ATAL BIHARI VAJPAYEE INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, GWALIOR

विश्वजीवनामृतं ज्ञानम्

MINI PROJECT

Neural Image Caption Generator

Project Report

Submitted To -
Dr Somesh Kumar

Submitted by -
Amit Kumar (2017-IMT-009)
Ankit Meena(2017-IMT-012)
Vishal Jain(2017-IMT-090)
Vishal Shivhare(2017-IMT-091)

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend our sincere thanks to all of them.

We are highly indebted to Dr. Somesh Kumar for his guidance and constant supervision as well as for providing necessary information regarding the project & also for his support in completing the project.

We would like to express our gratitude towards our parents for their kind cooperation and encouragement which helped us in completion of this project.

Our thanks and appreciations also go to our colleagues in developing the project and people who have willingly helped us out with their abilities.

INDEX

Chapter 1

INTRODUCTION

1.1 BACKGROUND

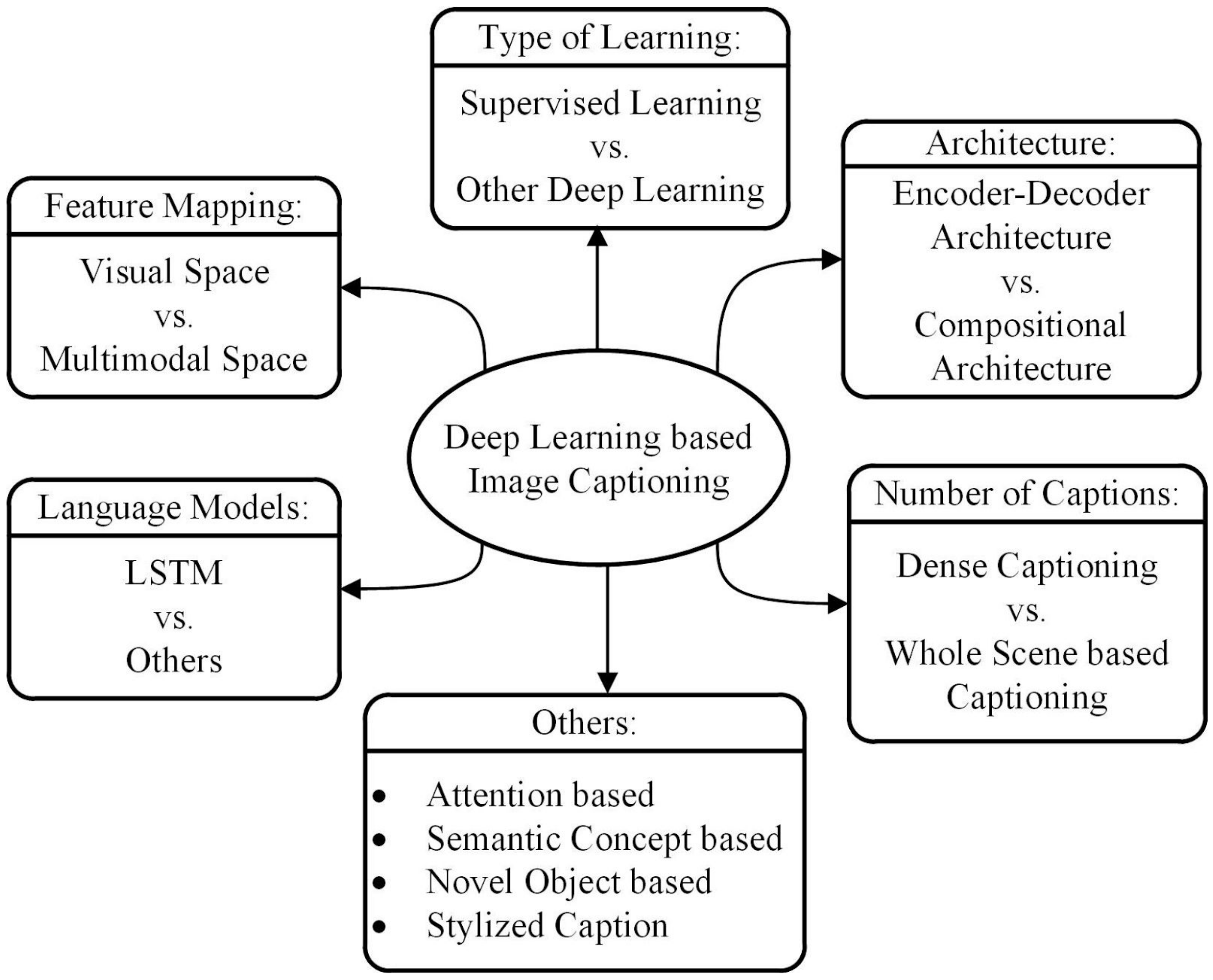
- The accuracy of generating captions depends on human expertise in order to generate or verify them.
- A wide variety of companies like microsoft's captionbot ai uses a deep learning approach to automatically generate captions.
- Image captioning is important for automatic image indexing, which is important for Content-Based Image Retrieval (CBIR) .

1.2 MOTIVATION

- The recent boom in the field of deep learning especially CNN has helped in creating better feature extraction models.
- In recent years there are various state of the art pretrained models developed for handling images like resnet, vg, mobilenet etc.

1.3DEEP LEARNING FOR IMAGE CAPTIONING

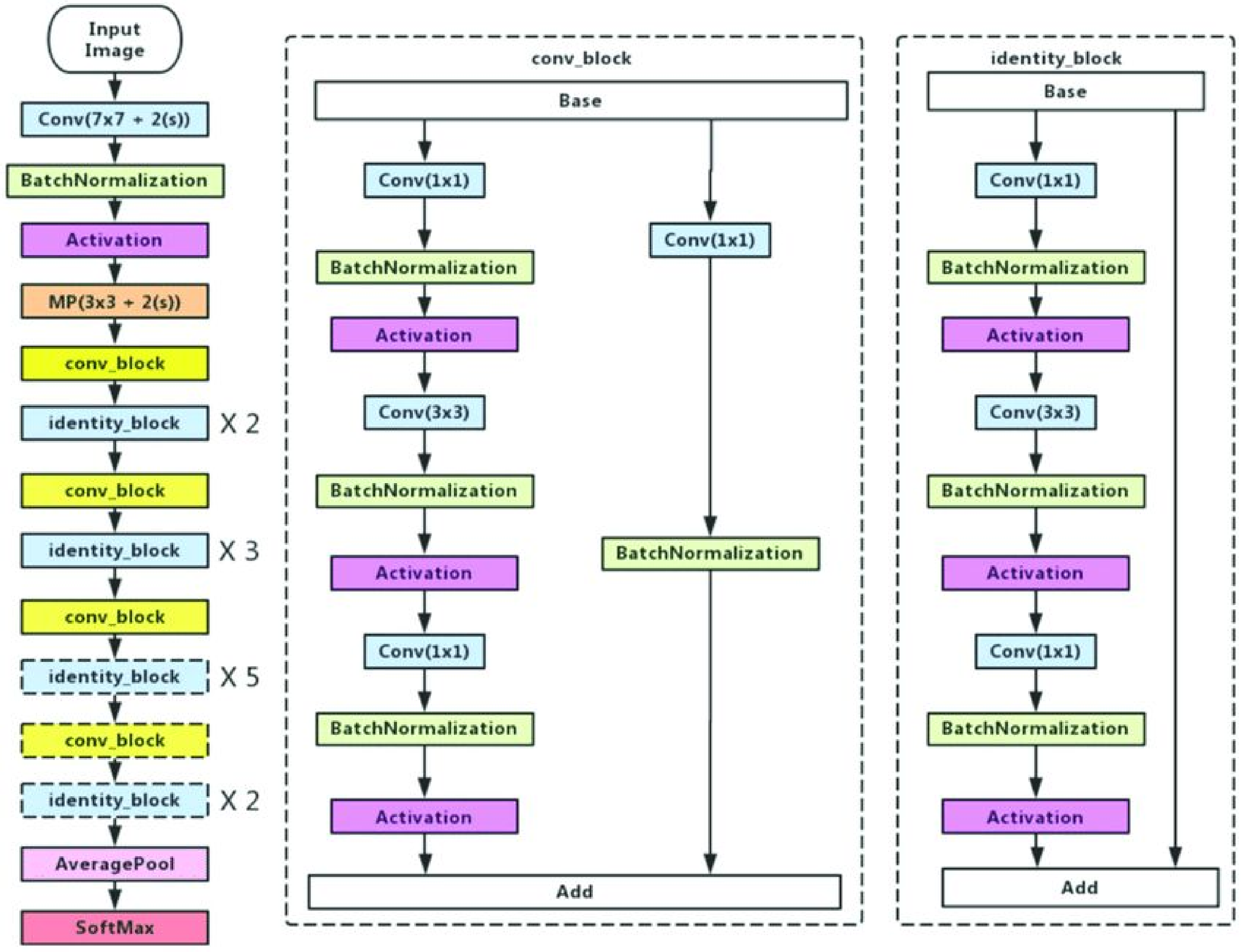
We use the deep learning algorithms for extracting features from images and generating captions. User can provide a new image that can be run on our trained classifier and the app will provide the caption for the image.


Type of Learning:
Supervised Learning
vs.
Other Deep Learning
Architecture:
Encoder-Decoder Architecture
vs.
Compositional Architecture
Feature Mapping:
Visual Space
vs.
Multimodal Space
Deep Learning based Image Captioning
Language Models:
LSTM
vs.
Others
Number of Captions:
Dense Captioning
vs.
Whole Scene based Captioning
Others:
Attention based
Semantic Concept based
Novel Object based
Stylized Caption

1.3.1 **Convolutional Neural Network**

A CNN is a deep neural network that is used for image extraction tasks in deep learning. There are multiple types of layers used in CNNs, some of which are listed below-

- **Input Layer:** " It is made up of artificial input neurons, and brings the initial data into the system for processing by the following layers of artificial neurons. The input layer is the starting of the workflow in the neural network."
- **Hidden Layers: "**It is situated between input and output layers of a neural network, hidden layers perform non linear transformations of the inputs coming in the network and the function applies weights to the inputs and directs them through an activation function as output.**"**
- **Output Layer: "** In an artificial neural network, the output layer is the final layer of neurons that produces required outputs for the neural network in form of probabilities."
- **Activation Function:** "It tells us that should a neuron be activated or not by finding a weighted sum and further adding bias to it."
- **Batch normalization: "**It is a technique used for training large neural networks that treats the inputs as a mini-batch. This has the effect of stabilizing the learning process and reducing the number of training epochs required to train large networks."
- **Average pooling: "**it performs down-sampling by breaking down the input into pooling regions and computing the average values over each pooling region."
- **Convolution: "**It is performed on the input data with the use of a filter or kernel by sliding the filter over the input to then produce a feature map."
- **Fully Connected - "**It is a typical layer which is obtained from previous layer by using a parameter matrix."



Architecture of Resnet 50

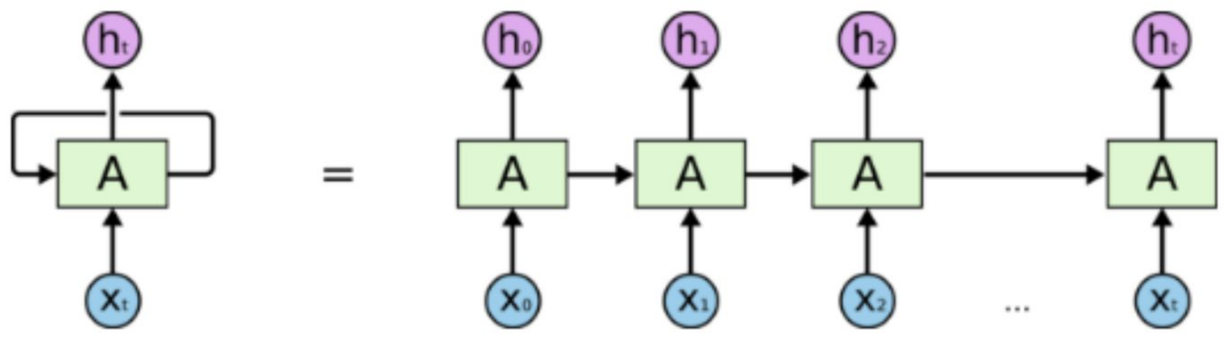
1.4 PYTHON

We have used the **PYTHON** to implement our project. Python is an "interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together". Python's simple programming language, easy to learn syntax have readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse.

Chapter 2

Used Algorithm and Methodology

2.1 Recurrent Neural Network



An unrolled recurrent neural network.

A RNN model referred as Recurrent neural network is used in the calculation of activation vectors(i.e feature map). The input is first fed to the embedding layer from that we get a metric of activation then this metric fed to the first block of Recurrent neural network and then it outputs a word with maximum likelihood.

Then this word with maximum likelihood is multiplied to the activation matric of the previous state and this gives the activation matric of the current state.

This function is repeated till we get an end sequence and maximum likelihood estimated value.

Recurrent neural networks a particularly long short-term memory to give an example of how long short-term memory works we will consider the question of what's for dinner let's say for a minute that you are a very lucky apartment dweller and you have a flatmate who loves to cook dinner every night he cooks one of three things

For the representation of images, we use a CNN. CNNs are primarily used in the study of images, and are currently the best for object identification and detection. Our particular choice of CNN uses a approach of batch normalization and produces the current best performance in the ILSVRC 2014 classification competition

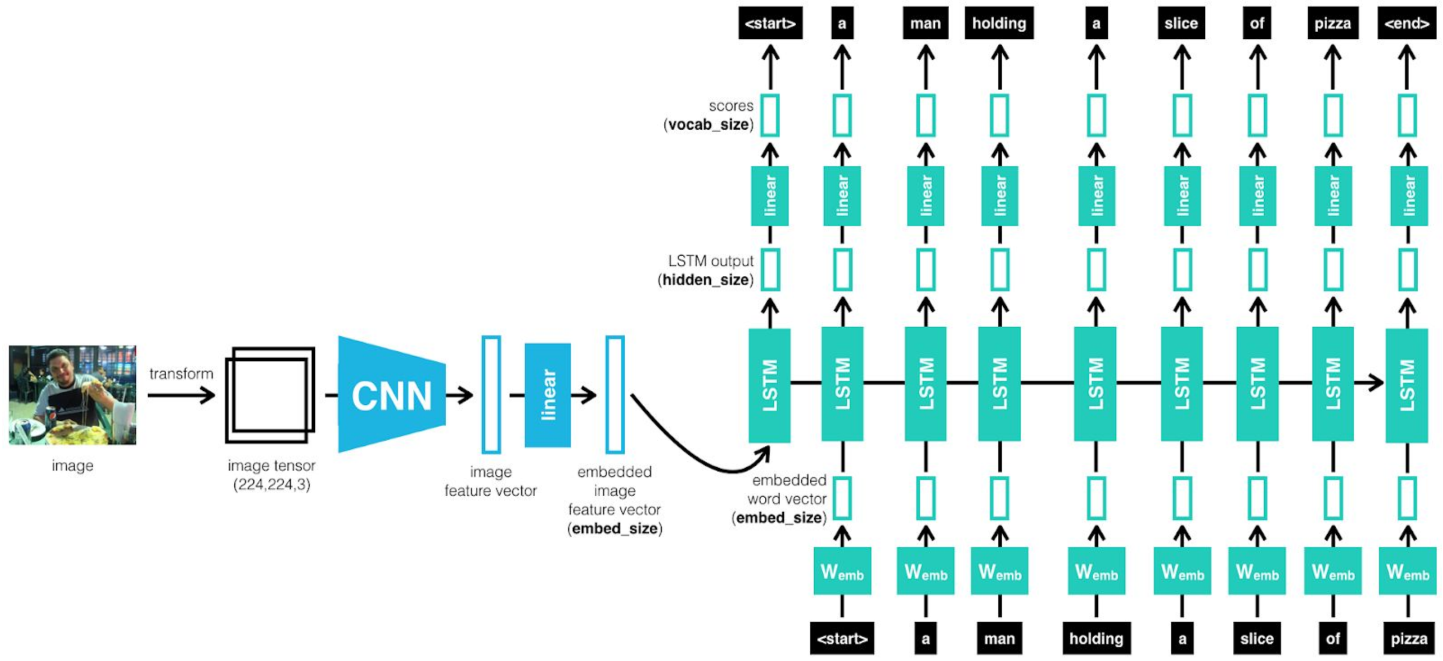
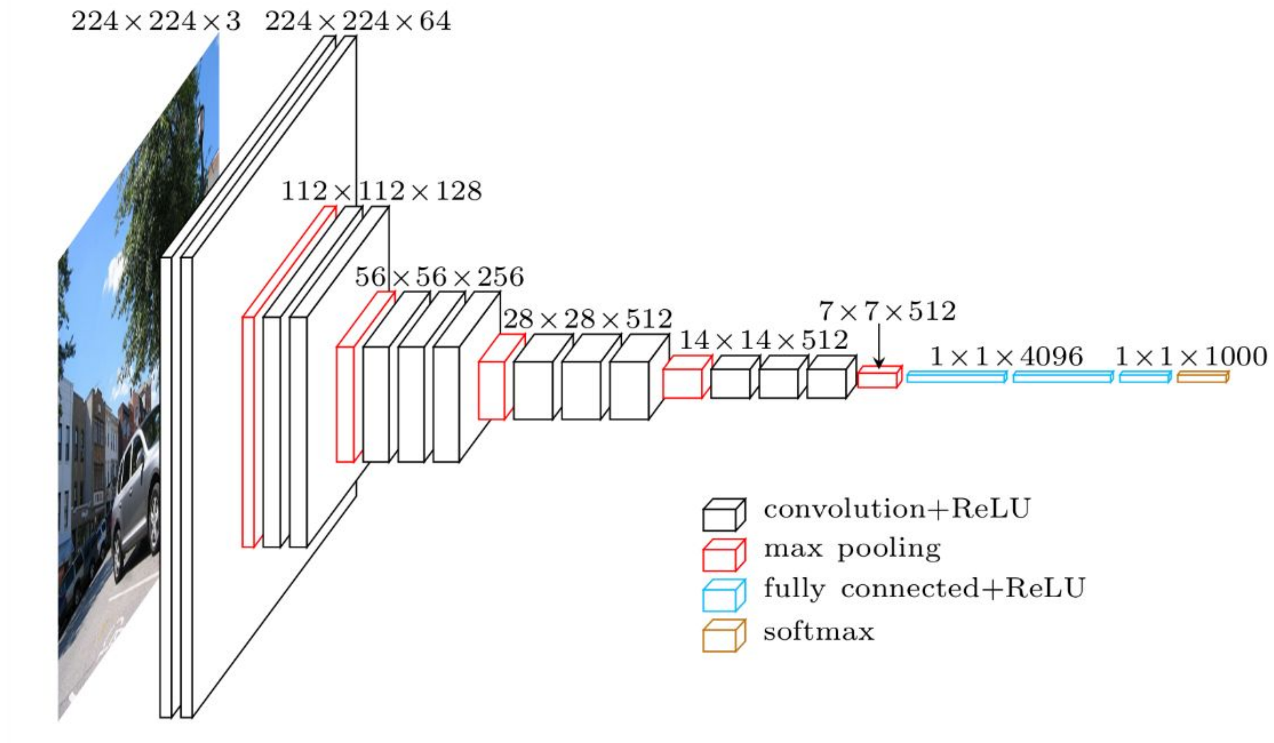


image
transform
image tensor (224,224,3)
CNN
image feature vector
linear
embedded image feature vector (embed_size)
scores (vocab_size)
LSTM output (hidden_size)
LSTM
linear
embedded word vector (embed_size)
W_emb
<start> a man holding a slice of pizza <end>
<start> a man holding a slice of pizza

2.2 "Image feature Extraction using Transfer learning"

Transfer learning a technique in which we extract the features from the target with the help of a pretrained model. The pretrained means the model is previously trained on the particular dataset and hyperparameters and weights are stored. So that these models can be reused for feature extraction.Like in this we are using a Pretrained Resnet model which was trained on imagenet dataset.



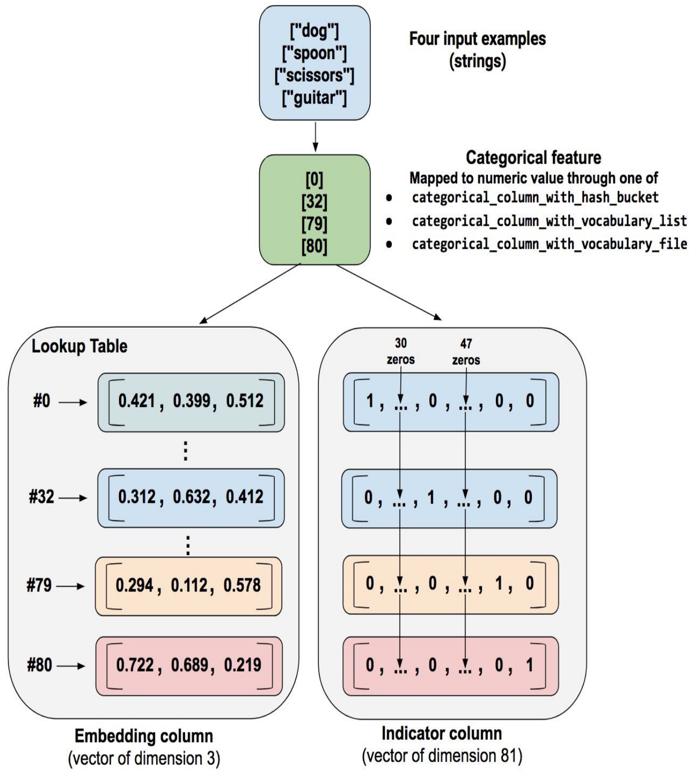
2.3 Resnet model

- Input size : 224x224x3
- Activation vector size: 7x7x512
- res5c_branch2c (Conv2D) : (None, 7, 7, 2048)
- output Gap layer : (None, 2048)

2.4 "Text Feature Engineering using Glove vector"

Word Embeddings are used to "convert texts into numerical representation".These numerical values are close for a given category/set.

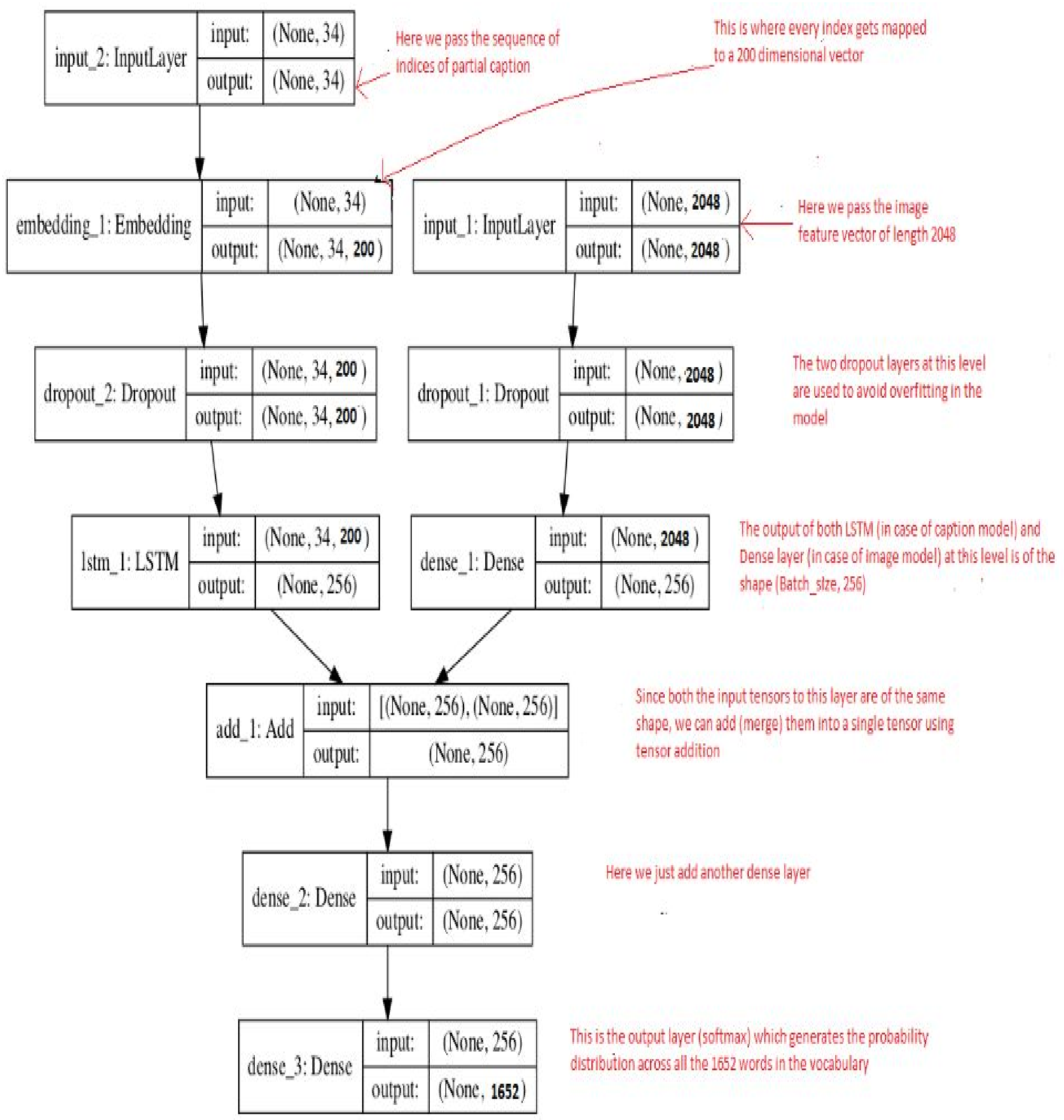
.



