CSCI 335 Machine Learning: Group Project

C. Gagliardi, J. Alvarado, M. Richmond

1. Task Definition, Evaluation Protocol, and Data

Task: Implement a model to classify encrypted network traffic into 15 application classes using byte vectors as input features. The byte vectors are extracted from the first 1480 bytes from the IP payload in the traffic captures for each application in the dataset. The reference paper used to implement this task is Deep Packet [1].

Data Set: Utilizes the University of New Brunswick, VPN-nonVPN dataset [2]. The captured packets in this dataset are separated into different Packet Capture (PCAP) files. The files are labeled according to the application that generated the packets, such as Skype or Hangout.The original DeepPacket paper utilized a massive 26GB data set from the university of New Brunswick, due to hardware limitations we were unable to utilize this full dataset and instead used a reduced version of this data set.

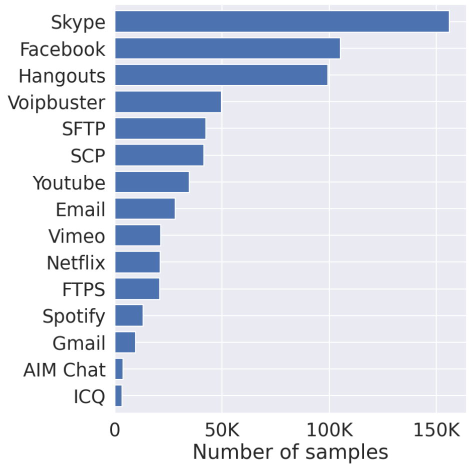


Figure 1: Number of samples for each application class [2].

Metrics: Performance measured using Recall(Rc), Precision(PR), and F1 Score(F1). These are calculated with the values of True Positive(TP), False Positive(FP), and False Negative(FN).[1]

$$\mathrm{Rc} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}, \quad \mathrm{Pr} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}, \quad F_1 = \frac{2 \cdot \mathrm{Rc} \cdot \mathrm{Pr}}{\mathrm{Rc}+\mathrm{Pr}}$$

Rc measures the proportion of actual positive instances that the model correctly predicts. A high recall means that the model is good at avoiding false negatives and is reliable for detecting positive instances.
Pr is a measure of the accuracy of a model's positive predictions. It indicates the proportion of predicted positive instances that are actually positive. A high precision means that the model is good at avoiding false positives and is reliable for making positive predictions.
F1 is a metric that combines precision and recall into a single score. It is calculated as the product of precision and recall divided by the sum of precision and recall. A high F1 score indicates that a model is making both accurate and complete predictions on a classification task.

2. Neural Network Machine Learning Model

Base Learning model:

Deep Packet [1] uses a one-dimensional Convolutional Neural Network (CNN). The model consists of two consecutive convolutional layers, followed by a max pooling layer. Next, the two-dimensional tensor is flattened into a one-dimensional vector and passed through three fully connected hidden layers with a dropout technique to avoid over-fitting. All layers employ Rectified Linear Unit (ReLU) as the activation function, except for the final softmax classifier layer. Categorical cross entropy and Adam were used as a loss function and optimizer, respectively. Categorical cross entropy measures the difference between the predicted probability distribution and the true probability distribution for each class. ADAM uses an adaptive learning rate to efficiently train the model. It adjusts the learning rate for each parameter, allowing the model to converge faster and with better results than traditional gradient descent. The network uses early stopping technique [3] to avoid overfitting the training data; in other words, the training stops when the loss function remains unchanged for several epochs on the training data.

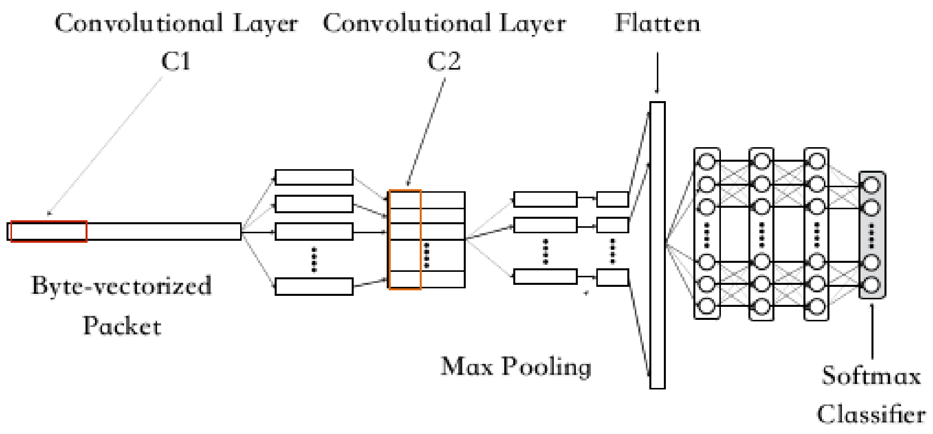


Figure 2: One-dimensional CNN architecture used for application classification model.

Experimental changes to be applied and compared:

The existing model uses Random Undersampling as a class balancing method. This algorithm randomly reduces the samples of all classes until they have the same number of samples as the minority class. The dataset used for the experiment has ICQ as the minority class with ~2700 samples; this means that all the other classes are reduced to the same number of samples as ICQ.

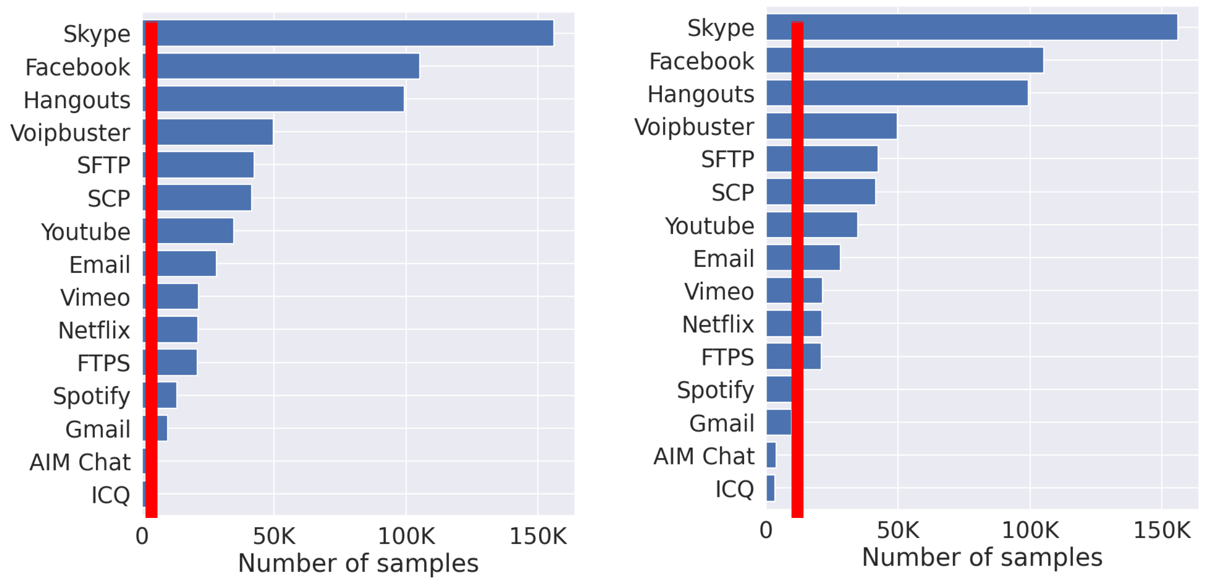


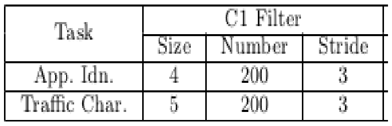
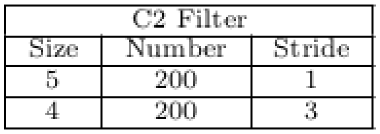
Figure 3: [Left] Red line showing the cutting point for Random Under-Sampling on training data. [Right] Red line showing a new cutting point after applying SMOTE and Random Under-Sampling on training data.

The proposed experiment on the model uses Synthetic Minority Over Sampling Technique (SMOTE)[5] as a class balancing method. This algorithm uses k-Nearest Neighbors to construct spatial relationships between samples of a given class. Then, it randomly selects a region encompassing its k-nearest neighbors and projects the feature space using the euclidean distance to construct a synthetic data point that can be used to increase the number of samples of the minority classes.

3. Experiment

Research question: What is the effect of using SMOTE (Synthetic Minority Over Sampling Technique) to generate synthetic samples for training the CNN model for application classification.

3.1. Design

Independent variables	**Control variables**	**Dependent variables**
SMOTE on Training data split Hyper-parameters: - Minority classes (c): 2 - Nearest Neighbors (k): 5 - Amount of SMOTE (n): [1, 2, 3, 4]	Testing data split CNN hyper-parameters:  	Recall, Precision, F1 Score
	- Training with early stopping Technique - Activation function: ReLU - Loss function: Categorical cross entropy - Output layer function:Softmax - Optimization function: Adam	Training time, Number of epochs to converge

Task	C1 Filter		
	Size	Number	Stride
App. Idn.	4	200	3
Traffic Char.	5	200	3

C2 Filter		
Size	Number	Stride
5	200	1
4	200	3

Hypothesis : Increasing the n parameter in SMOTE will result in a more balanced dataset causing the model to have a lower number of false negatives and a higher amount of true positives (better Precision, Recall, and F1 Scores.

3.2 Methodology

The codebase used was a deep convolutional neural network based on the Deep Packet paper [1]. A pytorch implementation base implementation was branched and can be accessed here https://github.com/munhouiani/Deep-Packet.

Data preprocessing is done to remove packet headers and timestamps from the pcap's using the preprocessing.py class This is not necessary since our repo at https://github.com/jja6522/Deep-Packet/ contains a zip file containing the processed captures we used. A second processing step is needed to create the test and train datasets along with doing SMOTE to synthesize new data points. The argument pattern for this is outlined on the github page along with some examples below.

The code we wrote was only inside of the create_train_test_set.py file. This is where we added the necessary functions to add SMOTE. We also changed the number of arguments that the script accepted to allow for k and n, variables used for SMOTE, to be easily adjusted from the command line.

Baseline algorithm: CNN model with undersampling only
Experimental algorithm: SMOTE data preprocessing used with the baseline algorithm and conditions for one variable are changing the n parameter from SMOTE from 1 to 5.

To run the experiment, first the data must be preprocessed using undersampling or SMOTE + undersampling for each n value. The testing and training data is created during this step, named based on the usage of SMOTE and its n value.
example) python create_train_test_set.py -s /home/stu15/s1/cgg3724/ml2022/project1/Deep-Packet/processed_small/ -t /home/stu15/s1/cgg3724/ml2022/project1/Deep-Packet/train_data_out_n4_k5/ --test /home/stu15/s1/cgg3724/ml2022/project1/Deep-Packet/test_data_out_n4_k5/ --class_balancing SMOTE+under_sampling -n 4 -k 5

Next the CNN model is trained on the training data.

example) python train_cnn.py -d /home/stu15/s1/cgg3724/ml2022/project1/Deep-Packet/train_data_out_n2_k5/application_classification/train.parquet/ -m model/application_classification.cnn.model.SMOTE.n2k5 -t app

Finally the model is tested on the testing data.

example) python test_cnn.py -d /home/stu15/s1/cgg3724/ml2022/project1/Deep-Packet/test_data_out_n1_k5/application_classification/test.parquet/ -m model/application_classification.cnn.model.SMOTE.n1k5 -t app

4. Experimental Results and Discussion

The baseline model for the experiment was isolated, re-trained and tested with a reduced data sample due to computational limitations. This model used random undersampling as a class balancing technique applied on the training data. The metrics used to evaluate the model performance were summarized in a precision-recall curve as shown in Figure 4.

The experimental model was modified to include a preprocessing step to apply SMOTE and Random undersampling to balance the class distribution. The metrics used to evaluate the model performance and compare it with the baseline were summarized in Figure 4.

The minority classes that were synthetically oversampled were AIM Chat and ICQ; the number of samples for those two classes was increased from 3132 and 2788 to 6189 and 5477, respectively. The classification model was re-trained with that enhanced training split and tested with the base test split used in the experiment as a control variable.

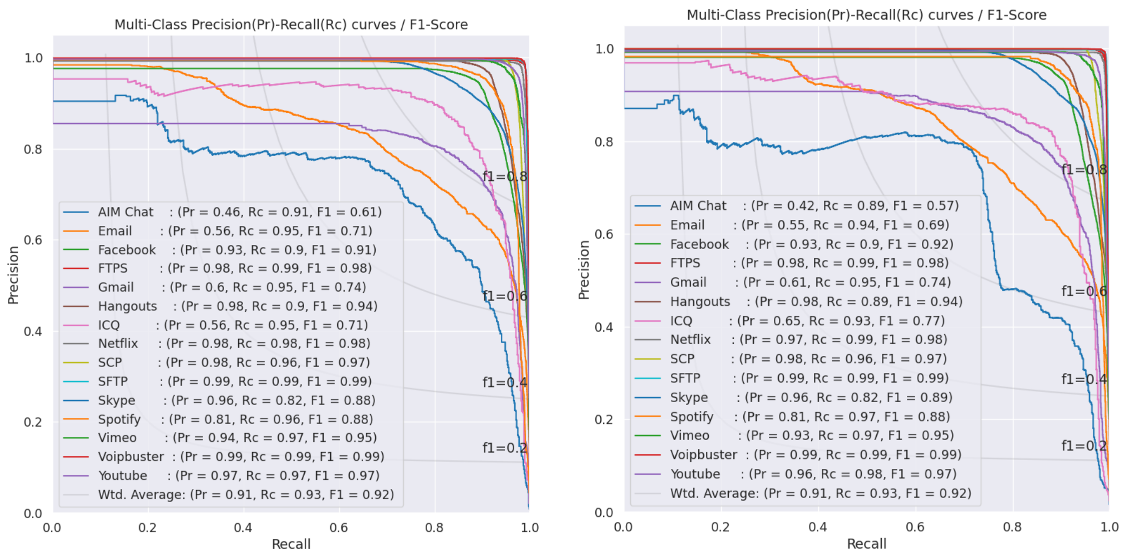
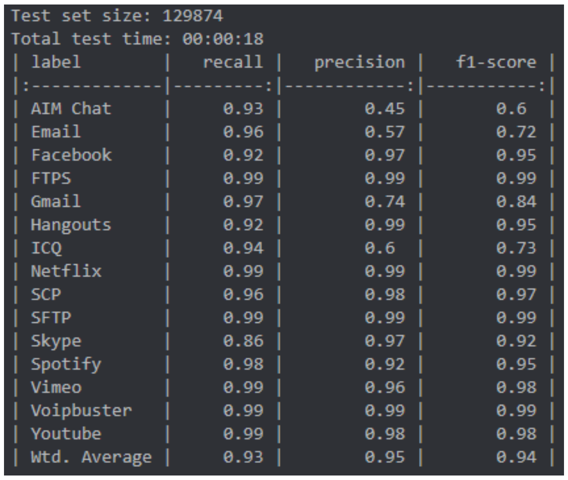
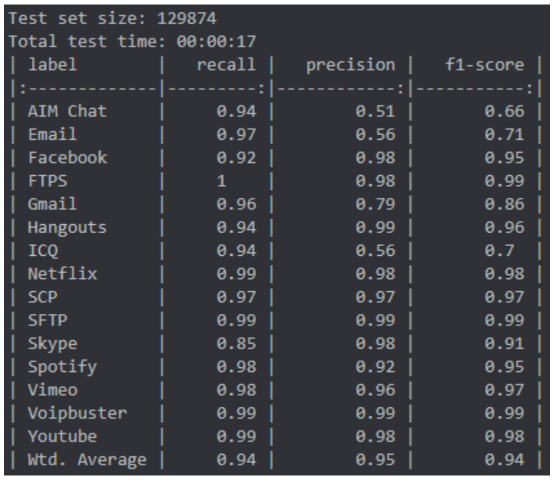


Figure 4: [Left] Precision-Recall curves for the base model with Random Undersampling. [Right] Precision-Recall curves for the experimental model with SMOTE and Random Undersampling applied (hyper-parameters: c = 2, n =2, k =5).

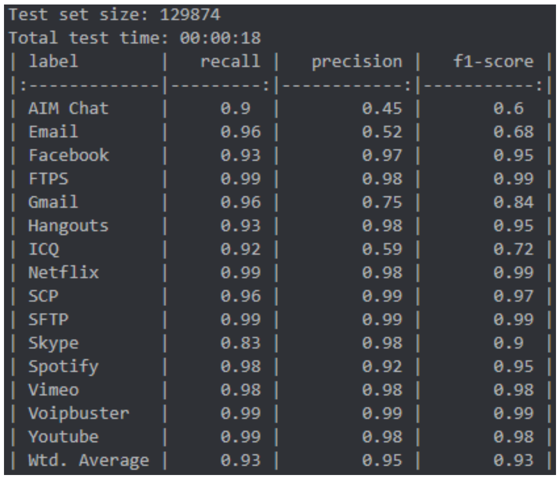
Below the subsequent list of results obtained for SMOTE with undersampling using the proposed list of hyperparameters for the experiment: c = 2, n = [2, 3, 4, 5], k =5 (left to right, top to bottom).



