DEEP LEARNING PROJECT

COMPUTER NETWORK ANOMALY DETECTION

hi

INTRODUCTION

With the enormous growth of computer networks usage and the huge increase in the number of Applications running on top of it, network security is becoming increasingly more important.
We built a Network intrusion detection system to detect anomalies and attacks in the Network. There are two problems.

1. Binomial Classification: Activity is normal or attack
2. Multinomial classification: Activity is normal or DOS or PROBE or R2L or U2R

We have encountered and used various approaches to correctly detect and classify the anomalies and improved the attack classification significantly compared to any other presently shared solution that we came across. We successfully implemented upsampling of Dataset to prevent the loss in learning of Minority attack class, while keeping the test set same to increase the difficulty for the model. The architectures used for the model are:

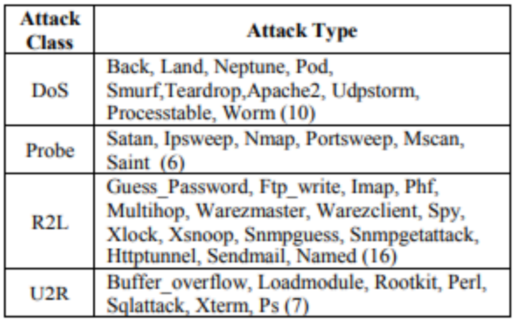
1.AutoEncoder and Neural Net classification
2.1D convolution
3.FT transformer

ABOUT THE DATASET

The data used is the **KDDCUP99** data set, which is widely used as one of the few publicly available datasets for network-based anomaly detection systems.

Number of Datapoints in the Train Set - 1,44,767 Number of Datapoints in the Test Set - 28954
Features of the Dataset- In total 42 Features.
LIST OF COLUMNS FOR THE DATA SET:
["duration","protocol_type","service","flag","src_bytes","dst_bytes","land","wrong_fragment","urgent","hot","num_failed_logins","logged_in","num_compromised","root_shell","su_attempted","num_root","num_file_creations","num_shells","num_access_files","num_outbound_cmds","is_host_login","is_guest_login","count","srv_count","serror_rate","srv_serror_rate","rerror_rate","srv_rerror_rate","same_srv_rate","diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_srv_count","dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate","dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate","dst_host_rerror_rate","dst_host_srv_rerror_rate","attack", "last_flag"]

Target Feature-"Attack"(Labels- DoS, U2R, Probing, R2L). Additional Target Feature-”Is_Anomaly”(1: Attack, 0: Normal)



Attack Class	Attack Type
DoS	Back, Land, Neptune, Pod, Smurf,Teardrop,Apache2, Udpstorm, Processtable, Worm (10)
Probe	Satan, Ipsweep, Nmap, Portsweep, Mscan, Saint (6)
R2L	Guess_Password, Ftp_write, Imap, Phf, Multihop, Warezmaster, Warezclient, Spy, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named (16)
U2R	Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps (7)

Distribution of Attack Classes in Test and train dataset,
Train Data Attack Class Distribution:{'DoS': 41334, 'Normal':61643, 'Probe': 10210, 'R2L': 2555, 'U2R': 71}
Test Data Attack Class Distribution:{'DoS': 10334, 'Normal':15411, 'Probe': 2552, 'R2L': 639, 'U2R': 18}

AUTOENCODERS APPROACH

The above given diagram(ref- https://onlinelibrary.wiley.com/doi/10.1155/2021/9054336) illustrates the basic architecture of the Model implemented by the team in the Project. Although due to severe low quantity of Attack some classes like U2R(2500) and R2L(71) in the train and test data, we decided to upsample the train dataset using SMOT(Synthetic Minority Oversampling Technique) and VAE(Variational AutoEncoder) .
Features of the Model-

- The First autoencoder layer compresses the dimensions of the data from 42 to 32.
- The Neural Network layer consists of two softmax fully connected layers, one for classification of anomaly in the network(Yes/No)
- The other softmax layer in the network gives the output about the type of attack in the network(DoS, R2L, Probing, U2R).

