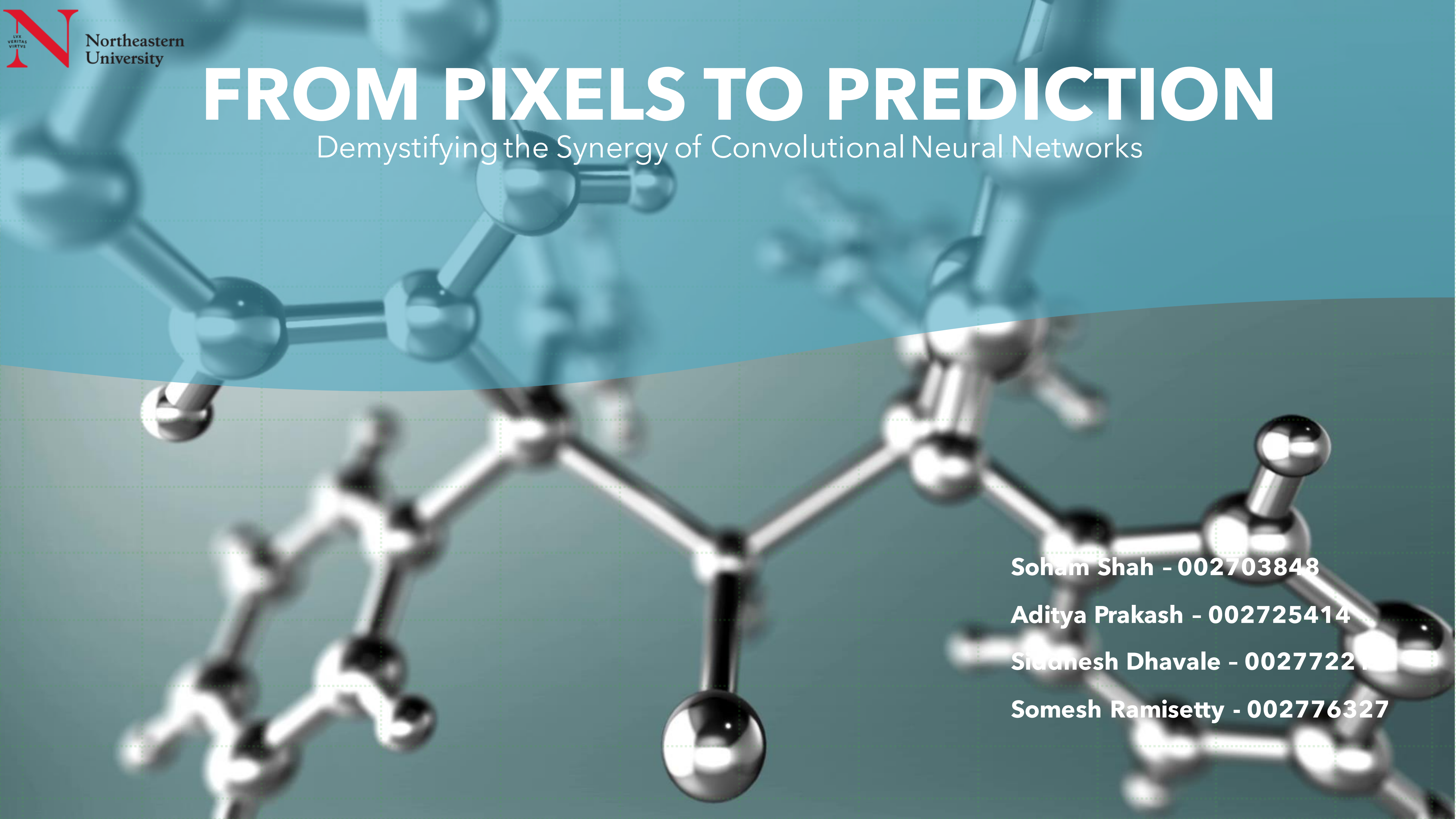
Northeastern University

FROM PIXELS TO PREDICTION

Demystifying the Synergy of Convolutional Neural Networks



Soham Shah – 002703848

Aditya Prakash – 002725414

Siddhesh Dhavale – 0027722[illegible]

