```
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in c:\users\jiyan\anaconda3\
lib\site-packages (2.16.1)
Requirement already satisfied: tensorflow-intel==2.16.1 in c:\users\
jiyan\anaconda3\lib\site-packages (from tensorflow) (2.16.1)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\jiyan\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\jiyan\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in c:\users\jiyan\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
c:\users\jiyan\anaconda3\lib\site-packages (from tensorflow-
intel==2.16.1->tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\jiyan\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in c:\users\jiyan\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\jiyan\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes~=0.3.1 in c:\users\jiyan\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (0.3.2)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\jiyan\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (3.3.0)
Requirement already satisfied: packaging in c:\users\jiyan\anaconda3\
lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (23.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!
=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in c:\users\jiyan\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (4.25.3)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\jiyan\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (2.31.0)
Requirement already satisfied: setuptools in c:\users\jiyan\anaconda3\
lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (68.0.0)
Requirement already satisfied: six>=1.12.0 in c:\users\jiyan\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\jiyan\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (2.4.0)
```

```
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\
jiyan\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (4.7.1)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\jiyan\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (1.14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\jiyan\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (1.64.1)
Requirement already satisfied: tensorboard<2.17,>=2.16 in c:\users\
jiyan\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (2.16.2)
Requirement already satisfied: keras>=3.0.0 in c:\users\jiyan\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (3.3.3)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
c:\users\jiyan\anaconda3\lib\site-packages (from tensorflow-
intel==2.16.1->tensorflow) (0.31.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in c:\users\jiyan\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (1.24.3)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\jiyan\
anaconda3\lib\site-packages (from astunparse>=1.6.0->tensorflow-
intel==2.16.1->tensorflow) (0.38.4)
Requirement already satisfied: rich in c:\users\jiyan\anaconda3\lib\
site-packages (from keras>=3.0.0->tensorflow-intel==2.16.1-
>tensorflow) (13.7.1)
Requirement already satisfied: namex in c:\users\jiyan\anaconda3\lib\
site-packages (from keras>=3.0.0->tensorflow-intel==2.16.1-
>tensorflow) (0.0.8)
Requirement already satisfied: optree in c:\users\jiyan\anaconda3\lib\
site-packages (from keras>=3.0.0->tensorflow-intel==2.16.1-
>tensorflow) (0.11.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\
jiyan\anaconda3\lib\site-packages (from requests<3,>=2.21.0-
>tensorflow-intel==2.16.1->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\jiyan\
anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorflow-
intel==2.16.1->tensorflow) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\jiyan\
anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorflow-
intel==2.16.1->tensorflow) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\jiyan\
anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorflow-
intel==2.16.1->tensorflow) (2023.7.22)
Requirement already satisfied: markdown>=2.6.8 in c:\users\jiyan\
anaconda3\lib\site-packages (from tensorboard<2.17,>=2.16->tensorflow-
intel==2.16.1->tensorflow) (3.4.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0
```

```
in c:\users\jiyan\anaconda3\lib\site-packages (from
tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\jiyan\
anaconda3\lib\site-packages (from tensorboard<2.17,>=2.16->tensorflow-
intel==2.16.1->tensorflow) (2.2.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\jiyan\
anaconda3\lib\site-packages (from werkzeug>=1.0.1-
>tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow)
(2.1.1)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\
jiyan\anaconda3\lib\site-packages (from rich->keras>=3.0.0-
>tensorflow-intel==2.16.1->tensorflow) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\
jiyan\anaconda3\lib\site-packages (from rich->keras>=3.0.0-
>tensorflow-intel==2.16.1->tensorflow) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in c:\users\jiyan\anaconda3\
lib\site-packages (from markdown-it-py>=2.2.0->rich->keras>=3.0.0-
>tensorflow-intel==2.16.1->tensorflow) (0.1.0)
```

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pathlib
import tensorflow as tf
import PIL
import os
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential

import zipfile

# Define the path to the zip file
zip_file_path = "CNN_assignment.zip"

# Define the directory where you want to extract the files
extracted_dir_path = "extracted_files"

# Function to extract zip files
def extract_zip(zip_file_path, extracted_dir_path):
    with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
        zip_ref.extractall(extracted_dir_path)

# Extract the zip file
extract_zip(zip_file_path, extracted_dir_path)


from pathlib import Path

# Define the paths for the training and testing directories
```

```
datate_dir_train = Path("extracted_files/Skin cancer ISIC The International Skin Imaging Collaboration/Test")
datate_dir_test = Path("extracted_files/Skin cancer ISIC The International Skin Imaging Collaboration/Train")


#list directory in train folder
dire_train = os.listdir(datate_dir_train)
dire_train.sort()
dire_train
```

```
['actinic keratosis',
 'basal cell carcinoma',
 'dermatofibroma',
 'melanoma',
 'nevus',
 'pigmented benign keratosis',
 'seborrheic keratosis',
 'squamous cell carcinoma',
 'vascular lesion']
```

```
#list dir in test folder
dire_test = os.listdir(datate_dir_test)
dire_test.sort()
dire_test
```

```
['actinic keratosis',
 'basal cell carcinoma',
 'dermatofibroma',
 'melanoma',
 'nevus',
 'pigmented benign keratosis',
 'seborrheic keratosis',
 'squamous cell carcinoma',
 'vascular lesion']
```

```
totl_train_data = len(list(datate_dir_train.glob("*/*.jpg")))
totl_train_data
```

```
118
```

```
totl_test_data = len(list(datate_dir_test.glob("*/*.jpg")))
totl_test_data
```

```
2239
```

```
# Initialize an empty DataFrame
data_detail_pd = pd.DataFrame()
```

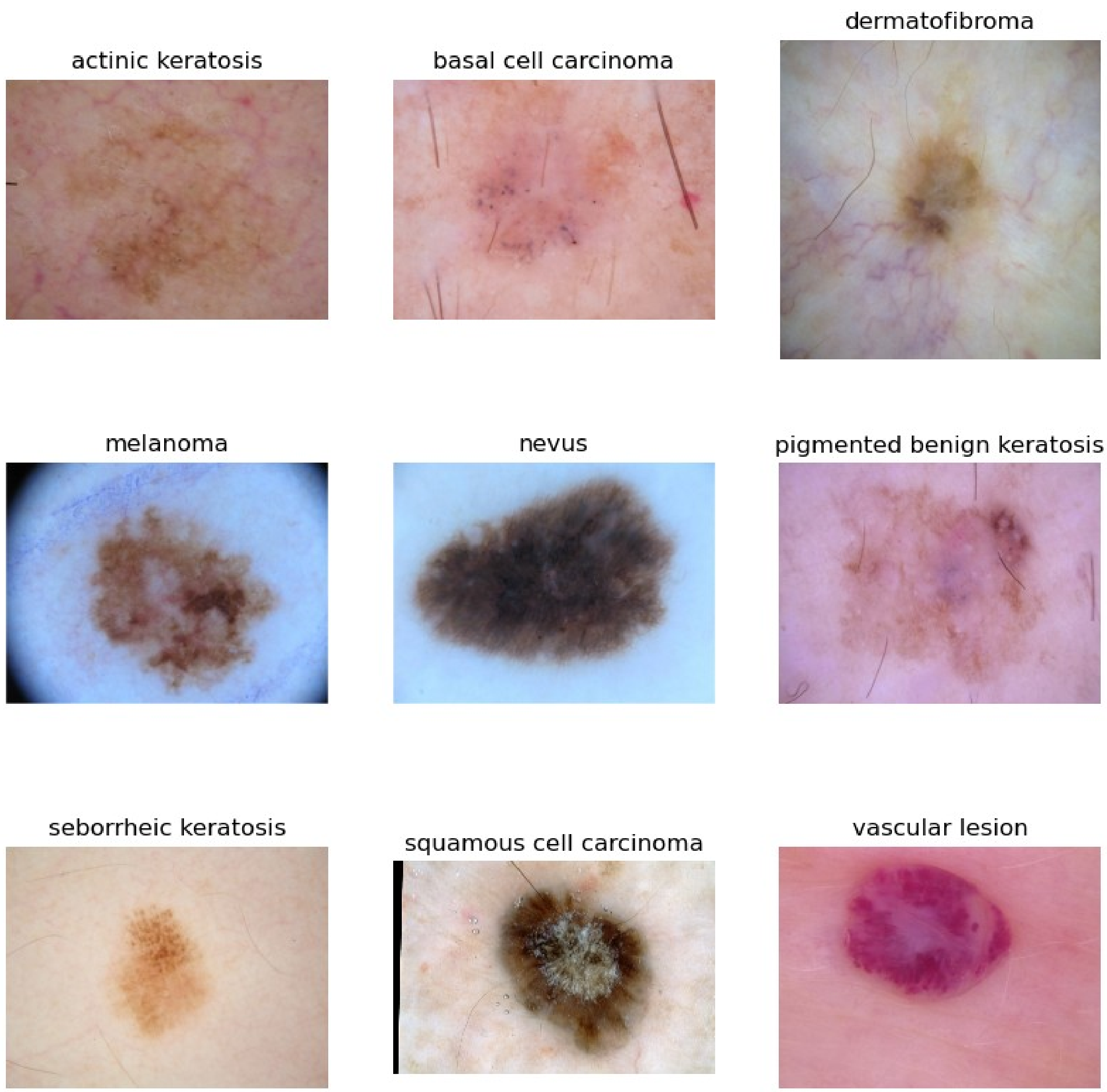
```
for dir_name in dire_train:
    total_image_in_folder = len(list(datate_dir_train.glob(dir_name +
"/*.jpg")))
    df = {"Dir_Name": dir_name, "Total Image(Train)":
total_image_in_folder,
          "Total Percentage(Train)": round((total_image_in_folder /
totl_train_data) * 100, 2)}
    # Append the current row as a DataFrame to the main DataFrame
    data_detail_pd = pd.concat([data_detail_pd, pd.DataFrame(df,
index=[0])], ignore_index=True)

data_detail_pd = data_detail_pd.set_index("Dir_Name")


data_detail_pd = pd.DataFrame()

for dir_name in dire_train:
    total_image_in_folder = len(list(datate_dir_train.glob(dir_name +
"/*.jpg")))
    df = {"Dir_Name": dir_name, "Total Image(Train)":
total_image_in_folder,
          "Total Percentage(Train)": round((total_image_in_folder /
totl_train_data) * 100, 2)}
    # Append the current row as a DataFrame to the main DataFrame
    data_detail_pd = pd.concat([data_detail_pd, pd.DataFrame(df,
index=[0])], ignore_index=True)

data_detail_pd = data_detail_pd.set_index("Dir_Name")

for dir_name in dire_test:
   total_image_in_folder =
len(list(datate_dir_test.glob(dir_name+"/*.jpg")))
   data_detail_pd.loc[dir_name,"Total Image(Test)"]  =
total_image_in_folder
   data_detail_pd.loc[dir_name,"Total Percentage(Test)"]  =
round((total_image_in_folder/totl_train_data)*100,2)
display(data_detail_pd.sort_values(by="Total
Percentage(Train)",ascending=False))
```

```
                            Total Image(Train)  Total
Percentage(Train)  \
Dir_Name

actinic keratosis                           16
13.56
basal cell carcinoma                        16
13.56
dermatofibroma                              16
13.56
```

```
melanoma                                          16
13.56
nevus                                             16
13.56
pigmented benign keratosis                        16
13.56
squamous cell carcinoma                           16
13.56
seborrheic keratosis                               3
2.54
vascular lesion                                    3
2.54

                            Total Image(Test)  Total Percentage(Test)

Dir_Name

actinic keratosis                       114.0                   96.61

basal cell carcinoma                    376.0                  318.64

dermatofibroma                           95.0                   80.51

melanoma                                438.0                  371.19

nevus                                   357.0                  302.54

pigmented benign keratosis              462.0                  391.53

squamous cell carcinoma                 181.0                  153.39

seborrheic keratosis                     77.0                   65.25

vascular lesion                         139.0                  117.80
```

```
#Dataset Visualization
#get one image from each folder
import glob
import matplotlib.image as mpimg

file_path = []
class_name = []

#get one file path from each folder
for dir_name in dire_train:
  path = str(datate_dir_train) +"/"+ dir_name
  for file_name in glob.iglob(path+'/*.jpg', recursive=True):
    #print(file_name)
    file_path.append(file_name)
    class_name.append(dir_name)
```

```
    break


#display one image from each folder
plt.figure(figsize=(10,10))
for i in range(len(class_name)):
  ax = plt.subplot(3,3,i+1)
  img = mpimg.imread(file_path[i])
  plt.imshow(img)
  plt.axis("off")
  plt.title(class_name[i])
```



```
#data loader params
batch_size = 32
img_height = 180
img_width = 180


# load train dataset in batches of size 32, resize the image into
180*180 pixel
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    datate_dir_train,
    validation_split=0.2,
    subset = "training",
    seed = 123,
    image_size = (img_height,img_width),
    batch_size = batch_size
)
```

```
Found 118 files belonging to 9 classes.
Using 95 files for training.
```

```
# load validation dataset in batches of size 32, resize the image into
180*180 pixel
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    datate_dir_train,
    validation_split = 0.2,
    subset = "validation",
    seed = 123,
    image_size = (img_height,img_width),
    batch_size = batch_size

)
```

```
Found 118 files belonging to 9 classes.
Using 23 files for validation.
```

```
# its a multiclassifier so lets see its number of different labels /
classes

num_classes = len(val_ds.class_names)
num_classes
```

```
9
```

```
#class names
val_ds.class_names
```

```
['actinic keratosis',
 'basal cell carcinoma',
 'dermatofibroma',
 'melanoma',
 'nevus',
 'pigmented benign keratosis',
 'seborrheic keratosis',
 'squamous cell carcinoma',
 'vascular lesion']

# Configure data set for performance
#Dataset.cache() keeps the images in memory after they're loaded off
disk during the first epoch.
#Dataset.prefetch() overlaps data preprocessing and model execution
while training.

AUTOTUNE = tf.data.AUTOTUNE
train_ds =
train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)

#M1 Model


model = Sequential([

layers.Rescaling(1./255,input_shape=(img_height,img_width,3)),
         layers.Conv2D(16,3,padding='same',activation="relu"),
         layers.MaxPool2D((2,2),strides=2),
         layers.Conv2D(32,3,padding='same',activation="relu"),
         layers.MaxPool2D((2,2),strides=2),
         layers.Conv2D(64,3,padding='same',activation="relu"),
         layers.MaxPool2D((2,2),strides=2),
         layers.Flatten(),
         layers.Dense(128,activation="relu"),
         layers.Dense(num_classes)
])


C:\Users\jiyan\anaconda3\Lib\site-packages\keras\src\layers\
preprocessing\tf_data_layer.py:18: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)

model.summary()

Model: "sequential"
```

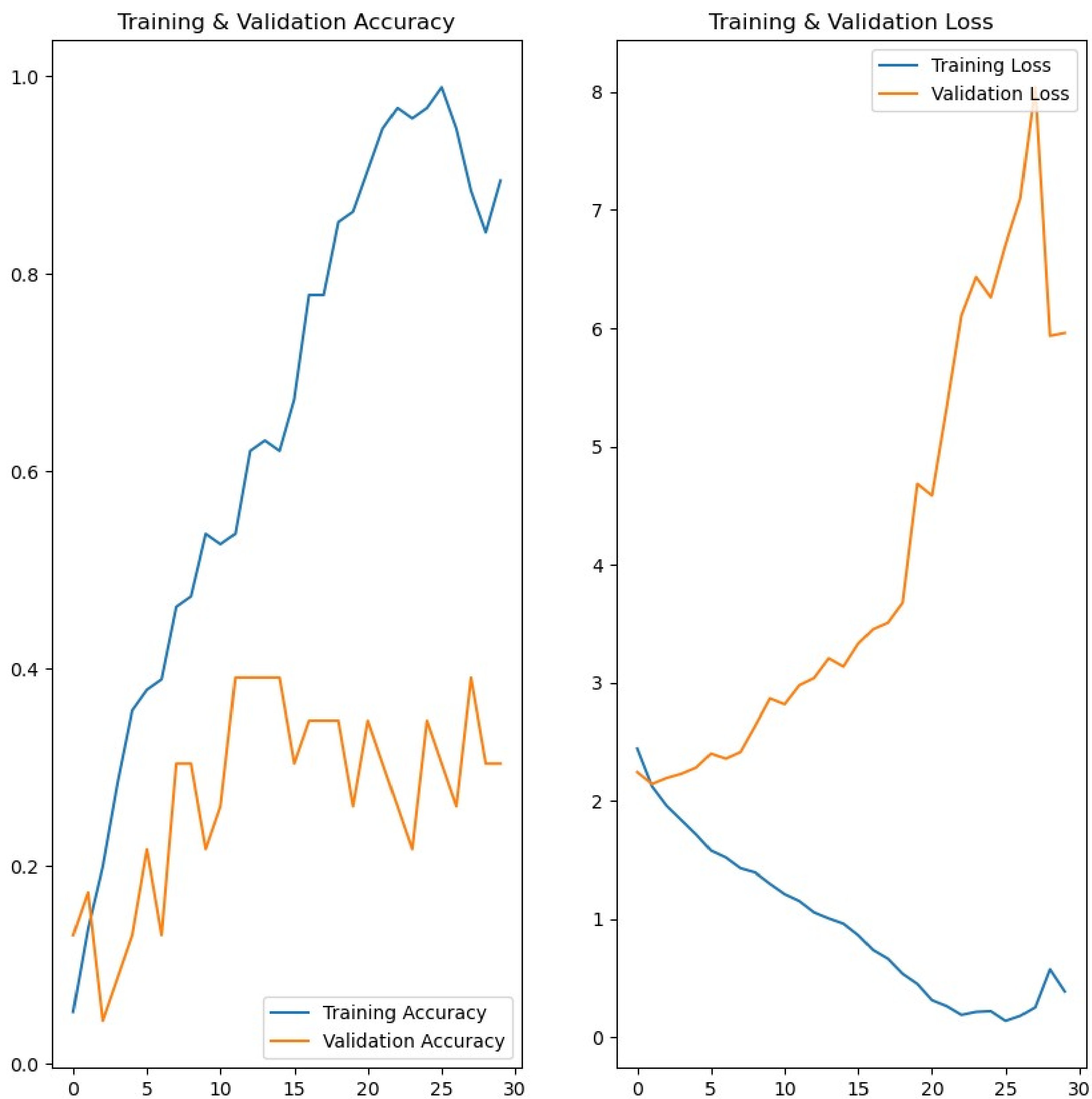
Layer (type)	Output Shape	Param #
rescaling (Rescaling)	(None, 180, 180, 3)	0
conv2d (Conv2D)	(None, 180, 180, 16)	448
max_pooling2d (MaxPooling2D)	(None, 90, 90, 16)	0
conv2d_1 (Conv2D)	(None, 90, 90, 32)	4,640
max_pooling2d_1 (MaxPooling2D)	(None, 45, 45, 32)	0
conv2d_2 (Conv2D)	(None, 45, 45, 64)	18,496
max_pooling2d_2 (MaxPooling2D)	(None, 22, 22, 64)	0
flatten (Flatten)	(None, 30976)	0
dense (Dense)	(None, 128)	3,965,056
dense_1 (Dense)	(None, 9)	1,161

```
 Total params: 3,989,801 (15.22 MB)

 Trainable params: 3,989,801 (15.22 MB)
```

```
 Non-trainable params: 0 (0.00 B)


