

소프트웨어융합대학

# 누구나 즐기는 딥러닝 오픈소스 Keras와 함께!

이정근

빅데이터 전공주임, 오픈소스SW교육센터장  
소프트웨어 융합 대학 (School of Software)

[JeongGun.Lee@hallym.ac.kr](mailto:JeongGun.Lee@hallym.ac.kr) / [www.onchip.net](http://www.onchip.net)

한림대학교 소프트웨어중심대학

<http://hls.w.hallym.ac.kr>

소프트웨어융합대학



# 개나 소나 딥러닝 오픈소스 Keras와 함께!



이정근

빅데이터 전공주임, 오픈소스SW교육센터장  
소프트웨어 융합 대학 (School of Software)

[JeongGun.Lee@hallym.ac.kr](mailto:JeongGun.Lee@hallym.ac.kr) / [www.onchip.net](http://www.onchip.net)

한림대학교 소프트웨어중심대학

<http://hls.w.hallym.ac.kr>



# 인공지능은 무엇인가 ? (정의/역사)

- ↳ 딥러닝 모델
- ↳ Keras ?
- ↳ Keras로 데이터 분석하기!
- ↳ Keras로 이미지 인식하기!

`import keras`



# 인공지능은 무엇인가? (정의/역사)



## 인공지능 ?



**알파고**(영어: AlphaGo)는 구글(Google)의 딥마인드(DeepMind Technologies Limited)가 개발한 인공지능(AI, Artificial Intelligence) 바둑프로그램이다. Alphago Lee는 **1,920개의 CPUs 와 280개의 GPUs**로 구성.

## 인공지능 ?



아이언맨 - 자비스

## 인공지능 ?

**똑똑해지는 인공지능(AI) 스피커** ※연내 도입 예정 기능도 포함

공통 기능 🎵 음악 재생 ☁ 날씨 안내 🔍 정보 검색 💬 간단한 대화 등

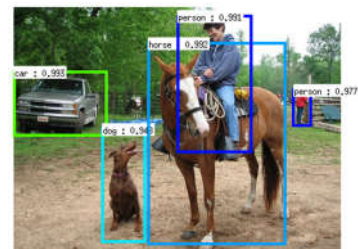
				
1 KT 기가지니	2 SK텔레콤 누구	3 카카오미니	4 네이버 클로바	5 LG유플러스 프렌즈 플러스
추가 음성 생체인증 기능	아파트관리비 조회	다국어 지원	음성 통화	매장 도우미
방식 음성 인식해 간편 결제	음성으로 관리비 내역 확인	영어·중국어· 일본어 음성 명령, 한국어 번역 지원	모바일 메신저 '라인'과 연동한 인터넷 전화	휴대폰 유통점서 요금제 등 안내
				자료=각 사

## 인공지능: 정의

人工知能 / **Artificial Intelligence; A.I.**

인공지능은 인간이 지닌 **지적 능력의 일부 또는 전체를 인공적으로 구현한 것...**

... SF물에서 흔히 볼 수 있는 소재 ...



## 인공지능: 정의

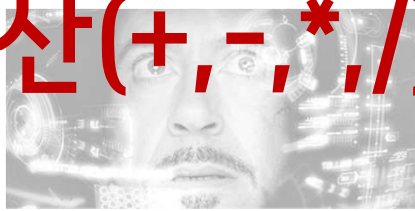
人工知能 / Artificial Intelligence; A.I.

인공지능은 인간이 지닌 **지적 능력의 일부 또는 전체를 인공적으로 구현한 것...**

... SF물에서 흔히 볼 수 있는 소재 ...



연산(+, -, \*, /)



## 인공지능: 정의

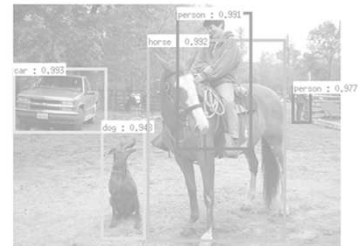
人工知能 / Artificial Intelligence; A.I.

인공지능은 인간이 지닌 **지적 능력의 일부 또는 전체를 인공적으로 구현한 것...**

... SF물에서 흔히 볼 수 있는 소재 ...



인식 또는 예측



## 인공지능:기계학습: 정의

기계학습 / **Machine Learning; ML**

Field of study that gives computers the ability to learn **without being explicitly programmed.**

- Arthur Lee Samuel, 1959

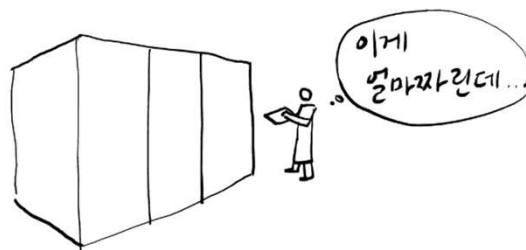


# 인식 또는 예측



## 인공지능: 역사

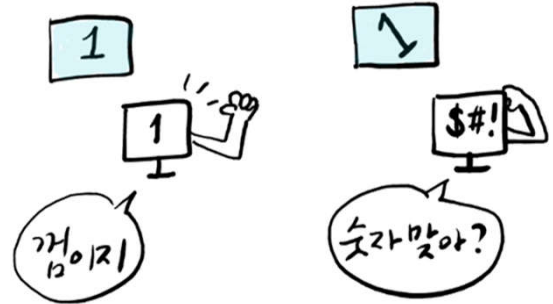
인간은 컴퓨터를 발명해서  
어마어마한 계산을 할 수 있었지만  
그림을 인식한다는 것은  
여전히 요원한 일이었습니다.



From <https://brunch.co.kr>

## 인공지능: 역사

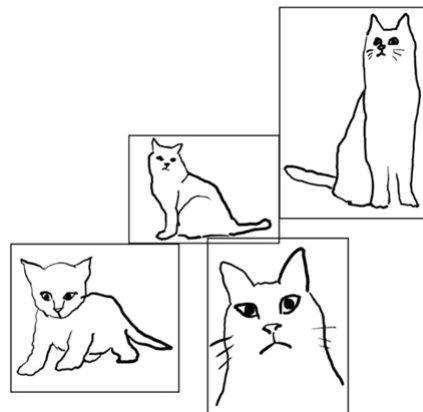
손글씨 숫자를 인식하는 데 있어  
사람은 너무나도 당연하게 해내지만  
손글씨를 픽셀로 받아들이고  
결국에는 매트릭스로 인식하는 컴퓨터는  
조금만 기울여쓰거나 약간만 왜곡시켜도  
숫자를 인식하지 못했습니다.



From <https://brunch.co.kr>

## 인공지능: 역사

하물며 이미지상의  
여러 형태의  
고양이를 인식한다는 건  
거의  
불가능에 가까운  
일이었습니다.



From <https://brunch.co.kr>



## 인공지능: 역사

이제부터 계산능력이 아닌

지능을 가진 기계를 꿈꾸기 시작합니다.

인간의 뇌에 대한

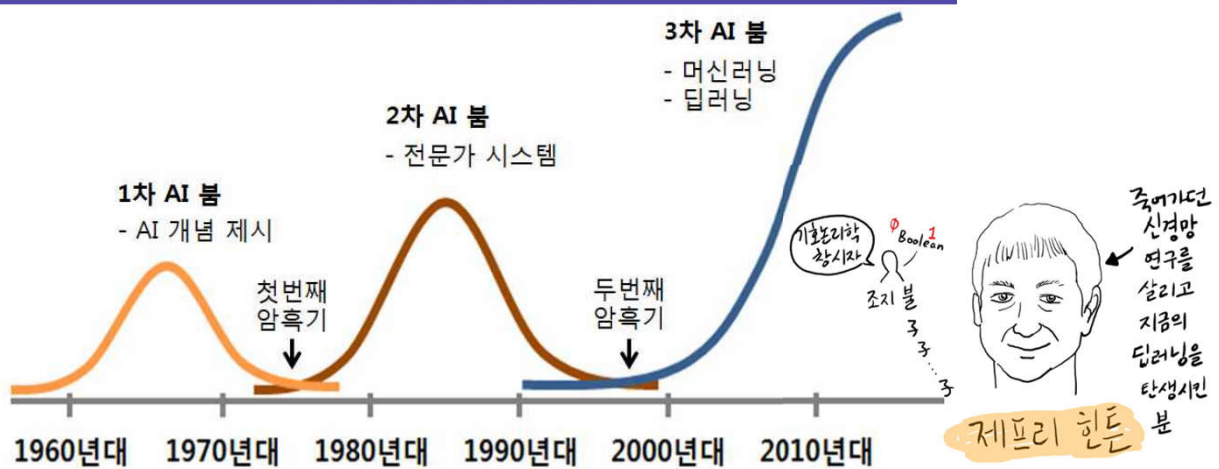
연구가 시작되어 집니다



From <https://brunch.co.kr>

## 인공지능: 역사 - 차가운 겨울

### 인공지능의 발전 과정





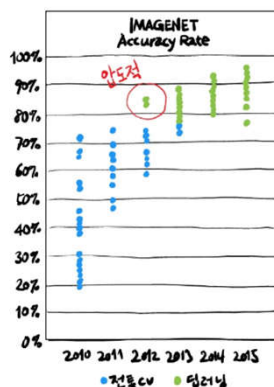
## 인공지능: 역사 – 이미지넷



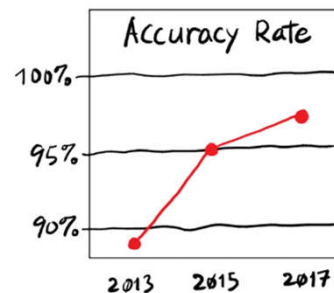
From <https://brunch.co.kr>

## 인공지능: 역사 – 현재 ...

2012년 IMAGENET 대회에서  
제프리 힌튼의 슈퍼비전팀이  
딥러닝 기술로 압도적 우승을 합니다.



이렇게 등장한 딥러닝은  
해를 거듭함에 따라 더욱 발전하여  
2017년에는 **에러율 3%**에  
도달하여 이제 인간의 능력을  
능가하는 수준까지 왔습니다.



From <https://brunch.co.kr>

## 딥러닝 모델

### 인공지능: 딥러닝

면접관 : 당신의 강점은 무엇이죠?

나 : 저는 머신러닝에 대해 잘 압니다.

면접관 :  $9 + 10$ 은?

나 : 3이요

면접관 : 비슷하지도 않군요. 답은 19입니다.

나 : 16이요

면접관 : 아니오, 그냥 19라고요.

나 : 18이요.

면접관 : 아니, 19라고.

나 : 19요.

면접관 : 채용하겠습니다.

