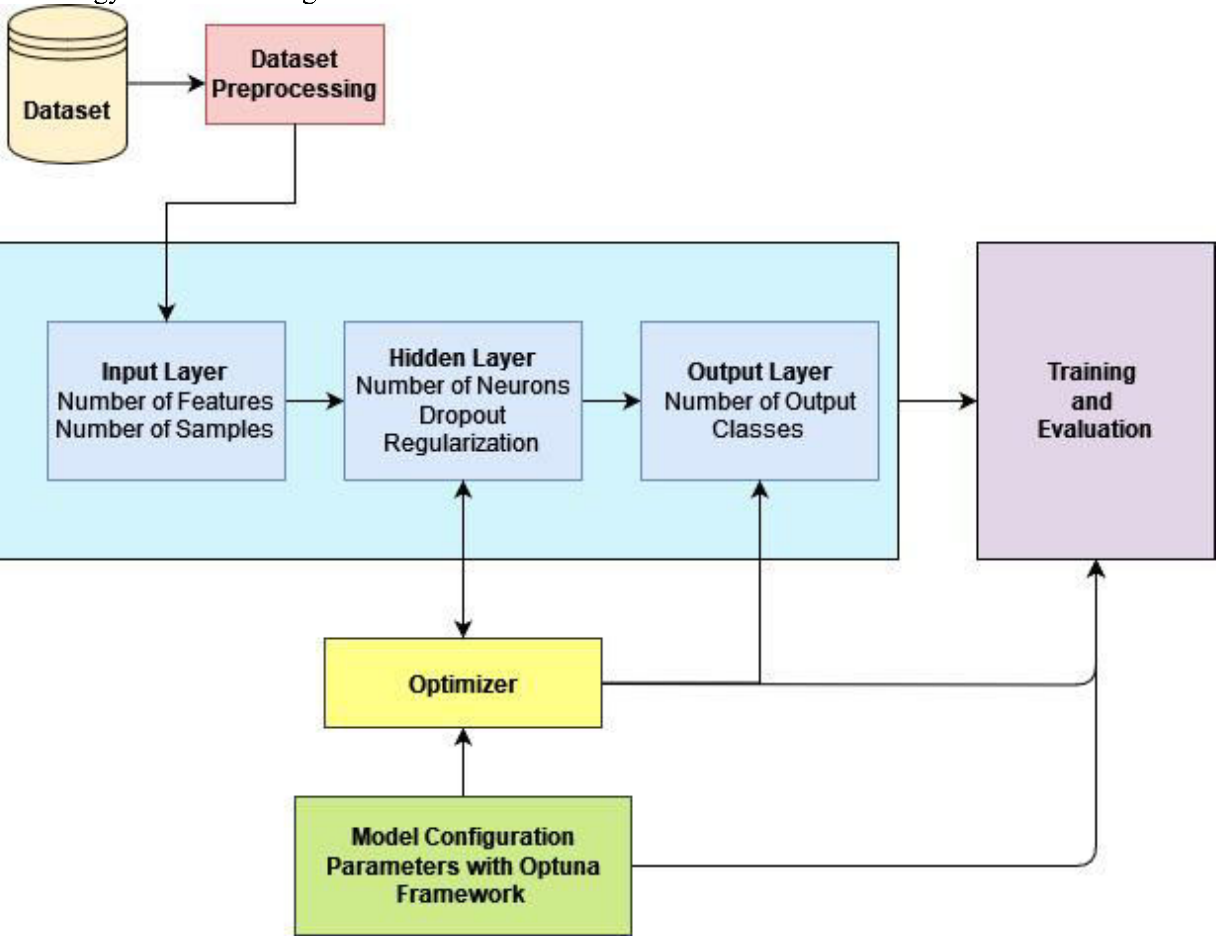

Figure 1: Proposed Architecture for Network Traffic Classification

3.1. Dataset and Preprocessing

The dataset [11] [12] [13] consists of 87 features that describe the characteristics of IP flows, which include:

- Source and Destination IP Addresses (Nominal)

- Source and Destination Ports (Numeric)
- Inter-arrival Times (Numeric)
- Timestamp (DateTime)
- Layer 7 Protocol (Application Class)

Given the heterogeneous nature of the features, preprocessing steps are crucial to standardize the data and make it suitable for the neural network.

Preprocessing Techniques:

- Data Cleaning: Let X be the feature matrix, where $X \in R^{N\times 87}$ and N represents the number of instances. The dataset undergoes cleaning to remove any null or corrupted entries. Define a binary mask matrix $M \in \{0,1\}^{N\times 87}$ where M_{ij}=1 if $X_{ij}$is valid, otherwise$M_{ij} = 0$. The cleaning operation is formalized as:
$$X_{clean} = X \circ M$$
where ∘ denotes the Hadamard product.
- Missing Value Imputation: Missing values in the feature matrix X_{clean} are imputed using a strategy such as mean imputation. If μ_j denotes the mean of the jth feature across all valid entries, the imputation operation for missing entries is given by:
$$X_{ij} = \mu_j if M_{ij} = 0$$
- One-Hot Encoding: For nominal categorical features, one-hot encoding transforms each category into a binary vector. Let X_{cat} be a nominal feature vector with Cunique categories. The one-hot encoding of X_{cat} results in a binary matrix $X_{onehot}\in\{0,1\}^{N\times C}$ where:
$$X_{onehot}[i,k] = 1 if instance i belongs to category k$$
- Normalization: To ensure all features contribute equally to the model, numerical features are normalized. For feature jj, normalization is defined as:

$$X`_{ij} = \frac{X_{ij} - \min(X_i)}{\max(X_j) - \min(X_j)}$$

resulting in a transformed feature matrix $X` \in [0,1]^{N\times 87}$
- Train-Test Split: The dataset is split into training $X_{train}, y_{train}$and test sets X_{test}, y_{test}. The split ratio is set as 70% for training and 30% for testing.

3.2. Model Architecture

In order to effectively capture the associations between the 87 input features and the target class (application protocol), the model is a deep feedforward neural network. TensorFlow/Keras is used to build the architecture, and Optuna is used to optimize the model's setup parameters.

Let the input feature matrix be $X' \in R^{N\times 87}$, and the corresponding output class vector be $Y \in R^{N\times C}$where C is thenumber of unique application classes.

The neural network comprises an input layer, L hidden layers, and an output layer. The transformations at each layer can be expressed as follows:

- Input Layer: The input layer applies a linear transformation followed by a activation function. If $W_0 \in R^{87*d1}$ represents the weight matrix and $b_0 \in R^{d1}$is the bias vector, the activation of the input layer is given by:

$$h_1 = ReLU(X'W_0 + b_0)$$

where $h_1 \in R^{N*d1}$
- Hidden Layers: Each hidden layer applies a linear transformation followed by a ReLU activation function and dropout regularization. For the $l - th$ hidden layer, the transformation is:
$$h_{l+1} = DROPOUT(ReLU(h_l W_l + b_l)$$

Where$W_l \in R^{dl+dl+1}$and $b_l \in R^{dl}$ represent the weight matrix and bias vector, respectively. The dropout function randomly sets a fraction of the activations to zero, controlled by a dropout rate pl, to prevent overfitting:

$$Dropout(h_{l+1}) = h_{l+1} \cdot D, \qquad D \sim Bernoulli(pl)$$

where D is a binary mask matrix.

- Output Layer: The output layer applies a softmax activation function to produce class probabilities:

  $$y' = Softmax(h_l W_l + b_l))$$

  where $y' \in R^{N\times C}$ represents the predicted probabilities for each class.
  The softmax function is defined as:

  $$Softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}$$

  ensuring that the output is a valid probability distribution.

Loss Function: The model is trained using the categorical cross-entropy loss function, which measures the divergence between the predicted probabilities y'and the true labels y:

$$L(y, y') = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(y'_{ij})$$

where$y_{ij}$is a binary indicator (0 or 1) if class label j is the correct classification for instance i.

Regularization:L2 regularization is applied to the weights to prevent overfitting by penalizing large weight values. The L2 regularization term is added to the loss function:

$$L_{total} = L + \lambda \sum_{l=0}^{L} \| W_l \|$$

where λ is the regularization strength.

3.3Model configuration parameter Optimization with Optuna

In order to effectively capture the associations between the 87 input features and the target class (application protocol), the model is a deep feedforward neural network. TensorFlow/Keras is used to build the architecture, and Optuna is used to optimize the model's setup parameters.

Optimization Process:

- Objective Function: The objective function, $f(\theta)$ is defined to train the model with a set of model configuration parameters θ and return the validation accuracy:

  $$f(\theta) = Accuracy_{val}(\theta)$$

  where θ includes parameters like the number of layers, neurons, learning rate, etc.
- Sampler: Optuna employs the Tree-structured Parzen Estimator (TPE) sampler, which models the objective function probabilistically. The TPE uses Bayesian optimization to efficiently explore the model configuration parameter space by balancing exploration and exploitation:

  $$\theta_{t+1} = arg\ max\, E[\frac{p(Accuracyval \mid \theta)}{q(\theta)}]$$

  where $p(Accuracyval \mid \theta)$is the likelihood of obtaining a high accuracy given the model configuration parameters θ, and $q(\theta)$ is a prior distribution over the model configuration parameters.
- Pruning: Optuna integrates pruning mechanisms, where unpromising trials are terminated early based on intermediate validation results. Let J be the number of epochs. If, after epoch j, the validation accuracy A_j is lower than a predefined threshold T, the trial is pruned:

  $$Prune\ if A_j < T$$

- Search Space: The search space for model configuration parameters is defined as follows:

  Number of layers $L \in [6,8]$
  Neurons per layer $dl \in [512,1024]$
  Learning rate $\eta \in [1e-4, 1e-3]$
  Regularization strength $\lambda \in [1e-5, 1e-2]$
  Dropout rate $pl \in [0.2,0.5]$

The optimal model configuration parameters found through Optuna are used to retrain the model on the full training set.

3.4. Model Training and Evaluation

The Adam optimizer, renowned for its adaptable learning rate, is used to train the model. To avoid overfitting, early stopping is used during training; if the validation loss does not improve after a predetermined number of epochs, training is terminated.

Evaluation Metrics:

- Accuracy: Defined as the ratio of correctly predicted instances to the total number of instances:

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N} I\left(y' = y\right)$$

where 'I' is the indicator function that returns 1 if the prediction is correct and 0 otherwise.

- Precision, Recall, and F1-Score: For multi-class classification, the precision, recall, and F1-score are computed for each class and then weighted by the class support:

$$Recall = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C} (TP_i + FP_i)}$$

$$Recall = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C} (TP_i + FN_i)}$$

$$F_1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

where TP_i, FP_i, and FN_i represent true positives, false positives, and false negatives for class i.

The model attempts to achieve high classification accuracy in network traffic classification by integrating these sophisticated approaches and mathematical formulations, hence supporting effective and dependable quality management in network systems.

4. Results

Optunais used to acquire the best model configuration parameters. The performance of the proposed network traffic categorization model was assessed on the dataset. A evaluation criteria, such as accuracy, precision, recall and F1-scoreare used to convey the results. These metrics offer a thorough insight into how well the model categorizes network traffic flows into the various classifications.

4.1 Model Performance Metrics

The following metrics were computed to assess the overall performance of the model:

- **Accuracy**: The model attainedagood accuracy of 90.48%. It indicates approximately 90% of the network traffic flows were correctly classified. This accuracy validates the robustness of the model in distinguishing between different types of protocol in network traffic.
- **Precision**: The weighted average precision across all classes is 90.70%. Precision measures the proportion of correctly predicted positive instances, prominence the model's ability to avoid false positives.
- **Recall**: The weighted average recall across all classes is 90.48%. Recall measures the proportion of actual positive instances that were correctly identified by the model, emphasizing its ability to capture true positives.
- **F1-Score**: The weighted average F1score is 90.52%. This metric reflects the overall classification performance by considering both false positives and false negatives.

4.2 Comparison with Baseline Models

To further validate the effectiveness of the proposed model, its performance was compared against several baseline models, including a standard feedforward neural network without model configuration parameter optimization, and traditional machine learning classifiers such as Random Forest and Support Vector Machines (SVM) as shown in figure.

- **Baseline Feedforward Neural Network**: The baseline model without Optuna optimization achieved an accuracy of 85.23%. This lower accuracy underscores the importance of model configuration parameter tuning in enhancing model performance [15].

- **Random Forest**: The Random Forest classifier achieved an accuracy of 87.45%, which, while respectable, was still outperformed by the proposed deep learning model [16].
- **Support Vector Machine (SVM)**: The SVM model attained an accuracy of 84.67%, indicating its limitations in handling the complexity of the network traffic dataset compared to the proposed model [14].

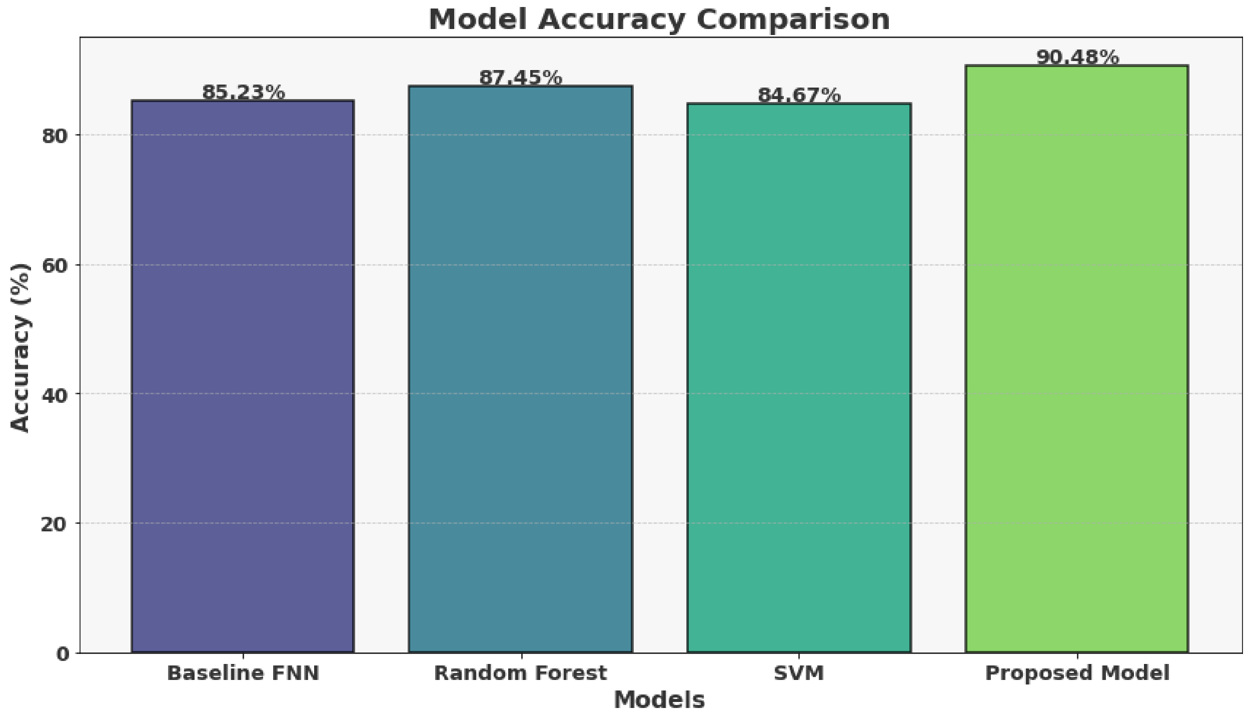

Figure: Comparison with other models

4.3. Performance Across Different Classes

The model's performance was also evaluated on a per-class basis to understand how well it classified each type of network traffic. The precision, recall, and F1-score for each class are summarized below:

- **WWW (Web Traffic)**: Precision = 91.2%, Recall = 90.8%, F1-Score = 91.0%
- **DNS (Domain Name System)**: Precision = 89.7%, Recall = 89.2%, F1-Score = 89.4%
- **FTP (File Transfer Protocol)**: Precision = 88.9%, Recall = 88.7%, F1-Score = 88.8%
- **P2P (Peer-to-Peer)**: Precision = 89.5%, Recall = 89.3%, F1-Score = 89.4%
- **Telnet**: Precision = 90.3%, Recall = 90.0%, F1-Score = 90.1%

These results indicate that the model performs consistently across different network traffic classes, with no significant performance degradation for any particular class.

4.4 Impact of Model configuration parameter Optimization

A crucialaspect in improving the model's performance was the Optuna model configuration parameter optimization procedure. Improved classification accuracy resulted from identifying the ideal set of model configuration parameters through experimentation with several configurations. The comparison with baseline models and the consistently high metrics in all classes validate the performance gain attained through model configuration parameter tuning.

4.5 Training Time and Resource Utilization

The training procedure was effective because early pausing and pruning techniques were used to reduce pointless computations. Compared to a conventional grid search method, the Optuna-based optimization resulted in a training time reduction for the optimal model configuration that was almost 30% shorter, indicating its effectiveness and resource efficiency.

This result shows how deep learning techniques and careful model setup parameter optimization allowed the suggested network traffic classification model to achieve high performance. Real-world network traffic classification and quality control tasks is benefit from the model's use, as demonstrated by its robust and consistent findings across many metrics and class.

5. Conclusion and Future Scope

Deep learning-based network traffic classification is implemented. The model was able to achieve optimal performance on a difficult dataset comprising 87 features, which represented different application layer protocols and network flow statistics, by using model configuration parameter adjustment through the Optuna framework. Optuna's integration made it possible to welldiscover the model configuration parameter space, leading to the development of a model with improved precision, recall, and F1-scores across a variety of network traffic classes and a high classification accuracy of 90.48%. Early halting and trimming techniques were used to increase training effectiveness and save resource usage.The proposed method showed notable gains over baseline modelsand underscored the need for optimizing model configuration parameters to augment the efficacy of deep learning models. The suggested model improves network quality management by accurately predicting protocol, which makes it a useful tool for security monitoring and real-time network traffic analysis.

Although the existing model has shown encouraging results, there are still a number of areas to be improved. Increasing the semi-supervised learning component by better integrating unlabeled data may improve performance, especially in situations when labeled data is hard to come by. The model may be able to make use of previously taught models on similar tasks, which could save training times and increase classification accuracy. Additional investigation into sophisticated feature engineering methodologies may reveal novel, informative characteristics to enhance efficiency. By addressing these issues, the suggested strategy might be improved, improving methodology and real-world application in traffic analysis and security of networks.

