NANYANG TECHNOLOGICAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

SC4001 Group Project Report

AY 2023-2024

Group members:

Name	Matric No.
Ng Shang Yau	U2121960G
Lam Hao Fah	U2121999H
Wong Chu Feng	U2140100D

1 Background Motivation

The emergence of Deep Learning has sparked a wave of innovation in the domain of Artificial Intelligence, revolutionizing the way we, humans, interact with the digital world. Our group has decided to explore the realm of image captioning. The challenge here lies in teaching our machines to not only “see”, but also “describe” what they see. This is a task that humans perform effortlessly, but for machines, it is an intricate balance of pattern recognition, context understanding, and language synthesis.

Our project is motivated by the fundamental human desire to make sense of the world through the lens of language. Images help to convey a vast amount of information immediately. However, this visual information is not readily accessible to everyone. For instance, visually impaired individuals rely heavily on descriptions to interpret images, while industries such as surveillance, seek to automate the tedious task of annotating vast quantities of visual data. Hence, our project showcases usefulness to various aspects of society.

2 Tasks

Creating an image captioning model that generates a suitable caption for any given picture. The caption generated will need to minimally identify the objects present and describe their relationship(s).

3 Approach

There are two main ways to approach image captioning: bottom-up and top-down. Bottom-up methods focus on generating individual objects observed in an image and then combining these identified objects to form a caption. This method begins with breaking down constituents of the image, such as position of objects, relations among objects and many other silent features.

Top-down methods, on the other hand, start from the image and converts it into words. This approach aims to create a semantic representation of an image by extracting its features, which is subsequently decoded into a caption using various architectures. The top-down approach utilizes an encode-decode network architecture, where a Convolution Neural Network (CNN) is used as the encoder whereas a Recurrent Neural Network (RNN) us used as the decoder.

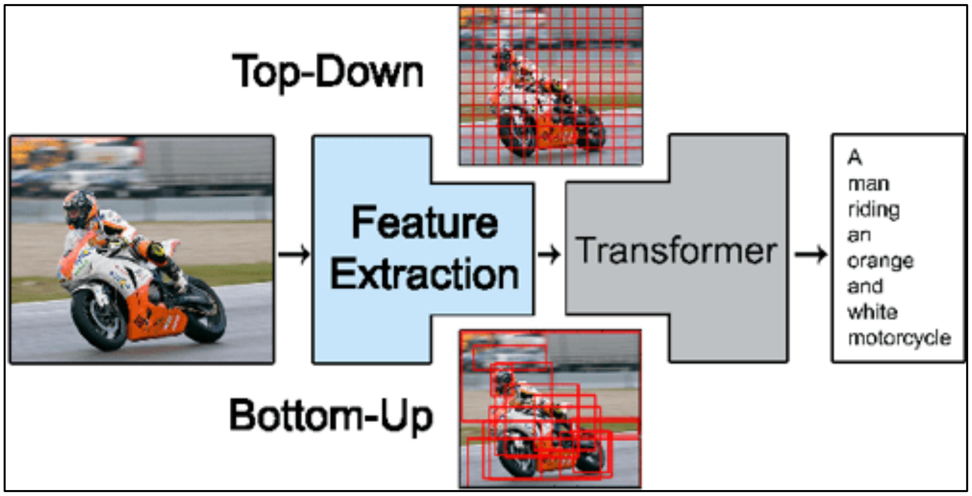


Fig 1. Visualization of top-down and bottom-up approach for image captioning (Parameswaran & Das, 2018)

Our group will be adopting the top-down approach as we acknowledge the Encoder-Decoder architecture as one of the most successful methods for image captioning (Al-Malla et al., 2022). Using this approach, we will need to have 2 main models, the encoder and decoder respectively.

We have decided to use a pre-trained CNN model VGG16 to extract and encode features from an image. VGG16 has been trained on large datasets, used widely, and proven to be versatile and extremely accurate in image classification. By leveraging on VGG16 as our encoder, we can save valuable time and resources while achieving impressive results.

Apart from the encoders, we will also need to identify a suitable decoder to generate our image captions. In general, RNN models are used as decoders due to their ability to process information sequentially and generate captions word by word. Specifically, we will be using Long Short-Term Memory (LSTM).

