Project Report 1- Neuroscience

Han Shen
Institute of Interdisciplinary Information Sciences
Tsinghua University
thushenhan@gmail.com

Abstract

MNIST is a public dataset used for digital image classification problem. Neural Networks have been proved to be the best classifier for this problem so far. Here we have used three different neural networks, multilayer perceptron, convolutional neural networks, and spiking neural networks, training on 60000 labelled images and testing on rest 10000 images. All of these neural networks have only two hidden layers but with slightly difference on architecture. CNN has shown best results, which achieves near 0.1% top-5 error rate and 8% top-1 error rate in 10 epochs. Correspondingly, MLP achieves 1% on top-5 error rate and 9.5% top-1 error rate in 30 epochs. By training a MLP network, we migrate the coefficients to LIF model neurons, constructing a Spiking neural network(SNN). SNN performs slightly worser than original MLP classifier, but ... .

1. Introduction

Object detection is a major research topic in computer vision field. Digital number recognition, however, is a subproblem of Optical Character Recognition(OCR) problem which has been researched over years. OCR was widely used to convert scanned documents, text photos, handwritten photos into machine-encoded text. MNIST is a large database of handwritten digits that is commonly used for training various image processing systems. It contains 70000 labelled written digit images, with 60000 for training and 10000 for testing. All images are grey and formalized to have same size 28×28 Recently, neural networks are proved to be extremely suitable for object detection tasks, when provided with large amount of labelled data. In the category of neural models, adding more hidden layers always endows the network larger capacity to fit more complicated model, which leaves early handy feature engineering methods out-dated. We implemented three neural models, including MLP, CNN, and SNN, comparing their performance on MNIST dataset. We adopted MatConvNet as the basis for building neural network architectures, and compared the influence of different activation functions, optimization approaches of CNN such like dropout and batch normalization.

2. Methods

2.1. MLP

MLP is constructed by input layer, output layer and hidden layers. Successional layers are fully-connected, and there is no connection between neurons in same layer. We use MatConvNet to construct a two hidden layer MLP, with 64 and 128 neurons, correspondingly. The output layer is a softmax layer with 10 output neurons. The weights are randomly initialized and the model is trained for 30 epochs with batch size of 128. Though MatConvNet is specially developed for CNN, we can use it to construct MLP since MLP is a special case of CNN. To fit MLP in CNN structure, we adopted the 4-tensor of (28,28,1, 64) for the first hidden layer, and (1,1,64,128) for the second. In this case, no weight-sharing among pixels and a 784-64-128-10 MLP is successfully built. We compared the performance of using two different activation functions, sigmoid and ReLU based on this architecture.

2.2. CNN

Since convolutional neural network usually contains convolution layers and pooling layers, we tried out a two layer convolutional network with a convolution layer of and a pooling layer. The parameters shows in Fig 3. Constructing such neural networks by MatconvNet are straightforward. Normalizations and Dropout are regular optimization method people usually adopted to improve CNN's performance. Here we did some experiments among models training with or without them.

2.3. SNN

Spiking Neural Network(SNN) is categorized into third generation of neural network models. It incorporate the concept of time into operating models, with the consideration of actual way how human neurons fires— That is, not

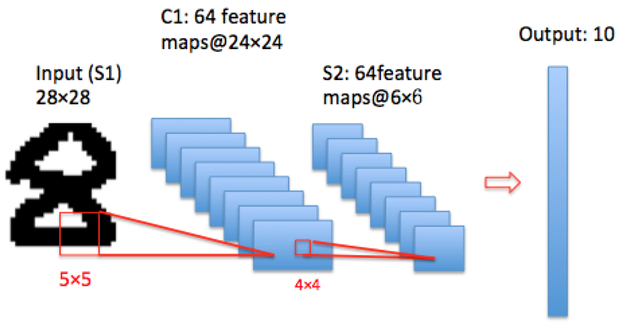

Figure 1. Two hidden layer convolutional neural network, the upper network uses two convolutional layer, while the bottom network uses a convolutional layer along with a pooling layer.

fire at each propagation cycle, but fire only when a membrane potential reaches certain threshold. SNN consists of leaky integrate-and-fire neurons, whose membrane potentials follow

$$\frac{dV}{dt} = \frac{1}{\tau_m}(E_L - V + R_m I_e) \tag{1}$$

where $V(t)$ denotes the membrane potential, E_L denotes the leakage potential, R_m denotes the total membrane resistance, I_e denotes the input current. Whenever $V(t)$ gets larger than or equal to $V_{threshold}$, fire a spike and reset $V(t)$ to a resetting potential V_{reset}. For simplicity, here we use only integrate-and-fire model, that is ,

$$\frac{dV}{dt} = \frac{1}{\tau_m}(R_m I_e) \tag{2}$$

In SNN, we replace I_e with the summed weighted spiking input of previous layer.

2.4. Build SNN based on MLP

According to Work from Cao, et. al, to construct a SNN, we can firstly train a tailored MLP, which means to use ReLU after every fully-connected layer, remove all bias. We then migrate the MLP to SNN, adding a SpikeGenerator model between normalized input and first hidden layer, a SpikeCounter after last fully connected layer.

$$V(t) = V(t-1) + dt * R/\tau * X(t) \tag{3}$$

$$X(t) =< A(t), K > \tag{4}$$

Denote I_{ij} as the processed image input, SpikeGenerator produces a spike if $rand() < cI_{ij}$. Here we set c to be 1/3.

The reason why SNN directly works in MLP trained weights is that in MLP, final classifier concerns the magnitude of output, while in SNN, final classifier concerns the firing rate. If we control the neuron firing parameter not to change linearity and non-linearity of MLP, then the output neuron with largest spiking rate is right the predicted class.

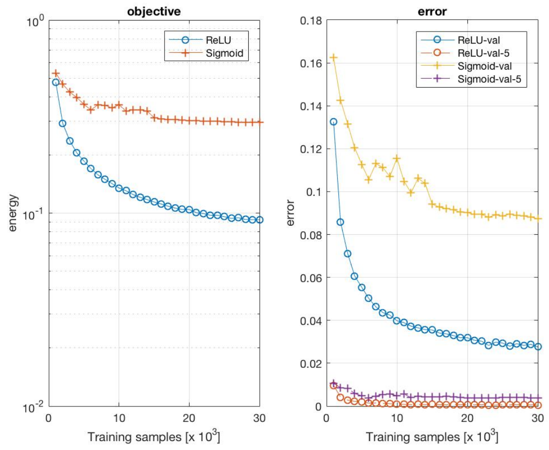

Figure 2. MLP Result With ReLU/Sigmoid Activation Function

3. Results

3.1. MLP

We trained the MLP with both ReLU activation function and logistic sigmoid function. Final result shows in Fig 2. Using sigmoid is easy to saturate, thus we should initialize weight and bias small, and modify learning rate carefully. To make sigmoid case converge normally, we adopt learning rate as 0.1 at the beginning, while adopt 0.01 after 15 epochs. As the figure shows, ReLU case achieves top-1 error rate around 3%, while Sigmoid case achieves around 9%, which is 3X worser than the former. With ReLU, MLP achieves 0.1% top-5 error rate.

3.2. CNN

The result of CNN in different configuration cases shows in Fig 3. Surprisingly, neither batch normalization nor dropout achieves better performance than the baseline. The best performance CNN achieves around 1.3% for top-1 error and 0.1% for top-5 error. Adding dropout results in slightly harm in performance, while batch normalization results in large performance decrease. Usually, batch normalization are used in deep networks to improve convergence speed and accuracy, while dropout is used to prevent overfitting. The optimization doesn't work probably because of the shallowness of our network.

3.3. SNN

The result of SNN shows in Fig 4. Based on a tailored MLP of structure 784-30-50-10 with 3.6% top-1 error rate and 0.1% top-5 error rate, we achieve a SNN with around 28.8% top-1 error rate and 1.27% top-5 error rate. Though a significant performance decrease migrating MLP to SNN, we successfully proved that spiking rate could be used in Deep Network Models as a classifier.

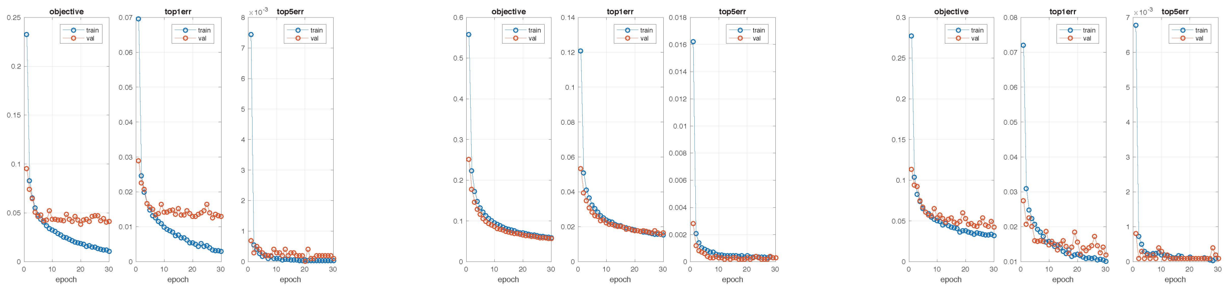

Figure 5. CNN Training Process of Non-Optimization(Left), Batch Normalization(Center) and Dropout(Right)

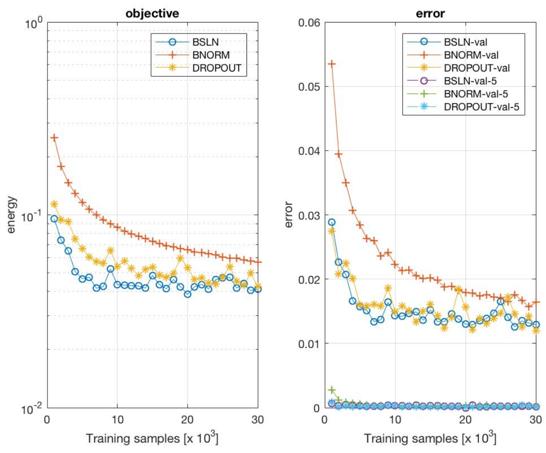

Figure 3. CNN Results Comparing Non-Optimization, Batch Normalization and Dropout

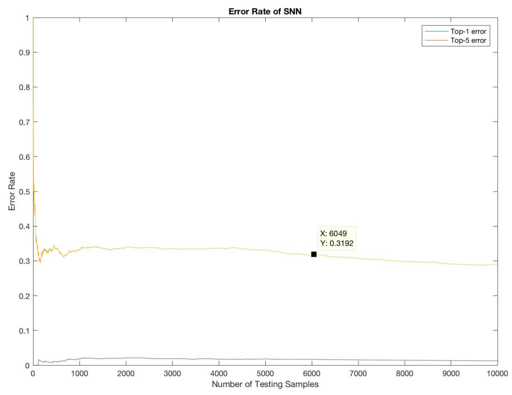

Figure 4. SNN Performance Over Test Set

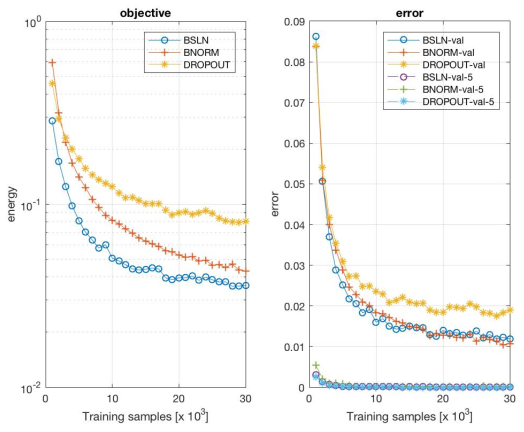

Figure 6. A Deeper CNN with 3 Convolution layer, 2 Pooling layer

4. Discuss

CNN has proved to be the best classifier comparing with MLP and SNN. Considering the batch normalization and dropout harmed performance of baseline, we showed the training epochs in Fig 5. We found that baseline was slightly overfitted, while by using normalization and dropout we reduced overfitting. We run another deeper CNN with 3 Convolution layer, 2 Pooling layer, the result show in Fig **??**fig:cnnlg], In this case, batch normalization performs better than previous case, but still, batch normalization and dropout harms the performance. Maybe we have to fine tuning the parameters and network design.