Setting up Environment

In [ ]:
```
!nvidia-smi
```

```
Sun Jun  9 06:07:53 2024
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.05             Driver Version: 535.104.05   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Tesla T4                       Off | 00000000:00:04.0 Off |                    0 |
| N/A   40C    P8              11W /  70W |      0MiB / 15360MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
|  No running processes found                                                           |
+---------------------------------------------------------------------------------------+
```

In [ ]:
```
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession

config = ConfigProto()
config.gpu_options.per_process_gpu_memory_fraction = 0.5
config.gpu_options.allow_growth = True
session = InteractiveSession(config=config)
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Importing Required Libraries

In [11]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report

import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing import image_dataset_from_directory,image
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Model
```

In [ ]:
```

```

Preprocessing Our Dataset

In [ ]:
```

Image_size = (128, 128)
Batch_size = 32
EPOCHS = 50

train_ds = image_dataset_from_directory(
    "/content/drive/MyDrive/dataset(75-25)/train",
    labels='inferred',
    label_mode='int',
    Image_size=Image_size,
    batch_size=Batch_size
)

test_ds = image_dataset_from_directory(
    "/content/drive/MyDrive/dataset(75-25)/test",
    labels='inferred',
    label_mode='int',
    Image_size=Image_size,
```

```
    batch_size=Batch_size
)

norm_layer = layers.Rescaling(1./255)
train_dataset = train_ds.map(lambda x, y: (norm_layer(x), y))
test_dataset = test_ds.map(lambda x, y: (norm_layer(x), y))
```

```
Found 4500 files belonging to 3 classes.
Found 1500 files belonging to 3 classes.
```

In [ ]:

Defining our Resnet model With weights trained on imagenet

In [ ]:

```
base_model = ResNet50(include_top=False, weights='imagenet', input_shape=(*Image_size, 3))
base_model.trainable = False

x = layers.GlobalAveragePooling2D()(base_model.output)
x = layers.Dense(512, activation='relu')(x)
output = layers.Dense(3, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=output)

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

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 [==============================] - 1s 0us/step
```

In [ ]:

In [ ]:

```
history = model.fit(
    train_ds,
    epochs=EPOCHS,
    validation_data=test_ds,
    verbose=1
)


test_loss, test_acc = model.evaluate(test_ds)
print(f"Test accuracy: {test_acc}")
```

```
Epoch 1/50
141/141 [==============================] - 22s 107ms/step - loss: 0.6690 - accuracy: 0.6391 - val_loss: 0.5332 - val_accuracy: 0.6987
Epoch 2/50
141/141 [==============================] - 10s 72ms/step - loss: 0.5070 - accuracy: 0.6996 - val_loss: 0.4950 - val_accuracy: 0.7120
Epoch 3/50
141/141 [==============================] - 9s 62ms/step - loss: 0.4763 - accuracy: 0.7089 - val_loss: 0.5261 - val_accuracy: 0.6700
Epoch 4/50
141/141 [==============================] - 11s 72ms/step - loss: 0.4576 - accuracy: 0.7389 - val_loss: 0.4742 - val_accuracy: 0.7293
Epoch 5/50
141/141 [==============================] - 11s 73ms/step - loss: 0.4461 - accuracy: 0.7440 - val_loss: 0.4708 - val_accuracy: 0.7453
Epoch 6/50
141/141 [==============================] - 9s 58ms/step - loss: 0.4431 - accuracy: 0.7547 - val_loss: 0.4855 - val_accuracy: 0.6993
Epoch 7/50
141/141 [==============================] - 11s 76ms/step - loss: 0.4390 - accuracy: 0.7449 - val_loss: 0.4623 - val_accuracy: 0.7580
Epoch 8/50
141/141 [==============================] - 12s 82ms/step - loss: 0.4267 - accuracy: 0.7653 - val_loss: 0.4533 - val_accuracy: 0.7660
Epoch 9/50
141/141 [==============================] - 8s 58ms/step - loss: 0.4163 - accuracy: 0.7682 - val_loss: 0.4526 - val_accuracy: 0.7713
Epoch 10/50
141/141 [==============================] - 11s 78ms/step - loss: 0.4163 - accuracy: 0.7733 - val_loss: 0.4678 - val_accuracy: 0.7667
Epoch 11/50
141/141 [==============================] - 12s 85ms/step - loss: 0.4023 - accuracy: 0.7896 - val_loss: 0.5068 - val_accuracy: 0.7247
Epoch 12/50
141/141 [==============================] - 10s 67ms/step - loss: 0.4125 - accuracy: 0.7733 - val_loss: 0.4798 - val_accuracy: 0.7327
Epoch 13/50
141/141 [==============================] - 11s 74ms/step - loss: 0.3952 - accuracy: 0.7920 - val_loss: 0.4482 - val_accuracy: 0.7773
Epoch 14/50
141/141 [==============================] - 9s 64ms/step - loss: 0.3889 - accuracy: 0.7951 - val_loss: 0.4552 - val_accuracy: 0.7840
Epoch 15/50
141/141 [==============================] - 10s 66ms/step - loss: 0.3851 - accuracy: 0.7936 - val_loss: 0.4488 - val_accuracy: 0.7860
Epoch 16/50
141/141 [==============================] - 11s 77ms/step - loss: 0.3999 - accuracy: 0.7860 - val_loss: 0.4460 - val_accuracy: 0.7853
Epoch 17/50
141/141 [==============================] - 8s 58ms/step - loss: 0.3909 - accuracy: 0.7964 - val_loss: 0.4406 - val_accuracy: 0.7927
Epoch 18/50
141/141 [==============================] - 10s 72ms/step - loss: 0.3732 - accuracy: 0.8073 - val_loss: 0.4668 - val_accuracy: 0.7727
Epoch 19/50
141/141 [==============================] - 9s 63ms/step - loss: 0.3921 - accuracy: 0.7849 - val_loss: 0.5209 - val_accuracy: 0.7113
Epoch 20/50
```

```
141/141 [==============================] - 10s 70ms/step - loss: 0.3767 - accuracy: 0.8027 - val_loss: 0.5085 - val_accuracy: 0.7393
Epoch 21/50
141/141 [==============================] - 12s 83ms/step - loss: 0.3565 - accuracy: 0.8258 - val_loss: 0.4764 - val_accuracy: 0.7767
Epoch 22/50
141/141 [==============================] - 10s 70ms/step - loss: 0.3612 - accuracy: 0.8151 - val_loss: 0.4448 - val_accuracy: 0.8007
Epoch 23/50
141/141 [==============================] - 12s 82ms/step - loss: 0.3728 - accuracy: 0.8044 - val_loss: 0.4446 - val_accuracy: 0.7967
Epoch 24/50
141/141 [==============================] - 10s 69ms/step - loss: 0.3679 - accuracy: 0.8071 - val_loss: 0.4414 - val_accuracy: 0.8027
Epoch 25/50
141/141 [==============================] - 9s 64ms/step - loss: 0.3637 - accuracy: 0.8113 - val_loss: 0.4607 - val_accuracy: 0.7793
