M2CAI Surgical Tool Detection Challenge Report

Ashwin Raju, Sheng Wang, and Junzhou Huang

Department of Computer Science and Engineering,
University of Texas at Arlington, Arlington TX 76019, USA

Abstract. In this report, we describe our method for the Surgical Tool Detection Challenge. Our method is mainly based on the ensembling two most used deep neural network architecture for classification task – GoogleNet and VGGNet. We have customized the parameters in both models to achieve better accuracy. On our training and validation data sets, both the methods get more than 0.75 hamming scores. On the test set evaluated by M2CAI challenge, our model achieves 63.7% mean average precision, leading to one of the top three methods in the challenge.

1 Introduction

Convolutional Neural Networks (CNNs) are the current trend in machine learning and computer vision tasks. One of the tough problems in general machine learning and computer vision is feature extraction. Traditional feature extraction requires expert knowledge about the data and the methodology differs in each task. We can automatically extract more complicated and useful features by training CNNs. Beside, with the advances in high speeding computing, CNNs become the-state-of-art methods for machine learning and computer vision areas.

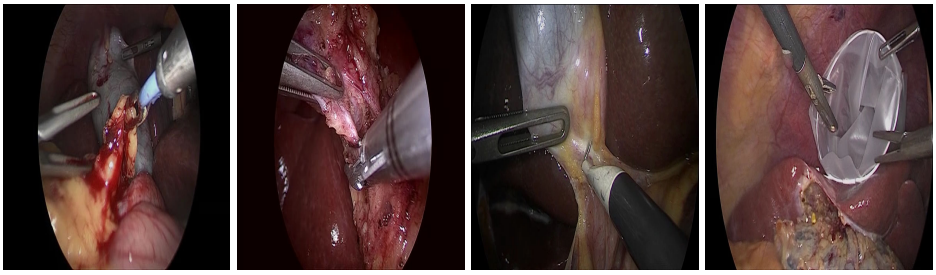
In our method, we have taken the advantage of CNNs to solve this problem. Our model uses ensemble of two different deep CNN models (GoogleNet [10] and VGGNet [9]). VGGNet won the ImageNet competition in 2014 and GoogleNet was the champion for the ImageNet competition in 2015. Both of the two models are widely used for image classification tasks. In our method, we train both the networks and ensemble the results of the two networks.

2 Data description

2.1 Data source and data sets description

All the data we used in our training, validation and testing stages are from the data given by this challenge. We didn't use any extra data. We increased the dataset size by data augmentation in the training process.

For the training data, since the raw data are videos with frames annotated at 1 fps. We extract frames from the video and took only the frames that are



$[1,1,0,0,0,0,0]$ $[1,0,0,0,1,0,0]$ $[1,0,1,0,0,0,0]$ $[1,0,0,0,0,0,1]$

Fig 1. Different images and their corresponding vector labels. The first image from left has the label $[1,1,0,0,0,0,0]$, which is of the form [grasper, bipolar, hook, scissors, clipper, irrigator and specimen bag]. Thus,for the first image, both grasper and bipolar are present.

given in the annotation file. For each model, we use 5-fold cross validation with random shuffle and splitting the data with 90% as training data and 10% as validation data. We use about 21000 original training images.
For the testing data, we extracted the videos into frames similar to training data. Fig 1 shows different images and their corresponding labels extracted from the annotation file in such order: grasper, bipolar, hook, scissors, clipper, irrigator and specimen bag.

2.2 Data preprocessing and augmentation

All the images are resized to $224 \times 224 \times 3$ so the images can be directly used in the GoogleNet and VggNet. The images are later normalized by dividing each pixel by 255 so that the range of pixels will be between [0,1) rather than [0-255). The size of the dataset is increased by performing real time image augmentations randomly in each epoch of the whole training process. We find that real time augmentation produced better result than off-line augmentation. The image augmentation methods include image rotation, horizontal flipping and vertical flipping. Any of the three augmentation is taken at the probability of 0.5.

