CAPSTONE PROJECT

Network Intrusion Detection

Presented By:
ANKIT SEN - UNIVERSITY OF KALYANI - DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

edunet foundation

OUTLINE

- **Problem Statement**
- **Proposed System/Solution**
- **System Development Approach** (Technology Used)
- **Algorithm & Deployment**
- **Result (Output Image)**
- **Conclusion**
- **Future Scope**
- **References**

PROBLEM STATEMENT

Developed a robust network intrusion detection system (NIDS) using machine learning. The
system should be capable of analyzing network traffic data to identify and classify various
types of cyber-attacks (e.g., DoS, Probe, R2L, U2R) and distinguish them from normal
network activity. The goal is to build a model that can effectively secure communication
networks by providing an early warning of malicious activities.

PROPOSED SOLUTION

- **The proposed system aims to address the challenge of detecting and classifying network intrusions by leveraging contrastive pre-training and graph neural networks for robust anomaly detection.The solution will consist of the following components:**

- **Data Collection:**
  - **Raw TCP/IP connection data (Train_data.csv & Test_data.csv) has been loaded from Kaggle.**
  - **Class labels ("normal" vs "anomaly") have been extracted and binarized.**

- **Data Preprocessing:**
  - Categorical features (`protocol_type`, `service`, `flag`) have been label-encoded.
  - Connection hashes have been generated to check for potential leakage.
  - Train/Test split has been done chronologically (60/40), and further split into pre-train (70%) and fine-tune (30%) subsets.

- **Machine Learning Algorithm:**
  - **A 4-node graph per connection has been built using 41 features, with edge attributes like byte count and duration.**
  - **A GIN encoder has been contrastively pre-trained with graph augmentations and loss regularization.**
  - **The model has been fine-tuned with a classification head using cross-entropy loss and scheduler support.**

- **Deployment:**
  - The final encoder + classifier have been bundled into an inference script.
  - Saved model weights and preprocessing scalers have been serialized.
  - A simple API or CLI has been prepared to load a CSV row, build its graph, and output a "normal" vs "anomaly" prediction.

- **Evaluation:**
  - **Test set performance has been measured: AUC = 0.9909, F1 = 0.9553, Accuracy = 0.9598.**
  - **Detailed precision/recall/F1 per class and overall metrics have been reported.**
  - **Data leakage check (connection_id overlap) has been done (0% overlap).**

SYSTEM APPROACH

The "System Approach" section outlines the overall strategy and methodology for developing and implementing the Network Intrusion Detection system. Here's a suggested structure for this section:

System Requirements

- **Hardware:** CPU with RAM >= 4GB
- **Software:** Python 3.8+, CUDA toolkit (if using GPU), Git

Required Libraries

- **Deep Learning & GNN:** `torch`, `torch_geometric`
- **Data & ML:** `pandas`, `numpy`, `scikit-learn`
- **Graph & Utils:** `networkx`, `tqdm`, `hashlib`, `warnings`
- **Technologies used:** IBM Cloud Lite Services (Watsonx AI studio)

WOW FACTOR

Extremely High Accuracy

- Achieved an AUC of **0.9912**, meaning the model almost perfectly separates normal traffic from attacks.
- Reached an overall accuracy of **96.08%** on unseen test data.

Balanced Performance Across Classes

- Both normal and anomalous connections get around **96%** F1-score, showing the model is equally good at detecting attacks and avoiding false alarms.

Novel Contrastive Pre-training

- Self-supervised pre-training steps **have been used** to teach the GNN to pull similar network graphs closer and push different ones apart—without labels.

Graph-based Representation

- Converting each connection into a small 4-node graph (source, destination, protocol, service) captures both packet features and protocol relationships. This structural view is more powerful than flat feature vectors.

ALGORITHM & DEPLOYMENT

- **Model:** A two-stage Graph Neural Network (GNN) workflow, combining contrastive pre-training (via a 3-layer GIN encoder with attention pooling and projection head) and supervised fine-tuning with a lightweight classification head.

- **Justification:** Contrastive pre-training learns robust structural and attribute representations from unlabeled graphs, improving downstream anomaly detection on highly imbalanced, temporal network-traffic data.

Data Input

- **Node features (per connection graph):**
  - Source/Destination byte ratios, connection counts, service rates (`src_bytes`, `dst_bytes`, `count`, `srv_count`, `same_srv_rate`, `diff_srv_rate`, etc.)
  - Protocol metadata (`duration`, `wrong_fragment`, `urgent`)
  - Service indicators (`hot`, `num_failed_logins`, `logged_in`, `num_compromised`)
  - Additional rates (`serror_rate`, `rerror_rate`, `srv_serror_rate`, `srv_rerror_rate`, `dst_host_serror_rate`, `dst_host_rerror_rate`)

- **Edge attributes:** Pairwise metrics between src↔dst, protocol↔service (bytes, durations, counts).

ALGORITHM & DEPLOYMENT

Training Process

1. **Contrastive Pre-training (50 epochs):**
    - **Augmentations:** Random node/edge dropout, feature masking, temporal noise.
    - **Loss:** Symmetric contrastive loss with temperature scaling and embedding-space regularization.
    - **Optimization:** AdamW with cosine-annealing LR scheduler.

2. **Fine-tuning (30 epochs):**
    - Freeze encoder except last GIN layer + its batch-norm.
    - Attach a 3-layer MLP classification head (cross-entropy loss).
    - Use step-LR scheduling and gradient clipping for stability.

Prediction Process

- **Graph Construction:** Given a new TCP/IP record, compute connection hash, encode categorical fields, build a 4-node graph with the same feature/edge schema.

RESULT

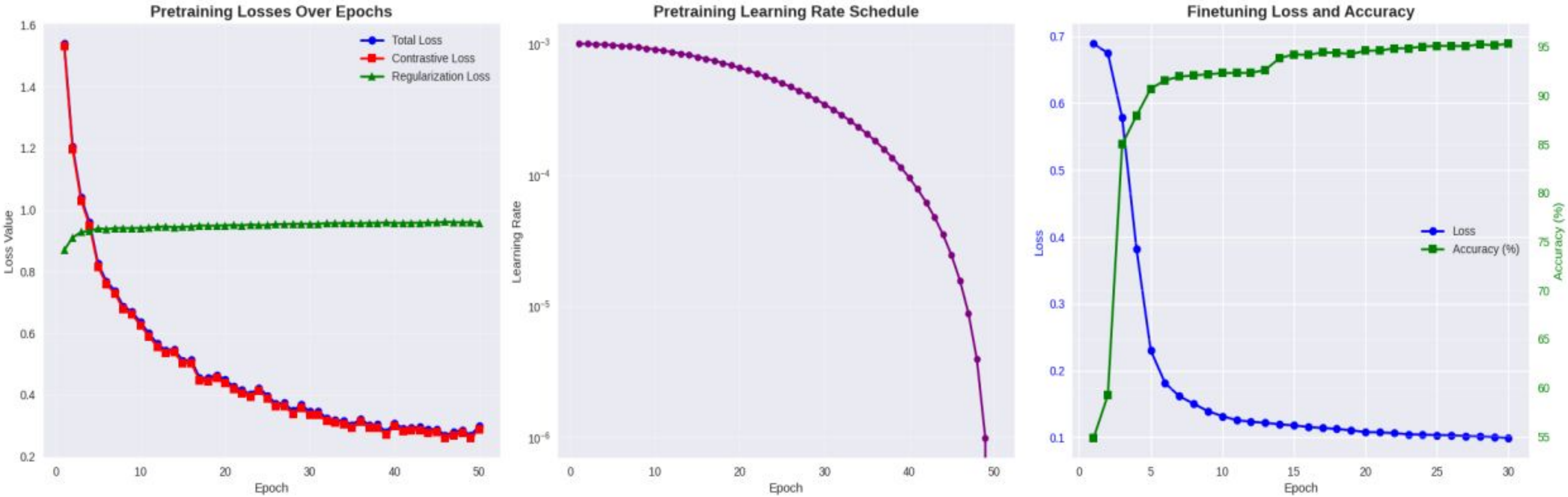
Test Performance

- **Accuracy:** 0.9598
- **Area Under ROC Curve (AUC):** 0.9909
- **F1-Score:** 0.9553

Effectiveness:

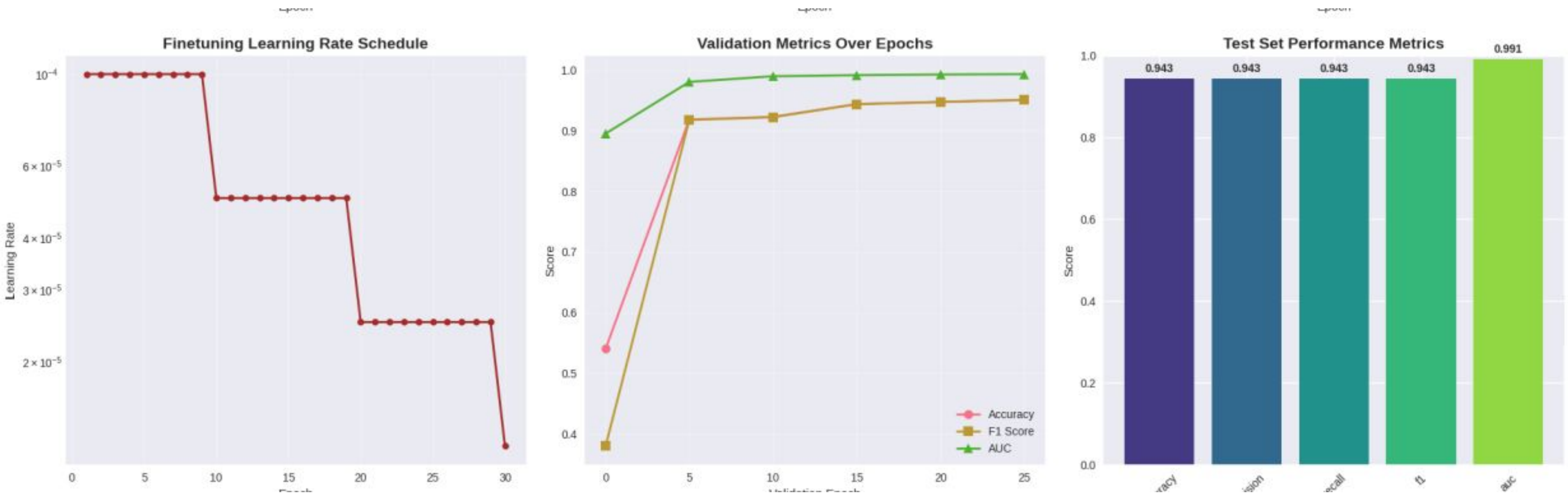
The contrastively pre-trained GNN encoder combined with a lightweight classification head delivers near-perfect discrimination of normal vs. anomalous network connections, achieving an AUC of 0.99 and F1 of 0.96 on held-out traffic.

RESULT



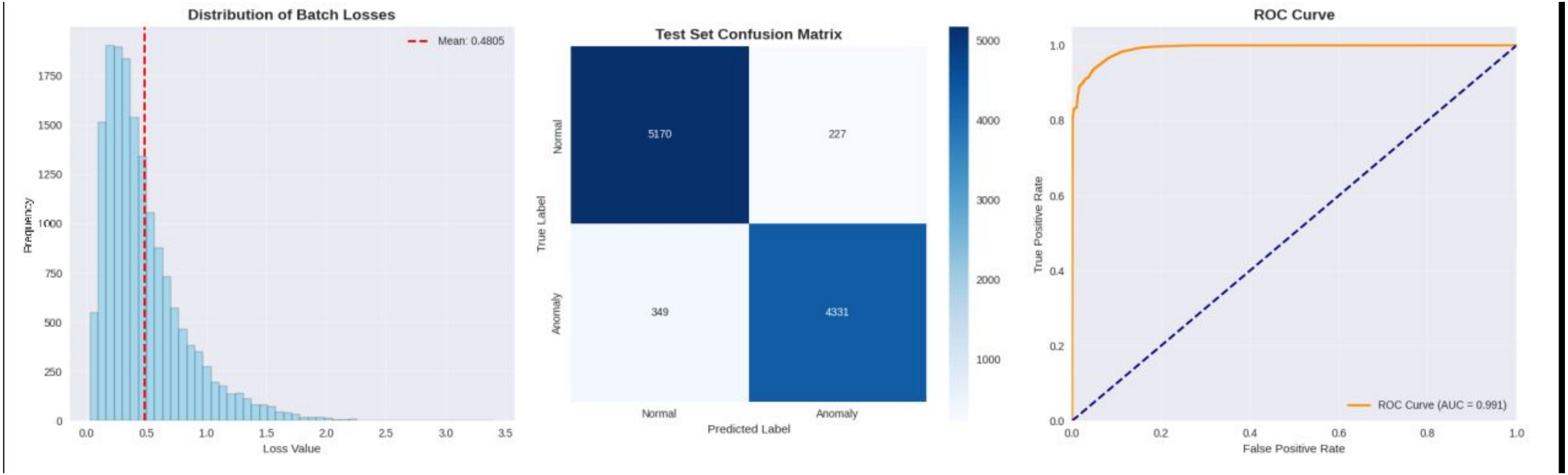
edunet foundation

RESULT



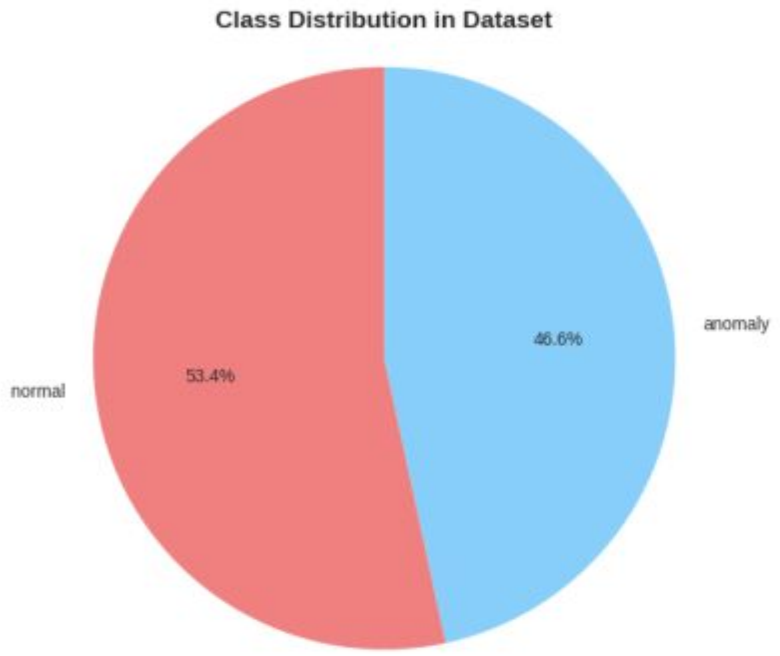
edunet
foundation

RESULT

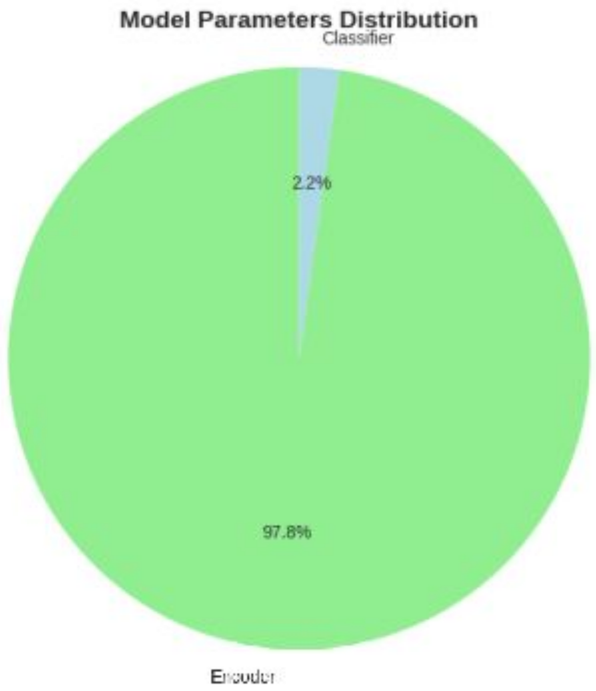


Distribution of Batch Losses
Mean: 0.4805
Frequency
Loss Value
Test Set Confusion Matrix
True Label
Predicted Label
Normal
Anomaly
5170
227
349
4331
ROC Curve
True Positive Rate
False Positive Rate
ROC Curve (AUC = 0.991)

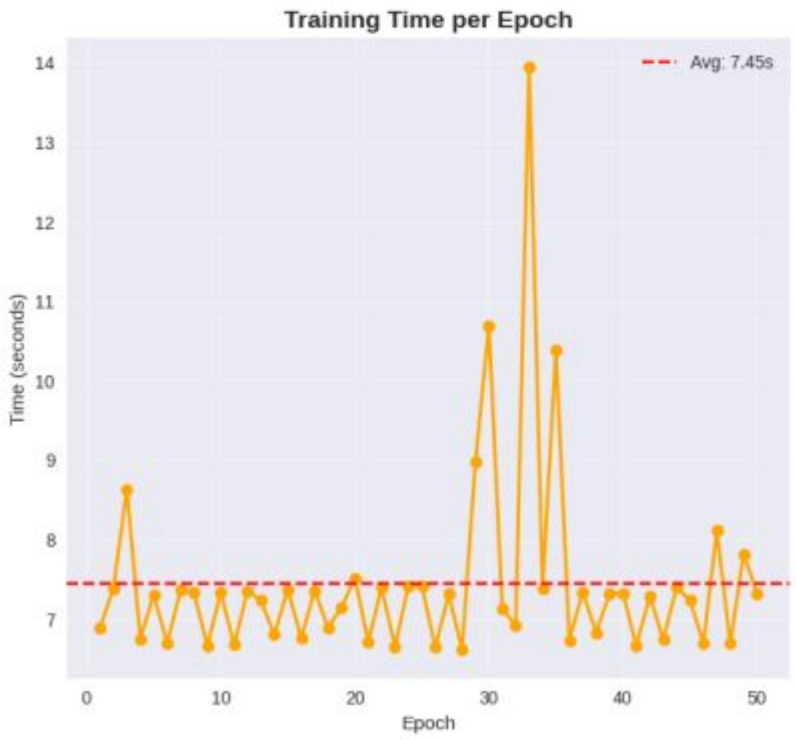
RESULT



Class Distribution in Dataset
normal
53.4%
46.6%
anomaly



Model Parameters Distribution
Classifier
2.2%
97.8%
Encoder



Training Time per Epoch
Avg: 7.45s
Time (seconds)
Epoch

CONCLUSION

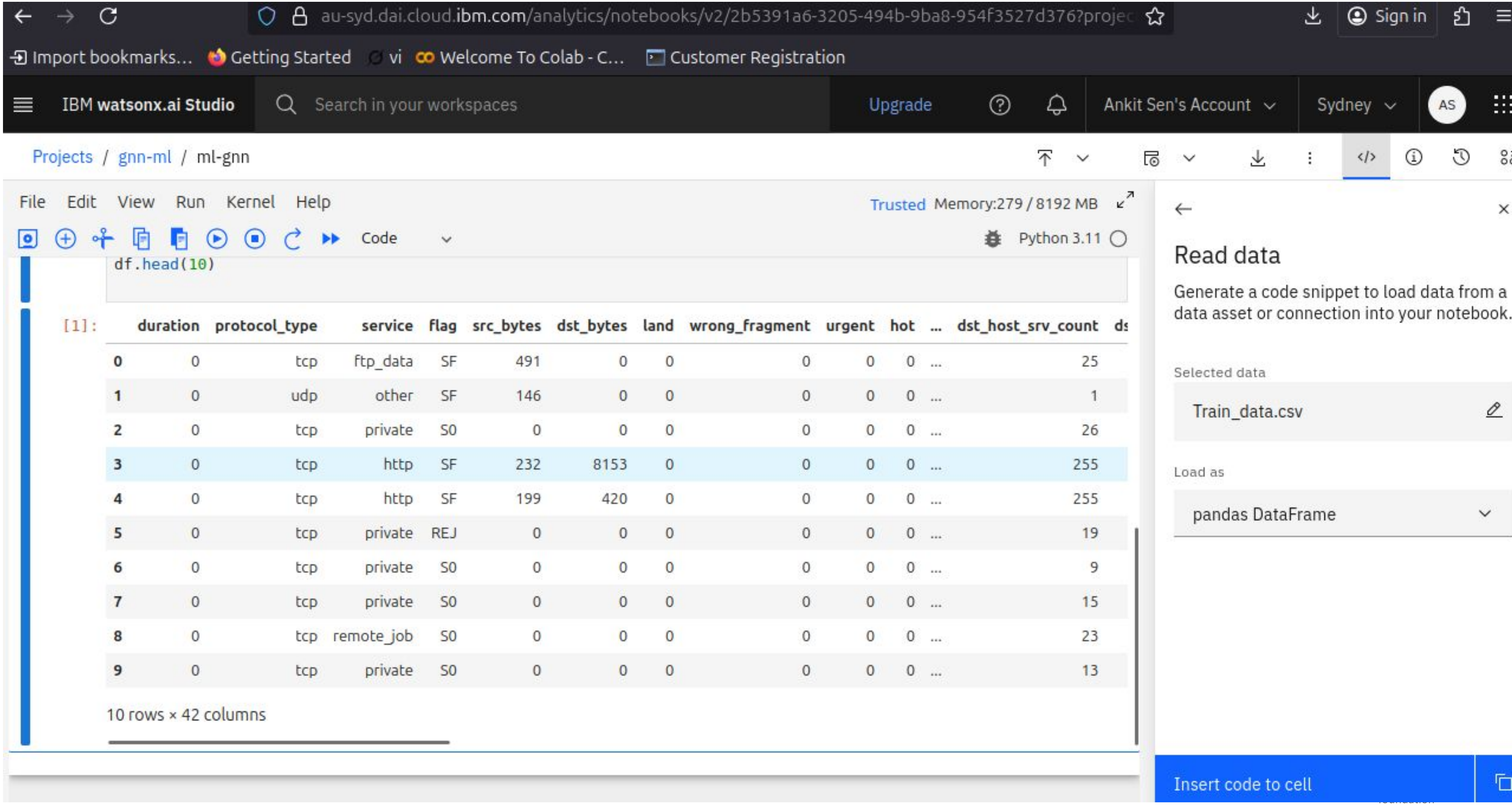
- **Summary of Findings**
  The contrastively pre-trained GNN encoder with a lightweight classification head achieved an AUC of 0.9909, F1-score of 0.9553, and accuracy of 0.9598 on the held-out test set, with zero data-leakage.
- **Effectiveness**
  Graph augmentations and attention pooling yielded robust, noise-resistant embeddings that accurately separate normal vs. anomalous traffic.
- **Challenges**
  Preprocessing per-connection graphs for ~25K samples was time-consuming, and tuning contrastive temperature, augmentation rates, and learning schedules required careful calibration under limited CPU memory.
- **Potential Improvements**
  Incorporating temporal graph edges, semi-supervised fine-tuning, and model pruning/ONNX conversion could further boost performance and enable real-time edge deployment.
- Accurate NIDS is essential for early breach detection, reducing false alarms, and safeguarding network integrity.

FUTURE SCOPE

- **Real-Time Edge Deployment:** Run the model on edge devices (e.g., routers or gateways) to detect attacks as they happen, reducing response time and network load.
- **Temporal Graph Modeling:** Add time-based edges or sequences of connections to capture how attacks evolve over time.
- **Semi-Supervised Learning:** Use unlabeled network data alongside labels to improve detection in low-data scenarios.
- **Model Compression:** Apply pruning or quantization to shrink the model for faster inference on limited-resource hardware.
- **Explainable AI:** Integrate techniques that highlight which nodes or features triggered an alarm, helping security teams understand and trust alerts.
- **Federated Learning:** Train models across multiple networks without sharing raw data, enhancing privacy and collaboration between organizations.
- **Advanced Augmentations:** Experiment with new graph augmentations (e.g., mixup or subgraph sampling) to make representations even more robust.

REFERENCES

- Denning, D. E. (1987). *An Intrusion-Detection Model.* IEEE Transactions on Software Engineering, SE-13(2), 222–232.
- Mukkamala, S., Janoski, G., & Sung, A. H. (2002). *Intrusion Detection Using Neural Networks and Support Vector Machines.* Proceedings of the International Joint Conference on Neural Networks (IJCNN), 1702–1707.
- Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2018). *How Powerful Are Graph Neural Networks?* International Conference on Learning Representations (ICLR).
- Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). *A Simple Framework for Contrastive Learning of Visual Representations (SimCLR).* International Conference on Machine Learning (ICML).
- Yousefi-Azar, M., & Azmi, R. (2020). *Network Intrusion Detection Using Graph Neural Networks.* arXiv:2007.01105.




```
df.head(10)
```

[1]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	ds
0	0	tcp	ftp_data	SF	491	0	0	0	0	0	...	25	
1	0	udp	other	SF	146	0	0	0	0	0	...	1	
2	0	tcp	private	S0	0	0	0	0	0	0	...	26	
3	0	tcp	http	SF	232	8153	0	0	0	0	...	255	
4	0	tcp	http	SF	199	420	0	0	0	0	...	255	
5	0	tcp	private	REJ	0	0	0	0	0	0	...	19	
6	0	tcp	private	S0	0	0	0	0	0	0	...	9	
7	0	tcp	private	S0	0	0	0	0	0	0	...	15	
8	0	tcp	remote_job	S0	0	0	0	0	0	0	...	23	
9	0	tcp	private	S0	0	0	0	0	0	0	...	13	

10 rows × 42 columns

au-syd.dai.cloud.ibm.com/analytics/notebooks/v2/2b5391a6-3205-494b-9ba8-954f3527d376?projec

Sign in

Import bookmarks... Getting Started vi Welcome To Colab - C... Customer Registration

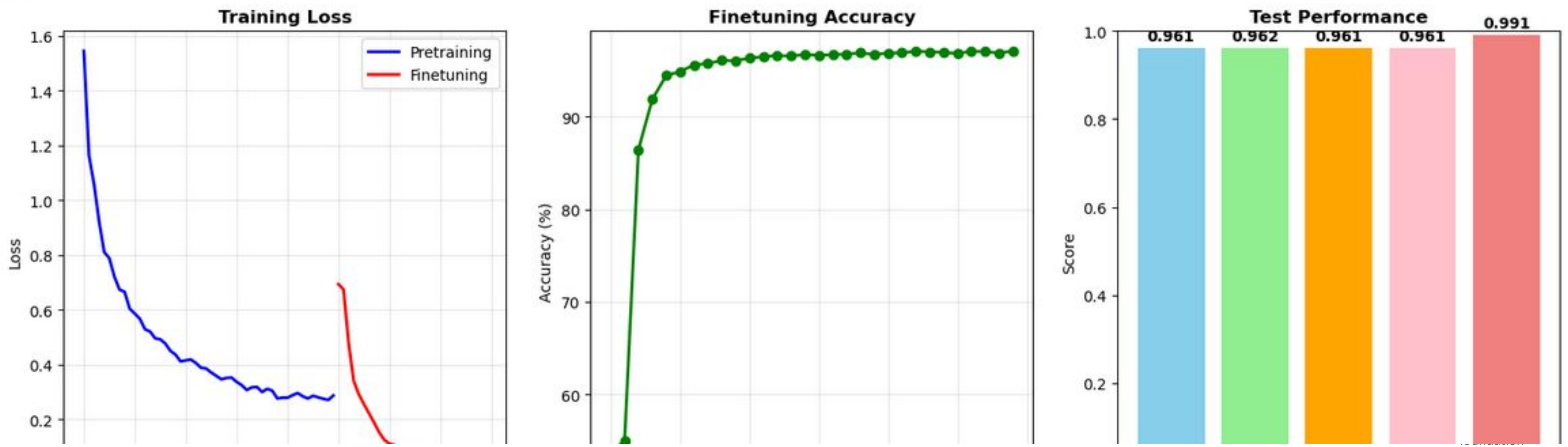
IBM watsonx.ai Studio Search in your workspaces Upgrade Ankit Sen's Account Sydney AS

Projects / gnn-ml / ml-gnn

File Edit View Run Kernel Help Trusted Memory:753 / 8192 MB

Code Python 3.11

```
print("=" * 50)
print(f"Final Test AUC: {training_history['test_metrics']['auc']:.4f}")
print(f"Final Test Accuracy: {training_history['test_metrics']['accuracy']:.4f}")
print(f"Final Test F1 Score: {training_history['test_metrics']['f1']:.4f}")
print(f"Pretraining Loss Reduction: {training_history['pretrain_losses'][0]:.4f} → {training_history['pretrain_losses'][-1]:.4f}")
print(f"Best Finetuning Accuracy: {max(training_history['finetune_accuracies']):.2f}%")
print("=" * 50)
```



Session 1 × +

ⓘ Current account: **Ankit Sen's Account**

```
meetpro70@cloudshell:~$ ls
contrastive-gnn-nids  hello  new
meetpro70@cloudshell:~$ cd contrastive-gnn-nids
meetpro70@cloudshell:~/contrastive-gnn-nids$ ls
LICENSE  README.md
meetpro70@cloudshell:~/contrastive-gnn-nids$ # Inside the contrastive-gnn-nids directory
meetpro70@cloudshell:~/contrastive-gnn-nids$ mkdir -p presentation images
meetpro70@cloudshell:~/contrastive-gnn-nids$
meetpro70@cloudshell:~/contrastive-gnn-nids$ touch .gitignore LICENSE README.md requirements.txt ml-gnn.ipynb \
>       presentation/slides.pptx presentation/report.pdf \
>       images/graph_architecture.png images/training_loss_plot.png \
>       images/test_results.png images/ibmcloud_run.png images/ibmcloud_notebook.png
meetpro70@cloudshell:~/contrastive-gnn-nids$ ls
LICENSE  README.md  images  ml-gnn.ipynb  presentation  requirements.txt
meetpro70@cloudshell:~/contrastive-gnn-nids$ rm ml-gnn.ipynb
meetpro70@cloudshell:~/contrastive-gnn-nids$ ls
LICENSE  README.md  images  presentation  requirements.txt
meetpro70@cloudshell:~/contrastive-gnn-nids$ ls
LICENSE  README.md  images  presentation  requirements.txt
meetpro70@cloudshell:~/contrastive-gnn-nids$ cd ..
meetpro70@cloudshell:~$ ls
contrastive-gnn-nids  hello  ml-gnn.ipynb  new
meetpro70@cloudshell:~$ mv ml-gnn.ipynb ~/contrastive-gnn-nids/
meetpro70@cloudshell:~$ cd contrastive-gnn-nids
meetpro70@cloudshell:~/contrastive-gnn-nids$ ls
LICENSE  README.md  images  ml-gnn.ipynb  presentation  requirements.txt
meetpro70@cloudshell:~/contrastive-gnn-nids$ cd ..
meetpro70@cloudshell:~$ mv P1.png P2.png P3.png ~/contrastive-gnn-nids/images/
meetpro70@cloudshell:~$ cd contrastive-gnn-nids/images/
meetpro70@cloudshell:~/contrastive-gnn-nids/images$ ls
P1.png  P2.png  P3.png  graph_architecture.png  ibmcloud_notebook.png  ibmcloud_run.png  test_results.png  training_loss_plot.png
meetpro70@cloudshell:~/contrastive-gnn-nids/images$ rm graph_architecture.png  ibmcloud_notebook.png  ibmcloud_run.png  test_results.png  training_loss_plot.png
meetpro70@cloudshell:~/contrastive-gnn-nids/images$ ls
P1.png  P2.png  P3.png
meetpro70@cloudshell:~/contrastive-gnn-nids/images$
```

IBM CERTIFICATIONS

In recognition of the commitment to achieve professional excellence

Getting Started with Artificial Intelligence

Ankit Sen

Has successfully satisfied the requirements for:

Getting Started with Artificial Intelligence

Issued on: Jul 16, 2025
Issued by: IBM SkillsBuild

Verify: https://www.credly.com/badges/6b2f2337-1935-4149-802d-dc161c031321

IBM

IBM CERTIFICATIONS

In recognition of the commitment to achieve professional excellence

Journey to Cloud: Envisioning Your Solution

Ankit Sen

Has successfully satisfied the requirements for:

Journey to Cloud: Envisioning Your Solution

Issued on: Jul 18, 2025
Issued by: IBM SkillsBuild

Verify: https://www.credly.com/badges/2bfd3510-df26-46d5-a3ad-f365cfe8cde9

IBM

IBM CERTIFICATIONS

IBM **SkillsBuild** Completion Certificate

This certificate is presented to

Ankit Sen

for the completion of

Lab: Retrieval Augmented Generation with LangChain

(ALM-COURSE_3824998)

According to the Adobe Learning Manager system of record

Completion date: 18 Jul 2025 (GMT)

Learning hours: 20 mins

GITHUB LINK

https://github.com/ankitsencode123/contrastive-gnn-nids.git

THANK YOU