2022 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)

Concatenate Word Embedding for Text to Image through Generative Adversarial Network

1st Rakhmi Khalida
Department of Informatics Engineering
Gunadarma University
Depok, Indonesia
rakhmikhalida7@gmail.com

2nd Sarifuddin Madenda
Department of Informatics Engineering
Gunadarma University
Depok, Indonesia
sarifuddinmadenda@gmail.com

3rd Suryadi Harmanto
Department of Informatics Engineering
Gunadarma University
Depok, Indonesia
suryadi.hs@gunadarma.ac.id

4th I Made Wiryana
Department of Informatics Engineering
Gunadarma University
Depok, Indonesia
mwiryana.mobile@gmail.com

2022 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS) | 978-1-6654-7327-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICIMCIS56303.2022.10017727

***Abstract*— Since the explosion of deep learning, automatic image generation from natural language is highly desirable through artificial intelligence (AI) as it makes it easier for users to create visually rich images through the ease of language. One of the methods used to generate images from text is GAN, in this study using word embedding is to process natural language and develop GAN by concatenate word embedding. In this work, we carry out our early-stage research which is to explore the simple technique of concatenate word embedding as the input of the GAN neural network. We show that this model is a novelty for the GAN model with the concept of multimodal input that is able to generate text to image and is expected to improve performance on a stronger GAN. Based on the explanation of our research method, this model can be implemented and can be developed for various GAN tasks such as style transfer, image to image, face inpainting or image repair semantically, and super resolution.**

Keywords*— *text, image, word embedding, GAN

I. INTRODUCTION

Since the explosion of deep learning has begun, many researchers have begun to build many neural network architectures. It is often speculated that neural networks are inspired by neurons and their networks in the brain. Artificial Neural Network (ANN) algorithms often mimic and copy these biological structures, but much remains to be discovered about how the brain works [1]. The Artificial Neural Network (ANN) Algorithm simply cannot solve the mystery of the brain. That is why artificial intelligence scientists have to invent multiple neural network architectures to accomplish different tasks.

A picture is worth a thousand words, while the text on a picture makes a description or storyline clear. Such abilities are essential in generating creative thinking for the visual arts. Machines cannot have such capabilities, this can be formulated into the text to image. Text to image is the creation of an image from a text description. Text to image is a complex task of computer vision and deep learning which has made great progress over the last few years. Automatic image generation from natural language or generating images from text allows users to visually depict rich elements through text descriptions. Such capabilities are effectively highly desirable to be created through artificial intelligence (AI). Advances in deep learning have resulted in network architectures that promise to produce realistic images.

One of the algorithms that can generate images from text is the Generative Adversarial Network (GAN). GAN is an artificial neural network created [2], which can generate realistic images or artistic images such as paintings. In recent years, GANs have been developed to perform increasingly complex tasks, such as style transfer[3], [4], image-to-image [5], [6], human faces [7], face inpainting or semantic image repair [8], [9], text to image [10], super-resolution [11], representation learning [12], and data augmentation [13]. The proprietary Deep Convolution Generative Adversarial Network (DC-GAN) architecture [10] pioneered the algorithm for text-to-image using GAN. DC-GAN is a multimodal learning model that tries to bridge the two unsupervised machine learning algorithms, namely RNN and GAN neural networks to generate text to image. The RNN network serves to represent text into vectors for GAN network input. DC- GAN's contribution is to produce reasonable but effective images of natural language. The DC-GAN algorithm was tested on the Caltech-UCSD Birds datasets. Each image in this data set has five text descriptions.

The main work in generating images from text is to represent natural language in the form of text information in vector form and enter it into the GAN neural network. Text representation can generally be expressed in a distributed form, where the semantics are spread into the vector dimensions through matrix dimension reduction or neural network dimension reduction, which is called word embedding.

Mikolov et al from Google [14] released the word vector training toolkit Word2vec for expressing word semantic information. Word2vec is a static word embedding model that can get words from a predicted local context by maximizing conditional probabilities. Pennington et al. [14] launched a word vector model with the name Global Vector (GloVe) to create word vectors based on global information. Glove and word2vec is a static word embedding that can be used to handle word similarity, word analogy, and named entity recognition.

Previous studies often used single static word vectors which could not take into account multipart feature information from sentences. In our research, we propose a way to build a more robust GAN that can generate images from text by using the concatenate word embedding method. This research is still in its early stages, the result of this research is a word embedding design that is combined as input from the GAN neural network. We show that this model is a novelty for the GAN model with the concept of multimodal input capable of generating text to image. This research use dataset is grouped into 8K images for training and 3K images for testing.

This work was supported in full by Indonesia Endowment Fund for Education (LPDP) Ministry of Finance Republic Indonesia.

This paper is structured as follows, section 2 contains similar work done with word embedding and GAN networks that can generate images from text. We present word2vec and glove approach methods and the GAN architecture in section 3, In Section 4, we describe the experimental setup and the data set used. The results of this study are presented in Section 5, and section 6 is the conclusions of this paper.

II. Related Work

The development of GPU computing and significant advances in hardware performance has pushed the task of generating images from text with GAN deep learning technology to attract people's attention. Since GAN's inception in 2014 [1], the task of generating images from text with GANs has attracted more and more attention, and many GAN-based models have been developed.

In general, the GAN architecture consists of 2 networks which are referred to as generator and discriminator networks. The shape of the generator network can be seen in contrast to the structure of the neural network in general. The generator network uses a deconvolutional layer whose task is to change the dimensions of the input data into a larger dimension (Upsampling). The output of this generator is a 3-dimensional matrix which will then be referred to as data generation. A generator network usually consists of a fully connected layer, dense layer, batch normalization, Leaky ReLu, and reshape layer.

The discriminator network is a binary classification network that accepts three-dimensional image input and outputs a classification value of 0 or 1. The classification of the discriminator is an assessment of the generator in generating the generated image. If the generated image is like the original dataset then the discriminator value is 1 and vice versa.

The way GAN works to create generative images is through the adversarial process. Two models, namely generator G and discriminator D are designed to compete with each other. G tries to imitate generating a sample distribution, while D tries to judge the data comes from G or which is sample data of the dataset. The results of the assessment are then calculated with an optimizer function or loss function to optimize the network. Both can learn from the results. This process is executed many times. According to research [1], it is possible for the probability value of D is 0.5 to assess the data from the sample or from the results generator.

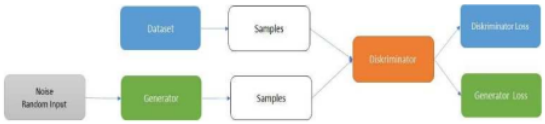

Fig. 1. The Step of Original GAN

Reed et al, were the first to develop GAN with the aim of generating images from text. His research uses RNN network to convert text input into character level so that it becomes code. Another contribution of his research is pairing original images with mismatched text other than original images with matching text and images generate with matching text. It aims to be able to generate images outside of the context of text input.

Another architecture developed by Gorti et al created MirrorGAN. The idea is to extend the cycleGAN architecture and add stacking, attention, and cycle consistency. MirrorGAN also uses word embedding to convert text input into word vectors, that code that feeds into the GAN generator [15].

Zhang et al built StackGAN by stacking two GANs to generate images from text. In his research, the stage-I GAN produced a low-resolution outline of the resulting image from the desired text and the stage-II GAN filled in the details of the image sketch. Another contribution from Zhang et al. is that the fully connected layer produces the mean and variance before taking text input and samples from the normal distribution. This sampling can add data and increase robustness [16].

The use of GAN and word embedding to generate text has been explored by Rajeswar et al [17]. This work tells the story by using embeddings at the sentence level allowing the model to represent the expression of an author in creating a text in a simple and effective way, rather than using word-level embedding.

Two important and widely used word embedding models because they have significant results in downstream NLP tasks are word2vec and glove embedding. Naderalvojoud et al. [18] applied pre-trained GloVe and Word2vec embedding vectors in their deep learning model and improved sentiment classification accuracy. Rezaeinia et al. [19] proposed to combine several word vector representations such as Word2vec, GloVe, POS2Vec, Lexicon2Vec, and Word-position2Vec to verify the accuracy of sentiment prediction.

Text-to-Face (TTF) synthesis has been carried out. In this paper, we propose a Text-to-Face model that not only produces images in high resolution (1024x1024) with text-to-image consistency and also displays multiple multifaceted faces to cover various facial features that are not naturally defined [20].

III. Method

The three stages carried out in this research are the stages of data loading, training, and generates. The steps of proposed system GAN can be seen in Fig 2.

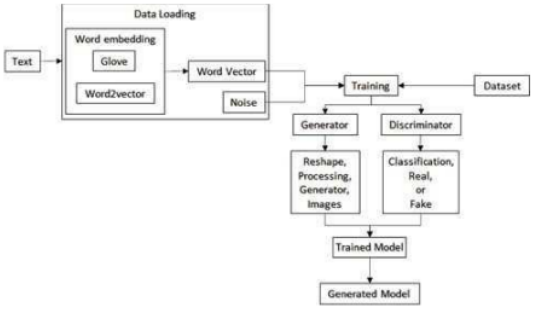

Fig. 2. Steps of Proposed System

In our formulation, not only noise as input to Generator, textual description is first converted into word vector by text insertion, then combined with random noise vector as input to Generator. For example, the textual description has been converted to a 256-dimensional insert and merged with a 100-dimensional noise vector [sampled from the Normal distribution. This formulation will help the Generator to generate an image that matches the input description and does not generate random images.

A. Data Loading

At the data loading stage, the text description is converted into word vector by word embedding, and the image data is converted into an array using a trained network such as Inception V3 CNN. Changes in text data into word vectors through the word embedding model, where each word is mapped to a different vector form. Word vector can be seen in fig 3.

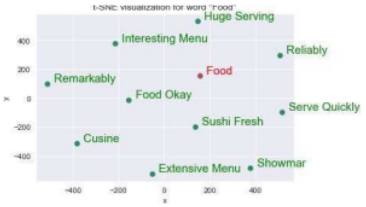

Fig. 3. Representation of Word Vector

B. Training

The training stage is the stage of the system recognizing image and text objects. Generators and discriminators are trained together, each of which may consist of a network layer consisting of a ResNet network (Residual Network).. The training process for a GAN means training the generator and discriminator simultaneously. The generator work to generate images that match the text description provided, while the discriminator evaluates the generated images based on the input of three pairs, namely the original image with matching text, generated image with matching text, and the original image with unmatched text. The process of training GAN is not easy, in training that uses adversarial and backpropagation methods, neither generator nor discriminator can win. When the generator wins from the discriminator there will be overfitting or producing the same generated image. When the discriminator wins from the generator, there will be no learning process from the generator. Hyperparameters on the GAN architecture are very important to set in the training process. In the training process using learning rate, batch, and epoch.

C. Generate

The generate stage is the stage after getting a trained model and is able to produce images according to text input, whether text is in the dataset or not. At this stage the system will create a checkpoint that is stored in the file directory. This checkpoint file can be reused someday to continue train GAN network with parameter update or not. The system will also save the Generator and Discriminator models in a predefined format such as h5. When this step is over, you can evaluate whether the GAN is performing well enough.

IV. Data, Tools, And Environment

This study using hardware with specifications for an Intel Core i7 processor, clock speed of 2.3 GHz, Nvidia Geforce RTX 3060 GPU engine graphics, 16GB RAM, and Win- dows 11 operating system with 64bit architecture. The soft- ware specifications used include python version 3, anaconda, jupyter, and google colab for the editor, then Pytorch, Tensor-Flow, tensorlayer, hard, and the python library.

The image and text description dataset used is Caltech-UCSD, the dataset overall contains 11k images of birds that have been collected from the labeled bird in the wild. The dataset is grouped into 8K images for training and 3K images for testing, which have been described by special annotators, and the corpus dataset is used as word embedding to change the text description. to be word vector is Wikipedia and Gigaword 5 vector with 300 dimensions.

V. Implementation

A. Word Embedding

Word embedding is one of the Natural Language Processing (NLP) techniques in which words or phrases i.e., strings of vocabulary are mapped onto a vector of real numbers. The need to map strings to vectors of real numbers stems from the computer's inability to perform operations with strings. Since the one hot encoding method is inefficient and unique ID does not offer a relational representation, word embedding is able to see the relational representation of words and can represent each word with a vector that is simpler than one hot encoding. The final array size is also much smaller than the one hot encoding encoded vocabulary.

Word embedding can adopt a supervised, unsupervised or semi-supervised learning approach. The word embedding model that will be used is Word2vec and glove. The Word2vec and glove models will create an embedding matrix for all words in the corpus. The word2vec model contains a matrix of vector length with a fixed vector size while glove builds a large matrix by taking into account the global word for word count in the corpus.

Word2vec and glove are word embedding types of static words with an unsupervised approach. This is because to find the ideal vector value for each word and usually use language models to predict the next word based on its context. The word embedding structure can be seen in the Fig 4.

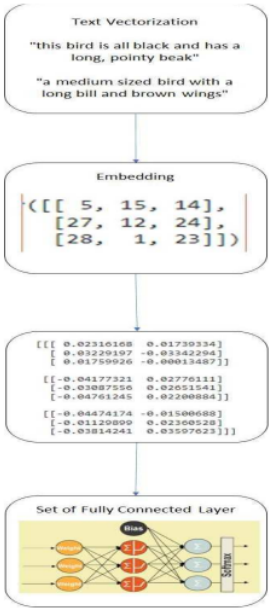

Fig. 4. Word Embedding Structure

The explanation of the word embedding structure in the image above is as follows:

1) *Text Vectorization Layer* to convert text to 1D integer tensor This layer performs the following procedure:

a) Change every uppercase letter in a sentence to lowercase.

b) Remove every punctuation.

c) Turning sentences into words.

d) Merge words into tokens.

e) Doing token index.

f) Converts each sentence using an index into a 1D tensor of an integer token index or float value.

2) Embedding layer to convert 1D integer tensor to solid vector with a fixed size.

3) *Neural network* fully connected layers with backpropagation algorithm and loss function as well as other networks. The values that have been obtained are still very random. Therefore, we need an optimizer to calculate the loss function and adjust its value with backpropagation and this optimizer must run on a set of layers called fully connected layers such as LSTM, GRU, or Convolution layers. Using a set of fully connected layers can insert strings into labels to match vector values.

B. Concatenate Word Embedding

First, in the word2vec model, training was conducted on the Wikipedia corpus with skip-gram architecture, 300 dimensions in size, 3 threads for multiprocessing, and window size 3. The use of a smaller window will give a more specific term to a word context. The frequency of occurrence of words or called min count is 1. Words that rarely appear are usually not very important, so the model will ignore words that do not meet the min-count value.

The word embedding glove model was trained on a Giga-word 5 vector corpus with memory parameters is 4, verbose 2, size 300, and binary is 2. Verbose is a value that tells the function how much information to print when training the model, the larger the value, the more detail. generated information. Memory in glove and thread in word2vec both serve as a measure of memory usage used during model training. Binary is an option to output the model, a value of 0 means text output type, 1 value for binary output type, and 2 value for text and binary output. This parameter is more influential for the efficiency of memory usage and model size. Both word embedding models are trained using CNN networks because using a set of fully connected layers can insert strings into labels to match vector values. and the backpropagation algorithm is needed as an optimizer that calculates the loss function and adjust its value to produce a better semantic value.

