BITS F464: Machine Learning- Course Project

Machine Learning for Anomaly Detection

Group ML06

April 22, 2022

1 Motivation

An anomaly is something that differs from a norm: a deviation, an exception. Anomaly detection is an important tool for detecting fraud, network intrusion, and other rare events that may have great significance but are hard to find. In the context of healthcare, for example, this can be used to alert health practitioners to anomalous physiological data that can be indicative of health complications. However, it is tedious to build an anomaly detection system by hand. This requires domain knowledge and foresight. For an ecosystem where the data changes over time, like fraud,this cannot be a good solution. Machine learning, then, suits the purpose of creating an Anomaly Detection system that is adaptive and on time, and can handle large datasets.

2 Abstract

Generative adversarial networks (GANs) are able to model the complex high dimensional distributions of real-world data, which suggests they could be effective for anomaly detection. Intuitively, a GAN that has been trained to fit the distribution of normal samples should be able to reconstruct such a normal sample from a latent representation while also distinguishing the sample as coming from the genuine data distribution. However, because GANs only implicitly model data distribution, using them to detect anomalies requires a time-consuming optimization technique to recover the latent representation of a specific input example, making this an unfeasible solution for big datasets or real-time applications. In this project, we first implement a GAN-based network for anomaly detection. We then attempt to improve upon this model, through optimizing hyperparameters during training. We then consider another GAN-based model of adversarially learned anomaly detection. We show that our improved model yields better results than the original model, in terms of precision, recall, F1 score, and mean inference time.

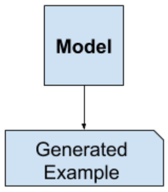
3 Introduction

The selected research paper is **Efficient GAN-based Anomaly Detection**, authored by **Zenati et al.**[1] . Anomaly detection is a critical issue in a variety of fields, including manufacturing, medical imaging, and cyber-security. Anomaly detection approaches, at their most basic level, must simulate the distribution of normal data, which can be complex and high-dimensional. One family of models that have been effectively utilised to model such complicated and high-dimensional distributions, particularly over natural photos, is generative adversarial networks. In this paper, we use recently discovered GAN approaches to construct an anomaly detection system that is efficient at test time by simultaneously learning an encoder during training. Intuitively, a GAN that has been trained to fit the distribution of normal samples should be able to reconstruct such a normal sample from a latent representation while also distinguishing the sample as coming from the genuine data distribution. However, because GANs only implicitly model data distribution, using them to detect anomalies requires a time-consuming optimization technique to recover the latent representation of a specific input example, making this an unfeasible solution for big datasets or real-time applications. In this paper, we use recently discovered GAN approaches to construct an anomaly detection system that is efficient at test time by simultaneously learning an encoder during training.

4 Implementation Details

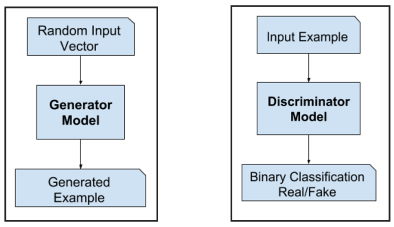
4.1 Generative Modeling

It refers to the unsupervised models that take a set of input variables and summarise them that can generate new examples of the input distribution. The model can produce new variables that can plausibly fit into the distribution of the input variables.

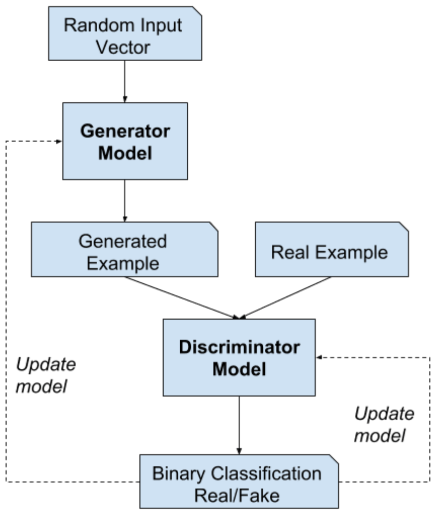


4.2 Generative Adversarial Networks

These are deep learning-based generative models and involve two sub-models, a generator and a discriminator model. The generator model helps in generating new examples while its counterpart discriminator helps in classifying whether the given example is real (produced from the domain) or fake (generated by the generator model)



The generator and discriminator models are trained jointly. The generator creates a batch of samples, which are then given to the discriminator, along with real instances from the domain, to be classified as real or false. In the next round, the discriminator is modified to improve its ability to distinguish between real and false samples, while the generator is updated based on how effectively the generated samples deceived the discriminator.



