Streaming Video Classification Using Machine Learning

Adnan Shaout and Brennan Crispin
The Electrical and Computer Engineering, the University of Michigan-Dearborn, Michigan

Abstract: *This paper presents a method using neural networks and Markov Decision Process (MDP) to identify the source and class of video streaming services. The paper presents the design and implementation of an end-to-end pipeline for training and classifying a machine learning system that can take in packets collected over a network interface and classify the data stream as belonging to one of five streaming video services: You Tube, You Tube TV, Netflix, Amazon Prime, or HBO.*

Keywords: *Machine Learning, Neural Networks, Deep Packet Inspection, MDP, Video Streaming, AI.*

Received February 29, 2020; accepted June 9, 2020
https://doi.org/10.34028/iajit/17/4A/13

1. Introduction

Methods, such as IP and port scanning, are not always available for clients using Virtual Private Networks (VPNs) or with providers (You Tube, You Tube TV, Netflix, Amazon Prime, or HBO) using varying IP addresses to be able to identify the types of data and content providers that are being used on their networks. Neural networks and Markov Decision Process (MDP) are potential methods in identifying the source and class of video streaming services. This paper is an extension to our paper [18], which was present at ACIT20 UAE.

The objective of this paper is to design and implement an end-to-end pipeline for training and classifying a machine learning system that can take in packets collected over a network interface and classify the data stream as belonging to one of five streaming video services: You Tube, You Tube TV, Netflix, Amazon Prime, or HBO. This paper will layout method employing Markov Decision Process applied to a simple multi-layer perceptron neural network in order to more accurately classify these video services.

Previously, it has been possible for researchers to identify network traffic and by extension streaming services using IP addresses and ports; however, due to increasing optimizations, service providers no longer use reliably identifiable ports [14]. As such, new methods are being developed that make use of encrypted data and characteristics of the packets and traffic to better identify traffic, however, very little of this has been used to classify different streaming services, and so few tools are available to discriminate between the services for providers.

The rest of this paper will be organized as follows: Section 2 describes previous work in classifying network traffic and streaming data. Section 3 describes the methodology used to collect our datasets and for the implementation of our classifiers. Section 4 covers experimental results and analysis. Section 5 is discussion and conclusion.

2. Related Work

A lot of work in network traffic classification has been done [14] that has focused on using NetFlow [8] to identify traffic type or destination, rather than traditional deep packet inspection. For example, Erman *et al.* [5] proposed a traffic classification system that only uses flow stats. They also provided a method to estimate network flow at both edge and core network nodes [6]. Zhang *et al.* [22] proposed a method to classify network traffic using correlation with comparatively low sample sizes for their data.

Further attempts to improve classification performance appear in Moore and Zuev [13] and Auld *et al.* [2], in which a supervised naïve Bayes tree is applied to estimate traffic identity, and further extended using neural networks based on Bayesian methods to classify internet traffic using flow data. BLINd Classification (BLINC) [9] also attempted to classify hosts and associate them with applications rather than classifying pure IP flow. Bernaille and Teixeira [3], used the size of first packets to identify applications by using deep packet inspection.

Most relevant to classifying video, traffic, however, is work by Bonfiglio *et al.* [4], in which they identify VOIP related traffic to identify skype traffic on a network. There has been an increase in interest in how deep learning techniques can be applied to network traffic classification [1, 10, 17, 19, 20]. Work was done to estimate You Tube Quality of Experience (QoE) [15, 16] metrics from packet metadata by applying several techniques to estimate these packet statistics. Fast Orthogonal Search applied to a k Nearest Neighbor (kNN) classifier to select optimal feature sets

[12] was also presented. Finally, Hubballi and Mayank [7] described using bit-level information to identify the differences in bit-coding signatures between different sources using a Hamming distance for the bit coding to discern different traffic sources.

3. Implementation

This section discusses the data collection methodology, data pre-processing steps, and algorithms used to classify traffic.

A. *Data Collection*: Data was collected using Wireshark [21] to collect raw Packet Captures (PCAPS) of streaming video and then aggregated together with named labels for each streaming service being studied. The data was collected by connecting to a streaming service and collecting all packets sent to and received by the local computer during this time.

