Image Captioning using Deep Learning

Malick Fairoz – sayeedabuthahir.m@husky.neu.edu
Mitesh Puthran – puthran.m@husky.neu.edu
INFO 7390, Fall 2018, Northeastern University

Abstract -- Automatically generating a natural language description of an image is a task close to the heart of image understanding. In this paper, we present a multi-model neural network method closely related to the human visual system that automatically learns to describe the content of images. Our model consists of two sub-models: an object detection and localization model, which extract the information of objects and their spatial relationship in images respectively; Besides, a deep recurrent neural network (RNN) based on long short-term memory (LSTM) units with attention mechanism for sentences generation. Each word of the description will be automatically aligned to different objects of the input image when it is generated. This is similar to the attention mechanism of the human visual system. Experimental results on the Flickr dataset show case the merit of the proposed method, which outperform previous benchmark models.

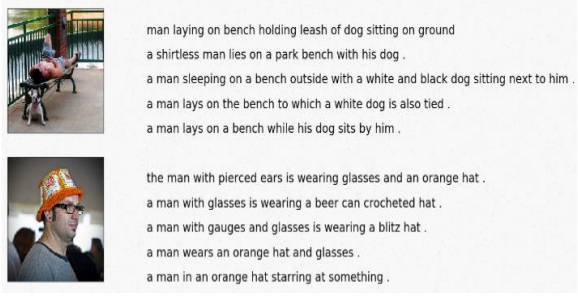
I. Introduction

In the past few years, deep neural network has made significant progress in image processing area, like image classification, object detection. However, tasks like image classification and object detection are far from the end of image understanding. One of the goal of image processing is deep image understanding by the machines i.e. understanding the whole image scenario not individual objects. Image captioning follows the same path: by extracting the complete detail of individual object and their associated relationship from image. Finally, the system can automatically generate captions to describe the image. This problem is extremely important, as well as difficult because it connects two major artificial intelligence fields: computer vision and natural language processing.

II. Dataset

The paper implements three different models of TensorFlow Object Detection API and compares them in terms of speed and accuracy. The dataset used is a Flickr 8K Dataset from the link https://forms.illinois.edu/sec/1713398 that contains 8091 images in a folder and other test files which contain 5 captions for each image with their image name. We also need VGG-16 pre-trained weights for feature extraction which can be downloaded from https://github.com/fchollet/deep-learning-models/releases/tag/v0.1

Samples images and description from dataset.



III. Project framework

Previous studies are mostly inspired by the works of neural machine translation, where the task is to translate a source sentence written in one language (like French) into a target sentence written in a different language (like English), keeping logic and syntax precise. Associating image captioning and machine translation together makes sense because they can be placed in the same framework, called Encoder-Decoder framework. In the Encoding step, we encode the source information, which is an image in image captioning task and source

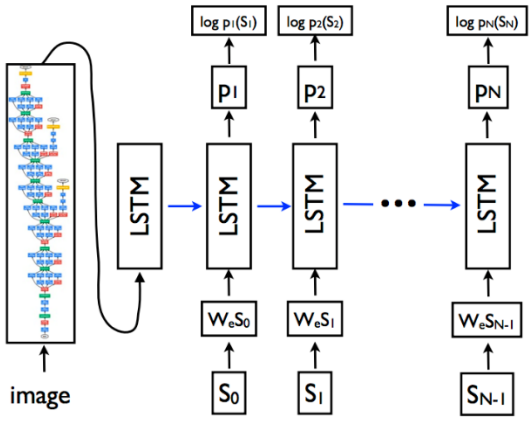
sentence in translation task, into a target vector. Followed by the decoding step, in which sentences are generated by decoding the target vector. The core part of this framework is how to encode the source information (image or sentence) and how to decode the target vectors.

Previous work align in the decoding step, they usually use long short-term memory (LSTM) based on recurrent neural networks (RNN) units as a decoder. As for the encoding step is concern, the work is divided into two major classes: CNN-LSTM models and attention-based models. CNN-LSTM models represent an image as a single feature vector from the top layer of a pre-trained convolutional network whereas, attention-based models use the vector made up by the representations of image's subregions as the source vector.

Our model also follows the Encoder-Decoder framework; however, it is totally different from the previous models. Our motivation in describing an image is to find out its contents, rather than focusing on some meaningless regions associated with it. As the locations of different objects in an image is also a positive information for describing an image, since they reflect the spatial position relation-ships of objects in an image. Due to the aforementioned reason, we combine object detection with image captioning to focus on the real meaningful information domain in the images to generate the resulting sentences much better and efficient. The encoding part of our model consists of two steps. Initially, we use VGG16 model to extract the image spatial relationships features. All these information along with the caption data will be represented as a set of feature vectors, which is then fed into the LSTM network where the model will learn to generate captions.

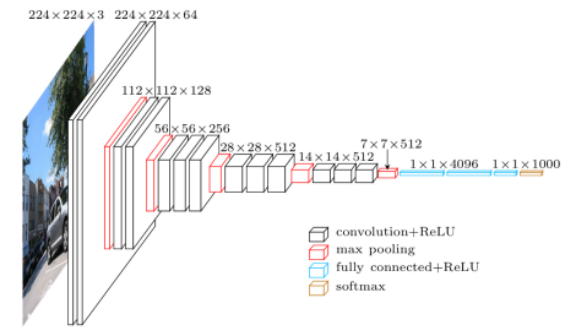
In order to measure the performance, we evaluate our model on Flickr dataset using standard metric: BLEU results shows that our proposed model perform better than the baseline soft attention model and is similar to the benchmark

The LSTM Architecture is shown below



IV. Object Detection

In the past few years, significant progress has been done in object detection. These advances are driven by the success of region proposal methods and region-based convolutional neural networks (R-CNN). In our model, we choose VGG16 as object detection model due to its efficiency and effectiveness in object detection task. VGG16 is a convolutional neural network and composed of 16 layers. This network takes input of size (224,224,3). The output layer contains 1,000 nodes. VGG16 is developed to classify images into 1,000 different classes. As I am not using VGG16 for the sake of the classification, but I just need it for extracting features, I will remove the last layer from the network. For each image, 4096 features are created.



V. Text Preparation

The description text will need to be encoded to numbers before it can be presented to the model as in input or compared to the model's predictions. The first step in encoding the data is to create a consistent mapping from words to unique integer values. Keras provides the Tokenizer class that can learn this mapping from the loaded description data. The to_lines() to convert the dictionary of descriptions into a list of strings and the create_tokenizer() function that will fit a Tokenizer given the loaded photo description text. Each words will be split into tokens. The model will be provided with one word and the image to generate the next word. Then the first two words of the description will be provided to the model as input with the image to generate the next word. This is how the model will be trained.

For example, the input sequence "little girl running in field" would be split into 6 input-output pairs to train the model:

X1,	X2 (text sequence),	y (word)
photo	startseq,	little
photo	startseq, little,	girl
photo	startseq, little, girl,	running
photo	startseq, little, girl, running,	in
photo	startseq, little, girl, running, in,	field
photo	startseq, little, girl, running, in, field,	endseq

Later, when the model is used to generate descriptions, the generated words will be concatenated and recursively provided as input to generate a caption for an image.

The function below named create_sequences(), given the tokenizer, a maximum sequence length, and the dictionary of all descriptions and photos, will transform the data into input-output pairs of data for training the model. There are two input arrays to the model: one for photo features and one for the encoded text. There is one output for the model which is the encoded next word in the text sequence.

The input text is encoded as integers, which will be fed to a word embedding layer. The photo features will be fed directly to another part of the model. The model will output a prediction, which will be a probability distribution over all words in the vocabulary.

The output data will therefore be a one-hot encoded version of each word, representing an idealized probability distribution with 0 values at all word positions except the actual word position, which has a value of 1.

We now have enough to load the data for the training and development datasets and transform the loaded data into input-output pairs for fitting a deep learning model.

VI. Model

This section describes the detail of the model, which consists of two main parts: encoding and decoding. The input to our model is a single image I, while the output is a descriptive sentence S consists of K encoded words: $S = \{w_1, w_{2,\dots,} w_k\}$. In the encoding part, firstly, we present a model that recognizes objects in the input image followed by a deep CNN to extract their locations, which reflect the spatial relationship associated. All the information will be represented as a set of feature vectors referred as annotation vectors. The encoding part produces L annotation vectors, each of which is a D-dimensional representation corresponding to an object and also its spatial location in the input image: $A = \{A_1, A_{2,\dots,} A_L\}$.

In the decoding part, all these annotation vectors are fed into a deep LSTM Network model to generate a description sentence.

We will describe the model in three parts:

- **Photo Feature Extractor**. This is a 16-layer VGG model pre-trained on the ImageNet dataset. We have pre-processed the photos with the VGG model (without the output layer) and will use the extracted features predicted by this model as input.
- **Sequence Processor**. This is a word embedding layer for handling the text input, followed by a Long Short-Term Memory (LSTM) recurrent neural network layer.
- **Decoder** (for lack of a better name). Both the feature extractor and sequence processor output a fixed-length vector. These are merged together and processed by a Dense layer to make a final prediction.

The Photo Feature Extractor model expects input photo features to be a vector of 4,096 elements. These are processed by a Dense layer to produce a 256 element representation of the photo.

The Sequence Processor model expects input sequences with a pre-defined length (30 words) 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 5% dropout. This is to reduce overfitting the training dataset, as this model configuration learns very fast.

The Decoder model merges the vectors from both input models using an addition operation. This is then fed to a Dense 256 neuron layer and then to a final output Dense layer that makes a softmax prediction over the entire output vocabulary for the next word in the sequence.

VII. Tensorflow

TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

VIII. BLEU

The Bilingual Evaluation Understudy Score, or BLEU for short, is a metric for evaluating a generated sentence to a reference sentence. A perfect match results in a score of 1.0, whereas a perfect mismatch results in a score of 0.0. The score was developed for evaluating the predictions made by automatic machine translation systems. BLEU score calculation is implemented in nltk.util It is not perfect, but does offer 5 compelling benefits:

- It is quick and inexpensive to calculate.
- It is easy to understand.
- It is language independent.
- It correlates highly with human evaluation.
- It has been widely adopted.

This approach works by counting matching n-grams in the candidate translation to n-grams in the reference text, where 1-gram or unigram would be each token and a bigram comparison would be each word pair. The comparison is made regardless of word order.

IX. Results

These generated captions are compared to the actual captions from the dataset and evaluated using BLEU scores as the evaluation metrics. A

score closer to 1 indicates that the predicted and actual captions are very similar. As the scores are calculated for the whole test data, we get a mean value as 0.398 which includes good and not so good captions.

Some of the examples for good caption can be seen below.

true: white and black dog and brown dog in sandy terrain

pred: black and white dog is running along the beach

BLEU: 0.730633242659

true: man and baby are in yellow kayak on water

pred: man in blue shirt is sitting in the water

BLEU: 0.759835685652

true: black dog leaps over log

pred: black and white dog is running over the grass

BLEU: 0.759835685652

Some of the examples for bad caption can be seen below.

true: collage of one person climbing cliff

pred: couple at the background

BLEU: 0

true: black and white dog jumping in the air to get toy

pred: dog is running in the grass

BLEU: 0.24439253249722204

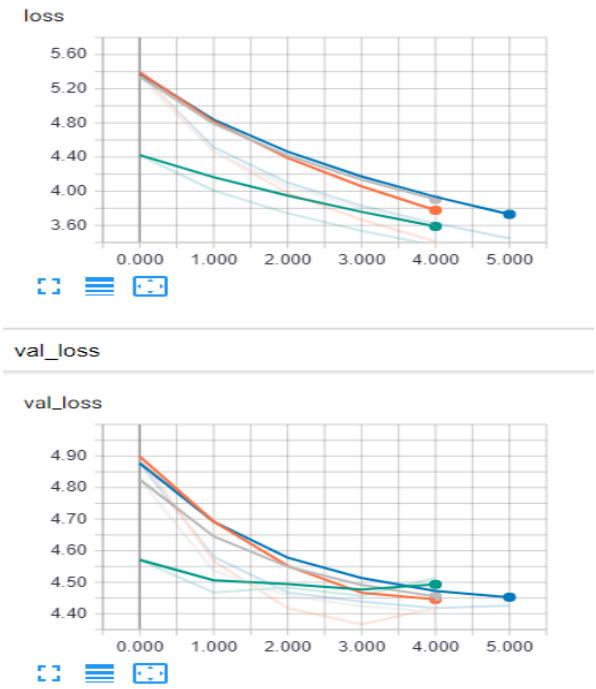
X. Conclusion

The trained model generates captions for the input images. The caption generation capability has been improved by tuning the hyper parameters of the model. The below table shows the various combination of parameter values and the BLEU score changes accordingly.

	A	B	C	D	E	F	G	Output
2	DEEP LEARNING MODEL	ACTIVATION FUNCTION	COST FUNCTION	EPOCHS	GRADIENT ESTIMATION	NETWORK ARCHITECTURE	NETWORK INITIALIZATIO	Mean BLEU score
3	**Gradient Estimation**							
4	1	ReLU	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.37
5	2	ReLU	Cross-Entropy	6	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.351
6	3	ReLU	Cross-Entropy	5	Adagrad	3 layer, 256 nodes, LSTM, vgg16	default	0.404
7	4	ReLU	Cross-Entropy	5	RMSProp	3 layer, 256 nodes, LSTM, vgg16	default	0.374
8	5	ReLU	Cross-Entropy	5	Adadelta	3 layer, 256 nodes, LSTM, vgg16	default	0.353
9	6	ReLU	Cross-Entropy	5	Nadam	3 layer, 256 nodes, LSTM, vgg16	default	0.353
10	7	ReLU	Cross-Entropy	5	SGD	3 layer, 256 nodes, LSTM, vgg16	default	0.028
11	**Cost Function**							
12	1	ReLU	mean_squared_error	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.215
13	2	ReLU	hinge	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0
14	3	ReLU	kullback_leibler_divergence	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.373
15	4	ReLU	cosine_proximity	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0
16	**Network Initialization**							
17	1	ReLU	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	glorot_uniform	0.381
18	2	ReLU	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	random_uniform	0.388
19	3	ReLU	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	lecun_uniform	0.367
20	4	ReLU	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	he_uniform	0.389
21	5	ReLU	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	glorot_normal	0.398
22	**Activation Function**							
23	1	ReLU	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.374
24	2	tanh	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.384
25	3	elu	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.392
26	4	selu	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.363
27	5	linear	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.192
28	6	sigmoid	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.375
29	7	softsign	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.396
30	8	softplus	Cross-Entropy	5	Adam	3 layer, 256 nodes, LSTM, vgg16	default	0.381
31	**Epochs**							
32	1	ReLU	Cross-Entropy	3	Adam	3 layers, 256 nodes each	default	0.429
33	2	ReLU	Cross-Entropy	4	Adam	3 layers, 256 nodes each	default	0.394
34	3	ReLU	Cross-Entropy	5	Adam	3 layers, 256 nodes each	default	0.408
35	4	ReLU	Cross-Entropy	6	Adam	3 layers, 256 nodes each	default	0.38
36	5	ReLU	Cross-Entropy	7	Adam	3 layers, 256 nodes each	default	0.405
37	**Network Architecture**							
38	1	ReLU	Cross-Entropy	5	Adam	3 layers, 256 nodes each	default	0.407
39	2	ReLU	Cross-Entropy	5	Adam	3 layers, 128 nodes each	default	0.405
40	3	ReLU	Cross-Entropy	5	Adam	3 layers, 512 nodes each	default	0.394
41	4	ReLU	Cross-Entropy	5	Adam	4 layers, 256 nodes each	default	0.406
42	5	ReLU	Cross-Entropy	5	Adam	4 layers, 128 nodes each	default	0.386

It should be noted that the higher BLEU score doesn't always reflect better caption generation. We also have to look at the validation losses and make sure that the model doesn't overfit. Sometimes higher BLEU score models generate bas captions which can be seen from the images above.

We also use Tensorboard which helped us visualize how the losses changed while training different models with different parameters.



X. Acknowledgement

We are thankful to our Prof. Nik Bear Brown, Assistant Professor, Northeastern University, Boston, MA for his valuable guidance, encouragement and co-operation during the implementation of this project.

X. References

Image captioning using deep learning material
Machine Learning Mastery
LSTM Encoder Decoder
VGG16
Framing image description
Flickr Terms
FiryOnIce
FiryOnIce Repo
Flickr form