텐서플로가 제공하는 데이터셋 확인

MNIST 필기 숫자 데이터셋

CIFAR-10 자연영상 데이터셋

In [2]:

```
# !pip install tensorflow
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting tensorflow
  Downloading tensorflow-2.11.0-cp39-cp39-win_amd64.whl (1.9 kB)
Collecting tensorflow-intel==2.11.0
  Downloading tensorflow_intel-2.11.0-cp39-cp39-win_amd64.whl (266.3 MB)
     ---------------------------------------- 266.3/266.3 MB 5.5 MB/s eta 0:00:00
Collecting opt-einsum>=2.3.2
  Downloading opt_einsum-3.3.0-py3-none-any.whl (65 kB)
     ---------------------------------------- 65.5/65.5 kB 3.7 MB/s eta 0:00:00
Requirement already satisfied: setuptools in c:WprogramdataWanaconda3WlibWsite-packages
(from tensorflow-intel==2.11.0->tensorflow) (63.4.1)
Collecting flatbuffers>=2.0
  Downloading flatbuffers-23.1.21-py2.py3-none-any.whl (26 kB)
Collecting tensorflow-estimator<2.12,>=2.11.0
  Downloading tensorflow_estimator-2.11.0-py2.py3-none-any.whl (439 kB)
     ---------------------------------------- 439.2/439.2 kB 13.4 MB/s eta 0:00:00
Requirement already satisfied: typing-extensions>=3.6.6 in c:WprogramdataWanaconda3Wlib
Wsite-packages (from tensorflow-intel==2.11.0->tensorflow) (4.3.0)
Requirement already satisfied: numpy>=1.20 in c:WprogramdataWanaconda3WlibWsite-packages
```

In [3]:

```
import tensorflow as tf
import tensorflow.keras.datasets as ds
import matplotlib.pyplot as plt
```

In [4]:

```
(x_train,y_train),(x_test,y_test)=ds.mnist.load_data()
print(x_train.shape,y_train.shape,x_test.shape,y_test.shape)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.np
z (https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz)
11490434/11490434 [==============================] - 1s 0us/step
(60000, 28, 28) (60000,) (10000, 28, 28) (10000,)
```

In [5]:

```
plt.figure(figsize=(24,3))
plt.suptitle('MNIST',fontsize=30)
```

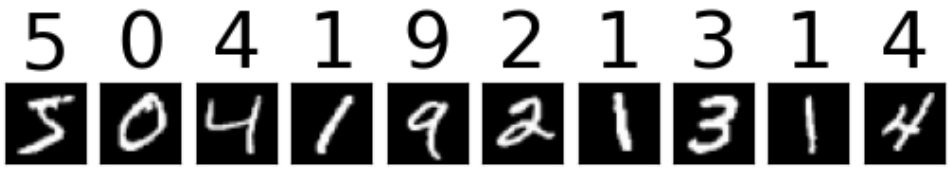
Out[5]:

```
Text(0.5, 0.98, 'MNIST')
```

```
<Figure size 2400x300 with 0 Axes>
```

In [6]:

```
for i in range(10):
    plt.subplot(1,10,i+1)
    plt.imshow(x_train[i],cmap='gray')
    plt.xticks([]); plt.yticks([])
    plt.title(str(y_train[i]),fontsize=30)
```



In [8]:

```
(x_train,y_train),(x_test,y_test)=ds.cifar10.load_data()
print(x_train.shape,y_train.shape,x_test.shape,y_test.shape)
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz (https://ww
w.cs.toronto.edu/~kriz/cifar-10-python.tar.gz)
170498071/170498071 [==============================] - 17s 0us/step
(50000, 32, 32, 3) (50000, 1) (10000, 32, 32, 3) (10000, 1)
```

In [9]:

```
class_names=['airplane','car','bird','cat','deer','dog','frog','horse','ship','truck']
```

In [10]:

```
plt.figure(figsize=(24,3))
plt.suptitle('CIFAR-10',fontsize=30)
```

Out[10]:

```
Text(0.5, 0.98, 'CIFAR-10')
```

```
<Figure size 2400x300 with 0 Axes>
```

In [11]:

```
for i in range(10):
    plt.subplot(1,10,i+1)
    plt.imshow(x_train[i])
    plt.xticks([]); plt.yticks([])
    plt.title(class_names[y_train[i,0]],fontsize=30)
```

필기 숫자 인식

다층 퍼셉트론으로 MNIST 인식하기(SGD 옵티마이저)

In [12]:

```
import numpy as np
import tensorflow as tf
import tensorflow.keras.datasets as ds
```

In [13]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import SGD
```

In [14]:

```
(x_train,y_train),(x_test,y_test)=ds.mnist.load_data()
```

In [15]:

```
x_train=x_train.reshape(60000,784)
x_test=x_test.reshape(10000,784)
x_train=x_train.astype(np.float32)/255.0
x_test=x_test.astype(np.float32)/255.0
y_train=tf.keras.utils.to_categorical(y_train,10)
y_test=tf.keras.utils.to_categorical(y_test,10)
```

In [16]:

```
mlp=Sequential()
mlp.add(Dense(units=512,activation='tanh',input_shape=(784,)))
mlp.add(Dense(units=10,activation='softmax'))
```

In [17]:

```
mlp.compile(loss='MSE',optimizer=SGD(learning_rate=0.01),metrics=['accuracy'])
```

In [18]:

```
mlp.fit(x_train,y_train,batch_size=128,epochs=50,validation_data=(x_test,y_test),verbose=2)
```

```
Epoch 1/50
469/469 - 3s - loss: 0.0899 - accuracy: 0.1602 - val_loss: 0.0864 - val_accuracy: 0.2810 -
3s/epoch - 6ms/step
Epoch 2/50
469/469 - 2s - loss: 0.0826 - accuracy: 0.3589 - val_loss: 0.0787 - val_accuracy: 0.4076 -
2s/epoch - 5ms/step
Epoch 3/50
469/469 - 2s - loss: 0.0751 - accuracy: 0.4385 - val_loss: 0.0714 - val_accuracy: 0.4694 -
2s/epoch - 5ms/step
Epoch 4/50
469/469 - 2s - loss: 0.0683 - accuracy: 0.4983 - val_loss: 0.0649 - val_accuracy: 0.5359 -
2s/epoch - 5ms/step
Epoch 5/50
469/469 - 2s - loss: 0.0624 - accuracy: 0.5646 - val_loss: 0.0593 - val_accuracy: 0.6101 -
2s/epoch - 5ms/step
Epoch 6/50
469/469 - 2s - loss: 0.0572 - accuracy: 0.6360 - val_loss: 0.0542 - val_accuracy: 0.6744 -
2s/epoch - 5ms/step
Epoch 7/50
469/469 - 2s - loss: 0.0526 - accuracy: 0.6912 - val_loss: 0.0497 - val_accuracy: 0.7260 -
2s/epoch - 5ms/step
Epoch 8/50
469/469 - 2s - loss: 0.0485 - accuracy: 0.7321 - val_loss: 0.0457 - val_accuracy: 0.7617 -
2s/epoch - 5ms/step
Epoch 9/50
469/469 - 2s - loss: 0.0449 - accuracy: 0.7624 - val_loss: 0.0423 - val_accuracy: 0.7867 -
2s/epoch - 4ms/step
Epoch 10/50
469/469 - 2s - loss: 0.0419 - accuracy: 0.7846 - val_loss: 0.0395 - val_accuracy: 0.8031 -
2s/epoch - 5ms/step
Epoch 11/50
469/469 - 2s - loss: 0.0393 - accuracy: 0.7997 - val_loss: 0.0370 - val_accuracy: 0.8153 -
2s/epoch - 5ms/step
Epoch 12/50
469/469 - 2s - loss: 0.0370 - accuracy: 0.8113 - val_loss: 0.0349 - val_accuracy: 0.8257 -
2s/epoch - 5ms/step
Epoch 13/50
469/469 - 2s - loss: 0.0352 - accuracy: 0.8197 - val_loss: 0.0332 - val_accuracy: 0.8329 -
2s/epoch - 5ms/step
Epoch 14/50
469/469 - 2s - loss: 0.0336 - accuracy: 0.8260 - val_loss: 0.0317 - val_accuracy: 0.8386 -
2s/epoch - 5ms/step
Epoch 15/50
469/469 - 2s - loss: 0.0322 - accuracy: 0.8314 - val_loss: 0.0304 - val_accuracy: 0.8439 -
2s/epoch - 5ms/step
Epoch 16/50
469/469 - 2s - loss: 0.0310 - accuracy: 0.8367 - val_loss: 0.0293 - val_accuracy: 0.8494 -
2s/epoch - 5ms/step
Epoch 17/50
469/469 - 2s - loss: 0.0299 - accuracy: 0.8403 - val_loss: 0.0283 - val_accuracy: 0.8528 -
2s/epoch - 5ms/step
Epoch 18/50
469/469 - 2s - loss: 0.0290 - accuracy: 0.8438 - val_loss: 0.0274 - val_accuracy: 0.8556 -
2s/epoch - 5ms/step
Epoch 19/50
469/469 - 2s - loss: 0.0282 - accuracy: 0.8470 - val_loss: 0.0266 - val_accuracy: 0.8592 -
2s/epoch - 5ms/step
Epoch 20/50
469/469 - 2s - loss: 0.0274 - accuracy: 0.8498 - val_loss: 0.0259 - val_accuracy: 0.8628 -
2s/epoch - 5ms/step
Epoch 21/50
469/469 - 2s - loss: 0.0268 - accuracy: 0.8522 - val_loss: 0.0253 - val_accuracy: 0.8649 -
2s/epoch - 5ms/step
Epoch 22/50
```

