Self-supervised Image-to-text and Text-to-image Synthesis

A Thesis

Presented to

The Academic Faculty

by

Anindya Sundar Das

1811MC02

Under the guidance of

Dr. Sriparna Saha

and

Dr. Iryna Gurevych

In Partial Fulfilment
of the Requirements for the M.Tech Degree of

Department of Mathematics
Indian Institute of Technology, Patna
July, 2020

To my parents
Mr. & Mrs. Ajay and Namita Das
and
to my beloved sister
Ananya
Without your support, encouragement, love
None of my success would be possible.

THESIS CERTIFICATE

This is to certify that the thesis titled **Self-supervised Image-to-text and Text-to-image Synthesis**, submitted by **Anindya Sundar Das**, to the Indian Institute of Technology, Patna, for the award of the degree of **Master of Technology**, is a bonafide record of the research work done by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

[illegible]

Dr. Sriparna Saha	**Dr. Iryna Gurevych**
Research Guide	Research Guide
Associate Professor	Professor
IIT - Patna, Bihta	UKP Lab, TU Darmstadt
Bihar- 801103, India	Darmstadt-64289, Germany
Place:	Place: Darmstadt, Germany
Date:	Date: 16.07.2020

DECLARATION

I certify that

a. The work contained in this thesis is original and has been done by myself under the general supervision of my supervisors.

b. The work has not been submitted to any other Institute for degree or diploma.

c. I have followed the Institute norms and guidelines and abide by the regulation as given in the Ethical Code of Conduct of the Institute.

d. Whenever I have used materials (data, theory and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the reference section.

e. The thesis document has been thoroughly checked to exclude plagiarism.

Anindya Sundar Das

Anindya Sundar Das
1811MC02

CERTIFICATE OF APPROVAL

This is certified that the thesis entitled **Self-supervised Image-to-text and Text-to-image Synthesis**, submitted by **Anindya Sundar Das** to Indian Institute of Technology Patna, for the award of the degree of M.Tech has been accepted by the examination committee and that the student has successfully defended the thesis in the viva-voce examination held today.

[illegible]

Prof.Dr. Iryna Gurevych

(Supervisor) **(Supervisor)**

(External Examiner) **(Internal Examiner)**

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my supervisor Dr. Sriparna Saha for continuous support and valuable suggestions during my M.Tech thesis work and research at IIT Patna. Throughout my M.Tech period she has nurtured my research abilities through her immense knowledge, insightful comments, incessant motivation. I cherish the learning experience and believe her knowledge and ideas would reverberate throughout my professional journey. I would also like to express my sincere gratitude to Prof. Dr. Iryna Gurevych for giving me an opportunity to work at her extremely qualified research group at Ubiquitous Knowledge Processing (UKP) Lab at TU Darmstadt, Germany, leading me working on exciting projects and allowing me to be part of the research collaborations between India and Germany under DAAD KOSPIE scholarship program. I would also like to extend my warmest regards and thanks to Mr. Jonas Pfeiffer for guiding and enriching me with his knowledge and giving me his support in every adverse situation during my research work.

Lastly, I would like to take the opportunity to express my gratitude and love towards my friends and family for being there with me when I needed them the most, for standing by me through thick and thin.

ABSTRACT

A comprehensive understanding of vision and language and their interrelation are crucial to realize the underlying similarities and differences between these modalities and to learn more generalized, meaningful representations. In recent years, most of the works related to Text to Image synthesis and Image to Text generation focused on supervised generative deep architectures to solve the problems, where very little interests were placed on learning the similarities between the embedding spaces across modalities. In this thesis, we propose a novel self-supervised deep learning based approach towards learning the cross-modal embedding spaces; for both Image to Caption and Caption to Image generations. In our approach, we first investigated different deep neural architectures to obtain a dense vector representation of images in an autoencoder based set up; next we explored text autoencoder models to obtain the dense vector representation on sentence level; then we attain the mapping from embedding space of one modality to embedding space of the other modality by minimizing maximum mean discrepancy between the probability spaces of two embeddings and also by using traditional generative models (GANs) and did a comparative study of these methods. Qualitative and quantitative evaluation of our unsupervised models, demonstrates that it learns to generate textual description from image data as well as image from textual data. However the semantic correlation between the two modalities is found to be low in this self-supervised approach.

[Keywords: Cross-modal, semantic space, embedding space, maximum mean discrepancy, mapping network]

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

The web contains a multitude of images; most of the content images are unannotated. Describing the content of an image automatically using proper natural languages is a very vital task relevant to the area of both Computer Vision and Natural Language Processing, the impact of which could be significant, for example, it will help visually impaired people to have a better understanding of the content of images on the web. Image captions in the form of natural languages make the content more accessible that can be presented using existing text-to-speech systems (Dutoit, 1997) to the visually impaired people. It has many other important applications such as photo and video sharing in social media, semantic visual search (Fan, 2011), visual intelligence in chatbots (Das et al., 2017). The reverse problem is the generation of realistic images from the human-written descriptions such as *"this small bird has red and pointy beak"* or *"this bird is blue with brown and white on its back and has a long, pointy beak"*. Although notable progress has been made in generating visually realistic images but those are still far from this goal. The Generative Adversarial Network (Goodfellow et al., 2014) based models showed promising results by generating many probable visual representations of a given textual description (S. Reed et al., 2016). Recent deep architectures such as GAWWN (S. E. Reed et al., 2016), GAN-INT-CLS (S. Reed et al., 2016), StackGAN-v2 (Zhang et al., 2018) outperform previous state-of-the-art text to image synthesis models in terms of generating plausible visual representations of text data.

However, one difficult issue in both text to image synthesis and image to text generation is, the recent state-of the-art deep models are based on supervised learning which requires annotated data; while most of the available web data is unannotated, also annotations require human intervention, hence expensive. Similar problems have been addressed in machine translation where unsupervised machine translation (Lample, Conneau, Denoyer, & Ranzato, 2017) attains astounding results even in cases of low-resource languages, with monolingual corpora only and also performs reasonable well in case of distant languages (Lample, Ott, Conneau, Denoyer, & Ranzato, 2018). Motivated by the success in machine translation, we aim to investigate whether it is possible to learn the cross-modal embedding spaces between visual and textual data in a self-supervised approach.

1.1 Related Works

Caption generation using neural networks was first proposed in the paper (Kiros, Salakhutdinov, & Zemel, 2014) which used multimodal log-bi-linear model. In this paper the authors (Vinyals, Toshev, Bengio, & Erhan, 2015) used deep convolution neural networks (CNN) as image encoder while RNN as decoder which generates captions. The authors of (Xu et al., 2015), introduced attention-based approach for image caption generation which uses convolutional networks for image feature extraction and attention based RNN as decoder for caption generation. A deep visual semantic captioning model is proposed in (Venugopalan et al., 2017) that takes the advantages of external sources and exploits the semantic information to generate captions and can describe the objects not present in image-caption datasets. Recently, a caption generation model that is based on the dependencies between caption words, image regions and RNN language model has been proposed in (Pedersoli, Lucas, Schmid, & Verbeek, 2017). There have been several works on generation of text for a given image, but there are also the ones that generate an image for a given text. Generative Adversarial Networks(GANs) (Goodfellow et al., 2014) are proven to be useful in generating photo-realistic images (S. Reed et al., 2016) (Zhang et al., 2017). StackGAN-v2 (Zhang et al., 2018) employs a tree like structure comprising of multiple generators and discriminators. It generates images at multiple scales at different branches of the tree; achieving state-of-the art performance.

Our work is as closely related to machine translation as it is related to image-caption generation. A recent work in unsupervised machine translation (Lample et al., 2017) uses shared encoder decoder architecture and maps the sentences from two different languages into common latent space. Another work (Artetxe, Labaka, Agirre, & Cho, 2017) uses different decoders while shares the encoders utilizing monolingual corpora only. A work related to unsupervised learning (C.-L. Li, Chang, Cheng, Yang, & Póczos, 2017) that employs maximum mean discrepancy (MMD) (Tolstikhin, Sriperumbudur, & Schölkopf, 2016) based on two sample test minimizes the distance (MMD) between the two distributions to obtain the mapping. Leveraging on the ideas and advances in these previous related works, we propose a novel architecture for cross-modal (image to text and vice versa) generations that utilizes GANs and MMD GANs (C.-L. Li et al., 2017) for unsupervised learning.

In this thesis our contribution is, first to develop an generative deep image autoencoder setup which finds compelling image vector semantic space as per the reconstructions are concerned. Secondly, we develop a recurrent neural network based autoencoder which embeds sentences in textual semantic space. Third we obtain the cross-modal semantic space mapping using both Maximum Mean Discrepancy (Tolstikhin et al., 2016) based generative networks (C.-L. Li et al., 2017) which uses adversarial kernel learning technique and Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), thus generates one modality at the output given the other modality as input without any supervision. Experimental results on Caltech-UCSD Birds-200-2011 data set and Oxford-102 Flower data set illustrate that this unsupervised caption and image generation technique learns to generate one modality from other, however on semantic level we find correlation between textual description and the image . Comparative analysis shows the proposed image autoencoder perform well in-comparison to other baselines and GAN performs slightly better than MMD-GAN based mapper specifically for image-to-text case, for determining the correspondence between embedding spaces.

The contribution made in this research work can be summarized as follows:

- A novel approach to obtain dense image embedding using StackGAN based semantic Image autoencoder setup.
- An LSTM based text autoencoder setup to obtain dense sentence embedding.
- GAN and MMD-GAN based embedding space mapping networks that obtain the cross-modal mapping in unsupervised fashion.

Rest of the thesis work is arranged in the subsequent chapters as follows:
Chapter 2, explores with different approaches for obtaining image embeddings using various autoencoders and a comparative analysis of their performances.
Chapter 3 investigates the text autoencoders for obtaining the text embeddings on sentences label.
Chapter 4 discusses the mapping networks based on MMD and GANs that attain the crossmodal translations between the semantic spaces.
Lastly, chapter 5 concludes this research work and discusses future scope and direction in this domain.

Chapter 2

A Deep Image Autoencoder for Image Embedding

2.1 Introduction

The first step is to obtain a dense vector representation of the images. The objective is to reduce redundancy and irrelevant information, while maintaining the quality of the reconstruction. The dense representation is required for this task as presence of gaps and discontinuity in the latent space will directly affect the cross modal mapping network in handling with the mapping of those gaps in semantic spaces. To achieve this we have experimented with different autoencoder architectures and did a thorough analysis of their performances in order to attain the best structure of this task.

2.2 Research Methods

2.2.1 Variational autoencoder

The inherent problem with autoencoders for generation task is that the latent space or the encoded vector space, obtained from the inputs, may not be continuous. Gaps inside the cluster means when samples are taken from the latent space for generations, decoder has no idea how to handle with those gap regions in encoding space; thus generates unrealistic outputs.
Variational autoencoder (VAE) (Kingma & Welling, 2013) solves this problem by generating continuous latent space which makes random sampling and interpolation easier. It achieves so using reparameterization trick as described in (Kingma & Welling, 2013); instead of encoder generating an n-dimensional vector of real values, the encoder outputs two n-dimensional vectors: the first vector (μ) is a vector of means and the second vector (σ) is a vector of standard deviations. The center of the encoding of an input is decided by mean vector while the variation of the encoding area is controlled by standard deviations.

However since there is no restrictions on what values the mean vector μ and standard deviation vector σ can take on, the encoder may learn to generate very different μ for different classes, thus grouping the clusters apart while minimizing σ, which further reduces uncertainty for the decoder. However we want our encodings as close as possible while still being distinct for different classes. In order to achieve this, we add Kullback–Leibler divergence into the loss function. The KL divergence measures how much the two probability distributions diverges from each other. We minimize KL divergence loss to resemble the encoding distribution (μ and σ) to standard normal distribution. VAE architecture from paper (Hou, Shen, Sun, & Qiu, 2017) is used for this task.

2.2.2 Resnet autoencoder

As the network goes deeper, vanishing gradient problems becomes much worse; Resnet solves this problem by using skip connections and identity networks. In the first model (Figure 2.1) we have used resnet-50 to obtain the image encoding. The last layer of resnet comprises of average pooling followed by a liner dense layer. We override the last layer to obtain image encodings of dimension 1024; The pretrained resnet encoder is fine tuned while training. The decoder we used in this model comprises of several layers of upsampling followed by deconvolution layer as used in (Zhang et al., 2017).

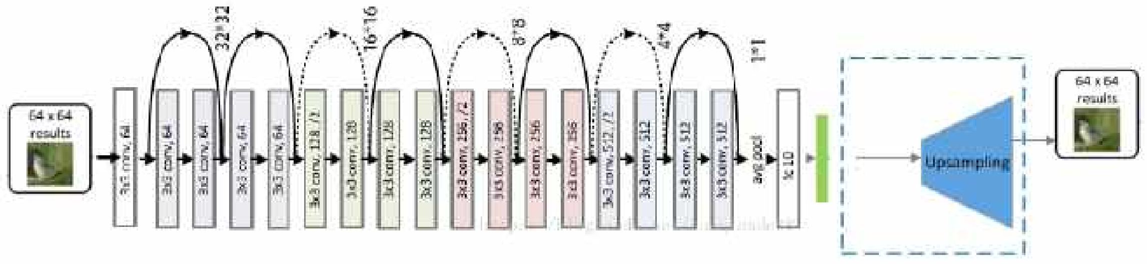


Figure 2.1: Resnet based Image Autoencoder

2.2.3 Com-Rec Autoencoder

The second architecture is completely CNN based encoding and decoding network. We have used the model from (Jiang et al., 2017). The first CNN takes input images and produces feature maps after each convolutional layers and learns a dense representation of the input images which preserves the structural information and then a second

CNN, comprises of upsampling followed by several covolutional layer reconstructs the decoded image. In the decoder, linear interpolation is applied to obtain an up-scaled image. The decoder network produces a residual image which is added to the up-scaled image to generate the final output image. We have experimented with different variants of the original architecture. The original architecture has a tensor of dimension $3 \times 32 \times 32$. Along with the original architecture we experimented with one variant in which we reduced the bottleneck size to $3 \times 16 \times 16$ and in other variant we flattened the bottled neck output and applied to a fully connected layer in the compression side to obtain the dense embedding and also a fully connected layer and reshaping at the reconstruction side is introduced as shown in Figure 2.2.

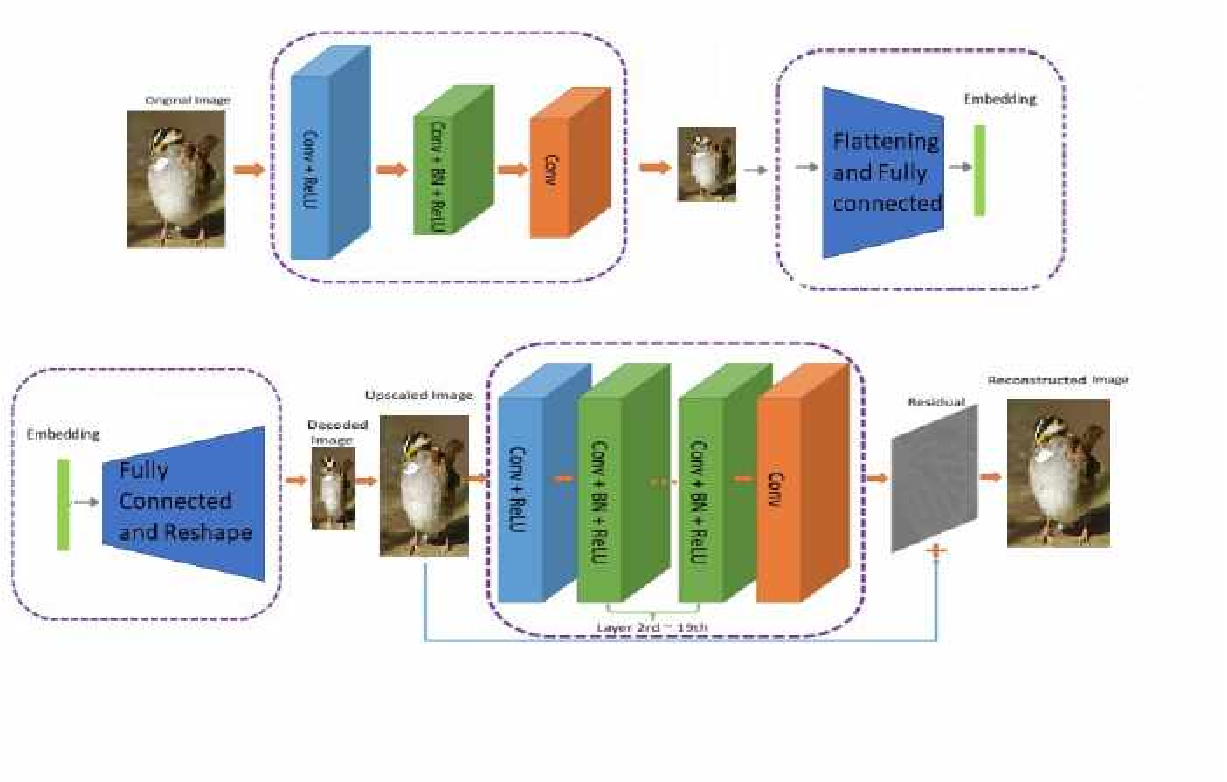


Figure 2.2: Com-Rec CNN based Image Autoencoder

2.2.4 Compressive autoencoder

Our next architecture is based on model as discussed in the paper (Theis, Shi, Cunningham, & Huszár, 2017) which comprises on an encoder, a decoder and a probabilistic model. In this architecture noise is added in the bottleneck to obtain the embedding (Figure 2.3) . The bottleneck layer is $32 \times 32 \times 32$ dimensional tensor. We have modified the bottleneck layer to obtain 1024 dimensional vector representation of the images.

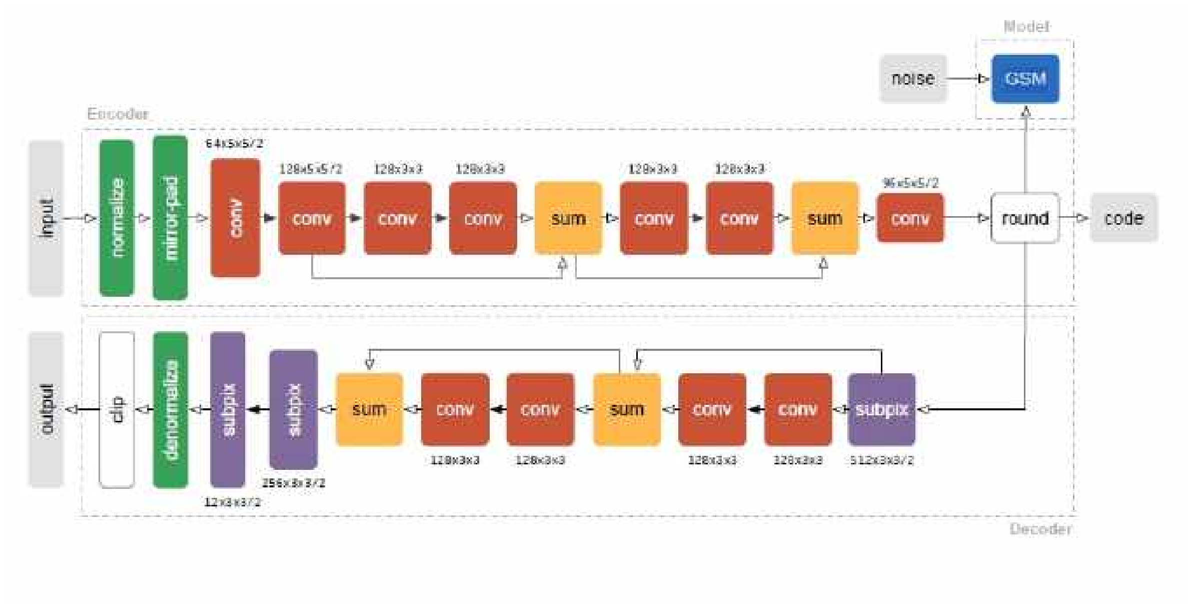


Figure 2.3: Compressive Autoencoder

2.2.5 Com-Rec autoencoder with GAN

In order to improve the performance of the model mentioned in 2.2.3, adversarial training has been incorporated using generative adversarial network (Figure 2.4). Single stage discriminator from the paper (Zhang et al., 2017) has been used in this modified architecture. Generator loss along with the reconstruction loss were optimized during the generator network training.

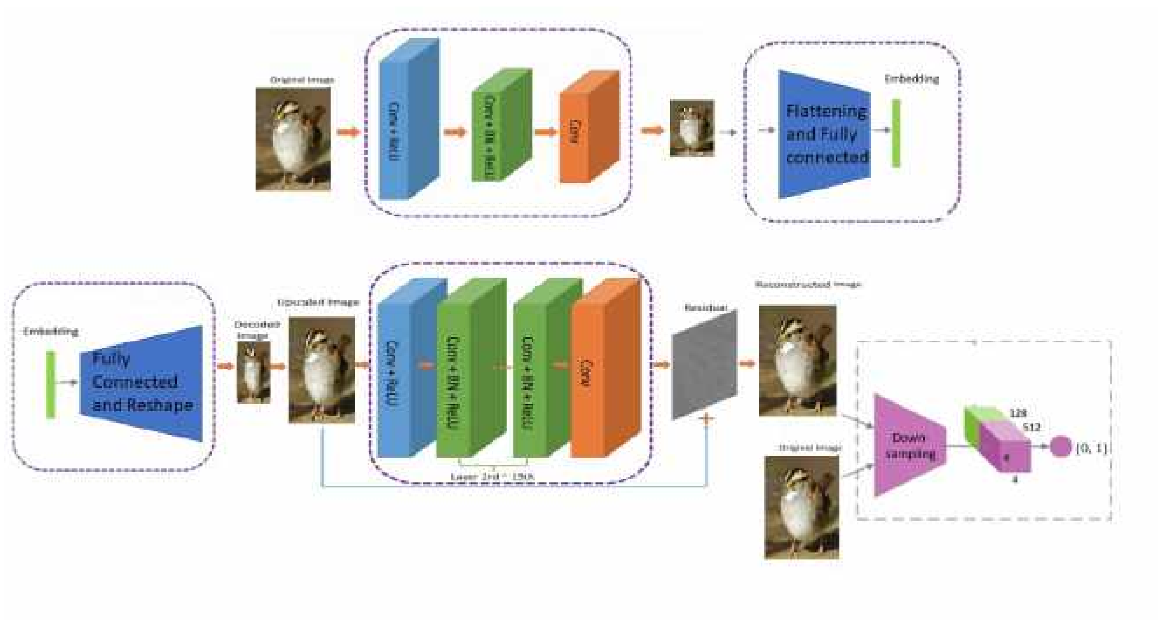


Figure 2.4: Com-Rec Autoencoder with GAN

2.2.6 StackGAN-v2 based Image autoencoder

Our proposed model for image autoencoder is StackGAN-v2 based. As discussed in the original paper (Zhang et al., 2018), StackGAN-v2 takes text embedding vector as input and produces images of increasing resolution at different branches of the network. We modified the architecture to make it an encoder-decoder based network as shown in Figure 2.5, where an encoder takes an image, extracts different features at different layers of the deep network to obtain an image embedding; this image embedding subsequently is fed as input to original conditional stackGAN-v2 model which reconstructs the image at the output of the conditional stackGAN-v2 thus working as an image autoencoder. For encoder part of the autoencoder, we use resnet-50 (He, Zhang, Ren, & Sun, 2016) which employs residual learning to extract the features. We redefine the last layer of resnet-50 to obtain 1024 dimensional image embedding. Our decision of choosing resnet as an encoder and stackGAN-v2 as decoder are motivated by following reasons:

1. Though standard convolutional autoencoders (Jiang et al., 2017) (Theis et al., 2017) perform well in reconstructing images, the bottleneck is a three-dimensional tensor instead of a vector. For our work we need a dense vector representation in the bottleneck. There are evidences that reducing bottle-neck height and width adversely affects the image reconstruction quality (Manakov, Rohm, & Tresp, 2019). We also experimentally verified, that replacing the bottleneck tensor in standard state-of-the art autoencoders with vector greatly impacts reconstruction quality. On the other hand, resnet is very well-known for extracting image features and performs astoundingly well in image classification task, also it suits our requirement of an encoder as the output is a vector instead of a multidimensional tensor.

2. However Variational autoencoder (Kingma & Welling, 2013) solves the problem in which bottleneck is a vector from continuous latent space. However it suffers from blurry generation problem which becomes more cumbersome as image data becomes more complex. For example authors of the paper (Shi, Siddharth, Paige, & Torr, 2019) used an Euclidean distance based look-up table approach instead of directly generating the image space to avoid blurry generations.

3. To solve the problem of blurry generations and for effective reconstruction of the complex image space, generative adversarial networks as decoder seems to be the correct choice; as GANs can start from any base distribution $\mathbb{P}_{\mathcal{Z}}$ and train the generative network g_θ to find a distribution $\mathbb{P}_\theta$, such that $\mathbb{P}_\theta \approx \mathbb{P}_{\mathcal{Z}}$, where $g_\theta(z) \sim \mathbb{P}_\theta$. Moreover, the motivation behind choosing stackGAN-v2 as decoder is that it specifically models the data distributions at different scales, so if real data distribution at any scale supports the model distribution of the respective

scale, it stabilizes the training of the entire network. As a result, the first branch could focus on basic details such as structures and color, while other generators at the succeeding branches could focus on other detailed features (Zhang et al., 2018).

For our work we have used the stackGAN-v2 decoder for conditional image generation; the image is conditioned on image-embedding vector. As described in Figure 2.5, firstly the resnet encoder encodes image i into image embedding ψ_i. Then Conditional Augmentation technique (Zhang et al., 2018) is applied on image embedding ψ_i to produce continuous latent space, yielding condition variable $\hat{c} \sim \mathcal{N}(\mu(\psi_i), \sigma(\psi_i))$. The following Kullback-Leibler (KL) divergence loss term is optimized during generator training which acts as regularization that ensures the smoothness of the conditioning variable distribution:

$$D_{KL}(\mathcal{N}(\mu(\psi_i), \sigma(\psi_i))|||\mathcal{N}(0,1)) \tag{2.1}$$

For each generator, G_i, at different branches of the tree, the hidden feature h_i at i-th branch is computed as $h_0 = Fn_0(c, z)$ and $h_i = Fn_i(h_{i-1}, c)$ where Fn_i is neural network, $i = 0, 1, ..., n-1$; n is the total number of branches. Generators at different stages produce images $u_i = G_i(hi)$ from low-to-high resolutions gradually adding more details. Both conditional loss and unconditional loss are being optimized while training the discriminator, D_i:

$$\begin{aligned} Loss_{D_i} = &-\mathbb{E}_{x_i \sim p_{data_i}}[\log(D_i(x_i)] - \mathbb{E}_{u_i \sim p_{G_i}}[\log(1 - D_i(u_i)] \\ &- \mathbb{E}_{x_i \sim p_{data_i}}[\log(D_i(x_i, c)] - \mathbb{E}_{u_i \sim p_{G_i}}[\log(1 - D_i(u_i, c)] \end{aligned} \tag{2.2}$$

The unconditional loss dictates whether the image at discriminator input is fake or real; the conditional loss decides whether the generated image corresponds to respective condition or not (Figure 2.5). During training of the generators, the following loss function is being optimized:

$$Loss_{G_i} = -\mathbb{E}_{u_i \sim p_{G_i}}[\log(D_i(u_i)] - \mathbb{E}_{u_i \sim p_{G_i}}[\log(D_i(u_i, c)] \tag{2.3}$$

The overall loss function for training generators is given by

$$Loss_G = \sum_{0}^{n-1} Loss_{G_i} \tag{2.4}$$

In order to maintain the consistency in color and basic structure among the generated images at different branches from same input, a color-consistency loss term is optimized which minimizes the differences in mean and standard deviations of the generated

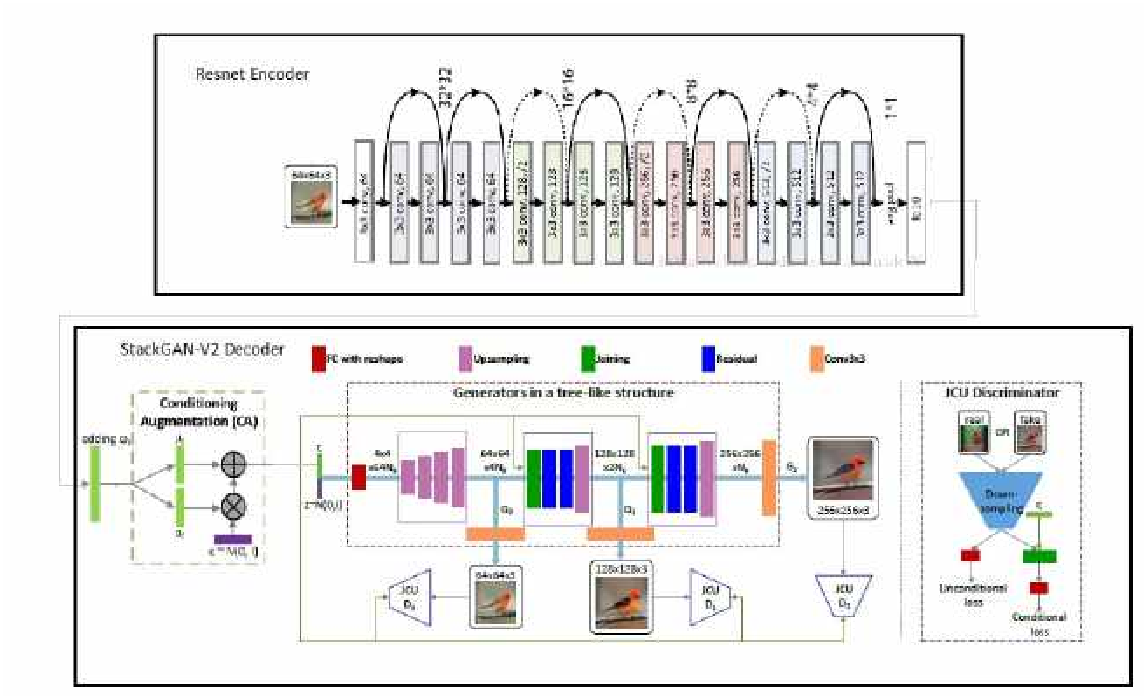


Figure 2.5: StackGAN-v2 Image Autoencoder. c is conditioning variable; N_g, N_d are number of channels

images at different scale and the input image . Let $p_i = (r, g, b)^T$ represents an image pixel value in a generated or an input image with total N number of pixels in a given image, the pixel mean and pixel co-variance is given by $\mu = \sum_i p_i/N$ and $\Sigma = \sum_i (p_i - \mu)(p_i - \mu)^T/N$. Then the color-consistency loss is given by

$$Loss_C = \frac{1}{B}\sum_{k=1}^{B}\sum_{j=1}^{n}(p_1\|\mu_{S_j^k} - \mu_{S_{j-1}^k}\|^2 + p_2\|\Sigma_{S_j^k} - \Sigma_{S_{j-1}^k}\|^2) \tag{2.5}$$

where S_j^k denotes k^{th} sample generated by j^{th} generator, B be the batch size, n is the number of generators. S_0^k denotes the k_{th} input sample. Note that color loss contribution due to input image is added in our architecture which acts as reconstruction loss like in normal autoencoders. The default values for p_1 and p_2 are set as 1 and 5 respectively as used in original paper. Color consistency loss $Loss_C$ is added to generator loss in equation 2.4 to obtain the overall loss for generators training i.e. $Loss_G + \alpha * Loss_C$.

2.3 Evaluation

2.3.1 Datset

For the task, we have used publicly available standard Caltech-UCSD Birds-200-2011 Dataset (Wah, Branson, Welinder, Perona, & Belongie, 2011). It contains 11,788 images of 200 birds species, with 312 binary attributes. All attributes are visual in nature; mostly related to shape, color or pattern of the particular part. We split the dataset into train and test set such that the respective classes do not overlap (Refer Table 2.1). The object-image size ratio in the dataset is less than 50%, for 80% of the images, so we pre-processed the images to maintain the object-image size ratio greater than 75%.

Data	**Number of Samples**
Train Images	8,855
Test Images	2,933

Table 2.1: Dataset Statistics: Number of Images samples in Train and Test sets of Caltech-UCSD Birds-200-2011 Dataset

We have also used Oxford-102 flower dataset for our StackGan-v2 autoencoder setup. Oxford-102 contains 8,189 images of 102 different categories of flowers. The dataset is splited into train and test set of non-overlapping classes (Refer Table 2.2)

Data	**Number of Samples**
Train Images	7,034
Test Images	1,155

Table 2.2: Dataset Statistics: Number of Image samples in Train and Test sets of 102-Oxford Flower Dataset

2.3.2 Experimental Setup

All the experiments are performed in python environment. **torch, torchvision, numpy, pandas, PIL** are the libraries required for carrying out the experiments.

Hyperparameters

For the Variational autoencoder and Resnet autoencoder described in section 2.2.1 and section 2.2.2 respectively the embedding dimension is set as 1024 and the best learning rate is found to be 2×10^{-4}. For the Com-Rec autoencoder (Section 2.2.3) setups, the experiments are performed with different bottleneck dimensions $3 \times 32 \times 32$, $3 \times 16 \times 16$ and 1024; learning rate for the autoencoder is set to 10^{-3}. Learning rate for Compressive autoencoder (Section 2.2.4) is set to 2×10^{-4} while different variants of the model bottleneck dimensions $32 \times 32 \times 32$, $16 \times 16 \times 16$ and 1024 are examined to evaluate their performance. The learning rate for both generator and discriminator is set to 10^{-4} in Com-Rec autoencoder with GAN (Section 2.2.5). For StackGan-v2 based autoencoder the generator and discriminator learning rates are set to 10^{-4}, α is set to 40 (Section 2.2.6).

Evaluation Metric

As we used generative network for our StackGan-v2 autoencoder, we chose Inception score (IS) (Salimans et al., 2016) and Fréchet Inception distance (FID) (Heusel, Ramsauer, Unterthiner, Nessler, & Hochreiter, 2017) to evaluate our image model quantitatively[1]. Inception score is quite popular and widely used metric for GANs. Higher the Inception score better is the quality of generated samples. It computes the KL divergence between marginal class distribution, $p(y)$, and conditional class distribution, $p(y|x)$, where y is predicted label by inception model (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2016) and x is the generated image. It measures how the generated samples can be confidently categorized into a specific class and how diverse are the samples generated by the model, however it has limitations as it does not evaluate intra-class diversity or measure the quality of realism in generated samples. FID which measures the distance between generated distribution and real data distribution, is more extensive and effective in evaluating the realism and diversity of samples. Lower FID indicates generated distribution is close to real data distribution. We computed the Inception score and FID on 2,933 samples for birds and on 1,155 for flowers generated from test sets.

2.3.3 Experiments

We trained our Variational, Resnet, Com-Rec, Compressive autoencoders each for 200 epochs and used early-stopping as regularization on validation error. Except for resnet, in all other cases images are resized to 128×128 and normalized to the range from -1 to 1. For Resnet autoencoder, images were resized to 224×224 dimension and normalized using Imagenet dataset mean and standard deviation. We then trained our StackGAN-v2 image autoencoder (Section 2.2.6) for 600 epochs. We used pre-trained resnet-50 encoder (He et al., 2016), the last layer of resnet is redefined to fully connected layer of 1024 dimension, which is also our image vector dimension. During the training of the entire autoencoder, the weight of the resnet except the last layer is kept fixed, while we set the parameters of the remaining network as trainable. As the resnet model is pre-trained on Imagenet dataset (Deng et al., 2009), we resized images to 224×224 and normalized our dataset images using the mean and standard deviation of Imagenet dataset.

[1]For evaluation we have used the scripts from original StackGAN paper https://github.com/hanzhanggit/StackGAN-inception-model

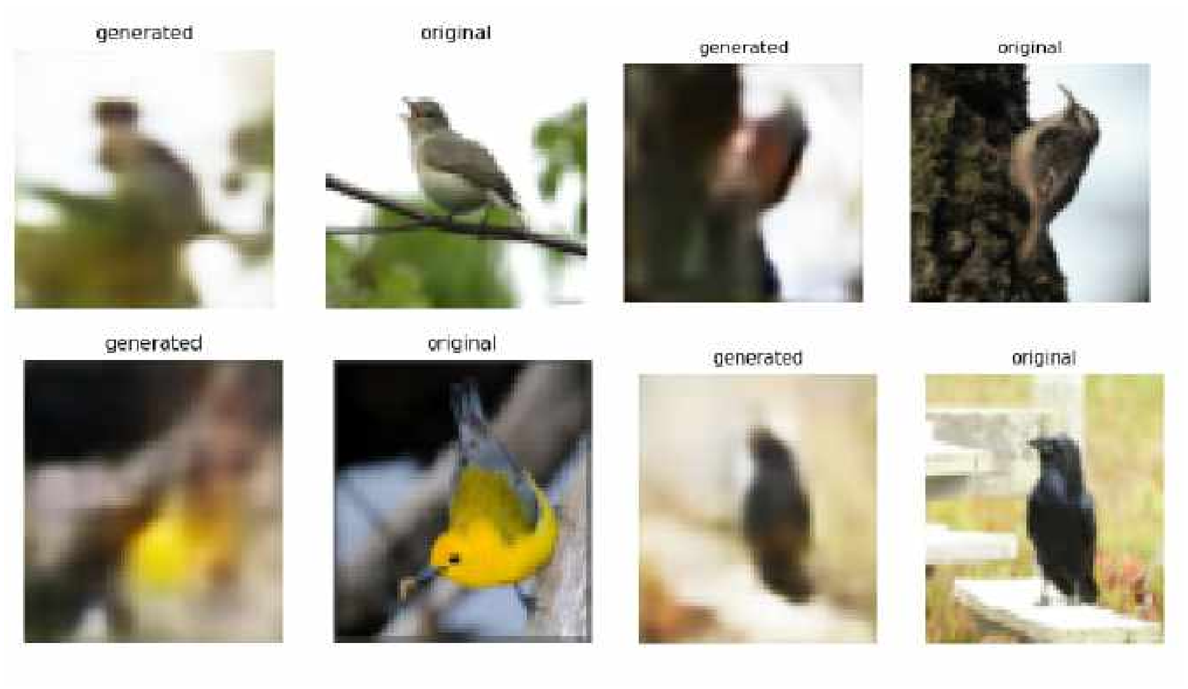


Figure 2.6: Images generated by Variational autoencoder

2.3.4 Evaluation Results and Analysis

We observed that images generated by Variational autoencoder (refer section 2.2.1) are blurry and misses out important features and structural information. Though Resnet is currently state of the art model for image classification related problems, from our observations of generated images it turns out Resnet autoencoders are not good fit for generation related problems.Our third model which is completely CNN based encoder-decoder network (Section 2.2.3) actually regenerates good quality images (Figure 2.8). Here input is $3 \times 128 \times 128$ images and the compressed latent representation is a tensor of dimension $3 \times 32 \times 32$. We observe that as we keep reducing the dimension of latent representation to $3 \times 16 \times 16$ the reconstruction quality deteriorates (Figure 2.9); We observe that as we reduce the compressed representation further from $3 \times 16 \times 16$ dimensional tensor to a vector of dimension 1024, there is a significant amount of information loss and regenerated images are of poor quality (Figure 2.10).

We notice similar phenomena for our Compressive autoencoder model (Section 2.2.4). Reconstructed image quality is best when latent representation is a tensor of dimension $32 \times 32 \times 32$ (Figure 2.11) and declines with reduction in bottleneck size (Figure 2.12). In our third model (Section 2.2.5) we tried to improve the quality of the generated image for dense latent representation by introducing adversarial components. We do not experience any significant improvement in our reconstructed image quality (Figure 2.13). However we are able to regenerate high quality image by using stacked GAN approach as discussed in section 2.2.6. Here though the latent dimension is

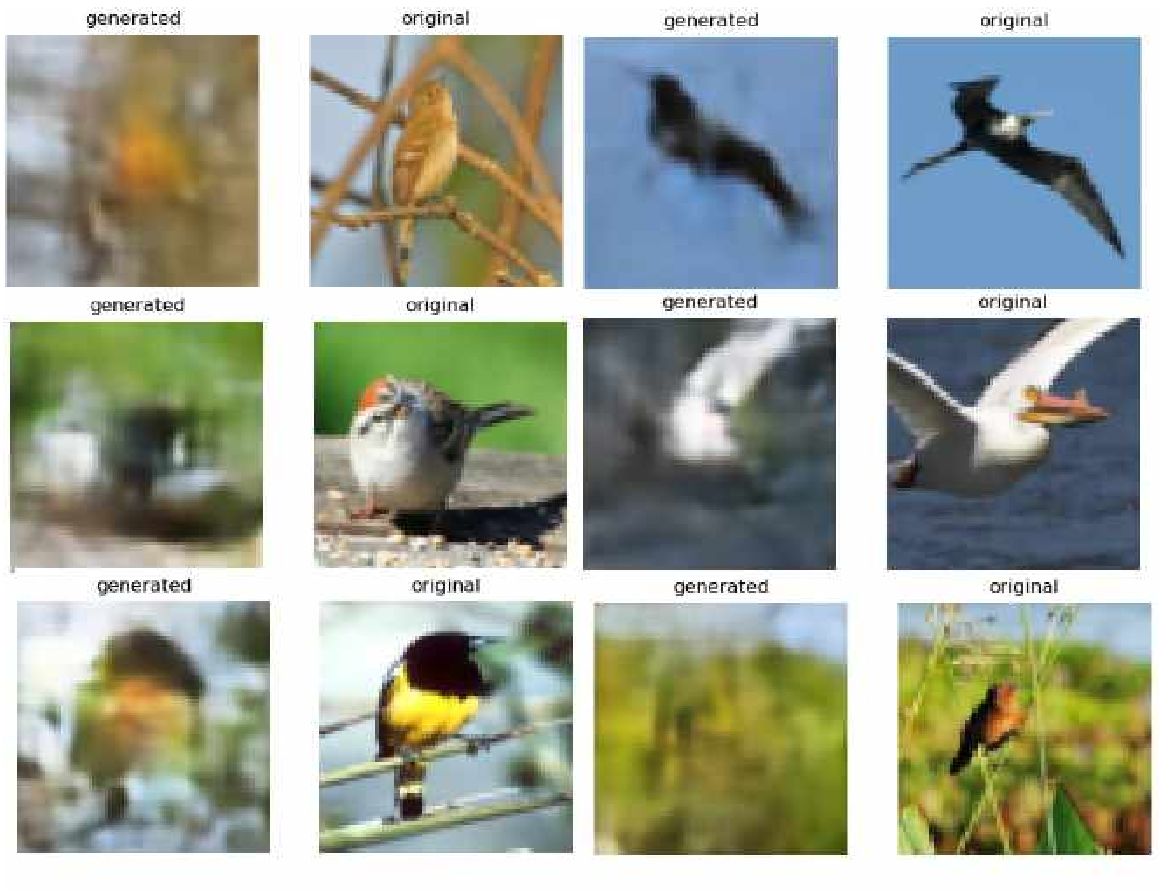


Figure 2.7: Images generated by Resnet autoencoder

a 1024-dimensional vector, the reconstructed images preserves the shapes, structure, color and it generates realistic images. However, as it's a generative network instead of a pure autoencoder the reconstructed images are somewhat different from the actual images while retaining the major attributes and information.

The images generated by StackGAN-v2 autoencoder is shown in figure 2.14 and figure 2.16 along with real images in figure 2.15 and figure 2.17 for qualitative evaluation; Inception score and FID have been reported and compared with the baselines in Table 2.3. All the reported results are statistically significant.

From the qualitative evaluation images generated by different models, it is quite evident that convolutional autoencoders with latent space as tensor generate reasonably good quality images and the reconstruction quality gets poor as the spatial information is lost when latent space becomes a vector instead of a tensor, which can also be verified quantitatively from the results of Table 2.3. StackGAN-v2 based autoencoder actually generates high quality images while satisfying our requirement of having a dense vector latent representation.

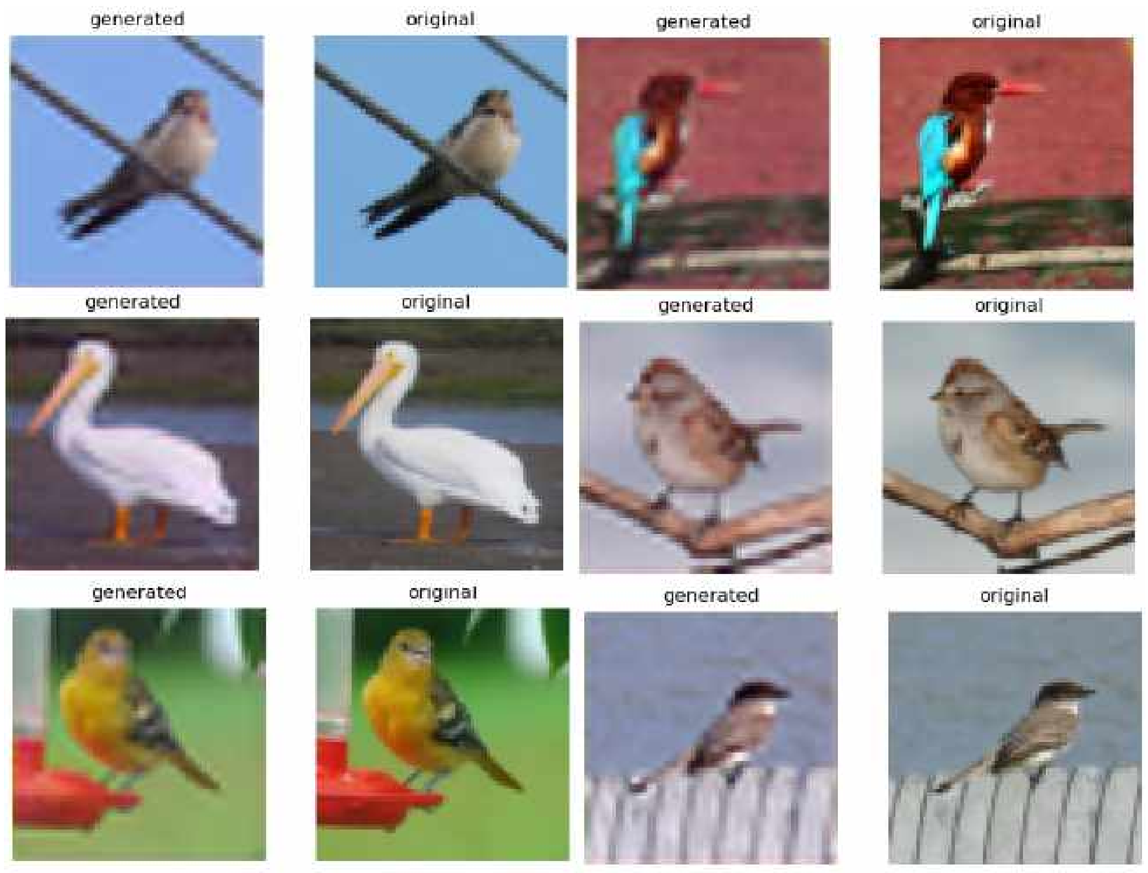


Figure 2.8: Images generated by Model 3: Com-Rec autoencoder with bottleneck dimension $3 \times 32 \times 32$

Ablation Study

We performed ablation study on StackGan-v2 based autoencoder to study how some features are impacting on over all performance. We added pixel based reconstruction loss for images of different scales. However, addition of reconstruction loss does not have any significant impact on reconstruction quality of the images. We have added an embedding loss which is mean square error between the embedding of the image and embedding of reconstructed images, which actually makes the GANs unstable and have adverse effect on the performance. We found that generator loss along with the color consistency loss yields best possible results.

