Voice input based automated story generation

Master Thesis

by

Rohit Keshav Bewoor

Matriculation number: 11011831

2020-11-29

SRH University Heidelberg

School of Information, Media and Design

Degree course "M.Sc."

Major field "Big Data and Business Analytics"

Reviewers	Prof. Dr. Barbara Sprick
	Dr. Frank Schulz

Affidavit (for English theses only)

Herewith I declare:

- that I have composed the chapters for the Master Thesis for which I am named as the author independently,
- that I did not use any other sources or additives than the ones specified,
- that I did not submit this work at any other examination procedure.

Heidelberg, ______________(date) Signature _______________________________

Abstract

This thesis work explores building a system as a proof of concept to generate short stories for young children aged around five to eight years old. The user was expected to speak exactly three sentences describing the kind of story required. A neural network processed the audio input and passed its inference to other downstream neural networks. These other neural networks performed object detection, image caption generation, and natural language generation. Images to accompany the story were selected from a graph database that stored information about which objects were present in which images. A survey was conducted to gauge the coherence and suitability of the generated stories with their accompanying images. The effect of using different image caption models, changing the parameters to generate the story, and manual correction of captions was evaluated. While each neural network performed its own task individually as expected with some success, it was found that the generated stories had a rather poor link to user input. Despite this, it is felt that with the current implementation as a starting point, with further training of models used, usage of more complex models, and certain design changes, the quality of generated stories could be improved further.

Zusammenfassung

Not filled as English language course and knowledge of German insufficient to translate without errors.

Table of Contents

1 Introduction

Research in the field of artificial neural networks (ANN) and Artificial Intelligence (AI) has been on-going for several decades. It can be traced back to the seminal work by Frank Rosenblatt introducing the “Perceptron” in 1958, the concept of “Backpropagation” by Paul Werbos in 1974 and several other developments in the field (Wang, Raj and Xing, 2017). However, the recent ubiquitous applications of ANNs have become possible due to a) the availability of cheaper computational hardware, b) the widespread use of machine learning frameworks like Pytorch and Tensorflow, and c) the availability of large corpus of data to train such models. Due to these reasons, the size of ANNs can now be huge with both a large number of neurons in a layer and the number of layers running into even hundreds or thousands. Such networks are therefore also called Deep Neural Networks (DNNs).

Research was renewed with vigour since the ground-breaking classification model “AlexNet” was demonstrated at the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) challenge and the model surpassed all other competitors (Krizhevsky, Sutskever and Hinton, 2017). Since then, different architectures of ANNs have been developed to cater to different tasks. The use of so-called “end-to-end neural networks” allows a specific type of network to perform tasks like speech recognition, image detection, image segmentation, image classification, language translation, and image captioning. Other applications are being developed even today with continuing research. These ANN models can then be deployed for a real-world problem. For example autonomous driving to allow real time inputs and suitable action by the AI control system - which relies heavily on an image detector model to output an inference on each frame of a video captured from a car mounted camera. Another exciting use-case is with speech transcription that allows a user to interact with natural speech as the input to a system. For speech transcription, prior to end-to-end ANNs being available, one had to perform extensive feature engineering based on methods like HoG (Histogram of Gradients) and still handle only limited vocabulary. But now systems like Google Voice assistant on Android phones and Apple's Siri on the iphone, Amazon's Alexa, etc. are among a host of popular systems that use speech recognition models in the backend. The author, being inspired by the these examples, decided to pursue a thesis topic that involved implementing a system that would process some user input via multiple neural networks to achieve an overall goal.

Now a chapter-wise breakdown, followed by the objective, the motivation for pursuing this thesis work, and a high level introduction will be covered.

This study is divided into various chapters with the reader currently at **Chapter 1 "Introduction"**. The **"Literature Review"** is covered in **Chapter 2** summarising the on-going research. **Chapter 3 "Fundamentals"** has a brief explanation of the various concepts used in this thesis work. This is to act as an aid to the uninitiated reader. The various datasets and resources that were used to build the system are described in **Chapter 4 "Datasets used"**. This will explain the data itself and any pre-processing that was done to make the data usable for the study. **Chapter 5 "Implementation"** explains how the solution in this study was implemented. It covers the models used for various tasks required in the pipeline and the manner in which data flows from one stage to the next. **Chapter 6 "Evaluation"** will cover the overall results by showing some sample outputs accompanied by the results of a survey asking respondents to rate the stories generated and their respective images. Finally, **Chapter 7 "Conclusion"** summarises the results, contributions, learning's for the author, and scope for further improvements to meet identified shortcomings.

1.1 Objective

Allow a user speak and describe elements of a required story and then generate a short story suitable for young children (around five to eight years old) with accompanying images.

1.2 Motivation

As part of the coursework of the author, several types of neural networks and their applications were introduced. During the course period, the Covid-19 pandemic struck with lockdowns being enforced. People had to limit their outdoor activities and stay at home for long periods. The author noticed that mothers with young children were hard hit with having to keep their children entertained at home. Further, they wanted to limit the time their children could watch television or videos on the internet. Young children quickly get bored of their toys and books available at home and it is a real challenge to keep them continuously engaged.

Therefore, the author wondered whether it would be possible to use the concepts studied in the coursework to create a story generator for such a target group. The user here being the mother, could describe the kind of story required in a few short sentences. The system would generate a story with images that could be shown to the child with the story being read by the mother. If the proof of concept worked, by training DNNs with sufficiently large data and an exhaustive database of images, there would be no limit to the kind of stories generated. The novelty of each new story would keep both the mother and the child engaged. Hence, the author decided to attempt implementing such a system by weaving together different neural networks.

1.3 High level approach

A quick introduction to the implementation approach is explained in the diagram below. Audio input is provided as Wav format files. The final output expected is a block of text and some image files to accompany that text. Between one to three images would be selected and the text would be around 300 words in length.

Models 1-4 are various neural networks that carry out specific tasks and progressively the data is passed between them to achieve the overall goal.

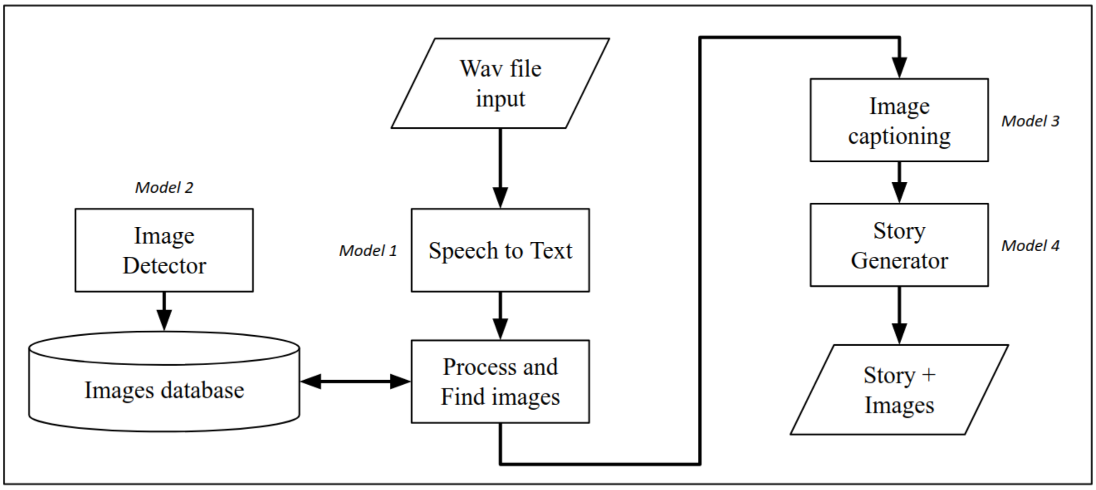


Figure 1: High Level Block Diagram

Model 1: A pre-trained Speech-to-text (STT) neural network that processes an input audio file and outputs the English text transcription of what was said in the audio.

Model 2: An object detector neural network that processed an image file as input and output the various objects that it is trained to detect. The objects it can detect are also called the "labels" or "classes". The image database was a Neo4j graph database that stored information about which objects were detected in which image.

Thus, later on one could query the database to find images containing any objects of interest.

Model 3: An image caption generator neural network that processed an input image file and output a short sentence describing what it thought was in the image. This is called the “caption”.

Model 4: A transformer based neural network that generates new text on the basis of some input text. The input text is also called the “seed”.

2 Literature Review

2.1 Speech-to-Text (STT)

Speech recognition models had always been an on-going area of research and prior to 2006, such systems were heavily reliant on feature engineering and the use of Hidden Markov Models (HMM). Then there was the approach to use Gaussian Mixture Models (GMM) which led to improved performance. However, with the use of Deep Neural Networks for this task, they were found to outperform the HMM and GMM based models. For example, Microsoft released its MAVIS system in 2012 and it had a word error rate reduction of more than 30% compared to other state-of-the-art (SOTA) GMM based models at the time. (Nassif, Shahin, Attilli, Azzeh and Shallan, 2019)

Since then Automatic Speech Recognition (ASR) models have been incorporated into commercial products released by the tech-giants. The performance of the models of leading five ASR providers viz. Google Cloud, IBM Watson, Microsoft Azure, Trint and YouTube is quite good with the word error rate in the range of 4-8% (Kim, Liu, Calvo, McCabe, Taylor, Schuller and Wu, 2019).

However these services were not free and for this thesis work and therefore unsuitable for use in this thesis work. Luckily, the author came across an open source implementation of a Speech-to-text (STT) model by Mozilla foundation (DeepSpeech Project Website, 2020). A pre-trained model could be downloaded and used for any application.

The DeepSpeech model was based on a paper released by Baidu research in 2014. The model consisted of 5 hidden layers. The first three were non-recurrent, while the fourth was a bi-directional recurrent layer with two sets of hidden units. The final layer processed both outputs from the bi-directional layer to make a word prediction.

Training was done with a 5-10% dropout in the layers to improve robustness. (Hannun, Case, Casper, Catanzaro, Diamos, Elsen, Prenger, Satheesh, Sengupta, Coates and Ng, 2014).

Mozilla foundation continued developing the model and as of June 2020, had released its stable version 0.7.3 that had a word error rate of approximately 6.5% (DeepSpeech FAQ Website, 2020). This was comparable to the performance of the paid STT services and thus deemed suitable for use in the thesis work.

2.2 Object Detection

Object detection neural networks are essentially really deep CNN based networks. The layers treat the input feature maps as a 3D matrix of the pixel intensities for different color channels. At any given layer, the values of the outputs in the feature maps are affected by the surrounding locations in the previous layer. A series of transformations like filtering and pooling are performed at each layer progressively. This reduces the size of the features map but increases the number of channels. The filters used, also known as kernels, have a relatively small fixed size (typically 3x3 or 5x5). Such networks can then not only output the class label of the objects found but also the bounding boxes showing the exact location of the objects detected.

One group of object detectors is based on the region proposal methods. These include the R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN to name a few. Such models operate via a two-step process. First the regions of interest are identified during the so called Region Proposal Generation stage. Thereafter, each region proposal is converted into a fixed input size and processed by a CNN module. (Zhao, Zheng, Xu and Wu, 2019).

The other category is called the One-stage detectors as they do not have a region proposal step. The popular models are YOLO, RetinaNet and SSD. These work directly by mapping straight from the input image features to being able to predict bounding boxes and the class probabilities of objects detected. The first version of YOLO had only 24 convolutional layers but faced problems detecting objects that were relatively small and occupied fewer pixels in the input image. Subsequently, newer versions of YOLO made improvements. The third version - YOLOv3 - could inspect the input image at multiple scales that allowed even smaller objects to be detected successfully. In general, single-stage detectors are orders of magnitude faster than two-stage detectors but at the cost of poorer accuracy. For example, the Faster R-CNN (with a ResNet101 backbone) had an mAP of 83.8% taking an average of 2.24 seconds per image for inference. But YOLOv2 had an mAP of 78.6% with an inference time of only 30 milliseconds per image. (Zhao, Zheng, Xu and Wu, 2019)

YOLOv3 introduced the concept of anchor boxes which allowed detection of multiple objects within the same grid cell. Thus, with 3 scales and 3 anchor boxes per grid cell there were 9 anchor boxes in the model. These anchor boxes were defined based on a k-means clustering of the typical objects that the model was trained to detect. This model was also much deeper with 53 convolutional layers. Despite this, the inference time per image averaged only around 25 milliseconds per image and therefore did not compromise on speed. The added advantage of detecting objects at different scales (allowing even small objects to be detected successfully) made this model super-fast and quite adept at handling a varied set of input images. (Redmon and Farhadi, 2018)

Based on the above observations, a YOLOv3 based system will allow decent accuracy with fast inference – making it suitable for implementation in the thesis work.

2.3 Image Captioning

Image captioning models can be broadly divided into three types: Retrieval based, Template based, and End-to-end learning based. The Retrieval based models are no longer popular as they try to find similar images to the ones in the repository and then depend on the captions of the similar images to generate the output. These models were therefore heavily dependent on the breadth of the repository and did not scale well. In the second method, Template based, two tasks are involved in the generation of the output. First, object detection produces a "bag of likely words". Then these likely words are used to create templates which are used by an LM to actually generate the output caption. The limitation of such an approach is that the model does not truly understand the input image features per se, and can only attempt to combine the objects detected in a meaningful way using the pre-defined templates. Such models are falling out of favour now. Finally, the third type has developed most recently and is the so called End-to-end models. These use a deep-CNN model as an encoder to convert the input image into a feature map encoding. The feature map remains static but the caption words are generated one at a time. Together, these are continuously processed by an RNN decoder to create the most meaningful captions that are truly dependent on the input image features. This is the most popular approach now. There are various architectures that involve the use of "attention" to improve performance. Other approaches are to combine the features via the "Merge" or the "Inject" architectures. (Liu, Xu and Wang, 2018)

To reduce the computational workload during training and during inference, GRUs can be used in place of LSTMs in the RNN decoder. This works because while the GRU has a similar structure to that of an LSTM, it does not have separate memory cells and has lesser number of gates compared to the LSTM. In a study comparing the BLEU-1 score of various state-of-the-art end-to-end type models, they had scores of around 0.7 (Hossain, Sohel, Shiratuddin and Laga, 2018)

The debate around Merge vs Inject architecture is still not settled however decisively. The inject method was the popular method initially. This was because the Merge method was counter-intuitive as one would expect that the RNN ought to be influenced by the features of the input image. However, the late merging of the word features and the image features in Merge architecture does not seem to degrade performance. The BLEU-1 scores for Merge were almost always higher than the Inject models scores for the Flickr-30k dataset. (Tanti, Gatt and Camilleri, 2017)

On the contrary though, another study by the same authors found the BLEU scores for the COCO dataset were slightly better for the Inject architecture, but only marginally better. (Tanti, Gatt and Camilleri, 2018)

Regarding with and without attention models, two well-known models are the “Show and Tell: Neural Image Caption” which used no attention, and the “Show Attend and Tell: Neural Image Caption Generation with Visual Attention” model that did use attention mechanism.

The Show-Tell model was based on the Inject architecture and had BLEU-1 scores of over 0.6 on the Flickr-30k and Flickr-8k datasets. (Vinyals, Toshev, Bengio and Erhan, 2015)

On the COCO dataset, the Show-Attend-Tell model had BLEU scores that were around 5% better than the Show-Tell model. For the Flickr-8k and Flickr-30k datasets, the scores were generally better, but not as consistently as with COCO dataset (Xu, Ba, Kiros, Cho, Courville, Salakhutdinov, Zemel and Bengio, 2015).

Therefore in this study image caption generators of both types were used.

2.4 Natural Language Generation (NLG)

NLG as a field has a very vast scope but can broadly be divided into two categories. The “text-to-text” generation maps to tasks like machine translation, text summarization, etc. The other category is “data-to-text” generation for tasks like sports reports, weather or financial reports, etc. In the context of generating creative text, the task could be to generate short jokes or puns, metaphors and similes, or the more complex task of outputting narratives. (Gatt and Krahmer, 2018).

However, specifically related to the task that is required for the use-case in this thesis work, a team from IBM Research India attempted to create a coherent story using short sentences as independent descriptions. They explored approaching the problem from two viewpoints: as a translation problem (called Statistical Machine Translation approach) or as a Deep learning problem by treating the task as a sequence-to-sequence generation task. Their sequence-to-sequence model was called Desc2Story and used RNN based encoder and decoders. The encoder was bi-directional type while the decoder used attention mechanism. The BLEU scores were however very low (all under 5% in various types of models). (Jain, Agrawal, Mishra, Sukhwani, Laha and Sankaranarayanan, 2017)

However, sometime during the course of the thesis-work a state-of-the-art NLG model called Generative Pre-training version 3 (GPT-3) was released by the Open AI project. Their model had around 175 billion parameters. The model architecture was the same as for an earlier model called GPT-2. Even for a language modelling task with zero-shot evaluation, the GPT-3 model set a new benchmark by beating the previous SOTA model’s score by a margin of 15 points on the Penn Tree Bank dataset (PTB dataset). (Brown, Mann, Ryder, Subbiah, Kaplan, Dhariwal, Neelakantan, Shyam, Sastry, Askell, Agarwal, Herbert-Voss, Krüger, Henighan, Child, et al., 2020)

GPT-2 was a transformer based model that was in turn based heavily on the prior GPT-1 architecture - with certain changes like the normalization of the layer outputs being moved earlier in the network. (Radford, Wu, Child, Luan, Amodei and Sutskever, 2019)

With GPT-1, the Open AI team used a semi-supervised approach for training the language model (LM). This so called pre-training phase was totally unsupervised and was followed by a fine-tuning phase that was supervised. The first phase allowed their LM to learn general trends in the huge corpus of data so as to set the LM parameters as a baseline. Then, in the second phase, the parameters were tweaked for a specific task. The model could handle multiple tasks like text classification, text similarity, selecting an answer to a question out of multiple choices presented as the answer, etc. The overall model was a multi-layer transformer-decoder (with 12 layers in GPT-1). The attention mechanism was multi-headed with self-attention capabilities. (Radford 2018)

Due to non-availability of access to the GPT-3 model for the story generation stage of this thesis work, the author had to contend with using the GPT-2 model after some further training.

3 Fundamentals

A basic understanding of certain concepts is necessary to understand this thesis work. Therefore, for the reader unfamiliar with these concepts, this section covers a brief introduction to these topics related to deep learning in AI:

1. Convolutional Neural Network (CNN)
    a. Object Detection
2. Recurrent Neural Networks (RNN)
3. Speech-to-Text (STT)
4. Image Captioning
    a. Merge and Inject architectures
    b. Attention based architecture
5. Natural Language Generation (NLG)
    a. Language Models (LMs)
    b. Beam search and Greedy Search
    c. BLEU score

3.1 Convolutional Neural Networks (CNN)

This is a type of neural network that has been found to work very well for image processing related tasks. Any image is essentially a long array of floating point numbers that represent the colors in each pixel. Using a filter (also called a kernel), a series of mathematical operations (called convolutions) are performed on each part of the image to output a new number. These values are recorded so as to retain the spatial information and have an idea of which number was influenced by which part of the input image. That is, for the same kernel being convolved with different parts of the input image, we get different numbers in the output of the convolution result. This convolution operation is performed across many layers in what is typically a deep neural network.

3.1.1 Object detection

In this study, a model pre-trained on the COCO dataset was used for object detection. It was based on the YOLO-v3 algorithm. This is a single-stage detector and hence fast compared to alternative methods based on the RCNN family of algorithms.

The basic steps involved in the YOLOv3 algorithm are described below:

- Image is divided into a grid of S x S grid-boxes
- 3 scales are used to make the predictions. This causes the model to divide a 416 x 416 pixel input image into grids of sizes 13x13, 26x26 and 52x52
- Each grid can predict a maximum of some number of objects, say B, if the model thinks that an object's center lies in the pixels covered by that particular grid
- Model outputs four regressed values for the bounding box information: t_x, t_y, t_w, t_h
    - Mathematical formulae are applied later on these regressed values to compute the actual bounding box center coordinates and dimensions: b_x, b_y, b_w, b_h
- Model also outputs a confidence score to indicate how strongly it feels there is actually an object present in that grid.

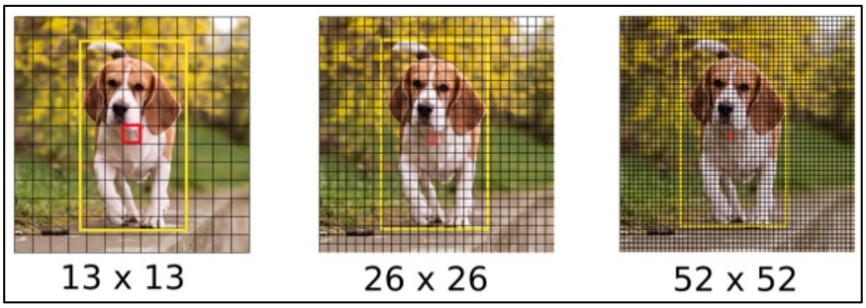


Figure 2: Grids at different scales shown against input image

Source for image above: https://medium.com/analytics-vidhya/yolo-v3-theory-explained-33100f6d193 (Balys, 2019)

For example, suppose that:

- input image is divided into 13 x 13 grid (i.e. S = 13)
- we want to detect 80 classes for COCO dataset (i.e. C = 80)
- 3 objects can be predicted per grid cell (B = 3)
- Number of calculations for Detections will be = (13 x 13) * (3 * (5 + 80)) = 13 x 13 x 255 = 43,095 total detections
- For all possible detections, by comparing the confidence scores output against some minimum threshold value, one can decide whether an object is present or not

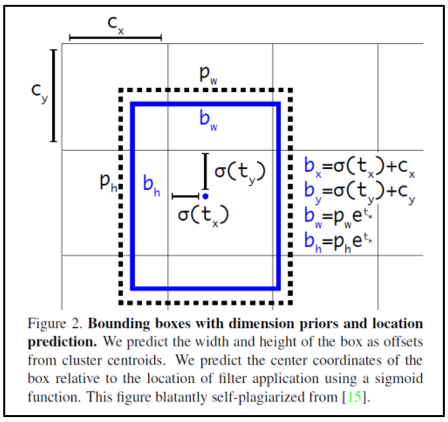


Figure 3: Regression value for bounding box predictions

Source for image above: YOLO v3 paper (Redmon and Farhadi, 2018)

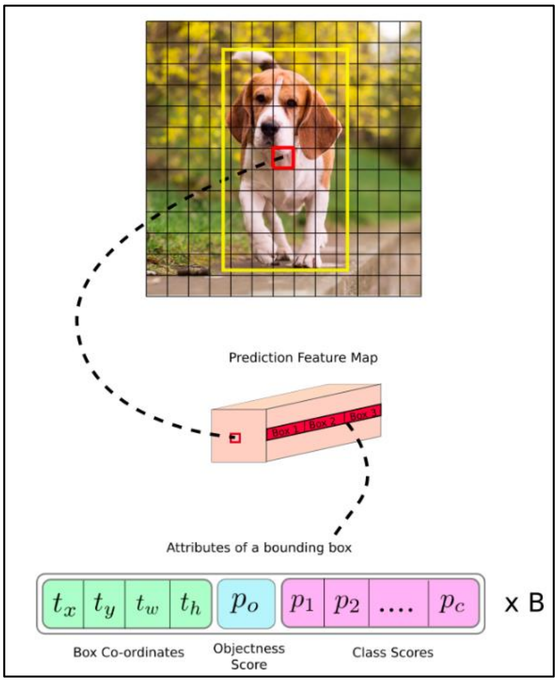


Figure 4: Bounding box for detection

Source for image above: https://medium.com/analytics-vidhya/yolo-v3-theory-explained-33100f6d193 (Balys, 2019)

The detailed architecture of the model is shown in Appendix A.

3.2 Recurrent Neural Networks (RNN)

A regular feed-forward neural network will always produce the same output for a given input, as the output is only influenced by the value of the weights in the network. However, RNNs are different because they employ a concept of “hidden state”. This hidden state allows the output to be influenced not only by the present input, but also by the current value of the hidden state. The current value of the hidden state depends on the inputs that have already appeared before in previous time steps. Thus, RNNs are heavily used where the output should be influenced not only by the current input value but also by the previous values. Networks based on RNNs are used extensively in application for Speech processing, Language Models, etc. Under the broad category of RNNs, there are various types of basic units that can be used such as Long-short Term Memory (LSTM) and Gated Recurrent Unit (GRU). These are more complex in their operation than the basic vanilla RNN units but have advantages that make them popular in specific use cases. This thesis work uses models that are based on LSTM and GRU units.

3.3 Image Captioning

The task of such neural networks is to produce a short description in natural language to describe an input image. These neural networks are usually based on a combination of a CNN and an RNN network. The salient points about such networks are:

- Input is an image.
- Output is a sentence describing the image i.e. the caption.

- Encoder: Processes the input image into a numerical vector of features that the Decoder can then process.
  - Typically a pre-trained model is used here.
  - The last layer just before final softmax of a Deep-CNN classifier network is tapped to access the encoded values
- Decoder: Processes the input image feature vector along with any words predicted till that time step to predict the next word in the caption.
  - After training: Processes only the Image encoding to output the Predicted caption
  - During training: Processes the image encoding + Input ground truth captions to learn correct predictions

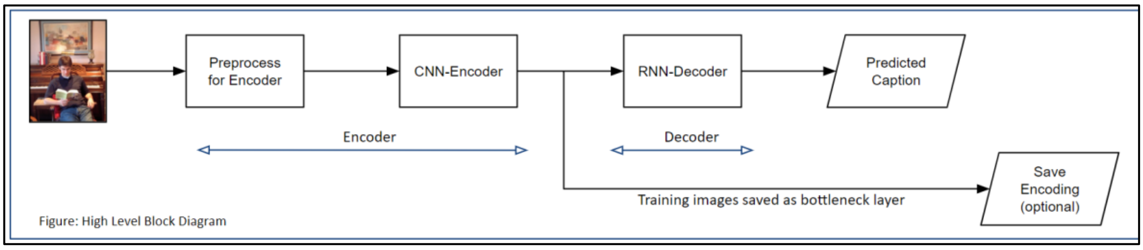


Figure 5: High level block diagram of an Image caption network

For a detailed architecture diagram of one of the decoder networks used in this study, please refer to the Appendix.

3.3.1 Merge and Inject architectures

The decoder has to somehow process two types of data:

- the image encodings that come from the encoder
- the tokenized representation of the words that form the caption

These two types of input data can be combined either before the LSTM RNN layer or after it. The choice of how the model is defined is the idea behind the Merge vs Inject architecture. In inject type, the image features are allowed to influence the RNN language model. But in the merge type architecture, since the combination occurs after the RNN LSTM layer, there is no direct influence on it.

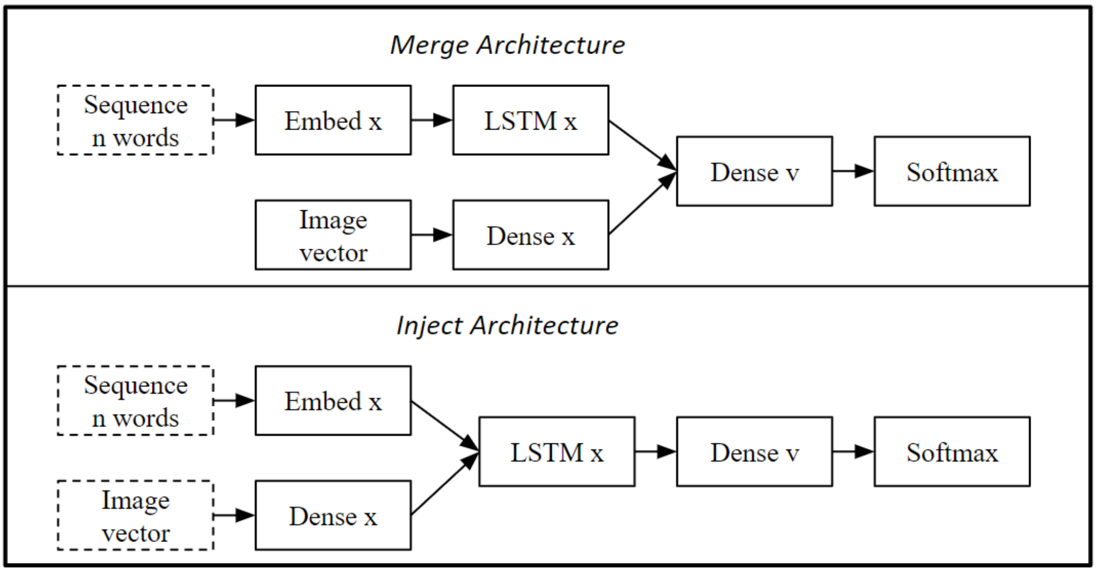


Figure 6: Comparison of Merge vs Inject Architecture

Source for image above: (Tanti, Gatt and Camilleri, 2017)

3.3.2 Attention based architecture

When one thinks of the concept of attention, one can imagine that there is a lot of information around us. And we are focussed on some particular aspect of information because only that subset of information is sufficient and necessary to get the job done. The same logic applies in the context of neural networks that have an attention mechanism. Essentially the network focuses its view on only some parts of the total input features. Such techniques have been applied successfully in the field of image captioning. Two popular methods are the so called Bahdanau attention and the Luong attention mechanisms.

3.4 Natural Language Generation (NLG)

One way to define Natural Language Generation is “the task of generating text or speech from non-linguistic input” (Gatt and Krahmer, 2018). However, the same paper elaborates on this by stating there are two broad subcategories here:

- *text-to-text generation*: applications that take existing texts as their input, and automatically produce a new, coherent text as output
- *data-to-text generation*: applications that automatically generate text from non-linguistic data

In this study, the the first scenario is applicable. The input text was a seed value consisting of two to three sentences. The output was a story of around 300 words of text.

3.4.1 Language Models (LMs)

Language models are required to predict the next word - given some set of preceding words(Brownlee, 2019). These are also referred to as Statistical Language Models. A programming language, like Python, C, etc. has limited vocabulary and very rigid syntax and rules to write the code. But the languages used by humans, like English, have a huge vocabulary and ways of expressing ideas in written form.

The number of words that are tracked by an LM, as the sequence of words to consider before selecting the next best word, is denoted by “n”. The LM is then categorised as an n-gram LM. E.g. a 5-gram LM keeps track of the 5 preceding words and then decides what the next word should be.

The actual strategy used for predictions, relies on a probabilistic algorithm to select what could be the next best word. Here, two popular approaches are the Beam and Greedy Search as explained below.

3.4.2 Beam search and Greedy search

The problem being addressed by these algorithms is to select the next best word to output, given a series of some words that have already appeared.

Both methods rely on computing a joint probability of a new candidate word and the preceding set of words. The table below summarises the differences.

Description	Greedy search	Beam search
Number of sets of preceding words tracked for each word prediction	Always only one	Depends on the Beam Width parameter. E.g. width = 3 means there are always a three sets of preceding word sequences that are under consideration at any given time.
Complexity	Low	High (very high if the beam width is large)
Inference time	Fast	Can be very slow if large beam width is chosen.
Variability in output	More rigid	Less rigid

Table 1: Beam search and Greedy search - pros and cons

3.4.3 BLEU score (Bilingual evaluation understudy)

This is a metric used to gauge - how good is the quality of the output text generated by a model? It is a popular technique in machine translation and is used during training time to compare the results achieved by the model after training.

For every data point, one needs several versions of the ground truth text for comparison with the actual output of a model. Roughly speaking, the metric finds how many words that appeared in the model output, were also present in the ground truth sentences. In reality, the score considers several n-gram sequences from the ground truth text and then blends multiple scores together into a single metric – but that is beyond the scope of this work to get into details at that level.

Being a popular metric, the score can be easily calculated by software packages like NLTK – which was used in this thesis work.

4 Datasets and Tools used

4.1 Description and Collection

Dataset 1: Common Objects in Context (COCO)

This was an exhaustive dataset ideal for image processing tasks using neural networks. The data can be used to train models for image classification, detection, segmentation, caption generation, keypoints detection, etc. It consists of 80 object labels with more than 300,000 images. Of these, labelled data is provided for more than 200,000 images. This study has used the 2017 version of the dataset.

In the context of this thesis work, the image files were obviously used. But the other key file was the annotations file. It was a json format file that contained the ground truth captions. For every input image, approximately 5 ground truth captions were available.

The project was sponsored by the likes of Microsoft and Facebook, among others.

Project link: https://cocodataset.org/

Dataset 2: Children's Book Test (CBT)

This dataset was made available by the Facebook research team as part of its bAbI project. This dataset was suitable for training Language Models and it was built from free to download e-books that appeared on the Project Gutenberg website.

While other studies had used this dataset for question-and-answer training use-cases, it was felt that using the stories in this dataset to train the story generator would be ideal.

Project link: https://research.fb.com/downloads/babi/

Dataset 3: Flickr 30k

This was a dataset of images sourced from the company Flickr, by the computer science department of University of Illinois. It was meant for use for educational purposes only. The corpus consisted of over 150,000 image captions that described the approximately 30,000 images.

In the use-case, while the dataset could have been used for image caption training, there was enough data from the COCO dataset. Therefore, the images were only used for populating the graph database.

Project link: https://shannon.cs.illinois.edu/DenotationGraph/

4.2 Preparation and Cleaning

COCO dataset:

No clean up or pre-processing was required in the process of loading the image database. The image files were simply used for inference by the object detector.

However, for the training of the image captioning models, there was some pre-processing required. This involved steps such as:

- Removing punctuation and special characters
- Removing hanging letters after the above step. E.g. “He’s going to eat now” would become “He s going to eat now” after removing the apostrophe. The single “s” character became a garbage value and would degrade a ground truth caption used for training.

```
In [24]: ## example of caption with accidental newline \n in the caption
         descriptions_test['000000482917']

Out[24]: ['A dog sitting between its masters feet on a footstool watching tv\n',
          'A dog between the feet of a person looking at a TV.',
          'A dog and a person are watching television together.',
          'A person is sitting with their dog watching tv.',
          'A man relaxing at home, watching television with his dog.']

In [25]: # prepare translation table for removing punctuation
         ## string.punctuation  gives   '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~' and will take care of all these characters being made in
         to a space
         tran_table = str.maketrans(string.punctuation, ' ' * len(string.punctuation))
         for key, desc_list in descriptions_test.items():
             for idx in range(len(desc_list)):
                 desc = desc_list[idx]
                 # replace all punctuation with space in description before tokenizing
                 desc = desc.translate(tran_table)
                 # tokenize
                 desc = desc.split()
                 # convert to lower case
                 desc = [word.lower() for word in desc]
                 # remove hanging 's' and 'a'
                 desc = [word for word in desc if len(word)>1]
                 # remove any non-alphabetic tokens
                 desc = [word for word in desc if word.isalpha()]
                 # overwrite with cleaned description
                 desc_list[idx] =  ' '.join(desc)

In [26]: ## example of caption with accidental newline \n in the caption  -- POST CLEANUP
         descriptions_test['000000482917']

Out[26]: ['dog sitting between its masters feet on footstool watching tv',
          'dog between the feet of person looking at tv',
          'dog and person are watching television together',
          'person is sitting with their dog watching tv',
          'man relaxing at home watching television with his dog']
```

Listing 1: Code snippet - caption cleaning for without-attention model training

```
In [49]: # Choose the top 5000 words from the vocabulary
         top_k = 5000
         tokenizer = tf.keras.preprocessing.text.Tokenizer(num_words=top_k,
                                                           oov_token="<unk>",
                                                           filters='!"#$%&()*+.,-/:;=?@[\]^_`{|}~ ')
         tokenizer.fit_on_texts(train_captions)
         train_seqs = tokenizer.texts_to_sequences(train_captions)

In [50]: type(train_seqs)

Out[50]: list

In [51]: train_seqs[:10]

Out[51]: [[3, 848, 6, 2804, 6, 61, 27, 1990, 242, 10, 437, 4],
          [3, 2, 430, 11, 3410, 8, 1024, 396, 498, 1139, 4],
          [3, 64, 20, 1038, 144, 9, 191, 948, 6, 735, 4],
          [3, 301, 727, 26, 346, 210, 264, 10, 437, 4],
          [3, 2, 172, 6, 1139, 27, 445, 191, 61, 4],
          [3, 2, 119, 113, 61, 97, 7, 33, 6, 7, 129, 4],
          [3, 2, 119, 18, 35, 679, 2, 129, 4],
          [3, 2, 119, 773, 9, 155, 206, 8, 7, 435, 4],
          [3, 16, 207, 18, 8, 2, 129, 144, 102, 4],
          [3, 2, 119, 18, 21, 13, 2, 435, 144, 9, 130, 4]]
```

Listing 2: Code snippet - caption cleaning for with-attention model training

The requirement was to map every unique word that was part of the vocabulary to a unique integer value. This interger value was used to extract the word-embeddings vector that represented the vocabulary word.

The standard procedure to achieve this was to use Python dictionary data structures called "WordToIx" and "IxToWord". The "WordToIx" was used to map from the word tokens to the unique integer representation. The "IxToWord" was for the reverse mapping from integer to the unique word token.

Thus, even after clearning the ground truth descriptions and then following the steps mentioned earlier, the other typical steps for further pre-processing are explained below.

Here, the Image Feature (**ImgFeat** in the table below) was a vector of 2048 floating point values that were the encodings received from Encoder. The Description was the cleaned ground truth sentence with insertion of special tokens for start and end. These special tokens allowed the decoder to recognize the starting and ending of the actual captions.

- Image Feature (ImgFeat) = 2048 vector from Encoder
- Description = cleaned ground truth sentence with insertion of special tokens for start and end
- Build the “Wordtoix” and “IxtoWord” to map between work tokens and unique integer representation
- Find the maximum length of ground truth description and decide the value for MAX_LENGTH_CAPTION
- Each image feature and its caption is converted to a series of training inputs
    - Xt = vector of words represented as integer with 0-pad
    - Yt = expected word prediction as integer
    - Input = ImgFeat + Xt

The figure below shows an example of the mapping and token insertions.

Image 1	Ground Truth caption 1: “A dog is barking outside.”			
	Cleaned + special tokens: “startseq dog is barking outside endseq”			
	Training data Length before equalising = 6			
Image 2	Ground Truth caption 2: “That is a sweater.”			
	Cleaned + special tokens: “startseq that is sweater endseq”			
	Training data Length before equalising = 5			
Max Length of Caption: Parameter value chosen = 8				
SN	LogicalData	ImgFeat	Xt	Yt
1	1	imgF1	startseq	dog
2	1	imgF1	startseq dog	is
3	1	imgF1	startseq dog is	barking
4	1	imgF1	startseq dog is barking	outside
5	1	imgF1	startseq dog is barking outside	endseq
6	2	imgF2	startseq	that
7	2	imgF2	startseq that	is
8	2	imgF2	startseq that is	sweater
9	2	imgF2	startseq that is sweater	endseq
Above word inputs get mapped using the wordtoix data structure and 0 padding				
1	1	imgF1	1 0 0 0 0 0 0 0	11
2	1	imgF1	1 11 0 0 0 0 0 0	12
3	1	imgF1	1 11 12 0 0 0 0 0	13
4	1	imgF1	1 11 12 13 0 0 0 0	14
5	1	imgF1	1 11 12 13 14 0 0 0	9
6	2	imgF2	1 0 0 0 0 0 0 0	21
7	2	imgF2	1 21 0 0 0 0 0 0	12
8	2	imgF2	1 21 12 0 0 0 0 0	23
9	2	imgF2	1 21 12 23 0 0 0 0	9

Figure 7: Pre-processing of text data for decoder

Source for image above: (Lamba, 2018)

CBT dataset:

No clean up or pre-processing was required. The files were used as is while training the GPT-2 model as the story generator model.

Flickr 30k dataset:

No clean up or pre-processing was required. The files were simply used for inference by the object detector to populate the image database.

4.3 Tools and other Resources

The following tools and resources were required to complete this study:

- Limequery
  An open source tool to design and conduct surveys
- Kaggle and Google Colab
  Cloud computing virtual machine platforms. Used the free versions to train the various models.
- Google Drive
  Online storage tool to save the huge amount of images and miscellaneous files. A paid version with 200 GB of storage had to be used for this study.
- Anaconda
  Virtual environment management for Python

5 Implementation

The detailed stage-wise implementation pipeline is shown below.

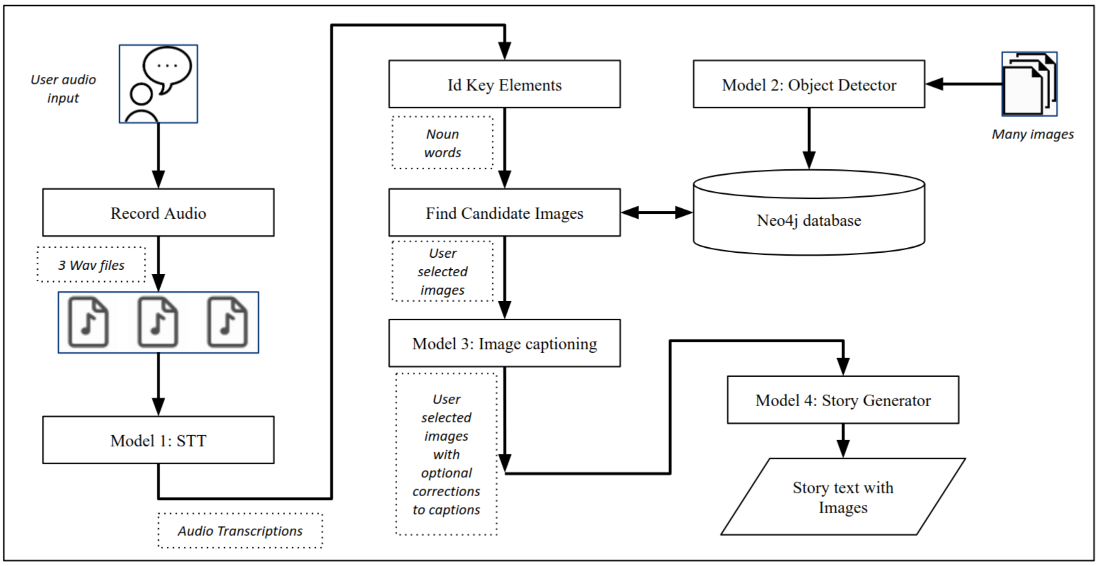


Figure 8: Stage-wise Detailed Block Diagram

The implementation was done using Python-3 and a basic graphical user interface was built using the Tkinter module of Python. The pipeline implementation was broken into stages as described below.

5.1 One-time Setup

First a database had to be created to store which images had which objects. This would allow the system to query the database to retrieve images which matched the objects identified from the users audio input description. Thus, as a one-time exercise, object detection inference was done on many images.

A YOLO-v3 object detector model was used that was pre-trained on the COCO dataset. Thus only the following 80 classes of objects could be identified in this implementation:

'aeroplane', 'apple', 'backpack', 'banana', 'baseball bat', 'baseball glove', 'bear', 'bed', 'bench', 'bicycle', 'bird', 'boat', 'book', 'bottle', 'bowl', 'broccoli', 'bus', 'cake', 'car', 'carrot', 'cat', 'cell phone', 'chair', 'clock', 'cow', 'cup', 'diningtable', 'dog', 'donut', 'elephant', 'fire hydrant', 'fork', 'frisbee', 'giraffe', 'hair drier', 'handbag', 'horse', 'hot dog', 'keyboard', 'kite', 'knife', 'laptop', 'microwave', 'motorbike', 'mouse', 'orange', 'oven', 'parking meter', 'person', 'pizza', 'pottedplant', 'refrigerator', 'remote', 'sandwich', 'scissors', 'sheep', 'sink', 'skateboard', 'skis', 'snowboard', 'sofa', 'spoon', 'sports ball', 'stop sign', 'suitcase', 'surfboard', 'teddy bear', 'tennis racket', 'tie', 'toaster', 'toilet', 'toothbrush', 'traffic light', 'train', 'truck', 'tvmonitor', 'umbrella', 'vase', 'wine glass', 'zebra'

The schema for the database was as shown below. There were two types of Nodes: “Image” and “Object”. There was only one type of relationship: “HAS”. Nodes are represented as circles and the relationship as an arrow.

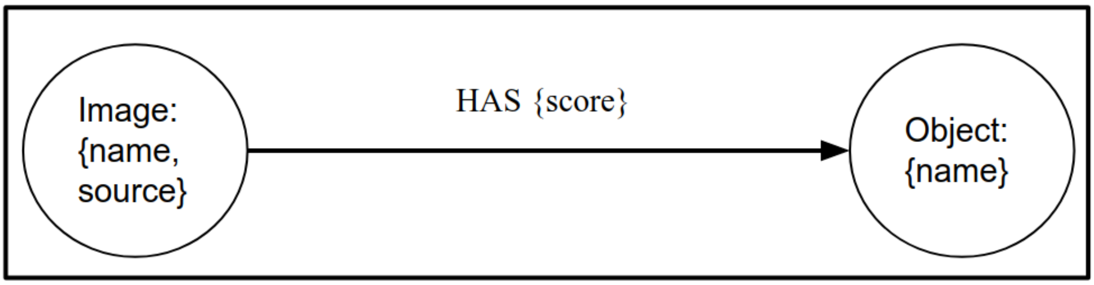


Figure 9: Neo4j schema

The table below explains why this graph schema was designed that way of this graph and the information stored in it:

<table>
<tr><th>Graph element</th><th>Properties</th><th>Description</th></tr>
<tr><td>Node “Image”</td><td>name

source</td><td>Image filename in the “name” property.

The dataset to which image belongs was stored in the “source property”.</td></tr>
<tr><td>Node “Object”</td><td>name</td><td>Name of the label output by object detector stored as the “name” property.</td></tr>
<tr><td>Relationship “HAS”</td><td>score</td><td>The confidence score of the object detector. A higher value indicates more certainty that the object is present in the image.
Range of values: 0 to 1</td></tr>
<tr><td colspan="3">Notes:

1) Approximately 50,000 images loaded into the database. 25k from the COCO-Test dataset and 15k from the Flickr-30 dataset.

2) The image itself still resides outside the database and only meta-data about which image has which object is stored in the graph.

3) Relationship “HAS” was many-to-many type: as the same image could contain multiple objects and the same object could be detected in multiple images.</td></tr>
</table>

Table 2: Neo4j graph elements

For a detailed architecture diagram of the YOLOv3 model, kindly refer to the Appendix.

The source code for the model was taken and adapted from a github repository (Github jbrownlee/keras-yolov3, 2020).

5.2 Pipeline stages – user audio to story output

Stage 1: Capture audio input from user and process to output transcriptions

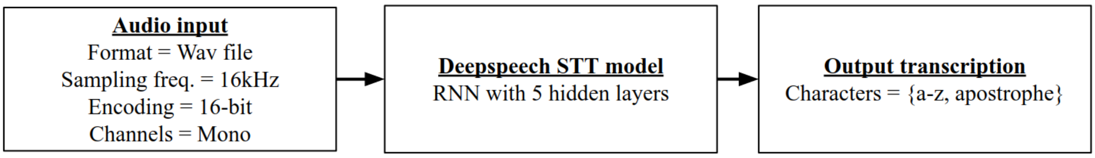


Figure 10: Speech to Text model block diagram

The DeepSpeech model expected a Wav file as an input with the characteristics shown in the first box above. It processed this file and output the transcription using the characters that were part of its vocabulary. This set of characters would have been specified as an "alphabet file" provided to the model as part of its training process. In this study, the output characters were the lowercase English letters and the apostrophe. The pre-trained model version 0.7.3 was used in this study. With careful enunciation of the sentences during the recording process, the word error rate was found to be acceptably low to produce the desired transcriptions in most cases. The user was provided an option to either record the audio as part of the system implementation or to provide three previously recorded Wav files.

The restrictions were that exactly three Wav files had to be provided. Each Wav file could consist of a maximum of 10 seconds of speech, with one complete sentence being spoken per file.

All this functionality could be invoked via a single GUI window which had options to record audio, playback audio, perform inference on the audio files, and finally confirm if the user was happy with the inference.

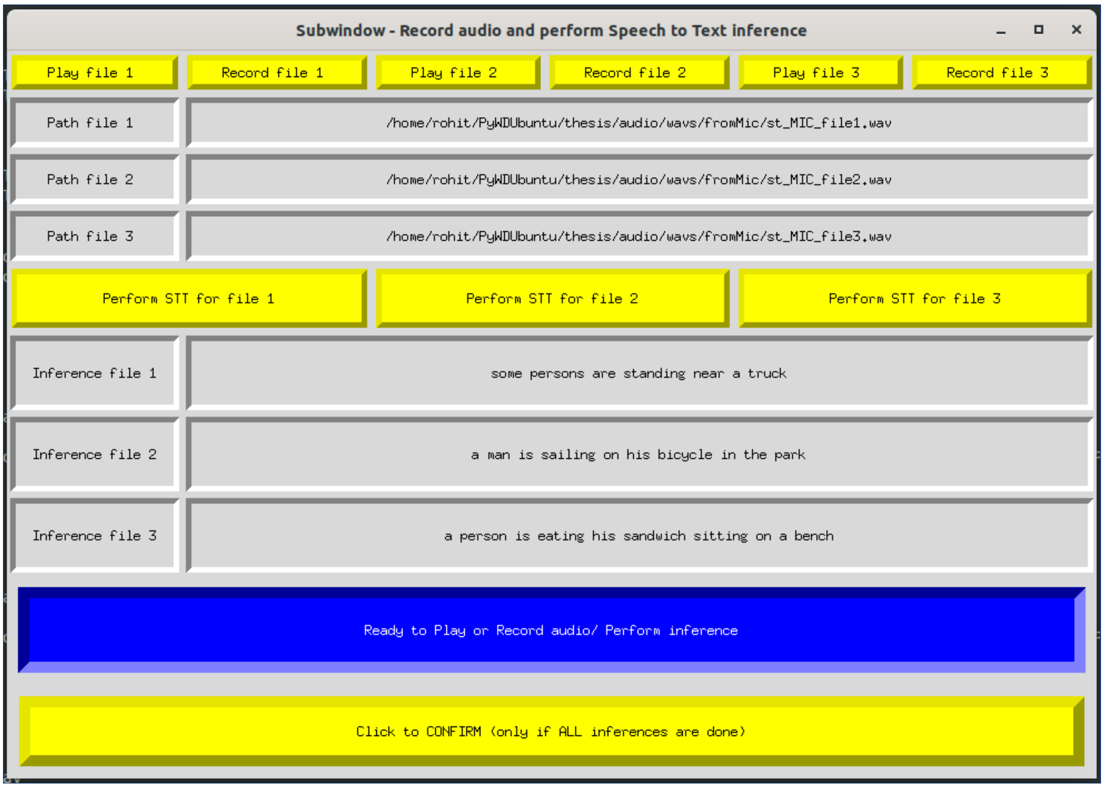


Figure 11: GUI window for STT stage

The transcriptions for each of the Wav files were passed to the next stage for further processing.

```
LOG_LEVEL INFO ::
After MIC INPUT + STT logic execution:
mic_stt_logic_RC = 0
mic_stt_logic_msg = None
mic_stt_module_results = ['some persons are standing near a truck', 'a man is sailing on his bicycle in the park', 'a person is eating his sandwich sitting on a bench']
```

Listing 3: Sample output after Stage 1 - displayed as logging data to console

Stage 2: Identify Key Elements using Parts of Speech (POS) tagging

spaCy (version 2.3.1) “large model” was used to identify the nouns in the input transcriptions. As per the official website for the spaCy project, the large model could handle around 685,000 unique words and was trained on news, comments and blogs from the web. Using the Parts-of-speech Tagger functionality, the nouns were identified by only selecting the tags of type “NN” (Noun, singular or mass), “NNS” (Noun, plural), or ”NNP” (Noun, noun proper singular). These words were then matched against the 80 types of objects available for querying in the Neo4j database and only those words were retained. These words were then presented to the user for selection or deselection via a GUI. The GUI could display at most 30 words per input sentence. In the screenshot below, there were two words presented to the user.

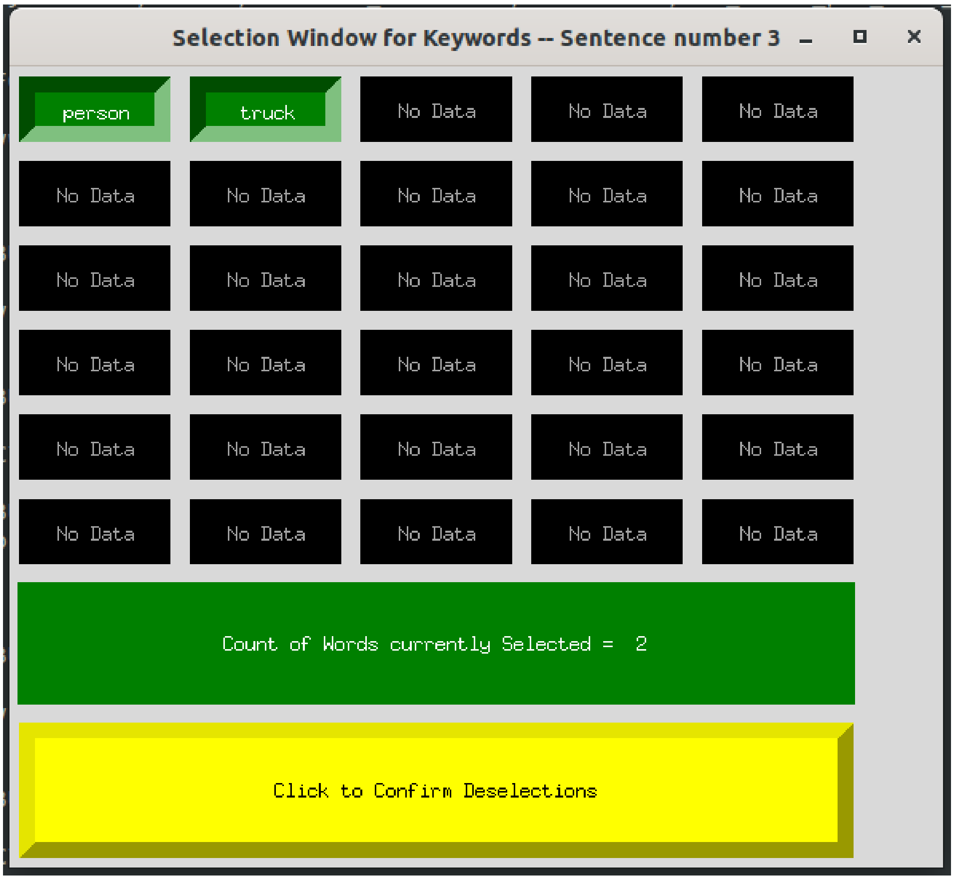


Figure 12: GUI for word selection during Identify Elements stage

Of these 30 possible words, the restrictions imposed were to select a maximum of 3 words per input sentence. The user could deselect all words for a certain sentence if so desired. Thus, the next stage received a maximum of 9 total keywords or elements for processing if the user selected the maximum possible keywords for each input sentence. Or, a minimum of 1 keyword if the user selected only 1 keyword for some input sentence but deselected all keywords for the other two sentences.

Stage 3: Retrieve images via query to Neo4j database

As described earlier, the Neo4j schema allowed the system to query for images containing the objects of interest. These were the key elements passed from the Id Key Elements stage. For each sentence, the query returned a maximum of 20 images where at least one of the objects had been detected with a confidence score greater than 90%. These images were displayed via a GUI window arranged as a grid of image thumbnails. The user was allowed to select or deselect images by clicking the thumbnail image. The restriction imposed was that a maximum of 5 images per sentence could be selected out of the possible 20 images displayed for each sentence. Further, the user could enlarge an image and perform a one-off inference using the same object detector that was used to populate the database. This was to allow the user to verify what objects were detected and then decide whether to retain an image or not. This was a crucial step as it was found there were frequent cases of false-positives where the Neo4j query returned images that did not actually have the object of interest. This was due the object detector itself having populated the wrong information into the database. From an implementation standpoint, this was a static database and therefore the results had to be presented to the user for verification and explicit selection upon inspection.

Figure 13: Close-up of grid selection window for readability

Figure 14: Enlarged image window showing output of detection inference

Out of these 60 possible images, the restrictions imposed were to select a maximum of 5 images per input sentence to pass on to the next stage. The user could deselect all images for a certain sentence if so desired. Thus the next stage received a maximum of 15 total images if the user selected the maximum possible images for each input sentence. Or, a minimum of 1 image if the user selected only 1 image for some input sentence but deselected all images for the other two sentences.

Stage 4: Image captioning

Two different image captioning models were trained and used in this study. This was done to evaluate the effect on the final story generated. Both models had the same encoder which was a Google Inception-v3 model pre-trained on the Imagenet database and was downloaded from the Keras library. But the models differed in their decoders as explained below.

The decoder in Approach 1 was an end-to-end type with merge architecture and without-attention mechanism. But for Approach 2, it was an end-to-end type with inject architecture and with-attention mechanism. The model for with-attention was more complex as it had more RNN units and the embedding dimensions were also slightly larger. This increased the time and effort to train it.

For a detailed architecture diagram of the without-attention decoder, kindly refer to the Appendix.

Decoder Model Parameters		
Parameter	**Model Type**	**Parameter Description**
Vocab_Size	Both	Number of words model can output - also called Vocabulary
Embedding_Dims	Both	Size of the vector representing each word as per embeddings type chosen
Embedding_Matrix_Shape	Both	Shape of the embedding matrix = (VOCAB_SIZE, EMBEDDING_DIMS)
Max_Length_Caption	Both	Maximum length of ground truth description with "startseq" and "endseq"
Unique words in Vocab	Both	Total unique words in all the ground truth captions
Unique words (high frequency)	Both	Total unique words in the model can handle
High Frequency Decision	Both	With attention: occurs > 10 times ; Without Attention: top 5000 by frequency
Number of RNN Units	Both	Number of RNN cells in the model
Attention Size	only With type	Attention span of the model

Table 3: Decoder model parameters and their selection process

Decoder Model Parameters Values		
Parameter	**Approach 1: Without- Attention**	**Approach 2: With-Attention**
Vocab_Size	6758	5001
Embedding_Dims	200	256
Embedding_Matrix_Shape	(6758, 200)	(5001, 256)
Max_Length_Caption	49	52
Unique words in Vocab	24323	28062
Unique words (high frequency)	6757	5000
Number of RNN Units	256	512
Attention Size	NA	10

Table 4: Decoder model parameters for both models compared

From a training standpoint, similar data was used to train models for with and without attention. But there were slight differences as shown in the two tables below.

Data splits for Approach 1: Without-Attention			
Original Source	**Dataset type during training**	**No. of Images**	**Descriptions?**
Coco2017_Train	Train	97k	Yes
Coco2017_Train	Validation	3k	Yes
Coco2017_Val	Test	5k	Yes

Table 5: Datasets used during training for Without-attention model

Data splits for Approach 2: With-Attention			
Original Source	**Dataset type during training**	**No. of Images**	**Descriptions?**
Coco2017_Train	Train	100k	Yes
-	Validation	-	-
Coco2017_Train	Test	5k	Yes

Table 6: Datasets used during training for With-attention model

- Without-attention model was trained on 97,000 images while the with-attention model with 100,000 images.
- For the without-attention model, a validation set of 3,000 images was used to explicitly check overfitting. This could not be done due to run-time constraints for the other model. However, seeing the training losses table for that model, it is highly unlikely that overfitting had occurred.

- The BLEU scores for both models were calculated on 5,000 images not used for the training. Thus the results could be compared from an evaluation standpoint.

For the without-attention type, two models were trained with different training hyperparameters. The one shown as “Model 2” in the table below was selected as it had better losses after 10 epochs. This model was trained further till 18 epochs and was used as the without-attention model. Note: the only aberrant value for loss after 14 epochs could be due to random chance or the incorrect weights file being used. Due to time constraints it was impossible to perform the whole activity again to verify.

Epoch	Model 1 Losses		Model 2 Losses	
	Train	Val	Train	Val
2	3.4819	3.52842	3.1985	3.27492
4	3.2286	3.36663	3.0032	3.17518
6	3.1432	3.29084	2.9315	3.14292
8	3.0858	3.24664	2.8673	3.12330
10	**3.0448**	**3.21695**	**2.8443**	**3.11437**
12	-	-	2.8397	3.11331
14	-	-	2.8013	3.08822
16	-	-	2.7977	3.09369
18	-	-	**2.7704**	**3.08865**

Table 7: Training losses for the without-attention model

The training hyperparameters were as follows:

- Epoch 1 to 13: Batch size=64, Learning Rate=0.001
- Epoch 14 to 15: Batch size=128, Learning Rate=0.001
- Epoch 16 to 18: Batch size=32, Learning Rate=0.0005
- Common parameters: Adam optimizer and Categorical cross-entropy loss

Then, greedy search and beam search with beam widths of 3 and 5 were explored to compare the output captions.

For with-attention type, only one model was trained due to resource and time-out constraints being faced even on the Kaggle cloud computing platform.

Epoch	Train Losses
1	0.676393
4	0.549900
7	0.521033
10	0.501702
13	0.487315
16	0.476580
19	0.468022
21	0.463329
22	0.461324

Table 8: Training losses for the with-attention model

The training hyperparameters were as follows:

- All 22 epochs: Batch size=128,   Learning Rate=0.001
- Adam optimizer and Categorical cross-entropy loss function were used

Then only greedy search was used to output the captions.

Both models thus received the same images from the previous stage. They ran inferences on every image using either beam search (width = 3) for the without-attention model or greedy search for the with-attention model to output the captions. This information was then passed on to the next stage.

The steps to implement the without-attention model was done using an approach found on the internet (Lamba, 2018).

The source code and approach for the with-attention model was taken from a tensorflow tutorial (Tensorflow Tutorials Image Captioning, 2020).

Stage 5: Image captioning display via GUI and optional correction by user

The captions from both models were presented to the user via a GUI window alongside a thumbnail of the image. The user was allowed to correct any captions if so desired. This was done as it was found the models did not always produce good results. Any errors at this stage would adversely affect the downstream processing during the story generation stage. Again, the thumbnails were clickable to select or deselect the images.

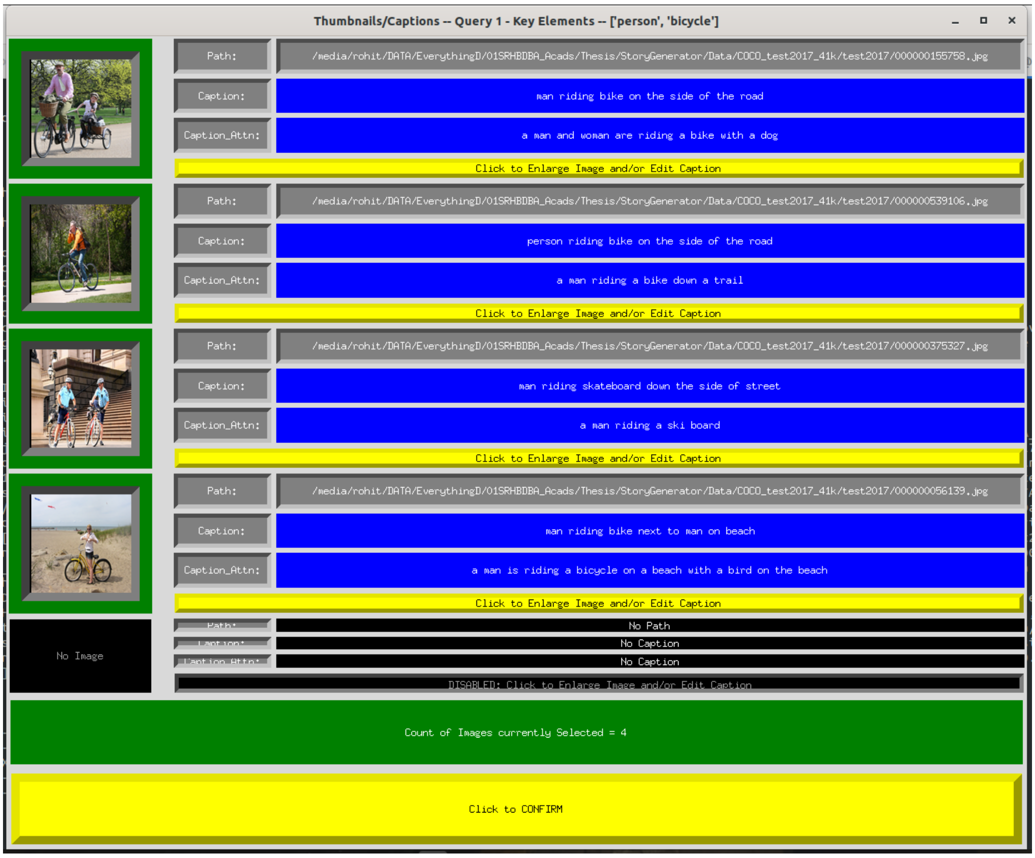


Figure 15: Displaying results of image captioning stage

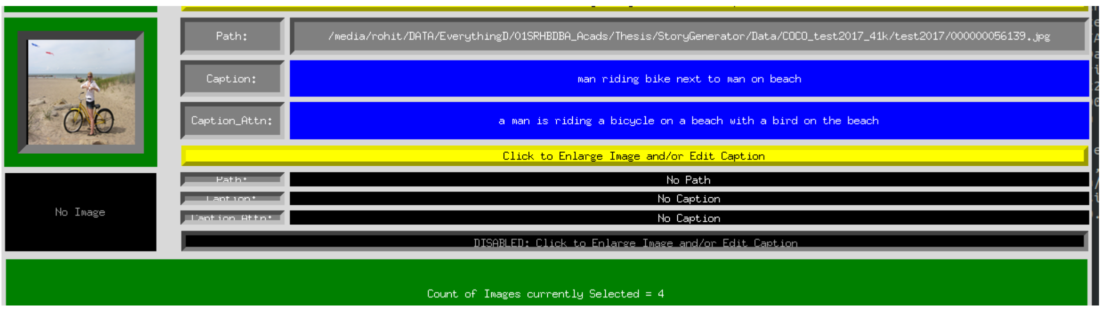


Figure 16: Partial Screenshot of above window for readability

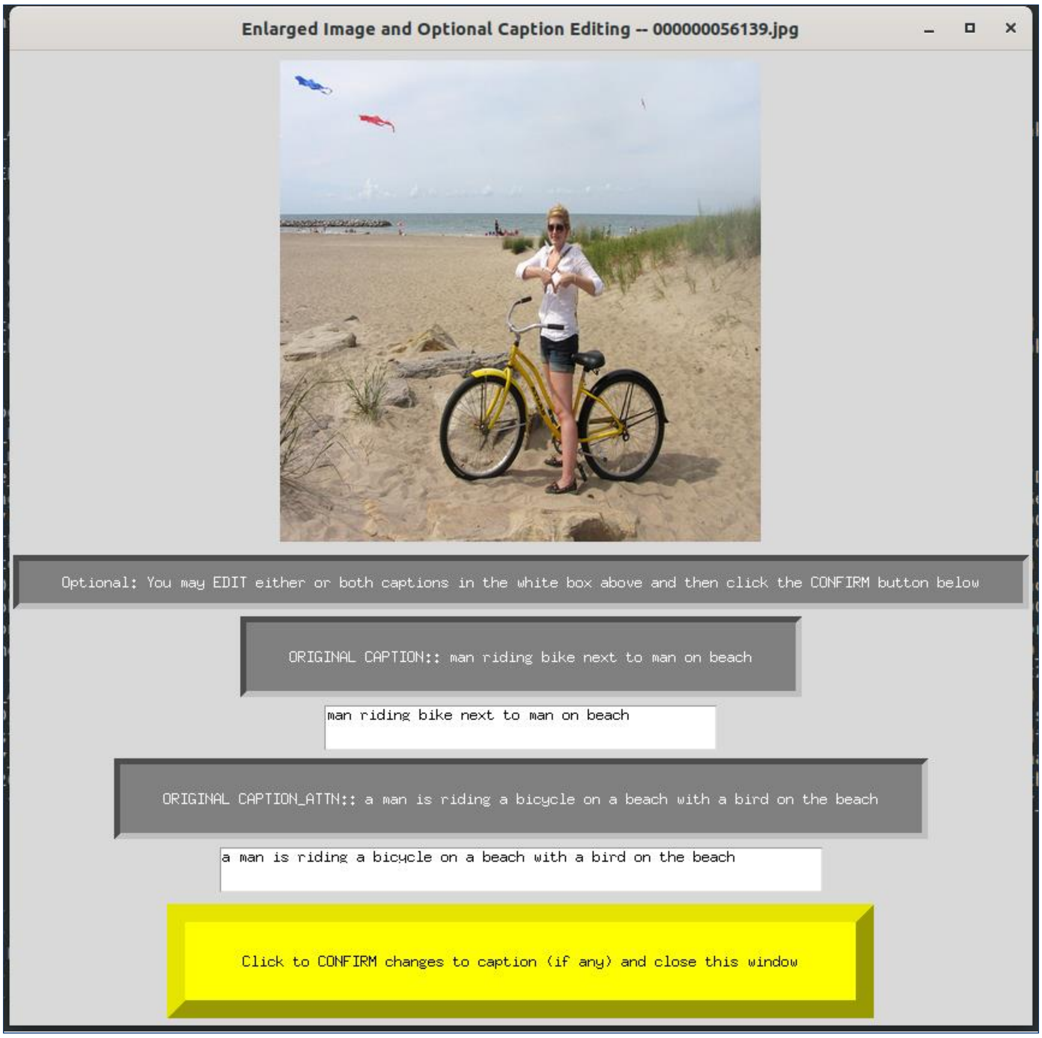


Figure 17: Window to edit captions

The white boxes were to edit the captions while the grey boxes showed the original captions.

No restrictions were placed on the user at this point by allowing the user to retain all the 5 images per input sentence if desired. But the user could deselect any images if desired. Thus, out of the 15 possible input images, both captions for all 15 images could potentially be passed on to the next stage for story generation. These captions could individually be edited depending on how happy the user was with the original captions.

Stage 6: Story generation

The story generation was done using the Generative Pre-Training version 2 (GPT-2) model. From a Python library available via Github, the “medium” sized model consisting of 355 million parameters was downloaded as the baseline model. Then further training was done to fine-tune the model with 11 files from the Children’s Book Dataset (CBT). The aim was to train the model with text that was suitable for children. This was extremely resource intensive and was done for 1500 steps on each input file. The input text files were not cleaned in any way and used as is. Each file varied in size from around 1 MB to 25 MB and had to be trained individually with checkpointing to pick up training from the last checkpoint. The total size of the training data was around 100 MB.

After fine-tuning, the checkpoint files were saved to use it to load the final model used in the story generation process.

Preparation of the seed value for the Story Generator: The data from the previous stage in the pipeline consisted of the images and their captions (with or without correction based on the user behaviour). The seed for the story generation had to be decided somehow. Rather than using just one caption of a unique image and generating the story, the whole aim of this study was to combine up to 3 images and generate a meaningful story around those images. Therefore, considering one image from an input sentence at a time, and then considering one caption for that image, triplets of captions in different permutations were created. Thus, different combinations of 3 captions were taken together as one long seed value for the story generator. Note: only in a situation wherein the user had deselected everything for a particular input audio sentence, the seed would be generated with less than 3 captions and the final story would have less than 3 images. The Evaluation section will show some outputs of the stories with 2 and 3 images - which were used for grading by respondents in the survey

For a given seed input, the output story could be changed by setting the parameters of the story generator. These parameters were:

- “Temperature”: allowed range of values from 0.0 to 1.0
  A higher value would allow the output to be more non-deterministic and thus creative from the perspective of this study.
  This study used values of 0.6 and 0.95.
- “Top_k”: Integer value controlling diversity of output
  Served a similar function as the temperature parameter but works in different manner.
  The study always used a value of 0 as that was the recommended value to allow maximum variability.
- “Length: Parameters setting the number of words output in the story
  The study always used a value of 300. This was chosen arbitrarily as it was felt that the text would not be too long for the survey respondents to read and grade.

No GUI was used in this stage.

The source code for training and running inferernce using GPT-2 model was taken and adapted from a github repository (Github minimaxir/gpt-2-simple, 2020).

6 Evaluation

First, a discussion about the image captioning models and the choice of the exact model are covered here. Then the evaluation of the stories by a group of people laypersons will be shown and discussed.

6.1 BLEU score based evaluation of image captioning models

After training the image captioning models the BLEU score metric was used to evaluate their performance. This was done using the in-built functionality of the Natural Language Tool-kit (NLTK). NLTK is a very popular Python library that is used to process human language data with a broad range of functionalities available. To compute the BLEU score, one needs to provide arrays of the ground-truth text and a model predicted text as shown below:

```
import nltk.translate.bleu_score as nltk_bleu
bleu_score = nltk_bleu.sentence_bleu( [list of GT descriptions] , “the predicted
caption from model” )
```

Listing 4: Code snippet for BLEU score calculation using NLTK

For the without-attention image captioning model, BLEU scores were used to compare greedy search against beam search. For beam search, widths of 3 and 5 were evaluated.

The tables below show the BLEU scores evaluated on 5,000 images not used for training.

Bleu Scores Model 2: Greedy v/s Beam search			
Bleu Scores	**Greedy**	**Beam (3)**	**Beam (5)**
Max	1.0	1.0	1.0
Min	0.0	0.0	0.0
Median	0.6495	0.6431	0.6270
Average	0.6426	0.6385	0.6235
Std. Dev.	0.1629	0.1600	0.1617

Table 9: BLEU score summary statistics for without-attention model with Greedy vs Beam search

Key takeaways from the above table:

- Average score for greedy search was the highest at 0.6426
- Average score for beam search with beam width of 3 was slightly poorer than for greedy search
- Average score for beam search with beam width of 5 was the worst
- In all three cases, the median value is almost the same as the average

Based on the above observations, it was possible to use either greedy search or beam search with beam width of 3

Bleu Scores by Bins – frequency comparison – 5k data points Model 2 – 18 Epochs – Comparison Greedy / Beam Search									
	Greedy			Beam Search (Width = 3)			Beam Search (Width = 5)		
Score Bin	Count	Rel. Freq	Cum. Freq	Count	Rel. Freq	Cum. Freq	Count	Rel. Freq	Cum. Freq
0.0 – 0.1	2	0.00	0.00	4	0.00	0.00	6	0.00	0.00
0.1 – 0.2	12	0.00	0.00	9	0.00	0.00	14	0.00	0.00
0.2 – 0.3	81	0.01	0.02	91	0.02	0.02	112	0.03	0.03
0.3 – 0.4	169	0.05	0.07	272	0.05	0.07	323	0.06	0.09
0.4 – 0.5	609	0.12	0.19	590	0.12	0.19	661	0.13	0.22
0.5 – 0.6	988	0.20	0.39	999	0.20	0.39	1031	0.21	0.43
0.6 – 0.7	959	0.23	0.62	1235	0.25	0.64	1201	0.24	0.67
0.7 – 0.8	1028	0.20	0.83	1008	0.21	0.85	964	0.19	0.86
0.8 – 0.9	570	0.12	0.94	565	0.11	0.96	495	0.10	0.96
0.9 – 1.0	230	0.05	1.00	189	0.04	1.00	159	0.04	1.00
Total	4648	-	-	4962	-	-	4966	-	-

Table 10: Bin-wise comparison of BLEU scores for without-attention model with Greedy vs Beam search

Key takeaways from the above table:

- More data points could be scored successfully for beam search (4962) vs greedy search (4648)
- Most of additional approximately 300 data points were scored with a reasonably high score in the 0.6-0.7 bin.

Therefore, for the without-attention model, it was decided to use beam search with beam width of 3 in the overall implementation.

Similarly, BLEU scores were computed for **the with-attention image captioning model** as well. But here only greedy search was used. No exploration of beam search was attempted due to limitation of resources and inference output time.

The tables below show the BLEU scores evaluated on 5,000 images not used for training.

Attention Model after 22 epochs	
Bleu Scores	**Greedy**
Max	1.0
Min	0.0
Median	0.6110
Average	0.6000
Std. Dev.	0.1800

Table 11: Bleu score summary statistics for with-attention model with greedy search

Key takeaways from the above table:

- The average score at 0.6000 was substantially poorer than for the without-attention model with beam search.
- The standard deviation was also higher (0.18) compared to the value for without-attention beam search with width of three (0.16)
- The median value at 0.6110 was somewhat higher than the average at 0.6000

This meant that this model had a wider range of scores on the 5,000 data points tested and a larger proportion got scores that were far below the median.

Bleu Scores by Bins – frequency comparison – 5k data points			
Greedy Search			
	Attention Model – 22 epochs		
Score Bin	**Count**	**Rel. Freq**	**Cum. Freq**
0.0 – 0.1	35	0.01	0.01
0.1 – 0.2	94	0.02	0.03
0.2 – 0.3	154	0.03	0.06
0.3 – 0.4	381	0.08	0.14
0.4 – 0.5	724	0.14	0.28
0.5 – 0.6	997	0.20	0.48
0.6 – 0.7	1099	0.22	0.70
0.7 – 0.8	867	0.17	0.87
0.8 – 0.9	468	0.09	0.96
0.9 – 1.0	167	0.03	1.00
Total	4986	-	-

Table 12: Bin-wise comparison of BLEU scores for with-attention model with greedy search

Key takeaways from the above table:

- For scores below 0.03, the cumulative frequency is 0.06. This is far double the value for the other model.
- This confirmed the suspicion that there was a higher prevalence of low scored data points.

