[Team 41] Image Caption Generator using LSTMs and Recurrent Neural Networks

Aarohi Joshi
ajoshi8@ncsu.edu

Binoy Thomas
bthomas7@ncsu.edu

Rishal Shah
rshah27@ncsu.edu

***Abstract*—Automated captioning of images is a challenging problem in Artificial Intelligence because it demands an understanding of the fields of Computer Vision as well as Natural Language Processing. The generated captions must accurately encapsulate the essence of the image in a human-readable description. An RNN is viewed as a generative component and that is the view in most literatures. We look at the RNN as a component which is used only for encoding the words that have been generated previously. This view suggests that the RNN ust only be used for encoding the words and the images must be merged with the encoding at a later stage. We have implemented this model and presented the generated results in this paper. The BLEU score is used a factor to evaluate the quality of the results. The main purpose of this paper is to show that using the RNN as a component for encoding rather than a generative component is much easier than the latter.**

I. Model Training and Selection

Many approaches to image captioning are present in the world. We use a model that relies on a neural model which generates cations using a Recurrent Neural Network(RNN) usually a Long short-term memory(LSTM) rather than perform a wholesome cation retrieval.

We define a deep learning model based on the merge model described by Marc Tanti et al. [1]. The paper defines two methods to use an RNN. We use the 'merge' method to generate captions for images. The key property of these models is that to condition the predictions of the best caption to describe the image, the CNN image features are used. The role of the RNN depends in large measure on the mode in which CNN and RNN are combined. This can be done in different ways. We define the methodology as presented in Fig. 1. In

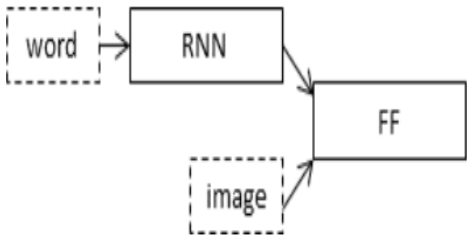

Fig. 1. Merging the word embedding with the image

this method, the liguistic features and the perceptual features are kept seperate and the are joined together at a later stage after encoding the liguistic features with the help of an adder. The RNN is functioning as an encoder of sequences of word embeddings, with the visual features merged with the linguistic features in a later, multimodal layer. The RNN never sees the image and hence would not be able to direct the generation process.

A. Model

According to the ideas presented in the paper we modelled a structure which was of the form in Fig.2.

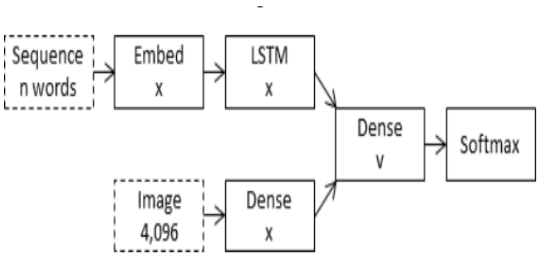

Fig. 2. The merge architecture

The model is described as having three parts

1) Photo extractor: The feature extraction model is a neural network that given an image is able to extract the salient features, often in the form of a fixed-length vector. The features that are extracted are salient and hence are not directly intelligible. In order to this a CNN is used. We used the VGG16 model to get these results. We use various other models like VGG19 and RESNET50 but we decided to use the VGG16 model. The network was trained on state of the art models because in order to extract the features these models are required. For the VGG16 model, we separated the last softmax layer because VGG models are used for classification and this is not the purpose of our project. What we need are features of the images. Hence we use the features from the penultimate layer.

2) Sequence processor: This is the embedding layer which processes the captions of the images in the training data set and splits it into words which are fed into the merging layer for training. This will be done using the Long-Short Term Memory model(LSTM) and Recurrent neural network. A language model predicts the probability of the next word in the sequence given the words already present in the sequence. The laguage model is a model taht is able to generate a combination of the description of the words given the already generated

words describing the image. Each output time step generates a new word in the sequence. Each word that is generated is then encoded using a word embedding and passed as input to the decoder for generating the subsequent word.

3) The merging layer: The photo extractor and the Sequence processor both output a vector. This is put through a Dense layer to train the model. This model is used for prediction and generating captions for images.

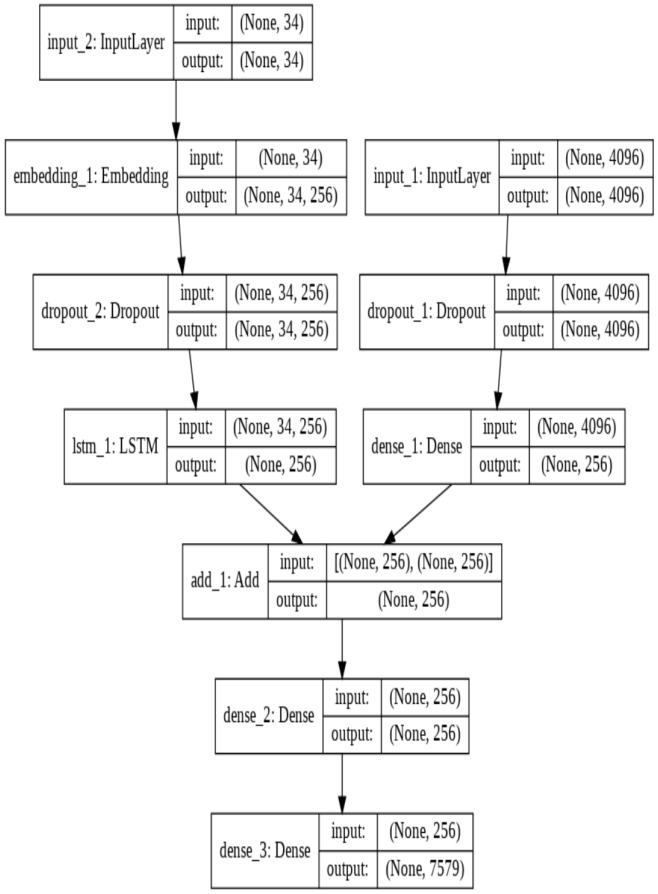

Fig. 3. The model architecture

The Photo Feature Extractor model expects input photo features to be a vector of 4,096 elements which were already processed from the VGG16 model. The vectors are then processed by a Dense layer to produce a 256 element representation of the photo. The Sequence Processor model gets input sequences with a 34- word length word which are fed into an Embedding layer that uses a mask to ignore padded values. This is followed by an LSTM layer with 256 memory units. Both the input models produce a 256 element vector. Further, both input models use regularization in the form of 50 percent dropout to reduce over fitting. The learning rate for this model is very fast. The Decoder model uses an addition operation to merge both the linguistic data and the perceptive data. This is then fed to a Dense 256 neuron layer.Finally, they are fed to final output dense layer which contains all the words in the dataset. This is softmax layer and as seen in the image it has about 7519 words to predict the next word that is generated.

B. Baseline

Moses et al. [2] developed a CNN model for caption generation and described his approach in a paper with the title "Learning CNN-LSTM Architectures for Image Caption Generation".

We have considered this paper as a baseline and improved upon it using the method proposed in the above section. We have compared the results that we have obtained with the results from the above paper.

This paper describes two types of approaches for the task of caption generation. Namely, top-down approach and bottom-up approach.

1) Bottom-up approach: The model first generates the items observed in an image, and then attempts to combine these identified items and convert them into a caption.
2) Top-down approach: The model attempts to generate a semantic representation of an image that is then decoded into a caption using various architectures.

This paper has implemented the top-down approach using a deep CNN to generate a vectorized representation of an image that is given as input to the LSTM to generate the caption.

They fed an image into the CNN which produced an encoding that was in-turn fed into the LSTM for decoding. As the words began to emit, the current predicted sentence was fed back into the LSTM for it to predict the next word. As each new predicted word was fed into the LSTM, the hidden state was recorded that resulted from the combination of the new word and the previous hidden state.

This paper also explains the max-out limit of the single

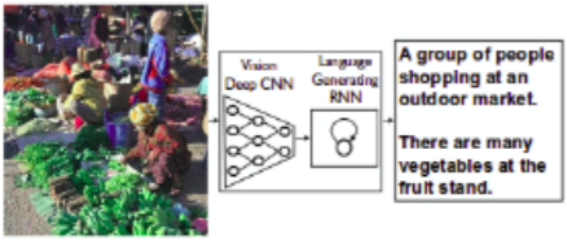

Fig. 4. The CNN-LSTM architecture

layer decoder LSTM network and the effects of increasing the number of layers of the decoder LSTM network on the captions generated. It was found that adding additional layers leads to improvements in the BLEU-4 metric. But, adding additional layers beyond a second layer did not show

improvements as the model began to overfit.

This approach described by Moses et al. [2] has two major drawbacks:

1) Overfitting: This was a drawback discovered after adding additional layers beyond the second layer to the decoder LSTM network.
2) Failure to capture minute details: This is the primary drawback of Moses's model. The captions generated by the top-down approach were unable to capture the minute details of the image. For example, an image of a man on a skateboard on a ramp received the same caption as an image of a man on a skateboard on a table.

To overcome these drawbacks, we have referred to Tanti's [2] merging RNN model for caption generation which separately encodes the image and description in a layer to be decoded later generating the next word of the caption.

II. Experimental Section

A. Metrics

For evaluating the performance of our model, we have used BLEU (Bilingual Evaluation Understudy) metric that compares the machine generated summaries with human reference summaries.

BLUE measures precision: how many words/phrases in the machine generated summaries appeared in the human reference summaries.

The BLEU score measures the precision by checking how many words or phrases in the machine generated caption appeared in the human reference summaries. It essentially is a string-matching algorithm.

BLEU's biggest strength is that it correlates with human reference summaries by averaging out individual sentence errors over a test corpus, instead of attempting to devise the exact human reference summarie for every sentence. This is how it ensures the retention of the meaning whilst being flexible with the language.

There are 4 BLEU scores generated for every model which refer to the cumulative n-grams.

B. Model Selection

To select the hyperparameters, we first decided to choose the model for extracting the features from the images by testing the various pre-defined models of Keras that have been trained with large datasets like ImageNet. We compared five models and used the Validation Loss as the deciding metric. The five models that were used to test were VGG16, VGG19, InceptionV3, ResNet50 and InceptionResNetV2. Fig 5 shows that the VGG16 model has the lowest average validation loss and performs the best amongst all the models for a fixed set of hyperparameter values.

The BLEU Score obtained from each of the models in Table I reiterates that VGG16 is the best performing model amongst all the models.

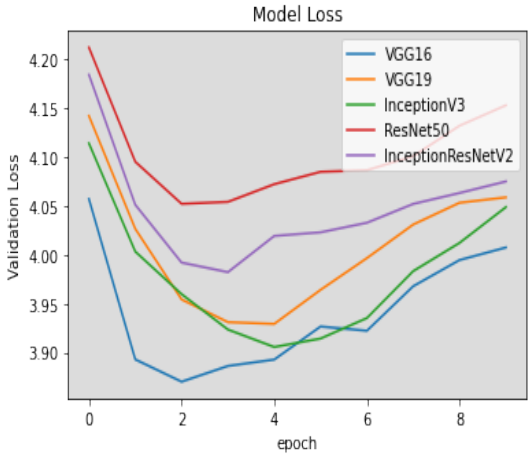

Fig. 5. Validation Loss vs Epochs for different feature extraction models

Model	BLEU1 Score	BLEU2 Score	BLEU3 Score	BLEU4 Score
VGG16	0.565	0.3342	0.2377	0.164
VGG19	0.5169	0.2554	0.1661	0.0932
InceptionV3	0.5283	0.284	0.199	0.1003
ResNet50	0.4533	0.1971	0.1322	0.0634
InceptionResNetV2	0.5069	0.2398	0.1634	0.0861

TABLE I
BLEU Score comparison of various feature extraction models

On finalizing the model for feature extraction, we decided to vary the dropout parameter between 0.25 to 1. The model was repeatedly run for different dropout values keeping the other hyperparameters constant. The dropout values tested were [0.25, 0.50, 0.75, 0.87, 1]. Fig 6 plots the Validation Loss obtained for different values of the Dropout parameter. We found that the average validation loss decreased when set from 0.25 to 0.5 but then started increasing when changed from 0.5 to 1. Thus the dropout parameter was set to 50%.

