```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
!pip install PyWavelets
import cv2
import pywt  # Correct import for PyWavelets
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
```

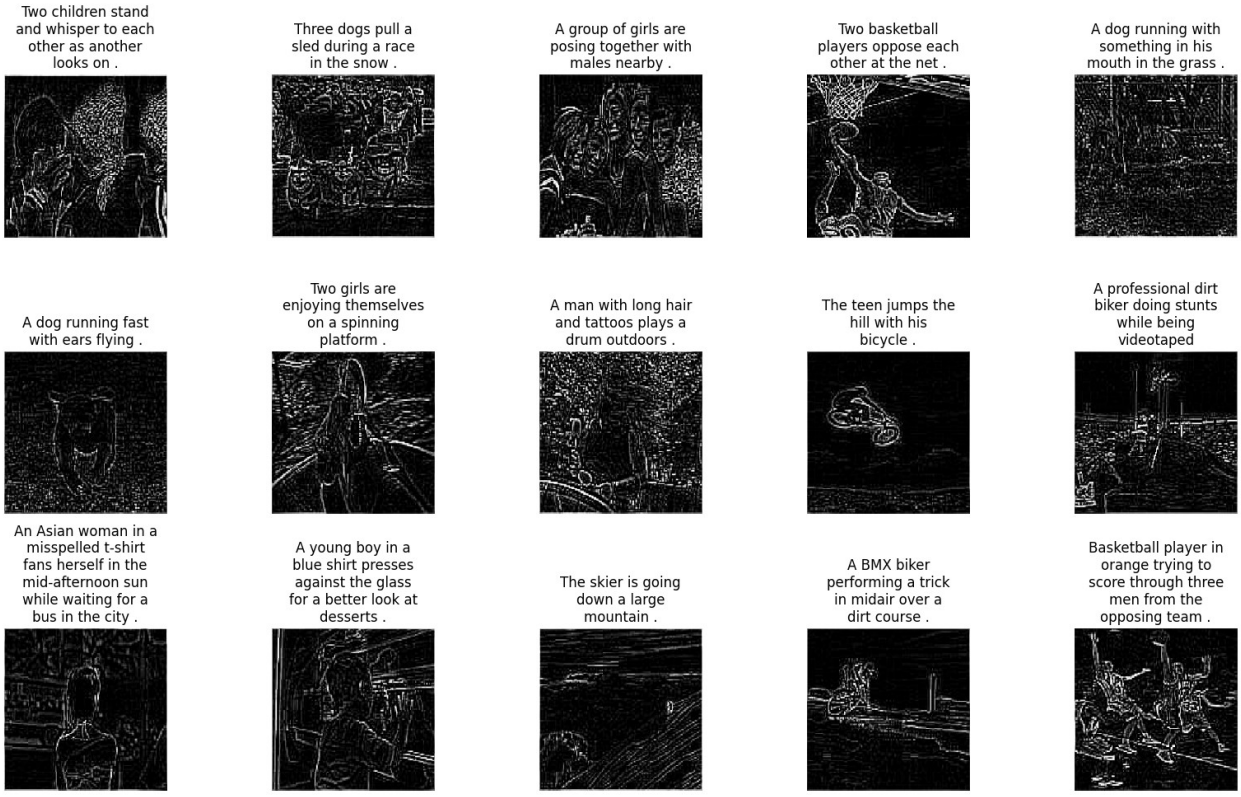
```
Collecting PyWavelets
  Downloading pywavelets-1.7.0-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (9.0 kB)
Requirement already satisfied: numpy<3,>=1.23 in
/usr/local/lib/python3.10/dist-packages (from PyWavelets) (1.26.4)
Downloading pywavelets-1.7.0-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (4.5 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0.0/4.5 MB ? eta -:--:--
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.5/4.5 MB 139.9 MB/s eta
0:00:01 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.5/4.5 MB 84.3 MB/s
eta 0:00:00
```

```
import pandas as pd
data = pd.read_csv('/content/drive/MyDrive/captions.txt')
data.head()
```

```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 40455,\n
\"fields\": [\n    {\n      \"column\": \"image\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 8091,\n        \"samples\": [\n
\"3139895886_5a6d495b13.jpg\",\n
\"3133825703_359a0c414d.jpg\",\n
\"244910177_7c4ec3f65b.jpg\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"caption\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 40201,\n
\"samples\": [\n          \"A girl plays T-ball .\",\n          \"A
woman in riding attire rides a jumping horse .\",\n          \"A brown
dog wearing a pink shirt is followed by a brown dog wearing a yellow
shirt .\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"data"}
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator,
load_img, img_to_array
from textwrap import wrap

def readImage(path,img_size=224):
    img =
```

```
load_img(path,color_mode='rgb',target_size=(img_size,img_size))
    img = img_to_array(img)
    img = img/255.

    return img

def display_images(temp_df):
    temp_df = temp_df.reset_index(drop=True)
    plt.figure(figsize = (20 , 20))
    n = 0
    for i in range(15):
        n+=1
        plt.subplot(5 , 5, n)
        plt.subplots_adjust(hspace = 0.7, wspace = 0.3)
        image =
readImage(f"/content/drive/MyDrive/FilteredImages/{temp_df.image[i]}")
        plt.imshow(image)
        plt.title("\n".join(wrap(temp_df.caption[i], 20)))
        plt.axis("off")

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




```
def text_preprocessing(data):
    data['caption'] = data['caption'].apply(lambda x: x.lower())
    data['caption'] = data['caption'].apply(lambda x: x.replace("[^A-
```

```
Za-z]",""))
    data['caption'] = data['caption'].apply(lambda x: x.replace("\
s+"," "))
    data['caption'] = data['caption'].apply(lambda x: " ".join([word
for word in x.split() if len(word)>1]))
    data['caption'] = "startseq "+data['caption']+" endseq"
    return data

data = text_preprocessing(data)
captions = data['caption'].tolist()
captions[:5]
```

```
['startseq child in pink dress is climbing up set of stairs in an
entry way endseq',
 'startseq girl going 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']
```

```
from tensorflow.keras.preprocessing.text import Tokenizer

tokenizer = Tokenizer()
tokenizer.fit_on_texts(captions)
vocab_size = len(tokenizer.word_index) + 1
max_length = max(len(caption.split()) for caption in captions)

images = data['image'].unique().tolist()
nimages = len(images)

split_index = round(0.85*nimages)
train_images = images[:split_index]
val_images = images[split_index:]

train = data[data['image'].isin(train_images)]
test = data[data['image'].isin(val_images)]

train.reset_index(inplace=True,drop=True)
test.reset_index(inplace=True,drop=True)

tokenizer.texts_to_sequences([captions[1]])[0]
```

```
[1, 18, 315, 63, 195, 116, 2]
```

```
image_path = '/content/drive/MyDrive/FilteredImages'

from tensorflow.keras.applications import DenseNet201
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import load_img,
img_to_array
import numpy as np
import os
```

```
from tqdm import tqdm

# Load the DenseNet201 model
model = DenseNet201()

# Create a new model that outputs features from the second-to-last
layer
fe = Model(inputs=model.input, outputs=model.layers[-2].output)

img_size = 224
features = {}

# Assuming 'data' is a DataFrame that contains an 'image' column
for image in tqdm(data['image'].unique().tolist()):
    img = load_img(os.path.join(image_path, image),
target_size=(img_size, img_size))
    img = img_to_array(img)
    img = img / 255.0
    img = np.expand_dims(img, axis=0)

    # Extract features using the new feature extractor model
    feature = fe.predict(img, verbose=0)
    features[image] = feature
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/densenet/
densenet201_weights_tf_dim_ordering_tf_kernels.h5
82524592/82524592 ━━━━━━━━━━━━━━━━━━━━ 6s 0us/step

100%|██████████| 8091/8091 [1:36:05<00:00,  1.40it/s]
```

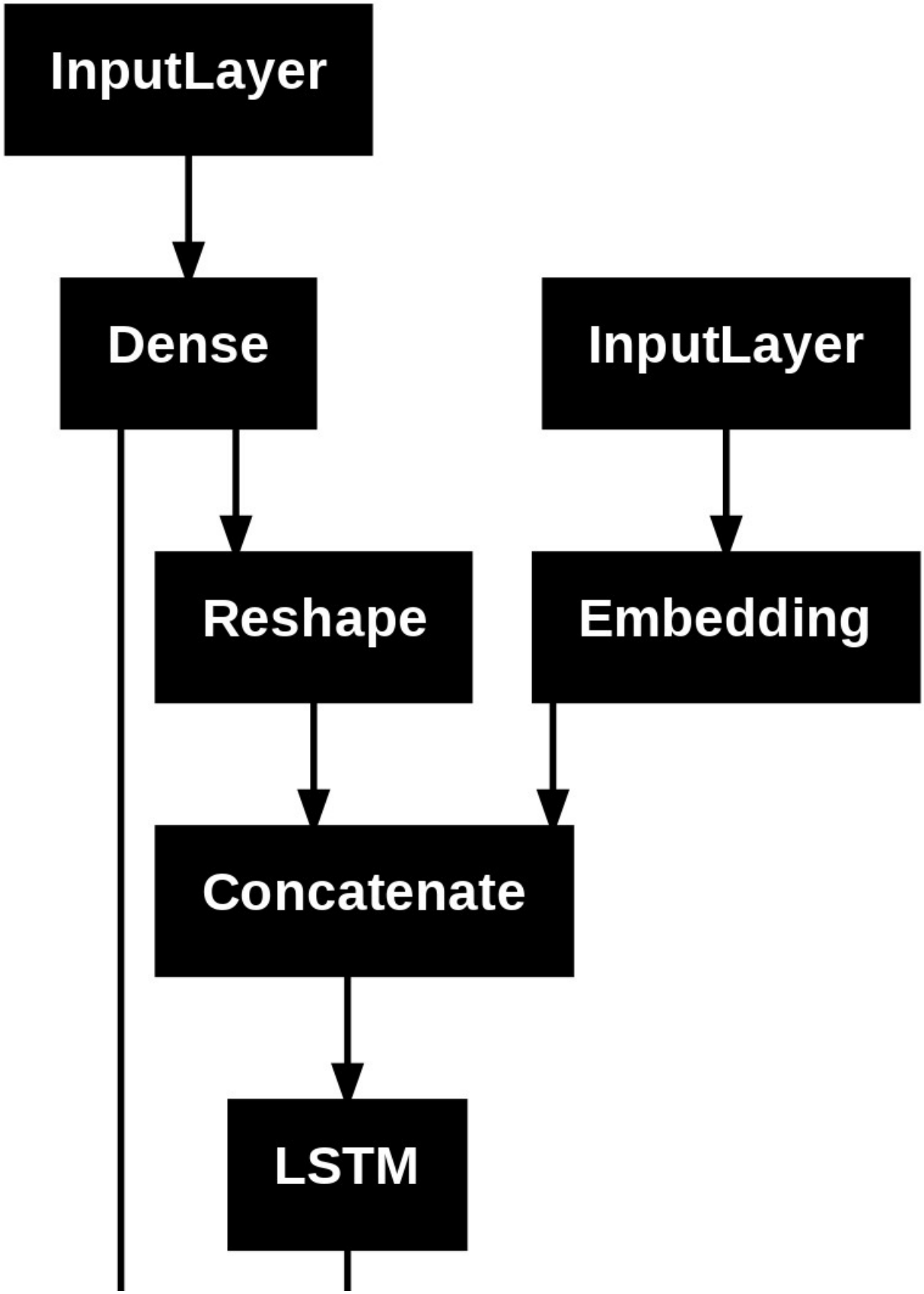
```
from tensorflow.keras.utils import Sequence

class CustomDataGenerator(Sequence):

    def __init__(self, df, X_col, y_col, batch_size, directory,
tokenizer,
                 vocab_size, max_length, features,shuffle=True):

        self.df = df.copy()
        self.X_col = X_col
        self.y_col = y_col
        self.directory = directory
        self.batch_size = batch_size
        self.tokenizer = tokenizer
        self.vocab_size = vocab_size
        self.max_length = max_length
        self.features = features
        self.shuffle = shuffle
        self.n = len(self.df)
```

```
    def on_epoch_end(self):
        if self.shuffle:
            self.df = self.df.sample(frac=1).reset_index(drop=True)

    def __len__(self):
        return self.n // self.batch_size

    def __getitem__(self,index):

        batch = self.df.iloc[index * self.batch_size:(index + 1) *
self.batch_size,:]
        X1, X2, y = self.__get_data(batch)
        return (X1, X2), y

    def __get_data(self,batch):

        X1, X2, y = list(), list(), list()

        images = batch[self.X_col].tolist()

        for image in images:
            feature = self.features[image][0]

            captions = batch.loc[batch[self.X_col]==image,
self.y_col].tolist()
            for caption in captions:
                seq = self.tokenizer.texts_to_sequences([caption])[0]

                for i in range(1,len(seq)):
                    in_seq, out_seq = seq[:i], seq[i]
                    in_seq = pad_sequences([in_seq],
maxlen=self.max_length)[0]
                    out_seq = to_categorical([out_seq],
num_classes=self.vocab_size)[0]
                    X1.append(feature)
                    X2.append(in_seq)
                    y.append(out_seq)

        X1, X2, y = np.array(X1), np.array(X2), np.array(y)

        return X1, X2, y

import numpy as np
import pandas as pd
import os
import tensorflow as tf
from tqdm import tqdm
from tensorflow.keras.preprocessing.image import ImageDataGenerator,
load_img, img_to_array
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

```
from tensorflow.keras.utils import Sequence
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Conv2D, MaxPooling2D,
GlobalAveragePooling2D, Activation, Dropout, Flatten, Dense, Input,
Layer
from tensorflow.keras.layers import Embedding, LSTM, add, Concatenate,
Reshape, concatenate, Bidirectional
from tensorflow.keras.applications import VGG16, ResNet50, DenseNet201
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,
ReduceLROnPlateau
import warnings
import matplotlib.pyplot as plt
import seaborn as sns
from textwrap import wrap

from tensorflow.keras.layers import Conv2D, MaxPooling2D,
GlobalAveragePooling2D, Activation, Dropout, Flatten, Dense, Input,
Layer
input1 = Input(shape=(1920,))
input2 = Input(shape=(max_length,))

img_features = Dense(256, activation='relu')(input1)
img_features_reshaped = Reshape((1, 256), input_shape=(256,))
(img_features)

sentence_features = Embedding(vocab_size, 256, mask_zero=False)
(input2)
merged = concatenate([img_features_reshaped,sentence_features],axis=1)
sentence_features = LSTM(256)(merged)
x = Dropout(0.5)(sentence_features)
x = add([x, img_features])
x = Dense(128, activation='relu')(x)
x = Dropout(0.5)(x)
output = Dense(vocab_size, activation='softmax')(x)

caption_model = Model(inputs=[input1,input2], outputs=output)
caption_model.compile(loss='categorical_crossentropy',optimizer='adam'
)
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/reshaping/
reshape.py:39: UserWarning: Do not pass an `input_shape`/`input_dim`
argument to a layer. When using Sequential models, prefer using an
`Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

```
from tensorflow.keras.utils import plot_model

plot_model(caption_model)
```


InputLayer
Dense
InputLayer
Reshape
Embedding
Concatenate
LSTM

```
caption_model.summary()
```

```
Model: "functional_1"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┃ Connected to                    ┃                        ┃               ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ input_layer_1                   │ (None, 1920)           │             0 │
│ -                               │                        │               │
│ (InputLayer)                    │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 256)            │       491,776 │
│ input_layer_1[0][0]             │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ input_layer_2                   │ (None, 34)             │             0 │
│ -                               │                        │               │
│ (InputLayer)                    │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ reshape (Reshape)               │ (None, 1, 256)         │             0 │
│ dense[0][0]                     │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ embedding (Embedding)           │ (None, 34, 256)        │     2,172,160 │
│ input_layer_2[0][0]             │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ concatenate (Concatenate)       │ (None, 35, 256)        │             0 │
│ reshape[0][0],                  │                        │               │
│ embedding[0][0]                 │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ lstm (LSTM)                     │ (None, 256)            │       525,312 │
│ concatenate[0][0]               │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)               │ (None, 256)            │             0 │
│ lstm[0][0]                      │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ add (Add)                       │ (None, 256)            │             0 │
│ dropout[0][0],                  │                        │               │
```

```
│                                           │                            │
  dense[0][0]                             │
├───────────────────────────────────────────┼────────────────────────────┼───────────────
┤                                         ┤
  dense_1 (Dense)                           │ (None, 128)                │         32,896
  add[0][0]                               │
├───────────────────────────────────────────┼────────────────────────────┼───────────────
┤                                         ┤
  dropout_1 (Dropout)                       │ (None, 128)                │              0
  dense_1[0][0]                           │
├───────────────────────────────────────────┼────────────────────────────┼───────────────
┤                                         ┤
  dense_2 (Dense)                           │ (None, 8485)               │      1,094,565
  dropout_1[0][0]                         │
└───────────────────────────────────────────┴────────────────────────────┴───────────────
┘                                         ┘

 Total params: 4,316,709 (16.47 MB)

 Trainable params: 4,316,709 (16.47 MB)

 Non-trainable params: 0 (0.00 B)

train_generator = 
CustomDataGenerator(df=train,X_col='image',y_col='caption',batch_size=
64,directory=image_path,
                                      
tokenizer=tokenizer,vocab_size=vocab_size,max_length=max_length,featur
es=features)

validation_generator = 
CustomDataGenerator(df=test,X_col='image',y_col='caption',batch_size=6
4,directory=image_path,
                                      
tokenizer=tokenizer,vocab_size=vocab_size,max_length=max_length,featur
es=features)

model_name = "model.keras"
checkpoint = ModelCheckpoint(model_name,
                            monitor="val_loss",
                            mode="min",
                            save_best_only = True,
                            verbose=1)

earlystopping = EarlyStopping(monitor='val_loss',min_delta = 0, 
patience = 5, verbose = 1, restore_best_weights=True)

learning_rate_reduction = ReduceLROnPlateau(monitor='val_loss', 
                                            patience=3, 
                                            verbose=1, 
```

```
                                            factor=0.2,
                                            min_lr=0.00000001)

history = caption_model.fit(
        train_generator,
        epochs=50,
        validation_data=validation_generator,
        callbacks=[checkpoint,earlystopping,learning_rate_reduction])

Epoch 1/50

/usr/local/lib/python3.10/dist-packages/keras/src/trainers/
data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset`
class should call `super().__init__(**kwargs)` in its constructor.
`**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will
be ignored.
  self._warn_if_super_not_called()

537/537 ━━━━━━━━━━━━━━━━━━━━ 0s 520ms/step - loss: 5.7390
Epoch 1: val_loss improved from inf to 4.36969, saving model to
model.keras
537/537 ━━━━━━━━━━━━━━━━━━━━ 336s 607ms/step - loss: 5.7379 -
val_loss: 4.3697 - learning_rate: 0.0010
Epoch 2/50
537/537 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - loss: 4.4077
Epoch 2: val_loss improved from 4.36969 to 4.11180, saving model to
model.keras
537/537 ━━━━━━━━━━━━━━━━━━━━ 89s 164ms/step - loss: 4.4076 - val_loss:
4.1118 - learning_rate: 0.0010
Epoch 3/50
537/537 ━━━━━━━━━━━━━━━━━━━━ 0s 144ms/step - loss: 4.1528
Epoch 3: val_loss improved from 4.11180 to 3.99827, saving model to
model.keras
537/537 ━━━━━━━━━━━━━━━━━━━━ 91s 168ms/step - loss: 4.1528 - val_loss:
3.9983 - learning_rate: 0.0010
Epoch 4/50
536/537 ━━━━━━━━━━━━━━━━━━━━ 0s 139ms/step - loss: 4.0011
Epoch 4: val_loss improved from 3.99827 to 3.92543, saving model to
model.keras
537/537 ━━━━━━━━━━━━━━━━━━━━ 88s 162ms/step - loss: 4.0010 - val_loss:
3.9254 - learning_rate: 0.0010
Epoch 5/50
537/537 ━━━━━━━━━━━━━━━━━━━━ 0s 138ms/step - loss: 3.8911
Epoch 5: val_loss improved from 3.92543 to 3.87179, saving model to
model.keras
537/537 ━━━━━━━━━━━━━━━━━━━━ 87s 161ms/step - loss: 3.8911 - val_loss:
3.8718 - learning_rate: 0.0010
Epoch 6/50
537/537 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - loss: 3.8109
```

```
Epoch 6: val_loss improved from 3.87179 to 3.84704, saving model to
model.keras
537/537 ━━━━━━━━━━━━━━━━━━━━ 143s 164ms/step - loss: 3.8109 -
val_loss: 3.8470 - learning_rate: 0.0010
Epoch 7/50
537/537 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - loss: 3.7257
Epoch 7: val_loss improved from 3.84704 to 3.82203, saving model to
model.keras
537/537 ━━━━━━━━━━━━━━━━━━━━ 88s 163ms/step - loss: 3.7257 - val_loss:
3.8220 - learning_rate: 0.0010
Epoch 8/50
536/537 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step - loss: 3.6771
Epoch 8: val_loss improved from 3.82203 to 3.81419, saving model to
model.keras
537/537 ━━━━━━━━━━━━━━━━━━━━ 91s 168ms/step - loss: 3.6771 - val_loss:
3.8142 - learning_rate: 0.0010
Epoch 9/50
537/537 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step - loss: 3.6262
Epoch 9: val_loss did not improve from 3.81419
537/537 ━━━━━━━━━━━━━━━━━━━━ 90s 166ms/step - loss: 3.6262 - val_loss:
3.8167 - learning_rate: 0.0010
Epoch 10/50
537/537 ━━━━━━━━━━━━━━━━━━━━ 0s 141ms/step - loss: 3.5712
Epoch 10: val_loss improved from 3.81419 to 3.81142, saving model to
model.keras
537/537 ━━━━━━━━━━━━━━━━━━━━ 89s 165ms/step - loss: 3.5713 - val_loss:
3.8114 - learning_rate: 0.0010
Epoch 11/50
537/537 ━━━━━━━━━━━━━━━━━━━━ 0s 149ms/step - loss: 3.5307
Epoch 11: val_loss improved from 3.81142 to 3.81115, saving model to
model.keras
537/537 ━━━━━━━━━━━━━━━━━━━━ 93s 172ms/step - loss: 3.5307 - val_loss:
3.8111 - learning_rate: 0.0010
Epoch 12/50
537/537 ━━━━━━━━━━━━━━━━━━━━ 0s 145ms/step - loss: 3.5052
Epoch 12: val_loss did not improve from 3.81115
537/537 ━━━━━━━━━━━━━━━━━━━━ 140s 168ms/step - loss: 3.5052 -
val_loss: 3.8142 - learning_rate: 0.0010
Epoch 13/50
536/537 ━━━━━━━━━━━━━━━━━━━━ 0s 144ms/step - loss: 3.4557
Epoch 13: val_loss did not improve from 3.81115
537/537 ━━━━━━━━━━━━━━━━━━━━ 90s 165ms/step - loss: 3.4558 - val_loss:
3.8117 - learning_rate: 0.0010
Epoch 14/50
536/537 ━━━━━━━━━━━━━━━━━━━━ 0s 144ms/step - loss: 3.4292
Epoch 14: val_loss did not improve from 3.81115

Epoch 14: ReduceLROnPlateau reducing learning rate to
0.00020000000949949026.
```

```
537/537 ━━━━━━━━━━━━━━━━━━━━ 91s 167ms/step - loss: 3.4292 - val_loss:
3.8340 - learning_rate: 0.0010
Epoch 15/50
537/537 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step - loss: 3.3690
Epoch 15: val_loss did not improve from 3.81115
537/537 ━━━━━━━━━━━━━━━━━━━━ 89s 165ms/step - loss: 3.3690 - val_loss:
3.8305 - learning_rate: 2.0000e-04
Epoch 16/50
537/537 ━━━━━━━━━━━━━━━━━━━━ 0s 142ms/step - loss: 3.3415
Epoch 16: val_loss did not improve from 3.81115
537/537 ━━━━━━━━━━━━━━━━━━━━ 89s 165ms/step - loss: 3.3415 - val_loss:
3.8546 - learning_rate: 2.0000e-04
Epoch 16: early stopping
Restoring model weights from the end of the best epoch: 11.

def idx_to_word(integer,tokenizer):

    for word, index in tokenizer.word_index.items():
        if index==integer:
            return word
    return None

def predict_caption(model, image, tokenizer, max_length, features):

    feature = features[image]
    in_text = "startseq"
    for i in range(max_length):
        sequence = tokenizer.texts_to_sequences([in_text])[0]
        sequence = pad_sequences([sequence], max_length)

        y_pred = model.predict([feature,sequence])
        y_pred = np.argmax(y_pred)

        word = idx_to_word(y_pred, tokenizer)

        if word is None:
            break

        in_text+= " " + word

        if word == 'endseq':
            break

    return in_text

samples = test.sample(15)
samples.reset_index(drop=True,inplace=True)

for index,record in samples.iterrows():

    img =
```

```
load_img(os.path.join(image_path,record['image']),target_size=(224,224
))
    img = img_to_array(img)
    img = img/255.

    caption = predict_caption(caption_model, record['image'],
tokenizer, max_length, features)
    samples.loc[index,'caption'] = caption
```

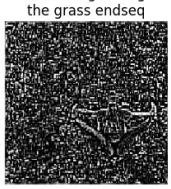
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 621ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 107ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 101ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 113ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 84ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 101ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 87ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step

display_images(samples)
```

startseq man in red shirt is riding bike on the road endseq

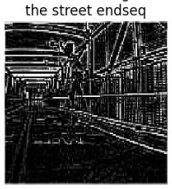
startseq two dogs are running through the grass endseq



startseq the dog is running through the snow endseq

startseq man in blue shirt is standing in the snow endseq

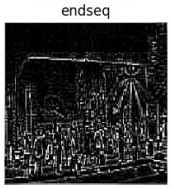
startseq man in blue shirt is standing on the street endseq



startseq man in blue shirt is sitting on the bed endseq

startseq two dogs are playing in the snow endseq

startseq group of people are walking down the street endseq



startseq group of people are sitting on the street endseq

startseq two dogs are playing in the snow endseq

startseq man in blue shirt is standing on the street endseq

startseq man is standing on the beach endseq

startseq man in blue shirt is standing on the beach endseq

startseq group of people are standing in the snow endseq

startseq group of people are standing in the snow endseq