B. *Data Preprocessing*: In order to make the raw PCAPS collected using Wireshark usable for data analysis, several steps were taken.

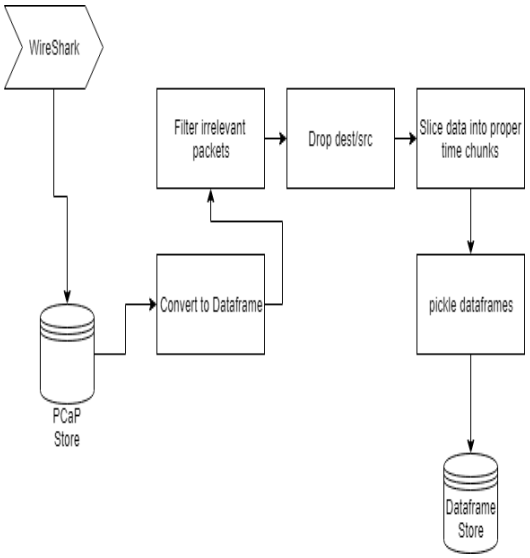

Figure 1. Pre-processing box diagram.

First, the data was loaded and transformed them into a Pandas dataset for use with Python's machine learning libraries.

As part of this pre-processing step, shown in Figure 1, data was taken that had been captured at the data-link layer (which includes information such as MAC addresses). As part of this step, we removed certain elements, such as packets that are considered irrelevant: Synchronous (SYN), Acknowledge (ACK), Finish (FIN) flags, since they are used for hand shake generation and do not contain any useful information about the protocol or service. Additionally, we deleted all (Destination) DNS and similar segments from the dataset. While these can be extremely useful for service identification, as DNS lookups can return information on the identity of the IP address, they do not provide useful information for traffic characterization and classification. Further, DNS look-ups may provide one-to-one associations between the IP address and the streaming service, so while extremely precise within our dataset they would be useless generalizing the model outside of it.

Finally, certain columns were removed from the dataset, such as IP address, since these are generally 1-1 correlation with services, which would create a severe data leakage in our models - the models would collapse to using IP address solely, but be unable to identify streams that come from new IP addresses. Finally, packets were transformed to bytes from bits to reduce input size in the neural networks being used.

C. *Neural Network:* After pre-processing of the raw PCAP data collected from the streaming services, the 23 remaining PCAP columns are passed into a simple multi-layer perceptron neural network to create a set of classification probabilities on the possible outputs. In order to ensure that the features later passed to the Markov decision tree would be of maximum quality, we then iterated on varying configurations of the neural network to find the highest performing number of hidden layers and other meta-features. The neural network was programmed using Matlab'sMatConvNet [11,19] library.

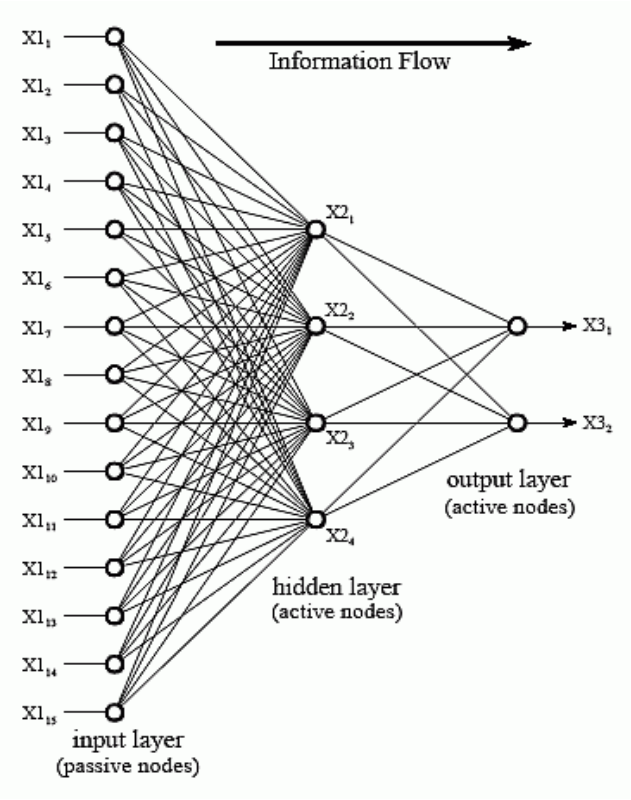

Figure 2. Starting neural network structure.

