Histopathologic Cancer Detection using CNNs

Objective: Create a Convolutional Neural Network (CNN) model capable of identifying metastatic cancer in small images taken from larger digital pathology scans of lymph node sections. This binary classification task uses data obtained from the Kaggle Histopathologic Cancer Detection dataset.

Dataset: The dataset consists of thousands of 96x96 pixel RGB image patches. A positive label (1) indicates that the central 32x32 pixel region of the patch contains at least one pixel of tumor tissue. A negative label (0) indicates no tumor tissue in the central region.

1. Setup and Libraries

```
import numpy as np
import pandas as pd
import os
import cv2
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report,
roc_curve, auc, accuracy_score

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout, BatchNormalization, Input
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint,
ReduceLROnPlateau

sns.set(style="whitegrid")
plt.rcParams['figure.figsize'] = (10, 6)
```

2. Data Loading and Exploration

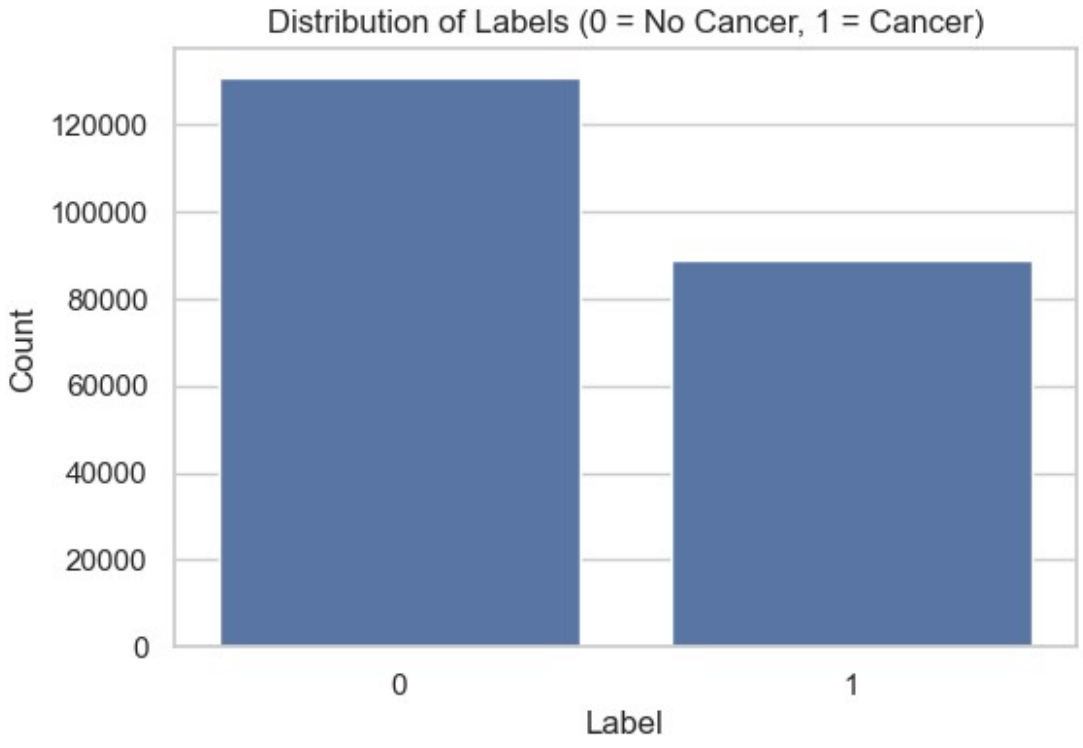
```
BASE_PATH = "~/histopathologic-cancer-detection/"

TRAIN_DIR = os.path.join(BASE_PATH, 'train/')
TEST_DIR = os.path.join(BASE_PATH, 'test/')
TRAIN_LABELS_PATH = os.path.join(BASE_PATH, 'train_labels.csv')

df_labels = pd.read_csv(TRAIN_LABELS_PATH)
df_labels['id'] = df_labels['id'].apply(lambda x: f"{x}.tif")
```

```
print("\nFirst 5 rows of labels:")
print(df_labels.head())
```

```
First 5 rows of labels:
                                       id  label
0  f38a6374c348f90b587e046aac6079959adf3835.tif      0
1  c18f2d887b7ae4f6742ee445113fa1aef383ed77.tif      1
2  755db6279dae599ebb4d39a9123cce439965282d.tif      0
3  bc3f0c64fb968ff4a8bd33af6971ecae77c75e08.tif      0
4  068aba587a4950175d04c680d38943fd488d6a9d.tif      0
```

```
label_counts = df_labels['label'].value_counts(normalize=True) * 100
print(label_counts)

plt.figure(figsize=(6, 4))
sns.countplot(x='label', data=df_labels)
plt.title('Distribution of Labels (0 = No Cancer, 1 = Cancer)')
plt.xlabel('Label')
plt.ylabel('Count')
plt.show()
```