## 인공지능: 딥러닝

면접관 : 당신의 강점은 무엇이죠?

나 : 저는 머신러닝에 대해 잘 압니다.

면접관 :  $9 + 10$ 은?

나 : 3이요

면접관 : 비슷하지도 않군요. 답은 19입니다.

나 : 16이요

면접관 : 아니오, 그냥 19라고요.

나 : 18이요.

면접관 : 아니, 19라고.

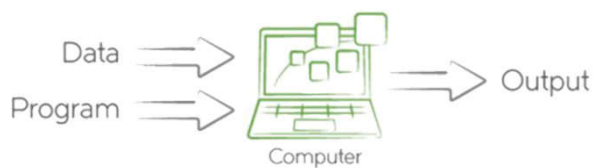
나 : 19요.

면접관 : 채용하겠습니다.

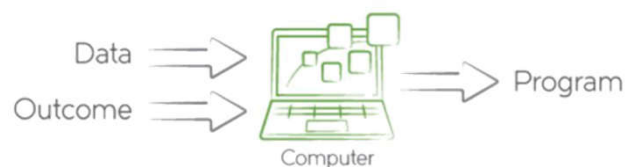


## 인공지능: 딥러닝

### Traditional Programming

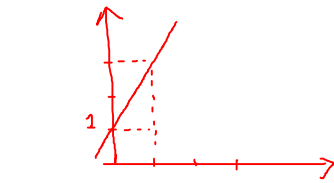
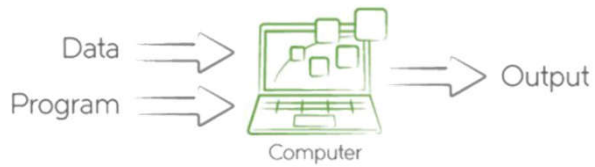


### Machine Learning



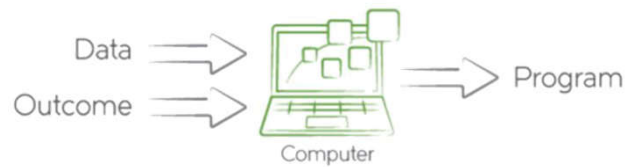
# 인공지능: 딥러닝

## Traditional Programming



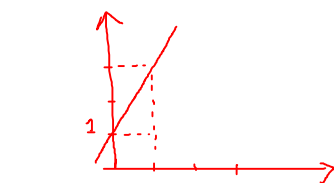
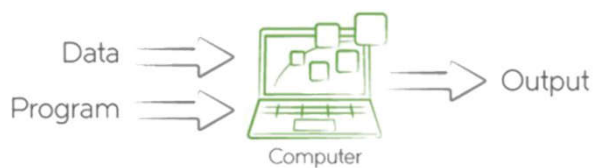
$$\text{Output} = \text{Data} * 2 + 1;$$

## Machine Learning



# 인공지능: 딥러닝

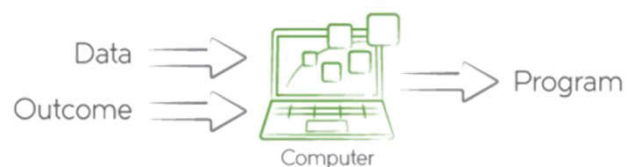
## Traditional Programming



$$\text{Output} = \text{Data} * 2 + 1;$$

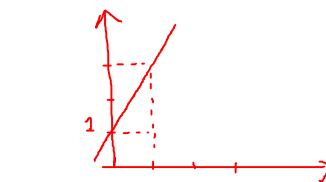
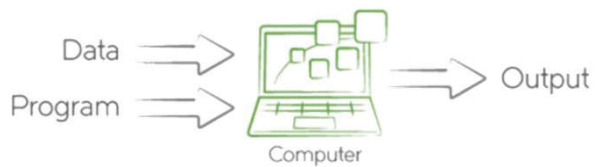
## Machine Learning

(Data, Outcome)  
= {1, 3}, {2, 5}, {3, 7} ...



# 인공지능: 딥러닝

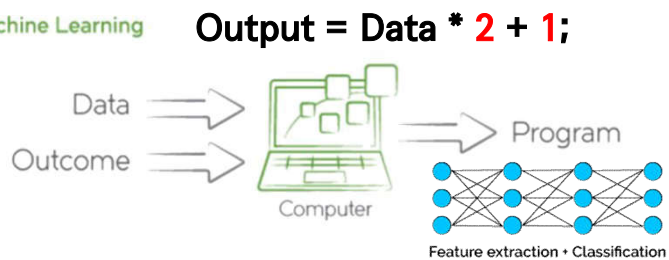
## Traditional Programming



$$\text{Output} = \text{Data} * 2 + 1;$$

## Machine Learning

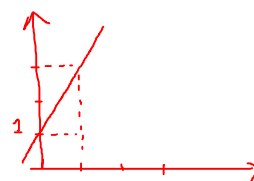
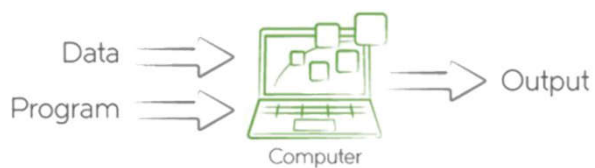
(Data, Outcome)  
= {1, 3}, {2, 5}, {3, 7} ...



