A Comparison of Altered Convolutional Filter Prevalence for U-Net in MRI Brain Tumor Segmentation

1st Brian Sharber
Department of Computer Science
Middle Tennessee State University
Murfreesboro, Tennessee USA
bws2u@mtmail.mtsu.edu

2nd Lucas Remedios
Department of Computer Science
Middle Tennessee State University
Murfreesboro, Tennessee USA
lwr2k@mtmail.mtsu.edu

3rd Christine Monchecourt
Department of Computer Science
Middle Tennessee State University
Murfreesboro, Tennessee USA
cfm3a@mtmail.mtsu.edu

4th David Woods
Department of Computer Science
Middle Tennessee State University
Murfreesboro, Tennessee USA
dmw6c@mtmail.mtsu.edu

5th Joshua Ortner
Department of Computer Science
Middle Tennessee State University
Murfreesboro, Tennessee USA
jo3f@mtmail.mtsu.edu

6th Joshua LaFever
Department of Computer Science
Middle Tennessee State University
Murfreesboro, Tennessee USA
jrl5z@mtmail.mtsu.edu

***Abstract*—Computer-aided diagnosis (CAD) is an approach that uses software to enhance radiologist interpretations of medical images. The goal of our project is to design a CAD software system to detect the pixel locations of tumors in 2D slices of 3D brain MRIs. To this end, we will train a deep convolutional neural network to perform the desired tumor segmentation. Our model will output a 2D image explicitly showing which pixels in a given 2D brain MRI slice it has predicted to be exhibiting a tumor.**

***Index Terms*—Neural Networks, Convolutional Neural Networks, CNN, Python, Keras**

I. Introduction

The heart of our software approach will be a convolutional neural network (CNN) — this will be the tool to enable the segmentation of tumors. CNNs are specialized neural networks that are commonly applied to analyzing images; they use filters to scan an entire image left-to-right and top-to-bottom to determine image features. For this project, a deep CNN will be utilized for our medical image analysis. The input to the CNN will consist of brain images, in the form of 2D matrices of pixels, which will be processed for tumor segmentation.

Our specific deep CNN architecture will be U-net, a popular architecture for this type of complex medical image segmentation task. We will implement our network using the Keras library for the Python programming language.

II. Background

A. Data

The dataset utilized for this project is the BraTS 2019 data, and can be acquired at the "Registration" page at http://www.braintumorsegmentation.org/. The following is a description of the dataset, provided by the website: "Ample multi-institutional routine clinically-acquired pre-operative multimodal MRI scans of glioblastoma (GBM/HGG) and lower grade glioma (LGG), with pathologically confirmed diagnosis and available OS, are provided as the training, validation and testing data for this year's BraTS challenge. Specifically, the datasets used in this year's challenge have been updated, since BraTS'18, with more routine clinically-acquired 3T multimodal MRI scans, with accompanying ground truth labels by expert board-certified neuroradiologists".

The following is a description of the imaging data: "All BraTS multimodal scans are available as NIfTI files (.nii.gz) and describe a) native (T1) and b) post-contrast T1-weighted (T1Gd), c) T2-weighted (T2), and d) T2 Fluid Attenuated Inversion Recovery (T2-FLAIR) volumes, and were acquired with different clinical protocols and various scanners from multiple (n=19) institutions, mentioned as data contributors here. All the imaging datasets have been segmented manually, by one to four raters, following the same annotation protocol, and their annotations were approved by experienced neuroradiologists. Annotations comprise the GD-enhancing tumor (ET — label 4), the peritumoral edema (ED — label 2), and the necrotic and non-enhancing tumor core (NCR/NET — label 1), as described both in the BraTS 2012-2013 TMI paper and in the latest BraTS summarizing paper (also see Fig.1). The provided data are distributed after their pre-processing, i.e. co-registered to the same anatomical template, interpolated to the same resolution (1 mm^3)$and skull - stripped$".

