AI, DS, ML, DL : BASIC #04

2026년 1월
.NowSome MakerBox.

딥러닝/CNN의 이해

오늘은

ML : scikit-learn

DL : tensorflow

오늘은

scikit-learn : 전통적 ML 알고리즘, 정형 유리

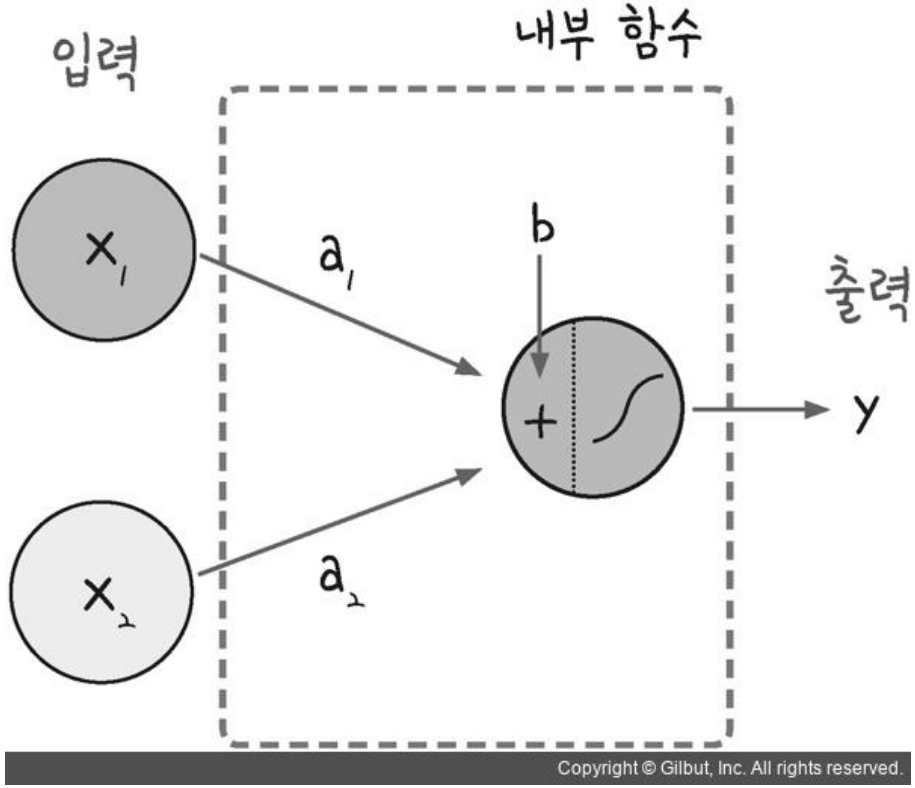
Tensorflow : DL 특화, 대규모 데이터, 비정형 유리

딥러닝

어떻게 예측 할 것인가?

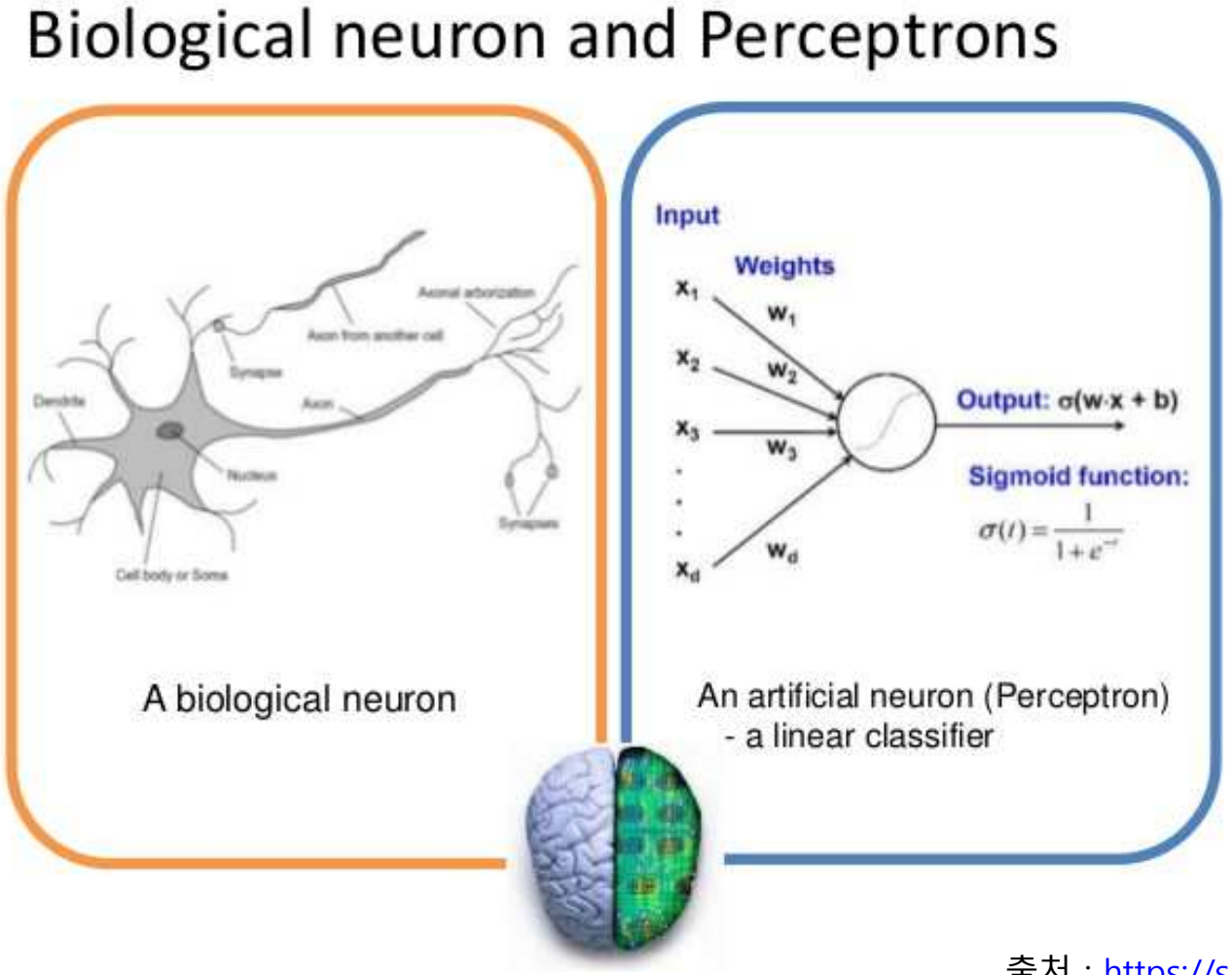
입출력함수 -> 퍼셉트론

$y=a_1x_1 + a_2x_2 + b$ ->



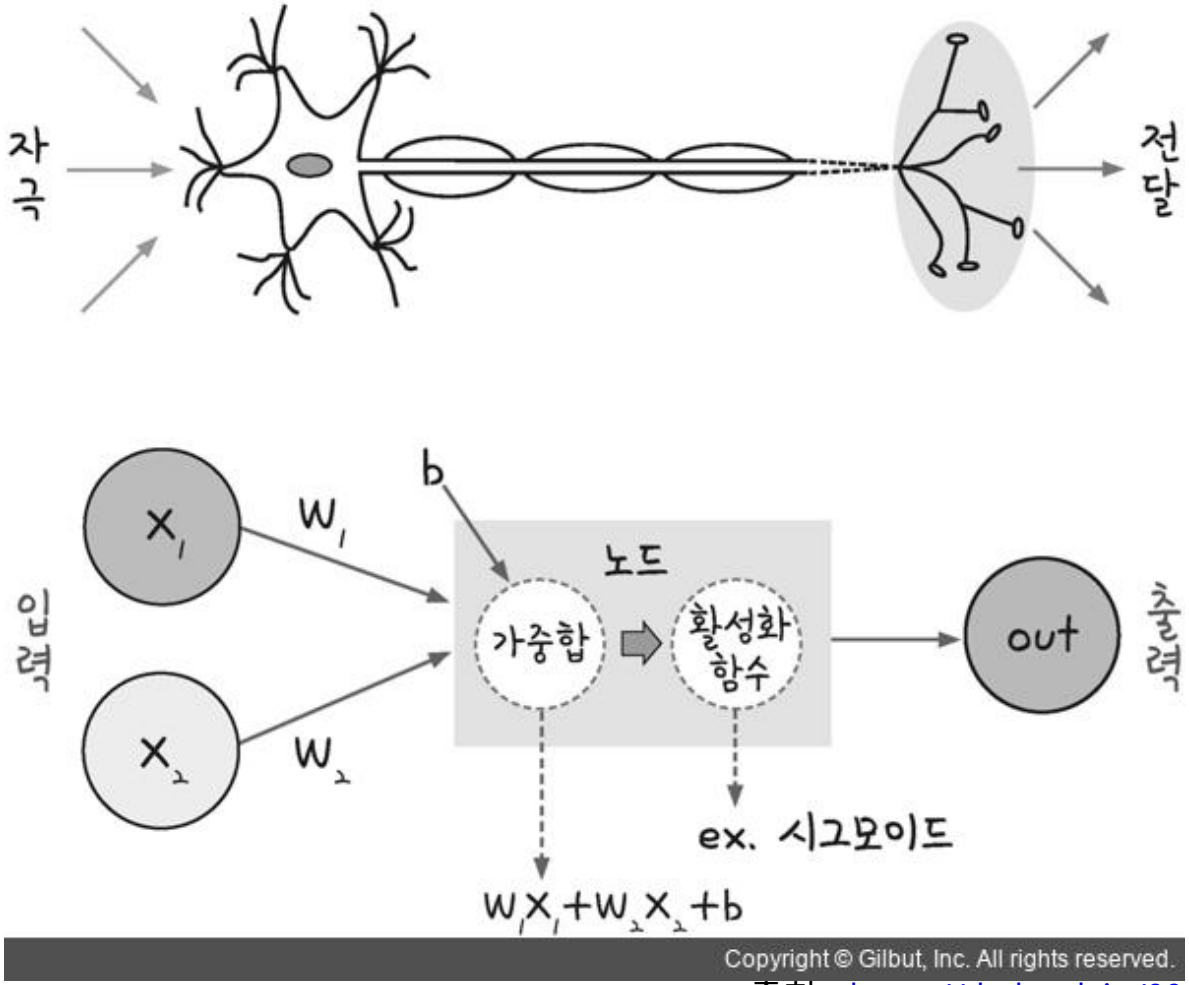
출처 : https://thebook.io/006958/part03/ch06-01/

퍼셉트론 : 신경망의 기본 단위



출처 : https://sacko.tistory.com/10

퍼셉트론 : 신경망의 기본 단위



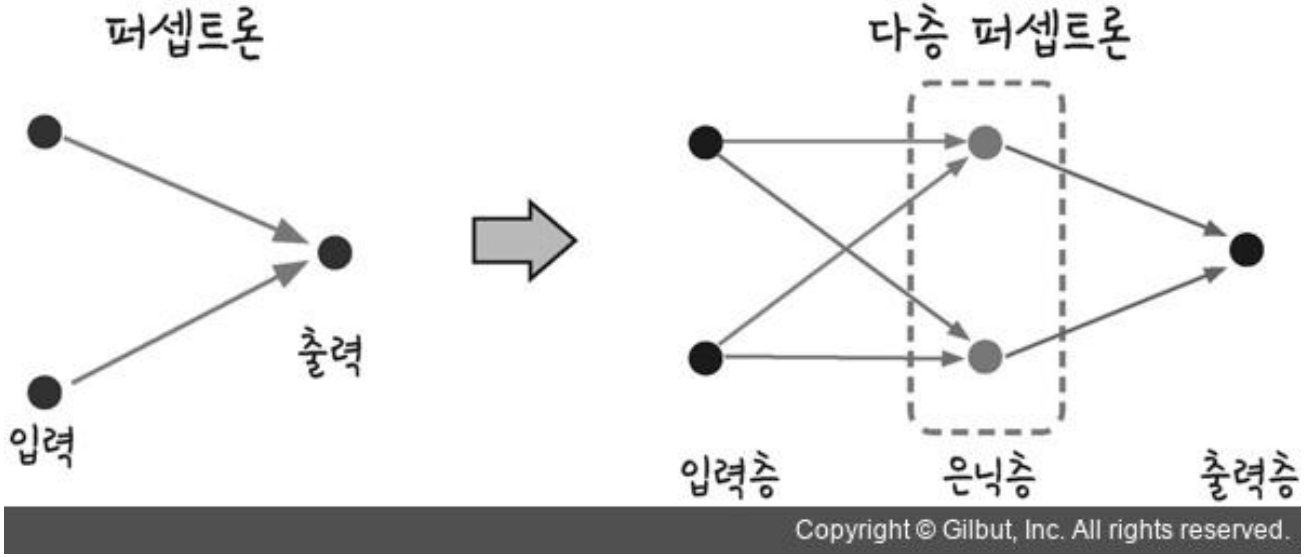
출처 : https://thebook.io/006958/part03/ch06-01/

가중치, 가중합, 바이어스

$$y = ax + b \rightarrow y = wx + b$$

출처 : https://thebook.io/006958/part03/ch06-01/

다층 퍼셉트론



출처 : https://thebook.io/006958/part03/ch06-01/

Perceptron to MLP

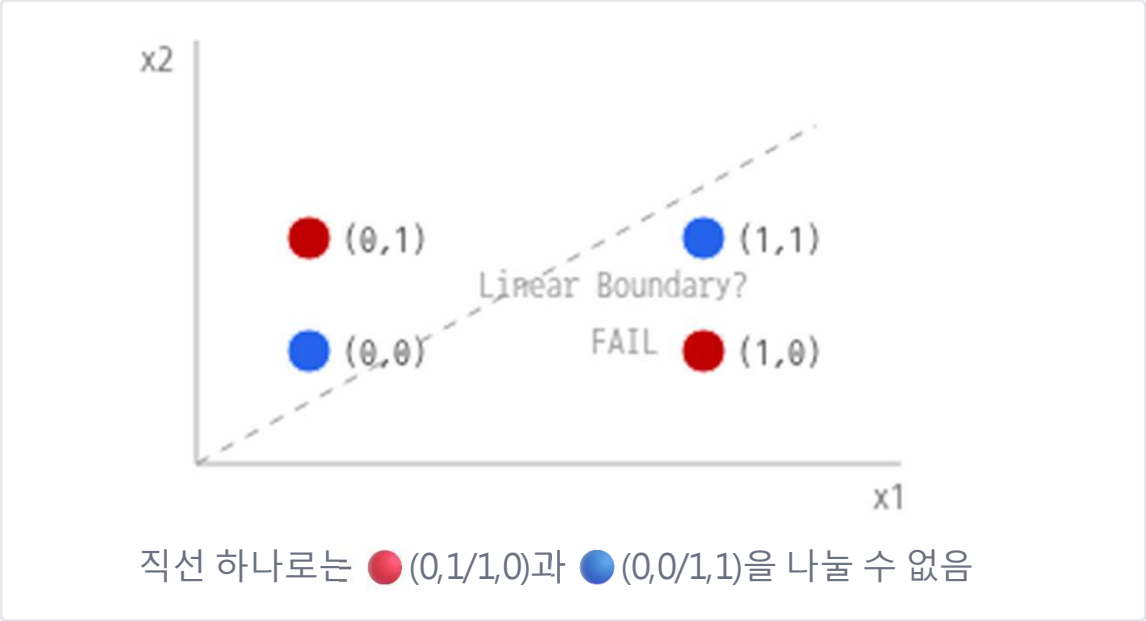
단층 구조의 한계와 다층 신경망의 필요성

1. 퍼셉트론(Perceptron)의 한계

⚠ 선형 분리 불가능(Linear Inseparability): 단층 퍼셉트론은 하나의 직선으로만 데이터를 나눌 수 있음.

- XOR 문제: AND, OR 연산은 가능하지만, 배타적 논리합(XOR)과 같은 비선형 패턴은 학습할 수 없음이 증명됨 (Minsky & Papert, 1969).

The XOR Problem (선형 분리 불가)



Perceptron to MLP

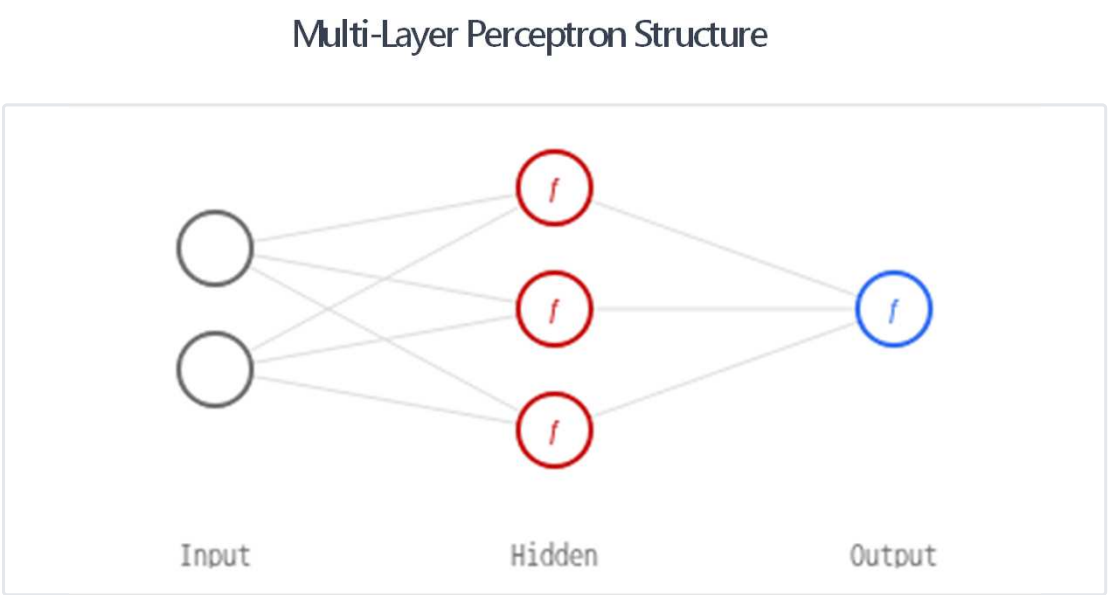
단층 구조의 한계와 다층 신경망의 필요성

2. 다층 퍼셉트론(MLP)의 해결책

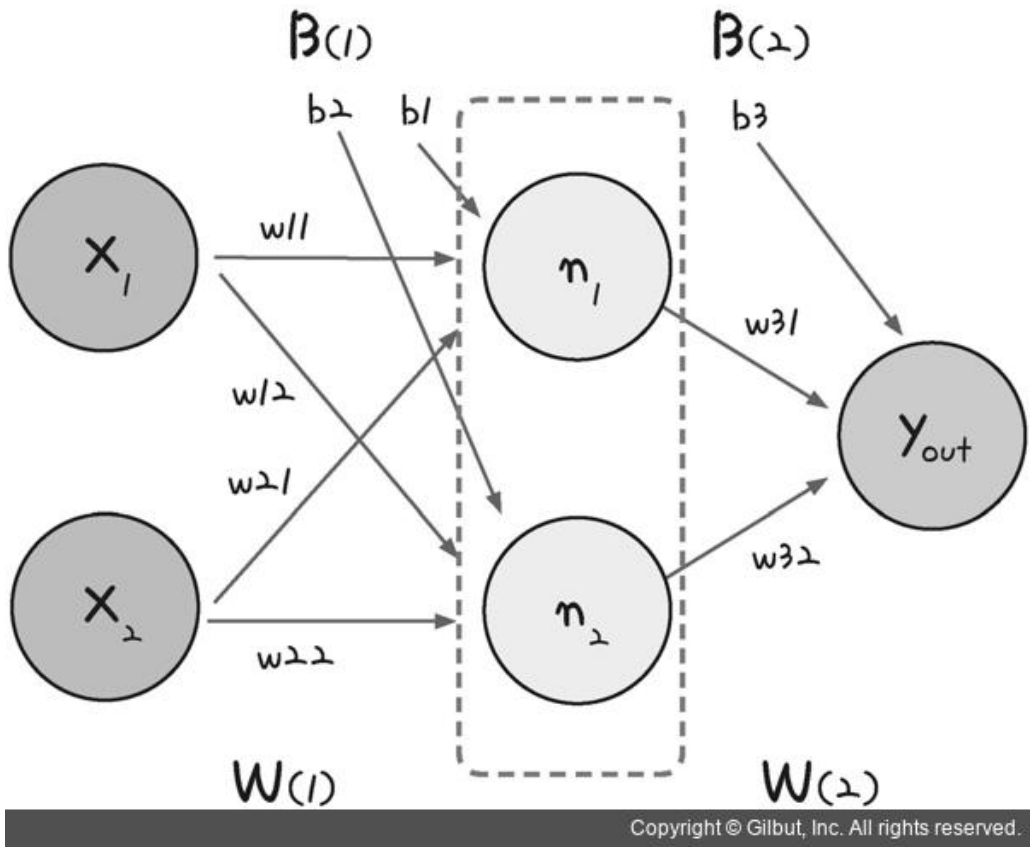
은닉층(Hidden Layer) 추가: 입력층과 출력층 사이에 층을 쌓아 특징 공간을 왜곡/변환

비선형 활성함수(Activation Function): 단순 선형 결합의 반복은 결국 선형, ReLU/Sigmoid 같은 비선형 함수가 필수

범용 근사 정리(Universal Approximation): 충분한 은닉 노드가 있다면 어떤 연속 함수도 근사할 수 있음

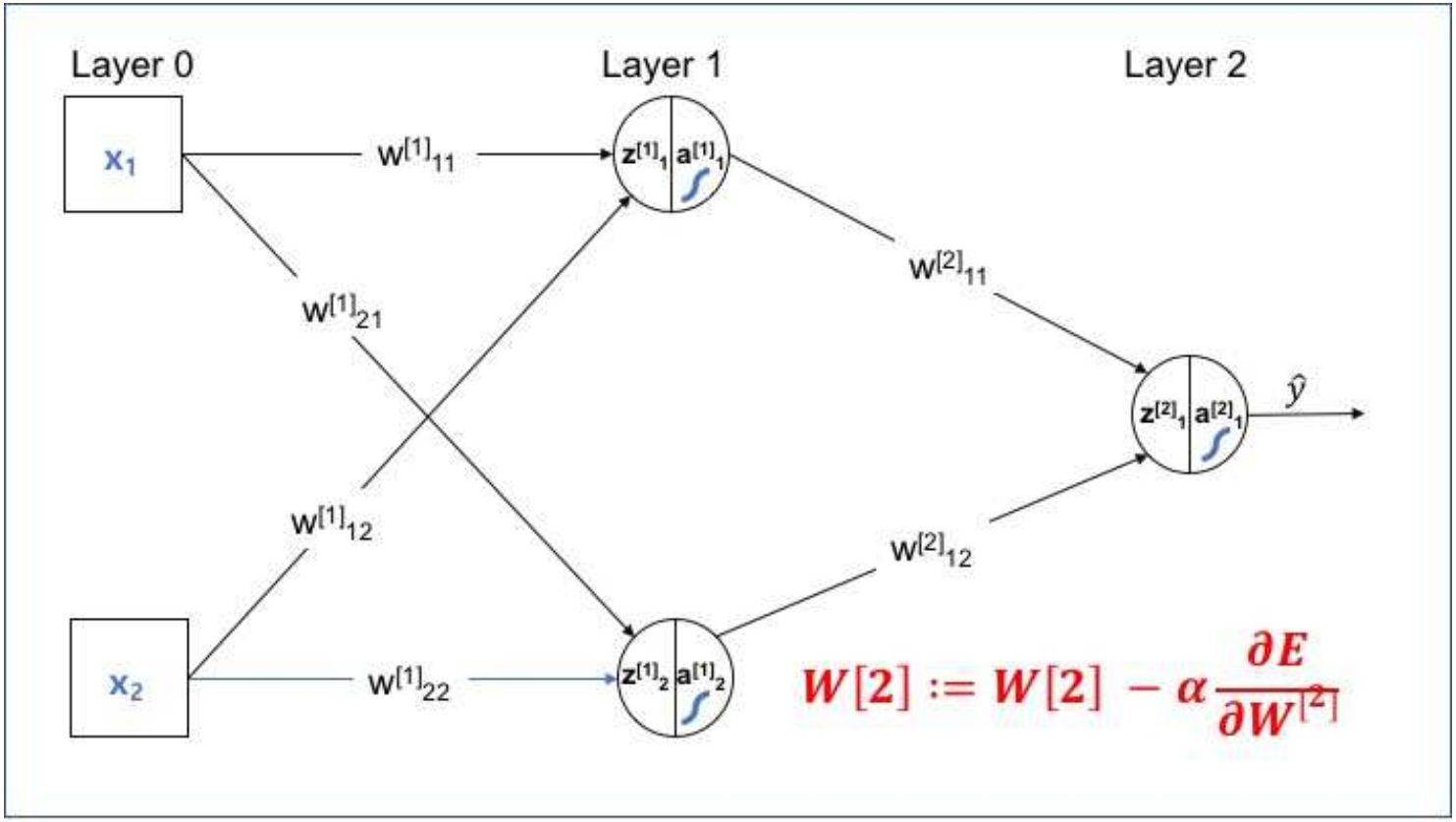


다층 퍼셉트론



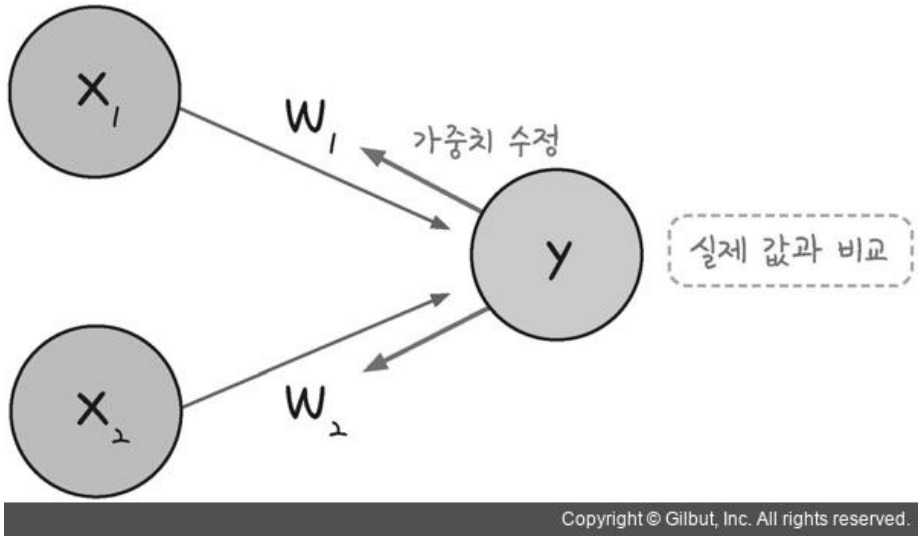
출처 : https://thebook.io/006958/part03/ch06-01/