$$\text{Output} = \text{Data} * 2 + 1;$$

# 인공지능: 딥러닝

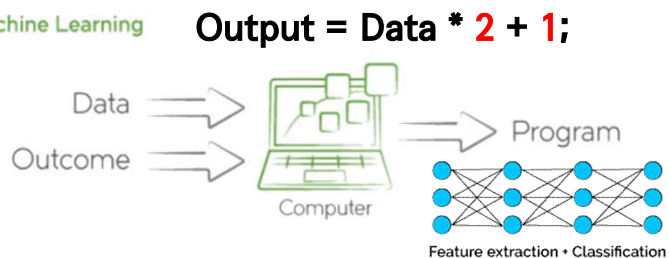
## Traditional Programming



$$\text{Output} = \text{Data} * 2 + 1;$$

## Machine Learning

(Data, Outcome)  
= {1, 3}, {2, 5}, {3, 7} ...



$$\text{Output} = \text{Data} * 2 + 1;$$

# 인공지능: 딥러닝

Machine Learning



$$\text{Output} = \text{Data} * \mathbf{w} + \mathbf{b};$$

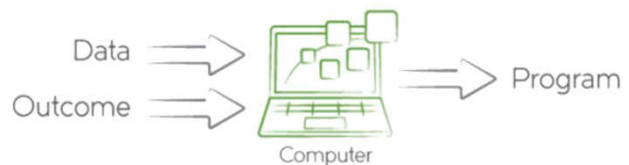
(Data, Outcome)  
= {1, 3}, {2, 5}, {3, 7} ...

$$w = 1, b = 1 \rightarrow \{1, \mathbf{2}\}, \{2, \mathbf{3}\}, \{3, \mathbf{4}\}$$

Error:     1            2            3

# 인공지능: 딥러닝

Machine Learning



$$\text{Output} = \text{Data} * \mathbf{w} + \mathbf{b};$$

(Data, Outcome)  
= {1, 3}, {2, 5}, {3, 7} ...

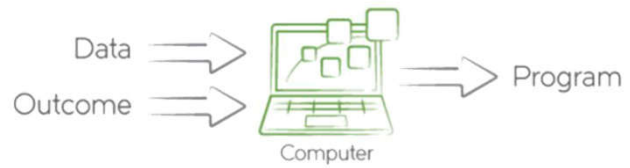
$$w = 1, b = 1 \rightarrow \{1, \mathbf{2}\}, \{2, \mathbf{3}\}, \{3, \mathbf{4}\}$$

update?

Error:     1            2            3

# 인공지능: 딥러닝

Machine Learning



$$\text{Output} = \text{Data} * \mathbf{w} + \mathbf{b};$$

(Data, Outcome)

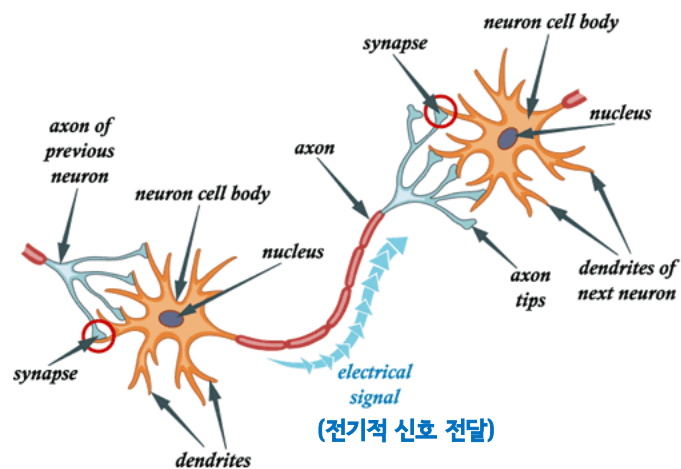
= {1, 3}, {2, 5}, {3, 7} ...

$w = 1.5, b = 1 \rightarrow \{1, 2\}, \{2, 3\}, \{3, 4\}$

update!

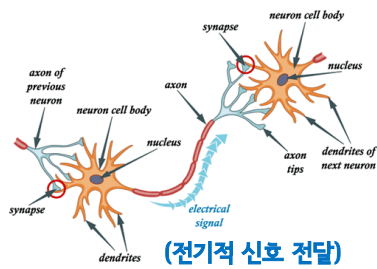
Error: 0.5 1 1.5

# 인공지능: 딥러닝

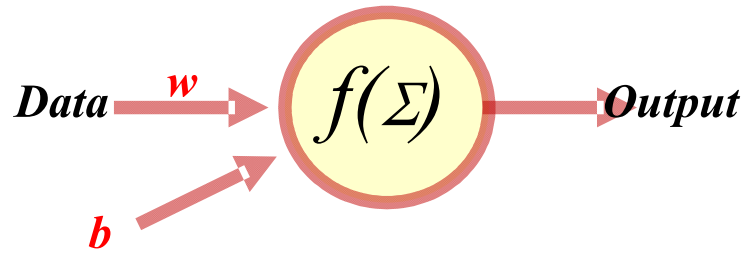




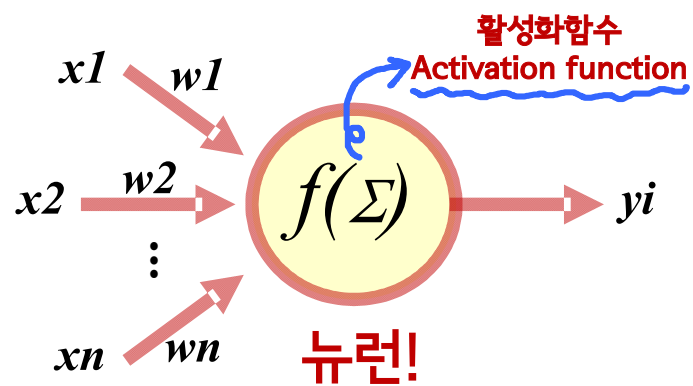
## 인공지능: 딥러닝



$$\text{Output} = f(\text{Data} * \mathbf{w} + \mathbf{b});$$

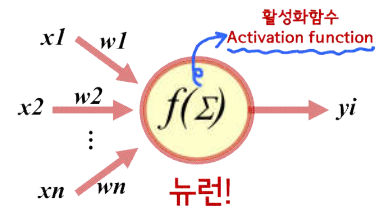


## 인공지능: 딥러닝



$$\begin{aligned} y_i &= f(x_1, x_2, \dots, x_n) \\ &= \max(0, w_1 * x_1 + w_2 * x_2 + \dots + w_n * x_n) \end{aligned}$$

# 인공지능: 딥러닝



## 활성화 함수의 역할

→ 신경망 모델에 **비선형성 (Non-Linearity)** 제공

$$y_i = f(x_1, x_2, \dots, x_n)$$

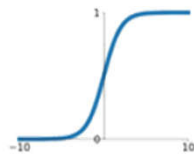
$$= \max(0, w_1 * x_1 + w_2 * x_2 + \dots + w_n * x_n)$$

# 인공지능: 딥러닝

> 활성화 함수 (Activation function)

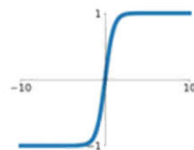
**Sigmoid**

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



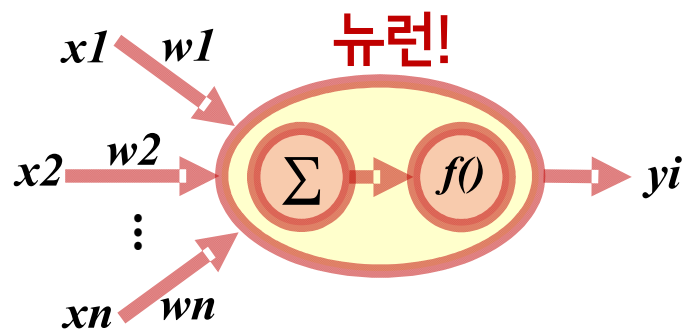
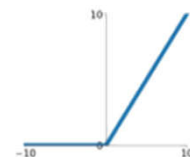
**tanh**

$$\tanh(x)$$

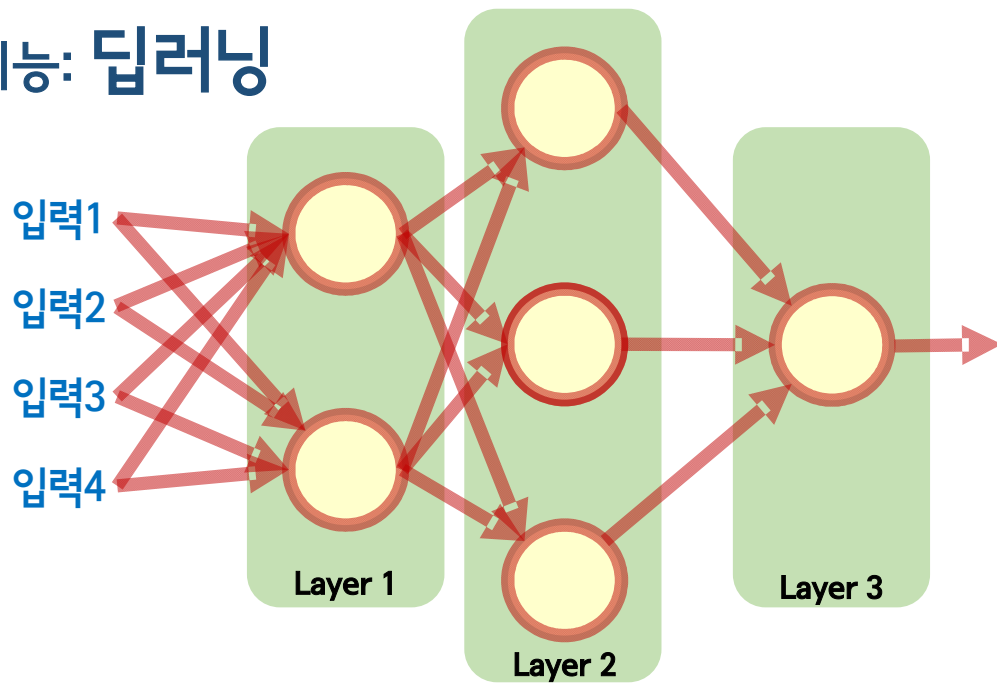


**ReLU**

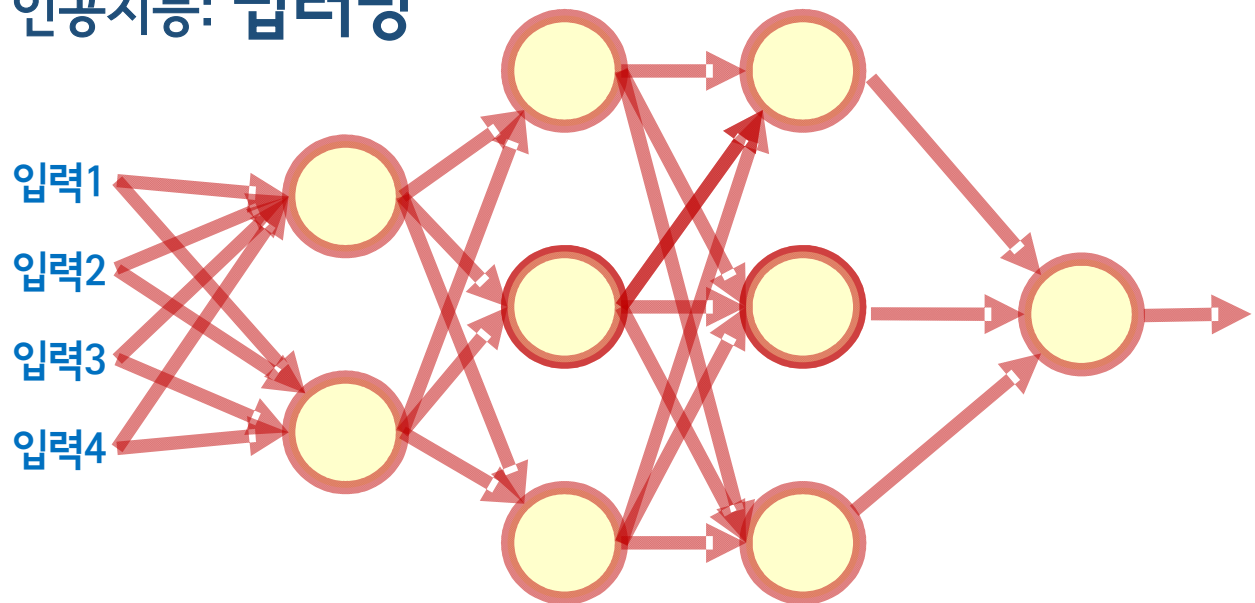
$$\max(0, x)$$



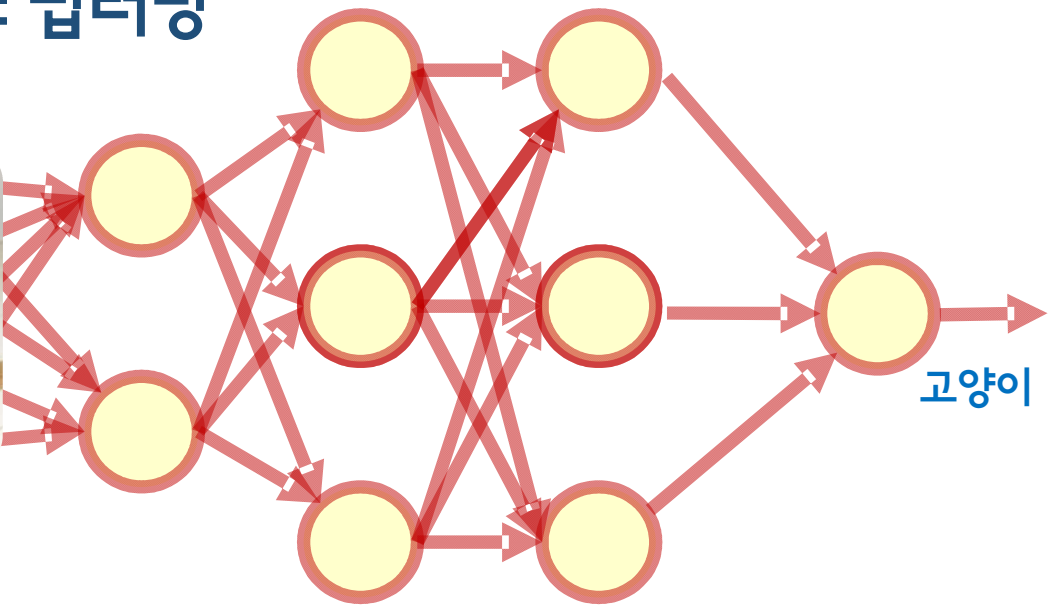
## 인공지능: 딥러닝



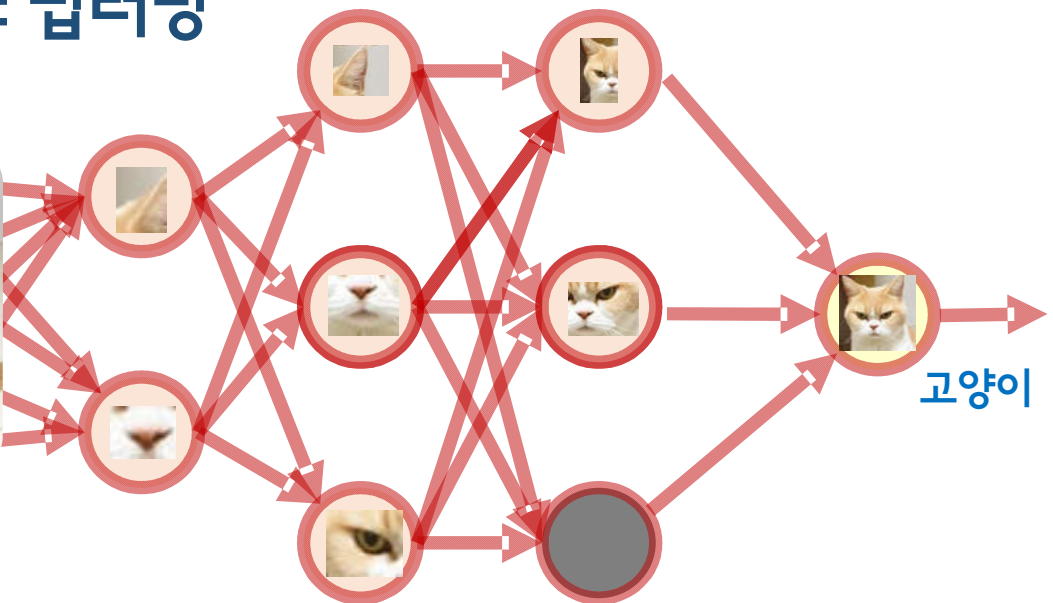
## 인공지능: 딥러닝



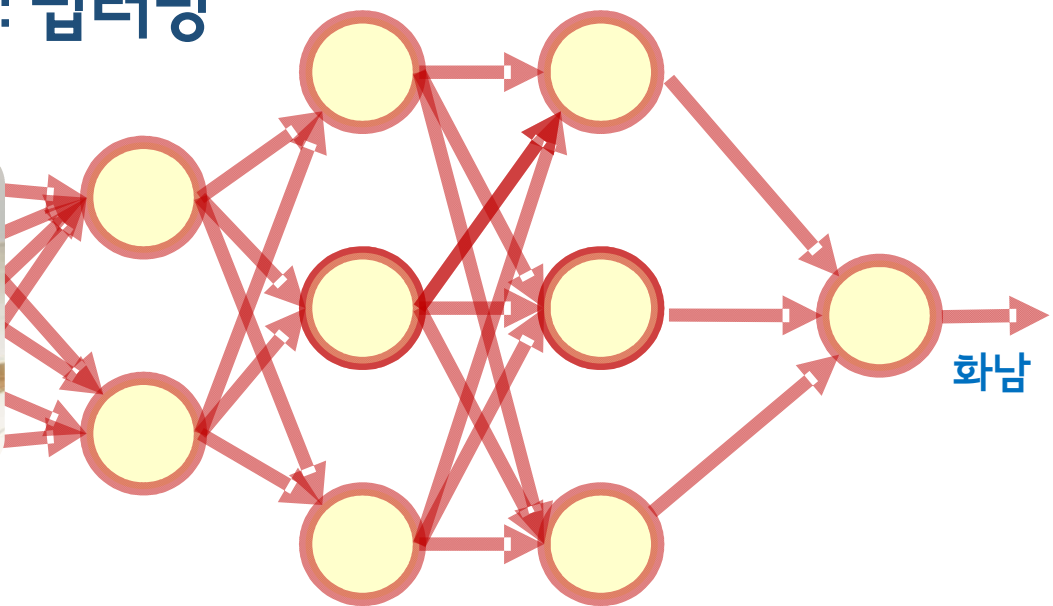
## 인공지능: 딥러닝



## 인공지능: 딥러닝

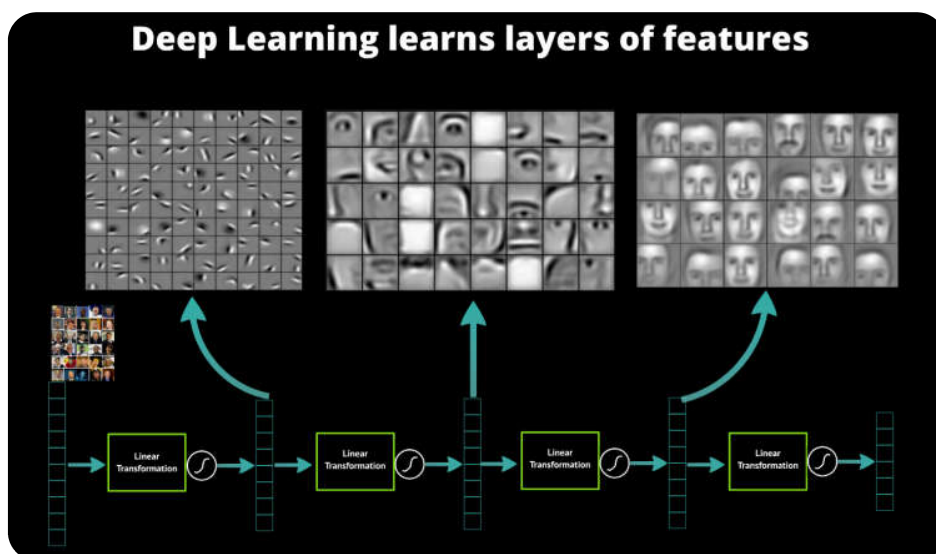


## 인공지능: 딥러닝

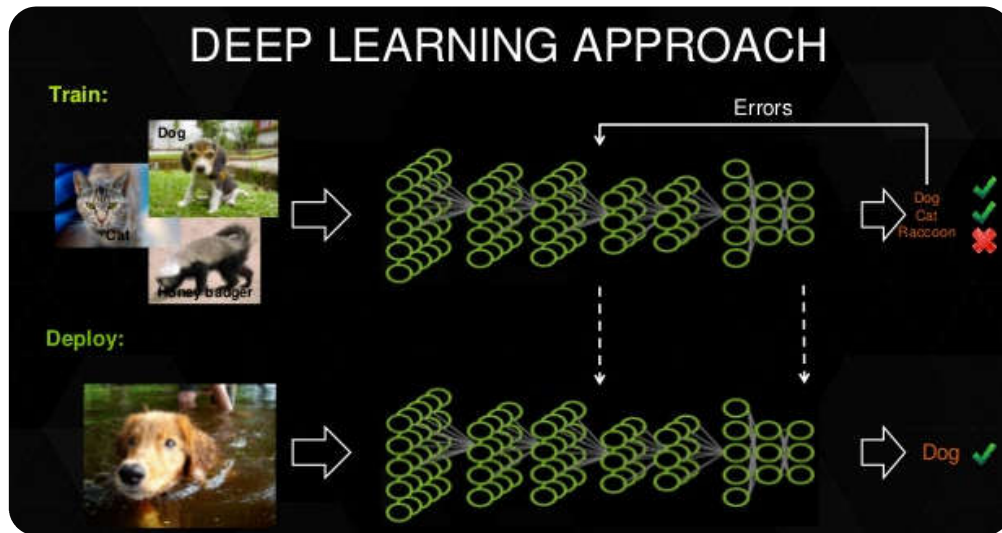


## 인공지능: 딥러닝

### Deep Learning learns layers of features



## 인공지능: 딥러닝



1부 끝 ~

피곤한가요 ?



Keras (케라스):



