MATLAB Deep Learning Notes XI

Key Words : MNIST

1. MNIST Database

1. **MNIST** Database contains 70,000 images of handwritten numbers. In general, 60,000 images are used for training, and the remaining 10,000 images are used for the validation test. Each digit image is a 28-by-28 pixel black-and-white image.

2. In this note, we employ only 10,000 images with the training data and verification data in an 8:2 ratio. Therefore, we have 8,000 MNIST images for training and 2,000 images for validation of the performance of the neural network.

2. CNN Design and Implementation

3. The MNIST problem is caused by the multiclass classification of the 28-by-28 pixel image into one of the ten digit classes of 0 - 9. So the CNN can be designed as follows

Layer	Remark	Activation Function
Input	28×28 nodes	-
Convolution	20 convolution filters (9×9)	ReLU
Pooling	1 mean pooling (2×2)	-
Hidden	100 nodes	ReLU
Output	10 nodes	Softmax

Figure 1: the CNN that we designed to process MNIST.

4. Although it has many layers, only three of them contain the weight matrices that require training; they are W_1, W_5, and W_o in the square blocks. W_5 and W_o contain the connection weights of the classification neural network, while W_1 is the convolution layer's weight, which is used by the convolution filters for image processing.

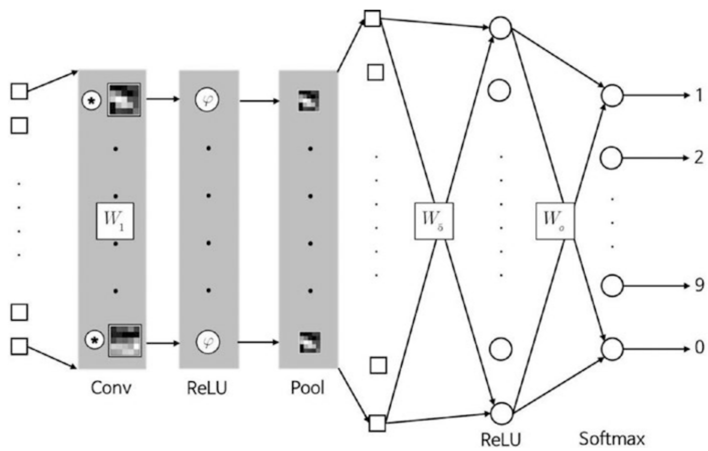

Figure 2: The architecture of this neural network.

5. In this CNN, we use mini-batch and momentum in order to get a better performance.

Program 26: MnistConv

Listing 1: MnistConv.m

```
function [W1, W5, Wo] = MnistConv(W1, W5, Wo, X, D)

alpha = 0.01;
beta = 0.95;

momentum1 = zeros(size(W1));
momentum5 = zeros(size(W5));
momentumo = zeros(size(Wo));

N = length(D);

bsize = 100;
blist = 1 : bsize : (N-bsize+1);

% one epoch loop
%
for batch = 1 : length(blist)
    dW1 = zeros(size(W1));
    dW5 = zeros(size(W5));
    dWo = zeros(size(Wo));
    % mini-batch loop
    %
    begin = blist(batch);
    for k = begin : begin + bsize - 1
        % forward pass = inference
        %
        x = X(:, :, k); % Input, 28 x 28
        y1 = conv(x, W1); % Convolution, 20x20x20
        y2 = ReLU(y1); %
        y3 = pool(y2); % Pool, 10x10x20
        y4 = reshape(y3, [], 1); % 2000
        v5 = W5 * y4; % ReLU, 360
        y5 = ReLU(v5);
        v = Wo * y5; % softmax, 10
        y = softmax(v); %

        % One-hot encoding
        %
        d = zeros(10, 1);
        d(sub2ind(size(d), D(k), 1)) = 1;
        % Backpropagation
        %
```

```
        e = d - y; % Output layer
        delta = e;

        e5 = Wo' * delta;  % Hidden(ReLU) layer
        delta5 = (y5 > 0) .* e5;

        e4 = W5' * delta5; % Pooling layer

        e3 = reshape(e4, size(y3));

        e2 = zeros(size(y2));
        W3 = ones(size(y2)) / (2 * 2);
        for c = 1 : 20
            e2(:, :, c) = kron(e3(:, :, c), ones([2 2])) .* W3(:, :, c);
        end

        delta2 = (y2 > 0) .* e2; % ReLU layer

        delta1_x = zeros(size(W1)); % Convolutional layer
        for c = 1 : 20
            delta1_x(:, :, c) = conv2(x(:, :), rot90(delta2(:, :, c), 2),...
            'valid');
        end

        dW1 = dW1 + delta1_x;
        dW5 = dW5 + delta5 * y4';
        dWo = dWo + delta * y5';

    end
    % Update weights
    dW1 = dW1 / bsize;
    dW5 = dW5 / bsize;
    dWo = dWo / bsize;

    momentum1 = alpha*dW1 + beta*momentum1;
    W1 = W1 + momentum1;

    momentum5 = alpha*dW5 + beta*momentum5;
    W5 = W5 + momentum5;

    momentumo = alpha*dWo + beta*momentumo;
    Wo = Wo + momentumo;
end
end
```

Program 27: testMnistConv

Listing 2: testMnistConv.m

```matlab
clear
Images = loadMNISTImages('t10k-images-idx3-ubyte');
Images = reshape(Images, 28, 28, [ ]);
Labels = loadMNISTLabels('t10k-labels-idx1-ubyte');
Labels(Labels == 0) = 10; % 0 --> 10
rng(1);
% Learning
%
W1 = 1e-2 * randn([9, 9, 20]);
W5 = (2 * rand(100, 2000) - 1) * sqrt(6) / sqrt(360 + 2000);
Wo = (2 * rand(10, 100) - 1) * sqrt(6) / sqrt(10 + 100);

X = Images(:, :, 1:8000);
D = Labels(1 : 8000);

for epoch = 1 : 3
    epoch
    [W1, W5, Wo] = MnistConv(W1, W5, Wo, X, D);
end

save('MnistConv.mat');

% Test
%
X = Images(:, :, 8001 : 10000);
D = Labels(8001 : 10000);

acc = 0;
N = length(D);
for k = 1:N
    x = X(:, :, k); % Input, 28x28

    y1 = conv(x, W1); % Convolution, 20x20x20
    y2 = ReLU(y1); %
    y3 = pool(y2); % Pool, 10x10x20
    y4 = reshape(y3, [ ], 1); % 2000
    v5 = W5 * y4; % ReLU, 360
    y5 = ReLU(v5); %
    v = Wo * y5; % Softmax, 10
    y = softmax(v); %

    [~, i] = max(y);
```

```
    if i == D(k)
        acc = acc + 1;
    end
end

acc = acc / N;
fprintf('Accuracy is %f\n', acc);
```

Output 27:

```
...

Accuracy is 0.936000
```