For a given image

- Train : 5 sentences
- Total Words 373837
- Total unique word (vocab):8424
- Reducing vocab size
- Thresholding : 1652+2
- Embedding matrix shape: (1654, 200)
- Max length of sentence : 34

2.5 Model Architecture


input_2: InputLayer
input: (None, 34)
output: (None, 34)
Here we pass the sequence of indices of partial caption
This is where every index gets mapped to a 200 dimensional vector
embedding_1: Embedding
input: (None, 34)
output: (None, 34, 200)
input_1: InputLayer
input: (None, 2048)
output: (None, 2048)
Here we pass the image feature vector of length 2048
dropout_2: Dropout
input: (None, 34, 200)
output: (None, 34, 200)
dropout_1: Dropout
input: (None, 2048)
output: (None, 2048)
The two dropout layers at this level are used to avoid overfitting in the model
lstm_1: LSTM
input: (None, 34, 200)
output: (None, 256)
dense_1: Dense
input: (None, 2048)
output: (None, 256)
The output of both LSTM (in case of caption model) and Dense layer (in case of image model) at this level is of the shape (Batch_size, 256)
add_1: Add
input: [(None, 256), (None, 256)]
output: (None, 256)
Since both the input tensors to this layer are of the same shape, we can add (merge) them into a single tensor using tensor addition
dense_2: Dense
input: (None, 256)
output: (None, 256)
Here we just add another dense layer
dense_3: Dense
input: (None, 256)
output: (None, 1652)
This is the output layer (softmax) which generates the probability distribution across all the 1652 words in the vocabulary

Chapter 3

Implementation Language and Source Code

3.1 Used Python Libraries

- Keras
- Numpy
- Pandas
- Regex
- Pillow
- Tensorflow
- Matplotlib
- Seaborn
- Flask

3.2 Source Code

Although The Complete Code is Very Long and Can’t be Completely Written in This Document ,Here are Main Model architecture

```
“”input_img_features = Input(shape=(2048,))
inp_img1 = Dropout(0.3)(input_img_features)
inp_img2 = Dense(256,activation='relu')(inp_img1)

# Captions as Input
input_captions = Input(shape=(max_len,))
inp_cap1 = Embedding(input_dim=vocab_size,output_dim=50,mask_zero=True)(input_captions)
inp_cap2 = Dropout(0.3)(inp_cap1)
inp_cap3 = Long Short Term Memory networks(256)(inp_cap2)
decoder1 = add([inp_img2,inp_cap3])
decoder2 = Dense(256,activation='relu')(decoder1)
outputs = Dense(vocab_size,activation='softmax')(decoder2)

# Combined Model
model = Model(inputs=[input_img_features,input_captions],outputs=outputs)
model.compile(loss='categorical_crossentropy',optimizer="adam")
epochs = 20
```

```
batch_size = 3
number_pics_per_batch = 60
steps = len(train_descriptions)//number_pics_per_batch
def train():
    for i in range(epochs):
        generator = data_generator(train_descriptions,encoding_train,word_to_idx,max_len,batch_size)
        model.fit_generator(generator,epochs=1,steps_per_epoch=steps,verbose=1)""
```

3.3 Discussion

Flask is the way to create an endpoint with Python, and then uses some of other Keras Libraries to work around when building an endpoint for predictions using Flask.

Making deep learning models available to end users or systems is challenging. Here we used some Basic HTML , CSS and Javascript to create Front End and then used the Flask as BackEnd and to Connect the Front End with the BackEnd .Some of the reasons of it being challenging are as follows :-

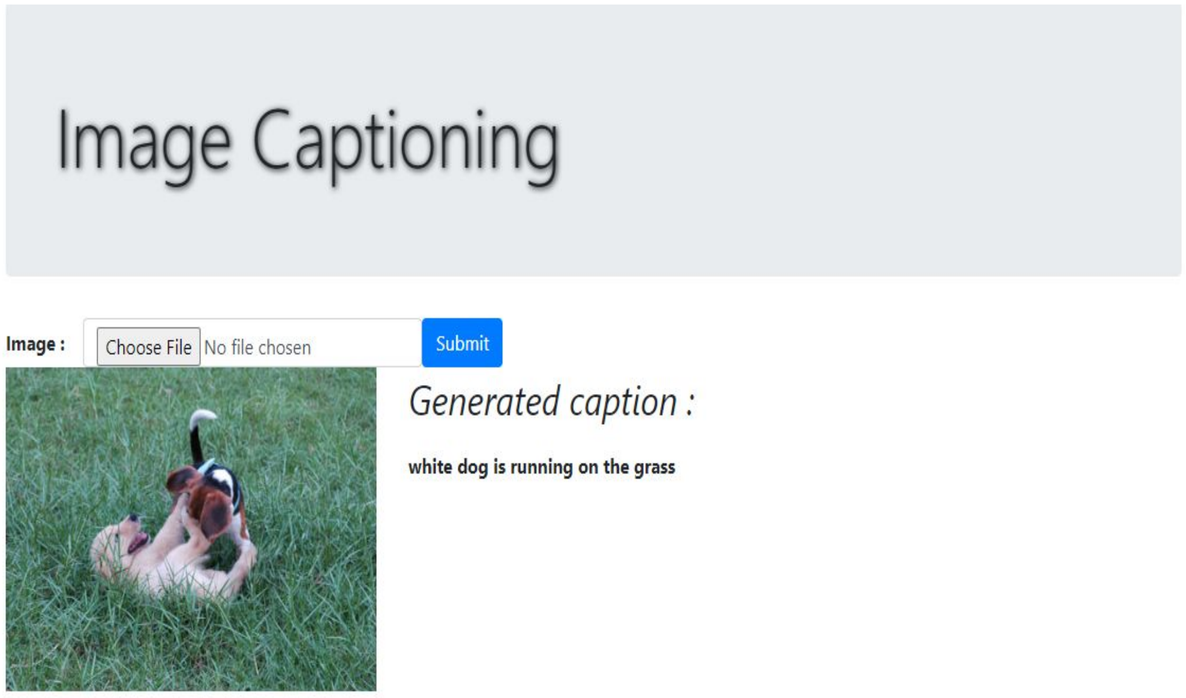
- **Model Serialization:** As we know that standard approaches for serializing the models, such as PMML, have only limited support. For e.g. keras2pmml lacks the relu activations, which suggests that the DataFlow plus PMML approach that I presented in my model production post isn't viable
- **Run Time:** We also know that both batch predictions and real-time predictions are so hard to scale, And as since most of my model prediction expertise is in Python. Flask in the place of Jetty, you got to use a Python library for creating estimates.

Using this , We can create a scalable service that takes an image as input and runs our model. Now we build a web app that sends a request to our service and expects the Caption of the image in response.

Chapter 4

RESULT AND EVALUATION

1 OUTPUT SCREENSHOTS (WEB BASED APPLICATION)



Generated caption :

white dog is running on the grass

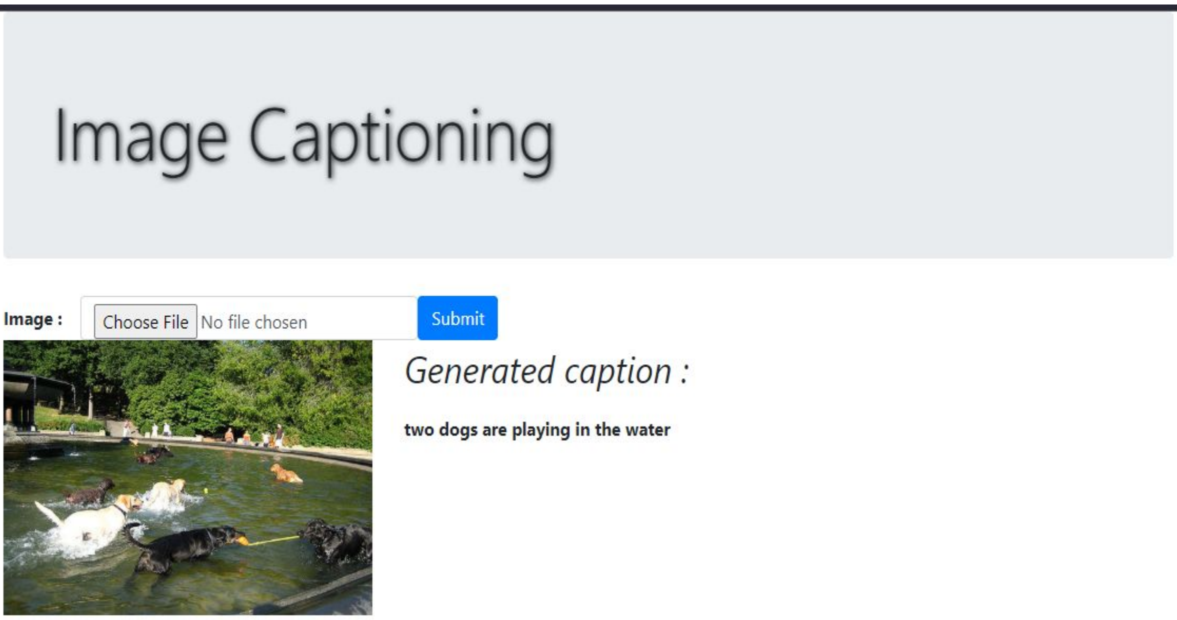


Image Captioning

Image : Choose File No file chosen Submit

Generated caption :