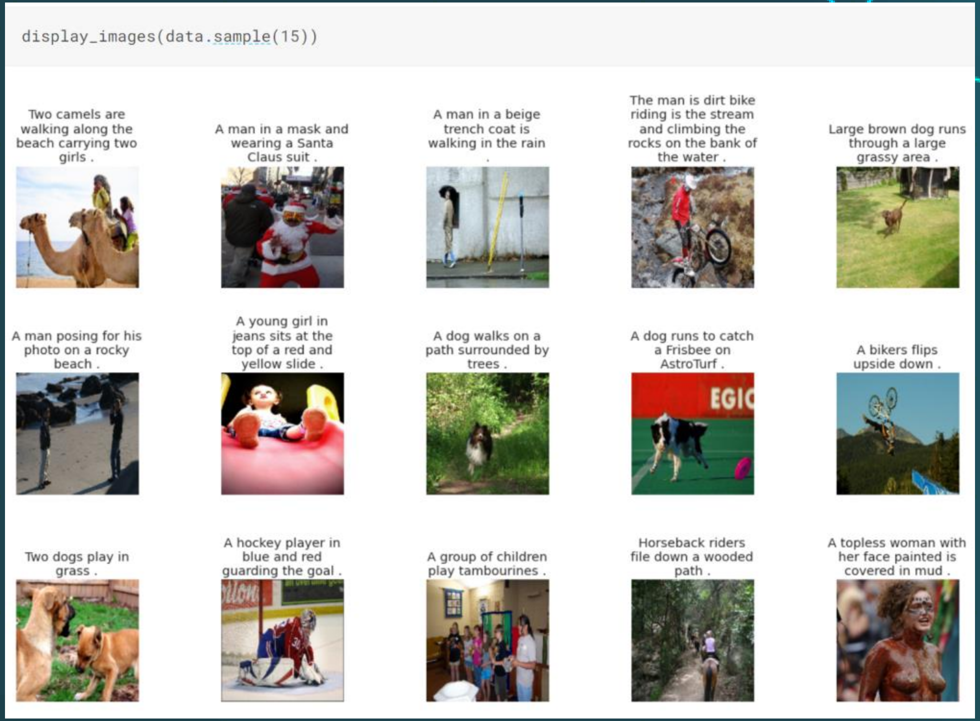
Somesh Ramisetty - 002776327

TABLE OF CONTENTS

INTRODUCTION

- Automatic image caption generation is a challenging task that involves two main steps:
  - Image encoding: **Extracting features** from the image
  - Caption decoding: **Generating a caption** from the extracted features
- In this presentation, we will apply two different approaches to automatic image caption generation:
  - **CNN + LSTM**
  - **ResNet + GRU**
- We will use the **Flickr8k** dataset and **pre-trained** CNN/ResNet models to demonstrate the approaches
- We will **compare the performances** of the two approaches by computing their **BLEU scores**

OVERVIEW OF DATASET



```
display_images(data.sample(15))
```

Two camels are walking along the beach carrying two girls .

A man in a mask and wearing a Santa Claus suit .

A man in a beige trench coat is walking in the rain .

The man is dirt bike riding is the stream and climbing the rocks on the bank of the water .

Large brown dog runs through a large grassy area .

A man posing for his photo on a rocky beach .

A young girl in jeans sits at the top of a red and yellow slide .

A dog walks on a path surrounded by trees .

A dog runs to catch a Frisbee on AstroTurf .

A bikers flips upside down .

Two dogs play in grass .

A hockey player in blue and red guarding the goal .

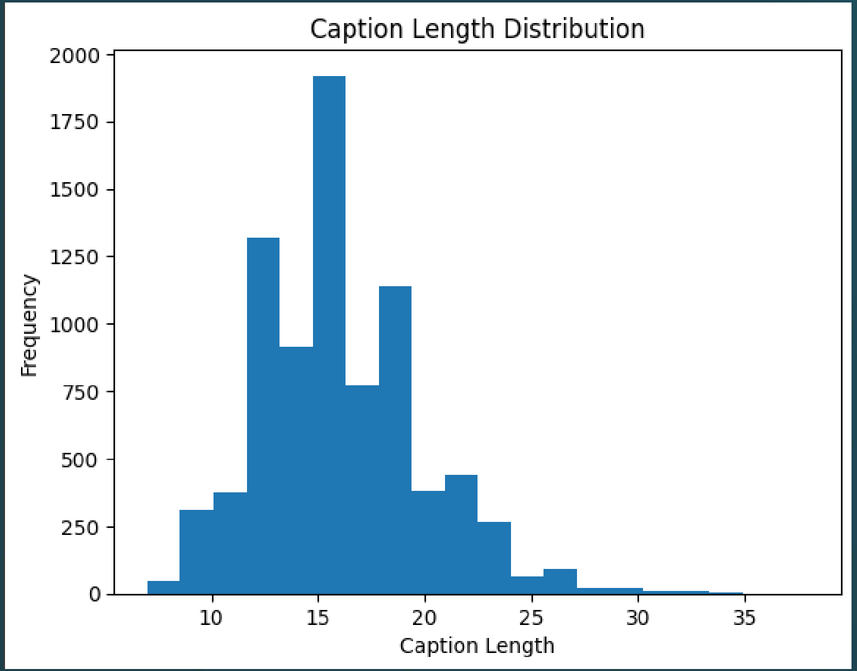
A group of children play tambourines .

Horseback riders file down a wooded path .

A topless woman with her face painted is covered in mud .

- Popular dataset for image captioning research
- A collection of 8,000+ images with corresponding captions
- Each image in the dataset is accompanied by five **human-generated** captions for training and evaluation. Exclusive of any well-known people or locations
- Manually selected to depict a variety of scenes and situations
- Relatively small, making it **easy to train and evaluate models**
- Contains a variety of images, making it **challenging** for models to generalize to new images

CAPTION DISTRIBUTION



- To compute the length of each caption, the captions are first **tokenized** into individual words, and then the data is transformed into its root form using **stemming**.
- Represented with a histogram with **20 bins**
- Useful in **analyzing captions characteristics**
- Useful in identifying **any patterns** or **outliers**

DEEP DIVE INTO MODELS

Convolutional Neural Networks (CNNs)

- CNN is a type of neural network that is commonly used for **image-processing** tasks.
- CNNs are able to learn **spatial features** from images, such as **edges, shapes, and textures.**

Long Short-Term Memory (LSTM) Networks

- LSTM is a type of **recurrent neural network** that is commonly used for **natural language processing** tasks.
- LSTMs are able to **learn long-term dependencies** in sequences, such as the order of words in a sentence.

