In [ ]:
```
pip install tensorflow pandas matplotlib seaborn scikit-learn
```

In [ ]:
```
# Step 1: Setup and Import Libraries
!pip install kaggle tensorflow numpy pandas matplotlib scikit-learn

import os
import zipfile
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion_matrix, classification_report

# Step 2: Download and Extract Dataset from Kaggle
os.environ['KAGGLE_CONFIG_DIR'] = '/content/'

!kaggle competitions download -c histopathologic-cancer-detection

with zipfile.ZipFile('/content/histopathologic-cancer-detection.zip', 'r') as zi
    zip_ref.extractall('/content/')

# Step 3: Load and Explore Dataset
train_labels = pd.read_csv('/content/train_labels.csv')
print(train_labels.head())

print("Total images:", len(train_labels))
print("Class distribution:", train_labels['label'].value_counts())

# Visualize Sample Images
def display_images(folder_path, image_ids, labels, num_images=5):
    plt.figure(figsize=(10, 10))
    for i, img_id in enumerate(image_ids[:num_images]):
        img_path = os.path.join(folder_path, f"{img_id}.tif")
        img = plt.imread(img_path)
        plt.subplot(1, num_images, i + 1)
        plt.imshow(img)
        plt.title(f"Label: {labels[i]}")
        plt.axis('off')
    plt.show()

display_images('/content/train', train_labels['id'], train_labels['label'])

# Step 4: Preprocess Data
IMG_SIZE = 96
BATCH_SIZE = 32

train_datagen = ImageDataGenerator(
    rescale=1.0/255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.2
)
```

```
train_generator = train_datagen.flow_from_dataframe(
    dataframe=train_labels,
    directory='/content/train/',
    x_col='id',
    y_col='label',
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='training'
)

val_generator = train_datagen.flow_from_dataframe(
    dataframe=train_labels,
    directory='/content/train/',
    x_col='id',
    y_col='label',
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='validation'
)

# Step 5: Build CNN Model
def build_cnn(input_shape=(IMG_SIZE, IMG_SIZE, 3)):
    model = models.Sequential()

    model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_sha
    model.add(layers.MaxPooling2D((2, 2)))

    model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))

    model.add(layers.Conv2D(128, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))

    model.add(layers.Flatten())
    model.add(layers.Dense(256, activation='relu'))
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(1, activation='sigmoid'))

    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accura
    return model

cnn_model = build_cnn()
cnn_model.summary()

# Step 6: Train the Model
EPOCHS = 20
history = cnn_model.fit(train_generator, validation_data=val_generator, epochs=E

# Step 7: Evaluate the Model
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.show()

# Step 8: Predict on Test Data
test_images = [f for f in os.listdir('/content/test/') if f.endswith('.tif')]
```

```
# Prepare test generator
test_datagen = ImageDataGenerator(rescale=1.0/255)
test_generator = test_datagen.flow_from_dataframe(
    dataframe=pd.DataFrame(test_images, columns=['id']),
    directory='/content/test/',
    x_col='id',
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=1,
    class_mode=None,
    shuffle=False
)

predictions = cnn_model.predict(test_generator)

# Step 9: Save Predictions
submission = pd.DataFrame({
    'id': [img.split('.')[0] for img in test_images],
    'label': predictions.flatten()
})
submission['label'] = (submission['label'] > 0.5).astype(int)
submission.to_csv('submission.csv', index=False)

print("Submission file saved successfully!")
```

In [ ]: