

# 2100998이수찬 인공지능

## 7-1

The screenshot shows the Spyder Python IDE interface. On the left, there are two code editors: '7-1.py' and '7-2.py'. The '7-1.py' editor contains Python code for a CNN model using TensorFlow and Keras. The '7-2.py' editor is currently active. On the right side of the interface, there is a plot titled 'Model loss' showing training and validation loss over 100 epochs. The training loss (blue line) decreases rapidly from approximately 1.5 to 0.2. The validation loss (orange line) decreases more slowly, stabilizing around 0.7. Below the plot, the Python console shows the command 'hist=cnn.fit(x\_train,y\_train,batch\_size=128,epochs=100,validation\_data=(x\_test,y\_test),verbose=2)' and its execution results, including accuracy and loss values. The file browser at the bottom shows the project structure with files like 'test\_images', '7-1.py', '7-2.py', and 'my\_cnn\_for\_deploy.h5'.

```
4  from tensorflow.keras.models import Sequential
5  from tensorflow.keras.layers import Conv2D,MaxPooling2D, Flatten, Dense, Dropout
6  from tensorflow.keras.optimizers import Adam
7
8
9  (x_train,y_train),(x_test,y_test)=cifar10.load_data()
10 x_train=x_train.astype(np.float32)/255.0
11 x_test=x_test.astype(np.float32)/255.0
12 y_train=tf.keras.utils.to_categorical(y_train,10)
13 y_test=tf.keras.utils.to_categorical(y_test,10)
14
15
16 cnn=Sequential()
17 cnn.add(Conv2D(32,(3,3),activation='relu',input_shape=(32,32,3)))
18 cnn.add(Conv2D(32,(3,3),activations='relu'))
19 cnn.add(MaxPooling2D(pool_size=(2,2)))
20 cnn.add(Dropout(0.25))
21 cnn.add(Conv2D(64,(3,3),activation='relu'))
22 cnn.add(Conv2D(64,(3,3),activations='relu'))
23 cnn.add(MaxPooling2D(pool_size=(2,2)))
24 cnn.add(Dropout(0.25))
25 cnn.add(Flatten())
26 cnn.add(Dense(512,activations='relu'))
27 cnn.add(Dropout(0.5))
28 cnn.add(Dense(10,activation='softmax'))
29
30
31 cnn.compile(loss='categorical_crossentropy',optimizer=Adam(),metrics=['accuracy'])
32 hist=cnn.fit(x_train,y_train,batch_size=128,epochs=100,validation_data=(x_test,y_test),verbose=2)
33
34
35 res=cnn.evaluate(x_test,y_test,verbose=0)
36 print("정확률은",res[1]*100)
```

📁 test_images	2025-05-22 오전 10:12	파일 폴더
🐍 7-1.py	2025-05-27 오전 12:03	Python File
🐍 7-2.py	2025-05-22 오전 10:08	Python File
📄 my_cnn_for_deploy.h5	2025-05-27 오전 12:58	H5 파일

## 7-2.

The screenshot shows a Jupyter Notebook interface with several components:

- Code Editor:** A left panel containing Python code for a CNN model. The code imports TensorFlow, PIL, and os, loads a pre-trained model, processes test images, and generates bar charts for predictions. The code is numbered from 9 to 40.
- Image Preview:** A grid of small images representing the test dataset, showing various animals and objects.
- Console:** A bottom panel showing the output of running the script. It includes the command "%runfile C:/soochan/7-2.py --wdir", the model's accuracy (1/1), and two bar chart plots labeled In [6] and In [7].

```
9 import tensorflow as tf
10 from PIL import Image
11 import os
12
13 cnn=tf.keras.models.load_model("my_cnn_for_deploy.h5")
14 class_names=['airplane','automobile','bird','cat','deer','dog','frog','horse'
15
16 x_test=[]
17 for filename in os.listdir("./test_images"):
18     if 'jpg' not in filename:
19         continue
20     img=Image.open("./test_images/"+filename)
21     x=np.asarray(img.resize([32,32]))/255.0
22     x_test.append(x)
23 x_test=np.asarray(x_test)
24
25 pred=cnn.predict(x_test)
26
27 import matplotlib.pyplot as plt
28
29 n=len(x_test)
30 plt.figure(figsize=(18,4))
31
32 for i in range(n):
33     plt.subplot(2,n,i+1)
34     plt.imshow(x_test[i])
35     plt.xticks([]);plt.yticks([])
36     plt.subplot(2,n,n+i+1)
37     if i==0:
38         plt.barh(class_names,pred[i])
39     else:
40         plt.barh(['a','A','b','c','d','f','h','s','t'],pred[i])
```

In [6]: %runfile C:/soochan/7-2.py --wdir  
1/1 [=====] - 0s 198ms/step  
In [7]:

### 7-3

C:\Wsoochan\W7-3.py

```

 5  cnn=tf.keras.models.load_model("my_cnn_for_deploy.h5")
 6  class_names=['airplane','automobile','bird','cat','deer','dog','frog','horse'
 7
 8  x_test=[]
 9  img_orig=[]
10  fname=[]
11  for filename in os.listdir('./test_images'):
12    if 'jpg' not in filename:
13      continue
14    img=Image.open('./test_images/'+filename)
15    img_orig.append(img)
16    fname.append(filename)
17    x=np.asarray(img.resize([32,32]))/255.0
18    x_test.append(x)
19  x_test=np.asarray(x_test)
20
21  pred=cnn.predict(x_test)
22
23  os.chdir('./test_images')
24  if not os.path.isdir('class_buckets'):
25    os.mkdir('class_buckets')
26
27  os.chdir('class_buckets')
28  for i in range(len(class_names)):
29    if not os.path.isdir(class_names[i]):
30      os.mkdir(class_names[i])
31
32  for i in range(len(x_test)):
33    folder_name=class_names[np.argmax(pred[i])]
34    os.chdir(folder_name)
35    img_orig[i].save(fname[i])

```

class\_buckets

In [8]: %runfile C:/soochan/untitled2.py --wdir

WARNING:tensorflow:5 out of the last 5 calls to <function Model.make\_predict\_function.<locals>.predict\_function at 0x000001F3CC540040>, triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has reduce\_retracing=True option that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling\_retracing and https://www.tensorflow.org/api\_docs/python/tf.function for more details.

1/1 [=====] - 0s 66ms/step

파일 탐색기

이름	수정한 날짜	유형	크기
airplane	2025-05-27 오전 1:14	파일 폴더	
automobile	2025-05-27 오전 1:14	파일 폴더	
bird	2025-05-27 오전 1:14	파일 폴더	
cat	2025-05-27 오전 1:14	파일 폴더	
deer	2025-05-27 오전 1:14	파일 폴더	
dog	2025-05-27 오전 1:14	파일 폴더	
frog	2025-05-27 오전 1:14	파일 폴더	
horse	2025-05-27 오전 1:14	파일 폴더	
ship	2025-05-27 오전 1:14	파일 폴더	
truck	2025-05-27 오전 1:14	파일 폴더	