```python
import numpy as np
import cv2
import os
import random
import matplotlib.pyplot as plt
import PIL.Image as Image
import pickle
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import Model
from tensorflow.keras.utils import plot_model
import tensorflow as tf
from sklearn.metrics import classification_report,confusion_matrix
```

```python
import tensorflow as tf
os.environ["KERAS_BACKEND"] = "tensorflow"
from tensorflow.keras import layers
import keras
from keras.layers import *
from keras import backend
```

```python
DIRECTORY = r'/kaggle/input/brain-tumor-mri-dataset/Training'
CATEGORIES = ['glioma', 'meningioma', 'notumor','pituitary']
IMG_SIZE = 256
patch_size = 16
expansion_factor = 2
train_data = []
```

```python
for category in CATEGORIES:
    folder = os.path.join(DIRECTORY,category)
    #print(folder)
    label = CATEGORIES.index(category)
    for img in os.listdir(folder):
        img_path = os.path.join(folder, img)
        #print(img_path)
        img_arr = cv2.imread(img_path)
        img_arr = cv2.resize(img_arr, (IMG_SIZE, IMG_SIZE))
        #plt.imshow(img_arr)
        #break
        train_data.append([img_arr, label])
```

```python
len(train_data)
```

```
5712
```

```python
random.shuffle(train_data)
```

```python
X_train = []
y_train = []
for features, labels in train_data:
```

```
    X_train.append(features)
    y_train.append(labels)
```

```
DIRECTORY = r'/kaggle/input/brain-tumor-mri-dataset/Testing'
CATEGORIES = ['glioma', 'meningioma', 'notumor','pituitary']
IMG_SIZE = 256
patch_size = 16
expansion_factor = 2
test_data = []
```

```
for category in CATEGORIES:
    folder = os.path.join(DIRECTORY,category)
    #print(folder)
    label = CATEGORIES.index(category)
    for img in os.listdir(folder):
        img_path = os.path.join(folder, img)
        #print(img_path)
        img_arr = cv2.imread(img_path)
        img_arr = cv2.resize(img_arr, (IMG_SIZE, IMG_SIZE))
        #plt.imshow(img_arr)
        #break
        test_data.append([img_arr, label])
```

```
len(test_data)
```

```
1311
```

```
X_test = []
y_test = []
for features, labels in test_data:
    X_test.append(features)
    y_test.append(labels)
```

```
X_train = np.array(X_train)
y_train = np.array(y_train)
X_test = np.array(X_test)
y_test = np.array(y_test)
```

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((5712, 256, 256, 3), (1311, 256, 256, 3), (5712,), (1311,))
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
```

```python
    fill_mode='nearest'
)
datagen.fit(X_train)

def conv_block(x, filters=16, kernel_size=3, strides=2):
    conv_layer = layers.Conv2D(
        filters,
        kernel_size,
        strides=strides,
        activation=keras.activations.swish,
        padding="same",
    )
    return conv_layer(x)


# Reference:
https://github.com/keras-team/keras/blob/e3858739d178fe16a0c77ce7fab88
b0be6dbbdc7/keras/applications/imagenet_utils.py#L413C17-L435


def correct_pad(inputs, kernel_size):
    img_dim = 2 if backend.image_data_format() == "channels_first"
else 1
    input_size = inputs.shape[img_dim : (img_dim + 2)]
    if isinstance(kernel_size, int):
        kernel_size = (kernel_size, kernel_size)
    if input_size[0] is None:
        adjust = (1, 1)
    else:
        adjust = (1 - input_size[0] % 2, 1 - input_size[1] % 2)
    correct = (kernel_size[0] // 2, kernel_size[1] // 2)
    return (
        (correct[0] - adjust[0], correct[0]),
        (correct[1] - adjust[1], correct[1]),
    )


# Reference: https://git.io/JKgtC


def inverted_residual_block(x, expanded_channels, output_channels,
strides=1):
    m = layers.Conv2D(expanded_channels, 1, padding="same",
use_bias=False)(x)
    m = layers.BatchNormalization()(m)
    m = keras.activations.swish(m)

    if strides == 2:
        m = layers.ZeroPadding2D(padding=correct_pad(m, 3))(m)
    m = layers.DepthwiseConv2D(
```

```
        3, strides=strides, padding="same" if strides == 1 else
"valid", use_bias=False
    )(m)
    m = layers.BatchNormalization()(m)
    m = keras.activations.swish(m)

    m = layers.Conv2D(output_channels, 1, padding="same",
use_bias=False)(m)
    m = layers.BatchNormalization()(m)

    if keras.ops.equal(x.shape[-1], output_channels) and strides == 1:
        return layers.Add()([m, x])
    return m

def mlp(x, hidden_units, dropout_rate):
    for units in hidden_units:
        x = layers.Dense(units, activation=keras.activations.swish)(x)
        x = layers.Dropout(dropout_rate)(x)
    return x

def transformer_block(x, transformer_layers, projection_dim,
num_heads=2):
    for _ in range(transformer_layers):
        # Layer normalization 1.
        x1 = layers.LayerNormalization(epsilon=1e-6)(x)
        # Create a multi-head attention layer.
        attention_output = layers.MultiHeadAttention(
            num_heads=num_heads, key_dim=projection_dim, dropout=0.1
        )(x1, x1)
        # Skip connection 1.
        x2 = layers.Add()([attention_output, x])
        # Layer normalization 2.
        x3 = layers.LayerNormalization(epsilon=1e-6)(x2)
        # MLP.
        x3 = mlp(
            x3,
            hidden_units=[x.shape[-1] * 2, x.shape[-1]],
            dropout_rate=0.1,
        )
        # Skip connection 2.
        x = layers.Add()([x3, x2])

    return x

def mobilevit_block(x, num_blocks, projection_dim, strides=1):
    # Local projection with convolutions.
    local_features = conv_block(x, filters=projection_dim,
strides=strides)
    local_features = conv_block(
        local_features, filters=projection_dim, kernel_size=1,
```

```
strides=strides
    )

    # Unfold into patches and then pass through Transformers.
    num_patches = int((local_features.shape[1] *
local_features.shape[2]) / patch_size)
    non_overlapping_patches = layers.Reshape((patch_size, num_patches,
projection_dim))(
        local_features
    )
    global_features = transformer_block(
        non_overlapping_patches, num_blocks, projection_dim
    )

    # Fold into conv-like feature-maps.
    folded_feature_map = layers.Reshape((*local_features.shape[1:-1],
projection_dim))(
        global_features
    )

    # Apply point-wise conv -> concatenate with the input features.
    folded_feature_map = conv_block(
        folded_feature_map, filters=x.shape[-1], kernel_size=1,
strides=strides
    )
    local_global_features = layers.Concatenate(axis=-1)([x,
folded_feature_map])

    # Fuse the local and global features using a convoluion layer.
    local_global_features = conv_block(
        local_global_features, filters=projection_dim, strides=strides
    )

    return local_global_features

def create_mobilevit(num_classes=5):
    inputs = keras.Input((IMG_SIZE, IMG_SIZE, 3))
    x = layers.Rescaling(scale=1.0 / 255)(inputs)

    # Initial conv-stem -> MV2 block.
    x = conv_block(x, filters=16)
    x = inverted_residual_block(
        x, expanded_channels=16 * expansion_factor, output_channels=16
    )

    # Downsampling with MV2 block.
    x = inverted_residual_block(
        x, expanded_channels=16 * expansion_factor,
output_channels=24, strides=2
    )
```

```
    x = inverted_residual_block(
        x, expanded_channels=24 * expansion_factor, output_channels=24
    )
    x = inverted_residual_block(
        x, expanded_channels=24 * expansion_factor, output_channels=24
    )

    # First MV2 -> MobileViT block.
    x = inverted_residual_block(
        x, expanded_channels=24 * expansion_factor,
output_channels=48, strides=2
    )
    x = mobilevit_block(x, num_blocks=2, projection_dim=64)

    # Second MV2 -> MobileViT block.
    x = inverted_residual_block(
        x, expanded_channels=64 * expansion_factor,
output_channels=64, strides=2
    )
    x = mobilevit_block(x, num_blocks=4, projection_dim=80)

    # Third MV2 -> MobileViT block.
    x = inverted_residual_block(
        x, expanded_channels=80 * expansion_factor,
output_channels=80, strides=2
    )
    x = mobilevit_block(x, num_blocks=3, projection_dim=96)
    x = conv_block(x, filters=320, kernel_size=1, strides=1)

    # Classification head.
    x = layers.GlobalAvgPool2D()(x)
    outputs = layers.Dense(num_classes, activation="softmax")(x)

    return keras.Model(inputs, outputs)

model=create_mobilevit()
model.summary()

Model: "functional_2"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                 ┃ Output Shape            ┃        Param #
┃ Connected to                 ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
│ input_layer_2                │ (None, 256, 256, 3)     │              0
│ -                            │
│ (InputLayer)                 │                         │
│                              │
```

rescaling_2 (Rescaling) input_layer_2[0][0]	(None, 256, 256, 3)	0
conv2d_56 (Conv2D) rescaling_2[0][0]	(None, 128, 128, 16)	448
conv2d_57 (Conv2D) conv2d_56[0][0]	(None, 128, 128, 32)	512
batch_normalization_42 conv2d_57[0][0] (BatchNormalization)	(None, 128, 128, 32)	128
silu_28 (Silu) batch_normalization_4…	(None, 128, 128, 32)	0
depthwise_conv2d_14 silu_28[0][0] (DepthwiseConv2D)	(None, 128, 128, 32)	288
batch_normalization_43 depthwise_conv2d_14[0… (BatchNormalization)	(None, 128, 128, 32)	128
silu_29 (Silu) batch_normalization_4…	(None, 128, 128, 32)	0
conv2d_58 (Conv2D) silu_29[0][0]	(None, 128, 128, 16)	512
batch_normalization_44 conv2d_58[0][0] (BatchNormalization)	(None, 128, 128, 16)	64

add_42 (Add) batch_normalization_4… conv2d_56[0][0]	(None, 128, 128, 16)	0
conv2d_59 (Conv2D) add_42[0][0]	(None, 128, 128, 32)	512
batch_normalization_45 conv2d_59[0][0] (BatchNormalization)	(None, 128, 128, 32)	128
silu_30 (Silu) batch_normalization_4…	(None, 128, 128, 32)	0
zero_padding2d_8 silu_30[0][0] (ZeroPadding2D)	(None, 129, 129, 32)	0
depthwise_conv2d_15 zero_padding2d_8[0][0] (DepthwiseConv2D)	(None, 64, 64, 32)	288
batch_normalization_46 depthwise_conv2d_15[0… (BatchNormalization)	(None, 64, 64, 32)	128
silu_31 (Silu) batch_normalization_4…	(None, 64, 64, 32)	0
conv2d_60 (Conv2D) silu_31[0][0]	(None, 64, 64, 24)	768
batch_normalization_47 conv2d_60[0][0]	(None, 64, 64, 24)	96

(BatchNormalization)			
conv2d_61 (Conv2D)	(None, 64, 64, 48)	1,152	batch_normalization_4…
batch_normalization_48 (BatchNormalization)	(None, 64, 64, 48)	192	conv2d_61[0][0]
silu_32 (Silu)	(None, 64, 64, 48)	0	batch_normalization_4…
depthwise_conv2d_16 (DepthwiseConv2D)	(None, 64, 64, 48)	432	silu_32[0][0]
batch_normalization_49 (BatchNormalization)	(None, 64, 64, 48)	192	depthwise_conv2d_16[0…
silu_33 (Silu)	(None, 64, 64, 48)	0	batch_normalization_4…
conv2d_62 (Conv2D)	(None, 64, 64, 24)	1,152	silu_33[0][0]
batch_normalization_50 (BatchNormalization)	(None, 64, 64, 24)	96	conv2d_62[0][0]
add_43 (Add)	(None, 64, 64, 24)	0	batch_normalization_5… batch_normalization_4…

conv2d_63 (Conv2D) add_43[0][0]	(None, 64, 64, 48)	1,152
batch_normalization_51 conv2d_63[0][0] (BatchNormalization)	(None, 64, 64, 48)	192
silu_34 (Silu) batch_normalization_5…	(None, 64, 64, 48)	0
depthwise_conv2d_17 silu_34[0][0] (DepthwiseConv2D)	(None, 64, 64, 48)	432
batch_normalization_52 depthwise_conv2d_17[0… (BatchNormalization)	(None, 64, 64, 48)	192
silu_35 (Silu) batch_normalization_5…	(None, 64, 64, 48)	0
conv2d_64 (Conv2D) silu_35[0][0]	(None, 64, 64, 24)	1,152
batch_normalization_53 conv2d_64[0][0] (BatchNormalization)	(None, 64, 64, 24)	96
add_44 (Add) batch_normalization_5… add_43[0][0]	(None, 64, 64, 24)	0
conv2d_65 (Conv2D) add_44[0][0]	(None, 64, 64, 48)	1,152

batch_normalization_54 conv2d_65[0][0] (BatchNormalization)	(None, 64, 64, 48)	192
silu_36 (Silu) batch_normalization_5…	(None, 64, 64, 48)	0
zero_padding2d_9 silu_36[0][0] (ZeroPadding2D)	(None, 65, 65, 48)	0
depthwise_conv2d_18 zero_padding2d_9[0][0] (DepthwiseConv2D)	(None, 32, 32, 48)	432
batch_normalization_55 depthwise_conv2d_18[0… (BatchNormalization)	(None, 32, 32, 48)	192
silu_37 (Silu) batch_normalization_5…	(None, 32, 32, 48)	0
conv2d_66 (Conv2D) silu_37[0][0]	(None, 32, 32, 48)	2,304
batch_normalization_56 conv2d_66[0][0] (BatchNormalization)	(None, 32, 32, 48)	192
conv2d_67 (Conv2D) batch_normalization_5…	(None, 32, 32, 64)	27,712
conv2d_68 (Conv2D)	(None, 32, 32, 64)	4,160