Epoch 26/50
141/141 [==============================] - 9s 64ms/step - loss: 0.3576 - accuracy: 0.8211 - val_loss: 0.5568 - val_accuracy: 0.7273
Epoch 27/50
141/141 [==============================] - 11s 72ms/step - loss: 0.3812 - accuracy: 0.8060 - val_loss: 0.4524 - val_accuracy: 0.7940
Epoch 28/50
141/141 [==============================] - 9s 64ms/step - loss: 0.3489 - accuracy: 0.8260 - val_loss: 0.4457 - val_accuracy: 0.8040
Epoch 29/50
141/141 [==============================] - 10s 64ms/step - loss: 0.3427 - accuracy: 0.8293 - val_loss: 0.4863 - val_accuracy: 0.7760
Epoch 30/50
141/141 [==============================] - 10s 71ms/step - loss: 0.3447 - accuracy: 0.8276 - val_loss: 0.4422 - val_accuracy: 0.8080
Epoch 31/50
141/141 [==============================] - 10s 66ms/step - loss: 0.3438 - accuracy: 0.8227 - val_loss: 0.4634 - val_accuracy: 0.7947
Epoch 32/50
141/141 [==============================] - 10s 70ms/step - loss: 0.3568 - accuracy: 0.8200 - val_loss: 0.5351 - val_accuracy: 0.7460
Epoch 33/50
141/141 [==============================] - 10s 70ms/step - loss: 0.3367 - accuracy: 0.8344 - val_loss: 0.4496 - val_accuracy: 0.8107
Epoch 34/50
141/141 [==============================] - 9s 63ms/step - loss: 0.3391 - accuracy: 0.8340 - val_loss: 0.4438 - val_accuracy: 0.8073
Epoch 35/50
141/141 [==============================] - 11s 72ms/step - loss: 0.3552 - accuracy: 0.8213 - val_loss: 0.4383 - val_accuracy: 0.8100
Epoch 36/50
141/141 [==============================] - 10s 67ms/step - loss: 0.3397 - accuracy: 0.8322 - val_loss: 0.4419 - val_accuracy: 0.8100
Epoch 37/50
141/141 [==============================] - 8s 58ms/step - loss: 0.3372 - accuracy: 0.8344 - val_loss: 0.4441 - val_accuracy: 0.8120
Epoch 38/50
141/141 [==============================] - 11s 78ms/step - loss: 0.3357 - accuracy: 0.8302 - val_loss: 0.4830 - val_accuracy: 0.7813
Epoch 39/50
141/141 [==============================] - 12s 84ms/step - loss: 0.3507 - accuracy: 0.8238 - val_loss: 0.4419 - val_accuracy: 0.8147
Epoch 40/50
141/141 [==============================] - 9s 62ms/step - loss: 0.3413 - accuracy: 0.8322 - val_loss: 0.4593 - val_accuracy: 0.8027
Epoch 41/50
141/141 [==============================] - 10s 67ms/step - loss: 0.3574 - accuracy: 0.8196 - val_loss: 0.5422 - val_accuracy: 0.7320
Epoch 42/50
141/141 [==============================] - 10s 72ms/step - loss: 0.3252 - accuracy: 0.8427 - val_loss: 0.4606 - val_accuracy: 0.7987
Epoch 43/50
141/141 [==============================] - 9s 60ms/step - loss: 0.3371 - accuracy: 0.8358 - val_loss: 0.4473 - val_accuracy: 0.8067
Epoch 44/50
141/141 [==============================] - 9s 63ms/step - loss: 0.3411 - accuracy: 0.8287 - val_loss: 0.4446 - val_accuracy: 0.8060
Epoch 45/50
141/141 [==============================] - 11s 75ms/step - loss: 0.3255 - accuracy: 0.8398 - val_loss: 0.6454 - val_accuracy: 0.7027
Epoch 46/50
141/141 [==============================] - 12s 86ms/step - loss: 0.3368 - accuracy: 0.8371 - val_loss: 0.5030 - val_accuracy: 0.7747
Epoch 47/50
141/141 [==============================] - 11s 75ms/step - loss: 0.3308 - accuracy: 0.8424 - val_loss: 0.4438 - val_accuracy: 0.8107
Epoch 48/50
141/141 [==============================] - 9s 64ms/step - loss: 0.3332 - accuracy: 0.8382 - val_loss: 0.4564 - val_accuracy: 0.8027
Epoch 49/50
141/141 [==============================] - 9s 61ms/step - loss: 0.3266 - accuracy: 0.8449 - val_loss: 0.4463 - val_accuracy: 0.8133
Epoch 50/50
141/141 [==============================] - 11s 73ms/step - loss: 0.3222 - accuracy: 0.8440 - val_loss: 0.4649 - val_accuracy: 0.8000
47/47 [==============================] - 2s 38ms/step - loss: 0.4649 - accuracy: 0.8000
Test accuracy: 0.800000011920929
```

