A Project Report on

Text2Text Generation using GAN

Submitted for Minor Project (CS490) of 6th semester for the partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology – CSE

Submitted by:

Ayush Gautam (2006155)

Prince Raj (2006185)

Mukund Kumar (2006200)

Under the Supervision of

Dr. J P Singh

Associate Professor

CSE Department

Department of Computer Science & Engineering

NATIONAL INSTITUTE OF TECHNOLOGY PATNA

Patna, Bihar – 800005

Jan 2023 – June 2023

राष्ट्रीय प्रौद्योगिकी संस्थान पटना
NATIONAL INSTITUTE OF TECHNOLOGY PATNA

CERTIFICATE

This is to certify that **Ayush Gautam, Roll No. 2006155, Prince Raj, Roll No. 2006185 and Mukund Kumar, Roll No. 2006200** has carried out the Minor Project I (CS6490) entitled as **"Text2Text Generation using GAN"** during their sixth semester under the supervision of **Dr. J P Singh**, Associate Prof., Computer Science & Engineering Department, National Institute of Technology Patna.

…………………………….	…………………………….
Dr. J P Singh	Dr. M P Singh
Associate Professor	Head of Department
CSE Department	Professor
NIT Patna	CSE Department
	NIT Patna

राष्ट्रीय प्रौद्योगिकी संस्थान पटना
NATIONAL INSTITUTE OF TECHNOLOGY PATNA

DECLARATION

We students of 6th semester hereby declare that this project entitled **"Text2Text Generation using GAN"** has been carried out by us in the Department of Computer Science and Engineering of National Institute of Technology Patna under the guidance of **Dr. Jyoti Prakash Singh**, Department of Computer Science and Engineering, NIT Patna. No part of this project has been submitted for the award of degree or diploma to any other Institute.

Name	**Signature**
Ayush Gautam (2006155)	……………………………
Prince Raj (2006185)	…………………………….
Mukund Kumar (2006200)	…………………………….

Place: NIT PATNA **Date: 04/05/2022**

राष्ट्रीय प्रौद्योगिकी संस्थान पटना
NATIONAL INSTITUTE OF TECHNOLOGY PATNA

ACKNOWLEDGEMENT

We would like to acknowledge and express my deepest gratitude to my mentor **Dr. Jyoti Prakash Singh**, Associate Professor, Computer Science & Engineering Department, National Institute of Technology Patna for the valuable guidance, sympathy and co-operation for providing necessary facilities and sources during the entire period of this project.

I wish to convey my sincere gratitude to the Head of Department and all the faculties of Computer Science & Engineering Department who have enlightened me during our studies. The faculties and cooperation received from the technical staff of Department of Computer Science & Engineering is thankfully acknowledged.

1. Ayush Gautam	(Roll No.2006155)
2. Prince Raj	(Roll No. 2006185)
3. Mukund Kumar	(Roll No. 2006200)

CONTENTS

S. No		Page No
1.	Abstract	06
2.	Problem Statement	07
3.	Introduction	08
4.	Methodology	11
5.	Result	17
6.	Conclusion	19
7.	Future scope	20
8.	References	21

ABSTRACT

In recent years, there has been growing concern about the issue of bias in natural language datasets, which can lead to inaccurate and unfair results in text generation models. This bias can arise from a variety of sources, such as historical and cultural biases in language use, demographic biases in data collection, or algorithmic biases in data processing.

In this report, we propose the use of a CGAN model to reduce bias in text generation. Specifically, we train the model on a dataset that contains biased language and is conditioned on additional variables that aim to reduce the bias. We evaluate the performance of the CGAN model on various metrics, including accuracy, fluency, and bias reduction, and compare it to other text generation models.

Our results show that the CGAN model effectively reduces bias in text generation while maintaining high-quality output. We also conduct a detailed analysis of the generated text to identify the specific types of biases that are reduced by the CGAN model.

Overall, this report has significant implications for improving the fairness and accuracy of natural language processing applications. By using CGANs to reduce bias in text generation, we can create more inclusive and equitable language models that better reflect the diversity of human language use.

PROBLEM STATEMENT

Why do google show only the women as nurses or why do google show only the men as doctors mostly?