```
│ conv2d_67[0][0]                │
├───────────────────────────────┼─────────────────────────┼────────────────┤
┤                                ┤
│ reshape_12 (Reshape)           │ (None, 16, 64, 64)      │              0 │
│ conv2d_68[0][0]                │
├───────────────────────────────┼─────────────────────────┼────────────────┤
┤                                ┤
│ layer_normalization_36         │ (None, 16, 64, 64)      │            128 │
│ reshape_12[0][0]               │
│ (LayerNormalization)           │                         │
│                                │
├───────────────────────────────┼─────────────────────────┼────────────────┤
┤                                ┤
│ multi_head_attention_18        │ (None, 16, 64, 64)      │         33,216 │
│ layer_normalization_3…         │
│ (MultiHeadAttention)           │                         │
│ layer_normalization_3…         │
├───────────────────────────────┼─────────────────────────┼────────────────┤
┤                                ┤
│ add_45 (Add)                   │ (None, 16, 64, 64)      │              0 │
│ multi_head_attention_…         │
│                                │                         │
│ reshape_12[0][0]               │
├───────────────────────────────┼─────────────────────────┼────────────────┤
┤                                ┤
│ layer_normalization_37         │ (None, 16, 64, 64)      │            128 │
│ add_45[0][0]                   │
│ (LayerNormalization)           │                         │
│                                │
├───────────────────────────────┼─────────────────────────┼────────────────┤
┤                                ┤
│ dense_38 (Dense)               │ (None, 16, 64, 128)     │          8,320 │
│ layer_normalization_3…         │
├───────────────────────────────┼─────────────────────────┼────────────────┤
┤                                ┤
│ dropout_55 (Dropout)           │ (None, 16, 64, 128)     │              0 │
│ dense_38[0][0]                 │
├───────────────────────────────┼─────────────────────────┼────────────────┤
┤                                ┤
│ dense_39 (Dense)               │ (None, 16, 64, 64)      │          8,256 │
│ dropout_55[0][0]               │
├───────────────────────────────┼─────────────────────────┼────────────────┤
┤                                ┤
│ dropout_56 (Dropout)           │ (None, 16, 64, 64)      │              0 │
│ dense_39[0][0]                 │
├───────────────────────────────┼─────────────────────────┼────────────────┤
┤                                ┤
│ add_46 (Add)                   │ (None, 16, 64, 64)      │              0 │
│ dropout_56[0][0],              │
```

add_45[0][0]		
layer_normalization_38 add_46[0][0] (LayerNormalization)	(None, 16, 64, 64)	128
multi_head_attention_19 layer_normalization_3… (MultiHeadAttention) layer_normalization_3…	(None, 16, 64, 64)	33,216
add_47 (Add) multi_head_attention_… add_46[0][0]	(None, 16, 64, 64)	0
layer_normalization_39 add_47[0][0] (LayerNormalization)	(None, 16, 64, 64)	128
dense_40 (Dense) layer_normalization_3…	(None, 16, 64, 128)	8,320
dropout_58 (Dropout) dense_40[0][0]	(None, 16, 64, 128)	0
dense_41 (Dense) dropout_58[0][0]	(None, 16, 64, 64)	8,256
dropout_59 (Dropout) dense_41[0][0]	(None, 16, 64, 64)	0
add_48 (Add) dropout_59[0][0], add_47[0][0]	(None, 16, 64, 64)	0

reshape_13 (Reshape) add_48[0][0]	(None, 32, 32, 64)	0
conv2d_69 (Conv2D) reshape_13[0][0]	(None, 32, 32, 48)	3,120
concatenate_6 batch_normalization_5… (Concatenate) conv2d_69[0][0]	(None, 32, 32, 96)	0
conv2d_70 (Conv2D) concatenate_6[0][0]	(None, 32, 32, 64)	55,360
conv2d_71 (Conv2D) conv2d_70[0][0]	(None, 32, 32, 128)	8,192
batch_normalization_57 conv2d_71[0][0] (BatchNormalization)	(None, 32, 32, 128)	512
silu_38 (Silu) batch_normalization_5…	(None, 32, 32, 128)	0
zero_padding2d_10 silu_38[0][0] (ZeroPadding2D)	(None, 33, 33, 128)	0
depthwise_conv2d_19 zero_padding2d_10[0][… (DepthwiseConv2D)	(None, 16, 16, 128)	1,152
batch_normalization_58 depthwise_conv2d_19[0… (BatchNormalization)	(None, 16, 16, 128)	512

silu_39 (Silu) batch_normalization_5…	(None, 16, 16, 128)	0
conv2d_72 (Conv2D) silu_39[0][0]	(None, 16, 16, 64)	8,192
batch_normalization_59 conv2d_72[0][0] (BatchNormalization)	(None, 16, 16, 64)	256
conv2d_73 (Conv2D) batch_normalization_5…	(None, 16, 16, 80)	46,160
conv2d_74 (Conv2D) conv2d_73[0][0]	(None, 16, 16, 80)	6,480
reshape_14 (Reshape) conv2d_74[0][0]	(None, 16, 16, 80)	0
layer_normalization_40 reshape_14[0][0] (LayerNormalization)	(None, 16, 16, 80)	160
multi_head_attention_20 layer_normalization_4… (MultiHeadAttention) layer_normalization_4…	(None, 16, 16, 80)	51,760
add_49 (Add) multi_head_attention_… reshape_14[0][0]	(None, 16, 16, 80)	0
layer_normalization_41 add_49[0][0] (LayerNormalization)	(None, 16, 16, 80)	160

dense_42 (Dense) layer_normalization_4…	(None, 16, 16, 160)	12,960
dropout_61 (Dropout) dense_42[0][0]	(None, 16, 16, 160)	0
dense_43 (Dense) dropout_61[0][0]	(None, 16, 16, 80)	12,880
dropout_62 (Dropout) dense_43[0][0]	(None, 16, 16, 80)	0
add_50 (Add) dropout_62[0][0], add_49[0][0]	(None, 16, 16, 80)	0
layer_normalization_42 add_50[0][0] (LayerNormalization)	(None, 16, 16, 80)	160
multi_head_attention_21 layer_normalization_4… (MultiHeadAttention) layer_normalization_4…	(None, 16, 16, 80)	51,760
add_51 (Add) multi_head_attention_… add_50[0][0]	(None, 16, 16, 80)	0
layer_normalization_43 add_51[0][0] (LayerNormalization)	(None, 16, 16, 80)	160
dense_44 (Dense)	(None, 16, 16, 160)	12,960

Layer (type)	Output Shape	Param #	Connected to
			layer_normalization_4…
dropout_64 (Dropout)	(None, 16, 16, 160)	0	dense_44[0][0]
dense_45 (Dense)	(None, 16, 16, 80)	12,880	dropout_64[0][0]
dropout_65 (Dropout)	(None, 16, 16, 80)	0	dense_45[0][0]
add_52 (Add)	(None, 16, 16, 80)	0	dropout_65[0][0], add_51[0][0]
layer_normalization_44 (LayerNormalization)	(None, 16, 16, 80)	160	add_52[0][0]
multi_head_attention_22 (MultiHeadAttention)	(None, 16, 16, 80)	51,760	layer_normalization_4…, layer_normalization_4…
add_53 (Add)	(None, 16, 16, 80)	0	multi_head_attention_…, add_52[0][0]
layer_normalization_45 (LayerNormalization)	(None, 16, 16, 80)	160	add_53[0][0]
dense_46 (Dense)	(None, 16, 16, 160)	12,960	layer_normalization_4…

dropout_67 (Dropout)	(None, 16, 16, 160)	0	dense_46[0][0]
dense_47 (Dense)	(None, 16, 16, 80)	12,880	dropout_67[0][0]
dropout_68 (Dropout)	(None, 16, 16, 80)	0	dense_47[0][0]
add_54 (Add)	(None, 16, 16, 80)	0	dropout_68[0][0], add_53[0][0]
layer_normalization_46 (LayerNormalization)	(None, 16, 16, 80)	160	add_54[0][0]
multi_head_attention_23 (MultiHeadAttention)	(None, 16, 16, 80)	51,760	layer_normalization_4…, layer_normalization_4…
add_55 (Add)	(None, 16, 16, 80)	0	multi_head_attention_…, add_54[0][0]
layer_normalization_47 (LayerNormalization)	(None, 16, 16, 80)	160	add_55[0][0]
dense_48 (Dense)	(None, 16, 16, 160)	12,960	layer_normalization_4…
dropout_70 (Dropout)	(None, 16, 16, 160)	0	dense_48[0][0]

dense_49 (Dense)	(None, 16, 16, 80)	12,880	dropout_70[0][0]
dropout_71 (Dropout)	(None, 16, 16, 80)	0	dense_49[0][0]
add_56 (Add)	(None, 16, 16, 80)	0	dropout_71[0][0], add_55[0][0]
reshape_15 (Reshape)	(None, 16, 16, 80)	0	add_56[0][0]
conv2d_75 (Conv2D)	(None, 16, 16, 64)	5,184	reshape_15[0][0]
concatenate_7 (Concatenate)	(None, 16, 16, 128)	0	batch_normalization_5…, conv2d_75[0][0]
conv2d_76 (Conv2D)	(None, 16, 16, 80)	92,240	concatenate_7[0][0]
conv2d_77 (Conv2D)	(None, 16, 16, 160)	12,800	conv2d_76[0][0]
batch_normalization_60 (BatchNormalization)	(None, 16, 16, 160)	640	conv2d_77[0][0]
silu_40 (Silu)	(None, 16, 16, 160)	0	batch_normalization_6…
zero_padding2d_11	(None, 17, 17, 160)	0	silu_40[0][0]

(ZeroPadding2D)		
depthwise_conv2d_20 zero_padding2d_11[0][… (DepthwiseConv2D)	(None, 8, 8, 160)	1,440
batch_normalization_61 depthwise_conv2d_20[0… (BatchNormalization)	(None, 8, 8, 160)	640
silu_41 (Silu) batch_normalization_6…	(None, 8, 8, 160)	0
conv2d_78 (Conv2D) silu_41[0][0]	(None, 8, 8, 80)	12,800
batch_normalization_62 conv2d_78[0][0] (BatchNormalization)	(None, 8, 8, 80)	320
conv2d_79 (Conv2D) batch_normalization_6…	(None, 8, 8, 96)	69,216
conv2d_80 (Conv2D) conv2d_79[0][0]	(None, 8, 8, 96)	9,312
reshape_16 (Reshape) conv2d_80[0][0]	(None, 16, 4, 96)	0
layer_normalization_48 reshape_16[0][0] (LayerNormalization)	(None, 16, 4, 96)	192
multi_head_attention_24	(None, 16, 4, 96)	74,400

(MultiHeadAttention)			layer_normalization_4… layer_normalization_4…
add_57 (Add)	(None, 16, 4, 96)	0	multi_head_attention_… reshape_16[0][0]
layer_normalization_49 (LayerNormalization)	(None, 16, 4, 96)	192	add_57[0][0]
dense_50 (Dense)	(None, 16, 4, 192)	18,624	layer_normalization_4…
dropout_73 (Dropout)	(None, 16, 4, 192)	0	dense_50[0][0]
dense_51 (Dense)	(None, 16, 4, 96)	18,528	dropout_73[0][0]
dropout_74 (Dropout)	(None, 16, 4, 96)	0	dense_51[0][0]
add_58 (Add)	(None, 16, 4, 96)	0	dropout_74[0][0], add_57[0][0]
layer_normalization_50 (LayerNormalization)	(None, 16, 4, 96)	192	add_58[0][0]
multi_head_attention_25 (MultiHeadAttention)	(None, 16, 4, 96)	74,400	layer_normalization_5… layer_normalization_5…

add_59 (Add) multi_head_attention_… add_58[0][0]	(None, 16, 4, 96)	0
layer_normalization_51 add_59[0][0] (LayerNormalization)	(None, 16, 4, 96)	192
dense_52 (Dense) layer_normalization_5…	(None, 16, 4, 192)	18,624
dropout_76 (Dropout) dense_52[0][0]	(None, 16, 4, 192)	0
dense_53 (Dense) dropout_76[0][0]	(None, 16, 4, 96)	18,528
dropout_77 (Dropout) dense_53[0][0]	(None, 16, 4, 96)	0
add_60 (Add) dropout_77[0][0], add_59[0][0]	(None, 16, 4, 96)	0
layer_normalization_52 add_60[0][0] (LayerNormalization)	(None, 16, 4, 96)	192
multi_head_attention_26 layer_normalization_5… (MultiHeadAttention) layer_normalization_5…	(None, 16, 4, 96)	74,400
add_61 (Add)	(None, 16, 4, 96)	0

multi_head_attention_… add_60[0][0]		
layer_normalization_53 add_61[0][0] (LayerNormalization)	(None, 16, 4, 96)	192
dense_54 (Dense) layer_normalization_5…	(None, 16, 4, 192)	18,624
dropout_79 (Dropout) dense_54[0][0]	(None, 16, 4, 192)	0
dense_55 (Dense) dropout_79[0][0]	(None, 16, 4, 96)	18,528
dropout_80 (Dropout) dense_55[0][0]	(None, 16, 4, 96)	0
add_62 (Add) dropout_80[0][0], add_61[0][0]	(None, 16, 4, 96)	0
reshape_17 (Reshape) add_62[0][0]	(None, 8, 8, 96)	0
conv2d_81 (Conv2D) reshape_17[0][0]	(None, 8, 8, 80)	7,760
concatenate_8 batch_normalization_6… (Concatenate) conv2d_81[0][0]	(None, 8, 8, 160)	0
conv2d_82 (Conv2D) concatenate_8[0][0]	(None, 8, 8, 96)	138,336

```
├──────────────────────────────┼───────────────────────────┼─────────────────
─┤
 conv2d_83 (Conv2D)            │ (None, 8, 8, 320)         │         31,040
 conv2d_82[0][0]               │
├──────────────────────────────┼───────────────────────────┼─────────────────
─┤
 global_average_pooling2d…     │ (None, 320)               │              0
 conv2d_83[0][0]               │
 (GlobalAveragePooling2D)      │                           │
                               │
├──────────────────────────────┼───────────────────────────┼─────────────────
─┤
 dense_56 (Dense)              │ (None, 5)                 │          1,605
 global_average_poolin…        │
└──────────────────────────────┴───────────────────────────┴─────────────────
─┘
```

```
 Total params: 1,307,621 (4.99 MB)

 Trainable params: 1,305,077 (4.98 MB)

 Non-trainable params: 2,544 (9.94 KB)
```

```python
class_weights = {0:1.1347,
                 1:0.9095,
                 2:0.9811}

from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3),

loss='sparse_categorical_crossentropy',
                                                 metrics=['accuracy'])

from tensorflow.keras.callbacks import EarlyStopping

 filepath = "/kaggle/working/My_model.keras"
 checkpoint = ModelCheckpoint(filepath, monitor="val_accuracy",
                              verbose=1, save_best_only=True,
                              mode='max')
 reduce_lr = ReduceLROnPlateau(monitor='val_accuracy',factor=1e-1,
patience=5, verbose=1)
 early_stop = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)
 callbacks = [checkpoint, reduce_lr,early_stop]

import tensorflow as tf
print(tf.config.list_physical_devices('GPU'))
```

```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU'),
PhysicalDevice(name='/physical_device:GPU:1', device_type='GPU')]
```

```python
fine_tune_epochs = 100

with tf.device('/GPU:0'):
             history= model.fit(X_train, y_train,
                        validation_data=[X_test, y_test],
                        class_weight=class_weights,
                        callbacks=callbacks,
                        epochs=fine_tune_epochs,batch_size=32)
```

```
Epoch 1/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 601ms/step - accuracy: 0.6472 - loss:
0.8473
Epoch 1: val_accuracy improved from -inf to 0.23494, saving model
to /kaggle/working/My_model.keras
179/179 ━━━━━━━━━━━━━━━━━━━━ 193s 665ms/step - accuracy: 0.6477 -
loss: 0.8463 - val_accuracy: 0.2349 - val_loss: 2.3662 -
learning_rate: 0.0010
Epoch 2/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 257ms/step - accuracy: 0.8198 - loss:
0.4691
Epoch 2: val_accuracy did not improve from 0.23494
179/179 ━━━━━━━━━━━━━━━━━━━━ 49s 272ms/step - accuracy: 0.8199 - loss:
0.4688 - val_accuracy: 0.2288 - val_loss: 5.4711 - learning_rate:
0.0010
Epoch 3/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 238ms/step - accuracy: 0.8897 - loss:
0.3064
Epoch 3: val_accuracy improved from 0.23494 to 0.36079, saving model
to /kaggle/working/My_model.keras
179/179 ━━━━━━━━━━━━━━━━━━━━ 46s 257ms/step - accuracy: 0.8898 - loss:
0.3063 - val_accuracy: 0.3608 - val_loss: 2.9929 - learning_rate:
0.0010
Epoch 4/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 244ms/step - accuracy: 0.9285 - loss:
0.2138
Epoch 4: val_accuracy improved from 0.36079 to 0.69184, saving model
to /kaggle/working/My_model.keras
179/179 ━━━━━━━━━━━━━━━━━━━━ 47s 263ms/step - accuracy: 0.9285 - loss:
0.2139 - val_accuracy: 0.6918 - val_loss: 0.9211 - learning_rate:
0.0010
Epoch 5/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 245ms/step - accuracy: 0.9466 - loss:
0.1530
Epoch 5: val_accuracy improved from 0.69184 to 0.75210, saving model
to /kaggle/working/My_model.keras
179/179 ━━━━━━━━━━━━━━━━━━━━ 47s 264ms/step - accuracy: 0.9466 - loss:
0.1531 - val_accuracy: 0.7521 - val_loss: 0.8482 - learning_rate:
0.0010
Epoch 6/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 242ms/step - accuracy: 0.9588 - loss:
```

```
0.1170
Epoch 6: val_accuracy improved from 0.75210 to 0.82838, saving model
to /kaggle/working/My_model.keras
179/179 ━━━━━━━━━━━━━━━━━━━━ 47s 261ms/step - accuracy: 0.9588 - loss:
0.1172 - val_accuracy: 0.8284 - val_loss: 0.5292 - learning_rate:
0.0010
Epoch 7/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 245ms/step - accuracy: 0.9648 - loss:
0.1008
Epoch 7: val_accuracy did not improve from 0.82838
179/179 ━━━━━━━━━━━━━━━━━━━━ 46s 260ms/step - accuracy: 0.9648 - loss:
0.1008 - val_accuracy: 0.6468 - val_loss: 1.8150 - learning_rate:
0.0010
Epoch 8/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 243ms/step - accuracy: 0.9624 - loss:
0.1028
Epoch 8: val_accuracy improved from 0.82838 to 0.90694, saving model
to /kaggle/working/My_model.keras
179/179 ━━━━━━━━━━━━━━━━━━━━ 47s 262ms/step - accuracy: 0.9624 - loss:
0.1027 - val_accuracy: 0.9069 - val_loss: 0.3123 - learning_rate:
0.0010
Epoch 9/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 244ms/step - accuracy: 0.9804 - loss:
0.0527
Epoch 9: val_accuracy improved from 0.90694 to 0.93593, saving model
to /kaggle/working/My_model.keras
179/179 ━━━━━━━━━━━━━━━━━━━━ 47s 263ms/step - accuracy: 0.9804 - loss:
0.0528 - val_accuracy: 0.9359 - val_loss: 0.2125 - learning_rate:
0.0010
Epoch 10/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 243ms/step - accuracy: 0.9770 - loss:
0.0691
Epoch 10: val_accuracy did not improve from 0.93593
179/179 ━━━━━━━━━━━━━━━━━━━━ 46s 259ms/step - accuracy: 0.9769 - loss:
0.0692 - val_accuracy: 0.6880 - val_loss: 1.9316 - learning_rate:
0.0010
Epoch 11/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 244ms/step - accuracy: 0.9757 - loss:
0.0591
Epoch 11: val_accuracy did not improve from 0.93593
179/179 ━━━━━━━━━━━━━━━━━━━━ 46s 259ms/step - accuracy: 0.9757 - loss:
0.0591 - val_accuracy: 0.8734 - val_loss: 0.4793 - learning_rate:
0.0010
Epoch 12/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 242ms/step - accuracy: 0.9866 - loss:
0.0381
Epoch 12: val_accuracy did not improve from 0.93593
179/179 ━━━━━━━━━━━━━━━━━━━━ 46s 257ms/step - accuracy: 0.9865 - loss:
0.0382 - val_accuracy: 0.8993 - val_loss: 0.3947 - learning_rate:
```

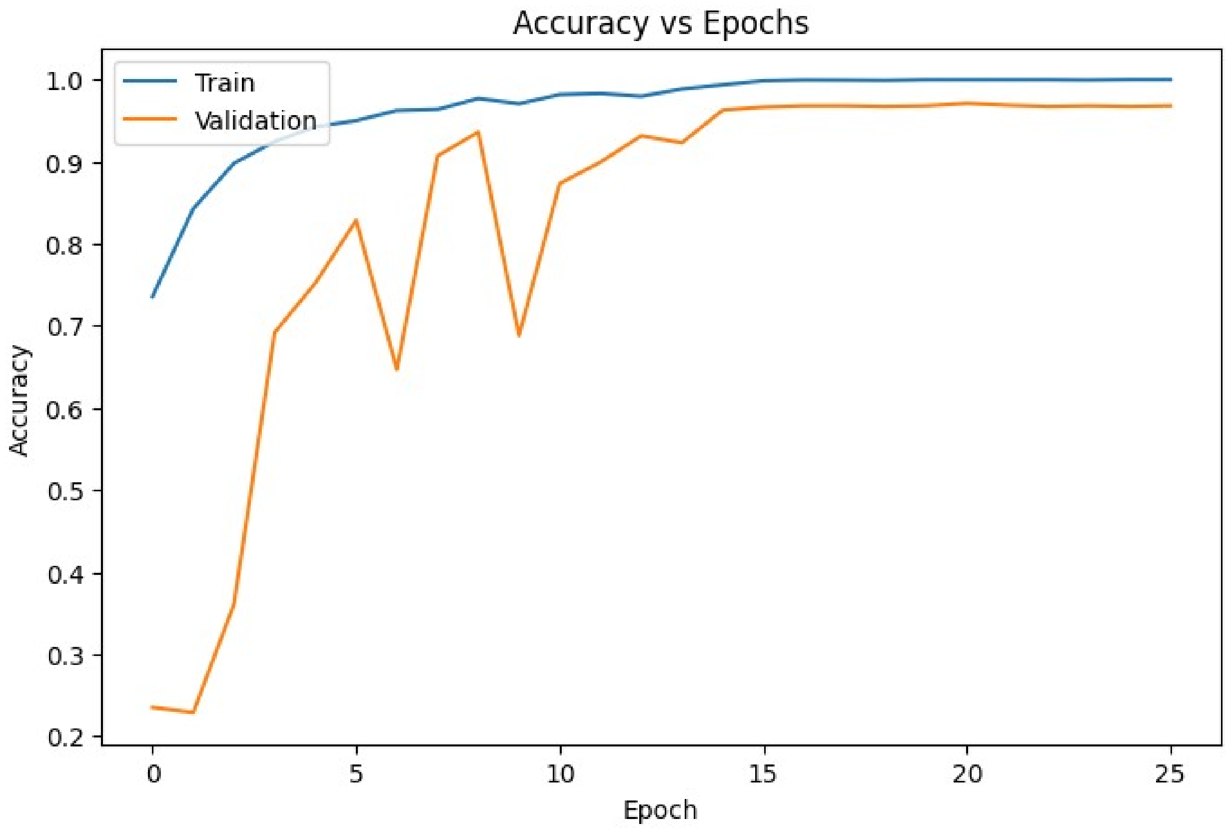
```
0.0010
Epoch 13/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 242ms/step - accuracy: 0.9811 - loss:
0.0485
Epoch 13: val_accuracy did not improve from 0.93593
179/179 ━━━━━━━━━━━━━━━━━━━━ 46s 258ms/step - accuracy: 0.9811 - loss:
0.0485 - val_accuracy: 0.9314 - val_loss: 0.2109 - learning_rate:
0.0010
Epoch 14/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 243ms/step - accuracy: 0.9884 - loss:
0.0320
Epoch 14: val_accuracy did not improve from 0.93593

Epoch 14: ReduceLROnPlateau reducing learning rate to
0.00010000000474974513.
179/179 ━━━━━━━━━━━━━━━━━━━━ 46s 258ms/step - accuracy: 0.9884 - loss:
0.0320 - val_accuracy: 0.9230 - val_loss: 0.3387 - learning_rate:
0.0010
Epoch 15/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 243ms/step - accuracy: 0.9926 - loss:
0.0212
Epoch 15: val_accuracy improved from 0.93593 to 0.96262, saving model
to /kaggle/working/My_model.keras
179/179 ━━━━━━━━━━━━━━━━━━━━ 47s 262ms/step - accuracy: 0.9926 - loss:
0.0211 - val_accuracy: 0.9626 - val_loss: 0.1365 - learning_rate:
1.0000e-04
Epoch 16/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 244ms/step - accuracy: 0.9987 - loss:
0.0061
Epoch 16: val_accuracy improved from 0.96262 to 0.96644, saving model
to /kaggle/working/My_model.keras
179/179 ━━━━━━━━━━━━━━━━━━━━ 47s 263ms/step - accuracy: 0.9987 - loss:
0.0061 - val_accuracy: 0.9664 - val_loss: 0.1260 - learning_rate:
1.0000e-04
Epoch 17/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 243ms/step - accuracy: 0.9991 - loss:
0.0038
Epoch 17: val_accuracy improved from 0.96644 to 0.96796, saving model
to /kaggle/working/My_model.keras
179/179 ━━━━━━━━━━━━━━━━━━━━ 47s 262ms/step - accuracy: 0.9991 - loss:
0.0038 - val_accuracy: 0.9680 - val_loss: 0.1378 - learning_rate:
1.0000e-04
Epoch 18/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 243ms/step - accuracy: 0.9997 - loss:
0.0033
Epoch 18: val_accuracy did not improve from 0.96796
179/179 ━━━━━━━━━━━━━━━━━━━━ 46s 258ms/step - accuracy: 0.9997 - loss:
0.0033 - val_accuracy: 0.9680 - val_loss: 0.1281 - learning_rate:
1.0000e-04
```

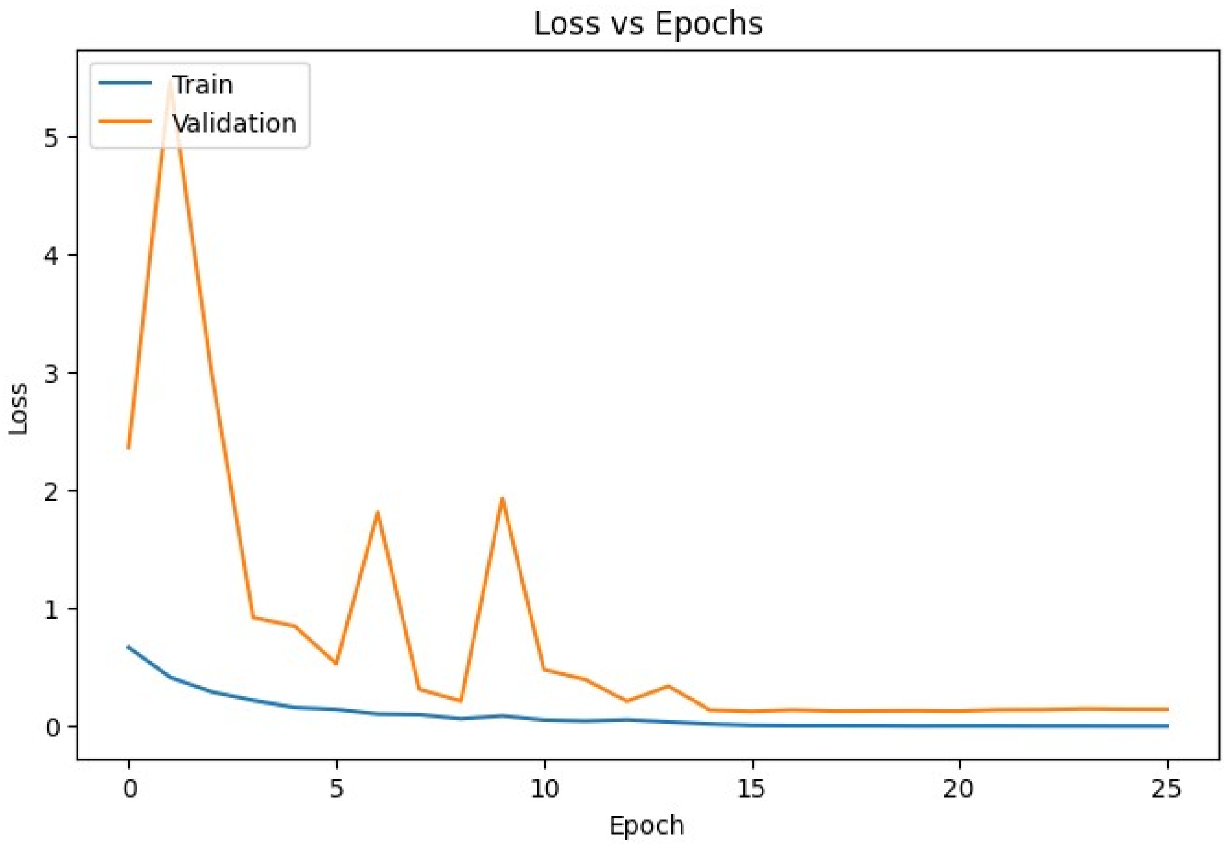
```
Epoch 19/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 244ms/step - accuracy: 0.9989 - loss:
0.0040
Epoch 19: val_accuracy did not improve from 0.96796
179/179 ━━━━━━━━━━━━━━━━━━━━ 46s 259ms/step - accuracy: 0.9989 - loss:
0.0040 - val_accuracy: 0.9672 - val_loss: 0.1298 - learning_rate:
1.0000e-04
Epoch 20/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 244ms/step - accuracy: 1.0000 - loss:
0.0015
Epoch 20: val_accuracy did not improve from 0.96796
179/179 ━━━━━━━━━━━━━━━━━━━━ 46s 259ms/step - accuracy: 1.0000 - loss:
0.0015 - val_accuracy: 0.9680 - val_loss: 0.1309 - learning_rate:
1.0000e-04
Epoch 21/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 244ms/step - accuracy: 1.0000 - loss:
0.0019
Epoch 21: val_accuracy improved from 0.96796 to 0.97101, saving model
to /kaggle/working/My_model.keras
179/179 ━━━━━━━━━━━━━━━━━━━━ 47s 262ms/step - accuracy: 1.0000 - loss:
0.0019 - val_accuracy: 0.9710 - val_loss: 0.1280 - learning_rate:
1.0000e-04
Epoch 22/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 243ms/step - accuracy: 0.9998 - loss:
0.0023
Epoch 22: val_accuracy did not improve from 0.97101
179/179 ━━━━━━━━━━━━━━━━━━━━ 46s 258ms/step - accuracy: 0.9998 - loss:
0.0023 - val_accuracy: 0.9687 - val_loss: 0.1392 - learning_rate:
1.0000e-04
Epoch 23/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 244ms/step - accuracy: 0.9998 - loss:
0.0016
Epoch 23: val_accuracy did not improve from 0.97101
179/179 ━━━━━━━━━━━━━━━━━━━━ 46s 259ms/step - accuracy: 0.9998 - loss:
0.0016 - val_accuracy: 0.9672 - val_loss: 0.1404 - learning_rate:
1.0000e-04
Epoch 24/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 244ms/step - accuracy: 0.9998 - loss:
0.0011
Epoch 24: val_accuracy did not improve from 0.97101
179/179 ━━━━━━━━━━━━━━━━━━━━ 46s 260ms/step - accuracy: 0.9998 - loss:
0.0011 - val_accuracy: 0.9680 - val_loss: 0.1476 - learning_rate:
1.0000e-04
Epoch 25/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 244ms/step - accuracy: 1.0000 - loss:
9.2886e-04
Epoch 25: val_accuracy did not improve from 0.97101
179/179 ━━━━━━━━━━━━━━━━━━━━ 46s 259ms/step - accuracy: 1.0000 - loss:
9.2828e-04 - val_accuracy: 0.9672 - val_loss: 0.1441 - learning_rate:
```

```
1.0000e-04
Epoch 26/100
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 244ms/step - accuracy: 1.0000 - loss:
4.3905e-04
Epoch 26: val_accuracy did not improve from 0.97101

Epoch 26: ReduceLROnPlateau reducing learning rate to
1.0000000474974514e-05.
179/179 ━━━━━━━━━━━━━━━━━━━━ 46s 259ms/step - accuracy: 1.0000 - loss:
4.3965e-04 - val_accuracy: 0.9680 - val_loss: 0.1428 - learning_rate:
1.0000e-04
```

```python
print(history.history.keys())
#  "Accuracy"
plt.figure(figsize=(8, 5))
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Accuracy vs Epochs')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
# "Loss"
plt.figure(figsize=(8, 5))
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Loss vs Epochs')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

```
dict_keys(['accuracy', 'loss', 'val_accuracy', 'val_loss',
'learning_rate'])
```


Accuracy vs Epochs
Train
Validation
Accuracy
Epoch
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
5
10
15
20
25




```
re_model =
tf.keras.models.load_model('/kaggle/working/My_model.keras')

loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {accuracy:.4f}")
```

```
41/41 ━━━━━━━━━━━━━━━━━━━━ 3s 65ms/step - accuracy: 0.9465 - loss:
0.2078
Test Accuracy: 0.9664
```

```
y_pred = re_model.predict(X_test)
```

```
41/41 ━━━━━━━━━━━━━━━━━━━━ 12s 183ms/step
```

```
y_true = y_test

y_pred = np.argmax(y_pred, axis=1)
y_pred
```

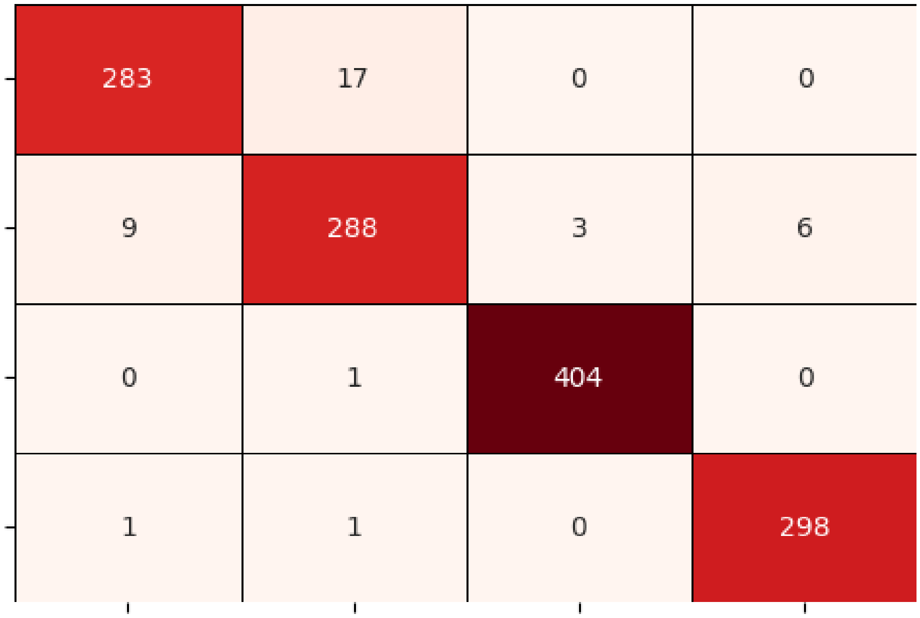
```
array([0, 0, 0, ..., 3, 3, 3])
```

```
print(classification_report(y_true, y_pred))
```

```
              precision    recall  f1-score   support
```

```
           0       0.97      0.94      0.95       300
           1       0.94      0.94      0.94       306
           2       0.99      1.00      1.00       405
           3       0.98      0.99      0.99       300

    accuracy                           0.97      1311
   macro avg       0.97      0.97      0.97      1311
weighted avg       0.97      0.97      0.97      1311
```

```python
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Compute confusion matrix
cm = confusion_matrix(y_true, y_pred)

# Plot heatmap with red-black color scheme
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Reds", cbar=False,
linewidths=0.5, linecolor='black')

# Set black background
plt.gca().set_facecolor('black')
plt.xlabel("Predicted", color="white")
plt.ylabel("Actual", color="white")
plt.title("Confusion Matrix", color="white")

# Change tick colors to white
plt.xticks(color="white")
plt.yticks(color="white")

plt.show()
```


283
17
0
0
9
288
3
6
0
1
404
0
1
1
0
298