**UNSW Business School**

ZZEN9444-Neural Network, Deep Learning

**Assessment 2: Image Processing**

Author: Ben Behar

Student Number: Z5348153

Due Date: 26/09/2022

Table of Contents

[1 Part 1: Japanese character recognition 3](#_Toc115091553)

[1.1 Step 1: NetLin 3](#_Toc115091554)

[1.2 Step 2: NetFull 3](#_Toc115091555)

[1.3 Step 3: NetConv 4](#_Toc115091556)

[1.4 Report 7](#_Toc115091557)

[2 Twin Spiral 11](#_Toc115091558)

[2.1 Converting input to Polar co-ordinates 11](#_Toc115091559)

[2.2 Minimum Epochs 11](#_Toc115091560)

[2.3 RawNet 12](#_Toc115091561)

[2.4 Hidden Unit Graphs 13](#_Toc115091562)

[2.5 Report 15](#_Toc115091563)

[2.5.1 PolarNet vs RawNet 15](#_Toc115091564)

[2.5.2 –hid and –init in RawNet 15](#_Toc115091565)

[2.5.3 Manipulate parameters 16](#_Toc115091566)

[3 Hidden Unit Dynamics 3](#_Toc115091567)

[3.1 Star 16 3](#_Toc115091568)

[3.2 9-2-9 4](#_Toc115091569)

[3.3 Heart18 6](#_Toc115091570)

[3.4 Free Design 7](#_Toc115091571)

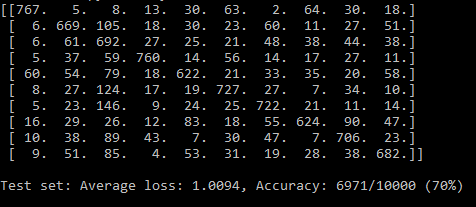
[3.4.1 Target 1 7](#_Toc115091572)

[3.4.2 Target 2 9](#_Toc115091573)

# Part 1: Japanese character recognition

## Step 1: NetLin

Figure 1: NetLin Final Results



## Step 2: NetFull

Figure 2: NetFull Final Results, 390 Hidden Nodes

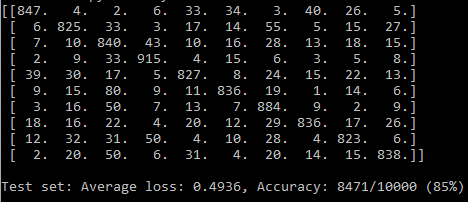


Figure 3: NetFull Final Results, 720 Hidden Nodes

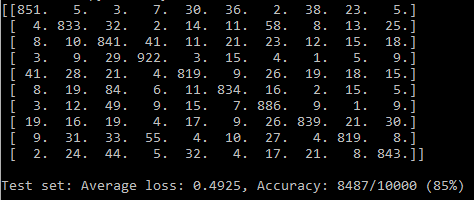
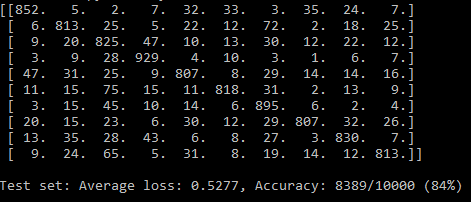


Figure 4: : NetFull Final Results, 100 Hidden Nodes



## Step 3: NetConv

Try 1 – “Kernal = 5”, “Padding = 2” for first Conv Layer

Figure 5: NetConv 1

Text

Description automatically generated

A picture containing calendar

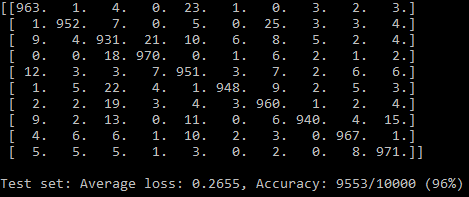
Description automatically generated

Try 2 – Add “padding =2” to second conv layer (best)

Figure 6: NetConv 2

Text

Description automatically generated

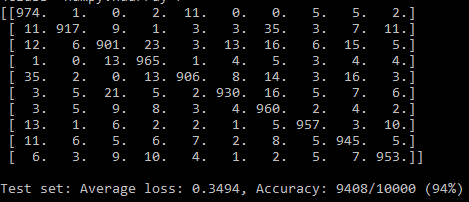


Try 3 – kernel=3 and no padding

Figure 7: NetConv 3

Text

Description automatically generated



## Report

Table 1: Accuracy Summary

|  |  |  |
| --- | --- | --- |
| **Network** | **Correct/10,000** | **Accuracy** |
| NetLin | 6,971 | 70% |
| NetFull (720 HU) | 8,487 | 85% |
| NetConv (Try 2) | 9,553 | 96% |

