Image Captioning with Deep Neural Networks

Amogh Yatnatti ARY180001 | Jacob Glenny JDG180001 | Aditi Chakravarthi AXC200021 | Erum Hooda EJH170130

***Abstract* - For our final project, we implemented an image captioning system using Python and PySpark which, given an image, can generate a descriptive caption that concisely represents what is shown in the image given. There are three main components implemented in our system: a Convolutional Neural Network (CNN) which takes images as input, a Recurrent Neural Network (RNN) which takes captions as input, and a fully connected (dense) Feed-Forward Neural Network (FFNN) which combines the image and the caption to pair them together.**

In our implementation, we used the Flickr30k dataset, which contains more than 30,000 images. Each image has 5 captions associated with it.

To implement the CNN, our image encoder, which works to extract the important features from each image it is given, we used Google's EfficientNet CNN, which is pre-trained on ImageNet data. We decided to go this route to save computation time, since our dataset has many thousands of images, and training a CNN on this large of a dataset, especially one made up of images, would be time consuming.

For our linguistic encoder, we implemented an RNN, specifically a Long Short-Term Memory (LSTM) RNN. Once trained, the RNN builds the caption sentences one word at a time, and using an LSTM for this task means the system remembers the words that came before in any given sentence when trying to choose the next word.

The final step pairs an image with a caption. Our system merges the output of the CNN and RNN using addition. Then, a fully connected FFNN decodes the output of the merge step by extracting patterns encoded in previous steps.

I. INTRODUCTION AND BACKGROUND WORK

A. Description of the Image Captioning Problem

Given an image, our system generates a description which describes the contents of the image. This combines computer vision and natural language processing, as you need the application to both analyze the image and generate an English sentence describing the image.

B. Dataset Used

The dataset used is Flickr30k. This dataset contains 31,000 images from Flickr along with a CSV file containing five captions per image. The captions have been provided by human annotators. For the purposes of this project, we will be using all five captions for each image. We ran into issues with uploading dataset images to AWS so we were only able to upload around 16,000 images [1]. This amount should still be more than enough to train and test our model.

C. Background on Deep Neural Networks and Machine Learning in General

A two layer neural network can, with arbitrarily small error, approximate any finite-degree polynomial function over a closed domain of input values [2]. From there, polynomial functions can approximate any continuous function over a closed domain. This gives some intuition about how incredibly expressive neural networks can be, especially for Deep Neural Networks (DNN) which are neural networks with many layers.

DNNs are capable of learning many challenging problems such as those in computer vision and natural language processing. For example, a classic image-classification neural network, VGG16, has 16 layers. More modern networks, such as the EfficiencyNet family of models "(EfficiencyNetB0, …, EfficiencyNetB7) range from 237 to 831 layers" [3]. These layers are more sparsely connected compared to VGG16's fully connected layers.

D. Background on RNN and CNN

Convolutional Neural Networks (CNN) are a type of DNN where special operations called convolutions are used. In general, these convolutions mutate the dimensional shape of their input. The goal of convolutions in image classification, for example, is to extract information from nearby groups of

pixels. This makes intuitive sense for image data, as nearby pixels are likely to make up the same object.

Recurrent Neural Networks (RNN) are a type of DNN where layers are allowed to have connections to previous layers. This is opposite to Feed Forward Neural Networks (FFNN), where layers are only connected in the forwards direction. The recurrent connections allow for a type of "memory". In our use case, caption generation, each word in the sentence depends on the words before it. For example, the sentence "The family wearing swimsuits plays in the sand at the beach". The words "swimsuits" and "sand" help determine that the rest of the sentence should include "beach". Another use-case which is not relevant to us is video analysis, where a series of images may be necessary rather than just a single image. RNN does well for these series/sequential data.

II. THEORETICAL AND CONCEPTUAL STUDY OF THE TECHNIQUES AND ALGORITHMS IMPLEMENTED

A. Generic Image-Caption Generation Strategy

At a high level, the algorithm we implemented is a sentence-generation network.

Assume the model is already trained. To generate a caption, the steps are:

- input ['<begofseq>'] to the model and receive as output word y_1.
- input ['<begofseq>', y_1] to the model and receive as output word y_2.
- input ['<begofseq>', y_1, y_2] to the model and receive as output word y_3…
- repeat until y_m is '<endofseq>'.

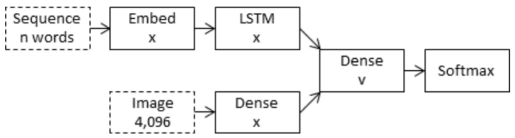
In other words, the sentence is generated one word at a time, where each word depends on the full sequence of previously generated words. The model is trained with '<endofseq>' appended to every description so that it learns when a caption should be finished.

Finally, we also need to input the image data, since we want image captions and not arbitrary English sentences. The next section discusses two methods for including the images.

B. Encoder-Decoder Architecture

The algorithm and neural network architecture we selected comes from the paper titled, "What is the Role of Recurrent Neural Networks (RNNs) in an Image Caption Generator?" [4]. This paper [4] mainly compares two options: the merge architecture and the inject architecture.

(a) The merge architecture.



(b) The inject architecture.

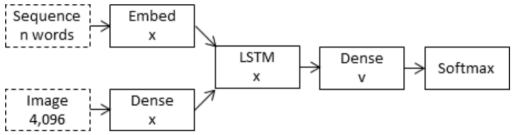


Fig. 1. Merge vs. Inject architecture [4].

In the merge architecture, the image data does not flow through the RNN (LSTM). In this case, the RNN is only "encoding linguistic features" [4]. In other words, it is simply a sentence generator. Then, the image features are included later where a dense layer "decodes" the association between image features and linguistic features to decide on the most relevant sentence.

In the inject architecture, the RNN (LSTM) does receive input from the images. In this case, the RNN is "encoding the linguistic and image features together" [4]. In other words, it is more directly acting as a caption generator.

The paper [4] provides experimental data to show that the merge architecture slightly outperforms the inject architecture in accuracy, but really shines because of its "higher performance to model size ratio" [4]. We implement the merge architecture because of this efficiency.

C. CNN as an Image Encoder or Image Feature Extractor and Which CNN We Used

Transfer learning is the method of using a pre-trained model to aid in solving a new, related problem [5]. In our case, a CNN trained on ImageNet classification data can be used to extract image features useful to our related filckr30k image captioning problem.

There is an annual CNN benchmark for ImageNet data classification tasks. Looking at the results [6], it is clear that EfficiencyNets, the "easily scalable neural networks" [7] developed by Google, are among the current state of the art. At the time of writing, they make up 13 of the top 20 image classifiers ranked by top-5 accuracy [6]. Compared to their competition, EfficiencyNets require much fewer parameters. Google's original paper about these networks [8] shows this success visually, where the top left corner is optimal, having the best accuracy and the fewest parameters.

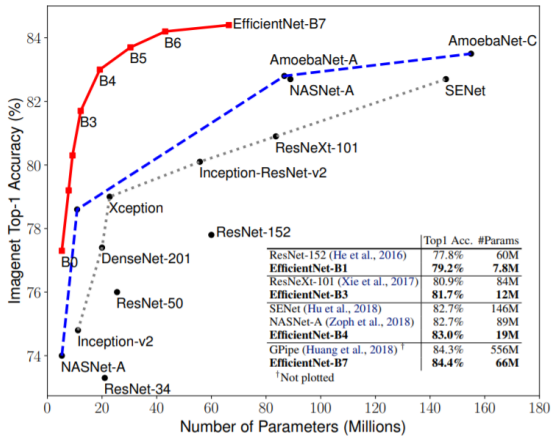


Fig. 2. Comparison of different CNNs trained on ImageNet [8].

EfficiencyNet has several pre-trained versions increasing in size: B0 to B7.

TABLE I. EFFICIENCYNET VERSIONS

EfficiencyNet	#params(10^6)	features_output_shape	bytes
B0	4.0	(7, 7, 1280)	21MB
B1	6.6	(7, 7, 1280)	32MB
B2	7.8	(7, 7, 1408)	..
B3	10.8	(7, 7, 1536)	..
B4	17.7	(7, 7, 1792)	..
B7	64.1	(7, 7, 2560)	266MB

Fig. 3. Comparison of different EfficiencyNet versions.

Others [4] have used VGG16 [9] as their image CNN, but we wanted an efficient and modern image CNN. VGG16 has 138*10^6 params, is 500MB, and was created in 2014 [10]. EfficiencyNetB0 is small and fast, and has sufficient accuracy for our needs.

To make the computation more efficient, we first ran the EfficentNetB0 model on all the images in our dataset and saved their features to a file. This way, the features are only calculated once per image. These features are then used in the merge architecture as the image input. As we used the CNN as a feature extractor, we used the option to remove the top (final) layers trained and intended for ImageNet classification [11].

D. RNN as a Linguistic Encoder and How LSTM Improves Upon Basic RNN

As described above, the desired behavior for our caption generator is to generate a sentence one word at a time, where each word depends on the full sequence of previously generated words. Although RNN is technically capable of short and long term (full sequence) memory, often the longer-term gradients “vanish”, i.e. “round down to zero because of limited precision floating point arithmetic” [12].

LSTM enforces long term memory and therefore avoids the vanishing gradient problem. Practically, this means that the end of the captions will be more related to the beginning of the captions and create a logical sentence. For example, the sentence: "UTD students walk together to the school bookstore where they plan to rent textbooks and buy t-shirts depicting the school mascot Temoc." The last word of the sentence, "Temoc", depends not only on the words immediately before it, "school mascot", but also the first word of the sentence, "UTD". The blog post titled “Understanding LSTM Networks” [13] has a detailed yet accessible explanation of the specific mechanism LSTMs use for long term memory. In short, an LSTM is stateful. It learns to manage its state by “remembering” or “forgetting” words which will or will not be useful for determining the rest of the sentence. The equations for LSTM definition are given in [4], [13], [12].

E. The Final Layers: Merge, Dense-Decoder, and Output

The merge architecture necessitates somehow combining the outputs of the image feature and linguistic feature layers. Since the feature vectors are defined as the same length, the merge step combines these two feature vectors element-wise into a vector of the same size.

For the merging operation, one paper [4] uses concatenation, while a more recent paper [14] claims “addition outperforms concatenation and multiplication” [14]. We implement addition.

There is one dense (fully connected) standard feed forward layer responsible for decoding (extracting patterns from) the merged feature-encodings.

The output of the final layer of the model is a vector with length equal to the number of distinct words in the training caption set. Using a softmax activation function, the output is a probability distribution over this vocabulary. The word with the highest probability is the output word.

III. RESULTS AND ANALYSIS

A. NLP Accuracy

To evaluate accuracy of the model, we will be using Bilingual Evaluation Understudy (BLEU) scores. It estimates the accuracy and quality of generated text relative to the known English caption [15]. This is a great way to compare our results because we are able to generate accurate captions in a myriad of ways.

BLEU scores range from 0 to 1 and we can specify the weighting of N-grams from 1 to 4. A 1-gram represents single words whereas a 2-gram represents word pairs and so on. If we were to generate an identical caption to the actual, then we would get a 1 BLEU score across all N-grams.

However, since we are not likely to get identical captions (without overfitting), we need to use a range that indicates an acceptable response.

B. Insightful Examples of Test Image and Model-Generated Caption Pairs

Fig. 4. Test Image Put Through Our System.

Predicted Caption: woman in blue shirt is sitting on bench
Original Captions: [an asian child looking towards something wearing traditional clothing, young female child carrying light blue bag around her shoulders, little oriental girl glancing upward toward her right, the little girl looked in awe at the sight, native girl in blue top looking skyward]

Fig. 5. Test Image Put Through Our System.

Predicted Caption: young boy in blue shirt is playing with bubbles
Original Captions: [little boy plays in water that has formed puddle, little kid in black playing in fountain, young child is playing in fountain,little boy is walking through water, boy is playing in water fountain]

Fig. 6. Test Image Put Through Our System.

Predicted Caption: man in blue shirt is riding bike
Original Captions: [man in blue jacket and khaki pants bicycling on sidewalk next to tree, man wearing blue jacket is riding bicycle in an empty park, man in blue jacket rides bicycle on the sidewalk, man in blue coat rides bicycle along sidewalk, man rides bike past an old baseball field]

Analyzing the predicted captions, the word "man" or "woman" appears in most captions as well as descriptions of clothing items, especially "shirt". These three examples are quite good. There are small errors such as using "woman" instead of "girl" and confusing the colors "blue" and "black".

TABLE II. EXPERIMENT RESULTS

Trial	Parameters Chosen	Results
1	512 training images. 1 caption per image. 24 max caption length. 826,064 NN params. 2 epochs	BLEU: 1: 0 2: 0 3: 0 4: 0
2	16,188 training images. 1 caption per image. 24 max caption length. 2,368,064 NN params. 2 epochs	BLEU: 1: 0 2: 0 3: 0 4: 0 Loss = 7.678
3	16,188 training images. 5 captions per image. 20 max caption length. 3,557,717 NN params. 4 epochs	BLEU: 1. 0.548 2: 0.298 3: 0.209437 4: 0.098722 Loss = 4.501

Fig. 7. Experiment Parameters and Their Results.

C. Interpreting Table Results

Loss given is from validation data. The model efficacy improved significantly once we were considering enough training data. Five captions per image performs much better, with our only non-zero BLEU scores.

According to the 2017 paper, "Where to put the Image in an Image Caption Generator," they derived that an "acceptable" range BLEU 1-gram is from 0.401 to 0.578. For BLEU 2-gram, they derived from 0.176 to 0.390. For BLEU 3-gram, 0.099 to 0.260. For BLEU 4-gram, from 0.059 to 0.170 [16]. Our final model meets all four of these "acceptable" ranges.

The number of neural network parameters is primarily determined by the vocabulary size, which scales with the number of training captions. The only parameter which decreased was the maximum number of words per caption, from 24 to 20. This was a compromise to slightly reduce training time.

D. Measuring and Avoiding Overfitting

As the model trains/learns from the training data, it improves its ability to generate captions for the training data. This does not necessarily imply an improvement on captions for the testing data. When the training performance is high but the testing performance is low, the model is overfit to the training data.

So, there is an optimal number of epochs to maximize testing performance. Typically when the validation-data performance starts decreasing, the model is becoming overfit, so stop training.

"Epoch" is the number of complete passes through the entire training data. One implementation is to do a fixed number, ex. 10, epochs and save to disk the 10 (model, validation accuracy) pairs. Then, keep the model with the highest validation accuracy and discard the others.

For our implementation, we decided to train the model for 10 epochs. Our final model stopped improving on validation data after 4 epochs. Compared to other machine learning models we are familiar with, less than 5 epochs is a very fast training time, although it makes sense with a model as expressive as a deep neural network. Additionally, by using transfer learning with the pre-trained EfficiencyNet CNN, our model had less work to do.

IV. CONCLUSION AND FUTURE WORK

While our model doesn't provide state-of-the-art captioning, it achieved reasonable success on a subset of the flickr30k dataset. A more sophisticated model would use deep-captioning to describe multiple objects in the scene rather than just the main subject. However, limiting ourselves to 20 word captions, it's difficult to describe more than one subject anyways.

Due to time and storage limitations, we were only able to use 16,188 images of the 31,000 images in the dataset. We removed all the images with greater than 20-word captions. The actual maximum in the dataset was 74 words.

With more computational resources, we would use more training examples and higher resolution images which might also lead to better predictions.

One alternative source of data is the MS COCO (Common Objects in Context) dataset. This dataset is commonly used for object recognition, image captioning, and scene recognition (with foreground and background depth distinguishment) [19]. Using this dataset might be more effective for deep-captioning.

REFERENCES

[1] Young et al., "Papers with code - flickr30k dataset," *Dataset | Papers With Code*. [Online]. Available: https://paperswithcode.com/dataset/flickr30k. [Accessed: 28-Nov-2021].

[2] A. Andoni, R. Panigrahy, G. Valiant, and L. Zhang, "Learning polynomials with Neural Networks," *PMLR*, 18-Jun-2014. [Online]. Available: http://proceedings.mlr.press/v32/andoni14.pdf. [Accessed: 28-Nov-2021].

[3] V. Agarwal, "Complete architectural details of all EfficientNet models," *Medium*, 23-May-2020. [Online]. Available: https://towardsdatascience.com/complete-architectural-details-of-all-efficientnet-models-5fd5b736142. [Accessed: 28-Nov-2021].

[4] M. Tanti, A. Gatt, and K. P. Camilleri, "What is the role of recurrent neural networks (RNNS) in an image caption generator?," *arXiv.org*, 25-Aug-2017. [Online]. Available: https://arxiv.org/abs/1708.02043. [Accessed: 28-Nov-2021].

[5] F. Chollet, "Keras Documentation: Transfer Learning & Fine-tuning," *Keras*, 12-May-2020. [Online]. Available: https://keras.io/guides/transfer_learning/. [Accessed: 28-Nov-2021].

[6] "Image Classification on ImageNet - ImageNet Benchmark," *Papers With Code*. [Online]. Available: https://paperswithcode.com/sota/image-classification-on-imagenet. [Accessed: 28-Nov-2021].

[7] M. Tan and Q. V. Le, "EfficientNet: Improving accuracy and efficiency through AutoML and model scaling," *Google AI Blog*, 29-May-2019. [Online]. Available: https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html. [Accessed: 28-Nov-2021].

[8] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for Convolutional Neural Networks," *arXiv.org*, 23-Nov-2019. [Online]. Available: https://arxiv.org/pdf/1905.11946v3.pdf. [Accessed: 28-Nov-2021].

[9] M. ul Hassan, "VGG16 - convolutional network for classification and detection," *VGG16 - Convolutional Network for Classification and Detection*, 20-Nov-2018. [Online]. Available: https://neurohive.io/en/popular-networks/vgg16/. [Accessed: 28-Nov-2021].

[10] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv.org*, 10-Apr-2015. [Online]. Available: https://arxiv.org/abs/1409.1556. [Accessed: 28-Nov-2021].

[11] Y. Fu, "Keras documentation: Image Classification via fine-tuning with EfficientNet," *Keras*, 16-Jul-2020. [Online]. Available: https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/#keras-implementation-of-efficientnet. [Accessed: 28-Nov-2021].

[12] "Long short-term memory," *Wikipedia*, 25-Nov-2021. [Online]. Available: https://en.wikipedia.org/wiki/Long_short-term_memory. [Accessed: 28-Nov-2021].

[13] C. Olah, "Understanding LSTM networks," *Understanding LSTM Networks -- colah's blog*, 27-Aug-2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed: 28-Nov-2021].

[14] K. Doshi, "Image captions with Deep Learning: State-of-the-art architectures," *Medium*, 23-Apr-2021. [Online]. Available: https://towardsdatascience.com/image-captions-with-deep-learning-state-of-the-art-architectures-3290573712db. [Accessed: 28-Nov-2021].

[15] J. Brownlee, "A gentle introduction to calculating the BLEU score for text in Python," *Machine Learning Mastery*, 18-Dec-2019. [Online]. Available: https://machinelearningmastery.com/calculate-bleu-score-for-text-python/. [Accessed: 28-Nov-2021].

[16] M. Tanti, A. Gatt, and K. P. Camilleri, "Where to put the Image in an Image Caption Generator," *arXiv.org*, 14-Mar-2018. [Online]. Available: https://arxiv.org/abs/1703.09137. [Accessed: 29-Nov-2021].

[17] F. Chollet, "Keras documentation: Making new layers and models via subclassing," *Keras*, 13-Apr-2020. [Online]. Available: https://keras.io/guides/making_new_layers_and_models_via_subclassing/. [Accessed: 28-Nov-2021].

[18] "Module: TF: Tensorflow core v2.7.0," *TensorFlow*, 27-Nov-2021. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/. [Accessed: 28-Nov-2021].

[19] T.-Y. Lin et al., "Microsoft COCO: Common Objects in Context", in Computer Vision -- ECCV 2014, 2014, bll 740–755. https://cocodataset.org/#home/. [Accessed: 28-Nov-2021].