Project Report

**Classification of Brain Tumors using CNN**

**Reg No :** 953621243037

**Name :** Muthuvel S

**AIM :**

The aim of this mini-project is to develop a Convolutional Neural Network (CNN) for the classification of brain tumor images obtained from MRI scans. The objective is to create a model that can accurately classify brain tumor images into different categories, such as glioma tumor, meningioma tumor, pituitary tumor, and no tumor.

**ABSTRACT:**

In this mini-project, we implement a CNN-based image classification model for brain tumor classification. We start by collecting and preprocessing the dataset, which consists of MRI images of different brain tumor types. After data preparation, we design a CNN architecture for the model. The model is trained on the training data and evaluated on a separate test dataset. The evaluation includes calculating accuracy, plotting loss and accuracy curves, and visualizing a confusion matrix. The project is implemented in Python using TensorFlow and other relevant libraries.

**PROCEDURE:**

1. **Data Collection:**
   * MRI images of brain tumors, including glioma, meningioma, pituitary, and no tumor, are collected.
   * The dataset is divided into training and testing sets.
2. **Data Preprocessing:**
   * Images are loaded and converted to NumPy arrays.
   * Data augmentation techniques like rescaling, shearing, horizontal flipping, and zooming are applied to the training data.
   * Images are resized to a common size (256x256) and converted to grayscale.
3. **Model Development:**
   * A Convolutional Neural Network (CNN) model is created using TensorFlow and Keras.
   * The model consists of convolutional layers, activation functions, max-pooling layers, fully connected layers, and dropout layers to prevent overfitting.
   * The output layer has neurons equal to the number of classes (tumor types) with a softmax activation function for classification.
4. **Training and Evaluation:**
   * The model is trained using the training dataset with a specified number of epochs.
   * Training and validation accuracy and loss are monitored and visualized during training.
   * After training, the model is evaluated on the test dataset.
   * Model performance metrics such as accuracy and a confusion matrix are generated to assess the model's classification performance.
5. **Tools and Libraries Used:**
   * Python: The programming language used for the project.
   * TensorFlow and Keras: Deep learning libraries for building and training neural networks.
   * NumPy: For numerical operations and array handling.
   * Pandas: For data manipulation and handling datasets.
   * Matplotlib: For data visualization.
   * ImageDataGenerator: A tool for image data augmentation.
   * Seaborn: For creating the confusion matrix visualization.
   * Google Colab: The environment for running the code.

**DATA COLLECTION:**

The dataset consists of MRI images of brain tumors, including glioma, meningioma, pituitary, and images without tumors (no tumor). The dataset is divided into training and testing sets.

**DATA PREPROCESSING:**

1. Images are loaded and converted to NumPy arrays using the **load\_img** and **img\_to\_array** functions.
2. Data augmentation is applied to the training dataset to increase its size and diversity.
3. All images are resized to a common size of 256x256 pixels and converted to grayscale to simplify processing.

**MODEL DEVELOPMENT:**

The CNN model consists of the following layers:

* Convolutional layers with ReLU activation functions.
* Max-pooling layers to downsample the feature maps.
* Fully connected (dense) layers with ReLU activation functions.
* Dropout layers to reduce overfitting.
* The output layer with softmax activation for classification.

The model is compiled with categorical cross-entropy loss and the Adam optimizer.

**TRAINING AND EVALUATION:**

The model is trained using the training dataset with a specified number of epochs. During training, training and validation accuracy and loss are monitored and plotted. After training, the model is evaluated on the test dataset to calculate accuracy. A confusion matrix is generated and visualized to assess the model's classification performance.

**IMPLEMENTATION:**

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.utils import img\_to\_array,load\_img

from glob import glob

from google.colab import drive

drive.mount('/content/gdrive')

train\_path="/content/gdrive/MyDrive/brain\_tumor/Training"

test\_path="/content/gdrive/MyDrive/brain\_tumor/Testing"

img=[load\_img(train\_path+'/glioma\_tumor/gg (514).jpg'),

load\_img(train\_path+'/meningioma\_tumor/m3 (121).jpg'),

load\_img(train\_path+'/no\_tumor/image(323).jpg'),

load\_img(train\_path+'/pituitary\_tumor/p (82).jpg')]

plt.figure(figsize=(14,14))

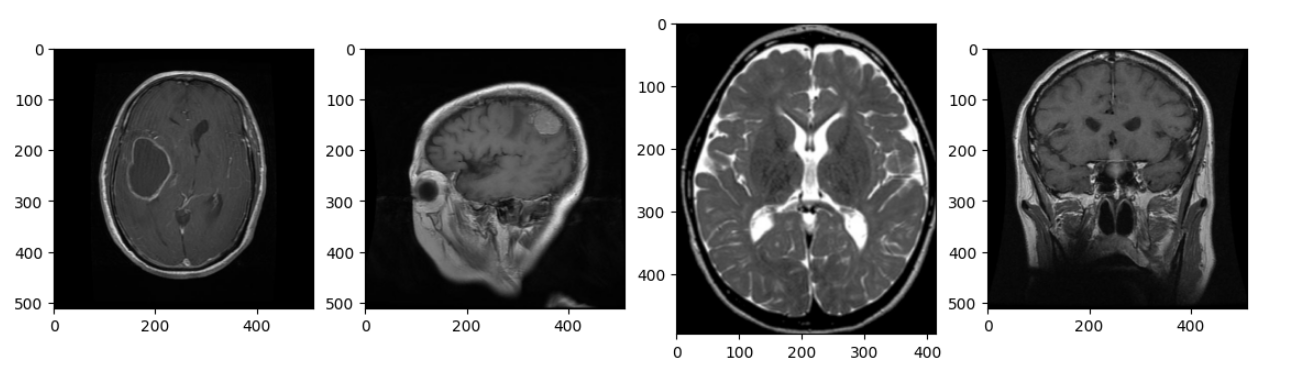
x, y = 1, 4

for i in range(4):

    plt.subplot(x, y, i+1)

plt.imshow(img[i], cmap='gray')

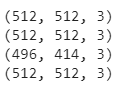
plt.show()



for i in range(4):

    x=img\_to\_array(img[i])

    print(x.shape)



#Class isimlerini ve sayısını bulduk.

className=glob(train\_path+"/\*")

numberOfClass=len(className)

print("number of class:",numberOfClass)



#Data Generation -Train-Test

train\_datagen=ImageDataGenerator(rescale=1./255,

                                shear\_range=0.3,

                                horizontal\_flip=True,

                                zoom\_range=0.3)

test\_datagen=ImageDataGenerator(rescale=1./255)

val\_datagen=ImageDataGenerator(rescale=1./255)

validation\_generator = val\_datagen.flow\_from\_directory(

    train\_path,

    target\_size=(256,256),

    batch\_size=32,

    class\_mode="categorical",

    color\_mode="grayscale"

)

train\_generator=train\_datagen.flow\_from\_directory(

        train\_path,

        target\_size=(256,256),

        batch\_size=32,

        class\_mode="categorical",

        color\_mode="grayscale")

test\_generator=test\_datagen.flow\_from\_directory(

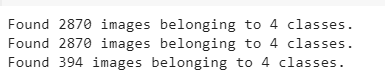
        test\_path,

        target\_size=(256,256),

        batch\_size=32,

        class\_mode="categorical",

        color\_mode="grayscale")



batch\_1\_img = train\_generator[0]

plt.figure(figsize=(14,14))

x, y = 3, 5

for i in range(15):

    img = batch\_1\_img[0][i]

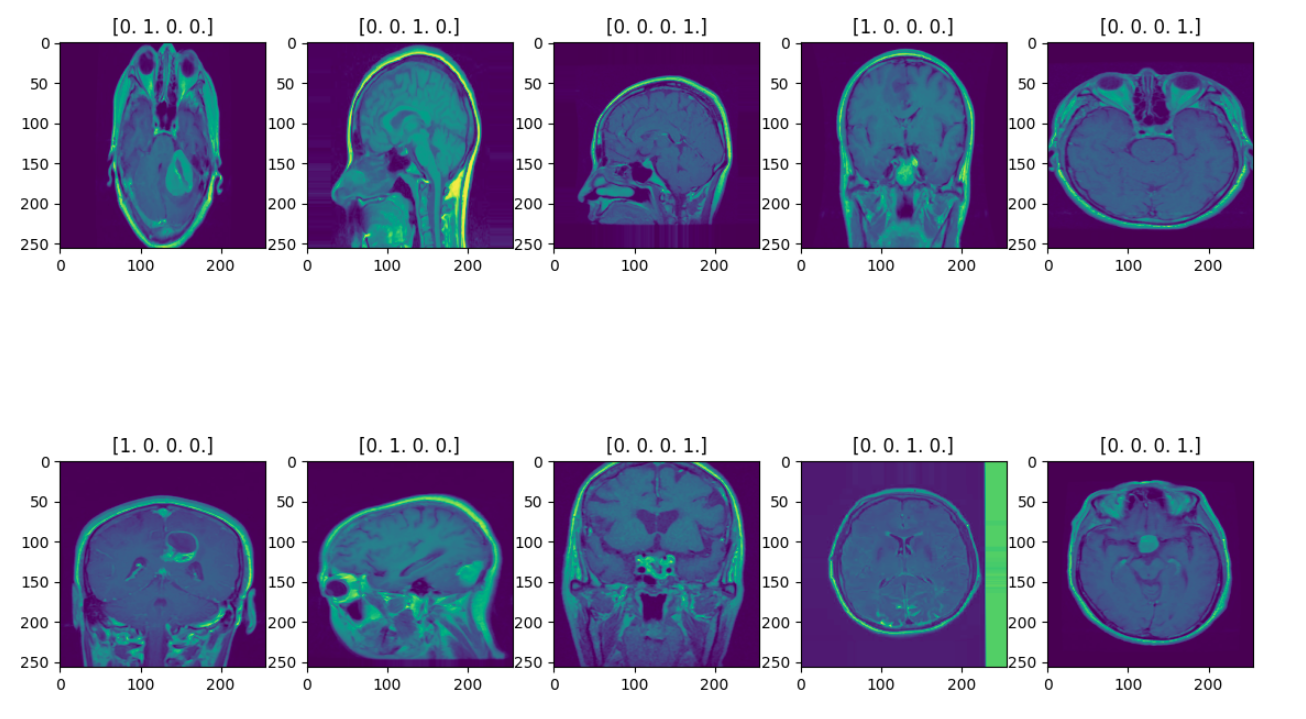
    lab = batch\_1\_img[1][i]

    plt.subplot(x, y, i+1)

    plt.imshow(img)

    plt.title(lab)

plt.show()



from tensorflow.keras.models import Sequential # initialize neural network library

from tensorflow.keras.layers import Dense,Conv2D,MaxPooling2D,Activation,Dropout,Flatten # build our layers library

# CNN MODEL

model=Sequential()

model.add(Conv2D(32,(3,3),input\_shape=(256,256,1)))

model.add(Activation("relu"))

model.add(MaxPooling2D())

model.add(Conv2D(32,(3,3)))

model.add(Activation("relu"))

model.add(MaxPooling2D())

model.add(Conv2D(64,(3,3)))

model.add(Activation("relu"))

model.add(MaxPooling2D())

model.add(Flatten())

model.add(Dense(64))

model.add(Activation("relu"))

model.add(Dropout(0.4))

model.add(Dense(64))

model.add(Activation("relu"))

model.add(Dropout(0.4))

model.add(Dense(numberOfClass))# output class sayısı kadar nöron

model.add(Activation("softmax"))

model.compile(loss="categorical\_crossentropy",

             optimizer="adam",

             metrics=["accuracy"])

hist=model.fit(

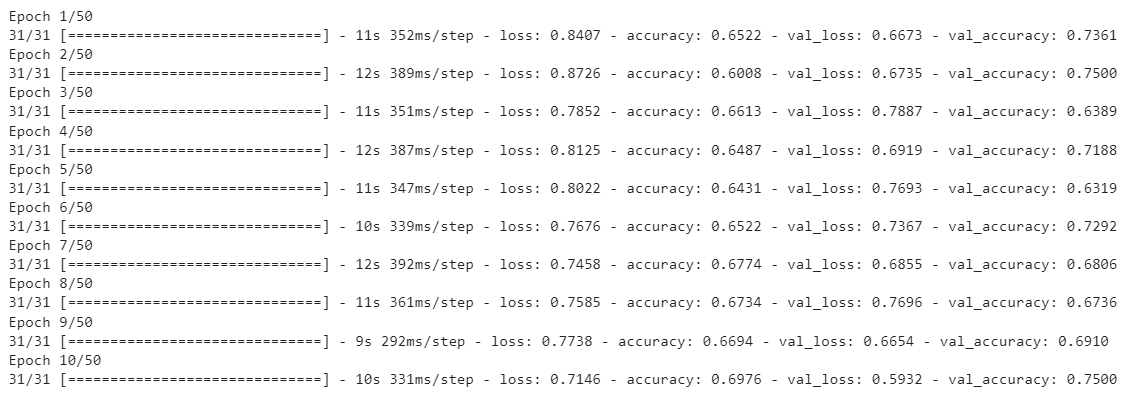
            train\_generator,

            steps\_per\_epoch=2581// 82,

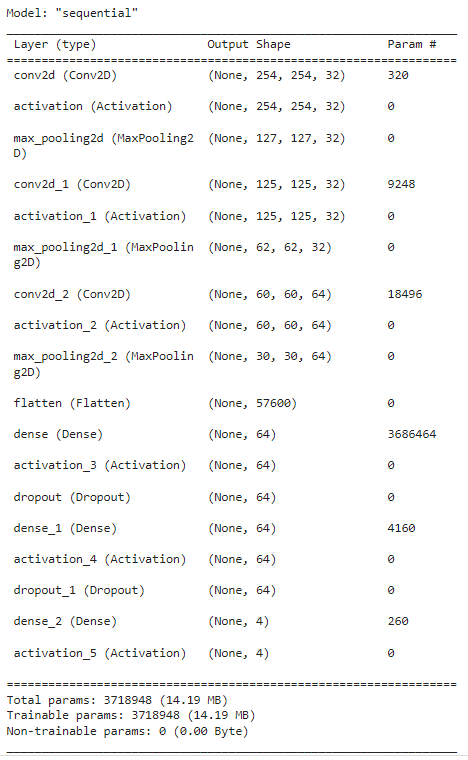
            epochs=50,

            validation\_data=validation\_generator,

            validation\_steps=289// 32)



model.summary()



#model evaluation

print(hist.history.keys())

plt.plot(hist.history["loss"],label="Train Loss")

plt.plot(hist.history["val\_loss"],label="Validation Loss")

plt.legend()

plt.show()

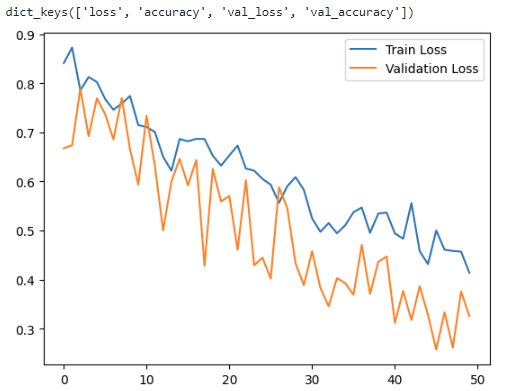
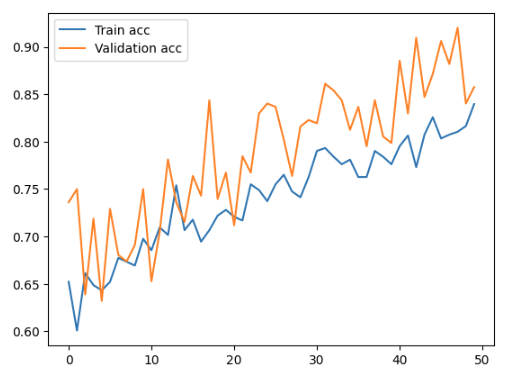
plt.figure()

plt.plot(hist.history["accuracy"],label="Train acc")

plt.plot(hist.history["val\_accuracy"],label="Validation acc")

plt.legend()

plt.show()



loss, accuracy = model.evaluate(test\_generator, steps=test\_generator.samples // 32)

print("Test Loss:", loss)

print("Test Accuracy:", accuracy)



Y\_true = []

num\_batches = len(test\_generator)

for i in range(num\_batches):

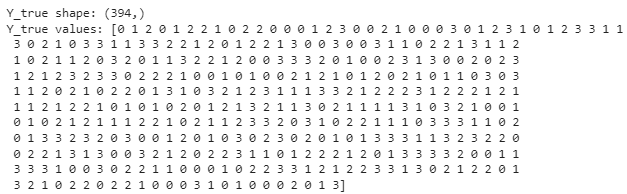
    \_, batch\_labels = test\_generator[i]

    Y\_true.extend(np.argmax(batch\_labels, axis=1))

Y\_true = np.array(Y\_true)

print("Y\_true shape:", Y\_true.shape)

print("Y\_true values:", Y\_true)



from sklearn.metrics import confusion\_matrix

import seaborn as sns

Y\_true = np.expand\_dims(Y\_true, axis=1)

Y\_pred = model.predict(test\_generator)

Y\_pred\_classes = np.argmax(Y\_pred, axis=1)

confusionMatrix = confusion\_matrix(Y\_true, Y\_pred\_classes)

# Şekil (figure) ve eksen (axis) nesnesini al

f, ax = plt.subplots(figsize=(10, 10))

# Isı haritasını çiz

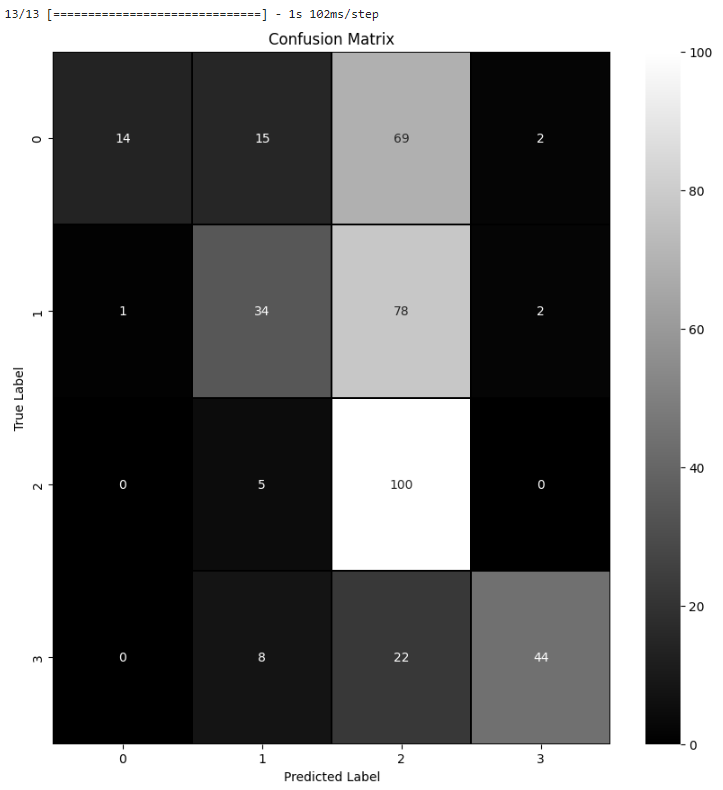
sns.heatmap(confusionMatrix, annot=True, linewidths=0.1, cmap="gist\_yarg\_r", linecolor="black", fmt='.0f', ax=ax)

plt.xlabel("Predicted Label")

plt.ylabel("True Label")

plt.title("Confusion Matrix")

plt.show()



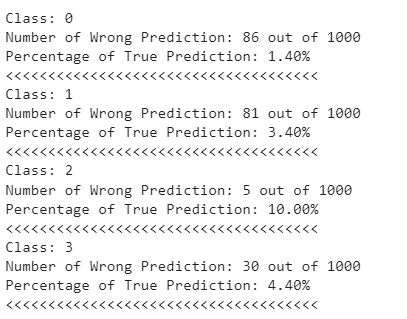
for i in range(len(confusionMatrix)):

    print("Class:",str(i))

    print("Number of Wrong Prediction:", str(sum(confusionMatrix[i])-confusionMatrix[i][i]), "out of 1000")

    print("Percentage of True Prediction: {:.2f}%".format(confusionMatrix[i][i] / 10))

    print("<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<")



**CONCLUSION:**

This mini-project successfully demonstrates the development of a Convolutional Neural Network (CNN) for brain tumor classification using MRI images. It encompasses data collection, preprocessing, model development, training, and evaluation, employing common deep learning tools and libraries. The project's goal was to create an accurate classification model for different brain tumor types, and it serves as a practical example for similar image classification tasks.