4.3 GAN Model

The authors focused on building GAN Models which comprised of an encoder E, Generator G and discriminator D. The encoder E mapped the input symbols x to a latent representation z, the use of the encoder in such a way helps us save time by trying to recover a latent representation during testing. The authors based on past research in this field also decided to learn E jointly with G and the training strategies to learn encoder such that $E = G^{-1}$. Hence we need to $min_{G,E} MAX_D V(D, E, G)$ where V(D,E,G) is $E_{x-pE}[E_{z-pE}(.|x)[logD(z, x)]] + E_{z-pz}[E_{x-pG}(.|z)[1 - logD(x, z)]]$.

4.4 Datasets Used

Image Dataset: The MNIST database (Modified National Institute of Standards and Technology database) is a massive library of handwritten digits that is often used for image processing system training. In the field of machine learning, the database is also commonly utilised for training and testing. There are 60,000 training images and 10,000 testing images in the MNIST database.
Network Intrusion Dataset: This is the data set for the Third International Knowledge Discovery and Data Mining Tools Competition, held in connection with KDD-99, the Fifth International Conference on Knowledge Discovery and Data Mining. The goal of the competition was to create a network intrusion detector, which was a prediction model capable of distinguishing between "bad" connections (intrusions or attacks) and "good" normal connections. This database comprises a standard set of auditable data, including a wide range of intrusions simulated in a military network.

5 Results of Original Implementation

Loss Variant	Precision	Recall	F1	Mean Inference Time (s)
Cross Entropy	0.8637	0.8775	0.8705	0.010
Feature Matching	0.8636	0.8774	0.8704	0.014

Table: Performance on the KDD99 dataset for the label 0

Epoch	Loss Gen	Loss Enc	Loss Dis
0	0.5415	113.2680	22.5615
1	0.2061	184.2287	7.5680
2	0.3426	193.7989	7.6567
3	1.0237	163.7737	1.1112
4	2.1272	155.5335	0.4264
5	2.3886	172.9662	0.3005
6	1.3845	181.7575	2.4365
7	1.5083	145.3106	0.5634
8	2.0974	131.2692	0.3146
9	2.1685	133.9415	0.3141

Table: Feature Matching results for epoch=10 and label=0

Epoch	Loss Gen	Loss Enc	Loss Dis
0	1.0724	163.2685	0.8603
1	2.1178	129.4127	0.3055
2	1.9659	142.8006	0.8206
3	2.3728	127.5226	0.2969
4	2.6355	117.7211	0.2522
5	3.2236	115.8845	0.1811
6	3.5860	119.5238	1.1956
7	4.0558	117.7322	0.1080
8	4.1398	121.5964	0.1172
9	4.3147	120.7257	0.0746

Table: Cross Entropy results for epoch=10 and label=0

6 Proposed Improvements

For the given paper, we have chosen to implement two kinds of improvements. First, we attempt to alter the hyperparameters of the GAN model presented in the paper 'Efficient GAN-based Anomaly Detection'[1]. Next, we go a step further by instead choosing to deal with the anomaly detection problem using an entirely different model. This section will describe the above-mentioned proposed improvements in greater detail, while the next section will deal with their results.

6.1 Experiment: Hyperparameter Optimization

GAN models can suffer badly in the following areas comparing to other deep networks.

- Non-convergence: the models do not converge and worse they become unstable.
- Mode collapse: the generator produces limited modes, and
- Slow training: the gradient to train the generator vanished.

An improvement we propose is related to hyperparameter optimization of the neural network attributes. Optimizers allow us to alter these attributes (such as weights, learning rate) in an attempt to increase the accuracy of the model and reduce losses. These optimizers typically aim to minimize the cost function associated with a neural network. The Starting learning rate parameter of the neural network that the paper implements have been optimized. The learning rate is a hyperparameter that governs how much the model changes each time the model weights are changed in response to the predicted error. A too-high learning rate can cause the model to converge too rapidly to a suboptimal solution, while a too-low learning rate can cause the process to stall.

6.2 Adversarially Learned Anomaly Detection (ALAD) Model

While GAN-based anomaly detection models have been shown to achieve state-of-the-art performance, along with increased usage in several real-life applications, one of the problems that has been observed in GANs is that these models become ineffective at test time, making them less than ideal for larger, more complex datasets. One method that has been proposed to increase the efficacy of the GAN models in anomaly detection is adversarially learned anomaly detection (ALAD) [2], and this is the model that we choose to implement as our second improvement. This method is based on bi-directional GANs that, while in the training phase, learn an encoder network simultaneously, which has the effect of making the model as a whole faster during the testing phase. We implement and test this model on the KDDCup99 dataset [ref] that was used in the original paper.

7 Results and Findings

7.1 Results of Hyperparameter Optimization

The table below shows the performance of the GAN model on the KDD99 dataset in terms of precision, recall, F1 score, and mean inference time for varied values of the learning rate.

Learning rate	Precision	Recall	F1 score	Mean Inference Time (s)
1e-05	0.6857	0.6966	0.6911	0.8120
1e-04	0.3738	0.3798	0.3768	1.1221
1e-03	0.5375	0.5510	0.6993	0.9442
1e-02	0.5347	0.7718	0.6986	1.1409

Table: Performance on the KDD99 dataset for different learning rate values

According to our experiments, the learning rate value of 1e-05 is optimal for the GAN model on the KDD99 dataset in terms of Precision, Recall and F1 score. These scores decrease significantly on increasing the learning rate parameter. Additionally, the mean inference time increases on increasing the learning rate parameter.

7.2 Results of ALAD Model

The table below shows the average performance of the ALAD model on the KDD99 dataset in terms of precision, recall, F1 score, and mean inference time.

Loss Variant	Precision	Recall	F1 score	Mean Inference Time (s)
Cross-Entropy	0.9076	0.9220	0.9148	0.0037
Feature Matching	0.7016	0.9286	0.7993	0.0039

Table: Performance on the KDD99 dataset for the label 0

Tables depicting the loss per epoch for the same are given below.

Epoch	Loss Gen	Loss Enc	Loss Dis
0	81.6233	158.4915	46.2228
1	70.9358	140.7554	43.1075
2	52.6017	131.5104	44.9686
3	79.5772	154.2546	46.5407
4	65.4283	139.6308	51.8580
5	65.6082	132.7184	41.3788
6	70.5473	139.4371	40.4915
7	63.6778	130.5292	47.4383
8	63.8568	129.1542	39.9017
9	53.5889	113.9814	35.2062

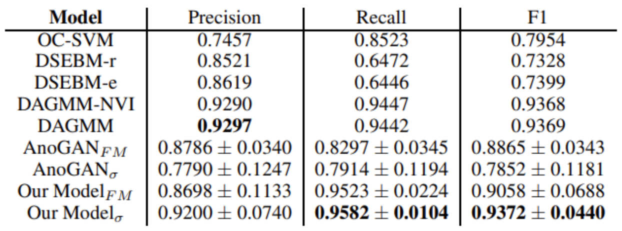
Table: Cross Entropy results for epoch=10 and label=0

Epoch	Loss Gen	Loss Enc	Loss Dis
0	73.8997	152.6351	43.4227
1	75.8413	146.7532	31.9663
2	64.1203	140.3860	32.1118
3	79.3102	155.6217	48.5419
4	59.3165	134.3129	57.7683
5	64.8907	140.8899	43.9632
6	65.1841	137.0153	43.0497
7	61.8988	133.9729	44.7495
8	61.2568	134.2238	40.8981
9	57.7515	122.6137	35.3598

Table: Feature Matching results for epoch=10 and label=0

8 Conclusions

The usage of GAN models has helped us get better performance on complex datasets compared to other anomalies detection models like OC-SVM, DSEBM-r, DSEBM-e, DAGMM-NVI and DAGMM. The technique to learn the encoder simultaneously helped in reduced the computation required to recover the latent representation. We can see from this table as seen in the original paper, that the cross entropy model is able to give better recall and F1 compared to other models and comparable precision (with respect to DAGMM).



Model	Precision	Recall	F1
OC-SVM	0.7457	0.8523	0.7954
DSEBM-r	0.8521	0.6472	0.7328
DSEBM-e	0.8619	0.6446	0.7399
DAGMM-NVI	0.9290	0.9447	0.9368
DAGMM	**0.9297**	0.9442	0.9369
AnoGAN$_{FM}$	0.8786 ± 0.0340	0.8297 ± 0.0345	0.8865 ± 0.0343
AnoGAN$_{\sigma}$	0.7790 ± 0.1247	0.7914 ± 0.1194	0.7852 ± 0.1181
Our Model$_{FM}$	0.8698 ± 0.1133	0.9523 ± 0.0224	0.9058 ± 0.0688
Our Model$_{\sigma}$	0.9200 ± 0.0740	$\mathbf{0.9582 \pm 0.0104}$	$\mathbf{0.9372 \pm 0.0440}$

Performance on the KDD99 dataset. Values for OC-SVM, DSEBM, DAGMM

8.1 Comparisons

ALAD uses reconstruction errors based on these adversarially learned features to determine if a data sample is anomalous. ALAD builds on recent advances to ensure data-space and latent-space cycle-consistencies and stabilize GAN training, which results in significantly improved anomaly detection performance. First, we observe that ALAD appears to be several hundred-fold faster at test time than the original GAN model. This can be proved by looking at the mean inference time in both the results.

Loss Variant	Original MIT (s)	Improved MIT (s)
Cross-Entropy	0.010	0.0037
Feature Matching	0.014	0.0039

Precision can be seen as a measure of quality, and recall as a measure of quantity. Higher precision means that an algorithm returns more relevant results than irrelevant ones.

Loss Variant	Original Precision	Improved Precision
Cross-Entropy	0.8637	.9076
Feature Matching	0.8636	0.7016

High recall means that an algorithm returns most of the relevant results. There is significant improvement in both these metrics when we utilize the ALAD model, over the original GAN model.

Loss Variant	Original Recall	Improved Recall
Cross-Entropy	0.8775	0.9220
Feature Matching	0.8774	0.9286

We also see that the F1 score is much closer to 1 (a perfect F1 score) in our proposed solution than it was in the original implementation. A good F1 score corresponds to low false positives and low false negatives, meaning real threats are correctly identified, without being disrupted by false alarms.

Loss Variant	Original F1	Improved F1
Cross-Entropy	0.8705	0.9148
Feature Matching	0.8704	0.7993

-

9 Work Distribution

Complete Team and Their Contribution:

Adithya Jayan Warrier	2018B3A70873H	GAN Model Implementation
Devanshi Gupta	2019A7PS1265H	Results and Findings
Grahithaa Sarathy	2018B3A70895H	ALAD Model Implementation
Saloni Singh	2019A7PS1261H	Hyperparameter Optimization
Vinita Bhat	2019A7PS1206H	Report Writing

-

10 References

(1) Zenati, H., Foo, C., Lecouat, B., Manek, G., Chandrasekhar, V.R. (2018). Efficient GAN-Based Anomaly Detection. ArXiv, abs/1802.06222.
(2) Zenati, H., Romain, M., Foo, C., Lecouat, B., Chandrasekhar, V.R. (2018). Adversarially Learned Anomaly Detection. 2018 IEEE International Conference on Data Mining (ICDM), 727-736.
-

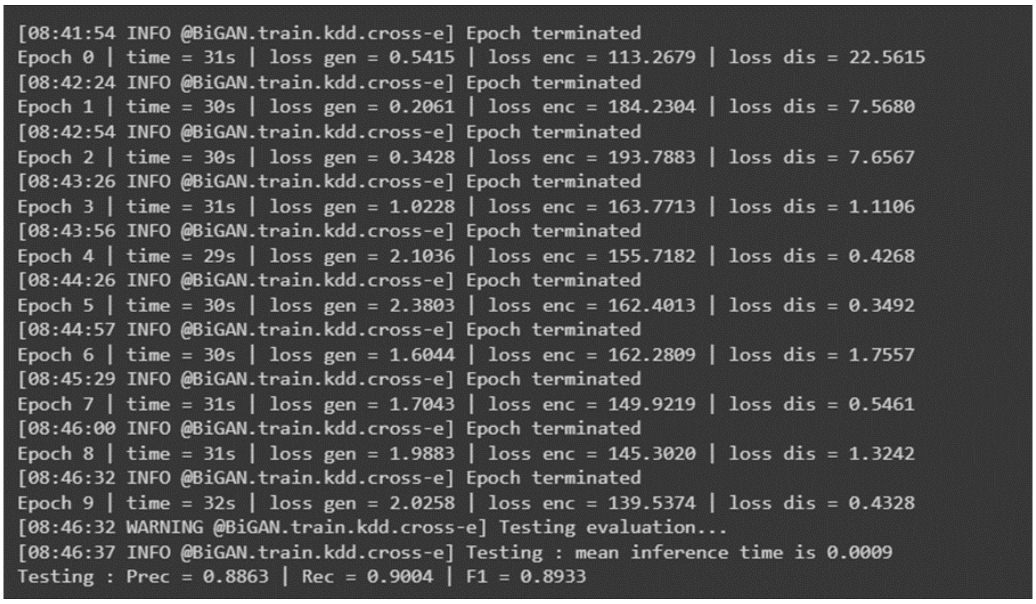
11 Appendix

11.1 Code

The code used in this project can be found at https://github.com/adithyawarrier2000/ML-Project

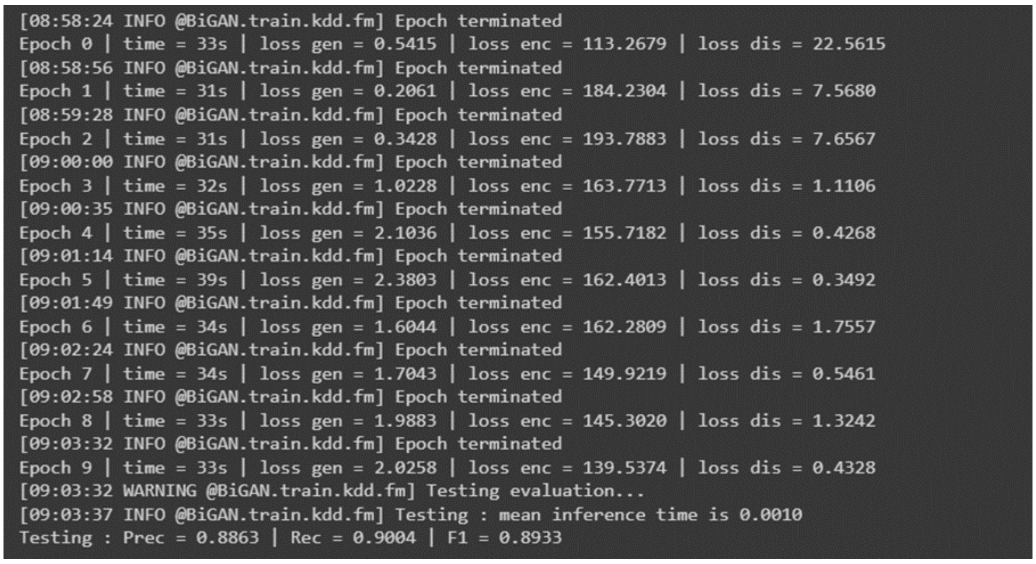
11.2 Selected Screenshots

Below are screenshots of from the running of the implemented models (both original and improved), for different loss variants.



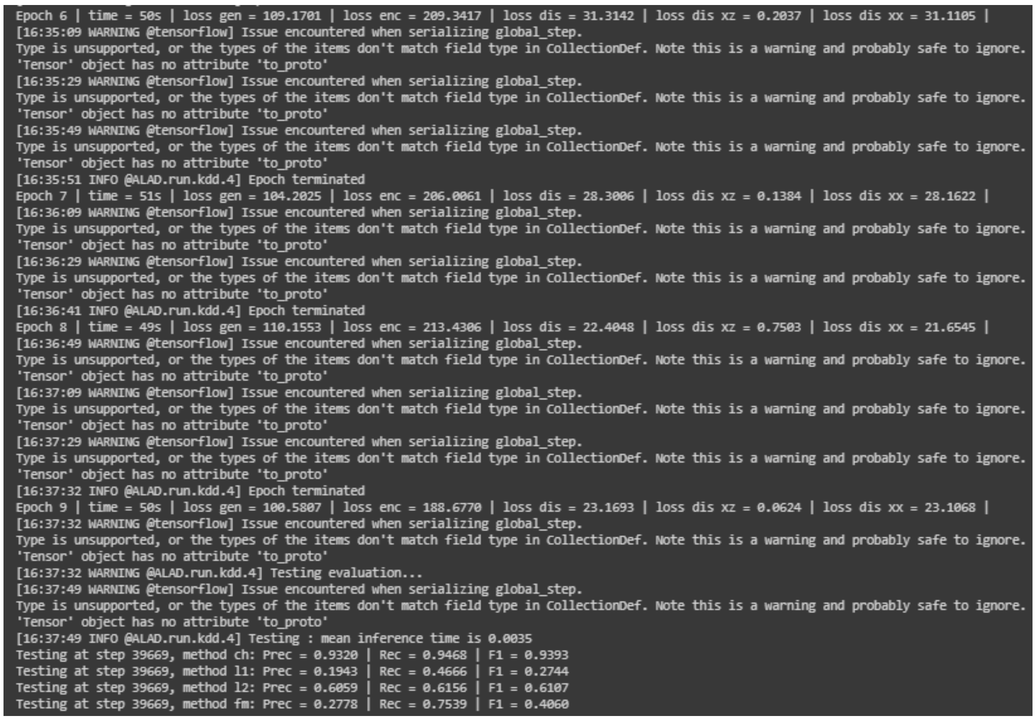
```
[08:41:54 INFO @BiGAN.train.kdd.cross-e] Epoch terminated
Epoch 0 | time = 31s | loss gen = 0.5415 | loss enc = 113.2679 | loss dis = 22.5615
[08:42:24 INFO @BiGAN.train.kdd.cross-e] Epoch terminated
Epoch 1 | time = 30s | loss gen = 0.2061 | loss enc = 184.2304 | loss dis = 7.5680
[08:42:54 INFO @BiGAN.train.kdd.cross-e] Epoch terminated
Epoch 2 | time = 30s | loss gen = 0.3428 | loss enc = 193.7883 | loss dis = 7.6567
[08:43:26 INFO @BiGAN.train.kdd.cross-e] Epoch terminated
Epoch 3 | time = 31s | loss gen = 1.0228 | loss enc = 163.7713 | loss dis = 1.1106
[08:43:56 INFO @BiGAN.train.kdd.cross-e] Epoch terminated
Epoch 4 | time = 29s | loss gen = 2.1036 | loss enc = 155.7182 | loss dis = 0.4268
[08:44:26 INFO @BiGAN.train.kdd.cross-e] Epoch terminated
Epoch 5 | time = 30s | loss gen = 2.3803 | loss enc = 162.4013 | loss dis = 0.3492
[08:44:57 INFO @BiGAN.train.kdd.cross-e] Epoch terminated
Epoch 6 | time = 30s | loss gen = 1.6044 | loss enc = 162.2809 | loss dis = 1.7557
[08:45:29 INFO @BiGAN.train.kdd.cross-e] Epoch terminated
Epoch 7 | time = 31s | loss gen = 1.7043 | loss enc = 149.9219 | loss dis = 0.5461
[08:46:00 INFO @BiGAN.train.kdd.cross-e] Epoch terminated
Epoch 8 | time = 31s | loss gen = 1.9883 | loss enc = 145.3020 | loss dis = 1.3242
[08:46:32 INFO @BiGAN.train.kdd.cross-e] Epoch terminated
Epoch 9 | time = 32s | loss gen = 2.0258 | loss enc = 139.5374 | loss dis = 0.4328
[08:46:32 WARNING @BiGAN.train.kdd.cross-e] Testing evaluation...
[08:46:37 INFO @BiGAN.train.kdd.cross-e] Testing : mean inference time is 0.0009
Testing : Prec = 0.8863 | Rec = 0.9004 | F1 = 0.8933
```

Figure 1: Original Model: kdd, anomalous label 4, cross-entropy



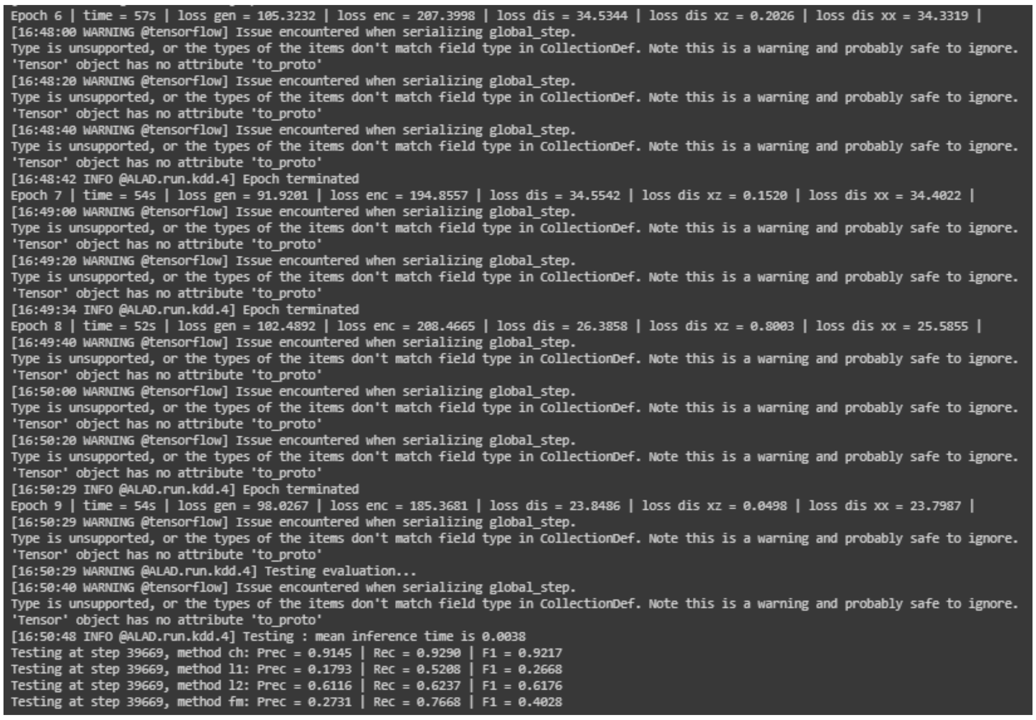
```
[08:58:24 INFO @BiGAN.train.kdd.fm] Epoch terminated
Epoch 0 | time = 33s | loss gen = 0.5415 | loss enc = 113.2679 | loss dis = 22.5615
[08:58:56 INFO @BiGAN.train.kdd.fm] Epoch terminated
Epoch 1 | time = 31s | loss gen = 0.2061 | loss enc = 184.2304 | loss dis = 7.5680
[08:59:28 INFO @BiGAN.train.kdd.fm] Epoch terminated
Epoch 2 | time = 31s | loss gen = 0.3428 | loss enc = 193.7883 | loss dis = 7.6567
[09:00:00 INFO @BiGAN.train.kdd.fm] Epoch terminated
Epoch 3 | time = 32s | loss gen = 1.0228 | loss enc = 163.7713 | loss dis = 1.1106
[09:00:35 INFO @BiGAN.train.kdd.fm] Epoch terminated
Epoch 4 | time = 35s | loss gen = 2.1036 | loss enc = 155.7182 | loss dis = 0.4268
[09:01:14 INFO @BiGAN.train.kdd.fm] Epoch terminated
Epoch 5 | time = 39s | loss gen = 2.3803 | loss enc = 162.4013 | loss dis = 0.3492
[09:01:49 INFO @BiGAN.train.kdd.fm] Epoch terminated
Epoch 6 | time = 34s | loss gen = 1.6044 | loss enc = 162.2809 | loss dis = 1.7557
[09:02:24 INFO @BiGAN.train.kdd.fm] Epoch terminated
Epoch 7 | time = 34s | loss gen = 1.7043 | loss enc = 149.9219 | loss dis = 0.5461
[09:02:58 INFO @BiGAN.train.kdd.fm] Epoch terminated
Epoch 8 | time = 33s | loss gen = 1.9883 | loss enc = 145.3020 | loss dis = 1.3242
[09:03:32 INFO @BiGAN.train.kdd.fm] Epoch terminated
Epoch 9 | time = 33s | loss gen = 2.0258 | loss enc = 139.5374 | loss dis = 0.4328
[09:03:32 WARNING @BiGAN.train.kdd.fm] Testing evaluation...
[09:03:37 INFO @BiGAN.train.kdd.fm] Testing : mean inference time is 0.0010
Testing : Prec = 0.8863 | Rec = 0.9004 | F1 = 0.8933
```

Figure 2: Original Model: kdd, anomalous label 4, feature matching



```
Epoch 6 | time = 50s | loss gen = 109.1701 | loss enc = 209.3417 | loss dis = 31.3142 | loss dis xz = 0.2037 | loss dis xx = 31.1105 |
[16:35:09 WARNING @tensorflow] Issue encountered when serializing global_step.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Tensor' object has no attribute 'to_proto'
[16:35:29 WARNING @tensorflow] Issue encountered when serializing global_step.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Tensor' object has no attribute 'to_proto'
[16:35:49 WARNING @tensorflow] Issue encountered when serializing global_step.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Tensor' object has no attribute 'to_proto'
[16:35:51 INFO @ALAD.run.kdd.4] Epoch terminated
Epoch 7 | time = 51s | loss gen = 104.2025 | loss enc = 206.0061 | loss dis = 28.3006 | loss dis xz = 0.1384 | loss dis xx = 28.1622 |
[16:36:09 WARNING @tensorflow] Issue encountered when serializing global_step.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Tensor' object has no attribute 'to_proto'
[16:36:29 WARNING @tensorflow] Issue encountered when serializing global_step.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Tensor' object has no attribute 'to_proto'
[16:36:41 INFO @ALAD.run.kdd.4] Epoch terminated
Epoch 8 | time = 49s | loss gen = 110.1553 | loss enc = 213.4306 | loss dis = 22.4048 | loss dis xz = 0.7503 | loss dis xx = 21.6545 |
[16:36:49 WARNING @tensorflow] Issue encountered when serializing global_step.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Tensor' object has no attribute 'to_proto'
[16:37:09 WARNING @tensorflow] Issue encountered when serializing global_step.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Tensor' object has no attribute 'to_proto'
[16:37:29 WARNING @tensorflow] Issue encountered when serializing global_step.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Tensor' object has no attribute 'to_proto'
[16:37:32 INFO @ALAD.run.kdd.4] Epoch terminated
Epoch 9 | time = 50s | loss gen = 100.5807 | loss enc = 188.6770 | loss dis = 23.1693 | loss dis xz = 0.0624 | loss dis xx = 23.1068 |
[16:37:32 WARNING @tensorflow] Issue encountered when serializing global_step.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Tensor' object has no attribute 'to_proto'
[16:37:32 WARNING @ALAD.run.kdd.4] Testing evaluation...
[16:37:49 WARNING @tensorflow] Issue encountered when serializing global_step.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Tensor' object has no attribute 'to_proto'
[16:37:49 INFO @ALAD.run.kdd.4] Testing : mean inference time is 0.0035
Testing at step 39669, method ch: Prec = 0.9320 | Rec = 0.9468 | F1 = 0.9393
Testing at step 39669, method l1: Prec = 0.1943 | Rec = 0.4666 | F1 = 0.2744
Testing at step 39669, method l2: Prec = 0.6059 | Rec = 0.6156 | F1 = 0.6107
Testing at step 39669, method fm: Prec = 0.2778 | Rec = 0.7539 | F1 = 0.4060
```

Figure 3: Improved Model: kdd, anomalous label 4, cross-entropy



```
Epoch 6 | time = 57s | loss gen = 105.3232 | loss enc = 207.3998 | loss dis = 34.5344 | loss dis xz = 0.2026 | loss dis xx = 34.3319 |
[16:48:00 WARNING @tensorflow] Issue encountered when serializing global_step.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Tensor' object has no attribute 'to_proto'
[16:48:20 WARNING @tensorflow] Issue encountered when serializing global_step.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Tensor' object has no attribute 'to_proto'
[16:48:40 WARNING @tensorflow] Issue encountered when serializing global_step.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Tensor' object has no attribute 'to_proto'
[16:48:42 INFO @ALAD.run.kdd.4] Epoch terminated
Epoch 7 | time = 54s | loss gen = 91.9201 | loss enc = 194.8557 | loss dis = 34.5542 | loss dis xz = 0.1520 | loss dis xx = 34.4022 |
[16:49:00 WARNING @tensorflow] Issue encountered when serializing global_step.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Tensor' object has no attribute 'to_proto'
[16:49:20 WARNING @tensorflow] Issue encountered when serializing global_step.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Tensor' object has no attribute 'to_proto'
[16:49:34 INFO @ALAD.run.kdd.4] Epoch terminated
Epoch 8 | time = 52s | loss gen = 102.4892 | loss enc = 208.4665 | loss dis = 26.3858 | loss dis xz = 0.8003 | loss dis xx = 25.5855 |
[16:49:40 WARNING @tensorflow] Issue encountered when serializing global_step.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Tensor' object has no attribute 'to_proto'
[16:50:00 WARNING @tensorflow] Issue encountered when serializing global_step.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Tensor' object has no attribute 'to_proto'
[16:50:20 WARNING @tensorflow] Issue encountered when serializing global_step.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Tensor' object has no attribute 'to_proto'
[16:50:29 INFO @ALAD.run.kdd.4] Epoch terminated
Epoch 9 | time = 54s | loss gen = 98.0267 | loss enc = 185.3681 | loss dis = 23.8486 | loss dis xz = 0.0498 | loss dis xx = 23.7987 |
[16:50:29 WARNING @tensorflow] Issue encountered when serializing global_step.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Tensor' object has no attribute 'to_proto'
[16:50:29 WARNING @ALAD.run.kdd.4] Testing evaluation...
[16:50:40 WARNING @tensorflow] Issue encountered when serializing global_step.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Tensor' object has no attribute 'to_proto'
[16:50:48 INFO @ALAD.run.kdd.4] Testing : mean inference time is 0.0038
Testing at step 39669, method ch: Prec = 0.9145 | Rec = 0.9290 | F1 = 0.9217
Testing at step 39669, method l1: Prec = 0.1793 | Rec = 0.5208 | F1 = 0.2668
Testing at step 39669, method l2: Prec = 0.6116 | Rec = 0.6237 | F1 = 0.6176
Testing at step 39669, method fm: Prec = 0.2731 | Rec = 0.7668 | F1 = 0.4028
```

Figure 4: Improved Model: kdd, anomalous label 4, feature matching