Table 2: NetLin Matrix Extreme Misprediction

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| NetLin 70% | | **Prediction** | | | | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **Target** | 0 |  | | | | | | | | | |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |

Table 3: NetFull Matrix Extreme Misprediction

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| NetFull 85% | | **Prediction** | | | | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **Target** | 0 |  | | | | | | | | | |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |

Table 4: NetConv Matrix Extreme Misprediction

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| NetConv 94% | | **Prediction** | | | | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **Target** | 0 |  | | | | | | | | | |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |

From the tables above, we can see that one combination repeatedly has a high misprediction rate.

The models are predicting 2 (“Su”) すwhen the target is 5 (“ha”) は

2

|  |  |
| --- | --- |
| 2 =”su” | 5 = “ha” |
| す  2  1 | は  1 |

Another character that all models had a hard time predicting is “ma” an “ki”

Prediction 6 ("ma") ま for target 1 ("ki") き

4

|  |  |
| --- | --- |
| 6= “ma” | 1 = “ki” |
| ま  3 | き  4  3 |

As we can see in both accusations, we can see that the characters have similar features, such as straight horizontal line 1, and loop 3, between “su” and “ha”.

And double cross line 3, and loop 4, between “ki” and “ma”.

The models did a good job of improving the results by making fewer mistakes predicting 2 ("su”) すfor target 6 ("ma”) ま.

|  |  |
| --- | --- |
| 2 =”su” | 6= “ma” |
| す | ま |

As there are similar features, the model improves from 146 (highest) misprediction for the matrix to 49 and, eventually, only 19 incorrect answers in NetConv.

After changing networks as required in the assignment, including trying different hidden layers in NetFull, and architectures in NetConv, it appears that NetConv produces the best result.

I have tried to manipulate the architecture and parameters to understand what is happening to the model while doing so and the results are as follows:

**No max pooling** - Input to full connected layer 16\*28\*28 = 12,544

was very hard to compute and took a long time; I had to terminate the process in the middle.

**--mom =1** - Immediately increased the loss function and “explode”. At the 2nd epoch, when --mom =1.5 explode on the first epoch.

Figure 8: Momentum = 1, 2nd Epoch

A picture containing text

Description automatically generated

**--lr = 0.1** – Achieved high results very fast (93% at first epoch), increased to 95%, dropped to 94%, and finished at 96% accuracy. Loss function kept on increasing and decreasing.

**--lr = 0.5 and –mom = 0.7** – Produce similar results as above, more stability in loss function and accuracy; however, a maximum of 96% accuracy was achieved only at the 10th epoch.

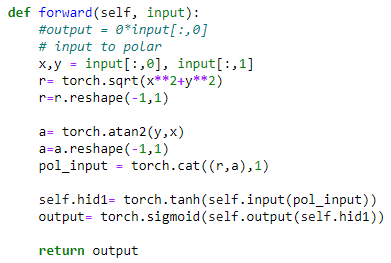
**Epoch = 15** I tried to implement 15 epochs on the above parameters, thinking it would keep increasing the accuracy, as they were growing consistently.

However, the accuracy remained at 96% until it dropped to 95% at the 14th epoch and returned to 96% in the final epoch.

# Twin Spiral

## Converting input to Polar co-ordinates

Figure 9: PolarNet



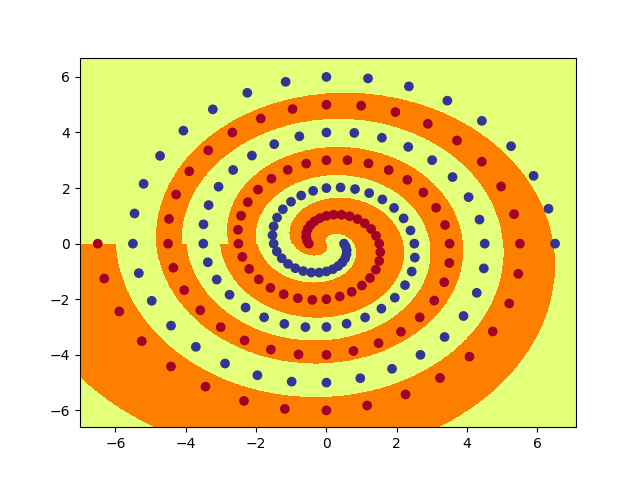
## Minimum Epochs

At –hid = 5, the model stock on 99.48% from around epoch 3600.

At –hid = 6, the model solved the problem a few times; however, it mainly stayed around 90%, reaching 20,000 epochs.

At –hid = 7, models solve the problem every time around 10,000 epochs.

Figure 10: PolarNet output with 7 hidden layers



## RawNet

Figure 11:RawNet Code

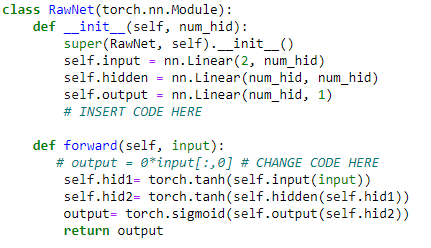
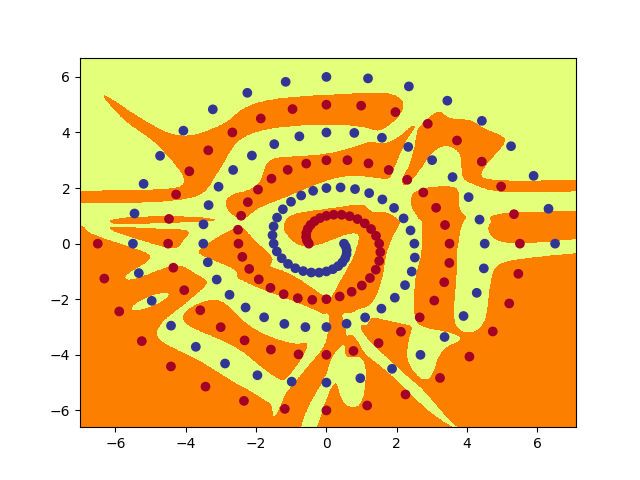


Figure 12: RawNET Hid=16 init=0.5 output



## Hidden Unit Graphs

Table 5: PolarNet Hidden Units Graphs

|  |  |  |
| --- | --- | --- |
| **PolarNet** | | |
|  |  |  |
|  |  |  |
|  |  |  |

Table 6: RawNet Hidden Units Graphs

|  |  |  |  |
| --- | --- | --- | --- |
| **RawNet** | | | |
| **First layer** | | **Second layer** | |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Report

### PolarNet vs RawNet

The models show that in current architectures, PolarNet is much more efficient in finding the solution than RawNET (Table 7).

We can also observe it in the graphical presentation of the models Figure 14) & (Figure 13)

It uses fewer nodes in the hidden layer, 7 compared to 16+16=32.

And much fewer weights between the input to hidden layers

(1+2) \*7 = 21 in-hid

Compare to (1+2) \*16 + (1+16) \*16 = 48 + 272 = 320 in-hid-hid

Table 7: PolarNet and RawNet Hidden layers Comparison

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **PolarNet** | **RawNet** | | |
|  |  | **Hid 1** | **Hid 2** | **Total** |
| Weight per neuron in hid layer | 3 | 3 | 17 |  |
| Nodes In hidden layers | 7 | 16 | 16 | 32 |
| Weights In-hid | 21 | 48 | 272 | 320 |

### –hid and –init in RawNet

**Hid =10-15 and init=0.1** – when hidden units are between 10 and 15, the model couldn’t learn and reached only around 50% after 20,000 epochs.

**Hid=16 and init = 0.1** – model is successfully learned at around 6000 epochs

**Hid=16 and init = 0.5** – Increasing init to 0.5 improved the learning process and achieved 100% around 3000 epochs

**Hid=16 and init = 1** - Achieved higher accuracy faster, however, stayed on around 95% until “late” epochs and sometimes failed learning before reaching 20,000.

**Hid=25 and init = 0.5** – Dramatically improved the model and was able to learn after only 2000 epochs and less.

It looks like the number of hidden nodes is much more critical to the model than the initial weight size.

### Manipulate parameters

I have manipulated the architecture and parameters in RawNet and Polar to observe how it changes the process, and the results are as follows:

|  |  |  |
| --- | --- | --- |
|  | **PolarNet**  hid=7 | **RawNet**  hid=7 init=0.5 |
| **Batch = 194** | Improving results most of the time and completing the learning process around 2000 epochs; however, other times, minor improvement. | Achieving results much faster, approximately after 2000 epochs |
| **Optimizer = SGD**  **No momentum**  **Batch =97** | Couldn’t learn before reaching 20,000  And reaching up to approx. 65% | |
| **Momentum = 0.5** | No significant change from above. | |
| **Momentum = 1** | Reaching approximately 75% but still can’t learn within 20,000 epochs. | |
| **Momentum = 0.9** | Reaching approx. 90% but still can’t learn within 20,000 epochs. | |
| **Momentum = 0.9**  **Batch =194** | No significant change from the same momentum and batch=97. | |
| **Original parameters**  **Activation=relu** | Couldn't learn within 20,000 epochs and remain on 67% | In some accusations, the model could learn within 20,000 and even 10,000 epochs. However, in some attempts, it took as many as 40,000 epochs. |
| **Add 3rd layer to RawNet** | n/a | Significant improvement. The model was able to learn within 1000 epochs on every attempt. |

Figure 14: PolarNet Diagram

Figure 13: RawNet Diagram

Input

Hidden

Output

Hidden

1

2

3

4

5

**. ..**

16

1

2

3

4

5

**. ..**

16

Output

Hidden

Input

# Hidden Unit Dynamics

## Star 16

Figure 15: Star16 Graph

Chart

Description automatically generated

## 9-2-9

From the graphs below, we can see how all 9 unit activations (dots) start in approximately the "same place", slowly receiving different values, and separate from the "group" (Table 8).

Output boundaries (lines) began to appear in our view, and even able to separate a unit around epoch 300 when the loss function is approximately 0.2003 (Table 8).

From there, the learning process is roughly going at the same rate as it takes the model to correctly separate another 6 inputs at epoch 1500 (Table 8)..

However, to learn the last 2 inputs, the model is working much "harder" and was able to learn 1 more input just after 500 epochs, at epoch 2000 (Table 8)..

At epoch 3000, we can only identify improvement in the output lines; however, the model was not able yet to separate the last input. Only at epoch 16,270, the model divides the last input (approximately 13,000 epochs after) (Table 8).

Table 8: 9-2-9 Hidden layers Graphs

|  |  |  |
| --- | --- | --- |
| **1 - Ep 50: loss = 0.3501** | **2 - Ep 100: loss = 0.3487** | **3 - Ep 150: loss = 0.3456** |
|  |  |  |
| **4 - Ep 200: loss = 0.2924** | **5 - Ep 300: loss = 0.2003** | **6 - Ep 500: loss = 0.1663** |
|  |  |  |
| **7 - Ep 700: loss = 0.1491** | **8 - Ep 1000: loss = 0.1299** | **9 - Ep 1500: loss = 0.1106** |
|  |  |  |
| **10 - Ep 2000: loss = 0.0969** | **11 - Ep 3000: loss = 0.0784** | **Final - Ep 16270:**  **loss = 0.0200** |
|  |  |  |

## Heart18

Figure 16: Heart18 Excel Worksheet

Table

Description automatically generated

Figure 17: Heart18 Output

Chart, scatter chart

Description automatically generated

## Free Design

### Target 1

The tensor made from 60 rows and 24 columns, and the model have the following architecture and parameters (Figure 20).

Input = 60

Output = 24

lr=0.75

Stop = 0.0125

All other parameters are as default.

Figure 18: Target1 Result "BEN"

Chart, scatter chart

Description automatically generated

Figure 19: Target1 Excel Spreadsheet diagram

Table

Description automatically generated

Figure 20: Target1 Excel Spreadsheet one-hot code

Background pattern

Description automatically generated

### Target 2

The tensor made from 40 rows and 20 columns, and the model have the following architecture and parameters (Figure 23).

Input = 40

Output = 20

lr=0.9

Stop = 0.0325

All other parameters are as default.

Figure 21: Target 2 Result "SUN"

Chart, scatter chart

Description automatically generated

Figure 22: Target 2 Excel Spreadsheet diagram

Table

Description automatically generated

Figure 23: Target 2 Excel Spreadsheet one-hot code