Despite this, as it was the only model available and further training was not possible, it was used as the second image captioning model in the study.

6.2 Designing and conducting the Survey

The final output consisted of a story of 300 words accompanied by up to 3 images. It was necessary to evaluate the outputs on the basis of coherence, relevance and suitability. For this, a survey was conducted on a group of people. These were selected by convenience sampling by approaching fellow students, the professors and family relatives of the author. However, the author was mindful of potential bias due to familiarity of the respondents with the author. Explicit instructions were given to the respondents to grade the stories with complete objectivity and not shy from grading poorly if they felt so.

Using the survey, this study evaluated the effect of the following changes in the scores given by the respondents:

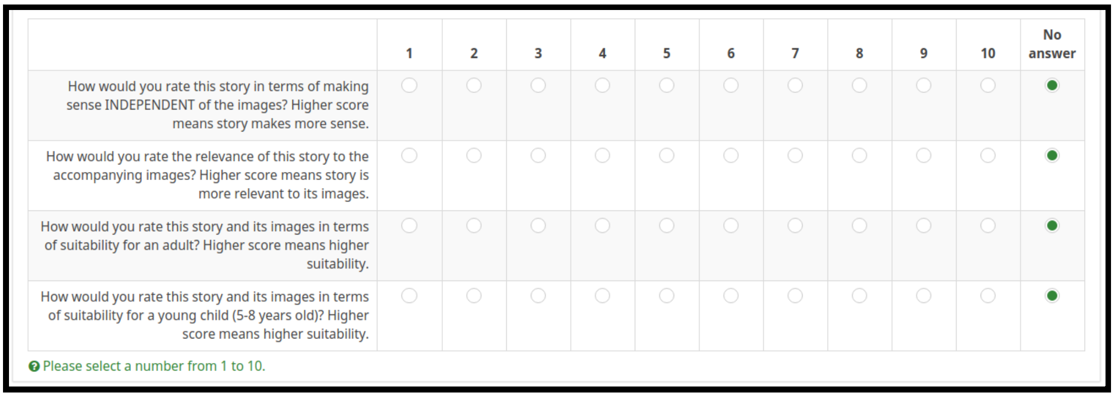
- Using a temperature value of 0.95 vs 0.60
- Two different stories: one with 3 images and other with 2 images
- Image captioning using the with-attention vs no-attention models
- Using the original captions output by the image captioning models vs correcting the captions to match the images more accurately

Thus there were 10 stories in total for the survey. The first five stories were generated for the 3 images case and the rest for the 2 images case. The table below summarizes these aspects:

Story No.	Three or Two images	Temperature	Image Caption Model Type	Captions corrected?
1	Three	0.95	No attention	No
2		0.60	No attention	No
3		0.95	No attention	Yes
4		0.95	With attention	No
5		0.95	With attention	Yes
6	Two	0.95	No attention	No
7		0.60	No attention	No
8		0.95	No attention	Yes
9		0.95	With attention	No
10		0.95	With attention	Yes

Table 13: Model parameter settings used to generate the stories used in survey

A respondent saw images followed by the story text. Below that were four questions, each of which were to be scored on a 10 point scale with a value of 10 being the highest and 1 being the lowest score.



	1	2	3	4	5	6	7	8	9	10	No answer
How would you rate this story in terms of making sense INDEPENDENT of the images? Higher score means story makes more sense.	○	○	○	○	○	○	○	○	○	○	●
How would you rate the relevance of this story to the accompanying images? Higher score means story is more relevant to its images.	○	○	○	○	○	○	○	○	○	○	●
How would you rate this story and its images in terms of suitability for an adult? Higher score means higher suitability.	○	○	○	○	○	○	○	○	○	○	●
How would you rate this story and its images in terms of suitability for a young child (5-8 years old)? Higher score means higher suitability.	○	○	○	○	○	○	○	○	○	○	●

Please select a number from 1 to 10.

Figure 18: Survey questions - common in all stories

The table below shows the questions and the reason for framing them that way.

SN	Question asked	Reason
1	How would you rate this story in terms of making sense INDEPENDENT of the images? Higher score means story makes more sense.	Gauge the coherence of the story text. Without explicitly saying so, this score would serve as an overall proxy for the grammar, sentence structure, choice of words, etc in the story text.
2	How would you rate the relevance of this story to the accompanying images? Higher score means story is more relevant to its images.	Gauge the relevance of the images to the story text.
3	How would you rate this story and its images in terms of suitability for an adult? Higher score means higher suitability.	Gauge the suitability of the story and images for an adult.
4	How would you rate this story and its images in terms of suitability for a young child (5-8 years old)? Higher score means higher suitability.	Gauge the suitability of the story and images for the actual intended audience - a young child.

Table 14: Survey questions - wording and explanation

The figures below show one story with 2 images and another one with 3 images. The respondent saw the images, followed by the story text. At the end of the screen were the four questions with radio buttons to mark their scores.

Figure 19: Survey story sample – 3 image variant

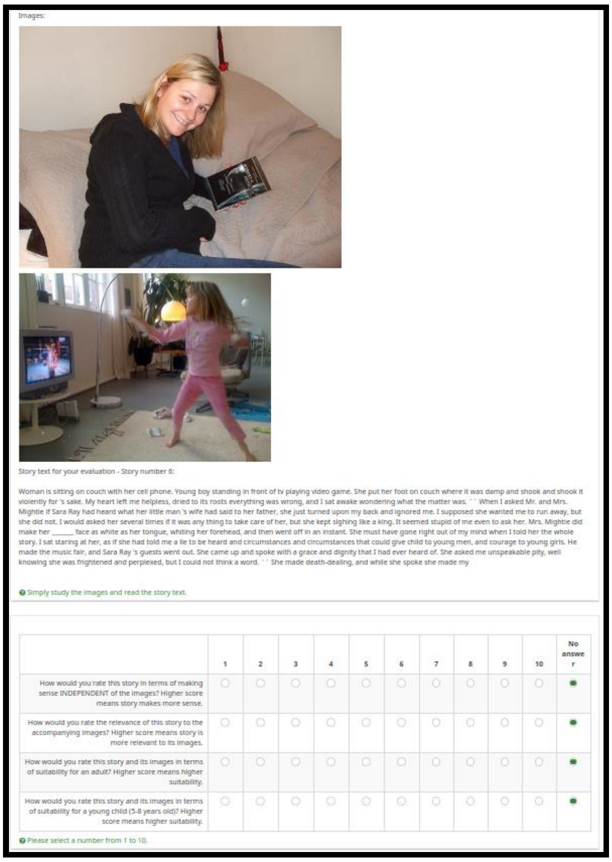


Figure 20: Survey story sample – 2 image variant

For the story with two images, we shall now see the text for two different stories generated for the same seed value. In both cases, the seed value input to the story generator was: "*Woman is sitting on couch with her cell phone. Young boy standing in front of tv playing video game.*".

Using different parameter settings (e.g. temperature) different stories were created. Here story 6 was generated with a temperature of 0.95 while story number 7 was with 0.60.

Story number 6:

“Woman is sitting on couch with her cell phone. Young boy standing in front of tv playing video game. She put her foot on couch where it was damp and shook and shook it violently for 's sake. My heart left me helpless, dried to its roots everything was wrong, and I sat awake wondering what the matter was. `` When I asked Mr. and Mrs. Mightie if Sara Ray had heard what her little man 's wife had said to her father, she just turned upon my back and ignored me. I supposed she wanted me to run away, but she did not. I would asked her several times if it was any thing to take care of her, but she kept sighing like a king. It seemed stupid of me even to ask her. Mrs. Mightie did make her _______ face as white as her tongue, whiting her forehead, and then went off in an instant. She must have gone right out of my mind when I told her the whole story. I sat staring at her, as if she had told me a lie to be heard and circumstances and circumstances that could give child to young men, and courage to young girls. He made the music fair, and Sara Ray 's guests went out. She came up and spoke with a grace and dignity that I had ever heard of. She asked me unspeakable pity, well knowing she was frightened and perplexed, but I could not think a word. `` She made death-dealing, and while she spoke she made my”.

Story number 7:

“Woman is sitting on couch with her cell phone. Young boy standing in front of tv playing video game. You know what music I listened to yesterday. You have been there to-day, I hope. You are not there to-night. I am laughing to think of you. Come in, and we will have a good talk, and Mrs. King will see to you . " I came in and sat down on the edge of the sofa. The young lady was sitting on the sofa, her white hands clasped to her knees. She had a nice, calm face, with a certain, benevolent, maternal smile of which the young lady was always proud. I sat like a boy, and I was all the time a little girl. I did not mind it. I was a little boy. I had not been a boy so long. I had not cared, but I was not a little girl. I was a little girl. I looked up at the big, tall, pale-blue eyes that had been watching me from the sofa. I felt as if I had been looking down at a mirror. The black eyes were large and dark and strong and deep. They were frowning at me. I had been in a hurry of the morning to get ready for the day 's work, but I had never been at work since the morning. I was glad to get my breakfast, and I went to work. The whole world was full of men and women and children who".

Single sentence seed story:

Note that in the stories created above, the seed value consisted of the image captions of two images combined into two consecutive sentences. However, below are two more stories with only one sentence being provided as the seed value.

With seed value: "*Woman is sitting on couch with her cell phone.*".

Story output:

"Woman is sitting on couch with her cell phone. She can be annoying -- she and her fashion. I wonder if she will have any idea -- no posture, or attitude whatever. Any way she is, her weight is such a troublesome thing to me. Every fine and dainty thing she asks me to lay down among it -- feet, arms, or books that I did not mind because I thought I would future for a horse, if I knew how to ride. She is a good advocate. Nothing frustrates her like my blemishes I have worn down my beard -- it was so smooth and soft, and she was so kind to me with them -- now she looks so stern and nosy. But she is never hurt herself -- but what do you think of it? I could hate her . " Anne closed her cell phone and looked out on the plain, brooding sunshine. Plucked wind and heather in the clear air dovetailed in her eye as she ran in and sat down on the bare mattress in the beach. The weather rose suddenly, as a wave lifted her out of her bed ; for it had taken up the denim with its corks, covered her with thereabouts ... and her hair against the damp yell had covered her with the gaud of sunlight. Being a sea-stalk, excited, and sore, she felt tired to death. She stole her seat to the old bed. `` I wanted to see her tomorrow,"

With seed value: "*Young boy standing in front of tv playing video game.*".

Story output:

“Young boy standing in front of tv playing video game. Sammy broke his way down up to his room and devoured everything in to take the dare. Then he began to sing with a beaming voice. `` Let the boy go and play with Reddy Fox ! " `` What da -- forget it, Sammy Brown ! " cried Reddy Fox. `` Sammy Brown is a man that can not be leavin ' the Green Meadows alone . " That was not to Sammy Brown, though he knew it. They were all afraid, even the ones to his face like Sammy. Sammy had been looking out for Reddy Fox for ages. It was never been his custom to do so, because he commanded the younger Sams in Old Mother Ned 's guesthouse. He used to climb into a box, and do whatever was necessary to rot down and earn in the hole he saw through the window. I do not doubt that when Sammy Brown made that cruel claim and threw him in full flight, that he was very followed. This definitely did not show, though! As he was dismounting this people, Reddy Fox pulled down the rabbit and held it up in his voice. He would heard Sammy Brown 's claim that he would be one of the mustangs when he grew up. Sammy was ever so much bigger than alive, but he was not such an easy-going fellow as Sammy Brown, so he hardly dared to give him such”

6.3 Survey results

Of the 22 respondents approached, only 12 filled the survey completely, 8 had filled it partially, and 2 did not start the survey itself. The table below summarises the scores given for each of the 4 questions against the 10 stories in the survey.

Story/ Question No.	Generation settings and the Scores
S1	**Images=3, Temp=0.95, No-attention, No correction**
Q1	Actual scores = 4, 2, 4, 1, 2, 3, 1, 7, 1, 1, 3, 3 Average score = 2.67
Q2	Actual scores = 4, 2, 2, 2, 1, 3, 1, 4, 3, 2, 4, 2 Average score = 2.50
Q3	Actual scores = 6, 2, 4, 2, 5, 4, 1, 6, 1, 2, 8, 2 Average score = 3.58
Q4	Actual scores = 4, 1, 6, 1, 5, 3, 1, 3, 1, 1, 6, 2 Average score = 2.83
S2	**Images=3, Temp=0.60, No-attention, No correction**
Q1	Actual scores = 3, 1, 1, 1, 1, 2, 2, 1, 3, 1, 2, 4 Average score = 1.83
Q2	Actual scores = 3, 1, 1, 1, 1, 2, 1, 1, 3, 1, 1, 2 Average score = 1.50
Q3	Actual scores = 4, 2, 4, 1, 1, 2, 1, 1, 3, 1, 7, 2 Average score = 2.42
Q4	Actual scores = 3, 1, 4, 1, 1, 1, 1, 1, 3, 1, 5, 2 Average score = 2.00
S3	**Images=3, Temp=0.95, No-attention, With correction**
Q1	Actual scores = 3, 2, 6, 1, 1, 2, 1, 2, 8, 1, 5, 5 Average score = 3.08
Q2	Actual scores = 3, 2, 3, 1, 1, 2, 3, 1, 4, 2, 2, 2 Average score = 2.17
Q3	Actual scores = 3, 3, 5, 1, 1, 2, 1, 3, 4, 2, 8, 3 Average score = 3.00
Q4	Actual scores = 3, 1, 5, 1, 1, 2, 1, 4, 1, 1, 2, 2 Average score = 2.00
S4	**Images=3, Temp=0.95, With-attention, No correction**

Q1	Actual scores = 3, 1, 3, 1, 1, 2, 1, 1, 8, 1, 3, 2 Average score = 2.25
Q2	Actual scores = 3, 1, 1, 1, 1, 2, 1, 2, 4, 1, 1, 3 Average score = 1.58
Q3	Actual scores = 3, 1, 4, 1, 1, 2, 1, 1, 4, 1, 8, 1 Average score = 2.33
Q4	Actual scores = 3, 1, 5, 1, 1, 1, 1, 1, 1, 1, 1, 1 Average score = 1.50
S5	**Images=3, Temp=0.95, With-attention, With correction**
Q1	Actual scores = 3, 2, 5, 1, 1, 2, 1, 4, 9, 1, 8, 2 Average score = 3.25
Q2	Actual scores = 3, 2, 2, 1, 1, 2, 5, 3, 7, 2, 1, 1 Average score = 2.50
Q3	Actual scores = 3, 2, 8, 1, 1, 2, 1, 1, 6, 2, 8, 3 Average score = 3.25
Q4	Actual scores = 3, 1, 2, 1, 1, 2, 1, 2, 2, 1, 1, 1 Average score = 1.50
S6	**Images=2, Temp=0.95, No-attention, No correction**
Q1	Actual scores = 4, 2, 6, 1, 3, 2, 2, 6, 9, 1, 9, 5 Average score = 4.17
Q2	Actual scores = 4, 1, 4, 1, 4, 2, 1, 5, 8, 1, 1, 2 Average score = 2.83
Q3	Actual scores = 3, 1, 6, 1, 3, 2, 2, 8, 9, 1, 8, 3 Average score = 3.92
Q4	Actual scores = 3, 1, 2, 1, 4, 1, 1, 7, 1, 1, 1, 1 Average score = 2.00
S7	**Images=2, Temp=0.60, No-attention, No correction**
Q1	Actual scores = 3, 2, 6, 2, 3, 2, 2, 6, 9, 1, 8, 3 Average score = 3.92
Q2	Actual scores = 3, 1, 7, 1, 3, 2, 1, 5, 7, 2, 1, 2 Average score = 2.92

Q3	Actual scores = 3, 2, 7, 1, 2, 2, 1, 6, 1, 2, 8, 2 Average score = 3.08
Q4	Actual scores = 3, 1, 7, 1, 2, 2, 1, 6, 7, 1, 1, 2 Average score = 2.83
S8	**Images=2, Temp=0.95, No-attention, With correction**
Q1	Actual scores = 3, 2, 4, 1, 2, 2, 2, 7, 10, 1, 9, 3 Average score = 3.83
Q2	Actual scores = 3, 2, 2, 1, 1, 2, 4, 3, 8, 2, 1, 2 Average score = 2.58
Q3	Actual scores = 3, 2, 4, 1, 1, 2, 2, 7, 8, 2, 8, 2 Average score = 3.50
Q4	Actual scores = 3, 1, 4, 1, 1, 2, 1, 7, 1, 2, 3, 1 Average score = 2.25
S9	**Images=2, Temp=0.95, With-attention, No correction**
Q1	Actual scores = 3, 1, 3, 1, 2, 1, 2, 5, 10, 1, 9, 2 Average score = 3.33
Q2	Actual scores = 3, 1, 1, 1, 2, 1, 1, 4, 5, 1, 1, 1 Average score = 1.83
Q3	Actual scores = 3, 1, 5, 1, 2, 1, 2, 6, 9, 1, 8, 1 Average score = 3.33
Q4	Actual scores = 3, 1, 5, 1, 2, 1, 1, 6, 1, 1, 2, 1 Average score = 2.08
S10	**Images=2, Temp=0.95, With-attention, With correction**
Q1	Actual scores = 3, 2, 4, 1, 4, 3, 1, 4, 8, 1, 9, 2 Average score = 3.50
Q2	Actual scores = 4, 2, 2, 1, 4, 1, 1, 4, 9, 2, 1, 1 Average score = 2.67
Q3	Actual scores = 3, 1, 8, 1, 4, 2, 1, 5, 8, 2, 8, 2 Average score = 3.75
Q4	Actual scores = 3, 1, 4, 1, 1, 2, 1, 5, 1, 2, 2, 1 Average score = 2.00

Table 15: Survey responses - Scores for each question

The results regarding the objectives of the survey are now covered.

Using a temperature value of 0.95 vs 0.60

The chart below shows the average scores for the stories generated with all things being equal except the temperature values.

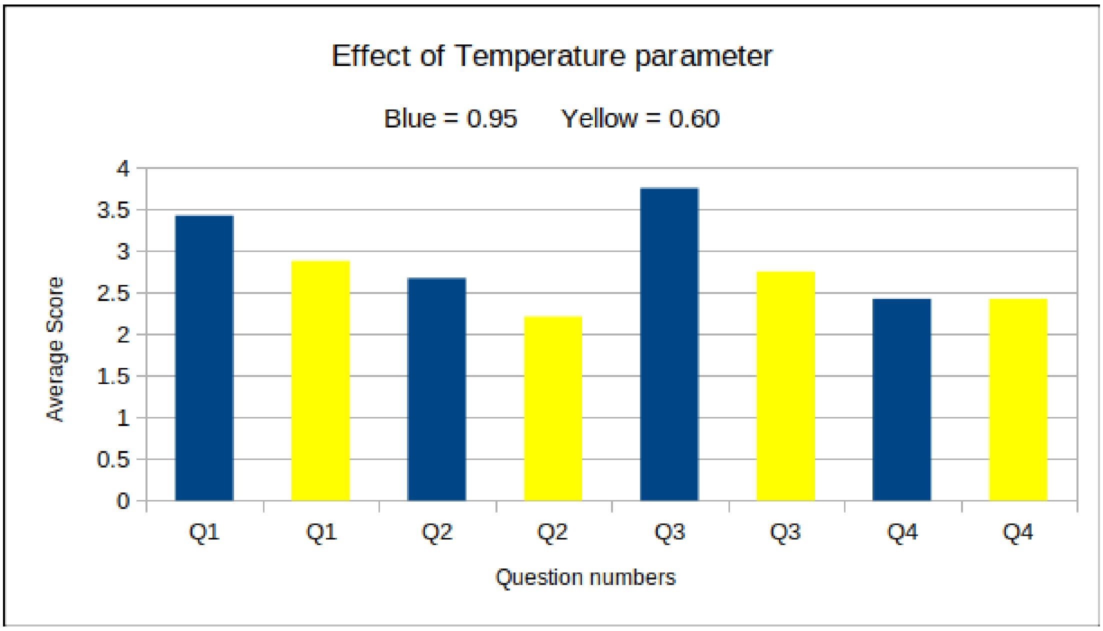


Figure 21: Results - effect of Temperature parameter

We see that except for the question about suitability for a young child (i.e. question 4) generally the scores were better when a higher temperature was used.

Two different stories: one with 3 images and other with 2 images

Excluding the data points for the temperature of 0.60, the chart below shows the average scores for the stories generated with all things being equal except the number of images.

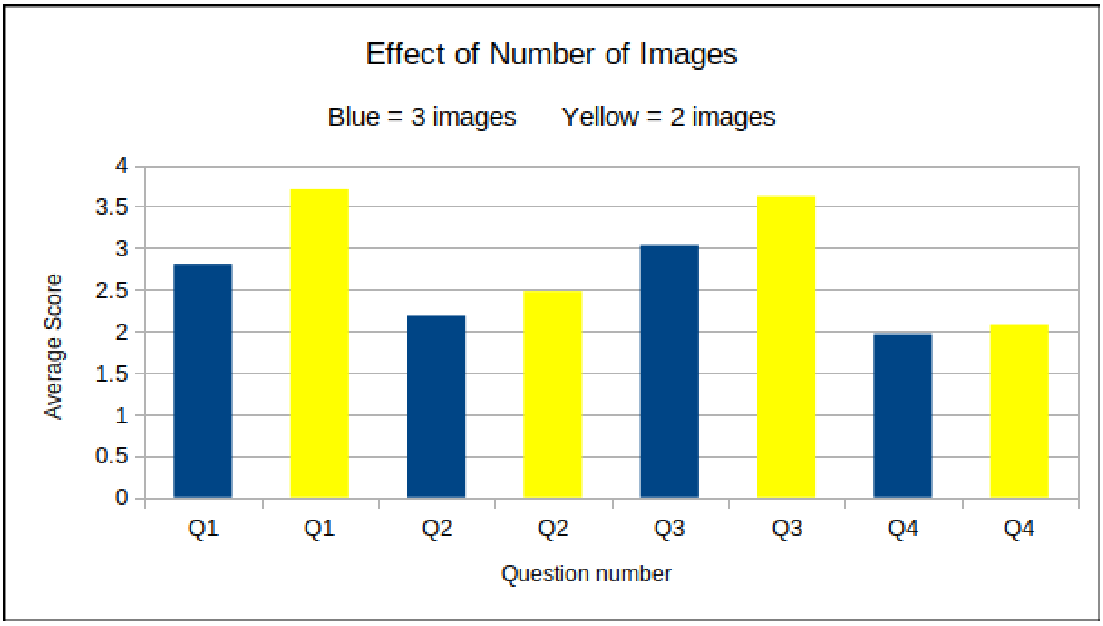


Figure 22: Results - effect of Number of Images in story

We see that in all cases, generally the scores were better when fewer images were used.

Image captioning using the with-attention vs no-attention models

Excluding the data points for the temperature of 0.60, the chart below shows the average scores for the stories generated with all things being equal except the type of image caption model used.

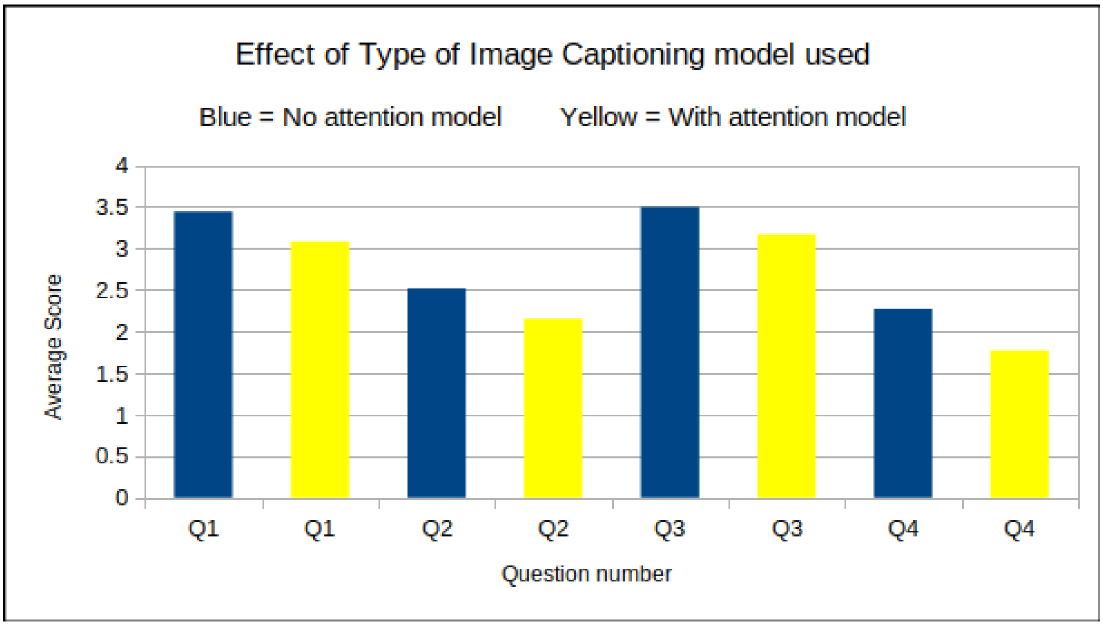


Figure 23: Results - effect of type of Image Captioning model

We see that in all cases, generally the scores were better for the model without attention.

Using the original captions output by the image captioning models vs correcting the captions to match the images more accurately

Excluding the data points for the temperature of 0.60, the chart below shows the average scores for the stories generated with all things being equal except whether the image captions were corrected manually or not.

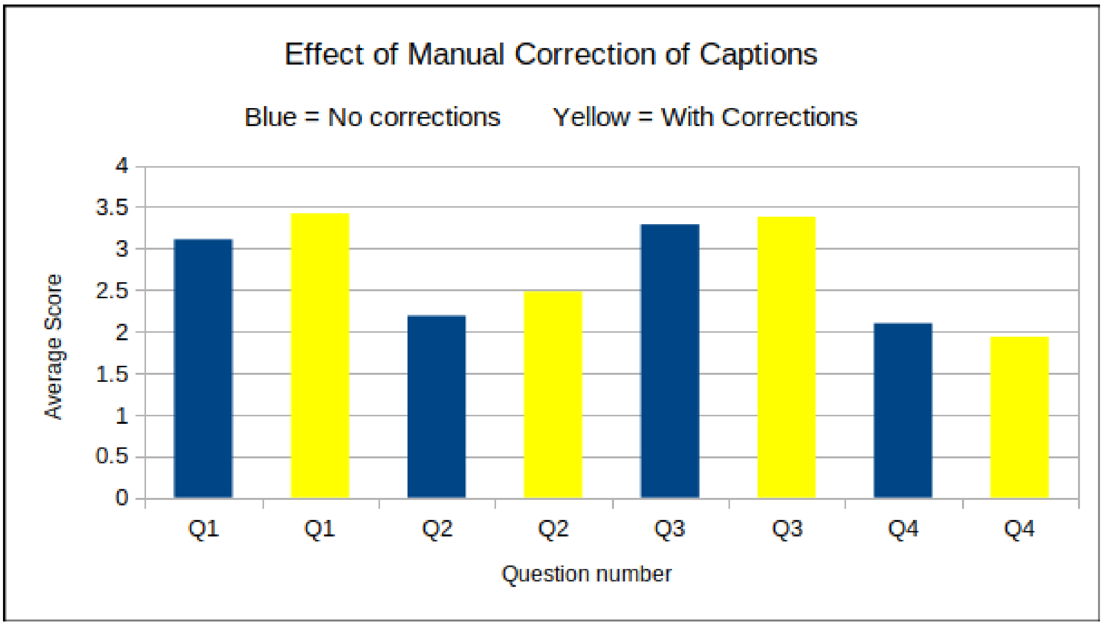


Figure 24: Results - effect of Manual Correction of the Captions

We see that except for the question about suitability for a young child (i.e. question 4), generally the scores were better when corrections were made to the captions.

7 Conclusion

7.1 Findings

Based on the results and evaluation of the overall implementation, the following conclusions were drawn:

- Overall, the respondents did not feel that the stories generated were suitable for young children. Most of the respondents gave very low scores to question 4 of the survey.
- Overall, the respondents did not feel that stories generated were suitable for adults as well. However, the scores for this question 3 in the survey were consistently higher than that for question 4. Therefore, it seems they felt that the stories were more suitable for adults.
- The “medium” sized model of GPT-2 did not seem to do a good job for this use-case implementation, despite some fine-tuning being performed.
- Using a high temperature parameter while generating stories was important as those stories got better scores.
- When the desired story had lesser images (2 instead of 3), the respondents felt the stories were better overall.
- Manual correction of the captions and then generating stories caused limited improvements in the scores of the survey.
- In the image captioning model, despite the with-attention model being larger and more complex, there was no real improvement in the quality of stories output. If fact, contrary to the expectations prior to the study, the without-attention model had better survey scores.
- The image captioning models had BLEU scores around 0.6 for with and without attention models. But it was found that the models did not always generate suitable captions.

- The current implementation pipeline and models were not good enough to allow automatic flow of information. It was deemed crucial to have manual intervention via GUIs. This was at multiple stages: selection of keywords, selection of images from database, and to verify the captions generated.
- It was found that even if a selected image met the specifications as described by the user's audio input, there was no guarantee that the image captioning model would output a caption description that matched whatever the user actually described.
- The survey was only done for stories generated with multiple sentence seed values (either two or three sentences). However, by generating stories with one caption only as a single sentence seed value, there was a better chance of a story being coherent and making sense. However, this was just explored as a one-off and not as part of the survey.

7.2 Future scope

Based on the overall results and findings, there were several areas that could be addressed in the future:

- Much more training could be done for the with-attention model. Both in terms of the training data and the number of epochs.
- Use of beam search method for the with-attention model may allow better caption generation.
- The database of images only stored the object classes detected in the image. That is, only the nouns of the input audio sentences were of use during image retrieval. There was no other type of information like the action being performed (verbs), relative arrangement of objects (prepositions), etc. that were captured into the database. If a method could be found to capture

more detailed information, it could be possible to retrieve more suitable images in that pipeline.

- The image database consisted of only around 50,000 images and the author had to perform multiple trials to specify an input that found matches to proceed with the full pipeline being executed. However, this was a matter of simply loading many more images and could be tackled relatively easily with more storage space being available.
- From an implementation standpoint, the limited programming skills of the author did not allow for more than the first 20 images to be displayed in the pipeline GUI. Thus for a particular query to the database, only the first 20 hits could be used. This was not a show-stopper but an area of improvement nonetheless.
- The survey was deemed long and slightly cumbersome as per the author's viewpoint. Therefore, the number of stories that could be evaluated were only limited to 10. A more robust sample size of stories and larger number of respondents may have changed the overall results mentioned in this study. However, even if this was done, the author does not feel that with the current implementation the scores would be markedly improved.

7.3 Contributions

- The concept of the application of neural networks for such a use-case: As explained in the motivation section, the idea came to the author based on observations of how the pandemic had affected people. And then, what could be done to alleviate some aspect of people's trials in such circumstances.

- The design of the pipeline architecture and the choice of the neural network models.
- Implementing a GUI to allow an easy method for user interaction during the pipeline execution:
  Initially the implementation was planned by having each stage output files. The user would have to manually edit the files and then run each stage one at a time. This was deemed too cumbersome and hence the GUI had to be implemented.
- Conducting a survey to get an objective metric for the overall output quality:
  As per the guidance and inputs from the supervisors, the author was always told to be mindful of how the evaluation of the overall goal should be done. Thus a decision was made to conduct a survey as described in the sections above.

7.4 Learning's from the study

Several concepts had been applied to be to complete the thesis work and the author felt it has had positive impacts on several fronts: programming skills, planning and designing a pipeline, and an appreciation of independent evaluation via a survey.

The author provided regular feedback to the thesis supervisors via presentations and that helped track progress and to eventually complete the report in a short timeframe.

8 Bibliography

Wang, H., Raj, B., & Xing, E. (2017). On the Origin of Deep Learning. ArXiv, abs/1702.07800.

Nassif, A.B., Shahin, I., Attili, I.B., Azzeh, M., & Shaalan, K. (2019). Speech Recognition Using Deep Neural Networks: A Systematic Review. IEEE Access, 7, 19143-19165.

Hannun, A.Y., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., & Ng, A. (2014). Deep Speech: Scaling up end-to-end speech recognition. ArXiv, abs/1412.5567.

Kim, J.Y., Liu, C., Calvo, R., McCabe, K., Taylor, S., Schuller, B., & Wu, K. (2019). A Comparison of Online Automatic Speech Recognition Systems and the Nonverbal Responses to Unintelligible Speech. ArXiv, abs/1904.12403.

Deepspeech Project Website. 2020. “Welcome to DeepSpeech’s documentation!” medium.com. https://deepspeech.readthedocs.io/en/v0.7.3/ (Accessed on 03 June 2020)

DeepSpeech by Mozilla. 2020. "DeepSpeech FAQ". https://github.com/mozilla/DeepSpeech/wiki#frequently-asked-questions (Accessed on 25 May 2020)

Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., & Qu, R. (2019). A Survey of Deep Learning-Based Object Detection. IEEE Access, 7, 128837-128868.

Zhao, Z., Zheng, P., Xu, S., & Wu, X. (2019). Object Detection With Deep Learning: A Review. IEEE Transactions on Neural Networks and Learning Systems, 30, 3212-3232.

Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. ArXiv, abs/1804.02767.

Liu, X., Xu, Q., & Wang, N. (2018). A survey on deep neural network-based image captioning. The Visual Computer, 35, 445-470.

Hossain, M.Z., Sohel, F., Shiratuddin, M.F., & Laga, H. (2018). A Comprehensive Survey of Deep Learning for Image Captioning. ACM Computing Surveys (CSUR), 51, 1 - 36.

Tanti, M., Gatt, A., & Camilleri, K. (2018). Where to put the Image in an Image Caption Generator. Nat. Lang. Eng., 24, 467-489.

Tanti, M., Gatt, A., & Camilleri, K. (2017). What is the Role of Recurrent Neural Networks (RNNs) in an Image Caption Generator? INLG.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A.C., Salakhutdinov, R., Zemel, R., & Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. ICML.

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 3156-3164.

Jain, P., Agrawal, P., Mishra, A., Sukhwani, M., Laha, A., & Sankaranarayanan, K. (2017). Story Generation from Sequence of Independent Short Descriptions. ArXiv, abs/1707.05501.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krüger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., & Amodei, D. (2020). Language Models are Few-Shot Learners. ArXiv, abs/2005.14165.

Gatt, A., & Krahmer, E. (2018). Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation. J. Artif. Intell. Res., 61, 65-170.

spaCy official website. https://spacy.io/ , https://spacy.io/models/en#en_core_web_lg (Accessed on 20 November 2020)

COCO dataset. Official website. https://cocodataset.org/ (Accessed on 18 June 2020)

Childrens Book Test dataset. Official website. https://research.fb.com/downloads/babi/ (Accessed on 26 June 2020)

Flickr 30k dataset. Official website. https://shannon.cs.illinois.edu/DenotationGraph/ (Accessed on 14 June 2020)

Balys, Rokas. 2019. “YOLO v3 theory explained” medium.com. https://medium.com/analytics-vidhya/yolo-v3-theory-explained-33100f6d193 (Accessed on 03 July 2020)

Brownlee, Jason. 2019. "Gentle Introduction to Statistical Language Modeling and Neural Language Models". machinelearningmastery.com. https://machinelearningmastery.com/statistical-language-modeling-and-neural-language-models/ (Accessed on 25 November 2020)

Lamba, Harshall. 2018. "Image Captioning with Keras". towardsdatascience.com. https://towardsdatascience.com/image-captioning-with-keras-teaching-computers-to-describe-pictures-c88a46a311b8 (Accessed on 05 October 2020)

unknown. 2020. "Image captioning with visual attention". tensorflow.org. https://www.tensorflow.org/tutorials/text/image_captioning (Accessed on 10 October 2020)

jbrownlee. 2020. "keras-yolov3". Github repository (forked from https://github.com/jbrownlee/keras-yolo3 on 05 June 2020) minimaxir. 2020. "gpt-2-simple". Github repository (forked from https://github.com/minimaxir/gpt-2-simple on 18 October 2020)

Index of Figures

Index of Tables

Index of Listings

Appendix A – Detailed architecture of YOLOv3

The detailed architecture of the YOLOv3 neural network is shown below. It is created by the author based on an analysis of the network definition code and careful thinking through of the operations that happen for the model to work.

Essentially the model repeats some basic elements multiple times. This involves a convolutional layer, a batch normalisation layer and the activation function of a leaky ReLU. In the images below, the number preceding the X indicates how many times the basic building block operations are performed. For example, a 4X mean repetition of 4 times.

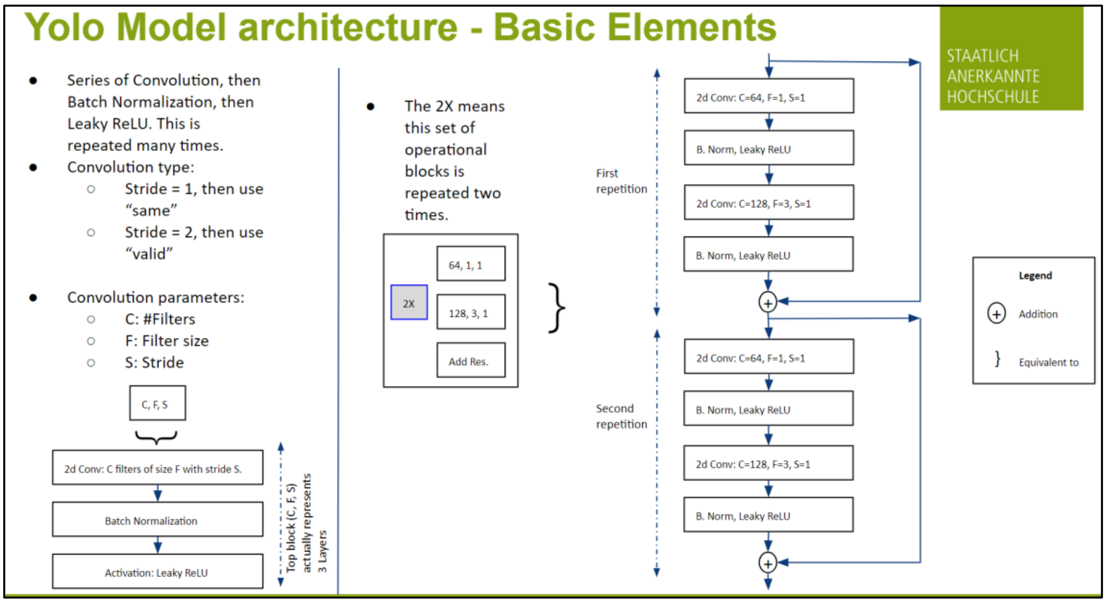


Figure 25: YOLOv3 architecture - part 1 of 3

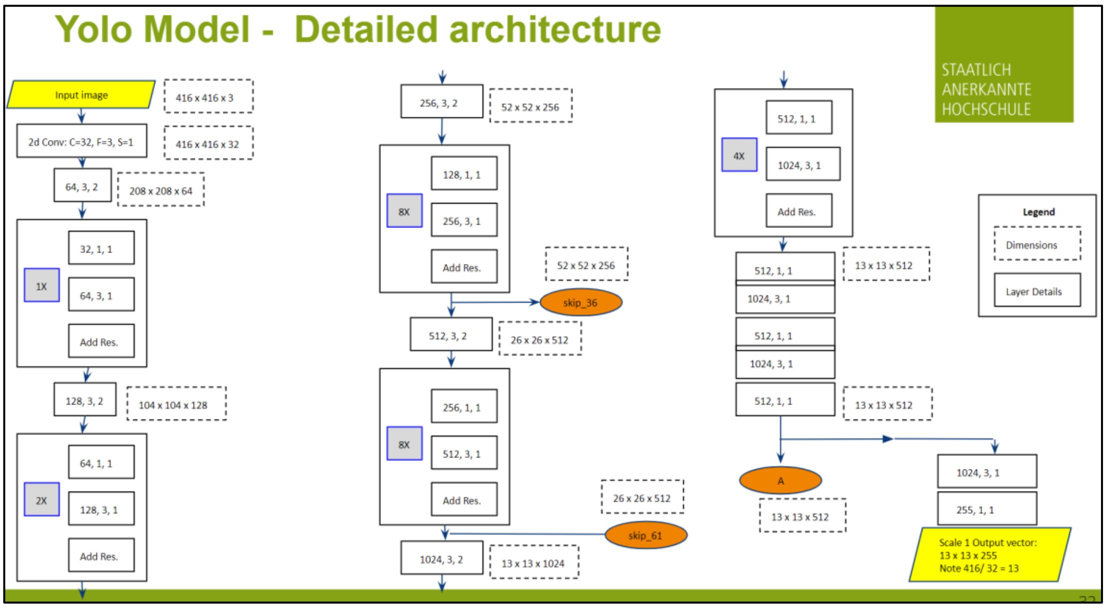


Figure 26: YOLOv3 architecture - part 2 of 3

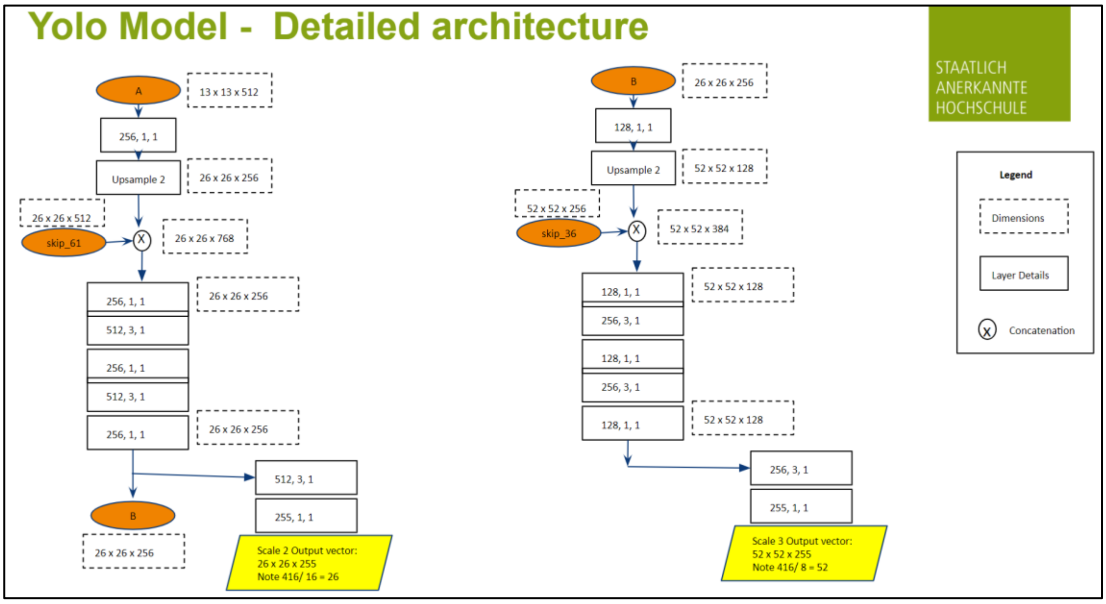


Figure 27: YOLOv3 architecture - part 3 of 3

Appendix B – Detailed architecture of Decoder for without-attention model

The detailed architecture of the decoder neural network for the Image caption generator model without-attention is shown below. It is created by the author based on an analysis of the network definition code and careful thinking through of the operations that happen for the model to work.

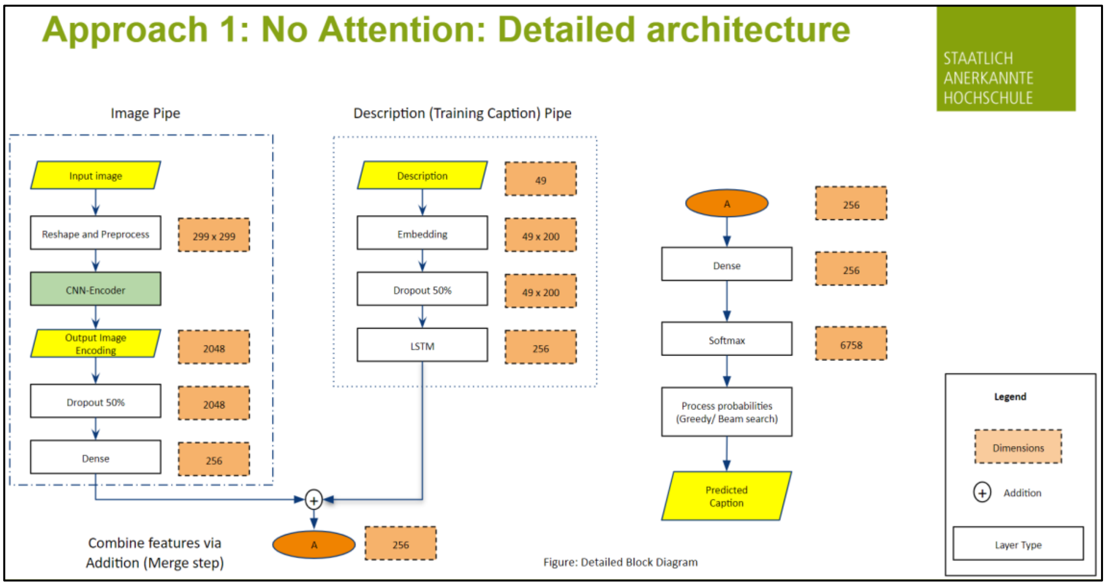


Figure 28: Decoder architecture for without-attention model

Appendix C – Table of Abbreviations

Abbreviation	Expanded form
AI	Artificial Intelligence
ANN	Artificial neural network
CBT	Children's Book Dataset
COCO	Common Objects in Context
CNN	Convolutional neural network
DNN	Deep neural network
GPT-2	Generative Pre-training version 2
GRU	Gated recurrent unit
GUI	Graphical User Interface
LM	Language model
LSTM	Long-short term memory
NLG	Natural language generation
POS	Part of Speech
RNN	Recurrent neural network
STT	Speech-to-text
SOTA	State-of-the-art
YOLO	You Only Look Once
YOLOv3	You Only Look Once version 3