university of groningen / faculty of science and engineering

Artificial Intelligence, Faculty of Science and Engineering, University of Groningen

Master's Thesis

Exploring the robustness and generalizability of visual question answering and image-generating models using a cyclic architecture

Author: Camilo Nathan Jansen

First supervisor: Prof. dr. L.R.B. Schomaker
Second supervisor: MA, Z. Zhang

August 09, 2021

Abstract

In recent artificial intelligence research, a shift has started to occur from studying single domain problems to multi-domain ones. These multi-domain problems require more general intelligence from the models making them an interesting area of research. Visual question answering (VQA), the task of answering open-ended questions about images, is such a problem. Current VQA models are not showing a deep fundamental understanding of the visual contents of the images. Instead, one of their problems is that they abuse statistical aspects of the data set to achieve their performance, ultimately lacking generalizability and robustness for real-world applications. Another multi-domain problem is the task of image generation (IG). In this task, a semantically matching image is synthesized from a text description. Current IG models can generate good images for narrow data sets but struggle when moving to more broad ones, often lacking semantic consistency and compositionality. This thesis aimed to explore the generalizability and robustness of these tasks to better understand their abilities. To relate the tasks more the IG task was modified to synthesize the image from a question-answer pair so it uses the same data points as the VQA task. By making this modification, the models can be employed in cyclic architectures. The applications of these cyclic architectures towards improving the generalizability and robustness were also explored. As an additional evaluation metric for the IG task, a VQA consistency metric was explored, which aims to measure the semantic consistency through a VQA model serving as a critic of the IG model. The study was performed on a data set with images of abstract stylized shapes instead of real-world photos to reduce the problem space. A set of attributes for each shape defines the question-answer pairs for each image and determines question type and specificity information, enabling more analysis opportunities. Also, the attribute combinations are split between testing and training to ensure the model sees unseen combinations of attributes in the test set, requiring more generalizability. The quantitative side of the results shows that a VQA architecture using a joint-embedding architecture with visual top attention was the most effective at generalizing to the test set. While performing the best, this model struggled with spatial understanding like the other explored architectures. Similarly, the IG models struggled with producing clear spatial understanding in their synthesized images. The quantitative metrics for IG failed to follow the results of the qualitative review due to them seeing the background features as high-quality. The VQA consistency metric proved to be the only effective metric at matching the quantitative results with the qualitative ones proving its effectiveness. However, the IG models generally failed to capture the full essence of the question-answer pairs they generated from. Generally, they were only able to semantically matching parts of them. This most likely was caused by lookup-table-like behavior, matching textual features with visual features instead of showing a deeper understanding of what it needs to generate. Lastly, the VQA model can learn more variation through a cyclic architecture with the IG model but fails to improve generalizability due to the lack of semantic consistency of the IG model. Some future work ideas are presented to improve some of the problems found with the models.

Preface

Throughout the making of this thesis, we have been going in and out of lockdowns due to the novel coronavirus. It made the process of the thesis much different than what I expected when starting this master's degree. A large part of the work on this thesis had to be done from home instead of being able to work at the university. This made it hard to stay motivated at points but through the support of my friends and family, it was manageable. In addition to them, I would like to thank my supervisor Lambert Schomaker for the insights and guidance he provided throughout the making of this thesis. I was completely new to this topic, so the guidance proved invaluable. Also, I would like to thank my second supervisor Zhenxing Zhang for helping me with writing the proposal and understanding the tasks of visual question answering and image generation at the start of the thesis. Last, I would also like to thank the CIT for providing and maintaining the peregrine HPC cluster, enabling the models of this thesis to be trained. I learned a lot throughout making this thesis and hope it can provide the reader with some interesting insights.

Contents

Chapter 1

Introduction

The areas of artificial intelligence (AI) that have received most of the attention and yielded successful real-world applications are traditionally limited to single-domains such as computer vision (CV) and natural language processing (NLP). The research in these areas has become considerably advanced through deep-learning neural networks, replacing classical methods of machine learning as default state-of-the-art. However, the advances in single-domain research do not necessarily indicate AI has started to achieve general intelligence, which ultimately is the goal of AI. Problems in the real world almost always require an intelligent agent to engage with multiple domains. A domain like natural language can describe the world but requires the imagination of a different modality such as vision or speech to do so. For example, when describing an object in natural language, you transform the visual knowledge in your brain of this object into a language description. Therefore, only utilizing the language domain is not enough. To get closer to intelligence, AI models need to be able to handle multiple domains. To this end, research has started to shift towards multimodality and multi-domain problems. In a multi-domain problem, an AI model combines multiple domains to transform one to the other or use their combined information to perform a task such as classification. A common trend in the research has been to combine CV and NLP to perform a myriad of novel tasks. The visual and textual worlds translate naturally into each other, as language can describe the contents of the visual world. Conversely, the visual world can be represented in a compact manner by means of a textual description. Therefore, choosing AI challenges that incorporate these two domains at the same time presents a good starting point for multi-domain AI research.

In this thesis, two multi-domain problems that lie at the intersection of CV and NLP are explored. The first explored multi-domain problem is visual question answering (VQA). VQA is the task of answering free-form open-ended questions about the visual contents of an image [5]. The task was proposed as an advancement of the related multi-domain task of image captioning [3] to require more advanced information processing and reasoning [5]. As the questions are open-ended, the possibilities for them are endless, in principle. Answering image-content-related questions requires a vast amount of different visual capabilities in combination with language understanding and reasoning. These required capabilities are best shown through a couple of examples. For example, a relatively simple question for a human: "Will the dog catch the frisbee?" not only requires object recognition of the dog and frisbee but also requires spatial understanding between them and reasoning about the future state of the scene. Even a less complex question like: "How many chairs are around the table?" requires an understanding and counting ability concerning the chairs that surround the table. Some more VQA examples can be seen in Figure 1.1. These are just a couple of possibilities that exemplify how complicated the task can be, even for relatively simple questions. Therefore, the diverse nature of the questions and the required skills to determine an accurate answer make this task a more suitable test of intelligence than previous research.

Visual question answering (VQA) has advanced massively from the earliest approaches towards the current state-of-the-art but is still far from seeing real-world applications. However, it is important to consider the possibilities for this in the research. For example, VQA could be used as a visual aid for the visually impaired, assisting them in understanding the visual world that surrounds them. It can also be used as a tool for AI-based medical understanding, supporting doctors in a second opinion by allowing them to ask the model questions about a medical image. These are just some of the possible applications for VQA [8]. For VQA to receive real-world applications, the models must be robust and generalize well to the real world from their training. Currently, the robustness of VQA models is far from desired. They are vulnerable to adversarial

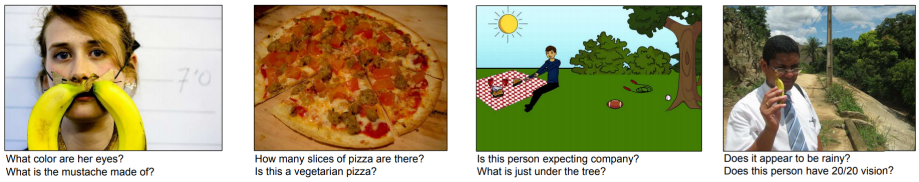


Figure 1.1: Examples of the visual question answering task from the VQA 1.0 data set [5]. Figure modified from [5].

attacks, which fool the model by making minor changes to the image without affecting the actual visual contents [55]. The models are also likely to abuse language priors in the training data. In [23, 1] question-only and visual-only VQA models were trained in addition to a regular VQA model. They found that the question-only model performed drastically better than the visual-only one. This phenomenon can be attributed to two factors: The type of question constrains the possible answers. For example, the question "What color is the table?" constrains the possible answers to the set of colors. The other more problematic factor is the bias that exists in the data set. In one of the most important VQA data sets, VQA 1.0 [5] questions that start with the n-gram "Do you see a ..." can be blindly answered "yes" correctly 87% of the time [15]. These kinds of language priors are used by the models to answer questions correctly without considering visual information. The abuse of language priors was further researched in [15], where the same authors of VQA 1.0 rebalanced it to avoid these priors. Rebalancing the data set caused a significant drop in the performance of the VQA model, indicating the use of language priors by the model. VQA models are also not robust against rephrasings of the same question as shown in [43]. They introduced a VQA-Rephrasing data set which contains rephrasings of each question in the data set. Performing an analysis on state-of-the-art VQA models on this data set showed that the models produced different answers for the same question phrased in different ways without changing the meaning. These problems with VQA models show that there is still a long way to go to achieve real-world applications. Therefore, understanding and exploring the generalizability and robustness of these models can help us learn to understand the failure points and abilities of the models.

A different multiple-domain problem that involves the visual and textual domains is the task of image generation (IG). In IG, a semantically matching image is synthesized based on a text description, transforming the textual domain into the visual. In the research towards IG, the text description has traditionally been a descriptive caption describing the contents of the image. To generate the image, a conditional extension of the generative adversarial network (GAN) has been the most common approach throughout IG research [57, 58, 54, 48]. In a GAN, an image generator is trained simultaneously with a discriminator through a zero-sum game where the generator tries to synthesize images that fool the discriminator while the discriminator tries to discriminate between real and fake images [14]. By having both networks optimize their objectives, the generator eventually learns to generate real-looking images. The generator synthesizes images from a latent space based on the learned real-data distribution. For the IG task, conditioning is applied to the generation based on a text embedding of the description. This ensures the generated image is related to the text description, while the latent space ensures variety in the generated images for this description [32]. The state-of-the-art GAN approaches can create high-quality images on specific data sets such as the CUB birds [50] data set, which contains images of birds with long descriptive captions, but struggle when images require multiple objects like in the MSCOCO data set [29]. They can match important visual features with textual features in a lookup-table-like manner, but fail to understand the underlying relations between them, which explains why these approaches struggle on the more advanced generation tasks such as MSCOCO [21]. To achieve real-world applications, the IG models should be able to understand these relations such that they can generalize them to unseen data. For example, in the real-world application of data augmentation, the IG model should generate a variety of novel images, while consistently being semantically correct towards the text description. If an IG model can do this, it would be a

very effective tool for data augmentation. However, the current state of IG models is not at this level yet. Therefore, it is important to understand their current ability to generalize and also their general robustness with regards to semantic consistency.

While not necessarily being directly related, the visual question answering (VQA) and image generation (IG) tasks share some characteristics. They both involve the transformation of one or more domains into a single domain. In VQA, a combination of the visual and textual domain is transformed to the textual domain. Reversely, the textual domain is transformed to the visual one in IG. To relate the tasks more, the IG task can be adopted to synthesize an image from a question-answer (QA) pair instead of a caption. Approaching IG this way can have multiple benefits towards exploring its robustness. A QA pair is less descriptive than a caption, opening up more possible variations of matching images. For example, a generated image for the QA pair "Q: Is there a dog in the image? A: Yes" could have a dog everywhere in the image, with a background such as a grass-field, a room in a house, etc. Also, the dog can be of any species as this is not specified in the question. Therefore, there are many possible semantically correct interpretations the IG model could make as the QA pair is less restrictive than a caption. The other benefit of this approach is that the IG model must show generalizability to achieve good semantic consistency. Simply matching the visual with textual features in a lookup-table-like manner is not enough to achieve this. To be semantically consistent, it has to understand the relation between the question and answer, which requires the model to generalize from training. For example, if the model was shown the QA pair "Q: How many kittens are there? A: Four" during training and has seen the answer "A: Three" in a different training example, it should be able to generalize that a testing example "Q: How many kittens are there? A: Three" should be an image of three kittens to be semantically consistent. Therefore making this modification to this task forces the IG model to show generalizability. However, a problem with measuring this is that the current IG evaluation metrics fail to measure semantic consistency [41, 19]. In this thesis, we propose a new method that tries to measure if the image content of the synthesized image is semantically consistent with the intended interpretation of the text input used to generate it. To do this we utilize the modification made to the IG task to generate from question-answer pairs. This allows us to measure the consistency of the VQA model's answer with the intended answer from the question-answer pair used to synthesize the image. The accuracy of the VQA answers with the intended answer is the metric, which should measure the semantic consistency of the models. The effectiveness of this VQA consistency metric is explored in this thesis.

The newly proposed semantic-consistency metric can be seen as a single step in a cyclic architecture of the image generation (IG) and visual question answering (VQA) models. In a cyclic architecture, two models pass input and output to each other in a cyclical manner. For example, consider a cyclic architecture consisting of a main model that maps a data point from a data set to an output and a remapping model that maps a new data point from this output, the original data point, or a combination of both. Then, the output of the extra model can be reused as input for the main model. Performing this process once is a single cycle pass and can be done multiple times. An example of such a structure can be seen in Figure 1.2. In this study, such architectures are made possible by the modification of the IG task to generate from a QA pair, making both tasks utilize the same data points. There are many different applications for these cyclic architectures. The cycle concept has been used to force consistency (also called cycle consistency) between the original output and further cycle outputs [59, 43, 35]. When the remapping model ensures variation, the main model becomes more robust to variance through this approach. A cyclic architecture has been applied to VQA in [43]. They made their VQA model more robust against rephrasings by utilizing the VQA model in a cyclic architecture with a module that rephrases the question. The question and image are passed through the VQA model to generate the first answer. Then, the question is rephrased and passed through the VQA model again with the image to generate a second answer. By forcing consistency between the first and second answers, the VQA model became more robust against rephrasings and improved its general performance. A different kind of cyclic architecture has also been applied to IG called MirrorGAN to ensure semantic consistency of the IG model [35]. In their model, an image is synthesized based on a caption. Then, a module generates a new caption based on the synthesized image. A consistency

is forced between the original caption and the new caption to ensure the image contents are semantically consistent. Forced cycle consistency is just one of the possible applications of a cyclic architecture. It can also function as a data augmentation method by storing the newly generated data points in each cycle step. The more cycle steps, the more the data varies from the original data, as each model in the cyclic architecture introduces more variance. By training models end-to-end in a cyclic architecture it could also function as a way to make them more explainable. This would be achieved by looking at and analyzing the intermediate outputs during the cycles. By training the models end-to-end, the IG model could be trained to synthesize images that represent the thinking of the VQA model. The synthesized images would contain what the VQA model thinks are the required visual elements for answering the question correctly. However, as training these models end-to-end is quite difficult and resource-intensive this is not explored. Instead, the applications of cyclic architectures towards improving the generalizability and robustness of the VQA and IG tasks are explored by using pre-trained VQA and IG models and cyclically training one of them while keeping the other model frozen.

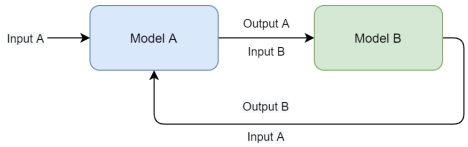


Figure 1.2: A visual example of the basic cyclical architecture described in the text. In the cycle, model A produces an output for the initial input. This output is used by model B to produce a new input for model A. Model A can now use this input to produce a new output for model B and continue the cycle. This encompasses a single cycle step and can be continued for more cycles.

To perform this study, a new data set containing images of abstract, stylized shapes is used instead of real-world photos (see Figure 1.3 for examples). The current VQA models do not seem to be showing a deep fundamental understanding of the visual contents of the images. Instead, they abuse statistical aspects of the data set they train on like language priors to achieve their current performance. It is important that the VQA models can show a deeper understanding of the task, as it is impossible to cover the whole real world in a comprehensive data set. To better understand what the current models are doing we want to simplify the multi-domain problem so that we can look at them at a lower level. Therefore this data set that reduces the problem space to an abstract world of stylized shapes was chosen. It allows us to look at the abilities of the models at a low level. The abstract data set is called ShapeVQA and was purpose-built to explore generalizability and robustness. Instead of being based on manually labeled real-world images, this data-set is automatically generated by utilizing sets of attributes that define shapes in the images. Each image contains one or two shapes, where each shape is defined by the combination of size, color, location, and shape attributes. This vastly constrains the world of the data set. The question-answer pairs for each image are generated by considering the attributes of the shapes present in them. This enables two important things: we can keep track of the question types as we know the attributes it relates to the most and we can determine the level of specificity of the questions based on the number of attributes present in them. These two aspects make studying the generalizability and robustness much more straightforward than the common VQA data sets, which lack such clear information on the questions. The specificity level functions as a kind of difficulty measure of the question, while the type functions as an analysis tool on the performance of different types of questions. The design of this data set also has another important aspect for studying generalizability. The permutations of attribute combinations are split into train, validation, and test set and used to sample images for the corresponding sets. This ensures that the shapes in the sets use unique combinations of attributes not present in the other sets. To handle this aspect, the VQA model should show compositionality so it can generalize from the train set to the test set. For example, if it was shown a small green circle in the center during training and a small yellow circle in testing, it should generalize the small, circle, and center

aspects of the circle. For these reasons, the ShapeVQA data set should be an effective data set to study the models.

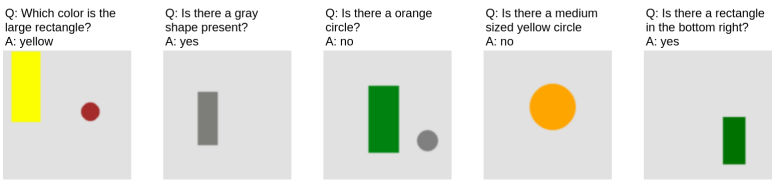


Figure 1.3: Example of the images present in the ShapeVQA data set. On top is a single corresponding question-answer pair. In total ten were defined for each image. See Figure 3.1 for more examples.

Instead of following the trend in the research of modifying existing architectures on a real-world data set to improve performance, the goal of this study is to explore the generalizability and robustness of the two tasks on an abstract simplified data set. This should give us more of an understanding of the strengths and weaknesses of the models for the two tasks. The conclusions drawn from this can be used to direct future research towards more general improvements. To do this multiple setups of VQA and IG models are trained. Both tasks require text embeddings of the textual inputs. To compare the abilities of different embedding methods for the tasks multiple methods are explored. These are used for each setup of VQA and IG. The VQA setups are analyzed by utilizing the question information present in the ShapeVQA data set such as the type and specificity. The trained VQA model is also used as a performance measure on the semantic consistency of the IG synthesized images by calculating its accuracy on data points replacing the original image with the synthesized image. In addition, three qualitative reviews are done on a small selection of question-answer pairs for the IG model. The applications of cyclic architectures towards improving the generalizability and robustness are also explored for both tasks. Assuming the IG model produces semantically correct images, it can provide the VQA model with robustness to visual variation through further cyclical training. Additionally, the VQA model could force the IG model to become more semantically consistent through a cyclic architecture. The VQA model can perform a redescription of the question-answer pair based on the generated image and corresponding question to determine a new answer. Forcing a consistency between this answer and the answer used to synthesize the image could improve the semantic consistency of the IG model similar to the IG model in the MirrorGAN [35]. Both these applications of the cyclic architecture are explored to study the generalizability of the tasks. All these aspects of the thesis lead to the following four research questions:

- Can we show degrees of generalizability and robustness of visual question answering models by measuring performance on different question types and specificities?
- Can we show degrees of generalizability and robustness of an image generation model by generating images from question-answer pairs?
- What are the applications of combining image generation and visual question answering in a cyclic architecture towards improving their generalizability and robustness?
- Can a visual question answering model trained on the same data set as an image generation model be used as a performance measure on the semantic consistency of image generation?

The rest of the thesis is concerned with answering these research questions. First, the theoretical background is presented which outlines the relevant theory to the methods used in the experiments. The next chapter builds upon this by presenting the methodology of the study. Following this, the experimental procedure and corresponding results of the experiments are presented. In the final chapter, these results are discussed, answering the research questions and drawing conclusions from this. The study is evaluated and some ideas are presented for future work.

Chapter 2

Theoretical Background

In this chapter, the theoretical background to this thesis is presented. To train the models to learn the tasks of visual question answering (VQA) and image generation (IG) machine learning was used. The types of machine learning utilized are briefly described in the first section 2.1. All of the models utilize the recent developments in deep learning to perform the learning. Deep learning builds upon the theory of artificial neural networks (ANN), so the background to ANN is first presented in section 2.2. The background to ANN provides the building blocks for the relevant deep-learning architectures presented in section 2.3. The last important background before the theory towards the tasks can be presented is the theory behind the natural language processing of the text in section 2.4. The preceding sections encompass the background towards the methods utilized for the visual question answering and image generation tasks. The theory for these are presented in the final two sections 2.5 and 2.6.

2.1 Machine Learning

Machine learning is a research area of AI, which involves the study of algorithms that automatically improve by learning from data. Machine learning is a broad area of research that spans multiple domains and many tasks. A machine learning model learns patterns from training data, which it utilizes to generalize to unseen data. How these patterns are learned from the data depends on the learning method used and the goal. The three main learning methods are supervised learning, unsupervised learning and reinforcement learning. For this thesis supervised and unsupervised learning are utilized. In addition to these, transfer learning is also used. This involves the transferring of learned knowledge from a trained model to a new one. Supervised and unsupervised learning are further described in the following sections.

2.1.1 Supervised learning

In supervised learning, the goal is to train a model that uses the patterns in the training data to learn a function $f : x \rightarrow y$ that maps an input x to a target output y. The model is presented with example (x, y) pairs during the training process. As the target is shown to the model, the learning process is supervised. The types of inputs and outputs depend on the task and do not necessarily have to be of the same domain. For example, a model could have as input a feature vector of an image and as output a sequence of words. Common supervised learning tasks are classification and regression. In this thesis supervised learning is used for the visual question answering task.

2.1.2 Unsupervised learning

In unsupervised learning, the target y is not provided to the model in the training process, which means it can only use the input data x. The goal of unsupervised learning is to learn the patterns that exist in the input data x. An example of an unsupervised learning task is dimensionality reduction. For this task, the model is trained to learn a lower-dimensional representation of the data by utilizing only the most relevant information in the data. Another use case is clustering, where the model is trained to group similar data points into separate clusters. Unsupervised learning is also used in generative adversarial networks and variational autoencoders, which are both utilized for image generation. These networks are described later in this chapter.

2.2 Artificial Neural Networks

Artificial Neural Networks (ANN) are networks with interconnected artificial neurons forming a directed weighted graph. The architecture is loosely inspired by the way neurons operate in the human brain. The set of weights are learnable giving the network the ability to learn patterns. The first ANNs were shallow only utilizing a couple of relatively simple layers. Later, more complex layers in combination with deeper networks were introduced. These kinds of networks are called Deep Neural Networks (DNN). ANNs and DNNs have been some of the most prevalent machine learning methods and also form the basis of the methods used in this thesis. In this section, their theoretical background is presented.

2.2.1 McCulloch-Pitts Neuron & Perceptron

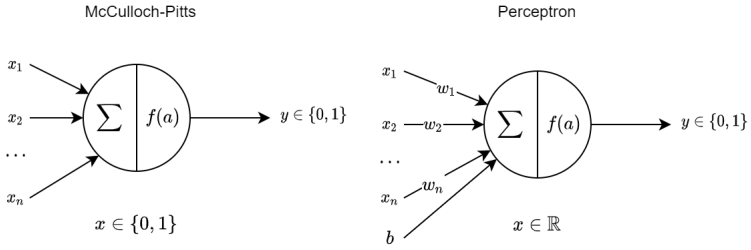


Figure 2.1: **Left**: A McCulloch-Pitts Neuron with 1 to N inputs and a single output. **Right**: A perceptron with weighted inputs, a bias term and a single output.

The first abstraction of a neuron was made by McCulloch and Pitts in 1943 [30]. The neuron was expressed in a mathematical model. It consists of a single neuron with multiple binary-valued inputs. In the neuron the input values are summed (see equation 2.1) and passed through an activation function to determine the binary output (see equation 2.2). The activation function is a step function based on a threshold that determines the binary value. See Figure 2.1 for a graphical view of this model.

$$\sum_{i=1}^{n} x_i = a \tag{2.1}$$

$$f(x) = \begin{cases} 0 & x < threshold \\ 1 & x \geq threshold \end{cases} \tag{2.2}$$

The network can model linearly separable logical functions such as AND and OR by setting the correct threshold. Consider a model with two inputs attempting to model the AND function. It should output true only when both inputs are true. A threshold value of 2 achieves this. Similarly, for OR the output should be true when at least one of the inputs is true. This is modeled by a threshold value of 1. The threshold can be seen as a linear decision boundary, which separates the truth values. The McCulloch-Pitts neuron model is quite simple and limited but introduced concepts that are still commonly used in ANNs

To introduce learning to the McCulloch-Pitts model Rosenblatt proposed the perceptron in 1957 [38]. The perceptron is a binary classifier that accepts a vector of real-valued inputs. These inputs can be seen as features collectively representing a data point. Each input is weighted and added to a bias term (see equation 2.3). The bias term b is a learnable constant added to move the linear decision boundary from the origin. The activation function is changed to the unit step function (see equation 2.4). See Figure 2.1 for a graphical view of this.

$$\sum_{i=1}^{n} x_i w_i + b = a \tag{2.3}$$

$$f(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases} \tag{2.4}$$

By utilizing supervised learning, the perceptron learns the values for the weights and bias iteratively. Intuitively, the learning works by initializing a random decision boundary and moving the boundary towards a location where it can best separate both classes. The decision boundary is determined by the weights and bias parameters, so the process is started by randomly initializing these. To move the boundary towards the best location, training example pairs (x, y) are presented, where x is the input and y is the target output. For each training pair, a forward pass through the network is performed to calculate the target estimate $\hat{y}$ for the input x. If the target estimate is correct, the weight vector is moved towards the input vector, else it is moved away. How much the vector is moved depends on the learning rate η, which is a small constant. This process is done until convergence or a certain error threshold is achieved. The weight update rule for a single example pair follows equation 2.5.

$$w \leftarrow w + \eta(y - \hat{y})x \tag{2.5}$$

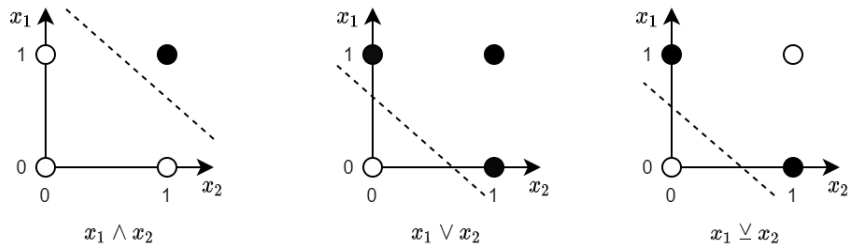


Figure 2.2: Linear separability for the different logical functions mentioned. The black dots are true outputs, the white false. The AND and OR functions are linearly separable through a single linear decision boundary, unlike XOR.

With the introduction of learning, the perceptron was a major step forward, but it still has a major limitation; It only works for linearly separable problems. This problem can best be exemplified through a visual example for the different logical functions AND, OR and XOR as can be seen in Figure 2.2. Each graph contains four data points for each true/false combination, colored by their respective class for the corresponding logical function. The dotted line represents the linear decision boundary. For the AND and OR logical functions there exists a single linear decision boundary that can separate both classes. For XOR this is not the case as it requires a non-linear boundary to properly separate the data points, which is not achievable with a perceptron model. To solve this some innovations were required which are presented in the next section.

2.2.2 Multi-layer perceptron

The original perceptron has a linear activation function and only the two layers of input and output. These aspects of the perceptron limit it to linearly separable problems. To extend the applications of the perceptron to non-linearly separable problems a couple of additions had to be made. The two main additions were the introduction of a hidden layer between the input and output and the utilization of non-linear activation functions. This kind of model is either named a multilayer perceptron (MLP) or a feedforward neural network as the network represents a directed acyclic graph. The general structure of an MLP is outlined in Figure 2.3. It is fully connected, which means that each node has a connection to every node from the previous layer. This allows the input to flow throughout the whole network. By combining this with non-linear activation functions in each hidden neuron, it can solve non-linearly separable problems. The MLP

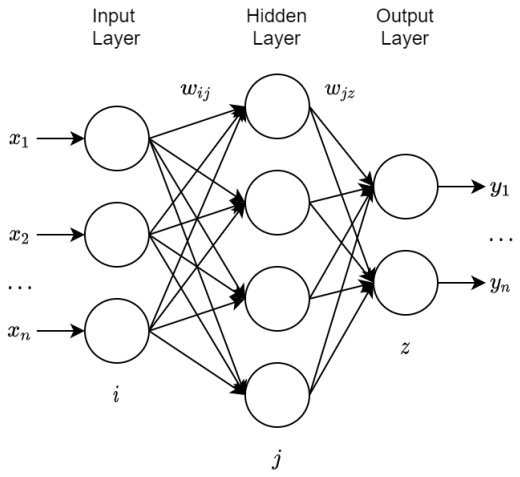


Figure 2.3: Example of multi-layer perceptron with a single hidden layer. x_n represents the variable-sized input and y_n the output. Each line represents a weighted connection for the weight w.

is still used as a standalone model for tasks such as classification and regression. The concepts of the MLP are also seeing much use in Deep Learning through the fully connected layer (FC). Throughout the models in this thesis, these FC layers are utilized.

2.2.3 Activation functions

To introduce non-linearity to ANN, non-linear activation functions were introduced. There are many different activation functions, but they all share two important characteristics: There is no linear relation between the input and output of the function and the function is differentiable. In this section, some of the most common and important activation functions are presented.

The two traditional non-linear activation functions used for MLPs were the sigmoid and tanh functions. The sigmoid is a logistic function following an S-shaped curve (see equation 2.6). It squashes the input to the range (0, 1), thus always producing a positive output. The sigmoid suffers from two problems. The function is not zero-centered, which can cause zigzagging in the learning as the gradients will become either all-positive or all-negative. The other problem is illustrated by its derivative function (see Figure 2.4). For large-valued inputs, the gradient becomes very small, which can lead to the vanishing gradient problem discussed in section 2.2.5.

$$\sigma(x) = \frac{1}{1+e^{-x}} \qquad \sigma'(x) = \sigma(x)(1-\sigma(x)) \tag{2.6}$$

The hyperbolic tangent (tanh) activation function is similar to the sigmoid. It also follows a S-shaped curve, but with a range of (-1, 1), which causes it to be zero-centered (see Figure 2.4 and equation 2.7). This avoids the zig-zagging problem that can occur with the gradients in the sigmoid function. Therefore this activation function can lead to more stable learning.

$$\tanh(x) = \frac{1-e^{-x}}{1+e^{-x}} \qquad \tanh'(x) = (1-\tanh(x)^2) \tag{2.7}$$

An activation function that has become widely used, especially in deep learning, is the Rectified Linear Unit (ReLU) (see equation 2.8). The function has a constant gradient which is undefined at 0. To handle the undefined part a random value can be assigned or chosen from the possible values 0 and 1. The constant gradient causes ReLU to suffer less from the vanishing gradient problem than the sigmoid as large inputs result in the same gradient, unlike the sigmoid where this causes small gradients. Another benefit of ReLU is sparsity in the network caused by inputs lower than 0

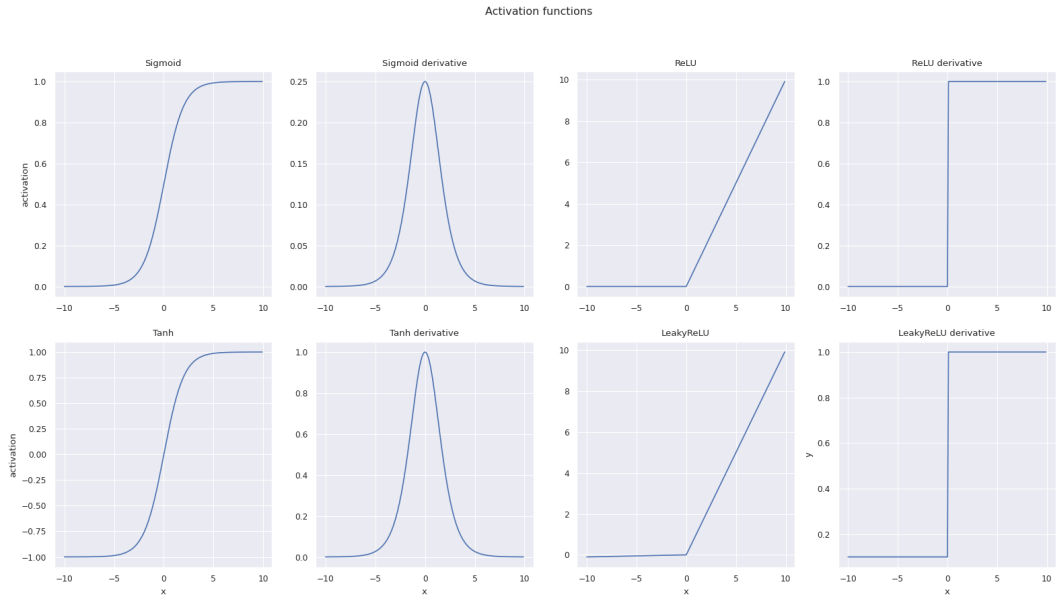


Figure 2.4: Plots of the different activation functions that were discussed with their corresponding derivatives. For the LeakyRelU graph, the c value was set at 0.1.

effectively turning off the neuron. Sparse representations have been shown to perform better than dense [13]. While ReLU suffers less from the vanishing gradient problem, a similar problem can occur. A neuron may become inactive for all inputs. If this happens its gradients will always be 0 causing the neuron to effectively die out, as it will no longer learn. To solve this problem, a small amount of the input can be leaked for $x < 0$. This concept is called the LeakyReLU (see equation 2.9). The small gradient allows the neuron to recover from that state. See Figure 2.4 for plots of ReLU and LeakyRelu.

$$\text{ReLU}(x) = max\{0, x\} \qquad \text{ReLU}'(x) = \begin{cases} 0 & x < 0 \\ 1 & x > 1 \\ undefined & x = 0 \end{cases} \tag{2.8}$$

$$\text{LeakyReLU}(x) = max\{cx, x\} \qquad \text{LeakyReLU}'(x) = \begin{cases} c & x < 0 \\ 1 & x > 1 \\ undefined & x = 0 \end{cases} \tag{2.9}$$

The last activation function discussed here is the Softmax, which is quite different from the others mentioned before. It takes a vector as input and normalizes its values to produce a probability distribution vector. The values lie in the range (0, 1) and sum to a total of 1. Softmax is commonly used as a final layer to produce prediction probabilities over a set of targets.

$$\text{Softmax}(\vec{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{2.10}$$

2.2.4 Loss functions

For learning to be possible in ANNs there has to be a measure of the error on the output it produces. The larger the error, the worse the network is performing. This error is determined through loss functions by capturing the difference between the predicted output $\hat{y}$ and the target

output y through a cost. In gradient descent (see section 2.2.5), the loss function is used to move the model's parameters towards the minimum loss point in the loss landscape. Therefore their design is vital to the learning ability of a network. Also, as gradient-based learning is used, the loss function has to be differentiable. In this thesis, different loss functions are used depending on the task. The mean squared error (MSE) is a loss function designed for continuous numerical output commonly used for tasks such as regression. It is defined by equation 2.11. The lower the value the closer the target is to the output.

$$\mathrm{L}_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.11}$$

For classification problems, the network outputs probabilities instead of a continuous numerical target and therefore requires a different loss function. In this case, cross-entropy (CE) loss is used. CE loss is based on the statistical concept of entropy, which measures the level of uncertainty in a random variable. This concept is used to calculate the difference between two probability distributions. For multi-class classification, CE is used in tandem with a final softmax activation layer as softmax produces a probability distribution. CE is defined by equation 2.12 where y_i is the target probability and $\hat{y}_i$ the predicted probability for the ith class C. For binary classification problems, binary cross-entropy (BCE) is used. The BCE equation is the same as CE with the number of classes C set at 2.

$$\mathrm{L}_{CE} = -\sum_{i=1}^{C}(y_i \log \hat{y}_i) \tag{2.12}$$

2.2.5 Backpropagation & Optimization

To update the weights in the perceptron a simple learning rule was used that moves the decision boundary based on a learning rate and the error. This algorithm is a simplified form of gradient descent that works for networks with only an input and output layer. In the more general form, gradient descent is an optimization algorithm that minimizes a function by iteratively moving towards a negative gradient until it reaches a global or local minimum. The learning rate determines how large the steps are in the optimization process. The correct value of this hyperparameter is crucial for stable learning. Ideally, the steps are small and controlled. If the learning rate is set too high the optimization can become unstable, overshooting minima. If set too low the steps are not strong enough to find the right trajectory towards a minimum.

To perform gradient descent-based learning the loss function is minimized by changing the weights in the direction of their gradients with regard to the loss function. To calculate the gradients the backpropagation algorithm is used. The algorithm utilizes the chain rule to calculate the gradient through the partial derivative of the loss with regard to each weight. By propagating the gradients backward from the final to the first layer this calculation is possible. The calculated gradients for each weight then represent how much contribution they had to the final output and can be used to move them in the right direction. The weight update for a weight at w_{ij} with regards to loss E and a learning rate η follows equation 2.13, similar to the perceptron learning rule from equation 2.5 [39].

$$w_{ij} = w_{ij} + \nabla w_{ij} \qquad \nabla w_{ij} = -\eta\frac{\partial E}{\partial w_{ij}} \tag{2.13}$$

There are two problems that can occur with the gradients in backpropagation. The vanishing gradients problem occurs when gradients are too small, effectively vanishing as they are passed through the network by backpropagation. This causes the network to train much slower than is desired, especially in the first layers. In the worst case, this stops the network from training as the gradients are too small to effectively update the weights. The vanishing gradient problem is commonly solved through network design (e.g. the LSTM in section 2.2.7). Conversely, the exploding gradients problem occurs when the gradients are too large. In this case, the weight

updates are too large, resulting in unstable learning. The exploding gradient problem is commonly solved through methods such as gradient clipping [34].

In gradient descent, each training data sample has to be considered before the update is done. For large data sets, this takes a considerably large amount of time. Therefore, it is common to use the faster stochastic gradient descent (SGD) method. In SGD a random training example is chosen and processed to update the network. This has the benefit of converging faster and taking less training time than using classic gradient descent. In practice, often a mini-batch is used to improve the speed even more and smoothen the training process.

In recent years more advanced optimizers have been introduced as well. The most widely used of these new optimizers has been Adam [24]. It computes individual adaptive learning rates for the weights by scaling the global learning rate with exponential decay rates based on the square and running average of past gradients. In addition to the global learning rate, it has a β_1 and β_2 parameter. The β_1 parameter sets the exponential decay rate for the past squared gradients and β_2 for the running average. Following most of the recent literature, the Adam optimizer is used throughout this thesis [9].

2.2.6 Convolutional neural networks

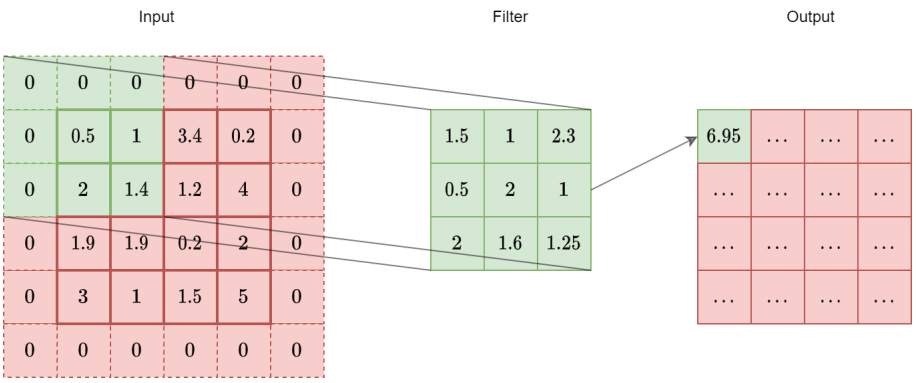


Figure 2.5: Example of a 2-dimensional convolution using a stride of 1 with an input feature map of size 4x4, filter of 3x3, and output feature map of 4x4. The input is zero-padded, which is indicated by the cells that have dotted lines. This results in the input and output feature maps having the same dimensions as shown.

Multi-layer perceptrons (MLP) are very effective at extracting features from an input vector but are far from ideal for the high-dimensional nature of image data. As each pixel requires a weight, the amount of parameters required quickly explodes as image size increases and color information is used. In addition, the MLP does not utilize the spatial information in the image and fails to be translation invariant. To handle high-dimensional data like images convolutional neural networks were invented (CNN). They can extract translation-invariant descriptive image features while using much fewer parameters through weight sharing.

A CNN extracts features by combining convolutional layers with pooling layers to go from low-level features, such as edges, to high-level such as specific objects. In each convolutional layer, a kernel is defined. The kernel defines a set of filters with a corresponding learnable weight matrix. The kernel size defines the receptive field of the filters and is often set to have equal width and height for image data. For each filter in the kernel, the convolution between the filter and each input feature map is calculated and summed together to produce a new feature map. Therefore, the amount of feature maps a kernel produces is equal to the filter size of the kernel. To perform the convolutional operation the filter is slid across the input, calculating the convolution of the kernel with each point it passes. The stride parameter determines how many points the filter moves in between convolution steps. Larger values than one result in points being skipped causing

down-sampling of the input. The convolutional operation cannot be performed on border pixels, so these pixels can either be skipped or padded to enable the operation. Skipping the border pixels means the output is down-sampled. Therefore to avoid down-sampling, padding is required. An example of convolution operation can be seen in Figure 2.5. Each filter affixes to certain features that exist in the data. For example, filters in the first layer affixes to low-level features such as edges and lines.

In most CNNs, a pooling layer is applied after a convolutional layer. The pooling layer down-samples the input feature maps, summarizing them in the process and reducing subsequent parameter amounts. It is performed through a kernel with a stride larger than one and a single filter designed to perform an operation such as averaging or maxing. The output of the final convolutional layer is flattened and represents a feature vector describing the input. To perform tasks such as classification this feature vector is passed through an MLP with softmax output to predict classes.

There has been much research into CNN architectures, with the ImageNet [40] large scale visual recognition task serving as a general benchmark of their performance. The state-of-the-art architectures on this task extract visual features which are general enough such that they can be utilized by a new network on a different task [26]. Instead of training it from scratch, the feature extraction part from one of these architectures can be taken and used to extract visual features, possibly fine-tuning it some more on the task at hand. This concept falls under transfer learning and can save a lot of training time. It can even improve the generalizability of the model as it forces the model to learn from general features instead of data set-specific ones. The use of transfer learning this way has become commonplace in computer vision tasks and is used in this thesis as well.

In this thesis, three networks that were state-of-the-art on the ImageNet task at some point are used for this purpose. The first is VGGNet, which achieved state-of-the-art performance at the time by using a large-scale CNN [44]. It achieved this by effectively utilizing 16-19 (dependant on the setup) convolutional layers by using a small filter size of 3x3. Next is ResNet, which improved upon VGGNet and by introducing skip-connections that allow alternative paths for gradients to flow [18]. The skip-connections reduced the vanishing gradient problem that occurs with very deep networks enabling them to utilize many more layers. These skip-connections are also used in network architectures presented in section 2.3.3 and 2.6. Last, the Inception-V3 network [47], which builds upon the previous Inception [46] network by adding an auxiliary classifier and asymmetric convolutions among other things. The original Inception network introduced the Inception Block which applies multiple different sized convolutional filters to the input, concatenating their output. The design of this block allows multi-scale processing at every convolutional stage of the model.

2.2.7 Recurrent neural networks

The MLP is only able to handle fixed-length input and fails to recognize the temporal context in a sequence. To deal with the temporal and variable-length nature of domains such as language and speech, the Recurrent Neural Network (RNN) was invented. An RNN processes input sequentially, keeping track of the previously seen information in a hidden state. It can use the hidden state to produce a different kind of output dependant on the RNN architecture. There are four general setups of RNNs: one-to-many, many-to-one and two possible many-to-many setups. These different setups are visualized in Figure 2.6. For each setup, there is a set of weights for the input, hidden state, and output shared between repeated nodes. In one-to-many, the RNN receives a single input, which it uses to produce a sequence of outputs. This setup is used for sequence generation from a single input such as a condition. In many-to-one, an input sequence is processed to a single output. Often this is used for tasks involving classification or regression based on a sequence. In many-to-many, the RNN maps an input sequence to an output sequence. This can be a sequence of the same length, effectively producing an output element for each element in the input sequence, thus being a direct mapping. It can also be produced after seeing the full input sequence first. In this case, the input and output sequences do not have to be of the same length

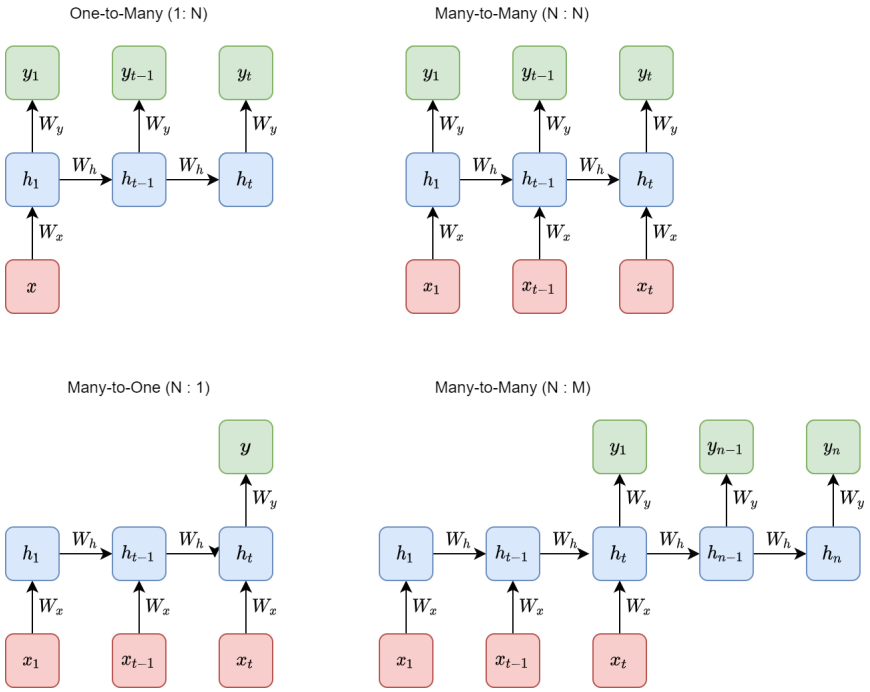


Figure 2.6: The four general setups of recurrent neural networks. In the figure, x is the input, h is the hidden state, and y is the output. W_x, W_h, and W_y are their respective weight matrices. One-to-many is used for pattern generation. Many-to-one is used for sequence classification. The two setups of many-to-many are used for an ordered sequence to sequence transform in the N : N case and an unordered sequence to sequence transform in the N : M case.

producing an unordered sequence. At each time point that has an output, the loss of the target and output element at that time point is determined. This loss is backpropagated through each time point.

The classic RNN can handle sequences but fails to recognize long-term dependencies in longer ones. For these longer sequences, the vanishing or exploding gradient problems are also likely to occur. These happen due to the many gradient multiplications which can cause exponential growth or decay. It can make the training significantly harder. A general solution exists for the exploding gradient problem is gradient clipping (see section 2.2.5). Such a general solution does not exist for the vanishing gradient problem, thus requiring a change in the architecture. To solve this and also consider long-term dependencies in the sequence, the Long Short-Term Memory (LSTM) [20] cell was invented. The LSTM cell is visualized in Figure 2.7. An LSTM cell is composed of an forget f_t, input i_t, cell g_t and output o_t gate. In addition to the hidden state h, a context state c is passed throughout the RNN. Each gate has weights for the hidden state and the input, which are weighted and added together as input. The forget gate is designed to determine the amount of context to take into account from c_{t-1}. It calculates a coefficient for c_{t-1} by using a sigmoid activation on the weighted hidden state and input. This coefficient is then multiplied with the c_{t-1} state. Similarly, the input gate determines how much input should be considered by using a sigmoid activation. The resulting coefficient is multiplied with the output of the cell gate, which uses tanh activation to keep the sign information from the input. The result of this is added with the result of the multiplied c_{t-1} state to determine the output context c_t. The context c_t is also passed through a tanh activation to be multiplied with the output of the output gate, which uses sigmoid activation to determine how much context should go to the hidden state output h_t. The h_t and c_t are used for the next LSTM cell in the RNN. The hidden state is also passed to the output if required. The combination of the forget gate and the context ensures vanishing gradients are much less likely.

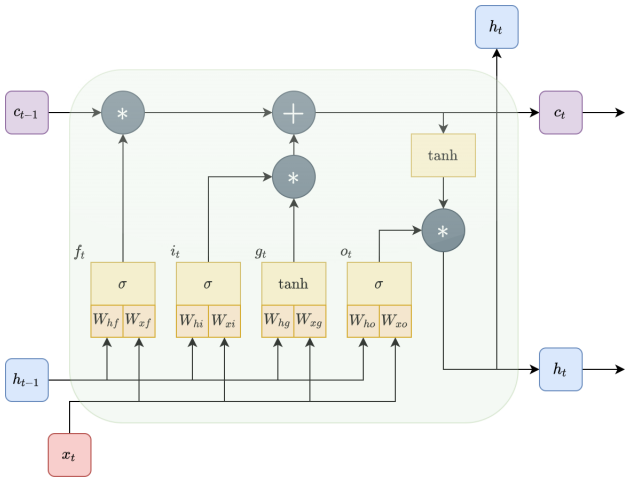


Figure 2.7: The structure of a single LSTM cell for a timepoint t. The input x_t and previous hidden state h_{t-1} are used as input for the forget f_t, input i_t, cell g_t and output o_t gates. The σ block indicates sigmoid actvation. The tanh block indicates tanh activation. Each gate has the input weights W_{hz} for the hidden state and input W_{xz}, where z is a placeholder for the gate indicator, under the activation block. c_{t-1} is the context from the previous LSTM cells. The outputs of the cell are the new hidden state h_t and c_t.

The tracking of the context also ensures the long-term dependencies are considered throughout the RNN. This makes RNNs with LSTM cells very effective for long-sequence problems, such as natural language and speech.

2.3 Deep-Learning architectures

In the previous section, the concepts required to build deep-learning architectures were presented. Deep learning utilizes the increase of computational resources and data available to take these concepts to a higher level by building large-scale neural networks. Over the recent years, there have been many innovations in deep learning architectures. These architectures have enabled machine learning to be performed on ever more complex tasks. The relevant deep-learning architectures for this thesis are presented in this section.

2.3.1 Generative adversarial networks

The Generative Adversarial Networks (GAN) is a generative deep-learning architecture where two neural networks, a generator and a discriminator, are battling in a zero-sum game [14]. The discriminator decides if a data point is real or fake. The generator tries to fool the discriminator by synthesizing data points that resemble the real data. The networks are playing a min-max game. The discriminator aims to maximize its ability to discern the data points. The generator tries to minimize the discriminator's ability to do this. By simultaneously training both networks, the generator learns to estimate a mapping of the real data distribution from a latent space. The discriminator act as a kind of critic to how realistic the synthesized samples are. Formally, the discriminator D is trained to maximize the probability of assigning the correct label to samples from the generator G and the real data x, thus maximizing $\log D(x) + \log(1 - D(G(z))$. The generator is trained to minimize the probability the discriminator assigns the correct label to its synthesized samples from the latent space z, thus minimizing the $\log(1 - D(G(z))$ part from the discriminator. To avoid saturation in the generator, the loss function is often framed as maximizing the probability the discriminator classifies its samples as real, thus becoming $\log(D(G(z))$ [14]. The architecture is visualized in Figure 2.8. The GAN learning process is unsupervised because

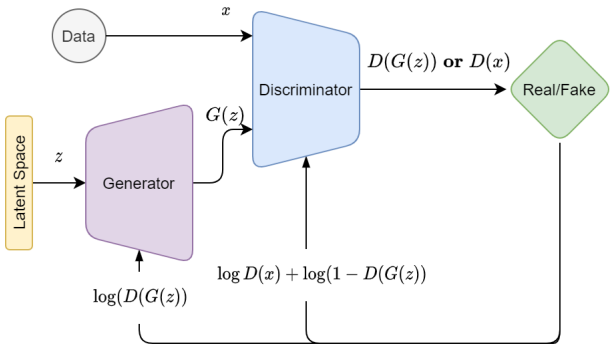


Figure 2.8: A generative adversarial network architecture with the generator G and discriminator D. The generator synthesizes an image from the latent space, which the discriminator assigns a real/fake label too. Also included are the loss functions of the generator and discriminator based on the assigned label of the discriminator.

the real data distribution is indirectly learned by the generator. However, the generator and discriminator are learned in a supervised manner by assigning the known real or false labels to the samples and using a binary cross-entropy loss for both models.

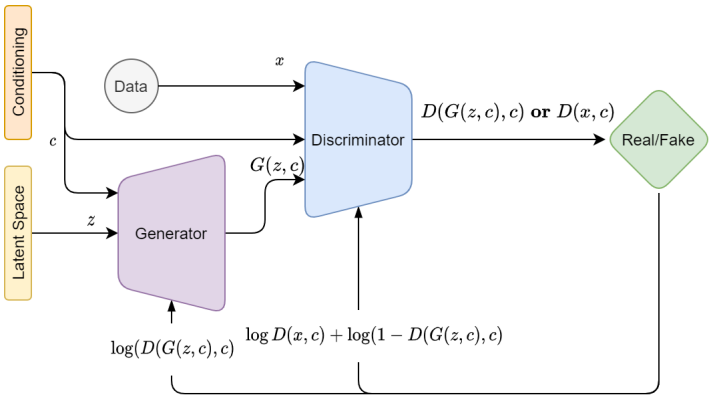


Figure 2.9: A conditional extension of the GAN architecture which introduces a conditioning variable c to the inputs of both the generator and discriminator.

The original GAN architecture is an unconditional generative model. This means that there is no direct control on the synthesized samples. The image generation task in this thesis requires the model to synthesize an image from a question and answer. Therefore the generation needs to be conditioned on the question and answer. For this purpose, an extension of the original architecture called conditional GAN (cGAN) was proposed [32]. They introduce a conditioning vector that is concatenated with the input of the generator and discriminator (see Figure 2.9). The conditioning variable can be a one-hot encoded label, a text embedding vector or anything else relevant. It directs the generation to be based on the conditioning while keeping the variability from the latent space. The GAN architectures used throughout this thesis utilize the cGAN approach.

When the GAN training is smooth and stable these networks can be very effective, but this is often not the case. GAN training is very sensitive to the choice of hyperparameters and the network design. For the model to converge, the networks must go back and forth in a stable manner, with neither network overpowering the other. When one network overpowers the other too much, learning will cease to happen. The generator overpowering the discriminator is not something that often occurs. However, the discriminator overpowering the generator is [51]. When this

happens the generator gradients vanish, causing the generator to get stuck in the loss landscape, ultimately failing the training. To solve this the discriminator can be handicapped such that it does not overpower the generator too fast. A method for this is adding noise to the inputs of the discriminator [6]. The samples generated are often noisy at the start of training, which the discriminator can abuse to discriminate between the real and fake samples. Adding this noise avoids the discriminator abusing this noise.

Another problem that can occur within GANs is mode collapse. This is a problem that affects the diversity of the synthesized samples. It happens when the generator learns to map multiple points in the latent space to the same output sample, not utilizing all the modes of the real data distribution [45]. It causes low diversity of the synthesized samples, as the same ones are synthesized multiple times. Especially for non-conditional GANs, this is a problem as there is no guiding variable. However, it also occurs frequently in cGANS. Mode collapse is a hard problem to solve but is generally best improved by modifying the structure of the GAN training [45, 16].

2.3.2 Variational Autoencoders

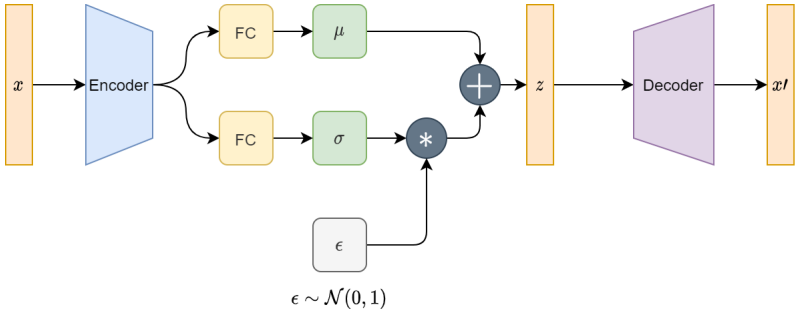


Figure 2.10: The components of a Variational Autoencoder. The encoder encodes the input vector x into the μ and σ vectors through seperate fully-connected (FC) layers. These are used by the reparametrization trick to sample the latent vector. The decoder reconstructs the input x' from the latent space.

Autoencoders (AE) are networks that learn to reconstruct an input vector from a lower-dimensional representation in an unsupervised manner. They work by encoding an input vector x to a lower-dimensional latent space $z = e(x)$ and subsequently decoding the latent space back to the original dimension $x' = d(z)$. By optimizing a reconstruction loss between the input and the output it learns the most efficient way to represent the input in a lower dimension. The latent space of the AE is not suitable for enforcing variation in the reconstruction, as the encoding has no inherent variation. To this end, the Variational Autoencoder (VAE) was invented as a generative extension of the AE architecture. Instead of encoding the input to a latent vector, the VAE encodes the input to two separate vectors that represent the mean μ and standard deviation σ values of a set of random variables. To sample the latent vector the σ is multiplied by a sampled standard normal distribution $\epsilon \sim \mathcal{N}(0, 1)$ and added to the μ to produce z [25]. This process is called the reparameterization trick and is visualized in Figure 2.10. It introduces variation to the latent vector, which the decoder uses to produce a decoded input the same way as in the regular AE. The VAE introduces an additional loss based on the Kullback-Leibler divergence between the μ and σ and the standard normal distribution (see equation 2.14). The Kullback-Leibler divergence measures the difference between two probability distributions. Therefore, it is minimized to ensure the distributions of μ and σ are close to the normal distribution. The generation in the VAE can be conditioned the same way as the cGAN; By concatenating a conditioning variable to the latent vector. These modifications to the regular AE allow it to generate more diverse samples in an unsupervised manner.

$$KL(\mathcal{N}(\mu, \sigma), \mathcal{N}(0, 1)) \tag{2.14}$$

The decoder component of the VAE is very similar to the generator component used in the GAN, as they both utilize a latent vector to generate a higher-dimensional sample. A problem for GANs mentioned in section 2.3.1 is that the discriminator can overpower the generator. To avoid this the generator of the GAN could be pre-trained using a VAE as proposed in [17]. This gives the generator a better starting point, ideally leading to more stable training. In addition, they found that it reduced mode collapse, which also is a problem for GANs mentioned in section 2.3.1. They achieved increased image quality on an unconditional image generating task. In this thesis, conditional image generation is done. Therefore, the VAE is modified to conditionally generate by concatenating a conditioning vector with the latent space in the architecture.

2.3.3 Transformers

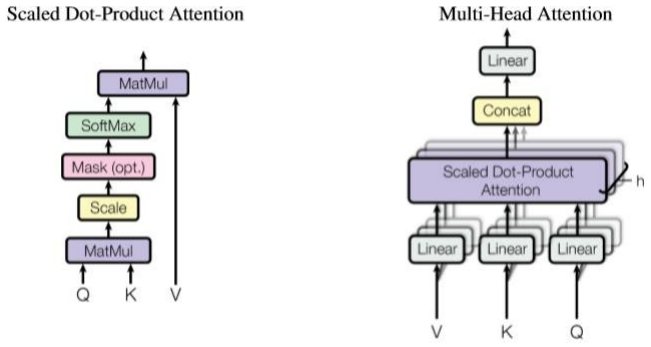


Figure 2.11: **Left**: Scaled dot-product attention, the self-attention mechanism used for transformers in [49]. It uses the query Q, key K, and value V vectors to perform the self-attention. **Right**: The multi-head-attention block combining parallel self-attention heads by concatenating their outputs and passing this through a fully connected layer. Note that the output sizes of the scaled dot-product attention and multi-head attention are the same. Figure taken from [49].

The transformer architecture is an encoder-decoder sequence transduction model that achieved state-of-the-art performance on multiple NLP tasks through a multitude of innovations [49]. It utilizes a self-attention mechanism, which attends each element in the sequence to the other elements, relating the different elements. Being able to relate different elements of a sequence is important for understanding context in sentences. For example, in the sentence "The dog was afraid of the cat because it was angry." the only way to understand what it refers to is by looking at the other words in the sentence. Self-attention learns to find these kinds of relations in the sequence. It works by calculating a query Q, key K and value V matrix for the input embeddings through fully connected layers. Each row represents the Q, K and V vector for the input embedding at the same position. The Q and K matrices are multiplied and divided by the square root of the K dimension. Then softmax is applied to this to determine the attention coefficients. These are multiplied with the value matrix to result in the attended embedding. To jointly attend to the sequence in different subspaces, the self-attention mechanism is performed multiple times in parallel through a method called multi-head attention. Multi-head attention concatenates the different self-attention heads and scales them down with a fully connected layer. See Figure 2.11 for a visual example of this process.

In their original architecture, there are six sequential identical encoder layers. The encoder component consists of a multi-head attention and feedforward (fully connected) sub-layer. Each sub-layer is followed by an addition and normalization layer. This layer adds a residual connection

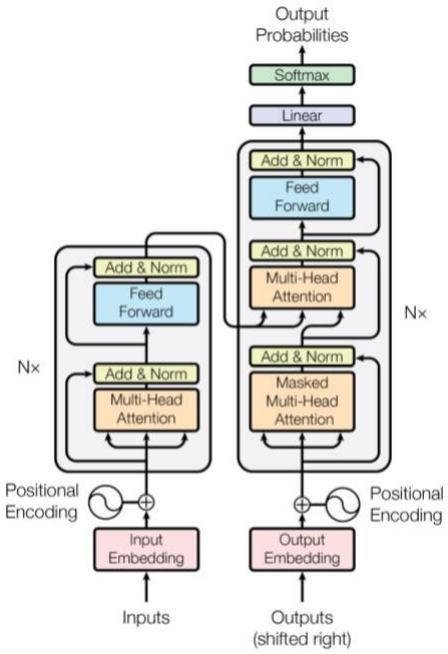


Figure 2.12: The structure of the transformer architecture. The left block represents the N encoder layers. The first encoder layer receives the positionally encoded input embedding as input. Subsequent encoder layers receive the output of the previous one as input. The right block represents the N decoder layers. The first one receives the positionally encoded output embedding and the final encoder output as inputs. Subsequent decoder layers receive the output of the previous one in addition to the final encoder output. Figure taken from [49].

[18] to the output of the sub-layer and applies layer normalization [7]. The fully connected sub-layer applies the linear transformation to each position separately by the sharing weights. The input to the first encoder layer is a matrix consisting of the input elements embedded into vectors. These are also positionally encoded to preserve the ordering of the sequence. The input to the following encoder layers is the output of the previous encoder layer.

An equal amount of decoder layers are present in the model. They have a similar structure to the encoder layer but insert an additional multi-head attention layer between the sub-layers, which uses K and V matrices calculated from the output of the final encoder layer and a Q matrix calculated from the previous layer. The first multi-head attention layer is also changed to apply masking of future positions before the softmax calculation (see Figure 2.11). The input to the first decoder layer is the positionally encoded output elements shifted one spot right and embedded. The shifting and masking ensure that the predictions for an element only depend on the previous elements. The output of the final decoder layer is passed through a fully connected and softmax layer to determine the output sequence. All these components describe the full transformer architecture visualized in Figure 2.12. It has been used as a basis for many recent NLP models. One of these is used in this thesis and discussed in section 2.4.3.

2.4 Natural language processing

Processing the question and understanding the textual information in the VQA and IG tasks requires natural language processing (NLP). NLP is a sub-field of AI concerned with the processing and analysis of natural language by computers. Language is built from the atomic level of individual characters to the higher level of words, phrases, sentences and eventually full texts. Much

of this is governed by a set of specific rules, conventions and principles. However, there are many ambiguities when it comes to the understanding of language. Words and phrases can have multiple meanings that depend on the context of the text or even the external context. Fully comprehending sentences requires a high-level understanding of the connections between the words and the context. Language also changes over time new words are introduced, the meaning of words are changed, rules are changed, etc. These aspects make NLP quite challenging for computers.

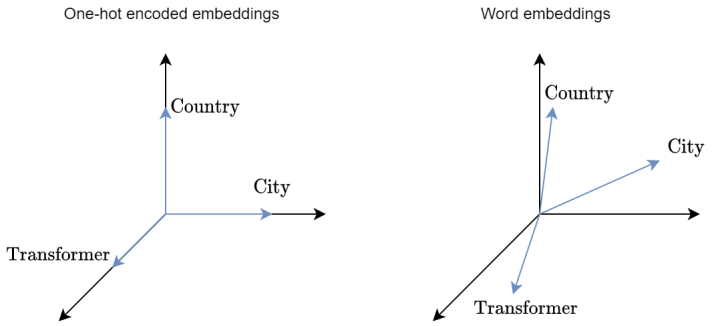


Figure 2.13: Example of the lack of semantic information in one-hot encoded embeddings for a three-word vocabulary. The vectors for one-hot encoding lie equally far apart from each other. In word embeddings, the vectors country and city, which are related, lie closer together than the other unrelated word.

A computer requires language to be in a format it can understand to process it. Neural networks only work with numerical inputs and thus require language to be transformed to a numerical format. This format is called an text embedding. The simplest form is a one-hot encoded embedding, where each element of the vector encodes the presence of a word. It works as a base method but fails to capture the meaning of language as exemplified in Figure 2.13. Ideally, semantically similar vectors should lie close together in the vector space, while dissimilar ones should lie further apart. To solve this more extensive embedding methods have been invented. These utilize different levels of language such as the character, word or sentence level. To generate a text embedding the text first has to be tokenized. Depending on the method this can be into words, characters or other levels. Then the embedding method utilizes the relations between these tokens to learn how to embed text. In this thesis, sentence embeddings are used to vectorize the question and answer. Three different methods are applied, which build the sentence embedding from different levels of language information. These are presented in the following sub-sections.

2.4.1 Bag of words

Bag of words (BOW) is one of the simplest ways to represent text as a vector. It works on a vocabulary of all the tokenized unique words in the corpus. Each word in the vocabulary is attached to a feature index in the embedding. An additional feature index is added for out of vocabulary (OOV) words. To embed a sentence the feature index corresponding to the vocabulary index of each tokenized word is incremented. This results in a sparse vector containing counts only for the words that exist in the sentence. Tokens that are not recognized are added to the OOV feature index. The larger the vocabulary, the larger the resulting text embedding dimension, as each word is a feature. It can be regarded as an extended version of one-hot encoded word embeddings to a sentence level.

BOW builds a sentence embedding from the word level without considering the context. This reduces the semantic accuracy of the embedding when words require this context. It is also not robust against linguistic variations or unseen words, as it can only match words from its own limited vocabulary. The main benefits of BOW are that it is simple, easy to understand and quick.

2.4.2 Pyramidal Histogram of Characters

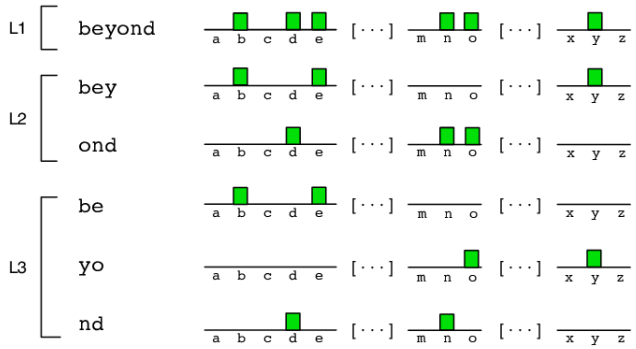


Figure 2.14: Example of the PHOC embedding process for the split levels L1, L2, and L3. At each level, the word is split into sub-segments to build character histograms for different sub-segments of the word. These character histograms are concatenated to determine a word embedding. Figure taken from [2].

A different embedding approach is the pyramidal histogram of characters (PHOC), which was proposed as a text embedding for a handwriting recognition task [2]. It encodes a word into histograms of its alphabetical characters at different levels. At each level, the word is split into 1 to n equally-sized segments. For each segment, a histogram of alphabetical characters is calculated resulting in a vector of 26 dimensions. The levels 1 to 3 are visualized in Figure 2.14. All the histograms are concatenated to create the final word embedding. In [2] the levels 2, 3 and 4 were used resulting in a dimension of $(2 + 3 + 4) * 26 = 234$. To generate a sentence embedding the word embeddings for each tokenized word are averaged together.

PHOC works at the character level by encoding how much and where particular characters exist in the word. For example, the final histogram in the 3rd level measures the last few suffix characters of the word. By working at the character level, it is much more robust to linguistic variations. This makes it more suitable for real-world scenarios where linguistic variations are very likely. Like BOW, it does not consider the context of the individual words in the sentence. Its strength thus lies in the robustness to linguistic variations and also its simplicity and speed.

2.4.3 Sentence-BERT

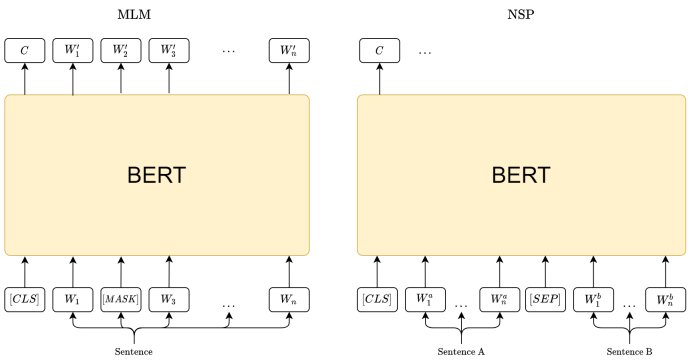


Figure 2.15: **Left**: The masked language modeling (MLM task performed by BERT predicting W_2' for the masked second token. **Right**: The next-sentence prediction(NSP) task performed by BERT predicting the classification token C, which determines if the sentences belong together or not.

A more advanced embedding method utilizes the Bidirectional Encoder Representations from Transformers (BERT) model, which has seen considerable popularity for a multitude of NLP tasks [12]. BERT learns a language model from a corpus in a bidirectional fashion by utilizing the encoder of a multi-layer transformer architecture (see section 2.3.3). It learns this by applying the two training tasks visualized in Figure 2.15. The first method is masked language modeling (MLM). The model is trained to predict a masked part of an input sequence. It does this by first encoding the masked sequence and then using the hidden encodings of the elements to predict the masked ones. This task requires the model to understand the context of the surrounding words to predict the masked part, learning this context in the process. The other training task is next-sentence prediction (NSP). For this task, the model receives two sentences as input. The model has to predict if the second sentence follows the first. It does this by encoding both sentences simultaneously with a $[SEP]$ token in between them. Then the hidden encodings are used to make the prediction. During training, it receives an equal amount of sentences that follow each other and that do not. This method learns the model to understand the higher-level sentence context.

The input sequences require some processing before being used in BERT. A $[CLS]$ token is inserted at the beginning of the sequence. The token stores a hidden state which relates the full sequence. It can be utilized for classification tasks such as NSP. As mentioned, a $[SEP]$ token is inserted between the different segments in the sequence. For the NSP task, this would be between sentence A and sentence B. The input tokens are embedded by the WordPiece method [53], which uses sub-word level splitting to handle out-of-vocabulary words. To help distinguish between the elements of the different sequences a segment embedding is added to each token. The positional encodings, similar to the transformer ones, are also added to the tokens. The total input is visualized in Figure 2.16.

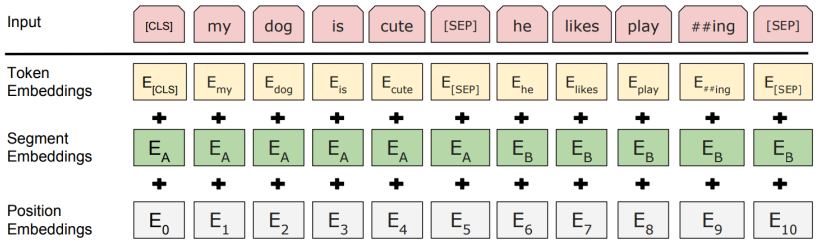


Figure 2.16: The input representation of BERT for the combination of the sentences A "my dog is cute" and sentence B "he likes playing". It is the sum of the token, segment, and positional embeddings. Figure taken from [12].

Through these methods, BERT learns a strong understanding of language. BERT has enabled transfer learning for NLP-related tasks as a viable alternative to training a model from scratch. The BERT model can be pre-trained on a large corpus and finetuned to perform downstream tasks on a different corpus. The inner representations of BERT can also be used to extract contextualized word embeddings. This is done by combining the hidden outputs of the encoders for the word. Sentence embeddings can be constructed by combining the hidden outputs for each word or using the [CLS] token output. However, this has proven to be an ineffective sentence embedding method in [36]. To improve this, they propose an extension of BERT called Sentence-BERT (SBERT) that achieves much better sentence embeddings. To achieve this, the SBERT extension finetunes a BERT model on a sentence-level language task. Multiple setups of the SBERT model exist, but only the one used in this thesis is described. It finetunes on a natural language inference task using both the SNLI [10] and Multi-Genre NLI data sets [52]. These data sets contain annotated sentence pairs with a label describing their relation. The DistilBERT model [42], a smaller and faster BERT model, is used as a base. A pooling method is added at the end of the BERT model to calculate a sentence embedding. The pooling method takes the mean of the output tokens to calculate this. The model is finetuned by optimizing a classification objective function on the sentence pair label. The classification is achieved through a siamese network structure (see

Figure 2.17. Each sentence is embedded through the BERT model separately. The embeddings are concatenated together with their element-wise difference and passed through a fully connected and softmax layer to determine the classification. The model is trained to optimize the classification. Finetuning the model this way significantly improves the semantic meaningfulness of the sentence embeddings generated by the base BERT model used.

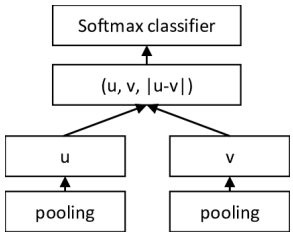


Figure 2.17: Siamese SBERT architecture for classification. The output of the pooling layers from two separate, but identical BERT networks are concatenated with their element-wise difference. A softmax classifier performs the classification from this embedding. Figure taken from [36].

The SBERT embeddings work at a higher level than the two previously mentioned methods. Unlike these, it takes the context and relations between words in the sentence into account. Therefore its embeddings have a stronger semantic value, which can be very beneficial. However, to achieve this it trades in the speed and simplicity seen in the other methods.

2.5 Visual question answering

The previous sections covered the theoretical background towards the methods used in the visual question answering (VQA) task. VQA is the task of answering free-form open-ended questions about an image. Formally, the task requires a model to learn the mapping $(Q, I) \rightarrow A$ of a question Q and image I to an answer A. To come up with a correct answer, the model has to combine visual and language understanding. Throughout most VQA research, there have been three common principles to approach the task. According to the first principle, the task is treated as a supervised-learning problem, which means pairs of questions and images with corresponding target answers are used to train a model. The second principle entails that the task is treated as a classification problem instead of a generation problem. This means that a probability distribution on the most likely answer is predicted instead of generating an open-ended answer. Ideally, such an open-ended answer would be generated, as this captures the diversity of the real world. However, this would make the task even more complex than it already is for a neural network. Also, in some of the most important data sets used for VQA research [27, 5, 15], the answers generally come from a relatively small subset of phrases. Therefore the set of possible answers can be limited, enabling a classification approach where a forced-choice out of N possible answer sentences is realized. Approaching the task this way was a good choice to initiate research in this difficult area. The third principle concerns the use of deep neural networks following a joint-embedding architecture, where a joint-embedding of the visual and language features passed through a classification architecture predicts the probability distribution of the answers. An embedding aims to map a high-dimensional space, such as an image, into a lower-dimensional latent space that can be effectively used by a neural network. This lower-dimensional space should capture the most important aspects of the higher-dimensional space in a set of representative features. By mapping both the visual and textual parts of the VQA problem to a common space their interactions can be combined and processed together. Following these three principles allowed researchers to start studying the task of VQA. The theory behind the joint-embedding architecture and an extension that incorporates visual attention are presented in the following sections.

2.5.1 Joint embedding

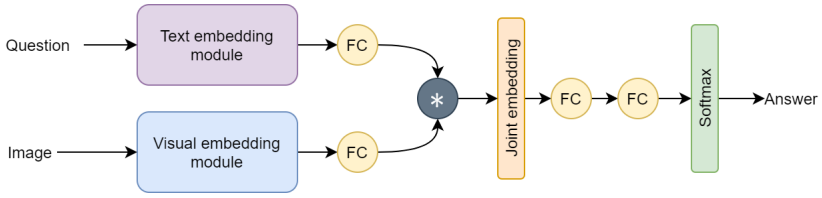


Figure 2.18: The structure of the joint embedding approach towards visual question answering. The visual and text embedding modules are separate models or methods which are interchangeable. FC represents a fully connected layer.

In the joint-embedding approach to VQA, a classification structure is stacked up on a joint-embedding of the visual and textual embeddings to produce an answer. The visual and textual embeddings can be combined in a myriad of ways. Most prevalent is first mapping the embeddings to a common space by separate fully connected layers. Then, combining this through a method such as element-wise multiplication. The visual and textual embeddings are extracted from the image and question by separate models. The classification structure generally consists of two fully connected layers with a final softmax layer. This takes the joint-embedding as input to predict the most likely answer. A schematic view of this architecture is visualized in Figure 2.18.

The joint-embedding model utilizes a simple architecture. Therefore, the main design elements lie in the models that extract the embeddings and the way these are combined. The model only utilizes the information that is present in the embeddings. Therefore these embeddings must contain highly informative features. To extract visual embeddings CNN architectures (see section 2.2.6) are utilized by flattening the output of the final convolutional layer in the architecture. Instead of training a CNN from scratch, often large-scale pre-trained CNN architectures are used for their general and informative features. To extract textual embeddings, a prevalent method in the literature has been the LSTM (see section 2.2.7). An LSTM is pre-trained on contextualized word embeddings of the questions to use the hidden state as an embedding. One of the first joint-embedding approaches was presented in [5]. They utilized the final layer of VGGNet [44] to extract image features in combination with a two-layer LSTM to extract language features. A network similar to this is explored in this thesis. However, instead of the LSTM, a couple of different textual embedding methods are utilized. These were outlined in section 2.4. In the next chapter, the explored setups of visual and text embedding modules are outlined.

2.5.2 Bottom-Up and Top-Down Attention

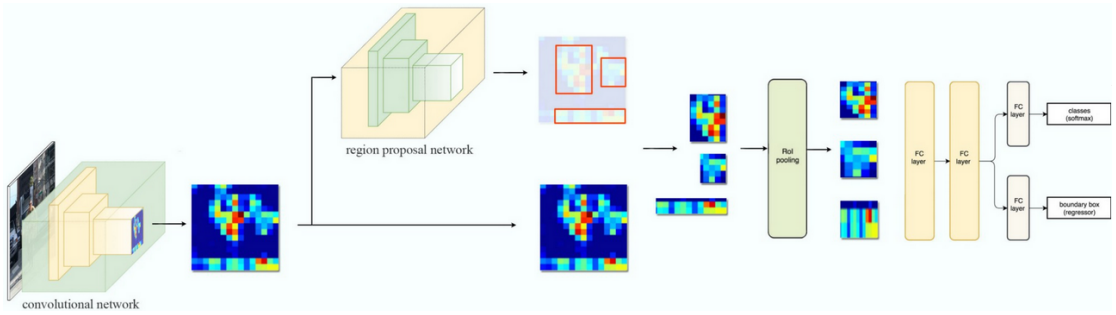


Figure 2.19: The general structure of the Faster R-CNN network. The convolutional network extracts feature maps from the image used by the region proposal network to generate bounding box proposals. The proposals are pooled and further processed to determine a class distribution and bounding box per proposal. Figure taken from [33].

To improve upon the relatively simple joint-embedding approach researchers started experimenting with adding attention mechanisms to these models. In [56] they proposed utilizing two stacked attention layers to make the model attend to the most important visual features based on the question embedding. This kind of attention is called top-down attention. To further improve upon this bottom-up attention was added in [4]. In bottom-up attention, the aim is to select only the most salient regions to extract features from. The regular approach extracts all the features from an image, disregarding their relevance. Often large parts of an image can be irrelevant. Therefore, only extracting the most relevant features can be beneficial. To achieve this a different type of CNN called Region-based CNN (R-CNN) is used. The R-CNN was originally designed for object-detection tasks, which naturally require finding the most salient regions to locate and segment the objects. In [4] the Faster R-CNN model (FRCNN) [37] was used for bottom-up attention. The general structure of the FRCNN is outlined in Figure 2.19. It works by first extracting convolutional feature maps from the image. These feature maps are used by a region proposal network (RPN) to find the regions of interest (ROI). The RPN uses a sliding window over the convolutional feature maps. At each sliding window location, multiple region proposals are predicted at different scales and aspect ratios. For each region proposal, its features are extracted and used to predict a class-agnostic objectness score and a bounding box. Non-maximum suppression (NMS) is used to filter region proposals with similar bounding boxes. NMS works by selectively filtering bounding boxes based on their classification scores and the intersection-over-union between them, resulting in the most unique and confident proposals being left over. These ROI proposals are pooled to fixed-size feature maps and mapped to a feature vector through two sequential fully-connected layers. The feature vector is used to predict a distribution over a set of classes and a class-specific bounding box. Again NMS is performed, this time to select the top k proposals based on the estimated classification confidence. After all these steps, the feature maps kxd corresponding to the top k proposals represent the most salient regions. Effectively this is a hard-attention mechanism as only the set of salient regions is selected and further used.

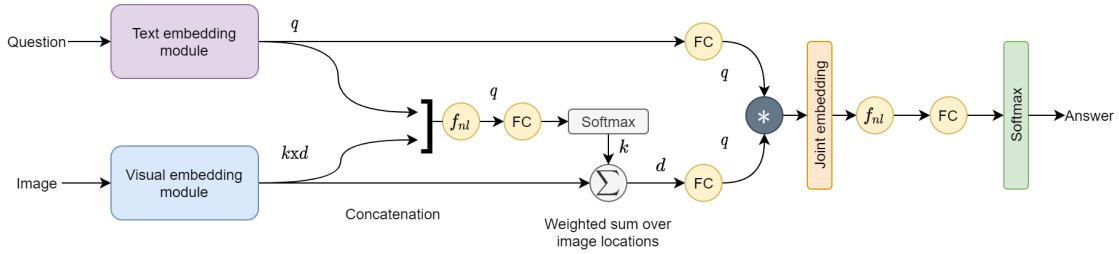


Figure 2.20: The full outline of the VQA bottom-up and top-down attention model. For bottom-up attention, the visual embedding extractor utilizes an R-CNN architecture. k is the amount of visual features, d their dimension, q is the question embedding dimension and fc is a fully-connected layer. Furthermore f_{nl} is defined in equation 2.15 and FC is a fully-connected layer. Figure adapted from [49].

In top-down attention, the importance of the features is weighted through top-level information. In other words, the information guides the attention. It functions as a soft-attention mechanism as the features are not removed but weighted based on their importance. For [4] the visual top-down attention is guided by the question. Their top-down attention mechanism is visualized in Figure 2.20. The image features are weighted by attention weights determined through the context of the question. To calculate the attention weights, each image feature k_i is concatenated with the question embedding q to the vector kq_i. Then, each kq_i vector is passed through a non-linear f_{nl}, fully connected and softmax layer sequentially resulting in the attention weights a_i for k_i. As the softmax layer outputs a set of probabilities that sum to 1, each element predicts how much the corresponding visual feature element should contribute. The non-linear layer f_{nl} is an implementation of the gated tanh concept from [11]. It passes the input vector through two separate fully connected layers, applying tanh activation on one and sigmoid on the other. The

sigmoid activation then acts as a gate on the tanh activations through element-wise multiplication (see equation 2.15). Each attention weight a_i is multiplied with the corresponding feature k_i. As the final attentional step, the visual features are summed together to create the attended visual feature vector.

$$f_{nl}(x) = \tanh(Wx + b) \odot \sigma(W'x + b') \tag{2.15}$$

The rest of their model functions as a joint-embedding model concatenating the attended visual feature vector with the question, passing these through two fully-connected layers and a softmax layer to predict the answer distribution (see Figure 2.20). The bottom-up attention part of the model can also be substituted for a regular CNN, thus utilizing only top-down attention. Both setups are explored in this thesis.

