Anomaly detection with Generative Adversarial Networks and text patches

Andriy Drozdyuk
Department of Computer Science
Carleton University
andriy@drozdyuk.com

Norbert Eke
Department of Computer Science
Carleton University
NorbertEke@cmail.carleton.ca

Abstract

In this research work the possibility of adapting image based anomaly detection into text based anomaly detection is explored. Two main approaches are being proposed, namely anomaly detection as a task of classification and unsupervised anomaly detection using text patches. Both approached explore the use of generative adversarial networks to perform anomaly detection and results presented show that such can be fruitful.

1 Introduction

Anomaly detection is the task of identifying out-of-ordinary, unusual or unexpected data points. Text based anomaly detection could be interpreted as the identification of text containing malicious intent, like offensive language, hate speech, cyber-bullying, sexual predatory behavior, or even text containing suicidal or depressive thoughts or behavior. Such textual data is more difficult to obtain, as it is mostly present on online chat-rooms, forums and social networking platforms. Nowadays more and more social media data is filled with offensive language, hate speech, cyber bullying, which endangers the cyber-safety of both children and adults online. Automated anomaly detection in this context would lighten the workload on website and chat-room moderators whose job it is to maintain a safe communication and interaction in a cyber-society. In the context of depressive or suicidal behavior detection an automated anomaly detection system could make a huge difference in people's life, as proper help and a chance for reaching out to those in need could be possible with an early detection of users at risk, as Jamil et al.'s work suggests [1].

The biggest challenges with such data is the lack of labelled textual data (normal and anomalous text labels), the lack of negative examples, as the data sets are very unbalanced (usually only less than 10% of the sample size is anomalous data) and the messy, unstructured, unfiltered nature of the textual data makes it difficult to analyze.

By using generative models like GAN we are able to learn the distribution of the normal data well. The question is then how to use this knowledge to identify anomalies. In [2] the authors make use of the fact that anomalous image is an image that was not learned by the generative model. To detect whether a given query image is anomalous or not, all one has to do is attempt to generate a similar image and compute it's distance to the query image. If this distance is "small", then it can be reasoned that it is not anomalous. However, if our generative model is unable to produce an image that is *close* to the query image, then it is anomalous.

We extend this notion of similarity between generated and query images to the space of text. In the process we face two issues: calculating the distance between two text samples and backprogagating gradient from discriminative model to the generative model. We solve the former by making use of word-embeddings and calculating the distance between them. The latter issue arises because if we generate sequences of words, our discriminator is only able to evaluate complete sentences. We

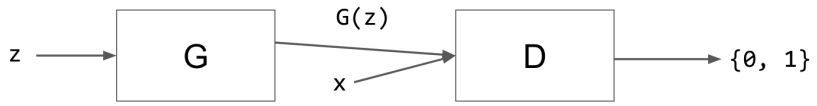

Figure 1: The generative model G captures the data distribution, and discriminative model D estimates the probability that the sample came from training data (x) rather than generative model ($G(z)$).

solve this issue by grouping words into *text-patches*. This enables our generative model to produce sequences of words, which can then be fed into the discriminator directly.

2 Related Work

In 2014 Goodfellow et. al. [3] proposed the idea of Generative adversarial nets, which consisted of two models: the generative and discriminative. The generative model can *generate* inputs to the discriminative model. The discriminative model estimates the probability that the input came from the real data rather than the generative model. The generative model's goal is to maximize the probability of discriminative model making a mistake. See Figure 1.

Collectively the two models play the following min-max game:

$$\min_G \max_D V(D,G) = E_{x\sim p_d(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z))]$$

where p_d is the distribution from which real data is drawn, while $p_z(z)$ is distribution produced by the generative model by sampling z from some (usually uniformly distributed) noise.

In [4], authors tackle the problem of generating sequences of discrete tokens with GANs. Since the discriminative model can only judge complete sequences while generative model can produce tokens that form partial sequences, it is not obvious how to reconcile the intermediate sequence scores with the score for the complete sequence. The proposed SeqGAN approach models the data generator as a stochastic policy in reinforcement learning (RL): "The RL reward signal comes from the GAN discriminator judged on a complete sequence, and is passed back to the intermediate state-action steps using Monte Carlo search."

Basing their research on *SeqGAN*, in [5] authors proposed *FakeGAN*, an augmentation to the GAN model which is effective at detecting deceptive reviews. The problem that the paper tackles is differentiating *truthful* from *deceptive* movie reviews. Deceptive reviews here mean reviews generated by bots or paid agents. Truthful reviews are those posted by real people. Note that truthful or deceptive reviews say nothing about the sentiment (positive or negative) of the review itself. To solve this problem, instead of a single discriminator, authors use two discriminator models D and D'. There is still just one generative model G. The D model must be able to differentiate between truthful and deceptive reviews, while D' must differentiate between data produced by the generative model G and samples from the *deceptive* reviews distribution. Unlike most GAN proposals, which aim to improve the generative model, this paper tries to improve the discriminator(s) instead. Similar to SeqGAN, authors use RL approach, where the discriminator uses Monte Carlo search to produce a reward signal that is then passed on as a gradient update to the generative model, with the generator itself modeled as a stochastic policy, with the following action value function:

$$A_{G_\alpha,D,D'}(a = S_L, s = S_{1:L-1} = D(S_{1:L}) + D'(S_{1:L})$$

where G_α is the generator, s is the sequence of produced tokens so far, a action is the next token. Authors expand this to a function of t so that it is able to complete a sentence using Monte Carlo search:

$$\{S^1_{1:L}, S^2_{1:L}, \ldots, S^N_{1:L}\} = MC_{G'_\gamma}(S_{1:t}, N)$$

where $S^i_{t+1:L}$ is "sampled via roll-out policy G'_γ based on the current state $S^i_{1:t-1}$".