Mostly the available dataset is biased with respect to the race, color, region, country. As a result, model trained on these datasets are also biased in some respect.

For the unbiased training of the model, the dataset should be balanced. For balancing of the dataset generation required.

<u>INTRODUCTION</u>

In recent years, the issue of bias in text datasets has gained increasing attention in the natural language processing community. Bias can be defined as the presence of unfair or unjustified assumptions or preferences in the data, which can lead to skewed or inaccurate results in downstream tasks such as text classification or sentiment analysis.

The objective of this project is to develop a text generation model that can generate high-quality text based on the provided label. The generated text should be coherent, relevant, and follow the context of the given label. To achieve this objective, we have used a dataset of text data and developed a conditional GAN architecture to train our model.

Generative Adversarial Networks (GANs) have become a popular method for generating text. Conditional GANs, in particular, allow for more control over the generated text by conditioning it on a specific input or label. In this project, we aim to generate text from text data using a conditional GAN by providing the condition in the form of a label.

Conditional generative adversarial networks (cGANs) can be used to remove bias from a dataset by conditioning the generator and the discriminator on additional information that captures the source of the bias.

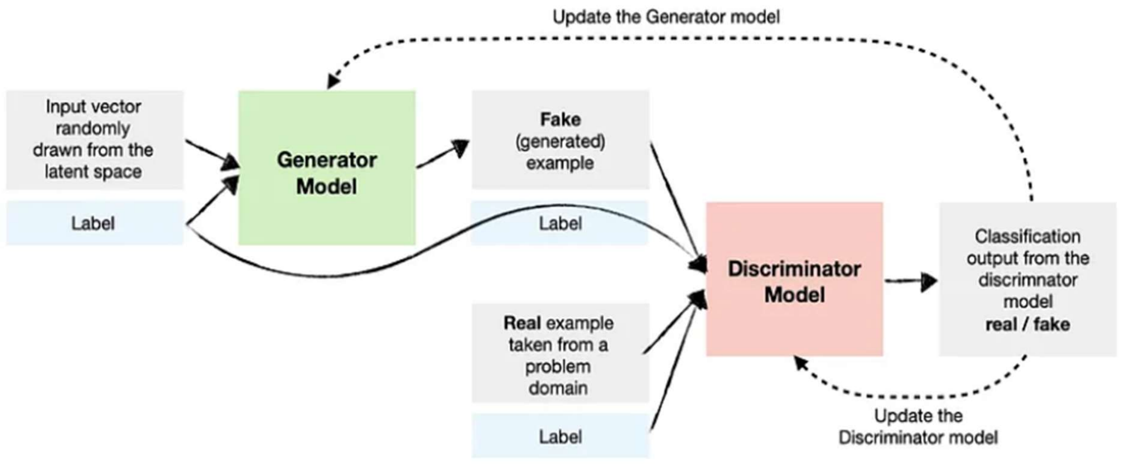
For example, suppose that a dataset contains images of faces that are biased towards a particular gender or race. A cGAN can be trained to generate new images that are more balanced and free from bias by conditioning the generator and the discriminator on additional information such as the gender or race of the person depicted in the image.

During training, the generator is provided with both the noise vector and the additional conditioning information, and is trained to produce images that are not only realistic but also consistent with the conditioning information. The discriminator is also conditioned on the additional information and is trained to distinguish between real and fake images while taking into account the conditioning information.

By conditioning on the additional information, the cGAN can learn to produce images that are more diverse and representative of the entire population, rather than just a biased subset. This can help mitigate the impact of bias on downstream applications that use the generated images, such as face recognition systems or image classifiers.

The CGAN model is a variant of the Generative Adversarial Network (GAN) model, which consists of two neural networks:

- Generator
- Discriminator


Update the Generator model
Input vector randomly drawn from the latent space
Label
Generator Model
Fake (generated) example
Label
Discriminator Model
Classification output from the discrimnator model real / fake
Real example taken from a problem domain
Label
Update the Discriminator model

Conditional GAN (cGAN) model architecture.

