One Network for All

Image captioning, similar words, relevant images for text query
and similar images for image query

Adithya Chowdary Boppana
Indiana University
aboppana@iu.edu

Sahaj Singh Maini
Indiana University
sahmaini@iu.edu

Siva Charan Mangavalli
Indiana University
simang@iu.edu

Abstract

Each layer in a neural network holds important information, depending on how deep the model is a layer can give high-level or low-level features. We tried to exploit this fact by extracting the information from layers of a neural network built for image captioning task and use it to solve different problems such as extracting similar images for a given image, extracting similar words for given word during inference.

The main task is image captioning which involves generating a textual description of an image. It is a difficult task given we must incorporate visual understanding in a linguistical sense. Further, we have explored the use of the same model to solve the complete opposite problem of retrieving an image for a given input caption.

We were able to perform all the mentioned tasks by using the embeddings generated from the intermediate layers. Also, we explored an interesting method of approximating the embedding given the caption and used it to find relevant images for a given caption. We have examined the results of all individual tasks in the results section.

1. Introduction

The task of generating an apt description for a given image is, indeed, a difficult task as the generated description should agree with linguistic semantics and recognize all the objects in the image and their dependency to get the contextual information. There have been many attempts to combine two different models, one to extract image features and the other to use this information and built a caption using linguistic rules.

After the advent of deep learning, researchers have tried replacing these two models with a single deep neural network that does both the tasks by learning information from input data. By doing so, we are avoiding an extra step of providing linguistic information to the system.

One such implementation was done by Vinyals [1], where he uses a deep network consisting of CNN, RNN and FC layers. The model is trained to maximize the probability of the output caption for a given image. He was able to get better results than Karpathy [4] as highlighted in the paper. We have implemented the same network with minor modifications. We have used GRU cells instead of LSTM cells. Once the network for image captioning was built, as we have converted every image and word to a fixed length vector representation using the network embeddings. Using the embedding vectors, we computed cosine similarity for retrieving similar images and words for a given input image or input word. Then, we have fixed the weights of the model and updated a randomly initialized tensor for a user entered caption to generate a tensor that is similar to the embeddings of the images that are relevant to the caption.

To capture information from the image we have used pre-trained InceptionV3 network. This image information was then fed to an RNN layer followed by an fully connected neural network, along with the word vectors to get the image caption.

2. Previous Work

Initially, people used logic systems (rule-based) [10] to convert image information to natural language. But as these systems are rule-based these are heavily biased to specific domains. Later for a given image, captions were assigned based on ranking similarity [11]. But the problem here is we will not be able to generate new captions for unseen images.

Then Vinyals [1] uses a deep convolution net for image combined with RNN for sequence modeling and thereby creates a single network for caption generation. It was mentioned in the same paper how we can extract similar words using the network embeddings.

Karpathy [4] has worked on generating textual description of images and subregions using a deep CNN and a bidirectional RNN was the state of the art for 2015.

Kelvin [5] have worked on attention-based model using CNN and RNN layers to get state of the art performance on three benchmark datasets. The model they trained automatically learns to adjust its gaze to generate captions.

3.1. **Data**

We have used Flickr 8K data-set for the Project. The data set consists of 6000 training images along with five human-created captions for each image. It has 1000

validation and 1000 testing images. We took lemmatized captions to reduce the #words in vocabulary.

3.2. Implementation

Pre-Processing of data: Every image and its caption data was Pre-Processed as required by the network

Image Pre-Processing: We have removed the top FC layer of inception-V3 [12] which was built for classification of ImageNet images and used for extracting image features. Every input image was converted to 299,299,3 shape as required for the inception network. These image pixels were then normalized between -1 to 1. The input image was passed through inception-V3 frozen network to get the 2048 vector.

Caption Pre-Processing: Two seed words ”<start>”, “<end>” have been added to all the captions at the start and end of it. All the captions are then tokenized and padded to an equal length where length equals to that of longest caption present. For padding, we add a token '<pad>' with a token value of 0 to the set of tokens extracted from the training data. We later use this token to pad all the sentences to equal length. All the sentences in the training data are then represented using a list of token values.

For instance, let’s consider the longest caption in a dataset made of eight different words with each word having the following token values- '<start>' = 1, '<end>' = 11, 'A' = 2, 'dog' = 6, 'plays' = 10 and '<pad>' = 0. So, a sentence like "<start> A dog plays <end>" will be represented by a list of token values of the words in the sentence as [1,2,6,10,11,0,0,0,0,0]

3.3. Architecture

After receiving the image vector in 2048 dimensions from InceptionV3 for all images, we pass this vector into a fully connected neural network that outputs a three hundred dimensional embeddings for every image. The model also contains another fully connected neural network to output a 300-dimensional embedding for each word when the token index for a word is passed through this network. The 300-dimensional vectors of images and the 300-dimensional vectors of words are passed to a layer of recurrent neural network which is made up of a GRU cell. The output from this recurrent neural network is now passed into a fully connected layer that outputs a vector of dimension equal to the size of the vocabulary (which is used to find the next word with highest probability) in the training dataset. This is used to predict the next word for the given sequence.

