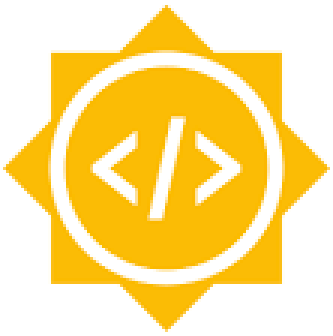

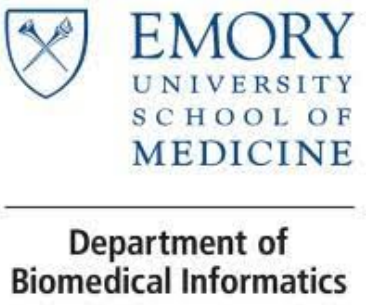

GSoC Proposal for Department of Biomedical Informatics, Emory University

A Methodology To Categorize Exam Series

HIMANSHU KUMAR

INDIAN INSTITUTE OF TECHNOLOGY (BHU), VARANASI

Github : https://github.com/Chevenger1

Email :himanshu.kumar.mec20@itbhu.ac.in

Phone No : 8287521526

Timezone : (GMT + 05:30) Asia / Kolkata - IST

Prospective Mentors :- Tony Pan , Puneet Sharma

Table of Contents :

1. Personal Introduction 4
   - 1.1 Introduction 4
   - 1.2 Previous projects 4
   - 1.3 Why do I wish to take part in GSoC 23 ? 5
   - 1.3 Why Emory BMI? 5
2. The Project 6
   - 2.1 Project summary 6
   - 2.2 Dicom : A nonstandard standard ? 6
   - 2.3 MRI series identification 6
3. Project Goals 7
   - 3.1 Series Identification Series 7
   - 3.2 Categorization 7
   - 3.3 Finding Anomaly In Study 7
   - 3.4 Confidence Score 7
4. The Plan 7
   - 4.1 Model_S 7
     - 4.1.1 Using DICOM meta data for training 8
     - 4.1.2 Feature selection 10
     - 4.1.3 Feature engineering and transformation 10
     - 4.1.4 Deep Neural Network architecture 11
   - 4.2 Anatomical Plane 13
   - 4.3 Model_C 14
   - 4.4 Input 14
   - 4.5 Sequence & anatomical plane 15
   - 4.6 Finding repeated & missing series 16
   - 4.7 Parameter compatibility 16
   - 4.8 Confidence score 18
5. The Timeline 19
   - 5.1 Community Bonding Period (May 4 - May 28) 19
   - 5.2 Development Phase 1 19
     - 5.2.1 Week 1 (May 29 - June 4) 19
     - 5.2.2 Week 2 (June 5 - June 11) 19
     - 5.2.3 Week 3 (June 12 - June 18) 19
     - 5.2.4 Week 4 (June 19 - June 25) 19
     - 5.2.5 Week 5 (June 26 - July 3) 19
     - 5.2.6 Week 6 (July 3 - July 9) 19
     - 5.2.7 Week 7 (July 10 - July 13) 19
   - 5.3 Development Phase 2 19
     - 5.3.1 Week 7 (July 14 - July 16) 19
     - 5.3.2 Week 8 (July 17 - July 23) 19

5.3.3 Week 9 (July 24 - July 30) 19
5.3.4 Week 10 (July 31 - August 6) 19
5.3.5 Week 11 (August 7 - August 13) 19
5.3.6 Week 12 (August 14 - August 20) 19
5.3.7 Week 13 (August 21 - August 28) 19
5.4 Post GSoC 20
6 Reference 20

1. Personal Introduction

1.1 Introduction

My name is Himanshu Kumar, and I am a junior year student pursuing my B.Tech degree at the **Indian Institute of Technology - BHU** in Varanasi, India. I am deeply interested in data science, as it allows me to conduct research, explore various fields, and engage in practical work. I have been pursuing my data science journey for the past year.I have background knowledge Linear Algebra and Abstract Algebra, Probability and Statistics, Python , C++ , Data Structure and Algorithms (ongoing).
I like exploring for new information , listening to different genres of music , philosophy and interesting historical events.

1.2 Previous Projects

These are some of my previous projects :

1. **Disaster Risk Monitoring Using Satellite Imagery (Github) :**The objective was to develop an efficient approach to detect and segment flooded areas in satellite images. achieve this, the team used a semi-supervised learning technique to analyze the Sentinel-1 SAR imagery. They incorporated data augmentation techniques to address the class imbalance problem present in the dataset. Additionally, the team used transfer learning to fine-tune a pre-trained model, which helped them achieve an impressive IOU (Intersection over Union) score of 76%.
2. **Discount Rate Prediction (Github) :** The project aimed to develop a predictive model that could accurately forecast discount rates for future transactions based historical data. The project involved data cleaning and feature extraction as the first step. The use of the Random Forest Regressor model ensured that the predictions were robust and reliable.The model was fine-tuned to improve its performance.

3. **Credit Card Fraud Detection Using Deep Learning(Github) :** This project developed a deep learning model to detect credit card fraud. Applied various techniques to balance the dataset, such as oversampling the minority class using SMOTE (Synthetic Minority Over-sampling Technique) and undersampling the majority class. A deep neural network was trained on a dataset of both fraudulent and non-fraudulent transactions. We used binary cross-entropy as the loss function and Adam as the optimizer. After training and testing the model, we achieved a high accuracy and F1-score, with minimal false negatives. The results demonstrate the effectiveness of deep learning for credit card fraud detection.

4. Fast AI Project (Github) : This project involves the implementation of a text classification model using PyTorch. The model is trained on the AG News dataset using a neural network architecture consisting of an embedding layer, a linear layer, and a cross-entropy loss function. The training and validation data are loaded using PyTorch's DataLoader and are preprocessed using a tokenizer and a vocabulary. The project also uses TensorBoard to visualize the training and validation loss and accuracy metrics.

Github discussion : A method to categorize exam series

1.3 Why do I wish to take part in GSoC 23 ?

I have been pursuing my deep learning journey with great enthusiasm and curiosity. I am fascinated by the way machines can learn and make decisions on their own by analyzing and processing vast amounts of data. In my quest to learn more, I have taken various online courses and attended workshops to gain a strong foundation in the field. To further deepen my understanding, I have also worked on several personal projects. Through these projects, I have honed my skills in Python, Pytorch, Keras, and other relevant technologies. I believe that open-source software is critical to the success of deep learning and machine learning as a whole. That is why I am excited to participate in open-source initiatives such as Hacktoberfest and GSoC. I am eager to contribute to the community by sharing my knowledge and expertise and learning from other experienced developers. By participating in GSoC 23, I hope to gain more exposure to the open-source community and improve my workflow strategies. Ultimately, my goal is to use my skills and knowledge to create innovative solutions that have a positive impact on society.

1.4 Why Emory BMI ?

Emory BMI is at the forefront of cutting-edge medical technology and research. With their commitment to advancing the field of medical imaging, they are paving the way for a better future in healthcare. As an engineering student with a keen interest in biology, I am impressed by their work and the impact it has on society. I am particularly drawn to the A methodology to categorize exam series, which aligns with my expertise in tech stacks and my desire to make a positive contribution to biomedical. Through my participation in the GSoC program, I look forward to expanding my knowledge of new technologies and medical concepts. I am excited about the opportunity to work with Emory BMI and to contribute to the advancement of medical research.

2. The Project

Niffler is a DICOM networking research project that is designed to help researchers efficiently retrieve and process DICOM images and metadata for machine learning workflows. With Niffler, researchers can quickly receive DICOM images from multiple Picture Archiving and Communication Systems (PACS) in real-time and on-demand. This approach saves time and resources for researchers, allowing them to focus on analyzing the data and producing meaningful results.

2.1 Project Summary

I am going to add a new module to run on DICOM images of MRI modality. This module will execute on DICOM images acquired in real-time or on-demand by Niffler.
This module will identify the MRI series (orientation, weighting) and will categorize to different brain sequences. It will help in identification of repeated or missing series in a MRI study. It will also check whether sequence parameters are meeting standard thresholds or not. At last confidence score will be given.

2.2 DICOM : A Nonstandard Standard?

DICOM is the global standard for managing and exchanging medical images and data, but it lacks a consistent series-naming scheme. This makes it difficult for PACS to automatically organize imaging examinations, so radiologists often have to manually arrange sequences for interpretation. The "Series Description" attribute in DICOM is not standardized and may not provide enough information to determine the sequence, requiring radiologists to check visually. DICOM data objects have numerous attributes, including patient demographics and technical parameters, and manufacturers have significant flexibility in tagging them as required, potentially null, or optional, and may also include their own private elements.

2.3MRI Series Identification

Automating the identification and selection of image series poses a significant challenge due to various sources of variability, which can cause acquisition parameters and meta data to differ from patient to patient. While relying on a uniform naming scheme for series may seem like a simple solution, it is not always reliable. Instead, leveraging the characteristics of the image pixels appears to be the most natural way to perform this task. This approach is similar to the one used by radiology professionals and has been explored in previous research. Another approach involves utilizing DICOM meta data, rather than pixel data, for the series categorization of brain MRI studies. Both of these methods can help improve the automation of series identification and selection, despite the challenges posed by variability. I will explore first DICOM meta data approaches for series identification which depends upon how informative meta data is available to us.

3. Project Goals

- ## 3.1 Series Identifcation

  The system will identify the series in a brain sequence and determine the orientation and weighting of each series

- ## 3.2 Study Categorization

  After identifying the series, the system will categorize the study into different brain protocols based on the predefined standard protocols.

- ## 3.3 Parameters Threshold

  The system will check whether the DICOM parameters of each series meet certain threshold values of acceptance.

- ## 3.4 Finding Anomaly In Study

  The system will identify whether any series in a MRI Study is repeated or missing altogether.

- ## 3.5 Confidence Score

  Finally, the system will give a confidence score to each study.

4. The Plan

In this section, I will describe the strategy and process that I will use for my project. The specific steps for carrying out this plan will be outlined in the following subsection. Additionally, I will work closely with my mentor to review and improve the plan's effectiveness throughout the project's duration.

4.1 The Model_S

I am going to create a deep neural network, let's call it "Model_S", which will identify the sequence and its weighting. It will accept a dataframe, let's call it "study_dfs", which contains the selected metadata for this task. I have described the whole process in consecutive topics, from data extraction to training. Some implementation details may be changed during the GSoC period, as suggested by mentors.

Note : The code snippets used in the proposal are for skill demonstration purposes and serve as proof of concept. The final implementation will be carried out after discussions with mentors and considering their suggestions. Therefore, it may vary from the code snippets presented.

4.1.1 Data Extraction For Training

The first step of our project is to extract and transform the data into a usable form that is convenient to process and understand.

For data extraction, I plan to use Pydicom as we will be using TCIA images during the starting period and training duration. If data is provided from MongoDB during the contribution period, I am happy to perform ETL on it as well. For the final stage, I will switch to extracting data from MongoDB.

```
import pydicom as dicom
import pandas as pd
import glob
import os
from tqdm import tqdm
folder_path = r"C:\Users\HIMANSHU GUPTA\Desktop\trial dataset"
images_path = os.listdir(folder_path)
```

This code defines a function called "dictify", which takes a DICOM (Digital Imaging and Communications in Medicine) dataset object as input and recursively converts it into a nested dictionary. The output dictionary corresponds to each DICOM element, where the tag of each element is used as a key and the value is the value of that element. If the element is a sequence, its value is a list of dictionaries representing the items in the sequence.

The function first creates an empty dictionary called "output". It then iterates through each element in the DICOM dataset using a for loop. If the VR (Value Representation) of the element is not "SQ" (Sequence), it adds a new key-value pair to the output dictionary where the key is the element's tag and the value is the element's value. If the VR is "SQ", it creates a list of dictionaries using the recursive "dictify" function and adds this list as the value for the current element's tag in the output dictionary.

Finally, the function returns the output dictionary.

```
def dictify(ds: dicom.dataset.FileDataset) -> Dict:
    """Recursively converts a DICOM dataset object to a nested dictionary.

    Args:
        ds (dicom.dataset.FileDataset): The DICOM dataset object to be converted.

    Returns:
        dict: A dictionary representation of the DICOM dataset object, where each item in the dictionary
        corresponds to a DICOM element. If the element is a sequence, its value is a list of dictionaries
        representing the items in the sequence."""

    output = dict()

    for elem in ds:
        if elem.VR != 'SQ':
            output[elem.tag] = elem.value
        else:
            output[elem.tag] = [dictify(item) for item in elem]
    return output
```

The code defines a function called read_dicom_metadata that takes in two arguments: folder_path which is a string representing the path to the root directory containing the DICOM files, and images_path which is a list of strings representing the relative paths to subdirectories containing the DICOM files.

The function uses a loop to iterate through the images_path list and for each subdirectory, it uses the os.walk function to traverse the directory and its subdirectories to find all DICOM files with a '.dcm' extension. For each DICOM file found, it reads the metadata using the dicom.dcmread function and converts the metadata into a Python dictionary using the dictify function. The resulting dictionaries are appended to a list called metadata_train.

```
def read_dicom_metadata(folder_path: str, images_path: List[str]) -> List[Dict[str, Any]]:
    """Reads DICOM files from a specified directory and converts their metadata to a list of dictionaries.

    Args:
        folder_path (str): The path to the root directory containing the DICOM files.
        images_path (List[str]): A list of relative paths to subdirectories containing the DICOM files.

    Returns:
        List[Dict[str, Any]]: A list of dictionaries, where each dictionary represents the metadata of a DICOM file.
        The keys of each dictionary are the DICOM tags (in string format), and the values are the corresponding values
        from the DICOM file (in the appropriate Python data type).
    """
    metadata_train = []
    for img_path in tqdm(images_path):
        for dirpath, dirnames, filenames in os.walk(folder_path + img_path):
            for filename in filenames:
                if filename.endswith('.dcm'):
                    filepath = os.path.join(dirpath, filename)
                    ds = dicom.dcmread(filepath, stop_before_pixels=True)
                    metadata_train.append(dictify(ds))
    return metadata_train
```

Then converting above metadata_train into dataframe using pd.dataframe().This will be used for train. Similarly we can do with test data and validation data.

4.1.2 Feature Selection

Now, here I come to one of the most important part for model training.A list of sequence metadata, including sequence specific headers,timing parameters, spatial and contrast properties, and hardware manufacturers, of DICOM images are selected as the features for the present machine learning study. In comparison with data-driven feature selection, domain knowledge guided feature selection can help prevent the problem of overfitting where the constructed models may not reflect the true relationships in the data set. Later I can confirm our preassumption using 'feature_importances_' attribute of the model.

	DICOM Attribute	DICOM TAG
1	Scanning sequence	(0018,0020)
2	Sequence variant	(0018,0021)
3	Echo time	(0018,0081)
4	Inversion time	(0018,0082)
5	Pixel bandwidth	(0018,0095)
6	Repetition time	(0018,0080)
7	MR acquisition type	(0018,0023)
8	Image type	(0008,0008)
9	Pixel spacing	(0028,0030)
10	Slice thickness	(0018,0050)
11	Photometric interpretation	(0028,0100)
12	Contrast bolus agent	(0018,0010)
13	Scan options	(0018,0022)

The list of sequence metadata mentioned above is provided for reference purposes only. It may vary when I receive the data from Emory for training. Afterward, I will select only those columns containing the aforementioned metadata for model training.

Note : I will also explore on using study description(0008, 1030) and series description(0008, 103e)attributes which is not discussed here. But I will try to explore it in pre GSoC period before 4th May.

4.1.3 Feature Engineering

To transform attribute values, I use different techniques depending on the type of value. Numeric values are converted to floats by default. For string values, I convert them into binary features using a specific method. If there's a finite set of predefined values, each value is turned into a binary feature. If there are no predefined values, I use the training dataset to identify possible values, and convert them into binary features. Lists of strings are also converted into binary features using the same procedure, where each value in the list is transformed into a binary feature. This approach ensures that all attribute values are properly transformed into a format that can be effectively used for further analysis.

Note : I am assuming that the list of final features you will get are important and will not be empty or absent during use. This is a reasonable assumption, but it is important to be aware of the possibility that the list of features could change in the future. If this happens, I may need to update our data transformation process accordingly.

I will discuss this problem with project mentors and try to reach the optimal solution. They may have additional insights or suggestions that can help me to improve data transformation process.

There are a number of different ways to transform data, and the best approach will vary depending on the specific data set . However, some common data transformation techniques include:

Scaling: This involves converting the data into a common scale, such as a range from 0 to 1. This can help to improve the accuracy of the machine learning algorithm.

Normalization: This involves adjusting the data so that it has a normal distribution. This can help to improve the performance of the machine learning algorithm.

Cleaning: This involves removing any errors or inconsistencies from the data. This can help to improve the accuracy of the machine learning algorithm.

Data transformation is a critical step in the machine learning process. It can help to improve the performance, accuracy, and cost-effectiveness of machine learning algorithms.

4.1.4 Deep Neural Network Architecture

As there in no research done with sequence classification using deep neural network . I will follow the path of experimentation as the model architecture and the model parametres will vary depending upon training data, model performance

Input Layer : Number of neural networks in this layer will depend upon the final number of input features(columns) I will get after feature engineering and transformation.

Hidden Layer :I will start from 10 hidden layers having number of neurons equal to the number of input. I will monitor the performance of model and computational time . I wil finally decide after hyper parameter tuning of model.

Optimizer: The optimizer should be Adam.

Loss function: The loss function should be cross-entropy.

Output layer : Number of neuronswill depend upon the number of sequences I have to classify which I will get from protocol(like T2, T1, FLAIR,Diffusion, fmri, mra etc.). I will use softmax layer for final confidence score.

Regularization :I will use drop out as the regularization method to stop overfitting.

Example code :

This code defines a DeepNet class that inherits from nn.Module. The constructor of the DeepNet class creates 40 linear layers with 20 neurons each, and an output layer with 8 neurons. The forward method defines the forward pass through the network.

The code then defines the number of folds for cross-validation and initializes the model, loss function

```
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.model_selection import KFold
```

```python
class DeepNet(nn.Module):
    def __init__(self):
        super(DeepNet, self).__init__()
        self.layers = nn.ModuleList()
        for i in range(40):
            self.layers.append(nn.Linear(20, 20))
        self.out = nn.Linear(20, 8)

    def forward(self, x):
        for layer in self.layers:
            x = layer(x)
        x = self.out(x)
        return x

# create dataset with 40 features and 8 categories
# X_train, y_train, X_test, y_test = ...

# define the number of folds for cross-validation
n_splits = 5

# initialize the model
model = DeepNet()

# define the loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters())

# create cross-validation folds
kf = KFold(n_splits=n_splits, shuffle=True)

# train the model using cross-validation
for train_index, val_index in kf.split(X_train):
    # split the data into training and validation sets
    X_train_fold, X_val_fold = X_train[train_index], X_train[val_index]
    y_train_fold, y_val_fold = y_train[train_index], y_train[val_index]

    # convert data to PyTorch tensors
    X_train_fold, y_train_fold = torch.Tensor(X_train_fold), torch.Tensor(y_train_fold).long()
    X_val_fold, y_val_fold = torch.Tensor(X_val_fold), torch.Tensor(y_val_fold).long()

    # train the model for one epoch
    for epoch in range(100):
        # forward pass
        outputs = model(X_train_fold)
        loss = criterion(outputs, y_train_fold)

        # backward pass and optimization
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        # print the training loss every 10 epochs
        if epoch % 10 == 0:
            print(f"Fold: {fold}, Epoch: {epoch}, Loss: {loss.item()}")

    # evaluate the model on the validation set
    with torch.no_grad():
        outputs = model(X_val_fold)
        val_loss = criterion(outputs, y_val_fold)
        _, predicted = torch.max(outputs.data, 1)
        accuracy = (predicted == y_val_fold).sum().item() / y_val_fold.size(0)
```

We will use the Model_S mentioned above to predict the weighting of the series instance. Afterwards, we will add this weighting as a column to the parsed study_dfs. Subsequently, we will perform one-hot encoding on this column. Now, we need to determine the anatomical plane of the image.

4.2 Anatomical Plane

This information can also be obtained from the Image Orientation Patient DICOM (Digital Imaging and Communications in Medicine) attribute. This attribute provides the direction cosines of the first row and first column of an image relative to the patient. By calculating the main direction of the normal to the slices, the anatomical plane information can be determined. Therefore, relying on the Image Orientation Patient attribute is a more reliable method than depending on manually entered information in the series description.[7]

Code concept example:

```
image_orientation = dicom_file[0x0020, 0x0037].value

# Step 1: Calculate the direction cosines of the first row and first column of the image
row_cosines = image_orientation[:3]
column_cosines = image_orientation[3:]

# Step 2: Calculate the normal vector to the slice using the cross product of the direction cosines
normal_vector = np.cross(row_cosines, column_cosines)

# Step 3: Determine the anatomical plane of the image
abs_normal_vector = np.abs(normal_vector)
largest_component = np.argmax(abs_normal_vector)

if largest_component == 0:
    if normal_vector[0] > 0:
        anatomical_plane = "Sagittal"
    else:
        anatomical_plane = "Sagittal (flipped)"
elif largest_component == 1:
    if normal_vector[1] > 0:
        anatomical_plane = "Coronal"
    else:
        anatomical_plane = "Coronal (flipped)"
else:
    if normal_vector[2] > 0:
        anatomical_plane = "Axial"
    else:
        anatomical_plane = "Axial (flipped)"

# Step 4: Use the determined anatomical plane information in subsequent analysis or processing
print("Anatomical plane: ", anatomical_plane)
```

First, I will add sequence type and anatomical information to the dataframe. Then, we will perform one-hot encoding on these two newly added columns in study_dfs. This encoding will be used for the final categorization by Model_C into different types of brain protocols.

4.3 Model_C

Now, I will use this model for series categorization. Instead of using a single model, I utilized two models to achieve classification and obtain a sequence type that represents the weighting of the image. This information is crucial for precisely identifying the repeated or missing series type; without it, I did not believe it was possible. However, I will be discussing this approach with my mentors to optimize it or improve my plans. The training data will be in the form of a dataframe called **"proto_dfs".**

t1	t2	DWI	exam type
1	1	0	brain trauma

The columns will include all the sequences and their parameters, with the last column dedicated to the exam type. I will create data in this format by utilizing the provided pre-defined protocol template for each type of exam. These dataframes will be appended to form the final training data. Then, I will carry out data transformation to train Model_C. As most of the processes are similar to Model_S, I will not elaborate on them here. Once the training is complete, I will utilize the model for identifying the exam type of the study and series.

4.4 Input

Here I am going to perform extraction of data from MongoDB . I am considering that all the sequences images belonging to particular exam are present in the folder and we have the MongoDB URI for this. Later after discussion with mentor if there is need arises of another approach I am willing to that.
This code will iterate through each study directory, then each series directory within that study, and finally through each DICOM file in the series. It will extract the metadata fields of interest from each DICOM file and add them to a list of dictionaries representing the data for each image in the series. This list of dictionaries is then converted to a dataframe for the series.. Here is an example:

```
def parse_dicom_dir(dir_path: str, mongo_uri: str, db_name: str, collection_name: str) -> list:
    """
    Parses a directory containing DICOM files and returns a list of dataframes for each study.

    Parameters:
    dir_path (str): Path to the directory containing DICOM files
    mongo_uri (str): URI of the MongoDB instance
    db_name (str): Name of the MongoDB database
    collection_name (str): Name of the MongoDB collection to store parsed data

    Returns:
    list: List of dataframes for each study in the directory
    """
    # Connect to MongoDB
    client = MongoClient(mongo_uri)
    db = client[db_name]
    collection = db[collection_name]

    # Define a list to store the dataframes for each study
    study_dfs = []

    # Loop through each study directory
    for study_dir in os.listdir(dir_path):
        study_path = os.path.join(dir_path, study_dir)
        if os.path.isdir(study_path):
            # Define a list to store the dataframes for each series in the study
            series_dfs = []
            # Loop through each series directory
            for series_dir in os.listdir(study_path):
                series_path = os.path.join(study_path, series_dir)
                if os.path.isdir(series_path):
```

```
                # Define a list to store the data for each image in the series
                series_data = []
                # Loop through each DICOM file in the series directory
                for file_name in os.listdir(series_path):
                    if file_name.endswith('.dcm'):
                        file_path = os.path.join(series_path, file_name)
                        # Read the DICOM file
                        ds = pydicom.dcmread(file_path)
                        # Extract the metadata fields of interest and add to the series data list
                        series_data.append({
                            'StudyID': ds.StudyID,
                            'SeriesInstanceUID': ds.SeriesInstanceUID,
                            'PatientID': ds.PatientID,
                            'Modality': ds.Modality,
                            'BodyPartExamined': ds.BodyPartExamined,
                            'PixelSpacing': ds.PixelSpacing,
                            'ImagePositionPatient': ds.ImagePositionPatient
                        })
                # Convert the series data list to a dataframe and add to the series dataframes list
                series_df = pd.DataFrame(series_data)
                series_dfs.append(series_df)
        # Concatenate the series dataframes into a single dataframe for the study
        study_df = pd.concat(series_dfs, ignore_index=True)
        study_df.drop_duplicates(subset='SeriesInstanceUID', inplace=True)
        study_dfs.append(study_df)
        # Insert the study dataframe into the MongoDB collection
        study_dict = study_df.to_dict('records')
        collection.insert_many(study_dict)

    return study_dfs
```

Then, this study_dfs will go through our data transformation pipeline. We will pass this data to our Model_S, which will add the column of sequence weighting to it. We will also add the anatomical plane information as described earlier. After that, we will perform one-hot encoding for both of these columns and add them to the study_dfs example.

```
# perform one-hot encoding on the "weighting" and "orientation" columns
one_hot = pd.get_dummies(study_dfs, columns=["weighting", "orientation"])


# concatenate the one-hot encoded DataFrame with the original DataFrame
study_dfs = pd.concat([study_dfs, one_hot], axis=1)
```

Series UID	Echo time	Inversion time	Pixel bandwidth	Repetition time	Pixel spacing	Slice thickness	Scanning sequence_GRE	Scanning sequence_SE	Sequence variant_T1	Sequence variant_T2	MR Weighting_T1	MR Weighting_T2
1234567890	-0.371647	-1.341641	-1.362770	-1.341641	-1.0	-1.0	0	1	1	0		
9876543210	1.709577	0.447214	0.733799	0.447214	-1.0	-1.0	1	0	0	1	1	0
63543135	-0.520306	-0.447214	-0.524142	-0.447214	1.0	1.0	0	1	1	0	0	1
32543145	0.817624	1.341641	1.153113	1.341641	1.0	1.0	1	0	0	1	1	0
											0	1

Above image is for visualization of logic , it is not the output of above code.

We will now transform study_dfs to create a new dataframe that can be parsed by Model_C. The new dataframe, named exam_dfs, will contain additional columns that are not used in Model_S but will be useful in Model_C and for later use. These columns will include sequence weightings, anatomical planes, slice thickness, DWI, and more.

	Study UID	T1 Weighted	T1 Repetition Time	T1 Echo Time	T2 Weighted	T2 Repetition Time	T2 Echo Time	FLAIR	FLAIR Repetition Time	FLAIR Echo Time
0	P001	SE	500	10	FSE	2000	80	IR	8000	120
1	P002	SE	500	10	FSE	2000	80	IR	8000	120
2	P003	GRE	600	15	SE	2500	100	IR	10000	150

Above dataframe is for visualization of exam_dfs to help in understanding of below topic..

The term "SID" refers to "Study UID". We will pass this data to "Model_C", which will predict the exam. Afterward, we will add the exam column to the "exam_dfs" dataframe. Additionally, we will include this exam column in the "study_dfs".

4.6 Finding Repeated & Missing Series

In the function study_dfs, we will add up the weightings for each exam's rows. If any sequence or series is repeated, we will obtain a value greater than 1. Therefore, any weighting that has a sum greater than 1 will be identified as repeated and printed as such.

To find any missing weightings, we will loop through all weightings in exam_dfs and compare them to the corresponding weightings in proto_dfs for the same exam, in order to match their binary values. If any mismatches occur, it indicates that the weighting was missing from exam_dfs, and that particular series will be identified as the missing one.

4.7 Parameter Compatibility

Each MRI sequence has its own specific set of parameters that are optimized for the particular type of contrast and tissue characteristics being imaged. However, the values of these parameters can vary widely between different sequences, making it essential to establish predetermined threshold values for each sequence to ensure compliance with the imaging protocol. In this part, do checking the threshold values of MRI sequence parameters for compliance with the predetermined values.

Till here we know the all the sequence in our study, exam type of study and all the series, which series are repeated or missing.

I have a database called study_dfs which contains information about different MRI sequences of brain scans. The schema of study_dfs is as follows:

Index: Series UID
Columns:
sequence_name: Name of the MRI sequence (e.g. T1, T2, FLAIR, etc.)
exam_type: Type of exam protocol used for the scan (e.g. standard brain protocol, research protocol, etc.)

The sequence parameter columns in the study_data will be like as T1_echo, T1_TR, T1_TE, T1_flip_angle, T2_echo, T2_TR, T2_TE, T2_flip_angle, FLAIR_inversion_time, FLAIR_echo, FLAIR_TR, FLAIR_TE, FLAIR_flip_angle etc

I want to check whether the sequence parameters for each scan comply with the predefined threshold values for the respective exam protocol. For example the threshold values for each exam protocol are predefined as follows:
Standard brain protocol:

T1: echo: 20
TR: 2000
TE: 10
flip angle: 90
sequence type: MAGNETIZATION_PREPARED_GRADIENT_ECHO

T2: echo: 80
TR: 5000
TE: 80
flip angle: 180
sequence type: T2_TSE

FLAIR: inversion time: 2000
echo: 100
TR: 9000
TE: 120
flip angle: 90
sequence type: FLAIR

The code should first check the exam_type of each scan from the database and match it with the corresponding predefined exam protocol. Then, it should check each sequence parameter of the scan against its respective threshold value for the predefined exam protocol. If a parameter value is found to be non-compliant, the code should output a message indicating which parameter value failed to comply with its threshold value.
We will check if the parameters are not within threshold, it will count it for cutting the confidence score.

4.7 Confidence Score

At last we will give confidence score.It will be calculated by checking for each successful sequence as 1 and if there is any non compliance of threshold value for any sequence parameter then we will deduct 1 point.
It is calculated like this as suggested by mentors earlier in discussion.

5. The Timeline

With the summer vacation just before the official coding period begins, I am eager to devote myself to this project with utmost dedication and commitment. Without any prior commitments or distractions, I can easily commit 35-40 hours per week towards achieving our goals within the given timeframe. Even if it means going the extra mile, I am more than willing to put in the extra effort required to ensure that the project is completed to the best of my abilities.
Moving forward, my next semester commences from July 23rd. However, I have taken measures to ensure that my workload is minimized during the initial phase, allowing me to devote 30 hours per week towards this project during the final phase. I understand the importance of staying ahead of deadlines and plan to prioritize my work accordingly to avoid any unnecessary stress.
Il'll also responsibly keep my mentor updated in case of any emergency that occurs with suitable details.

5.1 Community Bonding Period

Having dedicated over a year to studying deep learning, I feel confident in my ability to begin work on my project and engage in more detailed discussions with my mentor regarding its implementation. As I delve deeper into this phase, I also plan to explore related modules and gain familiarity with the niffler code base. Additionally, I intend to expand my knowledge on converting models from research to production and familiarize myself with the proper etiquette and guidelines for using Github.

May 4 - May 28

Learning more about niffler modules and code base.

I will start a blog documenting my journey with Emory BMI.

Discussing my project with my mentors and the community to see if more improvements can be made in the plan of action.

Interact with maintainers to get my doubts/technical design problems clear if there are any.

I will learn all the etiquettes for software development and testing. As testing standards are very high for software used in medical purposes.

Note : Documentation and testing of code will run in parallel in during coding phase.

5.2 Development Phase 1

5.2.1 Week 1 (May 29 - June 4)

- Developing the ETL pipeline for training data .

5.2.2 Week 2 (June 5 - June 11)

- Doing the exploratory data analysis

5.2.3 Week 3 (June 12 - June 18)

- Starting the iterative process for development of Model_S

5.2.4 Week 4 (June 19 - June 25)

- Working on model improvement and hyper parameter tuning

5.2.5 Week 5 (June 26 - July 2)

- Working on function for finding the Anatomical plane

5.2.6 Week 6 (July 3 - July 9)

- Converting the predefined protocol to useful form

5.2.7 Week 7 (July 10 - July 13)

- Data transformation for training of Model_C

5.3 Development Phase 2

5.3.1 Week 7 (July 14 - July 16)

- Starting the iterative process for Model_C

5.3.2 Week 8 (July 17 - July 23)

- working on model improvement and hyperparameter tuning

5.3.3 Week 9 (July 24 - July 30)

- Working on ETL pipeline for data loading to module

5.3.4 Week 10 (July 31 - August 6)

- Working on function for finding repeated and missing series

5.3.5 Week 11 (August 7 - August 13)

- Working on function for checking parameter compatibilty

5.3.5 Week 12 (August 14 - August 20)

- Working on confidence score giving function and score system

5.4.2 Week 13 (August 21 - August 28)

- completion of project and submission of mentor evaluation

5.4 Post GSoC

As someone deeply interested in Deep Learning, I will be excited to continue contributing to Emory BMI in the future. I believe that working on the Niffler Project will align perfectly with my interests and skills. I will be committed to assisting new contributors to the project and continuing to contribute myself. Being a part of such a supportive and innovative community will make me proud. Additionally, I plan to conduct research on new and more robust techniques that can be added to Niffler to enhance its power and capabilities.

6 References

1. Niffler
2. Emory HITI
3. DICOM
4. Pydicom
5. Deep Learning
6. Deep Learning for NLP and Speech Recognition 2019 ISBN : 978-3-030-14595-8
7. Liang, Shuai et al. “Magnetic Resonance Imaging Sequence Identification Using a Metadata Learning Approach.” Frontiers in neuroinformatics vol. 15 622951. 17 Nov. 2021, doi:10.3389/fninf.2021.622951
8. Pytorch
9. Pizarro, Ricardo et al. “Using Deep Learning Algorithms to Automatically Identify the Brain MRI Contrast: Implications for Managing Large Databases.” Neuroinformatics vol. 17,1 (2019): 115-130. doi:10.1007/s12021-018-9387-8