Authors state that the FakeGAN discriminator convergences in practice, and avoids mode collapse, yet they do not provide a formal proof. The model training proceeds by some g steps for generator training followed by some d steps for discriminator updates.

Objective function for the generator is:

$$J(\alpha) = \sum_{S_1 \in X} G_\alpha(S_1|S_0) A_{G_\alpha, D, D'}(a = S_1, s = S_0)$$

Given the gradient $\delta_\alpha J(\alpha)$ of the above, the generator's parameters are updated as follows:

$$\alpha \leftarrow \alpha + \lambda \delta_\alpha J(\alpha)$$

Here are the objective functions for D and D' respectively:

$$\texttt{min}(-E_{S\sim X_T}[\log D(S)] - E_{S\sim X_D \vee G_\alpha}[1 - \log D(S)])$$

$$\texttt{min}(-E_{S\sim X_D}[\log D'(S)] - E_{S\sim G_\alpha}[1 - \log D'(S)])$$

where X_D are the deceptive reviews and X_T are the truthful reviews. For architecture, authors used RNNs for the generator, using LSTM with softmax output layer. For discriminator a CNN is used. FakeGAN was able to achieve accuracy of 89.1%.

In [6], authors propose to make use of *Earth-Mover* (EM) distance or Wasserstein-1 as the measure between real and model distributions. For some two distributions P_r and P_g it is defined as:

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} E_{(x,y)\sim\gamma}\left[||x - y||\right]$$

Authors explain the intuitive rationale behind this:

> Intuitively, $\gamma(x, y)$ indicates how much "mass" must be transported from x to y in order to transform the distributions P_r into the distribution P_g. The EM distance then is the "cost" of the optimal transport plan.

It is shown that the EM distance, if used as a loss metric, can be used to improve stability and get rid of mode collapse. Since it is hard to calculate EM directly in GAN, authors propose an approximation called Wasserstein-GAN (WGAN) which they successfully use to train their model. Furthermore, the paper is the first to propose a loss measure that correlates well with the visual quality of the generated samples. Unlike other measures, for example, JS, for which the loss goes up and down regardless of the sample quality, WGAN continually improves both. This is due to the fact that EM distance is continuous and differentiable, which means that WGAN can continue learning, while other models get vanishing gradients.

In [7] authors tackle the task of image inpainting. This forms the foundation for the AnoGAN algorithm described below. Unlike the previous approaches that use local or non-local information to recover the image, the authors instead use a GAN:

> Given a trained generative model, we search for the closest encoding of the corrupted image in the latent image manifold using ou context and prior losses. This encoding is then passed through the generative model to infer the missing content.

The back-propagation is used to find the encoding in the latent space (noise space) that is closest to the requested image. Both the generative G and discriminative D models are trained on normal (uncorrupted) image data. The generator can then take a given point z in the latent space p_z and produce a point $G(z)$ that is similar to the sample from the data distribution $p_{\texttt{data}}$. The authors aim to find the optimal $\hat{z}$ that is closest to the presented corrupted image. Once found, $G(\hat{z})$ is computed to produce an image, which is then blended with the corrupted image to fill in the missing information. The authors use two losses, the first one being the *contextual loss*:

$$L_c(z|y, M) = ||W \odot (G(z) - y)||_1$$

where W is an importance weighting term that assigns greater value to those uncorrupted pixels which have more corrupted pixels around them. The y is the corrupted image, and M is some mask that defines where the information is to be filled-in. The second type of loss is *prior loss*:

$$L_p(z) = \lambda \log(1 - D(G(z))$$

where λ regulates the contribution of this loss. The prior loss penalizes the unrealistic images that are produced by the generator.

Given the losses, the authors now continuously map the corrupted image y to the z in the latent space, until they arrive at $\hat{z}$ which is the *closest*. The weights are updated using the back-propagation of the total loss as follows:

$$\hat{z} = \texttt{arg}\min_z \{L_c(z|y, M) + L_p(z)\}$$

The authors get good results on image in-painting, doing better than state-of-the art models. Curiously enough, the model sometimes has trouble finding $\hat{z}$ which results in an incorrect image being generated. This difference is exactly what is used by the AnoGAN model below to compute the anomaly score.