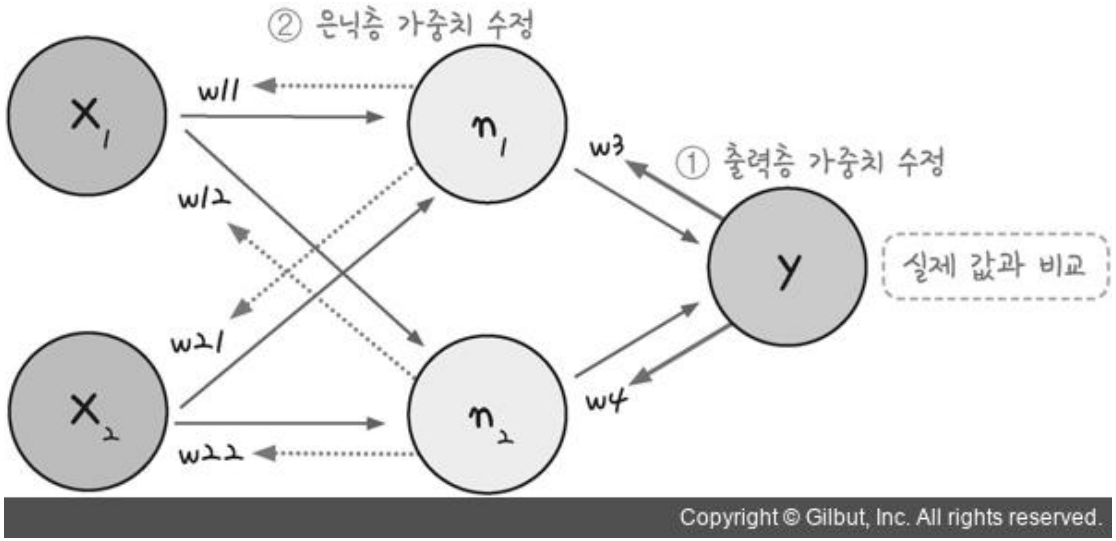
오차 역전파



출처 : http://taewan.kim/post/backpropagation_matrix_transpose/

오차 역전파





출처 : https://thebook.io/006958/part03/ch08/01-01/

Forward & Backward Propagation

딥러닝 모델 학습의 핵심 메커니즘: 예측과 최적화의 순환

→ 1. Forward Propagation (순전파)

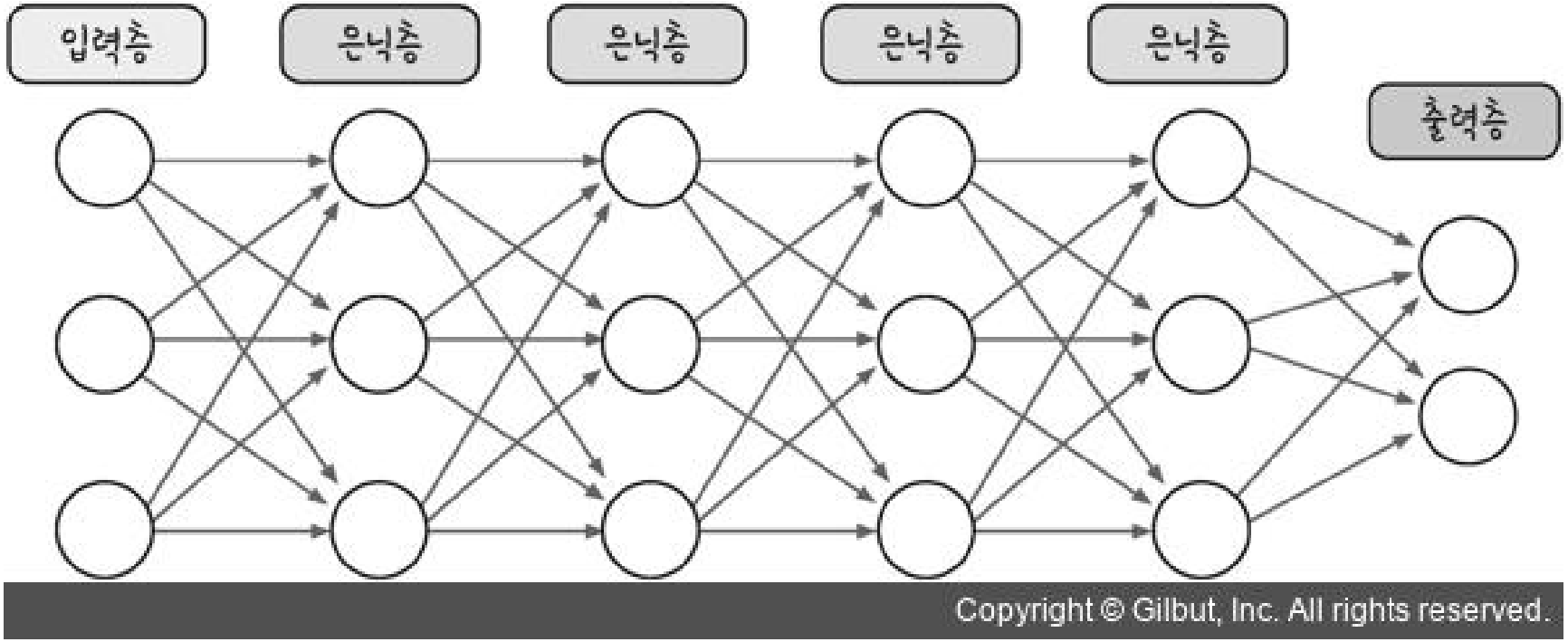
입력 데이터(X)가 신경망의 가중치(W)와 편향(B)을 통과하며 활성함수를 거쳐 최종 예측값을 출력하는 과정입니다.
마지막 단계에서 손실 함수(Loss Function)를 통해 실제값(Y)의 오차를 계산합니다.

← 2. Backward Propagation (역전파)

계산된 손실(Loss)을 줄이기 위해 출력층에서 입력층 방향으로 거슬러 올라가며,
연쇄 법칙(Chain Rule)을 적용해 각 파라미터가 오차에 미친 영향(Gradient)을 계산합니다.

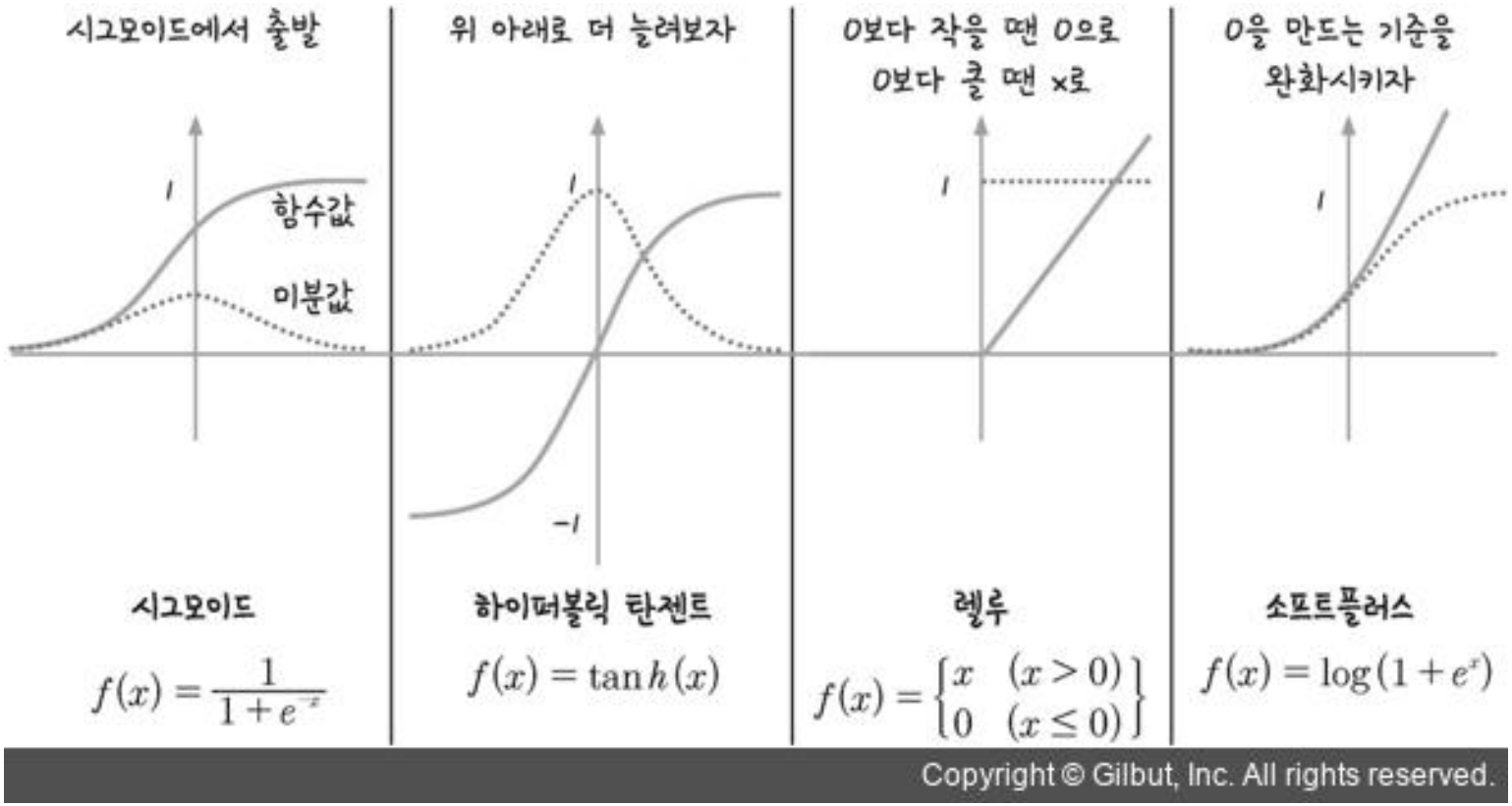
딥러닝

신경망 → 딥러닝 but...



출처 : https://thebook.io/006958/part03/ch09/

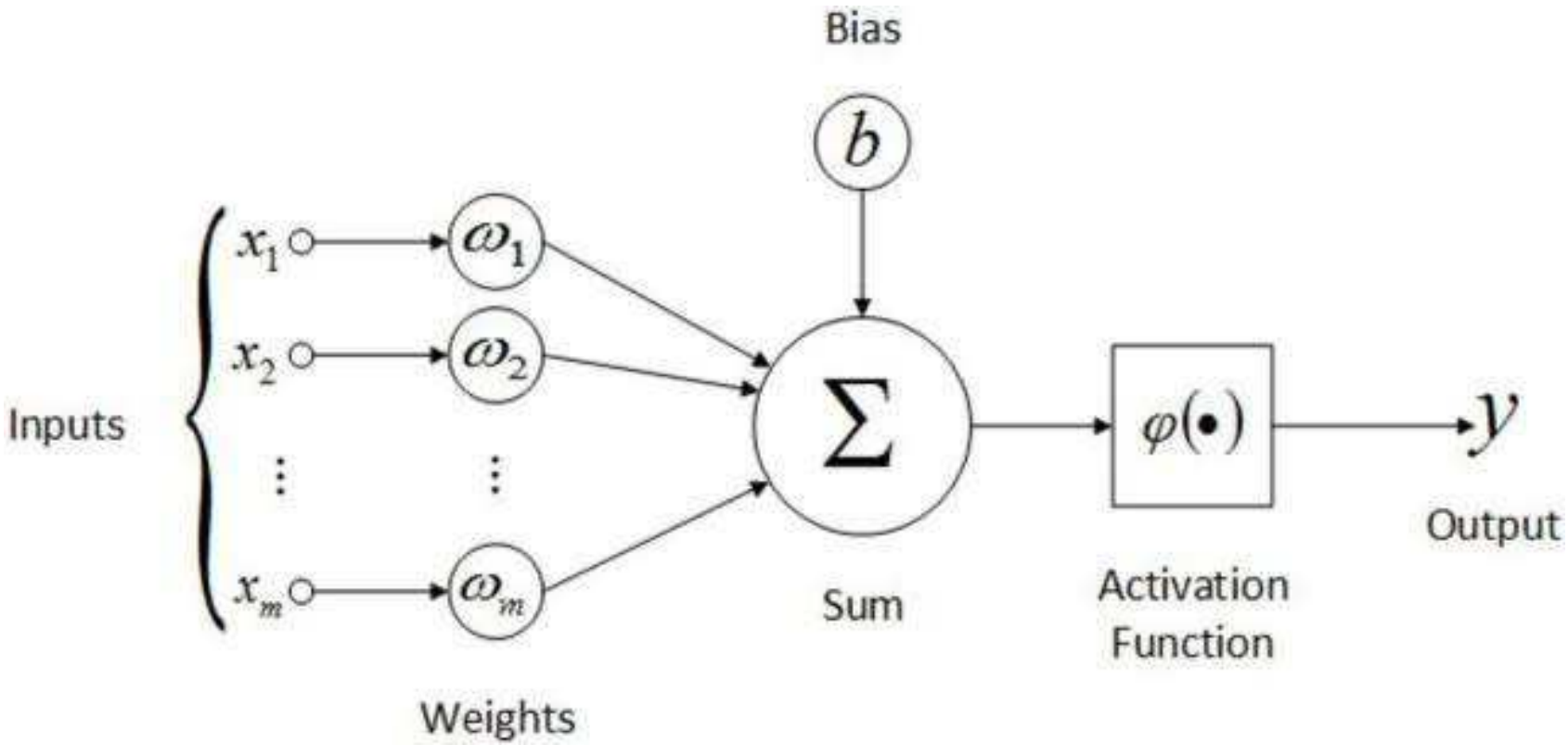
활성화함수, 출력에서 활성화 여부, 정도 결정



출처 : https://thebook.io/006958/part03/ch09/01-01/

DL, CNN

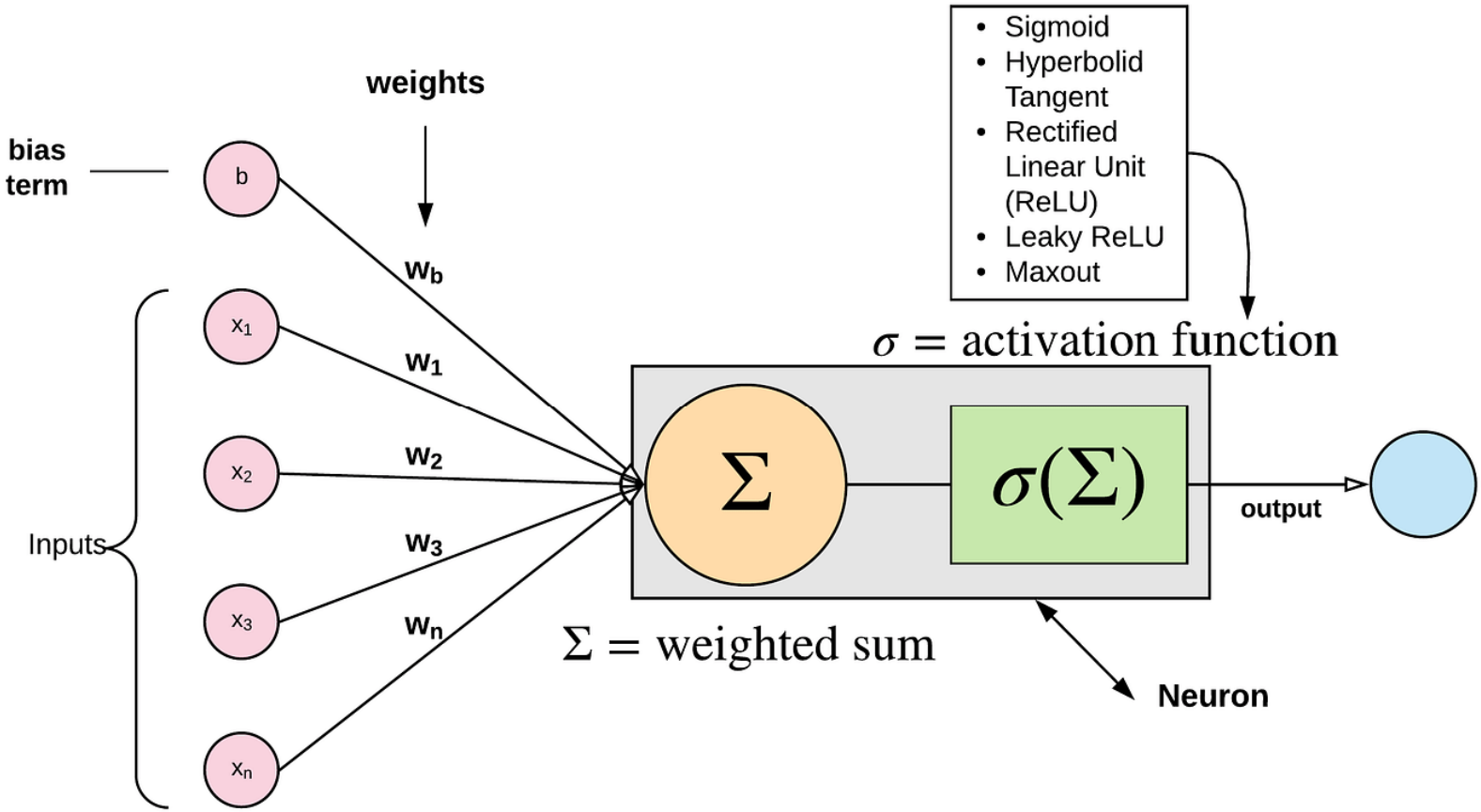
퍼셉트론 : X1(입력)W1(가중치) + ...+ b(바이어스) → 활성화함수 → 출력



Source : https://analyticsindiamag.com/ai-mysteries/perceptron-is-the-only-neural-network-without-any-hidden-layer/

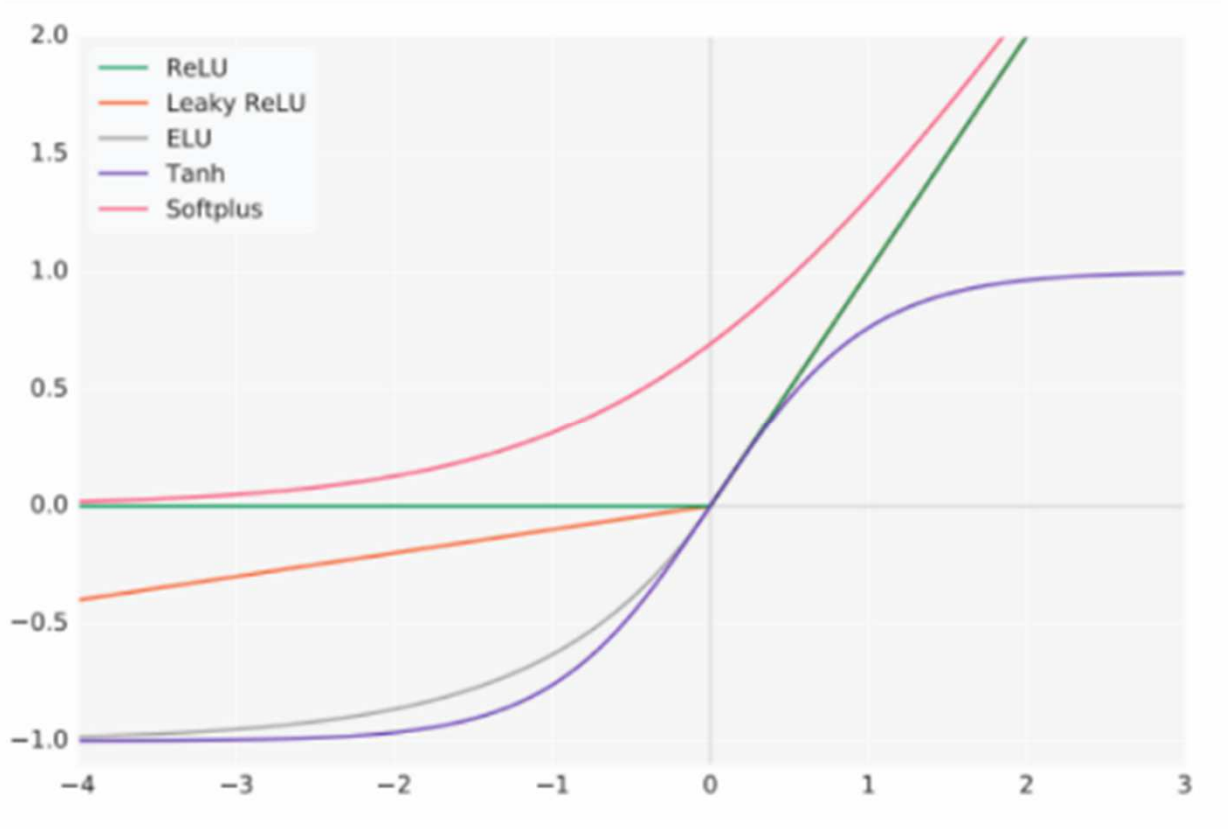
DL, CNN

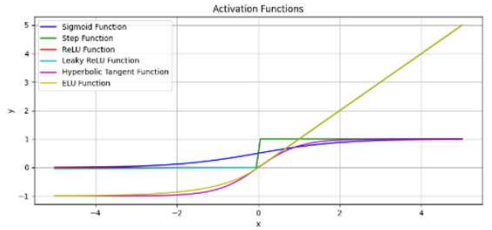
퍼셉트론 : X1(입력)W1(가중치) + ...+ b(바이어스) → 활성화함수 → 출력



Source : https://2809ayushic.medium.com/types-of-activation-functions-fc9e71c2d991

활성화 함수 : 특정 영역을 강조/소거해서 최종 예측의 정확도를 높임





Source : https://velog.io/@kisx97/Machine-Learning-%ED%99%9C%EC%84%B1%ED%95%A8%EC%88%98-activation-function

DL, CNN

활성화 함수 : 특정 영역을 강조/소거해서 최종 예측의 정확도를 높임

Activation Function	Equation	Output Range	Advantages	Disadvantages	Application
Sigmoid	$\frac{1}{1+e^{-x}}$	0 ~ 1	Interpretable as probability in binary classification	Vanishing Gradient problem	Binary classification problems
Tanh	$\frac{2}{1+e^{-2x}} - 1$	-1 ~ 1	Centered at zero, improves learning speed	Vanishing Gradient problem	Recurrent Neural Networks (RNNs)
ReLU	$\max(0, x)$	0 ~ ∞	Simple and fast, mitigates vanishing gradient	Dead neurons problem (0 output)	Image classification, regression tasks
Leaky ReLU	x, αx	-∞ ~ ∞	Solves dead neurons issue	Still limited non-linearity	Tasks where negative values are important
ELU	$\alpha(e^x - 1)$, x	-α ~ ∞	Smooth output, more stable learning	Computationally complex	Deep networks, image processing
Softmax	$\frac{e^{x_i}}{\sum_j e^{x_j}}$	0 ~ 1	Probabilistic interpretation in multi-class classification	Only one class selected at a time	Multi-class classification problems

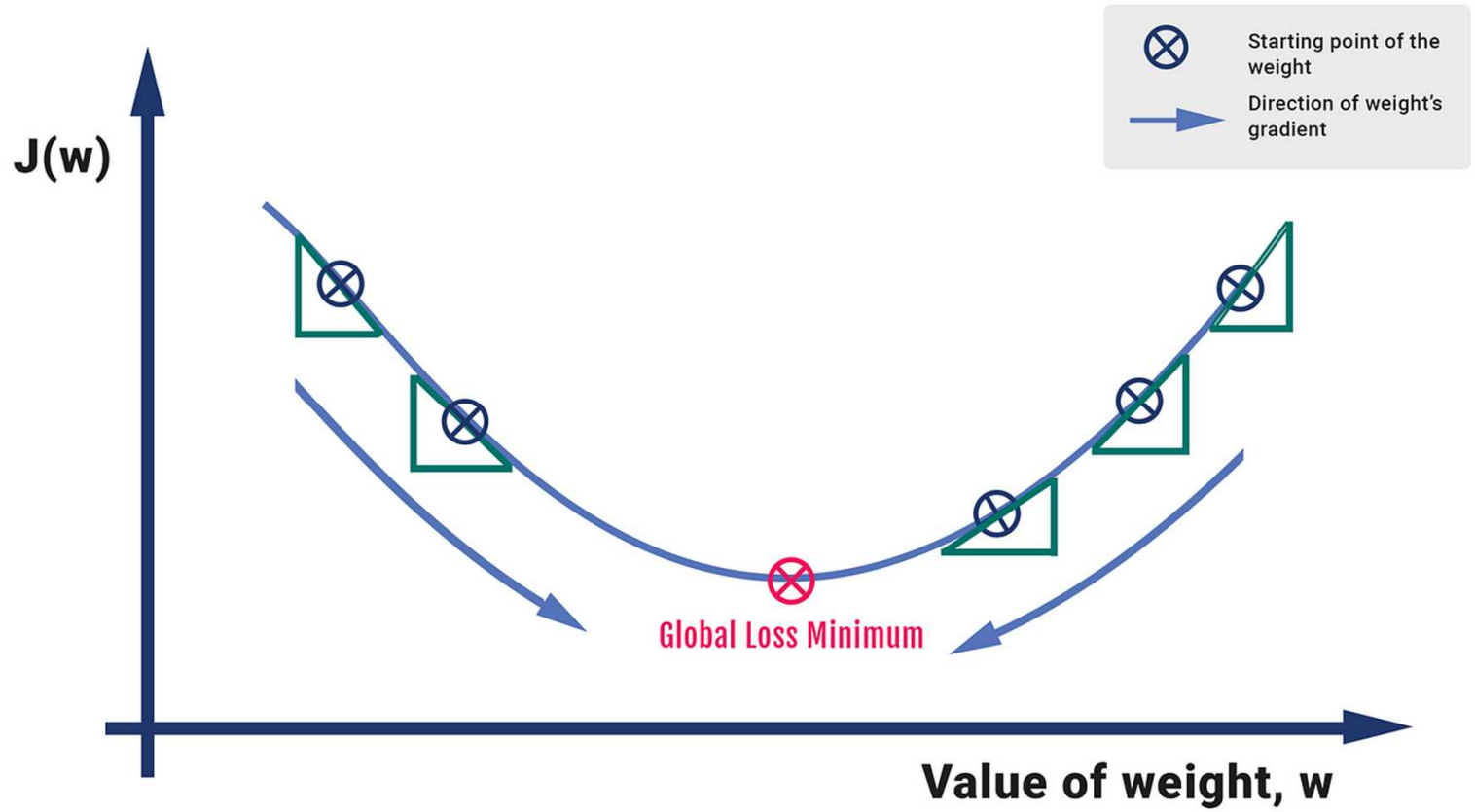
Source : https://velog.io/@kisx97/Machine-Learning-%ED%99%9C%EC%84%B1%ED%95%A8%EC%88%98-activation-function

