Semantic Segmentation of MRI Brain Scans

ECGR 4106 & 5106 - Introduction to Deep Learning

Timothy Niven
UNC Charlotte
Charlotte, NC
rniven1@charlotte.edu

Cristian Salitre
UNC Charlotte
Charlotte, NC
csalitre@charlotte.edu

Aiden Lamar
UNC Charlotte
Charlotte, NC
alamar2@charlotte.edu

***Abstract*— In order to segment images of brain MRI scans to reveal likely tumor locations, machine learning algorithms can be utilized to great effect. This project aims to segment images of brain scans to identify likely areas to have tumors using multiple variations of convolutional neural networks. This project focuses on an encoder to decoder architecture, which is critical for reconstructing images after their features are brought into the latent space of the network. After providing the encoder with many examples of tumor brain scans, and then masking parts of the image, the model is trained to predict pixels likely to be tumors, and represent this with masking in the reconstruction created by the decoder.**

I. Introduction

Introduction to deep neural networks has been a class shaped around deep learning in the aspects of advanced CNNs, RNNs, Transformers, and GNNs. This project was introduced as a way for students to apply their knowledge learned throughout the course in one fail swoop. The team decided the best way to apply this knowledge was through Semantic Segmentation of MRI Scans. This was a way to not only apply the concepts learned in the course but do something that would be extremely useful in the future. This could not only be applied to brain imaging but heart and other organ imaging techniques.

II. Motivation

Use semantic segmentation to determine the location of a tumor on an MRI scan. The goal is to accurately determine the location of a brain tumor in the MRI Brain scan image. The use of semantic segmentation will allow categorizing each pixel in an image into a class or object. This will allow the regions to separate cancerous and non-cancerous tissues. The broader impact of this MRI scanning model is that it could be used on various other MRI imaging and detect abnormalities, and provide healthcare professionals with valuable information needed to treat patients. This would be useful in the medical field when training new doctors and simplify the hours of work doctors have to do to visually detect a brain tumor. Not only this but this application could be used for various types of tumors and abnormalities in the body.

III. Approach

The first approach to this project was finding which methods worked best. Semantic Segmentation was something the team was set on using. But what networks worked best for this? Would it be basic RNN's or something more advanced like GRU models or skip connections. Diving into this, the team developed all of these algorithms and ran them simultaneously against each other to get a true direct comparison. After the initial testing each individual group member took one of the models and was tasked with optimizing it. This not only showed the baseline use but how far each model could truly be pushed

IV. Dataset and Training Setup

A. Dataset

The dataset for this project was acquired through Kaggle. The dataset was prelabeled and organized in a way to make it very simple to run test, train, and validation sets. Each folder had an _annations.coco.json file which contained important information that corresponded to the images in the dataset such as ID and categories that correspond to 'tumor' and 'no tumor'.

The use of libraries and tools such as OpenCV and PIL were utilized to properly mask the images in order to correctly isolate the area where the tumor was located. Creating a separate directory for the masked images.

Transforms were applied to the dataset, such as random flips, rotations and normalization through torchvision library. Doing so helps with creating noise in the training, to help with generalization and prevent the model from memorizing instead of learning.

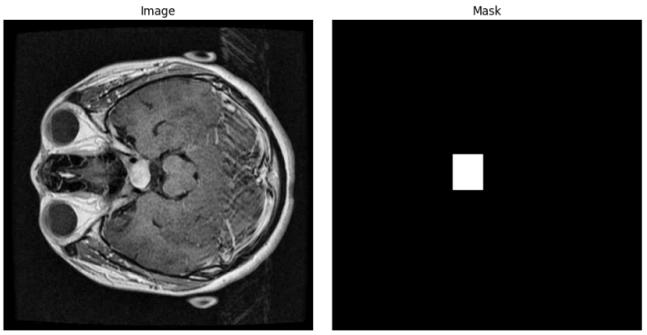

Figure 1: Dataset Composition

B. Training Setup

Training setup was fairly straightforward with all models being run through a similar pipeline in regards to feeding the training data, constructing the model and training in accordance to past encounters in homeworks. Then testing the model simultaneously on the validation set and at the very end seeing the test accuracy with visuals which will be broken down in the next few sections

V. Results and Analysis

A. *Simple CNN Encoder-Decoder Neural Network*

The first model we tested was a simple CNN model with encoder and decoder architecture.. The encoder consists of 2 conv2d layers that increase the feature space while reducing the image resolution to 64x64 pixel size in the model's latent space. The decoder consists of 2 transposed conv2d layers that increase the spatial resolution back to the original input size, in this case 256x256, which represents a reconstruction of the image with a predicted mask. Through this process, the model learns from the most prominent features in the feature map and applies it to test data.

The DiceLoss function seemed to segment the regions, slightly better than the BCELoss function when we ran an inference. Training this model on different numbers of epochs up to 50 epochs. The models seemed to converge and stop learning after 20 epochs, with both los functions.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 256, 256]             320
              ReLU-2         [-1, 32, 256, 256]               0
         MaxPool2d-3         [-1, 32, 128, 128]               0
            Conv2d-4         [-1, 64, 128, 128]          18,496
              ReLU-5         [-1, 64, 128, 128]               0
         MaxPool2d-6           [-1, 64, 64, 64]               0
   ConvTranspose2d-7         [-1, 32, 128, 128]           8,224
              ReLU-8         [-1, 32, 128, 128]               0
   ConvTranspose2d-9          [-1, 1, 256, 256]             129
         Sigmoid-10          [-1, 1, 256, 256]               0
================================================================
Total params: 27,169
Trainable params: 27,169
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.25
Forward/backward pass size (MB): 63.00
Params size (MB): 0.10
Estimated Total Size (MB): 63.35
```

Figure 2: CNN Model Architecture

In Figure 3, the loss and accuracy history for both train and validation set is shown per epoch. We can observe that the loss drops significantly after 20 epochs and becomes more stable which relates to the model learning better after that point in training. It can also be seen that the learning stagnates a bit after epoch 40, which shows there is not much use in running it for longer. We found that by training the model for longer we could avoid it becoming biased by the imbalance in our data. This imbalance refers to the fact that the vast majority of pixels in each image was not tumor, as tumors usually took up only around 2%-5% of the image. This meant that since we were classifying pixels individually, guessing no tumor across the image would deliver very high accuracy and low loss. training the model longer and forcing it to recognize further patterns in the latent space seemed to solve this issue.

Figure 3: Loss and Accuracy History of simple CNN

Figure 4: Confusion Matrix

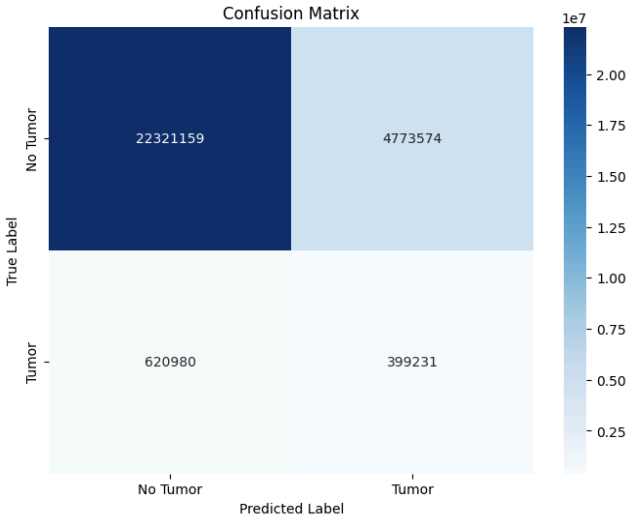

Table 1: Metrics for Simple CNN Encoder Decoder

Metrics	Precision	Recall	F1 Score
Validation	0.077	0.391	0.129

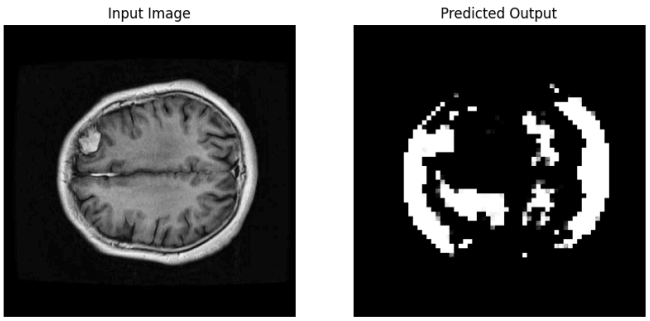

Figure 5: Inference

B. *Deep CNN Encoder-Decoder Neural Network*

The second model adds another layer, this will decrease the latent space to 32x32 pixel size. Allowing the model to extract more prominent information from the feature map. This significantly increases the model complexity from 27,000 trainable parameters to 133,000 in total.

For this model, we obtained better results with the BCELoss function. It properly segments the areas of significance.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 256, 256]             320
              ReLU-2         [-1, 32, 256, 256]               0
         MaxPool2d-3         [-1, 32, 128, 128]               0
            Conv2d-4         [-1, 64, 128, 128]          18,496
              ReLU-5         [-1, 64, 128, 128]               0
         MaxPool2d-6           [-1, 64, 64, 64]               0
            Conv2d-7           [-1, 128, 64, 64]          73,856
              ReLU-8           [-1, 128, 64, 64]               0
         MaxPool2d-9           [-1, 128, 32, 32]               0
  ConvTranspose2d-10            [-1, 64, 64, 64]          32,832
             ReLU-11            [-1, 64, 64, 64]               0
  ConvTranspose2d-12         [-1, 32, 128, 128]           8,224
             ReLU-13         [-1, 32, 128, 128]               0
  ConvTranspose2d-14           [-1, 1, 256, 256]             129
          Sigmoid-15           [-1, 1, 256, 256]               0
================================================================
Total params: 133,857
Trainable params: 133,857
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.25
Forward/backward pass size (MB): 76.00
Params size (MB): 0.51
Estimated Total Size (MB): 76.76
```

Figure 5. Deep CNN Model Architecture

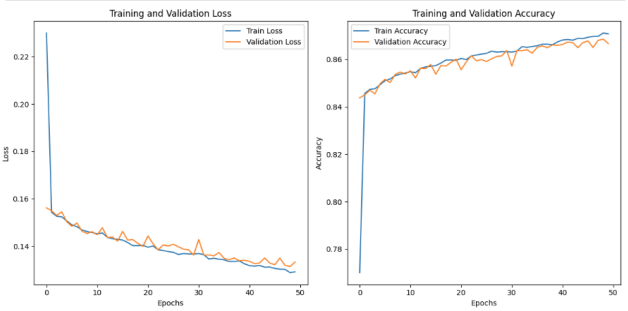

Figure 6. Loss and Accuracy of Deep CNN

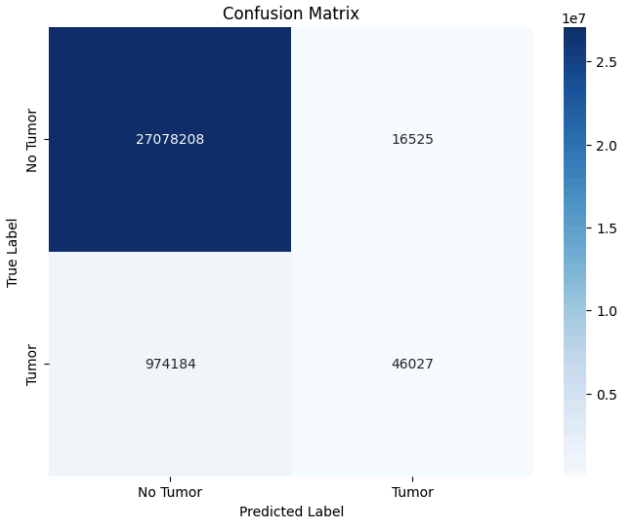

Figure 7. Confusion Matrix

Table 2: Metrics for Deep CNN Encoder Decoder

Metrics	Precision	Recall	F1 Score
Validation	0.736	0.045	0.085

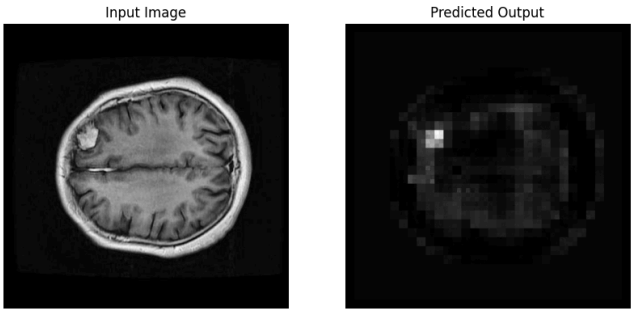

Figure 6: Inference

C. ResNet CNN Encoder-Decoder Neural Network

The third and final model has a significantly increased complexity, and adds residual layers to help with vanishing gradients which will allow feature patterns to propagate through the latent space and ideally allow the model to generalize better. After receiving feedback during the class presentation, this model with residual blocks was implemented in place of the simple CNN in the encoder decoder architecture.

We struggled the most with this model, although loss was low and the accuracy was high, the model did not perform as desired. To help improve the model, we performed extensive hyperparameter tuning. We experimented with layer sizes, learning rates, batchnorm and dropout, but it seemed to always have similar results and would end up predicting masks across the entire image. This can be seen in Figure 11

After doing more research we came across the BCEWithLogitsLoss function which implements a sigmoid activation function. With that we did not need a sigmoid layer at the end of our model. We also experimented with BCELoss and DiceLoss functions, but to no avail. It seems the issue is not the loss function but the complexity and architecture of our model, along with the imbalanced data.

```
----------------------------------------------------------------
      Layer (type)               Output Shape         Param #
================================================================
          Conv2d-1         [-1, 32, 256, 256]             320
     BatchNorm2d-2         [-1, 32, 256, 256]              64
          Conv2d-3         [-1, 32, 256, 256]           9,248
            ReLU-4         [-1, 32, 256, 256]               0
       Dropout2d-5         [-1, 32, 256, 256]               0
          Conv2d-6         [-1, 32, 256, 256]           9,248
            ReLU-7         [-1, 32, 256, 256]               0
    ResidualBlock-8         [-1, 32, 256, 256]               0
       MaxPool2d-9         [-1, 32, 128, 128]               0
         Conv2d-10        [-1, 128, 128, 128]          36,992
    BatchNorm2d-11        [-1, 128, 128, 128]             256
         Conv2d-12        [-1, 128, 128, 128]         147,584
           ReLU-13        [-1, 128, 128, 128]               0
      Dropout2d-14        [-1, 128, 128, 128]               0
         Conv2d-15        [-1, 128, 128, 128]         147,584
           ReLU-16        [-1, 128, 128, 128]               0
    ResidualBlock-17        [-1, 128, 128, 128]               0
       MaxPool2d-18          [-1, 128, 32, 32]               0
  ConvTranspose2d-19         [-1, 32, 128, 128]          65,568
    BatchNorm2d-20         [-1, 32, 128, 128]              64
         Conv2d-21         [-1, 32, 128, 128]           9,248
           ReLU-22         [-1, 32, 128, 128]               0
      Dropout2d-23         [-1, 32, 128, 128]               0
         Conv2d-24         [-1, 32, 128, 128]           9,248
           ReLU-25         [-1, 32, 128, 128]               0
    ResidualBlock-26         [-1, 32, 128, 128]               0
  ConvTranspose2d-27          [-1, 1, 256, 256]             129
================================================================
Total params: 435,553
Trainable params: 435,553
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.25
Forward/backward pass size (MB): 293.50
Params size (MB): 1.66
Estimated Total Size (MB): 295.41
----------------------------------------------------------------
```

Figure 8. ResNet Architecture.

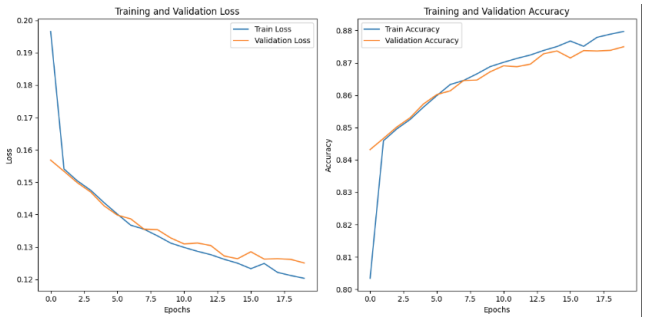

Figure 9. Loss history

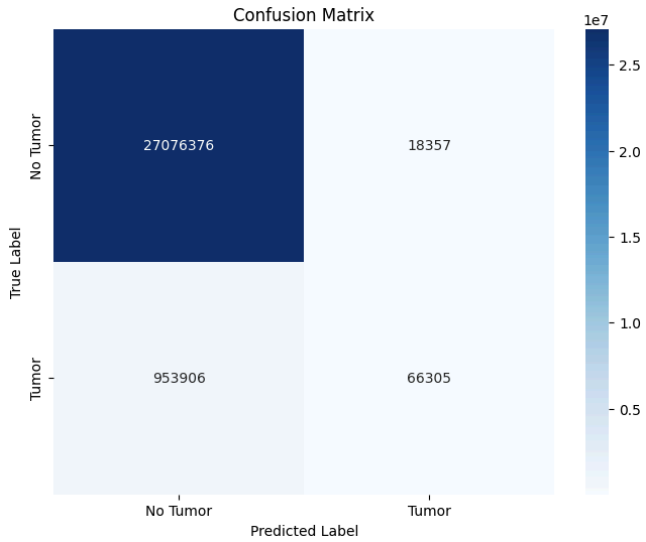

Figure 10. Confusion matrix

Table 3: ResNet Encoder Decoder Metrics

Metrics	Precision	Recall	F1 Score
Validation	0.717	0.055	0.103

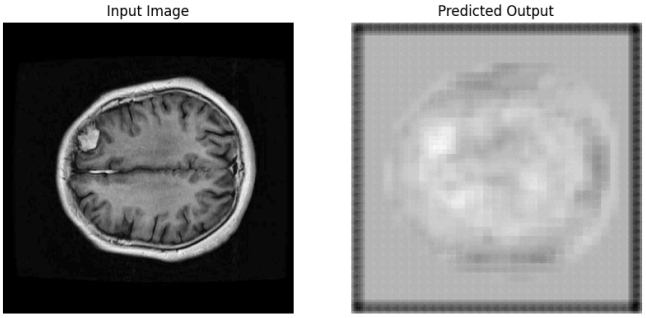

Figure 11: Inference

D. Results Summary

The results of this project did not turn out as expected. Metrics wise, simple CNN had the highest Recall score, which is the target metric of our problem, since not missing a tumor is more important than a false positive. The traditional CNN models did well in recognizing tumors but had issues with brighter lights within the images. Early on the models would just create an image mask of the brain but now it is just lighting up the area it needs to identify.

On the other hand, the ResNet model which we performed extensive hyperparameter tuning on, did not do well in segmenting the areas where the tumors are located. We tried many model sizes in order to find the cause of its poor performance. This included decreasing the model significantly to only 8,000 parameters and did not perform well. We then also increased the model complexity up to 600,000+ parameters in hopes that the model is too simple for segmentation. The results were similar, it seems as if the model is paying more attention to the non-tumor areas, which is very prominent.

In the end, we have come to believe that a more complex network such as ResNet may have bee;p. able to improve performance with proper tuning, that a simple convolutional network is best for the homogeneity and low feature size of our data. This data has an input image pixel size of 256x256 and only 1,500 images that are gray scale. It seems that traditional CNN with encoder decoder architecture is sufficient for this task.

VI. Lessons Learned

One of the biggest lessons learned in this project was to be open to obstacles. Various obstacles came into play such as image size, training time, and other hindrances that didn't allow us to fully accomplish all goals we set for ourselves. Although the team felt very good about what we were able to accomplish, especially with the simple CNN encoder decoder, there is still much more exploring to do within this topic.

Having a balanced data is crucial to achieving readable and useful results. A lot of time was spent on hyperparameter tuning and due to the time constraints we were not able to attempt to add to do more preprocessing.

GITHUB

[1] https://github.com/Salitre97/ECGR5105_Final_Report

[2] P.K.Darabi,"Brain Tumor Image Dataset"- Semantic Segmentation",2023.[Online].Available: https://www.kaggle.com/datasets/pkdarabi/brain-tumor-image-dataset-semantic-segmentation
[Accessed: 29-03-2024]

[3] F. Author, "Understanding and Visualizing ResNets," Towards Data Science, [Online]. Available: https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8
[Accessed: 29-03-2024]

[4] R.Yang, , "Artificial NeuralNetworks,"
Frontiers in Oncology, vol. 11, Art. no. 638182, 2021. [Online]. Available : https://www.frontiersin.org/journals/oncology/articles/10.3389/fonc.2021.6 38182/full. [Accessed: 03-29, 2024].