IMAGE CAPTIONING USING CNN AND RNN

Akkem Hema Bhargavi, Ganesh G, Jagan Mohan Reddy Dwarampudi, Sowmiya TN
Under the guidance of
Ms. Akshita Mehta, Mr. Naveen Nanda

***Abstract*—The purpose of this model is to generate captions for an image. Image captioning aims at generating captions of an image automatically using deep learning techniques. Initially, the objects in the image are detected using a Convolutional Neural Network (InceptionV3). Using the objects detected, a syntactically and semantically correct caption for the image is generated using Recurrent Neural Networks (LSTM) with attention mechanism. In our project, we are using a traffic sign dataset that is captioned by the above mentioned process. This model is of great benefit to the visually impaired in order to cross roads safely.**

***Index Terms*—CNN, Deep learning, RNN**

I. INTRODUCTION

Image captioning or generating the description of an image in natural language has received a lot of attention in recent times. It emerged as an important and challenging area with research advancing in image recognition. It is interesting due to the fact that it has a lot of practical applications like labeling large image datasets, assisting the visually impaired. It requires the level of understanding way beyond the general object detection and image classification and so, is regarded as a grand challenge. The field is a crossover of the modern Artificial Intelligence models of **Natural Language Processing and Computer Vision**.

Top-down and Bottom-up are the general image captioning approaches. While the top-down starts from the image which is later converted into words, the bottom-up approach starts with words which describe the various aspects that are combined. Both the approaches suffer from describing finer details and formulation of sentences from individual aspects respectively.

Visual attention is a selective mapping process in the human visual system. It is important for the semantic natural language description of images. So, people often talk about the crucial parts rather than everything. We will talk about the visual attention approach for image captioning. We have used an attention-based model to achieve the same. Overfitting the training data poses a massive difficulty due to the fact that the biggest available dataset has around only two million labeled individual data. This data has to be used for

a. Extraction of the features
b. Correlation with the labels
c. Syntactic understanding of the labels
d. Modeling of the language for syntactically sounding captions

The overfitting problem lies in the fact that the model memorises the inputs and uses captions that sound similar for the images which have a bit variation of specific details like a person on a ramp with a skateboard and a person on a table with a skateboard.

II. METHODOLOGY

We have trained a visual attention based model for the captioning of images with traffic signs. This model can also be extended for a general captioning which might require a larger dataset of images. Our model first pre-processes the images through InceptionV3 pre-trained model. Then, the last output layer is further taken through some convolution layers which is then processed to output the features with dimensions as required.

Parallelly, the captions of every image are tokenized and <start>, <end> tokens are appended for all the sentences. Finally, the model is trained with each image and it's corresponding real captions. The final weights are used to predict the captions for the test images. These captions are not generated at a time but rather word by word. The prediction of every word requires the probability of occurrence of that particular word, given the previous words.

This method turned out to be quite effective and this is proposed by us from our research.

Convolutional Neural Networks:

1. **Mask RCNN**

Mask Region-based Convolutional Neural Networks has been the new technology in terms of instance segmentation. There are many papers, tutorials with good quality open source codes for reference. Mask RCNN is a deep neural network aimed to unravel instance segmentation problem in machine learning or computer vision. In other words, it can be used to separate different objects in an image or a video.

You provide it with an image, it gives you the output bounding boxes for the objects, their classes and masks.

There are two stages of Mask RCNN. First, it generates proposals about the regions where there could be an object

supported the input image. Second, it predicts the category of the output, refines the bounding box and generates a mask in pixel level of the object supporting the primary stage proposal. Both stages are connected to the backbone structure which is ResNet.

Now let's check out the primary stage, a light-weight neural network called Region Proposal Network(RPN) scans all Feature Pyramid Network(FPN) top-bottom pathways (hereinafter mentioned feature map) and proposes regions which can contain objects. While scanning feature map in an efficient way, we'd like a way to bind features to its raw image location. Here come the anchors. Anchors are a group of boxes with predefined locations and scales relative to pictures. Ground-truth classes (only object or background binary classified at this stage) and bounding boxes are assigned to individual anchors consistent with some Intersection over Union value.

As anchors with different scales bind to different levels of feature map, RPN uses these anchors to work out where of the feature map 'should' get an object and what size of its bounding box is.
Here we may agree that convolving, downsampling and upsampling would keep features staying an equivalent relative locations as the objects in the original image, and wouldn't mess them around. At the second stage, another neural network takes proposed regions by the primary stage and assigns them to many specific areas of a feature map level, scans these areas, and generates objects classes, bounding boxes and masks. The procedure looks almost like RPN. Differences are that without the assistance of anchors, stage-two used a trick called Region Of Interest Align to locate the relevant areas of feature map, and there's a branch which generates masks for every object in the pixel level.

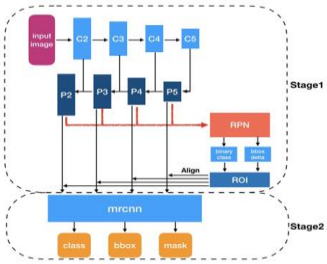


Illustration of Mask RCNN

2. YOLO

"You Only Look Once" abbreviated or most ordinarily referred to as YOLO is the best available object detection algorithm that works in real-time. Traditional object classification can be done with a simple neural network but object detection in a given scenario is a whole lot different. It requires a different approach as the conventional classification algorithms use predefined image dimensions whereas real-time object detection requires the system or model to scan the whole image for the recognition of the object. YOLO solves the problem also providing the bounding boxes for the corresponding trained objects.

YOLO works great for real-time detection as its name suggests: You Only Look Once. It has 24 convolutional layers and 2 dense layers totalling to 26 layers. It works on the Darknet architecture, created by the first author of the YOLO paper.

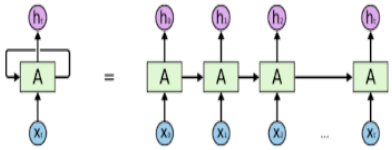
The algorithm divides any given image into an N x N grid which are then sent to the neural network to detect the presence of any objects in any of the grids and comes up with bounding boxes for the corresponding objects with probabilities and labels. The later modified models of YOLO are YOLOv2 and YOLOv3 which had 30 and 106 layers respectively. They were modified for more accurate predictions that could work with even smaller and finer details in the image.

We have tested a **YOLOv3** pretrained model with the **COCO** dataset which was able to detect 80 different labelled classes. Later, the same was implemented with our custom dataset of 790 traffic sign images for training. These images were taken from Google Open Image Dataset v5. The data was pre annotated which made it easy for us to train the model. Our model was able to detect traffic signs in an image and in real-time.

The hyperparameters for this model are tuned to have the optimizer as Adam, batch size as 8 and the model was trained for 50 epochs with 88 steps per epoch.

Recurrent Neural Networks:

A **Recurrent neural network (RNN)** has the ability to process a sequence of arbitrary length and hence used for generation of a sequence of text.



An unrolled recurrent neural network.

In this model, we worked with a Shakespeare dataset. Given a sequence of characters from this data ("Shakespear"), a model is trained in order to predict the next character in the sequence ("e"). Longer sequences of text can be generated by calling the model repeatedly. The model demonstrates how to generate text using character-based RNN.

It is trained on small batches of text, and is able to generate a longer sequence of text with coherent structure. Tensorflow library is used along with other libraries such as numpy, os.
The dataset is then downloaded and read.
Later, the data is vectorized i.e. before training, the strings are mapped to a numerical representation. The input to the model

is a sequence of characters, and we train the model to predict the output. We used tf.data to split the text into manageable sequences and the data is shuffled and packed into batches.

GRU is used in order to build the model. The loss function and optimizer used are sparse categorical cross entropy and adam respectively. The model is then trained using a certain number of epochs. After training,the model is used to generate the text from the Shakespeare data.

Hyperparameters are variables which determine the network structure and how the network is trained (predefined). The hyperparameters in this model are sequence length, batch size, buffer size, embedding dimension, rnn units, optimizer, epochs, number of char to generated.

Hyperparameter tuning is the process of tuning the above parameters in order to optimize the model. When the sequence length, batch size, buffer size, embedding dimension, rnn units, number of char is increased to a greater extent, resource exhausted error was displayed.
And if the optimizer was changed to RMSprop then the accuracy of the model gets reduced, so we use the adam optimizer as it gives the best accuracy for this model. More the number of epochs, more is the accuracy.

Image Captioning:

Caption generation is an artificial intelligence problem where a textual description must be generated for a photograph.
Try captioning the picture below.

Caption generation is an artificial intelligence problem where a textual description must be generated for a photograph.
Try captioning the below picture.

The caption a human may give will be “A dog on grass and some pink flowers”. There might be other descriptions too but all of them mean the same thing.It’s so easy for us, as human beings, to just have a glance at a picture and describe it in an appropriate language. But how does a computer program generate suitable caption for the image?

Deep learning comes into picture at this point, two models: CNN and RNN are used in order to caption the images. CNN is used to detect the objects in the image and RNN is used to generate the captions for the image. The working model of both these models are shown previously. In our model, we used MS-COCO dataset to train our model. The dataset has about 82,000 images, every image having at least 5 different caption annotations.

Transfer learning is a machine learning method where a model is developed for a task is reused as the starting point for a model on a second task. Here, we use InceptionV3 (which is pretrained on Imagenet) to preprocess and classify the images. After preprocessing, the output is cached to the disk and the captions are tokenized with the vocabulary size being limited to 5000 words to save memory while the remaining words are replaced with UNK token. The features are extracted from the lower convolutional layer of InceptionV3 giving us a vector of shape (8, 8, 2048).

This vector is then passed through the CNN Encoder (consists of a single Fully connected layer). Then RNN (here GRU) attends over the image to predict the next word. The next step is to train the model by extracting the features stored in the respective .npy files and then pass those features through the encoder. The encoder output, hidden state (initialized to 0) and the decoder input (which is the start token) is passed to the decoder.The decoder returns the predictions and the decoder hidden state.
The decoder hidden state is then passed back into the model and the predictions are used to calculate the loss.Use teacher forcing to decide the next input to the decoder.

Teacher forcing is a technique where the target word is passed as the next input to the decoder. The final step is to calculate the gradients and apply it to the optimizer and back propagate the model .

III. Experimental Results

Some of our results after implementing Mask RCNN are as follows:

Figure 1: The figure shows how Mask RCNN works on a given image for object detection

The result of using YOLOv3 algorithm on the traffic signs dataset is as follows:

Figure 2: The stop sign is detected as a traffic sign using YoloV3.

The result for RNN model of text generation is:

```
ROMEO: have but time
More plague entirest you, uncle Romeo's hand,--love Succlas't,
Some prick awhile.

DUKE VINCENTIO:
Think, it is assion, Gent
To grant her fault wase is the barky.
Out morest haste the first way's likely flattering solumine?
I like a romandrank in whinst thy flish, 'til the time to urge!
O, twice be seen,--
```

Figure 3

The result of image captioning are:

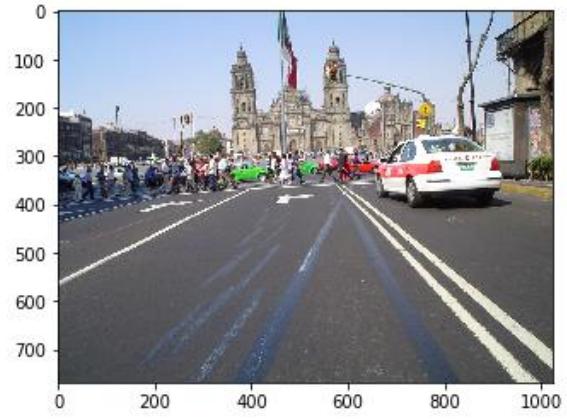


Figure 4

Figure 4 and 5 shows how the traffic signs are captioned.

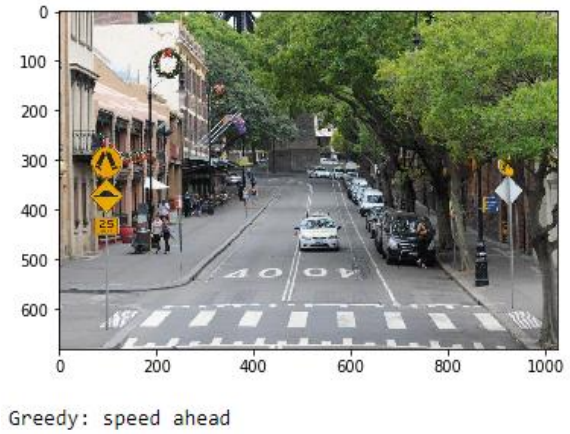


Figure 5

IV. CONCLUSION

In our project, we have developed a model to caption the images. We also extended our model by converting our captions to speech.
We hope to do future works on voice synthesis.
We have done research in order to understand our models in depth and have executed each model separately. We learned how the deep learning techniques work and how to create these models.
We faced many challenges while running the model and with our datasets. But later on we learned how to rectify the mistakes and made an efficient model.

V. ACKNOWLEDGMENT

We would like to thank our mentors Mr.Naveen Nanda and Ms.Aksitha Mehta for helpful communications and discussions regarding the work.

VI. REFERENCES

[1] Andrej Karpathy, Li Fei-Fei, "DeepVisual-Semantic Alignments for Generating Image Description", Department of Computer Science, Stanford University, published in IEEE Xplore 2015.

[2] Oriol Vinyals, Alexander Toshev, Samy Bengio,Dumitru Erhan ,"ShowandTell: ANeuralImageCaptionGenerator", a rXiv 2015.

[3] KelvinXu, JimmyLeiBa, RyanKiros, KyunghyunCho ,AaronCourville,RuslanSalakhutdinov, RichardS.Zemel Z,YoshuaBengio, "Show,AttendandTell:NeuralImageCaption GenerationwithVisualAttention",arXiv 2015.

[4] Rahul Singha, Aayush Sharmaa, "Image captioning using Deep Neural Networks", Iowa State University,published in Research gate 2018.