Image Caption Generator

Akhil Kumar Mengani amengan@ncsu.edu
Pranav Venkata Singaraju pvsingar@ncsu.edu
Samson Reddy Mulkur smulkur@ncsu.edu
Sai Naga Vamshi Chidara schidar@ncsu.edu

1 Introduction

In recent years, image captioning has gathered widespread attention, which involves generating a concise description of an image. This task involves Computer Vision (CV), Natural Language Generation (NLG), and Machine Learning (ML) methods. Deep learning is rapidly advancing and one of the most researched fields of study that makes its way into many aspects of our daily lives. It is a sub field of machine learning concerned with algorithms and is inspired by the structure and the functioning of the brain.

Image Captioning is the process of generating a brief textual description of an image. The process involves both Computer Vision and Natural Language Generation[1]. In this project, we built a model that generates a concise natural language description of an image. We used the Convolutional Neural Networks to extract features and then the Recurrent Neural Networks (Long Short Term Memory) to generate text from these features. We have built and trained a model which generates a caption that nearly describes the image. We have used Microsoft Common Objects in Context (MS COCO) dataset [2] for this project which has class labels, labels for different segments of an image, and a set of captions for a given image.

2 Related Work

Describing images with natural language is a problem that has gained broad interest recently because of advances in the recognition of objects, their attributes and locations. Different approaches have been tried in the past to generate text that describes an image. In this section, we have a look at some of the work previously undertaken for this problem. These approaches depend on templates rather than probabilistic models for image caption generation [3][4].

Farhadi et al [5] uses triplets of object, action and scene with pre-determined templates to generate caption. A multilabel Markov Random Field is discriminately trained to predict the values of the triplets. Farhadi et al [5], identify objects in the image, predict a set of attributes with spatial information against other objects for every object, create a labelled Conditional Random Field graph and generate sentences using the labels and a template. These approaches fail to describe the previously unseen composition of objects even if the individual objects were present in the training data. In other words, these approaches do not generalize well. Another problem with approaches involving templates is their proper evaluation.

To overcome these issues, deep neural network architectures are used together with a language model. The language model embeds the images and captions in one vector space. The basic approach is essentially the same-a Convolutional Neural Network (encoder) that generates a sequence of features that are fed into a sequence-to-sequence model (decoder) that learns a language model to generate natural language descriptions. We have followed the basic approach taken by Show and Tell [1] which uses GoogLeNet CNN and LSTM to extract image features and generate captions. In our project, we have taken a simple ResNet50 and LSTM architectures for the same. We have used ResNet as it has better accuracy compared to GoogleNet and also, we have used ResNet50 due to its simple structure which has 50 layers.

3 Method

In this section, we describe about Convolutional Neural Network and Recurrent Neural Network architecture which are used to generate captions from images. Image data is mapped to an output variable using Convolutional Neural Networks. They've proven to be so successful that they're now the method of choice for any form of prediction problem utilizing image data as an input. RNNs, or recurrent neural networks, were created to solve sequence prediction problems. The most successful RNNs are LSTM networks, which allowed us to encapsulate a larger sequence of words or sentences for prediction. We mixed these two neural network models (CNN and RNN) to build a hybrid model. Thus we accomplished the challenge of creating a human-readable textual description of an image with a hybrid model of computer vision and natural language processing.

3.1 Convolutional Neural Network

Convolutional Neural Network (CNN) takes an image as input and assigns weights and biases to various objects in the image. These weights and biases helps it differentiate one image from the other. One of the most popular applications of CNN is image classification. The neural network we created consists of convolutional layers together with nonlinear and pooling layers. Our architecture consists of Conv2D layers along with max pool and Relu activation function. The input image is passed through the network such that the output of one layer becomes the input to the next layer. The last layer is fully connected layer. Attaching a fully connected layer to the end of the network results in an N dimensional vector, where N is the number of classes from which the model selects the desired class.

