



North South University
Department of Electrical and Computer Engineering

“Automatic Image Caption Generator Using CNN, LSTM, ResNet-50, VGG-16, VGG-19, Inception-V3, Transformer, OFA. Working on (Blip-ViT, ViT)”

Project Report
Spring 2022
CSE-465
Section: 02
Submitted By:

Group Members	ID
S M Gazzali Arafat Nishan	1831513642
Rofiqul Alam Shehab	1831185042
Shafiul Bashar	1831446642

Submitted To: Dr. Mohammad Ashrafuzzaman Khan (AZK)
Date of submission: 27th April, 2022

1. How did we train?

Model Name	Training Procedure
LSTM-CNN, VGG16-VGG19	For training the model as a supervised learning task, we needed to feed it with input and output sequences. It's impossible to hold such a large amount of data in memory, so we will use a generator method that yields batches. Secondly, define the CNN-LSTM and VGG16-VGG19 model. From the Functional API, we have used the Keras Model to determine the model's structure. Finally, we trained the Image Caption Generator model. We generated the input and output sequences to train our model with 6000 training images. We created a function named model. fit_generator() to fit the batches to the model. At last, we saved the model to our models' folder. After successful model training, our task is to test the model accuracy by inputting test image data. Let's create a python file named test_caption.py to load the model and generate predictions.
ResNet-50	We used the flickr8k dataset for Resnet50. We started by removing all of the numeric and punctuation from the dataset and increasing the image size to 214 pixels. It was divided into three parts: 60% for training, 20% for validations, and 20% for testing. Creating train, test, and validation dataset files with 'image id' and 'captions' headers. Filling in the image ids and captions for each image in the created files for the train, test, and validation datasets. We also used a 50-layer Residual Network Model to get a model summary. In addition to opening the train encoded images.p file and dumping its contents, Loading an image and its caption into a data frame, then storing values from the data frame into 'ds', then Putting all of ds's captions into a list. Creating a list by splitting each caption stored in sentences and storing it in 'words.' following that, Making a list of all the unique words. We took over 2000 images finally, we invoked the fit generator function, where we had previously set the batch size to 512 epochs 10.
Inception-V3 & Transformer	Firstly, we have to preprocess our flickr8k Dataset for the InceptionV3 by eliminating punctuations and numeric values. We had to extract the features vector from the image, and then we used Inceptionv3 to generate an image model from the Keras application. The captions have to be tokenized after that. After that, we divided the dataset into 80 percent training and 20 percent testing. We attempted to calculate the attention weights using scaled dot product attention. MHA (MultiHeadAttention) was also deployed to draw attention to the image. After all, we utilized look ahead mask on several encoders, decoders, encoding and decoding layers so that present queries don't evaluate future tokens during self-attention. The parameters (num layer = 4, d model = 512, dff = 2048, num heads = 8, row size = 8, col size = 8, target vocab size = top k + 1, dropout rate = 0.1) were supplied into the transformer received a startling result of just 28.90 percent after running 100 epochs for varied batch sizes ranging from 0 to 450. The model was then saved, and the captions for the images were reviewed. We've also utilized distinct BLEU scores to explain captions due to a poor result.

2. The training curve:

2.1 Training curve for CNN-LSTM

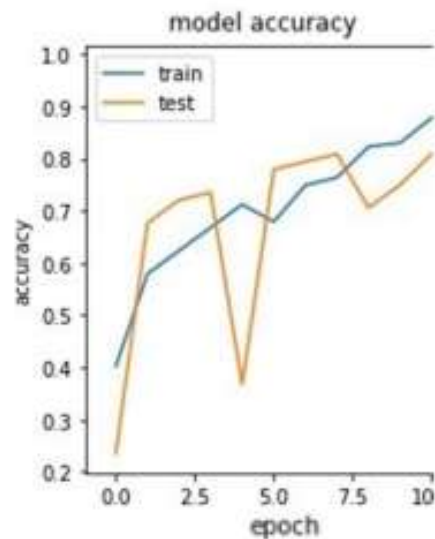


Figure 1: Training curve for CNN-LSTM

2.2 Training curve for VGG-16 & VGG-19

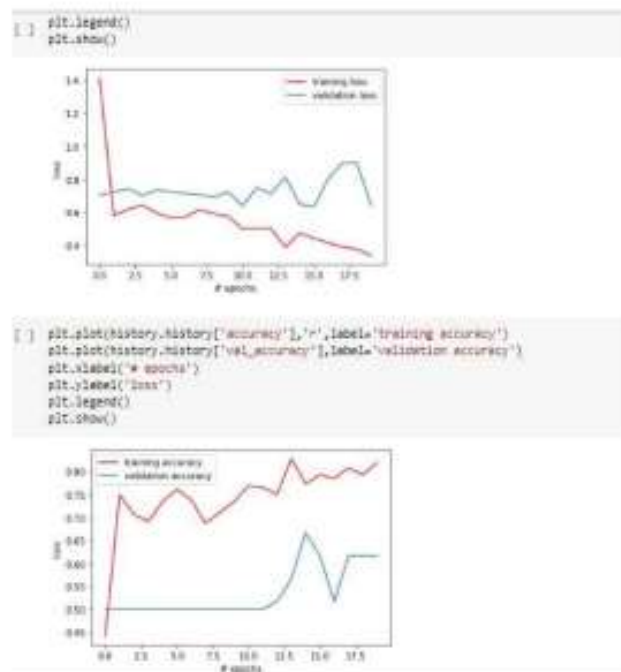


Figure 2: Training curve for VGG-16 & VGG-19

3. Results analysis:

Algorithms Used	Num of Epochs	Accuracy	Num of Batch Size	Num of Hidden layers	Activation function	Progress
1. CNN	10	85.30%	256	3	ReLU	Done
2. LSTM	10	85.30%	256	3	SoftMax	Done
3. ResNet-50	200	90.50%	256	5	ReLU	Done
4. VGG-16	10	77.21%	256	-	SoftMax	Done
5. VGG-19	10	-	256	-	SoftMax	Done
6. Inception-V3	100	28.90%	256	4	ReLU	Done
7. Custom Transformers	100	28.90%	-	8	-	Done

Figure 3: Result Analysis.

3.1 Results Snippet from CNN-LSTM:



Figure 4: Image Captions from CNN-LSTM.

3.2 Results Snippet from VGG-16 & VGG-19:

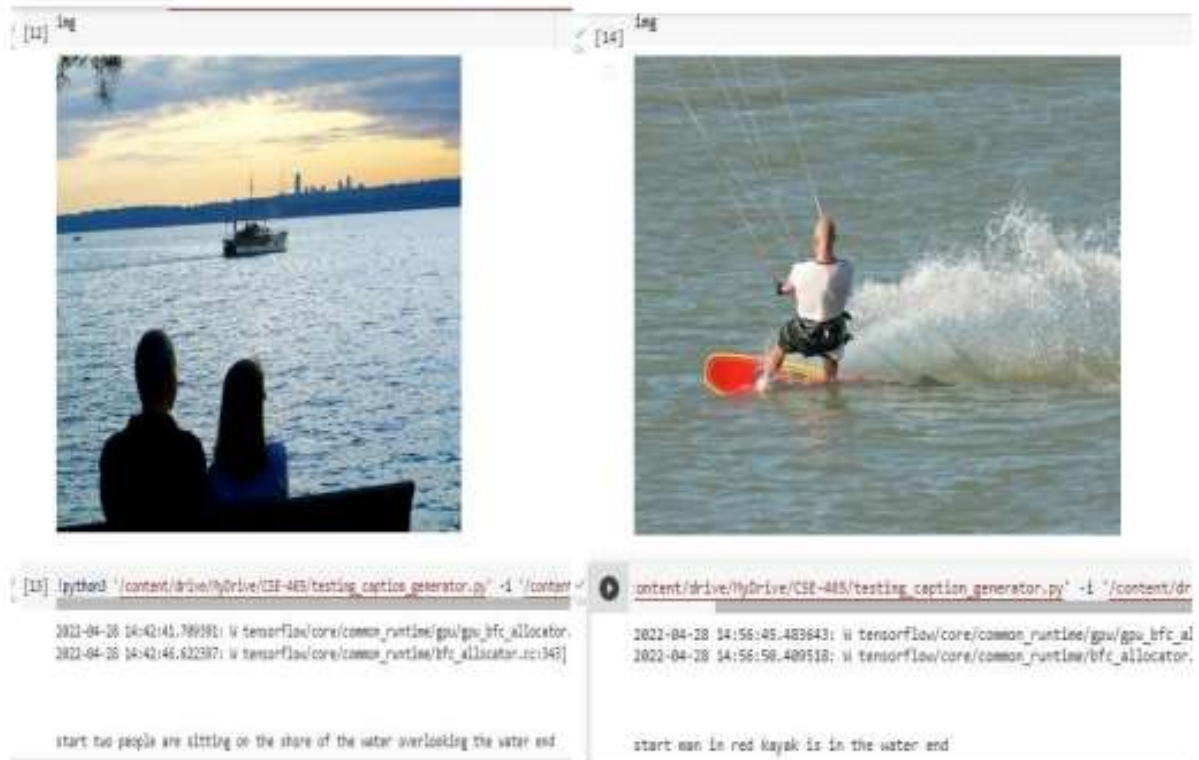


Figure 5: Image Captions from VGG16-VGG19.

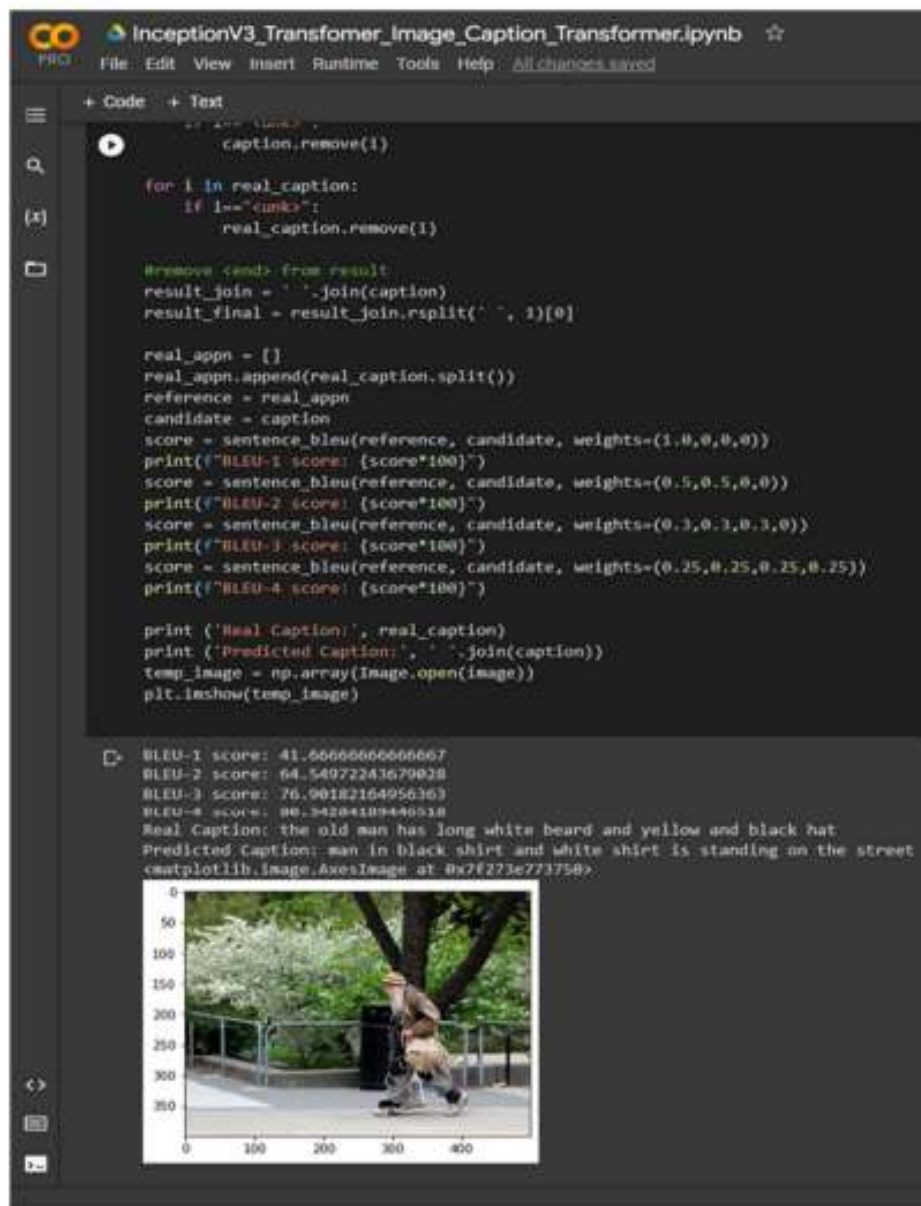
3.3 Results Snippet from ResNet-50:

```
[ ] z = Image(filename='img')
display(z)
print(Argmax_Search)
```



Figure 6: Image Captions from ResNet50.

3.4 Results Snippet from Inception-V3 & Transformer:



```

caption.remove(1)

for i in real_caption:
    if i=="<unk>":
        real_caption.remove(i)

#Remove <end> from result
result_join = ' '.join(caption)
result_final = result_join.rsplit(' ', 1)[0]

real_appn = []
real_appn.append(real_caption.split())
reference = real_appn
candidate = caption

score = sentence_bleu(reference, candidate, weights=(1.0,0.0,0.0))
print(f"BLEU-1 score: {score*100}")
score = sentence_bleu(reference, candidate, weights=(0.5,0.5,0.0))
print(f"BLEU-2 score: {score*100}")
score = sentence_bleu(reference, candidate, weights=(0.3,0.3,0.3,0.0))
print(f"BLEU-3 score: {score*100}")
score = sentence_bleu(reference, candidate, weights=(0.25,0.25,0.25,0.25))
print(f"BLEU-4 score: {score*100}")

print ('Real Caption:', real_caption)
print ('Predicted Caption:', ' '.join(caption))
temp_image = np.array(Image.open(image))
plt.imshow(temp_image)

```

BLEU-1 score: 41.66666666666667
 BLEU-2 score: 64.54972243679028
 BLEU-3 score: 76.90182164956363
 BLEU-4 score: 80.34284189446518
 Real Caption: the old man has long white beard and yellow and black hat
 Predicted Caption: man in black shirt and white shirt is standing on the street
 c:\plotlib.image.AxesImage at 0x7f273e773758




Figure 7: Image Captions from Inception-V3 & Transformer.

BLEU	BLEU Score
BLEU-1	41.66
BLEU-2	64.54
BLEU-3	76.90
BLEU-4	80.34

4. What did we learn from the project:

The Image Caption Generator really helps us to understand deep learning technology at a time. And then we can finally merge that technology to make something outstanding in the future. Through this project, we have learned and implemented different models. To build an image caption generator model, we had to first merge CNN with LSTM, second merge VGG16 and VGG19, third merge Inception-V3& Transformer, and fourth, individually implement ResNet50. To train our model, we used a very small dataset of 8000 images. However, the business-level model used larger datasets of over a million images for higher accuracy. So, if we want to build a more accurate caption generator, we can try this model with large datasets. In the context of a computer vision project, we also discovered that in order to train our model, we needed to define the number of epochs, which may range from 100 to 200. We also realized that in order to train our model, we needed to define the number of epochs, therefore in this circumstance, we needed to utilize a high GPU and RAM, so we purchased Google Collab pro to work on. As a result, Due to lack of compute resources, we had to work on our project on a Google Collab pro with a high GPU and RAM.