ABSTRACT

Title of Thesis:	AN ANALYSIS OF BOTTOM-UP ATTENTION MODELS AND MULTI-MODAL REPRESENTATION LEARNING FOR VISUAL QUESTION ANSWERING
	Venkatraman Narayanan, Master of Science, 2019
Thesis Directed By:	Dr. Abhinav Shrivastava, Computer Science, UMIACS and CfAR

A Visual Question Answering (VQA) task is the ability of a system to take an image and an open-ended, natural language question about the image and provide a natural language text answer as the output. The VQA task is a relatively nascent field, with only a few strategies explored. The performance of the VQA system, in terms of accuracy of answers to the image-question pairs, requires a considerable overhaul before the system can be used in practice. The general system for performing the VQA task consists of an image encoder network, question encoder network, a multi-modal attention network that combines the information obtained image and question, and answering network that generates natural language answers for the image-question pair.

In this thesis, we follow two strategies to improve the performance (accuracy) of VQA. The first is a representation learning approach (utilizing the state-of-the-art Generative Adversarial Models (GANs) (Goodfellow, et al., 2014)) to improve the image encoding system of VQA. This thesis evaluates four variants of GANs to identify a GAN architecture that best captures the data distribution of

the images, and it was determined that GAN variants become unstable and fail to become a viable image encoding system in VQA. The second strategy is to evaluate an alternative approach to the attention network, using multi-modal compact bilinear pooling, in the existing VQA system. The second strategy led to an increase in the accuracy of VQA by 2% compared to the current state-of-the-art technique.

AN ANALYSIS OF BOTTOM-UP ATTENTION MODELS AND MULTI-MODAL REPRESENTATION LEARNING FOR VISUAL QUESTION ANSWERING

by

Venkatraman Narayanan

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of Science
2019

Advisory Committee:
Dr. Abhinav Shrivastava, Assistant Professor and Committee Chair
Dr. Yiannis Aloimonos, Professor
Dr. Dana Nau, Professor

Acknowledgments

I want to thank Dr. Abhinav Shrivastava, Assistant Professor in the Department of Computer Science at the University of Maryland. His continued support and direction, despite his busy schedule, scintillating thought experiments, as opposed to verbal direction, were beyond words. He nurtured independence in work while providing the much-needed reality check.

I would also like to thank Dr. Yiannis Aloimonos, Professor at the University of Maryland, and Dr. Dana Nau, Professor at the University of Maryland, for their incredible support and feedback on the work. I want to extend a special thanks to Dr. Aloimonos for providing additional guidance and an alternative perspective to the problem.

I would also like to express my sincere gratitude to Dr. John MacCarthy, Director of Systems Engineering Education Program at the University of Maryland, who supported me throughout my Master's degree and prepared me for my thesis research. His support unmatchable when it came to providing feedback on the report and handling the logistics of the master's thesis research. A special thanks to the University of Maryland for providing valuable resources in the form of software licenses, computers, and lab spaces.

I owe all my knowledge and accomplishments in artificial intelligence to Dr. Andrew Ng, Co-founder of Coursera. His online courses on machine learning and deep learning provided both the motivation and essential background knowledge needed for this work.

None of this would have been possible without the support of my brother, Krishna Mohan, and his family. I would also like to extend my gratitude to my parents, (Late) Narayanan Mani and Kamala Narayanan, for imparting in me the confidence to face any situation fearlessly and making me who I am. The support I received from my sister, Swetha and her family gave the confidence boost at times when I needed the most.

Finally, I would like to thank my friends, Bala Murali Manoghar Sai Sudhakar and Rama Prashanth Rama Vijayakumar, for sticking with me and providing motivation, confidence boost, much-needed distraction and critical feedback throughout the work.

Table of Contents

List of Figures

Chapter 1: Introduction

1.1 Purpose

Artificial Intelligence has evolved a lot in recent years. With the advent of Deep Learning (a class of machine learning methods based on large artificial neural networks), we have intelligent machines that work efficiently in single-disciple tasks such as Computer Vision, Natural Language Processing, Knowledge Representation, etc. The current challenge in the community, however, is learning multi-disciplinary tasks (Agrawal, et al., 2015). Free-form or open-ended Visual Question Answering (VQA) is one such task that requires multi-disciplinary knowledge beyond a single sub-domain.

A VQA task is the ability of a system to provide a natural language text answer to an open-ended, natural language question about an image. Open-ended questions require a potentially vast set of AI capabilities to answer – fine-grained recognition, object detection, activity recognition, knowledge base reasoning, and common-sense reasoning.

The VQA task is a relatively nascent field, with only a few strategies explored. The performance of the system, in terms of accuracy of answers to the image-question pairs, requires a considerable overhaul before the system can be used in practice. The purpose of this research is to improve the performance of the VQA system through cutting-edge methods in AI and deep learning.

1.2 Background

Recent advances in deep learning have made it a ubiquitous solution for most tasks in the field of artificial intelligence. Various approaches in visual question answering have adopted a deep learning based approach similar to Figure [1]. The VQA system generally consists of an (i) image encoder network that embeds the image represented by pixel values to a lower-dimensional space vector, (ii) question encoder network that converts natural language text to machine-understandable embedding, (iii) a multi-modal attention network that combines the information obtained from multiple domains (specifically from image and question in this case), (iv) an answering network that generates natural language answers for the image-question pair.

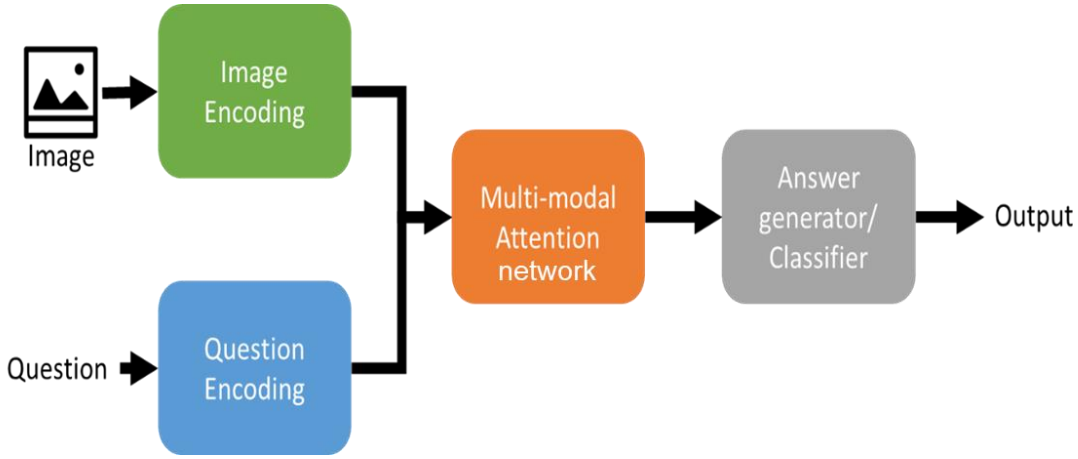

Figure 1 A Generic Visual Question Answering Pipeline

The image encoder network embeds the image represented by pixel values in a $\mathbb{R}^{N\times N\times 3}$ dimensional space, where *N* is the resolution of the image (i.e., several pixels used to represent the image), to a lower-dimensional abstract space vector. The image encoder network is usually a partial pre-trained object classification

network. The hidden layers in an object classification are tuned to be a lower-dimensional embedding of objects in the image and hence used in the image encoding pipeline of the VQA network. The question encoder network converts natural language text to machine-understandable embedding. The encoder is tuned to bring out the contextual meaning necessary for answering the question. Popular recurrent neural network techniques are used to achieve question encoding. The multi-modal attention network combines the information from multiple domains (specifically from image and question in this case) and retrieves the necessary information for achieving the VQA task. Several simple to more complex mechanisms has been used to realize this network layer, the details of which will follow. The final stage in the system is an answering system. It generates natural language answers for the image-question pair. It is the reverse process of question encoding.

Several implementations following the pipeline mentioned above were realized. The implementation aspects of the most popular implementations are discussed in Table 1.

1. (Agrawal, et al., 2015).	
Image Encoding	The image embedding was realized from l_2 normalized activations from the last hidden layer of VGGNet (Simonyan & Zisserman, 2015) image classification network to get a $\mathbb{R}^{4096}$ dimensional image embedding. The image embedding is further reduced to $\mathbb{R}^{1024}$ dimensional embedding by a fully connected layer and a *tanh (details in appendix I)* non-linearity.

Question Encoding Network	The question embedding consists of a Bag-of-Words (BoW) embedding. The top 1000 words in the questions are used to create a bag-of-words representation. This embedding is further enhanced by Long Short-term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) network with two hidden layers is used to obtain 2048-dim embedding for the question.
Attention Mechanism	The image and question embedding are then fused via element-wise multiplication.
Answering mechanism	This combined image + question embedding from the attention mechanism is then passed to a Multi-Layer Perceptron (MLP) – a fully connected neural network, followed by a *soft-max* (details in appendix I) layer to obtain a probability over K (=1000) frequent answers in the dataset. The answer (out of K) with the highest probability of given as the answer for the image-question pair.
Comments	The team kick-started the research in visual question answering by developing a dataset (detailed in chapter 2) for the task and coming up with a novel implementation
2. (Lu, Yang, Batra, & Parikh, 2016)	
Image Encoding	The image embedding consists of using the last hidden layer of Resnet (He, Zhang, Ren, & Sun, 2016) architecture.
Question Encoding Network	The question embedding happens in three-fold owing to the hierarchical nature of attention mechanism. With the one-hot encoding of the question words, the words are embed via an

	LSTM (Hochreiter & Schmidhuber, 1997) network to a vector space. A 1-D convolution computes the phrase level features on the word embedding vectors. Concretely, at each word location, the inner product of the word vectors with filters of three window sizes: unigram, bigram, and trigram are computed. The word-level features are appropriately 0-padded before feeding into bigram and trigram convolutions to maintain the length of the sequence after convolution. A max-pooling layer is added to across the n-grams to obtain phrase-level features.
Attention Mechanism	The attention mechanism sequentially alternates between generating the image and question encoding. It consists of three steps:1) summarize the question into a single vector; 2) attend to the image based on the question summary; 3) attend to the question based on the attended image feature (from the previous step).
Answering mechanism	Same as the approach by (Agrawal, et al., 2015) (mentioned above)
Comments	
3. (Xiong, Merity, & Socher, 2016)	
Image Encoding	The image encodings are generated, using the VGGNet (Simonyan & Zisserman, 2015) and retrieving the last hidden layer of the network. The hidden layer output is then followed linear layer with *tanh* activation. An input fusion layer is added by

	snaking through the image vectors and applying bi-directional Gated Recurrent Unit (GRU) (Cho, et al., 2014), to obtain globally aware inputs.
Question Encoding Network	The question encoding consists of two components. The first component is a sentence reader, responsible only for encoding the words into a sentence embedding. The second component is the input fusion layer, allowing for interactions between sentences. The implementation uses bi-directional GRU (Cho, et al., 2014) for this input fusion layer because it allows information from both past and future sentences.
Attention Mechanism	The image-question attention mechanism consists of an episodic memory module. The episode memory module may pass over the input multiple times, updating episode memory after each pass. It retrieves information by focusing attention on a subset of the input vectors through an attention gate.
Answering mechanism	Answer prediction is based on the episodic memory after the last pass and the question embedding. The prediction module has two configurations: (i) For simple (one-word) prediction - a linear layer with *soft-max* activation, (ii) For sequence output – the concatenated memory and question vectors are passed through an LSTM (Hochreiter & Schmidhuber, 1997) framework.
Comments	

A detailed explanation of the implementations mentioned above is provided in Appendix II. Table 2 contrasts the performance of the implementations mentioned above, along with the state-of-the-art implementation in this discipline (discussed later).

Table 2: Comparison of VQA baselines

S.No	***Implementation***	***Score****
1	Baseline VQA (Agrawal, et al., 2015)	51.62%
2	Hierarchical Co-Attention (Lu, Yang, Batra, & Parikh, 2016)	54.57%
3	DMNs (Xiong, Merity, & Socher, 2016)	59.14%
4	Bottom-Up Attention Mechanism (Detailed later) (Anderson, et al., 2017)	63.2%

*Score is based on evaluation metric defined as,

$$Score\ (of\ predicted\ answer) = min\left\{\frac{\#humans\ predicting\ the\ same\ answer}{3}, 1\right\} \qquad (1)$$

the accuracy (score) is calculated on the validation dataset (detailed later)

In work by (Goyal, Khot, Summers-Stay, Batra, & Parikh, 2017), it has been observed that in the above-mentioned network architectures, nullifying (replacing with zeros) the image features had little to no effect on the performance of the networks. According to their work, the reason for such behavior is because the questions have strong language priors (i.e., the questions have trivial answers, making the images unnecessary in the context). The language priors are the phenomenon by which the structure and bias in language, eliminates the need for

the image to answer the question (in the VQA task). For example, questions like “what is the color of the sky?” and “how many doors are in the car?” have a very high probability of being “blue” and “4” than other answers. This underlying structure/bias in language encourages the VQA system to predict the most probable answer, without looking at the associated image. As a remedy to the problem, the authors proposed a modified dataset with image pairs that are almost identical but have a different answer for the same question. An example of the modified dataset is provided in figure [2].

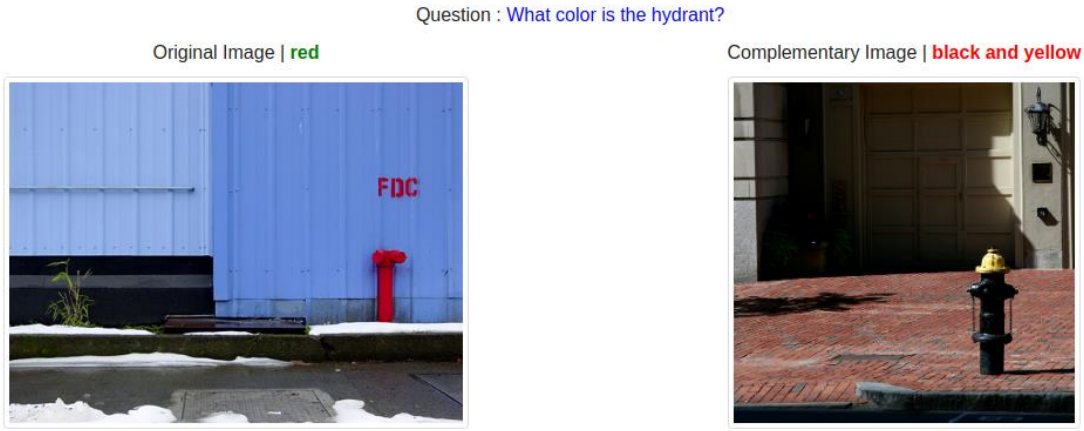

Image Source: (Goyal, Khot, Summers-Stay, Batra, & Parikh, 2017)

Figure 2 Example of modified VQA 2.0 dataset with image pairs

An alternative approach to eliminate the effect of language priors in the VQA task would be to develop an image encoding network that can generate a lower-dimensional representation of the image with on questions, rather than just the objects in the image. There have been numerous research attempts to learn representations: bilinear models, multi-view perceptron, variational auto-encoders, and Adversarial Auto-encoders were shown to learn representations (Kingma & Welling, 2014).

Generative adversarial networks (GAN) (Goodfellow, et al., 2014) have been one of the promising techniques in learning complex representations of multi-modal data. GANs are a subset of generative learning models based on game theory (Nash, 1950). GANs have proven to have excellent results for unsupervised learning tasks. GANs can be trained end-to-end through the differentiable networks and do not require approximation methods for intractable functions or inference. The basic idea of GANs is to train a discriminator and a generator simultaneously, by pitting one against the other. The goal of the discriminator is to distinguish between real samples and generated samples, and the generator tries to generate fake samples as real as possible, bridging the gap between real data and generated data. GANs have played a significant role in various tasks, such as image generation, image super-resolution, and semi-supervised learning.

The Generative adversarial network (GAN) consists of two simultaneously trained networks: a generator network G that tries to capture the data distribution of the training samples, and a discriminator network D that estimates the probability that a sample came from the training data. The generator network (G) is trained to maximize the probability of the discriminator (D) making a mistake. This framework is similar to a minimax two-player game. A unique solution lies in the space of function G and D, with G recovering the training data distribution.

The entire network is trained by competition between G and D. The nature of the competition is between G and D is different. The discriminator tries to differentiate real data samples from generator output. The generator network tries to produce output similar to the training data. The output of the discriminator is a

single scalar, representing the probability of the sample being real data. The generated data, along with a sample from training data, is used as the input to the discriminator network.

Rather than a typical convex optimization problem (which is the case in most artificial neural networks), GANs are based on a minimax game. The generator agent seeks to minimize the value function, and the discriminator seeks to maximize it. The game terminates at a saddle point that is optimum for both the agents. The generator mapping $G(z)$, where G is the differentiable function representation of input data space to output mapping and z is the noise/lower-dimensional signal input and θ_g is the perceptron parameter. The discriminator is defined by $D(x)$, where $D(x)$ is the probability of real data sample and θ_d is the network parameters of the discriminator. The discriminator is trained to maximize the probability of accurately classifying the generated (fake) data vs. real data, while the generator is trained to minimize $\log(1 - D(G(z))$. Hence D and G take part in a minimax game with a value function $\mathrm{V(G, D)}$:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_g(z)}[\log(1 - D\big(G(z)\big))]. \qquad (2)$$

Where E is the expectation value, x is an image sample from the dataset, p_{data} is the probability distribution function of the dataset and $p_g$is the probability distribution function of the generator.

1.3 Research Approach

As detailed in Chapter 1.2, incorporating multi-disciplinary knowledge representation has been the struggle in Visual Question Answering. In this thesis, we follow two strategies to improve the performance of VQA. The first is a representation learning approach to improve the **image encoding system** of VQA. The second is to evaluate an alternative approach for the **attention network** in the VQA task. An overview of these strategies is outlined below. All the code and material needed for reproducing both strategies I and II can be found at *https://github.com/vijay4313/vqa_GAN.git*

Strategy I

A representation learning approach to improve the **image encoding system** of VQA is evaluated. State-of-the-art Generative Adversarial Models (GANs) (Goodfellow, et al., 2014) are utilized for the representation learning approach. This thesis evaluates four variants of GANs to identify a GAN architecture that best captures the data distribution of the images.

The evaluation process is done by developing the various GAN networks in a Python environment with the help of TensorFlow or Pytorch deep learning libraries (details are in chapter 2). All the code necessary for developing the various GAN network architectures were developed in this thesis. Also, the code necessary for training and evaluating the network was developed in this thesis. The networks are trained using VQA 2.0 dataset. Specifically, the generator takes questions from the dataset to generate the corresponding image, and the discriminator takes either generated image or image from the dataset to predict real (data in training dataset) vs. fake (generated by generator) data. The networks

are trained until optimal (minimum) error values (defined later) are obtained for both generator and discriminator networks.

Strategy II

This thesis also evaluates an alternative approach to attention network, using multi-modal compact bilinear pooling, in the existing VQA system. The evaluation process is done by developing the VQA network architecture (detailed later) in a Python environment with the help of the Pytorch deep learning library. All the code necessary for developing the VQA network architecture was developed in this thesis. Also, the code necessary for training and evaluating the network was developed in this thesis. The network is trained using VQA 2.0 dataset with image-question pair as input, and the probability of top K (1000) frequent answers in the entire dataset is the answer for the pair as output. The network is trained until the optimal (minimum) error value (defined later) is obtained.

Chapter 2: Dataset Overview

The primary dataset utilized for Visual Question Answering is VQA 2.0 dataset (Goyal, Khot, Summers-Stay, Batra, & Parikh, 2017). The VQA 2.0 is built upon the VQA dataset (Agrawal, et al., 2015). The images for VQA 2.0 dataset are derived from COCO (Common Objects in Context) (Lin, et al., 2014) dataset by Microsoft. In total, the dataset consists of 443K train, 214K validation question, image, and answer triplets. The data collection methodology is briefly described in chapter 2.1.

2.1 Data Collection Technique

The VQA 2.0 dataset balanced the existing VQA dataset by collecting complementary images such that almost every question is associated with a pair of similar images that result in two different answers to the question. An annotation interface was used to collect complementary images from Amazon Mechanical Turks (AMT). The image, along with 24 nearest-neighbor images, are shown to AMT workers along with the question and the answer. The AMT workers are then tasked to pick an image from the list of 24 images for which the question is relevant, and the answer to the question is not. The AMT workers were adequately trained for the task in order to achieve the best results.

The 24 nearest neighbors for the image are computed by first representing each image with the activations from the penultimate ('fc7') layer of VGG deep convolutional network (Simonyan & Zisserman, 2015) and then using L_2 distances to compute neighbors. After the complementary images are collected, the second round of data annotation was conducted to collect answers on these new images. This exercise was performed to collect ten ground-truth answers to the image-question pair by showing it to 10 new AMT workers. The most common answer among the ten is chosen as the answer. This two-stage data collection process finally results in pairs of complementary images that are semantically (visually) similar but have a different answer for the same question Q. Since the two images are semantically similar, a VQA model will have to understand the subtle differences between the images to provide the right answer to both images.

In some cases, it was impossible for the AMT to select a complementary image from the 24 nearest-neighbors. In such cases, the AMTs chose “Not Possible” option. In total, such “not possible” selections make up 22% of all the questions in the VQA 2.0 dataset.

Chapter 3: Methodology

This chapter discusses in detail the two strategies taken in this thesis for improving the accuracy of the VQA system. The network architecture, training, and evaluating methodologies are discussed in detail for strategies I and II (discussed in chapter 1.3) in chapters 3.1 and 3.2, respectively.

3.1 Strategy I

Since training a GAN architecture is not a convex optimization problem, the source of instability while training a GAN cannot be determined. Hence, several variations of GANs are often evaluated to achieve stability in training. This thesis evaluates the following four GAN architectures. The following GAN architectures are discussed in detail in chapters 3.1.1 to 3.1.4

1. The Deep Convolutional GAN (DCGAN) (Radford, Metz, & Chintala, 2016) - It is one of the earliest and prominent methods for generating images from random noise input. The generator consists of learnable upscaling convolutions to generate the image, while the discriminator predicts the probability of the image being real/fake.

2. Least Squares GAN (LSGAN) - It utilizes a least-squares loss function for the discriminator.
3. Wasserstein GANs – The discriminator of Wasserstein GAN (WGAN) minimizes an approximation of Earth Movers (EM) distance.
4. Attentional Generative Adversarial Network (AttnGAN) (Xu, et al., 2018)- It is an attention-driven, multi-stage refinement GAN architecture for a fine-grained text-to-image generation. Attentional GAN can generate fine-grained details at different sub-regions of the image by paying attention to the relevant words in the natural language description.

The Python code for developing the network architecture for each GAN variant is developed in this thesis. TensorFlow and Pytorch deep learning libraries are used in developing the network architecture in Python. Also, The Python code necessary for training and evaluating the network is developed in this thesis.

A GAN architecture is essentially a complex neural network. Hence, training the GAN architecture consists of optimizing the network parameters of the architecture such that the error (defined individually for each GAN variant in chapters 3.1.1 to 3.1.4) is minimum. The image-question data pairs from the training pool of the VQA 2.0 dataset (details in chapter 2) are fed as inputs to the GAN, specifically, the question is fed to the generator part of GAN and image from either dataset or generator is fed to the discriminator. The error for the GAN architecture is identified and is minimized by the gradient descent optimizer

(definition in appendix 1) technique. In this work, the GAN training is ended under the following conditions:

1. When GAN training becomes unstable (determined based on the growth of error during GAN training)
2. When the GAN training error is unchanged for more than two training iterations.

The validation of the GAN architecture consists of feeding the image-question data pairs from the validation pool of the VQA 2.0 dataset and tracking the GAN error. The important thing to note is that the network is not optimized during validation. The central idea behind validation is to check that the GAN error does not deviate by a large amount for new data.

3.1.1 Deep Convolutional GAN (DC-GAN)

The Deep Convolutional GAN (Radford, Metz, & Chintala, 2016) is one of the earliest and prominent methods for generating images from random noise input. The generator consists of learnable upscaling convolutions to generate the image, while the discriminator predicts the probability of the image being real/fake.

Historical attempts to scale up GANs using CNNs to model images have been unsuccessful due to the sensitive stability constraints of GANs. Adopting and modifying three changes to CNN architectures was core to the success of DC GANs. Spatial pooling functions (such as max pooling) are replaced with strided convolutions, allowing the network to learn its spatial downsampling. This approach is used both in the generator, allowing it to learn its own spatial upsampling, and discriminator. Second is eliminating fully connected layers on top

of convolutional features. Global average pooling increased model stability but hurt convergence speed. The first layer of the GAN, which takes a uniform noise distribution Z as input, is a fully connected layer and is reshaped into a 4-dimensional tensor. For the discriminator, the last convolution layer is flattened and then fed into a single sigmoid output. Third is Batch Normalization, which stabilizes learning by normalizing the input to each unit to have zero mean and unit variance. This helps deal with training problems that arise due to poor initialization and helps gradient flow in deeper models. This proved critical to get deep generators to begin learning, preventing the generator from collapsing all samples to a single point, which is a standard failure mode observed in GANs.

Implementation Details

Network Architecture

The network is developed based on DC-GAN architecture. The generator consists of staggered transposed convolutional layers that perform the deconvolution process with an upscaling layer in between the filter banks. The input to the generator is the question embedding. The questions are embedded using Bag of Word encoding for the top 1000 frequent words in the training data, followed by a stacked (2 hidden layer) LSTM (Hochreiter & Schmidhuber, 1997). The convolutional layers are followed by a *Rectified Linear Unit (ReLU)* non-linearity. The final layer of the generator consists of *tanh* non-linearity. The architecture of the generator is given in figure [3]. From figure [3], the generator consists of 16 layers. The input question embeddings are passed through two fully connected layers (details in Appendix 1) (represented as a 2-dimensional

rectangular box in the figure), followed by seven convolutional layers (details in Appendix 1) (represented as 3-dimensional cuboid in the figure) each with 16 convolutional filters (details in Appendix 1). The generator consists of 7 more convolutional layers, each with eight filters. As mentioned above, all convolutional layers are transposed to perform deconvolution. The final convolutional layer produces the image.

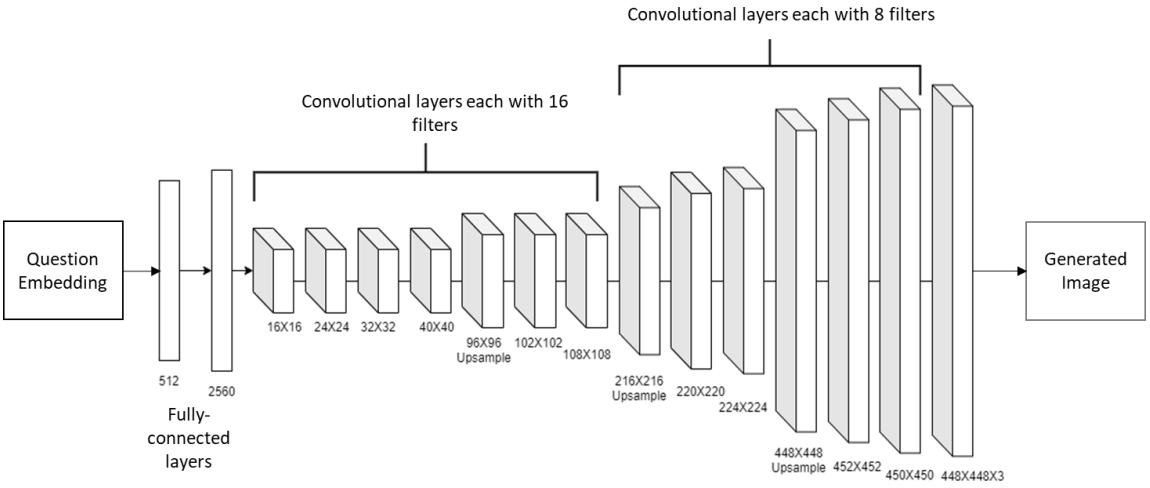

Figure 3 Network Architecture of DC-GAN Generator

The discriminator is also inspired by DC-GAN architecture. Which takes in the images (real/fake) and generates the probability of being real. The discriminator consists of multiple convolutional layers followed by the maximum pooling. The discriminator consists of dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) layers to prevent overfitting. The architecture of discriminator is given in figure [4]. From figure [4], the discriminator consists of 16 layers. The input images (of size 448X448X3 pixels) are passed through a BatchNorm layer, that normalizes the input image data by subtracting the mean pixel value of the image and dividing by the standard deviation of the pixel values

from each pixel value of the image. The BatchNorm layer is followed by 2, 6, 2, and 2 convolutional layers, each with 4, 8, 18, 32 convolutional filters, respectively. Finally, three fully connected layers are added to get the probability of input image being real (from the dataset) or fake (from the generator).

The complete GAN architecture was implemented in Python using the Pytorch library and was trained with VQA 2.0 dataset. The inputs to the generator were the questions and the output of an image. The discriminator took as input both images from the generator (fake) and dataset (real) and predicted the probability of real/fake. With the predicted probability, the sigmoid cross-entropy (detailed in Appendix 1) is determined. The sigmoid cross-entropy is the error metric that is used to train the GAN.

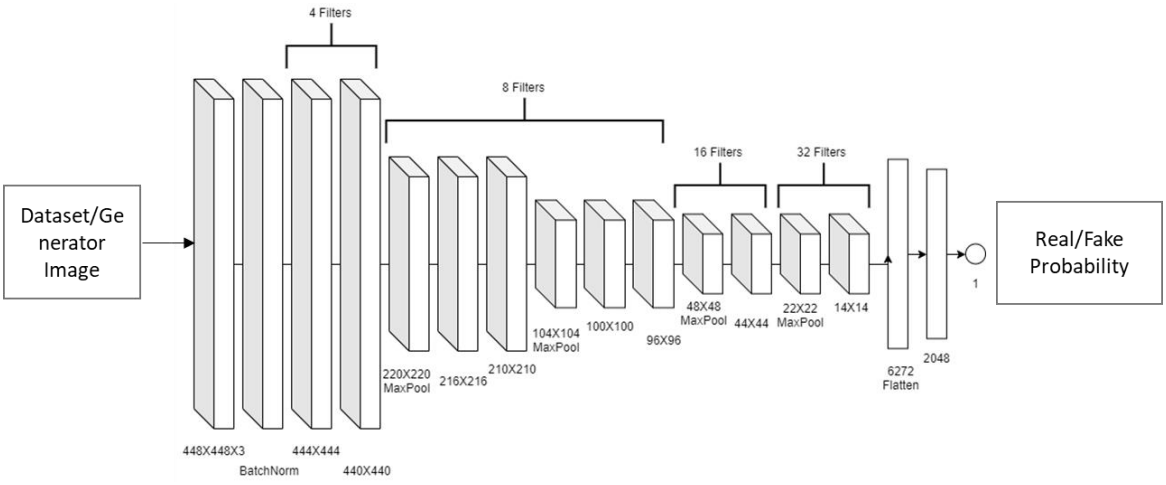

Figure 4 Network Architecture of DC-GAN Discriminator

3.1.2 Least Squares GAN (LS-GAN)

Vanilla GANs with sigmoid cross-entropy loss function have been observed (Mao, et al., 2016) to suffer from vanishing gradients (the case where neural network weights/parameters become zero), thus making the training unstable. A

Least Squares GAN utilizes a least-squares loss function for the discriminator. Least Squares GANs are shown to have improved stability in the learning process (Mao, et al., 2016).

Since, the VQA 2.0 dataset consists of a complementary image each image-question pair that has a different answer for the question with respect to the image, a modified version of the least-squares optimization function has been used for training the GAN. Triplet loss function (Cao, et al., 2017) for a data triplet, $t = (x^a, x^p, x^n)$, where x^a (is the anchor image) and x^p(a positive sample to anchor image) have similar samples and x^n(a negative sample to anchor image) is different from them, is defined as,

$$\sum_{i}^{N} \left[\left\| f(x_i^a) - f\left(x_i^p\right) \right\| - \left\| f(x_i^a) - f(x_i^n) \right\| + a \right]_+ \quad (3)$$

where *a* is the threshold, $[.]_+$ refers to $\max(., 0)$, *f* represents the non-linear activation function of the neural network (such as tanh, ReLU, etc.) and *i* represents a single sample from the dataset.

The triplet loss is an implementation of least squares loss and is used to map data with similar attributes to be close in embedding space, and data of different nature to be far from each other. Adapting it for the GAN setup, we have,

$$\min_G \max_D V(D, G) = E_{\substack{x^p, x^n \sim p_{data}(x) \\ z \sim p_g(z)}} [\|D(G(z)) - D(x^p)\| + \|D(G(z)) - D(x^n)\| + a] \quad (4)$$

Where E is the expectation value, p_{data} is the probability distribution function of the dataset and $p_g$is the probability distribution function of the generator. $G(z)$ differentiable function representation of input data space to output

mapping and z is the noise/lower-dimensional signal input. $D(x)$ is the probability of the image being a real data sample. Several implementations have successfully used Triplet Loss in GAN training, (Cao, et al., 2017), (Novoselov, Shchemelinin, Shulipa, Kozlov, & Kremnev, 2018).

Implementation Details

Network Architecture

The network is developed based on DC-GAN (Radford, Metz, & Chintala, 2016) architecture, like in the previous experiment. The generator consists of staggered transposed convolutional layers that perform the deconvolution process with an upscaling layer in between the filter banks. The input to the generator is the question embedding. The questions are embedded using Bag of Word encoding for the top 1000 frequent words in the training data, followed by a stacked (2 hidden layer) LSTM (Hochreiter & Schmidhuber, 1997). The convolutional layers are followed by a *Rectified Linear Unit (ReLU)* non-linearity. The final layer of the generator consists of *tanh* non-linearity. The architecture of the generator is given in figure [5]. The generator architecture is exactly the same as that of DC-GAN. The details of the network can be found in chapter 3.1.1

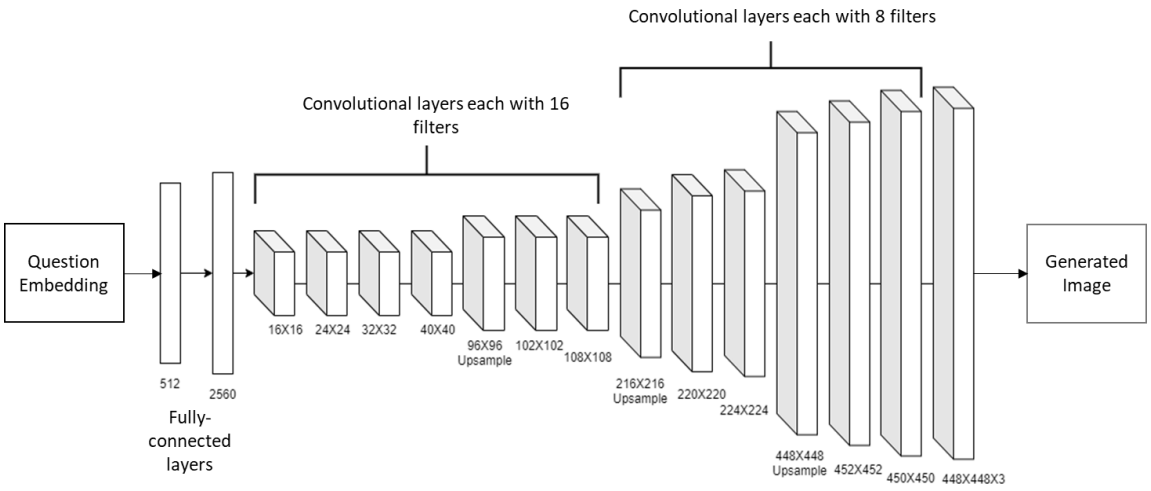

Figure 5 Network Architecture of LS-GAN Generator

The discriminator is also inspired by DC-GAN architecture. Which takes in the images (real/fake) and generates a lower-dimensional abstract embedding for the LS-GAN loss. The discriminator consists of multiple convolutional layers followed by the maximum pooling. The discriminator consists of dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) layers to prevent overfitting. The discriminator generates a 2048 dimensional vector representation of the images which are used by the triplet loss framework for optimization. The architecture of discriminator is given in figure [6]. The discriminator architecture is the same as that of DC-GAN except for the final layer. The details of the network can be found in chapter 3.1.1. The final layer of DC-GAN discriminator is removed, such that the discriminator doesn't predict the probability of the image being real/fake. The discriminator generator a 2048-dimensional vector representation of the image, which will be used to determine the least-squares triplet loss.

The complete LS-GAN architecture was implemented in Python using the Pytorch library and was trained with VQA 2.0 dataset. The inputs to the generator were the questions and the output of an image. The discriminator takes as input both image from the generator (fake) and dataset (real) and generates a lower-dimensional abstract embedding for the LS-GAN loss.

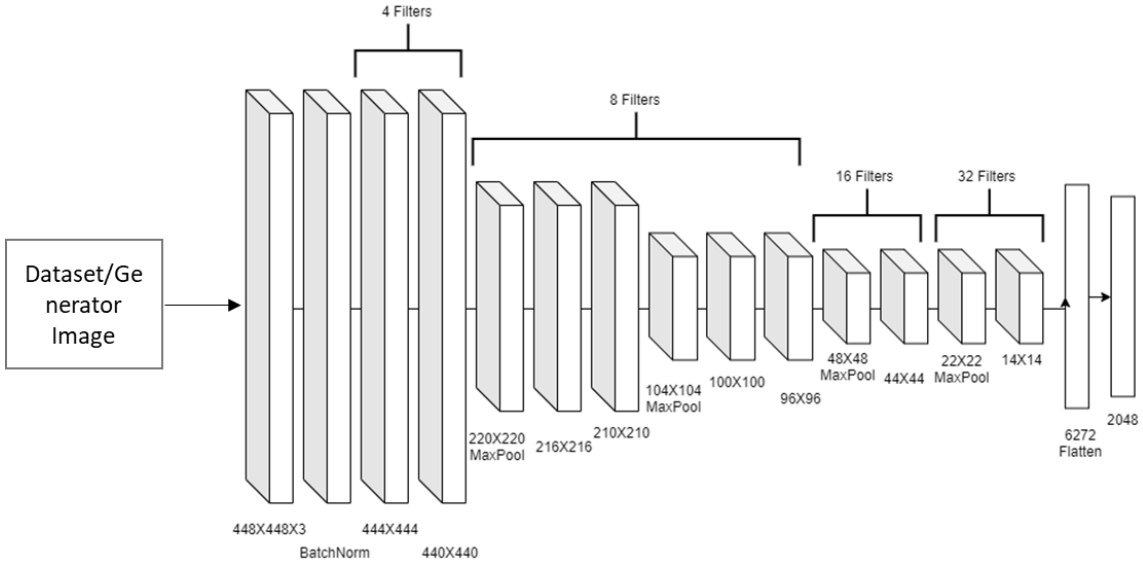

Figure 6 Network Architecture of LS-GAN Discriminator

3.1.3 Wasserstein GAN (W-GAN)

Wasserstein GAN (Gulrajani, Ahmed, Arjovsky, Dumoulin, & Courville, 2017) is an implementation of a GAN network which is trained with the objective function that represents the divergence between two probability distributions. Unlike other techniques that use Kullback-Leibler divergence, Wasserstein GANs minimizes an approximation of Earth Movers (EM) distance. This optimization technique means that WGANs does not require a careful balance in training the

discriminator and the generator, and does not require careful design of the network architecture either.

In cases where there are not any labels available for computing the sigmoid cross-entropy loss for the GANs, a divergence metric is used to quantify the similarity/difference between probability distributions. Contextually, for divergence metric is defined to quantify the similarity between training images and generated images. Several divergence metrics have been used in the past, with each one having its pros and cons. Let χ be a compact metric set ($[0,1]^d$ for images) and Σ denote set of all Borel subsets of χ. Some of the majorly used divergence metrics are between two distributions – generated image distribution $\mathbb{P}_g$ and training image distribution $\mathbb{P}_r$:

1. The Total Variation (TV) distance:

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = \sup_{A \in \Sigma} |\mathbb{P}_r(A) - \mathbb{P}_g(A)| \tag{5}$$

2. The Kullback-Leibler (KL) (Kullback & Leibler, 1951) divergence

$$KL(\mathbb{P}_r \parallel \mathbb{P}_g) = \int \log\left(\frac{P_r(x)}{P_g(x)}\right) P_r(x) d\mu(x) \tag{6}$$

where both $\mathbb{P}_r$ and $\mathbb{P}_g$ are assumed to be absolutely continuous, and therefore exhibits density, with respect to the measure μ defined on χ. P_g and $P_r$are probability being a generated image and a real image from the dataset, respectively, and x is a data sample. The KL divergence is asymmetric and possibly infinite when $P_g(x) = 0$ and $P_r(x) > 0$.

3. The Jensen-Shannon (JS) divergence

$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r \parallel \mathbb{P}_m) + KL(\mathbb{P}_g \parallel \mathbb{P}_m) \tag{7}$$

where $\mathbb{P}_m = \frac{(\mathbb{P}_g + \mathbb{P}_r)}{2}$. The JS divergence is symmetrical and is always defined when $\mu = \mathbb{P}_m$

4. The Earth-Mover (EM) distance or Wasserstein-1

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{(\gamma \in \prod(\mathbb{P}_r, \mathbb{P}_g))} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|] \quad (8)$$

where $\prod(\mathbb{P}_r, \mathbb{P}_g)$ is the set of joint distributions $\gamma(x, y)$ with marginals $\mathbb{P}_r$ and $\mathbb{P}_g$ respectively. In Earth Mover's context, $\gamma(x, y)$ represents how much "mass" must be transferred from x to y such that $\mathbb{P}_g$ has the same distribution as $\mathbb{P}_r$. The EM distance is known as the cost of optimal transport.

Several studies have determined that the convergence characteristics of EM distance are much better compared to other divergence metrics. The best trait of WGAN is the ability to train the critic/discriminator till optimality. The critic can be trained to completion, and it directly provides a loss to the generator that we can be trained as an isolated neural network. This eliminates the need to balance generator and discriminator's capacity properly. The generator training gradients improve with the quality of the discriminator.

On the flip side, training WGAN becomes unstable upon using a momentum-based optimizer such as Adam (Kingma & Ba, 2015) (with $\beta_1 > 0$) on the critic, or with high learning rates. Due to the nonstationary nature of the critic's loss function, momentum-based methods perform worse. WGANs work best with RMSProp optimization, which is known to perform well even on very nonstationary problems. Another drawback of WGAN is that the training suffers mode breakdown

due to Lipschitz constraint, resulting in exploding or vanishing gradients. Hence, as a remedial fix, the network parameters are clamped to be within a specified range (generally between [-0.01, 0.01]) after each gradient update.

Alternatively, the weight clipping is substituted with a gradient penalty (Gulrajani, Ahmed, Arjovsky, Dumoulin, & Courville, 2017). The weight clipping method requires careful and rigorous training setup to avoid mode collapse. The gradient penalty technique directly constrains the gradient norm of the critic's output for its input. A soft constraint is enforced using a penalty on the gradient norm to avoid tractability issues. Hence the updated objective function (*L*) becomes,

$$L = \underbrace{\left(\mathbb{E}_{z \sim \mathbb{P}_g}[D(G(z))] - \mathbb{E}_{r \sim \mathbb{P}_r}[D(x)]\right)}_{Original\ WGAN\ critic\ loss} + \underbrace{\left(\lambda \mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]\right)}_{Gradient\ penalty} \quad (9)$$

Where $\mathbb{E}$ is the expectation value, $\mathbb{P}_g$ is the probability distribution function of the generator. $\mathbb{P}_r$ is the probability distribution function of the dataset. λ is just a multiplier. ∇ is the gradient operator. $G(z)$ differentiable function representation of input data space to output mapping and z is the noise/lower-dimensional signal input. $D(x)$ is the probability of real data sample. The sampling distribution $\mathbb{P}_{\hat{x}}$ is defined along the straight lines connecting sampled pairs from distributions $\mathbb{P}_r$ and $\mathbb{P}_g$. $\hat{x}$ are samples from distribution $\mathbb{P}_{\hat{x}}$. This is due to the fact that optimal critic contains straight lines with gradient norm 1 connecting coupled points from $\mathbb{P}_r$ and $\mathbb{P}_g$.

Implementation Details

The network is developed based on DC-GAN (Radford, Metz, & Chintala, 2016) architecture, like in the previous experiment. The generator consists of staggered transposed convolutional layers that perform the deconvolution process with an upscaling layer in between the filter banks. The input to the generator is the question embedding. The questions are embedded using Bag of Word encoding for the top 1000 frequent words in the training data, followed by GRU (Cho, et al., 2014). The convolutional layers are followed by batch normalization and *Leaky Rectified Linear Unit (Leaky ReLU) (details in appendix I)* non-linearity. The final layer of the generator consists of *tanh* non-linearity. The architecture of the generator is given in figure [7]. In figure [7], the generator consists of 7 convolutional layers. The input question embeddings (256-dimensional vector) are concatenated with random Gaussian noise (with zero mean and standard deviation of 1) (256-dimensional vector) and passed through 7 convolutional layers (details in Appendix 1) (represented as 3-dimensional cuboid in the figure). The convolutional filters in each convolution are listed under each convolution layer in the figure. For example, the first convolutional layer has the value 4X4X96. It represents 96 convolutional filters, each with 4X4 hidden neurons. All convolutional layers are transposed to perform deconvolution. The final convolutional layer produces the image.

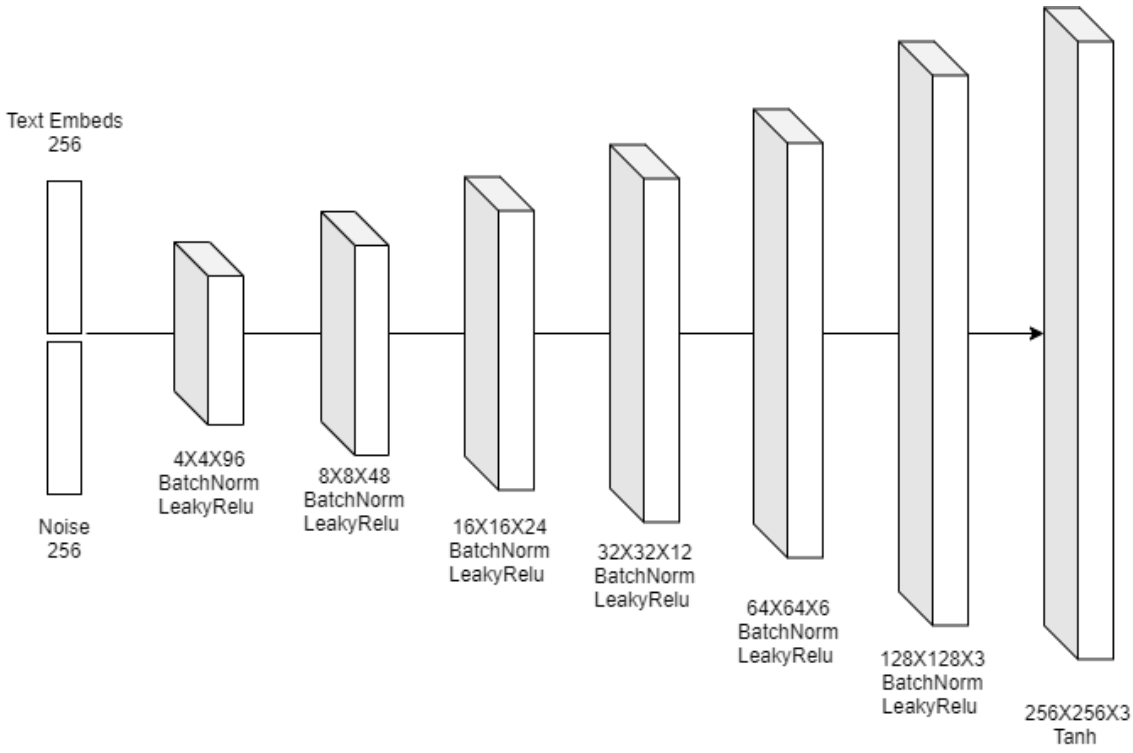

Figure 7 Network Architecture of W-GAN Generator

The discriminator is also inspired by DC-GAN architecture, which takes in the images (real/fake) and generates the probability of being real. The discriminator consists of multiple convolutional layers followed batch normalization and *Leaky Rectified Linear Unit (Leaky ReLU)* non-linearity. Based on the implementation of W-GANs, the pooling layers have replaced with strided convolutions. The architecture of discriminator is given in figure [8]. In figure [8], the discriminator consists of 7 convolutional layers. The input images (of 448X448X3 pixel values) are passed through 7 convolutional layers (details in Appendix 1) (represented as 3-dimensional cuboid in the figure). The convolutional filters in each convolution are listed under each convolution layer in the figure. For example, the first convolutional layer has the value 128X128X3. It represents three

convolutional filters, each with 128X128 hidden neurons. The final convolutional layer predicts the probability of input image being real (from the dataset) or fake (from the generator). The predicted probability is used to determine the Wasserstein loss error value.

The complete GAN architecture was implemented in Python using the Pytorch library and was trained with VQA 2.0 dataset. The inputs to the generator were the questions and the output of an image. The discriminator takes as input both image from the generator (fake) and dataset (real) and predicts the probability of real/fake.

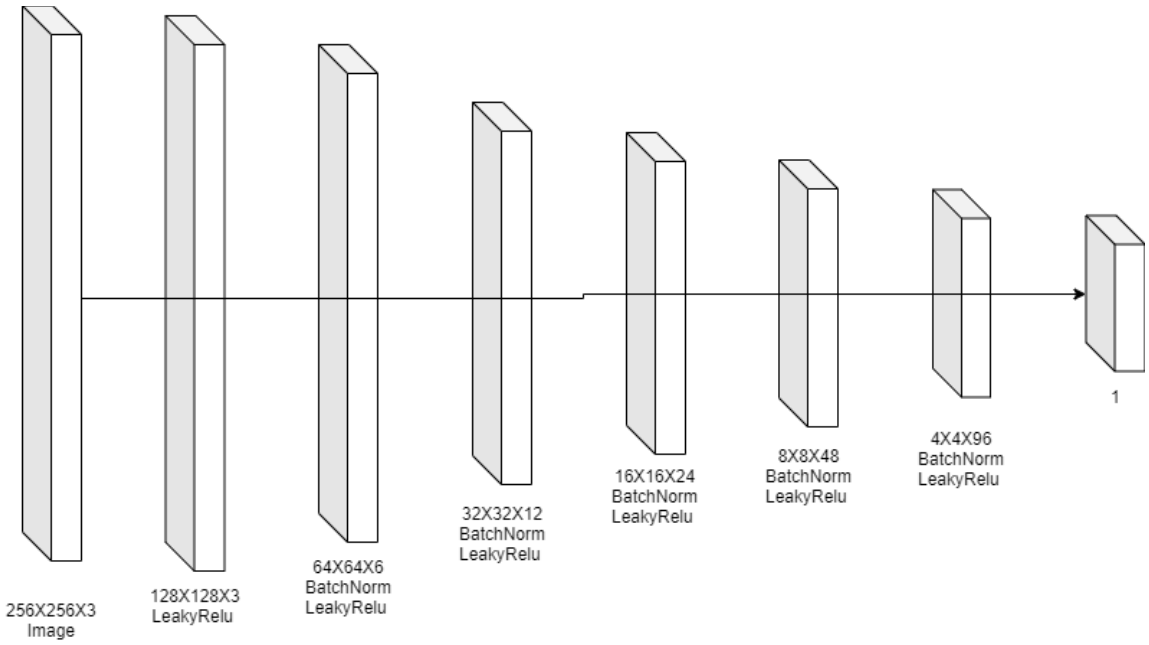

Figure 8 Network Architecture of W-GAN Discriminator

3.1.4 Attentional GAN (AttnGAN)

Attentional Generative Adversarial Network (AttnGAN) (Xu, et al., 2018) is an attention-driven, multi-stage refinement GAN architecture for a fine-grained

text-to-image generation. Attentional GAN can generate fine-grained details at different sub-regions of the image by paying attention to the relevant words in the natural language description. Figure [9] represents the network architecture of the AttnGAN. All the components of the network are detailed below.

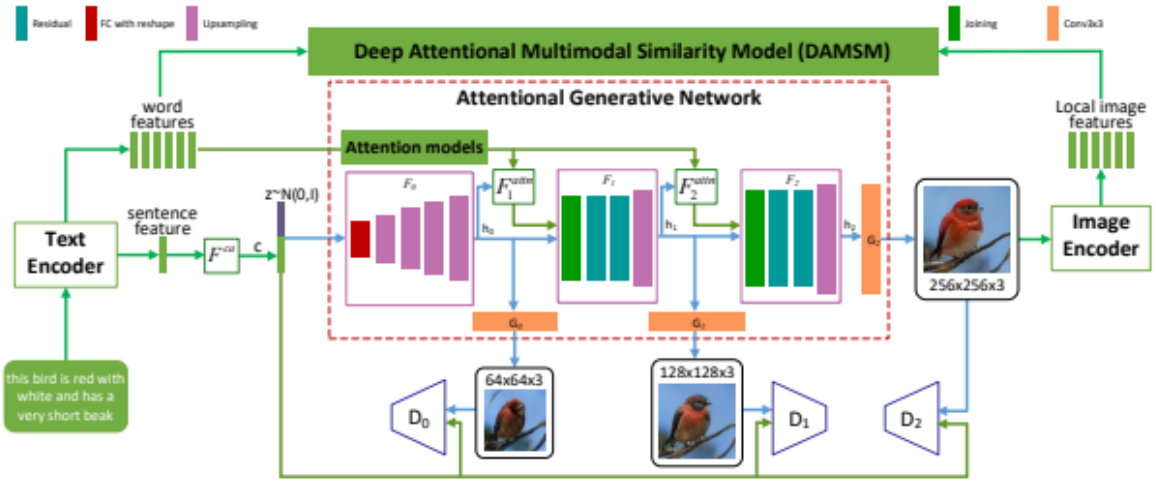

Image Source: (Xu, et al., 2018)

Figure 9 Network Architecture of AttnGAN

The AttnGan consists of an attentional generator with an attention mechanism to draw different sub-regions of the image by focusing on words that are most relevant to the sub-region being drawn. The text encoding is in two-fold- (1) global sentence encoding and (2) each word in the sentence is encoded into a word vector. The generative network utilizes the global sentence vector to generate a low-resolution image in the first stage. In the following stages, the image vector in each sub-region is combined with word vectors by using an attention layer to form a word-context vector. It then combines the regional image vector and the corresponding word-context vector to form a multimodal context vector, based on which the model generates new image features in the

surrounding sub-regions. This process is shown to produce higher quality images. This process of iterative attentional generative model is represented as,

$$h_0 = F_0\left(z, F^{ca}(\bar{e})\right) \tag{10}$$

$$h_i = F_i\left(h_{i-1}, F_i^{attn}(e, h_{i-1})\right) for\ i = 1, 2, .., m-1 \tag{11}$$

$$\widehat{x_i} = G_i(h_i) \tag{12}$$

The first pixelated image ($\widehat{x_0} = G_0(h_0)$ is generated, where G_0 represents the function form of first stage generator architecture and h_0 is the first hidden stage generated by global sentence vector $\bar{e}$ and noise vector z. F^{ca} is the Conditioning Augmentation that converts the sentence vector to the conditioning vectorF_i^{attn} is the attention network at the i[th] stage of the AttnGAN. F^{ca}, F_i^{attn}, F_i, and G_i are modeled as multi-layer perceptron networks. h_i are the successive hidden stages generated by the network.

The objective function ($\mathcal{L}$) for training the generator consists of two parts: a divergence metric and a multi-modal similarity metric. The objective function is defined as,

$$\mathcal{L} = \underbrace{\mathcal{L}_G}_{\substack{Divergence \\ metric}} + \underbrace{\lambda\, \mathcal{L}_{DAMSM}}_{\substack{multi-modal \\ similarity\ metric}} \tag{13}$$

Now $\mathcal{L}_G$ defined as

$$\mathcal{L}_G = \sum_{i=0}^{m-1} \underbrace{-\frac{1}{2}\mathbb{E}_{\widehat{x_i}\sim P_{G_i}}[\log(D_i(\widehat{x_i})]]}_{unconditional\ loss} \underbrace{-\frac{1}{2}\mathbb{E}_{\widehat{x_i}\sim P_{G_i}}[\log(D_i(\widehat{x_i})]]}_{conditional\ loss} \tag{14}$$

Where $\mathbb{E}$ is the expectation value, $p_{G_i}$is the probability distribution function of the generator G_i. G_i differentiable function representation of input data space to

output mapping. D_i is the probability of the image being a real data sample. At the i[th] stage of the AttnGAN, the generator G_i has a corresponding discriminator D_i. $\widehat{x_i}$ is the image from the generator. The unconditional loss determines whether the image is real or fake, while the conditional loss determines whether the image and the sentence match or not.

The multi-modal similarity metric $\mathcal{L}_{DAMSM}$, is a word-level fine-grained image-text matching loss computed by a Deep Attentional Multimodal Similarity Model (DAMSM). The DAMSM learns two neural networks that map sub-regions of the image and words of the sentence to a common semantic space, thus measures the image-text similarity at the word level to compute a fine-grained loss for image generation.

The first neural network is the text encoder, which is a bi-directional Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) that extracts semantic vectors from the text description. The two hidden states corresponding to each word of the bi-directional LSTM (Hochreiter & Schmidhuber, 1997) are concatenated to represent the semantic meaning of a word.

The second network is the image encoder, which is a Convolutional Neural Network (CNN) that maps images to semantic vectors. The image encoder is built upon the Inception-v3 (Szegedy, et al., 2015), (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2016) model pre-trained on ImageNet, that learns local features of image sub-regions in the intermediate layers and global image features in the later layers.

The attention-driven image-text matching score is designed to measure the matching of an image-sentence pair based on an attention model between the

image and the text. First, a *soft-max* normalized dot-product similarity between all possible pairs of words in the sentence and sub-regions in the image is calculated. Then, an attention model is built to compute a region-context vector for each word. The region-context vector, computed as the weighted sum over all regional visual vectors, produces a dynamic representation of the image's sub-regions related to the i^{th} word of the sentence. The DAMSM loss is designed to learn the attention model in a semi-supervised manner, in which the only supervision is the matching between entire images and whole sentences (a sequence of words).

Implementation Details

The pre-implemented architecture from the original authors of the work was utilized to re-train VQA 2.0 dataset. The inputs to the generator were the questions and the output of an image. The discriminator takes as input both image from the generator (fake) and dataset (real) and predicts the probability of real/fake.

3.2 Strategy II

This thesis also evaluates an alternative approach to attention network, using multi-modal compact bilinear pooling, in the existing VQA system. The state-of-the-art VQA system by (Anderson, et al., 2017) was utilized to realize the VQA system. However, the system was modified to incorporate the multi-modal compact bilinear pooling attention network.

The Python code for developing the network architecture for the VQA system with a modified attention network was developed in this thesis. Pytorch deep learning library was used in developing the network architecture in Python.

Also, The Python code necessary for training and evaluating the network is also developed in this thesis.

The VQA system was trained to optimize the network parameters of the architecture such that the error (defined in 3.2.4) is minimum. The image-question data pairs from the training pool of the VQA 2.0 dataset (details in chapter 2) are fed as inputs. The error for the VQA system is identified and is minimized by the gradient descent optimizer (definition in appendix 1) technique. In this work, the VQA system training is ended when the training error is unchanged for more than 2 training iterations.

The validation of the VQA system consists of feeding the image-question data pairs from the validation pool of the VQA 2.0 dataset and tracking the VQA error. The important thing to note is that the network is optimized in anyway during validation. The central idea behind validation is to check that the VQA system error doesn't deviate by a large amount for new data.

Figure [11] represents the network architecture of the state-of-the-art VQA system (unmodified) (Anderson, et al., 2017). The individual network components of the VQA system (image encoding, question encoding, attention network, and output classifier) are detailed in chapters 3.2.1 to 3.2.4

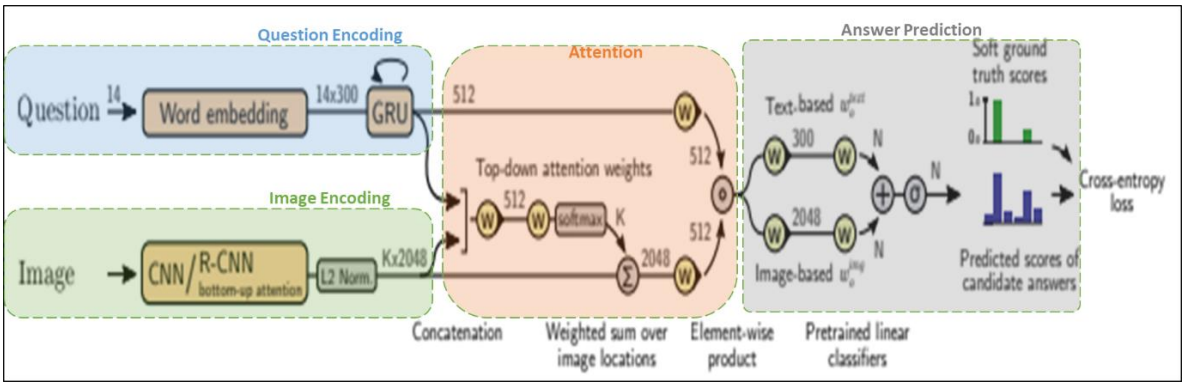

Image Source: (Anderson, et al., 2017)

Figure 10 Original Bottom-up Attention Network Architecture

3.2.1 Image Encoding

As detailed in chapter 1.2, predominantly, VQA implementations have used Top-down visual encoding mechanisms. A bottom-up mechanism proposes focus regions within the image, each with an associated feature vector. The top-down mechanism leads to a uniform grid of equally sized and shaped neural receptive fields – irrespective of the content of the image. Objects and other salient image regions are shown to improve the attention network, in turn aiding in the generation of human-like captions and question answers.

A combined bottom-up and top-down visual encoding mechanism is observed to boost the overall performance of the attention network (Anderson, et al., 2017). The bottom-up mechanism proposes a set of salient image regions represented as feature vectors.

Given an image I, the bottom-up mechanism provides a variably-sized set of k image features, $V = \{v_1, \dots, v_k\}, v_i \in R^D$, such that each image feature encodes

a salient region of the image VQA model takes as input. The top-down attention consists of one-pass attention mechanisms.

The image features of the bottom-up attention mechanism are represented in terms of bounding boxes using Faster R-CNN (Ren, He, Girshick, & Sun, 2015) architecture. Faster R-CNN is an object detection model designed to identify instances of objects belonging to certain classes and localize them with bounding boxes. While other object localization techniques can be used, the trade-off between accurate region proposals and computation complexity makes Faster R-CNN the ideal candidate.

Faster R-CNN consists of a training scheme that alternates between fine-tuning for the region proposal task and then fine-tuning for object detection while keeping the proposals fixed. Faster R-CNN detects objects in two stages.

The first stage of Faster R-CNN is a Region Proposal Network (RPN), which generates object proposals. It takes an image input and outputs a set of refined rectangular bounding box object proposals, each with a class-agnostic objectness score. Using greedy non-maximum suppression with an intersection-over-union (IoU) threshold, the top box proposals are selected as input to the second stage.

In the second stage, region of interest (RoI) pooling is used to extract a small feature map for each box proposal. These feature maps are then batched together as input to the final layers of the CNN. The final output of the model consists of a *soft-max* distribution over class labels and class-specific bounding box refinements for each box proposal.

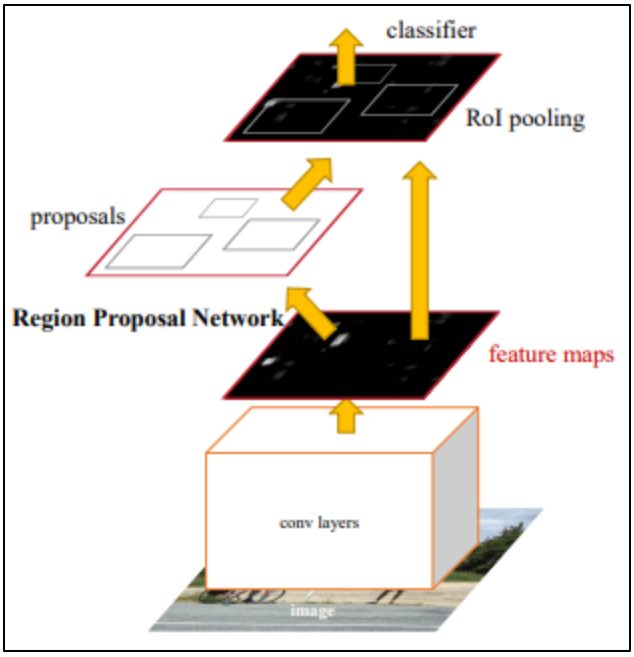

Image Source: (Ren, He, Girshick, & Sun, 2015)

Figure 11 Faster R-CNN Network Pipeline

While several architectures have been proposed to generate region proposals and object recognition, (Girshick, Donahue, Darrell, & Malik, 2014), (He, Gkioxari, Dollar, & Girshick, 2018), Faster R-CNN seems to perform best considering the computation overhead vs accurate proposals (Anderson, et al., 2017). In the bottom-up attention model, Faster R-CNN is used in conjunction with the ResNet-101 (He, Zhang, Ren, & Sun, 2016) CNN. A convolution hyper-space encoding of the image from intermediate convolutional layers of ResNet-101 is fed to the Faster R-CNN network.

3.2.2 Text Encoding

Primarily, the input questions are tokenized by splitting into words using spaces and punctuation. Since only 0.25% of the questions have a length greater than 14 words, questions are trimmed to a maximum of 14 words for computational efficiency. The words are turned into a 300-dimensional embedding vector learned during training. The vectors are however initialized with pre-trained GloVe word embedding (Global Vectors for Word Representation) (Pennington, Socher, & Manning, 2014). The questions shorter than 14 words are end-padded with vectors of zeros. The resulting sequence of word embedding is of size 14 × 300, and it is followed by a recurrent neural network. The recurrent network has an internal state of dimension 512, and its final state is used as the question embedding q.

3.2.3 Attention Network

In the original work by (Anderson, et al., 2017) , several attention networks were used for fusing the information gathered from question encoding and image encoding. This was done to determine the attention network that provides the best accuracy.

1. The representations of the question (q) and of the image (v) are passed through non-linear layers (tanh, ReLU, etc.) ($f_q \; and \; f_v$) and then combined with a simple Hadamard product (i.e., element-wise multiplication)

$$h = f_q(q) \circ f_v(v) \tag{15}$$

2. The question and image representations are concatenated together after passing through a linear layer that ensures dimensional similarity

$$h = \left(f_q(q)\middle|f_v(v)\right) \tag{16}$$

3.2.4 Output classifier

A set of candidate answers is predetermined from all the correct answers in the training set that appear more than 8 times. This amounts to N=3129 candidate answers. The VQA model is trained as a multi-label classification task. Each training question in the VQA v2 dataset is associated with one or several answers, each labeled with soft accuracies in [0, 1]. Multiple answers and accuracies in (0, 1) arise in case of disagreement between human annotators, particularly with ambiguous questions and multiple or synonymous correct answers. The answer prediction is the result of passing the joint embedding *h* is through a non-linear layer f_o and then through a linear mapping w_o predict a score *s* for each of the N candidate answers.

$$s = \sigma\left(w_o . f_o(h)\right) \tag{17}$$

where σ is the sigmoid logistic function and s gives the probability of N candidate answers being the answer for the pair as output. The error for the VQA system is determined as the cross-entropy error (details in appendix 1) of the predicted answers.

3.3 Modified VQA system

Figure [12], represents the modified architecture for VQA proposed in this (our) implementation. The VQA system consists of a modified attention network (multi-modal compact bilinear pooling). The rest of the components of the VQA system remain the same.

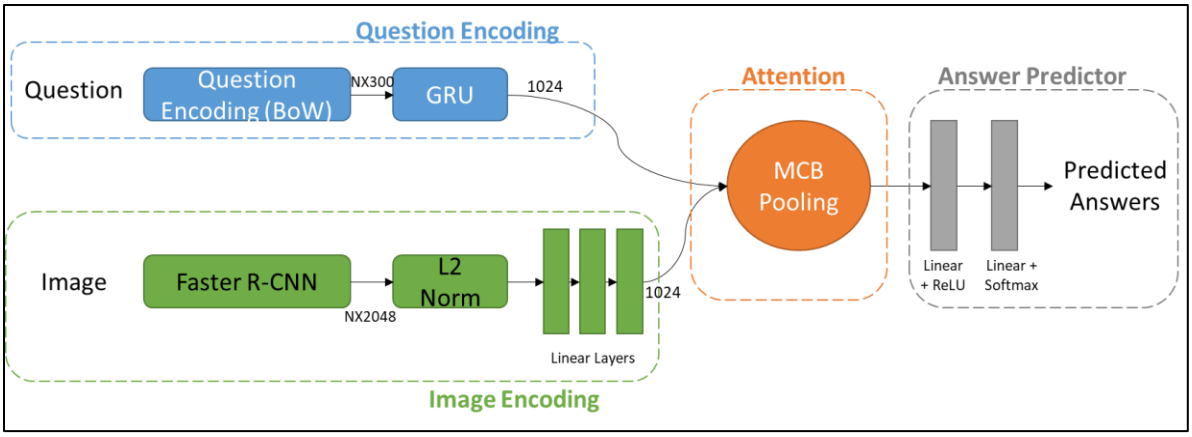

Figure 12 Network Architecture for current (our) VQA implementation

Modified Attention Network:

The implementation (Fukui, et al., 2016) proposes "outer product" of vectors for complete feature extraction, to overcome the myopic feature extraction strategies like element-wise multiplication or concatenation of image (x) and question (q) encoding for attention mechanism. However, since the outer product is computation-intensive and extracting features from an outer product requires large dimensional weight matrices, the outer product is achieved with the help of the Count Sketch projection (Ψ) method. It consists of two vectors $s \in \{-1, 1\}^n$ and h $\in \{1, .., d\}^n$. This methods projects vector x $\in \mathbb{R}^n$ on y $\in \mathbb{R}^d$ on a lower dimensional space by adding s.x to y. The computational complexity of the problem is further reduced by Pham and Phag's theorem of expressing count sketch of the outer product as convolution. Hence, the outer product becomes $FFT^{-1}(FFT(\Psi(x)) \odot FFT(\Psi(q)))$, simplifying back to element-wise multiplication. Furthermore, soft attention is applied by adding two convolutional layers, followed by the *soft-max* layer, after multi-modal compact bilinear pooling.

Apart from the alternative multi-modal fusion logic, the implementation deviates from the original implementation of the bottom-up and top-down attention model, in that, this uses only a fixed number of objects per image (K=36) for the R-CNN proposals. It uses a simple, single-stream classifier without pre-training. A simple *ReLU (details in appendix I)* activation is used instead of gated *tanh*. This significantly reduced the training time. Along with it other modifications such as adding dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) to alleviate overfitting, adding weight normalization instead of batch normalization and switching to Adamax (Kingma & Ba, 2015) optimizer with Gradient clipping enabled much efficient network performance. The modification is based on a supplemental paper (Teney, Anderson, He, & Hengel, 2018) from the bottom-up mechanism. The architecture is implemented using Torch (Paszke, et al., 2017), with pre-trained R-CNN on MS-COCO dataset.

Chapter 4: Results

This chapter discusses the results of the two strategies taken in this thesis. In chapter 4.1, the results from the strategy I are discussed. The results from strategy II are discussed in chapter 4.2.

4.1 Strategy I

A representation learning approach to improve the **image encoding system** of VQA is evaluated in strategy I. As discussed in chapter 3.1, four variants of GANs are evaluated in this approach. The results from these approaches are discussed in chapter 4.1.1 through 4.1.4.

A general desired trend of discriminator and generator losses (of GANs) over training steps is depicted in figures [13] & [14], respectively. Figure [13] shows that it is desired for the discriminator loss to increase initially and saturate over training steps. Figure [14] shows that the generator loss is desired to decrease initially and saturate over the training steps. Some of the other characteristics of generator and discriminator losses, depicted in the figures [13] & [14] are:

1. Both generator and discriminator loss, never become 0. A GAN becomes unstable, if generator or discriminator loss becomes zero.
2. The rate at which the losses increase/decrease is nearly same between generator and discriminator loss. A large gap in rate of change of losses between generator and discriminator, is a sign of instability in GAN training.

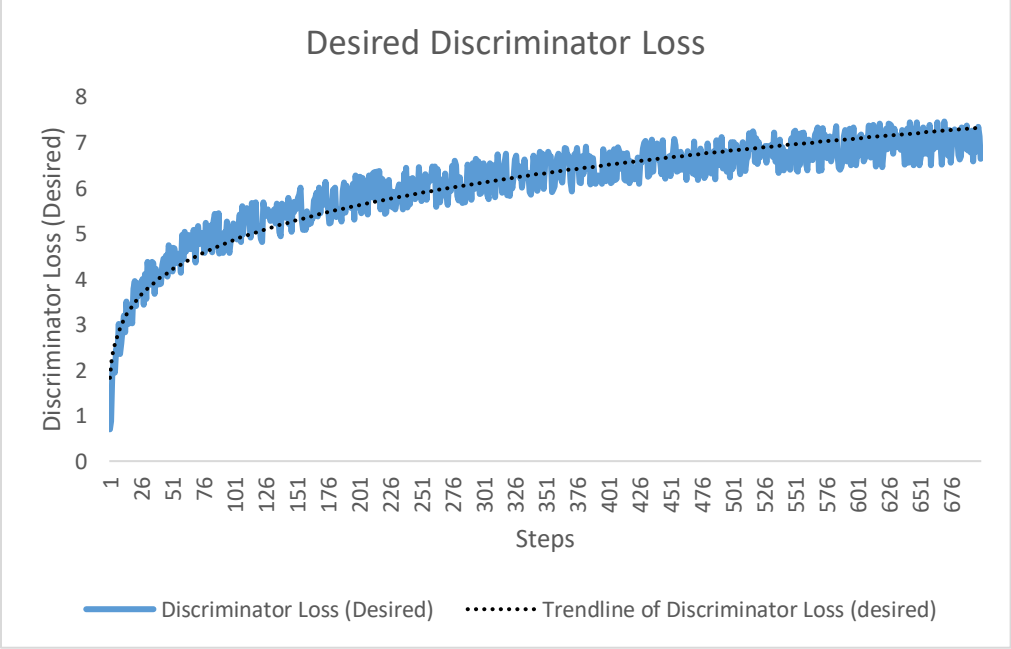

Figure 13 Desired Discriminator loss in GAN training

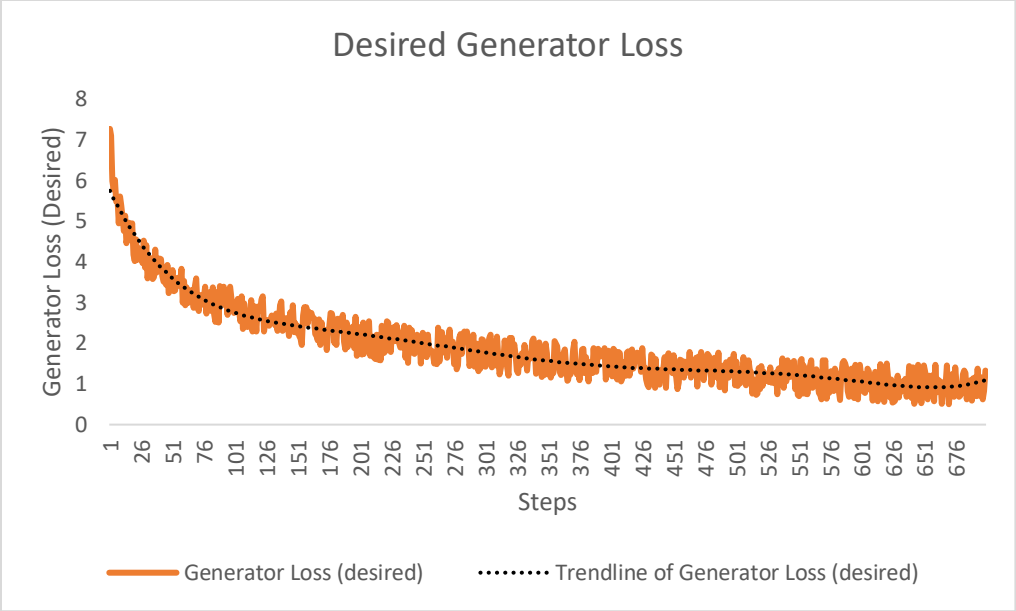

Figure 14 Desired Generator loss in GAN training

4.1.1 Deep Convolutional GAN (DC-GAN)

The training is based on gradient descent optimizer (details in appendix 1) for both generator and discriminator. The gradient descent optimizer is chosen primarily based on the fact that several studies have indicated that GANs become unstable upon using averaging optimizers like RMSprop, ADAM (Kingma & Ba, 2015), etc. The networks are trained with a learning rate of 1e-3 and for 5 epochs with a batch size of 32, totaling to ~30,000 runs. The training plots indicating the loss development for discriminator and generator are shown in figure [15] and [16], respectively.

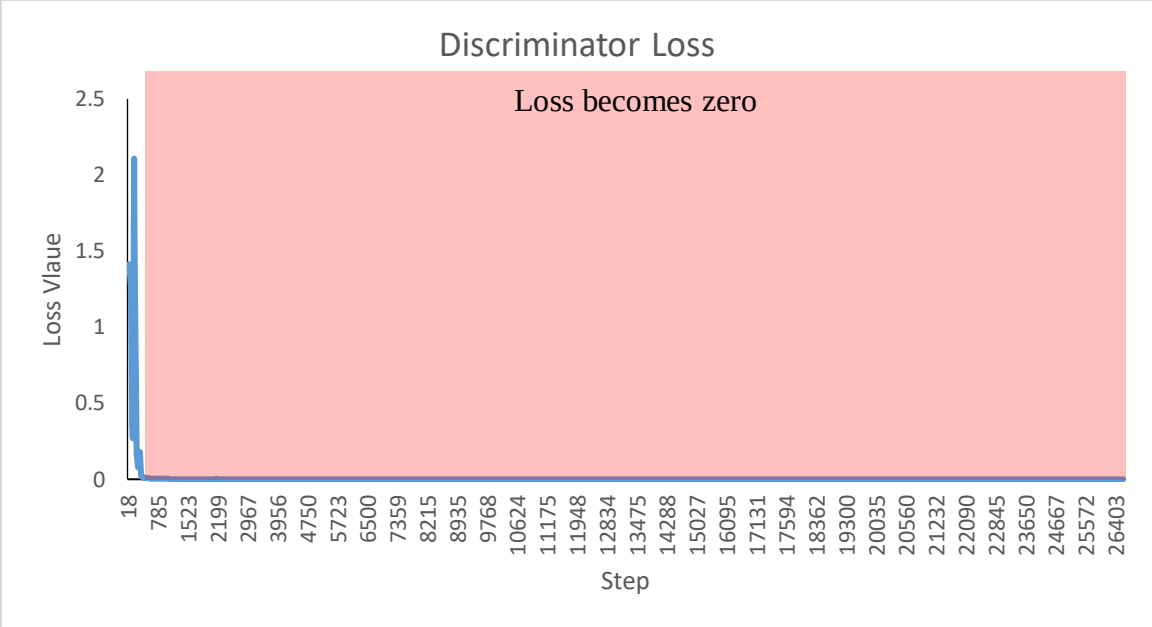

Figure 15 Training Loss Plot of DC-GAN Discriminator

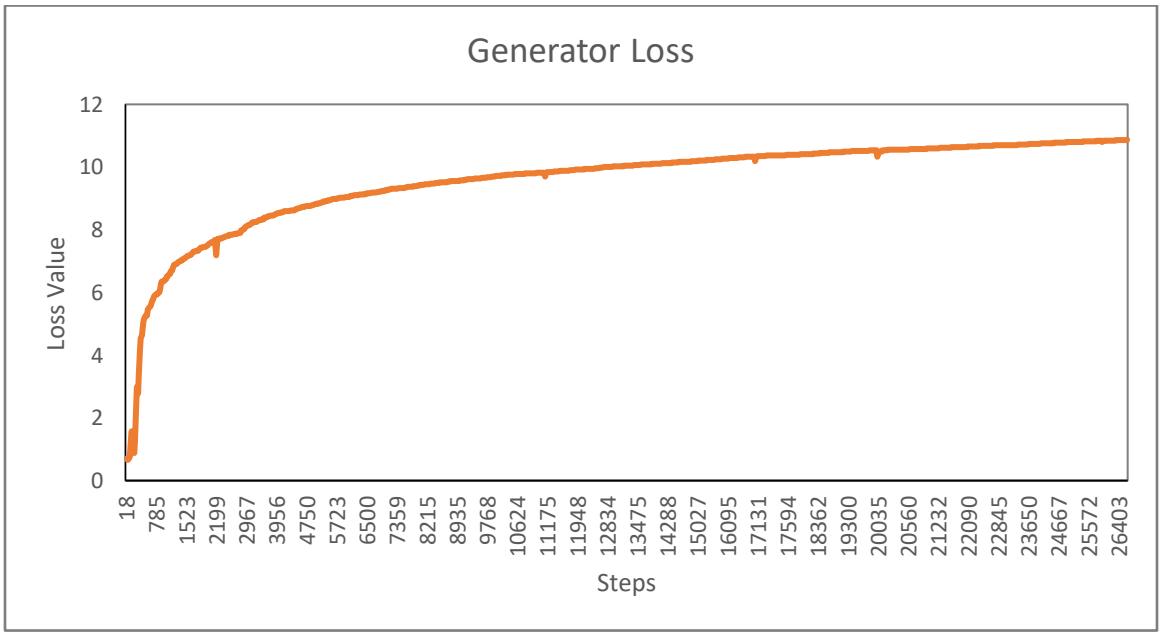

Figure 16 Training Loss Plot of DC-GAN Generator

Based on figures 15 and 16, the architecture quickly became unstable since the discriminator loss became zero at very early stages of training (indicated by the red region in figure 15). This means that the discriminator was perfectly able

to differentiate real vs. fake images. This led to the generator not having any information to learn from, which can be seen from the steady increase in generator loss. Thus, the equilibrium could not be maintained by this architecture.

4.1.2 Least Squares GAN (LS-GAN)

The training is based on gradient descent optimizer for both generator and discriminator. The gradient descent optimizer is chosen primarily based on the fact that several studies have indicated that GANs become unstable upon using averaging optimizers like RMSprop, ADAM (Kingma & Ba, 2015), etc. The network was implemented and trained using TensorFlow (Abadi, et al., 2015) and Keras. The networks are trained with a learning rate of 1e-3 and for 10 epochs with a batch size of 32, totaling to ~50,000 runs. The training plots indicating the loss development for discriminator and generator are shown in figure [17] and [18], respectively.

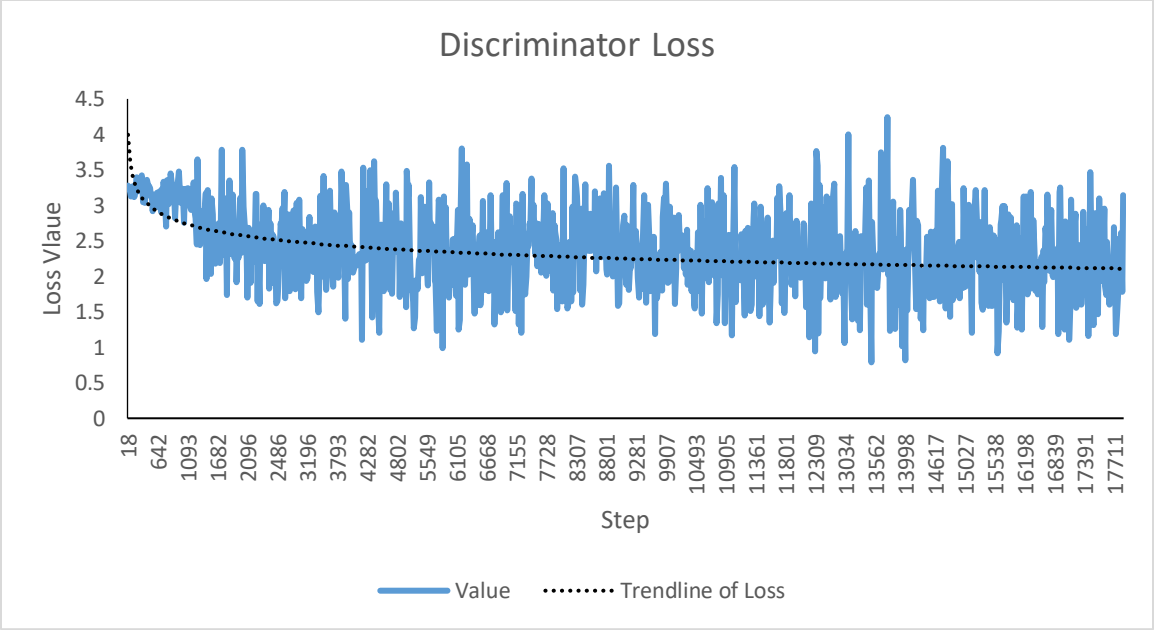

Figure 17 Training Loss Plot of LS-GAN Discriminator

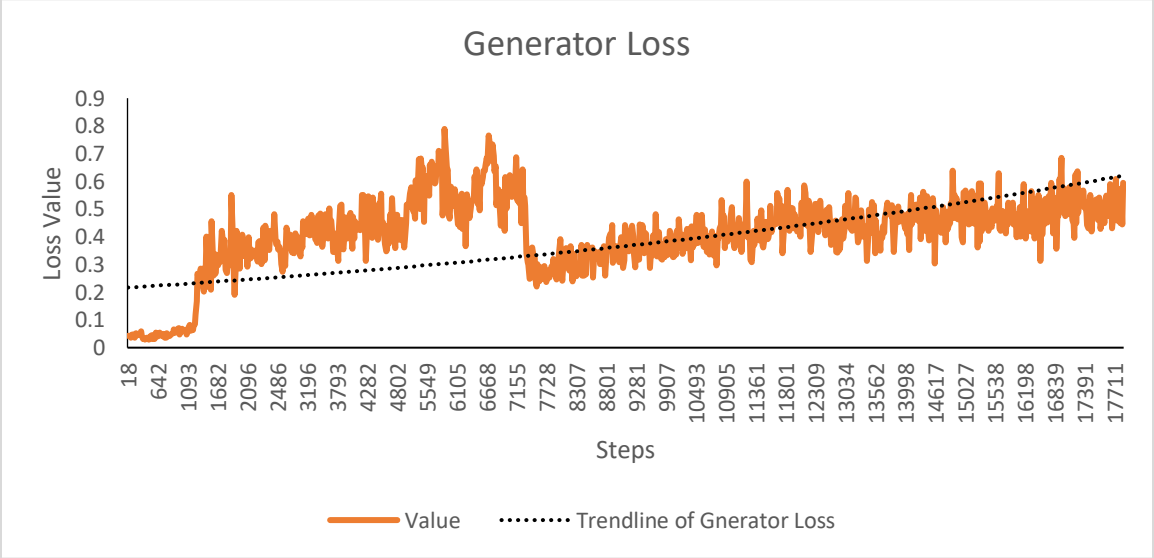

Figure 18 Training Loss Plot of LS-GAN Generator

Based on figures 17 and 18, the architecture is unstable, since the discriminator loss saturates eventually, indicated by the trend line in the graph. This means that the discriminator weights saturated, leading the gradients in successive steps are tending to zero. This led to the generator not having any information to learn from,

which can be seen from the steady increase in generator loss. Thus, the equilibrium could not be maintained by this architecture.

4.1.3 Wasserstein GAN (W-GAN)

The training is based on ADAM (Kingma & Ba, 2015) optimizer for both generator and discriminator. The networks are trained with a learning rate of 2e-4 and beta1=0.5 and beta2 = 0.999 for the ADAM optimizer. The network is for 30 epochs with a batch size of 128. The training plots indicating the loss development for generator and discriminator are shown in figure [19] and [20], respectively.

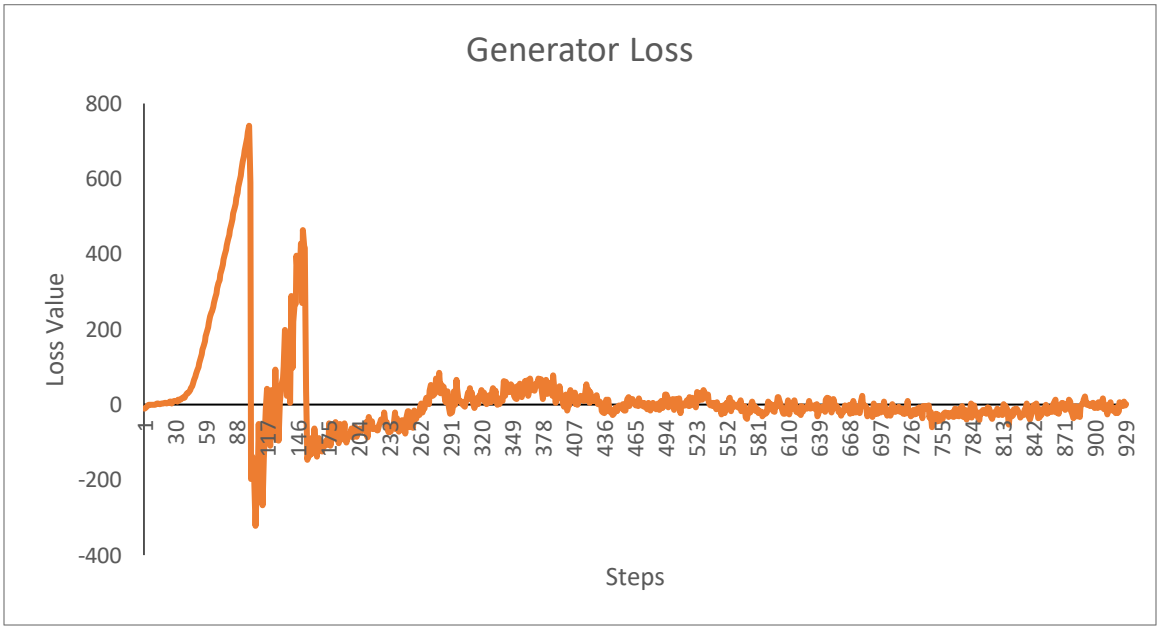

Figure 19 Training Loss Plot of W-GAN Generator

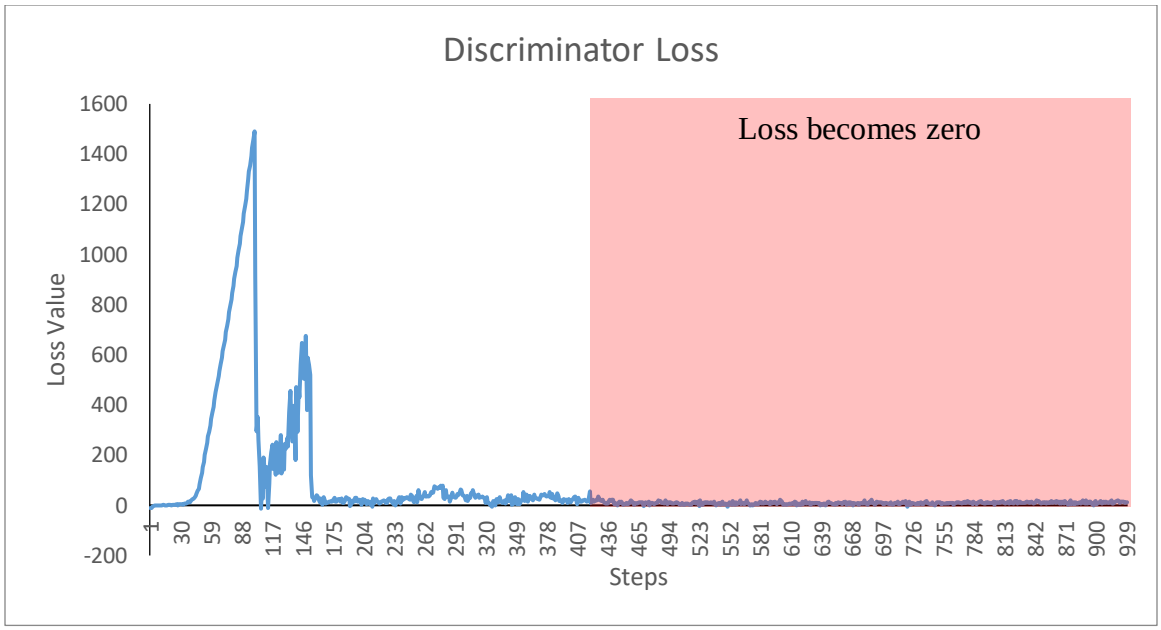

Figure 20 Training Loss Plot of W-GAN Discriminator

Based on figures 19 and 20, the architecture quickly became unstable, since the discriminator loss became zero at very early stages of training. This means that the discriminator was perfectly able to differentiate real vs. fake images. This led to the generator not having any information to learn from, which can be seen from the steady increase in generator loss. Thus, the equilibrium could not be maintained by this architecture.

4.1.4 Attentional GAN (AttnGAN)

The training is based on ADAM (Kingma & Ba, 2015) optimizer (a variant of gradient descent) for both generator and discriminator. The networks are trained with a learning rate of 2e-4 and beta1=0.5 and beta2 = 0.999 for the ADAM optimizer. The network is for 10 epochs with a batch size of 16. All other network parameters are retained from the original implementation of the network for image generation from captions on COCO (Lin, et al., 2014) dataset. The training plots

indicating the loss development for discriminator and generator are shown in figure [21] and [22], respectively.

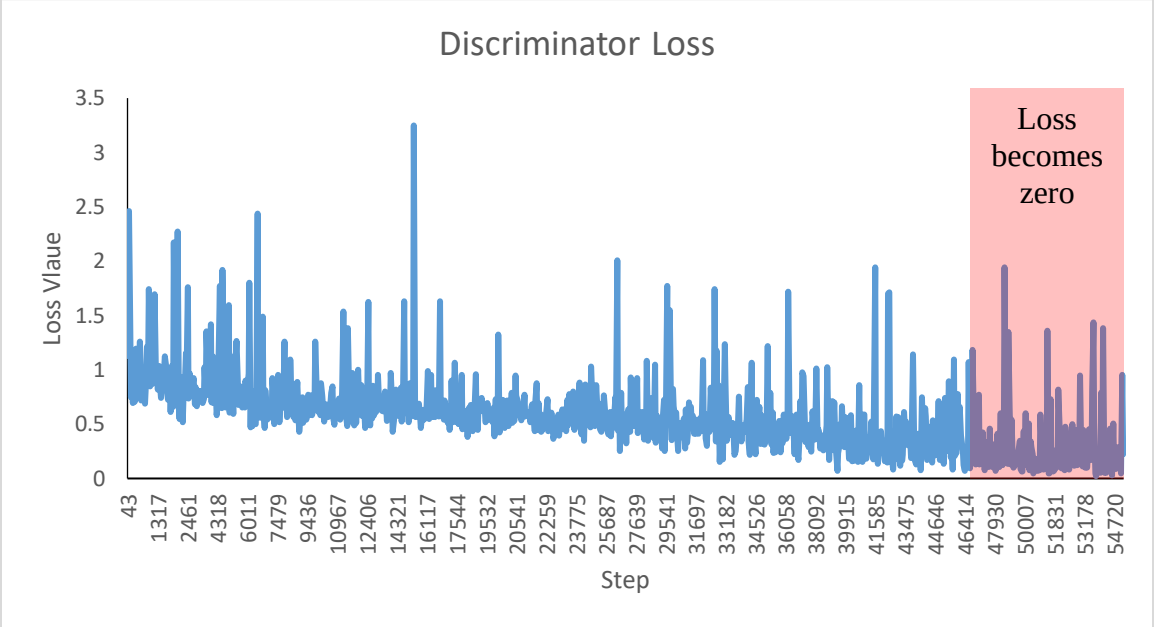

Figure 21 Training Loss Plot of AttnGAN Discriminator

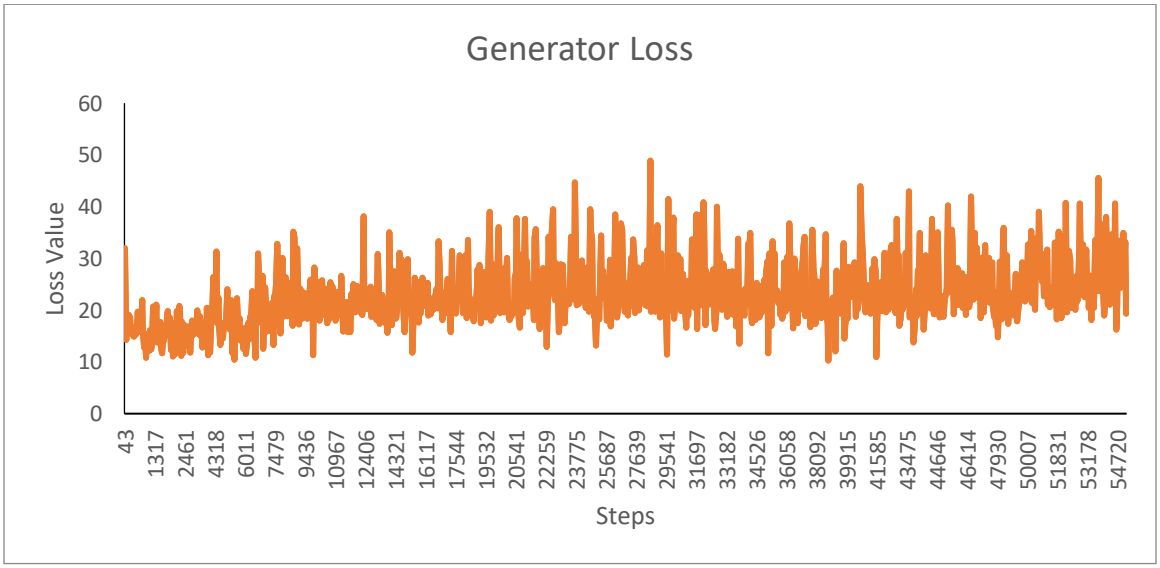

Figure 22 Training Loss Plot of AttnGAN Generator

Based on figures 21 and 22, the architecture became unstable eventually, since the discriminator loss became zero (indicated in figure 21). This means that

the discriminator was perfectly able to differentiate real vs. fake images. This led to the generator not having any information to learn from, which can be seen from the steady increase in generator loss. Thus, the equilibrium could not be maintained by this architecture.

Based on the results from the strategy I, all the evaluated architectures become unstable and do not converge, making the approaches unwieldy to be used as a replacement for the VQA image encoding network.

4.2 Strategy II

This thesis also evaluated multi-modal compact bilinear (MCB) pooling as an alternative approach to the attention network in the existing VQA system. This chapter discusses the results of the approach taken in strategy II.

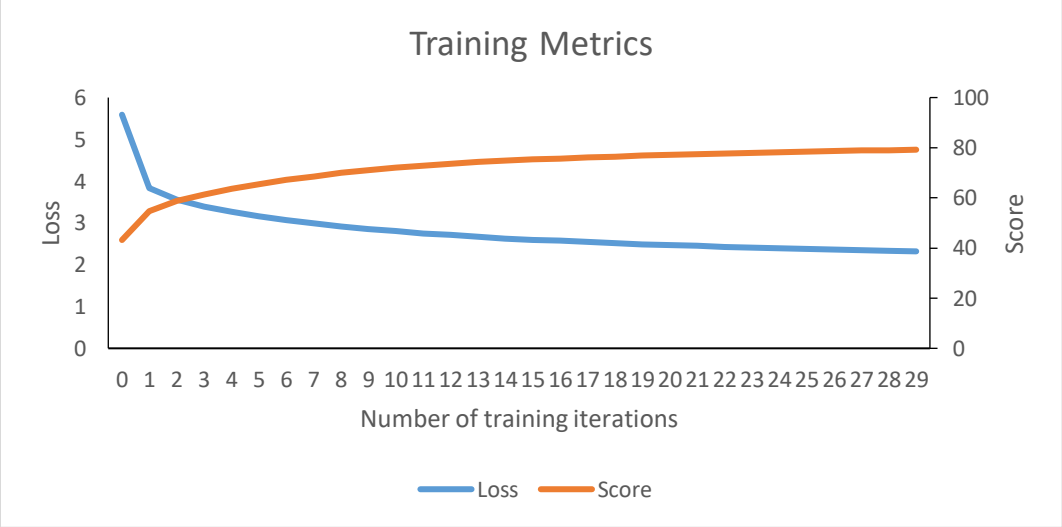

Figure 23 Training Loss and Accuracy Plot for these implementations

The figure [23] consists of training iterations (epochs) along X-axis. The primary Y-axis (left side) consists of cross-entropy loss (Loss) defined in chapter

3.2.4, and the secondary Y-axis consists of accuracy/score defined in equation (1) in chapter 1.2. The network converged much faster (30 epochs) than all other implementations previously mentioned (taking at least 1000 epochs to converge), as indicated in figure [23]. The usage of MCBs enabled both faster interference and an increase in validation accuracy to **65.32%** as opposed to **63.2% in the original bottom-up attention implementation** (Anderson, et al., 2017).

The accuracy (score) is defined as,

$$Score\ (of\ predicted\ answer) = min\left\{\frac{\#humans\ predicting\ the\ same\ answer}{3}, 1\right\}$$

Hence it can be concluded that the bottom-up attention mechanism for VQA performs much better with the multi-modal bilinear pooling technique as the attention mechanism to combine image and question information.

Chapter 5: Conclusions and Future Work

This thesis followed two strategies to improve the performance (accuracy) of VQA. The first is a representation learning approach to improve the image encoding system of VQA. This thesis evaluated four variants of GANs to identify a GAN architecture that works best for the representation learning approach. It was determined that GANs under consideration become unstable and fail to become a viable image encoding system in VQA. Given this, an alternative approach to attention network, using multi-modal compact bilinear pooling, was developed and evaluated. It led to an increase in the accuracy of VQA by 2% compared to the current state-of-the-art technique.

Although the VQA performance from strategy II is much better than the earlier efforts, deep learning inherently has the disadvantage of inexplicability. The decision process of predicting an answer for a given image-question by a deep learning network is mathematically too complex to be critically analyzed. While the primary focus of this thesis is improving the VQA performance based on the existing VQA system. Two alternative approaches that use a combination of deep learning and Bayesian networks, pivot towards a VQA system that can better explain its decision (predictions) are:

1. A scene description graph (SDG) (Aditya, Yang, Baral, Aloimonos, & Fermüller, Image Understanding using vision and reasoning through Scene Description Graph, 2017) , which can be tuned to perform the necessary task.
2. Probabilistic Soft Logic (PSL) to generate knowledge base from the image (Aditya, Yang, Baral, & Aloimonos, Combining Knowledge and Reasoning through Probabilistic Soft Logic for Image Puzzle Solving, 2018) that predicts the answer using a set of weighted if-then rules in first-order logic.

Leveraging SDG/PSL to answer a VQA question requires the development of a sophisticated probabilistic logical mechanism that can sift through the noise generated in the knowledge base and understand the natural language question and give an answer. Due to lack of extensive research, the current implementations of approaches 1 & 2 (mentioned above) are incapable of generalizing the question answering task even though they generate a robust

understanding of the image. The future goal of this thesis is to explore both the approaches 1 & 2 to develop a better performing VQA system. A detailed description of the above-mentioned methods have been detailed in appendix III.

Following the approaches 1 & 2 (mentioned above), the Visual Question Answering system will be able to achieve its primary goal (i.e., to develop a system guaranteed of learning multi-disciplinary tasks). These approaches pivot towards a system that generates a descriptive representation of an image, rather than forcing the system to generate/select the most relevant answer based on similarity.

Appendix I: Definitions

1. Tanh – a non-linear function used as the activation function in neural networks. It binds the input data within the range of [-1, 1]. It is close to a sigmoid curve.

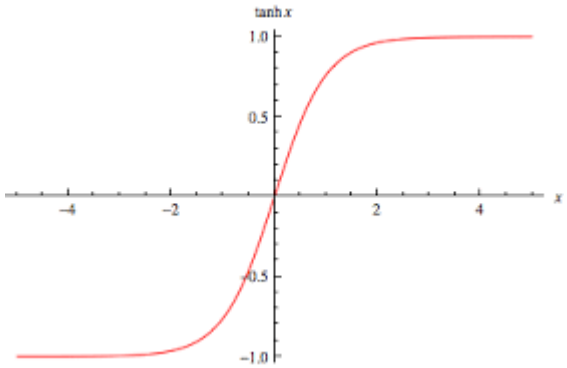

Figure 24 Tanh curve (courtesy: Wolfrom Math)

2. ReLU – Short for Rectified Linear Unit. It is also a non-linear function used in neural networks. Consists of the formula,

$$f(x) = \max(0, x)$$

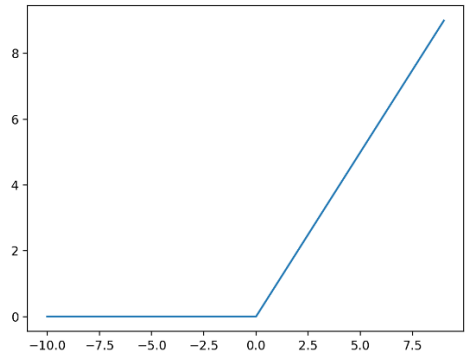

Figure 25 ReLU curve (courtesy: MachineLearningMastery)

3. Leaky ReLU – Variant of ReLU. Leaky ReLUs allow a small, positive gradient when the unit is not active.

$$f(x) = \begin{cases} x \ \ if\ x > 0 \\ 0.01x \ \ otherwise \end{cases}$$

4. Soft-max – Also an activation for neural networks. It is a function that takes as input a vector of K real numbers and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers.
5. Sigmoid – It is a mathematical function having a characteristic "S"-shaped curve or sigmoid curve. It is a standard activation function used in many machine learning algorithms. The sigmoid function is defined as,

$$Sigmoid\ or\ \sigma = \frac{1}{1 + e^{-x}}$$

6. N-grams – It is a contiguous sequence of n items from a given sample of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application. The n-grams typically are collected from a text or speech corpus.
7. Gradient descent optimizer – Gradient descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient. In machine learning, we use gradient descent to update the parameters of our model. Parameters refer to coefficients in Linear Regression and weights in neural networks.
8. Fully connected layer – fully connected networks are multilayer (at least 2 layers) artificial neural networks in which each neuron in one layer is connected to all neurons in the next layer.
9. Convolutional layers – A convolutional layer contains a set of filters whose parameters need to be learned. The height and weight of the filters are smaller than those of the input volume. Each filter is convolved with the input volume to compute an activation map made of neurons. In other words, the filter is slid across the width and height of the input and the dot products between the input and filter are computed at every spatial position.
10. Convolutional filters – A convolutional filter is a unit of a convolutional layer in Convolutional neural networks. The convolutional filter is a set of learnable parameters, that are slid over the input of convolutional layer and determine the dot product output. A set of convolutional filters make up a convolutional layer. An example for convolutional filter is:

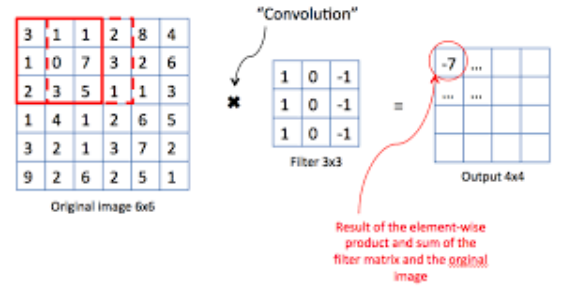

Image source: (medium.com)

Figure 266 Convolutional Filter

11. Cross-entropy – Cross-entropy is a measure of the difference between two probability distributions for a given random variable or set of events.

Appendix II: Literature review

One of the first works in open-ended visual question answering is by (Agrawal, et al., 2015). The team kick-started the research in visual question answering by developing a dataset for the task and coming up with a novel implementation following the above-mentioned pipeline.

The implementation had a vision (image) and language (question) model that culminates with a classifier realized by a *soft-max* over *K (=1000)* possible outputs. The answer for the image-question pair was chosen from top K frequent answers from the training and validation set. The top 1000 most frequent answers made up 82.76% of all the answers in the training and validation set.

The image embedding was realized from l_2 normalized activations from the last hidden layer of VGGNet (Simonyan & Zisserman, 2015) to get a $\mathbb{R}^{4096}$ dimensional image embedding. The image embedding is further reduced to $\mathbb{R}^{1024}$ dimensional embedding by a fully connected layer and a *tanh* non-linearity.

The question embedding consists of a Bag-of-Words Question (BoW Q). The top 1000 words in the questions are used to create a bag-of-words representation. Since there is a strong correlation between the words that start a question and the answer, the top 10 first, second, and third words of the questions are used to create a 30-dimensional bag-of-words representation. These features are concatenated to get a 1,030-dim embedding for the question. This embedding is further enhanced by Long Short-term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) network with two hidden layers is used to obtain 2048-dim embedding for the question. The embedding obtained from the LSTM is a concatenation of last cell state and last hidden state representations. This is

followed by a fully connected layer and a *tanh* non-linearity to transform 2048-dim embedding to 1024-dim.

The transformed image and LSTM embedding are then fused via element-wise multiplication. This combined image + question embedding is then passed to a Multi-Layer Perceptron (MLP) – a fully connected neural network classifier with 2 hidden layers and 1000 hidden units (dropout 0.5) in each layer with *tanh* non-linearity, followed by a *soft-max* layer to obtain a distribution over K answers. The entire model is learned end-to-end with a cross-entropy loss. VGGNet (Simonyan & Zisserman, 2015) parameters are frozen to those learned for ImageNet classification and not fine-tuned in the image channel.

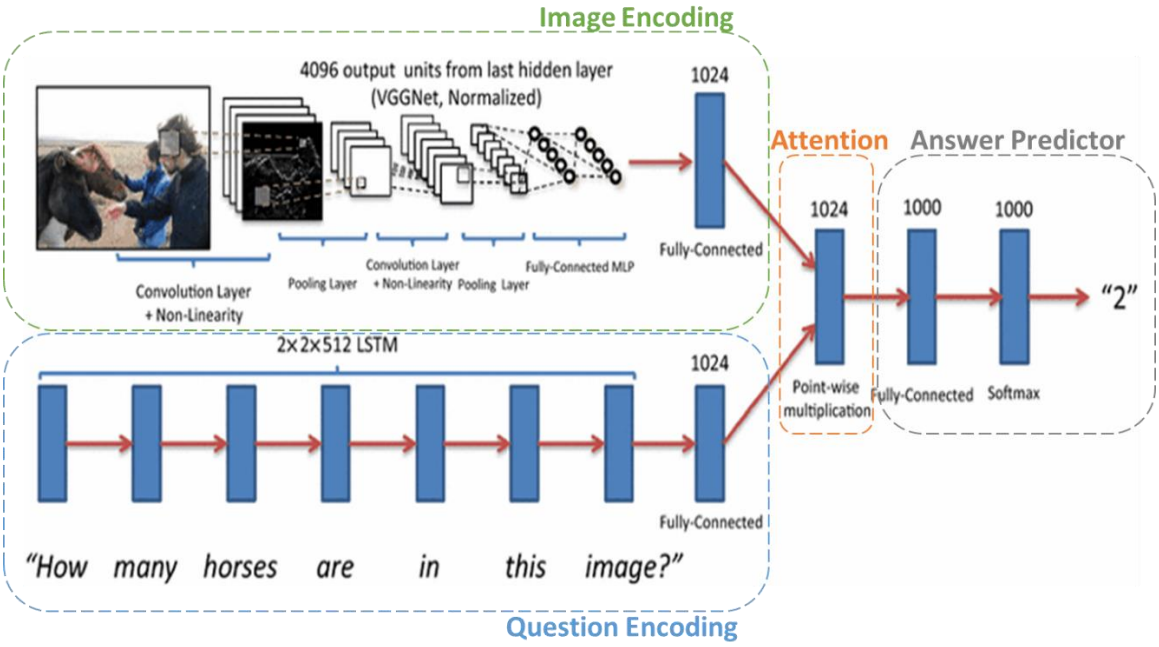

Image Source: (Agrawal, et al., 2015)

Figure 27 Network Architecture of VQA

Another implementation by (Lu, Yang, Batra, & Parikh, 2016), implements a complex and novel approach to the attention mechanism.

The image embedding consists of using the last hidden layer of Resnet (He, Zhang, Ren, & Sun, 2016) architecture.

The question embedding happens in three-fold owing to the hierarchical nature of attention mechanism. With the one-hot encoding of the question words, the words are embed via an LSTM (Hochreiter & Schmidhuber, 1997) network to a vector space. The phrase level features are computed by a 1-D convolution on the word embedding vectors. Concretely, at each word location, the inner product of the word vectors with filters of three window sizes: unigram, bigram, and trigram are computed. The word-level features are appropriately 0-padded before feeding into bigram and trigram convolutions to maintain the length of the sequence after convolution. A max-pooling layer is added to across the n-grams to obtain phrase-level features.

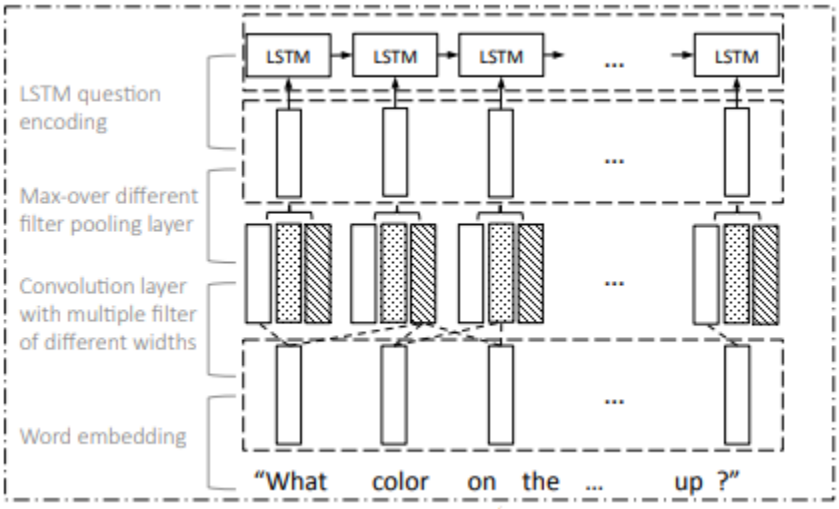

Image Source: (Lu, Yang, Batra, & Parikh, 2016)

Figure 28 Question Encoding in Hierarchical Co-Attention Mechanism

The attention mechanism sequentially alternates between generating image and question attention. It consists of three steps depicted in the figure [29]: 1) summarize the question into a single vector; 2) attend to the image based on the question summary; 3) attend to the question based on the attended image feature.

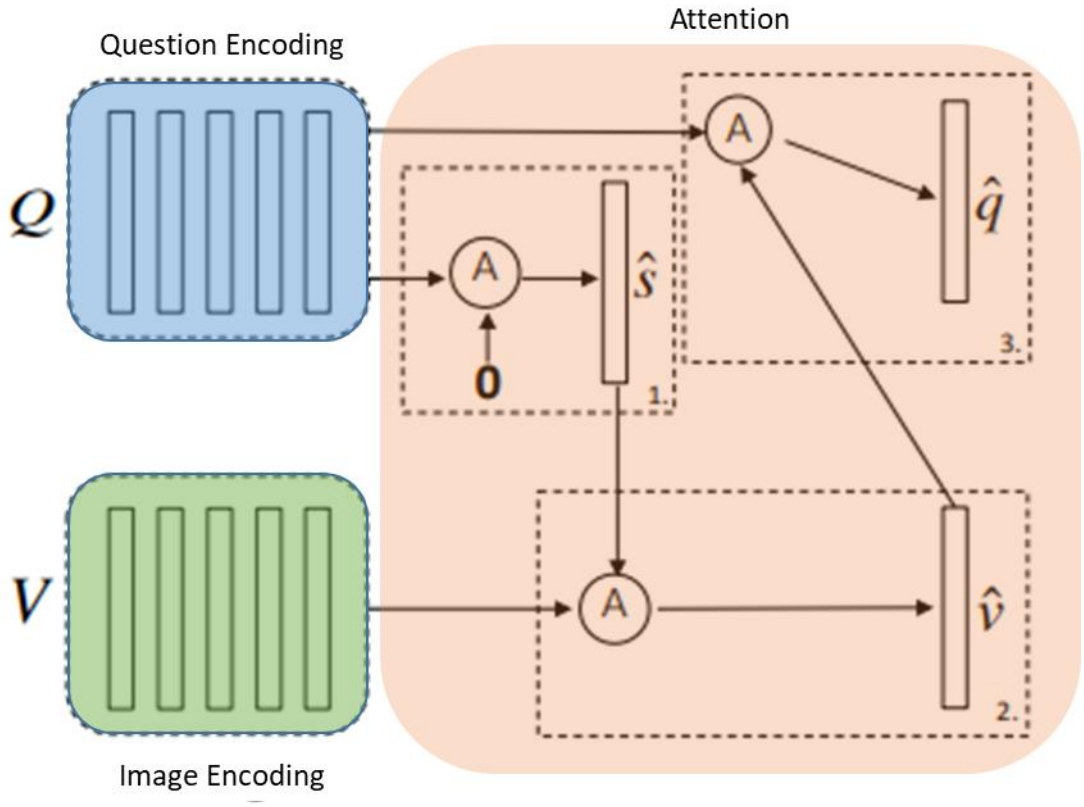

Image Source: (Lu, Yang, Batra, & Parikh, 2016)

Figure 29 Alternating Attention Mechanism in Hierarchical Co-Attention

For the answering routine, the VQA is treated as a classification task. The answer based on the co-attended image and question features from all three levels. A multi-layer perceptron (MLP) is used to encode the attention features recursively and the final answer is predicted using a soft-max layer.

The network is trained using Rmsprop optimizer with a base learning rate of 4e-4, momentum 0.99, and weight-decay 1e-8. The batch size is set to be 300 and trained for up to 256 epochs with early stopping if the validation accuracy has not improved in the last 5 epochs.

An implementation by (Xiong, Merity, & Socher, 2016), uses a dynamic memory network (DMN) module that comprises of (i) input module and (ii) memory module. The input module comprises of modified 2-level sentence encoder with sentence reader and input fusion layer. The memory module consists of an attention module made of dynamic memory Gated Recurrent Units (GRUs) (Cho, et al., 2014). It utilizes episodic memory updates to extract global information with each of the modules through input vectors.

The image encodings are generated, using the VGG19 (Simonyan & Zisserman, 2015) and retrieving the last hidden layer of the network. Then the image encodings are projected from the local, regional vectors to the textual feature space by adding a linear layer with *tanh* activation. To retrieve global information from the local regions generated from the image encodings (also to eliminate scaling or locational variance problems), an input fusion layer is added by snaking through the image vectors and applying bi-directional Gated Recurrent Unit (GRU) (Cho, et al., 2014), to obtain globally aware inputs.

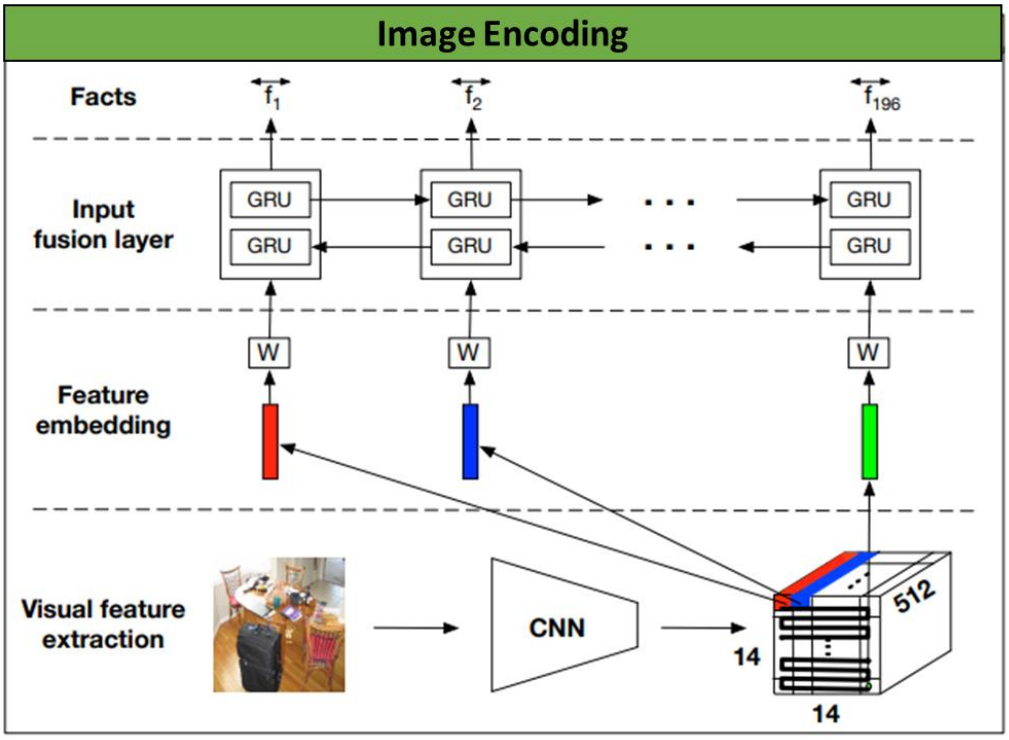

Image source - (Xiong, Merity, & Socher, 2016)

Figure 30 Image Encoding in DMN

The DMN architecture implementation consists of an input fusion layer for question encoding with two components. The first component is a sentence reader, responsible only for encoding the words into a sentence embedding. The second component is the input fusion layer, allowing for interactions between sentences. The sentence reader consists of a positional encoder, as opposed to GRUs (Cho, et al., 2014) and LSTMs (Hochreiter & Schmidhuber, 1997) which are computation-intensive and prone to overfitting. The implementation uses bi-directional GRU (Cho, et al., 2014) for this input fusion layer because it allows information from both past and future sentences.

Image source - (Xiong, Merity, & Socher, 2016)

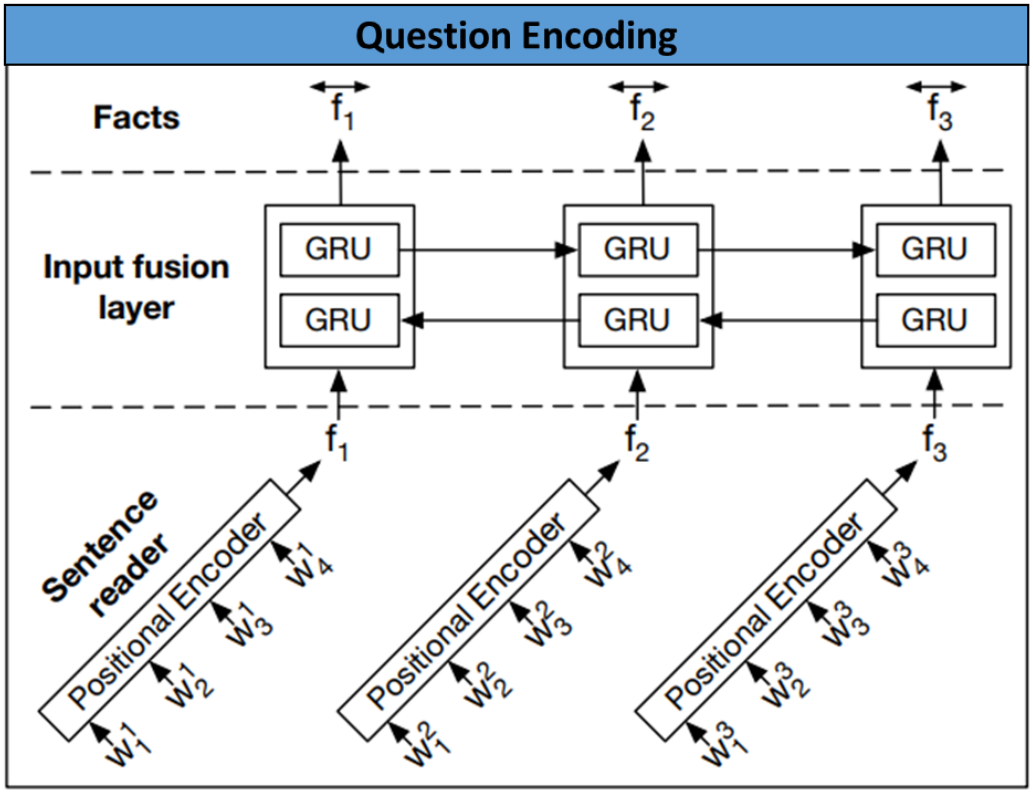

Image source - (Xiong, Merity, & Socher, 2016)

Figure 31 Question Encoding in DMN

The image-question attention mechanism consists of an episodic memory module. The episode memory module may pass over the input multiple times, updating episode memory after each pass. It retrieves information by focusing attention on a subset of the input vectors through an attention gate (derived based on Image vector, question vector, and previous memory vector).The attention mechanism in this method is implemented in two configurations:

Soft attention - Soft attention produces a contextual vector through a weighted summation of the sorted list of vectors and corresponding attention

gates. However, the main disadvantage of soft attention is that the summation process loses both positional and ordering information.

Attention-based GRU – a modified GRU, with the update gate replaced with the attention GRU overcomes the shortcomings of soft attention.

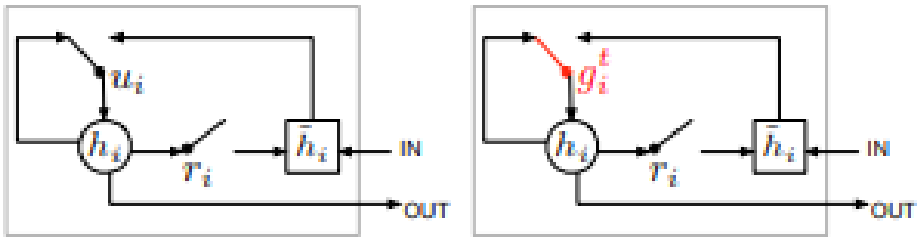

Image source - (Xiong, Merity, & Socher, 2016)

Figure 32 Traditional (Left) vs Attention (right) GRU

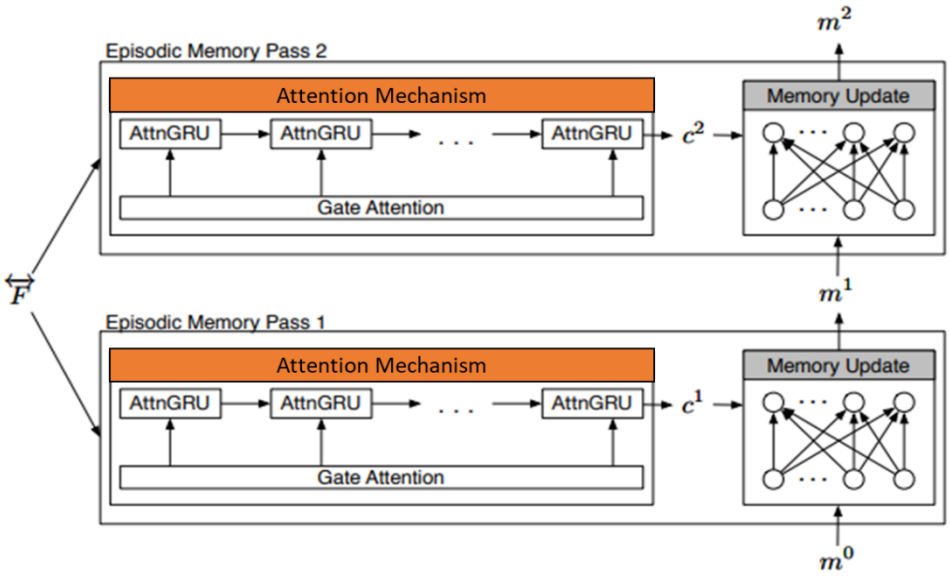

Image source - (Xiong, Merity, & Socher, 2016)

Figure 33 Attention and Answer Prediction Architecture in DMN

Answer prediction is based on the episodic memory after the last pass and the question embedding. The prediction module has two configurations: (i) For simple (one-word) prediction - a linear layer with *soft-max* activation, (ii) For sequence output – the concatenated memory and question vectors are passed through an LSTM (Hochreiter & Schmidhuber, 1997) framework.

The network was trained with Adam (Kingma & Ba, 2015) optimizer with a learning rate of 0.003 and a batch size of 100. The training was run for up to 256 epochs with early stopping if the validation loss has not improved in the last 10 epochs. The weights were initialization from a random uniform distribution with range [−0.08, 0.08] (Glorot & Bengio, 2010). Both the word embedding and hidden layers were vectors of size d = 512. Dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) was applied to the initial image output from the VGG (Simonyan & Zisserman, 2015) as well as the input to the answer module, keeping input with probability p = 0.5.

Appendix III: Future Work (Detailed)

The generation of a scene description graph is inspired by nature. It is based on the fact that humans/animals interpret visual input through the knowledge of activities, events, and objects. Analyzing a visual scene involves continuous interaction with high-level knowledge, some of which are represented in the form of language. In some sense, perception and language interact with each other, and through the exchange, semantics, and understanding are

developed. Thus, the VQA problem can be modeled as a two-module solution: (a) A vision module and (b) a reasoning module that interacts with each other.

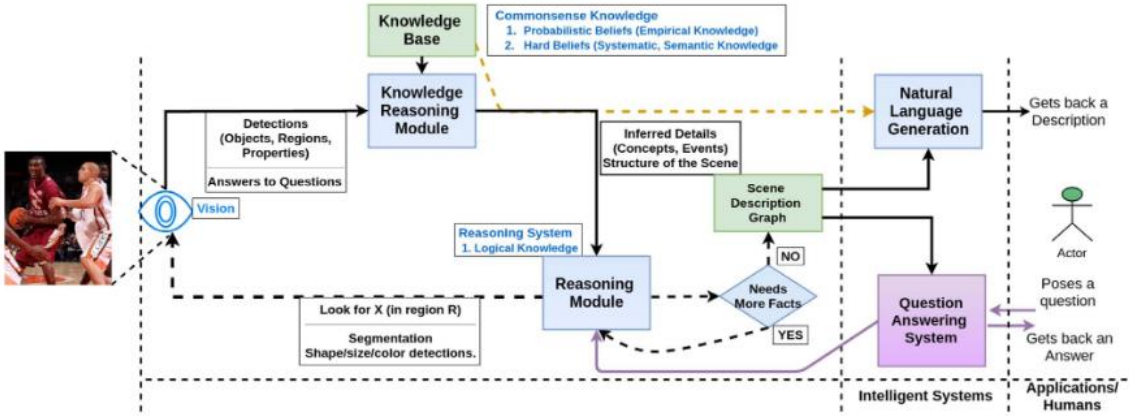

Image source: (Aditya, Yang, Baral, Aloimonos, & Fermüller, Image Understanding using vision and reasoning through Scene Description Graph, 2017)

Figure 34 Network Pipeline for generating Scene Description Graph

The implementation consists of an image understanding architecture, that in turn consists of 1) Visual Detection, 2) Knowledge representation, and 3) Reasoning system.

Visual Detection

Ideally, a visual detection module should consist of a large set of object and scene detection classifiers, relationship detection classifiers, attribute (color, shape, and size) and relative attribute classifiers and detection and Image Segmentation modules. The current approach uses deep object recognition, deep scene (category) recognition, and deep Observed Scene Constituent recognition as the components of the Visual Detection module.

Knowledge representation (Scene Description Graph)

The image is abstracted as a Scene Description Graph to develop an intuitive representation of the image. Primarily, meaningful regions of images that capture relevant semantics and their observable attributes (location, shape, size, color, contour, etc.) are defined. In a scene, these components further are grouped to form observable and inferable components. Observed Scene Constituents (OSC) are descriptions of objects, actions or that can be directly grounded in the image. For example, a person wearing shorts, person skateboarding, tall person, people playing, etc. are all Observed Scene Constituents. Inferred Scene Constituents (ISC) are concepts that cannot be directly grounded in the image but can be inferred. For example, the open space and bright days are ISCs. To develop the knowledge graph, first, the annotations and information from the training images are pre-processed to capture the required commonsense knowledge. Then a rule-based reasoning algorithm is used to infer a knowledge structure. To capture the commonsense and probabilistic knowledge about the domain, a Knowledge Base and a Bayesian Network using the pre-processed data is created. A semantic parser called K-parser is used extensively to extract knowledge from the annotations. K-Parser (kparser.org) is a semantic parser that extracts an Entity–Event-based representation from a sentence, adding additional semantic knowledge. For a sentence such as “A boy wearing swimming trunks jumps over some sprinkler water in a backyard”, the K-parser extracts the Events (actions and linking verbs) wear, jump, and their participant Entities (concrete nouns) boy and trunks, boy and water respectively as a set of Entity and Event-nodes connected

by meaningful relations. It also extracts Traits (attributes) swimming, sprinkler corresponding to the entities. Internally, K-parser uses the Stanford Parser to get the syntactic dependency graph from a sentence. The K-parser then uses a rule-based mapping algorithm to map these dependency relations to the set of KM-Relations.

Leveraging SDGs to answer a question requires the development of a sophisticated probabilistic logical mechanism that can sift through the noise in the generated SDG and understand the natural language question and give an answer. Such mechanisms haven't been researched much. Current approaches fall to a straightforward reasoning module that is incapable of generalizing the question answering task albeit generating a robust understanding of the image.

Another approach involves using the Probabilistic Soft Logic to generate knowledge base from the image (Aditya, Yang, Baral, & Aloimonos, Combining Knowledge and Reasoning through Probabilistic Soft Logic for Image Puzzle Solving, 2018). Probabilistic soft logic (PSL) differs from most other probabilistic formalisms in that its ground atoms have continuous truth values in the interval [0, 1], instead of having binary truth values. The syntactic structure of rules and the characterization of the logical operations have been chosen judiciously so that the space of interpretations with non-zero density forms a convex polytope. This makes inference in PSL a convex optimization problem in continuous space, which in turn allows efficient inference.

A PSL model is defined using a set of weighted if-then rules in first-order logic. PSL relaxes the Boolean truth values of each ground atom a (constant term or predicate with all variables replaced by constants) to the interval [0, 1]. Lukasiewicz's relaxation of conjunctions (∧), disjunctions (∨), and negations (¬) is used to compute soft truth values. In PSL, the ground atoms are considered as random variables and the distribution is modeled using Hinge-Loss Markov Random Field.

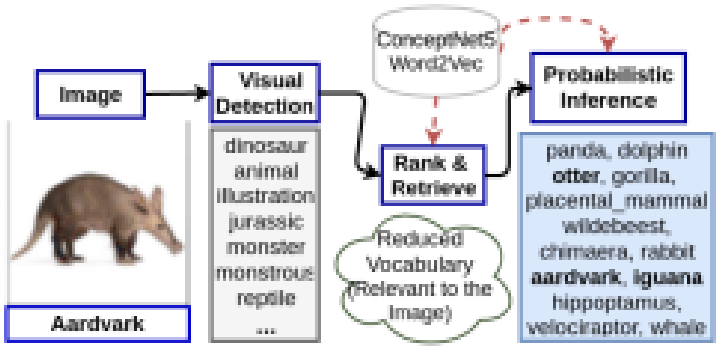

Image source: (Aditya, Yang, Baral, & Aloimonos, Combining Knowledge and Reasoning through Probabilistic Soft Logic for Image Puzzle Solving, 2018)

Figure 35 Network Pipeline for generating Probabilistic Soft Logic (PSL)

Given a set of images, the objective is to determine a set of ranked words based on how well they semantically connect the images. This is achieved by using a Probabilistic Reasoning framework on top of a probabilistic Knowledge Base (ConceptNet) (Liu & Singh, 2004). It also uses additional semantic knowledge from Word2vec (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013). Using these knowledge sources, the answers are predicted. PSL engine is a general tool. It

could be used for further research along with the conjunction of vision, language, and reasoning.

Following the approaches, the Visual Question Answering system will be able to achieve the primary motivation goal of developing a system that is “AI-complete.” These approaches pivot to the direction of developing a reasoning system that entirely and intuitively understands the image, rather than forcing the system to generate/select the most relevant answer based on similarity.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., . . . Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv preprint arXiv:1603.04467*.
2. Aditya, S., Yang, Y., Baral, C., & Aloimonos, Y. (2018). Combining Knowledge and Reasoning through Probabilistic Soft Logic for Image Puzzle Solving. *UAI 2018: The Conference on Uncertainty in Artificial Intelligence (UAI)*, (pp. 238-248).
3. Aditya, S., Yang, Y., Baral, C., Aloimonos, Y., & Fermüller, C. (2017). Image Understanding using vision and reasoning through Scene Description Graph. *Computer Vision and Image Understanding, 173*, 33-45.
4. Agrawal, A., Lu, J., Antol, S., Mitchell, M., Zitnick, C., Batra, D., & Parikh, D. (2015). VQA: Visual Question Answering. *arXiv preprint arXiv:1505.00468*.
5. Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., & Zhang, L. (2017). Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. *arXiv preprint arXiv:1707.07998*.
6. Cao, G., Yang, Y., Lei, J., Jin, C., Liu, Y., & Song, M. (2017). TripletGAN: Training Generative Model with Triplet Loss. *arXiv preprint arXiv:1711.05084*.
7. Che, T., Li, Y., Jacob, A., Bengio, Y., & Li, W. (2017). Mode Regularized Generative Adversarial Networks. *ICLR 2017 : International Conference on Learning Representations 2017*.
8. Cho, K., Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder--Decoder for Statistical Machine Translation. *Proceedings of the 2014 Conference*

on Empirical Methods in Natural Language Processing (EMNLP), (pp. 1724-1734).

9. Feizi, S., Suh, C., Xia, F., & Tse, D. (2018). Understanding GANs: the LQG Setting. *arXiv preprint arXiv:1710.10793*.
10. Fukui, A., Park, D., Yang, D., Rohrbach, A., Darrell, T., & Rohrbach, M. (2016). Multimodal Compact Bilinear Pooling for Visual Question Answering and Visual Grounding. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* , (pp. 457-468).
11. Girshick, R. (2015). Fast R-CNN. *2015 IEEE International Conference on Computer Vision (ICCV)*, (pp. 1440-1448).
12. Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *CVPR '14 Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 580-587).
13. Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, (pp. 249-256).
14. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., . . . Bengio, Y. (2014). Generative Adversarial Nets. *Advances in Neural Information Processing Systems 27*, (pp. 2672-2680).
15. Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., & Parikh, D. (2017). Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question

Answering. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 127*, pp. 398-414.

16. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. (2017). Improved Training of Wasserstein GANs. *Advances in Neural Information Processing Systems*, (pp. 5767-5777).
17. He, K., Gkioxari, G., Dollar, P., & Girshick, R. (2018). Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1-1.
18. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 770-778).
19. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735-1780.
20. Kingma, D., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. *ICLR 2015 : International Conference on Learning Representations 2015.*
21. Kingma, D., & Welling, M. (2014). Auto-Encoding Variational Bayes. *ICLR 2014 : International Conference on Learning Representations (ICLR) 2014.*
22. Kullback, S., & Leibler, R. (1951). ON INFORMATION AND SUFFICIENCY. *Annals of Mathematical Statistics, 22*(1), 79-86.
23. Kurach, K., Lucic, M., Zhai, X., Michalski, M., & Gelly, S. (2019). A Large-Scale Study on Regularization and Normalization in GANs. *ICML 2019 : Thirty-sixth International Conference on Machine Learning*, (pp. 3581-3590).

24. Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., . . . Zitnick, C. (2014). Microsoft COCO: Common Objects in Context. *European Conference on Computer Vision*, (pp. 740-755).

25. Liu, H., & Singh, P. (2004). ConceptNet — A Practical Commonsense Reasoning Tool-Kit. *Bt Technology Journal, 22*(4), 211-226.

26. Lu, J., Yang, J., Batra, D., & Parikh, D. (2016). Hierarchical Question-Image Co-` for Visual Question Answering. *Advances in Neural Information Processing Systems*, (pp. 289-297).

27. Mao, X., Li, Q., Xie, H., Lau, R., Wang, Z., & Smolley, S. (2016). Least Squares Generative Adversarial Networks. *arXiv preprint arXiv:1611.04076*.

28. Mescheder, L., Geiger, A., & Nowozin, S. (2018). Which Training Methods for GANs do actually Converge. *arXiv: Learning*.

29. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems 26*, (pp. 3111-3119).

30. Mroueh, Y., Sercu, T., & Goel, V. (2017). McGan: Mean and Covariance Feature Matching GAN. *International Conference on Machine Learning*, (pp. 2527-2535).

31. Nash, J. (1950). Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America, 36*(1), 48-49.

32. Novoselov, S., Shchemelinin, V., Shulipa, A., Kozlov, A., & Kremnev, I. (2018). Triplet Loss Based Cosine Similarity Metric Learning for Text-independent Speaker Recognition. *Interspeech 2018*, (pp. 2242-2246).

33. Nowozin, S., Cseke, B., & Tomioka, R. (2016). f -GAN: training generative neural samplers using variational divergence minimization. *NIPS'16 Proceedings of the 30th International Conference on Neural Information Processing Systems*, (pp. 271-279).

34. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., . . . Lerer, A. (2017). Automatic differentiation in PyTorch.

35. Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (pp. 1532-1543).

36. Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *ICLR 2016 : International Conference on Learning Representations 2016* .

37. Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. (2016). Generative adversarial text to image synthesis. *ICML'16 Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, (pp. 1060-1069).

38. Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. *NIPS'15 Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, 2015*, pp. 91-99.

39. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training GANs. *NIPS'16 Proceedings of the 30th*

International Conference on Neural Information Processing Systems, (pp. 2234-2242).

40. Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *ICLR 2015 : International Conference on Learning Representations 2015*.
41. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research, 15*(1), 1929-1958.
42. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., . . . Rabinovich, A. (2015). Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 1-9).
43. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 2818-2826).
44. Teney, D., Anderson, P., He, X., & Hengel, A. (2018). Tips and Tricks for Visual Question Answering: Learnings from the 2017 Challenge. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (pp. 4223-4232).
45. Xiong, C., Merity, S., & Socher, R. (2016). Dynamic memory networks for visual and textual question answering. *ICML'16 Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, (pp. 2397-2406).
46. Xu, T., Zhang, P., Huang, Q., Zhang, H., Gan, Z., Huang, X., & He, X. (2018). AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative

Adversarial Networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (pp. 1316-1324).

47. Yang, Z., He, X., Gao, J., Deng, L., & Smola, A. (2016). Stacked Attention Networks for Image Question Answering. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 21-29).
48. Zhao, J., Mathieu, M., & LeCun, Y. (2016). Energy-based Generative Adversarial Network. *arXiv preprint arXiv:1609.03126*.