Untitled13

November 14, 2024

[7]:
```
#data library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[9]:
```
#library for cnn model and for preprocessing data
from sklearn.model_selection import train_test_split
import keras
from keras.models import Sequential
from keras.layers import Conv2D,MaxPooling2D,Dense,Flatten,Dropout,Input
from keras.optimizers import Adam
from keras.callbacks import TensorBoard
from sklearn.preprocessing import LabelEncoder
```

[10]:
```
df = pd.read_csv(r"C:\Users\Public\Documents\Python_project\plant_diseases.csv")
df.head()
```

[10]:
```
   pixel_0  pixel_1  pixel_2  pixel_3  pixel_4  pixel_5  pixel_6  pixel_7  \
0      112      109      124      109      106      121      113      110
1      158      135      149      149      126      140      155      132
2      104       87      101       20       10       13       21       16
3       10       10        6       10        7        7       17        6
4      166      165      175      168      167      177      171      170

   pixel_8  pixel_9  …  pixel_12279  pixel_12280  pixel_12281  pixel_12282  \
0      125      112  …          163          154          167          164
1      146      156  …          139          118          133          157
2       15       17  …          201          184          193          206
3       12       42  …          192          169          183          213
4      180      175  …           97           94          109          102

   pixel_12283  pixel_12284  pixel_12285  pixel_12286  pixel_12287  \
0          155          168          160          151          164
1          136          151          154          133          148
2          189          198          208          191          200
3          190          204          166          143          157
4           99          114          111          108          123
```

```
                        label
0  Pepper__bell___Bacterial_spot
1  Pepper__bell___Bacterial_spot
2  Pepper__bell___Bacterial_spot
3  Pepper__bell___Bacterial_spot
4  Pepper__bell___Bacterial_spot

[5 rows x 12289 columns]
```

[13]:
```
le = LabelEncoder()
y = le.fit_transform(df['label'])
X = df.iloc[:,:-1]
```

[15]:
```
x_data_train,x_data_test,y_train,y_test = train_test_split(X,y,test_size=0.
 ↪20,random_state=42)
X_trainn = np.array(x_data_train,dtype='float32')
X_testt = np.array(x_data_test,dtype='float32')
X_train = X_trainn[:,:]/255
X_test = X_testt[:,:]/255

# print(X_test)
```

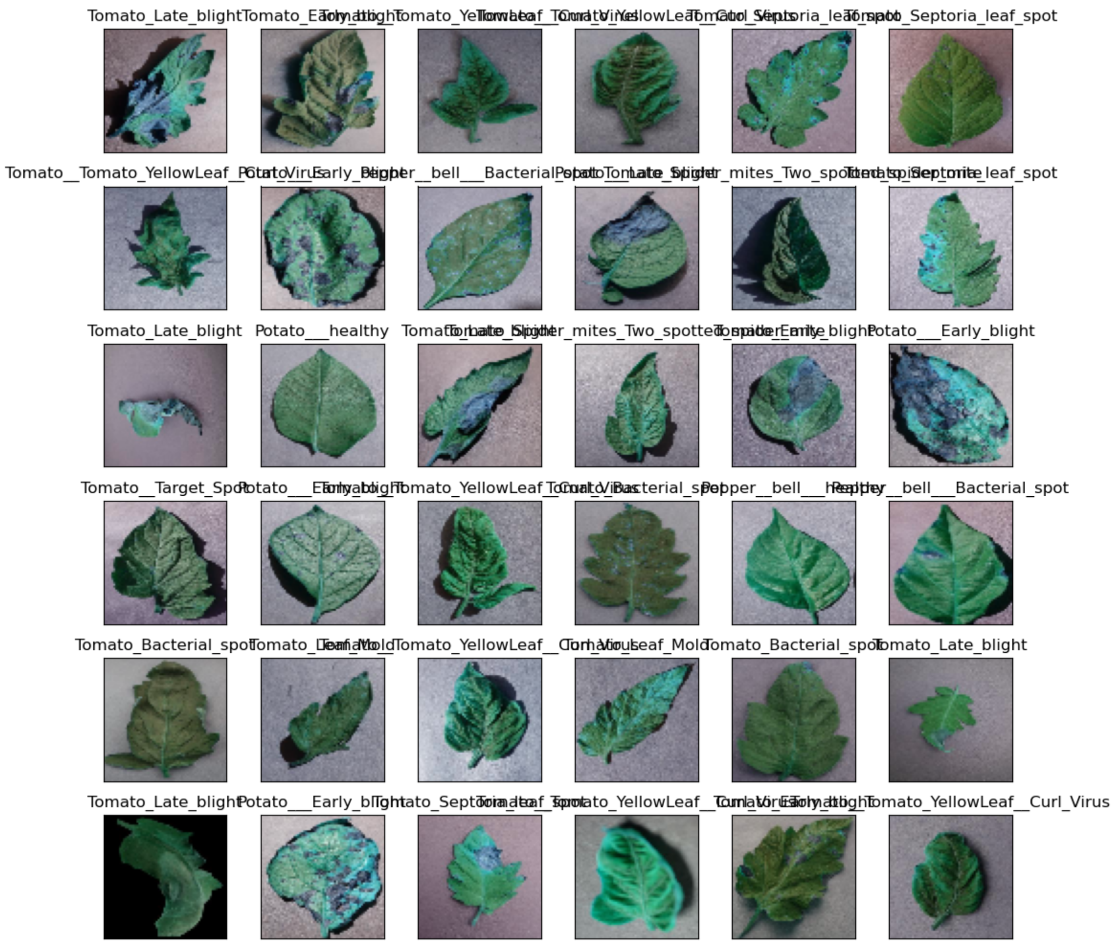
[16]:
```
print(df['label'].unique())
```

```
['Pepper__bell___Bacterial_spot' 'Pepper__bell___healthy'
 'Potato___Early_blight' 'Potato___Late_blight' 'Potato___healthy'
 'Tomato_Bacterial_spot' 'Tomato_Early_blight' 'Tomato_Late_blight'
 'Tomato_Leaf_Mold' 'Tomato_Septoria_leaf_spot'
 'Tomato_Spider_mites_Two_spotted_spider_mite' 'Tomato__Target_Spot'
 'Tomato__Tomato_YellowLeaf__Curl_Virus']
```

[17]:
```
class_names = ['Pepper__bell___Bacterial_spot', 'Pepper__bell___healthy',
 'Potato___Early_blight', 'Potato___Late_blight', 'Potato___healthy',
 'Tomato_Bacterial_spot', 'Tomato_Early_blight', 'Tomato_Late_blight',
 'Tomato_Leaf_Mold', 'Tomato_Septoria_leaf_spot',
 'Tomato_Spider_mites_Two_spotted_spider_mite', 'Tomato__Target_Spot',
 'Tomato__Tomato_YellowLeaf__Curl_Virus']
```

[18]:
```
plt.figure(figsize=(10, 10))

# Loop through the first 36 images
for i in range(36):
    plt.subplot(6, 6, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)

    # Display the image (assuming each image can be reshaped to 64x64x3)
```

```
    plt.imshow(X_train[i].reshape((64, 64, 3)))

    # Set the title based on the label
    label_index = int(y_train[i])  # Ensure label is an integer
    plt.title(class_names[label_index])

# Adjust layout
plt.tight_layout()
plt.show()
```



[23]:
```
image_rows = 64
image_cols = 64
image_shape = (image_rows,image_cols,3)
```

[27]:
```
X_train = X_train.reshape(X_train.shape[0],*image_shape)
X_test = X_test.reshape(X_test.shape[0],*image_shape)
```

[41]:
```
cnn_model = Sequential([
    Input(shape=image_shape),
    Conv2D(filters=32,kernel_size=3,activation='relu'),
    MaxPooling2D(pool_size=2) ,# down sampling the output instead of 28*28 it␣
↪is 14*14
    Dropout(0.2),
    Flatten(), # flatten out the layers
    Dense(32,activation='relu'),
    Dense(13,activation = 'softmax')

])
```

[43]:
```
cnn_model.compile(loss ='sparse_categorical_crossentropy',␣
↪optimizer=Adam(learning_rate=0.001),metrics =['accuracy'])
```

[ ]:

[45]:
```
history = cnn_model.fit(
    X_train,
    y_train,
    batch_size=500,
    epochs=75,
    verbose=1,
    validation_data=(X_test,y_test),
)
```

```
Epoch 1/75
30/30                16s 420ms/step -
accuracy: 0.1219 - loss: 3.2478 - val_accuracy: 0.1686 - val_loss: 2.3943
Epoch 2/75
30/30                9s 292ms/step -
accuracy: 0.2075 - loss: 2.3292 - val_accuracy: 0.2792 - val_loss: 2.1503
Epoch 3/75
30/30                9s 289ms/step -
accuracy: 0.3085 - loss: 2.0860 - val_accuracy: 0.3776 - val_loss: 1.9414
Epoch 4/75
30/30                8s 277ms/step -
accuracy: 0.3963 - loss: 1.8725 - val_accuracy: 0.4051 - val_loss: 1.7917
Epoch 5/75
30/30                9s 290ms/step -
accuracy: 0.4390 - loss: 1.7236 - val_accuracy: 0.4897 - val_loss: 1.6198
Epoch 6/75
30/30                9s 290ms/step -
accuracy: 0.5369 - loss: 1.5429 - val_accuracy: 0.5497 - val_loss: 1.4730
Epoch 7/75
30/30                9s 292ms/step -
accuracy: 0.5718 - loss: 1.4338 - val_accuracy: 0.5865 - val_loss: 1.3698
Epoch 8/75
```

```
30/30            9s 290ms/step -
accuracy: 0.5973 - loss: 1.3106 - val_accuracy: 0.6162 - val_loss: 1.2693
Epoch 9/75
30/30            9s 286ms/step -
accuracy: 0.6298 - loss: 1.2243 - val_accuracy: 0.6241 - val_loss: 1.2274
Epoch 10/75
30/30            9s 291ms/step -
accuracy: 0.6524 - loss: 1.1333 - val_accuracy: 0.6595 - val_loss: 1.1294
Epoch 11/75
30/30            9s 289ms/step -
accuracy: 0.6697 - loss: 1.0690 - val_accuracy: 0.6686 - val_loss: 1.0817
Epoch 12/75
30/30            9s 292ms/step -
accuracy: 0.6838 - loss: 1.0215 - val_accuracy: 0.6789 - val_loss: 1.0368
Epoch 13/75
30/30            9s 282ms/step -
accuracy: 0.7038 - loss: 0.9520 - val_accuracy: 0.6757 - val_loss: 1.0288
Epoch 14/75
30/30            9s 292ms/step -
accuracy: 0.7015 - loss: 0.9407 - val_accuracy: 0.6943 - val_loss: 0.9603
Epoch 15/75
30/30            9s 296ms/step -
accuracy: 0.7165 - loss: 0.8912 - val_accuracy: 0.7057 - val_loss: 0.9370
Epoch 16/75
30/30            9s 294ms/step -
accuracy: 0.7282 - loss: 0.8538 - val_accuracy: 0.7089 - val_loss: 0.9283
Epoch 17/75
30/30            9s 294ms/step -
accuracy: 0.7328 - loss: 0.8416 - val_accuracy: 0.7124 - val_loss: 0.9011
Epoch 18/75
30/30            9s 287ms/step -
accuracy: 0.7392 - loss: 0.8169 - val_accuracy: 0.7270 - val_loss: 0.8700
Epoch 19/75
30/30            9s 281ms/step -
accuracy: 0.7546 - loss: 0.7762 - val_accuracy: 0.7308 - val_loss: 0.8519
Epoch 20/75
30/30            9s 292ms/step -
accuracy: 0.7551 - loss: 0.7660 - val_accuracy: 0.7205 - val_loss: 0.8476
Epoch 21/75
30/30            9s 300ms/step -
accuracy: 0.7686 - loss: 0.7286 - val_accuracy: 0.7362 - val_loss: 0.8216
Epoch 22/75
30/30            9s 289ms/step -
accuracy: 0.7628 - loss: 0.7376 - val_accuracy: 0.7300 - val_loss: 0.8485
Epoch 23/75
30/30            9s 284ms/step -
accuracy: 0.7687 - loss: 0.7174 - val_accuracy: 0.7454 - val_loss: 0.7999
Epoch 24/75
```

```
30/30              8s 280ms/step -
accuracy: 0.7832 - loss: 0.6776 - val_accuracy: 0.7486 - val_loss: 0.7913
Epoch 25/75
30/30              9s 290ms/step -
accuracy: 0.7803 - loss: 0.6772 - val_accuracy: 0.7486 - val_loss: 0.7790
Epoch 26/75
30/30              9s 290ms/step -
accuracy: 0.7940 - loss: 0.6500 - val_accuracy: 0.7505 - val_loss: 0.7942
Epoch 27/75
30/30              9s 291ms/step -
accuracy: 0.7909 - loss: 0.6435 - val_accuracy: 0.7403 - val_loss: 0.7794
Epoch 28/75
30/30              9s 282ms/step -
accuracy: 0.8023 - loss: 0.6129 - val_accuracy: 0.7557 - val_loss: 0.7595
Epoch 29/75
30/30              8s 281ms/step -
accuracy: 0.8044 - loss: 0.6112 - val_accuracy: 0.7595 - val_loss: 0.7506
Epoch 30/75
30/30              9s 292ms/step -
accuracy: 0.8138 - loss: 0.5795 - val_accuracy: 0.7654 - val_loss: 0.7268
Epoch 31/75
30/30              9s 290ms/step -
accuracy: 0.8158 - loss: 0.5730 - val_accuracy: 0.7489 - val_loss: 0.7661
Epoch 32/75
30/30              9s 291ms/step -
accuracy: 0.8222 - loss: 0.5596 - val_accuracy: 0.7689 - val_loss: 0.7378
Epoch 33/75
30/30              9s 282ms/step -
accuracy: 0.8218 - loss: 0.5580 - val_accuracy: 0.7757 - val_loss: 0.7116
Epoch 34/75
30/30              8s 281ms/step -
accuracy: 0.8311 - loss: 0.5310 - val_accuracy: 0.7705 - val_loss: 0.7124
Epoch 35/75
30/30              9s 291ms/step -
accuracy: 0.8415 - loss: 0.5125 - val_accuracy: 0.7746 - val_loss: 0.6983
Epoch 36/75
30/30              9s 291ms/step -
accuracy: 0.8470 - loss: 0.4902 - val_accuracy: 0.7746 - val_loss: 0.6996
Epoch 37/75
30/30              9s 293ms/step -
accuracy: 0.8469 - loss: 0.4885 - val_accuracy: 0.7776 - val_loss: 0.6896
Epoch 38/75
30/30              8s 278ms/step -
accuracy: 0.8497 - loss: 0.4793 - val_accuracy: 0.7884 - val_loss: 0.6752
Epoch 39/75
30/30              9s 284ms/step -
accuracy: 0.8619 - loss: 0.4466 - val_accuracy: 0.7776 - val_loss: 0.6878
Epoch 40/75
```

```
30/30             9s 292ms/step -
accuracy: 0.8607 - loss: 0.4498 - val_accuracy: 0.7911 - val_loss: 0.6697
Epoch 41/75
30/30             9s 292ms/step -
accuracy: 0.8627 - loss: 0.4386 - val_accuracy: 0.7824 - val_loss: 0.6748
Epoch 42/75
30/30             9s 293ms/step -
accuracy: 0.8650 - loss: 0.4172 - val_accuracy: 0.7924 - val_loss: 0.6550
Epoch 43/75
30/30             9s 284ms/step -
accuracy: 0.8682 - loss: 0.4262 - val_accuracy: 0.7886 - val_loss: 0.6713
Epoch 44/75
30/30             9s 286ms/step -
accuracy: 0.8767 - loss: 0.4035 - val_accuracy: 0.7857 - val_loss: 0.6686
Epoch 45/75
30/30             9s 288ms/step -
accuracy: 0.8659 - loss: 0.4174 - val_accuracy: 0.7881 - val_loss: 0.6720
Epoch 46/75
30/30             9s 292ms/step -
accuracy: 0.8811 - loss: 0.3892 - val_accuracy: 0.7905 - val_loss: 0.6646
Epoch 47/75
30/30             9s 294ms/step -
accuracy: 0.8882 - loss: 0.3723 - val_accuracy: 0.7781 - val_loss: 0.7060
Epoch 48/75
30/30             8s 280ms/step -
accuracy: 0.8883 - loss: 0.3695 - val_accuracy: 0.7995 - val_loss: 0.6493
Epoch 49/75
30/30             9s 303ms/step -
accuracy: 0.8964 - loss: 0.3493 - val_accuracy: 0.7949 - val_loss: 0.6536
Epoch 50/75
30/30             9s 293ms/step -
accuracy: 0.8968 - loss: 0.3432 - val_accuracy: 0.8073 - val_loss: 0.6418
Epoch 51/75
30/30             9s 295ms/step -
accuracy: 0.9040 - loss: 0.3300 - val_accuracy: 0.7919 - val_loss: 0.6641
Epoch 52/75
30/30             9s 290ms/step -
accuracy: 0.8986 - loss: 0.3321 - val_accuracy: 0.8046 - val_loss: 0.6342
Epoch 53/75
30/30             8s 278ms/step -
accuracy: 0.9114 - loss: 0.3099 - val_accuracy: 0.7962 - val_loss: 0.6459
Epoch 54/75
30/30             9s 301ms/step -
accuracy: 0.9064 - loss: 0.3154 - val_accuracy: 0.8024 - val_loss: 0.6347
Epoch 55/75
30/30             9s 291ms/step -
accuracy: 0.9106 - loss: 0.2976 - val_accuracy: 0.7995 - val_loss: 0.6396
Epoch 56/75
```

```
30/30              9s 291ms/step -
accuracy: 0.9127 - loss: 0.2894 - val_accuracy: 0.8051 - val_loss: 0.6409
Epoch 57/75
30/30              10s 278ms/step -
accuracy: 0.9166 - loss: 0.2892 - val_accuracy: 0.8008 - val_loss: 0.6377
Epoch 58/75
30/30              9s 286ms/step -
accuracy: 0.9218 - loss: 0.2807 - val_accuracy: 0.7949 - val_loss: 0.6530
Epoch 59/75
30/30              9s 290ms/step -
accuracy: 0.9249 - loss: 0.2707 - val_accuracy: 0.7892 - val_loss: 0.6548
Epoch 60/75
30/30              9s 294ms/step -
accuracy: 0.9214 - loss: 0.2739 - val_accuracy: 0.8008 - val_loss: 0.6460
Epoch 61/75
30/30              9s 292ms/step -
accuracy: 0.9212 - loss: 0.2654 - val_accuracy: 0.8057 - val_loss: 0.6302
Epoch 62/75
30/30              8s 280ms/step -
accuracy: 0.9304 - loss: 0.2459 - val_accuracy: 0.7989 - val_loss: 0.6453
Epoch 63/75
30/30              9s 289ms/step -
accuracy: 0.9304 - loss: 0.2440 - val_accuracy: 0.7851 - val_loss: 0.6698
Epoch 64/75
30/30              9s 293ms/step -
accuracy: 0.9288 - loss: 0.2472 - val_accuracy: 0.7973 - val_loss: 0.6473
Epoch 65/75
30/30              9s 292ms/step -
accuracy: 0.9363 - loss: 0.2310 - val_accuracy: 0.8024 - val_loss: 0.6486
Epoch 66/75
30/30              9s 290ms/step -
accuracy: 0.9420 - loss: 0.2209 - val_accuracy: 0.7997 - val_loss: 0.6727
Epoch 67/75
30/30              8s 281ms/step -
accuracy: 0.9409 - loss: 0.2183 - val_accuracy: 0.8076 - val_loss: 0.6572
Epoch 68/75
30/30              9s 287ms/step -
accuracy: 0.9434 - loss: 0.2107 - val_accuracy: 0.8038 - val_loss: 0.6639
Epoch 69/75
30/30              9s 290ms/step -
accuracy: 0.9432 - loss: 0.2093 - val_accuracy: 0.8027 - val_loss: 0.6766
Epoch 70/75
30/30              9s 297ms/step -
accuracy: 0.9424 - loss: 0.2099 - val_accuracy: 0.8000 - val_loss: 0.6554
Epoch 71/75
30/30              9s 290ms/step -
accuracy: 0.9472 - loss: 0.1927 - val_accuracy: 0.8111 - val_loss: 0.6439
Epoch 72/75
```

```
30/30                 8s 280ms/step -
accuracy: 0.9513 - loss: 0.1921 - val_accuracy: 0.8070 - val_loss: 0.6433
Epoch 73/75
30/30                 9s 285ms/step -
accuracy: 0.9482 - loss: 0.1909 - val_accuracy: 0.8035 - val_loss: 0.6436
Epoch 74/75
30/30                 9s 294ms/step -
accuracy: 0.9511 - loss: 0.1872 - val_accuracy: 0.8032 - val_loss: 0.6842
Epoch 75/75
30/30                 9s 292ms/step -
accuracy: 0.9575 - loss: 0.1712 - val_accuracy: 0.8051 - val_loss: 0.6649
```

[49]:
```
plt.figure(figsize=(10, 10))

plt.subplot(2, 2, 1)
plt.plot(history.history['loss'], label='Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Training - Loss Function')

plt.subplot(2, 2, 2)
plt.plot(history.history['accuracy'], label='Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.title('Train - Accuracy')
```

[49]:
```
Text(0.5, 1.0, 'Train - Accuracy')
```

[55]:
```python
from tensorflow.keras.preprocessing.image import img_to_array, load_img
import joblib

# Load and preprocess the image
image_path = r"C:\Users\svish\Downloads\images.jpeg"  # Path to the image you
↪want to classify
image = load_img(image_path, target_size=image_shape[:2])  # Resize to (height,
↪width)
image_array = img_to_array(image)  # Convert to array
image_array = image_array / 255  # Normalize (if your model was trained with
↪normalized images)
image_array = np.expand_dims(image_array, axis=0)  # Add batch dimension

# Predict the label
predictions = cnn_model.predict(image_array)
predicted_class_index = np.argmax(predictions, axis=1)[0]  # Get the index of
↪the highest probability

# Print the predicted class
print(f"Predicted label: {class_names[predicted_class_index]}")
joblib.dump(cnn_model,r"C:
↪\Users\Public\Documents\Python_project\plant_diseases_detector.pkl")
```

```
1/1            0s 42ms/step
Predicted label: Pepper__bell___healthy
```

[55]:
```
['C:\\Users\\Public\\Documents\\Python_project\\plant_diseases_detector.pkl']
```

[3]:
```python
import joblib
```

[7]:
```python
cnn_model = joblib.load(r"C:
↪\Users\Public\Documents\Python_project\plant_diseases_detector.pkl")
```

[3]:
```python
import psutil
memo_info = psutil.virtual_memory()
available_memo_gb = memo_info.available /(1024**3)
print("ram: ",available_memo_gb)
```

```
ram:  1.6958770751953125
```

[5]:
```python
# Image dimensions (e.g., 256x256 RGB image)
height, width, channels = 64, 64, 3
bytes_per_pixel = 4  # for float32 data type

# Calculate memory per image (in MB)
memory_per_image_mb = (height * width * channels * bytes_per_pixel) / (1024 **
↪2)
print(f"Memory per image: {memory_per_image_mb:.2f} MB")
```

```python
# Adjust batch size based on available RAM
available_ram_gb = 1  # replace with your actual available RAM in GB
max_batch_size = int((available_ram_gb * 1024) / memory_per_image_mb)
print(f"Approximate maximum batch size: {max_batch_size}")
```

```
Memory per image: 0.05 MB
Approximate maximum batch size: 21845
```

[ ]:

```python

```