

```
#efficientnetb0, fully unfroze layers (cropped images using mediaPip, fixed dataset issue)
```

🏠 ⬆ ⬇ 📄 📝 🗑

```
#Load models and evaluate
ef_model = load_model('best_ef_model_plus_mediapipe_cropped_V2_unfreeze_test.keras')
y_pred = ef_model.predict(test_generator3)
y_true = test_generator3.classes
y_pred_classes = np.argmax(y_pred, axis=1)

evaluate_model_new("EfficientNetB0", ef_model, test_generator3, y_true, y_pred_classes)
```

173/173 ————— 22s 99ms/step

Model: EfficientNetB0

Accuracy: 0.9662186705412278

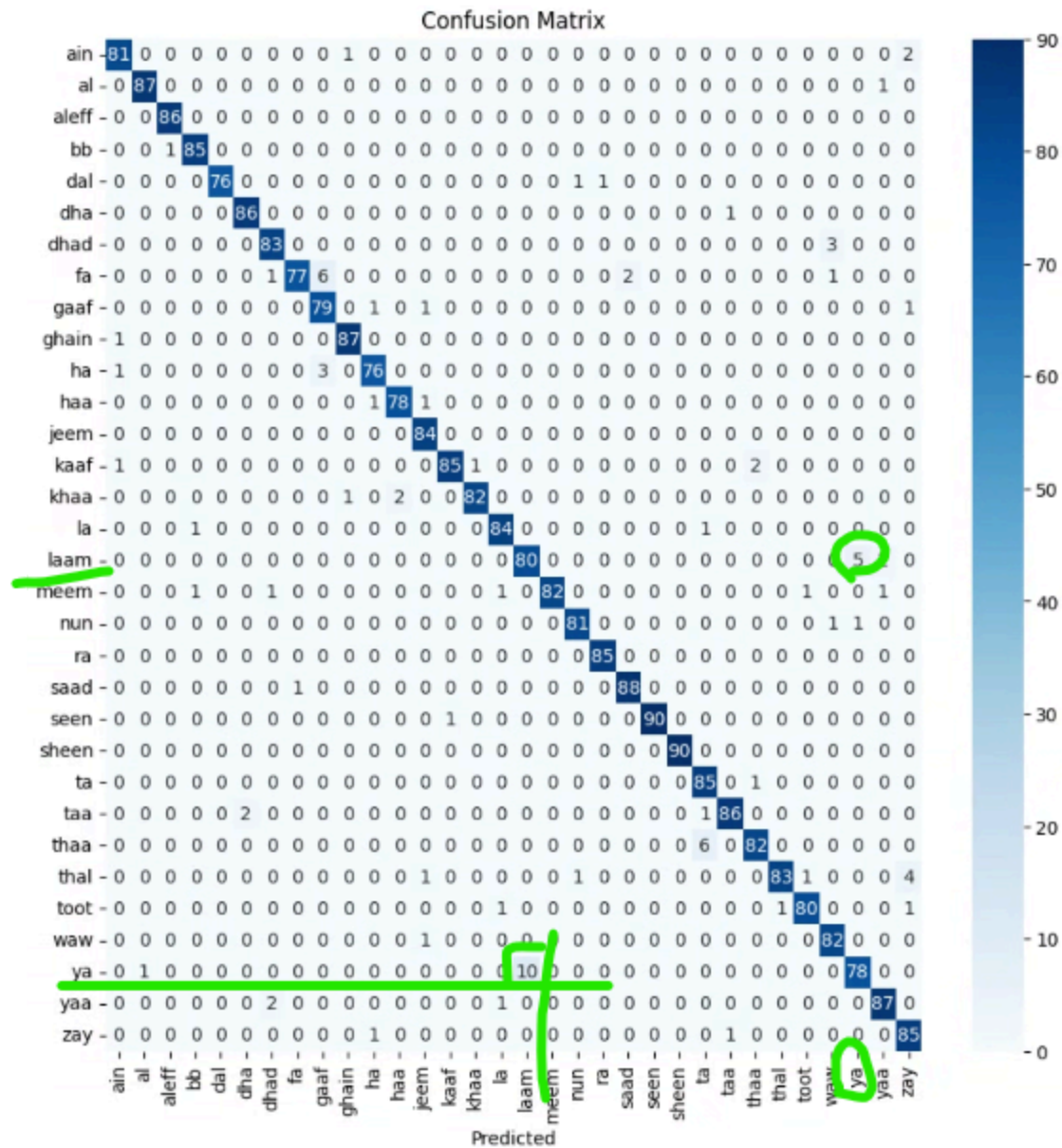
Precision: 0.966916451030635

Recall: 0.9662186705412278

F1 Score: 0.9661969512877646

Classification Report:

	precision	recall	f1-score	support
ain	0.96	0.96	0.96	84
al	0.99	0.99	0.99	88
aleff	0.99	1.00	0.99	86
bb	0.98	0.99	0.98	86
dal	1.00	0.97	0.99	78
dha	0.98	0.99	0.98	87
dhad	0.95	0.97	0.96	86
fa	0.99	0.89	0.93	87
gaaf	0.90	0.96	0.93	82
ghain	0.98	0.99	0.98	88
ha	0.96	0.95	0.96	80
haa	0.97	0.97	0.97	80
jeem	0.95	1.00	0.98	84
kaaf	0.99	0.96	0.97	89
khaa	0.99	0.96	0.98	85
la	0.97	0.98	0.97	86
laam	0.89	0.92	0.90	87
meem	1.00	0.94	0.97	87
nun	0.98	0.98	0.98	83
ra	0.99	1.00	0.99	85
saad	0.98	0.99	0.98	89
seen	1.00	0.99	0.99	91
sheen	1.00	1.00	1.00	90
ta	0.91	0.99	0.95	86
taa	0.98	0.97	0.97	89
thaa	0.96	0.93	0.95	88
thal	0.99	0.92	0.95	90
toot	0.98	0.96	0.97	83
waw	0.94	0.99	0.96	83
ya	0.93	0.88	0.90	89
yaa	0.96	0.97	0.96	90
zay	0.91	0.98	0.94	87
accuracy			0.97	2753
macro avg	0.97	0.97	0.97	2753
weighted avg	0.97	0.97	0.97	2753



Landmarks model

```

def build_enhanced_landmark_model(input_shape, num_classes, learning_rate=0.001):
    |

    model = Sequential([
        # Input Layer
        Dense(256, activation='relu', input_shape=input_shape),
        BatchNormalization(),
        Dropout(0.4),

        # Hidden layers with different sizes
        Dense(128, activation='relu'),
        BatchNormalization(),
        Dropout(0.3),

        Dense(64, activation='relu'),
        BatchNormalization(),
        Dropout(0.2),

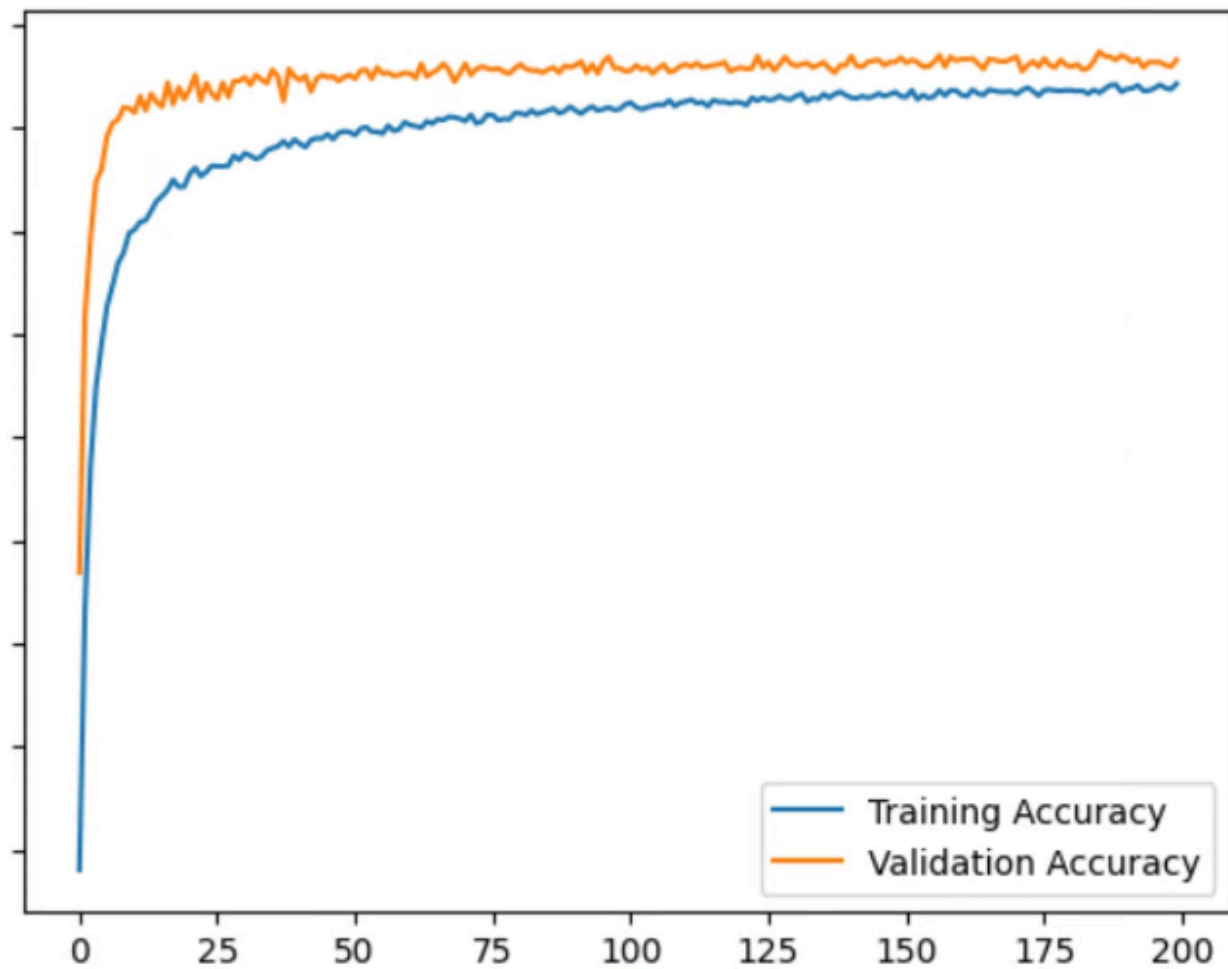
        Dense(32, activation='relu'),
        BatchNormalization(),
        Dropout(0.1),

        # Output Layer
        Dense(num_classes, activation='softmax')
    ])

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

    return model

```



87/87 ————— 1s 6ms/step
Model: Enhanced Landmark Dense Neural Network
Accuracy: 0.9633526850507983
Precision: 0.964171288683201
Recall: 0.9633526850507983
F1 Score: 0.9633067003743021
Classification Report:

	precision	recall	f1-score	support
ain	0.98	0.99	0.98	84
al	0.99	1.00	0.99	89
aleff	0.97	0.98	0.97	87
bb	1.00	0.95	0.98	85
dal	0.88	0.97	0.93	78
dha	1.00	0.98	0.99	87
dhad	0.99	0.95	0.97	86
fa	0.96	0.91	0.94	88
gaaf	0.99	0.94	0.96	82
ghain	0.98	1.00	0.99	88
ha	0.92	0.96	0.94	81
haa	0.94	0.99	0.96	80
jeem	0.99	0.95	0.97	84
kaaf	0.97	0.97	0.97	89
khaa	0.99	0.96	0.98	85
la	1.00	0.99	0.99	86
laam	0.88	0.95	0.92	87
meem	1.00	0.99	0.99	87
nun	0.94	0.81	0.87	83
ra	0.98	0.98	0.98	85
saad	0.93	0.98	0.95	89
seen	0.96	0.99	0.97	91
sheen	0.99	0.99	0.99	91
ta	0.94	0.95	0.95	86
taa	0.98	0.99	0.98	89
thaa	0.99	0.99	0.99	88
thal	1.00	0.97	0.98	90
toot	0.94	0.95	0.95	82
waw	0.94	0.98	0.96	83
ya	0.89	0.90	0.89	89
yaa	1.00	0.99	0.99	90
zay	0.95	0.93	0.94	87
accuracy			0.96	2756
macro avg	0.96	0.96	0.96	2756
weighted avg	0.96	0.96	0.96	2756

Confusion matrix for the Arabic Vowel Segmentation dataset. The matrix shows True vs Predicted counts for 26 classes. A color bar on the right indicates the count scale from 0 to 81. Several cells are circled in green, highlighting specific misclassifications or low counts.

True \ Predicted	ain	al	aleff	ba	dal	dha	dhad	fa	gaaf	ghain	ha	haa	jeem	kaaf	khaa	la	laam	meem	nun	ra	saad	seen	sheen	ta	taa	thaa	thal	toot	waw	ya	yaa	zay			
ain	83	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
al	0	89	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
aleff	0	0	85	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0		
ba	0	0	0	81	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
dal	0	0	0	0	76	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
dha	0	0	0	0	0	85	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
dhad	0	0	0	0	0	0	82	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	
fa	0	0	0	0	0	0	0	80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
gaaf	0	0	0	0	0	0	0	0	1	77	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
ghain	0	0	0	0	0	0	0	0	0	0	88	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ha	0	0	0	0	0	0	0	0	0	0	0	78	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
haa	0	0	0	0	0	0	0	0	0	0	0	0	79	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
jeem	0	0	0	0	0	0	0	0	0	0	0	0	0	80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
kaaf	0	0	0	0	0	0	0	0	0	0	0	0	0	0	86	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
khaa	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	82	0	0	0	0	0														



```
: #display the image
loaded_image = load_new_image('images/nun/1_21_M_nun_2.jpg')
loaded_image_2 = load_new_image('images/dal/1_21_M_dal_3.jpg')

plt.imshow(loaded_image, cmap = "gray")
plt.axis('off')
plt.show()
plt.imshow(loaded_image_2)
plt.axis('off')
plt.show()
```



X

```
#display the image
loaded_image = load_new_image('images/laam/1_21_M_laam_0.jpg')
loaded_image_2 = load_new_image('images/ya/2_20_M_ya_1.jpg')

plt.imshow(loaded_image, cmap = "gray")
plt.axis('off')
plt.show()
plt.imshow(loaded_image_2)
plt.axis('off')
plt.show()
```