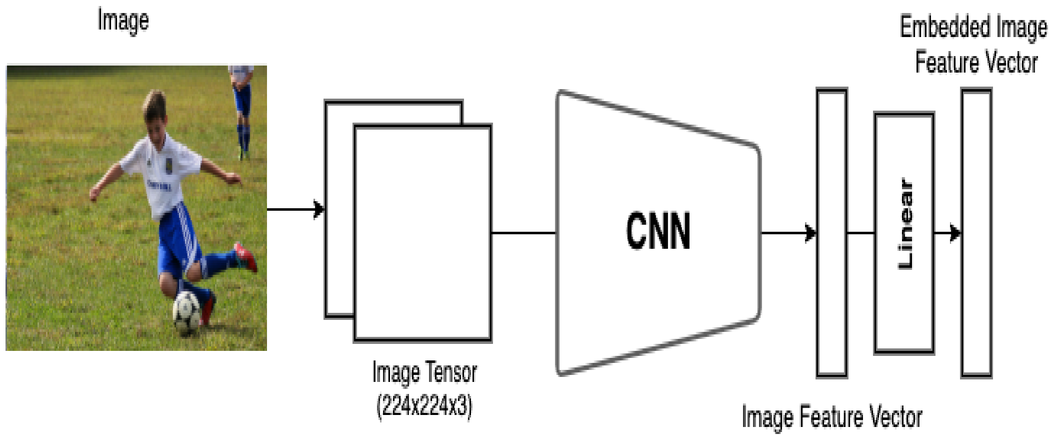


Figure 1: CNN Architecture

3.2 Text Generation using LSTM

Because there might be delays of undetermined duration between key occurrences in a time series, we used LSTM [6] networks for to categorizing, processing, and generating predictions based on time series data. We developed LSTM network to address the vanishing gradient problem that might occur when training standard RNNs. In many situations, LSTM outperforms RNNs, hidden Markov models, and other sequence learning algorithms due to its relative insensitivity to gap length.

The cell memory unit of an LSTM cell gives it an edge over a common recurrent unit. The cell vector can embody the concept of forgetting a portion of its previously stored memory as well as adding a portion of the new information. To demonstrate this, examine the cell's equations and how it handles sequences beneath the hood.

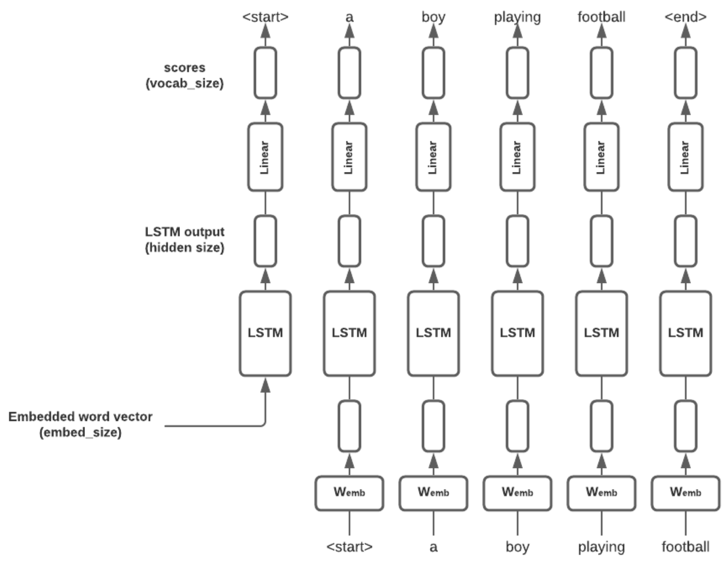


Figure 2: LSTM Architecture

3.3 BLEU Score

Here, we need a metric to decide how good our model is. For this, we have to compare the decoder's output which is the caption generated with the original caption of the image. For this, we used Bilingual Evaluation Understudy (BLEU) [7] score which is a metric for evaluating a generated sentence to a reference sentence. The BLEU score is calculated by counting matching n-grams in the output sentence to n-grams in the given reference sentence. Unigram or 1-gram is each token and a bi-gram comparison would be each word pair. Ordering of words is omitted while calculating BLEU score. The BLEU score ranges from 0 to 1. Few translations will attain a score of 1 unless they are identical to a reference translation.