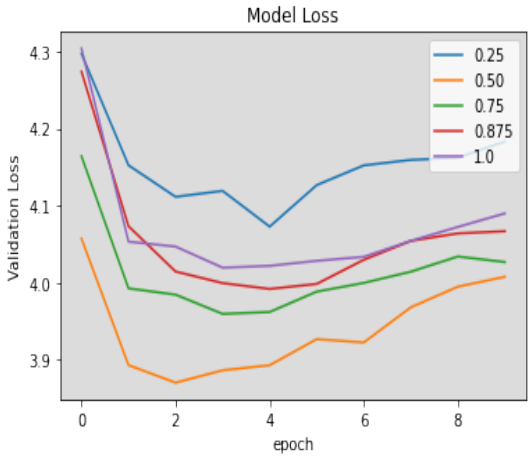

Fig. 6. Validation Loss vs Epochs for different Dropout values

Here too the BLEU Score obtained from various values

Dropout	BLEU1 Score	BLEU2 Score	BLEU3 Score	BLEU4 Score
0.25	0.4232	0.1988	0.0813	0.0312
0.50	0.565	0.3342	0.2377	0.164
0.75	0.5422	0.2994	0.1855	0.1449
0.87	0.5239	0.281	0.1824	0.1253
1.0	0.4911	0.2672	0.1693	0.1058

TABLE II
BLEU SCORE COMPARISON OF DROPOUT FOR VGG16 MODEL

of the Dropout parameter shown in Table II shows that the parameter set to 50% gives the best BLEU scores amongst all the values of the Dropout parameter.

The number of Epochs was set to 10 because for most of the models, the validation loss improved until epoch 4-7, but then started decreasing from epoch 8 on-wards. The number of LSTM layers were set as per Marc Tanti at. el. [1] model architecture as depicted from Fig 3. The optimizer was set to adam for training our neural network and the learning rate was set to 0.001. The loss function was set to categorical_crossentropy as this is a multi-class problem. Hence Table III shows the final set of hyperparameters for our model.

Hyperparameter	Value
Feature Extraction Model	VGG16
Epochs	10
Dropout	0.5
LSTM Layers	256
Optimizer	adam(1e-3)
Loss Function	categorical_crossentropy

TABLE III
HYPERPARAMETER SELECTION

C. Performance and Comparison to Baseline

Some of the outputs that we obtained for the images in the Flickr8k dataset are presented below.

Fig. 7. Man standing near a lake

Fig. 8. Man in black shirt is standing on the street

Fig. 9. Two dogs are running through the grass

We compared our results to the expected outcomes presented in the paper be Mark Tanti et. al. and with the results using just a CNN and we discoved that the results obtianed by us were not quite as good as the CNN model but it was in the higher range of the results obtained in the original papaer. A comparitive table is given in Table 1. It is seen that the results are very good.

Metrics(BLEU score)	Model(Mark Tanti et al.)	Model(VGG16 +LSTM)	CNN model(Moses Soh et al.)
BLEU-1	0.401 to 0.578	0.565	~ 0.61
BLEU-2	0.176 to 0.390	0.3342	~ 0.43
BLEU-3	0.099 to 0.260	0.2377	~ 0.39
BLEU-4	0.059 to 0.170	0.164	0.244

TABLE IV
FINAL BLEU SCORES

From the results obtained it can be inferred that the results using the CNN model are better because it is trained on a

robust model. THis model is much difficult to train and is a large model with many layers. Out model using the VGG16 and the RNN LSTM is much easier to train and gives very good results.

When given a neural network architecture that is expected to process input sequences from multiple modalities, arriving at a joint representation, as it is our case where we need to get linguistic data as well as perceptive data, it would be better to have a separate component to encode each input, bringing them together at a late stage. It is better than passing them all into the same RNN through separate input channels.

REFERENCES

[1] Tanti, Marc, Albert Gatt, and Kenneth P. Camilleri. "What is the Role of Recurrent Neural Networks (RNNs) in an Image Caption Generator?." arXiv preprint arXiv:1708.02043 (2017).

[2] Soh, Moses. "Learning CNN-LSTM architectures for image caption generation." Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, Tech. Rep (2016).