Object and Action Detection using Android

Ashwani Kumar (17JE003253)

Shubham Mishra (17JE003230)

Under the supervision of

Dr Haider Banka

IIT ISM Dhanbad

Image

Acknowledgement

Our sincere thanks go to our mentor, Dr. Haider Banka, who has guided us throughout the stages of this project. His expertise in Machine Learning and Computer Vision, along with her work experience in these domains has significantly improved the quality of our final report.

We would also like to thank our college for giving us this chance to complete our project and learn so many new things.

Abstract

Computer vision has become ubiquitous in our society, with applications in several fields. In this project, we focus on one of the visual recognition facets of computer vision, i.e image captioning. The problem of generating language descriptions for visual data has been studied for a long time but in the field of videos. In the recent few years emphasis has been lead on still image descriptions with natural text. Due to the recent advancements in the field of object detection, the task of scene description in an image has become easier.

The project aimed to train convolutional neural networks with several hundreds of hyperparameters and apply it on a huge dataset of images (MS-COCO), and combine the results of this image classifier with a recurrent neural network to generate a caption for the classified image. This generated caption will be used by our Android device to speak out the results. In this report, we present the detailed architecture of the model used by us.

Table of Contents

1. Introduction

Artificial Intelligence (AI) is now at the heart of the innovation economy and thus the base for this project is also the same. In the recent past, a field of AI namely Deep Learning has turned a lot of heads due to its impressive results in terms of accuracy when compared to the already existing Machine learning algorithms. The task of being able to generate a meaningful sentence from an image is a difficult task but can have a great impact, for instance helping the visually impaired to have a better understanding of images.

The task of image captioning is significantly harder than that of image classification, which has been the main focus in the computer vision community. A description of an image must capture the relationship between the objects in the image. In addition to the visual understanding of the image, the above semantic knowledge has to be expressed in a natural language like English, which means that a language model is needed. The attempts made in the past have all been to combine the two models.

In the model proposed, we try to combine this into a single model which consists of a Convolutional Neural Network (CNN) encoder which helps in creating image encodings. We use the VGG16 architecture proposed with some modifications. We could have used some of the recent and advanced classification architectures but that would have increased the training time significantly. These encoded images are then passed to an LSTM network which is a type of Recurrent Neural Network. The network architecture used for the LSTM network work similarly to the ones used in machine translators. The input to the network is an image that is first converted in a 224*224 dimension. We use the MS COCO dataset to train the model. The model outputs a generated caption based on the dictionary it forms from the tokens of the caption in the training set. The generated caption is compared with the human given caption via the BLEU score measure.

In the following sections, we discuss briefly the model used. We discuss the CNN encoder and the LSTM decoder in detail. The code module of both the architectures is also explained briefly. We used the BLEU score metric to compare the accuracy of the model proposed with the ones already present. In the end, we report a few examples tested on the model.

2. Related Work

Image caption generation is a core part of scene understanding, which is important because of its use in a variety of applications (eg. - image search, telling stories from albums, helping visually impaired people understand the web, etc.). Over the years, many different image captioning approaches have been developed.

The architectures used by the winners of ILSVRC have contributed a lot to this field. One such architecture used by us was the VGG16 proposed by He et. al. in 2014. Apart from that the research in the tasks of machine translation has consistently helped in improving the state of the art performance in language generation.

In 2015, researchers at Microsoft's AI Lab used a pipeline approach to image captioning. They used CNN to generate high-level features for each potential object in the image. Then they used Multiple Instance Learning (MIL) to figure out which region best matches each word. The approach yielded a 21.9% BLEU score on MS COCO. After the pipeline approach, researchers at Google came up with the first end-to-end trainable model. They were inspired by the RNN model used in machine translation.

One of the most recent work was inspired by the NIC model and was proposed by Xu et. al. in 2016. They were inspired by the advancements in the field of machine translation and object detection and introduced an attention-based model that automatically learned to describe the content of images.

In the past few years, progress has been made not only in image captioning models but also in various evaluation metrics. The accuracy metric used by us was the BLEU score. BLEU - which was a standard evaluation metric adopted by many of the groups - is slowly being replaced by CIDEr proposed by Vedantam et. al. in 2015.

3. Image Captioning Model

3.1 Model Overview

The model proposed takes an image *I* as input and is trained to maximize the probability of sequence of words generated from the model and each word S_t is generated from a dictionary built from the training dataset. The input image *I* is fed into a deep vision Convolutional Neural Network (CNN) which helps in detecting the objects present in the image. The image encodings are passed on to the Language Generating Recurrent Neural Network (RNN) which helps in generating a meaningful sentence for the image as shown in the fig. However, in our model the encoder RNN which helps in transforming an input sentence to a fixed length vector is replaced by a CNN encoder. Recent research has shown that the CNN can easily transform an input image to a vector.

For the task of image classification, we use a pretrained model VGG16. The details of the models are discussed in the following section. A Long Short-Term Memory (LSTM) network follows the pretrained VGG16 . The LSTM network is used for language generation. LSTM differs from traditional Neural Networks as a current token is dependent on the previous tokens for a sentence to be meaningful and LSTM networks take this factor into account.

In the following sections we discuss the components of the model i.e. the CNN encoder and the Language generating RNN in details.

3.2 Dataset

For the task of image captioning we use “MS-COCO” dataset. The dataset contains 82,000 images with 5 captions per image. The dataset by default is split into image and text folders. Each image has a unique id and the caption for each of these images is stored corresponding to the respective id.

The dataset contains 6000 training images, 1000 development images and 1000 test images. A sample from the data is given in fig.

- A biker in red rides in the countryside.
- A biker on a dirt path.
- A person rides a bike off the top of a hill and is airborne.
- A person riding a bmx bike on a dirt course.
- The person on the bicycle is wearing red.

Fig: Sample image and corresponding captions from the Flickr8k dataset

Captions generated using these datasets may prove to be better than the ones generated after training on Flickr8k because the dictionary of words used by RNN decoder would be larger in case of Flickr30k and MSCOCO.

3.3 Deep CNN Architecture

Convolutional Neural Network (CNN) have improved the task of image classification significantly. Imagenet Large Scale Visual Recognition competition have provided various opensource deep learning frameworks like ZFnet, Alexnet, Vgg16, Resnet etc have shown great potential in the field of image classification. For the task of image encoding in our model we use Vgg16 which is a 16- layered network proposed in ILSVRC 2014. VGG16 significantly decreased the top-5 error rate in the year 2014 to 7.3%.

The image taken for classification needs to be a 224*224 image. The only preprocessing done is by subtracting the mean RGB values from each pixel determined from the training images.

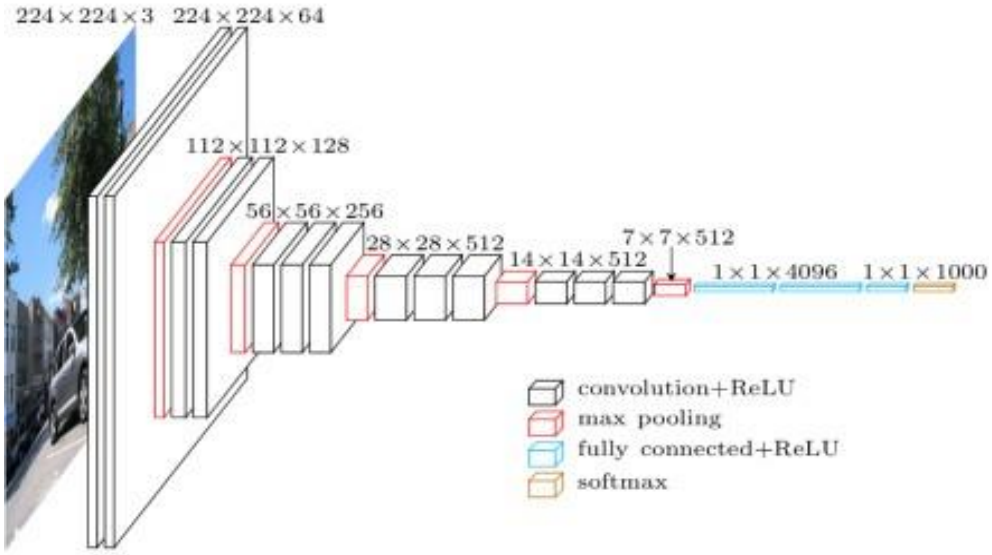


Fig: VGG16 architecture

The convolution layer consists of 3*3 filters and the stride length is fixed at 1. Max pooling is done using 2*2-pixel window with a stride length of 2. All the images need to be converted into 224*224-dimensional image. A Rectified Linear Unit (ReLU) activation function is follows every convolution layer. A ReLU computes the function $(x) = \max(0, x)$. The output of the ReLU function is given below:

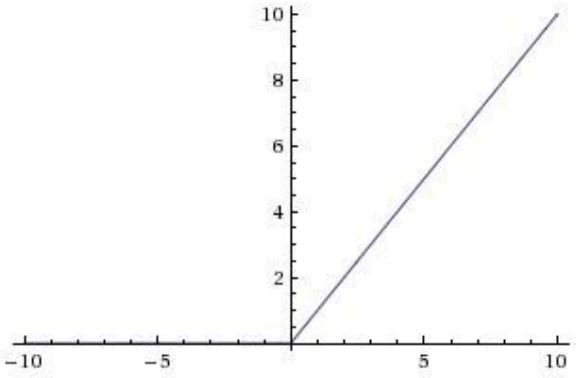


Fig: Rectified linear unit activation function

The advantage of using a ReLU layer over sigmoid and tanh is that it accelerates the stochastic gradient descent. Also unlike the extensive operations (exponential etc.) the ReLU operation can be easily implemented by thresholding a matrix of activations at zero. For our purpose however, we need not classify the image and hence we remove the last 1*1*1000 classification layer.

The output of our CNN encoder would thus be a 1*1*4096 encoded which is then passed to the language generating RNN. There have been more successful CNN frameworks like Resnet but they are computationally very expensive since the number of layers in Resnet was 152 as compared to vgg16 which is only a 16-layered network.

3.4 Recurrent Neural Net (RNN) Decoder Architecture

The LSTM networks have revolutionized the fields of speech recognition, machine translation etc. Like the conventional RNNs, LSTMs also have a chain like structure, but the repeating modules have a different structure in case of a LSTM network. A simple LSTM network is shown .

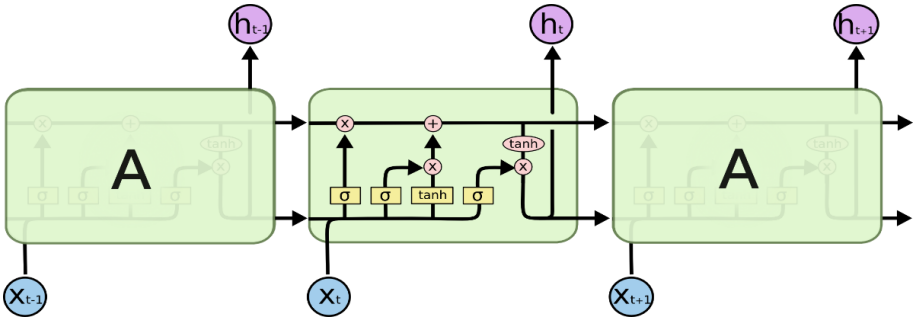


Fig. : Four interacting layers in a LSTM layer

The key behind the LSTM network is the horizontal line running on the top which is known as the cell state. The cell state runs through all the repeating modules and is modified at every module with the help of gates. This causes the information in a LSTM network to persist.

We use this LSTM network with a slight variation. The architecture of the LSTM network used is given below.

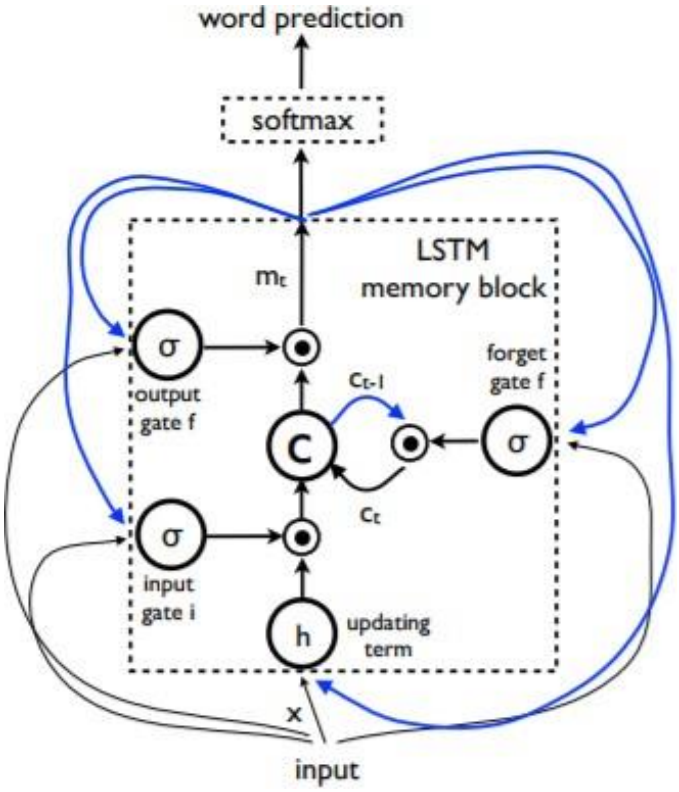


Fig: LSTM architecture for language generation

$$p_{t+1} = Softmax(m_t)$$

The output p_{t+1} of a module gives the word prediction. The same LSTM network is repeated until an end token (.) is encountered by the network. The series of these word prediction generate the caption for a given image. The complete training process for the combined model (CNN encoder + RNN language generator) and the LSTM network in unravelled form is given below.

The LSTM model is trained to predict each word of the sentence after it has seen the image as well as all preceding words as defined by *p(St|I, S0, . . . , St−1)*.

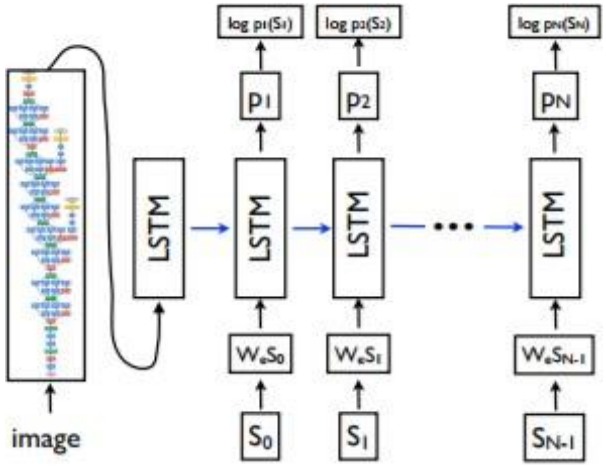


Fig: Working of the image captioning model

3.5 Code

The code for the model was built on keras. Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. We use tensorflow as the backend to build the code.

Some important functions in the CNN code are listed below

3.5.1 Conv2D

This operation creates a layer which is a convolution kernel that is convolved with the layer input to produce a tensor of outputs. The important arguments to this function are-

Filters: The first parameter to the function is filters i.e. an Integer that determines the dimensionality of the output space (the number output of filters in the convolution).

Strides: The second argument to the function is an integer or tuple/list of 2 integers, specifying the strides of the convolution along the width and height. Can be a single integer to specify the same value for all spatial dimensions.

Activation: The third argument to the function is which Activation function to use which in this case is defined as ReLU. If you don't specify anything, linear activation is applied.

3.5.2 MaxPooling2D

This operation performs maximum spatial pooling. The important arguments are-

Pool_size: First argument to the function is an integer or tuple of 2 integers, factors by which to downscale (vertical, horizontal). If only one integer is specified, the same window length will be used for both dimensions.

Strides: Second argument to the function is an Integer, tuple of 2 integers, or None. The argument governs the same operation as discussed in Conv2D layer. If None, it will default to pool_size.

3.5.3 Flatten

The flatten() function in keras is used to flatten the input. It does not effect the batch size. For instance, the output of a particular layer is of dimension (32, 32, 64) then after applying the flatten function the dimension of the feature vector would become 65536 i.e. 32*32*64. The operation of the flatten function is similar to that of numpy.reshape() function.

3.5.4 Dense

The dense layer is fully connected layer, so all the neurons in a layer are connected to those in a next layer. One important parameter to the Dense function is –

Activation: It specifies the activation function to be used. The softmax activation function in the last line maps the encoded vector in the previous step to all possible output classes.

Some important functions in the LSTM code are listed below

1. **Sequential**

   The Sequential model is generally a linear stack of layers one after the other.

2. **Embedding**

   This layer turns positive integers (indexes) into dense vectors of fixed size. This layer can only be used as the first layer in a model. The first parameter to the function is input dimensions which is equal to the size of the vocabulary and the parameter defined as 256 refers to the output dimensions of the layer.

3. **LSTM**

   This function introduces the Long-Short Term Memory layer. The first parameter specified as 1000 in this case is the output dimensions of the layer. One can also define the activation functions to be used in the layer.

4. **Merge**

   The merge functionality is used to merge two models together. For instance, in the above-mentioned case the mode ‘concat’ specifies that lang_model is concatenated after the image_model.

3.6 Results

We define the accuracy of the model by BLEU score. Bilingual evaluation understudy (BLEU) is an algorithm that evaluated the quality of text which has been translated by a machine. It was one of the first metrics to achieve high correlation with human judgement.

Blue score is always defined between 0 and 1, 0 being the machine translation is not at all related to the reference sentence. BLEU's evaluation system requires two inputs:

(i) a numerical translation closeness metric, which is then assigned and measured against
(ii) a corpus of human reference translations.

For example,

Candidate	the	the	the	the	the	the	the
Reference 1	the	cat	is	on	the	mat	
Reference 2	there	is	a	cat	on	the	mat

The candidate in this example has all its words contained in the reference thus giving an unigram precision score of 1. A unigram precision score of 1 means the candidate and reference sentences are highly correlated. However, as we can see that the two are very different from each other.

The modification that BLEU makes is straightforward. For each word in the candidate translation, the algorithm takes its maximum total count, m_{max}, in any of the reference translations. In the example above, the word "the" appears twice in reference 1, and once in reference 2. Thus $m_{max} = 2$.

For the candidate translation, the count m_w of each word is clipped to a maximum of m_{max} for that word. In this case, "the" has $m_w = 7$ and $m_{max} = 2$, thus m_w is clipped to 2. These clipped counts m_w is then summed over all distinct words in the candidate. This sum is then divided by the total number of words in the candidate translation. In the above example, the modified unigram precision score would be:

$$P = \frac{2}{7}$$

For calculating BLUE score we first generate captions for all the test images and then use theses machine generated captions as candidate sentences. We compare this candidate sentences with 5 of the captions given by humans and average the BLEU score of candidate corresponding to each of the references. Thus for 1000 test images we calculate 1000 BLEU scores using Natural Language Toolkit (NLTK), a python package.

3.7 Examples

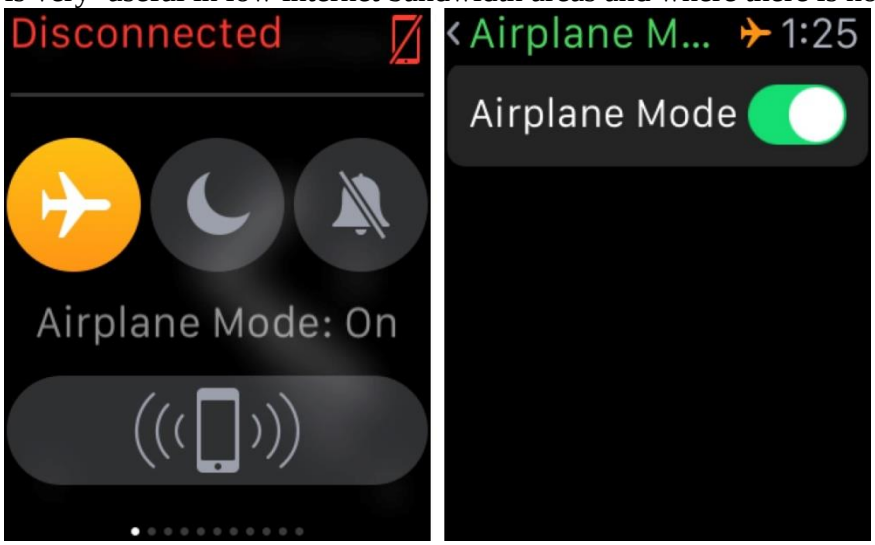
The model was able to predict the meaningful captions for an image in most of the cases. However, in some of the cases it got confused due to lack of the tokens in dictionary to define a event. Also, it got confused in the cases where almost the entire image was covered with one colour.

Following two cases show when the model could cover the details given in the image.

	**Generated Caption** A group of people are walking on a busy street. **Human provided caption** A crowd of people walk down a busy sidewalk
	**Generated Caption** A child in a helmet riding a bike **Human provided caption** A young boy rides a purple bike

4. Working on Android

The blind man can take an image of the surrounding through his camera and using our saved trained models we identify a suitable caption for it. The android mobile then speaks out aloud the caption generated and blind man will be able to hear it and visualize the environment.
An important feature of our app than it can work in airplane/offline mode which
is very useful in low internet bandwidth areas and where there is no signal.



Internal Working of Android App

The main activity of the android app has 2 buttons. The button on the right is used to click a image in front of you using a phone camera.
On clicking on the right button the camera intent button and you can capture the image. Additional functionality such as zoom is also available which capturing image.
On clicking on the left button you get an option to select an existing
Image.
After you select/click the image, the automatically genrates a caption for you an displays in the textview situated in the top of the screen.
The tts feature of Android device is then used to speak out loud the description of the image.

5. Conclusion

Our end-to-end system neural network system is capable of viewing an image and generating a reasonable description in English depending on the words in its dictionary generated on the basis of tokens in the captions of train images. The model has a convolutional neural network encoder and a LSTM decoder that helps in generation of sentences. This sentence generated is also send to Android without using any internet connection and the samrtphone speaks out the sentence to help the visually impaired get a understanding of the environment.

6. Future Prospects

The task of image captioning can be put to great use for the visually impaired. The model proposed integrated with an android or ios application works as a real-time scene descriptor.

This can be a really efficient solution for to help the visually impared.

7. References

[1] Karpathy, Andrej, and Li Fei-Fei. "Deep visual-semantic alignments for generating image descriptions." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

[2] Johnson, Justin, Andrej Karpathy, and Li Fei-Fei. "Densecap: Fully convolutional localization networks for dense captioning." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.