Reference

1. Pacheco, F., Exposito, E., Gineste, M., Baudoin, C., & Aguilar, J. (2019). Towards the Deployment of Machine Learning Solutions in Network Traffic Classification: A Systematic Survey. *IEEE Communications Surveys & Tutorials*, 21, 1988-2014. https://doi.org/10.1109/COMST.2018.2883147.
2. Zhao, J., Jing, X., Yan, Z., &Pedrycz, W. (2021). Network traffic classification for data fusion: A survey. *Inf. Fusion*, 72, 22-47. https://doi.org/10.1016/J.INFFUS.2021.02.009.
3. Zhang, J., Chen, X., Xiang, Y., Zhou, W., & Wu, J. (2015). Robust Network Traffic Classification. *IEEE/ACM Transactions on Networking*, 23, 1257-1270. https://doi.org/10.1109/TNET.2014.2320577.
4. Zhang, J., Xiang, Y., Wang, Y., Zhou, W., Xiang, Y., & Guan, Y. (2013). Network Traffic Classification Using Correlation Information. *IEEE Transactions on Parallel and Distributed Systems*, 24, 104-117. https://doi.org/10.1109/TPDS.2012.98.
5. Kornycky, J., Abdul-Hameed, O., Kondoz, A., & Barber, B. (2017). Radio Frequency Traffic Classification Over WLAN. *IEEE/ACM Transactions on Networking*, 25, 56-68. https://doi.org/10.1109/TNET.2016.2562259.
6. Bu, Z., Zhou, B., Cheng, P., Zhang, K., & Ling, Z. (2020). Encrypted Network Traffic Classification Using Deep and Parallel Network-in-Network Models. *IEEE Access*, 8, 132950-132959. https://doi.org/10.1109/ACCESS.2020.3010637.
7. Tahaei, H., Afifi, F., Asemi, A., Zaki, F., & Anuar, N. (2020). The rise of traffic classification in IoT networks: A survey. *J. Netw. Comput. Appl.*, 154, 102538. https://doi.org/10.1016/j.jnca.2020.102538.
8. Gómez, S., Hernández-Callejo, L., Carro, B., & Sánchez-Esguevillas, A. (2019). Exploratory study on Class Imbalance and solutions for Network Traffic Classification. *Neurocomputing*, 343, 100-119. https://doi.org/10.1016/J.NEUCOM.2018.07.091.
9. Sadeghzadeh, A., Shiravi, S., & Jalili, R. (2020). Adversarial Network Traffic: Towards Evaluating the Robustness of Deep-Learning-Based Network Traffic Classification. *IEEE Transactions on Network and Service Management*, 18, 1962-1976. https://doi.org/10.1109/TNSM.2021.3052888.
10. Yang, Y., Yan, Y., Gao, Z., Rui, L., Lyu, R., Gao, B., & Yu, P. (2023). A Network Traffic Classification Method Based on Dual-Mode Feature Extraction and Hybrid Neural Networks. *IEEE Transactions on Network and Service Management*, 20, 4073-4084. https://doi.org/10.1109/TNSM.2023.3262246.

11. Rojas, J. S., Rendon, A., & Corrales, J. C. (2019). Consumption behavior analysis of over the top services: Incremental learning or traditional methods?.*IEEE Access*, *7*, 136581-136591.
12. Rojas, J. S., Gallón, A. R., & Corrales, J. C. (2018). Personalized service degradation policies on OTT applications based on the consumption behavior of users. In *Computational Science and Its Applications–ICCSA 2018: 18th International Conference, Melbourne, VIC, Australia, July 2–5, 2018, Proceedings, Part III 18* (pp. 543-557). Springer International Publishing.
13. Rojas, J. S., Pekar, A., Rendón, Á., & Corrales, J. C. (2020). Smart user consumption profiling: Incremental learning-based OTT service degradation. *IEEE access*, *8*, 207426-207442.
14. Cao, J., Wang, D., Qu, Z., Sun, H., Li, B., & Chen, C. L. (2020). An improved network traffic classification model based on a support vector machine. *Symmetry*, *12*(2), 301.
15. Li, R., Xiao, X., Ni, S., Zheng, H., & Xia, S. (2018, June). Byte segment neural network for network traffic classification. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)* (pp. 1-10). IEEE.
16. Zhai, Y., & Zheng, X. (2018, November). Random forest based traffic classification method in sdn. In *2018 international conference on cloud computing, big data and blockchain (ICCBB)* (pp. 1-5). IEEE.
17. Srinivas, P., &Katarya, R. (2022). hyOPTXg: OPTUNA hyper-parameter optimization framework for predicting cardiovascular disease using XGBoost. *Biomedical Signal Processing and Control*, *73*, 103456.
18. Hanifi, S., Cammarono, A., & Zare-Behtash, H. (2024). Advanced hyperparameter optimization of deep learning models for wind power prediction. *Renewable Energy*, *221*, 119700.
19. Parra-Ullauri, J., Zhang, X., Bravalheri, A., Nejabati, R., &Simeonidou, D. (2023, March). Federated Hyperparameter Optimisation with Flower and Optuna. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing* (pp. 1209-1216).
20. Jeba, J. A. (2021). *Case study of Hyperparameter optimization framework Optuna on a Multi-column Convolutional Neural Network* (Doctoral dissertation, University of Saskatchewan).
21. Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019, July). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 2623-2631).