In [5]:
```
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_score, precision_score, confusion_matrix,
                             classification_report, precision_recall_curve,
                             average_precision_score)
import os
import cv2
import tensorflow as tf
from tensorflow.keras import layers, models
import seaborn as sns
import pywt

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

# Load and preprocess data
X = []
Y = []
for cls in classes:
    pth = 'brain_tumor/Training/' + cls
    for j in os.listdir(pth):
        img = cv2.imread(pth + '/' + j, 0)
        img = cv2.resize(img, (200, 200))
        img_filtered = cv2.GaussianBlur(img, (5, 5), 0)
        X.append(img_filtered)
        Y.append(classes[cls])

X = np.array(X)
Y = np.array(Y)

# Function to apply wavelet transform
def apply_wavelet_transform(image, wavelet='db1', level=1):
    coeffs = pywt.wavedec2(image, wavelet=wavelet, level=level)
    reconstructed_image = pywt.waverec2(coeffs, wavelet=wavelet)
    reconstructed_image = np.uint8(reconstructed_image)
    reconstructed_image = cv2.resize(reconstructed_image, (200, 200))
    return reconstructed_image

# Apply wavelet transform to each image
X_wavelet = np.array([apply_wavelet_transform(img) for img in X])

# Reshape and normalize data for CNN
X_wavelet_cnn = X_wavelet.reshape(len(X_wavelet), 200, 200, 1)
X_wavelet_cnn = X_wavelet_cnn / 255.0
Y_cnn = tf.keras.utils.to_categorical(Y, num_classes=2)

# Split the dataset
xtrain_wavelet_cnn, xtest_wavelet_cnn, ytrain_cnn, ytest_cnn = train_test_sp
    X_wavelet_cnn, Y_cnn, random_state=10, test_size=.20)
```

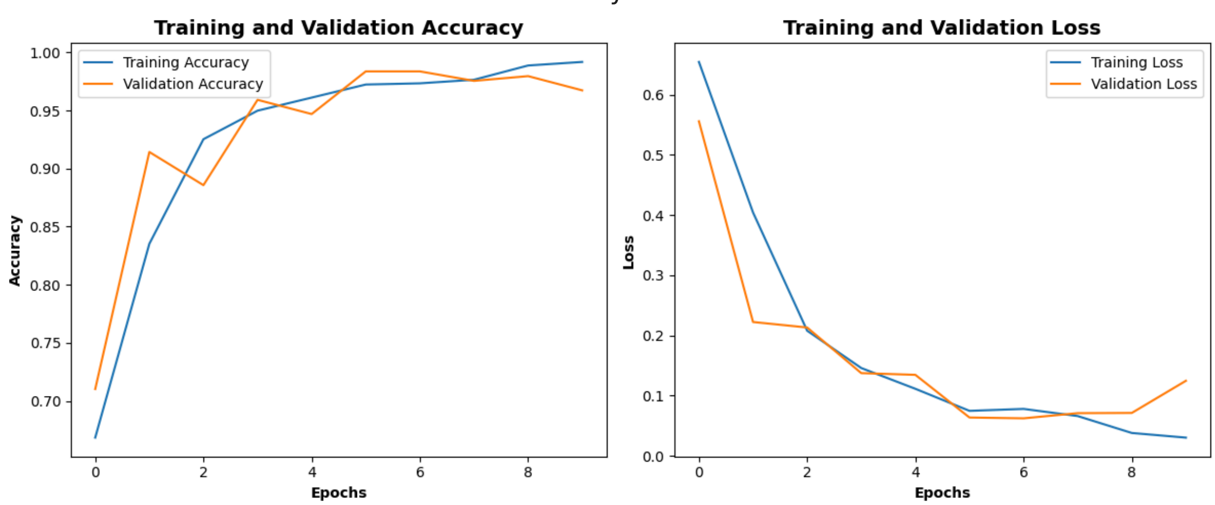
```
# Build Enhanced CNN model
def build_cnn_model(input_shape):
    model = models.Sequential([
        layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(128, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(256, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),
        layers.Flatten(),
        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, compile and train CNN model
input_shape = (200, 200, 1)
cnn_model_wavelet = build_cnn_model(input_shape)
cnn_model_wavelet.compile(optimizer='adam',
                        loss='categorical_crossentropy',
                        metrics=['accuracy'])

history_cnn_wavelet = cnn_model_wavelet.fit(xtrain_wavelet_cnn, ytrain_cnn,
                                          epochs=10, batch_size=32,
                                          validation_data=(xtest_wavelet_cnn

# Evaluate model
cnn_test_loss_wavelet, cnn_test_acc_wavelet = cnn_model_wavelet.evaluate(xte
print(f"Enhanced CNN with Wavelet Test Accuracy: {cnn_test_acc_wavelet:.4f}"

# Plot training history
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history_cnn_wavelet.history['accuracy'], label='Training Accuracy')
plt.plot(history_cnn_wavelet.history['val_accuracy'], label='Validation Accu
plt.title('Training and Validation Accuracy', fontsize=14, fontweight='bold'
plt.xlabel('Epochs', fontweight='bold')
plt.ylabel('Accuracy', fontweight='bold')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history_cnn_wavelet.history['loss'], label='Training Loss')
plt.plot(history_cnn_wavelet.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss', fontsize=14, fontweight='bold')
plt.xlabel('Epochs', fontweight='bold')
plt.ylabel('Loss', fontweight='bold')
plt.legend()
plt.tight_layout()
plt.show()

# Get predictions and metrics
```

```python
cnn_probs_wavelet = cnn_model_wavelet.predict(xtest_wavelet_cnn)[:, 1]
cnn_preds_classes_wavelet = np.argmax(cnn_model_wavelet.predict(xtest_wavele
true_labels = np.argmax(ytest_cnn, axis=1)

# ROC Curve
fpr, tpr, _ = roc_curve(true_labels, cnn_probs_wavelet)
roc_auc = auc(fpr, tpr)

# Precision-Recall Curve
precision, recall, _ = precision_recall_curve(true_labels, cnn_probs_wavelet
avg_precision = average_precision_score(true_labels, cnn_probs_wavelet)

# Plot ROC and Precision-Recall curves
plt.figure(figsize=(12, 5))

# ROC Curve
plt.subplot(1, 2, 1)
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc_a
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate', fontweight='bold')
plt.ylabel('True Positive Rate', fontweight='bold')
plt.title('Receiver Operating Characteristic', fontweight='bold')
plt.legend(loc="lower right")

# Precision-Recall Curve
plt.subplot(1, 2, 2)
plt.plot(recall, precision, color='blue', lw=2,
        label=f'Precision-Recall curve (AP = {avg_precision:.2f})')
plt.xlabel('Recall', fontweight='bold')
plt.ylabel('Precision', fontweight='bold')
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
plt.title('Precision-Recall Curve', fontweight='bold')
plt.legend(loc="lower left")

plt.tight_layout()
plt.show()

# Calculate metrics
f1 = f1_score(true_labels, cnn_preds_classes_wavelet)
recall = recall_score(true_labels, cnn_preds_classes_wavelet)
precision = precision_score(true_labels, cnn_preds_classes_wavelet)
accuracy = accuracy_score(true_labels, cnn_preds_classes_wavelet)

# Enhanced Confusion Matrix
plt.figure(figsize=(8, 6))
conf_matrix = confusion_matrix(true_labels, cnn_preds_classes_wavelet)
ax = sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
                cbar=True, annot_kws={'size': 16, 'weight': 'bold'},
                linewidths=0.5, linecolor='gray')

# Customize the plot
title_font = {'fontsize': 18, 'fontweight': 'bold', 'ha': 'center'}
label_font = {'fontsize': 14, 'fontweight': 'bold'}
```

