Configuration Manual

MSc Research Project
MSc Data Analytics

Harisankar
Student ID: x22247564

School of Computing
National College of Ireland

Supervisor: Jorge Basilio

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Harisankar		
Student ID:	x22247564		
Programme:	Msc Data Analytics	**Year:**	2024-2025
Module:	Research Project		
Lecturer:	Jorge Basilio		
Submission Due Date:	29-01-2025		
Project Title:	Enhancing image caption quality using an ensemble of deep learning models		
Word Count:	552	**Page Count:**	6

Signature: Harisankar

Date: 29-01-2025

Configuration Manual

Harisankar
Student ID: x22247564

1 Overview

This configuration manual guides on running the project and testing the results of established model for the research project “Enhancing image caption quality using an ensemble of Deep Learning models”.

2 System requirements

2.1 Hardware Requirements

- RAM – 8GB or more
- Storage – 500 GB SDD or HDD with 50gb available space
- Processor – Intel i7 or AMD Ryzen7 or equivalent
- GPU – 4GB or more

2.2 Software requirements

- Programming language – Python 3
- IDE – Jupyter notebook, VS code or Google colab
- Operating system – Windows 11, MacOS or Linux
- Browser – Google chrome

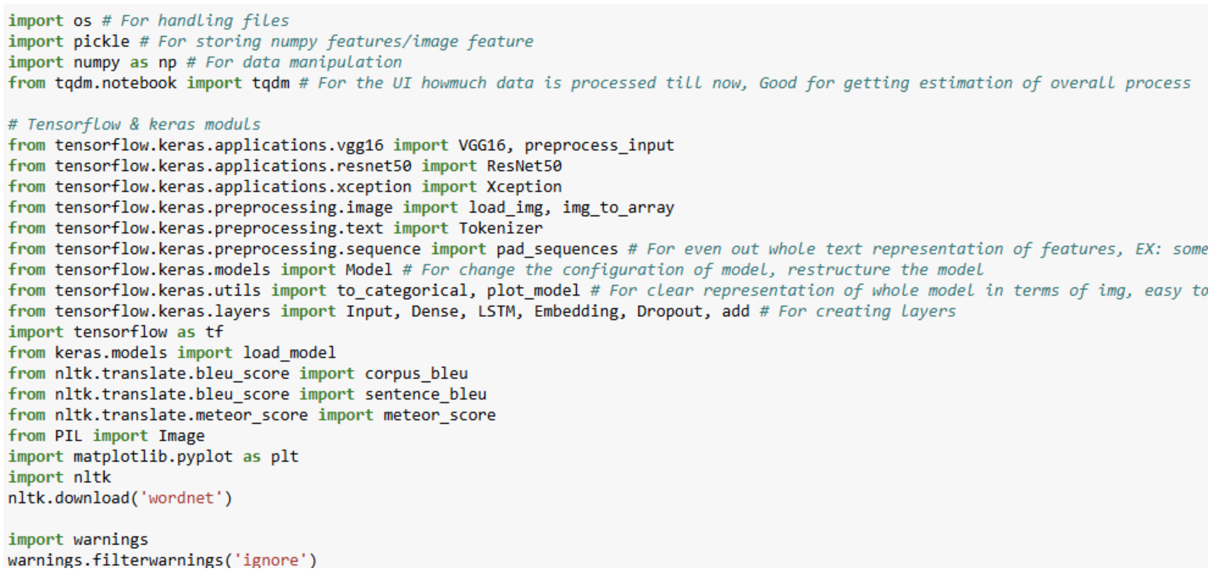
3 Installed versions

- Python 3.11.2
- Numpy 1.26.0
- Tqdm 4.64.1
- Tensorflow 2.18.0
- Keras 3.6.0
- Nltk 3.7
- Matplotlib 3.7.0
- Pillow 9.4.0

```
!pip install numpy
!pip install tqdm
!pip install tensorflow
!pip install keras
!pip install nltk
```

!pip install matplotlin
!pip install pillow

The given pip commands can be used to install the packages required for the project. The required libraries and its functionalities are imported in the beginning of the project.



