VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi – 590018

A TECHNICAL SEMINAR REPORT

ON

"Generative Adversarial Networks"

A Dissertation Submitted in partial fulfilment of the requirement for the degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE & ENGINEERING

Submitted by

BAHADURI PRACHITI JAGDISH (1RG17CS009)

Under The Guidance of

Mrs. Pushplata Dubey

Asst Professor, Dept of CSE

RGIT, Bengaluru – 32

Department of Computer Science & Engineering

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

Cholanagar, R.T. Nagar Post, Bengaluru-560032

2021 – 2022

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University)

Cholanagar, R.T. Nagar Post, Bengaluru-560032

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the Seminar Report titled **"Generative Adversarial Networks"** is a bona fide work carried out by **Ms. Bahaduri Prachiti Jagdish (1RG17CS009)** in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science and Engineering** under **Visvesvaraya Technological University, Belagavi**, during the year **2021-2022.** It is certified that all corrections/suggestions given for Internal Assessment have been incorporated in the report. This technical seminar report has been approved as it satisfies the academic requirements in respect of technical seminar (17CSS86) work prescribed for the said degree.

Signature of Guide

Mrs. Pushplata Dubey

Asst. Professor

Dept. of CSE,

RGIT, Bengaluru

Signature of HOD

Mrs. Arudra A

Asst. Professor

Dept. of CSE,

RGIT, Bengaluru

External Evaluation

Name of the Examiners **Signature with date**

1.

2.

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi – 590018

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DECLARATION

I hereby declare that the technical seminar report entitled **"Generative Adversarial Networks"** submitted to the **Visvesvaraya Technological University, Belagavi** during the academic year **2021-2022**, is record of an original work done by me under the guidance of **Mrs. Pushplata Dubey,** Asst Professor, Department of Computer Science and Engineering, RGIT, Bengaluru in the partial fulfillment of requirements for the award of the degree of **Bachelor of Engineering** in **Computer Science & Engineering.** The results embodied in this technical seminar report have not been submitted to any other University or Institute for award of any degree or diploma.

BAHADURI PRACHITI JAGDISH (1RG17CS009)

ACKNOWLEDGEMENT

I take this opportunity to thank my college **Rajiv Gandhi Institute of Technology, Bengaluru** for providing me with an opportunity to carry out this seminar work.

I extend my sincere regards and thanks to **Dr. Nagaraj A M**, Principal, RGIT, Bengaluru and to **Mrs. Arudra A**, Associate Professor and Head, Department of Computer Science and Engineering, RGIT, Bengaluru, for being a pillar of support and encouraging me in the face of all adversities.

I would like to acknowledge the thorough guidance and support extended towards us by **Mrs. Pushplata Dubey**, Assistant Professor, Dept of CSE, RGIT, Bengaluru. Their incessant encouragement and valuable technical support have been of immense help. Their guidance gave me the environment to enhance my knowledge and skills and to reach the pinnacle with sheer determination, dedication and hard work.

I also want to extend my thanks to the entire faculty and support staff of the Department of Computer Science and Engineering, RGIT, Bengaluru, who have encouraged me throughout the course of the Bachelor's Degree.

I want to thank my family for always being there with full support and for providing me with a safe haven to conduct and complete my technical seminar. I will ever grateful to them for helping me in these stressful times.

Lastly, I want to acknowledge all the helpful insights given to me by all my friends during the course of this technical seminar.

BAHADURI PRACHITI JAGDISH (1RG17CS009)

ABSTRACT

The emergence of generative adversarial networks has been called one of the most interesting successes in recent AI development and could make AI applications more creative. A novel approach for pitting algorithms against each other, called generative adversarial networks, or GANs, is showing promise for improving AI accuracy and for automatically generating objects that typically require human creativity. This study focused on efficient text generation using generative adversarial networks (GAN). Assuming that the goal is to generate a paragraph of a user-defined topic and sentimental tendency, conventionally the whole network has to be re-trained to obtain new results each time when a user changes the topic. This would be time-consuming and impractical. Therefore, we propose a User-Defined GAN (UD – GAN) with two – level discriminators to solve this problem. The first discriminator aims to guide the generator to learn paragraph level information and sentence syntactic structure, which is constructed by multiple – LSTMs. The second one copes with higher level information, such as the user-defined sentiment and topic for text generation. The cosine similarity based on TF – IDF and length penalty are adopted to determine the relevance of the topic.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

CHAPTER 1

INTRODUCTION

Generative Adversarial Networks (GANs) is a novel class of deep generative models which has recently gained significant attention. GANs learns complex and high-dimensional distributions implicitly over images, audio, and data. It was designed by Ian Goodfellow and his colleagues in the year 2014. Given a training set, this technique learns to generate new data with the same statistics as the training set. For example, a GAN trained on photographs can generate new photographs that look at least superficially authentic to human observers, having many realistic characteristics. Though originally proposed as a form of generative model for unsupervised learning, GANs have also proven useful for semi-supervised learning, fully supervised learning, and reinforcement learning. GAN has two different models for completing the data generating task. One of them is Generator G, which is responsible for generating data, and an other one is discriminator D, which determines whether the input data is the real data or not. The main focus for GAN (Generative Adversarial Networks) is to generate data from scratch, mostly images but other domains including music have been done.

