Proactive Flow Rule Installation in SDN using Machine Learning Techniques

Haider Ali
Virginia Tech

Momina Nofal
LUMS

Muhammad Fareed Zaffar
LUMS

ABSTRACT

Software Defined networking (SDN) has become widely popular in recent years, a necessary abstraction and step-up from traditional routing. While SDN offers operational flexibility, it may induce network latency for unknown flows by fetching the flow rule from the controller to the switch. To address this latency problem, we attempted to predict future flows using a flow popularity approach. Flow popularity of IP source/destination (src/dst) pair is the function of the number of flows of that pair over a certain time period. Assuming that we know the flow rule for a particular src/dst pair, then we can install the flow rule proactively if we know which pair's flow rule is to be installed. We want to therefore predict the popularity of IP source/destination pairs based on the first few seconds (20,30 seconds etc.) of the pair's behavior (features). Then we can install flow rules for the most popular pairs. Using CAIDA 2019 Data center dataset and the supervised learning classification approach, we first established the ground truth of popularity based on the number of flows of a pair in 4 minutes and then extracted features of the first few seconds of those pairs. We used 300 different types of models which include 13 types of classifiers like Support Vector Machine (SVM) for prediction of popularity. We gained initial accuracy upto 74% using SVM. Later, we redefined the ground truth of popularity using different data labeling techniques and we also used over-sampling techniques in order to cater imbalanced dataset. In this way, we got interesting results of accuracies up to 98%. We found out that recall is a better metric for imbalanced data and we got up to 93% recall. Since a time-series based approach might be more suitable to this scenario instead of a classification approach, we are now exploring an approach involving LSTM and GRU, where we analyze the flows of each pair separately to predict when the next one will come. We have replicated our results thus far on another dataset, university traces, to strengthen our results and also further smoothen our data preprocessing pipeline.

KEYWORDS

SDN, CAIDA, Data center, proactive approach, SVM, ANN, Logistic Regression, Decision Tree, Machine Learning, LSTM, RNN

ACM Reference Format:
Haider Ali, Momina Nofal, and Muhammad Fareed Zaffar. 2021. Proactive Flow Rule Installation in SDN using Machine Learning Techniques. In , .

1 INTRODUCTION

Before the deployment of SDNs in networks, forwarding and controlling functions were confined in individual routers. If any changes were to be made in the infrastructure they needed to be done individually at the router or switches. However, changes in the network can happen quite fast due to link failures and changing network policy. SDN has emerged as a network paradigm which separates the control plane and the forwarding plane. Its key feature is that it enables programming the network by a logically centralized controller, where the switches are simply forwarding devices while controller holds the power of decision making. SDN helps in designing innovative network functions and protocols in a much easier, flexible, and more powerful way.

SDN can install flow table entries in two ways:

- Proactive flow management: In the proactive approach, the controller is able to install flow entries before they are actually needed. This approach is disastrous if a large number of flow table entries are installed. Because in practice, the TCAM memory of the switches is fundamentally limited in size due to its high cost and power consumption. Thus the vast majority of SDN hardware vendors limit the TCAMs to less than 4K L2/L3 rules, which is much smaller compared to the tens of thousands of flows that an SDN-enabled switch can concurrently forward [7, 18].Therefore, installing flow entries proactively needs to be done in an optimized way.
- Reactive flow management: The controller installs the flow table entries when the flow arrives and switch has no entry to entertain. Hence the switch forwards the first packet or the header to the controller. The controller identifies the path for the packet and installs appropriate rules in all switches along the path. Then the packets of that flow can be forwarded to their destination. One drawback of this model is that it can induce significant latency which is very detrimental to the flows [16].

To solve the problem of optimization in the proactive approach, we are using Machine Learning to predict flows and then proactively install flow rules. We attempted to predict future flows using flow popularity approach. Flow popularity of IP source/destination (src/dst) pair is the function of the number of flows of that pair. We assumed that we know the flow rule for particular src/dst pair in most of the cases because mostly flow rule depends on type of application which can be inferred by destination IP. For example, if pair's destination is HTTP server and we know where to route HTTP traffic, then we know the flow rule to be installed for this kind of pair.

We want to therefore predict the popularity of IP source/destination pairs to find out the most frequent or most popular pairs. Rule installation of most popular pairs can save huge latency cost by saving several signalling trips to the controller from a switch. We want to predict popularity based on the first few seconds of pair's behavior or features. We have used the features of 10, 20, 30, 60 and 90 seconds. By telling the popularity based on initial seconds, we

, ,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00

can know beforehand that which pairs will remain frequent or will increase its frequency in future.

We have used CAIDA 2019 Data center dataset [1] because SDN is mostly implemented on large Data centers [13]. We used the supervised learning approach of Machine Learning. We have tried N = 1,2 and 3. We first established the ground truth of popularity on the scale of 1 to N with 1 being the least popular and N being the most popular. We have tried N = 1,2 and 3. We gave these labels based on the number of flows of a pair in X minutes where X = 1,4. We want to predict popularity of X minutes based on initial behavior or features therefore we extracted features of first 10, 20, 30, 60 and 90 seconds of those pairs. We used 300 different Machine Learning models which include 13 classifiers like Support Vector Machine (SVM), Decision Trees, Multi Layer Perceptron (MLP) and Logistic Regression for prediction of popularity. We gained accuracy upto 74 % using SVM. Later, we redefined the ground truth of popularity using different data labeling techniques (equal size buckets, equal frequency range buckets, threshold based buckets) and we also used over-sampling techniques like SMOTE in order to cater imbalanced dataset. In this way, we got interesting results of accuracies up to 98 %. We also observed the tradeoff between recall and precision in our results and out of all metrics, recall is a better metric for imbalanced data and we got up to 96 % recall. Since a time-series based approach might be more suitable to this scenario instead of a classification approach, we are now exploring an approach involving LSTM and GRU, where we analyze the flows of each pair separately to predict when the next one will come. We are learning a more regression-based model that will give the predicted number of flows on the (n+1)th time-stamp using data from the previous n time-stamps. We have replicated our results thus far on another dataset, the IMC 2010 Data Center Measurement Univ1 traces to strengthen our results and also further smoothen our data preprocessing pipeline.

2 MOTIVATION

There are number of challenges faced by the controller due to which we cannot rely on the reactive approach. These challenges are our motivation to optimize proactive flow rule installation using Machine Learning.

Following are the problems which our approach intends to solve:

- A controller could potentially be a bottleneck in data-control plane communication if controller has to install flow rules for every unknown flow.
- Controller can be vulnerable to control plane saturation attack[4].
- Data plane to control plane and vice versa communication can incur significant latency and some short-term latency-sensitive flows can adversely suffer due to such delays.
- Controllers are difficult to scale.
- Vulnerability to other attacks such as DOS[20].
- Predicting future traffic can help in pre-route planning and efficient Quality of Service provisioning.

3 RELATED WORK

One category of papers we looked at was those that simply model network traffic trends with machine learning to see what particular models work best on this type of data [8, 14]. We also use these to find how these projects dealt with the inherent burst-y quality of the data, which brought us to the conclusion that the solution we want to work on may not be applicable on the higher up levels of the network topology, due to the volume of data. The main set of work we looked into was one that described and/or proposed solutions to the problem of rule placement in SDNs, primarily with the use of machine learning. There have been a few such approaches in previous works. In the case of unsupervised learning, [11] has set up a reinforcement learning algorithm. Supervised learning implementations [3, 5, 9, 15, 17, 19] range from models like Bayesian to SVMs and MLPs, but they aim to predict the actual flow rule in most cases, or differentiate between elephant and mice flows. To our knowledge, the technique of predicting the flow popularity in a future time to leverage for flow rule installation has not been implemented before. Furthermore, the majority of these works have used generated traffic data for their model training and simulations for their evaluation steps.

4 DATASET

We used The CAIDA Anonymized Internet Traces 2019 Dataset. This dataset contains anonymized passive traffic traces from CAIDA's passive monitors in 2019. It contains traffic traces from the 'equinix-nyc' high-speed monitor collected over one hour. The equinix-nyc Internet data collection monitor is located at an Equinix datacenter in New York, NY, and is connected to an OC192 backbone link (9953 Mbps) of a Tier1 ISP between New York, NY and Sao Paulo, Brazil.

Total IPv4 flows in this data are 2180813, and this data only has their header information. In order to study the dataset, we analyzed the first 50 seconds of it. Using headers, we extracted application layer protocols based on ports and transport layer protocol based on header's protocol field.

Distribution of transport and application layer protocols in DataSet is shown in Figure 1.

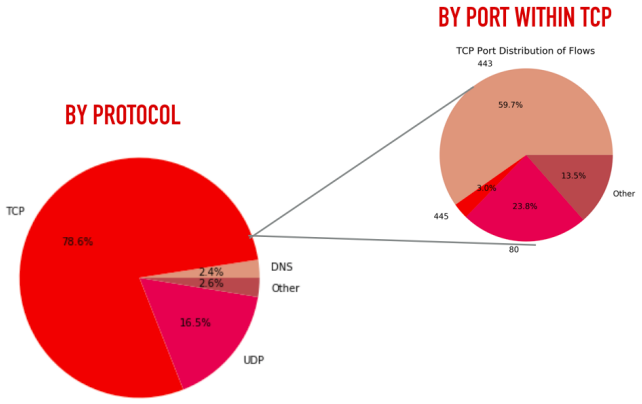

Figure 1: Distribution of transport and application layer protocols

We identified each flow by TCP three way handshake (1- SYN, 2- SYN+ACK, 3- ACK) and FIN packets. Then, we calculated number of flows per second that is frequency of flows for not only all applications but also the applications like HTTP (identified by port

80) and HTTPS (identified by port 443) as shown in Figures 2,3 and 4. We also visualized the noise in each of the above applications by creating flows per 100 milliseconds graphs as shown in Figure 5.

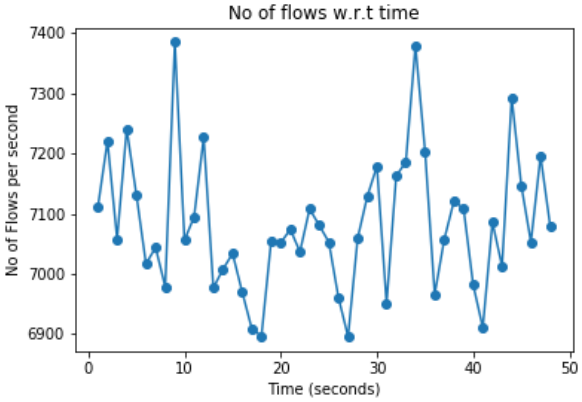

Figure 2: Flows per second in initial 50 seconds data

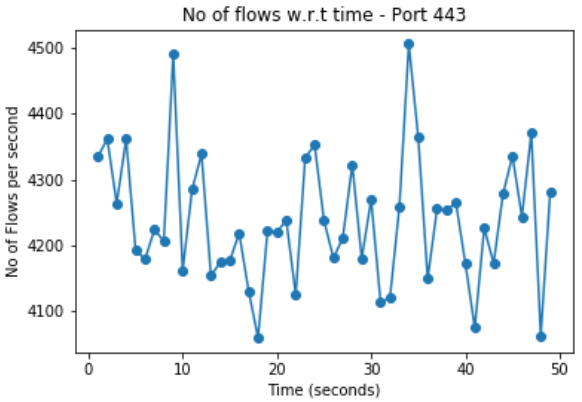

Figure 3: Flows per second of HTTPS traffic in initial 50 seconds data

The data preprocessing pipeline was prepared using CAIDA Dataset. Major part of analysis was also conducted on CAIDA 2019 Dataset. But, we also used the university data center UNI1 dataset given by Wisconsin and IMC conference [2, 6] for the purpose of smoothing our data preprocessing pipeline and also strengthen our results. There are 500 servers and 22 devices in this dataset. It is a one minute long packet trace pcap file and it uses a topology that is similar to a canonical 2-Tier architecture.

5 PROBLEM FORMULATION

We also have seen the distributions of number of flows for each unique IP source/destination pairs in initial 50 seconds data. 0.05% of flows that is 122/231273 unique pairs have greater than 50 flows in 50 seconds which means on average more than 1 flow per second. We can see the distribution of number of flows for these 122 pairs

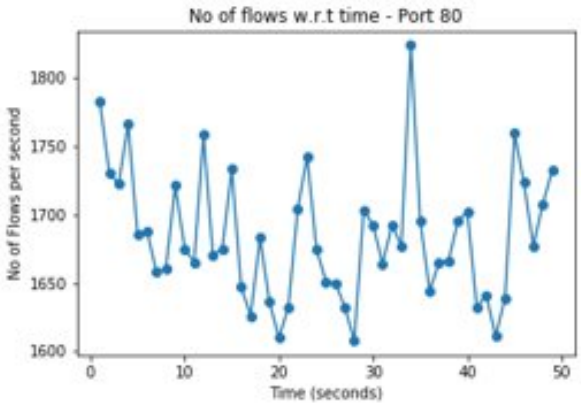

Figure 4: Flows per second of HTTP traffic in initial 50 seconds data

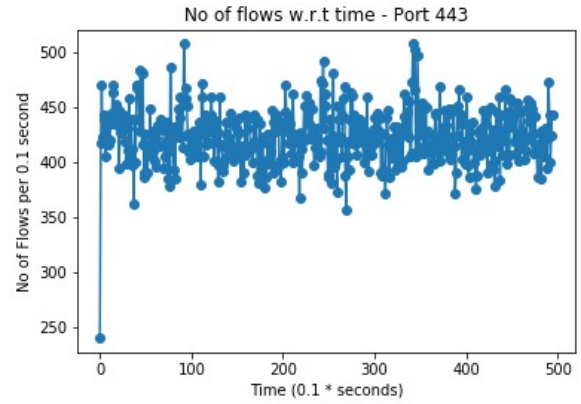

Figure 5: Flows per 100 milliseconds in 50 seconds - visualising noise

in Figure 6. Median number of flows for these 122 pairs are around 100.

If there are 122 unique pairs have on average 100 flows per 50 seconds then it means there are 120 flows per minute for each pair on average. If we rightly predict those pairs based on first 10 seconds of their header features, we can save 122*120 = 14640 signaling trips to controller per minute, considering there no flow rules in switch for those pairs.

To save these signaling trips, our main problem statement in this paper is to predict the popularity (a function of pair's flows frequency) of pairs based on its initial seconds features. This will help us identify the most popular pairs and assuming we have flow rules for pairs, we can install the flow rules of most popular pairs.

6 DATA EXTRACTION APPROACH AND ITS CHALLENGES

We attempted to solve our problem through a supervised learning approach and considered it firstly as a classification problem. We

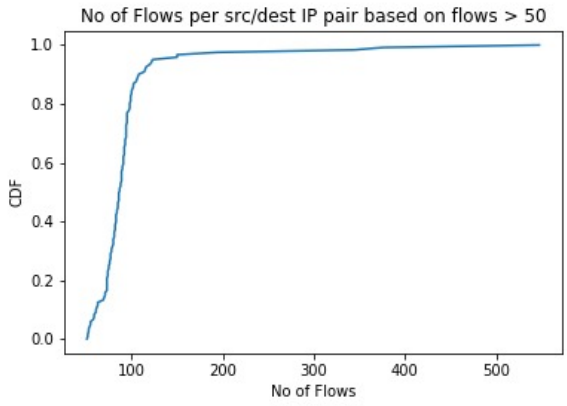

Figure 6: Distribution of number of flows of IP src/dest pairs having flows > 50

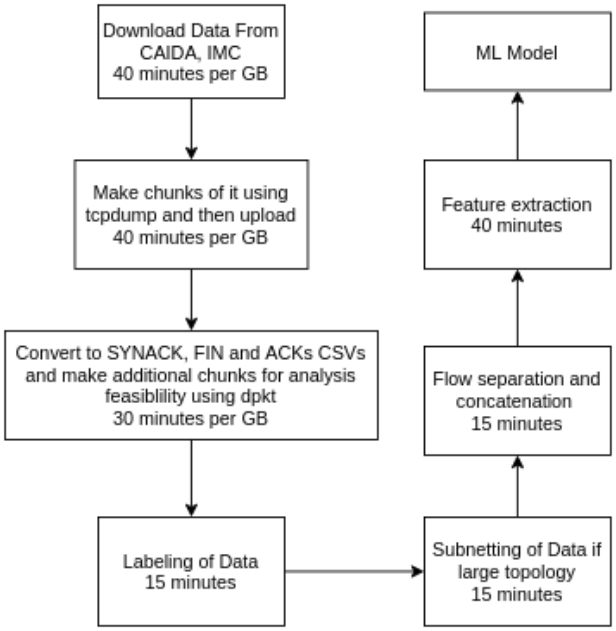

Figure 7: Preprocessing Pipeline

are also currently exploring the timeseries based dynamic/runtime approach. The preprocessing pipeline is shown in Fig 7. Currently, it takes us 2-3 hours to preprocess per GB of pcap file.

6.1 Packets Extraction to CSV Format

We needed ACK, SYN+ACK and FIN packets in CSVs to identify flows and store features. We needed ACK packets for features like number of bytes of each flow and to identify the start of flow as third step in three-way handshake is an ACK. As ACK file contains packets of all flows, it incurred huge time cost in extracting features. Therefore, We needed SYN+ACK and FIN packets separately to extract many features that depend on them so that time cost due to ACK packets can be mitigated.

We leverage the python dpkt library to convert CAIDA dataset pcap files to csv files. Then we extracted all the ACK, SYN+ACK and FIN packets of 4 minutes dataset separately using dpkt. Each trace of 1 minute was of 2 GB and it was a challenging task to extract it to CSVs on our computers. Before using dpkt, we were using Wireshark and tcpdump to convert pcap files to csv files which was really time consuming. We significantly reduced the overall time of data preprocessing by using dpkt in place of Wireshark.

6.2 Subset of Dataset by Considering Subnet or Prefix

There are 21,80,813 IPV4 flows inside CAIDA dataset which are too large for analysis. Number of flows per second on average are too large that is around 7000 as shown in Figure 2. These issues make it a difficult problem to extract features of all the flows. To solve these challenges, instead of considering the whole topology of New York, we based our solution on a subset of dataset by considering only one topology. We first calculated the flows of each /24 prefix and choose one of them. We considered only the 199.124.196.0/24 prefix from CAIDA dataset making ultimately a total of 9694 pairs. To extract subset of large dataset, we needed GPU. We didnt have our personal GPU therefore we used GPU of google colab. But, we uploaded around 30 GB of data in CSV format to colab which was a time consuming task too. Developing the script for such large data was also a challenging task. There is no need of considering subnet in datasets like UNIV1 dataset as that is a small dataset. Currently, we are also trying to use the 60 minutes dataset and whole topology (all subnets) so that our results show more representation for enterprise data centers like Equinix.

6.3 Grouping of Packets of each pair

One of the challenges we faced is that there were a lot of packets (GBs of Data) that our feature extraction script took hours for execution. We therefore broke down the task by doing a sequence of steps. We grouped all the packets of each pair together sorted by time so that overall complexity of extracting each feature can become low. We then take only few seconds of packets of each pair to ultimately get packets on which we could extract features. This grouping script took us huge time to develop as we tried different approaches like parallel programming, vectorization approaches.

7 APPROACH FOR CLASSIFICATION

7.1 Ground Truth

Popularity of a pair is based on frequency of pairs in **X** (say this parameter as **Label_Time**) minutes of data. We have used Label_Time = 1 minute and 4 minutes in our analysis. There are several ways to define the popularity of a pair based on number of buckets (**N**) and data labeling techniques (equal size buckets, equal frequency range buckets, threshold based buckets). Both are explained below in detail.

7.1.1 ***Number of Buckets:*** We have formulated a N number of buckets problem in which the popularity of pair is the number between 0 and N such that 0 being the least popular and N being

the most popular. We have tried N = 1,2 and 3. When N = 1, it becomes a two bucket problem or a binary problem in which we are labeling a given pair as 0 or 1. 0 means less popular and 1 means more popular. When N = 2, it becomes a three buckets problem in which we are labeling a given pair as 0 or 1 or 2. 0 means not popular, 2 means moderately popular and 1 means more popular. Similarly, we tried N = 3 as well. We intend to try more buckets in future.

7.1.2 ***Data labeling techniques:*** The three major data labeling techniques: equal size buckets, equal frequency range buckets and threshold based buckets are explained below. But we have used other techniques as well to further explore different and better ways of labeling.

- **Equal Size Buckets:** We first calculated the frequency of each pair and sort them. Then we divide the number of pairs into N equal size intervals/buckets. Popularity of the pair is the bucket number in which the frequency of that pair lies.
- **Equal Frequency range buckets:** It is the labeling of data based on equal buckets of frequency range. For instance, in 60 seconds of data, we first calculate the highest possible frequency and lowest possible frequency so that we will find the range of frequency. Then we divide the range in N number of buckets and popularity of the pair is the bucket number in which the frequency of that pair lies.
- **Threshold based buckets:** We have used thresholds to define the popularity of a pair for all of the two, three and four bucket problems. For instance, if we consider a two bucket problem and use 60 seconds (Label_Time = 1) of data to calculate frequency of a pair then we use a single threshold value 'thresh' such that any pair having frequency less than thresh is assigned 0 label and any pair having frequency greater than thresh is assigned 1 as a label. Similarly, we can choose 2 thresholds for 3 buckets and 3 thresholds for 4 buckets problem and assign labels accordingly.

7.1.3 ***Comparison of Data labeling techniques:*** Equal size buckets technique is not optimal because there are many pairs which come only once. It is quite often the case that two buckets have different labels and all of the pairs in those buckets have same or comparable frequencies. For example, in a four bucket problem, two of the buckets have only those pairs which come only once that is their frequency is 1. But due to equal size of buckets, two buckets having similar frequent pairs are given different labels. We have also tuned it later to give such buckets same label but this was not the best approach for the ground truth.

We needed a more practical approach in which we could incorporate frequency such that above issue would not arise. So, we tried equal frequency range buckets technique. It was a relatively better technique but it is still not optimal because most of the pairs get the same label because outliers (highly frequent ones) make range too high. So, we had to neglect outliers while calculating range and then we assigned label accordingly.

The most suitable technique is threshold based buckets because that way we can actually define the pair as mildly popular or moderately popular or most popular based on frequency thresholds.

Out of all techniques, our results would be more focused on threshold based buckets technique because it was the best possible way of ground truth. We have extensively used the other two techniques as well but we have added few results from those techniques.

7.2 Feature Extraction

After above data extraction steps, it was possible for us to extract features in linear time. After extensively reading up on features suitable for network traffic prediction [10, 12] We used the following features:

- Mean Inter arrival time b/w flow
- Standard Deviation of Inter arrival time b/w flow
- Minimum of Inter arrival time b/w flow
- Maximum of Inter arrival time b/w flow
- 1st Quartile of Inter arrival time b/w flow
- 2nd Quartile of Inter arrival time b/w flow
- 3rd Quartile of Inter arrival time b/w flow
- Inter-quartile Range of Inter arrival time b/w flow
- Fast fourier transform of Inter arrival time b/w flow
- Mean Inter arrival time b/w packets
- Standard Deviation of Inter arrival time b/w packets
- Minimum of Inter arrival time b/w packets
- Maximum of Inter arrival time b/w packets
- 1st Quartile of Inter arrival time b/w packets
- 2nd Quartile of Inter arrival time b/w packets
- 3rd Quartile of Inter arrival time b/w packets
- Inter-quartile Range of Inter arrival time b/w packets
- Fast fourier transform of mean Inter arrival time b/w packets
- frequency of flow
- Fast fourier transform of frequency of flow
- Mean of Byte size of flow
- Standard Deviation of Byte Size of flow
- Packet Count
- Application Layer Protocol (Port)

7.3 Classifiers and Feature engineering

We used 300 different models after tuning parameters of state of the art 13 classifiers to try to achieve best possible accuracy. We also scaled our features using Standard Scaler. We have checked correlations between different features and try to exclude highly correlated features in order to include independent features. Also, We tried each possible subset of features. Out of all classifiers as shown in Table 1, Gaussian SVM outperformed and therefore we used it for further analysis.

7.4 Imbalanced Data Issue:

We have observed that around 400 pairs out of 4000 have greater than 60 flows per minute. So, there are very less data points of frequent pairs (around 10%) which we intend to identify through our machine learning model. To increase the data points of frequent pairs and solve the issue of imbalanced data, We used oversampling techniques which include SMOTE, Random Oversampler, SVMSMOTE, KMeansSMOTE, SMOTETomek etc. Out of all oversampling techniques, we found SMOTE to be most effective and therefore we used it extensively in our further analysis. We could

Table 1: 13 Classifiers

Classifiers
XGB Classifier
Random Forest Classifier
Gaussian SVM
Decision Tree Classifier
Logistic Regression
MLP
Gradient Boosting Classifier
Extra Trees Classifier
Linear SVC
One Hot Encoder
K-Neighbours Classifier
Bernoulli NB
SGD Classifier

use undersampling of non-frequent pairs but that would be equivalent to losing information of non-frequent pairs which was not good for our training model.

7.5 Time Frame of labels and features issue:

We initially labeled the popularity of pair based on 4 minutes (say this parameter as **Label_Time**) and then predicted it based on feature sets collected on initial **T** seconds where T = 10, 20, 30, 60 and 90. But this approach is not optimal for dynamic traffic for instance if T = 20 seconds which means we have collected features of only 20 seconds and label was of 4 minutes then some pairs will be frequent in initial 20 seconds and some will be frequent after 20 seconds. So, this approach will not predict rightly and was not practical. So, we decided to reduce Label_Time from 4 minutes to 1 minute and used T = 20 seconds for further analysis. This makes more sense than initial approach because we do not need to predict the popularity of pair in long time frame of 4 minutes. Rather we need a dynamic model which knows the popularity of pair in short time frame like in the case of Label_Time = 60 seconds and T = 20 seconds. We are also exploring the approach in which we will consider Label_Time = T + 1 so that we can install the flow rule for that pair right away. We are in a need of more dynamic model but most of our results will be of Label_Time = 60 seconds and T = 20 seconds which is also a good approach to classify pairs.

7.6 Preferred Metric for Evaluation:

Recall is the preferred metric in this case. We actually need to identify popular pairs which means we need lowest possible false negatives which is what high recall will give to us. High recall means low false negatives and more true positives. While achieving lowest possible false negatives, we will encounter more false positives that is it will predict unpopular ones to popular ones. By debugging, it came out that false positives were also moderately popular. So, high false positives will not effect our model but high false negatives will effect our model as we will wrongly predict popular pairs. Therefore, we prefer recall as our preferred metric for evaluation.

As far as accuracy is concerned, it will not be a good metric because our data is imbalanced and we have more data points of unpopular pairs. In this way, we can easily achieve 95% accuracy by rightly predicting all unpopular ones.

Precision is not a good metric because high precision means low false positives which in turn will give high false negatives and hence will not identify maximum possible popular pairs.

However, we cannot ignore precision and accuracy while achieving high recall because we observed a trade-off between precision and recall when we tuned our parameters such as threshold of ground truth. Therefore, high recall, moderate precision and high accuracy would be an optimal metric to evaluate our results.

7.7 Parameters Tuning Summary

Before discussing the results, let us summarize the parameters or things we discussed earlier and were tuned or preferred to obtain results.

- Dataset: We used two types of datasets namely CAIDA 2019 and UNIV1 2010 IMC Dataset but CAIDA 2019 was preferred.
- No_of_Buckets (B): We used 2,3 and 4 buckets for our results.
- Feature_Time(T): Feature_Time is the seconds/minutes on which feature set was prepared. We used 20 seconds feature set more often. But we explored T = 10,30,60 and 90 as well.
- Label_Time: We used Label_Time = 4 minutes in the start but we used Label_Time = 1 minute more often.
- Oversampling techniques: SMOTE was used as it outperformed.
- Machine Learning models: SVM was used as it outperformed.
- Data labeling techniques: Threshold based labeling was frequently used out of all techniques

8 RESULTS

8.1 Preliminary Results

In these results, **Feature_Time(T)** = 10,20,30,60,90 seconds and **Label_Time** = 4 minutes. We used SVM using Gaussian Kernel as it outperformed out of all machine learning models. We used 8 features for these results. We used equal size 4 buckets to assign labels. This was not a good approach of ground truth therefore we did not use it for further efforts. But this became basis of further results because of high recall and precision of bucket number 4 as observed in the last row of Table 2. By using 90 seconds feature set and labels on 4 minutes, we gained bucket 4 recall 79 % with 91 % precision. We are seeing statistics of only bucket number 4 (most popular ones) here because other 3 buckets had comparable frequencies which was the issue due to usage of equal size buckets. Accuracy is not important to see here because ground truth was not established accurately. However, results of accuracies are visible in Figure 13. As we increase the seconds of feature set, accuracy and recall of predicting popularity increases.

These results were unsatisfactory but high recall and precision of bucket 4 are the basis of further good results. Following are the feature sets and their results in figures.

- 10 Seconds Feature Set: shown in Figure 8.
- 20 Seconds Feature Set : shown in Figure 9.
- 30 Seconds Feature Set: shown in Figure 10.
- 60 Seconds Feature Set: shown in Figure 11.
- 90 Seconds Feature Set: shown in Figure 12.

	precision	recall	f1-score
1	0.41	0.99	0.58
2	0.51	0.41	0.45
3	0.59	0.30	0.40
4	0.78	0.35	0.48
accuracy			0.50
macro avg	0.57	0.51	0.48
weighted avg	0.58	0.50	0.48

Figure 8: Preliminary Results: Summary of SVM using all features and gaussian kernel on features of 10 seconds

	precision	recall	f1-score
1	0.49	0.98	0.65
2	0.56	0.46	0.50
3	0.65	0.35	0.45
4	0.75	0.49	0.59
accuracy			0.57
macro avg	0.61	0.57	0.55
weighted avg	0.61	0.57	0.55

Figure 9: Preliminary Results: Summary of SVM using Gaussian kernel on features of 20 seconds

	precision	recall	f1-score
1	0.53	0.99	0.69
2	0.59	0.41	0.48
3	0.61	0.39	0.47
4	0.77	0.60	0.67
accuracy			0.60
macro avg	0.62	0.60	0.58
weighted avg	0.62	0.60	0.58

Figure 10: Preliminary Results: Summary of SVM using Gaussian kernel on features of 30 seconds

	precision	recall	f1-score
1	0.62	1.00	0.76
2	0.63	0.48	0.54
3	0.68	0.51	0.58
4	0.86	0.70	0.77
accuracy			0.68
macro avg	0.70	0.67	0.67
weighted avg	0.70	0.68	0.67

Figure 11: Preliminary Results: Summary of SVM using Gaussian kernel on features of 60 seconds

Table 2: Results for the feature sets of 10,20,30,60 and 90 and labels of 4 minutes

Feature_Time(T)	Precision of bucket 4	Recall of bucket 4
10	78	35
20	75	49
30	77	60
60	86	70
90	91	79

	precision	recall	f1-score
1	0.66	1.00	0.80
2	0.74	0.53	0.62
3	0.72	0.67	0.69
4	0.91	0.79	0.85
accuracy			0.74
macro avg	0.76	0.75	0.74
weighted avg	0.76	0.74	0.74

Figure 12: Preliminary Results: Summary of SVM using Gaussian kernel on features of 90 seconds

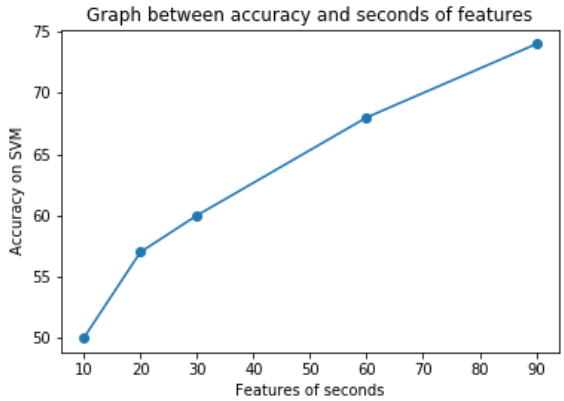

Figure 13: Preliminary Results: Summary of SVM accuracy when we change time on which features are collected

8.2 Metric for further results

We are not analyzing our results based on the recall and precision of unpopular ones because we are getting these metrics more than 90 % in almost all the cases. We are not also analyzing our results on the average of recalls and precisions of both popular and unpopular ones because that would not reflect how much popular ones were identified correctly. We are also not seeing accuracy for analysis of our results because it is more than 90 % in almost all the cases.

8.3 University Data Results

This dataset was of 60 seconds so we chose Label_Time = 60 seconds and Feature_Time = 20 seconds. We have considered a two bucket

	precision	recall	f1-score
0	0.93	0.99	0.96
1	0.91	0.43	0.58
accuracy			0.93
macro avg	0.92	0.71	0.77
weighted avg	0.93	0.93	0.92

Figure 14: Without SMOTE Univ1 Dataset Results

	precision	recall	f1-score
0	0.97	0.91	0.94
1	0.43	0.71	0.54
accuracy			0.89
macro avg	0.70	0.81	0.74
weighted avg	0.92	0.89	0.90

Figure 15: SMOTE Univ1 Dataset Results with threshold = 60

	precision	recall	f1-score
0	0.95	0.91	0.93
1	0.54	0.67	0.60
accuracy			0.88
macro avg	0.74	0.79	0.76
weighted avg	0.90	0.88	0.89

Figure 16: SMOTE Univ1 Dataset Results with threshold = 30

problem in this dataset and used many data labeling techniques. We have used oversampling technique of SMOTE. We used SVM in it.

- Without SMOTE, precision for bucket number 2 (popular pairs) is high (91 %) and recall is low as shown in Figure 14.
- With SMOTE and threshold = 60 seconds, precision for popular pairs is 43 %, recall for popular pairs is 71 % as shown in Figure 15
- With SMOTE and threshold = 30 seconds, precision for popular pairs is 54 %, recall for popular pairs is 67 % as shown in Figure 16
- With SMOTE and equal size two buckets, average precision is 75 % and average recall is 76 % as shown in Figure 17.

8.4 CAIDA 2019 Data Results

In these results, we chose Label_Time = 60 seconds and Feature_Time = 20 seconds. We have used oversampling technique of SMOTE. We used SVM in it. We have considered a two, three and four bucket problem in this dataset based on calculated thresholds and used many data labeling techniques. We have checked many thresholds =

	precision	recall	f1-score
0	0.64	0.81	0.71
1	0.86	0.72	0.78
accuracy			0.75
macro avg	0.75	0.76	0.75
weighted avg	0.77	0.75	0.75

Figure 17: SMOTE Univ1 Dataset Results with equal size two buckets

	precision	recall	f1-score
0	0.95	0.90	0.92
1	0.64	0.80	0.71
accuracy			0.88
macro avg	0.80	0.85	0.82
weighted avg	0.89	0.88	0.89

Figure 18: CAIDA Dataset, two bucket problem and threshold = 5 seconds

5,10,15,20...60 and have quoted here the best results for two bucket problems. Four bucket and three bucket give best results when the thresholds are calculated. We used those thresholds in three bucket which outperformed in two bucket and similarly, those in four bucket which outperformed in three bucket. For three bucket and four bucket, we are considering average of recall and average of precision as best metric because all buckets has comparable values.

8.4.1 ***Two Buckets.*** We are using recall and precision for the popular ones for evaluation because recall and precision of unpopular were higher than 90 %.

- Threshold 5 sec: 80 % recall, 64 % precision, 88 % accuracy as shown in Figure 18.
- Threshold 10 sec: 84 % recall, 49 % precision, 92 % accuracy as shown in Figure 19
- Threshold 30 sec: 81 % recall, 40 % precision, 97 % accuracy as shown in Figure 20
- Threshold 40 sec: 89 % recall, 32 % precision, 98 % accuracy as shown in Figure 21.

8.4.2 ***Three Buckets.*** We used threshold of 10 seconds and 40 seconds in three buckets. Average of recall and average of precision would be a best metric here because all buckets has comparable values as shown in Figure 22. Results: 84 % recall, 70 % precision, 91 % accuracy .

8.4.3 ***Four Buckets.***

- Theshold 10, 40, 60: 71 % precision, 77 % recall, 92 % accuracy as shown in Figure 23.
- Threshold 5,10, 40: 72 % recall, 76 % precision, 88 % accuracy as shown in Figure 24.

	precision	recall	f1-score
0	0.99	0.93	0.96
1	0.49	0.84	0.62
accuracy			0.92
macro avg	0.74	0.89	0.79
weighted avg	0.95	0.92	0.93

Figure 19: CAIDA Dataset, two bucket problem and threshold = 10 seconds

	precision	recall	f1-score
0	1.00	0.98	0.99
1	0.40	0.81	0.53
accuracy			0.97
macro avg	0.70	0.89	0.76
weighted avg	0.98	0.97	0.98

Figure 20: CAIDA Dataset, two bucket problem and threshold = 30 seconds

	precision	recall	f1-score
0	1.00	0.98	0.99
1	0.32	0.89	0.47
accuracy			0.98
macro avg	0.66	0.93	0.73
weighted avg	0.99	0.98	0.98

Figure 21: CAIDA Dataset, two bucket problem and threshold = 40 seconds

	precision	recall	f1-score	s
0	0.99	0.92	0.95	
1	0.38	0.80	0.52	
2	0.74	0.81	0.78	
accuracy			0.91	
macro avg	0.70	0.84	0.75	
weighted avg	0.95	0.91	0.92	

Figure 22: CAIDA Dataset, three bucket problem and threshold = 10, 40 seconds

9 A FUTURE APPROACH

Our approach so far has focused on the classification task of predicting the popularity of a certain IP pair, but we are now also exploring a time-series analysis of the data for flow prediction, as another angle to this problem. After analysing the paper [14] we have noticed

	precision	recall	f1-score
0	0.99	0.93	0.96
1	0.42	0.83	0.56
2	0.43	0.64	0.51
3	1.00	0.67	0.80
accuracy			0.92
macro avg	0.71	0.77	0.71
weighted avg	0.95	0.92	0.93

Figure 23: CAIDA Dataset, four bucket problem and threshold = 10,40,60 seconds

	precision	recall	f1-score
0	0.95	0.92	0.94
1	0.47	0.61	0.53
2	0.85	0.72	0.78
3	0.77	0.63	0.69
accuracy			0.88
macro avg	0.76	0.72	0.74
weighted avg	0.89	0.88	0.88

Figure 24: CAIDA Dataset, four bucket problem and threshold = 5,10,40 seconds

that one of the problems they address, of traffic volume prediction, is analogous to what we could call a flow volume problem ie, the prediction of how many flows of a particular IP pair would show up at a certain time-stamp, taking into account past data. It is then not a far-off assumption to make that we could follow their approach exactly, which is what we are doing currently with both the IMC and the CAIDA datasets. What is novel about their approach is that they use RNNs to perform accurate predictions on the time-series network traffic data, and the results obtained are better off than simple Neural Networks. We are hoping this would be mirrored in our formulation of this problem as well.

REFERENCES

[1] [n.d.]. https://data.caida.org/datasets/passive-2019/

[2] [n.d.]. http://pages.cs.wisc.edu/~tbenson/IMC10_Data.html

[3] Pedro Amaral, Joao Dinis, Paulo Pinto, Luis Bernardo, Joao Tavares, and Henrique S. Mamede. 2016. Machine Learning in Software Defined Networks: Data collection and traffic classification. *2016 IEEE 24th International Conference on Network Protocols (ICNP)* (2016). https://doi.org/10.1109/icnp.2016.7785327

[4] Moreno Ambrosin, Mauro Conti, Fabio De Gaspari, and Radha Poovendran. 2017. LineSwitch: Tackling Control Plane Saturation Attacks in Software-Defined Networking. *IEEE/ACM Transactions on Networking* 25, 2 (2017), 1206–1219. https://doi.org/10.1109/tnet.2016.2626287

[5] Abdullah Baz. 2018. Bayesian Machine Learning Algorithm for Flow Prediction in SDN Switches. *2018 1st International Conference on Computer Applications Information Security (ICCAIS)* (2018). https://doi.org/10.1109/cais.2018.8441969

[6] Theophilus Benson, Aditya Akella, and David A Maltz. 2010. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. 267–280.

[7] Rajesh Challa, Yongseung Lee, and Hyunseung Choo. 2016. Intelligent eviction strategy for efficient flow table management in OpenFlow Switches. *2016 IEEE*

NetSoft Conference and Workshops (NetSoft) (2016). https://doi.org/10.1109/netsoft.2016.7502427

[8] Yuehui Chen, Bin Yang, and Qingfang Meng. 2012. Small-time scale network traffic prediction based on flexible neural tree. *Applied Soft Computing* 12, 1 (2012), 274–279.

[9] Aysse Rumeysa Mohammed, Shady A. Mohammed, and Shervin Shirmohammadi. 2019. Machine Learning and Deep Learning Based Traffic Classification and Prediction in Software Defined Networking. *2019 IEEE International Symposium on Measurements Networking (MN)* (2019). https://doi.org/10.1109/iwmn.2019.8805044

[10] AysŞe Rumeysa Mohammed, Shady A Mohammed, and Shervin Shirmohammadi. 2019. Machine Learning and Deep Learning Based Traffic Classification and Prediction in Software Defined Networking. In *2019 IEEE International Symposium on Measurements & Networking (M&N)*. IEEE, 1–6.

[11] Ting-Yu Mu, Ala Al-Fuqaha, Khaled Shuaib, Farag M. Sallabi, and Junaid Qadir. 2018. SDN Flow Entry Management Using Reinforcement Learning. *ACM Transactions on Autonomous and Adaptive Systems* 13, 2 (2018), 1–23. https://doi.org/10.1145/3281032

[12] Thuy T.t. Nguyen and Grenville Armitage. 2008. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys Tutorials* 10, 4 (2008), 56–76. https://doi.org/10.1109/surv.2008.080406

[13] Mohammad Noormohammadpour and Cauligi S. Raghavendra. 2018. Datacenter Traffic Control: Understanding Techniques and Tradeoffs. *IEEE Communications Surveys Tutorials* 20, 2 (2018), 1492–1525. https://doi.org/10.1109/comst.2017.2782753

[14] Nipun Ramakrishnan and Tarun Soni. 2018. Network Traffic Prediction Using Recurrent Neural Networks. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 187–193.

[15] Mohammad Reza, Mohammad Javad, Seyed Raouf, and Reza Javidan. 2017. Network Traffic Classification using Machine Learning Techniques over Software Defined Networks. *International Journal of Advanced Computer Science and Applications* 8, 7 (2017). https://doi.org/10.14569/ijacsa.2017.080729

[16] Walber Silva. 2017. Performance Evaluation of Flow Creation Inside an OpenFlow Network. *Anais de XXXV Simpósio Brasileiro de Telecomunicações e Processamento de Sinais* (2017). https://doi.org/10.14209/sbrt.2017.269

[17] Sebastian Troia, Alberto Rodriguez, Ignacio Martin, Jose Alberto Hernandez, Oscar Gonzalez De Dios, Rodolfo Alvizu, Francesco Musumeci, and Guido Maier. 2018. Machine-Learning-Assisted Routing in SDN-Based Optical Networks. *2018 European Conference on Optical Communication (ECOC)* (2018). https://doi.org/10.1109/ecoc.2018.8535437

[18] Anilkumar Vishnoi, Rishabh Poddar, Vijay Mann, and Suparna Bhattacharya. 2014. Effective switch memory management in OpenFlow networks. *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems - DEBS 14* (2014). https://doi.org/10.1145/2611286.2611301

[19] Pan Wang, Feng Ye, Xuejiao Chen, and Yi Qian. 2018. Datanet: Deep Learning Based Encrypted Network Traffic Classification in SDN Home Gateway. *IEEE Access* 6 (2018), 55380–55391. https://doi.org/10.1109/access.2018.2872430

[20] Qiao Yan and F. Richard Yu. 2015. Distributed denial of service attacks in software-defined networking with cloud computing. *IEEE Communications Magazine* 53, 4 (2015), 52–59. https://doi.org/10.1109/mcom.2015.7081075