Project Report: Neural Network from Scratch

Eden Afouri, Noam Siegel

January 10, 2021

Project Report: Neural Network from Scratch

This software was developed for the course Practical Deep Learning, BGU, January 2020. The code is available on GitHub.

In this project we integrated the different parts of the Artical Neural Network framework which have been discussed in the course.

We present the results of the gradient tests and the accuracy plots for various learning-rates/mb sizes.

Preliminaries

First, go ahead and clone the repository:

```
$ git clone https://github.com/noamsgl/PDL201_HW1
```

Project Structure
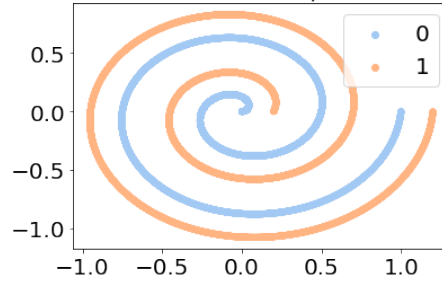
Structure your project as shown:

```
project_name
 NNdata
    GMMData.mat
    PeaksData.mat
    SwissRollData.mat
 utils.py
 tests.py
 parts123.py
 network.py
 Report.ipynb
```

Tasks 1-3

Execute SGD: zero hidden layers
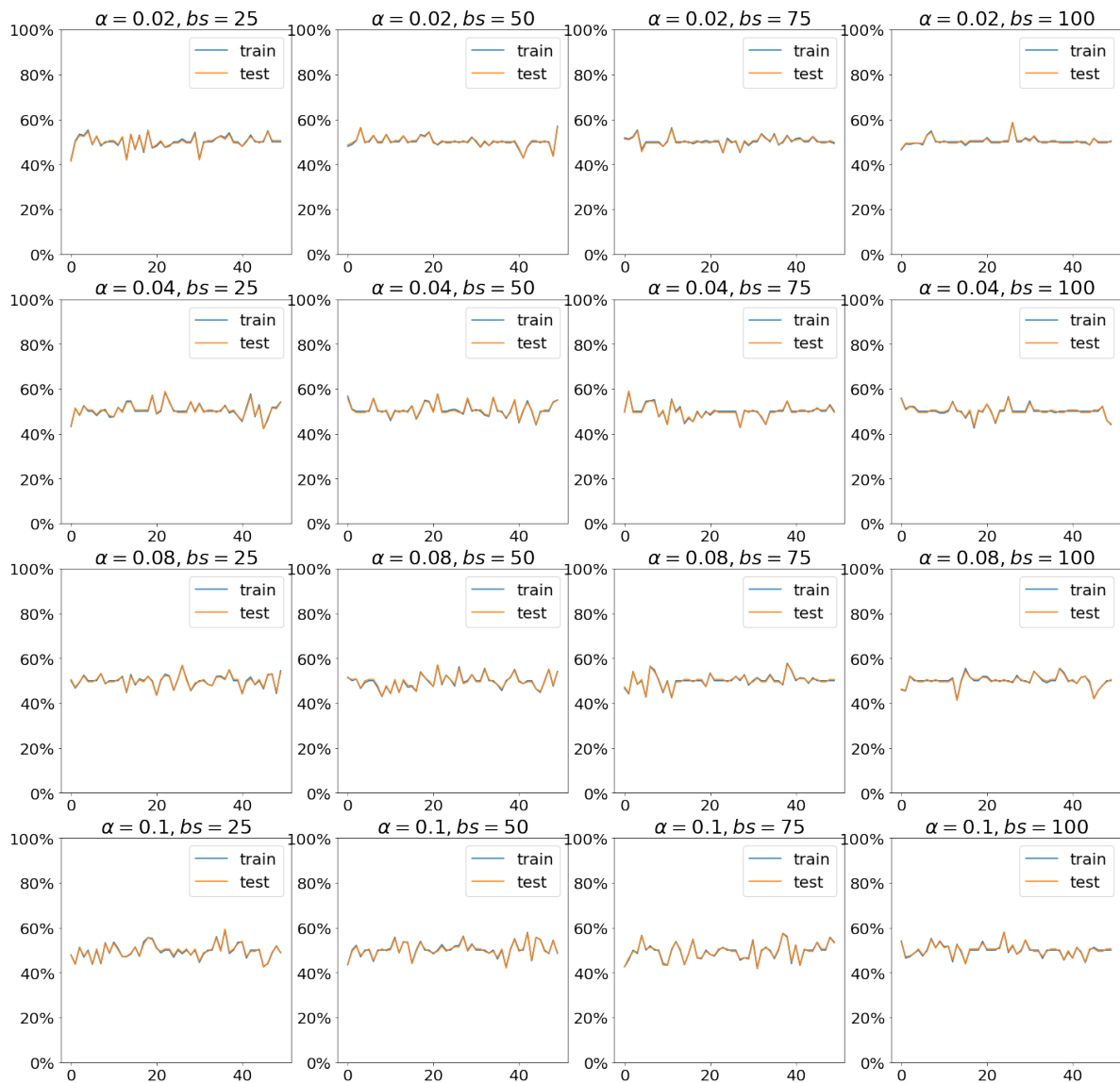
```
[9]: from parts123 import run_part_3
     run_part_3(learning_rates=(0.02, 0.04, 0.08, 0.1),
                batch_sizes=(25, 50, 75, 100),
                iters=50)
```

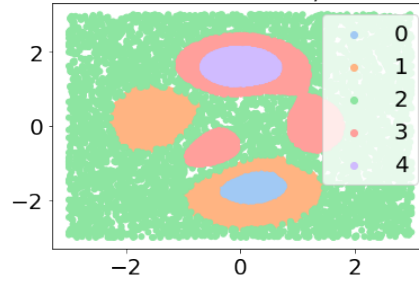2d visualization of NNdata/SwissRollData.mat

Accuracy vs. Epoch
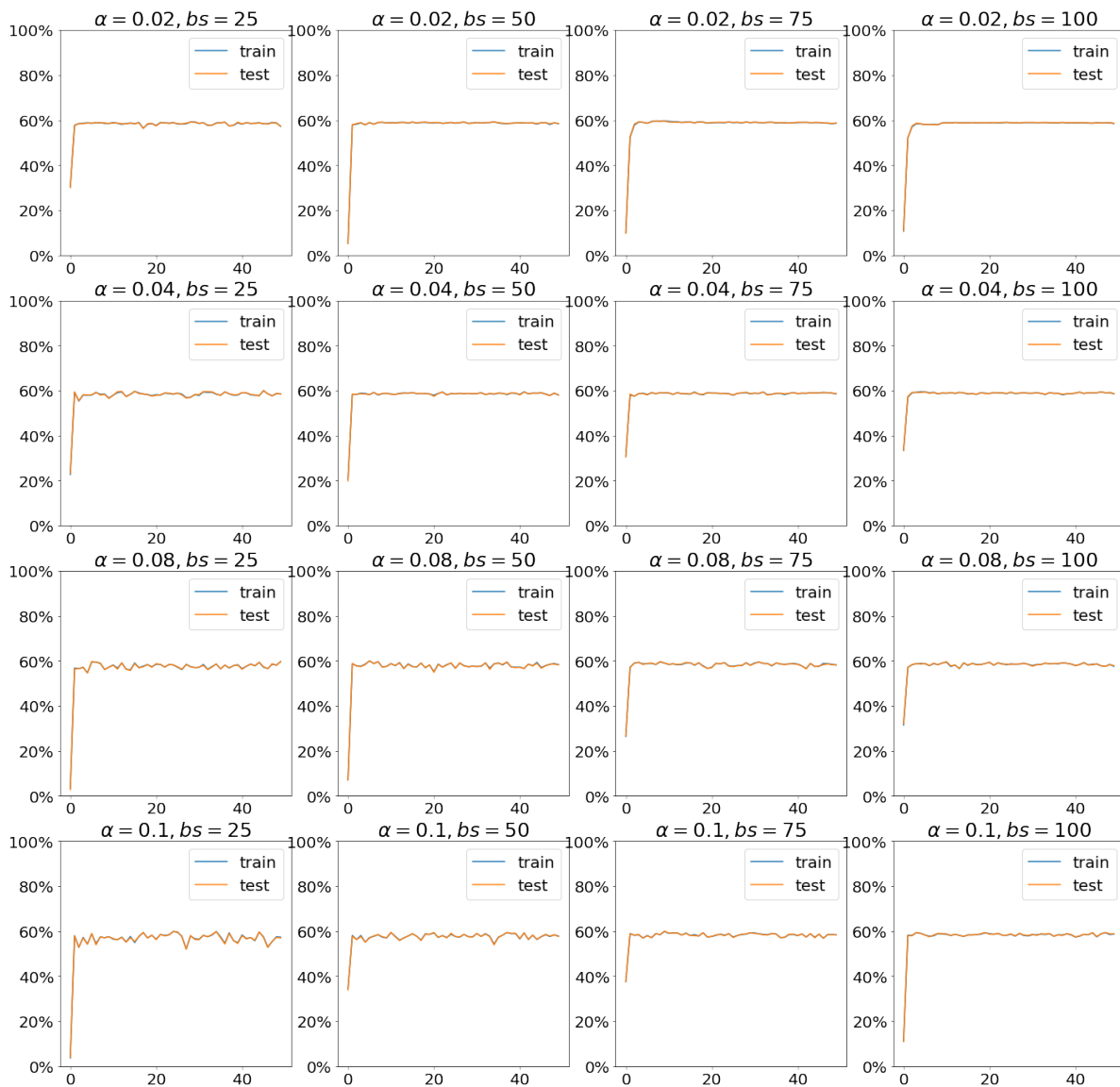Dataset: NNdata/SwissRollData.mat, Network Layers: []

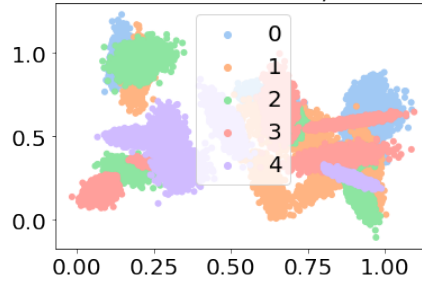## 2d visualization of NNdata/PeaksData.mat

Accuracy vs. Epoch
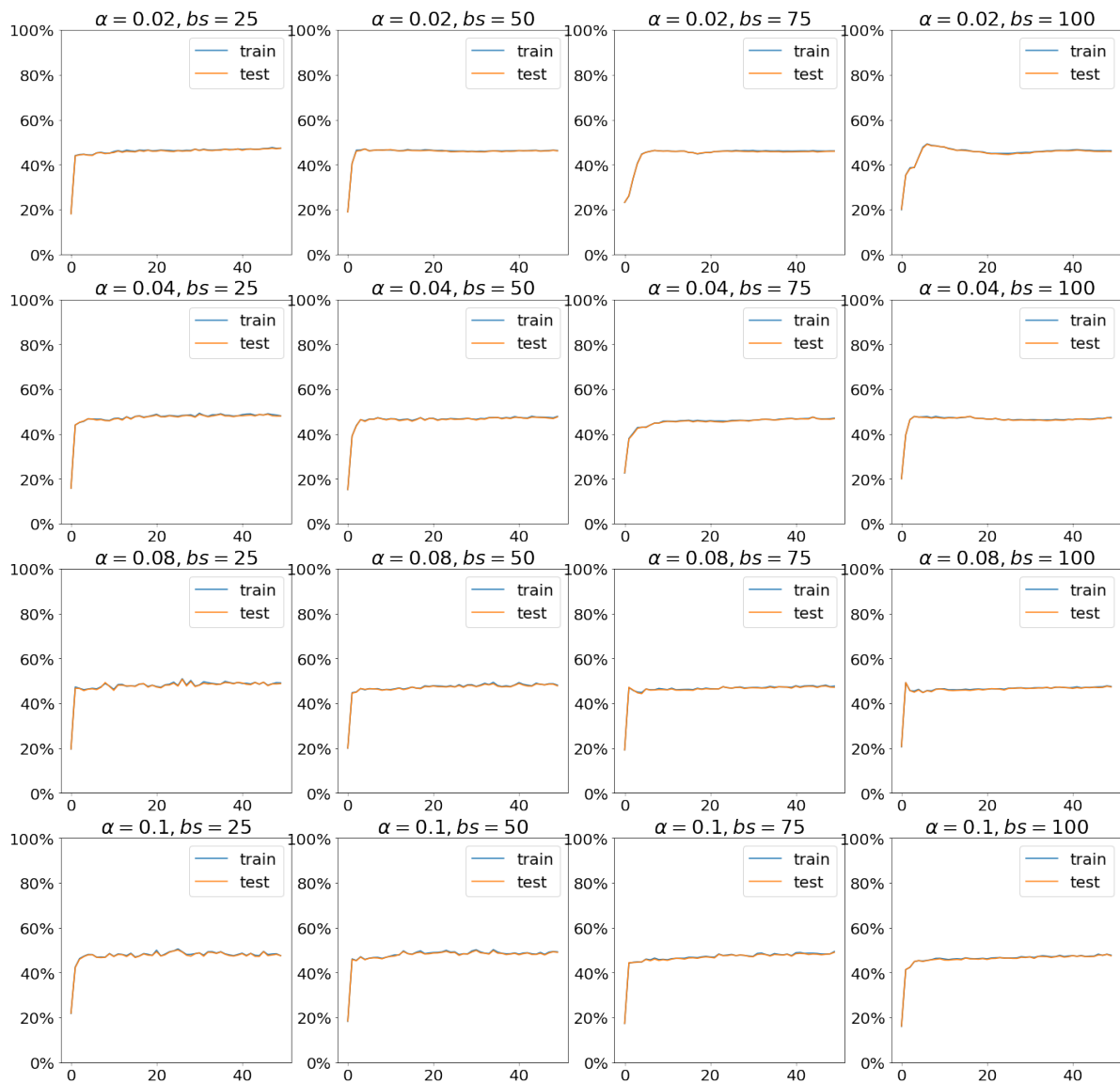## Dataset: NNdata/PeaksData.mat, Network Layers: []

2d visualization of NNdata/GMMData.mat

Accuracy vs. Epoch
## Dataset: NNdata/GMMData.mat, Network Layers: []

Task 4

Jacobian Tests: one hidden layer

We perform the Jacobian test on $\frac{\partial F^T}{\partial w_1}\epsilon d$, $\frac{\partial F^T}{\partial b_l}\epsilon d$ for $l = 1$.

```
[10]: from tests import gradient_test_for_hidden_layer_jacobian
      input_dim = 4
      output_dim = 4
      hidden_layers_for_tests = [6]
      gradient_test_for_hidden_layer_jacobian(input_dim, output_dim, samples=1,␣
       ↪hidden_layers_sizes=hidden_layers_for_tests,
                                              iters=20, optimizer='None')
```


Verification Test: JacMV(x)

Network Layers: [4, 6, 4], Parameter: w_1

Verification Test: JacMV(x)
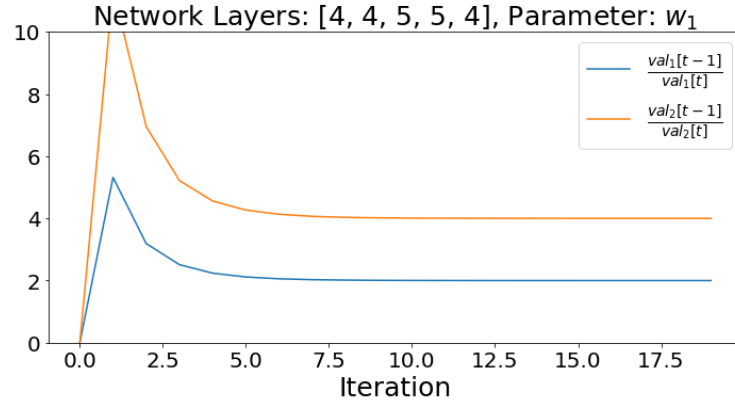
Network Layers: [4, 6, 4], Parameter: b_1
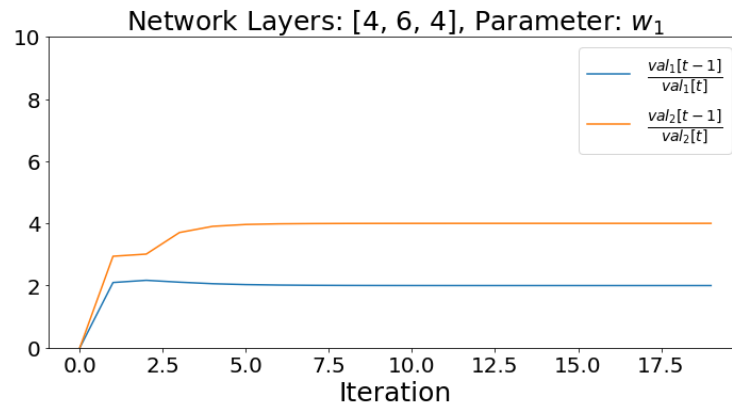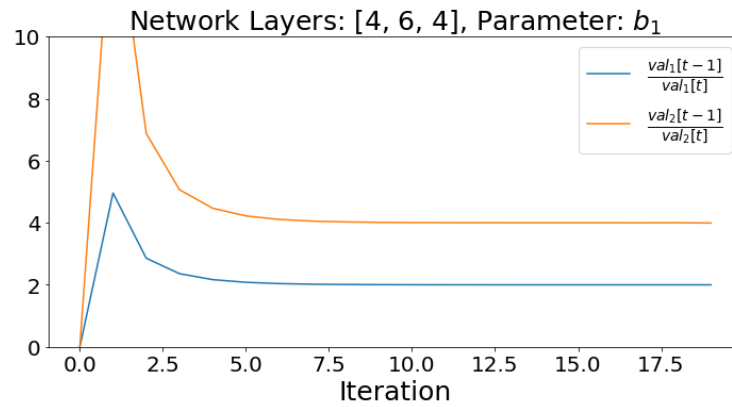
Jacboian Tests: several hidden layers

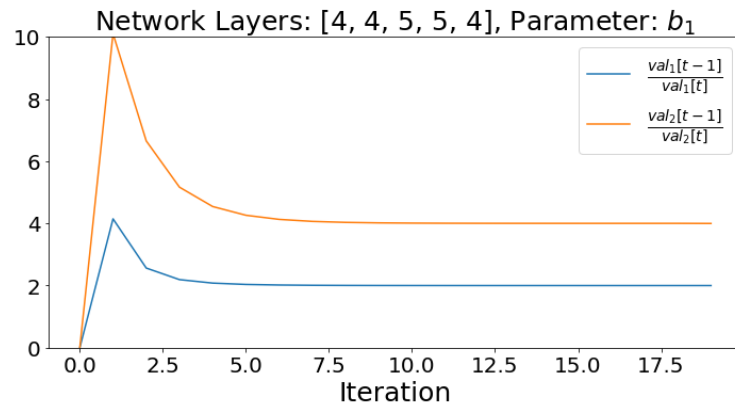We perform the Jacobian test on $\frac{\partial F^T}{\partial w_l}\epsilon d$, $\frac{\partial F^T}{\partial b_l}\epsilon d$ for all layers $l \in \{1, ..., L\}$.

```
[11]: hidden_layers_for_tests = [4, 5, 5]
      gradient_test_for_hidden_layer_jacobian(input_dim, output_dim, samples=1,
                                              =␣
       ↪hidden_layers_sizes=hidden_layers_for_tests, iters=20, optimizer='None')
```
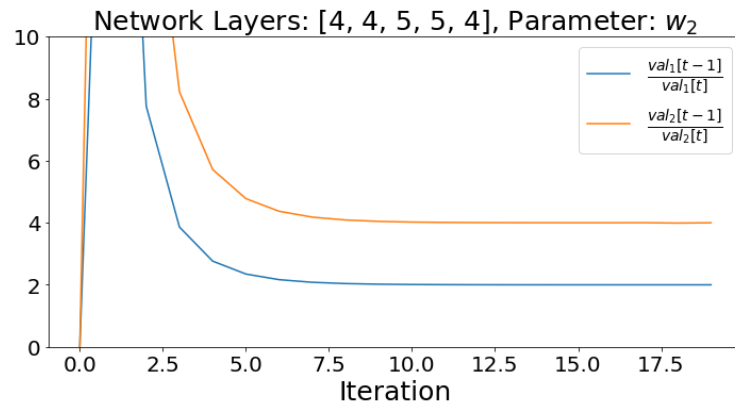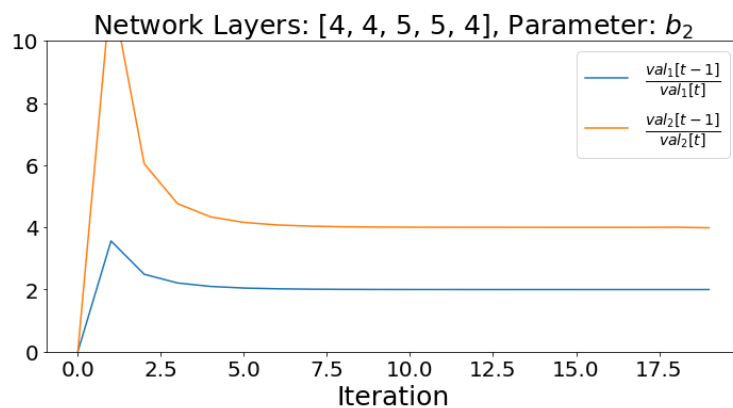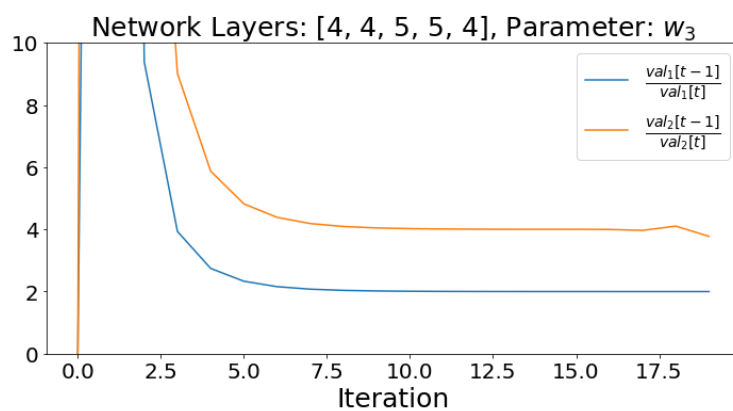
5

Verification Test: JacMV(x)

Network Layers: [4, 4, 5, 5, 4], Parameter: w_1

Verification Test: JacMV(x)

Network Layers: [4, 4, 5, 5, 4], Parameter: b_1

Verification Test: JacMV(x)

Network Layers: [4, 4, 5, 5, 4], Parameter: w_2

Verification Test: JacMV(x)

Network Layers: [4, 4, 5, 5, 4], Parameter: b_2

Verification Test: JacMV(x)

Network Layers: [4, 4, 5, 5, 4], Parameter: w_3

Verification Test: JacMV(x)

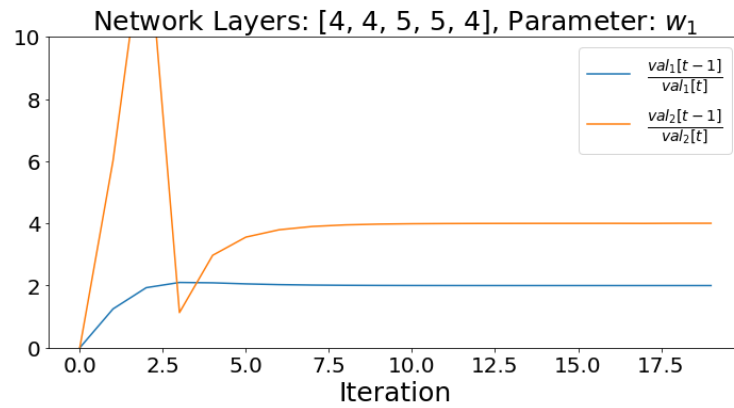### Network Layers: [4, 4, 5, 5, 4], Parameter: b_3
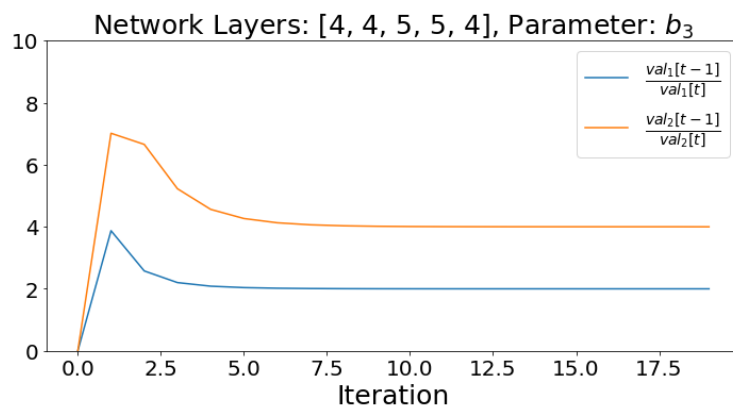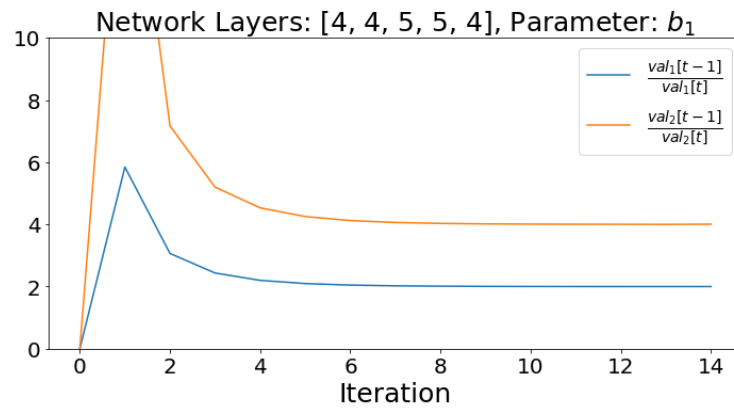
Task 6

Gradeient Tests: several hidden layer

We perform the gradient test on $\frac{\partial F}{\partial w_l}, \frac{\partial F}{\partial b_l}$ for all layers $l \in \{1, ..., L\}$.

```
[12]:  from tests import gradient_test_for_all_params
       gradient_test_for_all_params(input_dim, output_dim, samples=1,
                           hidden_layers_sizes=hidden_layers_for_tests, iters=20,␣
        →optimizer='None')
```
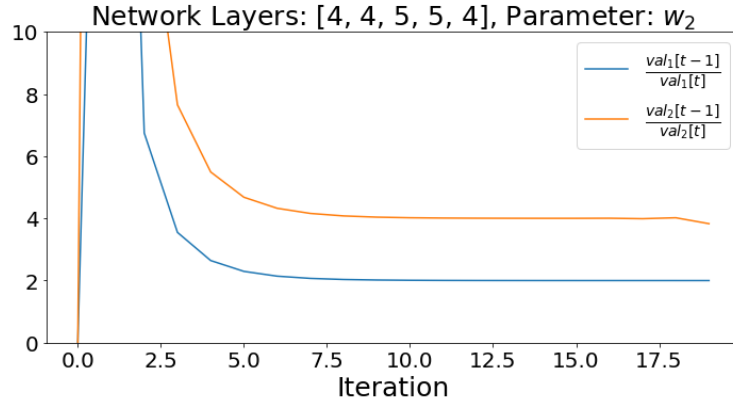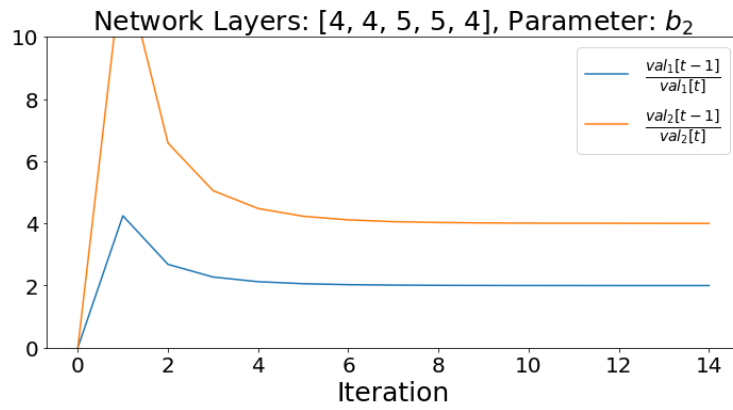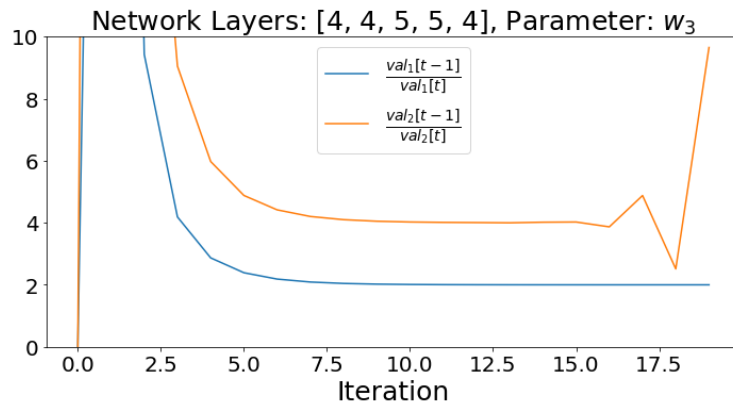
Verification Test: grad(x)

Network Layers: [4, 4, 5, 5, 4], Parameter: w_2

Verification Test: grad(x)

Network Layers: [4, 4, 5, 5, 4], Parameter: b_2

Verification Test: grad(x)

Network Layers: [4, 4, 5, 5, 4], Parameter: w_3

Verification Test: grad(x)

Network Layers: [4, 4, 5, 5, 4], Parameter: b_3

Verification Test: grad(x)

Network Layers: [4, 4, 5, 5, 4], Parameter: w_4

Verification Test: grad(x)

### Network Layers: [4, 4, 5, 5, 4], Parameter: b_4
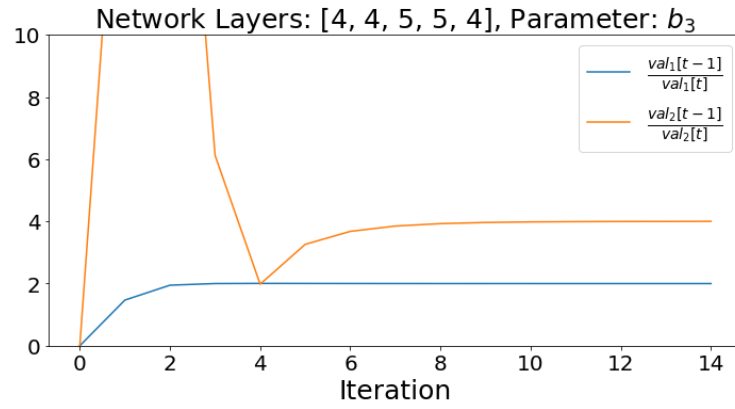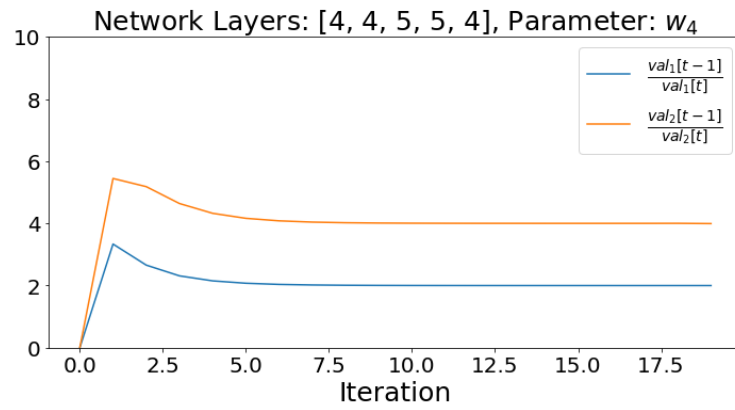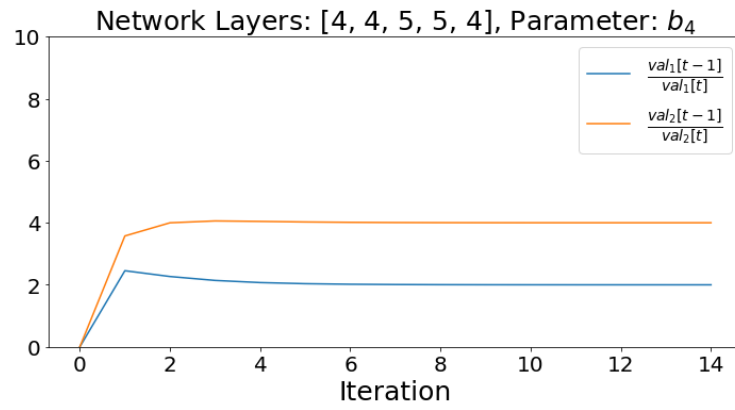
Task 7

Execute SGD: several hidden layers

We repeat Task 3 for a multi-layer network:
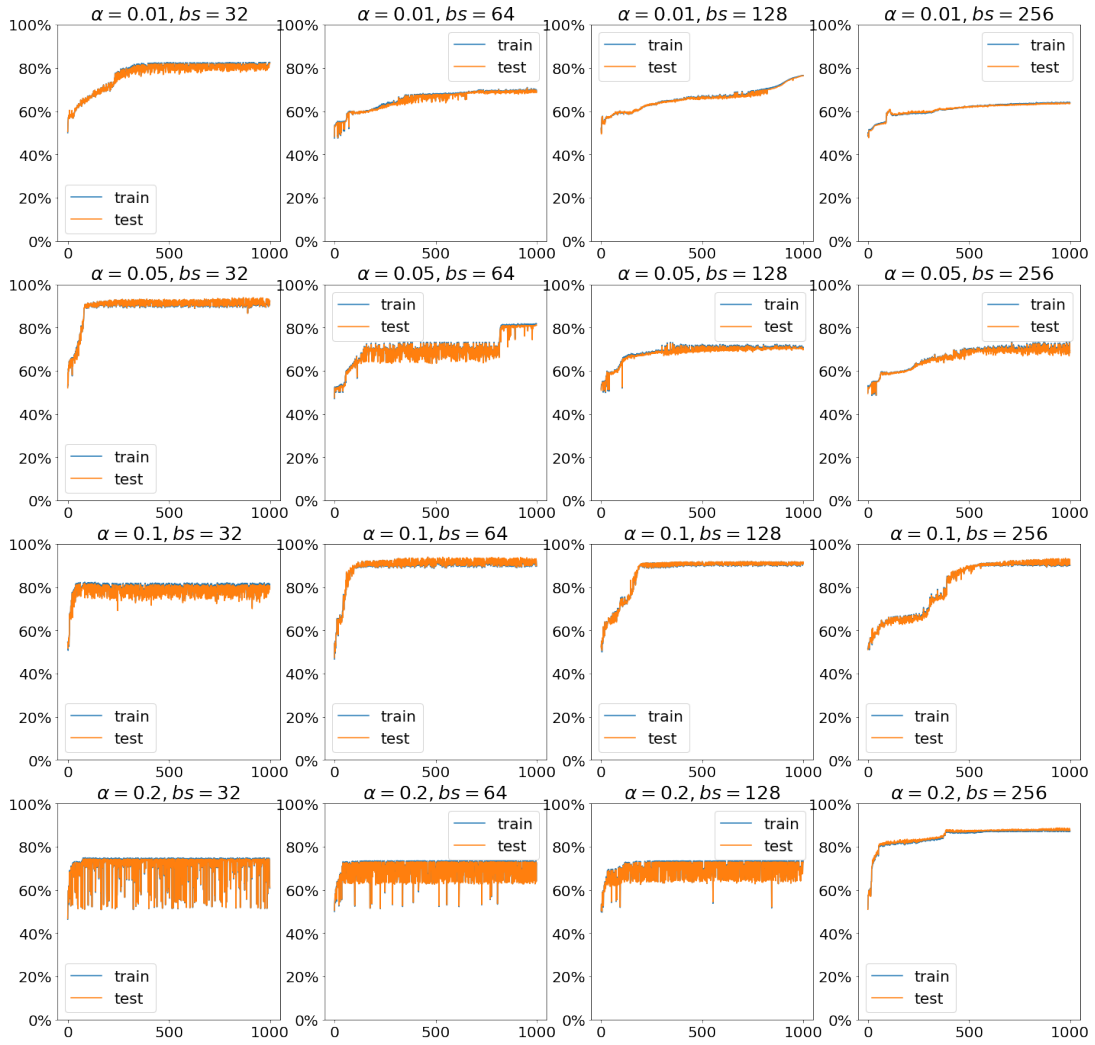
```
[13]:  HIDDEN_LAYERS_FOR_DATASET = {
           'NNdata/SwissRollData.mat': [20, 10],
           'NNdata/PeaksData.mat': [6, 10],
           'NNdata/GMMData.mat': [20, 10, 5]
       }
       run_part_3(learning_rates=(0.01, 0.05, 0.1, 0.2),
                  batch_sizes=(32, 64, 128, 256),
                  hidden_layers_for_dataset=HIDDEN_LAYERS_FOR_DATASET,
                  iters=1000)
```
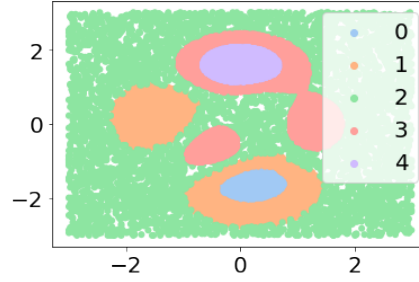
2d visualization of NNdata/SwissRollData.mat
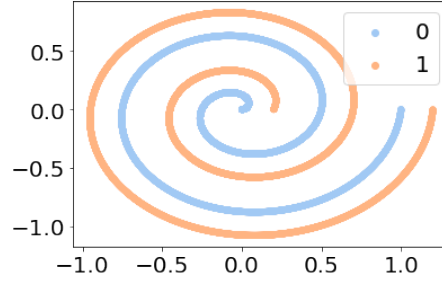
Accuracy vs. Epoch
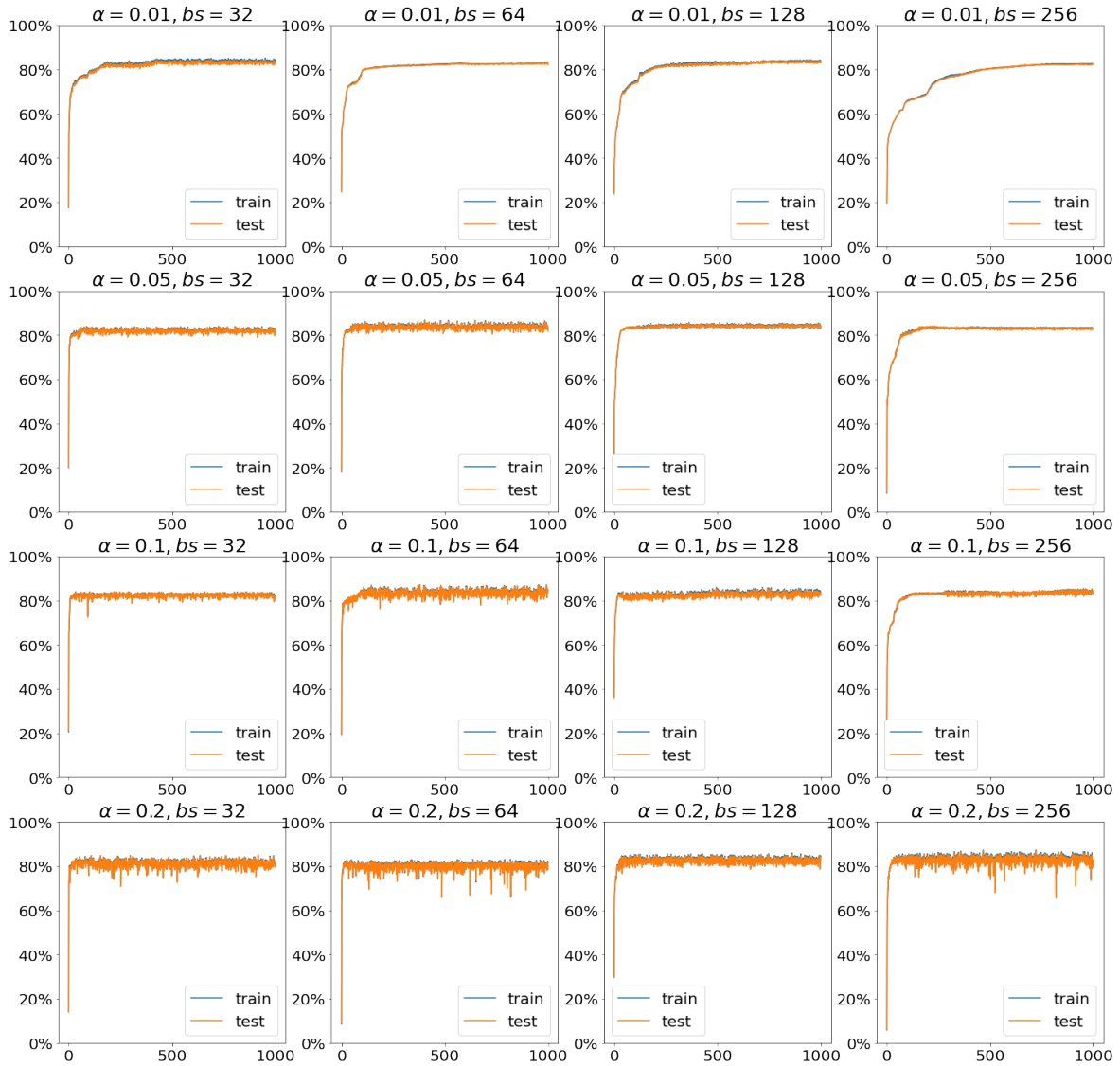## Dataset: NNdata/SwissRollData.mat, Network Layers: [20, 10]
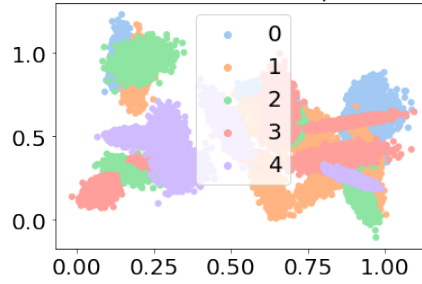
2d visualization of NNdata/PeaksData.mat

Accuracy vs. Epoch
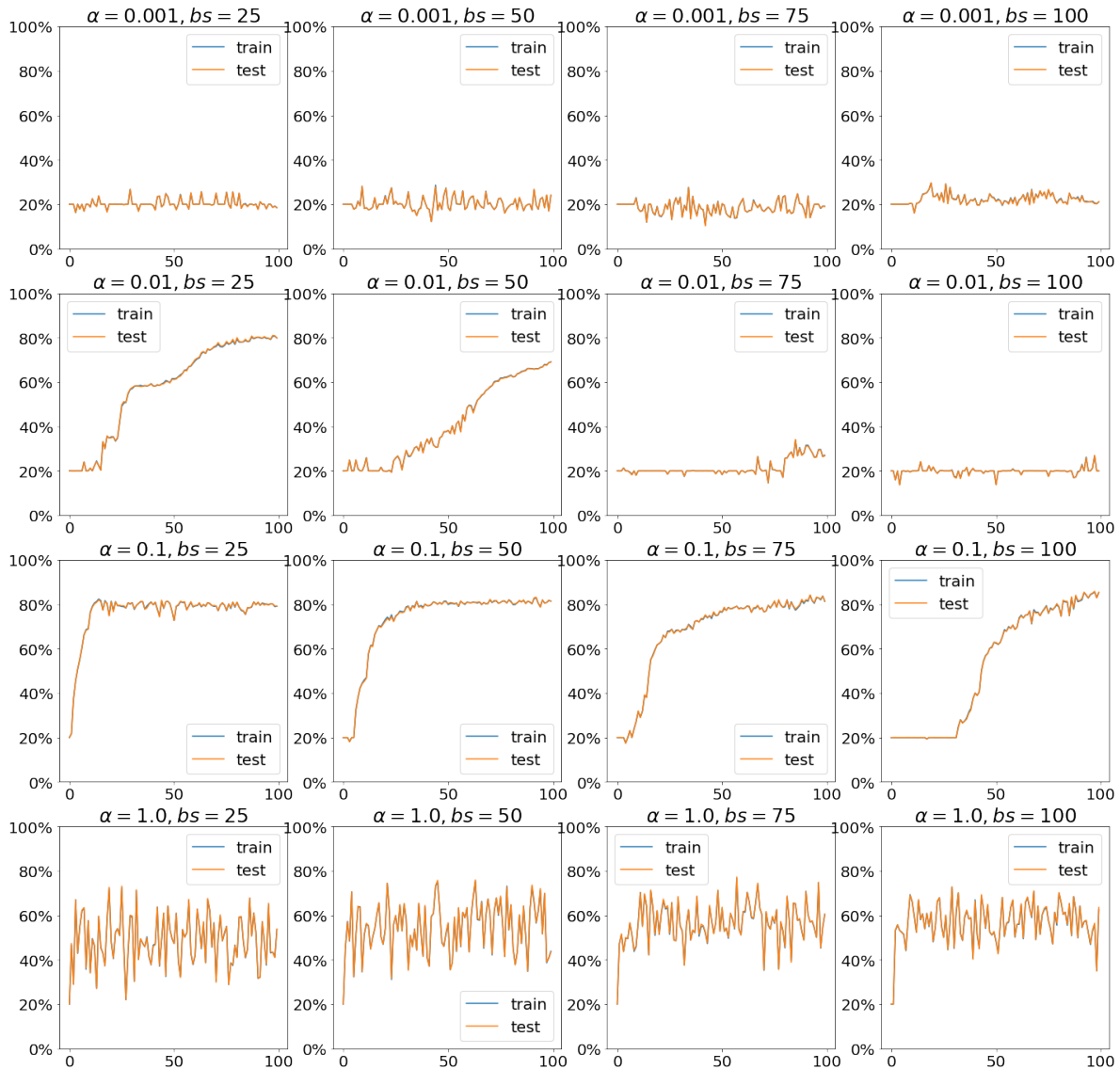# Dataset: NNdata/PeaksData.mat, Network Layers: [6, 10]

2d visualization of NNdata/GMMData.mat

Accuracy vs. Epoch
Dataset: NNdata/GMMData.mat, Network Layers: [6, 10]
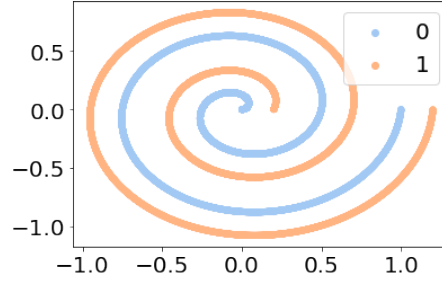
Ablation Study: SGD with momentum

We run SGD with momentum:
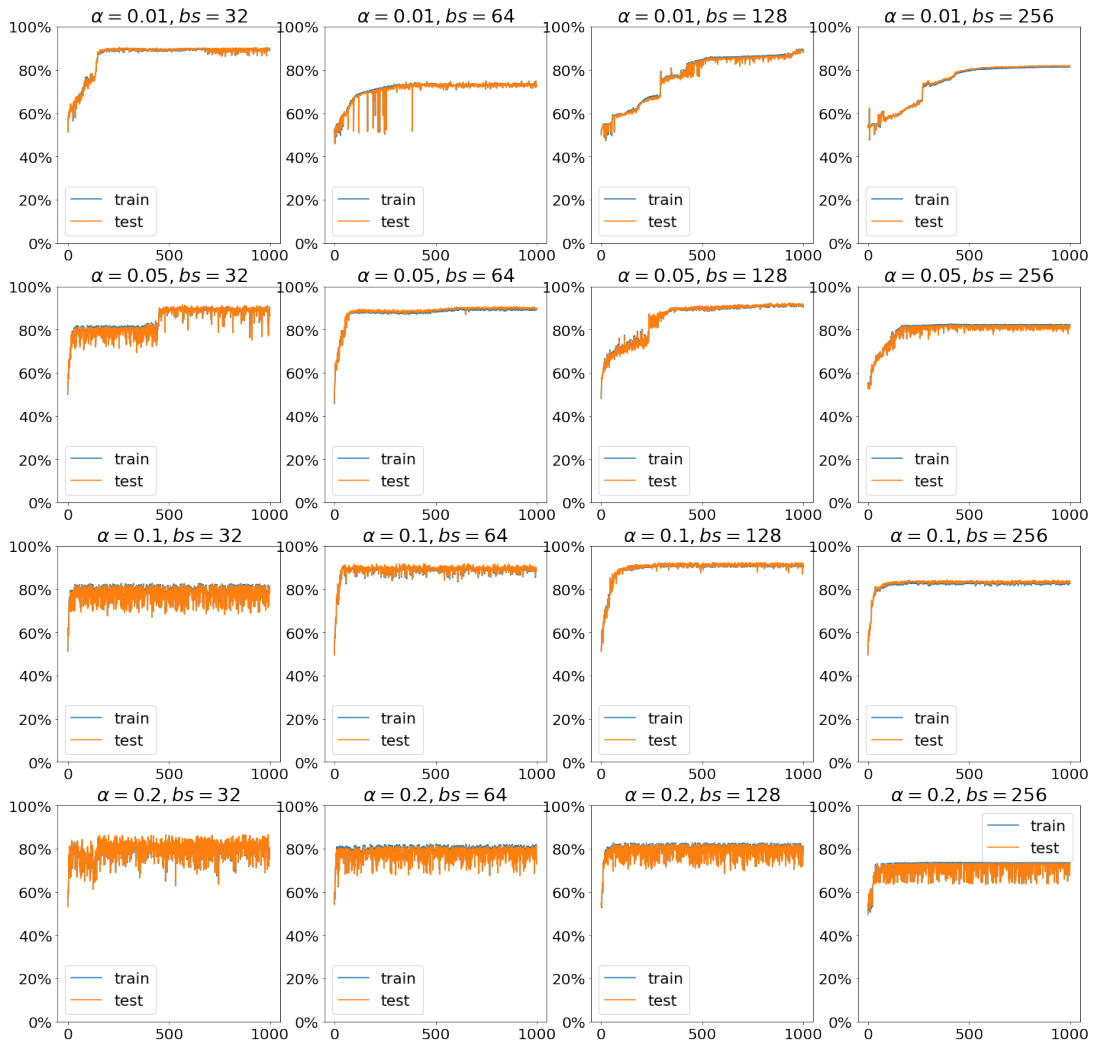
```
[14]: run_part_3(learning_rates=(0.01, 0.05, 0.1, 0.2),
                 batch_sizes=(32, 64, 128, 256),
                 hidden_layers_for_dataset=HIDDEN_LAYERS_FOR_DATASET,
                 iters=1000,
                 optimizer='momentum',
                 gamma=0.7)
```
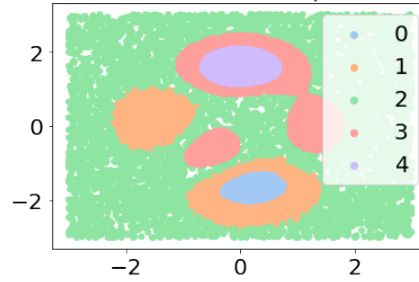
2d visualization of NNdata/SwissRollData.mat

Accuracy vs. Epoch
Dataset: NNdata/SwissRollData.mat, Network Layers: [20, 10]
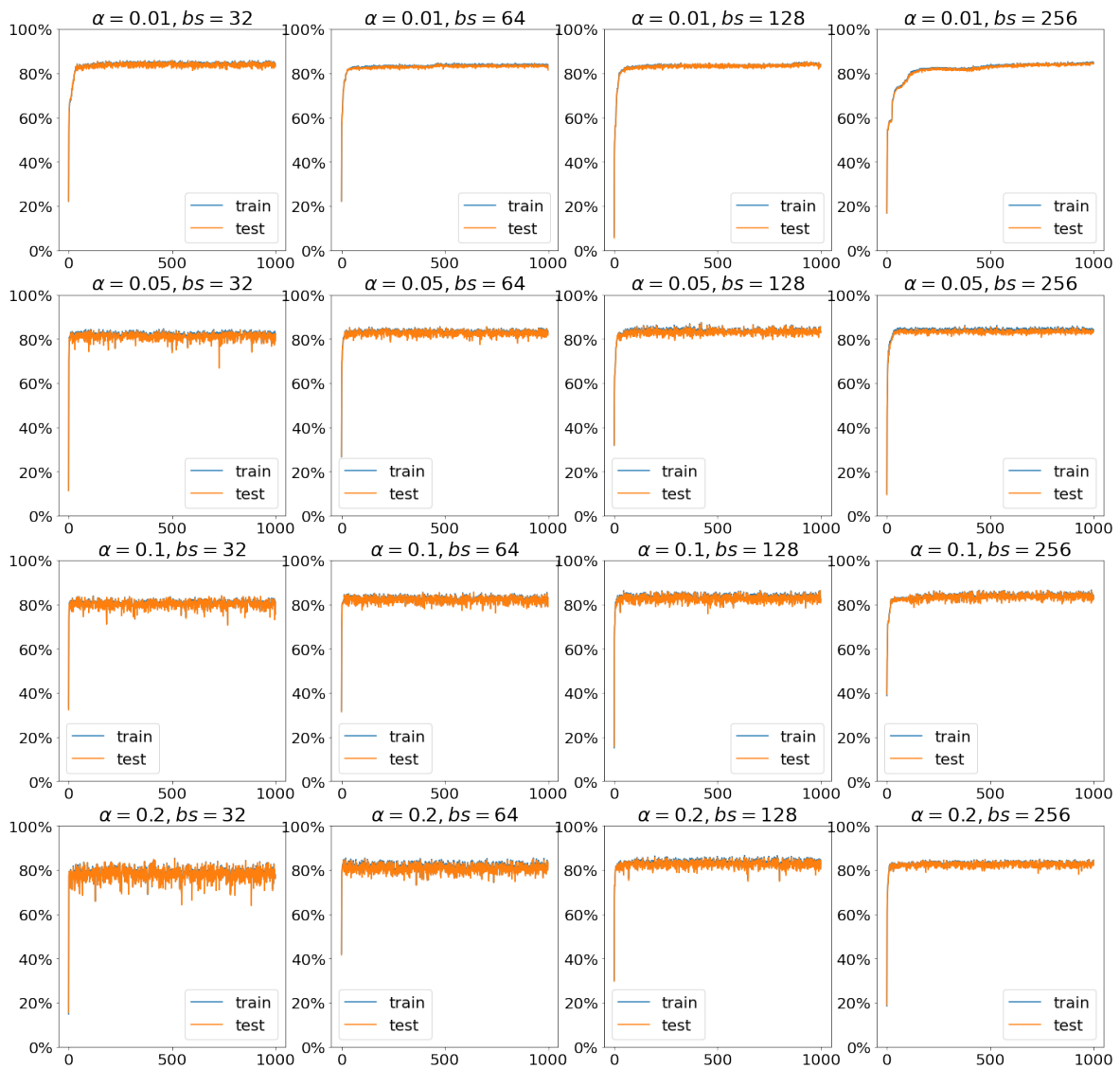
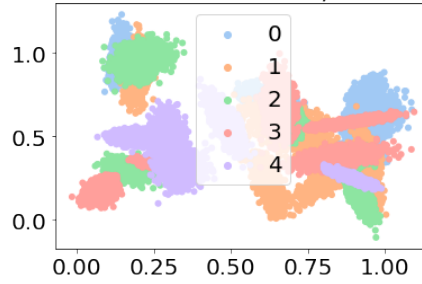## 2d visualization of NNdata/PeaksData.mat

Accuracy vs. Epoch
Dataset: NNdata/PeaksData.mat, Network Layers: [6, 10]

2d visualization of NNdata/GMMData.mat

Accuracy vs. Epoch
Dataset: NNdata/GMMData.mat, Network Layers: [6, 10]