CSE 472
Machine Learning Project

Network Flow Classification & Anomaly Detection

Submitted By:
Sadif Ahmed, 1905058
Abdullah Al Mohaimin, 1905041

Problem Definition

- Network flows: communication between two network endpoints at a specific time interval
- **Network flow classification**: anomalous/malicious traffic is detected and stopped/prevented
- Classifying network flows is an important problem, given that it needs to be fast and accurate
- **We are attempting to perform the classification and detection of flows in a benchmarked dataset**
- **Related Works** (utilizing same dataset)
    - FlowTransformer: A Transformer Framework for Flow-based Network Intrusion Detection Systems **(2024)** - Focuses on general encoder/decoder transformers and specialized models such as GPT, Bert for binary classification of network flows. This is the paper which we worked on.
    - A Novel Multi-Stage Approach for Hierarchical Intrusion Detection **(2023)** - This is the state of the art work on this dataset. Using a 2 stage approach where first stage uses One Class Support Vector Machine to do binary classification in Attacks vs Benign flow and the second stage uses Random Forest models to classify the Attacks, it achieves high accuracy and a good F1-Score.

Dataset: Improved CSE-CIC-IDS2018

- Original PCAP File Dataset: CSE-CIC-IDS2018
- Extracted Flow Source Paper: Towards a Standard Feature Set for Network Intrusion Detection System Datasets
- Description:
  - **Simulated network communication was collected in PCAP captures**
  - **Netflow, a network flow collection and analysis tool, was used to extract the flows**
  - Organized in CSV file, rows are separate flows
  - Each flow is labelled in detail
- High Level Information
  - **Columns: 43 features, 1 target; Rows: 18,893,708**
  - **Columns: Mostly numerical,** many can be dropped before training
- **Dataset is highly imbalanced as benign network traffic is the most prevalent flow in general**
- **Attack Sample Size: 2,258,141 (11.95%)**
- **Benign Sample Size: 16,635,567 (88.05%)**

Downsampling Dataset

- The original dataset was too large for processing with our computational resources
- It was **downsampled in 2 steps**
- **Step 1:** Extraction of all **Attack Samples** and almost same number of **Benign Samples**
- **Step 1** resulted in a balanced dataset of 4558141 rows, **24%** of original dataset
- **Step 2:** Extraction of **4%** of step 1 dataset in both balanced and skewed ratio
- **Step 2** resulted in datasets of roughly 50000 rows, **0.2%** original dataset
- We found that LLM models trained better on balanced dataset
- Both step 1 and step 2 dataset were used for training and evaluation

Proposed Solution

Transformer Architecture

Dataset Ingestion: Improved CICIDS2017 and CSECICIDS2018

Pre-Processing: One-hot Encoding

Input Encoder: Record Level Projection

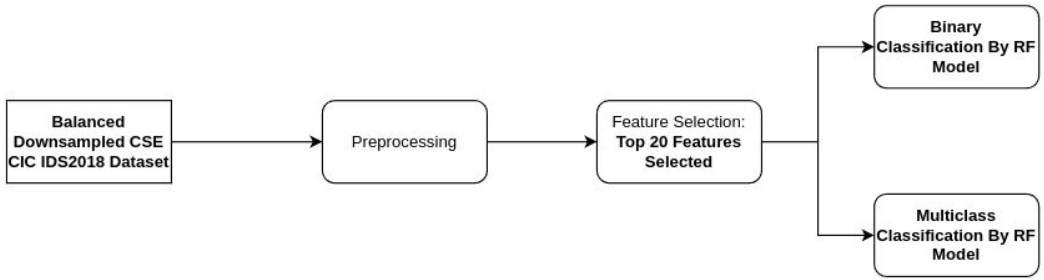
Transformers: Shallow Transformer Encoder GPT (Deep Decoder)

Classification Head: Last Token

Transformer Model: Train & Evaluate
Hyperparameters:
- Number of Attention Heads
- Number of Transformer Layers
- Internal Transformer Size
- Learning Rate
- Sequence Length

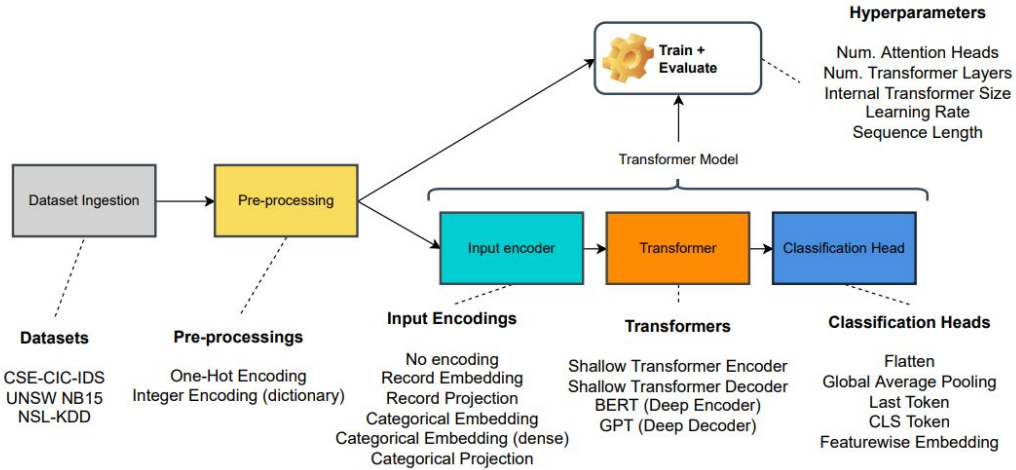
Models Utilized: Random Forest

- These are the most common and often most accurate models used in Network Flow Classification problems
- Random forest with feature selection gave good results
- Random forest was also very fast compared to any transformer based models
- The SOTA model uses random forest in its second stage for multiclass classification
- We performed both multiclass and binary classification using random forest model



Models Utilized: FlowTransformer

- **FlowTransformer:** This is the framework developed in the paper we chose to work on
- It uses different types of transformer models to do binary classification of flows
- Both shallow/deep and Encoder/Decoder based models are implemented
- We used only the best performing options at each choice and could replicate the results
- Using **feature selection** beforehand, the **results were incrementally improved**



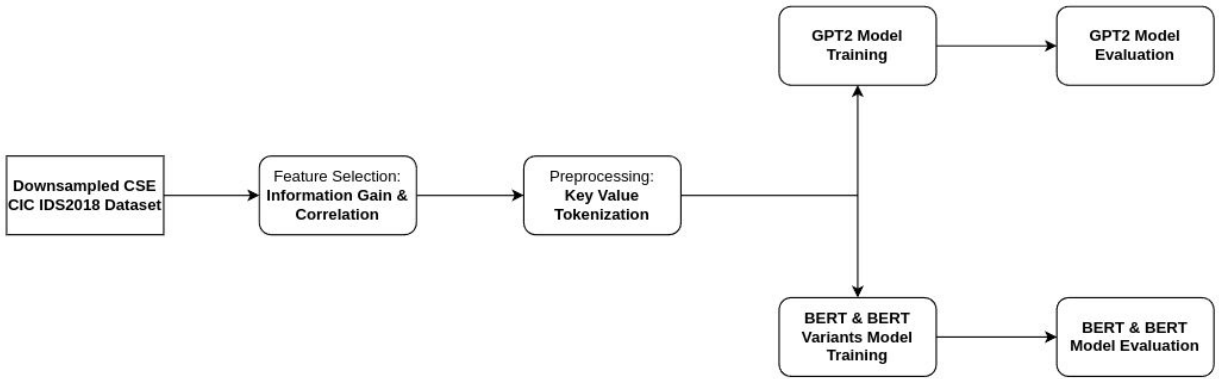
Models Utilized: BERT and Variants

- **Bidirectional encoder representations from transformers (BERT)** are encoder only transformer models
- It is one of the first well known Large Language Models(LLM) and have been modified to do many different tasks
- Encoders take in input vector sequences, uses **self-attention and feed forward mechanisms** to extract important contexts/information from the input
- Encoders were shown to be most promising models in the **FlowTransformer** paper
- We chose to use different variants of **BERT for multiclass classification**:
  - **BERT**
  - **CodeBERT:** A BERT variant specially modified for code generation
  - **DistillBERT:** A smaller distilled version of BERT, with comparable performance
  - **Electra:** Uses Generative Adverserial Network(GAN) alongside BERT
  - **RoBERTa:** An implementation improvement of BERT
  - **SpanBERT:** Improved variant, focusing on text span representation and prediction
- We used **Key Value** pair encoding for preprocessing the flows

Models Utilized: GPT Variants

- **Generative Pretrained Transformers(GPT)** are decoder only transformer models
- Well known as an LLM, GPT based models can also do classification
- **Decoders** function similar to Encoders, but with additional attention mechanism inserted
- Decoders take in the inputs, apply cross and self attention which are modified such that **autoregressive** text generation is possible
- For classification, different classification heads take the generated output and classify data
- Decoder base GPT were found to do well surprisingly in the paper, partly because decoders consider only previously seen output and input embeddings, allowing for temporal data like network flow to be classified
- We chose to use **GPT2** for its relatively small size and good performance
- We used **Key Value** pair encoding for preprocessing the flows

BERT & GPT Pipeline



Previous Results: FlowTransformer and State of the Art Model

- FlowTransformer: Binary Classification

Model and Dataset		*Best Performing Components and Hyperparameters*					*Model Results*					
Dataset	**Transformer**	**Layers**	**Heads**	**FF Dim**	**Classification Head**	**Input Encoding**	**Parameters**	***Throughput (flows/s)* Train**	***Throughput (flows/s)* Inference**	**F1 Score**	**Detection Rate**	**False Alarm Rate**
CSE_CIC_IDS	Shallow Encoder	2	2	128	Featurewise Emb.	Record Emb. Dense	186,370	1,606	8,534	97.05%	95.96%	0.10%
		2	2	256	Last Token	Record Emb. Dense	714,369	1,281	8,394	97.00%	95.77%	0.14%
	Shallow Decoder	2	2	128	Last Token	No Input Encoding	1,538,673	961	8,127	96.23%	95.21%	0.34%
		2	2	128	Flatten	No Input Encoding	1,798,513	945	5,626	96.05%	95.02%	0.35%
	GPT Model	12	12	768	Last Token	Record Projection	3,607,425	536	834	96.93%	95.86%	0.11%
	(Deep Decoder)	12	12	768	Last Token	No Input Encoding	53,979,633	59	786	96.90%	95.14%	0.13%
	BERT Model	12	12	768	Flatten	Record Projection	29,921,153	89	277	95.80%	95.48%	0.47%
	(Deep Encoder)	12	12	768	Last Token	Categorical Emb. Dense	79,637,585	74	149	95.41%	94.73%	0.43%

