2. 4 개의 화소를 다음과 같이 분류하는 심층 신경망 프로그램을 작성하여 훈련 및 테스트를 실시하고 결과를 분석하시오.

예상했던 정확도보다 훨씬 낮게 나왔다.

```
from tensorflow import keras
from sklearn.model_selection import train_test_split
from tensorflow.keras import datasets, layers, models
import numpy as np
import tensorflow as tf

train_input = np.array([[[1, 1], [1, 1]], [[0, 0], [0, 0]], [[1, 0], [1, 0]], [[0, 1], [0, 1]]
train_labels = np.array([0, 0, 1, 1, 2, 2, 3, 3])
label = ["solid", "vertical", "diagonal", "horizontal"]

train_input = train_input / 1.
model = models.Sequential()
model.add(layers.Flatten(input_shape=(2, 2)))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
model.fit(train_input, train_labels, epochs=5)
test_loss, test_acc = model.evaluate(train_input, train_labels)
print('정확도:', test_acc)
```

```
Epoch 1/5
1/1 [==============================] - 0s 418ms/step - loss: 2.2766 - accuracy: 0.2500
Epoch 2/5
1/1 [==============================] - 0s 16ms/step - loss: 2.2563 - accuracy: 0.2500
Epoch 3/5
1/1 [==============================] - 0s 10ms/step - loss: 2.2360 - accuracy: 0.3750
Epoch 4/5
1/1 [==============================] - 0s 8ms/step - loss: 2.2160 - accuracy: 0.3750
Epoch 5/5
1/1 [==============================] - 0s 11ms/step - loss: 2.1962 - accuracy: 0.3750
1/1 [==============================] - 0s 146ms/step - loss: 2.1765 - accuracy: 0.3750
정확도: 0.375
```

더블클릭 또는 Enter 키를 눌러 수정

3 패션 아이템(fashion-MNIST)을 기본 MLP 와 심층신경망을 이용하여 분류하는 프로그램을 각각 작성한 후 훈련 및 테

스트를 통해 성능을 비교하시오

기본 mlp보다 심층 신경망의 정확도가 더 높은 것을 볼 수 있다.

7 컨볼루션 신경망을 이용해서 시도하시오. 그리고 MLP, 심층신경망과 컨볼루션 신경망의 성능을 비교하시오.

CNN > DNN > MLP 순으로 높은 정확성을 보이는 것을 볼 수 있다.

```
# 심층 신경망
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras import datasets, layers, models
fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

train_images = train_images / 255.0
test_images = test_images / 255.0
model = models.Sequential()
model.add(layers.Flatten(input_shape=(28, 28)))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5)
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('정확도:', test_acc)
```

```
Epoch 1/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.4784 - accuracy: 0.8285
Epoch 2/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.3610 - accuracy: 0.8677
Epoch 3/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.3239 - accuracy: 0.8811
Epoch 4/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.2997 - accuracy: 0.8887
Epoch 5/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.2796 - accuracy: 0.8961
313/313 [==============================] - 1s 4ms/step - loss: 0.3499 - accuracy: 0.8722
정확도: 0.8722000122070312
```

```
# 기본 mlp
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras import datasets, layers, models
```

```
7
8 (train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
9 model = tf.keras.models.Sequential()
10 model.add(tf.keras.layers.Dense(512, activation='relu', input_shape=(784, )))
11 model.add(tf.keras.layers.Dense(10, activation='softmax'))
12 model.compile(optimizer='rmsprop', loss='mse', metrics=['accuracy'])
13 train_images = train_images.reshape((60000, 784)) #flatten
14 train_images = train_images.astype('float32')/255.
15 test_images = test_images.reshape((10000, 784))
16 test_images = test_images.astype('float32')/255.
17 train_labels = tf.keras.utils.to_categorical(train_labels)
18 test_labels = tf.keras.utils.to_categorical(test_labels)
19 model.fit(train_images, train_labels, epochs=5, batch_size=128)
20 test_loss, test_acc = model.evaluate(test_images, test_labels)
21 print('정확도:', test_acc)
```

```
Epoch 1/5
469/469 [==============================] - 6s 11ms/step - loss: 0.0330 - accuracy: 0.7686
Epoch 2/5
469/469 [==============================] - 5s 11ms/step - loss: 0.0209 - accuracy: 0.8562
Epoch 3/5
469/469 [==============================] - 5s 11ms/step - loss: 0.0185 - accuracy: 0.8731
Epoch 4/5
469/469 [==============================] - 6s 13ms/step - loss: 0.0172 - accuracy: 0.8814
Epoch 5/5
469/469 [==============================] - 5s 11ms/step - loss: 0.0162 - accuracy: 0.8888
313/313 [==============================] - 1s 4ms/step - loss: 0.0191 - accuracy: 0.8691
정확도: 0.8690999746322632
```

```
1 # CNN
2 import tensorflow as tf
3 from tensorflow.keras import datasets, layers, models
4 (train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
5 train_images = train_images.reshape((60000, 28, 28, 1))
6 test_images = test_images.reshape((10000, 28, 28, 1))
7 # 픽셀 값을 0~1 사이로 정규화한다.
8 train_images, test_images = train_images / 255.0, test_images / 255.0
9 model = models.Sequential()
10 model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
11 model.add(layers.MaxPooling2D((2, 2)))
12 model.add(layers.Conv2D(64, (3, 3), activation='relu'))
13 model.add(layers.MaxPooling2D((2, 2)))
14 model.add(layers.Conv2D(64, (3, 3), activation='relu'))
15 model.add(layers.Flatten())
16 model.add(layers.Dense(64, activation='relu'))
17 model.add(layers.Dense(10, activation='softmax'))
18 model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
19 model.fit(train_images, train_labels, epochs=5)
20 test_loss, test_acc = model.evaluate(test_images, test_labels)
21 print('정확도:', test_acc)
```

```
Epoch 1/5
1875/1875 [==============================] - 75s 40ms/step - loss: 0.4920 - accuracy: 0.8189
Epoch 2/5
1875/1875 [==============================] - 71s 38ms/step - loss: 0.3138 - accuracy: 0.8860
Epoch 3/5
```

```
1875/1875 [==============================] - 72s 39ms/step - loss: 0.2701 - accuracy: 0.9002
Epoch 4/5
1875/1875 [==============================] - 69s 37ms/step - loss: 0.2408 - accuracy: 0.9122
Epoch 5/5
1875/1875 [==============================] - 69s 37ms/step - loss: 0.2177 - accuracy: 0.9186
313/313 [==============================] - 4s 12ms/step - loss: 0.2867 - accuracy: 0.8974
정확도: 0.8974000215530396
```

4 붓꽃 데이터 세트를 이용하여 꽃받침 길이와 너비, 꽃잎의 길이와 너비 등 4 가지 징을 입력으로 받아서 어떤 붓꽃인지 예측하고자 한다. 케라스로 심층신경망을 구현하고 훈련하여 테스트 하시오

```
from sklearn import datasets
from sklearn.model_selection import train_test_split

iris = datasets.load_iris()
X = iris.data
y = iris.target
train_images, test_images, train_labels, test_labels = train_test_split(X, y, test_size=0.2, ra

model = models.Sequential()
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5)
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('정확도:', test_acc)

```

```
Epoch 1/5
4/4 [==============================] - 0s 4ms/step - loss: 2.0657 - accuracy: 0.2667
Epoch 2/5
4/4 [==============================] - 0s 4ms/step - loss: 1.4277 - accuracy: 0.3750
Epoch 3/5
4/4 [==============================] - 0s 4ms/step - loss: 1.0875 - accuracy: 0.4917
Epoch 4/5
4/4 [==============================] - 0s 5ms/step - loss: 0.9264 - accuracy: 0.5583
Epoch 5/5
4/4 [==============================] - 0s 4ms/step - loss: 0.8397 - accuracy: 0.6333
1/1 [==============================] - 0s 143ms/step - loss: 0.7759 - accuracy: 0.8333
정확도: 0.8333333134651184
```

```
from google.colab import files
myfile = files.upload()
```

파일 선택 winequality-white.csv

- **winequality-white.csv**(text/csv) - 264426 bytes, last modified: 2022. 11. 22. - 100% done

5 심층 신경망을 통해서 와인을 분류해보자

```
import pandas as pd
import io
wines = pd.read_csv(io.BytesIO(myfile['winequality-white.csv']), sep=';', header=0, skiprows=[
wines = np.array(wines)
wines = np.delete(wines, 0, axis=1)
X = wines[:,:10]
y = wines[:,10]
train_images, test_images, train_labels, test_labels = train_test_split(X, y, test_size=0.2, ra

model = models.Sequential()
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5)
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('정확도:', test_acc)

```

```
Epoch 1/5
123/123 [==============================] - 1s 3ms/step - loss: 1.5354 - accuracy: 0.4215
Epoch 2/5
123/123 [==============================] - 0s 2ms/step - loss: 1.3521 - accuracy: 0.4373
Epoch 3/5
123/123 [==============================] - 0s 2ms/step - loss: 1.3158 - accuracy: 0.4457
Epoch 4/5
123/123 [==============================] - 0s 2ms/step - loss: 1.3450 - accuracy: 0.4340
Epoch 5/5
123/123 [==============================] - 0s 2ms/step - loss: 1.3305 - accuracy: 0.4442
31/31 [==============================] - 0s 2ms/step - loss: 1.2970 - accuracy: 0.4592
정확도: 0.4591836631298065
```

```
myfile = files.upload()
```

파일 선택 seeds_dataset.txt

- **seeds_dataset.txt**(text/plain) - 8835 bytes, last modified: 2022. 11. 22. - 100% done

```
Saving seeds_dataset.txt to seeds_dataset (2).txt
```

6 종을 예측해보자

```
import pandas as pd
import io
seeds = pd.read_csv(io.BytesIO(myfile['seeds_dataset.txt']), sep='₩t', header=0, skiprows=[1])
```

```
seeds = np.array(seeds)
print(seeds.shape)
X = seeds[:,:7]
y = seeds[:,7]
train_images, test_images, train_labels, test_labels = train_test_split(X, y, test_size=0.2, ra
model = models.Sequential()
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5)
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('정확도:', test_acc)
```

```
(197, 8)
Epoch 1/5
5/5 [==============================] - 0s 4ms/step - loss: 1.5988 - accuracy: 0.4713
Epoch 2/5
5/5 [==============================] - 0s 3ms/step - loss: 0.9800 - accuracy: 0.3758
Epoch 3/5
5/5 [==============================] - 0s 5ms/step - loss: 0.8540 - accuracy: 0.7070
Epoch 4/5
5/5 [==============================] - 0s 4ms/step - loss: 0.7870 - accuracy: 0.7261
Epoch 5/5
5/5 [==============================] - 0s 3ms/step - loss: 0.7575 - accuracy: 0.7006
2/2 [==============================] - 0s 7ms/step - loss: 0.7105 - accuracy: 0.7250
정확도: 0.7250000238418579
```