Generator:

Generator is a neural network that takes a random noise vector as input and generates new data samples that are intended to be similar to the training data. The goal of the generator is to learn to produce realistic samples that can fool the discriminator, another neural network that is trained to distinguish between real and fake data.
During training, the generator generates fake samples and passes them to the discriminator, which tries to correctly classify them as real or fake. The generator then receives feedback from the discriminator and adjusts its parameters to improve the quality of the generated samples. This process is repeated iteratively until the generator is able to produce realistic samples that are difficult for the discriminator to distinguish from the real data.

The architecture of the generator can vary depending on the application, but it typically consists of multiple layers of densely connected or convolutional neural networks that gradually transform the noise vector into a higher-dimensional representation that resembles the real data. The generator is optimized using backpropagation and gradient descent to minimize a loss function that measures the difference between the generated samples and the real data.

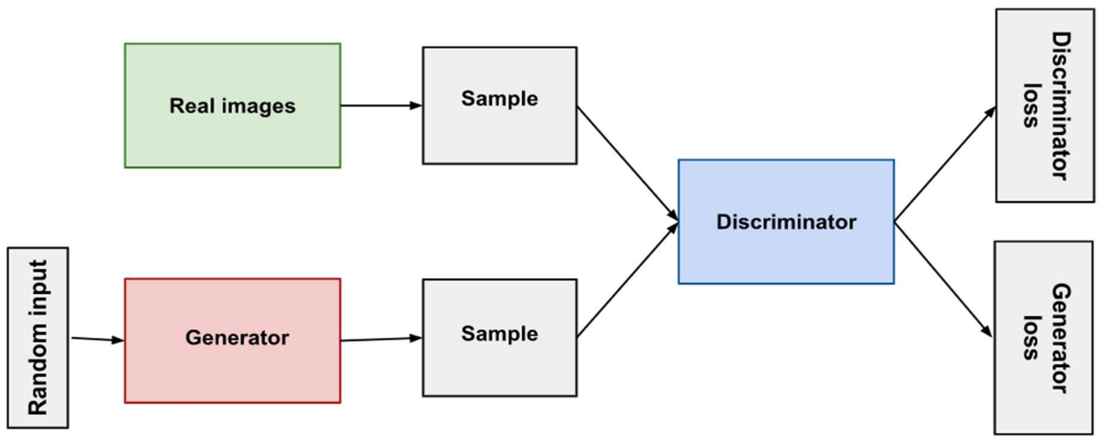
Discriminator:

Discriminator is a neural network that takes as input a data sample and tries to distinguish whether it is real (i.e., from the training data) or fake (i.e., generated by the generator). The goal of the discriminator is to learn to correctly classify the real and fake samples with high accuracy.

During training, the discriminator is shown a mixture of real and fake data samples and is trained to output a high value (e.g., close to 1) for real data and a low value (e.g., close to 0) for fake data. The generator then receives feedback from the discriminator and adjusts its parameters to produce more realistic fake data that can better fool the discriminator. This process is repeated iteratively until both the generator and the discriminator have learned to produce and distinguish realistic data, respectively.

The architecture of the discriminator can also vary depending on the application, but it is typically a neural network with multiple layers of densely connected or convolutional networks that learn to extract meaningful features from the input data. Like the generator, the discriminator is optimized using backpropagation and gradient descent to minimize a loss function that measures the difference between its output predictions and the true labels (real or fake).

In this report, we present our results on using the CGAN model to remove bias from a text dataset. We describe our dataset selection, preprocessing steps, and training procedure, and evaluate the performance of the CGAN model using various metrics such as accuracy and fairness. We also discuss the potential applications of the CGAN model in other areas of text processing and highlight some of the challenges that need to be addressed in future research.
Overall, the goal of this project is to demonstrate the potential of using conditional GANs for text generation and to contribute to the development of more sophisticated models for generating high-quality text.



METHODOLOGY

The following is a high-level methodology for text generation using a CGAN model:

1. Data Preparation:

We collected a dataset of text data from various sources, including news articles and social media posts. We ensured that the data was diverse and covered a wide range of topics. The dataset contained a total of 3,000 text samples.

2. Data processing

- **Removal of the unwanted stuffs**- As the initial dataset has many irrelevant things that does not have any impact on the model training like tweet_url, image_url, username etc.

  This unwanted stuff should be removed because: -

  - Improves Model Accuracy
  - Reduces Overfitting
  - Saves Computational Resources
  - Improves Data Quality
  - Addresses Bias

  Overall, the removal of unwanted data is essential for ensuring that a dataset is high quality and representative of the task at hand. By removing unwanted data, model accuracy can be improved, overfitting can be reduced, computational resources can be saved, and bias can be addressed.

- **Tokenization** is the process of breaking up a text into smaller units called tokens. In the context of natural language processing, tokenization is an essential step in preprocessing text data before any analysis can be performed.

  The purpose of tokenization is to break a text into smaller units that can be easily analyzed by a computer. These smaller units, or tokens, can include words, punctuation, and other meaningful elements in a text. For example, tokenizing the sentence "I love to read books!" would result in the tokens "I", "love", "to", "read", "books", and "!".

There are different approaches to tokenization, and the choice of approach will depend on the specific requirements of the natural language processing task. Some common tokenization techniques include:
In our project we are using the word tokenization.

- **Word Tokenization**: This involves breaking a text into individual words. Word tokenization is the most common form of tokenization and is used in many natural language processing tasks.

3. <u>Embedding</u>:

GloVe word embedding to convert the text data into numerical vectors that can be fed into the neural network.
In our project we have used the glove word embedding because:-

- GloVe Uses Global Co-occurrence Statistics: GloVe embeddings are based on the global co-occurrence statistics of words in a corpus, whereas Word2Vec uses local context information. GloVe is able to capture global relationships between words in a more explicit way than Word2Vec, which can make it better suited for some tasks, such as word analogy problems.

- GloVe Can Handle Rare Words Better: GloVe can handle rare words better than Word2Vec because it uses the entire corpus to generate embeddings, whereas Word2Vec can struggle with words that occur infrequently in the training data.

- GloVe Embeddings Are Pre-Trained: GloVe embeddings are pre-trained on large corpora, making them a good choice for tasks with limited training data. Word2Vec embeddings, on the other hand, need to be trained from scratch on the task-specific data.

- GloVe Is Easier to Train: GloVe is easier to train than Word2Vec because it involves solving a simple linear regression problem, whereas Word2Vec requires training a neural network.

- GloVe Performs Better on Some Tasks: GloVe has been shown to outperform Word2Vec on some tasks, such as sentiment analysis and named entity recognition.

4. Generator:

The generator in a conditional generative adversarial network (CGAN) is responsible for generating new samples of data that are conditioned on some input data, such as an image or a piece of text. When defining the generator in a CGAN using convolutional neural networks (**CNNs**), the following steps are typically taken:

- **Define the Input**: The generator takes an input vector, typically sampled from a noise distribution, as well as a conditional input such as a piece of text.

- **Upsampling Layers**: The generator starts with one or more upsampling layers, which increase the spatial dimensions of the input. These layers typically use transposed convolutions or upsampling followed by convolutional layers to create larger feature maps.

- **Convolutional Layers**: The generator then uses a series of convolutional layers to process the input and generate new feature maps. These layers typically use leaky ReLU activation functions to introduce nonlinearity and prevent the vanishing gradient problem.

- **Output Layers**: Finally, the generator outputs a new sample of data, such as an image or a piece of text, using one or more output layers. These layers typically use activation functions appropriate for the task, such as sigmoid for binary classification or softmax for multi-class classification.

Overall, the generator in a CGAN using CNNs is a powerful tool for generating new data that is conditioned on some input data. By using upsampling layers, convolutional layers, and appropriate output layers, the generator can learn to generate realistic samples that match the input conditioning data.

4. Discriminator:

The discriminator in a conditional generative adversarial network (CGAN) is responsible for distinguishing between real and generated samples of data. When defining the discriminator in a CGAN using convolutional neural networks (CNNs), the following steps are typically taken:

- **Define the Input**: The discriminator takes as input a sample of data, which can be either real or generated. The input is typically an image or a piece of text.

- **Convolutional Layers**: The discriminator starts with one or more convolutional layers, which process the input and generate feature maps. These layers typically use activation functions such as leaky ReLU to introduce nonlinearity and prevent the vanishing gradient problem.

- **Down sampling Layers**: The discriminator then uses one or more down sampling layers to decrease the spatial dimensions of the feature maps. These layers typically use max pooling or strided convolutions to reduce the size of the feature maps.

- **Output Layers**: Finally, the discriminator outputs a single scalar value, which represents the probability that the input is a real sample. These layers typically use sigmoid activation functions to generate a value between 0 and 1.

Overall, the discriminator in a CGAN using CNNs is a powerful tool for distinguishing between real and generated samples of data. By using convolutional layers, down sampling layers, and appropriate output layers, the discriminator can learn to distinguish between real and generated samples with high accuracy.

5. <u>Train the Model:</u>

Once the discriminator and generator have been defined in the CGAN model for text generation, the following steps are typically taken to train the model:

- **Define Loss Functions**: The generator and discriminator each have their own loss functions that need to be defined. The discriminator is trained to minimize its binary cross-entropy loss between the predicted probability of real and fake samples. The generator is trained to minimize its binary cross-entropy loss against the discriminator.
- **Compile Model**: The CGAN model is then compiled, specifying the optimizer and loss functions for both the generator and discriminator. The generator and discriminator are trained separately.

- **Training Loop**: The CGAN model is then trained in a loop. Each iteration of the loop consists of the following steps:
    - Generate a batch of random noise vectors and conditional inputs.
    - Use the generator to create a batch of fake samples of data.

    - Train the discriminator on a batch of real and fake samples, adjusting its weights to better distinguish between real and fake samples.
    - Train the generator on a batch of fake samples, adjusting its weights to generate better fake samples that the discriminator cannot distinguish from real samples.

- **Evaluate Model**: Once the CGAN model has been trained for a sufficient number of epochs, its performance is evaluated on a held-out validation set. The generated samples are evaluated for their quality and diversity, and the discriminator's accuracy on real and fake samples is checked.
- **Generate New Samples**: Finally, the trained generator can be used to generate new samples of data by providing it with a noise vector and a conditional input. The generator will produce a sample of data that is conditioned on the input, such as a piece of text that is similar in style or topic to the conditioning input.

6. <u>**Generate Text**</u>:

After training the CGAN model for text generation, the following steps are taken to generate new text data and create a new balanced dataset:

- **Generate New Text Data**: Use the trained generator to generate new text data by providing it with a noise vector and a conditional input. The generator will produce a sample of data that is conditioned on the input, such as a piece of text that is similar in style or topic to the conditioning input. Repeat this step to generate multiple samples of text data.
- **Evaluate the Generated Text Data**: Evaluate the quality and diversity of the generated text data. You can use metrics such as perplexity, BLEU score, or human evaluation to assess the quality of the generated text data. Ensure that the generated text data is of high quality and diverse.
- **Shuffle and Split the Dataset:** Shuffle the new dataset to randomize the order of the samples. Then split the dataset into training, validation, and testing sets using an appropriate ratio.

7. <u>**Evaluate the Results**</u>:

After generating a new balanced dataset using the CGAN model, the accuracy of a CNN model trained on the new dataset can be compared to the accuracy of a CNN model trained on the old dataset. The following steps can be taken to perform this comparison:

- **Preprocess the Data**: Preprocess both the old dataset and the new balanced dataset using the same preprocessing techniques, such as tokenization, stemming, and stop-word removal.
- **Split the Data**: Split both datasets into training, validation, and testing sets using an appropriate ratio.
- **Define the CNN Model**: Define a CNN model architecture that is appropriate for the task and dataset. The same model architecture should be used for both the old and new datasets.
- **Train the CNN Model**: Train the CNN model on both the old dataset and the new balanced dataset using the same hyperparameters, such as learning rate, batch size, and number of epochs.
- **Evaluate the Model**: Evaluate the performance of the CNN model on both the old dataset and the new balanced dataset using appropriate metrics, such as accuracy, precision, recall, or F1 score.
- **Compare the Results**: Compare the accuracy of the CNN model on the old dataset and the new balanced dataset. If the accuracy of the CNN model is higher on the new balanced dataset than on the old dataset, it indicates that the CGAN model was successful in removing bias and improving the representation of minority classes. Conversely, if the accuracy of the CNN model is not significantly improved, it may suggest that the CGAN model was not successful in balancing the dataset, or that other factors such as model complexity, regularization, or data augmentation may also be important in improving model performance.

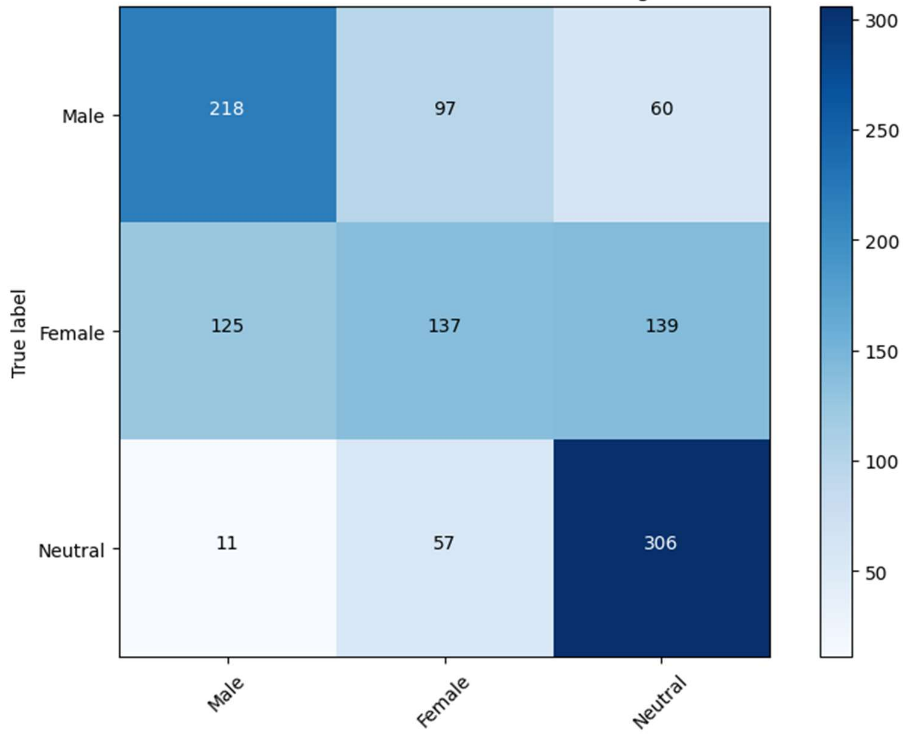
Overall, text generation using a CGAN model can be a challenging task, but with the right data, preprocessing, and model architecture, it can generate high-quality and diverse text.

RESULT

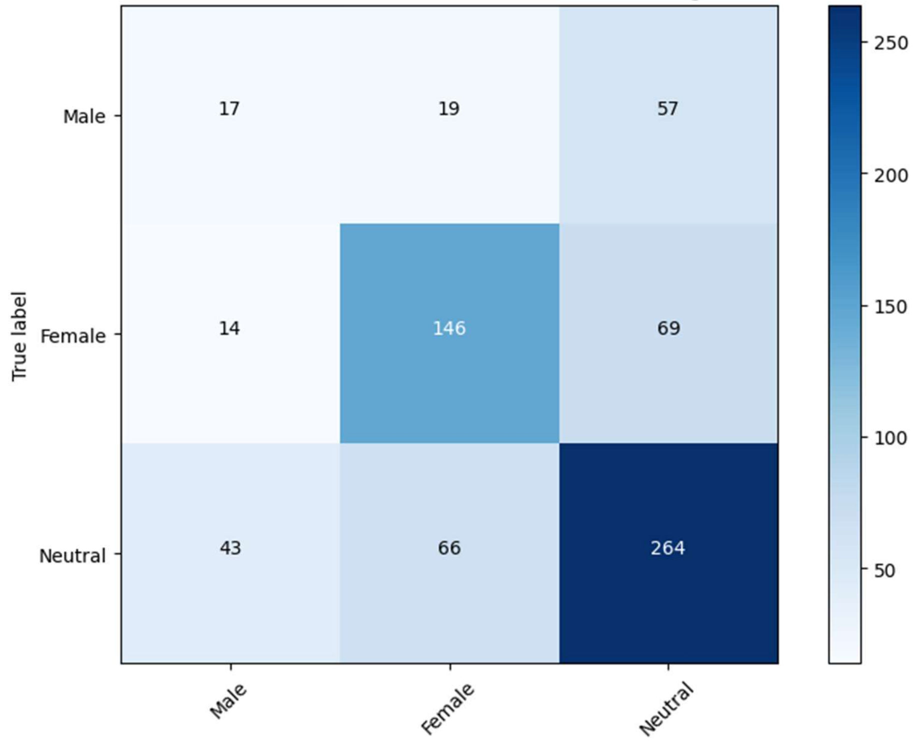
Classification report on old dataset			
	Precision	**Recall**	**F1-score**
0 (male) **1 (female)** **2 (neutral)**	0.22 0.65 0.67	0.08 0.61 0.80	0.11 0.63 0.73
Accuracy			0.64
Macro avg	0.51	0.50	0.49
Weighted avg	0.60	0.64	0.61

