UST 과학기술연합대학원대학교
UNIVERSITY OF SCIENCE & TECHNOLOGY

www.ust.ac.kr

14장

CNN 소개

이 홍 석 (hsyi@kisti.re.kr)
한국과학기술정보연구원 슈퍼컴퓨팅응용센터

Contents

목차

UST 과학기술연합대학원대학교
UNIVERSITY OF SCIENCE & TECHNOLOGY

01. 2D-CNN
Cifar10 데이터

이홍석 (hsyi@kisti.re.kr)

UST 과학기술연합대학원대학교
UNIVERSITY OF SCIENCE & TECHNOLOGY

01. 합성곱 신경망

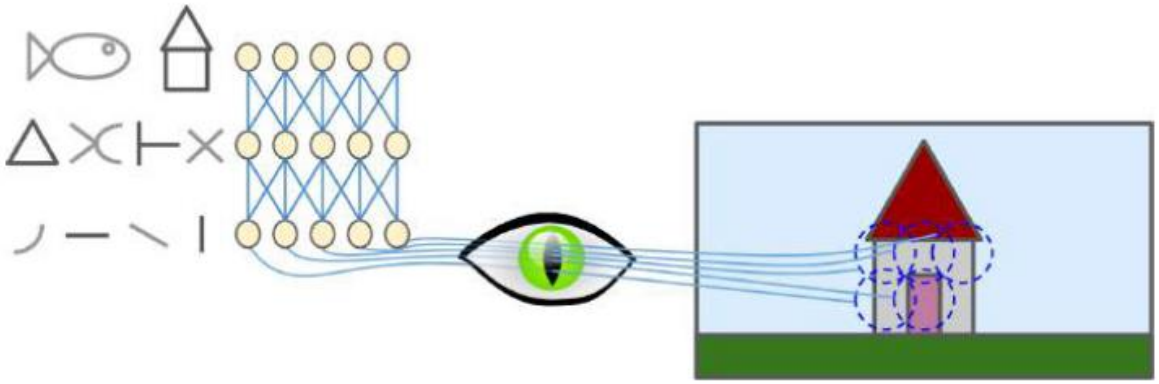
이홍석 (hsyi@kisti.re.kr)

합성곱 신경망

- 합성곱 신경망 (CNN, Convolutional Neural Network)
  - 대뇌의 시각피질(cortex) 연구로부터 시작되었고, 이미지 인식분야는 1980년말
  - CNN에서 일부 사람을 능가하는 성능을 달성
    - 컴퓨터 성능 발달,
    - 많은 양의 데이터,
    - 딥러닝 훈련 기술
- 이장의 학습 목표는
  - CNN이 어디에서 나왔는지,
  - 구성요소는 무엇인지,
  - 텐서플로우 2.0 이상에서 구현하는 방법
  - 가장 뛰어난 성능의 CNN 구조

시각 피질의 구조 (Visual Cortex)

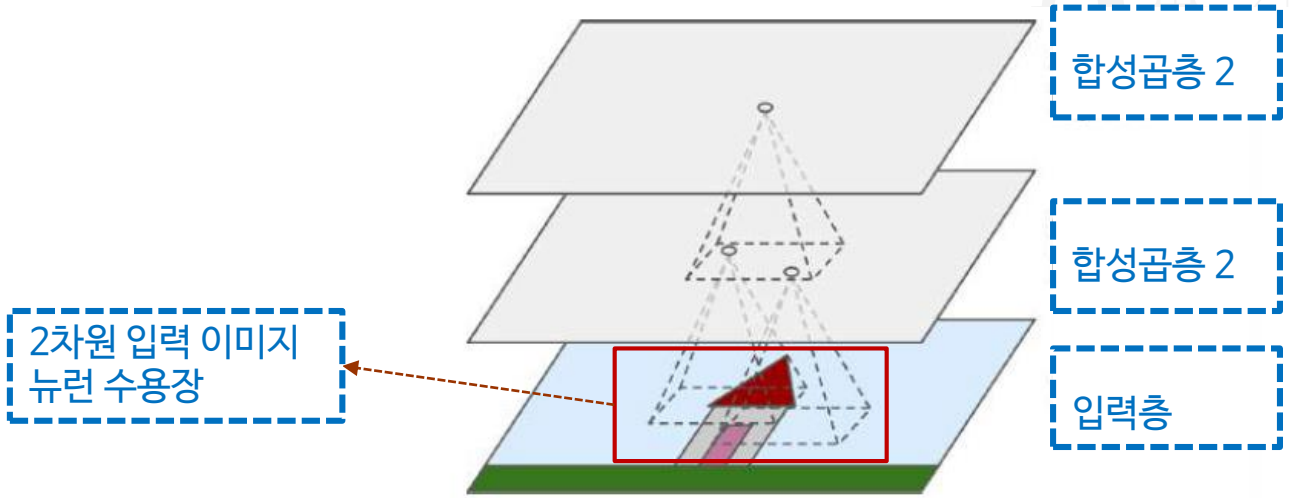
- 1958년 David Hubel, Torsten Wiesel (노벨 생리의학상)
  - 시각 피질 안의 많은 뉴런이 작은 국부 수용장을 가진다.



  - 1980년대 신인식기에 영감을 주고, 합성곱 신경망으로 진화
  - 1998년 LeNet-5 구조 소개
    - 얀 르쿤, 레옹 보토, 요수아 벤지오, 피트릭 하프너
    - 합성곱층과 플링층 요소를 도입

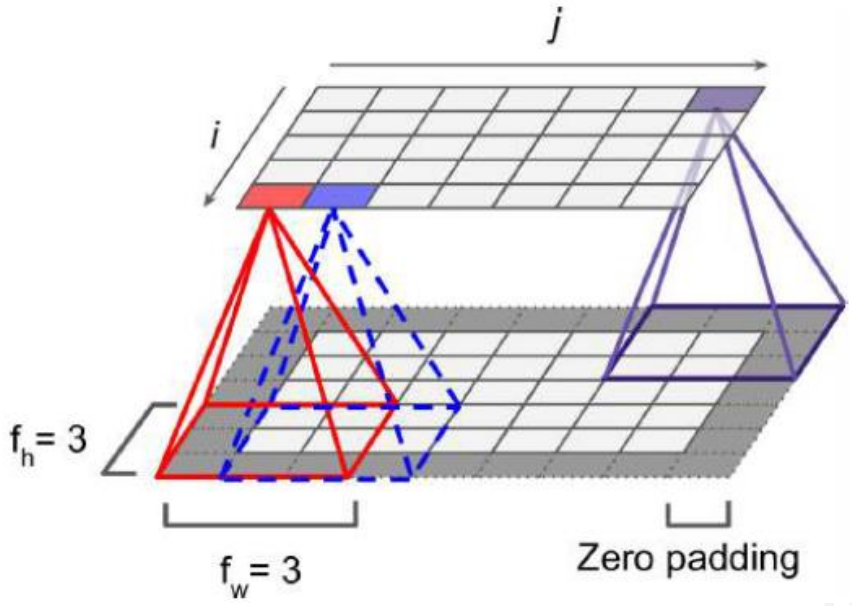
합성곱층 (1)

- 합성곱층은 CNN에서 가장 중요한 구성 요소이다.
  - 합성곱층 1에 있는 뉴런은 뉴런 수용장 안에 있는 픽셀에만 연결함
    - 입력 이미지의 모든 픽셀에 연결되는 것이 아님.
    - 저수준 특성에 집중
  - 합성곱층 2에 있는 각 뉴런은 합성곱층 1의 작은 사각 영역 안에 뉴런에 연결됨.
    - 고순준 특성에 집중

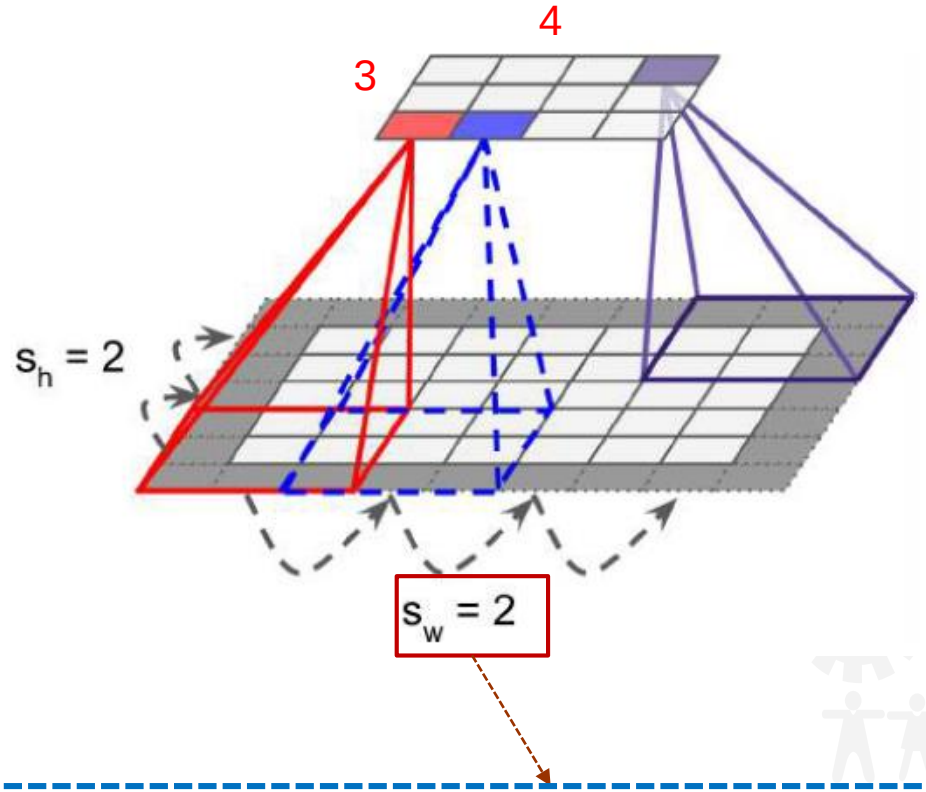


합성곱층 (2) : 제로패딩

- 수용장의 높이와 너비. 그리고 제로 패딩(zero padding)



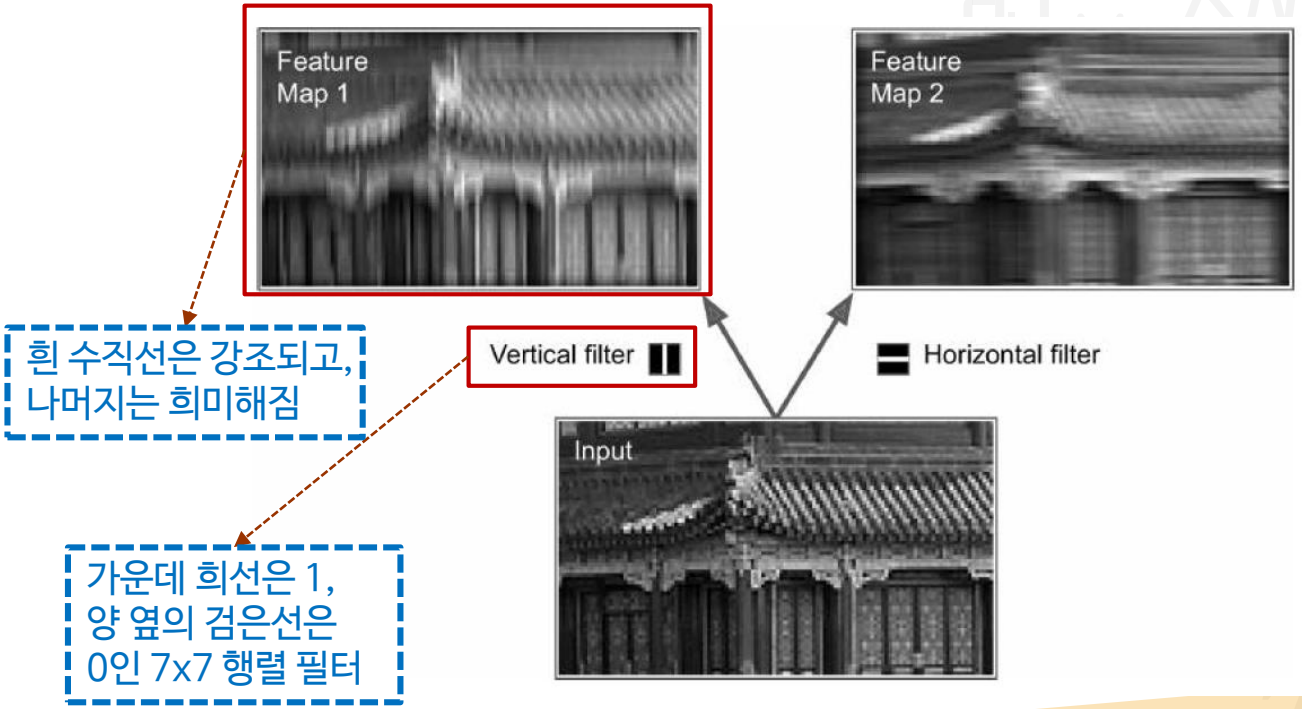
합성곱층 (3) : 스트라이드



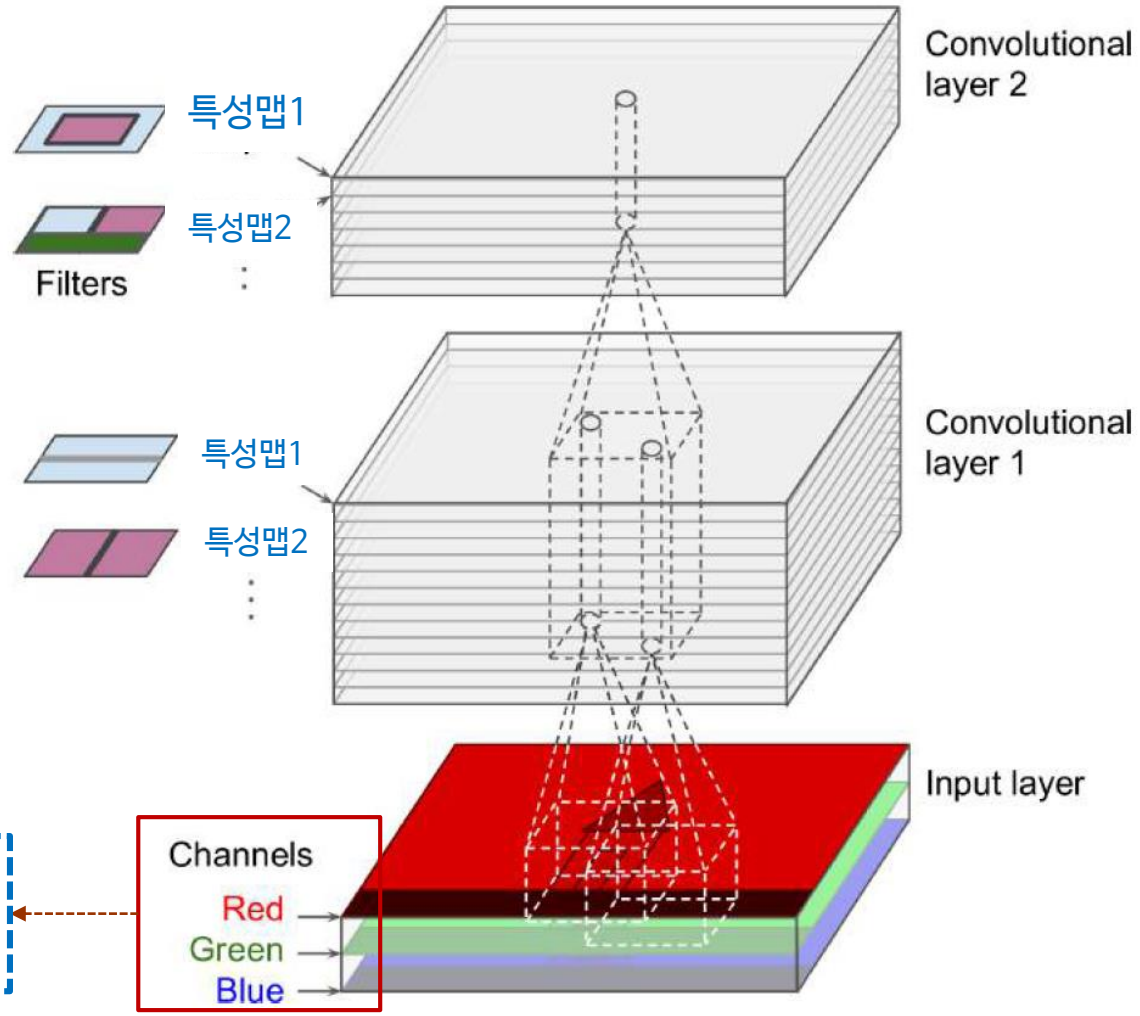
연속된 두 개의 수용장 사이의 거리를 스트라이드(stride)
- 여기서는 제로패딩이 적용되었고, 5x7 입력층이 3x3 수용장과 스트라이드 2를 사용함.
- 3x4 새로운 층이 생김

필터

- 뉴런의 가중치는 수용장 작은 크기로 표현된다.
  - 아래 그림에서 필터(합성곱 커널)는 2개의 가중치 세트
  - 같은 필터를 사용한 전체 뉴런의 층은 필터와 유사한 이미지의 영역을 강조하는 특성맵을 만듬.



여러 개의 특성맵 쌓기


Convolutional layer 2
특성맵1
특성맵2
Filters
Convolutional layer 1
특성맵1
특성맵2
Input layer
Channels
Red
Green
Blue
컬러 채널(R,G,B) 마다 하나씩 여러 개의 서브층으로 구성

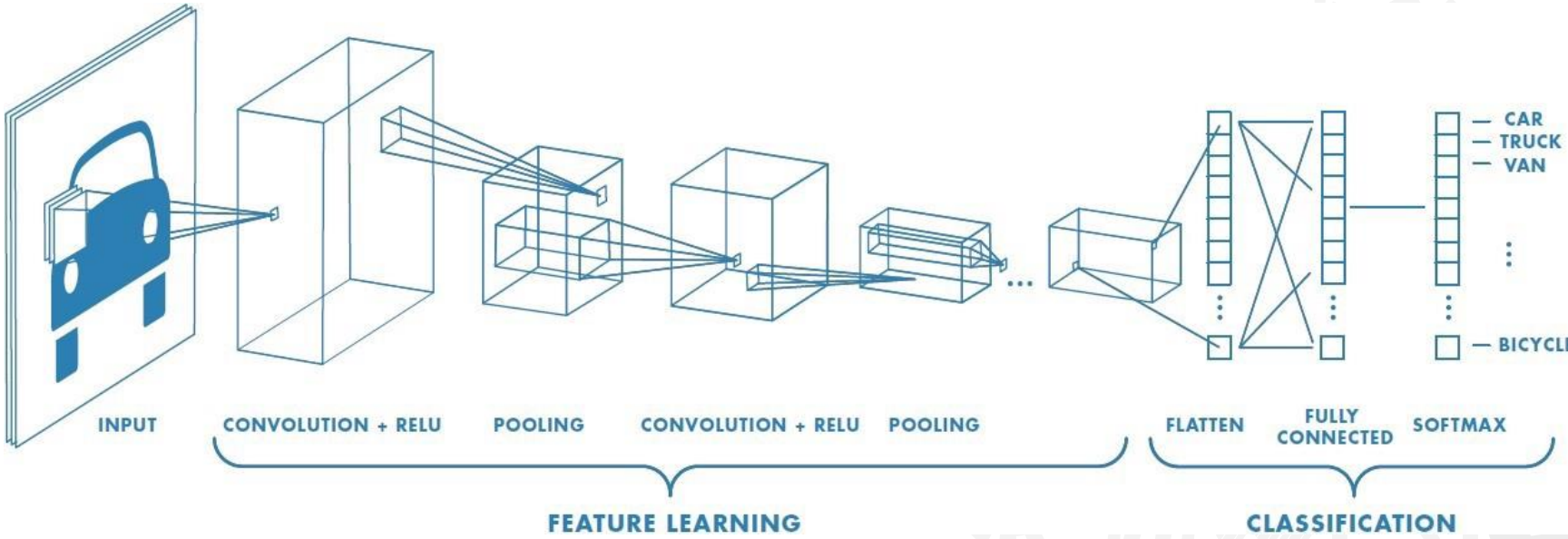
합성곱층에 있는 뉴런의 출력 계산

- 합성곱층에서 한 뉴런의 출력을 계산하는 방법
  - 입력에 대한 가중치 합을 계산하고
  - 편향을 더하는 것

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} \cdot w_{u,v,k',k} \quad \text{with} \begin{cases} i' = i \times s_h + u \\ j' = j \times s_w + v \end{cases}$$

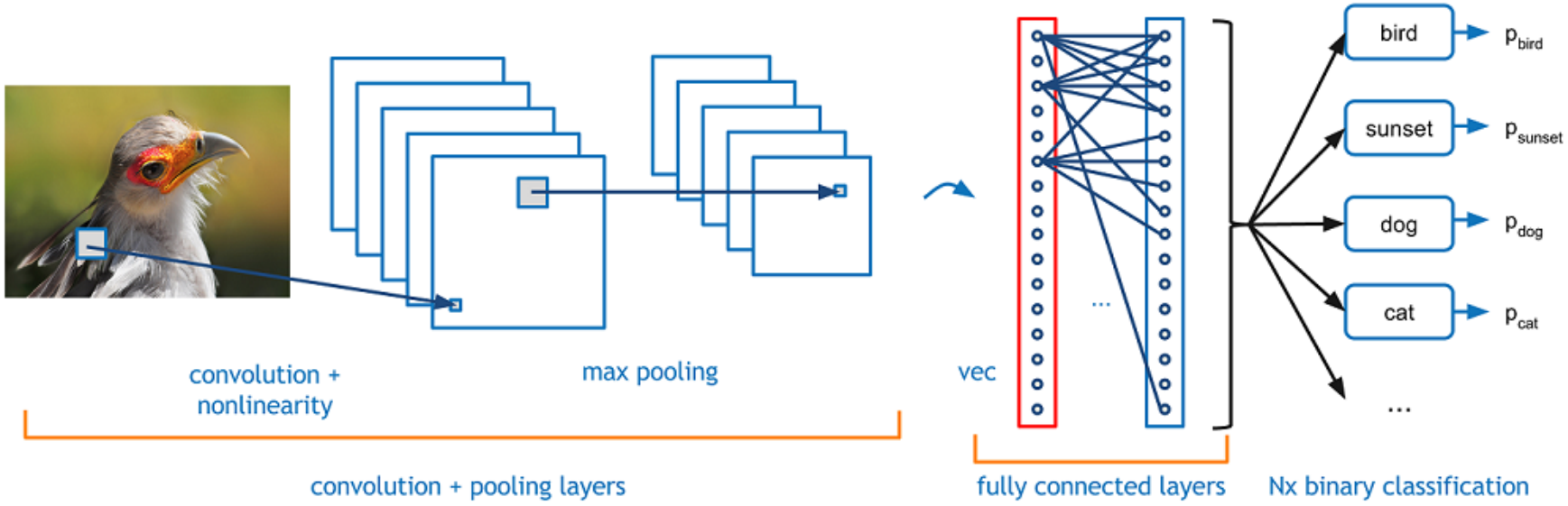
- 텐서플로우에서 구현하는 방법
  - 각 입력 이미지는 3차원 텐서이다. [높이, 너비, 채널]
  - 미니패치 사용하면 4차원 텐서로, [미니배치 크기, 높이, 너비, 채널]

CNN이란


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

CNN의 주요한 항목 소개 (1)

콘불루션 층 (Convolutional Layer)

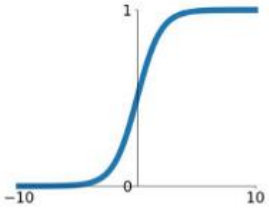


CNN의 주요한 항목 소개 (2)

활성함수

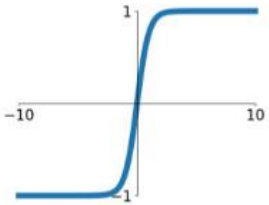
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



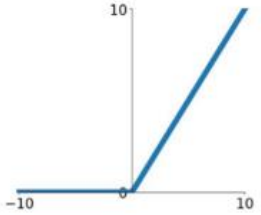
tanh

$$\tanh(x)$$



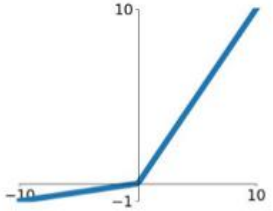
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

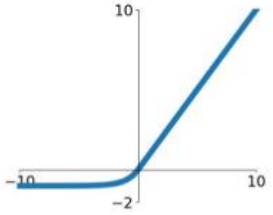


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

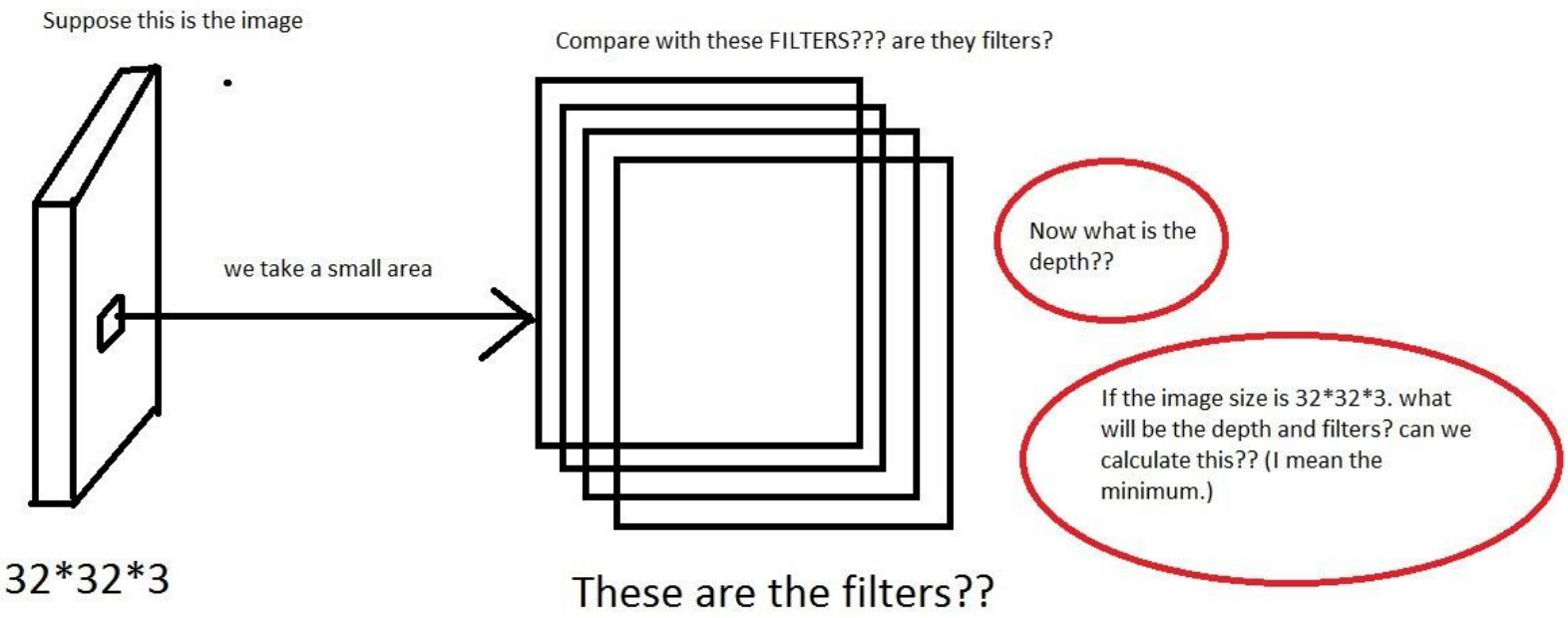
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



CNN의 주요한 항목 소개 (3)

필터, 커널 크기, 필터 개수



CNN의 주요한 항목 소개 (4)

스트라이트 크기와 패딩

0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

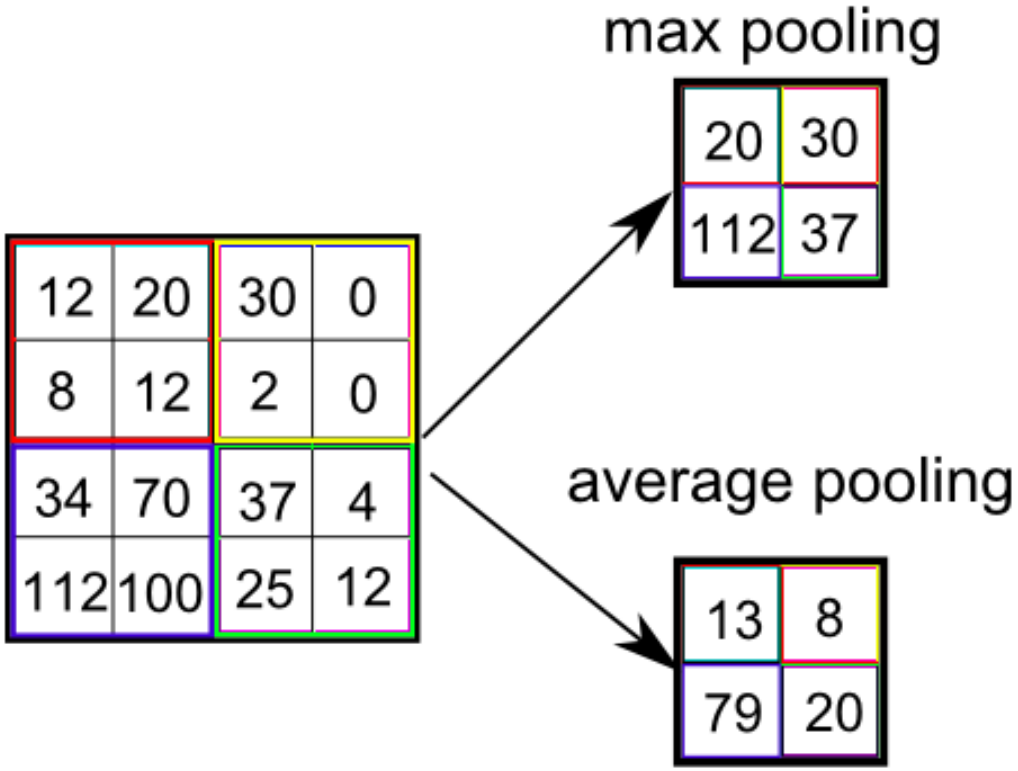
Kernel

0	-1	0
-1	5	-1
0	-1	0

114				

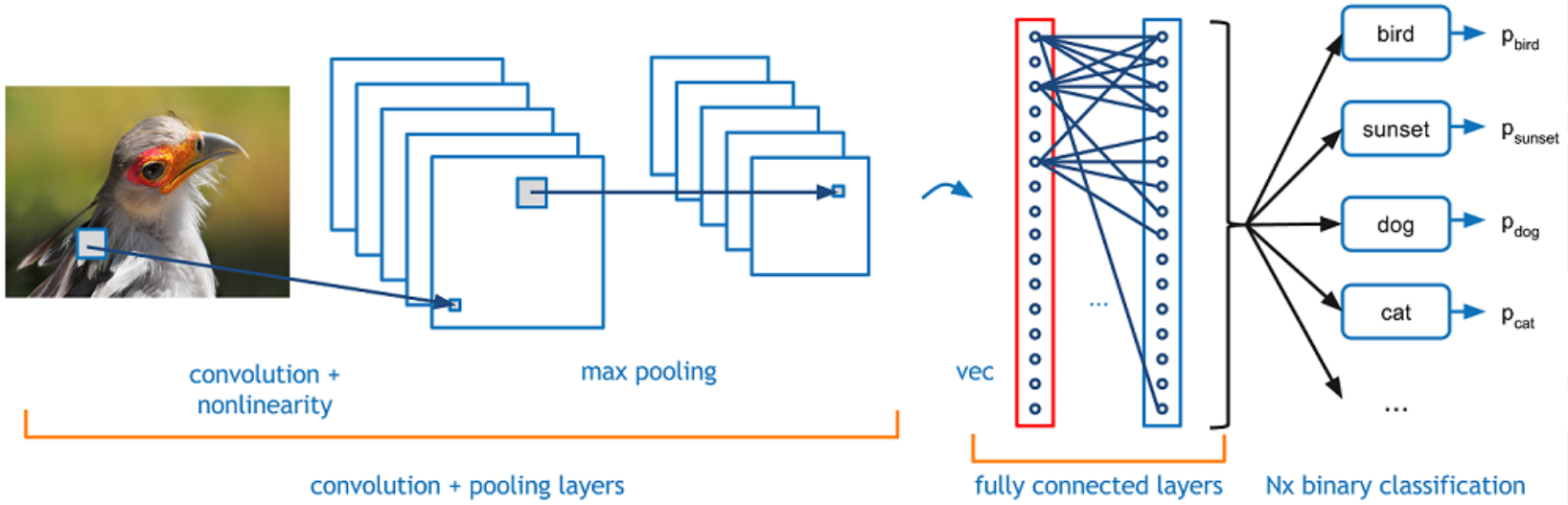
CNN의 주요한 항목 소개 (5)

맥스 풀링(max pooling)



CNN의 주요한 항목 소개 (6)

- 플랫튼 층 (Flatten layer)
  - 콘볼루션 층과 완전연결 (fully connected) 층 사이에 플랫튼 층이 있음
  - 평탄화는 피처의 2차원 행렬을 1차원 벡터로 변환하며, 이 벡터가 완전연결 신경망 분류를 위해 사용됨



CNN 적용한 Cifar10 이미지 (1)

```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Flatten, Dense, Dropout
from tensorflow.keras.layers import Conv2D, MaxPool2D

import numpy as np
import matplotlib.pyplot as plt
import matplotlib
```

CNN 적용한 Cifar10 이미지 (2)

```
from tensorflow.keras.datasets import cifar10

(X_train, y_train), (X_test, y_test) = cifar10.load_data()

classes_name = ['airplane', 'automobile', 'bird', 'cat',
                'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
```

```
X_train.max()
```

```
255
```

```
X_train = X_train/255
X_test = X_test/255
```

```
X_train.shape
```

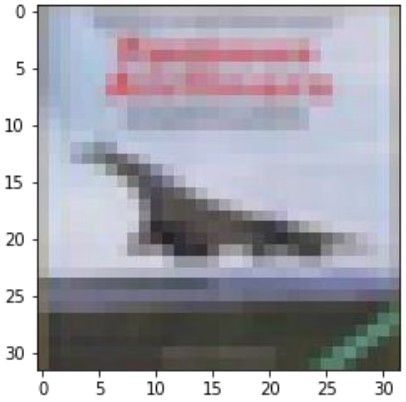
```
(50000, 32, 32, 3)
```

```
X_test.shape
```

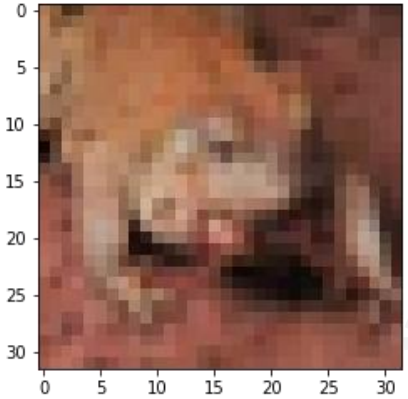
```
(10000, 32, 32, 3)
```

CNN 적용한 Cifar10 이미지 (3)

plt.imshow(X_test[3])



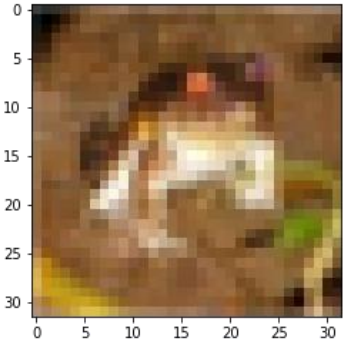
plt.imshow(X_test[5])



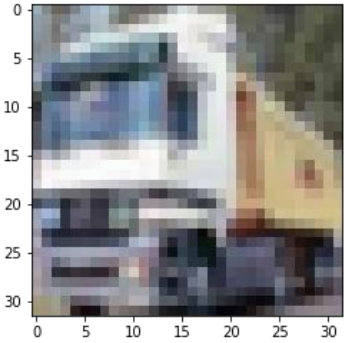
```
y_test
```

```
array([[3],
       [8],
       [8],
       ...,
       [5],
       [1],
       [7]], dtype=uint8)
```

plt.imshow(X_test[0])



plt.imshow(X_test[1])



CNN 적용한 Cifar10 이미지 (4)

Build CNN Model

```
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(3, 3),
    padding='same', activation='relu', input_shape = [32, 32, 3]))

model.add(Conv2D(filters=32, kernel_size=(3, 3),
    padding='same', activation='relu'))

model.add(MaxPool2D(pool_size=(2,2), strides=2, padding='valid'))

model.add(Dropout(0.5))
```

CNN 적용한 Cifar10 이미지 (5)

Build CNN Model

```
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(3, 3),
    padding='same', activation='relu', input_shape = [32, 32, 3]))

model.add(Conv2D(filters=32, kernel_size=(3, 3), padding='same', activation='relu'))
model.add(MaxPool2D(pool_size=(2,2), strides=2, padding='valid'))
model.add(Dropout(0.5))

model.add(Flatten())
model.add(Dense(units = 128, activation='relu'))
model.add(Dense(units=10, activation='softmax'))
```

CNN 적용한 Cifar10 이미지 (6)

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 16, 16, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense (Dense)                (None, 128)               1048704
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 1,060,138
Trainable params: 1,060,138
Non-trainable params: 0
_________________________________________________________________
```

CNN 적용한 Cifar10 이미지 (7)

```
model.compile(optimizer='adam', loss = 'sparse_categorical_crossentropy',
              metrics=['sparse_categorical_accuracy'])
```

```
history = model.fit(X_train, y_train, batch_size=10, epochs=10,
                    verbose=1, validation_data=(X_test, y_test))
```

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/10
50000/50000 [==============================] - 159s 3ms/sample - loss: 1.4253 - sparse_categ
orical_accuracy: 0.4872 - val_loss: 1.1328 - val_sparse_categorical_accuracy: 0.5979
Epoch 2/10
50000/50000 [==============================] - 192s 4ms/sample - loss: 1.0973 - sparse_categ
orical_accuracy: 0.6126 - val_loss: 1.0143 - val_sparse_categorical_accuracy: 0.6443
```

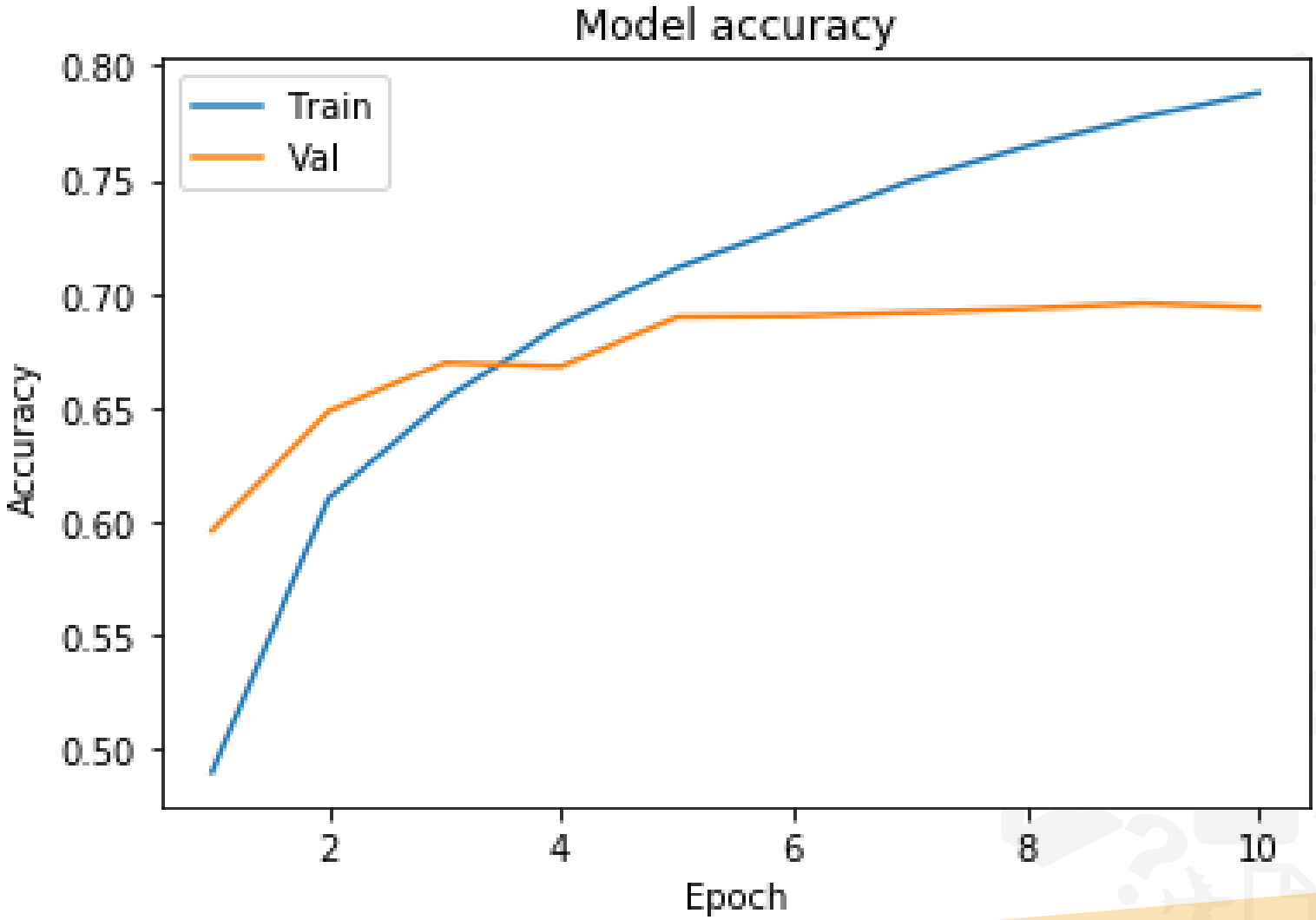
```
Epoch 9/10
50000/50000 [==============================] - 180s 4ms/sample - loss: 0.6528 - sparse_categ
orical_accuracy: 0.7693 - val_loss: 0.9086 - val_sparse_categorical_accuracy: 0.6965
Epoch 10/10
49140/50000 [============================>.] - ETA: 3s - loss: 0.6185 - sparse_categorical_a
ccuracy: 0.7828
```

CNN 적용한 Cifar10 이미지 (8)

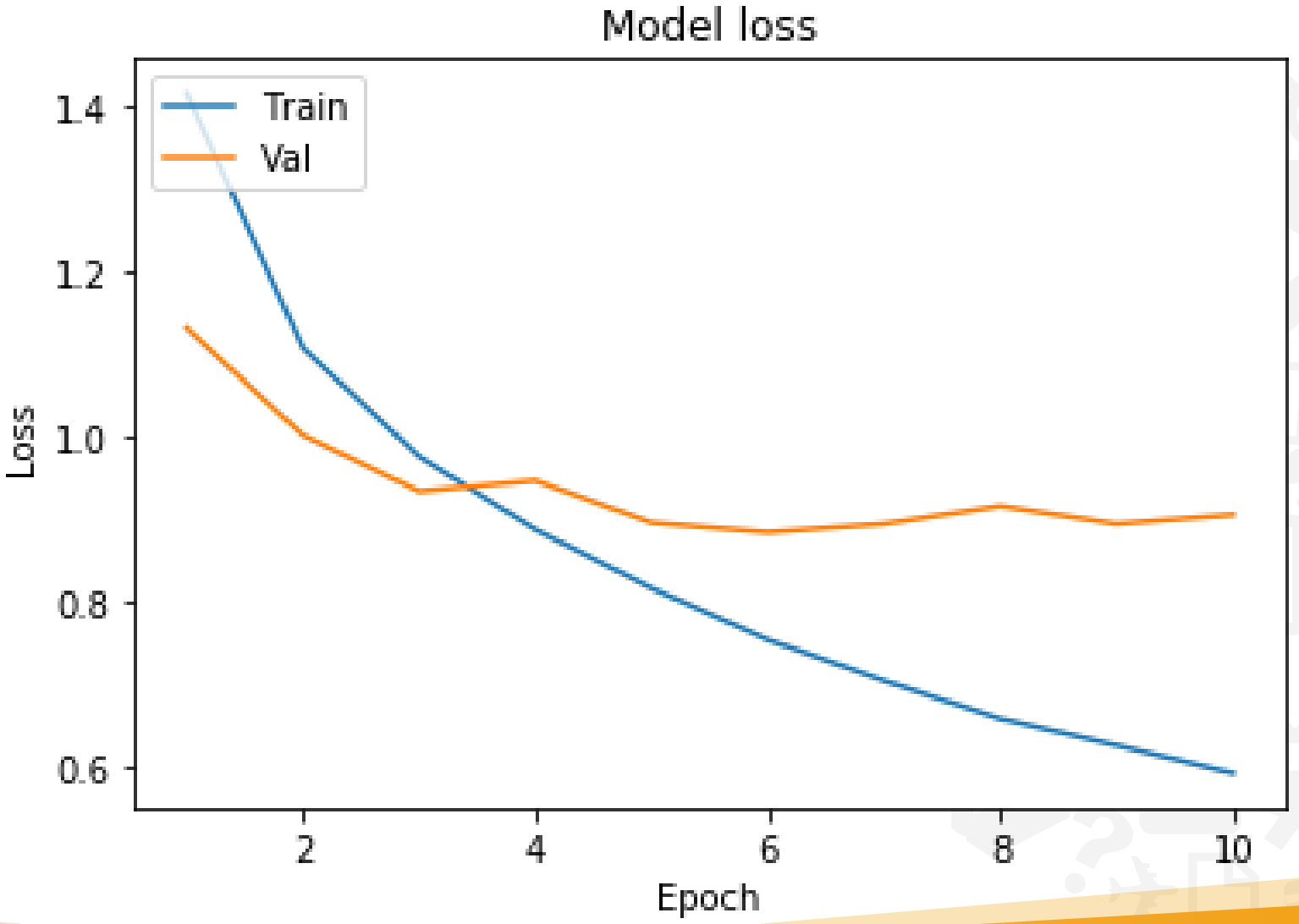
```
epoch_range = range(1, 11)
plt.plot(epoch_range, history.history['sparse_categorical_accuracy'])
plt.plot(epoch_range, history.history['val_sparse_categorical_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()

# Plot training & validation loss values
plt.plot(epoch_range, history.history['loss'])
plt.plot(epoch_range, history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()
```

CNN 적용한 Cifar10 이미지 (9)



CNN 적용한 Cifar10 이미지 (10)



CNN 적용한 Cifar10 이미지 (11)

```
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix
```

```
y_pred = model.predict_classes(X_test); y_pred
```

```
array([3, 8, 8, ..., 5, 1, 7], dtype=int64)
```

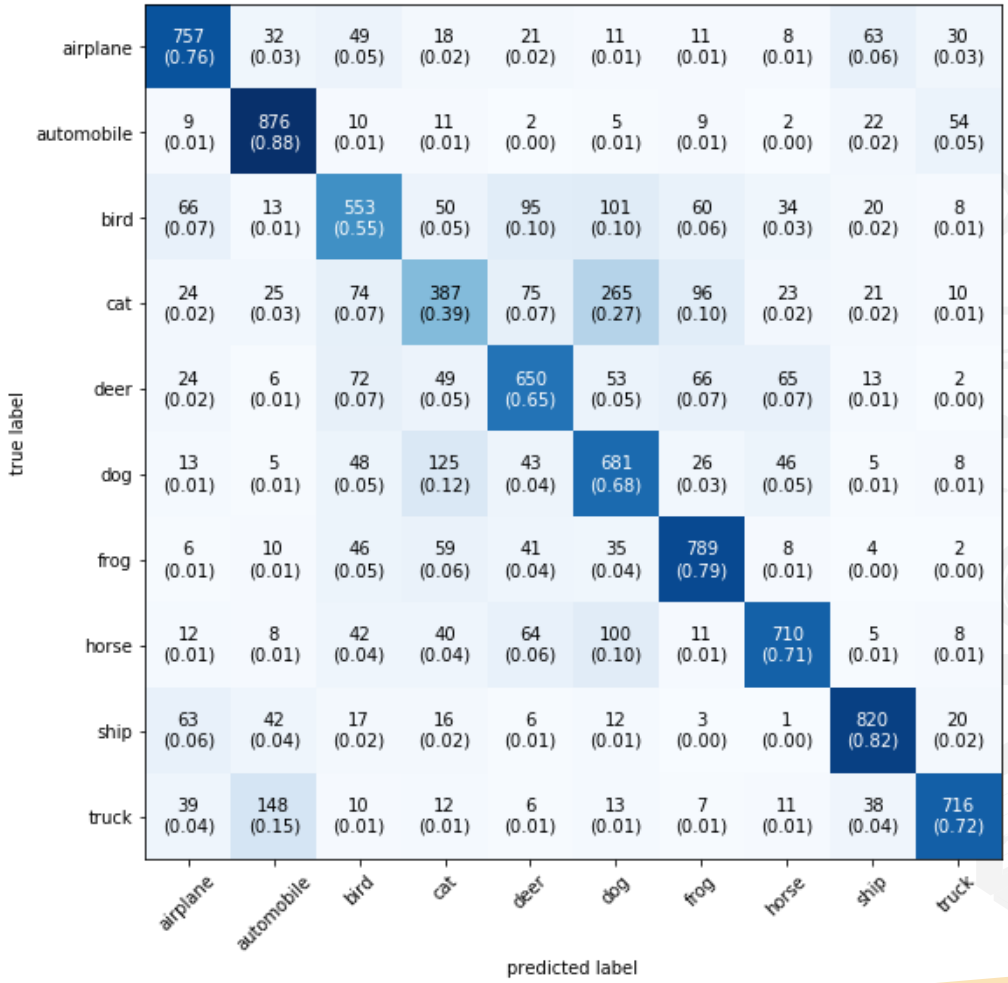
CNN 적용한 Cifar10 이미지 (12)

```
mat = confusion_matrix(y_test, y_pred); mat
```

```
array([[757,  32,  49,  18,  21,  11,  11,   8,  63,  30],
       [  9, 876,  10,  11,   2,   5,   9,   2,  22,  54],
       [ 66,  13, 553,  50,  95, 101,  60,  34,  20,   8],
       [ 24,  25,  74, 387,  75, 265,  96,  23,  21,  10],
       [ 24,   6,  72,  49, 650,  53,  66,  65,  13,   2],
       [ 13,   5,  48, 125,  43, 681,  26,  46,   5,   8],
       [  6,  10,  46,  59,  41,  35, 789,   8,   4,   2],
       [ 12,   8,  42,  40,  64, 100,  11, 710,   5,   8],
       [ 63,  42,  17,  16,   6,  12,   3,   1, 820,  20],
       [ 39, 148,  10,  12,   6,  13,   7,  11,  38, 716]],
```

CNN 적용한 Cifar10 이미지 (13)

```
plot_confusion_matrix(mat,figsize=(9,9), class_names=classes_name, show_normed=True)
```



UST 과학기술연합대학원대학교
UNIVERSITY OF SCIENCE & TECHNOLOGY

02. (따라해보기) 고객이탈 예측 적용

이홍석 (hsyi@kisti.re.kr)


Walking
X-Axis
5
0
−5
−10
Y-Axis
20
10
0
Z-Axis
10
0

UST 과학기술연합대학원대학교
UNIVERSITY OF SCIENCE & TECHNOLOGY

Thank You!

www.ust.ac.kr