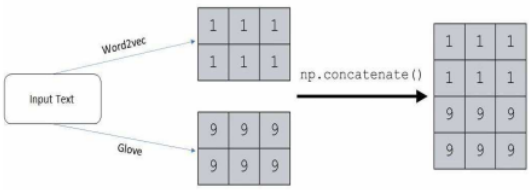

Fig. 5. Illustration of Concat Word Embedding

Inspired by the form of a word vector is the fixed array, for example, 50, 100, or 300 dimensions, which when concatenate with the model, number of arrays becomes the total dimensions of the sum of the two models. The process of concatenate two models word embedding into one model can be seen in the Fig. 5.

C. Text to Image with GAN

The GAN architecture for this model consists of a GAN neural network that has a multimodal input consisting of images and text. The combination of word embedding as a text modifier and together with images becomes input for the generator and will be judged genuine or fake by the discriminator in the GAN.

Details of the GAN generator network consists is 5 layers of deconvolution layer. The generator works to change the dimensions of the input data into a larger dimension. The generator consisting of a 2-dimensional deconvolution layer receives input in is noise, this noise will go through a dense layer, batch normalization, RELU, and the output layer is a reshape layer. This is repeated in up to 4 parts. The output of this generator is a 3-dimensional matrix which will then be referred to as data generation. An image of the structure of the generator can be seen in Fig 6.

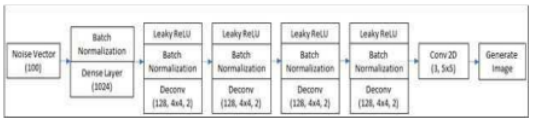

Fig. 6. Structure of Generator

The discriminator consists of 5 convolutional layers, the image that is input to the discriminator will be entered in the first part consisting of a 2-dimensional convolutional layer. Leaky RELU, and drop out, then at the end, the data will go through a flattened, dense, fully connected layer, so that it produces 1 value in the form of a decision. If the data entered is an image from the dataset, the discriminator produces a value close to 1, vice versa. Image of the structure of the discriminator can be seen as fig 7.

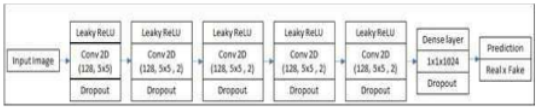

Fig. 7. Structure of Discriminator

Detailed image of GAN architecture with multimodal input using concatenate word embedding is described as fig 8.

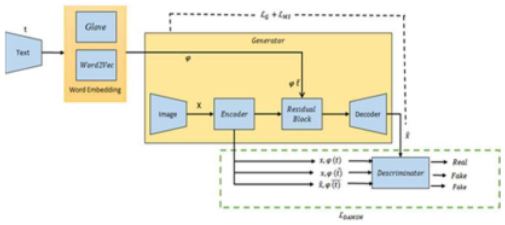

Fig. 8. Proposed Structure of GAN

In the process of training the GAN network, there is a loss function that functions to measure the balance between the power of the generator and the power of the discriminator. it's a min-max optimization formulation where Generator wants

to minimize the objective function whereas Discriminator wants to maximize the same objective function. The weakness in the GAN that often occurs is the collapse or overfitting mode. The loss function generator is made maximal so that it does not lose with the discriminator loss function.

This min-max formulation of the objective function has a global optimum. The loss function in this research adopts AttnGAN's property[21]. DAMSM loss function is able to calculate the similarity between the resulting image and the sentence using global sentence level information (t) and detailed word level infor- mation (w). DAMSM loss function measures the degree of match between the image and the text description. Based on the DAMSM loss function, the discriminator classification the image according to 3 categories:

1) *Original image* with matching text $D(x, \varphi(t))$
2) *Generated image* with matching text $D(x^{\wedge}, \varphi(\overline{t}))$
3) *Original image* with mismatched text $D(x, \varphi(\overline{t}))$

The described GAN architecture has the basic parameters of DCGAN [10]and adopts AttnGAN's DAMSM loss function [10]. DAMSM loss function is designed to study model attention in a semi-supervised manner, where the only supervision is matching between the all of picture and the all of sentence (order word). A collection of pairs of pictures-description $(Q_i, D_i)^M$. Posterior the probability of a Di sentence being paired with a Qi image is calculated as

$$P(Qi, Di) = \frac{exp(\gamma R(Qi,Di)}{\sum_{i=1}^{M} exp(\gamma R(Qi,Di)} \quad (1)$$

Substitute value for R(Q, D) is

$$R(Qi, Di) = \log \sum_{i=1}^{T-1} \exp(\gamma \mathrm{R}(\mathrm{ci}, \mathrm{ei})))^{\gamma 2} \quad (2)$$

The following is loss function formula for DAMSM based (1) and (2)

$$L_{DÆMSM} = L^{W} + L^{W} + L^{t} + L^{t} \quad (3)$$

when,

$$L^{w} = -\sum_{i=1}^{M} \log P(Qi|Di) \quad (4)$$

Also applies to Lw and if (2) defined with $R = (Q|D) = \frac{(\bar{v}\mathrm{T}\bar{e})}{(||\bar{v}||||\bar{e}||)}$ then subtitution to (1) and (4) is obtained L_1^t and L_2^t.

The following is loss function formula for generator:

$$L_G = -\frac{1}{2}[E_{x\sim pG} \log(D(x)) + E_{x\sim pG} \log(D(x, \varphi(t)))] \quad (5)$$

When the input is a noise distribution, many probabilities are generated due to vector differences. Some exist mode is preferred over other modes. The mode-seeking loss function tries to amplify the distribution of minor modes between major modes. When the noise distribution is used to generate two different samples based on the same input. The first point z1 causes the formation of a mode image I1 = G(c, z1) and the second point z2 points to the image I2 = G(c, z2) after applying generate image and text (c). D(. , .) is a distance function that calculates the distance between two tensors or vectors. The numerator calculates the distance between object vectors while the denominator is the distance between ordinary vectors. The purpose of this formula is to maximize the distance between vector objects. The following is the MSGAN loss function formula.

$$L_{MS} = \max G\left(\frac{\mathrm{DI}(\mathrm{G}(\mathrm{c},\mathrm{Z1}),\mathrm{G}(\mathrm{c},\mathrm{Z2}))}{\mathrm{D_Z}(\mathrm{Z_1},\mathrm{Z_2})}\right) \quad (6)$$

Based on (1) until (6), The calculation of the loss function formula is (7)

$$L = \sum_{\mathrm{i}} L_G + L_{DÆMSM} + l_{MS} \quad (7)$$

The structure of the GAN architecture that adopts DCGAN [10], the difference between the input generators is the distribution of noise and text. The text encoder does not use word embedding but a Skip-thought model[22]. Adoption of the DCGAN structure as it mimics the workings of discriminator networks studying the appropriate relationship between description of text and image content with GAN-CLS and GAN-INT function.

At the start of the training, the discriminator ignores the text and easily reject the image generated from the generator, because it doesn't look reasonable, but after the generator receives feedback update weight from the discriminator, the generator learns to generate a reasonable generated image and learns to align it with the text, the the discriminator also learn to evaluate the resulting image to satisfy this conditioning problem. Based on this intuition it may complicate the dynamics of learning, but GAN-CLS pays attention to this by adding an alternate decision to the discriminator, namely a third input consisting of the original image with mismatched text. It is in accordance with the DAMSM loss function.

GAN-INT works with word embedding on the generator side. GAN-INT utilizes manifold interpolation learning to generate additional text for word embedding during training. This matter serves to make the generator stronger to generate images and can deceive the discriminators from being able to distinguish the output. The results of the added text to be inserted do not need to match the facts so that GAN-INT does not require a cost.

VI. CONCLUSION

Word embedding glove and word2vec are static word models which are the most widely used word embedding. The availability of pretrained word embedding models and with vocabulary area coverage can reduce the high computational requirements for training word embedding models. In this work, we explore a simple technique, namely concatenate word embedding, which contributes to developing GAN for the task of generating images from text and is expected to improve performance on a more robust GAN. Based on the explanation of our research method, this model can be implemented and can be developed for various GAN tasks such as style transfer, image to image, face inpainting or image repair semantically, and super resolution.

ACKNOWLEDGMENT

Thank to Indonesia Endowment Fund for Education (LPDP) Ministry of Finance Republic Indonesia have been made possible this research and presentation.

REFERENCES

[1] Martin C Nwadiugwu, "Neural Networks, Artificial Intelligence and the Computational Brain," *ArXiv*, 2021.
[2] Ian J. Goodfellow *et al.*, "Generative Adversarial Nets," 2014.
[3] Y. Jing, Y. Yang, Z. Feng, J. Ye, and M. Song, "Neural style transfer," *IEEE Trans Vis Comput Graph*, 2019.
[4] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEEComputer Vision and Pattern Recognition*, 2016, pp. 2414–2423.

[5] J.-Y. Z. T. Z. A. A. E. P. Isola, " Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE Computer Vision and Pattern Recognition*, 2016, pp. 1125–1134.
[6] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2223–2232.
[7] T. A. S. L. J. L. T. Karras, "Progressive growing of gans for improved quality, stability, and variation," 2018.
[8] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Free- form image inpainting with gated convolution," in *Proceedings ofthe IEEE International Conference on Computer Vision*, 2019, pp.4471–4480.
[9] R. A. Yeh, C. Chen, T.-Y. Lim, A. G. Schwing, and M. N. M. Hasegawa Johnson, "Semantic image inpainting with deep generative models," in *Proceedings of the IEEE Computer Visionand Pattern Recognition*, 2016, pp. 5485–5493.
[10] Scott Reed and Zeynep Akata and Xinchen Yan and Lajanugen Logeswaran and Bernt Schiele and Honglak Lee, "Generative Adversarial Text to Image Synthesis," *Proc Mach Learn Res*, 2016.
[11] C. Ledig *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE Computer Vision and Pattern Recognition*, 2016, pp. 4681–4690.
[12] J. Donahue and K. Simonyan, "Large scale adversarial representation learning," *Advances in Neural Information Processing Systems*, pp. 10452–10552, 2019.
[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," 2013.
[14] R. S. and C. D. M. J. Pennington, "GloVe : Global Vectors for Word Representation," 2014.
[15] S. K. Gorti and J. Ma, "Text-to-Image-to-Text Translation using Cycle Consistent Adversarial Networks," Aug. 2018, [Online]. Available: http://arxiv.org/abs/1808.04538
[16] Han Zhang and Tao Xu and Hongsheng Li and Shaoting Zhang and Xiaogang Wang and Xiaolei Huang and Dimitris Metaxas, "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks." International Conference on Computer Vision (ICCV), 2017.
[17] Sai Rajeswar, Sandeep Subramanian, Francis Dutil, Christopher Joseph Pal, and Aaron C. Courville, "Adversarial generation of natural language," *CoRR*, 2017.
[18] Behzad Naderalvojoud and Ebru Akcapinar Sezer, "Sentiment aware word embeddings using refinement and senti-contextualized learning approach," *Neurocomputing*, vol. 405, pp. 149–160, 2020.
[19] Seyed Mahdi Rezaeiniaa, Rouhollah Rahmania, Ali Ghodsib, andHadi Veisia, "Sentiment analysis based on improved pre-trained word embeddings," *Expert Syst Appl*, vol. 117, pp. 139–147, 2019.
[20] A. D. Pavllo, Lucchi, and T. Hofmann, "Faces à la Carte: Text-to-Face Generation via Attribute Disentanglement," *European Conference on Computer Vision*, pp. 482–499, 2020.
[21] Tao Xu and Pengchuan Zhang and Qiuyuan Huang and Han Zhangand Zhe Gan and Xiaolei Huang and Xiaodong He, "AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks," *CoRR*, 2017.
[22] Ryan Kiros *et al.*, "Skip-Thought Vectors," *CoRR*, 2015.

Cek Plagiasi 12

ORIGINALITY REPORT

23%	16%	16%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

MATCH ALL SOURCES (ONLY SELECTED SOURCE PRINTED)

3%

★ Submitted to University of Melbourne

Student Paper

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		