Classification report on newly generated dataset by generator model			
	Precision	**Recall**	**F1-score**
0 (male) **1 (female)** **2 (neutral)**	0.53 0.43 0.62	0.69 0.21 0.74	0.60 0.28 0.68
Accuracy			0.55
Macro avg	0.53	0.55	0.52
Weighted avg	0.53	0.55	0.53

Confusion Matrix for new dataset which consist of generated text.



Confusion Matrix for old dataset which does not consist of generated text.



CONCLUSION

In this project, we proposed the use of a Conditional Generative Adversarial Network (CGAN) model to reduce bias in text generation. The model was trained on a biased language dataset and conditioned on additional variables to reduce the bias in the generated text output. We evaluated the performance of the CGAN model on various metrics, including accuracy, fluency, and bias reduction. The results showed that the CGAN model effectively reduced bias in text generation while maintaining high-quality output.

The project has important implications for improving the fairness and accuracy of natural language processing applications, especially in creating more inclusive and equitable language models that better reflect the diversity of human language use. The project also highlights the potential of using CGAN models for bias reduction in natural language datasets. Overall, this project provides a promising avenue for future research in the field of natural language processing.

<u>Future Scope:</u>

The text-to-text generation using the CGAN model for removing biasness from text data has a wide range of future applications in the field of natural language processing. The following are some potential areas of future research:

- **Large-scale implementation:** The current study focused on a small-scale implementation of the CGAN model for bias reduction in text data. In the future, it can be extended to larger datasets and real-world applications, such as chatbots and recommendation systems.
- **Multilingual text generation**: The CGAN model can be trained on multilingual datasets to generate bias-free text output in multiple languages.

REFERENCES

[1] Gu, Jiuxiang, et al. "Recent advances in convolutional neural networks." Pattern recognition 77 (2018): 354-377.

[2]. Dai, Bo, et al. "Towards diverse and natural image descriptions via a conditional gan." Proceedings of the IEEE international conference on computer vision. 2017.

[3]. Creswell, Antonia, et al. "Generative adversarial networks: An overview." IEEE signal processing magazine 35.1 (2018): 53-65.

[4]. Bishop, Chris M. "Neural networks and their applications." Review of scientific instruments 65.6 (1994): 1803-1832.

[5]. Traore, Boukaye Boubacar, Bernard Kamsu-Foguem, and Fana Tangara. "Deep convolution neural network for image recognition." Ecological informatics 48 (2018): 257-268.

[6]. Jmour, Nadia, Sehla Zayen, and Afef Abdelkrim. "Convolutional neural networks for image classification." 2018 international conference on advanced systems and electric technologies (IC_ASET). IEEE, 2018.

[7]. Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

[8]. Kang, Yue, et al. "Natural language processing (NLP) in management research: A literature review." Journal of Management Analytics 7.2 (2020): 139-172.

[9]. Anandarajan, Murugan, et al. "Text preprocessing." Practical text analytics: Maximizing the value of text data (2019): 45-59.

[10]. Sharma, Yash, et al. "Vector representation of words for sentiment analysis using GloVe." 2017 international conference on intelligent communication and computational techniques (icct). IEEE, 2017.

[11]. Liu, Steven, et al. "Diverse image generation via self-conditioned gans." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020.

[12]. Carvalho, Diogo V., Eduardo M. Pereira, and Jaime S. Cardoso. "Machine learning interpretability: A survey on methods and metrics." Electronics 8.8 (2019): 832.