In [2], authors proposed AnoGAN, an anomaly detection approach for medical images, that extends the approach in [7] by interpreting the distance of generated images from the query image as a measure of anomaly. The algorithm proceeds in two phases: training and anomaly detection. Before continuining, authors split up the images into $c \times c$ image patches. This allows them to reduce the dimentionality of the model. Subsequently, only random samples of these patches are used for both training and anomaly detection. For training, the usual GAN training procedure is applied to normal data only. Given a randomly sampled image patch x first train the discriminative model on x and $G(z)$ for some $z \in Z$ where Z is some uniformly distributed noise. For anomaly detection, the task is to determine whether a given query image patch is anomalous or not. The procedure is to pick a random $z \in Z$ and propagate it to two outputs: $G(z)$ and $f(G(z))$, where $f(.)$ is the output of some intermediate layer in discriminative model that encodes a feature. Then two losses are calculated. First, the *residual loss*, which is a measure of dissimilarity between the query image x and the generated image $G(z)$:

$$L_R(z_\gamma) = \sum |x - G(z_\gamma)|$$

Second, the *discrimination loss*, which measures the dissimilarity in features extracted by the discriminator:

$$L_D(z_\gamma) = \sum |f(x) - f(G(z_\gamma))|$$

Together, they produce an overall loss function:

$$L(z_\gamma) = (1 - \lambda)L_R(z_\gamma) + \lambda L_D(z_\gamma)$$

Here λ is the relative weight that each type of loss has. Then, the coefficients of z are backpropagated, while everything else remains fixed. This effectively produces a new $z' = kz$, and the process is repeated again for Γ iterations.

Finally, an anomaly score is computed as follows:

$A(x) = (1 - \lambda)R(x) + \lambda D(x)$

where $R(x) = L_R(z_\Gamma)$ and $D(x) = L_D(z_\Gamma)$ are the two losses evaluated at the last iteration of the anomaly detection procedure. Low anomaly score is interpreted as data being non-anomalous, while a large anomaly score means that the data is anomalous. Authors provide no way to normalize the scale of anomaly scores.

3 Method

3.1 Origin of our proposed work

The origin of our work comes from Schlegl et al.'s [2] work on image based anomaly detection (AnoGAN). The authors proposed to use healthy anatomy image patches (normal data) to train a generative adversarial model, then use anomaly scores to detect anomalous image patches. Our proposed work is an adaptation of this approach towards text based anomaly detection.

The use of generative adversarial networks in text based anomaly detection is in an early phase of development, as to our knowledge FakeGAN [5] was the first approach. While image based anomaly detection was proven to be successful in Schlegl et al.'s work [2], our main hypothesis is that text based anomaly detection is possible using the same general approach, with additional adaptation towards textual data input.

Throughout the process of coming up with potential approaches our initiative was to start out with two different approaches, later evaluate which one has more potential, then continue experimenting with the more successful approach. We describe these two possible anomaly detection adaptations from image to text data in section 3.2 and 3.3.

3.2 Anomaly detection as text classification using GANs

The backbone of this approach is to formulate the anomaly detection task as a two-class classification problem of discriminating between normal and anomalous data. During the training process the generative adversarial model is only allowed to see one class, and it will learn to treat that class as the "real" data. Anything that the generator starts generating will be classified as "fake" by the discriminator, until the generator does not get good enough at learning the distribution of this one class, the "real" data. Once the generator learns the distribution of the "real" data, the discriminator had to get good at learning to accurately classify "real" data. After the training process ended the discriminator can be used to classify between "real" and "not-real" data, which could be thought of as normal and not-normal, thus anomalous data.

Our approach aims to learn what normal data looks like by feeding only normal data into the GAN model, same as Schlegl et al.'s work [2], which is our main motivation for this approach. Training on the class with larger sample size could provide a better chance for the generator to learn the distribution of normal data, while the discriminator should learn to recognize normal data. Our hypothesis is that the discriminator will learn what normal data looks like, and will be able to classify it, while when presented with anomalous data, it will recognize it as not normal data, thus classifying it as anomalous data.

However, there is one big assumption that we are making. The discriminator learns to distinguish between real and generated data, and we are trying to classify normal and anomalous data. We made the assumption that classifying real and generated data would behave the same way as classifying normal and anomalous data. If this assumption fails, this can be corrected with a similar approach to FakeGAN [5] in future work.

The final system will use only the discriminator to classify both normal and anomalous unseen text sequences. We chose to train only on normal data, as that is the class that is always available, with or without annotated labels, and that is the approach used by Schlegl et al.'s AnoGAN [2]. If one would train on anomalous data, it would certainly be more difficult for the generative models to learn the real distribution of the data, as even by joining multiple anomalous data sources, there is barely over 1000 samples to train on, which does not tend to be enough for training generative models.

3.3 AnoGAN based approach using Text patches

We propose to replace image patches found in [2] with text-patches and adapt AnoGAN to work with discrete data for anomaly detection. We train our GAN architecture using non-anomalous data only. Given a randomly sampled query text-patch from a given text we use the generator to generate the closest possible match, and use word-vector distance to compute the anomaly score. If any text-patch is anomalous then we consider the whole text anomalous.

The main problem with applying GAN models to text generation is that they are not well suited to generating sequences of discrete tokens. The reason for this is that the generative model outputs discrete outputs one at a time, while discriminative model *judges* only complete sequences. This makes it difficult to pass gradient update from discriminative model to the generative model. Unlike other GAN approaches [4] [5][6][8] which use RL techniques to predict the discriminator output, we eliminate the issue entirely. By making the generative model produce a fixed sequence of outputs, we allow the model to learn the predictive capability. Our final system consists of two parts: generative and discriminative models. The generative model produces a set of word-embeddings, called *text-patches* (see Section 4.2.1) which are then passed to discriminative model. This way our discriminative model is able to immediately act on every output produced by the generator. Our hypothesis is that using text-patches will allow our generative model to learn the distribution inherent in the text data, and our anomaly detection procedure will be able to make use of this to distinguish between depressive and non-depressive samples.

While the original AnoGAN paper made use of convolutional layers quite extensively for image anomaly detection, we chose to implement our system with LSTM due to the sequential nature of textual data.

4 Experiments

4.1 Anomaly detection as text classification

Our implementation of training the generative adversarial model is based on Shibuya's deep learning applications [9], which is based on a Udacity tutorial on how to train generative adversarial models. FastText pre-trained word embeddings [10] were used to represent text in vector form. We used the model containing two million word vectors trained on Common Crawl. This model uses 300 dimensional word embeddings, it is publicly available and it can be downloaded from FastText's official website.

Text pre-processing was done in two text cleaning stages. In the first stage, all excessive white spaces, all numeric values, HTML tags, punctuation and hyperlinks were stripped, then text got converted into lower case. In the second stage we used a tweet processing library specially designed for cleaning tweet data by removing mentions, reserved words, emoticons and hash-tag signs. After the text cleaning stage there is a quick pass through all texts to find the longest sequence of text. In our data set of tweets this is a pre-processed tweet of 32 words. Using this number we feed all text tokens into a sequence padding algorithm and apply post-padding to all text sequences shorter than the longest text sequence. This step creates text sequences of equal length, which can be fed into the word embedding model and create a matrix of n (number of observations) X 32 (maximum length for sequence of text) X 300 (word embedding dimension).

4.1.1 Experiments with anomaly detection trained on normal data

When training the generative adversarial model on normal data, the focus shifts on the discriminator learning to classify into normal and anomalous classes after the training process. For the experiments in this section we used *Bell - Let's Talk* annotated tweet data set, which has annotations for tweets showing signs of depression [1]. This data set contains 5655 tweets labeled as normal and 547 tweets labeled as anomalous. Since we are training only on normal, but testing on normal and anomalous, we split the normal set into training and testing set, 90% training data, 10% testing data, just to put more emphasis on having a larger training set. This resulted in having 5090 normal tweets used during training, while the testing set had 565 normal tweets and 547 anomalous tweets.

The experiments in this section involved only feeding normal data into the generative adversarial model, while during testing both normal and anomalous data was presented to the discriminator.

Table 1: Performance measures on classifying using the discriminator when trained on normal data

Model	F1 Score	Recall	Precision	Sensitivity	Specificity
1 - LSTM	**0.6586**	0.4932	0.9909	0.4932	0.6154
2 - SimpleRNN 1	0.6091	0.4895	0.8062	0.4895	0.4976
3 - SimpleRNN 2	0.6110	0.4860	0.8227	0.4860	0.4785
4 - GRU 1	**0.6527**	0.4926	0.9671	0.4926	0.5263
5 - GRU 2	0.6242	0.5129	0.7971	0.5129	0.5763
AnoGAN Discriminator	0.6381	0.5119	0.8471	0.5124	0.8970
AnoGAN Anomaly Score	0.7980	0.7277	0.8834	0.7279	0.8928

Experiments consisted of trying out different model architectures for both the generator and discriminator, keeping the hyper parameters constant and seeing which model architecture produces better F1 scores on distinguishing between normal and anomalous data. We started with the discriminator. We tried out different combinations of dense and RNN layers, only dense layers and only RNN layers. We observed that using RNNs in the discriminator just for the sole purpose of classifying would cause over-fitting, as it seemed like the RNN would memorize entire sequences of data. This manifested in the form of after the training process the discriminator would classify anything that is has not seen as anomalous and anything close to what it has seen as normal. For an unseen test set all observations were classified as anomalous. We made the decision to drop RNNs from the discriminator, and have a really simple MLP architecture using just a dense layer, a max pooling layer, and an output layer of 1 using the sigmoid activation function to predict a label.

For the generator we experimented with a combination of dense layers and RNN layers like an LSTM, SimpleRNN and GRU layer. We have build 5 different models, labeled 1 to 5 in Table 1. All five models have the same discriminator, the one described above, but each model has a different generator. Throughout experimentation we noticed that the generator has to be a "stronger" model, while the discriminator should be a simple model that could classify between 2 classes. Model 1 has a time distributed dense layer with 300 hidden units with a Leaky ReLU using the parameter 0.01, followed by an LSTM layer of 300 hidden units. Model 2 the same architecture as model 1, except that it uses a SimpleRNN layer instead of the LSTM layer. Model 3 is similar to model 2, but it has a time distributed dense layer with only 100 hidden units. Model 4 has a time distributed dense layer with 100 hidden units using the same Leaky ReLU activation function, then followed by a Gated Recurrent Unit (GRU) layer using 300 hidden units. Model 5 is similar to model 4, expect that it used 300 hidden units instead of only 100 for the time distributed dense layer. All RNNs had to have 300 hidden units to match the architecture of 300 dimensional word embeddings, while the actual input into the generator is a sequence of word embeddings, 32 (maximum length for sequence of text, padded) X 300. We trained out generative adversarial networks for 5000 epochs. We used 0.0001 as learning rate for both the generator and discriminator, as this was the default value set by previous implementations. We experimented with different latent space sizes for our generator, 32 and 100, but noticed that 100 worked better. We used training batch size and 64, and evaluation batch size as 16, as these were default values from previous implementation.

We evaluated these models based on overall F1 score, but most importantly we tried to make sure that the overall model works properly. Model 1 and model 4 have a very high precision value, over 90%. This is not happening because the model is really good at predicting anomalous data, but because the model almost always predicts anomalous, thus it has high precision, but its recall value shows how it does not do well overall. In our opinion model 1 and model 4 should not be declared the best models, even though their F1 score beats the score obtained by Schlegl et al.'s discriminator model in their AnoGAN [2]. However, our Model 5 using a GRU layer is close to matching the performance of the AnoGAN discriminator with an F1 score of 0.6242, recall being the same as the AnoGAN discriminator's, while we are losing out by 2 percent on precision, thus we would like to declare model 5 our best model, based on good F1 score, but not having the fault of model 1 and 4, when the model always predicts one class. This model's architecture can be seen in Figure 2.

4.2 AnoGAN based approach using text patches

Our main experiments for this section came from augmenting our data set with more depressive data collected from a mental health forum called *Time To Change*. The nature of the data is perfectly

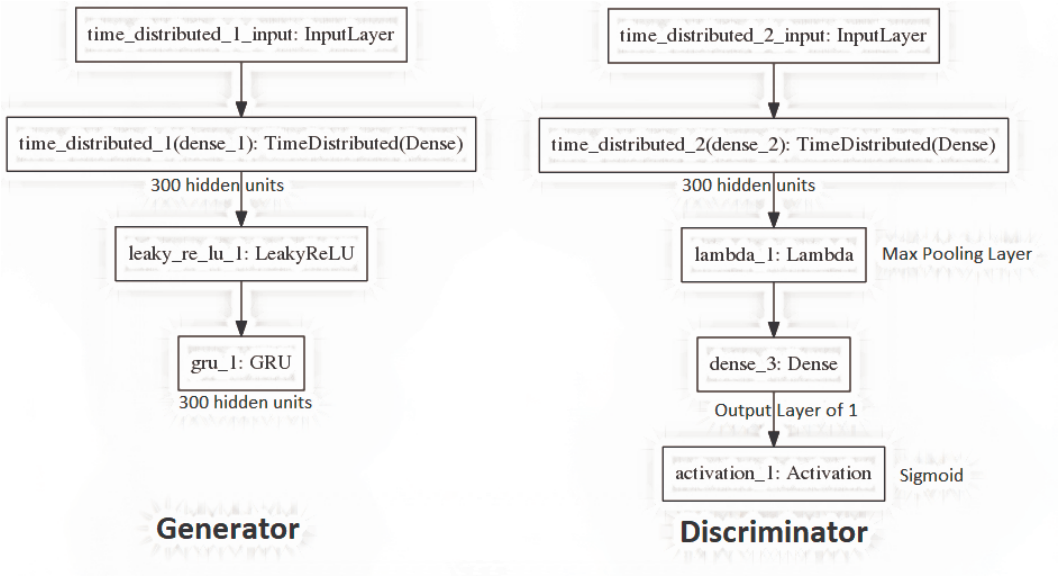

Figure 2: GAN Model 5 - GRU based model architecture from Table 1

Many attributes of dogs' personalities make them great pets. The first reason dogs are great pets is because they are often very loyal. Because dogs are unendingly loyal, many people consider them to actually be the best type of pet. Knowing that the family dog is watching out for everyone in the family gives everyone peace of mind. Another great trait of dogs is that they can be very gentle. Even the biggest dog can be calm and careful around a newborn or very small child, though dogs are not a substitute for parental supervision. Lastly, dogs can be so friendly that they make guests feel welcome in your home. Some dogs like nothing more than to lay at the feet of a guest as if to say, "I am here to help with whatever you might need. To conclude, dogs are great pets, and our lives would be less full without them.

Figure 3: Example of text patches five words long or, equivalently, $c = 5$.

suited for anomaly detection, as the textual data contains the personal stories of people struggling with depression. In order to make sure that the models are learning exactly what we intend them to learn we wanted the normal class be the total opposite of depressive stories. For the normal data we chose the positive reviews from the IMDB movie review data set, thus maximizing the semantic separation between the normal and anomalous classes.

4.2.1 Text Patches

In Schlegl et al.'s work [2] the training data for the GAN comes in the form of image patches. The authors define image patches to be k number of "2D image patches of size c by c from randomly sampled positions with $k = 1, 2, \ldots, K$". For our text based anomaly detection we needed to come up with an adaptation that is equivalent to image patches for text. We introduce the concept of text patches.

A *text patch* is a sequence of c words (Figure 3). It is a vector of size $c \times m$, where c is the number of words in a patch and m is the dimension of the embedding for a word.

4.2.2 Datasets

We use two datasets in our training and testing. The *IMDB* dataset [11], which contains both positive and negative movie reviews, and depressive data collected from *Time To Change* mental health forum.

We choose to use only the positive reviews from IMBD dataset, and produced a train-test split. Data was then cleaned from any html tags. The depressive data was similarly cleaned. After cleaning the data it was converted to word vectors using the FastText model trained on Common Crawl and Wikipedia [12].

To prepare our data for *training*, we simply scan over the sentences, and produce patches of K words each. Each patch has a dimension of $K \times M$ where M is the embedding dimension size. In the case of FastText $M = 300$. For a sentence i having length of W_i words, we can extract P_i patches, where:

$$P_i = \left\lfloor \frac{W_i}{K} \right\rfloor$$

If there are extra words remaining in the sentence we drop them. The number of patches in our data sets is usually around 50-150 per sentence. We proceed by creating as many patches as will fit into each sentence and then collect all the patches together into a single sequence of size $T \times K \times M$ where $T = \sum_i P_i$, or the total number of patches in all the samples. This, then, forms the input to our training procedure.

For *anomaly detection* we require our data be sets of patches randomly sampled from each sentence. The reason we don't use all the data for anomaly detection is mainly due to performance. It takes a long time to compute anomaly score (see below). To prepare data for anomaly detection phase we start by converting our data set of sentences into a data set of text-patches.

For a given sentence i, we produce P_i patches and then randomly sample B of them. If there are samples which do not contain sufficient number of patches, e.g. $P_i < B$, we skip them.

At the end of the data preparation step, our *anomaly detection* procedure receives a vector of size $N \times B \times K \times M$. Here, again, the N is the number of samples, B is the number of patches per sample, K is the size of each patch in word vectors and M is the size of the embedding of each word-vector (with $M = 300$ for FastText).

Recall that we split the *IMDB* dataset into training and testing, while we keep the depressive dataset as is. We used the training portion of the IMDB dataset as *normal* data for training the GAN. We used the testing portion as *normal* data for testing the GAN, while the depressive data was used as anomalous data for testing the GAN. For easy reference, let us denote the data that is normal by a $+$ and anomalous by a $-$, then we can describe our data as follows:

- T_+ - Training portion of the IMDB dataset used for for training the GAN.
- A_+ - Test portion of the IMDB dataset used for anomaly detection.
- D_- - The depressive dataset.

Note that the $+$ and $-$ have nothing to do with positive or negative connotation in the data, they simply indicate that the data was either normal or anomalous from the point of view of discriminative model of the GAN. We next proceed to training our model and detecting anomalies.

4.2.3 Training

For training we make use of T_+ data only, which is the training split of the IMDB dataset. Let G be the generative model, let D be the discriminative. Then, given N samples of size $P_i \times K \times M$, where $i = 1 \ldots N$, we proceed to train our GAN as follows:

1. For next patch $x \in T_+$, first train the discriminator:
   (a) Sample some noise $z \in Z$ from a uniform distribution
   (b) Use generator to generate some data $G(z)$
   (c) Compute discriminator loss $D(G(z)) = 0$
   (d) Compute discriminator loss $D(x) = 1$
   (e) Update discriminator weights
2. Then train the generator:
   (a) Disable discriminator training
   (b) Compute generator loss: $D(G(z)) = 1$

Table 2: Different configurations of anomaly detection experiment.

Samples	Num Patches	Patch Size	Epochs	Runtime (hours)
100	10	4	100	2
400	20	8	300	13

(c) Update generator weights
(d) Enable discriminator training

4.2.4 Anomaly detection

After our model has been successfully trained, we can use it to detect anomalies in new data. We start by presenting our model with a "query". This is just a text-patch for which we would like to know whether it is *anomalous* or not. The model should return low anomaly score if the query text-patch is not-anomalous, and high anomaly score if it is anomalous. In our case we take anomalous to mean a sample from *depressive* dataset D_-, and non-anomalous to mean a sample from test split of movie reviews dataset A_+.

When presented with a query text-patch x we proceed to compute an anomaly score as follows:

1. Form an anomaly detector A from our previously trained D and G models (see Figure 5)
2. Sample some noise $z \in Z$ from a uniform distribution
3. Produce $d_x = f(x)$ by passing our query through the discriminator but stopping at the *middle* layer f (see Figure 4, left)
4. Compute loss $A(x) - A(d_x)$
5. Backpropagate the gradients to our anomaly detector's trainable dense layer and repeat.

4.2.5 Model Architecture

For our model architecture (see Figure 4) we used a combination of LSTM and fully-connected layers. Our generative model used three subsequent LSTM layers, with 128, 256 and $M = 300$ hidden units respectively. The output was kept as a sequence. For discriminative model we used another set of LSTM layers, 256, 128 and 64 in size. Next, we added a fully-interconnected layer of size 32, named *middle* layer, in order to be able to extract the feature representation that will be used for anomaly detection later. We used *sigmoid* activation as the final layer. Both models were trained with binary crossentropy loss functions.

For anomaly detection, we insert a dense layer (*dense_15* in Figure 5) in front of our GAN architecture. We then disable the training of all the other layers in the model except for this new layer. This layer is then responsible for *finding* the $z \in Z$ in the noise space that will produce the closest output to our given query text-patch. The loss that we compute is the l_1 norm of the difference between the word vectors produced by generator and the query text-patch provided.

We trained our model on 4 Core 3.3Ghz i5-2500K CPU, 32GB of RAM, using NVIDIA RTX 2070 GPU with 8GB of memory. We were able to make use of *CuDNNLSTM* layers, which decreased our training time by a factor of 5 over just using simple LSTM.

For training we used a batch size of 256 and ran the training for 300 epochs. The training stage took around 8 hours to complete. For anomaly detection, we initially used a small number of samples, patches per sample and number of epochs. See Table 2 for execution times.

4.2.6 Interpreting Anomaly Scores

Anomaly scores were calculated on text patches sampled from both normal and anomalous text. The lower the anomaly score is, the less anomalous the data is. If the score is 0, it means data came from generators distribution. The higher the anomaly score is, the more anomalous the data is. Based on the work of Schlegl et al. [2] if one patch is classified anomalous, then the whole observation can be considered anomalous. We needed to come up with a way to evaluate whether or not a text patch can be considered anomalous enough to classify the whole text sample as anomalous. We are proposing

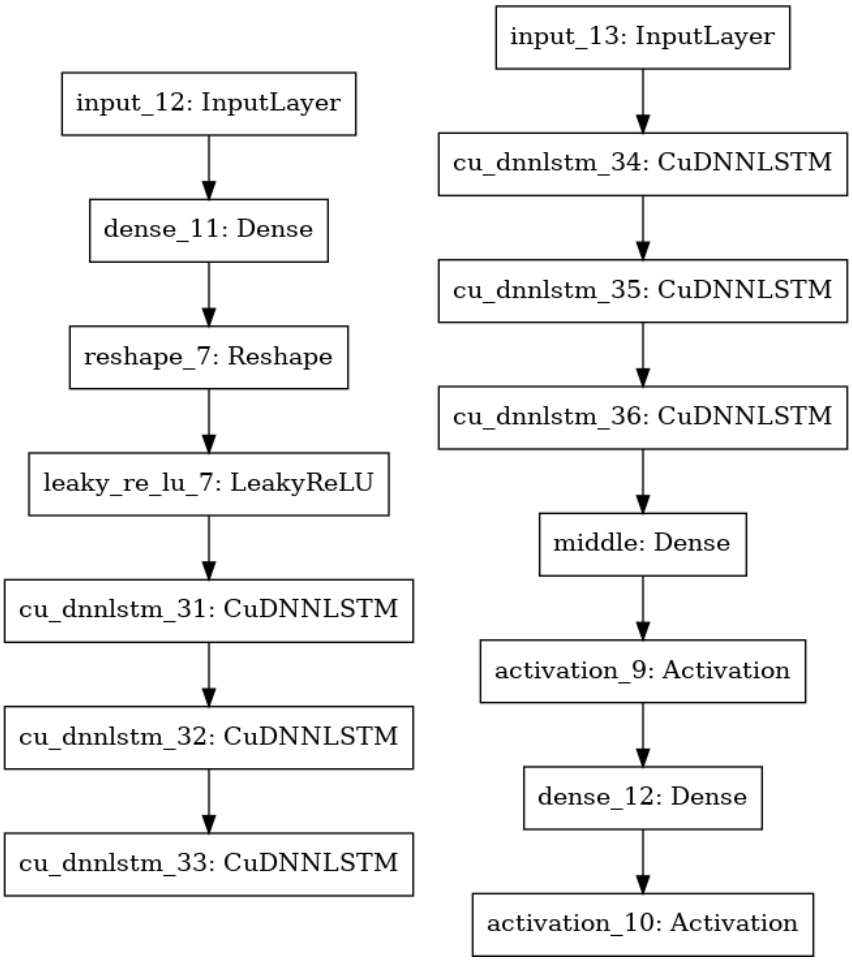

Figure 4: Architecture of the Generative G and Discriminative D Models. The Discriminative Model contains a *middle* layer that is used during the anomaly detection phase.

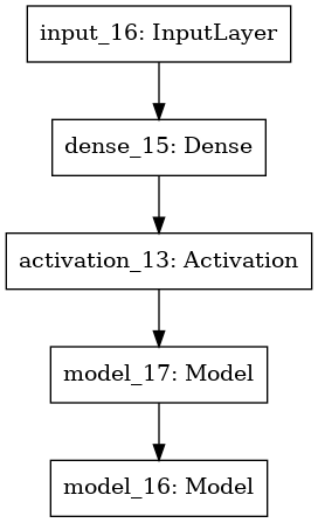

Figure 5: Architecture of the Anomaly Detector. Here *model_17* is the generative model G and *model_16* is the discriminative model D.

Table 3: Performance results of anomaly detection using text patches

Method	F1 Score	Precision	Recall	Sensitivity	Specificity
AnoGAN	**0.7980**	**0.8834**	**0.7277**	0.7279	0.8928
#1 - 10 patches, patch size 4	0.4700	0.5100	0.4359	0.4359	0.4096
#2 - 10 patches, patch size 8	0.4602	0.5200	0.4127	0.4127	0.3514
#3 - 20 patches, patch size 8	**0.6592**	**0.7400**	**0.5944**	0.5944	0.6556

to create an anomaly score threshold value based on the normal text sample's text patch anomaly score values. This anomaly score threshold value consists of mean of the normal text patch anomaly scores, plus 2 standard deviations of the normal text patch anomaly scores. Any anomaly score that is above this threshold can be considered an anomalous text patch, thus the text sample that contains this text patch is classified anomalous.

$$T = \mu_{\text{normal score}} + 2 * \sigma_{\text{normal score}}$$

where T is the Anomaly score threshold value, and normal score is the normal text patch anomaly score.

To evaluate all text samples from both normal and anomalous classes we go through all anomaly scores created by the text patches of each individual text sample. If there is a text patch containing an anomaly score larger than the threshold value, it gets classifies as an anomalous text. We measure the performance of this technique by comparing precision, recall and F1 scores of different patch sizes and text patch numbers to the original AnoGAN anomaly score performances. This can be seen in table 3.

We tried out 3 different methods. In method 1 we sampled 10 text patches from each text sample, and used patch size 4 for the text patches. In method 2 we increased the patch sizes to 8. In method 3 we increased the number of text patches to 20. We noticed significant improvement in results as we doubled the number of text patches randomly sampled from the data, however there was not a noticeable improvement on doubling the patch size. Method 3 is our best performing method so far, with an F1 score of 0.6592, precision of 0.74 and recall of 0.5944, which is drastically better than method 1 and 2, but falls short on achieving the same performance as the original AnoGAN technique [2].

Lastly, to make sure that the anomaly scores obtained from anomalous text are not the same as the ones coming from normal text, we ran an unpaired two-sample t-test for each of method's normal and anomalous scores. This hypothesis would test if there is no difference between the means of anomaly scores obtained from normal and anomalous text's text patches. The null hypothesis is that the two groups of anomaly scores are the same, meaning there is no real statistical difference between the two groups of anomaly scores. The alternative hypothesis is that the two groups of anomaly scores are different. For method 1's t test the p-value was 3.797e-05, for method 2 it was 2.121e-07, while for method 3 it was 2.2e-16. In all 3 cases there is enough evidence to reject the null hypothesis and conclude that the two groups of anomaly scores are different and statistically significant.

5 Conclusion

This research work investigated the possibility of adapting current research on image based anomaly detection into text based anomaly detection. Two main approaches have been proposed, namely anomaly detection as a task of classification and unsupervised anomaly detection using text patches, while both approaches using generative adversarial networks. The best model for the first approach achieves an F1 score of 0.6242, while the second approach outperforms the first with an F1 score of 0.6592. Both approaches could use further improvement, as they both fall short on matching the state of the art performance of AnoGAN with an F1 score of 0.7980, but nonetheless both newly proposed approaches were proven to work. Future work could involve expanding the AnoGAN architecture to make use of larger text patches and deeper model architecture. Perhaps using Capsule Networks might be able to better capture the normal data distribution. Additionally, it would be highly beneficial to try to use WGAN-style loss function, which correlates well with the quality of the generated samples.

References

[1] Zunaira Jamil, Diana Inkpen, Prasadith Buddhitha, and Kenton White. Monitoring tweets for depression to detect at-risk users. In *Proceedings of the Fourth Workshop on Computational Linguistics and Clinical Psychology—From Linguistic Signal to Clinical Reality*, pages 32–40, 2017.

[2] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International Conference on Information Processing in Medical Imaging*, pages 146–157. Springer, 2017.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[4] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858, 2017.

[5] Hojjat Aghakhani, Aravind Machiry, Shirin Nilizadeh, Christopher Kruegel, and Giovanni Vigna. Detecting deceptive reviews using generative adversarial networks. *2018 IEEE Security and Privacy Workshops (SPW)*, May 2018.

[6] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[7] Raymond A Yeh, Chen Chen, Teck-Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with deep generative models. In *CVPR*, volume 2, page 4, 2017.

[8] Liqun Chen, Shuyang Dai, Chenyang Tao, Haichao Zhang, Zhe Gan, Dinghan Shen, Yizhe Zhang, Guoyin Wang, Ruiyi Zhang, and Lawrence Carin. Adversarial text generation via feature-mover's distance. In *Advances in Neural Information Processing Systems*, pages 4667–4678, 2018.

[9] Naoki Shibuya. Using gan for generating hand-written digit images, 2017.

[10] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[11] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[12] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[13] Lucas Deecke, Robert Vandermeulen, Lukas Ruff, Stephan Mandt, and Marius Kloft. Anomaly detection with generative adversarial networks, 2018.