2.6 Image Generation

Image generation (IG) is the task of generating a realistic and semantically matching image from a text description. The state-of-the-art approaches to image generation over the recent years have all utilized cGANs (see section 2.3.1), referred to as GAN in this section, in some way. A single GAN can learn to synthesize images, but often these are unrealistic and blurry, lacking the details to semantically match the corresponding text description. To improve upon this multiple GANS were stacked together [57, 58]. By stacking multiple GANs in a sequence, the image is constructed iteratively. First, a small-sized image with coarse-level details is constructed, which the further GANs enlarge and enhance, filling in the further fine-level details. Through this method, the quality of the synthesized images improved a lot, but still are far from perfect. To further improve this an attention mechanism was added [54]. It uses top-down attention to weigh the intermediate visual features based on the individual word features of the text description. While this helps introduce more fine-grained details to the synthesized image, it fails to consider the higher-level meaning of the sentence and as a result, fails to capture this in the synthesized image.

These approaches can generate realistic images for narrow data sets such as CUB birds [50] but fail when moving to more broad data sets such as MSCOCO [29]. The stacked architecture also adds multiple levels of complexity and instability to the training due to the nature of GANs. A model that aims to solve some of these problems by moving away from the stacked GAN approach towards a single GAN is the Deep-Fusion Generative Adversarial Network. This model is explored for the image generation part of this thesis. Its workings are described in detail in the following section.

2.6.1 Deep-Fusion Generative Adversarial Networks

The Deep-Fusion Generative Adversarial Network (DF-GAN) introduces a multitude of innovations towards the GAN image generation approach. These allow it to produce high-quality realistic images while only employing a single GAN. The first innovation lies in the design of the architecture. In the stacked GAN architecture, the first generator receives a latent space vector concatenated with a conditioning variable extracted from the textual features. In DF-GAN this conditioning variable is omitted in favor of fusing the textual features at each upsampling step of the generator. This design ensures the generator takes the condition into account at each step, making a single-stage network possible.

The main part of the generator is a set of sequential upsampling blocks (UpBlock) that generate the image from the latent space. In the UpBlock, the input is first bi-linearly interpolated, doubling the size. Then within the UpBlock, the upsampled input is passed through two deep fusion blocks (DFBlock). In the DFBlocks, the textual features are fused with the visual ones. The output of the DFBlock is connected with the up-sampled input through a residual connection to produce the output of the UpBlock. In the DFBlock, an affine transformation followed by ReLU activation is performed twice, with a final convolutional step. In the affine transformation, the textual features are mapped by two fully connected layers to a set of shifting parameters γ and scaling parameters

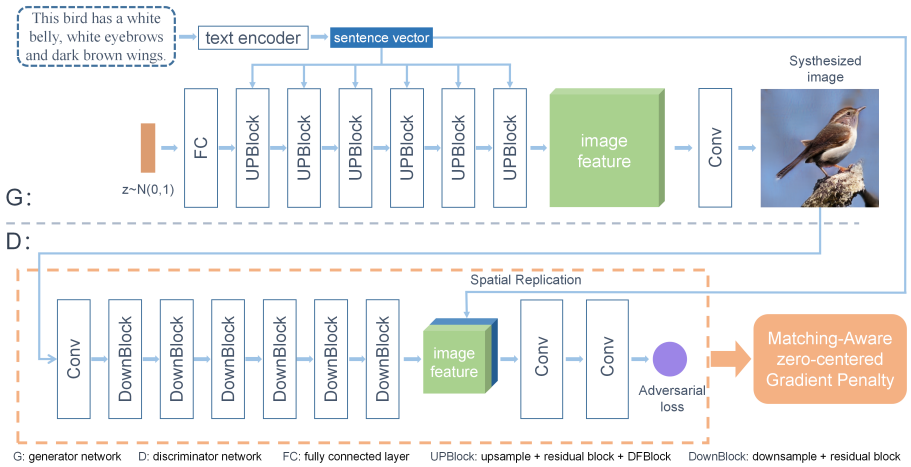


Figure 2.21: The full architecture of the DF-GAN. On the top half is the generator, which generates the image from the latent space by fusing the textual features of the text descriptions at the UpBlock steps. The discriminator receives this image or an image from the real data as input. It extracts visual features from this, which it combines with the textual features to label the input as fake or real. The adversarial loss containing the MA-GP is used to optimize the models. Figure taken from [48].

β. Each visual channel c_i is transformed through the parameter γ_i and β_i per equation 2.16. All the mentioned blocks are visualized in Figure 2.22.

$$Affine(c_i) = \gamma_i * c_i + \beta_i \tag{2.16}$$

The latent space vector is passed through a fully connected layer and reshaped to 4x4xNg*8, to be used as input for the first UpBlock. Here Ng is a hyperparameter that determines the channel amount of the final UpBlock output. The channel amount decreases from Ng*8 to Ng throughout the generator. The final UpBlock output is passed through a final set of layers to transform the visual features into the complete image. These layers are in sequence, LeakyReLU activation, a convolutional layer mapping the Ng channels to 3 RGB channels and a final TanH activation. The main innovations in the generator are the affine transformations and the incorporation of residual connections.

The discriminator receives an image, generated or from the data set, as input. First, the input channels are mapped by a convolutional layer to Nd channels. Here Nd is a hyperparameter that determines the base channel amount. This channel amount increases throughout the discriminator. Then the data is passed through a set of DownBlocks to extract the image features. Like the UpBlock, the DownBlocks utilize a residual connection. In the DownBlock the input is first downsampled through average pooling. Then the input is passed through a combination of convolution and LeakyReLU two times. The output of the final LeakyReLU is multiplied with the initial downsampled input to complete the residual connection and the block. This block is also visualized in Figure 2.22. The final DownBlock outputs the extracted visual features, which are combined with the textual features through spatial replication. Spatial replication involves the replication and reshaping of the text embedding to the same form as the visual features, such that they can be concatenated. The combined representation is passed in sequence through a convolutional layer that outputs Nd*2 channels, LeakyReLU activation and a convolutional layer that outputs a single channel. The result is a single value, which is used for the prediction.

A second important innovation used in the DF-GAN is in its loss function. It follows the general GAN loss structure, with some changes and additions. The discriminator loss defined as L_D in equation 2.18 consists of four components. Three of these utilize a hinge loss defined as $\min(0, cost)$, because of its ability to help stabilize the training process [28]. These three formulate

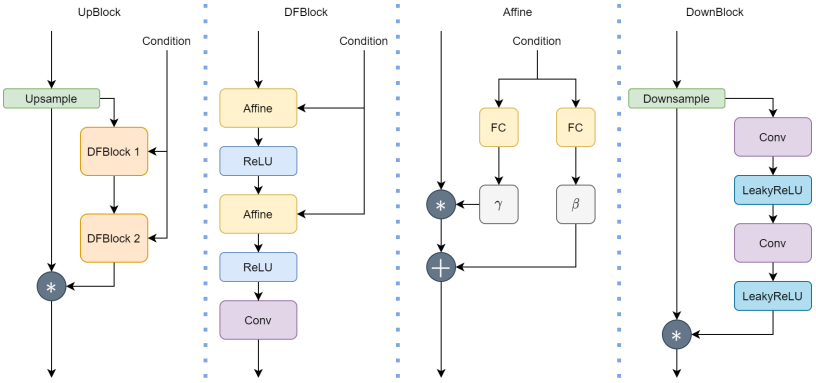


Figure 2.22: The different blocks of layers used in the DF-GAN. FC indicates a fully connected layer. γ and β are the shifting parameters for the affine transformation.

the loss for the generated images $\mathbb{P}_g$, the real images $\mathbb{P}_r$ and mismatched images $\mathbb{P}_{mis}$. The mismatched images are an extra part of the training procedure, where the discriminator also receives images that have their corresponding text description changed to be something mismatched. The combined loss of these three components limits the discriminator loss from -1 to 1.

The fourth component to the loss function is the matching-aware gradient penalty (MA-GP). It is based on the zero-centered gradient penalty (0-GP) proposed in [31], which is a regularization method introduced to help the generator converge. It applies a penalty based on gradients of the real image data points in the discriminator (see L_{0-GP} in equation 2.17). Doing this smooths the vicinity of the discriminator loss surface around the real data points which helps the generator generate more realistic images. The 0-GP loss was defined for an unconditional GAN. To extend it to the conditional one used in DF-GAN they added a gradient penalty on the text description, as defined by L_{MA-GP} in equation 2.17. In the equation, the k and p are hyperparameters set to determine the contribution of the penalty. The penalty is added to the other components of the discriminator loss to complete the full loss.

$$\begin{aligned} L_{0-GP} =& k\mathbb{E}_{x\sim\mathbb{P}_r}\left[(\|\nabla_x D(x)\|)^p\right] \\ L_{MA-GP} =& k\mathbb{E}_{x\sim\mathbb{P}_r}\left[(\|\nabla_x D(x,e)\| + \|\nabla_e D(x,e)\|)^p\right] \end{aligned} \tag{2.17}$$

$$\begin{aligned} L_D =& -\mathbb{E}_{x\sim\mathbb{P}_r}[\min(0,-1+D(x,e))] \\ & -(1/2)\mathbb{E}_{G(z)\sim\mathbb{P}_g}[\min(0,-1-D(G(z),e))] \\ & -(1/2)\mathbb{E}_{x\sim\mathbb{P}_{mis}}[\min(0,-1-D(x,e))] \\ & +k\mathbb{E}_{x\sim\mathbb{P}_r}\left[(\|\nabla_x D(x,e)\| + \|\nabla_e D(x,e)\|)^p\right] \\ L_G =& -\mathbb{E}_{G(z)\sim\mathbb{P}_g} D(G(z),e) \end{aligned} \tag{2.18}$$

Through these innovations, the DF-GAN was able to achieve state-of-the-art performance on the CUB and MSCOCO image generation tasks, while taking less training time. The full model has a single instance of the described generator and discriminator architectures. These are trained through the aforementioned loss function with the gradient penalty. The whole structure is visualized in 2.21. Due to the improved performance, relative simplicity, and speed this model is utilized for the image generation task.

This covers the relevant theoretical background for the tasks of visual question answering and image generation. Both tasks utilize the recent advancements in deep learning as described in the theory. For both tasks, multiple different setups are explored. Also, they are utilized in a cyclic architecture. These are described in the following chapter covering the methodology in addition to the design and details of the ShapeVQA data set.

Chapter 3

Methods

In this chapter, the methodology for the experiments is outlined. It builds upon the theoretical background presented in the previous chapter. The chapter is outlined in two main sections. To perform the experiments a data set was purpose-built to explore the generalizability and robustness of the models. The detailed information on its construction and contents is presented in the first section. The second main section covers the methodology for the tasks. The tasks are first described at an abstract general level with a modification to the IG task. Next, the approach to text embeddings for the textual part of the tasks is described. This is incorporated in the methodology towards the visual question answering (VQA) and image generation (IG) tasks presented in the next two sections. For both tasks, the evaluation metrics are also presented in corresponding sub-sections. The final section applies the methodology of the tasks to build two cyclic architectures in which one model is frozen, while the other is further trained.

3.1 Data set

In this section, the design and details of the ShapeVQA data set are presented. The ShapeVQA data set contains images with abstract stylized shapes instead of real-world photos like classical VQA data sets. The problem space of such a real-world data set is immense, making it hard to analyze the generalizability and robustness of the models like we want to do in this study. By constraining the problem space we want to see if the models can handle such a reduced problem space better and ultimately understand their abilities and limitations. Making the data set from scratch also gives a lot of flexibility in its design, enabling us to design it specifically for exploring generalizability and robustness. A final benefit is increased prototyping speed which goes in hand with the reduced problem space, as less data is required to cover it. For these reasons, ShapeVQA is chosen as the data set of choice for this thesis.

As mentioned, the ShapeVQA data set contains images with abstract stylized shapes. All the images in the data set contain one or two simple shapes placed upon an RGB 128x128 image with a grey background as seen in Figure 3.1. Each shape is constructed and defined by the four attributes, shape, size, color and location. The shape can either be a circle, a triangle, or a rectangle. The rectangles vary in which direction their longest side is. The triangles also vary in their side lengths. A shape can be small, medium or large as defined by the size attribute. Each size attribute defines the mean of the normal distribution $\mathcal{N}(\mu, 4)$ as a fraction of the image size. The size value, which is used to construct the shape, is sampled from this distribution. This adds variation to the sizes of the shapes. The color attribute defines the set of 10 colors: {Black, Gray, Brown, Red, Orange, Yellow, Green, Blue, Indigo, Violet}. The final attribute is the location, which defines the spatial positioning of the shape in the image. There are 9 locations as defined in Figure 3.2. The bottom and top locations span the full x-axis and half of their respective parts of the y-axis. Oppositely, the left and right locations span the full y-axis and half of their respective parts of the x-axis. The center spans a quarter-sized area in the center. The location of a shape is defined by the top-left most point of its bounding box. The images contain one or two shapes at a 50% chance per image. When two shapes are defined for an image, they are only allowed to share the same location attribute. This was done to avoid ambiguity in the construction of the question-answer pairs, which are described in the next paragraph.

Besides ensuring variation, the attributes are used to generate question-answer (QA) pairs for the image. The QA pairs cover the question answering aspect of the data set. The attributes

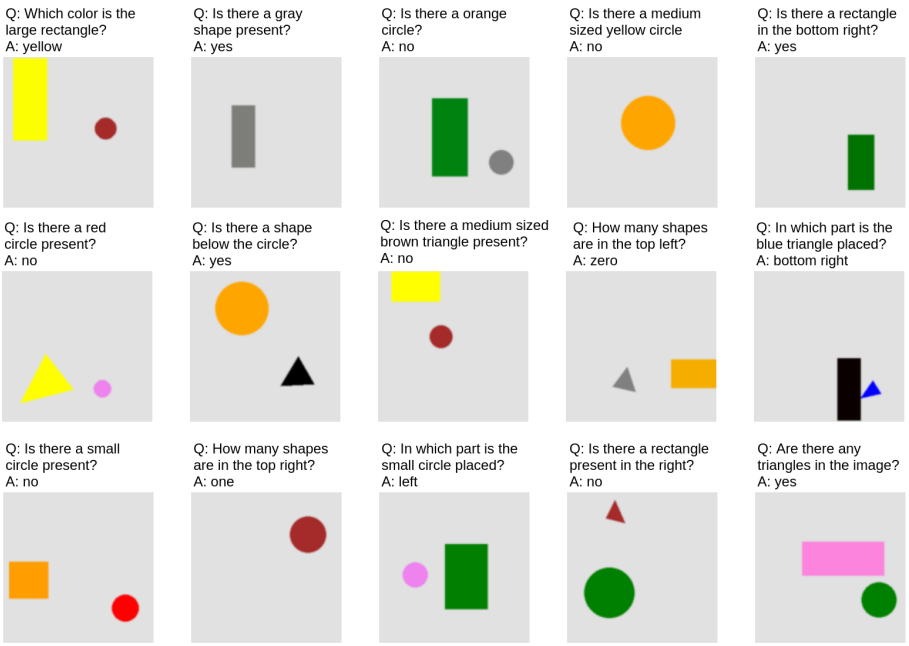


Figure 3.1: Example images from the ShapeVQA data set. On top of each image is a question-answer pair selected from the ten defined for each image. Note that each image contains either one or two shapes.

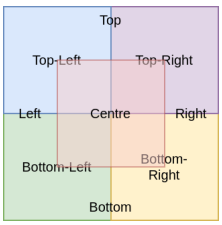


Figure 3.2: The possible locations of the shapes in the image based on the value of their location attribute. The Bottom, Left, Right, and Top values cover both squares they cross. The others only cover the square they are located in.

define the set of possible QA pairs for each image. Each question has a type defined by the most important attribute required to answer it. For questions that involve counting the shapes, the count type is attached instead. An example is the QA-pair, "Q: How many shapes are in the top? A: Two". Slight variations exist in the wording of the questions, but in general, this is limited. For example, "Q: Where is the shape placed?", "Q: In which part is the shape placed?" and "Q: Where can you find the shape?" are variations of the same question. There are approximately 5100 unique questions. The difficulty of the questions varies from relatively simple questions like "Q: Is there a circle present" to more complicated questions like "Q: Is there a circle to the left of the rectangle?". To capture the complexity of the questions, they are each assigned a specificity level. The specificity level measures how much attribute information the model has to understand to answer the question. There are four specificity levels defined for the questions. These are outlined in Table 3.1. The lower the specificity level value, the lower the specificity of the question. In general, there are two kinds of answers to the questions. Open questions have an open-ended answer, e.g. "Q: What is the shape? A: Circle". In reality, the open-ended answers are limited to 29 possibilities based on the available attributes. The other kind is Yes/No questions, which have

a simple yes or no answer, e.g. "Q: Is the shape a circle? A: Yes". Thus, in total there are 31 possible answers. Examples of the generated images with a corresponding QA-pair can be seen in Figure 3.1.

Table 3.1: Example questions for the four specificity levels. The lower the specificity level value, the lower the specificity of the question.

Level 0	Level 1
Which color can the shape be considered?	Are there any shapes located in the top left?
How large is the shape?	Is there a black shape present?
Are there any shapes overlapping?	Which color is the medium sized rectangle?
Where is the shape located?	Does the image contain a shape in the bottom left?
Level 2	**Level 3**
How large is the green circle?	Is there a small green rectangle?
Which color is the large circle?	Is there a black rectangle above the red triangle
How large is the grey circle?	Is there a yellow rectangle present in the bottom?
Which color is the large rectangle?	Is there a shape to the left of the violet rectangle?

In the data set, each image is assigned ten question-answer (QA) pairs. These are selected through the following procedure. For each image, all the possible QA pairs are generated based on the attributes. Directly sampling from this would result in an unbalanced distribution of question types. Also, for yes/no questions, the "no" answer is much more frequent than yes. Therefore a more sophisticated sampling method is required. To ensure balance, the possible QA-pairs are split into shape, count and location QA-pairs. The shape set concerns all questions about non-spatial attributes of the shape. The location set covers all questions about the locations of individual shapes and spatial relations between them. The count set covers all the questions that require the counting of the shapes. These sets are each split into yes/no and open questions. As there are always more "no" answers, the yes/no split is re-sampled by randomly removing QA pairs with "no" as the answer until the amount of yes/no answers is equal. Then, to ensure an even spread between open and yes/no questions, these are randomly sampled based on a 50% probability for either type. In total, five QA-pairs are sampled from the shape set, three from the location set, and two from the count set, resulting in ten total QA-pairs.

To make the models require generalizability, the attribute combinations are used to split the data set in a way that ensures unseen data in the test set. First, all the possible combinations of attributes are determined. In total, there are 840 possible combinations. These combinations are split into a train, test and validation group based on predefined split percentages. For each split, each attribute combination is used to generate the first shape of an image. To generate an additional shape in the image an extra attribute combination is sampled from the split. As the number of attribute combinations in the splits is generally not enough to sample the desired number of images, once each attribute combination has been exhausted, further ones required are randomly sampled from the splits. By sampling the images this way, the unique combinations of attributes in the test set do not exist in the train set. This requires the models to generalize from the attribute combinations seen in the train set. Consider an image in the training set that contains a yellow large circle in the center and an image in the test set that contains a green large circle in the center. It should be able to apply its understanding of this training attribute combination to the test attribute combination, as they only differ in the color attribute. It would not be able to effectively use a lookup table to do this as the exact attribute combination would not exist in such a table due to not being in the training set. Therefore, this aspect of the data set should allow us to say something about the generalizability of the models by looking at the performance on the test set.

These paragraphs cover the construction methodology of the ShapeVQA data set. The final constructed data set contains a total of 15000 images and 90000 questions. The split was set at 60% train, 20% test, and 20% validation, resulting in the data amounts shown in Table 3.2. Also, in Table 3.3 the number of questions per type are shown for the training set. With these data set details, we have covered all the important aspects of the ShapeVQA data set.

Table 3.2: Total data counts for the different data splits.

Data split	Unique attribute combinations	Images	Question-Answer Pairs
Train	486	9000	90000
Validation	162	3000	30000
Test	162	3000	30000

Table 3.3: Number of questions per type for the training set.

Category	N questions
Total	90000
Yes/No	44993
Open	45007
Size	12692
Count	18000
Location	27000
Color	23768
Shape	8540
Specificity 0	5805
Specificity 1	38685
Specificity 2	31721
Specificity 3	13789

3.2 Tasks

The two tasks explored in this thesis are visual question answering (VQA) and image generation (IG). The approach to the VQA task is the same as in the literature. It is a supervised learning problem predicting the probability distribution of possible answers based on an image and a question. The approach to the IG task slightly differs from the literature. Instead of a detailed text description, a question-answer pair is used to synthesize the image. Both tasks utilize the same data points but use them in different ways. Each data point is the triad $(Image, Question, Answer)$. In the VQA task, the model is trained to learn the mapping $(Image, Question) \rightarrow Answer$. The IG task is a different permutation of this, where the mapping $(Question, Answer) \rightarrow Image$ is learned. This similarity allows models for tasks to be used cyclically, as explored in the later section 3.2.4. First, the methodology towards the text embeddings used to process the question and answer parts of the task is explored followed by the methods for the VQA and IG tasks.

3.2.1 Text Embeddings

The tasks require the textual input to be in the form of a text embedding. The strengths and weaknesses of the embedding methods can have a large effect on the abilities of the models as they define the important textual features the model uses as a representation of the text. There are three general methods applied to embed the text. The first approach is the Pyramidal Histogram of Characters (PHOC) which generates embeddings by looking at the character level. For this approach splits at the L2, L3, and L4 levels were used like in [2]. See section 2.4.2 for the detailed workings of this method. The second approach is Bag of words (BOW) which looks at the word level to generate embeddings. The vocabulary is constructed from the train and validation texts. This covers all the vocabulary that was used to construct the question-answer pairs. Any words outside of this vocabulary are labeled as OOV. The rest of the BOW procedure is as outlined in section 2.4.1. The third and last embedding method is Sentence-BERT (SBERT) which looks at the sentence level. It uses the implementation from [36] with DistilBERT [42] as the base BERT model. The details for this method are described in section 2.4.3. These methods were chosen because each method looks at a different language level. They are used to embed both the questions and the answers separately. For the image generation task, these embeddings need to be combined into a single representation of the question-answer pair. This was done by concatenating the vectors of the question and answer embeddings.

The dimensions of the embeddings vary between the methods. SBERT has the largest di-

mensions of 768, with PHOC relatively large as well at 234, while BOW is quite small at 31. The PHOC embeddings also have a sparse embedding vector, as some of the characters in the alphabet never occur in the text. SBERT and PHOC also have their dimensions reduced through principal component analysis (PCA). This was done to compare better with BOW and see if a lower-dimensional space is sufficient. Also, for PHOC, the embeddings are sparse because some of the characters in the alphabet never occur in the text, so a more dense representation might be beneficial. The dimensions are reduced to keep 98% of the variation that exists in the data. Both the reduced and full versions of these embedding methods are explored. Therefore, in total there are five embedding methods: SBERT-Full, SBERT-Reduced, PHOC-Full, PHOC-Reduced, and BOW. The dimensions of all the different embeddings for the single question and answer and their combined counterparts are summarized in Table 3.4.

Table 3.4: Dimensions for the kinds of text embeddings used in this thesis, where Q is a question and A an answer.

	SBERT Full	SBERT Reduced	PHOC Full	PHOC Reduced	BOW
Q | A	768	70	234	31	64
Q + A	1536	140	468	62	128

3.2.2 Visual question answering

There are two main architectures explored for the VQA task. A joint embedding architecture as described in Figure 2.18 and the attentional extension attention model as described in Figure 2.20. These modules both utilize separate text and visual embedding modules. Multiple setups of these modules are explored for both models. The text embedding module embeds the question by utilizing one of the five text embedding methods. The LSTM traditionally used for VQA was omitted in favor of SBERT as it allows the use of transfer learning like used in CNNs, in natural language processing. The visual embedding module depends on the type of VQA model used and is presented later in this section. The models share some general architectural design choices. Both models utilize a single 50% dropout layer before the final classification layer. In the joint-embedding models, this is before the final FC + softmax layer. For the attention models it is before the final f_{nl} + FC + softmax layer. The models also share the hyperparameter N_{hidden}, which sets the number of neurons in the FC layers. Only the final FC layer has a differing amount of neurons, which is 31 for the number of possible answers. Each FC layer is followed by ReLU activation, besides the final FC layer and the f_{nl} layers from the attention model. Cross entropy loss between the predicted and target answer is used by both models. This is optimized through Adam using a learning rate of 0.002 with $\beta_1 = 0$, $\beta_2 = 0.999$.

For the regular joint-embedding model, there are two visual embedding modules explored. One extracts a feature vector by L2-normalizing the penultimate layer of VGG16 [44] to achieve a 1x4096 feature vector of the image. VGG16 was chosen because it was used in [5] for the same purpose. The VGG16 network is frozen throughout the training, so it will not learn data set-specific visual features. To explore the general features of a pre-trained architecture compared to the more data set-specific features of a CNN trained from scratch, a CNN is also trained to perform the visual embedding part. The CNN consists of four blocks that contain a convolutional, batch normalization, ReLU and max-pooling layer sequentially. After the four blocks, a fully connected layer followed by ReLU activation is used to calculate the visual embedding. The convolutional layers have a kernel size of 5x5, stride of 1 and utilize zero padding. The amount of filters goes from low to high, with the first layer having 6, second 16, third 32 and fourth 64. Both visual embedding modules are combined with the five text embedding ones for a total of 10 models. In addition, *language-only* and *vision-only* models are trained to explore the possible biases in the model. The *language-only* models follow the same structure as the joint-embedding model omitting the element-wise multiplication with the visual embedding. Conversely, the *vision-only* model omits the element-wise multiplication with the text embedding. There are five *language-only* models

trained, one for each embedding type. Two *vision-only* models are trained, one utilizing the same CNN architecture and the other the pre-trained VGG16 features.

The attention model employs either both bottom-up and top-down attention or only top-down attention. For bottom-up attention, the visual embedding module utilizes a pre-trained FRCNN model. The model was pre-trained on the MSCOCO data set [29] and utilizes the same weights as used in [4]. The FRCNN model struggled to find many features for this data with their original parameters, so these are changed. The score and NMS thresholds are set at the low values of 0.0001 and 0.001. This is far from ideal, as it means there is barely any filtering. However, it is necessary to find any features for the images in the data set. As not many were found, the amount of features is set at 6, resulting in an embedding dimension of 6x2048. In the top-down attention-only model the FRCNN is replaced with pre-trained Resnet18 [18] embeddings as done in [4]. These are extracted from the final convolutional layer of the Resnet architecture, resulting in 49 features of 512 dimensions. Like VGG16, these are L2-normalized to calculate the visual embedding of 49x512 dimensions. Both methods are trained with the five text embedding types resulting in 10 models.

Evaluation Metrics

In total, there are 27 VQA models trained and analyzed. The metric used to analyze the performance of the models is accuracy. The accuracy is calculated by determining the percentage of predictions that match the target. This definition for the accuracy metric holds for all the accuracy metrics utilized in this thesis. During the training of each model, its accuracy on the training and validation sets is computed for each epoch. To compare the final performance of the models their accuracy is computed on the test set. On the test set, the accuracy for the different types of questions and specificity levels are also computed.

3.2.3 Image generation

The models trained for the image generation (IG) task follow an adapted version of the DF-GAN architecture described in section 2.6.1. The original DF-GAN model was designed to generate 256x256 images. For this image size, the model takes a considerably large amount of training time to train. Also, such a high-resolution image size is unnecessary for the relatively simplistic images of the ShapeVQA data set. Therefore the architecture is modified to generate 128x128 images instead. This image size strikes a good balance between detail and training speed for this problem. For the generator a sequence of six UpBlocks is utilized, upsampling from 4x4 to 128x128. The latent space z used to generate the images has a dimension of 100 and is sampled from the standard normal distribution. The discriminator utilizes five DownBlocks to downsample the input image. The rest of the general model structure remains the same as the original implementation described in section 2.6.1.

Two additional design choices were made to combat the discriminator overpowering the generator early on in training. The Nd and Ng parameters, which determine the number of filters in the discriminator and generator convolutional layers, are set at 8 and 16 respectively. This ensures that the discriminator can utilize fewer parameters than the discriminator, making it relatively weaker in training. In the early stages of training, the output of the generator is likely to be quite noisy as it is learning to generate the images. The discriminator can quickly learn this noise to distinguish the images from the real images. To combat this, noise sampled from the standard normal distribution is added to the image input of the discriminator. The added noise is sampled from the $\mathcal{N}(0, 1)$ distribution and decays over time in the training following the formula $2^{-0.5x}$ where x is the current epoch value. This was done to give the discriminator more strength later on in training. These two design choices were required to make the training stable enough to produce high-quality images.

Another approach to improve learning stability is also explored. This method involves employing a Variational autoencoder (VAE) as pretraining for the generator. As mentioned in section 2.3.2, a large part of the VAE and GAN architectures share a similar network structure. When

the encoder and decoder of the VAE model use the same underlying structure as the discriminator and generator of the GAN, the decoder can be used as an initialization point for the generator. The generator architecture of the DF-GAN is used as the decoder of the VAE. The discriminator architecture is slightly modified to work as the encoder of the VAE model. The final convolutional layer that outputs the decision value is changed to two fully connected layers that output μ and β vectors used for the reparameterization trick in the VAE. The dimension for these vectors is set at the same as the z dimension. In addition to the Kullback-Leibler loss, the VAE uses a reconstruction loss based on the MSE (see section 2.2.4) between the original and reconstructed image. A problem with this loss is that it can cause exploding gradients which can result in unstable training. Therefore, L2-norm gradient clipping with a max of 100 is used to avoid this. It uses the same optimizer as the DF-GAN model, with a learning rate of 0.0002. Once the VAE is finished training, the decoder is used as the starting point for the generator in a further trained DF-GAN model.

To train the model, the same loss function as for the original DF-GAN architecture defined in equation 2.18 is used. The k and p hyperparameters in the loss function were set at $k = 2$ and $p = 6$ in accordance with [48]. The generator and discriminator are optimized by the Adam [24] optimizer with $\beta_1 = 0.0$ and $\beta_2 = 0.999$. The learning rates are set according to the Two Timescale Update Rule [19], which helps the GAN converge. It sets the generator learning rate at 0.0001 and the discriminator learning rate at 0.0004. This covers the important aspects of the IG models.

Evaluation Metrics

In total, there are 10 image generation models explored, a model for each text embedding type without and with VAE pretraining. For each model, the fake and real accuracies of the discriminator, are calculated during the training. Additionally, the losses of the generator and discriminator are measured during the training. Computing the quality and semantic consistency of the images is more involved, as this is not easily quantitatively computed. Ideally, a group of humans would grade the images to determine their quality, but this is neither a viable nor sustainable method. Therefore in the research towards image generation, two metrics have commonly been used that try to quantify this. Namely, the Inception score (IS) [41] and the Fréchet inception distance (FID) [19]. The IS aims to measure two things simultaneously. The level of variety in the images and the level of distinctness between the images. It uses a pre-trained Inception model [47] to calculate a class-label distribution y on the generated image x. The idea behind the measure is that the entropy of the conditional distribution $p(y|x)$ should be low, as this means there is clear distinctness in the class-label predictions. In contrast, the entropy of the marginal distribution $p(y)$ should be high, as this indicates variety in the predicted labels. This means that the Kullback-Leibler (KL) divergence between $p(y|x)$ and $p(y)$ should be large, which is what IS measure is defined as (see equation 3.1). The Inception-V3 model pre-trained on the ImageNet task was used as the Inception model for the metric. Its final softmax layer output is used as the class-label distribution for an image.

$$IS = \exp(\mathbb{E}_x D_{KL}(p(y|x)||p(y))) \tag{3.1}$$

The Fréchet inception distance (FID) was proposed as an alternative to the Inception Score (IS) that not only evaluates the distribution of the images but also compares this with the distribution of the real images by calculating a Fréchet distance between them. Therefore, the FID metric also measures how the generated images relate to the real images. It works by estimating the multidimensional gaussian $\mathcal{N}(\mu, \Sigma)$ based on the mean and covariance of the image features extracted by the Inception network. It calculates this distribution for the real images and the generated images. The FID metric is then calculated by taking the Wasserstein distance between the distributions. The lower this value, the better. The FID metric is defined per equation 3.2, where g indicates the generated images and x the real images. The FID metric uses the same Inception model as the IS. However, it takes the output from the penultimate layer instead of the final one.

$$FID = |\mu_x - \mu_g|^2 + \text{tr}(\Sigma_x + \Sigma_g - 2(\Sigma_x\Sigma_g)^{1/2}) \tag{3.2}$$

These metrics can measure the visual quality of the images, but they fail to measure the semantic consistency of the image with the question-answer pair used to generate it. Because the image generation and VQA models are trained on the same data, the VQA model can be a critic of the generated images. The idea behind this is that there should be consistency between the answer used to synthesize the image and the answer of the VQA model towards the synthesized image. It can be measured by using the question and generated image as input for the VQA model. Then, the answer used to synthesize the image can be compared with the VQA answer to calculate its accuracy. This accuracy is used as an extra performance metric on the IG model. The performance on the validation set of the IS and FID metrics is calculated for each epoch. For the test set the IS, FID and VQA critic metrics are measured.

3.2.4 Cyclic

To explore the applications of cyclic architectures towards improving the generalizability and robustness of the models, two cyclic architectures utilizing them are explored. The models used in the cyclic architecture have first been trained on their corresponding tasks. In the architecture, one model is frozen, while the other is further trained. There are two cyclic architectures used, one that further trains the VQA model and one that further trains the IG model. These are visualized in Figure 3.3.

The cyclic architecture for visual question answering (VQA) works as follows: First, a forward pass is done through the VQA model with the original image I and question Q to predict the answer A'. Next, a reversal pass is performed by using the original question Q and the predicted answer A' as input for the IG network to generate a new image I'. To use this new answer the same text embedding method as the original is used to embed it. Last, another forward pass is done through the VQA model now using the generated I' and the original Q to predict A''. To further train the VQA model a new loss function was defined that utilizes the extra answer and image. It consists of three components that are utilized in different combinations to explore their effect. The first component is the regular VQA loss L_{vqa}, which measures the cross-entropy (CE) between the real answer and the predicted answer from the original data. The second is a consistency loss L_{cns}, which measures the CE between the real answer and the predicted answer from the generated image. The third is an image consistency loss L_{im_cns}, which measures the consistency between the original image and the generated one. The image consistency loss is measured by calculating the cosine similarity between the VGG16 features of the images. These different components are defined in equation 3.3. Four different setups of these components are compared. These are indicated in equation 3.4. The L_{full_coeff} loss utilizes the image consistency as a way to determine the importance of the VQA consistency loss. The higher the image consistency loss, the lower the VQA consistency loss is in this equation. The idea behind this is that for low image consistency the image likely can not be properly used to answer the question by the VQA model. Besides the loss function, the image consistency loss is also used in another way. Inspired by [43] it is used as a gating mechanism. When the image consistency loss is too high, the batch is skipped. Ideally, this would make the training only consider cases where the images are of high enough quality. The threshold for not skipping a batch was set at a maximum of 0.20. The L_{full} and L_{full_coeff} losses are both trained with and without this gating mechanism, therefore in total, there are six different setups explored.

$$\begin{aligned} L_{vqa} &= CE(A, A') \\ L_{cns} &= CE(A, A'') \\ L_{im\_cns} &= 1 - \frac{I \cdot I'}{\|I\| * \|I'\|} \end{aligned} \tag{3.3}$$

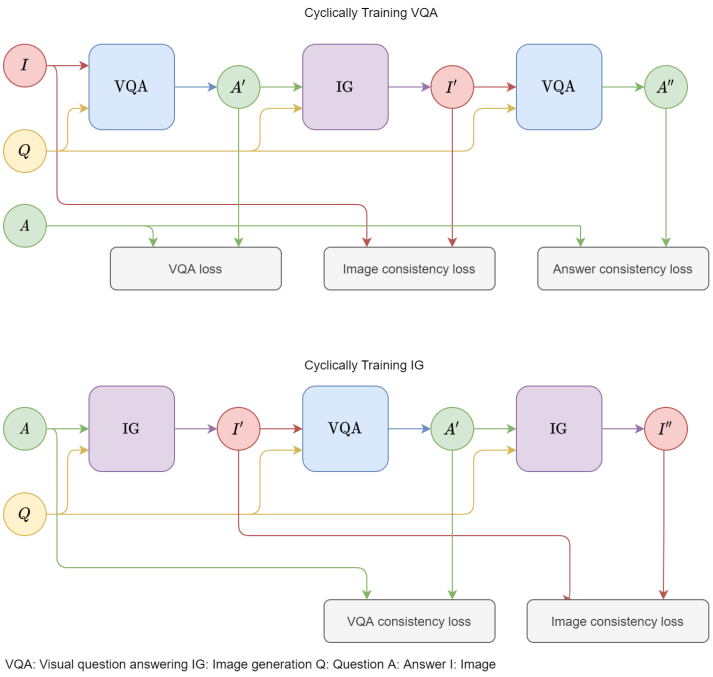


Figure 3.3: The data flow in the cyclic architectures is indicated by the arrows and colors. The cyclic architecture to improve the visual question answering task is on the top and for the image generation task on the bottom. The data for I, Q, and A is a data point from the data set.

$$\begin{aligned} L_{vqa\_only} &= L_{vqa} + L_{cns} \\ L_{cns\_only} &= L_{cns} \\ L_{full} &= L_{vqa} + 0.5L_{cns} + 0.5L_{im\_cns} \\ L_{full\_coeff} &= L_{vqa} + L_{cns}/L_{im\_cns} \end{aligned} \tag{3.4}$$

The image generation (IG) cyclic architecture works as follows: First, the question Q and answer A are used to generate the image I'. Next, the I' and Q are used as input for a forward pass through the VQA model to predict A'. Last, the original Q is used in combination with the new A' to generate another image I''. In this case, there are only two loss components explored. The same VQA loss as used in the VQA cyclic architecture and the same image consistency loss, but now between the generated images I' and I''. These are combined into two losses, a loss that only has the L_{vqa} component and one that combines the L_{vqa} with the L_{im_cns} as defined in equation 3.5. Effectively, these two loss setups define two separate architectures for the cyclic training of IG. The one with only the L_{vqa} component omits the second IG forward pass, while the other with both components use both forward passes.

$$\begin{aligned} L_{vqa\_only} &= L_{vqa} \\ L_{full} &= L_{vqa} + L_{im\_cns} \end{aligned} \tag{3.5}$$

In the visual question answering (VQA) and image generation (IG) cyclic architectures, L2-norm gradient clipping with a limit of 0.25 was used to limit the movements in the loss space of the models. They are optimized by the same optimizers used in their original training. This means both use Adam is with $\beta_1 = 0$, $\beta_2 = 0.999$. For IG the learning rate is the same as the generator so 0.0001 and for VQA it is 0.002. The same evaluation metrics are used as in the original models. This covers the methodology behind the cyclical part of the experiments. In the next chapter, the results from the experiments using the described methods are presented.

Chapter 4

Results

In this chapter, the results of the experiments are presented. First, the experimental procedure is described which builds upon the methodology to describe the experimental process. Next, the results of the experiments are presented. The results for the visual question answering (VQA) task followed by the results for the image generation (IG) task are presented first. After these sections, the results for the cyclic architectures are shown to finish off this chapter.

4.1 Experimental procedure

In this section, the procedure behind the experiments is described. To perform the experiments a multitude of models were trained and evaluated. A batch size of 24 was used throughout all experiments. Each experiment is performed three times to achieve more statistically significant results. For each numerical result, a mean and standard deviation was calculated between the three runs. As the first step of the procedure, all setups of the VQA and IG models are trained. The VQA models are trained for 30 epochs with early stopping based on the validation accuracy to avoid overfitting. The IG models are trained for 400 epochs. For each of the pre-trained IG models, a Variational autoencoder (VAE) is first trained. These are trained the same amount of epochs as the IG model, but with early stopping based on the loss to avoid the model collapsing. The performance of the models is measured through their corresponding evaluation metrics. For the IG models, a qualitative review of the generated images is done as well. For the cycle experiments, only the best-performing models are selected from VQA and IG. First, the best IG model is selected based on the highest VQA accuracy, as semantic consistency is important for cyclical training. If there is a tie, the model with the lowest FID measure is selected. Next, the best-performing VQA model with the same text embedding type as the selected IG model is selected. These are the models used in the cycle experiments. There are two cycle experiments done. In the first experiment, the VQA model is further trained by the selected IG model through cyclical training. Additionally, the IG model is also further trained by the selected VQA model. In the second experiment first, the original selected IG model is further trained by the cyclically trained VQA model. Next, the originally selected VQA model is further trained by the cyclically trained IG model from the second experiment. The second experiment explores what happens when the cyclical training is extended to multiple cycles. These cycle experiments are respectively called Cycle 1 and Cycle 2 in the results section. Each cycle experiment is performed for 100 epochs. This covers the experimental procedure behind the results presented in the following sections.

4.2 Visual question answering

For the visual question answering (VQA) task, there are four network types: Bottom + Top attention (*BTA*), Top attention (*TA*), *VQA-CNN* and *VQA-Pretrained*. *BTA* utilizes the VQA Bottom-up + Top-down attention model (see section 2.5.2). *TA* skips the Bottom-up attention part of this model in favor of Resnet18 [18] visual features. *VQA-CNN* utilizes a convolutional neural network (CNN) trained from scratch in a joint embedding approach (see section 2.5.1). Lastly, *VQA-Pretrained* replaces the CNN architecture of the previous model with pre-trained features from VGG16 [44]. Each network was trained with each of the five text embedding types. Also, a *vision-only* model is trained for *VQA-CNN* and *VQA-Pretrained* indicated by the omittance of the text embedding in the figures and tables. Furthermore, to explore language priors, the

network type *language only* is added for models that only utilize the text embeddings to predict the answer. First, the plots for the training process are shown followed by an evaluation on the test set.

4.2.1 Training

During the training of the VQA models, their training accuracy, validation accuracy, and training loss were measured in each epoch. Only the model with the highest general accuracy on the test set is shown here for the sake of brevity. The others are found in appendix A.1.1. The three runs were summarized by taking their mean and standard deviations. The training metrics for *TA* can be seen in 4.1. Note that due to early stopping the models have differing total epoch lengths. The graphs show that there are no large variations between the runs. The training accuracy develops fast to $\sim$ 98% for each embedding type besides SBERT-Full. SBERT-Full takes more training time to achieve lower training accuracy. The same is indicated by the loss, where SBERT-Full consistently has a higher loss than the other methods. The validation accuracy converges quicker than the training accuracy at around $\sim$ 83% for each embedding type besides SBERT-Full. SBERT-Full performs worse here as well converging around $\sim$ 80%. The training metrics for *TA* indicate that the SBERT-Full embedding performs worse than the others, which perform closely together. There also is an indication of overfitting as the training accuracy is much than the validation accuracy for all the models.

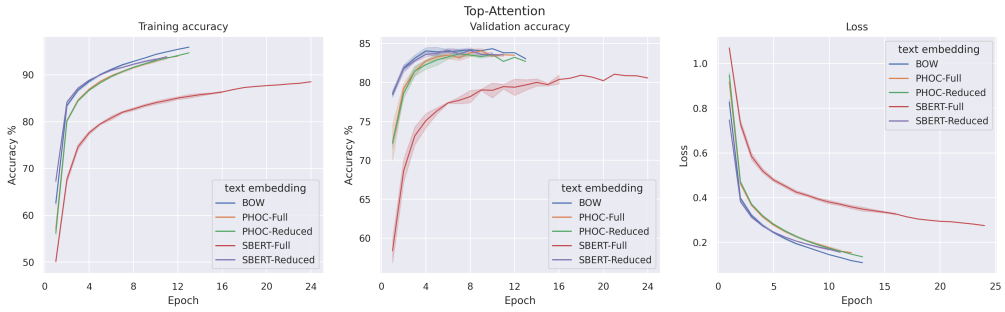


Figure 4.1: From left to right: the training accuracy, validation accuracy, and training loss of the *Top attention* VQA model for each text embedding type. The number of epochs differs between the different text embedding types due to early stopping in training.

4.2.2 Evaluation

After the training, the performance of each visual question answering (VQA) model was evaluated on the test set. The accuracies in percentages are measured on the complete set (All), the yes/no answerable question-answer pairs (Yes/No) and the open-ended answerable question-answer pairs (Open) as seen in Table 4.1. For each metric, the mean and standard deviation between the three runs is calculated and displayed. The number of parameters for each setup is also determined to compare the relative strengths of the models.

Top attention (*TA*) is the best performing architecture for all three categories while using the least amount of parameters (ignoring *language-only* models). The high performance of *TA* can partly be attributed to it minimizing its performance difference between yes/no and open questions. All models perform better on yes/no than open questions. In the *language-only* models, the open question performance is much higher than random chance, which would be $\sim$ 3.4%. Furthermore, *vision-only* models perform much worse on the open questions, being close to the random chance. The *language-only* models also perform better than random on the yes/no questions, while *vision only* once again performs at a random level. These results indicate that the questions have much more predictive ability than the images on the answer. Accordingly, this is an indication of language priors existing in the data set.

Table 4.1: Performance of the visual question answering models on the test set. The yes/no column indicates questions answerable by a yes or no answer. The open column indicates questions with an open answer. The bold-faced values indicate the highest mean value within the network type for the category. Underlined values indicate the highest mean value for the category.

Network	Text Embedding	Parameters	All	Yes/No	Open
Bottom + Top Attention	SBERT-Full	3.6M	43.39 ± 0.75	59.83 ± 0.28	**27 ± 1.24**
	SBERT-Reduced	2.8M	43.38 ± 0.29	61.22 ± 0.82	25.56 ± 0.32
	PHOC-Full	3.0M	**43.96 ± 0.47**	62.16 ± 1.04	25.81 ± 0.11
	PHOC-Reduced	2.8M	43.72 ± 0.29	62.28 ± 0.57	25.18 ± 0.54
	BOW	2.8M	43.9 ± 0.22	**62.32 ± 0.79**	25.53 ± 1.01
Top Attention	SBERT-Full	1.6M	75.55 ± 0.7	83.74 ± 0.94	67.29 ± 0.51
	SBERT-Reduced	868K	78.61 ± 2.28	**86 ± 1.92**	71.19 ± 2.64
	PHOC-Full	983K	**78.99 ± 0.88**	85.86 ± 0.65	72.06 ± 1.51
	PHOC-Reduced	826K	78 ± 0.73	85.13 ± 0.36	70.83 ± 1.12
	BOW	862K	78.97 ± 1.38	85.43 ± 1.12	**72.47 ± 1.89**
VQA-CNN	SBERT-Full	1.5M	64.8 ± 6.39	70.66 ± 11.12	58.91 ± 7.35
	SBERT-Reduced	1.3M	70.07 ± 0.48	79.13 ± 1.09	61.01 ± 0.57
	PHOC-Full	1.3M	**71.14 ± 2.29**	**80.32 ± 2.08**	**62.02 ± 2.84**
	PHOC-Reduced	1.3M	60.26 ± 9.46	72.38 ± 4.81	48.18 ± 14.27
	BOW	1.3M	69.3 ± 5.22	79.54 ± 2.52	59.14 ± 8.02
(Vision only)	-	1.1M	24.89 ± 0.16	49.85 ± 0.31	0.01 ± 0.01
VQA-Pretrained	SBERT-Full	1.3M	50.63 ± 6.31	62.7 ± 5.54	38.65 ± 7.14
	SBERT-Reduced	1.1M	64.57 ± 9.08	75.19 ± 7.9	54.02 ± 10.28
	PHOC-Full	1.2M	**66.31 ± 0.3**	**77.15 ± 0.96**	**55.53 ± 0.74**
	PHOC-Reduced	1.1M	64.95 ± 0.82	74.68 ± 1.45	55.32 ± 1.3
	BOW	1.1M	65.31 ± 1.69	76.54 ± 0.1	54.15 ± 3.5
(Vision only)	-	1.1M	24.96 ± 0.25	49.97 ± 0.52	0.02 ± 0.02
Language only	SBERT-Full	204K	42.28 ± 1.07	57.7 ± 2.1	26.89 ± 0.6
	SBERT-Reduced	25.9K	43.67 ± 0.73	61.53 ± 1.31	**25.84 ± 0.92**
	PHOC-Full	54.6K	**43.99 ± 1.15**	62.67 ± 1.89	25.34 ± 0.41
	PHOC-Reduced	15.4K	43.85 ± 0.35	**62.74 ± 0.78**	25 ± 0.6
	BOW	24.4K	43.55 ± 0.55	62.31 ± 1.6	24.84 ± 0.61

The *BTA* architecture performs much worse than the others, even than the related *TA* model. It performs similarly to the *language-only* models, which is an indication that the visual bottom-up attention part of the model is not working well on this data set. The visual parts of the *VQA-CNN* and *VQA-Pretrained* networks do seem to work well. Across the categories, *VQA-CNN* performs better than *VQA-Pretrained*, but not by a large margin.

For the best performing architecture of Top attention (*TA*), the different text embeddings perform closely together besides SBERT-Full, which is worse than the others. Similarly in the other architectures, the three main embedding types (considering either the full or reduced version), perform close together. The dimensionality reduction of SBERT seems to be effective as SBERT-Full outperforms SBERT-Reduced for each architecture, even for the *language only* models. Conversely, the dimensionality reduction of PHOC does not appear to be an improvement as it performs worse than PHOC-Full throughout. Especially on *VQA-CNN* PHOC-reduced seems to perform poorly. However, its standard deviation is quite high for that value, so that might explain why the difference between PHOC-Full and PHOC-Reduced is larger in *VQA-CNN* than the other architectures. The results do not indicate a clear best text embedding type.

In addition to the general accuracies, the accuracies per question type and specificity levels were also measured as seen in Table 4.2 and 4.3 respectively. Once again, *TA* generally works best throughout the question types, only for the location type *TA* performs slightly worse than *VQA-CNN*. However, all the models perform poorly on the location type in comparison to the other ones. *TA* outperforms the other models the most in the size, shape and color types. Looking at the non-attentional joint embedding models, *VQA-CNN* and *Pretrained* only differ slightly. However, for color, *VQA-CNN* is much better. Both *BTA* and *language only* perform reasonably high on the shape and count types while *vision only* performs much worse. This might be an indication that these types of questions are more easily correctly answered through language priors.

Looking at the different specificity levels, *TA* also performs the best besides on level 0. For *TA* we see a pattern that matches the intuition of a higher specificity level being more difficult as

Table 4.2: Performance of the visual question answering models per question type on the test set. The bold-faced values indicate the highest mean value within the network type for the category. Underlined values indicate the highest mean value for the category.

Network	Text Embedding	Size	Shape	Color	Location	Count
Bottom + Top Attention	SBERT Full	**39.25 ± 1.95**	69.84 ± 0.54	35.39 ± 0.65	**28.56 ± 1.22**	66.47 ± 1.25
	SBERT Reduced	38.83 ± 1.44	70.52 ± 0.59	35.18 ± 0.7	26.3 ± 0.6	**70.05 ± 2.35**
	PHOC-Full	39.9 ± 1.42	70.94 ± 0.18	35.05 ± 0.65	27.85 ± 0.61	69.87 ± 0.36
	PHOC-Reduced	37.9 ± 1.11	70.71 ± 0.46	**36.25 ± 0.58**	27.14 ± 0.63	69.62 ± 0.51
	BOW	38.77 ± 3.77	**71.1 ± 0.49**	35.12 ± 1.23	28.06 ± 1.19	69.88 ± 0.65
Top Attention	SBERT Full	74.29 ± 3.49	89.1 ± 0.76	89.65 ± 1.1	54.43 ± 1.63	82.89 ± 0.89
	SBERT-Reduced	79.29 ± 1.91	90.54 ± 1.65	**91.66 ± 1.89**	57.73 ± 3.18	**86.44 ± 2.17**
	PHOC-Full	**81.45 ± 3.02**	**90.84 ± 0.42**	91.17 ± 1.08	**58.82 ± 0.42**	85.69 ± 0.48
	PHOC-Reduced	79.33 ± 4.1	89.69 ± 0.53	90.42 ± 0.77	58.3 ± 0.52	84.52 ± 1.14
	BOW	83.06 ± 4.54	90.49 ± 1.11	90.96 ± 0.68	57.96 ± 2.81	86.19 ± 0.34
VQA-CNN	SBERT-Full	62.89 ± 9.93	71.47 ± 11.57	70.23 ± 7.91	53.96 ± 6.13	72 ± 7.31
	SBERT-Reduced	69.71 ± 2.66	81.84 ± 1.96	71.64 ± 2.34	58.66 ± 0.91	**79.74 ± 0.5**
	PHOC-Full	**71.93 ± 2.78**	82.78 ± 2.94	**74.32 ± 6.84**	59 ± 0.88	79.02 ± 2.01
	PHOC-Reduced	62.61 ± 27.9	74.49 ± 6.77	46.44 ± 7.65	57.27 ± 5.94	74.64 ± 6.12
	BOW	69.51 ± 15	**83.07 ± 3.71**	68.88 ± 7.99	**59.18 ± 0.88**	78.29 ± 3.18
(Vision only)	-	5.93 ± 1.23	51.61 ± 7	25.35 ± 1.47	24.85 ± 0.13	24.72 ± 0.13
VQA-Pretrained	SBERT-Full	59.5 ± 3.77	71.29 ± 6.48	44.71 ± 6.05	34.09 ± 10.58	67.29 ± 2.91
	SBERT-Reduced	66.64 ± 5.66	79.84 ± 4.85	59.46 ± 8.7	54.26 ± 14.1	78.09 ± 7.33
	PHOC-Full	69.78 ± 3.42	**80.44 ± 1.29**	62.2 ± 0.31	**55.71 ± 0.68**	78.49 ± 0.94
	PHOC-Reduced	**69.86 ± 3.46**	79.07 ± 0.56	59.99 ± 1.21	52.78 ± 0.7	**79.64 ± 0.1**
	BOW	63.72 ± 7.14	80.15 ± 0.85	**63 ± 0.95**	54.3 ± 1.8	78.92 ± 0.87
(Vision only)	-	6.25 ± 1.5	49.48 ± 6.49	26.21 ± 1.45	24.3 ± 0.27	25.52 ± 0.84
Language only	SBERT-Full	37.96 ± 0.79	68.19 ± 2.9	34.7 ± 1.39	**28.73 ± 1.58**	63.26 ± 0.98
	SBERT-Reduced	**38.94 ± 0.56**	69.7 ± 1.18	36.42 ± 0.63	26.96 ± 2.02	69.19 ± 3.31
	PHOC-Full	37.56 ± 2.35	70.86 ± 1.13	**36.09 ± 0.91**	28.64 ± 0.99	69.11 ± 2
	PHOC-Reduced	37.2 ± 1.68	**71.45 ± 0.1**	35.89 ± 0.95	27.92 ± 1.03	69.69 ± 0.63
	BOW	36.28 ± 0.3	70.73 ± 0.72	35.48 ± 0.71	27.54 ± 1.03	**70.32 ± 0.47**

Table 4.3: Performance of the visual question answering models per specificity level on the test set. The bold-faced values indicate the highest mean value within the network type for the category. Underlined values indicate the highest mean value for the category.

Network	Text Embedding	Level 0	Level 1	Level 2	Level 3
Bottom + Top Attention	SBERT-Full	18.05 ± 1.93	54.53 ± 1	**32.95 ± 1.29**	46.64 ± 0.57
	SBERT-Reduced	18.46 ± 1.23	56.43 ± 1.04	31.16 ± 0.54	45.24 ± 0.64
	PHOC-Full	**19.53 ± 0.42**	56.47 ± 0.17	31.83 ± 1.18	**47.23 ± 0.05**
	PHOC-Reduced	18.65 ± 0.92	56.25 ± 0.06	31.79 ± 0.54	46.59 ± 0.47
	BOW	18.96 ± 1.85	**56.54 ± 0.22**	31.74 ± 0.97	46.9 ± 0.79
Top Attention	SBERT Full	70.63 ± 1.01	82.92 ± 1.07	71.54 ± 0.58	66.17 ± 1.2
	SBERT Reduced	75.51 ± 2.74	**86.46 ± 2.37**	74.27 ± 2.19	67.96 ± 2.02
	PHOC Full	74.1 ± 1.73	85.88 ± 0.79	**75.26 ± 1.13**	**70.03 ± 0.33**
	PHOC Reduced	74.63 ± 1.3	84.79 ± 0.94	74.24 ± 1	69.23 ± 0.3
	BOW	**75.53 ± 1.01**	86.07 ± 0.83	75.09 ± 1.81	69.3 ± 2.18
VQA-CNN	SBERT-Full	72.62 ± 12.51	70.49 ± 6.7	59.23 ± 6.27	58.32 ± 6.91
	SBERT-Reduced	**78.65 ± 0.82**	76.28 ± 0.41	63.46 ± 0.67	64.18 ± 0.44
	PHOC-Full	77.85 ± 3.51	**76.96 ± 1.96**	**64.66 ± 2.79**	**66.85 ± 1.76**
	PHOC-Reduced	48.14 ± 17.31	66.56 ± 8.56	53.21 ± 11.15	63.88 ± 5.14
	BOW	71.81 ± 13.88	75.18 ± 5.08	62.83 ± 5.25	66.54 ± 2.12
(Vision only)	-	0.31 ± 0.05	25.81 ± 1.53	23.06 ± 0.44	36.97 ± 4.26
VQA-Pretrained	SBERT-Full	36.84 ± 9.76	60.13 ± 3.76	42.32 ± 8.6	48.93 ± 7.09
	SBERT-Reduced	**56.88 ± 13.2**	70.95 ± 6.78	59 ± 10.69	62.75 ± 10.08
	PHOC-Full	59.86 ± 2.89	**72.56 ± 0.18**	**60.59 ± 0.4**	**64.64 ± 0.87**
	PHOC-Reduced	59.37 ± 2.53	72.35 ± 0.86	58.05 ± 0.93	62.41 ± 0.8
	BOW	56.58 ± 4.01	72.31 ± 1.48	59.05 ± 2.09	63.78 ± 0.82
(Vision only)	-	0.26 ± 0.05	26.74 ± 1.57	22.91 ± 0.53	35.15 ± 3.95
Language only	SBERT-Full	18.34 ± 2.21	52.8 ± 0.53	31.7 ± 2.64	**47.17 ± 0.91**
	SBERT-Reduced	19.24 ± 1.98	**56.19 ± 1.52**	32.01 ± 1.93	45.67 ± 0.53
	PHOC-Full	17.76 ± 3.09	55.93 ± 1.62	**32.91 ± 0.62**	47.04 ± 0.35
	PHOC-Reduced	**19.75 ± 0.94**	56.09 ± 0.04	31.96 ± 0.75	46.98 ± 0.4
	BOW	18.22 ± 2.14	56.27 ± 0.52	31.26 ± 1.1	46.82 ± 0.65

the performance decreases from level 1 to level 3. Only for level 0 is this not the case which sits at a similar performance to level 2 for *TA*. Such a pattern is generally not present in the other architectures as level 2 is the most difficult specificity level there. *VQA-CNN* performs the best on level 0, while *Pretrained* performs much worse on this level. On the other specificity levels, *Pretrained* is slightly worse than *VQA-CNN*. Both *BTA* and *language only* once again perform similarly. The performance on levels 1 and 3 is relatively high compared to the other specificity levels. This indicates more language priors being available for these specificity levels. For level 2, this is the case to a lesser extent. Level 0 seems to have the least amount of language priors available. *BTA* and *language only* outperform *vision only* on all these levels as well. A final thing to note is that *vision only* also performs very poorly on level 0 in comparison to the other levels.

To conclude the visual question answering results, the Top attention architecture with SBERT-Reduced, PHOC, or BOW embeddings outperforms the other models throughout the results. However, it is far from perfect as it performs poorly on questions of the location type. Also, the results seem to suggest language priors. These results are further discussed in the next chapter to answer the research questions.

4.3 Image Generation

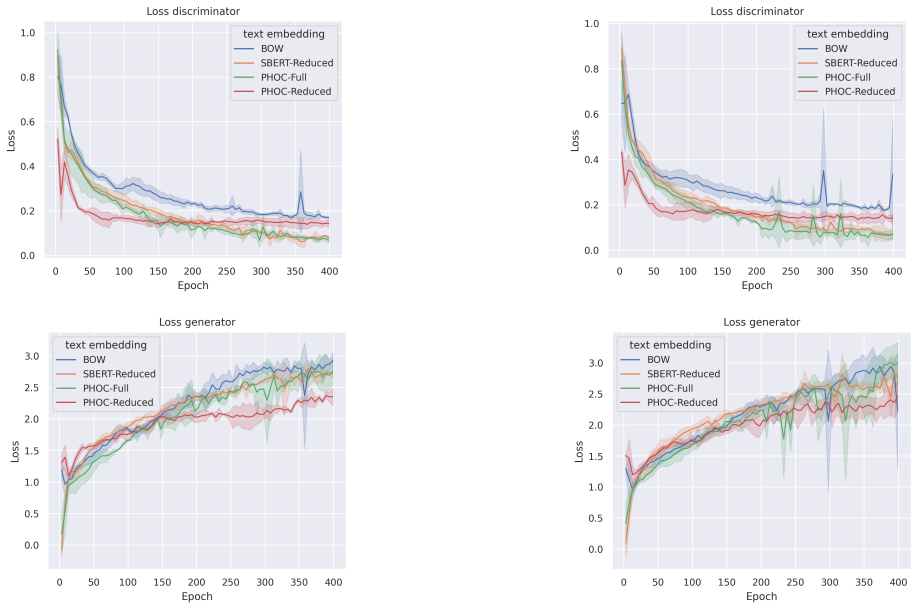
For the image generation (IG) task two setups of the DF-GAN architecture (see 2.6.1) were explored for the five text embeddings types. *Pretrained* are setups utilizing VAE pretraining of the generator. *Non-pretrained* are setups that train the GAN from scratch. In this section, the training metrics are presented first, followed by a quantitative evaluation on the test set. The final subsection is a qualitative review of the best-performing model.

4.3.1 Training

During the training of the models, the losses of the discriminator and generator were recorded for each epoch. Also, the Fréchet inception distance (FID) and Inception Score (IS) metrics were evaluated on the validation set for each epoch. The accuracy of the discriminator on fake and real inputs and the VAE loss are omitted from this section. Instead they can be found in Appendix A.2.1. The lines in the plots of all the image generation metrics were averaged every five epochs to make the plots more legible.

The loss of the discriminator and generator for both *Pretrained* and *Non-pretrained* can be seen in Figure 4.2. The loss for SBERT-Full has been omitted for legibility as its generator loss was significantly larger than the other methods. It steadily cycled between 20 and 50 throughout the training. The discriminator loss for SBERT-FULL was close to 0 throughout the training, so it was omitted as well. The *Pretrained* discriminator loss starts of higher than *Non-pretrained*, but both converge to the same level. In general, the discriminator loss steadily decreases, while the generator loss increases besides a couple of spikes. The spikes are most likely to occur later in the training for BOW and PHOC-Full. SBERT-Reduced and PHOC-Reduced are more stable. In general, the *Pretrained* loss seems to be more stable than the *Non-Pretrained* indicating the VAE pretraining shows some effectiveness.

In Figure 4.3 the validation evaluation of the Fréchet inception distance (FID) and Inception Score (IS) metrics are shown. The FID score quickly decreases, while the IS quickly rises. After about 50 epochs, FID slowly decreases. PHOC-Full and to a lesser extent SBERT-Reduced, are both unstable, sometimes spiking upwards. For *Pretrained*, this seems to occur less for these embedding types. For all methods, the IS is very unstable. However, it seems like once again, *Pretrained* is a bit more stable than *Non-pretrained* for this metric. SBERT-Full completely fails on both FID and IS, barely moving at all. These graphs also indicate the effectiveness of the VAE pretraining.



Pretrained Non-Pretrained

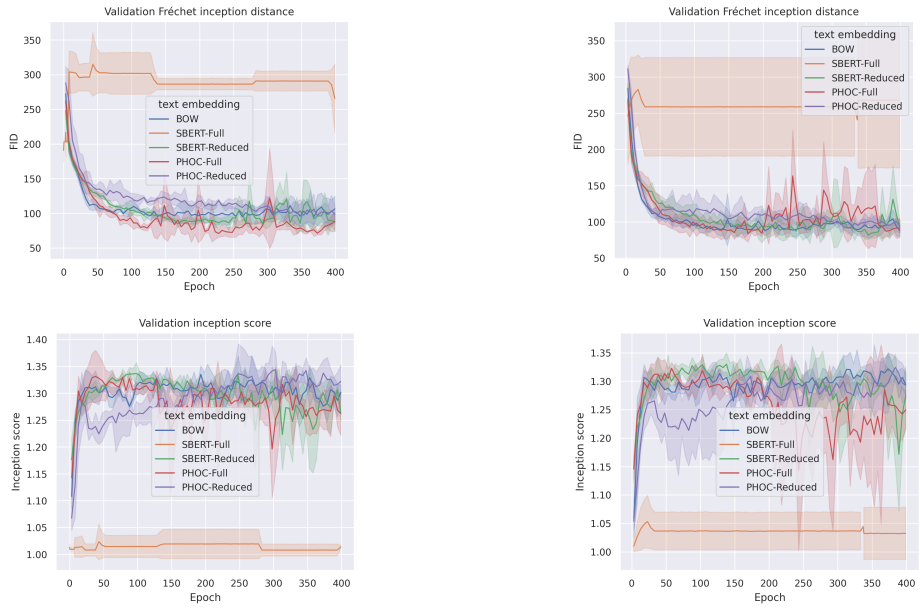
Figure 4.2: The training loss of the generator and discriminator for the VAE *Pretrained* models and the *Non-Pretrained* models. The text embedding SBERT-Full was omitted from these plots to improve the legibility.

4.3.2 Evaluation

The models were evaluated on the test set by measuring their Fréchet inception distance (FID) and Inception Score (IS). As an additional measure for semantic consistency, the accuracy of the VQA model receiving the generated images as input is determined. The best-performing VQA model utilizing the same text embedding as the IG setup was chosen to do this. Therefore the base VQA architecture for this is Top attention. In Table 4.4 these results can be seen. There is no clear consensus between the metrics. *Pretrained* PHOC-Reduced performs best on the Inception score by a slight margin. Both *Non-pretrained* and *Pretrained* PHOC-Full perform best on FID with a slight margin above *Non-Pretrained* BOW. In general, there is a large variation in the FID metric, especially for the SBERT-Reduced text embedding. Similar to the validation, SBERT-Full scores poorly on FID and Inception score. However, it scores close to the other models on the VQA consistency metric, which could indicate that this is some kind of minimum score. Therefore, SBERT-reduced being the only one that improved on the metric could indicate it can synthesize

Table 4.4: Evaluation of the image generation models on the test set. Boldface values indicate the best in the column.

Type	Text embedding	FID ↓	Inception ↑	VQA ↑
Pretrained	SBERT-Full	246.12 ± 51.99	1.02 ± 0.01	0.41 ± 0.01
	SBERT-Reduced	107.3 ± 57.23	1.19 ± 0.05	**0.47 ± 0.06**
	PHOC-Full	82.84 ± 15	1.18 ± 0.04	0.39 ± 0.04
	PHOC-Reduced	96 ± 20.78	**1.25 ± 0**	0.41 ± 0.03
	BOW	93.73 ± 11.13	1.23 ± 0.01	0.4 ± 0
Non-pretrained	SBERT-Full	253.92 ± 64.35	1.03 ± 0.03	0.41 ± 0
	SBERT-Reduced	110.78 ± 49.55	1.16 ± 0.12	**0.47 ± 0.05**
	PHOC-Full	**82.27 ± 10.44**	1.18 ± 0	0.4 ± 0.04
	PHOC-Reduced	92.16 ± 14.37	1.21 ± 0.04	0.4 ± 0.01
	BOW	83.6 ± 7.52	1.2 ± 0.02	0.39 ± 0.02



Pretrained Non-Pretrained

Figure 4.3: The Fréchet inception distance (FID) and Inception score on the validation set for the VAE-Pretrained models and the Non-Pretrained models.

more semantically accurate images.

4.3.3 Qualitative review

To perform the qualitative review of the models ten representative question-answer (QA) pairs of the different question types and specificity levels were selected. Throughout the qualitative review, the best of the three runs based on the FID metric are used for the models. The qualitative review contains three parts. First, a single image is sampled for each of the ten QA pairs by each model. Second, the best model based on the first qualitative review is selected to sample multiple images for the QA pairs. This was done to look at the variation of the model. Third, the same process as the second review is done but with noise applied to the text embeddings to determine its robustness to this variation.

In Figure 4.4 the first qualitative review can be seen. The general quality of the images is good. Sometimes the shape is too small, making it hard to discern what the shape is supposed to be. Also, for the PHOC-Full models, the 3rd QA pair causes poor-looking images. However, these low-quality images only occur rarely in general. A large problem that is visible for the PHOC and BOW models concerns empty images. In these images, only the background texture is displayed without any shapes. It is important to note that such images do not occur in the data set; Each image has at least one shape. Therefore, this could be an indication of a mode collapse happening for the respective models. For the SBERT-Full models, this happens for each question, indicating a complete collapse in the model.

Looking at the semantic consistency of the images with the QA pair, most methods struggle. PHOC-FULL performs poorly, rarely matching attributes from the QA Pair even in the non-empty images it generates. PHOC-Reduced is slightly better but still suffers from the same problem. BOW suffers less from empty images but similarly fails to fully match the QA pair. By far, the best at this is SBERT-Reduced, fully matching the QA pairs more often and generally at least partially. This matched the results of the evaluation section, where SBERT-Reduced was the only

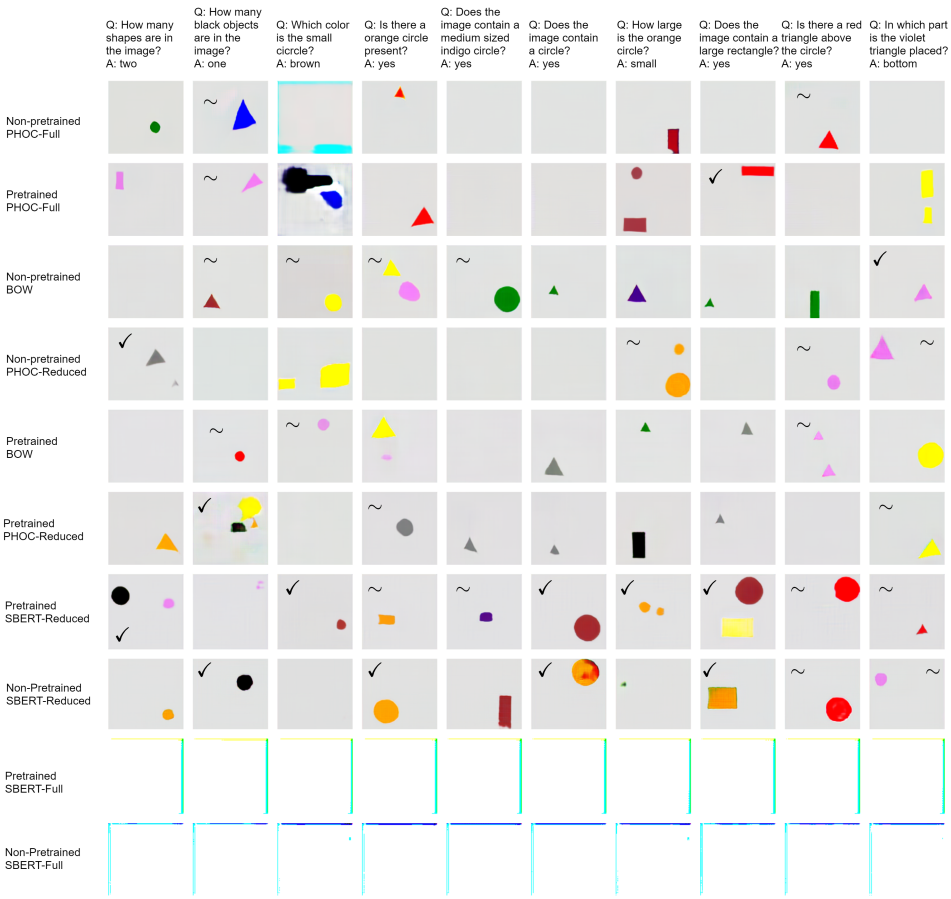


Figure 4.4: Synthesized images for a selection of question-answer (QA) pairs. The models are sorted from best to worst on the Fréchet inception distance metric. A ✓ sign indicates that the image semantically fully matches the QA pair. A ∼ sign indicates that one or more attributes present in the QA are present in the image. No indicator means the image has no semantic relation with the QA. Note that the two bottom rows are images from a model that failed to learn in every run.

model with a higher VQA accuracy. Therefore, this also is an indication that the VQA metric works to determine semantic consistency. This first qualitative review shows that a better FID or Inception score does not necessarily indicate better semantic consistency as SBERT-Reduced outperforms the others.

The second qualitative review is shown in Figure 4.5. It shows that the SBERT-Reduced model produces variation mostly in the location and size of the shapes. In general, it gets a similar amount of questions (partially) correct throughout the different tries. What seems clear from the retries is that it struggles to semantically match more than two attributes from the question-answer (QA) pair. The best example of this is question 5 and 7. For question 5, in row A, it matches the size and color attribute but fails the shape; In row D, it matches the shape and color but fails the size. Only in row C does it get it correct. For question 7, in rows A, B, and C, it matches the color and size but fails the shape; In row D, it does get it correct. It also seems like it struggles with localization, as it never manages to get the location attribute correct in questions 9 and 10. Seemingly it determines the location randomly throughout the synthesized images.

The third qualitative review can be seen in Figure 4.6. The images are of similar quality to the retried images in Figure 4.5. The main noticeable difference is the slight noise in the colors for question 6 and also the rectangles in question 1. The noise also does not seem to cause

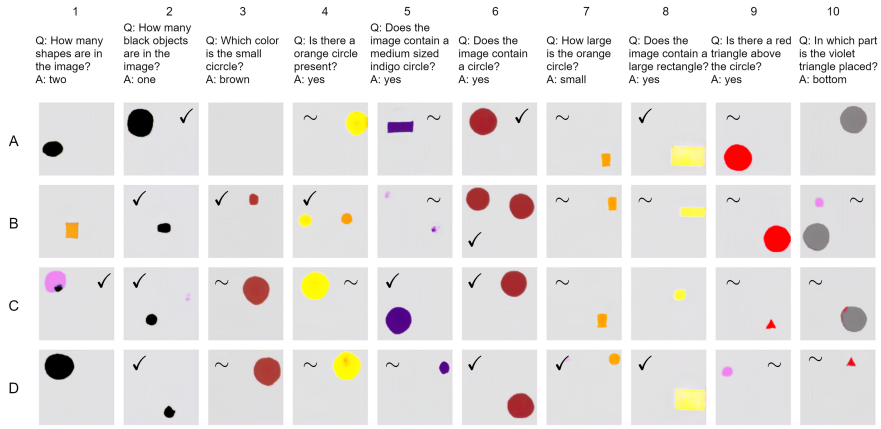


Figure 4.5: Generated images for SBERT-Reduced *Pretrained* four times with a different latent space each time. Q is the question and A the answer together QA, used to generate the image. A ✓ sign indicates that the image semantically fully matches the QA combination. A ∼ sign indicates that one or more attributes present in the QA are present in the image. No indicator means the image has no semantic relation with the QA.

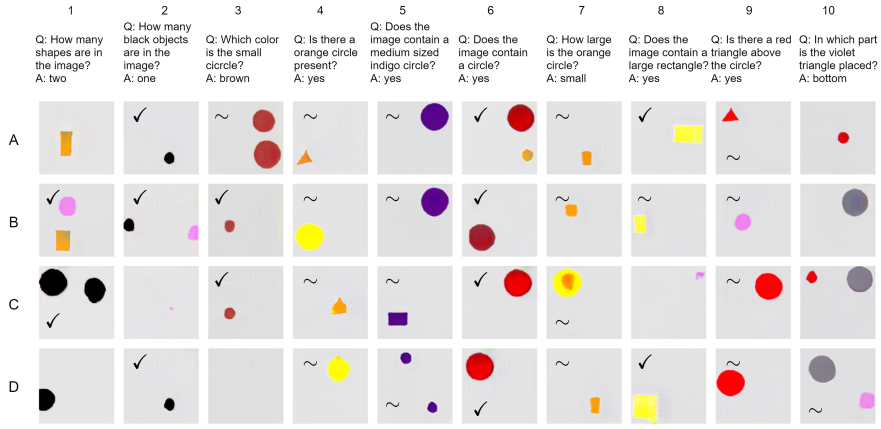


Figure 4.6: Generated images for SBERT-Reduced *Pretrained* with random noise applied to the question-answer (QA) embedding four times. Q is the question and A the answer together QA, used to generate the image. A ✓ sign indicates that the image semantically fully matches the QA combination. A ∼ sign indicates that one or more attributes present in the QA are present in the image. No indicator means the image has no semantic relation with the QA.

lower semantic consistency. In general, the difference between the retried images of Figure 4.5 is minimal. Therefore we can say that applying noise to the SBERT-Reduced embeddings does not have a significant effect on the performance of the model.

To conclude the qualitative review, it is clear that the FID and Inception score metrics do not match with the actual quality of the images we saw in the review. Only the VQA consistency metric was able to match the insight of the qualitative review that SBERT-Reduced produced the most semantically consistent images.

4.4 Cyclic

To explore the applications of cyclic architectures towards improving generalizability and robustness a cyclic architecture was experimented with for each of the two tasks. Both the best visual question answering (VQA) and image generation (IG) models were further trained through cyclic architectures. The best VQA architecture was Top attention (TA) in the evaluation and is used in the cyclic architectures. In IG, the most semantically consistent model utilized the SBERT-Reduced text embedding with VAE-Pretraining. For the cycle architecture, both models require the same text embedding dimension. Therefore, *TA* + SBERT-Reduced was used for VQA and *Pretrained* + SBERT-Reduced for IG. For both, the different loss setups are explored. In this section, the results of VQA further training are first shown followed by IG.

4.4.1 Visual question answering

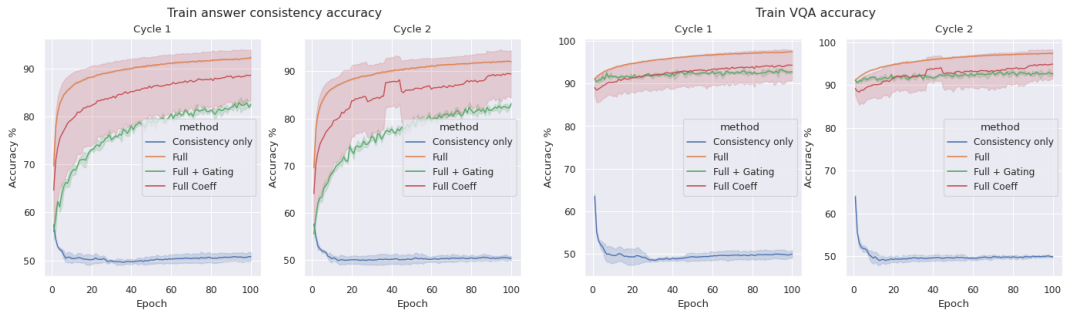


Figure 4.7: The accuracies during the cycle training of VQA. Answer consistency is the accuracy of the second forward pass, while VQA is the first forward pass.

Table 4.5: General accuracies of cyclically further trained VQA model evaluated on the test set. The bold-faced numbers indicate the best value for the cycle.

	Full	Yes/No	Open
Top attention + SBERT-Reduced	78.61 ± 2.28	86 ± 1.92	71.19 ± 2.64
Cycle 1			
VQA only	75 ± 0.22	83.1 ± 0.4	66.86 ± 0.39
Consistency only	43.99 ± 0.87	63.07 ± 0.41	24.95 ± 1.37
Full	74.37 ± 0.38	82.43 ± 0.34	66.28 ± 0.44
Full + Gating	**76.34 ± 0.31**	**84.17 ± 0.34**	**68.47 ± 0.28**
Full Coeff	73.8 ± 0.83	81.73 ± 0.82	65.87 ± 0.99
Full Coeff + Gating	73.76 ± 0.48	82.57 ± 0.43	64.9 ± 0.59
Cycle 2			
VQA only	74.77 ± 0.49	82.39 ± 0.73	67.1 ± 0.26
Consistency only	43.6 ± 0.3	62.04 ± 0.96	25.18 ± 0.86
Full	74.91 ± 0.3	82.94 ± 0.18	**66.89 ± 0.49**
Full + Gating	**75.88 ± 0.34**	**83.72 ± 0.26**	68.01 ± 0.46
Full Coeff	73.26 ± 0.34	81.59 ± 0.61	64.94 ± 0.16
Full Coeff + Gating	73.47 ± 0.36	82.25 ± 0.4	64.66 ± 1.06

Table 4.6: Accuracies for the different types of questions for the cyclically further trained VQA model evaluated on the test set. The bold-faced numbers indicate the best value for the cycle.

	Size	Shape	Color	Location	Count
Top attention + SBERT-Reduced	79.29 ± 1.91	90.54 ± 1.65	91.66 ± 1.89	57.73 ± 3.18	86.44 ± 2.17
Cycle 1					
VQA only	77.16 ± 1.54	88.99 ± 0.72	88.83 ± 0.32	50.99 ± 0.18	84.46 ± 0.39
Consistency only	37.26 ± 0.5	72.41 ± 0.25	37.78 ± 3.04	25.93 ± 0.11	70.37 ± 0.07
Full	76.74 ± 0.66	88.91 ± 0.32	87.42 ± 0.94	50.57 ± 0.29	84.16 ± 0.26
Full + Gating	**79.96 ± 0.56**	**91.16 ± 0.39**	**90.43 ± 0.42**	**51.94 ± 0.78**	**84.6 ± 0.29**
Full Coeff	75.87 ± 2.02	87.98 ± 0.76	85.64 ± 0.56	51.26 ± 0.55	83.64 ± 1.07
Full Coeff + Gating	75.92 ± 2.03	89.84 ± 1.09	89.01 ± 0.52	48.17 ± 1.05	82.69 ± 0.83
Cycle 2					
VQA only	79.17 ± 0.84	88.47 ± 0.81	88.04 ± 0.65	50.61 ± 1.26	83.76 ± 0.54
Consistency only	37.68 ± 2	71.8 ± 0.54	37.1 ± 1.01	25.57 ± 0.49	69.88 ± 0.33
Full	78.8 ± 0.12	89.7 ± 0.26	88.4 ± 0.21	50.39 ± 0.74	84.02 ± 0.69
Full + Gating	**79.93 ± 1.23**	**90.53 ± 0.98**	**89.43 ± 0.36**	**51.58 ± 0.61**	**84.51 ± 0.28**
Full Coeff	75.26 ± 0.67	87.38 ± 1.04	86.08 ± 0.83	49.97 ± 0.21	83.05 ± 0.31
Full Coeff + Gating	76.74 ± 1.83	88.81 ± 0.26	87.51 ± 0.49	48.53 ± 1.24	82.63 ± 0.22

Table 4.7: Accuracies for the different specificity levels of questions for the cyclically further trained VQA model evaluated on the test set. The bold-faced numbers indicate the best value for the cycle.

	Level 0	Level 1	Level 2	Level 3
Top attention + SBERT-Reduced	75.51 ± 2.74	86.46 ± 2.37	74.27 ± 2.19	67.96 ± 2.02
Cycle 1				
VQA only	72.16 ± 0.61	84.13 ± 0.04	69.69 ± 0.64	62.8 ± 0.38
Consistency only	18.12 ± 2.97	57.11 ± 0.81	32.75 ± 0.95	43.92 ± 0.4
Full	70.29 ± 0.8	83.43 ± 0.35	69.14 ± 0.58	62.74 ± 0.3
Full + Gating	**72.32 ± 0.38**	**84.94 ± 0.14**	**71.65 ± 0.34**	**64.7 ± 0.94**
Full Coeff	69.08 ± 1	83 ± 1.16	68.88 ± 0.57	61.31 ± 0.55
Full Coeff + Gating	68.13 ± 0.82	83.19 ± 0.43	68.47 ± 0.24	61.86 ± 1.16
Cycle 2				
VQA only	70.68 ± 0.52	83.95 ± 0.32	69.65 ± 0.69	62.54 ± 0.84
Consistency only	18.84 ± 1.11	56.55 ± 0.3	32.19 ± 0.65	43.96 ± 0.17
Full	**71.11 ± 2.83**	83.82 ± 0.72	69.76 ± 0.47	63.39 ± 0.25
Full + Gating	70.51 ± 1.23	**84.77 ± 0.34**	**71.13 ± 0.74**	**64.14 ± 0.89**
Full Coeff	69.7 ± 0.85	82.57 ± 0.46	67.89 ± 0.28	61.03 ± 0.45
Full Coeff + Gating	67.96 ± 3.07	82.51 ± 0.52	68.58 ± 0.89	61.72 ± 1.27

Throughout this section, cycle 1 is the original VQA model further trained by the original IG model. Cycle 2 is the original VQA model further trained by the further trained IG model (IG model after cycle 1 in the next section). The training accuracies as seen in Figure 4.7 show that the VQA accuracy slightly increases, while the consistency accuracy significantly increases during the training for most loss methods. The training answer consistency accuracy is the highest for the *Full Coeff* and *Full* loss setups. For the *Full + Gating setup*, this is a bit lower. These accuracies indicate that the model stays good at answering questions on the real images while also learning to do this on the variation of the synthesized images.

In Table 4.5 the general performance on the test set after cyclical training can be seen. The loss setups are close together, but the *Full + Gating* setup edges out the other methods. The *Full + Gating* setup might perform better because it has less answer consistency accuracy, thus being more focused on the general VQA accuracy. The addition of gating does not seem to have the same positive effect for the *Full Coeff* setup as for the *Full* setup. The differences between the original model and the cycles are minimal throughout the categories looking at the best setup *Full + Gating*. Cycle 1 slightly reduces the performance compared to the original model and cycle 2 reduces it slightly compared to cycle 1.

In Table 4.6 the accuracies for the different question types are shown. It shows a significant decrease in the location category for the best loss method. Note that the image generation (IG) model also performed poorly on the location attribute. The color and count types also show a decrease but much less. Conversely, the size and shape types show a slight increase. Cycle 2 once

again performs slightly worse than Cycle 1 throughout the types. The other information on the questions was the specificity level. The results of the cyclic architecture for VQA are in Table 4.7 with regards to the specificity levels. It shows that the performance on the specificity levels decreases for each level at a similar rate in cycle 1. Once again cycle 2 reduces the performance of cycle 1 slightly.

The cyclic results for VQA showed that the general performance of the model decreased in cycle 1 and further in cycle 2. However, in cycle 1 the performance on some of the question types did improve. Cycle 2 did not improve in any area on cycle 1. The cycle models did manage to learn the variation of the IG model as indicated by the training plots.

4.4.2 Image generation

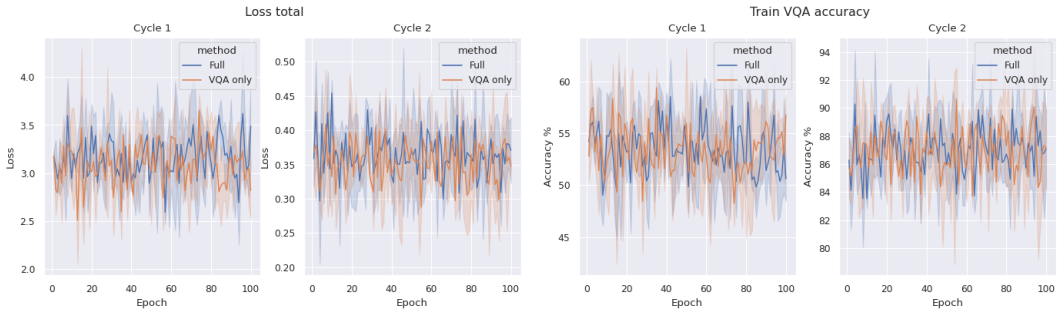


Figure 4.8: Training loss and visual question answering accuracy (VQA) plots for the cyclical training of the image generation model.

Table 4.8: Evaluation of the image generation model further trained by cycle architecture on the test set.

	FID ↓	Inception ↑	VQA ↑
Pretrained + SBERT-Reduced	72.87	1.22	0.52
Cycle 1			
VQA	72.99 ± 0.33	1.29 ± 0	0.52 ± 0.01
VQA + Image Consistency	72.98 ± 0.52	1.3 ± 0	0.51 ± 0.01
Cycle 2			
VQA	72.65 ± 0.63	1.29 ± 0	0.83 ± 0.01
VQA + Image Consistency	72.5 ± 0.69	1.3 ± 0	0.83 ± 0.01

In this section, Cycle 1 is the original IG model further trained by the original VQA model. Cycle 2 is the original IG model further trained by the VQA model after cycle 1 from the previous section. The evaluation of the models is shown in Table 4.8. It shows there are no significant differences in the FID and VQA metrics for Cycle 1 and the loss methods. The Inception score does increase through the cycle training. Cycle 2 shows the same as cycle 1, but the VQA accuracy is much improved. The VQA model that evaluates the metric is the one after cycle 1 of the previous section. Therefore, this shows that the VQA model has learned to predict answers on the IG images well. The loss and VQA training accuracy plots in Figure 4.8 also reflect these results. The loss cycles but does not decrease. The same is the case for the VQA accuracy. The FID and Inception scores were also evaluated on the validation set during the training but are omitted from this section as they showed nothing exceptional. Instead it can be found in Appendix A.3.2.

In Figure 4.9 a short qualitative review of the generated images after the cycle also shows no major differences. It does seem like Cycle 2 produces slightly more semantically correct images, but most likely this is just a random chance. Especially because the VQA accuracy during Cycle

2 training does not improve. Therefore, it seems like the qualitative review matches with the insights from the quantitative results in that there are no major differences between the cycles and the original model.

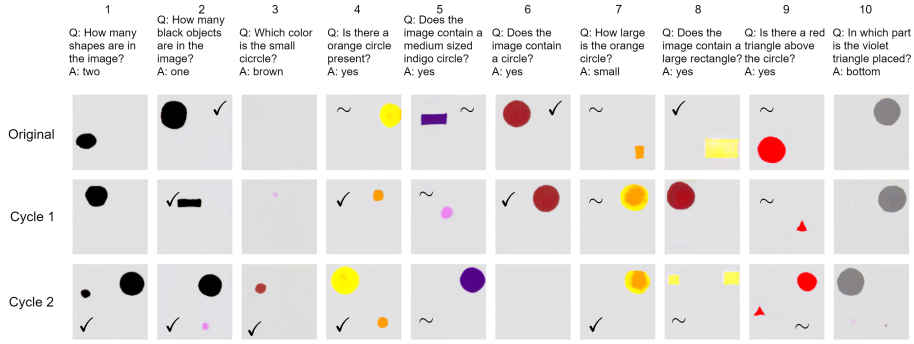


Figure 4.9: Qualitative review of the image generation models after cyclical training. A ✓ indicates that the image semantically fully matches the questin-answer (QA) combination. A ∼ sign indicates that one or more attributes present in the QA are present in the image. No indicator means the image has no semantic relation with the QA.

The experiments with the cyclic experiments cover the final part of the results. The results for the different tasks are applied to answer the research questions in the next chapter. The most interesting insights from them are presented and conclusions for future work are drawn from them there.

Chapter 5

Discussion

In this final chapter, the results from the experiments are used to answer the research questions. Conclusions are drawn from this and future work possibilities are discussed. The main goal of this thesis was to explore the generalizability and robustness in visual question answering (VQA) and image generation (IG) tasks. Additionally, the applications of cyclic architectures towards improving this were explored. To introduce an evaluation metric to IG that measures semantic consistency, a setup utilizing a VQA model to perform this was also introduced. The study was performed on the ShapeVQA data set containing images of abstract stylized shapes. The data set was purpose-built to explore the generalizability and robustness of the models. In this chapter, the results of the separate tasks are first discussed followed by the results of the cyclic architectures.

5.1 Can visual question answering models show degrees of generalizability and robustness?

To receive real-world applications it is important models show high degrees of generalizability and robustness. Visual question answering (VQA) models have many potential real-world applications but are lacking in their generalizability and robustness due to abuse of language priors [15], failure to understand rephrasings [43] and general lack of performance among other things. Therefore, in this thesis, a study was done to determine the current abilities of the VQA models to show degrees of generalizability and robustness, giving a deeper understanding of the models. To perform the study multiple setups of VQA architectures were trained on the abstract and simplified ShapeVQA data set. In the rest of this section, the results from these experiments are discussed.

5.1.1 Comparing the architectures

To compare the generalizability of the different architectures, we can look at their general performance on the test set. High performance on this set indicates the model was able to generalize well from training to testing. In the ShapeVQA data set, this is made more difficult by ensuring the unique attribute combinations of the train set do not exist in the test set. Therefore the models require compositionality relating the parts of the attribute combinations it has seen in the train set to the unseen ones in the test set to generalize well.

There were two main architectures explored, an attentional joint-embedding architecture and a non-attentional one. For the attentional architecture, a version with Bottom + Top attention (*BTA*) and one with only Top attention (*TA*) was explored. For the non-attentional architecture, a version with pre-trained VGG16 visual features called *VQA-Pretrained* and one with visual features from a simple CNN called *VQA-CNN* trained from scratch was explored. Considering the results of previous VQA research, it would be expected that the *BTA* should outperform the *TA* model, which in turn should outperform the non-attentional joint-embedding approaches of *VQA-CNN* and *VQA-Pretrained* [4]. Partly the results show this as *TA* performs better than *VQA-CNN* and *VQA-Pretrained*. However, *BTA* performs worse than all of these methods. This mismatch with previous research can be explained by the failure of the pre-trained FRCNN visual features which were used for the *BTA* architecture. In the setup of this FRCNN, the thresholds had to be set at a very low value to find any visual features, of which there also were not many. This caused the visual part of the *BTA* to fail. Further proof for this can be seen by comparing the results of this model with the *language-only* model. The *language-only* model was trained by

omitting the visual part of the VQA problem. The full accuracy of *BTA* and *language only* is within 1% of each other as seen in Table 4.1, but more notably across all types and specificities, they perform similarly as seen in Table 4.2 and Table 4.3. This indicates that the *BTA* model is only able to utilize its language part failing to utilize the visual component. The FRCNN was pre-trained on a much different data set containing real-world images (MSCOCO [29]). Therefore, it is clear that this does not generalize well to this kind of simple shape data set. To properly explore this model, the FRCNN model would either have to be finetuned or trained from scratch on this data set. This would require major modifications to the data set, such as storing bounding boxes with class labels of the shapes. Unfortunately, this fell out of the scope of this thesis but would be interesting for future work.

The pre-trained FRCNN visual features did not work adequately on this VQA task, but the other pre-trained visual features performed much better. The *TA* architecture utilized the strength of Resnet18 [18] visual features by attending to the most important features guided by the question. It performs the best throughout the general accuracies outperforming both non-attentional joint-embedding architectures of *VQA-CNN* and *VQA-Pretrained*. While the CNN convolutional visual features outperform the pre-trained visual features of VGG16 [44] they do not outperform the combination of top-attention with the pre-trained visual features of Resnet18. The top-attention mechanism improves the strength of the general features by using the question information to guide attention to the most relevant features. The CNN can learn data set-specific features, which could be the reason it outperforms the VGG16 visual features in the same architecture, but the attentional mechanism bridges this gap for the TA architecture. Therefore, our results suggest that top-down visual attention is an effective tool for improving the generalizability of the VQA architectures and the *TA* architecture is the best approach to VQA for this data set.

Interesting to note is the fact that *TA* utilized the least amount of parameters out of the architectures (ignoring *language only*). The amount of parameters in a model directly influences the amount of information that can be stored in the model. Too few parameters would mean there is not enough space in the network to learn, while too many could lead to overfitting and lookup-table behavior. It seems like the TA architecture strikes a good balance in the number of parameters, while also being able to apply them the best for learning and generalizability.

5.1.2 Comparing the text embeddings

The differences in the architectures mostly lie in how they handle the visual part of the VQA task like for example applying attention to it or not. In the experiments, multiple approaches to the text embedding aspect of the architectures were also explored. There were three main embedding methods explored, Sentence-BERT (SBERT), Bag of words (BOW) and Pyramidal Histogram of Characters (PHOC). The main differences in the embedding types lie in the level they look at the sentence. SBERT looks at the sentence level by using the semantic relations between the words to generate the embedding. BOW looks at the word level by encoding the presence of words in the embedding. PHOC looks at the character level by encoding the positioning of separate characters through the sub-segments of the words in the sentence. The PHOC and SBERT embeddings were also dimensionally reduced to form two extra embedding types to see if a more dense embedding space is also effective for the problem. The three main embedding types and the dimensionally reduced counterpart of PHOC show similar performance. The only type that differs is the SBERT-Full version which performs worse than all the other methods. The SBERT embeddings saw a large reduction in dimensions from 768 to 70, which improved the performance to the same level as the other methods. The high dimensionality of the full version most likely causes too much sparsity in the embedding space resulting in the reduced performance. Therefore, a more dense reduced dimensionality works better for this embedding method. For PHOC the dimensionality reduction from 256 to 31 showed no significant improvements, so sparsity in the embedding space seems to be less of an issue for this method.

It would be assumed that understanding the semantic relations between words in a question should be important to answer it. However, the results showed that the SBERT embedding method that does take this into account does not outperform the others that do not. This could

be attributed to the relatively low amount of questions and their simplicity. For example, the VQA 1.0 data set [5] has 614,163 total questions about real-world images, while ShapeVQA only has 15000 about simple images in a reduced problem space. Naturally, the questions in VQA 1.0 are more complicated as they concern the real world. It is possible that on such a data set the semantic information present in the SBERT method makes more of a difference. This would be interesting to explore in future research. In such a study it would be interesting to compare the embedding methods with an LSTM embedding approach as it has been commonly used throughout VQA research. For this study, we can say that the extra semantic information present in the SBERT embeddings does not provide a significant advantage over the other embedding methods for VQA. Additionally, we can say that there is no significant difference in the generalizability of the three embedding methods, besides the difference between SBERT and its reduced version. For SBERT a reduced representation improves the generalizability.

For real-world applications, it is important to note that BOW would not be able to generalize well to questions outside of the data set. Due to the design of BOW, it is limited to words that are in the vocabulary of the data set. Any other words would be labeled as out-of-vocabulary (OOV), which means their information would not be used properly. Also, any spelling errors or other linguistic variations would cause words to be labeled as OOV. Therefore, it would not be a viable method for real-world applications where the input is varied and OOV words are likely. PHOC can handle this varied input well as it looks at the character level. Likewise, the base BERT model used in SBERT utilizes WordPiece embeddings [53], which also solve OOV words by looking at the character level. Additionally, the base BERT model was trained on a large corpus and SBERT was finetuned even more on a Natural Language Inference task, thus having seen a large enough number of words to cover most possibilities in combination with the WordPiece embeddings. Therefore for any real-world applications, the PHOC and SBERT embeddings would be a better choice as these are more robust.

5.1.3 Question types and specificities

In the ShapeVQA data set, each question has a type assigned based on the most important attribute required to answer it and a specificity level based on how specific the question is. By looking at the results for the different types and specificity levels, we can say something about the robustness of the models. Comparing the *language only* results with the *vision only* results shows that the model abuses language priors for count questions and somewhat for shape questions because the *language only* results are much higher. This makes it hard to say something definitive about the ability of VQA models to handle these types. However, the full VQA architectures do improve their performances on these types, indicating that the visual part is still important. When looking at the best performing architecture *TA* we see that it excels in the color type, improving almost 20% on the second-best architecture. A large part of the image is background, so a lot of the visual features are irrelevant for questions concerned with the color of the shapes. The attention mechanism in *TA* seems to guide the model well to the visual features containing only the colors of the shapes for these questions. Because the other methods lack the attention mechanism, they consider each image feature equally, even the irrelevant ones, ultimately resulting in worse performance. The same reasoning most likely also applies to the other types *TA* excels in as well. Only for the location type does *TA* not improve on the other architectures. However, all the methods perform poorly on the location type indicating that the VQA architectures are not robust against questions requiring spatial understanding. A reason for this could be the lack of spatial relations in the CNN image embeddings. A CNN can extract high-level features from an image but does not keep track of the spatial relationships between them. Therefore, this can be a large bottleneck in understanding the spatial relations between the features. Most likely, when moving to more complex questions regarding multiple objects understanding spatial dependencies becomes even more important to show generalizability.

When looking at the results at the specificity levels in Table 4.3, there are no clear patterns in the results for most models. Generally, level 2 or 3 are the most difficult (lowest performance), but sometimes level 0 is too. What we do see is that for levels 1 and 3 the performance is significantly

higher than levels 0 and 2 for the *language only* models. This suggests that questions at these specificity levels have statistical information the models can exploit. Consequently, this could explain why for most models there is no clear pattern, as these language priors affect the results. In the Top attention (TA) architecture, which performed the best, we do see an interesting pattern. The performance decreases the more specific the questions get, except for level 0. This matches the intuition that a higher specificity means a more difficult question. A higher specificity means there are more elements of the question the model has to relate to the visual part, thus possibly making it more difficult. The lower performance on level 0 could mean that the model requires at least some specificity to guide the question with the visual part to perform best. However, it could also be that the low specificity of this level lends itself less to the exploitation of statistical priors. The performance of the *language only* model supports this as it indicates a low amount of language priors for this specificity level. Also, the *vision only* model performed much worse on level 0 than the other levels, even performing close to zero on this level. This gives further indicating that the low specificity of this level makes it harder to exploit statistical priors. These results suggest that while being the simplest questions due to the low specificity, they are difficult due to a lack of available statistical priors.

It is hard to say something definitive about the specificity levels due to these language priors. It is possible that for *TA* the results indicate that a higher specificity level means a higher difficulty of the VQA problem, ultimately requiring more generalizability. However, it is also possible that *TA* is better at exploiting the language priors due to the attention mechanism. To say something definitive the language priors would need to be eliminated from the data set through more diverse question-answer pairs in the data set.

Even with the simplifications of the ShapeVQA data set, we saw that there are still areas the VQA models clearly struggle with like the location question. While the images of the ShapeVQA data set are much simpler than real-world images, the lower amount of visual content and the monotony of the background texture raises other problems for the CNN architectures used for the visual embeddings. The most salient issue with this was for the FRCNN which completely failed. The difference between the top-attention architecture and the other architectures can be explained by its ability to give the most weight to the important visual features. The language part also showed significant use of language priors for some of the question types, which gives further proof that these play a large role in the performance of VQA models. Although these points hold, we can say that the VQA models can show degrees of generalizability and robustness, as they improve on the *language only* models on the question types with language priors and can generalize to the unseen attribute combinations of the test set. The top-attention mechanism also proved to be an important mechanism for improving the generalizability of the models.

5.2 Can image generating models show degrees of generalizability and robustness?

As discussed in the first chapter, current image generation (IG) approaches work well on narrow data sets such as CUB birds [50], but struggle when moving to more complex multi-object data sets like MSCOCO [29]. A reason for this could be that the model functions as a lookup table, being able to match important visual features with textual features, but ultimately failing to understand underlying relations between them and in turn struggling in a more complex setup that requires such understanding. If this is the case, the generalizability of the models would be poor, as deeper understanding is required to generalize well and consistently from training. Additionally, for a real-world application like data augmentation, the models must be able to show compositionality in generating new unseen data by generalizing concepts from training. For such an application it is also important that the models are robust in their generation, creating variety while being semantically consistent throughout. Therefore it is important to understand the generalizability and robustness of IG models. To explore this experiments were done by modifying the IG task to work on the question-answer pair text of the ShapeVQA data set. By

generating from question-answer pairs instead of a descriptive caption the task requires more from the model as it needs to semantically relate the two sentences, ultimately requiring large degrees of generalizability. In addition to the study on generalizability and robustness, a semantic consistency metric utilizing the VQA architectures discussed in the previous section is a critic for the IG model because current metrics fail to measure this. The experiments were done on the DF-GAN image generation architecture [48] with multiple text embedding approaches. In this section, the results of these experiments towards the generalizability and robustness of the IG models and the effectiveness of the VQA consistency metric are discussed.

5.2.1 Comparing the models

A problem faced in the training of the image generation (IG) models was instability due to the discriminator overpowering the generator at the start of training. To avoid this a method to pretrain the generator with a variational autoencoder (VAE) was explored. For each text embedding type, a pretrained and non-pretrained version was trained. The loss for the pretrained IG models looks to be more stable, with fewer and lower amplitude peaks. Also, the discriminator loss starts higher, indicating the generator is better at fooling the discriminator at the start. However, some of the instability is still unavoidable.

In the quantitative review, the best-performing methods differ between the metrics. On the FID score, the non-pretrained and pretrained versions of PHOC-Full and BOW perform the best, while on the Inception metric pretrained PHOC-Reduced and pretrained BOW performs best. The VQA consistency metric also differs by having SBERT-Reduced perform the best. Therefore, there is no clear consensus on the best method from looking at the quantitative results. The improved stability of pretraining also does not show any significant improvement. Therefore we need to look at the qualitative review to determine the differences in the models. In general, there seems to be a mismatch between the qualitative and quantitative review when considering the FID metric. When looking at the best-performing models on the FID metric, we see that they perform poorly in the qualitative review. They show signs of mode collapse by producing empty images frequently. The other methods besides SBERT-Reduced do as well but to a much lesser extent. A reason for the fact that the methods performing best on the FID metric perform poorly on the qualitative review could have to do with a large number of background features present in the mode collapsed images. The background is the largest part of the images in the data set and is the same color and texture throughout. Therefore, the FID metric could be seeing the large presence of background features in these synthesized images a high-quality as it matches the data set. Thus, the metric is not effective at measuring the quality of the produced images for this data set

As the FID metric does not seem to be effective here and the models performing best on the Inception metric show similar mode collapse behavior, we can say that the SBERT-Reduced embedding type is most effective following the qualitative review. A reason that this method performs worse on the FID and Inception metrics could have to do with the large standard deviations it has for these metrics. The validation plots for these scores show performance degradation peaks happening in the later stages of training. From a deeper look into training SBERT-Reduced, these are related to mode collapse, being solved after some epochs (see Appendix A.2.1 for an example at epoch 350). Therefore one of the SBERT-Reduced runs may have hit a mode collapse around the final epochs, reducing the mean performance.

Like in the VQA task, the dimensionality reduction of the SBERT embeddings is important to make it work effectively for the task. In this case, it is even more important as the SBERT-Full models completely fail by producing junk images. The high-dimensions of the SBERT-Full embeddings seem to allow the discriminator to completely overpower the generator at the start. Even pretraining is not enough to avoid this. Therefore, reducing the dimensionality of high-dimensional embeddings could be a potential solution to the discriminator overpowering the generator for future IG research.

5.2.2 VQA consistency metric

In the previous section, we saw that the quantitative FID and Inception metrics do not match the reality of the qualitative review. The only metric that matches the semantic consistency and quality of the images was the VQA consistency metric, which determined SBERT-Reduced as the clear best model. It seems like there is a kind of base accuracy at around 40% as the SBERT-Full method can achieve this with junk images. The best accuracy on the VQA consistency metric is quite low but is a significant improvement relative to this base accuracy.

A problem with this metric is that it assumes an effective VQA model as the quality of the VQA model influences its ability to criticize the IG model. Therefore it is hard to use this metric as an absolute quantitative evaluation. Instead, it should function as a way to compare the models in a relative sense. Knowing this, we can say that the VQA consistency metric is effective at measuring semantic consistency, but it can not be used as a direct quantitative measure. Instead, it has to be used as a metric of the relative difference in semantic consistency. Also, it can not be used to compare studies if different VQA models were used as a critic, because the difference in the effectiveness of the VQA models affects the measure.

5.2.3 Robustness and generalizability

From the qualitative review, it seemed like the pretraining improved the semantic consistency of the SBERT-Reduced approach slightly. Therefore the pretrained SBERT-Reduced model was chosen as the best model to further evaluate in the qualitative review. Looking at the first qualitative review, we see that it can get one or two of the attributes in the questions correctly portrayed in the image but often fails to capture the full essence of the question-answer pair. Instead of showing a deep understanding of the question-answer pair, the models show the lookup-table-like behavior as discussed earlier. In a look-up table, it would be hard to accurately match multiple attributes at the same time as the test set utilizes different unique combinations of attributes than the train set. Therefore it fails to do this in the qualitative review. It is also hard to say the model shows much compositionality. When generating the images multiple times in Figure 4.5, it produces different attribute combinations for the same question-answer pair, but these are often not semantically consistent. Thus it does not seem to show the deep understanding required for compositionality.

Looking deeper into the types of problems the model faces, it especially struggles with localization questions such as questions 9 and 10 in Figure 4.4. It never manages to get the localization right even when retried multiple times (see Figure 4.5). This could also be related to the main variations in the images, which are the location and size of the shapes. The variation on the location does not ensure the location attribute matches the question-answer pair. The problem with localization could also be a similar reason to the VQA model struggling with these attributes. Namely, the lack of spatial understanding between features in convolutional layers as these layers are used throughout the DF-GAN architecture to generate visual features.

The reason SBERT-Reduced outperforms the other embedding types on semantic consistency may be because of the semantic power its embeddings have. A possibility could be that the model can effectively use this for its lookup-table-like behavior. However, adding noise to the text embeddings does not seem to have a large effect on the lookup-table-like behavior as the results largely stay the same. This suggests that the semantic power is what makes SBERT more effective and not its effectiveness for lookup-table-like behavior. The reason for this is that these slight perturbations in the embeddings should have affected the results more if the lookup-table-like behavior stems from the embeddings as it would modify the lookup-table key.

The generalizability and robustness of IG models are still far from desired as our results show. Even in the simplified world, they show clear signs of lookup-table behavior to produce the images, indicating poor generalizability. To improve the generalizability this lookup-table behavior of the IG models needs to be solved. Also, the spatial understanding of the models needs to be improved as they struggle with this aspect of the ShapeVQA data set. To further study IG, a metric like

the VQA consistency metric can be used to compare the semantic consistency of the models more directly than the FID and Inception score metrics.

5.3 Applications of the cyclic architectures

In the first chapter, some previous examples and applications of cyclic architectures were shown. They have a vast amount of applications such as explainability, increasing robustness against variation, data augmentation, and many others. In this thesis, their applications towards improving generalizability and robustness were explored through two experiments.

In the first experiment, a cyclic architecture was built to further improve the robustness to variance and the ability to generalize of the visual question answering (VQA) models. As shown in the results, the VQA model performance decreased after the cyclical training. Therefore it did not increase the generalizability on the test set. However, the training indicates that the VQA model is learning the variance of the synthesized images while simultaneously improving on the training data so we can say it is improving its robustness against the variance present in the synthesized images. The main variations produced by the image generation (IG) model were with regards to the size and location attributes. However, it showed poor performance on the semantic validity of the location attributes within these variations. This is reflected in the cyclical results for VQA with the performance decrease on the location type in Table 4.6. The size variation does seem to help the model as it improves on this type. The other types all show either a small increase or decrease. The reduced general performance of the cyclically trained VQA model can be attributed to the large performance decrease in the location type and the relatively low semantic consistency of the IG model synthesized images as indicated in the previous section. With a model that can produce semantically correct images more consistently, the performance gains could be more significant. Using the IG model that has been trained further by the VQA model to further train the VQA model as in cycle 2 does not perform better than the first cycle. So for the application of robustness in this experiment, a single cycle seems to be enough. As the model can learn the variance from the IG model we can say this cyclical approach can improve the robustness of the VQA model. However, the variance the model learns here is not improving the generalizability throughout all categories because the IG models do not show the required compositionality and semantic consistency.

The cyclical training of IG seems to show no improvements towards the quality of the images for both the first and second cycles. The results do show an improvement on the VQA metric in the second cycle, but the VQA model used for this metric has learned the variation of the IG model. It also does not show an increase in this metric throughout the training, thus it does not indicate improves semantic consistency. The qualitative review of the cyclical images shows neither an improvement nor a decrease in performance.

A reason that the model does not improve could have to do with the ineffectiveness of the loss functions for learning generation. The loss structure used in a GAN is quite different from using the auxiliary loss of the VQA model, so it is possible that this auxiliary loss does not function well to learn this task. It is also possible that the auxiliary loss should be used in combination with the GAN loss. However, this would best work during the training instead of afterward because the discriminator has to be trained in tandem with the generator. In such a setup, the VQA model serves as an auxiliary loss that keeps track of the semantic consistency, while the IG model learns to generate the images. This could be an interesting future work application of the cyclical idea.

Cyclic architectures show some interesting applications but suffer from some problems. For a cyclic architecture to be effective all of its components must be working optimally such that none of them end up being a bottleneck. In our study, we saw that the IG model could not consistently generate good examples for the VQA model, ultimately not improving its generalizability. Another problem is that setup up of such an architecture is complicated and takes a large amount of effort to make work. This in combination with the required effectiveness of the components makes it hard to research. Even in the simplified world of the ShapeVQA data set, we saw that it struggles. Running multiple cycles also did not seem to help. However, it still would be interesting to explore

running more cycles to see the behavior of the models that occurs. For example, it is possible the performance cycles upwards and downwards around a certain point. Even though the applications in this study are limited to improving robustness to variation, there are some more potential unexplored applications. If the components of the architecture function effectively, they can be used in a lifelong learning setup by continuously using new question-answer pairs to improve a VQA model during its long-term use. Such an application should be a long-term goal of cyclical research, as it could be a very effective way to keep models updated and continuously learning.

5.4 Future work

Throughout this chapter, some future work ideas have been presented. To add to these ideas some more are discussed in this section. An interesting opportunity the cyclic architecture provides is lifelong learning by continuously using new question-answer pairs on the image generation (IG) model. A web platform could be built to provide such a setup. In this platform, users could interact with the models by asking them to generate from their provided input, which they could provide feedback on. This feedback could then be used to improve both the IG model by positive reinforcement and the visual question answering (VQA) model by providing it with a new unseen data point to learn from. Furthermore, it could also be used to perform a qualitative study on the generalizability of a VQA model by asking users questions on random images from the test set where they indicate if they agree with the answer of the model or not. A study like this would be effective in exploring the generalizability required for real-world applications as the questions would truly be free-form and open-ended.

Both the VQA models and the IG models struggled with the location attribute due to their use of convolutional layers. An interesting way to solve this could be using a Recurrent-CNN [60] instead of the regular CNN. It keeps a recurrent relation between the features it extracts into the embedding, which could provide more spatial understanding to the VQA models. Such an architecture might be too complicated for IG as it has to generate instead of extract. For this task, a recent novel approach utilizing scene graphs could be used [22]. A scene graph contains a graph structure of objects and their relations, which represent a visual scene. It could improve the localization as the relations between the shapes would be more clearly defined in the generation. The main issue with this problem is the requirement that the input is a scene graph. This would require some kind of module that can transform a question-answer pair into a scene-graph which is not a trivial task. Still, this could be an interesting approach to explore and could have much future potential.

Another problem specific to the IG model was the lookup-table-like behavior. Something that might improve this is akin to the third qualitative review of IG. By applying noise to the text embeddings and possibly somewhere in the visual stage as well, it becomes more difficult for the model to learn in a lookup-table-like manner as the keys of the table would differ constantly during training. In turn, this should increase the diversity of the synthesized images. However, it might still struggle with compositionality as it requires a deeper understanding of the problem for this, which is not currently present in the models.

The final proposition for future work is concerning the evaluation of IG models. To perform IG research on complex multi-object data sets, MSCOCO [29] has often been the data set of choice. The most widely used VQA data sets VQA 1.0 [5] and VQA 2.0 [15] get their images from MSCOCO, annotating them with question-answers pairs. Because these data sets use the same images, the VQA consistency metric could be applied to IG models trained on MSCOCO. Any pre-trained VQA model on VQA 1.0 or VQA 2.0 could be used to do this. It could work as follows: The IG model generates an image from the caption. Then, the VQA model answers questions from the VQA data set on the MSCOCO image. The VQA answers are compared with the real answer as a semantic consistency metric. This metric would be model agnostic as any VQA model could be used with any IG model as long as they are trained on MSCOCO images. Applying such a method to measure semantic consistency would be a good idea for future and previous IG research to enable better comparisons between the IG models.

For the ShapeVQA data set specifically, it is also possible to build a more direct evaluation metric for the IG task. Such an evaluation metric would be based on the attributes used to generate the original ShapeVQA image. The synthesized image should display similar attributes to the shapes of the original image. How far it strays from this can be measured quantitatively and used as an IG metric that gives a much broader overview of the IG performance. It would require some kind of general detection of the synthesized shapes. Then, the type of shape has to be determined. The colors of the shapes can be matched by matching a histogram of the pixels of the shape with the pixels of the same shape with the correct color. The size could be measured by the number of pixels that should be present in the shape by the definition in the data set generation. The location could be measured by looking at the x,y point of the top-left of the shape. Measuring how well the calculated attributes match with the required attributes gives the metric. Such a metric could give a deeper insight into the abilities of the model to handle the different attributes similar to the question type accuracy for the VQA models. Therefore it could be an interesting addition to the ShapeVQA data set in future work

5.5 Conclusions

To conclude this thesis, a study was done on the generalizability and robustness of VQA, while also looking at the potential applications of a cyclic architecture towards this. Through the design of the ShapeVQA data set, this was analyzed on a set of VQA architectures. The Top attention (TA) visual question answering (VQA) architecture was able to generalize the best. It showed clear degrees of generalizability by being able to generalize the attribute combinations from training to testing, but still has room for improvement. For example, it abused language priors on the shape and count categories, which is undesirable for generalizability. Another takeaway from the results is that a reduced version of SBERT proved to be more effective. Additionally, The *TA* model showed that increased specificity level increased the difficulty ultimately reducing generalizability. In general, the model struggles most with spatial understanding for which a potential solution was proposed in the previous section.

Similar to VQA, the image generation (IG) models also struggled with producing images semantically consistent with the location attribute parts of the question-answer pairs. A possible solution for this was also proposed in the previous section. Looking at the results of IG more generally, it was clear that the IG model struggled to produce semantically consistent images for question-answer pairs that required multiple attributes to be consistent. The most likely reason presented for this was lookup-table-like behavior in the IG model, which ultimately meant low degrees of generalizability for the model. Another takeaway was the poor quality of the classical IG metrics in this research. The Inception Score and FID metrics did not match with the qualitative review due to the background of the images. Also, they failed to measure the semantic consistency which our proposed VQA consistency metric did manage to do.

The final part of the thesis was concerned with the applications of cyclic architectures combining VQA and IG towards improving their generalizability and robustness. It showed that the architecture was effective at making the VQA model learn the variations the IG model produced, ultimately making it more robust against variations. However, the IG model did not produce semantically consistent images enough for it to improve the performance. This is a large bottleneck and would need to be improved to make the cyclic architecture beneficial for generalizability. Improving the IG model output also did not seem to work with the cyclic architecture so a proposal was made to use the VQA model as an auxiliary loss instead during the training of the IG model. Some other potential applications were also presented such as a lifelong learning setup, with the possibility of using a web platform to continuously improve the models.

The final takeaway is that designing a simple purpose-built data set like ShapeVQA can be an effective tool to understand the models at a deeper level. It provided a reduced problem space which allowed us to go deeper into individual strengths and weaknesses of the model. Also, it enabled us to say something about the generality through its construction. Therefore, designing such a data set or reusing ShapeVQA could be useful in future research.

Bibliography

[1] Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. Analyzing the behavior of visual question answering models. *arXiv preprint arXiv:1606.07356*, 2016. 2

[2] Jon Almazan, Albert Gordo, Alicia Fornés, and Ernest Valveny. Handwritten word spotting with corrected attributes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1017–1024, 2013. 21, 32

[3] Soheyla Amirian, Khaled Rasheed, Thiab R Taha, and Hamid R Arabnia. A short review on image caption generation with deep learning. In *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*, pages 10–18. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2019. 1

[4] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086, 2018. 25, 34, 51

[5] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015. 1, 2, 23, 24, 33, 53, 58

[6] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017. 17

[7] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 19

[8] Silvio Barra, Carmen Bisogni, Maria De Marsico, and Stefano Ricciardi. Visual question answering: which investigated applications? *arXiv preprint arXiv:2103.02937*, 2021. 1

[9] Sebastian Bock and Martin Weiß. A proof of local convergence for the adam optimizer. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019. 12

[10] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015. 22

[11] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR, 2017. 25

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 22

[13] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011. 10

[14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 2, 15

[15] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 23, 51, 58

[16] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017. 17

[17] Hyungrok Ham, Tae Joon Jun, and Daeyoung Kim. Unbalanced gans: Pre-training the generator of generative adversarial network using variational autoencoder. *arXiv preprint arXiv:2002.02112*, 2020. 18

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 13, 19, 34, 38, 52

[19] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv preprint arXiv:1706.08500*, 2017. 3, 35

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 14

[21] He Huang, Philip S Yu, and Changhu Wang. An introduction to image synthesis with generative adversarial nets. *arXiv preprint arXiv:1803.04469*, 2018. 2

[22] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1219–1228, 2018. 58

[23] Kushal Kafle and Christopher Kanan. Answer-type prediction for visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4976–4984, 2016. 2

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 12, 35

[25] Diederik P Kingma and Max Welling. An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*, 2019. 17

[26] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2661–2671, 2019. 13

[27] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017. 23

[28] Jae Hyun Lim and Jong Chul Ye. Geometric gan. *arXiv preprint arXiv:1705.02894*, 2017. 27

[29] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2, 26, 34, 52, 54, 58

[30] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943. 7

[31] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR, 2018. 28

[32] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 2, 16

[33] Arun Mohan. Review on faster rcnn. https://medium.datadriveninvestor.com/a-review-on-faster-rcnn-72d31f50cc52. Accessed: 2010-09-30. 24

[34] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013. 12

[35] Tingting Qiao, Jing Zhang, Duanqing Xu, and Dacheng Tao. Mirrorgan: Learning text-to-image generation by redescription. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1505–1514, 2019. 3, 5

[36] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019. 22, 23, 32

[37] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015. 25

[38] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957. 7

[39] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985. 11

[40] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 13

[41] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016. 3, 35

[42] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019. 22, 32

[43] Meet Shah, Xinlei Chen, Marcus Rohrbach, and Devi Parikh. Cycle-consistency for robust visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6649–6658, 2019. 2, 3, 36, 51

[44] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 13, 24, 33, 38, 52

[45] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. *arXiv preprint arXiv:1705.07761*, 2017. 17

[46] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 13

[47] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 13, 35

[48] Ming Tao, Hao Tang, Songsong Wu, Nicu Sebe, Xiao-Yuan Jing, Fei Wu, and Bingkun Bao. Df-gan: Deep fusion generative adversarial networks for text-to-image synthesis. *arXiv preprint arXiv:2008.05865*, 2020. 2, 27, 35, 55

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. 18, 19, 25

[50] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 2, 26, 54

[51] Maciej Wiatrak, Stefano V Albrecht, and Andrew Nystrom. Stabilizing generative adversarial networks: A survey. *arXiv preprint arXiv:1910.00927*, 2019. 16

[52] Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017. 22

[53] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016. 22, 53

[54] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1316–1324, 2018. 2, 26

[55] Xiaojun Xu, Xinyun Chen, Chang Liu, Anna Rohrbach, Trevor Darrell, and Dawn Song. Fooling vision and language models despite localization and attention mechanism. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4951–4961, 2018. 2

[56] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 21–29, 2016. 25

[57] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5907–5915, 2017. 2, 26

[58] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1947–1962, 2018. 2, 26

[59] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 3

[60] Zhen Zuo, Bing Shuai, Gang Wang, Xiao Liu, Xingxing Wang, Bing Wang, and Yushi Chen. Convolutional recurrent neural networks: Learning spatial dependencies for image representation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 18–26, 2015. 58

Appendix A

Results

A.1 Visual question answering

A.1.1 Training

In the results chapter only the training graphs for *Top Attention* was shown. In this appendix the training graphs for the other VQA architectures in addition to the *language only* and *vision only* versions are shown. For the VQA architecture the training graphs of *Bottom + Top attention* is shown in Figure A.1, for *VQA-Pretrained* in Figure A.2 for *VQA-Pretrained* and for *VQA-CNN* in Figure A.3. The *language only* training graphs are shown in Figure A.4 and *vision only* in Figure A.5.

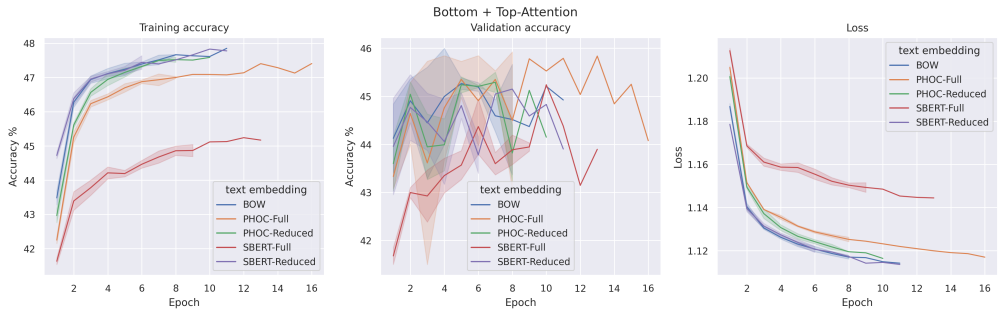


Figure A.1: From left to right: The training accuracy, validation accuracy and training loss of the *Bottom + Top attention* VQA model for each text embedding type.

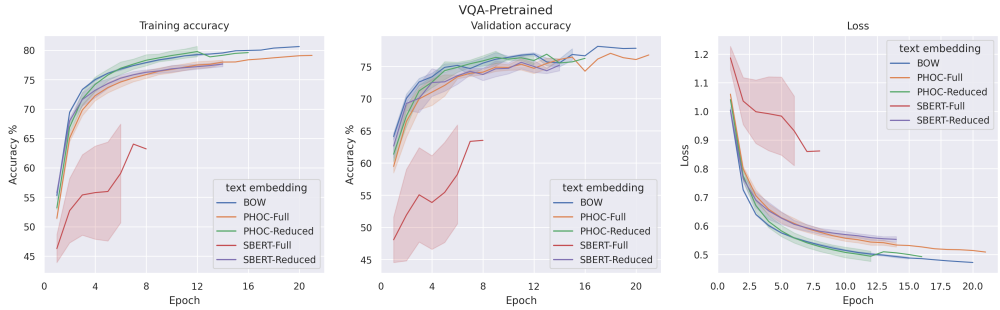


Figure A.2: From left to right: The training accuracy, validation accuracy and training loss of the *VQA-Pretrained* model for each text embedding type.

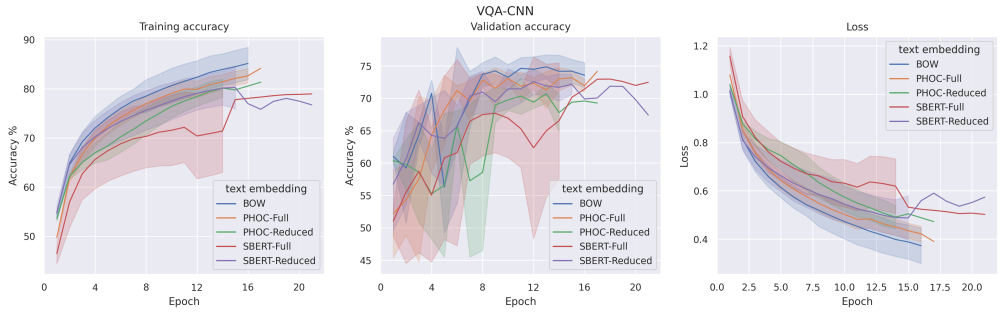


Figure A.3: From left to right: The training accuracy, validation accuracy and training loss of the *VQA-CNN* VQA model for each text embedding type.

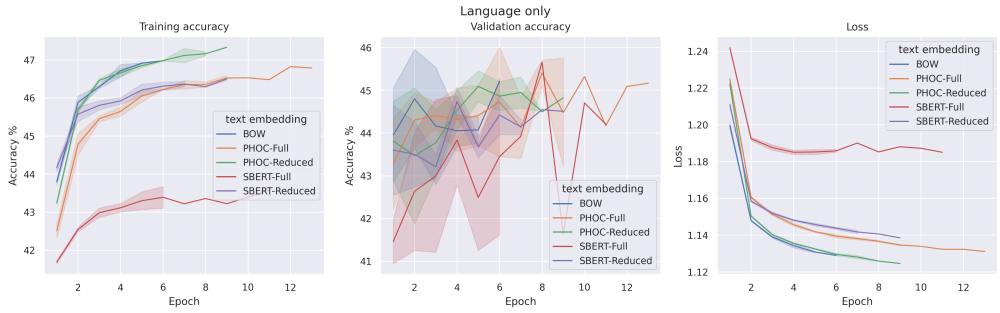


Figure A.4: From left to right: The training accuracy, validation accuracy and training loss of the *language only* VQA model for each text embedding type.

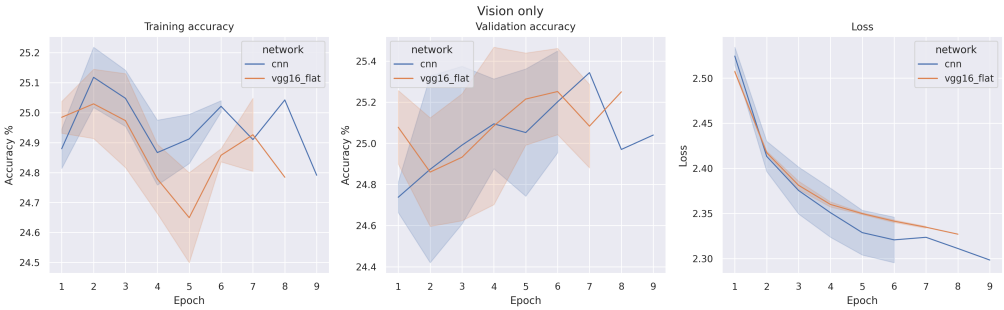


Figure A.5: From left to right: The training accuracy, validation accuracy and training loss of the *vision only* VQA model for each text embedding type.

A.2 Image generation

A.2.1 Training

In this appendix, some extra figures on the training of the image generation (IG) task are presented. In Figure A.6 an example of a mode collapse happening in the later stages of training is shown. For the Variational autoencoder (VAE) used in the pretraining, the loss was measured throughout the training. This is shown in Figure A.8. Additionally, some example images produced at the end of training are shown in Figure A.7. For the IG models, the accuracy of the discriminator on the real and fake data is also recorded in Figure A.9.

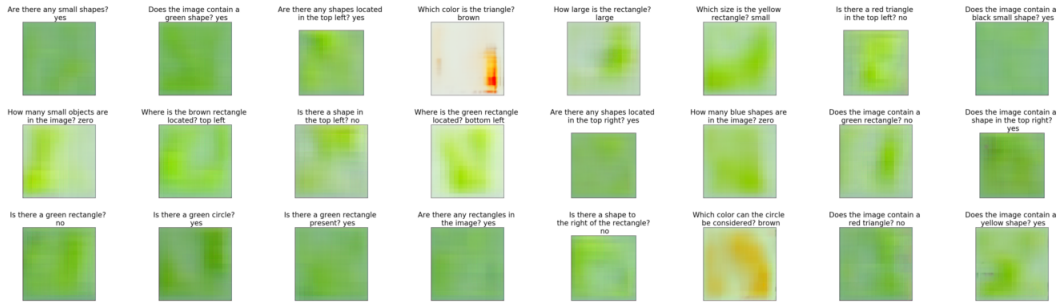


Figure A.6: Example of a mode collapse happening in a run of *Pretrained* SBERT-Reduced at the 350th epoch.

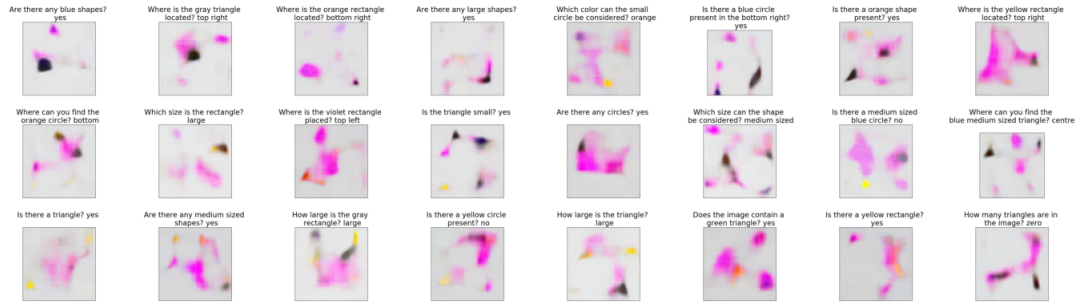


Figure A.7: Examples of the images produced by the Variational autoencoder at the end of training.

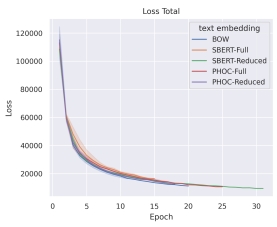
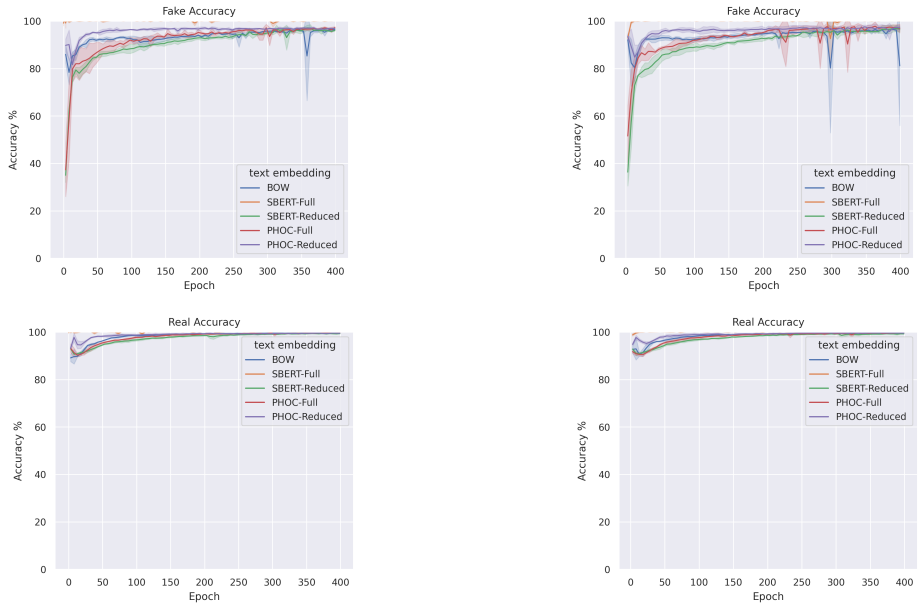


Figure A.8: Loss of the Varational autoenocder during training.



Pretrained Non-Pretrained

Figure A.9: The accuracy of the discriminator on the fake and real data for the VAE-Pretrained models and the Non-Pretrained models.

A.3 Cyclic

A.3.1 Visual question answering

In the cyclic training of the visual question answering model (VQA) the separate VQA and answer consistency losses were also recorded. These are seen in figure A.10 and Figure A.11 respectively. Additionally, the validation accuracy of the VQA model during cyclic training is shown in A.12.

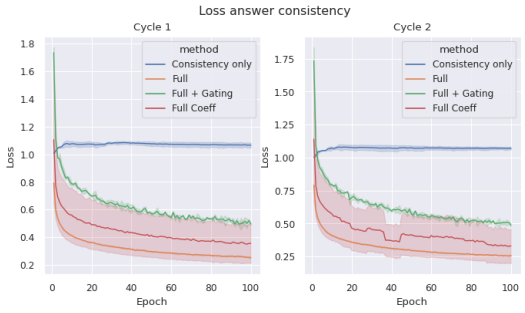


Figure A.10: Loss during the cyclic training of the answer consistency loss component.

A.3.2 Image generation

In the cyclic training of the image generation model the validation FID and Inception score metrics were also measured each epoch in addition to the VQA metric. These are shown in Figure A.13 and A.14 respectively.

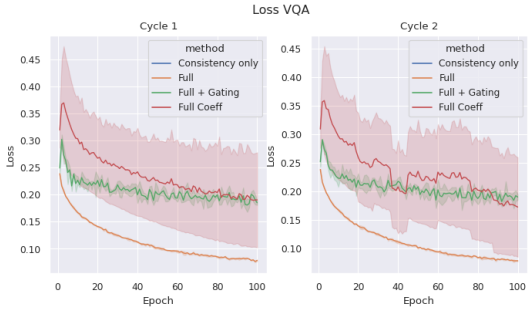


Figure A.11: Loss during the cyclic training of the VQA loss component.

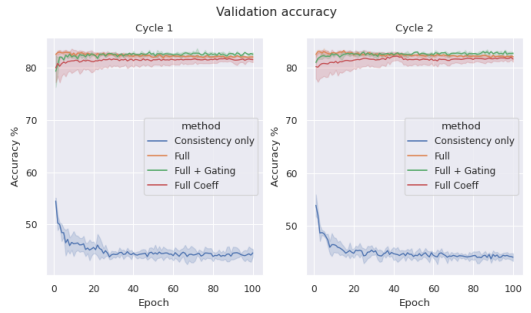


Figure A.12: Validation accuracy during the cyclic training of VQA.

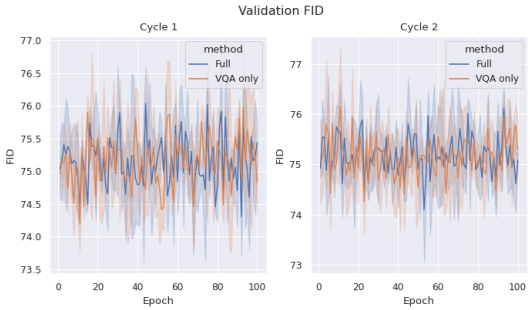


Figure A.13: Fréchet inception distance on the validation set during the cyclic training of IG.

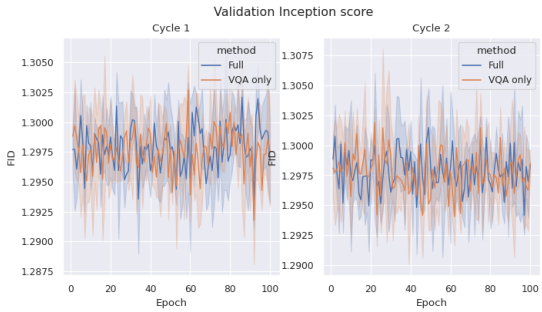


Figure A.14: Inception score on the validation set during the cyclic training of IG.