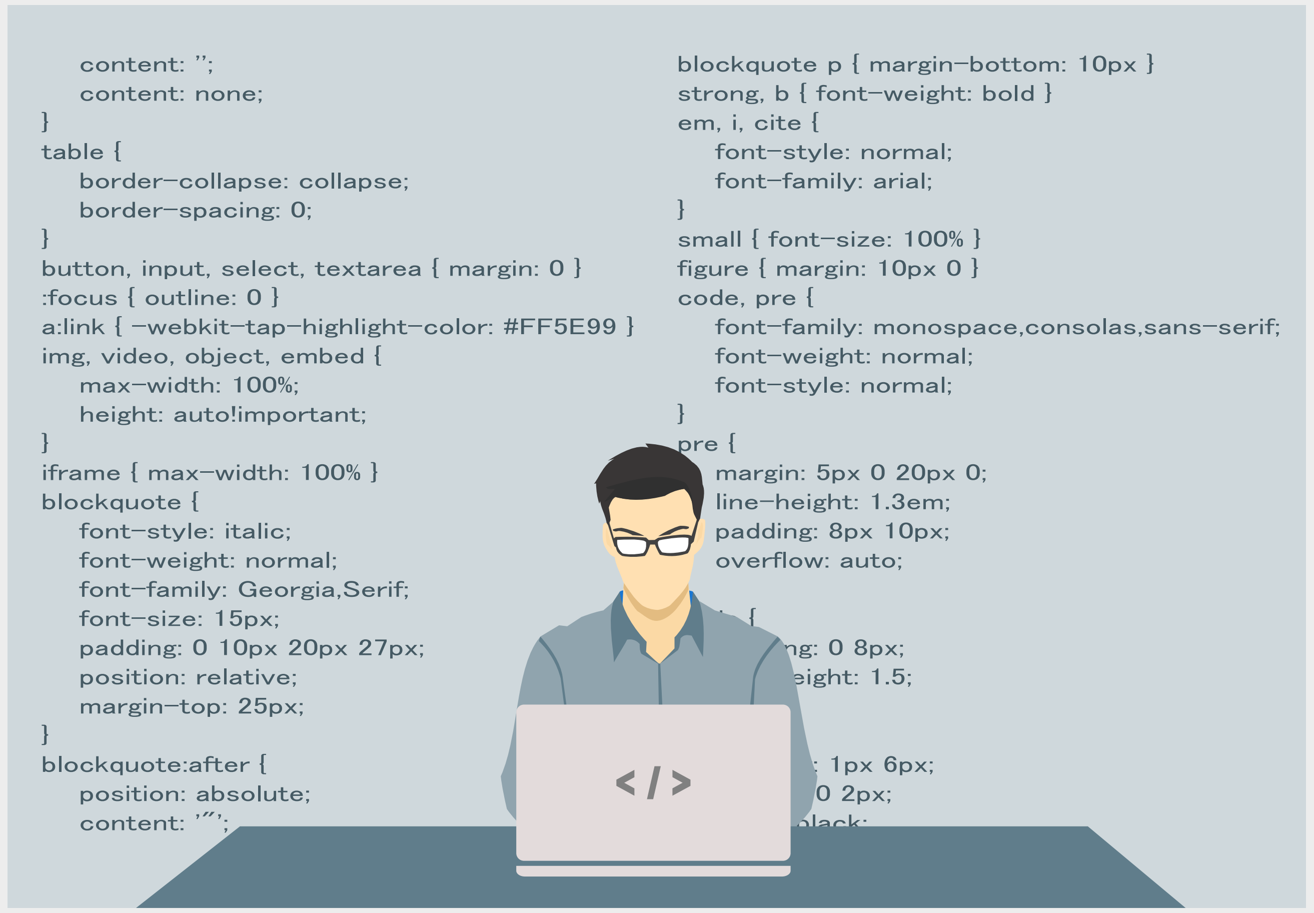
Caption Generation Using Deep Learning

Subject: Deep Learning A

Submitted to: Dr. Dinesh K Vishwakarma

Presentation Structure

Index



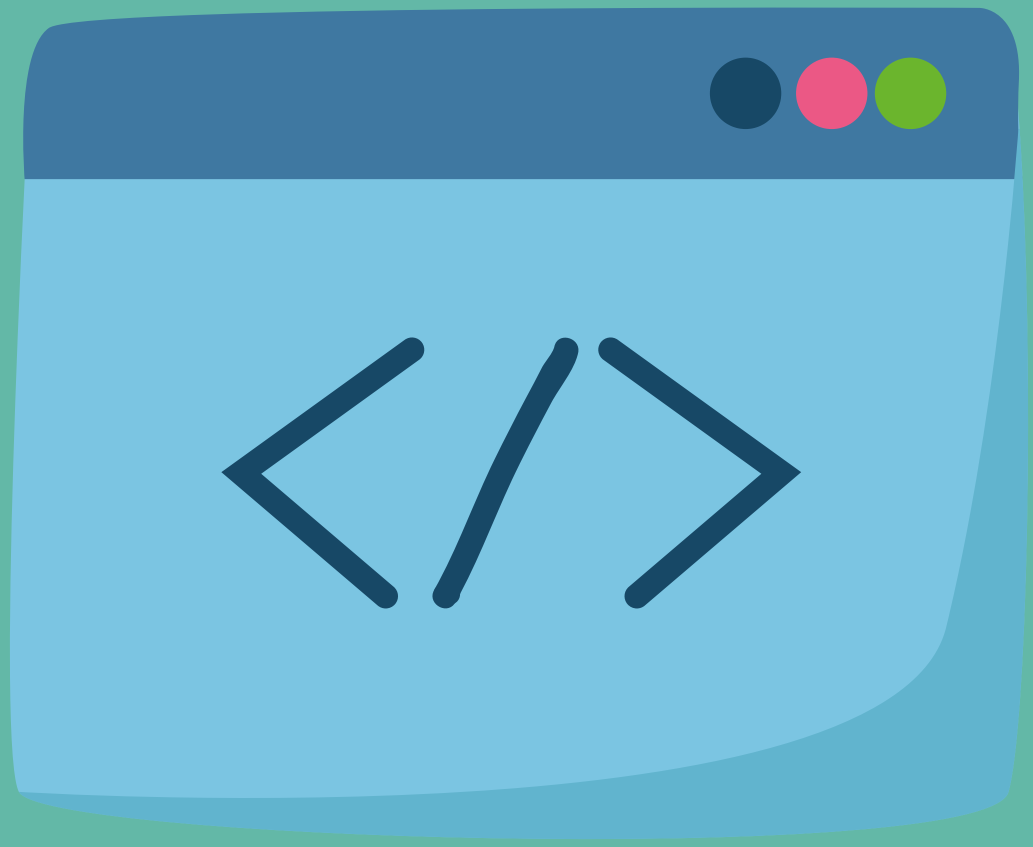
```
    content: '';
    content: none;
}
table {
    border-collapse: collapse;
    border-spacing: 0;
}
button, input, select, textarea { margin: 0 }
:focus { outline: 0 }
a:link { -webkit-tap-highlight-color: #FF5E99 }
img, video, object, embed {
    max-width: 100%;
    height: auto!important;
}
iframe { max-width: 100% }
blockquote {
    font-style: italic;
    font-weight: normal;
    font-family: Georgia,Serif;
    font-size: 15px;
    padding: 0 10px 20px 27px;
    position: relative;
    margin-top: 25px;
}
blockquote:after {
    position: absolute;
    content: '"';
```

```
blockquote p { margin-bottom: 10px }
strong, b { font-weight: bold }
em, i, cite {
    font-style: normal;
    font-family: arial;
}
small { font-size: 100% }
figure { margin: 10px 0 }
code, pre {
    font-family: monospace,consolas,sans-serif;
    font-weight: normal;
    font-style: normal;
}
pre {
    margin: 5px 0 20px 0;
    line-height: 1.3em;
    padding: 8px 10px;
    overflow: auto;
  {
    ng: 0 8px;
    eight: 1.5;

    : 1px 6px;
    0 2px;
    lack;
```

OBJECTIVES

- Preprocessing of the image and text data to train the Deep Neural Network. The dataset will be divided into Train, Development and test sets.
- Build a merged deep learning neural network using CNN and RNN and train it on the preprocessed dataset. The model will be trained to predict the caption(sequence of words), given an image.
- Evaluate the trained caption generator model on the Test set using BLEU Score.



Introduction

- Caption generation is an artificial intelligence problem where a textual description is generated for a given photograph.
- It requires both methods from computer vision to understand the content of the image and a language model from the field of natural language processing to turn the understanding of the image into words in the right order.
- A Convolutional Neural Network (CNN) was used to extract the features from the image.
- A Recurrent Neural Network (RNN), is a sequence model that was used to represent linguistic features of the Text Data.

MODELS USED

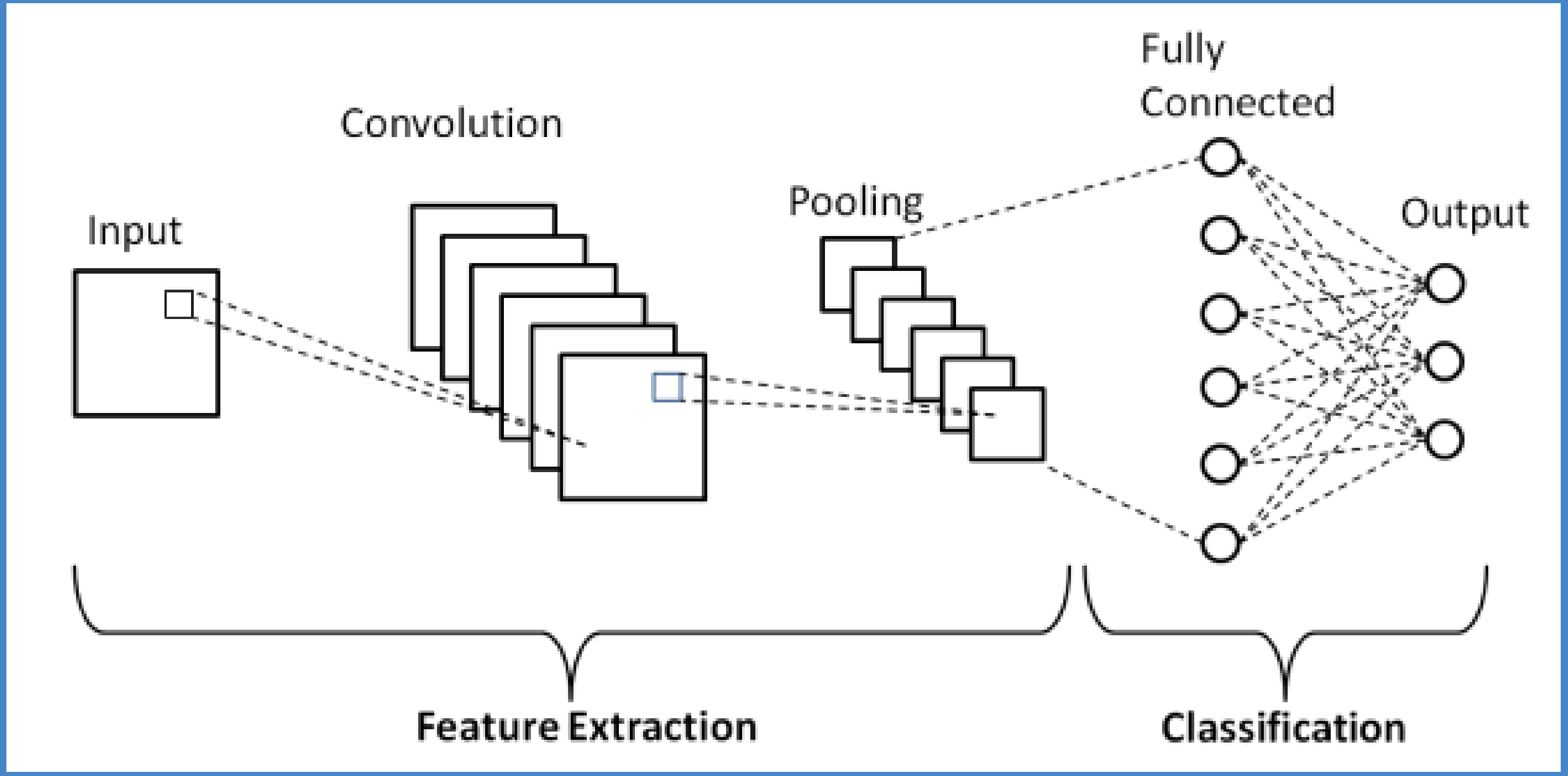
Deep Convolutional Neural Network: VGG-16

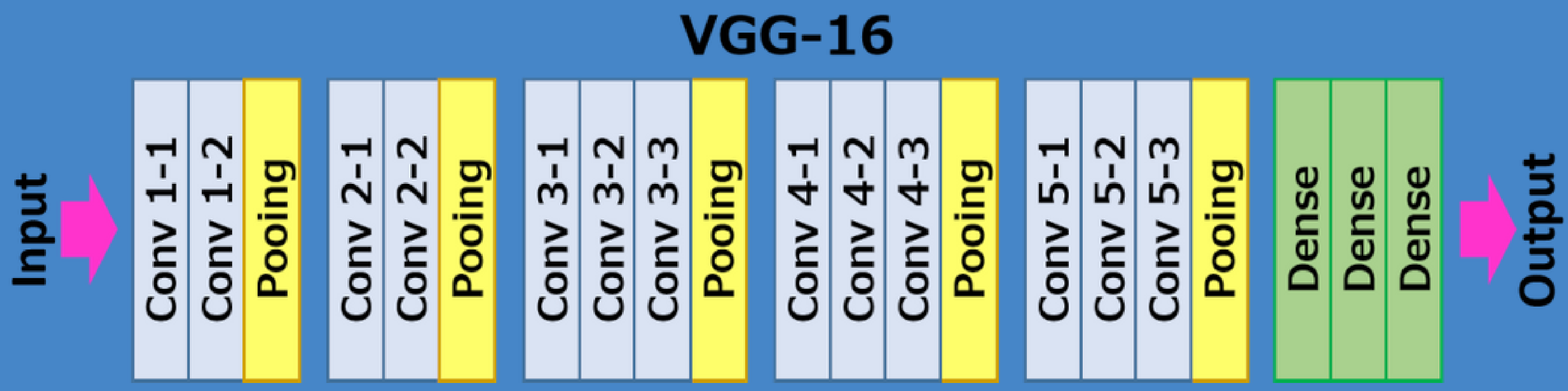
CNN is a class of deep, feed-forward artificial neural systems, extracting features from the Image Data.

CNN collect valuable features from an image by passing it through a series of convolution layers, non-linear activation functions, pooling (down-sampling), and fully connected dense layers.

We used a pre-trained Deep Convolutional Neural Network called VGG-16 imported from keras.applicaitons.

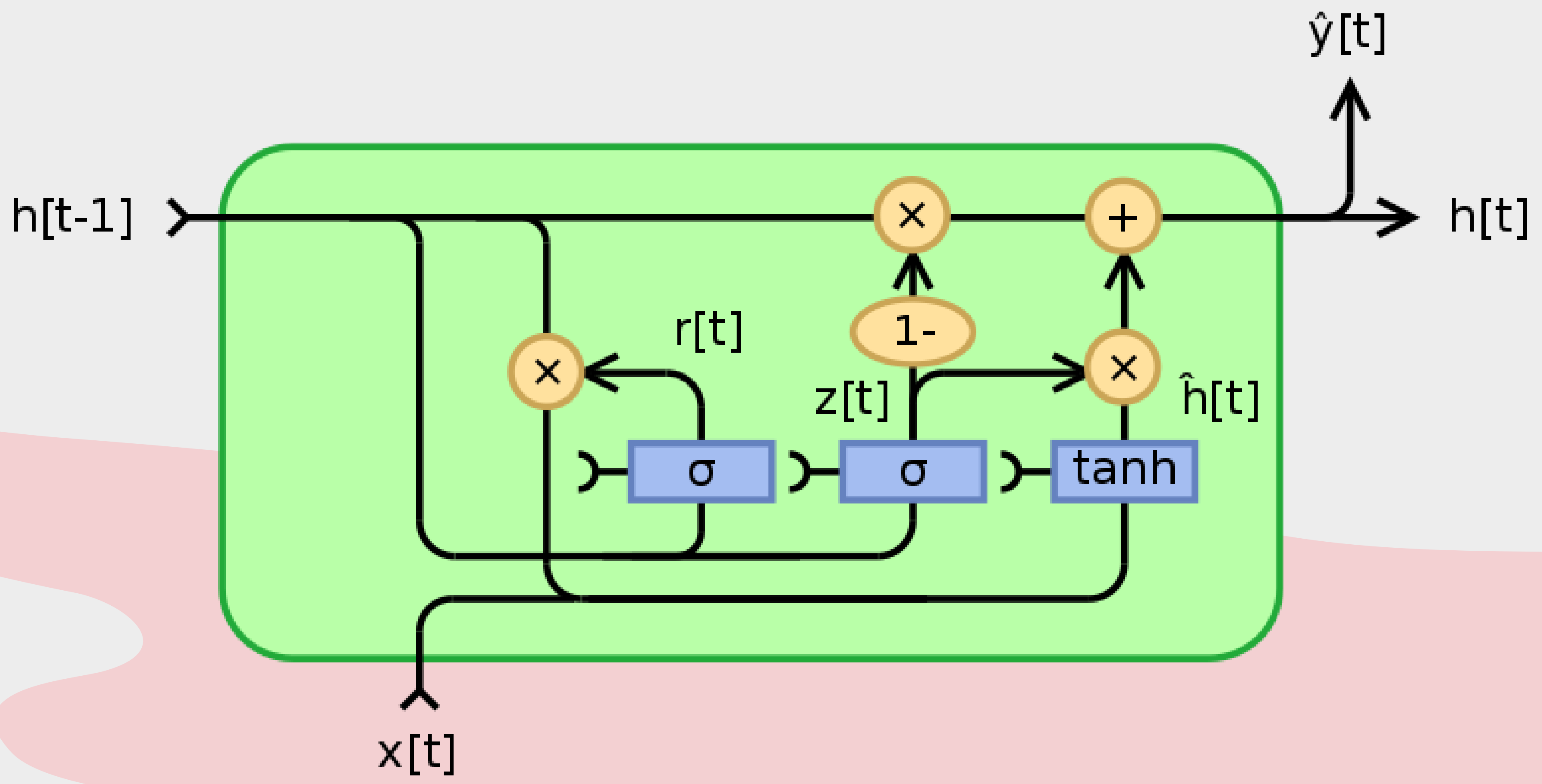
VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman (2014.) having 13 Convolutional layers, 5 Max-Pooling Layers and 3 Fully connected Dense Layers. (First 2 having 4096 Nodes.)





2. GRU

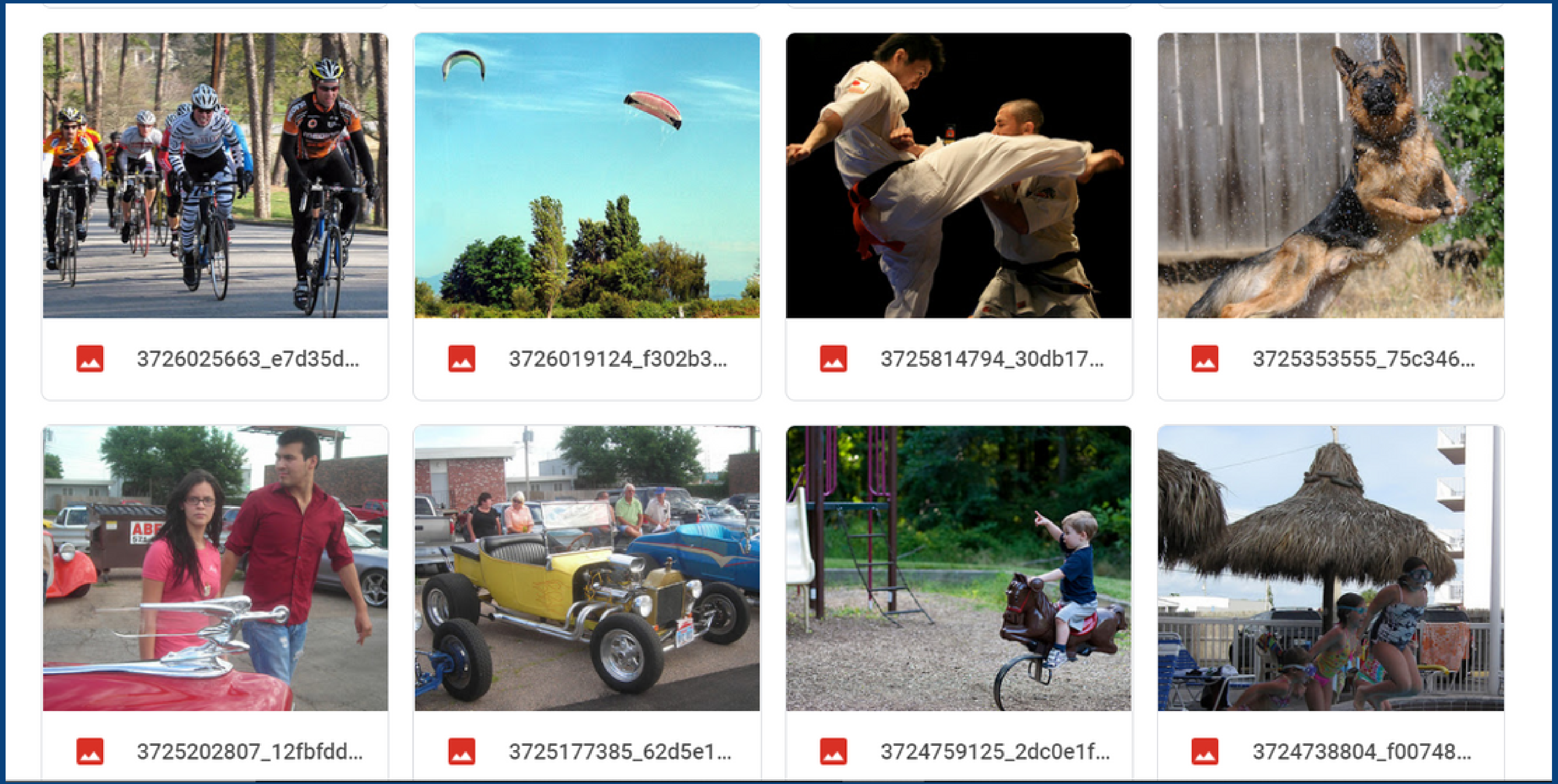
- GRU is a Recurrent Neural Networks (RNNs) / Sequence Model.
- GRU are a variant of Long Short Term Memory (LSTM) but with fewer training parameters. They were introduced in 2014 by Kyunghyun Cho et al.
- It uses two Gates: Update Gate and Reset Gate to decide the flow of information onto next time steps..
- The memory unit present in GRUs solves the problem of vanishing gradient descent for longer sequences.
- It in Natural Language Processing tasks to encode linguistic features and Generate word sequences (Captions in this Case).

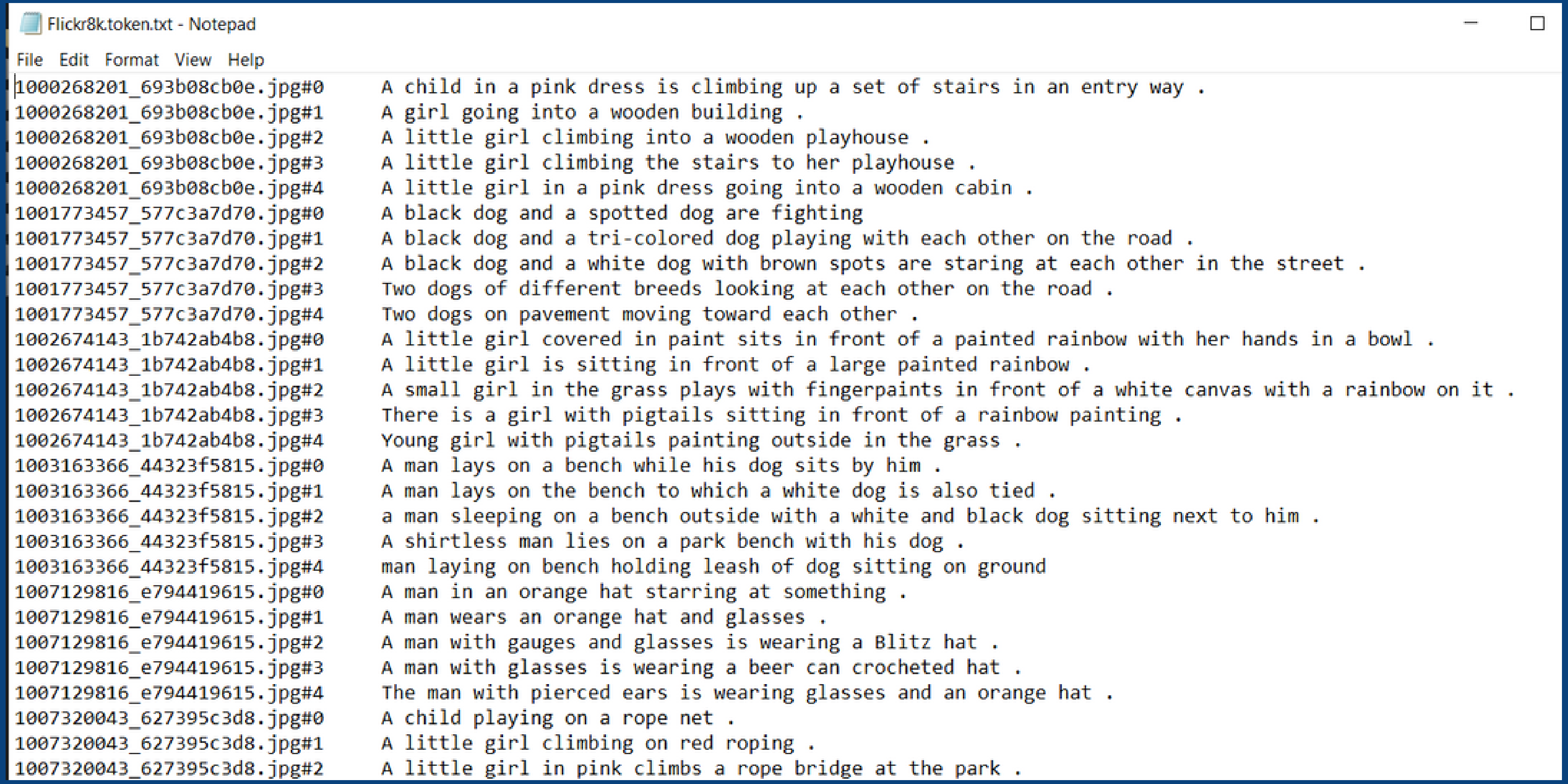


$$z = \sigma(W_z \cdot x_t + U_z \cdot h_{(t-1)} + b_z)$$
$$r = \sigma(W_r \cdot x_t + U_r \cdot h_{(t-1)} + b_r)$$
$$\tilde{h} = tanh(W_h \cdot x_t + r * U_h \cdot h_{(t-1)} + b_z)$$
$$h = z * h_{(t-1)} + (1 - z) * \tilde{h}$$

DATASET

- We used Flickr8k Dataset containing 8000 images in JPEG Format, and multiple text descriptions corresponding to each photograph.
- These models will be trained and deployed using Python along with the Deep Learning API TensorFlow-Keras.
- The dataset will be divided into training dataset (6,000 images), development dataset (1,000 images), and test dataset (1,000 images).
- We will use the BLEU score (Bilingual Evaluation Understudy) to evaluate the Captions predicted from the Model.





```
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_693b08cb0e.jpg#4	A little girl in a pink dress going into a wooden cabin .
1001773457_577c3a7d70.jpg#0	A black dog and a spotted dog are fighting
1001773457_577c3a7d70.jpg#1	A black dog and a tri-colored dog playing with each other on the road .
1001773457_577c3a7d70.jpg#2	A black dog and a white dog with brown spots are staring at each other in the street .
1001773457_577c3a7d70.jpg#3	Two dogs of different breeds looking at each other on the road .
1001773457_577c3a7d70.jpg#4	Two dogs on pavement moving toward each other .
1002674143_1b742ab4b8.jpg#0	A little girl covered in paint sits in front of a painted rainbow with her hands in a bowl .
1002674143_1b742ab4b8.jpg#1	A little girl is sitting in front of a large painted rainbow .
1002674143_1b742ab4b8.jpg#2	A small girl in the grass plays with fingerpaints in front of a white canvas with a rainbow on it .
1002674143_1b742ab4b8.jpg#3	There is a girl with pigtails sitting in front of a rainbow painting .
1002674143_1b742ab4b8.jpg#4	Young girl with pigtails painting outside in the grass .
1003163366_44323f5815.jpg#0	A man lays on a bench while his dog sits by him .
1003163366_44323f5815.jpg#1	A man lays on the bench to which a white dog is also tied .
1003163366_44323f5815.jpg#2	a man sleeping on a bench outside with a white and black dog sitting next to him .
1003163366_44323f5815.jpg#3	A shirtless man lies on a park bench with his dog .
1003163366_44323f5815.jpg#4	man laying on bench holding leash of dog sitting on ground
1007129816_e794419615.jpg#0	A man in an orange hat starring at something .
1007129816_e794419615.jpg#1	A man wears an orange hat and glasses .
1007129816_e794419615.jpg#2	A man with gauges and glasses is wearing a Blitz hat .
1007129816_e794419615.jpg#3	A man with glasses is wearing a beer can crocheted hat .
1007129816_e794419615.jpg#4	The man with pierced ears is wearing glasses and an orange hat .
1007320043_627395c3d8.jpg#0	A child playing on a rope net .
1007320043_627395c3d8.jpg#1	A little girl climbing on red roping .
1007320043_627395c3d8.jpg#2	A little girl in pink climbs a rope bridge at the park .
```

Libraries & Functions

- Numpy
- Pickle
- OS
- Tensorflow-Keras
- keras.applications.vgg16 -> VGG16, preprocess_input
- keras.preprocessing.image
- keras.utils -> to_categorical, plot_model
- keras.models -> Model
- keras.layers -> Input, GRU, Add, Dense, Embedding, Dropout
- keras.callbacks
- keras.preprocessing.text -> Tokenizer
- keras.preprocessing.sequence -> pad_sequences
- nltk.translate.bleu_score -> corpus_bleu

Model Architecture

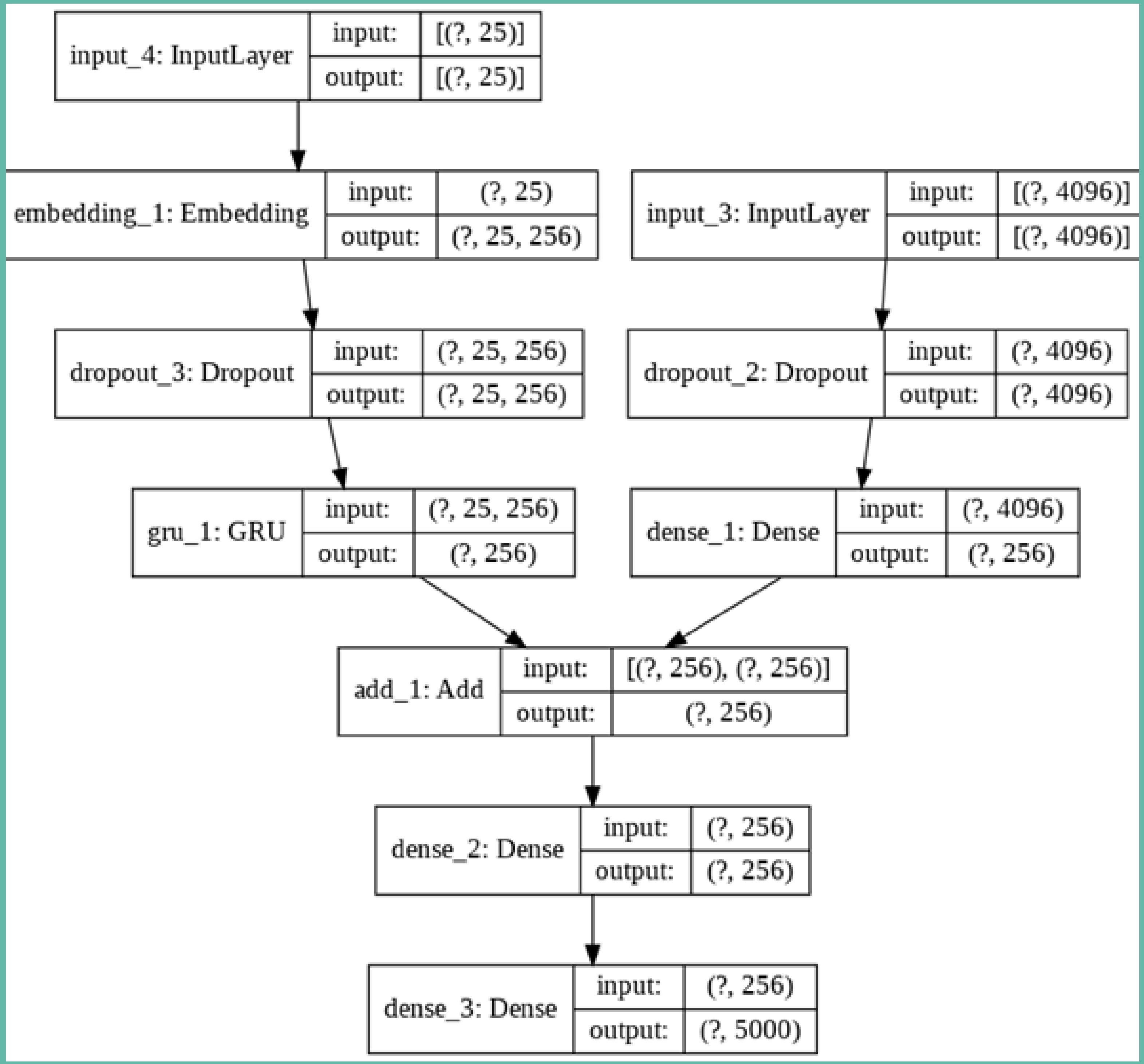
VGG-16 model takes a photo input and produces a feature vector of 4,096 elements. These are processed by a FC Dense layer to produce a 256 element representation of the image.

