



클라우드 컴퓨팅과 AI서비스 (11주차)

융합학과 권오영

oykwon@koreatech.ac.kr

학습내용

- ❖ Neural Network의 구성
 - MNIST
- ❖ 인공지능프로젝트 구현 절차
- ❖ 합성곱 개념
- ❖ LSTM 개념
- ❖ Stream 설치

NEURAL NETWORK

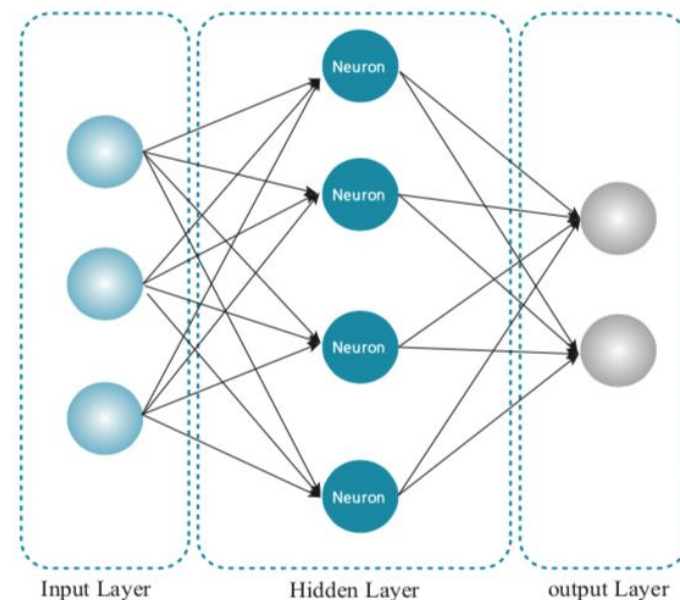
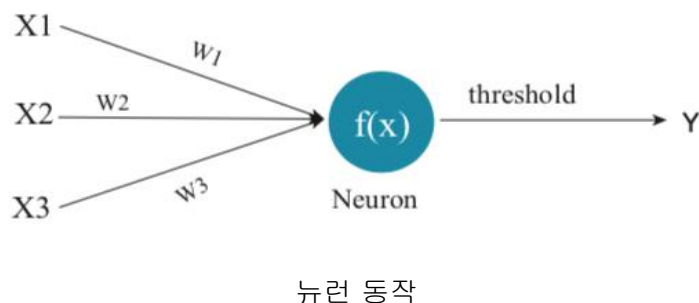
Deep Learning

❖ Deep Learning 이란

- 여러 층을 가진 인공 신경망(Artificial Neural Network)을 사용하여 머신러닝 학습을 수행
- 딥러닝은 기계가 자동으로 학습하려는 데이터에서 특징을 추출하여 학습

❖ 인공신경망(Artificial Neural Network)

- 인공 신경망은 인간의 신경세포 뉴런(Neuron)과 같은 서로 연결된 뉴런은 서로의 입력신호와 출력 신호를 이용하여 동작함
- 뉴런과 신경망 연결 구조

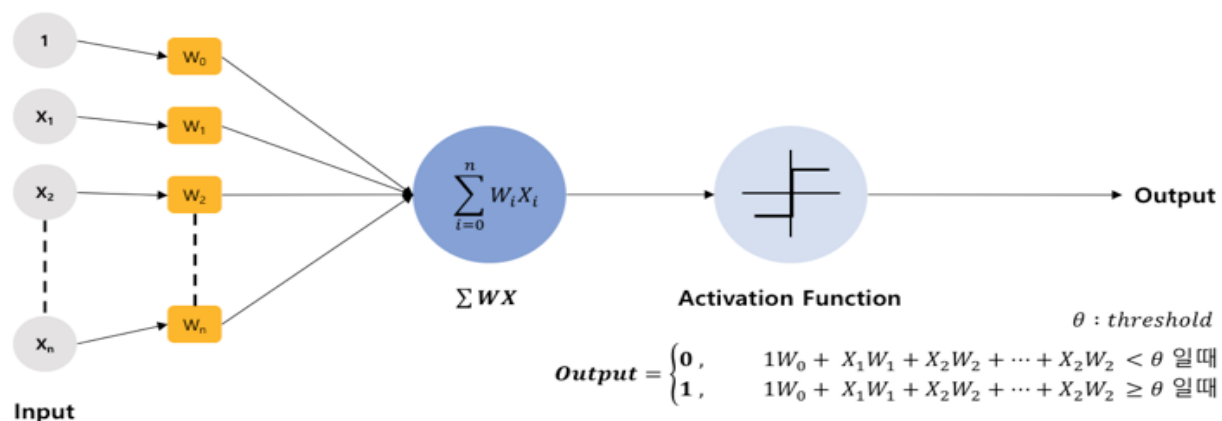


신경망 구성

Perceptron

❖ 퍼셉트론(Perceptron)

- 가장 간단한 인공 신경망 구조
- 다수의 신호(Input)를 입력받아서 하나의 신호(Output)를 출력
- 퍼셉트론 동작 순서
 - ✓ 각각의 입력 신호에 부여된 W(Weight)와 계산
 - ✓ 계산 결과의 총합이 활성화 함수(Activation Function)로 입력
 - ✓ 활성화 함수에서는 정해진 임계값(threshold)을 넘었을때 1을 출력 넘지 못한 경우 0 혹은 -1 을 출력
- W값이 크면 해당 신호는 중요한 신호라고 판단하게 됨
- 일반적으로 퍼셉트론에서 사용되는 활성화 함수는 헤비사이드 계단함수(Heaviside Step Function)이 사용됨



Perceptron

❖ 퍼셉트론 결과값에서 임계값

- 활성화 함수에서 사용하는 임계값(threshold)은 θ 로 표현

- $1W_0 + X_1W_1 + X_2W_2 + \dots + X_nW_n < \theta$ 수식에서 θ 를 $-b$ (bias, 편향)로 치환하여 수식을 변경

$$\text{Output} = \begin{cases} 0, & 1W_0 + X_1W_1 + X_2W_2 + \dots + X_nW_n < \theta \\ 1, & 1W_0 + X_1W_1 + X_2W_2 + \dots + X_nW_n \geq \theta \end{cases} \Rightarrow \text{Output} = \begin{cases} 0, & b + 1W_0 + X_1W_1 + X_2W_2 + \dots + X_nW_n < 0 \\ 1, & b + 1W_0 + X_1W_1 + X_2W_2 + \dots + X_nW_n \geq 0 \end{cases}$$

- 편향(bias)는 학습 데이터(입력신호)와 가중치(Weight)의 계산에 의한 값이 넘어야 할 값
- 편향보다 높으면 1 혹은 0으로 분류되는 기준이 높아지기 때문에 분류할때 엄격하게 분류하게 됨
- 편향값이 높을 수록 학습 모델은 간단해지는 경향을 보이고 Underfitting(과소적합)이 될 수 있음
- 편향값이 낮을 수록 학습 모델은 복잡해지는 경향을 보이고 Overfitting(과적합)이 될 수 있음

❖ W 역할 : 입력 신호가 결과 출력에 주는 영향을 조절

❖ b 역할 : 얼마나 쉽게 활성화(결과를 1로 출력)되는지를 조절

❖ 다층 퍼셉트론(Multi Layer Perceptron, MLP - 다수의 퍼셉트론 사용하는 신경망)을 활용하여 어려운 문제 혹은 비선형적 문제를 해결 할 수 있음

Activation Function

❖ Activation Function(활성화 함수)

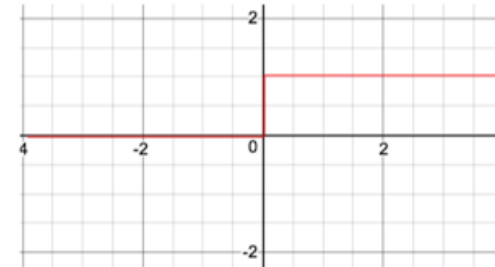
- threshold(임계값)을 이용하여 출력값을 결정하는 함수
- 출력값에 따라서 다음 단계(뉴런)의 입력값의 상태를 결정하게 됨

❖ 종류

▪ Step Function

- ✓ 가장 기본이 되는 활성화 함수로 계단 형태를 가지고 있음
- ✓ 0을 기준으로 0 혹은 1을 출력

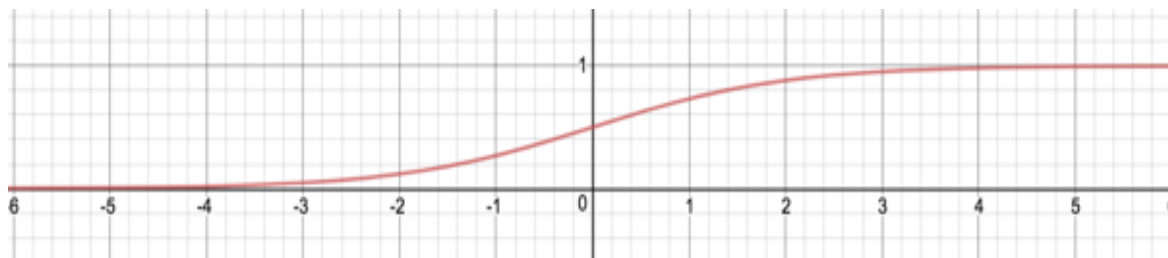
$$\text{Output} = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$$



▪ Sigmoid Function

- ✓ 0과 1 사이의 연속적인 출력값을 가질 수 있도록 하는 비선형 함수
- ✓ 신경망 초기에는 많이 사용되었지만 Gradient Vanishing 현상이 발생하여 최적화가 안되는 현상이 발생하여 최근에는 많이 사용하지 않음

$$P = \frac{1}{1 + e^{-x}}$$



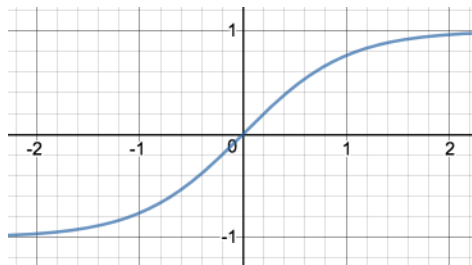
Activation Function

❖ 종류

▪ Hyperbolic Tangent Function, tanh

- ✓ 함수의 중심값을 0으로 옮겨 출력값의 범위는 -1~1 사이의 연속적인 출력값까지는 비선형 함수
- ✓ Sigmoid Function 보다 최적화가 빠르지만 Gradient Vanishing 현상이 발생함

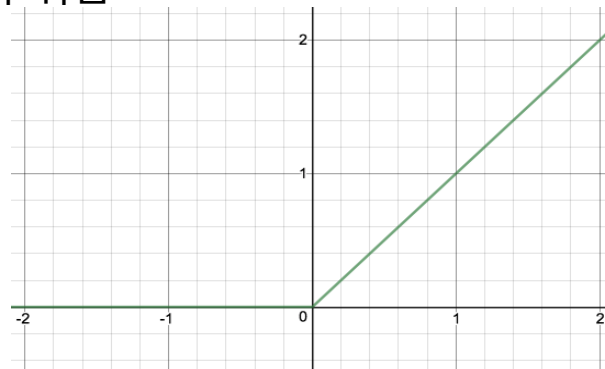
$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



▪ ReLU(Rectified Linear Unit) Function

- ✓ 최근 많이 사용되는 활성화 함수로 x가 0보다 크면 기울기가 1인 직선을 가짐
- ✓ Sigmoid, tanh Function보다 학습이 빠르며 구현이 쉬움
- ✓ x가 0보다 작은 값들에 대해서는 미분시 기울기가 0이기 때문에 뉴런이 활성화가 되지 않음

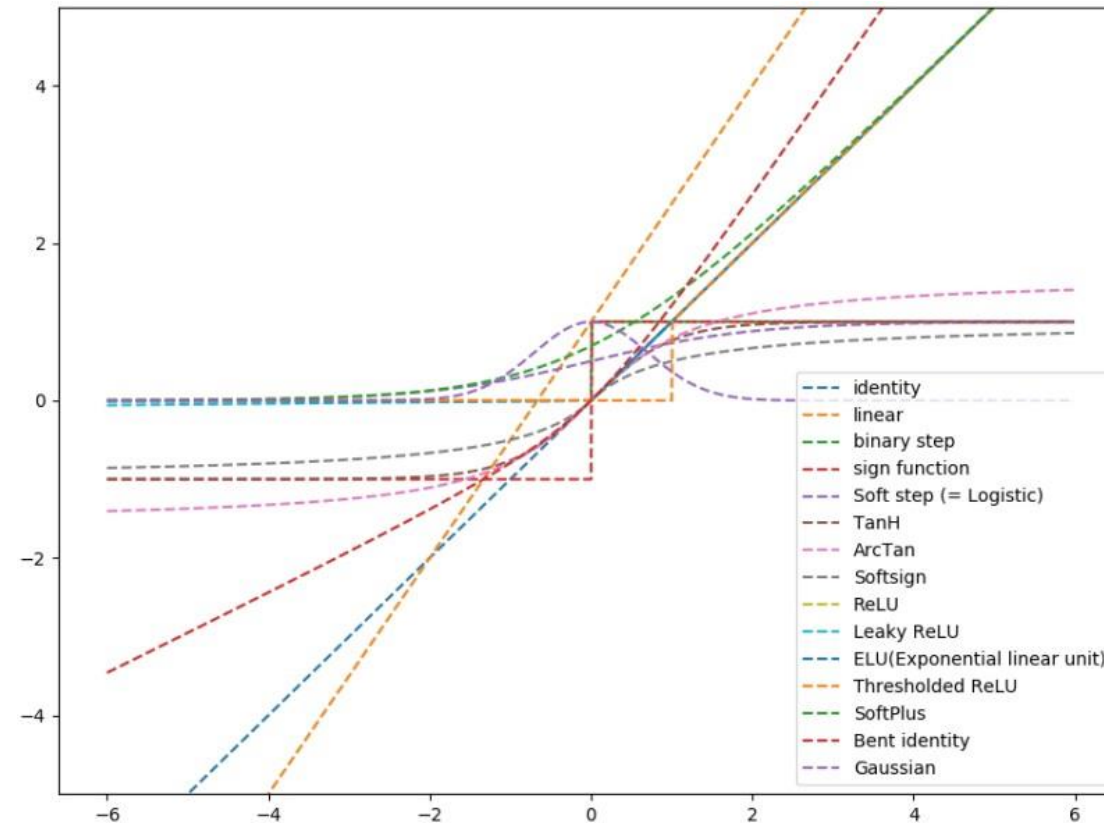
$$f(x) = \max(0, x)$$



Activation Function

❖ 종류

■ 이외의 Activation Function 그래프

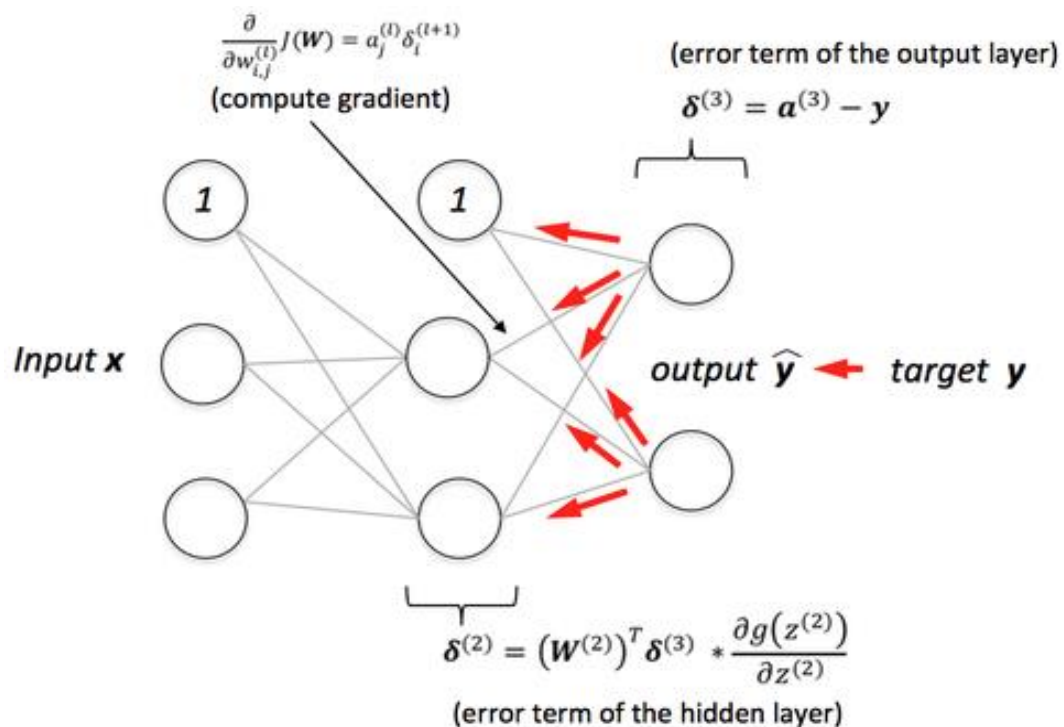


출처 : https://mblogthumb-phinf.pstatic.net/MjAxNzA2MDNfMTQ2/MDAxNDk2NDU0NjE5OTY1.KDNgrWWc2BIWJzitH-7kd6HkA_7tR-uBhSA1SBNhBdgg.-G6q8LTex-T7CvoRCSkuCfULFEFoGSjHa6TxkA7Qm58g.JPEG.wideeyed/%25EC%25A0%2584%25EC%25B2%25B4%25EA%25B7%25B8%25EB%259E%2598%25ED%2594%2584.jpg?type=w800

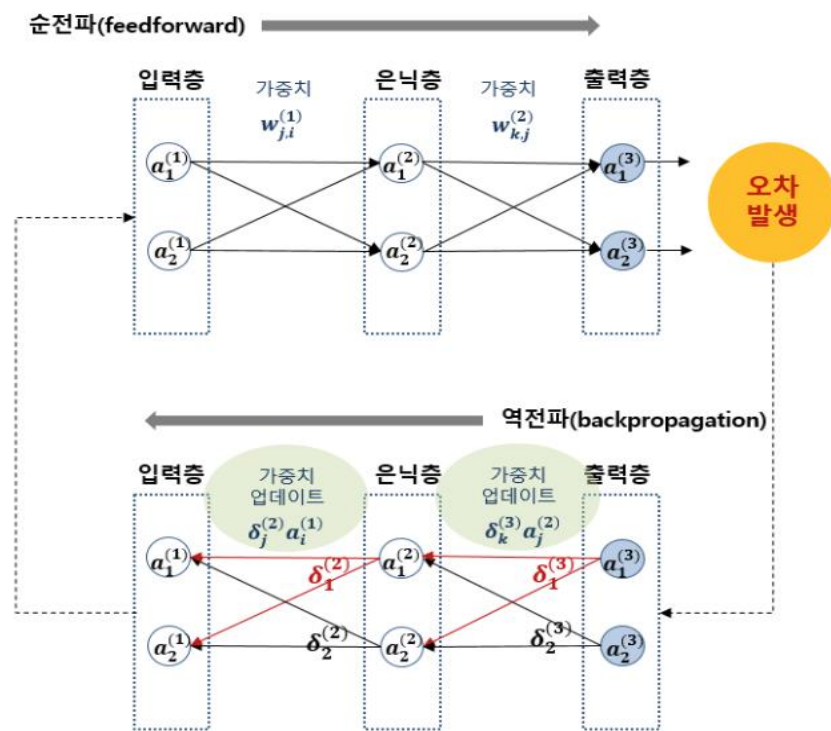
Backpropagation

❖ Backpropagation 알고리즘을 이용한 모델 학습 과정

- 순전파 -> 역전파 -> 가중치업데이트 -> 순전파 -> 역전파 -> 가중치 업데이트 과정을 반복하여 예측값과 결과값의 오차가 최소가 되는 W, b를 찾음



출처 : <https://sebastianraschka.com/faq/docs/visual-backpropagation.html>



출처 : https://m.blog.naver.com/samsjang/221033626685?view=img_75

- 역전파 데모 참고 자료 : <https://google-developers.appspot.com/machine-learning/crash-course/backprop-scroll/?hl=ko>

Backpropagation

❖ 일반적인 비용함수 최적화

- Gradient descent 알고리즘을 이용하여 비용함수 미분을 통하여 오차가 최소가 되는 W (Weight), b (bias) 를 최적화함
- 순전파(Forward propagation) 과정(Input->Hidden->Output Layer)을 통하여 미분값을 업데이트됨

❖ Backpropagation 알고리즘 학습 과정

- 신경망의 W (가중치)를 적당한 값으로 초기화
- Input Layer에 학습데이터를 입력하여 순전파(Forward propagation) 과정을 통하여 비용함수의 미분값 연산 수행
- Output Layer의 출력한 예측값과 실제값의 오차를 계산
- 계산된 오차를 신경망의 각각의 뉴런들에 오차를 역전파(Backpropagate)하여 에러값을 이전 Layer로 전달
- 전달된 오차는 뉴런들의 W 로 사용되며, 오차가 최소가 되는 W, b 를 최적화 함
- 순전파와 역전파과정을 타겟과 예측값의 오차가 허용범위 이내가 될때까지 반복

MNIST NEURAL NETWORK

MNIST Neural Network 개요

❖ MNIST

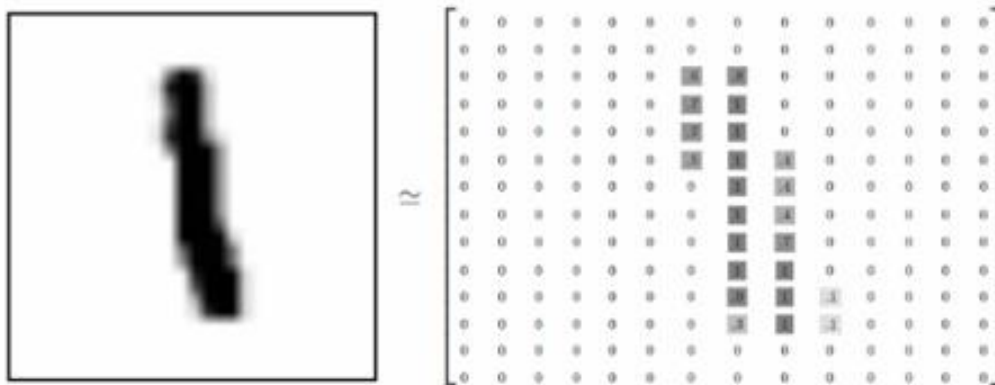
- MNIST(Modified National Institute of Standards and Technology database) 데이터세트
 - ✓ 손으로 쓴 숫자들로 이루어진 대형 데이터베이스
 - ✓ 다양한 화상 처리 시스템을 트레이닝 하기 위해 일반적으로 사용
 - ✓ 55,000개의 훈련데이터와 10,000개의 테스트 데이터 5,000개의 검증 데이터로 구성
 - ✓ 데이터 샘플 이미지



MNIST Neural Network 개요

❖ MNIST

- 손글씨 이미지를 픽셀 데이터로 변환하여 학습에 사용할 수 있도록 함



- 하나의 이미지는 28 x 28 픽셀로 구성되어 있으며 픽셀 데이터를 784(28*28)의 벡터로 변환하여 학습에 사용
- Scikit Learn의 **MLP**를 이용하여 인식기 구성해보자.

MNIST

❖ Random Forest

```
import mnist
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Load dataset
train_images = mnist.train_images()
train_labels = mnist.train_labels()
test_images = mnist.test_images()
test_labels = mnist.test_labels()

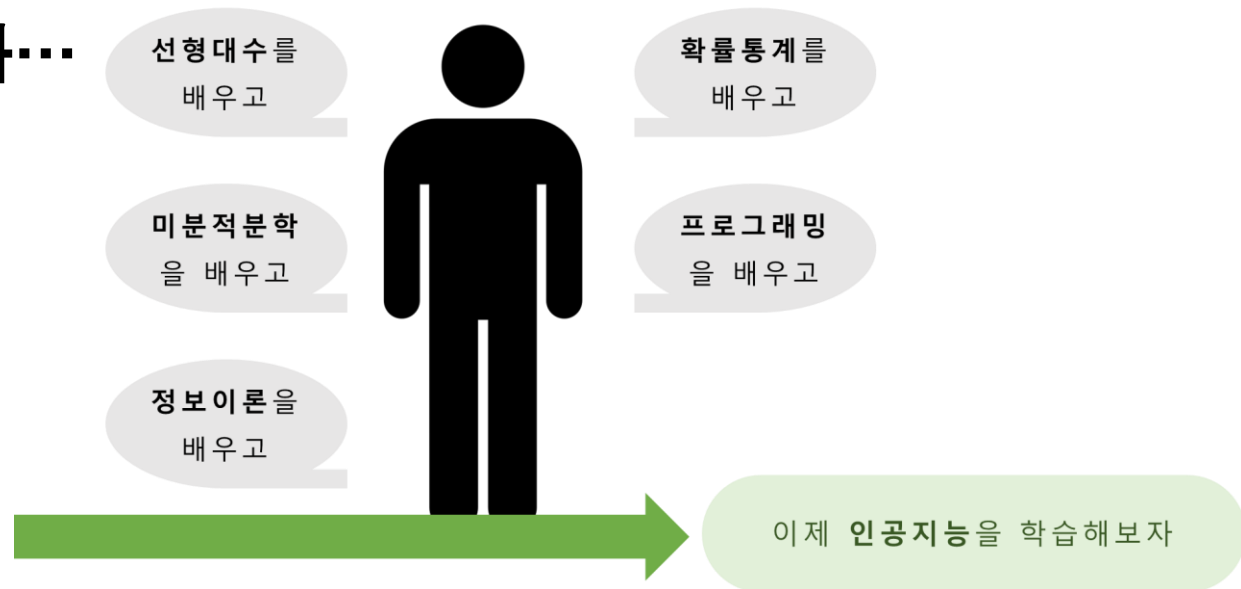
# preprocessing
train_images = train_images.reshape(-1, 28*28)
test_images = test_images.reshape(-1, 28*28)

clf = RandomForestClassifier(n_estimators=100)
clf.fit(train_images[:10000], train_labels[:10000])
# Test on the next 1000 images:
test_x = train_images[10000:11000]
expected = train_labels[10000:11000].tolist()
print("Compute predictions")
predicted = clf.predict(test_x)
print("Accuracy: ", accuracy_score(expected, predicted))
```

인공지능 프로젝트 만들기

인공지능응용 개발은 어디서부터 시작해야하나?

- Bottom up 접근
- 선형대수, 미분적분학, 확률과통계,
프로그래밍등 차근차근 배워나가자...



<https://machinelearningmastery.com/machine-learning-for-programmers/>

인공지능응용 개발은 어디서부터 시작해야하나?

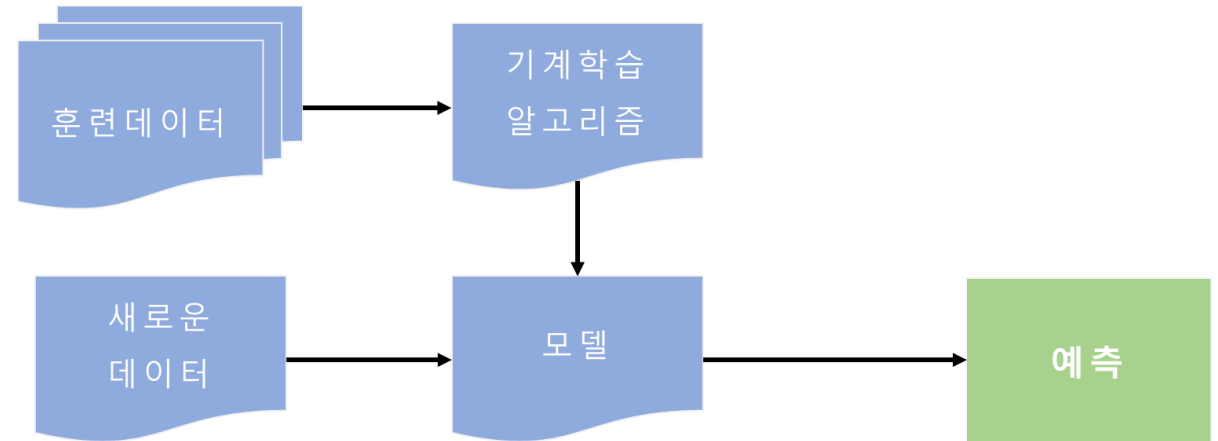
- Top down 접근
- 목표(결과물)를 먼저 설정하고, 목표 달성에 필요한 내용을 채워나감
 - Step 1: 마음가짐 조정 (신념:할 수 있다)
 - Step 2: 절차(프로세스) 선택
 - Step 3: 도구 선택 (구현을 하기 위해)
 - Step 4: 데이터세트를 갖고 연습
 - Step 5: 포트폴리오 구축 (실력을 보여주자).

출처 <https://machinelearningmastery.com/machine-learning-mastery-method/>

인공지능응용 가장 많이 쓰이는 분야

- 예측 모델링

- 데이터 수집
- 데이터를 활용해서 모델학습
- 새로운 데이터가 입력되었을 때 예측



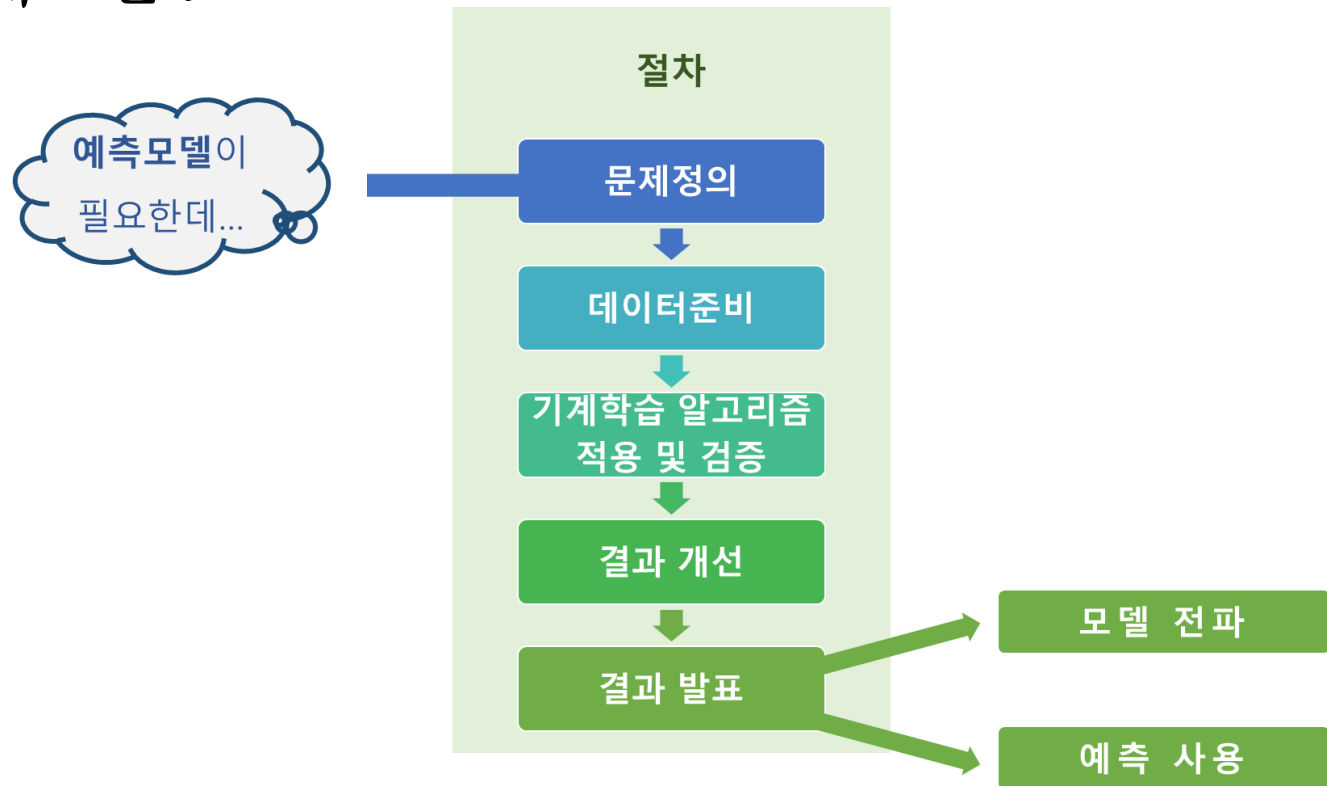
- 예) 구글 티처블머신

<https://teachablemachine.withgoogle.com>

인공지능응용 만들기

• 인공지능응용개발 프로세스는?

- 문제정의
- 데이터준비
- 알고리즘점검
- 결과개선
- 결과발표 (활용)



출처 <https://machinelearningmastery.com/machine-learning-mastery-method/>

인공지능응용개발에 적절한 도구는?

- Python 패키지 (계산)

- NumPy: 수치 및 과학 계산을 위한 기본 라이브러리로 숫자 형 배열, 선형 대수 도구, 난수 기능 등을 위한 데이터 구조가 포함되어 있음
- SciPy: 최적화, 보간, 통계 및 신호 처리와 같은 과학 컴퓨팅을 위한 다양한 기능을 제공
- Matplotlib: Python의 핵심 plotting 라이브러리며 주피터 노트북에서 매직 명령어를 이용하여 `%matplotlib notebook` 또는 `%matplotlib inline` 형태로 사용가능
- Sympy: Symbolic 계산 지원
- Pandas: 데이터 분석을 위한 리소스와 레이블이 지정된 테이블 형식의 데이터를 위한 유연한 데이터 구조 제공

인공지능응용개발에 적절한 도구는?

- Python 패키지 (기계 학습)

- Scikit-learn: 머신러닝 라이브러리(예측분석을 위한 각종 도구와 알고리즘 제공)
- Keras: TensorFlow위에서 수행할 수 있는 상위 수준의 오픈 소스 딥러닝 라이브러리 (초보자가 신경망을 쉽게 구성할 수 있다.)
- TensorFlow: 데이터 흐름(data flow) 프로그래밍을 위한 오픈소스 라이브러리로 인공 신경망과 같은 기계학습 프로그램에 널리 사용



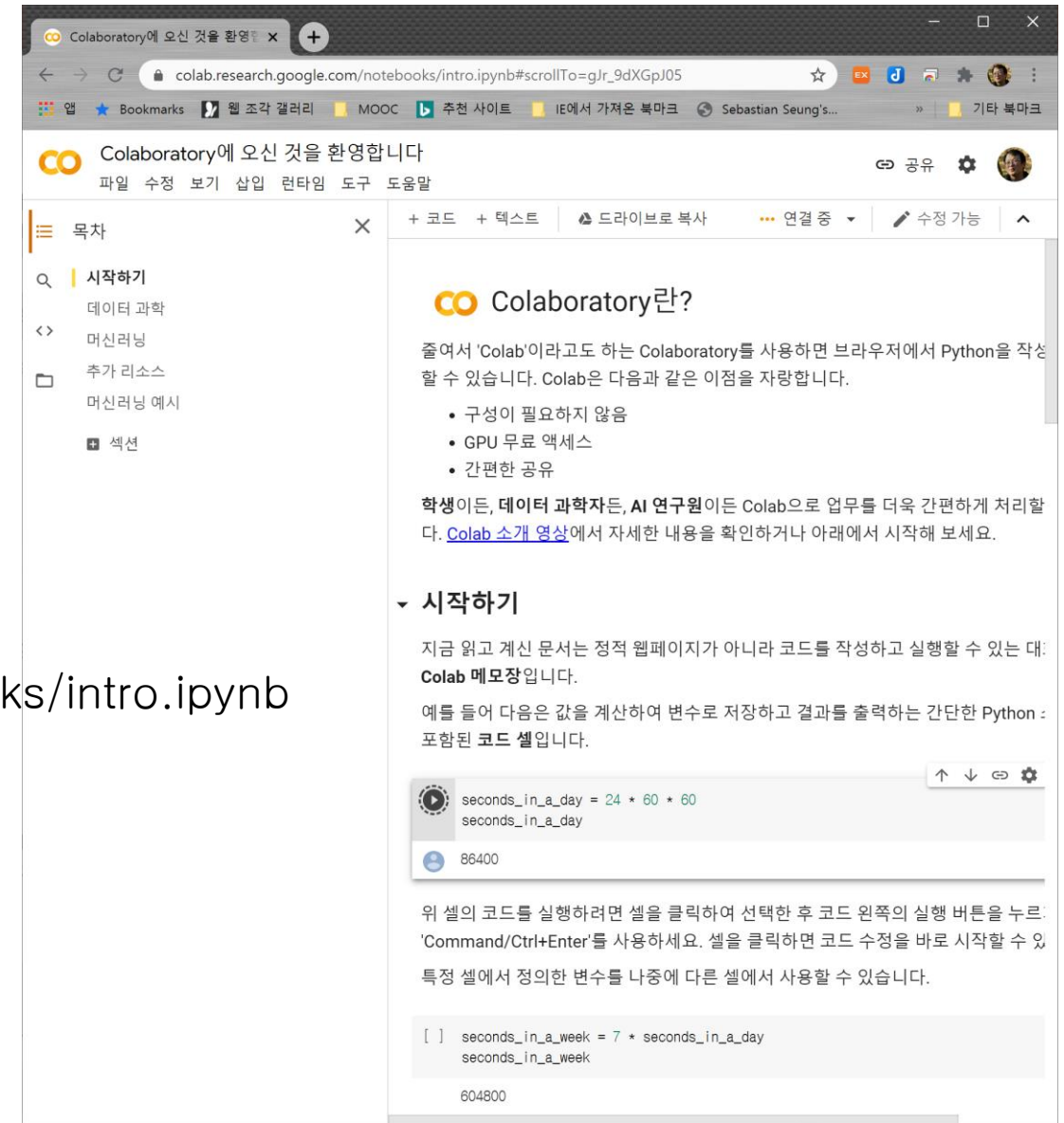
파이선 IDE (통합개발환경)

- Thonny (초보자용 <https://thonny.org/>)
 - MicroPython, Micro:Bit, Raspberry Pi 등도 쉽게 연결
- PyCharm (전문개발자용 <https://www.jetbrains.com/ko-kr/pycharm/>)
- Visual Studio Code (<https://code.visualstudio.com/>)
- Jupyter Notebook (<https://jupyter.org/>)
- Ananconda (<https://www.anaconda.com/>)

클라우드 활용

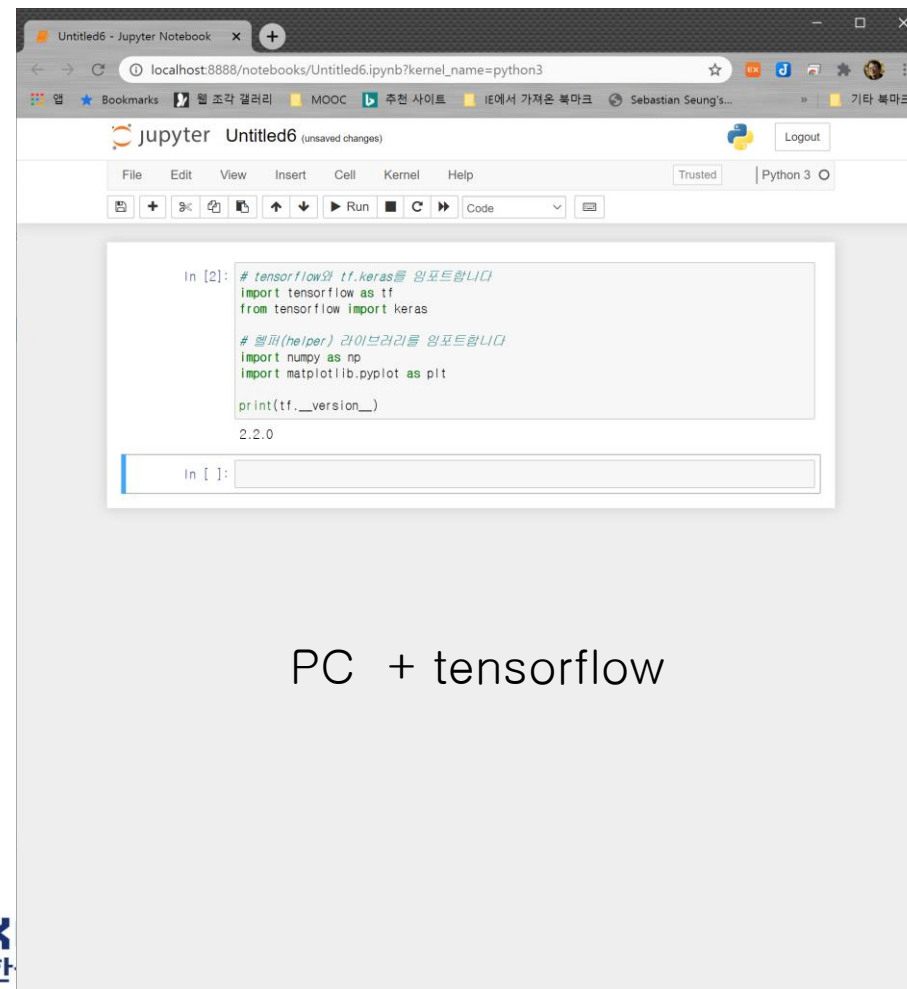
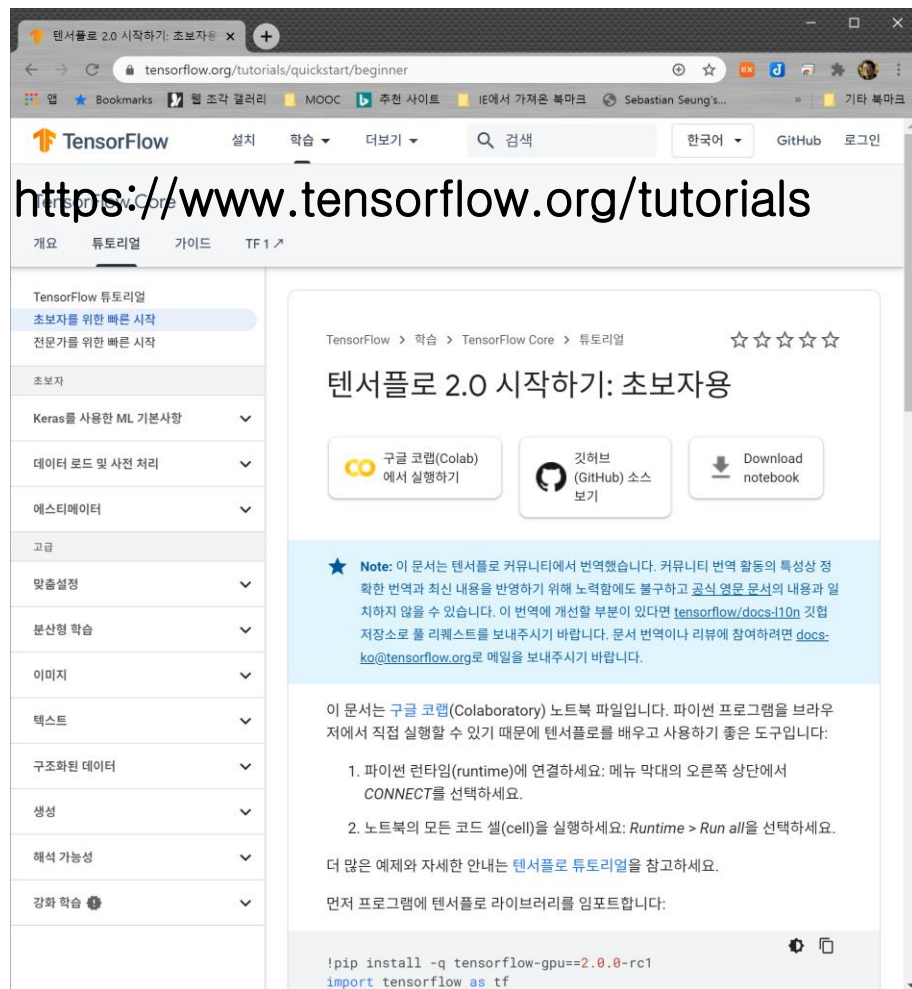
- 구글 colab
 - 구글이 제공
 - Jupyter notebook 확장
 - GPU/TPU 무료 액세스

<https://colab.research.google.com/notebooks/intro.ipynb>



클라우드 활용

• 인공지능활용(Tensorflow)



PC + tensorflow

연습하기

<https://machinelearningmastery.com/machine-learning-mastery-method/>

- **표준화된 기계 학습 데이터 세트를 가지고 연습**
 - 실제 문제 도메인에서 수집 된 데이터 세트를 사용
 - 메모리 또는 Excel 스프레드 시트에 맞는 작은 데이터 세트를 사용
 - 예상되는 결과를 알 수 있는 잘 이해된 데이터 세트를 사용
- **UCI 기계 학습 저장소 사용**
 - 가장 많이 사용되고 가장 잘 이해되는 데이터 세트로 처음 시작하기 좋은 곳
 - <http://archive.ics.uci.edu/ml/datasets.php>
- **Kaggle과 같은 기계 학습 대회용 사용**
 - 대회 데이터라 규모가 크고, 모델을 잘 만들기 위한 좀 더 준비해야 하는 데이터
 - <https://www.kaggle.com/datasets>
- **자신이 고안한 문제에 대한 연습**

나만의 프로젝트에 도전하십시오.

- 간단한 프로세스와 간단한 도구로 시작한 다음,
자신감이 생기면 더 어려운 단계로 진행
- 더 어려운 문제를 해결하려고 도전
(쉬운 문제는 배우는 것이 적을 수 있음)
- 포럼과 Q & A 사이트와 같은 커뮤니티에 참여

CNN응용

응용: 딥러닝을 활용한 말라리아 판별

- 소스:
<https://opensource.com/article/19/4/detecting-malaria-deep-learning>
- 데이터 125x125 로 resize
- 감염된 데이터 1000개,
건강한 데이터 1000개
(13779중 일부사용)
- CNN 활용

```
Windows PowerShell

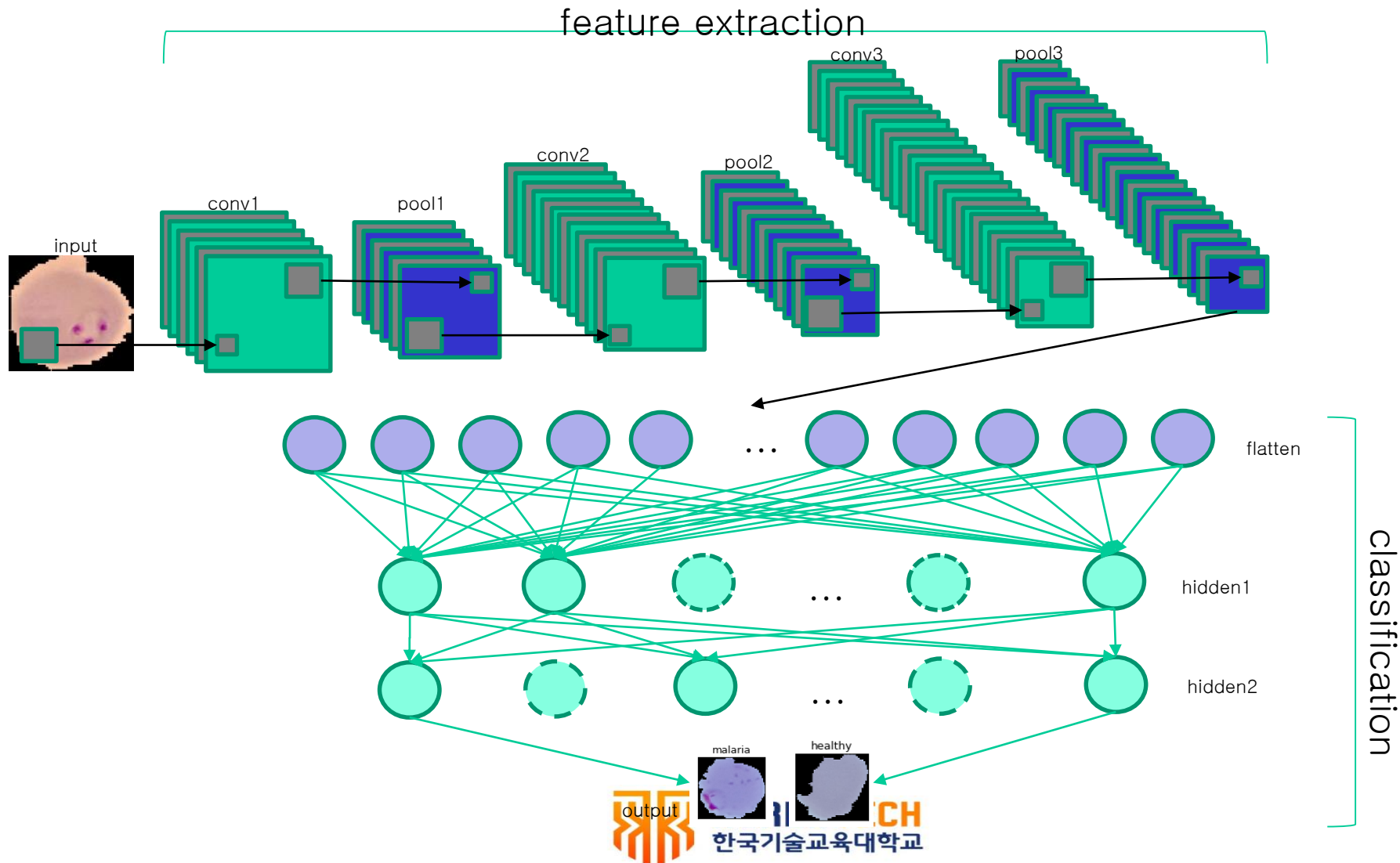
PS D:\mycodes\malaria> tree .\cell_images\
폴더 PATH의 목록입니다.
볼륨 일련 번호는 1E4A-BE1D입니다.
D:\MYCODES\MALARIA\CELL_IMAGES
├─Parasitized
└─Uninfected
PS D:\mycodes\malaria> ls .\cell_images\Parasitized\ | Measure-Object -Line

Lines Words Characters Property
-----
2000

PS D:\mycodes\malaria> ls .\cell_images\Uninfected\ | Measure-Object -Line

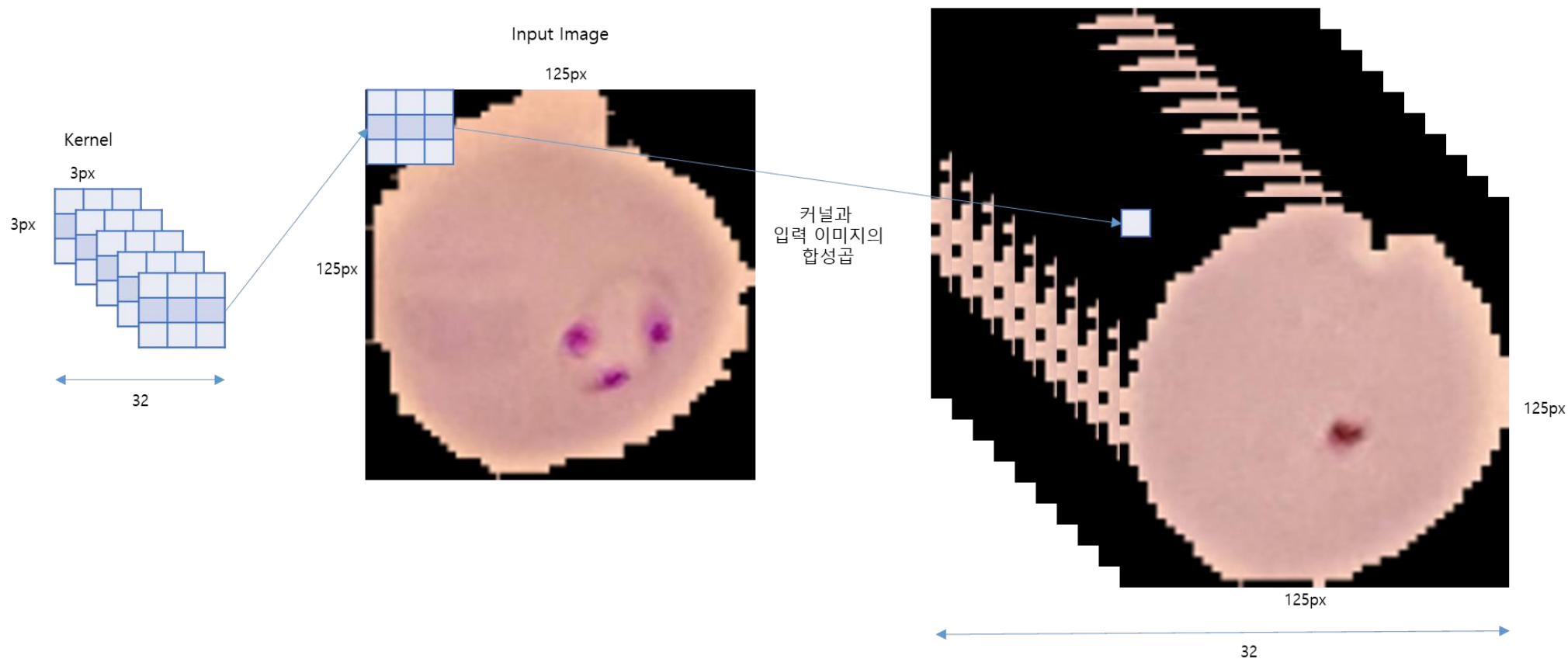
Lines Words Characters Property
-----
2000
```

응용: 딥러닝을 활용한 말라리아 판별



응용: 딥러닝을 활용한 말라리아 판별

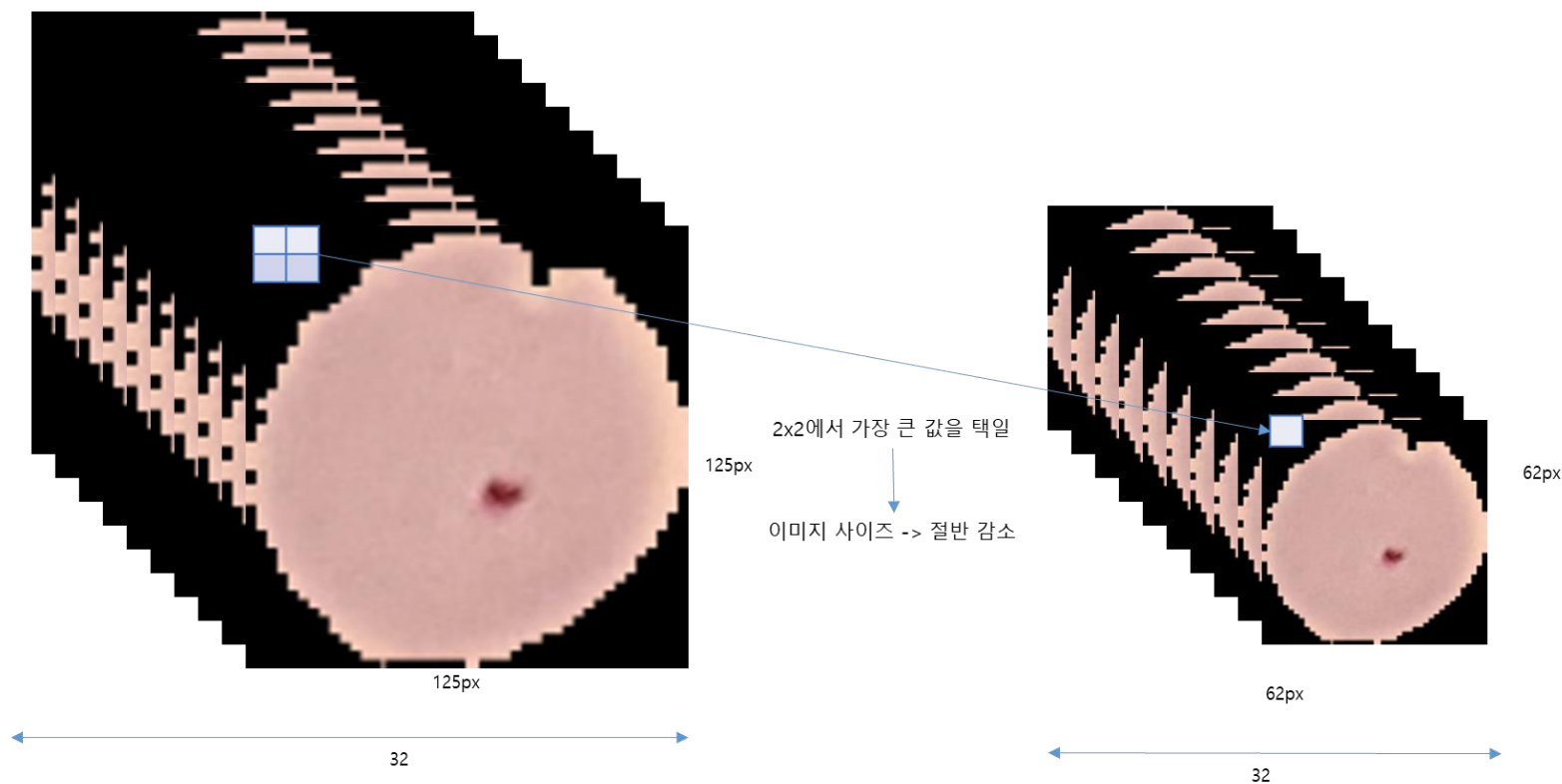
합성곱 계층(conv1)



응용: 딥러닝을 활용한 말라리아 판별

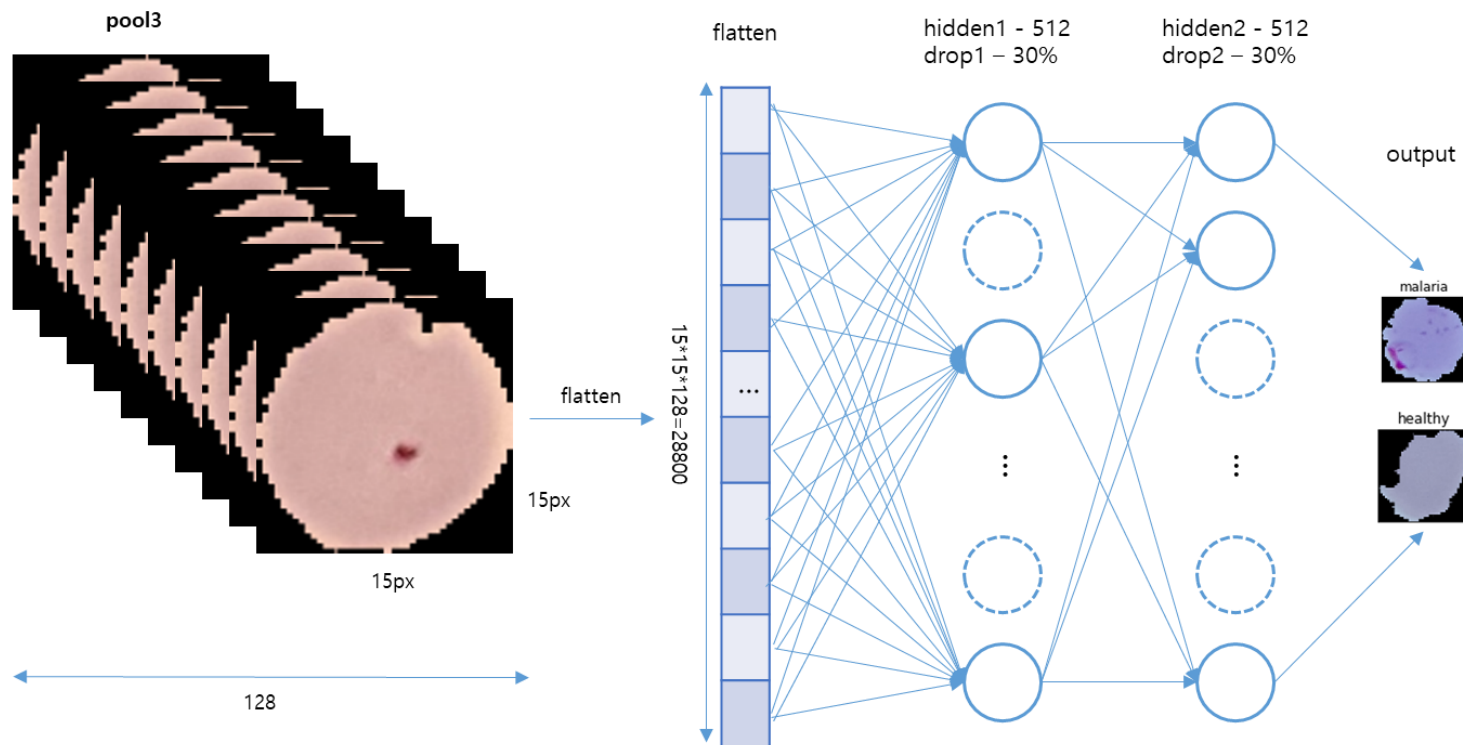
풀링 계층(pool1)

conv2 -> pool2 -> conv3 -> pool3 반복



응용: 딥러닝을 활용한 말라리아 판별

Flatten – Hidden – Output



응용: 딥러닝을 활용한 말라리아 판별

Thonny - D:\Wmycodes\Wmalaria\Wmalaria.py @ 140 : 1

File Edit View Run Device Tools Help

```
malaria.py * x
123
124 inp = tf.keras.layers.Input(shape=INPUT_SHAPE)
125
126 conv1 = tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu', padding='same')(inp)
127 pool1 = tf.keras.layers.MaxPooling2D(pool_size=(2, 2))(conv1)
128 conv2 = tf.keras.layers.Conv2D(64, kernel_size=(3, 3), activation='relu', padding='same')(pool1)
129 pool2 = tf.keras.layers.MaxPooling2D(pool_size=(2, 2))(conv2)
130 conv3 = tf.keras.layers.Conv2D(128, kernel_size=(3, 3), activation='relu', padding='same')(pool2)
131 pool3 = tf.keras.layers.MaxPooling2D(pool_size=(2, 2))(conv3)
132
133 flat = tf.keras.layers.Flatten()(pool3)
134
135 hidden1 = tf.keras.layers.Dense(512, activation='relu')(flat)
136 drop1 = tf.keras.layers.Dropout(rate=0.3)(hidden1)
137 hidden2 = tf.keras.layers.Dense(512, activation='relu')(drop1)
138 drop2 = tf.keras.layers.Dropout(rate=0.3)(hidden2)
139
140 out = tf.keras.layers.Dense(1, activation='sigmoid')(drop2)
141
142 model = tf.keras.Model(inputs=inp, outputs=out)
143 model.compile(optimizer='adam',
144               loss='binary_crossentropy',
145               metrics=['accuracy'])
146 model.summary()
```

Shell x

Epoch 14/25
21/40 [=====>.....] - ETA: 18s - loss: 0.6931 - accuracy: 0.5112

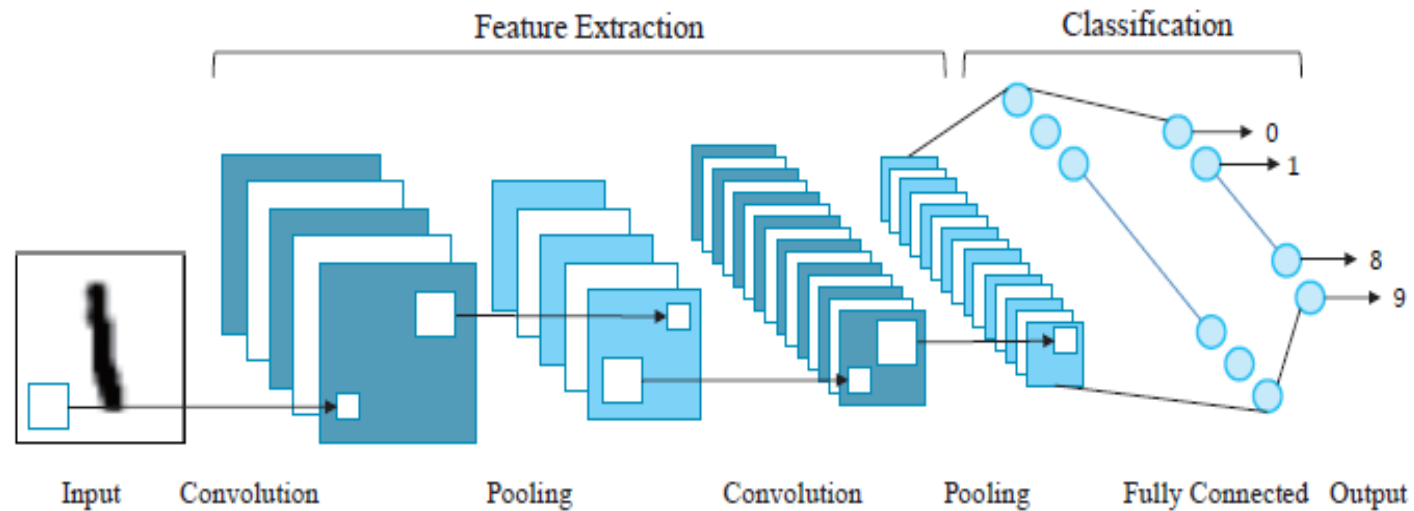
합성곱 신경망(CNN) 구현

❖ 합성곱 신경망이란?

- 1998년 Yann Lecun이 처음 제안한 알고리즘
- 페이스북의 자동 사진 태그, Google과 네이버의 이미지 검색, 아마존의 제품 추천, 카카오의 형태소 분석기 등

❖ 합성곱 신경망 구조

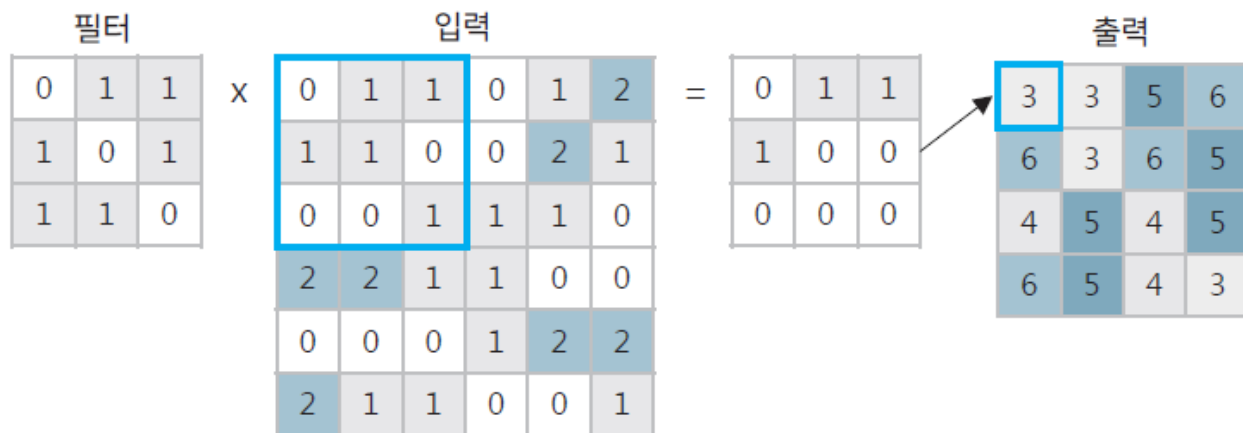
- 합성곱 계층, 풀링 계층, 완전 결합 계층으로 구성



합성곱 신경망(CNN) 구현

- 합성곱 계층

- 특징을 추출하기 위한 필터(filter) / 커널(kernel)
- 이미지의 행렬을 합성곱 하여 특성 맵(feature map) 구성



- ✓ 필터와 이미지의 각 위치에 있는 값들을 곱하고 모든 행렬의 값을 더하여 구성
- 스트라이드(stride) : 옆으로 이동하며 동일한 연산을 계속 진행
 - ✓ 스트라이드의 크기에 따라 출력값의 크기 변경
 - 이미지의 크기가 6x6, 필터가 3x3으로 구성
 - 스트라이드 값이 1이면 출력 이미지 크기는 4x4 로 구성
- 활성화 함수로는 ReLU를 주로 사용

합성곱 신경망(CNN) 구현

- 합성곱 계층

- 제로패딩(Zero Padding)

- ✓ 필터의 크기와 스트라이드 값에 따라 출력 이미지 크기가 줄어드는 것 방지
 - ✓ 입력 이미지의 행렬의 상, 하, 좌, 우에 0을 채움

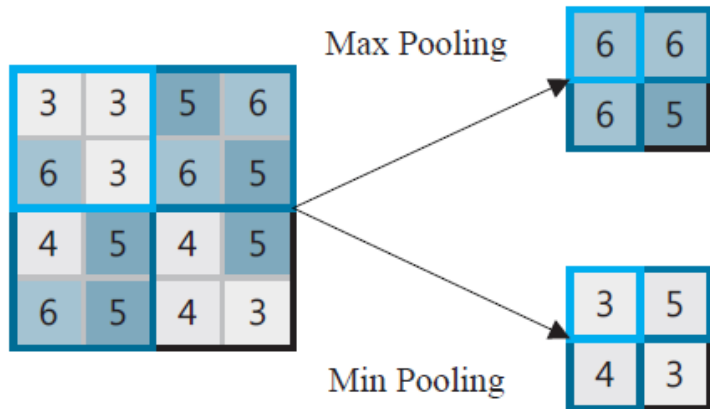


- 입력 이미지 6×6 크기에 제로 패딩을 사용하여 8×8 로 구성
 - 3×3 크기의 필터를 한 칸씩 스트라이드
 - 출력 이미지의 크기는 입력 이미지의 크기와 동일하게 구성됨

합성곱 신경망(CNN) 구현

- 풀링 계층

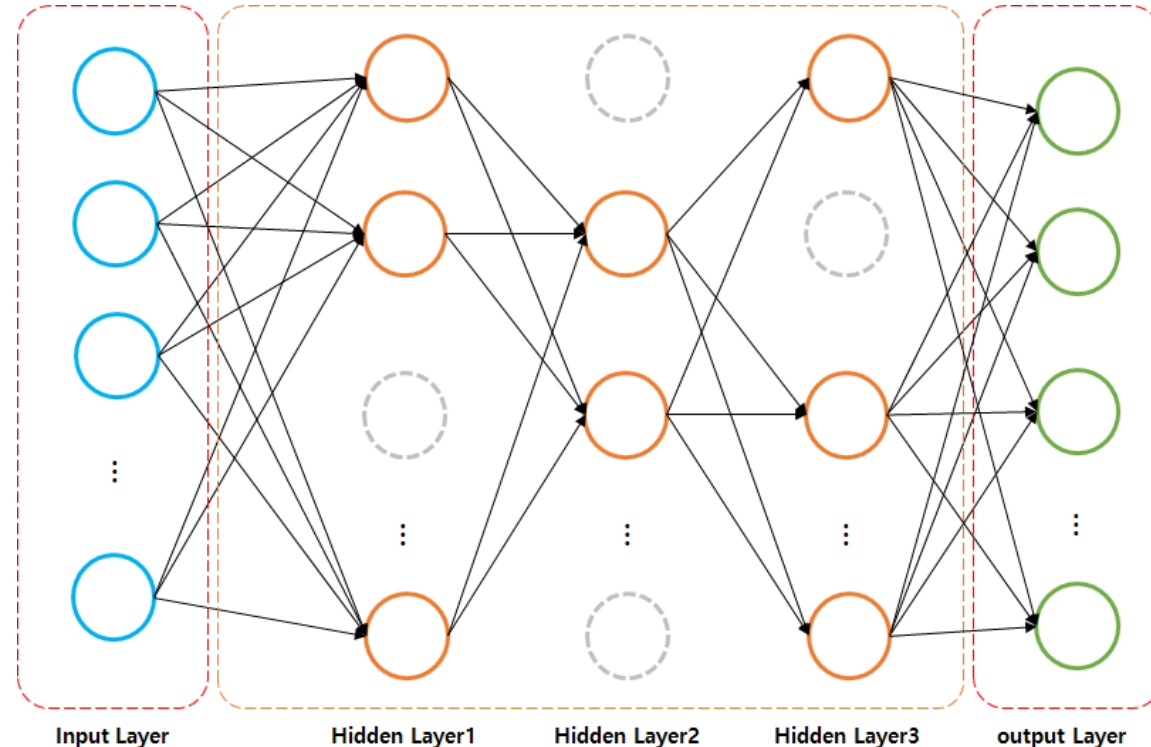
- 선택된 영역에서의 최솟값(Min Pooling), 최댓값(Max Pooling), 평균값(Average Pooling)을 풀링하여 이미지를 축소 처리
 - ✓ 차원을 축소함에 따라 연산량 감소
 - ✓ 과적합(Overfitting)을 방지
 - ✓ 영역 내에서의 특징을 가진 부분을 추출
- 4×4 크기의 입력 이미지를 2×2 크기의 필터와 스트라이드 값을 2로 설정



- 합성곱 신경망에서는 주로 최댓값 풀링을 사용

합성곱 신경망(CNN) 구현

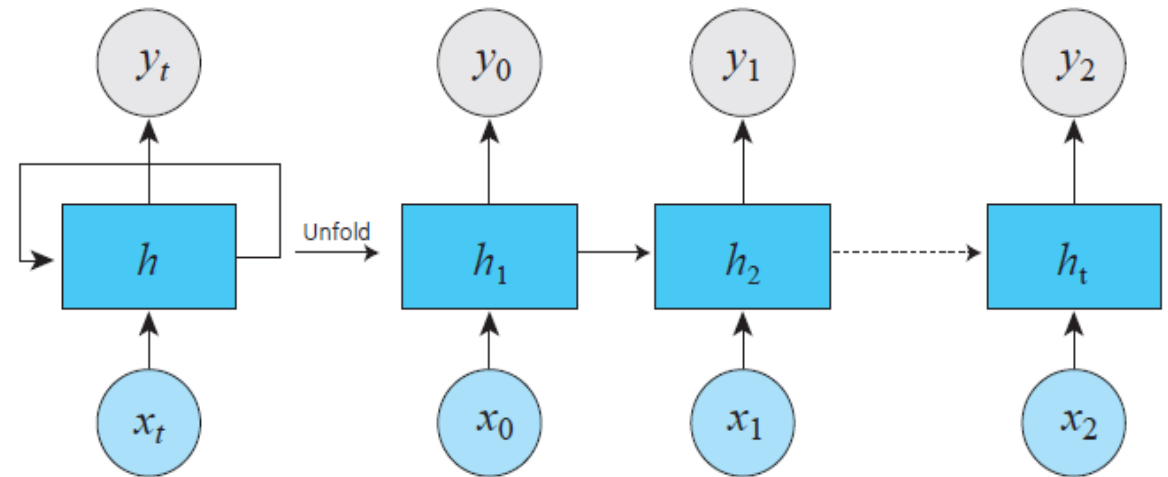
- 드롭 아웃(DropOut)
 - 완전 결합 계층에서의 과적합(Overfitting)을 방지
 - 신경망에서의 뉴런들을 임의적으로 선택하여 버린 후 나머지 뉴런들에 대해서만 학습
 - 학습 시에는 드롭아웃 을 사용하고, 학습 이후 검증 시에는 모든 뉴런들을 사용하도록 드롭아웃을 사용하지 않는 것이 일반적인 방식



장단기 기억 네트워크 (LONG-SHORT TERM MEMORY NETWORK)

장단기 기억 네트워크(LSTM) 구현

- ❖ 순환신경망(Recurrent Neural Network, RNN) 이란?
 - 문서 감정 분류, 필기체 인식, 음성 인식과 같은 자연어 처리
 - 주가 등 시간을 중심으로 앞, 뒤의 내용이 연관 관계가 있는 시계열 데이터를 처리에 좋은 성능
 - 시간 스텝 t 에서의 입력값 x_t , 출력값 y_t 와 h 인 은닉층이 존재
 - ✓ 은닉층의 출력이 다음 시간 스텝에서의 은닉층으로 입력되는 구조가 반복되는 형태
 - ✓ 하나의 네트워크 구조가 여러 개가 연결되어 다음 단계로의 정보를 전달
 - ✓ 메모리 셀(memory cell) : 이전 정보를 은닉층에서 일시적으로 메모리(memory) 형태로 기억
 - ✓ 은닉 상태(hidden state) : 메모리 셀의 상태
 - ✓ 은닉 상태값은 현재 입력값과 이전의 은닉 상태의 값을 가중치를 곱하고 편향을 더함
 - ✓ 활성화 함수로 하이퍼볼릭 탄젠트(tanh) 함수



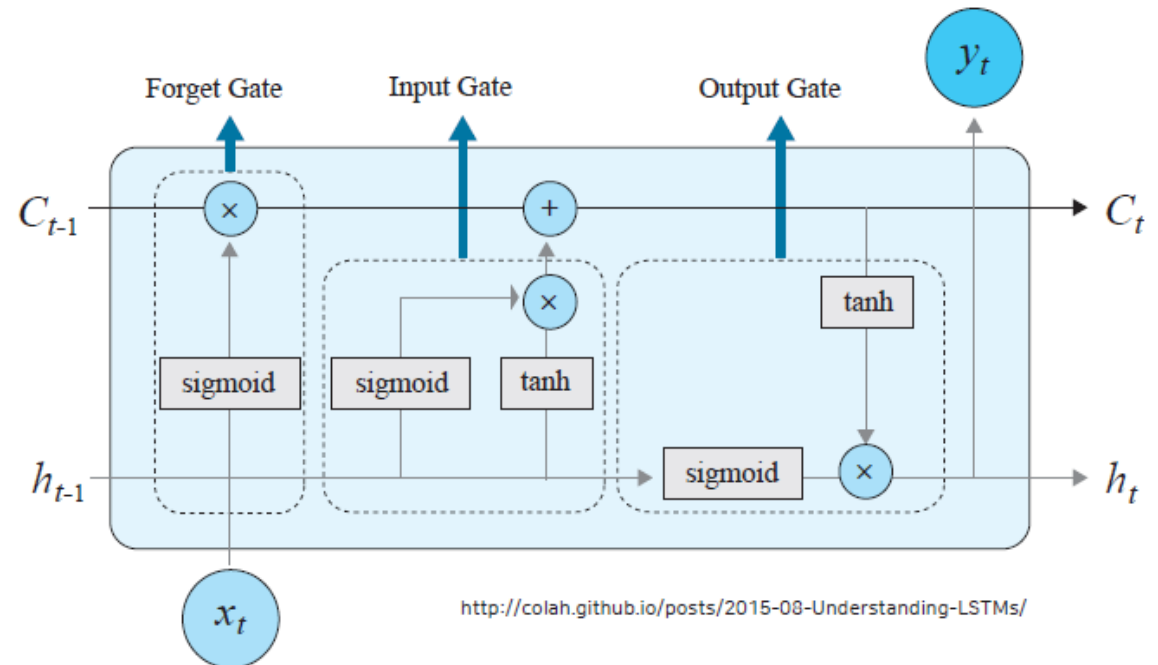
장단기 기억 네트워크(LSTM) 구현

- ❖ 순환신경망(Recurrent Neural Network, RNN) 이란?
 - 순환 신경망의 학습에는 경사 하강법을 이용하며 출력에서의 경사가 현재 시간에만 의존하는 게 아니라 이전 시간 스텝에도 의존
 - 시간 기반 역전파(BackPropagation Through Time, BPTT)라는 변형된 알고리즘으로 가중치를 업데이트
 - 경사도 사라짐 문제(Gradient Vanishing Problem)
 - ✓ 시간을 많이 거슬러 올라가게 되면 신경망이 곱하기 연산으로 되어 있기 때문에 역전파에서의 경사가 점점 줄어들어 학습 능력이 저하하는 단점
- ❖ 장단기 기억 네트워크(Long-Short Term Memory Network, LSTM) 이란?
 - 1997년 Hochreiter & Schmidhuber 이 제안한 알고리즘
 - 순환 신경망에서의 장기 의존성(Long-Term Dependencies) 문제를 해결
 - 순차적으로 입력되는 데이터의 시간 흐름이 길더라도 잊어야 할 정보들은 잊고 유지해야 될 정보는 유지하면서 성능을 최적화
 - 순환 신경망에서 존재하지 않던 ct인 셀 스테이트(Cell State)가 추가

장단기 기억 네트워크(LSTM) 구현

❖ 장단기 기억 네트워크구조

- 망각, 입력, 출력의 정도를 조절하는 3개의 게이트(Gate)가 추가
 - ✓ 셀 스테이트 : 각 게이트의 정보들이 다음 단계로 진행될 수 있도록 역할
 - ✓ 망각 게이트 : 셀 스테이트에서 버릴 정보를 정하는 단계
 - 입력값과 이전 은닉층에서 입력된 값과 함께 시그모이드 출력값 생성
 - 시그모이드 출력값이 1인 경우 과거의 값을 그대로 유지하고, 0인 경우에는 완전히 값을 버림
 - ✓ 입력 게이트 : 새로운 정보에 대해 셀 스테이트에 저장할지를 결정하는 단계
 - 시그모이드를 통해 업데이트할 정보 결정
 - tanh 레이어를 통해 셀 스테이트에 더할 새로운 후보 값을 만들고 두 값을 합쳐 새로운 셀 스테이트로 정보를 업데이트
 - ✓ 출력 게이트
 - 어떤 값을 출력할지 시그모이드 레이어를 통해 결정
 - 셀 스테이트를 tanh 레이어를 통한 곱값을 곱하여 원하는 곱값만 반영

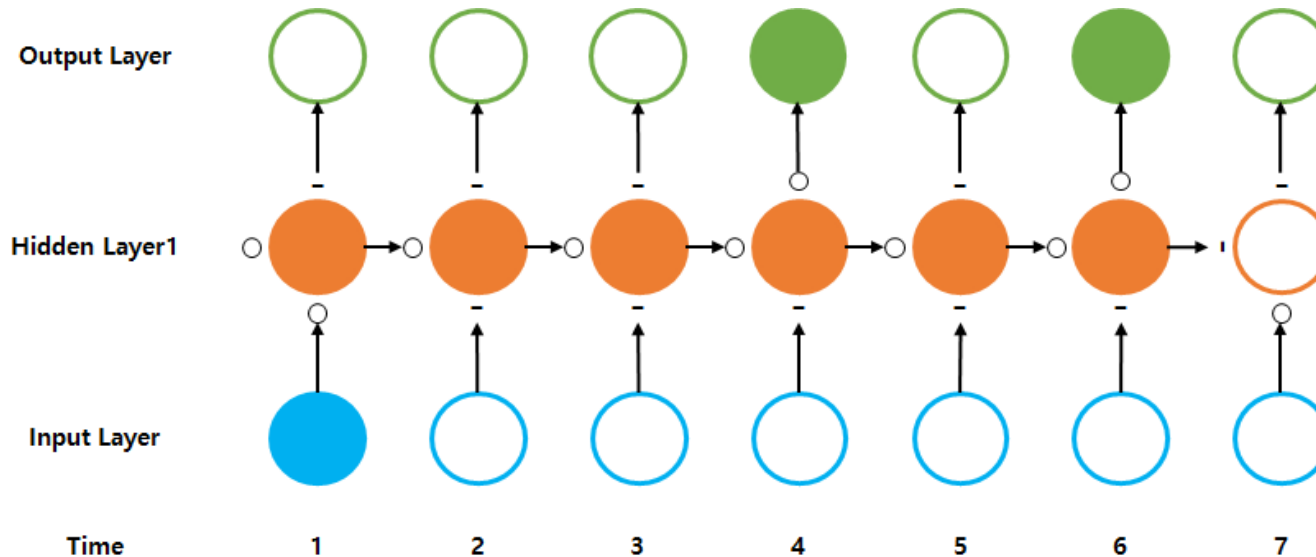


장단기 기억 네트워크(LSTM) 구현

❖ 장단기 기억 네트워크구조

■ 시간에 따라 각 게이트 동작

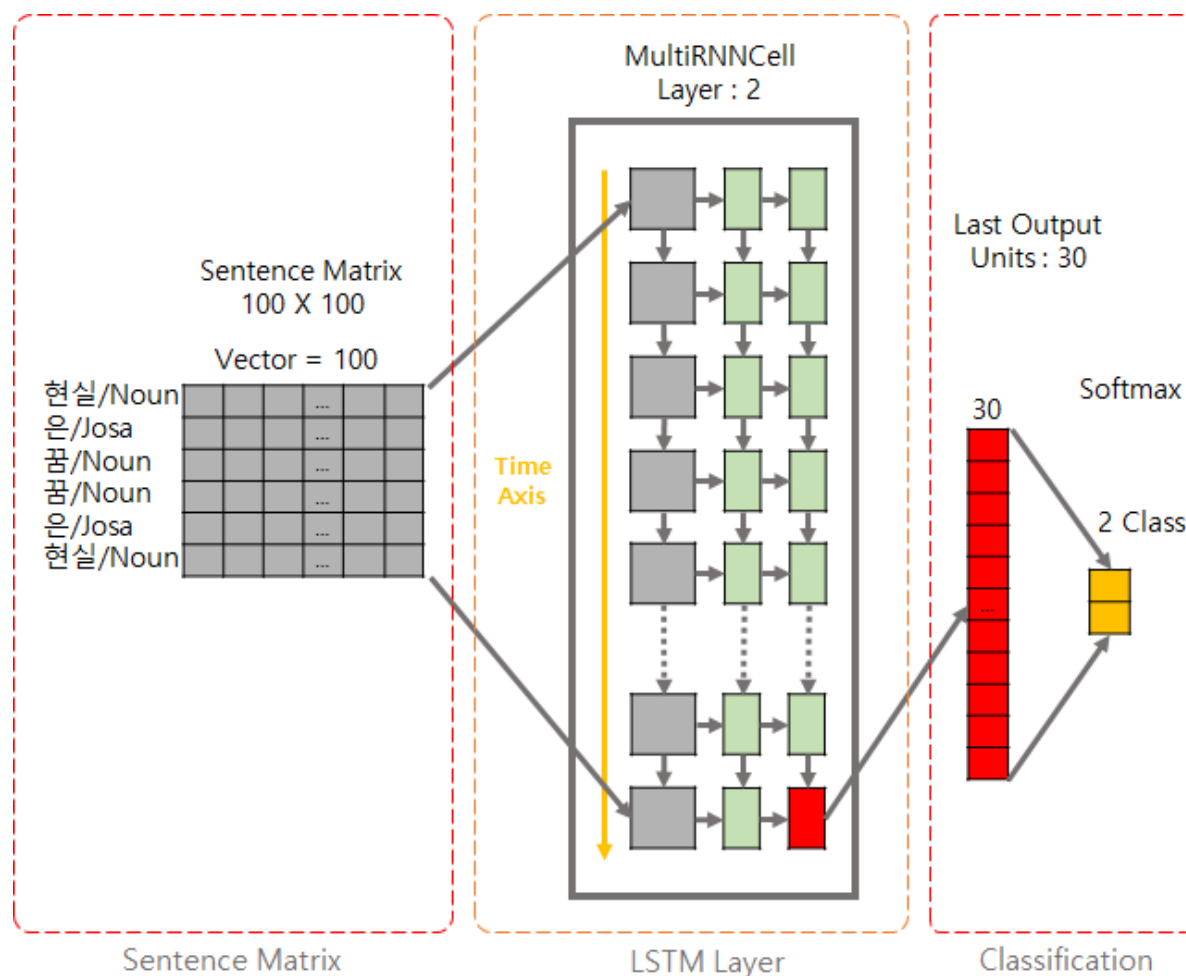
- ✓ 망각, 입력, 출력 게이트를 열고 닫으면서 오랜 시간이 지나더라도 기억을 오랫동안 보존
- ✓ 직선은 닫힌 게이트, 동그라미는 열린 게이트
- ✓ 은닉층의 위, 왼쪽, 아래는 게이트가 출력, 망각, 입력 게이트를 표현
- ✓ 입력층에서는 2~6번째의 시간에서 입력 게이트를 닫음
- ✓ 출력층에서는 4, 6번째 시간에서만 출력 게이트를 열어 경사도 사라짐을 방지



<https://skymind.ai/wiki/lstm/>

장단기 기억 네트워크(LSTM) 구현

❖ 장단기 기억 네트워크 구현 전반적 구조



STREAMLIT

Streamlit

- ❖ <https://streamlit.io/>
- ❖ 데이터 앱을 빠르게 만들고 배포할 수 있는 오픈소스 파이썬 라이브러리
- ❖ 일반적인 컴퓨터
\$ pip install streamlit

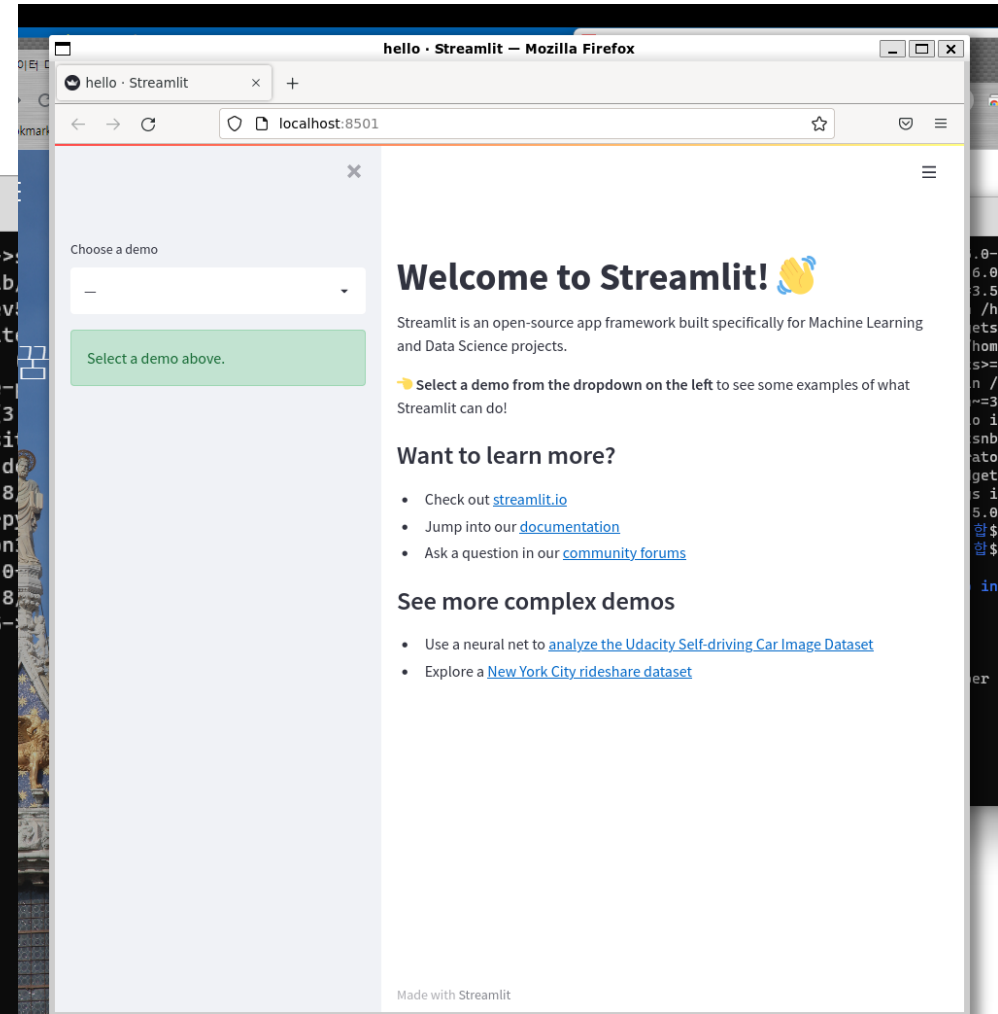
```
oykwon@Ohyoung-K506: /mr x + v
->notebook>=4.4.1->widgetsnextension~=3.5.0->ipywidgets>=7.0.0->pydeck>=0.1.dev5->streamlit
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in /home/oykwon/anaconda3/lib/python3.8/site-packages/nbclient-0.5.1-py3.8.egg/nbclient.egg-info (from notebook==4.4.1)
Requirement already satisfied: testpath in /home/oykwon/anaconda3/lib/python3.8/site-packages/testpath-0.4.4-py3.8.egg/testpath.egg-info (from notebook==4.4.1)
Requirement already satisfied: bleach in /home/oykwon/anaconda3/lib/python3.8/site-packages/bleach-3.1.0-py3.8.egg/bleach.egg-info (from notebook==4.4.1)
Requirement already satisfied: pycparser in /home/oykwon/anaconda3/lib/python3.8/site-packages/pycparser-2.19-py3.8.egg/pycparser.egg-info (from notebook==4.4.1)
Requirement already satisfied: nest-asyncio in /home/oykwon/anaconda3/lib/python3.8/site-packages/nest-asyncio-1.5.1-py3.8.egg/nest_asyncio.egg-info (from notebook==4.4.1)
Requirement already satisfied: nbconvert in /home/oykwon/anaconda3/lib/python3.8/site-packages/nbconvert-6.0.0-py3.8.egg/nbconvert.egg-info (from notebook==4.4.1)
Requirement already satisfied: async-generator in /home/oykwon/anaconda3/lib/python3.8/site-packages/async-generator-1.10-py3.8.egg/async_generator.egg-info (from notebook==4.4.1)
Requirement already satisfied: webencodings in /home/oykwon/anaconda3/lib/python3.8/site-packages/webencodings-0.5.1-py3.8.egg/webencodings.egg-info (from notebook==4.4.1)
(base) oykwon@Ohyoung-K506: /mnt/c/Users/oykwon$ pip install streamlit
(base) oykwon@Ohyoung-K506: /mnt/c/Users/oykwon$ streamlit hello

Welcome to Streamlit. Check out our demo in your browser.

Local URL: http://localhost:8501
Network URL: http://172.24.133.115:8501

Ready to create your own Python apps super quickly?
Head over to https://docs.streamlit.io

May you create awesome apps!
```



Streamlit Demo

❖ Interactive Web App with Streamlit and Scikit-learn (<https://github.com/python-engineer/streamlit-demo>)

Installation

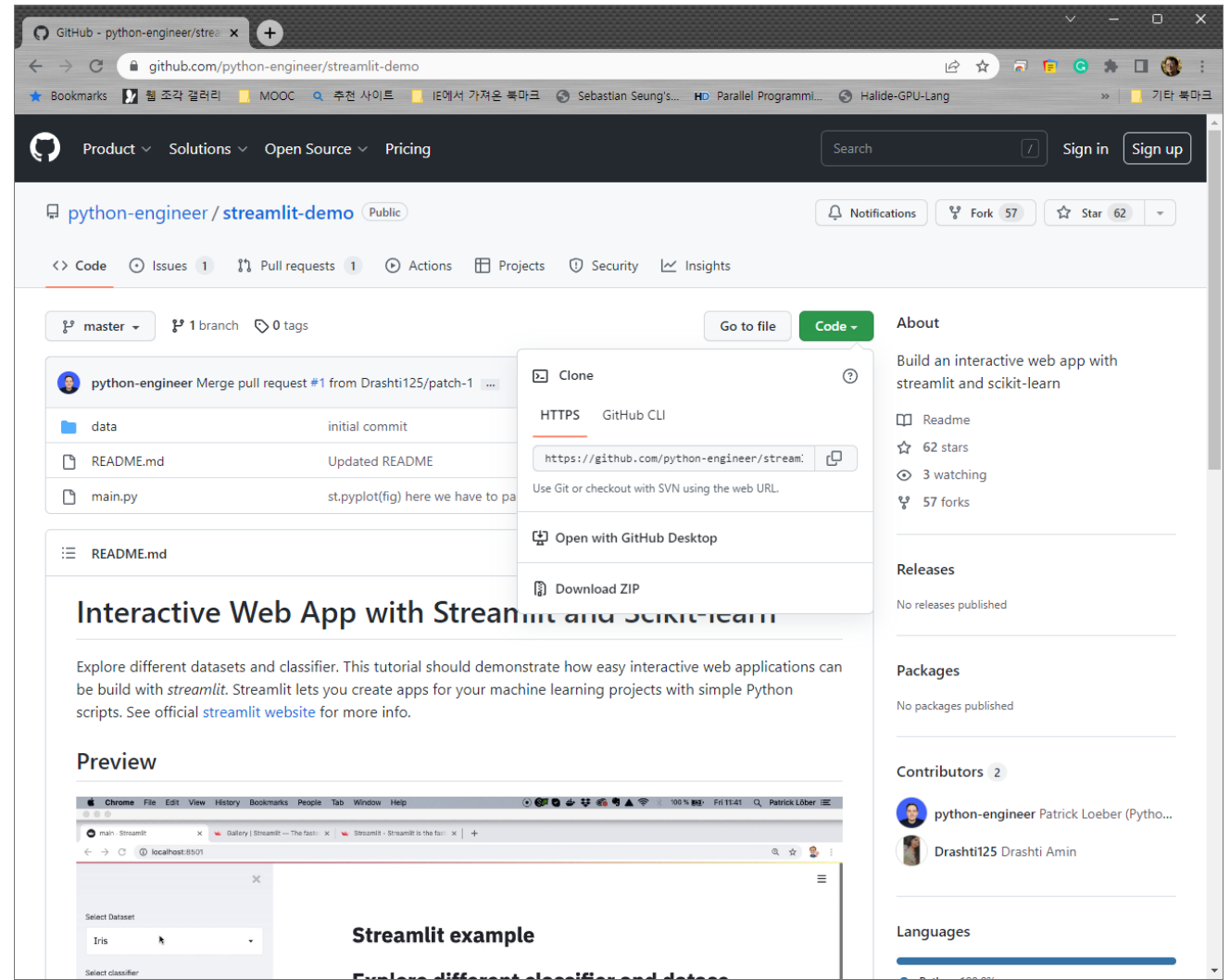
You need these dependencies:

```
pip install streamlit
pip install scikit-learn
pip install matplotlib
```

Usage

Run

```
streamlit run main.py
```



Streamlit Demo

```
(base) oykwon@Ohyoung-K506: ~/codes$ git clone https://github.com/python-engineer/streamlit-demo.git
Cloning into 'streamlit-demo'...
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 16 (delta 4), reused 9 (delta 1), pack-reused 0
Unpacking objects: 100% (16/16), 10.62 MiB | 6.92 MiB/s, done.
(base) oykwon@Ohyoung-K506:~/codes$ cd streamlit-demo/
(base) oykwon@Ohyoung-K506:~/codes/streamlit-demo$ ls
README.md  data  main.py
(base) oykwon@Ohyoung-K506:~/codes/streamlit-demo$ streamlit run main.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://172.24.133.115:8501
```

main · Streamlit — Mozilla Firefox

localhost:8501

Select Dataset

Iris

Select classifier

KNN

K

1

15

Streamlit Example

Explore different classifier and datasets

Which one is the best?

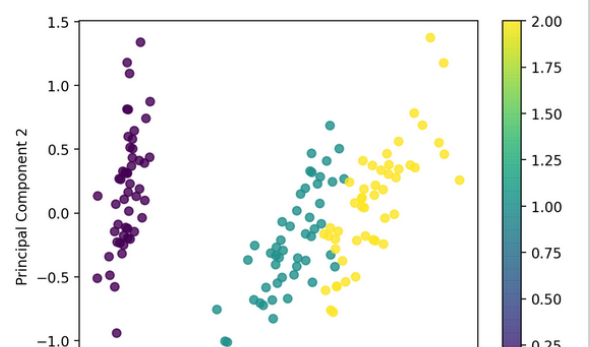
Iris Dataset

Shape of dataset: (150, 4)

number of classes: 3

Classifier = KNN

Accuracy = 1.0



streamlit-demo/main.py at mas

github.com/python-engineer/streamlit-demo/blob/master/main.py

streamlit-demo / main.py / <> Jump to

Drashti125 st.pyplot(fig) here we have to pass a parameter

2 contributors

109 lines (86 sloc) | 2.67 KB

```
1 import streamlit as st
2 import numpy as np
3
4 import matplotlib.pyplot as plt
5 from sklearn import datasets
6 from sklearn.model_selection import train_test_split
7
8 from sklearn.decomposition import PCA
9 from sklearn.svm import SVC
10 from sklearn.neighbors import KNeighborsClassifier
11 from sklearn.ensemble import RandomForestClassifier
12
13 from sklearn.metrics import accuracy_score
14
15 st.title('Streamlit Example')
16
17 st.write("""
18 # Explore different classifier and datasets
19 Which one is the best?
20 """)
21
22 dataset_name = st.sidebar.selectbox(
23     'Select Dataset',
24     ('Iris', 'Breast Cancer', 'Wine')
25 )
26
```

streamlit-demo/main.py at mas


github.com/python-engineer/streamlit-demo/blob/master/main.py

```
27 st.write(f"## {dataset_name} Dataset")
28
29 classifier_name = st.sidebar.selectbox(
30     'Select classifier',
31     ('KNN', 'SVM', 'Random Forest')
32 )
33
34 def get_dataset(name):
35     data = None
36     if name == 'Iris':
37         data = datasets.load_iris()
38     elif name == 'Wine':
39         data = datasets.load_wine()
40     else:
41         data = datasets.load_breast_cancer()
42     X = data.data
43     y = data.target
44     return X, y
45
46 X, y = get_dataset(dataset_name)
47 st.write('Shape of dataset:', X.shape)
48 st.write('number of classes:', len(np.unique(y)))
49
50 def add_parameter_ui(clf_name):
51     params = dict()
52     if clf_name == 'SVM':
53         C = st.sidebar.slider('C', 0.01, 10.0)
54         params['C'] = C
55     elif clf_name == 'KNN':
56         K = st.sidebar.slider('K', 1, 15)
57         params['K'] = K
58     else:
59         max_depth = st.sidebar.slider('max_depth', 2, 15)
60         params['max_depth'] = max_depth
61         n_estimators = st.sidebar.slider('n_estimators', 1, 100)
62         params['n_estimators'] = n_estimators
63
```

```
streamlit-demo/main.py at mas... x
github.com/python-engineer/streamlit-demo/blob/master/main.py
62     params['n_estimators'] = n_estimators
63     return params
64
65     params = add_parameter_ui(classifier_name)
66
67     def get_classifier(clf_name, params):
68         clf = None
69         if clf_name == 'SVM':
70             clf = SVC(C=params['C'])
71         elif clf_name == 'KNN':
72             clf = KNeighborsClassifier(n_neighbors=params['K'])
73         else:
74             clf = RandomForestClassifier(n_estimators=params['n_estimators'],
75                                         max_depth=params['max_depth'], random_state=1234)
76         return clf
77
78     clf = get_classifier(classifier_name, params)
79     ##### CLASSIFICATION #####
80
81     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1234)
82
83     clf.fit(X_train, y_train)
84     y_pred = clf.predict(X_test)
85
86     acc = accuracy_score(y_test, y_pred)
87
88     st.write(f'Classifier = {classifier_name}')
89     st.write(f'Accuracy =', acc)
90
91     ##### PLOT DATASET #####
92     # Project the data onto the 2 primary principal components
93     pca = PCA(2)
94     X_projected = pca.fit_transform(X)
95
96     x1 = X_projected[:, 0]
97     x2 = X_projected[:, 1]
```

```
streamlit-demo/main.py at mas... x
github.com/python-engineer/streamlit-demo/blob/master/main.py
83     clf.fit(X_train, y_train)
84     y_pred = clf.predict(X_test)
85
86     acc = accuracy_score(y_test, y_pred)
87
88     st.write(f'Classifier = {classifier_name}')
89     st.write(f'Accuracy =', acc)
90
91     ##### PLOT DATASET #####
92     # Project the data onto the 2 primary principal components
93     pca = PCA(2)
94     X_projected = pca.fit_transform(X)
95
96     x1 = X_projected[:, 0]
97     x2 = X_projected[:, 1]
98
99     fig = plt.figure()
100     plt.scatter(x1, x2,
101                 c=y, alpha=0.8,
102                 cmap='viridis')
103
104     plt.xlabel('Principal Component 1')
105     plt.ylabel('Principal Component 2')
106     plt.colorbar()
107
108     #plt.show()
109     st.pyplot(fig)
```

[Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

 © 2022 GitHub, Inc.

Streamlit 참고사이트

- ❖ 유튜브 Python Engineer
 - streamlit 검색 상위 4개