활성화 함수 정리표

활성화 함수	수식	출력 범위	장점	단점	적용 문제
Sigmoid	$\frac{1}{1+e^{-x}}$	0 ~ 1	이진 분류에서 확률로 해석 가능	Vanishing Gradient 문제	이진 분류 문제
Tanh	$\frac{2}{1+e^{-2x}} - 1$	-1 ~ 1	중앙값이 0에 가까워 학습 속도 개선	Vanishing Gradient 문제	순환 신경망(RNN)
ReLU	$\max(0, x)$	0 ~ 무한대	계산이 간단하고 빠름, Vanishing 문제 감소	Dead Neurons 문제 (0 출력)	이미지 분류, 회귀 문제
Leaky ReLU	$x,\ \alpha x$	-무한대 ~ 무한대	Dead Neurons 문제 해결	여전히 비선형성 부족	음수 값이 중요한 문제
ELU	$\alpha(e^x - 1),\ x$	-α ~ 무한대	부드러운 출력으로 안정적 학습	계산이 복잡함	딥 신경망, 영상 처리
Softmax	$\frac{e^{x_i}}{\sum_j e^{x_j}}$	0 ~ 1	다중 클래스 분류에서 확률적 해석 가능	한 번에 하나의 클래스만 분류 가능	다중 클래스 분류 문제 (ex. 이미지 분류)

Source : https://velog.io/@kisx97/Machine-Learning-%ED%99%9C%EC%84%B1%ED%95%A8%EC%88%98-activation-function

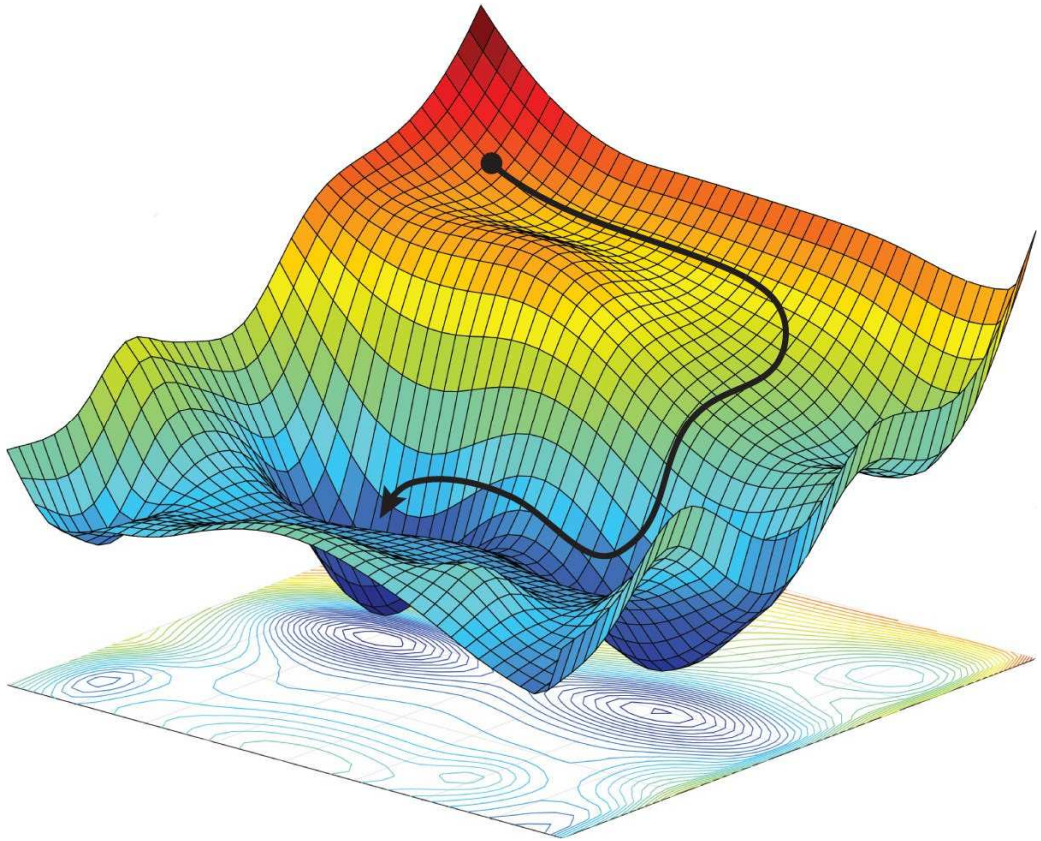
옵티마이저 : 모델 학습과정에서 손실함수를 최소화하도록, 가중치를 업데이트하는 방법



Source : https://towardsdatascience.com/gradient-descent-3a7db7520711

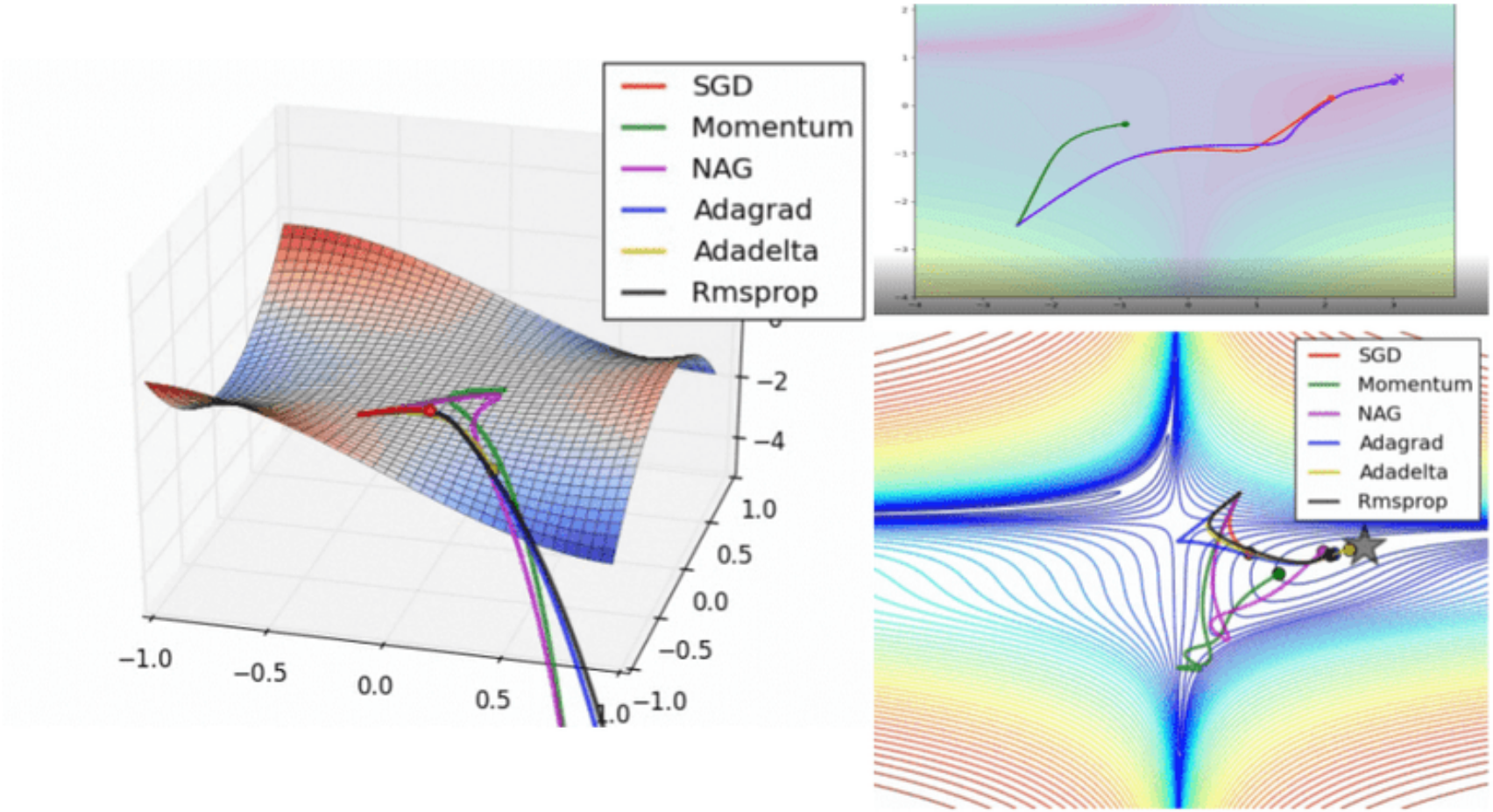
DL, CNN

옵티마이저 : 모델 학습과정에서 손실함수를 최소화하도록, 가중치를 업데이트하는 방법



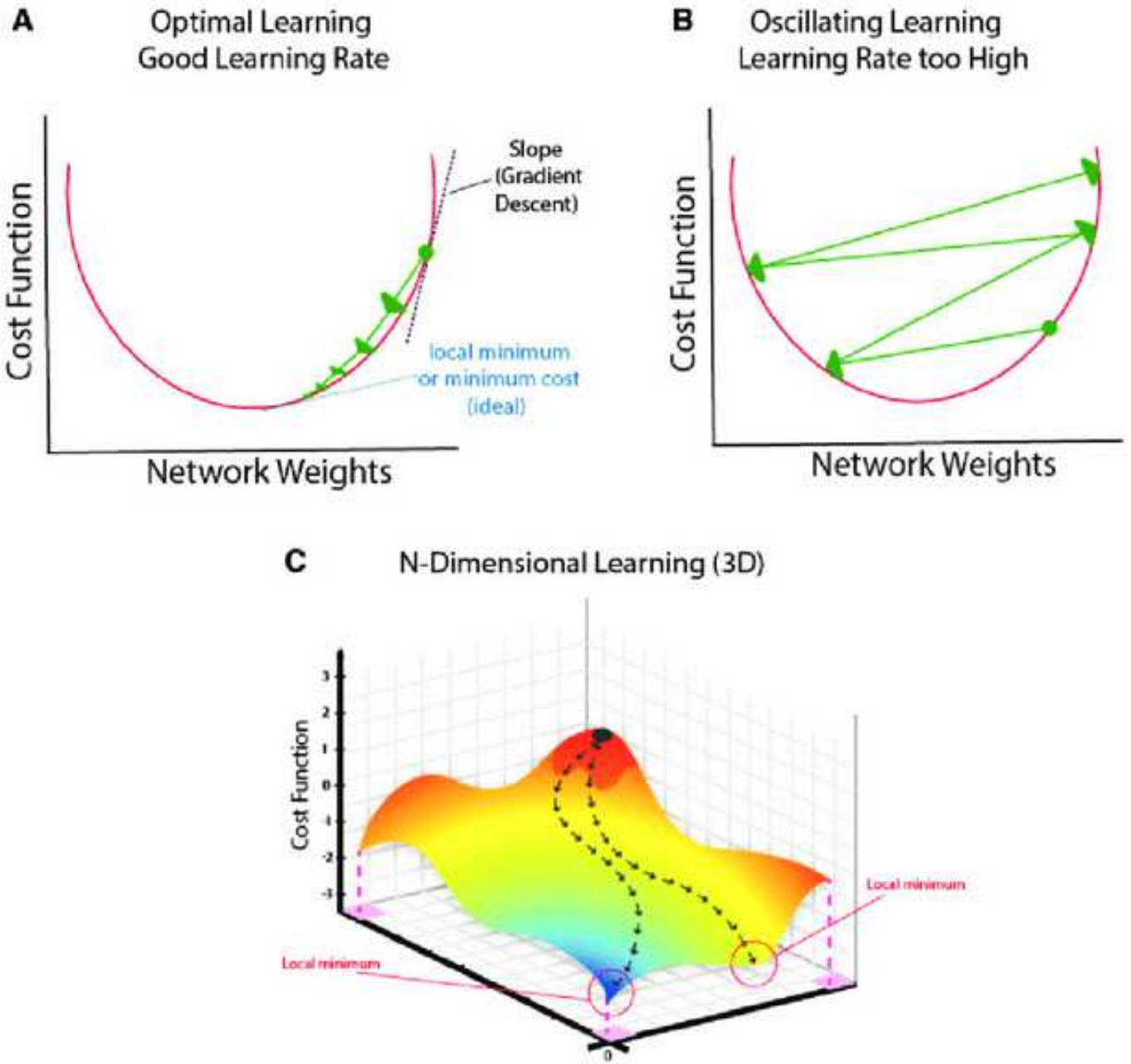
Source : https://www.greghilston.com/post/3_optimizers/

옵티마이저 : 모델 학습과정에서 손실함수를 최소화하도록, 가중치를 업데이트하는 방법



Source : https://theaisummer.com/optimization/

DL,



Source : https://www.researchgate.net/figure/mpact-of-deep-learning-design-on-learning-effect-of-learning-rate-A-Efficient_fig2_331439931

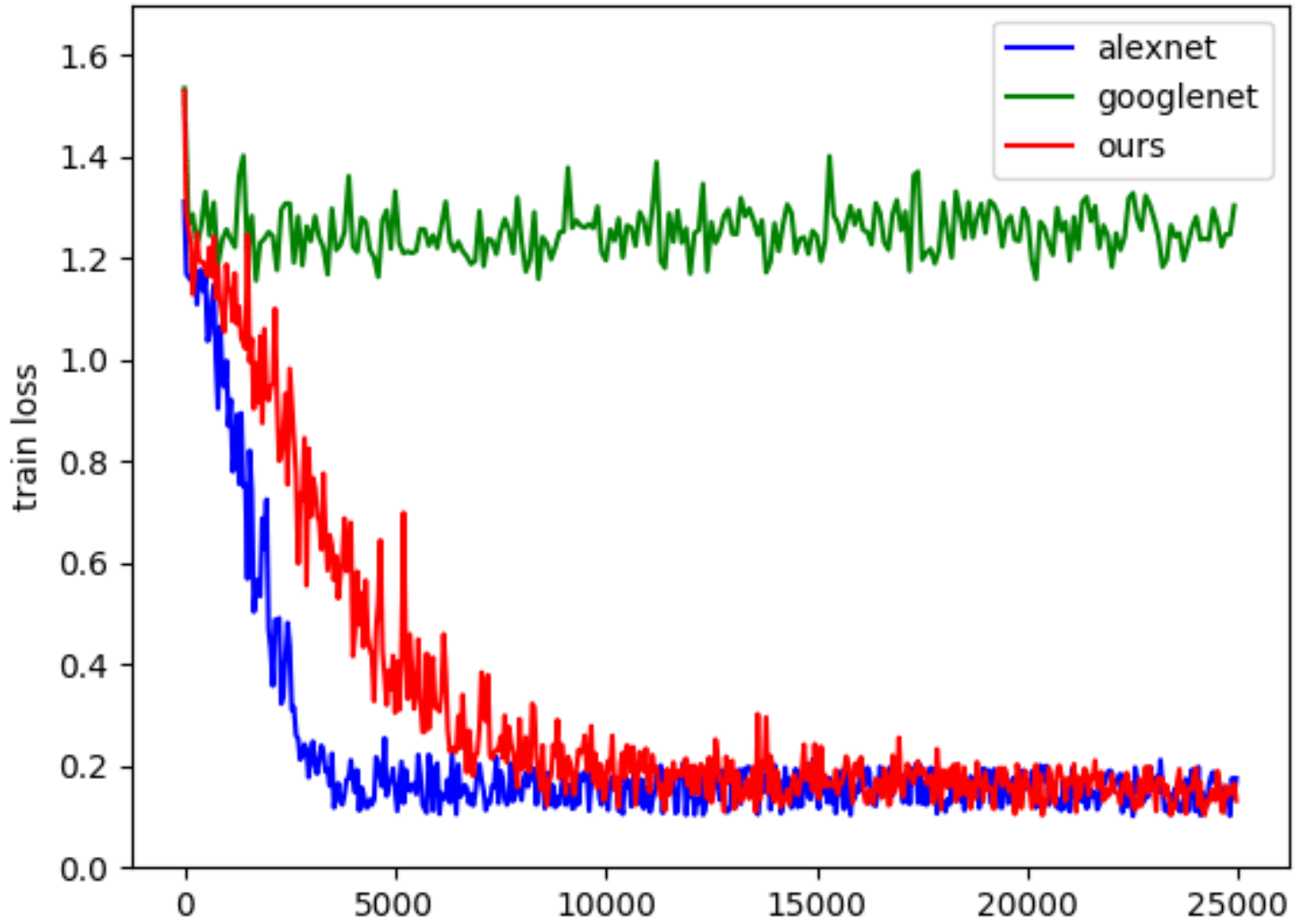
DL, CNN

옵티마이저	개념	장점	단점	적용 문제
Gradient Descent	전체 데이터셋을 사용하여 가중치 업데이트	간단하고 이론적으로 명확함	학습 속도가 느리고 지역 최솟값에 빠질 수 있음	작은 데이터셋, 이론적 연구
SGD	데이터셋의 랜덤 샘플을 사용하여 가중치 업데이트	메모리 효율적, 빠른 학습	손실 값의 변동이 크며 부드럽게 수렴하지 않을 수 있음	대규모 데이터셋, 이미지 분류, 텍스트 분석
Momentum	이전 업데이트 방향을 고려하여 관성을 추가	진동 감소, 빠른 수렴	관성 값 설정이 필요함	이미지 분류, 회귀 문제
NAG	Momentum의 향상된 버전으로, 미래 기울기를 미리 예측하여 가중치 업데이트	빠른 수렴, 더 나은 예측 능력	계산 비용이 더 크며 파라미터 조정이 필요함	복잡한 신경망 구조, 이미지 인식 문제
Adagrad	피처의 빈도에 따라 학습률을 조정	희소한 데이터와 드문 피처에 유리	학습이 진행될수록 학습률이 너무 작아짐	자연어 처리, 희소한 데이터
RMSprop	지수 가중 이동 평균을 사용하여 학습률 동적으로 조정	Adagrad의 학습률 감소 문제 해결	최적의 학습률 찾기 어려움	RNN, 시계열 분석 문제
Adam	Momentum과 RMSprop 결합, 1차 및 2차 모멘트 사용	빠른 학습 속도, 다양한 문제에 적합	메모리 사용량이 많고 파라미터 튜닝 필요	CNN, RNN, 이미지 및 텍스트 분석
Adamax	Adam의 변형, 무한 노름을 사용하여 가중치 업데이트	Adam보다 안정적, 고차원 데이터에 유리	큰 데이터셋에서는 성능 저하 가능	고차원 데이터 분석
Nadam	Adam에 Nesterov 관성 적용	빠른 수렴 속도, 안정적인 학습	Adam보다 계산 비용이 큼	복잡한 신경망 구조, NLP, 이미지 처리

Source : https://velog.io/@kisx97/Machine-Learning-%ED%99%9C%EC%84%B1%ED%95%A8%EC%88%98-activation-function

DL, CNN

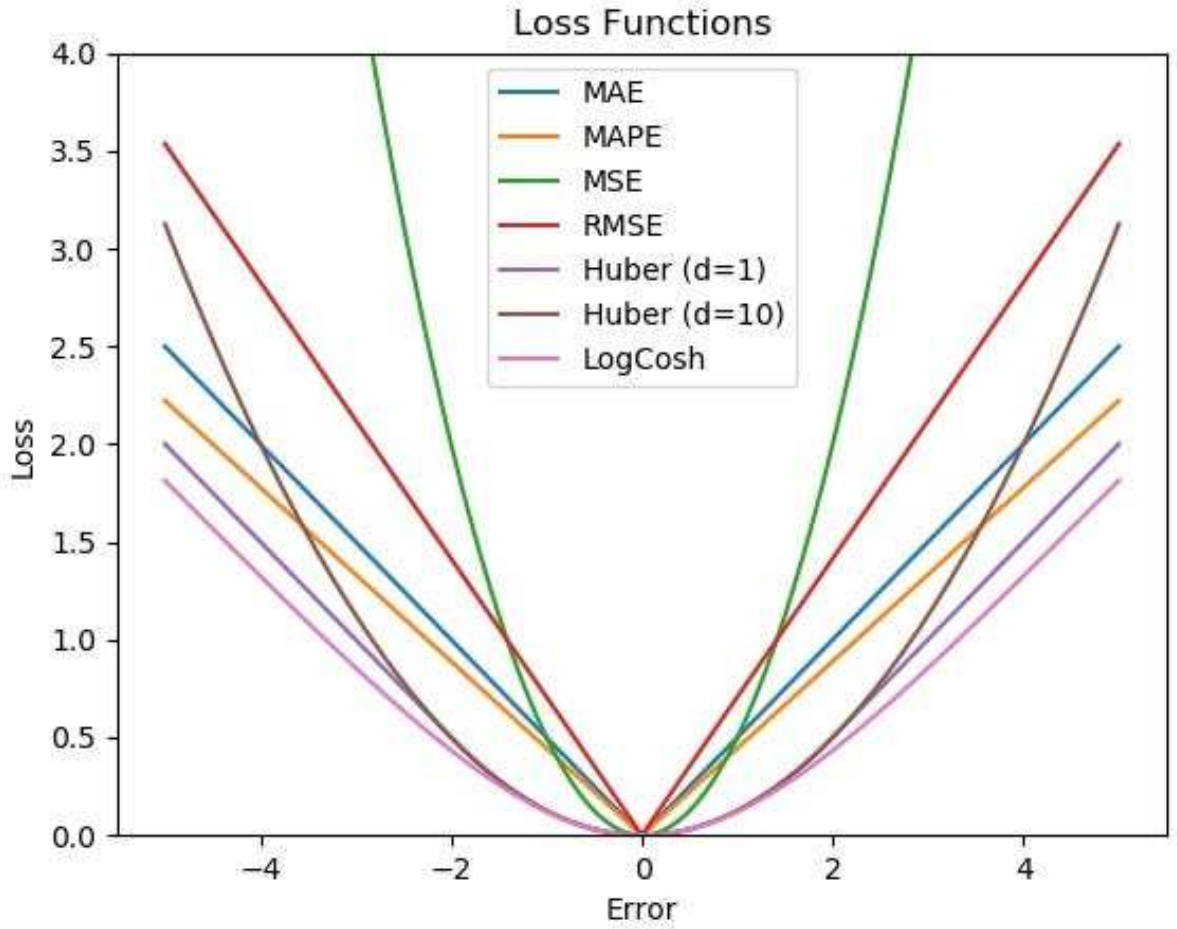
손실함수 : 모델 학습과정에서 모델과 정답의 차이를 보여줌



Source : https://www.researchgate.net/figure/Training-loss-and-test-accuracy_fig3_326103682

DL, CNN

손실함수 : 모델 학습과정에서 모델과 정답의 차이를 보여줌



Source : https://medium.com/analytics-vidhya/a-comprehensive-guide-to-loss-functions-part-1-regression-ff8b847675d6

DL, CNN

손실 함수	설명	장점	단점	쓰이는 곳
Mean Squared Error (MSE)	예측 값과 실제 값의 차이를 제곱해서 평균을 내는 방식.	오류가 클수록 더 큰 패널티를 줌.	작은 차이에도 큰 영향.	회귀 문제 (숫자 예측).
Mean Absolute Error (MAE)	예측 값과 실제 값의 차이를 절대값으로 계산해서 평균을 냄.	이상치에 덜 민감함.	오류가 작은 경우 더 느리게 학습될 수 있음.	회귀 문제, 이상치가 많은 데이터.
Cross-Entropy	분류 문제에서 주로 사용, 확률을 바탕으로 예측의 차이를 계산.	확률적 예측에서 매우 효율적임.	예측 확률이 0에 가까우면 손실 값이 커짐.	이진 및 다중 클래스 분류 문제.
Huber Loss	MSE와 MAE의 중간 형태로, 오류가 크면 MAE처럼 계산하고 작으면 MSE처럼 계산.	이상치에 강건하며, 안정적인 학습 가능.	미세한 조정 필요.	회귀 문제, 이상치가 포함된 데이터.
Hinge Loss	이진 분류 문제에서 사용, 데이터가 제대로 분류되면 손실이 없고, 잘못되면 손실을 줌.	분류 경계가 확실할 때 유용함.	소프트 클래스 분류에 적합하지 않음.	이진 분류 문제 (SVM).
Kullback-Leibler Divergence (KL Divergence)	두 확률 분포 간의 차이를 계산.	확률 분포 간 차이를 효율적으로 계산 가능.	계산이 복잡할 수 있음.	확률 분포 학습 문제 (GAN 등).

Source : https://medium.com/analytics-vidhya/a-comprehensive-guide-to-loss-functions-part-1-regression-ff8b847675d6

TensorFlow & Keras Architecture

딥러닝 모델 개발의 표준 라이프사이클

1. 모델 정의 (Define)
레이어를 쌓아 신경망 구조 설계
`model = tf.keras.Sequential([...])`

2. 컴파일 (Compile)
최적화 도구 및 손실 함수 설정
`model.compile(optimizer='adam', ...)`

3. 학습 (Fit)
데이터로 가중치 최적화
`model.fit(x_train, y_train, epochs=10)`

4. 평가 (Evaluate)
검증 데이터로 성능 측정
`model.evaluate(x_test, y_test)`

5. 추론 (Predict)
새로운 데이터 결과 예측
`model.predict(new_data)`

TensorFlow & Keras Architecture

```
# 모델 설정
model = Sequential()
model.add(Dense(12,  input_dim=4, activation='relu'))
model.add(Dense(8,  activation='relu'))
model.add(Dense(3, activation='softmax'))
model.summary()

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

# 모델 실행
history=model.fit(X, y, epochs=30, batch_size=5)
```

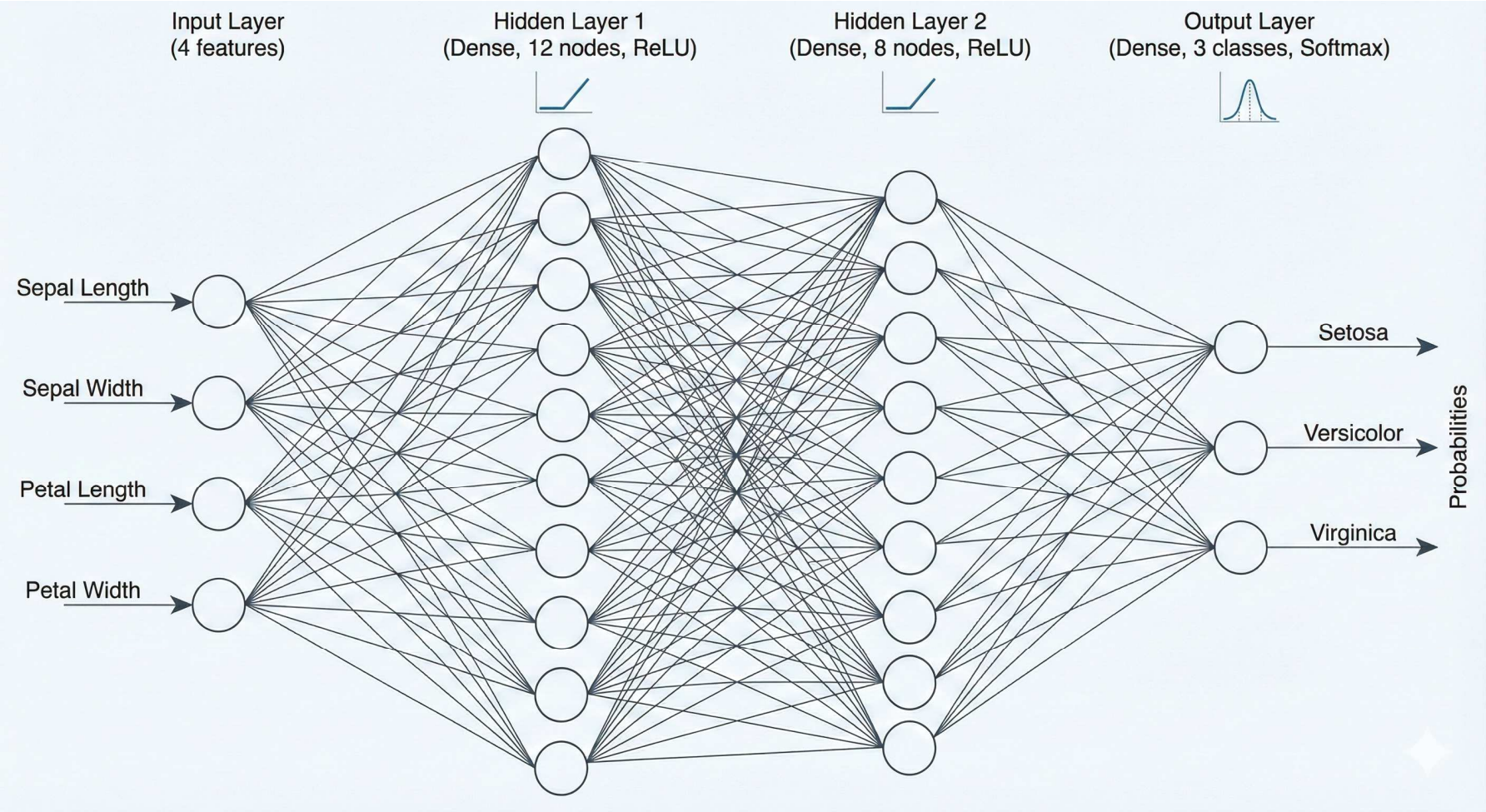
TensorFlow & Keras Architecture

- **Batch size** : 한 번의 forward/backward에 넣는 샘플 개수 (예: 32)
- **Epoch** : 데이터셋 전체를 1번 “끝까지” 학습에 사용한 횟수 (예: 50 epoch)

- 작은 batch:
  때로는 **일반화에 유리**하다는 경험적 경향이 있음
  대신 학습이 불안정해 보일 수 있어 학습률 튜닝 필요할 수도

- 큰 batch:
  gradient가 비교적 안정적(변화량 작은 배치에 비해 감소)
  GPU 활용 효율 좋은 경향

TensorFlow & Keras Architecture



TensorFlow & Keras Architecture

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_3 (Dense)	(None, 12)	60
dense_4 (Dense)	(None, 8)	104
dense_5 (Dense)	(None, 3)	27

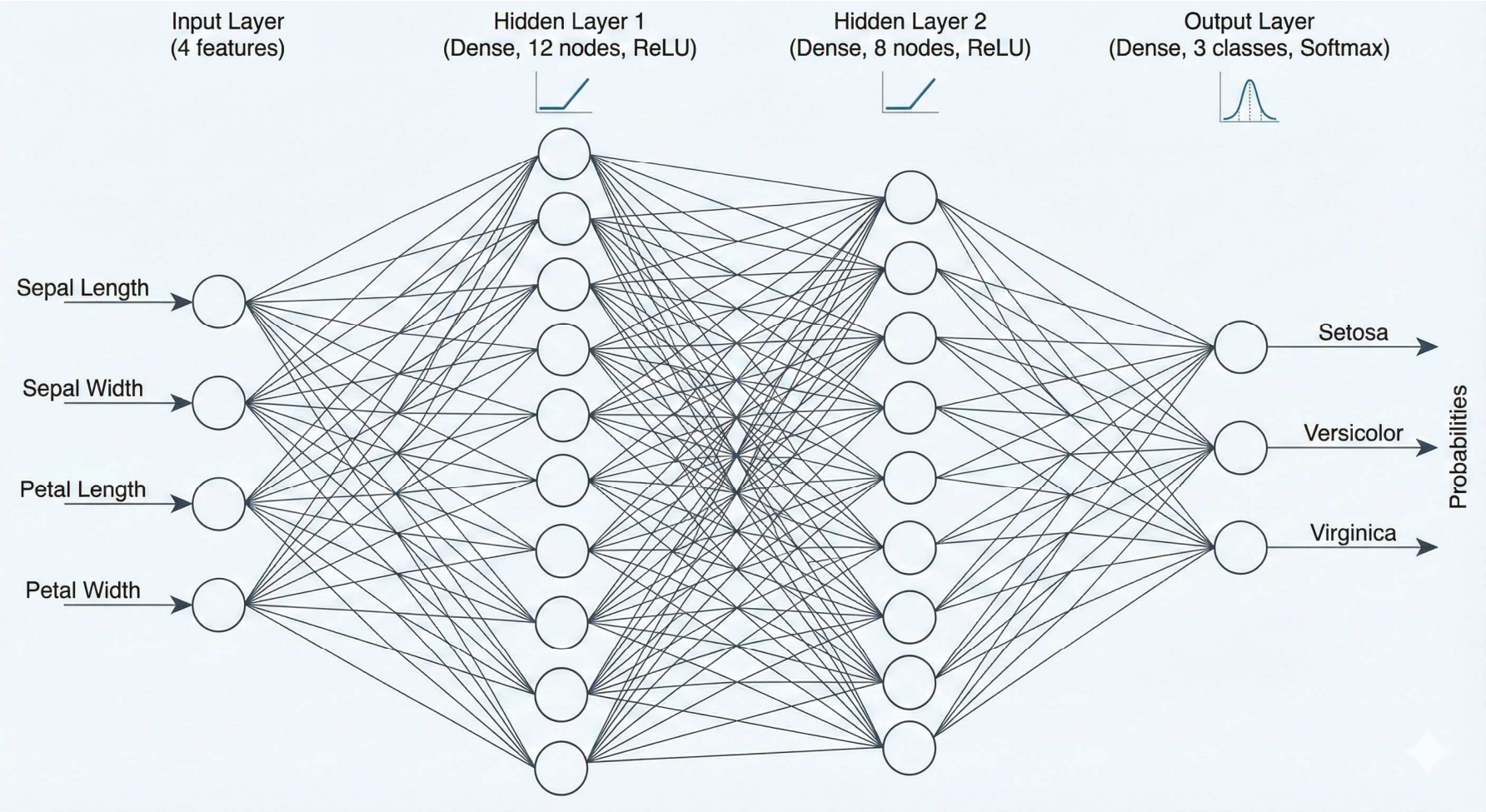
Total params: 191 (764.00 B)
Trainable params: 191 (764.00 B)
Non-trainable params: 0 (0.00 B)

아이리스 꽃, Tensorflow 활용

TensorFlow & Keras Architecture

```
# 모델 설정
model = Sequential()
model.add(Dense(12,  input_dim=4, activation='relu'))
model.add(Dense(8,  activation='relu'))
model.add(Dense(3, activation='softmax'))
model.summary()

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

# 모델 실행
history=model.fit(X, y, epochs=30, batch_size=5)
```

TensorFlow & Keras Architecture



TensorFlow & Keras Architecture

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_3 (Dense)	(None, 12)	60
dense_4 (Dense)	(None, 8)	104
dense_5 (Dense)	(None, 3)	27

Total params: 191 (764.00 B)
Trainable params: 191 (764.00 B)
Non-trainable params: 0 (0.00 B)

딥러닝

```
import numpy as np
import tensorflow as tf
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder

# 1. 아이리스 데이터셋 로드
iris = load_iris()
X = iris['data']  # 입력 데이터: 꽃받침 길이, 꽃받침 너비, 꽃잎 길이, 꽃잎 너비
y = iris['target'].reshape(-1, 1)  # 출력 데이터: 세 가지 붓꽃 품종 (0, 1, 2)

# 2. One-Hot Encoding: y값을 One-Hot 형태로 변환 (예: 0 -> [1, 0, 0], 1 -> [0, 1, 0])
encoder = OneHotEncoder(sparse_output=False)
y_onehot = encoder.fit_transform(y)

# 3. 훈련 데이터와 테스트 데이터로 나누기 (train: 80%, test: 20%)
X_train, X_test, y_train, y_test = train_test_split(X, y_onehot, test_size=0.2, random_state=42)

# 4. TensorFlow 모델 구성
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(10, activation='relu', input_shape=(X_train.shape[1],)),  # 첫 번째 Dense 레이어, ReLU 활성화 함수
    tf.keras.layers.Dense(10, activation='relu'),  # 두 번째 Dense 레이어
    tf.keras.layers.Dense(3, activation='softmax')  # 출력 레이어, 세 가지 클래스(품종)를 분류하기 위한 softmax
])

# 5. 모델 컴파일: 손실 함수(categorical_crossentropy), 옵티마이저(Adam), 평가 지표(accuracy)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# 6. 모델 학습
history = model.fit(X_train, y_train, epochs=50, batch_size=10, validation_split=0.2)
```

보스턴 집값, DL 버전

딥러닝

데이터 전처리

```
[ ]  import pandas as pd
     import numpy as np

     # 데이터 다운로드 및 전처리
     data_url = "http://lib.stat.cmu.edu/datasets/boston"
     raw_df = pd.read_csv(data_url, sep="₩s+", skiprows=22, header=None)
     data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
     target = raw_df.values[1::2, 2]

     # 특성 이름 정의
     feature_names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT']

     # 판다스 데이터프레임으로 변환
     data = pd.DataFrame(data, columns=feature_names)
     target = pd.DataFrame(target, columns=['Target'])

     # 데이터셋 크기
     print("데이터 shape:", data.shape)
     print("타겟 shape:", target.shape)
```

딥러닝

신경망 학습

```
[ ]  # 심층 신경망
     from tensorflow.keras import Sequential
     from tensorflow.keras.layers import Dense
     def build_model(num_input=1):
         model = Sequential()
         model.add(Dense(128, activation='relu', input_dim=num_input))
         model.add(Dense(64, activation='relu'))
         model.add(Dense(32, activation='relu'))
         model.add(Dense(16, activation='relu'))
         model.add(Dense(1, activation='linear'))

         model.compile(optimizer='adam', loss='mse', metrics=['mae'])

         return model

     model = build_model(num_input=13)
     model.summary()
```

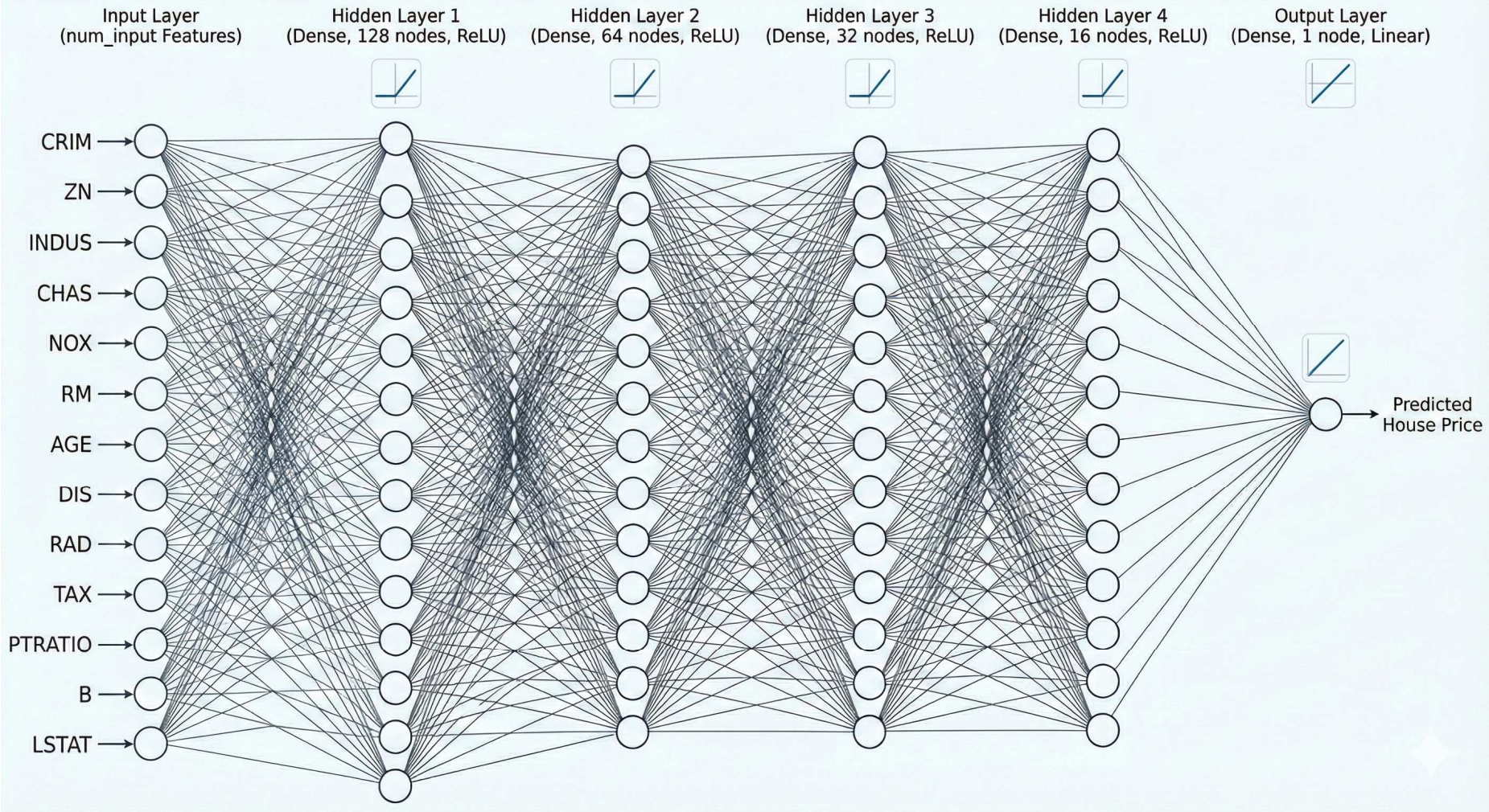
딥러닝

Model: "sequential"

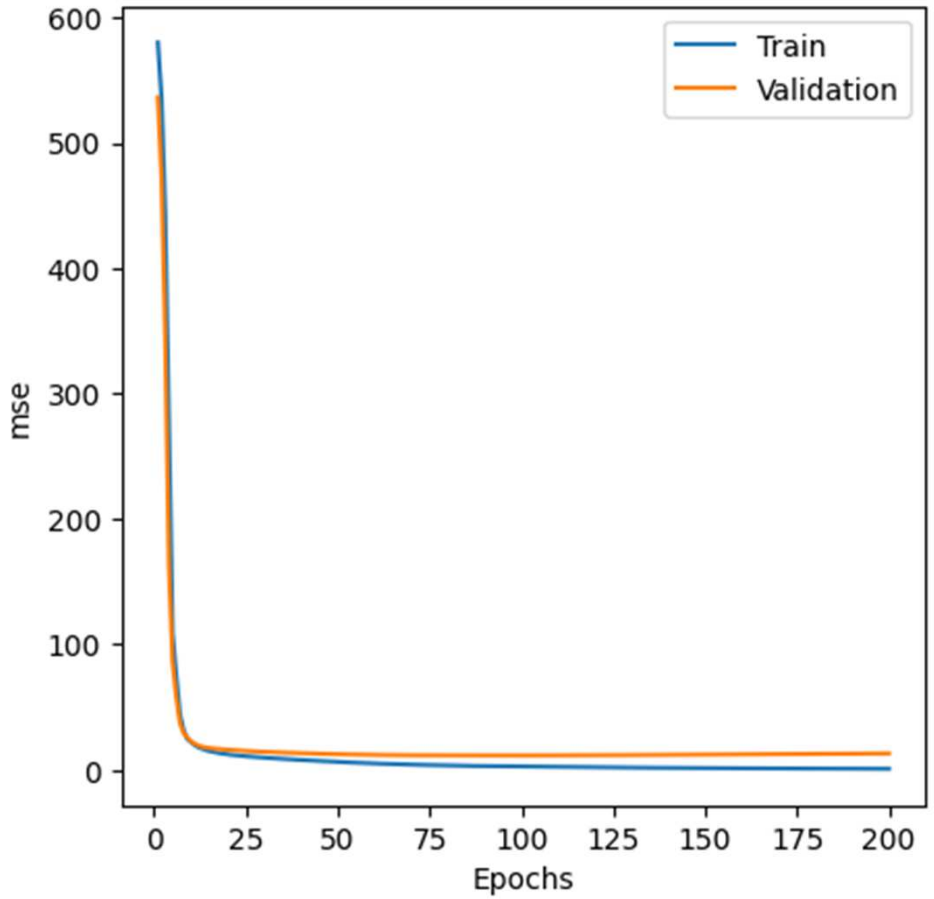
Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	1,792
dense_1 (Dense)	(None, 64)	8,256
dense_2 (Dense)	(None, 32)	2,080
dense_3 (Dense)	(None, 16)	528
dense_4 (Dense)	(None, 1)	17

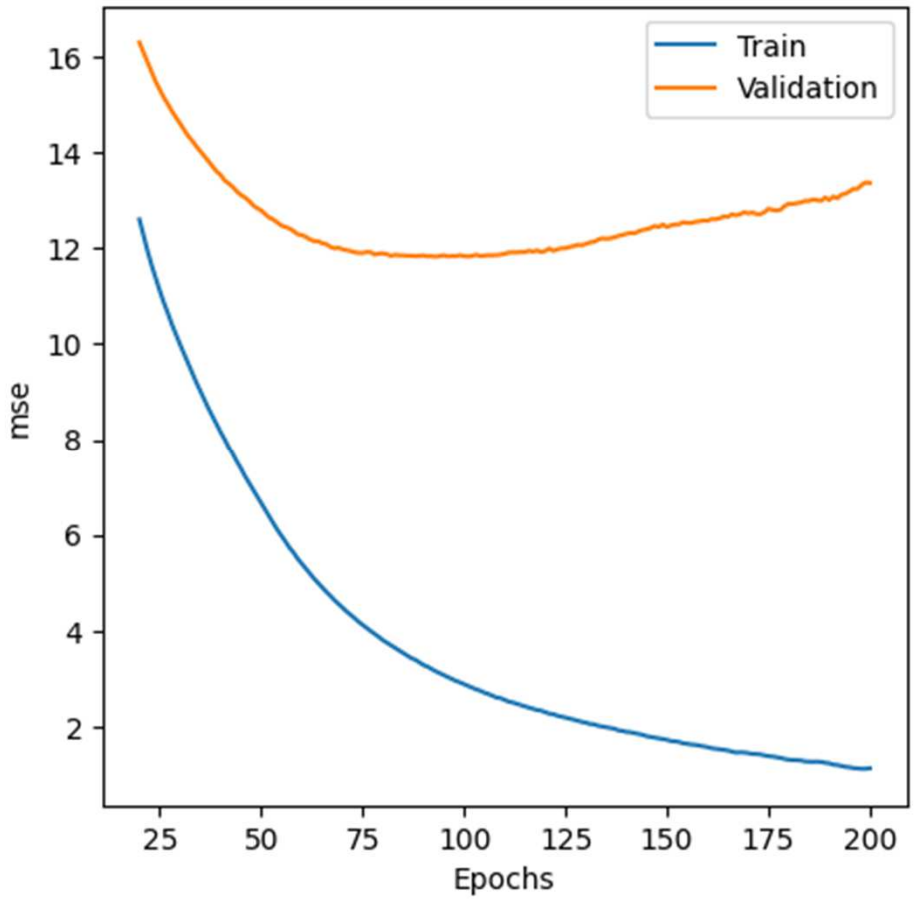
Total params: 12,673 (49.50 KB)
Trainable params: 12,673 (49.50 KB)
Non-trainable params: 0 (0.00 B)

딥러닝



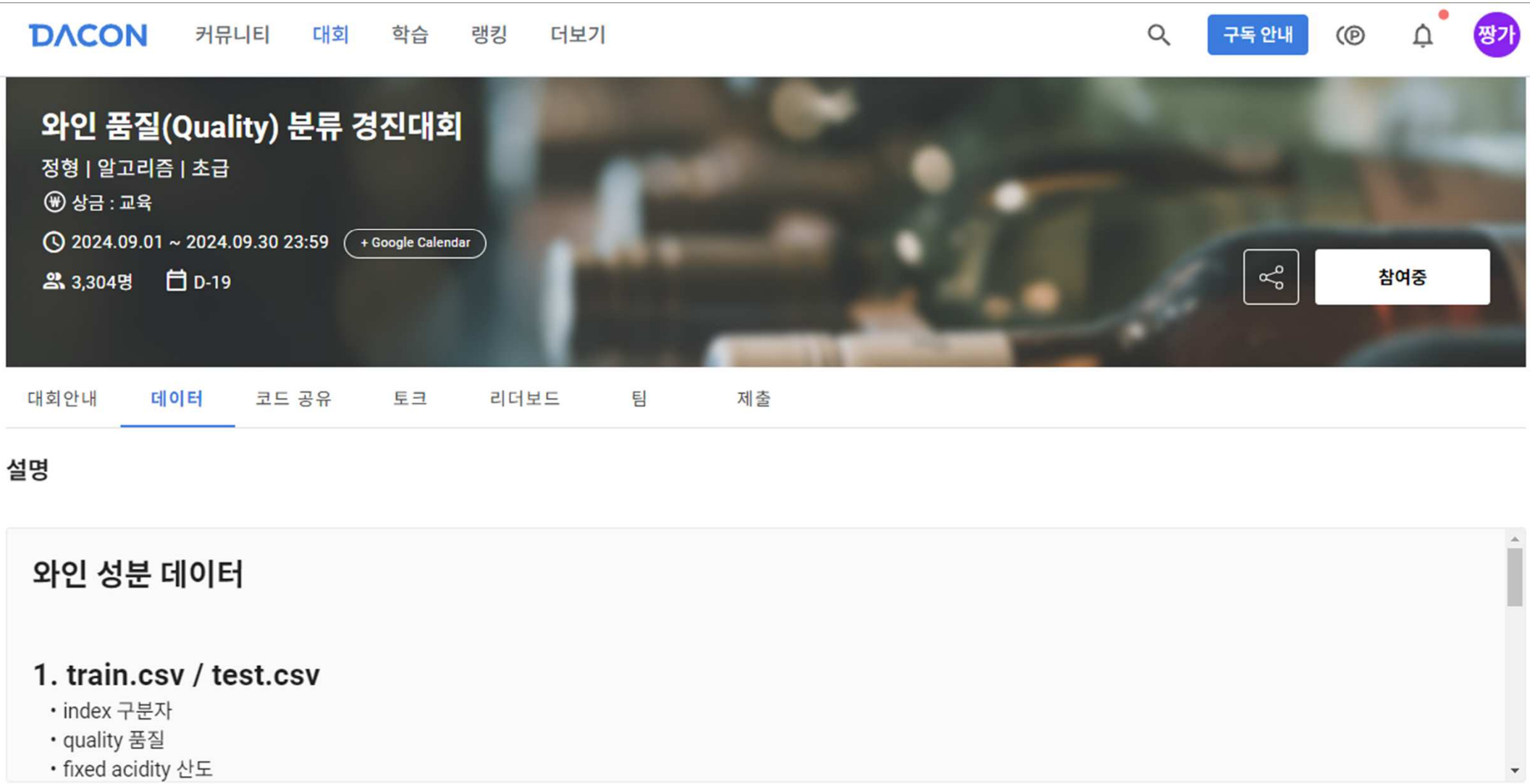
딥러닝





와인분류, DL 버전

딥러닝



딥러닝

train.csv / test.csv

•index 구분자
•quality 품질
•fixed acidity 산도
•volatile acidity 휘발성산
•citric acid 시트르산
•residual sugar 잔당 : 발효 후 와인 속에 남아있는 당분
•chlorides 염화물
•free sulfur dioxide 독립 이산화황
•total sulfur dioxide 총 이산화황
•density 밀도
•pH 수소이온농도
•sulphates 황산염
•alcohol 도수
•type 종류

※ 데이터 출처 : https://archive.ics.uci.edu/ml/datasets/Wine+Quality

데이터 전처리

[3]
```
# 데이콘 사이트에서 다운로드한 CSV파일을 읽어오기
drive_path = "/gdrive/My Drive/"

train = pd.read_csv(drive_path + "01/wine/data/train.csv")
test = pd.read_csv(drive_path + "01/wine/data/test.csv")
submission = pd.read_csv(drive_path + "01/wine/data/sample_submission.csv")

print(train.shape, test.shape, submission.shape)
```

```
(5497, 14) (1000, 13) (1000, 2)
```

[4]
```
train.head(2)
```

	index	quality	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides
0	0	5	5.6	0.695	0.06	6.8	0.042
1	1	5	8.8	0.610	0.14	2.4	0.067

딥러닝

0초
```
from tensorflow.keras.utils import to_categorical

y_train = to_categorical(train.loc[:, 'quality'] - 3)
y_train
```

```
array([[0., 0., 1., ..., 0., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

1초
```
[10] # 피처 선택
X_train = train.loc[:, 'fixed acidity':]
X_test = test.loc[:, 'fixed acidity':]

# 피처 스케일링
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
scaler.fit(X_train)
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)

print(X_train_scaled.shape, y_train.shape)
print(X_test_scaled.shape)
```

```
(5497, 12) (5497, 7)
(1000, 12)
```

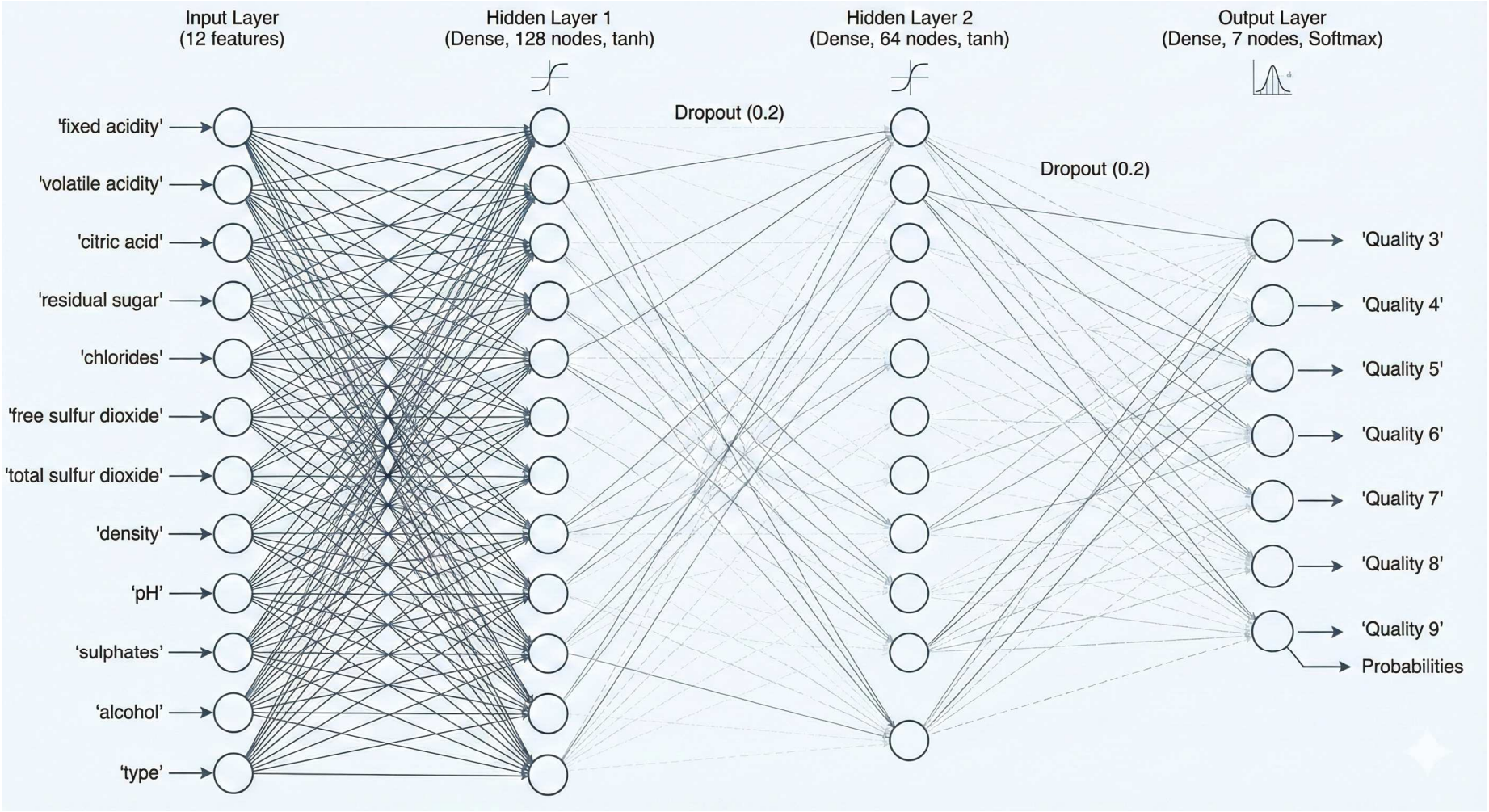
딥러닝

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	1,664
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 32)	2,080
dense_3 (Dense)	(None, 7)	231

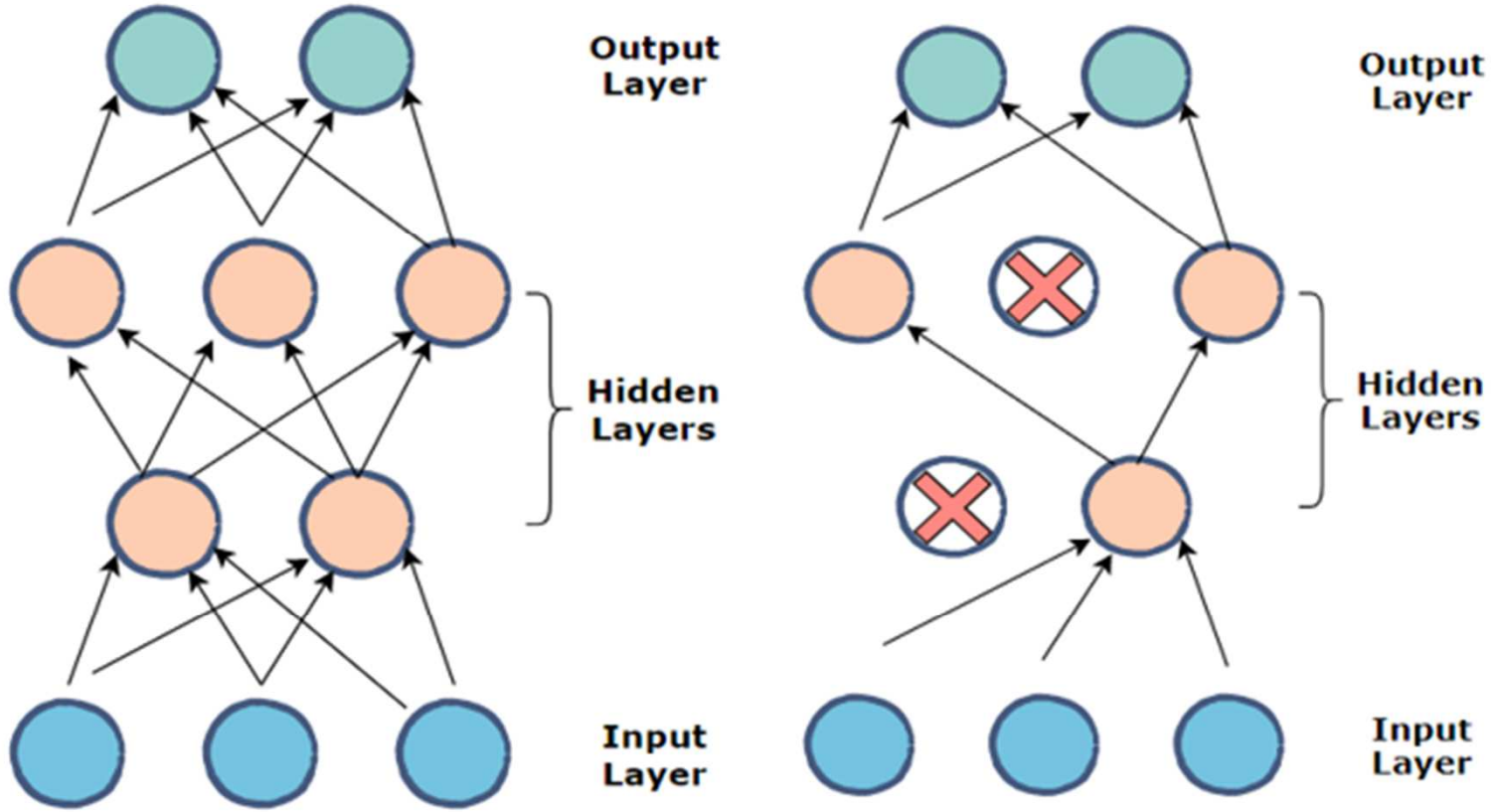
Total params: 12,231 (47.78 KB)
Trainable params: 12,231 (47.78 KB)
Non-trainable params: 0 (0.00 B)

딥러닝



딥러닝

dropout : 과적합을 방지하기위해서 노드를 랜덤하게 off



Original network

drop out networks

```
# test 데이터에 대한 예측값 정리
y_pred_proba = model.predict(X_test_scaled)
y_pred_proba[:5]
```

```
32/32 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
array([[1.9972272e-04, 5.5295196e-03, 1.8350047e-01, 5.0586289e-01,
        2.6980424e-01, 3.5021175e-02, 8.1995160e-05],
       [5.2239760e-03, 5.2661773e-02, 6.7706573e-01, 2.4830759e-01,
        1.4321792e-02, 2.4157367e-03, 3.4108484e-06],
       [2.1068929e-03, 1.4283603e-02, 6.2976193e-01, 3.4011158e-01,
        1.1041567e-02, 2.6776546e-03, 1.6713402e-05],
       [1.4349334e-03, 3.9099649e-02, 3.9881280e-01, 4.5246890e-01,
        9.6432358e-02, 1.1719960e-02, 3.1412339e-05],
       [3.3740522e-04, 2.0633303e-03, 3.4092590e-02, 2.8969890e-01,
        5.7847834e-01, 9.4987348e-02, 3.4208802e-04]], dtype=float32)
```

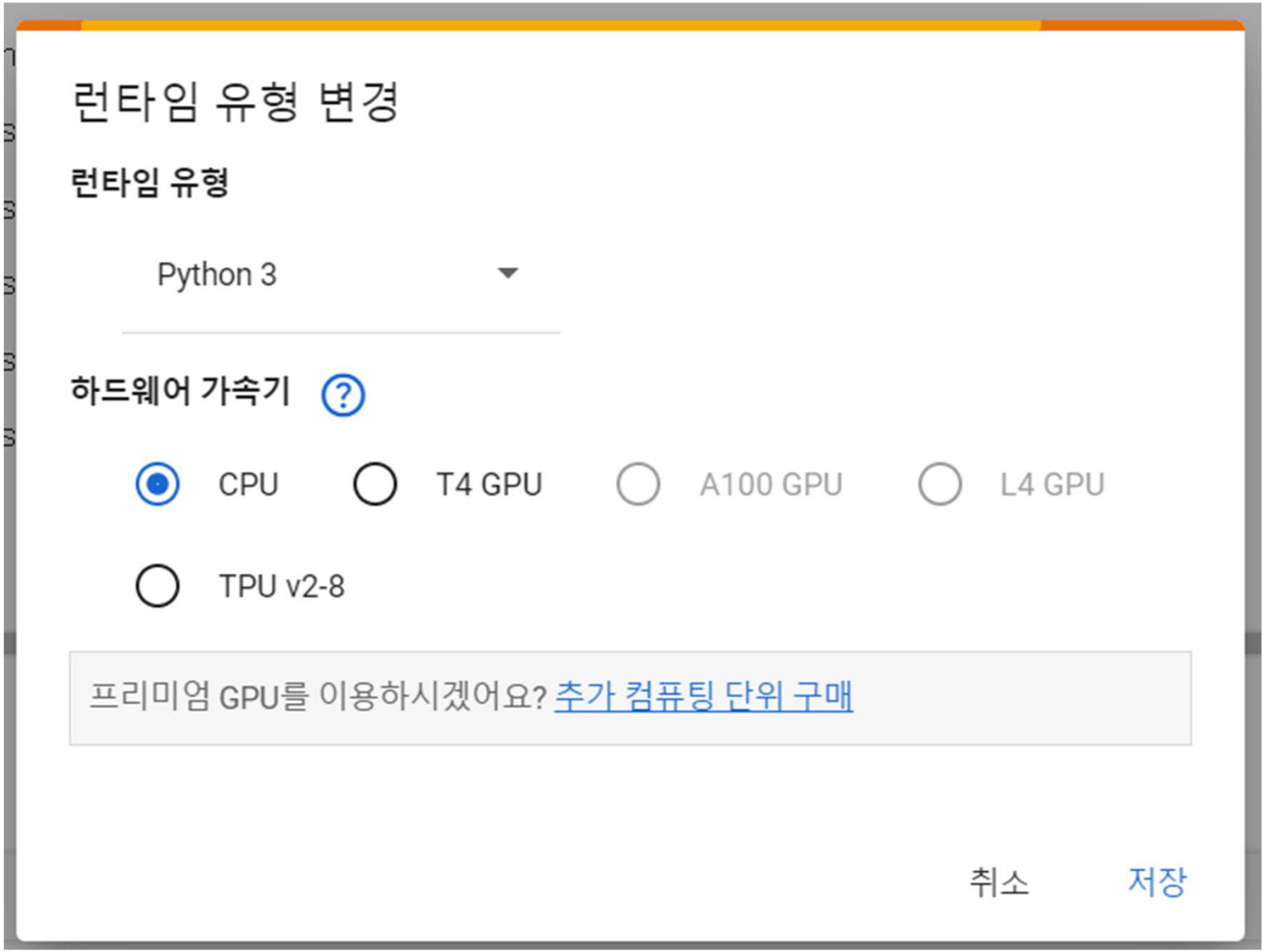
```
[15] y_pred_label = np.argmax(y_pred_proba, axis=-1) + 3
     y_pred_label[:5]
```

```
array([6, 5, 5, 6, 7])
```

손글씨 숫자인식 데이터

예제 : RP16MNIST

딥러닝

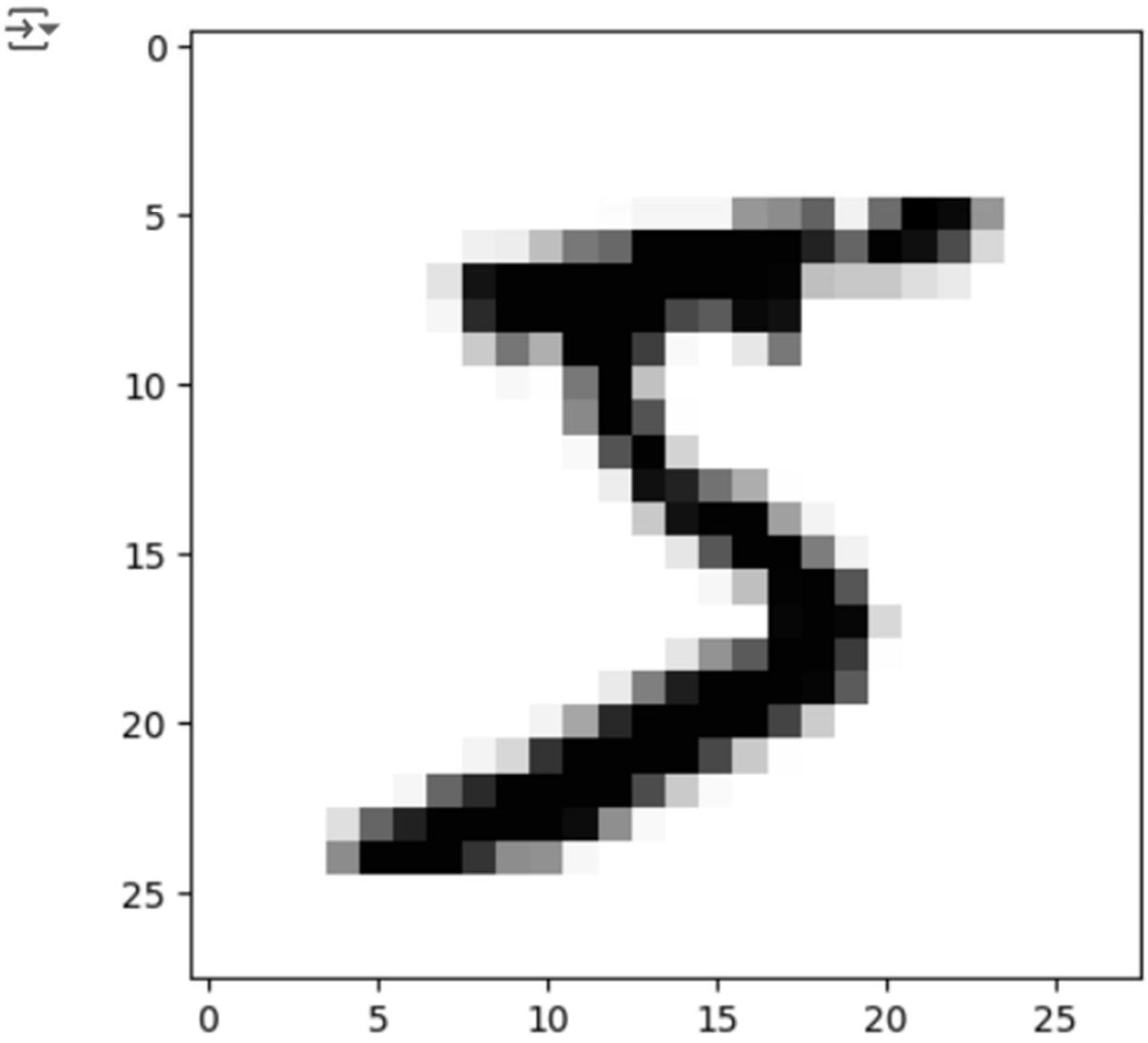


딥러닝

```
[1]  from tensorflow.keras.datasets import mnist
     from tensorflow.keras.utils import to_categorical

     import matplotlib.pyplot as plt
     import sys

     # MNIST 데이터셋을 불러와 학습셋과 테스트셋으로 저장합니다.
     (X_train, y_train), (X_test, y_test) = mnist.load_data()

     # 학습셋과 테스트셋이 각각 몇 개의 이미지로 되어 있는지 확인합니다.
     print("학습셋 이미지 수 : %d 개" % (X_train.shape[0]))
     print("테스트셋 이미지 수 : %d 개" % (X_test.shape[0]))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
학습셋 이미지 수 : 60000 개
테스트셋 이미지 수 : 10000 개
```

```
# 첫 번째 이미지를 확인해 봅시다.
plt.imshow(X_train[0], cmap='Greys')
plt.show()
```



딥러닝

```
# 이미지가 인식되는 원리를 알아봅시다.
for x in X_train[0]:
    for i in x:
        sys.stdout.write("%-3s" % i)
    sys.stdout.write('₩n')
```

```
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   3   18  18  18  126   136   175   26  166   255   247   127   0   0   0   0
0   0   0   0   0   0   0   0   30  36  94  154 170   253   253   253   253   253   225   172   253   242   195   64  0   0   0   0
0   0   0   0   0   0   0   49  238   253   253   253   253   253   253   253   253   251   93  82  82  56  39  0   0   0   0   0
0   0   0   0   0   0   0   18  219   253   253   253   253   253   198   182   247   241   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   80  156   107   253   253   205   11  0   43  154 0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   14  1   154 253 90  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   139 253 190 2   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   11  190 253 70  0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   35  241 225 160 108 1   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   81  240 253 253 119 25  0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   45  186 253 253 150 27  0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   16  93  252 253 187 0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   249 253 249 64  0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   46  130 183 253 253 207 2   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   39  148 229 253 253 253 250 182 0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   24  114 221 253 253 253 253 201 78  0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   23  66  213 253 253 253 253 198 81  2   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   18  171 219 253 253 253 253 195 80  9   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   55  172 226 253 253 253 253 244 133 11  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   136 253 253 253 212 135 132 16  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

딥러닝

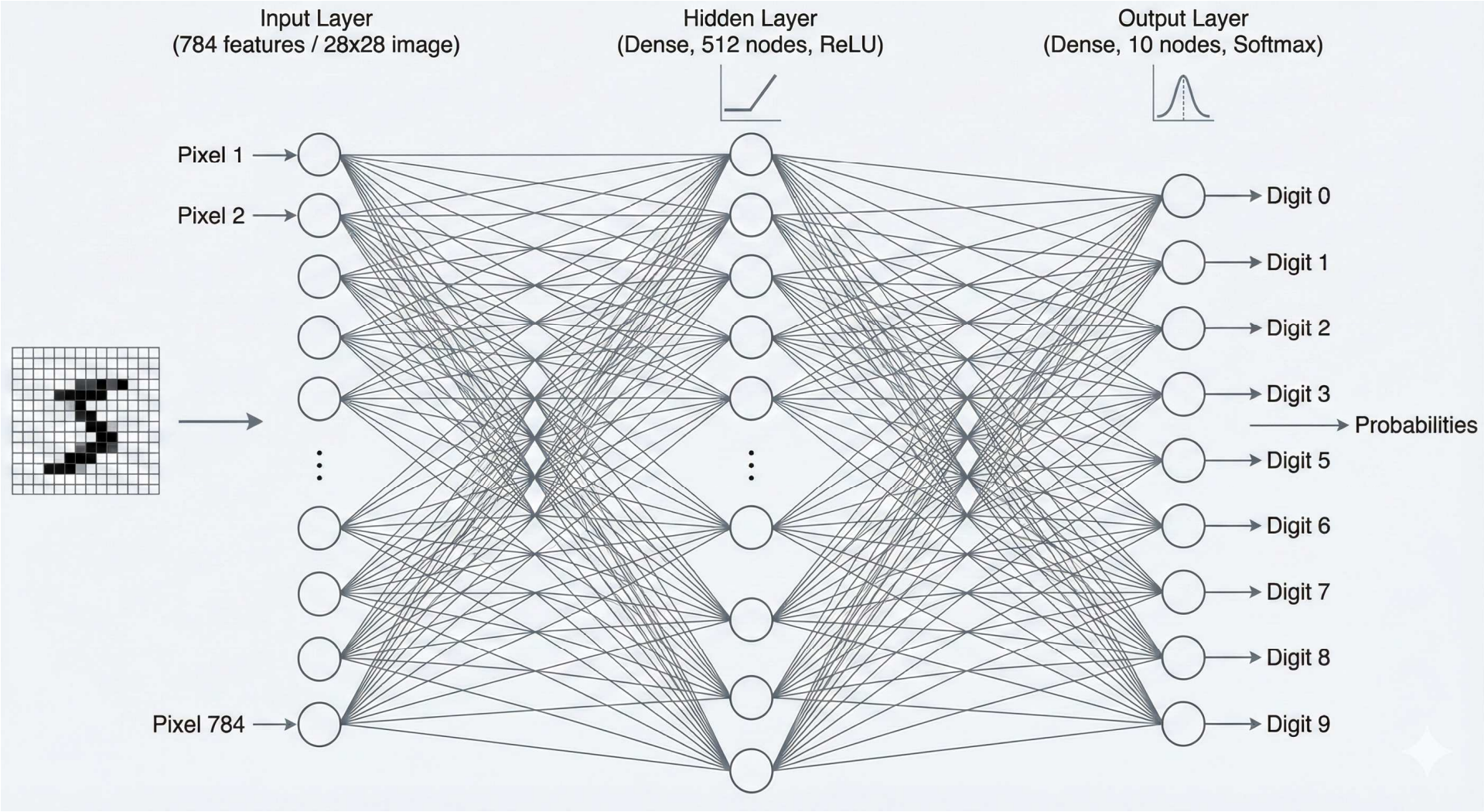
```
# 모델 구조를 설정합니다.
model = Sequential()
model.add(Dense(512, input_dim=784, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.summary()
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 512)	401,920
dense_1 (Dense)	(None, 10)	5,130

```
Total params: 407,050 (1.55 MB)
Trainable params: 407,050 (1.55 MB)
Non-trainable params: 0 (0.00 B)
```

딥러닝



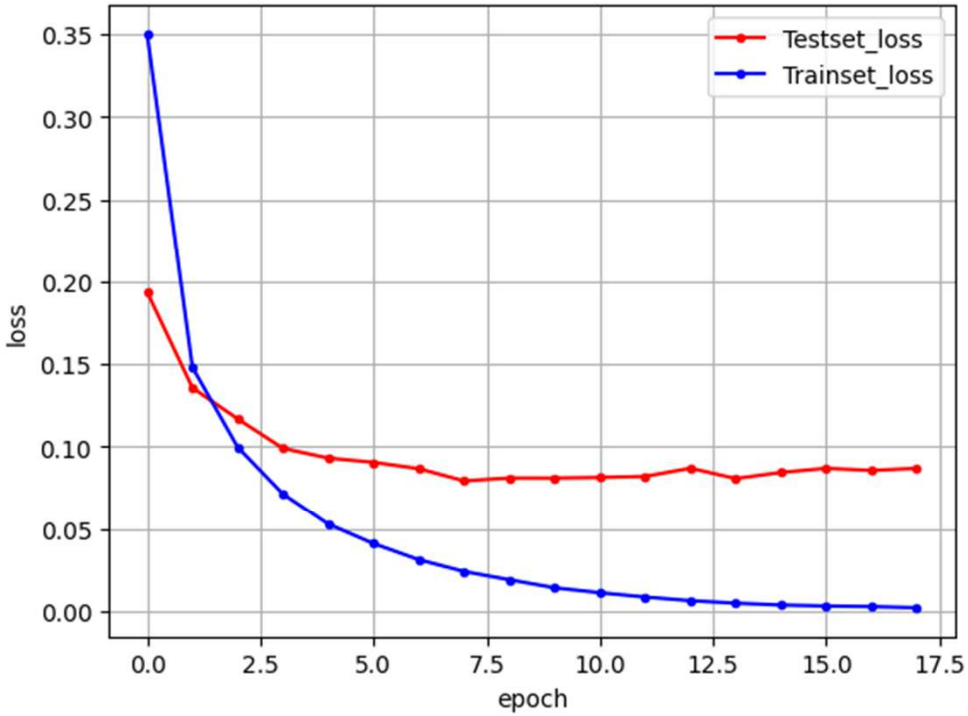
딥러닝

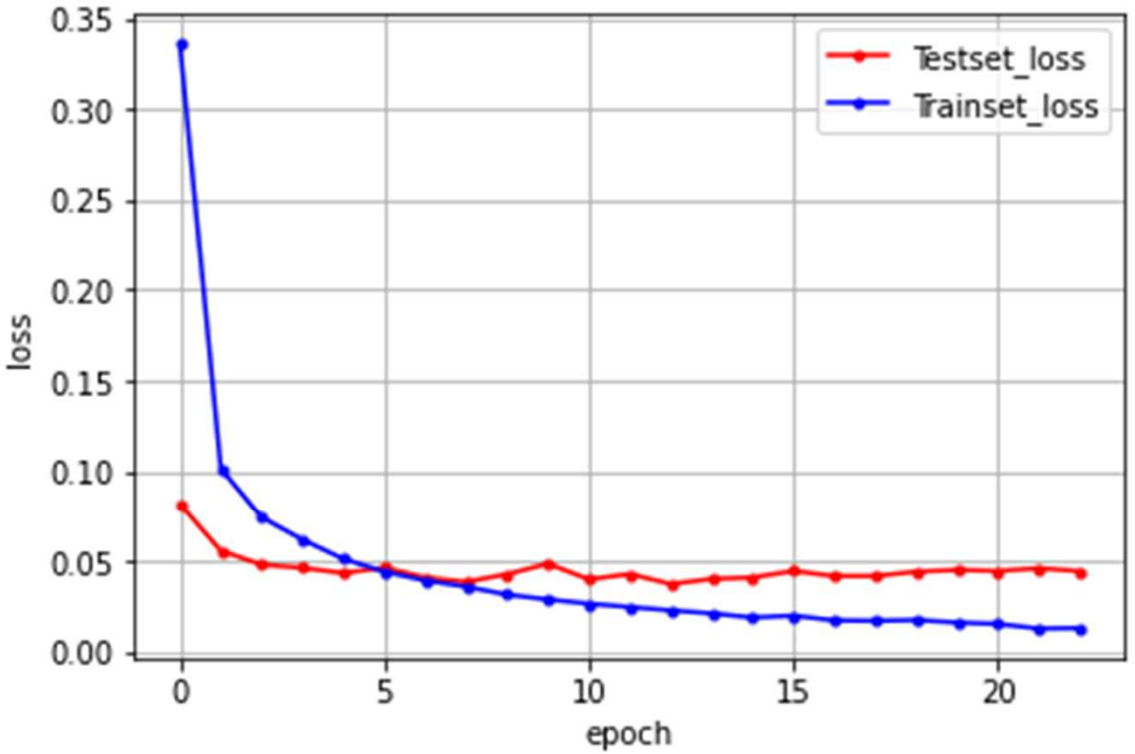
```
Epoch 7: val_loss improved from 0.09053 to 0.08671, saving model to ./MNIST_MLP.keras

Epoch 8: val_loss improved from 0.08671 to 0.07932, saving model to ./MNIST_MLP.keras

Epoch 9: val_loss did not improve from 0.07932

Epoch 10: val_loss did not improve from 0.07932

Epoch 11: val_loss did not improve from 0.07932

Epoch 12: val_loss did not improve from 0.07932

Epoch 13: val_loss did not improve from 0.07932

Epoch 14: val_loss did not improve from 0.07932

Epoch 15: val_loss did not improve from 0.07932

Epoch 16: val_loss did not improve from 0.07932

Epoch 17: val_loss did not improve from 0.07932

Epoch 18: val_loss did not improve from 0.07932
313/313 ━━━━━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9784 - loss: 0.0797

 Test Accuracy: 0.9813
```

딥러닝

```
# 모델 실행 환경을 설정합니다.
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# 모델 최적화를 위한 설정 구간입니다.
#modelpath="./MNIST_MLP.hdf5"
modelpath="./MNIST_MLP.keras" # Change the file extension to .keras
checkpointer = ModelCheckpoint(filepath=modelpath, monitor='val_loss', verbose=1, save_best_only=True)
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=10)

# 모델을 실행합니다.
history = model.fit(X_train, y_train, validation_split=0.25, epochs=30, batch_size=200, verbose=0, callbacks=[early_stopping_callback,checkpointer])

# 테스트 정확도를 출력합니다.
print("₩n Test Accuracy: %.4f" % (model.evaluate(X_test, y_test)[1]))
```

딥러닝

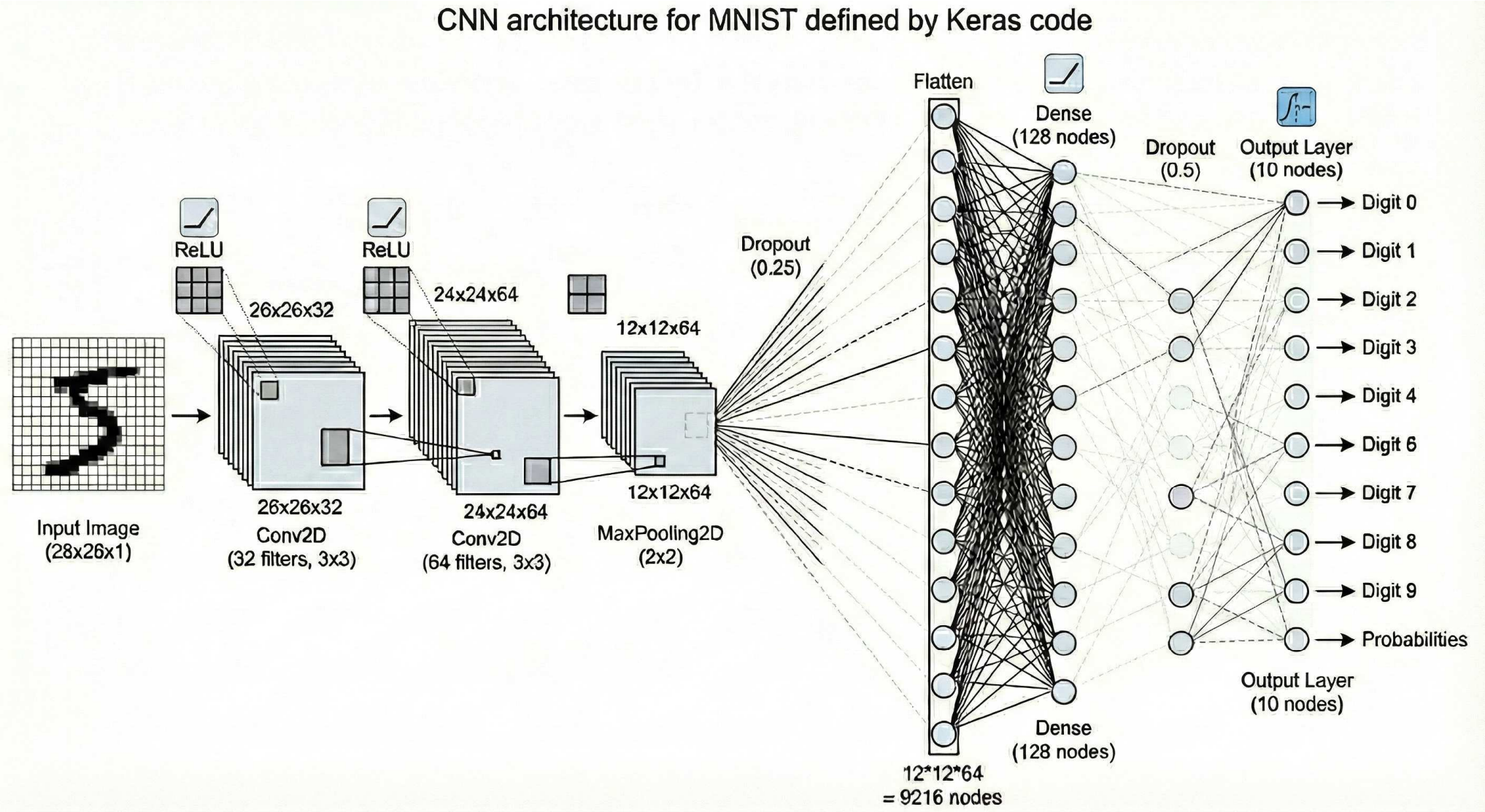




딥러닝

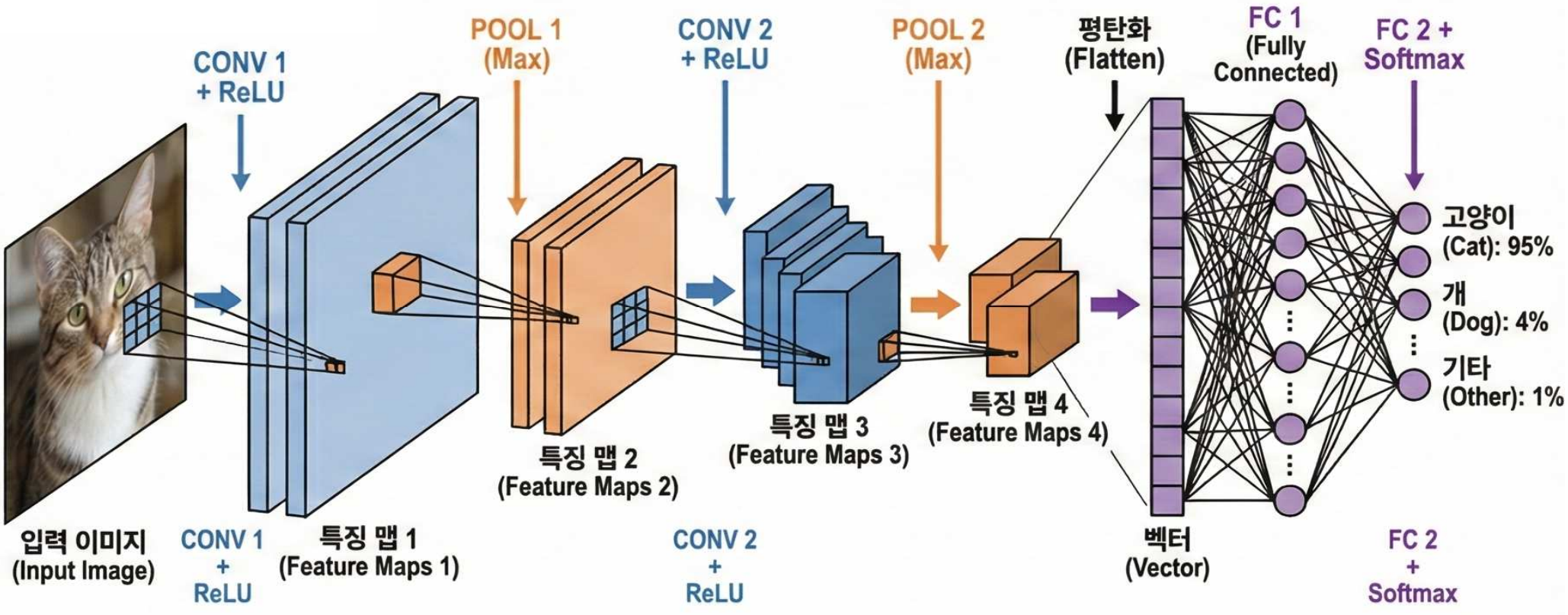
```
# 컨볼루션 신경망의 설정
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), input_shape=(28, 28, 1), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128,  activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))
```

딥러닝

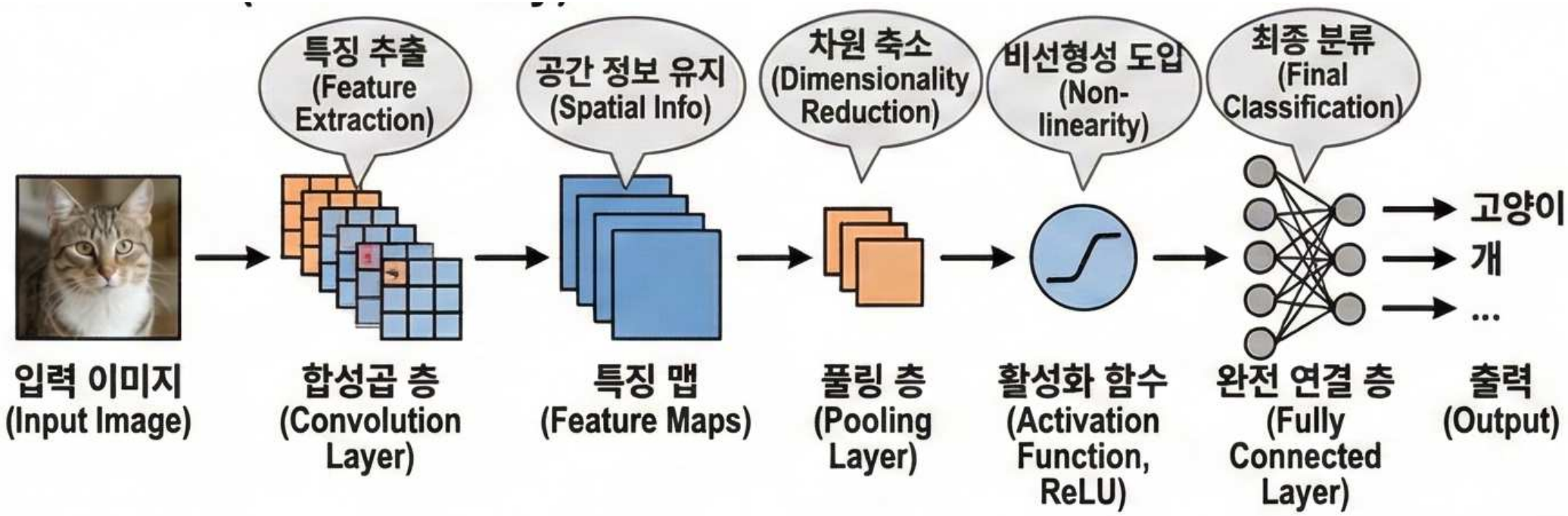


CNN : convolutional neural network

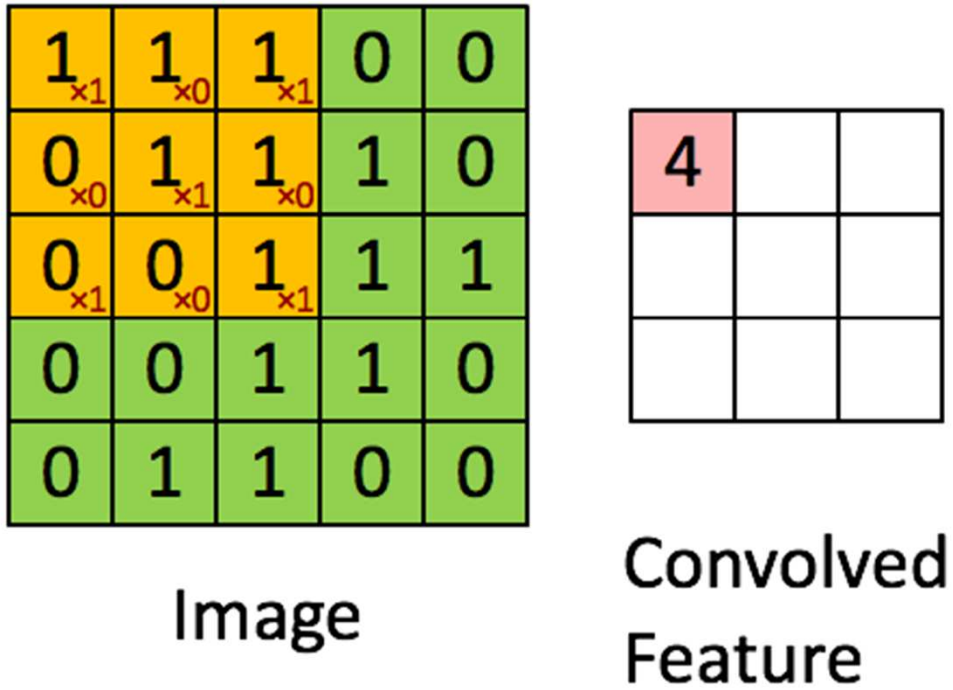
딥러닝



딥러닝

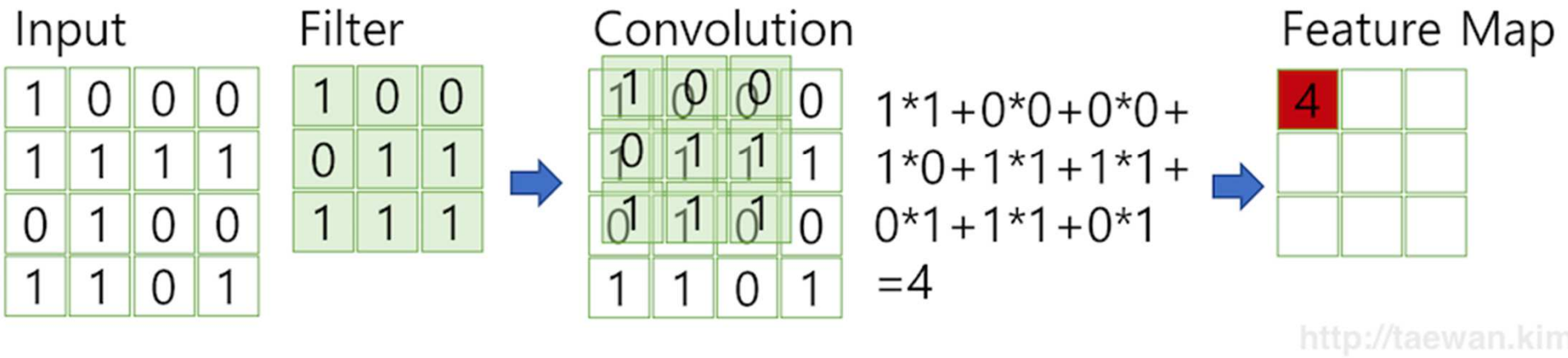


필터/커널



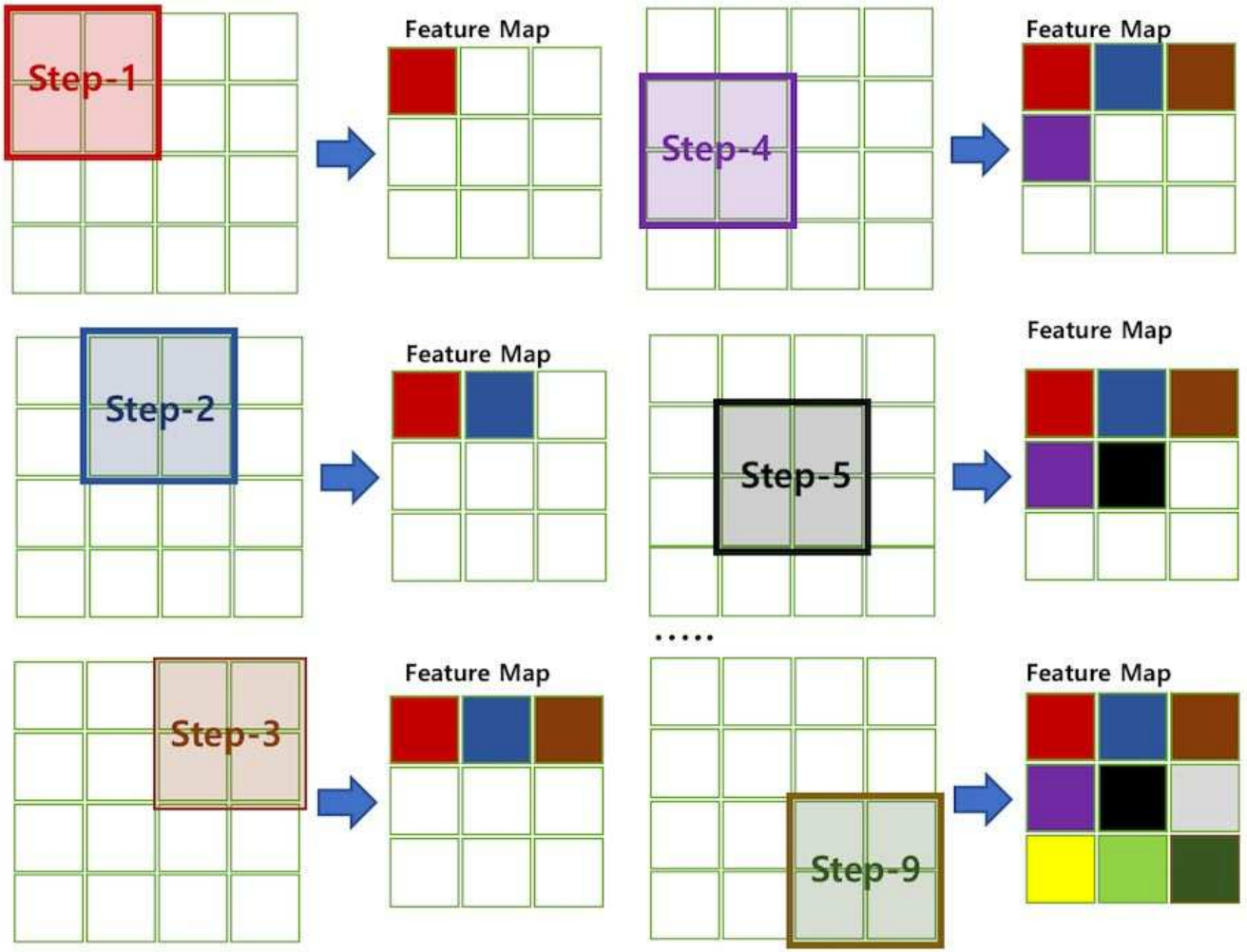
출처 : http://taewan.kim/post/cnn/

딥러닝



출처 : http://taewan.kim/post/cnn/

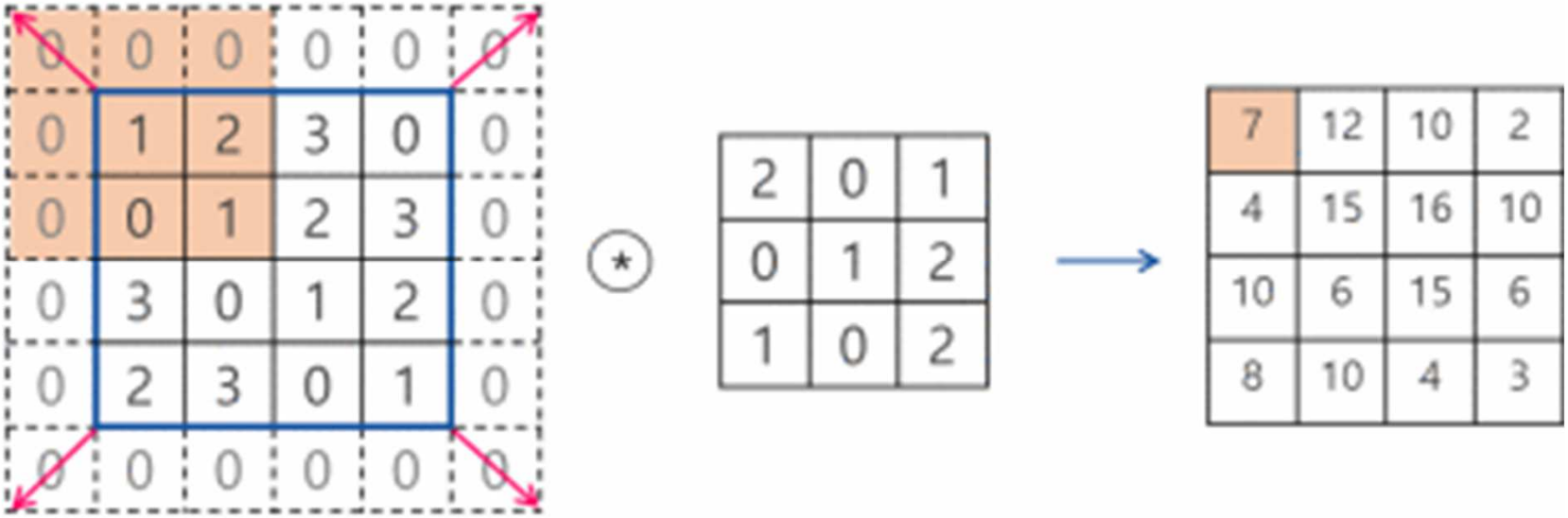
딥러닝



출처 : http://taewan.kim/post/cnn/

스트라이드

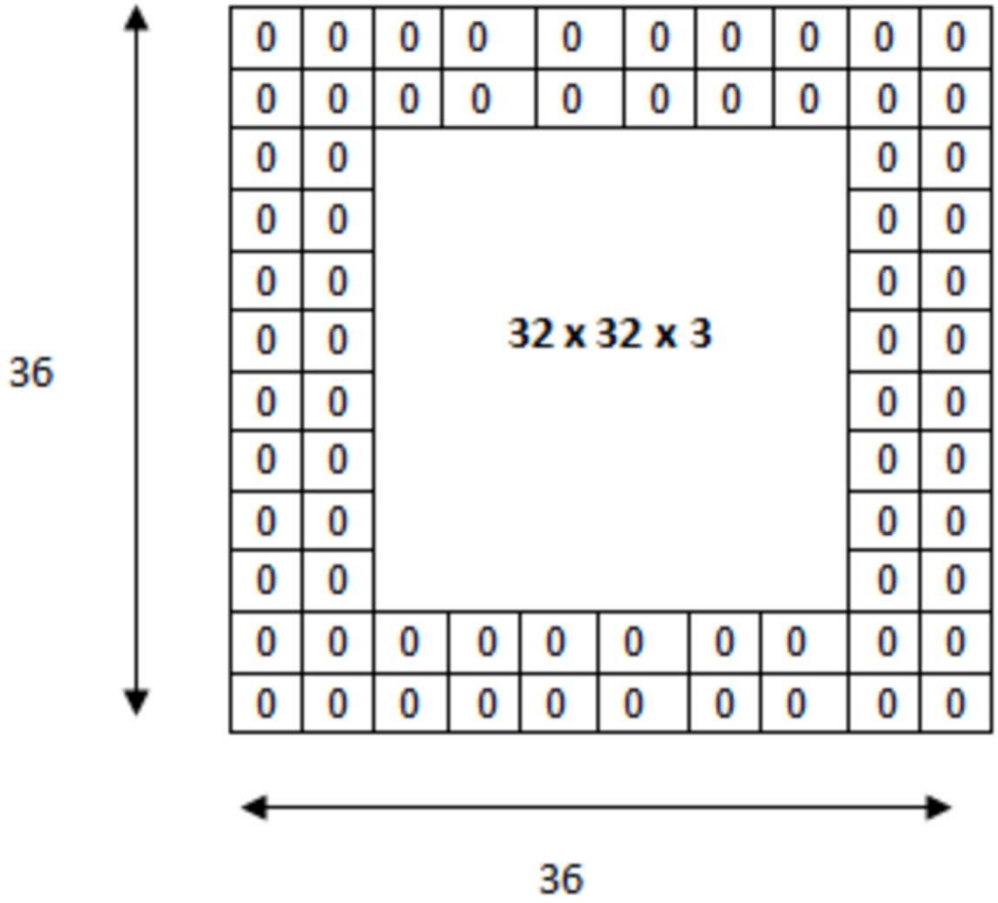
딥러닝



출처 : https://excelsior-cjh.tistory.com/79

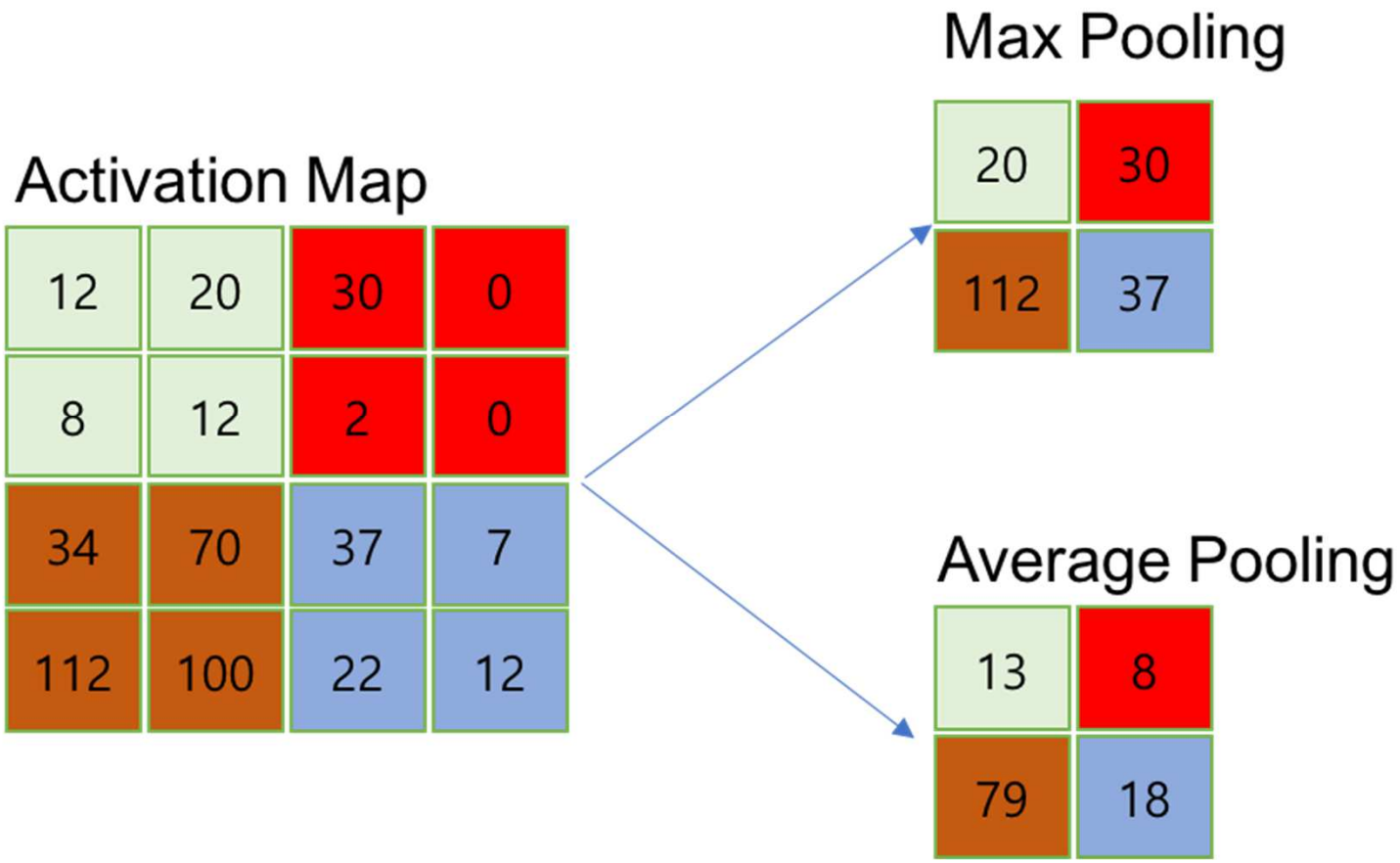
패딩

딥러닝



출처 : http://taewan.kim/post/cnn/

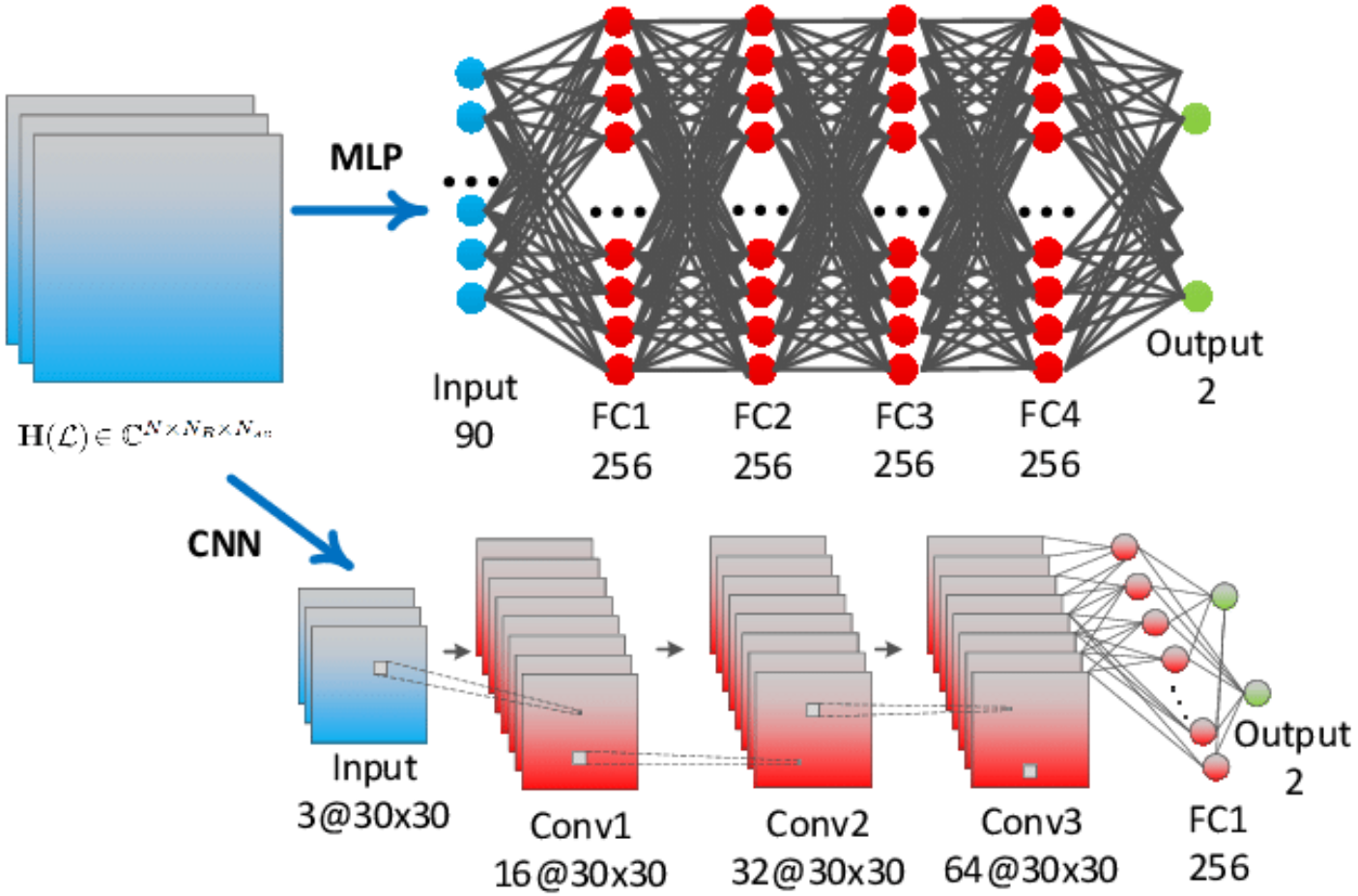
풀링



출처 : http://taewan.kim/post/cnn/

합성곱층, 완전연결계층

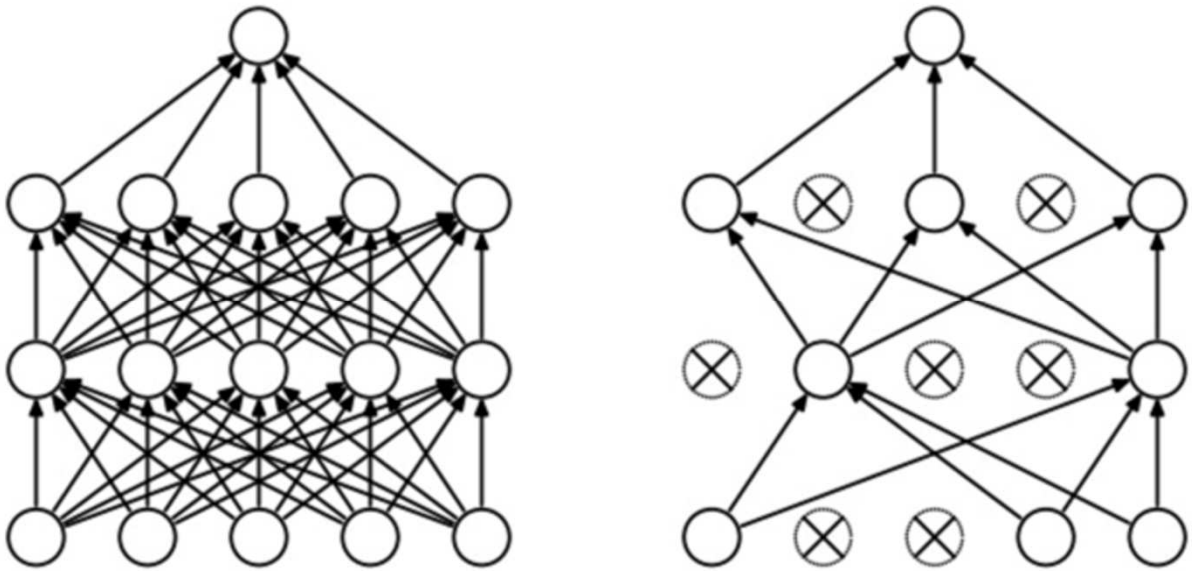
딥러닝



출처 : https://www.researchgate.net/figure/The-architecture-of-MLP-and-CNN-MLP-is-consisted-of-fullyconnected-FC-layers-and-CNN_fig2_331195277

과적합, 드롭 아웃

딥러닝

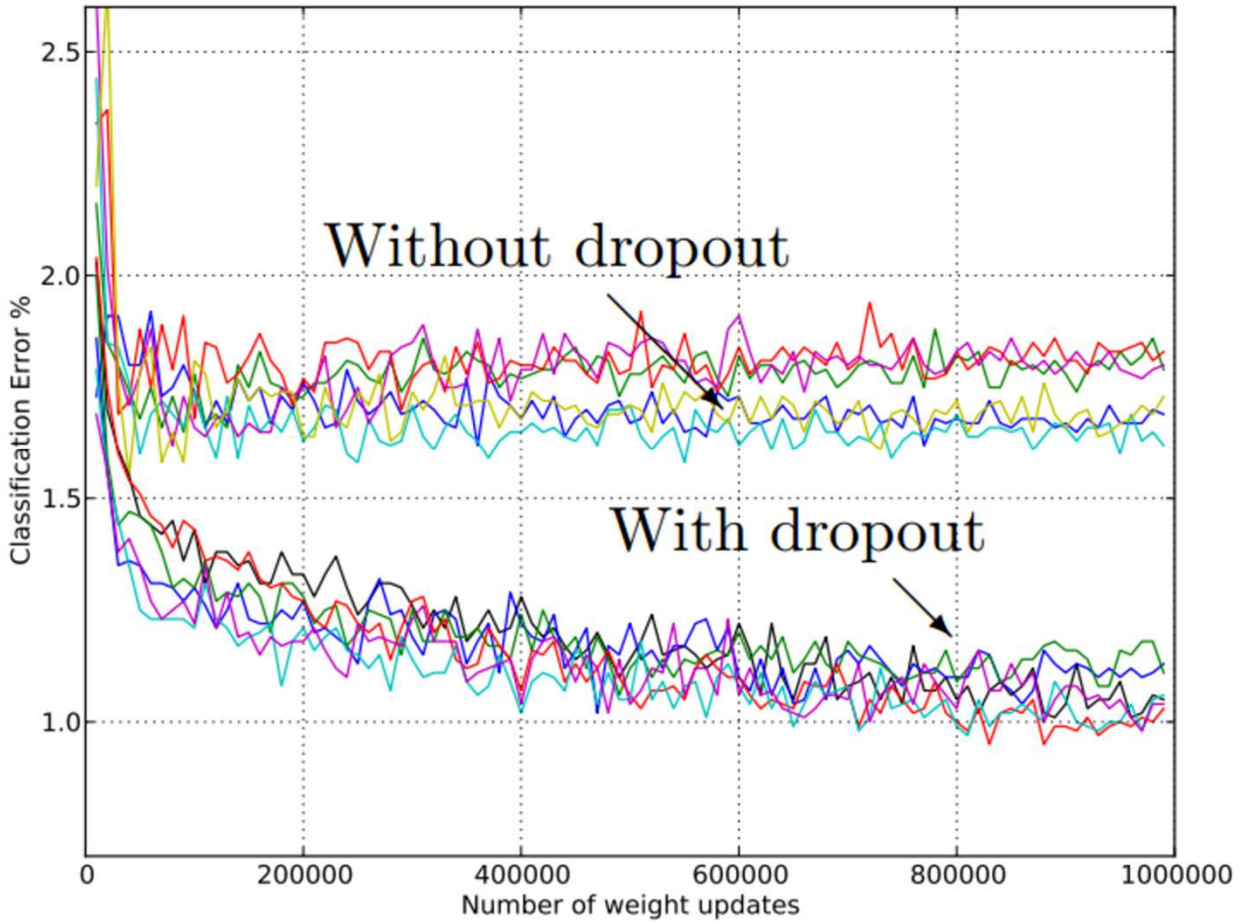


(a) Standard Neural Net

(b) After applying dropout.

출처 : https://bcho.tistory.com/tag/%EA%B0%95%EC%9D%98

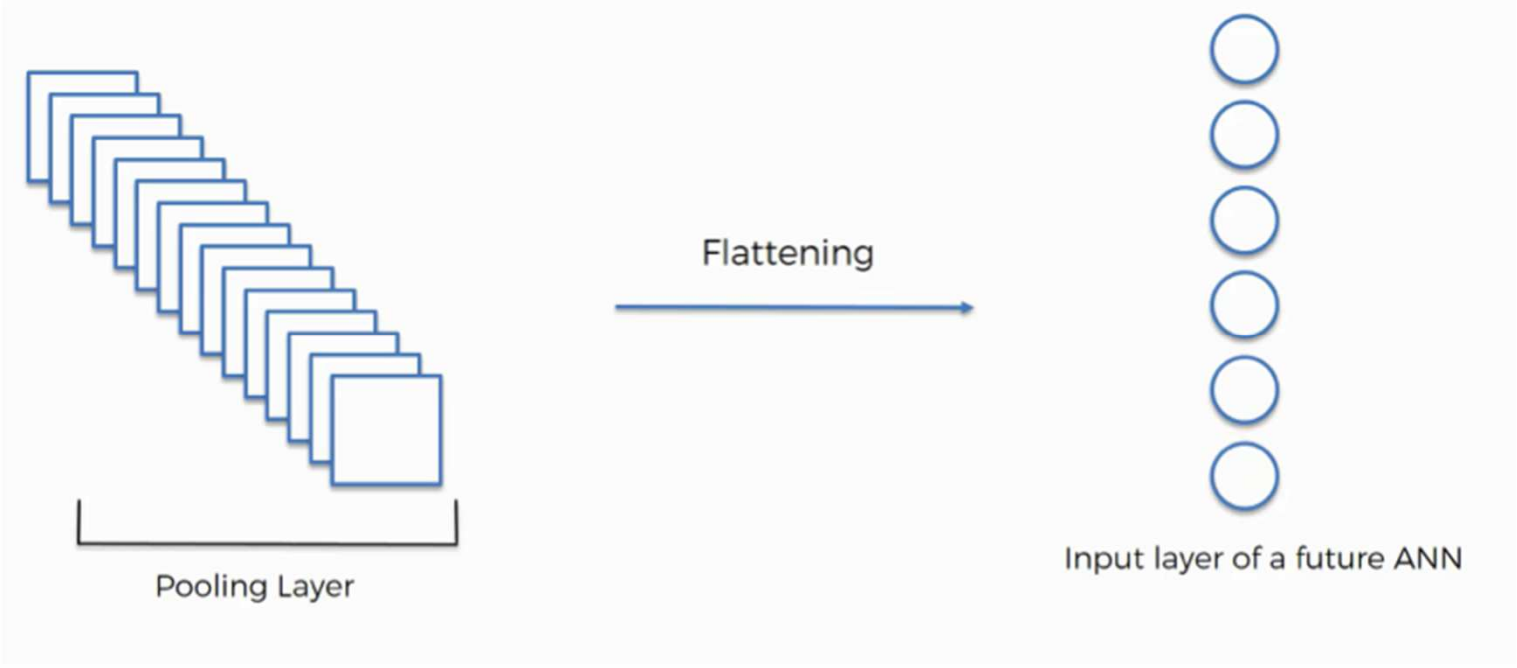
딥러닝



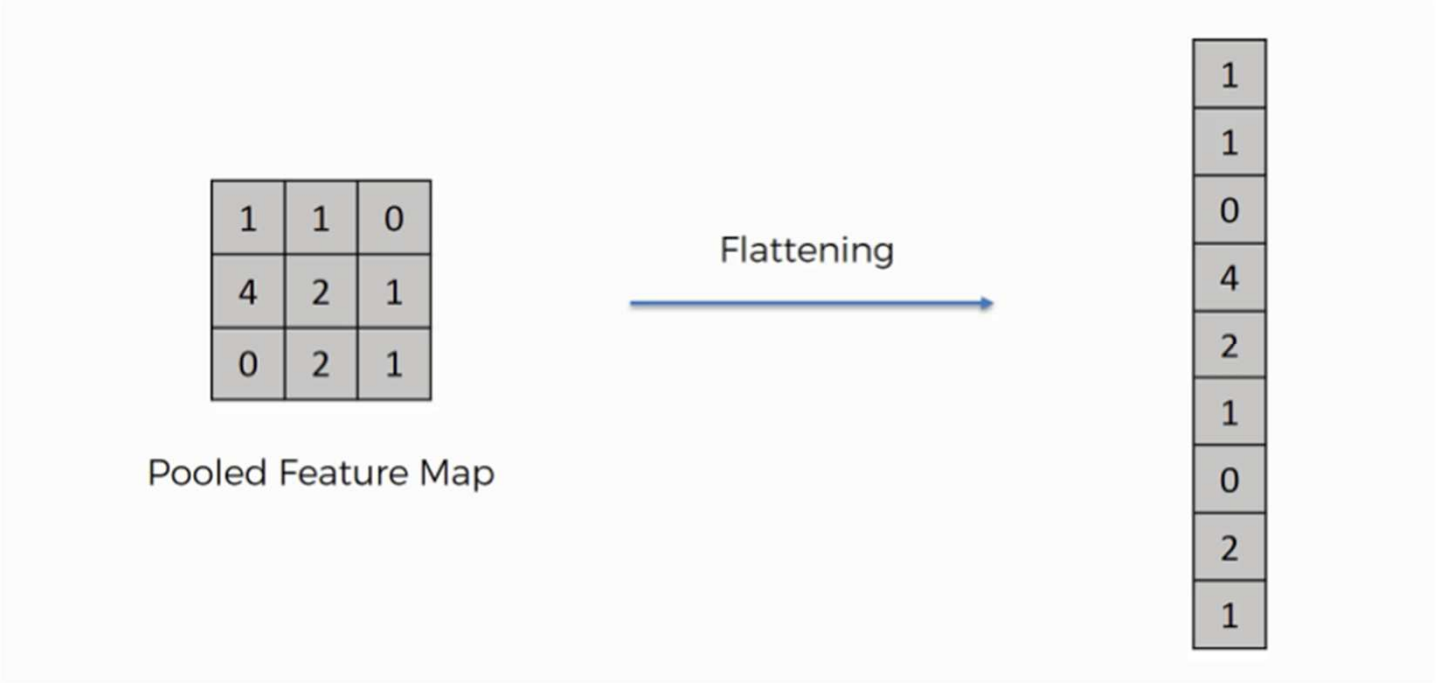
출처 : https://bcho.tistory.com/tag/%EA%B0%95%EC%9D%98

플래튼

딥러닝



출처 https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening



출처 https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening

CNN 시각화

https://tensorspace.org

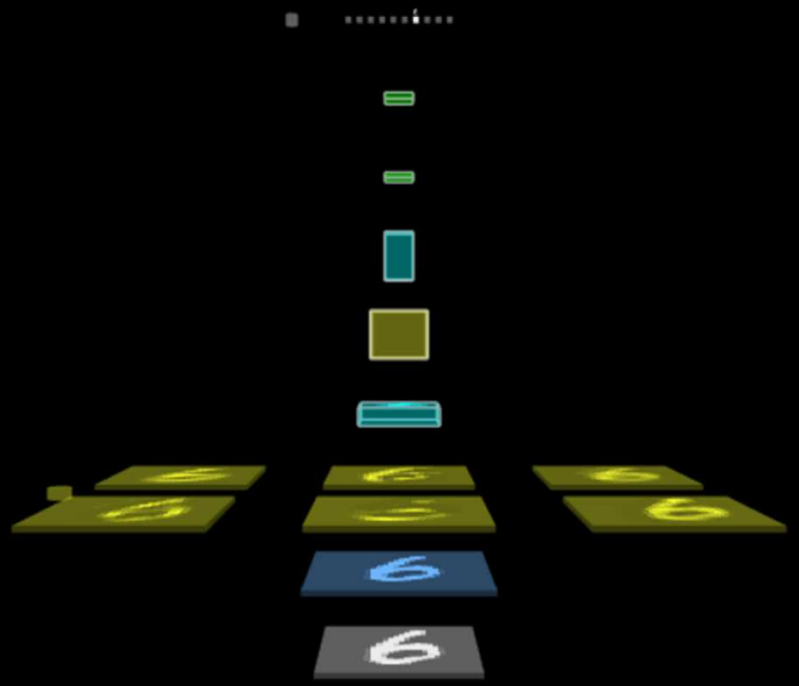
TensorSpace.js Documentation Playground Download EN 中

Neural Network

3D Visualization Framework

Build interactive and intuitive model in browsers

Get Started Playground GitHub



Interactive

Use Layer API to build interactive models in browsers

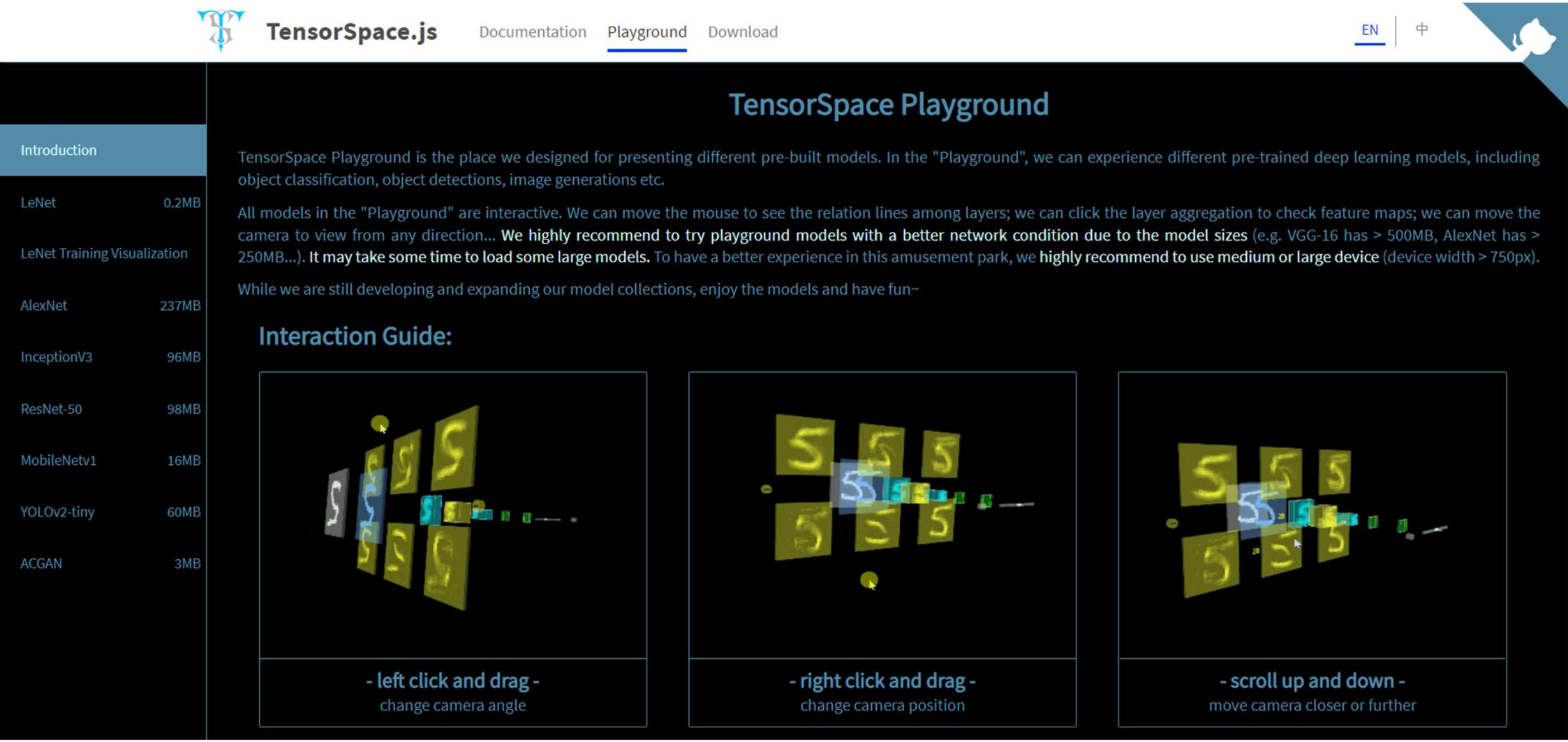
Intuitive

Visualize the information from intermediate inferences

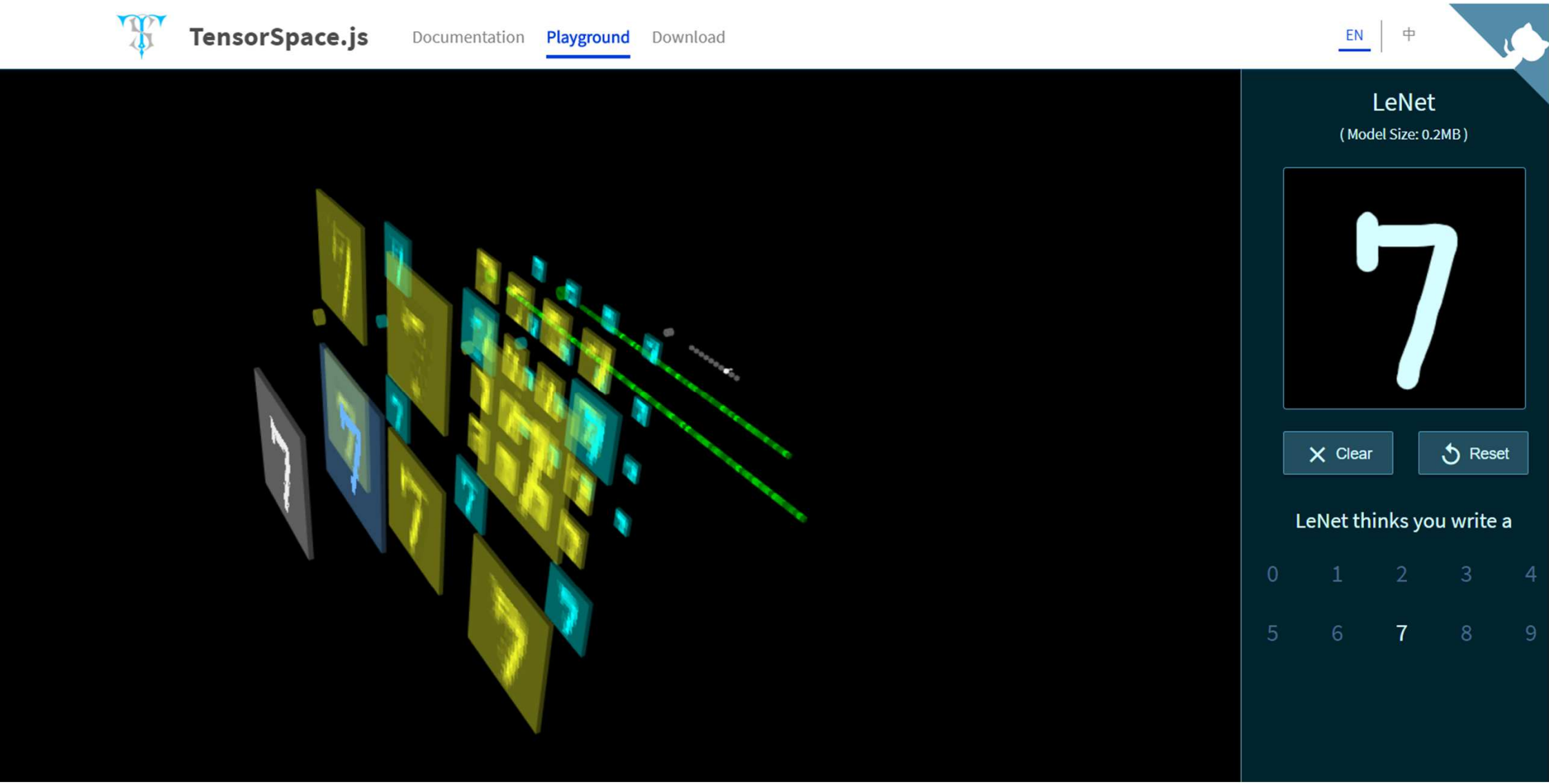
Integrative

Support pre-trained models from TensorFlow, Keras, TensorFlow.js

딥러닝

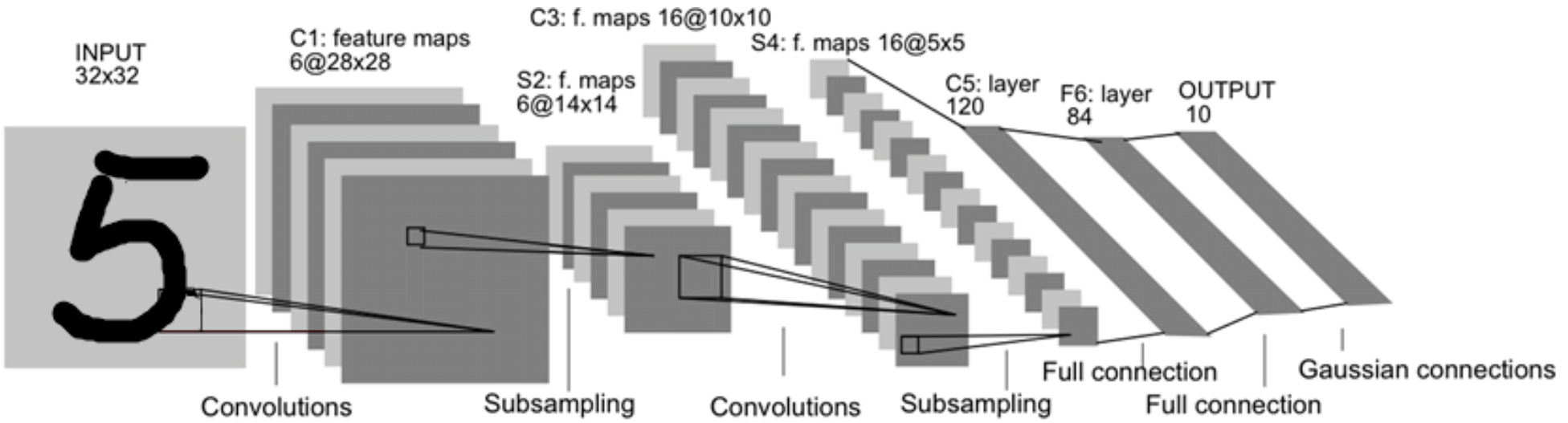


딥러닝



이미지 인식과 딥러닝

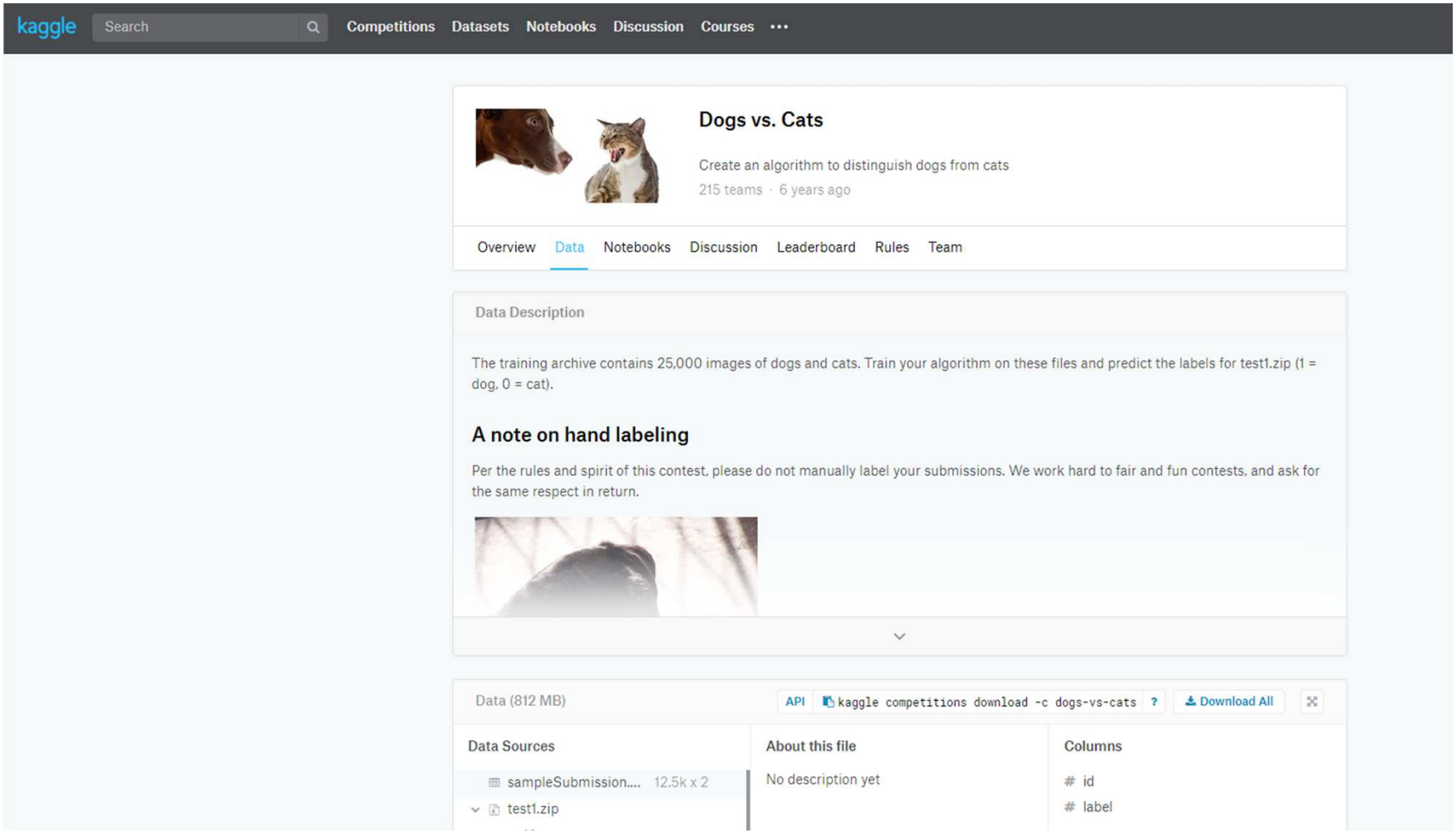
딥러닝



An early (Le-Net5) Convolutional Neural Network design, LeNet-5, used for recognition of digits

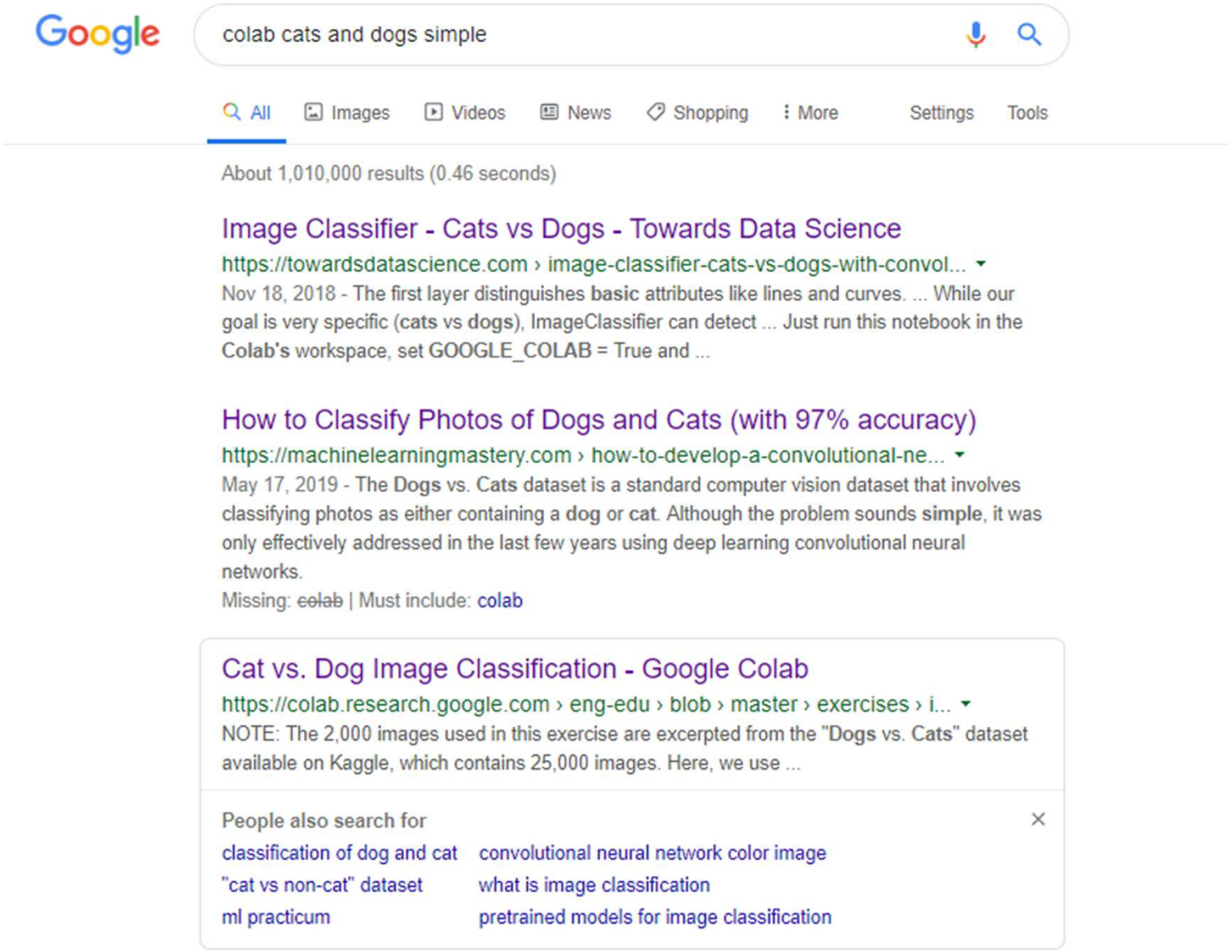
출처 https://www.researchgate.net/figure/The-architecture-of-LeNet-5-23-a-CNN-used-for-digits-recognition-for-the-MNIST-dataset_fig2_321665783

Cats and Dogs

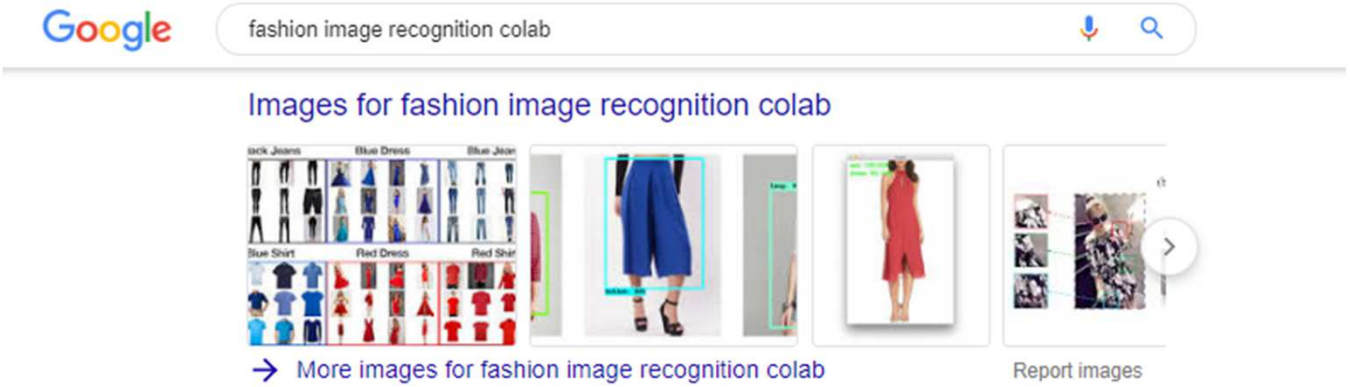


출처 : https://www.kaggle.com/c/dogs-vs-cats/data

Cats and Dogs



Fashion



Train your first neural network: basic classification - TensorFlow
https://www.tensorflow.org › tutorials › keras › basic_classification
This guide trains a neural network model to classify **images** of **clothing**, like sneakers and shirts. It's okay if you don't understand all the details, this is a ...

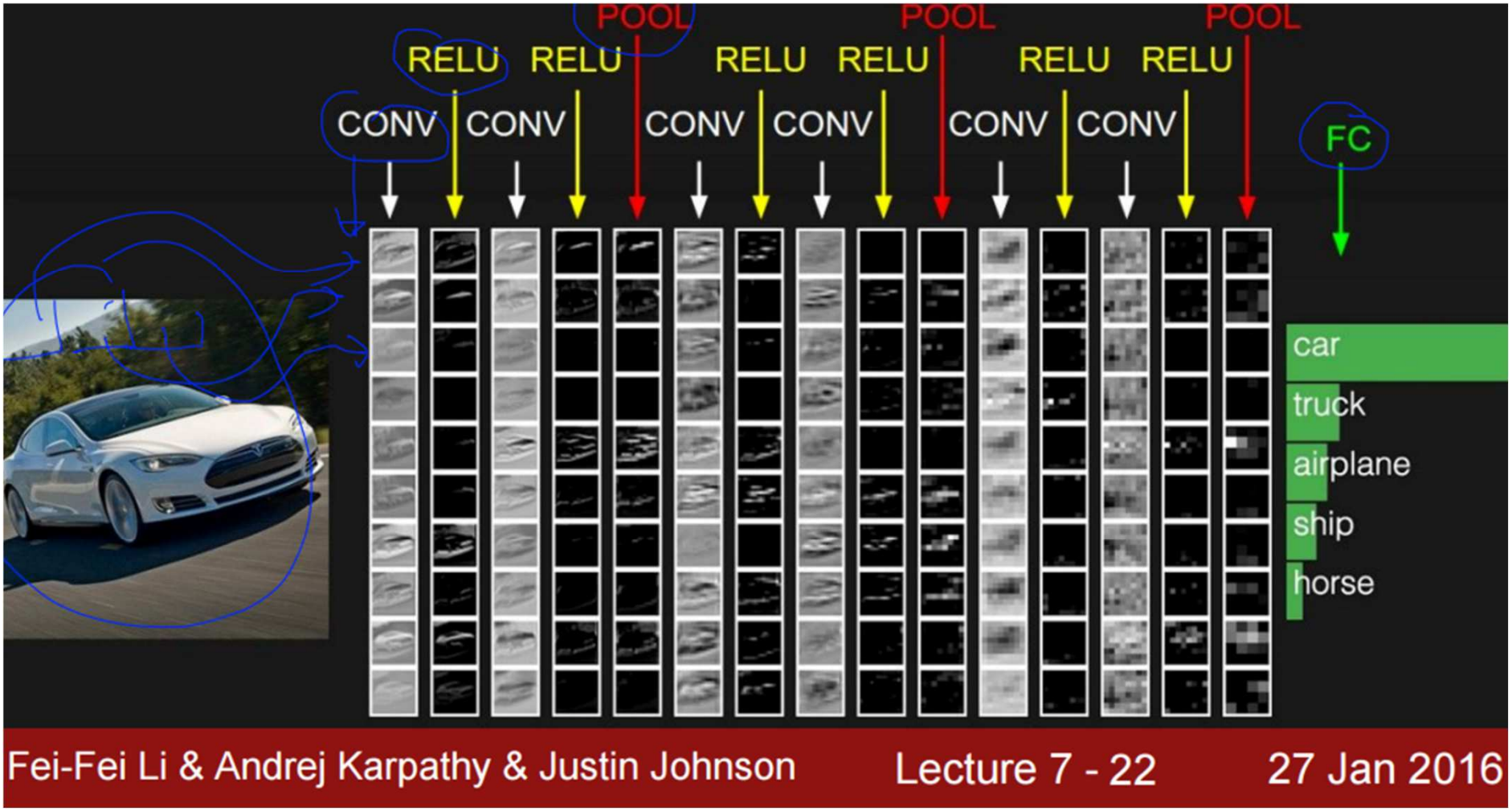
People also search for

- tensorflow binary classification
- tensorflow sentiment analysis
- tensorflow image classification github
- learn tensorflow online
- neural network classification python
- tensorflow neural network regression example

How to create a clothing classifier program using Deep Neural ...
https://becominghuman.ai › how-to-create-a-clothing-classifier-fashion-mn...
Apr 17, 2019 - The process that has occurred, when a 2d **image** is converted into a vector is ... the loss function will need to be specified when doing **classification** ... the **clothing** classifier(**Fashion** MNIST): https://**colab**.research.google.com/ ...

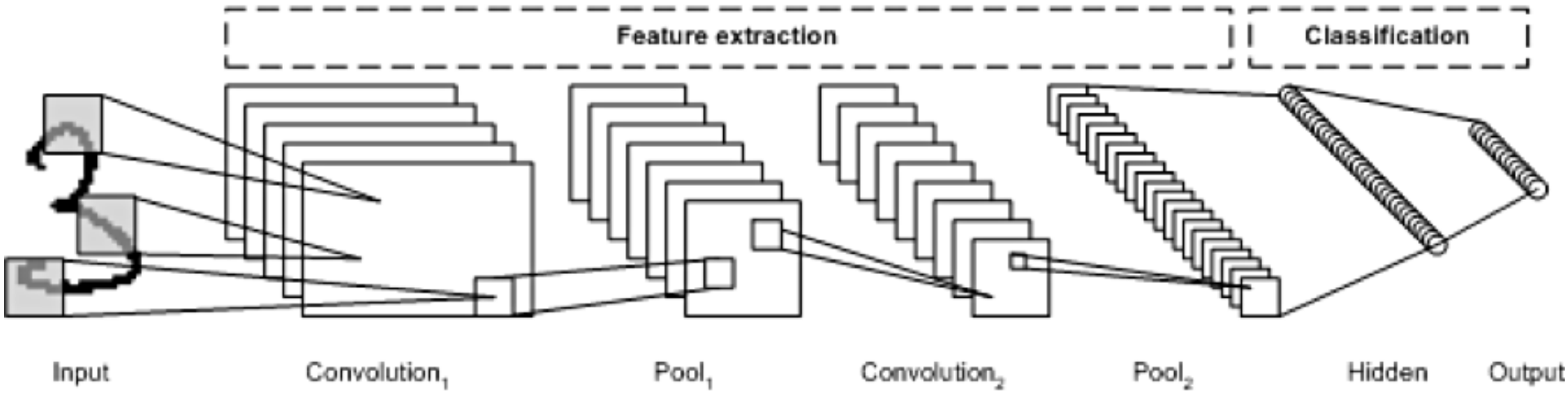
Handy Google Colab notebook for Fashion MNIST classification ...
https://medium.com › handy-google-colab-notebook-for-fashion-mnist-cla...
Nov 22, 2018 - In Lesson 4, we get to classify **fashion images** from MNIST database. This exercise requires access to a GPU. Since not many of us do not have ...

딥러닝



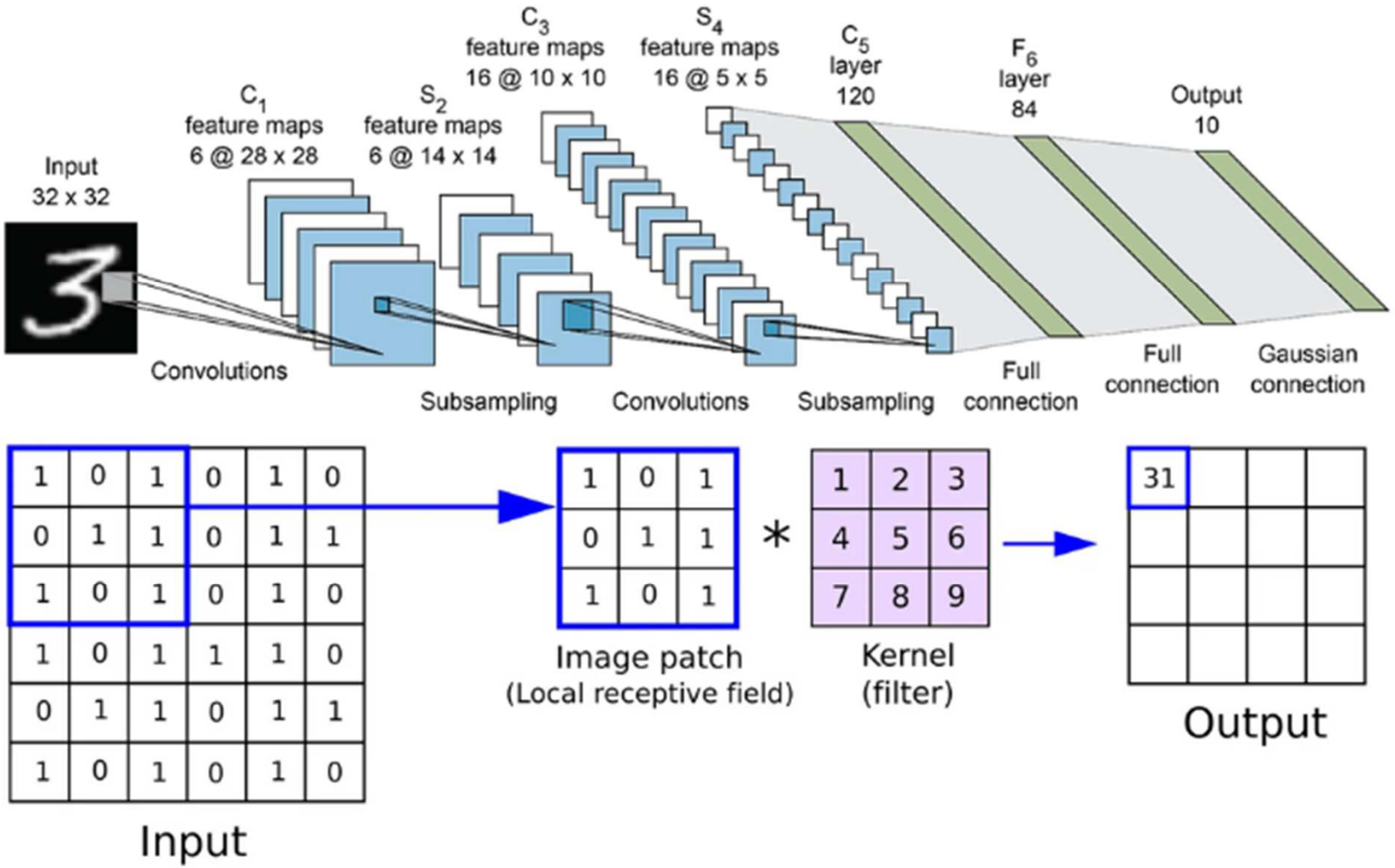
CNN : convolutional neural network

딥러닝



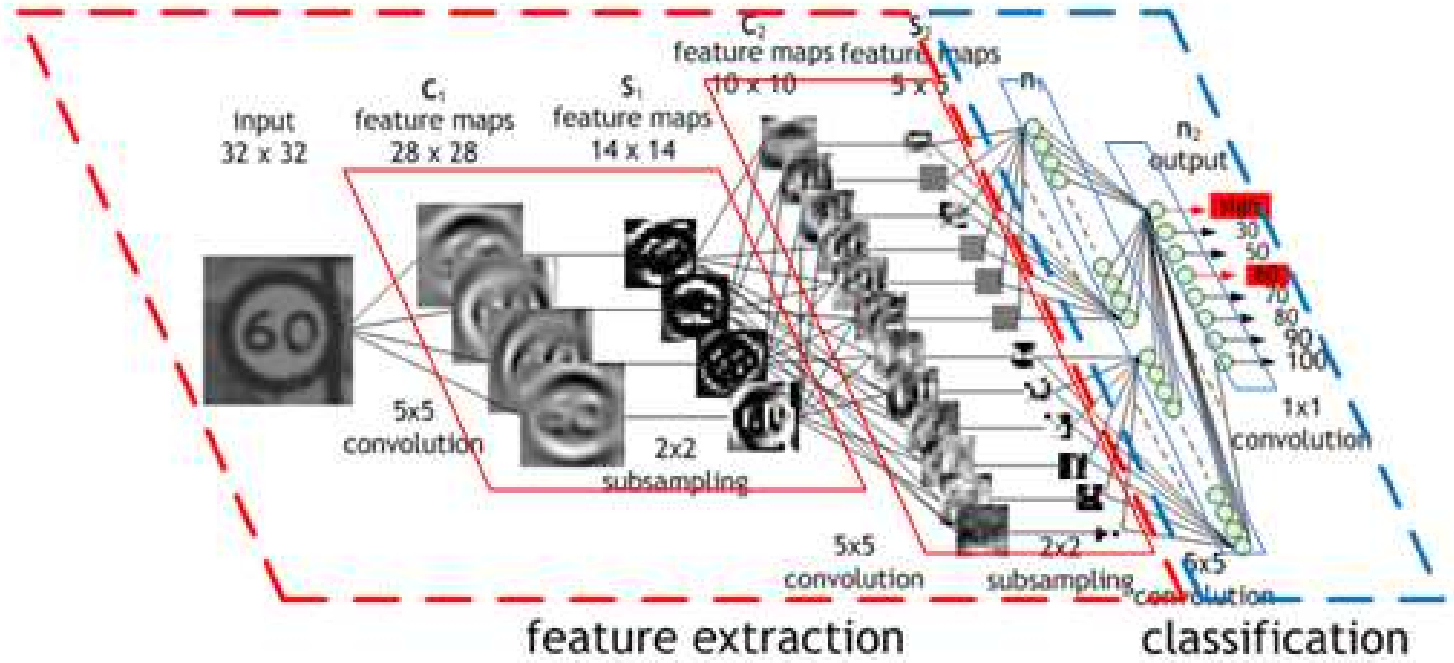
출처 : http://taewan.kim/post/cnn/

딥러닝



출처 : https://www.superannotate.com/blog/guide-to-convolutional-neural-networks

딥러닝



출처 : http://blog.creation.net/mxnet-part-5-vgc16-resnet152

이미지 인식/객체 인식

Source : https://aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=realm&dataSetSn=79

NowSome

이미지 인식/객체 인식

Source : https://aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=realm&dataSetSn=79

NowSome

Q&A

Thank you !!!!