4 Experiment Setup

In this project, we built and trained a CNN-RNN (Convolutional Neural Network - Recurrent Neural Network) model to generate image captions automatically for a given input image. So, the input for the model would be the images and the output the model generates will be the caption or description of the images, which is a set of words or a sentence. Sentence can be thought of as word sequences. Sequential models can assist us in processing word sequences (or characters, or other sequential data such as time-series).

We followed the following steps for this project : Data-set Collection, Text (Caption) Preprocessing, Image Preprocessing, Building a CNN Encoder, Building a RNN Decoder, Training and Hyper-parameter tuning and Testing.

4.1 Dataset Collection

We have used the Microsoft Common Objects in Context (MS COCO) dataset[2] to train the network. The model for picture captioning is shown below.

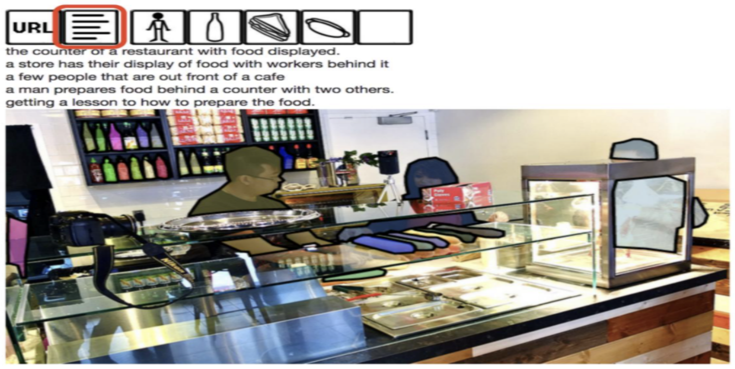

URL
the counter of a restaurant with food displayed.
a store has their display of food with workers behind it
a few people that are out front of a cafe
a man prepares food behind a counter with two others.
getting a lesson to how to prepare the food.

Figure 3: Dataset Image Sample

The dataset contains images in JPEG format with different shapes and sizes and each image has 5 different captions. These images depict a variety of scenes and situations.

The images are bifurcated as follows in the code:
• Training Set — 87000 images
• Test Set — 40000 images

As mentioned, each image has 4-5 different captions stored in a separate file in the dataset.

4.2 Caption Preprocessing and Tokenizing Captions

Captions must also be preprocessed and prepared for training. In this example, we intended to construct a model that predicts the next token of a sentence based on previous tokens, thus we tokenized the caption linked with any image before casting it to a PyTorch tensor that we can use to train the network.

First, we go through all of the training captions and build a dictionary that maps all unique terms to numerical indices. As a result, each word we see will have a matching numeric value that can be located in this dictionary. This dictionary's terms are referred to as vocabulary.

4.3 Conversion of words to vectors

The words must first be converted into a numerical form so that a network can compute the difference between a forecast word and a ground truth word (from a known, training caption) using standard loss functions and optimizers. As a result, we generally convert a string of words into a string of numerical values; a vector of numbers, each of which corresponds to a single word in our dictionary.

4.4 Preprocessing of Images

Before passing the input images to the CNN model for training, we have applied transformations which include resize, horizontal flip and normalization.

4.5 Encoder CNN

The Encoder CNN needs an image 224x224x3 size. To extract features from a batch of pre-processed images, we utilized the pre-trained ResNet-50 [8] architecture. We have removed the last fully-connected layer and the result is then flattened to a vector and sent through a Linear layer, which transforms the feature vector to be the same size as the word embedding.

4.6 Decoder RNN

RNN decodes the process vector and turn it into a sequence of words. In this case, LSTM (Long Short Term Memory), is used which is a special kind of RNN that includes a memory cell, in order to maintain the information for a longer period of time. We have considered a simple decoder architecture having one layer of LSTM with 512 units.