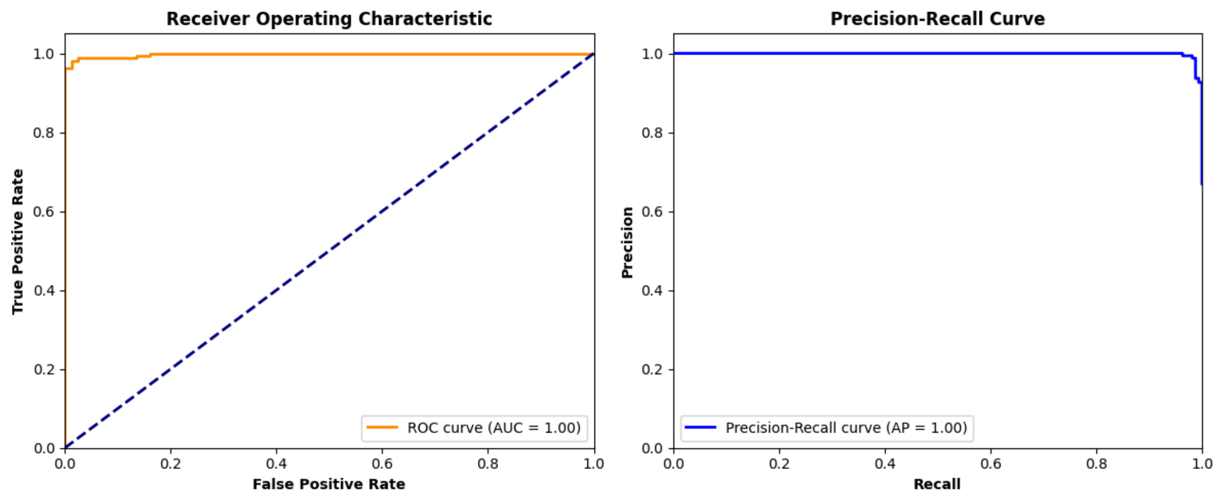
```
ax.set_title('Brain Tumor Classification\nConfusion Matrix', **title_font, p
ax.set_xlabel('\nPredicted Diagnosis', **label_font)
ax.set_ylabel('True Diagnosis\n', **label_font)

# Customize tick labels
ax.set_xticklabels(['Healthy\n(No Tumor)', 'Pituitary\nTumor'],
                   fontsize=12, rotation=0, ha='center')
ax.set_yticklabels(['Healthy\n(No Tumor)', 'Pituitary\nTumor'],
                   fontsize=12, rotation=0, va='center')

# Add performance metrics inside the plot
metrics_text = (f"Accuracy: {accuracy:.2f}\n"
                f"Precision: {precision:.2f}\n"
                f"Recall: {recall:.2f}\n"
                f"F1 Score: {f1:.2f}")
plt.text(2.5, 0.5, metrics_text,
        fontsize=12, bbox=dict(facecolor='white', alpha=0.8))

plt.tight_layout()
plt.show()

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

# Print additional metrics
print("\nDetailed Metrics:")
print(f"Accuracy: {accuracy:.4f}")
print(f"F1 Score: {f1:.4f}")
print(f"Recall: {recall:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Average Precision: {avg_precision:.4f}")
print(f"ROC AUC: {roc_auc:.4f}")
```

