ENHANCING TEXT EMBEDDING FOR GAN-BASED TEXT-TO-IMAGE TRANSLATION

Abhishek Bhatt, Tianzhi Cao
Rutgers University
{ab2083,tc796}@scarletmail.rutgers.edu

ABSTRACT

Text-to-Image (T2I) translation aims to generate a semantically consistent and visually realistic image conditioned on a textual description. Over the recent years, advancement in GAN architectures has showed very promising results for generative image problems in general, and T2I in particular. In this work, we focus on the task of learning text representations for T2I problem, such that they effectively capture the visual characters of the corresponding images. We try to jointly learn these embeddings with the GAN model parameters, instead of pre-training the encoder as proposed in most state-of-the-art models. We expect that this experiment will provide insights into learning representations that improve the generated image qualities from existing SoTA GAN-based T2I models.

1 INTRODUCTION

Conventionally, images have been captured using a camera. Synthetic images on the other hand, are images computed based upon some other type of signal. The humongous amounts of easily accessible text data on the web and other digital forms, is thus a natural motivation to synthesize images from text or other human language descriptions. Such systems are of great interest for applications like data augmentation, photo-editing, computer-aided designing, etc.
The problem combines Natural Language Processing and Computer Vision, in the sense we need to learn text representations that capture visual characteristics and then spatially up-sample them to generate images. This generative task can be achieved as a supervised learning problem using Generative Adversarial Networks. By conditioning the Generator on text, we can train it with the help of a discriminator, to generate images from random input noise. All influential work to solve T2I leveraging this idea rely on recent advances in sequential architectures (like Recurrent Neural Nets) and deep Convolutional Neural Nets, in order to encode text and images respectively.
There are two major challenges that need to be addressed for generating plausible photo-realistic images from text descriptions.

- Text embedding : The context of the word as represented by Word2Vec does not capture the visual properties very well. For T2I, we need to learn a joint text-image encoding such that the text encoder learns embeddings that are not too different from image encodings.
- Multimodal Learning : For T2I it is useful to capture correspondences between different modalities of the data. For example, multiple captions may describe a single image, or multiple images may correspond to a single caption.

In the following sections, we present our approach, experimentation and observations on improving text embedding for T2I.

2 RELATED WORK

GANs were introduced by Goodfellow et al. (2014). Reed et al. (2016b) combined advances with deep RNN text embeddings with image synthesis using GANs to propose DC-GAN for T2I. The work by Reed et al. (2016a) further expanded upon the T2I capability to both generate images from text, and use bounding boxes and key points as hints as to where to draw a described object. The problem of low resolution (64x64) synthesized images with not much coherent generated

scenes with DC-GANs, was resolved by Zhang et al. (2017). Using two-stage GANs, this work first sketches basic shapes and colors, and then corrects defects in the first stage to yield high resolution (256x256) images. Besides, a Conditioning Augmentation technique is used for stabilized training and diversity of generated samples. AttnGAN by Xu et al. (2017) focuses on generation of high quality images with complex scenes as in the MS-COCO dataset. Using generated image sub-regions to query word-context from global sentence vector via attention, as well as defining a fine-grained image-text matching loss, this architecture generates higher resolution (256×256) images as well as improve upon the previous best inception scores for evaluating synthesized images. Finally, the TIME model by Liu et al. (2020) uses GANs to jointly learn a T2I generator and an image captioning discriminator, rather than approaching T2I T2I as a uni-directional task and using pre-trained text embeddings. TIME uses transformer encoder, that has also been used in our text encoding implementation. Self-attention transformer for sequence-to-sequence modeling was proposed in the landmark work by Vaswani et al. (2017), that can possibly provide a unifying framework for deep learning architectures. Our image encoding implementation uses the architecture proposed by He et al. (2015) for training very deep CNNs.

3 APPROACH

3.1 DATASET

Figure 1: An image in Oxford-102 flowers dataset

We trained our model against the Oxford-102 flowers (Nilsback & Zisserman (2008)) dataset. The dataset consists of 102 flower categories, with each class consisting of 40 to 258 flower images (see Figure 1). For our implementation, we used the images that can be accessed here : `https://drive.google.com/drive/folders/1uORr7J-8jWaovhcH7IhOzFIb2liV2w7j?usp=sharing`. For each image, we have text descriptions that can be accessed here : `https://drive.google.com/drive/folders/18H5iIRidsH7FHuz0VBI3toSWQ8M4caIt?usp=sharing`. Each text file contains 10 descriptions for one image. In our data preparation step, we randomly pick one of the descriptions corresponding to each of the images. We split the dataset of 8190 image-caption pairs as 80%-10%-10% train-dev-test sets. Each split is divided into mini-batches of size 128. Due to limited RAM/GPU memory on Google Colaboratory, we resized the RGB training images to 64 x 64 spatial size. From the 8190 text captions, we build a vocabulary including all word tokens and two special tokens PAD (for ensuring same sentence lengths while training) and UNK (for any unknown tokens in the vocabulary).

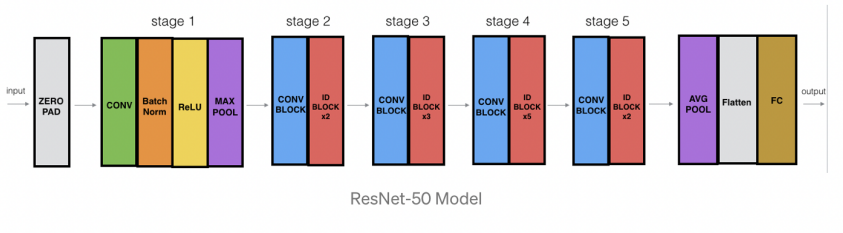


Figure 2: ResNet-50

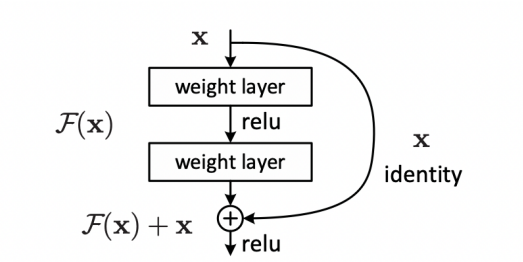


Figure 3: A residual block

3.2 Proposed Model

- Image Encoder : We implemented the ResNet-50 (see Figure 2) convolutional model to compute the encodings of images in the training set. While training our model, these encodings are used to compute the mean squared error loss against the text embeddings. Using residual or skip connections (see Figure 3) across blocks of convolution and pooling layers, this architecture allows training a very deep CNN for learning efficient image representations. All parameters of the image encoder are jointly learnt with the GAN.

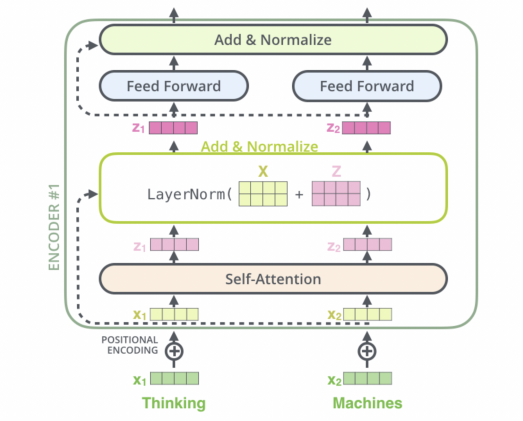


Figure 4: A transformer block

- Text Encoder : We implemented a Transformer text encoder that uses multiple transformer blocks (see Figure 5) with multihead attention in each block. The padding token positions in the text are masked out while computing the dot-product self attention. Unlike RNN models, transformer (see Figure 4) does not enforce sequence ordering. So we compute positional encodings from a sinusoidal function and add them to text embeddings, before passing the embeddings as input to the stack of transformer blocks. All parameters of the text encoder are jointly learnt with the GAN. Transformer or self-attention models

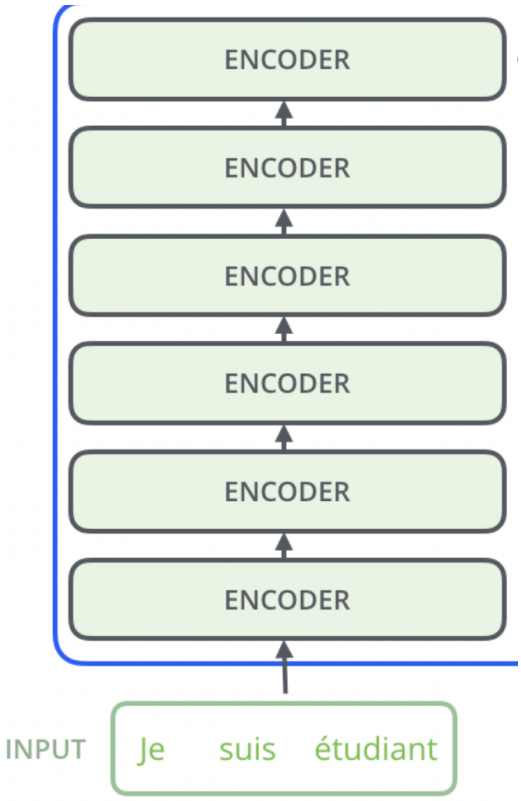


Figure 5: Text encoder

are widely used for SoTA performance across various NLP tasks. So we expect that our text encoder implementation will learn better embeddings for the image descriptions as compared to RNN or LSTM encoders used in previous T2I models.

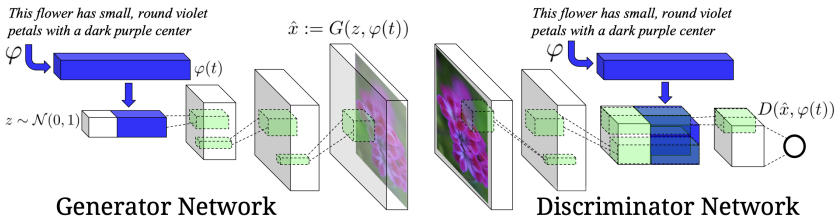


Figure 6: DC-GAN

- Generator : We use a vanilla DC-GAN (see Figure 6) generator with our text encoder implementation. The generator uses a sequence of transposed convolutions with batch normalization and ReLU / tanh non-linearities to synthesize 64x64 images from an input vector by spatial upsampling. The input vector is a noise vector sampled from a standard normal distribution, and then appended to the output of our text encoder.
- Discriminator : The discriminator is a convolutional image classifier that takes as input a batch of images. After a series of convolutions with batch normalization and LeakyReLU activation, the output sigmoid layer predicts labels for each input image. An image from the dataset is labeled as real (1), while a synthesize image is labeled as fake (0) while training.

- Loss -
  Discriminator : For the discriminator we compute the Binary Cross Entropy loss between the predicted labels and the ground truth labels (1 for images in the dataset, 0 for synthesized images). This loss formulation comes from Kullback-Leibler Divergence that measures how one probability distribution (predicted labels) is different from a second, reference probability distribution (ground truth).

$$\mathcal{L}_D = -\frac{1}{N}\sum_{i=1}^{N} \log D(I_i) + \log(1 - D(G_i))$$

  Generator : The generator loss is the Binary Cross Entropy loss between predicted labels for synthesized images and ground truth labels for real images (1). By minimizing this loss, we encourage the generator to synthesize images that can pass through the discriminator as real.

$$\mathcal{L}_{Generative} = -\frac{1}{N}\sum_{i=1}^{N} \log D(G_i)$$

  For our enhancement hypothesis, we also compute an embedding loss as the Mean Squared Error between the input image encodings (from image encoder) and the corresponding text embeddings (from text encoder). This loss is added to the generative loss to get the total generator loss. We expect that minimizing this part of the total generator loss will encourage the text encoder to learn a representation that is not too different from the image described in the text. For inference, we use the text encoder trained this way.

$$\mathcal{L}_{Embedding} = \frac{1}{N}\sum_{i=1}^{N}(imageEnc(I_i) - textEnc(T_i))^2$$

$$\mathcal{L}_G = \mathcal{L}_{Generative} + \mathcal{L}_{Embedding}$$

In the above loss equations, G_i is the i^{th} synthesized image by generator, D is the label predicted by discriminator, I_i is the i^{th} image in the training set, T_i is the i^{th} text caption in the training set, and N is the number of training examples.
Our training objective in terms of model parameters θ is thus :

$$\theta_D^*, \theta_G^* = \arg\min_{\theta_D, \theta_G} \mathcal{L}_D(\theta_D) + \mathcal{L}_G(\theta_G)$$

The gradients $\frac{\partial \mathcal{L}_D}{\partial \theta_D}$ and $\frac{\partial \mathcal{L}_G}{\partial \theta_G}$ are back-propagated during training to update the parameters of the discriminator θ_D and the generator(including input encoder) θ_G respectively.

The training setup is as follows :
Optimizer : Adam
Size of latent or random noise vector : 512
Number of training epochs : 40
Learning rate for discriminator and generator optimizers : 0.0002
Learning rate for input encoder optimizer : 0.0006
Beta1 hyperparam for Adam optimizers : 0.5
Number of transformer blocks : 9
Word embedding length : 512
Number of heads in multi-head self attention : 8
Attention padding mask fill value : -1e10
Dropout rate : dropout = 0.1

The implementation is done using the PyTorch deep learning library, on Google Co-laboratory. Our code can be accessed here : `https://github.com/a1code/Image-Synthesis-from-Text-with-DCGAN/blob/main/code.ipynb`.

4 RESULTS

Our model generates 64 x 64 images. We used the trained encoder and generator to generate images for the text descriptions in the dev set. The trained models can be accessed here : https://drive.google.com/drive/folders/1HD-aTKy2Ll_qjXA5hdY9YaH1Gm3T7Kjj?usp=sharing. Based on analyzing the generated images on the dev set we did hyperparameter tuning for our model. The results on dev set for the last round of model tuning can be accessed here : https://drive.google.com/drive/folders/1wk-dBL39o2_OJWK2kqs82zU0keGJadwT?usp=sharing. Finally, we run the generation on our held-out test set. The results can be accessed here : https://drive.google.com/drive/folders/1oHnMIbz7cTV8eXt44XZtcuU7cGsQJDZb?usp=sharing.
Due to time constraints we did not quantitatively measure the quality of the generated images. Instead, we relied on human evaluation to check whether generated images are well conditioned on given text descriptions.
Some generated samples by our model on the dev set are presented in Figure 7. Some generated samples on the test set are presented in Figure 8.

4.1 OBSERVATIONS AND DISCUSSION

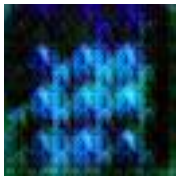
From the results, we observe the following.

- Most of the generated samples do not look like flowers, especially cases where text descriptions are more generic than specific. These samples simply contain colored patches or textures.
- For some samples, the model does seem to capture the color and structural information. The samples look like a flower-shaped object with colors corresponding to those mentioned in the description. Usually, these are more specific text descriptions.
- The generated images are low resolution, and the scenes are not coherent.

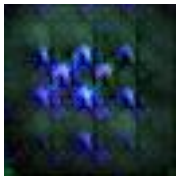
Overall, based on human evaluation, we can conclude that the model generates few plausible flower images. However, such images do not look photo-realistic. Also, the generated samples do not seem to capture all the visual information in the text descriptions.
Some possible reasons for our implementation not obtaining expected results are summarized below. This can be verified with experiments as a future extension of this project.

- Most SoTA word embedding models are computationally expensive and trained on billions of text documents. Our text encoder on the other hand gets less than 8000 training examples to learn text encoding, and improve them guided by image encodings. We expect that this resulted in a huge performance drop.
- Due to memory constraints on Google Colab GPU, we had to resize training images to 64x64 from the original 512 x 512 information. We believe we are losing out a lot of crucial information by decreasing the resolution that could be useful for learning the image encoder. This could have consequently improved the text encoder with the MSE loss formulation.
- We used a very simple DCGAN setup for our experimentation. SoTA T2I models incorporate multi-stage GANs with more complex architectures within the generator. Our understanding is that our formulation should give better results with the generated samples are refined with further stages of GANs.
- Finally, we have formulated a simple MSE loss between text and image encodings. MSE simply penalizes for the global text embedding being different from the corresponding image encoding. There is no way to compare local correspondences between text descriptions and the images. We believe that this on average leaves out some crucial visual information in the text embeddings which would be useful to coherently draw generated images.

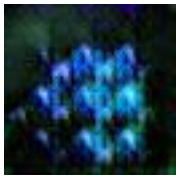
Figure 7: Generated samples from dev set. (a) a blue bell shaped flower with green sepal and a white tipped pollen tube. (b) this flower has a five pointed star configuration of rounded petals that are either blue or light pink in color. (c) this flower has a large blue petal with a white anther in the center. (d) this flower has large white petals that have purple specks scattered towards the tips. (e) this flower has petals that are pink and is folded together. (f) this flower has petals that are white and has flowery stigma. (g) this flower has petals that are white with purple patches. (h) this flower has petals that are yellow and very ruffled together. (i) this flower has pink leaves purple petals and a light green pedicel. (j) this flower has red petals a green ovule and white anther filaments. (k) this flower has rounded orange petals with the color graduating to yellow and lighter orange inside. (l) this flower has rows of red petals and long yellow stamen



(a)



(b)



(c)

(d)

(e)

(f)

(g)

(h)

(i)

(j)

(k)

(l)

Figure 8: Generated samples from test set. (a) a couple of small but large golden pedaled flowers with a dull white center. (b) a flower with groups of tubular yellow purple and white petals. (c) a flower with short and wide petals that are a burnt orange. (d) a flower with wide petals that are white and purple. (e) a large velvet flower with a bell shaped attached to a flat base. (f) a yellow and white flower with bell shaped petals and a brown pediciel. (g) all parts of the flower are yellow including the ovary the long thin petals and the pistil pedicel is not visible. (h) sharp pink petals are staggered around a circular region of bright yellow stamens. (i) the flower petals are needle shped and are purple in color. (j) the greenish white flower has petal that is fused at sepal and suddenly flaring out to form a star like shape. (k) the petals of the flower are pink in color and are arranged in numbers of five. (l) the yellow anthers are around and close to the petals.

(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l)

5 CONCLUSION

In this work, we demonstrated an approach to learning better text embeddings for text-to-image translation using GANs. Though our current implementation results do not reflect the effectiveness of our proposed enhancement, we believe that the approach can yield better results with more experimentation and training with much larger datasets and complex architectures. This may indeed require more computational resources like a multi-GPU setup to efficiently process training workloads. Nevertheless, this whole exercise gave us a good insight into the T2I problem, SoTA T2I models and the challenges involved in effectively solving this problem through deep learning. We also learnt how to do academic literature survey and approach implementing proposed architectures, which was a good introduction to research for us. The project has given us a good starting point to dig deeper into this area of active research and interest.

REFERENCES

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

Bingchen Liu, Kunpeng Song, Yizhe Zhu, Gerard de Melo, and Ahmed Elgammal. Time: Text and image mutual-translation adversarial networks, 2020.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

Scott Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, and Honglak Lee. Learning what and where to draw, 2016a.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis, 2016b.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

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, 2017.

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks, 2017.