

```

*****
# ResNet50 model
*****
def build_model_resnet50(input_shape, num_classes, learning_rate=0.0001):
    base_model = ResNet50(weights='imagenet', include_top=False, input_shape=input_shape)
    base_model.trainable = False #freeze the base model

    model = Sequential([
        base_model,
        GlobalAveragePooling2D(),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(num_classes, activation='softmax')
    ])

    model.compile(optimizer=Adam(learning_rate=learning_rate),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    return model

```

```

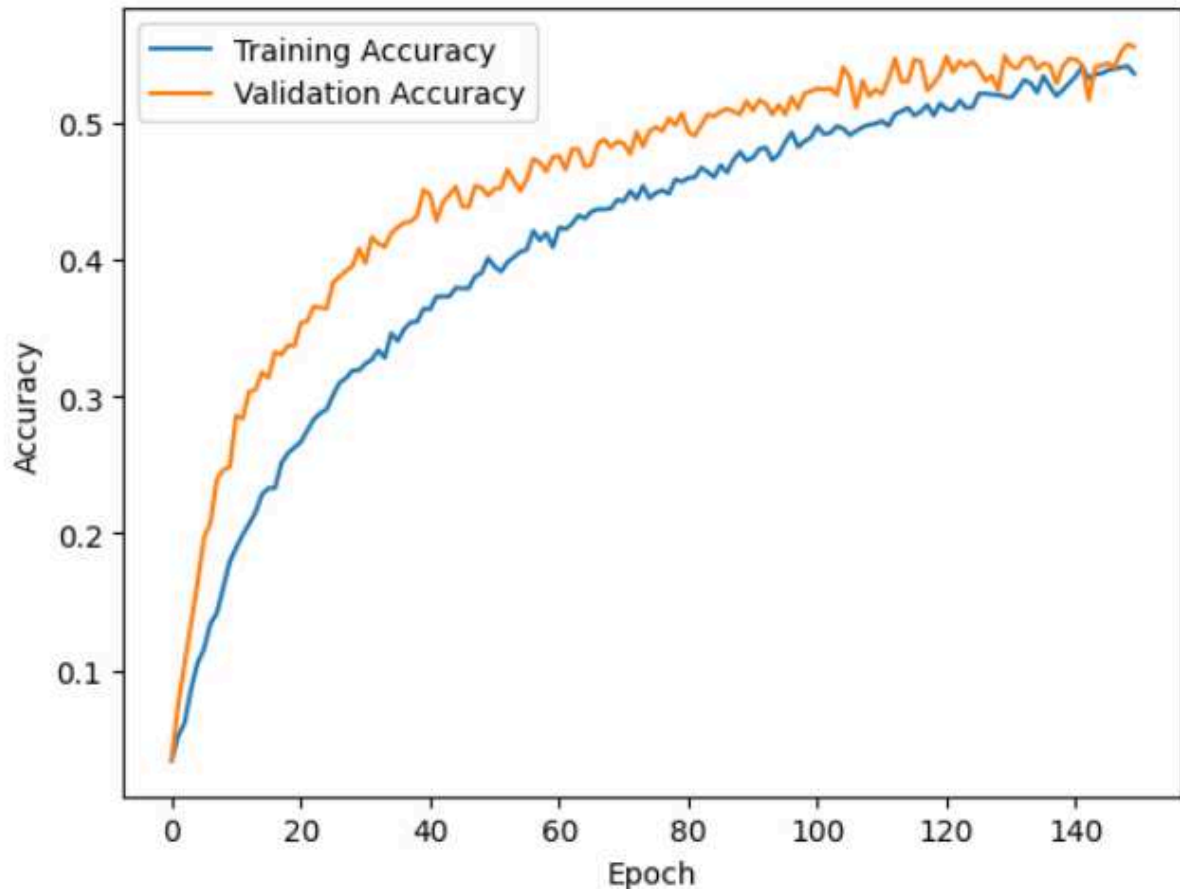
history = restNet50_model.fit(improved_train_gen, epochs=150, validation_data=improved_val_gen)
print("Done")

