```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt


data_dir='/content/drive/MyDrive/ALL_DS'
data=tf.keras.preprocessing.image_dataset_from_directory(data_dir,
                                        batch_size=10,
                                       image_size=(224,224))
```

```
Found 172 files belonging to 3 classes.
```

```
class_names=data.class_names
print(class_names)
```

```
['daisy', 'dandelion', 'roses']
```

```
train_batches=tf.data.experimental.cardinality(data)
print('Training Batch=',train_batches.numpy())
```

```
Training Batch= 18
```

```
# a single batch
# visulize
plt.figure(figsize=(10,10))
for image_batch,label_batch in data.take(1):
    print(image_batch.shape)
    print(label_batch.shape)
    print(label_batch)
    for i in range(10):
        plt.subplot(1,10,i+1)
        plt.imshow(image_batch[i]/255.0) # float [0-1],int[0-255]
        plt.axis('off')
        plt.title(class_names[label_batch[i]])
```

```
(10, 224, 224, 3)
(10,)
tf.Tensor([2 1 2 1 0 2 0 0 0 0], shape=(10,), dtype=int32)
```


```
# WAP to load a data in a batch and visulize any one batch
# WAP to load a data in a batch and compile whole data in one matrix
# 63 batch of 32x160x160x3 = 2000x160x160x3
img=[]
label=[]
for image_batch,label_batch in data:
    img.append(image_batch)
    label.append(label_batch)
data_inputs=np.concatenate(img)
targets=np.concatenate(label)
print(data_inputs.shape)
print(targets.shape)
#################
i=1
image=data_inputs[i]
label=targets[i]
plt.imshow(image/255.0)
plt.title(class_names[label])
plt.axis('off')
```

```
(172, 224, 224, 3)
(172,)
(-0.5, 223.5, 223.5, -0.5)
```


```
# Data augmentation
data_aug=tf.keras.Sequential([
    tf.keras.layers.experimental.preprocessing.RandomFlip('horizontal'),
    tf.keras.layers.experimental.preprocessing.RandomFlip('vertical'),
    tf.keras.layers.experimental.preprocessing.RandomZoom(0.2),
    tf.keras.layers.experimental.preprocessing.RandomRotation(0.3),
    #tf.keras.layers.experimental.preprocessing.Randomcrop(100,100),
    # Random tranlate, random height, Random width
])
plt.figure(figsize=(10,2))
for image_batch,label_batch in data.take(1):
    aug_img=data_aug(image_batch)
    for i in range(10):
        plt.subplot(2,10,i+1)
        plt.imshow(image_batch[i]/255.0) # float [0-1],int[0-255]
        plt.axis('off')
        plt.title(class_names[label_batch[i]])
        plt.subplot(2,10,i+11)
        plt.imshow(aug_img[i]/255.0) # float [0-1],int[0-255]
        plt.axis('off')
        plt.title(class_names[label_batch[i]])
```


```
# load a pretrained NN
base_model=tf.keras.applications.vgg16.VGG16(input_shape=(224,224,3),
                                include_top=False,weights='imagenet')
base_model.trainable=False # freeze the model for training
print('Number of layers=',len(base_model.layers))
print('Number of weights[W/B]=',len(base_model.weights))
print('Number of trainable variables=',len(base_model.trainable_variables))
base_model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kerne
58889256/58889256 [==============================] - 0s 0us/step
Number of layers= 19
Number of weights[W/B]= 26
Number of trainable variables= 0
Model: "vgg16"
_________________________________________________________________
 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

=================================================================
Total params: 14714688 (56.13 MB)
Trainable params: 0 (0.00 Byte)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
```

```
# custom model
inputs=tf.keras.Input(shape=(224,224,3)) # input layer
x=data_aug(inputs)                       # data augmentation
x=tf.keras.applications.vgg16.preprocess_input(x)     #imagenetv2           # preprocessing as model
x=base_model(x,training=False)           # base model
################################################
x=tf.keras.layers.GlobalAveragePooling2D()(x) # Nonex1280  or FLATTEN()(x)
x=tf.keras.layers.Dense(128,activation='relu')(x)
x=tf.keras.layers.Dropout(0.2)(x)
outputs=tf.keras.layers.Dense(30)(x) # outer layer 1 neuron, 30 featues
model=tf.keras.Model(inputs,outputs)
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0

 sequential_1 (Sequential)   (None, 224, 224, 3)       0

 tf.__operators__.getitem (  (None, 224, 224, 3)       0
 SlicingOpLambda)

 tf.nn.bias_add (TFOpLambda  (None, 224, 224, 3)       0
 )

 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 global_average_pooling2d (  (None, 512)               0
 GlobalAveragePooling2D)

 dense (Dense)               (None, 128)               65664

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 30)                3870

=================================================================
Total params: 14784222 (56.40 MB)
Trainable params: 69534 (271.62 KB)
Non-trainable params: 14714688 (56.13 MB)
```

```
model.compile(optimizer='sgd',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

Start coding or generate with AI.

```
TensorShape([None, 224, 224, 3])
```

```
pred=model.predict(data_inputs)
pred.shape
```

```
6/6 [==============================] - 145s 24s/step
(172, 30)
```

```
## save feat
import pandas as pd
pd.DataFrame(pred).to_csv('data_feature.csv',index=False)
pd.DataFrame(targets).to_csv('target_feature.csv',index=False)
```

```
x_data=pd.read_csv('data_feature.csv')
x_label=pd.read_csv('target_feature.csv')
print(x_data.shape)
print(x_label.shape)
x_label=np.squeeze(x_label,axis=1)
print(x_label.shape)
########### train_test_split
from sklearn.model_selection import train_test_split
(x_train,x_test,y_train,y_test)=train_test_split(x_data,x_label,
                                                 test_size=0.2)
########### Standard scaler
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
sc.fit(x_train)
x_train_std=sc.transform(x_train)
x_test_std=sc.transform(x_test)
############################
from sklearn.svm import SVC
model=SVC()
model.fit(x_train_std,y_train)
y_pred=model.predict(x_test_std)
######################
from sklearn.metrics import accuracy_score
acc=accuracy_score(y_pred,y_test)
print('Testing ACC=',acc*100)
```

```
(172, 30)
(172, 1)
(172,)
Testing ACC= 77.14285714285715
```