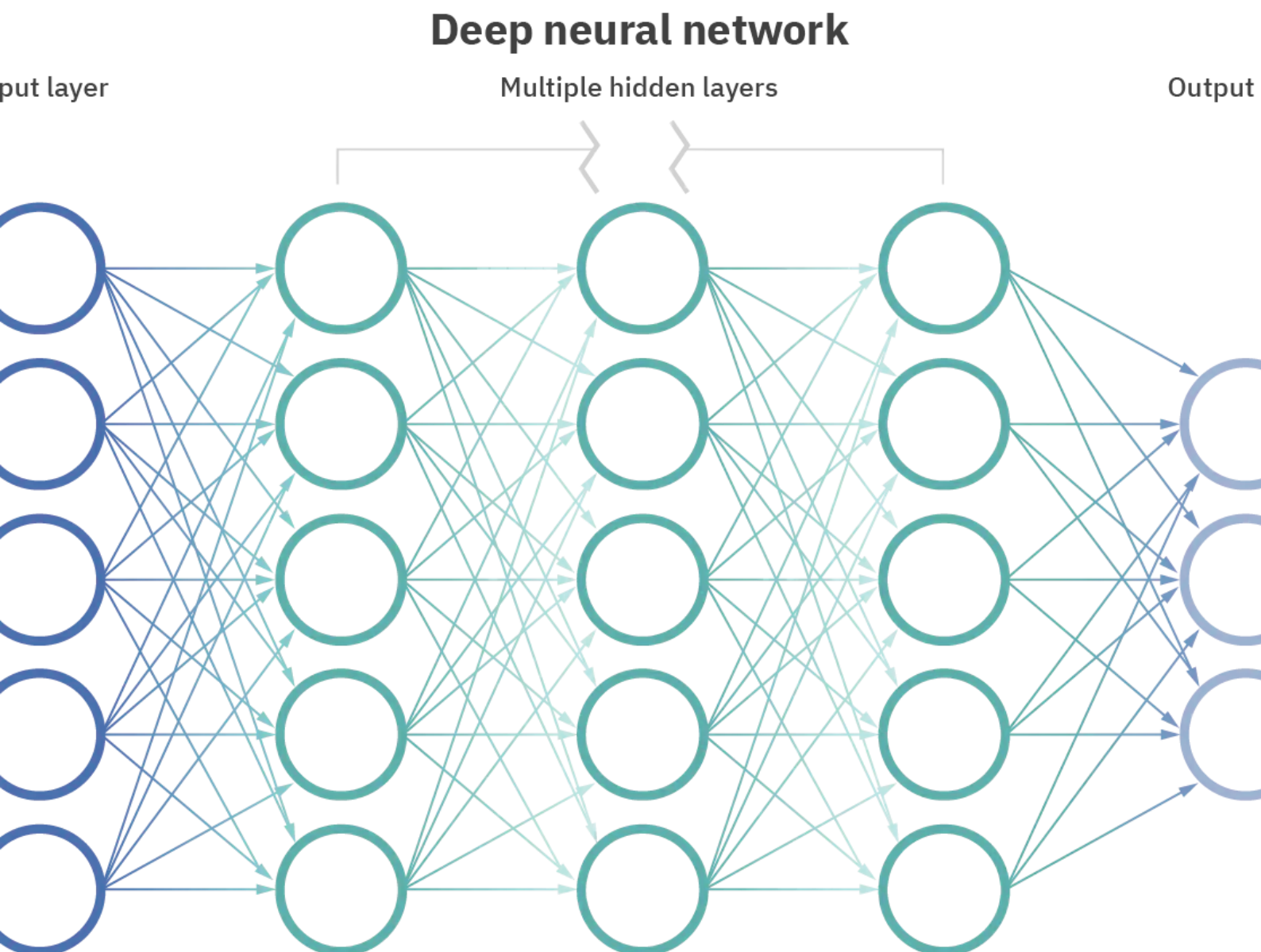


ASSIGNMENT-4

Deep Learning with Artificial Neural Network



Prepared by:
Bipin Kumar Singh (2K19/EE/077)



Submitted To:
Prof. Sudarshan Kumar Babu Valluru

QUES.

We implement a neural network that takes the input image and recognizes the digit that it represents. The training data is the MNIST9 database, which 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, as shown here.



Ans:

Considering the training time, this example employs 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.

The function `MnistConv`, which trains the network using the back-propagation algorithm, takes the neural network's weights and training data and returns the trained weights.

```
function [W1, W5, Wo] = MnistConv(W1, W5, Wo, X, D)
alpha = 0.01;
beta = 0.95;
momentum1 = zeros(size(W1));
momentum5 = zeros(size(W5));
momentum0 = 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, 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); %
% 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(:, :, c), 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

```

The following listing shows the function Conv, which the function MnistConv calls. This function takes the input image and the convolution filter matrix and returns the feature maps.

```

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, 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
    end
end

```

```

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(:, :, c), 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

```

The function *MnistConv* also calls the function *Pool*, which is implemented in the following listing . This function takes the feature map and returns the image after the 2'2 mean pooling process.

```

function y = Pool(x)
%
% 2x2 mean pooling
%
[xrow, xcol, numFilters] = size(x);
y = zeros(xrow/2, xcol/2, numFilters);
for k = 1:numFilters
filter = ones(2) / (2*2); % for mean
image = conv2(x(:, :, k), filter, 'valid');
y(:, :, k) = image(1:2:end, 1:2:end);
end
end

```

TestMnistConv.m file tests the function *MnistConv*. 10 This program calls the function *MnistConv* and trains the network three times. It provides the 2,000 test data points to the trained network and displays its accuracy. The test run of this example yielded an accuracy of 93% in 2 minutes and 30 seconds.

```

clear all
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
[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);

```

Followig is the output:

The screenshot shows the MATLAB environment with the following components:

- Editor:** Displays the script `PlotFeatures.m` with line numbers 1 to 21. The code includes initialization, training, and testing phases.
- Workspace:** A table listing variables and their values:

Name	Value
acc	0.9350
D	2000x1 double
epoch	3
i	6
Images	28x28x10000 double
k	2000
Labels	10000x1 double
N	2000
v	[-6.0778;4.7810;-5.859...
v5	100x1 double
W1	9x9x20 double
W5	100x2000 double
Wo	10x100 double
x	28x28 double
X	28x28x2000 double
y	[2.1497e-08;0.0011;2...
y1	20x20x20 double
y2	20x20x20 double
y3	10x10x20 double
y4	2000x1 double
y5	100x1 double
- Command Window:** Shows the command `>> TestMnistConv` and the output `Accuracy is 0.935000`.

The final result after passing the convolution and pooling layers is as many smaller images as the number of the convolution filters; ConvNet converts the input image into the many small feature maps.

Program for plotfeatures is:

```
clear all
load('MnistConv.mat')
k = 2;
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); %
figure;
display_network(x(:));
title('Input Image')
convFilters = zeros(9*9, 20);
for i = 1:20
    filter = W1(:, :, i);
    convFilters(:, i) = filter(:);
end
figure
display_network(convFilters);
title('Convolution Filters')
fList = zeros(20*20, 20);
for i = 1:20
    feature = y1(:, :, i);
    fList(:, i) = feature(:);
end
figure
display_network(fList);
title('Features [Convolution]')
fList = zeros(20*20, 20);
for i = 1:20
    feature = y2(:, :, i);
    fList(:, i) = feature(:);
end
figure
display_network(fList);
title('Features [Convolution + ReLU]')
fList = zeros(10*10, 20);
for i = 1:20
    feature = y3(:, :, i);
    fList(:, i) = feature(:);
end
figure
display_network(fList);
title('Features [Convolution + ReLU + MeanPool]')
```

Now the outputs are:

