CIDL_project

September 12, 2023

[ ]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import os
import datetime
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, f1_score , confusion_matrix

import itertools
```

[ ]:
```
# from google.colab import drive
# # drive.mount('/content/drive/Shareddrives/CIDL_Project', force_remount=True)
# drive.mount('/content/drive', force_remount=True)

# print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
print(tf.config.list_physical_devices('GPU'))

# gpu_devices = tf.config.experimental.list_physical_devices('GPU')
# for device in gpu_devices:
#     tf.config.experimental.set_memory_growth(device, True)
```

```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

```
2023-09-12 16:45:11.959905: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:995]
successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero. See more at
https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-
pci#L344-L355
2023-09-12 16:45:12.548703: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:995]
successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero. See more at
https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-
pci#L344-L355
```

```
2023-09-12 16:45:12.548919: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:995]
successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero. See more at
https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-
pci#L344-L355
2023-09-12 16:45:12.548703: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:995]
successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero. See more at
https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-
pci#L344-L355
2023-09-12 16:45:12.548919: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:995]
successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero. See more at
https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-
pci#L344-L355
```

[ ]:
```python
PATH = os.getcwd()

# dataPath = os.path.join(PATH, 'drive', 'Shareddrives', 'CIDL_Project',␣
 ↪'leaves')
dataPath = os.path.join(PATH, 'archive')

scratchPath = os.path.join(PATH, 'models', 'from_scratch')

fExtractorPath = os.path.join(PATH, 'models', 'feature_extraction')

fineTunedPath = os.path.join(PATH, 'models', 'fine_tuned')

docPath = os.path.join(PATH, 'doc')

imgPath = os.path.join(docPath, 'img')

print(dataPath)
print(scratchPath)
print(fExtractorPath)
print(fineTunedPath)
print(docPath)
print(imgPath)
```

```
/home/chuck/Documents/Project/archive
/home/chuck/Documents/Project/models/from_scratch
/home/chuck/Documents/Project/models/feature_extraction
/home/chuck/Documents/Project/models/fine_tuned
/home/chuck/Documents/Project/doc
/home/chuck/Documents/Project/doc/img
```

[ ]:
```python
import shutil
import random

# Definisci il percorso della cartella principale del tuo dataset
dataset_dir = os.path.join(dataPath, "color_prova")

# Definisci il percorso delle cartelle di addestramento, test e convalida
train_dir = dataPath + '/train'
test_dir = dataPath + '/test'
validation_dir = dataPath + '/validation'

# Definisci le proporzioni per la divisione dei dati
train_ratio = 0.8
test_ratio = 0.15
validation_ratio = 0.15

# Crea le cartelle di addestramento, test e convalida se non esistono gia'
os.makedirs(train_dir, exist_ok=True)
os.makedirs(test_dir, exist_ok=True)
os.makedirs(validation_dir, exist_ok=True)

# Itera attraverso le sottocartelle delle classi nel dataset
for class_folder in os.listdir(dataset_dir):
    class_path = os.path.join(dataset_dir, class_folder)

    # Assicurati che sia una cartella
    if os.path.isdir(class_path):
        # Ottieni la lista dei file nella sottocartella
        files = os.listdir(class_path)
        random.shuffle(files)  # Mescola i file per rendere la divisione casuale

        # Calcola i punti di divisione
        num_files = len(files)
        num_train = int(num_files * train_ratio)
        valid_test = int(num_files * test_ratio)

        # Suddividi i file nelle cartelle di addestramento, test e convalida
        train_files = files[:num_train]
        validation_files = files[num_train:num_train + valid_test]
        test_files = files[num_train + valid_test:]

        # Crea le sottocartelle delle classi nelle cartelle di addestramento,␣
↪test e convalida
        for folder in [train_dir, test_dir, validation_dir]:
            class_subfolder = os.path.join(folder, class_folder)
            os.makedirs(class_subfolder, exist_ok=True)
```

```
        # Copia i file nelle sottocartelle appropriate
        for file in train_files:
            src_path = os.path.join(class_path, file)
            dest_path = os.path.join(train_dir, class_folder, file)
            shutil.copy(src_path, dest_path)

        for file in test_files:
            src_path = os.path.join(class_path, file)
            dest_.take(1)path = os.path.join(test_dir, class_folder, file)
            shutil.copy(src_path, dest_path)

        for file in validation_files:
            src_path = os.path.join(class_path, file)
            dest_path = os.path.join(validation_dir, class_folder, file)
            shutil.copy(src_path, dest_path)

# Rimuovi tutte le altre directory presenti nella cartella principale del␣
 ↪dataset
for item in os.listdir(train_dir):
    item_path = os.path.join(train_dir, item)
    if os.path.isdir(item_path) and (item == "train" or item == "validation" or␣
 ↪item == "test"):
        shutil.rmtree(item_path)
for item in os.listdir(test_dir):
    item_path = os.path.join(train_dir, item)
    if os.path.isdir(item_path) and (item == "train" or item == "validation" or␣
 ↪item == "test"):
        shutil.rmtree(item_path)
for item in os.listdir(validation_dir):
    item_path = os.path.join(train_dir, item)
    if os.path.isdir(item_path) and (item == "train" or item == "validation" or␣
 ↪item == "test"):
        shutil.rmtree(item_path)
```

[ ]:
```
BATCH_SIZE = 32
IMAGE_HEIGHT = 256
IMAGE_WIDTH = 256
```

[ ]:
```
diseases = os.listdir(os.path.join(dataPath, "train"))
nums = {}
for disease in diseases:
    nums[disease] = len(os.listdir(os.path.join(dataPath, "train") + '/' +␣
 ↪disease))
img_per_class = pd.DataFrame(nums.values(), index=nums.keys(), columns=["no. of␣
 ↪images"])
img_per_class
```

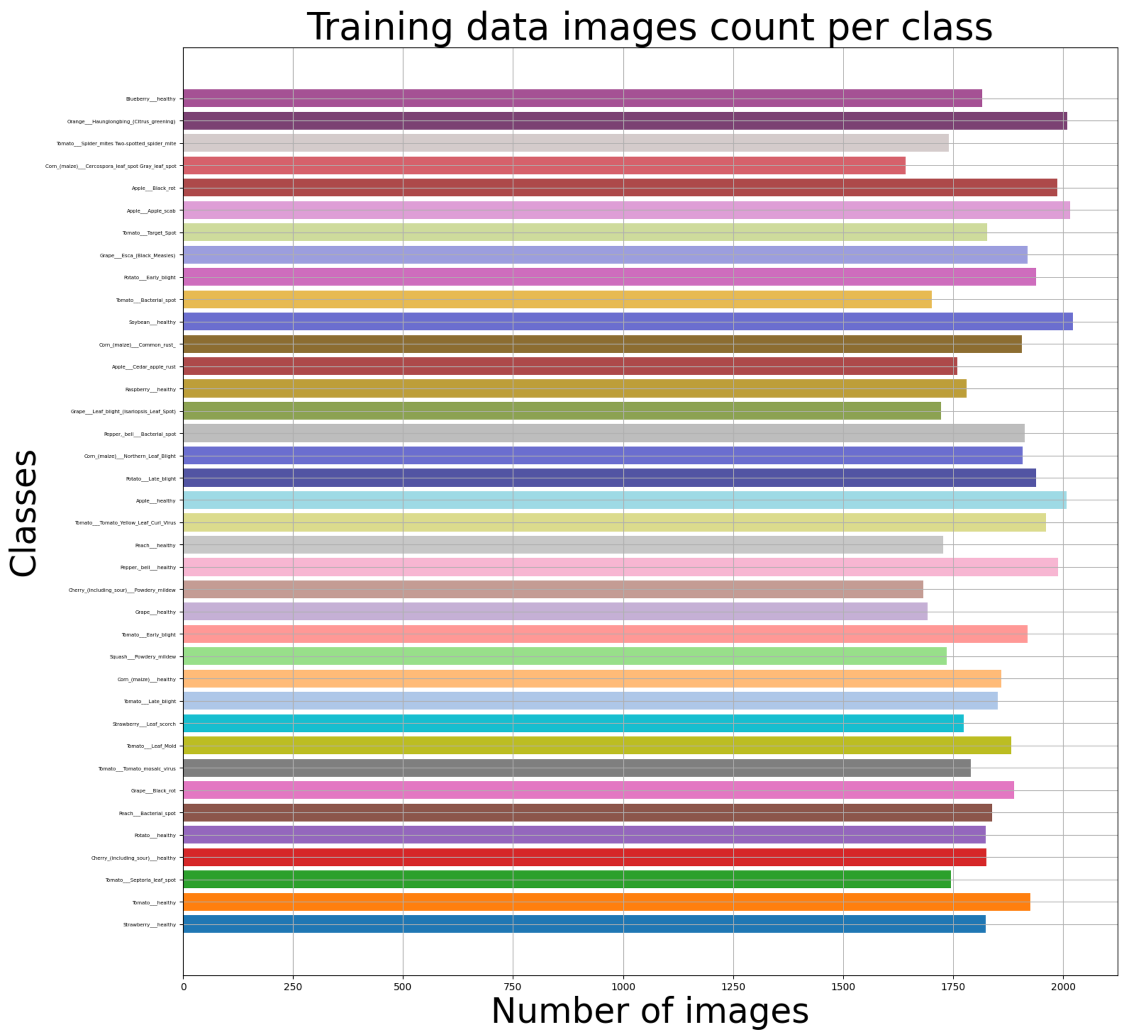
[ ]:

```
                                                 no. of images
Strawberry___healthy                                      1824
Tomato___healthy                                          1926
Tomato___Septoria_leaf_spot                               1745
Cherry_(including_sour)___healthy                         1826
Potato___healthy                                          1824
Peach___Bacterial_spot                                    1838
Grape___Black_rot                                         1888
Tomato___Tomato_mosaic_virus                              1790
Tomato___Leaf_Mold                                        1882
Strawberry___Leaf_scorch                                  1774
Tomato___Late_blight                                      1851
Corn_(maize)___healthy                                    1859
Squash___Powdery_mildew                                   1736
Tomato___Early_blight                                     1920
Grape___healthy                                           1692
Cherry_(including_sour)___Powdery_mildew                  1683
Pepper,_bell___healthy                                    1988
Peach___healthy                                           1728
Tomato___Tomato_Yellow_Leaf_Curl_Virus                    1961
Apple___healthy                                           2008
Potato___Late_blight                                      1939
Corn_(maize)___Northern_Leaf_Blight                       1908
Pepper,_bell___Bacterial_spot                             1913
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)                1722
Raspberry___healthy                                       1781
Apple___Cedar_apple_rust                                  1760
Corn_(maize)___Common_rust_                               1907
Soybean___healthy                                         2022
Tomato___Bacterial_spot                                   1702
Potato___Early_blight                                     1939
Grape___Esca_(Black_Measles)                              1920
Tomato___Target_Spot                                      1827
Apple___Apple_scab                                        2016
Apple___Black_rot                                         1987
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot        1642
Tomato___Spider_mites Two-spotted_spider_mite             1741
Orange___Haunglongbing_(Citrus_greening)                  2010
Blueberry___healthy                                       1816
```