2.4 Conclusion

In this work we are able to figure out the best autoencoder for our task of finding dense vector representation of image space while maintaining reasonable reconstruction quality. Experimental results on the mentioned datasets demonstrate the robustness of our StackGAN-v2 based autoencoder model over other baseline autoencoders by achieving best performance for a compressed latent vector representations.

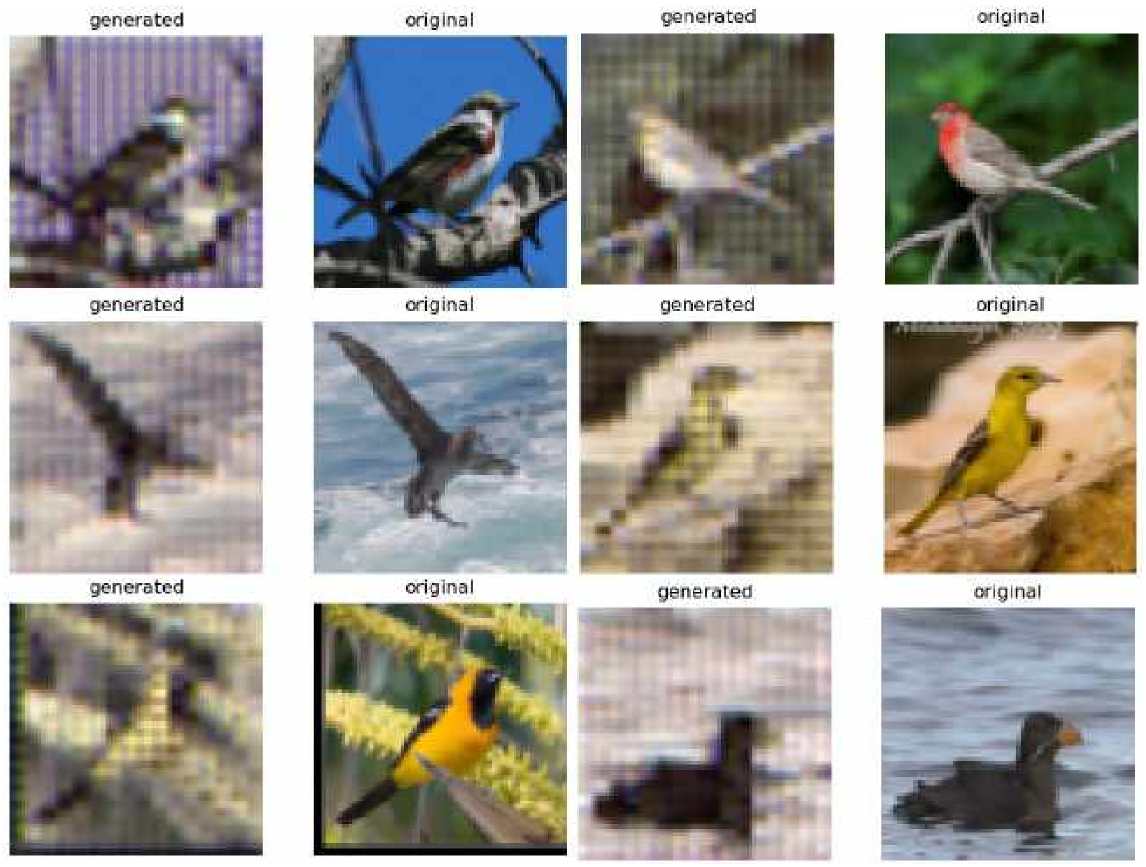


Figure 2.9: Images generated by Model 3: Com-Rec autoencoder with bottleneck dimension $3 \times 16 \times 16$

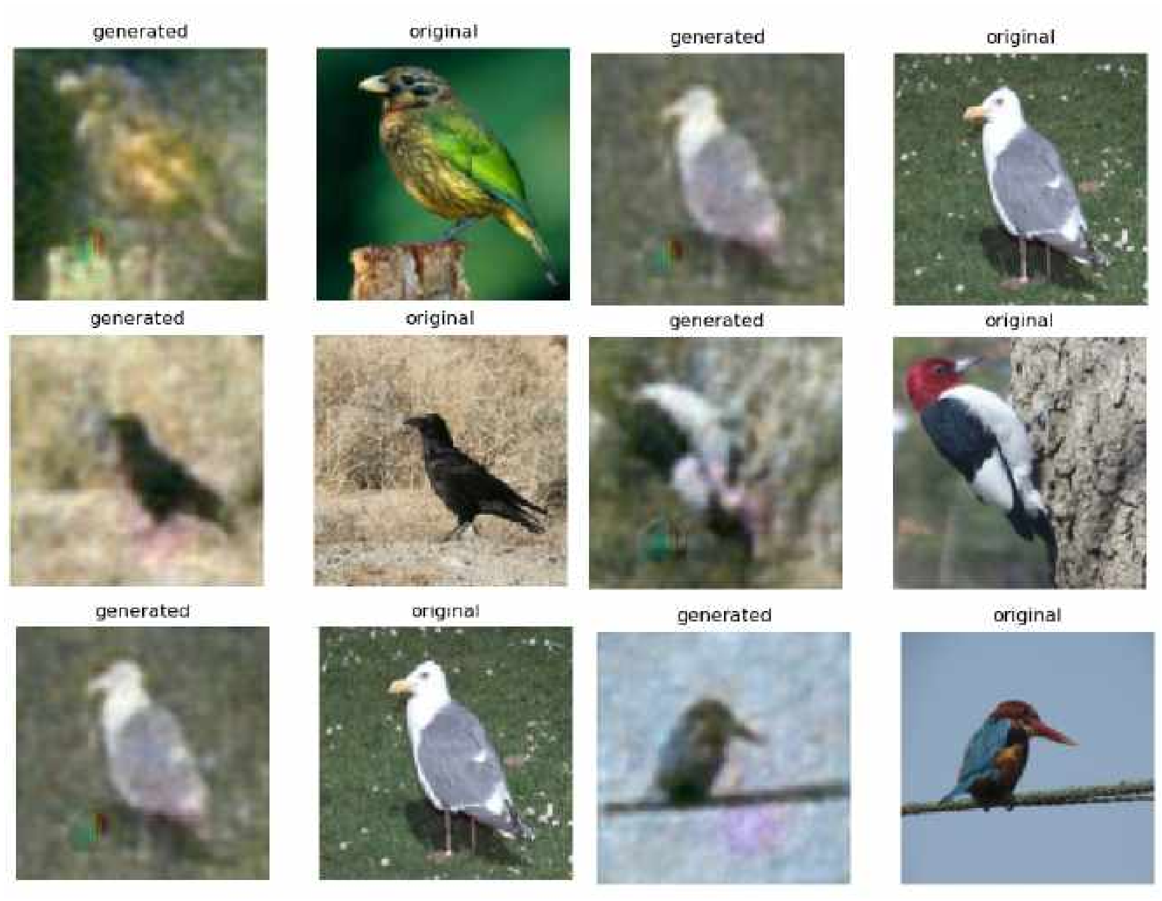


Figure 2.10: Images generated by Model 3: Com-Rec autoencoder with bottleneck 1024-dimensional vector

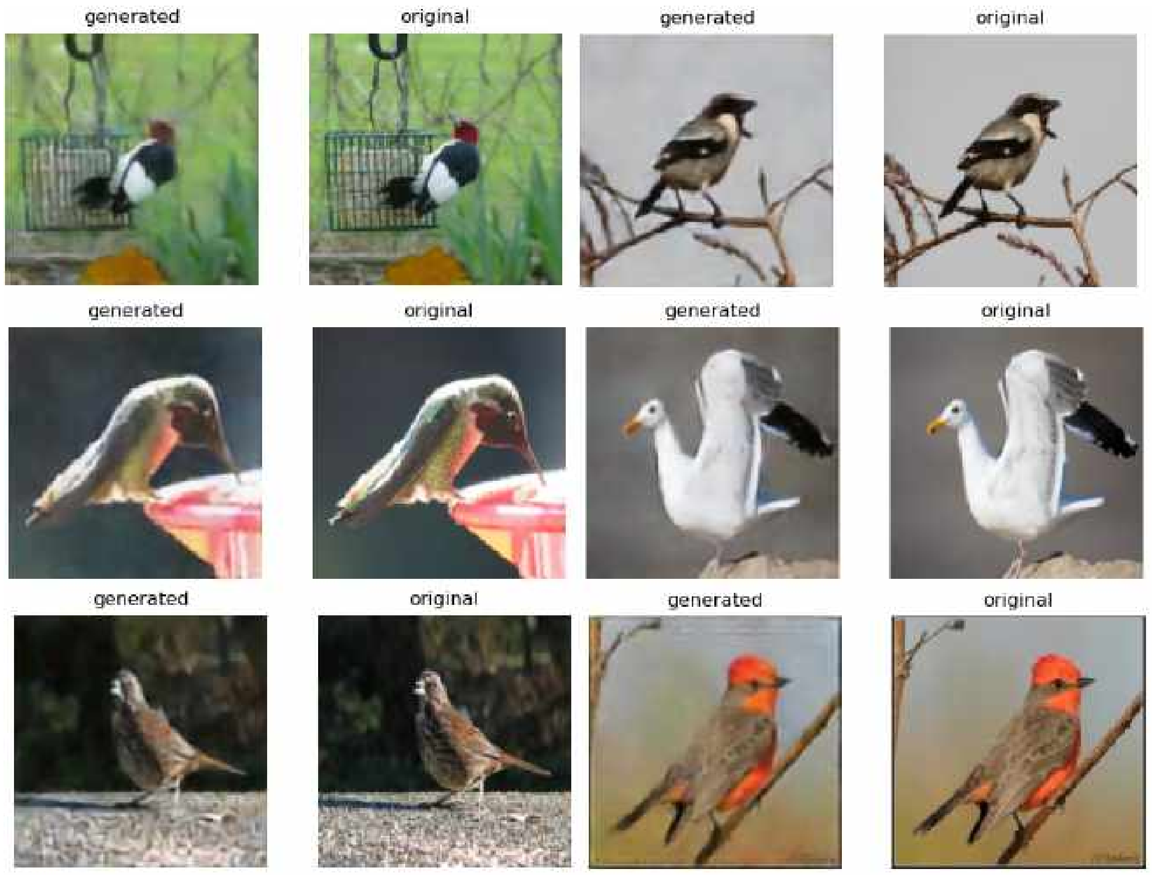


Figure 2.11: Images generated by Model 4: Compressive autoencoder with bottleneck $32 \times 32 \times 32$-dimensional vector

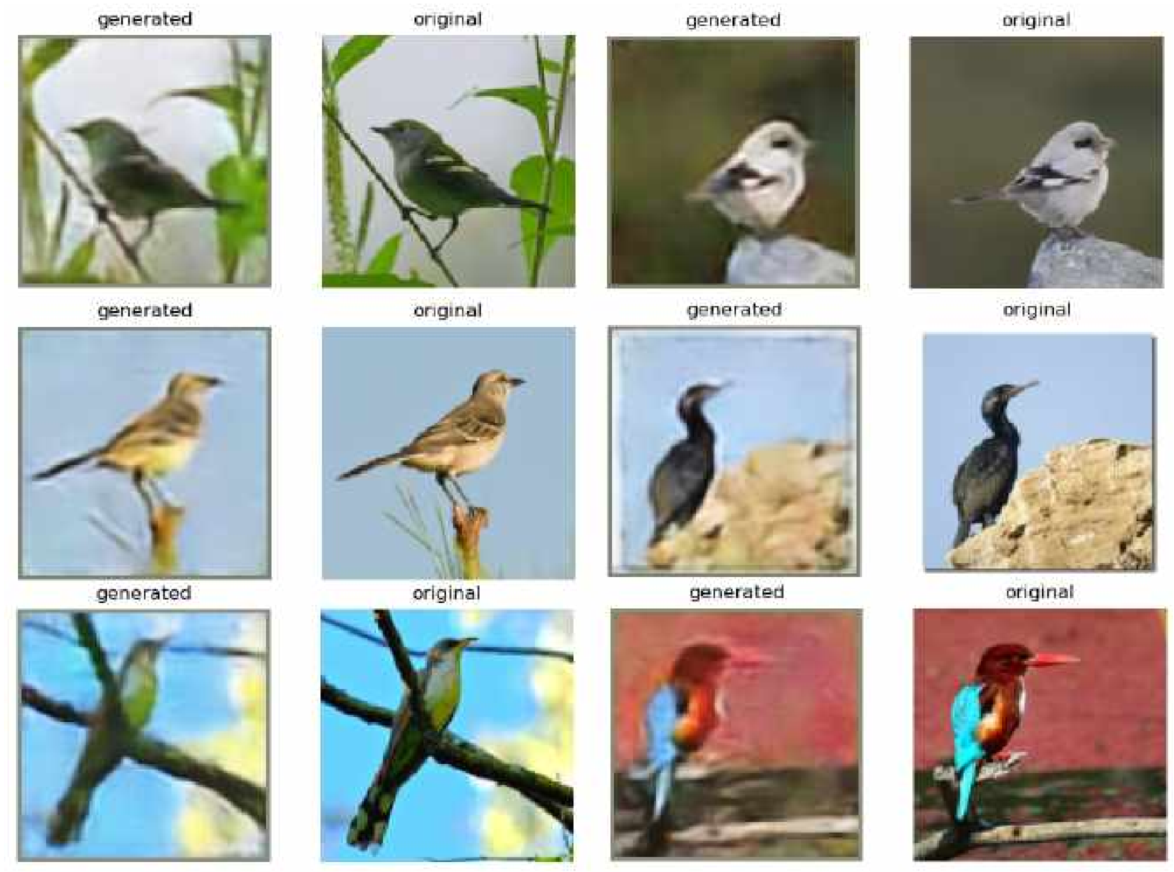


Figure 2.12: Images generated by Model 4: Compressive autoencoder with bottleneck $16 \times 16 \times 16$-dimensional vector

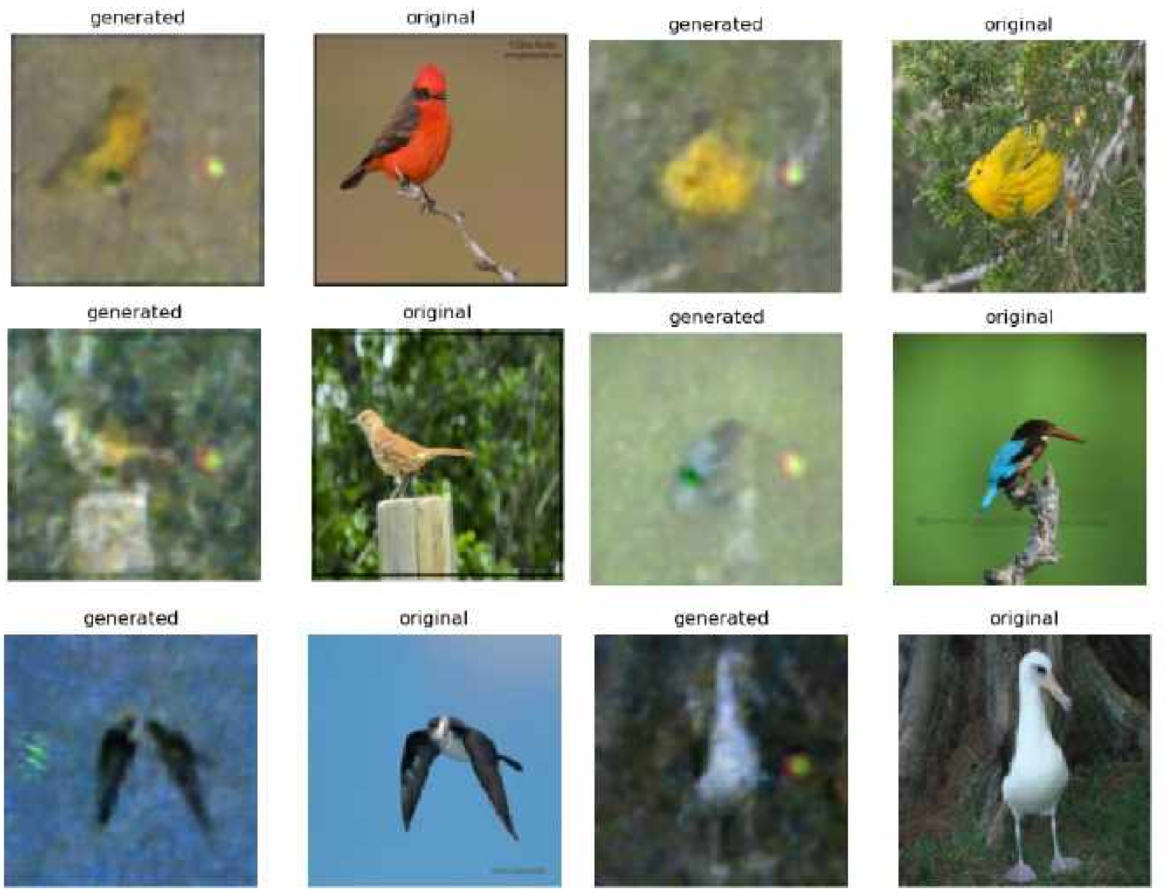


Figure 2.13: Images generated by Model 5: Com-Rec autoencoder with GAN

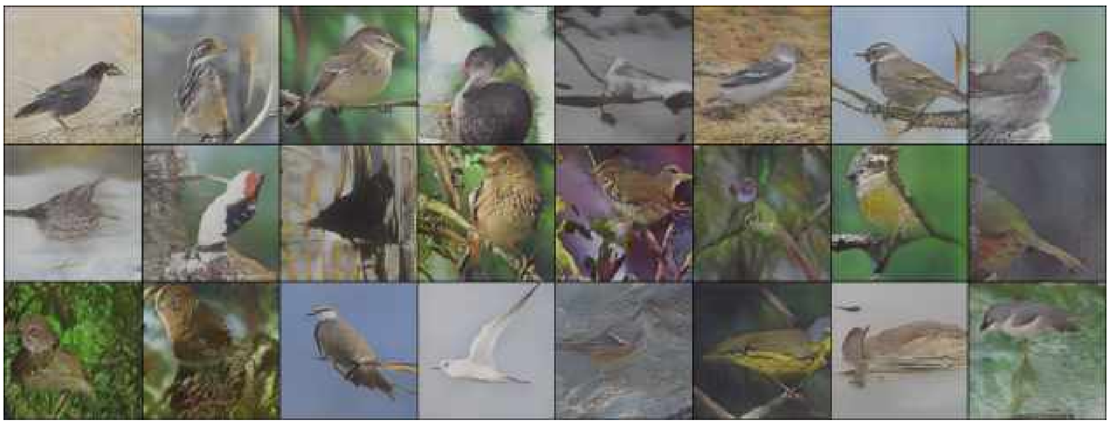


Figure 2.14: Birds images generated Images by StackGAN-v2 autoencoder

Figure 2.15: Real Images: Birds

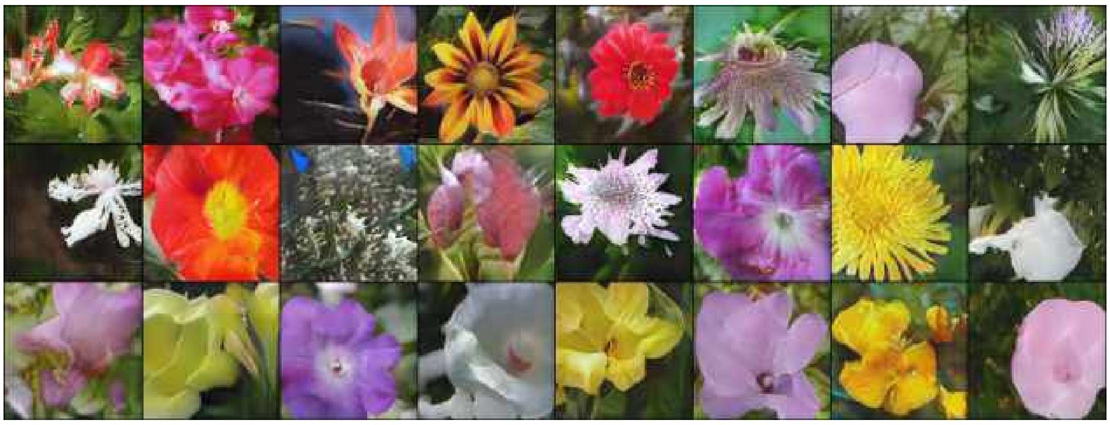


Figure 2.16: Flower images generated Images by StackGAN-v2 autoencoder

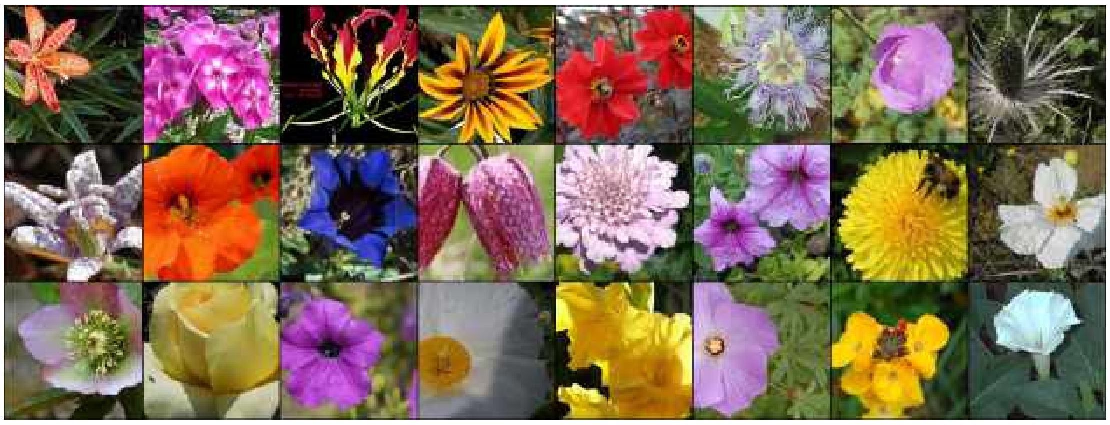


Figure 2.17: Real Images: Flowers

Dataset	**Model**	**Inception Score**	**FID**
CUB	**Resnet autoencoder**	1.08 ± 0.03	275.71
CUB	**Com-Rec autoencoder with latent dimension** $3 \times 32 \times 32$	4.66 ± 0.21	30.92
CUB	**Com-Rec autoencoder with latent dimension** $3 \times 16 \times 16$	1.28 ± 0.02	198.18
CUB	**Com-Rec autoencoder with latent dimension** 1024	1.15 ± 0.02	248.95
CUB	**Compressive autoencoder with latent dimension** $32 \times 32 \times 32$	9.32 ± 0.31	14.17
CUB	**Compressive autoencoder with latent dimension** $16 \times 16 \times 16$	4.09 ± 0.15	86.44
CUB	**Compressive autoencoder with latent dimension** 1024	1.09 ± 0.08	283.64
CUB	**Com-Rec autoencoder with GAN**	1.18 ± 0.05	236.65
CUB	**StackGAN-v2 autoencoder**	**3.66 ± 0.09**	**21.58**
Oxford-102	**StackGAN-v2 autoencoder**	**3.23 ± 0.14**	**57.96**

Table 2.3: Inception scores (IS), Fréchet Inception distance (FID) (computed for 256×256 images)

Chapter 3

Sequence-To-Sequence Text Autoencoder for Text Embedding

3.1 Introduction

The next step is to obtain a dense latent representation of a text. Sequence to sequence architectures are useful in such cases in which the model predicts a sequence of tokens or words given an input sequence. We leverage the recent advancements in machine translation and used those models for our text autoencoder setups. We have conducted our experiments on two post popular sequence-to-sequence models for our autoencoders and did an in-depth analysis of the models to select the best architecture based on our requirements for the end-to-end structure.

3.2 Research Methods

3.2.1 Convolutional sequence to sequence text autoencoder model

We first worked with CNN-based sequence to sequence model (Gehring, Auli, Grangier, Yarats, & Dauphin, 2017) which uses Convolutional Neural Networks (CNN) instead of recurrent networks and thus computations are performed in parallel. In the encoder (Figure 3.1), the input sentence $(y^{<1>}, ..., y^{<m>})$, where $y^{<t>}$ represents the t-th word in a given sentence, is first embedded into an embedding vector $(e_1, ..., e_m)$ by an embedding layer. As the model does not use any recurrent connections, a positional embedding layer provides a sense of order by embedding the position of the tokens in the input sequence $(p_1, ..., p_m)$. These two embeddings are element wise added to obtain the overall representation of the input sequence $r = (r_1, ..., r_m) = (e_1 + p_1, ..., e_m + p_m)$. This representational vector is fed into linear layer , which transforms into a hidden vector. The hidden vector then passed onto the next step i.e. N-Convolutional blocks where each block comprises of a $1D$ convolution layer, followed by non-linear activation, gated linear units (GLU). Then the output of the convolutional blocks are fed into another linear layer which transforms back to embedding dimension size from hidden dimension size, which yield convoluted vectors $c = (c_1, ..., c_m)$, one per each token of the input sequence. This convoluted vectors c are element wise added to input representation r through residual connections, yielding the combined vectors

$cm = (r_1 + c_1, ..., r_m + c_m)$.

The CNN-decoder predicts all the tokens simultaneously in parallel fashion, unlike sequential model where the tokens of the target sequence are predicted one by one. The decoder network architecture is similar to the encoder network, with some changes such as there is no residual connection between the convoluted output and input representation. The encoder convoluted output vectors c and combined output vectors cm are fed with convolutional blocks of the decoder. The output linear layer converts embedding dimension to vocabulary dimension which makes the probabilistic prediction of the next token (Figure 3.2).

3.2.2 LSTM-based sequence to sequence text autoencoder

LSTM model can be used to generate embedding of sentences (Palangi et al., 2016). The LSTM model embeds a sentence into a semantic vector by sequentially taking each word in a sentence as input and therefore extracting information from the words. As it goes through the sentence the LSTM model gathers increasingly richer information. The hidden layer of the network produces a semantic representation of the entire sentence as it reaches the last word. If $x(t)$ be the t-th word, coded as a word vector, W_h is a hashing operator, W_{rec} is the recurrent weight matrix, W is the input weight matrix, the hidden activation vector $y(t)$, can be used as a semantic representation till the t-th word and if $x(m)$ is the last word then the associated activation $y(t)$ is the semantic representation vector of the entire sentence (Figure 3.3). If $f(.)$ is $\mathtt{tanh}(.)$ activation and b is the bias vector, then the LSTM model can be mathematically formulated as

$$l(t) = W_h x(t) \tag{3.1}$$

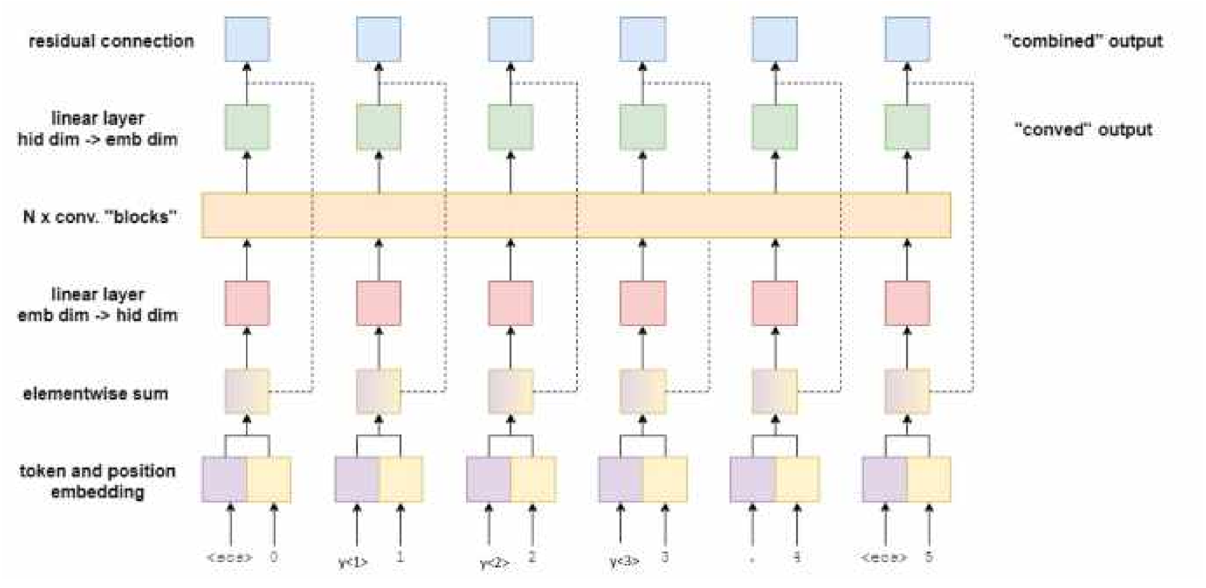


Figure 3.1: Encoder: Convolutional sequence to sequence text autoencoder

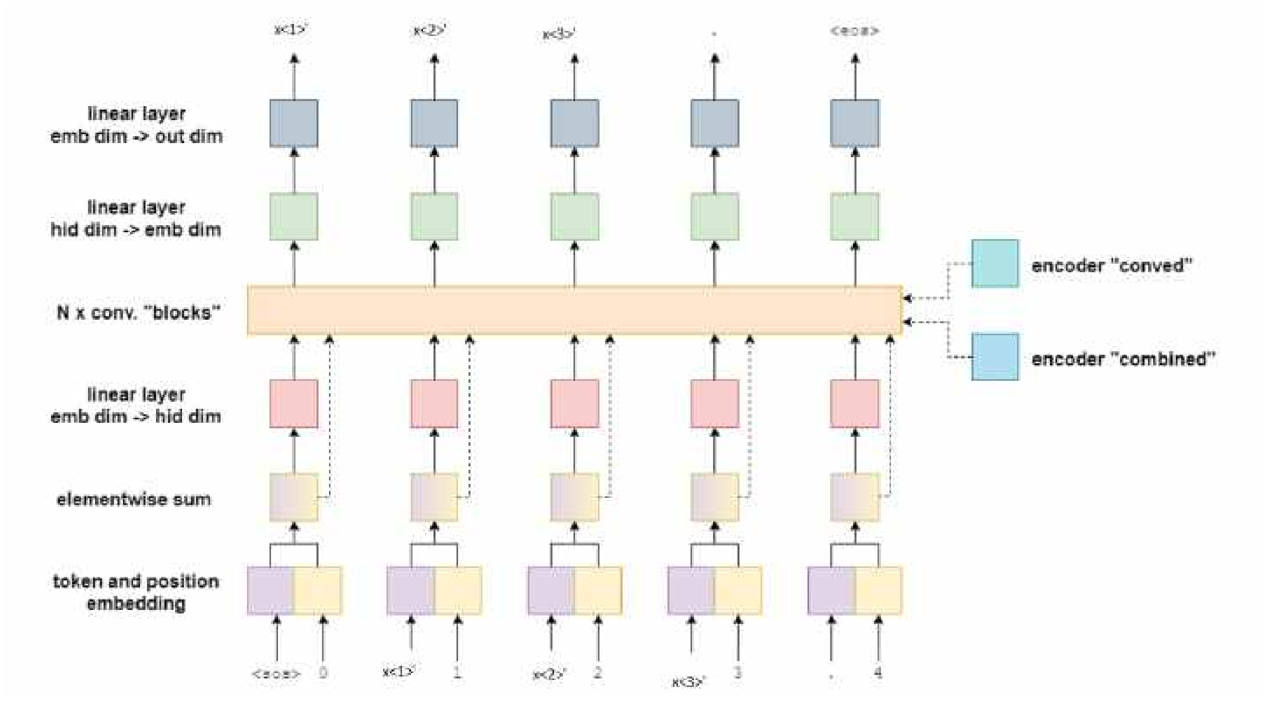

x<1>'
x<2>'
x<3>'
.
<eos>
linear layer emb dim -> out dim
linear layer hid dim -> emb dim
N x conv. "blocks"
encoder "conved"
encoder "combined"
linear layer emb dim -> hid dim
elementwise sum
token and position embedding
<sos> 0
x<1>' 1
x<2>' 2
x<3>' 3
. 4

Figure 3.2: Decoder: Convolutional sequence to sequence text autoencoder

$$y(t) = f(Wl(t) + W_{rec}y(t-1) + b) \tag{3.2}$$

LSTM can be used as a language model that predicts next word in a sentence. Formally, given a training sentence $(y^{<1>}..., y^{<T>})$, where $y^{<t>}$ represents the t-th word in a given sentence, the LSTM obtains a sequence of estimated distributions $P(y^{<t+1>}|y^{<1>}....., y^{<t>}) = softmax(r^{t+1})$ using the sequence of its output vector $(r^1, r^2....., r^T)$. The LSTM learns the probability distribution of the word sequence in sentences by maximizing log probability of training sentence, given by $\sum_{t=1}^{T}[\log P(y^{<t+1>}|y^{<1>}....., y^{<t>})]$ (Figure 3.3).

We have used single layer LSTM encoder and decoder in our text autoencoder as sequence-to-sequence model (Sutskever, Vinyals, & Le, 2014). The encoder is bidirectional LSTM with hidden dimension 50, while decoder is unidirectional with hidden size 100. The text encoder takes the word embedding vectors at each time step and generate corresponding hidden vectors. In this paper we have considered all the hidden vectors to obtain the sentence embedding by maxpooling over all the hidden vectors i.e. if there are m hidden vectors corresponding to m words in a given sentence, where each hidden vector is of dimension n, then the i-th component of resulting sentence vector is computed by taking maximum at position i over all m vectors, similarly all n components of the sentence vector are computed from m hidden vectors, yielding n-dimensional sentence vector. This latent vector is used to initialize the decoder LSTM network which regenerates the text description at the output (Figure 3.4).

3.2.3 Dataset

We have used standard Caltech-UCSD Birds-200-2011 Dataset captions (Wah et al., 2011) and Oxford-102 flower dataset captions for our text autoencoders (Table 3.1).

Dataset	**Number of Training Samples**	**Number of Test Samples**
CUB-200-2011	88,540	29,330
Oxford-102	70,340	11,550

Table 3.1: Dataset Statistics: Number of Text samples in Train and Test sets

3.2.4 Experimental Setup

We conducted this experiments in python environment with the following python libraries : **NLTK, Torch, Spacy, Numpy, Torchvision**.

Hyperparameter

For Convolutional text autoencoder, we set same values of the hyper-parameters for both encoder and decoder. The embedding dimension is set to 256 and hidden dimension

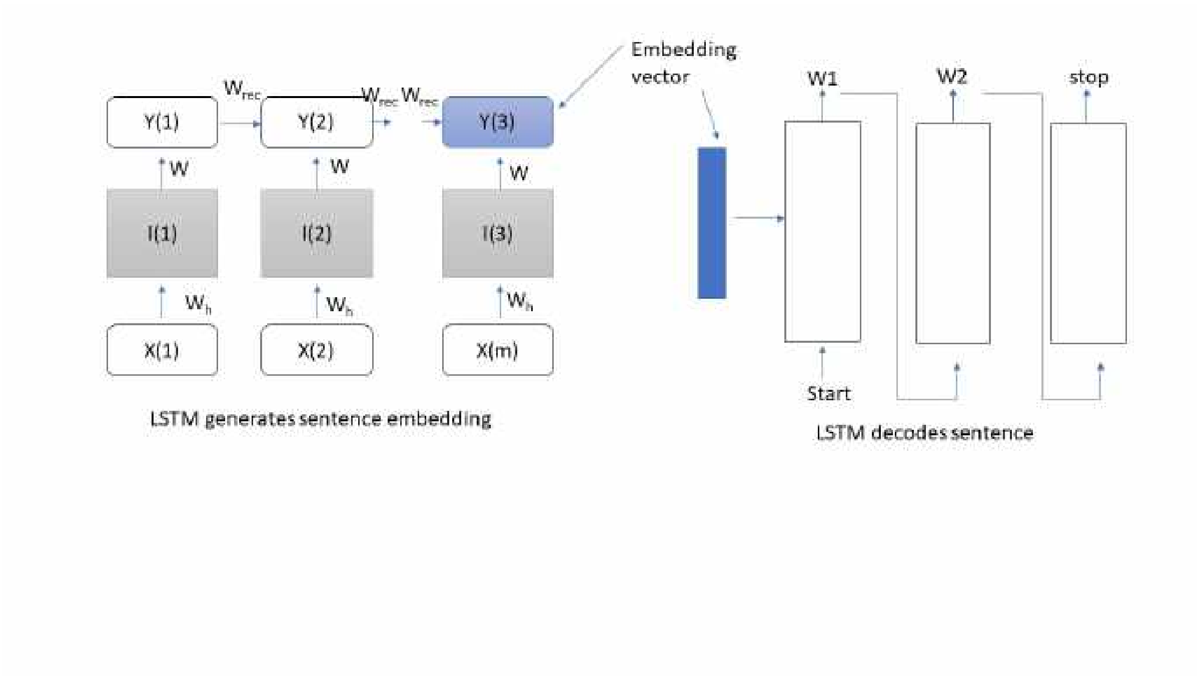


Figure 3.3: Architecture: LSTM generates sentence embedding and LSTM decodes embedding to generate sentence

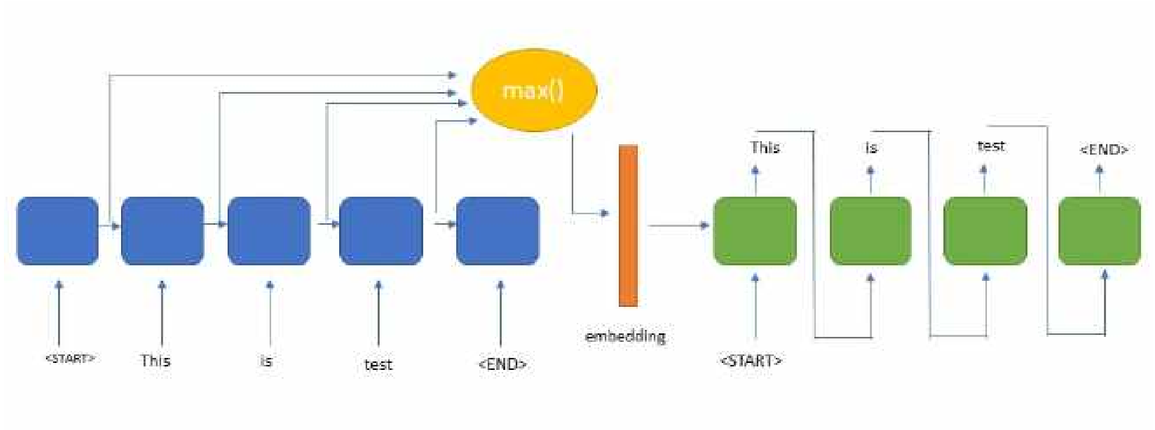


Figure 3.4: Architecture LSTM-based sequence to sequence text autoencoder. The encoder encodes the sentences into an embedding; the decoder reconstructs the original sentence from sentence embedding

is fixed to 512. Number of $1D$ convolutional blocks are set to 10 in both encoder and decoder. Kernel size and dropout are set to 3 and 0.25 respectively.

In LSTM-based text autoencoder the encoder is a bidirectional LSTM and the decoder is unidirectional LSTM. The hidden dimension for the encoder is 50 and for decoder it is 100, embedding dimension in encoder is set to 100. Adam optimizer with learning rate set to 10^{-4} is used in both the models.

Evaluation Metric

We have used four standard evaluation metrics to evaluate the performance of our text autoencoders. These metrics are BLEU (Papineni, Roukos, Ward, & Zhu, 2002), METEOR (Banerjee & Lavie, 2005), CIDEr (Vedantam, Lawrence Zitnick, & Parikh, 2015), ROUGE_L (Lin, 2004) to quantify the similarity between the ground reference sentence and autoencoder generated sentence. The official MSCOCO caption evaluation scripts[1] are used for evaluation.

3.2.5 Experiments

First, we train our Convolutional text autoencoder (Section 3.2.1) on both CUB caption dataset and Oxford-102 caption dataset (Table 3.1) separately. The embeddings are laerned during training. We did not use any pre-training as the accuracy for the convolutional model was already very high.

Next, we pre-train our LSTM-based text autoencoder (Section 3.2.2) on 1 million sentences extracted from One Billion Word Benchmark dataset (Chelba et al., 2013). For initialization of our text autoenecoder model we used top 50k 100-dimensional

[1]https://github.com/tylin/coco-caption

Dataset	**Model**	**BLEU-1,2,3,4**	**METEOR**	**ROUGE-L**	**CIDEr**
CUB-200-2011	CNN-based text autoencoder	**99.64, 99.51, 99.38, 99.25**	**79.45**	**99.78**	**9.88**
CUB-200-2011	LSTM-based text autoencoder	87.52, 81.49, 76.65, 72.55	49.89	89.04	7.04
Oxford-102	CNN-based text autoencoder	**99.65, 99.49, 99.33, 99.17**	**78.33**	**99.71**	**9.87**
Oxford-102	LSTM-based text autoencoder	85.35, 79.17, 74.26, 70.61	48.06	86.69	6.59

Table 3.2: BLEU-1,2,3,4, METEOR, ROUGE-L, CIDEr metrics are reported for both the datasets. All the values except CIDEr are shown in percentages.

Glove embeddings (Pennington, Socher, & Manning, 2014) along with Glove embeddings of the other unique words in the caption datasets (Table 3.1). The model was pre-trained for 50 epochs; this pre-trained model worked as initializer; the model was then trained on all the captions of CUB train set and Oxford-102 dataset separately (Refer to Table 3.1). We have used Cross Entropy between the generated and target sentences as loss function.

3.2.6 Evaluation Results and Analysis

BLEU-1,2,3,4, METEOR, ROUGE-L, CIDEr scores evaluated on the datasets for both of the models are reported on Table 3.2. The Convolutional model outperforms the LSTM-based model in all aspects, also convolutional model is faster to train as it performs computations in parallel. However, the encoder model in Convolutional text autoencoder generates two outputs: one is convoluted output and other one is combined output (Section 3.2.1). Each of these outputs are sequence of vectors i.e. if there are m tokens in input sequence, the encoder generates m convoluted vectors and m combined vectors yielding total $2m$ vectors for any sequence of length m, while our LSTM-based encoder embeds to one latent vector per sequence. Considering the capacity and ability to generate dense latent representations and performances of both the autoencoders, the LSTM-based autoencoder is still a better choice for text autoencoder in our end-to-end network.

3.3 Conclusion and Future Work

In this chapter we we discussed different approaches to obtain the latent representation of text data. We have built two deep learning autoencoder models for text; one is based on LSTM and another is based on CNN. We verified that CNN-based model outperforms the LSTM-based model; however in LSTM based model the latent representation is a dense vector while in CNN- based model the latent representations are group of vectors. As we need dense embedding vectors for our end-to-end setups, we select LSTM-based autoencoder for generating the sentence embedding that is to be used for our end-to-end synthesis models.

We are planning on to utilize the recent state-of-the-art language VAE model **Optimus**(C. Li et al., 2020) which generates smooth latent space and showed promising results as part of our future work to obtain text latent representation.

Chapter 4

End-To-end Image-to-text and Text-to-image Synthesis Networks

4.1 Introduction

We have the latent vector representation for both images and text from the models discussed in previous chapters. The next logical step is to design a suitable **Mapping Network** that learns the probability distributions between the two latent spaces and transforms image embedding into text embedding and vice versa. These two trained autoencoders along with the mapping networks are the building blocks for the end-to-end **Image-to-text** and **Text-to-image** synthesis networks.

4.2 Research Methods

The **Embedding Mapping Networks** are generative networks as the networks is supposed to learn the mapping between probability distributions. To achieve this we have used to different approaches; one employs GAN to learn the distributions and the second one uses a statistic Maximum Mean Discrepancy (Tolstikhin et al., 2016).

4.2.1 GAN-based Cross-modal Mapping Network

In this architecture simple GAN models are used for both image-to-text and text-to-image. The generator translates one modality embedding into the other modality embedding and the discriminator determines whether two embedding distributions matches or not (Figure 4.1).

We have two autoencoders of two different modalities; lets call these autoencoders as autoencoder A (e.g. Image autoencoder) and autoencoder B (e.g. Text autoencoder). The encoder of autoencoder A generates the latent representation z from input data I and the encoder of autoencoder B generates the latent representation $x \sim P_X$ of the other modality input data T. The generator $G_\theta(.)$ (A-to-B Embedding Mapping Network), takes $z \sim P_Z$ as input and learns the distribution $\tilde{x} = G_\theta(z) \sim P_\theta$, such that $P_\theta \approx P_X$. The discriminator try to estimate the distance between P_θ and P_X during the training of GAN by assigning a probability $y = D(x) \in [0, 1]$ to the latent variable, if x is actual training input and and probability $1 - y$, if $\tilde{x}$ is generated by the model i.e,

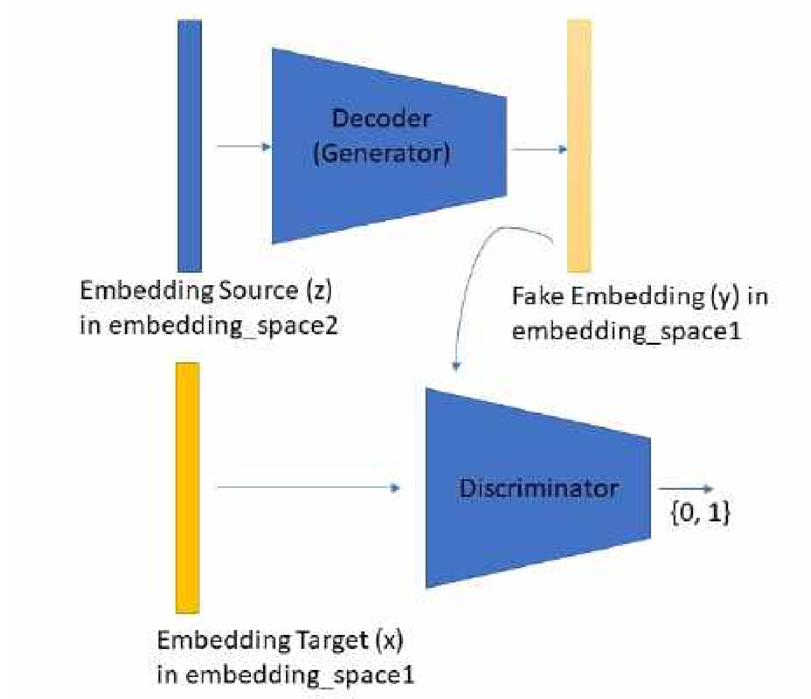


Figure 4.1: Architecture : GAN based cross-modal embedding mapping network. For each modality, we have designed separate models in which generator takes one modality embedding and produces other modality embedding.

$D(\tilde{x}) = D(G(z))$. So the discriminator tries to maximize the binary cross entropy loss function:

$$loss_{discriminator} = \log(D(x)) + \log(1 - D(G(z))) \tag{4.1}$$

While the generator tries to minimize the loss:

$$loss_{generator} = \log(1 - D(G(z))) \tag{4.2}$$

Likewise, B-to-A Embedding Mapping Network is trained to learn the other modality embedding.

4.2.2 MMD GAN-based Cross-Modal Embedding Space Mapping Networks

Maximum Mean Discrepancy is a distance measure on the space of embedding probabilities in Reproducing Kernel Hilbert Space (RKHS) (Berlinet & Thomas-Agnan, 2011). Given two probability distributions P and Q and a continuous positive definite real-valued kernel k ($\mathcal{H}$ to be corresponding RKHS); the corresponding kernel means be defined as $\mu_p = \int k(.,x)dP(x)$ and $\mu_q = \int k(.,y)dQ(y)$, the distance $MMD(P,Q) = \|\mu_p - \mu_q\|$ measures the similarity between the two distributions, is known as Maximum Mean Discrepancy (MMD) (Gretton, Borgwardt, Rasch, Schölkopf, & Smola, 2012). In this paper we propose a mapping network which is based on MMD-GAN (C.-L. Li et al., 2017) that trains the generator g_θ to minimize MMD distance between two distributions i.e. $\min_\theta M_k(\mathbb{P}_\mathcal{X}, \mathbb{P}_\theta)$, hence passes the hypothesis test.

where real data $x \sim \mathbb{P}_\mathcal{X}$ and generator distribution $g_\theta(z) \sim \mathbb{P}_\theta$ and $\mathbb{P}_\mathcal{Z}$ is the base distribution such that $z \sim \mathbb{P}_\mathcal{Z}$ and M_k is square of MMD distance :

$$M_k(\mathbb{P}_\mathcal{X}, \mathbb{P}_\theta) = \|\mu_{\mathbb{P}_\mathcal{X}} - \mu_{\mathbb{P}_\theta}\|^2 = \mathbb{E}_{\mathbb{P}_\mathcal{X}}\{k(x,x')\} + \mathbb{E}_{\mathbb{P}_\theta}\{k(g_\theta(z), g_\theta(z'))\} - 2\mathbb{E}_{\mathbb{P}_\mathcal{X},\mathbb{P}_\theta}\{k(x, g_\theta(z))\} \tag{4.3}$$

Ideally for two equal distributions, $P = Q$, $M_k(P,Q) = 0$, however $M_k(P,Q)$ may not be equal to zero due to sampling variance even when the null hypothesis ($P = Q$) is satisfied. We reject the null hypothesis which indicates P and Q are not equal when $M_k(P,Q)$ is greater than some chosen minimum threshold th_α, such that $th_\alpha > 0$, otherwise we say $P = Q$. So it is necessary that $M_k(P,Q)$ should be high and greater than minimum threshold when two distributions are dissimilar. So instead of having a fixed kernel such as Gaussian kernel $k(x,x') = exp(\|x - x'\|)^2$, we adversarially learn the kernel such that $\max_{k\in\mathcal{K}} M_k(\mathbb{P}_\mathcal{X}, \mathbb{P}_\theta)$, which implies kernel producing high value when two distributions are not identical. So we need to optimize :

$$\min_\theta \max_{k\in\mathcal{K}} M_k(\mathbb{P}_\mathcal{X}, \mathbb{P}_\theta) \tag{4.4}$$

As mentioned by the authors in the papers (Gretton, Borgwardt, et al., 2012) (Gretton, Sejdinovic, et al., 2012) if k is a characteristic and f is an injective function, then the kernel $k_f(x,x') = k(f(x), f(x'))$ is also a characteristic. So for a set of parameterized injective function f_ϕ, where ϕ denotes the parameter, the objective function mentioned in Equation 4.4 changes to:

$$\min_\theta \max_\phi M_{k_{f_\phi}}(\mathbb{P}_\mathcal{X}, \mathbb{P}_\theta) \tag{4.5}$$

Thus we combine the injective function with Gaussian kernel to get parameterized kernel, $k_{f_\phi}(x,x') = exp(\|f_\phi(x) - f_\phi(x')\|)^2$, which we have used in this paper. For generator g_θ, we have used feed forward neural network which takes the embedding vectors of one modality generated from the encoder of either autoencoders (text autoencoder or image autoencoder), and generates the embedding vector of the other modality (Figure 4.2). Another requirement that needs to be satisfied is f_ϕ to be an injective function. For any injective function there exists an inverse function f^{-1} such that $f^{-1}(f(x)) = x$. An autoencoder is used to match this criteria, where encoder of the autoencoder approximates the injective function requirement. So the discriminator in MMD-GAN comprises of an encoder f_{ϕ_e} and a decoder f_{ϕ_d} (Figure 4.2), where $f_{\phi_d} \approx f_{\phi_e}^{-1}$ and ϕ_e, ϕ_d are parameters of encoder and decoder of discriminator network, respectively. Taking account of these requirements the final objective function for training MMD-GAN is given by:

$$\min_{\theta} \max_{\phi} M_{k_{f_{\phi_e}}}(\mathbb{P}(\mathcal{X}), \mathbb{P}(g_\theta(\mathcal{Z}))) - \lambda \mathbb{E}_{y \in \mathcal{X} \cup g_\theta(\mathcal{Z})} \|y - f_{\phi_d}(f_{\phi_e}(y))\|^2 \tag{4.6}$$

4.2.3 Dataset

We have used Caltech-USCD-Birds-200-2011 dataset which comprises of 11,788 images of 200 bird species and Oxford-102 Flower dataset which contains 8,189 images of 102 categories of flowers. Each image from either of the datasets has multiple annotations per image. We split the dataset into train set (8,855 images, 88540 captions, 150 classes) and test set (2,933 images, 29,330 captions, 50 classes) for CUB-200-2011

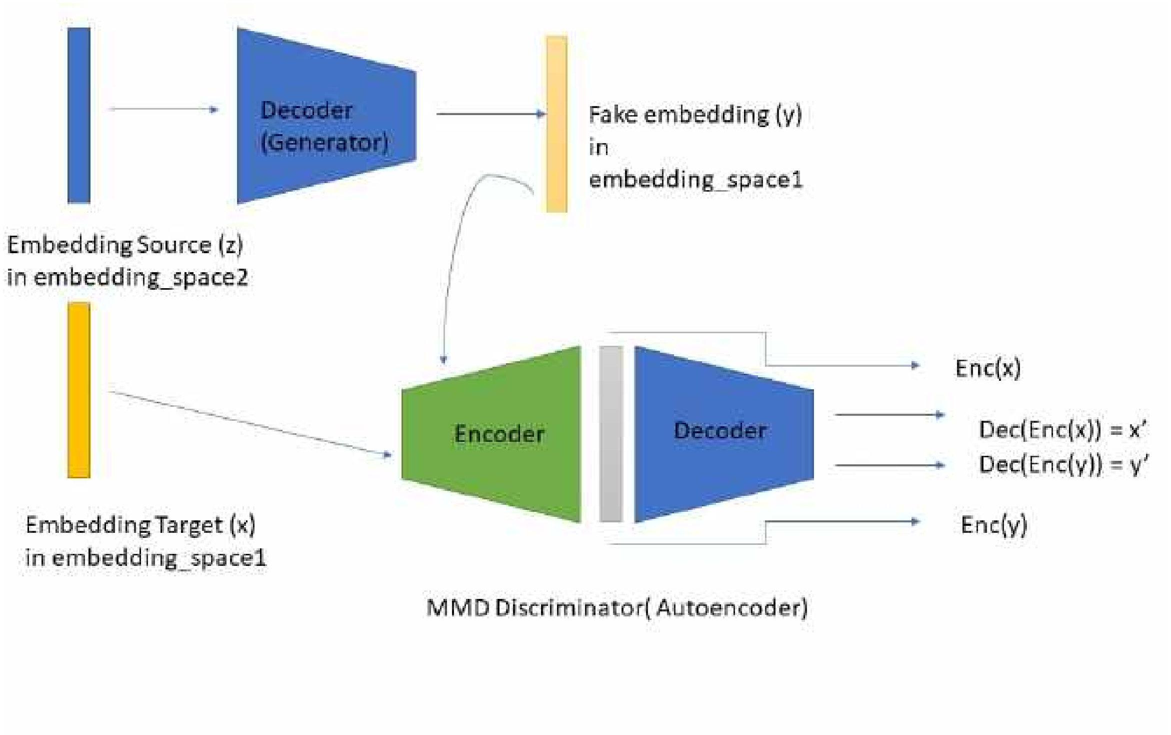

Decoder (Generator)
Fake embedding (y) in embedding_space1
Embedding Source (z) in embedding_space2
Encoder
Decoder
Enc(x)
Dec(Enc(x)) = x'
Dec(Enc(y)) = y'
Enc(y)
Embedding Target (x) in embedding_space1
MMD Discriminator(Autoencoder)

Figure 4.2: Architecture : MMD-GAN based cross-modal embedding mapping network. For each modality, we have designed separate models in which generator takes one modality embedding and produces other modality embedding. The discriminator is an autoencoder

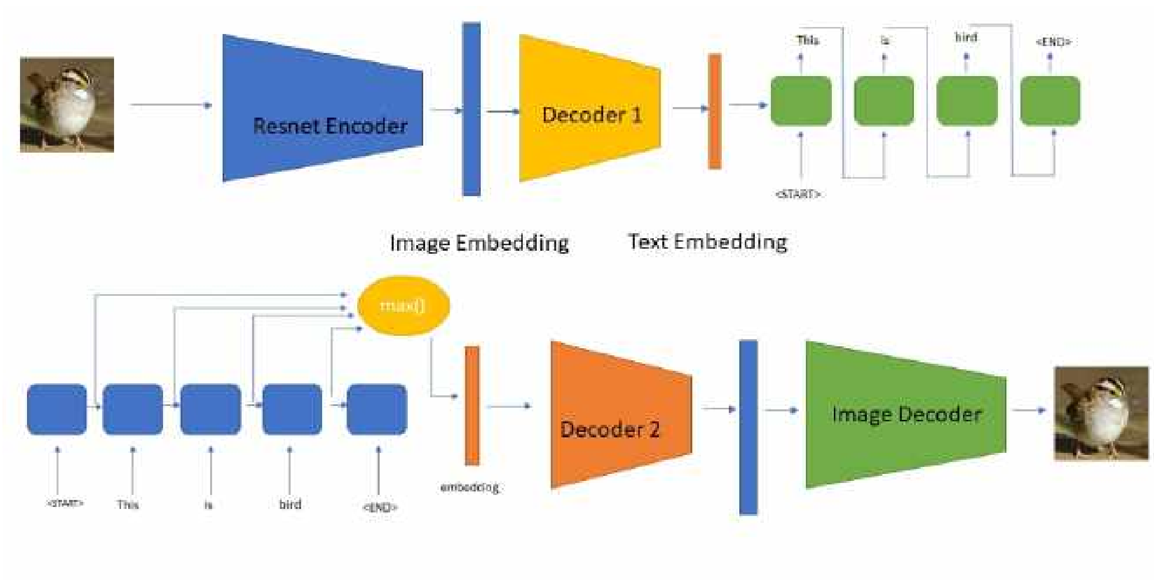


Figure 4.3: End-to-End Networks: Self-supervised Image-to-Text generator and Self-supervised Text-to-Image generator

dataset and train set (7,034 images, 70,340 captions, 82 classes) and test set (1,155 images, 11,550 captions, 20 classes) for Oxford-102 such that the respective classes do not overlap.

4.2.4 Experiment

After we trained both our autoencoders, the GAN and MMD-based Mapping Networks (Section 4.2.1 and 4.2.2) are trained separately for both image-to-text and text-to-image cases (Figure 4.1 and 4.2). The weights of the trained autoencoders are kept fixed while we train the mapping networks.In case of training image-to-text embedding mapping network, first the image encoder encodes the input image into image embedding, which is then used as input to image-to-text embedding mapping network, that generates corresponding text embedding, the batch of generated text embeddings are used as fake embeddings while the text embedding obtained from text encoder acts as gold embeddings; fake embeddings and true embeddings are then used to train the generative networks as discussed in Section 4.2.1 and 4.2.2). Likewise, text-to-image setup is trained.

After all the networks are trained components are assembled to build the end-to-end architecture. The image-to-text system composed of image encoder, image-to-text embedding mapping network (GAN-based or MMD-based), text decoder while the text-to-image system comprises of text encoder, text-to-image embedding mapping network (GAN-based or MMD-based) and image decoder as shown in Figure 4.3.

Evaluation Metric

As we are generating image-to-text and text-to-image in an unsupervised way, we expect our systems to map the semantic information between the two modalities, rather than word matching. So we developed four evaluation metrics to assess the performance of our end-to-end Image-To-text and Text-To-image synthesis systems:

1. ***Human Score:*** We conducted user studies for evaluation of our end-to-end text-to-image and image-to-text systems. We sampled 2000 samples for each of text-to-image and image-to-text cases, assigned 5 users to score the results. Generated samples are evaluated on a 4.0 scale. 1.0 being the lowest (worst case) and 4.0 being the highest (best case). Point 4.0 is awarded when text description matches completely the image without any errors. Point 3.0 is awarded when text description partially matches the image with minor errors. Point 2.0 is awarded when there is somewhat matches between text and image. Point 1.0 is awarded when the image and text are unrelated.

2. ***Class Accuracy:*** The CUB-200-2011 dataset has 150 train classes and and 50 test classes and the Oxford-102 dataset has 82 and 20 train and test classes respectively. Classes define different species for birds and different categories for flowers. Birds in the same class are from same species, have identical features and description; flowers in same class are from same category having similar features, texture, similar color and descriptions. It is logical to expect that when one modality embedding maps into the other modality embedding, they are mapped to the same class. We first obtain the embeddings of one modality (say modality **A**) call it **true embeddings in modality A**. Then we obtain the embeddings of other modality (say modality **B**) and use it to feed into the **"Modality B-To-Modality A" mapping network** to obtain the **fake embeddings in modality A**. Then we compute the cosine similarity between the true embeddings and fake embeddings and do an $argmax$ to determine the class of fake embedding based on the highest cosine similarity scores. Then we calculate the class accuracy based on class labels of the true embeddings and predicted class labels of the fake embeddings.

3. ***t-SNE Clustering Visualization:*** We use t-SNE algorithm (Maaten & Hinton, 2008) to visualize the embedding space mapping. As discussed in the paragraph 2, we have now the actual class label of the true embedding and predicted class label of fake embedding from cosine similarity score. The actual class label of the true embedding and predicted class label of the fake embedding are labelled as **o-ClassNumber** and **f-ClassNumber** respectively i.e if actual class number is e.g. 10, then it is tagged as **o-10** and if the predicted class number from fake

embedding is e.g. 12, then it is tagged as **f-12**. Then we apply PCA algorithm (Wold, Esbensen, & Geladi, 1987) to both real and fake embeddings to reduce the dimension to 50 along the axis of maximum variance, then we apply t-SNE algorithm on resultant embeddings to map it into a $2D$ plane. If the mapping networks learns the semantic correlation between the two modalities, then the center of the embeddings should be closer to each other e.g. the center of **o-10** should lie closer to **f-10** in ideal situation compared to any other class centers.

4. ***Recall at K:*** We have set of images $I_1, ..., I_m$ and set of texts $T_1, ..., Tm$, where I_j caption corresponds to text T_j. For text-to-image system we obtain the real embeddings $i_1, ..., i_m$ of the images from image encoder, then we obtain the embeddings of the text $t_1, ..., t_m$ from text encoder, these are then fed to text-to-image mapping networks to obtain fake image embeddings $i_{f_1}, ..., i_{f_m}$. Then we compute the cosine similarity between real embedding i_p and fake embedding i_{f_q} i.e $Cosine\_sim(i_p, i_{f_q})$ for q = 1 to m . We rank (i_p, i_{f_q}) combination based on strength of cosine similarity scores. Likewise all (i_p, i_{f_q}) combinations are ranked for all values of p =1 to m and q =1 to m. Recall@K determines the percentage of the combinations (i_p, i_{f_q}) where $p = q$ and $Rank(i_p, i_{f_q}) < K$. Similarly Recall@K can be calculated for Image-to-text case. To compute Recall@K at class level, we compute the percentage of combinations (i_p, i_{f_q}) where $Class(p) = Class(q)$ and $\min(Rank(i_p, i_{f_q})) < K$. So the Recall@1 at class level is clearly the class accuracy (Refer 2).

4.3 Result and Analysis

The sample outputs of the end-to-end Image-to-text synthesis for GAN-based mapping network are shown in Figure 4.13 and 4.15; Figure 4.12 and 4.14 depict the sample outputs for GAN-based Text-to-image synthesis network. Figure 4.17 and 4.19 are sample outputs for MMD-based Image-to-text synthesis network; MMD-based Text-to-image output samples are shown in Figure 4.16 and 4.18. Evaluation metric scores (Refer Evaluation Metric) are reported in Table 4.1 for GAN-based end-to-end networks and in Table 4.2 for MMD-based networks.

The human score indicates that there exists some correlation between the two modalities of Image-to-text and Text-to-image synthesis systems. However **Recall@K** score at individual caption and image level and also at class level are indicative of low semantic correlation between the two modalities for both GAN based and MMD based Image-to-text and Text-to-image systems. The scores at Table 4.1 and Table 4.2 imply the GAN-based system performs better than MMD-based system to some degree, specifically in case of Image-to-text synthesis. The t-SNE plots of the semantic

Metric	CUB-200-2011		Oxford-102	
	Image-to-text	Text-to-image	Image-to-text	Text-to-image
Human Score	**2.3**	**2.5**	**2.5**	**2.4**
Class Accuracy*	2.3	2.8	6.6	4.5
Recall@1*	0.05	0.1	0.1	0.12
Recall@5*	0.1	0.3	0.6	0.4
Recall@10*	0.2	0.6	1.1	0.8
Class Recall@1*	2.3	2.8	6.6	4.5
Class Recall@2*	4.5	4.9	11.8	11.0
Class Recall@3*	6.3	7.1	17.6	16.5
Class Recall@5*	10.5	11.7	28.0	25.0

Table 4.1: Human Score, Recall@K, Class Recall@K of GAN-based Image-To-Text and Text-To-Image End-To-End system on CUB and Oxford-102. (* means the values reported are in percentage).

Metric	CUB-200-2011		Oxford-102	
	Image-To-Text	Text-To-Image	Image-To-Text	Text-To-Image
Human Score	2.1	2.4	2.2	2.3
Class Accuracy*	2.0	2.5	5.6	4.1
Recall@1*	0.05	0.06	0.2	0.1
Recall@5*	0.1	0.2	0.6	0.3
Recall@10*	0.2	0.3	1.0	0.6
Class Recall@1*	2.0	2.5	5.6	4.1
Class Recall@2*	4.4	4.9	10.9	7.4
Class Recall@3*	6.1	7.4	15.9	10.8
Class Recall@5*	10.2	12.0	26.4	22.1

Table 4.2: Human Score, Recall@K, Class Recall@K of MMD-based Image-To-Text and Text-To-Image End-To-End system on CUB and Oxford-102. (* means the values reported are in percentage).

spaces (Refer 3) provided in Figures 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 4.10, 4.11 show how the real embeddings and mapping network generated embeddings lie in the latent space. The visual interpretation of t-SNE plots exhibit, though the mapping networks learn the transformation from one modality to another, these embeddings are not correctly aligned in semantic context i.e original embedding (e.g. center **o-18**) and the generated embedding (e.g. **f-18**) are not closest to each other (Refer 3), rather they are fairly distant on average.

To further investigate and ascertain whether the mapping networks are learning the semantic meaning across modalities, we generated images from Text-to-image networks by doing minor perturbation (one word change) in textual description and observed the impact on generated images (e.g. replace **"red"** with **"white"** or **"long"** with **"short"**). The observations are reported in Table 4.3 and Table 4.4 which understandably suggests mapping networks do not learn the semantic alignments of the words and image features, i.e word meaning of **"red"** does not represent visual meaning of **"red"**.

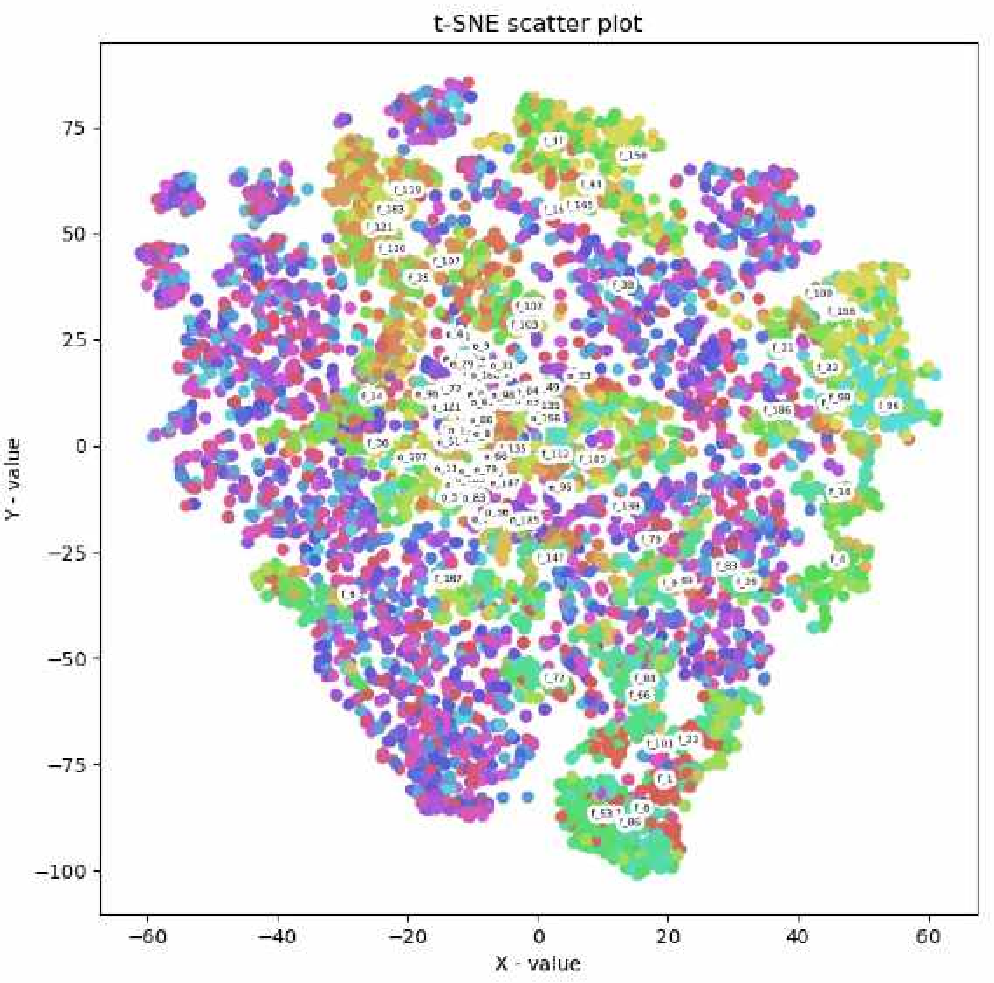


Figure 4.4: t-SNE Clustering Plot of Real and Fake Embeddings in Text Embedding Semantic Space on CUB dataset: GAN-based Image-To-Text Synthesis

