[Team 32] Image Caption Generation with Merge Networks

CHAITANYA RAJEEV
Email: crajeev@ncsu.edu

SAI SASHANK NAYABU
Email: snayabu@ncsu.edu

LAXMI AISHWARYA THELA
Email: lthela@ncsu.edu

I. METHODOLOGY

Humans have long mastered the ability to use our vision to segment, identify and interact with the world around us. While we may be able to recognize objects and faces instantaneously with ease, one must know that it has taken quite a long time for our neural cortex to evolve and develop this ability.

In this project, we aim to partially replicate this ability by training an algorithm that would be able to take an image as an input and provide a suitable caption that describe activities and subjects within the image in a natural tone that is linguistically similar to how a human would describe a similar image.

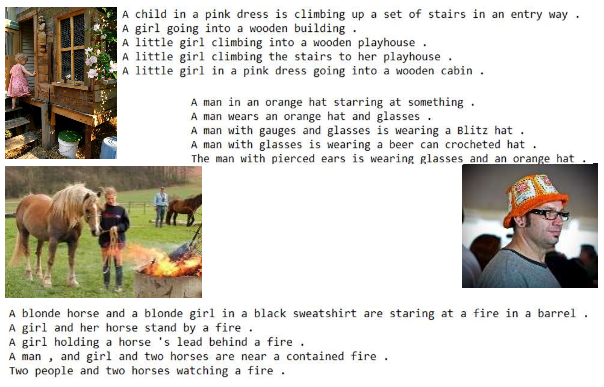
In order to train the algorithm, we would need as data, a large set of images with one or more captions/descriptions for each image that we can simultaneously show to the deep learning architecture that will be built. To achieve this, both Recurrent layers and Convolutional layers will be required. The Recurrent layers are good at processing language data and convolutional layers are great a capturing features within images.

A) Dataset Description

The Flickr8k data was chosen for training the algorithm. It was compact in size which allowed training to be done on a home PC which was convenient.

The dataset consists of 8092 images each having a set of 5 captions that describe subjects and activities within the image. A few examples for images and captions are shown in Figure 1.

The captions were stored with their respective image filenames in a separate text file which was part of the data bundle.



```
A child in a pink dress is climbing up a set of stairs in an entry way .
A girl going into a wooden building .
A little girl climbing into a wooden playhouse .
A little girl climbing the stairs to her playhouse .
A little girl in a pink dress going into a wooden cabin .

A man in an orange hat starring at something .
A man wears an orange hat and glasses .
A man with gauges and glasses is wearing a Blitz hat .
A man with glasses is wearing a beer can crocheted hat .
The man with pierced ears is wearing glasses and an orange hat .

A blonde horse and a blonde girl in a black sweatshirt are staring at a fire in a barrel .
A girl and her horse stand by a fire .
A girl holding a horse 's lead behind a fire .
A man , and girl and two horses are near a contained fire .
Two people and two horses watching a fire .
```

Figure 1: Examples from the Flick8k dataset

B) Data Pre-processing

1. *Image Pre-Processing*

The data was pre-processed in two distinct phases, one each for text pre-processing and image pre-processing. For image pre-processing, all images were first imported and converted into 224 X 224 X 3 tensors. This was done to make the image compatible with the input format of various transfer learning architectures.

2. *Text Pre-Processing*

For text pre-processing, the .txt file containing captions was loaded onto memory from which the individual images names were separated and discarded. This step returns a dictionary that matches the photo names to their respective captions.

The next step involved carrying out specific steps to reduce redundancy in tokenization. The steps are detailed as follow:

- All words converted to lower case.
- All punctuations were removed.
- Words with a length of one character or less were removed.
- All words with numbers as any character were removed.

After this, the next step was to generate a vocabulary of words used in the cleaned text corpus. This was done by creating a dictionary of unique words in the corpus. The cleaned vocabulary of the Flickr8k dataset consisted of 8,763 unique words.

The last step was tokenizing the vocabulary. This step involves encoding each word in the dictionary using a unique integer or a vector based on word embeddings. For this step, we chose to go with a simple one-hot encoding of the dictionary.

C) Modelling Approach

1. *Feature Extraction*

The first step in the modelling approach involves extracting feature vectors from every image by passing it through the layers of a pre-trained network optimized for object detection. This method allows for the generation of more nuanced and detailed captions which recognizes objects within the image without the need for extensively training our own Convolutional neural network.

In essence, we use different transfer learning learning architectures (displayed in Figure 2) like VGG16, Resnet-50 and InceptionV3 which are known to perform well on object classification tasks, to identify objects within the image and pair this ability with the generation of grammatically correct words.

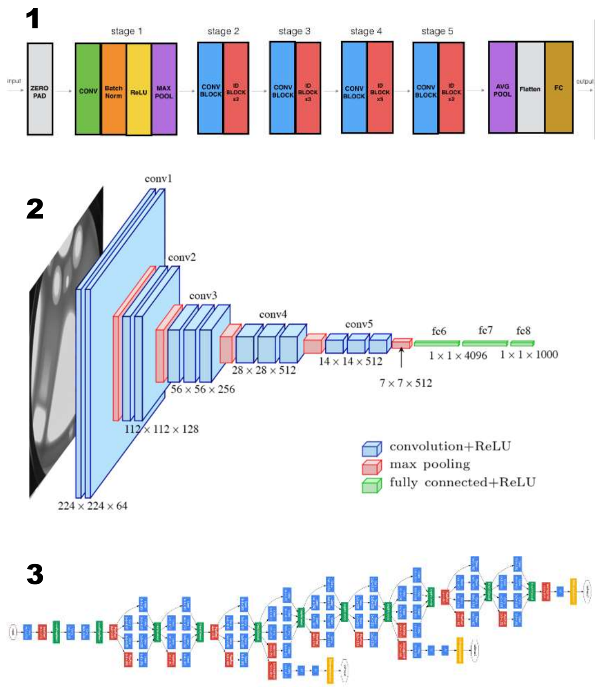


Figure 2: 1) Resnet-50 2) VGG16 3) InceptionV3

2. *Merge-model (Baseline)*

The feature extracted vector is then passed through one sub-chain of a merge-model architecture shown in Figure 3. The merge-network consists of two sub-chains merged into a set of fully connected layers.

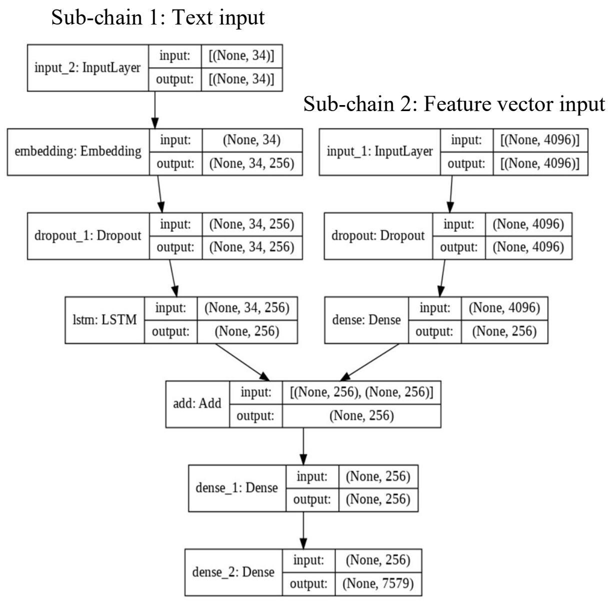

Sub-chain 1: Text input
input_2: InputLayer | input: [(None, 34)] | output: [(None, 34)]
embedding: Embedding | input: (None, 34) | output: (None, 34, 256)
dropout_1: Dropout | input: (None, 34, 256) | output: (None, 34, 256)
lstm: LSTM | input: (None, 34, 256) | output: (None, 256)
Sub-chain 2: Feature vector input
input_1: InputLayer | input: [(None, 4096)] | output: [(None, 4096)]
dropout: Dropout | input: (None, 4096) | output: (None, 4096)
dense: Dense | input: (None, 4096) | output: (None, 256)
add: Add | input: [(None, 256), (None, 256)] | output: (None, 256)
dense_1: Dense | input: (None, 256) | output: (None, 256)
dense_2: Dense | input: (None, 256) | output: (None, 7579)

Figure 3: Merge-model architecture

The first sub-chain takes encoded captions as input and processes them through an embedding and an LSTM layer. The second sub-chain takes as input, the feature extracted vector which was output from the transfer learning architecture. These two sub-chains are then merged into two dense layers from which the categorical cross-entropy loss of the activations are measured against the one-hot encoded vector of the next word in the caption.

II. Model Training and Selection

In this section, we shall discuss the pre-processing steps taken to feed data into the model for training. We shall also discuss the process of choosing the optimal transfer learning architecture and merge-model architecture, the hyper-parameter tuning process and present relevant plots.

A) Model Training

1. Training Illustration and Loss Evaluation

The training procedure for the model is described with an example in Table 1.

Actual Caption: A girl is sitting.			
Iteration	**X1**	**X2 (text sequence)**	**Y (word)**
1	photo	startseq,	A
2	photo	startseq, A	girl
3	photo	startseq, A, girl	is
4	photo	startseq, A, girl, is	sitting

Every caption and photo pair are trained for a number of iterations equal to the no. of words in each caption. X1 is the feature extracted vector from the photo fed into sub-chain 2 (refer Figure 3). X2 is the portion of the caption fed into sub chain 1. In the first iteration, X2 consists of the numerical encoding to start the sequence prediction (startseq) and Y is the first word in the caption. In the next iteration, X2 is updated to be [startseq, first word] and the second word in the caption is Y. In every iteration, the gradients are computed, and weights are updated. This procedure is carried out until Y is endseq, which is the numerical encoding that tells the network that the caption generated has ended.

For evaluating loss, the categorical cross-entropy loss is used. Since for any photo-caption pair at any iteration, the Y value is just one word from the dictionary, the truth vector is a binary sparse vector with length equal to the length of the vocabulary with a single element 1 at the position of the Y word and the rest of the elements as zeros. The activations generated by the merged second dense layer predicts the probability of each word in the dictionary being Y for that iteration.

As an example, for a dictionary of 4 words, [‘a’, ‘girl’, ‘is’, ‘sitting’], if the Y value is ‘girl’, then the truth vector is [0,1,0,0]. The neural network will have to predict the probabilities of the Y word being each word in the dictionary which could be a vector that looks like [0.3, 0.86, 0.1, 0.2]. With this prediction vector and the truth vector, the categorical cross-entropy can easily be calculated, and the gradients computed which could be used to optimize the network weights.

B) Model Selection

1. The BLEU score

The Bilingual Evaluation Understudy Score (BLEU) is a metric used to evaluate the level of match between any given sentence and a reference sentence.

$$\mathrm{BLEU} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

where,

$$\mathrm{BP} = \begin{cases} 1 & \text{if } c > r \\ \exp\left(1 - \frac{r}{c}\right) & \text{if } c \leq r \end{cases}$$

n is the number of words considered on grams. w_n is a weight between 0 and 1. p_n is the precision for the nth gram. Lastly, BP is a penalty that penalizes short translations. More details on the computational aspects of the evaluation score are available in the paper authored by Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu referenced later.

2. Validation losses for various architecture combinations

As mentioned previously, three different architectures were evaluated for feature extraction namely VGG16, Resnet-50 and InceptionV3. For choosing the feature extraction architecture we compared loss charts of each with the baseline merge-model and a couple of its variations. These variations include converting LSTM layers in Sub-Chain 1 to Bi-Directional LSTM layers and also adding additional LSTM layers (referred to as Bi-LSTM for two layers and not to be confused with Bi-Directional LSTM layer). The validation losses of each architecture are plotted in Figure 4.

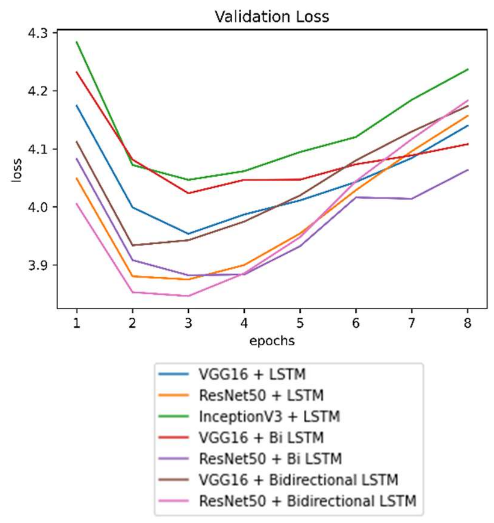


Figure 4: Validation losses for various model combinations

Note that all models achieved their lowest validation losses around Epoch 3. Validation loss was not the only criteria for selection. The BLEU score was also used to short-list and select the final combination of architectures.

3. Hyper-parameter tuning

Now that we have around 10-15 different combinations of the feature extraction and merge-model architectures, to keep computations simple and easy, we decided to investigate the effects of Dropout regularization for some of the better performing model combinations. Validation losses plotted as a result of this exercise are displayed in Figure 5.

From the plots, it is clear that a dropout value of 0.5 has a negligible effect on the model validation error as opposed to the transfer learning architecture. Similar patterns were discerned looking at the plots of other model combinations as well (not shown). Hence, we chose to go with a dropout rate of 0.5 for our final model.

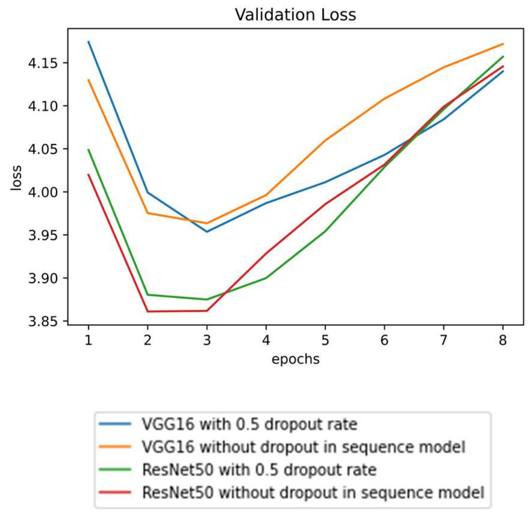


Figure 6: Validation losses for different Dropout rates

4. Final Model and Optimizer Selection

The BLEU scores of the top 6 model combinations including the baseline are presented in Table 2.

S.No.	**Model**	**Activation**	**BLEU -1**	**BLEU -2**	**BLEU -3**	**BLEU -4**
1	Merge Dual LSTM, VGG16	ReLU	0.573	0.31	0.21	0.094
2	Merge BiDir-LSTM, VGG16	sigmoid	0.584	0.344	0.243	0.12
3	Merge Dual LSTM, Resnet-50	ReLU	0.567	0.314	0.215	0.101
4	Merge BiDir-LSTM, Resnet-50	ReLU	0.561	0.307	0.209	0.096
5	Merge Single LSTM, InceptionV3	ReLU	0.504	0.243	0.143	0.059
6	Merge Single LSTM, VGG16(baseline)	ReLU	0.501	0.265	0.180	0.081

From the above table, the highest BLEU scores were reported by the VGG16 feature extraction architecture and the merge Bi-Directional LSTM merge-model with a dropout rate of 0.5 and the sigmoid activation.

In choosing the activation function, three different functions were tested for the best model. The best BLEU scores were obtained with the sigmoid activation function hence that was our choice of activation function.

In many cases, the best performance on BLEU scores didn't translate to the best performance on the validation loss. This is clearly shown in Figure 6 where the sigmoid activation function does not have the lowest validation loss.

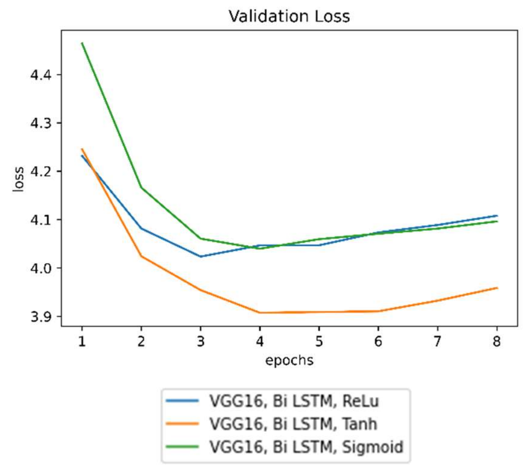


Figure 6: Validation losses for various activation functions

In these cases, we chose to go with the results of the BLEU scores as the BLEU score was better at quantifying the degree of match between two sentences and was specifically developed for evaluating models that generate text output.

III. EVALUATION

In the final section we shall summarize the results of our baseline model, a new comparison model and the best model that we obtained in the previous sections.

A) *Model Summaries*

We have chosen 'Where to put the Image in an Image Caption Generator' by Marc Tanti et.al.[3] as our comparison model.

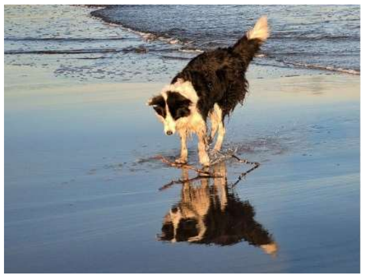
The original baseline model consisted of a VGG16 feature extraction architecture and a single LSTM layer in sub-chain 1 for text processing. The dropout rate was 0.5 and the ReLU activation was used for the last layer.

Our best model combination turned out to be the VGG16 feature extraction architecture paired with the merge-model where the LSTM layer was replaced with a single Bi-Directional LSTM layer. The best dropout rate was chosen to be 0.5 and the activation function chosen was the sigmoid function. The best model gave us BLEU-1, BLEU-2, BLEU-3 and BLEU-4 scores of 0.584, 0.344, 0.243, and 0.12 respectively. The BLEU-1 score of our model is better than that of the baseline model and BLEU-3 score is almost the same for both the models.The following are the BLEU scores for our comparison and best models.

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Comparison model by Marc Tanti et.al.[3]	0.569	0.378	0.249	0.162
Final model	0.584	0.344	0.243	0.12

B) *Prediction on new photos (Baseline vs final model)*

In the final section, a few examples for new caption predictions are shown to illustrate how the final model performs better than the baseline model. Figure 7 shows the captions generated by both baseline and final model for the same picture.

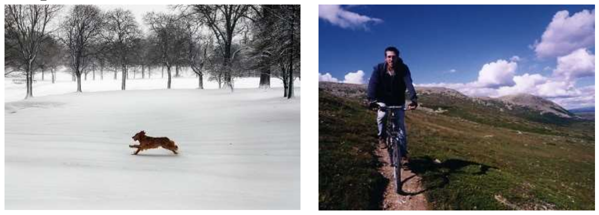


Baseline model caption:
Two dogs are running through the water.
Final model caption:
Black dog is running through the water.

Figure 7: Caption generation comparison

As shown above, the final model performs the captioning task better as it correctly identifies only one dog whereas the baseline model thought that the shadow of the dog was another dog.

In Figure 8, a few more examples of new, captioned images are presented.



Caption:
Brown dog is running through the snow.

Caption:
Man is riding bike on the hill.

Figure 8: New photo captioning examples

As seen above, the final model generates very good captions for the above examples. Its BLEU scores are better than the baseline model and quite similar to the comparison model by Marc Tanti. One point to note was that the new images were chosen keeping in mind similar subjects seen in the training images of the Flickr8k dataset.

A more general image caption generator would need much more training data on much larger datasets to perform well over a wide variety of images and captions.

REFERENCES

[1] https://machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/

[2] Papineni, Kishore, Roukos, Salim, Ward, Todd and Zhu, Wei-Jing. "BLEU: a method for automatic evaluation of machine translation.", 2002.

[3] Tanti, Marc et al. "Where to put the image in an image caption generator." *Natural Language Engineering* 24 (2018): 467 - 489