Gated Recurrent Unit (GRU) Networks

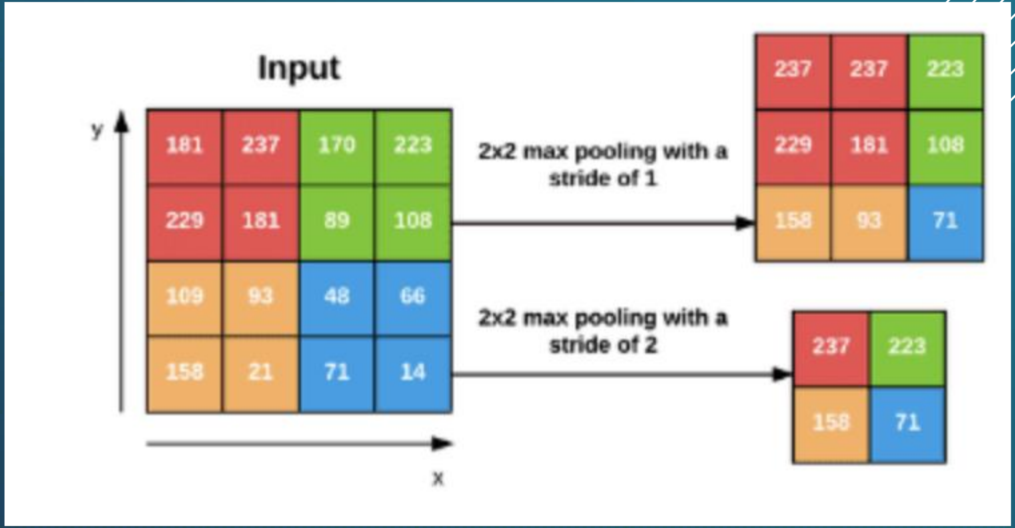
- GRU is a type of recurrent neural network that is similar to LSTM, but it has **fewer parameters**.
- GRUs are often used as a **simpler alternative to LSTMs.**

Residual Networks (ResNets)

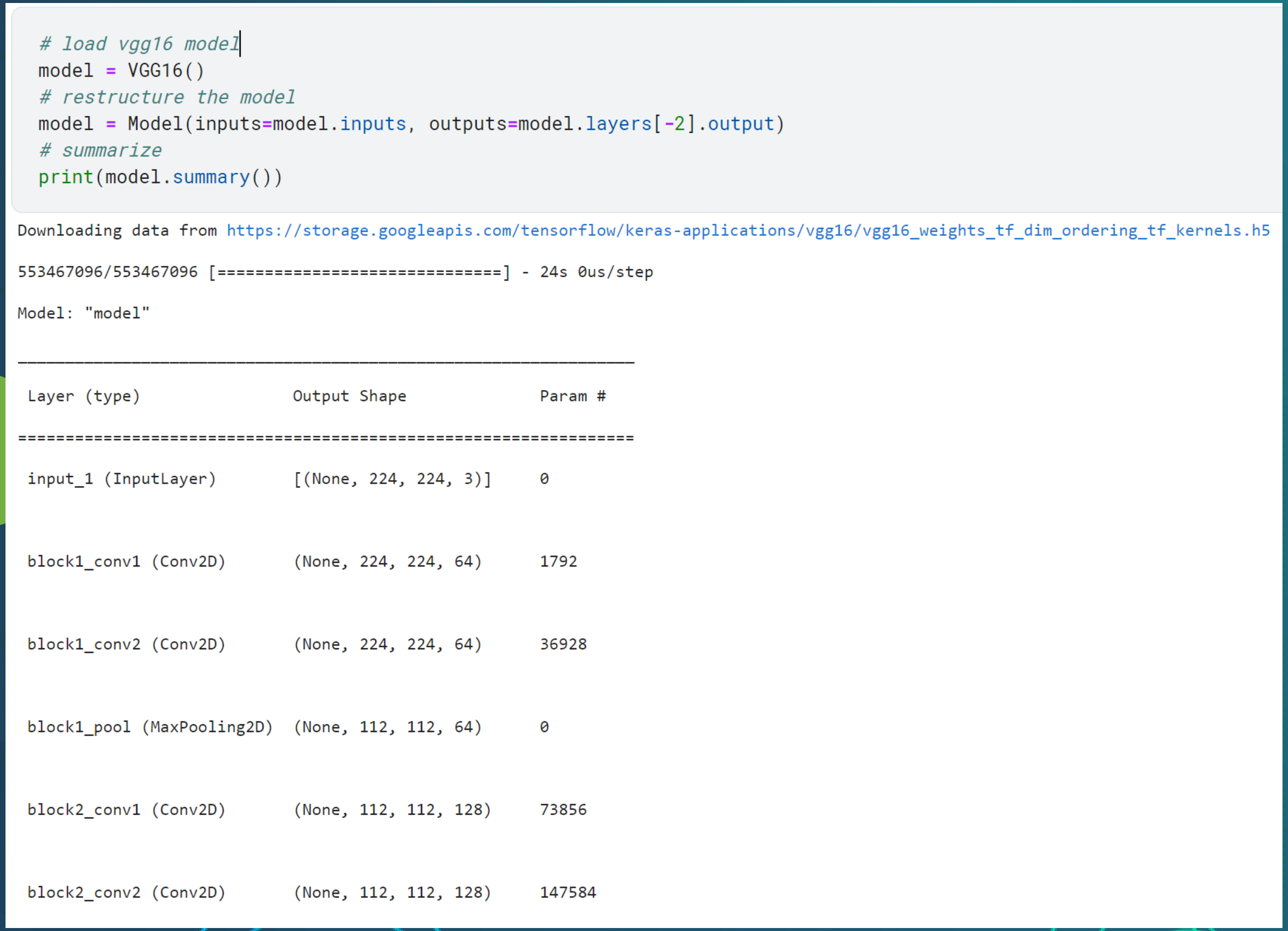
- RESNET is a type of convolutional neural network that was developed to address the **problem of vanishing gradients.**
- ResNets achieve this by using **skip connections,** which allow **information to flow more easily** through the network.

CNN + LSTM

- The CNN + LSTM approach works by first extracting features from the image using a CNN. The CNN is a type of neural network that is well-suited for **image processing tasks**.
- It consists of a series of convolutional layers, each of which applies a filter to the image to extract features at different scales. The features extracted by the CNN are then passed to an LSTM, which is a type of **recurrent neural network** that is well-suited for tasks that involve **sequential data**.
- The LSTM is used to generate the text description by predicting the next word in the sequence, given the previous words and the features extracted by the CNN.



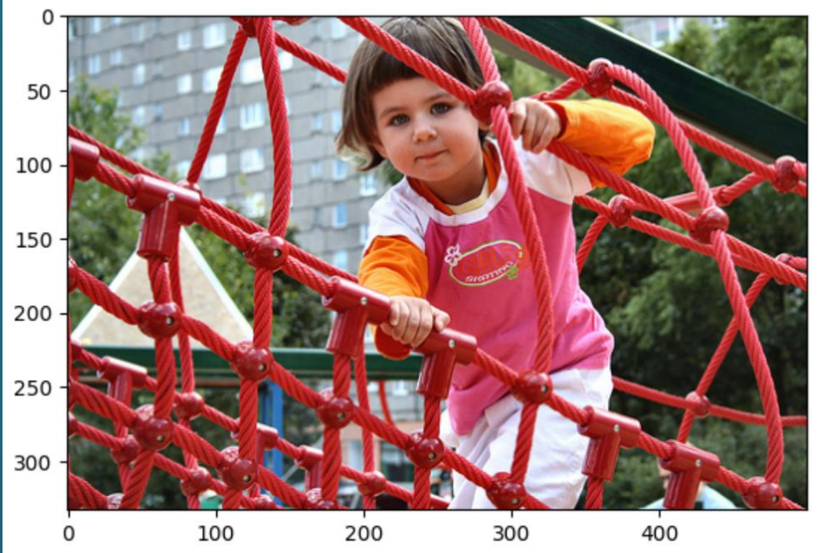
SNIPPETS



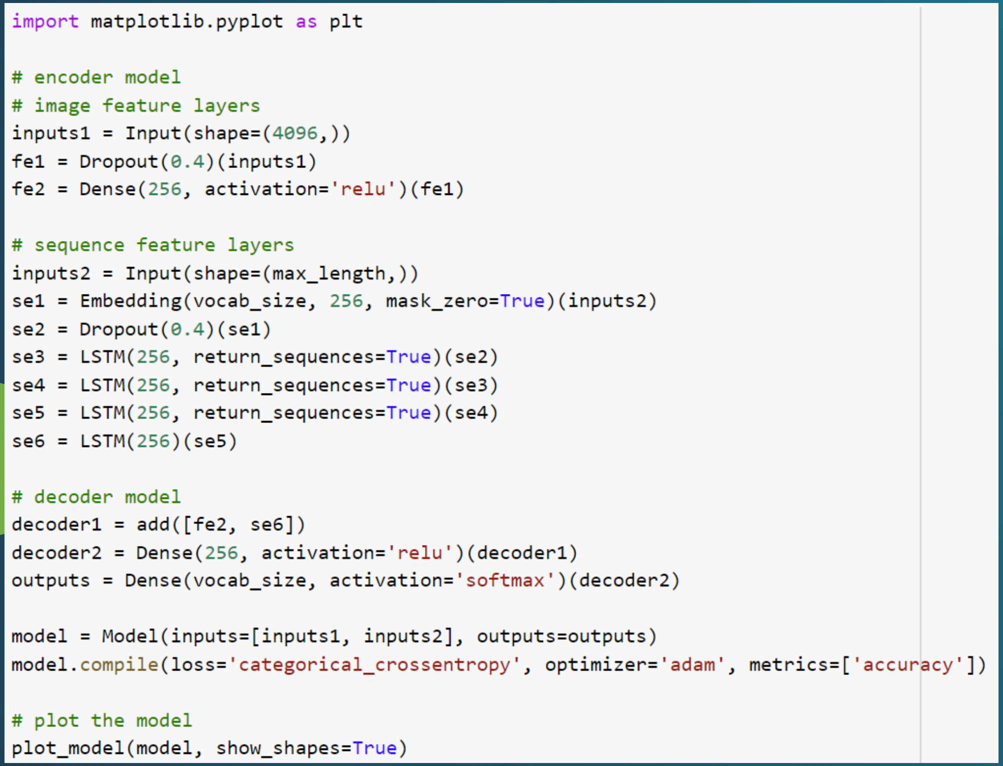
```
# load vgg16 model
model = VGG16()
# restructure the model
model = Model(inputs=model.inputs, outputs=model.layers[-2].output)
# summarize
print(model.summary())
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5

553467096/553467096 [==============================] - 24s 0us/step

Model: "model"

_________________________________________________________________

 Layer (type)                Output Shape              Param #

=================================================================

 input_1 (InputLayer)        [(None, 224, 224, 3)]     0


 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792


 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928


 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0


 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856


 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584
```

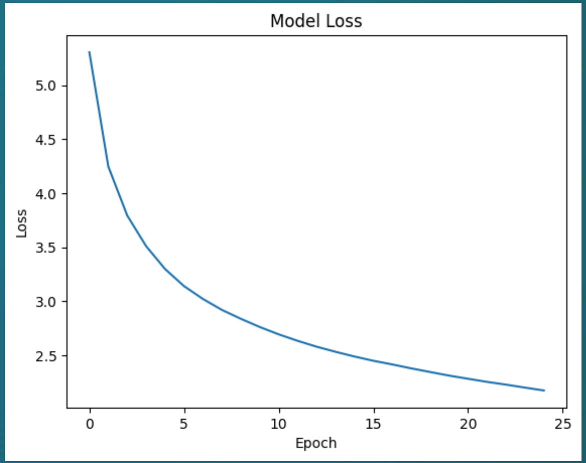
```
---------------------Actual--------------------

startseq child playing on rope net endseq

startseq little girl climbing on red roping endseq

startseq little girl in pink climbs rope bridge at the park endseq

startseq small child grips onto the red ropes at the playground endseq

startseq the small child climbs on red ropes on playground endseq

--------------------Predicted-------------------

startseq young boy in pink shirt is climbing on red roping endseq
```

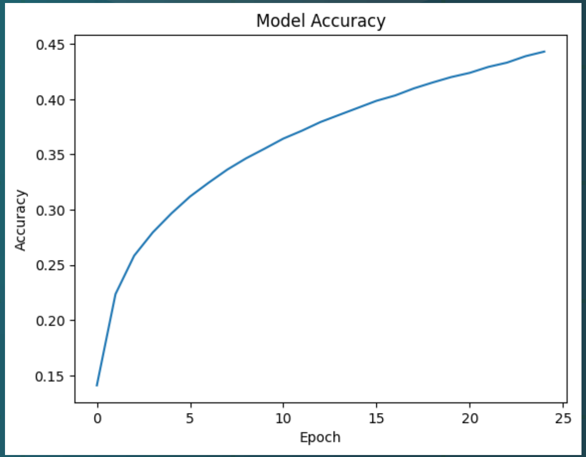


SNIPPETS



```
import matplotlib.pyplot as plt

# encoder model
# image feature layers
inputs1 = Input(shape=(4096,))
fe1 = Dropout(0.4)(inputs1)
fe2 = Dense(256, activation='relu')(fe1)

# sequence feature layers
inputs2 = Input(shape=(max_length,))
se1 = Embedding(vocab_size, 256, mask_zero=True)(inputs2)
se2 = Dropout(0.4)(se1)
se3 = LSTM(256, return_sequences=True)(se2)
se4 = LSTM(256, return_sequences=True)(se3)
se5 = LSTM(256, return_sequences=True)(se4)
se6 = LSTM(256)(se5)

# decoder model
decoder1 = add([fe2, se6])
decoder2 = Dense(256, activation='relu')(decoder1)
outputs = Dense(vocab_size, activation='softmax')(decoder2)

model = Model(inputs=[inputs1, inputs2], outputs=outputs)
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# plot the model
plot_model(model, show_shapes=True)
```

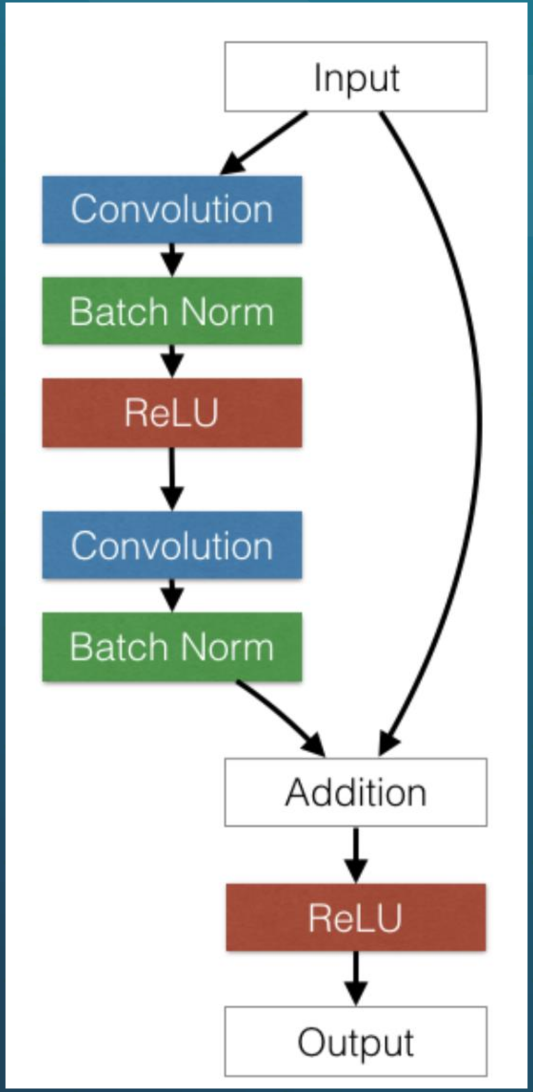




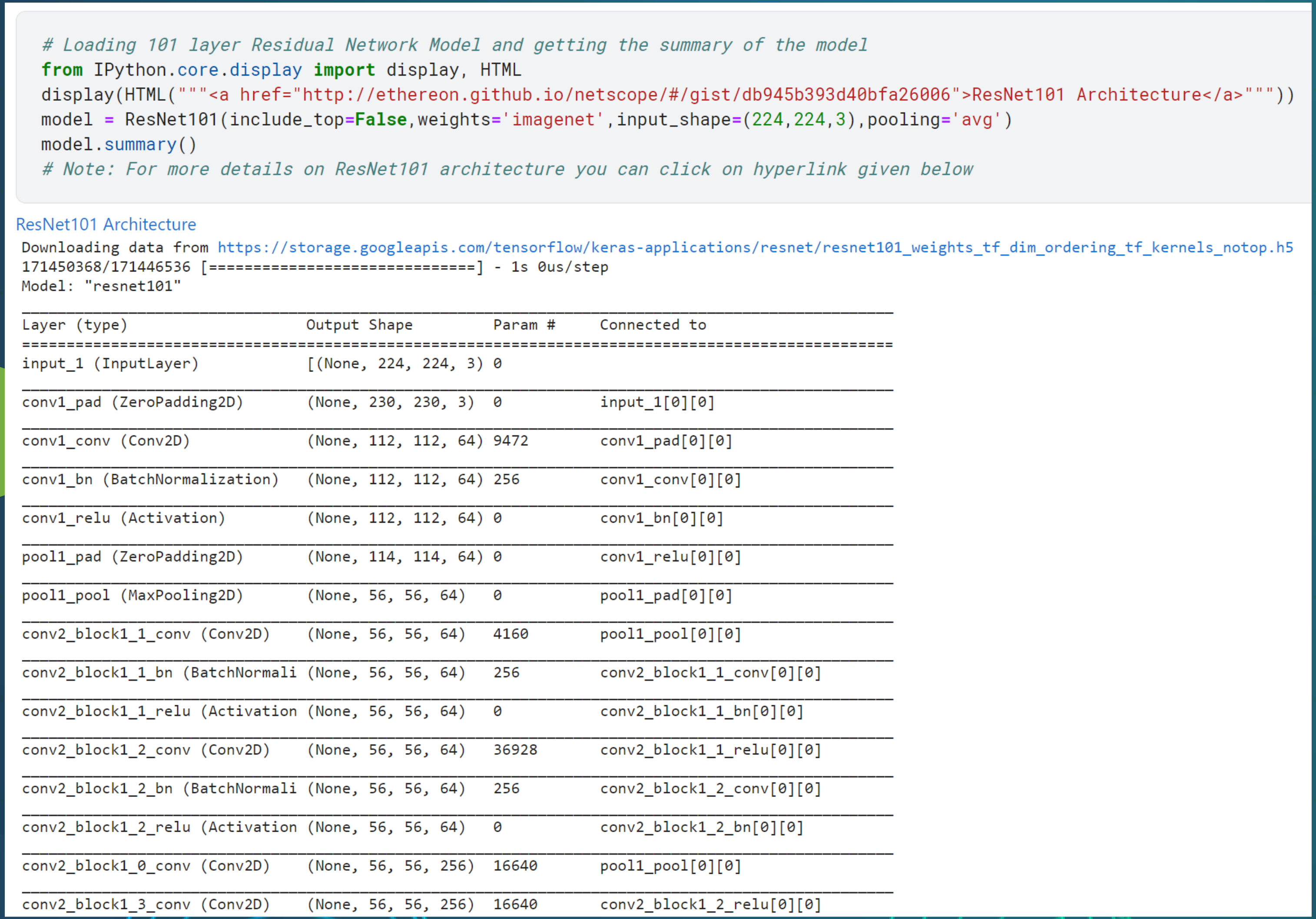
If the loss is decreasing and accuracy is increasing in an elbow curve manner, it means that the model is learning and improving over the epochs.

RESNET + GRU

- ResNet is a **deep neural network architecture** that can capture more complex features from images compared to traditional CNNs.
- GRUs are a type of **recurrent neural network (RNN)** that can model **sequential data efficiently.**
- In this approach, we use a prretrained ResNet to extract features from the input image and then **pass these features** to a **GRU-based caption generator.**
- The GRU generates the **captions word by word**, considering the image **features** and the previously generated words.
- This approach leverages the powerful feature extraction capabilities of ResNet and the **efficient sequential modeling** of GRUs.



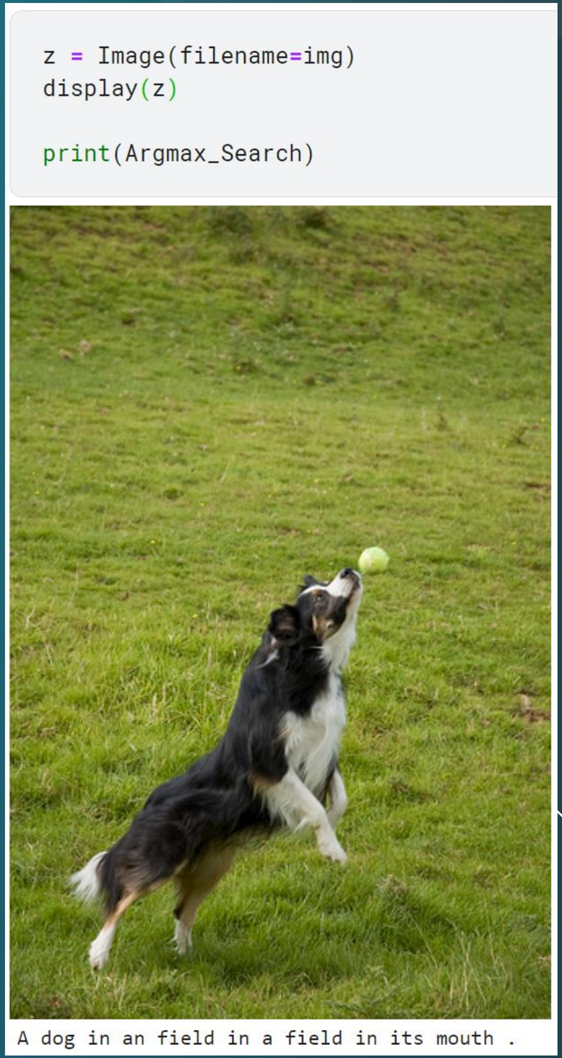
SNIPPETS



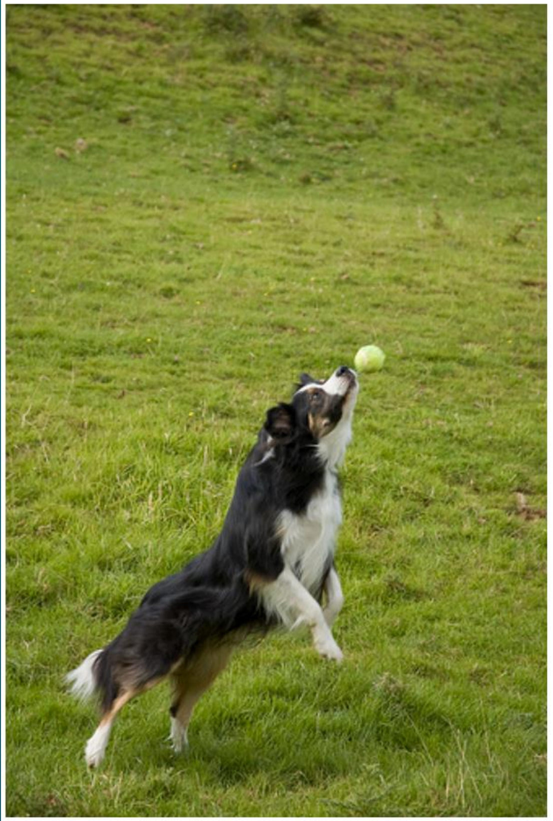
```
# Loading 101 layer Residual Network Model and getting the summary of the model
from IPython.core.display import display, HTML
display(HTML("""<a href="http://ethereon.github.io/netscope/#/gist/db945b393d40bfa26006">ResNet101 Architecture</a>"""))
model = ResNet101(include_top=False,weights='imagenet',input_shape=(224,224,3),pooling='avg')
model.summary()
# Note: For more details on ResNet101 architecture you can click on hyperlink given below
```

ResNet101 Architecture

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet101_weights_tf_dim_ordering_tf_kernels_notop.h5
171450368/171446536 [==============================] - 1s 0us/step
Model: "resnet101"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 224, 224, 3) 0
__________________________________________________________________________________________________
conv1_pad (ZeroPadding2D)       (None, 230, 230, 3)  0           input_1[0][0]
__________________________________________________________________________________________________
conv1_conv (Conv2D)             (None, 112, 112, 64) 9472        conv1_pad[0][0]
__________________________________________________________________________________________________
conv1_bn (BatchNormalization)   (None, 112, 112, 64) 256         conv1_conv[0][0]
__________________________________________________________________________________________________
conv1_relu (Activation)         (None, 112, 112, 64) 0           conv1_bn[0][0]
__________________________________________________________________________________________________
pool1_pad (ZeroPadding2D)       (None, 114, 114, 64) 0           conv1_relu[0][0]
__________________________________________________________________________________________________
pool1_pool (MaxPooling2D)       (None, 56, 56, 64)   0           pool1_pad[0][0]
__________________________________________________________________________________________________
conv2_block1_1_conv (Conv2D)    (None, 56, 56, 64)   4160        pool1_pool[0][0]
__________________________________________________________________________________________________
conv2_block1_1_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block1_1_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_1_relu (Activation (None, 56, 56, 64)   0           conv2_block1_1_bn[0][0]
__________________________________________________________________________________________________
conv2_block1_2_conv (Conv2D)    (None, 56, 56, 64)   36928       conv2_block1_1_relu[0][0]
__________________________________________________________________________________________________
conv2_block1_2_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block1_2_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_2_relu (Activation (None, 56, 56, 64)   0           conv2_block1_2_bn[0][0]
__________________________________________________________________________________________________
conv2_block1_0_conv (Conv2D)    (None, 56, 56, 256)  16640       pool1_pool[0][0]
__________________________________________________________________________________________________
conv2_block1_3_conv (Conv2D)    (None, 56, 56, 256)  16640       conv2_block1_2_relu[0][0]
```

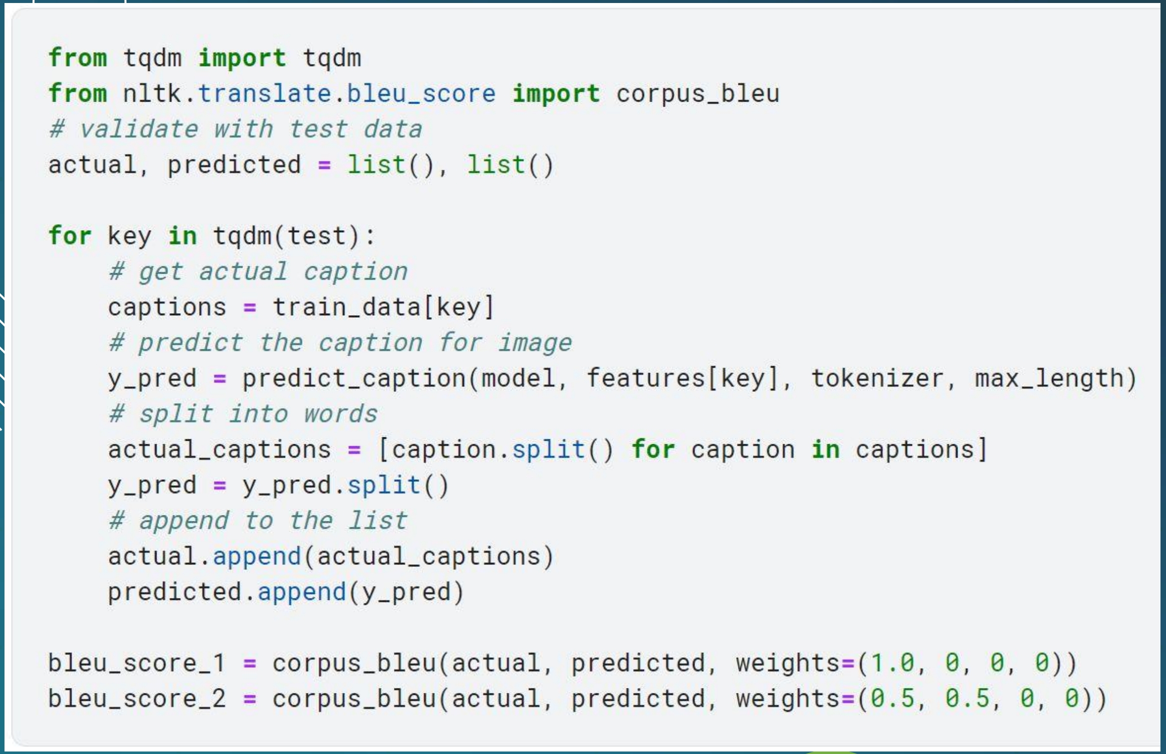


```
z = Image(filename=img)
display(z)

print(Argmax_Search)
```



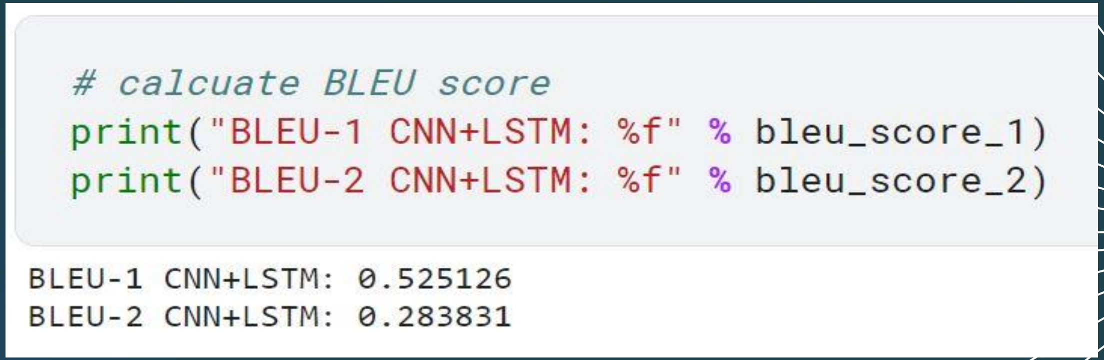
```
A dog in an field in a field in its mouth .
```

EVALUATION



```python
from tqdm import tqdm
from nltk.translate.bleu_score import corpus_bleu
# validate with test data
actual, predicted = list(), list()

for key in tqdm(test):
    # get actual caption
    captions = train_data[key]
    # predict the caption for image
    y_pred = predict_caption(model, features[key], tokenizer, max_length)
    # split into words
    actual_captions = [caption.split() for caption in captions]
    y_pred = y_pred.split()
    # append to the list
    actual.append(actual_captions)
    predicted.append(y_pred)

bleu_score_1 = corpus_bleu(actual, predicted, weights=(1.0, 0, 0, 0))
bleu_score_2 = corpus_bleu(actual, predicted, weights=(0.5, 0.5, 0, 0))
```

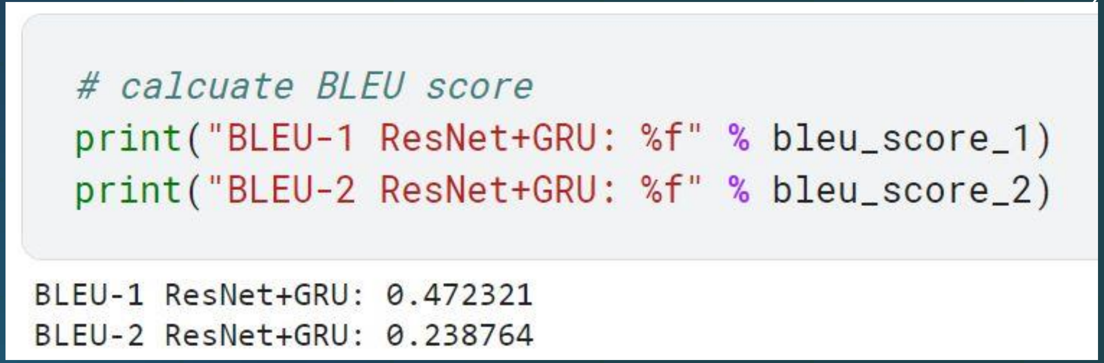
BLEU Score for CNN+LSTM



```python
# calcuate BLEU score
print("BLEU-1 CNN+LSTM: %f" % bleu_score_1)
print("BLEU-2 CNN+LSTM: %f" % bleu_score_2)
```

```
BLEU-1 CNN+LSTM: 0.525126
BLEU-2 CNN+LSTM: 0.283831
```

BLEU Score for RESNET+GRU



```python
# calcuate BLEU score
print("BLEU-1 ResNet+GRU: %f" % bleu_score_1)
print("BLEU-2 ResNet+GRU: %f" % bleu_score_2)
```

```
BLEU-1 ResNet+GRU: 0.472321
BLEU-2 ResNet+GRU: 0.238764
```

PERFORMANCE

- **BLEU(Bi-Lingual Evaluation Understudy)** scores are used to evaluate the performance of the CNN + LSTM and ResNet + GRU approaches.
- BLEU scores measure the similarity between the **generated captions** and the **ground truth captions** from the dataset.
- **Higher** BLEU scores indicate **better performance**.

BLEU SCORE	
CNN + LSTM	RESNET + GRU
0.525126	0.472321
0.283831	0.238764

CONCLUSION

- Both the CNN + LSTM and ResNet + GRU approaches are capable of automatically generating captions for images.
- The choice between the two approaches depends on factors such as the specific use case, available resources, and desired performance.
- CNN + LSTM may be suitable when more emphasis is placed on capturing long-term dependencies in the image features, and when computational resources are relatively abundant.
- ResNet + GRU may be preferable when a simpler model structure is desired, and when shorter-term dependencies in the image features are sufficient for generating accurate captions.
- Further research and experimentation may be needed to determine the optimal approach for a particular image captioning task.