Proceedings of 10th IOE Graduate Conference
Peer Reviewed
ISSN: 2350-8914 (Online), 2350-8906 (Print)
Year: 2021 Month: October Volume: 10

Synthesizing Human Face Image from Textual Description of Facial Attributes Using Attentional Generative Adversarial Network

Jupiter Tamrakar [a], Bal Krishna Nyaupane [b]

[a, b] *Department of Electronics and Computer Engineering, Pashchimanchal Campus, IOE, Tribhuvan University, Nepal*
Corresponding Email: [a] jupiter.752534@pasc.tu.edu.np , [b] bkn@wrc.edu.np

Abstract
GAN (Generative Adversarial Network) is a revolutionary network architecture that changed the landscape of unsupervised learning and photo-realistic image synthesis. Many types of research has been done for improving the stability of GAN and the quality of the image synthesized. Recent studies are focused on synthesizing images from text descriptions by conditioning the generator and discriminator on these text descriptions. This task falls under the domain of text to image synthesis. In this work, research focuses on implementing a GAN model, for synthesizing face images from a text description of facial attributes. To do so, the model must know what description best matches the image region and vice versa. Hence to accomplish this, the model must have an understanding of language and image. So, in this work, two pre-trained models BERT and Inception-v3, are used. From the pre-trained BERT model, we extract contextual word embedding and, from Inception-v3, image features. Now, using the information about both text and its image pair, the model can generate images accordingly.

Keywords
GAN, BERT, Bi-GRU, Bi-LSTM, Inception-v3, Image synthesis

1. Introduction

Developing systems for the synthesis of face images from text description have applications in various fields such as photo editing, photo manipulation, etc., including criminal investigations. Image generation tasks have been explored extensively since the introduction of GAN. After its success, a whole new domain of image synthesis has emerged. A subdomain of it is text to image synthesis, where the task is to generate images from the text description. One of the early works done for the synthesis of human face images from text description is Text2facegan [1]. Despite using a small dataset of 10,000 face images for training the GAN model, they were able to get promising results. In this work, a more recent GAN model named AttnGAN [2] is extended for text to face image synthesis. We used a pre-trained BERT model to provide word embedding for text encoder and pre-trained Inception-v3 to extract image features. For training and testing the proposed model, we sampled 12,000 text-image pairs from Multi-Modal-CelebA-HQ Dataset [3] which have 30,000 face images of size 1024 x 1024 with captions describing different attributes of the face.

2. Related Works

The introduction of GAN [4] has popularized research in the field of image synthesis. Some researchers focused on increasing the stability of GAN while minimizing hyperparameter sensitivity by introducing a new concept of progressive increment of the layers [5]. Recently more researchers are interested in synthesis of images semantically aligned with given text description. Moreover the researches are mainly focused on synthesizing images of birds, flowers etc. Following are summary of previous works for image synthesis from a text description -

Nasir et al. [1] introduced the use of DC-GAN [6] with matching-aware discriminator for synthesizing 64 x 64 face images from fine grained textual descriptions. Using matching-aware discriminator the network is able to learn relationship between matched and mismatched caption for synthetic and real image.

Reed et al. [7] introduced the use of GAN for synthesizing 64 x 64 image from text description along with training strategy such as matching aware discriminator (GAN-CLS) , learning with manifold interpolation (GAN-INT) and inverting the generator for style transfer. Using GAN-INT-CLS the network is able to increase variation in generated image for same caption and learn style variations by disentangling style and content.

Reed et al. [8] introduced the concept of bounding box conditioning and keypoint conditioning for synthesizing 128 x 128 image from text description. Using bounding-box-conditioning, the network is able to learn the location and position of objects and using keypoint conditioning, the network is able to learn what image to generate.

Zhang et al. [9] introduced the concept of multi stage generation of image and conditioning augmentation for synthesizing 256 x 256 photo-realistic images conditioned on text descriptions. Using multi-stage process, the network is able to refine details conditioned on text description and add resolution at each stage and randomness increased by conditioning augmentation helped network to learn various poses and appearances for same text caption.

Zhang et al. [10] improved multi stage training by interleaving multiple generators in tree structure and introduced simultaneous approximation of unconditional and conditional distribution. By interleaving multiple generators in tree structure, the training of network is stabilized and by simultaneously approximating unconditional and conditional distribution the network is able to learn both unconditional image generation and conditional image generation.

Xu et al. [2] introduced the concept of paying attention to relevant subregion for a word in text description and synthesizing fine-grained details at different subregion and also proposed deep attentional multimodal similarity model (DAMSM). Attention mechanism enabled the network to automatically select word level conditioning for generating different sub-regions of image and DAMSM helped network for understanding fine-grained text-image matching.

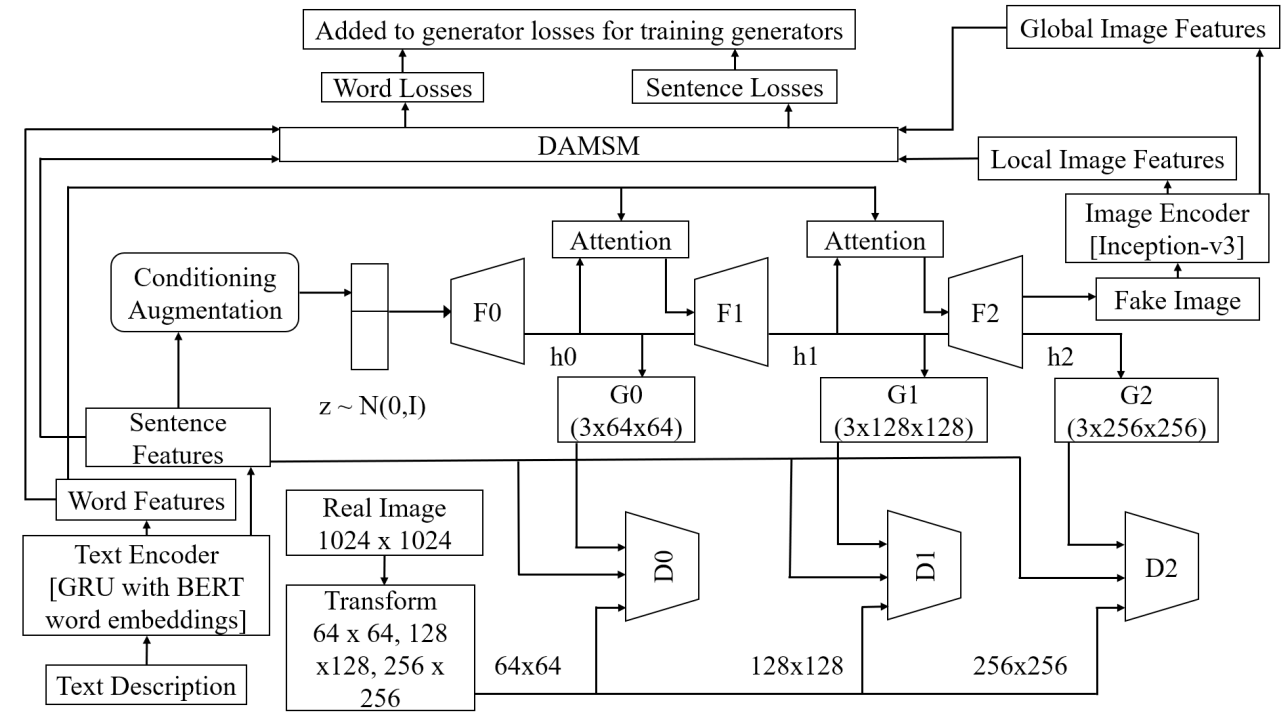
In this work, the research is focused on synthesis of human face images from textual description of facial attributes. Here, the AttnGAN architecture proposed in [2] is extended by using pretrained BERT model for providing word embedding to the text encoder. Also, text encoder is changed to Bi-GRU from Bi-LSTM. Previous work for human face image synthesis from text description [1] used skip-though vectors for providing embedding to text encoder. The concept of progressively increasing the layer [5] is also begin used in the form of stage-wise network where the network will generate images in multiple stages by increasing the number of layers and image size. And to provide variation in images and information about location of object in the image the sentence augmentation and word attention technique from AttnGAN architecture is used.

3. Methodology

We train the model in two phases. In the first phase, we train text and image encoders with real text-image pairs. For training, we used the loss from DAMSM model, equation 11, so that the text and image encoders have the knowledge of text matching the image and vice versa. Then in the second phase, the GAN model is trained using word features and sentence features from the text encoder and local image features, and global image features from the image encoder. Figure 1 shows the implemented model.

3.1 Implemented Model

Text Encoder The dataset has ten captions per image. We randomly select one caption during training. Here, pre-trained BERT is used to provide word embedding for the Bi-GRU encoder. Before getting the input vector for BERT, we do some text preprocessing. First of all, we remove punctuations in the caption then all the words are lower-cased. We used the BERT tokenizer to get tokens from the caption. After tokenizing the caption, we add [CLS] token at the beginning, [SEP] token at the end, and [PAD] tokens after [SEP] if the sequence length is less than the max length to make all sequence lengths the same. Then we convert the tokens to ids in BERT vocabulary. BERT has a vocabulary of 30,000 English words. The tokenizer handles the words not included in the BERT vocabulary by dividing them into sub-words all the way to letter level. For example, the tokenizer divides receding into subwords rec and ##eding. Here, as we do not require embedding for [CLS], [SEP], and [PAD], we make them zero. And as the actual tokens are divided into sub-words by the BERT tokenizer, we added the embedding of these sub-words to get the final embedding. Here, the



Z~N(0,I) – Noise drawn from Normal Distribution.
F0 – First Stage Generator Network.
F1 – Second Stage Generator Network.
F2 – Third Stage Generator Network.
G0, G1, G2 – Network to change image channel to RGB channel for output from F0, F1, F2 respectively.

h0 – F0 network hidden state output.
h1 – F1 network hidden state output.
h2 – F2 network hidden state output.
D0 – First Stage Discriminator Network.
D1 – Second Stage Discriminator Network.
D2 – Third Stage Discriminator Network.

