TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

BÁO CÁO BÀI TẬP LỚN

Đề tài: Sử dụng mạng nơ rôn CNN để nhận dạng số viết tay

Nhóm sinh viên:

Họ và tên	MSSV
Nguyễn Hữu Thuấn	20144353
Trần Công Minh	20142964

Giảng viên hướng dẫn: TS. Trần Thị Thanh Hải

Hà Nội, 6/2018

Mục Lục

1. Tóm Tắt 3
2. Dữ Liệu 4
3. Thiết Kế Phần Mềm 5
4. Kết quả 10
4.1 Kết quả huấn luyện mạng với Mini batch và cấu trúc mạng khác nhau 10
4.2 Cấu trúc mạng được lưu sau khi học và cách sử dụng 27
5. Kết Luận 28
6. Tài Liệu Tham Khảo 29

1. Tóm Tắt

Trong bài tập lớn này, nhóm sử dụng mô hình mạng noron từ bài báo[1] nói về mạng nơ rôn CNN để nhận dạng số viết tay

Nhờ các công thức đã được tính toán trong bài báo, nhóm xây dựng 1 mạng noron trên phần mềm matlab. Các giải thuật được xây được dựa trên các công thức được nêu trong bài báo.

Trong phần báo cáo cũng để cập đến sử dụng công thức bao nhiêu trong bài báo

Quy ước viết tắt: CT : Công thức

Tóm tắt chức năng các File trong mục code (chỉ nêu các file quan trọng)

Tên	Chức năng
mnist_uint8.mat	Chứa dữ liệu Train và Test
Net_Saved.mat	Chứa mạng noron đã được học
Train_Test.m	File code khởi tạo mạng, train và test tất cả dữ liệu
runNN.m	Hàm chạy ra kết quả khi cho dữ liệu vào và vẽ hình ảnh đầu vào
cnnConfigParameters.m	Cài đặt các thông số học của mạng
cnnConfigLayer.m	Cài đặt các lớp của mạng
cnnTrain.m	Train mạng
cnnTest.m	Test lại mạng với bộ số liệu test và train, và kiểm tra độ chính xác của mạng

2. Dữ Liệu

Dữ liệu em sử dụng là **bộ dữ liệu MNIST**, gồm có 60000 data train, 10000 data test để kiểm nghiệm lại mô hình mạng. Trong phần code này, dữ liệu đã được lưu dưới file.mat(mnist_uint8.mat) của matlab để tiện xử lý.

Dữ liệu vào(test_x, train_x) là ma trận 1x784 (ma trân 784 hàng) nên phải chuyển vị, và đầu vào của mạng là ma trận 28x28 nên trước khi xử dụng dữ liệu ta phải chuyển về ma trận 28x28

Dữ liệu ra (test_y, train_y) là ma trận 1x10, nên phải chuyển vị, đầu ra của mạng sử dụng phương pháp One hot coding, mạng có 10 đầu ra(tương ứng với nhãn 0 đến 9). Khi mạng dự đoán đầu ra nào có giá trị lớn nhất thì mạng dự đoán được nhãn tương ứng

3. Thiết Kế Phần Mềm

Trong phần thiết kế phần mềm được chia làm các phần chính như sau:

- Khởi tạo: khởi tạo số lớp ẩn, số noron, các tham số học, giá trị ban đầu các tham số của mạng
- Feed forward: Tính toán đầu ra của mạng khi cho đầu vào
- Back propagation: Sử dụng thuật toán lan truyền ngược đển tính các thông số của noron, sử dụng nó để cập nhật lại thông số
- Parameter update: Cập nhật tham số của mạng
- Tính sai số,vẽ đồ thị

Nhóm sử dụng hàm **convn** trong matlab dùng để tính tích chập

a) Khởi tạo

Cài đặt các lớp của mạng

Nằm trong hàm cnnConfigLayer(), có thể thay đổi các tham số phù hợp với mục đích

```
cnnConfig.layer{1}.type = 'input';

cnnConfig.layer{2}.type = 'conv';
cnnConfig.layer{2}.outputmaps = 6; % so noron lop ay
cnnConfig.layer{2}.kernelsize = 5;

cnnConfig.layer{3}.type = 'pool';
cnnConfig.layer{3}.scale = 2;

cnnConfig.layer{4}.type = 'conv';
cnnConfig.layer{4}.outputmaps = 12; % so noron lop ay
cnnConfig.layer{4}.kernelsize = 5;

cnnConfig.layer{5}.type = 'pool';
cnnConfig.layer{5}.scale = 2;
```

Khởi tạo các tham số học

Gồm các tham số: hệ số học, chạy số mẫu để cập nhật tham số (Minibatch), số lần duyệt dữ liệu (epochs)

```
opts.alpha = 1;
opts.batchsize = 50;
opts.numepochs = 10;
```

Khởi tạo các tham số ban đầu của mạng (CT 1,2,3)

Lần lượt duyệt, rồi khởi tạo các giá trị ngẫu nhiêu, đủ nhỏ (trong hàm cnnInit()).

b) Feed forward

Xử lý lớp Convolution (CT 5,8)

Tính đầu ra của từng noron: ở đây input_map là số noron đầu vào, `k{i}{j},`được hiểu là **Convolution Filter,** sau khi tính tổng ra 1 biến tạm, ta sử dụng hàm logsig để tính được giá trị

Ta chạy lần lượt các đầu ra của lớp ta được output của lớp

```
        for i = 1: input_map
           tem = tem + convn(net.layer{1}.a{i},net.layer{la}.k{i}{j},'valid');
        end
        net.layer{la}.a{j} = logsig(tem + net.layer{la}.b{j});
```

Xử lý lớp Pooling (CT 6,9)

Lớp này dùng để giảm tham số, từ đó giúp giảm thời gian tính toán, `net.layer{la}.scale ^ 2` được khởi tạo trong phần cài đặt lớp (trong trường hợp đang xét là 2)

```
for j = 1: input_map
    tem =  convn(net.layer{la 1}.a{j},
ones(net.layer{la}.scale)/(net.layer{la}.scale ^ 2),'valid');
    net.layer{la}.a{j}  =tem(1 : net.layer{la}.scale : end, 1 :
net.layer{la}.scale : end, :); % gia tri
end
```

Fully Connected

Giúp chuyển 2D về 1D (hoặc 3D về 2D trong trường hợp Minibatch > 1)

```
tem = [];
 for i = 1:input_map
     tem_size = size(net.layer{n}.a{i});
     tem = [tem;
reshape(net.layer{n}.a{i},tem_size(1)*tem_size(2),tem_size(3))];

 end
```

c) Back propagation

Sử dụng hàm lỗi: (CT 13)

$$L = \frac{1}{2}\sum_{i=1}^{10}(\hat{y}(i) - y(i))^2$$

Hình 1: Hàm lỗi

Tính dentaB và dentaW ở lớp cuối

`net.dfcb` là dentaB ở lớp cuối, sử dụng công thức 23,24, vì xét có trường hợp Minibatch nên lấy trung bình tổng số mẫu xét

```
% sai e: output - trueLabe
net.e = net.o - y;
%tinh loi: Loss Function
net.L = 1/2*sum((net.e(:)).^2) /size(net.e,2); % net.e,2 : so mau

net.fcb_tem = (net.e).*net.o.*(1-net.o); % ct 23,24
net.dfcb = mean(net.fcb_tem,2);
```

Tương tự với dentaW(CT19), `tem_size(3)` là số mẫu (vì sử dụng size đầu vào, nên đây là mảng 3 chiều, chiều thứ 3 là số mẫu đang xét) `net.fj` là output của mạng tính được(chưa qua transfer function)

```
net.dfcw = (net.fcb_tem * (net.fj)') ./ tem_size(3); % ct 19,
```

<u>Tính dentaK, dentaB các lớp ẩn</u>

Đầu tiên phải chuyển vecto lỗi (192x1) về 12 ma trận 4x4 (lưu ý: trong trường hợp này có Minibatch nên chiều của mảng được tăng lên 1, chiều thứ 3)

```
for j = 1: numel(net.layer{n}.a) % trong truong hop nay la 12
    net.layer{n}.dentaS{j} = [];
    tem = [];
    for i = 1: (tem_size(1) * tem_size(2))
        tem = [tem ; net.dentaF(count, 1:tem_size(3))];
        count = count+1;
    end
    net.layer{n}.dentaS{j} = reshape(tem(1:
size(tem,1),:),tem_size(1),tem_size(2),tem_size(3));
end
```

Theo công thức 37 tính dentaC, cnnUpsampling() là tính theo công thức 32

```
for j = 1: numel(net.layer{la}.a) % duyet het ouput
    % cong thuc 32: su dung lay phan nguyen tren: tuc la se bi nap lai 4x4 ->
8x8
    net.layer{la}.dentaC{j} = cnnUpsampling(net.layer{la + 1}.dentaS{j},
net.layer{la +1}.scale) ./ (net.layer{la +1}.scale^2) ;
    net.layer{la}.dentaC_teta{j} = net.layer{la}.dentaC{j} .*
net.layer{la}.a{j} .* ( 1 - net.layer{la}.a{j}); % ct 37

end
```

Tính dentaK, dentaB theo công thức 40,45,59,64, ở lớp sau lớp Convolution, hàm flipPlus()(trong file code): gần giống rot90(~,2) (trong matlab) ~ rot180 nhưng thêm chức năng cả chiều thứ 3, phù hợp với Minibatch. Các thông số tính được chia cho số mẫu Minibatch

```
for j = 1: numel(net.layer{la + 1}.a)
    for i = 1 : numel(net.layer{la}.a)
        net.layer{la+1}.dentaK{i}{j} = convn(flipPlus(net.layer{la}.a{i}),
net.layer{la+1}.dentaC_teta{j}, 'valid') ./ tem_size(3); % rot90(A,2), which
rotates by 180 degrees.
    end
    net.layer{la+1}.dentaB{j} = sum(net.layer{la+1}.dentaC_teta{j}(:)) ./
tem_size(3) ; % chia cho tong so mau
end
```

Từ lớp Convolution thứ 2 (tính từ output đến input) ta tính ΔS như sau (CT 51)

```
for i = 1 : numel(net.layer{la}.a)
    tem = zeros(size(net.layer{la}.a{1}));
    for j = 1: numel(net.layer{la + 1}.a)
        tem = tem + convn(net.layer{la + 1}.dentaC_teta{j},rot90(net.layer{la +
1}.k{i}{j},2)); % ct 51
        % rot 90 no khong thanh phan thu 3(:,:,x)
    end
    net.layer{la}.dentaS{i} = tem;
end
```

d) Parameter update

Duyệt tất cả các lớp Convolution và thay đổi dentaK, dentaB

```
for la = 2 : numel(net.layer)
    if strcmp(net.layer{la}.type, 'conv')
        for j = 1: numel(net.layer{la}.a)
            for i = 1: numel(net.layer{la - 1 }.a)
                net.layer{la}.k{i}{j} = net.layer{la}.k{i}{j} - opts.alpha *
net.layer{la}.dentaK{i}{j};
            end
            net.layer{la}.b{j} = net.layer{la}.b{j} - opts.alpha *
net.layer{la}.dentaB{j};
        end
    end
end
net.fcw = net.fcw - opts.alpha * net.dfcw;
net.fcb = net.fcb - opts.alpha * net.dfcb;
```

e) Tính sai số,vẽ đồ thị

Tính sai số

Sau khi mạng đã "học" xong, cho dữ liệu đầu vào và sử dụng mạng tính toán so sánh nó với dữ liệu ra(dữ liệu đúng)

Trong phần code còn xét đến sai số từng trường hợp tất cả nhãn (0 đến 9), `size(y,2)` là tổng số mẫu

```
err_band = zeros(1,10);
err = 0;
tem = cnnFeedForward(net,x);
[max1,duDoan] = max(tem.o);
[max2,thuc] = max(y);% vd: thuc 1 : la so 0

for i = 1: size(y,2)
    if duDoan(i) ~= thuc(i)
       err_band(thuc(i))  = err_band(thuc(i)) + 1;
       err = err+1;
    end
end

for i = 1 :10
    err_band(i) = round(err_band(i) / size(find(thuc == i),2),5);
end
err_rate = err/size(y,2);
```

Vẽ đồ thị

Trong phần này nhóm vẽ đồ thị :

- Trong lúc train, hiển thị sai số lúc học, số epochs đã chạy được, sai số ..

```
t = ['Epoch: ', num2str(i), '--Numbatches(of Epoch): ' num2str(j), '--Error: '
,num2str(round(net.L, 4)), '--Time: ',num2str(round(toc,2)),'s'];
title(t);
plot([last1(1) count] , [last1(2) net.L],'-*r');
hold on
xlabel('Numbatches')
ylabel('Error')
last1 = [count net.L];
```

- Lúc mạng “học” xong vẽ đồ thị cột sai số từng nhãn của dữ liệu test và train

```
tem = [];
for i = 1:10
    tem = [tem;100 - err_band(i)*100,err_band(i)*100];

end
bar(0:9,tem,'stacked')
ylim([0 105]);
xlim([-0.5 10]);
for i = 1: 10
    str = sprintf('%.2f', 100 - err_band(i)*100);
    text(i-1 -0.3 ,50,str,'fontsize', 8);
end
legend('Correct','Error')
title(['Sai so tung truong hop voi Data Testing']);
```

4. Kết quả

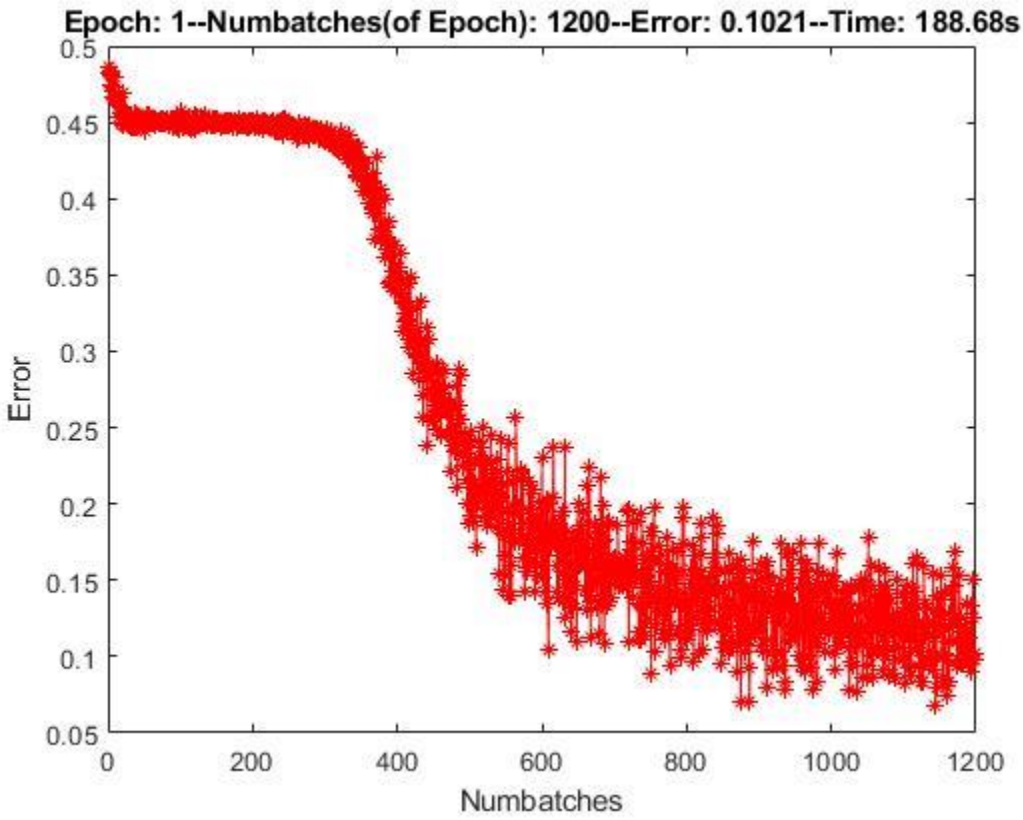
4.1 Kết quả huấn luyện mạng với Mini batch và cấu trúc mạng khác nhau

Numbatches là lần duyệt Minibatch dữ liệu

Các trường hợp đầu Mini batch = 50, trường hợp cuối Mini batch = 100

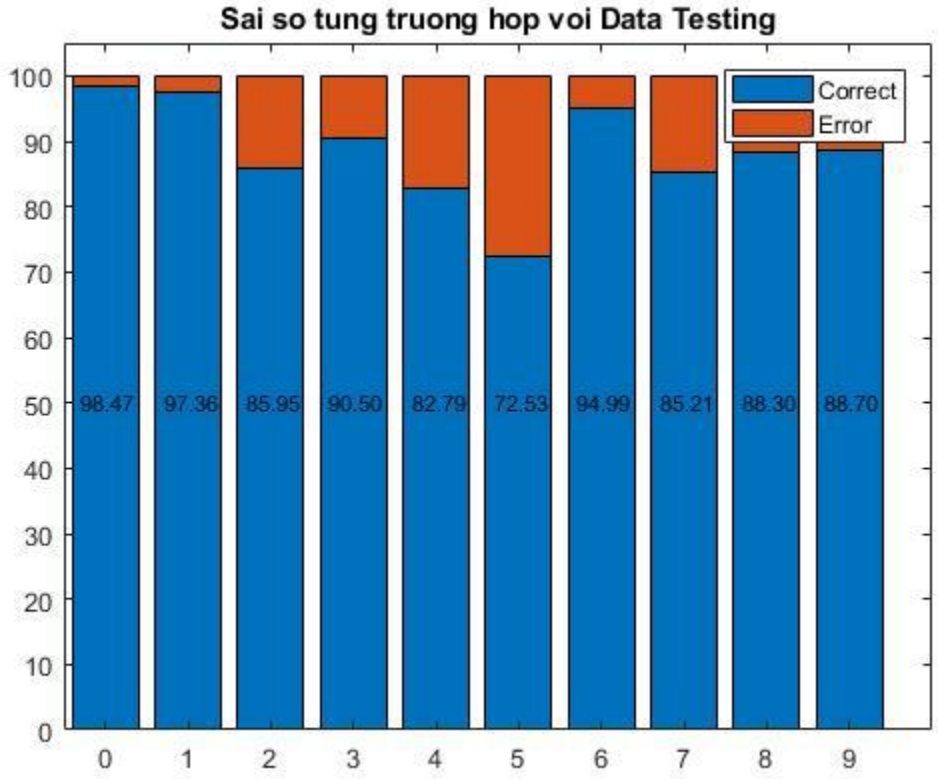
a) 1 Epoch

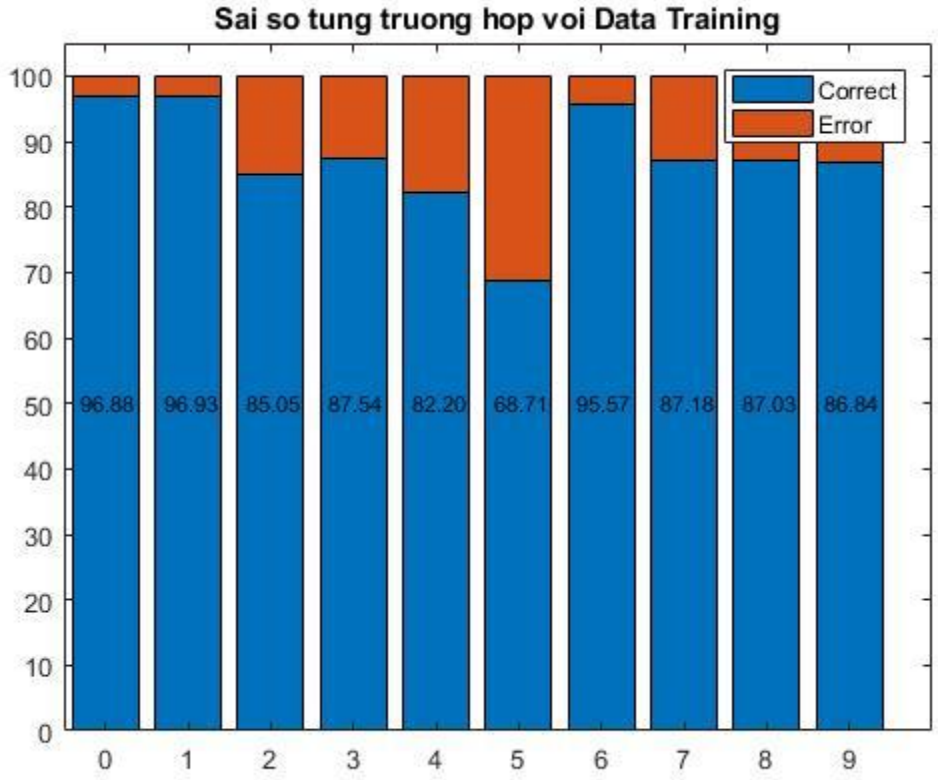
Trong quá trình học với 1 Epoch hệ số học: 1



Ta thấy bắt đầu từ Numbatches thứ khoảng 300 sai số bắt đầu giảm, nhưng tới khoảng 650 sai số giảm chậm lại rất đáng kể.

Sai số theo từng nhãn


Sai so tung truong hop voi Data Testing
Correct
Error
100
90
80
70
60
50
40
30
20
10
0
98.47
97.36
85.95
90.50
82.79
72.53
94.99
85.21
88.30
88.70
0
1
2
3
4
5
6
7
8
9

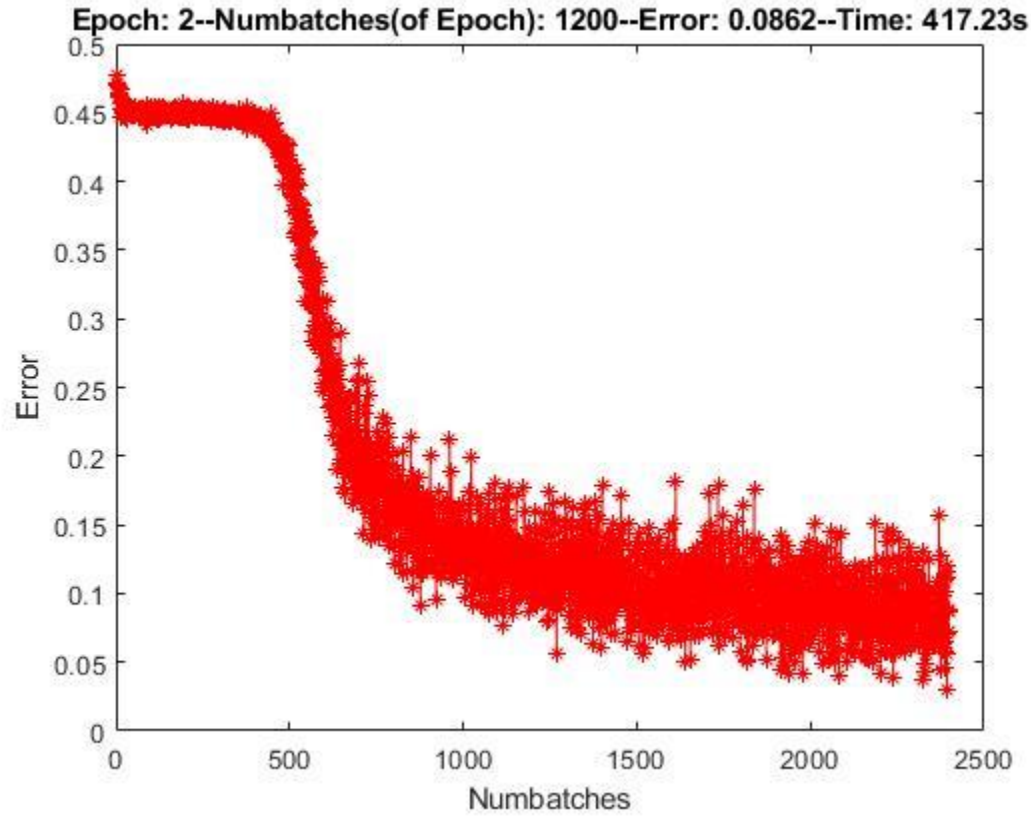

Sai so tung truong hop voi Data Training
Correct
Error
100
90
80
70
60
50
40
30
20
10
0
96.88
96.93
85.05
87.54
82.20
68.71
95.57
87.18
87.03
86.84
0
1
2
3
4
5
6
7
8
9

Sai số trên toàn dữ liệu:

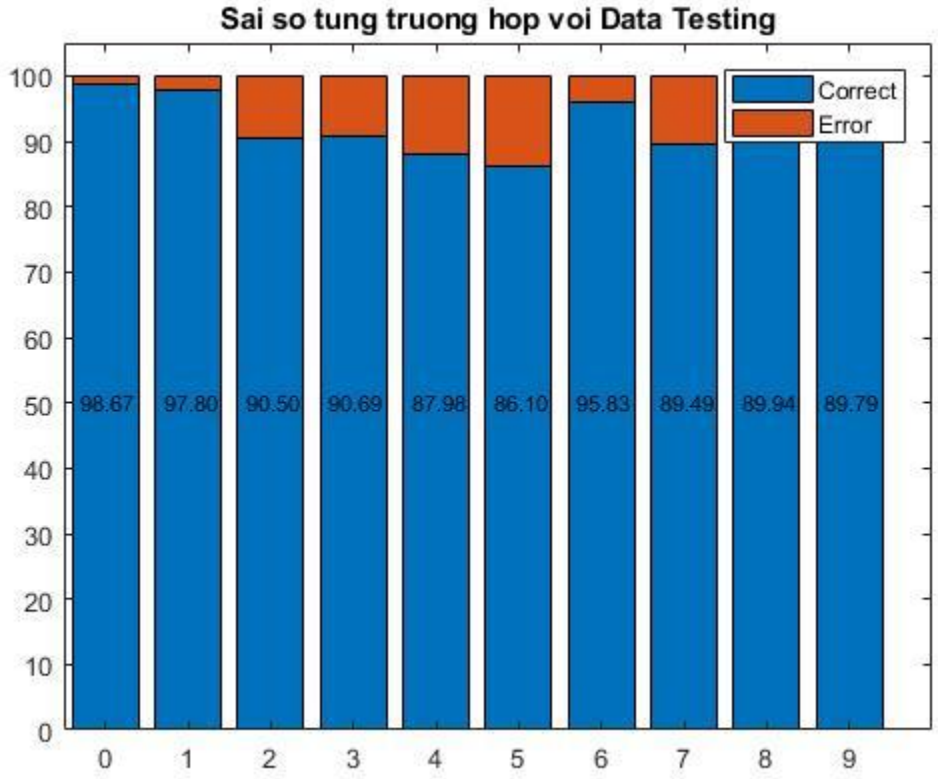
- Data Test: 11.28 %
- Data Train: 12.32%

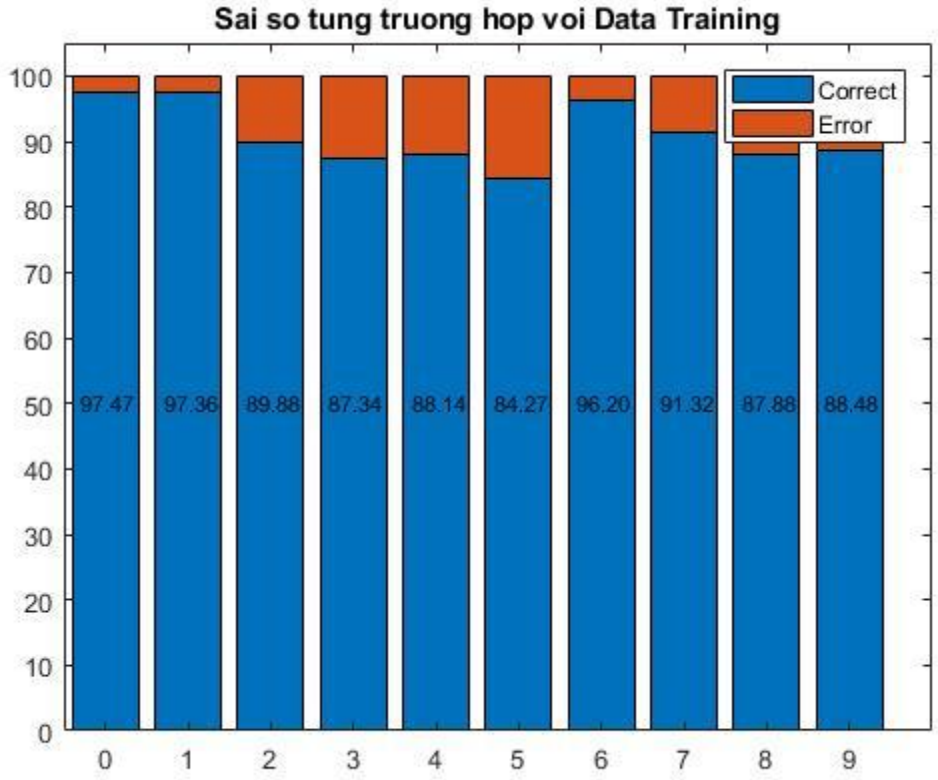
b) 2 Epoch

Trong quá trình học với 2 Epoch hệ số học: 1



Sai số theo từng nhãn


Sai so tung truong hop voi Data Testing
Correct
Error
100
90
80
70
60
50
40
30
20
10
0
98.67
97.80
90.50
90.69
87.98
86.10
95.83
89.49
89.94
89.79
0
1
2
3
4
5
6
7
8
9

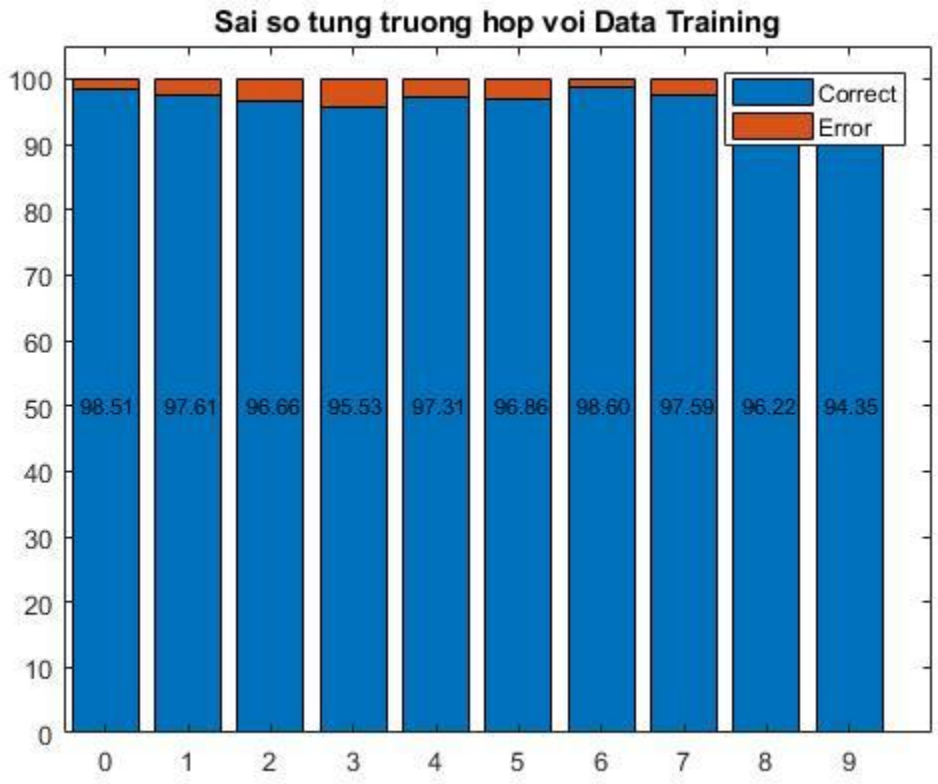

Sai so tung truong hop voi Data Training
Correct
Error
100
90
80
70
60
50
40
30
20
10
0
97.47
97.36
89.88
87.34
88.14
84.27
96.20
91.32
87.88
88.48
0
1
2
3
4
5
6
7
8
9

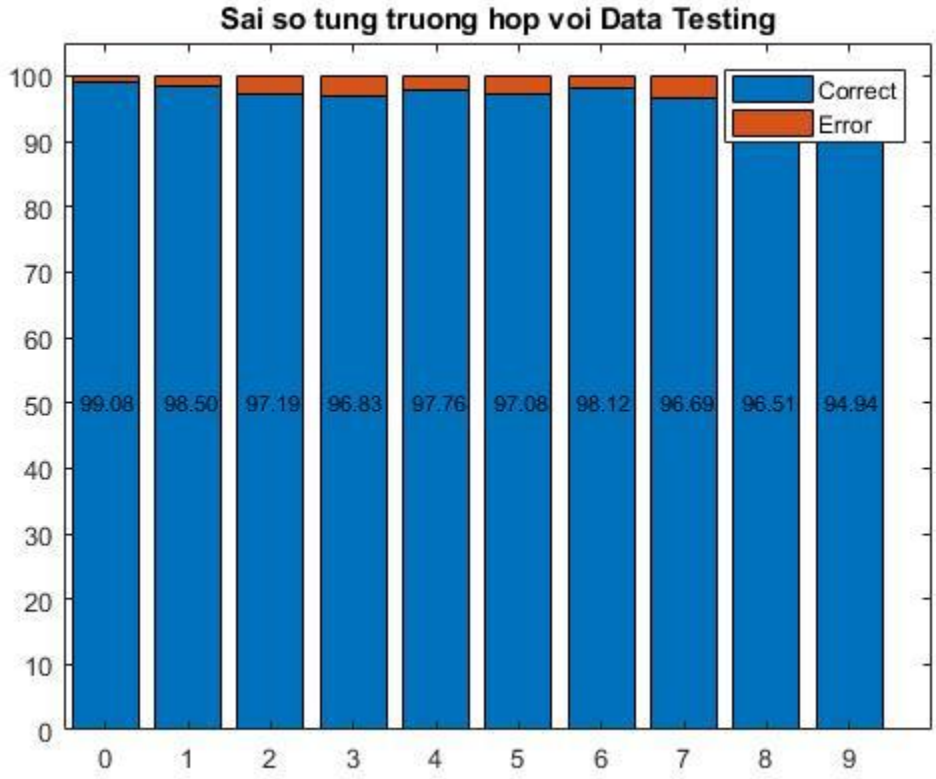
Sai số trên toàn dữ liệu:

- Data Test: 8.21 %
- Data Train: 9.027%

c) 10 Epoch

Trong quá trình học với 10 Epoch hệ số học: 1



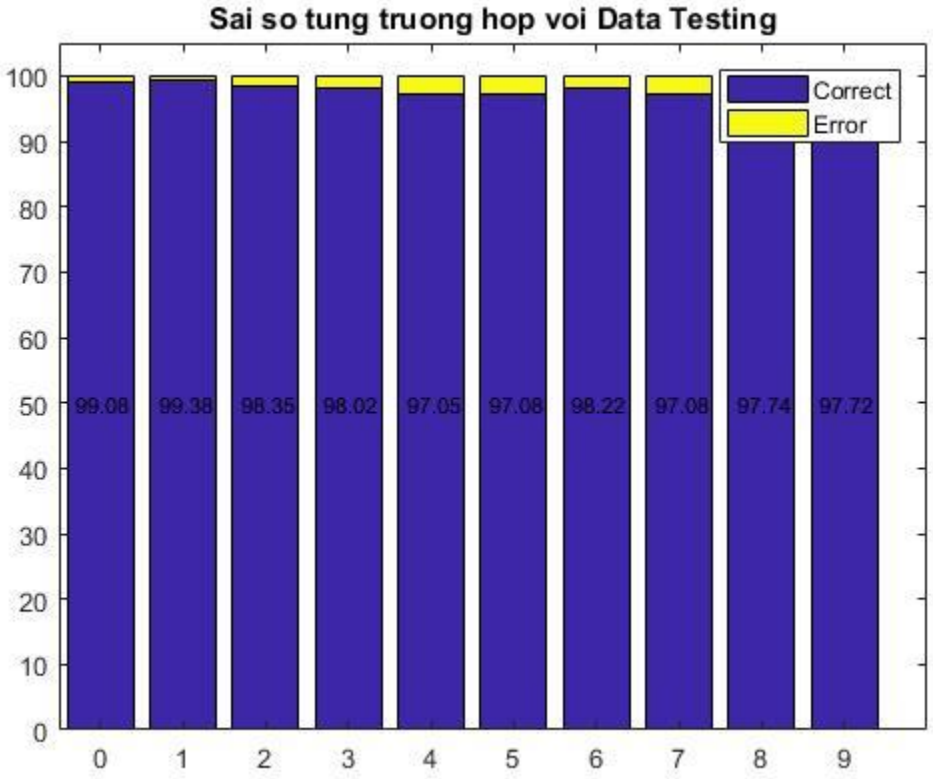


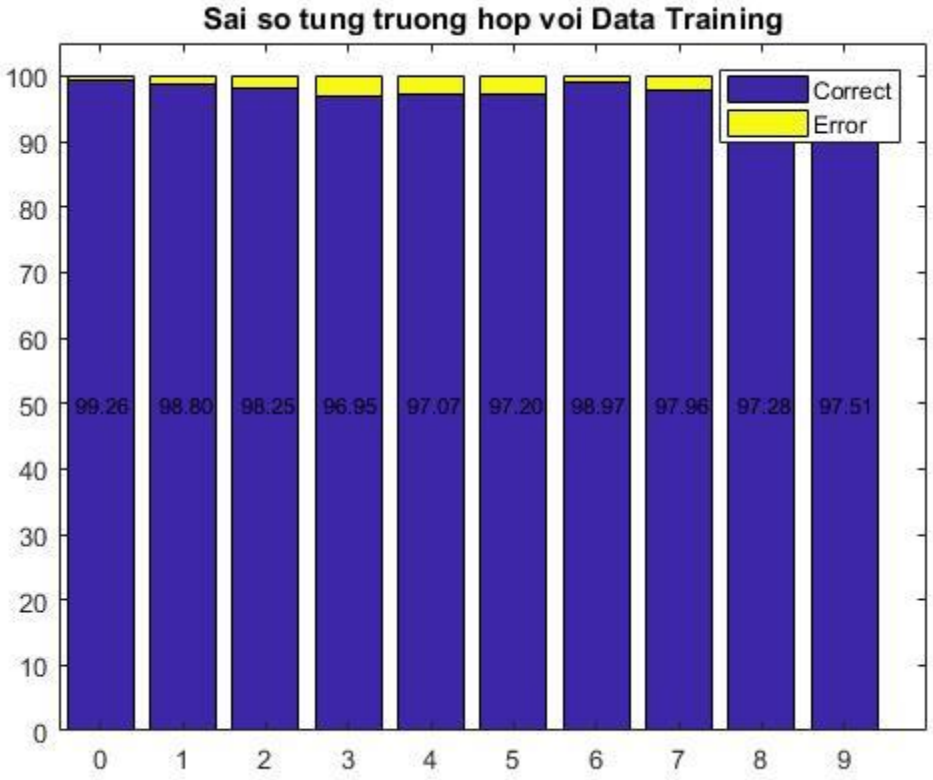
Sai số trên toàn dữ liệu:

- Data Test: 2.72 %
- Data Train: 3.07%

d) 20 Epoch

Trong quá trình học với 20 Epoch hệ số học: 1


Sai so tung truong hop voi Data Testing
Correct
Error
100
90
80
70
60
50
40
30
20
10
0
99.08
99.38
98.35
98.02
97.05
97.08
98.22
97.08
97.74
97.72
0
1
2
3
4
5
6
7
8
9

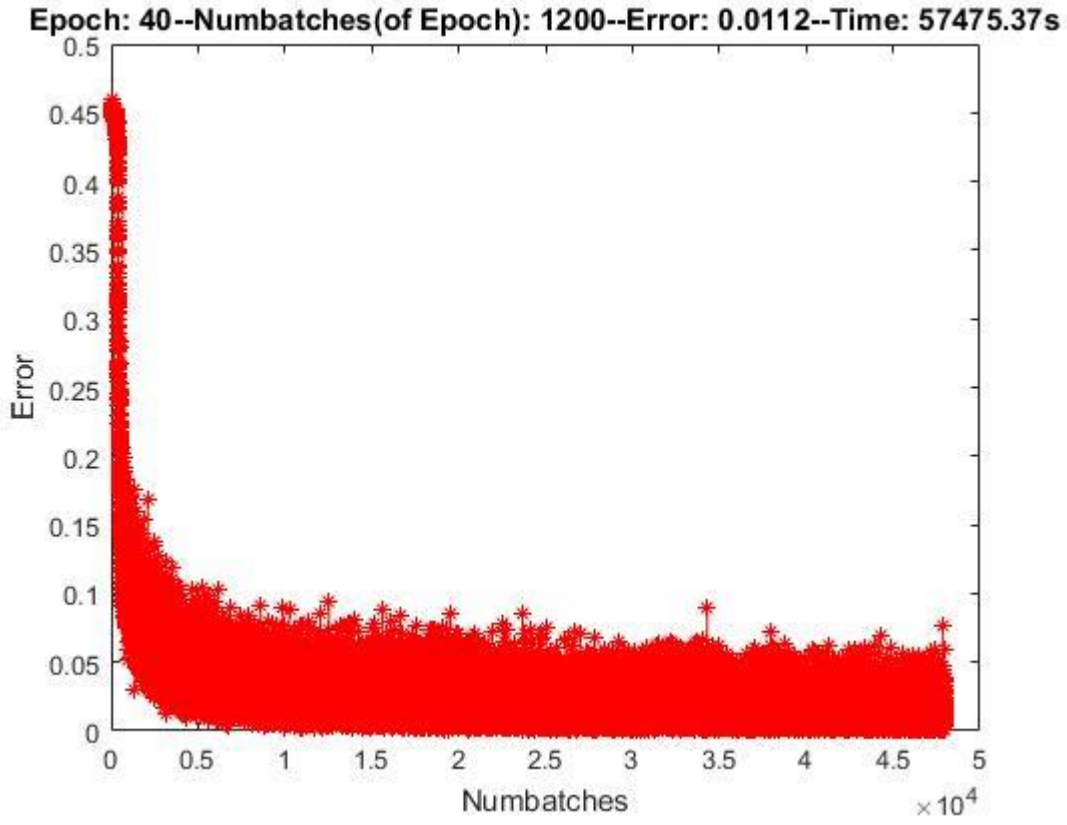

Sai so tung truong hop voi Data Training
Correct
Error
100
90
80
70
60
50
40
30
20
10
0
99.26
98.80
98.25
96.95
97.07
97.20
98.97
97.96
97.28
97.51
0
1
2
3
4
5
6
7
8
9

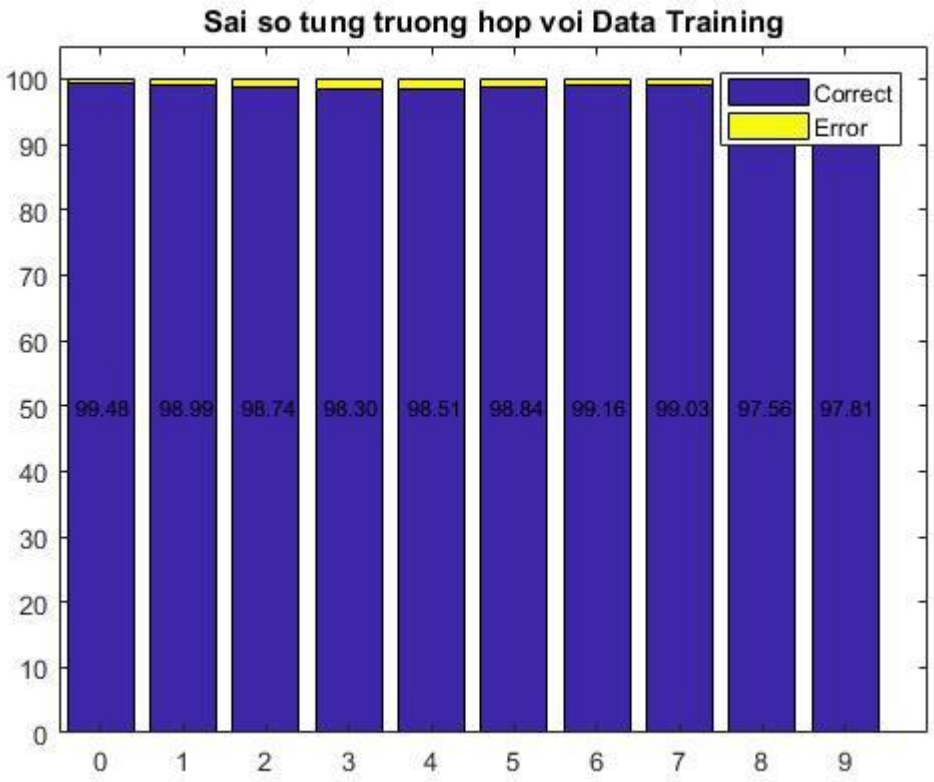
Sai số trên toàn dữ liệu:

- Data Test: 2 %
- Data Train: 2.058%

e) 40 Epoch

Trong quá trình học với 40 Epoch hệ số học: Từ 1: 1.4, từ 10: 1.2, từ 20: 1, từ 30: 0.8





Sai số trên toàn dữ liệu:

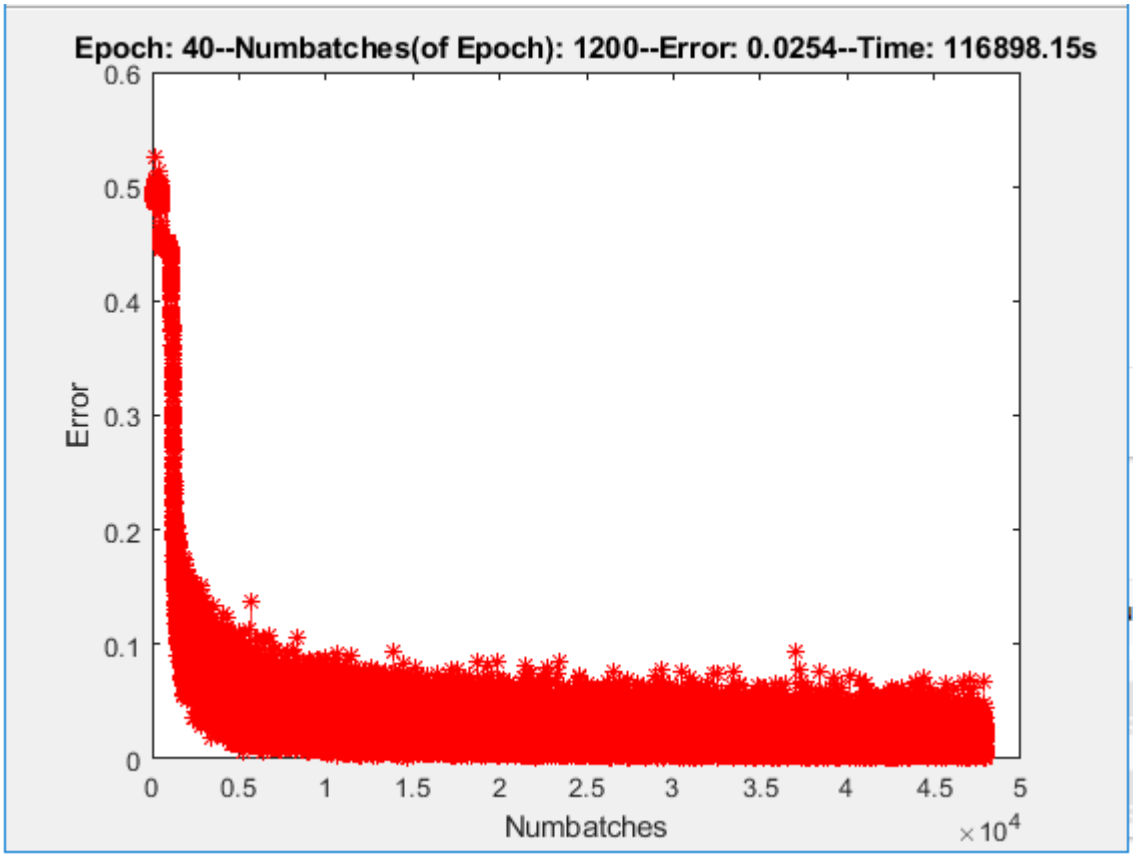
- Data Test: 1.17 %
- Data Train: 1.353%

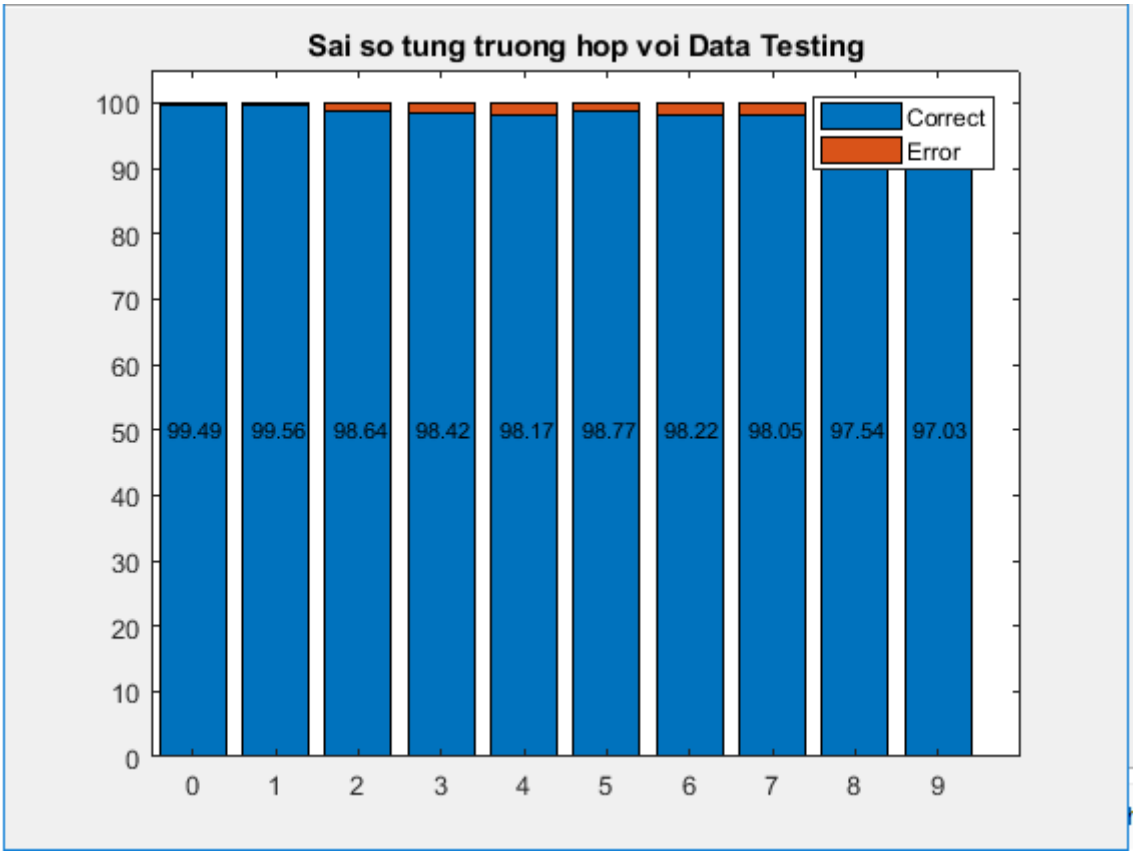
f) 40 Epoch

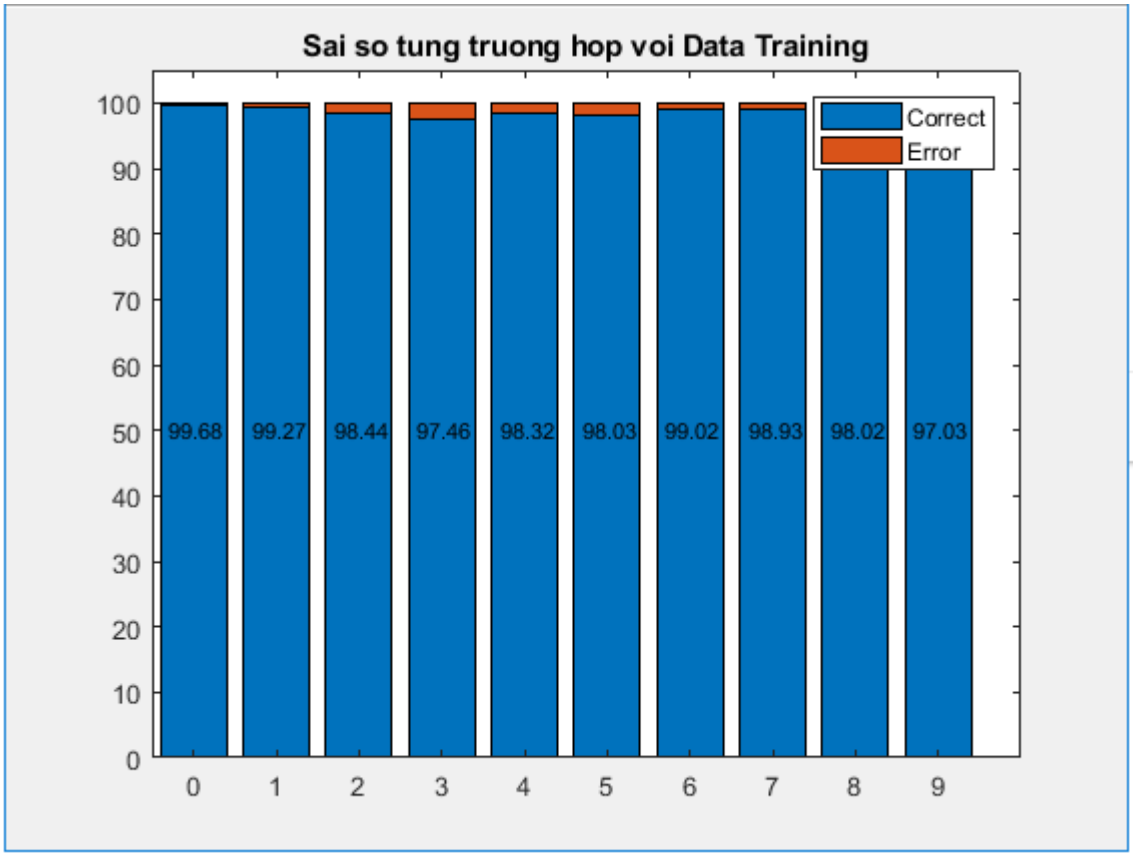
Thay đổi thông số của các lớp: (trong phần code đã comment)

```
cnnConfig.layer{2}.type = 'conv';
cnnConfig.layer{2}.outputmaps = 6; % so noron lop ay
cnnConfig.layer{2}.kernelsize = 5;

cnnConfig.layer{3}.type = 'pool';
cnnConfig.layer{3}.scale = 2;

cnnConfig.layer{4}.type = 'conv';
cnnConfig.layer{4}.outputmaps = 12; % so noron lop ay
cnnConfig.layer{4}.kernelsize = 5;

cnnConfig.layer{5}.type = 'pool';
cnnConfig.layer{5}.scale = 2;
```

Trong quá trình học với 40 Epoch hệ số học: Từ 1: 1.4, từ 10: 1.2, từ 20: 1, từ 30: 0.8


Epoch: 40--Numbatches(of Epoch): 1200--Error: 0.0254--Time: 116898.15s
Error
Numbatches
×10^4


Sai so tung truong hop voi Data Testing
Correct
Error
99.49
99.56
98.64
98.42
98.17
98.77
98.22
98.05
97.54
97.03

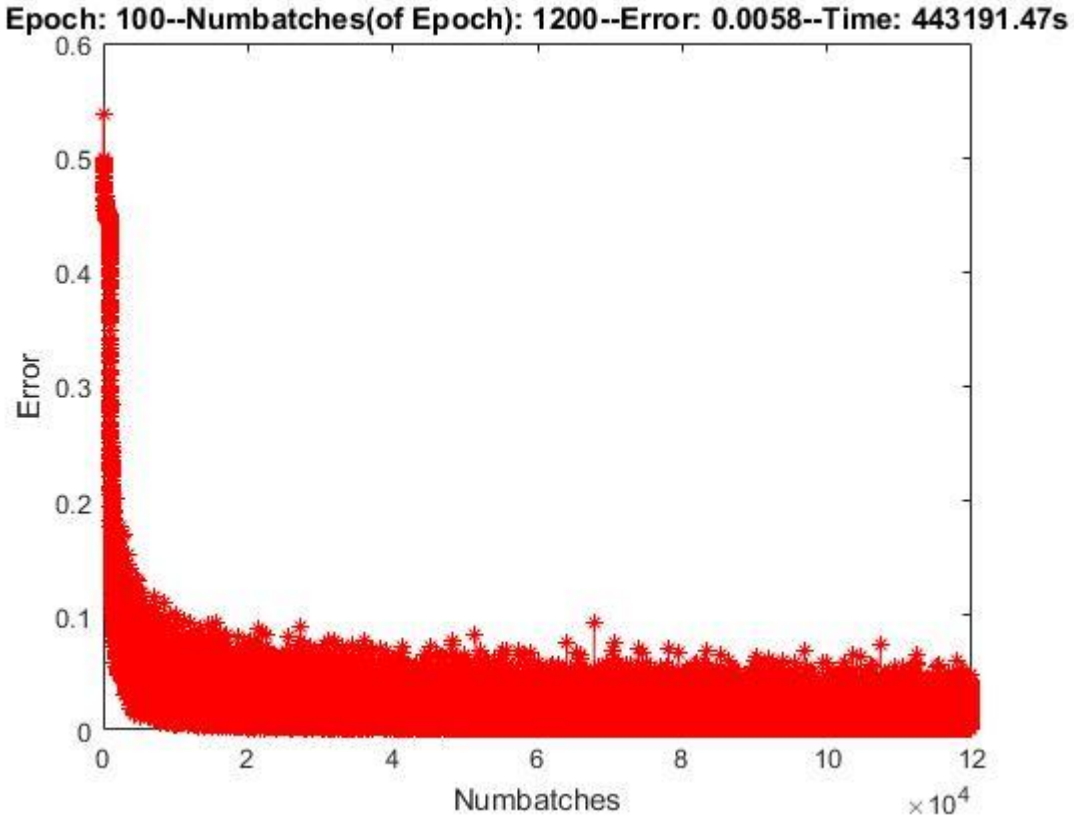


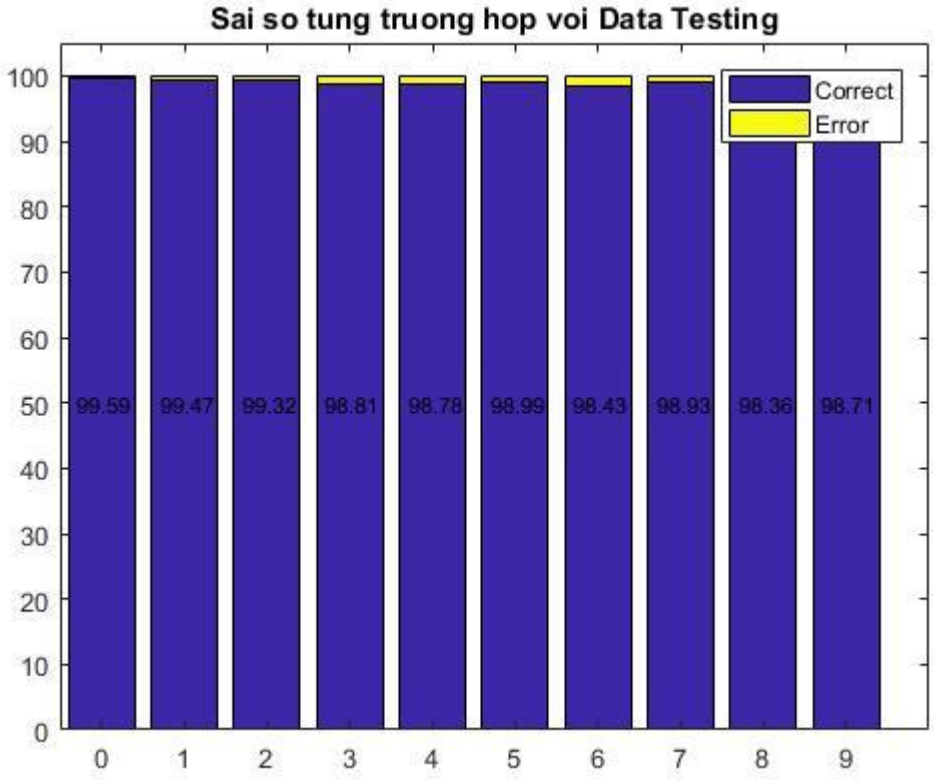
Sai số trên toàn dữ liệu:

- Data Test: 1.16 %
- Data Train: 1.567%

g) 100 Epoch

Trong quá trình học với 100 Epoch hệ số học: Từ 1: 1.4, từ 30: 1.2, từ 60: 1, từ 80: 0.8


Epoch: 100--Numbatches(of Epoch): 1200--Error: 0.0058--Time: 443191.47s
Error
Numbatches
×10^4


Sai so tung truong hop voi Data Testing
Correct
Error
99.59
99.47
99.32
98.81
98.78
98.99
98.43
98.93
98.36
98.71

Sai số trên toàn dữ liệu:

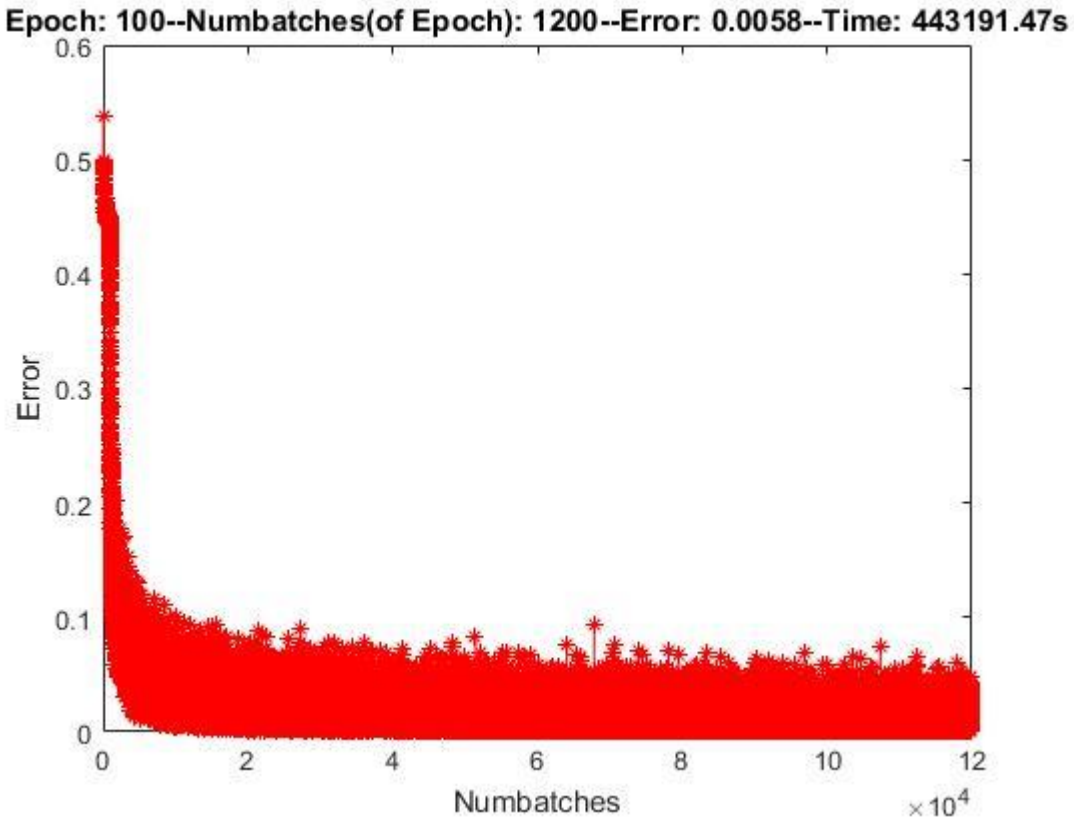
- Data Test: 1.05 %
- Data Train: 0.9%

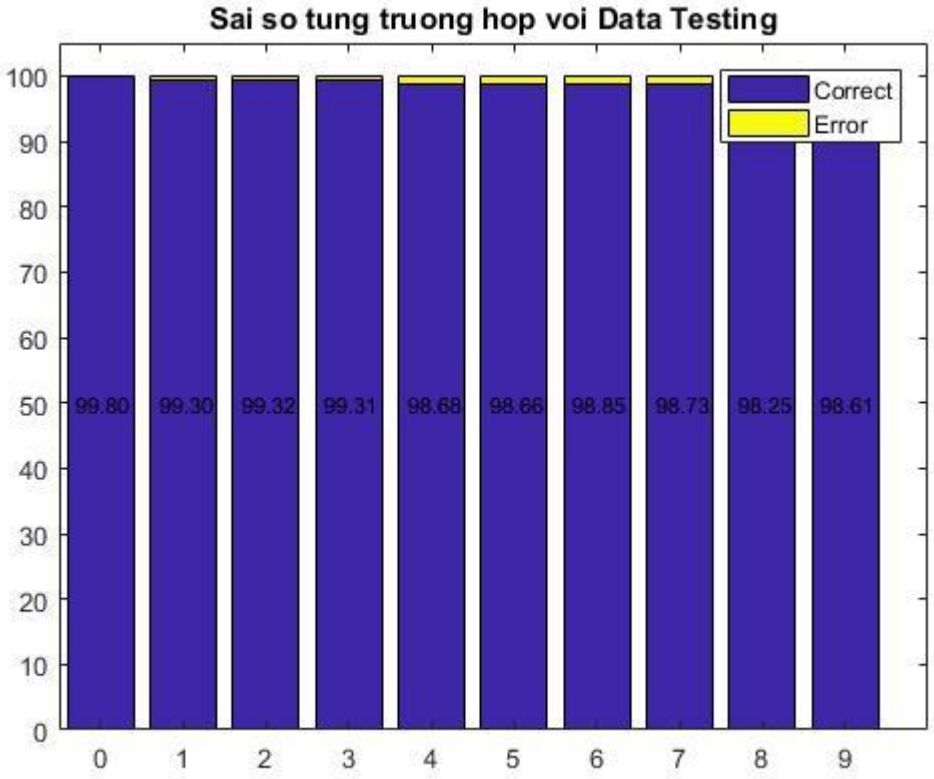
h) 100 Epoch (trường hợp 2)

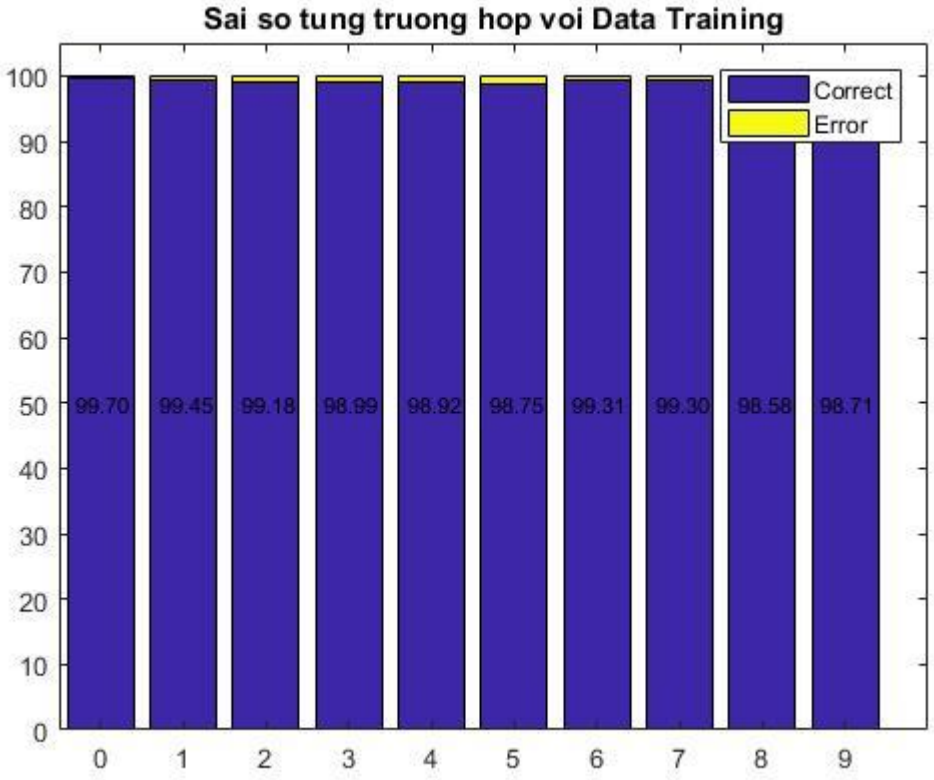
Thay đổi thông số của các lớp: (trong phần code đã comment)

```
cnnConfig.layer{2}.type = 'conv';
cnnConfig.layer{2}.outputmaps = 6; % so noron lop ay
cnnConfig.layer{2}.kernelsize = 5;

cnnConfig.layer{3}.type = 'pool';
cnnConfig.layer{3}.scale = 2;

cnnConfig.layer{4}.type = 'conv';
cnnConfig.layer{4}.outputmaps = 12; % so noron lop ay
cnnConfig.layer{4}.kernelsize = 5;

cnnConfig.layer{5}.type = 'pool';
cnnConfig.layer{5}.scale = 2;
```

Trong quá trình học với 100 Epoch hệ số học: Từ 1: 1.4, từ 30: 1.2, từ 60: 1, từ 80: 0.8




Sai so tung truong hop voi Data Testing
Correct
Error
100
90
80
70
60
50
40
30
20
10
0
99.80
99.30
99.32
99.31
98.68
98.66
98.85
98.73
98.25
98.61
0
1
2
3
4
5
6
7
8
9


Sai so tung truong hop voi Data Training
Correct
Error
100
90
80
70
60
50
40
30
20
10
0
99.70
99.45
99.18
98.99
98.92
98.75
99.31
99.30
98.58
98.71
0
1
2
3
4
5
6
7
8
9

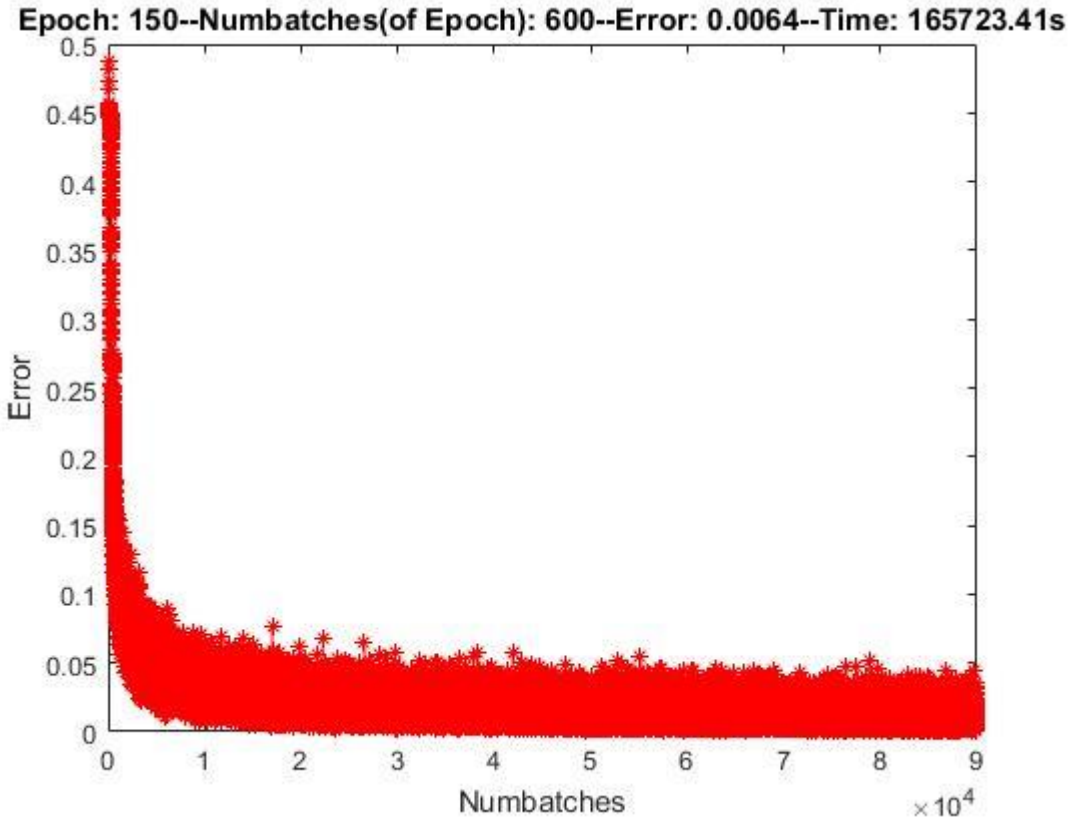
Sai số trên toàn dữ liệu:

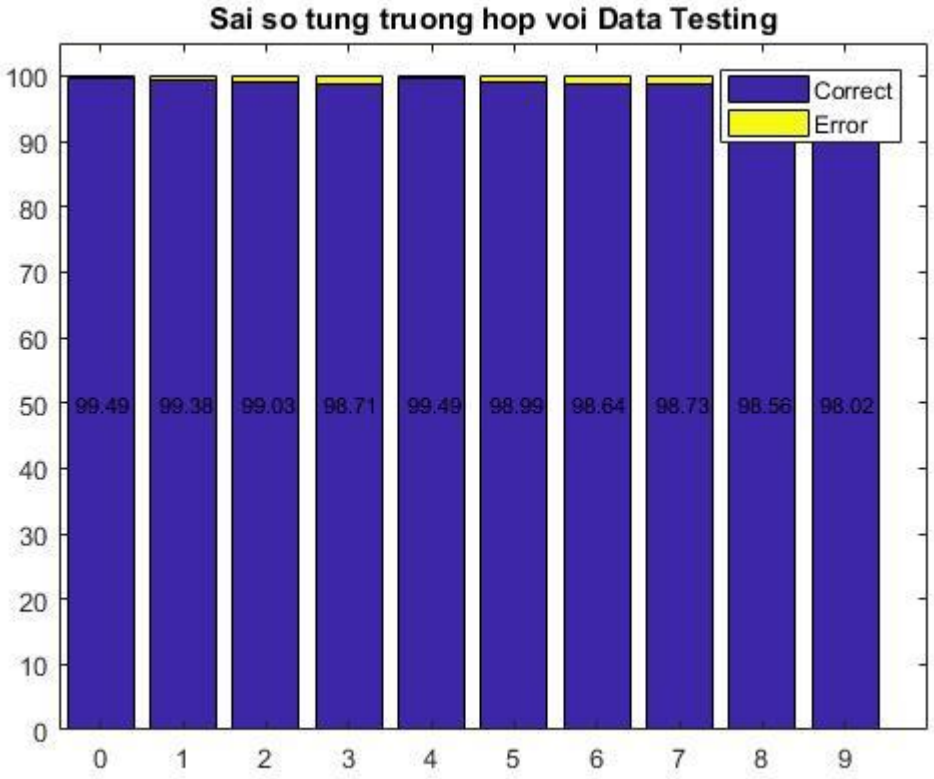
- Data Test: 1.04 %
- Data Train: 0.9%

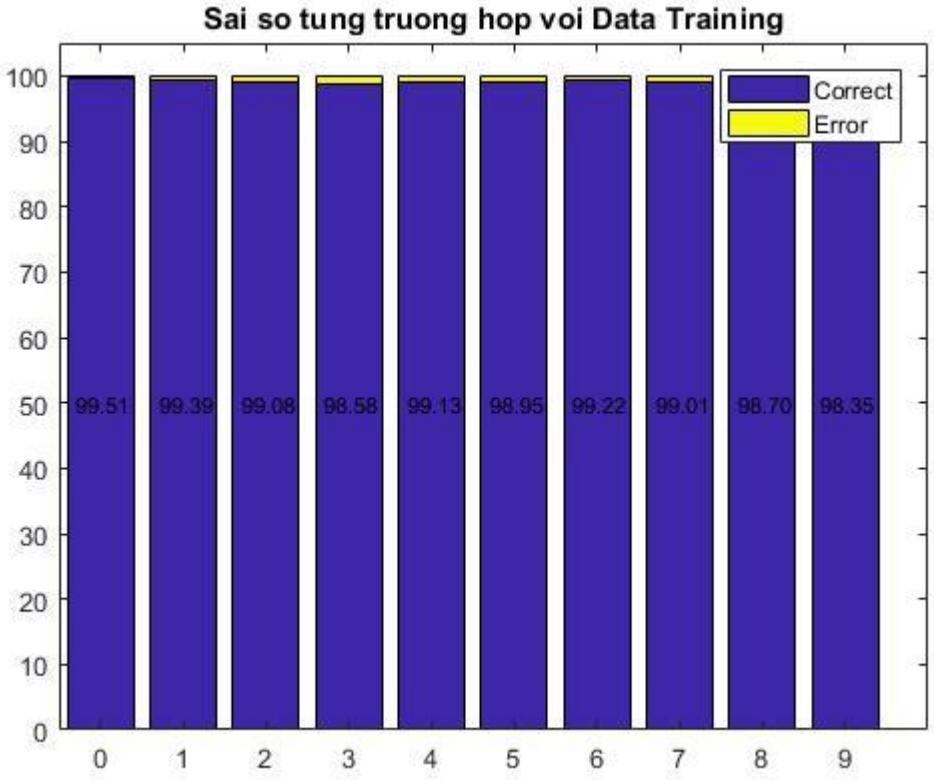
i) 150 Epoch

Thay đổi Batch Size từ 50 lên 100

Trong quá trình học với 150 Epoch hệ số học: Từ 1: 1.4, từ 40: 1.2, từ 80: 1, từ 110: 0.8, từ 130: 0.8




Sai so tung truong hop voi Data Testing
Correct
Error
100
90
80
70
60
50
40
30
20
10
0
99.49
99.38
99.03
98.71
99.49
98.99
98.64
98.73
98.56
98.02
0
1
2
3
4
5
6
7
8
9


Sai so tung truong hop voi Data Training
Correct
Error
100
90
80
70
60
50
40
30
20
10
0
99.51
99.39
99.08
98.58
99.13
98.95
99.22
99.01
98.70
98.35
0
1
2
3
4
5
6
7
8
9

Sai số trên toàn dữ liệu:

- Data Test: 1.09 %
- Data Train: 1%

4.2 Tổng kết kết quả đạt được với các tham số học khác nhau

a. Thay đổi số epoch và giữ mọi thông số còn lại

Mini batch = 50

Epoch	1	2	10	20	40	100	150
Error Train	12.32%	9.027%	3.07%	2.058%	1.353%	0.9%	0.9%
Error Test	11.28 %	8.21 %	2.72 %	2 %	1.17 %	1.05 %	1.04%

Nhận xét:

- Ta thấy từ Epoch 40 trở lên (100 hoặc 150) sai số giảm không đáng kể nhưng thời gian **Train** tăng rất nhiều, nên chọn dừng lại ở **epoch 40 là hợp lý**

b. Thay đổi mini batch và giữ các thông số còn lại

Mini batch	50	100
Error Train	0.9%	1.09 %
Error Test	1.04%	1%

Nhận xét:

- Ta thấy với mini batch bằng 100 và bằng 50 kết quả không khác nhau nhiều, nhưng thời gian **Train** giảm nhiều so với mini batch bằng 50 nên ta chọn mini batch bằng 100

c. Thay đổi số noron trong lớp và giữ các thông số còn lại

2 trường hợp mô hình mạng như đã trình bày ở trên

Với Epoch = 100

Trường hợp	1	2
Error Train	1.05%	1.04 %
Error Test	0.9%	0.9%

Nhận xét:

- Ta thấy 2 mô hình mạng có kết quả giống nhau, tuy vậy ở mô hình mạng 1 đơn gian hơn nên tối ưu thời gian và dữ liệu hơn(tránh hiện tượng **overfitting**) nên **chọn mô hình mạng 1**

4.3 Cấu trúc mạng được lưu sau khi học và cách sử dụng

Mạng được lưu với trường hợp cho mạng học lần sử dụng 150 Epoch

Kết quả mạng noron sau khi huấn luyện được lưu dưới dạng '.mat', dưới cấu trúc struc có tên là : ' Net_Saved.mat' trong thư mục code

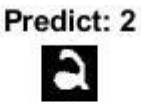
Muốn sử dụng mạng đã được học cho trường hợp mới chỉ cần load dữ liệu mạng và sử dụng hàm "runNN.m" trong thư mục code, với đầu vào là cấu trúc mạng, kết quả đầu ra là kết quả dự đoán ứng với đầu vào và hình ảnh của đầu vào.

Ví dụ:

```
load Net_Saved; % load mang vao workspase
test= runNN(net, test_x(:,:,1));
```

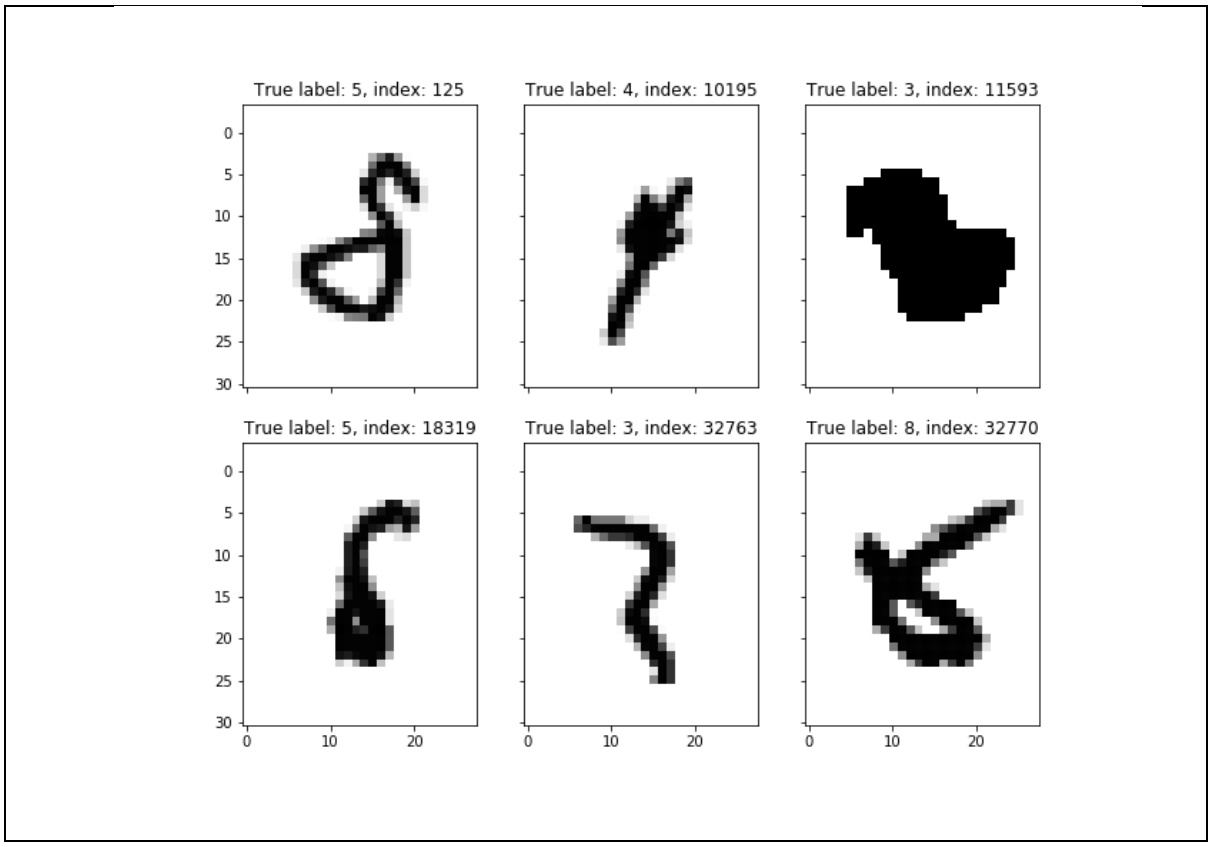
Kết quả: test = 1

Và hiển thị hình ảnh:



5. Kết Luận

- Nhờ các công thức toán học nhóm đã xây dựng được chương trình khởi tạo, huấn luyện, kiểm thử … cho mạng noron CNN
- Trong thử nghiệm, từ Epoch là 40 so với (100 hay 150 epoch) thì kết quả không cải thiện đáng kể
- Số Mini batch(50 hoặc 100) hoặc thay đổi thông các lớp khác bài báo như trên đã đề cập kết quả cũng không thay đổi nhiều
- Thời gian học của mạng rất lớn: do mạng khá phức tạp và sử dụng CPU(tính toán tuần tự)
- Mạng cho **kết quả chính xác cao 99%**, để đạt chính xác cao hơn rất khó, điều đó được thể hiện như hình dưới (nguồn internet), qua hình dưới ta thấy một số ảnh rất khó có thể phân biệt nhãn nào là đúng.



Em xin chân thành cảm ơn cô đã hướng dẫn em môn học này. Qua môn học này, em nhận được những kiến thức nền tảng và có một cái nhìn tổng quan giúp em tiếp cận với mạng noron nhân tạo (hoặc AI nói chung) một cách dễ dàng và khoa học hơn. Em chúc cô và gia đình có nhiều sức khỏe và thành công trong cuộc sống, công việc.

6. Tài Liệu Tham Khảo

[1] Z. Zhang, “Derivation of Backpropagation in Convolutional Neural Network (CNN),” p. 7.