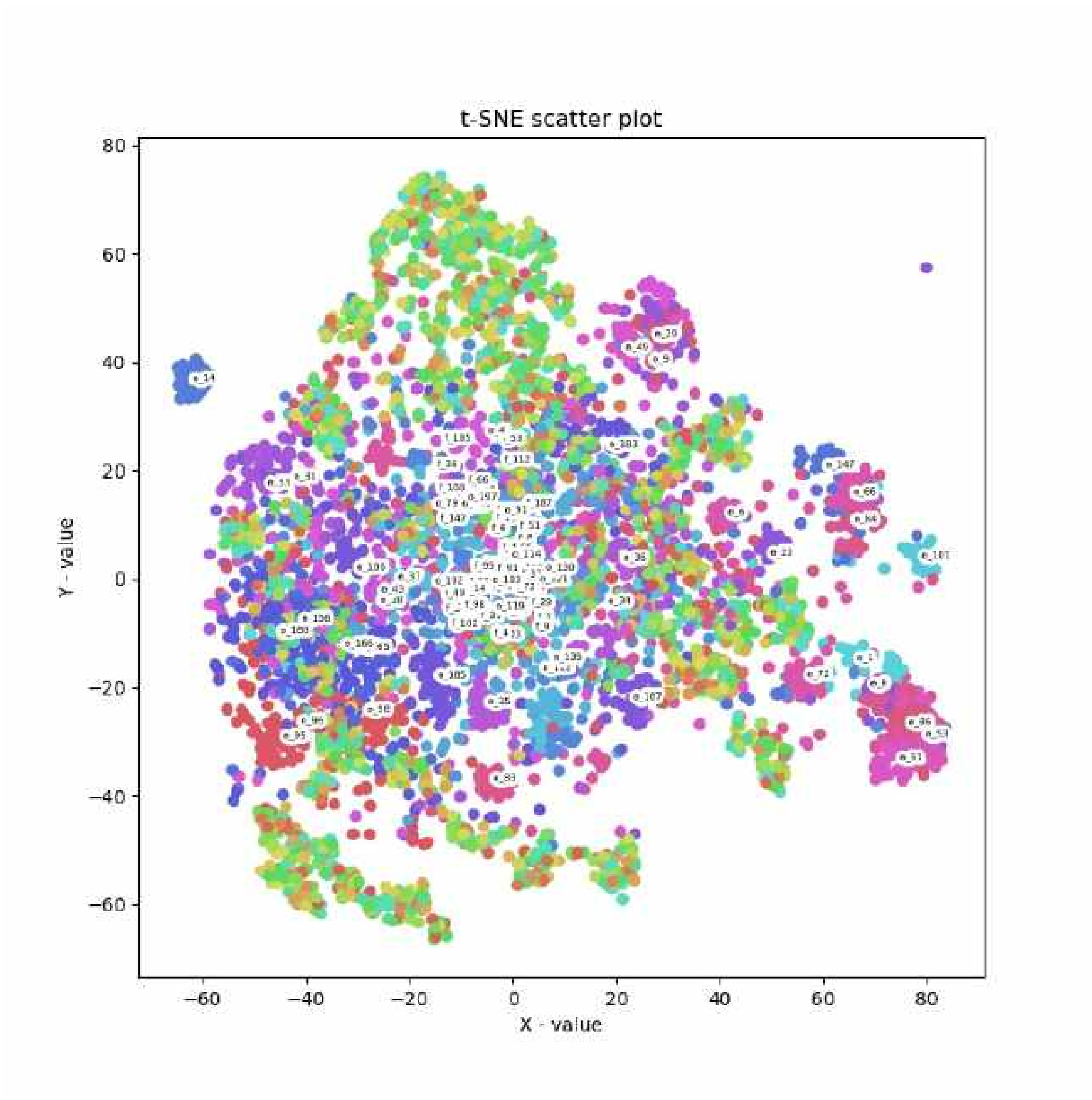
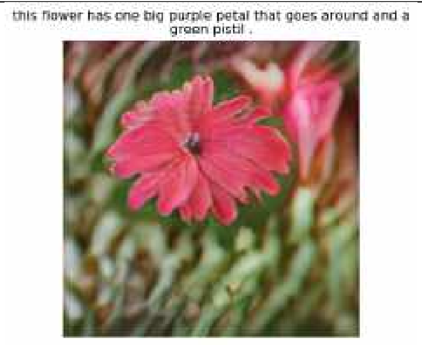
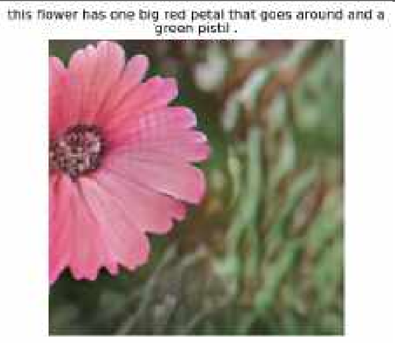
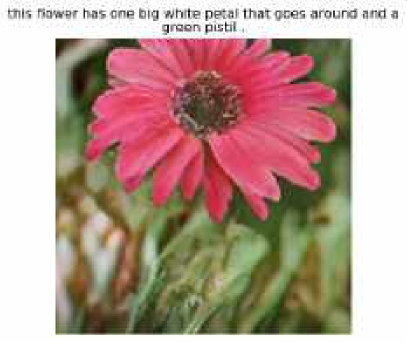
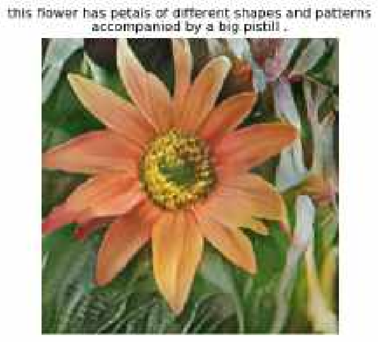
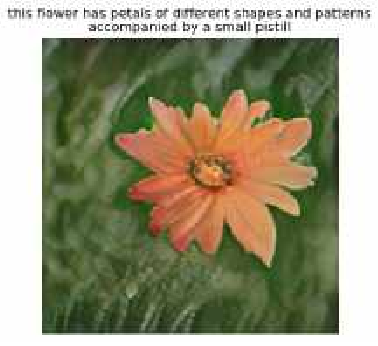


Figure 4.5: t-SNE Clustering Plot of Real and Fake Embeddings in Image Embedding Semantic Space on CUB dataset: GAN-based Text-To-Image Synthesis

Captions	Images
This flower has one big **purple** petal that goes around and a green pistil.	 this flower has one big purple petal that goes around and a green pistil .

This flower has one big **red** petal that goes around and a green pistil.	 this flower has one big red petal that goes around and a green pistil .
This flower has one big **white** petal that goes around and a green pistil.	 this flower has one big white petal that goes around and a green pistil .
This flower has petals of different shapes and patterns accompanied by a **big** pistil.	 this flower has petals of different shapes and patterns accompanied by a big pistil .
This flower has petals of different shapes and patterns accompanied by a **small** pistil.	 this flower has petals of different shapes and patterns accompanied by a small pistil

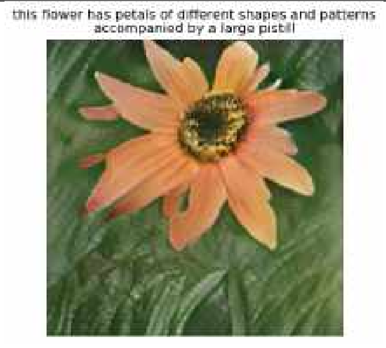
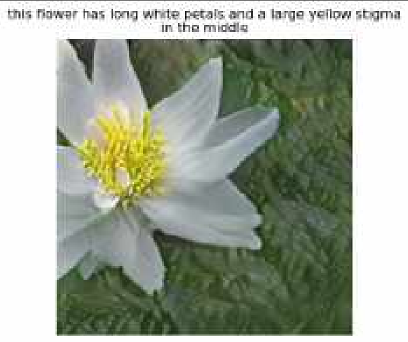
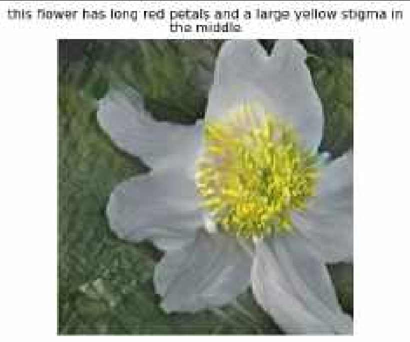
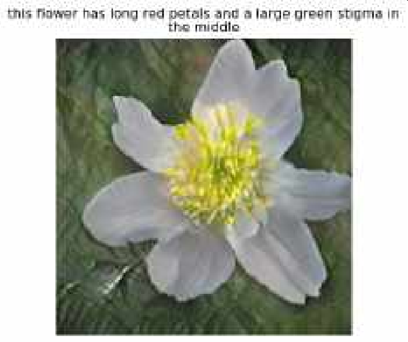
This flower has petals of different shapes and patterns accompanied by a **large** pistil.	 this flower has petals of different shapes and patterns accompanied by a large pistil
This flower has long **white** petals and a large yellow stigma in the middle.	 this flower has long white petals and a large yellow stigma in the middle
This flower has long **red** petals and a large yellow stigma in the middle.	 this flower has long red petals and a large yellow stigma in the middle
This flower has long **red** petals and a large **green** stigma in the middle.	 this flower has long red petals and a large green stigma in the middle

Table 4.3: Changes observed in generated images with minor changes in input text for Oxford-102 Dataset

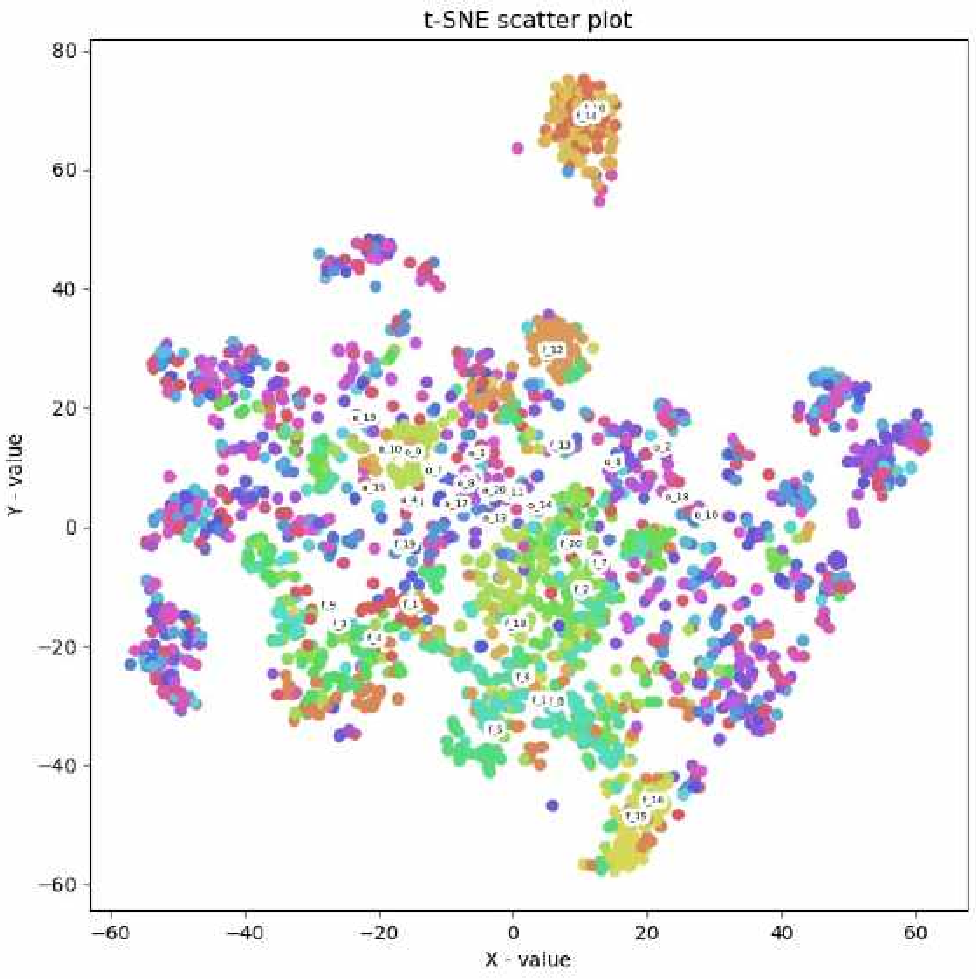
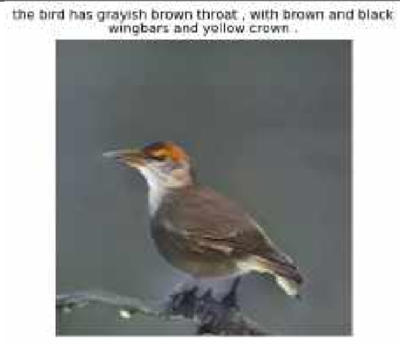
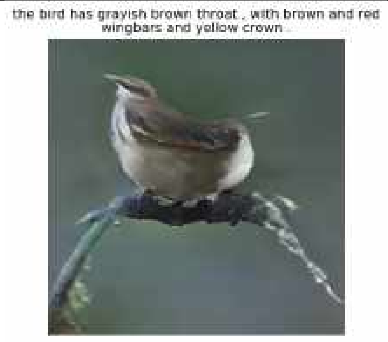
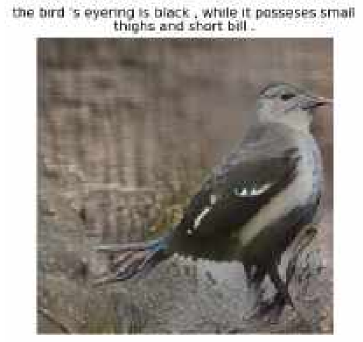
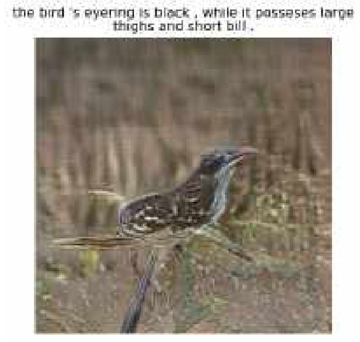
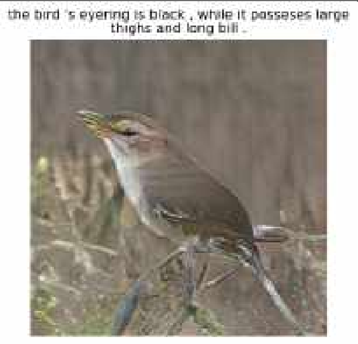
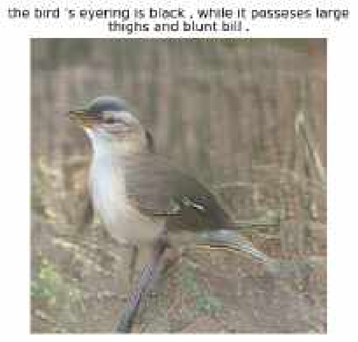
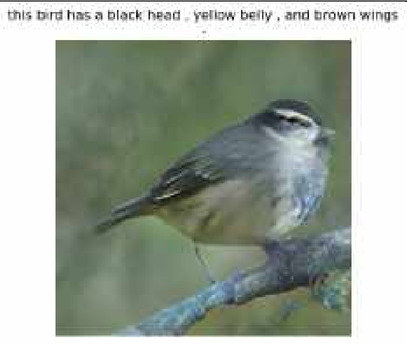
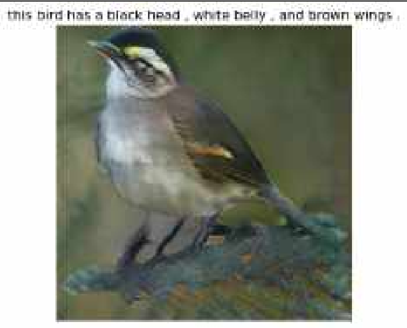


Figure 4.6: t-SNE Clustering Plot of Real and Fake Embeddings in Text Embedding Semantic Space on Oxford dataset: GAN-based Image-To-Text Synthesis

Captions	Images
The bird has greyish brown throat, with brown and **red** wingbars and **brown** crown.	the bird has grayish brown throat , with brown and red wingbars and brown crown .

The bird has greyish brown throat, with brown and **black** wingbars and **yellow** crown.	 the bird has grayish brown throat , with brown and black wingbars and yellow crown .
The bird has greyish brown throat, with brown and **red** wingbars and **yellow** crown.	 the bird has grayish brown throat , with brown and red wingbars and yellow crown .
The bird's eyeing is black, while it possess **small** thighs and short bill.	 the bird 's eyering is black , while it posseses small thighs and short bill .
The bird's eyeing is black, while it possess **large** thighs and short bill.	 the bird 's eyering is black , while it posseses large thighs and short bill .

The bird's eyeing is black, while it possess **large** thighs and **long** bill.	 the bird 's eyening is black , while it passeses large thighs and long bill .
The bird's eyeing is black, while it possess **large** thighs and **blunt** bill.	 the bird 's eyening is black , while it passeses large thighs and blunt bill .
This bird has a black head, **yellow** belly, and brown wings.	
This bird has a black head, **white** belly, and brown wings.	 this bird has a black head , white belly , and brown wings .

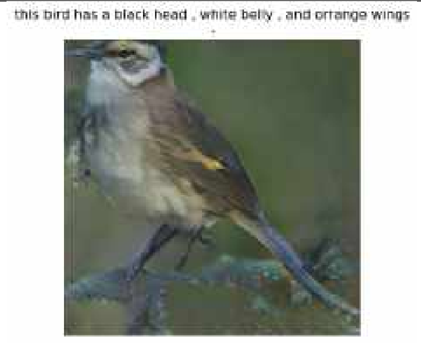
This bird has a black head, **white** belly, and **orange** wings.	 this bird has a black head , white belly , and orrange wings

Table 4.4: Changes observed in generated images with minor changes in input text for CUB-200-2100 Dataset

4.4 Error Analysis

A thorough analysis revealed scenarios for possible reasons of errors. The strength of the end-to-end model is dependant on strength of the individual components, as models are trained separately; so errors in each component have a cumulative impact on the end-to-end network. The error in image autoencoder and text autoencoder, also introduce error in image embeddings and text embeddings, which further affect the performance of end-to-end network.

Also the autoenders are trained separately and as they do not share weights or information, the latent space they generate for each of the modalities are completely disjoint. The mapping networks learns the semantic distribution without any cross-modal supervision, however during the training of the mapping networks, the encoder and the decoder weights of autoencoders are kept fixed; so it becomes difficult for the mapping networks to correctly align the learnt distribution so that embeddings are matched on semantic level.

4.5 Conclusion

In this work we have developed both GAN-based and MMD-based Mapping network, and integrated into end-to-end Image-to-text and Text-to-image synthesis network. We have also evaluated the performance of our proposed network. While our models learns to generate one modality from the other, we also figure out that the semantic correlation across modalities is low. The t-SNE plots showed us the model learns to map the semantic spaces, although it does not learn the correct alignments of the embeddings; the word meaning is not necessarily indicative of semantic meanings

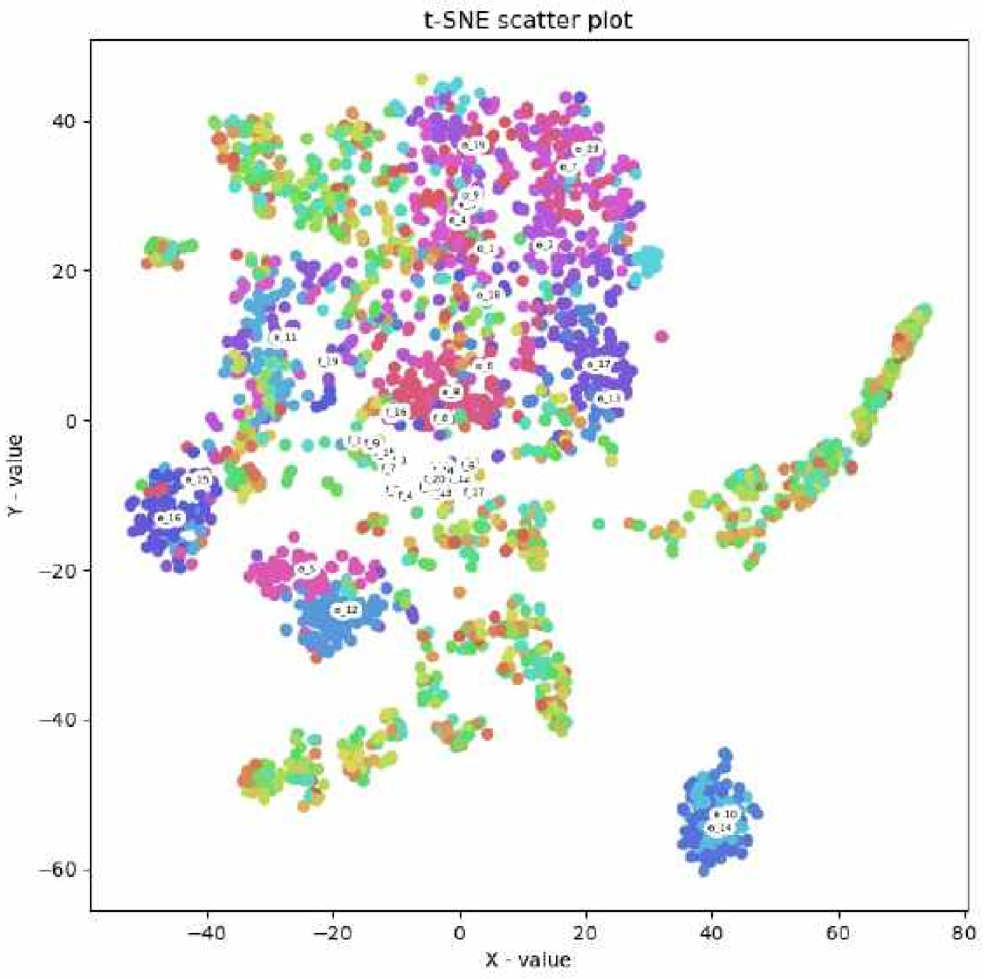


Figure 4.7: t-SNE Clustering Plot of Real and Fake Embeddings in Image Embedding Semantic Space on Oxford dataset: GAN-based Text-To-Image Synthesis

i.e. word "long" does not represent visually "long". We also figure out the GAN-based perform better than MMD-based model to some extent in case of Image-To-Text synthesis, while for Text-To-Image the performance is nearly identical.

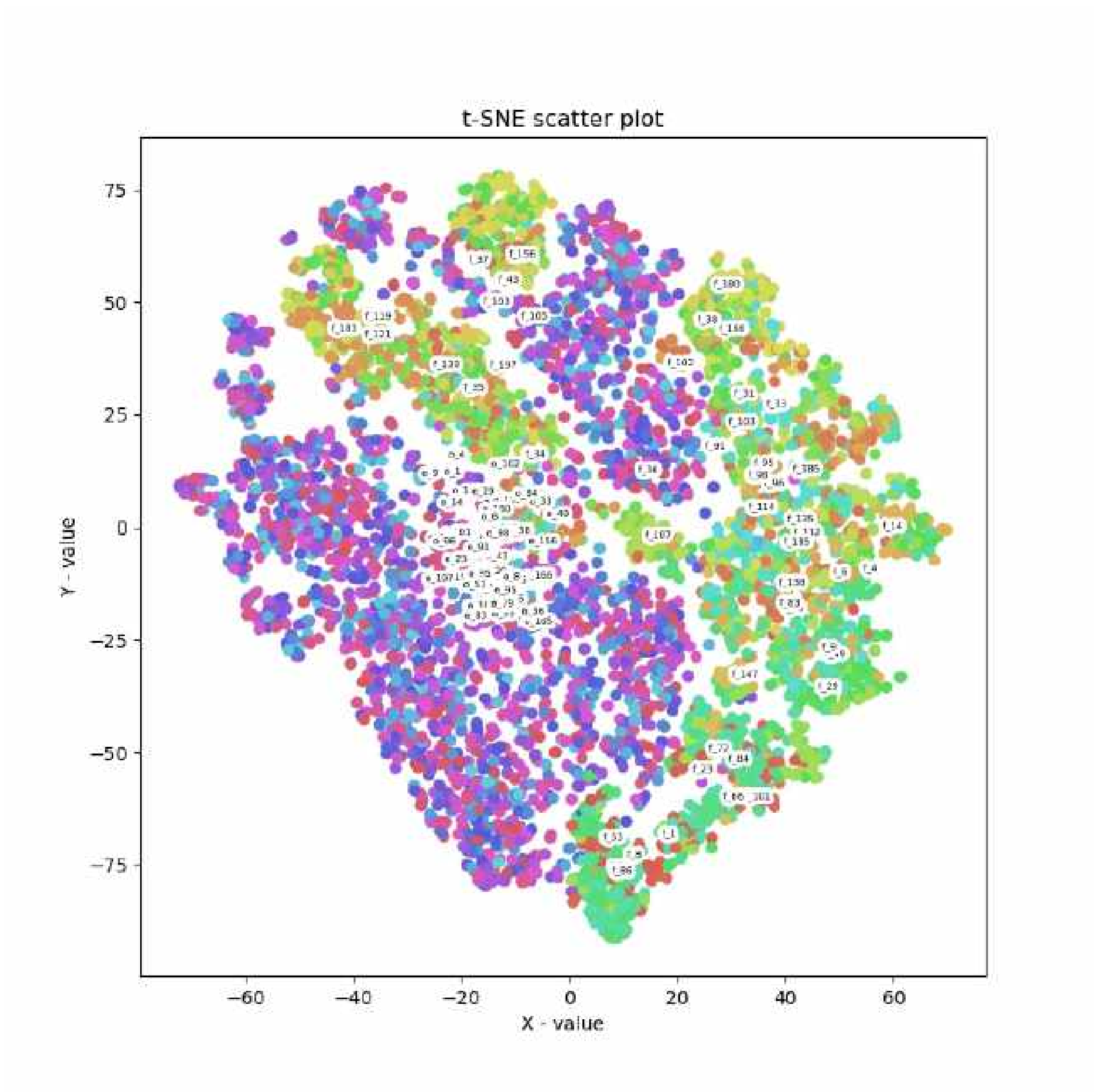


Figure 4.8: t-SNE Clustering Plot of Real and Fake Embeddings in Text Embedding Semantic Space on CUB dataset: MMD-based Image-To-Text Synthesis

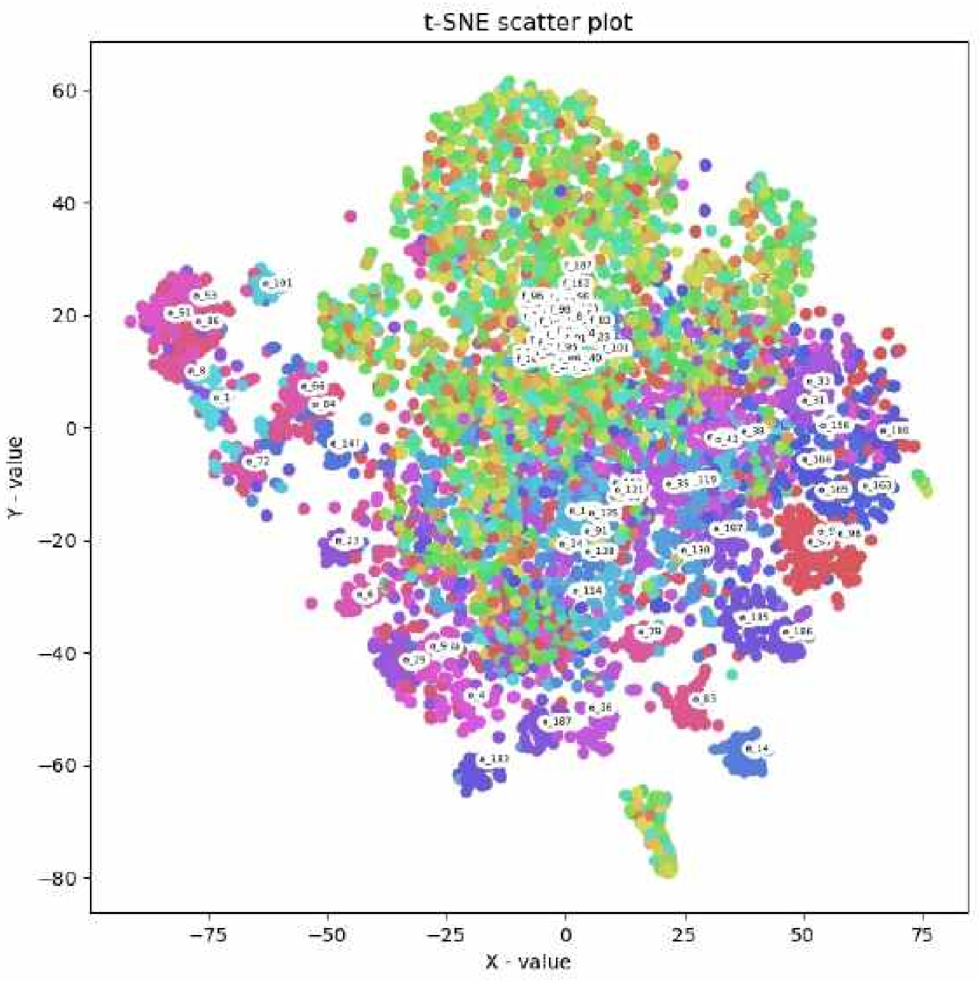


Figure 4.9: t-SNE Clustering Plot of Real and Fake Embeddings in Image Embedding Semantic Space on CUB dataset: MMD-based Text-To-Image Synthesis

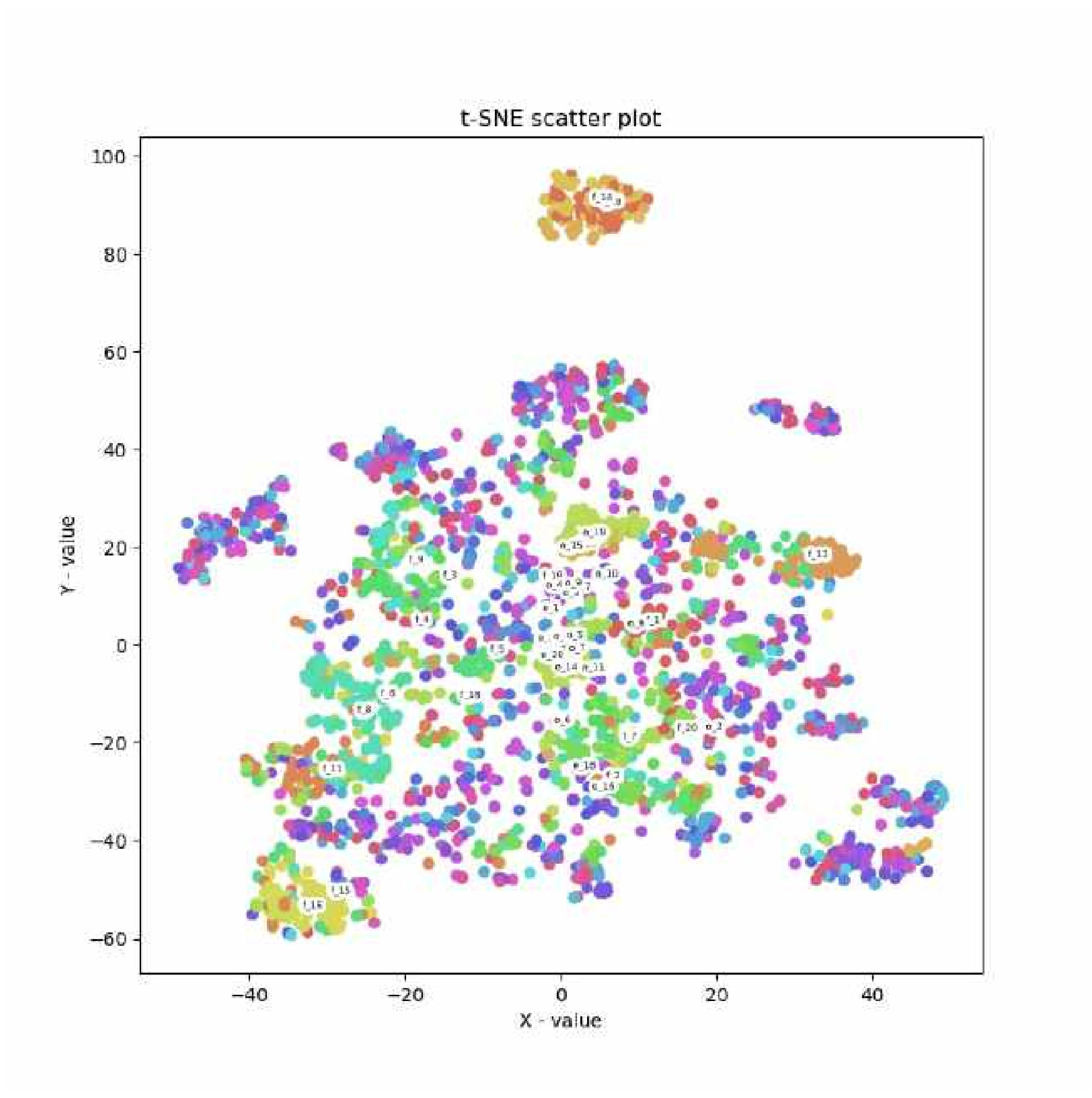


Figure 4.10: t-SNE Clustering Plot of Real and Fake Embeddings in Text Embedding Semantic Space on Oxford dataset: MMD-based Image-To-Text Synthesis

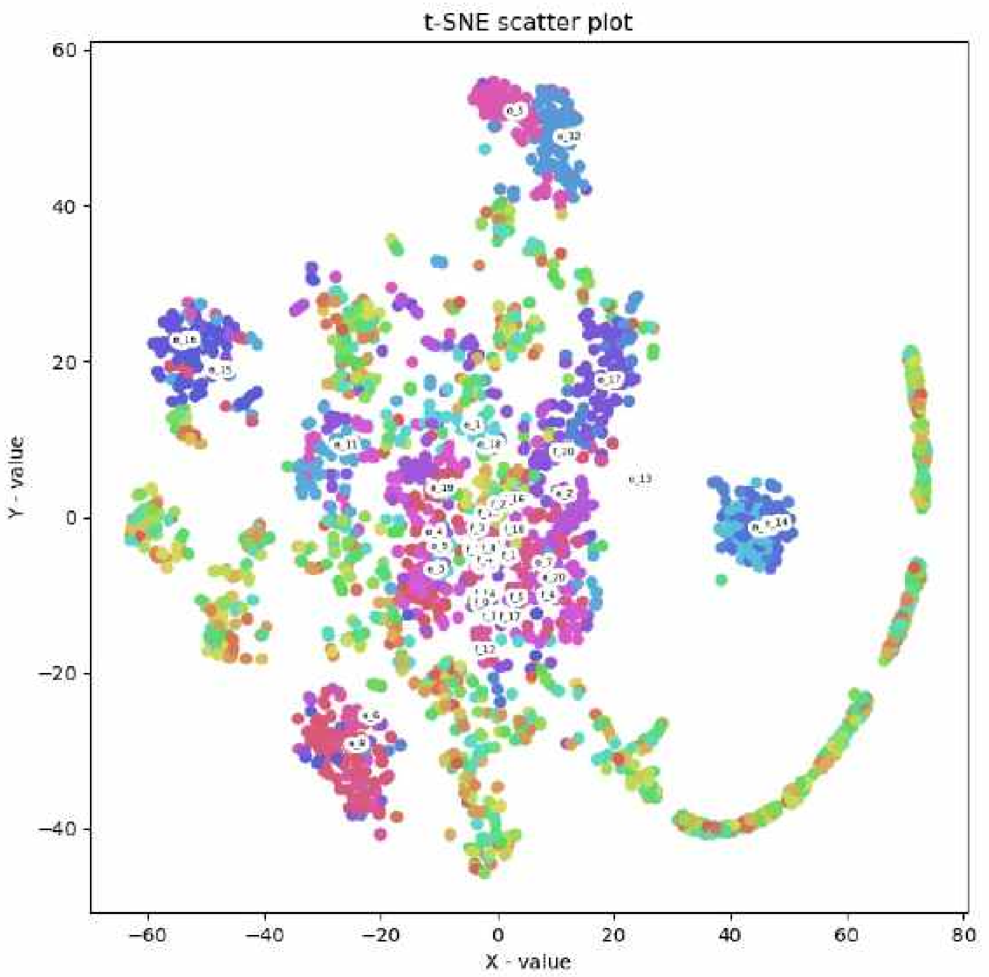


Figure 4.11: t-SNE Clustering Plot of Real and Fake Embeddings in Image Embedding Semantic Space on Oxford dataset: MMD-based Text-To-Image Synthesis

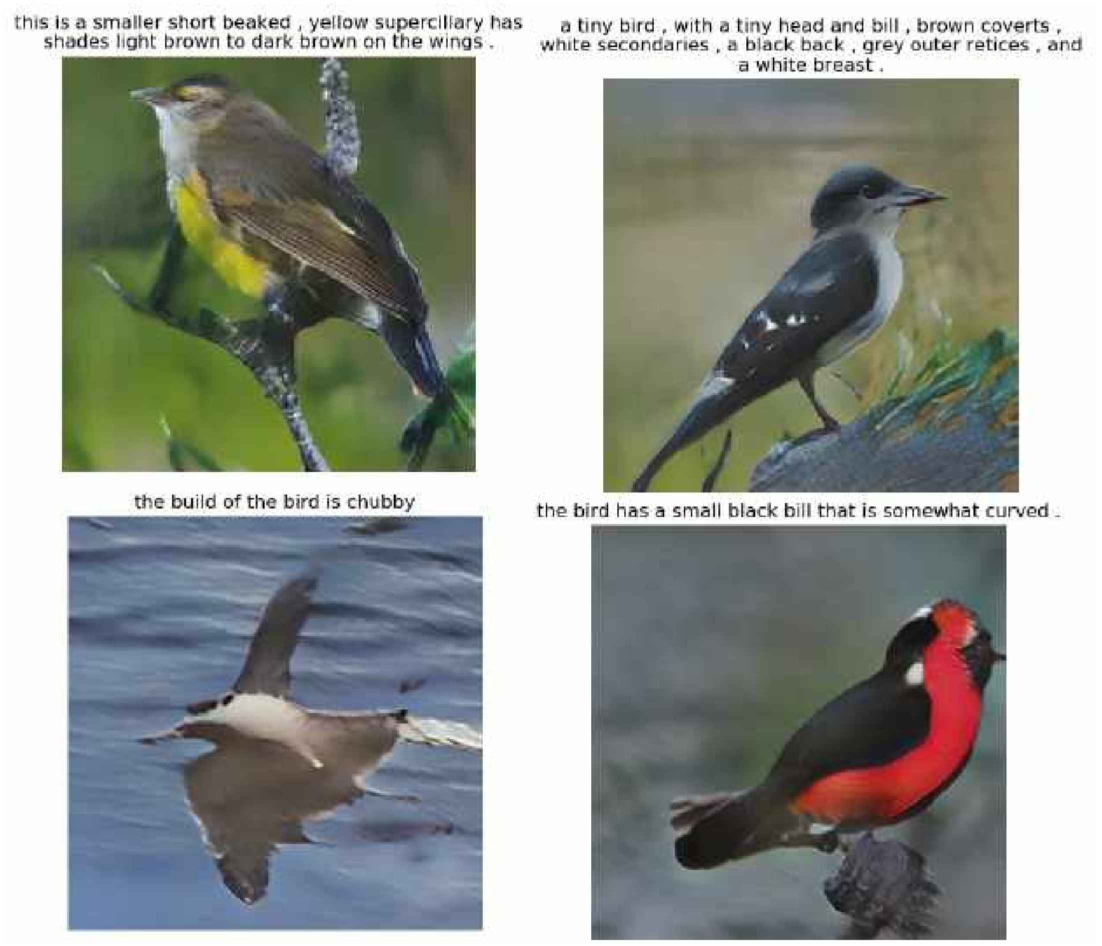

this is a smaller short beaked , yellow superciliary has shades light brown to dark brown on the wings .
a tiny bird , with a tiny head and bill , brown coverts , white secondaries , a black back , grey outer retices , and a white breast .
the build of the bird is chubby
the bird has a small black bill that is somewhat curved .

Figure 4.12: End-To-End Network Output for Birds: Text-To-Image Synthesis (GAN-based Mapping Network)

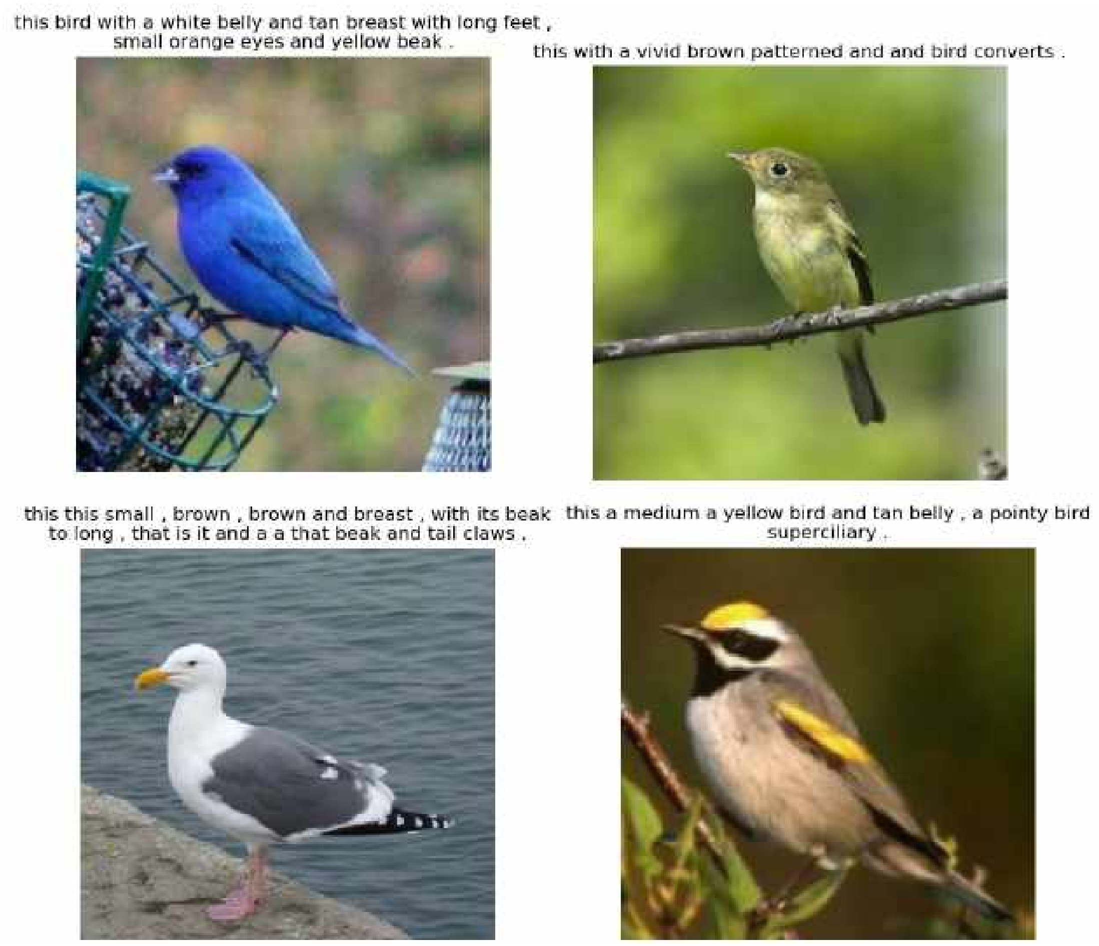

this bird with a white belly and tan breast with long feet , small orange eyes and yellow beak .
this with a vivid brown patterned and and bird converts .
this this small , brown , brown and breast , with its beak to long , that is it and a a that beak and tail claws .
this a medium a yellow bird and tan belly , a pointy bird superciliary .

Figure 4.13: End-To-End Network Output for Birds: Image-To-Text Synthesis (GAN-based Mapping Network)

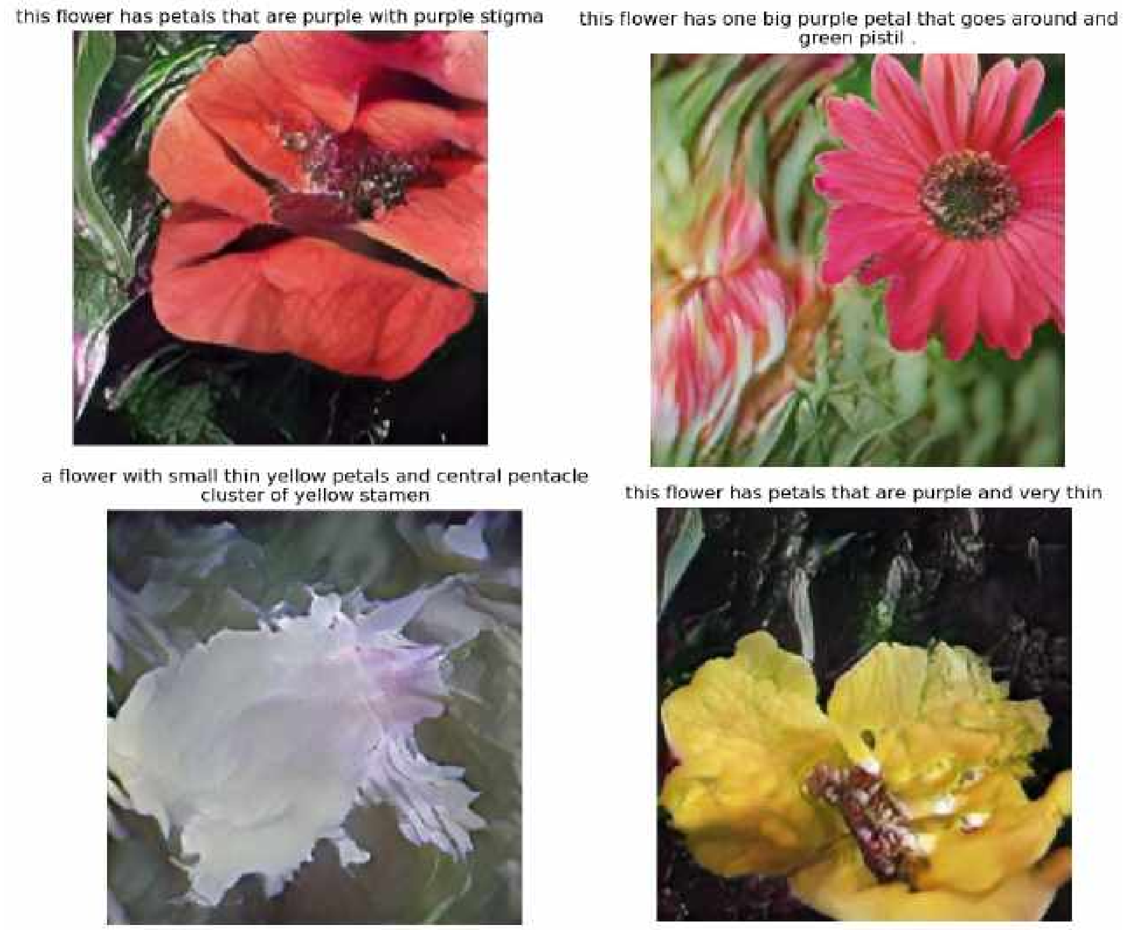
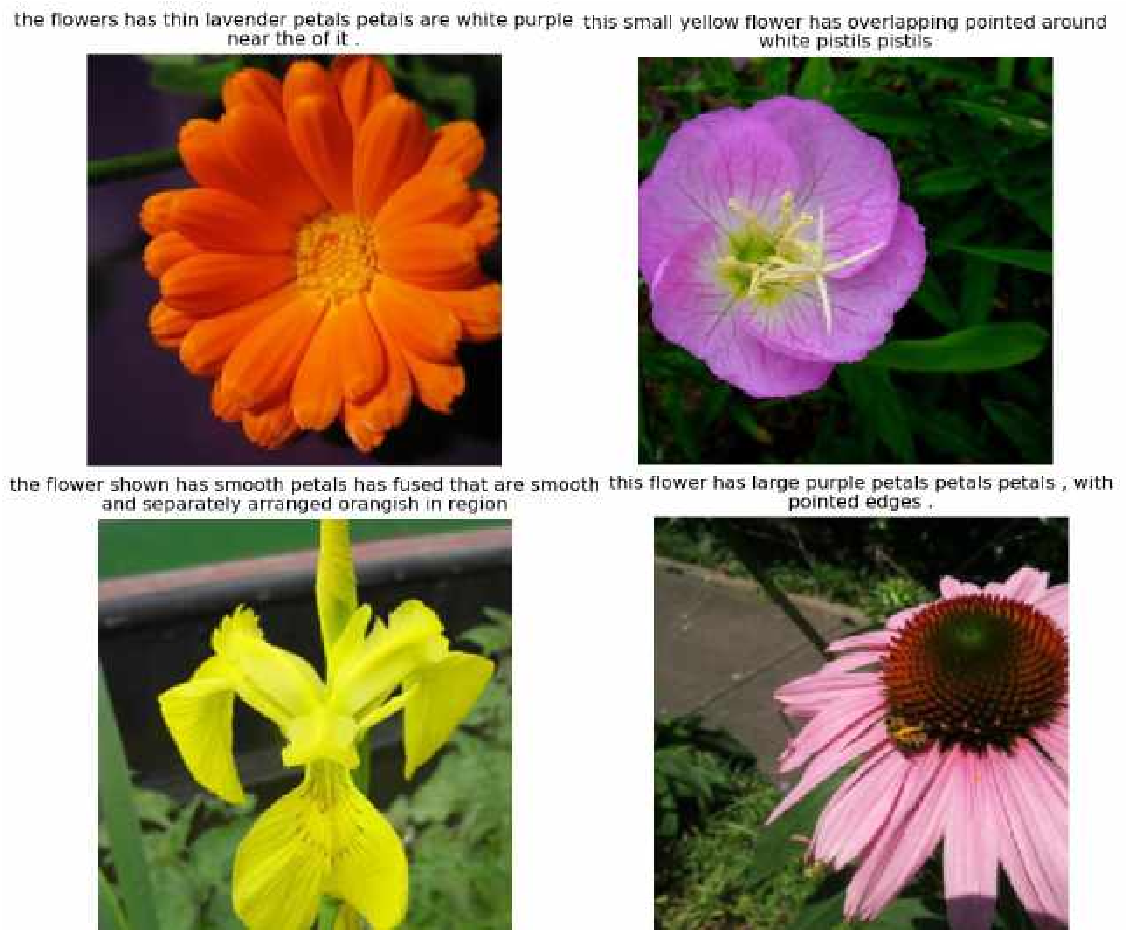

this flower has petals that are purple with purple stigma
this flower has one big purple petal that goes around and green pistil .
a flower with small thin yellow petals and central pentacle cluster of yellow stamen
this flower has petals that are purple and very thin

Figure 4.14: End-To-End Network Output for Flowers: Text-To-Image Synthesis (GAN-based Mapping Network)



the flowers has thin lavender petals petals are white purple near the of it .

this small yellow flower has overlapping pointed around white pistils pistils

the flower shown has smooth petals has fused that are smooth and separately arranged orangish in region

this flower has large purple petals petals petals , with pointed edges .

Figure 4.15: End-To-End Network Output for Flowers: Image-To-Text Synthesis (GAN-based Mapping Network)

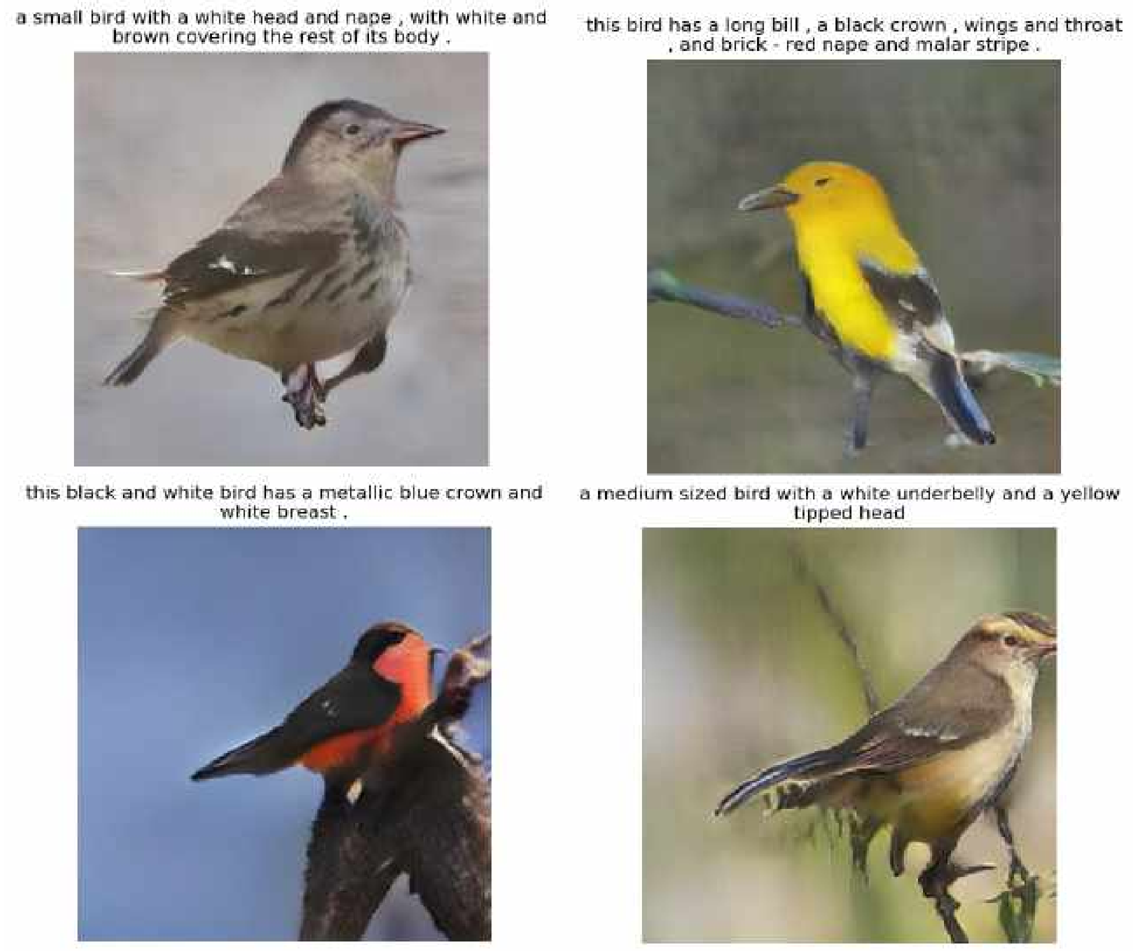

a small bird with a white head and nape , with white and brown covering the rest of its body .
this bird has a long bill , a black crown , wings and throat , and brick - red nape and malar stripe .
this black and white bird has a metallic blue crown and white breast .
a medium sized bird with a white underbelly and a yellow tipped head

Figure 4.16: End-To-End Network Output for Birds: Text-To-Image Synthesis (MMD-based Mapping Network)

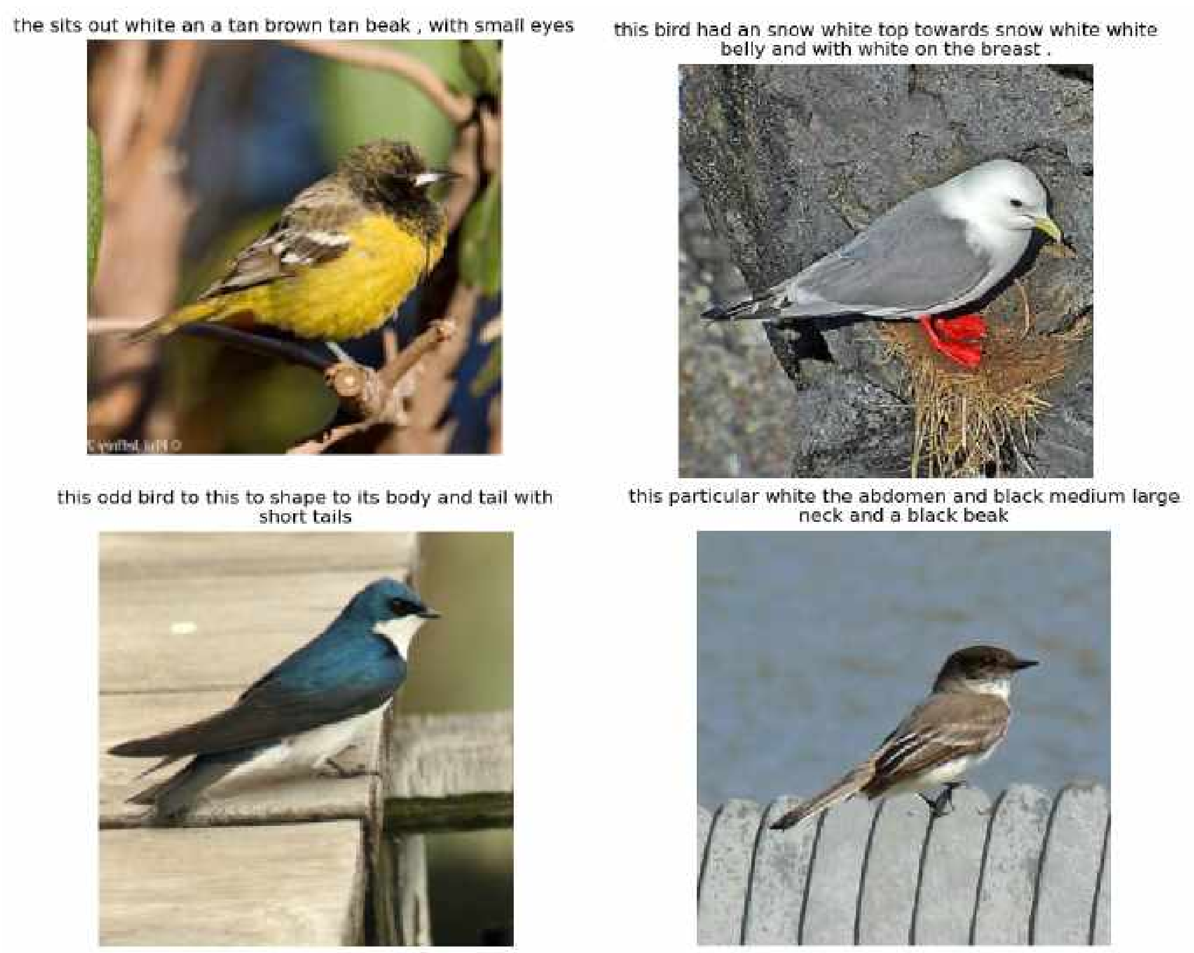

the sits out white an a tan brown tan beak , with small eyes
this bird had an snow white top towards snow white white belly and with white on the breast .
this odd bird to this to shape to its body and tail with short tails
this particular white the abdomen and black medium large neck and a black beak

Figure 4.17: End-To-End Network Output for Birds: Image-To-Text Synthesis (MMD-based Mapping Network)

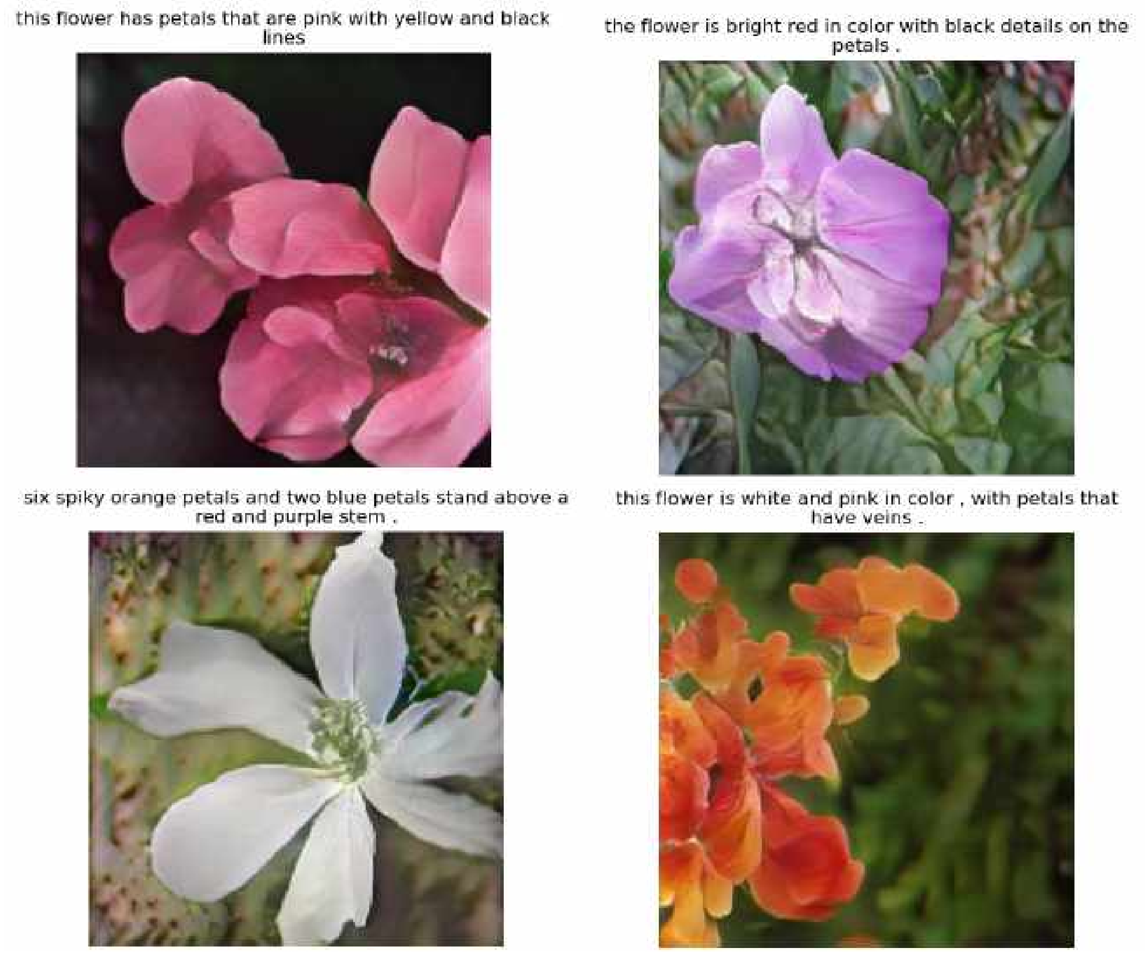

this flower has petals that are pink with yellow and black lines
the flower is bright red in color with black details on the petals .
six spiky orange petals and two blue petals stand above a red and purple stem .
this flower is white and pink in color , with petals that have veins .

Figure 4.18: End-To-End Network Output for Flowers: Text-To-Image Synthesis (MMD-based Mapping Network)

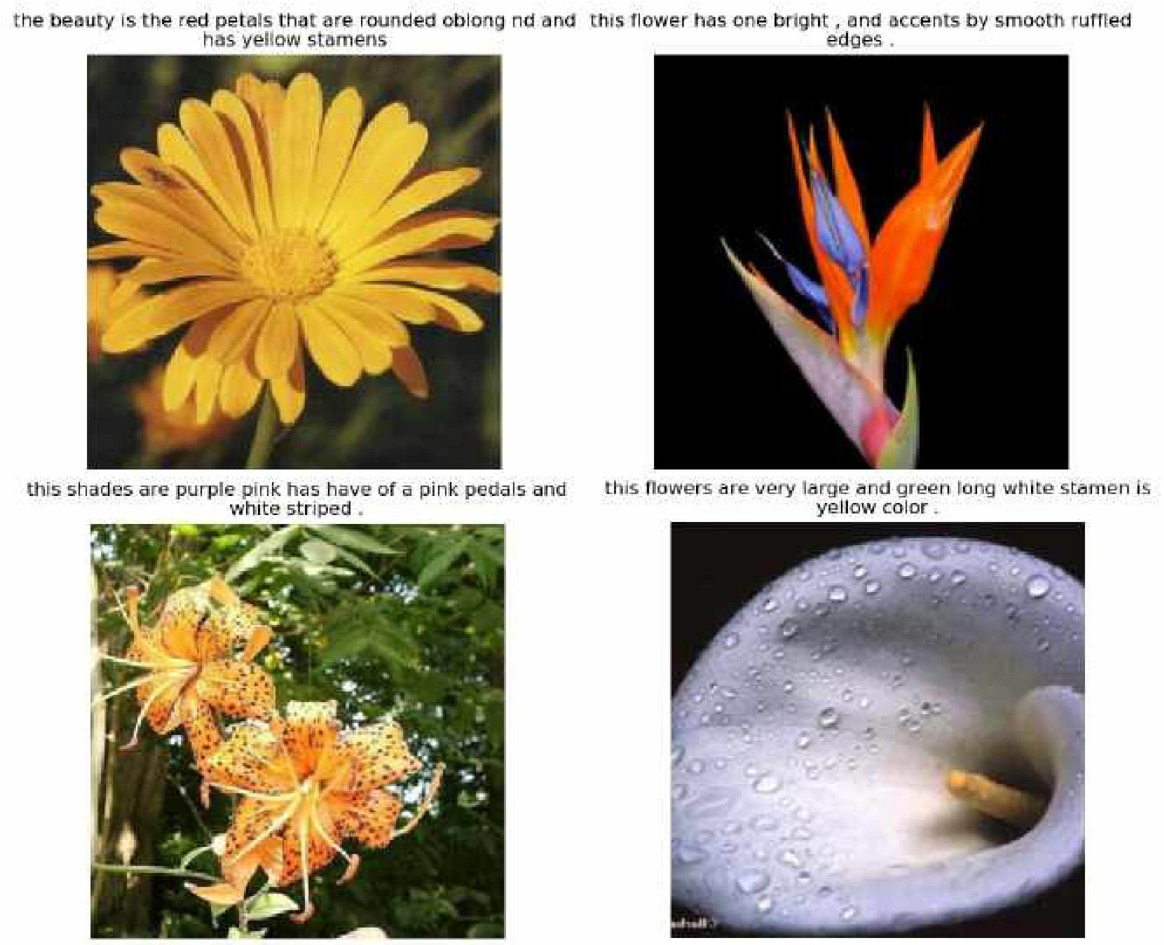

the beauty is the red petals that are rounded oblong nd and has yellow stamens
this flower has one bright , and accents by smooth ruffled edges .
this shades are purple pink has have of a pink pedals and white striped .
this flowers are very large and green long white stamen is yellow color .

Figure 4.19: End-To-End Network Output for Flowers: Image-To-Text Synthesis (MMD-based Mapping Network)

Chapter 5

Conclusion and Future Direction

In this work we have first built different image autoencoder setups and compare their performances and designed best architecture based on our requirements. Then we also explored text autoencoders to select the best configuration for our text latent representation. Finally we designed both MMD-based and GAN-based cross-modal mapping networks and assembled into Text-to-image and Image-to-text synthesis networks. We also evaluated the performance of our synthesis networks on human score and proposed evaluation metrics for automatic evaluations. We have analysed the results both qualitatively and quantitatively. The analysis shows that the proposed method learns the mapping without any supervision but fails to correctly align the two modalities such that their semantic meanings are preserved. However, considering it's a completely unsupervised approach, the mapping learnt by our network could work as initialization for the next step. In future we plan on using the learnt weights as the initialization of the end-to-end systems, then fine tune and jointly train the entire network, till the embedding spaces are properly aligned.

References

Artetxe, M., Labaka, G., Agirre, E., & Cho, K. (2017). Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*.

Banerjee, S., & Lavie, A. (2005). Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization* (pp. 65–72).

Berlinet, A., & Thomas-Agnan, C. (2011). *Reproducing kernel hilbert spaces in probability and statistics*. Springer Science & Business Media.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., & Robinson, T. (2013). One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Das, A., Kottur, S., Gupta, K., Singh, A., Yadav, D., Moura, J. M., ... Batra, D. (2017). Visual dialog. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 326–335).

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 ieee conference on computer vision and pattern recognition* (pp. 248–255).

Dutoit, T. (1997). *An introduction to text-to-speech synthesis* (Vol. 3). Springer Science & Business Media.

Fan, L. (2011, January 4). *Semantic visual search engine.* Google Patents. (US Patent 7,865,492)

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672–2680).

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., & Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*, *13*(Mar), 723–773.

Gretton, A., Sejdinovic, D., Strathmann, H., Balakrishnan, S., Pontil, M., Fukumizu, K., & Sriperumbudur, B. K. (2012). Optimal kernel choice for large-scale two-sample tests. In *Advances in neural information processing systems* (pp. 1205–1213).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems* (pp. 6626–6637).

Hou, X., Shen, L., Sun, K., & Qiu, G. (2017). Deep feature consistent variational autoencoder. In *2017 ieee winter conference on applications of computer vision (wacv)* (pp. 1133–1141).

Jiang, F., Tao, W., Liu, S., Ren, J., Guo, X., & Zhao, D. (2017). An end-to-end compression framework based on convolutional neural networks. *IEEE Transactions on Circuits and Systems for Video Technology*, *28*(10), 3007–3018.

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kiros, R., Salakhutdinov, R., & Zemel, R. (2014). Multimodal neural language models. In *International conference on machine learning* (pp. 595–603).

Lample, G., Conneau, A., Denoyer, L., & Ranzato, M. (2017). Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Lample, G., Ott, M., Conneau, A., Denoyer, L., & Ranzato, M. (2018). Phrase-based & neural unsupervised machine translation. *arXiv preprint arXiv:1804.07755*.

Li, C., Gao, X., Li, Y., Li, X., Peng, B., Zhang, Y., & Gao, J. (2020). Optimus: Organizing sentences via pre-trained modeling of a latent space. *arXiv preprint arXiv:2004.04092*.

Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., & Póczos, B. (2017). Mmd gan: Towards deeper understanding of moment matching network. In *Advances in neural information processing systems* (pp. 2203–2213).

Lin, C.-Y. (2004, July). ROUGE: A package for automatic evaluation of summaries. In *Text summarization branches out* (pp. 74–81). Barcelona, Spain: Association for Computational Linguistics. Retrieved from `https://www.aclweb.org/anthology/W04-1013`

Maaten, L. v. d., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, *9*(Nov), 2579–2605.

Manakov, I., Rohm, M., & Tresp, V. (2019). Walking the tightrope: An investigation of the convolutional autoencoder bottleneck. *arXiv preprint arXiv:1911.07460*.

Palangi, H., Deng, L., Shen, Y., Gao, J., He, X., Chen, J., . . . Ward, R. (2016). Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech* *and Language Processing (TASLP)*, *24*(4), 694–707.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics* (pp. 311–318).

Pedersoli, M., Lucas, T., Schmid, C., & Verbeek, J. (2017). Areas of attention for image captioning. In *Proceedings of the ieee international conference on computer vision* (pp. 1242–1250).

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp)* (pp. 1532–1543).

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. (2016). Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*.

Reed, S. E., Akata, Z., Mohan, S., Tenka, S., Schiele, B., & Lee, H. (2016). Learning what and where to draw. In *Advances in neural information processing systems* (pp. 217–225).

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training gans. In *Advances in neural information processing systems* (pp. 2234–2242).

Shi, Y., Siddharth, N., Paige, B., & Torr, P. (2019). Variational mixture-of-experts autoencoders for multi-modal deep generative models. In *Advances in neural information processing systems* (pp. 15692–15703).

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104–3112).

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2818–2826).

Theis, L., Shi, W., Cunningham, A., & Huszár, F. (2017). Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*.

Tolstikhin, I. O., Sriperumbudur, B. K., & Schölkopf, B. (2016). Minimax estimation of maximum mean discrepancy with radial kernels. In *Advances in neural information processing systems* (pp. 1930–1938).

Vedantam, R., Lawrence Zitnick, C., & Parikh, D. (2015). Cider: Consensus-based image description evaluation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 4566–4575).

Venugopalan, S., Anne Hendricks, L., Rohrbach, M., Mooney, R., Darrell, T., & Saenko, K. (2017). Captioning images with diverse objects. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 5753–5761).

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 3156–3164).

Wah, C., Branson, S., Welinder, P., Perona, P., & Belongie, S. (2011). *The Caltech-UCSD Birds-200-2011 Dataset* (Tech. Rep. No. CNS-TR-2011-001). California Institute of Technology.

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, *2*(1-3), 37–52.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning* (pp. 2048–2057).

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., & Metaxas, D. N. (2017). Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the ieee international conference on computer vision* (pp. 5907–5915).

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., & Metaxas, D. N. (2018). Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *IEEE transactions on pattern analysis and machine intelligence*, *41*(8), 1947–1962.