Figure 2 shows the starting NN structure. The initial architecture of the neural network was simple:

- 23 nodes in the input layer.
- A single hidden layer containing 4 nodes.

- A ReLU transform function.
- A final 5 node output layer.

After this test was run, further variations in the architecture of the neural network were explored to see which structures would generate the best results. The hyper-parameter range explored included additional hidden layers, increased nodes in the hidden layers, and an exploration of alternative activation functions such as sigmoid. Additionally, the batch size and learning rate were varied.

Figure 3 shows the diagram for training the NN.

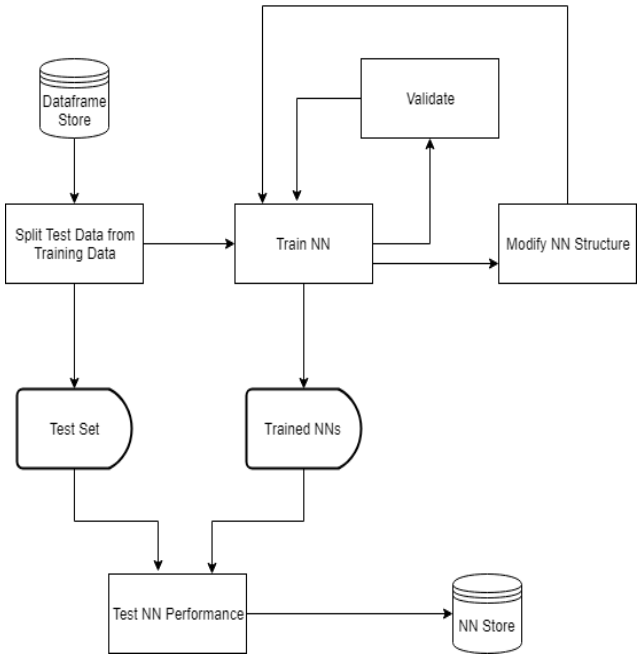

Figure 3. Box diagram for training the NN.

D. *Markov Decision Process*: With the best results from the neural network used as an input set, we attempted to use a Markov Decision Process to improve the accuracy of the classifications provided by the neural network. Instead of taking the classification provided by the neural network as the output, however, the set of probabilities created by MatConvNet were used to create a new set of features for each state in the Markov decision process.

To construct the Markov Decision chain, the output classification probabilities from each time slice generated by the neural network was used to create a series of 'states', where for each state a set of 5 probabilities were provided as features.

For each time slice, treated as a discrete state S, there was a potential subsequent state S` as well as 5 additional final states: one for each of the possible classification outcomes. Therefore, for any possible state, there would be 6 potential actions: wait for the next time slice, or go to one of the 5 final classification states. Since we would want a final decision to be made, the fifth time slice was chosen as a final 'undecided' state (Figure 4).

The decision rule was simple: there would be a minimum probability threshold required to be met by each state-if one state met that threshold then the action would be to progress to the selection state for that probability. If multiple met the condition, then the max would be selected as the action. If none met the threshold, then the action would be to proceed to the next time slice state.

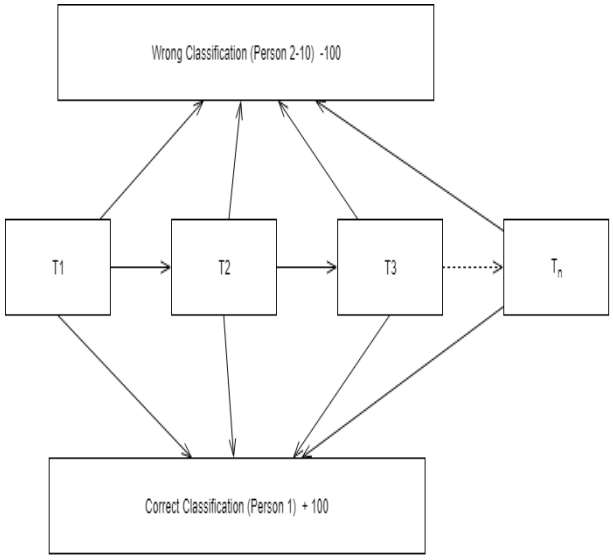

Figure 4. Markov decision process.

In order to further improve the results for the Markov Decision Process, a second policy set was created using multiple thresholds for all possible classification categories. This was then run through multiple iterations with changes to the threshold values to improve the final prediction output. To find the optimal thresholds, a reward function was associated with each of the output states. For a correct state a +100 score was assigned, and -100 was assigned for an incorrect score. Timeouts were set at 0. In order to optimize for fast resolutions, a gamma of 0.9 was added to minimize the value of later decisions being made.

Once the Markov Decision Process (MDP) had been designed, an initial test policy was created -for each state there would be a minimum threshold that the maximum probability had to clear. If that threshold was cleared, then the policy would select the 'Classify as X' action. If not, then the policy would progress to the next time state. This policy was implemented in Python.

E. *Summary of MDP Algorithm*

1. Assign initial thresholds to each state.
2. Calculate classification probabilities using optimized neural network to get 5 probabilities for each time slice.
3. For each time slice, get the next five time slice probabilities and generate states {s…s_n.

4. For each state s, check whether probability exceeds threshold, if yes, go to final state, else go to s`
5. Repeat 4 until either in classification state or timeout state.
6. Check classification against test data label and modify threshold accordingly.
7. If misclassified, raise threshold for selected stream. if timeout, lower threshold for correct stream
8. Go to step 3 until minimum error is met.

F. *Evaluation*: In order to evaluate the results from the Markov Decision Process and properly compare its results to those from the neural network, a test set was taken from the total dataset to be used for evaluation.

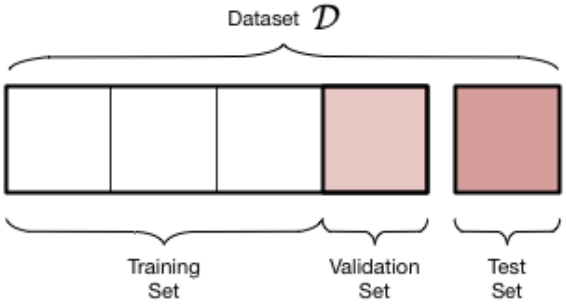

Figure 5. Training and evaluation sets.

Some 70% of the dataset was randomly selected and placed into the training set. Further, 20% of the data was used for validation and a final 10% was placed into the test set as shown in Figure 5.

For the validation set, data was used to fine-tune our results and to guard against overfitting by the training set. The accuracy of prediction from models trained on the training set was tested against the validation set. Poor results were indicative that the hyper-parameters of the model needed fine-tunings.

Once the training and validation sets had settled, we took the trained networks and had them classify each element from the labeled test set. In order to evaluate the accuracy of the trained model, the number of correct classifications was counted and a simple percentage of correct classifications was used to model the effectiveness of the MDP as compared to the simple neural networks used to feed it data.

4. Results

This section presents the results for the proposed classification method. A comparison was run of the accuracy against the known classification for both the training set and the testing set.

A. *Neural Network*: Figure 6 shows the final accuracy of 99.7% in the training set and 73.1% in the validation set. The relative disconnect between the training and validation sets indicates that overfitting was an issue with the data, and that further modifications to the neural network structure were unlikely to improve results.

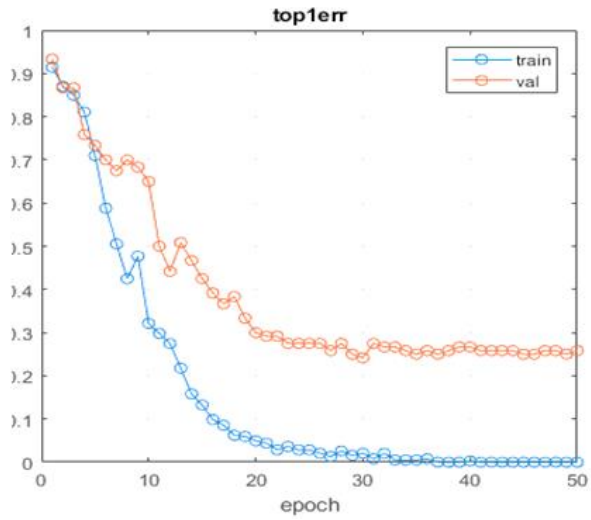

Figure 6. Training and validation accuracy per epoch.

To prove the overfitting, when modifying the number of hidden layers, the accuracy in the validation set generally went down as shown in Table 1.

Table 1. Prediction accuracy for layer parameters.

Variation	Training	Validation
2 Hidden Layers	91.5%	68.5%
3 Hidden Layers	92.2%	69.2%
4 Hidden Layers	94.1%	70.1%

Table 2 shows that tests with a single hidden layer but variances in the number of nodes have shown even more dramatic loss in accuracy.

Table 2. Prediction accuracy for node parameters.

Variation	Training	Validation
10 Hidden Nodes	90%	68.1%
15 Hidden Nodes	82%	60.4%
23 Hidden Nodes	33%	40%

B. *Cross Validation*: In order to solve the overfitting issue, K-fold cross validation was used. Data was broken into 5 separate validation sets, with the remainder used for training data. Table 3 shows that an average success rate of 81.7% was found, indicating that cross validation would be a good avenue to improve results for training the networks.

Table 3. Prediction accuracy for each service.

Stream	Accuracy
Amazon Prime	0.84
Netflix	0.82
HBO	0.86
You Tube	0.77
You Tube TV	0.75

C. *Markov Decision Process*: Originally, we had results in the 73% range; including the MDP in the decision making process improved results to around 85% on average. Table 4 shows that the classification results have improved to an average over 90%, even for the markedly poor results of You Tube TV. This was achieved by adding a

regression policy to fine-tune the thresholds used by the MDP for each possible classification.

Table 4. Prediction accuracy for each service.

Stream	Threshold	Accuracy
Amazon Prime	0.51	0.91
Netflix	0.75	0.92
HBO	0.6	0.902
You Tube	0.85	0.85
You Tube TV	0.9	0.845

5. Conclusions

Neural networks appear to have the accuracy required to make classification useful while there are many other methods for classifying streaming data based on collected PCaps. A trained neural network with a MDP enhancement was shown to handle the classification.

This paper presents a method that allows for discriminating between streaming services using similar, but not identical protocols. Others have used neural networks to discriminate between VOIP and non-VOIP traffic, or to classify disparate types of traffic using supervised neural networks [17]. The method presented in this paper allowed similar types of streaming traffic from different services to be classified accurately.

References

[1] Archanaa R. Athulya V., Rajasundari T., and Kiran M., "A Comparative Performance Analysis on Network Traffic Classification Using Supervised Learning Algorithms," *in Proceedings of 4th International Conference on Advanced Computing and Communication Systems*, Coimbatore, pp. 1-5, 2017.

[2] Auld T., Moore A., and Gull S., "Bayesian Neural Networks for Internet Traffic Classification," *IEEE Trans. Neural Networks*, vol. 18, no. 1, pp. 223-239, 2007.

[3] Bernaille L. and Teixeira R., "Early Recognition of Encrypted Applications," *in Proceedings of International Conference on Passive and Active Network Measurement*, Louvain-la-neuve, pp. 165-175, 2007.

[4] Bonfiglio D., Mellia M., Meo M., Rossi D., and Tofanelli P., "Revealing Skype Traffic: When Randomness Plays with You," *in Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Kyoto, pp. 37-48, 2007.

[5] Erman J., Mahanti A., Arlitt M., and Cohen I., "Semi-Supervised Network Traffic Classification," *ACM SIGMETRICS Performance Evaluation Review*, vol. 35, no. 1, 2007.

[6] Erman J. Mahanti A., Arlitt M., and Williamson C., "Identifying and Discriminating between Web and Peer-to-Peer Traffic in the Network Core," *in Proceedings of the 16th International Conference on World Wide WebMay*, Banff, pp. 883-892, 2007.

[7] Hubballi N. and Mayank S., "BitCoding: Network Traffic Classification Through Encoded Bit," *IEEE/ACM Transactions on Networking*, vol. 26, no. 5, pp. 2334-2346, 2018.

