DA 1

BINARY CLASSIFICATION OF BREAST CANCER

Ankit 23BAI1379 Amlan Sarkar 23BAI1247 Kanishq Tiwari 23BAI1150 | Artificial Intelligence
BCSE306L | January 24, 2025

https://github.com/Amlan131/DA_1

CODE

This code imports essential libraries for image-based machine learning tasks:

```
import numpy as np
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
```

This code configures TensorFlow to optimize GPU usage:

```
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)
```

This code preprocesses the training images for use in a deep learning model:

Loads and preprocesses images from the specified directory (train) for training.

Automatically assigns labels based on folder names and converts them to a one-hot encoded format.

Resized to 128x128 pixels

Batched in groups of 4

Shuffled for randomness in training

```
training_set = tf.keras.utils.image_dataset_from_directory(
 r"D:\01 STUDY MATERIAL\ai project\Breast-Splitted\train",
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=4,
    image_size=(128, 128),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)
class_name = training_set.class_names
print(class_name)
```

Found 4745 files belonging to 2 classes.

```
['benign', 'malignant']

validation_set = tf.keras.utils.image_dataset_from_directory(
  r"D:\01 STUDY MATERIAL\ai project\Breast-Splitted\val",
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=4,
    image_size=(128, 128),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)
class_name = validation_set.class_names
print(class_name)
Found 1581 files belonging to 2 classes.
['benign', 'malignant']
```

This code defines and compiles a Convolutional Neural Network (CNN) for binary image classification:

```
cnn = tf.keras.models.Sequential()
cnn.add(tf.keras.layers.Conv2D(filters=32,kernel_size=3,padding='same',activation='relu',input_sha
pe=[128,128,3]))
cnn.add(tf.keras.layers.Conv2D(filters=32,kernel_size=3,activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))
cnn.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,padding='same',activation='relu'))
cnn.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))
cnn.add(tf.keras.layers.Conv2D(filters=128,kernel_size=3,padding='same',activation='relu'))
cnn.add(tf.keras.layers.Conv2D(filters=128,kernel_size=3,activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))
cnn.add(tf.keras.layers.Conv2D(filters=256,kernel_size=3,padding='same',activation='relu'))
cnn.add(tf.keras.layers.Conv2D(filters=256,kernel_size=3,activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))
cnn.add(tf.keras.layers.Dropout(0.25))
cnn.add(tf.keras.layers.Dense(units=1500,activation='relu'))
cnn.add(tf.keras.layers.Dropout(0.4))
cnn.add(tf.keras.layers.Dense(units=2,activation='sigmoid'))
cnn.compile(optimizer=tf.keras.optimizers.legacy.Adam(
   learning_rate=0.0001),loss='binary_crossentropy',metrics=['accuracy'])
cnn.summary()
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
```

```
conv2d (Conv2D)              (None, 128, 128, 32)      896

conv2d_1 (Conv2D)            (None, 126, 126, 32)      9248

max_pooling2d (MaxPooling2D  (None, 63, 63, 32)       0
)

conv2d_2 (Conv2D)            (None, 63, 63, 64)        18496

conv2d_3 (Conv2D)            (None, 61, 61, 64)        36928

max_pooling2d_1 (MaxPooling  (None, 30, 30, 64)       0
2D)

conv2d_4 (Conv2D)            (None, 30, 30, 128)       73856

conv2d_5 (Conv2D)            (None, 28, 28, 128)       147584

max_pooling2d_2 (MaxPooling  (None, 14, 14, 128)      0
2D)

conv2d_6 (Conv2D)            (None, 14, 14, 256)       295168

conv2d_7 (Conv2D)            (None, 12, 12, 256)       590080

max_pooling2d_3 (MaxPooling  (None, 6, 6, 256)        0
2D)

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

flatten (Flatten)            (None, 9216)              0

dense (Dense)                (None, 1500)              13825500

dropout_1 (Dropout)          (None, 1500)              0

dense_1 (Dense)              (None, 2)                 3002

=================================================================
Total params: 15,000,758
Trainable params: 15,000,758
Non-trainable params: 0
```

Trains the CNN model for 10 epochs using the training and validation datasets, storing the training history.

```
training_history = cnn.fit(x=training_set,validation_data=validation_set,epochs=10)
Epoch 1/10
1187/1187 [==============================] - 23s 14ms/step - loss: 0.5513 - accuracy: 0.7583 -
val_loss: 0.5055 - val_accuracy: 0.8027
Epoch 2/10
```

```
1187/1187 [==============================] - 16s 14ms/step - loss: 0.4748 - accuracy: 0.8063 -
val_loss: 0.4729 - val_accuracy: 0.8109
Epoch 3/10
1187/1187 [==============================] - 16s 14ms/step - loss: 0.4420 - accuracy: 0.8333 -
val_loss: 0.4960 - val_accuracy: 0.8242
Epoch 4/10
1187/1187 [==============================] - 16s 14ms/step - loss: 0.3989 - accuracy: 0.8548 -
val_loss: 0.3815 - val_accuracy: 0.8564
Epoch 5/10
1187/1187 [==============================] - 16s 14ms/step - loss: 0.3885 - accuracy: 0.8626 -
val_loss: 0.3614 - val_accuracy: 0.8703
Epoch 6/10
1187/1187 [==============================] - 16s 14ms/step - loss: 0.3794 - accuracy: 0.8660 -
val_loss: 0.3514 - val_accuracy: 0.8722
Epoch 7/10
1187/1187 [==============================] - 16s 14ms/step - loss: 0.3492 - accuracy: 0.8717 -
val_loss: 0.4022 - val_accuracy: 0.8665
Epoch 8/10
1187/1187 [==============================] - 16s 14ms/step - loss: 0.3353 - accuracy: 0.8717 -
val_loss: 0.3446 - val_accuracy: 0.8760
Epoch 9/10
1187/1187 [==============================] - 16s 14ms/step - loss: 0.3131 - accuracy: 0.8843 -
val_loss: 0.3404 - val_accuracy: 0.8697
Epoch 10/10
1187/1187 [==============================] - 16s 14ms/step - loss: 0.2955 - accuracy: 0.8839 -
val_loss: 0.3230 - val_accuracy: 0.8748
```

Evaluates the CNN on the training set and prints the training accuracy.

```
train_loss, train_acc = cnn.evaluate(training_set)
print('Training accuracy:', train_acc)
1187/1187 [==============================] - 6s 5ms/step - loss: 0.2486 - accuracy: 0.9041
Training accuracy: 0.9041095972061157
```

Evaluates the CNN on the validation set and prints the validation accuracy.

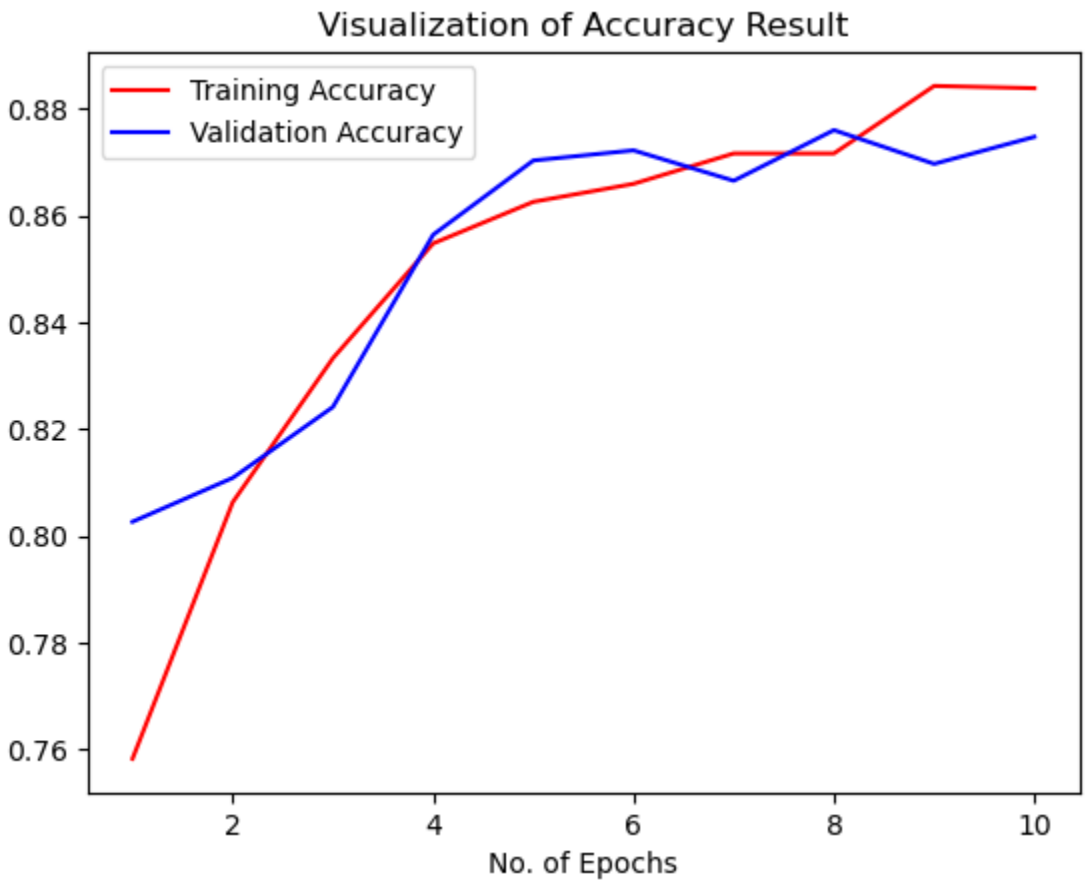
```
val_loss, val_acc = cnn.evaluate(validation_set)
print('Validation accuracy:', val_acc)
396/396 [==============================] - 2s 5ms/step - loss: 0.3230 - accuracy: 0.8748
Validation accuracy: 0.8747628331184387
```

Saves the trained CNN model to a file named trnew_cancer_modelv1.0.keras.

```
cnn.save('trnew_cancer_modelv1.0.keras')
training_history.history
{'loss': [0.5512648820877075,
  0.4747887849807739,
  0.4420156478881836,
  0.39887040853500366,
```

```
  0.388454407453537,
  0.3794495761394501,
  0.3492189943790436,
  0.33529233932495117,
  0.3130831718444824,
  0.29554569721221924],
 'accuracy': [0.7582718729972839,
  0.806322455406189,
  0.8332982063293457,
  0.8547945022583008,
  0.8625922203063965,
  0.8659641742706299,
  0.8716543912887573,
  0.8716543912887573,
  0.8842992782592773,
  0.8838777542114258],
 'val_loss': [0.5055041909217834,
  0.47289153933525085,
  0.49602824449539185,
  0.3814983665943146,
  0.3614308536052704,
  0.35138577222824097,
  0.4021930992603302,
  0.344621866941452,
  0.3404453694820404,
  0.32300633192062238],
 'val_accuracy': [0.8026565313339233,
  0.8108791708946228,
  0.824161946773529,
  0.8564199805259705,
  0.8703522212905884,
  0.8722327351570129,
  0.8665401935577393,
  0.8760278224945068,
  0.8697026968002319,
  0.8747628331184387]}
import json
with open('training_hist_cancer.json','w') as f:
  json.dump(training_history.history,f)
print(training_history.history.keys())
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])

epochs = [i for i in range(1,11)]
plt.plot(epochs,training_history.history['accuracy'],color='red',label='Training Accuracy')
plt.plot(epochs,training_history.history['val_accuracy'],color='blue',label='Validation Accuracy')
plt.xlabel('No. of Epochs')
plt.title('Visualization of Accuracy Result')
plt.legend()
plt.show()
```




```
class_name = validation_set.class_names
test_set = tf.keras.utils.image_dataset_from_directory(
  r"D:\01 STUDY MATERIAL\ai project\Breast-Splitted\test",
   labels="inferred",
   label_mode="categorical",
   class_names=None,
   color_mode="rgb",
   batch_size=1,
   image_size=(128, 128),
   shuffle=False,
   seed=None,
   validation_split=None,
   subset=None,
   interpolation="bilinear",
   follow_links=False,
   crop_to_aspect_ratio=False
)
```

Found 1583 files belonging to 2 classes.

```
y_pred = cnn.predict(test_set)
predicted_categories = tf.argmax(y_pred, axis=1)
```

1583/1583 [==============================] - 5s 3ms/step

```
true_categories = tf.concat([y for x, y in test_set], axis=0)
Y_true = tf.argmax(true_categories, axis=1)
Y_true
<tf.Tensor: shape=(1583,), dtype=int64, numpy=array([0, 0, 0, ..., 1, 1, 1], dtype=int64)>

from sklearn.metrics import confusion_matrix,classification_report
cm = confusion_matrix(Y_true,predicted_categories)
sklearn.metrics.roc_auc_score(y_true, y_score, *, average='macro', sample_weight=None,
max_fpr=None, multi_class='raise', labels=None)
Cell In[35], line 1
   sklearn.metrics.roc_auc_score(y_true, y_score, *, average='macro', sample_weight=None,
max_fpr=None, multi_class='raise', labels=None)
                         ^
SyntaxError: iterable argument unpacking follows keyword argument unpacking
```

Prints the precision, recall, and F1-score for each class using the true labels (Y_true) and predicted categories (predicted_categories).

```
print(classification_report(Y_true,predicted_categories,target_names=class_name))
precision    recall  f1-score   support

     benign       0.78      0.79      0.78       496
  malignant       0.90      0.90      0.90      1087

   accuracy                           0.86      1583
  macro avg       0.84      0.84      0.84      1583
weighted avg      0.86      0.86      0.86      1583

import seaborn as sns
plt.figure(figsize=(40, 40))
sns.heatmap(cm,annot=True,annot_kws={"size": 10})
plt.xlabel('Predicted Class',fontsize = 20)
plt.ylabel('Actual Class',fontsize = 20)
plt.title('Plant Disease Prediction Confusion Matrix',fontsize = 25)
plt.show()
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[1], line 2
      1 import seaborn as sns
----> 2 plt.figure(figsize=(40, 40))
      3 sns.heatmap(cm,annot=True,annot_kws={"size": 10})
      5 plt.xlabel('Predicted Class',fontsize = 20)

NameError: name 'plt' is not defined
TESTING
import numpy as np
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
cnn = tf.keras.models.load_model('trnew_cancer_model.keras')
```

```
import cv2
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import os
import random

# Define paths for benign and malignant folders
benign_path = r"D:\01 STUDY MATERIAL\ai project\Breast-Splitted\test\benign"
malignant_path = r"D:\01 STUDY MATERIAL\ai project\Breast-Splitted\test\malignant"

# Get all image file paths from both folders
benign_images = [os.path.join(benign_path, img) for img in os.listdir(benign_path) if
img.endswith(('.png', '.jpg', '.jpeg'))]
malignant_images = [os.path.join(malignant_path, img) for img in os.listdir(malignant_path) if
img.endswith(('.png', '.jpg', '.jpeg'))]

# Randomly select 9 images (combined from both classes)
selected_images = random.sample(benign_images, 4) + random.sample(malignant_images, 5)
random.shuffle(selected_images)  # Shuffle to mix benign and malignant images

# Predefined class names
class_name = ['Benign', 'Malignant']

# Load and process each image
plt.figure(figsize=(12, 12))
for idx, image_path in enumerate(selected_images):
    # Read and preprocess the image
    img = cv2.imread(image_path)
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  # Convert BGR to RGB for visualization

    # Determine the actual classification based on the folder name
    actual_label = 'Benign' if 'benign' in image_path.lower() else 'Malignant'

    # Resize and prepare for model prediction
    image = tf.keras.preprocessing.image.load_img(image_path, target_size=(128, 128))
    input_arr = tf.keras.preprocessing.image.img_to_array(image)
    input_arr = np.expand_dims(input_arr, axis=0)  # Convert single image to a batch

    # Make predictions
    predictions = cnn.predict(input_arr)
    result_index = np.argmax(predictions)  # Get the predicted class index
    predicted_label = class_name[result_index]  # Map index to class name

    # Plot the image with actual and predicted labels
    plt.subplot(3, 3, idx + 1)
    plt.imshow(img_rgb)
    plt.title(f"Actual: {actual_label}\nPredicted: {predicted_label}", color='green' if actual_label ==
predicted_label else 'red')
    plt.axis('off')
```

```
plt.tight_layout()
plt.show()
```

```
1/1 [==============================] - 6s 6s/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 12ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 10ms/step
1/1 [==============================] - 0s 10ms/step
1/1 [==============================] - 0s 20ms/step
```