In [1]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, roc_curve, auc, f1_score, recall
import os
import cv2
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.applications import VGG16
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import seaborn as sns
import pywt
from matplotlib import cm
from PIL import Image
from io import BytesIO

# Define tumor classes
path = os.listdir('brain_tumor/Training/')
classes = {'no_tumor': 0, 'pituitary_tumor': 1}

# Function to apply wavelet transform
def apply_wavelet_transform(image, wavelet='db1', level=1):
    # Perform wavelet transform
    coeffs = pywt.wavedec2(image, wavelet=wavelet, level=level)
    # Reconstruct the image from the wavelet coefficients
    reconstructed_image = pywt.waverec2(coeffs, wavelet=wavelet)
    return reconstructed_image

# Load and preprocess data with wavelet transform
X_vgg_wavelet = []
Y_vgg_wavelet = []
for cls in classes:
    pth = 'brain_tumor/Training/' + cls
    for j in os.listdir(pth):
        img_vgg = cv2.imread(pth + '/' + j)
        img_vgg = cv2.resize(img_vgg, (224, 224))

        # Apply wavelet transform to each channel
        img_wavelet = np.zeros_like(img_vgg, dtype=np.float32)
        for channel in range(3):
            img_wavelet[:, :, channel] = apply_wavelet_transform(img_vgg[:,

        X_vgg_wavelet.append(img_wavelet)
        Y_vgg_wavelet.append(classes[cls])

X_vgg_wavelet = np.array(X_vgg_wavelet)
Y_vgg_wavelet = np.array(Y_vgg_wavelet)

# Preprocess data for VGG-16
X_vgg_wavelet = preprocess_input(X_vgg_wavelet)
Y_vgg_wavelet = tf.keras.utils.to_categorical(Y_vgg_wavelet, num_classes=2)
```

```python
# Split the dataset
xtrain_vgg_wavelet, xtest_vgg_wavelet, ytrain_vgg_wavelet, ytest_vgg_wavelet
    X_vgg_wavelet, Y_vgg_wavelet, random_state=10, test_size=.20)

# Data Augmentation
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,
    fill_mode='nearest'
)
datagen.fit(xtrain_vgg_wavelet)

# Build Enhanced VGG-16 model
def build_vgg16_model(input_shape):
    base_model = VGG16(weights='imagenet', include_top=False, input_shape=ir
    base_model.trainable = False
    model = models.Sequential([
        base_model,
        layers.Flatten(),
        layers.Dense(512, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(256, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(2, activation='softmax')
    ])
    return model

# Initialize and compile VGG-16 model
input_shape_vgg = (224, 224, 3)
vgg16_model_wavelet = build_vgg16_model(input_shape_vgg)
vgg16_model_wavelet.compile(optimizer='adam',
                            loss='categorical_crossentropy',
                            metrics=['accuracy'])

# Train VGG-16 model
history_vgg_wavelet = vgg16_model_wavelet.fit(
    datagen.flow(xtrain_vgg_wavelet, ytrain_vgg_wavelet, batch_size=32),
    epochs=10,
    validation_data=(xtest_vgg_wavelet, ytest_vgg_wavelet)
)

# Evaluate VGG-16 model
vgg16_test_loss_wavelet, vgg16_test_acc_wavelet = vgg16_model_wavelet.evalua
print(f"Enhanced VGG-16 with Wavelet Test Accuracy: {vgg16_test_acc_wavelet:

# Plot training history
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(history_vgg_wavelet.history['accuracy'], label='Training Accuracy')
plt.plot(history_vgg_wavelet.history['val_accuracy'], label='Validation Accu
```

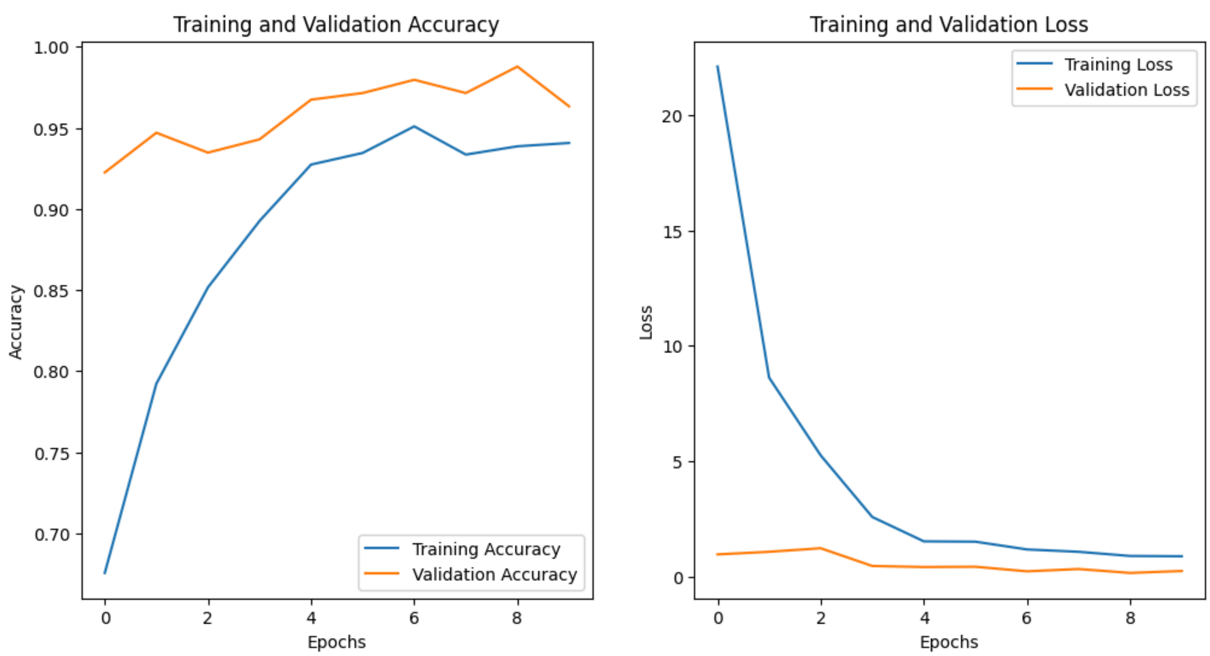
```python
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(history_vgg_wavelet.history['loss'], label='Training Loss')
plt.plot(history_vgg_wavelet.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Training and Validation Loss')
plt.show()

# Get predictions
vgg16_preds_wavelet = vgg16_model_wavelet.predict(xtest_vgg_wavelet)
vgg16_preds_classes_wavelet = np.argmax(vgg16_preds_wavelet, axis=1)
true_labels = np.argmax(ytest_vgg_wavelet, axis=1)

# Enhanced Confusion Matrix Visualization
conf_matrix_wavelet = confusion_matrix(true_labels, vgg16_preds_classes_wave
plt.figure(figsize=(8, 6))
sns.set(font_scale=1.2)
ax = sns.heatmap(conf_matrix_wavelet, annot=True, fmt='d', cmap='Blues',
                 cbar=True, annot_kws={"size": 16},
                 xticklabels=['No Tumor', 'Pituitary Tumor'],
                 yticklabels=['No Tumor', 'Pituitary Tumor'])

ax.set_xticklabels(ax.get_xticklabels(), rotation=0, ha='center')
ax.set_yticklabels(ax.get_yticklabels(), rotation=0, va='center')

plt.xlabel('Predicted Labels', fontsize=14, labelpad=10)
plt.ylabel('True Labels', fontsize=14, labelpad=10)
plt.title('Enhanced Confusion Matrix with Wavelet', fontsize=16, pad=20)

# Add color bar
cbar = ax.collections[0].colorbar
cbar.ax.tick_params(labelsize=12)

plt.tight_layout()
plt.show()

# Classification Report
print("Classification Report:")
print(classification_report(true_labels, vgg16_preds_classes_wavelet,
                            target_names=['No Tumor', 'Pituitary Tumor']))

# ROC Curve
vgg16_probs_wavelet = vgg16_preds_wavelet[:, 1]
fpr_vgg16_wavelet, tpr_vgg16_wavelet, _ = roc_curve(true_labels, vgg16_probs
roc_auc_vgg16_wavelet = auc(fpr_vgg16_wavelet, tpr_vgg16_wavelet)

plt.figure(figsize=(8, 6))
plt.plot(fpr_vgg16_wavelet, tpr_vgg16_wavelet,
         label=f'VGG-16 with Wavelet (AUC = {roc_auc_vgg16_wavelet:.2f})', l
plt.plot([0, 1], [0, 1], 'k--', linewidth=1)
```

```
plt.xlabel('False Positive Rate', fontsize=12)
plt.ylabel('True Positive Rate', fontsize=12)
plt.title('ROC Curve for VGG-16 Model with Wavelet', fontsize=14)
plt.legend(loc='lower right', fontsize=12)
plt.grid(True, alpha=0.3)
plt.show()

# Precision-Recall Curve
precision, recall, _ = precision_recall_curve(true_labels, vgg16_probs_wavel
average_precision = average_precision_score(true_labels, vgg16_probs_wavelet

plt.figure(figsize=(8, 6))
plt.plot(recall, precision,
         label=f'VGG-16 with Wavelet (AP = {average_precision:.2f})',
         linewidth=2, color='darkorange')
plt.xlabel('Recall', fontsize=12)
plt.ylabel('Precision', fontsize=12)
plt.title('Precision-Recall Curve', fontsize=14)
plt.legend(loc='upper right', fontsize=12)
plt.grid(True, alpha=0.3)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.show()

# User Input Prediction Function
def predict_user_image(image_path, model):
    # Load and preprocess the image
    img = cv2.imread(image_path)
    if img is None:
        print(f"Error: Could not read image from {image_path}")
        return

    # Display original image
    plt.figure(figsize=(6, 6))
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title('Original Image')
    plt.axis('off')
    plt.show()

    # Preprocess for prediction
    img = cv2.resize(img, (224, 224))

    # Apply wavelet transform
    img_wavelet = np.zeros_like(img, dtype=np.float32)
    for channel in range(3):
        img_wavelet[:, :, channel] = apply_wavelet_transform(img[:, :, chann

    img_wavelet = preprocess_input(img_wavelet)
    img_wavelet = np.expand_dims(img_wavelet, axis=0)  # Add batch dimensior

    # Make prediction
    pred = model.predict(img_wavelet)
    pred_class = np.argmax(pred, axis=1)[0]
    confidence = np.max(pred) * 100

    class_names = {0: 'No Tumor', 1: 'Pituitary Tumor'}
```

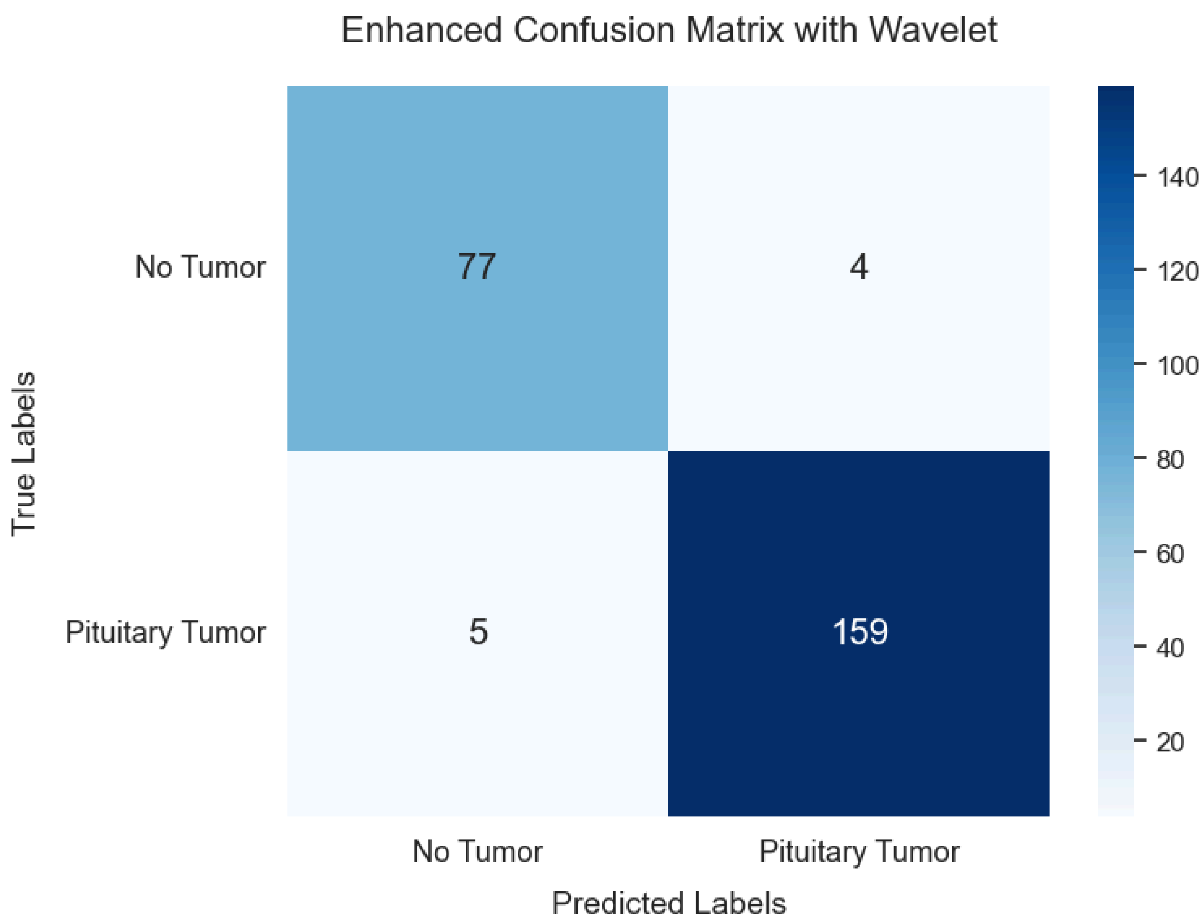
```python
    print("\nPrediction Result:")
    print(f"Class: {class_names[pred_class]}")
    print(f"Confidence: {confidence:.2f}%")
    print(f"Probability Distribution: No Tumor: {pred[0][0]*100:.2f}%, Pitui

    # Display prediction probabilities
    plt.figure(figsize=(8, 4))
    plt.bar(['No Tumor', 'Pituitary Tumor'], pred[0]*100, color=['green', 'r
    plt.title('Prediction Probabilities')
    plt.ylabel('Probability (%)')
    plt.ylim(0, 100)
    plt.grid(True, alpha=0.3)
    plt.show()

# Test with user input
user_image_path = 'image(12).jpg'  # Replace with your image path
if os.path.exists(user_image_path):
    predict_user_image(user_image_path, vgg16_model_wavelet)
else:
    print(f"File not found: {user_image_path}")
    print("Please ensure the image file exists in the current directory.")
```

```
C:\Users\Ananda kumar sahoo\AppData\Local\Programs\Python\Python312\Lib\site
-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWa
rning: Your `PyDataset` class should call `super().__init__(**kwargs)` in it
s constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max
_queue_size`. Do not pass these arguments to `fit()`, as they will be ignore
d.
  self._warn_if_super_not_called()
```

```
Epoch 1/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 175s 6s/step - accuracy: 0.6470 - loss: 20.7617 -
val_accuracy: 0.9224 - val_loss: 0.9649
Epoch 2/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 258s 7s/step - accuracy: 0.7565 - loss: 11.0149 -
val_accuracy: 0.9469 - val_loss: 1.0778
Epoch 3/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 249s 8s/step - accuracy: 0.8305 - loss: 6.1394 -
val_accuracy: 0.9347 - val_loss: 1.2317
Epoch 4/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 244s 8s/step - accuracy: 0.8988 - loss: 2.6568 -
val_accuracy: 0.9429 - val_loss: 0.4629
Epoch 5/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 246s 8s/step - accuracy: 0.9124 - loss: 1.9186 -
val_accuracy: 0.9673 - val_loss: 0.4183
Epoch 6/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 252s 8s/step - accuracy: 0.9351 - loss: 1.7335 -
val_accuracy: 0.9714 - val_loss: 0.4297
Epoch 7/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 201s 6s/step - accuracy: 0.9523 - loss: 1.0374 -
val_accuracy: 0.9796 - val_loss: 0.2313
Epoch 8/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 133s 4s/step - accuracy: 0.9395 - loss: 0.9399 -
val_accuracy: 0.9714 - val_loss: 0.3317
Epoch 9/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 135s 4s/step - accuracy: 0.9518 - loss: 0.8262 -
val_accuracy: 0.9878 - val_loss: 0.1615
Epoch 10/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 132s 4s/step - accuracy: 0.9278 - loss: 1.1832 -
val_accuracy: 0.9633 - val_loss: 0.2460
8/8 ━━━━━━━━━━━━━━━━━━━━ 24s 3s/step - accuracy: 0.9691 - loss: 0.1492
Enhanced VGG-16 with Wavelet Test Accuracy: 0.9633
```

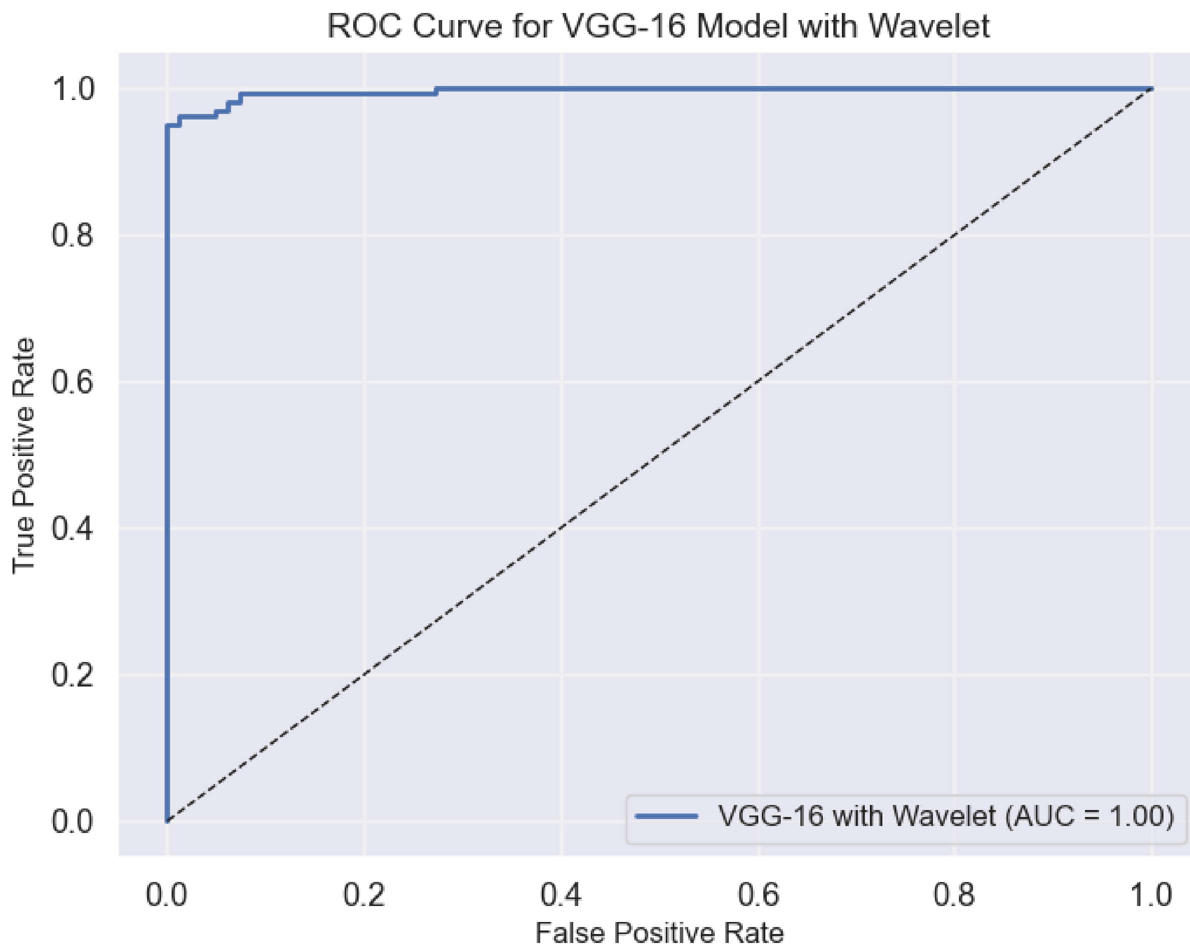


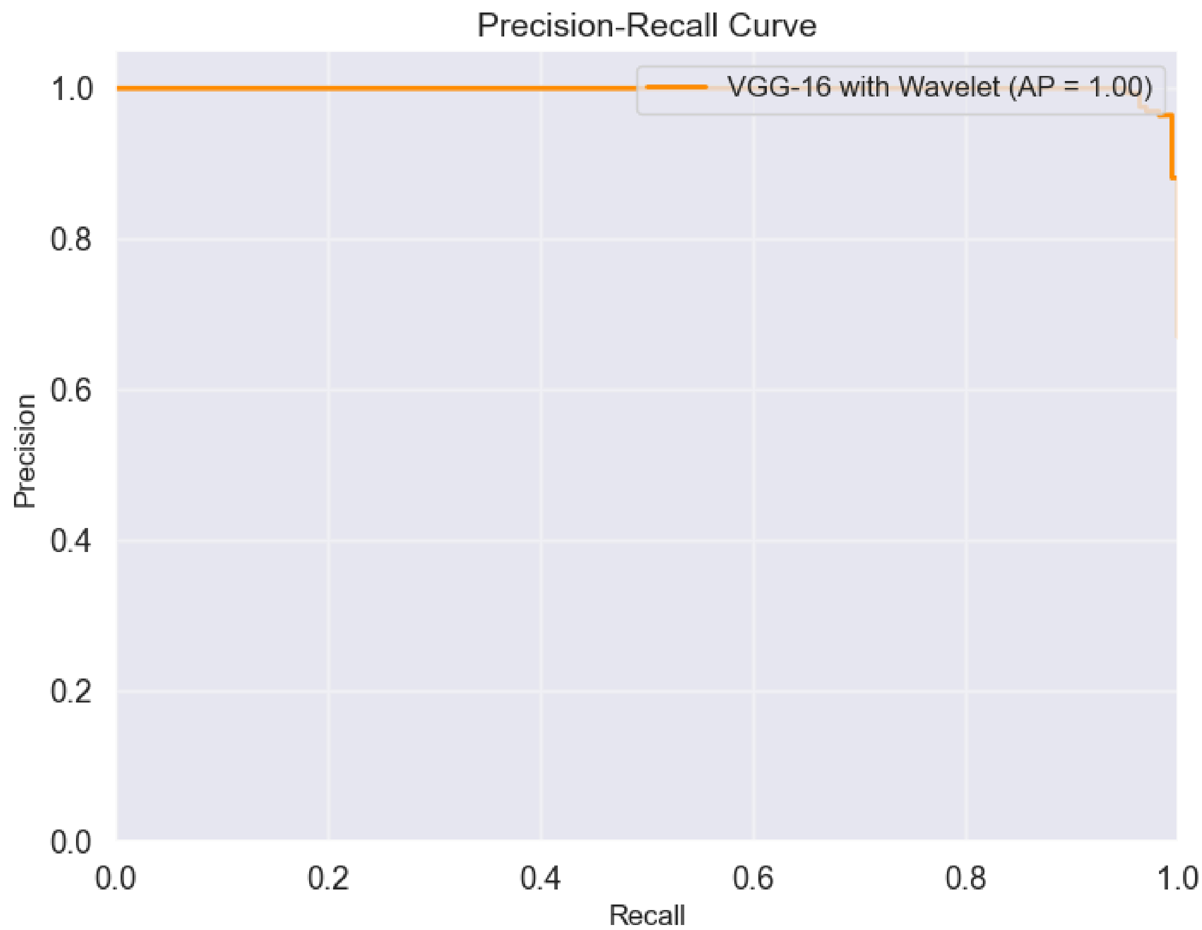

```
8/8 ━━━━━━━━━━━━━━━━━━━━ 24s 3s/step
```




```
Classification Report:
                 precision    recall  f1-score   support

       No Tumor       0.94      0.95      0.94        81
Pituitary Tumor       0.98      0.97      0.97       164

       accuracy                           0.96       245
      macro avg       0.96      0.96      0.96       245
   weighted avg       0.96      0.96      0.96       245
```

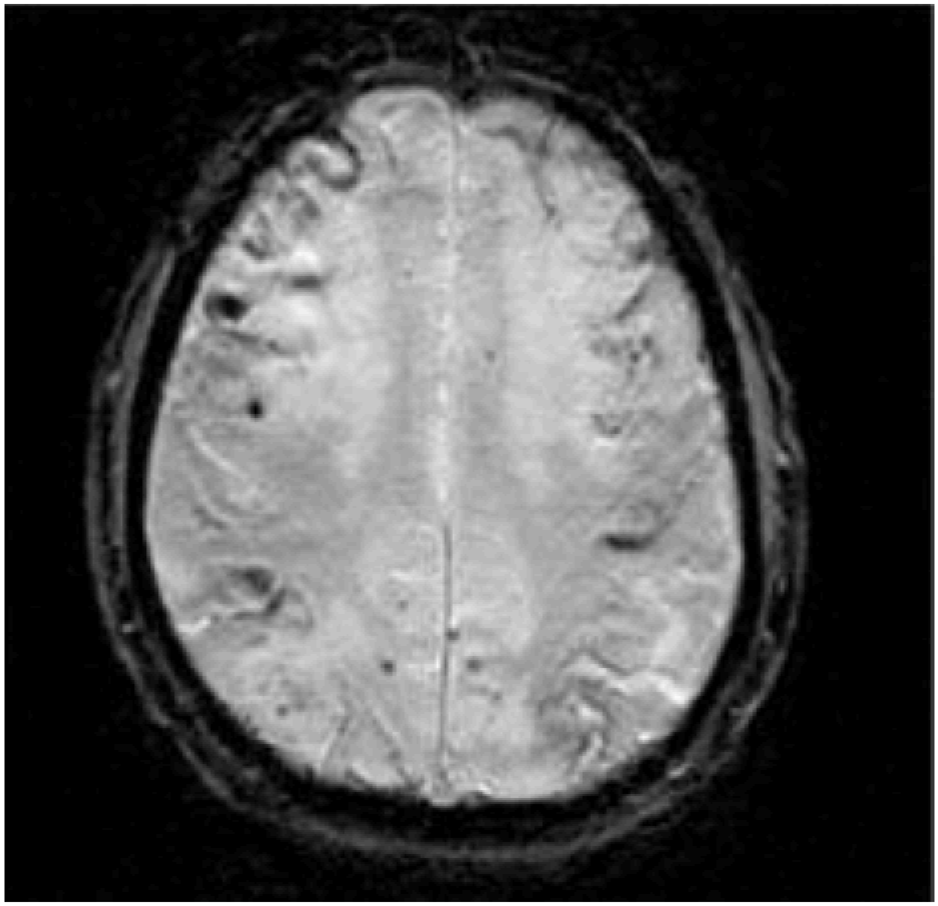
Loading [MathJax]/extensions/Safe.js


