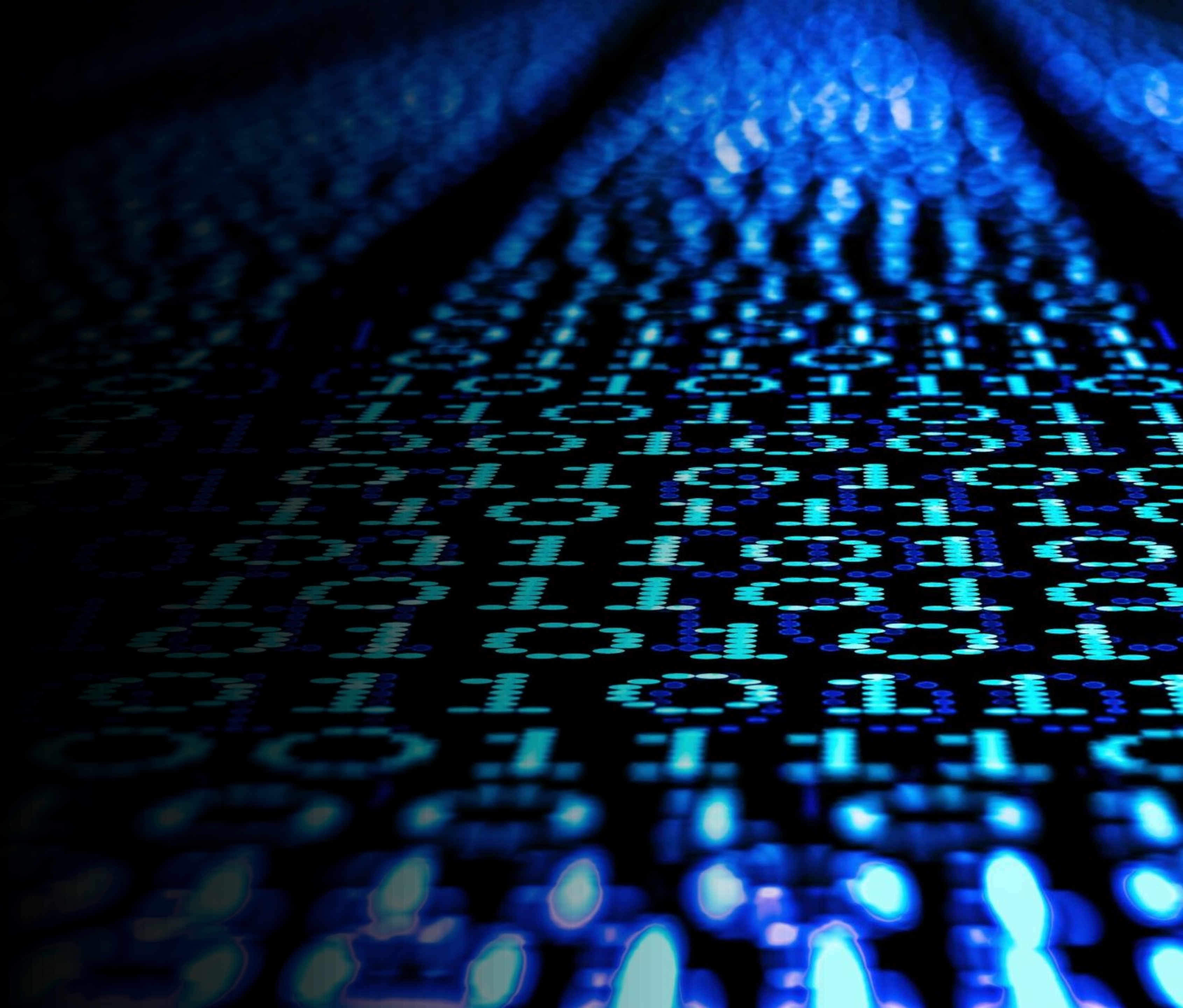
Project Exhibition-1
Group-54

AI_4_GOOD



Guided by:-

Dr. Komarasamy G

Team Members :-

- Ayush Vaidya
- Nikhil Chaurasiya
- Vivek Agrawal

Content

- Project title, guide & team members
- Introduction
- Existing work with limitations
- Proposed work and methodology
- Novelty of the project
- Real time usage
- Hardware & software requirements
- Overall system architecture diagram.
- Literature review
- Module description
- Module work flow explanation.
- Implementation and coding.
- Demo video.
- Snap shot of your project.
- Testing.
- Result and Discussion. (Input and output)
- Conclusion.

References

1. Introduction to image captioning

- Caption generation is a challenging artificial intelligence problem where a textual description must be generated for a given photograph.
- The objective of our project is to learn the concepts of a CNN and LSTM model and build a working model of Image caption generator by implementing CNN with LSTM.
- The task of image captioning can be divided into two modules logically .
  - Image based model
  - Language based model

A guy is holding another guy

The goal of image captioning is to convert a given input image into a natural language description.

2. Existing work with limitation

- There is a caption bot in google cloud program, which gives us services for image captioning and description of the image.
- There is a caption bot of Microsoft Azure cloud program.
- Nvidia is also working on glasses which will give vision to the visually impaired people.

Limitations:-

- There is only one large data set MS-COCO data set, which contains about 1,80,000 images and 9,00,000 image captions, which is still less to produce such glasses to give accurate predictions for what the glasses are seeing through the camera

3. Proposed work and Methodology

Proposed Work

- Caption generation is a challenging artificial intelligence problem where a textual description must be generated for a given photograph.
- Deep learning methods have demonstrated state-of-the-art results on caption generation problems.
- What is most impressive about these methods is a single end-to-end model can be defined to predict a caption, given a photo, instead of requiring sophisticated data preparation or a pipeline of specifically designed models.

Methodology

- We will be using the concept of CNN and LSTM and build a model of Image Caption Generator which involves the concept of computer vision and Natural Language Process to recognize the context of images and describe them in natural language like English.

Here, we will break down the module into following sections for better understanding:-

- Preprocessing of Image
- Creating the vocabulary for the image
- Train the set
- Evaluating the model
- Testing on individual images

4. Novelty of the project

The most beautiful part of this project is that it uses all concepts of AI i.e.

- Deep learning
- Machine Learning
- Natural Language Processing

at one place which covers all topics in AI and it uses module Tensorflow which is provided by Google.

