In [5]:
```
#CNN +WAVELET+INT FUNTION
```

In [12]:
```
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)

# INT (Intensity Normalization Transformation) function
def int_function(image):
    # Normalize intensity values to [0, 1]
    normalized = (image - np.min(image)) / (np.max(image) - np.min(image) +
    # Scale to [0, 255] and convert to uint8
    return np.uint8(normalized * 255)

# Function to apply wavelet transform and INT function
def apply_wavelet_transform(image, wavelet='db1', level=1):
    # Apply wavelet transform
    coeffs = pywt.wavedec2(image, wavelet=wavelet, level=level)
    reconstructed_image = pywt.waverec2(coeffs, wavelet=wavelet)

    # Apply INT function
    reconstructed_image = int_function(reconstructed_image)

    # Resize to standard dimensions
    reconstructed_image = cv2.resize(reconstructed_image, (200, 200))
```

```
    return reconstructed_image

# Apply wavelet transform with INT 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 and INT Function Test Accuracy: {cnn_test_

# 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')
plt.xlabel('Epochs')
```

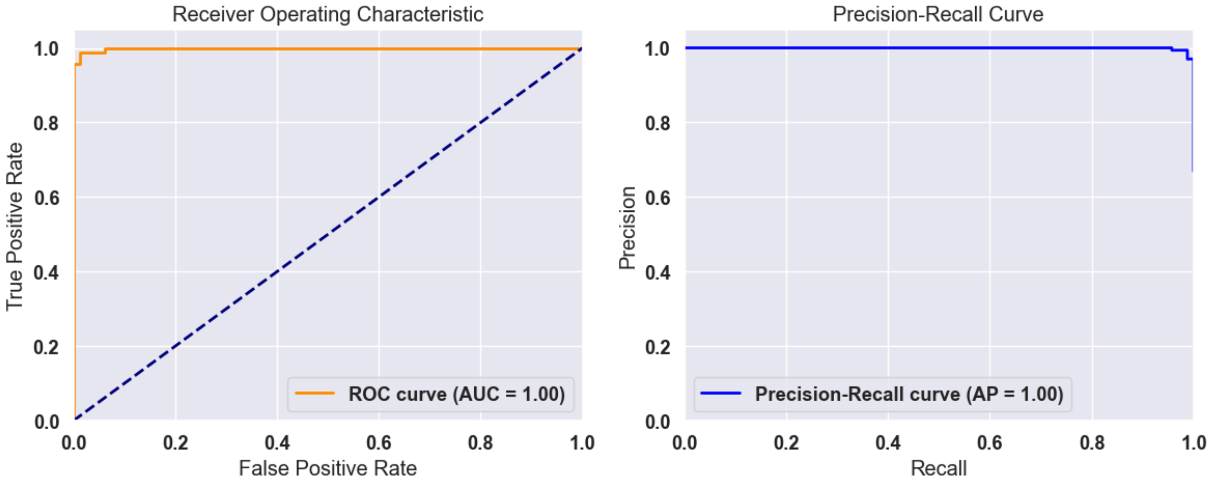
```
plt.ylabel('Accuracy')
plt.legend()

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')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.tight_layout()
plt.show()

# Get predictions and metrics
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')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
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')
plt.ylabel('Precision')
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
plt.title('Precision-Recall Curve')
plt.legend(loc="lower left")

plt.tight_layout()
plt.show()

# Confusion Matrix and Classification Report
conf_matrix = confusion_matrix(true_labels, cnn_preds_classes_wavelet)
```

```python
class_report = classification_report(true_labels, cnn_preds_classes_wavelet,
                                    target_names=['No Tumor', 'Pituitary Tum

print("Classification Report:")
print(class_report)

# Plot Confusion Matrix with counts and percentages
plt.figure(figsize=(6, 6))
group_counts = ["{0:0.0f}".format(value) for value in conf_matrix.flatten()]
group_percentages = ["{0:.1%}".format(value) for value in conf_matrix.flatte
labels = [f"{v1}\n({v2})" for v1, v2 in zip(group_counts, group_percentages)
labels = np.asarray(labels).reshape(2, 2)

sns.heatmap(conf_matrix, annot=labels, fmt='', cmap='Blues', cbar=False,
            xticklabels=['No Tumor', 'Pituitary Tumor'],
            yticklabels=['No Tumor', 'Pituitary Tumor'],
            annot_kws={"fontsize":12})

plt.xlabel('Predicted Label', fontsize=12)
plt.ylabel('True Label', fontsize=12)
plt.title('Confusion Matrix\n(Counts with Percentages)', fontsize=13, pad=20
plt.xticks(rotation=0)
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()

# Calculate and print additional 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)

print(f"\nAdditional Metrics:")
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}")
```

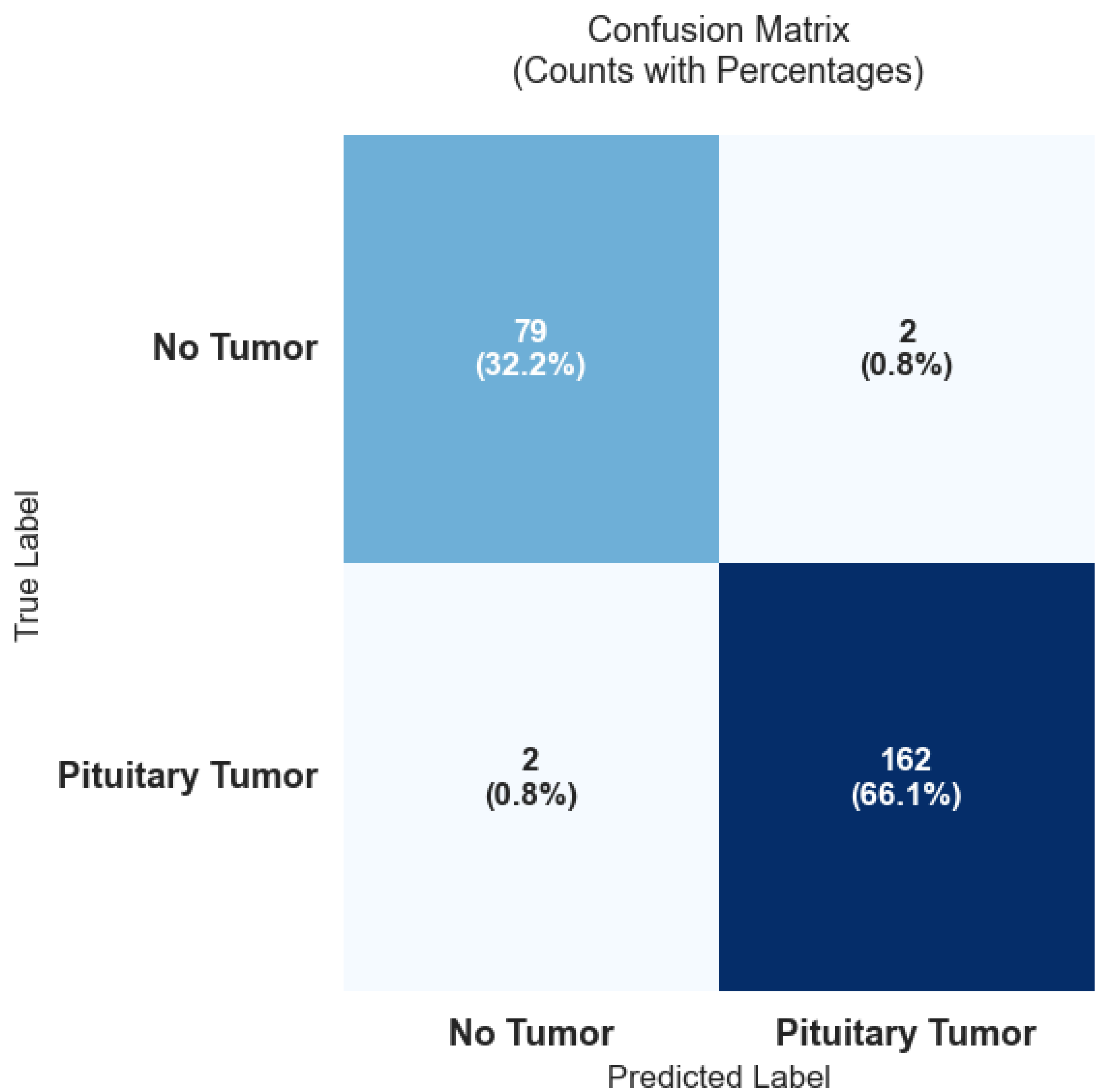
```
C:\Users\Ananda kumar sahoo\AppData\Local\Programs\Python\Python312\Lib\site
-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do n
ot pass an `input_shape`/`input_dim` argument to a layer. When using Sequent
ial models, prefer using an `Input(shape)` object as the first layer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

```
Epoch 1/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 24s 718ms/step - accuracy: 0.6559 - loss: 0.6682
- val_accuracy: 0.7143 - val_loss: 0.5216
Epoch 2/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 22s 702ms/step - accuracy: 0.7997 - loss: 0.4789
- val_accuracy: 0.8980 - val_loss: 0.2888
Epoch 3/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 22s 704ms/step - accuracy: 0.9206 - loss: 0.2395
- val_accuracy: 0.9551 - val_loss: 0.1340
Epoch 4/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 22s 720ms/step - accuracy: 0.9509 - loss: 0.1342
- val_accuracy: 0.9633 - val_loss: 0.0917
Epoch 5/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 22s 708ms/step - accuracy: 0.9800 - loss: 0.0772
- val_accuracy: 0.9796 - val_loss: 0.0480
Epoch 6/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 22s 702ms/step - accuracy: 0.9846 - loss: 0.0557
- val_accuracy: 0.9837 - val_loss: 0.0469
Epoch 7/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 22s 712ms/step - accuracy: 0.9938 - loss: 0.0301
- val_accuracy: 0.9796 - val_loss: 0.0558
Epoch 8/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 22s 710ms/step - accuracy: 0.9794 - loss: 0.0737
- val_accuracy: 0.9878 - val_loss: 0.0316
Epoch 9/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 22s 720ms/step - accuracy: 0.9879 - loss: 0.0468
- val_accuracy: 0.9878 - val_loss: 0.0448
Epoch 10/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 23s 726ms/step - accuracy: 0.9937 - loss: 0.0184
- val_accuracy: 0.9837 - val_loss: 0.0652
8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 164ms/step - accuracy: 0.9894 - loss: 0.0433
Enhanced CNN with Wavelet and INT Function Test Accuracy: 0.9837
```


```
8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 167ms/step
8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 148ms/step
```




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

       No Tumor       0.98      0.98      0.98        81
Pituitary Tumor       0.99      0.99      0.99       164

       accuracy                           0.98       245
      macro avg       0.98      0.98      0.98       245
   weighted avg       0.98      0.98      0.98       245
```




```
Additional Metrics:
F1 Score: 0.9878
Recall: 0.9878
Precision: 0.9878
Average Precision: 0.9994
ROC AUC: 0.9989
```

In [ ]:

Loading [MathJax]/extensions/Safe.js