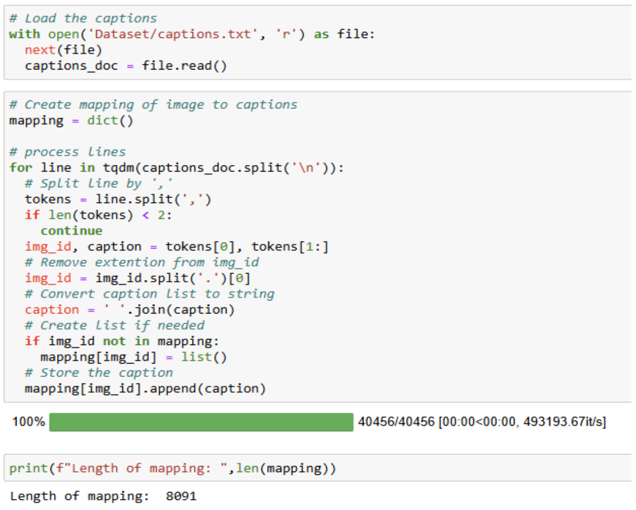
```
import os # For handling files
import pickle # For storing numpy features/image feature
import numpy as np # For data manipulation
from tqdm.notebook import tqdm # For the UI howmuch data is processed till now, Good for getting estimation of overall process

# Tensorflow & keras moduls
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.xception import Xception
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences # For even out whole text representation of features, EX: some
from tensorflow.keras.models import Model # For change the configuration of model, restructure the model
from tensorflow.keras.utils import to_categorical, plot_model # For clear representation of whole model in terms of img, easy to
from tensorflow.keras.layers import Input, Dense, LSTM, Embedding, Dropout, add # For creating layers
import tensorflow as tf
from keras.models import load_model
from nltk.translate.bleu_score import corpus_bleu
from nltk.translate.bleu_score import sentence_bleu
from nltk.translate.meteor_score import meteor_score
from PIL import Image
import matplotlib.pyplot as plt
import nltk
nltk.download('wordnet')

import warnings
warnings.filterwarnings('ignore')
```

4 Dataset

The dataset was downloaded from Kaggle using the link:
https://www.kaggle.com/datasets/adityajn105/flickr8k?resource=download
After downloading the dataset, which will be a zipped file, the zip is extracted to get an Image folder which contains 8091 images. A caption.txt file is also available which contains the captions corresponding to the image id/name. The unzipped file can be stored in the working directory of code.

5 Preprocessing

The captions are loaded from captions.txt file. The captions are then mapped into a dictionary along with its image id and captions.



```
# Load the captions
with open('Dataset/captions.txt', 'r') as file:
  next(file)
  captions_doc = file.read()
```

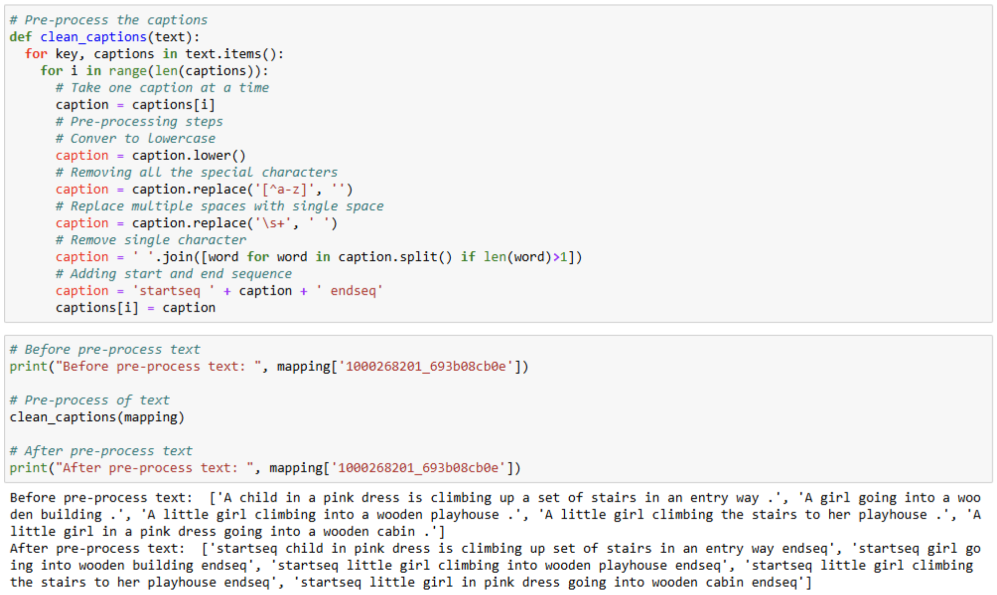
```
# Create mapping of image to captions
mapping = dict()

# process lines
for line in tqdm(captions_doc.split('\n')):
  # Split line by ','
  tokens = line.split(',')
  if len(tokens) < 2:
    continue
  img_id, caption = tokens[0], tokens[1:]
  # Remove extention from img_id
  img_id = img_id.split('.')[0]
  # Convert caption list to string
  caption = ' '.join(caption)
  # Create list if needed
  if img_id not in mapping:
    mapping[img_id] = list()
  # Store the caption
  mapping[img_id].append(caption)
```

```
100% 40456/40456 [00:00<00:00, 493193.67it/s]
```

```
print(f"Length of mapping: ",len(mapping))
```

```
Length of mapping:  8091
```

The captions are then cleaned using clean_captions function where the special characters, and extra spaces are removed. All the characters are changed to lower case and a token startseq and endseq is added at the beginning and end of each caption.



```
# Pre-process the captions
def clean_captions(text):
  for key, captions in text.items():
    for i in range(len(captions)):
      # Take one caption at a time
      caption = captions[i]
      # Pre-processing steps
      # Conver to Lowercase
      caption = caption.lower()
      # Removing all the special characters
      caption = caption.replace('[^a-z]', '')
      # Replace multiple spaces with single space
      caption = caption.replace('\s+', ' ')
      # Remove single character
      caption = ' '.join([word for word in caption.split() if len(word)>1])
      # Adding start and end sequence
      caption = 'startseq ' + caption + ' endseq'
      captions[i] = caption
```

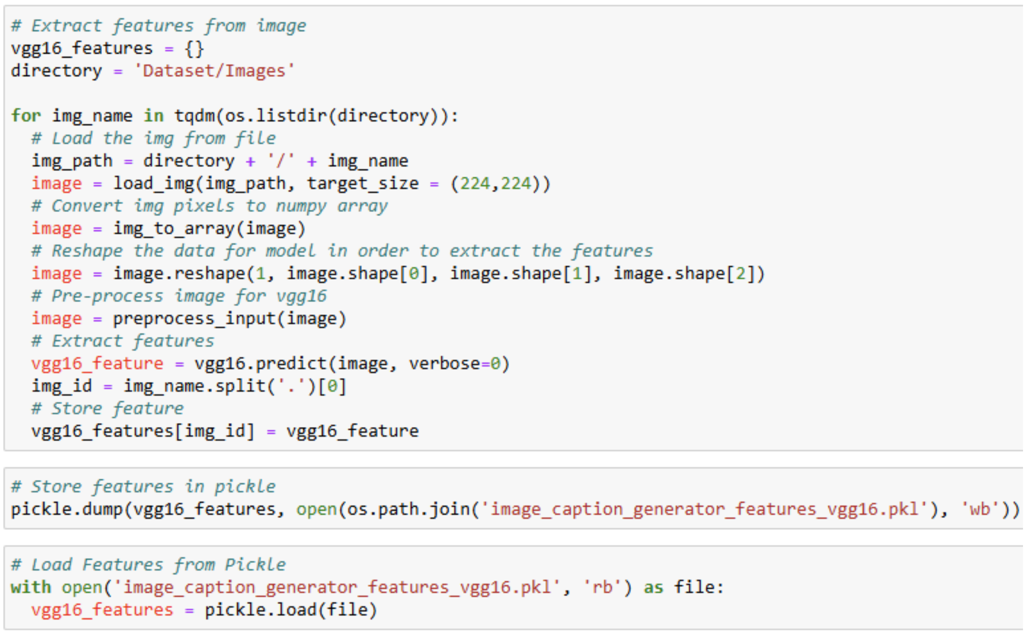
```
# Before pre-process text
print("Before pre-process text: ", mapping['1000268201_693b08cb0e'])

# Pre-process of text
clean_captions(mapping)

# After pre-process text
print("After pre-process text: ", mapping['1000268201_693b08cb0e'])
```

```
Before pre-process text:  ['A child in a pink dress is climbing up a set of stairs in an entry way .', 'A girl going into a woo
den 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 .']
After pre-process text:  ['startseq child in pink dress is climbing up set of stairs in an entry way endseq', 'startseq girl go
ing 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']
```

Tokenizer is used to represent the words as integers. The dataset is then split into 90 percent for training and 10 percent for testing.

6 Model

Pre-trained models VGG16, ResNet50 and Xception are used for the model. The feature extraction process which extracts the features in each images is time consuming and would take 1-2 hours depending on the system. To save time for running the code, the extracted features are stored as pickle file from all the three models. The extracted features in pickle file can be used directly for using the features in all models respectively.

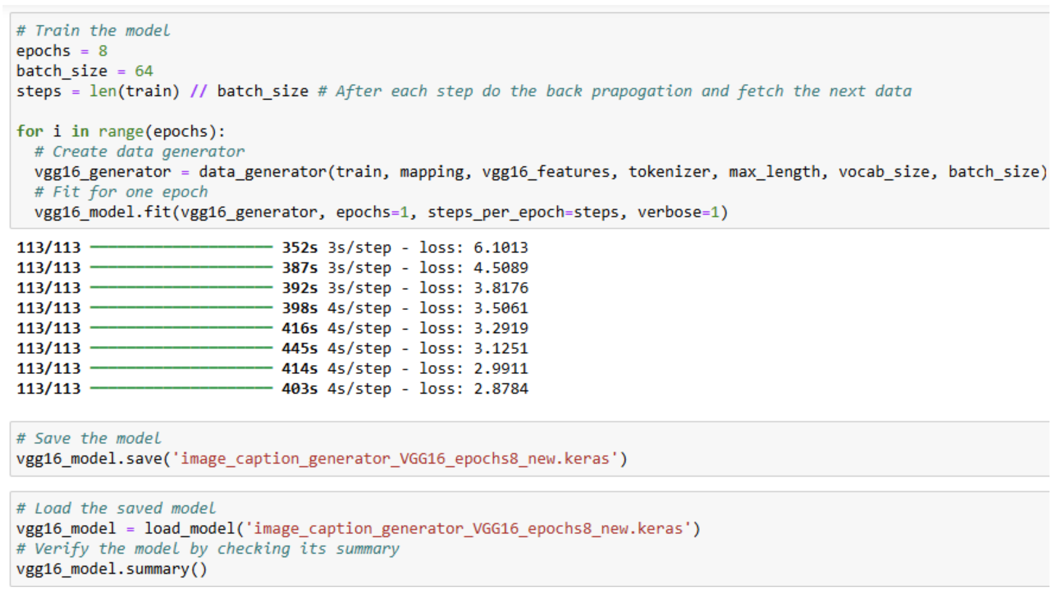


```
# Extract features from image
vgg16_features = {}
directory = 'Dataset/Images'

for img_name in tqdm(os.listdir(directory)):
  # Load the img from file
  img_path = directory + '/' + img_name
  image = load_img(img_path, target_size = (224,224))
  # Convert img pixels to numpy array
  image = img_to_array(image)
  # Reshape the data for model in order to extract the features
  image = image.reshape(1, image.shape[0], image.shape[1], image.shape[2])
  # Pre-process image for vgg16
  image = preprocess_input(image)
  # Extract features
  vgg16_feature = vgg16.predict(image, verbose=0)
  img_id = img_name.split('.')[0]
  # Store feature
  vgg16_features[img_id] = vgg16_feature
```

```
# Store features in pickle
pickle.dump(vgg16_features, open(os.path.join('image_caption_generator_features_vgg16.pkl'), 'wb'))
```

```
# Load Features from Pickle
with open('image_caption_generator_features_vgg16.pkl', 'rb') as file:
  vgg16_features = pickle.load(file)
```