[8] Hofstede R., Čeleda P., Trammell B., Drago I., Sadre R., Sperotto A., and Pras A., "Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX," *IEEE Communications Surveys and Tutorials*, vol. 16 , no. 4 , pp. 2037-2064, 2014.

[9] Karagiannis T., Konstantina P., and Michalis F., "BLINC: Multilevel Traffic Classification in the Dark," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, 2005.

[10] Lotfollahi M., Zade R., Siavoshani M., Saberian M., "Deep Packet: A Novel Approach for Encrypted Traffic Classification Using Deep Learning," *Soft Computing*, vol. 24, pp. 1999-2012, 2020.

[11] "MatConvNet: CNNs for MATLAB" MatconvNet. http://www.vlfeat.org/matconvnet/ Last Visited, 2020.

[12] McGaughey D. Semeniuk T., Smith R., and Knight S., "A Systematic Approach of Feature Selection for Encrypted Network Traffic Classification," *in Proceedings of Annual IEEE International Systems Conference*, Vancouver, pp. 1-8, 2018.

[13] Moore A. and Zuev D., "In¬¬ternet Traffic Classification Using Bayesian Analysis Techniques," *ACM SIGMETRICS Performance Evaluation Review (SIGMETRICS*, vol. 33, pp. 50-60, 2005.

[14] Nguyen T. and Armitage G., "A Survey of Techniques for Internet Traffic Classification Using Machine Learning," *IEEE Communications Surveys and Tutorials*, vol. 10, no. 4, pp. 56-76, 2008.

[15] Orsolic I., Pevec D., Suznjevic M., and Skorin-Kapov L., "You Tube QoE Estimation Based on the Analysis of Encrypted Network Traffic Using Machine Learning," *in Proceedings of Globecom Workshops (GC Wkshps)*, Washington, 2016.

[16] Ran D., Hadar O., Richman I., Trabelsi O., Dvir A., and Peles O., "Video Quality Representation Classification of Safari Encrypted DASH Streams," *in Proceedings of Digital Media Industry and Academic Forum (DMIAF).* Santorini, 2016.

[17] Shafiq M., Yu X., Laghari A., Yao L., Karn N., and Abdessamia F., "Network Traffic Classification Techniques and Comparative Analysis Using Machine Learning Algorithms," *in Proceedings of 2nd IEEE International Conference on Computer and Communications*, Chengdu, 2016.

[18] Shaout A. and Crispin B., "Markov Augmented Neural Networks for Streaming Video Classification," *in Proceedings of the IEEE International Arab Conference on Information Technology*, Al Ain, pp. 1-7, 2019.
[19] Shaout A., Mysuru D., Raghupathy K., "Vehicle Condition, Driver Behavior Analysis and Data Logging Through CAN Sniffing," *The International Arab Journal of Information Technology*, vol. 16, no. 3A, pp. 493-498, 2019.
[20] Wei W., Zhu M., Wang J., and Zeng X., "End-To-End Encrypted Traffic Classification With One-Dimensional Convolution Neural Networks," *in Proceedings of IEEE International Conference on Intelligence and Security Informatics*, Beijing, 2017.
[21] Wireshark, Wireshark, www.wireshark.org, Last Visited, 2020.
[22] Zhang J., Xiang Y., Wang Y., Zhou W., Xiang Y., and Guan Y., "Network Traffic Classification Using Correlation Information," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 104-117, 2013.

Adnan Shaout is a full professor and a Fulbright Scholar in the Computer Science Department at the Electrical and Computer Engineering Department at the University of Michigan–Dearborn. His current research is in applications of software engineering methods, embedded systems, fuzzy systems, real time systems and AI. Dr. Shaout has published over 260 papers in topics related to Computer Science, Electrical and Computer Engineering fields. Dr. Shaout has obtained his B.S.c, M.S. and Ph.D. in Computer Engineering from Syracuse University, Syracuse, NY, in 1982, 1983, 1987, respectively.

Brennan Crispin has an MS degree in Software Engineering from the University of Michigan – Dearborn. He is currently working as a Software Engineer at Deepfield-Ann Arbor, MI.