2D Retinal Vessel Segmentation using Convolutional Neural Networks

Kevin Huynh (704283105)
Computer Science
University of California, Los Angeles
Los Angeles, USA
kevinhkhuynh@gmail.com

Zicheng Liu (004460333)
Computer Science
University of California, Los Angeles
Los Angeles, USA
liuzicheng705@gmail.com

I. Introduction

In the medical domain, 2D retinal vessel imaging using fundus cameras is an integral tool in diagnosing "various cardiovascular and ophthalmologic diseases such as diabetes, hypertension, arteriosclerosis, and chorodial neovascularization" [1]. To extract more relevant medical information from these types of images, it is necessary to segment them, separating the retina blood vessels from the non-vessel tissue surrounding them. When done manually, this process is an extremely tedious task which takes an exorbitant amount of time and training to do [1]. Over the past two decades, many researchers have focused on developing computer-assisted techniques and machine learning algorithms to automatically segment these images so that health practitioners can diagnose patients more efficiently. Many of the current state-of-the-art systems build off of baseline techniques that divide into four categories: region, clustering, and classification methods [6].

The highest-performing models of the past few years in image segmentation are complicated variants of classification methods that attempt to discriminate between data points by labeling them with a distinct class [6]. Many in particular are based upon the max-pooling convolutional neural network (CNN) [5], and this is a topic we are particularly interested in and would like to explore. A max-pooling convolutional neural network is a CNN that alternates between convolutional filtering layers and max-pooling layers before finally outputting to a fully connected layer that predicts the most probable class of the pixel [4]. Because the filters extract local features from the images, CNNs do particularly well for retinal vessel image segmentation [5].

In this project, we built a machine learning model, specifically a max-pooling convolutional neural network [4], to automatically segment 2D retinal blood vessel images. We implemented this neural network using Python and TensorFlow, an open source machine learning framework built by Google Brain, and subsequently trained and tested our model on the publicly available dataset DRIVE: Digital Retinal Images for Vessel Extraction [7]. This dataset consists of 20 training and testing retinal images along with manual blood vessel segmentations completed by a human expert. After using these manual segmentations to train our convolutional neural network in a supervised setting, we evaluated our model on the DRIVE testing images using metrics such as AUC (Area Under the Curve), precision-recall, maximum average accuracy, and Cohen Kappa score and compared its performance to a variety of other systems with published results on the same dataset [2]. Our code is accessible on GitHub[1].

II. Architecture of the Model

Our neural network was derived from the architecture of the max-pooling convolutional neural network [4]. Specifically, our model consisted of the following structure: [input image patches] $->$ [conv - relu] $->$ [conv - relu - pool] x 3 $->$ flatten $->$ affine $->$ relu $->$ dropout $->$ affine $->$ softmax. First, our images were divided into image patches through a pre-processing step that we will explain later in the Methods section. For the convolutional layers, filters of size 6 x 6, 5 x 5, 4 x 4, and 2 x 2, all with stride 1, were used over 48 image maps to extract local features from the image patches. Similarly, for all of the max-pooling layers, windows of size 2 and stride 2 were used; however, padding was used to ensure that the image maps remained the same size. Furthermore, rectified linear units (ReLu) was used to speed up the training process by mapping all negative values to zero or positive values [3]. After flattening the maps created by the last max-pooling layer, the affine and softmax layers, consisting of vanilla, fully-connected neural networks with 512 hidden units each, act as a binary classification layer, classifying whether a given pixel is part of a retina blood vessel or not. Lastly, dropout, a common regularization technique for neural network, was also used to prevent overfitting [3]. The architecture of our model is shown in Figure 1.

III. Methods

A. Preprocessing the Images

Before training the model, we pre-processed the 21 training images. This pre-processing step is important because there are not enough training images to adequately train a data hungry model such as the neural network. Thus, artificial image patches must be created to supplement the dataset. This pre-processing step included dividing our images into

[1] https://github.com/kevinhkhuynh/CS168

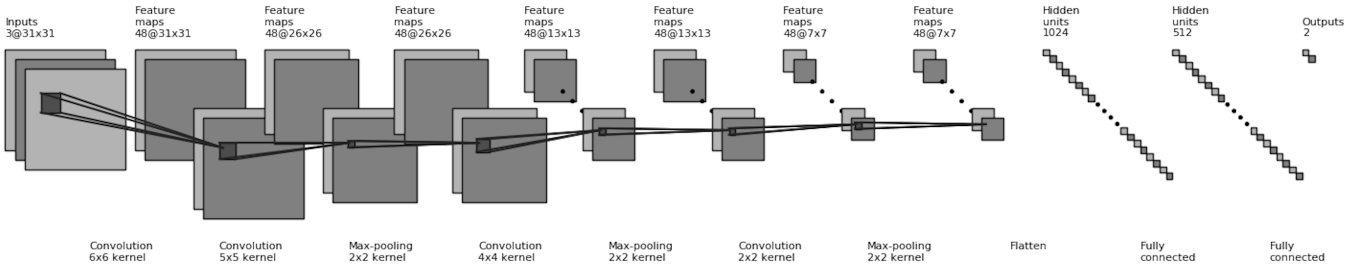

Fig. 1. CNN Architecture

120,000 patches of size 3 x 31 x 31, normalizing the images by subtracting the mean intensity of the pixels, and randomly shuffling the data. The images must be normalized and shuffled to prevent any biases within the training data from affecting the performance of the model. Note that the image patches contain an arbitrary proportion of positive and negative samples.

B. Training the Model

Afterwards, we built our own neural network on top of neural network training code leveraged from another source built on TensorFlow. Using a cross-entropy loss, a loss function standard for classification models, we trained our network using an Adam optimizer with a learning rate of 5×10^{-4} for 20000 steps, a batch size of 256 images, and a dropout probability of 0.5. The Adam optimizer was chosen because of its recent popularity and promising performance in a variety of other machine learning research fields [2].

C. Tuning the Hyperparameters

We trained 13 models of differing number of hidden units and dropout probabilities and ran cross-validation to choose the version that has the highest mean accuracy on the testing images as our model.

Model	# of Hidden Units	Dropout Probability	Mean Accuracy
1	512	0.6	0.9203
2	**512**	**0.5**	**0.9231**
3	512	0.4	0.9229
4	512	0.3	0.9201
5	512	0.2	0.9189
6	512	0.1	0.9203
7	256	0.5	0.9220
8	256	0.3	0.9229
9	256	0.1	0.9201
10	128	0.9	0.9189
11	128	0.6	0.9203
12	128	0.3	0.9220
13	128	0.1	0.9229

TABLE I
MEAN ACCURACY OF THE 13 DIFFERENT MODELS

The results shown in Table 1 demonstrate that there doesn't seem to be much difference between accuracy between these different neural networks considering their hyperparameters; however, we take model 2 to be our neural network. To prevent overfitting, we further validated the neural network against the testing images after every 500 steps. We chose the optimal number of steps to stop training as the threshold in which the mean accuracy on the testing images no longer increased. Figure 2 below shows that the optimal number of steps is 14,000.

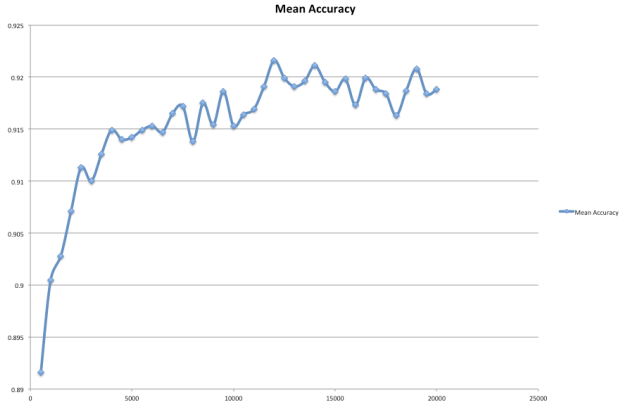

Fig. 2. Step vs. Mean Accuracy

IV. RESULTS

Using the tuned convolutional neural network we found in the previous section, we segmented the images by classifying each pixel as a part of a retina blood vessel or not. We then used metrics including the receiver operating characteristic (ROC) curve, the precision-recall curve, average precision score, maximum average accuracy score, and Cohen Kappa score to measure the performance of our model.

The ROC curve has the false positive rate on its X axis and the true positive rate on its Y axis. The larger the area under the curve (AUC), the better the performance as it signifies that our model has a low amount of false positives. Furthermore, the slope of the ROC curves indicates what threshold is best to minimize the false positive rate while simultaneously maximizing the true positive rate. As observed in Figure 3 below, our model has a AUROC of 0.9418, which is similar to other state-of-the-art models [3].

Next, the precision-recall curve has the precision of our model on its Y axis and the recall on its X axis, in which the

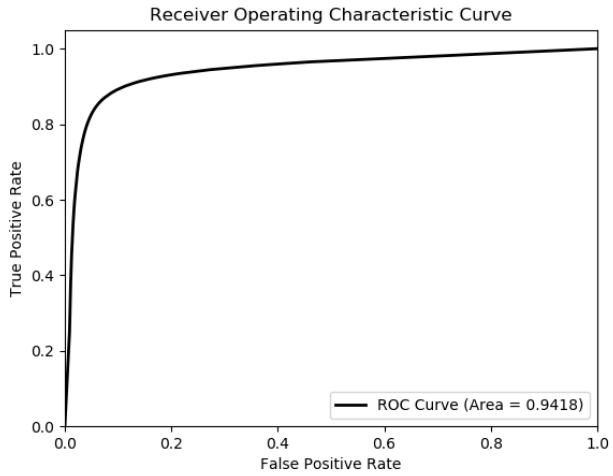

Fig. 3. ROC Curve

precision measures the percentage of true positives over all positive predictions, and the recall measures the percentage of true positive predictions over all true positive and false negative labels. In other words, high precision indicates that the model is returning accurate results, while high recall indicates that the model is returning a majority of all true positive labels. Similar to the ROC curve, the larger the area under the precision-recall curve, the better the performance of the model. As observed in Figure 4 below, our model has an average precision of 0.6729. The reason between the significant difference in score between the ROC curve and the precision-recall curve is the large class imbalance inherent in the data. Over 90% of pixels are not a retina blood vessel, and so the number of negative samples is much greater. Because the precision-recall curve only looks at how our model performs on the positive samples, it is clear why there may be such a discrepancy.

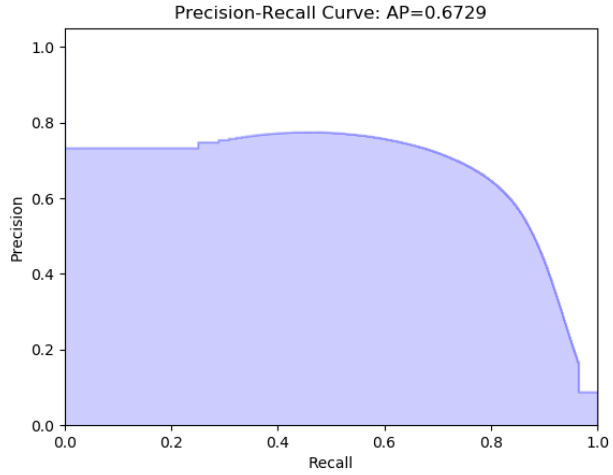

Fig. 4. Precision-Recall Curve

The average accuracy score is computed by obtaining the average of the accuracy score across all the pixels in all the testing images, where the accuracy is the fraction of correct predictions. We found that in our model, treating only pixels who have a likelihood of being part of a retina blood vessel greater than 94%, gave us the best results. With this threshold, we achieved a maximum average accuracy score of 0.9430.

Lastly, the Cohen's Kappa score is a statistic that measures inter-rater reliability and expresses the level of agreement between two annotators on a classification problem. In our model, we achieved a Kappa value of 0.6795, which means that our model agrees with the manual annotator for 67.95% of the pixels in the testing images. This may seem unsatisfactory, but the second manual annotator only achieves a Kappa score of 0.8213 against the first annotator as shown in Table 2 below. This table compares the maximum average accuracy score and the Kappa score of our model with other state-of-the-art competitors. In terms of these two metrics, our CNN does not perform the best; however, it is comparable to the state-of-the-art and performs very well considering its simple architecture.

Model	Max. Avg. Accuracy	Kappa Score
Our Proposed CNN	**0.9430**	**0.6795**
Maji et al.	0.9470	0.7031
Sheet et al.	0.9766	0.6287
Second Manual Annotator	0.9473	0.8213
Staal et al.	0.9422	-
Niemeijer et al.	0.9416	0.7145
Zana et al.	0.9377	0.6971
Jiang et al.	0.9212	0.6399
Martinez-Perez et al.	0.9181	0.6389
Chaudhuri et al.	0.8773	0.3357

TABLE II
COMPARISONS TO OTHER STATE-OF-THE-ART MODELS [3]

V. DISCUSSION

Currently, our model can be improved in a number of ways. Firstly, we can append more data to our training set by pre-processing the images more complexly, including rotating and scaling the images before obtaining the image patches. Secondly, we can further tune the hyperparameters by changing the filter sizes and strides of the convolutional and max-pooling in our model and by changing the learning rate of the Adam optimizer. With an un-optimal filter size, the most discriminative features may not be extractable. Thus, fine-tuning the filter size can improve our model's performance as it plays an important role in local feature extraction. Similarly, the learning rate can drastically affect the ability of the model to converge. We can try different learning rates to see if the model performs better. Thirdly, we can look into other state-of-the-art models and adapt some of their defining components to our model's architecture. It is safe to assume that by doing so, our model's performance will increase. In the end, while our CNN may not have beat the state-of-the-art, it has similar image segmentation performance to a human medical expert and can be extremely useful as an assistant for healthcare

professionals who must manually segment 2D retina blood vessel images every day.

REFERENCES

[1] M. M. Fraz et al., "Blood vessel segmentation methodologies in retinal images - A survey," Computer Methods and Programs in Biomedicine, 2012, vol. 108, no. 1, pp. 407-433.

[2] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," arXiv:1412.6980 [cs], Dec. 2014.

[3] D. Maji, A. Santara, P. Mitra, and D. Sheet, "Ensemble of Deep Convolutional Neural Networks for Learning to Detect Retinal Vessels in Fundus Images," arXiv:1603.04833, Mar. 2016.

[4] J. Masci, A. Giusti, D. Ciresan, G. Fricout, and J. Schmidhuber, "A fast learning algorithm for image segmentation with max-pooling convolutional networks," 2013, pp. 2713-2717.

[5] M. Melinscak, P. Prentasic, and S. Loncaric, "Retinal Vessel Segmentation using Deep Neural Networks:," 2015, pp. 577-582.

[6] A. Norouzi et al., "Medical Image Segmentation Methods, Algorithms, and Applications," IETE Technical Review, 2014, vol. 31, no. 3, pp. 199-213.

[7] J.J. Staal, M.D. Abramoff, M. Niemeijer, M.A. Viergever, B. van Ginneken, "Ridge based vessel segmentation in color images of the retina", IEEE Transactions on Medical Imaging, 2004, vol. 23, pp. 501-509.