

[Dashboard](#) / [My courses](#) / [ZZEN9444-6275_00043](#) / **Assessment 2 Implementing Neural Networks: Image processing**


Assessment 2 Implementing Neural Networks: Image processing



Submit your second assessment here.

[My Submissions](#)

[Submit Part 1 - written report PDF](#)[Submit Part 2 - kuzu.py](#)[Submit Part 3 - spiral.py](#)[Submit Part 4 - encoder.py](#)

Title	Start Date	Due Date	Post Date	Marks Available
 Assessment 2 Implementing Neural Networks: Image processing - Submit Part 1 - written report PDF	22 Aug 2022 - 00:00	26 Sep 2022 - 17:05	4 Oct 2022 - 16:05	100

TYPE	WEIGHT	DUE	EXPECTED TIME
Implementing Neural Networks - Image processing		Monday 5pm, Week 5 (26 September 2022)	4-6 hours

Before you attempt this assessment, complete or review coding exercise 3B: Image Classification.

← Coding Exercise 3B

Assessment instructions

What you need

You will need [PyTorch](#) installed on your computer.

Download the archive a2.zip and unzip it:

← Assessment 2 files

This should create a directory a2 with the data file spirals.csv as well as seven Python files:

- kuzu.py
- spiral.py
- encoder.py
- kuzu_main.py
- spiral_main.py
- endoder_main.py
- encoder_model.py.

Instructions

In this assignment, you will be implementing and training various neural network models for three different tasks, and analysing the results. You are to complete and submit three Python files kuzu.py, spiral.py and encoder.py, as well as a written report a2.pdf (in pdf format).

This assessment is comprised of 3 parts with multiple steps in each part. The steps require you to complete actions using the files provided and/or record your findings into the report. Ensure you read and complete each step as directed.

Please post any questions in the corresponding Ed forum.

You will be penalised 5% per day of the marks available for this assessment task if you submit it after the due date, unless you have an approved extension through Special Consideration.

Part 1 - Japanese character recognition

For Part 1 of the assignment, you will be implementing networks to recognize handwritten Hiragana symbols. The dataset to be used is Kuzushiji-MNIST or KMNIST for short. The paper describing the dataset is [available here](#). It is worth reading, but in short: significant changes occurred to the language when Japan reformed their education system in 1868, and the majority of Japanese today cannot read texts published over 150 years ago. This paper presents a dataset of handwritten, labeled examples of this old-style script (Kuzushiji). Along with this dataset, however, they also provide a much simpler one, containing 10 Hiragana characters with 7000 samples per class. This is the dataset we will be using.

This part of the assessment is comprised of four steps:

Step 1 [1 mark] - Implement a model [NetLin](#) which computes a linear function of the pixels in the image, followed by log softmax. In Python on your computer, run the code by typing:

```
python3 kuzu_main.py --net lin
```

Copy the final accuracy and confusion matrix into your report. The final accuracy should be around 70%. Note that the rows of the confusion matrix indicate the target character, while the columns indicate the one chosen by the network. (0="o", 1="ki", 2="su", 3="tsu", 4="na", 5="ha", 6="ma", 7="ya", 8="re", 9="wo"). More examples of each character can be [found here](#).

Step 2 [1 mark]- Implement a fully connected 2-layer network [NetFull](#) (i.e. one hidden layer, plus the output layer), using tanh at the hidden layer and log softmax at the output layer. Run the code by typing:

▲ Back to Top

Try different values (multiples of 10) for the number of hidden nodes and try to determine a value that achieves high accuracy (at least 84%) on the test set. Copy the final accuracy and confusion matrix into your report.

Step 3 [2 marks]- Implement a convolutional network called **NetConv**, with two convolutional layers plus one fully connected layer, all using relu activation function, followed by the output layer, using log softmax. You are free to choose for yourself the number and size of the filters, metaparameter values (learning rate and momentum), and whether to use max pooling or a fully convolutional architecture. Run the code by typing:

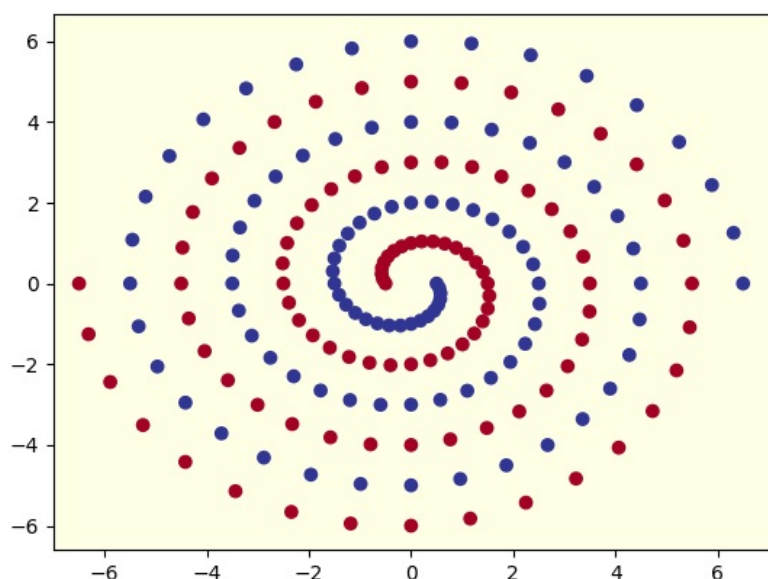
```
python3 kuzu_main.py --net conv.
```

Your network should consistently achieve at least 93% accuracy on the test set after 10 training epochs. Copy the final accuracy and confusion matrix into your report.

Step 4 [6 marks]- Discuss what you have learned from this exercise, including the following points:

- the relative accuracy of the three models,
- the confusion matrix for each model: which characters are most likely to be mistaken for which other characters, and why?
- experiment with other architectures and/or metaparameters for this dataset, and report on your results; the aim of this exercise is not only to achieve high accuracy but also to understand the effect of different choices on the final accuracy.

Part 2 - Twin spirals



For Part 2, you will be training on the famous Two Spirals Problem (Lang and Witbrock, 1988). The supplied code **spiral_main.py** loads the training data from **spirals.csv**, applies the specified model and produces a graph of the resulting function, along with the data. For this task there is no test set as such, but we instead judge the generalization by plotting the function computed by the network and making a visual assessment.

This part of the assessment is comprised of six steps:

Step 1 [1 mark]- Provide code for a Pytorch Module called **PolarNet** which operates as follows: First, the input (x,y) is converted to polar coordinates (r,a) with $r = \sqrt{x^2 + y^2}$, $a = \text{atan2}(y,x)$. Next, (r,a) is fed into a fully connected neural network with one hidden layer using tanh activation, followed by a single output using sigmoid activation. The conversion to polar coordinates should be included in your forward() method, so that the Module performs the entire task of conversion followed by network layers.

Step 2 [1 mark]- In Python on your computer, run the code by typing:

```
python3 spiral_main.py --net polar --hid 10.
```

Try to find the minimum number of hidden nodes required so that this PolarNet learns to correctly classify all of the training data within 20000 epochs, on almost all runs. The graph_output() method will generate a picture of the function computed by your PolarNet called polar_out.png, which you should include in your report.

Step 3 [1 mark]- Provide code for a Pytorch Module called RawNet which operates on the raw input (x,y) without converting to polar

shown in the diagram, but a larger number of hidden nodes are needed for the task. Secondly, our network will not have shortcut connections; although these appeared to be necessary in the original 1988 paper using SGD, it now appears that **the task can be learned without shortcut connections, with the help of the Adam optimizer.**

Step 4 [1 mark]- In Python on your computer, run the code by typing:

```
python3 spiral_main.py --net raw
```

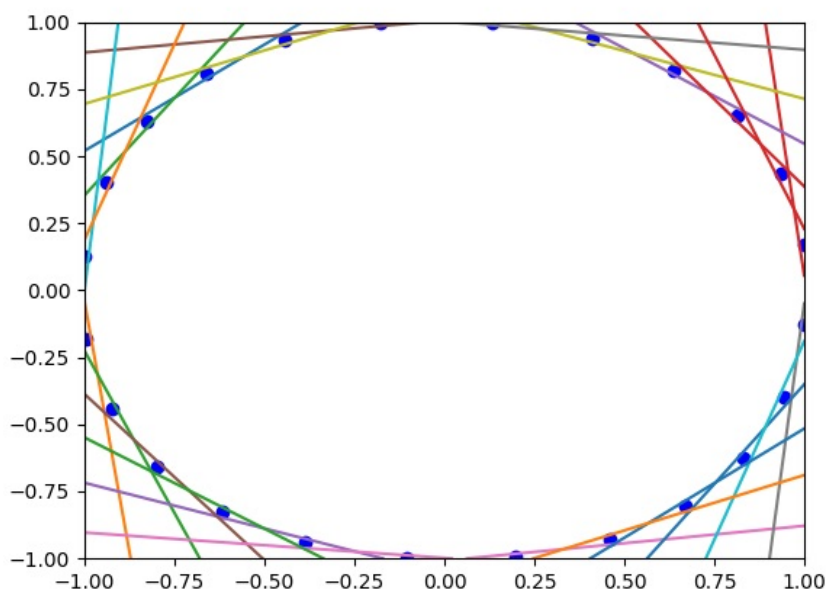
Try to choose a value for the number of hidden nodes (`--hid`) and the size of the initial weights (`--init`) such that this RawNet learns to correctly classify all of the training data within 20000 epochs, on almost all runs. Include in your report the number of hidden nodes, and the values of any other metaparameters. The `graph_output()` method will generate a picture of the function computed by your RawNet called `raw_out.png`, which you should include in your report.

Step 5 [2 marks]- Using `graph_output()` as a guide, write a method called `graph_hidden` (`net`, `layer`, `node`) which plots the activation (after applying the tanh function) of the hidden node with the specified number (`node`) in the specified layer (1 or 2). (Note: if `net` is of type PolarNet, `graph_output()` only needs to behave correctly when `layer` is 1). Hint: you might need to modify `forward()` so that the hidden unit activations are retained, i.e. replace `hid1 = torch.tanh(...)` with `self.hid1 = torch.tanh(...)`. Use this code to generate plots of all the hidden nodes in PolarNet, and all the hidden nodes in both layers of RawNet, and include them in your report.

Step 6 [6 marks]- Discuss what you have learned from this exercise, including the following points:

- The qualitative difference between the functions computed by the nodes in the hidden layer(s) in PolarNet and RawNet
- A brief description of how the network uses these functions to achieve the classification
- The effect of different values for initial weight size on the speed and success of learning for RawNet
- Experiment with other changes and comment on the result - for example, changing batch size from 97 to 194, using SGD instead of Adam, changing tanh to relu, adding a third hidden layer, etc. The aim is to understand how different choices impact the final result.

Part 3 - Hidden unit dynamics



In Part 3, you will be investigating hidden unit dynamics, as described in lesson 2, using the supplied code `encoder_main.py` and `encoder_model.py` as well as `encoder.py` (which you should modify and submit).

This part is comprised of four steps:

Step 1 [1 mark]- Run the code by typing:

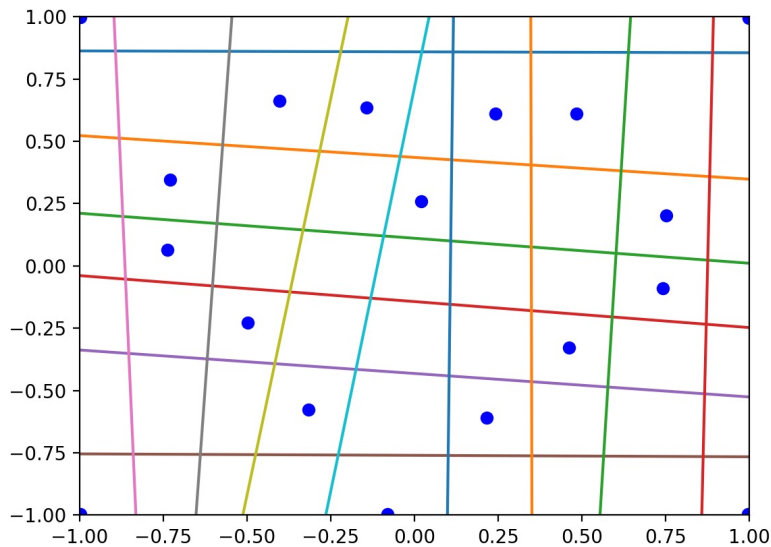
```
python3 encoder_main.py --target=star16
```

Save the final image and include it in your report. Note that `target` is determined by the tensor `star16` in `encoder.py`, which has 16 rows and 8 columns, indicating that there are 16 inputs and 8 outputs. The inputs use a one-hot encoding and are generated in the form of an identity matrix using `torch.eye()`

Step 2 [2 marks]- Run the code by typing `python3 encoder_main.py --target=input --dim=9 --plot`. In this case, the task is a 9-2-9 encoder. [Back to Top](#)

Step 3 [2 marks]- Create by hand a dataset in the form of a tensor called heart18 in the file encoder.py which, when run with the following command, will produce an image essentially the same as the heart-shaped figure shown below (but possibly rotated). Your tensor should have 18 rows and 14 columns. Include the final image in your report, and include the tensor heart18 in your file encoder.py

```
python3 encoder_main.py --target=heart18
```



Step 4 [3 marks]- Create training data in tensors target1 and target2, which will generate two images of your own design, when run with the command `python3 encoder_main.py --target=target1` (and similarly for target2). You are free to choose the size of the tensors, and to adjust parameters such as `--epochs` and `--lr` in order to achieve successful learning. Marks will be awarded based on creativity and artistic merit. Include the final images in your report, and include the tensors target1 and target2 in your file encoder.py

Presentation style/format

You will submit four files:

- A written report in pdf format (a2.pdf)
- kuzu.py
- spiral.py
- encoder.py

The written report should be comprised of your results and actions from the steps of each of the three parts of this assessment task. Before you submit please ensure you have clearly identified each action or response with appropriate sections, headings and sub-headings.

How to submit

Submit your pdf report and three Python files via Turnitin on this page. You may only submit a single file to each Turnitin part - use the tabs at the top of this page to navigate to each submission area for each file.

You may re-submit multiple times up to the due date. If you submit for the first time after the due date, you may only submit once (if you have submitted before the due date, you will be unable to re-submit after the due date). Normal late penalties apply for submission after the due date.

Additional information may be found in the [Ed discussion board](#) and will be considered as part of the specification for the project. You should check that page regularly.

Marking and feedback

You will receive feedback on your submission within five business days.

MARKING CRITERIA

UNSATISFACTORY - ZERO MARKS

SATISFACTORY - HALF MARKS

HIGH DISTINCTION - FULL MARKS

▲ Back to Top

		or does not achieve accuracy greater than 70%, but is a reasonable attempt.	accuracy greater than 70% in one of the three test runs.
Part 1, step 2 NetFull 1 mark	Did not attempt or overall approach is incorrect.	Code either not correct, does not produce sensible output or does not achieve accuracy greater than 82%, but is a reasonable attempt.	Code is correct, produces sensible output and achieves accuracy $\geq 82\%$ in one of the three test runs.
Part 1, step 3 NetConv 2 marks	Did not attempt or overall approach is incorrect.	Code either not correct, does not produce sensible output or does not achieve accuracy greater than 93%, but is a reasonable attempt.	Code is correct, produces sensible output and achieves accuracy $\geq 93\%$ (either in one of the three test runs, or in the report with plausible metaparameters mentioned)
Part 1, step 4 Discussion of accuracy, confusion matrix, other issues 6 marks	Did not attempt or overall approach is incorrect.	Accuracy is reported for fewer than three networks, or is not compared, or is not consistent with auto-testing. Discusses which characters are most likely to be mistaken but either does not discuss why or reasoning why is poor. Explores the effect of choices on the final accuracy, but either does not correctly identify impacts or exploration is limited.	Accuracy is reported and compared for all three networks, and is consistent with auto-testing. Discusses which characters are most likely to be mistaken and why. Correctly identifies and explores the effect of specific choices on the final accuracy.
Part 2, step 1 PolarNet 1 mark	Did not attempt or overall approach is incorrect.	Code is nearly correct, but contains one or more errors.	Code is correct and produces sensible output.
Part 2, step 2 PolarNet Graph 1 mark	Did not attempt or overall approach is incorrect.	Graph was generated by a Polar network, but it doesn't correctly classify the data.	Graph was generated by a Polar network and correctly classifies the data.
Part 2, step 3 RawNet 1 mark	Did not attempt or overall approach is incorrect.	Code is nearly correct, but contains one or more errors.	Code is correct and produces sensible output.
Part 2, step 4 RawNet graph 1 mark	Did not attempt or overall approach is incorrect.	Graph was generated by a raw network, but it doesn't correctly classify the data.	Graph was generated by a raw network and correctly classifies the data.
Part 2, step 5 Graph hidden 2 marks	Did not attempt or overall approach is incorrect.	Code runs, but images not quite correct or not all images are produced.	Code is correct, images are correct and are consistent with auto-testing.



computed by PolarNet and
RawNet
6 marks

discussed, but analysis is
incomplete. Explores the
effect of choices on the final
result, but either does not
correctly identify impacts or
exploration is limited.

functions computed by
hidden layers to achieve
classification. Correctly
identifies the effect of initial
weight size on the success of
learning for RawNet. Correctly
identifies and explores the
impact of different choices on
the final result.

Part 3, step 1 Star Image
1 mark

Did not attempt or overall
approach is incorrect.

Image is nearly correct.

Image is correct.

Part 3, step 2 Description of
hidden unit dynamics
2 marks

Did not attempt or overall
approach is incorrect.

Describes the pattern and
movement of the points
and/or the lines, but with
some errors or omissions.

Correctly describes the
pattern and movement of the
points and the lines.

Part 3, step 3 Heart Image
2 marks

Did not attempt or overall
approach is incorrect.

Image is nearly correct.

Image is correct.

Part 3, step 4 Two creative
images
3 marks

Did not attempt or overall
approach is incorrect.

Images are generated but
rudimentary, or only one
images is provided.

Images are creative and
artistic.

 Refresh Submissions

Submission
Title

Turnitin Paper
ID

Submitted

Similarity

Grade

Overall
Grade

-- --

--

--

--

--

--

Submit Paper 

--

--