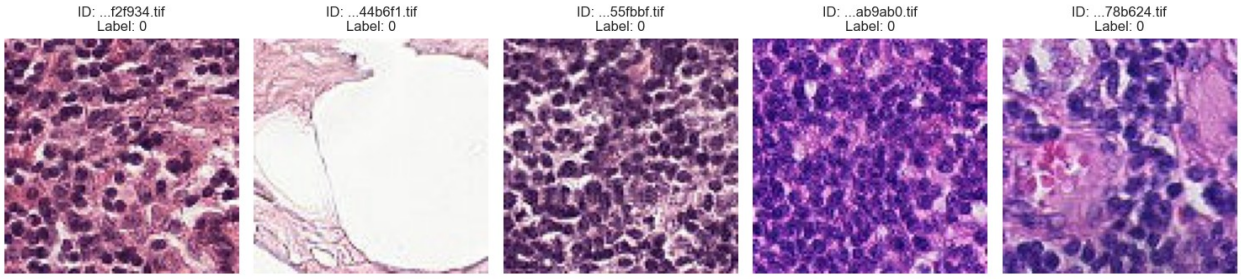
```
label
0    59.496875
1    40.503125
Name: proportion, dtype: float64
```



The dataset is somewhat imbalanced (~60% negative, ~40% positive). This isn't extreme imbalance, but we should keep it in mind for evaluation.

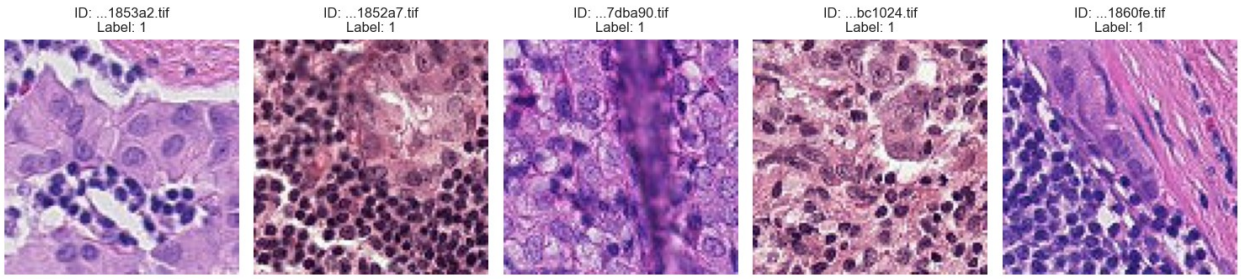
```
def display_samples(df, n_samples=5):
    """Displays random samples from the dataframe."""
    samples = df.sample(n_samples)
    plt.figure(figsize=(15, n_samples * 1.5)) # Adjust figure size
    for i, (idx, row) in enumerate(samples.iterrows()):
        img_path = os.path.join(TRAIN_DIR, row['id'])
        try:
            img = cv2.imread(img_path)
            if img is None:
                print(f"Warning: Could not load image {img_path}")
                continue
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) # OpenCV loads
as BGR

            plt.subplot(n_samples // 5 + 1, 5, i + 1)
            plt.imshow(img)
            plt.title(f"ID: ...{row['id'][-10:]}\nLabel:
{row['label']}")
            plt.axis('off')
        except Exception as e:
            print(f"Error loading/plotting image {img_path}: {e}")

    plt.tight_layout()
    plt.show()
    return

print("\nDisplaying random samples (Label 0 - No Cancer):")
display_samples(df_labels[df_labels['label'] == 0])
```

```
Displaying random samples (Label 0 - No Cancer):
```




```
print("\nDisplaying random samples (Label 1 - Cancer):")
display_samples(df_labels[df_labels['label'] == 1])
```

```
Displaying random samples (Label 1 - Cancer):
```