#train the model : run the model on train & validation set
# Compile the model
model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

# Train the model
epochs = 30
history = model.fit(train_ds, validation_data=val_ds, epochs=epochs)

Epoch 1/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 3s 412ms/step - accuracy: 0.0382 - loss:
2.4265 - val_accuracy: 0.1304 - val_loss: 2.2430
Epoch 2/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 166ms/step - accuracy: 0.1443 - loss:
2.1661 - val_accuracy: 0.1739 - val_loss: 2.1428
Epoch 3/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 144ms/step - accuracy: 0.2041 - loss:
1.9822 - val_accuracy: 0.0435 - val_loss: 2.1938
Epoch 4/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 142ms/step - accuracy: 0.2710 - loss:
1.8341 - val_accuracy: 0.0870 - val_loss: 2.2291
Epoch 5/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 177ms/step - accuracy: 0.3522 - loss:
1.7088 - val_accuracy: 0.1304 - val_loss: 2.2819
Epoch 6/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 227ms/step - accuracy: 0.4019 - loss:
1.5643 - val_accuracy: 0.2174 - val_loss: 2.4001
Epoch 7/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 189ms/step - accuracy: 0.4307 - loss:
1.5282 - val_accuracy: 0.1304 - val_loss: 2.3578
Epoch 8/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 204ms/step - accuracy: 0.4581 - loss:
1.4414 - val_accuracy: 0.3043 - val_loss: 2.4138
Epoch 9/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 186ms/step - accuracy: 0.5025 - loss:
1.3617 - val_accuracy: 0.3043 - val_loss: 2.6359
Epoch 10/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 196ms/step - accuracy: 0.5715 - loss:
1.2661 - val_accuracy: 0.2174 - val_loss: 2.8661
Epoch 11/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 206ms/step - accuracy: 0.5111 - loss:
1.2191 - val_accuracy: 0.2609 - val_loss: 2.8162
Epoch 12/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 190ms/step - accuracy: 0.5203 - loss:
1.1956 - val_accuracy: 0.3913 - val_loss: 2.9778
```

```
Epoch 13/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 194ms/step - accuracy: 0.6113 - loss:
1.0715 - val_accuracy: 0.3913 - val_loss: 3.0385
Epoch 14/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 198ms/step - accuracy: 0.6662 - loss:
0.9657 - val_accuracy: 0.3913 - val_loss: 3.2060
Epoch 15/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 192ms/step - accuracy: 0.6215 - loss:
0.9633 - val_accuracy: 0.3913 - val_loss: 3.1358
Epoch 16/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 183ms/step - accuracy: 0.6676 - loss:
0.8551 - val_accuracy: 0.3043 - val_loss: 3.3309
Epoch 17/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 187ms/step - accuracy: 0.7714 - loss:
0.7581 - val_accuracy: 0.3478 - val_loss: 3.4528
Epoch 18/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 191ms/step - accuracy: 0.7801 - loss:
0.6582 - val_accuracy: 0.3478 - val_loss: 3.5074
Epoch 19/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 190ms/step - accuracy: 0.8365 - loss:
0.5512 - val_accuracy: 0.3478 - val_loss: 3.6778
Epoch 20/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 208ms/step - accuracy: 0.8495 - loss:
0.4511 - val_accuracy: 0.2609 - val_loss: 4.6831
Epoch 21/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 185ms/step - accuracy: 0.8979 - loss:
0.3221 - val_accuracy: 0.3478 - val_loss: 4.5838
Epoch 22/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 187ms/step - accuracy: 0.9697 - loss:
0.2236 - val_accuracy: 0.3043 - val_loss: 5.3324
Epoch 23/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 207ms/step - accuracy: 0.9682 - loss:
0.1711 - val_accuracy: 0.2609 - val_loss: 6.1082
Epoch 24/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 191ms/step - accuracy: 0.9632 - loss:
0.2086 - val_accuracy: 0.2174 - val_loss: 6.4351
Epoch 25/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 220ms/step - accuracy: 0.9764 - loss:
0.1923 - val_accuracy: 0.3478 - val_loss: 6.2613
Epoch 26/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 212ms/step - accuracy: 0.9947 - loss:
0.1353 - val_accuracy: 0.3043 - val_loss: 6.7067
Epoch 27/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 198ms/step - accuracy: 0.9620 - loss:
0.1480 - val_accuracy: 0.2609 - val_loss: 7.0985
Epoch 28/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 207ms/step - accuracy: 0.8460 - loss:
0.2981 - val_accuracy: 0.3913 - val_loss: 8.0412
Epoch 29/30
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 199ms/step - accuracy: 0.7847 - loss:
0.7676 - val_accuracy: 0.3043 - val_loss: 5.9355
Epoch 30/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 190ms/step - accuracy: 0.9315 - loss:
0.2773 - val_accuracy: 0.3043 - val_loss: 5.9599


# accuracy & loss graph


acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(10,10))
plt.subplot(1,2,1)
plt.plot(epochs_range, acc, label = 'Training Accuracy')
plt.plot(epochs_range, val_acc, label = 'Validation Accuracy')
plt.legend(loc = 'lower right')
plt.title('Training & Validation Accuracy')

plt.subplot(1,2,2)
plt.plot(epochs_range, loss, label = 'Training Loss')
plt.plot(epochs_range, val_loss, label = 'Validation Loss')
plt.legend(loc = 'upper right')
plt.title('Training & Validation Loss')


Text(0.5, 1.0, 'Training & Validation Loss')
```




```
#M2 Model

data_augmentation = keras.Sequential([
    layers.RandomFlip("horizontal_and_vertical",
input_shape=(img_height, img_width, 3)),
    layers.RandomRotation(0.2, fill_mode='reflect'),
    layers.RandomZoom(height_factor=(0.2, 0.3), width_factor=(0.2,
0.3), fill_mode='reflect')
])
```

```
model = Sequential([
         data_augmentation,
         layers.Rescaling(1./255, input_shape=(img_height, img_width,
3)),  # Assuming you want to apply rescaling after data augmentation
         layers.Conv2D(16, 3, padding='same', activation="relu"),
         layers.MaxPool2D((2, 2), strides=2),
         layers.Conv2D(32, 3, padding='same', activation="relu"),
         layers.MaxPool2D((2, 2), strides=2),
         layers.Conv2D(64, 3, padding='same', activation="relu"),
         layers.MaxPool2D((2, 2), strides=2),
         layers.Flatten(),
         layers.Dense(128, activation="relu"),
         layers.Dense(num_classes)
])



model.compile(optimizer="adam",loss =
tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
metrics = ['accuracy'])

#train the model : run the model on train & validation set
epochs = 30
history = model.fit( train_ds , validation_data= val_ds , epochs =
epochs)
```

```
Epoch 1/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 3s 243ms/step - accuracy: 0.0881 - loss:
2.5306 - val_accuracy: 0.0000e+00 - val_loss: 2.4248
Epoch 2/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 194ms/step - accuracy: 0.1513 - loss:
2.0920 - val_accuracy: 0.1304 - val_loss: 2.2952
Epoch 3/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 229ms/step - accuracy: 0.2121 - loss:
2.0534 - val_accuracy: 0.1304 - val_loss: 2.2311
Epoch 4/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 220ms/step - accuracy: 0.2225 - loss:
1.9333 - val_accuracy: 0.1304 - val_loss: 2.2285
Epoch 5/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 218ms/step - accuracy: 0.2665 - loss:
1.8412 - val_accuracy: 0.0435 - val_loss: 2.4109
Epoch 6/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 252ms/step - accuracy: 0.2472 - loss:
1.7277 - val_accuracy: 0.1739 - val_loss: 2.4468
Epoch 7/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 255ms/step - accuracy: 0.2773 - loss:
1.6435 - val_accuracy: 0.0870 - val_loss: 2.7218
Epoch 8/30
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 233ms/step - accuracy: 0.3153 - loss:
1.6526 - val_accuracy: 0.0870 - val_loss: 3.0242
Epoch 9/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 229ms/step - accuracy: 0.3154 - loss:
1.6027 - val_accuracy: 0.1304 - val_loss: 2.9861
Epoch 10/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 232ms/step - accuracy: 0.3951 - loss:
1.5433 - val_accuracy: 0.1739 - val_loss: 2.9830
Epoch 11/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 222ms/step - accuracy: 0.5201 - loss:
1.4651 - val_accuracy: 0.3043 - val_loss: 2.9866
Epoch 12/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 251ms/step - accuracy: 0.3810 - loss:
1.4568 - val_accuracy: 0.1739 - val_loss: 3.1828
Epoch 13/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 245ms/step - accuracy: 0.4144 - loss:
1.4082 - val_accuracy: 0.2174 - val_loss: 3.1503
Epoch 14/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 222ms/step - accuracy: 0.4349 - loss:
1.3937 - val_accuracy: 0.1739 - val_loss: 3.1498
Epoch 15/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 262ms/step - accuracy: 0.5059 - loss:
1.3260 - val_accuracy: 0.3043 - val_loss: 3.1633
Epoch 16/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 242ms/step - accuracy: 0.4515 - loss:
1.3519 - val_accuracy: 0.1739 - val_loss: 3.4107
Epoch 17/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 234ms/step - accuracy: 0.5137 - loss:
1.3513 - val_accuracy: 0.2174 - val_loss: 3.4690
Epoch 18/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 277ms/step - accuracy: 0.5097 - loss:
1.3120 - val_accuracy: 0.2609 - val_loss: 3.5477
Epoch 19/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 270ms/step - accuracy: 0.5178 - loss:
1.2738 - val_accuracy: 0.3043 - val_loss: 3.6587
Epoch 20/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 284ms/step - accuracy: 0.5955 - loss:
1.2109 - val_accuracy: 0.2609 - val_loss: 3.5997
Epoch 21/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 279ms/step - accuracy: 0.5313 - loss:
1.1933 - val_accuracy: 0.1739 - val_loss: 3.8846
Epoch 22/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 229ms/step - accuracy: 0.5079 - loss:
1.2007 - val_accuracy: 0.2609 - val_loss: 3.8754
Epoch 23/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 269ms/step - accuracy: 0.5293 - loss:
1.2218 - val_accuracy: 0.2609 - val_loss: 3.8365
Epoch 24/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 233ms/step - accuracy: 0.5192 - loss:
```