two dogs are playing in the water

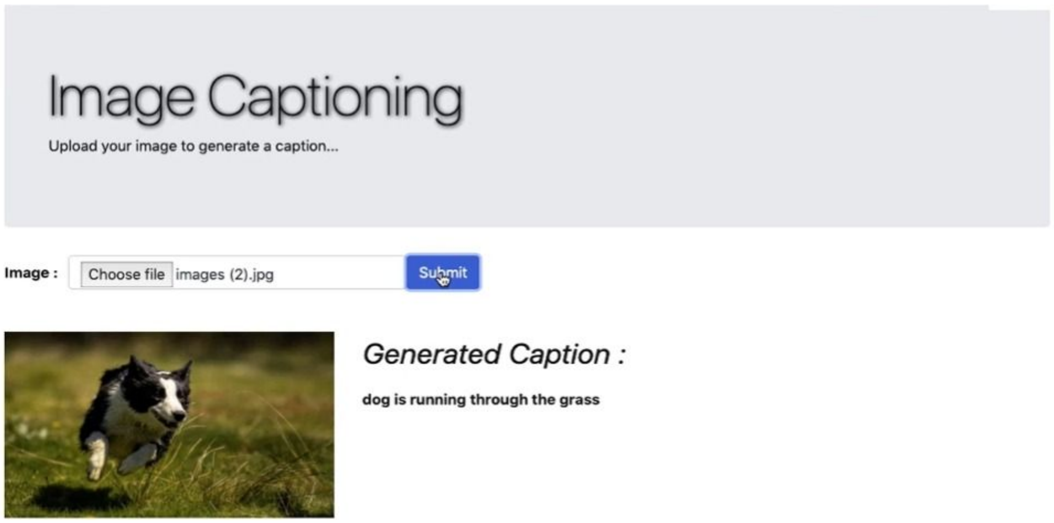


Image Captioning

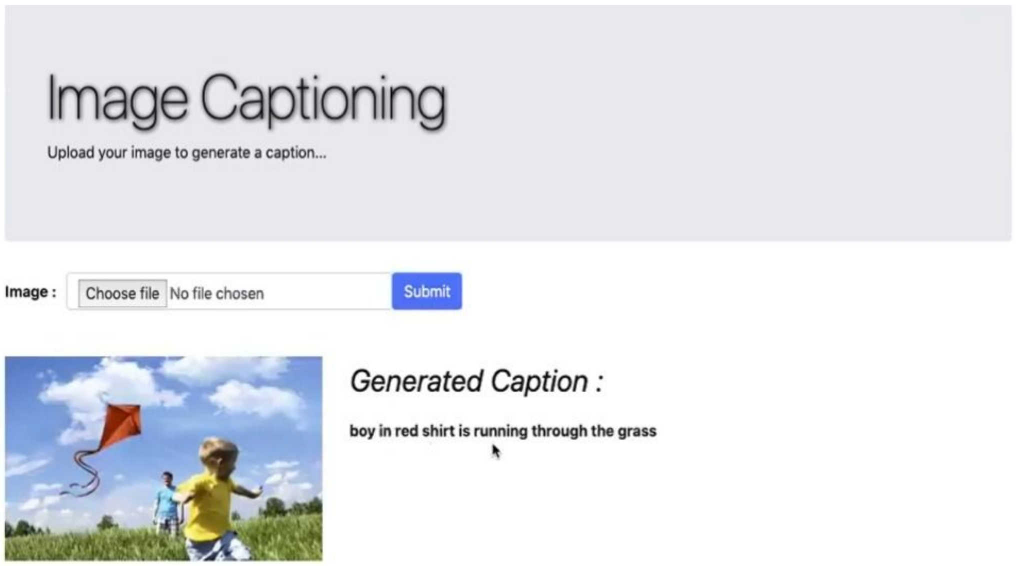
Upload your image to generate a caption...

Image : Choose file images (2).jpg Submit

Generated Caption :

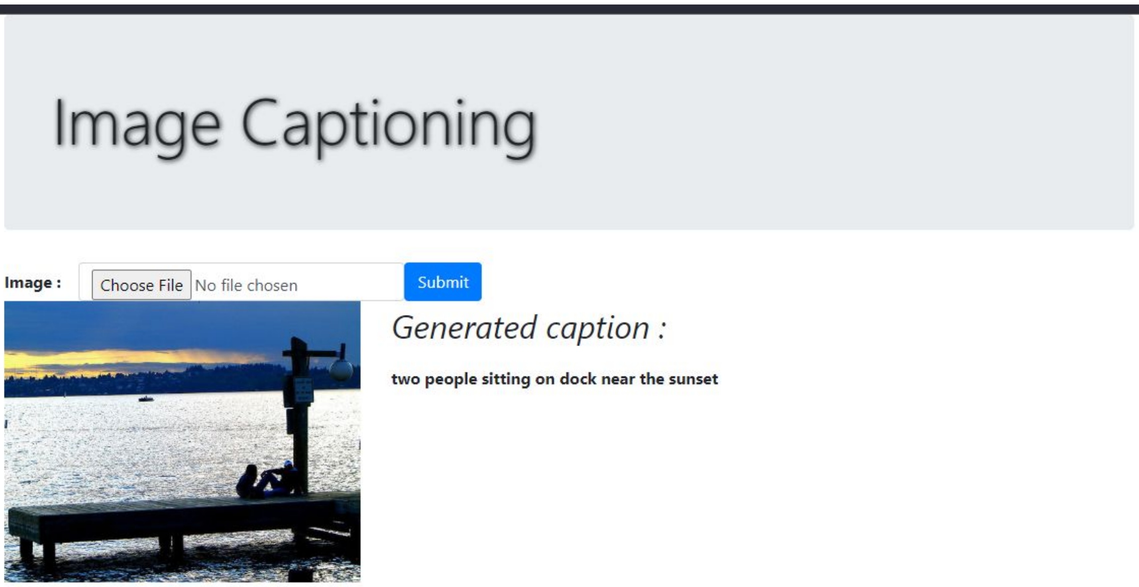
dog is running through the grass

Waiting for localhost...



Generated Caption :

boy in red shirt is running through the grass

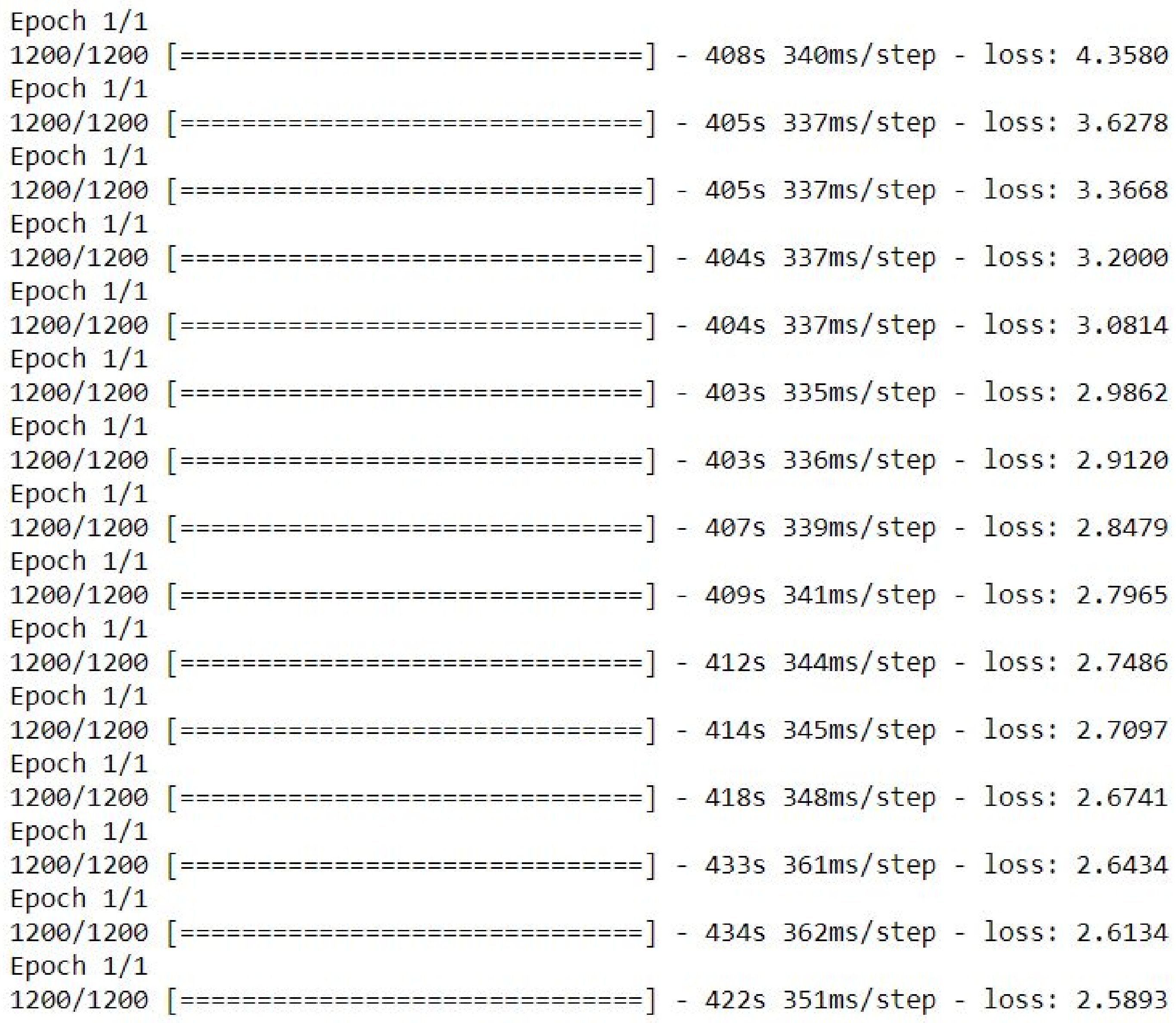


Generated caption :

two people sitting on dock near the sunset

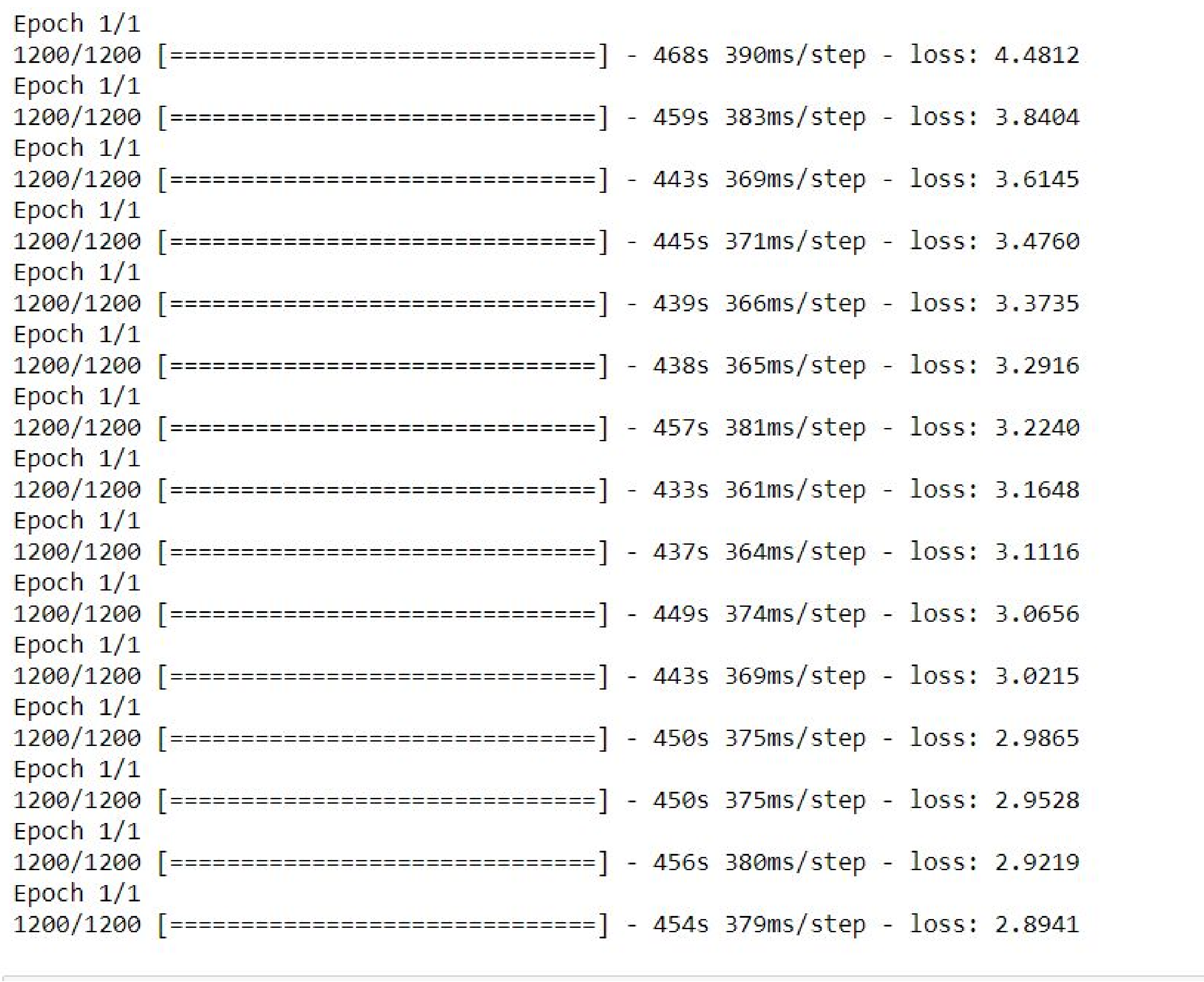
4.1 CATEGORICAL CROSSENTROPY LOSS FUNCTION

“It is the difference between the true value and predicted value. A loss function is for a single training example. It is also sometimes called an error function. It is a Softmax activation plus a Cross-Entropy loss. If we use this loss, we will train a CNN to output a probability over the classes for each image.**”**



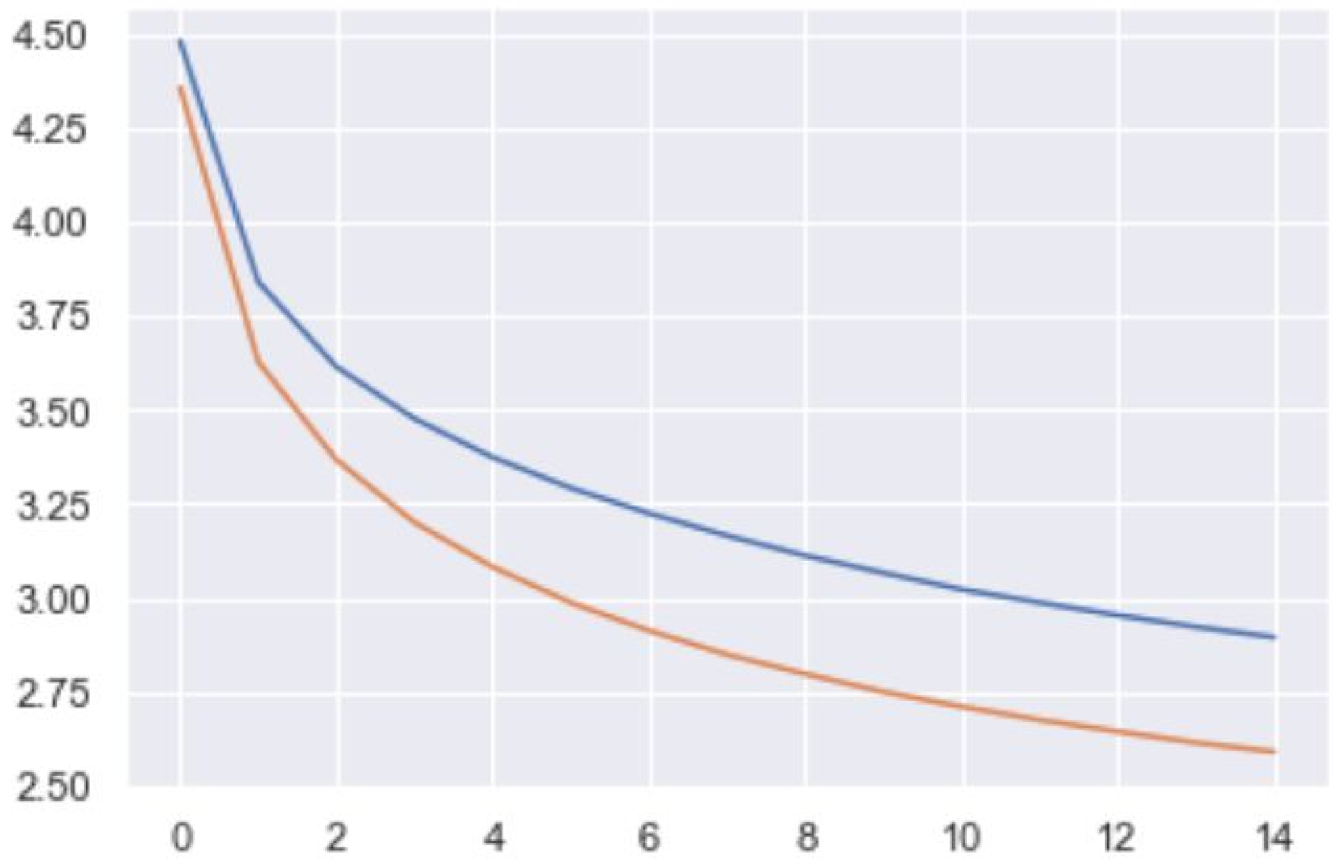
```
Epoch 1/1
1200/1200 [==============================] - 408s 340ms/step - loss: 4.3580
Epoch 1/1
1200/1200 [==============================] - 405s 337ms/step - loss: 3.6278
Epoch 1/1
1200/1200 [==============================] - 405s 337ms/step - loss: 3.3668
Epoch 1/1
1200/1200 [==============================] - 404s 337ms/step - loss: 3.2000
Epoch 1/1
1200/1200 [==============================] - 404s 337ms/step - loss: 3.0814
Epoch 1/1
1200/1200 [==============================] - 403s 335ms/step - loss: 2.9862
Epoch 1/1
1200/1200 [==============================] - 403s 336ms/step - loss: 2.9120
Epoch 1/1
1200/1200 [==============================] - 407s 339ms/step - loss: 2.8479
Epoch 1/1
1200/1200 [==============================] - 409s 341ms/step - loss: 2.7965
Epoch 1/1
1200/1200 [==============================] - 412s 344ms/step - loss: 2.7486
Epoch 1/1
1200/1200 [==============================] - 414s 345ms/step - loss: 2.7097
Epoch 1/1
1200/1200 [==============================] - 418s 348ms/step - loss: 2.6741
Epoch 1/1
1200/1200 [==============================] - 433s 361ms/step - loss: 2.6434
Epoch 1/1
1200/1200 [==============================] - 434s 362ms/step - loss: 2.6134
Epoch 1/1
1200/1200 [==============================] - 422s 351ms/step - loss: 2.5893
```

Model trained for 15 epoch on Inception model



```
Epoch 1/1
1200/1200 [==============================] - 468s 390ms/step - loss: 4.4812
Epoch 1/1
1200/1200 [==============================] - 459s 383ms/step - loss: 3.8404
Epoch 1/1
1200/1200 [==============================] - 443s 369ms/step - loss: 3.6145
Epoch 1/1
1200/1200 [==============================] - 445s 371ms/step - loss: 3.4760
Epoch 1/1
1200/1200 [==============================] - 439s 366ms/step - loss: 3.3735
Epoch 1/1
1200/1200 [==============================] - 438s 365ms/step - loss: 3.2916
Epoch 1/1
1200/1200 [==============================] - 457s 381ms/step - loss: 3.2240
Epoch 1/1
1200/1200 [==============================] - 433s 361ms/step - loss: 3.1648
Epoch 1/1
1200/1200 [==============================] - 437s 364ms/step - loss: 3.1116
Epoch 1/1
1200/1200 [==============================] - 449s 374ms/step - loss: 3.0656
Epoch 1/1
1200/1200 [==============================] - 443s 369ms/step - loss: 3.0215
Epoch 1/1
1200/1200 [==============================] - 450s 375ms/step - loss: 2.9865
Epoch 1/1
1200/1200 [==============================] - 450s 375ms/step - loss: 2.9528
Epoch 1/1
1200/1200 [==============================] - 456s 380ms/step - loss: 2.9219
Epoch 1/1
1200/1200 [==============================] - 454s 379ms/step - loss: 2.8941
```

Model trained for 15 epoch on Resnet50 model

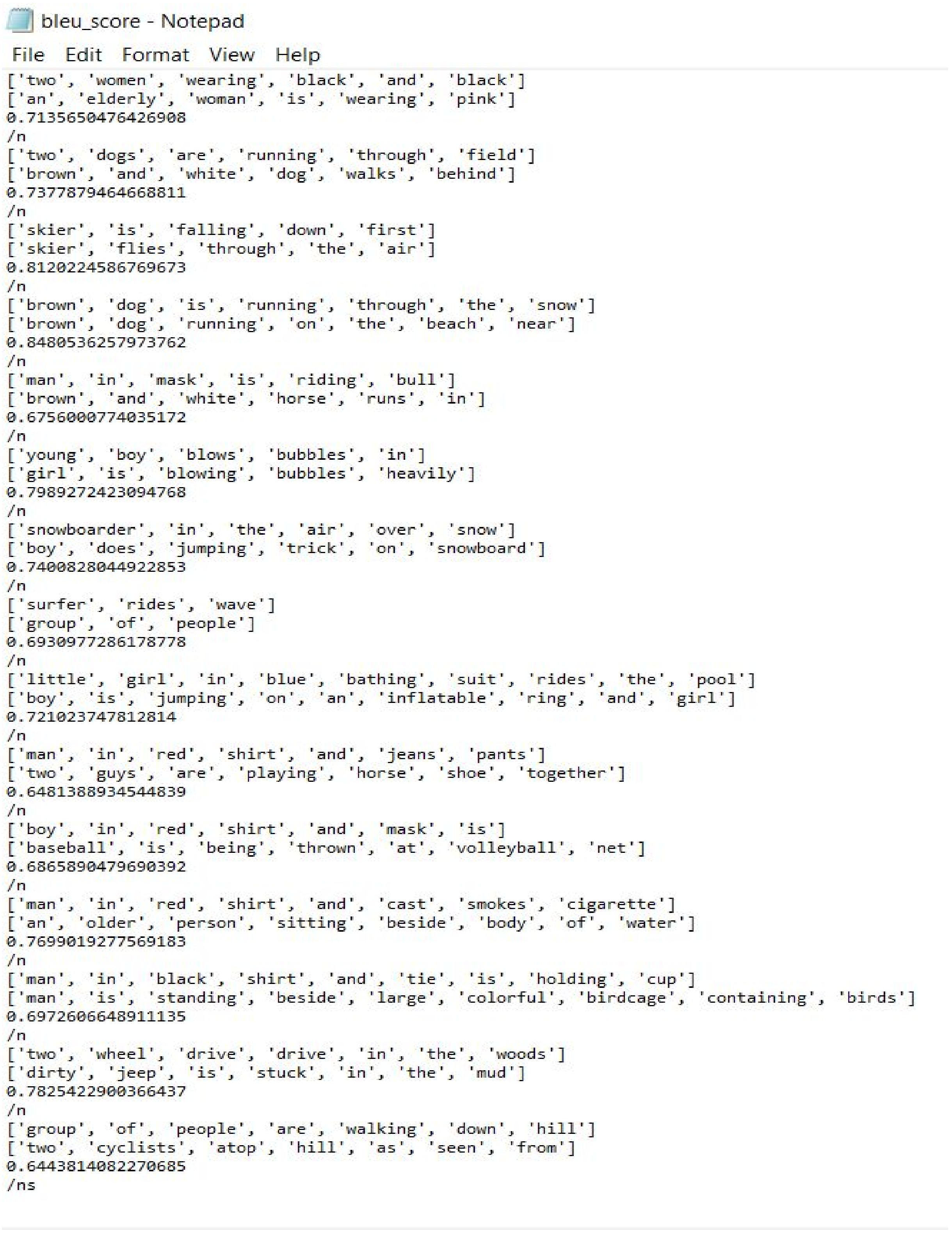


Comparison for 15 epoch of Loss function between Inception and Resnet50

The Blue line indicates loss for Resnet
The Red line indicates loss for Inception

4.2 EVALUATION METRIC

4.2.1 <u>**BLEU SCORE:**</u> It stands for "Bilingual Evaluation Understudy **Score". "The BLEU score was proposed by Kishore Papineni, et al. in their 2002 paper". "BLEU: a Method for Automatic Evaluation of Machine Translation**". It is a metric for evaluating a generated sentence to a reference sentence. For a perfect match results in a score of 1.0, whereas a perfect mismatch results in a score of 0.0."