3. Data Preprocessing and Pipeline

```
IMG_WIDTH, IMG_HEIGHT = 96, 96
BATCH_SIZE = 64
SEED = 42

df_labels['label'] = df_labels['label'].astype(str)

train_df, val_df = train_test_split(
    df_labels,
    test_size=0.2,
    random_state=SEED,
    stratify=df_labels['label']
)

print(f"\nTraining set size: {train_df.shape[0]}")
print(f"Validation set size: {val_df.shape[0]}")


Training set size: 176020
Validation set size: 44005

# Training generator with augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255.,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode='nearest'
)

# Create generators from dataframes
train_generator = train_datagen.flow_from_dataframe(
    dataframe=train_df,
    directory=TRAIN_DIR,
    x_col='id',
    y_col='label',
```

```
    target_size=(IMG_WIDTH, IMG_HEIGHT),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    seed=SEED
)

val_datagen = ImageDataGenerator(rescale=1./255.)

validation_generator = val_datagen.flow_from_dataframe(
    dataframe=val_df,
    directory=TRAIN_DIR,
    x_col='id',
    y_col='label',
    target_size=(IMG_WIDTH, IMG_HEIGHT),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    shuffle=False,
    seed=SEED
)
```

```
Found 176020 validated image filenames belonging to 2 classes.
Found 44005 validated image filenames belonging to 2 classes.
```

```
# Verify generator output shapes
x_batch, y_batch = next(train_generator)
print(f"\nSample batch shape (images): {x_batch.shape}")
print(f"Sample batch shape (labels): {y_batch.shape}")
```

```
Sample batch shape (images): (64, 96, 96, 3)
Sample batch shape (labels): (64,)
```

4. CNN Model Building

```
def build_cnn_model(input_shape=(IMG_WIDTH, IMG_HEIGHT, 3)):
    model = Sequential(name="Histopathologic_CNN")

    model.add(Input(shape=input_shape))

    model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D((2, 2)))

    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D((2, 2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(BatchNormalization())
```

```
    model.add(MaxPooling2D((2, 2)))
    model.add(Dropout(0.3))

    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D((2, 2)))
    model.add(Dropout(0.3))

    model.add(Flatten())
    model.add(Dense(512, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))

    model.add(Dense(1, activation='sigmoid'))

    model.compile(

optimizer=tf.keras.optimizers.legacy.Adam(learning_rate=0.0005),
        loss='binary_crossentropy',
        metrics=[
            'accuracy',
            tf.keras.metrics.AUC(name='auc')
        ]
    )
    return model

model = build_cnn_model()
model.summary()
```

```
2025-04-26 18:41:16.120443: I
metal_plugin/src/device/metal_device.cc:1154] Metal device set to:
Apple M1
2025-04-26 18:41:16.120564: I
metal_plugin/src/device/metal_device.cc:296] systemMemory: 16.00 GB
2025-04-26 18:41:16.120571: I
metal_plugin/src/device/metal_device.cc:313] maxCacheSize: 5.33 GB
2025-04-26 18:41:16.120654: I
tensorflow/core/common_runtime/pluggable_device/pluggable_device_facto
ry.cc:303] Could not identify NUMA node of platform GPU ID 0,
defaulting to 0. Your kernel may not have been built with NUMA
support.
2025-04-26 18:41:16.120674: I
tensorflow/core/common_runtime/pluggable_device/pluggable_device_facto
ry.cc:269] Created TensorFlow device
(/job:localhost/replica:0/task:0/device:GPU:0 with 0 MB memory) ->
physical PluggableDevice (device: 0, name: METAL, pci bus id:
<undefined>)

Model: "Histopathologic_CNN"
_________________________________________________________________
```

```
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d (Conv2D)              (None, 96, 96, 32)        896

 batch_normalization (Batch   (None, 96, 96, 32)        128
 Normalization)

 max_pooling2d (MaxPooling2   (None, 48, 48, 32)        0
 D)

 conv2d_1 (Conv2D)            (None, 48, 48, 64)        18496

 batch_normalization_1 (Bat   (None, 48, 48, 64)        256
 chNormalization)

 max_pooling2d_1 (MaxPoolin   (None, 24, 24, 64)        0
 g2D)

 dropout (Dropout)            (None, 24, 24, 64)        0

 conv2d_2 (Conv2D)            (None, 24, 24, 128)       73856

 batch_normalization_2 (Bat   (None, 24, 24, 128)       512
 chNormalization)

 max_pooling2d_2 (MaxPoolin   (None, 12, 12, 128)       0
 g2D)

 dropout_1 (Dropout)          (None, 12, 12, 128)       0

 conv2d_3 (Conv2D)            (None, 12, 12, 256)       295168

 batch_normalization_3 (Bat   (None, 12, 12, 256)       1024
 chNormalization)

 max_pooling2d_3 (MaxPoolin   (None, 6, 6, 256)         0
 g2D)

 dropout_2 (Dropout)          (None, 6, 6, 256)         0

 flatten (Flatten)            (None, 9216)              0

 dense (Dense)                (None, 512)               4719104

 batch_normalization_4 (Bat   (None, 512)               2048
 chNormalization)

 dropout_3 (Dropout)          (None, 512)               0
```

```
 dense_1 (Dense)             (None, 1)                 513

=================================================================
Total params: 5112001 (19.50 MB)
Trainable params: 5110017 (19.49 MB)
Non-trainable params: 1984 (7.75 KB)
_________________________________________________________________
```

5. Model Training

```
MODEL_CHECKPOINT_PATH = 'best_cancer_cnn_model.keras'

steps_per_epoch = np.ceil(train_generator.samples /
BATCH_SIZE).astype(int)
validation_steps = np.ceil(validation_generator.samples /
BATCH_SIZE).astype(int)

print(f"\nSteps per epoch: {steps_per_epoch}")
print(f"Validation steps: {validation_steps}")
```

```
Steps per epoch: 2751
Validation steps: 688
```

```
history = model.fit(
    train_generator,
    steps_per_epoch=steps_per_epoch,
    epochs=25,
    validation_data=validation_generator,
    validation_steps=validation_steps,
    verbose=1
)
```

```
Epoch 1/25

2025-04-26 18:41:16.926448: I
tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc
:114] Plugin optimizer for device_type GPU is enabled.

2751/2751 [==============================] - ETA: 0s - loss: 0.4110 -
accuracy: 0.8209 - auc: 0.8901

2025-04-26 18:46:53.307065: I
tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc
:114] Plugin optimizer for device_type GPU is enabled.

2751/2751 [==============================] - 360s 131ms/step - loss:
0.4110 - accuracy: 0.8209 - auc: 0.8901 - val_loss: 0.5383 -
val_accuracy: 0.7527 - val_auc: 0.9247
Epoch 2/25
2751/2751 [==============================] - 353s 128ms/step - loss:
```

```
0.3333 - accuracy: 0.8575 - auc: 0.9284 - val_loss: 0.5576 -
val_accuracy: 0.7734 - val_auc: 0.8528
Epoch 3/25
2751/2751 [==============================] - 357s 130ms/step - loss:
0.2987 - accuracy: 0.8755 - auc: 0.9426 - val_loss: 1.2485 -
val_accuracy: 0.5859 - val_auc: 0.6208
Epoch 4/25
2751/2751 [==============================] - 364s 132ms/step - loss:
0.2786 - accuracy: 0.8852 - auc: 0.9499 - val_loss: 0.9418 -
val_accuracy: 0.6502 - val_auc: 0.9013
Epoch 5/25
2751/2751 [==============================] - 365s 133ms/step - loss:
0.2668 - accuracy: 0.8915 - auc: 0.9541 - val_loss: 1.2594 -
val_accuracy: 0.6465 - val_auc: 0.8162
Epoch 6/25
2751/2751 [==============================] - 377s 137ms/step - loss:
0.2561 - accuracy: 0.8962 - auc: 0.9575 - val_loss: 0.2891 -
val_accuracy: 0.8775 - val_auc: 0.9464
Epoch 7/25
2751/2751 [==============================] - 335s 122ms/step - loss:
0.2491 - accuracy: 0.8997 - auc: 0.9598 - val_loss: 0.4483 -
val_accuracy: 0.8204 - val_auc: 0.9082
Epoch 8/25
2751/2751 [==============================] - 338s 123ms/step - loss:
0.2390 - accuracy: 0.9046 - auc: 0.9630 - val_loss: 1.0112 -
val_accuracy: 0.6776 - val_auc: 0.8319
Epoch 9/25
2751/2751 [==============================] - 357s 130ms/step - loss:
0.2341 - accuracy: 0.9067 - auc: 0.9643 - val_loss: 0.6370 -
val_accuracy: 0.7497 - val_auc: 0.8852
Epoch 10/25
2751/2751 [==============================] - 367s 133ms/step - loss:
0.2300 - accuracy: 0.9086 - auc: 0.9655 - val_loss: 0.7115 -
