```python
import numpy as np
import pandas as pd
from keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import random
import os
```

```python
img_width=128
img_height=128
img_size=(128,128)
img_channels=3
Directory = os.listdir(".\Users\PC\Desktop\project dl\train")

labels=[]
for name in Directory:
    label =name.split('.')[0]
    if label=='dog':
        labels.append(1)
    else:
        labels.append(0)
df=pd.DataFrame({
    'filename':Directory,
    'label':labels
})
df.head()
df.tail()
from keras.models import Sequential
from keras.layers import
Conv2D,MaxPooling2D,Dropout,Flatten,Dense,Activation,BatchNormalization

model=Sequential()

model.add(Conv2D(64,(3,3),activation='relu',input_shape=(128,128,3)))
model.add(BatchNormalization())
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.25))

model.add(Conv2D(64,(3,3),activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.25))
```

```
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(2,activation='softmax'))

model.compile(loss='categorical_crossentropy',
  optimizer='rmsprop',metrics=['accuracy'])
model.summary()
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
```

```
earlystop = EarlyStopping(patience=10)
learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc',
                                            patience=2,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr=0.00001)
callbacks = [earlystop, learning_rate_reduction]
```

```
df["label"] = df["label"].replace({0: 'cat', 1: 'dog'})
train_data, validation_data = train_test_split(df, test_size=0.20,
random_state=42)
train_data = train_data.reset_index(drop=True)
validation_data = validation_data.reset_index(drop=True)
final_train_data = train_data.shape[0]
final_validation_data = validation_data.shape[0]
batch_size=15
```

```
generate_train_data = ImageDataGenerator(rotation_range=15,
                                 rescale=1./255,
                                 shear_range=0.1,
                                 zoom_range=0.2,
                                 horizontal_flip=True,
                                 width_shift_range=0.1,
                                 height_shift_range=0.1
                                 )
train_gen = generate_train_data.flow_from_dataframe(train_data,
```

```
                                                    "./dogs-vs-
cats/train/train",x_col='filename',y_col='label',
                                                        target_size=img_size,
                                                        class_mode='categorical',
                                                        batch_size=batch_size)
```

```
generate_validation_data = ImageDataGenerator(rescale=1./255)
validation_gen = generate_validation_data.flow_from_dataframe(
    validation_data,
    "./dogs-vs-cats/train/train",
    x_col='filename',
    y_col='label',
    target_size=img_size,
    class_mode='categorical',
    batch_size=batch_size
)
```

```
epochs=10
history = model.fit_generator(
    train_gen,
    epochs=epochs,
    validation_data=validation_gen,
    validation_steps=final_validation_data//batch_size,
    steps_per_epoch=final_train_data//batch_size,
    callbacks=callbacks
)
```

```
model.save("model1_catsVSdogs_10epoch.h5")
```

```
test_filenames = os.listdir("./dogs-vs-cats/test1/test1")
test_data = pd.DataFrame({
    'filename': test_filenames
})
nb_samples = test_data.shape[0]
```

```
generate_test_data= ImageDataGenerator(rescale=1./255)
test_gen = generate_test_data.flow_from_dataframe(
    test_data,
    "./dogs-vs-cats/test1/test1",
    x_col='filename',
    y_col=None,
```

```
    class_mode=None,
    target_size=img_size,
    batch_size=batch_size,
    shuffle=False
)
```

```
prediction = model.predict_generator(test_gen,
steps=np.ceil(nb_samples/batch_size))
```

```
test_data['label'] = np.argmax(prediction, axis=-1)
label_map = dict((v,k) for k,v in train_gen.class_indices.items())
test_data['label'] = test_data['label'].replace(label_map)
test_data['label'] = test_data['label'].replace({ 'dog': 1, 'cat': 0 })
```

```
testing = test_data.head(10)
testing.head()
plt.figure(figsize=(12, 24))
for index, row in testing.iterrows():
    filename = row['filename']
    label = row['label']
    img = load_img("./dogs-vs-cats/test1/test1/"+filename, target_size=img_size)
    plt.subplot(6, 3, index+1)
    plt.imshow(img)
    plt.xlabel(filename + '(' + "{}".format(label) + ')' )
plt.tight_layout()
plt.show()
```

```
results={
    0:'cat',
    1:'dog'
}
from PIL import Image
import numpy as np
im=Image.open("download1.jpeg")
im=im.resize(img_size)
im=np.expand_dims(im,axis=0)
im=np.array(im)
im=im/255
pred=model.predict([im])[0]
print(pred)
```

Spyder (Python 3.8)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\PC

C:\Users\PC\Desktop\project dl\Cats_Vs_Dogs_GUI.py

temp.py | Cats_Vs_Dogs_GUI.py*

```python


import tkinter as tk
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image
import numpy
from keras.models import load_model
model = load_model("C:/Users/PC/Desktop/project dl/model1_catsVSdogs_10epoch.h5")
#dictionary to label all traffic signs class.
classes = {
    0:"It's a cat",
    1:"It's a dog",

}
#initialise GUI
top=tk.Tk()
top.geometry('800x600')
top.title('CatsVSDogs Classification')
top.configure(background='#CDCDCD')
label=tk.Label(top,background='#CDCDCD', font=('arial',15,'bold'))
sign_image = tk.Label(top)
def classify(file_path):
    global label_packed
    image = Image.open(file_path)
    image = image.resize((128,128))
    image = numpy.expand_dims(image, axis=0)
    image = numpy.array(image)
    image = image/255
    pred = model.predict([image])[0]
    if(pred[0] >= 0.5):
        sign = classes[0]
    else:
        sign = classes[1]
    label.configure(foreground='#011638', text=sign)
def show_classify_button(file_path):
```

Source | Console | Object

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

Help | Variable explorer | Plots | Files

Console 1/A

```
Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda.dll not found
2021-12-15 18:16:29.831384: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed
call to cuInit: UNKNOWN ERROR (303)
2021-12-15 18:16:29.834816: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169]
retrieving CUDA diagnostic information for host: DESKTOP-JP4927N
2021-12-15 18:16:29.834970: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176]
hostname: DESKTOP-JP4927N
2021-12-15 18:16:29.844224: I tensorflow/core/platform/cpu_feature_guard.cc:151] This
TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the
following CPU instructions in performance-critical operations:  AVX
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2021-12-15 18:16:30.028904: W tensorflow/core/framework/cpu_allocator_impl.cc:82]
Allocation of 51380224 exceeds 10% of free system memory.
2021-12-15 18:16:30.162270: W tensorflow/core/framework/cpu_allocator_impl.cc:82]
Allocation of 51380224 exceeds 10% of free system memory.
2021-12-15 18:16:30.190245: W tensorflow/core/framework/cpu_allocator_impl.cc:82]
Allocation of 51380224 exceeds 10% of free system memory.
```

IPython console | History

LSP Python: ready | conda: base (Python 3.8.8) | Line 27, Col 45 | UTF-8-GUESSED | CRLF | RW | Mem 81%

Spyder (Python 3.8)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\PC\Desktop\project dl\Cats_Vs_Dogs_GUI.py

temp.py | Cats_Vs_Dogs_GUI.py

```python


import tkinter as tk
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image
import numpy
from keras.models import load_model
model = load_model("C:/Users/PC/Desktop/project dl/model1_co
#dictionary to label all traffic signs class.
classes = {
    0:"It's a cat",
    1:"It's a dog",

}
#initialise GUI
top=tk.Tk()
top.geometry('800x600')
top.title('CatsVSDogs Classification')
top.configure(background='#CDCDCD')
label=tk.Label(top,background='#CDCDCD', font=('arial',15,'b
sign_image = tk.Label(top)
def classify(file_path):
    global label_packed
    image = Image.open(file_path)
    image = image.resize((128,128))
    image = numpy.expand_dims(image, axis=0)
    image = numpy.array(image)
    image = image/255
    pred = model.predict([image])[0]
    if(pred[0] >= 0.5):
        sign = classes[0]
    else:
        sign = classes[1]
    label.configure(foreground='#011638', text=sign)
def show_classify_button(file_path):
```

CatsVSDogs Classification

CatsVSDogs Classification

It's a cat

Classify Image

Upload an image

LSP Python: ready | conda: base (Python 3.8.8) | Line 27, Col 45 | UTF-8 | CRLF | RW | Mem 82%

Type here to search

62°F | ENG IN | 6:29 PM 12/15/2021