```
1.2219 - val_accuracy: 0.2609 - val_loss: 3.8496
Epoch 25/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 250ms/step - accuracy: 0.5797 - loss:
1.1835 - val_accuracy: 0.3043 - val_loss: 3.8653
Epoch 26/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 238ms/step - accuracy: 0.5001 - loss:
1.1314 - val_accuracy: 0.2174 - val_loss: 4.2012
Epoch 27/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 229ms/step - accuracy: 0.5007 - loss:
1.1581 - val_accuracy: 0.3913 - val_loss: 3.8269
Epoch 28/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 220ms/step - accuracy: 0.5706 - loss:
1.1056 - val_accuracy: 0.3478 - val_loss: 3.8536
Epoch 29/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 225ms/step - accuracy: 0.6059 - loss:
1.0979 - val_accuracy: 0.1304 - val_loss: 4.3037
Epoch 30/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 227ms/step - accuracy: 0.5419 - loss:
1.1319 - val_accuracy: 0.3043 - val_loss: 4.0241


#M3 Model Augmentation and dropout
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dropout,
Flatten, Dense, Rescaling
from tensorflow.keras.layers import RandomFlip, RandomRotation,
RandomZoom

# Define data augmentation pipeline
data_augmentation = Sequential([
    RandomFlip("horizontal_and_vertical", input_shape=(img_height,
img_width, 3)),
    RandomRotation(0.2),
    RandomZoom(height_factor=(0.2, 0.3), width_factor=(0.2, 0.3))
])

# Define the model
model = Sequential([
    data_augmentation,
    Rescaling(1./255),
    Conv2D(16, 3, padding='same', activation="relu"),
    MaxPooling2D((2, 2), strides=2),
    Conv2D(32, 3, padding='same', activation="relu"),
    MaxPooling2D((2, 2), strides=2),
    Conv2D(64, 3, padding='same', activation="relu"),
    MaxPooling2D((2, 2), strides=2),
    Dropout(0.2),  # dropout layer
    Flatten(),
    Dense(128, activation="relu"),
    Dense(num_classes)
```

```
])

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

# Print model summary






model.compile(optimizer="adam",loss =
tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
metrics = ['accuracy'])


#train the model : run the model on train & validation set
epochs = 30
history = model.fit( train_ds , validation_data= val_ds , epochs =
epochs)
```

```
Epoch 1/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 3s 256ms/step - accuracy: 0.0868 - loss:
2.8995 - val_accuracy: 0.0000e+00 - val_loss: 2.7244
Epoch 2/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 223ms/step - accuracy: 0.2214 - loss:
2.2890 - val_accuracy: 0.1304 - val_loss: 2.2889
Epoch 3/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 261ms/step - accuracy: 0.1377 - loss:
2.0495 - val_accuracy: 0.0435 - val_loss: 2.2852
Epoch 4/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 275ms/step - accuracy: 0.1858 - loss:
2.0401 - val_accuracy: 0.0435 - val_loss: 2.2182
Epoch 5/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 265ms/step - accuracy: 0.2105 - loss:
1.9875 - val_accuracy: 0.0435 - val_loss: 2.2995
Epoch 6/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 242ms/step - accuracy: 0.1957 - loss:
1.9273 - val_accuracy: 0.1304 - val_loss: 2.2767
Epoch 7/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 243ms/step - accuracy: 0.3576 - loss:
1.8363 - val_accuracy: 0.1304 - val_loss: 2.2732
Epoch 8/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 227ms/step - accuracy: 0.3219 - loss:
1.7541 - val_accuracy: 0.1304 - val_loss: 2.2525
Epoch 9/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 235ms/step - accuracy: 0.2751 - loss:
1.7514 - val_accuracy: 0.2174 - val_loss: 2.2127
Epoch 10/30
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 227ms/step - accuracy: 0.3560 - loss:
1.6222 - val_accuracy: 0.2609 - val_loss: 2.3603
Epoch 11/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 237ms/step - accuracy: 0.3312 - loss:
1.6379 - val_accuracy: 0.1304 - val_loss: 2.2383
Epoch 12/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 241ms/step - accuracy: 0.3574 - loss:
1.6179 - val_accuracy: 0.2174 - val_loss: 2.2743
Epoch 13/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 234ms/step - accuracy: 0.3906 - loss:
1.5329 - val_accuracy: 0.2609 - val_loss: 2.3557
Epoch 14/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 229ms/step - accuracy: 0.4502 - loss:
1.4839 - val_accuracy: 0.1739 - val_loss: 2.3780
Epoch 15/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 246ms/step - accuracy: 0.3177 - loss:
1.5420 - val_accuracy: 0.0870 - val_loss: 2.4315
Epoch 16/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 242ms/step - accuracy: 0.3810 - loss:
1.4694 - val_accuracy: 0.1739 - val_loss: 2.4948
Epoch 17/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 217ms/step - accuracy: 0.3959 - loss:
1.4335 - val_accuracy: 0.2174 - val_loss: 2.4974
Epoch 18/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 233ms/step - accuracy: 0.4568 - loss:
1.4525 - val_accuracy: 0.2174 - val_loss: 2.4282
Epoch 19/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 228ms/step - accuracy: 0.4802 - loss:
1.4046 - val_accuracy: 0.1304 - val_loss: 2.5801
Epoch 20/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 221ms/step - accuracy: 0.4342 - loss:
1.4150 - val_accuracy: 0.2174 - val_loss: 2.5248
Epoch 21/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 230ms/step - accuracy: 0.4986 - loss:
1.3440 - val_accuracy: 0.3043 - val_loss: 2.5004
Epoch 22/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 234ms/step - accuracy: 0.4979 - loss:
1.3153 - val_accuracy: 0.2174 - val_loss: 2.7714
Epoch 23/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 227ms/step - accuracy: 0.5083 - loss:
1.3049 - val_accuracy: 0.2174 - val_loss: 2.7122
Epoch 24/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 237ms/step - accuracy: 0.4909 - loss:
1.2948 - val_accuracy: 0.3043 - val_loss: 2.6683
Epoch 25/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 225ms/step - accuracy: 0.5627 - loss:
1.2159 - val_accuracy: 0.2174 - val_loss: 2.7679
Epoch 26/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 218ms/step - accuracy: 0.4847 - loss:
```

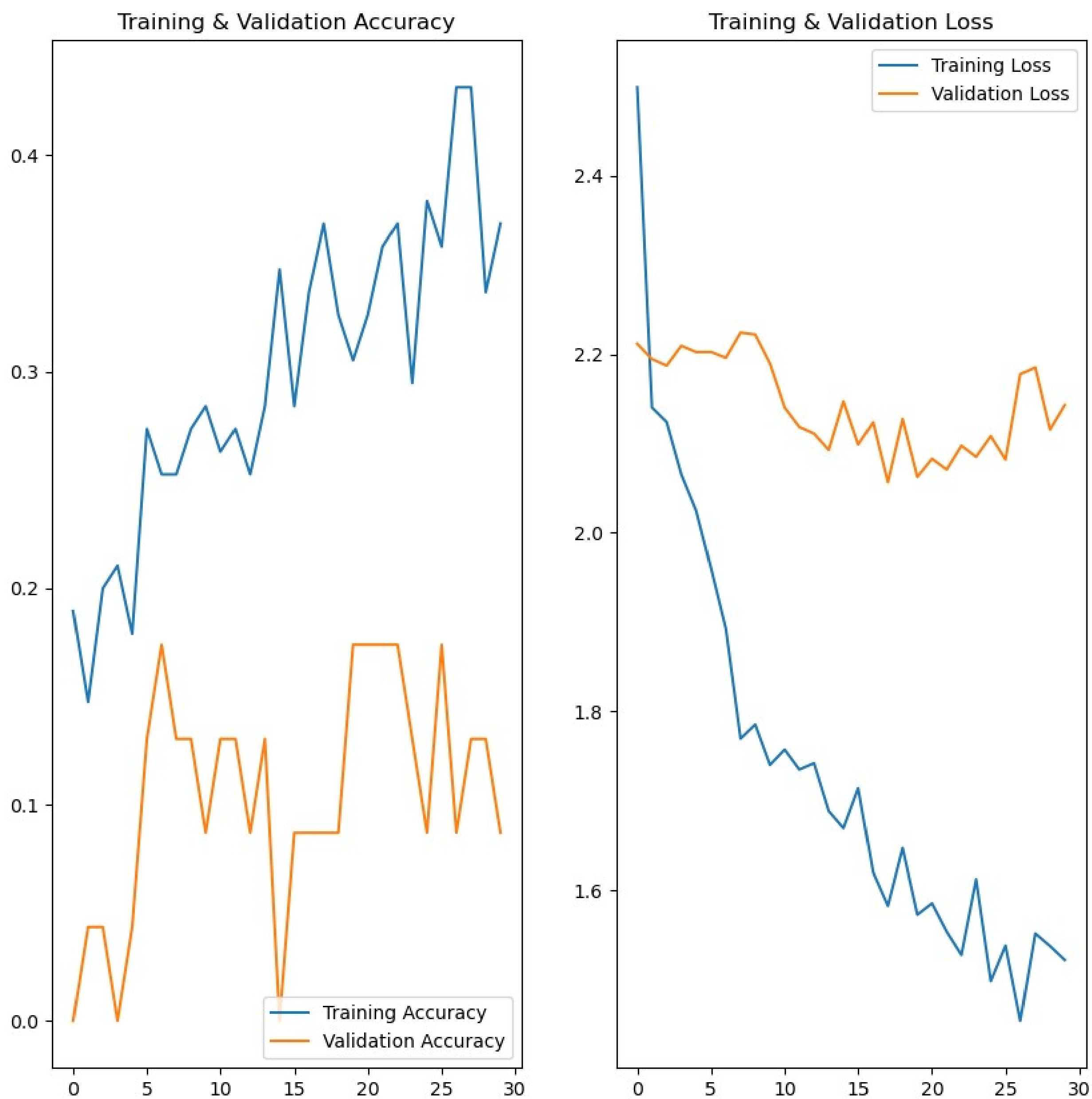
```
1.2078 - val_accuracy: 0.2174 - val_loss: 2.8386
Epoch 27/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 242ms/step - accuracy: 0.5308 - loss:
1.2208 - val_accuracy: 0.2609 - val_loss: 2.8478
Epoch 28/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 235ms/step - accuracy: 0.5746 - loss:
1.1632 - val_accuracy: 0.3043 - val_loss: 2.8886
Epoch 29/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 230ms/step - accuracy: 0.4979 - loss:
1.1797 - val_accuracy: 0.2174 - val_loss: 2.9689
Epoch 30/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 234ms/step - accuracy: 0.5784 - loss:
1.1700 - val_accuracy: 0.3043 - val_loss: 2.8989

#M4 Model with Augumentation + Droupouts ( to additional Layers))
model = Sequential([
         data_augmentation,

layers.Rescaling(1./255,input_shape=(img_height,img_width,3)),
         layers.Conv2D(16,3,padding='same',activation="relu"),
         layers.MaxPool2D((2,2),strides=2),

         layers.Conv2D(32,3,padding='same',activation="relu"),
         layers.MaxPool2D((2,2),strides=2),
         layers.Dropout(0.25), # droupout layer

         layers.Conv2D(64,3,padding='same',activation="relu"),
         layers.MaxPool2D((2,2),strides=2),
         layers.Dropout(0.25), # droupout layer

         layers.Conv2D(128,3,padding='same',activation="relu"),
         layers.MaxPool2D((2,2),strides=2),
         layers.Dropout(0.25), # droupout layer

         layers.Flatten(),
         layers.Dense(128,activation="relu"),
         layers.Dropout(0.25), # droupout layer

         layers.Dense(num_classes)
])

model.compile(optimizer="adam",loss =
tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
metrics = ['accuracy'])



#train the model : run the model on train & validation set
epochs = 30
history = model.fit( train_ds , validation_data= val_ds , epochs =
```

```
epochs)


Epoch 1/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 3s 283ms/step - accuracy: 0.1729 - loss:
2.4765 - val_accuracy: 0.0000e+00 - val_loss: 2.2116
Epoch 2/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 237ms/step - accuracy: 0.1440 - loss:
2.1629 - val_accuracy: 0.0435 - val_loss: 2.1945
Epoch 3/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 279ms/step - accuracy: 0.1781 - loss:
2.1428 - val_accuracy: 0.0435 - val_loss: 2.1871
Epoch 4/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 243ms/step - accuracy: 0.1999 - loss:
2.0850 - val_accuracy: 0.0000e+00 - val_loss: 2.2094
Epoch 5/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 252ms/step - accuracy: 0.1721 - loss:
2.0125 - val_accuracy: 0.0435 - val_loss: 2.2024
Epoch 6/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 259ms/step - accuracy: 0.2353 - loss:
1.9730 - val_accuracy: 0.1304 - val_loss: 2.2025
Epoch 7/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 256ms/step - accuracy: 0.2719 - loss:
1.9011 - val_accuracy: 0.1739 - val_loss: 2.1959
Epoch 8/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 264ms/step - accuracy: 0.2877 - loss:
1.7620 - val_accuracy: 0.1304 - val_loss: 2.2243
Epoch 9/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 260ms/step - accuracy: 0.2528 - loss:
1.7896 - val_accuracy: 0.1304 - val_loss: 2.2220
Epoch 10/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 272ms/step - accuracy: 0.2827 - loss:
1.7488 - val_accuracy: 0.0870 - val_loss: 2.1895
Epoch 11/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 268ms/step - accuracy: 0.2796 - loss:
1.7432 - val_accuracy: 0.1304 - val_loss: 2.1406
Epoch 12/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 250ms/step - accuracy: 0.2608 - loss:
1.7526 - val_accuracy: 0.1304 - val_loss: 2.1187
Epoch 13/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 250ms/step - accuracy: 0.2669 - loss:
1.7660 - val_accuracy: 0.0870 - val_loss: 2.1113
Epoch 14/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 263ms/step - accuracy: 0.2759 - loss:
1.6852 - val_accuracy: 0.1304 - val_loss: 2.0929
Epoch 15/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 268ms/step - accuracy: 0.3260 - loss:
1.6903 - val_accuracy: 0.0000e+00 - val_loss: 2.1475
Epoch 16/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 286ms/step - accuracy: 0.2782 - loss:
```

```
1.7217 - val_accuracy: 0.0870 - val_loss: 2.0991
Epoch 17/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 271ms/step - accuracy: 0.3022 - loss:
1.6366 - val_accuracy: 0.0870 - val_loss: 2.1239
Epoch 18/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 282ms/step - accuracy: 0.4003 - loss:
1.5804 - val_accuracy: 0.0870 - val_loss: 2.0567
Epoch 19/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 275ms/step - accuracy: 0.3128 - loss:
1.6636 - val_accuracy: 0.0870 - val_loss: 2.1277
Epoch 20/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 268ms/step - accuracy: 0.3089 - loss:
1.5768 - val_accuracy: 0.1739 - val_loss: 2.0627
Epoch 21/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 269ms/step - accuracy: 0.3624 - loss:
1.5630 - val_accuracy: 0.1739 - val_loss: 2.0831
Epoch 22/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 274ms/step - accuracy: 0.3664 - loss:
1.5482 - val_accuracy: 0.1739 - val_loss: 2.0709
Epoch 23/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 284ms/step - accuracy: 0.3401 - loss:
1.5172 - val_accuracy: 0.1739 - val_loss: 2.0978
Epoch 24/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 293ms/step - accuracy: 0.3115 - loss:
1.5706 - val_accuracy: 0.1304 - val_loss: 2.0852
Epoch 25/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 285ms/step - accuracy: 0.4004 - loss:
1.4998 - val_accuracy: 0.0870 - val_loss: 2.1084
Epoch 26/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 267ms/step - accuracy: 0.3562 - loss:
1.5431 - val_accuracy: 0.1739 - val_loss: 2.0821
Epoch 27/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 261ms/step - accuracy: 0.4238 - loss:
1.4430 - val_accuracy: 0.0870 - val_loss: 2.1776
Epoch 28/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 261ms/step - accuracy: 0.4198 - loss:
1.5511 - val_accuracy: 0.1304 - val_loss: 2.1850
Epoch 29/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 262ms/step - accuracy: 0.3337 - loss:
1.5617 - val_accuracy: 0.1304 - val_loss: 2.1158
Epoch 30/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 253ms/step - accuracy: 0.3642 - loss:
1.5490 - val_accuracy: 0.0870 - val_loss: 2.1433

#M5 Model Additional Experiment with Dropouts

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
```

```python
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(10,10))
plt.subplot(1,2,1)
plt.plot(epochs_range, acc, label = 'Training Accuracy')
plt.plot(epochs_range, val_acc, label = 'Validation Accuracy')
plt.legend(loc = 'lower right')
plt.title('Training & Validation Accuracy')

plt.subplot(1,2,2)
plt.plot(epochs_range, loss, label = 'Training Loss')
plt.plot(epochs_range, val_loss, label = 'Validation Loss')
plt.legend(loc = 'upper right')
plt.title('Training & Validation Loss')
```

```
Text(0.5, 1.0, 'Training & Validation Loss')
```




```
model = Sequential([
         data_augmentation,

layers.Rescaling(1./255,input_shape=(img_height,img_width,3)),
         layers.Conv2D(16,3,padding='same',activation="relu"),
         layers.MaxPool2D((2,2),strides=2),

         layers.Conv2D(32,3,padding='same',activation="relu"),
         layers.MaxPool2D((2,2),strides=2),
         #layers.Dropout(0.25), # droupout layer
```

```
        layers.Conv2D(64,3,padding='same',activation="relu"),
        layers.MaxPool2D((2,2),strides=2),
        #layers.Dropout(0.25), # droupout layer

        layers.Conv2D(128,3,padding='same',activation="relu"),
        layers.MaxPool2D((2,2),strides=2),
        #layers.Dropout(0.25), # droupout layer

        layers.Flatten(),
        layers.Dense(128,activation="relu"),
        layers.Dropout(0.25), # droupout layer

        layers.Dense(num_classes)
])



model.compile(optimizer="adam",loss =
tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
metrics = ['accuracy'])



#train the model : run the model on train & validation set
epochs = 30
history = model.fit( train_ds , validation_data= val_ds , epochs =
epochs)


Epoch 1/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 3s 243ms/step - accuracy: 0.1374 - loss:
2.2065 - val_accuracy: 0.0000e+00 - val_loss: 2.3216
Epoch 2/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 224ms/step - accuracy: 0.1733 - loss:
2.1094 - val_accuracy: 0.0000e+00 - val_loss: 2.1892
Epoch 3/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 267ms/step - accuracy: 0.1878 - loss:
2.0321 - val_accuracy: 0.0435 - val_loss: 2.3436
Epoch 4/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 255ms/step - accuracy: 0.2370 - loss:
1.9356 - val_accuracy: 0.0870 - val_loss: 2.3648
Epoch 5/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 265ms/step - accuracy: 0.2142 - loss:
1.8081 - val_accuracy: 0.1304 - val_loss: 2.2977
Epoch 6/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 280ms/step - accuracy: 0.2874 - loss:
1.7338 - val_accuracy: 0.0870 - val_loss: 2.2285
Epoch 7/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 254ms/step - accuracy: 0.2706 - loss:
1.7408 - val_accuracy: 0.0435 - val_loss: 2.2678
Epoch 8/30
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 259ms/step - accuracy: 0.2318 - loss:
1.7057 - val_accuracy: 0.1304 - val_loss: 2.3394
Epoch 9/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 271ms/step - accuracy: 0.3537 - loss:
1.6779 - val_accuracy: 0.0435 - val_loss: 2.4007
Epoch 10/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 265ms/step - accuracy: 0.2880 - loss:
1.7225 - val_accuracy: 0.0870 - val_loss: 2.3180
Epoch 11/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 271ms/step - accuracy: 0.3803 - loss:
1.6471 - val_accuracy: 0.0870 - val_loss: 2.1861
Epoch 12/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 275ms/step - accuracy: 0.2675 - loss:
1.6007 - val_accuracy: 0.0870 - val_loss: 2.2940
Epoch 13/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 245ms/step - accuracy: 0.3957 - loss:
1.5327 - val_accuracy: 0.0435 - val_loss: 2.2945
Epoch 14/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 247ms/step - accuracy: 0.3678 - loss:
1.5471 - val_accuracy: 0.0870 - val_loss: 2.4448
Epoch 15/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 236ms/step - accuracy: 0.4347 - loss:
1.4949 - val_accuracy: 0.0870 - val_loss: 2.2842
Epoch 16/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 255ms/step - accuracy: 0.4527 - loss:
1.4403 - val_accuracy: 0.2174 - val_loss: 2.2256
Epoch 17/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 246ms/step - accuracy: 0.4849 - loss:
1.3738 - val_accuracy: 0.0870 - val_loss: 2.3220
Epoch 18/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 254ms/step - accuracy: 0.4632 - loss:
1.4237 - val_accuracy: 0.2174 - val_loss: 2.5089
Epoch 19/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 251ms/step - accuracy: 0.4362 - loss:
1.4052 - val_accuracy: 0.2609 - val_loss: 2.3840
Epoch 20/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 255ms/step - accuracy: 0.4876 - loss:
1.3652 - val_accuracy: 0.1739 - val_loss: 2.4578
Epoch 21/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 246ms/step - accuracy: 0.5011 - loss:
1.3908 - val_accuracy: 0.1304 - val_loss: 2.4421
Epoch 22/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 252ms/step - accuracy: 0.4699 - loss:
1.3196 - val_accuracy: 0.1739 - val_loss: 2.3020
Epoch 23/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 260ms/step - accuracy: 0.5400 - loss:
1.3031 - val_accuracy: 0.1739 - val_loss: 2.4269
Epoch 24/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 258ms/step - accuracy: 0.4927 - loss:
```

```
1.2983 - val_accuracy: 0.1739 - val_loss: 2.4506
Epoch 25/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 259ms/step - accuracy: 0.3720 - loss:
1.4063 - val_accuracy: 0.2174 - val_loss: 2.4148
Epoch 26/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 254ms/step - accuracy: 0.5136 - loss:
1.2220 - val_accuracy: 0.2174 - val_loss: 2.5811
Epoch 27/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 263ms/step - accuracy: 0.4725 - loss:
1.2925 - val_accuracy: 0.2609 - val_loss: 2.5225
Epoch 28/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 264ms/step - accuracy: 0.5297 - loss:
1.3128 - val_accuracy: 0.1739 - val_loss: 2.6503
Epoch 29/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 263ms/step - accuracy: 0.4367 - loss:
1.3069 - val_accuracy: 0.3043 - val_loss: 2.5123
Epoch 30/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 265ms/step - accuracy: 0.4738 - loss:
1.2110 - val_accuracy: 0.2609 - val_loss: 2.6251
```

```python
#M6 Model ( Augumetation + Batch Normalization + Droupouts)

model = Sequential([
         data_augmentation,

layers.Rescaling(1./255,input_shape=(img_height,img_width,3)),
         layers.Conv2D(16,3,padding='same',activation="relu"),
         layers.MaxPool2D((2,2),strides=2),
         layers.BatchNormalization(),
         layers.Dropout(0.25), # droupout layer

         layers.Conv2D(32,3,padding='same',activation="relu"),
         layers.MaxPool2D((2,2),strides=2),
         layers.BatchNormalization(),
         layers.Dropout(0.25), # droupout layer

         layers.Conv2D(64,3,padding='same',activation="relu"),
         layers.MaxPool2D((2,2),strides=2),
         layers.BatchNormalization(),
         layers.Dropout(0.25), # droupout layer

         layers.Conv2D(128,3,padding='same',activation="relu"),
         layers.MaxPool2D((2,2),strides=2),
         layers.BatchNormalization(),
         layers.Dropout(0.25), # droupout layer

         layers.Flatten(),
         layers.Dense(128,activation="relu"),

         layers.Dense(num_classes)
```

```
])



model.compile(optimizer="adam",loss =
tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
metrics = ['accuracy'])


#train the model : run the model on train & validation set
epochs = 30
history = model.fit( train_ds , validation_data= val_ds , epochs =
epochs)
```

```
Epoch 1/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 5s 330ms/step - accuracy: 0.1906 - loss:
4.1456 - val_accuracy: 0.1739 - val_loss: 2.2830
Epoch 2/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 270ms/step - accuracy: 0.3415 - loss:
3.8210 - val_accuracy: 0.0000e+00 - val_loss: 2.2507
Epoch 3/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 306ms/step - accuracy: 0.4266 - loss:
2.2890 - val_accuracy: 0.0870 - val_loss: 2.2793
Epoch 4/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 304ms/step - accuracy: 0.4778 - loss:
2.1885 - val_accuracy: 0.0870 - val_loss: 2.2257
Epoch 5/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 315ms/step - accuracy: 0.4745 - loss:
1.9244 - val_accuracy: 0.1739 - val_loss: 2.2581
Epoch 6/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 308ms/step - accuracy: 0.5600 - loss:
1.3585 - val_accuracy: 0.0870 - val_loss: 2.3562
Epoch 7/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 314ms/step - accuracy: 0.6332 - loss:
1.6776 - val_accuracy: 0.0870 - val_loss: 2.5025
Epoch 8/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 329ms/step - accuracy: 0.6638 - loss:
1.2158 - val_accuracy: 0.0870 - val_loss: 2.4928
Epoch 9/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 324ms/step - accuracy: 0.6021 - loss:
1.6387 - val_accuracy: 0.1739 - val_loss: 2.4698
Epoch 10/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 332ms/step - accuracy: 0.5719 - loss:
1.1380 - val_accuracy: 0.1739 - val_loss: 2.5642
Epoch 11/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 320ms/step - accuracy: 0.6228 - loss:
1.2439 - val_accuracy: 0.1739 - val_loss: 2.7145
Epoch 12/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 327ms/step - accuracy: 0.7276 - loss:
```

```
0.8087 - val_accuracy: 0.1739 - val_loss: 2.9299
Epoch 13/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 328ms/step - accuracy: 0.6857 - loss:
0.7972 - val_accuracy: 0.1739 - val_loss: 3.1056
Epoch 14/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 338ms/step - accuracy: 0.7852 - loss:
0.6020 - val_accuracy: 0.1739 - val_loss: 3.2863
Epoch 15/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 324ms/step - accuracy: 0.7436 - loss:
0.7617 - val_accuracy: 0.1739 - val_loss: 3.5004
Epoch 16/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 316ms/step - accuracy: 0.7333 - loss:
0.6357 - val_accuracy: 0.1739 - val_loss: 3.8423
Epoch 17/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 303ms/step - accuracy: 0.8545 - loss:
0.4002 - val_accuracy: 0.1739 - val_loss: 4.4237
Epoch 18/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 310ms/step - accuracy: 0.8476 - loss:
0.4226 - val_accuracy: 0.1739 - val_loss: 4.9500
Epoch 19/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 308ms/step - accuracy: 0.8265 - loss:
0.5204 - val_accuracy: 0.1739 - val_loss: 5.1823
Epoch 20/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 306ms/step - accuracy: 0.8142 - loss:
0.5669 - val_accuracy: 0.1739 - val_loss: 5.2829
Epoch 21/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 316ms/step - accuracy: 0.7972 - loss:
0.5530 - val_accuracy: 0.1739 - val_loss: 5.3011
Epoch 22/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 310ms/step - accuracy: 0.8331 - loss:
0.6489 - val_accuracy: 0.1739 - val_loss: 5.4742
Epoch 23/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 310ms/step - accuracy: 0.8856 - loss:
0.3419 - val_accuracy: 0.1739 - val_loss: 5.7668
Epoch 24/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 304ms/step - accuracy: 0.8962 - loss:
0.4193 - val_accuracy: 0.1739 - val_loss: 6.2826
Epoch 25/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 332ms/step - accuracy: 0.8277 - loss:
0.4127 - val_accuracy: 0.1739 - val_loss: 6.3154
Epoch 26/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 325ms/step - accuracy: 0.9489 - loss:
0.2328 - val_accuracy: 0.1739 - val_loss: 6.4912
Epoch 27/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 336ms/step - accuracy: 0.8796 - loss:
0.3736 - val_accuracy: 0.1739 - val_loss: 7.2311
Epoch 28/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 326ms/step - accuracy: 0.9139 - loss:
0.2574 - val_accuracy: 0.1739 - val_loss: 7.9291
```

```
Epoch 29/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 326ms/step - accuracy: 0.8821 - loss:
0.4741 - val_accuracy: 0.1739 - val_loss: 7.6630
Epoch 30/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 328ms/step - accuracy: 0.9581 - loss:
0.1452 - val_accuracy: 0.1739 - val_loss: 7.8135
```

```python
# Using Another Way of Augmentation to Handle Class Imbalance
```

```
!pip install Augmentor
```

```
Requirement already satisfied: Augmentor in c:\users\jiyan\anaconda3\
lib\site-packages (0.2.12)
Requirement already satisfied: Pillow>=5.2.0 in c:\users\jiyan\
anaconda3\lib\site-packages (from Augmentor) (9.4.0)
Requirement already satisfied: tqdm>=4.9.0 in c:\users\jiyan\
anaconda3\lib\site-packages (from Augmentor) (4.65.0)
Requirement already satisfied: numpy>=1.11.0 in c:\users\jiyan\
anaconda3\lib\site-packages (from Augmentor) (1.24.3)
Requirement already satisfied: colorama in c:\users\jiyan\anaconda3\
lib\site-packages (from tqdm>=4.9.0->Augmentor) (0.4.6)
```

```python
import Augmentor
```

```python
# add 500 new sample to each folder
for class_name in data_detail_pd.index:
  #print(class_name)
  p = Augmentor.Pipeline(str(datate_dir_train)
+"/"+class_name,save_format='.jpg')
  p.rotate(probability=0.7,max_left_rotation=10,max_right_rotation=10)
  p.sample(500)
```

```
Initialised with 16 image(s) found.
Output directory set to extracted_files\Skin cancer ISIC The
International Skin Imaging Collaboration\Test/actinic keratosis\
output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at
0x1D074698350>: 100%|██████████| 500/500 [00:03<00:00, 135.16
Samples/s]

Initialised with 16 image(s) found.
Output directory set to extracted_files\Skin cancer ISIC The
International Skin Imaging Collaboration\Test/basal cell carcinoma\
output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at
0x1D020483810>: 100%|██████████| 500/500 [00:04<00:00, 111.02
Samples/s]
```

```
Initialised with 16 image(s) found.
Output directory set to extracted_files\Skin cancer ISIC The
International Skin Imaging Collaboration\Test/dermatofibroma\output.

Processing <PIL.Image.Image image mode=RGB size=6648x4459 at
0x1D07320CFD0>: 100%|██████████| 500/500 [02:13<00:00,  3.74
Samples/s]

Initialised with 16 image(s) found.
Output directory set to extracted_files\Skin cancer ISIC The
International Skin Imaging Collaboration\Test/melanoma\output.

Processing <PIL.Image.Image image mode=RGB size=1504x1129 at
0x1D037E1F450>: 100%|██████████| 500/500 [00:16<00:00, 29.65
Samples/s]

Initialised with 16 image(s) found.
Output directory set to extracted_files\Skin cancer ISIC The
International Skin Imaging Collaboration\Test/nevus\output.

Processing <PIL.Image.Image image mode=RGB size=1022x767 at
0x1D07466A710>: 100%|██████████| 500/500 [00:10<00:00, 49.55
Samples/s]

Initialised with 16 image(s) found.
Output directory set to extracted_files\Skin cancer ISIC The
International Skin Imaging Collaboration\Test/pigmented benign
keratosis\output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at
0x1D037C20690>: 100%|██████████| 500/500 [00:04<00:00, 114.47
Samples/s]

Initialised with 3 image(s) found.
Output directory set to extracted_files\Skin cancer ISIC The
International Skin Imaging Collaboration\Test/seborrheic keratosis\
output.

Processing <PIL.Image.Image image mode=RGB size=1024x768 at
0x1D037C08450>: 100%|██████████| 500/500 [00:09<00:00, 52.15
Samples/s]

Initialised with 16 image(s) found.
Output directory set to extracted_files\Skin cancer ISIC The
International Skin Imaging Collaboration\Test/squamous cell carcinoma\
output.

Processing <PIL.JpegImagePlugin.JpegImageFile image mode=RGB
size=600x450 at 0x1D07AC3E590>: 100%|██████████| 500/500 [01:45<00:00,
4.76 Samples/s]
```

```
Initialised with 3 image(s) found.
Output directory set to extracted_files\Skin cancer ISIC The
International Skin Imaging Collaboration\Test/vascular lesion\output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at
0x1D0731C4190>: 100%|██████████| 500/500 [00:04<00:00, 121.83
Samples/s]
```

```
data_detail_pd.index
```

```
Index(['actinic keratosis', 'basal cell carcinoma', 'dermatofibroma',
       'melanoma', 'nevus', 'pigmented benign keratosis',
       'seborrheic keratosis', 'squamous cell carcinoma', 'vascular
lesion'],
      dtype='object', name='Dir_Name')
```

```
datate_dir_train
```

```
WindowsPath('extracted_files/Skin cancer ISIC The International Skin
Imaging Collaboration/Test')
```

```
#count of additional images added

additional_images_added =
len(list(datate_dir_train.glob("*/output/*jpg")))
additional_images_added
```

```
4500
```

```
# we need to reinitalize the train_ds & val_ds
train_ds_new = tf.keras.preprocessing.image_dataset_from_directory(
    datate_dir_train,
    validation_split=0.2,
    subset = "training",
    seed = 123,
    image_size = (img_height,img_width),
    batch_size = batch_size
)
```

```
Found 4618 files belonging to 9 classes.
Using 3695 files for training.
```

```
#validation dataset

val_ds_new = tf.keras.preprocessing.image_dataset_from_directory(
    datate_dir_train,
    validation_split=0.2,
```

```
    subset = "validation",
    seed = 123,
    image_size = (img_height,img_width),
    batch_size = batch_size
)
```

```
Found 4618 files belonging to 9 classes.
Using 923 files for validation.
```

```
AUTOTUNE = tf.data.AUTOTUNE
train_ds =
train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)


# Model Defination

model = Sequential([
  layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.BatchNormalization(),
  layers.Dropout(0.25),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.BatchNormalization(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.25),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])

model.compile(optimizer="adam",loss =
tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
metrics = ['accuracy'])

model.summary()
```

```
Model: "sequential_8"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━
━━━━━┓
┃ Layer (type)                     ┃ Output Shape            ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━
━━━━━┩
│ rescaling_6 (Rescaling)          │ (None, 180, 180, 3)     │
0 │
├──────────────────────────────────┼─────────────────────────┼──────────
```

```
───────┤
│ conv2d_21 (Conv2D)              │ (None, 180, 180, 16)     │
448 │
├─────────────────────────────────┼──────────────────────────┼──────────
───────┤
│ max_pooling2d_21 (MaxPooling2D) │ (None, 90, 90, 16)       │
0 │
├─────────────────────────────────┼──────────────────────────┼──────────
───────┤
│ batch_normalization_4           │ (None, 90, 90, 16)       │
64 │
│ (BatchNormalization)            │                          │
│
├─────────────────────────────────┼──────────────────────────┼──────────
───────┤
│ dropout_10 (Dropout)            │ (None, 90, 90, 16)       │
0 │
├─────────────────────────────────┼──────────────────────────┼──────────
───────┤
│ conv2d_22 (Conv2D)              │ (None, 90, 90, 32)       │
4,640 │
├─────────────────────────────────┼──────────────────────────┼──────────
───────┤
│ max_pooling2d_22 (MaxPooling2D) │ (None, 45, 45, 32)       │
0 │
├─────────────────────────────────┼──────────────────────────┼──────────
───────┤
│ batch_normalization_5           │ (None, 45, 45, 32)       │
128 │
│ (BatchNormalization)            │                          │
│
├─────────────────────────────────┼──────────────────────────┼──────────
───────┤
│ conv2d_23 (Conv2D)              │ (None, 45, 45, 64)       │
18,496 │
├─────────────────────────────────┼──────────────────────────┼──────────
───────┤
│ max_pooling2d_23 (MaxPooling2D) │ (None, 22, 22, 64)       │
0 │
├─────────────────────────────────┼──────────────────────────┼──────────
───────┤
│ dropout_11 (Dropout)            │ (None, 22, 22, 64)       │
0 │
├─────────────────────────────────┼──────────────────────────┼──────────
───────┤
│ flatten_6 (Flatten)             │ (None, 30976)            │
0 │
├─────────────────────────────────┼──────────────────────────┼──────────
───────┤
```

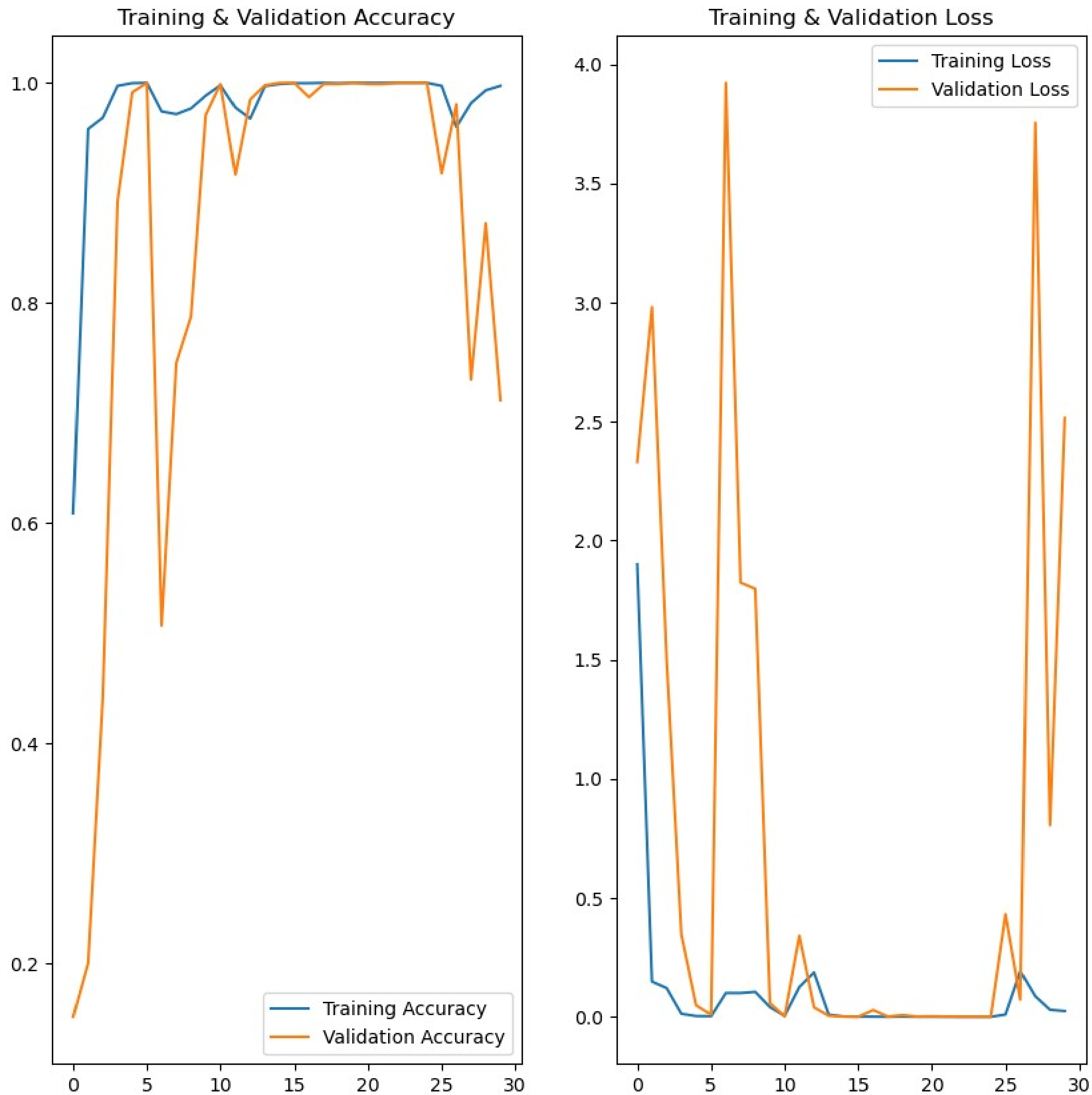
```
│ dense_12 (Dense)                     │ (None, 128)               │
3,965,056 │
├──────────────────────────────────────┼───────────────────────────┼───────────
─────┤
│ dense_13 (Dense)                     │ (None, 9)                 │
1,161 │
└──────────────────────────────────────┴───────────────────────────┴───────────
─────┘