val_accuracy: 0.7381 - val_auc: 0.8026
Epoch 11/25
2751/2751 [==============================] - 366s 133ms/step - loss:
0.2267 - accuracy: 0.9103 - auc: 0.9666 - val_loss: 0.8878 -
val_accuracy: 0.7301 - val_auc: 0.7899
Epoch 12/25
2751/2751 [==============================] - 359s 131ms/step - loss:
0.2244 - accuracy: 0.9112 - auc: 0.9671 - val_loss: 0.5028 -
val_accuracy: 0.7758 - val_auc: 0.9089
Epoch 13/25
2751/2751 [==============================] - 366s 133ms/step - loss:
0.2213 - accuracy: 0.9119 - auc: 0.9682 - val_loss: 0.4475 -
val_accuracy: 0.8190 - val_auc: 0.9297
Epoch 14/25
2751/2751 [==============================] - 362s 131ms/step - loss:
0.2195 - accuracy: 0.9135 - auc: 0.9686 - val_loss: 0.3826 -
```

```
val_accuracy: 0.8253 - val_auc: 0.9612
Epoch 15/25
2751/2751 [==============================] - 358s 130ms/step - loss:
0.2175 - accuracy: 0.9138 - auc: 0.9692 - val_loss: 0.9218 -
val_accuracy: 0.6750 - val_auc: 0.8204
Epoch 16/25
2751/2751 [==============================] - 366s 133ms/step - loss:
0.2141 - accuracy: 0.9151 - auc: 0.9701 - val_loss: 0.7625 -
val_accuracy: 0.7493 - val_auc: 0.8595
Epoch 17/25
2751/2751 [==============================] - 355s 129ms/step - loss:
0.2117 - accuracy: 0.9168 - auc: 0.9709 - val_loss: 0.4873 -
val_accuracy: 0.8055 - val_auc: 0.9098
Epoch 18/25
2751/2751 [==============================] - 365s 133ms/step - loss:
0.2116 - accuracy: 0.9166 - auc: 0.9707 - val_loss: 1.1007 -
val_accuracy: 0.6591 - val_auc: 0.8375
Epoch 19/25
2751/2751 [==============================] - 375s 136ms/step - loss:
0.2104 - accuracy: 0.9169 - auc: 0.9711 - val_loss: 0.7748 -
val_accuracy: 0.7254 - val_auc: 0.8883
Epoch 20/25
2751/2751 [==============================] - 367s 133ms/step - loss:
0.2144 - accuracy: 0.9164 - auc: 0.9699 - val_loss: 1.2249 -
val_accuracy: 0.6535 - val_auc: 0.7880
Epoch 21/25
2751/2751 [==============================] - 385s 140ms/step - loss:
0.2172 - accuracy: 0.9145 - auc: 0.9693 - val_loss: 0.4517 -
val_accuracy: 0.8169 - val_auc: 0.9380
Epoch 22/25
2751/2751 [==============================] - 435s 158ms/step - loss:
0.2135 - accuracy: 0.9150 - auc: 0.9703 - val_loss: 0.9816 -
val_accuracy: 0.6675 - val_auc: 0.8721
Epoch 23/25
2751/2751 [==============================] - 476s 173ms/step - loss:
0.2106 - accuracy: 0.9171 - auc: 0.9712 - val_loss: 1.2788 -
val_accuracy: 0.6373 - val_auc: 0.8401
Epoch 24/25
2751/2751 [==============================] - 445s 162ms/step - loss:
0.2085 - accuracy: 0.9181 - auc: 0.9717 - val_loss: 0.4570 -
val_accuracy: 0.8042 - val_auc: 0.9551
Epoch 25/25
2751/2751 [==============================] - 434s 158ms/step - loss:
0.2079 - accuracy: 0.9177 - auc: 0.9718 - val_loss: 0.3407 -
val_accuracy: 0.8589 - val_auc: 0.9618
```

6. Model Evaluation

```
# model.load_weights(MODEL_CHECKPOINT_PATH)

validation_generator.reset()
loss, accuracy, auc_score = model.evaluate(validation_generator,
steps=validation_steps, verbose=1)

print(f"\nValidation Loss: {loss:.4f}")
print(f"Validation Accuracy: {accuracy:.4f}")
print(f"Validation AUC: {auc_score:.4f}")
```

```
688/688 [==============================] - 22s 32ms/step - loss:
0.3407 - accuracy: 0.8589 - auc: 0.9618

Validation Loss: 0.3407
Validation Accuracy: 0.8589
Validation AUC: 0.9618
```

```
validation_generator.reset()
y_pred_proba = model.predict(validation_generator,
steps=validation_steps, verbose=1)
y_pred = (y_pred_proba > 0.5).astype(int).flatten()

y_true = validation_generator.classes

if len(y_pred) != len(y_true):
    y_pred = y_pred[:len(y_true)]
    y_pred_proba = y_pred_proba[:len(y_true)]
```

```
  2/688 [..............................] - ETA: 44s

2025-04-26 21:25:38.617134: I
tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc
:114] Plugin optimizer for device_type GPU is enabled.

688/688 [==============================] - 22s 32ms/step
```

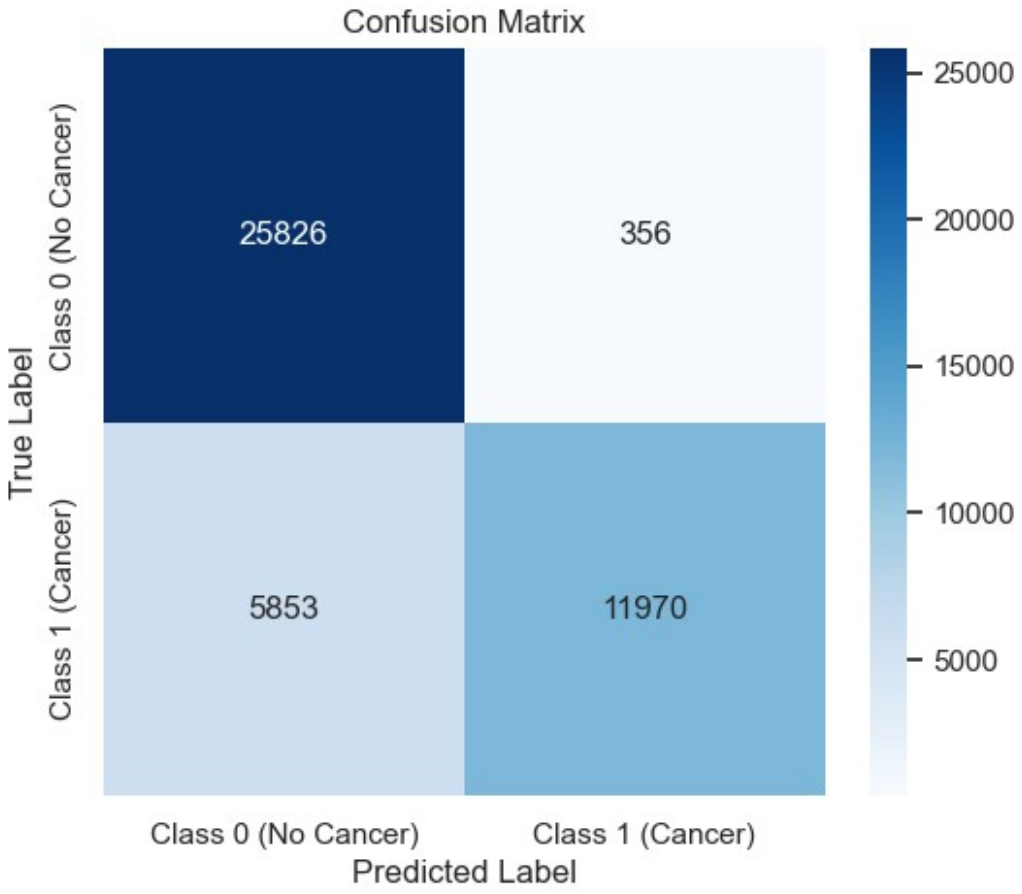
```
print("\nClassification Report:")
target_names = ['Class 0 (No Cancer)', 'Class 1 (Cancer)']
print(classification_report(y_true, y_pred,
target_names=target_names))
```

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

Class 0 (No Cancer)       0.82      0.99      0.89     26182
   Class 1 (Cancer)       0.97      0.67      0.79     17823

           accuracy                           0.86     44005
          macro avg       0.89      0.83      0.84     44005
```

```
       weighted avg       0.88      0.86      0.85     44005
```

```
print("\nConfusion Matrix:")
cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=target_names, yticklabels=target_names)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```

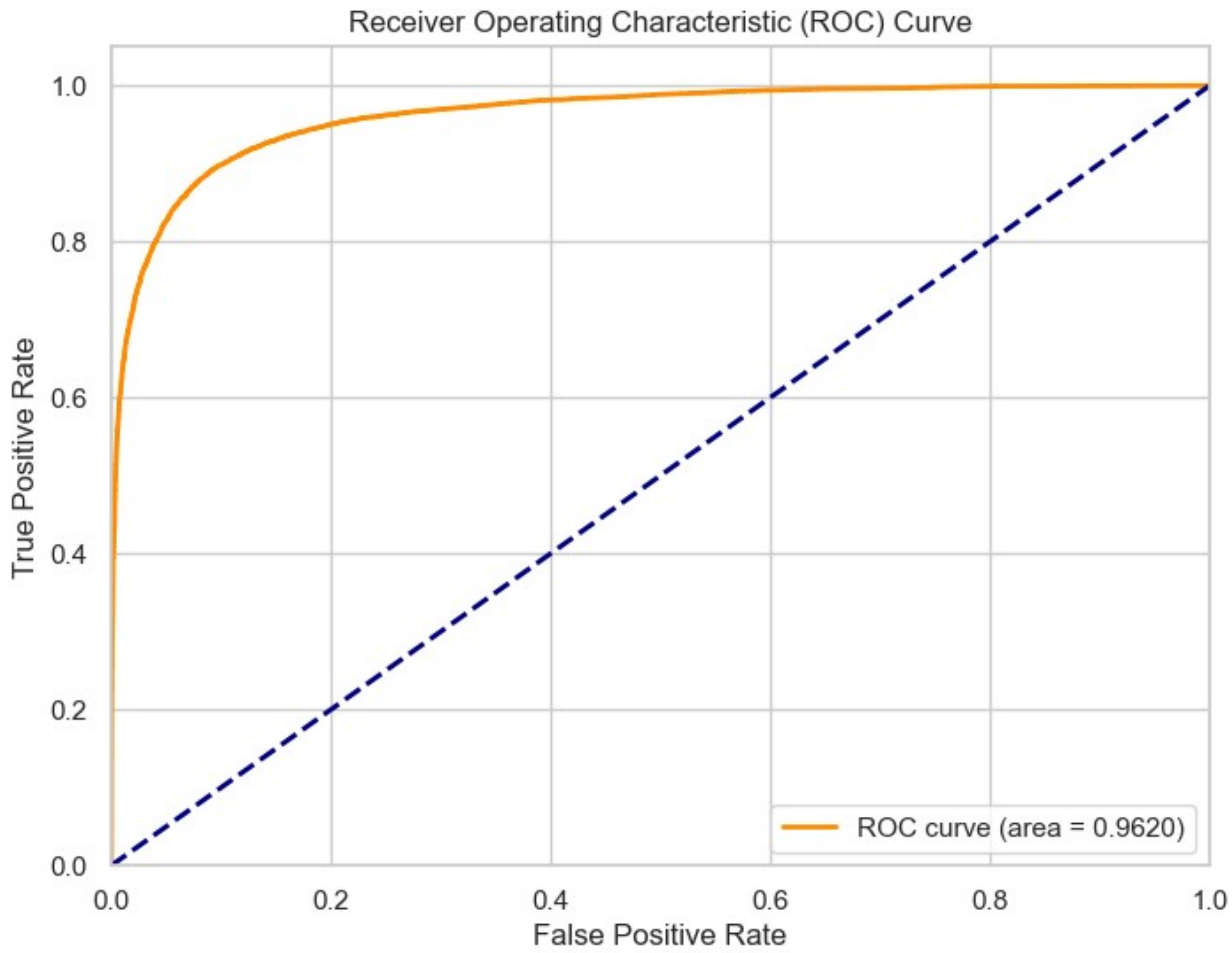
```
Confusion Matrix:
```




```
print("\nROC Curve:")
fpr, tpr, thresholds = roc_curve(y_true, y_pred_proba)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area =
```

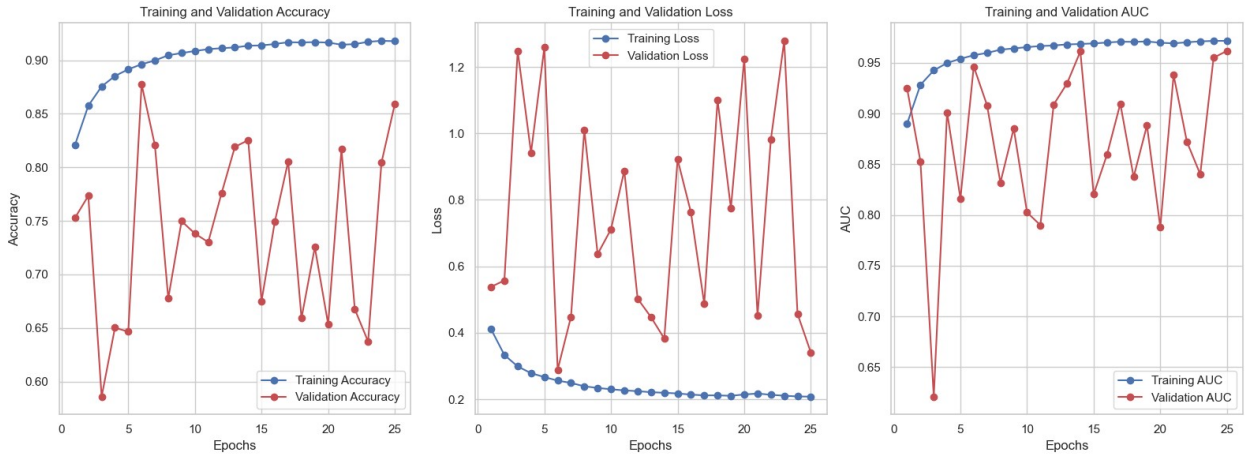
```
{roc_auc:.4f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```

ROC Curve:




```
def plot_history(history):
    """Plots training & validation accuracy and loss."""
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    auc = history.history['auc']
```

```
    val_auc = history.history['val_auc']
    epochs = range(1, len(acc) + 1)

    plt.figure(figsize=(16, 6))

    plt.subplot(1, 3, 1)
    plt.plot(epochs, acc, 'bo-', label='Training Accuracy')
    plt.plot(epochs, val_acc, 'ro-', label='Validation Accuracy')
    plt.title('Training and Validation Accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()

    plt.subplot(1, 3, 2)
    plt.plot(epochs, loss, 'bo-', label='Training Loss')
    plt.plot(epochs, val_loss, 'ro-', label='Validation Loss')
    plt.title('Training and Validation Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()

    plt.subplot(1, 3, 3)
    plt.plot(epochs, auc, 'bo-', label='Training AUC')
    plt.plot(epochs, val_auc, 'ro-', label='Validation AUC')
    plt.title('Training and Validation AUC')
    plt.xlabel('Epochs')
    plt.ylabel('AUC')
    plt.legend()

    plt.tight_layout()
    plt.show()
    return

plot_history(history)
```



7. Prediction on Test Set

```
test_datagen = ImageDataGenerator(rescale=1./255.)

test_files = os.listdir(TEST_DIR)
test_df = pd.DataFrame({'id': test_files})

test_generator = test_datagen.flow_from_dataframe(
    dataframe=test_df,
    directory=TEST_DIR,
    x_col='id',
    y_col=None,
    target_size=(IMG_WIDTH, IMG_HEIGHT),
    batch_size=BATCH_SIZE,
    class_mode=None,
    shuffle=False
)
```

```
Found 57458 validated image filenames.
```

```
test_steps = np.ceil(test_generator.samples / BATCH_SIZE).astype(int)
test_pred_proba = model.predict(test_generator, steps=test_steps,
verbose=1)
```

```
  5/898 [..............................] - ETA: 14s

2025-04-26 21:26:02.991030: I
tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc
:114] Plugin optimizer for device_type GPU is enabled.

898/898 [==============================] - 20s 22ms/step
```

```
submission_df = pd.DataFrame({
    'id': [os.path.splitext(f)[0] for f in test_generator.filenames],
    'label': test_pred_proba.flatten().round()
})

if len(submission_df) != len(test_df):
    print(f"Warning: Submission length ({len(submission_df)}) doesn't
match test set length ({len(test_df)}).")
else:
    SUBMISSION_PATH =
"~/histopathologic-cancer-detection/submission.csv"
    submission_df.to_csv(SUBMISSION_PATH, index=False)
```