Brain Tumor Detection System

#3 View of Progress as of 3/18/2024

Abdullah Alshamrani
Saint Joseph’s University

Last View 2/19/2024

I have concluded that I had done the following:

- **Data Visualization and understanding**
- **Exploratory Data Analysis**
- **Preprocessing the data, (resizing, normalization and augmentation)**
- **Splitting data into training/testing sets**

Since 2/19/2024: Three weeks passed

Sine 2/4/2024, Three weeks have passed, So I am expected to finish all tasks of week 3-4 and a start of tasks in week 5-6: The following is a reminder of what week 3-4 includes of tasks and week 5-6 includes. After that what have been done will be shown. + Midterm report already submitted.

- Weeks 3-4: Data Preprocessing and Model Selection

- Tasks:

Preprocess the dataset for training, including resizing, normalization, and augmentation.

Split the dataset into training and testing sets.

Implement and train traditional machine learning models (SVM, Random Forest).

Begin the implementation of Convolutional Neural Networks (CNNs) for deep learning.

From Specification Paper

The reminding tasks of 3-4

- Implementing Traditional Machine learning models(SVM, Random Forest).

- Implementation of Convolutional Neural Networks(CNNs) for Deep Learning.

Implementing Traditional Machine learning models(SVM, Random Forest).

```
#Getting data ready for machine Learning
# Flatten the images for traditional ML models: 2D to 1D to use for ML models
X_train_flat = X_train.reshape((X_train.shape[0], -1))
X_test_flat = X_test.reshape((X_test.shape[0], -1))
# Converting string labels to integers
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)


# SVM Model
svm_model = svm.SVC(gamma='scale')
svm_model.fit(X_train_flat, y_train_encoded)
# Predictions
svm_predictions = svm_model.predict(X_test_flat)
svm_accuracy = accuracy_score(y_test_encoded, svm_predictions)
print(f'SVM Accuracy: {svm_accuracy}')
print(classification_report(y_test_encoded, svm_predictions, target_names=label_encoder.classes_))


# Random Forest Model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train_flat, y_train_encoded)
# Predictions
rf_predictions = rf_model.predict(X_test_flat)
rf_accuracy = accuracy_score(y_test_encoded, rf_predictions)
print(f'Random Forest Accuracy: {rf_accuracy}')
print(classification_report(y_test_encoded, rf_predictions, target_names=label_encoder.classes_))
```

gamma='scale': automatically calculated based on number of futures (dimention of each sample in the dataset)in data giving a fair chance to each future to contribute.

Best after many experimental trials

Random Forest with %96.83 Accuracy

%95.33

```
SVM Accuracy: 0.9533333333333334
              precision    recall  f1-score   support

          no       0.98      0.93      0.95       313
         yes       0.93      0.98      0.95       287

    accuracy                           0.95       600
   macro avg       0.95      0.95      0.95       600
weighted avg       0.95      0.95      0.95       600

Random Forest Accuracy: 0.9683333333333334
              precision    recall  f1-score   support

          no       0.99      0.95      0.97       313
         yes       0.94      0.99      0.97       287

    accuracy                           0.97       600
   macro avg       0.97      0.97      0.97       600
weighted avg       0.97      0.97      0.97       600
```

%96.83

Before moving to the CNN architecture, What is “Sequential” Model, provided by Keras.

As discussed last time, I am using Keras Library. Keras is an open source library written in python.

Sequential(): A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.[1]

CNNs: Building and analysing. First CNN.32

Shape of input Image

Cov2d layer with 32 filters(kernals), 32 futures being learned from the image. 3*3 is the learning size of the image. Each kernel will cover 3*3 area of the image being processed.

```python
# CNN architecture
cnn_model = Sequential()

# Convolutional layer: 32 filters, 3x3 kernel, activation 'relu'
cnn_model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 1)))
cnn_model.add(MaxPooling2D((2, 2)))
```

Relu activation function: non-linear function, output: Positive output OR 0 for negative output

Second CNN.64

```
# convolutional layer: 64 filters
cnn_model.add(Conv2D(64, (3, 3), activation='relu'))
cnn_model.add(MaxPooling2D((2, 2)))
```

Third CNN.128

```
# convolutional layer: 128 filters
cnn_model.add(Conv2D(128, (3, 3), activation='relu'))
cnn_model.add(MaxPooling2D((2, 2)))
```

Week 3-4 Tasks are Done

Week 5-6: Model Evaluation and web interface development

- Evaluate performance of traditional machine learning and CNN
- Select most accurate model
- Start the development of web interface
- Implement the backend functionality

```
cnn_model = Sequential()

# Convolutional layer with 32 filters
cnn_model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 1)))
cnn_model.add(MaxPooling2D((2, 2)))

# convolutional layer with 64 filters
cnn_model.add(Conv2D(64, (3, 3), activation='relu'))
cnn_model.add(MaxPooling2D((2, 2)))

# convolutional layer with 128 filters
cnn_model.add(Conv2D(128, (3, 3), activation='relu'))
cnn_model.add(MaxPooling2D((2, 2)))

# 2d to 1d layer transformation. flattening the image for dense layers.
cnn_model.add(Flatten())

# dropout
cnn_model.add(Dense(128, activation='relu'))
cnn_model.add(Dropout(0.5))

# 'sigmoid' activation, output layer, binary classification
cnn_model.add(Dense(1, activation='sigmoid')) #Value output of a value between 0 & 1: if near 1 it goes to the

# Compiling the model
cnn_model.compile(optimizer='adam',
                  loss='binary_crossentropy',
                  metrics=['accuracy'])

# Fit the model using the data generator
history = cnn_model.fit(
    datagen.flow(X_train, y_train_encoded, batch_size=32),
    steps_per_epoch=len(X_train) // 32,  # Using integer division to ensure the correct number of steps
    epochs=25,
    validation_data=(X_test, y_test_encoded)
)
```

Training/Validation Accuracy

Training Accuracy: 90.87%

Validation Accuracy: 92.33%

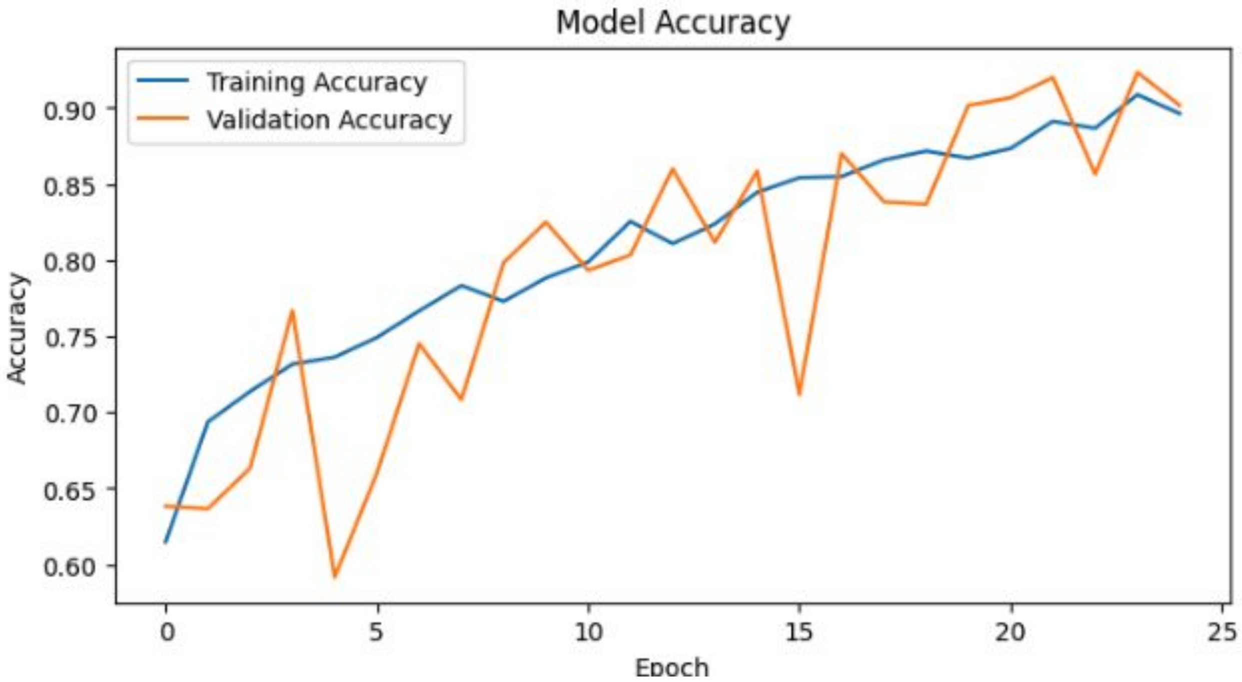
```
max_train_acc = max(history.history['accuracy']) * 100
max_val_acc = max(history.history['val_accuracy']) * 100
print(f'Maximum training accuracy: {max_train_acc:.2f}%')
print(f'Maximum validation accuracy: {max_val_acc:.2f}%')
```

```
Maximum training accuracy: 90.87%
Maximum validation accuracy: 92.33%
```

Maximum of history accuracy: at the end of each epochs

To show in percentage%

Model Accuracy Plot

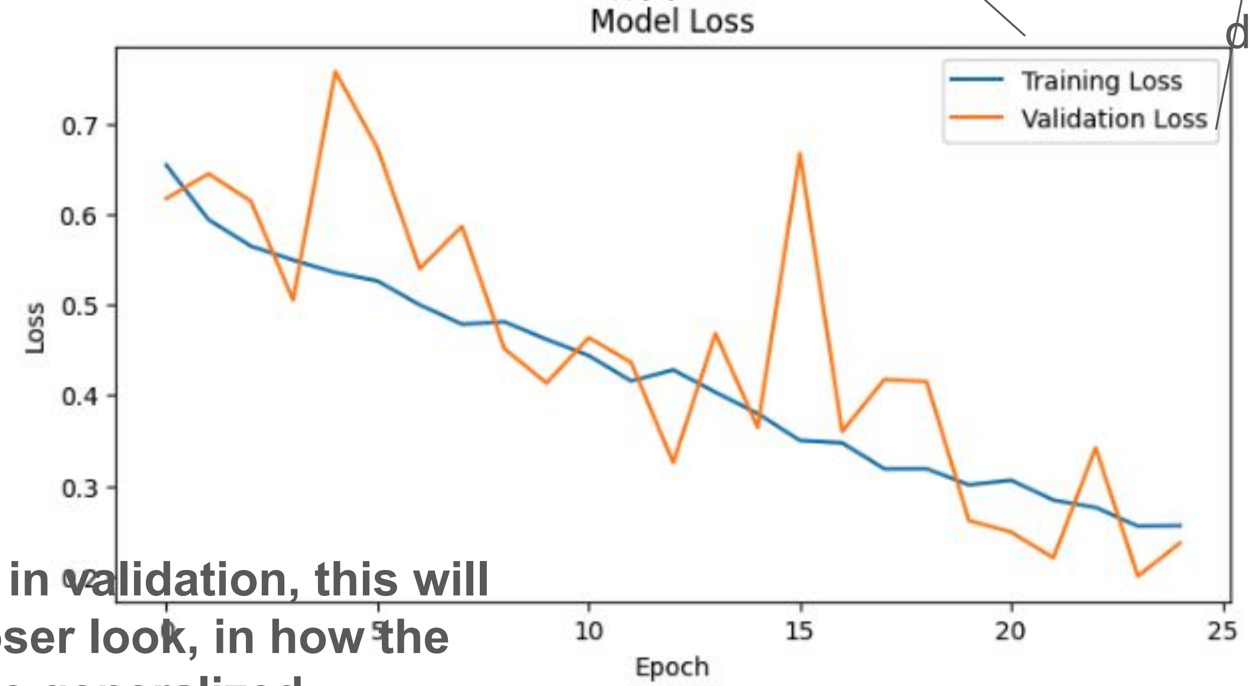


If training accuracy is higher than validation, then this is an indicator of overfitting in the data. **Training Accuracy and validating accuracy, are close to each other, this suggests that there are no overfitting in the data.**

Model Loss

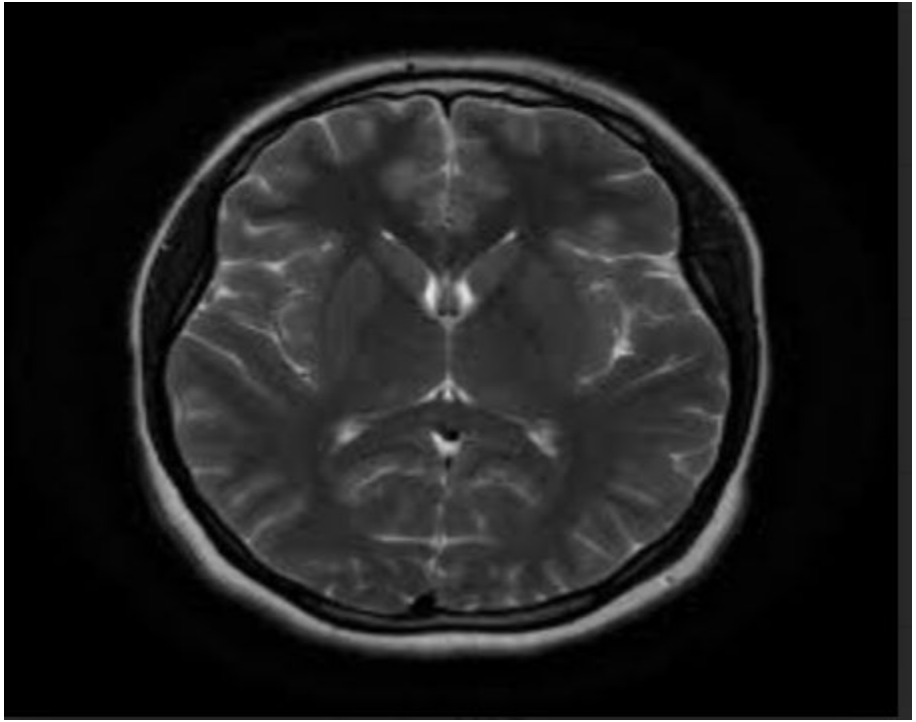
How well the training date is fitted

How well model is fitting the data



Variability in validation, this will need a closer look, in how the data will be generalized

Predicting an image for the first time



```python
# Loading Image
img_path = '/content/drive/MyDrive/BT-New dataset/pred/pred37.jpg'
image = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
image_resized = cv2.resize(image, (64, 64))
image_normalized = image_resized / 255.0
image_reshaped = np.reshape(image_normalized, (1, 64, 64, 1)) #batch dimensions

# prediction
prediction = cnn_model.predict(image_reshaped)
predicted_class = 'yes' if prediction[0][0] > 0.5 else 'no'

# Print the prediction
print(f"The model predicts this image is a '{predicted_class}' case.")
```

```
1/1 [==============================] - 0s 118ms/step
The model predicts this image is a 'no' case.
```

I am not satisfied with the accuracy I have achieved for, I am looking to improve accuracy through experimental trails/reading articles, look for methods, algs I can use to increase accuracy as much as possible.

Under the process to be completed by 4/3/2024

- Work in performing model accuracy. Select most accurate model

- Start the development of web interface

- Backend functionality of the web

Resources

1. https://keras.io/guides/sequential_model/