초보자 딥러닝 입문하기

- 딥러닝을 활용한 cifar10 데이터 셋 분석

학습 목표

- 실습을 통해 CNN에 대해 이해해 본다.

학습 내용

- cifar-10 데이터 셋을 활용하여 CNN 알고리즘의 이해를 위한 실습.

목차

01. 데이터 준비 및 라이브러리 임포트

목차로 이동하기

CIFAR-10

- 32x32픽셀의 60000개의 컬러이미지가 포함되어 있다.
- 각 이미지는 10개의 클래스로 라벨링이 되어 있음.
- 비행기, 자동차, 새, 고양이, 사슴, 개, 개구리, 말, 배, 트럭

In [1]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import keras

import tensorflow as tf
import sklearn as sk
from tensorflow.keras.layers import *
from tensorflow.keras.models import *
from tensorflow.keras.utils import *

from sklearn.preprocessing import *
import seaborn as sns
```

In [2]:

```
print(tf.__version__)
print(np.__version__)
print(pd.__version__)
print(sns.__version__)
print(sk.__version__)
```

```
2.11.0
1.21.5
1.4.4
0.11.2
1.0.2
```

데이터 불러오기

In [3]:

```
from keras.datasets import cifar10

(X_train, Y_train), (X_test, Y_test) = cifar10.load_data()

X_train.shape, Y_train.shape, X_test.shape, Y_test.shape
```

Out[3]:

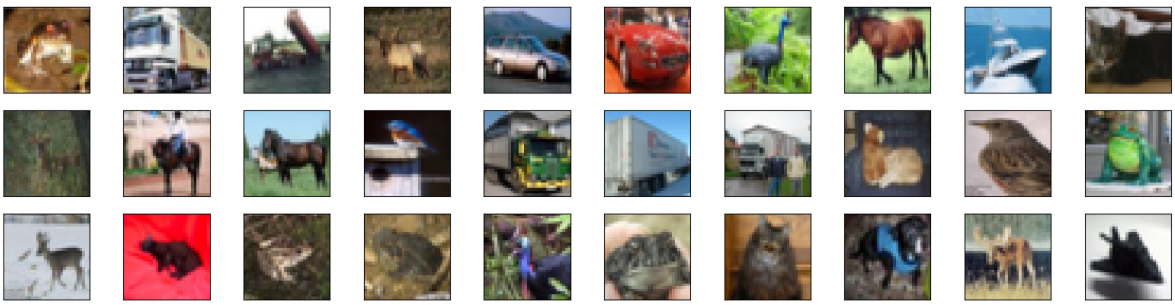
```
((50000, 32, 32, 3), (50000, 1), (10000, 32, 32, 3), (10000, 1))
```

이미지 데이터 표시하기

In [4]:

```
fig = plt.figure(figsize=(20,5))

for i in range(30):
    ax = fig.add_subplot(3, 10, i+1, xticks=[], yticks=[])
    ax.imshow(X_train[i])
```



데이터 정규화

In [5]:

```
print(X_train[0][0:1])

X_train_n = X_train/255.0
X_test_n = X_test /255.0

print(X_train_n[0][0:1])
```

```
[[[ 59  62  63]
  [ 43  46  45]
  [ 50  48  43]
  [ 68  54  42]
  [ 98  73  52]
  [119  91  63]
  [139 107  75]
  [145 110  80]
  [149 117  89]
  [149 120  93]
  [131 103  77]
  [125  99  76]
  [142 115  91]
  [144 112  86]
  [137 105  79]
  [129  97  71]
  [137 106  79]
  [134 106  76]
  [124  97  64]
  [139 113  78]
  [139 112  75]
  [133 105  69]
  [136 105  74]
  [139 108  77]
  [152 120  89]
  [163 131 100]
  [168 136 108]
  [159 129 102]
  [158 130 104]
  [158 132 108]
  [152 125 102]
  [148 124 103]]]
[[[0.23137255 0.24313725 0.24705882]
  [0.16862745 0.18039216 0.17647059]
  [0.19607843 0.18823529 0.16862745]
  [0.26666667 0.21176471 0.16470588]
  [0.38431373 0.28627451 0.20392157]
  [0.46666667 0.35686275 0.24705882]
  [0.54509804 0.41960784 0.29411765]
  [0.56862745 0.43137255 0.31372549]
  [0.58431373 0.45882353 0.34901961]
  [0.58431373 0.47058824 0.36470588]
  [0.51372549 0.40392157 0.30196078]
  [0.49019608 0.38823529 0.29803922]
  [0.55686275 0.45098039 0.35686275]
  [0.56470588 0.43921569 0.3372549 ]
  [0.5372549  0.41176471 0.30980392]
  [0.50588235 0.38039216 0.27843137]
  [0.5372549  0.41568627 0.30980392]
  [0.5254902  0.41568627 0.29803922]
```

```
 [0.48627451 0.38039216 0.25098039]
 [0.54509804 0.44313725 0.30588235]
 [0.54509804 0.43921569 0.29411765]
 [0.52156863 0.41176471 0.27058824]
 [0.53333333 0.41176471 0.29019608]
 [0.54509804 0.42352941 0.30196078]
 [0.59607843 0.47058824 0.34901961]
 [0.63921569 0.51372549 0.39215686]
 [0.65882353 0.53333333 0.42352941]
 [0.62352941 0.50588235 0.4        ]
 [0.61960784 0.50980392 0.40784314]
 [0.61960784 0.51764706 0.42352941]
 [0.59607843 0.49019608 0.4        ]
 [0.58039216 0.48627451 0.40392157]]]
```

출력 데이터 전처리

- 원핫

In [6]:

```
print(Y_train[0:3])

Y_train_n = to_categorical(Y_train)
Y_test_n = to_categorical(Y_test)

print(Y_train_n[0:3])
```

```
[[6]
 [9]
 [9]]
[[0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

02. 모델 구축하기

목차로 이동하기

In [9]:

```
model = Sequential()
model.add(Conv2D(filters=16, kernel_size=3,
                 padding='same', strides=1, activation='relu', input_shape=(32,32,3)))
model.add(MaxPool2D(pool_size=2))
model.add(Conv2D(filters=32, kernel_size=3,
                 padding='same', strides=1, activation='relu'))
model.add(MaxPool2D(pool_size=2))
model.add(Conv2D(filters=64, kernel_size=3,
                 padding='same', strides=1, activation='relu'))
model.add(MaxPool2D(pool_size=2))

# FCL(fully connected layer)
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 32, 32, 16)        448

 max_pooling2d_3 (MaxPooling  (None, 16, 16, 16)       0
 2D)

 conv2d_4 (Conv2D)           (None, 16, 16, 32)        4640

 max_pooling2d_4 (MaxPooling  (None, 8, 8, 32)         0
 2D)

 conv2d_5 (Conv2D)           (None, 8, 8, 64)          18496

 max_pooling2d_5 (MaxPooling  (None, 4, 4, 64)         0
 2D)

 flatten_1 (Flatten)         (None, 1024)              0

 dense_2 (Dense)             (None, 512)               524800

 dense_3 (Dense)             (None, 10)                5130

=================================================================
Total params: 553,514
Trainable params: 553,514
Non-trainable params: 0
_________________________________________________________________
```

모델 상세 설정

In [11]:

```
model.compile(loss='categorical_crossentropy',
              optimizer='adam', metrics=['accuracy'])
```

03. 모델 학습하기

목차로 이동하기

In [12]:

```
model.fit(X_train_n, Y_train_n,
          batch_size=128, epochs=5, validation_split=0.1)
```

```
Epoch 1/5
352/352 [==============================] - 131s 320ms/step - loss: 1.5761 - accurac
y: 0.4236 - val_loss: 1.3098 - val_accuracy: 0.5252
Epoch 2/5
352/352 [==============================] - 88s 248ms/step - loss: 1.2142 - accuracy:
0.5637 - val_loss: 1.1304 - val_accuracy: 0.5892
Epoch 3/5
352/352 [==============================] - 68s 193ms/step - loss: 1.0565 - accuracy:
0.6249 - val_loss: 0.9866 - val_accuracy: 0.6552
Epoch 4/5
352/352 [==============================] - 69s 195ms/step - loss: 0.9303 - accuracy:
0.6702 - val_loss: 0.9041 - val_accuracy: 0.6826
Epoch 5/5
352/352 [==============================] - 52s 146ms/step - loss: 0.8391 - accuracy:
0.7052 - val_loss: 0.8463 - val_accuracy: 0.7064
```

Out[12]:

```
<keras.callbacks.History at 0x164667a9280>
```

In [13]:

```
model.evaluate(X_test_n, Y_test_n)
```

```
313/313 [==============================] - 8s 25ms/step - loss: 0.8713 - accuracy:
0.6951
```

Out[13]:

```
[0.8713232278823853, 0.6951000094413757]
```

다양한 실습 확인

- adam : 5epochs, batch_size=128, val_split = 0.1
  - loss: 0.8713 - accuracy: 0.6951
- adam -> rmsprop 으로 변경 []
- adam, 마지막 pooling 없애기 [] - []
- sigmoid, 마지막 pooling 없애기 []- []
- leakyRelu 적용, 마지막 pooling 없애기 []- []

- Adamax, relu [L123전부 사용]-

[실습 확인] 활성화 함수 : LeakyRelu 적용

In [14]:

```
from tensorflow.keras.layers import LeakyReLU
```

In [18]:

```
model = Sequential()
model.add(Conv2D(filters=16, kernel_size=3,
                 padding='same', strides=1, input_shape=(32,32,3)))
model.add(LeakyReLU(alpha=0.2))
model.add(MaxPool2D(pool_size=2))

model.add(Conv2D(filters=32, kernel_size=3, padding='same', strides=1))
model.add(LeakyReLU(alpha=0.2))
model.add(MaxPool2D(pool_size=2))

model.add(Conv2D(filters=64, kernel_size=3, padding='same', strides=1))
model.add(LeakyReLU(alpha=0.2))
model.add(MaxPool2D(pool_size=2))

# FCL(fully connected layer)
model.add(Flatten())
model.add(Dense(512))
model.add(LeakyReLU(alpha=0.2))
model.add(Dense(10, activation='softmax'))

model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_9 (Conv2D)           (None, 32, 32, 16)        448

 leaky_re_lu_4 (LeakyReLU)   (None, 32, 32, 16)        0

 max_pooling2d_9 (MaxPooling  (None, 16, 16, 16)       0
 2D)

 conv2d_10 (Conv2D)          (None, 16, 16, 32)        4640

 leaky_re_lu_5 (LeakyReLU)   (None, 16, 16, 32)        0

 max_pooling2d_10 (MaxPoolin  (None, 8, 8, 32)         0
 g2D)

 conv2d_11 (Conv2D)          (None, 8, 8, 64)          18496

 leaky_re_lu_6 (LeakyReLU)   (None, 8, 8, 64)          0

 max_pooling2d_11 (MaxPoolin  (None, 4, 4, 64)         0
 g2D)

 flatten_3 (Flatten)         (None, 1024)              0

 dense_6 (Dense)             (None, 512)               524800

 leaky_re_lu_7 (LeakyReLU)   (None, 512)               0

 dense_7 (Dense)             (None, 10)                5130

=================================================================
Total params: 553,514
Trainable params: 553,514
```

```
Non-trainable params: 0
_________________________________________________________________
```

In [19]:

```
%%time

model.compile(loss='categorical_crossentropy',
              optimizer='adam', metrics=['accuracy'])

model.fit(X_train_n, Y_train_n,
          batch_size=128, epochs=5, validation_split=0.1)
```

```
Epoch 1/5
352/352 [==============================] - 57s 157ms/step - loss: 1.4769 - accuracy:
0.4704 - val_loss: 1.1868 - val_accuracy: 0.5820
Epoch 2/5
352/352 [==============================] - 56s 160ms/step - loss: 1.0984 - accuracy:
0.6134 - val_loss: 1.0091 - val_accuracy: 0.6464
Epoch 3/5
352/352 [==============================] - 58s 165ms/step - loss: 0.9546 - accuracy:
0.6650 - val_loss: 0.9837 - val_accuracy: 0.6532
Epoch 4/5
352/352 [==============================] - 77s 219ms/step - loss: 0.8504 - accuracy:
0.7012 - val_loss: 0.8774 - val_accuracy: 0.6884
Epoch 5/5
352/352 [==============================] - 61s 173ms/step - loss: 0.7689 - accuracy:
0.7284 - val_loss: 0.8407 - val_accuracy: 0.7092
Wall time: 5min 9s
```

Out[19]:

```
<keras.callbacks.History at 0x164bef5b0d0>
```

In [20]:

```
model.evaluate(X_test_n, Y_test_n)
```

```
313/313 [==============================] - 9s 29ms/step - loss: 0.8697 - accuracy:
0.6965
```

Out[20]:

```
[0.8697460889816284, 0.6965000033378601]
```

확인한 결과

- relu에서 leakyRelu 적용한 결과
  - 전 loss: 0.8391 - accuracy: 0.7052 - val_loss: 0.8463 - val_accuracy: 0.7064
  - 후 loss: 0.7689 - accuracy: 0.7284 - val_loss: 0.8407 - val_accuracy: 0.7092

04. 모델 학습 결과 저장 및 불러오기

목차로 이동하기

모델 확인 후, 저장

In [22]:

```
import os
```

In [23]:

```
path = os.path.join(os.getcwd(), "cifar_model")
savefile = os.path.join(path, "my_model_leaky.h5"  )
```

In [25]:

```
model.save(savefile)
```

In [28]:

```
!dir cifar_model
```

```
 D 드라이브의 볼륨: BackUp
 볼륨 일련 번호: 4EB1-0AD7

 D:₩GitHub₩DeepLearning_Basic_Class₩cifar_model 디렉터리

2022-11-21  오후 03:45    <DIR>          .
2022-11-21  오후 03:45    <DIR>          ..
2022-11-21  오후 03:45         6,698,688 my_model_leaky.h5
               1개 파일           6,698,688 바이트
               2개 디렉터리  250,605,187,072 바이트 남음
```

모델 불러오기, 평가

In [29]:

```
# 모델을 불러온다.
load_model = keras.models.load_model('cifar_model₩my_model_leaky.h5')
load_model
```

Out[29]:

```
<keras.engine.sequential.Sequential at 0x164e292bf70>
```

In [30]:

```
load_model.evaluate(X_test_n, Y_test_n)
```

```
313/313 [==============================] - 8s 25ms/step - loss: 0.8697 - accuracy:
0.6965
```

Out[30]:

```
[0.8697460889816284, 0.6965000033378601]
```