```
469/469 - 2s - loss: 0.0262 - accuracy: 0.8547 - val_loss: 0.0247 - val_accuracy: 0.8669 -
2s/epoch - 5ms/step
Epoch 23/50
469/469 - 2s - loss: 0.0256 - accuracy: 0.8569 - val_loss: 0.0242 - val_accuracy: 0.8691 -
2s/epoch - 5ms/step
Epoch 24/50
469/469 - 2s - loss: 0.0251 - accuracy: 0.8587 - val_loss: 0.0237 - val_accuracy: 0.8717 -
2s/epoch - 5ms/step
Epoch 25/50
469/469 - 2s - loss: 0.0247 - accuracy: 0.8606 - val_loss: 0.0233 - val_accuracy: 0.8724 -
2s/epoch - 5ms/step
Epoch 26/50
469/469 - 2s - loss: 0.0242 - accuracy: 0.8624 - val_loss: 0.0229 - val_accuracy: 0.8748 -
2s/epoch - 5ms/step
Epoch 27/50
469/469 - 2s - loss: 0.0238 - accuracy: 0.8641 - val_loss: 0.0225 - val_accuracy: 0.8763 -
2s/epoch - 5ms/step
Epoch 28/50
469/469 - 2s - loss: 0.0235 - accuracy: 0.8656 - val_loss: 0.0221 - val_accuracy: 0.8775 -
2s/epoch - 5ms/step
Epoch 29/50
469/469 - 2s - loss: 0.0231 - accuracy: 0.8669 - val_loss: 0.0218 - val_accuracy: 0.8785 -
2s/epoch - 5ms/step
Epoch 30/50
469/469 - 2s - loss: 0.0228 - accuracy: 0.8685 - val_loss: 0.0215 - val_accuracy: 0.8800 -
2s/epoch - 5ms/step
Epoch 31/50
469/469 - 2s - loss: 0.0225 - accuracy: 0.8698 - val_loss: 0.0212 - val_accuracy: 0.8808 -
2s/epoch - 5ms/step
Epoch 32/50
469/469 - 2s - loss: 0.0222 - accuracy: 0.8712 - val_loss: 0.0210 - val_accuracy: 0.8822 -
2s/epoch - 5ms/step
Epoch 33/50
469/469 - 2s - loss: 0.0220 - accuracy: 0.8724 - val_loss: 0.0207 - val_accuracy: 0.8833 -
2s/epoch - 5ms/step
Epoch 34/50
469/469 - 2s - loss: 0.0217 - accuracy: 0.8736 - val_loss: 0.0205 - val_accuracy: 0.8835 -
2s/epoch - 5ms/step
Epoch 35/50
469/469 - 2s - loss: 0.0215 - accuracy: 0.8747 - val_loss: 0.0202 - val_accuracy: 0.8844 -
2s/epoch - 5ms/step
Epoch 36/50
469/469 - 2s - loss: 0.0212 - accuracy: 0.8759 - val_loss: 0.0200 - val_accuracy: 0.8854 -
2s/epoch - 5ms/step
Epoch 37/50
469/469 - 2s - loss: 0.0210 - accuracy: 0.8769 - val_loss: 0.0198 - val_accuracy: 0.8860 -
2s/epoch - 5ms/step
Epoch 38/50
469/469 - 2s - loss: 0.0208 - accuracy: 0.8782 - val_loss: 0.0196 - val_accuracy: 0.8869 -
2s/epoch - 5ms/step
Epoch 39/50
469/469 - 2s - loss: 0.0206 - accuracy: 0.8790 - val_loss: 0.0194 - val_accuracy: 0.8878 -
2s/epoch - 5ms/step
Epoch 40/50
469/469 - 2s - loss: 0.0204 - accuracy: 0.8798 - val_loss: 0.0193 - val_accuracy: 0.8891 -
2s/epoch - 5ms/step
Epoch 41/50
469/469 - 2s - loss: 0.0203 - accuracy: 0.8805 - val_loss: 0.0191 - val_accuracy: 0.8895 -
2s/epoch - 5ms/step
Epoch 42/50
469/469 - 2s - loss: 0.0201 - accuracy: 0.8814 - val_loss: 0.0189 - val_accuracy: 0.8898 -
2s/epoch - 5ms/step
Epoch 43/50
469/469 - 2s - loss: 0.0199 - accuracy: 0.8819 - val_loss: 0.0188 - val_accuracy: 0.8905 -
2s/epoch - 5ms/step
```

```
Epoch 44/50
469/469 - 2s - loss: 0.0198 - accuracy: 0.8827 - val_loss: 0.0186 - val_accuracy: 0.8916 -
2s/epoch - 5ms/step
Epoch 45/50
469/469 - 2s - loss: 0.0196 - accuracy: 0.8834 - val_loss: 0.0185 - val_accuracy: 0.8925 -
2s/epoch - 5ms/step
Epoch 46/50
469/469 - 2s - loss: 0.0195 - accuracy: 0.8839 - val_loss: 0.0183 - val_accuracy: 0.8926 -
2s/epoch - 5ms/step
Epoch 47/50
469/469 - 2s - loss: 0.0193 - accuracy: 0.8846 - val_loss: 0.0182 - val_accuracy: 0.8933 -
2s/epoch - 5ms/step
Epoch 48/50
469/469 - 2s - loss: 0.0192 - accuracy: 0.8853 - val_loss: 0.0181 - val_accuracy: 0.8935 -
2s/epoch - 5ms/step
Epoch 49/50
469/469 - 2s - loss: 0.0191 - accuracy: 0.8858 - val_loss: 0.0180 - val_accuracy: 0.8939 -
2s/epoch - 5ms/step
Epoch 50/50
469/469 - 2s - loss: 0.0189 - accuracy: 0.8864 - val_loss: 0.0178 - val_accuracy: 0.8954 -
2s/epoch - 5ms/step
Out[18]:

<keras.callbacks.History at 0x1422ea187c0>
```

In [19]:

```
res=mlp.evaluate(x_test,y_test,verbose=0)
print('정확률=',res[1]*100)
```

정확률= 89.53999876976013

Adam 옵티마이저를 사용하여 성능 향상

In [20]:

```
import numpy as np
import tensorflow as tf
import tensorflow.keras.datasets as ds
```

In [21]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
```

In [22]:

```
(x_train,y_train),(x_test,y_test)=ds.mnist.load_data()
```

In [23]:

```
x_train=x_train.reshape(60000,784)
x_test=x_test.reshape(10000,784)
x_train=x_train.astype(np.float32)/255.0
x_test=x_test.astype(np.float32)/255.0
y_train=tf.keras.utils.to_categorical(y_train,10)
y_test=tf.keras.utils.to_categorical(y_test,10)
```

In [24]:

```
mlp=Sequential()
mlp.add(Dense(units=512,activation='tanh',input_shape=(784,)))
mlp.add(Dense(units=10,activation='softmax'))
```

In [25]:

```
mlp.compile(loss='MSE',optimizer=Adam(learning_rate=0.001),metrics=['accuracy'])
```

In [26]:

```
mlp.fit(x_train,y_train,batch_size=128,epochs=50,validation_data=(x_test,y_test),verbose=2)
```

```
Epoch 1/50
469/469 - 4s - loss: 0.0148 - accuracy: 0.9015 - val_loss: 0.0100 - val_accuracy: 0.9355 -
4s/epoch - 8ms/step
Epoch 2/50
469/469 - 3s - loss: 0.0092 - accuracy: 0.9403 - val_loss: 0.0076 - val_accuracy: 0.9510 -
3s/epoch - 7ms/step
Epoch 3/50
469/469 - 3s - loss: 0.0066 - accuracy: 0.9582 - val_loss: 0.0062 - val_accuracy: 0.9605 -
3s/epoch - 7ms/step
Epoch 4/50
469/469 - 3s - loss: 0.0052 - accuracy: 0.9682 - val_loss: 0.0057 - val_accuracy: 0.9617 -
3s/epoch - 7ms/step
Epoch 5/50
469/469 - 3s - loss: 0.0042 - accuracy: 0.9744 - val_loss: 0.0049 - val_accuracy: 0.9687 -
3s/epoch - 7ms/step
Epoch 6/50
469/469 - 3s - loss: 0.0035 - accuracy: 0.9790 - val_loss: 0.0045 - val_accuracy: 0.9709 -
3s/epoch - 7ms/step
Epoch 7/50
469/469 - 3s - loss: 0.0029 - accuracy: 0.9832 - val_loss: 0.0039 - val_accuracy: 0.9744 -
3s/epoch - 7ms/step
Epoch 8/50
469/469 - 3s - loss: 0.0024 - accuracy: 0.9862 - val_loss: 0.0039 - val_accuracy: 0.9746 -
3s/epoch - 7ms/step
Epoch 9/50
469/469 - 3s - loss: 0.0020 - accuracy: 0.9885 - val_loss: 0.0036 - val_accuracy: 0.9768 -
3s/epoch - 7ms/step
Epoch 10/50
469/469 - 3s - loss: 0.0017 - accuracy: 0.9906 - val_loss: 0.0032 - val_accuracy: 0.9794 -
3s/epoch - 7ms/step
Epoch 11/50
469/469 - 3s - loss: 0.0014 - accuracy: 0.9925 - val_loss: 0.0035 - val_accuracy: 0.9768 -
3s/epoch - 7ms/step
Epoch 12/50
469/469 - 3s - loss: 0.0013 - accuracy: 0.9932 - val_loss: 0.0029 - val_accuracy: 0.9808 -
3s/epoch - 7ms/step
Epoch 13/50
469/469 - 3s - loss: 0.0010 - accuracy: 0.9945 - val_loss: 0.0034 - val_accuracy: 0.9769 -
3s/epoch - 7ms/step
Epoch 14/50
469/469 - 3s - loss: 9.0885e-04 - accuracy: 0.9953 - val_loss: 0.0029 - val_accuracy: 0.98
04 - 3s/epoch - 7ms/step
Epoch 15/50
469/469 - 3s - loss: 8.2750e-04 - accuracy: 0.9957 - val_loss: 0.0036 - val_accuracy: 0.97
62 - 3s/epoch - 7ms/step
Epoch 16/50
469/469 - 3s - loss: 7.7372e-04 - accuracy: 0.9961 - val_loss: 0.0032 - val_accuracy: 0.97
90 - 3s/epoch - 7ms/step
Epoch 17/50
469/469 - 3s - loss: 6.4377e-04 - accuracy: 0.9966 - val_loss: 0.0036 - val_accuracy: 0.97
67 - 3s/epoch - 7ms/step
Epoch 18/50
469/469 - 3s - loss: 6.1341e-04 - accuracy: 0.9968 - val_loss: 0.0031 - val_accuracy: 0.97
94 - 3s/epoch - 7ms/step
Epoch 19/50
469/469 - 3s - loss: 5.5232e-04 - accuracy: 0.9971 - val_loss: 0.0030 - val_accuracy: 0.98
10 - 3s/epoch - 7ms/step
Epoch 20/50
469/469 - 3s - loss: 4.7142e-04 - accuracy: 0.9976 - val_loss: 0.0028 - val_accuracy: 0.98
22 - 3s/epoch - 7ms/step
Epoch 21/50
469/469 - 3s - loss: 4.5216e-04 - accuracy: 0.9977 - val_loss: 0.0030 - val_accuracy: 0.98
00 - 3s/epoch - 7ms/step
Epoch 22/50
```

```
469/469 - 3s - loss: 4.7376e-04 - accuracy: 0.9975 - val_loss: 0.0030 - val_accuracy: 0.98
12 - 3s/epoch - 7ms/step
Epoch 23/50
469/469 - 3s - loss: 4.3100e-04 - accuracy: 0.9977 - val_loss: 0.0037 - val_accuracy: 0.97
64 - 3s/epoch - 7ms/step
Epoch 24/50
469/469 - 3s - loss: 3.6493e-04 - accuracy: 0.9982 - val_loss: 0.0031 - val_accuracy: 0.97
94 - 3s/epoch - 7ms/step
Epoch 25/50
469/469 - 3s - loss: 3.2368e-04 - accuracy: 0.9983 - val_loss: 0.0031 - val_accuracy: 0.97
97 - 3s/epoch - 7ms/step
Epoch 26/50
469/469 - 3s - loss: 3.6111e-04 - accuracy: 0.9981 - val_loss: 0.0029 - val_accuracy: 0.98
08 - 3s/epoch - 7ms/step
Epoch 27/50
469/469 - 3s - loss: 3.9014e-04 - accuracy: 0.9979 - val_loss: 0.0031 - val_accuracy: 0.97
97 - 3s/epoch - 7ms/step
Epoch 28/50
469/469 - 3s - loss: 3.7615e-04 - accuracy: 0.9980 - val_loss: 0.0031 - val_accuracy: 0.98
04 - 3s/epoch - 7ms/step
Epoch 29/50
469/469 - 3s - loss: 3.3346e-04 - accuracy: 0.9981 - val_loss: 0.0028 - val_accuracy: 0.98
18 - 3s/epoch - 7ms/step
Epoch 30/50
469/469 - 3s - loss: 3.1440e-04 - accuracy: 0.9984 - val_loss: 0.0029 - val_accuracy: 0.98
17 - 3s/epoch - 7ms/step
Epoch 31/50
469/469 - 3s - loss: 2.6213e-04 - accuracy: 0.9987 - val_loss: 0.0027 - val_accuracy: 0.98
26 - 3s/epoch - 7ms/step
Epoch 32/50
469/469 - 3s - loss: 4.1449e-04 - accuracy: 0.9977 - val_loss: 0.0038 - val_accuracy: 0.97
54 - 3s/epoch - 7ms/step
Epoch 33/50
469/469 - 3s - loss: 4.1087e-04 - accuracy: 0.9976 - val_loss: 0.0028 - val_accuracy: 0.98
22 - 3s/epoch - 7ms/step
Epoch 34/50
469/469 - 3s - loss: 2.2589e-04 - accuracy: 0.9988 - val_loss: 0.0027 - val_accuracy: 0.98
35 - 3s/epoch - 7ms/step
Epoch 35/50
469/469 - 3s - loss: 2.0407e-04 - accuracy: 0.9990 - val_loss: 0.0029 - val_accuracy: 0.98
11 - 3s/epoch - 7ms/step
Epoch 36/50
469/469 - 3s - loss: 2.6038e-04 - accuracy: 0.9986 - val_loss: 0.0029 - val_accuracy: 0.98
15 - 3s/epoch - 7ms/step
Epoch 37/50
469/469 - 3s - loss: 3.6641e-04 - accuracy: 0.9979 - val_loss: 0.0031 - val_accuracy: 0.98
07 - 3s/epoch - 7ms/step
Epoch 38/50
469/469 - 3s - loss: 3.1126e-04 - accuracy: 0.9983 - val_loss: 0.0030 - val_accuracy: 0.98
09 - 3s/epoch - 7ms/step
Epoch 39/50
469/469 - 3s - loss: 2.5608e-04 - accuracy: 0.9987 - val_loss: 0.0029 - val_accuracy: 0.98
23 - 3s/epoch - 7ms/step
Epoch 40/50
469/469 - 3s - loss: 2.3552e-04 - accuracy: 0.9988 - val_loss: 0.0029 - val_accuracy: 0.98
25 - 3s/epoch - 7ms/step
Epoch 41/50
469/469 - 3s - loss: 2.7913e-04 - accuracy: 0.9984 - val_loss: 0.0032 - val_accuracy: 0.98
00 - 3s/epoch - 7ms/step
Epoch 42/50
469/469 - 3s - loss: 2.8043e-04 - accuracy: 0.9985 - val_loss: 0.0032 - val_accuracy: 0.98
01 - 3s/epoch - 7ms/step
Epoch 43/50
469/469 - 3s - loss: 1.7909e-04 - accuracy: 0.9991 - val_loss: 0.0031 - val_accuracy: 0.98
02 - 3s/epoch - 7ms/step
```

```
Epoch 44/50
469/469 - 3s - loss: 1.9574e-04 - accuracy: 0.9991 - val_loss: 0.0028 - val_accuracy: 0.98
27 - 3s/epoch - 7ms/step
Epoch 45/50
469/469 - 3s - loss: 1.6017e-04 - accuracy: 0.9992 - val_loss: 0.0029 - val_accuracy: 0.98
21 - 3s/epoch - 7ms/step
Epoch 46/50
469/469 - 3s - loss: 2.9569e-04 - accuracy: 0.9984 - val_loss: 0.0036 - val_accuracy: 0.97
82 - 3s/epoch - 7ms/step
Epoch 47/50
469/469 - 3s - loss: 3.4853e-04 - accuracy: 0.9979 - val_loss: 0.0029 - val_accuracy: 0.98
21 - 3s/epoch - 7ms/step
Epoch 48/50
469/469 - 3s - loss: 2.2140e-04 - accuracy: 0.9988 - val_loss: 0.0031 - val_accuracy: 0.98
08 - 3s/epoch - 7ms/step
Epoch 49/50
469/469 - 3s - loss: 1.9096e-04 - accuracy: 0.9990 - val_loss: 0.0029 - val_accuracy: 0.98
16 - 3s/epoch - 7ms/step
Epoch 50/50
469/469 - 3s - loss: 2.0988e-04 - accuracy: 0.9988 - val_loss: 0.0033 - val_accuracy: 0.97
88 - 3s/epoch - 7ms/step
Out[26]:

<keras.callbacks.History at 0x14230f82e80>
```

In [27]:

```
res=mlp.evaluate(x_test,y_test,verbose=0)
print('정확률=',res[1]*100)
```

```
정확률= 97.87999987602234
```

성능 시각화

SGD와 Adam 성능을 그래프로 비교

In [28]:

```
import numpy as np
import tensorflow as tf
import tensorflow.keras.datasets as ds
```

In [29]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import SGD, Adam
```

In [30]:

```
(x_train,y_train),(x_test,y_test)=ds.mnist.load_data()
```

In [31]:

```
x_train=x_train.reshape(60000,784)
x_test=x_test.reshape(10000,784)
x_train=x_train.astype(np.float32)/255.0
x_test=x_test.astype(np.float32)/255.0
y_train=tf.keras.utils.to_categorical(y_train,10)
y_test=tf.keras.utils.to_categorical(y_test,10)
```

In [32]:

```
mlp_sgd=Sequential()
mlp_sgd.add(Dense(units=512,activation='tanh',input_shape=(784,)))
mlp_sgd.add(Dense(units=10,activation='softmax'))
```

In [33]:

```
mlp_sgd.compile(loss='MSE',optimizer=SGD(learning_rate=0.01),metrics=['accuracy'])
```

In [34]:

```
hist_sgd=mlp_sgd.fit(x_train,y_train,batch_size=128,epochs=50,validation_data=(x_test,y_test),verbose=2)
```

```
Epoch 1/50
469/469 - 3s - loss: 0.0868 - accuracy: 0.2467 - val_loss: 0.0832 - val_accuracy: 0.3293 -
3s/epoch - 6ms/step
Epoch 2/50
469/469 - 2s - loss: 0.0797 - accuracy: 0.3849 - val_loss: 0.0761 - val_accuracy: 0.4399 -
2s/epoch - 5ms/step
Epoch 3/50
469/469 - 2s - loss: 0.0728 - accuracy: 0.4882 - val_loss: 0.0691 - val_accuracy: 0.5386 -
2s/epoch - 5ms/step
Epoch 4/50
469/469 - 2s - loss: 0.0661 - accuracy: 0.5870 - val_loss: 0.0622 - val_accuracy: 0.6389 -
2s/epoch - 5ms/step
Epoch 5/50
469/469 - 2s - loss: 0.0595 - accuracy: 0.6668 - val_loss: 0.0559 - val_accuracy: 0.6975 -
2s/epoch - 5ms/step
Epoch 6/50
469/469 - 2s - loss: 0.0537 - accuracy: 0.7092 - val_loss: 0.0505 - val_accuracy: 0.7396 -
2s/epoch - 5ms/step
Epoch 7/50
469/469 - 2s - loss: 0.0490 - accuracy: 0.7429 - val_loss: 0.0460 - val_accuracy: 0.7678 -
2s/epoch - 5ms/step
Epoch 8/50
469/469 - 2s - loss: 0.0450 - accuracy: 0.7710 - val_loss: 0.0424 - val_accuracy: 0.7919 -
2s/epoch - 5ms/step
Epoch 9/50
469/469 - 2s - loss: 0.0418 - accuracy: 0.7899 - val_loss: 0.0394 - val_accuracy: 0.8090 -
2s/epoch - 5ms/step
Epoch 10/50
469/469 - 2s - loss: 0.0391 - accuracy: 0.8029 - val_loss: 0.0368 - val_accuracy: 0.8198 -
2s/epoch - 5ms/step
Epoch 11/50
469/469 - 2s - loss: 0.0369 - accuracy: 0.8127 - val_loss: 0.0348 - val_accuracy: 0.8292 -
2s/epoch - 5ms/step
Epoch 12/50
469/469 - 2s - loss: 0.0350 - accuracy: 0.8209 - val_loss: 0.0330 - val_accuracy: 0.8367 -
2s/epoch - 5ms/step
Epoch 13/50
469/469 - 2s - loss: 0.0334 - accuracy: 0.8272 - val_loss: 0.0315 - val_accuracy: 0.8407 -
2s/epoch - 5ms/step
Epoch 14/50
469/469 - 2s - loss: 0.0320 - accuracy: 0.8322 - val_loss: 0.0302 - val_accuracy: 0.8455 -
2s/epoch - 5ms/step
Epoch 15/50
469/469 - 2s - loss: 0.0308 - accuracy: 0.8370 - val_loss: 0.0291 - val_accuracy: 0.8505 -
2s/epoch - 5ms/step
Epoch 16/50
469/469 - 2s - loss: 0.0297 - accuracy: 0.8416 - val_loss: 0.0281 - val_accuracy: 0.8553 -
2s/epoch - 5ms/step
Epoch 17/50
469/469 - 2s - loss: 0.0288 - accuracy: 0.8453 - val_loss: 0.0272 - val_accuracy: 0.8582 -
2s/epoch - 5ms/step
Epoch 18/50
469/469 - 2s - loss: 0.0280 - accuracy: 0.8489 - val_loss: 0.0264 - val_accuracy: 0.8600 -
2s/epoch - 5ms/step
Epoch 19/50
469/469 - 2s - loss: 0.0273 - accuracy: 0.8517 - val_loss: 0.0258 - val_accuracy: 0.8617 -
2s/epoch - 5ms/step
Epoch 20/50
469/469 - 2s - loss: 0.0266 - accuracy: 0.8543 - val_loss: 0.0251 - val_accuracy: 0.8647 -
2s/epoch - 5ms/step
Epoch 21/50
469/469 - 2s - loss: 0.0260 - accuracy: 0.8565 - val_loss: 0.0246 - val_accuracy: 0.8668 -
2s/epoch - 5ms/step
Epoch 22/50
```

```
469/469 - 2s - loss: 0.0255 - accuracy: 0.8590 - val_loss: 0.0241 - val_accuracy: 0.8687 -
2s/epoch - 5ms/step
Epoch 23/50
469/469 - 2s - loss: 0.0250 - accuracy: 0.8608 - val_loss: 0.0236 - val_accuracy: 0.8708 -
2s/epoch - 5ms/step
Epoch 24/50
469/469 - 2s - loss: 0.0245 - accuracy: 0.8629 - val_loss: 0.0232 - val_accuracy: 0.8724 -
2s/epoch - 5ms/step
Epoch 25/50
469/469 - 2s - loss: 0.0241 - accuracy: 0.8644 - val_loss: 0.0228 - val_accuracy: 0.8743 -
2s/epoch - 5ms/step
Epoch 26/50
469/469 - 2s - loss: 0.0237 - accuracy: 0.8662 - val_loss: 0.0224 - val_accuracy: 0.8756 -
2s/epoch - 5ms/step
Epoch 27/50
469/469 - 2s - loss: 0.0233 - accuracy: 0.8678 - val_loss: 0.0220 - val_accuracy: 0.8765 -
2s/epoch - 5ms/step
Epoch 28/50
469/469 - 2s - loss: 0.0230 - accuracy: 0.8690 - val_loss: 0.0217 - val_accuracy: 0.8782 -
2s/epoch - 5ms/step
Epoch 29/50
469/469 - 2s - loss: 0.0227 - accuracy: 0.8703 - val_loss: 0.0214 - val_accuracy: 0.8795 -
2s/epoch - 5ms/step
Epoch 30/50
469/469 - 2s - loss: 0.0224 - accuracy: 0.8711 - val_loss: 0.0211 - val_accuracy: 0.8813 -
2s/epoch - 5ms/step
Epoch 31/50
469/469 - 2s - loss: 0.0221 - accuracy: 0.8721 - val_loss: 0.0209 - val_accuracy: 0.8817 -
2s/epoch - 5ms/step
Epoch 32/50
469/469 - 2s - loss: 0.0218 - accuracy: 0.8734 - val_loss: 0.0206 - val_accuracy: 0.8820 -
2s/epoch - 5ms/step
Epoch 33/50
469/469 - 2s - loss: 0.0216 - accuracy: 0.8744 - val_loss: 0.0204 - val_accuracy: 0.8831 -
2s/epoch - 5ms/step
Epoch 34/50
469/469 - 2s - loss: 0.0214 - accuracy: 0.8754 - val_loss: 0.0202 - val_accuracy: 0.8839 -
2s/epoch - 5ms/step
Epoch 35/50
469/469 - 2s - loss: 0.0211 - accuracy: 0.8764 - val_loss: 0.0200 - val_accuracy: 0.8846 -
2s/epoch - 5ms/step
Epoch 36/50
469/469 - 2s - loss: 0.0209 - accuracy: 0.8773 - val_loss: 0.0198 - val_accuracy: 0.8852 -
2s/epoch - 5ms/step
Epoch 37/50
469/469 - 2s - loss: 0.0207 - accuracy: 0.8781 - val_loss: 0.0196 - val_accuracy: 0.8861 -
2s/epoch - 5ms/step
Epoch 38/50
469/469 - 2s - loss: 0.0205 - accuracy: 0.8793 - val_loss: 0.0194 - val_accuracy: 0.8872 -
2s/epoch - 5ms/step
Epoch 39/50
469/469 - 2s - loss: 0.0204 - accuracy: 0.8799 - val_loss: 0.0192 - val_accuracy: 0.8885 -
2s/epoch - 5ms/step
Epoch 40/50
469/469 - 2s - loss: 0.0202 - accuracy: 0.8808 - val_loss: 0.0191 - val_accuracy: 0.8892 -
2s/epoch - 5ms/step
Epoch 41/50
469/469 - 2s - loss: 0.0200 - accuracy: 0.8815 - val_loss: 0.0189 - val_accuracy: 0.8900 -
2s/epoch - 5ms/step
Epoch 42/50
469/469 - 2s - loss: 0.0199 - accuracy: 0.8822 - val_loss: 0.0187 - val_accuracy: 0.8913 -
2s/epoch - 5ms/step
Epoch 43/50
469/469 - 2s - loss: 0.0197 - accuracy: 0.8829 - val_loss: 0.0186 - val_accuracy: 0.8920 -
2s/epoch - 5ms/step
```

```
Epoch 44/50
469/469 - 2s - loss: 0.0195 - accuracy: 0.8837 - val_loss: 0.0185 - val_accuracy: 0.8925 -
2s/epoch - 5ms/step
Epoch 45/50
469/469 - 2s - loss: 0.0194 - accuracy: 0.8843 - val_loss: 0.0183 - val_accuracy: 0.8934 -
2s/epoch - 5ms/step
Epoch 46/50
469/469 - 2s - loss: 0.0193 - accuracy: 0.8849 - val_loss: 0.0182 - val_accuracy: 0.8939 -
2s/epoch - 5ms/step
Epoch 47/50
469/469 - 2s - loss: 0.0191 - accuracy: 0.8855 - val_loss: 0.0181 - val_accuracy: 0.8939 -
2s/epoch - 5ms/step
Epoch 48/50
469/469 - 2s - loss: 0.0190 - accuracy: 0.8863 - val_loss: 0.0180 - val_accuracy: 0.8944 -
2s/epoch - 5ms/step
Epoch 49/50
469/469 - 2s - loss: 0.0189 - accuracy: 0.8868 - val_loss: 0.0178 - val_accuracy: 0.8949 -
2s/epoch - 5ms/step
Epoch 50/50
469/469 - 2s - loss: 0.0188 - accuracy: 0.8874 - val_loss: 0.0177 - val_accuracy: 0.8955 -
2s/epoch - 5ms/step
```

In [35]:

```
print('SGD 정확률=',mlp_sgd.evaluate(x_test,y_test,verbose=0)[1]*100)
```

```
SGD 정확률= 89.55000042915344
```

In [37]:

```
mlp_adam=Sequential()
mlp_adam.add(Dense(units=512,activation='tanh',input_shape=(784,)))
mlp_adam.add(Dense(units=10,activation='softmax'))
```

In [38]:

```
mlp_adam.compile(loss='MSE',optimizer=Adam(learning_rate=0.001),metrics=['accuracy'])
```

In [39]:

```
hist_adam=mlp_adam.fit(x_train,y_train,batch_size=128,epochs=50,validation_data=(x_test,y_test),verbose=2
```

```
Epoch 1/50
469/469 - 4s - loss: 0.0149 - accuracy: 0.9005 - val_loss: 0.0103 - val_accuracy: 0.9332 -
4s/epoch - 8ms/step
Epoch 2/50
469/469 - 3s - loss: 0.0089 - accuracy: 0.9421 - val_loss: 0.0076 - val_accuracy: 0.9498 -
3s/epoch - 7ms/step
Epoch 3/50
469/469 - 3s - loss: 0.0066 - accuracy: 0.9580 - val_loss: 0.0061 - val_accuracy: 0.9619 -
3s/epoch - 7ms/step
Epoch 4/50
469/469 - 3s - loss: 0.0051 - accuracy: 0.9682 - val_loss: 0.0052 - val_accuracy: 0.9673 -
3s/epoch - 7ms/step
Epoch 5/50
469/469 - 3s - loss: 0.0041 - accuracy: 0.9746 - val_loss: 0.0049 - val_accuracy: 0.9670 -
3s/epoch - 7ms/step
Epoch 6/50
469/469 - 3s - loss: 0.0033 - accuracy: 0.9800 - val_loss: 0.0045 - val_accuracy: 0.9704 -
3s/epoch - 7ms/step
Epoch 7/50
469/469 - 3s - loss: 0.0029 - accuracy: 0.9835 - val_loss: 0.0041 - val_accuracy: 0.9729 -
3s/epoch - 7ms/step
Epoch 8/50
469/469 - 3s - loss: 0.0024 - accuracy: 0.9859 - val_loss: 0.0040 - val_accuracy: 0.9737 -
3s/epoch - 7ms/step
Epoch 9/50
469/469 - 3s - loss: 0.0020 - accuracy: 0.9882 - val_loss: 0.0039 - val_accuracy: 0.9743 -
3s/epoch - 7ms/step
Epoch 10/50
469/469 - 3s - loss: 0.0017 - accuracy: 0.9904 - val_loss: 0.0035 - val_accuracy: 0.9774 -
3s/epoch - 7ms/step
Epoch 11/50
469/469 - 3s - loss: 0.0015 - accuracy: 0.9919 - val_loss: 0.0035 - val_accuracy: 0.9771 -
3s/epoch - 7ms/step
Epoch 12/50
469/469 - 3s - loss: 0.0013 - accuracy: 0.9930 - val_loss: 0.0031 - val_accuracy: 0.9806 -
3s/epoch - 7ms/step
Epoch 13/50
469/469 - 3s - loss: 0.0011 - accuracy: 0.9941 - val_loss: 0.0034 - val_accuracy: 0.9781 -
3s/epoch - 7ms/step
Epoch 14/50
469/469 - 3s - loss: 0.0010 - accuracy: 0.9944 - val_loss: 0.0031 - val_accuracy: 0.9800 -
3s/epoch - 7ms/step
Epoch 15/50
469/469 - 3s - loss: 8.3910e-04 - accuracy: 0.9958 - val_loss: 0.0032 - val_accuracy: 0.97
92 - 3s/epoch - 7ms/step
Epoch 16/50
469/469 - 3s - loss: 8.4556e-04 - accuracy: 0.9954 - val_loss: 0.0030 - val_accuracy: 0.98
01 - 3s/epoch - 7ms/step
Epoch 17/50
469/469 - 3s - loss: 6.7372e-04 - accuracy: 0.9966 - val_loss: 0.0030 - val_accuracy: 0.98
10 - 3s/epoch - 7ms/step
Epoch 18/50
469/469 - 3s - loss: 5.8215e-04 - accuracy: 0.9971 - val_loss: 0.0029 - val_accuracy: 0.98
07 - 3s/epoch - 7ms/step
Epoch 19/50
469/469 - 3s - loss: 5.0395e-04 - accuracy: 0.9975 - val_loss: 0.0030 - val_accuracy: 0.98
05 - 3s/epoch - 7ms/step
Epoch 20/50
469/469 - 3s - loss: 5.6054e-04 - accuracy: 0.9970 - val_loss: 0.0033 - val_accuracy: 0.97
74 - 3s/epoch - 7ms/step
Epoch 21/50
469/469 - 3s - loss: 4.0770e-04 - accuracy: 0.9980 - val_loss: 0.0028 - val_accuracy: 0.98
20 - 3s/epoch - 7ms/step
Epoch 22/50
```

```
469/469 - 3s - loss: 3.6059e-04 - accuracy: 0.9982 - val_loss: 0.0031 - val_accuracy: 0.97
99 - 3s/epoch - 7ms/step
Epoch 23/50
469/469 - 3s - loss: 5.5558e-04 - accuracy: 0.9970 - val_loss: 0.0030 - val_accuracy: 0.98
07 - 3s/epoch - 7ms/step
Epoch 24/50
469/469 - 3s - loss: 4.0364e-04 - accuracy: 0.9980 - val_loss: 0.0031 - val_accuracy: 0.98
00 - 3s/epoch - 7ms/step
Epoch 25/50
469/469 - 3s - loss: 4.1715e-04 - accuracy: 0.9977 - val_loss: 0.0029 - val_accuracy: 0.98
10 - 3s/epoch - 7ms/step
Epoch 26/50
469/469 - 3s - loss: 3.2047e-04 - accuracy: 0.9983 - val_loss: 0.0030 - val_accuracy: 0.98
05 - 3s/epoch - 7ms/step
Epoch 27/50
469/469 - 3s - loss: 2.8144e-04 - accuracy: 0.9985 - val_loss: 0.0029 - val_accuracy: 0.98
12 - 3s/epoch - 7ms/step
Epoch 28/50
469/469 - 3s - loss: 2.6144e-04 - accuracy: 0.9987 - val_loss: 0.0030 - val_accuracy: 0.98
07 - 3s/epoch - 7ms/step
Epoch 29/50
469/469 - 3s - loss: 4.1562e-04 - accuracy: 0.9977 - val_loss: 0.0030 - val_accuracy: 0.98
06 - 3s/epoch - 7ms/step
Epoch 30/50
469/469 - 3s - loss: 3.9382e-04 - accuracy: 0.9978 - val_loss: 0.0032 - val_accuracy: 0.97
99 - 3s/epoch - 7ms/step
Epoch 31/50
469/469 - 3s - loss: 2.4956e-04 - accuracy: 0.9988 - val_loss: 0.0029 - val_accuracy: 0.98
09 - 3s/epoch - 7ms/step
Epoch 32/50
469/469 - 3s - loss: 2.2869e-04 - accuracy: 0.9988 - val_loss: 0.0029 - val_accuracy: 0.98
13 - 3s/epoch - 7ms/step
Epoch 33/50
469/469 - 3s - loss: 3.1255e-04 - accuracy: 0.9983 - val_loss: 0.0031 - val_accuracy: 0.97
97 - 3s/epoch - 7ms/step
Epoch 34/50
469/469 - 3s - loss: 4.1512e-04 - accuracy: 0.9976 - val_loss: 0.0029 - val_accuracy: 0.98
14 - 3s/epoch - 7ms/step
Epoch 35/50
469/469 - 3s - loss: 2.3323e-04 - accuracy: 0.9987 - val_loss: 0.0030 - val_accuracy: 0.98
09 - 3s/epoch - 7ms/step
Epoch 36/50
469/469 - 3s - loss: 1.7520e-04 - accuracy: 0.9991 - val_loss: 0.0032 - val_accuracy: 0.97
97 - 3s/epoch - 7ms/step
Epoch 37/50
469/469 - 3s - loss: 1.8398e-04 - accuracy: 0.9990 - val_loss: 0.0029 - val_accuracy: 0.98
19 - 3s/epoch - 7ms/step
Epoch 38/50
469/469 - 3s - loss: 2.0904e-04 - accuracy: 0.9989 - val_loss: 0.0029 - val_accuracy: 0.98
14 - 3s/epoch - 7ms/step
Epoch 39/50
469/469 - 3s - loss: 4.8881e-04 - accuracy: 0.9970 - val_loss: 0.0033 - val_accuracy: 0.97
89 - 3s/epoch - 7ms/step
Epoch 40/50
469/469 - 3s - loss: 2.5970e-04 - accuracy: 0.9985 - val_loss: 0.0032 - val_accuracy: 0.97
89 - 3s/epoch - 7ms/step
Epoch 41/50
469/469 - 3s - loss: 1.7350e-04 - accuracy: 0.9991 - val_loss: 0.0028 - val_accuracy: 0.98
18 - 3s/epoch - 7ms/step
Epoch 42/50
469/469 - 3s - loss: 1.6968e-04 - accuracy: 0.9991 - val_loss: 0.0030 - val_accuracy: 0.98
05 - 3s/epoch - 7ms/step
Epoch 43/50
469/469 - 3s - loss: 1.7187e-04 - accuracy: 0.9991 - val_loss: 0.0030 - val_accuracy: 0.98
02 - 3s/epoch - 7ms/step
```

```
Epoch 44/50
469/469 - 3s - loss: 4.0703e-04 - accuracy: 0.9976 - val_loss: 0.0036 - val_accuracy: 0.97
71 - 3s/epoch - 7ms/step
Epoch 45/50
469/469 - 3s - loss: 3.0251e-04 - accuracy: 0.9984 - val_loss: 0.0031 - val_accuracy: 0.98
20 - 3s/epoch - 7ms/step
Epoch 46/50
469/469 - 3s - loss: 2.0074e-04 - accuracy: 0.9989 - val_loss: 0.0030 - val_accuracy: 0.98
19 - 3s/epoch - 7ms/step
Epoch 47/50
469/469 - 3s - loss: 1.7614e-04 - accuracy: 0.9991 - val_loss: 0.0031 - val_accuracy: 0.98
07 - 3s/epoch - 7ms/step
Epoch 48/50
469/469 - 3s - loss: 1.7081e-04 - accuracy: 0.9991 - val_loss: 0.0030 - val_accuracy: 0.98
15 - 3s/epoch - 7ms/step
Epoch 49/50
469/469 - 3s - loss: 3.4651e-04 - accuracy: 0.9980 - val_loss: 0.0035 - val_accuracy: 0.97
90 - 3s/epoch - 7ms/step
Epoch 50/50
469/469 - 3s - loss: 3.2004e-04 - accuracy: 0.9983 - val_loss: 0.0035 - val_accuracy: 0.97
88 - 3s/epoch - 7ms/step
```

In [40]:

```
print('Adam 정확률=',mlp_adam.evaluate(x_test,y_test,verbose=0)[1]*100)
```

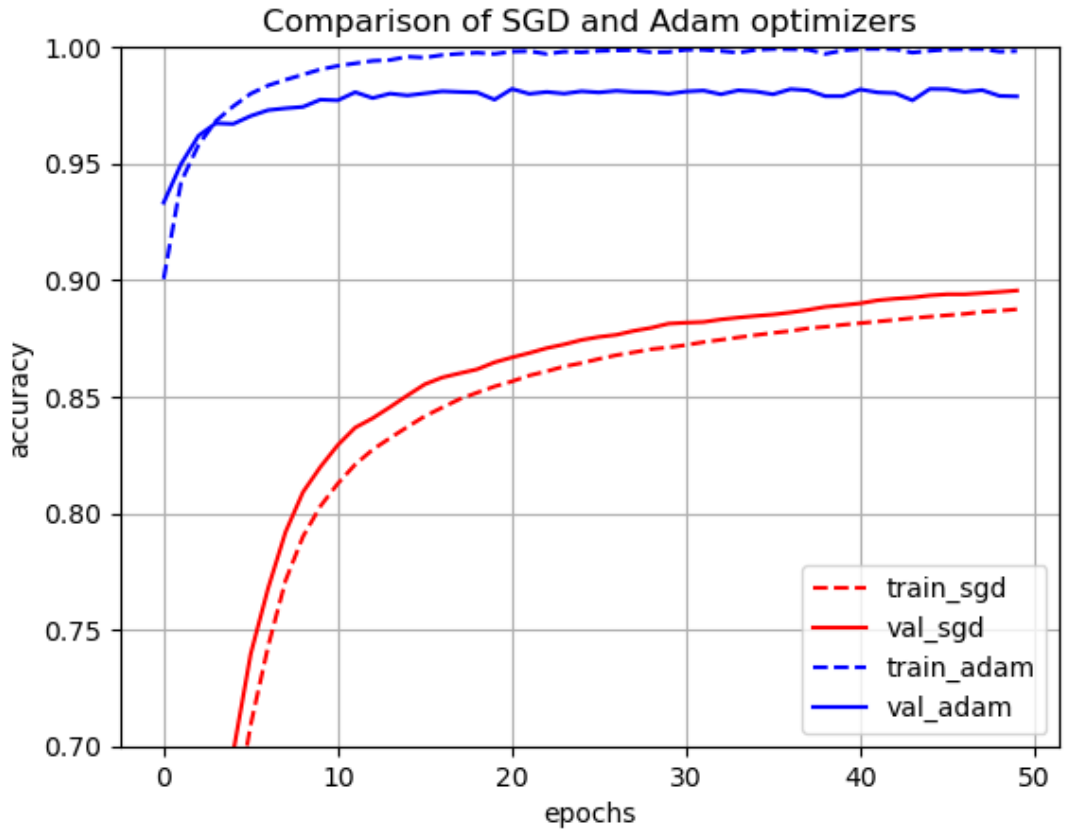
```
Adam 정확률= 97.87999987602234
```

In [41]:

```
import matplotlib.pyplot as plt
```

In [42]:

```
plt.plot(hist_sgd.history['accuracy'],'r--')
plt.plot(hist_sgd.history['val_accuracy'],'r')
plt.plot(hist_adam.history['accuracy'],'b--')
plt.plot(hist_adam.history['val_accuracy'],'b')
plt.title('Comparison of SGD and Adam optimizers')
plt.ylim((0.7,1.0))
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend(['train_sgd','val_sgd','train_adam','val_adam'])
plt.grid()
plt.show()
```



하이퍼 매개변수 다루기

깊은 다층 퍼셉트론으로 MNIST 인식

In [43]:

```
import numpy as np
import tensorflow as tf
import tensorflow.keras.datasets as ds
```

In [44]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
```

In [45]:

```
(x_train,y_train),(x_test,y_test)=ds.mnist.load_data()
```

In [46]:

```
x_train=x_train.reshape(60000,784)
x_test=x_test.reshape(10000,784)
x_train=x_train.astype(np.float32)/255.0
x_test=x_test.astype(np.float32)/255.0
y_train=tf.keras.utils.to_categorical(y_train,10)
y_test=tf.keras.utils.to_categorical(y_test,10)
```

In [47]:

```
dmlp=Sequential()
dmlp.add(Dense(units=1024,activation='relu',input_shape=(784,)))
dmlp.add(Dense(units=512,activation='relu'))
dmlp.add(Dense(units=512,activation='relu'))
dmlp.add(Dense(units=10,activation='softmax'))
```

In [48]:

```
dmlp.compile(loss='categorical_crossentropy',optimizer=Adam(learning_rate=0.0001),metrics=['accuracy'])
```

In [49]:

```
hist=dmlp.fit(x_train,y_train,batch_size=128,epochs=50,validation_data=(x_test,y_test),verbose=2)
```

```
Epoch 1/50
469/469 - 11s - loss: 0.3959 - accuracy: 0.8977 - val_loss: 0.1723 - val_accuracy: 0.9493
- 11s/epoch - 23ms/step
Epoch 2/50
469/469 - 11s - loss: 0.1440 - accuracy: 0.9586 - val_loss: 0.1208 - val_accuracy: 0.9630
- 11s/epoch - 24ms/step
Epoch 3/50
469/469 - 11s - loss: 0.0979 - accuracy: 0.9713 - val_loss: 0.0952 - val_accuracy: 0.9707
- 11s/epoch - 23ms/step
Epoch 4/50
469/469 - 11s - loss: 0.0711 - accuracy: 0.9796 - val_loss: 0.0850 - val_accuracy: 0.9730
- 11s/epoch - 23ms/step
Epoch 5/50
469/469 - 10s - loss: 0.0540 - accuracy: 0.9844 - val_loss: 0.0755 - val_accuracy: 0.9769
- 10s/epoch - 22ms/step
Epoch 6/50
469/469 - 12s - loss: 0.0415 - accuracy: 0.9880 - val_loss: 0.0675 - val_accuracy: 0.9789
- 12s/epoch - 25ms/step
Epoch 7/50
469/469 - 11s - loss: 0.0322 - accuracy: 0.9911 - val_loss: 0.0681 - val_accuracy: 0.9767
- 11s/epoch - 23ms/step
Epoch 8/50
469/469 - 11s - loss: 0.0247 - accuracy: 0.9936 - val_loss: 0.0650 - val_accuracy: 0.9799
- 11s/epoch - 23ms/step
Epoch 9/50
469/469 - 11s - loss: 0.0187 - accuracy: 0.9952 - val_loss: 0.0605 - val_accuracy: 0.9799
- 11s/epoch - 23ms/step
Epoch 10/50
469/469 - 11s - loss: 0.0151 - accuracy: 0.9959 - val_loss: 0.0634 - val_accuracy: 0.9811
- 11s/epoch - 23ms/step
Epoch 11/50
469/469 - 11s - loss: 0.0100 - accuracy: 0.9977 - val_loss: 0.0696 - val_accuracy: 0.9812
- 11s/epoch - 23ms/step
Epoch 12/50
469/469 - 12s - loss: 0.0085 - accuracy: 0.9982 - val_loss: 0.0670 - val_accuracy: 0.9807
- 12s/epoch - 26ms/step
Epoch 13/50
469/469 - 11s - loss: 0.0068 - accuracy: 0.9984 - val_loss: 0.0732 - val_accuracy: 0.9791
- 11s/epoch - 23ms/step
Epoch 14/50
469/469 - 11s - loss: 0.0057 - accuracy: 0.9987 - val_loss: 0.0682 - val_accuracy: 0.9825
- 11s/epoch - 23ms/step
Epoch 15/50
469/469 - 11s - loss: 0.0048 - accuracy: 0.9990 - val_loss: 0.0662 - val_accuracy: 0.9821
- 11s/epoch - 23ms/step
Epoch 16/50
469/469 - 10s - loss: 0.0024 - accuracy: 0.9997 - val_loss: 0.0777 - val_accuracy: 0.9799
- 10s/epoch - 22ms/step
Epoch 17/50
469/469 - 10s - loss: 0.0050 - accuracy: 0.9987 - val_loss: 0.0899 - val_accuracy: 0.9768
- 10s/epoch - 22ms/step
Epoch 18/50
469/469 - 11s - loss: 0.0063 - accuracy: 0.9980 - val_loss: 0.0955 - val_accuracy: 0.9766
- 11s/epoch - 23ms/step
Epoch 19/50
469/469 - 11s - loss: 0.0033 - accuracy: 0.9992 - val_loss: 0.0891 - val_accuracy: 0.9796
- 11s/epoch - 23ms/step
Epoch 20/50
469/469 - 11s - loss: 0.0038 - accuracy: 0.9990 - val_loss: 0.0754 - val_accuracy: 0.9830
- 11s/epoch - 23ms/step
Epoch 21/50
469/469 - 11s - loss: 0.0020 - accuracy: 0.9996 - val_loss: 0.0768 - val_accuracy: 0.9820
- 11s/epoch - 24ms/step
Epoch 22/50
```

```
469/469 - 11s - loss: 6.9245e-04 - accuracy: 0.9999 - val_loss: 0.0746 - val_accuracy: 0.9
832 - 11s/epoch - 23ms/step
Epoch 23/50
469/469 - 11s - loss: 2.6271e-04 - accuracy: 1.0000 - val_loss: 0.0738 - val_accuracy: 0.9
832 - 11s/epoch - 23ms/step
Epoch 24/50
469/469 - 11s - loss: 1.7728e-04 - accuracy: 1.0000 - val_loss: 0.0751 - val_accuracy: 0.9
834 - 11s/epoch - 23ms/step
Epoch 25/50
469/469 - 11s - loss: 1.3679e-04 - accuracy: 1.0000 - val_loss: 0.0776 - val_accuracy: 0.9
838 - 11s/epoch - 23ms/step
Epoch 26/50
469/469 - 11s - loss: 1.0989e-04 - accuracy: 1.0000 - val_loss: 0.0786 - val_accuracy: 0.9
834 - 11s/epoch - 24ms/step
Epoch 27/50
469/469 - 11s - loss: 1.0044e-04 - accuracy: 1.0000 - val_loss: 0.0813 - val_accuracy: 0.9
836 - 11s/epoch - 22ms/step
Epoch 28/50
469/469 - 10s - loss: 3.2653e-04 - accuracy: 0.9999 - val_loss: 0.1262 - val_accuracy: 0.9
773 - 10s/epoch - 22ms/step
Epoch 29/50
469/469 - 10s - loss: 0.0159 - accuracy: 0.9950 - val_loss: 0.0825 - val_accuracy: 0.9815
- 10s/epoch - 22ms/step
Epoch 30/50
469/469 - 11s - loss: 0.0019 - accuracy: 0.9996 - val_loss: 0.0803 - val_accuracy: 0.9821
- 11s/epoch - 23ms/step
Epoch 31/50
469/469 - 11s - loss: 3.5704e-04 - accuracy: 1.0000 - val_loss: 0.0763 - val_accuracy: 0.9
836 - 11s/epoch - 23ms/step
Epoch 32/50
469/469 - 11s - loss: 1.6828e-04 - accuracy: 1.0000 - val_loss: 0.0773 - val_accuracy: 0.9
832 - 11s/epoch - 23ms/step
Epoch 33/50
469/469 - 11s - loss: 1.1934e-04 - accuracy: 1.0000 - val_loss: 0.0794 - val_accuracy: 0.9
838 - 11s/epoch - 23ms/step
Epoch 34/50
469/469 - 11s - loss: 9.2154e-05 - accuracy: 1.0000 - val_loss: 0.0808 - val_accuracy: 0.9
834 - 11s/epoch - 23ms/step
Epoch 35/50
469/469 - 11s - loss: 7.2041e-05 - accuracy: 1.0000 - val_loss: 0.0825 - val_accuracy: 0.9
836 - 11s/epoch - 23ms/step
Epoch 36/50
469/469 - 11s - loss: 5.7927e-05 - accuracy: 1.0000 - val_loss: 0.0841 - val_accuracy: 0.9
832 - 11s/epoch - 23ms/step
Epoch 37/50
469/469 - 11s - loss: 4.7176e-05 - accuracy: 1.0000 - val_loss: 0.0857 - val_accuracy: 0.9
832 - 11s/epoch - 23ms/step
Epoch 38/50
469/469 - 11s - loss: 3.8735e-05 - accuracy: 1.0000 - val_loss: 0.0877 - val_accuracy: 0.9
836 - 11s/epoch - 23ms/step
Epoch 39/50
469/469 - 11s - loss: 3.1165e-05 - accuracy: 1.0000 - val_loss: 0.0886 - val_accuracy: 0.9
832 - 11s/epoch - 23ms/step
Epoch 40/50
469/469 - 11s - loss: 2.5124e-05 - accuracy: 1.0000 - val_loss: 0.0909 - val_accuracy: 0.9
834 - 11s/epoch - 23ms/step
Epoch 41/50
469/469 - 11s - loss: 2.2874e-05 - accuracy: 1.0000 - val_loss: 0.0921 - val_accuracy: 0.9
835 - 11s/epoch - 23ms/step
Epoch 42/50
469/469 - 11s - loss: 1.7786e-05 - accuracy: 1.0000 - val_loss: 0.0943 - val_accuracy: 0.9
829 - 11s/epoch - 23ms/step
Epoch 43/50
469/469 - 11s - loss: 1.4082e-05 - accuracy: 1.0000 - val_loss: 0.0958 - val_accuracy: 0.9
835 - 11s/epoch - 24ms/step
```

```
Epoch 44/50
469/469 - 11s - loss: 0.0105 - accuracy: 0.9972 - val_loss: 0.1046 - val_accuracy: 0.9768
- 11s/epoch - 23ms/step
Epoch 45/50
469/469 - 11s - loss: 0.0056 - accuracy: 0.9983 - val_loss: 0.0898 - val_accuracy: 0.9809
- 11s/epoch - 23ms/step
Epoch 46/50
469/469 - 11s - loss: 7.2811e-04 - accuracy: 0.9999 - val_loss: 0.0814 - val_accuracy: 0.9
826 - 11s/epoch - 23ms/step
Epoch 47/50
469/469 - 11s - loss: 1.8869e-04 - accuracy: 1.0000 - val_loss: 0.0827 - val_accuracy: 0.9
833 - 11s/epoch - 23ms/step
Epoch 48/50
469/469 - 11s - loss: 9.3052e-05 - accuracy: 1.0000 - val_loss: 0.0842 - val_accuracy: 0.9
830 - 11s/epoch - 23ms/step
Epoch 49/50
469/469 - 11s - loss: 6.9476e-05 - accuracy: 1.0000 - val_loss: 0.0861 - val_accuracy: 0.9
838 - 11s/epoch - 23ms/step
Epoch 50/50
469/469 - 11s - loss: 5.3241e-05 - accuracy: 1.0000 - val_loss: 0.0876 - val_accuracy: 0.9
837 - 11s/epoch - 23ms/step
```

In [50]:

```
print('정확률=', dmlp.evaluate(x_test,y_test,verbose=0)[1]*100)
```

정확률= 98.36999773979187

In [51]:

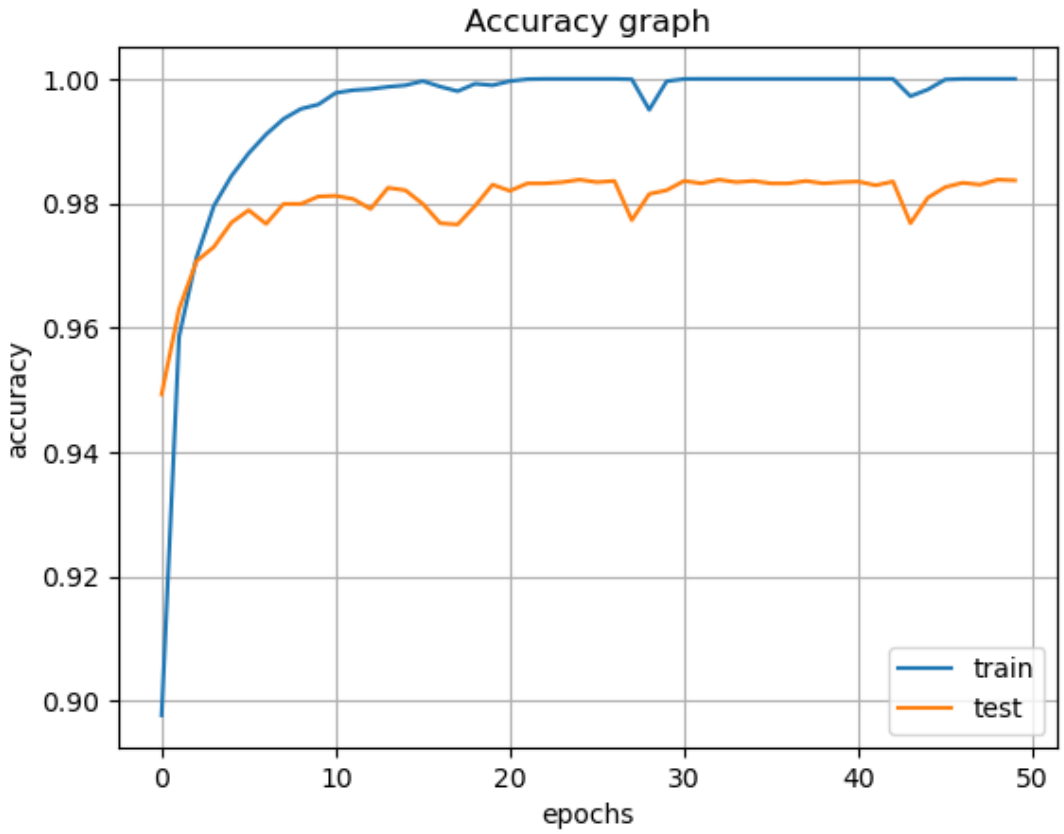
```
dmlp.save('dmlp_trained.h5')
```

In [52]:

```
import matplotlib.pyplot as plt
```

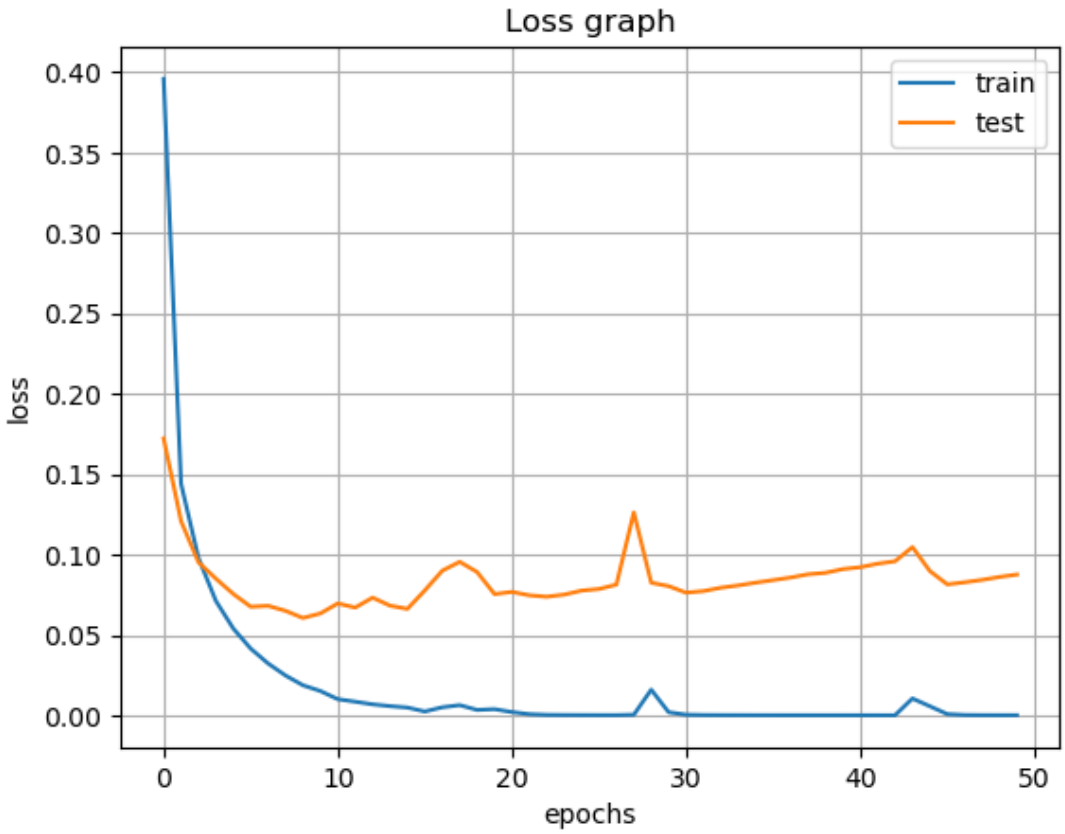
In [54]:

```
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.title('Accuracy graph')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend(['train','test'])
plt.grid()
plt.show()
```



In [55]:

```
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Loss graph')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend(['train','test'])
plt.grid()
plt.show()
```



자연 영상 인식

깊은 다층 퍼셉트론으로 CIFAR-10 인식하기

In [56]:

```
import numpy as np
import tensorflow as tf
import tensorflow.keras.datasets as ds
```

In [57]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
```

In [58]:

```
(x_train,y_train),(x_test,y_test)=ds.cifar10.load_data()
```

In [59]:

```
x_train=x_train.reshape(50000,3072)
x_test=x_test.reshape(10000,3072)
x_train=x_train.astype(np.float32)/255.0
x_test=x_test.astype(np.float32)/255.0
y_train=tf.keras.utils.to_categorical(y_train,10)
y_test=tf.keras.utils.to_categorical(y_test,10)
```

In [60]:

```
dmlp=Sequential()
dmlp.add(Dense(units=1024,activation='relu',input_shape=(3072,)))
dmlp.add(Dense(units=512,activation='relu'))
dmlp.add(Dense(units=512,activation='relu'))
dmlp.add(Dense(units=10,activation='softmax'))
```

In [61]:

```
dmlp.compile(loss='categorical_crossentropy',optimizer=Adam(learning_rate=0.0001),metrics=['accuracy'])
```

In [62]:

```
hist=dmlp.fit(x_train,y_train,batch_size=128,epochs=50,validation_data=(x_test,y_test),verbose=2)
```

```
Epoch 1/50
391/391 - 18s - loss: 1.8117 - accuracy: 0.3558 - val_loss: 1.6582 - val_accuracy: 0.4110
- 18s/epoch - 46ms/step
Epoch 2/50
391/391 - 17s - loss: 1.6174 - accuracy: 0.4261 - val_loss: 1.5642 - val_accuracy: 0.4452
- 17s/epoch - 43ms/step
Epoch 3/50
391/391 - 17s - loss: 1.5321 - accuracy: 0.4567 - val_loss: 1.5105 - val_accuracy: 0.4669
- 17s/epoch - 44ms/step
Epoch 4/50
391/391 - 18s - loss: 1.4739 - accuracy: 0.4766 - val_loss: 1.4734 - val_accuracy: 0.4776
- 18s/epoch - 46ms/step
Epoch 5/50
391/391 - 18s - loss: 1.4257 - accuracy: 0.4936 - val_loss: 1.4513 - val_accuracy: 0.4841
- 18s/epoch - 46ms/step
Epoch 6/50
391/391 - 18s - loss: 1.3828 - accuracy: 0.5107 - val_loss: 1.4476 - val_accuracy: 0.4872
- 18s/epoch - 46ms/step
Epoch 7/50
391/391 - 17s - loss: 1.3367 - accuracy: 0.5269 - val_loss: 1.4113 - val_accuracy: 0.4947
- 17s/epoch - 45ms/step
Epoch 8/50
391/391 - 18s - loss: 1.3045 - accuracy: 0.5388 - val_loss: 1.3852 - val_accuracy: 0.5059
- 18s/epoch - 45ms/step
Epoch 9/50
391/391 - 18s - loss: 1.2689 - accuracy: 0.5519 - val_loss: 1.3628 - val_accuracy: 0.5140
- 18s/epoch - 46ms/step
Epoch 10/50
391/391 - 22s - loss: 1.2393 - accuracy: 0.5629 - val_loss: 1.3605 - val_accuracy: 0.5171
- 22s/epoch - 56ms/step
Epoch 11/50
391/391 - 19s - loss: 1.2058 - accuracy: 0.5745 - val_loss: 1.3199 - val_accuracy: 0.5336
- 19s/epoch - 50ms/step
Epoch 12/50
391/391 - 19s - loss: 1.1798 - accuracy: 0.5860 - val_loss: 1.3259 - val_accuracy: 0.5286
- 19s/epoch - 48ms/step
Epoch 13/50
391/391 - 18s - loss: 1.1438 - accuracy: 0.5972 - val_loss: 1.3212 - val_accuracy: 0.5323
- 18s/epoch - 47ms/step
Epoch 14/50
391/391 - 18s - loss: 1.1152 - accuracy: 0.6091 - val_loss: 1.3065 - val_accuracy: 0.5396
- 18s/epoch - 46ms/step
Epoch 15/50
391/391 - 19s - loss: 1.0882 - accuracy: 0.6188 - val_loss: 1.3512 - val_accuracy: 0.5308
- 19s/epoch - 48ms/step
Epoch 16/50
391/391 - 18s - loss: 1.0591 - accuracy: 0.6261 - val_loss: 1.3261 - val_accuracy: 0.5396
- 18s/epoch - 47ms/step
Epoch 17/50
391/391 - 18s - loss: 1.0334 - accuracy: 0.6363 - val_loss: 1.3255 - val_accuracy: 0.5348
- 18s/epoch - 46ms/step
Epoch 18/50
391/391 - 18s - loss: 1.0047 - accuracy: 0.6494 - val_loss: 1.2828 - val_accuracy: 0.5515
- 18s/epoch - 47ms/step
Epoch 19/50
391/391 - 18s - loss: 0.9787 - accuracy: 0.6551 - val_loss: 1.3055 - val_accuracy: 0.5397
- 18s/epoch - 47ms/step
Epoch 20/50
391/391 - 18s - loss: 0.9455 - accuracy: 0.6698 - val_loss: 1.3319 - val_accuracy: 0.5402
- 18s/epoch - 47ms/step
Epoch 21/50
391/391 - 18s - loss: 0.9191 - accuracy: 0.6802 - val_loss: 1.3246 - val_accuracy: 0.5446
- 18s/epoch - 47ms/step
Epoch 22/50
```

```
391/391 - 18s - loss: 0.8985 - accuracy: 0.6865 - val_loss: 1.3106 - val_accuracy: 0.5518
- 18s/epoch - 47ms/step
Epoch 23/50
391/391 - 18s - loss: 0.8712 - accuracy: 0.6966 - val_loss: 1.3261 - val_accuracy: 0.5427
- 18s/epoch - 47ms/step
Epoch 24/50
391/391 - 19s - loss: 0.8426 - accuracy: 0.7081 - val_loss: 1.3353 - val_accuracy: 0.5439
- 19s/epoch - 48ms/step
Epoch 25/50
391/391 - 18s - loss: 0.8159 - accuracy: 0.7143 - val_loss: 1.3415 - val_accuracy: 0.5482
- 18s/epoch - 46ms/step
Epoch 26/50
391/391 - 18s - loss: 0.7898 - accuracy: 0.7297 - val_loss: 1.3573 - val_accuracy: 0.5466
- 18s/epoch - 47ms/step
Epoch 27/50
391/391 - 18s - loss: 0.7645 - accuracy: 0.7361 - val_loss: 1.3743 - val_accuracy: 0.5463
- 18s/epoch - 46ms/step
Epoch 28/50
391/391 - 19s - loss: 0.7359 - accuracy: 0.7467 - val_loss: 1.4037 - val_accuracy: 0.5411
- 19s/epoch - 48ms/step
Epoch 29/50
391/391 - 19s - loss: 0.7178 - accuracy: 0.7531 - val_loss: 1.3980 - val_accuracy: 0.5428
- 19s/epoch - 47ms/step
Epoch 30/50
391/391 - 19s - loss: 0.6834 - accuracy: 0.7639 - val_loss: 1.3986 - val_accuracy: 0.5508
- 19s/epoch - 49ms/step
Epoch 31/50
391/391 - 19s - loss: 0.6627 - accuracy: 0.7725 - val_loss: 1.4725 - val_accuracy: 0.5330
- 19s/epoch - 49ms/step
Epoch 32/50
391/391 - 18s - loss: 0.6403 - accuracy: 0.7820 - val_loss: 1.4478 - val_accuracy: 0.5446
- 18s/epoch - 47ms/step
Epoch 33/50
391/391 - 19s - loss: 0.6235 - accuracy: 0.7883 - val_loss: 1.4588 - val_accuracy: 0.5493
- 19s/epoch - 47ms/step
Epoch 34/50
391/391 - 18s - loss: 0.5976 - accuracy: 0.7946 - val_loss: 1.4609 - val_accuracy: 0.5460
- 18s/epoch - 47ms/step
Epoch 35/50
391/391 - 18s - loss: 0.5754 - accuracy: 0.8034 - val_loss: 1.5231 - val_accuracy: 0.5516
- 18s/epoch - 47ms/step
Epoch 36/50
391/391 - 18s - loss: 0.5537 - accuracy: 0.8108 - val_loss: 1.5244 - val_accuracy: 0.5454
- 18s/epoch - 47ms/step
Epoch 37/50
391/391 - 19s - loss: 0.5285 - accuracy: 0.8206 - val_loss: 1.5243 - val_accuracy: 0.5495
- 19s/epoch - 47ms/step
Epoch 38/50
391/391 - 19s - loss: 0.5113 - accuracy: 0.8275 - val_loss: 1.5329 - val_accuracy: 0.5490
- 19s/epoch - 47ms/step
Epoch 39/50
391/391 - 18s - loss: 0.4873 - accuracy: 0.8348 - val_loss: 1.5832 - val_accuracy: 0.5473
- 18s/epoch - 47ms/step
Epoch 40/50
391/391 - 18s - loss: 0.4683 - accuracy: 0.8426 - val_loss: 1.5984 - val_accuracy: 0.5479
- 18s/epoch - 47ms/step
Epoch 41/50
391/391 - 18s - loss: 0.4424 - accuracy: 0.8522 - val_loss: 1.6031 - val_accuracy: 0.5524
- 18s/epoch - 47ms/step
Epoch 42/50
391/391 - 18s - loss: 0.4360 - accuracy: 0.8533 - val_loss: 1.6473 - val_accuracy: 0.5492
- 18s/epoch - 47ms/step
Epoch 43/50
391/391 - 19s - loss: 0.4195 - accuracy: 0.8593 - val_loss: 1.6494 - val_accuracy: 0.5596
- 19s/epoch - 48ms/step
```

```
Epoch 44/50
391/391 - 19s - loss: 0.3927 - accuracy: 0.8685 - val_loss: 1.7282 - val_accuracy: 0.5418
- 19s/epoch - 48ms/step
Epoch 45/50
391/391 - 19s - loss: 0.3831 - accuracy: 0.8742 - val_loss: 1.6558 - val_accuracy: 0.5608
- 19s/epoch - 48ms/step
Epoch 46/50
391/391 - 19s - loss: 0.3639 - accuracy: 0.8788 - val_loss: 1.7565 - val_accuracy: 0.5478
- 19s/epoch - 48ms/step
Epoch 47/50
391/391 - 19s - loss: 0.3459 - accuracy: 0.8867 - val_loss: 1.7802 - val_accuracy: 0.5500
- 19s/epoch - 48ms/step
Epoch 48/50
391/391 - 20s - loss: 0.3404 - accuracy: 0.8888 - val_loss: 1.7691 - val_accuracy: 0.5545
- 20s/epoch - 51ms/step
Epoch 49/50
391/391 - 19s - loss: 0.3210 - accuracy: 0.8950 - val_loss: 1.8083 - val_accuracy: 0.5487
- 19s/epoch - 48ms/step
Epoch 50/50
391/391 - 20s - loss: 0.3071 - accuracy: 0.8970 - val_loss: 1.8534 - val_accuracy: 0.5482
- 20s/epoch - 51ms/step
```

In [63]:

```
print('정확률=', dmlp.evaluate(x_test,y_test,verbose=0)[1]*100)
```

```
정확률= 54.82000112533569
```

In [64]:

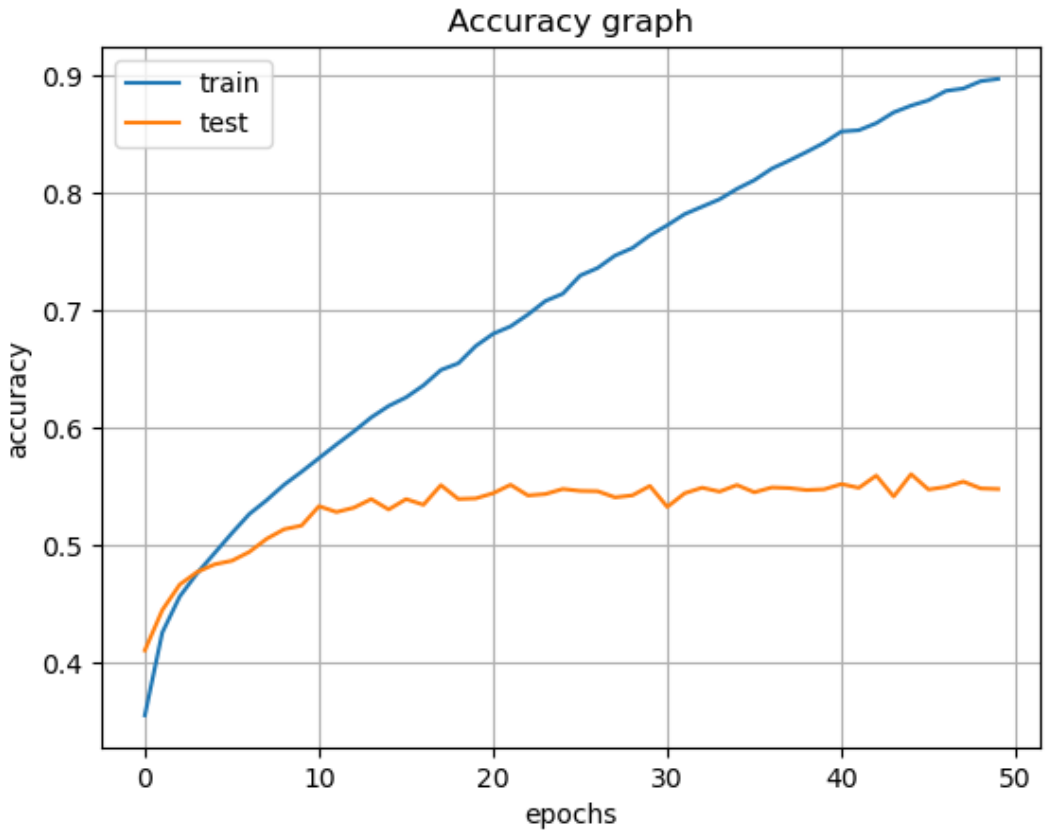
```
# dmlp.save('dmlp_trained.h5')
```

In [65]:

```
import matplotlib.pyplot as plt
```

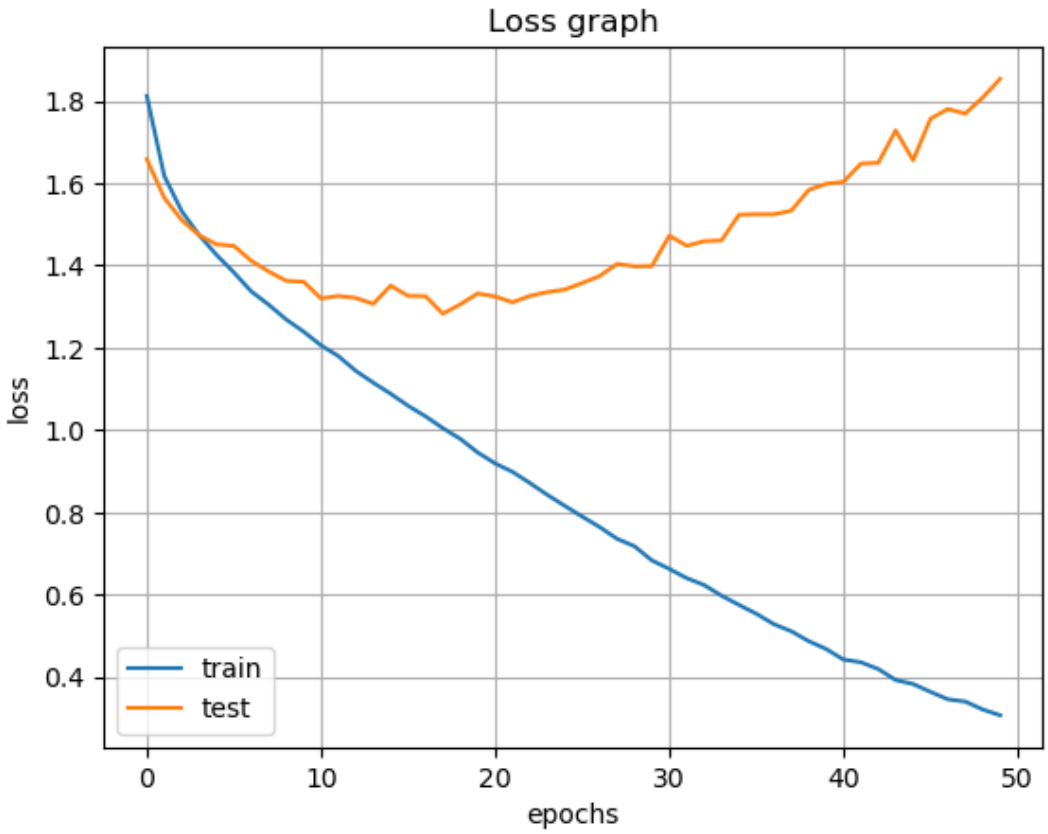
In [66]:

```
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.title('Accuracy graph')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend(['train','test'])
plt.grid()
plt.show()
```



In [67]:

```
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Loss graph')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend(['train','test'])
plt.grid()
plt.show()
```



In [ ]:

```
```