Figure 1: GAN model.

pre-trained BERT used is bert-base-uncased which has 12 encoder layers and 768 hidden units. We used BertModel as pre-trained BERT with output_hidden_states set to True. We do this to get outputs from all layers. We extracted word embedding by summing hidden state outputs from the last four layers. For making our model simple, we used a linear layer to change the BERT dimension to the Bi-GRU input dimension. Now, we use a single-layer Bi-GRU encoder for encoding these embedding. We get word features from the encoder's cell output and sentence features from the last hidden state output. We use word features in the attention model, equation 14, to get a word context vector for a region in the image and also in the DAMSM model, equation 11, to calculate word losses. We use sentence features as input to the first generator network, Figure 3, in the DAMSM model to calculate sentence losses and to provide conditional loss for both discriminator and generator network. This encoder is trained in the first phase along with the image encoder using word and sentence loss from the DAMSM model.

Image Encoder We used a pre-trained Inception-v3 model as an image encoder which we train along with the text encoder during the first phase of training. Using inception-v3, we extract two features, local features and global features . We used the "mixed_6e" layer to extract local features which have (17 x 17x 768) dimensions which we reshaped to (768 x 289). Also, we used the last average pooling layer to extract global features which have (1 x 1 x 2048) dimensions which we reshaped to (2048). During generator training, we use this trained encoder to extract image features of the generated fake images from which we get DAMSM losses equation 11 necessary for getting total generator loss equation 17.

DAMSM Model Deep Attentional Multimodal Similarity Model (DAMSM) takes word-level features, sentence-level features, image local features, and image global features as input. This model provides loss which tells us whether or not the generated image follows the provided description. We first compute a similarity matrix between word-level features and available sub-regions (289) by simply

calculating the dot product.

$$s = e^T v,\ s\ \varepsilon\ \mathbb{R}^{T\times 289} \tag{1}$$

Then this score is normalized over all words.

$$\bar{s}_{i,j} = \frac{\exp(s_{i,j})}{\sum_{k=0}^{T-1}\exp(s_{k,j})} \tag{2}$$

where, the score is computed for i^{th} word and j^{th} region. Now, the normalized score is used to evaluate region-context vector for a particular word. Here, the intuition is for a word we are looking at all the sub-regions and we select sub-region which is best described by this particular word. So, for every word we evaluate this region-context vector c_i as weighted sum over all sub-regions.

$$c_i = \sum_{j=0}^{288}\alpha_j v_j,\ where\ \alpha_j = \frac{\exp(\gamma_1\bar{s}_{i,j})}{\sum_{k=0}^{288}\exp(\gamma_1\bar{s}_{i,k})} \tag{3}$$

where, γ_1 is a hyperparameter which controls the attention of i^{th} word for its relevant sub-region and α_j is a similarity score normalized for j^{th} sub-region and i^{th} word over all sub-regions. Then we compute cosine similarity between the word and its corresponding region-context vector. This gives us word level relevance $\mathrm{R}(c_i,e_i)$ for i^{th} word.

$$R(c_i,e_i) = \frac{c_i^T e_i}{(\parallel c_i \parallel\parallel e_i \parallel)} \tag{4}$$

As we are concerned with finding relevance between whole description D and the entire image Q, we compute relevance score $R(Q,D)$ between image and description. Now, first using word level relevance $R(c_i,e_i)$ the image description match score $R(Q,D)$ is given by-

$$R(Q,D) = \log\Big(\sum_{i=1}^{T-1}\exp\left(\gamma_2\, R(c_i,e_i)\right)\Big)^{(\frac{1}{\gamma_2})} \tag{5}$$

where γ_2 determines the importance of word for given region context vector and the similarity score is summed for all T words. Again, using cosine similarity between sentence level features ($\bar{e}$) and global image features ($\bar{v}$), the image description match score $R(Q,D)$ is given by-

$$R(Q,D) = \frac{\bar{v}^T\bar{e}}{(\parallel \bar{v} \parallel\parallel \bar{e} \parallel)} \tag{6}$$

Now, we calculate probability distributions so that we can find exactly which image match which description. These distributions are posterior probability of description D_i matching with image Q_i, $P(D_i/Q_i)$, and posterior probability of image Q_i matching with description D_i, $P(Q_i/D_i)$. These two posterior probabilities are -

$$P(D_i/Q_i) = \frac{\exp\left(\gamma_3 R(Q_i,D_i)\right)}{\sum_{j=1}^{M}\exp(\gamma_3 R(Q_i,D_j))} \tag{7}$$

$$P(Q_i/D_i) = \frac{\exp(\gamma_3 R(Q_i,D_i))}{\sum_{j=1}^{M}\exp(\gamma_3 R(Q_j,D_i))} \tag{8}$$

When we calculate posterior probabilities using word level relevance from Eq.(4) in image description score Eq.(5) we get two losses with respect to word features-

$$L_1^w = -\sum_{i=0}^{M}\log\left(P(D_i/Q_i)\right),\ \text{from 5 and 7} \tag{9}$$

$$L_2^w = -\sum_{i=0}^{M}\log\left(P(Q_i/D_i)\right),\ \text{from 5 and 8} \tag{10}$$

Similarly, when we calculate posterior probabilities using image description score Eq.(6) we get two losses with respect to sentence features L_1^s and L_2^s. Finally, the total loss we get for DAMSM is:

$$L_{DAMSM} = L_1^w + L_2^w + L_1^s + L_2^s \tag{11}$$

Conditioning Augmentation ($F^{ca}(\bar{e})$) provides augmented sentence level features for first stage generator (F_0). Here, augmentation is done by taking the mean μ and standard deviation σ of the sentence level features. Now, σ is multiplied with a noise ε drawn from normal distribution N(0,I). Finally, μ is added with this product and we get our final conditioning vector. This is done so that higher variation in generated images can be obtained for the same caption. So, this block takes sentence level features $\bar{e}$ from text encoder and outputs augmented vector c.

$$c = \mu + \sigma * \varepsilon,\ \ \varepsilon\ \sim\ N(0,I) \tag{12}$$

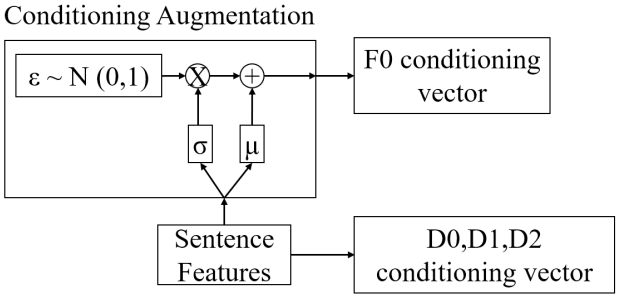


Figure 2: Conditioning Augmentation block.

Attention Model ($F_i^{attn}(e, h_{i-1})$) take inputs, word level features (e) and the hidden context vector of previous stage generator network h_{i-1} . Here, first the word level features are brought to generator working dimension ($\acute{e}$) and then combined with the previous hidden context. This is done by using dot product.

$$\acute{s}_{j,i} = h_j^T \acute{e}_i \tag{13}$$

From this step, an attention score is evaluated for j^{th} sub-region and i^{th} word. Finally, this score is used to evaluate word-context vector (c_j) for a particular sub-region.

$$c_j = \sum_{i=0}^{T-1} \beta_{j,i} \acute{e}_i, \; where \; \beta_{j,i} = \frac{\exp(\acute{s}_{j,i})}{\sum_{k=0}^{T-1} \exp(\acute{s}_{j,k})} \tag{14}$$

where, $\beta_{j,i}$ is an attention score for j^{th} sub region and i^{th} word normalized over all words.Here, the intuition is for one sub-region we are looking at all the words and we select a word which is most important for painting this particular sub-region. So, for every region we evaluate this word-context vector which is the final attention.

$$F^{attn}(e,h) = (c_0, c_1, .., c_{N-1}) \varepsilon \; \mathbb{R}^{\hat{D} \times N} \tag{15}$$

Generator The first generator network (F_0) generates the image using a latent noise vector drawn from normal distribution N(0,I) concatenated with output from $F^{ca}(\bar{e})$. The second (F_1) and third generator network (F_2) generates the images using output from $F_i^{attn}(e, h_{i-1})$ and previous hidden context. Hidden context for F_1 network is output from F_0 network i.e. $h_0 = F_0(z, \; F^{ca}(\bar{e}))$ and hidden context for following generator networks are $h_i = F_i(h_{i-1}, F_i^{attn}(e, h_{i-1})) \; for \; i = 1, 2$. The hidden context has dimension $h \; \varepsilon \; \mathbb{R}^{\hat{D} \times N}$, where $\hat{D}$ is generator's working dimension and N is context's available sub-regions. The generator networks F_0 , F_1 and F_2 are responsible for up-sampling the images respectively to 64 x 64, 128 x 128 and 256 x 256. The up-sampling is done using nearest neighbor interpolation with factor 2. The F_0 network consists of four up-sampling blocks so it brings the image size from 4 x 4 to 64 x 64 and remaining two, F_1 and F_2, consist of one up-sampling block each so they respectively bring the image size to 128 x 128 and 256 x 256. The generators G_0, G_1, and G_2 are responsible for bringing the output channel to 3 corresponding to RGB channel. Finally, the images generated from each generators are $\hat{x}_i = G_i(h_i) \; for \; i = 0, 1, 2$.

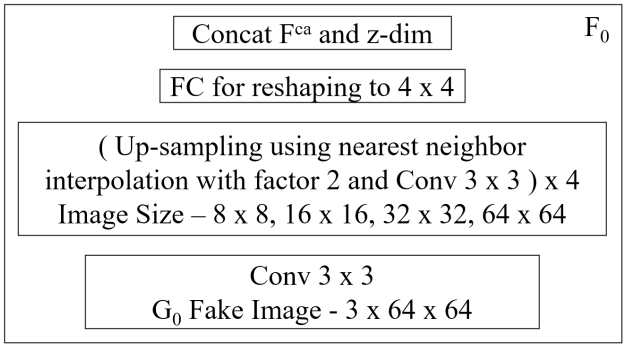


Figure 3: F_0 and G_0 generator networks.

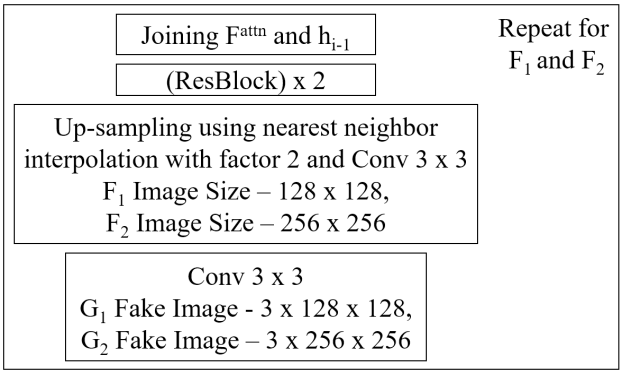


Figure 4: F_1, G_1, F_2 and G_2 generator networks.

Discriminator Three discriminators D_0, D_1, and D_2 for three corresponding generators G_0, G_1, and G_2 are used. Here D_0, D_1, and D_2 are responsible for classifying whether the provided image is real or fake. So, each discriminator network brings the image size down to 4 x 4 by using down sampling blocks which are implemented as convolution network with filter size 4 stride 2 and padding 1. Here input to discriminators are conditioning sentence embedding from text encoder, fake images from respective generators and real images resized to respective input image size for each discriminators. The discriminators D_0, D_1 and D_2 each takes input image of size 64 x 64, 128 x 128 and 256 x 256 respectively.

3.2 Training Objective Function and Losses

The training objective function comprises of losses from all the generators L_{G_i} and discriminators L_{D_i} along with the DAMSM loss L_{DAMSM}. The main objective function is -

$$L = L_G + \lambda L_{DAMSM}, \; where \; L_G = \sum_{i=0}^{m-1} L_{G_i} \tag{16}$$

where λ is a hyperparameter to control the overall effect of DAMSM loss.

The generator adversarial losses (L_{G_i}) for both unconditional and conditional losses is -

$$L_{G_i} = -\frac{1}{2}E_{\hat{x}_i \sim pG_i}[\log(D(\hat{x}_i))] - \frac{1}{2}E_{\hat{x}_i \sim pG_i}[\log(D(\hat{x}_i, \overline{e}))] \quad (17)$$

where $\hat{x}_i \sim pG_i$ means data is from generator. Here first part is unconditional loss which determines whether the image is real or fake and second part is conditional loss which determines whether the image and the sentence match or not.

The discriminator adversarial losses (L_{G_i}) for both unconditional and conditional losses is -

$$L_{D_i} = -\frac{1}{2}E_{x_i \sim pdata_i}[\log(D(x_i))] - \frac{1}{2}E_{\hat{x}_i \sim pG_i}[1 - \log(D(\hat{x}_i))] - \frac{1}{2}E_{x_i \sim pdata_i}[\log(D(x_i, \overline{e}))] - \frac{1}{2}E_{\hat{x}_i \sim pG_i}[1 - \log(D(\hat{x}_i, \overline{e}))] \quad (18)$$

where $\hat{x}_i \sim pG_i$ means data is from generator, $x_i \sim pdata_i$ means data belongs to true data distribution. Here first part is unconditional loss which classifies the image irrespective of the sentence input and second part is conditional loss which classifies the image with respect to conditioning sentence features.

4. Experimental Results and Discussion

4.1 Dataset

The dataset we used for training and testing the model is Multi-Modal-CelebA-HQ Dataset. The dataset contains 30,000 high-resolution face images each, paired with ten text descriptions. Out of 30,000 pairs, 18,943 are of female faces and, the remaining 11,057 pairs are of male faces. During training, we select one caption per image randomly. For this work, we created a dataset of 12,000 pairs. We randomly selected the pairs from the original dataset. Out of 12,000 samples, 6,000 are of male faces and 6,000 female faces. Now, we train the model on 9,600 text-image pair samples for 225 epochs, and we use the remaining 2,400 text-image pair samples for testing.

Although custom dataset of 12,000 is created, the vocabulary of words present in the dataset is same as that of original dataset. So, changing data in dataset may have same vocabulary of words in captions but result will definitely vary as the randomness during training is high because from provided ten captions per image, the selection of caption is random during training. And variation in the caption for a given image is also high. Also changing the image available to the model will change the real data distribution and as GANs are trained to approximate the real data distribution the generated images will also change. For example the word person may refer to male or female and hence as the captions are randomly selected for a given image it may some time refer to male in real image and sometime refer to female. So, the number of times the word person repeats for either male or female real image the model will approximate the image accordingly.

4.2 Model Training Details

We trained the model using Google Colab pro with GPU backend. We used PyTorch version 1.8.1 and python version 3.8.10 for implementing the model. The optimizer we used for training is AdamW. The parameters used in DAMSM model are $\gamma_1 = 4$, $\gamma_2 = 5$, $\gamma_3 = 10$ and $\lambda = 5$. Table 1 shows all other hyperparameters used for training.

Table 1: Training hyperparameters.

For Encoder Training	For GAN Training
Batch size = 32	Batch size = 32
Max epoch = 200	Max epoch = 225
Weight decay = 0.0002	Weight decay(D) = 0.0015
Learning rate = 0.00002	Weight decay(G) = 0.002
Embedding dim = 256	Learning rate(D) = 0.00015
Caption length = 26	Learning rate(G) = 0.0002
Z-dim = 100	Generator dim = 32
GRU input dim = 300	Discriminator dim = 64

Except batch size, max epoch, encoder learning rate, generator learning rate and weight decay all other hyperparameters are referenced from the paper [2]. We trained the model with the hyperparameters used in [2] but the training was unstable and model was colapsing. So we decreased the learning rate for encoder from 0.0002 to 0.00002 and discriminator learning rate from 0.0002 to 0.00015. Also we added a weight decay and as general rule of using weight decay ten times that of learning rate we set it to 0.0002 We set the batch size to 32 as using batch size of 64 increased the encoder training losses.

4.3 Results and Discussion

For analyzing the effect of changing text encoder in the GAN model, we trained two text encoders. One uses Bi-LSTM and, another uses Bi-GRU with BERT word embedding. So, this section discusses the sentence loss and word loss from the DAMSM model for these two encoders.

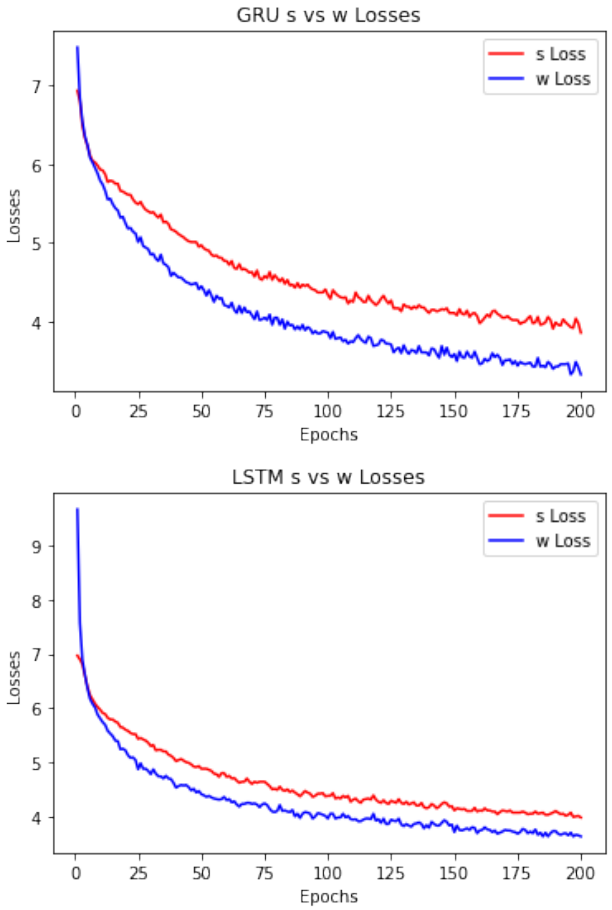


Figure 5: Sentence vs word losses for Bi-GRU and Bi-LSTM.

Figure 5 shows the plot of sentence loss ,vs word loss for both Bi-GRU and Bi-LSTM encoder. Here, the difference between sentence loss and word loss for Bi-GRU is more than that of Bi-LSTM. The difference for Bi-GRU to be more may be due to the breaking of words into subwords by the BERT tokenizer, which reduces the actual word count in the sentence. Also, the average number of words per sentence in the dataset is about 16. So, the sentences are to be padded, which also reduces the performance. In both plots, sentence loss is greater than word loss because word features provide more fine-grained information than sentence features. So, in addition to using sentence features for conditioning the discriminators, we used word features for conditioning the generators by adding an attention model between the generators.

Figure 6 shows the plot of Bi-GRU sentence-loss ,vs Bi-LSTM sentence-loss and Bi-GRU word loss ,vs Bi-LSTM word loss. The sentence loss in both cases is similar, but word loss is slightly less in the case of GRU as compared to Bi-LSTM. Also, we can see that, at starting of training, the word loss of Bi-LSTM is greater than that of Bi-GRU because Bi-GRU uses BERT word embedding, which is better than that used in Bi-LSTM.

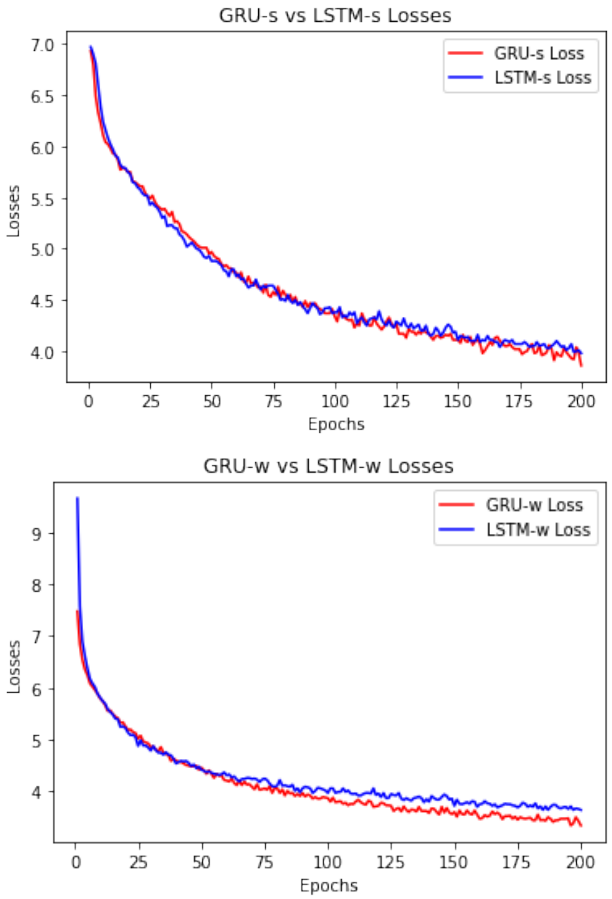


Figure 6: Sentence-loss and word-loss for Bi-GRU vs Bi-LSTM.

4.4 Output

All outputs are from the model after training it for 225 epochs. Figure 7 shows the output of the GAN model, with a Bi-GRU encoder on the top and, with a Bi-LSTM encoder on the bottom, for captions from test samples.

Figure 8 shows the images generated for custom short captions. Here, we can see that model using Bi-GRU can sense the word like eyeglass and eyeglasses are the same but, the model using Bi-LSTM cannot. Because the word eyeglass is not in the dataset vocabulary, its embedding is not available to Bi-LSTM. Hence Bi-LSTM cannot understand the word eyeglass. Bi-GRU uses BERT word embedding and, BERT has its own vocabulary. Because the word eyeglasses is not in the BERT vocabulary, the BERT tokenizer divides

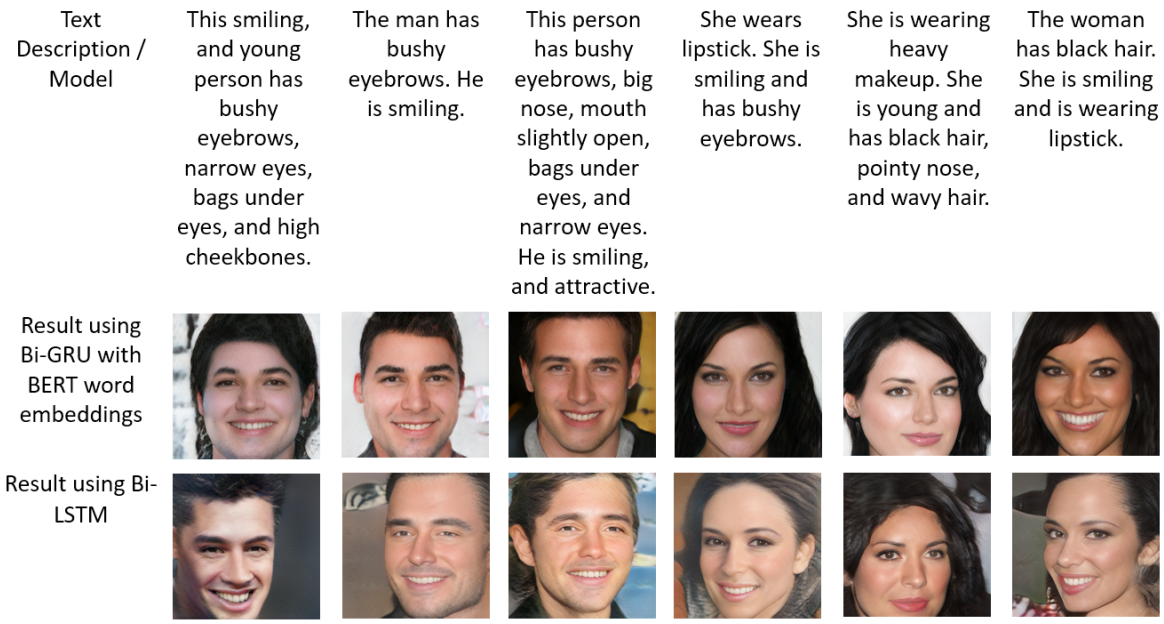


Figure 7: Output for captions from testing sample.

Figure 8: Output for short custom captions.

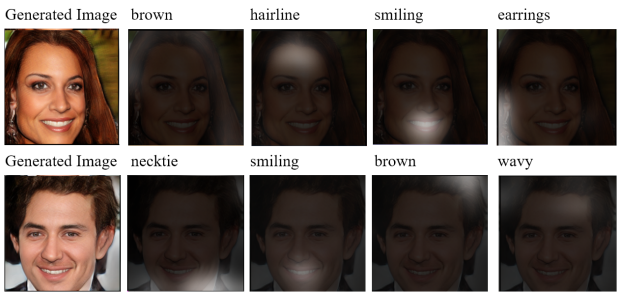


Figure 9: Word attention to image regions during training.

the word eyeglasses into subwords eye, ##glass, and ##es. So, Bi-GRU receives a sum of embedding for the subwords instead of eyeglasses. Hence, Bi-GRU has a sense that the words eyeglasses and eyeglass are the same. Also, we can see that model can generate different images for the same caption, which proves that the GAN model is not collapsing. Figure 9 shows word attention to image regions depicted in the image with white spots.

Evaluation Metric We used Fréchet inception distance (FID) [11] as metric for accessing the quality of generated image compared to the real image. We chose FID as it is considered a standard metric for accessing the quality of images generated by GAN model. The Table 2 shows FID scores for the aforementioned two models -

Table 2: FID score for the two models discussed.

Model	FID Score
Bi-GRU with BERT word embedding	61.19
Bi-LSTM	64.24

For evaluation purpose we generated images for 2400 captions from test sample. In Table 2 we can see that the model Bi-GRU with BERT word embedding have lower FID score than that of Bi-LSTM. As less FID score is considered better, the images generated from Bi-GRU with BERT word embedding is better than that of Bi-LSTM.

5. Conclusion

In this work, we tried to use AttnGAN [2] for synthesizing human face images from the textual description of facial features. Here we proposed to change the text encoder from Bi-LSTM to Bi-GRU with BERT word embedding to get better results from the same GAN model. In Figure 7 and Figure 8, we can see that the model using Bi-GRU with BERT word embedding as text encoder can generate better images as well as handles the same word with different forms better than those using Bi-LSTM. So, using a BERT word embedding with a text encoder in the GAN model, we can generate images more semantically aligned with the text description.

References

[1] Osaid Rehman Nasir, Shailesh Kumar Jha, Manraj Singh Grover, Yi Yu, Ajit Kumar, and Rajiv Ratn Shah. Text2facegan: Face generation from fine grained textual descriptions. *arXiv e-prints*, pages arXiv–1911, 2019.

[2] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1316–1324, 2018.

[3] Weihao Xia, Yujiu Yang, Jing-Hao Xue, and Baoyuan Wu. Tedigan: Text-guided diverse face image generation and manipulation. *arXiv preprint arXiv:2012.03308*, 2020.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[5] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[6] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[7] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *International Conference on Machine Learning*, pages 1060–1069. PMLR, 2016.

[8] Scott E Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, and Honglak Lee. Learning what and where to draw. *Advances in neural information processing systems*, 29:217–225, 2016.

[9] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5907–5915, 2017.

[10] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1947–1962, 2018.

[11] Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. `https://github.com/mseitzer/pytorch-fid`, August 2020. Version 0.2.1.