JOHNS HOPKINS
UNIVERSITY

Engineering for Professionals

Predicting Tumors in MRI Brain Scans Using UNet Architecture

525.733 Deep Learning for Computer Vision - Final Project

Written by:
Ian DeTore

Supervised by:
Prof. Nasser M. Nasrabadi

Contents

Abstract

This paper illustrates our implementation of the UNet deep learning architecture to predict image segmentation masks on tumors detected in the brain through MRI scans. The training sets and UNet architecture are publicly available via Kaggle and GitHub. To further encourage open-source use and increase the speed of training, we modified the code to perform training on images reduced to 64x64 pixels instead of 256x256, which may exhaust GPU resources. The UNet model successfully predicted tumors with a binary accuracy up to 99 percent, and showed phenomenal results for all other object detection metrics.

Keywords: *computer vision, deep-learning, image segmentation, MRI, UNet*

1 Introduction

1.1 Overview

In the twenty-first century, computer vision has grown prominent in a diverse set of industries, including (but not limited to) automotive, robotics, criminology, entertainment, and biomedical. More specifically, the ability to train artificial intelligence to interpret what it "sees" has augmented civilization through autonomous vehicles, recognizing the faces of criminals and multitudes more. In this paper, our primary focus is to utilize deep learning for computer vision within the biomedical realm, specifically detecting brain tumors in a 2000+ image database of MRI brain scans. Unlike other popular object-detection models like YOLO, we aim to score the model's accuracy based on the precise segmentation of the tumor instead of using a bounding box. Cutting the segmentation provides an extra challenge since the model needs to detect if there's a tumor and accurately draw the mask, as shown in Figure 1.

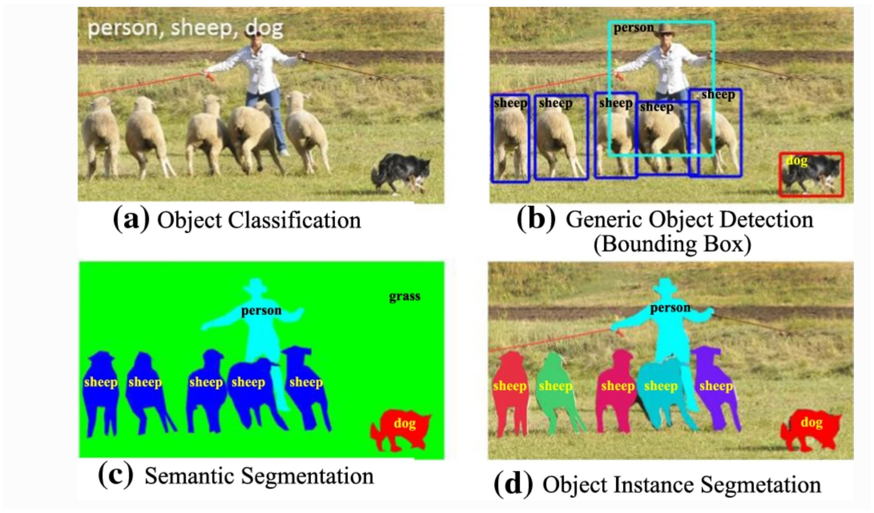

person, sheep, dog
(a) Object Classification
person
sheep
dog
(b) Generic Object Detection (Bounding Box)
grass
person
sheep
dog
(c) Semantic Segmentation
person
sheep
dog
(d) Object Instance Segmetation

Figure 1: Example of Object Detection vs Segmentation

[1]

1.2 Brain Tumors and MRIs

Tumors come in various shapes and sizes, which can directly correlate to the severity and maliciousness of cancer. The World Health Organization categorizes brain tumors into four groups of increasing risk to the patient [2]:

1. **Grade I:** *Tumors do not meet any of the criteria. These tumors are slow growing, nonmalignant, and associated with long-term survival*
2. **Grade II:** *Tumors meet only one criterion, i.e., only cytological atypia. These tumors are slow growing but recur as higher-grade tumors. They can be malignant or nonmalignant*
3. **Grade III:** *Tumors meet two criteria, i.e., anaplasia and mitotic activity. These tumors are malignant and often recur as higher-grade tumors*
4. **Grade IV:** *Tumors meet three or four of the criteria, i.e., showing anaplasia, mitotic activity with microvascular proliferation, and/or necrosis. These tumors reproduce rapidly and are very aggressive malignant tumors.*

Fortunately, Magnetic Resonance Imaging, or *MRI*, has made the detection of such cancers easier to diagnose. MRI is a non-invasive imaging technique that produces three-dimensional images, primarily for anatomical purposes. Imaging via MRI is possible by using powerful magnets to create a strong magnetic field that forces protons in the body to align with that field. Radio frequency current is then pulsed through the patient, the protons are stimulated, and spun out of equilibrium, straining against the pull of the magnetic field [3].

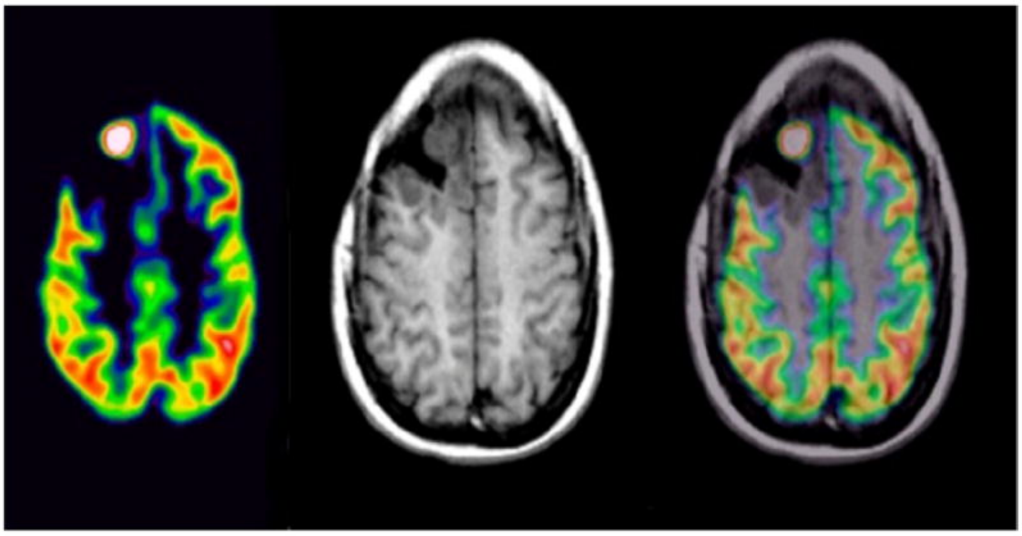


Figure 2: Example of imaging the brain via MRI and PET scans
[4]

1.3 Training Data

The MRI dataset we used is freely available on Kaggle, a cloud-based Jupyter Notebook that allows users to share datasets publicly and use the website's kernels to build and share their machine learning code with other members. We specifically used the MRI images uploaded by Navoneel Chakrabarty of Jalpaiguri Government Engineering College [5]. The dataset comes with numerous slices of a patient's MRI and a manually-drawn binary mask for any photos that contain a tumor, as shown in Figures 3 and 4. These masks will be used as our ground-truth for our model training.

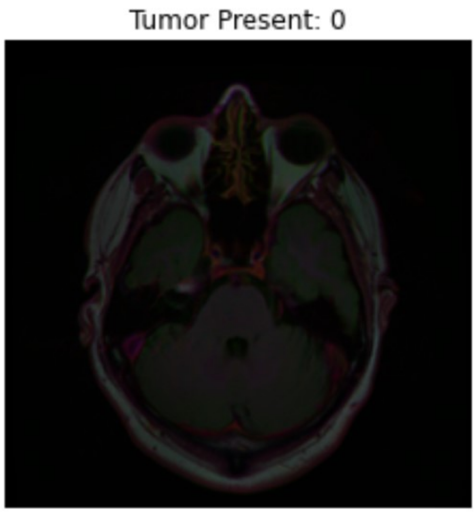


Figure 3: MRI Result without Tumor

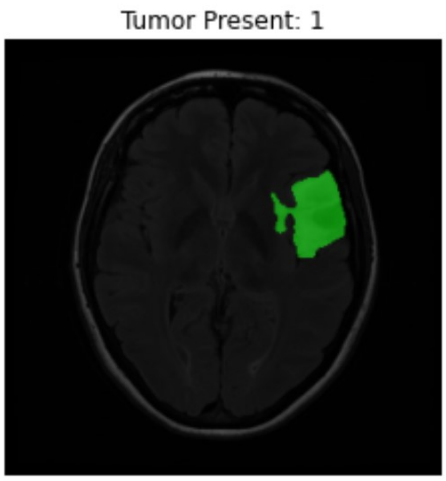


Figure 4: MRI Result with Tumor

2 Architecture

2.1 Overview

As of today, the UNet Convolutional Neural Network (CNN) remains one of the most reliable and top-performing models for biomedical image segmentation. It was developed in by Prof. Olaf Ronneberger, along with his research team, at the University of Freiburg in 2015. In that same year, it won first place at the *ISBI Grand Challange for Computer-Automated Detection of Caries in Bitewing Radiography* [6] [7].

2.2 Model

The entire network architecture is illustrated in Figure 5. Referring to this figure, the left side is referred to as the *"contracting path"* since the images are being down-sampled via 2x2 max-pooling. In contrast, the right side of the network is referred to as the *"expansive path"* since the images are being up-sampled by 2x2 convolutions [6]. The model goes through five stages of up-scaling and down-scaling, each with two 3x3 convolutions with a ReLU activation functions before max-pooling or up-conversion. For all stages besides the fifth (last) layer, the resulting output of the second convolution layer from the contracting path is concatenated with the up-converted result from the layer below. For example, the third-stage contracting output is concatenated with the third-stage expansive output which came from the up-sampling of the fourth stage output. The fifth and final layer does not concatenate with any other layers and acts as the bridge between the two paths. As obviously portrayed in the figure below, this architecture was named *U-Net* due to its shape as the letter "U" in the alphabet. In our model, we added batch-normalization after each convolution layer. The UNet architecture has been developed into a Keras model and can be easily accessed on GitHub [8].

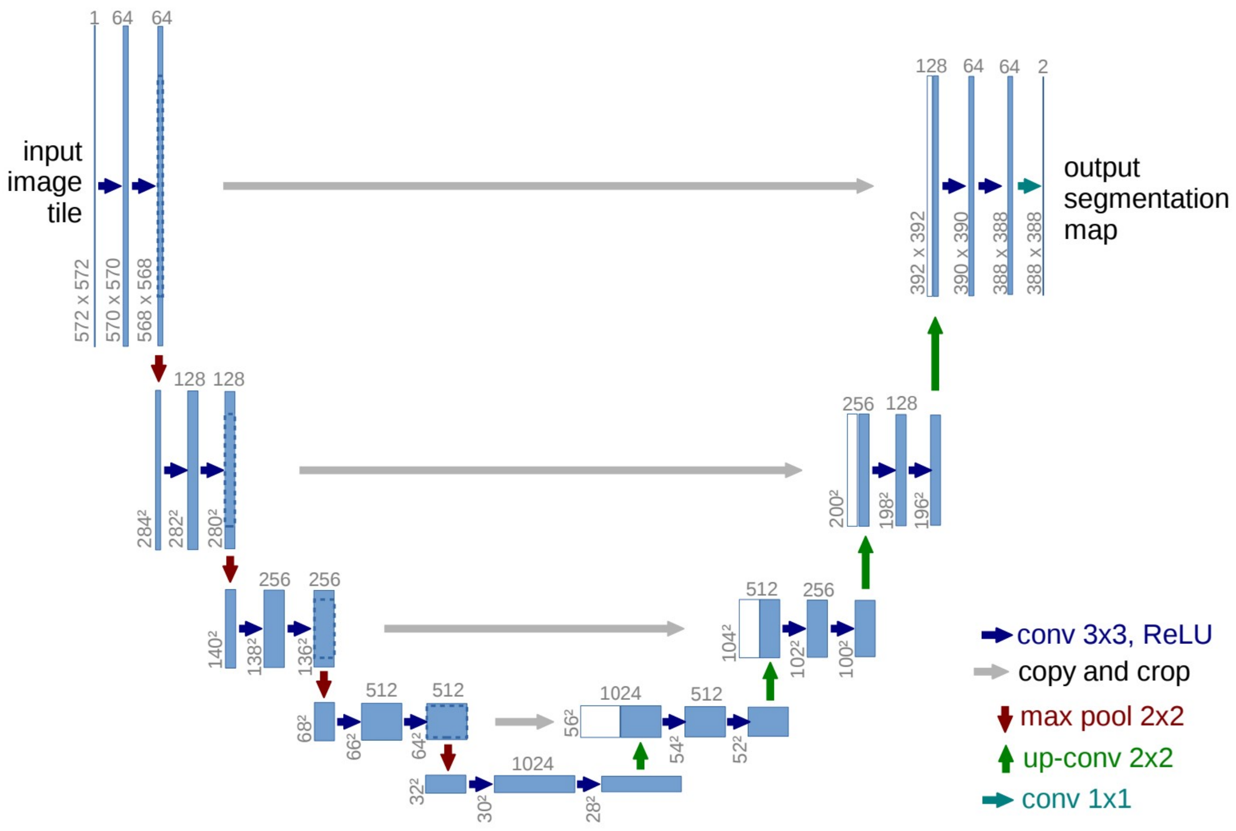


Figure 5: Illustration of Ronneberger's UNet model

3 Training

3.1 Metrics

There are several metrics we use to determine the validity of our training data. Although only one of them is required for training, we wanted to show how each one can be used in our training:

3.1.1 Intersection over Union (IoU)

Also referred to as the Jaccard Index, the Intersection over Union (IoU) metric is the most common computer vision metric for determining the area of overlap for both the ground-truth and predicted segment over the union of both segments. The values range between 0 and 1, where 0 signifies no overlap, while 1 denotes total overlap [9]. For two samples A and B, the IoU can be written as the following:

$$IoU = \frac{|A \bigcap B|}{|A \bigcup B|} \tag{1}$$

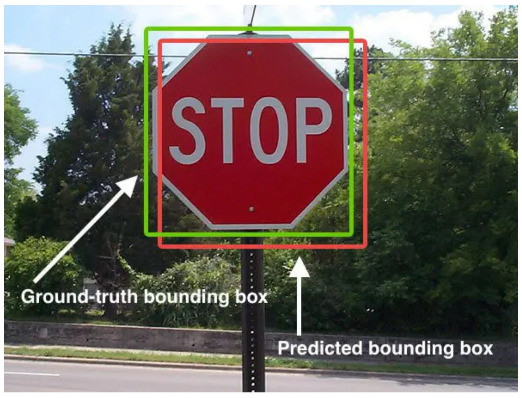


Figure 6: Bounding Box Example [10]

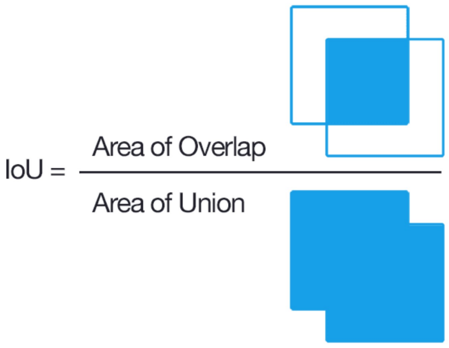


Figure 7: Illustration of IoU equation [10]

3.1.2 Dice Similarity Coefficient

Similar to the Jaccard Index, the Dice Similarity Coefficient (DSC) is another metric used to determine the spatial overlap of two samples ranging from 0 to 1 [11]. Given two data sets A and B, we can write the following equation:

$$DSC = \frac{2|A \bigcap B|}{|A| + |B|} \tag{2}$$

We can also add a smoothing coefficient, typically a positive integer, to our equation [8]:

$$DSC = \frac{2|A \bigcap B + S|}{|A| + |B| + S} \tag{3}$$

As a matter of fact, the DSC is very similar to the IoU metric. However, the DSC is a better indicator for the average performance of the overlap, while the IoU metric is better for measuring the worst case performance of the model.

3.1.3 Tversky index

Lastly, the Tversky is the metric we used to determine the relationship between our data in a confusion matrix [12]. For clarity, we can determine how the correct, positive results (or *True Positives*) compare to incorrect results, like *False Positives* or *False Negatives*. The equation for this is as follows with α set to 0.7:

$$TV = \frac{TP}{TP + \alpha FN + (1-\alpha)FP} \tag{4}$$

Again, we can add the smoothing coefficient to this equation [8]:

$$TV = \frac{TP + S}{TP + \alpha FN + (1-\alpha)FP + S} \tag{5}$$

3.2 Training Plots

We trained our network via the *Adam* optimizer using a learning rate of 0.01 and an epsilon value of 0.1. After training our model for 60 epochs, we obtain the set of plots below. Notice that all the metrics follow a similar trend and that the validation data is prone to bouncing around compared to the training data. For 2000+ images of 64x64 MRI scans, the training process took approximately 1.5 hours to run.

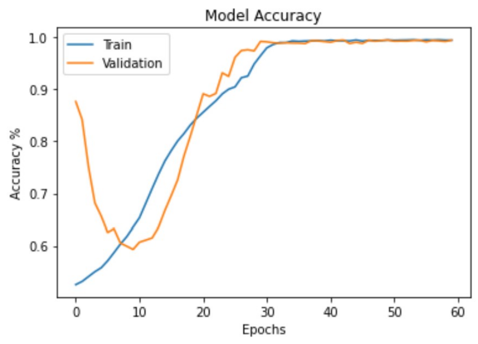


Figure 8: Training Accuracy after 60 Epochs

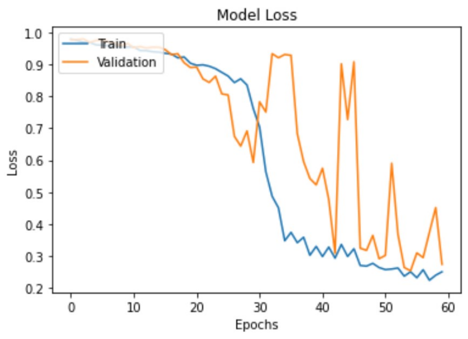


Figure 9: Training Loss after 60 Epochs

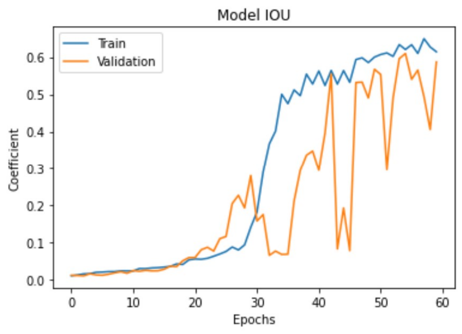


Figure 10: Training IOU after 60 Epochs

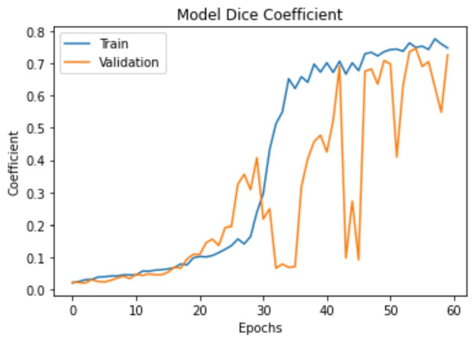


Figure 11: Training DSC after 60 Epochs

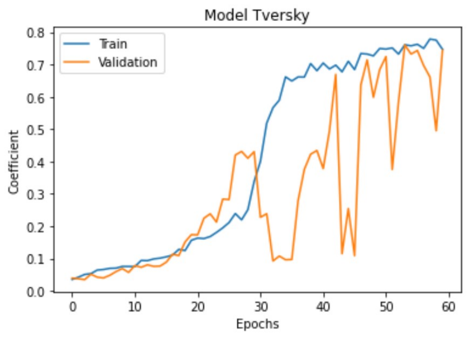


Figure 12: Training Tversky after 60 Epochs

4 Results

Table 1: Model Evaluation Results

Binary Accuracy [%]	Loss	IoU	DSC	Tversky
99.50%	0.2576	0.6029	0.7344	0.7390

Referring to table above, we can conclude that our model performed exceptionally well for running at 60 epochs. The only flaw was that by reducing the image sizes from 256x256, the fine detail in the tumor mask traces would become lost, making our prediction segmentation look more coarse than the ground truth. However, we can see that the tumor split into two areas was correctly determined by the prediction model in Figure 13. If a tumor was not present in the ground truth, then the model would not predict any segmentation, as seen in the first row of Figure 14. Sometimes, the model would experience False Positives for very small regions, as shown in the last row of the same figure.

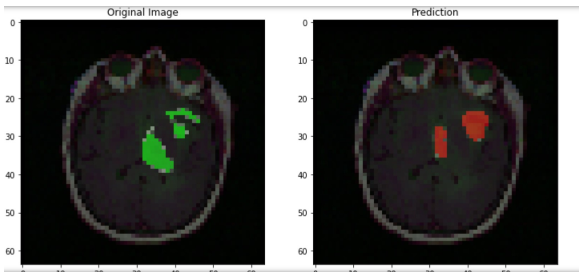


Figure 13: Ground Truth *(left)* vs Prediction Segmentation *(right)* for tumor split in two places

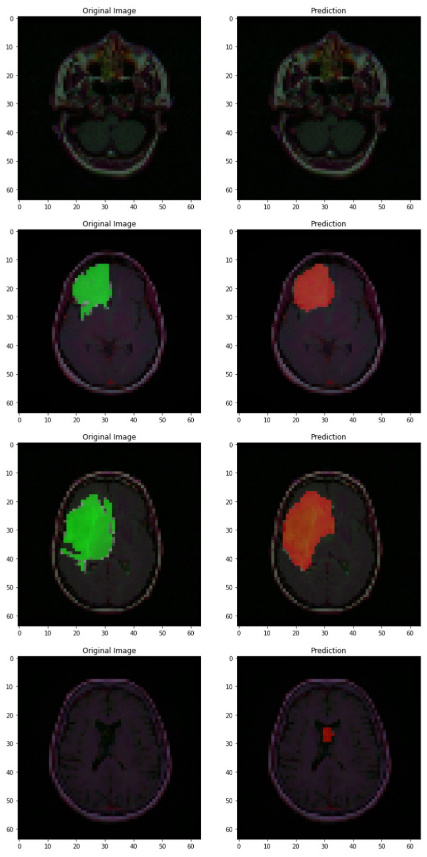


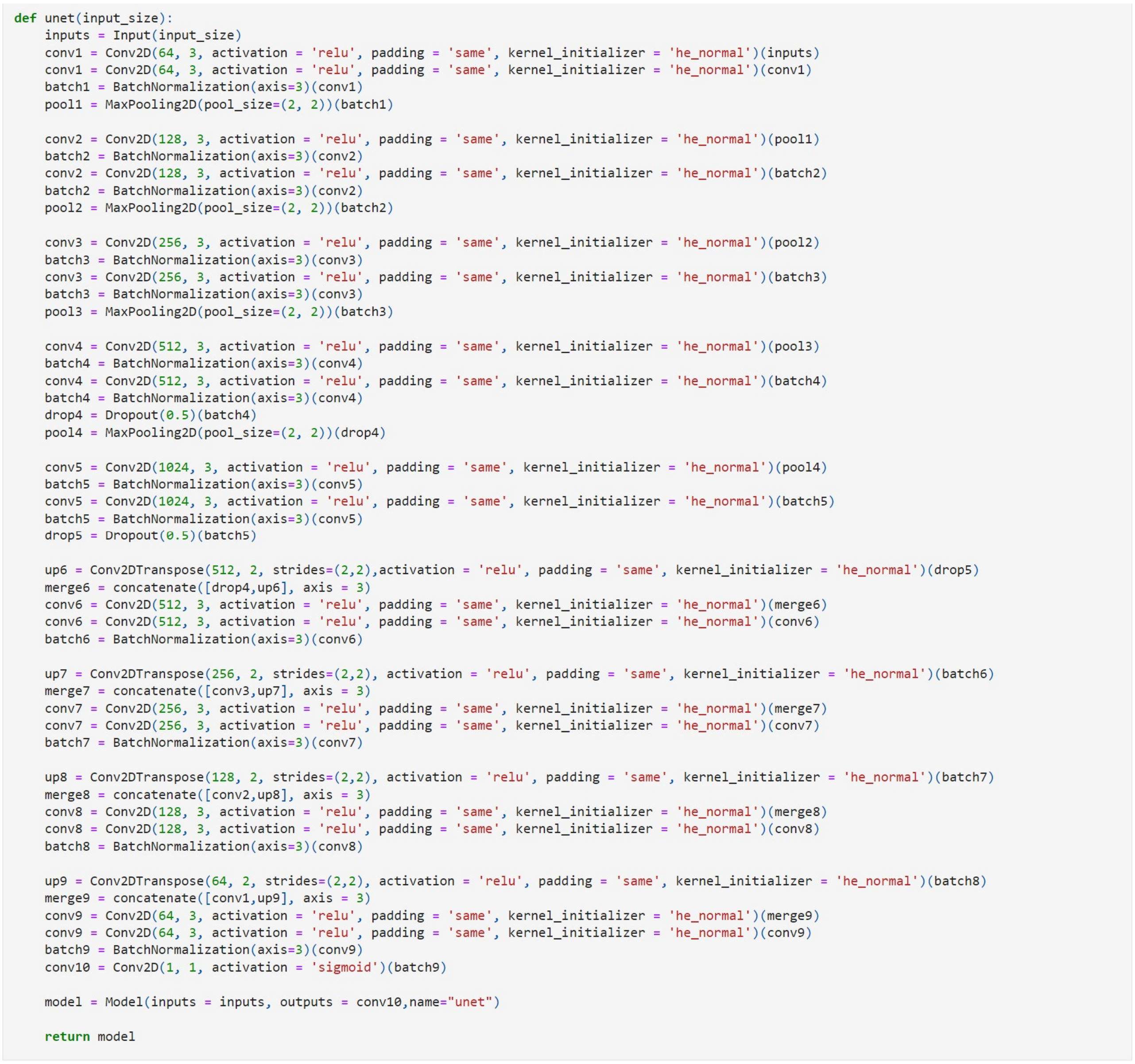
Figure 14: Ground Truth *(left)* vs Prediction Segmentation *(right)* for four randomly selected images

5 Discussion

The MRI dataset is an excellent free resource for learning machine learning. Due to hardware limitations and time constraints, we could not utilize this dataset and the UNet to its full potential. The results remain promising, but we hope to improve our run and obtain better outcomes for our data at 256x256 resolution soon. Additionally, we'd like to train and output our data so that we can slice through each patient's MRI scan and view the tumors in three-dimensions.

6 Code

The entire Jupyter Notebook for our code can be found in our GitHub repo [13]. All code we have included from external sources has been properly cited.



```python
def unet(input_size):
    inputs = Input(input_size)
    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(inputs)
    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv1)
    batch1 = BatchNormalization(axis=3)(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(batch1)

    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool1)
    batch2 = BatchNormalization(axis=3)(conv2)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(batch2)
    batch2 = BatchNormalization(axis=3)(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(batch2)

    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool2)
    batch3 = BatchNormalization(axis=3)(conv3)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(batch3)
    batch3 = BatchNormalization(axis=3)(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(batch3)

    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool3)
    batch4 = BatchNormalization(axis=3)(conv4)
    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(batch4)
    batch4 = BatchNormalization(axis=3)(conv4)
    drop4 = Dropout(0.5)(batch4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(drop4)

    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool4)
    batch5 = BatchNormalization(axis=3)(conv5)
    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(batch5)
    batch5 = BatchNormalization(axis=3)(conv5)
    drop5 = Dropout(0.5)(batch5)

    up6 = Conv2DTranspose(512, 2, strides=(2,2),activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(drop5)
    merge6 = concatenate([drop4,up6], axis = 3)
    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge6)
    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv6)
    batch6 = BatchNormalization(axis=3)(conv6)

    up7 = Conv2DTranspose(256, 2, strides=(2,2), activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(batch6)
    merge7 = concatenate([conv3,up7], axis = 3)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge7)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv7)
    batch7 = BatchNormalization(axis=3)(conv7)

    up8 = Conv2DTranspose(128, 2, strides=(2,2), activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(batch7)
    merge8 = concatenate([conv2,up8], axis = 3)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge8)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv8)
    batch8 = BatchNormalization(axis=3)(conv8)

    up9 = Conv2DTranspose(64, 2, strides=(2,2), activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(batch8)
    merge9 = concatenate([conv1,up9], axis = 3)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge9)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv9)
    batch9 = BatchNormalization(axis=3)(conv9)
    conv10 = Conv2D(1, 1, activation = 'sigmoid')(batch9)

    model = Model(inputs = inputs, outputs = conv10,name="unet")

    return model
```

Figure 15: Entire UNet Model written in Tensorflow Keras

References

[1] Li Liu et al. “Deep Learning for Generic Object Detection: A Survey”. In: *International Journal of Computer Vision* 128 (2020), pp. 261–318. DOI: https://doi.org/10.1007/s11263-019-01247-4.

[2] Tanima Dwivedi Anshu Gupta. “A Simplified Overview of World Health Organization Classification Update of Central Nervous System Tumors 2016”. In: *Journal of Neurosciences in Rural Practice* 8,4 (2017), pp. 629–641. DOI: doi:10.4103/jnrp.jnrp_168_17.

[3] U.S. Department of Health and Human Services. *Magnetic Resonance Imaging (MRI)*. URL: https://www.nibib.nih.gov/science-education/science-topics/magnetic-resonance-imaging-mri. (accessed: 05.08.2022).

[4] Karl Herholz et al. “Brain Tumors”. In: *Seminars in Nuclear Medicine* 42,6 (2012), pp. 356–370. DOI: 10.1053/j.semnuclmed.2012.06.001.

[5] Navoneel Chakrabarty. *Brain MRI Images for Brain Tumor Detection*. 2019. URL: https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection. (accessed: 05.08.2022).

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

[7] IEEE International Symposium on Biomedical Imaging. *Grand Challenges in Dental X-Ray Image Analysis*. 2015. URL: http://www-o.ntust.edu.tw/~cweiwang/ISBI2015/challenge2/index.html. (accessed: 05.08.2022).

[8] zhixuhao. *UNet Repo*. URL: https://github.com/zhixuhao/unet. (accessed: 05.08.2022).

[9] Scikit Learn. *Scikit Learn Jaccard Score API*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.jaccard_score.html. (accessed: 05.08.2022).

[10] Adrian Rosebrock. *Intersection over Union (IoU) for object detection*. 2016. URL: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/. (accessed: 05.08.2022).

[11] Kelly H Zou et al. “Statistical Validation of Image Segmentation Quality Based on a Spatial Overlap Index”. In: *Academic Radiology* 11,2 (2004), pp. 178–189. DOI: doi:10.1016/s1076-6332(03)00671-8.

[12] Amos Tversky. “Features of Similarity”. In: *Psychological Review* 84,4 (1977), pp. 327–352. DOI: 10.1037/0033-295X.84.4.327.

[13] Ian DeTore. *Brain Tumor UNet Repo*. URL: https://github.com/idetoreStarry/brain_tumor_unet. (accessed: 05.08.2022).