 Total params: 3,989,993 (15.22 MB)

 Trainable params: 3,989,897 (15.22 MB)

 Non-trainable params: 96 (384.00 B)


# run the model to fit train datapoint and check accuracy on
validation dataset

epochs = 30
history = model.fit(
  train_ds_new,
  validation_data=val_ds_new,
  epochs=epochs
)

Epoch 1/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 43s 350ms/step - accuracy: 0.4217 - loss:
4.2383 - val_accuracy: 0.1517 - val_loss: 2.3305
Epoch 2/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 35s 303ms/step - accuracy: 0.9301 - loss:
0.2236 - val_accuracy: 0.2004 - val_loss: 2.9809
Epoch 3/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 35s 302ms/step - accuracy: 0.9567 - loss:
0.1529 - val_accuracy: 0.4420 - val_loss: 1.4994
Epoch 4/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 37s 317ms/step - accuracy: 0.9963 - loss:
0.0164 - val_accuracy: 0.8917 - val_loss: 0.3479
Epoch 5/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 36s 311ms/step - accuracy: 0.9998 - loss:
0.0032 - val_accuracy: 0.9913 - val_loss: 0.0489
Epoch 6/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 35s 303ms/step - accuracy: 1.0000 - loss:
0.0034 - val_accuracy: 1.0000 - val_loss: 0.0100
Epoch 7/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 36s 306ms/step - accuracy: 0.9868 - loss:
0.0518 - val_accuracy: 0.5070 - val_loss: 3.9232
Epoch 8/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 36s 304ms/step - accuracy: 0.9729 - loss:
0.1041 - val_accuracy: 0.7454 - val_loss: 1.8247
Epoch 9/30
```

```
116/116 ━━━━━━━━━━━━━━━━━━━━ 36s 304ms/step - accuracy: 0.9648 - loss:
0.1637 - val_accuracy: 0.7876 - val_loss: 1.7986
Epoch 10/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 38s 326ms/step - accuracy: 0.9858 - loss:
0.0482 - val_accuracy: 0.9707 - val_loss: 0.0577
Epoch 11/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 36s 309ms/step - accuracy: 0.9984 - loss:
0.0042 - val_accuracy: 0.9989 - val_loss: 0.0022
Epoch 12/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 36s 310ms/step - accuracy: 0.9901 - loss:
0.0433 - val_accuracy: 0.9166 - val_loss: 0.3416
Epoch 13/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 36s 306ms/step - accuracy: 0.9544 - loss:
0.2721 - val_accuracy: 0.9848 - val_loss: 0.0402
Epoch 14/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 36s 305ms/step - accuracy: 0.9936 - loss:
0.0160 - val_accuracy: 0.9978 - val_loss: 0.0044
Epoch 15/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 36s 304ms/step - accuracy: 0.9998 - loss:
6.2042e-04 - val_accuracy: 1.0000 - val_loss: 0.0011
Epoch 16/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 36s 307ms/step - accuracy: 0.9997 - loss:
9.2389e-04 - val_accuracy: 1.0000 - val_loss: 4.2611e-04
Epoch 17/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 36s 305ms/step - accuracy: 0.9999 - loss:
5.2646e-04 - val_accuracy: 0.9870 - val_loss: 0.0286
Epoch 18/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 37s 314ms/step - accuracy: 1.0000 - loss:
4.4502e-04 - val_accuracy: 0.9989 - val_loss: 0.0013
Epoch 19/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 36s 311ms/step - accuracy: 0.9999 - loss:
7.1314e-04 - val_accuracy: 0.9989 - val_loss: 0.0076
Epoch 20/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 38s 328ms/step - accuracy: 1.0000 - loss:
3.9190e-04 - val_accuracy: 1.0000 - val_loss: 0.0014
Epoch 21/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 40s 343ms/step - accuracy: 1.0000 - loss:
9.6109e-05 - val_accuracy: 0.9989 - val_loss: 0.0024
Epoch 22/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 39s 333ms/step - accuracy: 1.0000 - loss:
2.4671e-04 - val_accuracy: 0.9989 - val_loss: 0.0012
Epoch 23/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 39s 335ms/step - accuracy: 1.0000 - loss:
4.5808e-05 - val_accuracy: 1.0000 - val_loss: 5.6654e-04
Epoch 24/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 38s 323ms/step - accuracy: 1.0000 - loss:
1.2463e-04 - val_accuracy: 1.0000 - val_loss: 5.4704e-04
Epoch 25/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 37s 313ms/step - accuracy: 1.0000 - loss:
```

```
3.5282e-05 - val_accuracy: 1.0000 - val_loss: 4.6863e-04
Epoch 26/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 35s 298ms/step - accuracy: 0.9993 - loss:
0.0026 - val_accuracy: 0.9177 - val_loss: 0.4322
Epoch 27/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 35s 300ms/step - accuracy: 0.9608 - loss:
0.2016 - val_accuracy: 0.9805 - val_loss: 0.0724
Epoch 28/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 35s 297ms/step - accuracy: 0.9862 - loss:
0.0555 - val_accuracy: 0.7302 - val_loss: 3.7556
Epoch 29/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 35s 297ms/step - accuracy: 0.9879 - loss:
0.0477 - val_accuracy: 0.8722 - val_loss: 0.8051
Epoch 30/30
116/116 ━━━━━━━━━━━━━━━━━━━━ 35s 298ms/step - accuracy: 0.9990 - loss:
0.0031 - val_accuracy: 0.7118 - val_loss: 2.5158
```

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(10,10))
plt.subplot(1,2,1)
plt.plot(epochs_range, acc, label = 'Training Accuracy')
plt.plot(epochs_range, val_acc, label = 'Validation Accuracy')
plt.legend(loc = 'lower right')
plt.title('Training & Validation Accuracy')

plt.subplot(1,2,2)
plt.plot(epochs_range, loss, label = 'Training Loss')
plt.plot(epochs_range, val_loss, label = 'Validation Loss')
plt.legend(loc = 'upper right')
plt.title('Training & Validation Loss')
```

```
Text(0.5, 1.0, 'Training & Validation Loss')
```




```
# Analysis on test data
test_ds = tf.keras.preprocessing.image_dataset_from_directory(
    datate_dir_test,
    seed = 123,
    image_size = (img_height,img_width),
    batch_size = batch_size
)
```

```
Found 2239 files belonging to 9 classes.
```

```
loss , accuracy = model.evaluate(test_ds)
```

```
70/70 ━━━━━━━━━━━━━━━━━━━━ 9s 124ms/step - accuracy: 0.2024 - loss: 
15.8855

print("Accuracy on test data ", accuracy)

Accuracy on test data  0.20232246816158295

#Prediction on New Test Data
melanoma_path = "extracted_files/Skin cancer ISIC The International 
Skin Imaging Collaboration/Test/melanoma/ISIC_0000002.jpg"

img = tf.keras.utils.load_img(
    melanoma_path, target_size=(img_height, img_width)
)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(score)


1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 102ms/step
tf.Tensor(
[0.0000000e+00 0.0000000e+00 0.0000000e+00 1.0000000e+00 1.9656479e-33
 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00], shape=(9,), 
dtype=float32)


print(
    "This image most likely belongs to {} with a {:.2f} percent 
confidence."
    .format(test_ds.class_names[np.argmax(score)], 100 * 
np.max(score))
)


This image most likely belongs to melanoma with a 100.00 percent 
confidence.
```