EXPLORING THE LATENT SPACES INDUCED BY A WASSERSTEIN GAN

Eric Stenton

Dr. Pablo Rivas, Faculty Advisor

Spring 2020

Submitted to the Honors Council in partial fulfillment of the requirements for the degree of Honors in Liberal Arts

Honors Program – Thesis
Marist College – Poughkeepsie, NY

Exploring the Latent Spaces Induced by a Wasserstein GAN

Eric Stenton

Received: April/10/2020 / Accepted by Advisor: April/24/2020

Abstract Predicting if freshmen students will drop out of college or transfer to another is often difficult due to limited and anomalous data. This paper explores using Generative Adversarial Networks (GANs) to learn the general features of student data and uses it to produce predictions with higher accuracy and lower false positive rates than neural networks trained with traditional techniques. Here we examine the differences between a classifier's latent space when it is trained with a GAN architecture versus traditionally for predicting if a freshman student will leave Marist College within their first year. Our experimental results suggest that GANs are an alternative to training neural models for student dropout/transfer prediction.

Keywords GAN · Classification · Anomaly Detection · Latent Spaces

E. Stenton
Tel.: +1 (845) 575-3000 Ext. 2601
E-mail: Eric.Stenton1@Marist.edu

Honors Thesis Advisor: **Pablo Rivas, Ph.D.**
School of Computer Science and Mathematics
Tel.: +1 (845) 575-3000 Ext. 2086
E-mail: Pablo.Rivas@Marist.edu

1 Introduction

Most colleges want to retain the number of freshman students enrolled and do what they can to prevent them from leaving within the first year. We will use the word ‘attrition’ to describe students who have either dropped out or transferred to another college. A strong tool in lowering the amount of student attrition is the ability to predict who will leave, as well determine a trend or commonality between those who do leave. An inevitable problem with developing a good manner of prediction is the small amount of data that is available as a result of a typically small incoming class and the even smaller amount of those who leave. In other words, predicting student attrition in the first year can be proposed as an anomaly detection problem with a very limited amount of data to use in creating prediction models. In this paper, the freshman population of Marist College of years 2016 and 2017 will be examined using a GAN architecture in order to predict attrition in 2018. First, the neural network model learns the characteristics of a first-year student through adversarial learning. Second, the model is fine-tuned to classify students as either those who will stay or those who will leave. Third, the latent space of the layer directly before the final one that gives the final prediction is inspected for comparing three versions of the model. The versions are the following: The model traditionally trained without a GAN (the control), one adversarially trained without tuning, and one adversarially trained with tuning. The hypothesis is that the model that is adversarially trained with tuning will have a latent space more representative of the freshman population producing a higher accuracy when predicting student attrition.

The following section will provide a brief background of the concepts in this paper. Following this section will be a description of the methodology used to test the models and how the models were built. The next section will be an overview of the three experiments performed, their accompanying diagrams, and a short explanation of the results. Finally, the last section will be a concluding paragraph on the findings of the experiments.

2 Background and Other Work

It is important to note this paper serves as an extension of research carried out by Dr. Eitel Lauria and colleagues in which the same population of students was used to predict attrition using multiple machine learning algorithms, the primary one being XGBoost [3]. Dr. Lauria's research produced models with accurate predictions of student attrition despite minimal amounts of data. This research extends the knowledge of neural models for student attrition introduced by E. Lauria et al [8].

Current insights in GAN architectures originated in a paper by Dr. Ian Goodfellow et al. where the concept of a discriminator model and generator model playing a minimax game first arose [5]. Their paper shows the following value function for how the GAN operates:

$$\min_{G} \max_{D} V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \quad (1)$$

In the value function *V(D, G)*, *G* is a differential function representing the generator model that takes noise input *Pz(z)* and maps it to a data space. This data space is meant to represent possible values that can mimic variables *Pdata(x)*, real data, when inputted into another function represented by the discriminator model and denoted as *D* that outputs a prediction of whether the input was generated or not. *D* is trained to maximize the probability of correctly labeling generated and real samples while *G* is trained to minimize *log(1 - D(G(z)))*, or lower the probability of *D* predicting correctly.

Shortly after Dr. Goodfellow's paper, the structure of the GAN training python code and the calculation of both the Wasserstein loss and gradient penalty for the training of the discriminator originated in an experiment from a paper by Martin Arjovsky, Soumith Chintala, and Lon Bottou [1]. The formula for the Wasserstein distance which is described in further detail in the referenced paper, is the following:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \gamma}[\|x - y\|] \quad (2)$$

In the Wasserstein distance equation, $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ represents the set of all joint distributions $\gamma(x, y)$ with marginals $\mathbb{P}_r$ and $\mathbb{P}_g$ respectively. In order to transform distributions $\mathbb{P}_r$ into distribution $\mathbb{P}_g$, $\gamma(x, y)$ denotes the amount of 'mass' to be transported from x to y while the Wasserstein distance describes the 'cost' of the optimal method of transport.