```

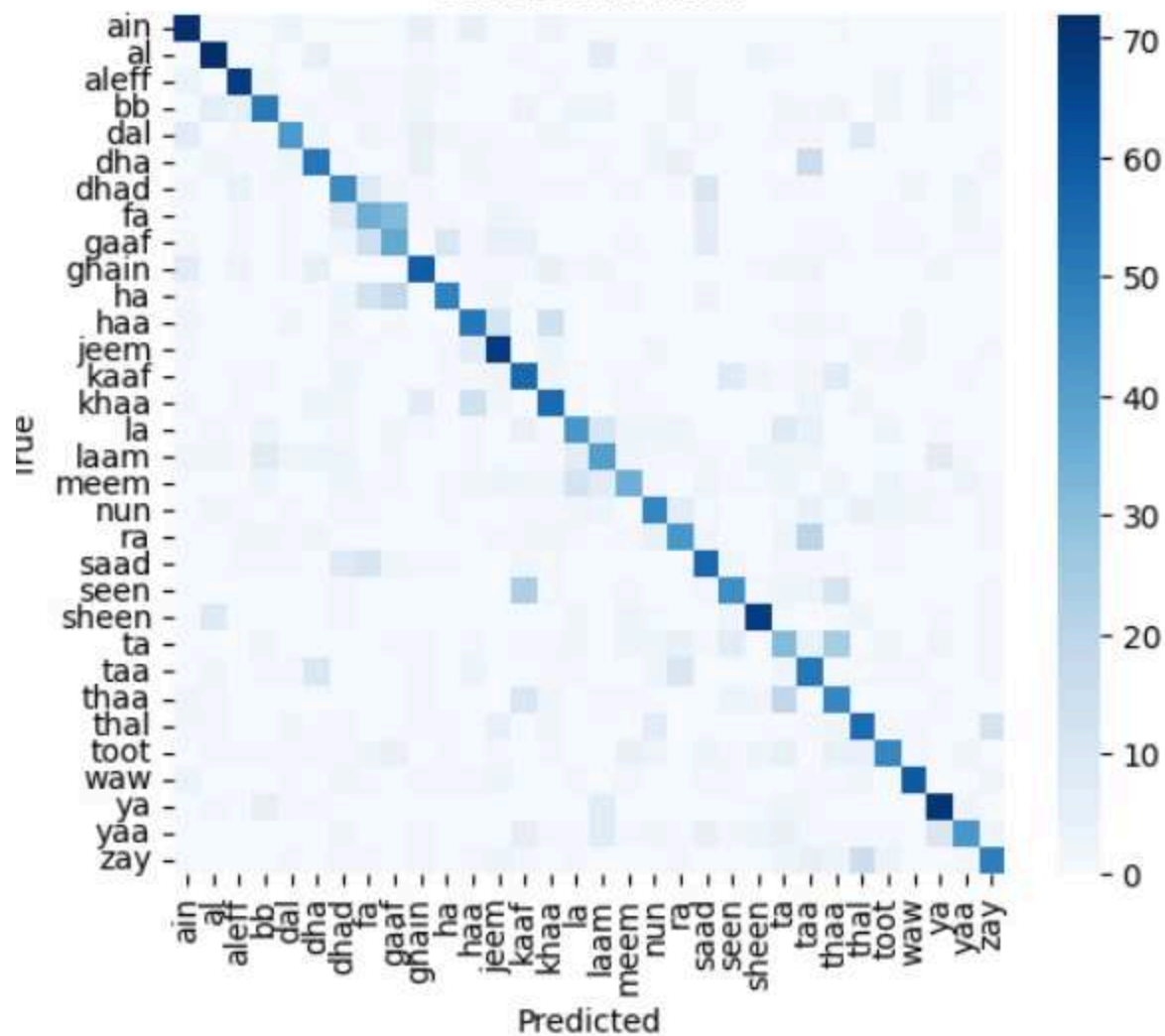
```

Epoch 150/150
620/620 — 598s 965ms/step - accuracy: 0.5353 - loss: 1.4353 - val_accuracy: 0.5549 - val_loss: 1.4505
Done

```



Confusion matrix for the Arabic Vowel Segmentation dataset. The matrix shows the relationship between predicted and true labels. The diagonal elements are dark blue, indicating high accuracy. The color scale on the right ranges from 0 (light blue) to 70 (dark blue).



```

*****
#resnet50 with 3 dense layers
*****
def build_model_resnet50_with_3_dense(input_shape, num_classes, learning_rate = 0.0001):
    base_model = ResNet50(weights='imagenet', include_top=False, input_shape=input_shape)
    base_model.trainable = False #freeze the base model

    model = Sequential([
        base_model,
        GlobalAveragePooling2D(),
        Dense(512, activation='relu'),
        Dropout(0.3),
        Dense(256, activation='relu'),
        Dropout(0.3),
        Dense(num_classes, activation='softmax')
    ])

    model.compile(optimizer=Adam(learning_rate=learning_rate),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

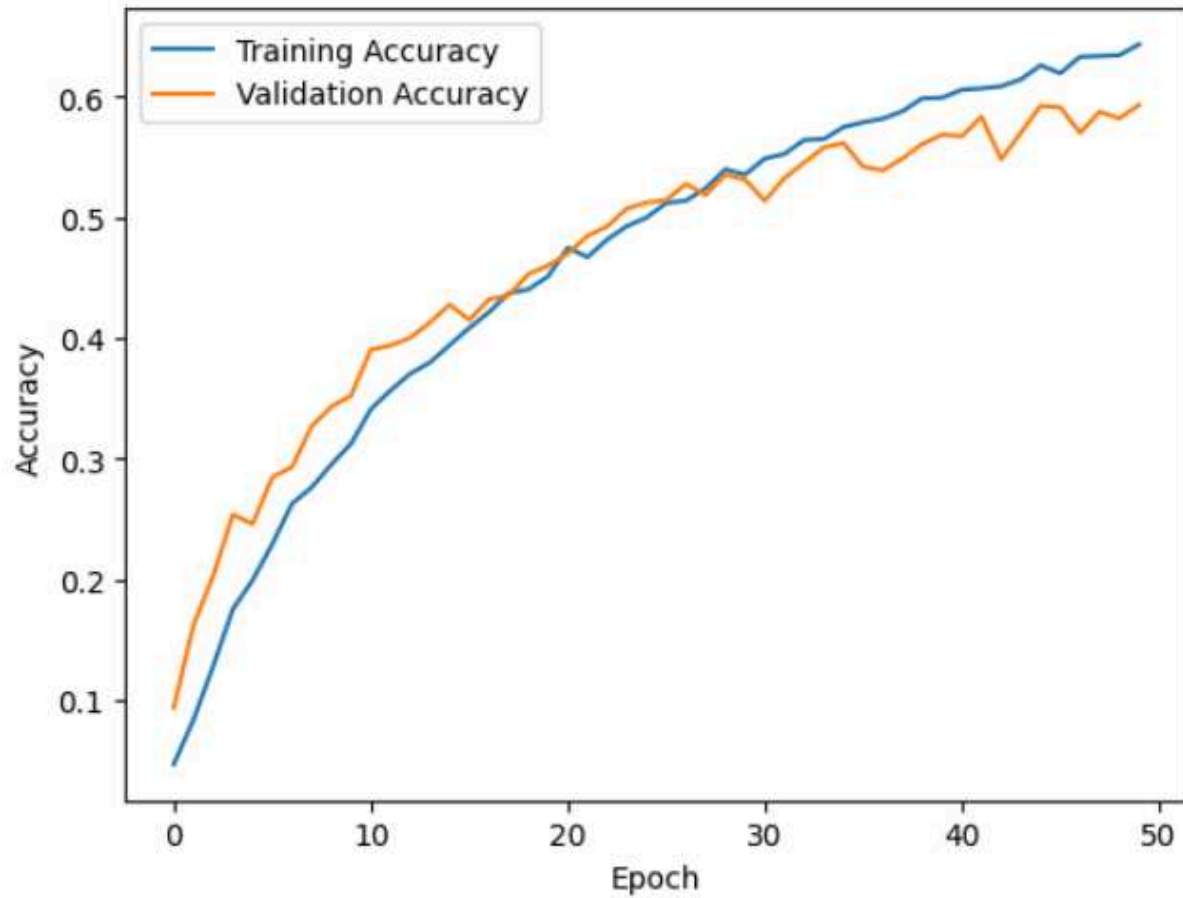
    return model

```

```

620/620 498s 804ms/step - accuracy: 0.6399 - loss: 1.1008 - val_accuracy: 0.5930 - val_loss: 1.3167
Done