The entities/adversaries are in constant battle as one(generator) tries to fool the other(discriminator), while the other tries not to be fooled. if your generator is not good enough, it will never be able to fool the discriminator and the model will never converge. If the discriminator is bad, then images which make no sense will also be classified as real and hence your model never trains and in turn you never produces the desired output.

1.1 Working of GANs

One neural network, called the generator, generates new data instances, while the other, the discriminator, evaluates them for authenticity; i.e. the discriminator decides whether each instance of data that it reviews belongs to the actual training dataset or not.

Let's say we're trying to do something more banal than mimic the Mona Lisa. We're going to generate hand-written numerals like those found in the MNIST dataset, which is taken from the real world. The goal of the discriminator, when shown an instance from the true MNIST dataset, is to recognize those that are authentic.

Meanwhile, the generator is creating new, synthetic images that it passes to the discriminator. It does so in the hopes that they, too, will be deemed authentic, even though they are fake. The goal of the generator is to generate passable hand-written digits is to lie without

being caught. The goal of the discriminator is to identify images coming from the generator as fake.

There are certain steps that GAN takes:

- The generator takes in random numbers and returns an image.
- This generated image is fed into the discriminator alongside a stream of images taken from the actual, ground-truth dataset.
- The discriminator takes in both real and fake images and returns probabilities, a number between 0 and 1, with 1 representing a prediction of authenticity and 0 representing fake.

So you have a double feedback loop:

- The discriminator is in a feedback loop with the ground truth of the images, which we know.
- The generator is in a feedback loop with the discriminator.

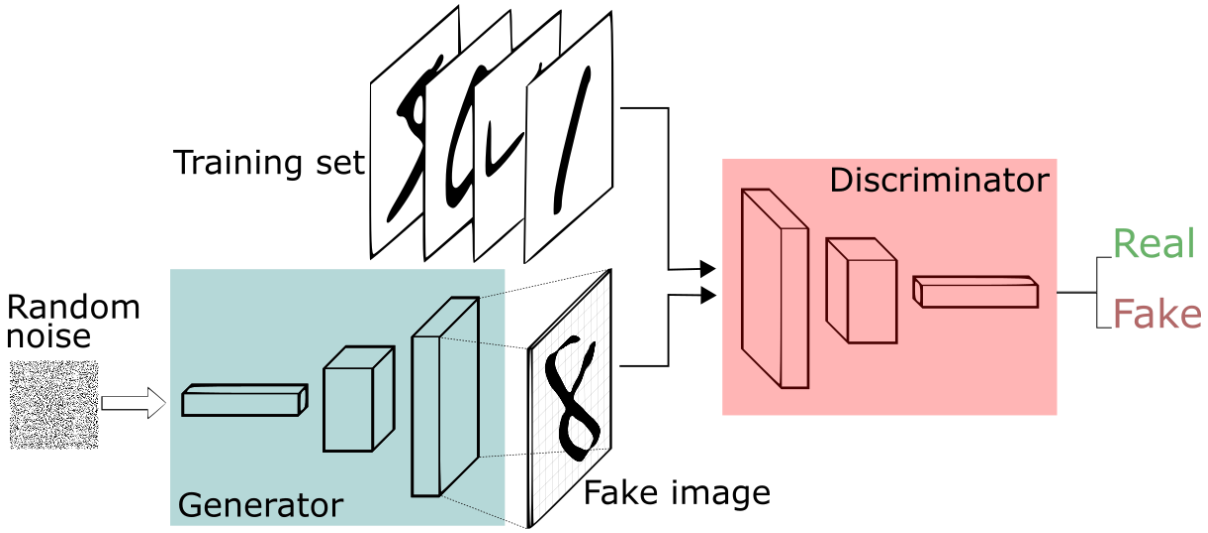

Figure 1.1(a): General structure of GAN

The goal of Generator is to maximize the likelihood that the discriminator misclassifies its output as real and the goal of Discriminator is to optimize toward a goal of 0.5 where the discrimiator cannot distinguish between real and generated images.

1.2 Objective Function

Our objective function is represented as a minimax function. A minimax problem seeks to minimize the maximum value of a number of decision variables. It is sometimes applied to minimize the possible loss for a worst case (maximum loss) scenario. A maximin problem maximizes the minimum value. It is used to maximize the minimum objective (such as profit

or revenue) for all potential scenarios. The discriminator tries to maximize the objective function, therefore we can perform gradient ascent on the objective function. The generator tries to minimize the objective function, therefore we can perform gradient descent on the objective function.

We thus arrive at the generative adversarial network formulation. The generator Gθ is a directed latent variable model that deterministically generates samples x from z, and the discriminator Dϕ is a function whose job is to distinguish samples from the real dataset and the generator. The image below is a graphical model of Gθ and Dϕ. x denotes samples (either from data or generator), z denotes our noise vector, and y denotes the discriminator's prediction about x.

The generator and discriminator both play a two player minimax game, where the generator minimizes a two-sample test objective (pdata=pθ) and the discriminator maximizes the objective (pdata≠pθ). Intuitively, the generator tries to fool the discriminator to the best of its ability by generating samples that look indisginguishable from pdata.

Formally, the GAN objective can be written as:

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

1.3 Comaparision of GANs

It is useful to compare GAN with other neural networks such as Autoencoders and Variational Encoders

- **Autoencoders**: Autoencoders encode input data as vectors. They create a hidden, or compressed, representation of the raw data. They are useful in dimensionality reduction; that is, the vector serving as a hidden representation compresses the raw data into a smaller number of salient dimensions. Autoencoders can be paired with a so-called decoder, which allows you to reconstruct input data based on its hidden representation, much as you would with a restricted Boltzmann machine.