The following is a description of the survival data: "The overall survival (OS) data, defined in days, are included in a comma-separated value (.csv) file with correspondences to the pseudo-identifiers of the imaging data. The .csv file also includes the age of patients, as well as the resection status. Note that only subjects with resection status of GTR (i.e., Gross Total Resection) will be evaluated, and you are

only expected to send your predicted survival data for those subjects".

The values of the pixel intensities in this data set vary widely in magnitude and the images are from one of four different modalities. Because of that its important to normalize the data to limit the range of possible values that may bias the network and so that images from different modalities are seen through the same lens. The images were normalized by patient and modality. First we found the max pixel intensity of a single modality from a single patient and divided all of the images pixel intensities by that max value. This ensures that the range of all values is between 0 and 1 and the images are scaled based on the entire modality.

B. Image Segmentation

The task of semantic segmentation is to predict the class of each pixel in an image. The prediction output shape will match the input's spatial resolution (height and width) with a channel depth, consisting of a binary mask that labels areas where a specific class is present, equivalent to the possible number of classes to be predicted.

Evaluating a semantic segmentation can be done in a couple of different ways. One such way is using the Dice coefficient, which is utilized as a loss function during training. The Dice Coefficient can be explained as "2 * the Area of Overlap divided by the total number of pixels in both images". This is similar to IoU (Intersection over Union) which "measures the number of pixels common between the target and prediction masks divided by the total number of pixels present across both masks".

To calculate the precision of a collection of predicted masks, compare each predicted mask with each available target masks for a given input. Observation-wise, a true positive is observed when "a prediction-target mask pair has an IoU score which exceeds some predefined threshold". "A false positive indicates a predicted object mask had no associated ground truth object mask". "A false negative indicates a ground truth object mask had no associated predicted object mask".

We'll need to define what a positive detection is in our model output in order to calculate the precision and recall of it. To do this, "we'll calculate the IoU score between each (prediction, target) mask pair and then determine which mask pairs have an IoU score exceeding a defined threshold value". Precision refers to the "purity of our positive detections relative to the ground truth. Of all of the objects that we predicted in a given image, how many had a matching ground truth annotation?". Recall refers to the "completeness of our positive predictions relative to the ground truth. Of all of the objects annotated in our ground truth, how many did we capture as positive predictions?".

C. UNet

UNet, inspired from traditional convolutional neural networks, was first designed to process biomedical images in 2015. Whereas a general convolutional neural network focuses its task on image classification, UNet is dedicated to solving the biomedical imaging problem of not only distinguishing whether the patient has a disease, but also localizing the area of abnormality. "The reason it is able to localise and distinguish borders is by doing classification on every pixel, so the input and output share the same size."

The UNet architecture is symmetric, consisting of two major parts: "the left part is called contracting path, which is constituted by the general convolutional process; the right part is expansive path, which is constituted by transposed 2d convolutional layers(you can think it as an upsampling technic for now)".

The contracting path consists of this formula: conv_layer1>conv_layer2>max_pooling>dropout(optional). "Each process constitutes two convolutional layers, and the number of channel changes from 1 to 64, as convolution process will increase the depth of the image. The red arrow pointing down is the max pooling process which halves down size of image (the size reduced from 572x572 to 568x568 is due to padding issues, but the implementation here uses padding = "same")". The process is repeated three more times; two convolutional layers are built with no max pooling.

The expansive path consists of this formula: conv_2d_transpose>concatenate>conv_layer1>conv_layer2. In this path, the image will be upsized to original size. After transposed convolution, an upsampling technique that expands image size, "the image is upsized from 28x28x1024 to 56x56x512, and then, this image is concatenated with the corresponding image from the contracting path and together makes an image of size 56x56x1024. The reason here is to combine the information from the previous layers in order to get a more precise prediction".

Reaching the uppermost architecture, the last step is to "reshape the image to satisfy our prediction requirements". This entire process makes a strong enough network able to do good prediction even if your data set is small, utilizing excessive data augmentation techniques.

D. Convolutional Neural Network

A Convolutional Neural Network (CNN) is a Deep Learning algorithm that takes in an input image, assigns priority (learnable biases and weights) to various properties in the image, and differentiates one image from another. These networks require less pre-processing than other classification algorithms and can learn characteristics, whereas with primitive methods the filters would be hand-engineered. This network was inspired by the organization of the Visual Cortex of the human brain and is similar to the connectivity pattern of neurons in the brain.

An image is nothing but a matrix of pixel values, but attempting to flatten a 3x3 matrix into a 9x1 vector for classification purposes will result in an average precision score for prediction of classes in cases of basic binary images and little accuracy for complex images with pixel dependencies. "A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting

to the image dataset due to the reduction in the number of parameters involved and reusability of weights".

Images exist in many color planes – RGB, Grayscale, HSV, etc. As images scale into dimensions of 4K or 8K, things can get computationally intensive. "The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets".

III. Methods

A. Normalization

This is some placeholder text

B. Outlier Detection

In order to eliminate a source of possible noise, it was necessary to examine the dataset for potential outliers. Outliers, in this case, could include images that had portions missing, or something in the image looked different from the rest. To complete this task, we used the MIPAV (Medical Image Processing, Analysis, and Visualization) application developed by the National Institute of Health. Using this application, we were able to open the different modalities for each patient at the same time and link them together so that when one is changed, the rest of modalities change synchronously. We used this functionality to inspect every patient's slices to detect abnormalities in the dataset. Ultimately, there were only a handful of potential outliers detected, and we opted to remove the ones we were unsure about just to be safe.

C. Experiment Design

To test how reducing the number of filters in each convolutional layer in the U-Net affects performance we ran 4 experiments. The experiments are titled DS 1 (full U-Net), DS 2 (U-Net with half the number of filters in each convolutional layer), DS 4 (U-Net with a quarter of the number of filters in each convolutional layer), and DS 8 (U-Net with an eighth of the filters in each convolutional layer). The experiments only varied by the number of filters.

Each experiment consisted of 5 runs with different neural network initializations. These 5 initializations were performed to generate an average sense of how the filter numbers affects performance. We used 60% of the data for training, 20% for validation, and 20% for testing.

D. Metrics

We used the dice coefficient as a way of measuring the performance of the model. A dice coefficient of 1 represents a perfect segmentation, and a dice coefficient of 0 represents a completely failed segmentation. We used dice loss as our loss function, which is 1 - the dice coefficient. Dice loss is bounded between 0 and 1.

E. Training Protocol

For each run (model initialization), a chunk of data of 5 patients was loaded into RAM at a time. The data for each patient consisted of 4 image volumes in 4 different MRI modalities (T1, T1CE, T2, FLAIR). These 4 image volumes were combined into a multimodal image tensor for each patient. The corresponding 5 patients' multimodal image tensors were then stacked and shuffled slice-wise, where each slice is a multimodal slice. 20% of each chunk was used as validation data.

Each neural network takes multimodal image tensor slices as input. We used a batch size of 32 for each chunk of patient data. We had 39 chunks of data per epoch, and trained each initialized model for 20 epochs.

F. Testing Protocol

We tested each initialized model by finding the performance on test patients' multimodal image tensors. Each patients' multimodal image tensor was read in, and the model made a prediction for each slice sequentially in the input tensor. Each input tensor has 155 multimodal slices, each of which has a corresponding dice coefficient value. We stacked all of the predicted masks into a volume and computed a single dice volume-wise against the corresponding mask volume. We then computed the mean dice coefficient for a run by averaging the dice coefficient that was computed for each patient.

IV. Results

A. Training Performance

During training, the experiments performed on average in this order, ranked best to worst: DS 1, DS 8, DS 2, DS 4 (Fig 1) (Fig 2). It is clear that none of the 5 initializations for DS 4 learned. DS 1 had 1 very good initialization.

B. Testing Performance

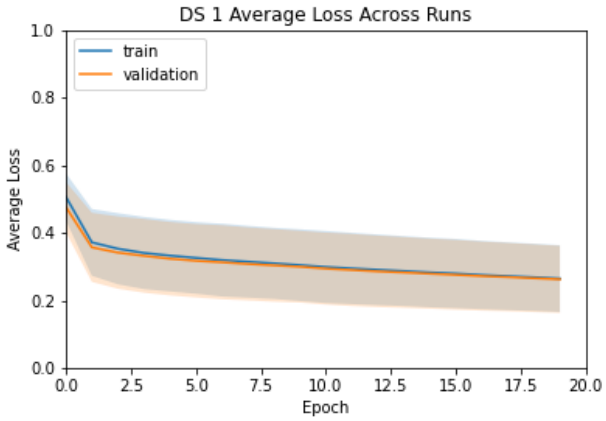
The ordering of best performance on the testing data, ranked best to worst is the same as that of the training: DS 1, DS 8, DS 2, DS 4 (Fig 3).

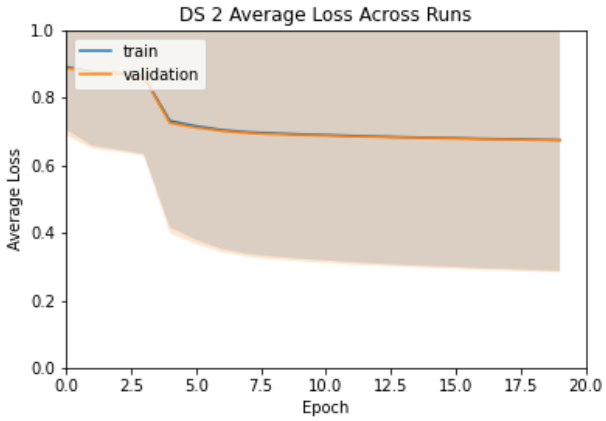
THRESHOLDING OF MASKS DURING TESTING VS NON-THRESHOLDING OF MASKS DURING TRAINING

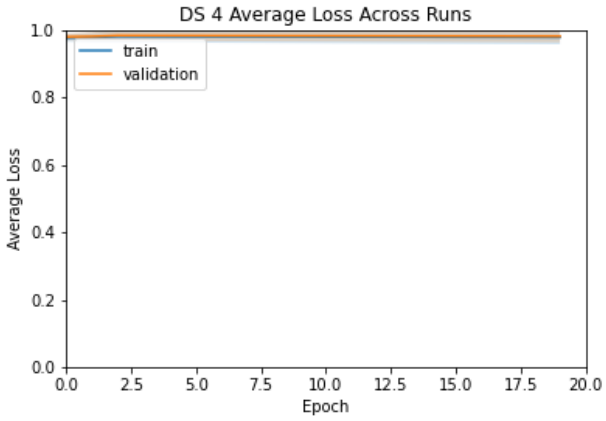
V. Discussion & Conclusion

DS 1 had the best performance, which is of little surprise, as it is the full U-Net. DS 8 performed second best, which is counter-intuitive, as it has an eighth of the filters as DS 1. The expected behavior would have been for the best to worst experiment ordering to have been: DS 1, DS 2, DS 4, DS 8.

DS 1 consistently had good initializations which enabled each run to learn the segmentation function. This was not the case for DS 4. This may expose how having less filters impacts the importance of having a good initialization, however more runs would be needed to verify this. None of the experiments that successfully showed learning behavior had plateaued in performance. This indicates that training the models in each experiment longer could have produced better mode performance.







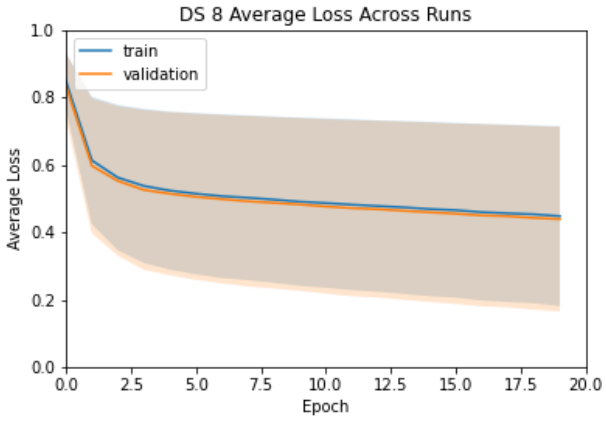
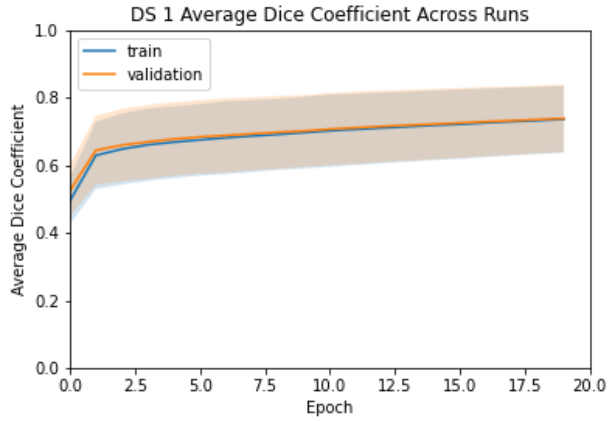
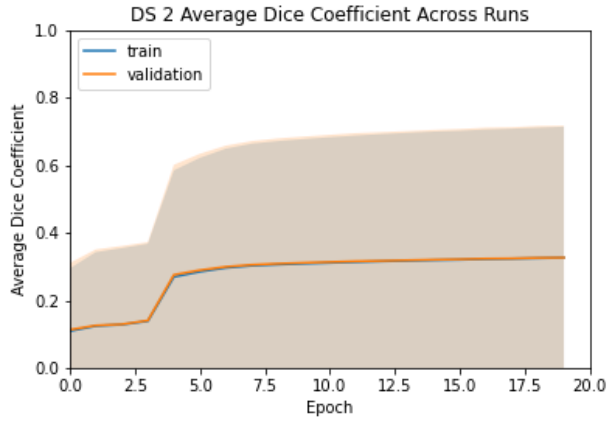
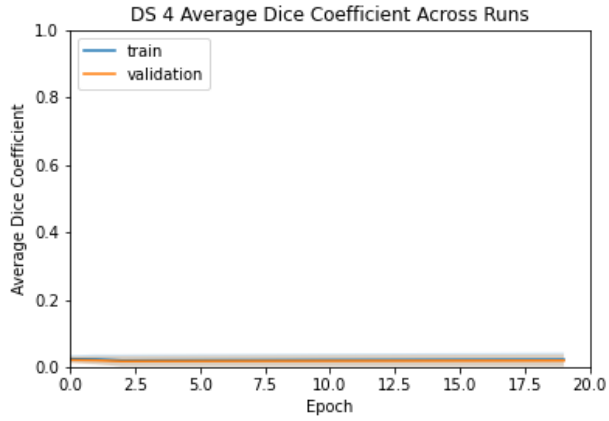


Fig. 1: Training losses for each experiment. Loss is the mean of the loss for each of the 5 initializations for each experiment. The loss for each epoch on each initialization was computed as the average loss for each of the 39 chunks in each epoch.







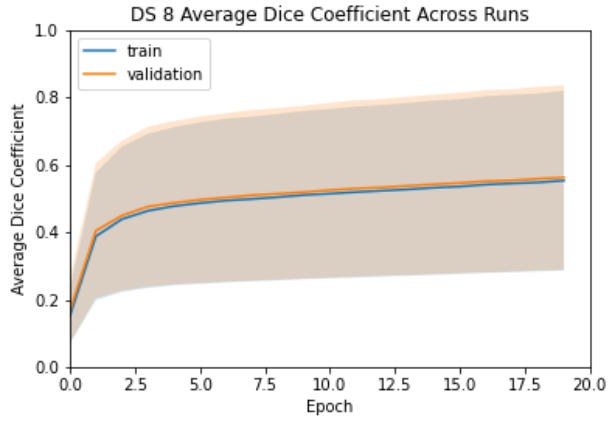
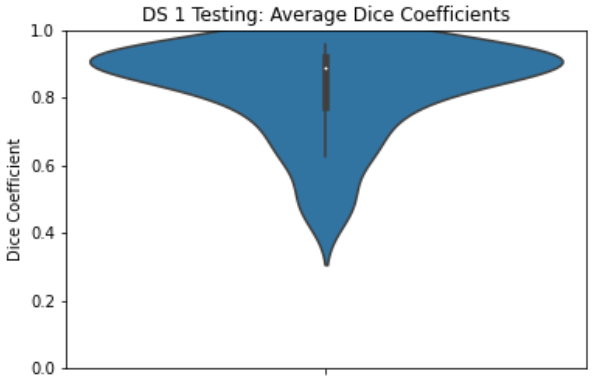
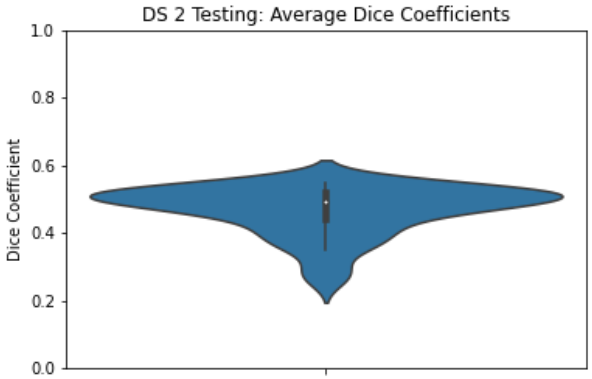
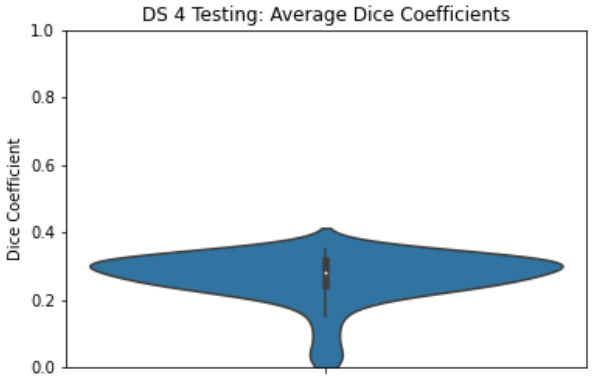


Fig. 2: Training dice coefficients for each experiment. The dice coefficient is the mean of the dice coefficient for each of the 5 initializations for each experiment. The dice coefficient for each epoch on each initialization was computed as the average







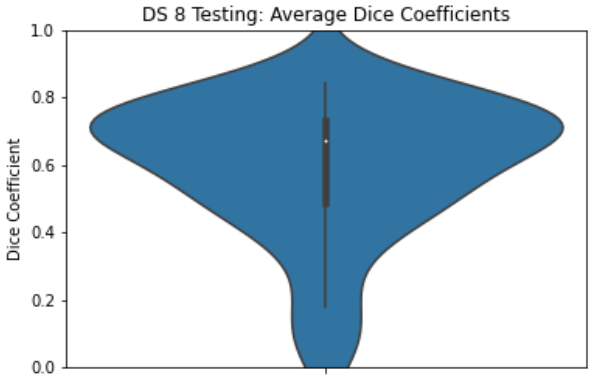


Fig. 3

Future work could include training the models in each experiment longer to see if a larger model, say DS 1, and a smaller model, say DS 8 perform similarly after plateauing in performance.

In conclusion MORE INITIALIZATIONS AND TRAIN LONGER Can segment with less filters Bigger netowrk is probably better but may not be necessary

REFERENCES

[1] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, et al. "The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)", IEEE Transactions on Medical Imaging 34(10), 1993-2024 (2015) DOI: 10.1109/TMI.2014.2377694

[2] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J.S. Kirby, et al., "Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features", Nature Scientific Data, 4:170117 (2017) DOI: 10.1038/sdata.2017.117

[3] S. Bakas, M. Reyes, A. Jakab, S. Bauer, M. Rempfler, A. Crimi, et al., "Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge", arXiv preprint arXiv:1811.02629 (2018)