Note- No upsampling was applied to the test dataset to check the performance of the model on the authentic dataset.

Result Metrics

1. Anomaly Detection

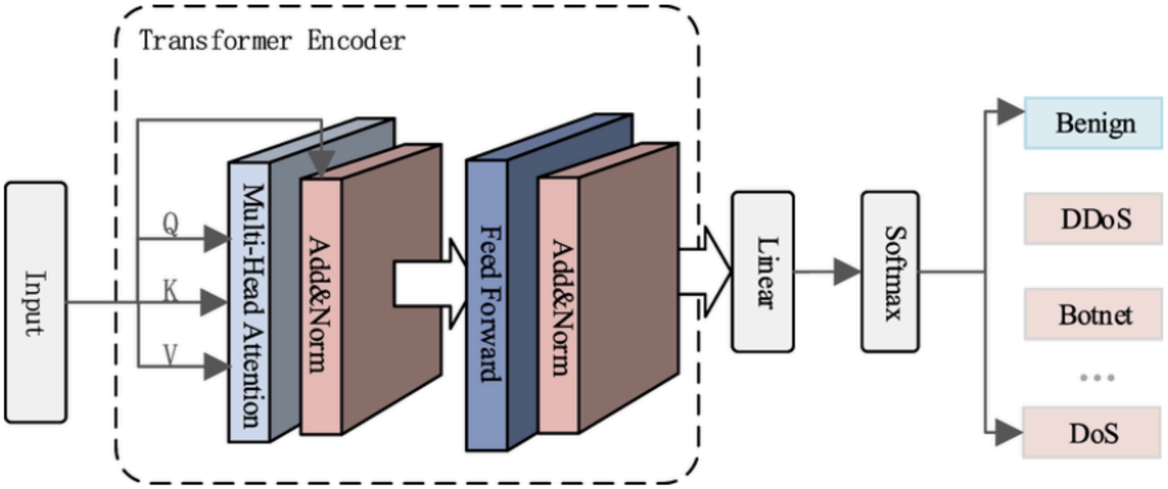
Model	Basic Autoencoder	SMOT Upsampled	VAE Upsampled
F1 Score	0.96	0.97	0.98

2. Attack Type Classification

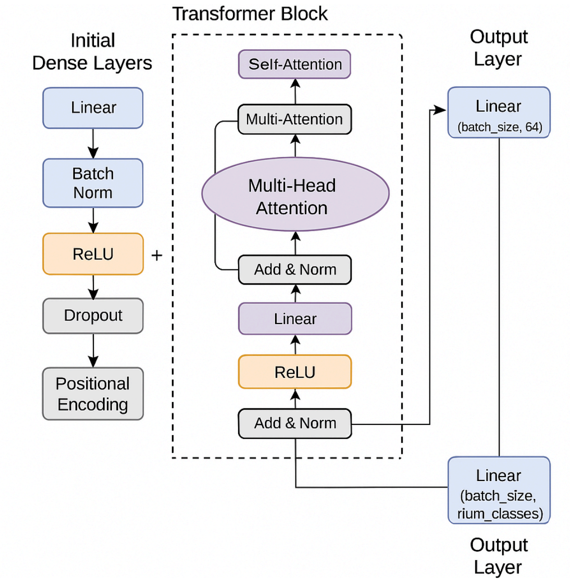
Attacks	DoS(F1 SCORE)	Probing(F1 SCORE)	R2L(F1 SCORE)	U2R(F1 SCORE)
Basic Autoencoder	98	93	34	06
SMOT Upsampled	96	88	68	03
VAE Upsampled	98	93	76	58

Clearly, VAE Upsampled Model Performed the best, so further hyperparameter finetuning is performed on the model, and using grid search the most accurate results came out to be for the parameters,
Learning rate=0.001, batch size=512, dropout=0.3, hidden sizes=(128,64).

Transformer-Based Approach



(Ref:https://www.researchgate.net/publication/377081740_A_Transformer-based_network_intrusion_detection_approach_for_cloud_security)



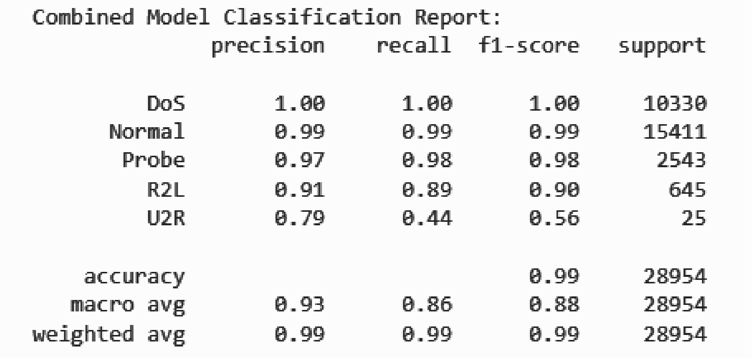
We developed a two-stage framework for network anomaly detection and attack classification using transformer-based architectures. The project begins by preprocessing a network traffic dataset where various attack types are mapped into higher-level categories such as DoS, Probe, R2L, and U2R, and a binary anomaly label is created to distinguish normal traffic from attacks. Categorical features are encoded, and numerical features are standardized before splitting the data into training and test sets.

The first stage of the framework is an anomaly detector. This model uses several dense layers with batch normalization, ReLU activations, and dropout to reduce dimensionality and extract robust features from the input data.
A transformer block—incorporating multi-head attention and a feed-forward network with residual connections and layer normalization—is then applied to capture complex inter-feature relationships. The final output layer produces logits for a binary decision on whether the input is normal or anomalous.

The second stage is an attack classifier that operates only on samples identified as anomalous. It follows a similar design with fully connected layers for feature extraction, followed by a transformer block that refines these features. A final classification layer outputs predictions for the specific attack types. This two-stage approach allows for a detailed classification of network attacks while filtering out normal traffic.

During training, both models are optimized using cross-entropy loss and the Adam optimizer. The anomaly detector is trained first, and its outputs determine which samples are passed on to the attack classifier. Evaluation of the combined models is performed using a classification report and confusion matrix. The final results show overall accuracy of approximately 99%, with particularly high precision and recall for DoS, Normal, and Probe classes. However, the U2R category shows lower performance metrics, likely due to its rarity in the dataset

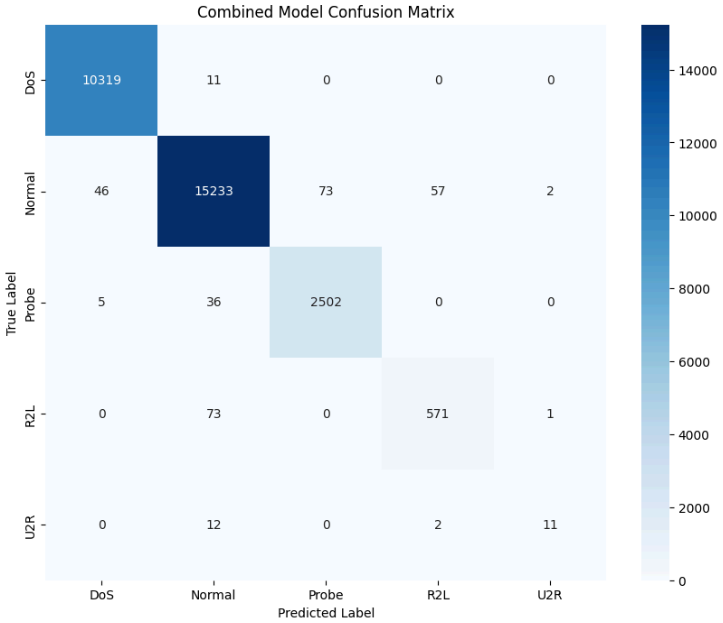
Result:



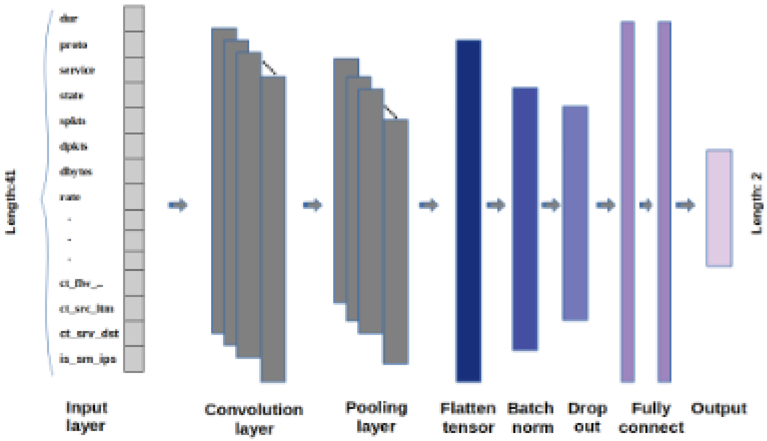
```
Combined Model Classification Report:
              precision    recall  f1-score   support

         DoS       1.00      1.00      1.00     10330
      Normal       0.99      0.99      0.99     15411
       Probe       0.97      0.98      0.98      2543
         R2L       0.91      0.89      0.90       645
         U2R       0.79      0.44      0.56        25

    accuracy                           0.99     28954
   macro avg       0.93      0.86      0.88     28954
weighted avg       0.99      0.99      0.99     28954
```



1D CONVOLUTION APPROACH



Inspired from this architecture

reference-https://www.researchgate.net/publication/340697891_1D_CNN_based_network_intrusion_detection_with_normalization_on_imbalanced_data#pf3

Stage 1

Model Architecture

(a) Convolutional Layers

CNNs are typically used for image or sequence data, but here it's applied to **tabular data`s=1, out_channels=64`**

  - **Takes 1D input** (batch size, 1, features).
  - Uses **64 filters**, each scanning over 3 features at a time.
  - **Padding = 1** ensures output size remains the same as input.
- **Batch Normalization ()**
  - Applied after each convolution to normalize activations and stabilize training.
- **ReLU Activation ()**
  - Introduces non-linearity to detect complex feature interactions.

(b) Feature Extraction Pipeline

- **Conv1 → BatchNorm → ReLU**
- **Conv2 → BatchNorm → ReLU**
- **Conv3 → BatchNorm → ReLU**

Each **convolutional layer** increases the number of filters, **extracting deeper feature representations**.

Stage 2

Model Training

- Uses **CrossEntropyLoss** for multi-class classification.

- Optimized with **Adam optimizer** ().

- **20 epochs** with loss tracking.

- Each batch passes through the CNN, computes loss, updates weights using **backpropagation**.

Stage 3

CONFUSION MATRIX

10321	13	0	0	0	Actual
17	15321	34	39	0	
3	49	2499	1	0	
0	48	0	589	2	
0	9	0	2	7	

DOS NORMAL Probe U2R R2L

Predicted

Extracting Local Patterns

- In **time-series or tabular data**, certain features might have local dependencies. A **1D CNN** can capture these local structures by applying filters (kernels) over small feature subsets.

- Unlike traditional **fully connected layers**, which treat all features as independent, a CNN can learn spatially related patterns.

REFERENCES

- https://onlinelibrary.wiley.com/doi/10.1155/2021/9054336

- https://www.researchgate.net/publication/377081740_A_Transformer-based_network_intrusion_detection_approach_for_cloud_security

- https://www.researchgate.net/publication/340697891_1D_CNN_based_network_intrusion_detection_with_normalization_on_imbalanced_data#pf3