3 Methodology

We consider this challenge a multi-label classification problems [3]. Thus, we use two most used image classification deep neural network models: GoogleNet and VGGNet. In this section, We explain the detail of the two models, loss function, our evaluation metric and optimization method used for network training.

3.1 Two Models

VGGNet VGGNet has 16 layers with convolution and max pooling layers. The architecture uses the same 3×3 kernel size for convolution and pooling in all

layers. We initialize the network weights with [4]. For the activation function ReLU [7] is used in our VGGNet training. We also use Batch Normalization in this architecture. The model is trained with 5-fold Cross validation with 90% of the data as training and 10% as testing data. Some of the hyper-parameters and settings are listed in Table 1.

Hyper-parameters or Settings	VGGNet
Loss function	Sigmoid Cross Entropy with Logits [1]
Optimization Method	Adam [6]
Activation Functions	ReLU [7]
Use Batch Normalization [5]	Yes
Weights initialization	HE-Normal [4]
Initial Learning Rate	0.0001
Momentum	0.9
Batch Size	32
Epochs	1000

Table 1: Hyper parameters or settings used in VGGNet

GoogleNet GoogleNet has 22 layers and in each layer it has an inception block. We have used batch normalization after each convolution operation. For the activation function, we use Leak ReLU [11]. The batch size used in GooglNet is 64 while in VGGNet it is 32. The main reason for this difference in size is that GoogleNet has 3 times less weights parameters than VGGNet. We initialize the weights from model trained in ImageNet competition. The pre-trained weights are initialized from Model Zoo Caffe. The model is trained with 5-fold Cross validation with 90% of the data as training and 10% as testing data. Some of the hyper-parameters and settings are listed in Table 2.

3.2 Ensembling

We have 10 trained models for both VGGNet and GoogleNet since we use 5-fold cross validation for each of them. In ensembling process, we ensemble the 10 models together to get the final results. We tried 3 methods to ensemble the results that we got from test data. The three methods are normal averaging, weighted average and geometric average. We find that normal averaging produced much satisfying results on the validation set so we choose normal averaging in our method.

Hyper-parameters and Settings	GoogleNet
Loss function	Sigmoid Cross Entropy with Logits [1]
Optimization Method	Adam [6]
Activation Functions	Leaky ReLU [11]
Use Batch Normalization [5]	Yes
Weights initialization	Pre-trained model
Initial Learning Rate	0.0001
Momentum	0.9
Batch Size	64
Epochs	1000

Table 2: Hyper parameters or settings used in GoogleNet

4 Results on the Validation Set

All our implementation are build on TensorFlow [2]. For our own evaluation, we use hamming score [8] as accuracy evaluation metric on our validation set to evaluate the performance of the trained models. Before ensembling, we get around 78% validation accuracy using GoogleNet and 75% validation accuracy using VGGNet.

5 Conclusion

With the ensemble of the two models we are able to get mean average precision of 63.7% on the test data evaluated by M2CAI challenge, placing us among the top-3 positions on the leader board.

References

1. Sigmoid cross entropy with logits. `https://www.tensorflow.org/versions/r0.11/api_docs/python/nn.html#sigmoid_cross_entropy_with_logits`, accessed: 2016-10-06
2. Tensorflow. `https://www.tensorflow.org`, accessed: 2016-10-06
3. Bi, W., Kwok, J.T.Y.: Efficient multi-label classification with many labels. In: ICML (3). pp. 405–413 (2013)
4. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1026–1034 (2015)
5. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
6. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
7. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10). pp. 807–814 (2010)

8. Norouzi, M., Fleet, D.J., Salakhutdinov, R.R.: Hamming distance metric learning. In: Advances in neural information processing systems. pp. 1061–1069 (2012)
9. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
10. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Computer Vision and Pattern Recognition (CVPR) (2015), http://arxiv.org/abs/1409.4842
11. Xu, B., Wang, N., Chen, T., Li, M.: Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853 (2015)