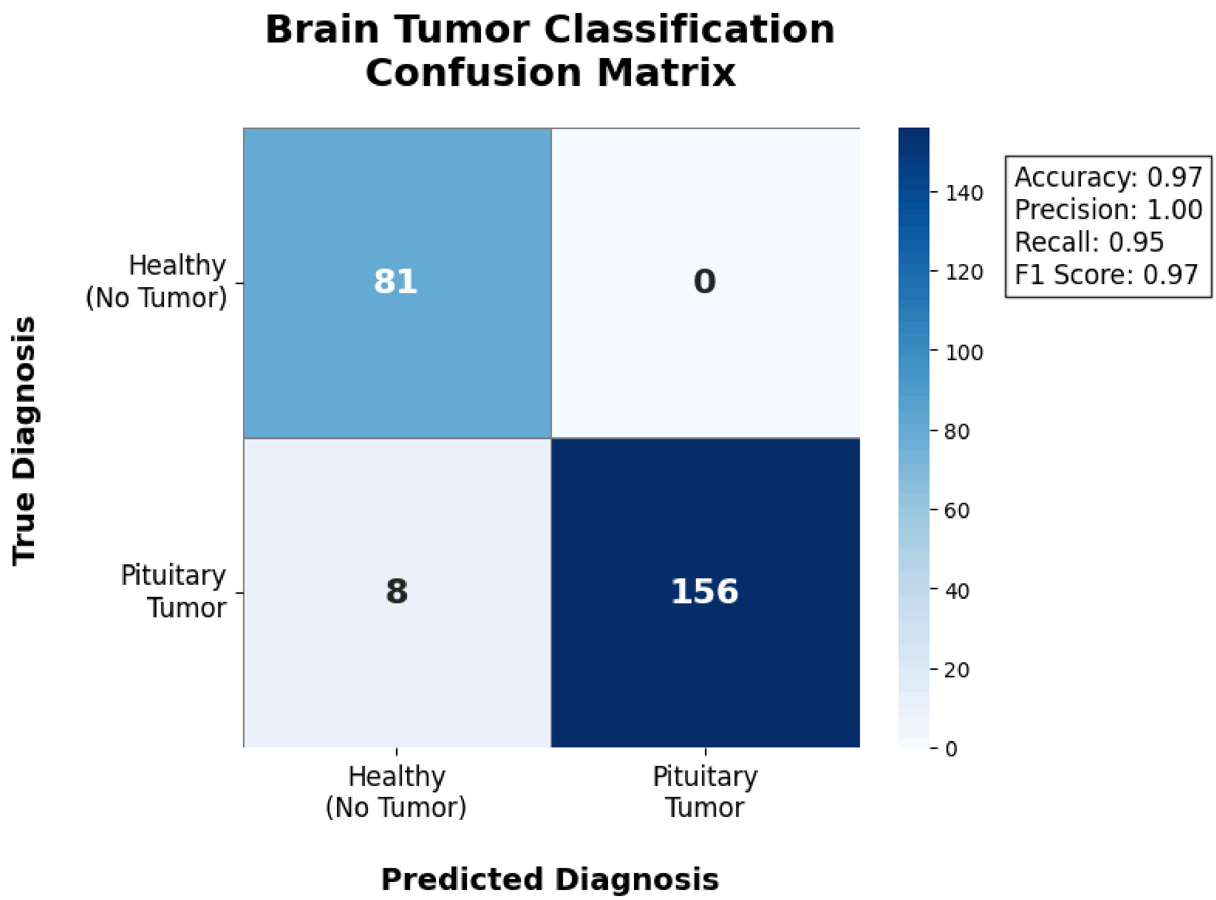
```
C:\Users\Ananda kumar sahoo\AppData\Local\Programs\Python\Python312\Lib\site
-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do n
ot pass an `input_shape`/`input_dim` argument to a layer. When using Sequent
ial models, prefer using an `Input(shape)` object as the first layer in the
model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
Epoch 1/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 25s 717ms/step - accuracy: 0.6162 - loss: 0.6933
- val_accuracy: 0.7102 - val_loss: 0.5560
Epoch 2/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 22s 709ms/step - accuracy: 0.7876 - loss: 0.4759
- val_accuracy: 0.9143 - val_loss: 0.2224
Epoch 3/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 22s 711ms/step - accuracy: 0.9206 - loss: 0.2194
- val_accuracy: 0.8857 - val_loss: 0.2132
Epoch 4/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 22s 709ms/step - accuracy: 0.9428 - loss: 0.1615
- val_accuracy: 0.9592 - val_loss: 0.1374
Epoch 5/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 24s 760ms/step - accuracy: 0.9595 - loss: 0.1008
- val_accuracy: 0.9469 - val_loss: 0.1346
Epoch 6/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 25s 799ms/step - accuracy: 0.9657 - loss: 0.0861
- val_accuracy: 0.9837 - val_loss: 0.0637
Epoch 7/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 25s 793ms/step - accuracy: 0.9861 - loss: 0.0435
- val_accuracy: 0.9837 - val_loss: 0.0623
Epoch 8/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 25s 798ms/step - accuracy: 0.9609 - loss: 0.0965
- val_accuracy: 0.9755 - val_loss: 0.0709
Epoch 9/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 25s 802ms/step - accuracy: 0.9907 - loss: 0.0325
- val_accuracy: 0.9796 - val_loss: 0.0713
Epoch 10/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 25s 803ms/step - accuracy: 0.9899 - loss: 0.0304
- val_accuracy: 0.9673 - val_loss: 0.1247
8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 171ms/step - accuracy: 0.9776 - loss: 0.0764
Enhanced CNN with Wavelet Test Accuracy: 0.9673
```




```
8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 182ms/step
8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 166ms/step
```





Loading [MathJax]/extensions/Safe.js

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

       No Tumor       0.91      1.00      0.95        81
Pituitary Tumor       1.00      0.95      0.97       164

       accuracy                           0.97       245
      macro avg       0.96      0.98      0.96       245
   weighted avg       0.97      0.97      0.97       245


Detailed Metrics:
Accuracy: 0.9673
F1 Score: 0.9750
Recall: 0.9512
Precision: 1.0000
Average Precision: 0.9990
ROC AUC: 0.9978
```

In [ ]: