Text to Image Synthesis using Neural Networks

Swapnil Taneja (PID: A53219137)
Department of CSE
University of California, San Diego
San Diego, CA 92093
swtaneja@eng.ucsd.edu

Saksham Sharma (PID: A53220021)
Department of CSE
University of California, San Diego
San Diego, CA 92093
sas111@eng.ucsd.edu

Kriti Aggarwal (PID: A53214465)
Department of CSE
University of California, San Diego
San Diego, CA 92093
kriti@eng.ucsd.edu

Prahal Arora (PID: A53219500)
Department of CSE
University of California, San Diego
San Diego, CA 92093
prarora@eng.ucsd.edu

Saicharan Duppati (PID: A53221873)
Department of CSE
University of California, San Diego
San Diego, CA 92093
sduppati@eng.ucsd.edu

Abstract

Synthesizing photo-realistic images for text-description is a challenging problem in computer vision and has many practical applications. Image samples generated by the model can reflect the semantic relationships in the text description. In this paper we adhered to this task . This can be done using Deep Convolutional Generative Adversarial Network. In our approach, we chose to explore the model proposed by Reed et. al 2014 [1]. Generating handwritten digits is a tremendous challenge. Any scribbled digit should look real and hence Generative Adversarial Network suits the purpose. The way a child learns semantic representation of textual relationships, our model learned to write single digits and was able to generate images up to a million using just 2000 training images.

1 Introduction

The goal of the project is to translate text description to generate photo-realistic images and test its portability to other datasets. While the existing work has been done on flowers dataset, etc., we chose to start with testing the existing network on MNIST dataset that is widely used as a benchmark and decided to run inference on various settings as proposed in section 6 (Experiments and Results). The main goal of the project is to run experiments and explore the network architecture that was described in the paper by Scott Reed et al [1] and to analyze how it works by using a simpler dataset. Through the experiments, we found that the model generalizes to other datasets and hence provides a unified methodology for performing this task.

Our motivation for this project comes from numerous possible applications that involve generating images from a different modality. Also while there has been a lot of work done in captioning the images, it's only recent that generation of images from textual description has been explored. Therefore, we think that this is an interesting problem to work on.

2 Related Work

Scott Reed et al in their paper[1], described an approach to do an automatic synthesis of images from text using GAN (Generative Adversarial Networks). Figure 1 is one of the results from this paper. We start off with adopting this model described in this paper.

Facebook AI Research developed a DCGAN (Deep Convolutional Generative Adversarial Networks) in tensorflow for generating Human Faces which do not resemble any human being. The results can be seen here at their Neural Faces demo web page [10].

Han Zhang et al in their paper [1] have used a stack of (two) GANs in two stages that have resulted in a higher resolution of the generated images that are more realistic than the ones from the previous approach. Also, images in many cases are detailed with vivid object parts, e.g., beaks and eyes of birds.

Our aim for this project is to use the trained model described in the first approach (Scott Reed) and run several experiments on it, as described in the section 6 of this report.

Goodfellow et al.[5] in their paper explained the concept of Generative Adversarial Networks, which we used as motivation for the previous two approaches.

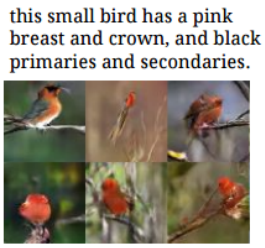

Figure 1: Result from Scott Reed et al.

3 Background

3.1 Skip Thought Vectors

In order to understand the semantic relationships in the text input the approach listed in the paper by Scott Reed et al [1], is to use Skip Thought Vectors for modeling these dependencies. Ryan Kiros. et al [11], in their paper developed a sentence modeling structure using the concept of word2vec [12], by applying the same concept to sentences. That is, instead of using a word to predict its surrounding context, they encode a sentence to predict the sentences around it. It can be seen in the figure 2 that skip thought vectors present a distributed representation of sentences that model semantic relationships. In addition to that, skip thought vectors were trained on BookCorpus Dataset which ensured that these vectors generalize to many different tasks and is not restricted to one particular task. In order to incorporate various other relationships that are not modeled in the BookCorpus dataset, they provide an approach of linear mapping from one vector space to another, supporting its portability.

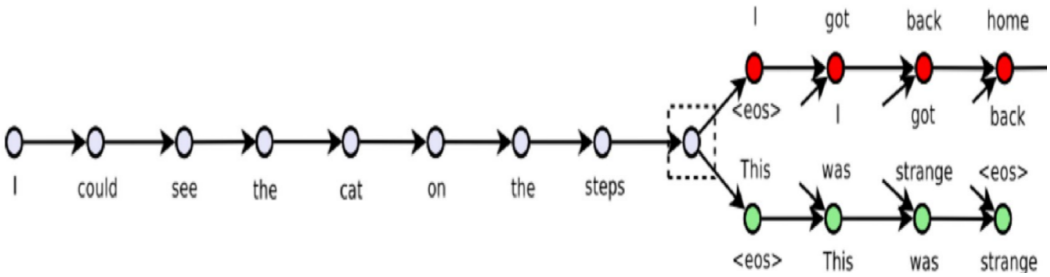

Figure 2: The skip-thoughts model. Given a tuple $(s_(i-1), s_i, s_(i+1))$ of contiguous sentences, with s_i the i-th sentence of a book, the sentence s_i is encoded and tries to reconstruct the previous sentence $s_(i-1)$ and next sentence $s_(i+1)$. In this example, the input is the sentence triplet *I got back home. I could see the cat on the steps. This was strange.* Unattached arrows are connected to the encoder output. Colors indicate which components share parameters. $< eos >$ is the end of sentence token.

3.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) consist of a generator *G* and a discriminator *D* that compete in a two-player minimax game. [5]. The discriminator tries to distinguish and increase the probability of the real image and at the same time decrease the probability of the fake image. The generator network is optimized to fool the adversarially trained discriminator into predicting that the synthetic images are real. By conditioning both generator and discriminator, we can model the text to image synthesis since the discriminator network acts as a "smart" adaptive loss function. Concretely, *D* and *G* play the following game on V(D,G):

$$min_G max_D V(D,G) = E_{x \sim p_data(x)}[log D(x)] + E_{x \sim p_z(z)}[log(1 - D(G(z)))] \quad (1)$$

Goodfellow et al. [5] prove that this minimax game has a global optimum precisely when $p_g = p_{data}$, and that under mild conditions p_g converges to p_{data}. In practice, in the start of training samples from D are extremely poor and rejected by D with high confidence. It has been found to work better in practice for the generator to maximize $log(D(G(z)))$ instead of minimizing $log(1 - D(G(z)))$.