- The other best thing is its use, that is so helpful to visually impaired people that it can let them to know what they are seeing.

5. Real time usage

- Aid to the Blind
- Google Image Search
- Self Driven cars
- CCTV Camera Alarm

6. Hardware and Software requirement

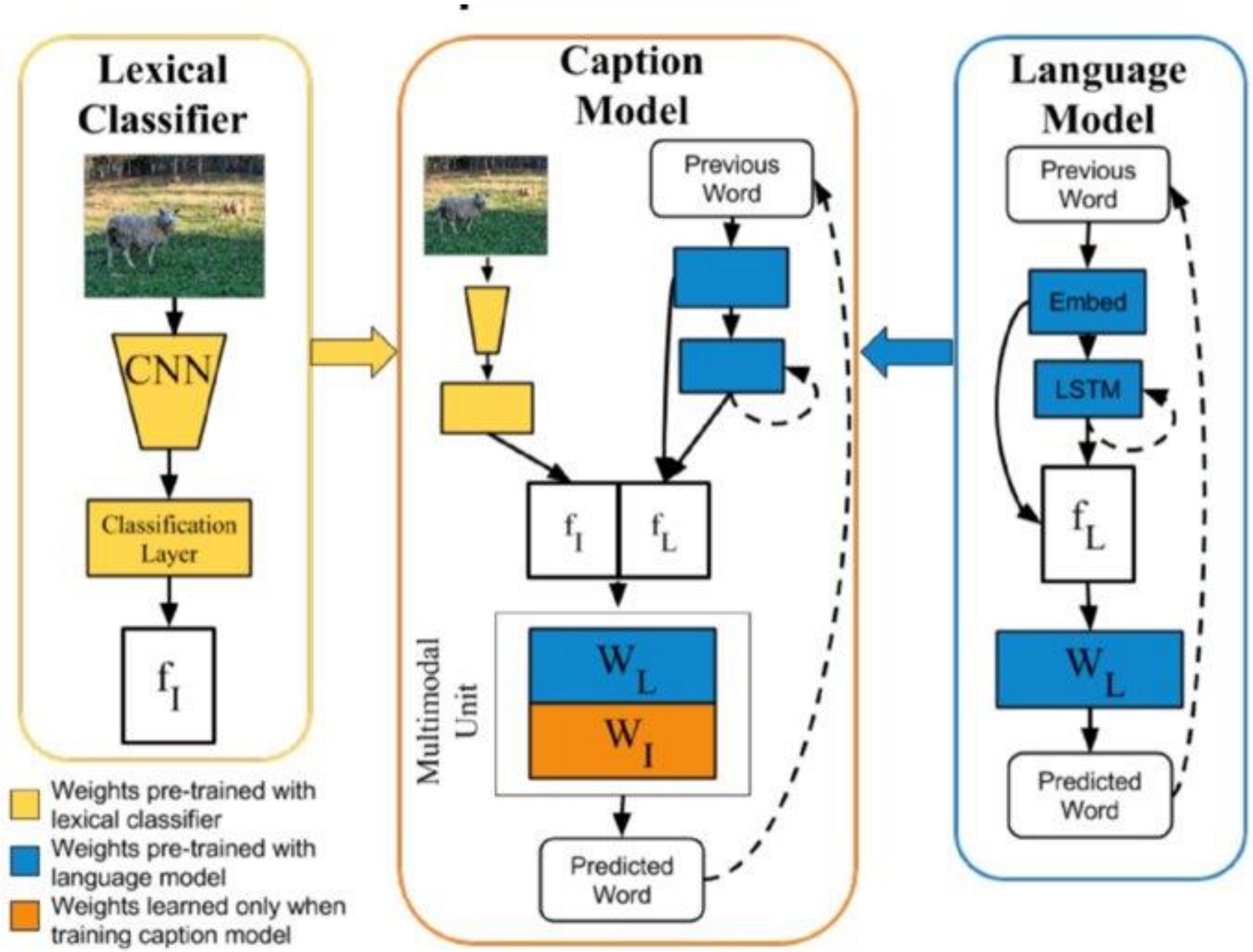
Hardware

- Camera
- Hard disk(SSD Preferred)
- Atleast 16 GB RAM
- Multicore GPU
- min. processor i7

Software

- Python and Conda virtual environment
- Deep Learning concepts like
  - Multi-layered Perceptrons,
  - Convolution Neural Networks,
  - Recurrent Neural Networks,
  - Transfer Learning,
  - Gradient Descent
  - Backpropagation
  - Overfitting
  - Probability
  - Text Processing
    - Keras library
    - LSTM
    - NLP

7. System Architecture Design



8. Literature review

- We have written data pre-processing scripts to process raw input data (both images and captions) into proper format; A pre-trained Convolutional Neural Network architecture as an encoder to extract and encode image features into a higher dimensional vector space; An LSTM-based Recurrent Neural Network as a decoder to convert encoded features to natural language descriptions; Attention mechanism which allows the decoder to see features from a specifically highlighted region of the input image to improve the overall performance; Beam Search to figure out a caption with the highest likelihood. Each individual component of our generator pipeline will be discussed in detail in below slides

9. Complete Module Split-Up...

Image Captioning

Generating Captions for Images

Steps

- Data collection
- Understanding the data
- Data Cleaning
- Loading the training set
- Data Preprocessing — Images
- Data Preprocessing — Captions
- Data Preparation using Generator Function
- Word Embeddings
- Model Architecture
- Inference

10. Every Module's Explanation

Data collection:

- Data collection is defined as the procedure of collecting, measuring and analyzing accurate insights for research using standard validated techniques.

Understanding the data

- here we will be reading our files i.e. our training images and test images with their captions which are stored in txt file. each image is having five captions with different ids.
- for eg:
- 1000268201_693b08cb0e jpg#0 A child in a pink dress is climbing up a set of stairs in an entry way
- 1000268201_693b08cb0e jpg#1 A girl going into a wooden building
- 1000268201_693b08cb0e jpg#2 A little girl climbing into a wooden playhouse
- 1000268201_693b08cb0e jpg#3 A little girl climbing the stairs to her playhouse
- 1000268201_693b08cboe jpg#4 A little girl in a pink dress going into a wooden cabin

here we see this image id has five id's numbering from 0 to 4. and different caption for same image.

Every Module's Explanation

Data Cleaning

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled.

- so to remove this anmalies in data, we have cleaned our captions as some were having null values ans we have extracted caption id and captions
- we also have removed the determiners like a the an, because they are used almost in all sentences and so we do not have count of unique words, get that we have cleaned our sentences
- create vocabulary for this model
- filter words from the vocab according to certain threshold frequency
- get the total word count and unique words count
- also check which words were used most number of times, getting the frequency count of each unique words

this was all we have done in this cleaning part of our data

preparing our training dataset

- Here we had prepared descriptions for training data
- tweak - add < s > and < e > token to our training data, added this START and END markers so as to get the model understand where exactly the sentence starts and where to stop.
- VECTORIZING THE TEXT DATA:
- We'll use the TextVectorization layer to vectorize the text data, that is to say, to turn the original strings into integer sequences where e integer represents the index of a word in a vocabulary. We will use a custom string standardization scheme (strip punctuation characte except < and >) and the default splitting scheme (split on whitespace).

Every Module's Explanation

image preprocessing - images

here first of all we will be handling with extracting features from the images

- so to meet this requirement we will be using keras module ResNet50 in which we have ImageNet model which is already pre trained to meet this requirement
- NORMALIZATION
- in this preprocessing of images we will be making each image of <224x224> pixel
- then converted image to array and stored to use it with numpy
- FOR TRAIN IMAGES: image_id -->feature_vector extracted from Resnet Image
- FOR TEST IMAGES: image_id -->feature_vector extracted from Resnet Image

image preprocessing - captions

- Building a tf.data.Dataset pipeline for training
- We will generate pairs of images and corresponding captions using a tf.data.Dataset object. The pipeline consists of two steps:
- mapped each word which we counted in total words with each number starting 0 and then to end
- we have added two more words < s > ans < e >, which we used to mark the start and end of sentence, so mapped this too
- Read the image from the disk
- Tokenize all the five captions corresponding to the image

Every Module's Explanation

- Our image captioning architecture consists of three models:
- **CNN:** used to extract the image features
- **TransformerEncoder:** The extracted image features are then passed to a Transformer based encoder that generates a new representation of the inputs
- **TransformerDecoder:** This model takes the encoder output and the text data (sequences) as inputs and tries to learn to generate the caption.

```
# Model architechture
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
input_4 (InputLayer)             (None, 35)            0
____________________________________________________________________________________________________
input_3 (InputLayer)             (None, 2048)          0
____________________________________________________________________________________________________
embedding_1 (Embedding)          (None, 35, 50)        92400       input_4[0][0]
____________________________________________________________________________________________________
dropout_1 (Dropout)              (None, 2048)          0           input_3[0][0]
____________________________________________________________________________________________________
dropout_2 (Dropout)              (None, 35, 50)        0           embedding_1[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 256)           524544      dropout_1[0][0]
____________________________________________________________________________________________________
lstm_1 (LSTM)                    (None, 256)           314368      dropout_2[0][0]
____________________________________________________________________________________________________
add_33 (Add)                     (None, 256)           0           dense_1[0][0]
                                                                   lstm_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 256)           65792       add_33[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 1848)          474936      dense_2[0][0]
====================================================================================================
Total params: 1,472,040
Trainable params: 1,472,040
Non-trainable params: 0
```

11. Implementation Screenshots

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow.keras
import re
import nltk
from nltk.corpus import stopwords
import string
import json
from time import time
import pickle
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input, decode_predic
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.preprocessing. sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import Input, Dense, Dropout, Embedding, LSTM
from tensorflow.keras.layers import add
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.sequence import pad_sequences
import tensorflow.keras.layers
```

[1]

Python

Implementation Screenshots

```
descriptions['1000268201_693b08cb0e']
```
[7]

```
['A child in a pink dress is climbing up a set of stairs in an entry way .',
 'A girl going into a wooden building .',
 'A little girl climbing into a wooden playhouse .',
 'A little girl climbing the stairs to her playhouse .',
 'A little girl in a pink dress going into a wooden cabin .']
```

```
IMG_PATH="/Users/nikhil/Downloads/Flicker8k_Dataset/"
import cv2
import matplotlib.pyplot as plt
```
[8]

```
img= cv2.imread(IMG_PATH+"1000268201_693b08cb0e.jpg")
img=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(img)
plt.axis("off")
plt.show()
```
[9]

Implementation Screenshots

```
#prepare descriptions for training data
# tweak - add <s> and <e> token to our training data

train_descriptions={}

for img_id in train:
    train_descriptions[img_id]=[]
    for cap in descriptions[img_id]:
        cap_to_append="startseq "+ cap+" endseq"
        train_descriptions[img_id].append(cap_to_append)
```

[28]

```
train_descriptions['1000268201_693b08cb0e']
```

[29]

```
['startseq child in pink dress is climbing up set of stairs in an entry way endseq',
 'startseq girl going into wooden building endseq',
 'startseq little girl climbing into wooden playhouse endseq',
 'startseq little girl climbing the stairs to her playhouse endseq',
 'startseq little girl in pink dress going into wooden cabin endseq']
```

Implementation Screenshots

```python
def preprocess_img(img):
        img = image.load_img(img, target_size=(224,224))
        img = image.img_to_array(img)
        img = np.expand_dims(img,axis=0)
        #normalization
        img = preprocess_input(img)
        return img
```

[32] Python

```python
# normalization
img = preprocess_img(IMG_PATH+"1000268201_693b08cb0e.jpg")
plt.imshow(img[0])
plt.axis('off')
plt.show()
print(img)
```

[33] Python

```
... Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

Implementation Screenshots

Data loader (Generator)

```python
def data_generator(train_descriptions, encoding_train, word_2_idx, max_len,batch_size):
    X1,X2,y=[],[],[]
    n=0
    while True:
        for key, desc_list in train_descriptions.items():
            n=n+1

            photo=encoding_train[key+".jpg"]
            for desc in desc_list:
                seq= [word_2_idx[word] for word in desc.split() if word in word_2_idx]
                for i in range(1, len(seq)):
                    xi=seq[0:i]
                    yi=seq[i]

                    # 0 denotes padding
                    xi = pad_sequences([xi], maxlen=max_len, value=0, padding='post')[0]
                    yi= to_categorical([yi], num_classes==vocab_size)[0]
                    X1.append(photo)
                    X2.append(xi)
                    y.append(yi)

                if n==batch_size:
                    yield[[np.array(X1), np.array(X2)],np.array(y)]

                X1,X2,y= [],[],[]
                n=0
```

