Brief Summary of Transfer Learning/Classification Experiment Approaches

Byron Barkhuizen

This document is a summary of what I have explored so far in the transfer learning and classification area for the problem of classifying smart device network traffic and ongoing approaches. It mostly serves to provide explanations and results for the files currently published to the project's github page (https://github.com/byronbark/IOTProject)

PCAP to CSV

- The conversion of PCAP to a CSV file is done with the java tool CICFlowmeter: (https://github.com/ahlashkari/CICFlowMeter/tree/master/src/main)

This tool generates 84 flow level statistics directly from a PCAP file after it has been split by MAC address with the SplitCap tool.

- Some Python tools Scapy and pyshark exist that are able to modify pcap files directly and generate csv files. A method for this is being worked on within the github project under 'pcap2csv.py'. It would require less manual work than using CICFlowmeter and can be customized to generate identical csv files. For the moment it is not required so CICFlowmeter is being used.
- Mainly flow level information is being used because other features such as mac address, port number can be spoofed and become unreliable.

PCAP to Image

- PCAP to image transformation is being done with the series of powershell scripts and python files found in '/PCAP Manipulation' to create MNIST style grayscale images.
- Images are easier to perform transfer learning on than CSV files as less feature engineering would be required.

One class SVM Classifier

- A one class classifier python notebook example can be found under '/OCSVM' and it uses the CSV files to generate panda dataframes to perform a one class classification on. 'Home' data is labelled as 1 and anything else is labelled -1.
- The source google home data is used labelled 1 and an assortment of different MAC address' PCAPs are used to fill the dataframe with data labelled -1 (not 'home').
- The model is able to classify source domain data points quite well at around 90% accuracy. The results are summarized in the picture below from the .ipynb file.

```
##Evaluate on unseen test data
preds = model.predict(test_data)
targs = test_target
print("accuracy: ", metrics.accuracy_score(targs, preds))
print("precision: ", metrics.precision_score(targs, preds))
print("recall: ", metrics.recall_score(targs, preds))
print("f1: ", metrics.f1_score(targs, preds))
print("area under curve (auc): ", metrics.roc_auc_score(targs, preds))
```

```
accuracy:  0.9147031102733271
precision:  0.9636996519144704
recall:  0.9472140762463344
f1:  0.9553857530194725
area under curve (auc):  0.49334388022843034
```

- When the same model without any adjustment is tested on completely new data in a target doman (In this case it's Mus' Google Home Mini data) the accuracy falls to 30%. With the assumption that the distribution of features is different between the source and target domain we did not expect a high result, and 30% is not too low. Those results are summarized in the picture below;

```
preds=model.predict(newhome_data)
targs=newhome_target
print("accuracy: ", metrics.accuracy_score(targs, preds))
print("precision: ", metrics.precision_score(targs, preds))
print("recall: ", metrics.recall_score(targs, preds))
print("f1: ", metrics.f1_score(targs, preds))
print("area under curve (auc): ", metrics.roc_auc_score(targs, preds))
```

```
accuracy:  0.27559412031341063
precision:  1.0
recall:  0.27559412031341063
f1:  0.43210315244428654
```

- This one class classifier needs to be extended to consider the effects of transfer learning, and it needs to be adapted to accept the data in the form of images.
- Images are fairly straightforward to use for the SVM as we can use the pandas package to import the previously generated PNG files as a dataframe with the grayscale converted to a numerical value (a form that SVM accepts).

Domain Adaptation Transfer Learning Technique with MNIST Images

- A collection of python files can be found within the github project page under '/TLWithDomainAdaptation' which is roughly based on the concepts found in the transfer learning paper 'DomainAdaptation.pdf' in the Papers directory.
- This technique takes a source and target domain set of images and attempts to align them. It is based on an assumption that the two domains share some generalizable characteristics which would be discovered over the course of the transfer learning process.
- It achieves a high level (95%+) of accuracy in just a short time (less than 5 epochs) even when the two datasets are not very similar. Without transfer learning the accuracy is quite low without any re-training of the model.

- This experiment shows the efficacy of implementing transfer learning in the domain of image/pattern recognition.
- This type of transfer learning (Domain Adaptation) appears to be the best suited as we know that our source and target domain will share some characteristics between locations/networks.

Currently Exploring Further Experiments

- Base performance of a one class classifier on images representing similar information as the CSV files
- Implement a grid search to find optimal values for gamma, nu and kernel type to see if the accuracy can be improved and to avoid overfitting and then see how this performs in the target
- Fine-tuning the OCSVM approach to improve accuracy
- Application of a simple transfer learning protocol such as TrAdaBoost to assess the changes in accuracy
- Domain Adaptation adapted from images to CSV files