- State Of The Art Model: Multiclass Classification

RESULTS FULL MULTI-STAGE APPROACH

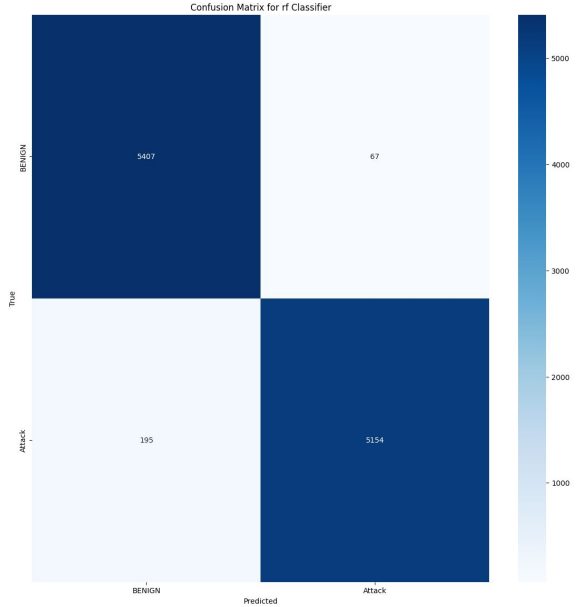
	τ_B	τ_M	τ_U	F1 weighted	F1 macro	Accuracy	Bal. Accuracy	Bandwidth reduction	Zero-day recall	Inference (s)
Max F-score	F5-8	F1	0.995	**0.9897**	**0.8276**	**0.9877**	0.8954	68.75%	0.5957	7.808 ± 0.009
Max bACC	F9	F1	0.95	0.9580	0.7496	0.9341	**0.9608**	57.91%	**0.9574**	8.043 ± 0.065
Balanced	F5-8	F1	0.99	0.9875	0.8231	0.9834	0.9342	68.75%	0.8723	7.882 ± 0.054
RF Baseline	-	-	-	0.9849	0.7981	0.9832	0.8877	-	0.8936	1.525 ± 0.013
Bovenzi et al. [16]	F3	F1	-	0.9383	0.7549	0.8957	0.8550	**86.22%**	**0.9574**	6.969 ± 0.399

Experimentation Result: Random Forest

- **Binary:**

```
rf METRICS:

ACCURACY:       0.9758
F1 SCORE:       0.9758
PRECISION:      0.9762
RECALL:         0.9757
```



- **Multiclass:**

```
rf METRICS:

ACCURACY:       0.9753
F1 SCORE:       0.7907
PRECISION:      0.8072
RECALL:         0.7825
```

Experimentation Result: BERT and Variants on Multiclass Classification

Here we report the highest performing **BERT** model: **DistillBERT**

```
Distillbert
Overall Validation Results:
Mean Validation Loss: 0.2003
Mean Validation Accuracy: 0.9501
Mean Precision: 0.5301
Mean Recall: 0.5735
Mean F1 Score: 0.5466
```

- Other BERT variants did poorly in terms of both Accuracy and F1-score.
- **CodeBERT** and **SpanBERT** did comparatively better, with high accuracy but still low (<0.50) F1-Score
- BERT models took a reasonable time to train, but their low F1-score shows they will need much finetuning/hyperparameter tuning before using on realistic network flow datasets

Experimentation Result: GPT2 on Multiclass Classification

For GPT2 we experimented on the learning rate. We found that **0.0005** gave the highest score

```
Model Name: gpt2lr: 0.0004999999999999999
Overall Validation Results:
Mean Validation Accuracy: 0.9781
Mean Precision: 0.8625
Mean Recall: 0.7757
Mean F1 Score: 0.7906
Mean Validation Loss: 0.0909
```

- **GPT2** model did very well in terms of both **accuracy and F1-Score**
- The FlowTransformer predicted this behaviour as **Autoregressive** models check out only the previous outputs and input embedding, making them suitable for such flow classification
- The drawback to GPT models would be their large size and very large training time

Discussion

- Our **attainment** has been doing **one stage multiclass flow classification using transformer models**
- As we observed in the experimental results section, **GPT models** did very well, **almost reaching SOTA scores**
- However, GPT models require large memory and long time
- Unsurprisingly, Random Forest models did very well and thus still remains the state of the art in network flow classification
- Our experimentation shows that **Feature Selection** can **improve results**
- An initial feature selection step over all of the dataset enabled us to find top 20 features, thereby reducing time and memory usage for any further model run
- Another find was weakness of BERT models in classifying flows given their weak F1-score

Thank You!

We welcome any questions!