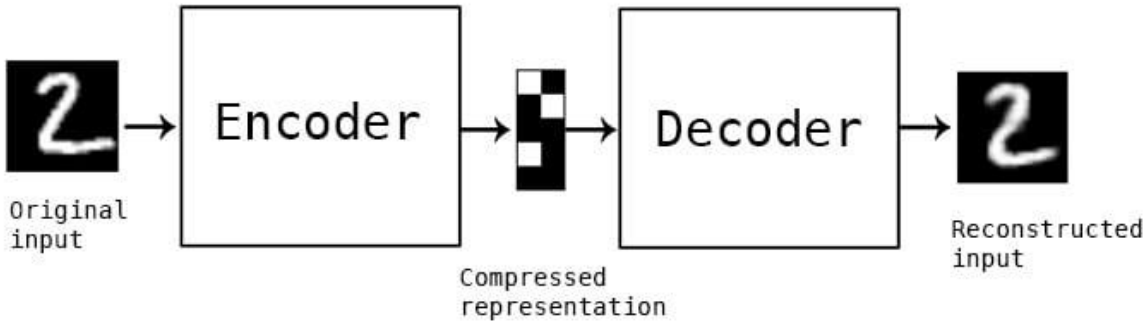

Figure 1.3(a): Autoencoders

- **Variational autoencoders**: These are generative algorithm that add an additional constraint to encoding the input data, namely that the hidden representations are normalized. Variational autoencoders are capable of both compressing data like an autoencoder and synthesizing data like a GAN. However, while GANs generate data in fine, granular detail, images generated by VAEs tend to be more blurred.

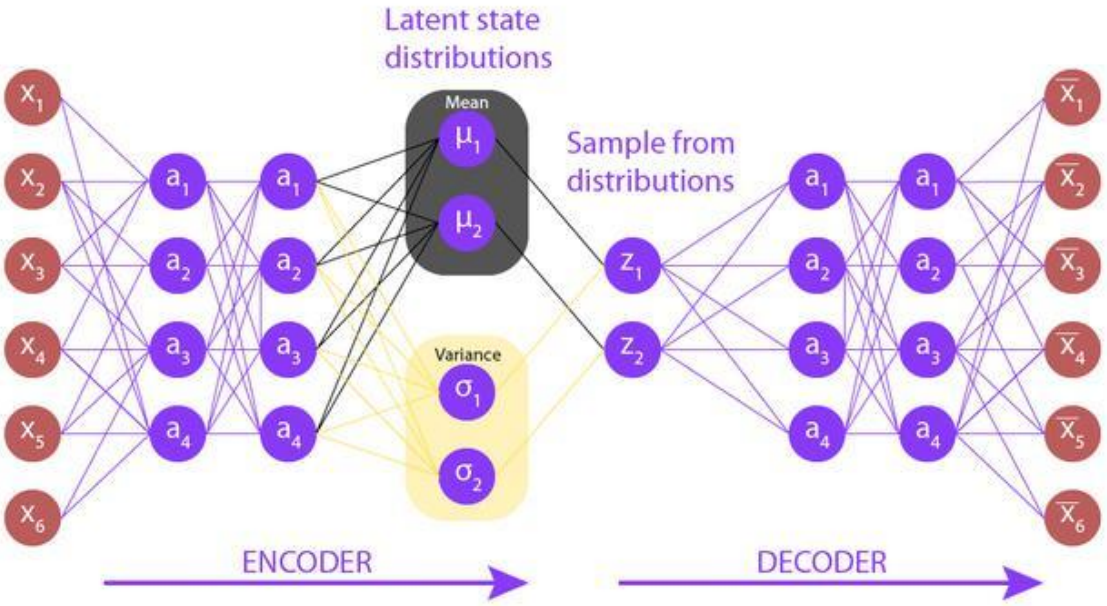

Figure 1.3(b): Variational Autoencoders (VAE)

1.4 Applications of GAN

GAN has interesting applications that are commonly used in the industry:

- **GANs for Image Editing**: Most image editing software these days do not give us much flexibility to make creative changes in pictures. For example, let us say we want to change the appearance of a 90-year-old person by changing his/her hairstyle. This can not be done by the current image editing tools out there. Another similar application is image de-raining (or literally removing rainy texture from images.
- **Using GANs for Security**: A constant concern of industrial applications is that they should be robust to cyber attacks. There is a lot of confidential information on the line! GANs are proving to be of immense help here, directly addressing the concern of adversarial attacks. These adversarial attacks use a variety of techniques to fool deep learning architectures. GANs are used to make existing deep learning models more robust to these techniques.
- **GANs for 3D Object Generation**: Game designers work countless hours recreating 3D avatars and backgrounds to give them a realistic feel. It certainly takes a lot of effort to create 3D models by imagination. GAN has incredible power to be used to automate the entire process and create 3D models.
- **GANs for Text Generation:** With the rise of deep learning, GANs have been introduced into the NLP community thereby paving a way to generate text accordingly.

REQUIREMENTS

CHAPTER 2

REQUIREMENTS

2.1 Data Acquisition

The opinion section of Newsweek is selected as training corpus because the paragraphs of the essays in Newsweek are generally closely related and not long. . The other reason is that through the articles in the opinion section, authors can often convey their own sentiment tendencies. NER is used to replace name-entities with their name-entity tags to decrease vocabulary. After tokenizing the corpus, long sentences of more than 45 words in the corpus were removed. The final training corpus has 425K sentences and 103K paragraphs.

2.1.1 Name Entity Recognition

Named entity recognition (NER) is probably the first step towards information extraction that seeks to locate and classify named entities in text into pre-defined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc. NER is used in many fields in Natural Language Processing (NLP), and it can help answering many real-world questions, such as:

- Which companies were mentioned in the news article?
- Were specified products mentioned in complaints or reviews?
- Does the tweet contain the name of a person? Does the tweet contain this person's location?

2.2 Reinforcement Learning

Reinforcement learning (RL) is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward.[1] Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning. Due to its generality, reinforcement learning is studied in many disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, and statistics. Reinforcement Learning (RL) refers to both the learning problem and the sub-field of machine learning which has lately been in the news for great reasons. RL based systems have now beaten world champions of Go, helped operate

datacenters better and mastered a wide variety of Atari games. The research community is seeing many more promising results. With enough motivation, let us now take a look at the Reinforcement Learning problem.

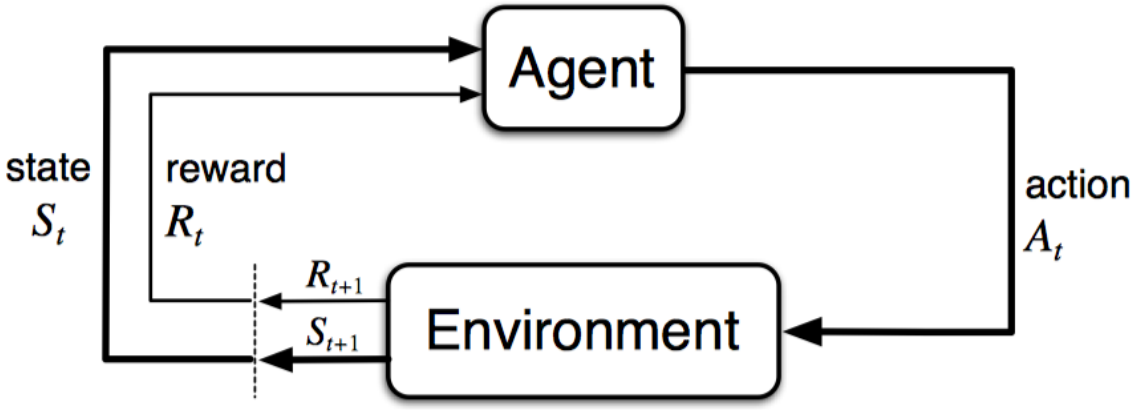

Figure 2.2: Reinforcement Learning

2.2.1 SeqGAN

In SeqGANs, the generator is treated as an **RL agent**.

- **State:** Tokens generated so far
- **Action:** Next token to be generated
- **Reward:** Feedback given to guide G by D on evaluating the generated sequence

  As the gradients cannot pass back to G due to discrete outputs, G is regarded as a stochastic parametrized policy. The policy is trained using policy gradient.

The objective of a Reinforcement Learning agent is to maximize the "expected" reward when following a policy π. Like any Machine Learning setup, we define a set of parameters θ (e.g. the coefficients of a complex polynomial or the weights and biases of units in a neural network) to parametrize this policy (also written a π for brevity) SeqGAN is used as the baseline system to evaluate UD – GAN and is trained for 20 epochs.

2.2.2 LeakGAN

LeakGan is a framework to address the problem of lon text generation. The discriminator net is allowed to leak its own high-level extracted features to the generative net to further help the guidance. The generator incorporates such informative signals into all generation steps. This framework is used as a baseline

system to evaluate UD – GAN and is trained for 20 epochs. A typical approach is to train a recurrent neural network (RNN) to maximize the log-likelihood of each ground-truth word given prior observed words, which, however, suffers from so-called exposure bias due to the discrepancy between training and inference stage: the model sequentially generates the next word based on previously generated words during inference but itself is trained to generate words given ground-truth words.

2.3 Recurrent Neural Networks

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Derived from feedforward neural networks, RNNs can use their internal state (memory) to process variable length sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition[4] or speech recognition.

The term "recurrent neural network" is used indiscriminately to refer to two broad classes of networks with a similar general structure, where one is finite impulse and the other is infinite impulse. Both classes of networks exhibit temporal dynamic behavior.[7] A finite impulse recurrent network is a directed acyclic graph that can be unrolled and replaced with a strictly feedforward neural network, while an infinite impulse recurrent network is a directed cyclic graph that can not be unrolled.

Both finite impulse and infinite impulse recurrent networks can have additional stored states, and the storage can be under direct control by the neural network. The storage can also be replaced by another network or graph, if that incorporates time delays or has feedback loops. Such controlled states are referred to as gated state or gated memory, and are part of long short-term memory networks (LSTMs) and gated recurrent units. This is also called Feedback Neural Network (FNN).

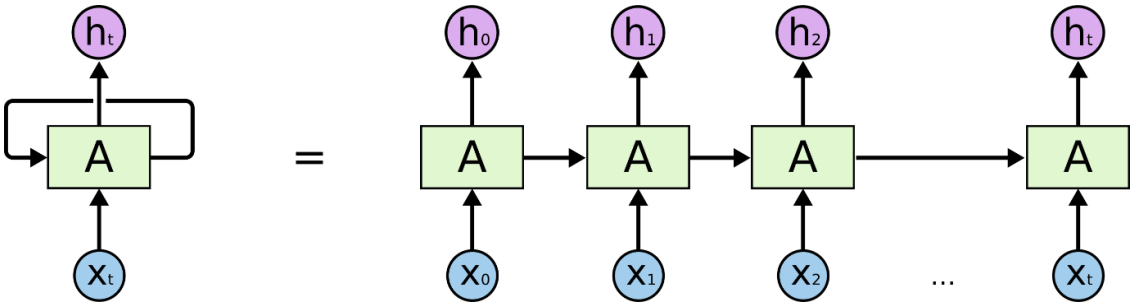

Figure 2.3: Structure of Recurrent Neural Network (RNN)

2.3.1 Long Short Term Memory (LSTM)

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or IDSs (intrusion detection systems).

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.

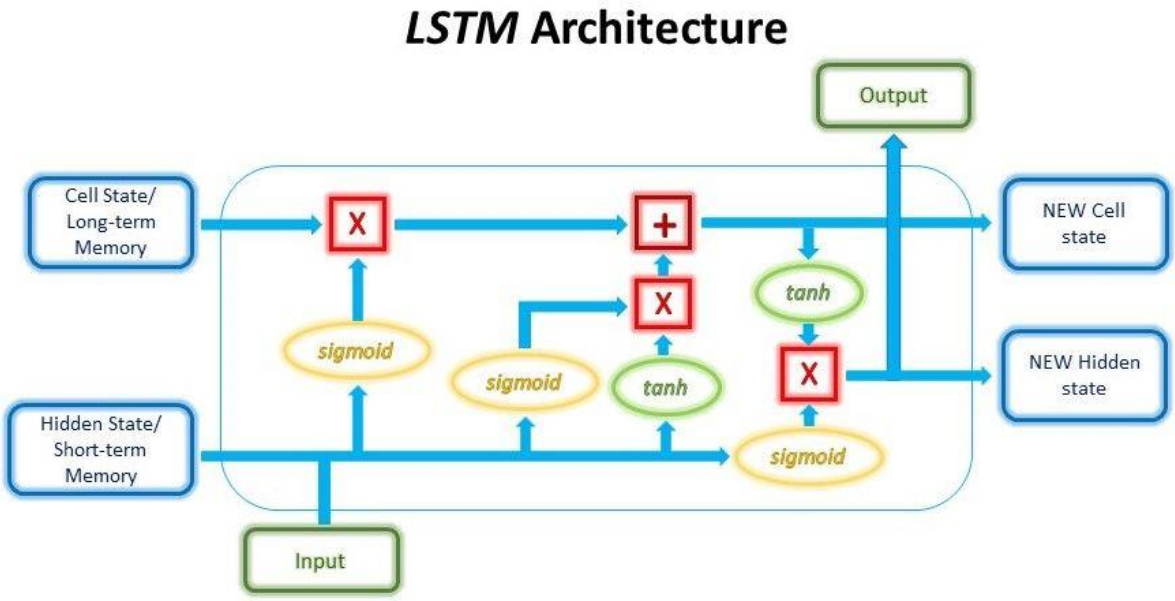

Figure 2.3.1: LSTM Architecture

PROPOSED METHOD

CHAPTER 3

PROPOSED METHOD

3.1 UD – GAN Structure

UD – GAN contains a generator Gθ that is capable of generating contextdependent sentences and the two – level discriminators. Discriminator – general Dφ guides the generator to learn the paragraph-level information and correct syntactic structure, while discriminatorspecial Dγ determines whether the generated text is related to the user-defined topic and sentiment. Discriminator – special Dγ is trained with synthetic perfect data and generated text data, while discriminator – general Dφ is trained with realtext data and generated text data. As we apply reinforcement learning with policy gradient to train the generator, the outputs of the two discriminators for the generated text will be combined and served as a reward to train the generator. Generator Gθ will choose the best next actions based on the reward it received.

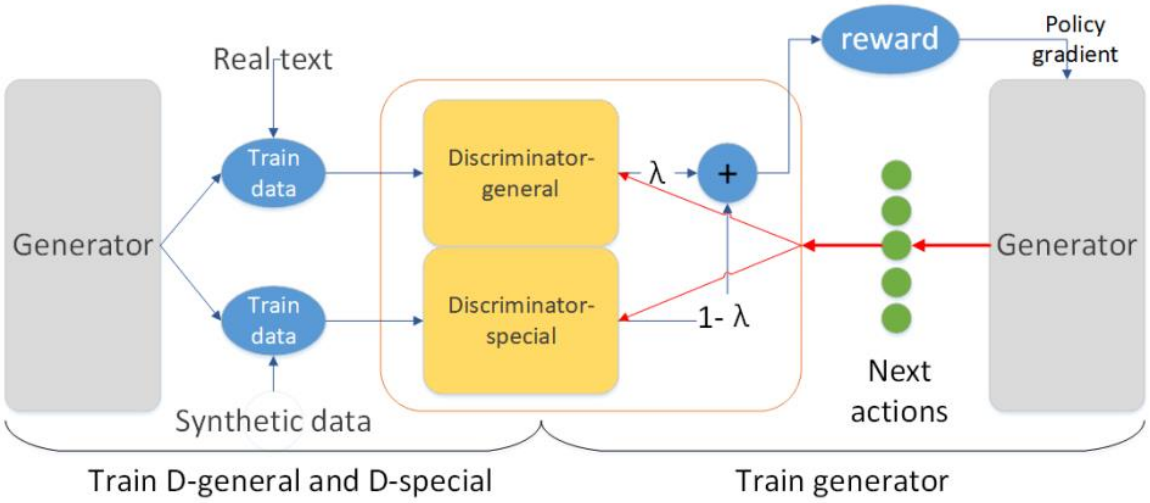

Figure 3.1: The framework of the proposed UD – GAN

Basically there are 2 algorithms where the next steps will be followed. The first algorithm deals with the initial training of the generator Gθ, discriminator – special Dγ and discriminator – general Dφ.