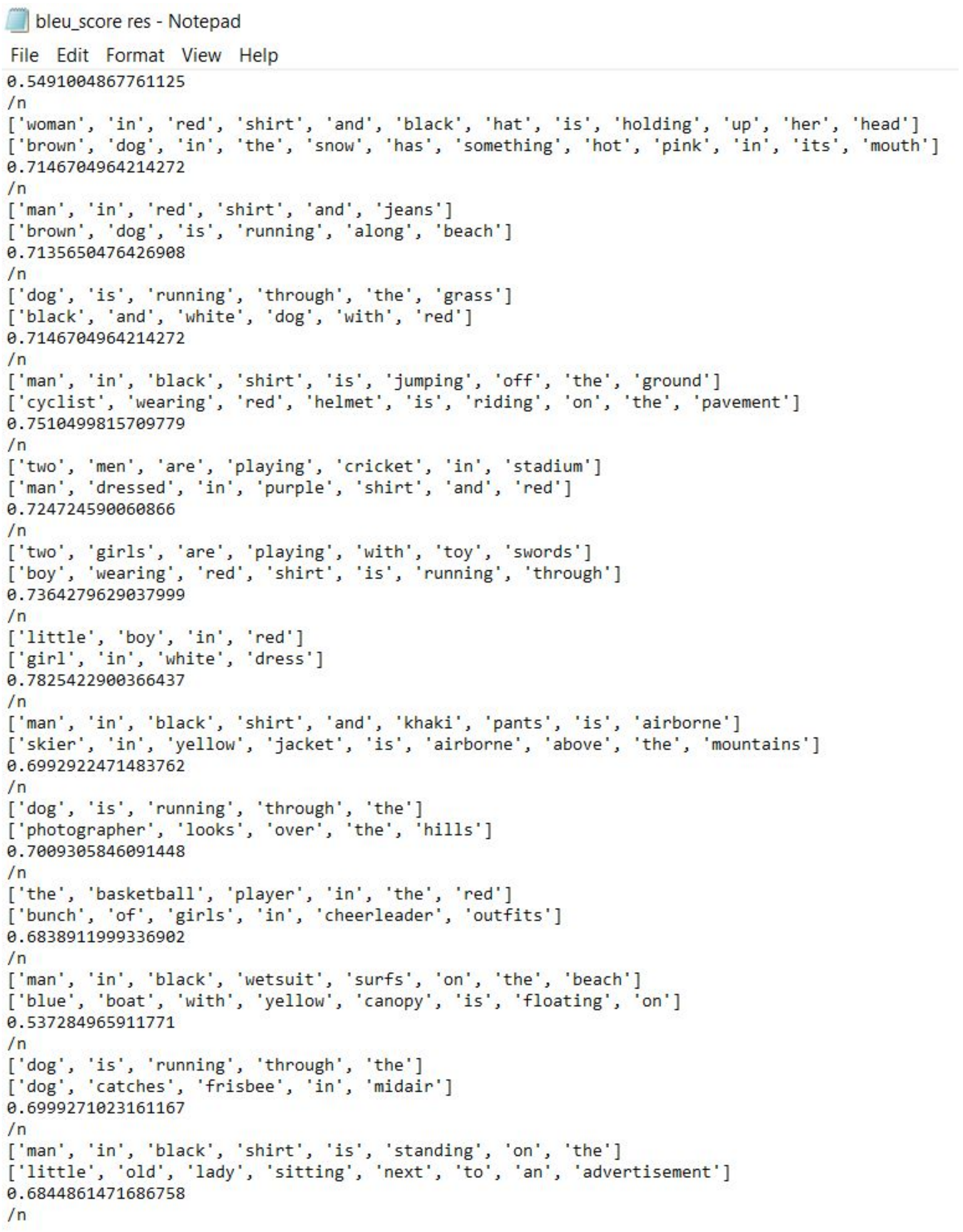


bleu_score - Notepad

File Edit Format View Help

```
['two', 'women', 'wearing', 'black', 'and', 'black']
['an', 'elderly', 'woman', 'is', 'wearing', 'pink']
0.7135650476426908
/n
['two', 'dogs', 'are', 'running', 'through', 'field']
['brown', 'and', 'white', 'dog', 'walks', 'behind']
0.7377879464668811
/n
['skier', 'is', 'falling', 'down', 'first']
['skier', 'flies', 'through', 'the', 'air']
0.8120224586769673
/n
['brown', 'dog', 'is', 'running', 'through', 'the', 'snow']
['brown', 'dog', 'running', 'on', 'the', 'beach', 'near']
0.8480536257973762
/n
['man', 'in', 'mask', 'is', 'riding', 'bull']
['brown', 'and', 'white', 'horse', 'runs', 'in']
0.6756000774035172
/n
['young', 'boy', 'blows', 'bubbles', 'in']
['girl', 'is', 'blowing', 'bubbles', 'heavily']
0.7989272423094768
/n
['snowboarder', 'in', 'the', 'air', 'over', 'snow']
['boy', 'does', 'jumping', 'trick', 'on', 'snowboard']
0.7400828044922853
/n
['surfer', 'rides', 'wave']
['group', 'of', 'people']
0.6930977286178778
/n
['little', 'girl', 'in', 'blue', 'bathing', 'suit', 'rides', 'the', 'pool']
['boy', 'is', 'jumping', 'on', 'an', 'inflatable', 'ring', 'and', 'girl']
0.721023747812814
/n
['man', 'in', 'red', 'shirt', 'and', 'jeans', 'pants']
['two', 'guys', 'are', 'playing', 'horse', 'shoe', 'together']
0.6481388934544839
/n
['boy', 'in', 'red', 'shirt', 'and', 'mask', 'is']
['baseball', 'is', 'being', 'thrown', 'at', 'volleyball', 'net']
0.6865890479690392
/n
['man', 'in', 'red', 'shirt', 'and', 'cast', 'smokes', 'cigarette']
['an', 'older', 'person', 'sitting', 'beside', 'body', 'of', 'water']
0.7699019277569183
/n
['man', 'in', 'black', 'shirt', 'and', 'tie', 'is', 'holding', 'cup']
['man', 'is', 'standing', 'beside', 'large', 'colorful', 'birdcage', 'containing', 'birds']
0.6972606648911135
/n
['two', 'wheel', 'drive', 'drive', 'in', 'the', 'woods']
['dirty', 'jeep', 'is', 'stuck', 'in', 'the', 'mud']
0.7825422900366437
/n
['group', 'of', 'people', 'are', 'walking', 'down', 'hill']
['two', 'cyclists', 'atop', 'hill', 'as', 'seen', 'from']
0.6443814082270685
/ns
```

BLEU-score for Inception model



```
0.5491004867761125
/n
['woman', 'in', 'red', 'shirt', 'and', 'black', 'hat', 'is', 'holding', 'up', 'her', 'head']
['brown', 'dog', 'in', 'the', 'snow', 'has', 'something', 'hot', 'pink', 'in', 'its', 'mouth']
0.7146704964214272
/n
['man', 'in', 'red', 'shirt', 'and', 'jeans']
['brown', 'dog', 'is', 'running', 'along', 'beach']
0.7135650476426908
/n
['dog', 'is', 'running', 'through', 'the', 'grass']
['black', 'and', 'white', 'dog', 'with', 'red']
0.7146704964214272
/n
['man', 'in', 'black', 'shirt', 'is', 'jumping', 'off', 'the', 'ground']
['cyclist', 'wearing', 'red', 'helmet', 'is', 'riding', 'on', 'the', 'pavement']
0.7510499815709779
/n
['two', 'men', 'are', 'playing', 'cricket', 'in', 'stadium']
['man', 'dressed', 'in', 'purple', 'shirt', 'and', 'red']
0.724724590060866
/n
['two', 'girls', 'are', 'playing', 'with', 'toy', 'swords']
['boy', 'wearing', 'red', 'shirt', 'is', 'running', 'through']
0.7364279629037999
/n
['little', 'boy', 'in', 'red']
['girl', 'in', 'white', 'dress']
0.7825422900366437
/n
['man', 'in', 'black', 'shirt', 'and', 'khaki', 'pants', 'is', 'airborne']
['skier', 'in', 'yellow', 'jacket', 'is', 'airborne', 'above', 'the', 'mountains']
0.6992922471483762
/n
['dog', 'is', 'running', 'through', 'the']
['photographer', 'looks', 'over', 'the', 'hills']
0.7009305846091448
/n
['the', 'basketball', 'player', 'in', 'the', 'red']
['bunch', 'of', 'girls', 'in', 'cheerleader', 'outfits']
0.6838911999336902
/n
['man', 'in', 'black', 'wetsuit', 'surfs', 'on', 'the', 'beach']
['blue', 'boat', 'with', 'yellow', 'canopy', 'is', 'floating', 'on']
0.537284965911771
/n
['dog', 'is', 'running', 'through', 'the']
['dog', 'catches', 'frisbee', 'in', 'midair']
0.6999271023161167
/n
['man', 'in', 'black', 'shirt', 'is', 'standing', 'on', 'the']
['little', 'old', 'lady', 'sitting', 'next', 'to', 'an', 'advertisement']
0.6844861471686758
/n
```

BLEU-score for Resnet50 model

Average bleu-score for the Inception model turns out to be 0.731262.

Average bleu-score for the Resnet50 model turns out to be 0.69103.

Chapter 5

CONCLUSION

1. We have used a Transfer learning algorithm, with image processing techniques.
2. The method has low computational cost and is suitable for detecting
features with reasonable accuracy and is suitable for image pre
processing.
4. Supervised methods (including Recurrent neural networks) require quality train
ing data to build a model that can perform well on testing data. Partial captions are used for training
5. The model is capable of intelligently determining the Maximum likelihood
corresponding to a word.

FUTURE SCOPE

Hence we Conclude that this Model can be useful in the following fields :-

1. **Help the Visually Impaired :** We can embed this technology in a headband or glasses, which a visually impaired person can wear, a camera is connected to the headband or glasses. That camera can send the video stream of the surroundings environment to the model and then the model can predict the obstacles in the surroundings . This prediction can be converted to audio using TTS and can be heard by the person.

2. **Self Driving Cars :** All the companies who are building the self driving cars are using image processing and video processing with neural networks to attain their goal.
3. **Skin Cancer Prediction :** In the medical field this technology can be used in the prediction of skin cancer like classifying the tumor as Benign or Malignant.
4. **Classifying images in our Phone:** This technology can also be used for the classification of images in our Gallery for example classify in different categories like mountains, beach, portraits etc.

Chapter 6

References

1. Show and Tell : - 'A Neural Image Caption Generator' [arXiv : 1411.4555]
2. A Comprehensive Survey of Deep Learning for Image Captioning' [arXiv : 1810.04020]
3. Shadab Hussain, Flickr8K dataset kaggle.
4. Fundamentals of RNN and Long Short Term Memory networks
5. Faizan Shaikh, Automatic Image Captioning using Deep Learning (CNN and Long Short Term Memory networks) in Pytorch, Analytics Vidhya.
6. Harshall Lamba, Image Captioning with Keras, Towards Data Science.