The next section will describe the methodology for building the discriminator and generator models as well as how the Wasserstein distance equation will be utilized.

3 Methodology

The main pieces of GAN architectures are the discriminator and generator models as shown in Equation 2. These models will be explained in this section in detail.

3.1 Discriminator

The discriminator is a neural model composed of 12 layers as shown in Figure 1. These layers are: dropout, ReLU, batch normalization, tanh, and sigmoid. First, in order to prevent any one feature of the input data becoming heavily weighted, the dropout layer disconnects about 20% of the features randomly on each training step [10]. Second, batch normalization layers are placed intermittently to prevent the outputs of the ReLU layers from becoming too large and slowing or preventing convergence [4]. Third, the Python implementation of our model is based on Keras' functional model due to its ability to work with the tanh layer separately as this will serve as a view into the latent space of the model directly before an output is computed. Fourth, the discriminator's loss is based on weighing two Wasserstein loss calculations with a weight of one and a gradient penalty with a weight of ten.

3.2 Generator

The generator is a sequential model made up of 9 layers with a similar layout to the discriminator in which it has ReLU layers with intermittent batch normalization layers and an output consisting of a sigmoid layer as shown in Figure 2. The most notable difference the generator has from the discriminator is the nature of its input which is 12 normally distributed random values between 0 and 1. These values are 'noise' or values within a latent dimension defining different vectors that will eventually become generated data mimicking the input to the discriminator. This latent dimension should not be confused with the latent space referenced in this paper describing the output of the tanh layer in the discriminator. Furthermore, the generator's loss function is simpler than the discriminator's as it only consists of a single Wasserstein calculation.

The following section will present the three experiments conducted using the aforementioned models in detail as well as expound on the results of each.