Overall, the model consists of two fully connected neural network layers, to create three hundred dimensional embeddings for images and words, a recurrent neural network and a fully connected neural network to predict the next word.

Network Architecture

We have used Adam Optimization for training with a learning rate of 0.001 and sparse categorical cross-entropy as the loss function. In the above image, the white components refer to the components whose weights will be updated during training.

3.4. Training

1. We get the 2048 image vector using Inception layer
2. The 2048-dimensional vector is passed to the FC layer and we get the 300-dimensional embedding of the image.
3. Then we calculate the 300-dimensional embedding of the ‘<start>’ word by giving it as input to the fully connected neural network meant for generating word embeddings.
4. We concatenate the image embedding with the embedding of the start word and predict the next word. We then calculate the loss between the predicted word and the actual word in the sentence.
5. Now we concatenate image embedding with the embedding of start word and the embedding of the next word in the sentence given as a label, and predict the next word using that.
6. We keep incrementing the words in the sentence and keep predicting the next word and keep calculating the loss for each predicted word till the end of the sentence.
7. After the loss has been calculated for all predicted words, we average the loss for all the predicted words for a sentence and update the network parameters accordingly.
8. We repeat the above steps for all the examples in the training data.

3.5. Inference

3.5.1. Caption generation:

1. We similarly generate the image vector and ‘<start>’ word vector as in training and feed them to an RNN layer followed by a fully connected neural network layer to predict the next word.
2. After predicting the next word, we concatenate the 300-dimensional image embedding, the 300-dimensional

embedding of the start word and the 300-dimensional embedding of the predicted word to predict the next word.
3. We keep increasing the sequence length of the list of 300-dimensional embeddings by including the previous prediction to predict the next word. We do this until we predict the '<end>' token and return the predicted caption.

3.5.2 Inferring Images:

In order to infer the image from the given caption, we follow a procedure which is very similar to training where we pass embeddings of image and words of the caption in variable length sequences as input and original caption as the label. Here, instead of passing a 300-dimensional image embedding, we pass a randomly initialized 300-dimensional tensor sampled from a uniform distribution along with word embeddings into the recurrent neural network layer to predict the next word. Now, we calculate the loss in the same way as the training but, in the process of reducing the loss, we update the values on the initially randomly generated tensor rather than updating the weights in the model. For each sentence we repeat this a number of times and use this approximated 300-dimensional tensor. to find similar images from the latent space. We assume that the initially randomly generated vector has been updated by the model to match the image embedding that might have produced the caption given as input. So, to get the image we use a similar image search, that gives images of top image embeddings that are most similar to the updated 300-dimensional input vector(which was randomly initialized).

3.5.3. Inferring similar images and words:

For a given input given we have first converted it to the 300-dimensional vector from the pre-final layer of inception model. We calculated the similarity via a dot product of these 300-dimensional vectors and returned the top 10 images by descending order.

In an identical way, we also have a 300-dimensional vector for every word in the training data, so for a new word (That should be within our vocabulary to have its 300-dimensional vector), we have calculated the similarity via dot product to returned the top 10 similar words

4. Results

4.1. Image Captioning:

Captions generated for the two input images are shown below.

A surfer be ski down a wave

A dog run through the snow

The image captioning evaluation metric score**s** BLEU - 1, BLEU – 2 have been calculated on the model. We are currently achieving a BLEU – 1 score of 0.45 and a BLEU – 2 score of 0.25.

4.2. Inferring Image using a given caption:

Below attached are two cases of image retrieval for a user enter caption.

Query: dog play in a beach

Query: people play football

4.3. Image Query:

We compared the image embeddings of an input image for a given query and below are the retrieved images.

Query

Retrieved Images

Query

Retrieved Images

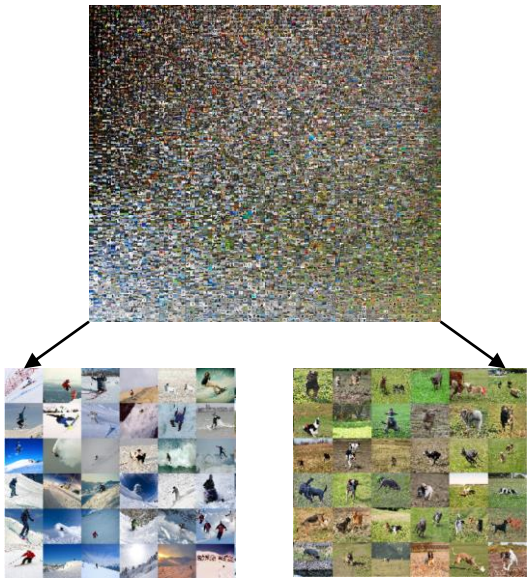
4.4. Similar words:

By comparing the word vectors generated with the input word's vector we obtained the following similar words.

