CAP 6615 - Neural Networks - Programming Assignment 1 – Single-Layer Perceptron

Zhounan Li, Huaiyue Peng, Tongjia Guo, Zhiyun Ling, Mingjun Yu

Spring Semester 2021
29 Jan 2021

1 Network parameters

In this project, we choose approach 2. We have 16x16 = 256 inputs, and n = 256x20 = 5,120 weights, and 20 output nodes that are valued vary over the interval [0,1]. The governing equation for the j-th output node is

$$y_j = f(\sum_{i=1}^{i=n}(x_i \cdot w_{ij} + b_i)) \tag{1}$$

Our loss function is

$$L = \sum_{i=1}^{i=n}(y_i log\hat{y}_i + (1 - y_i)log(1 - \hat{y}_i)) \tag{2}$$

The index of the output pattern is found by locating the maximum value in y. For example, if the maximum element is at index 1. Then we think this image belongs to class 1.

2 Python code for SLP

```
import numpy as np
import pandas as pd
import os
from PIL import Image
import tensorflow as tf
from tensorflow.keras.layers import *
import tensorflow.keras.backend as K
import matplotlib.pyplot as plt
import argparse

def get_model_approach_2():
    img_input = tf.keras.Input(shape=(256, ))
```

```
    output = Dense(20, activation='softmax')(img_input)
    model = tf.keras.Model(img_input, output)
    model.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=[
    return model
```

3 Training set configuration

We choose twenty images A, B, C,..., J and 1, 2, 3,..., 0 to be our dataset. In order to get better performance, we choose A, B, C, D, E, I, 3, 5, 6 and 7 as our train dataset because they are similar to the remaining images, for example, I is similar to 1 and D is similar to 0. Then we resize these pictures in 16*16 pixels images.

Figure 1: The dataset

Figure 2: The resized train dataset

Our code for building dataset:

```
#change RGB image to Gray
from PIL import Image
import glob
filenames = glob.glob('*.jpg')
print(filenames)
# img = Image.open('image.png').convert('LA')
for file in filenames:
        img = Image.open(file).convert('LA')
        img.save('gray'+file[0] + '.png')


#resize image
from PIL import Image
import glob
filenames = glob.glob('Gray/*.png')
print(filenames)

for file in filenames:
        img = Image.open(file).resize((16, 16))
        img.save(file)
```

```
#resize image
from PIL import Image
import glob
filenames = glob.glob('Gray/*.png')
print(filenames)

for file in filenames:
        img = Image.open(file).resize((16, 16))
        img.save(file)



#change gray to binary
from PIL import Image
import glob
import numpy as np
filenames = glob.glob('Gray/*.png')
print(filenames)

for file in filenames:
        arr = np.array(Image.open(file))[:, :, 0]
        for i in range(16):
                for j in range(16):
                        arr[i][j] = 255 if arr[i][j] > 128 else 0

        img = Image.fromarray(arr)
        # img.show()
        img.save(file)
```

4 SLP output results for noiseless input

After building our own dataset, we use out model to train the images we choose in last section. After training we use the following equations to calculate Fh and Ffa.

$$Fh = \frac{number\ of\ black\ pixels\ in\ output\ image\ that\ occur\ in\ correct\ places}{total\ number\ of\ black\ pixels\ in\ input\ image} \tag{3}$$

$$Fh = \frac{number\ of\ black\ pixels\ in\ output\ image\ that\ occur\ in\ wrong\ places}{total\ number\ of\ white\ pixels\ in\ input\ image} \tag{4}$$

After training, we get our loss function and the accuracy of our model. In this section, we set epoches to 500 because the loss function reaches convergence when epoch is 500. The

accuracy reaches 1 when epoch reaches 500, which means our model has very high accuracy on the train dataset.

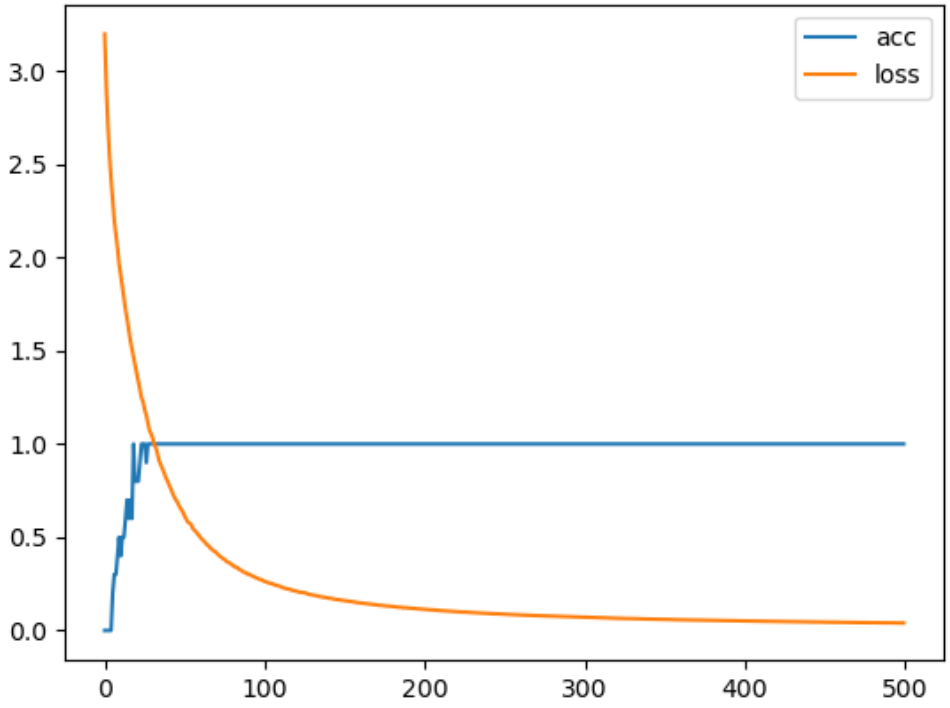

Figure 3: The train results of noiseless data

image	I	D	E	B	6	C	A	7	5	3
Fh	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Ffa	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
image	0	1	2	4	8	9	F	G	H	J
Fh	0.71	0.92	0.55	0.53	0.60	0.78	1.00	0.73	0.85	0.67
Ffa	0.29	0.16	0.35	0.44	0.21	0.32	0.15	0.02	0.26	0.40

Figure 4: The Fh and Ffa of noiseless data

But the test accuracy is just 0.5. This is because we only choose 10 images in our train dataset. So we can identify these images very well and the accuracy of train dataset is 1. But we can't classify the other ten images and the accuracy of the other ten images is 0.

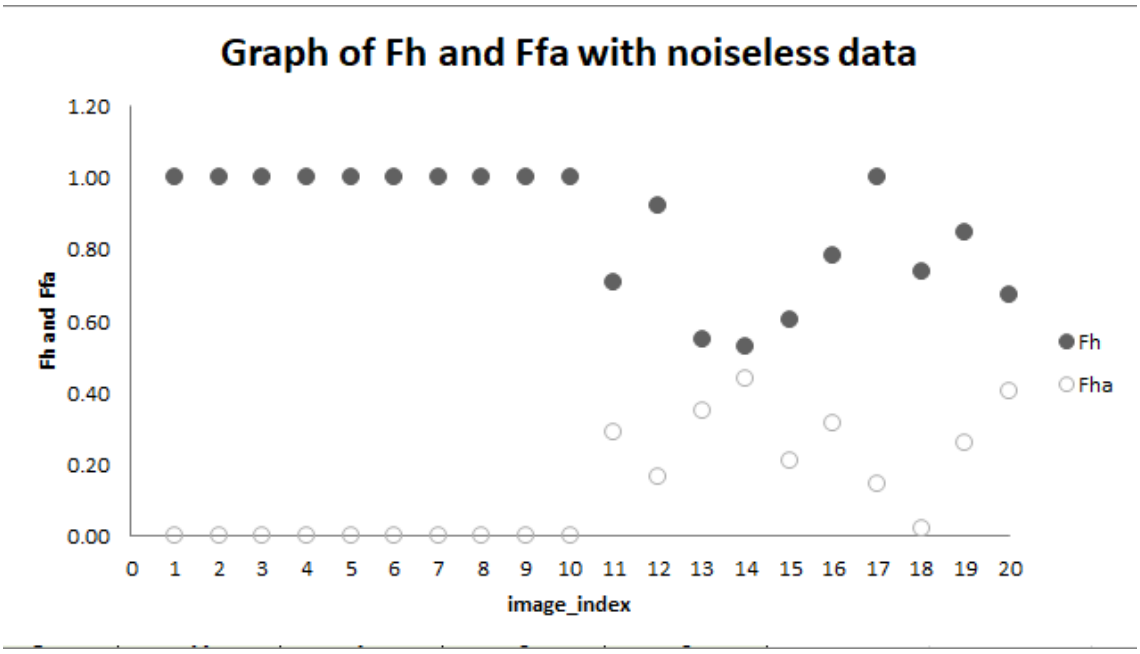

Figure 5: The Fh and Ffa of noiseless data in graph

5 Code for computing Fh and Ffa

```
# get Fh and Fha table by the way
    Fh = {}
    Fha = {}

    for i, file in enumerate(filenames):
        img = Image.open(os.path.join(dir, file))
        arr = np.reshape(np.array(img)/255, (256,)).astype('float64').tolist(
        dataset_x.append(arr)
        label = [0] * 20
        label[i] = 1
        dataset_y.append(label)
        label = tuple(label)
        onehot2label[label] = file[4]
        index2label[i] = file[4]

    # get Fh and Fha
    for i in range(20):
        for j in range(20):
            f1 = filenames[i][4]
            f2 = filenames[j][4]
            arr1 = np.array(dataset_x[i]).astype('int')
            arr2 = np.array(dataset_x[j]).astype('int')
```

```
            hit = np.sum((1-arr1) & (1-arr2))
            total = np.sum(1-arr1)
            pred = np.sum(1-arr2)
            Fh[f1+f2] = hit / total
            Fha[f1+f2] = (pred-hit) / pred

#the code for graph Fh and Ffa
from matplotlib import pyplot as plt
import xlrd
import numpy as np

f = xlrd.open_workbook("fh.xls")
sheet = f.sheet_by_index(0)
data = []
for i in range(7,27):
    one_data = []
    for j in range(18):
        one_data.append(round(sheet.cell_value(i,j+3),2))
    data.append(one_data)

data = np.array(data)
data = np.transpose(data)
data = data.tolist()

x = [0.001, 0.002, 0.003, 0.005, 0.01, 0.02, 0.03, 0.05, 0.1]
y = []
for i in range(0,18,2):
    print(i)
    y.append(data[i]+data[i+1])

for xe, ye in zip(x, y):
    print(ye)
    plt.scatter([xe] * 20, ye[:20], facecolors='b',edgecolors='b')
    plt.scatter([xe] * 20, ye[20:],facecolors='none',edgecolors='b')
plt.xlabel("Gaussian Noise Level")
plt.ylabel("Fh and Ffa")
plt.title("Graph of Fh and Ffa with noise data")
plt.xscale("log")
plt.xticks(x)
plt.savefig('t.png')
```

6 SLP output results for noise-corrupted input

After training with noiseless data, we add noise to the input data. Noise is Gaussian-distributed with 10 percent cross-section(i.e. There are 25 noise pixels in this project because there are 256 pixels in input data.) and have zero mean, and standard deviation of 0.001, 0.002, 0.003, 0.005, 0.01, 0.02, 0.03, 0.05, and 0.1. From the graph, we can know that the

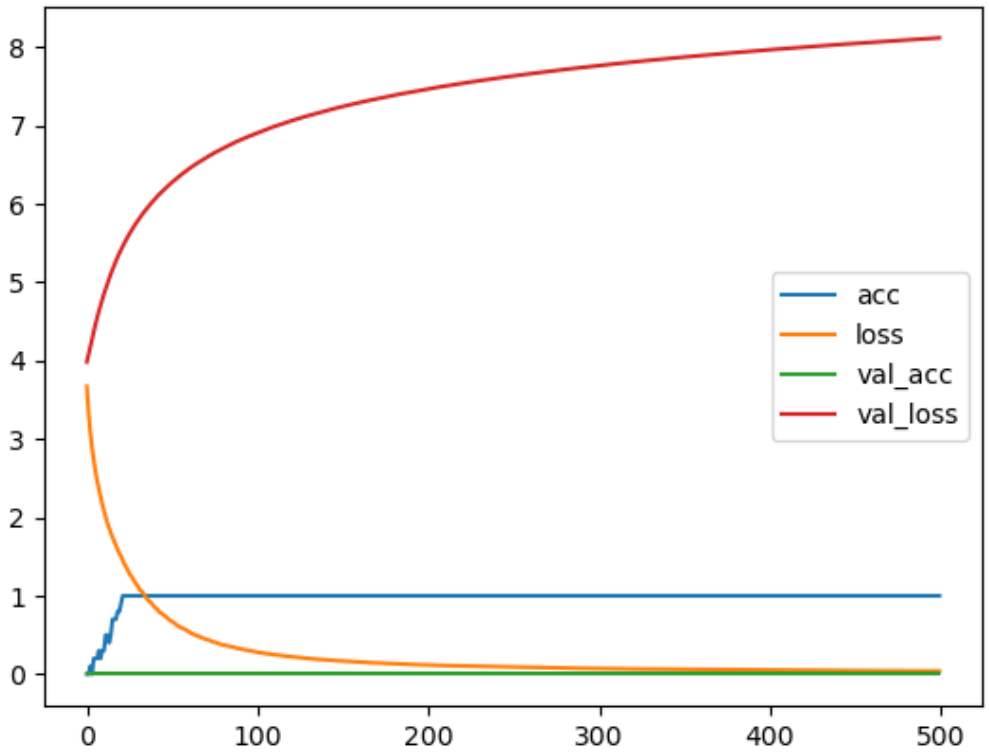

Figure 6: The train and test results of noise data in graph

noise has no effect on our train dataset because the noise is very small. The image with noise is very similar to original image. The accuracy of train dataset is 1 and 0.5 for test dataset.we can't classify the other 10 images correctly because we don't put them in our train dataset. The test loss becomes bigger because the test result is very bad on the other 10 images.

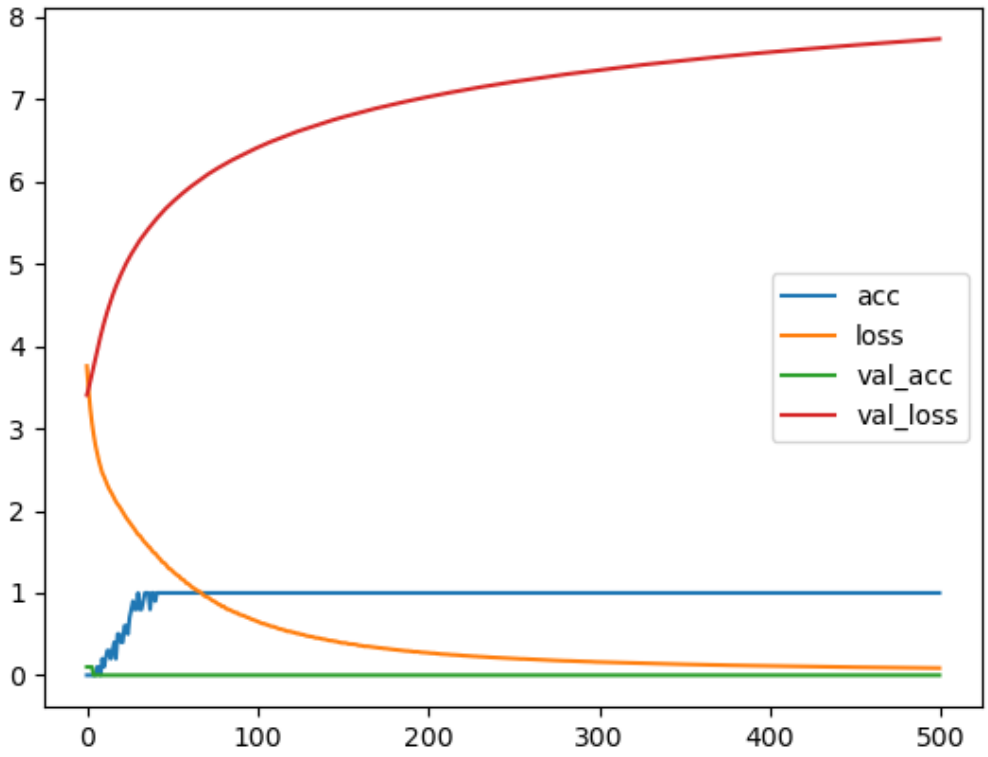

Figure 7: The train and test results of noise data in graph

The noise is 0.1 in this process. We can see that the train process will spend more to reach 1 accuracy because the noise is big enough to make the images different from the original images and we need more time to train them.

number of inputs = 16 * 16 =256
number of weights = 256 *20 = 5120
number of outputs = 20 * 1 = 20

std	0		0.001		0.002		0.003		0.005		0.01		0.02		0.03		0.05		0.1	
	Fh	Ffa	Fh	Ffa	Fh	Ffa	Fh	Ffa	Fh	Ffa	Fh	Ffa	Fh	Ffa	Fh	Ffa	Fh	Ffa	Fh	Ffa
I	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
D	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
E	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
B	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
6	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
C	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
A	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
7	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
5	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
3	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
0	0.71	0.29	0.71	0.29	0.71	0.29	0.71	0.29	0.71	0.29	0.71	0.29	0.59	0.30	0.59	0.30	0.71	0.29	0.71	0.29
1	0.92	0.16	0.92	0.16	0.92	0.16	0.92	0.16	0.92	0.16	0.92	0.16	0.92	0.16	0.92	0.16	0.92	0.16	0.92	0.16
2	0.55	0.35	0.55	0.35	0.55	0.35	0.55	0.35	0.55	0.35	0.55	0.35	0.55	0.35	0.55	0.35	0.55	0.35	0.51	0.53
4	0.53	0.44	0.27	0.60	0.53	0.44	0.34	0.56	0.53	0.44	0.33	0.56	0.53	0.44	0.53	0.44	0.53	0.44	0.53	0.44
8	0.60	0.21	0.60	0.21	0.60	0.21	0.60	0.21	0.60	0.21	0.60	0.21	0.60	0.21	0.60	0.21	0.60	0.21	0.60	0.21
9	0.78	0.32	0.78	0.32	0.78	0.32	0.78	0.32	0.78	0.32	0.78	0.32	0.78	0.32	0.78	0.32	0.43	0.42	0.78	0.32
F	1.00	0.15	1.00	0.15	1.00	0.15	1.00	0.15	1.00	0.15	1.00	0.15	1.00	0.15	1.00	0.15	1.00	0.15	1.00	0.15
G	0.73	0.02	0.73	0.02	0.73	0.02	0.73	0.02	0.73	0.02	0.73	0.02	0.73	0.02	0.73	0.02	0.73	0.02	0.73	0.02
H	0.85	0.26	0.85	0.26	0.85	0.26	0.85	0.26	0.85	0.26	0.85	0.26	0.85	0.26	0.85	0.26	0.85	0.26	0.85	0.26
J	0.67	0.40	0.67	0.40	0.67	0.40	0.67	0.40	0.67	0.40	0.67	0.40	0.67	0.40	0.67	0.40	0.67	0.40	0.67	0.40

Figure 8: The train and test results of noise data with different std

From the above results, we can see that our model performs well, especially on train dataset. Fh is always 1 and Ffa is 0. But as to other 10 images. The Fh is bigger than 0.5 and Ffa is smaller than 0.5.

7 Discussion

From the above results, we know that our model performs very well on train dataset. But as to test dataset, it doesn't perform very well. There are two main reasons. First, the test data is different from train data. They are disjoint. The model knows how to identify A, B, C but it doesn't know how to identify F, G, H. So if we want to get better performance on test dataset. We should include all images in our train dataset. In this project, 20 image is enough. The second reason is that the train dataset is so small that it causes over-fitting. We don't have enough data to train out model. So our model performs well on train dataset. We have two methods to solve this problem. First we can extend out train dataset. Add noise to the original dataset and use both the original dataset and the dataset with noise. Because if we have more train data, we have more information about all images and we may get better performance. The second method is adding a regularization in our model to avoid over-fitting. This will make our model simpler and may get good performance on test dataset.

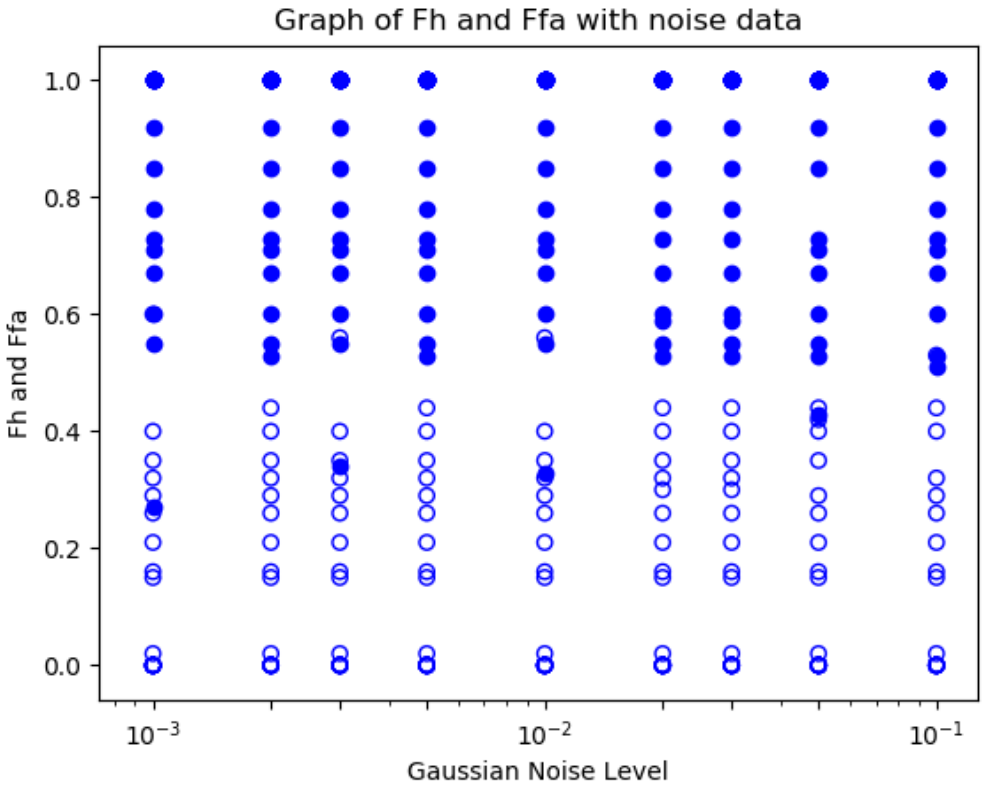

Figure 9: The train and test results of noise data with different std in graph