The CNN-LSTM model is trained for 8 epochs. Data_generator function is used to train the model in batches of size 64. The trained model is saved as keras file, which can be used to load the trained model to save the code running time.



```
# Train the model
epochs = 8
batch_size = 64
steps = len(train) // batch_size # After each step do the back prapogation and fetch the next data

for i in range(epochs):
  # Create data generator
  vgg16_generator = data_generator(train, mapping, vgg16_features, tokenizer, max_length, vocab_size, batch_size)
  # Fit for one epoch
  vgg16_model.fit(vgg16_generator, epochs=1, steps_per_epoch=steps, verbose=1)
```

```
113/113 ━━━━━━━━━━━━━━━━━━━━ 352s 3s/step - loss: 6.1013
113/113 ━━━━━━━━━━━━━━━━━━━━ 387s 3s/step - loss: 4.5089
113/113 ━━━━━━━━━━━━━━━━━━━━ 392s 3s/step - loss: 3.8176
113/113 ━━━━━━━━━━━━━━━━━━━━ 398s 4s/step - loss: 3.5061
113/113 ━━━━━━━━━━━━━━━━━━━━ 416s 4s/step - loss: 3.2919
113/113 ━━━━━━━━━━━━━━━━━━━━ 445s 4s/step - loss: 3.1251
113/113 ━━━━━━━━━━━━━━━━━━━━ 414s 4s/step - loss: 2.9911
113/113 ━━━━━━━━━━━━━━━━━━━━ 403s 4s/step - loss: 2.8784
```

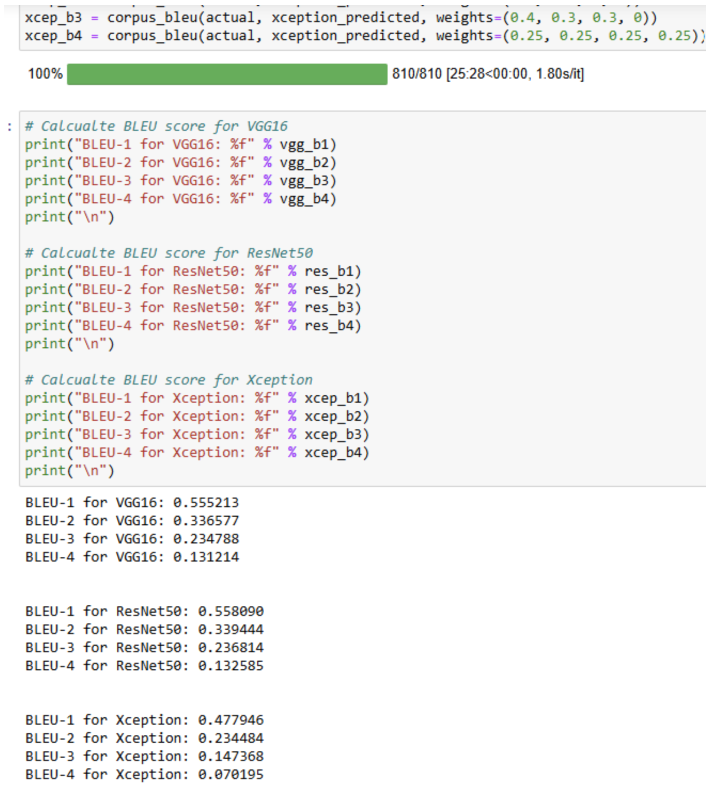
```
# Save the model
vgg16_model.save('image_caption_generator_VGG16_epochs8_new.keras')
```

```
# Load the saved model
vgg16_model = load_model('image_caption_generator_VGG16_epochs8_new.keras')
# Verify the model by checking its summary
vgg16_model.summary()
```

Using the pickle file and keras file saved for each model, these can be used for easiness when rerunning the files.

7 Evaluation and Testing

The models are evaluated using BLEU scores and METEOR scores which are imported from nltk library.



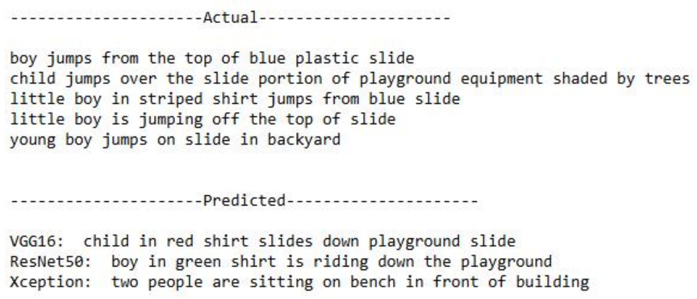
```
xcep_b3 = corpus_bleu(actual, xception_predicted, weights=(0.4, 0.3, 0.3, 0))
xcep_b4 = corpus_bleu(actual, xception_predicted, weights=(0.25, 0.25, 0.25, 0.25))
```

```
100% 810/810 [25:28<00:00, 1.80s/it]
```

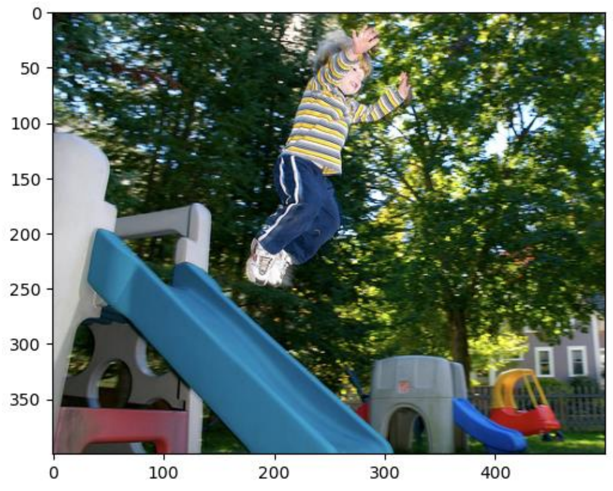
```
# Calcualte BLEU score for VGG16
print("BLEU-1 for VGG16: %f" % vgg_b1)
print("BLEU-2 for VGG16: %f" % vgg_b2)
print("BLEU-3 for VGG16: %f" % vgg_b3)
print("BLEU-4 for VGG16: %f" % vgg_b4)
print("\n")

# Calcualte BLEU score for ResNet50
print("BLEU-1 for ResNet50: %f" % res_b1)
print("BLEU-2 for ResNet50: %f" % res_b2)
print("BLEU-3 for ResNet50: %f" % res_b3)
print("BLEU-4 for ResNet50: %f" % res_b4)
print("\n")

# Calcualte BLEU score for Xception
print("BLEU-1 for Xception: %f" % xcep_b1)
print("BLEU-2 for Xception: %f" % xcep_b2)
print("BLEU-3 for Xception: %f" % xcep_b3)
print("BLEU-4 for Xception: %f" % xcep_b4)
print("\n")
```

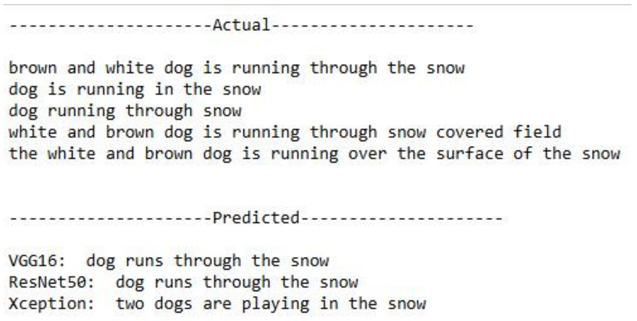
```
BLEU-1 for VGG16: 0.555213
BLEU-2 for VGG16: 0.336577
BLEU-3 for VGG16: 0.234788
BLEU-4 for VGG16: 0.131214


BLEU-1 for ResNet50: 0.558090
BLEU-2 for ResNet50: 0.339444
BLEU-3 for ResNet50: 0.236814
BLEU-4 for ResNet50: 0.132585


BLEU-1 for Xception: 0.477946
BLEU-2 for Xception: 0.234484
BLEU-3 for Xception: 0.147368
BLEU-4 for Xception: 0.070195
```

Some of the captions generated from the models are given below.

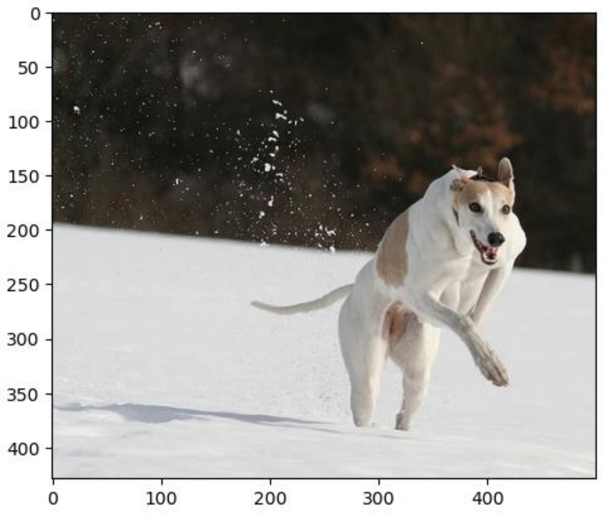


```
---------------------Actual--------------------

boy jumps from the top of blue plastic slide
child jumps over the slide portion of playground equipment shaded by trees
little boy in striped shirt jumps from blue slide
little boy is jumping off the top of slide
young boy jumps on slide in backyard


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

VGG16:  child in red shirt slides down playground slide
ResNet50:  boy in green shirt is riding down the playground
Xception:  two people are sitting on bench in front of building
```



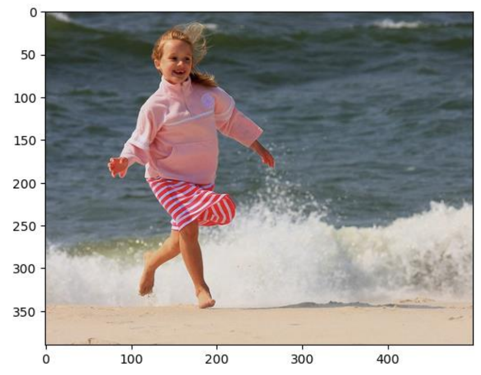


```
---------------------Actual--------------------

brown and white dog is running through the snow
dog is running in the snow
dog running through snow
white and brown dog is running through snow covered field
the white and brown dog is running over the surface of the snow


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

VGG16:  dog runs through the snow
ResNet50:  dog runs through the snow
Xception:  two dogs are playing in the snow
```



Bagging and boosting techniques are used to combine the base models. The base models and the extracted features from base models are passed as a list to the ensemble models. The ensemble technique then chooses the most common word for making predictions in bagging. For boosting method, the predictions is made by assigning weights for each methods and calculating its BLEU scores. The predictions of Bagging and boosting ensemble models are given below.

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

girl dressed in pink sweatshirt and pink and white striped skirt plays in the waves at the beach
girl dressed in pink and white runs along the beach
girl dressed in pink runs from the ocean to the beach
girl in pink runs away from wave coming onto the shore
girl is walking next to the small waves that crash in the ocean


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

Bagging:  young dogs pink playing is the pink the is running on the


Boosting:  young girl in red shirt and pink shorts is running on the beach
```



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

climber stops to take drink while climbing snow covered mountain
man holding cup on snow mountain
man in yellow suit is holding up cup while standing in snow
mountain climber stops for drink
mountaineer in yellow jacket is drinking from thermos cup


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

Bagging:  two children are skiing is snowy in front of an orange


Boosting:  two people are skiing on snowy ice
```