International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 9 Issue 8 ǁ 2021 ǁ PP. 64-71

Image Caption Generator Using RESNET-LSTM

Shobiya L
Branch- Information Technology
Sri Venkateswara College of Engineering

Pradheesha R
Branch- Information Technology
Sri Venkateswara College of Engineering

Prof. Kala
Branch- Information Technology
Sri Venkateswara College of Engineering

Abstract
Automatically describing the content of an image is a fundamental problem in artificial intelligence that connects computer vision and natural language processing. Image captioning models typically follow an encoder-decoder architecture which uses abstract image feature vectors as input to the encoder. In this project, a generative model based on a deep recurrent architecture that combines recent advances in computer vision and machine translation is present and that can be used to generate natural sentences describing an image. The model is trained to maximize the likelihood of the target description sentence given the training image. A deep neural networks based image caption generation method is analyzed systematically, with an image as the input, the method can output an English sentence describing the content in the image. Three components of the method are analyzed as Residual Networks (Res-Net), recurrent neural network (RNN) and sentence generation using Long Short Term Memory (LSTM). Experiments on several datasets show the accuracy of the model and the fluency of the language it learns solely from image descriptions. The performance of the proposed model is evaluated using standard evaluation matrices, which outperform previous benchmark models.
Keywords: *Deep Recurrent Neural Network, Long Short Term Memory, Neural Network*

Date of Submission: 16-07-2021 Date of acceptance: 01-08-2021

I. INTRODUCTION

Much research has been devoted on the automatic image captioning recently. The research can be briefly categorized into three different categories.

1.1 Existing System

Much research effort has been devoted to automatic image captioning, and it can be categorized into template- based image captioning, retrieval-based image captioning, and novel image caption generation. These methods are able to generate syntactically correct captions but are unable to generate image-specific and semantically correct captions. Template-based image captioning first detects the objects/attributes/actions and then fills the blanks slots in a fixed template. A system is described that can compute a score linking an image to a sentence. This score can be used to attach a descriptive sentence to a given image, or to obtain images that illustrate a given sentence. The score is obtained by comparing an estimate of meaning obtained from the image to one obtained from the sentence. Each estimate of meaning comes from a discriminative procedure that is learned using data. Evaluation is based on a novel dataset consisting of human-annotated images. While the underlying estimate of meaning is impoverished, it is sufficient to produce very good quantitative results, evaluated with a novel score that can account for synecdoche. The space of the meanings by triplets of object, action, and scene is represented. Node potentials are computed by linear combination of scores from several detectors and classifiers. Edge potentials are estimated by frequencies. The state space for each of the nodes have been reasonably sized. Learning involves setting the weights on the node and edge potentials and inference is finding the best triplets given the potentials. “O” stands for the node for the object, “A” for the action, and “S” for scene as shown in the figure 1.4. Learning involves setting the weights on the node and edge potentials and inference is finding the best triplets given the potentials. Retrieval-based approaches first find the visually similar images with their captions from the training dataset, and then the image caption is selected from similar

images with captions. The problem of associating images with descriptive sentences by embedding them in a common latent space have been studied by different papers. It is prohibitively expensive to fully annotate this many training images with ground-truth sentences. After investigating several state-of-the- art scalable embedding methods, a new algorithm is introduced and is called Stacked Auxiliary Embedding that can successfully transfer knowledge from millions of weakly annotated images to improve the accuracy of retrievalbased image description. Differently, the novel image caption generation approaches are to analyze the visual content of the image and then to generate image captions from the visual content using a language model. However, it still remains challenging to identify the proper CNN and RNN models for the image captioning.

1.2 RESNET-LSTM ARCHITECTURE FOR IMAGE CAPTION GENERATOR:

In this project, the image caption generator using ResNet and LSTM (Long Short Term Memory) is implemented. The image features will be extracted from a ResNet model and trained on the image-net dataset and then fed the features into the LSTM model which will be responsible for generating the image captions. For the image caption generator, Flickr_8K dataset will be used. There are also other big datasets like Flickr_30K and MSCOCO dataset but it can take weeks just to train the network so we will be using a small Flickr8k dataset. Figure 1.3 shows how captions are generated using Resnet and LSTM.

1.3 IMAGE CAPTION GENERATOR ARCHITECTURE

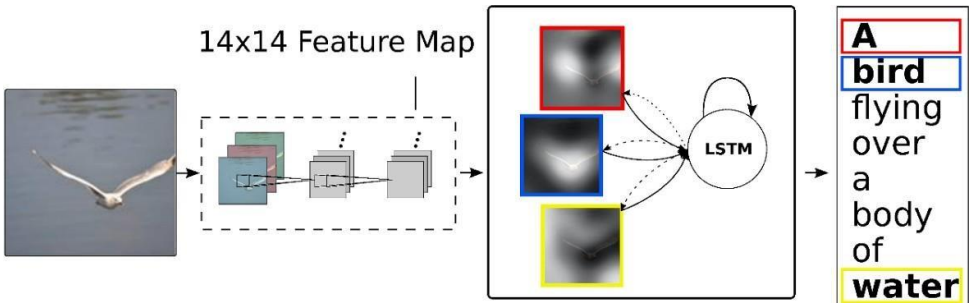


Figure 1: Resnet – LSTM Architecture

The proposed architecture of image caption generator in shown in figure 1.

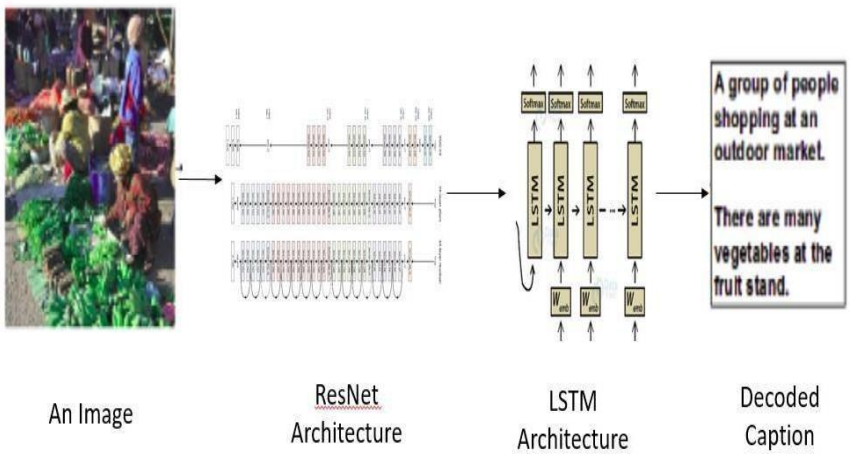


Figure 2: Image Caption Generation Architecture

Image caption generator is a task that involves computer vision and natural language processing concepts to recognize the context of an image. The generation of captions from images has various practical benefits, ranging from aiding the visually impaired, to enabling the automatic and cost-saving labelling of the millions of images uploaded to the Internet every day. The objective of our project is to learn the concepts of Res-Net and LSTM model and build a working model of Image caption generator by implementing Res-Net with LSTM. The AI- infused image caption generator is packed with deep learning neural networks
Namely, Recurrent Neural Networks (RNN), -Long Short Term Memory (LSTM) and ResNet, wherein-

- **ResNets** are deployed for extracting spatial information from the images
- **RNNs** are harnessed for generating sequential data of words
- **LSTM** is good at remembering lengthy sequences of words

1.4 RESIDUAL NETWORKS

ResNet is a powerful backbone model that is used very frequently in many computer vision tasks. It has been seen that as we go adding on more layers to the neural network, it becomes difficult to train them and the accuracy starts saturating and then degrades also. Res-Net is a network that is 50 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224. Residual Network is a specific type of deep learning network that was introduced to solve complex problems by adding some additional layers in deep learning networks which results in improved accuracy and performance. The process of training a model involves providing an algorithm (that is, the learning algorithm) with training data to learn from.The term model refers to the model artifact that is created by the training process. The training data must contain the correct answer, which is known as a target or target attribute. The learning algorithm finds patterns in the training data that map the input data attributes to the target (the answer that is to be predicted), and it outputs a model that captures these patterns. This model can be used to get predictions on new data for which the target is unknown. ResNet Architecure Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset. Training deep learning neural network models on more data can result in more skilful models and the augmentation techniques can create variations of the images that can improve the ability of the fit models to generalise what they have learned to new images. In our case the number of images is less.So, data augmentation will help in increasing the number of images and supply the images to the model in batches. The images in the batches are not repeated so it also helps in preventing the overfitting of model. Keras ImageDataGenerator (only rotation_range,zoom_range and horizontal_flip was used)generate training data from the numpy arrays in batches and process them with their labels. Training data was also shuffled during training, while validation data was used to get the validation accuracy and validation loss during training. After data augmentation is completed, both training generator and testing generator is built for creating the model.Keras- ImageDataGenerator has a method called flow_from_directory which is used to bulid the generator. It takes the target path where the data is saved, a batch size which specifies the number of images yielded per batch, class_mode which specifies the type of labels used and also shuffle and seed attributes can be enabled. ResNet-101 is a convolutional neural network that is 101 layers deep. Once all the pre-processing for the model is done, ResNet model is created using tensrflow keras library. Weights for the model is pretrained and used. Now additional layers are added to get much more accurate results. Since a very deep network is used, there is a very high chance for error. Now a fully connected layer is added. The fully connected layer contains flatten, dense and dropout layers. First layer is the flatten layer which is used to feed the output of convolutional layer into the dense layer. Once it is fed into the dense layer, it maps the scores of the convolutional layer into the correct labels which activates the function. In this case SoftMax function is used which converts the scores into probabilities that sum to 1.

At the end, dropout layer is added which is used to prevent the neural network from overfitting during the training. This prevents the network to learn redundant information. Batch normalization is a recently developed technique which tries to properly initializing neural networks by explicitly forcing the activations throughout a network to take a unit Gaussian distribution at the beginning of the training. In practice, we put the Batch Normalization layers right after dense layers. Networks that use Batch Normalization are significantly more robust to bad initialisation, because normalisation greatly reduces the ability of small number of outlying inputs to over-influence the training. It also tends to reduce the overfitting. Additionally, it can be interpreted as doing pre-processing at each layer, but integrated into the network itself. In our case, batch normalisation is applied in the model to prevent arbitrary large weights in the intermediate layers. As the batch normalization normalizes, the intermediate layers, help to converge well. In our case early stopping is used by monitoring the val_loss till best fit model is reached. Once, the best fit model is got, the keras ModelCheckpoint is used to auto save the best fit training model and the weights for our training data. Now the model being completed. It is compiled using Adam as the optimizer which is an update to RMSProp optimizer in which the running average of both the gradients and their magnitude is used. Adam is much better than RMSProp because Adadelta learns too fast. Now the model is trained and validated using Keras fir generator which executes the training of the generator. Once the best fit is reached the training will stop and the best model and its weight are saved. Here ResNet comes into rescue and helps solve this problem. In this network, a technique called skip connections is used as shown in the figure 3. The skip connection skips training from a few layers and connects directly to the output. The advantage of adding this type of skip connection is because if any layer hurt the performance of architecture then it will be skipped by regularization. Without using this skip connection, the input 'x' gets multiplied by the weights of the layer followed by adding a bias term. This term goes through the activation function, f() and we get our output as H(x).This network uses a 34-layer plain network architecture inspired by VGG- 19 in which then the shortcut connection is added. These shortcut connections then convert the

architecture into residual network. Figure 1.6 shows the architecture of Resnet.

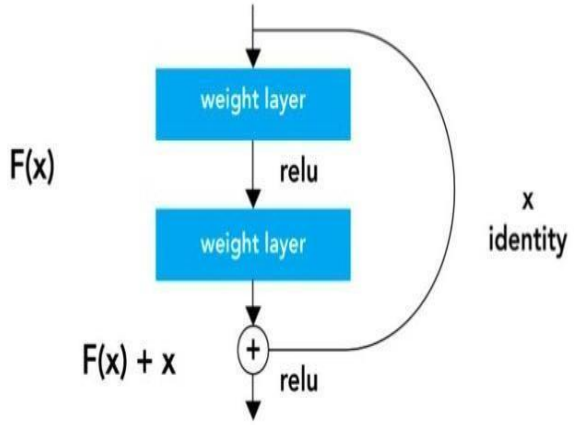


Figure 3: ResNet Model

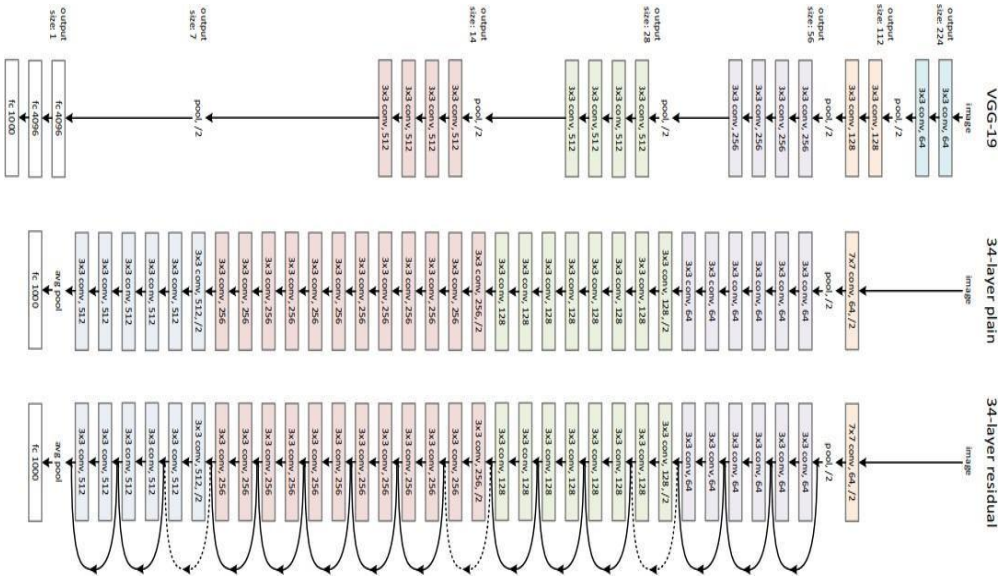


Figure 4: ResNet Architecture

1.5 LSTM ARCHITECTURE

LSTM networks are an extension of recurrent neural networks (RNNs) mainly introduced to handle situations where RNNs fail. Talking about RNN, it is a network that works on the present input by taking into consideration the previous output (feedback) and storing in its memory for a short period of time (short-term memory). Out of its various applications, the most popular ones are in the fields of speech processing, non-Markovian control, and music composition. Nevertheless, there are drawbacks to RNNs. First, it fails to store information for a longer period of time. At times, a reference to certain information stored quite a long time ago is required to predict the current output. But RNNs are absolutely incapable of handling such "long-term dependencies". Second, there is no finer control over which part of the context needs to be carried forward and how much of the past needs to be 'forgotten'. Thus, Long Short-Term Memory (LSTM) was brought into the picture. It has been so designed that the vanishing gradient problem is almost completely removed, while the training model is left unaltered. Long-time lags in certain problems are bridged using LSTMs where they also handle noise, distributed representations, and continuous values. LSTMs provide us with a large range of parameters such as learning rates, and input and output biases. Hence, no need for fine adjustments. Figure 3.4 shows the architecture of LSTM. To conclude, long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feed forward neural networks, LSTM has feedback connections. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell

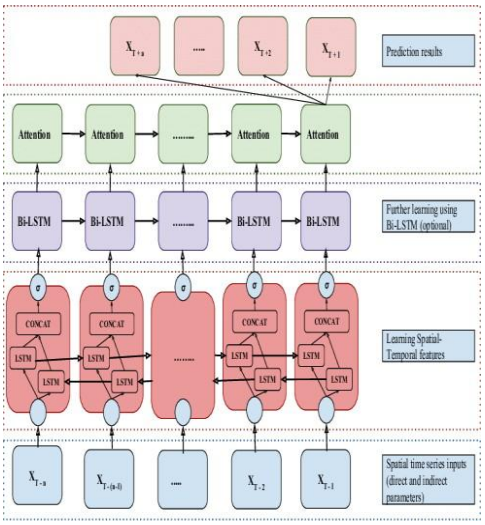


Figure 5: LSTM Architecture

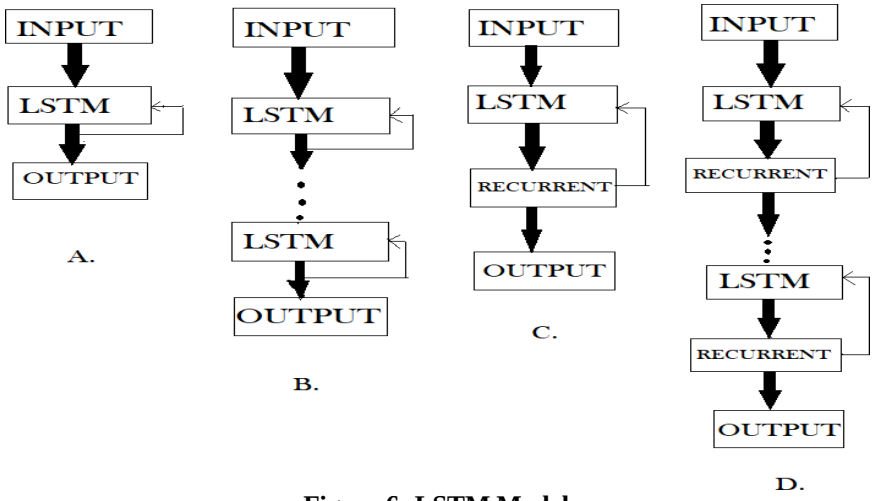


Figure 6: LSTM Model

The figure 3.5 shows the LSTM model which is explained below in detail.

1. Figure-A represents what a basic LSTM network looks like. Only one layer of LSTM between an input and output layer has been shown here.
2. Figure-B represents Deep LSTM which includes a number of LSTM layers in between the input and output. The advantage is that the input values fed to the network not only go through several LSTM layers but also propagate through time within one LSTM cell. Hence, parameters are well distributed within multiple layers. This results in a thorough process of inputs in each time step.
3. Figure-C represents LSTM with the Recurrent Projection layer where the recurrent connections are taken from the projection layer to the LSTM layer input. This architecture was designed to reduce the high learning computational complexity (O(N)) for each time step) of the standard LSTM RNN.
4. Figure-D represents Deep LSTM with a Recurrent Projection Layer consisting of multiple LSTM layers where each layer has its own projection layer. The increased depth is quite useful in the case where the memory size is too large. Having increased depth prevents overfitting in models as the inputs to the network need to go through many nonlinear functions.

II. RESULT AND DISCUSSION

2.1 TESTING THE MODEL

The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset. The test dataset provides the gold standard used to evaluate the model. It is only used once a model is completely trained (using the train and validation sets) which is shown in the figure 7.

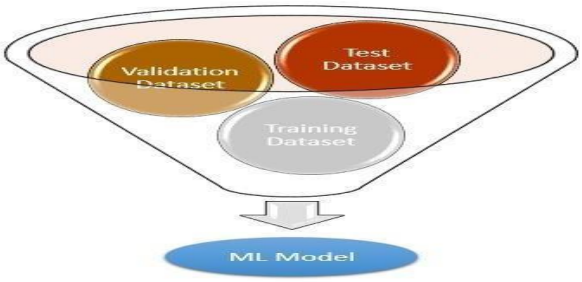
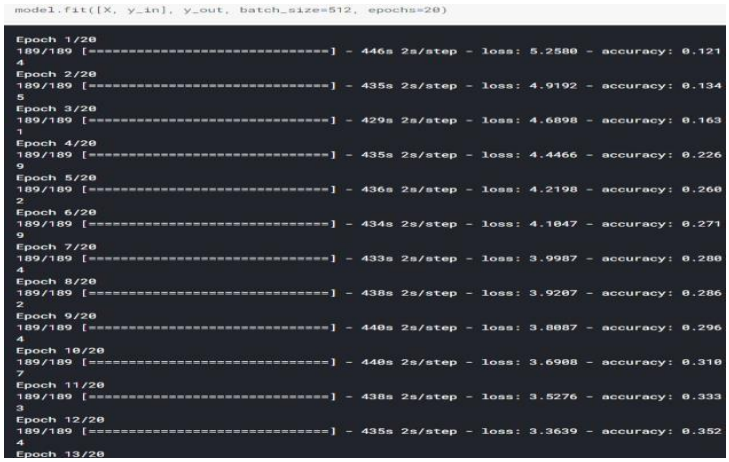


Figure 7: Testing Model

1. Tester first defines three datasets, training dataset (65%), validation dataset (20%) and test dataset (15%).
2. Tester once defines the data set, Will begin to train the models with the training dataset. Once this training model is done, the tester then performs to evaluate the models with the validation dataset. This is iterative and can embrace any tweaks/changes needed for a model based on results that can be done and re-evaluated. This ensures that the test dataset remains unused and can be used to test an evaluated model.
3. Once the evaluation of all the models is done, the best model that the team feels confident about based on the least error rate and high approximate prediction will be picked and tested with a test dataset to ensure the model still performs well and matches with validation dataset results. If you find the model accuracy is high then you must ensure that test/validation sets are not leaked into your training dataset.

2.2 IMPLEMENTATION

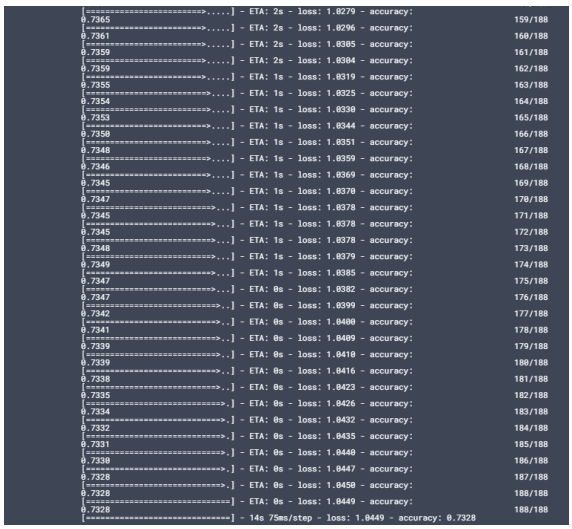


```
model.fit([X, y_in], y_out, batch_size=512, epochs=20)

Epoch 1/20
189/189 [==============================] - 446s 2s/step - loss: 5.2580 - accuracy: 0.121
4
Epoch 2/20
189/189 [==============================] - 435s 2s/step - loss: 4.9192 - accuracy: 0.134
5
Epoch 3/20
189/189 [==============================] - 429s 2s/step - loss: 4.6898 - accuracy: 0.163
1
Epoch 4/20
189/189 [==============================] - 435s 2s/step - loss: 4.4466 - accuracy: 0.226
9
Epoch 5/20
189/189 [==============================] - 436s 2s/step - loss: 4.2198 - accuracy: 0.260
2
Epoch 6/20
189/189 [==============================] - 434s 2s/step - loss: 4.1047 - accuracy: 0.271
9
Epoch 7/20
189/189 [==============================] - 433s 2s/step - loss: 3.9987 - accuracy: 0.280
4
Epoch 8/20
189/189 [==============================] - 438s 2s/step - loss: 3.9207 - accuracy: 0.286
2
Epoch 9/20
189/189 [==============================] - 440s 2s/step - loss: 3.8087 - accuracy: 0.296
4
Epoch 10/20
189/189 [==============================] - 440s 2s/step - loss: 3.6988 - accuracy: 0.310
7
Epoch 11/20
189/189 [==============================] - 438s 2s/step - loss: 3.5276 - accuracy: 0.333
3
Epoch 12/20
189/189 [==============================] - 435s 2s/step - loss: 3.3639 - accuracy: 0.352
4
Epoch 13/20
```

Figure 8: Epochs

The figure 2.2 shows the epochs ran during the training of the DenseNet model. It also shows the time taken by each epochs, loss, accuracy, and validation loss and validation accuracy of each epoch



```
[==========================>.....] - ETA: 2s - loss: 1.0279 - accuracy:
0.7365                                                                    159/188
[==========================>.....] - ETA: 2s - loss: 1.0296 - accuracy:
0.7361                                                                    160/188
[==========================>.....] - ETA: 2s - loss: 1.0305 - accuracy:
0.7359                                                                    161/188
[==========================>.....] - ETA: 2s - loss: 1.0304 - accuracy:
0.7359                                                                    162/188
[==========================>.....] - ETA: 1s - loss: 1.0319 - accuracy:
0.7355                                                                    163/188
[==========================>.....] - ETA: 1s - loss: 1.0325 - accuracy:
0.7354                                                                    164/188
[==========================>.....] - ETA: 1s - loss: 1.0330 - accuracy:
0.7353                                                                    165/188
[==========================>.....] - ETA: 1s - loss: 1.0344 - accuracy:
0.7350                                                                    166/188
[==========================>.....] - ETA: 1s - loss: 1.0351 - accuracy:
0.7348                                                                    167/188
[==========================>.....] - ETA: 1s - loss: 1.0359 - accuracy:
0.7346                                                                    168/188
[==========================>.....] - ETA: 1s - loss: 1.0369 - accuracy:
0.7345                                                                    169/188
[==========================>.....] - ETA: 1s - loss: 1.0370 - accuracy:
0.7347                                                                    170/188
[==========================>.....] - ETA: 1s - loss: 1.0378 - accuracy:
0.7345                                                                    171/188
[==========================>.....] - ETA: 1s - loss: 1.0378 - accuracy:
0.7345                                                                    172/188
[==========================>.....] - ETA: 1s - loss: 1.0378 - accuracy:
0.7348                                                                    173/188
[==========================>.....] - ETA: 1s - loss: 1.0379 - accuracy:
0.7349                                                                    174/188
[==========================>.....] - ETA: 1s - loss: 1.0385 - accuracy:
0.7347                                                                    175/188
[==========================>.....] - ETA: 0s - loss: 1.0382 - accuracy:
0.7347                                                                    176/188
[==========================>.....] - ETA: 0s - loss: 1.0399 - accuracy:
0.7342                                                                    177/188
[==========================>.....] - ETA: 0s - loss: 1.0400 - accuracy:
0.7341                                                                    178/188
[==========================>.....] - ETA: 0s - loss: 1.0409 - accuracy:
0.7339                                                                    179/188
[==========================>.....] - ETA: 0s - loss: 1.0410 - accuracy:
0.7339                                                                    180/188
[==========================>.....] - ETA: 0s - loss: 1.0416 - accuracy:
0.7338                                                                    181/188
[==========================>.....] - ETA: 0s - loss: 1.0423 - accuracy:
0.7335                                                                    182/188
[==========================>.....] - ETA: 0s - loss: 1.0426 - accuracy:
0.7334                                                                    183/188
[==========================>.....] - ETA: 0s - loss: 1.0432 - accuracy:
0.7332                                                                    184/188
[==========================>.....] - ETA: 0s - loss: 1.0435 - accuracy:
0.7331                                                                    185/188
[==========================>.....] - ETA: 0s - loss: 1.0440 - accuracy:
0.7330                                                                    186/188
[==========================>.....] - ETA: 0s - loss: 1.0447 - accuracy:
0.7328                                                                    187/188
[==========================>.....] - ETA: 0s - loss: 1.0450 - accuracy:
0.7328                                                                    188/188
[==============================] - ETA: 0s - loss: 1.0449 - accuracy:
0.7328                                                                    188/188
[==============================] - 14s 75ms/step - loss: 1.0449 - accuracy: 0.7328
```

Figure 9: Final Accuracy and Loss

The Figure 2.3, shows the final accuracy and final loss of the trained model by using validation data.

2.3 RESULT

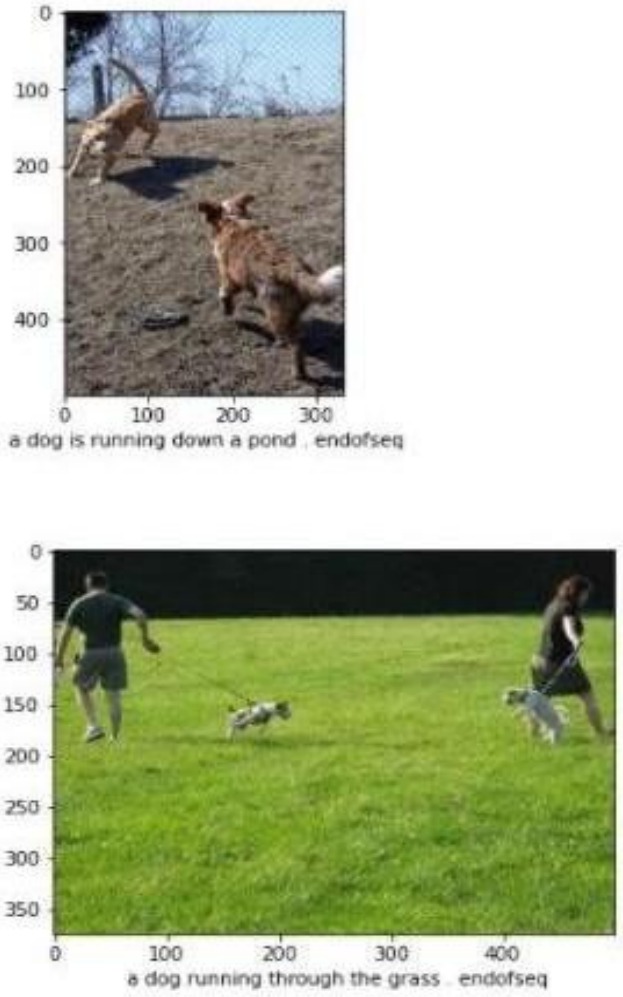


Figure 10: Final Detection.

The figure 10, shows the final output of the Image captioning model, when a random image from the dataset is provided for prediction.

III. CONCLUSION

In this proposal, we have presented one single joint model for automatic image captioning based on ResNet50 and LSTM with software attention. The proposed model was designed with one encoder- decoder architecture. We adopted ResNet50, a convolutional neural network, as the encoder to encode an image into a compact representation as the graphical features. After that, a language model LSTM was selected as the decoder to generate the description sentence. Meanwhile, we integrated the soft attention model with LSTM such that the learning can be focused on a particular part of the image to improve the performance. The whole model is fully trainable by using the stochastic gradient descent that makes the training process easier. The experimental evaluations indicate that the proposed model is able to generate good captions for images automatically.

REFERENCES

[1]. Donahue, J., Anne Hendricks, L., Guadarrama, M., Venugopalan, S., Saenko, K., & Darrell, T. (2015), 'Long-term recurrent convolutional networks for visual recognition and description. In Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 2625- 2634.

[2]. Devlin, J., Cheng, H., Fang, H., Gupta, S., Deng, L., He, X. & Mitchell, M. (2015) 'Language models for image captioning: The quirks and what works', pp- 155.809.

[3]. Farhadi, M. Hejrati, M. A. Sadeghi et al., (2010), 'Every picture tells a story: generating sentences from images, in Computer Vision', pp. 15–29.

[4]. Gong. Y, L. Wang, M. Hodosh, J. Hockenmaier, and S. Lazebnik, (2010) 'Improving image sentence embeddings using large

weakly annotated photo collections, in European Conference on Computer Vision', pp. 529– 545.
[5]. Hodosh.M , P. Young, and J. Hockenmaier, (2013), 'Framing image description as a ranking task: Data, models and evaluation metrics, Journal of Artificial Intelligence Research', vol. 47, pp. 853–899.
[6]. Karpathy, A., & Fei-Fei, L.(2013), 'Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 3128-3137
[7]. Kulkarni. G, V. Premraj, S. Dhar et al., (2011),' Baby talk: understanding and generating image descriptions, in CVPR means IEEE Conference on Computer Vision and Pattern Recognition', pp. 2891– 2903.
[8]. Li, G. Kulkarni, T. L. Berg, A. C. Berg, and Y. J. Choi, (2011), 'Composing simple image descriptions using web-scale n-grams, in Proceedings of the Fifteenth Conference on Computational Natural Language Learning. Association for Computational Linguistics', pp. 212- 1153\
[9]. Ordonez. V, G. Kulkarni, and T. L. Berg, (2010), 'Im2Text: Describing images using 1 million captioned photographs, Advances in Neural Information Processing Systems', pp. 1143–1