ROC Curve for VGG-16 Model with Wavelet
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
VGG-16 with Wavelet (AUC = 1.00)

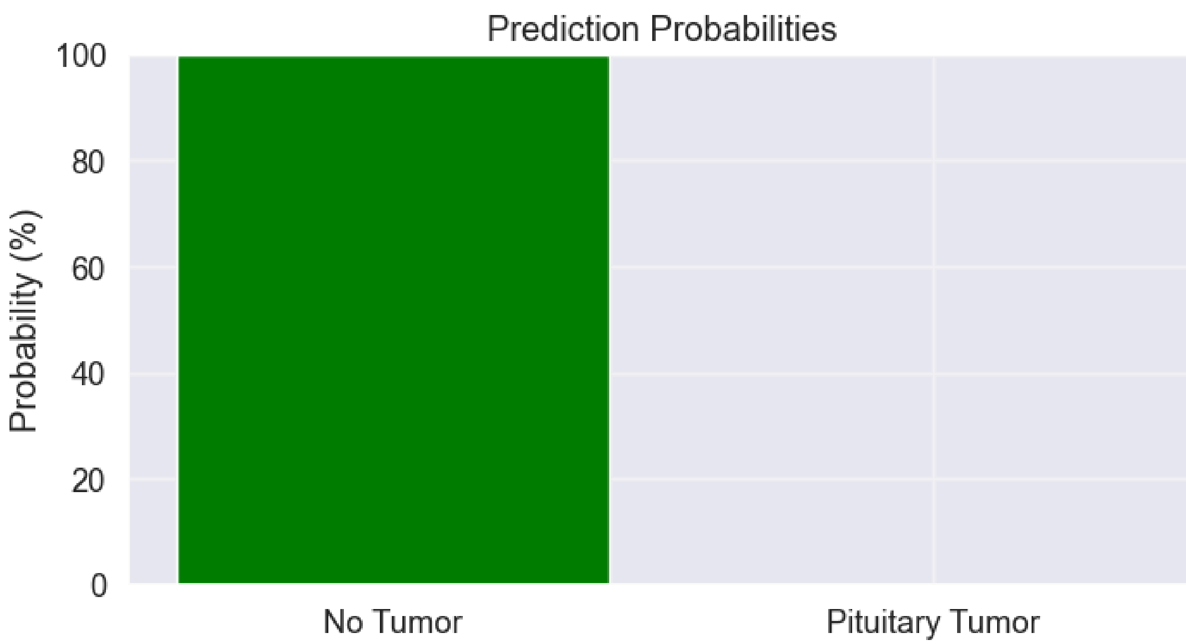


Loading [MathJax]/extensions/Safe.js

Original Image



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 338ms/step

Prediction Result:
Class: No Tumor
Confidence: 100.00%
Probability Distribution: No Tumor: 100.00%, Pituitary Tumor: 0.00%
```



In [ ]:

Loading [MathJax]/extensions/Safe.js