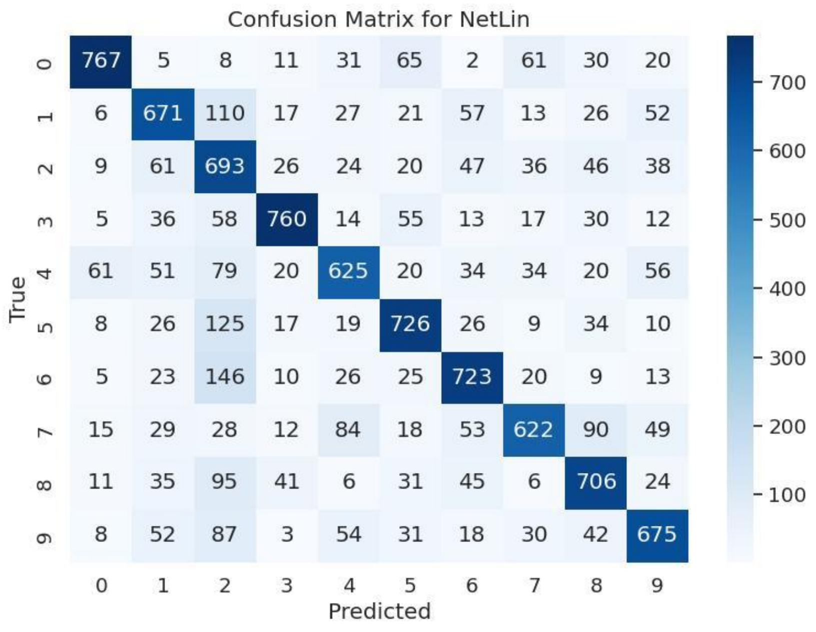
Assignment Report

COMP9444 Neural Networks and Deep Learning

Name: Hongbo Li   zID: z5569460

- ## Part 1: Japanese Character Recognition

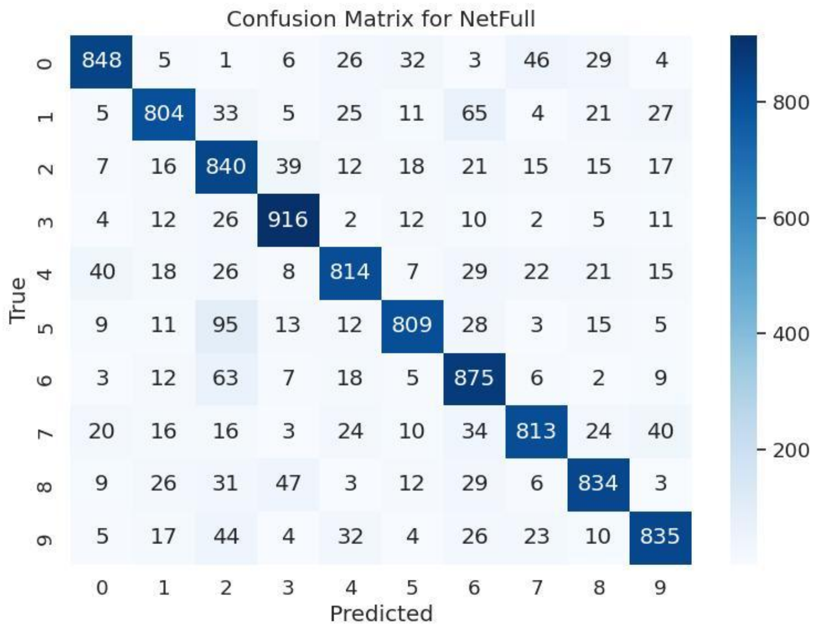
1. NetLin



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	767	5	8	11	31	65	2	61	30	20
1	6	671	110	17	27	21	57	13	26	52
2	9	61	693	26	24	20	47	36	46	38
3	5	36	58	760	14	55	13	17	30	12
4	61	51	79	20	625	20	34	34	20	56
5	8	26	125	17	19	726	26	9	34	10
6	5	23	146	10	26	25	723	20	9	13
7	15	29	28	12	84	18	53	622	90	49
8	11	35	95	41	6	31	45	6	706	24
9	8	52	87	3	54	31	18	30	42	675

Test set: Average loss: 1.0098, Accuracy: 6955/10000 (70%)

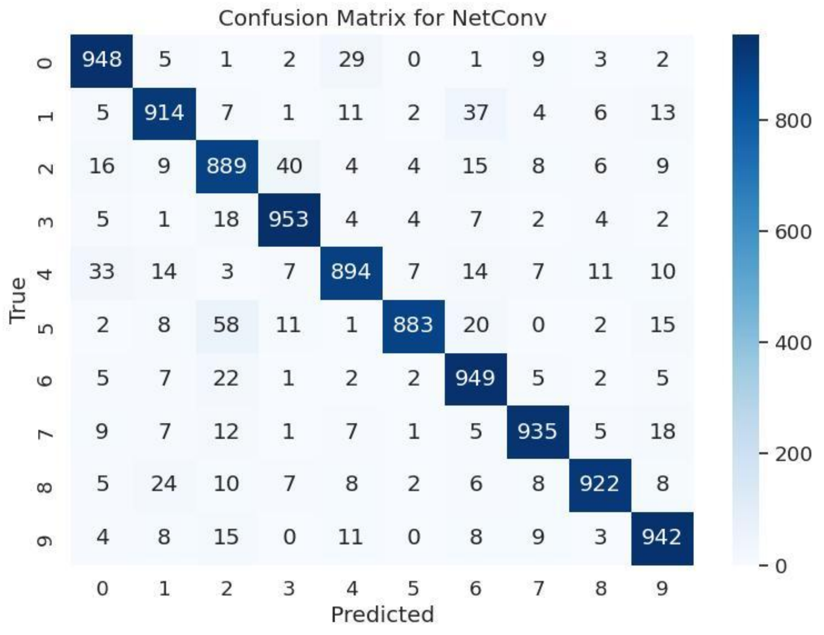
2. NetFull



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	848	5	1	6	26	32	3	46	29	4
1	5	804	33	5	25	11	65	4	21	27
2	7	16	840	39	12	18	21	15	15	17
3	4	12	26	916	2	12	10	2	5	11
4	40	18	26	8	814	7	29	22	21	15
5	9	11	95	13	12	809	28	3	15	5
6	3	12	63	7	18	5	875	6	2	9
7	20	16	16	3	24	10	34	813	24	40
8	9	26	31	47	3	12	29	6	834	3
9	5	17	44	4	32	4	26	23	10	835

Test set: Average loss: 0.5335, Accuracy: 8388/10000 (84%)

3. NetConv



Test set: Average loss: 0.2870, Accuracy: 9229/10000 (92%)

4. Briefly discuss the following points:

a: the relative accuracy of the three models

Model	**Training Accuracy (%)**	**Testing Accuracy (%)**	**Parameters**
NetLin	83	70	7850
NetFull	94	84	79510
NetConv	98	92	108618

NetConv is the best model due to its ability to extract and utilize spatial features; NetFull is better than NetLin due to its non-linear capacity but still limited; NetLin is the worst because it cannot model the complexity of the data. The performance gap between NetConv and the other models is large, mainly because convolutional architectures are much better suited for image recognition tasks.

b: the number of independent parameters in each of the three models,
The number is as follows:
NetLin: 7,850 parameters
NetFull: 79,510 parameters
NetConv: 108618 parameters

c: the confusion matrix for each model: which characters are most likely to be mistaken for which other characters, and why?

NetLin

"ma" (6) → "su" (2)
"ha" (5) → "su" (2) and "ma" (6)
"na" (4) → "o" (0), "su" (2), and "ma" (6)

"ki" (1) → "su" (2)
"ya" (7) → "ma" (6) and "na" (4)

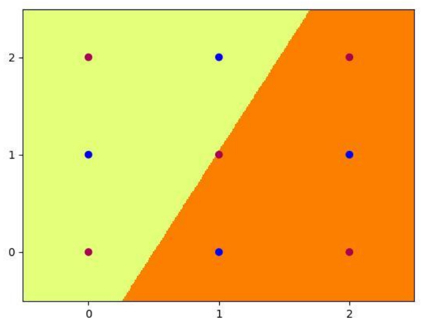
NetFull
"ha" (5) → "su" (2) and "ma" (6)
"ma" (6) → "su" (2) and "ha" (5)
"na" (4) → "o" (0), "su" (2), and "ma" (6)
"ki" (1) → "ma" (6) and "su" (2)

NetConv
"na" (4) → "o" (0) and "ma" (6)
"ha" (5) → "su" (2) and "ma" (6)
"su" (2) → "tsu" (3) and "ma" (6)
"ki" (1) → "ma" (6).

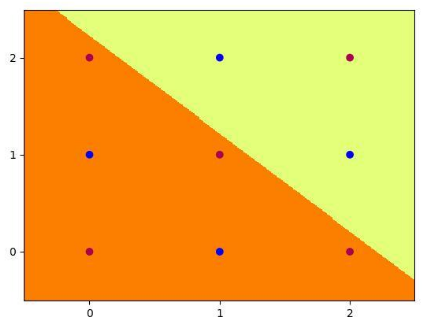
Across all models, characters such as "su" (2), "ma" (6), "ha" (5), and "na" (4) tend to be confused with each other, due to their similar handwritten forms.
The ‘NetConv’ significantly reduces these confusions by effectively extracting spatial and local features, leading to higher accuracy and fewer errors.

- ## Part 2: Multi-Layer Perceptron

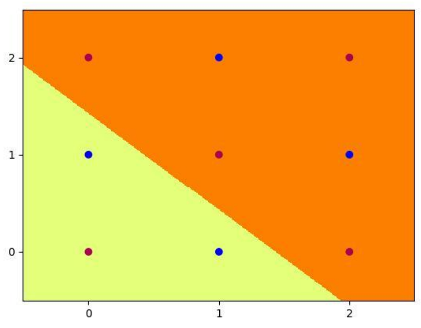
1. Copy these images into your report



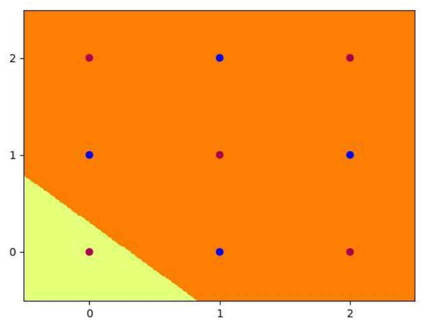
hid_5_0.jpg



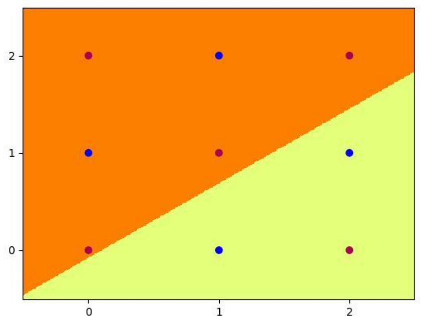
hid_5_1.jpg



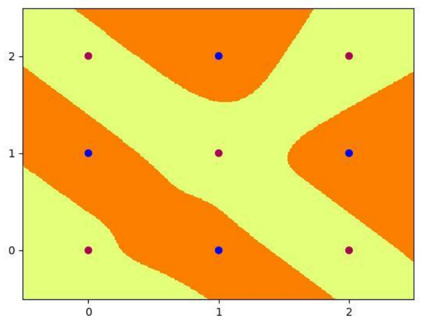
hid_5_2.jpg



hid_5_3.jpg



hid_5_4.jpg



out_5.jpg

2. Design by hand a 2-layer neural network with 4 hidden nodes

Network Structure

Input layer: 2 nodes (x, y)

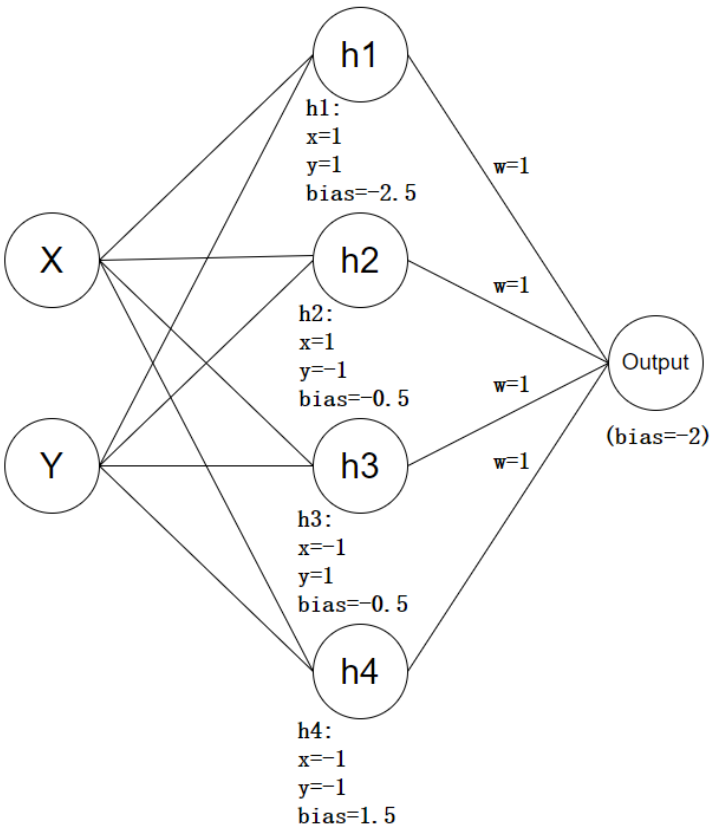
Hidden layer: 4 nodes

Output layer: 1 node

Weights and biases:

Hidden Node	Weights (x, y)	Bias	Dividing Line Equation
h1	(1, 1)	-2.5	x+y=2.5
h2	(1, -1)	-0.5	x−y=0.5
h3	(-1, 1)	-0.5	−x+y=0.5
h4	(-1, -1)	1.5	−x−y=−1.5

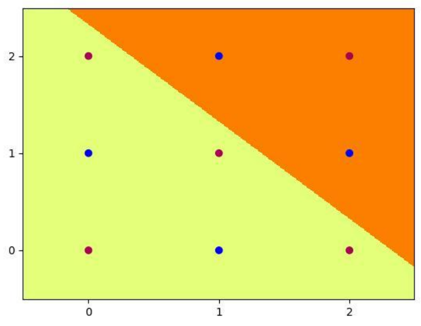
Network diagram:



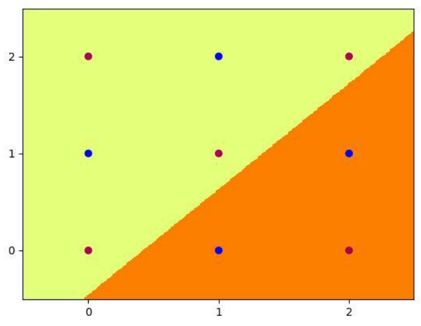
Activation Table

x	y	h1	h2	h3	h4	output
0	0	0	0	0	1	0
0	1	0	0	1	1	1
0	2	0	0	1	0	0
1	0	0	1	0	1	1
1	1	0	0	0	0	0
1	2	1	0	1	0	1
2	0	0	1	0	0	0
2	1	1	1	0	0	1
2	2	1	0	0	0	0

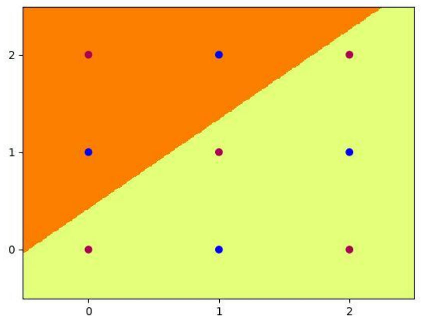
3. Copy these images into your report (rescaled)



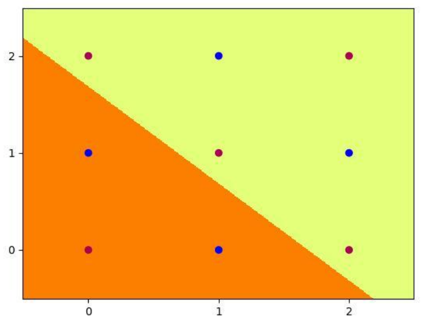
hid_4_0.jpg



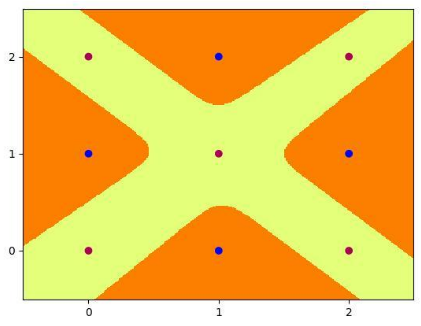
hid_4_1.jpg



hid_4_2.jpg



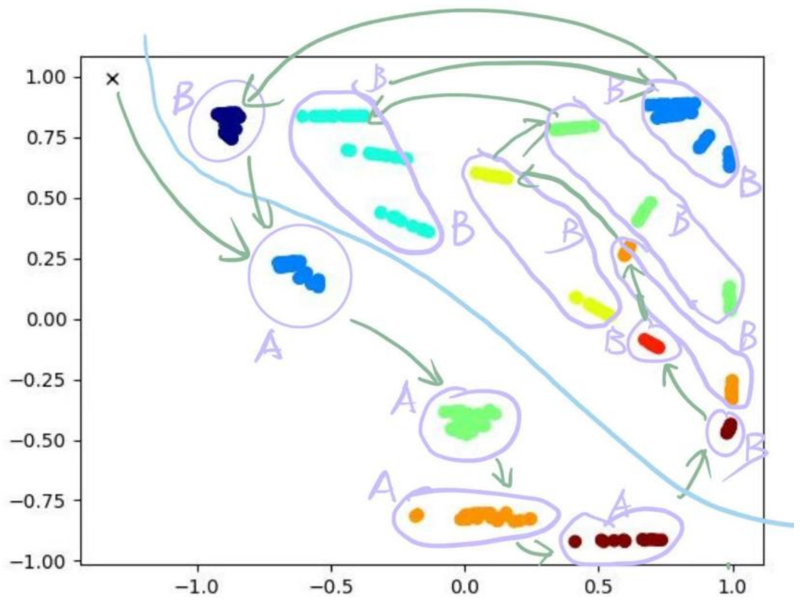
hid_4_3.jpg



out_4.jpg

- ## Part 3: Hidden Unit Dynamics for Recurrent Networks

1. Train a Simple Recurrent Network (SRN)



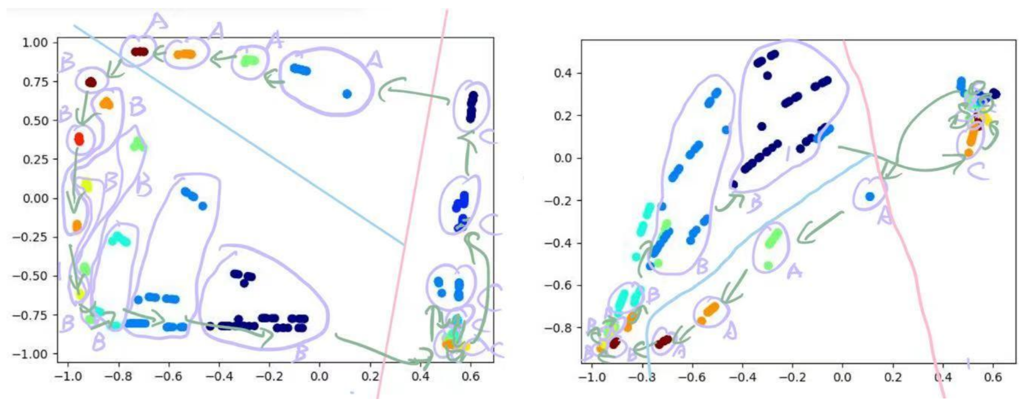
2. Draw a picture of a finite state machine

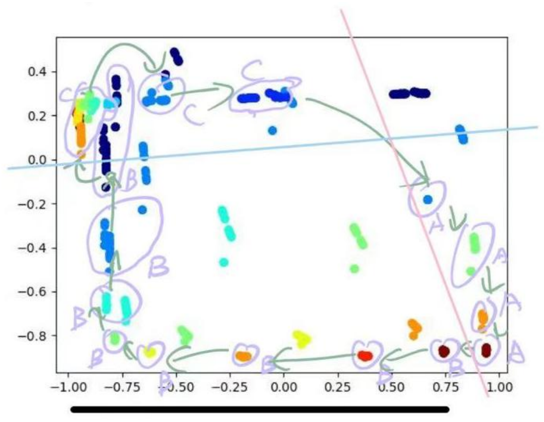
$$\begin{tikzcd}
X \arrow[r] & A \arrow[r] & A \arrow[r] & A \arrow[r] & A \arrow[r] & B \arrow[r] & B \arrow[d] \\
& B \arrow[u] & B \arrow[l] & B \arrow[l] & B \arrow[l] & B \arrow[l] & B \arrow[l]
\end{tikzcd}$$

3. Briefly explain how the network accomplishes the task

The network uses its hidden units to calculate and learn the number of A seen and then to track the required number of B; when A are read, hidden activations move through distinct states representing the count, and when B are processed, the activations transition through another sequence of states, allowing the network to correctly output B for twice the number of A and then switch to outputting A after the final B.

4. Train an LSTM with 3 hidden nodes





5. Try to analyze the dynamics of the hidden and/or context units, and explain how the LSTM successfully accomplishes the prediction task

LSTM is like a counter which can remember how many A's, B's, and C's it has seen. It will first record the A, then, when B, it knows it should be twice as many B as A, so it keeps track and outputs B at the times. After B, it does the same for C, making sure there are three times as many C as A.