```



```

*****
# Testing with Learning Rate Scheduler, another 50 epochs
*****

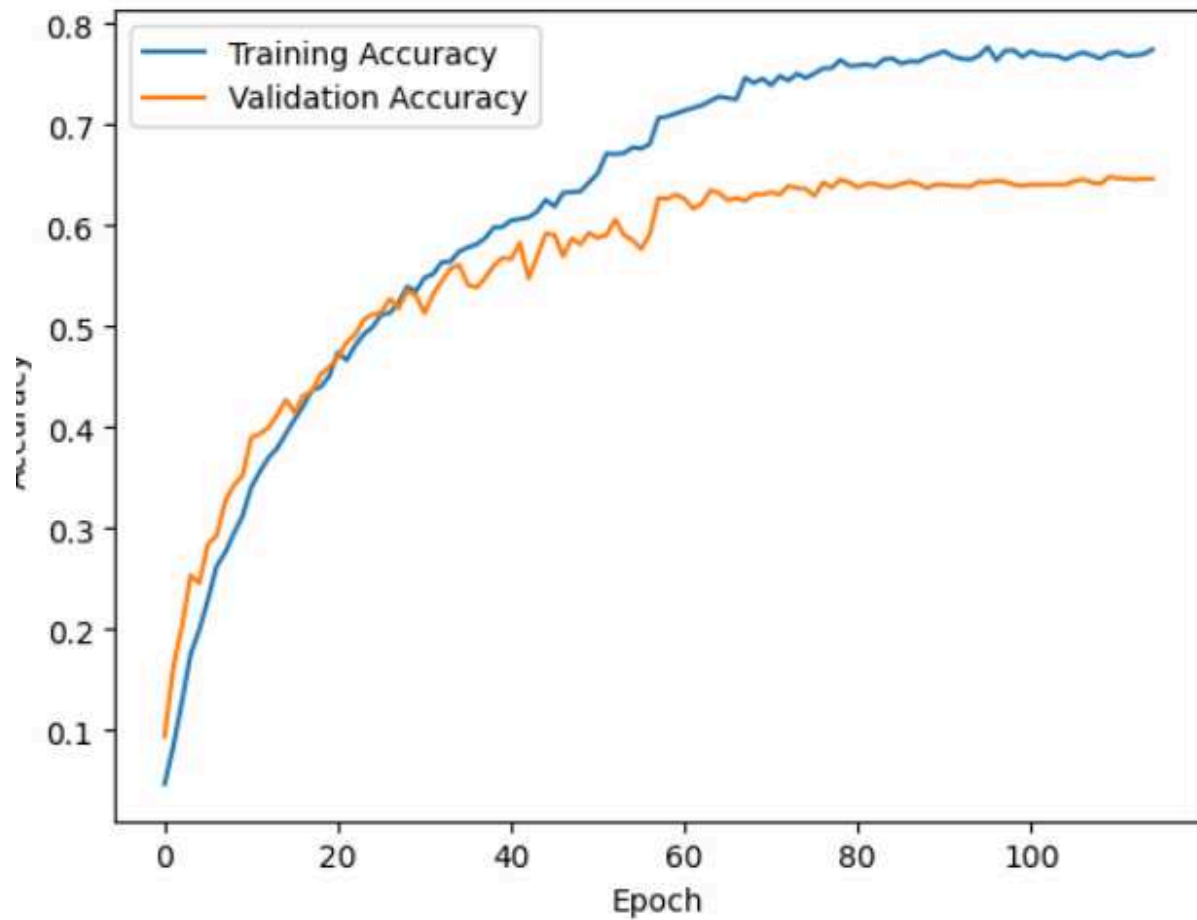
from tensorflow.keras.callbacks import ReduceLROnPlateau

# Define the learning rate scheduler
lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3, verbose=1, min_lr=1e-6)

history4 = new_resnet50_model.fit(improved_train_gen, epochs=50, validation_data=improved_val_gen, callbacks=[lr_scheduler])
acc = history4.history['accuracy']
val_acc = history4.history['val_accuracy']
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

```

Epoch 50/50
620/620 700s 1s/step - accuracy: 0.7733 - loss: 0.6987 - val_accuracy: 0.6394 - val_loss: 1.1416 - learning_rate: 1.5625e-06

