Computational Biology

Automated Particle Picking in Cryo-EM

Qinyao He, Dayu Yue, Xi Ye

1. Overview

In this project, we train several neural network (from simple multi-layer perceptron to convolutional network) to classify whether a crop of image (Cryo-EM) contains a particle. We make use of those network to detect position of particles in a full size Cryo-EM image.

2. Our Approach

2.1. Preprocessing

In order to provide the training data for neural network, we cropped positive and negative sample with size 180*180 from the original large image, the crop was then resized to 64*64.

For there are about 170 particles in each image, which is a relatively small number. To make it enough and various for training a neural network, we come up with two ideas for data augmentation.

a) For every particle image sample, we rotate the image for 90 degrees, 180 degrees and 270 degrees, together with the image itself, we get four positive samples in all. As for the negative samples, we randomly crop the patches whose center is at least 40 away from those of all particles. This method of sampling focus on the particle itself and we expect a high precision because it would be relative difficult for over classifier to regard a patch as a particle.

b) We consider that the previous method may not be able to provide enough varieties of particle. When we are detecting, it is not always possible to locate the extract center of any particle. So we propose another method. For every particle, we crop both the particle and some patches that have small offset from it (we adopt 30 in practice because the maximum of offset would be 36). And we take the same negative-

sampling method as a), but this time 0.3 * particle size is required for the minimum distance from particles.

In this way we gain 70000 train image with half positive and negative examples.

2.2. Multi-Layer Perceptron

For simplicity and faster training for validation on continuous process, a simple multilayer neural network with two hidden layer is built. We treat the input as $64 * 64 = 4096$ dimention vector, with 256 output unit in each hidden layer, and the last layer output one value as the probability for the positive hypothesis (is a particle).

2.3. Convolutional Neural Network

Convolutional Neural Network has recently been proved to have state-of-art performance in image-related tasks. Here we can regard the Cryo-EM as a large image, and we are going to learn and detect some pattern which correspond to the particle. CNN seems to be very good at extracting high level feature from the original image (without preprocessing and feature engineering) automatically.

We follow the practice of VGG-Net, use all 3*3 convolution filter size, with padding 1 to make sure the size unchanged after convolution. And a 2*2 max pooing is followed every one or two convolution layer. Channel size is doubled every one or two max pooing. And two dense layer is added after last convolution and before loss layer.

Generally, we stop add more convolution when image size is 2*2, end up with 5 convolution layer and two full connected layer.

2.4. General Settings

In deep convolutional network, weight initialization can greatly influence the convergence of training process. Bad initial weights can cause the network never converge. In this project we follow the practice suggested by Kaiming, He, which in practice guarantee convergence to as many as dozens of layers.

Input image is rescaled to 0~1 from the original 0~255, by dividing 256. We add dropout layer before every full connected layers to prevent overfitting. The final layer is activated using sigmoid function. The network has only one output value, represent the probability of detecting a particle. And we use binary cross entropy as loss function

2.5. Particle Detection

3. Experiments

3.1. Training Network

In all the network training, we use Stochastic Gradient Decent (SGD), which is commonly used in deep neural network training, and turns out to be the current best practice.

Since a lower learning rate generally push network to a better convergence, learning rate is set to 0.001 initially, and decay by 10^{-4} after each iteration (updates), with momentum set to 0.9. Batch size is 128. We train each network with 100 epoch.

The multilayer perceptron with two hidden layer tend to work very well, had a validation accuracy of 0.955 after 100 epochs.

The CNN seems to have no significantly improvement over MLP, on our limited configuration. Our best result has 0.965 accuracy on validation dataset, while require hundred times of time to compute.

3.2. Particle Detection

We do sliding window with size 180*180 and stride 36 on the original large image, and sent the cropped image to pre-trained network. We choose those with output probability greater than its surroundings, and larger than a threshold, which is 0.8, as the chosen particles. When use MLP, we get a precision of 0.3 and recall of 0.4. The CNN model is too large that we don't have enough time for validation.

We are still making better algorithms for this tasks. Our best result till now is precision of 0.5 and recall of 0.67. Some effort still to be made to have it synthesized

4. Conclusion

References

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics* (pp. 249-256).

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In*Proceedings of the IEEE International Conference on Computer Vision* (pp. 1026-1034).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting.*The Journal of Machine Learning Research*, *15*(1), 1929-1958.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley and Y. Bengio. “Theano: new features and speed improvements”. NIPS 2012 deep learning workshop.

J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley and Y. Bengio.“Theano: A CPU and GPU Math Expression Compiler”. *Proceedings of the Python for Scientific Computing Conference (SciPy) 2010. June 30 - July 3, Austin, TX*.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., ... & Darrell, T. (2014, November). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia* (pp. 675-678). ACM.