In [ ]:

Finetuning the model to imporve Accuracy

In [ ]:

```
init_epochs = 50
add_epochs = 50
total_epochs = init_epochs + add_epochs


for layer in base_model.layers[-20:]:
    layer.trainable = True


model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])


history_finetuning = model.fit(
    train_ds,
    initial_epoch=init_epochs,
    epochs=total_epochs,
    validation_data=test_ds,
    verbose=1
)
```

```
test_loss, test_acc = model.evaluate(test_ds)
print(f"Test accuracy after fine-tuning: {test_acc}")
```

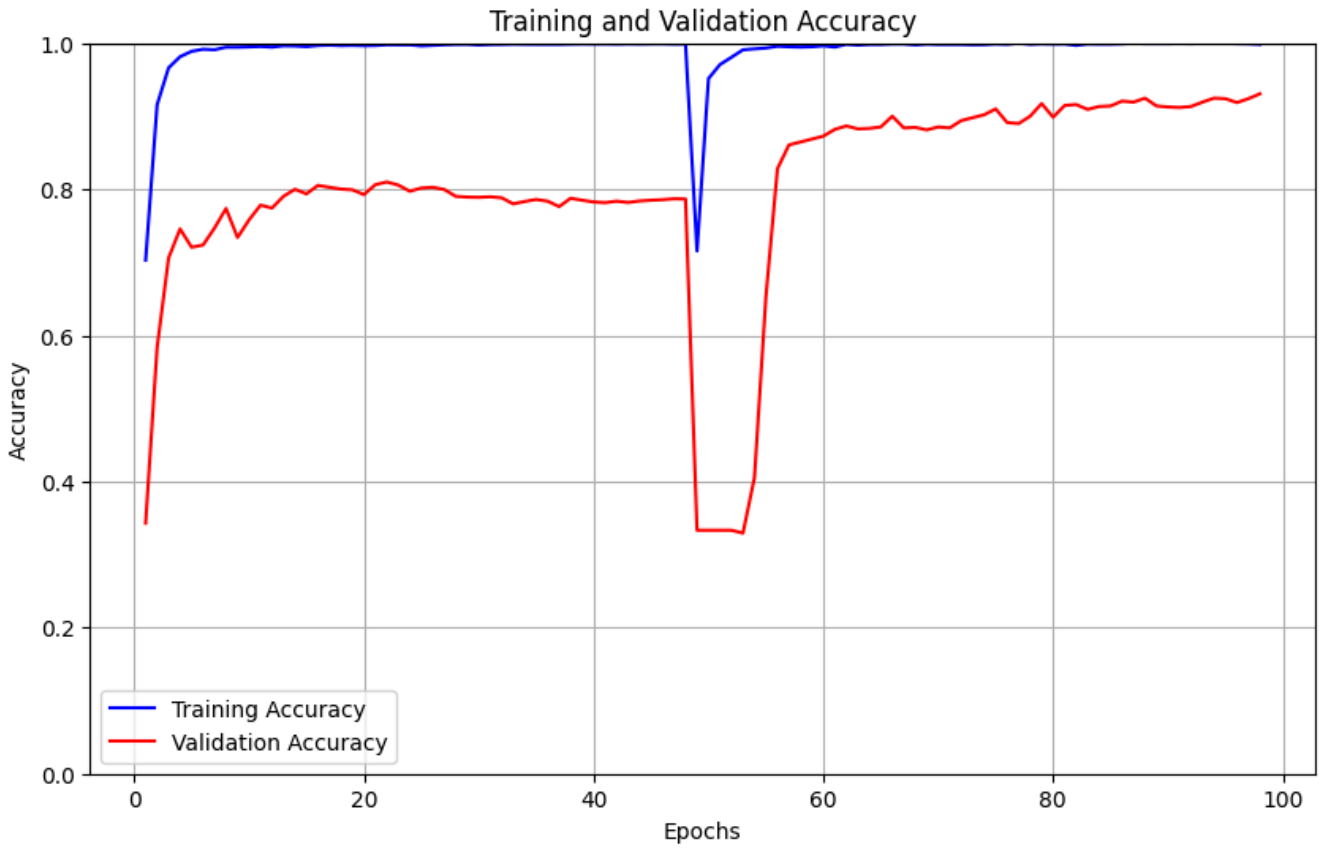
```
Epoch 51/100
141/141 [==============================] - 61s 156ms/step - loss: 20.0272 - accuracy: 0.7156 - val_loss: 864.6770 - val_accuracy: 0.3333
Epoch 52/100
141/141 [==============================] - 20s 143ms/step - loss: 0.9279 - accuracy: 0.9516 - val_loss: 3487.7163 - val_accuracy: 0.3333
Epoch 53/100
141/141 [==============================] - 23s 163ms/step - loss: 0.5947 - accuracy: 0.9707 - val_loss: 3391.1333 - val_accuracy: 0.3333
Epoch 54/100
141/141 [==============================] - 23s 163ms/step - loss: 0.2372 - accuracy: 0.9807 - val_loss: 808.9989 - val_accuracy: 0.3333
Epoch 55/100
141/141 [==============================] - 20s 140ms/step - loss: 0.0873 - accuracy: 0.9909 - val_loss: 127.7075 - val_accuracy: 0.3293
Epoch 56/100
141/141 [==============================] - 21s 149ms/step - loss: 0.0980 - accuracy: 0.9922 - val_loss: 62.1430 - val_accuracy: 0.4053
Epoch 57/100
141/141 [==============================] - 20s 143ms/step - loss: 0.0942 - accuracy: 0.9933 - val_loss: 16.9804 - val_accuracy: 0.6533
Epoch 58/100
141/141 [==============================] - 21s 145ms/step - loss: 0.0568 - accuracy: 0.9958 - val_loss: 4.3637 - val_accuracy: 0.8280
Epoch 59/100
141/141 [==============================] - 21s 144ms/step - loss: 0.0427 - accuracy: 0.9951 - val_loss: 3.9159 - val_accuracy: 0.8607
Epoch 60/100
141/141 [==============================] - 20s 143ms/step - loss: 0.0509 - accuracy: 0.9947 - val_loss: 4.2491 - val_accuracy: 0.8647
Epoch 61/100
141/141 [==============================] - 20s 140ms/step - loss: 0.0402 - accuracy: 0.9951 - val_loss: 4.1161 - val_accuracy: 0.8687
Epoch 62/100
141/141 [==============================] - 24s 165ms/step - loss: 0.0340 - accuracy: 0.9964 - val_loss: 4.1385 - val_accuracy: 0.8727
Epoch 63/100
141/141 [==============================] - 20s 141ms/step - loss: 0.0370 - accuracy: 0.9947 - val_loss: 3.7679 - val_accuracy: 0.8820
Epoch 64/100
141/141 [==============================] - 20s 142ms/step - loss: 0.0098 - accuracy: 0.9984 - val_loss: 3.6605 - val_accuracy: 0.8867
Epoch 65/100
141/141 [==============================] - 21s 144ms/step - loss: 0.0259 - accuracy: 0.9973 - val_loss: 3.7185 - val_accuracy: 0.8827
Epoch 66/100
141/141 [==============================] - 21s 144ms/step - loss: 0.0075 - accuracy: 0.9982 - val_loss: 3.6148 - val_accuracy: 0.8833
Epoch 67/100
141/141 [==============================] - 20s 142ms/step - loss: 0.0105 - accuracy: 0.9982 - val_loss: 3.5578 - val_accuracy: 0.8853
Epoch 68/100
141/141 [==============================] - 21s 145ms/step - loss: 0.0136 - accuracy: 0.9989 - val_loss: 3.3867 - val_accuracy: 0.9000
Epoch 69/100
141/141 [==============================] - 23s 162ms/step - loss: 0.0150 - accuracy: 0.9989 - val_loss: 3.3957 - val_accuracy: 0.8840
Epoch 70/100
141/141 [==============================] - 20s 139ms/step - loss: 0.0187 - accuracy: 0.9976 - val_loss: 3.3135 - val_accuracy: 0.8847
Epoch 71/100
141/141 [==============================] - 23s 164ms/step - loss: 0.0161 - accuracy: 0.9987 - val_loss: 3.3780 - val_accuracy: 0.8813
Epoch 72/100
141/141 [==============================] - 20s 143ms/step - loss: 0.0122 - accuracy: 0.9982 - val_loss: 3.3939 - val_accuracy: 0.8853
Epoch 73/100
141/141 [==============================] - 21s 148ms/step - loss: 0.0153 - accuracy: 0.9982 - val_loss: 3.6418 - val_accuracy: 0.8840
Epoch 74/100
141/141 [==============================] - 21s 143ms/step - loss: 0.0106 - accuracy: 0.9982 - val_loss: 3.5859 - val_accuracy: 0.8940
Epoch 75/100
141/141 [==============================] - 21s 145ms/step - loss: 0.0400 - accuracy: 0.9978 - val_loss: 3.0830 - val_accuracy: 0.8980
Epoch 76/100
141/141 [==============================] - 20s 140ms/step - loss: 0.0187 - accuracy: 0.9980 - val_loss: 3.1308 - val_accuracy: 0.9020
Epoch 77/100
141/141 [==============================] - 20s 140ms/step - loss: 0.0193 - accuracy: 0.9989 - val_loss: 2.4269 - val_accuracy: 0.9100
Epoch 78/100
141/141 [==============================] - 20s 138ms/step - loss: 0.0231 - accuracy: 0.9984 - val_loss: 2.8459 - val_accuracy: 0.8913
Epoch 79/100
141/141 [==============================] - 21s 145ms/step - loss: 9.7180e-06 - accuracy: 1.0000 - val_loss: 2.8459 - val_accuracy: 0.8900
Epoch 80/100
141/141 [==============================] - 20s 143ms/step - loss: 0.0149 - accuracy: 0.9984 - val_loss: 2.6405 - val_accuracy: 0.9000
Epoch 81/100
141/141 [==============================] - 20s 139ms/step - loss: 0.0061 - accuracy: 0.9991 - val_loss: 2.5677 - val_accuracy: 0.9173
Epoch 82/100
141/141 [==============================] - 20s 143ms/step - loss: 0.0283 - accuracy: 0.9987 - val_loss: 2.4600 - val_accuracy: 0.8987
Epoch 83/100
141/141 [==============================] - 21s 145ms/step - loss: 0.0059 - accuracy: 0.9989 - val_loss: 2.3962 - val_accuracy: 0.9147
Epoch 84/100
141/141 [==============================] - 21s 144ms/step - loss: 0.0330 - accuracy: 0.9969 - val_loss: 2.2905 - val_accuracy: 0.9160
Epoch 85/100
141/141 [==============================] - 21s 145ms/step - loss: 0.0041 - accuracy: 0.9989 - val_loss: 2.4623 - val_accuracy: 0.9093
Epoch 86/100
141/141 [==============================] - 23s 163ms/step - loss: 0.0134 - accuracy: 0.9987 - val_loss: 2.3083 - val_accuracy: 0.9133
Epoch 87/100
141/141 [==============================] - 21s 144ms/step - loss: 0.0078 - accuracy: 0.9987 - val_loss: 2.7671 - val_accuracy: 0.9140
Epoch 88/100
141/141 [==============================] - 20s 138ms/step - loss: 0.0032 - accuracy: 0.9991 - val_loss: 2.3161 - val_accuracy: 0.9207
Epoch 89/100
141/141 [==============================] - 21s 147ms/step - loss: 0.0014 - accuracy: 0.9998 - val_loss: 2.1287 - val_accuracy: 0.9193
Epoch 90/100
141/141 [==============================] - 21s 144ms/step - loss: 0.0127 - accuracy: 0.9993 - val_loss: 1.8356 - val_accuracy: 0.9247
Epoch 91/100
141/141 [==============================] - 20s 143ms/step - loss: 0.0031 - accuracy: 0.9993 - val_loss: 2.0769 - val_accuracy: 0.9140
Epoch 92/100
141/141 [==============================] - 21s 143ms/step - loss: 0.0042 - accuracy: 0.9993 - val_loss: 2.3272 - val_accuracy: 0.9127
Epoch 93/100
141/141 [==============================] - 21s 146ms/step - loss: 0.0020 - accuracy: 0.9996 - val_loss: 2.5582 - val_accuracy: 0.9120
Epoch 94/100
141/141 [==============================] - 21s 144ms/step - loss: 0.0079 - accuracy: 0.9993 - val_loss: 2.3333 - val_accuracy: 0.9133
Epoch 95/100
141/141 [==============================] - 20s 143ms/step - loss: 0.0066 - accuracy: 0.9998 - val_loss: 2.1890 - val_accuracy: 0.9193
Epoch 96/100
141/141 [==============================] - 23s 163ms/step - loss: 0.0028 - accuracy: 0.9996 - val_loss: 1.9015 - val_accuracy: 0.9247
Epoch 97/100
141/141 [==============================] - 21s 150ms/step - loss: 3.0708e-04 - accuracy: 0.9998 - val_loss: 1.9257 - val_accuracy: 0.9240
Epoch 98/100
141/141 [==============================] - 21s 142ms/step - loss: 0.0083 - accuracy: 0.9993 - val_loss: 1.8280 - val_accuracy: 0.9187
Epoch 99/100
141/141 [==============================] - 21s 144ms/step - loss: 0.0096 - accuracy: 0.9991 - val_loss: 1.9458 - val_accuracy: 0.9240
Epoch 100/100
141/141 [==============================] - 21s 145ms/step - loss: 0.0138 - accuracy: 0.9984 - val_loss: 1.6484 - val_accuracy: 0.9307
47/47 [==============================] - 2s 37ms/step - loss: 1.6484 - accuracy: 0.9307
Test accuracy after fine-tuning: 0.9306666851043701
```

In [ ]:
```
# Save the model
model.save('resnet_finetuned_model.h5')
```

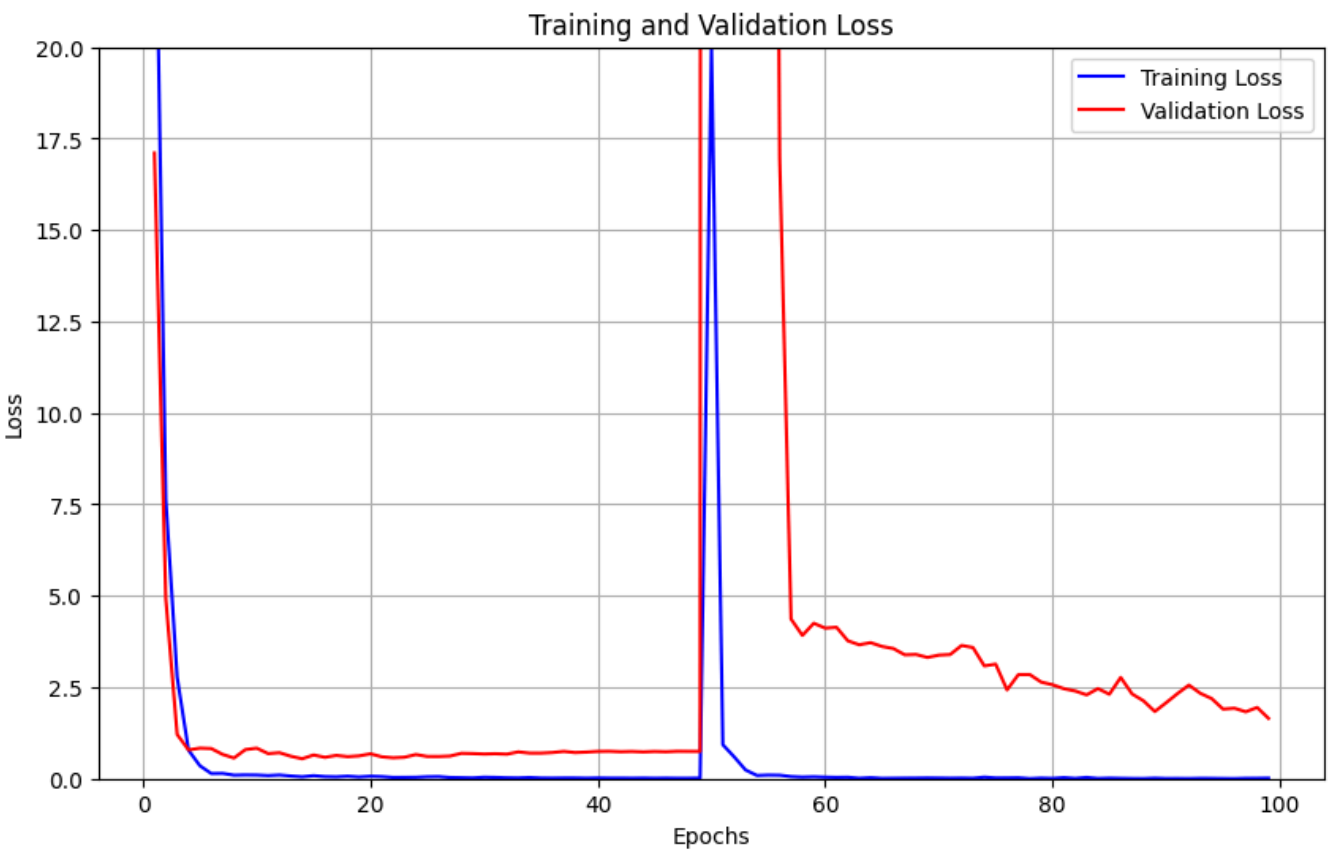
```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

Ploting Accuracy and Loss

In [ ]:
```
def Plot_acc(history, history_finetuning):
    plt.figure(figsize=(10, 6))
    plt.plot(history.history['accuracy'] + history_finetuning.history['accuracy'], label='Train Accuracy')
    plt.plot(history.history['val_accuracy'] + history_finetuning.history['val_accuracy'], label='Validation Ac
    plt.title('Model Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.grid(True)
    plt.show()


Plot_acc(history, history_finetuning)
```



In [ ]:
```
def plot_loss(history, history_finetuning):
    plt.figure(figsize=(10, 6))
    plt.plot(history.history['loss'] + history_finetuning.history['loss'], label='Train Loss')
    plt.plot(history.history['val_loss'] + history_finetuning.history['val_loss'], label='Validation Loss')
    plt.title('Model Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.grid(True)
    plt.show()


plot_loss(history, history_finetuning)
```

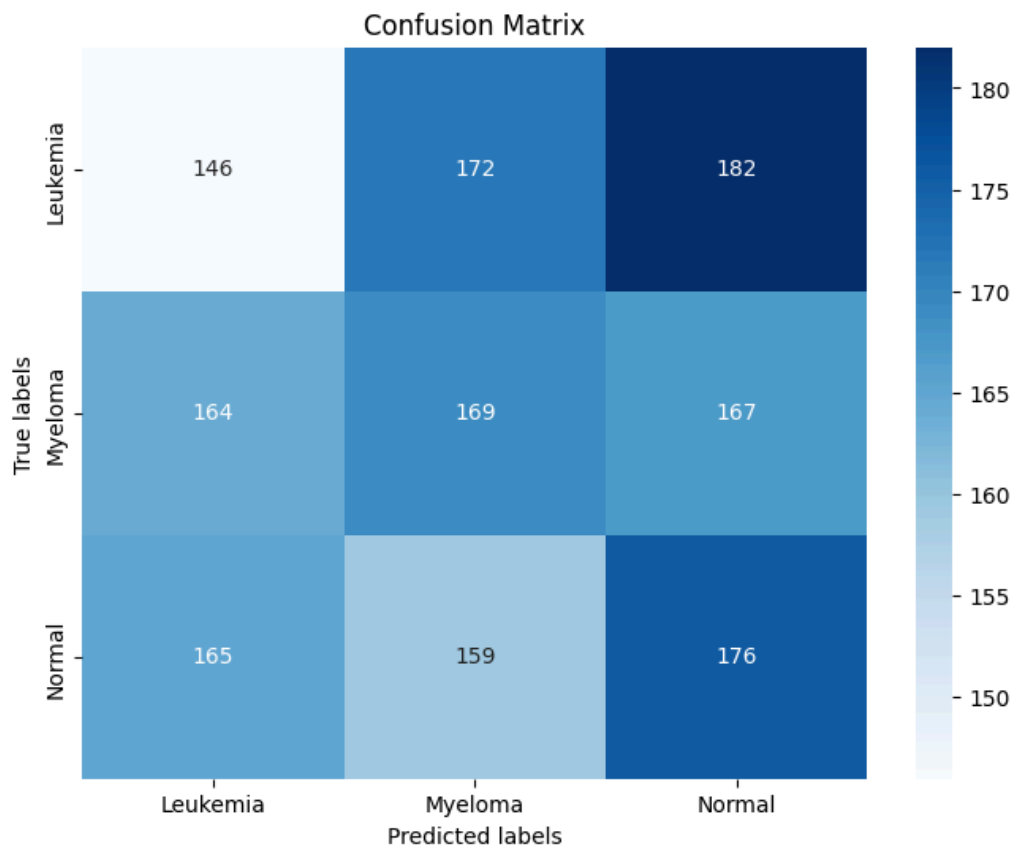


Confusion Matrix and Classification Report

In [10]:

```
model = tf.keras.models.load_model('resnet_finetuned_model.h5')

normalization_layer = layers.Rescaling(1./255)
test_ds = image_dataset_from_directory(
    "/content/drive/MyDrive/dataset(75-25)/test",
    labels='inferred',
    label_mode='int',
    Image_size=Image_size,
    batch_size=Batch_size
)
test_ds = test_ds.map(lambda x, y: (normalization_layer(x), y))

val_true_classes = np.concatenate([y for _, y in test_ds], axis=0)
val_pred = model.predict(test_ds)
val_pred_classes = np.argmax(val_pred, axis=-1)

classes = ['Leukemia', 'Myeloma', 'Normal']

cm = confusion_matrix(val_true_classes, val_pred_classes)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g', xticklabels=classes, yticklabels=classes)
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()

print(classification_report(val_true_classes, val_pred_classes))
```

```
Found 1500 files belonging to 3 classes.
47/47 [==============================] - 3s 38ms/step
```



	precision	recall	f1-score	support
0	0.31	0.29	0.30	500
1	0.34	0.34	0.34	500
2	0.34	0.35	0.34	500
accuracy			0.33	1500
macro avg	0.33	0.33	0.33	1500
weighted avg	0.33	0.33	0.33	1500

In [ ]:
```
from google.colab import files

files.download('resnet_finetuned_model.h5')
```

```
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
```

Testing Our Finetuned Resnet Model with test data (ie., Sample images)

In [ ]:
```
model = tf.keras.models.load_model('resnet_finetuned_model.h5')

classes = ['Leukemia', 'Myeloma', 'Normal']

def preprocess_image(img_path):
    img = image.load_img(img_path, target_size=Image_size)
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array = img_array / 255.0
    return img_array

def predict_image(model, img_path):
    img_array = preprocess_image(img_path)
    pred = model.predict(img_array)
    score = np.argmax(pred[0])
    return classes[score], pred[0]


img_path = '/content/_0_2680.jpeg'
pred_class, pred_scores = predict_image(model, img_path)
print(f"Predicted class: {pred_class}")
print(f"Prediction scores: {pred_scores}")
```

```
1/1 [==============================] - 1s 1s/step
Predicted class: Leukemia
Prediction scores: [1.000000e+00 0.000000e+00 4.346294e-20]
```

In [ ]:

In [ ]: