인공지능개론 정리노트 #5
ICT융합공학부 202204010 공성택

7장(하)

Q. cnn 신경망 과제를 진행하면서 계층이 많아질수록 오히려 정확도가 떨어지는 현상이 발생했는데 왜 그런지 궁금하다.

A. 신경망이 깊어질 수록 정확도가 떨어지는 것은 보통 과적합과 기울기 소실의 문제일 가능성이 크다. 하지만 과적합은 훈련 데이터는 학습이 잘 진행되는데 테스트 데이터가 훈련 데이터에 비해 정확도가 떨어지는 현상이다. 나는 훈련 데이터와 테스트 데이터 모두 학습이 잘 진행되지 않았던 경우이다. 따라서 과적합의 문제는 아니라고 생각한다. 조사한 결과, 과적합이 아니더라도 내가 계층을 늘렸을 때, 학습이 잘 되지 않은 이유로는 우선 학습률의 조정이 문제였을 수 있다. 학습률 조정을 잘못하여 더디게 진행된 것일 수도 있고, 또 다른 이유로는 훈련 데이터가 부족했을 수도 있다고 한다. 깊은 신경망은 더 많은 데이터를 필요로 한다고 하는데, 나의 경우 실제로 학습 시간을 줄이기 위해 데이터를 적게 사용하였다. 이 부분 때문에 정확도가 떨어졌을 수 있다고 생각한다.

Q. cnn 프로그램에 mnist 데이터를 줬을 때는 정확도가 매우 높게 나왔는데, 과제에서 fashion mnist 데이터를 줬을 때는 정확도가 비교적 낮게 나왔다. 왜 그런 것일까?

A. 조사한 결과, mnist에 비해 fashion mnist는 더 복잡하다고 한다. mnist는 비교적 잘 구별할 수 있는 '숫자'라는 이미지를 주제로 하지만, fashion minst는 의류, 악세서리 등 명확한 기준이 있는 것이 아니라 더 복잡한 패턴을 가진 데이터를 주제로 한다. 실제로 fashion mnist에는 셔츠, 코트 같이 비슷한 모양이지만 종류는 다른 것도 존재한다. 그렇기 때문에 신경망 모델이 더 학습하기 어려운 것이 사실이다. 따라서 이런 특징을 더 잘 파악하기 위해서는 계층을 늘려야 하는데, 무작정 늘리지 않고 적절한 하이퍼파라미터를 찾아가면서 늘려야 학습 정확도를 높일 수 있다고 생각한다.

8장

Q. 데이터 확장에서 예를 들어 2를 변형해 5와 같이 만들고 학습 시키면, 결국 5를 2로 분류하는 과적합 문제가 더 심해지는 것이 아닌가?

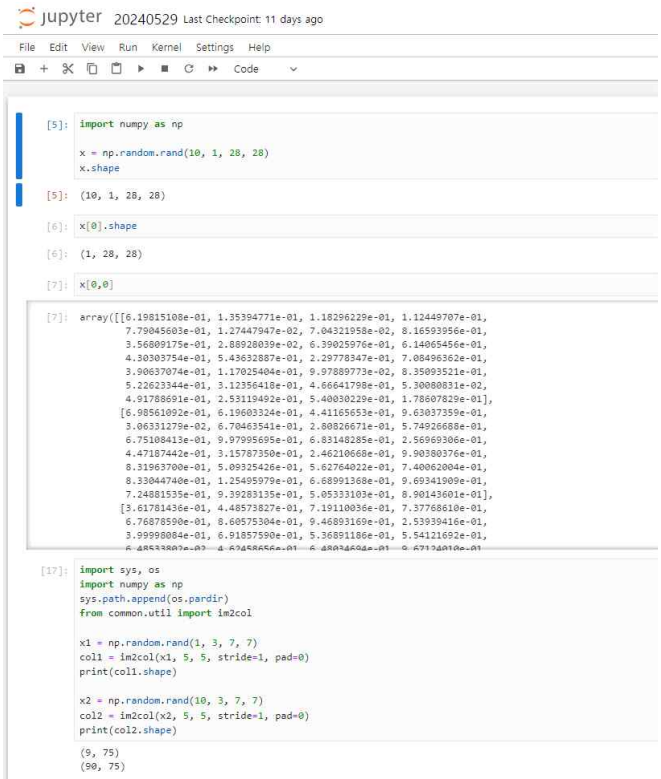
A. 조사한 결과, 데이터 확장을 위해 입력 데이터를 과하거나 부적절한 변형을 할 경우 과적합 문제가 더 심해질 수 있다. 내가 말한 예시처럼 5처럼 변형시킨 2를 학습해 5를 2로 분류하면 모델의 구분이 어려워지기 때문이다. 따라서 데이터 확장을 할 때에는 과한 변형을 하지 않고 원본 데이터의 특징을 유지하면서 조금씩 바꾸는 것이 중요하다고 한다.

Q. 딥러닝 고속화를 위해 GPU를 사용한다고 했는데, CPU보다 GPU가 더 딥러닝을 빠르게 할 수 있는 이유가 궁금하다.

A. 딥러닝은 매우 많은 연산을 필요로 한다. CPU는 상대적으로 적은 코어를 가지고 있고, 병렬 처리에 특화되지 않아 수많은 연산을 한번에 하고자 하면 당연히 연산이 밀리면서 속도가 느려질 것이다. 반면에 GPU는 매우 많은 코어를 가지고 있기 때문에 대규모 병렬 처리를 하는데 특화되어 있다. 따라서 딥러닝에서의 수많은 연산을 CPU에 비해 효과적으로 처리할 수 있다. 따라서 GPU를 딥러닝 고속화를 위해 많이 사용하는 것이다. 추후에 GPU보

다 더 병렬 처리와 연산 속도가 훌륭한 장치가 나온다면 GPU 또한 언젠가는 그것으로 대체 될 가능성이 크다고 생각한다.

7장(하) 수업시간 실습



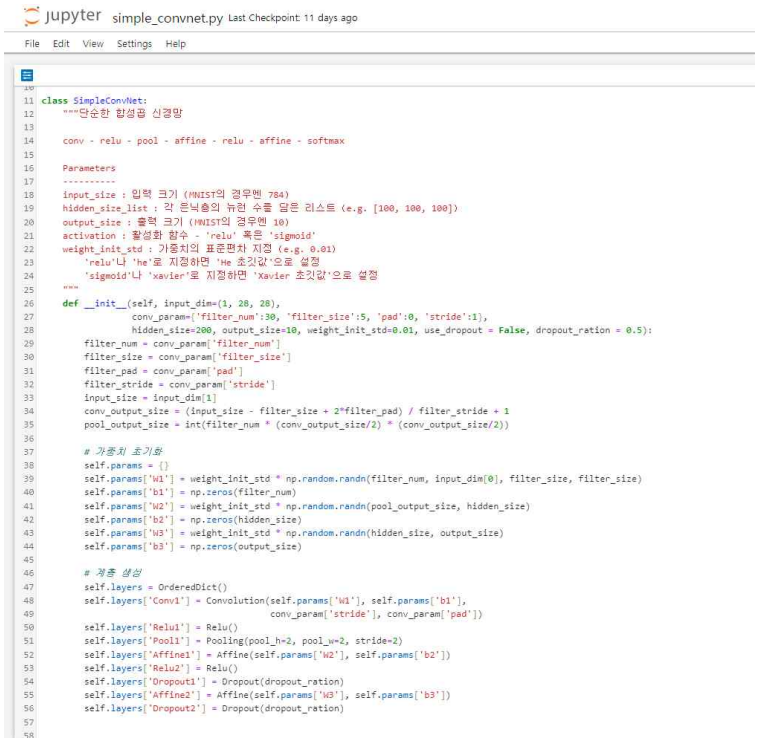
jupyter 20240529 Last Checkpoint: 11 days ago

File Edit View Run Kernel Settings Help

Code

```
[5]: import numpy as np

     x = np.random.rand(10, 1, 28, 28)
     x.shape

[5]: (10, 1, 28, 28)

[6]: x[0].shape

[6]: (1, 28, 28)

[7]: x[0,0]

[7]: array([[6.19815108e-01, 1.35394771e-01, 1.18296229e-01, 1.12449707e-01,
             7.79045603e-01, 1.27447947e-02, 7.04321958e-02, 8.16593956e-01,
             3.56809175e-01, 2.88928039e-02, 6.39025976e-01, 6.14065456e-01,
             4.30303754e-01, 5.43632887e-01, 2.29778347e-01, 7.08496362e-01,
             3.90637074e-01, 1.17025404e-01, 9.97889773e-02, 8.35093521e-01,
             5.22623344e-01, 3.12356418e-01, 4.66641798e-01, 5.30080831e-02,
             4.91788691e-01, 2.53119492e-01, 5.40030229e-01, 1.78607829e-01],
            [6.98561092e-01, 6.19603324e-01, 4.41165653e-01, 9.63037359e-01,
             3.06331279e-02, 6.70463541e-01, 2.80826671e-01, 5.74926688e-01,
             6.75108413e-01, 9.97995695e-01, 6.83148285e-01, 2.56969306e-01,
             4.47187442e-01, 3.15787350e-01, 2.46210668e-01, 9.90380376e-01,
             8.31963700e-01, 5.09325426e-01, 5.62764022e-01, 7.40062004e-01,
             8.33044740e-01, 1.25495979e-01, 6.68991368e-01, 9.69341909e-01,
             7.24881535e-01, 9.39283135e-01, 5.05333103e-01, 8.90143601e-01],
            [3.61781436e-01, 4.48573827e-01, 7.19110036e-01, 7.37768610e-01,
             6.76878590e-01, 8.60575304e-01, 9.46893169e-01, 2.53939416e-01,
             3.99998084e-01, 6.91857590e-01, 5.36891186e-01, 5.54121692e-01,
             6.48533807e-02, 4.62458656e-01, 6.48034694e-01, 9.67134018e-01

[17]: import sys, os
      import numpy as np
      sys.path.append(os.pardir)
      from common.util import im2col

      x1 = np.random.rand(1, 3, 7, 7)
      col1 = im2col(x1, 5, 5, stride=1, pad=0)
      print(col1.shape)

      x2 = np.random.rand(10, 3, 7, 7)
      col2 = im2col(x2, 5, 5, stride=1, pad=0)
      print(col2.shape)

      (9, 75)
      (90, 75)

[19]: class Convolution:
          def __init__(self, W, b, stride=1, pad=0):
              self.W = W
              self.b = b
              self.stride = stride
              self.pad = pad

          def forward(self, x):
              FN, C, FH, FW = self.W.shape
              N, C, H, W = x.shape
              out_h = 1 + int((H + 2*self.pad - FH) / self.stride)
              out_w = 1 + int((W + 2*self.pad - FW) / self.stride)

              col = im2col(x, FH, FW, self.stride, self.pad)
              col_W = self.W.reshape(FN, -1).T
              out = np.dot(col, col_W) + self.b

              out = out.reshape(N, out_h, out_w, -1).transpose(0, 3, 1, 2)

              return out

[20]: class Pooling:
          def __init__(self, pool_h, pool_w, stride=1, pad=0):
              self.pool_h = pool_h
              self.pool_w = pool_w
              self.stride = stride
              self.pad = pad

          def forward(self, x):
              N, C, H, W = x.shape
              out_h = int(1 + (H - self.pool_h) / self.stride)
              out.w = int(1 + (W - self.pool_w) / self.stride)

              col = im2col(x, self.pool_h, self.pool_w, self.stride, self.pad)
              col = col.reshape(-1, self.pool_h*self.pool_w)

              out = np.max(col, axis=1)

              out = out.reshape(N, out_h, out_w, C).transpose(0, 3, 1, 2)

              return out
```

SimpleConvNet 하이퍼 파라미터(계층) 설정 테스트(과제)_중략

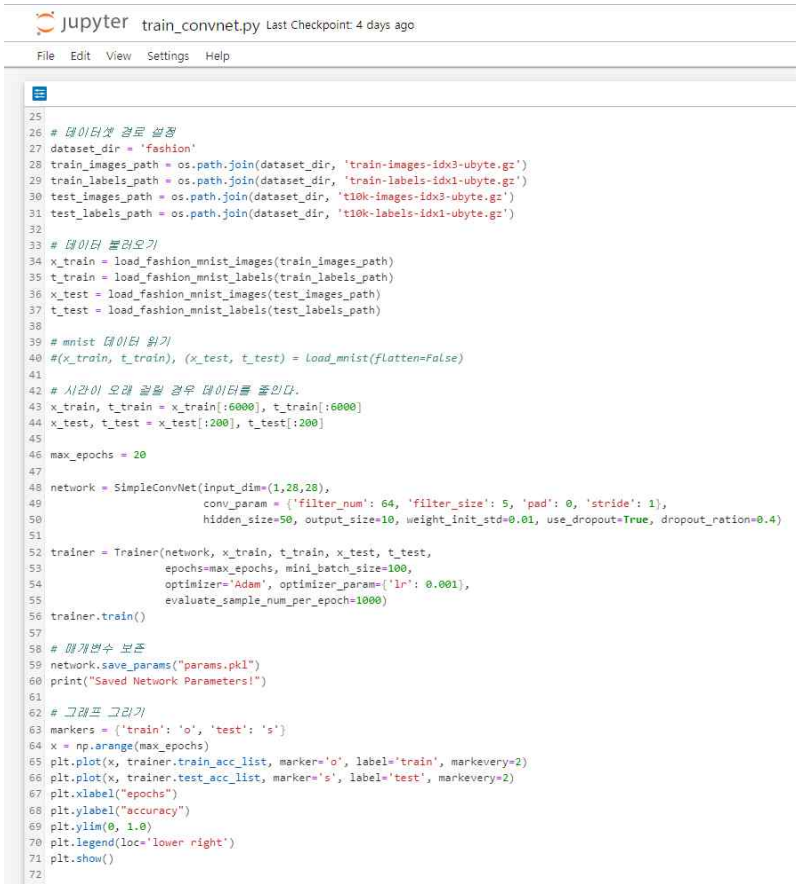


jupyter simple_convnet.py Last Checkpoint: 11 days ago

File Edit View Settings Help

```
class SimpleConvNet:
    """단순한 합성곱 신경망

    conv - relu - pool - affine - relu - affine - softmax

    Parameters
    ----------
    input_size : 입력 크기 (MNIST의 경우엔 784)
    hidden_size_list : 각 은닉층의 뉴런 수를 담은 리스트 (e.g. [100, 100, 100])
    output_size : 출력 크기 (MNIST의 경우엔 10)
    activation : 활성화 함수 - 'relu' 혹은 'sigmoid'
    weight_init_std : 가중치의 표준편차 지정 (e.g. 0.01)
        'relu'나 'he'로 지정하면 'He 초깃값'으로 설정
        'sigmoid'나 'xavier'로 지정하면 'Xavier 초깃값'으로 설정
    """
    def __init__(self, input_dim=(1, 28, 28),
                 conv_param={'filter_num':30, 'filter_size':5, 'pad':0, 'stride':1},
                 hidden_size=200, output_size=10, weight_init_std=0.01, use_dropout = False, dropout_ration = 0.5):
        filter_num = conv_param['filter_num']
        filter_size = conv_param['filter_size']
        filter_pad = conv_param['pad']
        filter_stride = conv_param['stride']
        input_size = input_dim[1]
        conv_output_size = (input_size - filter_size + 2*filter_pad) / filter_stride + 1
        pool_output_size = int(filter_num * (conv_output_size/2) * (conv_output_size/2))

        # 가중치 초기화
        self.params = {}
        self.params['W1'] = weight_init_std * np.random.randn(filter_num, input_dim[0], filter_size, filter_size)
        self.params['b1'] = np.zeros(filter_num)
        self.params['W2'] = weight_init_std * np.random.randn(pool_output_size, hidden_size)
        self.params['b2'] = np.zeros(hidden_size)
        self.params['W3'] = weight_init_std * np.random.randn(hidden_size, output_size)
        self.params['b3'] = np.zeros(output_size)

        # 계층 생성
        self.layers = OrderedDict()
        self.layers['Conv1'] = Convolution(self.params['W1'], self.params['b1'],
                                           conv_param['stride'], conv_param['pad'])
        self.layers['Relu1'] = Relu()
        self.layers['Pool1'] = Pooling(pool_h=2, pool_w=2, stride=2)
        self.layers['Affine1'] = Affine(self.params['W2'], self.params['b2'])
        self.layers['Relu2'] = Relu()
        self.layers['Dropout1'] = Dropout(dropout_ration)
        self.layers['Affine2'] = Affine(self.params['W3'], self.params['b3'])
        self.layers['Dropout2'] = Dropout(dropout_ration)

```

train_convnet 하이퍼 파라미터(가중치, 뉴런 수, 입력 데이터 등) 설정 테스트

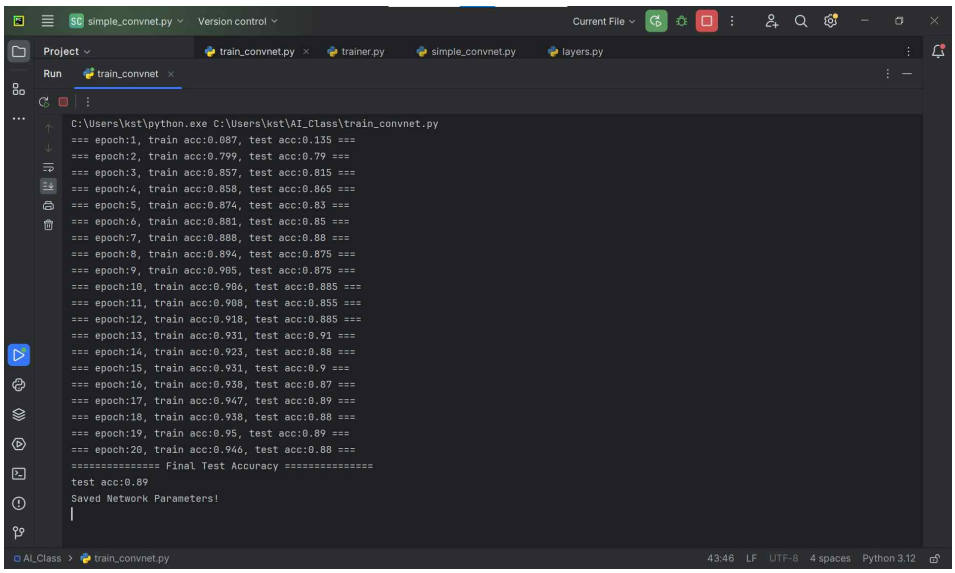


jupyter train_convnet.py Last Checkpoint: 4 days ago

File Edit View Settings Help

```
# 데이터셋 경로 설정
dataset_dir = 'fashion'
train_images_path = os.path.join(dataset_dir, 'train-images-idx3-ubyte.gz')
train_labels_path = os.path.join(dataset_dir, 'train-labels-idx1-ubyte.gz')
test_images_path = os.path.join(dataset_dir, 't10k-images-idx3-ubyte.gz')
test_labels_path = os.path.join(dataset_dir, 't10k-labels-idx1-ubyte.gz')

# 데이터 불러오기
x_train = load_fashion_mnist_images(train_images_path)
t_train = load_fashion_mnist_labels(train_labels_path)
x_test = load_fashion_mnist_images(test_images_path)
t_test = load_fashion_mnist_labels(test_labels_path)

# mnist 데이터 읽기
#(x_train, t_train), (x_test, t_test) = load_mnist(flatten=False)

# 시간이 오래 걸릴 경우 데이터를 줄인다.
x_train, t_train = x_train[:6000], t_train[:6000]
x_test, t_test = x_test[:200], t_test[:200]

max_epochs = 20

network = SimpleConvNet(input_dim=(1,28,28),
                        conv_param = {'filter_num': 64, 'filter_size': 5, 'pad': 0, 'stride': 1},
                        hidden_size=50, output_size=10, weight_init_std=0.01, use_dropout=True, dropout_ration=0.4)

trainer = Trainer(network, x_train, t_train, x_test, t_test,
                  epochs=max_epochs, mini_batch_size=100,
                  optimizer='Adam', optimizer_param={'lr': 0.001},
                  evaluate_sample_num_per_epoch=1000)
trainer.train()

# 매개변수 보존
network.save_params("params.pkl")
print("Saved Network Parameters!")

# 그래프 그리기
markers = {'train': 'o', 'test': 's'}
x = np.arange(max_epochs)
plt.plot(x, trainer.train_acc_list, marker='o', label='train', markevery=2)
plt.plot(x, trainer.test_acc_list, marker='s', label='test', markevery=2)
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.ylim(0, 1.0)
plt.legend(loc='lower right')
plt.show()

```

실행 결과 테스트(과제)


simple_convnet.py
Version control
Current File
Project
train_convnet.py
trainer.py
simple_convnet.py
layers.py
Run
train_convnet
C:\Users\kst\python.exe C:\Users\kst\AI_Class\train_convnet.py
=== epoch:1, train acc:0.087, test acc:0.135 ===
=== epoch:2, train acc:0.799, test acc:0.79 ===
=== epoch:3, train acc:0.857, test acc:0.815 ===
=== epoch:4, train acc:0.858, test acc:0.865 ===
=== epoch:5, train acc:0.874, test acc:0.83 ===
=== epoch:6, train acc:0.881, test acc:0.85 ===
=== epoch:7, train acc:0.888, test acc:0.88 ===
=== epoch:8, train acc:0.894, test acc:0.875 ===
=== epoch:9, train acc:0.905, test acc:0.875 ===
=== epoch:10, train acc:0.906, test acc:0.885 ===
=== epoch:11, train acc:0.908, test acc:0.855 ===
=== epoch:12, train acc:0.918, test acc:0.885 ===
=== epoch:13, train acc:0.931, test acc:0.91 ===
=== epoch:14, train acc:0.923, test acc:0.88 ===
=== epoch:15, train acc:0.931, test acc:0.9 ===
=== epoch:16, train acc:0.938, test acc:0.87 ===
=== epoch:17, train acc:0.947, test acc:0.89 ===
=== epoch:18, train acc:0.938, test acc:0.88 ===
=== epoch:19, train acc:0.95, test acc:0.89 ===
=== epoch:20, train acc:0.946, test acc:0.88 ===
============== Final Test Accuracy ==============
test acc:0.89
Saved Network Parameters!
AI_Class
train_convnet.py
43:46
LF
UTF-8
4 spaces
Python 3.12

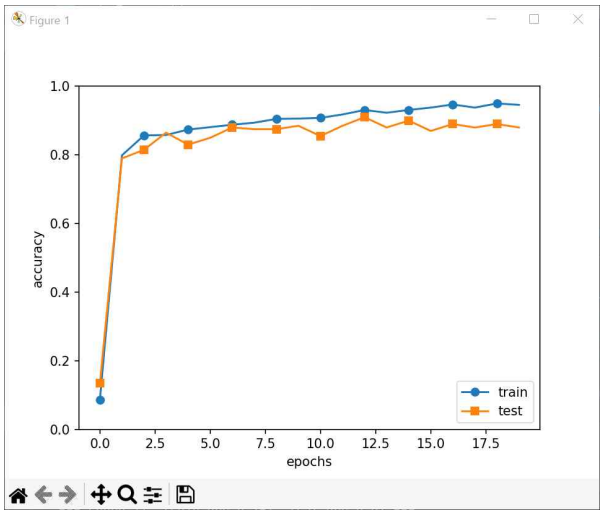
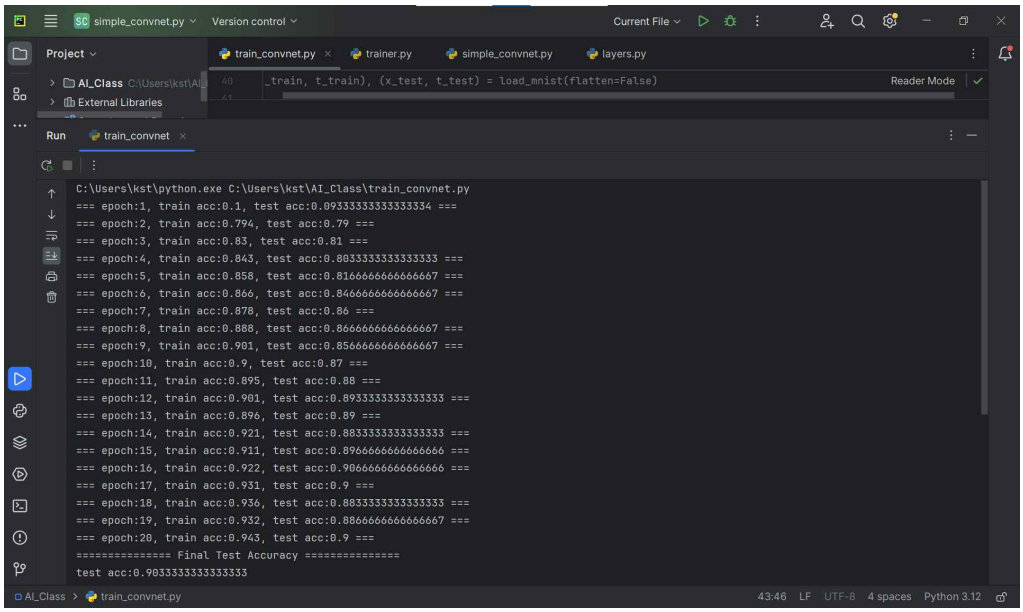


Figure 1
1.0
0.8
0.6
0.4
0.2
0.0
accuracy
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
epochs
train
test


simple_convnet.py
Version control
Current File
Project
train_convnet.py
trainer.py
simple_convnet.py
layers.py
AI_Class
External Libraries
_train, t_train), (x_test, t_test) = load_mnist(flatten=False)
Reader Mode
Run
train_convnet
C:\Users\kst\python.exe C:\Users\kst\AI_Class\train_convnet.py
=== epoch:1, train acc:0.1, test acc:0.09333333333333334 ===
=== epoch:2, train acc:0.794, test acc:0.79 ===
=== epoch:3, train acc:0.83, test acc:0.81 ===
=== epoch:4, train acc:0.843, test acc:0.8033333333333333 ===
=== epoch:5, train acc:0.858, test acc:0.8166666666666667 ===
=== epoch:6, train acc:0.866, test acc:0.8466666666666667 ===
=== epoch:7, train acc:0.878, test acc:0.86 ===
=== epoch:8, train acc:0.888, test acc:0.8666666666666667 ===
=== epoch:9, train acc:0.901, test acc:0.8566666666666667 ===
=== epoch:10, train acc:0.9, test acc:0.87 ===
=== epoch:11, train acc:0.895, test acc:0.88 ===
=== epoch:12, train acc:0.901, test acc:0.8933333333333333 ===
=== epoch:13, train acc:0.896, test acc:0.89 ===
=== epoch:14, train acc:0.921, test acc:0.8833333333333333 ===
=== epoch:15, train acc:0.911, test acc:0.8966666666666666 ===
=== epoch:16, train acc:0.922, test acc:0.9066666666666666 ===
=== epoch:17, train acc:0.931, test acc:0.9 ===
=== epoch:18, train acc:0.936, test acc:0.8833333333333333 ===
=== epoch:19, train acc:0.932, test acc:0.8866666666666667 ===
=== epoch:20, train acc:0.943, test acc:0.9 ===
============== Final Test Accuracy ==============
test acc:0.9033333333333333
AI_Class
train_convnet.py
43:46
LF
UTF-8
4 spaces
Python 3.12