Comp 775 Final Project: Machine Learning to Segment a Corpus Callosum PDM

Samuel D. George
Department of Computer Science
Department of Pharmacology
The University of North Carolina
Chapel Hill, NC 27599-2200
sdgeorge@cs.unc.edu

Abstract

Machine Learning can be a powerful tool, particularly for doing medical image analysis. For my project, I use machine learning to segment the Corpus Callosum from a set of given brain images by predicting the points on a point distribution model outlining this structure. I was able to successfully extract, in most cases, four landmark points on the Corpus Callosum. I consider this to be a good result, given that I limited myself to just 32 unique training images. In this project I utilize many cutting edge techniques in machine learning, and show the potential of these models to be used for medical image analysis.

1 Introduction

Developments in machine learning over the past decade have revolutionized many areas of data analysis. This revolution has probably had the largest impact on the field of image analysis. The ImageNet competition, for example, is a contest to build a neural network to recognize 1000 different classes of images. A task such as this would be impossible using traditional image analysis methods. In 2015 these networks advanced to a level that is considered to be superhuman(1). Despite the power of these networks, however, medical images still present several challenges that have prohibited machine learning from impacting the field. The first major challenge is the need for data. In order to train ImageNet, for example, you need to have over 1000 images of every class you wish to identify. This means that you may need upwards of 1 million images to train an accurate classifier. Collecting this amount of medical image data is often intractable, either due to cost or difficulty in accessing the data. Another critical challenge is the unpredictable nature of neural networks. Unlike traditional statistical methods that have a defined set of hyper-parameters and a well understood consequence for these parameters, neural networks are much more of a 'black box'. When neural networks perform well, they outperform even the best statistical methods. When neural networks fail, it is not entirely clear why and how to fix them. Despite this, advances in neural network design and techniques have allowed machine learning to begin to have an impact in the field of medical image analysis. For my project I intend to leverage several key neural network concepts in order to achieve good results with a relatively limited set of training data.

2 Project Goal

The initial goal of my project was to build a neural network that predicts the point distribution model (PDM) for the Corpus Callosum (CC). To simplify the problem at the start, I sought to only predict 4 critical points on the PDM. The head, back, and two key points in the middle of the CC. Had my neural network performed adequately, I would have expanded to more points. As we will see in the results, just predicting these four landmark points is very challenging. Therefore, I readjusted my goal to simply predict these four points. Figure 1 illustrates my projects goal.

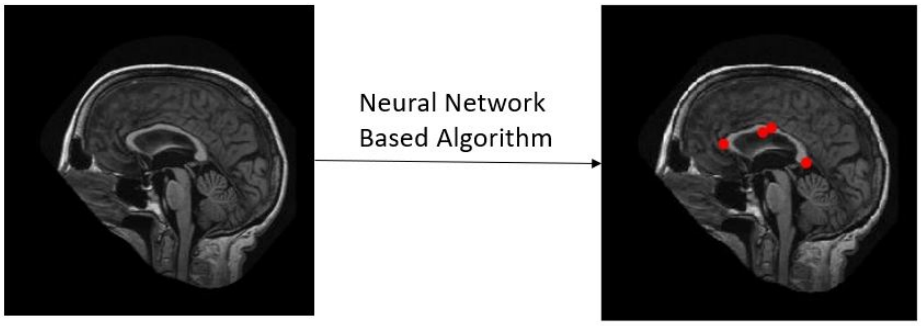

Figure 1: Project Goal

3 Pre-Project Research

3.1 Object Detection as a model for finding a PDM

Once machine learning reached superhuman levels in object classification, the next obvious challenge is to identify where in the image these objects occur. This class of problems is called object detection. Object detection is usually done by first training a classifier, and then altering its output to be a bounded box instead of a SoftMax value indicating the class. This problem is so widespread that specific data sets have been curated just to train this type of network (2). Most single object detection methods work by predicting the set of (x,y,height,width) for the box around the object. A wide variety of methods exist for doing this task, but most incorporate a pre-trained network for classification followed by a dense neural network (DNN) to predict the bounded box (3). L2-loss between a ground truth box and the predicted box is then used to calculate loss and the DNN weight's are altered by backpropagation. To address multiple objects, most networks then employ some kind of scanning algorithm to find multiple objects or detect regions of interest. I use a modified version of these concepts to design my neural network.

3.2 Multi-Task Learning

A critical aspect of getting my network to perform well was to implement a Multi-Task Learning (MTL) style neural network. This type of network seeks to treat a complex task as a set of independent, but related simpler tasks. An overview of the various ways this is done led me to implement a 'Hard-Sharing' model. 'Hard-Sharing' means that the independent tasks contain a set of shared layers, followed by a set of layers that are independent for each task. This is one of the primary models used in MTL networks (4). The use of MTL has been shown to greatly enhance the performance of neural networks, particularly when a task is complex or the amount of training data is sparse (5).

4 Methods

Predicting the four PDM landmark points required two major stages. The first stage was to generate training data from the 32 original images and PDMs given to us for assignment 2. The second stage was to design a neural network and train it to predict PDM points.

4.1 Building a Training Data Set: Sample Creation

Neural network training often requires thousands of examples in order to train. Given I only had 32 images and PDMs, my first objective was to generate additional training data. I did this by taking each image with all of its PDM and selecting the four landmark points for that image. I then took that image-PDM combination and rotated it by 5 degrees and saved that as an additional sample. This gave me 72 image-PDM combinations for each of the 32 original samples. This gives a total of 2304 samples that served as my data set for the project. Figure 2 shows an example of the samples generated.

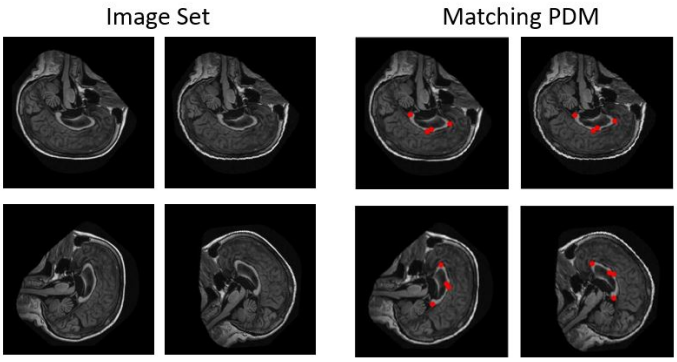

Figure 2: Sample Set of Training Data

4.1.1 Building a Training Data Set: Data Set Creation

Using the Image-PDM samples I created, I made a training and validation data set. The training data set contained 24 (75%) of the original images and their corresponding rotations, while the validation set contained the remaining 8 (25%) original images and their rotations. I divided the data this way so that my training and validation data sets did not contain any of the same original brain images. This is to prevent the neural network from simply memorizing a solution for each brain image and then learning to rotate the PDM. I then translated these two data sets into TensorFlow Record files. This condenses them into byte format and allows them to be used with the TensorFlow Dataset API. Code to create these records from an array of images and PDMs is shown in the Jupyter Notebook 'tf_record_maker.ibynb'. I have included these records as well as a zip of all my original data in the 'data' folder on the project github. With all my data collected, I was now ready to design my neural network.

4.2 Network Design: Implementing a Pre-Trained Network

To combat the issue of having a relatively small number of samples, I used a network that had been pre-trained on ImageNet. The common understanding of neural networks is that the last layer after all the convolutional layers (called the overlayer) of a CNN serves as an extracted set of features from the image. A network trained on ImageNet has a rich overlayer that should serve as a great starting point for my neural network. Figure 3 shows a diagram of the conventional wisdom for how a classification neural network works. I tried using the overlayer from four different networks trained on ImageNet: Inception V4, Resnet V2 50, ResNet V2 152, and PNASNet-5_Large_331. The Progressive Neural Architecture Search (PNAS) based network performed the best (6).

4.2.1 Network Design: Original Network

Using object detection as a guide, the original network I designed took the overlayer from a pre-trained network and ran the output through several DNN layers to finally produce an estimate for the eight values (four points) that should be present in the PDM. These PDM estimates are then compared to the true PDM and L2-loss is used to adjust the weights of the dense layers at each training step. Note, I froze training for the convolutional layers beneath the overlayer and these were not adjusted in training. This was to prevent over-fitting and to keep the overlayer feature rich. Figure 4 shows the diagram of my original network.

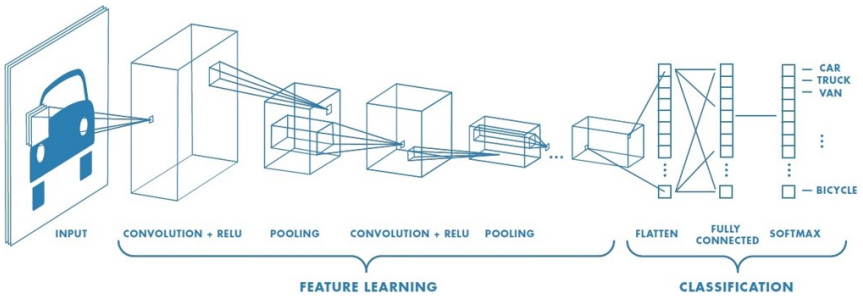

Figure 3: Typical Classification Network Design (*https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html)

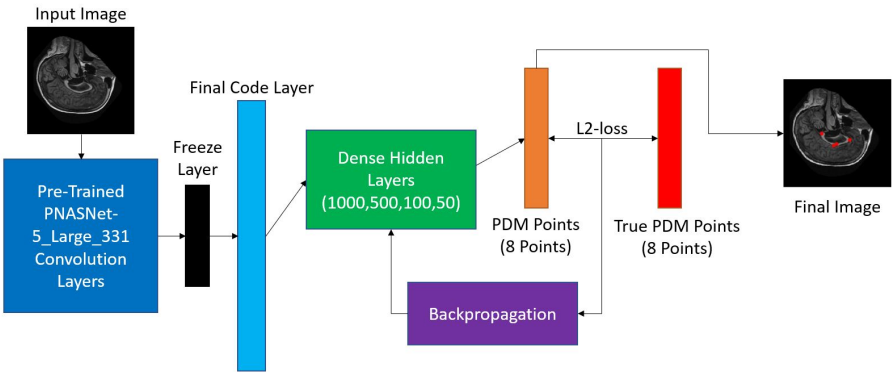

Figure 4: An Object Detection Like Network for estimating PDMs

However, the performance of this network was fairly poor and did not yield good PDMs for my images. It was at this point that I implemented MTL in my network.

4.2.2 Network Design: Multi-Task Learning Network

Despite tuning several different hyper-parameters, I was unable to get good performance with the original network I designed. At this point I adopted an MTL based network. Here I use the same overlayer and run it through several dense layers, but each point in the PDM is broken off and calculated with its own set of dense layers. L2-loss is calculated for these points separately and each point has its own optimizer during training. The theory behind this is that the system is now able to get a much more clear view of locality for each point and which direction to adjust based on the loss. Figure 5 shows a diagram of this revised network architecture.

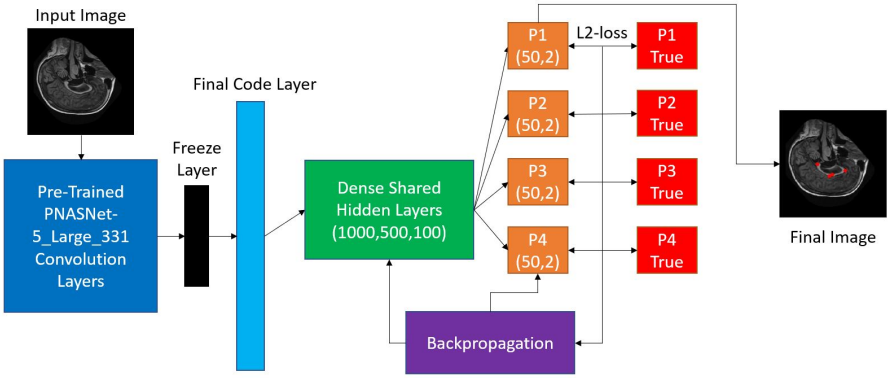

Figure 5: MTL based network allowing independent point PDM estimation

This network gave me the best overall performance for estimating the landmark PDM points. All my results use this architecture.

4.2.3 Network Design: Hyper-Parameter Tuning

Building a neural network can involve tuning many different hyper-parameters. Most of these parameters do not have a clear consequence related to the problem you are trying to solve. Table 1 shows a list of the hyper-parameters that I used for my network. These were acquired mostly through trial and error.

Table 1: Neural Network Hyper-Parameters

Parameter	Value
Convolutional Layers	PNASNet-5_Large_331
Dropout Rate	.01 (Basically None)
Batch Normalization	Yes (All but last layer)
Loss Function	L2-loss
Optimizer	Adam Optimizer (Learning Rate = 1)
Activation	Relu
Parameters Trained	2.55 Million
Total Parameters	91.77 Million

5 Results

Here I show the results of my experiments with the final neural network that I designed. Performance and figures are all using the validation data set.

5.1 Training

Looking at Figure 6 we see after about 150 epochs, the network has settled at an optimum solution. The high number of epochs speaks to the difficulty of the task that we are trying to accomplish. It is also interesting to observe that the losses for points 1 and 3 are generally smaller than the losses for points 2 and 4. Points 1 and 3 are the head and tail points of the CC, while 2 and 4 are the center points. It makes sense that the thin piece in the middle would be harder to locate accurately given image features trained on ImageNet.

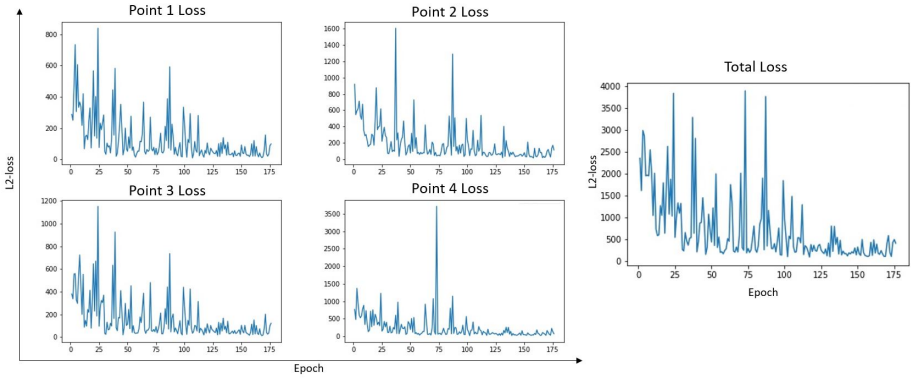

Figure 6: The Neural Network Learning the PDM points

5.2 Visual Results

One of the most challenging aspects of judging a PDM is that the result is mostly visual. While our loss functions indicate the network has indeed learned, it is difficult to judge how well my network performed without looking at a set of images. Figure 7 shows a different orientation of images 27,28,29,30,31, and 32 along with the guess PDM and true PDM.

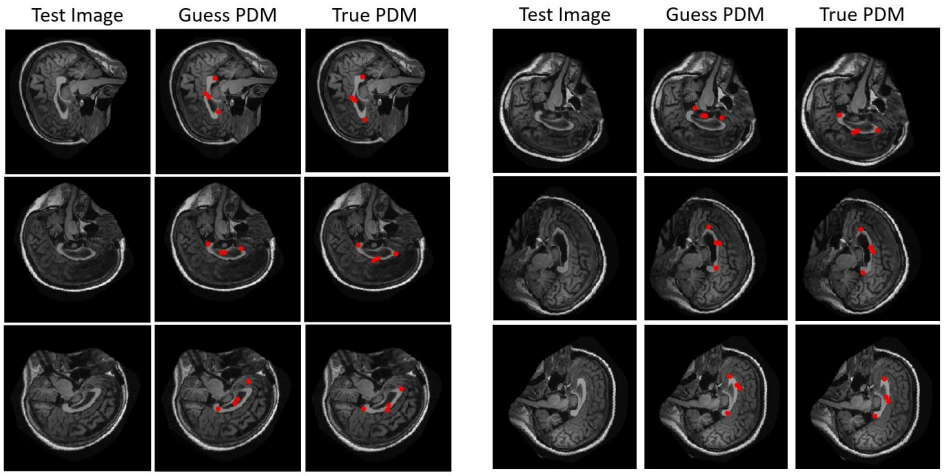

Figure 7: Visual of our estimated PDM points

Looking at the visual results, we see that almost all of the images find a correct head and tail point on the CC. Most of the mistakes made by my neural network are in estimating the two center points. This makes sense given the small visual space of this feature.

5.3 Statistical Results

Performing a PDM estimate on all 576 images we see that the network is doing a pretty good job of fitting our PDMs overall. Table 2 summarizes the L2-losses for our validation set.

Table 2: Validation Set Losses

Statistic	Value
Validation Set Average L2-loss	427 (Each pixel is off by about 7 pixels)
Validation Set Median L2-loss	391 (Each pixel is off by about 6 pixels)

Figure 8 shows the distribution of losses for the samples. Here we see that most of the samples have a very acceptable loss, with only a few images having really bad fits.

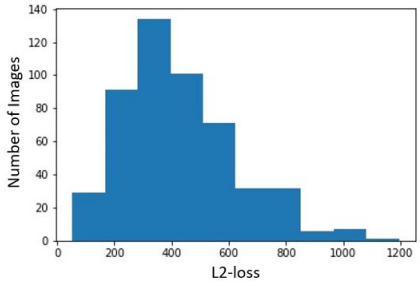

Figure 8: Validation Set Loss Distribution

6 Discussion

In light of the results, there are a few key points that are worth discussing.

6.1 Difficulties in Machine Learning

Given the small set of original samples that I had, the performance in picking out these four landmark points was very good. It is important to keep in mind that the 'true' PDM points I used are not

actually any distinct feature in the image. While the four points I choose are in a visually general region of the CC, they are not a true ground truth. This likely had a negative effect on my results. Given the results of my project, however, I think it would be very possible to build a neural network that could very effectively recognize a center of gravity for the CC about which a full PDM could be fit. This could serve as a valuable initial step before using something like Active Shape Models (ASM) to try and fit a more point dense PDM. While I didn't have time to test it concretely, I do not feel I would have been able to generate a complete 64 point PDM using this set of data. This illustrates a key weakness of machine learning compared to a traditional method such as ASM. ASM can perform very well even with a small number of images and produce a large number of accurate PDM points. With this little data, Machine learning fails. In light of this, given a small amount of data it makes more sense to design a hybrid method that uses machine learning to perhaps assist a traditional segmentation algorithm.

6.2 Hybrid Algorithm and More Data

As mentioned above, I think this project illustrates that a hybrid algorithm built on both machine learning and traditional segmentation methods would yield a superior result with a small data set. I can imagine using a network similar to mine that might give a set of landmarks that could then be used for any number of segmentation methods that we have studied.

Conversely, if I acquired many more samples, it would be possible to use machine learning end-to-end for segmentation. A well designed machine learning approach could be immune to scale, background noise, and maybe even modality. That would make it vastly superior to many of the segmentation algorithms that we studied. Once trained, neural networks are also very fast. An end-to-end neural network solution would likely produce an accurate PDM in far less time than almost any of the traditional algorithms we studied.

Overall, if data is scarce, it makes sense to use neural networks as part of a hybrid approach with a more traditional segmentation algorithm. Given an abundance of data, machine learning should be used for segmentation.

7 Conclusion

In conclusion, I think my project was very successful given the limited amount of training data available. The project also shows the importance of pre-trained models and creative neural network design when working with smaller data sets. Despite my success, the field of machine learning should seek to develop more standard and reliable techniques for working with smaller data sets. These developments would have a large impact on the field of medical image analysis.

Acknowledgments

I'd like to thank Dr. Pizer for his guidance throughout the semester.

Code

Code Location: https://github.com/SamuelDGeorge/Comp_775_Final_Project

Data: 'data' folder in github contains all the data I used for my project. The file 'Validation_Data_PDM.xlsx' contains the guess and true PDMs for all my validation data.

References

[1] K. He, X. Zhang, and et. al., "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *Computer Vision and Pattern Recognition*, vol. arXiv:1502.01852, 2015.

[2] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 740–755, Springer International Publishing, 2014.

[3] J. Yu, Y. Jiang, and et. al., "Unitbox: An advanced object detection network," *Computer Vision and Pattern Recognition*, vol. arXiv:1608.01471, 2016.

[4] S. Ruder, "An overview of multi-task learning in deep neural networks," *Machine Learning*, vol. arXiv:1706.05098, 2017.

[5] B. Ramsundar, S. Kearnes, and et. al., "Massively multitask networks for drug discovery," *Machine Learning*, vol. arXiv:1502.02072, 2015.

[6] C. Liu and et. al., "Progressive neural architecture search," *Computer Vision adn Pattern Recognition*, vol. arXiv:1712.00559, 2017.