Independent Study

Topic: Synthetic Network Traffic Generation

Name: Abijith Trichur Ramachandran
Instructor: Dr. Eric Keller

Topics we will be covering ...

1. Network Traffic
2. Synthetic Traffic Generation
3. CTGAN
4. Application of CTGAN in Network Traffic
5. Results
6. Future Work

Network Traffic

What is it?

- Network traffic is the amount of data moving across a computer network at any given time.
- Network traffic, also called data traffic, is broken down into data packets and sent over a network before being reassembled by the receiving device or computer.

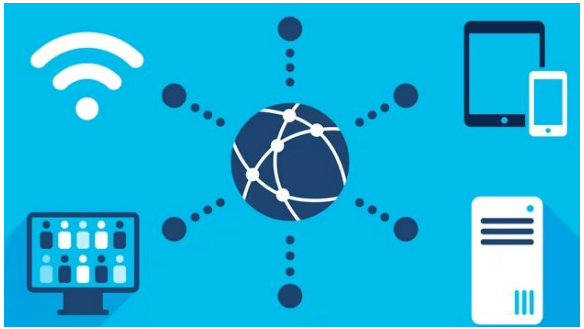
Why is it important?

- Network traffic analysis is an essential way to monitor network availability and activity to identify anomalies, maximize performance, and keep an eye out for attacks.

What are the challenges in acquiring it ?

- Challenges in obtaining real network traffic data: privacy issues, sensitive information, and limited access to diverse traffic scenarios.

Synthetic Traffic Generation



Creating artificial network data that imitates genuine user behavior and network traffic patterns.

Purpose:

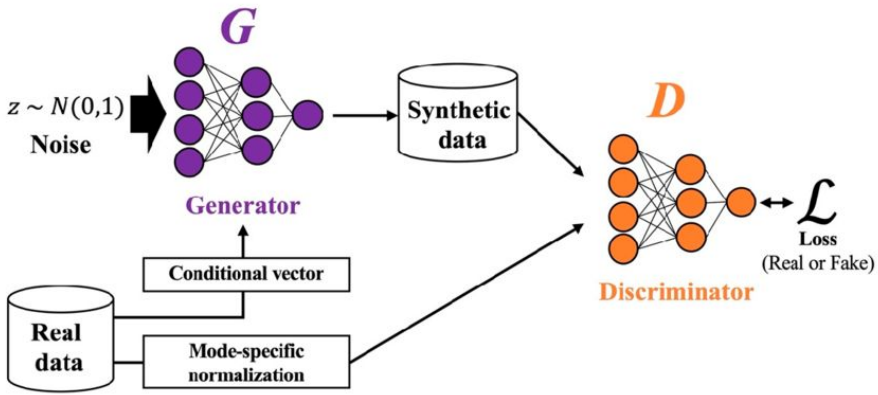
- To evaluate network performance, security systems, and scalability.
- To train and validate machine learning models used in anomaly detection.

Challenges:

- Ensuring realism and relevance of the generated traffic to actual network conditions.
- Balancing complexity and computational efficiency in traffic generation.

CTGAN

- A Conditional Tabular Generative Adversarial Network (CTGAN) is a variant of the traditional Generative Adversarial Network (GAN) that is designed to generate synthetic data that is conditional on class labels or other conditioning inputs.
- Purpose: Enhances the ability of GANs to handle imbalanced data, complex categorical variables, and to generate data samples for specific, under-represented conditions.



- **Training Process:**
- **Conditional Mechanism:** Incorporates additional information (e.g., class labels) into both the Generator and Discriminator to guide the data generation process.
- **Iterative Improvement:** The Generator and Discriminator iteratively improve through training, where the Generator learns to produce more realistic data, and the Discriminator becomes better at detecting differences.

Application of CTGAN in Generating Synthetic Network Traffic

Role of CTGAN:

- CTGANs generate realistic, synthetic network traffic conditioned on specific network behaviors or scenarios.

Capability:

- Able to produce diverse network traffic scenarios including normal operations, peak loads, and security threats like DDoS attacks.

Application of CTGAN in Generating Synthetic Network Traffic

Advantages of Using CTGAN:

- Enhanced Realism: Produces highly realistic traffic patterns by learning from real network data.
- Conditioning on Scenarios: Generates traffic based on specific conditions (e.g., type of attack, time of day), which is crucial for targeted testing and training.
- Improved Anomaly Detection: Helps in training more robust anomaly detection systems by providing varied, realistic examples of network anomalies.

Implementation Steps:

1. **Data Collection:** Gather real network traffic data to train the CTGAN model.
2. **Model Training:** Train the CTGAN model to learn the distributions and characteristics of the collected network traffic.
3. **Traffic Generation:** Use the trained model to generate synthetic traffic conditioned on desired scenarios.

Data Collection and Feature Removal

Netflow Dataset

Obtained the [netflow dataset from Kaggle]
Contains 33 Headers:

FLOW_ID,
PROTOCOL_MAP,
L4_SRC_PORT,
IPV4_SRC_ADDR,
L4_DST_PORT,
IPV4_DST_ADDR,
FIRST_SWITCHED,
FLOW_DURATION_MIL
LISECONDS,
LAST_SWITCHED,
PROTOCOL,
TCP_FLAGS,
TCP_WIN_MAX_IN,
TCP_WIN_MAX_OUT,
TCP_WIN_MIN_IN,
TCP_WIN_MIN_OUT,
TCP_WIN_MSS_IN,
TCP_WIN_SCALE_IN,
TCP_WIN_SCALE_OUT,
SRC_TOS,
DST_TOS,
TOTAL_FLOWS_EXP,
MIN_IP_PKT_LEN,
MAX_IP_PKT_LEN,
TOTAL_PKTS_EXP,
TOTAL_BYTES_EXP,
IN_BYTES,
IN_PKTS,
OUT_BYTES,
OUT_PKTS,
ANALYSIS_TIMESTAMP,
ANOMALY,
ALERT, ID

Acquiring geolocation data

Using the ip2geotools library we were able to obtain the following headers for each IP

'latitude': 22.28805345, 'longitude': 114.15557015547806, 'city': 'Hong Kong', 'region': 'Central and Western', 'country': 'HK'

Integration of the two and Removal of Features

The netflow dataset was then merged with geolocation data and features that have no major implications for generating the final data were removed thus the final headers were.

PROTOCOL_MAP,
L4_SRC_PORT,
IPV4_SRC_ADDR,
L4_DST_PORT,
IPV4_DST_ADDR,
PROTOCOL,
TCP_FLAGS,
IN_BYTES,
IN_PKTS,
OUT_BYTES,
OUT_PKTS,
ANOMALY,
ALERT,
src_hierarchy,
dst_hierarchy

Feature Preprocessing

Binary Columns - OneHotEncoder

- ANAMOLY
- ALERT

Count Columns- Function Transformer using log

- IN_BYTES
- IN_PKTS
- OUT_BYTES
- OUT_PKTS

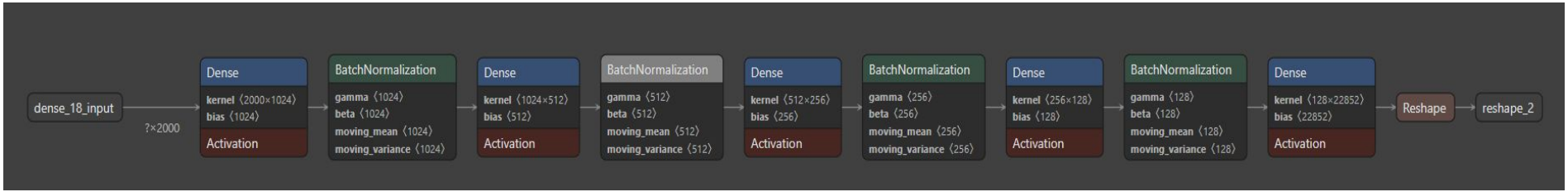
Categorical Columns - OneHotEncoder

- PROTOCOL_MAP
- IPV4_SRC_ADDR
- IPV4_DST_ADDR
- PROTOCOL
- TCP_FLAGS
- src_hierarchy
- dst_hierarchy

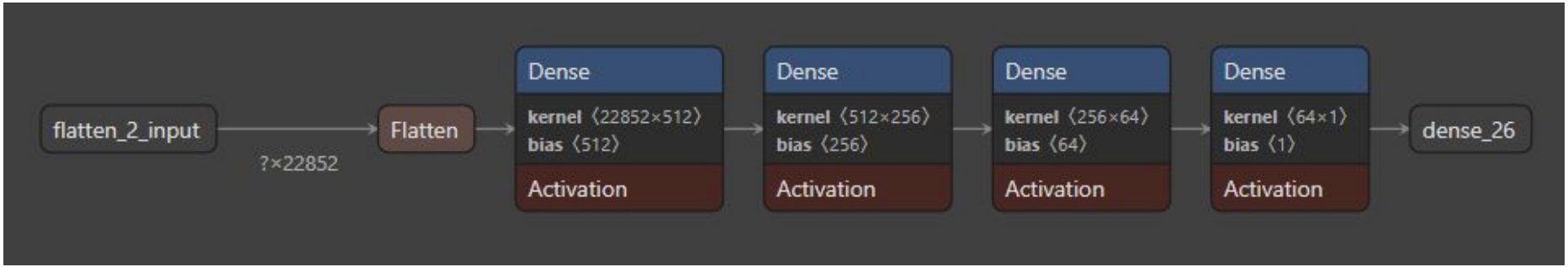
Numerical Columns - StandardScaler

- L4_SRC_PORT
- L4_DST_PORT
- src_latitude
- src_longitude
- dst_latitude
- dst_longitude

Generator Model without Dropout

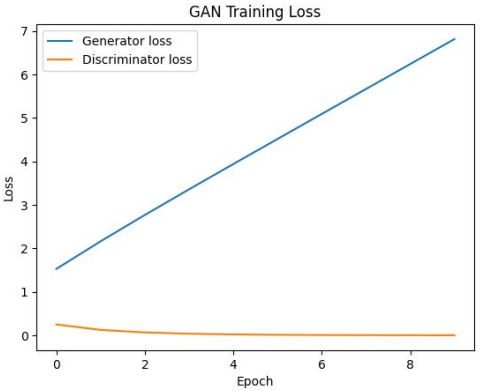


Discriminator Model without Dropout



Results

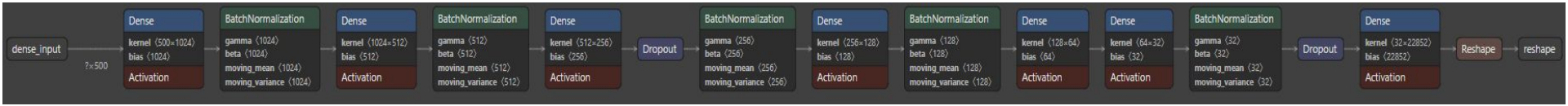
```
PROTOCOL_MAP,L4_SRC_PORT,IPV4_SRC_ADDR,L4_DST_PORT,IPV4_DST_ADDR,PROTOCOL,TCP_FLAGS,IN_BYTES,IN_PKTS,OUT_BYTES,OUT_PKTS,ANOMALY,ALERT,src_hierarchy,dst_hierarchy
gre,0,3.209.83.10,0,143.244.58.247,17,10,38800,8283,8,-6,21,-5,ID>Central Java>Semarang,ID>West Java>Cimahi
tcp,1,144.206.237.80,0,10.114.224.249,57,26,46319,5564,7,5,17,-15,US>California>Los Angeles (Downtown Los Angeles),US>Missouri>Kansas City
tcp,0,120.201.0.108,0,216.58.212.161,6,194,38515,12493,11,0,17,-13,NL>Drenthe>Meppel,IN>Bihar>Bānka
gre,0,54.183.13.24,0,178.62.192.243,6,10,34584,11162,11,-8,15,-5,RU>Kirov>Kirov,ID>West Java>Cimahi
tcp,0,193.121.171.125,0,67.199.248.15,17,26,42840,7847,10,0,15,-18,CH>Vaud>Renens,NL>North Holland>Amsterdam
icmp,1,165.22.221.186,0,68.79.49.181,57,26,43739,6928,7,4,18,-16,US>California>Los Angeles (Downtown Los Angeles),US>Missouri>Kansas City
udp,1,15.152.35.23,0,64.68.192.10,57,30,45182,7877,5,4,21,-14,US>California>Los Angeles (Downtown Los Angeles),GB>England>Croydon
icmp,0,13.233.15.206,0,98.138.11.157,17,10,36083,10314,11,-10,17,-19,IN>Delhi>Shahdara,IT>Apulia>Taranto
gre,0,185.102.217.173,0,54.192.235.97,1,10,38460,9393,4,-8,25,-13,DE>Hesse>Frankfurt am Main (Frankfurt am Main Ost),IT>Apulia>Taranto
udp,0,49.45.28.84,0,106.52.127.240,6,20,38409,10196,8,-4,18,-6,ID>Central Java>Semarang,CA>Ontario>Guelph
gre,0,172.19.6.215,0,54.194.68.215,1,152,41488,7851,7,0,19,-11,US>New Mexico>Roswell,HU>Vas>Szombathely
skip,1,195.230.103.250,0,10.114.224.117,57,26,53397,918,2,5,24,-26,BG>Sofia-grad>Sofia,US>Missouri>Kansas City
icmp,0,168.7.208.54,0,81.91.161.98,6,156,34439,15695,11,-7,19,-10,HK>Southern>Aberdeen,DK>South Denmark>Kolding
gre,0,34.226.141.97,0,192.168.1.90,47,28,47812,6595,0,4,26,-18,GB>Scotland>Scalloway,CA>Ontario>Hamilton
skip,1,185.142.236.36,0,104.85.13.156,57,26,49984,2548,5,7,19,-21,BG>Sofia-grad>Sofia,US>Missouri>Kansas City
tcp,1,116.98.170.103,0,10.114.225.204,1,27,39588,7616,12,-1,14,-15,BG>Sofia-grad>Sofia,CA>Quebec>Montreal
gre,0,129.211.124.204,0,178.62.192.243,6,10,34710,13832,8,-7,20,-4,JO>Amman>Amman,US>New York>Islip
gre,0,3.209.83.10,0,143.244.58.247,6,10,35347,8774,12,-8,17,-6,GB>England>Great Hanwood,ID>West Java>Cimahi
tcp,1,147.124.109.33,0,192.31.80.30,6,26,40066,8714,10,3,15,-12,US>California>Los Angeles (Downtown Los Angeles),TW>Taipei>Banqiao District
icmp,0,185.191.171.23,0,10.114.224.101,17,10,35860,9838,11,-6,16,-9,CN>Guizhou>Guiyang,ID>West Java>Cimahi
gre,0,44.201.44.159,0,52.114.76.233,17,10,40508,9560,5,-4,22,-5,ID>Central Java>Semarang,NL>Gelderland>Nijmegen (Nijmegen-Centrum)
skip,1,185.142.236.36,0,68.79.49.181,57,26,49163,3730,5,8,19,-20,US>California>Los Angeles (Downtown Los Angeles),US>Missouri>Kansas City
tcp,1,185.42.204.93,0,18.66.25.126,57,26,45679,5707,7,6,17,-13,US>California>Los Angeles (Downtown Los Angeles),RU>Kuzbass>Novokuznetsk
tcp,1,79.114.174.37,0,20.49.150.241,57,26,44373,7464,5,4,19,-8,DE>Saxony-Anhalt>Halle (Mitte),RU>Dagestan>Buynaksk
```



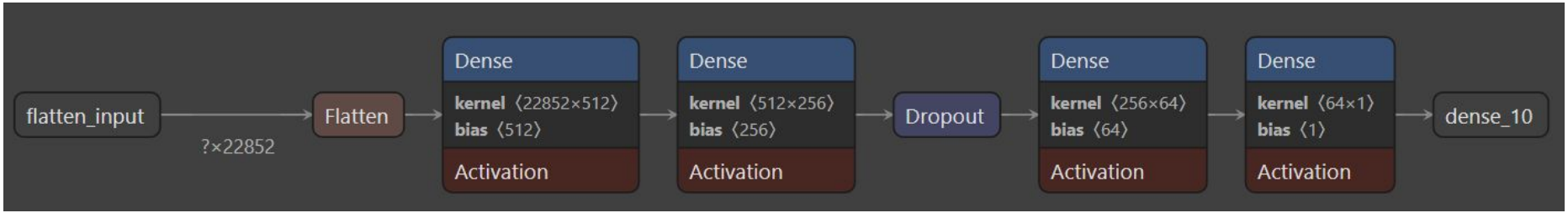
Batch size: 64
Latent Dimension: 1000
Size: 50000 data points

```
Epoch: 10, D loss: 0.0011042088735848665, G loss: 6.811031818389893
```

Generator Model with Dropout



Discriminator Model with Dropout



Hyper-parameter Tuning

Could also add the loss function as a hyper parameter but would significantly increase the total process of acquiring the best model

Generator: Runs every epoch
Discriminator: Runs every 3 epoch

Takes 10 hours to train with a training set of 20000 rows

Batch Size

Number of training samples in one iteration.

32, 64, 128

Latent Dimension

This is the size of the random noise vector that is input to the generator.

500, 1000, 2000

Learning Rate

This is a hyperparameter that determines the step size at each iteration while moving toward a minimum of a loss function.

0.0001, 0.00001, 0.00001

Hyperparameter Tuning Results
(After 10 epochs for 20000 data points)

Batch Size: 32

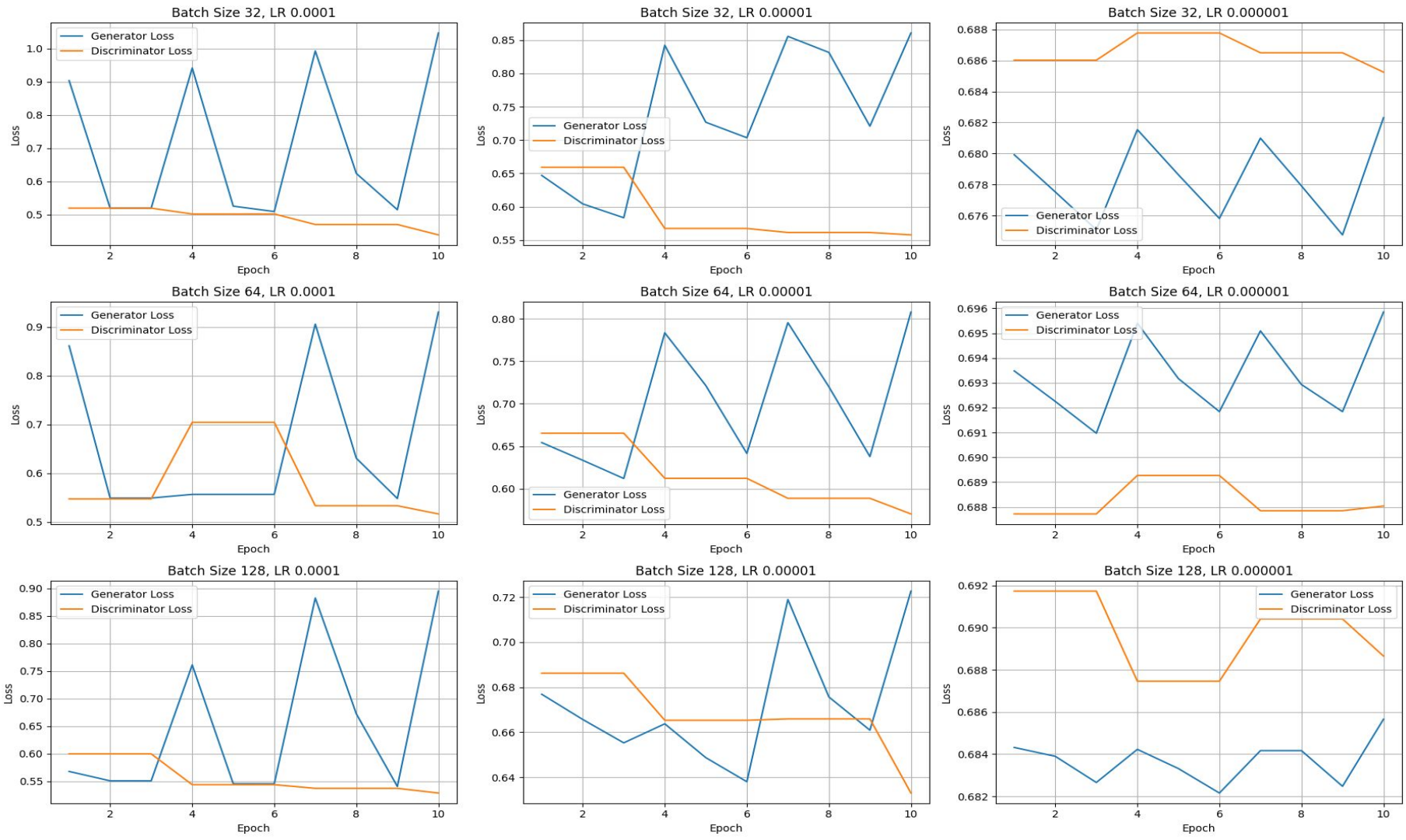
Latent Dimension	Learning Rate					
	0.0001		0.00001		0.000001	
	D_Loss	G_Loss	D_Loss	G_Loss	D_Loss	G_Loss
500	0.43872	1.04752	0.55775	0.86060	0.68525	0.68230
1000	0.45483	1.03492	0.54385	0.86684	0.68470	0.68607
2000	0.43362	1.05354	0.55152	0.85048	0.68561	0.68264

Batch Size: 64

Latent Dimension	Learning Rate					
	0.0001		0.00001		0.000001	
	D_Loss	G_Loss	D_Loss	G_Loss	D_Loss	G_Loss
500	0.51606	0.93045	0.57031	0.80789	0.68804	0.69585
1000	0.50156	0.94474	0.56245	0.82555	0.68881	0.70994
2000	0.49882	0.94539	0.56332	0.82204	0.68820	0.68043

Batch Size: 128

Latent Dimension	Learning Rate					
	0.0001		0.00001		0.000001	
	D_Loss	G_Loss	D_Loss	G_Loss	D_Loss	G_Loss
500	0.52841	0.89504	0.63296	0.72268	0.68865	0.68566
1000	0.52408	0.89949	0.63951	0.73657	0.68973	0.69643
2000	0.52974	0.89498	0.63091	0.73554	0.68973	0.70017


Batch Size 32, LR 0.0001
Batch Size 32, LR 0.00001
Batch Size 32, LR 0.000001
Batch Size 64, LR 0.0001
Batch Size 64, LR 0.00001
Batch Size 64, LR 0.000001
Batch Size 128, LR 0.0001
Batch Size 128, LR 0.00001
Batch Size 128, LR 0.000001
Generator Loss
Discriminator Loss
Loss
Epoch

Future Works

1. Further hyperparameter tuning involving 50000 data points, involving different epochs (10, 15, 20, 25).
   This would result in a total configuration of
   4 (epochs) x 3 (batch_size) x 3 (learning_rate) x 3 (latent_dimension) = 104 configurations. (72+ hours of training approx)
   Of Course from the previous result we can conclude that smaller learning rate and higher latent dimension provides better result thereby limiting our next hyperparameter tuning configuration to 12 configurations.
2. Acquisition of much cleaner and high quality dataset
3. Real-time Adaptation: Enhance CTGAN models to dynamically adapt to real-time network traffic data, allowing for on-the-fly training and generation that reflects current network conditions.
4. Deployment in Network Simulation Tools: Integrate CTGANs into network simulation tools to provide more realistic traffic patterns for network planning and testing scenarios.
5. Efficiency Improvements: Work on reducing the computational requirements of CTGANs, making them more efficient and accessible for use in less powerful devices, potentially for edge computing scenarios.

References

1. Wang, J., Yan, X., Liu, L., Li, L., & Yu, Y. (2022). CTTGAN: Traffic Data Synthesizing Scheme Based on Conditional GAN. Sensors (Basel, Switzerland), 22(14), 5243. https://doi.org/10.3390/s22145243

2. Anande, T. J., Al-Saadi, S., & Leeson, M. S. (2023). Generative adversarial networks for network traffic feature generation. International Journal of Computers and Applications, 45(4), 297–305. https://doi.org/10.1080/1206212X.2023.2191072

3. jia, jingbo and Wu, Peng and Dawood, Hussain, An Improved CTGAN for Data Processing Method of Imbalanced Disk Failure (October 10, 2023). INTERNATIONAL JOURNAL ON CYBERNETICS & INFORMATICS (IJCI), 2023, Available at SSRN: https://ssrn.com/abstract=4598620 or http://dx.doi.org/10.2139/ssrn.4598620

4. Habibi, O., Chemmakha, M., & Lazaar, M. (2023). Imbalanced tabular data modelization using CTGAN and machine learning to improve IoT Botnet attacks detection. Engineering Applications of Artificial Intelligence, 118, 105669. https://doi.org/10.1016/j.engappai.2022.105669

5. Cullen, D., Halladay, J., Briner, N., Basnet, R., Bergen, J., & Doleck, T. (2022). Evaluation of synthetic data generation techniques in the domain of anonymous traffic classification. IEEE Access, 10, 129612-129625.

6. Bourou, Stavroula, Andreas El Saer, Terpsichori-Helen Velivassaki, Artemis Voulkidis, and Theodore Zahariadis. 2021. "A Review of Tabular Data Synthesis Using GANs on an IDS Dataset" Information 12, no. 9: 375. https://doi.org/10.3390/info12090375

Thank you