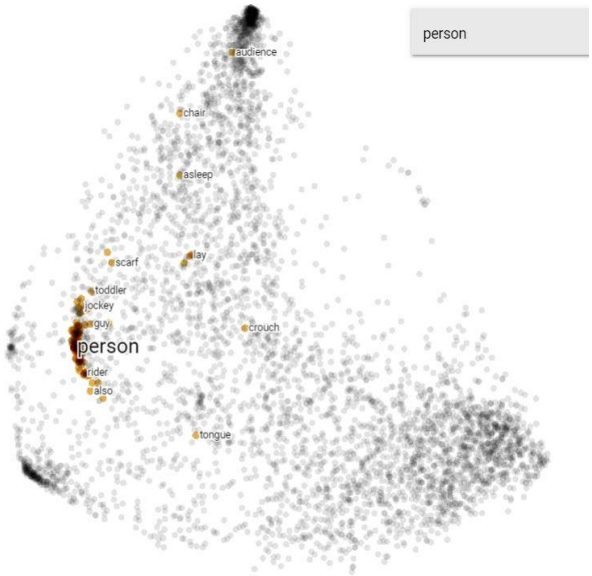
Query	Similar words
Black	White, Gray, Orange, Yellow, Brown
Soccer	Tennis, Baseball, Basketball, Football, Rugby

4.5. Experiments:

We have t-SNE on all training images embeddings and reduced the dimensionality from 300 to 2. Later we have distributed the 2-dimensional data in a square grid using Jonker-Volgenant algorithm[7]. Similar images appear nearby as shown in the left and the right most corners.



We have visualized the t-SNE distribution of the word embeddings using a perplexity of 15 and 2 dimensions. For a selected word "person" the closest words by cosine distance appear nearby mostly. This has been visualized using Embedding projector in Tensor board by using the tensors and metadata.



For the relevant image retrieval using a caption, we have tried updating the embedding with various iteration values and the embeddings tend to give good results after 1000 epochs as shown above.

4.6. Discussion

Vinyals [1] achieved BLEU – 1, BLEU – 2 scores around .60 and .40 respectively, it was trained and tested on similar data set (Flickr 30K), but not on Flickr 8K. Our Bleu scores were less owing to lower training time and lack of finetuning.

Obtaining similar images for a given image and similar words for a given similar work the same way. We compare the query vector with the embeddings and return the ones with the highest similarity and the results are contextually related to the query.

Finding the relevant images given a query text is the most challenging task. We changed the number of iterations for the embedding approximation process gradually. We observed that the returned images were more contextually similar if the #iterations are more than 1000. The drawback for this would be runtime, which increased linearly with #iterations.

5. Conclusion and Future work

Through the experiments and results, we have shown that a single neural network can be used for multiple applications using the embeddings from the internal layers. We extended the idea of using the embedding to generate similar words [1] by using the image embeddings to find similar images and using the same model to approximate the image embedding for a given caption. This approximated image embedding for a given caption has been used to find similar images from latent space. In the future, Sent2Vec vectors and Locality Sensitive Hashing can be used to obtain the relevant results in a more efficient and quicker way.

6. Acknowledgment

We would like to take this opportunity to thank all the course staff who supported and helped us in completing this project. Firstly, we thank Prof. Minje Kim and IU for making this course available for us to register. We got a chance to apply the learnings of class i.e. Inception model, RNN, comparing embeddings vectors and network optimization in our project. We are very grateful for giving us $50 Google cloud credit which has been very instrumental. We would like to thank TA's and fellow students for helping us with the questions on Piazza and office hours.

7. Contributions

We started with data collection and literature review, in which all of us are actively involved. Siva Charan has done the initial data exploration. All of us are involved in the image captioning modeling process. Sahaj worked on building the code for getting relevant images for a given caption. Adithya has worked on obtaining similar words and similar images for a given word or image query. We further optimized the model and tried few experiments, the optimization part was mainly done by Sahaj and Adithya, while experiments of getting the t-SNE plots of images and word embeddings were carried out by Siva Charan.

References

[1] Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan "Show and Tell: A Neural Image Caption Generator"

[2] Tanya Piplani "DeepSeek: Content Based Image Search & Retrieval"

[3] Shuang Bai "A Survey on Automatic Image Caption Generation"

[4] Andrej Karpathy, Li Fei-Fei "Deep Visual-Semantic Alignments for Generating Image Descriptions"

[5] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan "Show, Attend and Tell: Neural Image Caption. Generation with Visual Attention"

[6] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, Sanja Fidler "Skip-Thought Vectors"

[7] Flickr 8k https://forms.illinois.edu/sec/1713398

[8] Google Conceptual Captions https://ai.google.com/research/ConceptualCaptions

[9] https://blog.sourced.tech/post/lapjv/

[10] B. Z. Yao, X. Yang, L. Lin, M. W. Lee, and S.-C. Zhu. I2t:Image parsing to text description. Proceedings of the IEEE, 98(8), 2010

[11] M. Hodosh, P. Young, and J. Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. JAIR, 47, 2013.

[12] https://ai.googleblog.com/2016/03/train-your-own-image-classifier-with.html