MACHINE INTELLIGENCE FOR BRAIN SEGMENTATION: A TOOL TO IDENTIFY BRAIN ILLNESSES THROUGH SEGMENTATION

Vikranth Nara, Harshit Pottipati and Rishi Athavale

Rock Ridge High School, 43460 Loudoun Reserve Dr, Ashburn, VA, United States of America

ABSTRACT

Machine Learning is becoming a prominent force in the medical field. We created Machine Intelligence for Brain Segmentation (MIBS), a tool that segments brain MRIs into different colors that signify enhancing tumors, non-enhancing tumors, and edema. The dataset used was of 624 MRIs from the Medical Segmentation Decathlon. The model was trained with the U-Net algorithm, a Convolutional Neural Network made for Biomedical Image Segmentation, and resulted in an average accuracy of ~99% across the different classes and ~0.75 an average F1-score across the different classes.

1. Introduction

For a number of brain illnesses including brain cancer, Magnetic Resonance Imaging (MRI) scans are used as a method for doctors to visually identify anomalies. MRI's are useful because they are painless for patients and allow doctors to effectively plan necessary treatments (such as surgeries) [1]. However, this method as of now is flawed. This is because different doctors can arrive at different conclusions from the same MRI scan, resulting in diagnostic variance. Additionally, the accuracy of a doctor's diagnosis can be affected by their current mental state (such as them being tired). Research has shown that doctors make significantly more mistakes when tired [2]. Furthermore, it is time consuming for doctors to analyze MRI scans due to a large number of MRI scans taken per year (with 30 million being taken every year in the US alone [3]) and a limited supply of doctors.

2. Machine Intelligence for Brain Segmentation

Even though a conclusion may review the main results or contributions of the paper, do not duplicate the abstract or the introduction. For a conclusion, you might elaborate on the importance of the work or suggest the potential applications and extensions.

2.1 Backend(segmentation):

2.1.1 Dataset and Model Architecture

The segmentation of the MRI scan is done using machine learning, which is a subfield of AI (Artificial Intelligence). Artificial Intelligence revolves around creating machines capable of acting intelligently, and machine learning accomplishes this by having the machine learn on its own. In our project, the machine learning model learned to segment MRI scans by "practicing" segmenting many MRI scans from a dataset of MRI scans taken from the Medical Segmentation Decathlon [4]. The dataset includes images of MRI scans and the correct segmentation for that MRI scan. In the dataset, the MRI scans and segmentations are in the form of 3D images and are stored as .nii.gz files. They are extracted as numpy arrays. From those numpy arrays, every 2D "slice" from the 3D images that have a background to brain ratio (amount of the image that is just background) of less than or equal to 0.925 are selected and added to the dataset. Our dataset consists of 624 2D slices, of which 60% are used for the training set (which the models train on), 20% are used for the cross validation set (which is used to compare different models), and 20% are used for the test set (which is used to evaluate the model that performs the best on the cross validation set). Once various different models were trained and a model was selected, it is then saved and used later on to segment an MRI scan chosen by the user.

2.1.2 Data Processing

The dataset used for this is from the Decathlon 10 Challenge. Each of the MRI scans from the dataset is of dimensions 244x244x155x4, and the segmentations are of dimensions 244x244x155x1. 2D slices that don't have more than 92.5% of the image as just background are then taken from the MRI scans and the segmentations, of which there are 624. These 2D slices are then used for the dataset. The values in the segmentations are either 0, 1, 2, or 3, with 0 meaning background, 1 meaning edema, 2 meaning non-enhancing tumor, and 3 meaning enhancing tumor. During the extraction of the segmentations, they are one-hot-encoded into numpy arrays, which means that each individual value is converted to a vector that represents the value (for example, when 0 is one hot encoded it produces the vector [1, 0, 0, 0] and when 2 is one hot encoded it produces the vector [0, 0, 1, 0]). The reason that one-hot-encoding is used is that, since this is a classification problem, if the values remained as 0, 1, 2, 3, and 4, the machine would assume that there is a linear relationship between the values when in reality there isn't one. The "background" class is then removed (it is redundant, as if every other class is 0 then we

know that it is background), which changes the number of classes from 4 to 3. The data is then split into a training set, a cross-validation set, and a test set. The training set will be used to train the models, the cross-validation set will be used to compare the models, and the test set will be used to evaluate the model that is determined to perform the best on the cross-validation set. 60% of the data goes into the training set, 20% into the cross-validation set, and 20% into the test set. The reason that separate datasets are needed to train and evaluate the model (rather than just one dataset that is both used to train and evaluate the model) is that doing so would make it hard for us to see whether or not the model is overfitting to the data. Overfitting means that the model, rather than understanding the trend in the data and making predictions based on that, the model fails to understand the general trend in the data. This results in the model doing well on the training set, but when encountered with new data from outside of the training set, it performs far worse.

As an analogy, evaluating a model on its performance on the training set would be like evaluating a student's understanding of a subject based on a test that consists of the same questions as are in the homework. The student could get a high score on the test without understanding the material by simply memorizing the answers from the homework. After the data is split, it is then standardized, which means that the data is subtracted by the mean and then divided by the standard deviation. This is done to make the data easier for the machine to evaluate while also not losing any relevant information. This is done for every channel in the 2D MRI scans (the 2D MRI scans have 4 channels), and the mean and standard deviation value used is from the training set rather than from the whole dataset. This is done to prevent data leakage, which is when some information from the validation and test sets "leaks" into the training set, which allows the machine to have some information about the validation and test sets beforehand. This is a problem because it can result in the machine learning model performing better on the validation and test sets than it would on outside pieces of data.

3.1 Backend(segmentation):

3.2 Model creation and training

3.2.1 Convolutional Neural Networks

Much of modern machine learning and deep learning uses neural networks, which is a computer system modeled on the nervous system of the human brain. It includes a series of connected neurons, which pass values along the network. Neural networks have an input layer, which takes in inputs, and an output layer, which outputs values. Some neural networks also have a series of hidden layers between the input and output layers, where each one takes in information from the previous layer and sends information to the next layer. The process by which neural networks

generate outputs from inputs is known as forward propagation. In a neural network, each layer does some mathematical operation on the values it receives. When training a neural network, the network performs forward propagation on some input and then compares its output to the correct output. Based on how incorrect the network was, it will then change some of the mathematical operations in the network in a process known as back propagation in order to output a value closer to the correct output. Convolutional neural networks are a type of neural network that are typically used for image classification.

3.2.2 Technology used

The main libraries used are Keras, Tensorflow, and Numpy. Keras and Tensorflow are used for creating and training machine learning models, and Numpy is used for linear algebra calculations and creating numpy arrays. Other libraries used include Pandas, Nibabel, Sklearn, and Matplotlib. The programming language used was Python and the machine learning model was designed and evaluated in Google Colab.

The model architecture used is commonly referred to as the U-Net. The U-Net is a machine learning architecture for biomedical segmentation that includes a contracting path and an expanding path. The contracting path includes a series of convolutional layers, ReLu activation functions (the Relu activation function, for a given value x, returns x if x is greater than 0, and 0 if x is less than or equal to 0), and max pooling layers. The expanding path includes a series of up samplings, up convolutions, and concatenations between the corresponding convolutional layers of the contracting and expanding paths. How one can think of the U-Net is that the U-Net first contracts the image and then expands it into a full segmentation [5].

As shown in Figure 1, the U-Net has a "depth", which refers to the number of convolutional blocks used (each convolutional block has some number of layers). The U-Net in the above image has a depth of 5. To make creating U-Nets easier, we created a function unet() that takes in as input the depth of the U-Net and the number of labels (which in this case is 3). We created 4 U-Net models with one having a depth of 2, another a depth of 3, another a depth of 4, and another a depth of 5. They are saved as model0, model1, model2, and model3 respectively. The models are kept in an array (as to make it easier to iterate through them when training). The models were then trained on the training data.

3.3 Performance of Models

The models were compared to each other on the validation set. Factors used to evaluate them include true positive (the number of positive instances that were correctly predicted), false positive (the number of positive instances that were incorrectly predicted), true negative (the number of negative instances that were correctly predicted), false negative (the number of

negative instances that were incorrectly predicted), accuracy, sensitivity (the proportion of positive instances that were predicted as being positive), specificity (the proportion of false instances that were predicated as being negative), positive predictive value (the probability that a value is `positive when the model predicts it to be positive), negative predictive value (the probability that a value is negative when the model predicts it to be negative), and more. These factors were evaluated for all of the three labels.

In Table 2, Model0 had a low positive predictive value for all of the labels, so it was not picked as being the best model. model1 had a terrible accuracy, specificity, positive predictive value, negative predictive value, and AUC score for non-enhancing tumor, so it was not picked as being the best model. model3 had a terrible accuracy, specificity, positive predictive value, and AUC score for enhancing tumor, so it was not picked as being the best model. This left model2, which performed fairly well on all of the metrics, with its only negatives being a slightly slow sensitivity and F1 score for non-enhancing tumor, however, its sensitivity for non-enhancing tumor was still over 50%, so the model didn't necessarily do terribly in that metric. As such, model2 was the model chosen as being the best model.

In Table 1, Model2's performance on the test set was similar to its performance on the cross validation set. Figures 2 and 3 show an example of model2 segmenting an MRI scan.

3.4 Frontend (GUI) & Database

3.4.1 Main Window

For the Graphical User Interface (GUI), we utilized the library PySimpleGUI, which is a framework for the popular GUI library incorporated in Python: Tkinter.

Our GUI Main Window consists of a button to select an MRI scan, a text box to put a file name, and a Segment button to segment the image. In addition, our program consists of a menu with the item: Account. In this menu item, three buttons appear that link to their respective windows: Signup, Login, and Recently Saved.
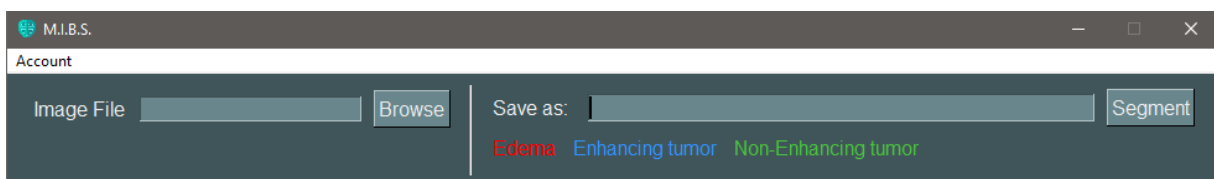
To Segment a file, the user has to select the Browse button and search for an image already downloaded onto their computer. The textbox to the left of the Browse button will show the directory of the image chosen. The user can only select PNG Image types and Numpy arrays.

After the user has successfully chosen an MRI scan to segment, the respective image will display below the row of Text and the Browse button. Next the user must type in a filename to save the file and then hit Segment. If the user hits the Segment button before saving the file, a popup shows up, reminding the user to give the file a designated filename. This then runs through the Artificial Intelligence model and downloads the Segmented file onto the user's local hard disk. If the user has logged in prior to segmenting a file, the segmented MRI scan will automatically be uploaded onto the user's designated storage in the Firebase cloud database. A key showing the different segmented parts of the MRI scan shows up as well to guide the user.

3.4.2 Database Integration

We used Firebase to create the log in system and access a database to store segmentations. Firebase allows us to create an encrypted log in system that allows users to create accounts and access databases which we can add segmentations to and retrieve segmentations from a user.

Using Firebase, our application is able to record new users authenticated signup emails and passwords. The Signup window is as follows:
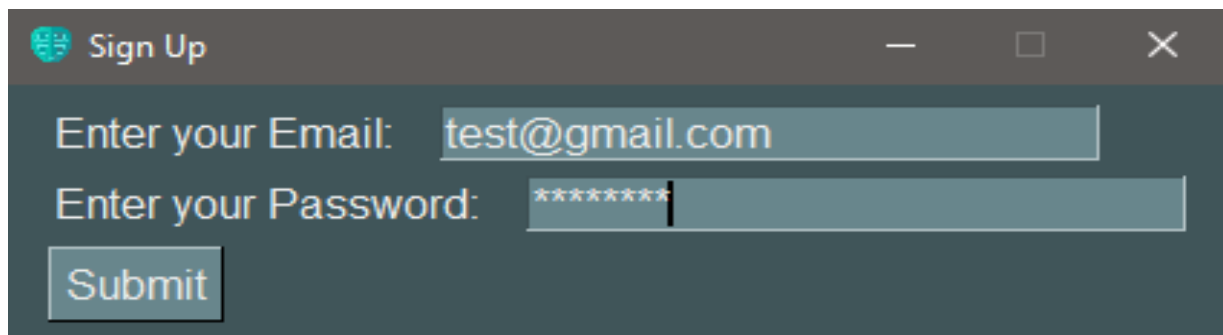
The Signup window allows the user to type in an email and password, which gets starred out with asterisks for privacy. If the user has typed in an email that has already been used, then a

popup shows up displaying this text: "Account already exists". These credentials then go to the Firebase to grant the user authentication and allow the user to access their segmented scans in the cloud.

After the user has been authenticated successfully, the user must login to their account to begin using it. The Login window displays as shown below:
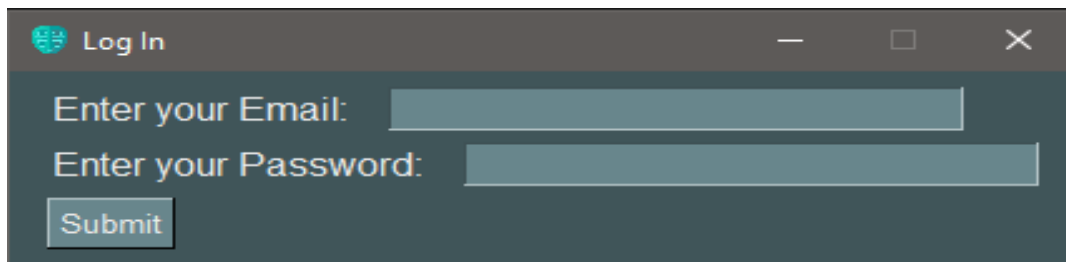

The Login window allows the user to type their email and password, which they used previously to signup, to authenticate their login. If the user types in an invalid email or password, a popup displays the text: "Invalid Email or Password". Once the user hits submit, the authentication will have been completed and the user can begin to save their files onto the cloud storage.

3.4.3 Recently Saved Window

The Recently Saved window can only be accessed if the user has been logged in. If the user has not logged in, a popup displaying this text will show up: "User must be logged in to access recently saved images!". Assuming the user has logged in, this window will show up, prompting the user to select a folder for Segmentations. Essentially, this window is a simple folder viewer, however the Segmentations folder will automatically be created and/or be reset every time the user runs the application. This folder will browse all the segmentations that the user has saved onto the Firebase storage and will download all of them to be displayed here. In detail, every time the user runs a segmentation, the program will send a database storage request to store the name of the file that was sent into the database, so that our code can later access that database storage request and therefore be able to access all the stored file names. With the stored file names, our code will iterate over this list, and download all of the files from the database that the user has stored into this Segmentations folder.

After the user selects the Segmentations folder that our program automatically creates, this window will be updated with the first image in that folder. The user can go back and forward through the folder using the "Prev" and "Next" buttons. This will change the current image that is displaying and the name of the image that the user is currently viewing also shows up. The

user can even select another folder in case the user has multiple folders with segmented files. The Recently Saved window after the user has selected a Segmentations folder is as follows:
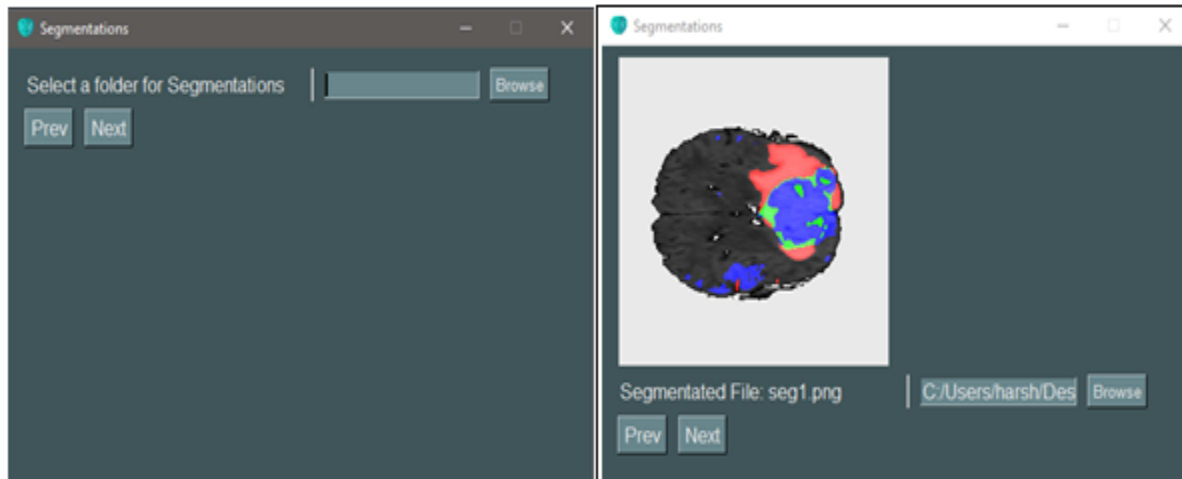
Throughout the M.I.B.S. GUI, we accounted for every use case by creating multiple windows for authentication and popup windows whenever any error occurs to guide the user through the application and to prevent our application from crashing or stopping. With this UI, our main priority was to create a simple environment for the user to segment their MRI scans.
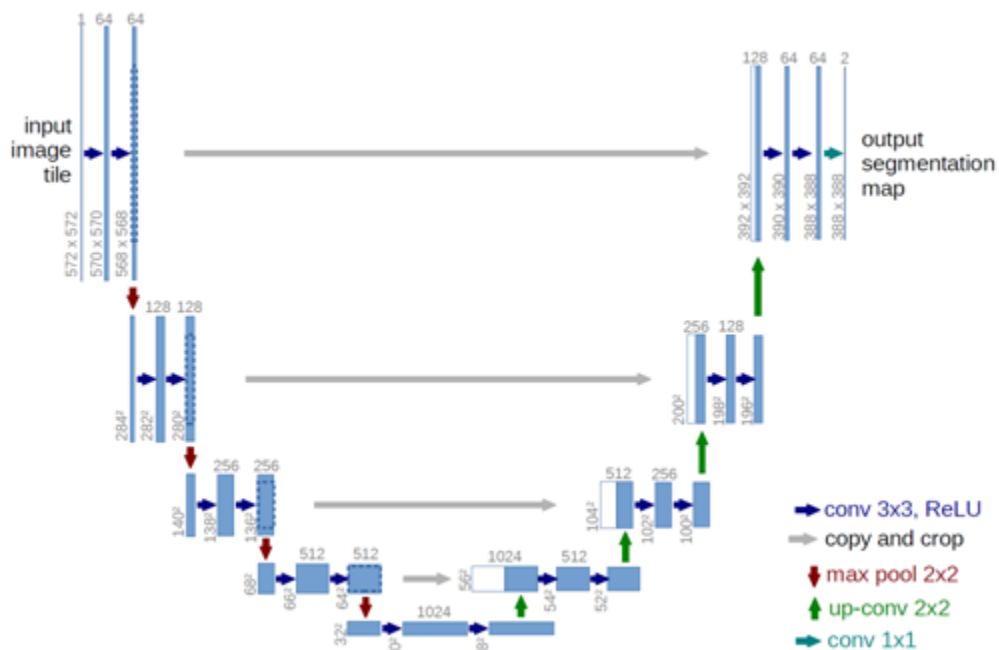
4.1 Figures

Fig. 1 U-Net Architecture

Fig. 2 Predicted Segmentation
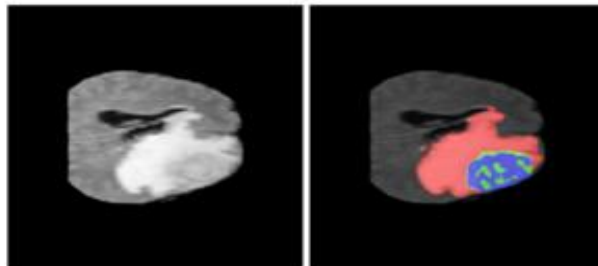
Fig. 3 Actual Segmentation

4.2 Tables

Table 1. Performance of the best model on the test set

model2 (best model)

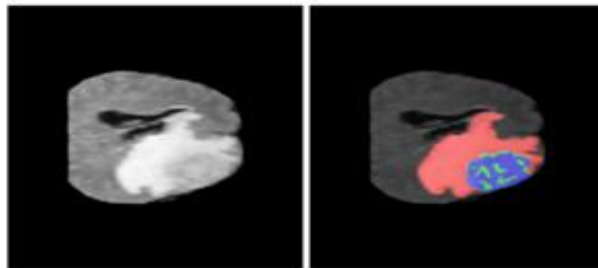|	TP	TN	FP	FN	Accuracy	Prevalence	Sensitivity	Specificity	PPV	NPV	AUC	F1	Threshold
edema	278707	6760771	119830	40692	0.977705	0.044361	0.872598	0.982584	0.699325	0.994017	0.990926	0.776412	0.5
non-enhancing tumor	80896	7023054	32301	63749	0.986660	0.020090	0.559273	0.995422	0.714648	0.991005	0.992924	0.627485	0.5
enhancing tumour	126661	7028088	38726	6525	0.993715	0.018498	0.951008	0.994520	0.765846	0.999072	0.998704	0.848442	0.5

Table 2. Evaluations on 4 different U-Net models

model0:

	TP	TN	FP	FN	Accuracy	Prevalence	Sensitivity	Specificity	PPV	NPV	AUC	F1	Threshold
edema	318517	6384393	485997	11093	0.930960	0.045779	0.966345	0.929262	0.395912	0.998265	0.953068	0.561697	0.5
non-enhancing tumor	134388	6494288	563995	7329	0.920649	0.019683	0.948284	0.920095	0.192427	0.998873	0.943690	0.319933	0.5
enhancing tumour	125266	6885317	183279	6138	0.973692	0.018251	0.953289	0.974071	0.405989	0.999109	0.960882	0.569457	0.5

model1:

	TP	TN	FP	FN	Accuracy	Prevalence	Sensitivity	Specificity	PPV	NPV	AUC	F1	Threshold
edema	287086	6648878	221512	42524	0.963328	0.045779	0.870987	0.967758	0.564465	0.993645	0.980322	0.684999	0.5
non-enhancing tumor	141717	0	7058283	0	0.019683	0.019683	1.000000	0.000000	0.019683	0.000000	0.063099	0.038606	0.5
enhancing tumour	0	7068596	0	131404	0.981749	0.018251	0.000000	1.000000	0.000000	0.981749	0.969889	0.000000	0.5

model2:

	TP	TN	FP	FN	Accuracy	Prevalence	Sensitivity	Specificity	PPV	NPV	AUC	F1	Threshold
edema	291252	6768064	102326	38358	0.980461	0.045779	0.883626	0.985106	0.740011	0.994364	0.992775	0.805467	0.5
non-enhancing tumor	83032	7028368	29915	58685	0.987694	0.019683	0.585900	0.995762	0.735141	0.991719	0.993568	0.652091	0.5
enhancing tumour	126133	7027099	41497	5271	0.993504	0.018251	0.959887	0.994129	0.752449	0.999250	0.998794	0.843603	0.5

model3:

	TP	TN	FP	FN	Accuracy	Prevalence	Sensitivity	Specificity	PPV	NPV	AUC	F1	Threshold
edema	326876	6612106	258284	2734	0.963747	0.045779	0.991705	0.962406	0.558610	0.999587	0.980383	0.714663	0.5
non-enhancing tumor	135809	6610128	448155	5908	0.936936	0.019683	0.958311	0.936507	0.232564	0.999107	0.952581	0.374294	0.5
enhancing tumour	94408	635446	6433150	36996	0.101369	0.018251	0.718456	0.089897	0.014463	0.944983	0.662793	0.028355	0.5

5. Conclusion

MIBS could be used in hospitals around the world. Specifically, in developing countries, MIBS is a solution for those that cannot afford expensive medical equipment and time. These resources are extremely important for those in critical conditions regarding brain cancer and with improvements on the applicability of our data, MIBS can be the solution. In the future, we plan to access 3D MRIs as it will yield more accurate results, and, thus, treatment for the patient.

Since our data isn't applicable to all hospitals, MIBS, at this point, cannot properly serve everyone, but, with steady improvements in our model and more representation and collection of the data it runs on, our software will definitely be able to one day provide quick diagnosis of cancer leading to faster treatment at early stages of cancer, possibly saving a life.

References

[1] MRI for cancer. American Cancer Society. (n.d.). Retrieved September 27, 2022, from https://www.cancer.org/treatment/understanding-your-diagnosis/tests/mri-for-cancer.html

[2] Inderscience. (2009, September 27). Tired Doctors Make More Mistakes. ScienceDaily. Retrieved September 26, 2022 from
 www.sciencedaily.com/releases/2009/09/090914111307.htm

[3] Developer. (2021, September 28). Facts about mris. Envision Radiology. Retrieved September 27, 2022, from https://www.envrad.com/facts-about-mris/

[4] Medical Segmentation Decathlon. (n.d.). Retrieved September 27, 2022, from http://medicaldecathlon.com/

[5] Net: Convolutional networks for biomedical image segmentation. (n.d.). Retrieved May 01, 2022 from https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/