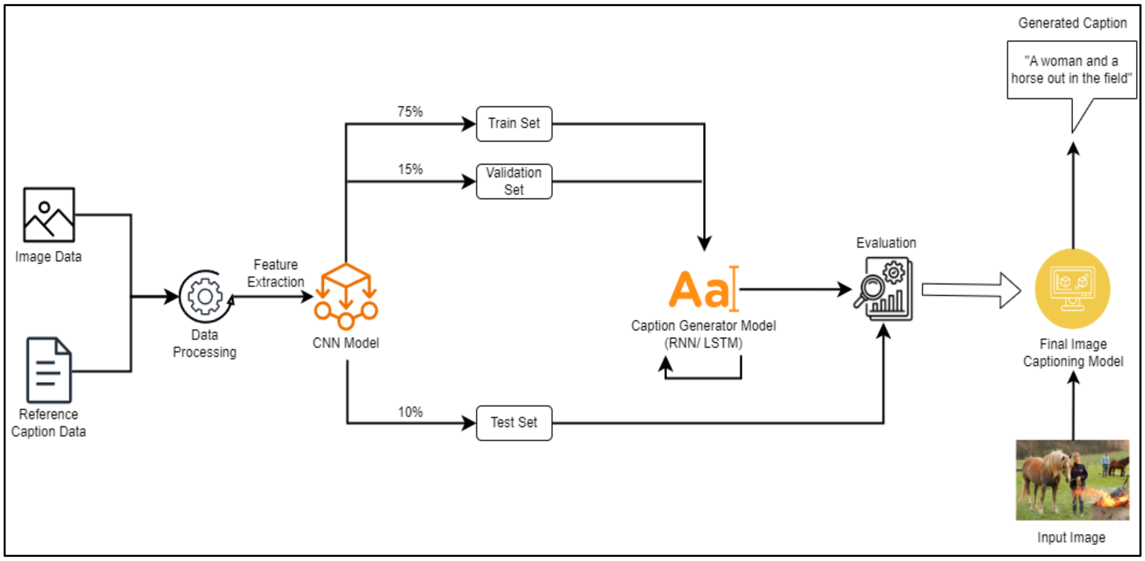


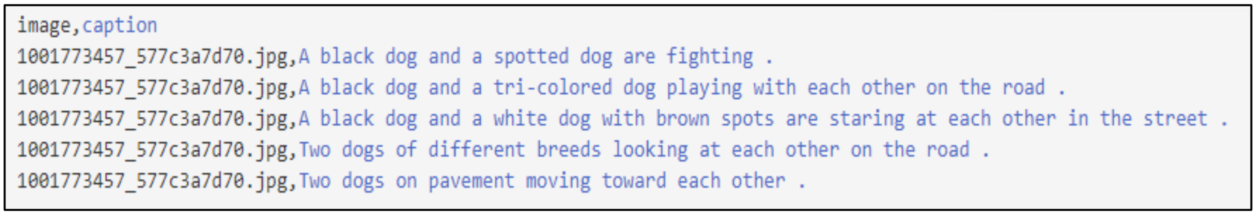
Fig 2. Our image captioning approach

For our project, we will be implementing a 75-15-10 train-validation-test split for the image dataset. After initial data pre-processing and feature extraction using VGG16, we will be generating captions through the use of LSTM. Through experimenting and introducing progressive enhancements to our hybrid CNN-LSTM network, we wish to **identify the most optimal hybrid network to perform image captioning**. Throughout the entire study, we will evaluate the accuracy of generated captions through the use of Unigram BLEU scores.

4 Datasets

In this section, we will be looking at what we used to train and evaluate our image-captioning model's architecture. Specifically, we used the *Flickr 8k Dataset* from (https://www.kaggle.com/datasets/adityajn105/flickr8k).

This dataset includes a folder containing 8,000 random images. Each image comes with 5 assigned captions that convey detailed descriptions of key objects and events which is stored in a *txt* file.



```
image,caption
1001773457_577c3a7d70.jpg,A black dog and a spotted dog are fighting .
1001773457_577c3a7d70.jpg,A black dog and a tri-colored dog playing with each other on the road .
1001773457_577c3a7d70.jpg,A black dog and a white dog with brown spots are staring at each other in the street .
1001773457_577c3a7d70.jpg,Two dogs of different breeds looking at each other on the road .
1001773457_577c3a7d70.jpg,Two dogs on pavement moving toward each other .
```

Fig 3. Example of txt containing assigned captions for each image

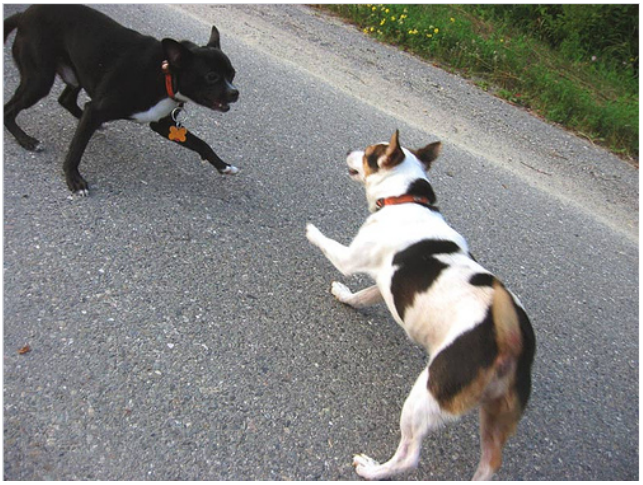


Fig 4. Example of image '1001773457_577c3a7d70.jpg'

For instance, image '1001773457_577c3a7d70.jpg' in Fig 4 has 5 captions under the 'caption' column in Fig 3.

1. A black dog and a spotted dog are fighting.
2. A black dog and a tri-colored dog playing with each other on the road.
3. A black dog and a white dog with brown spots are staring at each other in the street.
4. Two dogs of different breeds looking at each other on the road.
5. Two dogs on pavement moving toward each other.

5.1 Convolution Neural Network (CNN)

A CNN is a neural network commonly used for image processing tasks, designed to learn spatial hierarchies of features automatically and adaptively from input images. CNNs are particularly well-suited for our project on image captioning as they can learn to extract high-level features from raw image data. CNNs are generally composed of 3 layers: Convolutional layer, pooling layer, and fully connected layer (Mishra, 2020). The convolutional layer applies a set of filters to the input image to extract features such as edges, corners, and textures. The output of this layer is then passed through a non-linear activation function to introduce non-linearity into the model. The pooling layer then reduces the dimensionality of the data, lowering the number of parameters in the model, which can help prevent overfitting. The fully connected layer takes the output of the previous layer and flattens it into a one-dimensional vector, which is passed through a series of other fully connected layers to perform classification or regression.

For our project, we used a pre-trained CNN model VGG16 to extract features from our dataset of images.

5.1.1 VGG16

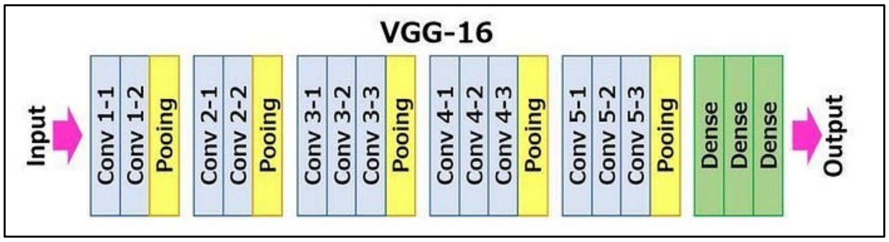


Fig 5. VGG16 architecture (Rohini, 2023)

VGG16 is a CNN model that is described as one of the best image classification models to date (Rohini, 2023). It stands out for its architecture which, despite being simplistic compared to other CNN models, achieves remarkable performance. The model is made up of 16 weight layers (13 convolutional layers & 3 Dense layers), as well as 5 pooling layers. Each of the convolutional layers is followed by the ReLU activation function, and after convolutional layers come a max pooling layer. The final part of the architecture consists of 3 fully connected (Dense) layers. VGG16's consistent use of 3x3 convolutional filters throughout the network contributes to its effectiveness, as these small filters help in capturing fine details in an image while keeping the network's parameters to a manageable number.

As aforementioned in section 3, VGG16 has been trained on extremely large datasets with over a million images and more than 1000 classification classes. It is possible that we reduce the scope of our image-captioning model to classify a smaller number of objects, for example, a model to recognize the type of animal. However, we feel that that will reduce our project to a simple classification problem where we only need to detect a few classes, making things a lot more trivial. Hence, we opted to utilize a pre-trained model for our CNN, which is otherwise unfeasible to create on our own feature extraction model, and focus on finding the optimal hybrid model instead.

5.2 Long Short-Term Memory (LSTM)

Our project requires the image-captioning model to generate a logical and grammatically sound sentence that describes the input image. A natural choice for such a task is a LSTM model. LSTM is a type of Recurrent Neural Network (RNN) used to predict sequences of data. The LSTM network architecture effectively tackles the issue of vanishing gradients when dealing with long sequence dependencies, a problem faced by traditional RNNs, by remembering information for long periods of time. This is crucial for us to generate coherent and accurate captions.

The LSTM architecture is made up of a memory cell, functioning as a form of memory to maintain cell states over time, as well as gates. The input gate, forget gate and output gate control the extent to which a new value flows into the memory cell, a value remains in the memory cell, and the value in the memory cell is used to compute output activation of the LSTM unit correspondingly. Each gate has a sigmoidal neural network layer and a pointwise multiplication operation which allows the LSTM to selectively remember or forget information (Saxena, 2023).

While we shall not delve deep into the technical mathematical details of each gate, we know from research that this unique architecture is integral to the image captioning process due to its proficiency in sequence modeling and prediction.

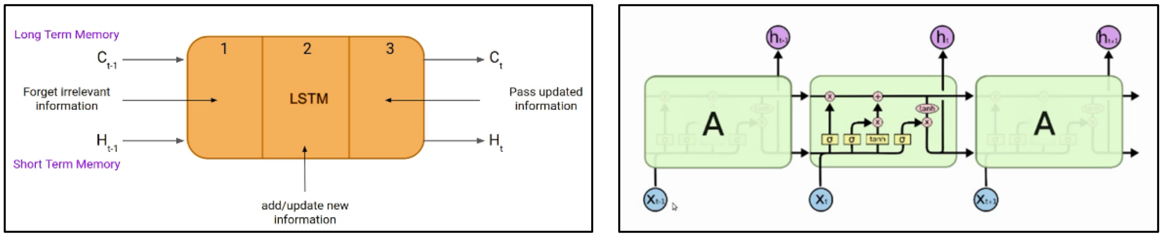


Fig 6. LSTM network architecture (Saxena, 2023)

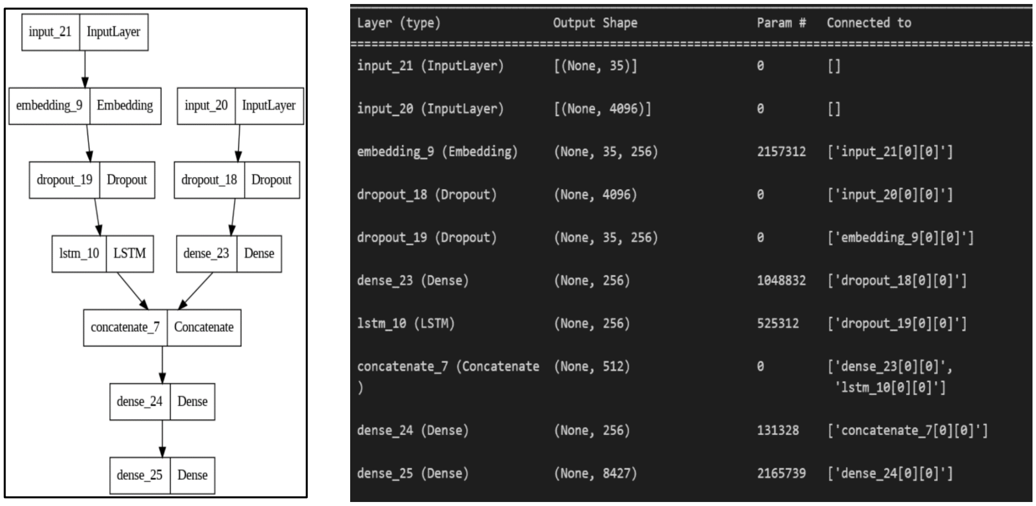
The concerns we have for using a LSTM model include computational complexity and data bias. Due to its more complex architecture, a LSTM is computationally expensive and training it will take a longer time as compared to a simple RNN. Furthermore, a LSTM learns to generate captions based on the distribution of the training data. If the training data is biased or lacks diversity, the LSTM will replicate these biases in its predictions. Thankfully, the dataset that we are using is relatively large and contains a variety of scenes and situations, reducing the onset of data bias.

5.3 CNN + LSTM network

The CNN and LSTM models work hand in hand to convert visual data into descriptive narratives. As outlined in section 3, the CNN serves as the encoder, extracting a rich feature vector from the input image, which encapsulates visual details necessary for caption generation. This feature vector is then fed into the LSTM, the decoder, which interprets the features as a sequence of words, outputting one word at each time step. The LSTM adjusts its output by considering both the features provided by the CNN and the sequence of words that have been generated thus far, ensuring that the captions are both consistent and contextually appropriate. This process continues iteratively until the LSTM produces an end-of-sequence token, signaling the completion of the caption. Throughout the training phase, the parameters of both the CNN and LSTM are refined in tandem, enhancing the model’s ability to correlate visual data with appropriate linguistic expressions.

5.3.1 Network 1 (Baseline CNN-LSTM network)

For our project, the first network that we established is a baseline CNN-LSTM network, characterized by the architectural configuration in Fig 7 below.



```
Layer (type)                Output Shape         Param #   Connected to
===========================================================================================
input_21 (InputLayer)       [(None, 35)]         0         []

input_20 (InputLayer)       [(None, 4096)]       0         []

embedding_9 (Embedding)     (None, 35, 256)      2157312   ['input_21[0][0]']

dropout_18 (Dropout)        (None, 4096)         0         ['input_20[0][0]']

dropout_19 (Dropout)        (None, 35, 256)      0         ['embedding_9[0][0]']

dense_23 (Dense)            (None, 256)          1048832   ['dropout_18[0][0]']

lstm_10 (LSTM)              (None, 256)          525312    ['dropout_19[0][0]']

concatenate_7 (Concatenate  (None, 512)          0         ['dense_23[0][0]',
)                                                           'lstm_10[0][0]']

dense_24 (Dense)            (None, 256)          131328    ['concatenate_7[0][0]']

dense_25 (Dense)            (None, 8427)         2165739   ['dense_24[0][0]']
```

Fig 7. Basic CNN + LSTM network

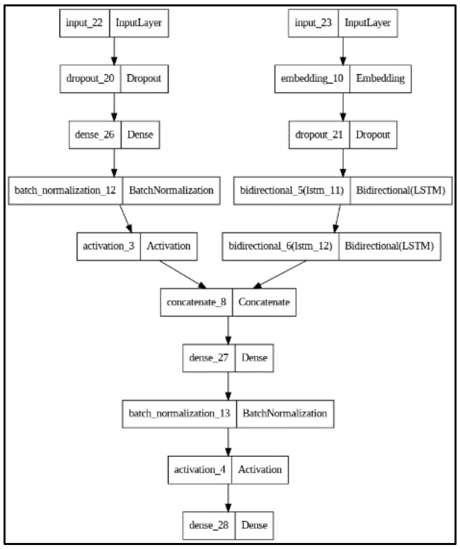
For processing image features, the architecture includes an input layer (input_20) designed to receive image features, that are outputted from a pre-trained VGG16. It then incorporates a dropout layer (dropout_18) to prevent overfitting by randomly omitting a fraction of the input units during training. This is followed by a dense layer (dense_23).

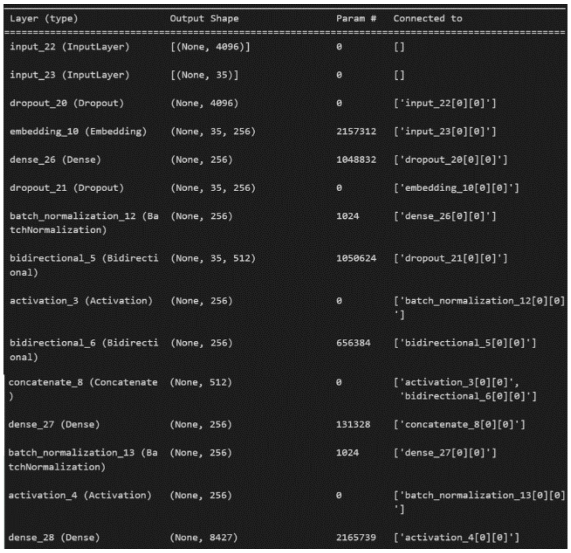
In parallel, the architecture manages text features through a separate input layer (input_21) designed for sequential text data. The data is processed through an embedding layer (embedding_9) that transforms text inputs into dense vectors of a fixed size, and another dropout layer (dropout_19) to mitigate overfitting on the text data. The LSTM layer (lstm_10) processes the sequence of word embeddings, considering temporal relationships between words.

Finally, the architecture features a concatenation layer (concatenate_7) that merges the outputs from the image and text processing streams. This consolidated data is then decoded through additional dense layers (dense_24 & dense_25). Each unit's activation in this layer correlates to the probability of a specific word being the next one in the sequence. This design enables the model to leverage both visual features extracted from images and the contextual information in text, allowing it to perform sophisticated tasks that demand an understanding of both visual and textual elements.

5.3.2 Network 2 (Bi-Directional LSTM)

To capture temporal dependencies in our data, whereby previous behavior affects current behavior, we introduced bidirectional LSTM layers as seen in Fig 8 (lstm_11 & lstm_12). In a standard LSTM, information flows only from the past to the present. Bi-directional LSTMs, on the other hand, process sequences in both forward and backward directions. This bidirectional information flow allows the model to consider not only the contextual information from preceding words in the caption but also information from subsequent words (Zvornicanin, 2023). This enhances contextual understanding for our caption generation.


input_22 InputLayer
input_23 InputLayer
dropout_20 Dropout
embedding_10 Embedding
dense_26 Dense
dropout_21 Dropout
batch_normalization_12 BatchNormalization
bidirectional_5(lstm_11) Bidirectional(LSTM)
activation_3 Activation
bidirectional_6(lstm_12) Bidirectional(LSTM)
concatenate_8 Concatenate
dense_27 Dense
batch_normalization_13 BatchNormalization
activation_4 Activation
dense_28 Dense



```
Layer (type)                Output Shape          Param #   Connected to
==================================================================================
input_22 (InputLayer)       [(None, 4096)]        0         []

input_23 (InputLayer)       [(None, 35)]          0         []

dropout_20 (Dropout)        (None, 4096)          0         ['input_22[0][0]']

embedding_10 (Embedding)    (None, 35, 256)       2157312   ['input_23[0][0]']

dense_26 (Dense)            (None, 256)           1048832   ['dropout_20[0][0]']

dropout_21 (Dropout)        (None, 35, 256)       0         ['embedding_10[0][0]']

batch_normalization_12 (Ba  (None, 256)           1024      ['dense_26[0][0]']
tchNormalization)

bidirectional_5 (Bidirecti  (None, 35, 512)       1050624   ['dropout_21[0][0]']
onal)

activation_3 (Activation)   (None, 256)           0         ['batch_normalization_12[0][0]
                                                            ']

bidirectional_6 (Bidirecti  (None, 256)           656384    ['bidirectional_5[0][0]']
onal)

concatenate_8 (Concatenate  (None, 512)           0         ['activation_3[0][0]',
)                                                            'bidirectional_6[0][0]']

dense_27 (Dense)            (None, 256)           131328    ['concatenate_8[0][0]']

batch_normalization_13 (Ba  (None, 256)           1024      ['dense_27[0][0]']
tchNormalization)

activation_4 (Activation)   (None, 256)           0         ['batch_normalization_13[0][0]
                                                            ']

dense_28 (Dense)            (None, 8427)          2165739   ['activation_4[0][0]']
```

Fig 8. Network introducing bi-directional LSTM layers

In this network, we increased the dropout from 0.2 to 0.5 to reduce the incidence of overfitting. We also employed batch normalization (batch_normalization_12 & batch_normalization_13) to standardize inputs to a layer for each mini-batch. This stabilizes the learning process and dramatically reduces the number of training epochs required to train deep networks. An activation layer (activation_4) introduces non-linearity to the model, allowing it to learn more complex functions.

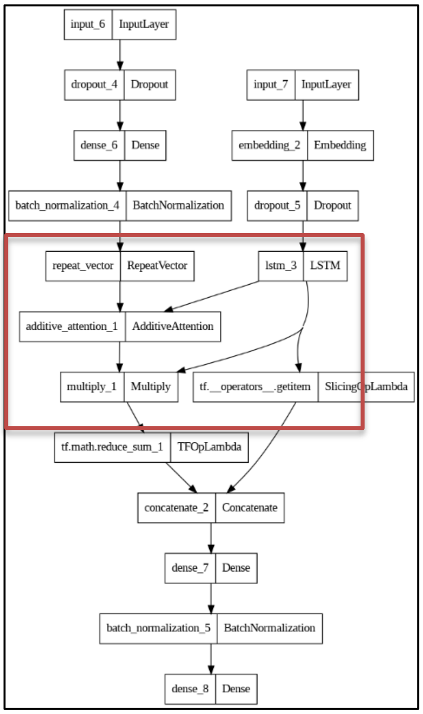
5.3.3 Network 3 (Varying L2 Regularization Constant)

Building on network 2, we increased the L2 regularization constant lambda (λ) from 0.0001 to 0.01 to impose higher penalties on larger weights in the model. This means that during training, the optimization algorithm will prioritize smaller weights, all else being equal, as larger weights will now contribute more to increasing the total loss. The network architecture remains the same as network 2 (Fig 8).

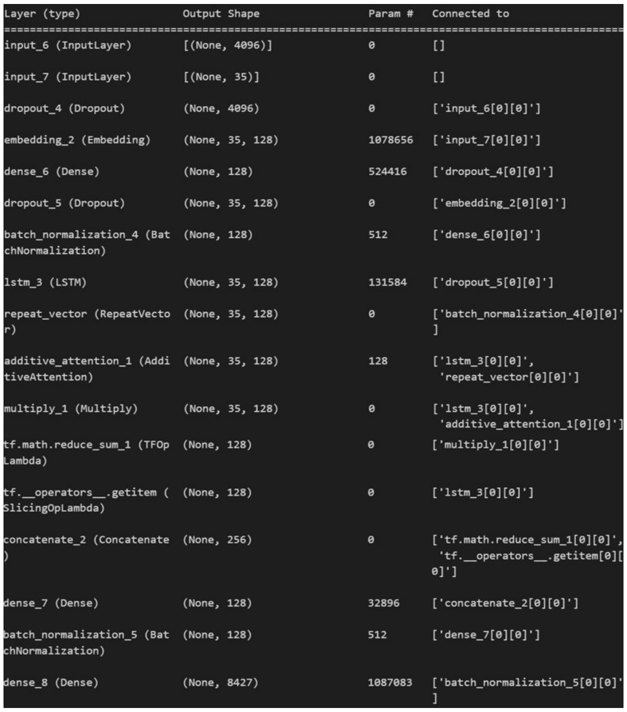
5.3.4 Attention Mechanism

Subsequently, we look to add an attention mechanism into our network (additive_attention). The attention mechanism serves to elevate the performance of deep learning models by concentrating on crucial input elements, thereby enhancing both prediction accuracy and computational efficiency. These mechanisms prioritize pertinent information to augment the overall effectiveness of the model (Xie et al., 2023).

5.3.4.1 Network 4 (Attention Mechanism – LSTM and Image Layers)



input_6 | InputLayer
dropout_4 | Dropout
input_7 | InputLayer
dense_6 | Dense
embedding_2 | Embedding
batch_normalization_4 | BatchNormalization
dropout_5 | Dropout
repeat_vector | RepeatVector
lstm_3 | LSTM
additive_attention_1 | AdditiveAttention
multiply_1 | Multiply
tf.__operators__.getitem | SlicingOpLambda
tf.math.reduce_sum_1 | TFOpLambda
concatenate_2 | Concatenate
dense_7 | Dense
batch_normalization_5 | BatchNormalization
dense_8 | Dense



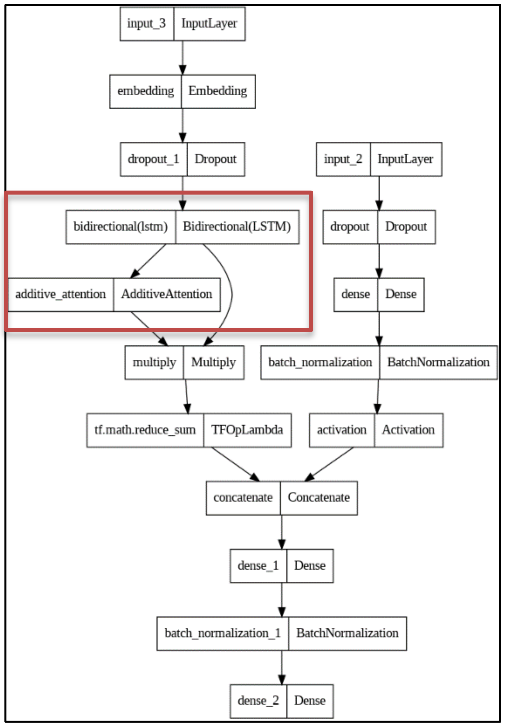
```
Layer (type)                Output Shape         Param #   Connected to
==================================================================================================
input_6 (InputLayer)        [(None, 4096)]       0         []

input_7 (InputLayer)        [(None, 35)]         0         []

dropout_4 (Dropout)         (None, 4096)         0         ['input_6[0][0]']

embedding_2 (Embedding)     (None, 35, 128)      1078656   ['input_7[0][0]']

dense_6 (Dense)             (None, 128)          524416    ['dropout_4[0][0]']

dropout_5 (Dropout)         (None, 35, 128)      0         ['embedding_2[0][0]']

batch_normalization_4 (Bat  (None, 128)          512       ['dense_6[0][0]']
chNormalization)

lstm_3 (LSTM)               (None, 35, 128)      131584    ['dropout_5[0][0]']

repeat_vector (RepeatVecto  (None, 35, 128)      0         ['batch_normalization_4[0][0]'
r)                                                         ]

additive_attention_1 (Addi  (None, 35, 128)      128       ['lstm_3[0][0]',
tiveAttention)                                              'repeat_vector[0][0]']

multiply_1 (Multiply)       (None, 35, 128)      0         ['lstm_3[0][0]',
                                                            'additive_attention_1[0][0]']
tf.math.reduce_sum_1 (TFOp  (None, 128)          0         ['multiply_1[0][0]']
Lambda)

tf.__operators__.getitem (  (None, 128)          0         ['lstm_3[0][0]']
SlicingOpLambda)

concatenate_2 (Concatenate  (None, 256)          0         ['tf.math.reduce_sum_1[0][0]',
)                                                           'tf.__operators__.getitem[0][
                                                           0]']

dense_7 (Dense)             (None, 128)          32896     ['concatenate_2[0][0]']

batch_normalization_5 (Bat  (None, 128)          512       ['dense_7[0][0]']
chNormalization)

dense_8 (Dense)             (None, 8427)         1087083   ['batch_normalization_5[0][0]'
                                                           ]
```

Fig 9. Network with attention mechanism implemented for both LSTM & image layers

When processing an image for captioning, the CNN (VGG16) extracts a feature map of the image. The LSTM then generates a caption sequence, where at each step, the attention mechanism decides which part of the output is more informative and hence focus on (Manu, 2021). This mimics the human ability to focus on specific aspects of a scene when describing it to others. With each new word, the focus shifts, allowing the network to ultimately construct a contextually relevant and accurate caption.

5.3.4.2 Network 5 (Attention Mechanism – LSTM layers only)

For our second implementation of the attention mechanism, we look to primarily apply only on the LSTM layers. This aims to strengthen our existing LSTM model, by introducing our implementation of self-attention. Self-attention is a variant of attention mechanism which focuses on a single sequence (Van Dongen, 2022). This allows greater comprehension of the syntactic function between words of its own generated caption. In this method, relationship between words in a sequence are captured.



input_3 | InputLayer
embedding | Embedding
dropout_1 | Dropout
input_2 | InputLayer
bidirectional(lstm) | Bidirectional(LSTM)
dropout | Dropout
additive_attention | AdditiveAttention
dense | Dense
multiply | Multiply
batch_normalization | BatchNormalization
tf.math.reduce_sum | TFOpLambda
activation | Activation
concatenate | Concatenate
dense_1 | Dense
batch_normalization_1 | BatchNormalization
dense_2 | Dense

Fig 10. Network with attention mechanism implemented for only LSTM layers

In our adaptation, the LSTM's output is used as both the query and key in an additive attention mechanism. We acknowledge that our implementation differs to the traditional self-attention found in Transformer models, where it does not parallelize the attention computation across all elements of the input sequence. Instead, this implementation helps the network focus on different parts of the generated caption sequentially. This still allows the contextualization of textual relationships, and thus possibly resulting in a more coherent caption.

5.3.5 Network 6 (Sparse Network)

Lastly, we tried to decrease our network's complexity by reducing the number of hidden neurons within the layers from network 3 and observe if we could further reduce overfitting. We wanted to see if this network is less likely to learn noise in the training data and more likely to generalize to new, unseen data. This network uses fewer neurons in its layers: Dense(64) & Bidirectional LSTM layers with 64 units and 32 units, versus Dense(256) & Bidirectional LSTM layers with 256 units and 128 units in network 3. We are ultimately experimenting the trade-off between performance (accuracy) and simplicity (to avoid overfitting).

6 Results

Network	Data set	BLEU Score (Unigram)
Baseline (Network 1)	Flicker8k	0.36297300260862025
Bi-Directional (Network 2)	Flicker8k	0.41547861023260674
Varied L2 Regularization (Network 3)	Flicker8k	**0.4733573183019591**
Image + LSTM Attention Mechanism (Network 4)	Flicker8k	0.38040711060112964
LSTM-only Attention Mechanism (Network 5)	Flicker8k	**0.503274195652104**
Sparse (Network 6)	Flicker8k	0.42126351969006637

7 Analysis

Analyzing the BLEU scores achieved by each network on our test set reveals that the inclusion of an LSTM-only Attention Mechanism (network 5) results in the most precise captions, achieving a BLEU score of approximately 0.503. Following closely is a bi-directional LSTM model (network 3) with an increased regularization constant (λ = 0.01) as the next best-performing network.

Expectedly, the least effective performer is our baseline model (network 1), which incorporated a straightforward LSTM architecture. The traditional implementation of an attention mechanism in an image captioning model, as illustrated in network 4, however has an unexpectedly low BLEU score. This can be attributed to limitations of the traditional attention mechanism such as exposure bias and overfitting (Choudhary, 2023). Exposure bias occurs since the model is trained on true target sequence (ground-truth context). However, when tested with the test set, the model may be exposed to different scenarios which forces the model to infer from unfamiliar inputs (Schmidt, 2022).

We will also need to acknowledge the limitations of BLEU score. While our implementation of the self-attention model in Network 5 resulted in the best BLEU score, we understand that it is not comprehensive to judge that as our best network using only 1 metric. Moreover, BLEU has a major blind spot. Since it merely compares the accuracy based on the degree of difference between a generated caption from those of the references, it does not consider the contextual inference. BLEU also does not evaluate take the gravity of each error into consideration (Dupont, 2020). As such, a generated caption can result in a high BLEU score, even if the caption is out of context, but is largely similar to the references. In this case, the LSTM with self-attention network will undoubtedly result in a higher BLEU score since it focuses on relationships between words of the reference captions and thus generates captions that have a more similar phrasing.

It is also to be noted that with the implementation of an attention mechanism, the resultant network is very complex. Network 5 currently contains 64 hidden units and has a batch size of 64. When we attempted to make the network denser by increasing the number of hidden units to past 64, even a V100 GPU was unable to allocate enough memory. This shows that even for a high-end GPU, the network might be too complex to handle efficiently, and hence is too computationally expensive and resource draining.

8 Conclusion

Our project successfully demonstrates the efficacy of deep learning in the field of image captioning. The top-down approach combining a CNN encoder (VGG16) and LSTM decoder has proven to be very effective. Through trial and error, our group managed to experiment with different network configurations and enhancements, leading to significant improvements in our image captioning network's performance from when we first started the project. After analysis, we conclude that network 3 is the most optimal hybrid network to perform image captioning. To reiterate, network 3 uses a VGG16 model as the encoder, has a dropout of 0.5, bidirectional LSTM layers as the decoder, and L2 Regularization constant 0f 0.01.

Additionally, one key takeaway from this project is that creating a network that is overly complex can very easily lead to overfitting, a common pitfall in machine learning. To prevent this, it is essential that we adjust various factors such as number of layers or hidden neurons and incorporate strategies such as L2 regularization. It is important that we strike a delicate balance between performance in terms of accuracy and generalization.

9 References

Al-Malla, M.A., Jafar, A. & Ghneim, N. (2022). Image captioning model using attention and object features to mimic human image understanding. *Journal of Big Data*, *9*(1). https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00571-w#citeas

Parameswaran, S. N., & Das, S. (2018). A Bottom-Up and Top-Down Approach for Image Captioning using Transformer. *ICVGIP '18: Proceedings of the 11th Indian Conference on Computer Vision, Graphics and Image Processing*, 38. https://doi.org/10.1145/3293353.3293391

Mishra, M. (2020, August 27). *Convolutional Neural Networks, Explained – Towards Data Science*. Medium. https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

Saxena, S. (2023, October 25). *What is LSTM? Introduction to Long Short-Term Memory.* Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/

Olah, C. (2015, August 27). *Understanding LSTM Networks.* Colah's Blog. https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Rohini, G. (2023, September 23). *Everything you need to know about VGG16.* Medium. https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918

Zvornicanin, E. (2023, June 8). *Differences Between Bidirectional and Unidirectional LSTM.* Baeldung. https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm#:~:text=Bidirectional%20LSTM%20(BiLSTM)%20is%20a,utilizing%20information%20from%20both%20sides

Xie, T., Ding, W., Zhang, J., Wan, X., & Wang, J. (2023). Bi-LS-ATTM: A Bidirectional LSTM and Attention Mechanism Model for Improving Image Captioning. *Applied Sciences, 13*(13), 7916. https://doi.org/10.3390/app13137916

Manu. (2021, January 30). *A simple overview of RNN, LSTM and Attention Mechanism – The Startup*. Medium. https://medium.com/swlh/a-simple-overview-of-rnn-lstm-and-attention-mechanism-9e844763d07b

Van Dongen, T. (2022, November 7). *Demystifying efficient self-attention – Towards Data Science.* Medium. https://towardsdatascience.com/demystifying-efficient-self-attention-b3de61b9b0fb#:~:text=Self%2Dattention%20is%20a%20specific,sequence%20learn%20information%20about%20itself.

Schmidt, F. (2019, November 4). *Generalization in Generation: A closer look at Exposure Bias. Proceedings of the 3rd Workshop on Neural Generation and Translation (WNGT 2019)*, 157–167. https://doi.org/10.18653/v1/d19-5616

Choudhary, A.S. (2023, June 27). *Learn Attention Models From Scratch.* Analytics Vidhya. https://www.analyticsvidhya.com/blog/2023/06/learn-attention-models-from-scratch/#:~:text=The%20major%20drawbacks%20are%3A,of%20data%20and%20computational%20resources.

Dupont, A. (2022, November 9). *MT for Beginners: What is BLEU and what is wrong with it?* https://www.lengoo.com/blog/mt-for-beginners-what-is-bleu-and-what-is-wrong-with-it/