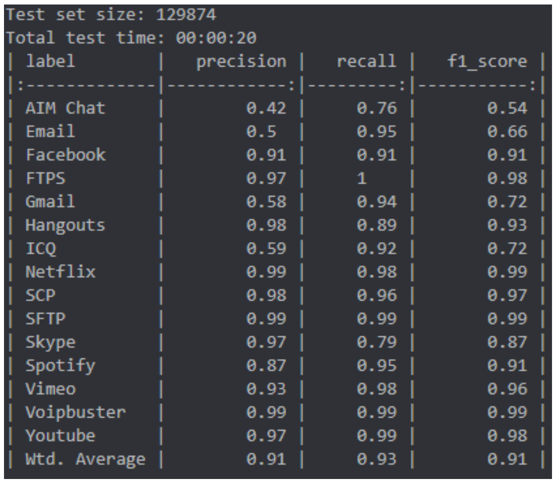
```
Test set size: 129874
Total test time: 00:00:18
| label         |   recall |   precision |   f1-score |
|:--------------|---------:|------------:|-----------:|
| AIM Chat      |     0.93 |        0.45 |       0.6  |
| Email         |     0.96 |        0.57 |       0.72 |
| Facebook      |     0.92 |        0.97 |       0.95 |
| FTPS          |     0.99 |        0.99 |       0.99 |
| Gmail         |     0.97 |        0.74 |       0.84 |
| Hangouts      |     0.92 |        0.99 |       0.95 |
| ICQ           |     0.94 |        0.6  |       0.73 |
| Netflix       |     0.99 |        0.99 |       0.99 |
| SCP           |     0.96 |        0.98 |       0.97 |
| SFTP          |     0.99 |        0.99 |       0.99 |
| Skype         |     0.86 |        0.97 |       0.92 |
| Spotify       |     0.98 |        0.92 |       0.95 |
| Vimeo         |     0.99 |        0.96 |       0.98 |
| Voipbuster    |     0.99 |        0.99 |       0.99 |
| Youtube       |     0.99 |        0.98 |       0.98 |
| Wtd. Average  |     0.93 |        0.95 |       0.94 |
```



```
Test set size: 129874
Total test time: 00:00:17
| label         |   recall |   precision |   f1-score |
|:--------------|---------:|------------:|-----------:|
| AIM Chat      |     0.94 |        0.51 |       0.66 |
| Email         |     0.97 |        0.56 |       0.71 |
| Facebook      |     0.92 |        0.98 |       0.95 |
| FTPS          |     1    |        0.98 |       0.99 |
| Gmail         |     0.96 |        0.79 |       0.86 |
| Hangouts      |     0.94 |        0.99 |       0.96 |
| ICQ           |     0.94 |        0.56 |       0.7  |
| Netflix       |     0.99 |        0.98 |       0.98 |
| SCP           |     0.97 |        0.97 |       0.97 |
| SFTP          |     0.99 |        0.99 |       0.99 |
| Skype         |     0.85 |        0.98 |       0.91 |
| Spotify       |     0.98 |        0.92 |       0.95 |
| Vimeo         |     0.98 |        0.96 |       0.97 |
| Voipbuster    |     0.99 |        0.99 |       0.99 |
| Youtube       |     0.99 |        0.98 |       0.98 |
| Wtd. Average  |     0.94 |        0.95 |       0.94 |
```



```
Test set size: 129874
Total test time: 00:00:18
| label         |   recall |   precision |   f1-score |
|:--------------|---------:|------------:|-----------:|
| AIM Chat      |     0.9  |        0.45 |       0.6  |
| Email         |     0.96 |        0.52 |       0.68 |
| Facebook      |     0.93 |        0.97 |       0.95 |
| FTPS          |     0.99 |        0.98 |       0.99 |
| Gmail         |     0.96 |        0.75 |       0.84 |
| Hangouts      |     0.93 |        0.98 |       0.95 |
| ICQ           |     0.92 |        0.59 |       0.72 |
| Netflix       |     0.99 |        0.98 |       0.99 |
| SCP           |     0.96 |        0.99 |       0.97 |
| SFTP          |     0.99 |        0.99 |       0.99 |
| Skype         |     0.83 |        0.98 |       0.9  |
| Spotify       |     0.98 |        0.92 |       0.95 |
| Vimeo         |     0.98 |        0.98 |       0.98 |
| Voipbuster    |     0.99 |        0.99 |       0.99 |
| Youtube       |     0.99 |        0.98 |       0.98 |
| Wtd. Average  |     0.93 |        0.95 |       0.93 |
```



```
Test set size: 129874
Total test time: 00:00:20
| label         |   precision |   recall |   f1_score |
|:--------------|------------:|---------:|-----------:|
| AIM Chat      |        0.42 |     0.76 |       0.54 |
| Email         |        0.5  |     0.95 |       0.66 |
| Facebook      |        0.91 |     0.91 |       0.91 |
| FTPS          |        0.97 |     1    |       0.98 |
| Gmail         |        0.58 |     0.94 |       0.72 |
| Hangouts      |        0.98 |     0.89 |       0.93 |
| ICQ           |        0.59 |     0.92 |       0.72 |
| Netflix       |        0.99 |     0.98 |       0.99 |
| SCP           |        0.98 |     0.96 |       0.97 |
| SFTP          |        0.99 |     0.99 |       0.99 |
| Skype         |        0.97 |     0.79 |       0.87 |
| Spotify       |        0.87 |     0.95 |       0.91 |
| Vimeo         |        0.93 |     0.98 |       0.96 |
| Voipbuster    |        0.99 |     0.99 |       0.99 |
| Youtube       |        0.97 |     0.99 |       0.98 |
| Wtd. Average  |        0.91 |     0.93 |       0.91 |
```

Discussion:

The previous tables show the results of changing the n parameter of SMOTE used with a CNN model for network traffic. The n value determines the number of synthetic data points generated for each sample in the minority class of an imbalanced dataset. Increasing the n value leads to more synthetic samples being generated, which can improve the balance of the dataset and potentially improve the performance of the CNN model. However, a larger n value can also lead to overfitting, as the model is trained on a higher proportion of synthetic samples that may not generalize well to unseen data. The k value is the number of nearest neighbors used to create a synthetic datapoint between.

In this experiment, the use of SMOTE generally improved the overall scores of the model, though in some classes the scores decreased slightly such as traffic from

ICQ which had the highest recall score of 0.95 without SMOTE. The baseline weighted averages without SMOTE were Rc = 0.91 Pr = 0.93 and f1-score = 0.92. SMOTE applied with an n value of 1 produced weighted averages that saw an improvement from only undersampling and were Rc = 0.92 Pr = 0.94 and f1-score = 0.93. As the n value increased from 1 to 3, the recall, precision, and f1 scores generally increased, with n = 3 producing the best scores overall which were Rc = 0.94 Pr = 0.95 and f1-score = 0.95. This indicates that the hypothesis is true because the number of False Negatives and False Positives lowered, and True Positives raised. However, as n increased from 3 to 5, the scores began to decrease, indicating that overfitting may have occurred.

The scores of classes that did not have SMOTE-generated data also changed, likely due to the shift in the overall balance of the dataset caused by SMOTE synthesis. As the number of synthetic samples increased, the distribution of classes in the dataset changed, which may have affected the performance of the model on classes without synthetic data.

The score increases on our minority classes was very low and the weighted average was almost constant due to decreases in majority class precision, accuracy and f1 score. This volatility between tests suggests that the randomness from SMOTE may be the factor that is causing the score increases and not necessarily the larger training set.

Overall our experiment results were inconclusive. The weighted average for all three metrics also appears to decline as we increase n beyond 3. The changes in scores produced by applying different n values did not seem to be a definite confirmation for our hypothesis. More experimentation is required to answer our research question of whether or not SMOTE+Undersampling can be used to generate usable training data for encrypted internet traffic.

5. References

Lotfollahi, M., Jafari Siavoshani, M., Shirali Hossein Zade, R. et al. Deep packet: a novel approach for encrypted traffic classification using deep learning. Soft Comput 24, 1999–2012 (2020). https://doi.org/10.1007/s00500-019-04030-2
Implementation: https://github.com/munhouiani/Deep-Packet [1]

Gerard Drapper Gil, Arash Habibi Lashkari, Mohammad Mamun, Ali A. Ghorbani, "Characterization of Encrypted and VPN Traffic Using Time-Related Features", In Proceedings of the 2nd International Conference on Information Systems Security and Privacy(ICISSP 2016) , pages 407-414, Rome, Italy. [2]

Chawla, Nitesh V., et al. "SMOTE: synthetic minority over-sampling technique." Journal of artificial intelligence research 16 (2002): 321-357. [3]

Jayanth Sivakumar, Karthik Ramamurthy, Menaka Radhakrishnan, and Daehan Won. "Synthetic sampling from small datasets: A modified mega-trend diffusion approach using k-nearest neighbors." Knowledge-Based Systems (2021): 107687. [4]

He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. [5]

Prechelt L (1998) Early stopping-but when? In: Neural Networks: Tricks of the trade, Springer, pp 55–69 [6]