HIPPOCAMPUS SEGMENTATION FROM BRAIN MRI USING ENSEMBLE DEEP LEARNING METHODS

Ritodeep Sikdar
Department of Computer Science and Engineering
Jadavpur University
Kolkata, India

Kushal Dey
Department of Computer Science and Engineering
Jadavpur University
Kolkata, India

Maqsud Alam Mallick
Department of Computer Science and Engineering
Jadavpur University
Kolkata, India

ABSTRACT

Medical images have made a great impact on medicine diagnosis and treatment. 3D and 2D image segmentation plays an important role in biomedical image analysis. However each segmentation model has its own strengths and weaknesses but by unifying them together we may be able to predict more accurate results. In this project, we present a method to segment biomedical images using an ensemble of deep learning segmentation architectures. We have made general n layer models to ensure scalability of the architectures and segment larger images like that of Lungs or Liver Computed Tomography or Magnetic Resonance Imaging to detect minute tumors and deformities. We have trained three models, namely U-Net, U-Net++ modified by us and ResNet-50 backbone with U-Net++. We have applied our ensemble technique on the Brain MRI Dataset for Hippocampus segmentation which is part of the Medical Image Segmentation Decathlon Challenge.

1 Introduction

1.1 What is medical image segmentation?

Medical image segmentation involves the extraction of regions of interest (ROIs) from 3D image data, such as from Magnetic Resonance Imaging (MRI) or Computed Tomography (CT) scans. The main goal of segmenting this data is to identify areas of the anatomy required for a particular study, for example, to simulate physical properties or virtually positioning CAD-designed implants within a patient. Medical image segmentation can be a time-consuming task, and recent advances in Artificial Intelligence (AI) software techniques are making it easier for routine tasks to be completed. []

1.2 Benefits of medical image segmentation

One of the key benefits of medical image segmentation is that it allows for a more precise analysis of anatomical data by isolating only necessary areas. For certain procedures, such as implant design, it is necessary to segment out certain structures, for example in the hip or knee. In addition, segmentation offers the benefit of removing any unwanted details from a scan, such as air, as well as allowing different tissues such as bone and soft tissues to be isolated. When combined with different software processing options, researchers and clinicians can generate a series of segmented masks that are ready for further analysis.

1.3 How does medical image segmentation work?

When working with CT, MRI, and other types of scans, segmentation generally works by taking information from the background image data and using it to generate a mask. Depending on the task, users may work on their scans in 2D or 3D. There are many different tools and algorithms available in segmentation software, for example fully manual options to paint on the data, or semi-automated operations such as thresholding and region growing. Applications are also available for cardiovascular image segmentation, with particular options for working with different heart cases. For many cases using medical data, it may only be necessary to use a few segmentation tools. As previously noted, studying the placement of medical devices can involve a few steps to segment regions of interest in a bone, which can often be automated using scripts or AI techniques. However, for certain projects, especially those that deal with unusual pathologies or complex traumas, more time and a wider range of software features may be needed to create the required segmentation result. To this end, in software like Synopsys Simpleware, some of the image segmentation tools available to users include:

- Paint/unpaint
- Threshold
- Flood fill
- Interpolation
- Split regions
- Boolean operations

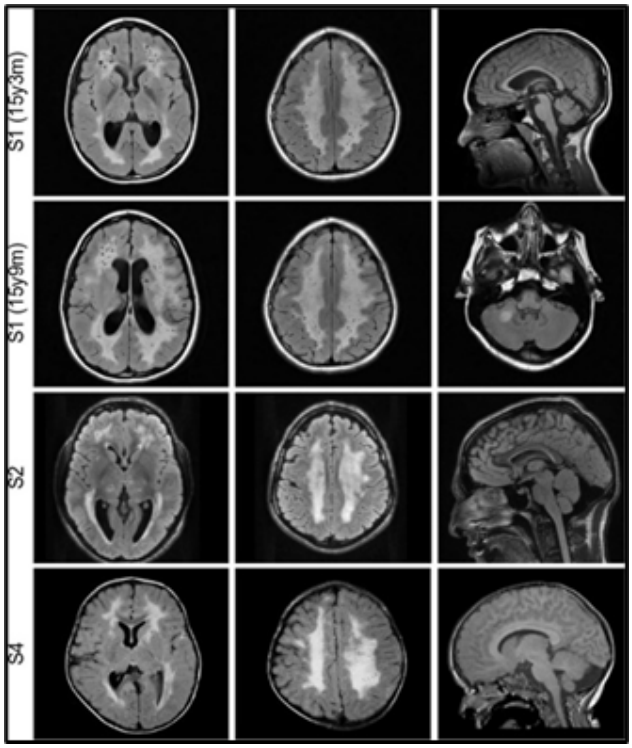

Figure 1: A MRI scan

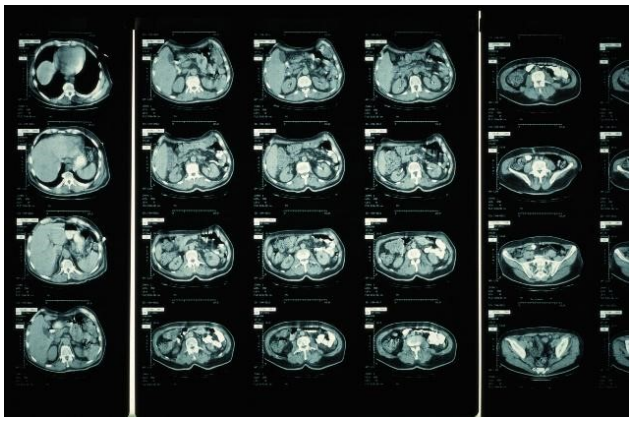

Figure 2: A CT scan

2 Proposed Solution

2.1 Hippocampus Dataset

Hippocampus head and body dataset by Vanderbilt University Medical Centre[1]. It is a Mono-modal MRI dataset. This dataset appeared in the Medical Image Segmentation Decathlon Challenge.

It has 263 .nii files, each a 3D image of hippocampus, every image having about 35 slices, we had a total of 9270 2D images of .jpg format along with their masks. We divided the total set in 80-20 ratio, 80% being used for training, 20% for validation.

We have 131 .nii files which are to be used for testing our model.

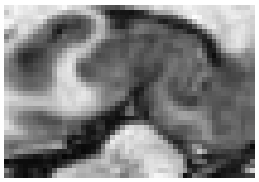

Figure 3: Brain CT Image Slice

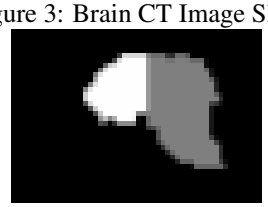

Figure 4: Hippocampus Mask

2.1.1 Data Preprocessing

Every image and mask had a variable dimension and while most of them were around 35x51, we resized all the images and masks to 36x52. We are using a 2-layer model for every architecture for every model we're using (the images are very small, it is really not very meritorious to use a deeper model), so the dimensions of the images will be reduced by half twice, thus they must be multiples of 4 (otherwise, the dimensions of images within the layer won't match and will create problem while concatenating).

Further we applied image and mask transformation in the forms of horizontal and vertical flip and finally normalised all the images to values between 0-1.

2.2 Deep Segmentation Architectures

We have applied three different segmentation architectures stated below with subtle modifications on our dataset to compute the results. We have made an Ensemble of the predictions from these models to get our final results.

2.2.1 U-Net

U-Net is an architecture proposed by Ronneberger et al.,2015[4] for semantic segmentation proposed. It consists of a contracting path and an expansive path. The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for downsampling. At each downsampling step we double the number of feature channels. Every step in the expansive path consists of an upsampling of the feature map followed by a 2x2 convolution ("up-convolution") that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU. The cropping is necessary due to the loss of border pixels in every convolution. At the final layer a 1x1 convolution is used to map each 64-component feature vector to the desired number of classes. In total, the network has 23 convolutional layers. We have modified the U-net architecture by generalizing it for n layers and by taking variable channel size for every layer. So now the network has (5n+3) convolutional layers. [2n convolutions while downsampling 2 convolutions in the bottom layer 2n convolutions while upsampling n up-convolutions 1 final convolution to make the channel size 1].
We have also kept kernel size = 3 and 1 padding layer for each convolution so that the size of processed image is same as that of pre-processed image. Hence, we don't need to crop the images while concatenating them with up-sampled images and the size of final output image will also be same as that of input image, requiring no further resizing of output image.
The total number of trainable parameters in this model are 3,542,145.

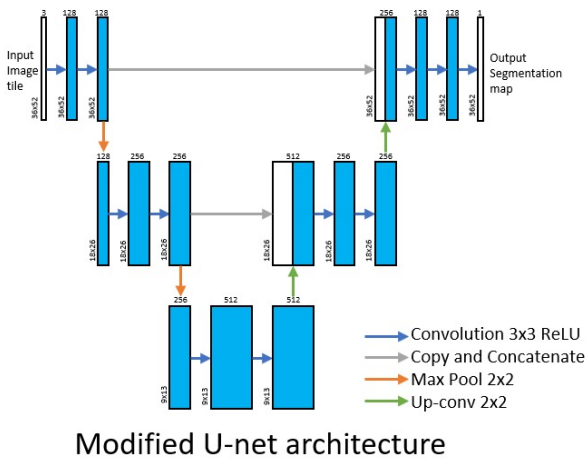

Figure 5: U-Net

2.2.2 Modified U-Net++

U-net++ is an architecture proposed by Zhou et al.,2018[6] for semantic segmentation based on the U-Net. Through the use of densely connected nested decoder sub-networks, it enhances extracted feature processing and was reported by its authors to outperform the U-net in Electron Microscopy (EM), Cell, Nuclei, Brain Tumor, Liver and Lung Nodule medical image segmentation tasks. As seen, U-net++ starts with an encoder sub-network or backbone followed by a decoder sub-network. What distinguishes U-net++ from U-Net is the re-designed skip pathways that connect the two sub-networks and the use of deep supervision. Re-designed skip pathways transform the connectivity of the encoder and decoder sub-networks. In U-Net, the feature maps of the encoder are directly received in the decoder; however, in U-net++, they undergo a dense convolution block whose number of convolution layers depends on the pyramid level. We also used an additional set of up-convolution, convolution and max-pool at the very end of the pyramidal structure. The stack feature maps represented by,

$$x^{(i,j)}$$

is computed by,

$$x^{(i,j)} = H(x^{(i-1,j)}) \qquad , j = 0 \quad (1)$$

$$x^{(i,j)} = H\left(\left[[x^{(i,k)}]_{(k=0)}^{(j-1)}, U(x^{(i+1,j-1)})\right]\right), j > 0 \quad (2)$$

Where H() denotes a convolution, U() denotes a up-convolution, [] denote concatenation. The total number of trainable parameters in this modified U-Net++ model are 3,542,531.

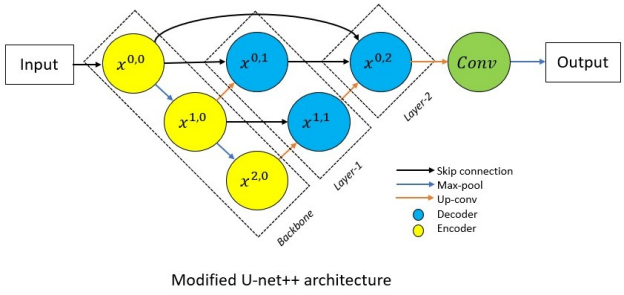

Figure 6: Modified U-Net++ Architecture

2.2.3 U-Net++ with ResNet-50 Backbone

It is a modified version of U-Net++, which uses the convolutional layers of ResNet-50 architecture(He et al., 2016)[3] in it's encoders.

A plain architecture is the one in which the input image is passed through a series of convolutional layers. It is mainly inspired by the philosophy of VGG nets. The convolutional layers mostly have 3×3 filters and follow two simple design rules: (i) for the same output feature map size, the layers have the same number of filters; and (ii) if the feature map size is halved, the number of filters is doubled so as to preserve the time complexity per layer. We perform downsampling directly by convolutional layers that have a stride of 2. The network ends with a global average pooling layer.

A residual network is a modification of plain architecture where we insert shortcut connections which turn the network into its counterpart residual version. The identity shortcuts can be directly used when the input and output are of the same dimensions. When the dimensions increase, we consider two options: (A) The shortcut still performs identity mapping, with extra zero entries padded for increasing dimensions. This option introduces no extra parameter; (B) The projection shortcut in is used to match dimensions (done by 1×1 convolutions). For both options, when the shortcuts go across feature maps of two sizes, they are performed with a stride 2.
The total number of trainable parameters in this model are 66,563.

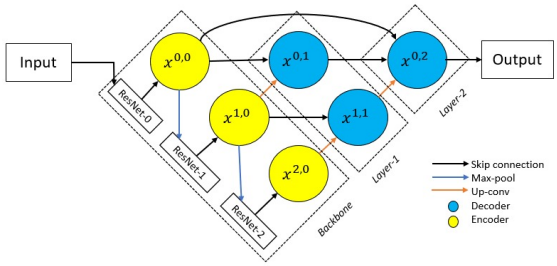

Figure 7: ResNet-50 with U-Net++

2.3 Ensemble of Predictions

Ensemble Learning [5] helps to combine the predictions from several architectures and give a better prediction based on them. We ensembled the the predictions using weighted average strategy. We assigned weights 0.3 to U-Net, 0.4 to U-Net++ 0.3 to ResNet-50 U-Net++ respectively. We tried various combination of weights but that did not alter the dice score significantly.

3 Results and Analysis

3.1 Experimental Setup

We primarily used google colaboratory and jupyter notebooks for writing our models. Pytorch library was used extensively for development and implementation of the architectures. We had access to NVIDIA GeForce RTX 3090 GPU for running our models on the dataset and computing results.

We are grateful to the Image Processing Lab at ECSU, ISI Kolkata for providing us the required computational facilities.

3.2 Performance Metrics

3.2.1 Loss Function

The loss function that we used is Binary Cross Entropy With Logits. BCE is used in binary classification tasks. These are tasks that answer a question with only two choices.

$$L = f(L_i, 1im), \tag{3}$$

$$L_i = -((x)_i log(y_i) + (1 - x_i) log(1 - y_i)) \tag{4}$$

Here, x_i and y_i indicate the i^{th}prediction and mask (5)

respectively and the functions are evaluated pixel-wise, m is the batch-size, the function f may be any kind of operation on the losses for individual images, in our case, we simply take the average.

3.2.2 Dice Coefficient/F-1 Score

The Dice score(Bertels et al.,2019)[2] is used to gauge model performance, ranging from 0 to 1. Simply put, the Dice Coefficient is 2 times Area of Overlap divided by the total number of pixels in both images. Mathematically,

$$DSC = \frac{2|x \cap y|}{(|x| + |y|)}. \tag{6}$$

Alternatively we can say,

$$DSC = \frac{2TP}{(2TP + FP + FN)} \tag{7}$$

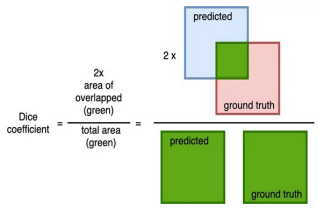

Figure 8: Dice Score

3.2.3 Pixel Accuracy

Pixel accuracy is the percent of pixels in your image that are classified correctly. However **high pixel accuracy does not always imply superior segmentation ability** as this metric has a major shortcoming when it comes to dealing with class imbalance. In our dataset most of the pixels are black which is not handled well by pixel accuracy measure. So if our model classifies all pixels as black, 95% of pixels are classified accurately while the other 5% are not. As a result, although the accuracy is a whopping 95%, the model is returning a completely useless prediction with no segmentation whatsoever.

3.3 Hyperparameter Tuning

Threshold - We passed the predictions of our model through a sigmoid function and used a range of thresholds to figure out which gives the best performance for our model. After careful testing we are using a threshold of 0.018 to classify the pixels as 1(white) or 0(black).

Epoch - After running our model several times we found that the performance flatlining occured at around 200 epochs. Hence we plotted our performance metrics on each model for 200 epochs.

Gaussian Blur - We applied Gaussian Blur on the predicted masks for sharpening. We tried various standard deviations for getting the best results.

3.4 Comparative Performance

We trained three models, one for each of the architectures mentioned previously, we trained each model with 2 layers for 200 epochs, with batches of size 16 each.

The ensemble process that involves combining the prediction of the three models through a weighted average function does generate predictions with a higher dice score taking the best of the three models while deciding on the weights according to dice score of the models. We gave 0.3, 0.4 and 0.3 weights to U-Net, U-Net++ and ResNet-50 U-Net++ respectively.

Performance Table

Model	Accuracy	Dice Score
U-Net	98.32	0.851
U-Net++	98.35	0.848
ResNet-50 U-Net++	98.47	0.839
Ensemble	98.69	0.858
After Post-Processing	98.88	0.869

We have taken the best results of dice score for every model by using checkpoints. We recorded the accuracy and dice score for 200 epochs. For the loss curves we considered first 50 batches of a single epoch as the loss remained near about constant after that point.
The dice score of U-Net architecture is higher than U-Net++ and ResNet-50 U-Net++. But on observing the masks predicted by U-Net, we observed that they were bloated and blurry in many cases when compared to the original masks. On the other hand, the masks predicted by ResNet-50 U-Net++ was more well defined and the boundaries were clearer. After we ensembled the best results that we got the masks which had both higher accuracy and dice score and were more well defined than the predictions of any of the models.

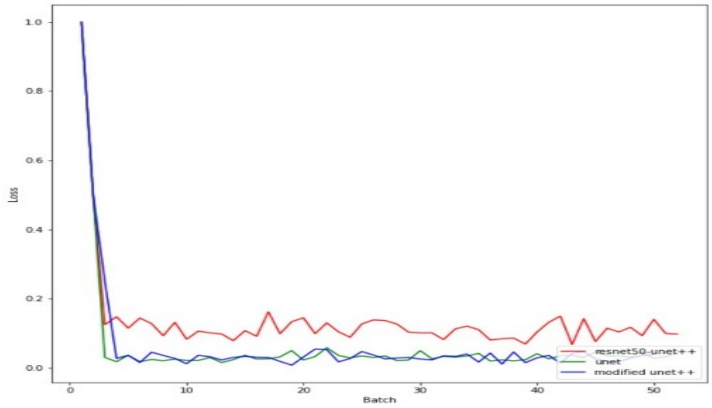

Figure 9: Loss Curves

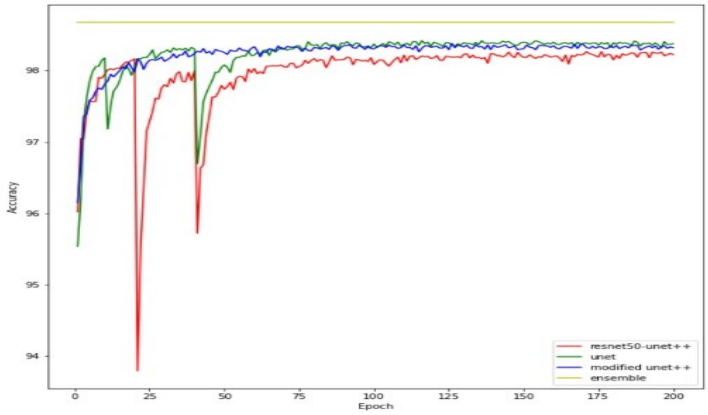

Figure 10: Accuracy graphs

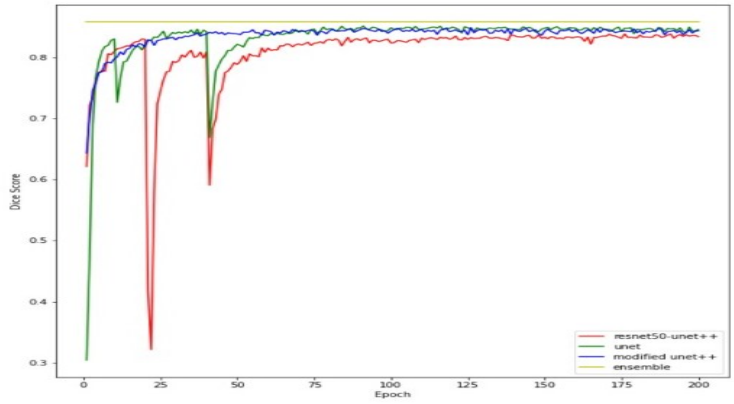

Figure 11: Dice Coefficient graphs

Figure 12: Precision Recall Curve

3.5 Image of Masks

We had taken batch size as 16 and below are the images of the actual and predicted mask as obtained by our process.

Figure 13: Actual Masks

Figure 14: Masks predicted by U-Net

Figure 15: Masks predicted by Modified U-Net++

Figure 16: Masks predicted by ResNet-50 U-Net++

Figure 17: Masks predicted using Ensemble

3.6 Post Processing

After obtaining the ensembled masks we applied some post processing on the masks for sharpening the boundary regions. This helped in a cleaner segmentation of Hippocampus from the CT slices. Due to the process of ensembling we encounter some random noise in the final

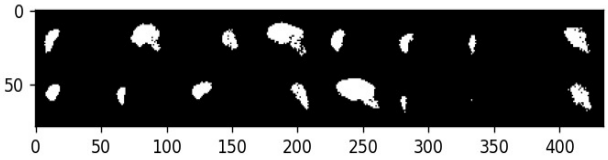

Figure 18: Ensembled example found to be prone to noise

segmented images despite the high dice score of 86. These random noises were removed by a use of morphological operations. Dilation and erosion are tested on all hundred and sixteen batches of predicted images and subjectively evaluated based on perceived quality of the enhanced images alone and not any other metrics like dice score or accuracy. Results of the experiments showed that noise can be effectively removed from binary images using combinations of erode-dilate operations. Also, the binary images are significantly enhanced using combinations of majority-close operations. For our approach involving this particular dataset, we have considered only a single pair of erosion followed by a dilation operation on the predicted images. The small size of the cropped 2D images prevents from using any more operations as the experiments conducted with more operations show significant distortion of the masks.

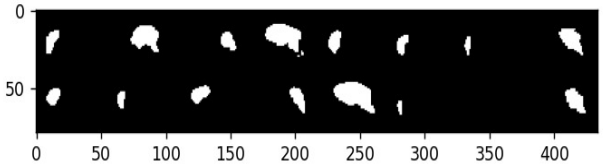

Figure 19: Noise removed after performing dilation-erosion

To further enhance the images, we have also considered the shortcomings of the individual results of the predictions of the three models respectively. As we have observed, predicted images of unet based architecture didn't have good edges when compared to the actual masks. The edges seemed somewhat 'bloated' compared to what we originally somewhat sought out to achieve. This very trait was passed onto the ensemble result as well. To counter this very problem, we considered a modified erosion operation aimed specifically for the edges such that it will leave the rest of the image unaffected so as to not distort the image and get good results. To achieve this we first perform an edge detection operation on the predicted images. We pass this matrix which contains only the information about the edges of the segmentation through a Guassian Blur function with a kernel size of 3x3 and with sigma value as 1. This leads to the smoothening out of the edges. Finally after multiplying the blurred images with a chosen weight of 0.2 we subtract these edges from the original predicted images sharpening out the edges of the predicted image in

the process. Do note that this process is somewhat similar to traditional 'unsharp masking'.

$$processed\_image = image - 0.2 * GaussianBlur(edges, kernel = (3,3), \sigma = 1) \quad (8)$$

The resultant images we get are highly enhanced and

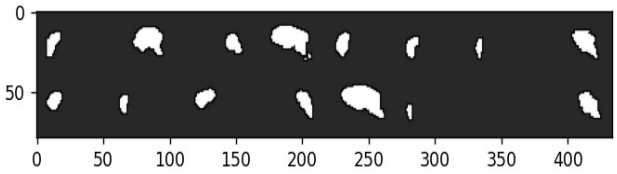

Figure 20: Result after edges sharpening

much of what we desire. Not only are there no longer any noise but also the edges are now much more prominent. But even so after the above operations the resultant image matrix is no longer a binary image. To counter this problem we performed a simple histogram thresholding. From our experiments we encountered that the ideal threshold value for getting visually pleasing resultant images to be 200.

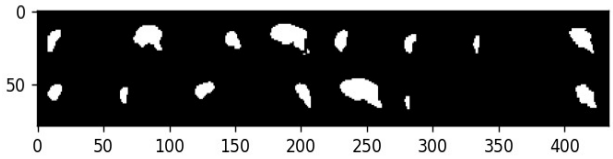

Figure 21: Final Result After Post Processing

4 Conclusion

After performing segmentation using all the three architectures and then comparing their final results we can safely conclude a few salient pros and cons of each of the architectures.

ResNet-50 backbone with U-Net++ gave good results with well defined edges and maintaining the general we would expect from the resultant mask. But the issue in these predictions was a lot of noise not seen in the other two architectures.

U-Net meanwhile gave good dice score and accuracy score and virtually no noise. But even so the results weren't desirable, since distortion in the shape of the segmentation results was obvious in our experiments. The masks were bloated in several cases.

Finally, in U-Net++ which we modified to perform better for this particular data set gave the best results out of the two. With negligible noise seen and also no distortion in the shape of the hippocampus, the predicted masks were better than the other two giving both high dice score and accuracy.
The modification which involved adding an up-convolution followed by a layer of convolution and again max pooling helps the architecture to solve the deformity that was seen in U-net. The nested nature of the U-Net++ architecture also helped to the cause without adding any unnecessary noise.

Finally when we performed ensembling of all the three results followed by morphological operations on the masks we got the best dice score and accuracy.

5 Further Work

1. We will train the proposed models in other segmentation based dataset based on biomedical segmentation, so that we can judge the architecture's effectiveness for other kinds of segmentation problems. Working on a variety of datasets will help us in tuning our hyperparameters further.

2. In this work all three architectures involve segmenting slices of 3D Mono-modal Magnetic Resonance Imaging (MRI) images in the preprocessing step for the the models to work on them. Thus we are essentially working on 2D images for segmentation. In future, we wish to incorporate some 3D segmentation architecture like V-Net on this hippocampus dataset as control experiment so that we can also compare their results with these 3 architectures.

3. We wish to apply some semi-supervised algorithm which will enable the model to segment images from unknown biomedical imaging datasets.

References

[1] Antonelli, M., Reinke, A., Bakas, S., Farahani, K., AnnetteKopp-Schneider, Landman, B.A., Litjens, G., Menze, B., Ronneberger, O., Summers, R.M., van Ginneken, B., Bilello, M., Bilic, P., Christ, P.F., Do, R.K.G., Gollub, M.J., Heckers, S.H., Huisman, H., Jarnagin, W.R., McHugo, M.K., Napel, S., Pernicka, J.S.G., Rhode, K., Tobon-Gomez, C., Vorontsov, E., Huisman, H., Meakin, J.A., Ourselin, S., Wiesenfarth, M., Arbelaez, P., Bae, B., Chen, S., Daza, L., Feng, J., He, B., Isensee, F., Ji, Y., Jia, F., Kim, N., Kim, I., Merhof, D., Pai, A., Park, B., Perslev, M., Rezaiifar, R., Rippel, O., Sarasua, I., Shen, W., Son, J., Wachinger, C., Wang, L., Wang, Y., Xia, Y., Xu, D., Xu, Z., Zheng, Y., Simpson, A.L., Maier-Hein, L., Cardoso, M.J.: The medical segmentation decathlon (2021), https://arxiv.org/abs/2106.05735

[2] Bertels, J., Eelbode, T., Berman, M., Vandermeulen, D., Maes, F., Bisschops, R., Blaschko, M.B.: Optimizing the dice score and jaccard index for medical image segmentation: Theory and practice. In: International conference on medical image computing and

computer-assisted intervention. pp. 92–100. Springer (2019)

[3] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)

[4] Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)

[5] Sagi, O., Rokach, L.: Ensemble learning: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8(4), e1249 (2018)

[6] Zhou, Z., Rahman Siddiquee, M.M., Tajbakhsh, N., Liang, J.: Unet++: A nested u-net architecture for medical image segmentation. In: Deep learning in medical image analysis and multimodal learning for clinical decision support, pp. 3–11. Springer (2018)