Handwritten digit recognition

Based on BP neural network

SC 6360 (Artificial Intelligence)

Semester 1/2017

Gao yang

6019545

CONTENTS

1 Introduction

2 Principle

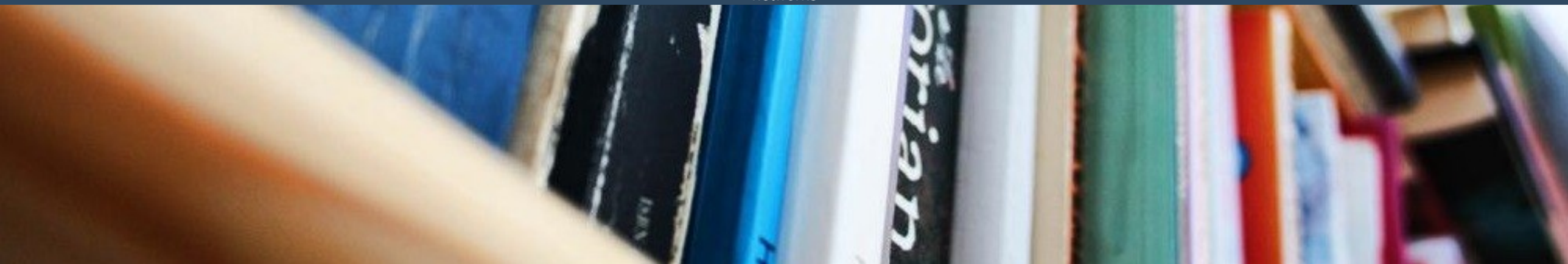
3 Component

4 Show

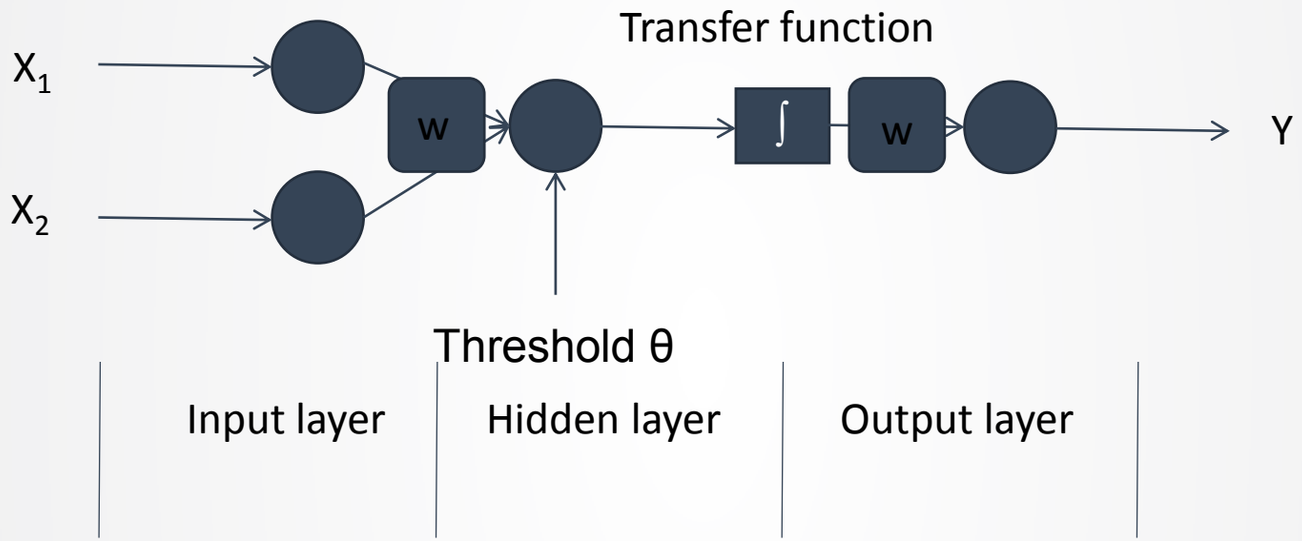
01

Introduction

Describe the basics to facilitate understanding of handwritten digital recognition based on neural networks

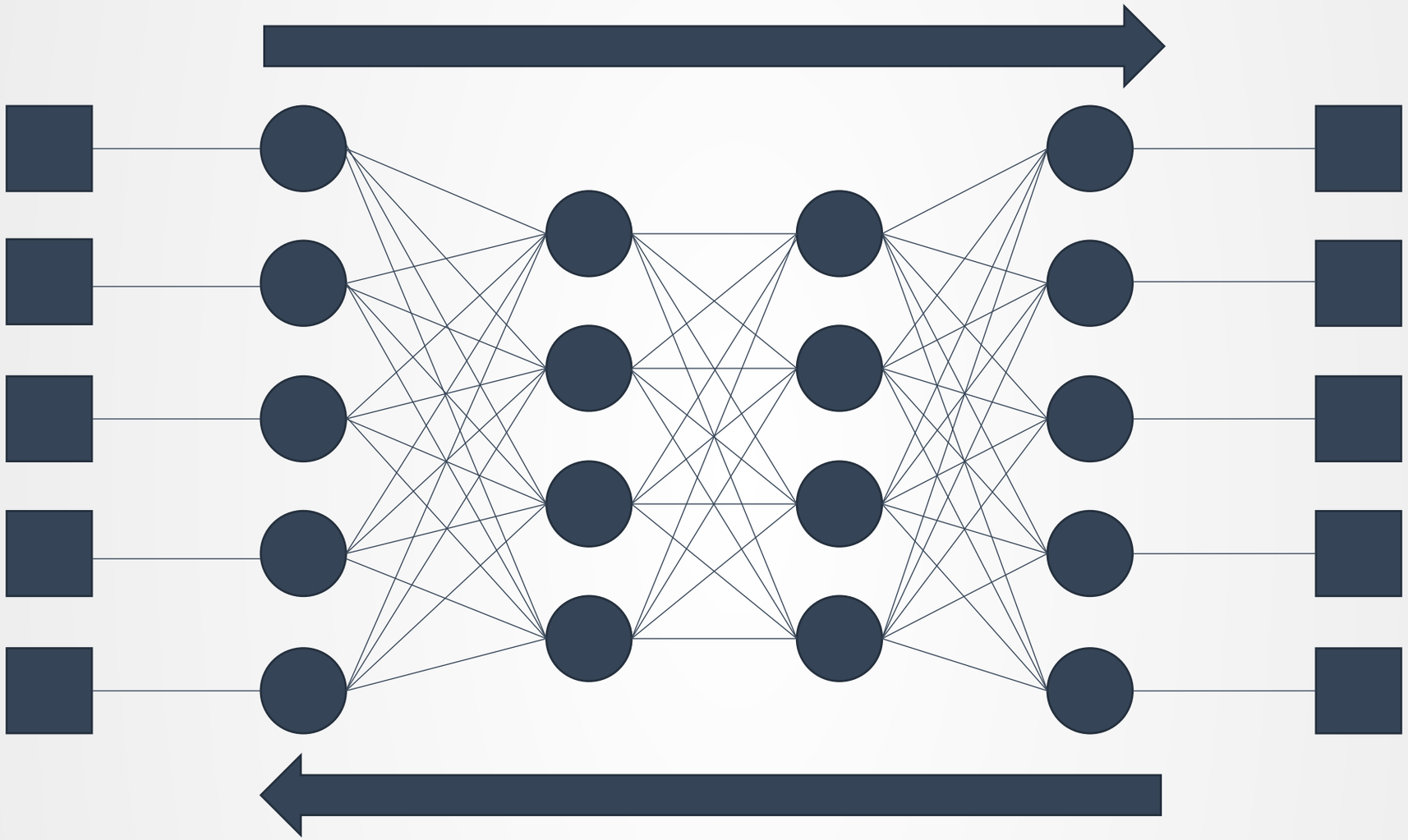


What is a neural network?



The principle of neural network

Calculate the output of each layer

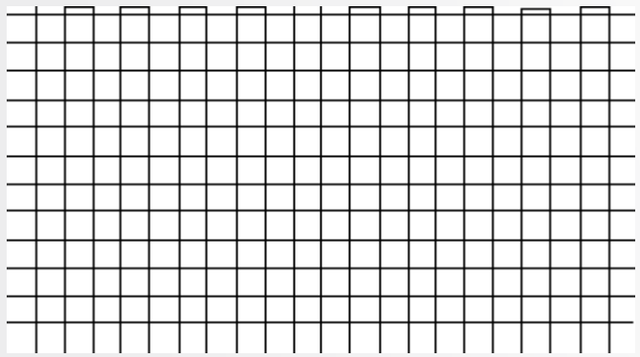


Update layer weights and thresholds

The composition of the image

Color picture

rgb picture



Pixel matrix

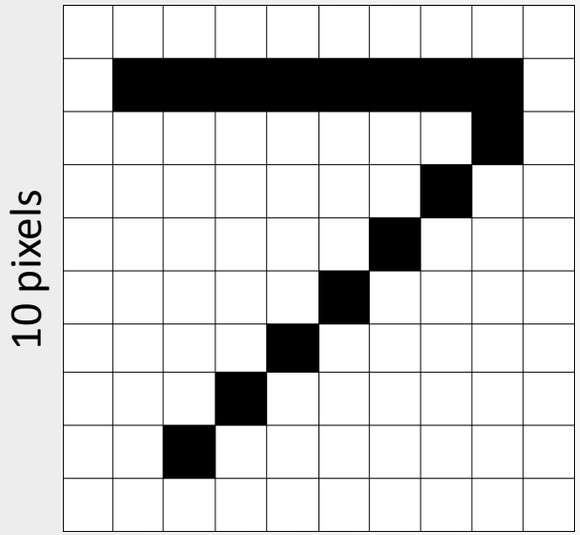
Grayscale picture

02

Principle

Show how to make the computer
recognize our handwritten numbers

Handwritten picture processing



0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0

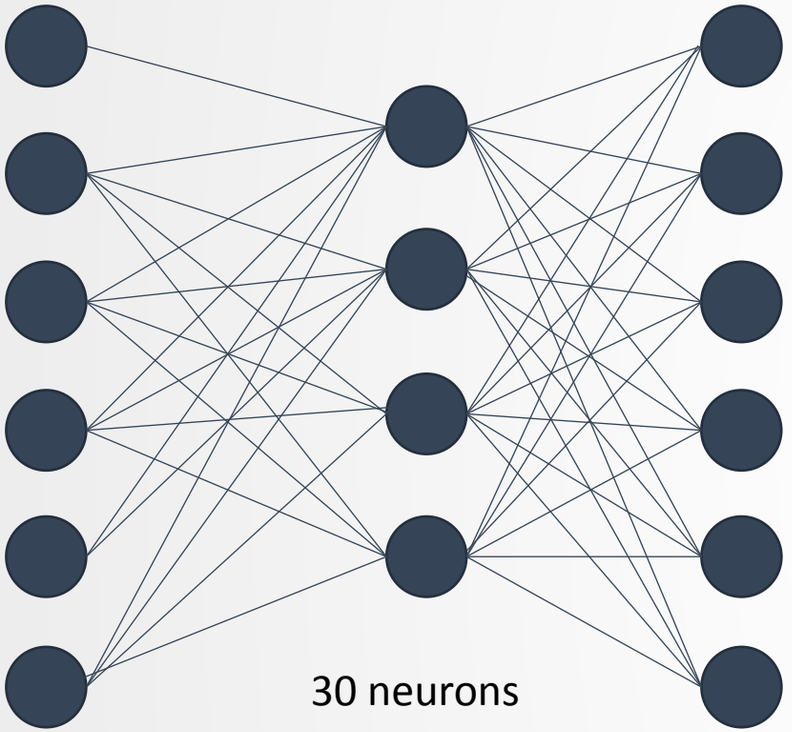
...

0	0	0	0	0	0	0	0	0	0

1 2 3 4 5 . . .
100

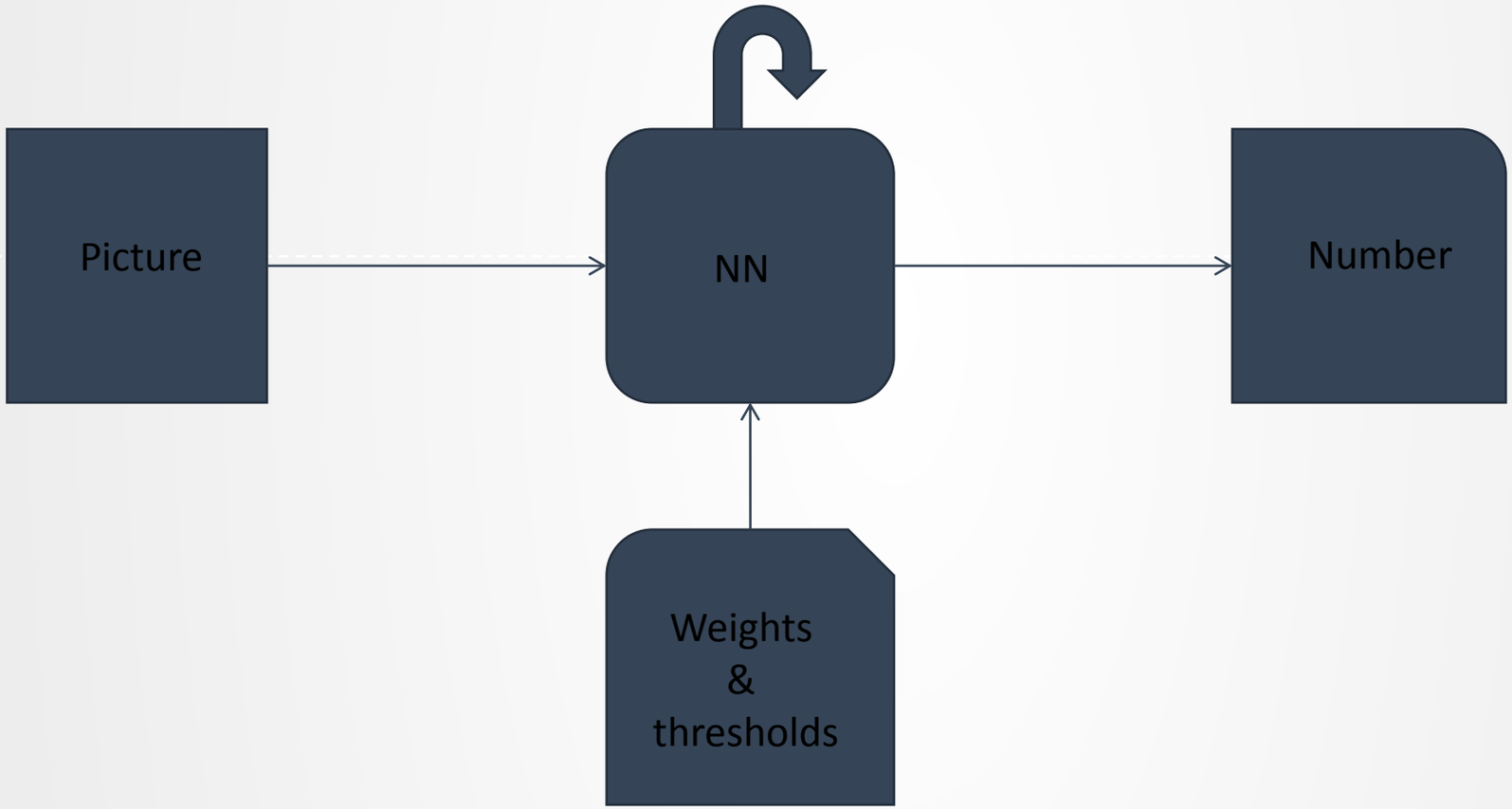
98 99

Build neural network



- 100 pixels as the input of 100 neurons
- Hidden layer 30 neurons can be arbitrarily changed
- 0 to 9 This 10 has numbers so output layer 10 neurons
- Random weights and thresholds are generated before the neural network is run
- The error accuracy will be less than 0.0001 after 5000 rounds of training
- Save weight and threshold after training to facilitate the direct use

Process



03

Component

Show how each component is implemented through the program

Hidden layer neurons actually output

```
for j=num_input_layer+1:num_input_layer+num_hidden_layer
        for i=1:num_input_layer%1到5
            x_ready(j,1)=train_data_sets(quanju,i)*weight_one(k,i);
            x(j,1)=x(j,1)+x_ready(j,1);
        end
        x(j,1)=x(j,1)-theta(1,s);
        y(j,1)=1/(1+exp(-(x(j,1))));
        s=s+1;
        k=k+1;
    end
```

y

140x1 double

	1
101	0.9141
102	0.9858
103	0.9714
104	3.8407e-05
105	0.9827
106	8.7418e-07
107	6.6888e-04
108	6.1187e-05
109	0.0117
110	0.9929
111	0.9406
112	0.9993
113	2.6454e-08
114	0.9938
115	0.9780
116	0.9490
117	0.8187
118	0.9919
119	1.0000
120	0.0074
121	2.1477e-05
122	0.0869
123	0.0086
124	0.3573
125	0.7300
126	4.4507e-05
127	0.0028
128	0.0401
129	0.9998
130	0.0168
131	1.9841e-04

Output layer neurons actually output

```
k=1;s=num_input_layer+num_hidden_layer+1;z=num_input_layer+1;
      for
j=num_input_layer+num_hidden_layer+1:num_input_layer+num_hidden_layer+num_outpu
t_layer
         for i=1:num_hidden_layer
            x_ready(j,1)=y(z,1)*weight_second(k,i);
            x(j,1)=x(j,1)+x_ready(j,1);
            z=z+1;
         end
         z=num_input_layer+1;
         x(j,1)=x(j,1)-theta(1,s);
         y(j,1)=1/(1+exp(-(x(j,1))));
         s=s+1;
         k=k+1;
      end
```

130	0.0168
131	1.9841e-04
132	8.2009e-05
133	1.3769e-04
134	0.0059
135	0.0010
136	6.5326e-07
137	7.3518e-04
138	0.0012
139	0.0056
140	0.9872

Output layer neuron error slope

```
r=1;k=1;
    for j=num_input_layer+num_hidden_layer+1:num_input_layer+num_hidden_layer+num_output_layer
      %for k=1:label_y
      delte(j,1)=y(j,1)*(1-y(j,1))*(train_label_sets(quanju,k)-y(j,1));
      k=k+1;
      %end
    end
```

130	5.6964e-08
131	-3.9358e-08
132	-6.7250e-09
133	-1.8956e-08
134	-3.5045e-05
135	-1.0937e-06
136	-4.2674e-13
137	-5.4009e-07
138	-1.3581e-06
139	-3.1494e-05
140	1.6204e-04

Output layer weight correction and update

```
s=1;d=1;
    for j=num_input_layer+1:num_input_layer+num_hidden_layer
      for i=num_input_layer+num_hidden_layer+1:num_input_layer+num_hidden_layer+num_output_layer

        tan_w(j,i)=alpha*y(j,1)*delte(i,1);
        weight_second(s,d)=weight_second(s,d)+tan_w(j,i);
        s=s+1;
      end
      d=d+1;
      s=1;
    end
```

weight_second

10x30 double

	1	2	3	4	5	6	7
1	-0.8577	-1.8236	-0.0149	0.6521	0.3504	-2.2322	-1.8806
2	-1.6824	0.5095	1.6883	1.7451	-1.3332	-1.8815	-1.3196
3	0.2325	-1.8378	1.8882	-3.0595	-1.0147	1.0747	2.3675
4	0.2723	-0.7783	1.3615	-1.5590	-0.2870	-1.0692	0.4370
5	-1.6895	-0.6546	-1.8431	-0.8058	1.0656	-1.4864	1.3373
6	-0.4130	0.1229	-2.1485	1.1804	-0.4333	1.9317	1.2578
7	-0.2692	-0.5864	-1.9941	-0.0977	-3.6117	1.2312	0.0194
8	-1.3742	1.3238	0.4712	-0.5719	1.1779	1.2579	-0.8969
9	-0.2586	-1.3175	-2.3511	-0.5708	2.1494	-1.0699	-1.1618
10	-0.0030	1.7389	-0.2526	-2.4143	-1.1058	-1.1571	-1.4364

Calculate hidden layer neuron error slope

```
 q=1;w=1;
      for i=num_input_layer+1:num_input_layer+num_hidden_layer
         for
j=num_input_layer+num_hidden_layer+1:num_input_layer+num_hidden_layer+num_output_layer
            delte_ready(i,1)=delte(j,1)*weight_second(q,w);
            delte_readys(i,1)=delte_readys(i,1)+delte_ready(i,1);
            q=q+1;
         end
         q=1;
         w=w+1;
         delte(i,1)=y(i,1)*(1-y(i,1))*delte_readys(i,1);
      end
```

140x1 double

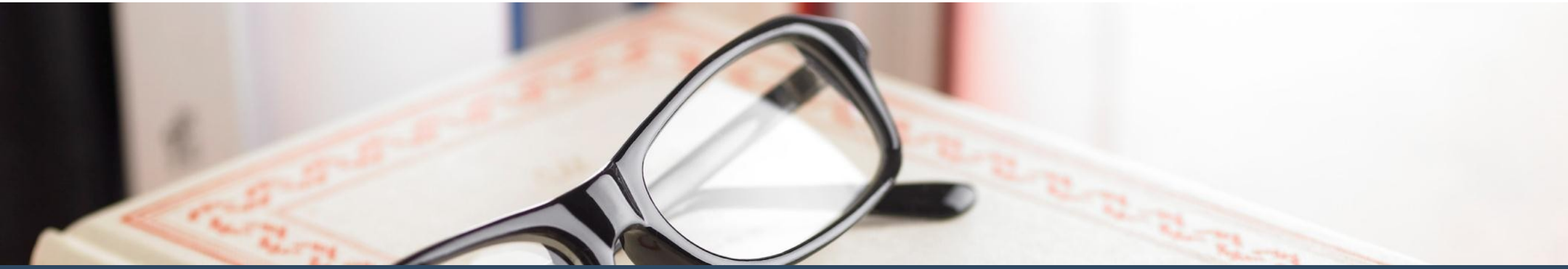
	1
101	1.5782e-07
102	4.9107e-06
103	-3.3827e-07
104	-1.2170e-08
105	-4.0499e-06
106	-1.0228e-10
107	-1.4151e-07
108	-8.7375e-09
109	2.5201e-06
110	-4.5597e-07
111	3.4224e-05
112	-6.5572e-08
113	2.6253e-12
114	-7.7446e-08
115	-4.2282e-06
116	3.1924e-05
117	-8.6410e-06
118	3.4495e-06
119	-8.2898e-10
120	-1.2787e-06
121	1.2379e-09
122	-1.4219e-05
123	-1.2985e-06
124	-2.5603e-05
125	3.7642e-06
126	-6.0796e-09
127	-1.8443e-07
128	-1.2076e-05
129	5.0434e-08
130	5.6964e-08

Calculate and update hidden layer weights

```
s=1;d=1;
    for j=num_input_layer+1:num_input_layer+num_hidden_layer
        for i=1:num_input_layer
            tan_w(i,j)=alpha*train_data_sets(quanju,s)*delte(j,1);
            weight_one(d,s)=weight_one(d,s)+tan_w(i,j);
            s=s+1;
        end
        s=1;
        d=d+1;
    end
```

30x100 double

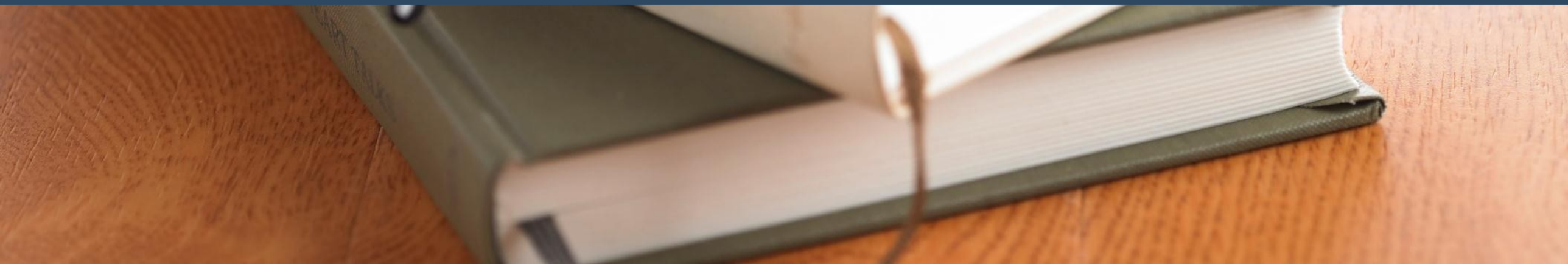
	1	2	3	4	5	6	7
1	0.1088	0.8653	0.0354	-0.9590	0.0748	0.4343	0.6045
2	0.7344	-0.7925	-0.3068	-0.8264	-0.1924	-0.4430	-0.8400
3	-0.5421	0.5956	-0.8360	-0.8915	0.7105	-0.9602	-0.5632
4	0.4258	-0.2006	-0.6204	0.1377	0.5801	-0.4968	-0.3541
5	0.5825	0.9542	-0.7278	0.3202	-0.2737	-0.8309	0.2131
6	-0.2456	-0.8423	-0.6262	0.5820	0.4847	0.1266	-0.5639
7	0.2735	0.0242	-0.4153	0.1006	0.3763	0.2373	0.7033
8	-0.8803	-0.8209	-0.2030	0.0530	-0.6833	-1.0362	-0.9513
9	-0.8227	1.1923	-0.8339	-0.4004	-0.4847	-0.5380	0.6898
10	0.2670	-0.8852	0.8708	0.6361	-0.0357	0.6074	0.1632
11	0.4190	0.4419	0.7533	-0.5961	-0.3953	-0.7212	0.3925
12	0.8747	0.6406	-0.2216	0.2691	0.2930	0.4938	-0.0021

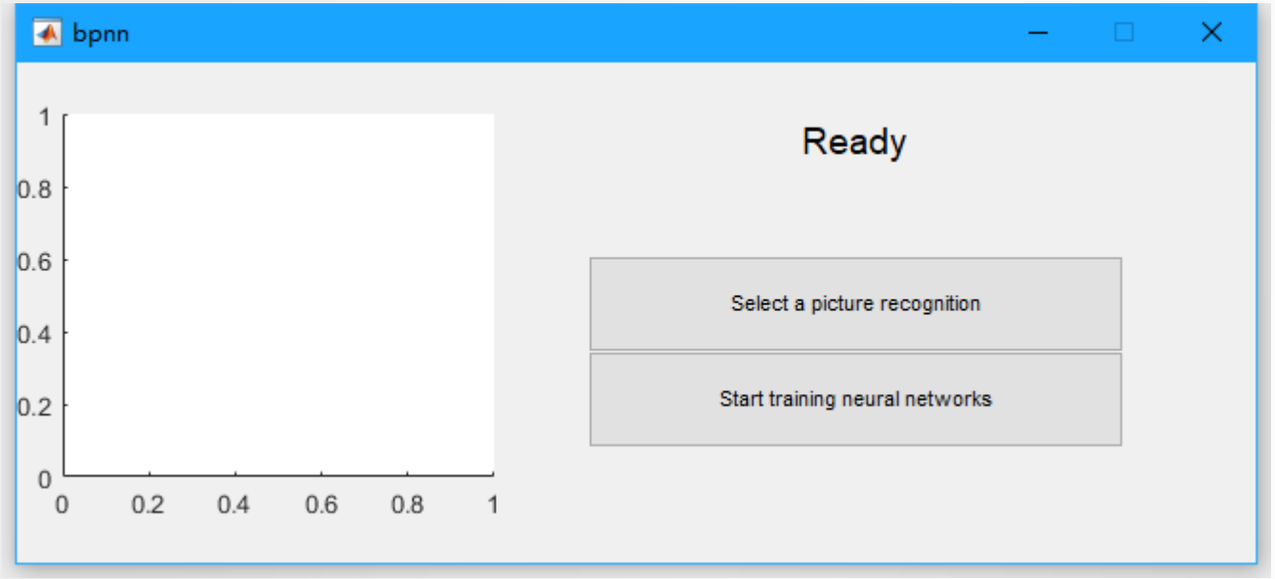


04

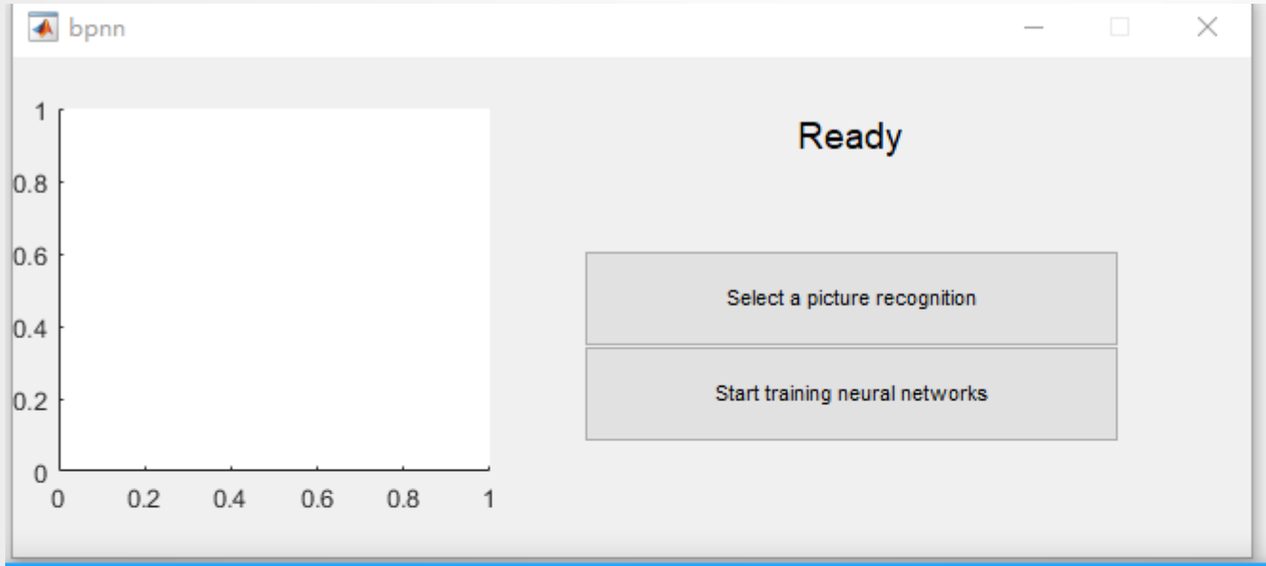
Show

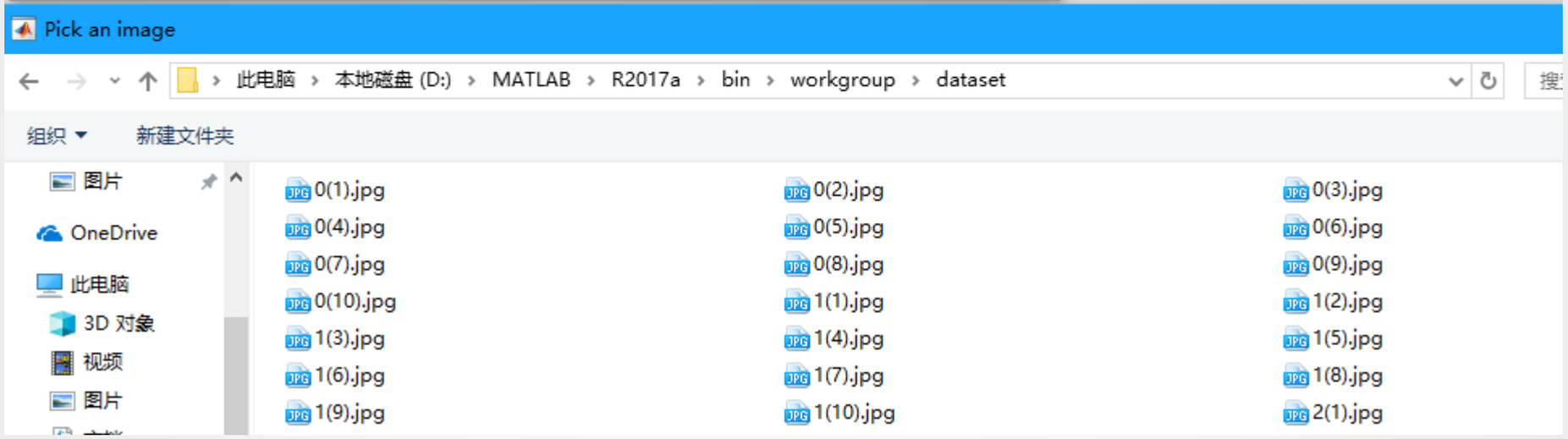
Program interface and use of the show

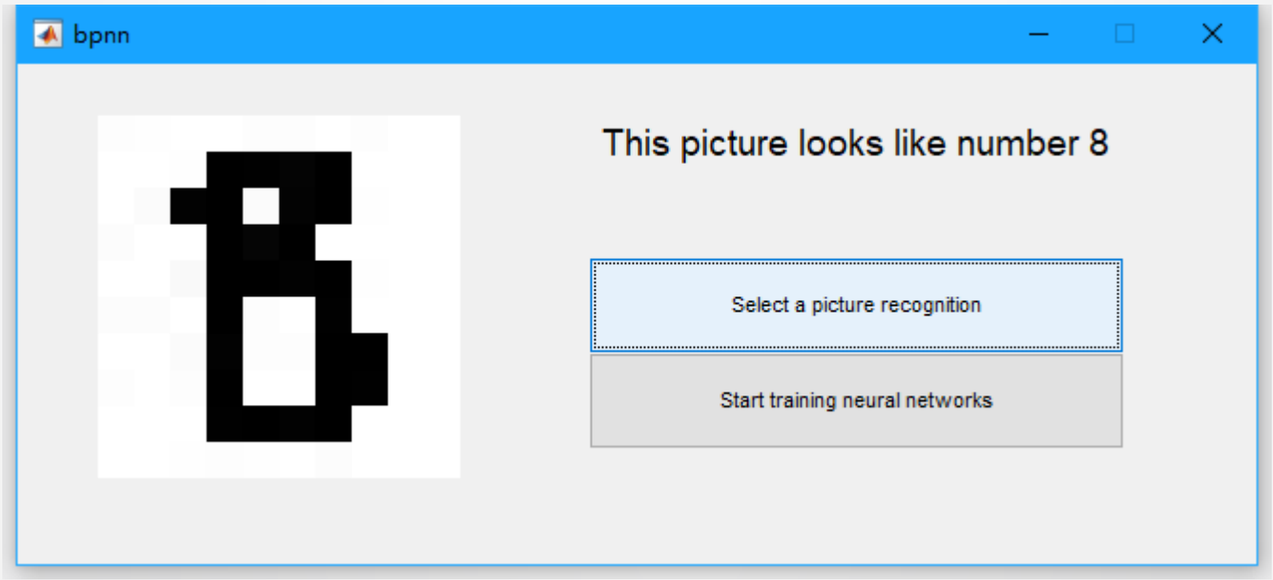


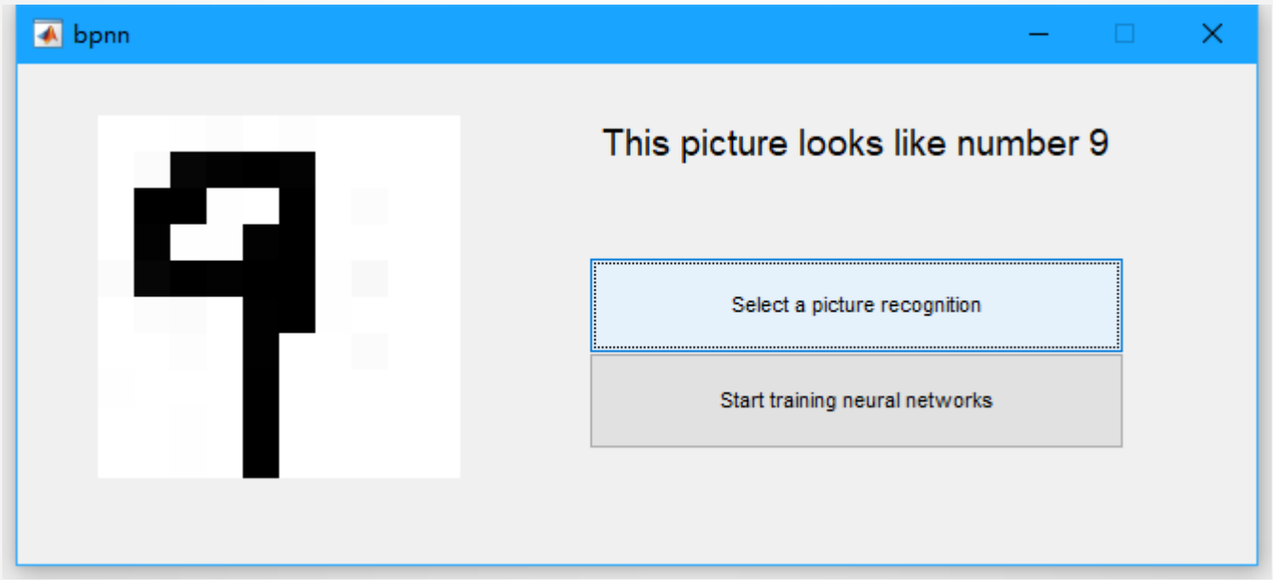



```
Training set data total 100 rows . . .
Training set data total 100 cols . . .
Set the initial parameters . . .
Generate weights and thresholds randomly . . .
Number of neurons in input layer -> 100
Number of neurons in hidden layer -> 30
Number of neurons in output layer -> 10
Learning rate -> 0.1
Sum-squared error -> 0.0001
Calculate the cycle -> 5000
Start training . . .
Neural network training is completed . . .
The correct rate is ->> 0.99996
```


bpnn
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
Ready
Select a picture recognition
Start training neural networks


Pick an image
此电脑 › 本地磁盘 (D:) › MATLAB › R2017a › bin › workgroup › dataset
组织
新建文件夹
图片
OneDrive
此电脑
3D 对象
视频
图片
0(1).jpg
0(2).jpg
0(3).jpg
0(4).jpg
0(5).jpg
0(6).jpg
0(7).jpg
0(8).jpg
0(9).jpg
0(10).jpg
1(1).jpg
1(2).jpg
1(3).jpg
1(4).jpg
1(5).jpg
1(6).jpg
1(7).jpg
1(8).jpg
1(9).jpg
1(10).jpg
2(1).jpg


bpnn
This picture looks like number 8
Select a picture recognition
Start training neural networks


bpnn
This picture looks like number 9
Select a picture recognition
Start training neural networks

THANK YOU