Image Captioning

Introduction to Cyber-Physical Systems

Period: Monsoon 2020

Group-03

V.Rishitha

G.Soumya

T.Bhavani

Motivation

- Scene understanding, one biggest challenge faced by CV
- An assistant to low visioned people.
- Self Driving cars, CCTV cameras,
- Easy access to web content etc.

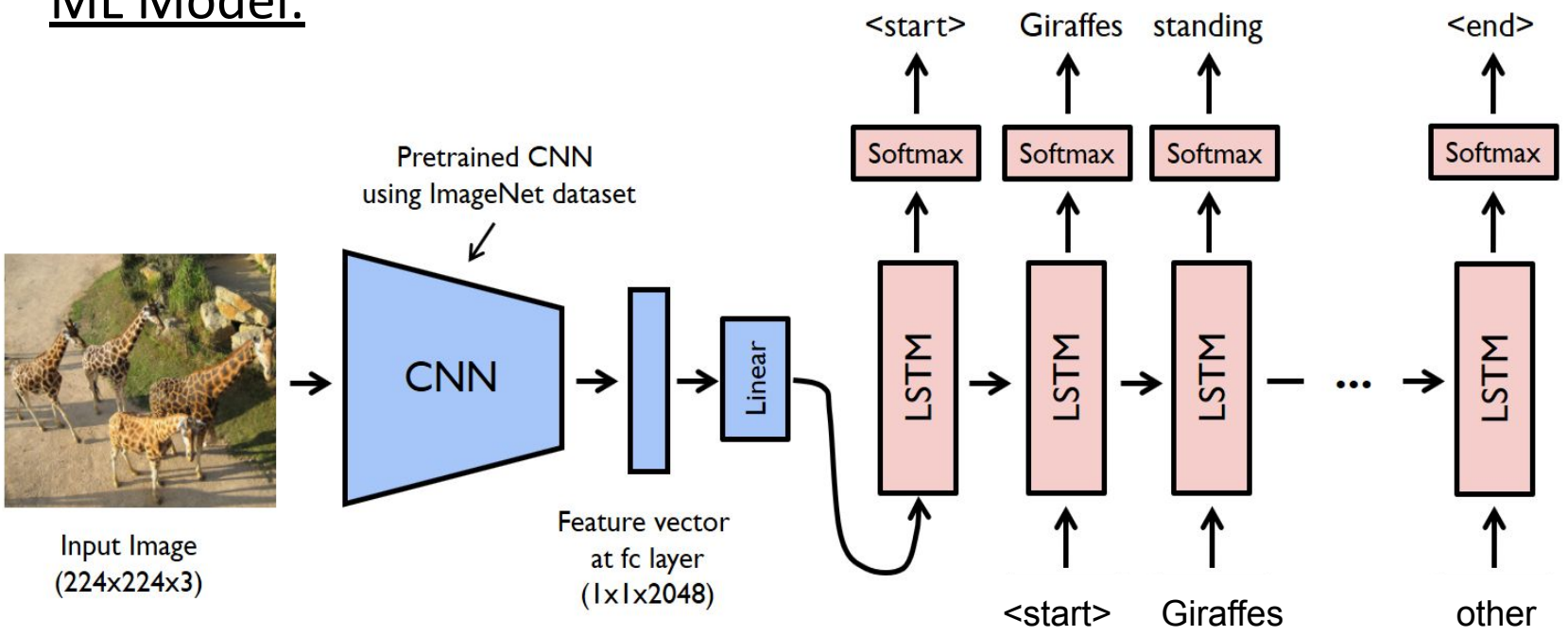
Workflow

- Data discovery
  - Identifying and loading the data
- Data preprocessing
  - Images - reshaped, encoded
  - Captions - adding tags(Start & end), padding sequences
- Building model
  - Resnet - predicting labels in image, Feature Extraction
  - LSTM - sequence prediction
- Model testing
- Model Deployment -Using ngrok

A little girl covered in paint sits in front of a painted rainbow with her hands in a bowl .
A little girl is sitting in front of a large painted rainbow .
A small girl in the grass plays with fingerpaints in front of a white canvas with a rainbow on it .
There is a girl with pigtails sitting in front of a rainbow painting .
Young girl with pigtails painting outside in the grass .

Image Captioning

ML Model:



Ref: https://www.analyticsvidhya.com/blog/2018/04/solving-an-image-captioning-task-using-deep-learning/

```python
#function to process images
from keras.preprocessing import image
import numpy as np

def preprocess_input(x):
    x /= 255.
    x -= 0.5
    x *= 2.
    return x

def preprocessing(img_path):
    im = image.load_img(img_path, target_size=(224,224,3))
    im = image.img_to_array(im)
    im = np.expand_dims(im, axis=0)
    im = preprocess_input(im)
    return im
```

```
#returns first 5 rows
pd_dataset.head()
```

	image_id	captions
0	2513260012_03d33305cf.jpg	<start> A black dog is running after a white d...
1	2513260012_03d33305cf.jpg	<start> Black dog chasing brown dog through sn...
2	2513260012_03d33305cf.jpg	<start> Two dogs chase each other across the s...
3	2513260012_03d33305cf.jpg	<start> Two dogs play together in the snow . <...
4	2513260012_03d33305cf.jpg	<start> Two dogs running through a low lying b...

```
# Creating a list of all unique words
unique = []
for i in words:
    unique.extend(i)
unique = list(set(unique))

print("Unique words in whole training captions data set: {}".format(len(unique)))

vocab_size = len(unique)
```

```
Unique words in whole training captions data set: 8253
```

```
[ ] # Vectorization
    word_2_indices = {val:index for index, val in enumerate(unique)}
    indices_2_word = {index:val for index, val in enumerate(unique)}
```

```
print(word_2_indices['traffic'])
print(indices_2_word[4011])
print(word_2_indices['<end>'])
print(indices_2_word[8252])
```

```
3760
Senior
874
passing
```

Model Description

For Images:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 128)               262272
_________________________________________________________________
repeat_vector (RepeatVector) (None, 40, 128)           0
=================================================================
Total params: 262,272
Trainable params: 262,272
Non-trainable params: 0
_________________________________________________________________
```

For Captions(Text):

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 40, 128)           1056384
_________________________________________________________________
lstm (LSTM)                  (None, 40, 256)           394240
_________________________________________________________________
time_distributed (TimeDistri (None, 40, 128)           32896
=================================================================
Total params: 1,483,520
Trainable params: 1,483,520
Non-trainable params: 0
_________________________________________________________________
```

Model Description(Contd..)

Final Model:

```
model.fit([images, captions], next_words, batch_size=512, epochs=200)
```

Concatenation of the above 2 Sequential Models:

Layer (type)	Output Shape	Param #	Connected to
embedding_input (InputLayer)	[(None, 40)]	0	
dense_input (InputLayer)	[(None, 2048)]	0	
embedding (Embedding)	(None, 40, 128)	1056384	embedding_input[0][0]
dense (Dense)	(None, 128)	262272	dense_input[0][0]
lstm (LSTM)	(None, 40, 256)	394240	embedding[0][0]
repeat_vector (RepeatVector)	(None, 40, 128)	0	dense[0][0]
time_distributed (TimeDistribut	(None, 40, 128)	32896	lstm[0][0]
concatenate (Concatenate)	(None, 40, 256)	0	repeat_vector[0][0] time_distributed[0][0]
lstm_1 (LSTM)	(None, 40, 128)	197120	concatenate[0][0]
lstm_2 (LSTM)	(None, 512)	1312768	lstm_1[0][0]
dense_2 (Dense)	(None, 8253)	4233789	lstm_2[0][0]
activation (Activation)	(None, 8253)	0	dense_2[0][0]

Total params: 7,489,469
Trainable params: 7,489,469
Non-trainable params: 0

Results

- For a training dataset of 6k Images, at

Epoch -1:

```
Epoch 1/200
150/150 [==============================] - 12s 78ms/step - loss: 5.4202 - accuracy: 0.1001
Epoch 2/200
150/150 [==============================] - 12s 78ms/step - loss: 5.0513 - accuracy: 0.1221
```

Epoch -200:

```
Epoch 199/200
150/150 [==============================] - 12s 77ms/step - loss: 0.2910 - accuracy: 0.9003
Epoch 200/200
150/150 [==============================] - 12s 77ms/step - loss: 0.2864 - accuracy: 0.9003
```

- Total Number of Unique words -8253.
- Maximum length of the caption found -40 for the following,

<start> An African-American man wearing a green sweatshirt and blue vest is holding up 2 dollar bills in front of his face , while standing on a busy sidewalk in front of a group of men playing instruments . <end>

```
def predict_captions(image):
    start_word = ["<start>"]
    while True:
        par_caps = [word_2_indices[i] for i in start_word]
        par_caps = sequence.pad_sequences([par_caps], maxlen=max_len, padding='post')
        preds = model.predict([np.array([image]), np.array(par_caps)])
        word_pred = indices_2_word[np.argmax(preds[0])]
        start_word.append(word_pred)


        if word_pred == "<end>" or len(start_word) > max_len:
            break

    return ' '.join(start_word[1:-1])

Argmax_Search = predict_captions(test_img)
```

Predicted Captions

a young child wearing a blue jacket and dark suit . . .

A boy is hiking across a mountain .

Improvement:

A person taking a white shirt climbs a rail .

a group of people are standing in a line . . . .

Predicted Captions (Contd..)

People shopping at see see see around a farmers market .

A racing car spins up along a track .

A young girl in a red jacket is eating a large White with her face in the camera .

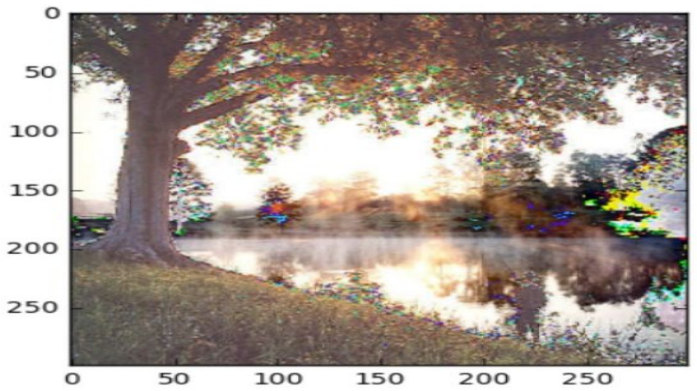
A black dog jumps over a lawn .

Challenges Anticipated

- RESNET 50 is used – Reshaping of image may lead to loss of data.
- Limited size of data set & Vocabulary.
- Can't guarantee that it would caption properly for all kinds of images.
- Due to stochastic nature of model, caption generated may vary.

Challenges Faced

- Loss of data due to reshaping the image



- Resolution

Model Deployment

Related Code:

```
@app.route('/after', methods = ['GET', 'POST'])
def after():
    start_word = ["<start>"]
    while True:
        par_caps = [word_2_indices[i] for i in start_word]
        par_caps = sequence.pad_sequences([par_caps], maxlen=max_len, padding='post')
        preds = model.predict([np.array([image]), np.array(par_caps)])
        word_pred = indices_2_word[np.argmax(preds[0])]
        start_word.append(word_pred)
        if word_pred == "<end>" or len(start_word) > max_len:
            break

    return render_template("after.html", data=' '.join(start_word[1:-1]))
```

➔ Model was deployed using flask with ngrok which will allow to host the model publicly when the server is running.

Thank you

Any Questions?

Please feel free to contact us at

seshasaisoumya.g18@iiits.in
rishitha.v18@iiits.in
bhavani.t18@iiits.in