[ ]:

```
colors = [
    "#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#9467bd",
    "#8c564b", "#e377c2", "#7f7f7f", "#bcbd22", "#17becf",
    "#aec7e8", "#ffbb78", "#98df8a", "#ff9896", "#c5b0d5",
    "#c49c94", "#f7b6d2", "#c7c7c7", "#dbdb8d", "#9edae5",
    "#5254a3", "#6b6ecf", "#bdbdbd", "#8ca252", "#bd9e39",
    "#ad494a", "#8c6d31", "#6b6ecf", "#e7ba52", "#ce6dbd",
```

```
    "#9c9ede", "#cedb9c", "#de9ed6", "#ad494a", "#d6616b",
    "#d4cbcb", "#7b4173", "#a55194", "#ce6dbd"
]

index = [n for n in range(38)]

plt.figure(figsize=(17,17))
plt.title("Training data images count per class",fontsize=38)
plt.xlabel('Number of images', fontsize=35)
plt.ylabel('Classes', fontsize=35)
plt.yticks(index, diseases, fontsize=5, rotation=0)
plt.barh(index, [n for n in nums.values()], color=colors)
plt.grid(True)
plt.show()
```



[ ]:
```
train_dataset = keras.utils.image_dataset_from_directory(
    os.path.join(dataPath, 'train'),
    image_size = (IMAGE_HEIGHT, IMAGE_WIDTH),
    batch_size = BATCH_SIZE,
    label_mode = 'categorical'
)

valid_dataset = keras.utils.image_dataset_from_directory(
    os.path.join(dataPath, 'valid'),
    image_size = (IMAGE_HEIGHT, IMAGE_WIDTH),
    batch_size = BATCH_SIZE,
    label_mode = 'categorical'
)

test_dataset = keras.utils.image_dataset_from_directory(
    os.path.join(dataPath, 'test'),
    image_size = (IMAGE_HEIGHT, IMAGE_WIDTH),
    batch_size = BATCH_SIZE,
    label_mode = 'categorical',
    shuffle=False
)
```

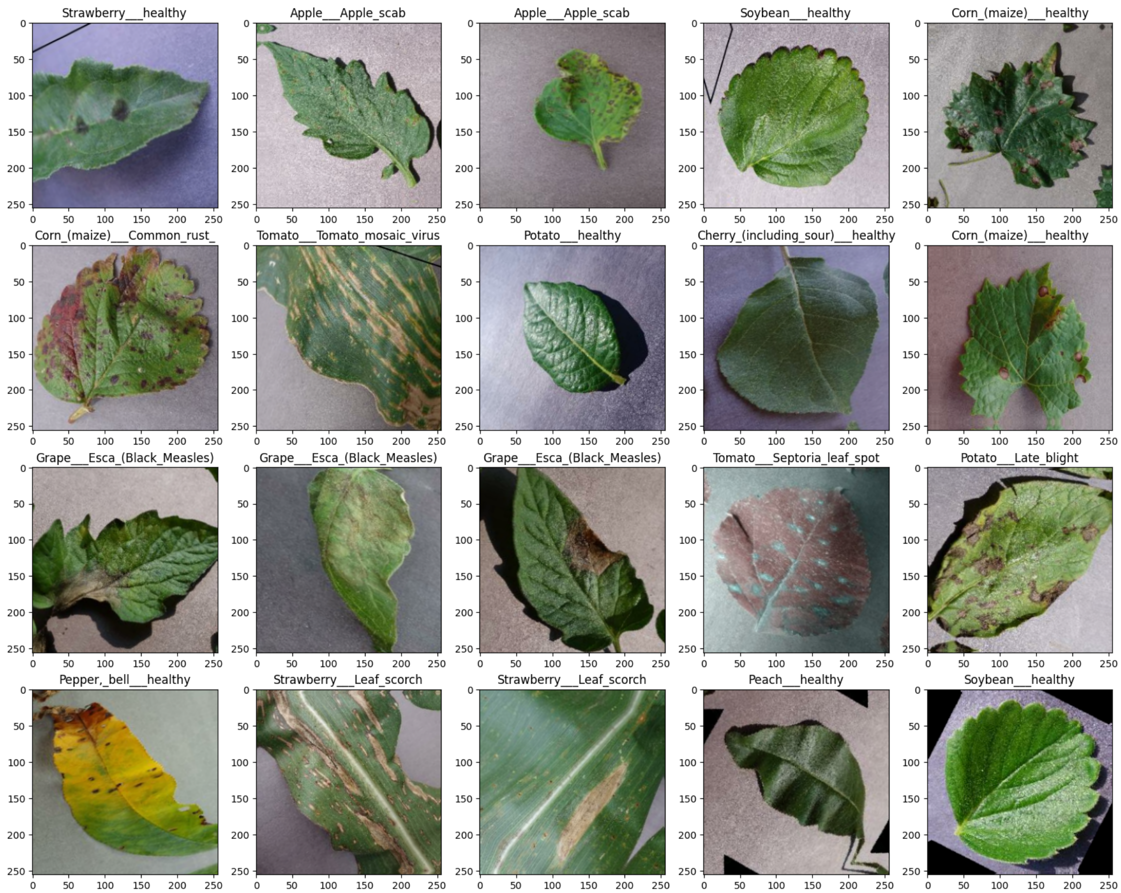
```
Found 70295 files belonging to 38 classes.
Found 17572 files belonging to 38 classes.
Found 5438 files belonging to 38 classes.
```

1 Data Augmentation

[ ]:
```
data_augmentation = keras.Sequential([
  layers.RandomFlip("horizontal"), # Applies horizontal flipping to a random␣
 ↪50% of the images
  layers.RandomRotation(0.1), # Rotates the input images by a random value in␣
 ↪the range[-10%, +10%] (fraction of full circle [-36°, 36°])
  layers.RandomZoom(0.1), # Zooms in or out of the image by a random factor in␣
 ↪the range [-20%, +20%]
  layers.RandomContrast(0.1),
],
name = "AugmentationLayer"
)
```

[ ]:
```
plt.figure(figsize=(20, 20))
for images, y_batch in train_dataset.take(1):
    for i in range(20):
        augmented_images = data_augmentation(images)
        ax = plt.subplot(5, 5, i + 1)
        plt.imshow(augmented_images[i].numpy().astype("uint8"))
```

```
    plt.title(diseases[np.where(y_batch[i]==1)[0][0]])
```



[ ]:
```
from tensorflow.keras import models

model = keras.Sequential()
model.add(data_augmentation)
model.add(layers.Rescaling(1./255))
model.add(layers.Conv2D(32, kernel_size = 3, activation = "relu6", padding = "same", input_shape = (256, 256,3))) # TODO see if using IMG_SIZE is correct
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.BatchNormalization())
model.add(layers.Conv2D(64, kernel_size = 3, activation='relu', padding = "same"))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.BatchNormalization())
model.add(layers.Conv2D(128, kernel_size = 3, activation='relu', padding = "same"))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.BatchNormalization())
```

```
model.add(layers.Conv2D(256, kernel_size = 3, activation='relu', padding =␣
 ↪"same"))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.BatchNormalization())

model.add(layers.Conv2D(512, kernel_size = 3, activation='relu', padding =␣
 ↪"same"))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.BatchNormalization())

model.add(layers.Conv2D(512, kernel_size = 3, activation='relu', padding =␣
 ↪"same"))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.BatchNormalization())

model.add(layers.Flatten())
model.add(layers.Dropout(0.5))

model.add(layers.Dense(256,activation="relu"))
model.add(layers.Dense(38,activation="softmax"))

model.build(input_shape=(None, 256, 256, 3))

model.compile(loss="categorical_crossentropy",
              optimizer="adam",
              metrics=["accuracy"])

model.summary()
```

```
Model: "sequential_6"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 AugmentationLayer (Sequent  (None, 256, 256, 3)       0
 ial)

 rescaling_3 (Rescaling)     (None, 256, 256, 3)       0

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 AugmentationLayer (Sequent  (None, 256, 256, 3)       0
 ial)

 rescaling_3 (Rescaling)     (None, 256, 256, 3)       0

 conv2d_6 (Conv2D)           (None, 256, 256, 32)      896
```

```
max_pooling2d_6 (MaxPoolin  (None, 128, 128, 32)      0
g2D)

batch_normalization_9 (Bat  (None, 128, 128, 32)      128
chNormalization)

conv2d_7 (Conv2D)           (None, 128, 128, 64)      18496

max_pooling2d_7 (MaxPoolin  (None, 64, 64, 64)        0
g2D)

batch_normalization_10 (Ba  (None, 64, 64, 64)        256
tchNormalization)

conv2d_8 (Conv2D)           (None, 64, 64, 128)       73856

max_pooling2d_8 (MaxPoolin  (None, 32, 32, 128)       0
g2D)

batch_normalization_11 (Ba  (None, 32, 32, 128)       512
tchNormalization)

conv2d_9 (Conv2D)           (None, 32, 32, 256)       295168

max_pooling2d_9 (MaxPoolin  (None, 16, 16, 256)       0
g2D)

batch_normalization_12 (Ba  (None, 16, 16, 256)       1024
tchNormalization)

conv2d_10 (Conv2D)          (None, 16, 16, 512)       1180160

max_pooling2d_10 (MaxPooli  (None, 8, 8, 512)         0
ng2D)

batch_normalization_13 (Ba  (None, 8, 8, 512)         2048
tchNormalization)

conv2d_11 (Conv2D)          (None, 8, 8, 512)         2359808

max_pooling2d_11 (MaxPooli  (None, 4, 4, 512)         0
ng2D)

batch_normalization_14 (Ba  (None, 4, 4, 512)         2048
tchNormalization)

flatten_4 (Flatten)         (None, 8192)              0
```

```
 dropout_4 (Dropout)         (None, 8192)              0

 dense_8 (Dense)             (None, 256)               2097408

 dense_9 (Dense)             (None, 38)                9766

=================================================================
Total params: 6041574 (23.05 MB)
Trainable params: 6038566 (23.04 MB)
Non-trainable params: 3008 (11.75 KB)
_________________________________________________________________
```

[ ]:
```
modelName = "32relu6_64_128_256_512_512_Drp_256d"
modelPath = os.path.join(scratchPath, modelName + ".keras")

save_best_model = tf.keras.callbacks.ModelCheckpoint(modelPath, verbose=True,␣
 ↪monitor='val_loss', save_best_only=True, save_weights_only=True)

earlyStopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',␣
 ↪patience=10)

reduceLR = tf.keras.callbacks.ReduceLROnPlateau(monitor = 'val_loss', factor =␣
 ↪0.2, patience = 0.2, mode = 'min')

history = model.fit(
    train_dataset,
    epochs=20,
    validation_data=valid_dataset,
    validation_steps=len(valid_dataset),
    callbacks=[earlyStopping, save_best_model] #, reduceLR],
  )
```

```
Epoch 1/20
2197/2197 [==============================] - ETA: 0s - loss: 0.9826 - accuracy:
0.7176
Epoch 1: val_loss improved from inf to 1.89372, saving model to /home/chuck/Docu
ments/Project/models/from_scratch/32relu6_64_128_256_512_512_Drp_256d.keras
2197/2197 [==============================] - 389s 175ms/step - loss: 0.9826 -
accuracy: 0.7176 - val_loss: 1.8937 - val_accuracy: 0.6201
Epoch 2/20
2197/2197 [==============================] - ETA: 0s - loss: 0.3943 - accuracy:
0.8775
Epoch 2: val_loss improved from 1.89372 to 0.86715, saving model to /home/chuck/
Documents/Project/models/from_scratch/32relu6_64_128_256_512_512_Drp_256d.keras
2197/2197 [==============================] - 384s 175ms/step - loss: 0.3943 -
accuracy: 0.8775 - val_loss: 0.8671 - val_accuracy: 0.7749
```

```
Epoch 3/20
2197/2197 [==============================] - ETA: 0s - loss: 0.2718 - accuracy:
0.9146
Epoch 3: val_loss improved from 0.86715 to 0.47868, saving model to /home/chuck/
Documents/Project/models/from_scratch/32relu6_64_128_256_512_512_Drp_256d.keras
2197/2197 [==============================] - 385s 175ms/step - loss: 0.2718 -
accuracy: 0.9146 - val_loss: 0.4787 - val_accuracy: 0.8712
Epoch 4/20
2197/2197 [==============================] - ETA: 0s - loss: 0.2158 - accuracy:
0.9333
Epoch 4: val_loss did not improve from 0.47868
2197/2197 [==============================] - 381s 173ms/step - loss: 0.2158 -
accuracy: 0.9333 - val_loss: 0.7814 - val_accuracy: 0.8090
Epoch 5/20
2197/2197 [==============================] - ETA: 0s - loss: 0.1716 - accuracy:
0.9457
Epoch 5: val_loss improved from 0.47868 to 0.34314, saving model to /home/chuck/
Documents/Project/models/from_scratch/32relu6_64_128_256_512_512_Drp_256d.keras
2197/2197 [==============================] - 385s 175ms/step - loss: 0.1716 -
accuracy: 0.9457 - val_loss: 0.3431 - val_accuracy: 0.9105
Epoch 6/20
2197/2197 [==============================] - ETA: 0s - loss: 0.1364 - accuracy:
0.9571
Epoch 6: val_loss did not improve from 0.34314
2197/2197 [==============================] - 383s 174ms/step - loss: 0.1364 -
accuracy: 0.9571 - val_loss: 1.2772 - val_accuracy: 0.7646
Epoch 7/20
2197/2197 [==============================] - ETA: 0s - loss: 0.1241 - accuracy:
0.9605
Epoch 7: val_loss did not improve from 0.34314
2197/2197 [==============================] - 391s 178ms/step - loss: 0.1241 -
accuracy: 0.9605 - val_loss: 0.8003 - val_accuracy: 0.8197
Epoch 8/20
2197/2197 [==============================] - ETA: 0s - loss: 0.1086 - accuracy:
0.9654
Epoch 8: val_loss did not improve from 0.34314
2197/2197 [==============================] - 389s 177ms/step - loss: 0.1086 -
accuracy: 0.9654 - val_loss: 1.0888 - val_accuracy: 0.7968
Epoch 9/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0990 - accuracy:
0.9694
Epoch 9: val_loss did not improve from 0.34314
2197/2197 [==============================] - 396s 180ms/step - loss: 0.0990 -
accuracy: 0.9694 - val_loss: 0.3893 - val_accuracy: 0.9177
Epoch 10/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0841 - accuracy:
0.9739
Epoch 10: val_loss did not improve from 0.34314
```

```
2197/2197 [==============================] - 388s 177ms/step - loss: 0.0841 -
accuracy: 0.9739 - val_loss: 1.7239 - val_accuracy: 0.7911
Epoch 11/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0768 - accuracy:
0.9762
Epoch 11: val_loss did not improve from 0.34314
2197/2197 [==============================] - 383s 174ms/step - loss: 0.0768 -
accuracy: 0.9762 - val_loss: 1.0023 - val_accuracy: 0.8545
Epoch 12/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0707 - accuracy:
0.9781
Epoch 12: val_loss improved from 0.34314 to 0.25615, saving model to /home/chuck
/Documents/Project/models/from_scratch/32relu6_64_128_256_512_512_Drp_256d.keras
2197/2197 [==============================] - 386s 175ms/step - loss: 0.0707 -
accuracy: 0.9781 - val_loss: 0.2561 - val_accuracy: 0.9302
Epoch 13/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0647 - accuracy:
0.9790
Epoch 13: val_loss improved from 0.25615 to 0.14344, saving model to /home/chuck
/Documents/Project/models/from_scratch/32relu6_64_128_256_512_512_Drp_256d.keras
2197/2197 [==============================] - 384s 175ms/step - loss: 0.0647 -
accuracy: 0.9790 - val_loss: 0.1434 - val_accuracy: 0.9582
Epoch 14/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0608 - accuracy:
0.9806
Epoch 14: val_loss did not improve from 0.14344
2197/2197 [==============================] - 385s 175ms/step - loss: 0.0608 -
accuracy: 0.9806 - val_loss: 1.7192 - val_accuracy: 0.7169
Epoch 15/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0539 - accuracy:
0.9830
Epoch 15: val_loss did not improve from 0.14344
2197/2197 [==============================] - 389s 177ms/step - loss: 0.0539 -
accuracy: 0.9830 - val_loss: 5.3982 - val_accuracy: 0.7331
Epoch 16/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0523 - accuracy:
0.9836
Epoch 16: val_loss did not improve from 0.14344
2197/2197 [==============================] - 397s 181ms/step - loss: 0.0523 -
accuracy: 0.9836 - val_loss: 0.1652 - val_accuracy: 0.9547
Epoch 17/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0491 - accuracy:
0.9846
Epoch 17: val_loss did not improve from 0.14344
2197/2197 [==============================] - 398s 181ms/step - loss: 0.0491 -
accuracy: 0.9846 - val_loss: 0.5955 - val_accuracy: 0.9221
Epoch 18/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0462 - accuracy:
```

```
0.9855
Epoch 18: val_loss did not improve from 0.14344
2197/2197 [==============================] - 396s 180ms/step - loss: 0.0462 -
accuracy: 0.9855 - val_loss: 0.7404 - val_accuracy: 0.8638
Epoch 19/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0470 - accuracy:
0.9859
Epoch 19: val_loss did not improve from 0.14344
2197/2197 [==============================] - 396s 180ms/step - loss: 0.0470 -
accuracy: 0.9859 - val_loss: 0.3681 - val_accuracy: 0.9199
Epoch 20/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0398 - accuracy:
0.9877
Epoch 20: val_loss improved from 0.14344 to 0.13112, saving model to /home/chuck
/Documents/Project/models/from_scratch/32relu6_64_128_256_512_512_Drp_256d.keras
2197/2197 [==============================] - 398s 181ms/step - loss: 0.0398 -
accuracy: 0.9877 - val_loss: 0.1311 - val_accuracy: 0.9627
```

[ ]:

```python
# Define needed variables
tr_acc = history.history['accuracy']
tr_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']
index_loss = np.argmin(val_loss)
val_lowest = val_loss[index_loss]
index_acc = np.argmax(val_acc)
acc_highest = val_acc[index_acc]
Epochs = [i+1 for i in range(len(tr_acc))]
loss_label = f'best epoch= {str(index_loss + 1)}'
acc_label = f'best epoch= {str(index_acc + 1)}'

# Plot training history
plt.figure(figsize= (20, 8))
plt.style.use('fivethirtyeight')

plt.subplot(1, 2, 1)
plt.plot(Epochs, tr_loss, 'r', label= 'Training loss')
plt.plot(Epochs, val_loss, 'g', label= 'Validation loss')
plt.scatter(index_loss + 1, val_lowest, s= 150, c= 'blue', label= loss_label)
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(Epochs, tr_acc, 'r', label= 'Training Accuracy')
plt.plot(Epochs, val_acc, 'g', label= 'Validation Accuracy')
```

```
plt.scatter(index_acc + 1 , acc_highest, s= 150, c= 'blue', label= acc_label)
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout

plt.savefig(os.path.join(imgPath, "scratch", modelName))
plt.show()

text_to_add = "* " + modelName + " | val_loss: " + "{:.4f}".format(val_lowest)␣
↪+ " | val_accuracy: **" + "{:.4f}".format(val_acc[index_loss]) + "**\n"
print(text_to_add)

with open(os.path.join(docPath, "accuracies.md"), "a") as file:
    file.write(text_to_add)
```


```
* 32relu6_64_128_256_512_512_Drp_256d | val_loss: 0.1311 | val_accuracy:
**0.9627**
```

2 Feature extraction

Used model: EfficientNetB5

[ ]:
```
cnn_base = tf.keras.applications.VGG16(
    include_top=False,
    weights="imagenet",
    input_shape=(IMAGE_HEIGHT, IMAGE_WIDTH, 3),
```

```python
    pooling='max',
)
cnn_base.summary()
```

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 256, 256, 3)]     0

 block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792

 block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928

 block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0

 block2_conv1 (Conv2D)       (None, 128, 128, 128)     73856

 block2_conv2 (Conv2D)       (None, 128, 128, 128)     147584

 block2_pool (MaxPooling2D)  (None, 64, 64, 128)       0

 block3_conv1 (Conv2D)       (None, 64, 64, 256)       295168
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 256, 256, 3)]     0

 block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792

 block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928

 block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0

 block2_conv1 (Conv2D)       (None, 128, 128, 128)     73856

 block2_conv2 (Conv2D)       (None, 128, 128, 128)     147584

 block2_pool (MaxPooling2D)  (None, 64, 64, 128)       0

 block3_conv1 (Conv2D)       (None, 64, 64, 256)       295168

 block3_conv2 (Conv2D)       (None, 64, 64, 256)       590080

 block3_conv3 (Conv2D)       (None, 64, 64, 256)       590080

 block3_pool (MaxPooling2D)  (None, 32, 32, 256)       0
```

```
 block4_conv1 (Conv2D)       (None, 32, 32, 512)       1180160

 block4_conv2 (Conv2D)       (None, 32, 32, 512)       2359808

 block4_conv3 (Conv2D)       (None, 32, 32, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 16, 16, 512)       0

 block5_conv1 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_conv2 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_conv3 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 8, 8, 512)         0

 global_max_pooling2d_1 (Gl  (None, 512)               0
 obalMaxPooling2D)

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

[ ]:
```
resize_and_rescale = tf.keras.Sequential([
  layers.Resizing(IMAGE_HEIGHT, IMAGE_WIDTH),
  layers.Rescaling(1./255)
])

cnn_base.trainable = False

print('This is the number of trainable weights '
      'after freezing the conv base:', sum(np.prod(x.shape) for x in cnn_base.
 ↪trainable_weights))
```

```
This is the number of trainable weights after freezing the conv base: 0
```

[ ]:
```
pretrained_model = tf.keras.Sequential([
    resize_and_rescale,
    data_augmentation,
    cnn_base,
    layers.Flatten(),
    layers.Dense(256, activation="relu"),
    layers.BatchNormalization(),
    layers.Dropout(0.4),
```

```
    layers.Dense(38, activation = 'softmax')
])


pretrained_model.compile(
    loss = 'categorical_crossentropy',
    optimizer = 'adam',
    metrics = ["accuracy"]
)

pretrained_model.build(input_shape=(None, IMAGE_HEIGHT, IMAGE_WIDTH, 3))

pretrained_model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_4 (Sequential)   (None, 256, 256, 3)       0

 AugmentationLayer (Sequent  (None, 256, 256, 3)       0
 ial)

 vgg16 (Functional)          (None, 512)               14714688

 flatten_3 (Flatten)         (None, 512)               0


_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_4 (Sequential)   (None, 256, 256, 3)       0

 AugmentationLayer (Sequent  (None, 256, 256, 3)       0
 ial)

 vgg16 (Functional)          (None, 512)               14714688

 flatten_3 (Flatten)         (None, 512)               0

 dense_6 (Dense)             (None, 256)               131328

 batch_normalization_8 (Bat  (None, 256)               1024
 chNormalization)

 dropout_3 (Dropout)         (None, 256)               0

 dense_7 (Dense)             (None, 38)                9766
```

```
=================================================================
Total params: 14856806 (56.67 MB)
Trainable params: 141606 (553.15 KB)
Non-trainable params: 14715200 (56.13 MB)
_________________________________________________________________
```

[ ]:
```
modelName = "FE_base_256d"
modelPath = os.path.join(fExtractorPath, modelName + ".keras")

save_best_model = tf.keras.callbacks.ModelCheckpoint(modelPath,␣
↪monitor='val_loss', verbose = 1, save_best_only=True, save_weights_only=True)

earlyStopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',␣
↪patience=10)

reduceLR = tf.keras.callbacks.ReduceLROnPlateau(monitor = 'val_loss', factor =␣
↪0.2, patience = 0.2, mode = 'min')

history = pretrained_model.fit(
    train_dataset,
    epochs=20,
    validation_data=valid_dataset,
    validation_steps=len(valid_dataset),
    callbacks=[earlyStopping, save_best_model], #, reduceLR],
  )
```

```
Epoch 1/20

2023-09-09 19:44:59.956793: W tensorflow/tsl/framework/bfc_allocator.cc:296]
Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.27GiB with
freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.
2023-09-09 19:45:02.991876: W tensorflow/tsl/framework/bfc_allocator.cc:296]
Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.27GiB with
freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.
2023-09-09 19:45:02.991944: W tensorflow/tsl/framework/bfc_allocator.cc:296]
Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.27GiB with
freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.

2196/2197 [============================>.] - ETA: 0s - loss: 0.8083 - accuracy:
0.7582

2023-09-09 19:55:34.038381: W tensorflow/tsl/framework/bfc_allocator.cc:296]
Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.27GiB with
freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.
```

```
2023-09-09 19:55:34.533007: W tensorflow/tsl/framework/bfc_allocator.cc:296]
Allocator (GPU_0_bfc) ran out of memory trying to allocate 3.25GiB with
freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.
2023-09-09 19:55:34.533073: W tensorflow/tsl/framework/bfc_allocator.cc:296]
Allocator (GPU_0_bfc) ran out of memory trying to allocate 3.25GiB with
freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.
```

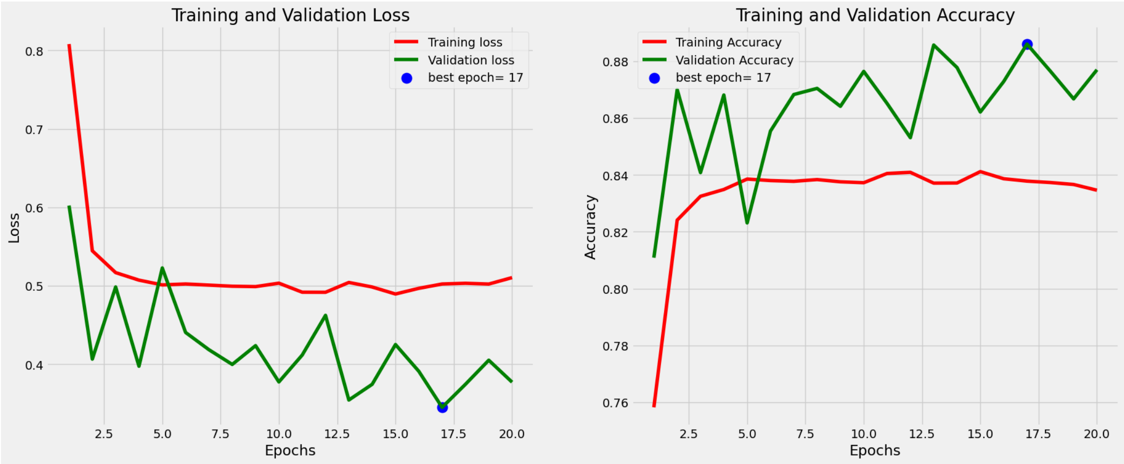
```
2197/2197 [==============================] - ETA: 0s - loss: 0.8082 - accuracy:
0.7582
Epoch 1: val_loss improved from inf to 0.60236, saving model to
/home/chuck/Documents/Project/models/feature_extraction/FE_base_256d.keras
2197/2197 [==============================] - 789s 354ms/step - loss: 0.8082 -
accuracy: 0.7582 - val_loss: 0.6024 - val_accuracy: 0.8109
Epoch 2/20
2197/2197 [==============================] - ETA: 0s - loss: 0.5447 - accuracy:
0.8241
Epoch 2: val_loss improved from 0.60236 to 0.40663, saving model to
/home/chuck/Documents/Project/models/feature_extraction/FE_base_256d.keras
2197/2197 [==============================] - 775s 353ms/step - loss: 0.5447 -
accuracy: 0.8241 - val_loss: 0.4066 - val_accuracy: 0.8701
Epoch 3/20
2197/2197 [==============================] - ETA: 0s - loss: 0.5169 - accuracy:
0.8325
Epoch 3: val_loss did not improve from 0.40663
2197/2197 [==============================] - 774s 352ms/step - loss: 0.5169 -
accuracy: 0.8325 - val_loss: 0.4986 - val_accuracy: 0.8408
Epoch 4/20
2197/2197 [==============================] - ETA: 0s - loss: 0.5073 - accuracy:
0.8349
Epoch 4: val_loss improved from 0.40663 to 0.39754, saving model to
/home/chuck/Documents/Project/models/feature_extraction/FE_base_256d.keras
2197/2197 [==============================] - 772s 351ms/step - loss: 0.5073 -
accuracy: 0.8349 - val_loss: 0.3975 - val_accuracy: 0.8681
Epoch 5/20
2197/2197 [==============================] - ETA: 0s - loss: 0.5012 - accuracy:
0.8386
Epoch 5: val_loss did not improve from 0.39754
2197/2197 [==============================] - 775s 353ms/step - loss: 0.5012 -
accuracy: 0.8386 - val_loss: 0.5228 - val_accuracy: 0.8231
Epoch 6/20
2197/2197 [==============================] - ETA: 0s - loss: 0.5024 - accuracy:
0.8380
Epoch 6: val_loss did not improve from 0.39754
2197/2197 [==============================] - 773s 352ms/step - loss: 0.5024 -
accuracy: 0.8380 - val_loss: 0.4406 - val_accuracy: 0.8555
Epoch 7/20
```

```
2197/2197 [==============================] - ETA: 0s - loss: 0.5010 - accuracy:
0.8378
Epoch 7: val_loss did not improve from 0.39754
2197/2197 [==============================] - 770s 350ms/step - loss: 0.5010 -
accuracy: 0.8378 - val_loss: 0.4186 - val_accuracy: 0.8683
Epoch 8/20
2197/2197 [==============================] - ETA: 0s - loss: 0.4995 - accuracy:
0.8384
Epoch 8: val_loss did not improve from 0.39754
2197/2197 [==============================] - 769s 350ms/step - loss: 0.4995 -
accuracy: 0.8384 - val_loss: 0.3998 - val_accuracy: 0.8705
Epoch 9/20
2197/2197 [==============================] - ETA: 0s - loss: 0.4990 - accuracy:
0.8376
Epoch 9: val_loss did not improve from 0.39754
2197/2197 [==============================] - 772s 351ms/step - loss: 0.4990 -
accuracy: 0.8376 - val_loss: 0.4238 - val_accuracy: 0.8642
Epoch 10/20
2197/2197 [==============================] - ETA: 0s - loss: 0.5034 - accuracy:
0.8373
Epoch 10: val_loss improved from 0.39754 to 0.37753, saving model to
/home/chuck/Documents/Project/models/feature_extraction/FE_base_256d.keras
2197/2197 [==============================] - 770s 350ms/step - loss: 0.5034 -
accuracy: 0.8373 - val_loss: 0.3775 - val_accuracy: 0.8765
Epoch 11/20
2197/2197 [==============================] - ETA: 0s - loss: 0.4919 - accuracy:
0.8405
Epoch 11: val_loss did not improve from 0.37753
2197/2197 [==============================] - 769s 350ms/step - loss: 0.4919 -
accuracy: 0.8405 - val_loss: 0.4118 - val_accuracy: 0.8652
Epoch 12/20
2197/2197 [==============================] - ETA: 0s - loss: 0.4918 - accuracy:
0.8409
Epoch 12: val_loss did not improve from 0.37753
2197/2197 [==============================] - 773s 352ms/step - loss: 0.4918 -
accuracy: 0.8409 - val_loss: 0.4624 - val_accuracy: 0.8531
Epoch 13/20
2197/2197 [==============================] - ETA: 0s - loss: 0.5044 - accuracy:
0.8371
Epoch 13: val_loss improved from 0.37753 to 0.35448, saving model to
/home/chuck/Documents/Project/models/feature_extraction/FE_base_256d.keras
2197/2197 [==============================] - 773s 352ms/step - loss: 0.5044 -
accuracy: 0.8371 - val_loss: 0.3545 - val_accuracy: 0.8857
Epoch 14/20
2197/2197 [==============================] - ETA: 0s - loss: 0.4985 - accuracy:
0.8372
Epoch 14: val_loss did not improve from 0.35448
2197/2197 [==============================] - 771s 351ms/step - loss: 0.4985 -
```

```
accuracy: 0.8372 - val_loss: 0.3745 - val_accuracy: 0.8778
Epoch 15/20
2197/2197 [==============================] - ETA: 0s - loss: 0.4896 - accuracy:
0.8412
Epoch 15: val_loss did not improve from 0.35448
2197/2197 [==============================] - 768s 350ms/step - loss: 0.4896 -
accuracy: 0.8412 - val_loss: 0.4252 - val_accuracy: 0.8622
Epoch 16/20
2197/2197 [==============================] - ETA: 0s - loss: 0.4969 - accuracy:
0.8387
Epoch 16: val_loss did not improve from 0.35448
2197/2197 [==============================] - 771s 351ms/step - loss: 0.4969 -
accuracy: 0.8387 - val_loss: 0.3911 - val_accuracy: 0.8729
Epoch 17/20
2197/2197 [==============================] - ETA: 0s - loss: 0.5024 - accuracy:
0.8378
Epoch 17: val_loss improved from 0.35448 to 0.34515, saving model to
/home/chuck/Documents/Project/models/feature_extraction/FE_base_256d.keras
2197/2197 [==============================] - 714s 325ms/step - loss: 0.5024 -
accuracy: 0.8378 - val_loss: 0.3451 - val_accuracy: 0.8861
Epoch 18/20
2197/2197 [==============================] - ETA: 0s - loss: 0.5033 - accuracy:
0.8373
Epoch 18: val_loss did not improve from 0.34515
2197/2197 [==============================] - 640s 291ms/step - loss: 0.5033 -
accuracy: 0.8373 - val_loss: 0.3745 - val_accuracy: 0.8766
Epoch 19/20
2197/2197 [==============================] - ETA: 0s - loss: 0.5023 - accuracy:
0.8367
Epoch 19: val_loss did not improve from 0.34515
2197/2197 [==============================] - 642s 292ms/step - loss: 0.5023 -
accuracy: 0.8367 - val_loss: 0.4052 - val_accuracy: 0.8668
Epoch 20/20
2197/2197 [==============================] - ETA: 0s - loss: 0.5105 - accuracy:
0.8346
Epoch 20: val_loss did not improve from 0.34515
2197/2197 [==============================] - 644s 293ms/step - loss: 0.5105 -
accuracy: 0.8346 - val_loss: 0.3773 - val_accuracy: 0.8770
```

[ ]:
```
# Define needed variables
tr_acc = history.history['accuracy']
tr_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']
index_loss = np.argmin(val_loss)
val_lowest = val_loss[index_loss]
index_acc = np.argmax(val_acc)
```

```
acc_highest = val_acc[index_acc]
Epochs = [i+1 for i in range(len(tr_acc))]
loss_label = f'best epoch= {str(index_loss + 1)}'
acc_label = f'best epoch= {str(index_acc + 1)}'

# Plot training history
plt.figure(figsize= (20, 8))
plt.style.use('fivethirtyeight')

plt.subplot(1, 2, 1)
plt.plot(Epochs, tr_loss, 'r', label= 'Training loss')
plt.plot(Epochs, val_loss, 'g', label= 'Validation loss')
plt.scatter(index_loss + 1, val_lowest, s= 150, c= 'blue', label= loss_label)
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(Epochs, tr_acc, 'r', label= 'Training Accuracy')
plt.plot(Epochs, val_acc, 'g', label= 'Validation Accuracy')
plt.scatter(index_acc + 1 , acc_highest, s= 150, c= 'blue', label= acc_label)
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout

plt.savefig(os.path.join(imgPath, "FExtraction", modelName))
plt.show()

text_to_add = "* " + modelName + " | val_loss: " + "{:.4f}".format(val_lowest)␣
 ↪+ " | val_accuracy: **" + "{:.4f}".format(val_acc[index_loss]) + "**\n"
print(text_to_add)

with open(os.path.join(docPath, "accuracies.md"), "a") as file:
    file.write(text_to_add)
```




```
* FE_base_256d | val_loss: 0.3451 | val_accuracy: **0.8861**
```

3 Fine Tuning

[ ]:
```python
# To recover a specific model
modelName = "FE_base_256d"
modelPath = os.path.join(fExtractorPath, modelName + ".keras")

pretrained_model.load_weights(modelPath)
```

[ ]:
```python
cnn_base.trainable = True


set_trainable = False

for layer in cnn_base.layers:
    if layer.name == 'block5_conv1':
        set_trainable = True
    if set_trainable:
        layer.trainable = True
    else:
        layer.trainable = False

cnn_base.summary()
```

```
Model: "vgg16"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 256, 256, 3)]     0
```

```
 block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792

 block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928

 block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 256, 256, 3)]     0

 block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792

 block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928

 block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0

 block2_conv1 (Conv2D)       (None, 128, 128, 128)     73856

 block2_conv2 (Conv2D)       (None, 128, 128, 128)     147584

 block2_pool (MaxPooling2D)  (None, 64, 64, 128)       0

 block3_conv1 (Conv2D)       (None, 64, 64, 256)       295168

 block3_conv2 (Conv2D)       (None, 64, 64, 256)       590080

 block3_conv3 (Conv2D)       (None, 64, 64, 256)       590080

 block3_pool (MaxPooling2D)  (None, 32, 32, 256)       0

 block4_conv1 (Conv2D)       (None, 32, 32, 512)       1180160

 block4_conv2 (Conv2D)       (None, 32, 32, 512)       2359808

 block4_conv3 (Conv2D)       (None, 32, 32, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 16, 16, 512)       0

 block5_conv1 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_conv2 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_conv3 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 8, 8, 512)         0
```

```
 global_max_pooling2d_1 (Gl  (None, 512)               0
 obalMaxPooling2D)

=================================================================
Total params: 14714688 (56.13 MB)
Trainable params: 7079424 (27.01 MB)
Non-trainable params: 7635264 (29.13 MB)
_________________________________________________________________
```

[ ]:
```
pretrained_model.compile(
    loss = 'categorical_crossentropy',
    optimizer = tf.keras.optimizers.Adam(learning_rate=0.00001),
    metrics = ["accuracy"]
)

pretrained_model.build(input_shape=(None, IMAGE_HEIGHT, IMAGE_WIDTH, 3))

modelName = "FT_base_256d"
modelPath = os.path.join(fineTunedPath, modelName + ".keras")

save_best_model = tf.keras.callbacks.ModelCheckpoint(modelPath,␣
 ↪monitor='val_loss', verbose = 1, save_best_only=True, save_weights_only=True)

earlyStopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',␣
 ↪patience=10)

reduceLR = tf.keras.callbacks.ReduceLROnPlateau(monitor = 'val_loss', factor =␣
 ↪0.2, patience = 2, mode = 'min')

history = pretrained_model.fit(
    train_dataset,
    epochs=20,
    validation_data=valid_dataset,
    validation_steps=len(valid_dataset),
    callbacks=[earlyStopping, save_best_model], #, reduceLR],
  )
```

```
Epoch 1/20
```

```
2023-09-10 16:57:55.309117: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.cc:432] Loaded cuDNN
version 8904
2023-09-10 16:57:55.697143: I tensorflow/tsl/platform/default/subprocess.cc:304]
Start cannot spawn child process: No such file or directory
2023-09-10 16:57:56.716732: W tensorflow/tsl/framework/bfc_allocator.cc:296]
Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.27GiB with
freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.
```

```
2023-09-10 16:57:59.644636: W tensorflow/tsl/framework/bfc_allocator.cc:296]
Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.27GiB with
freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.
2023-09-10 16:57:59.644708: W tensorflow/tsl/framework/bfc_allocator.cc:296]
Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.27GiB with
freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.
2023-09-10 16:58:04.354163: I tensorflow/compiler/xla/service/service.cc:168]
XLA service 0x56075fbe7330 initialized for platform CUDA (this does not
guarantee that XLA will be used). Devices:
2023-09-10 16:58:04.354211: I tensorflow/compiler/xla/service/service.cc:176]
StreamExecutor device (0): NVIDIA GeForce GTX 970, Compute Capability 5.2
2023-09-10 16:58:04.458797: I
tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:255] disabling MLIR
crash reproducer, set env var `MLIR_CRASH_REPRODUCER_DIRECTORY` to enable.
2023-09-10 16:58:04.842899: I tensorflow/tsl/platform/default/subprocess.cc:304]
Start cannot spawn child process: No such file or directory
2023-09-10 16:58:04.967816: I ./tensorflow/compiler/jit/device_compiler.h:186]
Compiled cluster using XLA!  This line is logged at most once for the lifetime
of the process.
```

```
2196/2197 [============================>.] - ETA: 0s - loss: 0.3535 - accuracy:
0.8861
```

```
2023-09-10 17:09:02.963166: W tensorflow/tsl/framework/bfc_allocator.cc:296]
Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.27GiB with
freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.
2023-09-10 17:09:03.543243: W tensorflow/tsl/framework/bfc_allocator.cc:296]
Allocator (GPU_0_bfc) ran out of memory trying to allocate 3.25GiB with
freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.
2023-09-10 17:09:03.543323: W tensorflow/tsl/framework/bfc_allocator.cc:296]
Allocator (GPU_0_bfc) ran out of memory trying to allocate 3.25GiB with
freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.
```

```
2197/2197 [==============================] - ETA: 0s - loss: 0.3535 - accuracy:
0.8861
Epoch 1: val_loss improved from inf to 0.17523, saving model to
/home/chuck/Documents/Project/models/fine_tuned/FT_base_256d.keras
2197/2197 [==============================] - 818s 365ms/step - loss: 0.3535 -
accuracy: 0.8861 - val_loss: 0.1752 - val_accuracy: 0.9455
Epoch 2/20
2197/2197 [==============================] - ETA: 0s - loss: 0.2422 - accuracy:
0.9225
Epoch 2: val_loss improved from 0.17523 to 0.15053, saving model to
/home/chuck/Documents/Project/models/fine_tuned/FT_base_256d.keras
```

```
2197/2197 [==============================] - 800s 364ms/step - loss: 0.2422 -
accuracy: 0.9225 - val_loss: 0.1505 - val_accuracy: 0.9515
Epoch 3/20
2197/2197 [==============================] - ETA: 0s - loss: 0.1911 - accuracy:
0.9402
Epoch 3: val_loss improved from 0.15053 to 0.12264, saving model to
/home/chuck/Documents/Project/models/fine_tuned/FT_base_256d.keras
2197/2197 [==============================] - 796s 362ms/step - loss: 0.1911 -
accuracy: 0.9402 - val_loss: 0.1226 - val_accuracy: 0.9607
Epoch 4/20
2197/2197 [==============================] - ETA: 0s - loss: 0.1614 - accuracy:
0.9494
Epoch 4: val_loss improved from 0.12264 to 0.11391, saving model to
/home/chuck/Documents/Project/models/fine_tuned/FT_base_256d.keras
2197/2197 [==============================] - 794s 362ms/step - loss: 0.1614 -
accuracy: 0.9494 - val_loss: 0.1139 - val_accuracy: 0.9628
Epoch 5/20
2197/2197 [==============================] - ETA: 0s - loss: 0.1378 - accuracy:
0.9578
Epoch 5: val_loss improved from 0.11391 to 0.10958, saving model to
/home/chuck/Documents/Project/models/fine_tuned/FT_base_256d.keras
2197/2197 [==============================] - 791s 360ms/step - loss: 0.1378 -
accuracy: 0.9578 - val_loss: 0.1096 - val_accuracy: 0.9632
Epoch 6/20
2197/2197 [==============================] - ETA: 0s - loss: 0.1226 - accuracy:
0.9639
Epoch 6: val_loss improved from 0.10958 to 0.09463, saving model to
/home/chuck/Documents/Project/models/fine_tuned/FT_base_256d.keras
2197/2197 [==============================] - 788s 358ms/step - loss: 0.1226 -
accuracy: 0.9639 - val_loss: 0.0946 - val_accuracy: 0.9709
Epoch 7/20
2197/2197 [==============================] - ETA: 0s - loss: 0.1079 - accuracy:
0.9684
Epoch 7: val_loss improved from 0.09463 to 0.07693, saving model to
/home/chuck/Documents/Project/models/fine_tuned/FT_base_256d.keras
2197/2197 [==============================] - 795s 362ms/step - loss: 0.1079 -
accuracy: 0.9684 - val_loss: 0.0769 - val_accuracy: 0.9759
Epoch 8/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0952 - accuracy:
0.9727
Epoch 8: val_loss did not improve from 0.07693
2197/2197 [==============================] - 784s 357ms/step - loss: 0.0952 -
accuracy: 0.9727 - val_loss: 0.0782 - val_accuracy: 0.9742
Epoch 9/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0857 - accuracy:
0.9759
Epoch 9: val_loss did not improve from 0.07693
2197/2197 [==============================] - 786s 358ms/step - loss: 0.0857 -
```

```
accuracy: 0.9759 - val_loss: 0.0790 - val_accuracy: 0.9746
Epoch 10/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0763 - accuracy:
0.9794
Epoch 10: val_loss improved from 0.07693 to 0.06993, saving model to
/home/chuck/Documents/Project/models/fine_tuned/FT_base_256d.keras
2197/2197 [==============================] - 777s 354ms/step - loss: 0.0763 -
accuracy: 0.9794 - val_loss: 0.0699 - val_accuracy: 0.9779
Epoch 11/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0695 - accuracy:
0.9813
Epoch 11: val_loss did not improve from 0.06993
2197/2197 [==============================] - 781s 355ms/step - loss: 0.0695 -
accuracy: 0.9813 - val_loss: 0.0820 - val_accuracy: 0.9738
Epoch 12/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0644 - accuracy:
0.9833
Epoch 12: val_loss did not improve from 0.06993
2197/2197 [==============================] - 782s 356ms/step - loss: 0.0644 -
accuracy: 0.9833 - val_loss: 0.1011 - val_accuracy: 0.9696
Epoch 13/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0598 - accuracy:
0.9840
Epoch 13: val_loss improved from 0.06993 to 0.06941, saving model to
/home/chuck/Documents/Project/models/fine_tuned/FT_base_256d.keras
2197/2197 [==============================] - 782s 356ms/step - loss: 0.0598 -
accuracy: 0.9840 - val_loss: 0.0694 - val_accuracy: 0.9776
Epoch 14/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0538 - accuracy:
0.9858
Epoch 14: val_loss did not improve from 0.06941
2197/2197 [==============================] - 777s 353ms/step - loss: 0.0538 -
accuracy: 0.9858 - val_loss: 0.0772 - val_accuracy: 0.9748
Epoch 15/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0512 - accuracy:
0.9865
Epoch 15: val_loss improved from 0.06941 to 0.05706, saving model to
/home/chuck/Documents/Project/models/fine_tuned/FT_base_256d.keras
2197/2197 [==============================] - 773s 352ms/step - loss: 0.0512 -
accuracy: 0.9865 - val_loss: 0.0571 - val_accuracy: 0.9828
Epoch 16/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0463 - accuracy:
0.9879
Epoch 16: val_loss improved from 0.05706 to 0.05100, saving model to
/home/chuck/Documents/Project/models/fine_tuned/FT_base_256d.keras
2197/2197 [==============================] - 781s 356ms/step - loss: 0.0463 -
accuracy: 0.9879 - val_loss: 0.0510 - val_accuracy: 0.9843
Epoch 17/20
```

```
2197/2197 [==============================] - ETA: 0s - loss: 0.0442 - accuracy:
0.9889
Epoch 17: val_loss did not improve from 0.05100
2197/2197 [==============================] - 783s 356ms/step - loss: 0.0442 -
accuracy: 0.9889 - val_loss: 0.0515 - val_accuracy: 0.9839
Epoch 18/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0408 - accuracy:
0.9897
Epoch 18: val_loss did not improve from 0.05100
2197/2197 [==============================] - 783s 356ms/step - loss: 0.0408 -
accuracy: 0.9897 - val_loss: 0.0614 - val_accuracy: 0.9801
Epoch 19/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0385 - accuracy:
0.9906
Epoch 19: val_loss improved from 0.05100 to 0.04805, saving model to
/home/chuck/Documents/Project/models/fine_tuned/FT_base_256d.keras
2197/2197 [==============================] - 811s 369ms/step - loss: 0.0385 -
accuracy: 0.9906 - val_loss: 0.0481 - val_accuracy: 0.9847
Epoch 20/20
2197/2197 [==============================] - ETA: 0s - loss: 0.0366 - accuracy:
0.9912
Epoch 20: val_loss did not improve from 0.04805
2197/2197 [==============================] - 792s 360ms/step - loss: 0.0366 -
accuracy: 0.9912 - val_loss: 0.0501 - val_accuracy: 0.9840
```

[ ]:
```python
# Define needed variables
tr_acc = history.history['accuracy']
tr_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']
index_loss = np.argmin(val_loss)
val_lowest = val_loss[index_loss]
index_acc = np.argmax(val_acc)
acc_highest = val_acc[index_acc]
Epochs = [i+1 for i in range(len(tr_acc))]
loss_label = f'best epoch= {str(index_loss + 1)}'
acc_label = f'best epoch= {str(index_acc + 1)}'

# Plot training history
plt.figure(figsize= (20, 8))
plt.style.use('fivethirtyeight')

plt.subplot(1, 2, 1)
plt.plot(Epochs, tr_loss, 'r', label= 'Training loss')
plt.plot(Epochs, val_loss, 'g', label= 'Validation loss')
plt.scatter(index_loss + 1, val_lowest, s= 150, c= 'blue', label= loss_label)
plt.title('Training and Validation Loss')
```

```
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(Epochs, tr_acc, 'r', label= 'Training Accuracy')
plt.plot(Epochs, val_acc, 'g', label= 'Validation Accuracy')
plt.scatter(index_acc + 1 , acc_highest, s= 150, c= 'blue', label= acc_label)
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout

plt.savefig(os.path.join(imgPath, "fine-tuning", modelName))
plt.show()

text_to_add = "* " + modelName + " | val_loss: " + "{:.4f}".format(val_lowest)␣
↪+ " | val_accuracy: **" + "{:.4f}".format(val_acc[index_loss]) + "**\n"
print(text_to_add)

with open(os.path.join(docPath, "accuracies.md"), "a") as file:
    file.write(text_to_add)
```


```
* FT_base_256d | val_loss: 0.0481 | val_accuracy: **0.9847**
```

4 Comparison on the test set

```
[ ]: cnn_base.summary()
```

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 256, 256, 3)]     0

 block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792

 block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928

 block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0

 block2_conv1 (Conv2D)       (None, 128, 128, 128)     73856

 block2_conv2 (Conv2D)       (None, 128, 128, 128)     147584

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 256, 256, 3)]     0

 block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792

 block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928

 block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0

 block2_conv1 (Conv2D)       (None, 128, 128, 128)     73856

 block2_conv2 (Conv2D)       (None, 128, 128, 128)     147584

 block2_pool (MaxPooling2D)  (None, 64, 64, 128)       0

 block3_conv1 (Conv2D)       (None, 64, 64, 256)       295168

 block3_conv2 (Conv2D)       (None, 64, 64, 256)       590080

 block3_conv3 (Conv2D)       (None, 64, 64, 256)       590080

 block3_pool (MaxPooling2D)  (None, 32, 32, 256)       0

 block4_conv1 (Conv2D)       (None, 32, 32, 512)       1180160
```

```
 block4_conv2 (Conv2D)       (None, 32, 32, 512)       2359808

 block4_conv3 (Conv2D)       (None, 32, 32, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 16, 16, 512)       0

 block5_conv1 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_conv2 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_conv3 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 8, 8, 512)         0

 global_max_pooling2d (Glob  (None, 512)               0
 alMaxPooling2D)

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

[ ]:

```
# Recovery of the best model from scratch
scratchName = "32relu6_64_128_256_512_512_Drp_256d"
path = os.path.join(scratchPath, scratchName + ".keras")

model.load_weights(path)

# Recovery of the best model fine-tuned
fTunedName = "FT_base_256d"
path = os.path.join(fineTunedPath, fTunedName + ".keras")

pretrained_model.load_weights(path)
```

[ ]:

```
# Create an empty list to store the labels
labels_list = []

# Iterate through the dataset using a for loop

for data_item, label in test_dataset:
    labels_list.extend(np.argmax(label.numpy(), axis=1))  # Append the label to␣
 ↪the list

# Now, labels_list contains all the labels in the dataset
print(len(labels_list))
```

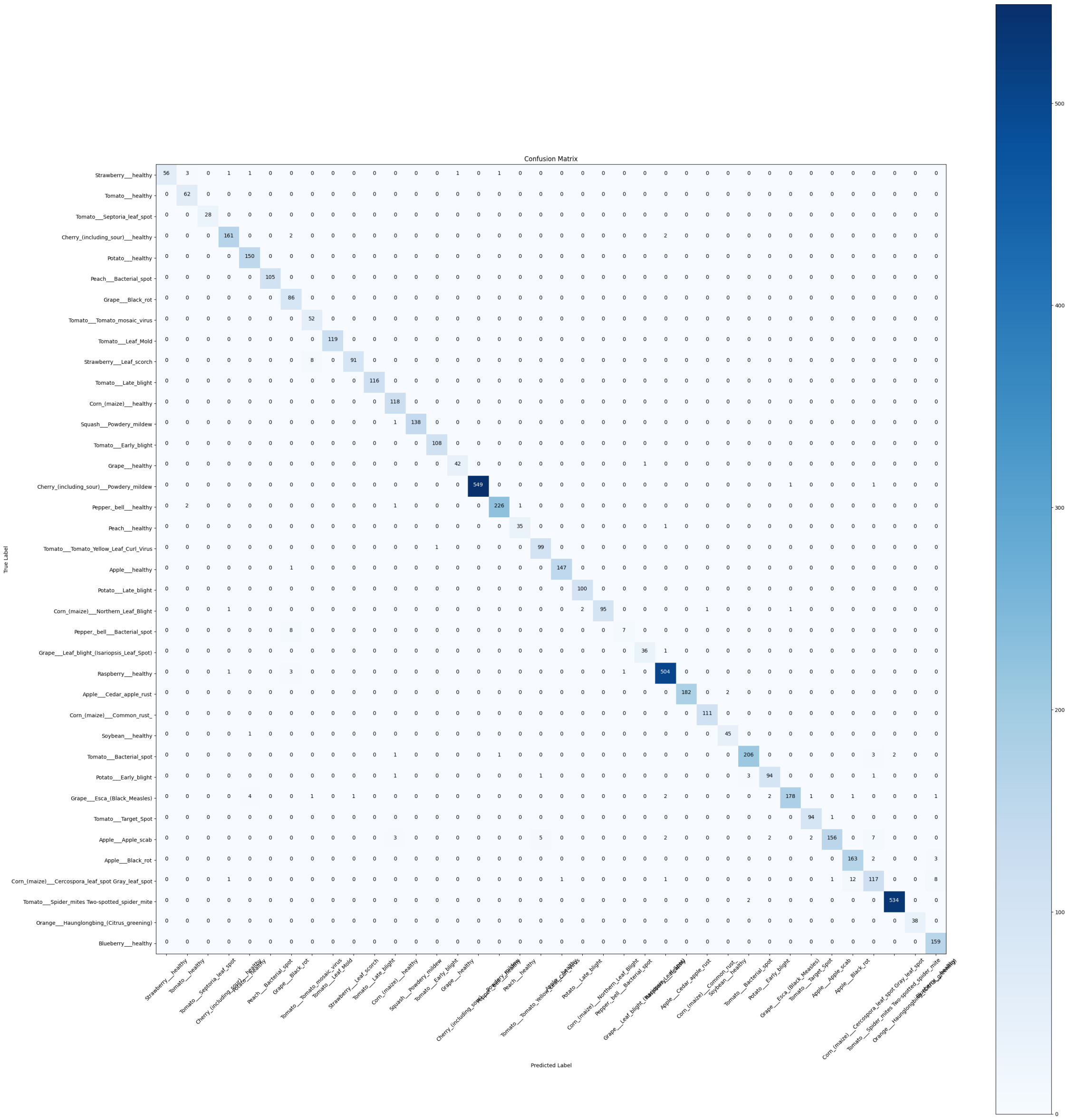
```
5438
```

[ ]:
```python
results = model.evaluate(test_dataset, verbose=0)

print("    Test Loss: {:.5f}".format(results[0]))
print("Test Accuracy: {:.2f}%".format(results[1] * 100))
```

```
    Test Loss: 0.08954
Test Accuracy: 97.59%
```

[ ]:
```python
preds = model.predict(test_dataset)
y_pred = np.argmax(preds, axis=1)

# Confusion matrix
cm = confusion_matrix(labels_list, y_pred)

plt.figure(figsize= (30, 30))
plt.imshow(cm, interpolation= 'nearest', cmap= plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()

tick_marks = np.arange(len(diseases))
plt.xticks(tick_marks, diseases, rotation= 45)
plt.yticks(tick_marks, diseases)


thresh = cm.max() / 2.
for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, cm[i, j], horizontalalignment= 'center', color= 'white' if␣
 ↪cm[i, j] > thresh else 'black')

plt.tight_layout()
plt.ylabel('True Label')
plt.xlabel('Predicted Label')

plt.show()
```

```
 89/170 [==============>…] - ETA: 3s170/170
[==============================] - 7s 39ms/step
```




```
[ ]: f1 = f1_score(labels_list, y_pred, average='macro')
     print("F1 Score:", f1)
     print(classification_report(labels_list, y_pred, target_names=diseases))
```

```
F1 Score: 0.962158352503775
                                                      precision    recall
f1-score   support

                                Strawberry___healthy       1.00      0.89
0.94         63
                                    Tomato___healthy       0.93      1.00
0.96         62
```

Tomato___Septoria_leaf_spot	1.00	1.00	1.00	28
Cherry_(including_sour)___healthy	0.98	0.98	0.98	165
Potato___healthy	0.96	1.00	0.98	150
Peach___Bacterial_spot	1.00	1.00	1.00	105
Grape___Black_rot	0.86	1.00	0.92	86
Tomato___Tomato_mosaic_virus	0.85	1.00	0.92	52
Tomato___Leaf_Mold	1.00	1.00	1.00	119
Strawberry___Leaf_scorch	0.99	0.92	0.95	99
Tomato___Late_blight	1.00	1.00	1.00	116
Corn_(maize)___healthy	0.94	1.00	0.97	118
Squash___Powdery_mildew	1.00	0.99	1.00	139
Tomato___Early_blight	0.99	1.00	1.00	108
Grape___healthy	0.98	0.98	0.98	43
Cherry_(including_sour)___Powdery_mildew	1.00	1.00	1.00	551
Pepper,_bell___healthy	0.99	0.98	0.99	230
Peach___healthy	0.97	0.97	0.97	36
Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.94	0.99	0.97	100
Apple___healthy	0.99	0.99	0.99	148
Potato___Late_blight	0.98	1.00	0.99	100
Corn_(maize)___Northern_Leaf_Blight	1.00	0.95	0.97	100
Pepper,_bell___Bacterial_spot	0.88	0.47	0.61	15
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	0.97	0.97	0.97	37
Raspberry___healthy	0.98	0.99	0.99	509
Apple___Cedar_apple_rust	1.00	0.99	0.99	184

```
                    Corn_(maize)___Common_rust_       0.99      1.00
1.00       111
                              Soybean___healthy       0.96      0.98
0.97        46
                        Tomato___Bacterial_spot       0.98      0.97
0.97       213
                          Potato___Early_blight       0.96      0.94
0.95       100
                   Grape___Esca_(Black_Measles)       0.99      0.93
0.96       191
                            Tomato___Target_Spot       0.97      0.99
0.98        95
                              Apple___Apple_scab       0.99      0.88
0.93       177
                               Apple___Black_rot       0.93      0.97
0.95       168
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot       0.89      0.83
0.86       141
     Tomato___Spider_mites Two-spotted_spider_mite       1.00      1.00
1.00       536
          Orange___Haunglongbing_(Citrus_greening)       1.00      1.00
1.00        38
                             Blueberry___healthy       0.93      1.00
0.96       159

                                        accuracy
0.98      5438
                                       macro avg       0.97      0.96
0.96      5438
                                    weighted avg       0.98      0.98
0.98      5438
```

[ ]:
```python
results = pretrained_model.evaluate(test_dataset, verbose=0)

print("    Test Loss: {:.5f}".format(results[0]))
print("Test Accuracy: {:.2f}%".format(results[1] * 100))
```

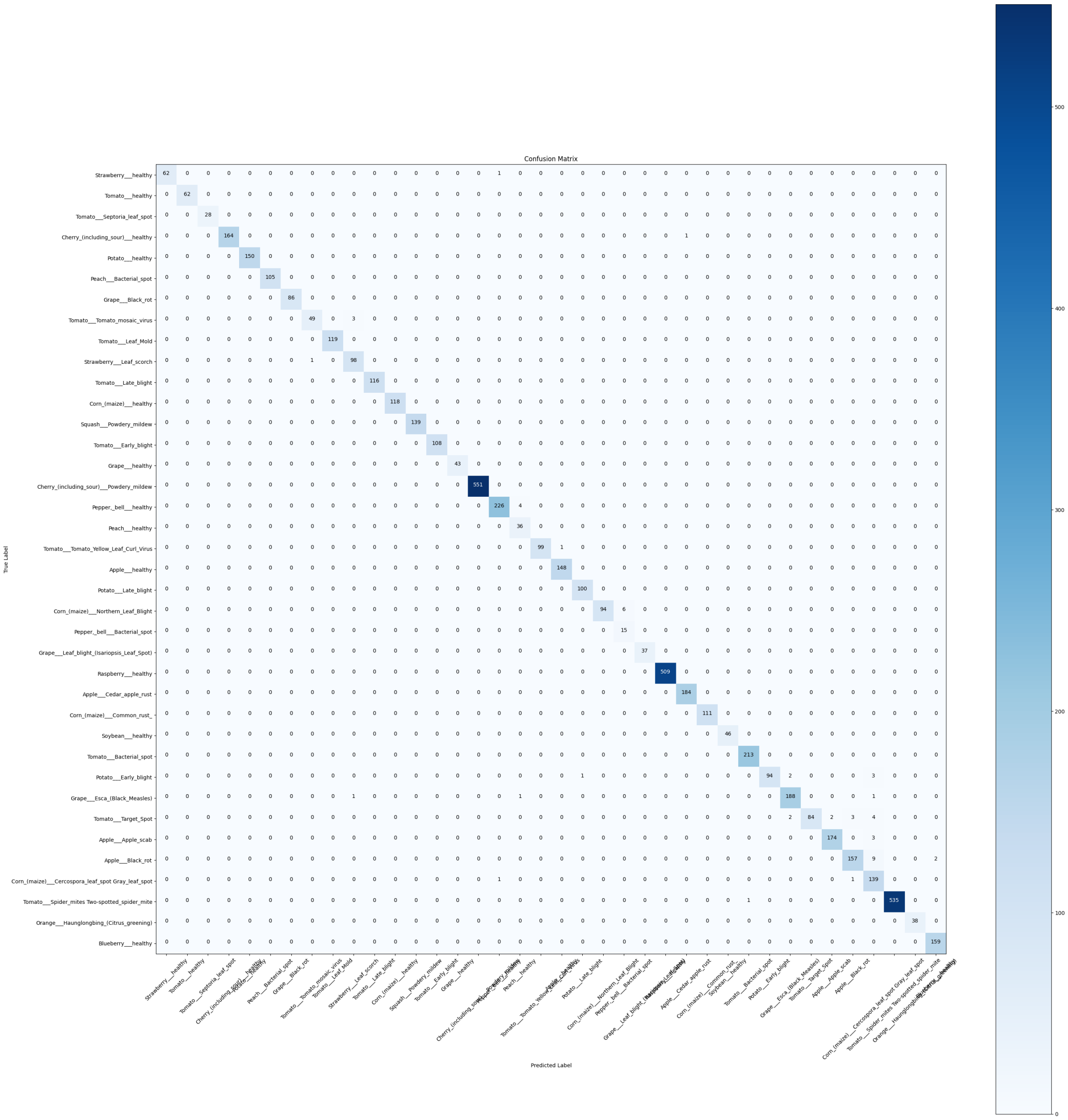
```
2023-09-12 17:58:00.337488: W tensorflow/tsl/framework/bfc_allocator.cc:296]
Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.27GiB with
freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.
2023-09-12 17:58:02.676602: W tensorflow/tsl/framework/bfc_allocator.cc:296]
Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.27GiB with
freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.
2023-09-12 17:58:02.676664: W tensorflow/tsl/framework/bfc_allocator.cc:296]
Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.27GiB with
```

```
freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.
2023-09-12 17:58:42.481601: W tensorflow/tsl/framework/bfc_allocator.cc:296]
Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.27GiB with
freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.
2023-09-12 17:58:44.422729: W tensorflow/tsl/framework/bfc_allocator.cc:296]
Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.12GiB with
freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.
2023-09-12 17:58:44.422790: W tensorflow/tsl/framework/bfc_allocator.cc:296]
Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.12GiB with
freed_by_count=0. The caller indicates that this is not a failure, but this may
mean that there could be performance gains if more memory were available.

    Test Loss: 0.03156
Test Accuracy: 99.01%
```

[ ]:

```
preds = pretrained_model.predict(test_dataset)
y_pred = np.argmax(preds, axis=1)

# Confusion matrix
cm = confusion_matrix(labels_list, y_pred)

plt.figure(figsize= (30, 30))
plt.imshow(cm, interpolation= 'nearest', cmap= plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()

tick_marks = np.arange(len(diseases))
plt.xticks(tick_marks, diseases, rotation= 45)
plt.yticks(tick_marks, diseases)


thresh = cm.max() / 2.
for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, cm[i, j], horizontalalignment= 'center', color= 'white' if␣
 ↪cm[i, j] > thresh else 'black')

plt.tight_layout()
plt.ylabel('True Label')
plt.xlabel('Predicted Label')

plt.show()
```

```
 49/170 [=======>…] - ETA: 25s170/170
[==============================] - 35s 208ms/step
```




```
[ ]: f1 = f1_score(labels_list, y_pred, average='macro')
     print("F1 Score:", f1)
     print(classification_report(labels_list, y_pred, target_names=diseases))
```

```
F1 Score: 0.9836812845075977
                                            precision    recall
f1-score   support

                     Strawberry___healthy       1.00      0.98
0.99        63
                         Tomato___healthy       1.00      1.00
1.00        62
```

Tomato___Septoria_leaf_spot	1.00	1.00	1.00	28
Cherry_(including_sour)___healthy	1.00	0.99	1.00	165
Potato___healthy	1.00	1.00	1.00	150
Peach___Bacterial_spot	1.00	1.00	1.00	105
Grape___Black_rot	1.00	1.00	1.00	86
Tomato___Tomato_mosaic_virus	0.98	0.94	0.96	52
Tomato___Leaf_Mold	1.00	1.00	1.00	119
Strawberry___Leaf_scorch	0.96	0.99	0.98	99
Tomato___Late_blight	1.00	1.00	1.00	116
Corn_(maize)___healthy	1.00	1.00	1.00	118
Squash___Powdery_mildew	1.00	1.00	1.00	139
Tomato___Early_blight	1.00	1.00	1.00	108
Grape___healthy	1.00	1.00	1.00	43
Cherry_(including_sour)___Powdery_mildew	1.00	1.00	1.00	551
Pepper,_bell___healthy	0.99	0.98	0.99	230
Peach___healthy	0.88	1.00	0.94	36
Tomato___Tomato_Yellow_Leaf_Curl_Virus	1.00	0.99	0.99	100
Apple___healthy	0.99	1.00	1.00	148
Potato___Late_blight	0.99	1.00	1.00	100
Corn_(maize)___Northern_Leaf_Blight	1.00	0.94	0.97	100
Pepper,_bell___Bacterial_spot	0.71	1.00	0.83	15
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	1.00	1.00	1.00	37
Raspberry___healthy	1.00	1.00	1.00	509
Apple___Cedar_apple_rust	0.99	1.00	1.00	184

Corn_(maize)___Common_rust_	1.00	1.00	1.00	111
Soybean___healthy	1.00	1.00	1.00	46
Tomato___Bacterial_spot	1.00	1.00	1.00	213
Potato___Early_blight	1.00	0.94	0.97	100
Grape___Esca_(Black_Measles)	0.98	0.98	0.98	191
Tomato___Target_Spot	1.00	0.88	0.94	95
Apple___Apple_scab	0.99	0.98	0.99	177
Apple___Black_rot	0.98	0.93	0.95	168
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	0.87	0.99	0.93	141
Tomato___Spider_mites Two-spotted_spider_mite	1.00	1.00	1.00	536
Orange___Haunglongbing_(Citrus_greening)	1.00	1.00	1.00	38
Blueberry___healthy	0.99	1.00	0.99	159
accuracy			0.99	5438
macro avg	0.98	0.99	0.98	5438
weighted avg	0.99	0.99	0.99	5438