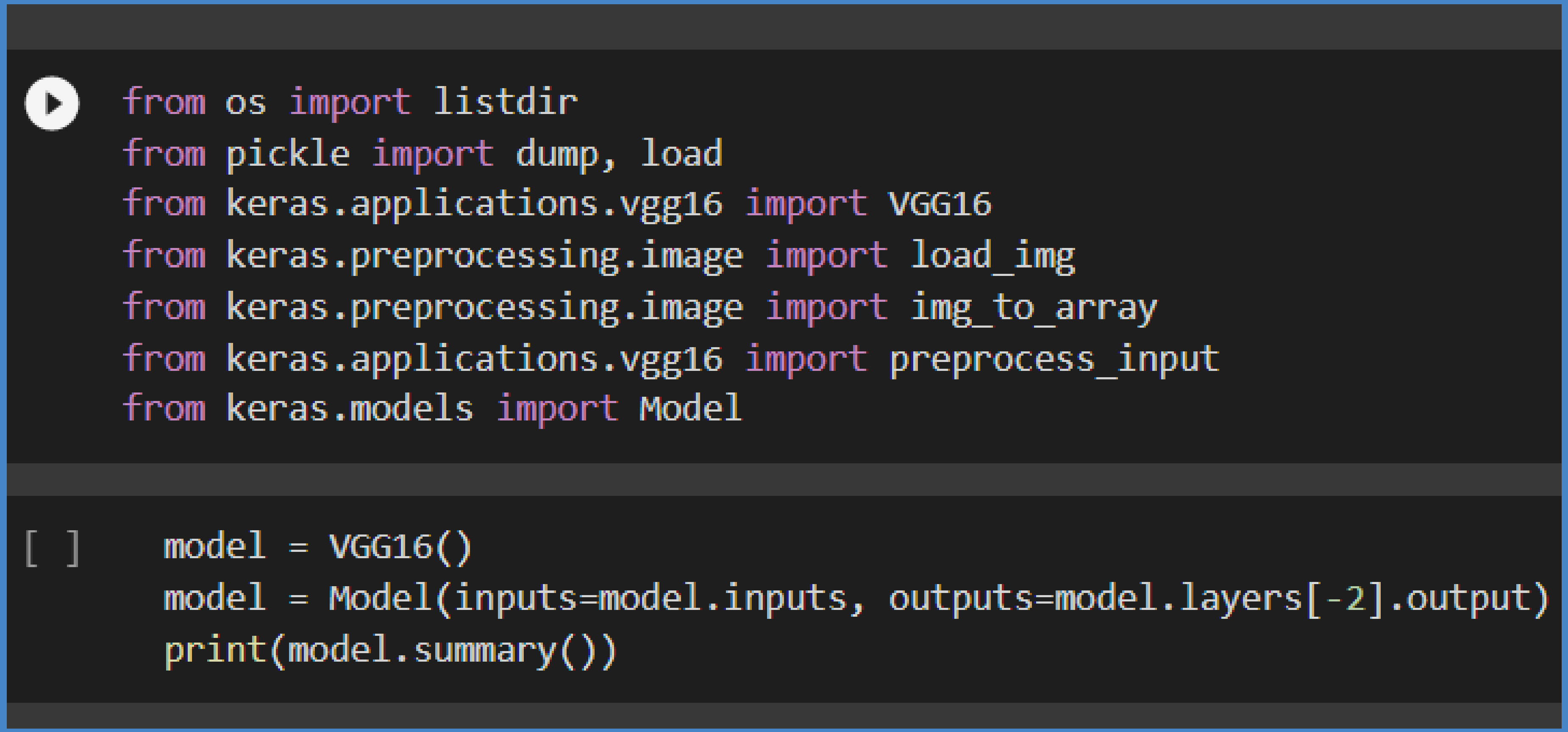
The Sequence model expects input sequences with a length of 25 words which are fed into an Word Embedding layer with 256 Embedding Dimension. It is followed by an GRU layer with 256 memory units.

Both the input models used 50% Dropout Regularization was used to prevent Overfitting, and finally produced a 256 element vector.

Features from both the Input models were merged using an addition layer. This is then fed to another Dense Layer with 256 neurons and then to a final output Dense layer that makes a SoftMax prediction over the entire vocabulary to predict the next word in the sequence.

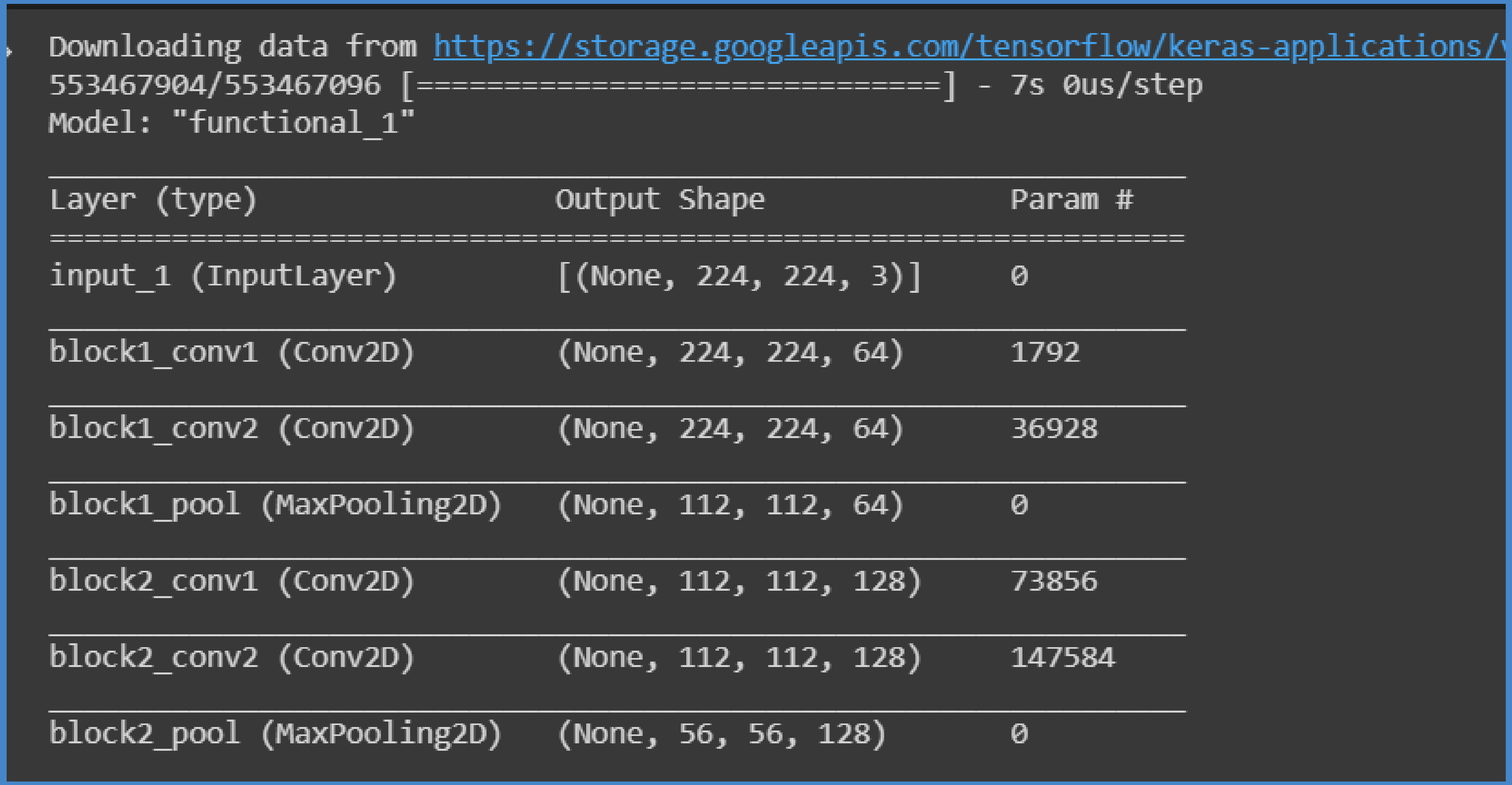
The model will output a prediction, which will be a probability distribution over all words in the vocabulary. The output data will therefore be a one-hot encoded version of each word, representing an idealized probability distribution with 0 values at all word positions except the actual word position, which has a value of 1.