- In first algorithm the Gθ, Dφ and Dγ are initialized with random weights θ, φ and γ. The generator Gθ is pre – trained using Maximum Likelihood Estimation on real text data set. Negative samples are generated via generator Gθ to train the discriminator – general Dφ and discriminator – special Dγ. Some synthetic positive samples are

generated to train discriminator – special Dγ. Cross – entropy is minimized for both discriminator – general Dφ and discriminator – special Dγ in order to pre – train. The sequence is generated, rewards are computed and then the parameters of generator Gθ are updated. Negative samples are generated using generator Gθ and the discriminator – general Dφ are trained with negative samples and real text data. Once that is done, the feature vectors corresponding to negative samples are generated by generator Gθ. Then the synthetic feature vectors are generated and discriminator – special Dγ is trained with negative and synthetic feature vectors.

- The second algorithm deals with the follow up training of generator Gθ and discrimnator – special Dγ. Initialize generator Gθ, discriminator – special Dγ with random weights θ, Dγ. And then load trained discriminator – general Dφ. This is followed by pre – training the generator Gθ using Maximum Likelihood Estimation and generating a sequnece followed by computing the rewards and updating the parameters generator Gθ.

3.2 The Framework of D – Special

The framework of D – Special is comprised of a binary LSTM for each discriminator and also multilayered LSTMs for each word which form upto a sentence.

3.2.1 The Feature Vector of D – Special

Discriminator – special Dγ takes a vector containing 5 elements as input, which could represent the sentimental and topical relevance of each sentence. In our model, users can describe the cause and effect of an event in one sentence, which is used as the topic for generating sentences. We use the first element to represent the similarity between sentence entered by the user and generated sentence, which could also represent the user-defined topic relevance of the generated text. Based on the TFIDF value of each word in the sentence that the user entered and the generated sentence, the cosine similarity between these two sentences is calculated as a parameter to measure the user-defined topic relevance of the generated sentence.

A larger value of cosine similarity means that the generated sentence is related to the user-defined topic. However, if only this element is used to instruct the generator Gθ to generate topic-related sentences, the resulting sentences will be substantially as long as the user – defined topic sentence. More importantly, the

generated sentences will lack diversity with same meaning. Therefore, we propose the second element, length penalty, to reduce the negative impact of the first element. The difference between the length of the generated sentence and the length of the topic sentence defined by user is mapped in [0, 1]. We set 0.5 to the optimal length penalty, which means that if the length of the sentence is very close to or very far from the length of the topic sentence, it is unqualified.

In conventional GAN training, the discriminator treats real text data as the positive sample and generated text as the negative sample. However, there is no sentence in real corpus that has exactly the same features as the positive sample, since its feature vector is constructed by applying the above mention algorithm, while the user – defined feature vector is a specific value. Therefore, we train the discriminator-special Dγ with synthetic data, which is treated as positive sample. For example, supposing that the user would like to generate an essay with one positive emotion, then the UD – GAN will generate [1,0.5,1,0,0] vectors corresponding to the number of generated sentences, which will be combined with vectors corresponding to the generated sentences as the input of discriminator – special.

3.2.2 The structure of D – Special

Two linear layers with Relu as the activation function are used as discriminator-special Dγ. The output of this network will be part of the reward to train generator Gθ after it passed through a softmax layer. The reason for the discriminator – special to be fully – connected layer is that in the subsequent training, the discriminator – special will be continuously retrained when demands of the user change which is a little time consuming, but a fully – connected layer has fewer parameters which means that the network will converge faster than the others.

3.3 The Framework of D – General

Unlike conventional ideas of using classifier based models as a discriminator, the discriminator – general Dφ needs to process sequence data and context information, such as the paragraph information for each sentence to generate paragraph level text. Therefore, as shown in the figure below, we designed a hierarchical-multiple-LSTM neural network as the discriminator-general Dφ. The bottom multilayers LSTM takes an embedding vector for each word in a sentence as the input and it outputs a feature matrix representing the corresponding

sentence. The top bidirectional LSTM (Graves and Schmidhuber, 2005) takes the feature matrices of these sentences, which belong to the same paragraph, as input and it outputs a feature matrix representing that paragraph. After transforming through two different linear layers respectively, the above two feature matrices will be combined together

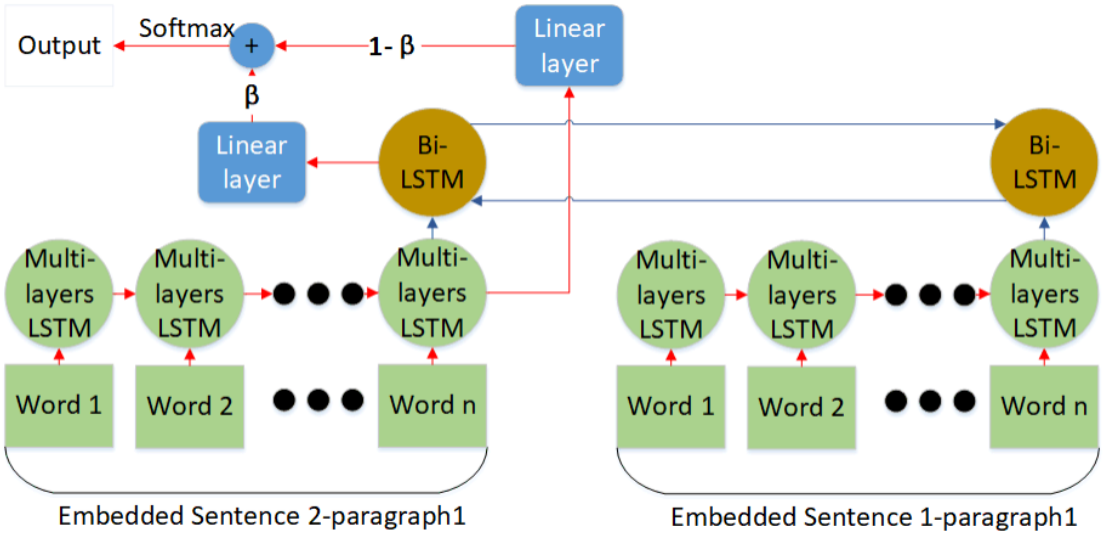

Figure 3.3: The proposed framework of Discriminator – general

Finally, the discriminator – general calculates the score of the input sentence via the equation below:

$$R(Y) = sof\,tmax[(1 - \beta)LSTM\alpha + \beta LSTM\eta]$$

where, β is the trainable parameter ranging from 0 – 1.

3.4 Generator

Generator Gθ is designed with GRU (Chung et al., 2014). In UD-GAN, due to the excessive parameters of the two discriminators, it is easy to guide the generator to be over-fitting. As a commonly used variant of LSTM, GRU avoids this over-fitting problem. In addition, having fewer parameters than conventional LSTM allows GRU to take less time to converge, which is the first priority in UD-GAN.

3.5 Reward and Policy Gradient training

Policy gradient is an approach to solve reinforcement learning problems. In this scenario, generator Gθ will use the results from discriminators on the generated text as reward to generate next words. In UD-GAN, the reward is calculated based on results of two discriminators. Generator Gθ tries to maximize expected reward from the initial state till the end state via the loss function. The goal of reinforcement learning is to find an optimal behavior strategy for the agent to obtain optimal rewards. The policy gradient methods target at modeling and optimizing the policy directly.

ANALYSIS AND EVALUATION

CHAPTER 4

ANALYSIS AND EVALUATION

For evaluating the effectiveness of the proposed method, we first compared the sentences relevance to user-defined topic and sentimental tendency, and then compare the training time of each system. Finally, the fluency and correctness of UD – GAN and baseliens were evaluated.

4.1 Relevance of Topic

As an objective summary accuracy evaluation method that is widely used, ROUGE (Lin, 2004) is also adopted here to evaluate whether generated sentences are related to user – defined topics. Generated sentences are treated as summaries to be evaluated, and the topic sentence defined by user is used as a reference summary to evaluate whether the generated sentence is related to the topic. Note that even if the ROUGE scores of the generated sentences are not high, it does not mean that these sentences are not closely related to the user-defined topic necessarily. One possibility is that the generated sentences will use other words or syntactic structures to describe the topic sentence. The ROUGE – L scores of all the sentences is reported. Based on the longest common subsequence, ROUGE – L is a score related to recall rate.

GAN-based models	ROUGE-L
UD-GAN(GS)	364.73
UD-GAN(S)	**370.54**
UD-GAN(G)	340.19
SeqGAN	342.27
LeakGAN	345.03

Table 4.1: The ROUGE – L Score for each system

4.2 Relevance of Sentimental Tendency

The VADER algorithm to calculate the probability that a generated sentence belongs to a positive, negative or neutral emotion class. The optimal sentiment is defined by the user. The system performance is evaluated by setting the target sentimental tendency as positive.

	Positive	Negative	Neutral
UD-GAN(GS)	0.39	0.05	0.56
UD-GAN(S)	**0.41**	**0.04**	**0.55**
UD-GAN(G)	0.10	0.08	0.82
SeqGAN	0.09	0.08	0.83
LeakGAN	0.08	0.07	0.85

Table 4.2: Probability of sentiment tendency of generated sentences

As shown in the above table, the average probability in each sentimental tendency category of all sentences is calculated. With training discriminator – special, UD – GAN (G+S) and UD – GAN (S) are more likely to generate positive sentences than baselines. Which proves that the proposed method is capable to generate the sentences with the desired sentiment. However, since the total number of sentences expressing positive sentimental tendency in the training corpus is quite low, the probability of UD – GAN generating positive sentiment is still not higher than 0.5 value.

4.2.1 Generate Context-dependent Sentences

To demonstrate that UD – GAN can generate context-dependent sentences, we show sentences generated by UD – GAN and baselines. The proposed UD – GAN does generate sentences related to the user-defined topic. UD – GAN tries to add some conjunctions when generating sentences so that the sentences seem to be related, and each sentence is extended with other related words based on the topic.

4.3 Training Time Evaluation

The time spending on gradient propagation and update of UD – GAN and baselines are compared, instead of the time spending on loading and saving data. All GAN – based models compared here are implemented in pytorch (Paszke et al., 2017) framework to eliminate the impact of different frameworks on time consumption. As shown in Table.4, because the structure of discriminator-general is more complex than the structure of discriminator D of baselines, initial training of UD – GAN takes the longest time. However, in the subsquent trainings, due to the gradient propagation and parameter update of discriminator-special is quite fast, the time required to train UD – GAN (S) is the shortest. The UD – GAN (S) takes only about an hour and a half to complete training, which is much less than the nearly eight hours of training time for baselines.

GAN-based models	Time s
UD-GAN(GS)	29061.48
UD-GAN(S)	**4841.99**
UD-GAN(G)	29036.65
SeqGAN	27011.08
LeakGAN	30471.95

Table 4.3: Training time for each model

4.4 Fluency and Accuracy

BLEU (Papineni et al., 2002) scores of UD – GAN are reported and baselines are used to compare the fluency and accuracy of the generated text. In the case of training the discriminator-general only, the BLEU score of the UD – GAN (G) is between SeqGAN and LeakGAN. Therefore, the accuracy and fluency evaluation of using multi-layer LSTMs as a discriminator is comparable to that of using a classifier-based model, such as CNN, as the discriminator. When the discriminator general and discriminator-special are simultaneously trained (initial training), UD – GAN (G+S) has a slightly higher BLEU score than UD – GAN (G). That is to say, even if discriminator-special is added and the result of discriminator-general, which can distinguish the correctness of the sentence, is less weighted, the resultant generator of UD – GAN (G+S) can still learn how to generate a sentence with the correct syntax.

GAN-based models	BLEU score
UD-GAN(G+S)	0.6412
UD-GAN(S)	0.6409
UD-GAN(G)	0.6357
SeqGAN	0.6303
LeakGAN	**0.7161**

Table 4.4: Average BLEU Score for each system

CONCLUSION

CONCLUSION

Generative Adversarial Networks are a recent development and have shown huge promises already. It is an active area of research and new variants of GANs are coming up frequently.

A UD – GAN method, which was used to re-train text generation model more efficiently, was to generate sentences that are consistent with the new user-defined topic and sentimental tendency. The accuracy and fluency of sentences generated by UD – GAN with other GAN – based text generation models was compared. The experimental results showed that sentences generated by UD – GAN are competent. Meanwhile, UD – GAN takes much less time in the re-train stage than other models. According to experimental results, UD – GAN can also successfully generate sentences related to the user-defined topic and sentimental tendency, while baselines does not have this capability. Besides, UD – GAN can also generate paragraph-level text.

REFERENCES

REFERENCES

Towards Data Sciece articles on Fundamentals of Generative Adversarial Networks
Deep insights on techniques and fundamentals of Generative Adversarial Networks by Google Developers
Article on New Machine Learning Approaches to Text generation

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

[2] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555

[3] William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. Maskgan: Better text generation via filling in the . arXiv preprint arXiv:1801.07736

[4] Xiaocheng Feng, Ming Liu, Jiahao Liu, Bing Qin, Yibo Sun, and Ting Liu. 2018. Topic-to-essay generation with neural networks. In IJCAI, pages 4078–4084

[5] Alex Graves and Jurgen Schmidhuber. 2005. Frame- ¨ wise phoneme classification with bidirectional lstm and other neural network architectures. Neural Networks, 18(5-6):602–610

[6] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. In Thirty-Second AAAI Conference on Artificial Intelligence

[7] Sepp Hochreiter. 1998. The vanishing gradient prolem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6(02):107–116.

[8] Sepp Hochreiter and Jurgen Schmidhuber. 1997. ¨ Long short-term memory. Neural computation, 9(8):1735–1780.

[9] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Controllable text generation. arXiv preprint arXiv:1703.00955, 7.

[10] Ferenc Huszar. 2015. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? arXiv preprint arXiv:1511.05101.

[11] Clayton J Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In Eighth international AAAI conference on weblogs and social media.

[12] Jiwei Li, Will Monroe, Tianlin Shi, Sebastien Jean, ´ Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. arXiv preprint arXiv:1701.06547.

[13] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. Text Summarization Branches Out.

[14] Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial ranking for language generation. In Advances in Neural Information Processing Systems, pages 3155–316

[15] Toma´s Mikolov, Martin Karafi ˇ at, Luk ´ a´s Burget, Jan ˇ Cernock ˇ y, and Sanjeev Khudanpur. 2010. Recurrent ` neural network based language model. In Eleventh annual conference of the international speech communication association.

[16] Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. Foundations and Trends R in Information Retrieval, 2(1–2):1–135.

[17] Kishore Papineni, Salim Roukos, Todd Ward, and WeiJing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting on association for computational linguistics, pages 311–318. Association for Computational Linguistics.

[18] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

[19] Richard Power and Donia Scott. 2005. Automatic generation of large-scale paraphrases

[20] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. nature, 529(7587):484.

[21] Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. Journal of documentation, 28(1):11–21.

[22] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In Advances in neural information processing systems, pages 1057–1063.

[23] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase level sentiment analysis. In Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing.

[24] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In Thirty-First AAAI Conference on Artificial Intelligence.

[25] Yizhe Zhang, Zhe Gan, and Lawrence Carin. 2016. Generating text via adversarial training. In NIPS workshop on Adversarial Training, volume 21.