Application of Convolutional Neural Network to Medical Image Detection and Segmentation

Shiyu Jin (3032131506, jsy@berkeley.edu),
Bo Yang (24978929, yangbo90@berkeley.edu),
Yujia Wu(24087824, yujia.wu@berkeley.edu)

Abstract

Medical images obtained from Magnetic Resonance Image (MRI) in real life carry useful information as well as garbage information, which comes from fat tissue, surrounding tissue, water drop, *etc.* In this report, Convolutional Neural Network (CNN) is used to process MRI images. Firstly, CNN is used to separate useless images from useful images. Then, CNN is applied to segment each of the useful images to separate the useful content of a MRI image, which in this case is disc tissue, from garbage information. TensorFlow is used to construct the CNN in Python. To demonstrate effectiveness of CNN, the prediction rate of categorize is shown, as well as the output image of segmentation.

1 Introduction

Medical imaging techniques such as Magnetic Resonance Image (MRI), Computed Tomography (CT) and Ultrasound Images provide us with high quality images of the anatomy and the physiological process of the body in both health and disease. These images enable quick and accurate medical diagnosis, brain function study, *etc.* For some cases, post-processing of these images are needed to extract useful information. For example, in biomechanics study, people process and binarize MRI scans of intervertebral disc Fig. 1(a) and then stack them to obtain a finite element model Fig. 1(b, c).

Currently, edge detection and intensity threshold method are being widely used for the post-processing (segmentation) and sometimes manual segmentation is also needed. These deterministic methods subject to many drawbacks: different scans can have different pixel brightness, threshold tuning is needed for different images, and manual work is time expensive and leads to a lot of bias.

In this project, we build two Convolutional neural networks (CNN) with the help of Tensorflow package to do image classification and pixel segmentation on the medical images. The first CNN divides useful image from useless images. The second CNN removes

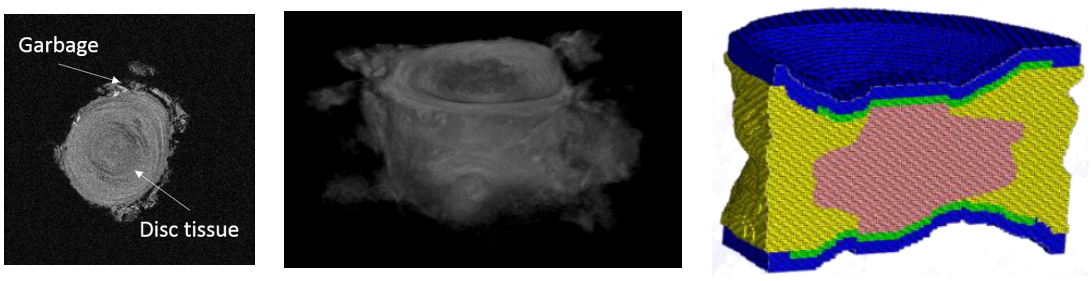

Figure 1: (a) A slice of MRI scan of bovine intervertebral disc (IVD), (b) Stacked MRI of IVD, (c) Voxel mesh used for IVD FEM simulation.

garbage tissue around disc tissue Fig.1(a), which returns pure disc region that can be used for finite element modeling. The rest of the report is organized as follows: Section 2 introduces data structure and data processing. Design of CNN is given in Section 3. Result of classification and segmentation is shown in Section 4. In Section 5, we discussed the pros and cons of CNN. Section 6 concludes the whole paper.

2 Data Structure

2.1 Raw Data

1082 slices of 256×256 resolution MRI scan for 9 bovine intervertebral disc samples have been collected from experiments, as shown in Fig. 2. Each pixel has an intensity magnitude between 0 and 255. The useful information in the MRI scans is the disc tissue, which is the bright round shape area in Fig. 2 (e, f), and bright ring shape in Fig. 2 (c, d, g, h, i, j). The black center of the ring shape indicates bone. There are some garbage information form each image, *fat dots, surrounding tissue, water drop*. For images contain mostly garbage information, such as (a, b) from Fig. 2, we regard the whole image useless.

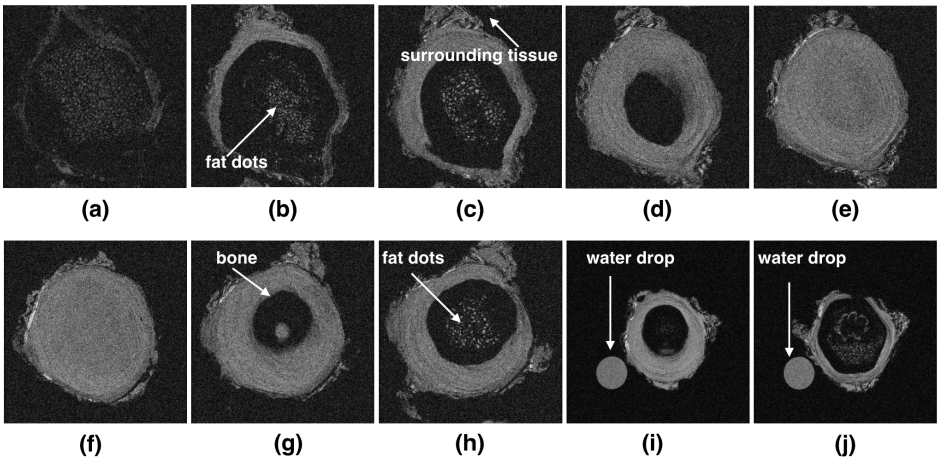

Figure 2: MRI scan of bovine intervertebral discs: (a) Scan for cross section in vertebral bone region; (b)-(d), (g)-(j) Bone disc interface scans; (e,f) scan of the disc region

2.2 Data Preprocessing

2.2.1 Labeling

As mentioned in the former part, the objective is to divide useful information from useless information, we binarize the pixels of these medical images to fully bright (if the pixel is useful) or fully dark (if the pixel is useless). A Matlab GUI has been developed to label the pixels semi-automatically Fig. 3. We used this GUI processed all images and obtained 256×256 labels matrix for each image.

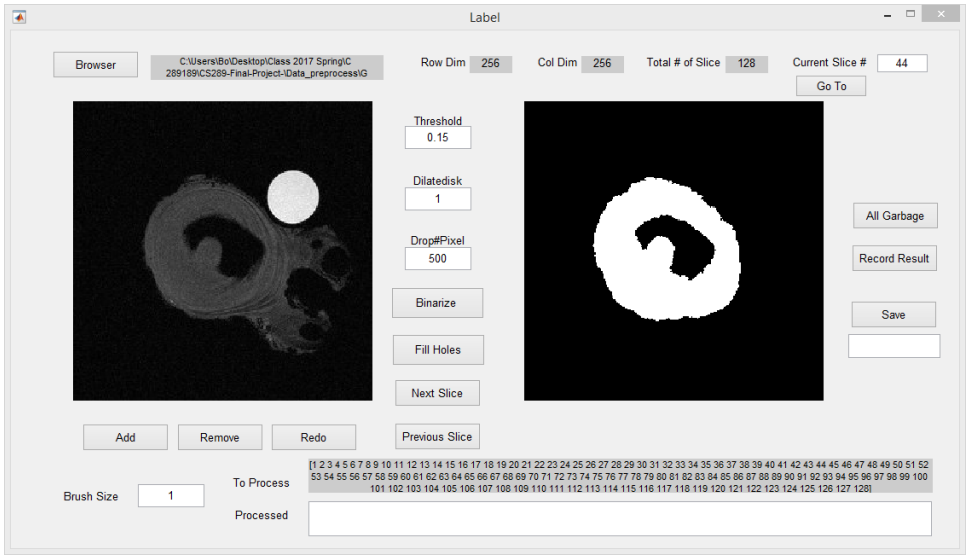

Figure 3: Matlab Gui for Labeling

2.2.2 Expanding, Data splitting, and Dawn Sample

Since CNN has many parameters to adjust, we realize sample size of 1092 may be not big enough for training, validating, and test a CNN. We generate more data by rotating images (90°, 180°, 270°). We split 5% data as validation set and testing set due to the limitation of original raw data size.

Table 1: Data sets and their dimensions.

Data	Dimension	Dimension (Down Sample)
Train	256× 256×7792	64× 64×7792
Validation	256× 256×432	64× 64×432
Test	256× 256×432	64× 64×432

Considering that it takes too long to train a CNN for 256×256 images and the limitation of our computation capacity, we decided to down sample all the images to 64×64 Fig. 4.

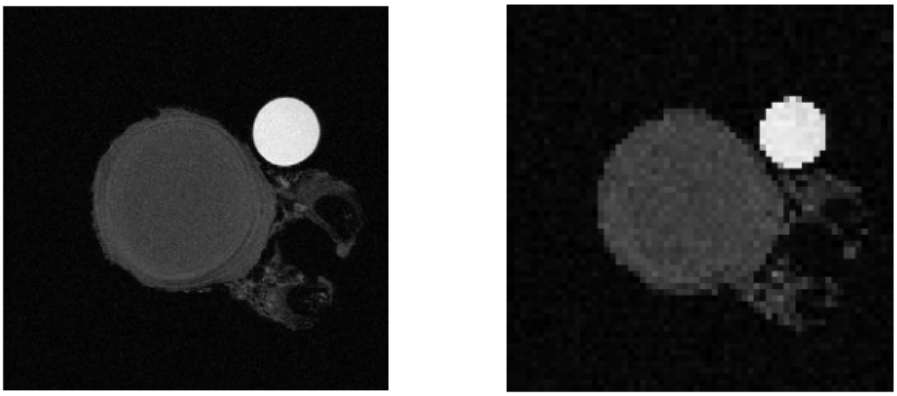

Figure 4: Original vs Down Sample

3 Design

3.1 CNN Design

In consideration of excellent performance, CNN is chosen for image classification (telling whether a image is all garbage), and image segmentation (telling a group of pixels are garbages or not). TensorFlow package [1] in python was used for CNN construction. The general structure of CNN is shown in Fig. 5.

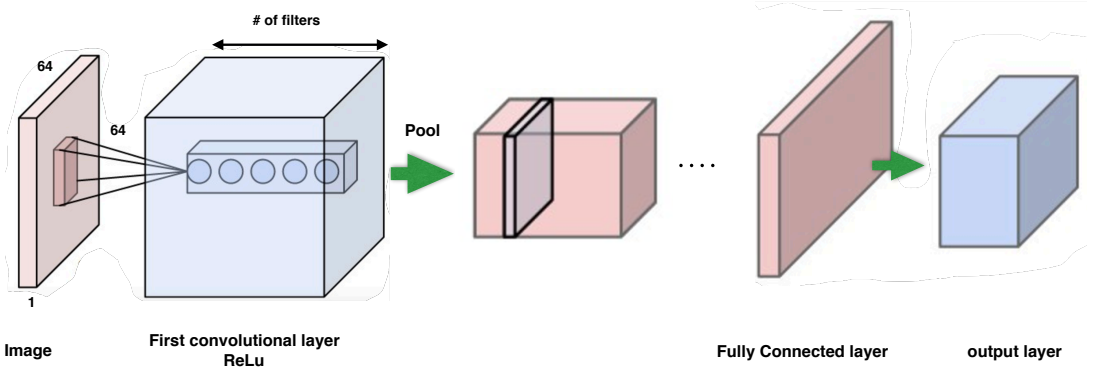

Figure 5: Structure of CNN used in this project.

In our design, we used three main types of layers to build CNN architectures: Convolutional Layer, Pooling Layer, and Fully-Connected Layer[2]. The sequence is:

⇒ Input layer
⇒ Converlutional Layer 1 (ReLu) ⇒ Maximum Pool Layer 1
⇒ Converlutional Layer 2 (ReLu) ⇒ Maximum Pool Layer 2
⇒ Fully connected
⇒ Output Layer

Input layer: Incorporated with down-sized image, the input layer has the dimension of 64×64. **Convolutional layer**. In this layer, N filters with $M \times M$ dimension is used here to perceive features of $M \times M$ group pixels. Zero padding is used here to maintain the dimensions after the convolution, *e.g.* after the first convolution layer, the data dimension should be $64 \times 64 \times N$. The parameter stride size denotes how many pixel will the filter move after each sampling move. In this project stride number is always chosen to be 1. After the convolution process, a ReLu activation function is applied element-wise, this will leave the dimensional unchanged. **Pool layer**. Max-pool is used to downsample the data. In this project, max-pool is chosen as a 2×2 filter with stride of 2. This means the max-pool layer will downsize the data 4 times. **Fully connected layer**. Fully connected layer is used to compute regulation result.**Ouput layer**. For the output layer, a sigmoid activation function is used. Also for the cost function, cross-entropy function related to sigmoid function is inherently applied here.

We developed two classifiers using this CNN: whole image classifier and image segmentation classifier.

3.2 CNN for Image Classification

First we want to design an Image Classier. Some MRI scan such as, Fig. 2(a), contain not useful information for modeling, so the whole image should be labeled as garbage. We first tune and test our CNN to classify if the whole image is garbage or not.

For image classification, the input and output dimensions are 64×64 and 1, respectively. The data size shrink from input to output implies that pooling layer will be useful to perform down_sampling process naturally. Two layers of convolution layer and pooling layer is used in this case. 4 filters are used in the this section with 3×3 dimension. After two layers of max-pool, the data shape has become 16×16.

3.3 CNN for Image Segmentation

After success in whole image classification, in the second step, we tune and test a classifier to classify each pixel, which can be used for segmentation.

The input and output dimensions now are both 64×64 . Since the sizes of input and output are same, the pooling layers, instead of improving the computation speed, will worsen the accuracy due to information loss. In this layer, to maintain dimension of each layer, no pooling layer is used, two convolutional layer with 4 filters of size 3×3 is used. Filter parameters will be tuned for different models.

4 Results

4.1 Image Classification

The weight and biases are initialized with a normally distributed random function with $\mu = 0$, and $\sigma = 0.02$. Cost decreases with number of epoch, training accuracy and validation accuracy increase with number of epochs, as shown in Fig. 6. We can achieve around 98% accuracy with the validate set, which means 98% validate images are correctly classified. In Fig. 7, a wrong classification is shown, the CNN labeled this as a useless image, and human labeled this as marginally useful image. As we could see from the graph the image contains only a thin circle of disc tissue, while has a lot of garbage information.

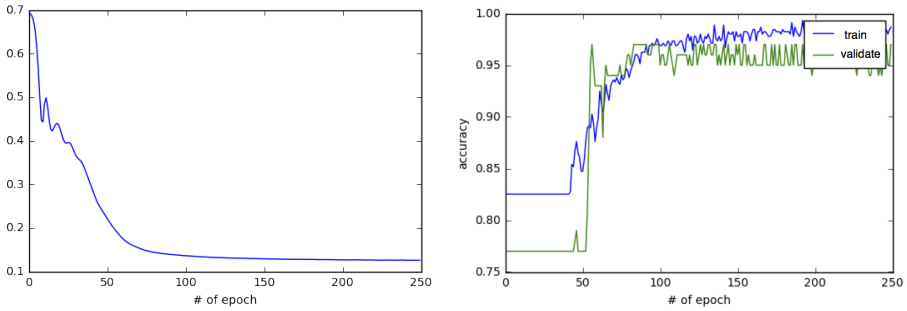

Figure 6: (a). Training cost of image classification. (b). Training and validation accuracy of image classification.

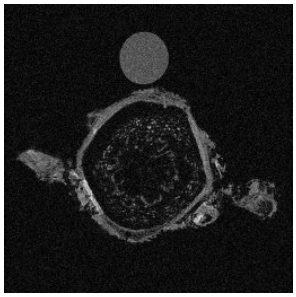

Figure 7: Wrong classified image: human labeled as useful, CNN predicts it to be useless.

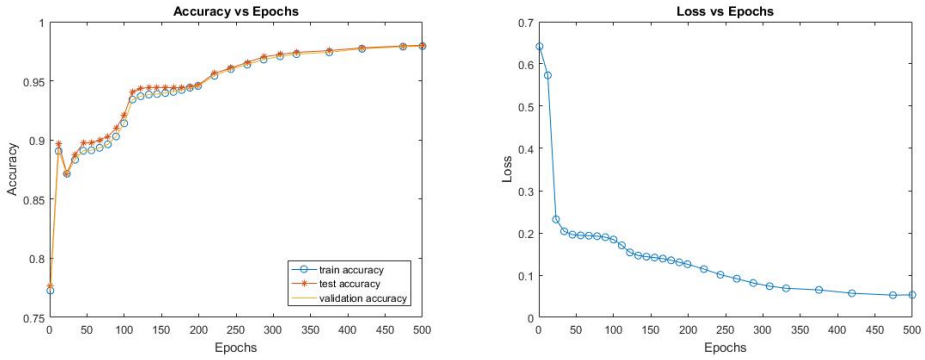

Figure 8: Accuracy vs number of epoch

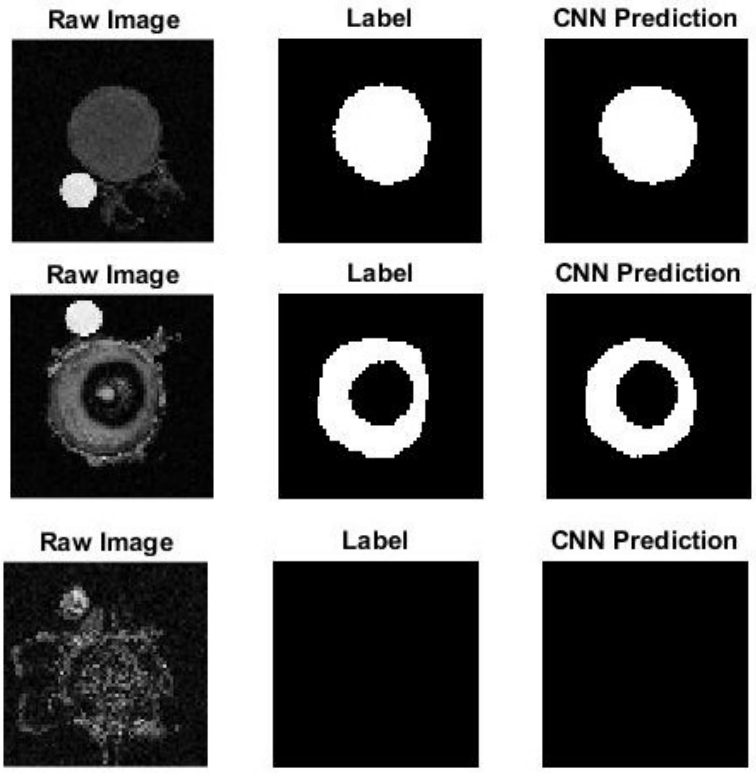

Figure 9: Well segmented disc

4.2 Image Segmentation

The weight and biases are initialized with a normally distributed random function with $\mu = 0$ and $\sigma = 0.001$. We train our model on a windows PC with 2.60GHz processor and 8GB RAM. It takes average 2 hours for our model to be trained, and it ended up with around 97.0% testing accuracy. This means on average for each test image, 97.0% pixels are correctly classified. With more filters(16 filters in the first layer and 32 filters in the second one) and 400 epochs(6 hours training), we get our best test accuracy 98.6% for image segmentation. The training, validation, and testing accuracies increase with epochs, as shown in Fig. 8.

With our CNN, most images were well segmented, as shown in Fig. 9. Pure water and fat that around disc were as well removed as we manually did; bone material inside disc was also well detected; useless image was also well classified by the segmentation classifier.

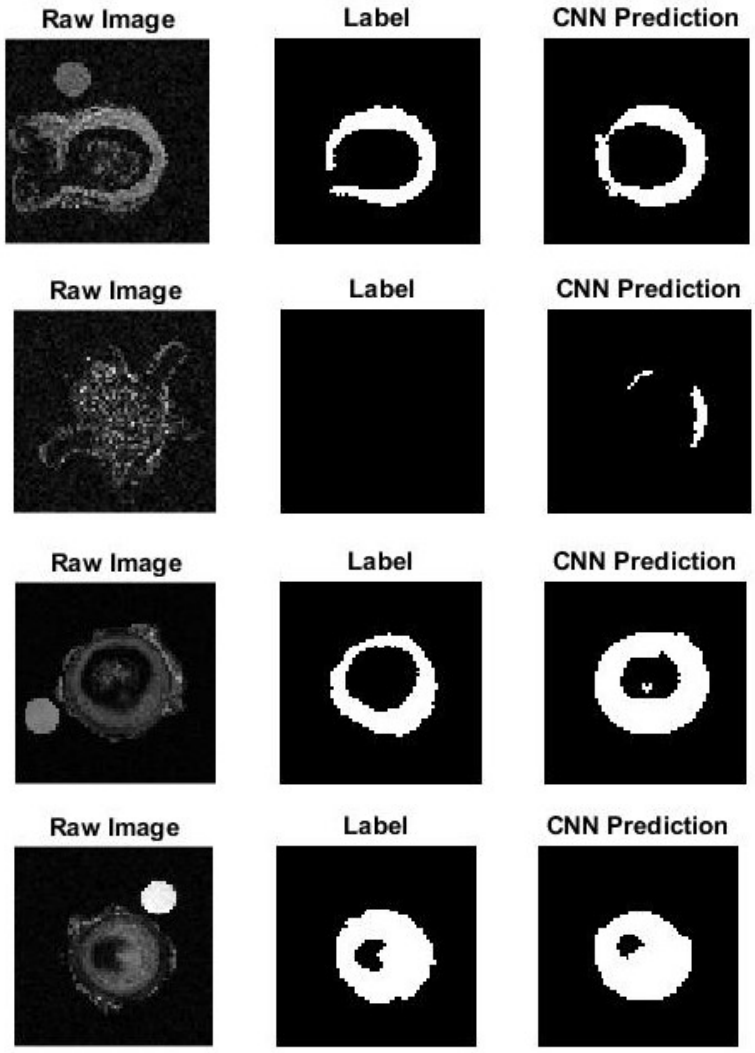

Figure 10: Not well segmented images

Some of the not-so-successful predictions were also shown in Fig. 10. Some parts of disc were not well segmented; useless image was classified as a small disc; curve boundary between bone and disc was not well detected.

5 Discussion

In general, our CNN achieved a good performance in both whole image classification and image segmentation. Some of the miss classification was tolerable due to the human bias in labeling, for example Fig. 7.

In image segmentation, as it can be seen in Fig. 8, we stop at 500 epochs, but accuracy can still increase with more epochs.

All three team member did labeling work, even though we set some general rules, bias is still considered to be big due to tedious manual work. We collected 8656 images, which is very limited for CNN training. To reduce computation time, we down sampled the original data by 4 resulting a information loss. Besides, a very small fully connected layer(288) was used, which may lead to a high bias in CNN.

6 Future Work

In the future, we would like to increase the sample size (currently 8656 samples) and improve the labeling consistence. We would also use full images (256×256) instead of down-sized images (64×64) for train and test. Some techniques such as drop out and ensemble would be used avoid over fitting and increase test accuracy. GPU will be used to train a bigger deeper CNN.

As we discovered in this project, CNN is a good method utilizing the pixel grouping paradigm. One other type of neural network, Recurrent Nerual Network (RNN), is have a premium performance utilizing the sequence paradigm of data. Both of CNN and RNN are good way to prevent overfitting, and sometimes RNN is better when the data set is small. One of future work is use RNN to do the same classification and segmentation work performed in this project and make comparison.

References

[1] Tensorflow: Deep mnist for experts. https://www.tensorflow.org/versions/r0.10/tutorials/mnist/pros/.

[2] Li, F.-F., Johnson, J., and Karpathy, A. Convolutional neural networks for visual recognition. *Lecture notes of cs231, Stanford University* (2016 Fall).