4.7 Training and Testing

We have two model components, encoder and decoder, and we train them together by sending the encoder's output, which is the latent space vector, to the decoder, which is the recurrent neural network. Because of the frequent disconnections of the run time environment during training, we saved the model after each epoch as a checkpoint and then retrieved the model from the saved model and used it in the epoch. To determine how well our model is performing, we can examine how the training loss and perplexity change throughout training — and for the sake of this research, we may adjust the hyper-parameters depending on this data. Because of the huge dataset and the architecture, we couldn't perform a grid search where we could decide the desired values for different hyper-paramters and hence we tuned each hyper parameter one after the other. Here the hyper-parameters that we tuned are learning-rate, number of epochs and batch size.

Firstly, we tuned the batch size and due to the limited computing resources, we couldn't decrease batch-size to below 512. So, we considered BLEU score for different batch sizes and trained the model from 1 to 5 epochs.

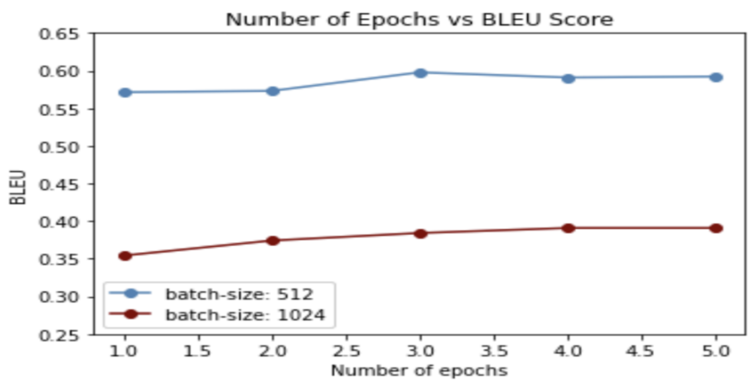


Figure 4: Epochs vs BLEU

So, from the above plot, we can see that the batch size of 512 gives a better BLEU score which is close to 1.0 Hence, we fixed the batch size to 512.

After fixing the batch size, we moved to learning rate and considered 3 different learning- rates which are 0.1, 0.01 and 0.001. The learning-rate of 0.001 gave us a less training loss and below are the following graphs where we calculated training loss after each epoch for each different learning rate.

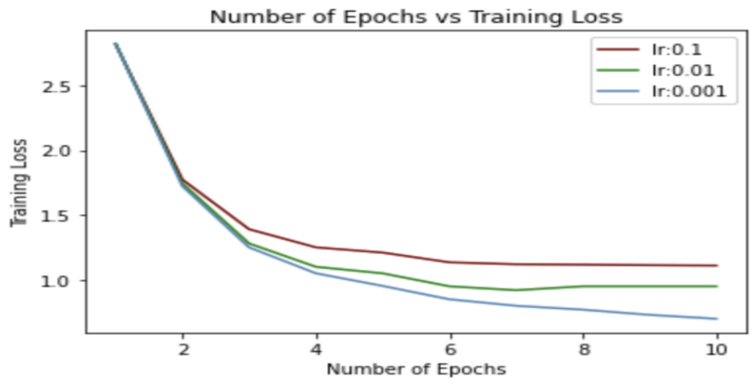


Figure 5: Epochs vs Training Loss

The next hyper parameter that we tuned is the number of epochs. For this we consider the BLEU scores and based on BLEU score, we will fix the number of epochs in training. Following plot shows the BLEU scores vs number of epochs.

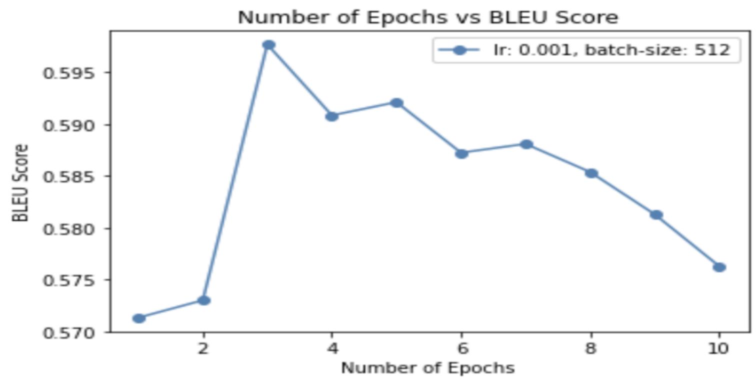


Figure 6: Epochs vs BLEU, lr:0.001, batch-size : 512

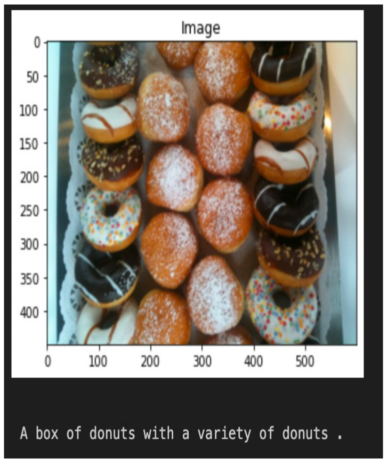
We have trained the model for 10 epochs and from the above graph, we can see that for this architecture, we have a high BLEU score for 3 epochs. After 3 epochs, the model underwent over-fitting as BLEU score reduced.

5 Results

After setting batch size to 512, learning rate to 0.001, number of epochs to 3, we have trained the model and tested the model on 40000 data records. We then calculated the BLEU score for each data record and calculated an average BLEU score for all the records in test dataset. There are many interpretations of how the BLEU score is expressed. Some consider between 0 and 1 and some scale it and express it between 1 and 100. We have considered not to scale as recommended and the final BLEU score for the model is 0.5876. The BLEU score for the Neural Image Caption (NIC) in [1] is 27.7. Below are few input images from the test dataset and the corresponding captions generated by our model.

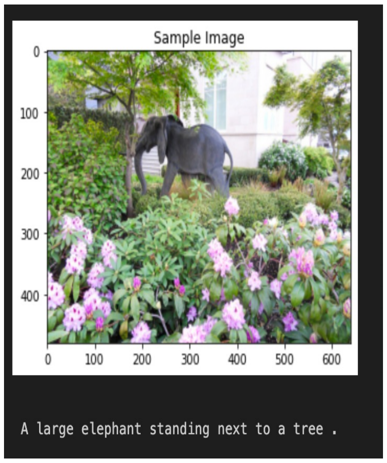
In the first three images 7(a), 7(b) and 7(c), the captions generated by the model exactly matches the input images.

In the last two images 8(a) and 8(b), we can see that the captions generated by the model almost describes the image but not completely. Also, few captions are different from the input images. As the model has a BLEU score of 0.5976, the caption that generated is very close to the original description of the image. If the model attains a BLEU score of 1 across all the records in the dataset (which is ideal and hypothetical), it will generate the exact captions.

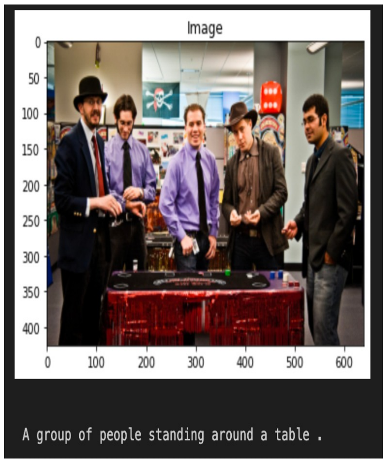


```
A box of donuts with a variety of donuts .
```

(a)



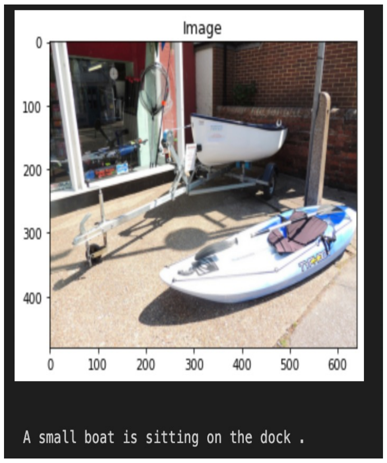
```
A large elephant standing next to a tree .
```



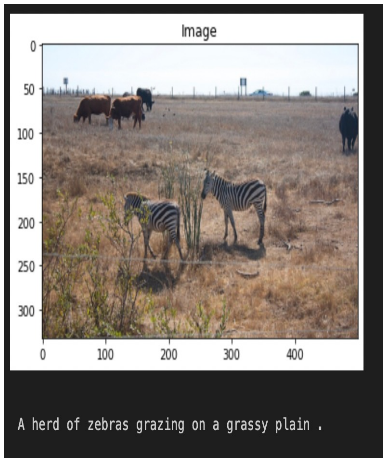
```
A group of people standing around a table .
```

(b) (c)

Figure 7: Captions generated exactly matches with the images above



```
A small boat is sitting on the dock .
```



```
A herd of zebras grazing on a grassy plain .
```

(a) (b)

Figure 8: Captions generated by the model

6 Conclusion

As we know that the training dataset which has nearly 87,000 images with captions and nearly 40,000 images in test dataset, and considering the computing resources at hand, we have considered a simple architecture for Encoder and Decoder and trained only for few number of epochs. Also, as we are dealing with multi-level classes, we have restricted to 87 classes which are across the training and testing data. The model will not be able to identify any class which is not in these 87 classes while generating the caption for the given image.

So, the further improvements of this project can be :

1. Including more vocabulary for the model to generate.

2. The ResNet-50 architecture can be replaced with Mobile Nets [9] or Efficient Nets [10] which are very light weight architecture as they have depth-wise convolutions which can be trained from scratch, instead of using pre-trained Image-Net weights. But, to get more accuracy than ResNet-50, we need to train for more number of epochs and require more computing resources.

3. We have used the classic LSTMs with a single layer of 512 units as the decoder model. With more computing resources, the decoder architecture can be made complex by adding more layers of LSTM.

4. Gated Recurrent Units [11] can also be employed for the decoding task to see if it can give better results compared to LSTMs.

References

[1] Oriol Vinyals et al. "Show and tell: A neural image caption generator". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3156–3164. DOI: 10.1109/CVPR.2015.7298935.

[2] "Common Objects in Context, COCO". In: *https://cocodataset.org/home*. URL: https://cocodataset.org/#overview.

[3] Pranay Mathur et al. "Camera2Caption: A real-time image caption generator". In: *2017 International Conference on Computational Intelligence in Data Science(ICCIDS)*. 2017, pp. 1–6. DOI: 10.1109/ICCIDS.2017.8272660.

[4] Seung-Ho Han and Ho-Jin Choi. "Explainable Image Caption Generator Using Attention and Bayesian Inference". In: *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*. 2018, pp. 478–481. DOI: 10.1109/CSCI46756.2018.00098.

[5] Ali Farhadi et al. "Every Picture Tells a Story: Generating Sentences from Images". In: *Computer Vision – ECCV 2010*. Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 15–29. ISBN: 978-3-642-15561-1.

[6] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[7] Kishore Papineni et al. "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002.

[8] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[9] Andrew G. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV].

[10] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: 1905.11946 [cs.LG].

[11] Kyunghyun Cho et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. arXiv: 1406.1078 [cs.CL].

Appendix

Image Caption Generator - https://github.ncsu.edu/schidar/image-captioning

Because of the computational limitations we had, we were only able to train the CNN-RNN model for more than 10 epochs. As the dataset has images which belong multiple classes and because the dataset is sparse for a particular set of classes, we were not able to sample the data and train for less data set. We have taken Google colab pro GPUs to train our model for 10 epochs and for hyperparameter tuning.