Quality Evaluation of Skull Stripped Brain MRI Images

Lakshmi Gayathri Rangaraju
lrangar@clemson.edu
Graduate Student
Clemson University

Shareef Shaik
shaik@clemson.edu
Graduate Student
Clemson University

Tejaswine Kantu
tkantu@clemson.edu
Graduate Student
Clemson University

1 Problem Statement and Goal

Skull stripped brain MRI images are usually shared in open access data repositories for collaboration and knowledge dissemination in order to advance neuro imaging research. Skull stripping is a process of removing the image voxels that represent the skull structure and facial features from raw 3D MRI images to ensure HIPAA compliance. Sometimes, the facial features are still in the processed images, and the other times, the voxels representing actual brain tissues are removed. To ensure the quality of the skull stripping process, researchers often spend a lot of time and effort to inspect the quality of the segmentation results. This paper suggests a deep neural network algorithm which makes the inspection of the skull stripped MRI images easy and reduces the manual intervention. The deep neural network algorithm which is explored in this paper is 3D Convolutional Neural Networks(CNN). CNNs makes it easy to process the images and evaluates the quality of skull stripped images with a good accuracy.

2 Introduction

Brain MRI imaging has been widely used by neurologists and neuroscientists in disease diagnosis and human brain study since the 1980s. There are quite a few different imaging modalities or sequences used in imaging the nervous system and producing different kinds of brain MRI images, such as T1-weighted (T1W) images, T2-weighted (T2W) images, Diffusion-weighted images (DWI), etc. To advance neuroimaging research, more and more researchers share neuroimaging data in open access data repositories for collaboration and knowledge dissemination. To ensure HIPAA compliance, researchers often use skull-stripping tools to remove the image voxels that represent the skull structure and facial features from the raw 3D MRI images before sharing data.

Skull stripping is a segmentation process to extract brain tissues from a raw 3D MRI image of a brain. This is one of the preliminary steps in neuroimaging research. Various skull-stripping tools and algorithms have been proposed to extract brain images [1].

However, using these algorithms and tools for brain image segmentation is a tedious task even for expert radiologists and the accuracy of results varies a lot from person to person. Sometimes, the facial features are still in the processed images, and the other times, the voxels representing actual brain tissues were removed. To ensure the quality of the skull stripping process, researchers often spend a lot of time and effort to inspect the quality of the segmentation results.

In order to reduce the manual effort and make the process of quality inspection easy, Convolution Neural Networks(CNN) can be adapted. In deep learning, CNN is a class of deep neural networks, which is most commonly applied to analyze visual imagery. In comparison to the traditional image classification methods, CNNs use relatively little pre-processing compared to other image classification algorithms. The network learns to optimize the filters (or kernels) through automated learning, whereas in traditional algorithms these filters are hand-engineered. This independence from prior knowledge and human intervention in feature extraction is a major advantage.

Among the CNN, the 3D convolution networks [2] have gained popularity especially for the analysis of medical images. This is because the MRI images are combination of 2D image slices resulting in a 3D scan. So, to extract the important local features of a pixel, it is necessary for us to slide through the volume dimension where the 3d-convolution comes into the picture. With the recent advancements in neural network architectures, data augmentation techniques and high-end GPUs, it is becoming possible to analyze the volumetric medical data using 3D deep learning.

In order to leverage the advantages of 3D convolution process, this paper designed a 3D convolution network to detect if the given brain MRI images have any recognizable facial features and if the brain features are lost.

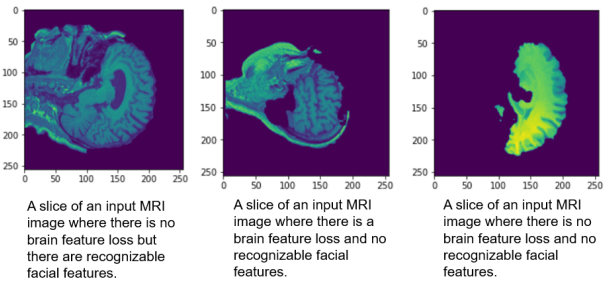


Figure 1: Data set visualization

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
├─Conv3d: 1-1                            [-1, 5, 31, 127, 127]     140
├─ReLU: 1-2                              [-1, 5, 31, 127, 127]     --
├─MaxPool3d: 1-3                         [-1, 5, 15, 63, 63]       --
├─BatchNorm3d: 1-4                       [-1, 5, 15, 63, 63]       10
├─Conv3d: 1-5                            [-1, 3, 13, 61, 61]       408
├─ReLU: 1-6                              [-1, 3, 13, 61, 61]       --
├─MaxPool3d: 1-7                         [-1, 3, 6, 30, 30]        --
├─BatchNorm3d: 1-8                       [-1, 3, 6, 30, 30]        6
├─Conv3d: 1-9                            [-1, 2, 4, 28, 28]        164
├─ReLU: 1-10                             [-1, 2, 4, 28, 28]        --
├─MaxPool3d: 1-11                        [-1, 2, 1, 13, 13]        --
├─BatchNorm3d: 1-12                      [-1, 2, 1, 13, 13]        4
├─Flatten: 1-13                          [-1, 338]                 --
├─Linear: 1-14                           [-1, 9]                   3,051
├─Linear: 1-15                           [-1, 2]                   20
├─Sigmoid: 1-16                          [-1, 2]                   --
==========================================================================================
Total params: 3,803
Trainable params: 3,803
Non-trainable params: 0
Total mult-adds (M): 87.60
```

Figure 2: Model Parameters

3 Dataset

The images in the data set where of the format ".NIFTI" [3]. The Neuroimaging Informatics Technology Initiative (NIFTI) is an open file format commonly used to store brain imaging data obtained using Magnetic Resonance Imaging methods. Each image is of height - 256, width - 256. The depth of the images varied from one image to another. A sample of different category data set images are shown in the figure-1.

The data set consists of two labels - Label-1) Recognizable-Facial-Feature and Label-2) Brain-Feature-Loss. As these two labels have to be predicted simultaneously by our model, combination of these two labels were taken and the entire data set is categorised into four groups. The four groups are y_y, y_n, n_y, n_n. Here y_y means "Recognizable-Facial-Feature as yes" and "Brain-Feature-Loss as yes". In the same way all other combinations of the two labels are considered. After categorising the data set, the number of images in each category are - Y_N:713, N_Y:1332 and N_N:15.

After the data set is categorised, 60% of each category were considered into training data set and remaining 40% of each category was added to validation(20%) and testing(20%) data set. After splitting the entire data set into train, validation and testing data sets, the final training data set size was 1237, validation data set size was 411 and testing data set size was 412.

4 Pre-Processing the data set

Before feeding the input images to the network, the images are pre-processed such that the network learns swiftly the important features of the images. The pre-processing steps which are adapted are 1) Min-Max normalization, 2) Resizing of the image, 3) Expanding the image dimensions.

Min-Max normalization: The reason for applying min-max normalization [4] to the input images is all the pixel values will be in the range of 0-1. This normalization process is important because the larger value pixel can disrupt or slow down the learning process. In order to bring all the pixel intensities of the image to the range of 0-1, the pixel value should be subtracted by the minimum pixel value and the result is divided by (maxPixel value - minPixel value). The formula for the min-max norm is given in the figure-2.

$$\frac{value - min}{max - min} \tag{1}$$

The advantage of using Min-Max normalization is it preserves the relative order and distance of the data points.

Resizing the image: The images in the data set are of shape height - 256, width - 256, however, they change in depth dimension. To maintain consistency among all the input images and due to computational limitations, the images in the data set are resized into (256*256*64). Only the first 64 channels are considered for all the images. After resizing the input images, the image are reshaped to channel first dimensions. This is because the convolution network accepts images with channel first dimensions. So the final image shape is equal to (64*256*256).

Expanding the image dimensions: As the network uses 3D convolution layer, the image dimensions need to be expanded to add another dimension to the image. After expanding the image dimensions, the new shape of the image was (1*64*256*256).

5 Network

The network built for this model, consists of three convolution 3D layers and three fully connected layers. The input to the network is the pre-processed images of shape (1*64*256*256), and the output of the network is the two labels. The two labels here determine if the input image has recognizable facial features and if there are any brain feature loss. The model parameters can be seen in the figure 2.

The network architecture is given in the figure 3. The activation function used for our model is "Relu", and the pooling that is performed here is "MaxPooling". The number of convolution 3D layers and number of fully connected layers in the network are decided after performing trial and error with different number of layers.

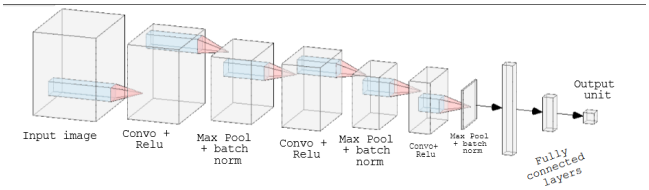


Figure 3: Network Architecture

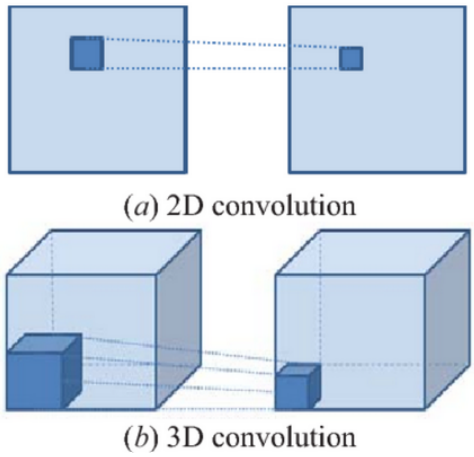


Figure 4: Comparison of 2D and 3D convolution

The significance of each layer in our model and why they are chosen for our model is discussed in the following sub sections.

5.1 Convolution-3D

Convolution-3D is a unique CNN layer used in deep learning for analyzing three-dimensional data, such as videos or 3D medical scans. Convolution-3D produces an output volume of filtered features by applying trainable filters to a 3D input volume. Typically, the filters are tiny 3D cubes that move over the input volume while computing the dot products between their weights and the values in the input volume. This procedure is done for each filter to create a set of filtered feature maps.

Convolution-3D layers automatically extract features from 3D data that apply to applications like segmentation, object recognition, and action recognition. They are frequently employed in deep learning since they have proven highly useful in many applications.

Why 3d conv-networks for our project? Due to its consideration of the data's spatial and temporal dimensions, Conv3D performs better [5] than Conv2D when analyzing 3D medical images. Each pixel or voxel in medical imaging carries data about a particular location in 3D space and is dynamic. Conv2D ignores the depth and temporal dimensions as it analyses images on a 2D grid. This implies that when processing 3D medical images with Conv2D, significant spatial and temporal information may be lost. Contrarily, Conv3D is built to handle 3D data and considers the data's spatial and temporal dimensions. It is helpful in tracking irregularities over time since it may capture information unique to particular areas of the 3D volume and represent temporal dependencies.

Conv3D can therefore collect more pertinent features and better maintain the spatial and temporal information in the data than Conv2D, leading to greater accuracy and performance in 3D medical picture analysis.

5.2 Batch Normalization

Batch normalization [6] is a normalization technique [7] between the layers of a neural network. As the name suggests, the normalization process is performed across a batch of input data. This is sometimes called the batch statistics. Specifically, batch normalization normalizes the output of a previous layer by subtracting the batch mean and dividing by the batch standard deviation. This is much similar to feature scaling which is done to speed up the learning process and converge to a solution.

Advantages of batch normalization are it standardizes the distribution of layer inputs to combat the internal covariance shift. It controls the amount by which the hidden units shift. Batch normalization facilitates
1) Faster training.
2) Use of higher learning rate.
3) Easier Parameter initialization.
4) Makes activation functions viable by regulating the inputs to them.
5) Better results overall.
6) It adds noise which reduces over fitting with a regularization effect. Thus make sure to use less dropout when you apply batch normalization as dropout itself adds noise.

In order to leverage the advantages of the batch normalization, and to make the training process faster and smoother, batch normalization layer is introduced after every convolution layer.

5.3 Loss function

The loss function used in our network is sigmoid cross entropy loss. Sigmoid cross entropy loss is nothing but sigmoid activation plus cross entropy loss.

Sigmoid activation function is used for the last layer of our network because, the labels in our problem statement are independent of each other. The formula for the Sigmoid activation function is:

$$\sigma(Z_j) = \frac{e^{Z_j}}{1+e^{Z_j}} \tag{2}$$

$$L_{\mathrm{CE}} = -\sum_{i=1}^{n} t_i \log(p_i), \quad \text{for n classes,}$$

where t_i is the truth label and p_i is the Softmax probability for the i^{th} class.

Figure 5: Cross Entropy loss formula

Zj in the above formula indicates the raw output of a single output unit in the network. Use of sigmoid activation function aids the network to learn to predict the two labels accurately.

The cross entropy loss can be defined as the summation of product of class probabilities predicted by the model and the ground truth values of each class. The formula of cross entropy loss is shown in figure 4.

The cross entropy loss calculates the loss of each class separately, which helps network to penalize the weights corresponding to each class. On the whole by using sigmoid cross entropy loss, the network learns to predict both the class labels with good accuracy.

6 Software used

The language in which the model is developed is "Python" using "Pytorch" as the deep learning framework. In order to read the input images, "nibabel" library was used which is available in python packages. Additional python package which is used to print the metrics in a tabular format is "torchsummary".

7 Hyper parameters

Hyperparameters play a crucial role in determining the performance of a model. These parameters are set prior to the training process and can impact the learning rate and other regulations during training. The hyper parameters which are set for our model are 1) Learning rate 2) Batch size 3) Number of epochs 4) Optimizer 5) Threshold value to determine the class label.

The learning rate parameter [8] controls the speed at which the model learns and determines the step size at each iteration while moving towards the minimum of a loss function. In our model, learning rate is set to 1e-4.

Another important hyperparameter is the batch size [9], which refers to the number of samples that are passed to the network at once. In our model, the batch size is set to 50. The number of epochs, which represents the total number of passes through the entire training data set, is another hyperparameter that was set to 35. The reason for choosing only 35 number of epochs is due to computational limitation.

Optimizers are algorithms that modify the attributes of the neural network, such as learning rate and weights, in order to

```
-----Testing accuracy and avg loss are-----
|---------------------------------------|---------|----------|
| Accuracy                              |    loss |     time |
|---------------------------------------|---------|----------|
| (88.83495145631069, 48.54368932038835) | 0.32848 | 66.94941 |
```

Figure 6: Test accuracy for 3D convolution

reduce the overall loss and improve accuracy. In our model, the ADAM optimizer [10] (Adaptive Moment Estimation) was used, which is a combination of two gradient descent methods, momentum and RMSP. Initially SGD ws used as the optimizer but later when changed to ADAM there is a significant increase in the accuracy of our model.

Finally, the threshold parameter [11] is used to determine the class label for a given input. To explain in detail, the two outputs units of our model, give the probability with which a particular input has "Recognizable facial features" and "Brain feature loss". Now the probabilities are to be converted to 0 or 1 in order to calculate the loss and accuracy for our model. So a threshold parameter is considered - 0.4 in our case, and the output probability which exceeds the threshold is considered as output 1 else 0.

Hence, hyper parameter tuning [12] is very important in order to train neural networks accurately. By carefully selecting these hyperparameters, our CNN model was optimized for better performance.

8 Metrics

Our model is evaluated using the accuracy metric. For the 3D convolution neural network, the training accuracy metric for label 1 and label 2 is shown in figure 10. Figure 11 shows the validation accuracy for label 1 and label 2. The testing accuracy is almost same as validation which is shown in the figure 5.

The change of training and validation accuracy over epochs is plotted for both the labels separately. For "Recognizable facial feature" label, the plot is shown in the figure 6. It can observe that the model initially gave an accuracy of 64%, later after training for 10 epochs the accuracy increased to 98%. Even for the label - "Brain Feature loss" in figure 7, it can be observed that the training accuracy changes from 64% to 94%. As batch normalization is used for our 3D convolution neural network, the accuracy increased in the early stages of our training.

The validation accuracy is calculated for every 10^{th} epoch. The trend of validation accuracy observed for the label 1 - "Recognizable Facial Features" is in the initial epoch it was around 34%. And after training, it increased to 94%. Where as for the label 2 - "Brain feature loss", the validation accuracy was 64% in the initial epoch and after training it decreased to 48%.

From the validation accuracy trend of the label 2, it can

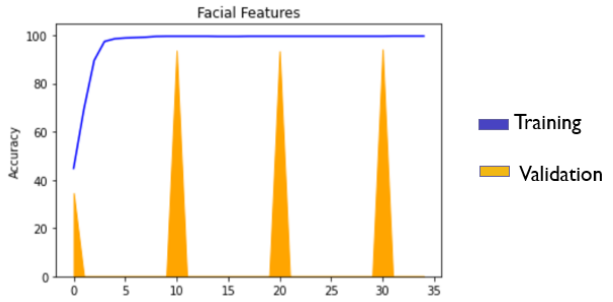


Figure 7: Plot of accuracy vs epoch for the label "brain features loss"

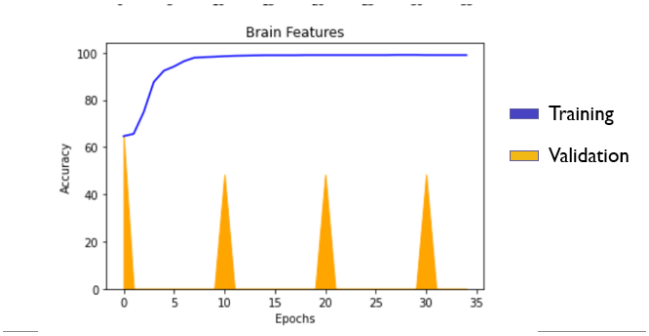


Figure 8: Plot of accuracy vs epoch for the label "recognizable facial features"

be noticed that our model didn't learn the label 2 features properly. This is one of the drawback of our model. Our model could only learn one label features accurately. Hence, there is a scope for future improvement of our model to make the model learn the second label features.

9 Experiments performed

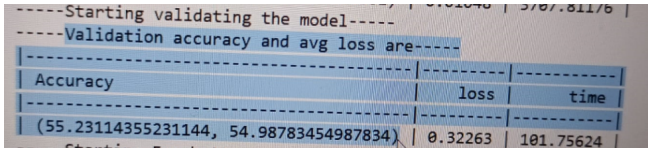
9.1 Conv2D to Conv3D

Initially the 2D convolutional network was built. Even though 2D convolutional neural network gave good accuracy on training data set, the validation and test set accuracy were inadequate. The training accuracy which was observed was 99% for label 1 and 64% for label 2 as shown in figure 8. The validation and test accuracies observed were around 54% for both the classes as shown in the figure 9.

This is obvious that 2D convolution just slides along height and width of a slice of the image. So it ignore the local pixels

Accuracy	loss	time
(99.59579628132579, 68.06790622473726)	0.27304	2078.70371

Figure 9: Training accuracy using 2D convolution



-----Starting validating the model-----
-----Validation accuracy and avg loss are-----

Accuracy	loss	time
(55.23114355231144, 54.98783454987834)	0.32263	101.75624

Figure 10: Validation accuracy using 2D convolution

Accuracy	loss	time
(98.86822958771221, 94.17946645109136)	0.12596	1182.72641

Figure 11: Training accuracy using 3D convolution

in the volume dimension. In order to consider all the local pixels of a particular pixel of brain MRI image, 3D convolutional is finalized.

When the network is changed from 2D convolution to 3D convolution, the training accuracy boosted along with the validation and test accuracy. Training accuracy observed for 3D convolutional neural network was 98% for label 1 and 94% for label 2 as shown in the figure 10. Moreover, the validation and testing accuracies increased to 94% for label 1 and 48% for label 2 as shown in figure 11 and figure 5 respectively.

9.2 Adding batch normalization layer

Initially the 3D model is constructed without using the batch normalization layer. Without the batch normalization layer, the training accuracy fluctuated around 34%. Even though other hyper parameters like optimizer, learning rate are tweaked, the training accuracy didn't show much progress.

After the addition of the batch normalization layer, the training accuracy boosted from 34% to 64% in early stages of training and ended up at 99% after the training is completed. Hence, batch normalization helped for smooth and fast training of our model without depending on how parameters are initialized.

10 Future Enhancements

It can be observed that the validation and test accuracy for the label-2 is only 48%. In order to increase the accuracy of label 2, two different models can be built. Later each model can be trained to predict only one label. The separate models for predicting each label works because the model can focus

-----Validation accuracy and avg loss are-----

Accuracy	loss	time
(94.16058394160584, 48.41849148418492)	0.15857	55.28105

Figure 12: Validation accuracy using 3D convolution

on learning a single label features rather than learning two independent label features.

Second improvement which can be tried is augmenting the data set. As the number of examples in each category for two labels is imbalanced, the labels can be separated and data augmentation [13] can be applied. Data augmentation helps to increase and diversify the data set. This diversified data helps the two models to learn features without getting biased to a single category of label.

References

[1] Sandabad Sara, Bara Samir, Hammouch Ahmed, and Cherradi Bouchaib, "A robust comparative study of five brain extraction algorithms (bet; bse; mcstrip; spm2; tmbe)," in *2014 Second World Conference on Complex Systems (WCCS)*, 2014, pp. 632–636.

[2] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri, "Learning spatiotemporal features with 3d convolutional networks," 2015.

[3] Xiangrui Li, Paul S. Morgan, John Ashburner, Jolinda Smith, and Christopher Rorden, "The first step for neuroimaging data analysis: Dicom to nifti conversion," *Journal of Neuroscience Methods*, vol. 264, pp. 47–56, 2016.

[4] Nazanin Vafaei, Rita Ribeiro, and Luis Camarinha-Matos, "Importance of data normalization in decision making: case study with topsis method," 05 2015.

[5] Hayit Greenspan, Bram van Ginneken, and Ronald M. Summers, "Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1153–1159, 2016.

[6] Vignesh Thakkar, Suman Tewary, and Chandan Chakraborty, "Batch normalization in convolutional neural networks — a comparative study with cifar-10 data," in *2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT)*, 2018, pp. 1–5.

[7] J. Sola and J. Sevilla, "Importance of input data normalization for the application of neural networks to complex industrial problems," *IEEE Transactions on Nuclear Science*, vol. 44, no. 3, pp. 1464–1468, 1997.

[8] Nikhil Iyer, V Thejas, Nipun Kwatra, Ramachandran Ramjee, and Muthian Sivathanu, "Lrtuner: A learning rate tuner for deep neural networks," 2021.

[9] Leslie N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay," *CoRR*, vol. abs/1803.09820, 2018.

[10] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," 2017.

[11] Carmen Esposito, Gregory A. Landrum, Nadine Schneider, Nikolaus Stiefl, and Sereina Riniker, "Ghost: Adjusting the decision threshold to handle imbalanced data in machine learning," *Journal of Chemical Information and Modeling*, vol. 61, no. 6, pp. 2623–2640, 2021, PMID: 34100609.

[12] Wei-Chang Yeh, Yi-Ping Lin, Yun-Chia Liang, and Chyh-Ming Lai, "Convolution neural network hyperparameter optimization using simplified swarm optimization," 2021.

[13] Agnieszka Mikołajczyk and Michał Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, 2018, pp. 117–122.