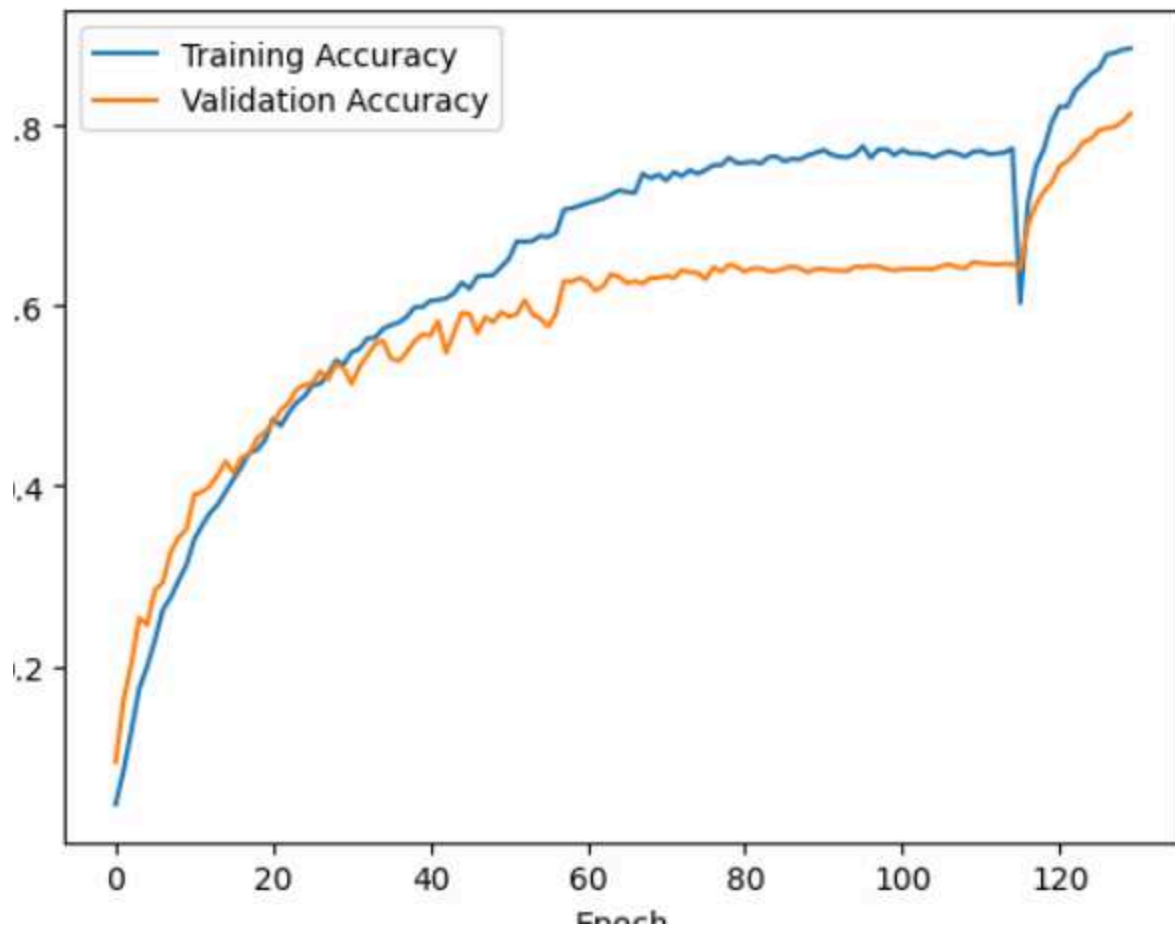


```
#almost no gain
#unfreeze and train on Lower LR

base_model = new_resnet50_model.layers[0]
# Unfreeze top layers of the base model
base_model.trainable = True
for layer in base_model.layers[:-50]: # Freeze all but last 50 layers
    layer.trainable = False

# Recompile with a smaller Learning rate
new_resnet50_model.compile(optimizer=Adam(learning_rate=0.00001), # Lower LR for fine-tuning
    loss='categorical_crossentropy',
    metrics=['accuracy'])
```

```
Epoch 15/15
620/620 857s 1s/step - accuracy: 0.8871 - loss: 0.3281 - val_accuracy: 0.8134 - val_loss: 0.6232 - learning_rate: 1.0000e-05
```



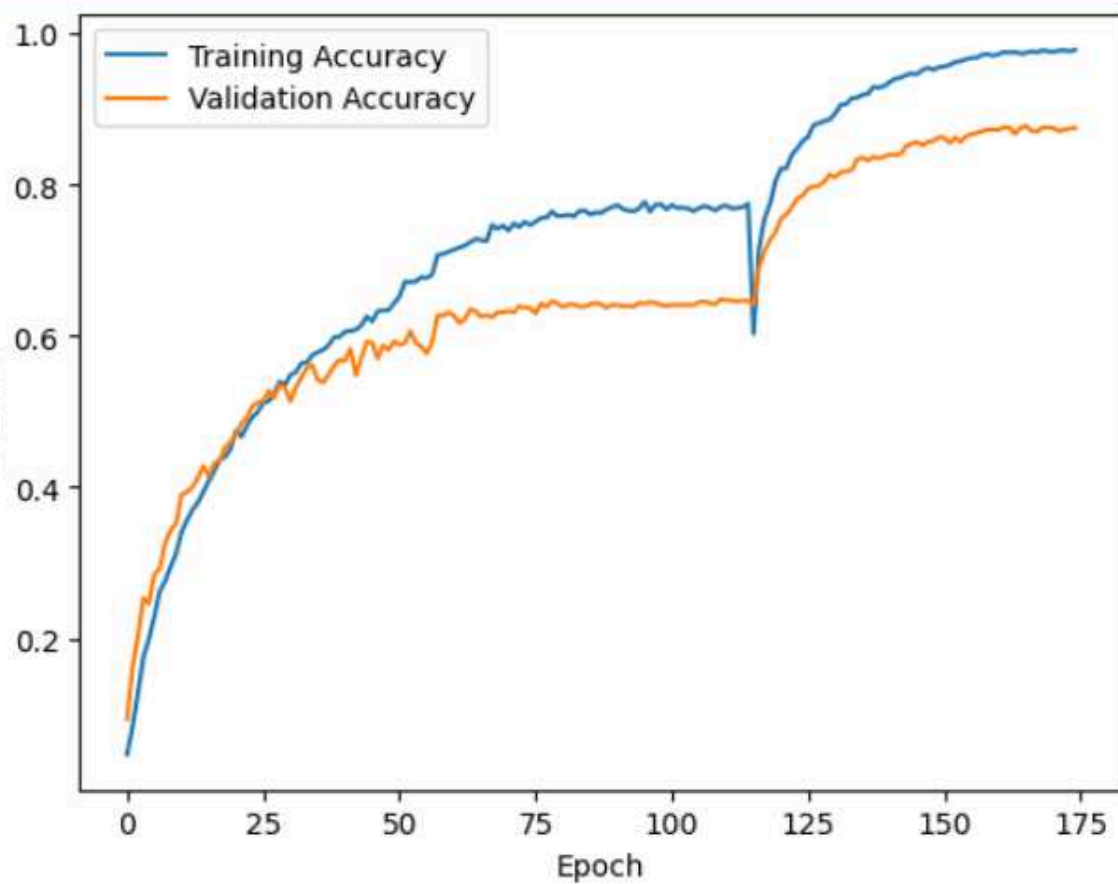
#adding call back to save best model

```
from tensorflow.keras.callbacks import ModelCheckpoint
```

```
checkpoint_cb = ModelCheckpoint(
    filepath='best_model.h5',
    monitor='val_loss',
    save_best_only=True,
    mode='min',
    verbose=1
)
```

```
history8 = new_resnet50_model.fit(improved_train_gen, epochs=15, validation_data=improved_val_gen, callbacks=[lr_scheduler, checkpoint_cb])
```

620/620 — 3923s 6s/step - accuracy: 0.9746 - loss: 0.0785 - val_accuracy: 0.8775 - val_loss: 0.4502 - learning_rate: 2.5000e-06



180/180 ————— **164s** 896ms/step

Accuracy: 0.8974537844436693

Precision: 0.8999788239166187

Recall: 0.8974537844436693

F1 Score: 0.8978338469210082

classification Report:

ain	0.97	0.97	0.97	90
al	0.94	0.93	0.94	91
aleff	0.96	0.91	0.94	90
bb	0.91	0.91	0.91	90
dal	0.93	0.84	0.88	81
dha	0.95	0.98	0.96	91
dhad	0.95	0.94	0.95	88
fa	0.79	0.75	0.77	91
gaaf	0.70	0.79	0.74	90
ghain	0.99	0.93	0.96	91
ha	0.93	0.85	0.89	91
haa	0.85	0.89	0.87	90
jeem	0.86	0.90	0.88	90
kaaf	0.88	0.93	0.90	90
khaa	0.89	0.93	0.91	91
la	0.86	0.86	0.86	91
laam	0.81	0.90	0.85	89
meem	0.91	0.78	0.84	91
nun	0.86	0.94	0.90	85
ra	0.93	0.87	0.90	86
saad	0.98	0.97	0.97	91
seen	0.96	0.90	0.93	91
sheen	0.97	0.95	0.96	91
ta	0.88	0.84	0.86	91
taa	0.86	0.91	0.89	90
thaa	0.85	0.87	0.86	91
thal	0.85	0.93	0.89	91
toot	0.91	0.91	0.91	91
waw	0.99	0.92	0.95	83
ya	0.94	0.92	0.93	90
yaa	0.91	0.93	0.92	91
zay	0.83	0.87	0.85	89
accuracy			0.90	2867
macro avg	0.90	0.90	0.90	2867
weighted avg	0.90	0.90	0.90	2867

