Reducing Training Time in Text Visual Question Answering

by

Ghazale Behboud

B.Sc., Islamic Azad University Karaj, 2015

A Thesis Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

Reducing Training Time in Text Visual Question Answering

by

Ghazale Behboud

B.Sc., Islamic Azad University Karaj, 2015

Supervisory Committee

Dr. T. Aaron Gulliver, Supervisor

(Department of Electrical and Computer Engineering)

Dr. Amirali Baniasadi, Departmental Member

(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. T. Aaron Gulliver, Supervisor

(Department of Electrical and Computer Engineering)

Dr. Amirali Baniasadi, Departmental Member

(Department of Electrical and Computer Engineering)

ABSTRACT

Artificial Intelligence (AI) and Computer Vision (CV) have brought the promise of many applications along with many challenges to solve. The majority of current AI research has been dedicated to single-modal data processing meaning they use only one modality such as visual recognition or text recognition. However, real-world challenges are often a combination of different modalities of data such as text, audio and images. This thesis focuses on solving the Visual Question Answering (VQA) problem which is a significant multi-modal challenge. VQA is defined as a computer vision system that when given a question about an image will answer based on an understanding of both the question and image. The goal is improving the training time of VQA models. In this thesis, Look, Read, Reason and Answer (LoRRA), which is a state-of-the-art architecture, is used as the base model. Then, Reduce Uni-modal Biases (RUBi) is applied to this model to reduce the importance of uni-modal biases in training. Finally, an early stopping strategy is employed to stop the training process once the model accuracy has converged to prevent the model from

overfitting. Numerical results are presented which show that training LoRRA with RUBi and early stopping can converge in less than 5 hours. The impact of batch size, learning rate and warm up hyper parameters is also investigated and experimental results are presented.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Dr. T. Aaron Gulliver, for mentoring, support, encouragement, and patience.

DEDICATION

I dedicate this work to my family. My parents, who I wouldn't be where I am if I didn't have their support. My better half, Moe, who's always been there for me.

Chapter 1

Introduction

Visual Question Answering (VQA) is the task of answering questions about an image. VQA systems can be used to help the visually impaired by acting as their eyes or add more functionality to future home assistant robots. In recent years, VQA has attracted tremendous research attention. This multi-modal problem is a combination of Computer Vision (CV) and Natural Language Processing (NLP) problems that can be solved using Deep Learning (DL) methods. The success of DL algorithms comes from their ability to learn patterns in input data and their accuracy on large amounts of training data. Both of these qualities are useful in solving VQA problems. The focus of this thesis is reading text in an image and answering questions about it. It has been shown that 21% of questions that sight-impaired users ask are related to text in an image [5]. However, most VQA models fail at answering these types of questions. Consider the question in Figure 1.1. The model first needs to understand the question, then find the traffic light and the sign next to it, and then find the answer to "what does the sign say?", by extracting the text that is written on the sign [5]. Most recent models are not capable of such complex reasoning. In this work, the Look, Read, Reason and Answer (LoRRA) model is used as the base model to

answer these types of questions. Training this complex network requires a large and comprehensive dataset that takes considerable time and resources. The goal of this study is to reduce the training time of the LoRRA model while limiting the effect on the accuracy.

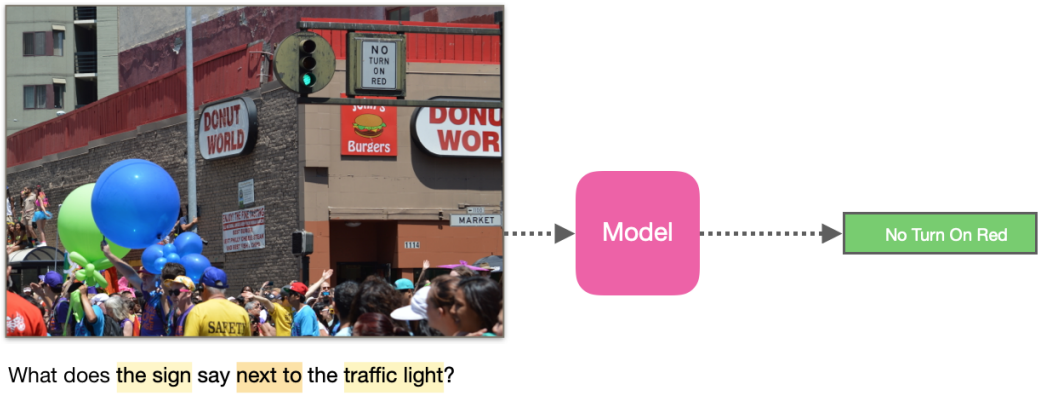


Figure 1.1: An example of a TextVQA task [3].

1.1 Research Objectives

In this thesis, practical strategies to reduce the training time of the LoRRA model are investigated. To avoid overfitting, an early stopping strategy is implemented. To lower the impact of biased training examples, the RUBi strategy is applied to the model. To study the impact of the learning rate and warm up hyper parameters, multiple experiments are conducted. The TextVQA model is implemented using the Pythia [21] framework which is an open source code base created by the Facebook AI team to bootstrap vision and language research. There are several VQA datasets [12][13][15] which are publicly available. The TextVQA [22] dataset is used in this thesis to train and test the LoRRA model. Experimental results are obtained and compared with previous results.

1.2 Thesis Organization

The rest of this thesis is organized as follows.

Chapter 2 gives the background on artificial intelligence, neural networks, deep learning, and attention models.

Chapter 3 explains visual question answering by reading text in images, then presents the LoRRA model and its performance evaluation.

Chapter 4 investigates practical strategies to reduce the training time and the effect of batch size, learning rate, and warm up hyper parameters on the performance of the LoRRA model. Then the experimental results are presented.

Chapter 5 provides some conclusions. It also presents areas for further research related to this work.

Chapter 2

Background

2.1 Overview

VQA is a multi-discipline Artificial Intelligence (AI) research problem that is attracting considerable interest due to the joint comprehension requirements of images and text. Data scientists have found Deep Neural Networks (DNNs) to be a promising approach to solving these sophisticated problems. This chapter provides a review of AI concepts in general as well as deep learning methods in particular, to provide the basics needed for the rest of the thesis.

2.2 Artificial Intelligence and Machine Learning

Interest in the concept of machines that can learn on their own has exploded over the past decades. AI denotes a machine that can think and behave like humans. Humans have changed the world tremendously due to their cognitive abilities. Using their exceptional learning abilities, they have invented technical devices, social behaviors, laws, institutions and complicated symbol systems such as mathematics. Thus, learning is an important trait that can be employed by machines. The process of using

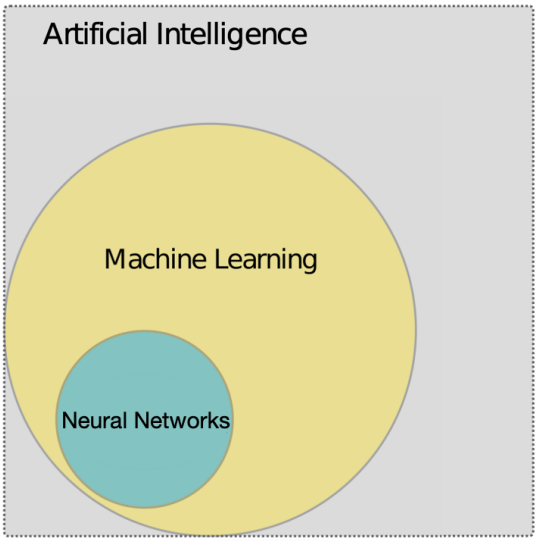


Figure 2.1: Relationship between artificial intelligence, machine learning, and neural networks [18].

algorithms to give machines the ability to learn is called Machine Learning (ML). The term learning here means obtaining a structural description from raw data. This structural description is called a model and it can have many forms like decision trees, linear regression, or neural network weights. Figure 2.1 illustrates the relationship between AI, ML, and neural networks.

2.3 Neural Networks

Neural Networks (NNs) are computational models that are based on the nervous system of animals. The architecture of NNs contains three components: neurons, layers, and connections between layers. Figure 2.2 shows the topology of a feed forward multi-layer neural network. This network has an input layer, one or more hidden layers, and an output layer. Each layer has one or more neurons and is fully connected to the adjacent layers. Each connection has a parameter called weight. The weights represent long-term information storage in the network. Neural networks

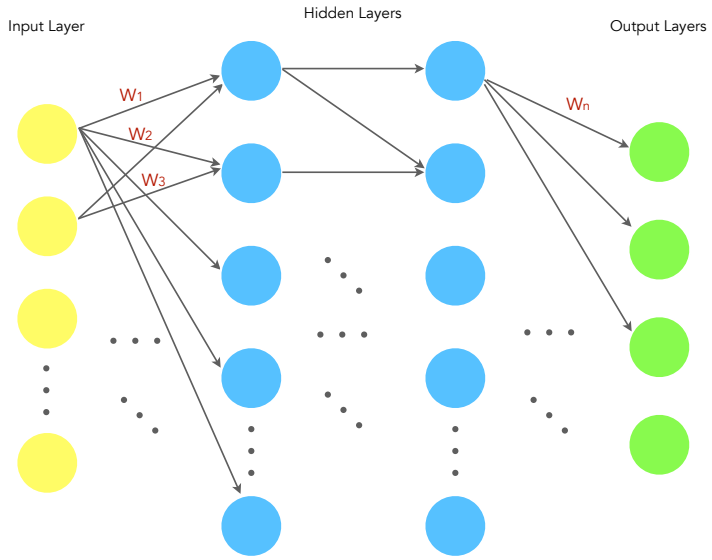


Figure 2.2: A multi layer feed forward neural network [18].

learn new information by updating these weights. In fact, training a NN is the process of adjusting the weights in order to improve prediction.

2.3.1 Back Propagation

This section provides an overview of the training process. The first step of training is the feed forward transmission of data from the input nodes through the hidden nodes to the output nodes. Since the task in this thesis is supervised learning and labeled data is available, the output values are compared with the expected ones and the loss function generates the error. This error is then fed back through the layers and used to adjust the weights to obtain more accurate predictions. After repeating this process a sufficient number of times, the error rate will fall below a certain threshold at which point the network is trained based on the given information.

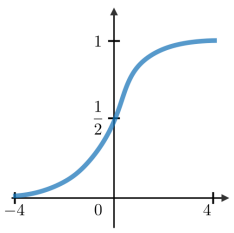
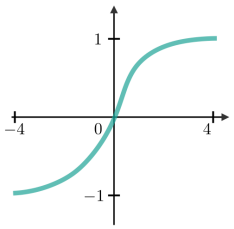
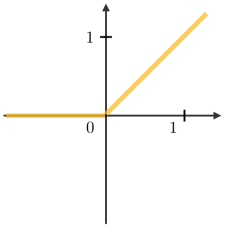
Name	Plot	Equation	Derivative
softmax		$f(x) = \frac{1}{1+e^{-x}}$	$f(x)' = f(x)(1 - f(x))$
tanh		$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	$f(x)' = 1 - f(x)^2$
ReLU		$f(x) = \begin{cases} 0 & x \leqslant 0 \\ x & x > 0 \end{cases}$	$f(x)' = \begin{cases} 0 & x \leqslant 0 \\ 1 & x > 0 \end{cases}$

Table 2.1: The softmax, tanh, and ReLU activation functions [2].

2.3.2 Activation Functions

Activation functions are used in NNs to convert the output of one layer into a form useful for the next layer as its input. Activation functions are scalar-to-scalar functions that add non-linearity to the network. Table 2.1 gives some of the most frequently used activation functions. ReLU is employed here because of its simple, linear and non-saturating form. This simplicity reduces the computational complexity and the linearity prevents the vanishing gradient problem [18].

ReLU

ReLU is an activation function that add non-linearity to NNs. This will change all the negative values to 0 and keep the positive values.

2.3.3 Loss Functions

Loss is the error in the network prediction. These errors are aggregated over the entire dataset and then averaged to obtain the total loss. This value shows how close the NN is to its ideal [18]. Loss functions calculate the difference between the predicted and ideal network outputs. The mean squared error loss is given by

$$L = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2, \tag{2.1}$$

where N is the number of training samples, $\hat{Y}_i$ is the target output, and Y_i is the actual output of the network.

2.3.4 Stochastic Gradient Descent (SGD)

One of the most important methods to solve optimization problems is Gradient Descent (GD), which was originally proposed in [16]. GD starts with an initial vector w_0 and updates the weights using partial derivative of the cost function $\frac{\partial L}{\partial w_k}$ and ratio α_k in each step

$$w_{k+1} = w_k - \alpha_k \frac{\partial L}{\partial w_k}, \qquad k = 0, ..., \tag{2.2}$$

where α_k is learning rate.

2.3.5 Hyper Parameters

Hyper parameters in ML are parameters that can be tuned to make a network work better and faster. Tuning here means finding the best values so that the model neither underfits nor overfits the training dataset, while learning quickly. In an underfit model, the prediction accuracy is very low. Overfitting occurs when models achieve high training accuracy by learning unwanted correlations in the training dataset. In

other words, an overfit model fits the training set too closely. The hyper parameters for ML models include learning rate, regularization, momentum and sparsity [18].

Learning Rate

Learning rate is an important hyper parameter used in training a neural network. The learning rate is how fast or slow the weights of the network are adjusted. Choosing a suitable learning rate is crucial because if the learning rate is too small, it may take many iterations to converge to the best weights while if the learning rate is too high, the best weights may not be obtained.

2.4 Deep Learning

Deep Learning (DL) is the next-generation of neural networks with more neurons, more complex ways of connecting layers, and more computational power. This allows the network to better extract features. Several DL network architectures have been developed. This thesis considers Convolutional Neural Networks (CNNs) and Recurrent Neural Network (RNNs).

2.5 Convolutional Neural Networks (CNNs)

CNNs are used when there are repeating patterns or spatial relations in the input data. In other words, CNNs uses convolution to extract higher-order features from the data. Because of this, they are well suited for analysing image and audio data. A high level general CNN architecture is given in Figure 2.3. This shows that there are three layer groups: input layer, feature extraction layers and classification layers.

2.5.1 Feature Extraction Layers

Convolution

Convolutional layers are the most influential part of CNNs. A convolutional layer consists of a set of learnable weights which are referred to as the filter. During the forward pass, the inner product of the filter and the input is computed at each position. Figure 2.4 illustrates how the filter moves across the input data to generate the result which is called the activation map. The activation maps for each filter are stacked together along the depth dimension to construct the output volume.

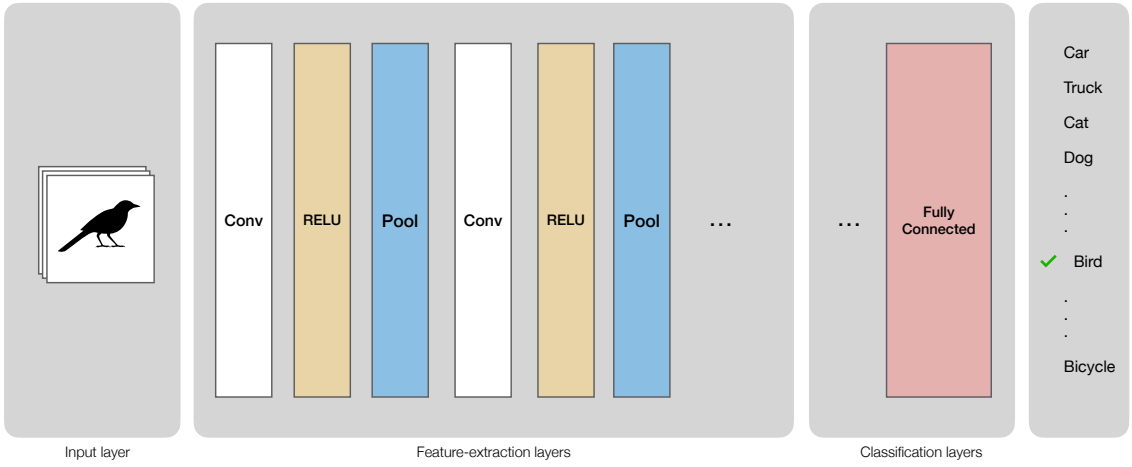


Figure 2.3: High level general CNN architecture [18].

Pooling

Pooling layers reduce the spatial size of the data. Reducing the number of dimensions progressively over the network prevents overfitting. There are several types of pooling such as mean pooling, max pooling, and sum pooling. The most common pooling operation is max pooling [18]. Down sampling using max pooling is depicted in Figure 2.5.

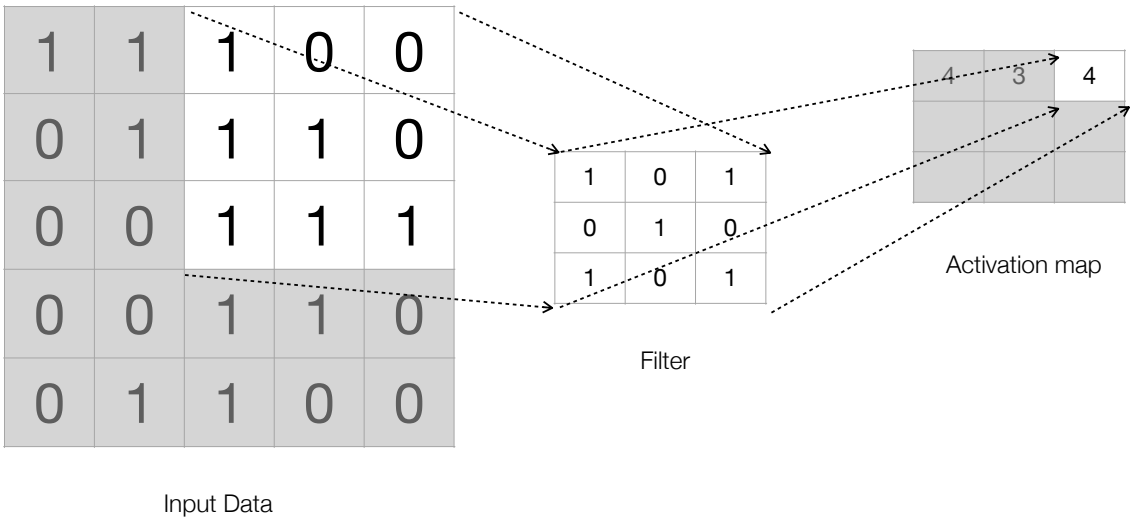


Figure 2.4: The convolution operation [18].

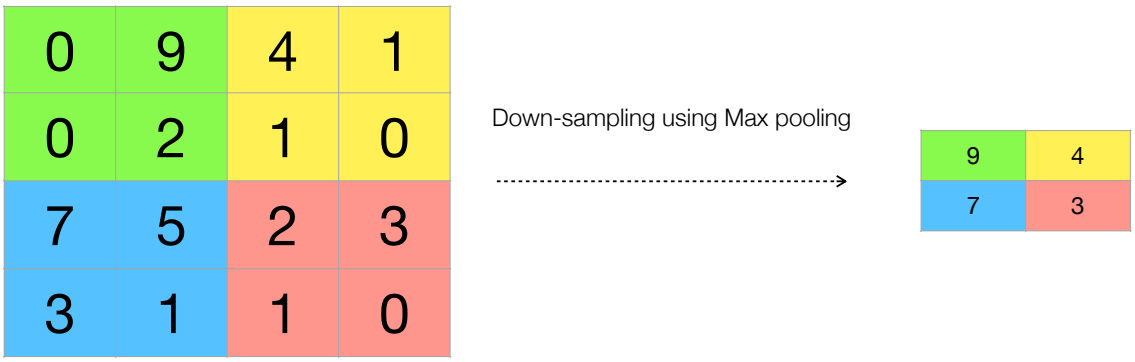


Figure 2.5: Down sampling using max pooling [18].

2.5.2 Fully Connected Layer

A fully connected layer in a neural network is used to predict the most probable network answer. This layer is the last block in the architecture and is a classifier to predict the model output.

2.6 Recurrent Neural Networks (RNNs)

An RNN is a feed forward NN that has been designed to process a sequence of values over time. For example, as you read this thesis you understand each word based on what was read previously. Thus, thoughts are persistent as is the case with RNNs. Because of this characteristic, RNNs are effective in analyzing language, audio, and text. Figure 2.6 shows how an RNN operates on a sequences of inputs. At each time step, the output is computed from the hidden nodes in the previous state of the network. Figure 2.7 demonstrates a recurrent edge at time t where the outputs are expressed as

$$a_t = g_1(W_{aa}a_{t-1} + W_{ax}x_t + b_a), \tag{2.3}$$

$$y_t = g_2(W_{ya}a_t + b_y), \tag{2.4}$$

where W_{ya},W_{aa},W_{ax},b_y and b_a are coefficients, and g_1 and g_2 are activation functions.

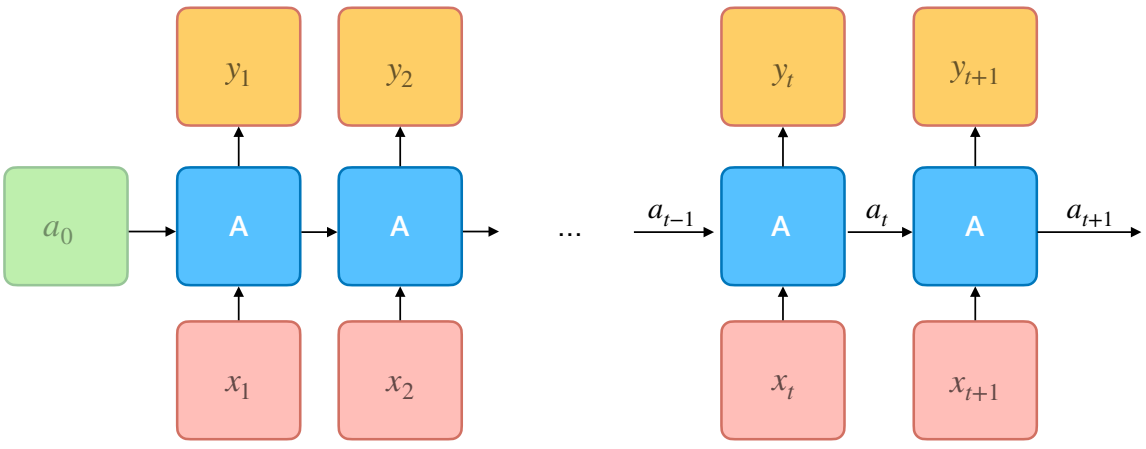


Figure 2.6: A high level general RNN architecture [2].

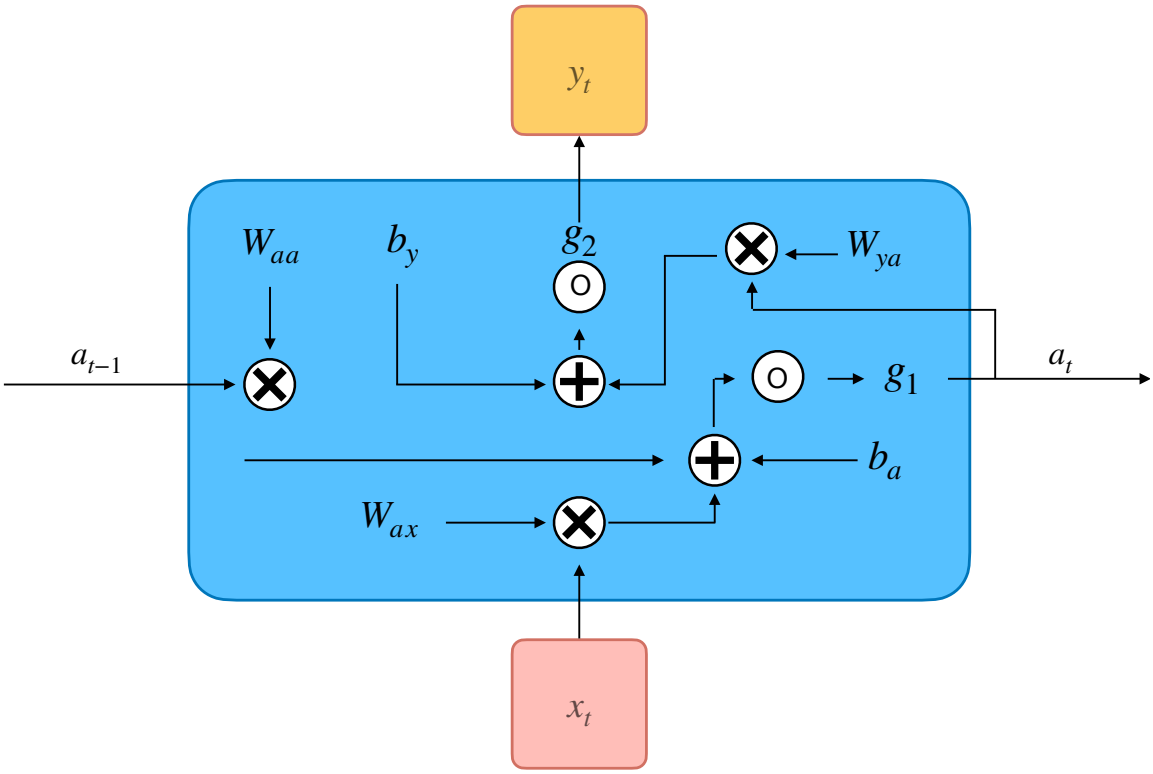


Figure 2.7: A recurrent edge at time t [17].

2.6.1 Vanishing and Exploding Gradients

RNNs are known to have vanishing and exploding gradient issues. These occur during network training when the gradient is exponentially decreasing or increasing with respect to the number of layers. Gradient clipping can be used to solve the exploding gradient problem. With this method, the gradients are clipped at a defined threshold. Long Short-Term Memory (LSTM) units can be used to solve the vanishing gradient problem.

2.7 Long Short Term Memory (LSTM)

LSTMs are a type of RNN which can learn long-term dependencies. The main differences between LSTMs and other types of RNNs are the memory cells and the ability

to change the information in these cells using structures called gates. Due to this gating structure, these models can overcome the vanishing gradient problem. The structure of LSTMs is explained below [17].

2.7.1 Forget Gate

A forget gate decides what information is discarded from the memory cell. Figure 2.8 shows this gate as a sigmoid function of y_{t-1} and x_t. The output is a number between 0 and 1. The forget gate function is

$$f_t = \sigma(W_f[y_{t-1}, x_t] + b_f), \tag{2.5}$$

where W_f and b_f are coefficients.

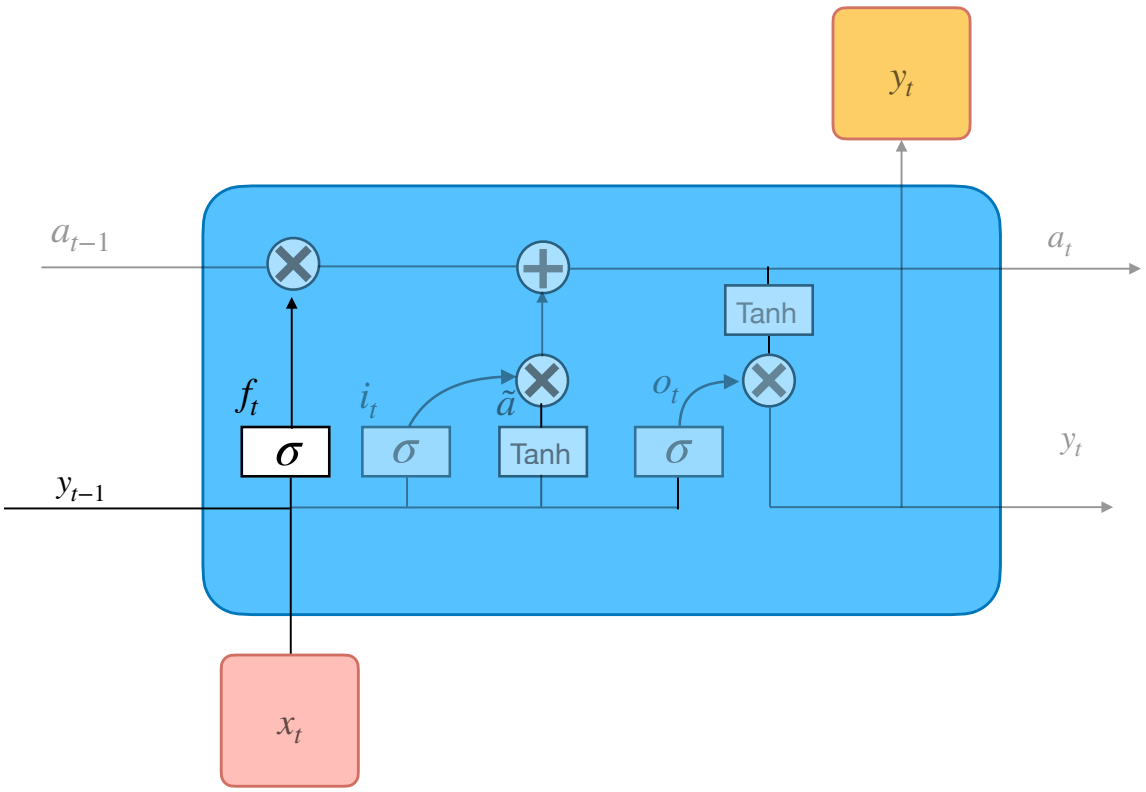


Figure 2.8: The forget gate structure [17].

2.7.2 Input Gate

An input gate decides which information is stored in the memory cell. This gate contains two parts. The first part is a sigmoid function that determines which values will be updated and the second part is a tanh function which determines new values. The input gate is defined as

$$i_t = \sigma(W_i[y_{t-1}, x_t] + b_i), \tag{2.6}$$

$$\tilde{a} = \tanh(W_a[y_{t-1}, x_t] + b_a). \tag{2.7}$$

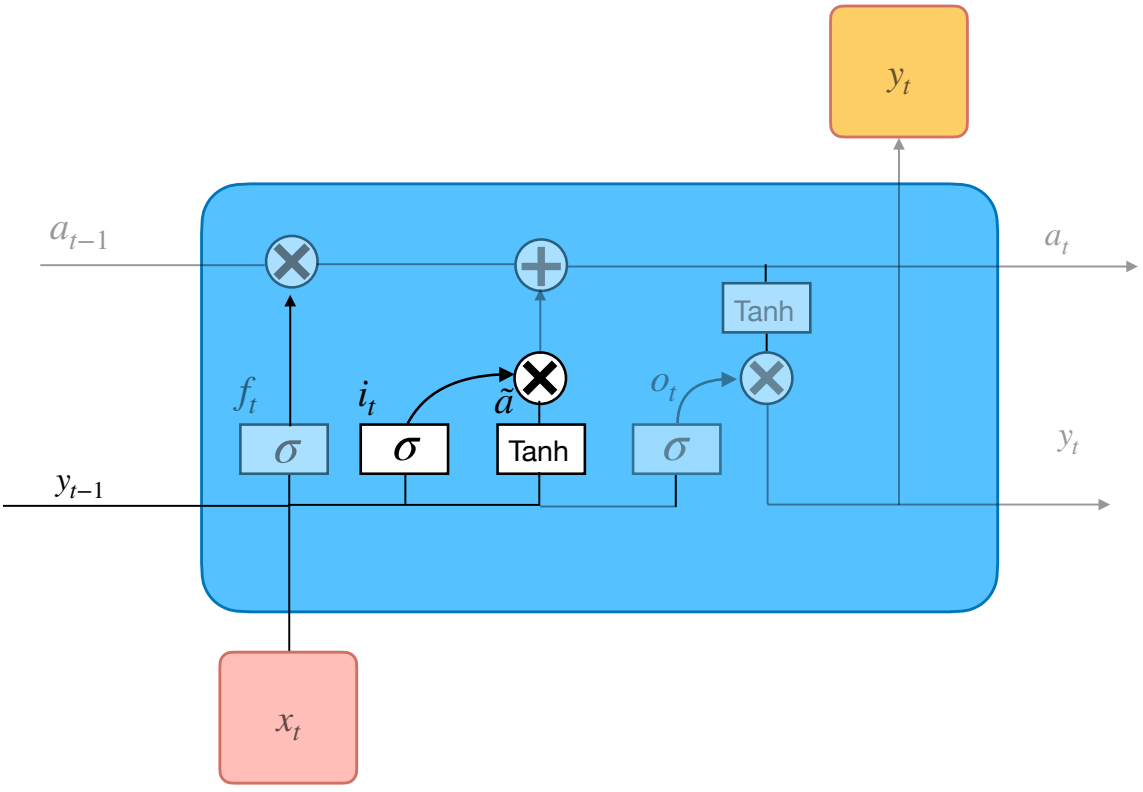


Figure 2.9: The input gate structure [17].

2.7.3 Update Cell State

The new cell state is obtained by multiplying the old state by f_t and scaling it by the state update value

$$a_t = f_t * a_{t-1} + i_t * \tilde{a}, \tag{2.8}$$

where $*$ is element wise multiplication.

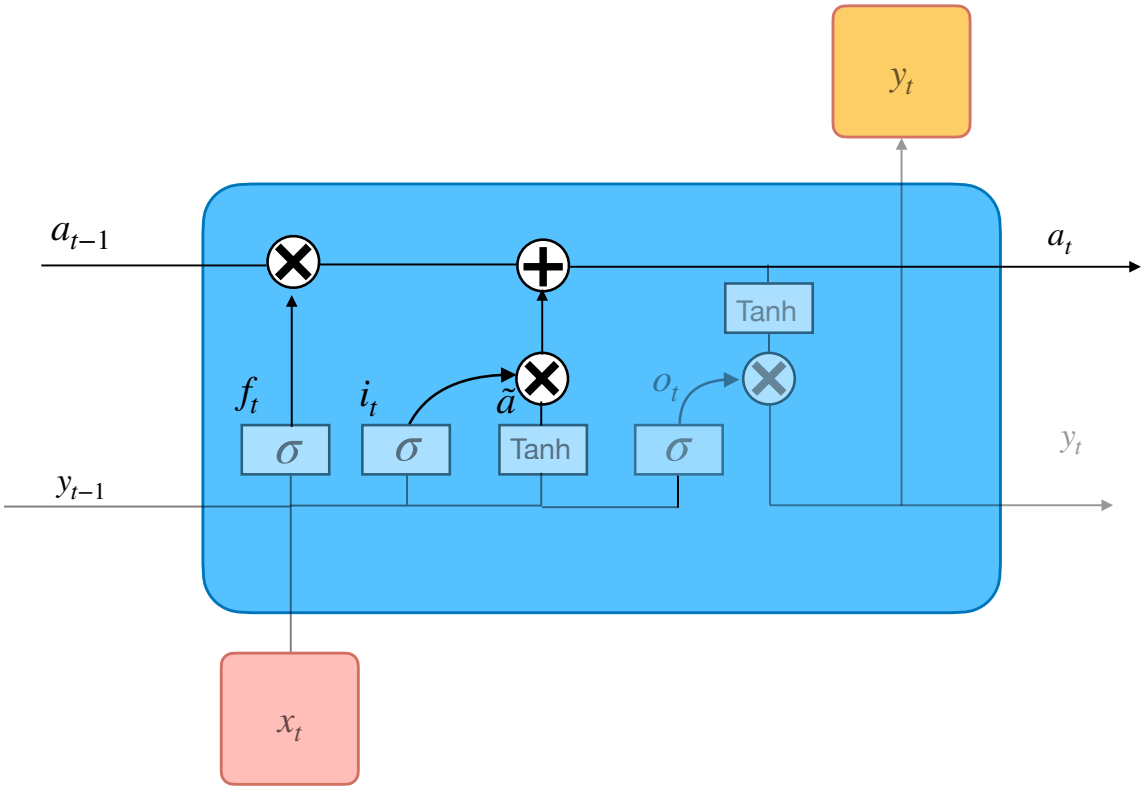


Figure 2.10: The update cell state structure [17].

2.7.4 Output Gate

The output gate is responsible for calculating the output of a block. This gate has two parts. A sigmoid function chooses how much of the cell state to output and a tanh function restricts the cell state value between -1 and 1. This is multiplied by

the sigmoid fucntion output

$$o_t = \sigma(W_o.[y_{t-1}, x_t] + b_o), \tag{2.9}$$

$$y_t = o_t * \tanh(a_t). \tag{2.10}$$

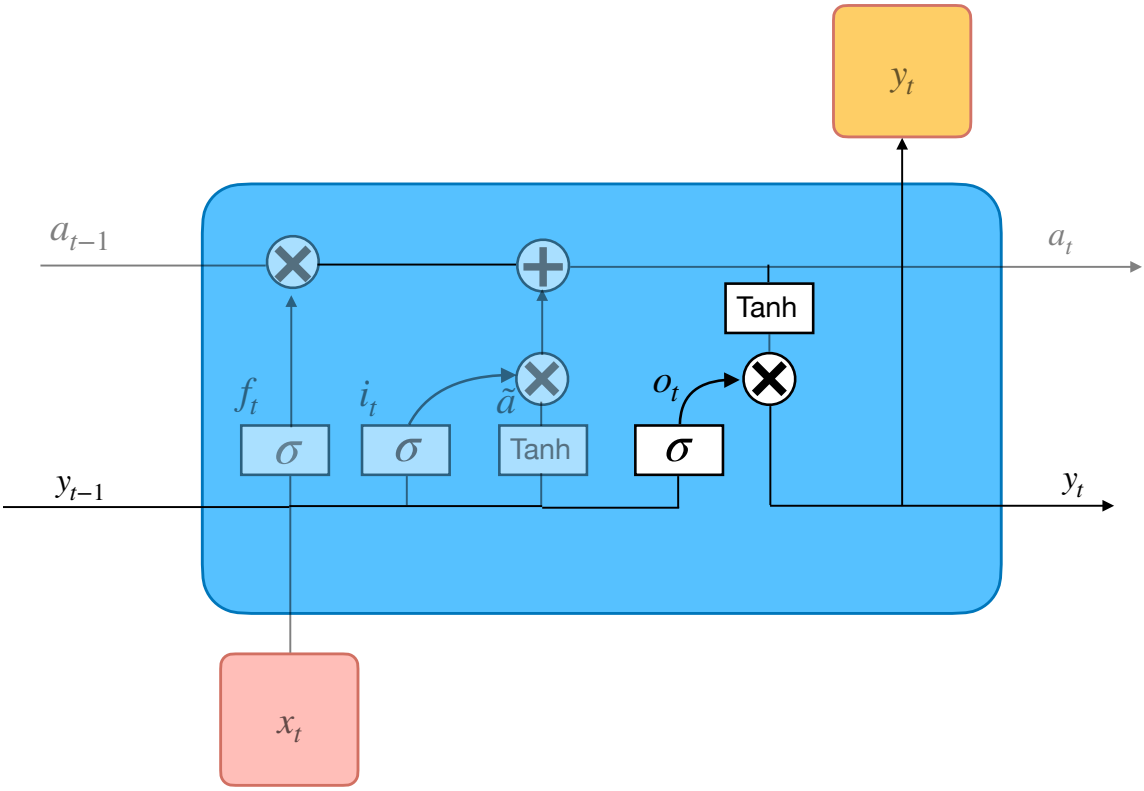


Figure 2.11: The output gate structure [17].

According to the above expressions, the input and forget gates control the memory cell contents. Thus, if both gates are closed, the cell state will remain unchanged. This capability solves the vanishing gradient problem because the information is preserved so the gradients are retained over multiple time steps.

2.8 Attention Model

The word attention by definition means the act or state of applying the mind to something. In other words, attention is a selective focus on a certain part of the information while ignoring the rest [18]. An NN can behave in this way during training using an attention mechanism. An attention model is a system that takes two sequences as inputs and returns a context vector c as the output. The weights of the output are chosen based on how relevant the sequences are to each other. First, the network computes e_{ij} which is an alignment function that determines how well the inputs around position j and the input at position i match

$$e_{ij} = a(y_{i-1}, x_j) \tag{2.11}$$

and then

$$a_{ij} = \frac{e^{e_{ij}}}{\sum_{k=1}^{T_x} e^{e_{ik}}} = \text{softmax}(e_{ij}). \tag{2.12}$$

The softmax function behaves like an argmax function

$$\text{argmax}(z_1, ..., z_n) = (0, ..., 0, 1, 0, ..., 0), \tag{2.13}$$

where $z_i = 1$ is the unique maximum of $(z_1, ..., z_n)$. The output of the attention block is

$$c_i = \sum_{j=1}^{T_x} a_{ij} x_i. \tag{2.14}$$

Chapter 3

Visual Question Answering by Reading Text in Images

3.1 Overview

The main focus of this thesis is improving the ability of VQA models to read text in images and answer questions about this text. As mentioned in Chapter 1, despite the fact that questions about text in images are commonly asked by visually impaired people [13], current VQA models usually fail to answer them. In this chapter, the model architecture [22] used as a basis for this research is introduced in detail followed by the performance evaluation.

3.2 TextVQA Dataset

DL or ML models need a suitable dataset to learn and work properly. The TextVQA dataset is publicly available at https://textvqa.org and is employed here. This dataset contains 28,408 images and 45,336 questions about text in the images.

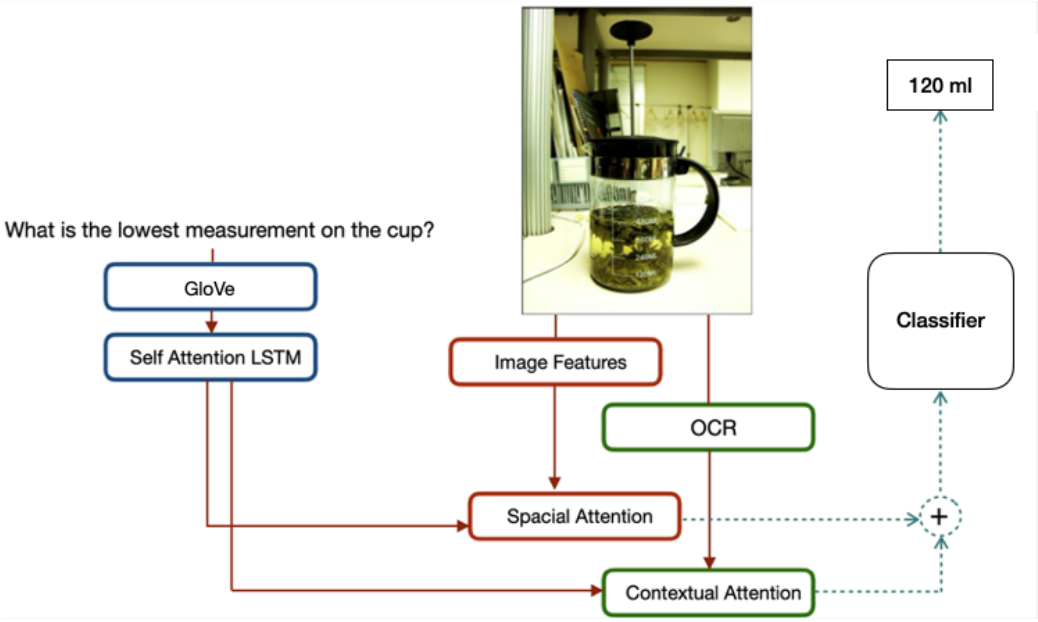


Figure 3.1: The Look, Read, Reason and Answer (LoRRA) model [22].

3.3 Look, Read, Reason and Answer (LoRRA)

The model proposed in this thesis employs LoRRA which is a state-of-the-art architecture for TextVQA [3]. Figure 3.1 shows that this model contains the following three components.

- Feature extraction which extracts the visual and text features from the images and questions.
- Reading which extracts words that are overlaid or embedded in the images.
- Answer module that predicts answers based on the results from previous components.

3.3.1 Feature Extraction Component

Feature extraction is given by

$$f_{feats}(v, q) = f_{comp}(f_A(f_I(v), f_Q(q)), f_Q(q)), \tag{3.1}$$

where $f_Q(q)$ represents the text features, $f_I(v)$ refers to the visual features, and f_A predicts the attention based on $f_I(v)$ as the input and $f_Q(q)$ as the context. This component provides a combination of text and visual features as shown in Figure 3.1.

Global Vectors for Word Representation (GloVe)

Global Vectors for Word Representation (GloVe) [19], is an unsupervised pre-trained model that provides vector representations for words. In LoRRA, this model is used as a word embedding module that takes question words as input and provides a vector representation for each word.

LSTM Self Attention

In LoRRA, the question embedding model consists of two parts. The first is LSTM and the second is a self-attention mechanism that allows the question to extract relevant aspects from itself. Suppose we have a question which has n words represented by a sequence of word embeddings

$$Q = (w_1, w_2, ..., w_n), \tag{3.2}$$

where w_i is the word embedding for the ith word in the question. Then these words are passed to the LSTM model to generate the hidden state

$$H = (h_1, h_2, ..., h_n). \tag{3.3}$$

Then, the attention vector is

$$a = \text{softmax}(w_{s2} \tanh\big(W_{s1} H^T\big)), \tag{3.4}$$

where W_{s1} and w_{s2} are weight matrices and vectors, respectively, which are obtained during training. The question vector is

$$m = Ha^T. \tag{3.5}$$

The attention vector focuses on a specific part of a question to find the relation between a set of words. However, multiple semantic relationships can exist in one question, so several attention vectors are obtained to capture the overall semantics of the question. Vector a becomes matrix A by combining all attention vectors

$$A = \text{softmax}(W_{s2} \tanh\big(W_{s1} H^T\big)). \tag{3.6}$$

Then the final question embedding is

$$f_Q(q) = HA. \tag{3.7}$$

Spatial Attention

The RestNet CNN architecture is used to extract visual features which are represented by $f_I(v)$. Spatial attention is used to focus on the parts of the image that are mentioned in the question. As mentioned in Chapter 2, the attention model has $f_I(v)$ and $f_Q(q)$ as inputs and the output is the weighted sum of the visual features

$$f_A(f_I(v), f_Q(q)) = \sum_{i=1}^{n} a_{ij} f_I(v), \tag{3.8}$$

where

$$a_{ij} = \text{softmax}(e_{ij}), \tag{3.9}$$

and e_{ij} is an alignment model which determines how well the image features around position j align with the question feature at position i [8].

3.3.2 Reading Component

In the reading component, Optical Character Recognition (OCR) [6] is used to extract M words that are overlaid or embedded in the image

$$s = s_1, s_2, ..., s_M. \tag{3.10}$$

These extracted words are given to a pre-trained word embedding model to generate f_O. Then the related extracted words are emphasized using the attention model

$$f_A(f_O, f_Q(q)) = \sum_{j=1}^{M} a_{ij} f_O, \tag{3.11}$$

and the output is

$$f_{read}(s, q) = f_{comp}(f_A(f_O, f_Q(q)), f_Q(q)). \tag{3.12}$$

3.3.3 Answering Module

The final unit in the LoRRA architecture is the answering module. Using the output of feature extraction and reading components, this module predicts the most probable answer over a fixed-size answer space. This answer space contains N answers and M OCR tokens. In fact, the answering module is a Multi-Layer Perceptron (MLP) feed-forward neural network which is shown as a classifier in Figure 3.1. The final answer

probabilities are

$$f_{LoRRA}(v, s, q) = f_{MLP}(f_{feats}(v, q); f_{read}(s, q)). \tag{3.13}$$

3.4 Training Model and Performance Evaluation

3.4.1 Text-VQA Evaluation Metric

The publicly released TextVQA evaluation script is used to calculate the model accuracy [1]. Each image in the dataset is paired with 3 questions written by crowd-sourced annotators. The evaluation metric uses 10 ground truth answers for each question to compute the model accuracy. Given an image and a question, the goal of a VQA system is to generate an answer that matches the answers provided. If c is the number of unique ground truth answers that match the answer provided by the system, then the evaluation metric is

$$Acc = \min(1, c/3). \tag{3.14}$$

If the system provides an answer that matches at least 3 unique ground truth answers, it gets the maximum score of 1. If the system generates an answer that does not match any ground truth answers, it gets a score of 0. In other cases, it gets a fractional score.

3.4.2 Training and Validation Results

In this thesis, the LoRRA model is implemented using the Pythia [21] framework which is an open source code base created by the Facebook AI team to bootstrap vision and language research. There are several VQA datasets [12][13][15] which are publicly available. The TextVQA dataset [22] is used in this study to train and validate the LoRRA model. This dataset is divided into training and validation

sets with sizes 34602 and 5000, respectively. Figure 3.2 shows the training accuracy of the LoRRA model versus computation time. Table 3.1 gives the LoRRA model performance on the validation dataset in comparison with the result from [22] along with the training computation time and number of iterations. The computation time to achieve a validation accuracy of 0.2218 is 23.24 h. Note that the Compute Canada server does not allow jobs to run longer than 24 h. Figure 3.2 shows that the training accuracy of the model stops improving after a certain point.

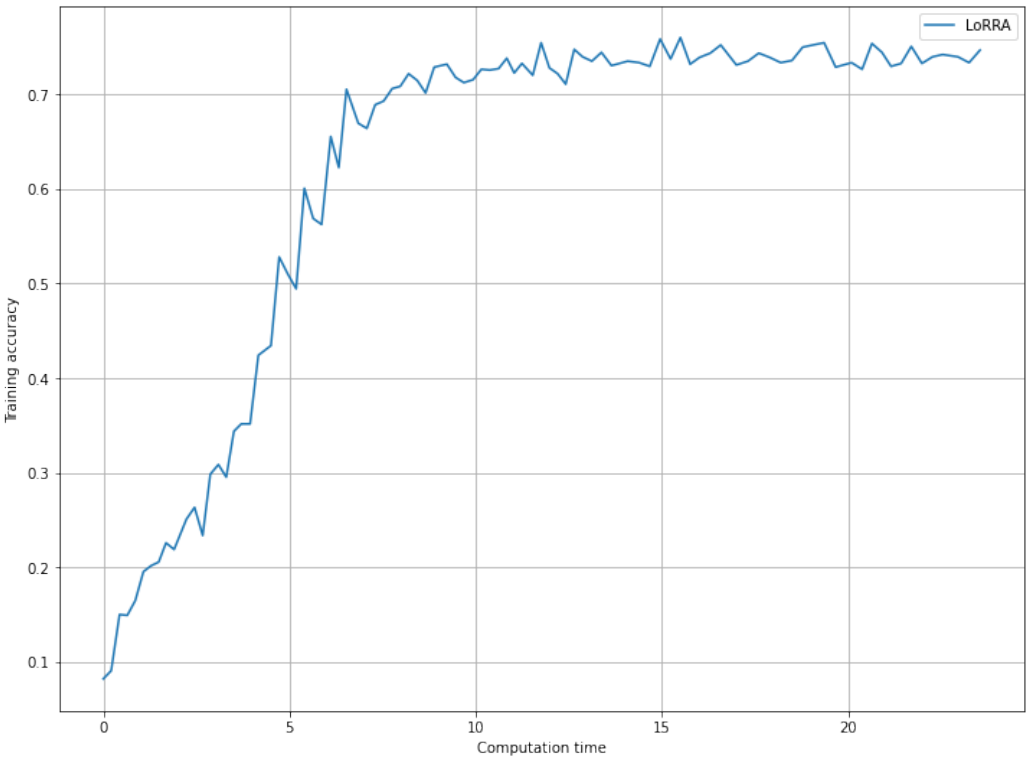


Figure 3.2: Training accuracy versus computation time for the LoRRA model.

Model	Validation accuracy	Total loss	Number of iterations	Computation time (h)
LoRRA [22]	0.2656	-	24000	-
LoRRA	0.2218	9.8498	9200	23.24

Table 3.1: LoRRA model performance on the validation dataset in comparison with the result from [22] along with the training computation time and number of iterations.

Chapter 4

Reducing Training Time in Answering Text Visual Questions

As mentioned in Chapter 2, training an NN requires finding suitable weights using the input dataset. These weights define the strengths of the connections between neurons. Properly determined weights will result in accurate neuron connections and good model accuracy. DL networks are next-generation NNs that have more neurons and more complex neuron connections than conventional NNs. Training these complex networks requires large and comprehensive datasets that can take considerable time and resources. Consequently, reducing DL network training time is an important research problem. The goal of this chapter is to explore practical ways to reduce the training time of TextVQA without a significant effect on the accuracy.

4.1 Training Strategies to Avoid Overfitting

Overfitting occurs when models achieve high training accuracy by learning unwanted correlations in the training dataset. In other words, an overfit model fits the training set too closely [14]. Consequently, the model performs poorly on unseen data due to

a lack of generalization. Generalization is the ability of a model to use the gained knowledge to perform well on unseen data. During training, there exists a point when the model stops generalizing and starts learning unwanted correlations in the training data. Models that pass this point are overfit. Thus, the goal is to stop the training process before this point. In this section, two training strategies are implemented to avoid overfitting: early stopping and Reduce Uni-Modal Biases (RUBi).

4.1.1 Early Stopping

As mentioned in Chapter 3, the TextVQA dataset [22] is used in this study to train and validate the LoRRA model. This dataset is divided into training and validation sets with sizes 34602 and 5000, respectively. The training set is used to train the model. The validation set is an unseen dataset that is used to validate the generalization ability of the model. Early stopping [20] is a strategy to halt the training process once model validation accuracy stops improving. This is achieved by evaluating the validation dataset during training. Throughout training, the validation accuracy of the model is tracked and compared with the global average accuracy on this dataset.

Because evaluating the validation dataset to obtain the validation accuracy is computationally expensive, it would be detrimental to the training time to run the early stopping algorithm every iteration. Thus, an evaluation interval (EV) is employed. This is the number of iterations after which the validation accuracy is calculated and the early stopping algorithm applied.

If v_i is the validation accuracy at the ith evaluation interval, EV_i, then the Global Average of Validation Accuracy (GAVA) at EV_i is defined as

$$\Lambda_i = \frac{S_i}{i}, \tag{4.1}$$

where

$$S_i = \begin{cases} S_{i-1} + v_i, & \text{for } i > 1, \\ v_1, & \text{for } i = 1. \end{cases} \tag{4.2}$$

The best GAVA at EV_i is

$$\Gamma_i = \max(\Lambda_j : j = 1, \ldots, i). \tag{4.3}$$

The early stopping algorithm detects when the GAVA has converged. It allows training to continue as long as Λ_i is more than Γ_i while allowing for limited periods of time, known as patience, when Λ_i does not improve. More precisely, patience is defined as the maximum number of consecutive EVs that Λ_i is allowed to be less that or equal to Γ_i before training is terminated.

Using GAVA instead of the validation accuracy has multiple advantages. First, the validation accuracy may not increase monotonically. It may experience multiple local maximums before reaching the optimum value and some local maximums may be the global maximum up to the point they have been observed [20]. Thus, using the validation accuracy may result in an early local maxima being mistaken for the maximum accuracy. Second, a stopping criterion based on the validation accuracy can be designed to allow for a limited decrease in validation accuracy but such an algorithm needs two parameters to work correctly, how much this accuracy is allowed to decrease before stopping and how long to wait for a better accuracy. In contrast, GAVA only needs one parameter, patience.

The early stopping criteria is applied using Algorithm 1 at the end of every EV. In Algorithm 1, function inputs are Λ_i, the GAVA at the end of the EV and i, the current EV index. The function output is either $True$ or $False$, with $True$ meaning stop and $False$ meaning continue. P is patience, L is the index of the latest EV where Λ_i is greater than Γ_{i-1}. It is assumed that Γ_0 is 0.

Function `ShouldStopEarly(`Λ_i`,`i`):`
  if $\Lambda_i > \Gamma_{i-1}$ **then**
    $\Gamma_i = \Lambda_i$
    $L = i$
    return $False$
  else if $L + P < i$ **then**
    $\Gamma_i = \Gamma_{i-1}$
    return $True$
  else
    return $False$
  end

Algorithm 1: The ShouldStopEarly function is called at the end of an EV to determine whether or not the early stopping criterion has been reached.

It is clear that selecting EV presents a tradeoff between performing as few training iterations as possible and executing the early stopping algorithm as few times as possible. For example, calculating the validation accuracy for the dataset used in this thesis takes an average of 6 min and the best validation accuracy is usually reached after a few thousand iterations.

Figure 4.1 shows that using the early stopping method with the LoRRA model decreased the computation time from 23.24 h to 5.48 h. It can be observed from Table 4.1 that the LoRRA model with early stopping can be trained in less than 6 h with no adverse effect on the accuracy. Note that the Compute Canada server does not allow jobs to run longer than 24 h, thus it is important to use early stopping.

	Accuracy	Total loss	Number of iterations	Computation time (h)
LoRRA	0.2218	9.8498	9000	23.24
LoRRA with early stopping	0.2157	8.5086	3000	5.48

Table 4.1: LoRRA model performance with and without early stopping, $EV = 1000$, and $P = 0$.

Table 4.2 shows the performance of the early stopping algorithm with different values of P and EV. The best validation accuracy is in the small range 0.21 to 0.23 while the computation time is between 5.7 h and 20.0 h. This shows the large impact

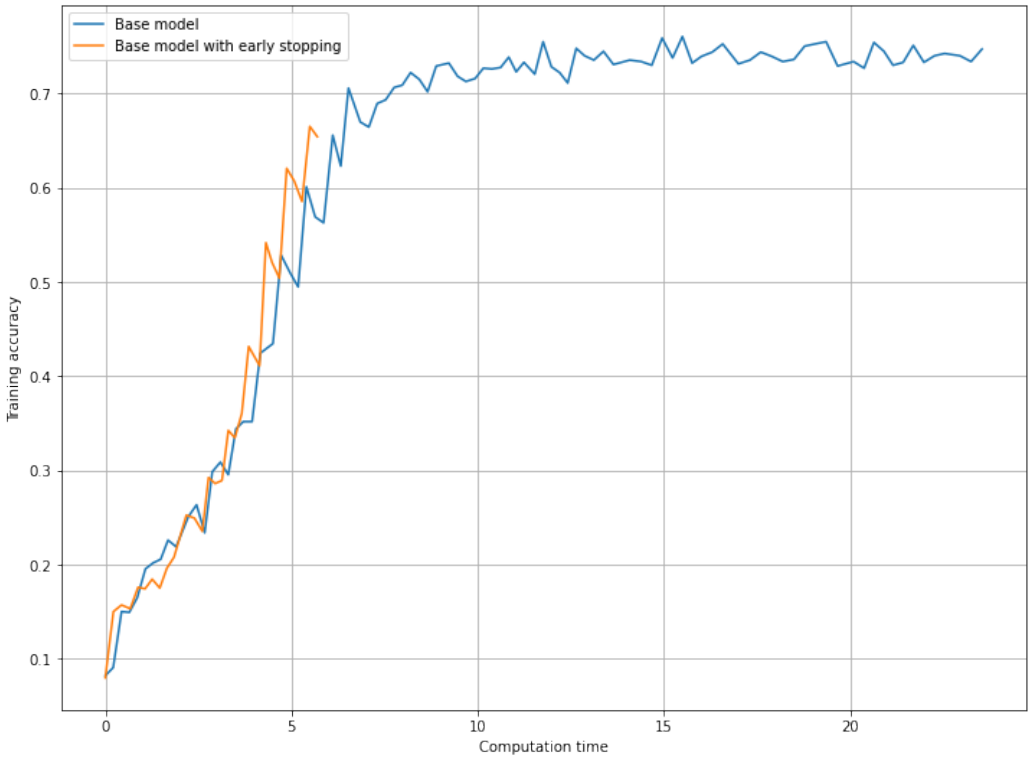


Figure 4.1: Training accuracy versus computation time for the LoRRA model with and without early stopping, $EV = 1000$, and $P = 0$.

that EV and P have on the computation time. In general, the computation time grows with EV and with a constant EV, the computation time is higher for larger values of P.

Note that two jobs with the same number of iterations may have vastly different computation times as all jobs in this thesis were run on the Compute Canada cloud which is a shared resource. Jobs are assigned to nodes depending on the system load and different nodes can have different computation power. Additionally, the state of the file system is not the same over time, so it can be slower at a given time which will have an impact on the run time because of reading from and/or writing to files.

Table 4.2 shows that the LoRRA model with $EV = 200$ and $P = 0$ achieves the highest validation accuracy (0.22) and lowest computation time (5.7 h). To ensure the validity and accuracy of this claim, the experiments were repeated three times under

Patience	Evaluation interval	Computation time (h)	Total loss	Number of iterations	Validation accuracy
0	100	1.14	7.24	300	0.14
1	100	7.01	5.67	1700	0.22
2	100	8.26	5.65	1800	0.22
0	200	5.70	5.87	2000	0.23
1	200	6.23	5.58	2000	0.22
2	200	4.75	5.65	2200	0.22
0	500	6.60	6.21	2000	0.23
1	500	11.32	5.66	2500	0.23
2	500	10.78	8.88	3000	0.23
0	750	6.83	5.64	2250	0.22
1	750	10.74	7.14	3750	0.22
2	750	9.05	5.72	3750	0.22
0	1000	11.58	6.32	3000	0.22
1	1000	9.86	8.51	5000	0.22
2	1000	20.00	9.94	5000	0.22
0	1500	10.43	5.73	3000	0.21
1	1500	10.05	5.78	4500	0.22
2	1500	13.44	5.72	6000	0.23

Table 4.2: LoRRA model performance with early stopping for different EV and patience values, learning rate 0.01 and minibatch size 128.

the same conditions for $EV = 100, 200,$ and 500. Table 4.3 confirms that $EV = 200$ and $P = 0$ results in the highest validation accuracy (0.22 on average) and lowest computation time (4.55 h on average) for the dataset and model used in this work.

Evaluation interval	Computation time (h)	Total loss	Number of iterations	Validation accuracy
100	1.14	7.24	300	0.14
100	2.06	6.27	700	0.16
100	2.79	5.66	1200	0.19
200	5.48	6.11	2000	0.23
200	3.42	5.64	2000	0.22
200	4.75	5.68	2200	0.22
500	6.60	6.21	2000	0.23
500	6.15	5.71	2000	0.22
500	7.48	6.30	2500	0.22

Table 4.3: LoRRA model performance with early stopping for $EV = 100, 200,$ and 500, $P = 0$, learning rate 0.01, and minibatch size 128.

4.1.2 Reduce Uni-modal Biases (RUBi)

As mentioned earlier, VQA requires understanding of both the image and question to provide the correct answer. It has been observed that VQA models often use biases in the training set to provide the correct answer without using the image information [7]. Figure 4.2 shows some examples from the TextVQA dataset and questions about *the city that is loved*. The majority of these examples have the same answer *Newyork*. This example shows the bias in the training set. When the most loved city is *Newyork* in a majority of the training examples, the model does not need to use both the text and image input to achieve high accuracy for questions about *the city that is loved*. Therefore, the trained model performs poorly on unseen data. In this section, the training strategy proposed in [7] is employed to address this issue. The goal is to minimize the effect of the biased training examples. This can be achieved by dynamically adjusting the model loss during training. A biased example is a question that can be answered correctly by the model without using the associated image. These biased examples can be captured by training the model using only questions as input.

Figure 4.3 shows the classical and RUBi training strategies. In the RUBi strategy, the output of the question-only branch is merged with the original prediction of the LoRRA model to obtain new predictions. Thus, the output of the question-only branch dynamically adjusts LoRRA model predictions. This branch provides a value between 0 and 1 for each answer. New predictions are the element-wise product of this branch value and the original model output. In Figure 4.3, q is the question, v is the image, a is the model prediction, and a_{target} is the correct answer. The left diagram shows how the TextVQA model is trained. At each step, the question and image are given to the model and the difference between a and a_{target} is the loss, L. The right diagram shows the RUBi strategy for training a TextVQA model. Using

RUBi, at each training step the TextVQA model is used to make two predictions, one using both q and v as input and another using only q as input (in practice a blank picture may be given as the image input to the question only branch). c_q is the model prediction using only the question as input. The element-wise product of a and c_q is the final prediction when using the RUBi strategy, a_{QM}.

Figures 4.4 and 4.5 present two examples to show why this strategy reduces bias in the model. Assuming that the target vector structure is [NY, Vancouver, SF], Figure 4.4 illustrates training a biased question in which case c_q which is $(0.8, 0.4, 0.4)$ will result in a correct answer. Here, the prediction using RUBi, a_{QM}, has lower loss than the prediction without RUBi, a, reducing the importance of this biased example in training. Figure 4.5 illustrates an unbiased question where prediction using the question alone, c_q, will result in the wrong answer. Here, a_{QM} has a higher loss than a alone and when back propagated through the TextVQA model, the higher loss emphasizes the importance of this example.

Figure 4.6 shows the training accuracy for the LoRRA model with and without RUBi and the corresponding computation times. Table 4.4 shows that the LoRRA with RUBi model can be trained in around 5 h with no adverse effect on accuracy.

	Accuracy	Total loss	Number of iterations	Computation time (h)
LoRRA with RUBi model	0.2228	8.3204	3000	5.04
LoRRA	0.2157	8.5086	3000	5.48

Table 4.4: Performance of the LoRRA model with and without RUBi.

4.2 Minibatch Stochastic Gradient Descent

Finding the best weights to minimize the network loss is an optimization problem. One way to solve optimization problems is Gradient Descent (GD) [16]. Assuming

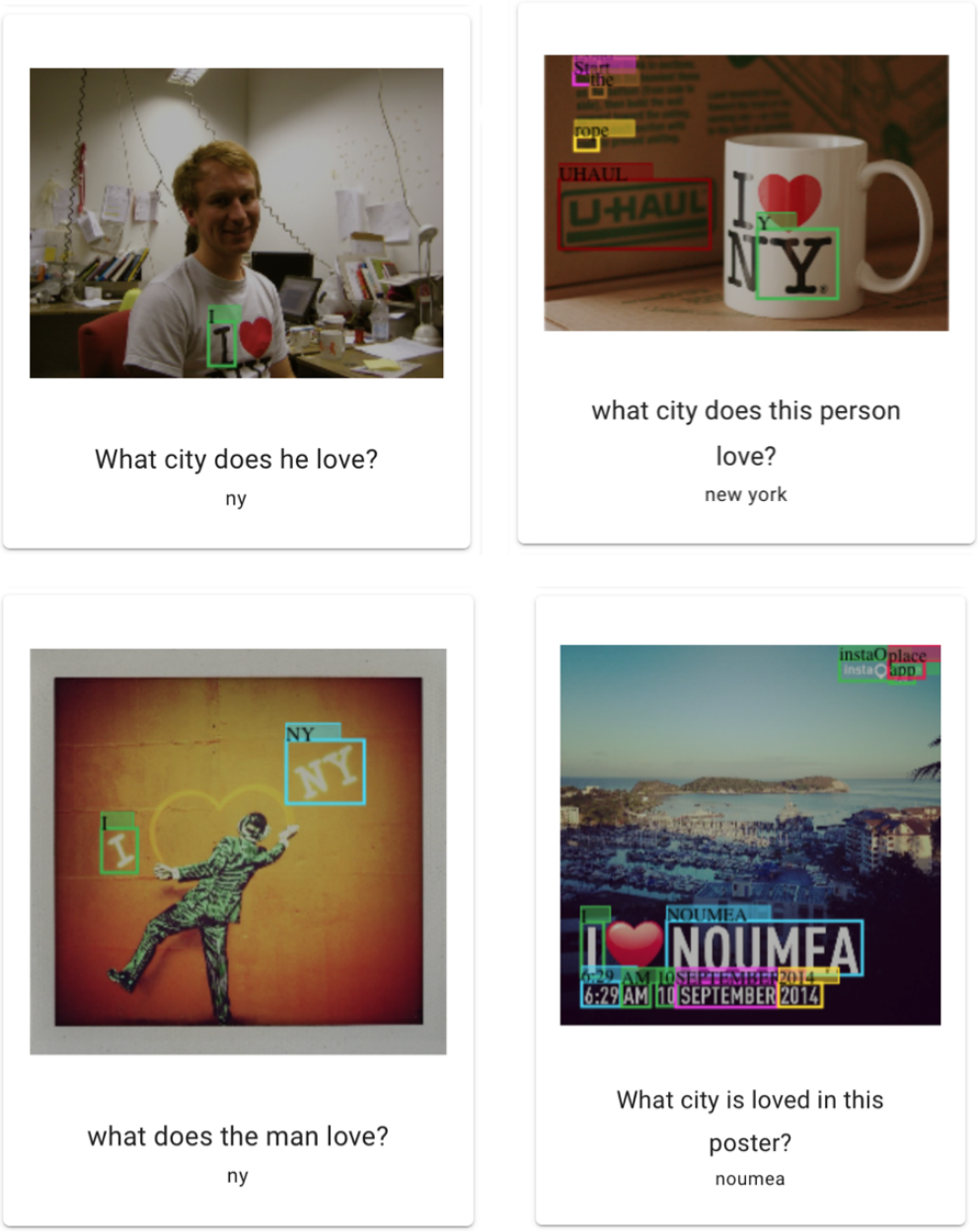


Figure 4.2: Examples from the TextVQA dataset with the question asking about *the city that is loved* [3].

training starts with the initial weight vector w_0 and w_k is the weight vector at training step k, the weights of the next step can be calculated using GD as

$$w_{k+1} = w_k + \Delta w_k = w_k - \alpha_k \frac{\partial L}{\partial w_k}, \qquad k = 0, ..., \tag{4.4}$$

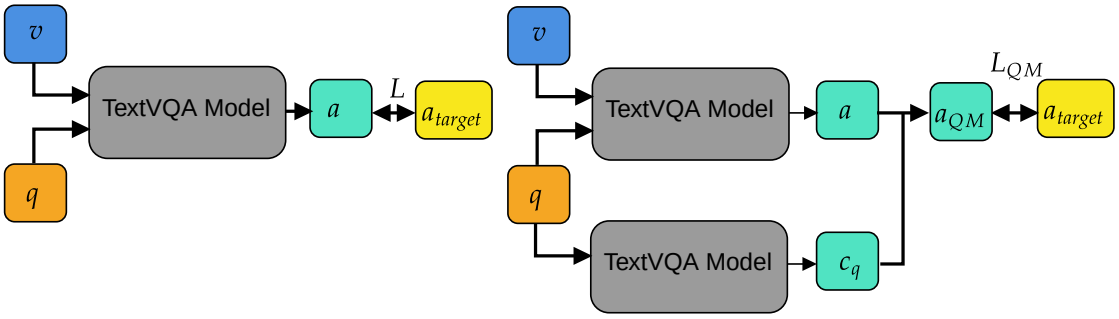


Figure 4.3: Training a TextVQA model with and without the RUBi strategy [7].

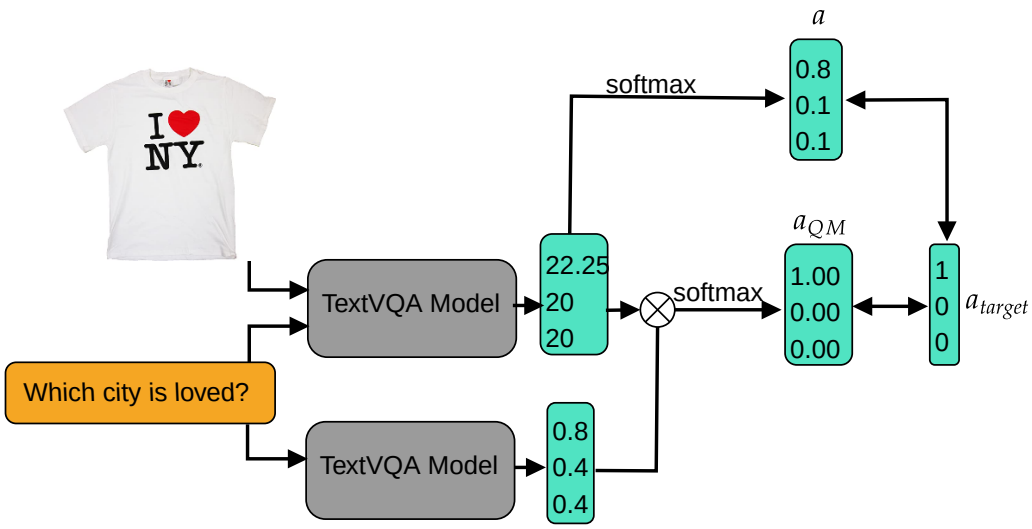


Figure 4.4: A biased example of training the TextVQA model with RUBi [7].

where Δw_k is the weight update vector at step k, L is the loss function, α_k is the learning rate, and $\frac{\partial L}{\partial w_k}$ is the partial derivative of the loss function. Using GD, the weight vector is updated in the direction of the steepest descent of the loss function. Using the training set, the loss function gradient at step k is

$$\Delta w_k = -\alpha_k \frac{\partial L}{\partial w_k} = \alpha_k \frac{1}{N} \sum_{i=1}^{N} (target^{(i)} - output^{(i)}) x_k^{(i)}, \tag{4.5}$$

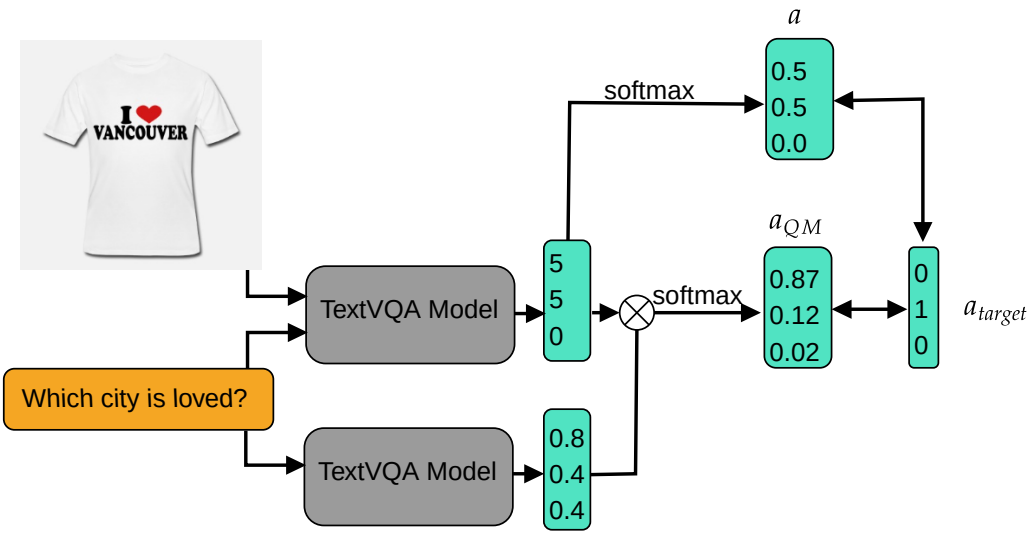


Figure 4.5: An unbiased example of training the TextVQA model with RUBi [7].

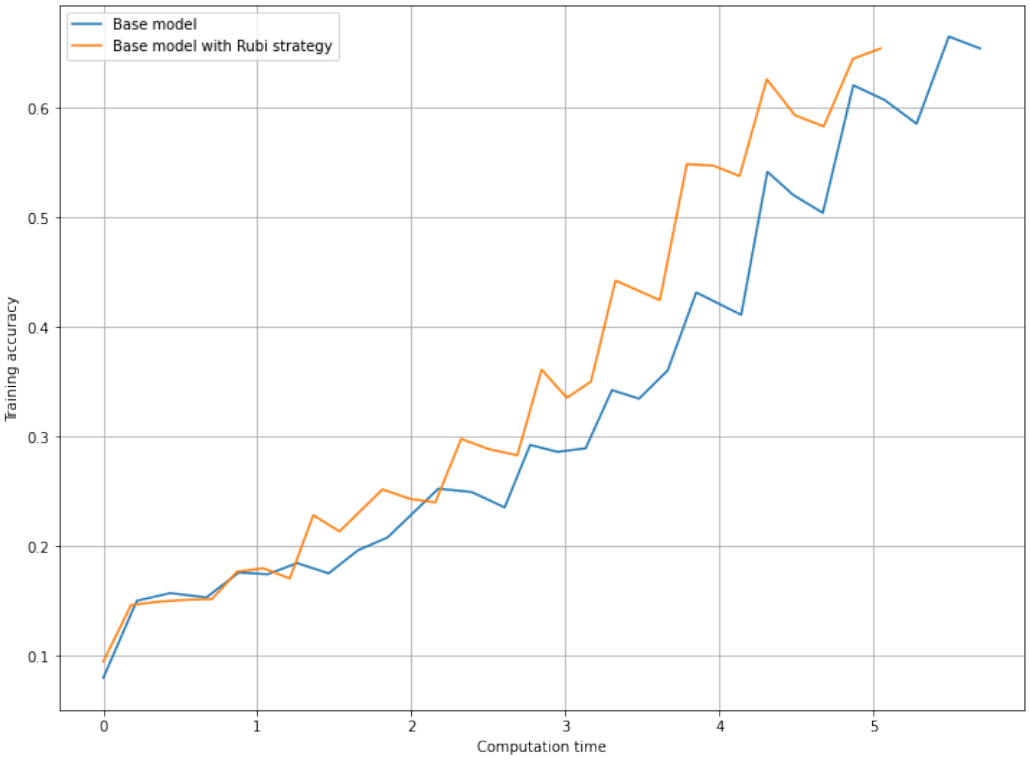


Figure 4.6: Training accuracy versus computation time for the LoRRA model with and without RUBi.

where N is the number of training samples. GD is not efficient for large datasets. Stochastic Gradient Descent (SGD) is an alternative method that approximates GD by using only one sample of the training set in each iteration. Assuming i is the index of the chosen sample

$$\Delta w_k = \alpha_k(target^{(i)} - output^{(i)})x_k^{(i)}. \tag{4.6}$$

SGD converges faster than GD on large datasets, but its performance can be improved by using a small subset of training samples instead of only one sample. The subset of training samples is usually referred to as a minibatch. Minibatch SDG calculates the loss function gradient as

$$\Delta w_k = \alpha_k \frac{1}{B} \sum_{i=1}^{B} (target^{(i)} - output^{(i)})x_k^{(i)}, \tag{4.7}$$

where B is the minibatch size and $1 < B < N$. Many computer architectures have specific vector instructions which simultaneously apply the same operation to multiple data entries. Minibatch SGD can be faster than single sample SGD because the implementation can use vectorization techniques to calculate multiple steps of single sample SGD simultaneously [9].

4.2.1 Minibatch Size

The minibatch size B is a hyper parameter which can be tuned to improve network accuracy and speed. As mentioned in Chapter 2, tuning means finding the best values so that the model neither underfits nor overfits the training dataset while learning as fast as possible. Typically, the minibatch size is a power of two since most vectorized operations perform faster with power of two inputs. Table 4.5 shows the accuracy of the LoRRA with RUBi model and the computation time for training using minibatch

sizes 32, 64, 128, and 256. The learning rate was set to 0.01 in all cases and every case was trained using the early stopping algorithm. To ensure the validity and accuracy of the results, each experiment was repeated three times. This table shows that as the minibatch size increases, validation accuracy improves until size 256 so that minibatch size 128 yields the best results. The LoRRA with RUBi model with minibatch size 128 can be trained in as low as 3.5 h.

Minibatch size	Validation accuracy	Total loss	Number of iterations	Computation time with early stopping (h)
32	0.10	7.69	400	0.97
32	0.19	5.81	2600	3.63
32	0.14	7.22	800	1.21
64	0.22	5.57	2600	7.76
64	0.17	6.01	1400	5.79
64	0.22	5.83	3000	8.78
128	0.22	5.68	1600	4.11
128	0.22	5.64	1400	3.42
128	0.23	6.11	2000	5.48
256	0.21	6.26	1400	9.22
256	0.21	6.21	1200	17.39
256	0.21	6.45	1400	9.45
512	Out of memory	N/A	N/A	N/A

Table 4.5: LoRRA with RUBi model performance with minibatch sizes 32, 64, 128, 256, and 512, early stopping with learning rate 0.01, $P = 0$, and $EV = 200$.

4.2.2 Learning Rate

The step size used to update the weights during training is called the learning rate. During back propagation, the error is calculated and fed back through the network layers to adjust the weights. The weight updates are scaled by the learning rate. In (4.4), α_k is the learning rate and this controls the speed of model learning. The best value for this parameter can be obtained via trial and error. The method used in this study to investigate the effect of learning rate on the LoRRA with RUBi model

is called grid search. Grid search is the process of searching exhaustively through a subset of hyper parameters to find the best values for a given model. Generally, grid search employs values on a logarithmic scale [10]. The range of values considered for the learning rate is 1 to 10^{-6} [4]. Therefore, in the first round of experiments, the LoRRA with RUBi model is trained and tested using learning rates 0.1, 0.01, 10^{-3}, and 10^{-4}.

Table 4.6 shows that learning rate 0.01 results in the best validation accuracy. For the second round of experiments, the LoRRA with RUBi model was trained using learning rates 0.007, 0.008, 0.009, 0.01, 0.0101, 0.0102, and 0.02 which are values close to 0.01. Table 4.7 indicates that values closer to 0.01 yield better validation accuracy. Finally, Table 4.8 shows the results of the third round of experiments using the best learning rates from the previous round three times under the same conditions to ensure the repeatability and validity of the results. These results show that small changes in the learning rate do not drastically change the validation accuracy so a learning rate of 0.01 is suitable for this model.

Learning rate	Validation accuracy	Total loss	Number of iterations	Computation time with early stopping (h)
0.1	0.00006	2805.16	400	3.03
0.01	0.22	5.68	1600	4.11
0.001	0.17	6.15	1200	8.60
0.0001	0.1	7.72	600	3.95

Table 4.6: Performance of the LoRRA with RUBi model using learning rates 0.1, 0.01, 10^{-3}, and 10^{-4}, early stopping, minibatch size 128, $EV = 200$, and $P = 0$.

4.2.3 Warm up Strategy

Warm up [11] is a strategy that involves using small learning rates at the beginning of training and increasing it until a given number of iterations n, called warm up iterations. This value is determined through trial and error. For instance, if the

Learning rate	Validation accuracy	Total loss	Number of iterations	Computation time with early stopping (h)
0.007	0.14	7.26	400	2.14
0.008	0.21	5.94	2000	11.11
0.009	0.22	5.69	1600	10.23
0.01	0.22	5.68	1600	4.11
0.0101	0.22	5.72	1400	7.62
0.0102	0.22	5.91	2000	9.17
0.02	0.0001	2805.16	400	3.63

Table 4.7: Performance of the LoRRA with RUBi model using learning rates 0.007, 0.008, 0.009, 0.01, 0.0101, 0.0102, and 0.02, early stopping, minibatch size 128, $EV = 200$, and $P = 0$.

Learning rate	Validation accuracy	Total loss	Number of iterations	Computation time with early stopping (h)
0.0008	0.21	5.94	2000	11.11
0.0008	0.22	5.63	1800	7.80
0.0008	0.20	5.92	1800	7.73
0.0009	0.22	5.69	1600	10.23
0.0009	0.22	5.76	1600	8.19
0.0009	0.23	5.61	1800	8.34
0.01	0.22	5.68	1600	4.11
0.01	0.22	5.64	1400	3.42
0.01	0.23	6.11	2000	5.48
0.0101	0.22	5.72	1400	7.62
0.0101	0.23	5.68	1600	6.67
0.0101	0.22	5.84	1600	4.15
0.0102	0.22	5.91	2000	9.17
0.0102	0.22	5.73	1800	5.36
0.0102	0.22	5.77	1600	5.19

Table 4.8: Performance of the LoRRA with RUBi model using learning rates 0.0008, 0.0009, 0.01, 0.0101, 0.0102, early stopping, minibatch size 128, $EV = 200$, and $P = 0$.

target learning rate is p and the number of warm up iterations is n, then the first minibatch iteration uses learning rate $\frac{p}{n}$, the second uses rate $\frac{2p}{n}$, and so on. At the beginning of training, the weights are randomly initialized, so choosing a large initial learning rate may lead to model instability. By applying a warm up phase, the model is initially more stable which improves convergence. In this section, warm up

is used in training the LoRRA with RUBi model to investigate the effectiveness of this method.

Table 4.9 presents the results of training the LoRRA with RUBi model with and without warm up. It is clear from these results that without warm up the validation accuracy is very close to 0 until training is stopped by early stopping. On the other hand, training using warm up reaches a validation accuracy of 0.22 in 5.04 h until stopped by early stopping.

Different numbers of warm up iterations were used in the next round of experiments and the results are presented in Table 4.10. The LoRRA with RUBi model was trained using warm up iterations 100, 200, 300, 500, 600, 700, 800, 900, and 1000. Each value was used with learning rates 0.01 (white rows) and 0.001 (gray rows). The best learning rate from the previous section is 0.01 and 0.001 is a learning rate with lower validation accuracy. Training with learning rate 0.01 (white rows), shows that as the number of warm up iterations changes from 100 to 1000, the validation accuracy and computation time are relatively stable. Another observation is that even though 0.21 is the smallest validation accuracy when using learning rate 0.01, 0.18 is the highest validation accuracy that is achieved with learning rate 0.001. This show that the warm up strategy cannot compensate for a poor learning rate which further highlights the importance of choosing an appropriate learning rate.

	Number of warm up iterations	Total loss	Accuracy	Computation time (h)
With warm up	500	5.62	0.2228	5.04 hr
Without warm up	0	2805.16	0.0001	7.41 hr

Table 4.9: Performance of the LoRRA with RUBi model with and without warm up and early stopping.

Number of warm up iterations	Learning rate	Validation accuracy	Total loss	Number of iterations	Computation time with early stopping (h)
100	0.001	0.18	6.01	1200	5.55
100	0.01	0.23	5.70	1600	4.66
200	0.001	0.19	6.11	2000	9.08
200	0.01	0.22	5.71	1600	3.69
300	0.001	0.16	6.03	1100	5.00
300	0.01	0.22	5.66	1400	6.19
400	0.001	0.14	7.07	600	2.89
400	0.01	0.23	5.68	1600	7.66
500	0.001	0.14	7.17	600	3.56
500	0.01	0.23	5.72	1400	4.96
600	0.001	0.17	6.00	1400	6.79
600	0.01	0.22	5.61	1400	4.33
700	0.001	0.14	6.95	600	2.54
700	0.01	0.22	5.62	1400	3.41
800	0.001	0.15	7.27	800	3.38
800	0.01	0.21	5.66	1400	3.33
900	0.001	0.15	6.83	800	3.36
900	0.01	0.23	5.81	2000	4.50
1000	0.001	0.16	6.38	1000	3.73
1000	0.01	0.21	5.69	1400	2.80

Table 4.10: Performance of the LoRRA with RUBi model using $n = 100$, 200, 300, 500, 600, 700, 800, and 900 warm up iterations, learning rate 0.001 (gray) and 0.1 (white), and early stopping.

Chapter 5

Conclusions

Investigating practical strategies to reduce the training time of the LoRRA model was the main focus of this thesis. LoRRA model is a state-of-the-art architecture for TextVQA dataset. An early stopping algorithm was presented. The purpose of this algorithm is to prevent the model from overfitting or spending time on training when there is no real gain in continuing the training. The early stopping algorithm has two parameters, patience and evaluation interval. The role of these parameters was discussed and their effect on computation time was analysed. A technique named Reducing Unimodal Biases (RUBi) [2] was explained. RUBi is a strategy to reduce training bias in VQA models.

RUBi was used to reduce the importance of biased examples in a training set. The results presented indicate that LoRRA model implemented with RUBi and early stopping can be trained in less than 5 h while maintaining the same level of accuracy as the LoRRA model.

Stochastic Gradient Descent (SGD) was explained and the impact of minibatch size on the LoRRA with RUBi model was examined. The results presented indicates that the LoRRA with RUBi model with minibatch size 128 can be trained in as low

as 3.5 h.

The effect of the learning rate hyper parameter on the LoRRA with RUBi model was examined. The results presented indicate that learning rate 0.01 results in the best validation accuracy. Warm up strategy was explained and the effect of warm up iterations on the LoRRA with RUBi model was examined. The results presented show that without warm up the validation accuracy is very close to 0. On the other hand, training using warm up reaches a validation accuracy of 0.22.

There are still many unsolved challenges and improvements that can be made to TextVQA. As future work, other parameters can be studied in detail since there are a number of parameters and hyper parameters that can be adjusted such as regularization, momentum and sparsity [18]. Future studies could also investigate the effect of using different activation functions on LoRRA model performance. In this thesis, the Rectified Linear Unit (ReLU) function was employed but there are a number of activation functions that can be used such as Exponential Linear Unit (ELU). In addition, using the context only branch in the RUBi strategy might prove an important area for future research. As mentioned in Chapter 4, the goal is to minimize the effect of biased training examples. A biased example is a question that can be answered correctly by the model without using the associated image. These biased examples can be captured by training the model using only words that are overlaid or embedded in the images.

Bibliography

[1] VQA Evaluation Code. https://visualqa.org/evaluation.html, 2015.

[2] Afshine Amidi, Shervine Amidi. CS 230 - Recurrent Neural Networks Cheatsheet. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks, 2018.

[3] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, Marcus Rohrbach. TextVQA dataset. https://textvqa.org/, 2019.

[4] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, pages 437–478. Springer, Berlin, Germany, 2012.

[5] Jeffrey P Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, and Samual White. Vizwiz: Nearly real-time answers to visual questions. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 333–342, 2010.

[6] Fedor Borisyuk, Albert Gordo, and Viswanath Sivakumar. Rosetta: Large scale system for text detection and recognition in images. In *Proceedings of the ACM*

SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 71–79, 2018.

[7] Remi Cadene, Corentin Dancette, Matthieu Cord, and Devi Parikh. Rubi: Reducing unimodal biases for visual question answering. *Advances in Neural Information Processing Systems*, 32:841–852, 2019.

[8] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. *Advances in Neural Information Processing Systems*, 28:577–585, 2015.

[9] Neil G Dickson, Kamran Karimi, and Firas Hamze. Importance of explicit vectorization for CPU and GPU software performance. *Journal of Computational Physics*, 230:5383–5398, 2011.

[10] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep Learning*, volume 1. MIT Press, Cambridge, MA, USA, 2016.

[11] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[12] 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 *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6325–6334, 2017.

[13] Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. VizWiz grand challenge: Answering

visual questions from blind people. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3608–3617, 2018.

[14] IBM. Overfitting. `https://www.ibm.com/cloud/learn/overfitting`, 2021.

[15] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Fei-Fei Li, C Lawrence Zitnick, and Ross Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910, 2017.

[16] Claude Lemaréchal. Cauchy and the gradient method. *Doc Math Extra*, 251:10, 2012.

[17] Christopher Olah. Understanding LSTM Networks. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015.

[18] Josh Patterson and Adam Gibson. *Deep Learning: A Practitioner's Approach*. O'Reilly Media, Inc. Sebastopol, CA, 2017.

[19] Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

[20] Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the Trade*, pages 55–69. Springer, Berlin, Germany, 1998.

[21] Amanpreet Singh, Vivek Natarajan, Yu Jiang, Xinlei Chen, Meet Shah, Marcus Rohrbach, Dhruv Batra, and Devi Parikh. Pythia-A platform for vision & language research. In *SysML Workshop, NeurIPS*, 2018.

[22] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards VQA models that can read. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8317–8326, 2019.