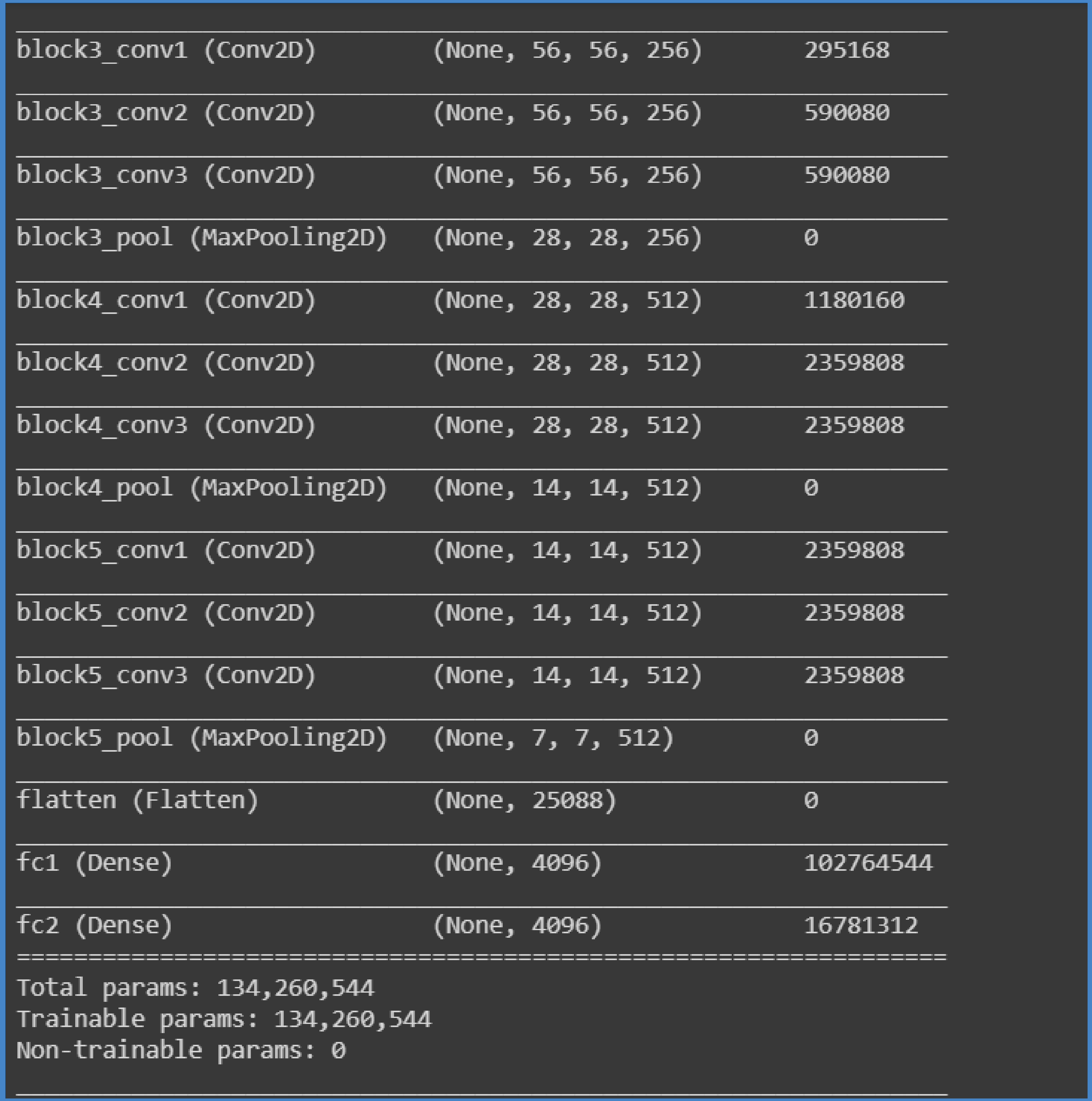


```
from os import listdir
from pickle import dump, load
from keras.applications.vgg16 import VGG16
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.applications.vgg16 import preprocess_input
from keras.models import Model
```

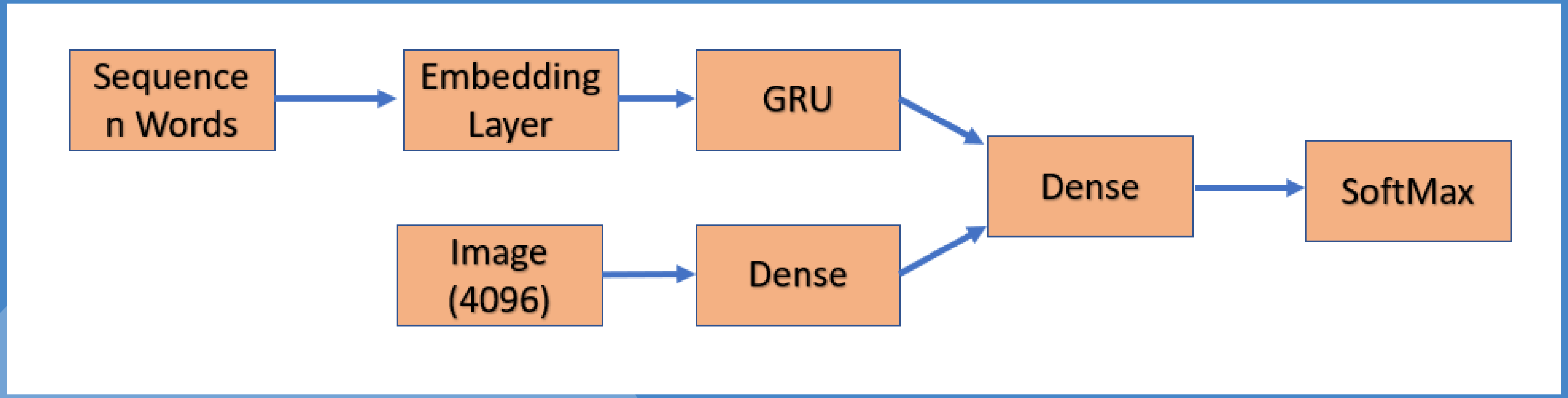
```
[ ]   model = VGG16()
      model = Model(inputs=model.inputs, outputs=model.layers[-2].output)
      print(model.summary())
```

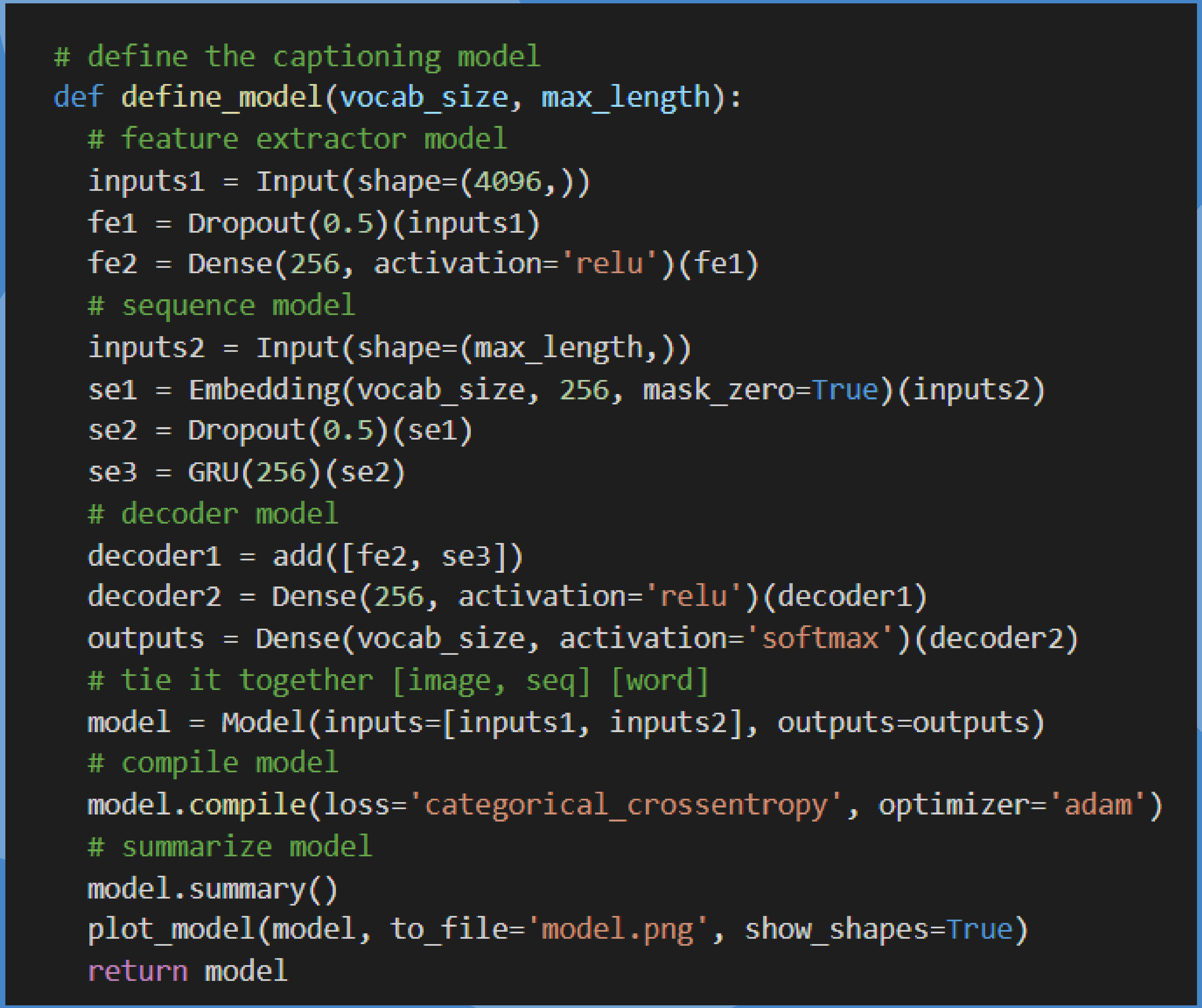


```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/
553467904/553467096 [==============================] - 7s 0us/step
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
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
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
```

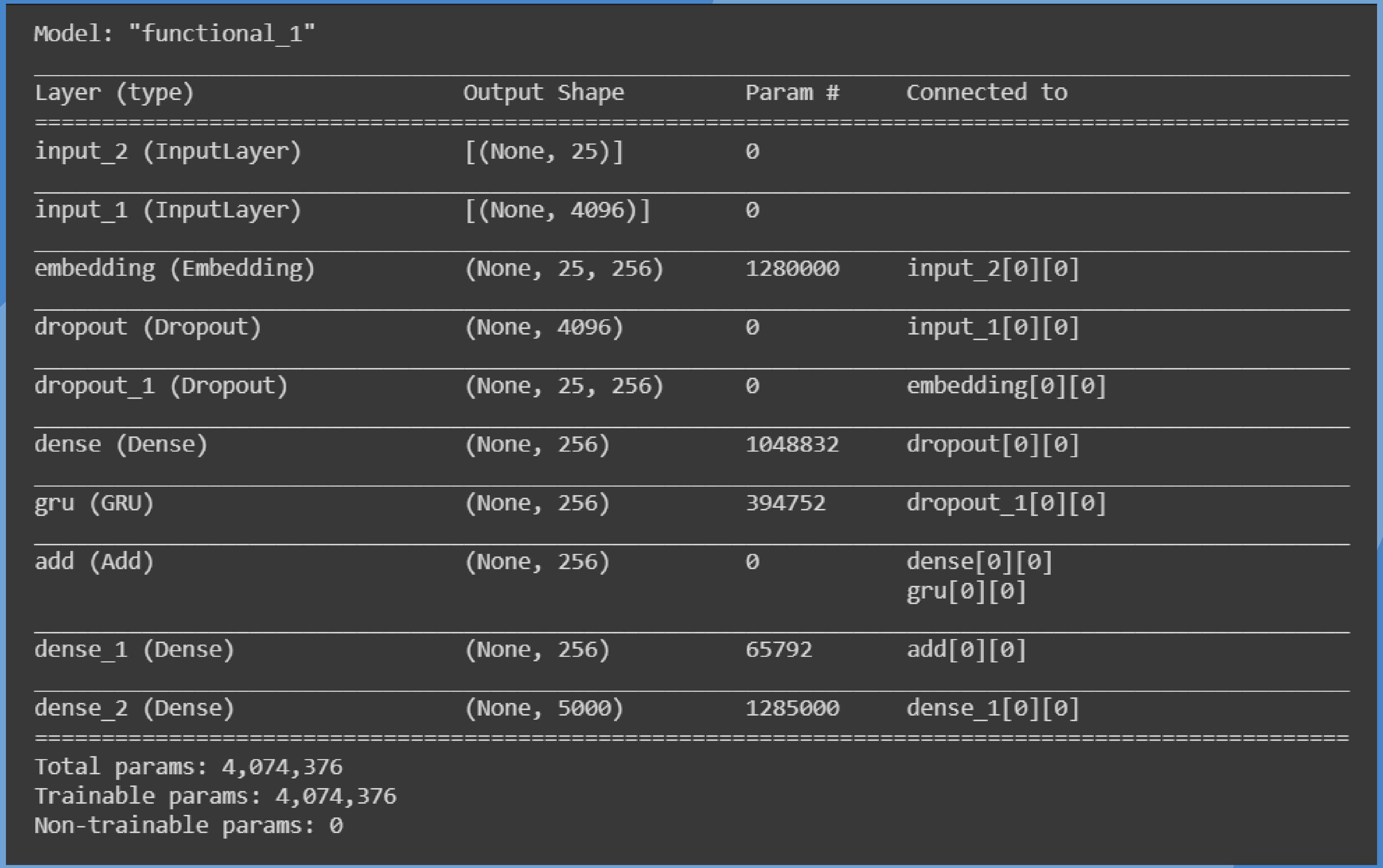


```
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
=================================================================
Total params: 134,260,544
Trainable params: 134,260,544
Non-trainable params: 0
_________________________________________________________________
```





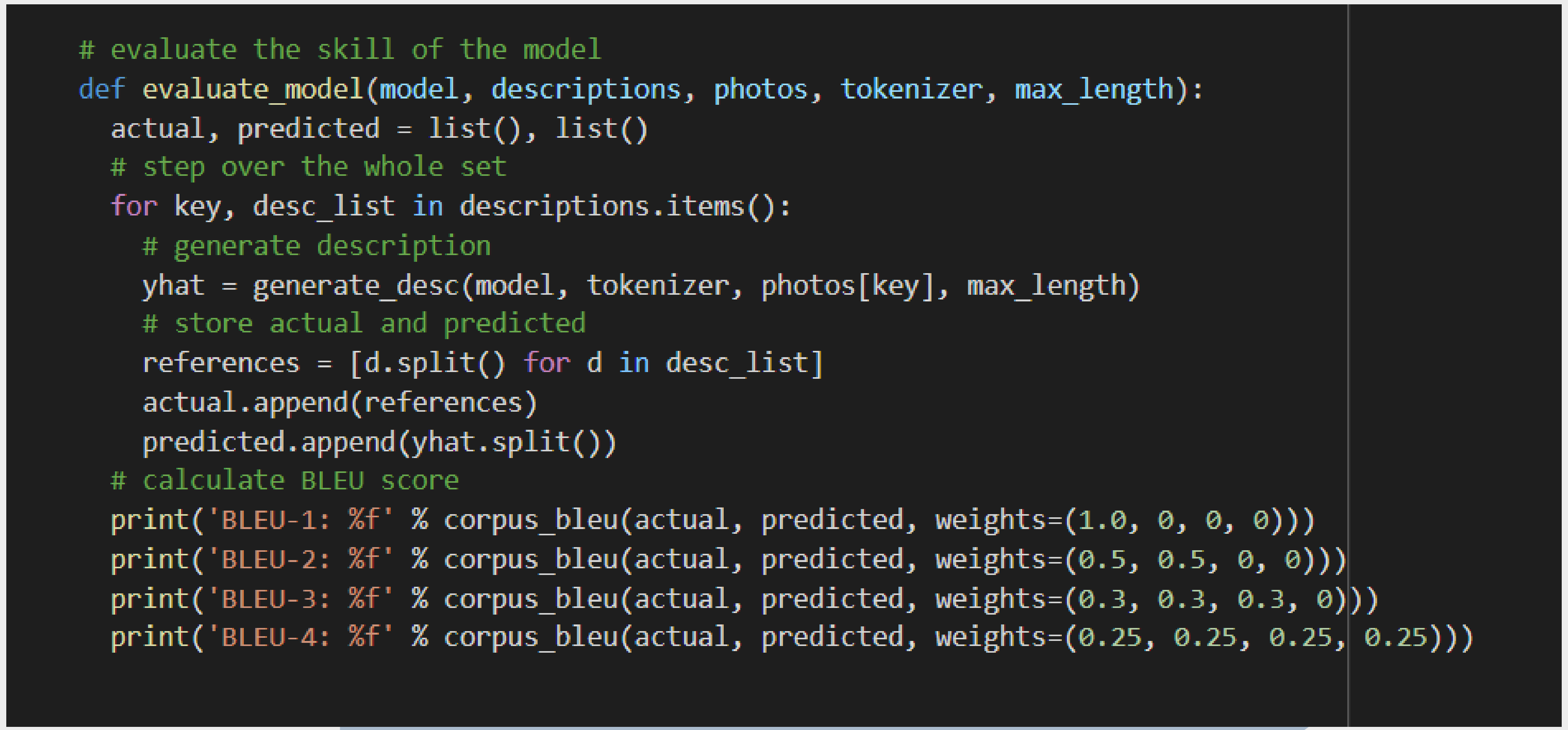
```python
# define the captioning model
def define_model(vocab_size, max_length):
  # feature extractor model
  inputs1 = Input(shape=(4096,))
  fe1 = Dropout(0.5)(inputs1)
  fe2 = Dense(256, activation='relu')(fe1)
  # sequence model
  inputs2 = Input(shape=(max_length,))
  se1 = Embedding(vocab_size, 256, mask_zero=True)(inputs2)
  se2 = Dropout(0.5)(se1)
  se3 = GRU(256)(se2)
  # decoder model
  decoder1 = add([fe2, se3])
  decoder2 = Dense(256, activation='relu')(decoder1)
  outputs = Dense(vocab_size, activation='softmax')(decoder2)
  # tie it together [image, seq] [word]
  model = Model(inputs=[inputs1, inputs2], outputs=outputs)
  # compile model
  model.compile(loss='categorical_crossentropy', optimizer='adam')
  # summarize model
  model.summary()
  plot_model(model, to_file='model.png', show_shapes=True)
  return model
```



```
Model: "functional_1"
________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================================
input_2 (InputLayer)            [(None, 25)]         0
________________________________________________________________________________________________
input_1 (InputLayer)            [(None, 4096)]       0
________________________________________________________________________________________________
embedding (Embedding)           (None, 25, 256)      1280000     input_2[0][0]
________________________________________________________________________________________________
dropout (Dropout)               (None, 4096)         0           input_1[0][0]
________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 25, 256)      0           embedding[0][0]
________________________________________________________________________________________________
dense (Dense)                   (None, 256)          1048832     dropout[0][0]
________________________________________________________________________________________________
gru (GRU)                       (None, 256)          394752      dropout_1[0][0]
________________________________________________________________________________________________
add (Add)                       (None, 256)          0           dense[0][0]
                                                                 gru[0][0]
________________________________________________________________________________________________
dense_1 (Dense)                 (None, 256)          65792       add[0][0]
________________________________________________________________________________________________
dense_2 (Dense)                 (None, 5000)         1285000     dense_1[0][0]
================================================================================================
Total params: 4,074,376
Trainable params: 4,074,376
Non-trainable params: 0
```

Model Evaluation

- We evaluated the model by generating descriptions for all photos in the test dataset and evaluating those predictions using the Bleu Score.
- The model generated a textual description given a trained model, and an image as input.
- The actual and predicted descriptions are collected and evaluated collectively using the BLEU score that summarizes how close the generated text is to the expected text.
- BLEU scores are used in text translation for evaluating translated text against one or more reference translations.
- We then calculate BLEU scores for 1, 2, 3 and 4 cumulative n-grams



```
# evaluate the skill of the model
def evaluate_model(model, descriptions, photos, tokenizer, max_length):
  actual, predicted = list(), list()
  # step over the whole set
  for key, desc_list in descriptions.items():
    # generate description
    yhat = generate_desc(model, tokenizer, photos[key], max_length)
    # store actual and predicted
    references = [d.split() for d in desc_list]
    actual.append(references)
    predicted.append(yhat.split())
  # calculate BLEU score
  print('BLEU-1: %f' % corpus_bleu(actual, predicted, weights=(1.0, 0, 0, 0)))
  print('BLEU-2: %f' % corpus_bleu(actual, predicted, weights=(0.5, 0.5, 0, 0)))
  print('BLEU-3: %f' % corpus_bleu(actual, predicted, weights=(0.3, 0.3, 0.3, 0)))
  print('BLEU-4: %f' % corpus_bleu(actual, predicted, weights=(0.25, 0.25, 0.25, 0.25)))
```

```
BLEU-1: 0.579114
BLEU-2: 0.344856
BLEU-3: 0.252154
BLEU-4: 0.131446
```

Scope of Improvement

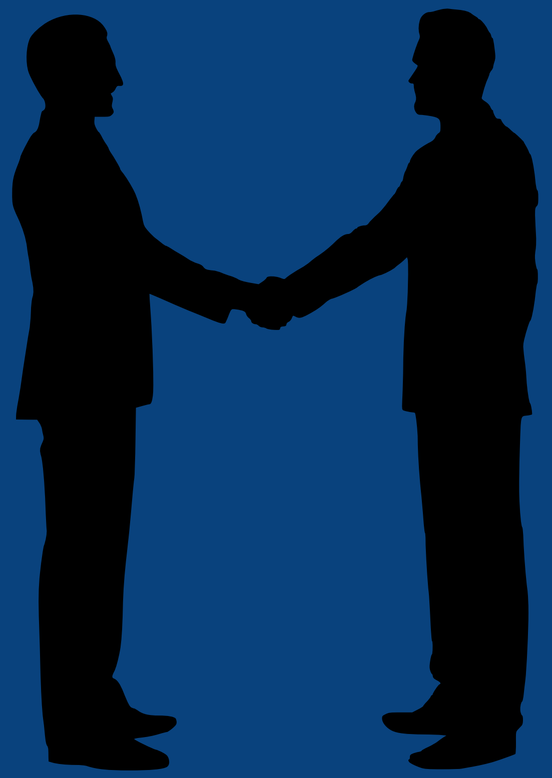
To increase the evaluation metrics for our model we could have used:

- Pre-trained Word2Vec embedding by Google or GloVe Embedding by Pennington (2014) Stanford University, instead of training our own embeddings.
- Other strong language models like using Attention mechanism on the Seq2Seq model.
- We could also have used Transformer models. They aim to solve sequence-to-sequence tasks while handling long-range dependencies with ease.
- Some Popular transformers models are:

  I. BERT by Google, (Bidirectional Encoder Representations from Transformers)
  II. RoBERTa A Robustly Optimized BERT Pretraining Approach, by Facebook
  III. ALBERT A Lite BERT for Self-supervised Learning of Language Representations, by Google
  IV. StructBERT
  V. ELMo
  VI. XLNet by Google

CONCLUSION

- We were able to create a robiust machine learning model using a combined architecture of CNN VGG-16 and GRU. The model was evaluated using the Bleu score upon the Flickr8K dataset that contains (8000 images).
- This model prvoides a dual functionality, first it detects the image and then it further goes onto generate captions for the specific image.



REFERENCES

- https://cs231n.github.io/convolutional-networks
- Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).
- Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv preprint arXiv:1412.3555 (2014
- https://keras.io/api/models/
- https://medium.com/towards-artificial-intelligence/the-architecture-and-implementation-of-vgg-16-b050e5a5920b
- https://neurohive.io/en/popular-networks/vgg16/
- https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be
- https://deepai.org/machine-learning-glossary-and-terms/gated-recurrent-unit
- https://towardsdatascience.com/bleu-bilingual-evaluation-understudy-2b4eab9bcfd1
- https://machinelearningmastery.com/calculate-bleu-score-for-text-python/

THE TEAM

Priyansh Kedia

2K18/EE/146

Dept. of Electrical Engineering

Tanish Deosaria

2K18/EE/219

Dept.of Electrical Engineering