Image Classification

Author: Simon Kim sxk190106@utdallas.edu

Date:03-12-2022

Data set source: https://www.kaggle.com/datasets/jehanbhathena/weather-dataset

The reason i chose this dataset is that it would be interesting and useful to classify weather from pictures or photos.

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

     Mounted at /content/gdrive
```

```
# importing libraries

import os
import glob
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import cv2
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from numpy import array
import random
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Dense
from keras.models import Model
```

```
#importing dataset
path = '/content/gdrive/MyDrive/Weather_Image_Recognition/dataset'
path_imgs = list(glob.glob(path+'/**/*.jpg'))
```

```
labels = list(map(lambda x:os.path.split(os.path.split(x)[0])[1], path_imgs))
file_path = pd.Series(path_imgs, name='File_Path').astype(str)
labels = pd.Series(labels, name='Labels')
```

```
data = pd.concat([file_path, labels], axis=1)
data = data.sample(frac=1).reset_index(drop=True)
data.head()
```

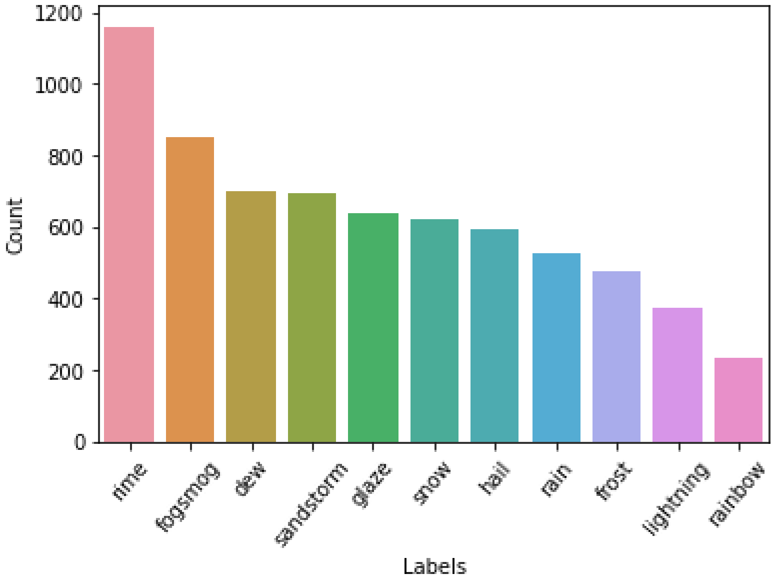
	File_Path	Labels
0	/content/gdrive/MyDrive/Weather_Image_Recognit...	rainbow
1	/content/gdrive/MyDrive/Weather_Image_Recognit...	lightning
2	/content/gdrive/MyDrive/Weather_Image_Recognit...	hail
3	/content/gdrive/MyDrive/Weather_Image_Recognit...	hail
4	/content/gdrive/MyDrive/Weather_Image_Recognit...	fogsmog

1. Create a graph showing the distribution of the target classes. Describe the data set and what the model should be able to predict.

```
# data visualization

fig, axes = plt.subplots(nrows=4, ncols=4, figsize=(15, 7),
                        subplot_kw={'xticks': [], 'yticks': []})
for i, ax in enumerate(axes.flat):
    ax.imshow(plt.imread(data.File_Path[i]))
    ax.set_title(data.Labels[i])
plt.tight_layout()
plt.show()
```

```
# plot graph
counts = data.Labels.value_counts()
sns.barplot(x=counts.index, y=counts)
plt.xlabel('Labels')
plt.ylabel('Count')
plt.xticks(rotation=50);
```



This data set has 11 classes of weather: dew, fogsmog, frost, glaze, hail, lightning, rain, rainbow, rime, sandstorm, and snow. From the dataset and training, we expect model to pridict weather from the given photographs.

```
# split train/test
train_df, test_df = train_test_split(data, test_size=0.2, random_state=2)


# image generator function

def gen(train,test):
    train_datagen = ImageDataGenerator(validation_split=0.2)
    test_datagen = ImageDataGenerator()

    train_gen = train_datagen.flow_from_dataframe(
        dataframe=train,
        x_col='File_Path',
        y_col='Labels',
        target_size=(100,100),
        class_mode='categorical',
        batch_size=32,
        shuffle=True,
        seed=0,
        subset='training',
        rotation_range=30,
        zoom_range=0.15,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.15,
        horizontal_flip=True,
        fill_mode="nearest")
    valid_gen = train_datagen.flow_from_dataframe(
        dataframe=train,
        x_col='File_Path',
        y_col='Labels',
        target_size=(100,100),
        class_mode='categorical',
        batch_size=32,
        shuffle=False,
        seed=0,
        subset='validation',
        rotation_range=30,
        zoom_range=0.15,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.15,
        horizontal_flip=True,
        fill_mode="nearest")
    test_gen = test_datagen.flow_from_dataframe(
        dataframe=test,
        x_col='File_Path',
        y_col='Labels',
        target_size=(100,100),
        color_mode='rgb',
        class_mode='categorical',
        batch_size=32,
        verbose=0,
        shuffle=False)
    return train_gen, valid_gen, test_gen
```

2. Create a sequential model and evaluate on the test data

```
train_seq, valid_seq, test_seq = gen(train_df,test_df)
```

```
Found 4392 validated image filenames belonging to 11 classes.
Found 1097 validated image filenames belonging to 11 classes.
Found 1373 validated image filenames belonging to 11 classes.
```

```
from keras import Sequential

from keras.layers import Activation, Dense, Flatten
from keras.optimizers import Adam

model_seq = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=[100, 100,3]),
  tf.keras.layers.Dense(300, activation='relu'),
  tf.keras.layers.Dense(100, activation='relu'),
  tf.keras.layers.Dense(11, activation='softmax'),
])

model_seq.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 30000)             0

 dense (Dense)               (None, 300)               9000300

 dense_1 (Dense)             (None, 100)               30100

 dense_2 (Dense)             (None, 11)                1111

=================================================================
Total params: 9,031,511
Trainable params: 9,031,511
Non-trainable params: 0
_________________________________________________________________
```

```
model_seq.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])

history = model_seq.fit(
    train_seq,
    validation_data=valid_seq,
    epochs=30,
    verbose=0
)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-11-101cf6c96f06> in <module>
      3               metrics=['accuracy'])
      4 
----> 5 history = model_seq.fit(
      6     train_seq,
      7     validation_data=valid_seq,

                                    8 frames
/usr/local/lib/python3.8/dist-packages/tensorflow/python/eager/execute.py in quick_execute(op_na
inputs, attrs, ctx, name)
```

*The code block took too long run time(over 40min), so I interrupted the execution.

```
---> 54     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
```

3. Try a different architectures like RNN, CNN, etc and evaluate on the test data

```
KeyboardInterrupt:
```

```
#CNN

# try datacleaning this time

#preprocessing
train_folder_list = array(os.listdir(path))

label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(train_folder_list)
onehot_encoder = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded),1)
onehot_encoded = onehot_encoder.fit_transform(integer_encoded)

trains_CNN = []
tests_CNN = []

for index in range(len(train_folder_list)):
  path= os.path.join(path,train_folder_list[index])

  train_path = path + '₩₩Training'
  img_list = os.listdir(train_path)
  for img in img_list:
    img_path= os.path.join(train_path, img)
    img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
    img = cv2.resize(img, (64,64), interpolation=cv2.INTER_CUBIC)

    trains_CNN.append( ([np.array(img)], [np.array(onehot_encoded[index])]))

  test_path = path + '₩₩Test'
  img_list = os.listdir(test_path)
  for img in img_list:
    img_path= os.path.join(test_path, img)
    img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
    img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
    img = cv2.resize(img, (64,64), interpolation=cv2.INTER_CUBIC)

    tests_CNN.append( ([np.array(img)], [np.array(onehot_encoded[index])]))


train_input = []
train_label = []
test_input = []
test_label = []

random.shuffle(trains_CNN)
random.shuffle(tests_CNN)

for (i,j) in trains_CNN:
  train_input.append(i)
  train_label.append(j)

for (i,j) in tests_CNN:
  test_input.append(i)
  test_label.append(j)

train_input= np.reshape(train_input, (-1,4096))
train_label= np.reshape(train_label, (-1,11))
train_input = np.array(train_input).astype(np.float32)
train_label = np.array(train_label).astype(np.float32)
np.save("train_data.npy", train_input)
np.save("train_label.npy", train_label)

test_input= np.reshape(test_input, (-1,4096))
test_label= np.reshape(test_label, (-1,11))
test_input = np.array(test_input).astype(np.float32)
test_label = np.array(test_label).astype(np.float32)
np.save("test_data.npy", test_input)
np.save("test_label.npy", test_label)


#hyper parameters

learning_rate = 0.001
tf.reset_default_graph()
keep_prob = tf.placeholder(tf.float32)

#input place holders
X = tf.placeholder(tf.float32, [None,4096])
X_img = tf.reshape (X, [-1, 64, 64, 1])
Y=tf.placeholder(tf.float32, [None,11])
```

4. Try a pretrained model and transfer learning, read more here: https://www.tensorflow.org/tutorials/images/transfer_learning

5. Write up your analysis of the performance of various approaches