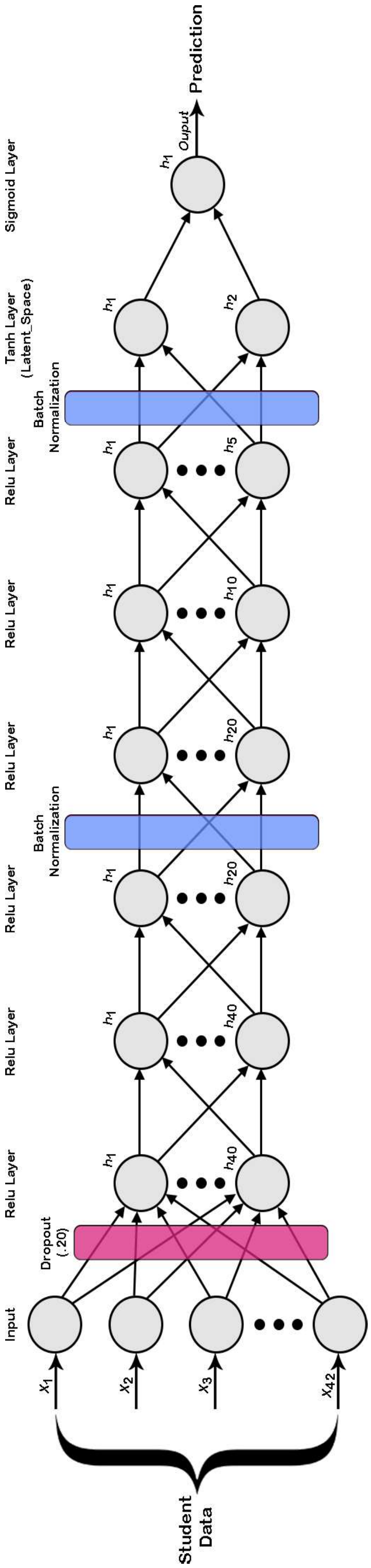


Fig. 1: Discriminator Architecture Diagram

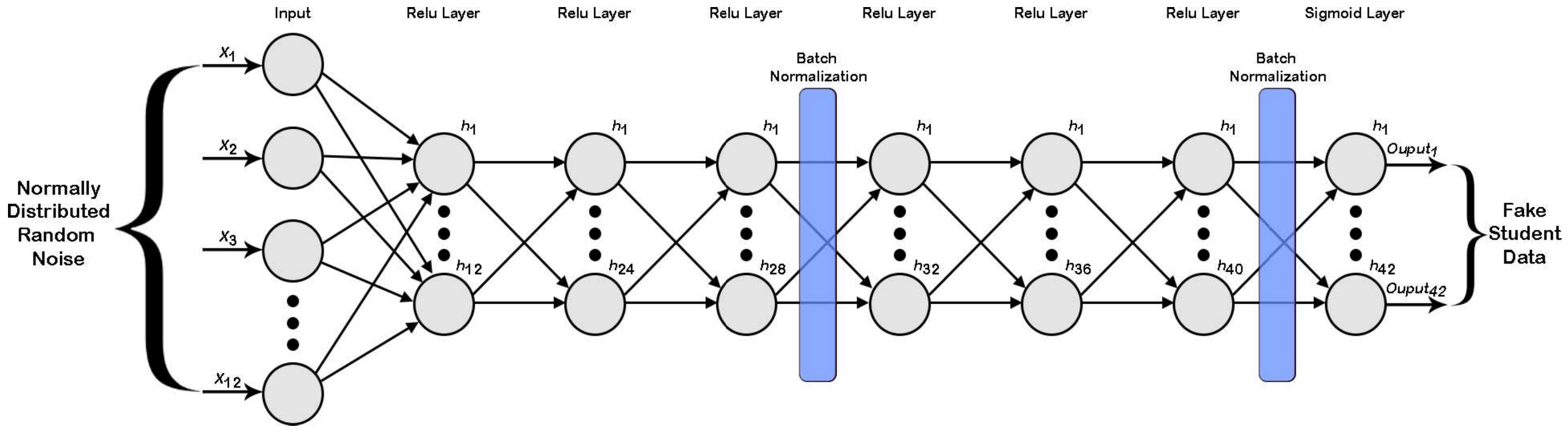


Fig. 2: Generator Architecture Diagram

4 Experiments and Results

Before getting into the details of the experiment, let us take a look at the input data. Table 1 describes the features and corresponding data types.

Table 1: Description of Predictors

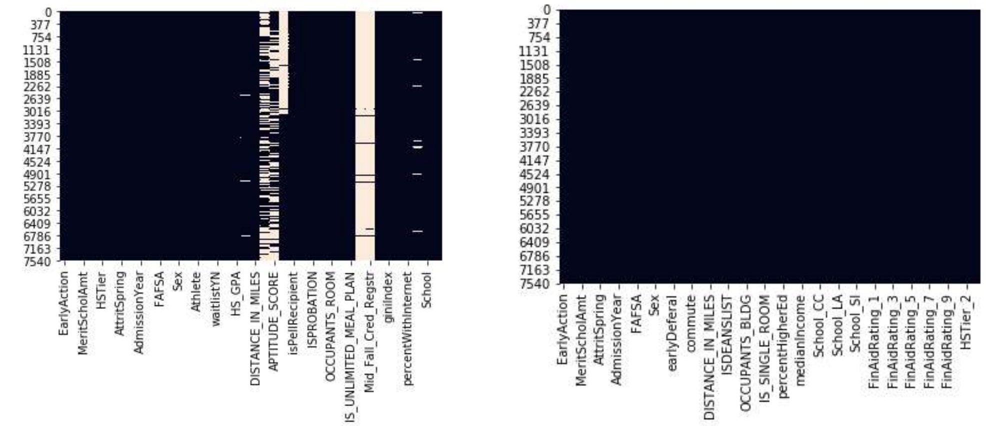
Feature	Description	Data Type
EarlyAction	Applied for early action	Binary (1/0)
EarlyDecision	Applied for early decision	Binary (1/0)
MeritScholAmt	Merit scholarship amount awarded	Binary (1/0)
FinAidRating	Financial aid rating	Categorical encoded as binary (1,0)
HSTier	High School Tier	Categorical encoded as binary (1,0)
Foreign	Foreign student	Binary (1/0)
FAFSA	Applied for Federal Student Aid	Binary (1/0)
APCourses	Took AP courses	Binary (1/0)
Sex	The sex of the student	Binary (1/0)
Athlete	Is a student athlete	Binary (1/0)
EarlyDeferral	Applied for early deferral	Binary (1/0)
WaitlistYN	Was waitlisted	Binary (1/0)
Commute	Is a commuter student	Binary (1/0)
HSGPA	High School GPA	Integer
DistanceInMiles	Distance from home (miles)	Integer
School	Member of a certain school, eg., CC (ComSci & Math)	Categorical encoded as binary (1,0)
IsPellRecipient	Is recipient of Pell Grant	Binary (1/0)
IsDeansList	Joined Dean's List	Binary (1/0)
IsProbation	Is on probabtion	Binary (1/0)
OccupantsBuilding	Number of occupants in dorm	Integer
OccupantsRoom	Number of occupants in dorm room	Integer
IsSingleRoom	Uses a single room	Binary (1/0)
IsUnlimitedMealPlan	Has unlimited meal plan	Binary (1/0)
PercentHigherEd	Percent of those with higher education in home area	Float
GiniIndex	Gini Index value of home area	Float
MedianIncome	Median income of home area	Float
PercentWithInternet	Percent with internet in home area	Float
Attrited (Target)	Left the college	Binary (1/0)

Some of the most noteworthy predictors in Table 1 are the following: 'HSGPA', 'DistanceInMiles', 'MeritScholAmt', and 'APCourses'. 'HSGPA' is a student's GPA from high school measured with a 4.0 scale. 'DistanceInMiles' is the distance from a student's hometown to the college measured in miles. 'MeritScholAmt' is the amount of money awarded to the student through a merit scholarship. Finally, the 'APCourses' feature is a binary value where 1 means the student has taken AP courses and 0 means they have not. It is important to note that the majority of the aforementioned predictors relate to how well the student has done academically in high school. Furthermore, the 'DistanceInMiles' predictor may indirectly relate to the student's emotional well-being as a larger distance away from their hometown may limit visits home. However, due to the difficulty in measuring the importance of predictors in a neural network, the speculation on the impact each feature has on predicting student attrition is rooted in the work by E. Lauria et al. where many of the same predictors are used and measured based on their importance in multiple machine learning models [8].

Besides the most important predictors, it is also imperative to point out the most 'noisy' predictors, or those that have a large number of null values, which are the following: 'DistanceInMiles', 'OccupantsBuilding', 'OccupantsRoom', and 'GiniIndex'. As mentioned previously, 'DistanceInMiles' is the amount of miles between the college and the student's hometown. 'OccupantsBuilding' is the number of students that live in a student's dorm building. Similarly, 'OccupantsRoom' is the number of students that live within the student's dorm room including themselves. Last, 'GiniIndex' is the Gini coefficient of the student's hometown which is a measurement of income distribution in the area where a high value indicates greater inequality. In order to handle these features, the data is cleaned.

Our method of preprocessing the data includes removing any feature that is comprised of more than 30% of nulls and imputing the remaining features with missing values using K nearest neighbors (KNN). The results of preprocessing null values can be seen in Figure 3. Additionally, the preprocessing step also included normalizing values between 0 and 1 for all integer and float type features. All categorical features mentioned in Table 1 are dummified.

After preprocessing the data, it is used to perform three experiments as described in the next few sections.



(a) Before Data Preprocessing

(b) After Data Preprocessing

Fig. 3: Heatmap of Nulls in Data

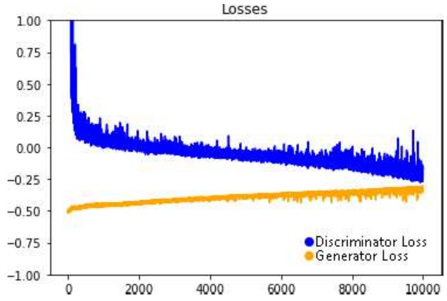
4.1 Experiment 1

In the first experiment, the GAN model was trained for 10,000 epochs. The weights were then transferred to two models, one that is tuned for 500 epochs to classify student attrition and the other that is left alone. This transference of weights is an example of transfer learning where the knowledge gained through adversarial training is applied to predicting student attrition (further details can be found in the referenced work) [6]. A control model was made from the same architecture as the GAN one, but trained separately on only the data previously used to tune for classification for 500 epochs. The results are shown in Figure 4. As we can see from the Receiver Operating Characteristic (ROC) diagrams, the control model performed marginally better with an accuracy of .68 (e) than the tuned GAN model with only .64 accuracy (g). A ROC curve is a plot of the true positive rate against the false positive rate across various thresholds that determine the dividing line between classifications for a given model (more info in the provided reference) [2]. The accuracy of the GAN model, before tuning, is extremely low at .42 (c). However, it displays a latent space (d) containing the data in what seems to be a linear relationship. Note that the orange dots represent students who left the college while purple dots are the students who stayed. It is important to also note the discriminator and generator loss converging at about 10,000 epochs (a), or around the amount of epochs this experiment ran.

Directing our attention to the Cohen's kappa statistics table (b), we can see that the figure relating the control and tuned model shows a kappa value of 0.5301 when a threshold resulting in about a 5% error rate is used. This value could be in the range of -1 to 1 and shows how close the model's outputs are where 1 is identical and anything 0 or below is akin to equivalent by chance [7]. The formula for the Cohen's kappa coefficient is the following:

$$\kappa = (p_o - p_e) \,/\, (1 - p_e) \tag{3}$$

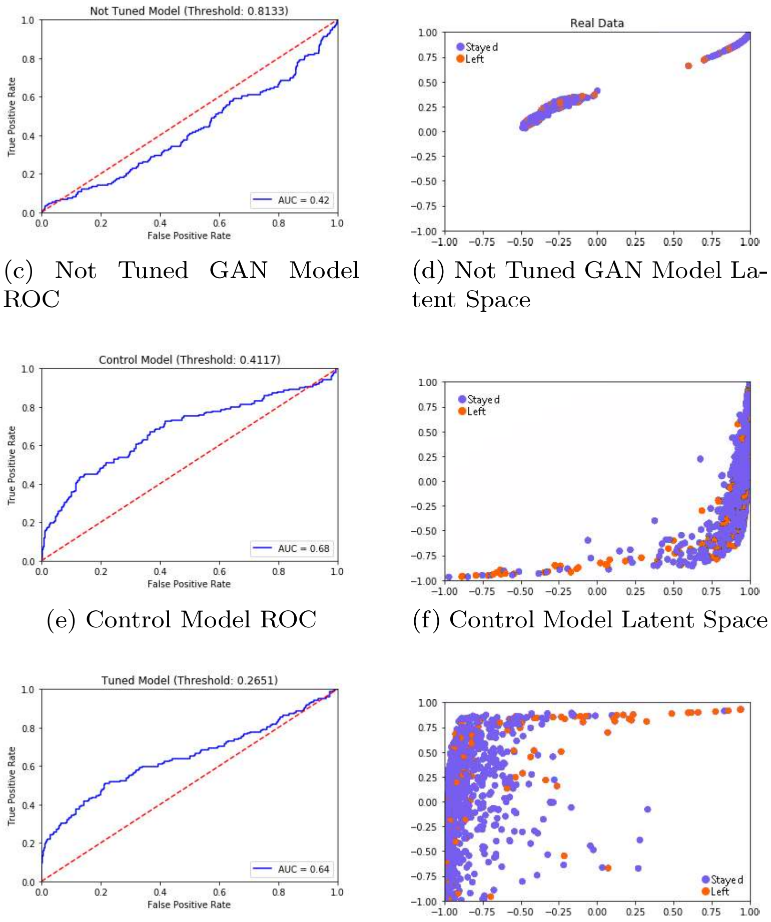
The p_o variable in the equation is the observed agreement of the labels applied to a sample by the models while p_e is the probability of chance agreement. The aforementioned value 0.5301 demonstrates that the control and tuned models for this experiment are outputting predictions that are similar but also having a good number of discrepancies. The fact that they are different suggest that the models are fundamentally different in their output distributions which is desired. In the second experiment, we will see how the Cohen's kappa coefficients change.



(a) Discriminator & Generator Loss

	Not Tuned	Control	Tuned
Not Tuned		-0.0136	-0.0029
Control	-0.0136		0.5301
Tuned	-0.0029	0.5301	
Actual Target	0.0298	0.2694	0.2492

(b) Cohen's Kappa Statistics



(c) Not Tuned GAN Model ROC

(d) Not Tuned GAN Model Latent Space

(e) Control Model ROC

(f) Control Model Latent Space

(g) Tuned Model ROC

(h) Tuned Model Latent Space

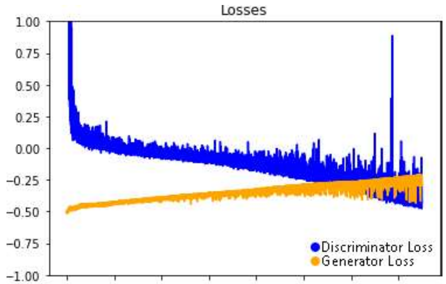
Fig. 4: Experiment 1 Training Results

4.2 Experiment 2

In the second experiment, the GAN model trained for 15,000 epochs. The results are shown in Figure 5. Here, we can see that the discriminator and generator losses have converged and begun separating again though on inverse sides (a). The GAN model, before tuning, still demonstrates a low accuracy (c) and a latent space with a similar linear relationship as in experiment 1 (d). The control model's accuracy remains at about .68 with 500 epochs of training (e). It is here that we see an improvement in the accuracy of the tuned model boasting a .69 which is .05 higher than its previous (g). Last, the kappa value (b) for the control and tuned model is 0.4426 which is noteworthily lower than in the first experiment when ran with a threshold resulting in about a 5% false positive rate despite the overall accuracy of the two models being different by a 0.01 margin. This means that despite their close accuracies, the two models are providing differing outputs which suggests the two models are correctly classifying students the other is misidentifying. The third and final experiment will demonstrate what happens to the Cohen's kappa coefficient when the accuracy of the tuned model is higher than the control model.

4.3 Experiment 3

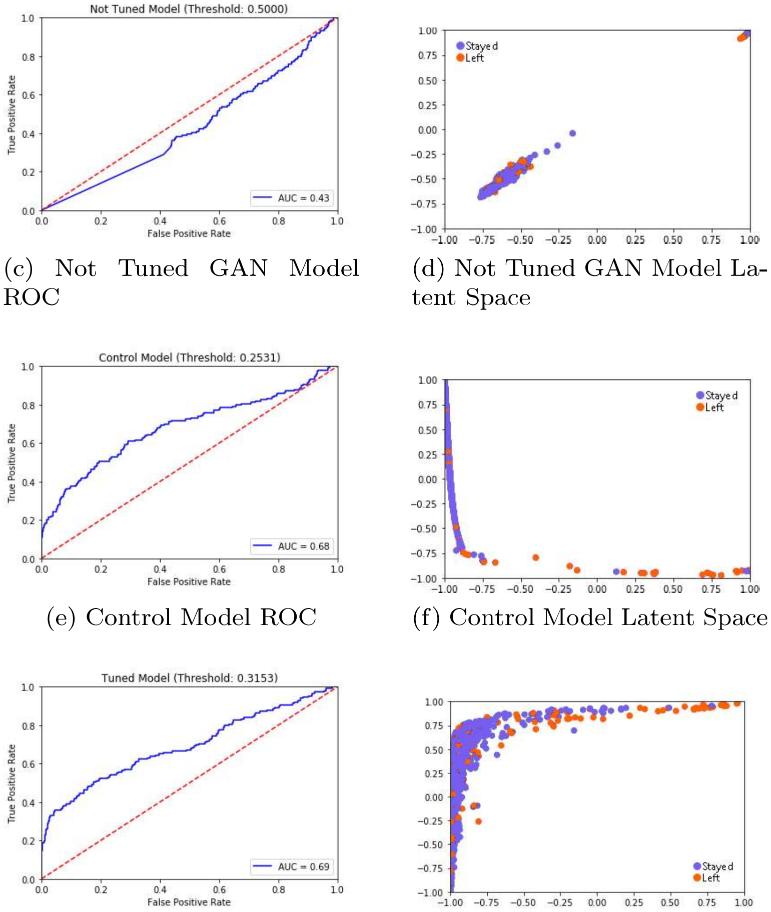
In the third experiment, the GAN model trained for 20,000 epochs. The results are shown in Figure 6. As shown in the discriminator and generator loss diagram, the losses converged, separated, and continued to grow apart though on inverse sides to where they began (a). Again, the GAN model, before tuning, shows a low accuracy (c) though now with a noticeably different latent space (d) that seems to still have some semblance of a linear relationship with a high amount of data clumping at the bottom left corner and some at the top right corner. The control model was able to reach an accuracy of .69 with 500 epochs of training (e). However, it is still .01 below the tuned model in this experiment as it reached a .70 accuracy (g). Both the control model (f) and tuned model (h) latent spaces show groupings of attrited students in the higher right corner suggesting there may be a correlation in the predictors for these cases. Finally, taking a look at the kappa score (b) of the control and tuned models predicting at a threshold resulting in about a 5% false positive rate will reveal a higher value than the previous two experiments. Here, we see that their output similarity is measured to be a 0.6358 kappa score. This increase in the kappa score is expected since both models have an increased accuracy from the previous two experiments which naturally leads to their outputs being similar as they both are making more correct predictions. While this value is higher than in experiment 1 and 2, it still demonstrates the predictions of the two models show a noteworthy degree of discrepancy and produce different output distributions. Next, we will examine the results of the three experiments.



(a) Discriminator & Generator Loss

	Not Tuned	Control	Tuned
Not Tuned		-0.0386	-0.0193
Control	-0.0386		0.4426
Tuned	-0.0193	0.4426	
Actual Target	-0.0239	0.2722	0.3078

(b) Cohen's Kappa Statistics



(c) Not Tuned GAN Model ROC

(d) Not Tuned GAN Model Latent Space

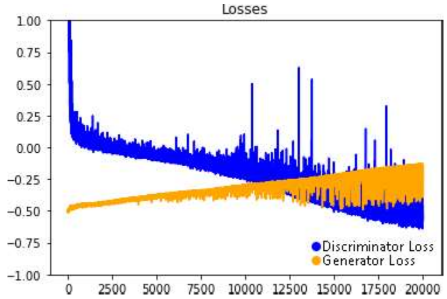
(e) Control Model ROC

(f) Control Model Latent Space

(g) Tuned Model ROC

(h) Tuned Model Latent Space

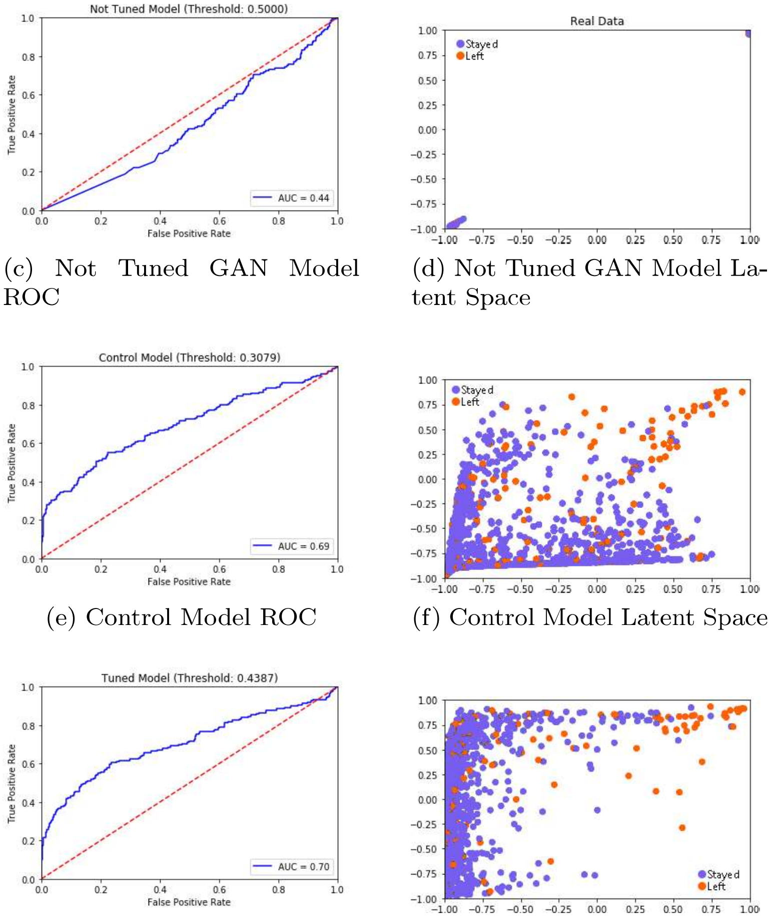
Fig. 5: Experiment 2 Training Results



(a) Discriminator & Generator Loss

	Not Tuned	Control	Tuned
Not Tuned		-0.0462	-0.0631
Control	-0.0462		0.6358
Tuned	-0.0631	0.6358	
Actual Target	-0.0239	0.2899	0.3545

(b) Cohen's Kappa Statistics



(c) Not Tuned GAN Model ROC

(d) Not Tuned GAN Model Latent Space

(e) Control Model ROC

(f) Control Model Latent Space

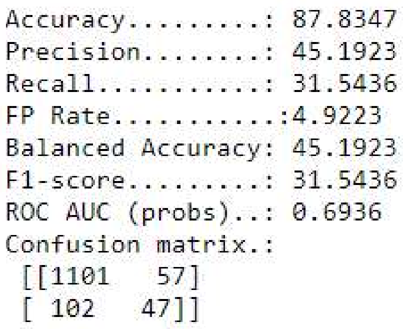
(g) Tuned Model ROC

(h) Tuned Model Latent Space

Fig. 6: Experiment 3 Training Results

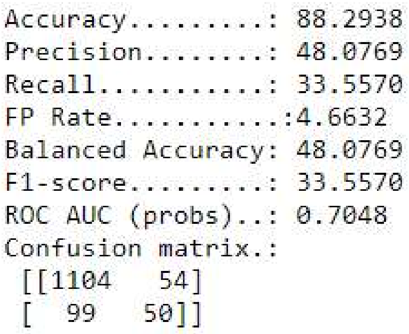
4.4 Results

The above three experiments demonstrate the effectiveness of adversarial training despite limited data and detecting a target between two very unbalanced classes. As the number of epochs of adversarial training increased, the accuracy of the tuned model is able to predict student attrition with a higher accuracy and a lower false positive rate. This can prominently be seen in experiment 3; using a threshold of about .31 and .44 for the control model and the tuned model respectively, we are able to make more accurate predictions while remaining at about a 5% false positive rate as can be seen in the confusion matrices in Figure 7. Furthermore, the Cohen's kappa values that relate the control and tuned models show that their outputs differ in each experiment to some degree suggesting the two models are predicting differently for a number of students. Increasing the amount of epochs for the GAN training may produce higher accuracy for the tuned model, but examining the loss diagram shows the discriminator and generator losses converging and later diverging at about 10,000 epochs which may reveal a problem in the GAN training on which we comment next.



```
Accuracy.........: 87.8347
Precision........: 45.1923
Recall...........: 31.5436
FP Rate..........:4.9223
Balanced Accuracy: 45.1923
F1-score.........: 31.5436
ROC AUC (probs)..: 0.6936
Confusion matrix.:
 [[1101   57]
 [ 102   47]]
```

(a) Control Model Confusion Matrix



```
Accuracy.........: 88.2938
Precision........: 48.0769
Recall...........: 33.5570
FP Rate..........:4.6632
Balanced Accuracy: 48.0769
F1-score.........: 33.5570
ROC AUC (probs)..: 0.7048
Confusion matrix.:
 [[1104   54]
 [   99   50]]
```

(b) Tuned Model Confusion Matrix

Fig. 7: Experiment 3 Confusion Matrices

5 Conclusions

As can be seen in the experiments, the classifier model of the GAN with tuning increases its accuracy the more epochs it trains and eventually is more accurate than the traditionally trained model. While the overall accuracy of either model is low, any increase in the ability to detect anomalous students who leave during freshmen year in such a small sample decreases the amount of false positives, which is important if the model is to be used in any official capacity. Thus, utilizing a GAN for better accuracy and a smaller false positive rate is a step in the right direction. If we are to compare the latent space of the GAN trained model that is tuned and the traditionally trained model, the layout of the data in the latent space in one model seems to form a semblance of a reflection along the diagonal of the layout in the other. The difference between the aforementioned latent spaces suggests the tuned model uses what it has learned about the input data in its prediction which may be the reason behind its higher accuracy in experiment 2 and experiment 3. This is further exemplified by the kappa values in each experiment comparing the control model to the tuned model displaying the two are making predictions that differ to a notable degree. It should also be noted that the loss diagrams show the discriminator and generator losses converging at about 10,000 epochs where the discriminator loss then continues to decrease and the generator's loss increases until both reach a point of stability with no major changes in their loss. This is most likely due to the generator suffering from mode collapse. The generator outputs data with most of its values hovering around .5 for the columns containing binary values which is likely the reason behind the immediate divergence in the loss diagrams after 10,000 epochs. Looking to the future, the adversarial training may produce better results using some degree of reinforcement training in order to introduce a penalty in the generator for values that are not 1 or 0 in binary columns rather than solely relying on unsupervised training to avoid mode collapse. Furthermore, a paper by Akash Srivastava and others shows promise in reducing mode collapse using implicit variational learning which is explained in detail in the referenced paper [9]. Nonetheless, using GANs in situations of limited data and anomaly detection shows promising results that should be explored further.

References

1. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein gan (2017)
2. Bewick, V., Liz, C., Jonathan, B.: Statistics review 13: Receiver operating characteristic curves. Critical Care (2004). URL ttps://doi.org/10.1186/cc3000
3. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 16, p. 785794. Association for Computing Machinery, New York, NY, USA (2016). DOI 10.1145/2939672.2939785. URL https://doi.org/10.1145/2939672.2939785
4. Dahl, G.E., Sainath, T.N., Hinton, G.E.: Improving deep neural networks for lvcsr using rectified linear units and dropout. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 8609–8613 (2013)
5. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks (2014)
6. Huang, Z., Pan, Z., Lei, B.: Transfer learning with deep convolutional neural network for sar target classification with limited labeled data. Remote Sensing **9**(9), 907 (2017). DOI 10.3390/rs9090907. URL http://dx.doi.org/10.3390/rs9090907
7. Kvlseth, T.O.: Note on cohen's kappa. Psychological Reports **65**(1), 223–226 (1989). DOI 10.2466/pr0.1989.65.1.223. URL https://doi.org/10.2466/pr0.1989.65.1.223
8. Lauria, E.J.M., Stenton, E., Presutti, E.: Boosting early detection of spring semester freshmen attrition: A preliminary exploration (2020)
9. Srivastava, A., Valkov, L., Russell, C., Gutmann, M.U., Sutton, C.: Veegan: Reducing mode collapse in gans using implicit variational learning. In: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (eds.) Advances in Neural Information Processing Systems 30, pp. 3308–3318. Curran Associates, Inc.

(2017). URL http://papers.nips.cc/paper/6923-veegan-reducing-mode-collapse-in-gans-using-implicit-variational-learning.pdf

10. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 19291958 (2014)