BINTV1051U Big Data Analytics for Managers

Final Examination - Home Assignment

Name: Ang Peng Seng

Image Captions Generation with Neural Networks

Abstract

Automatic Image captioning is one of the most recent problem to attract both Computer Vision and Natural Language community. Object and pattern recognition in images, as well as relationships between words, has to be trained in order to successfully come up with an image caption.

In this project, I would attempt to use a pre-trained Convolutional Neural Network (Xception) to process image data, followed by building a deep multi-layer recurrent neural network to generate the captions in English. All these would be done using Python.

There are many open datasets like Flickr8K, Flickr30K, MSCOCO available to be used to train the model. 10,001 images out of the Flickr30K Dataset was used in this report due to computational limitations and the trained model achieved a mean test BLEU score of **43.1**.

Keywords:

Convolutional Neural Network, Recurrent Neural Network, Image Recognition, Natural Language Processing, BLEU score

Table of Contents

1.	Introduction	4
2.	Related Work	4
3.	Training Methodology	5
3.1.	Dataset Description	5
3.2.	Processing the images	5
3.3.	Processing the captions	6
3.4.	Combining Images and Captions	6
3.5.	Model overview	7
3.6.	Training the Model	7
4.	Proposed Model	8
4.1.	Feature Extraction from Images	8
4.2.	Recurrent Neural Network Decoder Framework	11
4.3.	Combined Model Architecture	14
5.	Results	15
5.1.	Metrics used	15
5.2.	Model Performance	16
5.3.	Examples of captions generated	17
6.	Conclusion	17
7.	Limitations and Future Works	18
8.	References	19
9.	Appendixes	20

1. Introduction

With the recent rampant growth of deep learning, one of the most recent interesting applications is image captioning. Image captioning refers to using the objects in images to generate text descriptions. While humans can easily describe a picture with just a quick glance of the image, it is not that direct for computers to perform that task. The caption generated has to not only capture the relationship between objects in the image, but also express them in a natural language (e.g English). As such, it is a problem that combines deep learning, computer vision as well as natural language processing.

Some applications of image captioning would be helping the those that are visually-impaired understand images, or storing huge set of images based on captions so it can be retrieved later. Social media platforms could also use it to understand where and what an individual is doing based on the image posted.

In this paper, Section 2 will discuss related work regarding image captioning. Section 3 will explain my training methodology. Section 4 will elaborate more on the models and their architecture used for this task. Section 5 shows the model performance (mainly the BLEU score). Section 6 and 7 would conclude the paper as well as provide some insights for future work and improvements.

2. Related Work

Image Classification

Image Classification is one of the earliest and most prominent topic in computer vision and it refers to predicting which group an image belongs to. Many neural networks, mainly Convolutional Neural Network (CNN), have been implemented to tackle this issue. In Chen Wang's paper [1], he explains how CNNs are used to classify images.

Various pre-trained CNNs models for image classification has been built since, for example, ResNet50 [2], VGG16 [3], Inception [4] and Xception [5].

Machine Translation

Machine Translation refers to translating a sentence from a particular language (A) into another language (B), by maximising P(B|A). For many years, machine translation was done by breaking the process into smaller, separate steps (translating individual words, aligning words, reordering words, etc). However, recently with Recurrent Neural Networks (RNNs), machine translation can be done in a more direct way.

In K. Cho's paper [6] on machine translation, the proposed model includes transforming the sentence into a vector representation by passing the sentence into an "encoder" RNN. The vector representation can then be used as the initial hidden state of a "decoder" RNN which would generate the target translated sentence.

Image Captioning

In Vinyals' paper [7], they proposed a model called Neural Image Caption (NIC). Their model, inspired by the encoder-decoder model from K. Cho's paper, was to use a convolutional neural network (CNN) to replace the "encoder" Recurrent Neural Network (RNN). Thereafter, the image representation produced from the last hidden layer of CNN can be used as the input to the "decoder" RNN to generate the captions.

There are other works [8] relating to generating descriptions for visual data but most are mainly for videos.

3. Training Methodology

3.1 Dataset Description

To train the image captioning model, some publicly available datasets that could be used are MSCOCO and Flickr30k. However, both datasets contains more than 30,000 images and due to my computational limitations, it would not be feasible to process the images and train the model. Instead, I will use 10,000 images from the Flickr30k dataset to train my model. The Flickr30k dataset contains 31,783 images, each with 5 captions, which are descriptions of the image. The dataset is owned by University of Illinois at Urbana, Champaign and include images from Flickr website.

a child walks in a grassy field .

a blond toddler walks in a field .

a toddler is walking in on a grassy knoll .

a baby boy crying standing on a field of grass .

young white boy on a grassy area with a sad expression on his face .

Figure 1: Sample Image and corresponding captions from the Flickr30k dataset

I would use 8000 images for the training set, 1001 images in the validation set as well as 1000 images in the test set.

3.2 Processing the images

The images from the Flickr30k dataset were first transformed into an array of (299,299,3) pixels before feeding it into the Xception model (using the last hidden layer as the output). The output would be an array of length 2048 which represents the features extracted from the images. A dictionary with each image tagged to its features is formed and saved.

Format of the dictionary saved:

{Image 1: (array of length 2048),
Image 2; (array of length 2048),
…….
Image X: (array of length 2048}

3.3 Processing the captions

Analysing all the captions shows that there are 20324 unique words.

Vocabulary Size: 20324 unique words

	word	count
0	a	271698
1	.	151039
2	in	83466
3	the	62978
4	on	45669

Figure 2: Top 5 commonly found words in the captions

In order to reduce the vocabulary memory space, I cleaned up the dataset by removing numerical words (e.g 3,4,5), punctuations and single-character words (e.g 'a') as omitting them would not change the meaning of the text drastically. Due to computational limitations, only 10,001 images will be used and only the first captions of each image will be used to train. Start and end sequence tokens were added by adding 'startcap' before the first word and 'endcap' after the last word.

Using the function Tokenizer from Keras, all captions were tokenized.
For example, the caption 'man in green' will be tokenised to [4,2,5] if the mapping is {'man':4,'in':2,'green':5}
After cleaning up the data, we have a total vocabulary size of 7481, however our tokenizer has set a maximum of 7000 most common unique words to be used to reduce memory space and the words not included in the tokenizer vocabulary would be replaced by '<unk>'.

3.4 Combining Images and Captions

After all the data preprocessing, we would then combine the processed images and captions. 8000 images and captions was used for the training set, 1001 for the validation set and 1000 for the test set. The captions would also be further split by padding the sequences of each word in the caption so that every input would have the same length. The output we want to predict would then be an array of probabilities which signify the 'confidence' that the token selected would be the correct one.

Using the same example where **Text = 'Man in green'** tokenized as **[4,2,5].**

The padding sequence function would give:
Actual = "Man", "Man in"
Xtext = [0,0,4], [0,4,2] (INPUTS)
Ytext = [0.1 **, 0.8**, 0, 0.1, 0]**,** [0.1,0,0.2,0,**0.7**] **(OUTPUTS) → Length of array of probabilities assuming a maximum of 5 vocabulary.**

The final training and validation set details are:

Training		Validation	
Input Text	**(127470,35)** 127470 tokenized text inputs, each with array of length 35	**Input Text**	**(15869,35)** 15869 tokenized text inputs, each with array of length 35
Input Image Features	**(127470,2048)** 127470 image inputs, each with array of length 2048	**Input Image Features**	**(15869,2048)** 15869 image inputs, each with array of length 2048
Output Token	**(127470,7481)** 127470 tokenized text outputs, each with array of length 7481	**Output Token**	**(15869,7481)** 15869 tokenized text outputs, each with array of length 7481

The model would be trained on the training set and would be evaluated on the validation set. The test set would then be used as a final unbiased dataset to determine the final BLEU score of our model.

3.5 Model Overview

The model proposed would receive an image as an input and output a sequence of words describing the image, where the words are generated from the dictionary built by the training data. The model is also trained by maximising the probability P(S|I), where S is the sequence of words and I is the image.

The model follows the following steps:

1. Takes in an image and extract the image features using Xception Neural Network
2. The multi-layer deep neural network would use the inputs from the encoded image features along with the tokenized text captions to generate the sequence of words.

Section 4 will elaborate the components of the whole model in detail.

3.6 Training the model

The model is then trained on the 127,470 training data, in section 3.4, and evaluated on the 15,869 validation set. The model is then trained for 8 epochs.

4. Proposed Model

The simplified overview algorithm I used to predict and generate a caption for an image is described below:

At every step, the model takes in 2 inputs:

- 2048-dimensional images features extracted from the Xception model
- Tokenized captions up to the i^{th} word

After going through our neural network model, the output would be:

- The predicted tokenized $i+1^{th}$ word of the caption (output would be the probabilities of each token being the next word and the highest probability token will be the one chosen)

This process is repeated until the word “endcap” (signifying the end of a caption) is predicted. (Note that “startcap” and “endcap” was added before the first word and after the last word respectively so that the model knows when to stop predicting).

The following sections will elaborate on the details of my proposed model.

4.1 Feature extraction from Images

4.1.1 Convolutional Neural Networks

The usage of Convolutional Neural Networks (CNNs) for computer vision has risen over the years and is now deemed to be the master algorithm in this field. CNN is very effective and has been used for many applications, like object detection, image classification and self-driving cars. The 2 main components for a CNN are the Hidden Layers for feature extraction and the fully connected layers for classification.

4.1.1.1 Hidden layers for feature extraction

There are 3 main components in the hidden layer of a CNN:

1) **Convolution Filters:**

To extract features, numerous convolution filters will be applied on the input to produce different features map, which would be put together as the final output of the convolution layer. The following diagram would provide a visual representation.

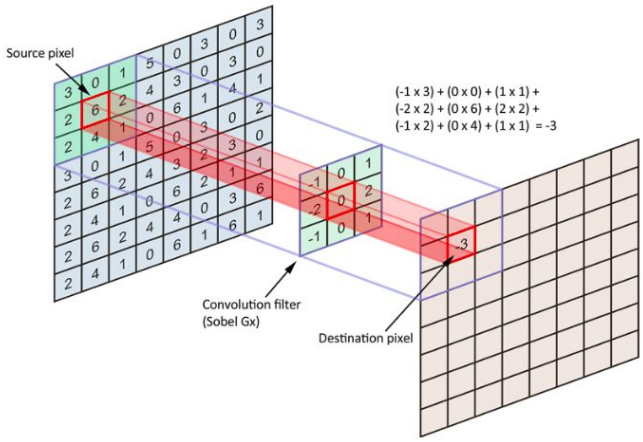

Figure 3: Convolution filter being slide over the input to produce feature maps. Source: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

2) Non-Linearity Activation Function

The output of the convolution layer would be linear as it results from multiplication and addition of the elements. Most of the data in real-life are non-linear hence it is also common to make the output non-linear by applying a non-linear activation function after every convolution operation. ReLu, Sigmoid and Tanh are some of the non-linearity functions, but the most commonly-used one is ReLU (Rectified Linear Unit), which transforms all negative inputs/pixels to zero.

3) Pooling

Pooling would then be done to reduce dimensionality of the image, to reduce computational time and prevent overfitting, but at the same time preserves the important features from the image. The most common type of pooling is max-pooling, where the maximum value of each window is kept.

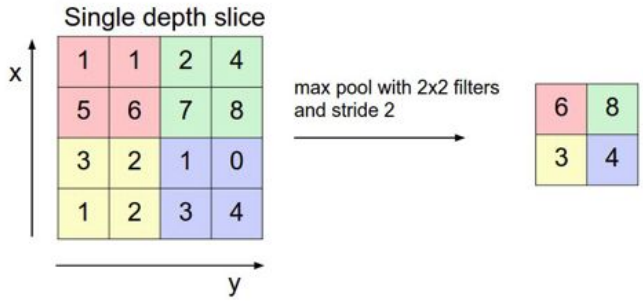

Figure 4: Max Pooling (Taking the largest value from each window). Source: http://cs231n.github.io/convolutional-networks/

4.1.1.2 Fully Connected Layer for Classification

The aim of the convolution and pooling layers would be to extract the important features of the image while the purpose of the fully-connected layer would be to map those features to the probabilities of the image being to each class. The fully connected layers would have full connections to the neurons from the previous layer, with Softmax Activation function used at the output layer. Softmax is most commonly used as it produces normalised class probabilities which are easier to interpret.

4.1.2 Feature Extraction using Xception

There are many different pre-trained models available and the following table below shows the models' performance against the ImageNet Validation set.

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	99 MB	0.749	0.921	25,636,712	168
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

Figure 5: Pre-trained Models' performance on ImageNet Validation dataset. Source:https://keras.io/applications/

While Inception and VGG16 are the most commonly used models for extracting features from images, I would instead use the **Xception** model for feature extraction. Xception has higher top-1 and top-5 accuracy compared to VGG and Inception, while also having lesser parameters, making the model lighter.

Xception was proposed by the creator and maintainer of the Keras library, François Chollet. The Xception model architecture can be seen in the diagram below.

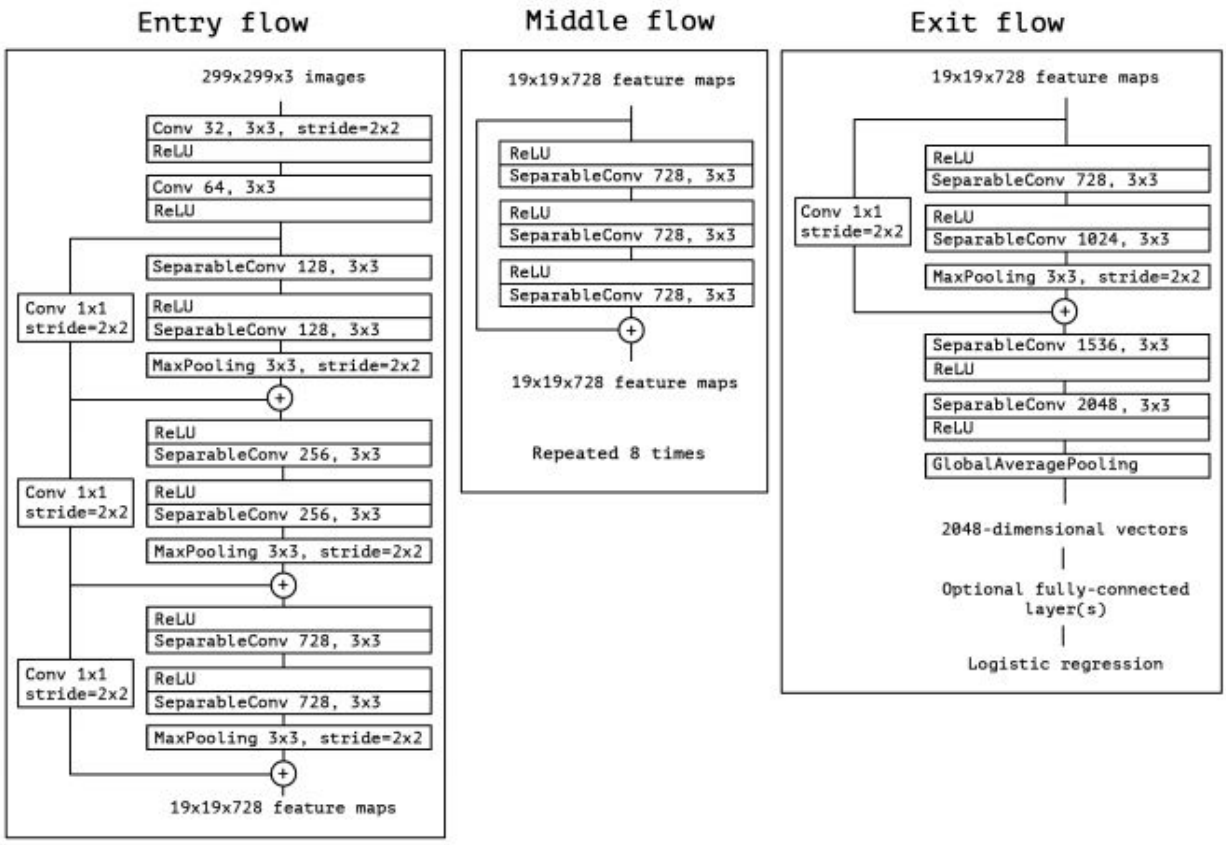

Figure 6: Xception Model Architecture. Source: https://arxiv.org/pdf/1610.02357.pdf

To extract features from images, we simply remove the output layer of the Xception model and instead output the last hidden layer of the model, which would represent the features extracted. Xception model takes in an input shape of (299,299,3) and outputs a vector of length 2048.
We can see that using the features extracted from Xception, we can form a 2D representation of the space and the clusters are indeed similar.

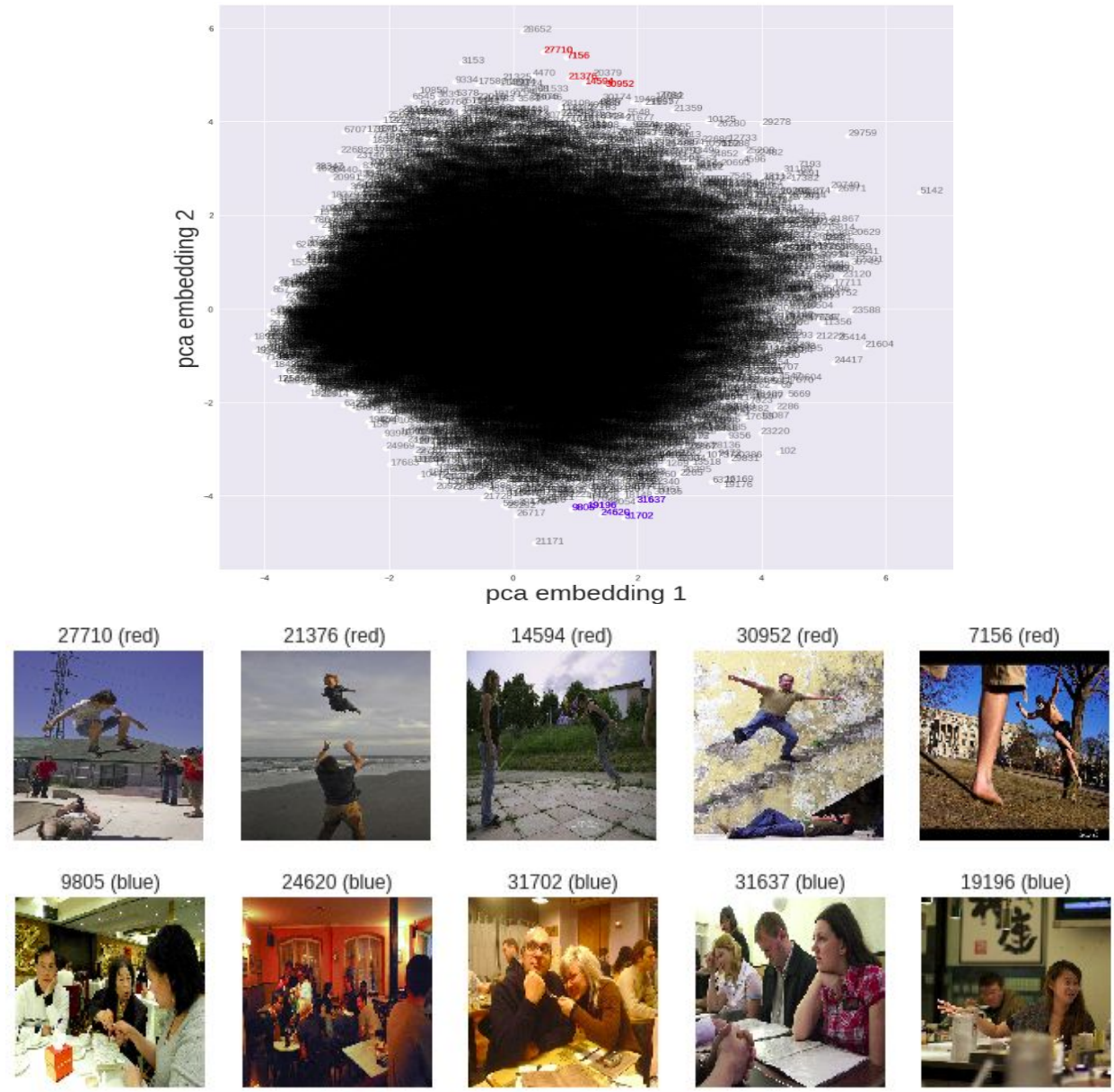

Figure 7: PCA representation and the 2 clusters

We can see that PCA of the features does cluster them relatively well,
- red clusters: People caught in mid-air
- blue clusters: Group of people sitting down at a table

4.2 Recurrent Neural Network Decoder Framework

4.2.1 Recurrent Neural Network

As humans, reading texts and understanding the words requires an understanding of the words you read before. Similarly, in building a language-based model to generate sentences, it is necessary to consider what output (words) that have been generated in the past in order to generate better predictions. However, traditional neural network does not have the property to reuse previous outputs. To address this issue, Recurrent Neural Network (RNN) can be used.

RNN [9] is a class of neural networks that can be thought of as having loops in them, or many copies of the same network with the output of each network passed on to its following network, hence making use of sequential information.

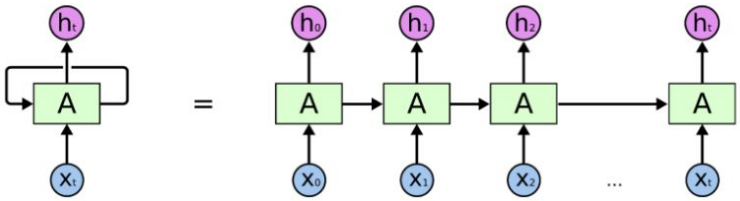

Figure 8: Unrolling the Recurrent Neural Network. Source from:
https://medium.com/explore-artificial-intelligence/an-introduction-to-recurrent-neural-networks-72c97bf0912

However, a problem with RNNs is that although it can connect information with short time lags, it is not effective in bridging the gaps between long time lags (5-10 time steps) [10]. For example, RNN would not have the problem predicting the sentence “The bird in the *sky*” as the gap between ‘bird’ and ‘sky’ is small. However, for a sentence like “I grew up in England since I was young and I speak *English*”, RNNs would have a hard time predicting ‘English” as the last word as the gap between “England” and the final word is huge. This is because RNNs suffer from the vanishing/exploding gradient problem, where in a multi-layer network, the gradients for the deeper layers are calculated by the product of many gradients of the previous activation functions (tanh or sigmoid functions). Hence, if the gradients are less than 1, the gradient might vanish over time and if the gradients are more than 1, it might explode over time.

To solve this long-term dependency problem mentioned, a special variation of RNNs, called Long Short-Term Memory (LSTM), could be used.

4.2.2 Long Short-Term Memory (LSTM)

The special components of LSTM model are its 3 gates, the input gate, the output gate and the forget gate. The input gate controls how much to store from the new cell state, the output gate controls the exposure of the output to the next layer of the network, while the forget gate controls how much of the existing memory to forget.

LSTMs solve the problem of vanishing gradient by using an identity function (f(x)= x) to store outputs (memory) if the model finds it useful and only re-inject them back into the network when they feel the time is right. This also enable the network to remember these values no matter how ‘long’ the sequence is.

A visualisation of how LSTMs work is shown here:

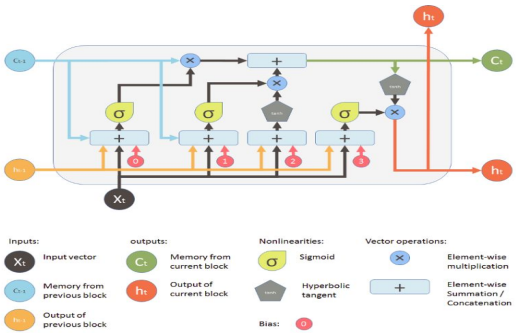

Figure 9: LSTM model. Source from:
https://github.com/shi-yan/FreeWill/blob/master/Docs/Diagrams/lstm_diagram.pptx

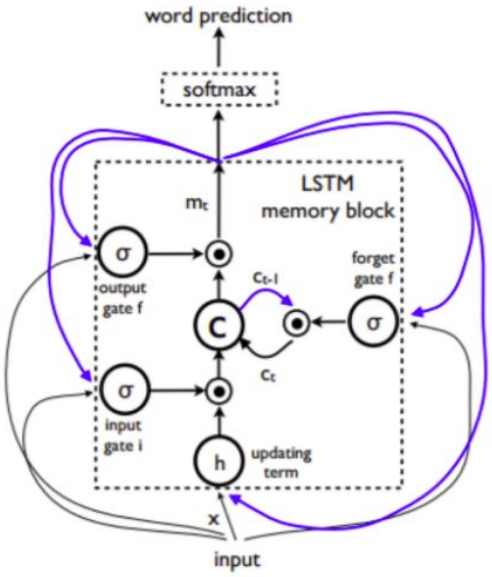

Figure 10: LSTM for language generation. Source from:
https://github.com/Shobhit20/Image-Captioning/blob/master/Report/Image%20Captioning.pdf

4.2.3 Gated Recurrent Unit (GRU)

A variant from LSTMs would be the Gated Recurrent Unit (GRU) [11]. GRU does not have a memory unit unlike LSTM and hence, GRU has only 2 gates, the reset gate and update gate. The reset gate decides on the combination of the previous memory with the new input while the update gate determines how much previous memory is to be kept.

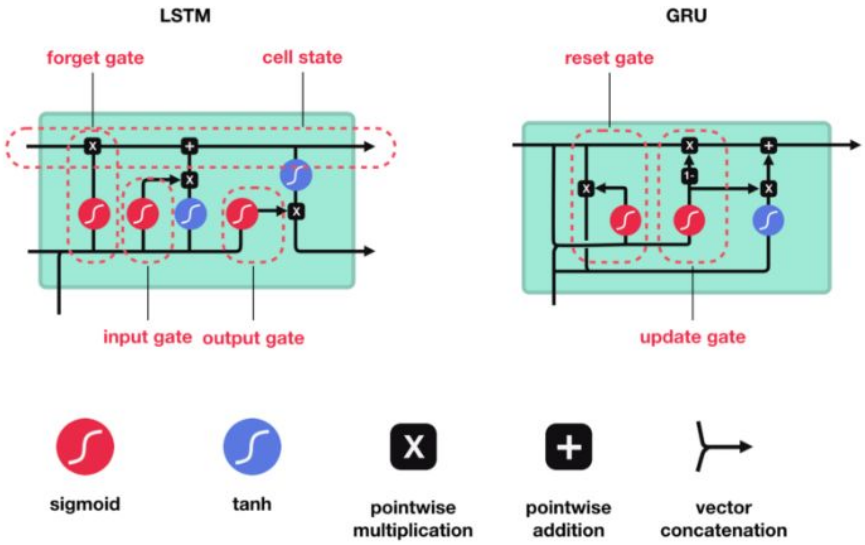

Figure 11: Comparison of LSTM and GRU. Source:
https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

GRU has also been tested that they are able to perform similarly, if not better than LSTM, and with shorter convergence time [11].

I would be using a mix of LSTM and GRU in my model.

4.3 Combined Model Architecture

The overall deep neural network model architecture can be seen here:

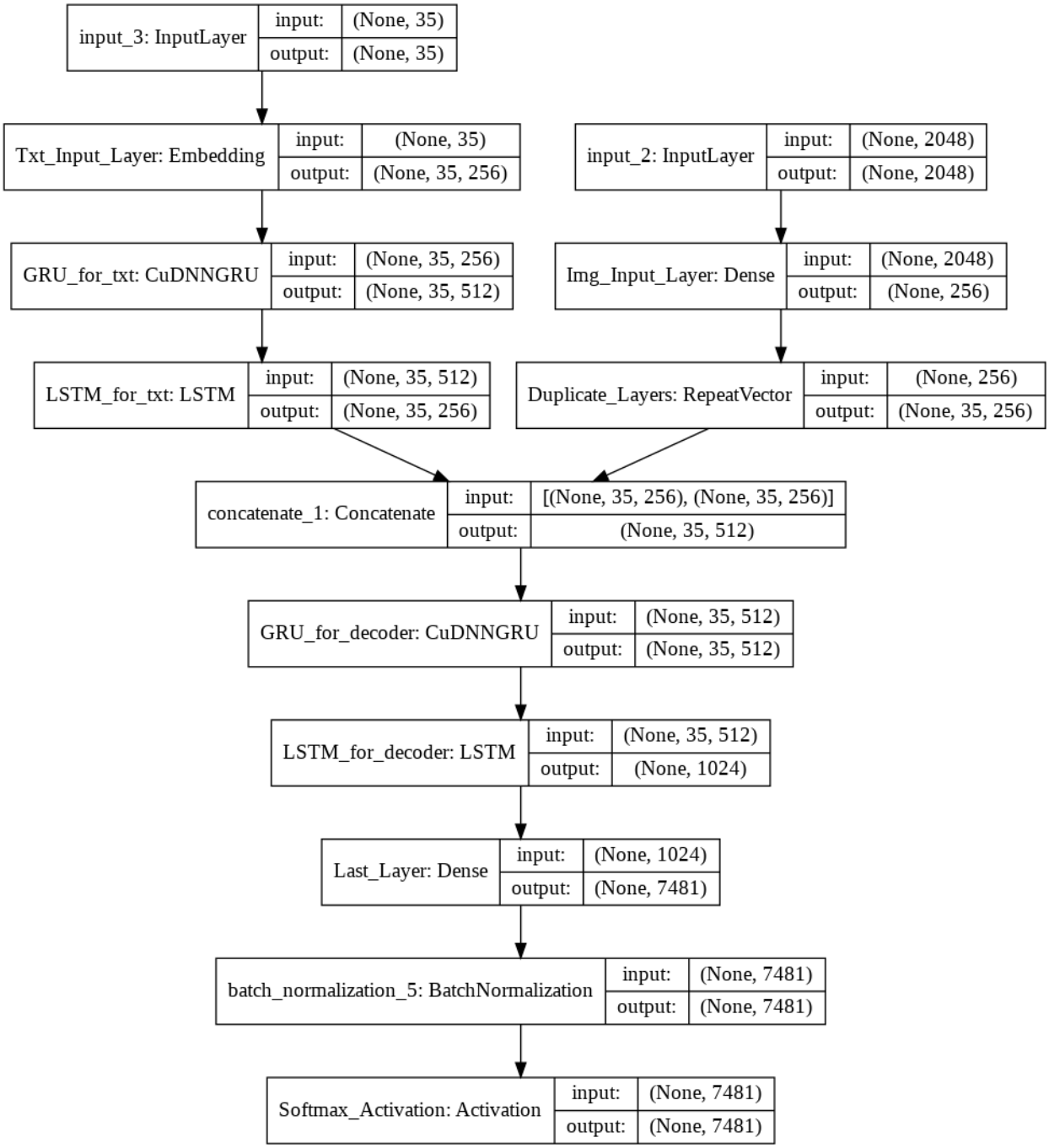

Figure 12: Full model architecture

Captions:

1. Current Tokenized Caption, up to i^{th} time step, (an array of length 35) will be passed through an embedding layer of 256 dimensions, which means each tokenized caption will be mapped to an array of weights of length 256.
2. The output will then be passed through a GRU layer followed by a LSTM layer which would produce a 35 * 256 output.

Image:

1. The image would be passed through the Xception model for feature extraction.
2. The features (array of length 2048) will then be passed through a hidden layer which would output an array of length 256.
3. The output will then duplicated 35 times to make the output a 3D shape as the future GRU layer requires a 3D input.

Decoder:

1. Following which, both the outputs of the tokenized caption and image are then multiplied together.
2. The output will then be passed through a GRU layer and a LSTM layer.
3. Finally, the output will then be put through a final layer of 7481 nodes with a softmax activation which would then return an array of probabilities that sums up to 1, where each node k represents the probability that the $i+1^{th}$ word is the k^{th} word, where k = 1…..4781 vocabulary words.

5. Results

5.1 Metrics used

The metric used to measure the model performance would be the **Bilingual Evaluation Understudy (BLEU)** score [12]. BLEU is one of the first metric to have a high correlation to human judgement of quality. BLEU indicates how similar a candidate (predicted) text is to the reference (actual) text. BLEU score ranges from 0 (total mismatch) to 1 (perfect match). In practice, it is almost impossible to achieve a BLEU score of 1 as both texts has to be matched exactly, and might not even be possible for humans too. There are however some limitation and drawbacks of the BLEU score, mainly that it does not take into account grammatical errors, weighs content words as well as common words equally and also tend to favor short translation.

The calculation of BLEU involved 2 parts, the modified precision and brevity penalty. The following example shows how to calculate BLEU score:

Reference: I really like eating apples
Hypothesis: I like eating apples

For BLEU-2 score, we would generate the 2-grams from both sentences to compare.

Reference 2-grams: (I, really), (really, like), (like, eating), (eating, apples)
Hypothesis 2-grams: (I,like) , (like,eating), (eating, apples)

Modified Precision is then calculated as the number of hypothesis 2-grams found in the reference 2-grams. In this case, there are two 2-grams ((like, eating) and (eating, apples)) from the hypothesis that is also found in the reference. Hence, the modified precision would be 2/3.

Using just the modified precision method has its limitations. For example, if the number of reference words is large, while the hypothesis length is small, the 2 sentences should not be

similar but the modified precision would still give the same score as long as a subset of the reference is similar to the hypothesis. Hence, the Brevity Penalty is used to penalise long reference sentence.

The Brevity Penalty [13] is: $e^{1-\frac{length\ of\ reference}{length\ of\ hypothesis}}$ if length(reference) $\geq$ length(hypothesis), else the Brevity Penalty is 1.

In this example, the BLEU-2 score is calculated by:
Modified Precision * Brevity Penalty = 2/3 * exp(1-(5/4)) = 0.519

BLEU [12] is then calculated by the **geometric average** of BLEU-1 to BLEU-X, where X is the length of the hypothesis sentence. BLEU is sometimes also reported by multiplication of 100.

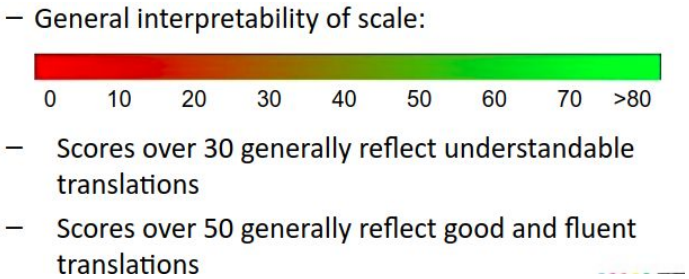

Figure 13: BLEU score ranges (X100) and interpretations. Source: https://slideplayer.com/slide/7774670/

5.2 Model Performance

Using the Adam optimizer and training for 8 epochs, the BLEU results are:

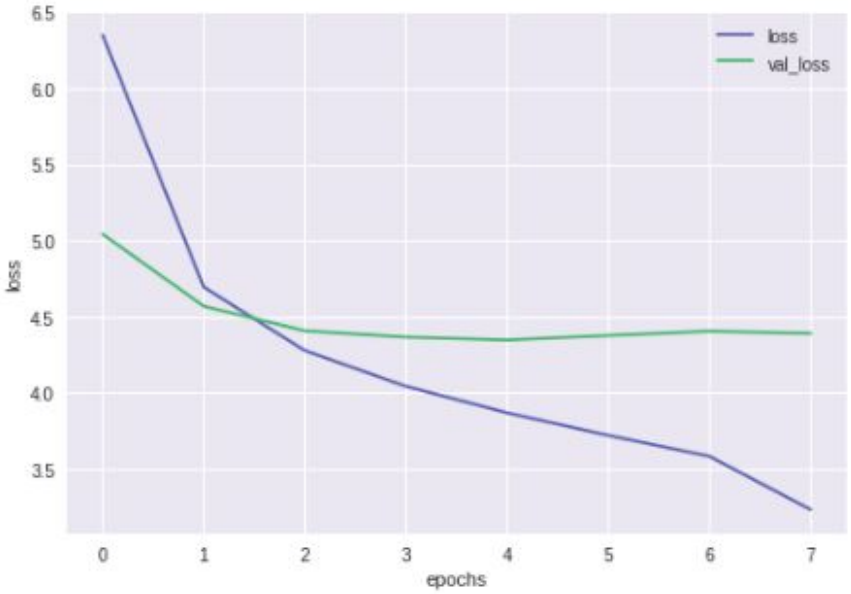

Figure 14: Categorical Cross-Entropy Loss for training and validation sets over epochs trained

BLEU score (Reported in X100):

Training Set (Mean BLEU score): **41.8**
Validation Set (Mean BLEU score): **42.7**
Test Set (Mean BLEU score): **43.1**

5.3 Examples of captions generated

Actual Caption: young boy with hat and brown shirt holding hand just above wooden box

Predicted Caption: man wearing blue shirt and blue hat is holding large camera in his hand

BLEU score: 0.773

Actual Caption: yellow dog and black and white dog are running in the dirt

Predicted Caption: black dog is running through the grass with ball in its mouth

BLEU score: 0.803

Actual Caption: man with black and blue backpack hikes down rock face

Predicted Caption: man in blue shirt and black shorts is walking on rock

BLEU score: 0.821

Figure 15: Examples of captions generated on the Test set

The results are not perfect but considerably good given the relatively small amount of data it uses to train on.

6. Conclusion

My proposed model combines CNN, GRU and LSTM to maximise the likelihood of the predicted caption given an image. The implemented trained model is able to decently take in an image and generate reasonable English captions. The model performance is measured using BLEU and training it on 10,000 images from the Flickr30k dataset gives a mean test BLEU score of **43.1.** With higher computational power, the model could be trained on a larger number of images and captions to improve the accuracy.

7. Limitations and Future Works

There are 2 limitations in the proposed model. Firstly, the training is based on maximising the probability of each word given the previous word and image by back-propagation and is only trained on the training and validation set, testing it with the test set might result in error accumulation as it does not get any feedback and exposure from its own predictions (exposure bias problem) [14]. Secondly, the model used minimizes cross-entropy loss but the metric used, BLEU, is non-differentiable (wrong-objective problem). Reinforcement learning methods, especially Actor-Critic model, could also be implemented to solve both problems [15]. In a Reinforcement Learning Actor-Critic model, the model would be trained by having a policy network (Actor) and a value network (Critic). The Critic job would be to predict the value of each state (Image and sequence of words generated so far), which is then used to train the Actor, whose job is to perform the best action at any given state. This model would use the metric (BLEU) directly as a reward function and hence optimise it directly.

8. References

1. W. Chen, X. Yang, Convolutional Neural Network for Image Classification https://pdfs.semanticscholar.org/b8e3/613d60d374b53ec5b54112dfb68d0b52d82c.pdf

2. H. Kaiming, Z. Xiangyu, R. Shaoqing, S. Jian Deep Residual Learning for Image Recognition https://arxiv.org/pdf/1512.03385.pdf

3. S. Karen, Z. Andrew Very Deep Convolutional Networks for Large-Scale Image recognition https://arxiv.org/pdf/1409.1556.pdf

4. S. Christian, L. Wei, J. Yangqing, S. Pierre, R. Scott, A. Dragomir, E. Dumitru, V. Vincent, R. Andrew Going deeper with convolutions https://arxiv.org/pdf/1409.4842.pdf

5. F. Chollet, Xception: Deep Learning with Depthwise Separable Convolutions https://arxiv.org/pdf/1610.02357.pdf

6. K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations. using RNN encoder-decoder for statistical machine translation In EMNLP, 2014. https://arxiv.org/pdf/1406.1078.pdf

7. Vinyals, Oriol, Toshev, Alexander, Bengio, Samy, and Erhan, Dumitru. Show and tell: A neural image caption generator. arXiv:1411.4555, November 2014. https://arxiv.org/pdf/1502.03044.pdf

8. B. Z. Yao, X. Yang, L. Lin, M. W. Lee, and S.-C. Zhu. I2t: Image parsing to text description. Proceedings of the IEEE, 98(8), 2010. 2 http://www.stat.ucla.edu/~sczhu/papers/I2T_IEEE_proc.pdf

9. C. Zachary, B. John, E. Charles A Critical Review of Recurrent Neural Networks for Sequence Learning https://arxiv.org/pdf/1506.00019.pdf

10. A.G Felix S. Jurgen, C. Fred Learning to Forget: Continual Prediction with LSTM http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.46.772&rep=rep1&type=pdf

11. C. Junyoung, G.Caglar, C. KyungHyun, B. Yoshua Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling https://arxiv.org/pdf/1412.3555.pdf

12. P. Kishore , R. Salim, W. Todd, and Z. Wei-Jing BLEU: a Method for Automatic Evaluation of Machine Translation https://www.aclweb.org/anthology/P02-1040.pdf

13. Kumari, Lalita & Debbarma, Swapan & Shyam, Radhey. (2011). Development of Machine Translation Evaluation Technique. https://www.researchgate.net/publication/224771080_Development_of_Machine_Translation_Evaluation_Technique

14. N. Ding, R. Soricut Cold-Start Reinforcement Learning with Softmax Policy Gradient https://papers.nips.cc/paper/6874-cold-start-reinforcement-learning-with-softmax-policy-gradient.pdf

15. L. Zhang, S. Flood, F. Liu, T. Xiang, G. Shaogang, Y. Yongxin, Timothy M. Hospedales Actor-Critic Sequence Training for Image Captioning https://qmro.qmul.ac.uk/xmlui/bitstream/handle/123456789/36515/Gong%20Actor-Critic%20Sequence%202017%20Accepted.pdf?sequence=1

9. Appendixes

9.1 Python Codes for the neural network model in keras

```
from keras import layers
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import merge,LSTM, Embedding,BatchNormalization, Dropout, TimeDistributed, Dense, RepeatVector, Activation, Flatten,Concatena
from keras.optimizers import Adam, RMSprop
from keras.layers.wrappers import Bidirectional

dim_embedding = 256
units=512

#image
input_image = layers.Input(shape=(Ximage_train.shape[1],))
image_feat = layers.Dense(dim_embedding, input_shape=(Ximage_train.shape[1],), activation='relu',name='Img_Input_Layer')(input_image)
image_feat = layers.RepeatVector(maxlen,name='Duplicate_Layers')(image_feat)

#captions
input_txt = layers.Input(shape=(maxlen,))
txt_feat = layers.Embedding(vocab_size,dim_embedding ,input_length=maxlen, name='Txt_Input_Layer')(input_txt)
txt_feat = layers.CuDNNGRU(units, return_sequences=True, return_state=False, recurrent_initializer='glorot_uniform',name='GRU_for_txt')(txt_fea
txt_feat = layers.LSTM(units=256,return_sequences=True,name='LSTM_for_txt')(txt_feat)

# combined txt and image into the decoder part of the framework
decoder = layers.multiply([txt_feat,image_feat])
decoder = layers.CuDNNGRU(units, return_sequences=True, return_state=False, recurrent_initializer='glorot_uniform',name='GRU_for_decoder')(deco
decoder =layers.LSTM(1024,return_sequences=False,name='LSTM_for_decoder')(decoder)
decoder =layers.Dense(2048,activation=None,name='Hidden_Layer_after_LSTM')(decoder)
output = layers.Dense(vocab_size,name='Last_Layer')(decoder)
output =layers.BatchNormalization()(output)
output =layers.Activation('softmax',name='Softmax_Activation')(output)
model = models.Model(inputs=[input_image, input_txt],outputs=output)

print(model.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_5 (InputLayer)         (None, 35)                0         
_________________________________________________________________
Txt_Input_Layer (Embedding)  (None, 35, 256)           1915136   
_________________________________________________________________
input_4 (InputLayer)         (None, 2048)              0         
_________________________________________________________________
GRU_for_txt (CuDNNGRU)       (None, 35, 512)           1182720   
_________________________________________________________________
Img_Input_Layer (Dense)      (None, 256)               524544    
_________________________________________________________________
LSTM_for_txt (LSTM)          (None, 35, 256)           787456    
_________________________________________________________________
Duplicate_Layers (RepeatVector) (None, 35, 256)        0         
_________________________________________________________________
multiply_2 (Multiply)        (None, 35, 256)           0         
                                                                 
_________________________________________________________________
GRU_for_decoder (CuDNNGRU)   (None, 35, 512)           1182720   
_________________________________________________________________
LSTM_for_decoder (LSTM)      (None, 1024)              6295552   
_________________________________________________________________
Hidden_Layer_after_LSTM (Dense) (None, 2048)           2099200   
_________________________________________________________________
Last_Layer (Dense)           (None, 7481)              15328569  
_________________________________________________________________
batch_normalization_6 (BatchNor (None, 7481)           29924     
_________________________________________________________________
Softmax_Activation (Activation) (None, 7481)           0         
=================================================================
Total params: 29,345,821
Trainable params: 29,330,859
Non-trainable params: 14,962
_________________________________________________________________
None
```

9.2 Full Python Jupyter Notebook:

https://github.com/Angps1995/Automated_Image_Captioning/blob/master/Flickr30k_Image_Captioning.ipynb