Python

Implementation Screenshots

Word Embeddings

```
f = open("/Users/nikhil/downloads/glove.6B.50d.txt",encoding='utf8')
```
[46] ✓ 0.1s

```
embedding_index = {}

for line in f:
    values = line.split()

    word = values[0]
    word_embedding = np.array(values[1:],dtype='float')
    embedding_index[word] = word_embedding
```
[47] ✓ 2.7s

```
f.close()
```
[48] ✓ 0.1s

```
embedding_index['apple']
```
[49] ✓ 0.1s

```
array([ 0.52042 , -0.8314  ,  0.49961 ,  1.2893  ,  0.1151  ,  0.057521,
       -1.3753  , -0.97313 ,  0.18346 ,  0.47672 , -0.15112 ,  0.35532 ,
        0.25912 , -0.77857 ,  0.52181 ,  0.47695 , -1.4251  ,  0.858   ,
        0.59821 , -1.0903  ,  0.33574 , -0.60891 ,  0.41742 ,  0.21569 ,
       -0.07417 , -0.5822  , -0.4502  ,  0.17253 ,  0.16448 , -0.38413 ,
        2.3283  , -0.66682 , -0.58181 ,  0.74389 ,  0.095015, -0.47865 ,
       -0.84591 ,  0.38704 ,  0.23693 , -1.5523  ,  0.64802 , -0.16521 ,
       -1.4719  , -0.16224 ,  0.79857 ,  0.97391 ,  0.40027 , -0.21912 ,
       -0.30938 ,  0.26581 ])
```

Implementation Screenshots

Model Architecture

```
input_img_features = Input(shape=(2048,))
inp_img1 = Dropout(0.3)(input_img_features)
inp_img2 = Dense(256,activation='relu')(inp_img1)
```
[53] ✓ 0.2s

```
print(np.__version__)
```
[54] ✓ 0.1s

```
... 1.19.5
```

```
# 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 = LSTM(256)(inp_cap2)
```
[55] ✓ 0.4s

```
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)
```
[56] ✓ 0.2s

```
model.summary()
```

Implementation Screenshots

Predictions

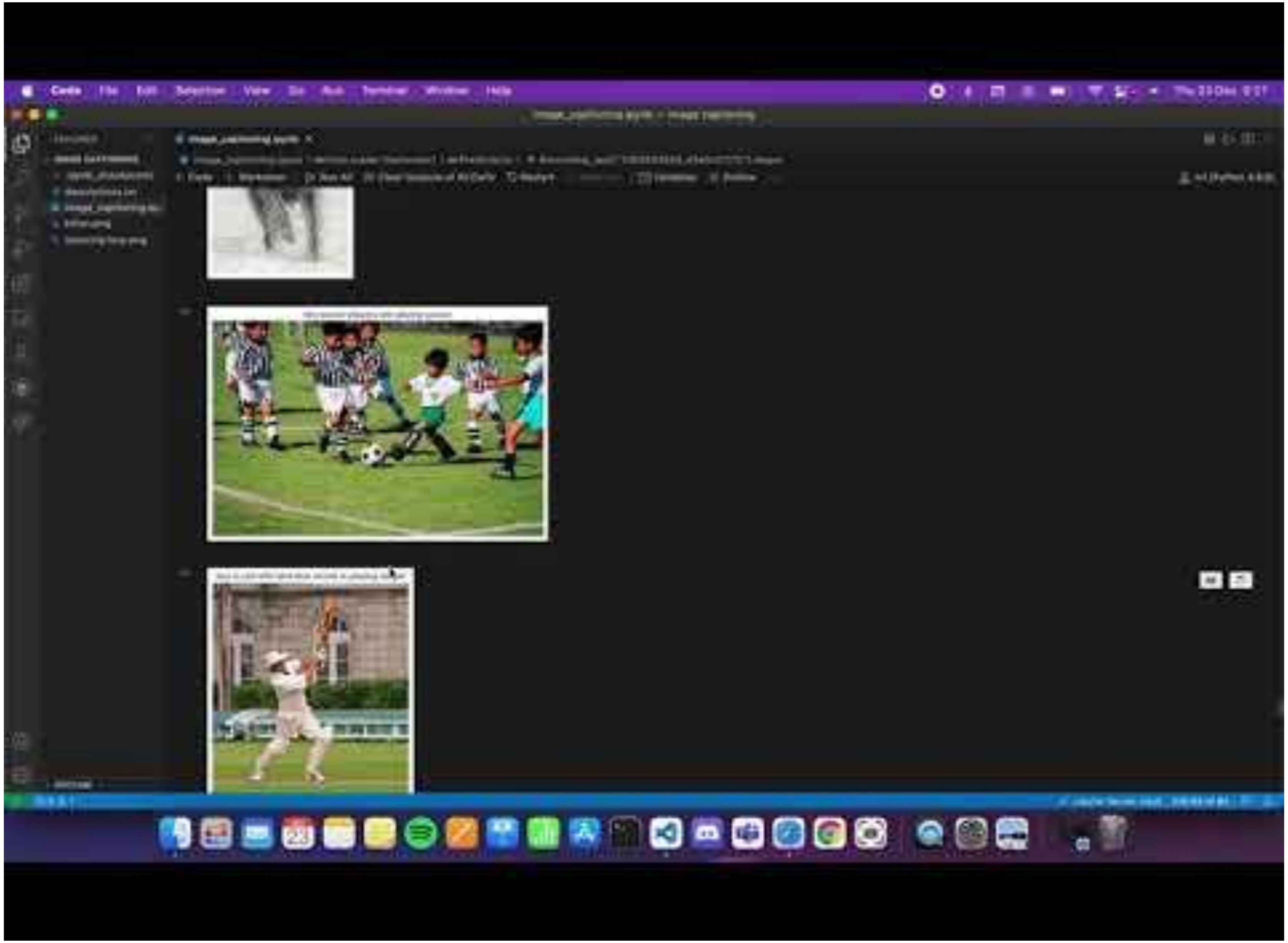
```python
def predict_caption(photo):

    in_text = "startseq"
    for i in range(max_len):
        sequence = [word_to_idx[w] for w in in_text.split() if w in word_to_idx]
        sequence = pad_sequences([sequence],maxlen=max_len,padding='post')

        ypred = model.predict([photo,sequence])
        ypred = ypred.argmax() #WOrd with max prob always - Greedy Sampling
        word = idx_to_word[ypred]
        in_text += (' ' + word)

        if word == "endseq":
            break

    final_caption = in_text.split()[1:-1]
    final_caption = ' '.join(final_caption)
    return final_caption
```

[63] ✓ 0.1s

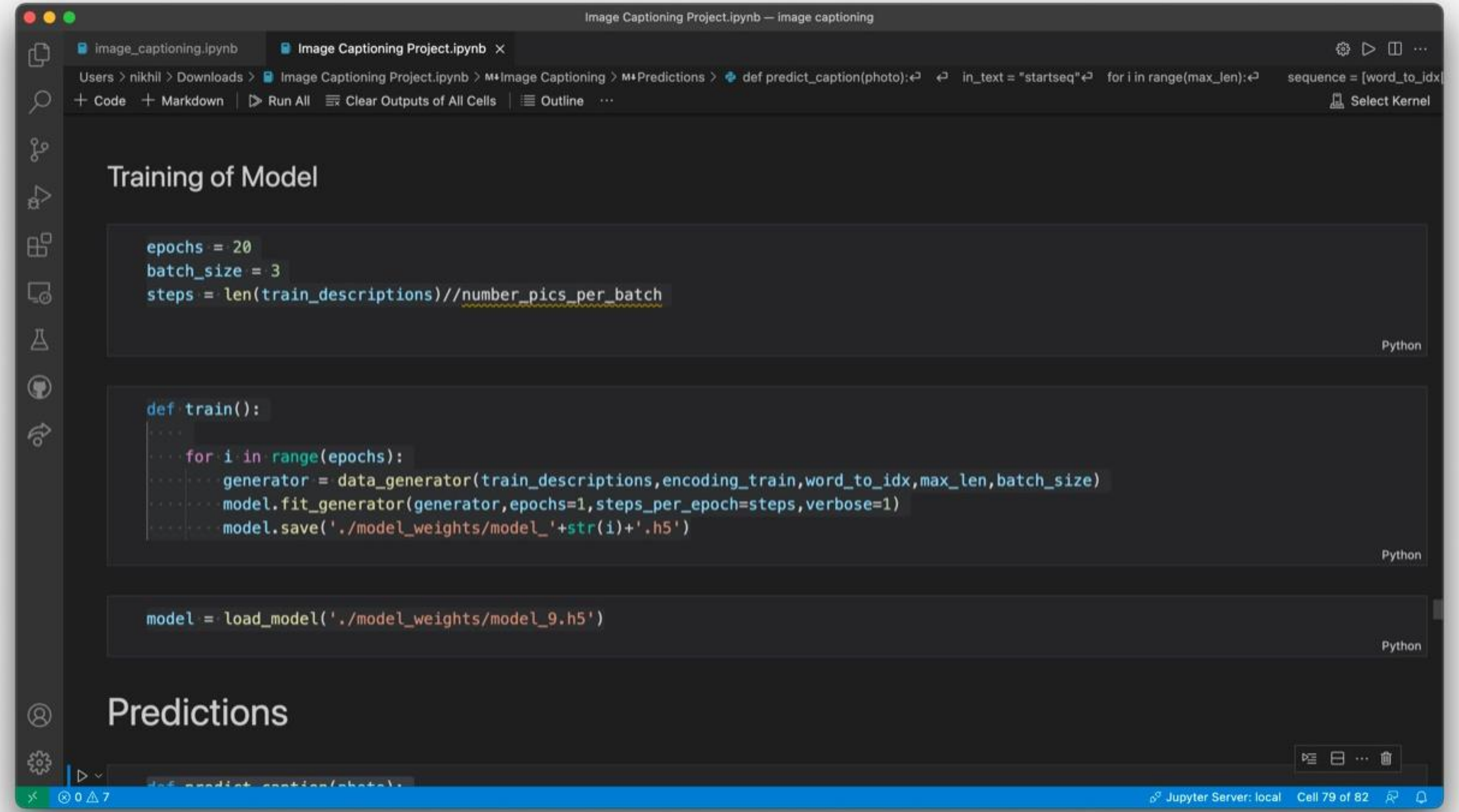
```python
# Pick Some Random Images and See Results
plt.style.use("seaborn")
for i in range(15):
    idx = np.random.randint(0,1000)
    all_img_names = list(encoding_test.keys())
    img_name = all_img_names[idx]
    photo_2048 = encoding_test[img_name].reshape((1,2048))

    i = plt.imread("/Users/nikhil/Downloads/Flicker8k_Dataset/"+img_name+".jpg")

    caption = predict_caption(photo_2048)
    #print(caption)

    plt.title(caption)
    plt.imshow(i)
    plt.axis("off")
    plt.show()
```

[67] ✓ 4.6s

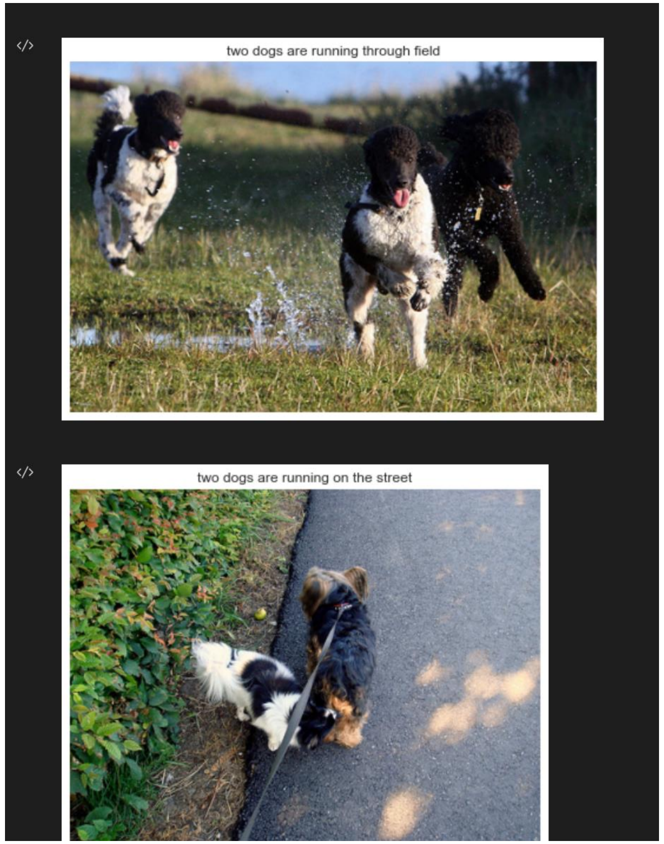
12. Demo Video

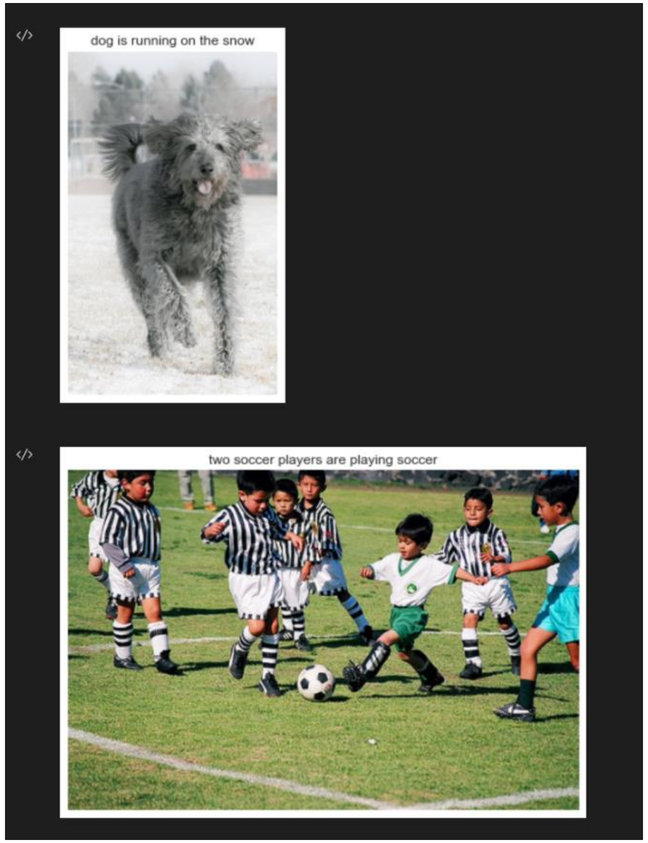


13. Testing of the project

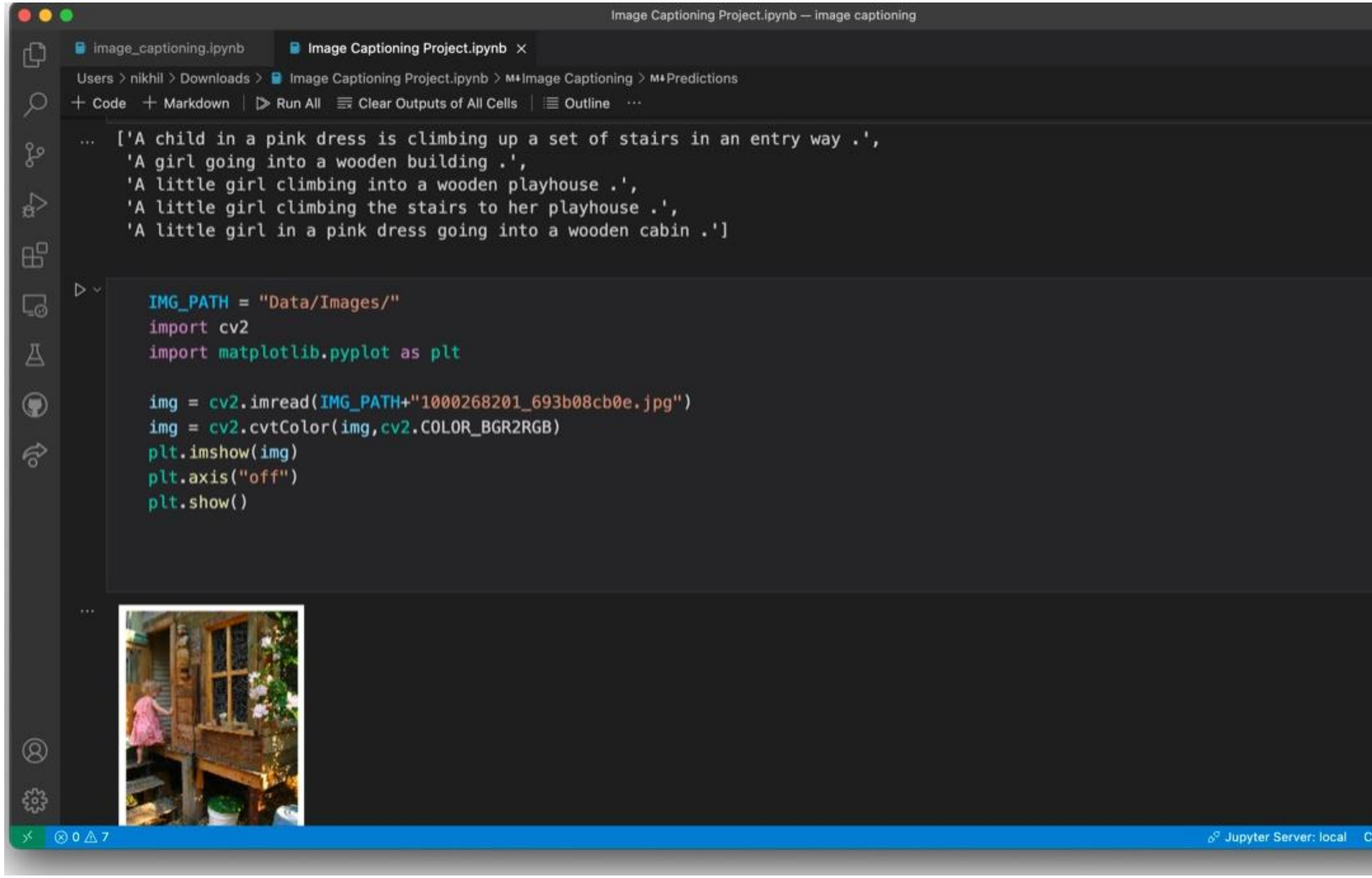


14. Snapshot of the project



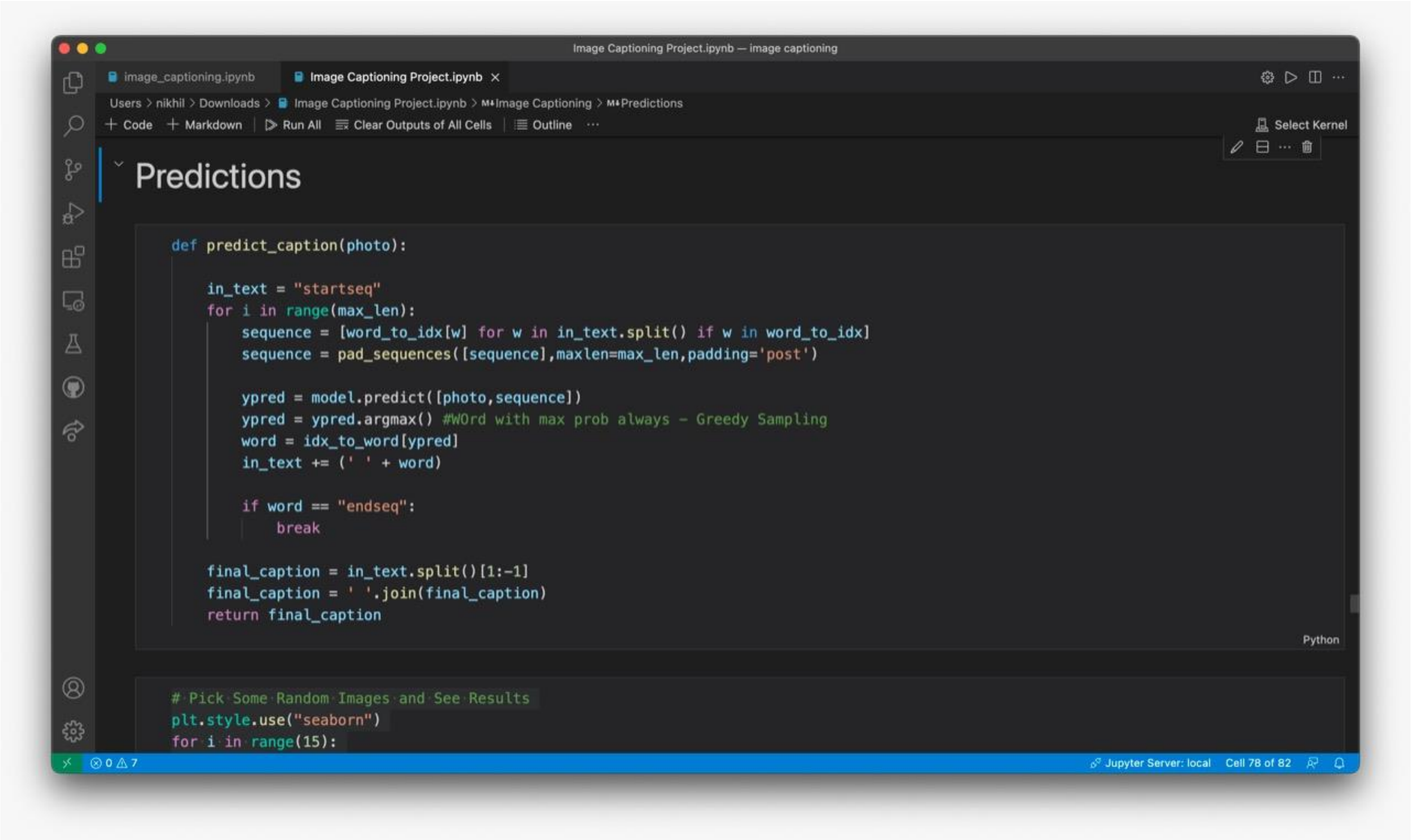


15. Results and Discussion

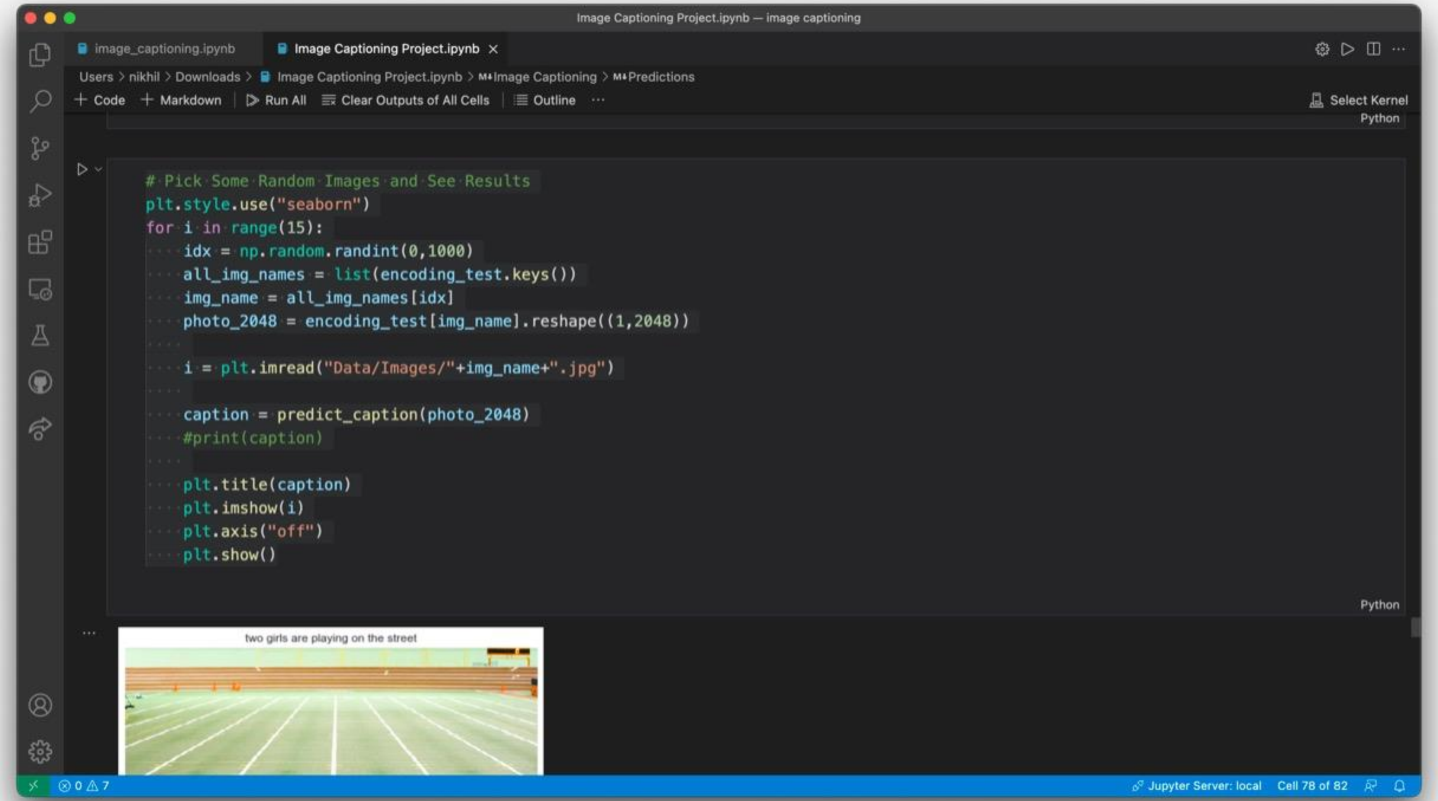


Input

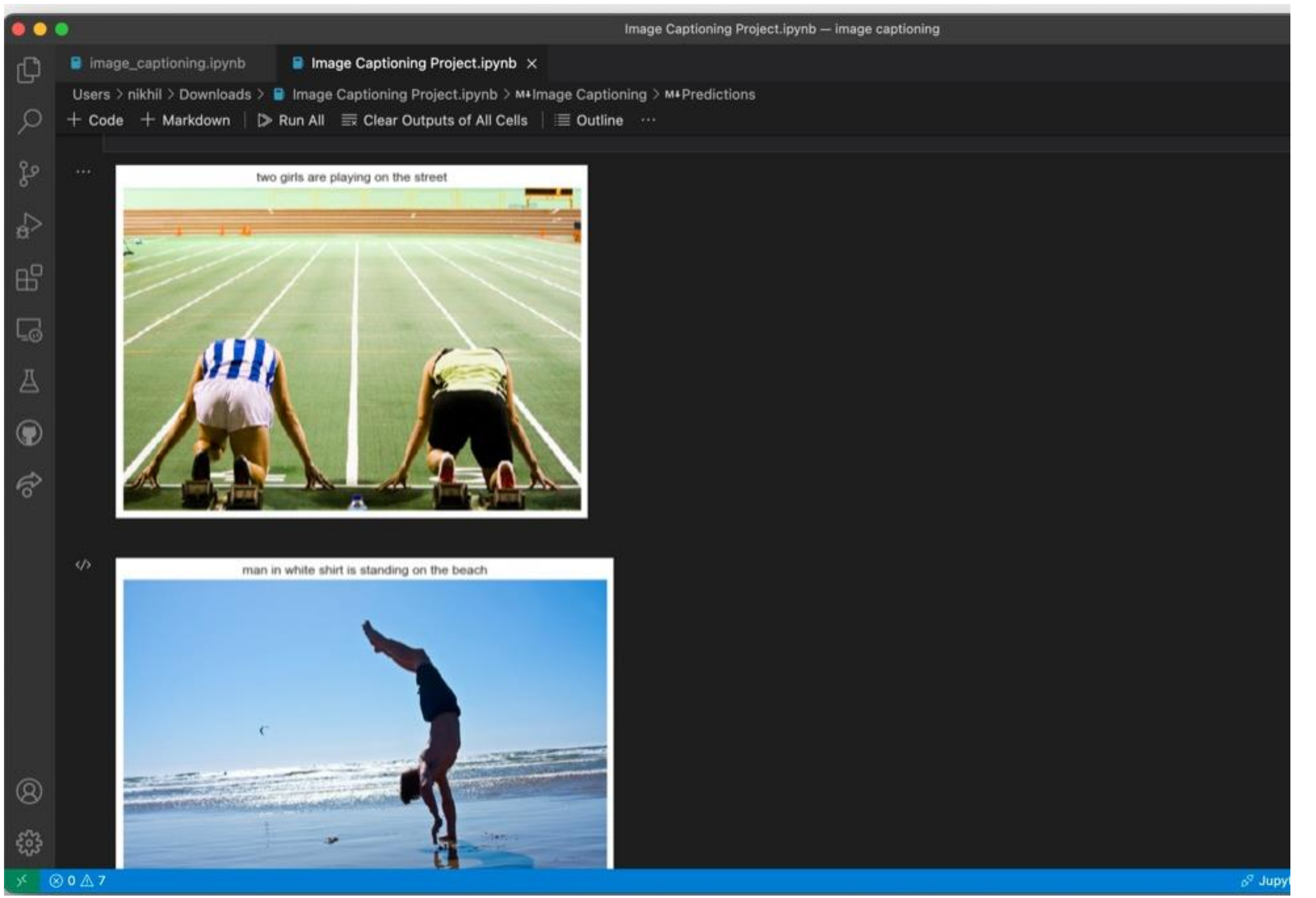
Results and Discussion



Results and Discussion



Results and Discussion



Output

16. CONCLUSION

In this advanced Python project, we have implemented a CNN-RNN model by building an image caption generator. Some key points to note are that our model depends on the data, so, it cannot predict the words that are out of its vocabulary. We used a small dataset consisting of 8000 images. For production-level models, we need to train on datasets larger than 100,000 images which can produce better accuracy models.

17. References

- https://www.tensorflow.org/tutorials/text/image_captioning
- https://towardsdatascience.com/image-captioning-in-deep-learning-9cd23fb4d8d2
- https://keras.io/examples/vision/image_captioning/
- https://machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/
- Jonathan Hui Blog. https://jhui.github.io/2017/03/15/Soft-and-hard-attention .
- Alex Graves. Generating sequences with recurrent neural networks. CoRR, abs/1308.0850, 2013. 8
- Micah Hodosh, Peter Young, and Julia Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. J. Artif. Int. Res., 47(1):853–899, May 2013.
- Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. IEEE Trans. Pattern Anal. Mach. Intell., 39(4):664–676, Apr. 2017.

THANK YOU!!

Team: AI_4_GOOD