4 Model and Network Architecture:

The architecture is adopted from Reed et al.(2016) [1]. The network architecture is illustrated in Figure 3.

Description:

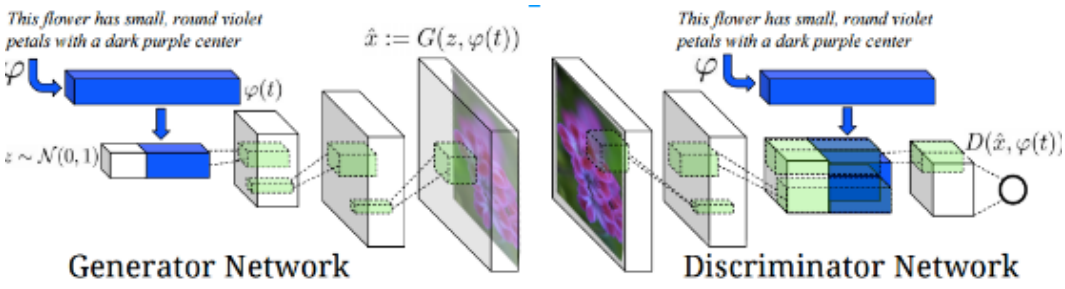

Figure 3: Network Architecture

The generator network is denoted as G : $\mathbb{R}^T \times \mathbb{R}^Z \rightarrow \mathbb{R}^D$, and the discriminator as D : $\mathbb{R}^D \times \mathbb{R}^T \rightarrow \{0, 1\}$, where T is the dimension of the text description embedding, D is the dimension of the image and Z is the dimension of the noise vector.

In the generator G, we encode the text query *t* using text encoder ϕ. The description embedding $\phi(t)$ is first compressed using a fully-connected layer to a small dimension followed by leaky-ReLU. Following this, inference proceeds as in a normal deconvolutional network: we feed-forward it through the generator G; a synthetic image x is generated via $x \leftarrow$ G(z, $\phi(t)$). In short, image generation corresponds to feed-forward inference in the generator G conditioned on query text and a noise sample.

In the discriminator D, we aim to perform several layers of stride-2 convolution with spatial batch normalization (Ioffe & Szegedy, 2015) followed by leaky ReLU. We again reduce the dimensionality of the description embedding $\phi(t)$ in a (separate) fully-connected layer followed by rectification. When the spatial dimension of the discriminator is 4 $\times$ 4, we replicate the description embedding spatially and perform a depth concatenation. We then perform a 1 $\times$ 1 convolution followed by rectification and a 4 $\times$ 4 convolution to compute the final score from D.

Algorithm (GAN-CLS) [From Reed et al (2016)]

Input: minibatch images *x*, matching text *t*, mismatching $\hat{t}$, number of training batch steps S
for n = 1 to S do
. $h \leftarrow \phi(t)$ {Encode matching text descriptions}
. $\hat{h} \leftarrow \phi(\hat{t})$ {Encode mis-matching text descriptions}
. $z \leftarrow N(0, 1)^Z$ {Draw sample of random noise}
. $\hat{x} \leftarrow G(z, h)$ {Forward through generator}
. $s_r \leftarrow D(x, h)$ {real image, right text}
. $s_w \leftarrow D(x, \hat{h})$ {real image, wrong text}
. $s_f \leftarrow D(\hat{,} h)$ {fake image, right text}
. $L_D \leftarrow log(s_r) + (log(1 - s_w) + log(1 - s_f))/2$
. $D \leftarrow D - \alpha \partial L_D / \partial D$ {Update discriminator}
. $L_G \leftarrow log(s_f)$
. $G \leftarrow G - \alpha \partial L_G / \partial G$ {Update generator}
end for

5 Dataset:

The pre-trained model is created using the Oxford-102 Flowers dataset, which has 82 train+val and 20 test classes. In addition to this, we came across the following relevant datasets that we might use:

- Oxford-102 Flowers dataset:
  It contains images of flowers along with 5 human-generated captions per image. This dataset has 8,189 images of flowers from 102 different categories.
- MNIST:
  It is a general set of images that display digits. They are labeled according to the digit number.
- Hand crafted dataset:
  We created our own dataset by concatenating 3 digit images from MNIST randomly and associated it with the label representing the whole number. For example, Three Hundred and Twenty-three is the label description for concatenating image of Three, two and three. We extended this dataset to add upto 7 digit numbers for checking extensibility. We tested on different settings of the data configurations as mentioned in the experimental results section.

6 Experiments and Results

In order to evaluate pre-trained models, we started experimenting with the code from the GitHub repository [8] with Reed et al architecture[1]. We ran inference on flower captions using the pre-

trained model on flowers dataset and we tested it by generating a few of the images corresponding to different caption texts. We replicated the image generation of the same caption texts mentioned on the GitHub project in addition to caption texts of our own. The images with their caption texts are shown in figures 9, 10, 4, 5, 6, 7 and 8. The images referred in figure 9, and 10 do not give us the required result. We believe it is because the model is not trained on black flowers and the text "The flower has white petals and the center of it is red" seems confusing to the model. So it generates tinge of white petals and mostly red.

Figure 4: The flower has yellow petals and the center of it is brown

Figure 5: The flower shown has yellow anther red pistil and bright red petals

Figure 6: The petals on this flower are white with a yellow center

Figure 7: This flower has petals that are yellow white and purple and has dark lines

Figure 8: This flower is orange in color and has petals that are ruffled and rounded

Figure 9: Black flowers with yellow stigma

Figure 10: The flower has white petals and the center of it is red

Now, in order to test its portability, we describe the following settings on which we evaluated our model.

Setting 1: Single Digits from MNIST

The first setting is when we chose to keep single digit captions as input. We trained our model on the data having single digits (MNIST original dataset). We were able to generate these images gracefully with a few exceptions. The results are given by figure 11

Figure 11: Setting 1: The images generated are corresponding to single digit as input ordered sequentially from 0 to 9

Setting 2: Multi-digit data generated from MNIST (With single digit labels)

In the second setting, we created three digit images from MNIST data by concatenating individual digits. For this setting, the labels for the three digit images is just a concatenation of the labels of the individual digits. For example, for the image that corresponds to '212', the label corresponding is 'Two One Two'. We generated the model and tested it on the three digit captions created this way. To our surprise, we were able to generate relatively good results. Please refer to the figure - 12

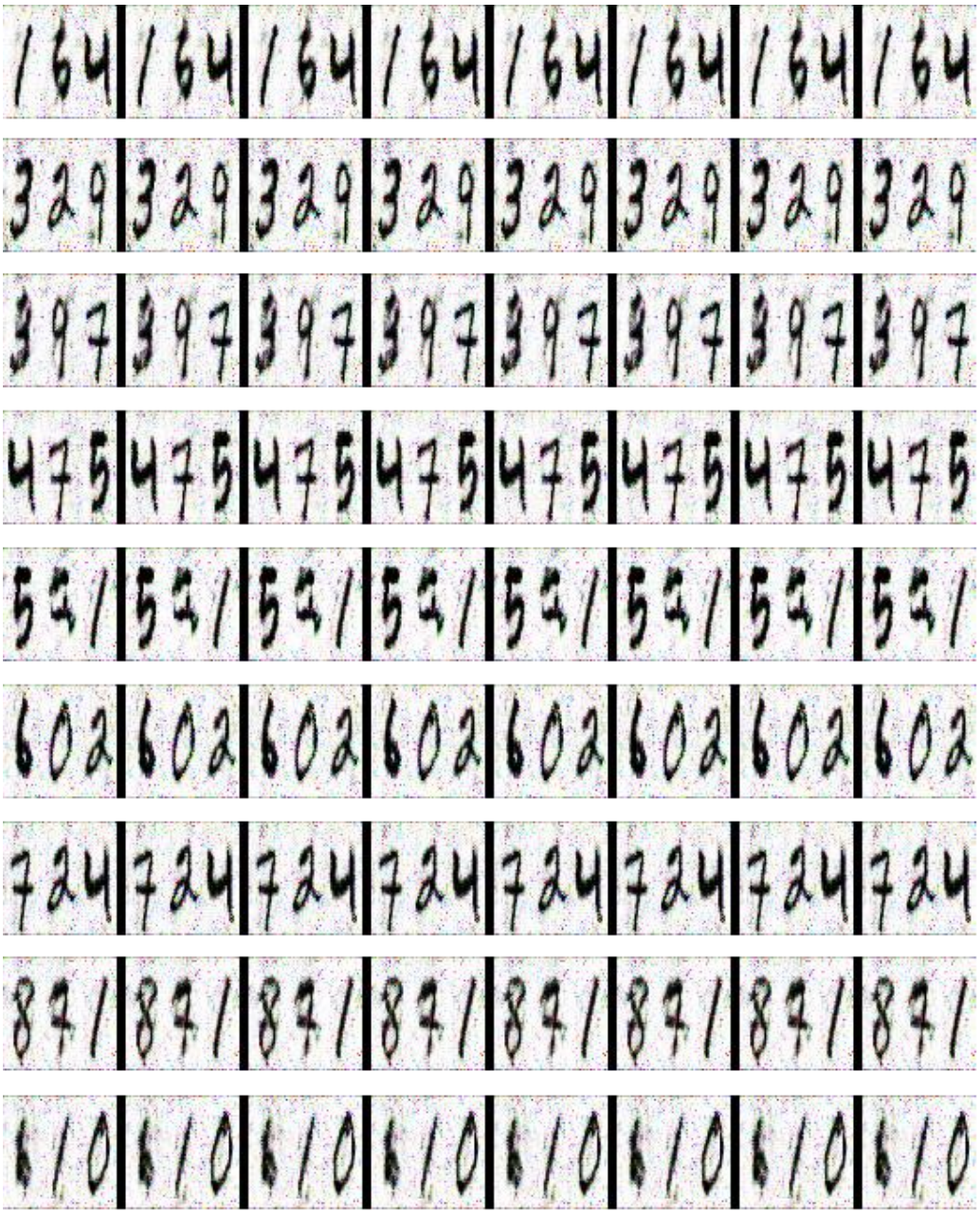

Figure 12: Setting 2: The images generated are corresponding to the captions listed here in their respective order – One Six Four (164), Two Three Nine (239), Three Nine Seven (397), Four Seven Five (475), Five Eight One (581), Six Zero Two (602), Seven Two Four (724), Eight Seven One (871), Nine One Zero (910)

Setting 3: Multi-digit data generated from MNIST (With multi-digit labels)

In this setting, we keep the concatenated images from the previous section but we change the labels of the images to notation involving hundreds and tens rather than concatenating labels of the individual digits. For example, the image corresponding to '212' is given a label 'Two Hundred Twelve'. We now present the images that describe our results. Please refer to the figure – 13

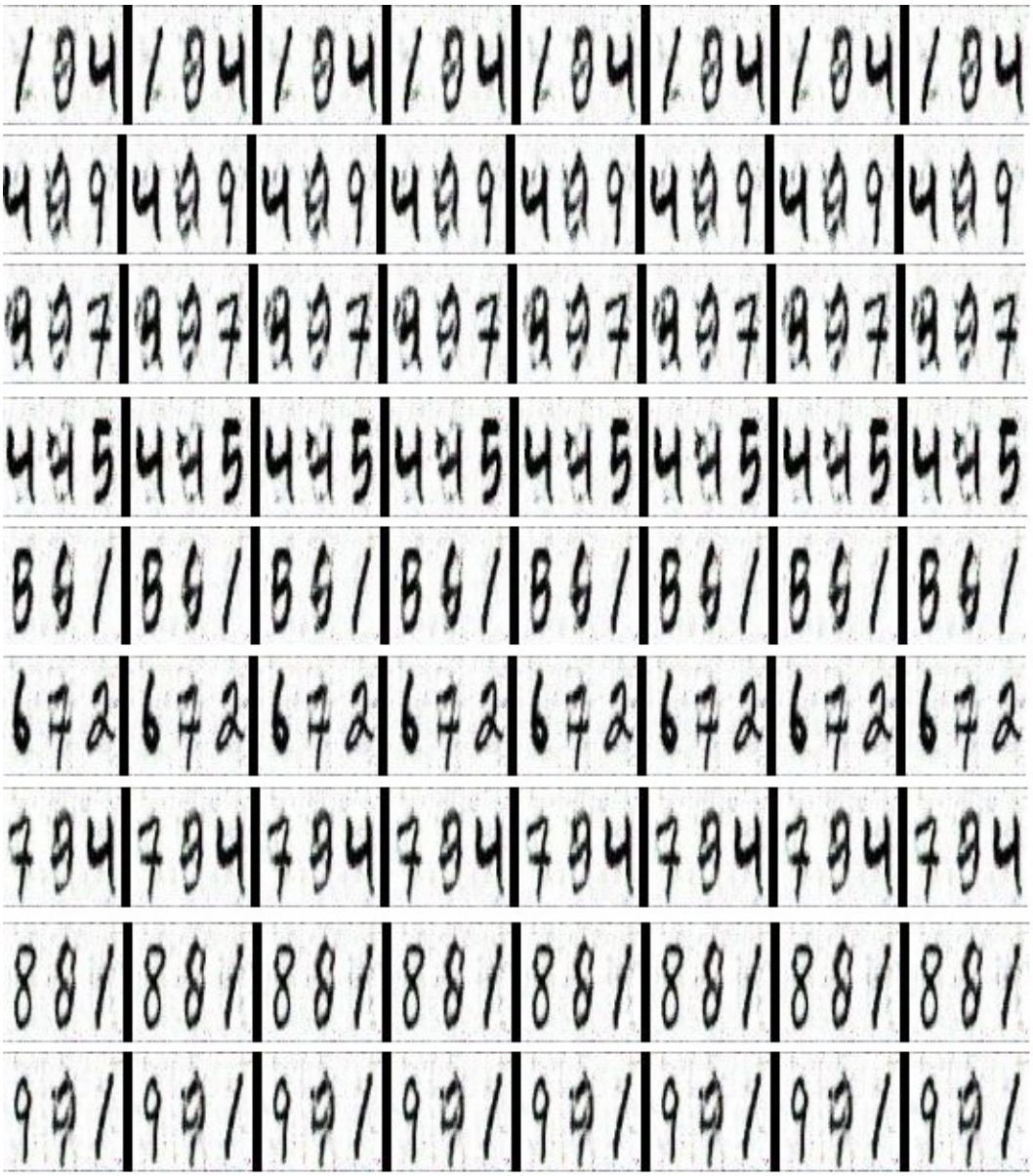

Figure 13: Setting 3: The images generated are corresponding to the captions listed here in their respective order – One Hundred Sixty Four (164), Two Hundred Thirty Nine (239), Three Hundred Ninety Seven (397), Four Hundred Seventy Five (475), Five Hundred Eighty One (581), Six Hundred Two (602), Seven Hundred Twenty Four (724), Eight Hundred Seventy One (871), Nine Hundred Ten (910)

Setting 4: Generation of numbers of variable length (1 to 3 digits)

For this setting, we trained our model on images of numbers consisting of 1, 2 and 3 digits. The captions for the digit images were created by the concatenation of the captions of the individual digits. For example, for the image that corresponds to '212', the caption corresponding is 'Two One Two'. The dataset had 1050 images (150 single digits, 300 double digits, 600 three digit numbers). After training the model, we tested the model of variable length numbers. We found that the model was able to generate one and two digit numbers without any mistake but it could not generalize well for 3 digits. We think the major reason for that is the small dataset, if we increase the dataset for the 3 digits the model should perform well even for the three digit numbers. Please refer to the figure 14

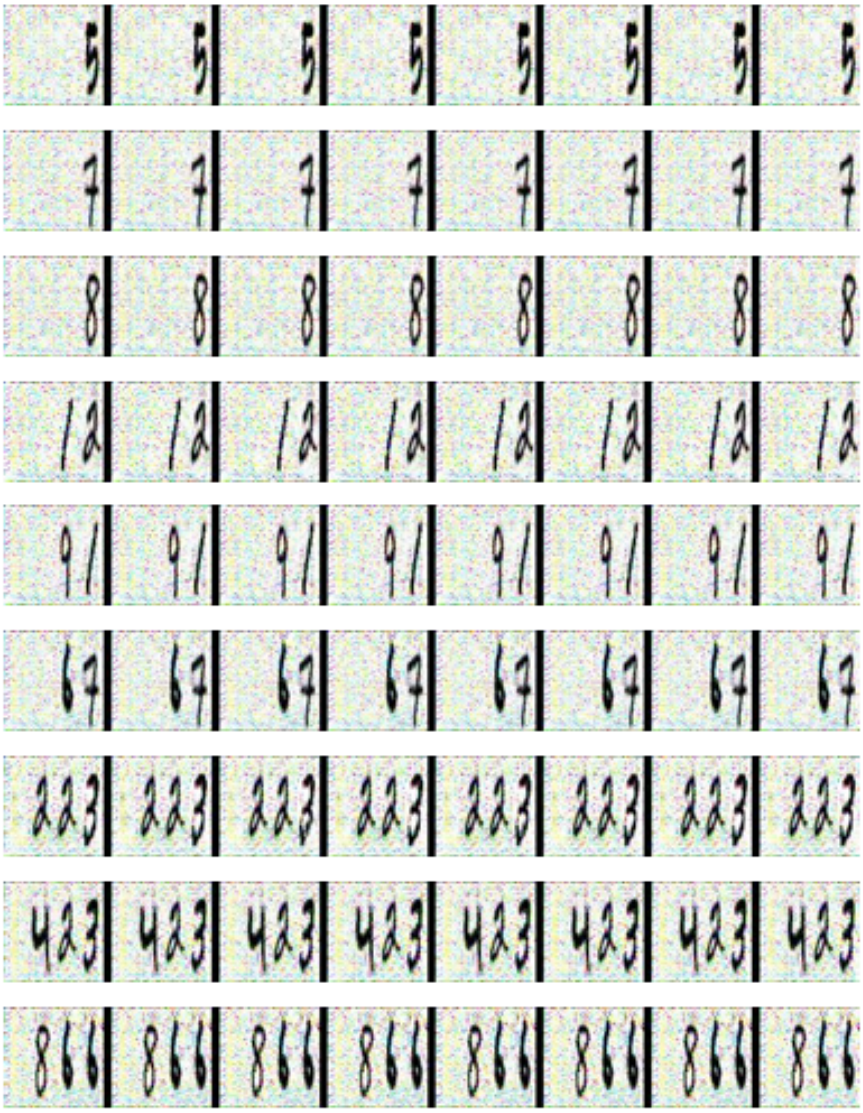

Figure 14: Setting 4: The images generated are corresponding to the captions listed here in their respective order – Five (5), Seven (7), Eight (8), One Two (12), Nine One (91), Six Seven (67), Two Two Two (222), Three Four Two (342), Six Six Eight (668)

Setting 5: Text encoding of numbers using bigrams

In the subsequent settings we experimented with the textual representation of the numbers. We found that, while skip thought vectors captured the sentences well, it made more sense to represent the number system through words. Hence, for this setting we replaced the skip thought vectors with bigram model. In this model, we created a list of all bigrams for 7 digit numbers. For creating the vectors of a number we counted the number of occurrences of that bigram in that number's English words.

Generation of 7 digit numbers

For this experiment we used English words as the caption of the numbers. As an example, caption for '34' is 'Thirty Four'. We trained our model on 600 seven digit numbers and the results were a huge improvement over those obtained using skip thought vectors. Though the system was unable to learn how to place zeros in places where no number was explicitly mentioned. For example: the system had hard time to generate an image for one million and one. But we think that is because of lesser samples of this type in the dataset and can be easily improved by a richer dataset. Please refer to the figure - 15

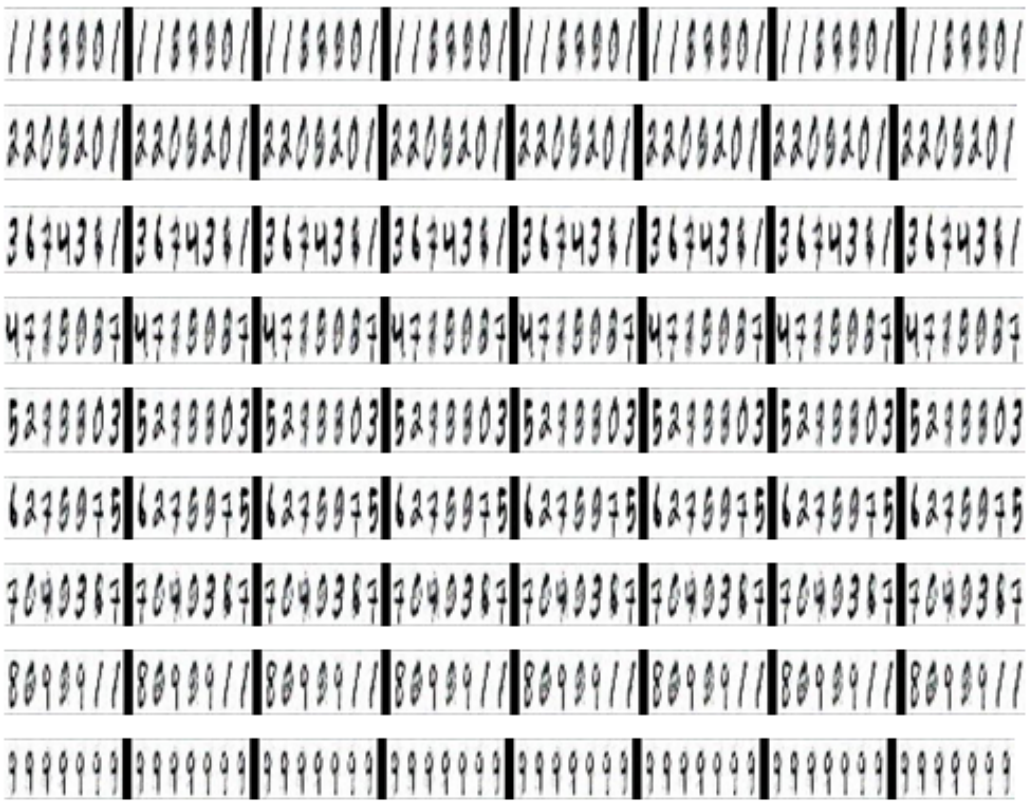

Figure 15: Setting 5 (Generation of 7 digit numbers): The images generated are corresponding to the captions listed here in their respective order – One Million One Hundred One (1,000,101), Two Million Two Hundred Thousand Two Hundred One (2,200,201), Three Million Six Hundred Seventy Four Thousand Three Hundred Sixty One (3,674,361), Four Million Seven (4,000,007), Five Million Two Hundred Three (5,000,203), Six Million Twelve Hundred Seventy Five (6,001,275), Seven Million Forty Thousand Three Hundred Sixty Seven (7,040,367), Eight Million Ninety Thousand Nine Hundred Eleven (8,090,911), Nine Million Nine Hundred Ninety Nine Thousand Nine Hundred Ninety Nine (9,999,999)

Generation of variable length numbers

For this experiment we trained our model on 2100 images of 1-7 digit numbers. We found that the model was able to generate images of different lengths with high accuracy. The highlight of this technique is that the system was able to learn 1000000 word representations using only 2000 images. Please refer to the figure – 16

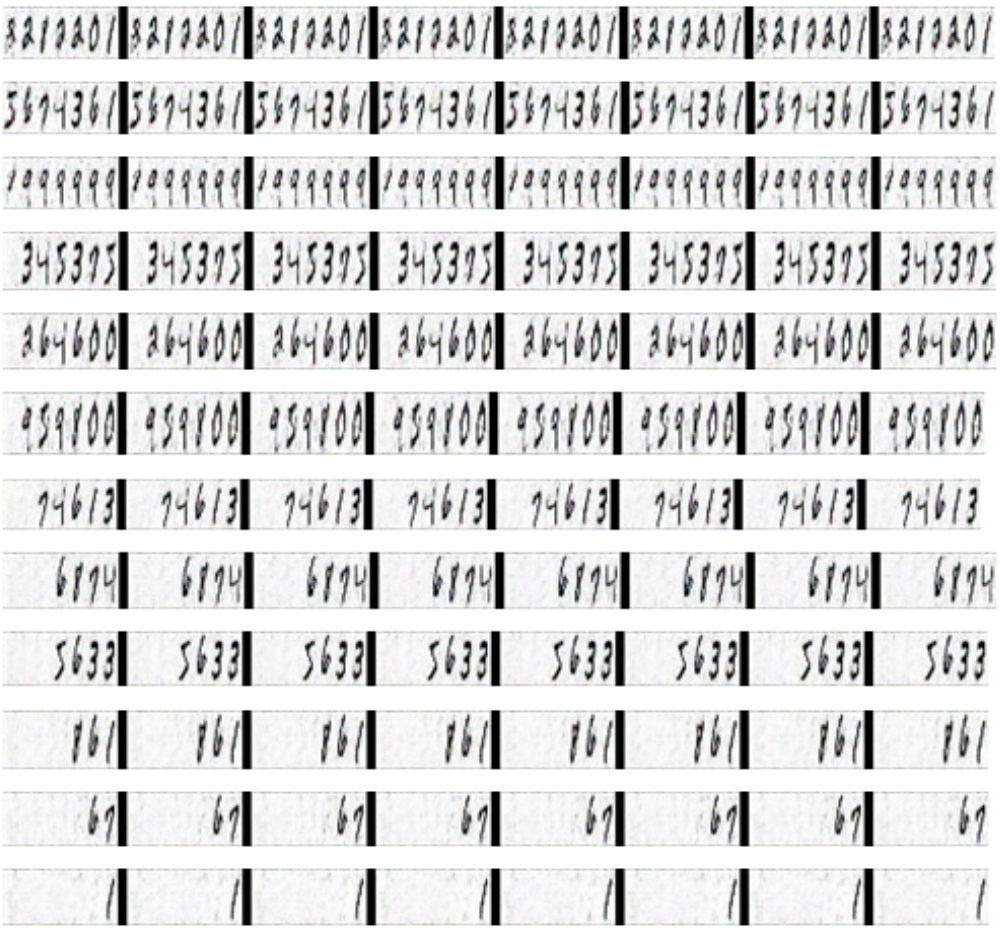

Figure 16: Setting 5 (Generation of variable length numbers): The images generated are corresponding to the captions listed here in their respective order – Two Million Two Hundred Thousand Two Hundred One (2,200,201), Five Million Two Hundred Three (5,000,203), Nine Million Nine Hundred Ninety Nine Thousand Nine Hundred Ninety Nine (9,999,999), Three Hundred Ninety Thousand Three Hundred Forty Five (390,345), Two Hundred Sixty Four Thousand Six Hundred (264,600), Nine Hundred Nine Thousand Eight Hundred Fifty (909,850), Seventy Four Thousand Six Hundred Thirteen (74,613), Six Thousand Eight Hundred Seventy Four (6,874), Five Thousand Six Hundred Thirty Three (5,633), Eight Hundred Sixty One (861), Sixty Seven (67), One (1)

Setting 6: Text encoding using binary representation of numbers

For this setting, we changed the encoding of the numbers and represented them using binary numbers. The captions for the digit images were created by the concatenation of the captions of the individual digits. For example, for the image that corresponds to '212', the caption corresponding is 'two one two'. The dataset had 300 images of 3 digit numbers. We then created the binary representation of numbers and used them as vectors to feed in the GAN. We found that the system was not

able to work well with this type of representation. The quality of the images was bad and the results were less accurate. Please refer to the figure 17

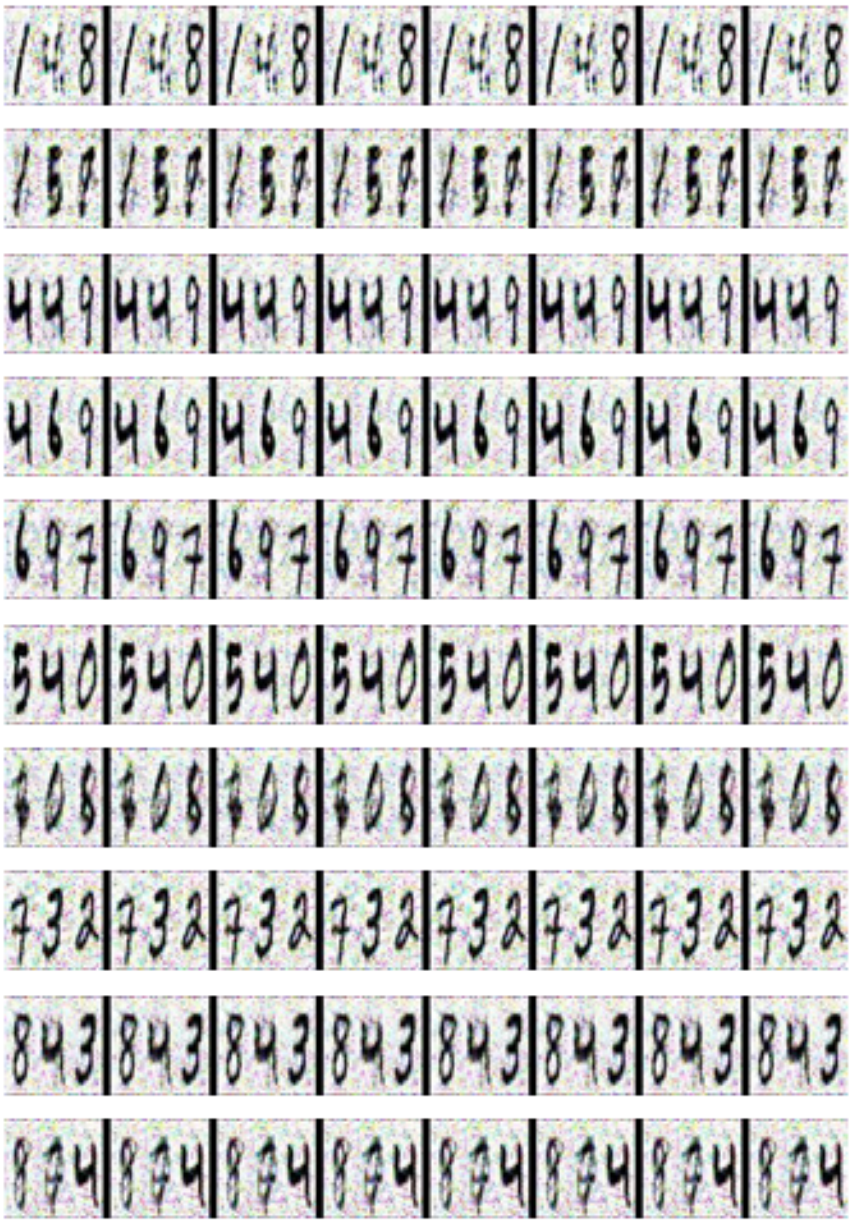

Figure 17: Setting 6: The images generated are corresponding to the captions listed here in their respective order – One Six Four (164), Two Three Nine (239), Three Nine Seven (397), Four Seven Five (475), Five Eight One (581), Six Zero Zero (600), Seven Zero Two (702), Eight Seven Four (874), Eight Seven One (871), Nine One Zero (910)

Setting 7: Text encoding using one hot encoding of numbers

For this setting, we changed the encoding of the numbers and represented them using one hot encoding. The captions for the digit images were created by the concatenation of the captions of the

individual digits exactly like setting 6. We then created the one hot encoding of numbers and used them as vectors to feed in the GAN. We found that with this encoding the system was able to generate all the images correctly. With this experiment, we realized that the method of text encoding plays a paramount role in the generation of the right image. Please refer to the figure – 18

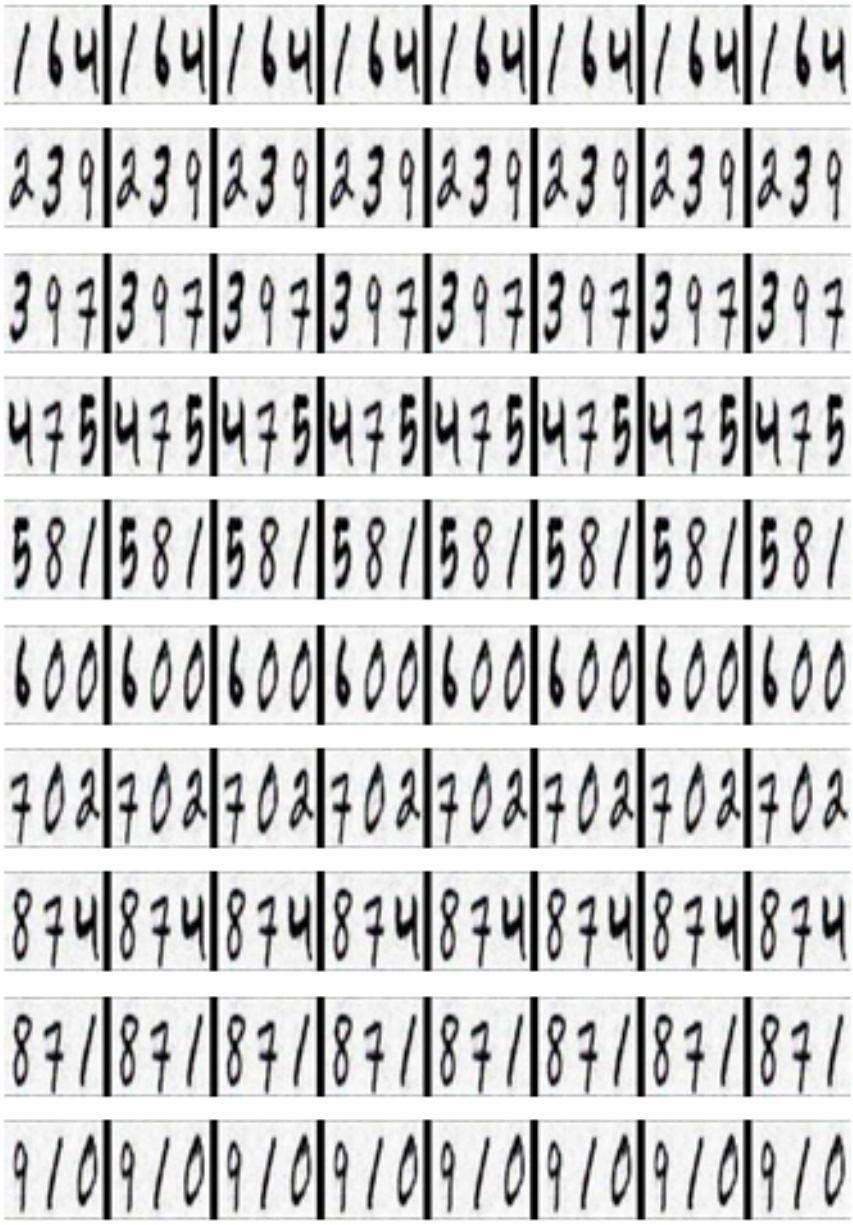

Figure 18: Setting 7: The images generated are corresponding to the captions listed here in their respective order – One Six Four (164), Two Three Nine (239), Three Nine Seven (397), Four Seven Five (475), Five Eight One (581), Six Zero Zero (600), Seven Zero Two (702), Eight Seven Four (874), Eight Seven One (871), Nine One Zero (910)

Setting 8: Text encoding of numbers using word2vec

In this experiment we used word2vec representation of the numbers. Vector space models (VSMs) represent (embed) words in a continuous vector space where semantically similar words are mapped to nearby points. We used continuous bag of words for the vector representation of the numbers.

Generation of 3 digit numbers

For this experiment we used English words as the caption of the numbers. As an example, caption for '34' is 'thirty four'. We trained our model on 300 3 digit numbers and the results were a huge improvement over those obtained using skip thought vectors. This experiment shows that word2vec was able to generate a vector representation that generalizes well for the number system. Please refer to the figure – 19

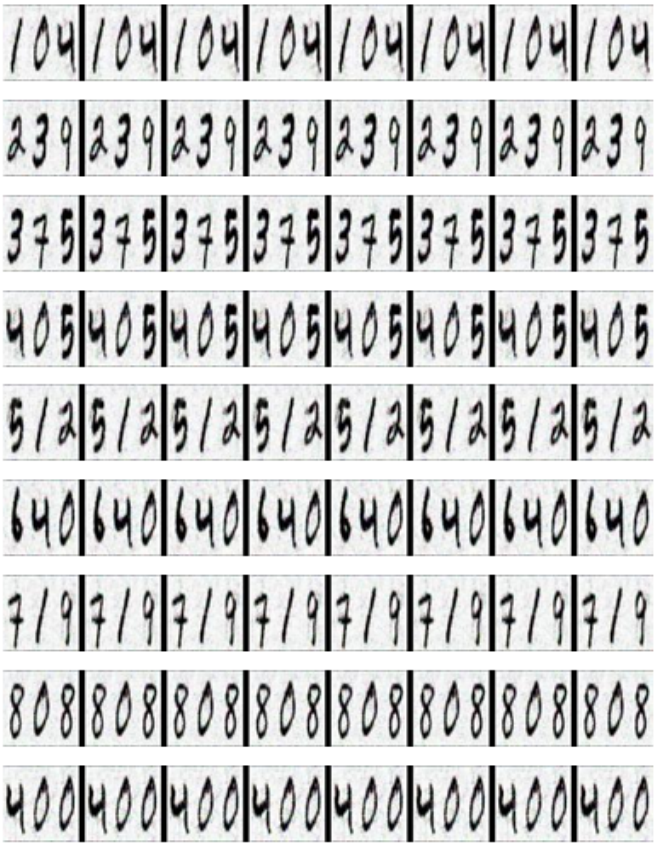

Figure 19: Setting 8 (Generation of 3 digit numbers): The images generated are corresponding to the captions listed here in their respective order – One Hundred Four (104), Two Hundred Thirty Nine (239), Three Hundred Seventy Five (375), Four Hundred Five (405), Five Hundred Twelve (512), Six Hundred Forty (640), Seven Hundred Nineteen (719), Eight Hundred Eight (808), Nine Hundred (900)

Generation of 7 digit numbers

For this experiment we trained our model on 590 images of 7 digit numbers. The images produced by this model were hazy and the model generated wrong images for certain digits. Please refer to the figure – 20

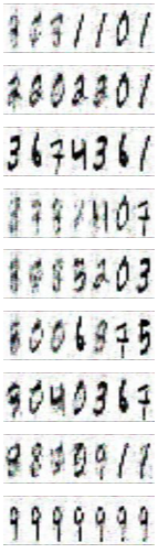

Figure 20: Setting 8 (Generation of 7 digit numbers): The images generated are corresponding to the captions listed here in their respective order – One Million One Hundred One (1,000,101), Two Million Two Hundred Thousand Two Hundred One (2,200,201), Three Million Six Hundred Seventy Four Thousand Three Hundred Sixty One (3,674,361), Four Million Seven (4,000,007), Five Million Two Hundred Three (5,000,203), Six Million Twelve Hundred Seventy Five (6,001,275), Seven Million Forty Thousand Three Hundred Sixty Seven (7,040,367), Eight Million Ninety Thousand Nine Hundred Eleven (8,090,911), Nine Million Nine Hundred Ninety Nine Thousand Nine Hundred Ninety Nine (9,999,999)

7 Discussion and Conclusions :

We were able to test and generalize the architecture well on MNIST dataset. It generalizes well partly because 'Skip Thought Vectors' and other encodings (which is like a skip-gram model), provides us a way to model semantic relationships in a generalized fashion. Another reason it generalizes well is because of the roles played by Generator and Discriminator networks.

Through our analysis, we observed that the results generated by the model on MNIST data were quite insightful. We observed that some of the images generated resembled the digits exactly while a few were a little hazy. For example, the image of 3 generated was inverted.

In the case of generating multiple digits, the network was able to learn the positioning of the digits in most of the cases but in a few cases, it forgets the ordering of the digits to produce. For example, our input of one two three might generate the same set of digits but in a different order (1 3 2).

The results we see from setting 1 are the digit representations the network learn for each digit name. We see that the results generated doesn't exactly resemble the actual digit representations, but it broadly captures the strokes that make that number unique.

The results from the setting 2 are much better than this setting 3, we think that this could be a result of the fixed three digits we choose as the training set. It is could also be because the size of the training set is small, while the semantics is involved in this are much complex than the previous setting. For example, the network should learn 'eleven' means two 'ones' rather than just one digit. Despite this, the three digit numbers it generated are close to the actual representation except the overfit cases where something like 'thirty' is more followed by 'four'.

8 Concept and/or Innovation :

The whole concept behind this project relies on the proper understanding of the Generative Adversarial Networks that carefully leverage the semantic relationships between images and text sentences to generate images. Specifically, we want the network to learn the semantic relationship from the textual description of the number to the pictorial representation using MNIST dataset.

Also, by exploiting the naming convention of the English number system, we want to generate zero shot n-digit numbers by training the network on a smaller set. So, the basic idea behind this project is to create a network that learns the number system through multimodal textual input on the small training set and extend its knowledge to pictorially represent any number. We infer that the GAN model is resilient to tweaking .

How generalizable can the model be ? We explored various approaches where we tried to replace the Skip Thought vectors with other models that define relationships with words over sentences.We explored that specific to our task of making GAN predictions on stricter word modeling gave better results.

We have also experimented and tested out various text encoding algorithms besides skip-thought vectors (which was used in [1]). We replaced the skip thought vector with bigram encoding and word2vec encoding, in which we noticed the performance actually improved.

Certain questions need to be explored . What happens if our dataset is a mix of flowers as well as MNIST dataset. What results shall we observe ? These questions arise due to our approach of dealing with GAN. Not just these two dataset. Can we make an intelligent GAN model that semantically understands human language. CNNs have been quite successful in image segmentation. Can the prediction on Bounding boxes (Object localization) or regions (Image segmentation) be understood by this model ? Generator is essentially a deconvolution. Can the reverse semantic understanding be learned in a Deconv network ? In addition to that, can we leverage encoders such as RBMs (Restricted Boltzmann Machines) capability to encode internal representations as well in the skip thought model to generate more useful vectors ?

9 Future proposals :

Reed et al [1] has implemented the model for either birds or flowers. We have checked the applicability to numbers using the carefully constructed MNIST dataset. In each of these cases, we have limited to either a number, or a bird or a flower. We wish to verify how the model performs on all of them at once. Can there be a unified model for all datasets ?

On another note, we are planning to rectify the defects in the images obtained from the stage-I GAN. The stage-II GAN will take the results from stage-I and text descriptions as input and it will generate high-resolution images with photo-realistic details.The literature regarding the same is mentioned in detail by Han Zhang et al [9].

10 Contributions

We made sure that most of the work is equally distributed and shared our individual contributions for the team assignment so that each of us understood the latters work. Following are roughly the contributions made (see Table 1.): We give special thanks to Paarth Neekhara whose python implementation we used for running inference [8].

Table 1: Contribution by each team member

S.no.	Name	Contribution
1	Swapnil Taneja	Inference on Flower dataset, Integration of code, Experimenting with setting 3
2	Saksham Sharma	Constructing datasets, Experimenting with Setting 1&4(pair programming)
3	Kriti Aggarwal	Experimenting with Setting 1 and 4 (pair programming), Configuring AWS
4	Prahal Arora	Experimenting with Setting 2, 5 and 8, Extending the code for MNIST dataset
5	Saicharan Duppati	Experimenting with Setting 2, 6 and 7, Bug fixes

11 Bibliography

[1] Reed, S., Akata, Z., Yan, X., and Logeswaran, L. Generative Adversarial Text to Image Synthesis. In ICML, 2016.

[2] Reed, S., Akata, Z., Lee, H., and Schiele, B. Learning deep representations for fine-grained visual descriptions. In CVPR, 2016.

[3] Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. The Caltech-ucsd birds-200-2011 dataset. 2011.

[4] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In ECCV. 2014.

[5] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In NIPS, 2014.

[6] Mirza, M. and Osindero, S. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784, 2014.

[7] Denton, E. L., Chintala, S., Fergus, R., et al. Deep generative image models using a laplacian pyramid of adversarial networks. In NIPS, 2015.

[8] https://github.com/paarthneekhara/text-to-image

[9] Zhang, H., Xu, T., Li, H., Zhang, S., Huang, X., Wang, X. and Metaxas, D., StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. In CVPR, 2016

[10]http://carpedm20.github.io/faces/

[11] Ryan Kiros ., Yukun Zhu ., Ruslan Salakhutdinov ., Richard S. Zemel ., Antonio Torralba ., Raquel Urtasun ., Sanja Fidler, Skip-Thought Vectors. In CVPR, 2015

[12] Tomas Mikolov. , Ilya Sutskever et. al. , Distributed Representations of Words and Phrases and their Compositionality. In Neural Information Processing Systems Conference, 2015