**딥러닝 라이브러리**

> Tensorflow와 Theano를 Backend로 사용



**특징?**

Keras (케라스)



**딥러닝 라이브러리**

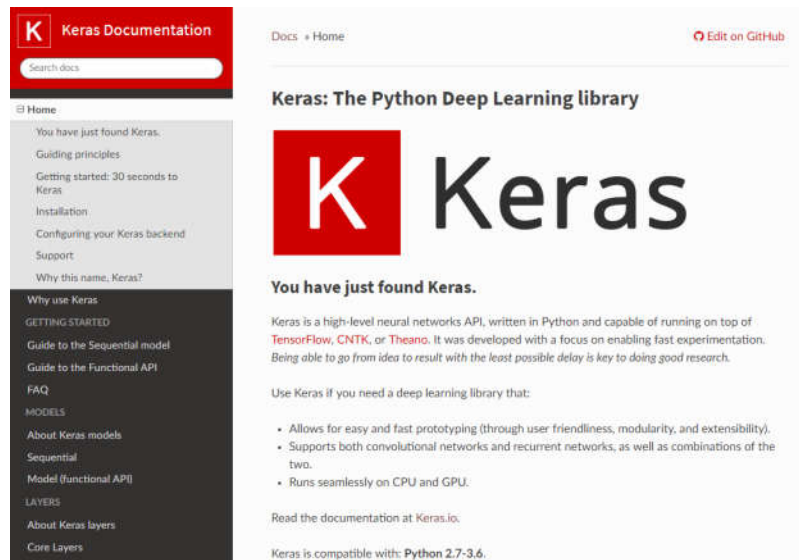
> Tensorflow와 Theano를 Backend로 사용

**특징?**

- > 쉽다!
- > 쉬우니까 빠르게 구현할 수 있다.
- > CPU / GPU 모두 지원!
- > Python 언어 기반



## Keras (케라스)- <https://keras.io/>



## Keras (케라스): 실습

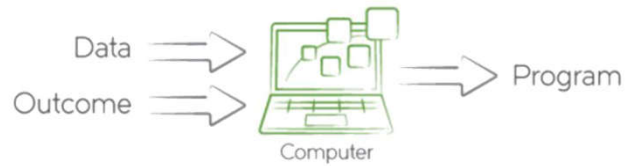


<https://colab.research.google.com>

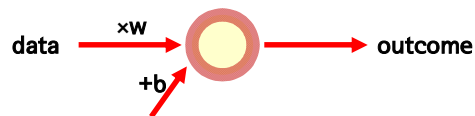
## Getting started: 30 seconds to Keras

Machine Learning

(Data, Outcome)  
= {1, 3}, {2, 5}, {3, 7} ...



data = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
outcome = [3, 5, 7, 9, 11, 13, 15, 17, 19, 21]



## Getting started: 30 seconds to Keras

```
from keras.models import Sequential
from keras.layers import Dense
```

라이브러리  
가져오기

```
model = Sequential()
model.add(Dense(1, input_dim=1))
```

딥러닝  
모델 만들기

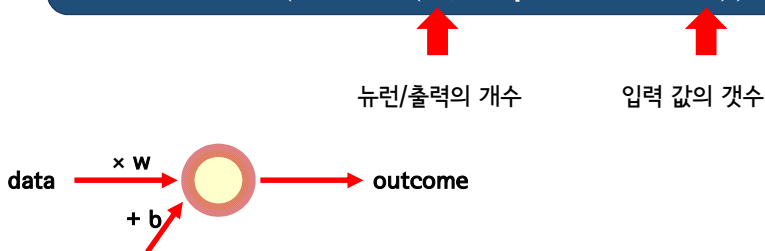
## Getting started: 30 seconds to Keras

```
from keras.models import Sequential
from keras.layers import Dense
```

라이브러리  
가져오기

```
model = Sequential()
model.add(Dense(1, input_dim=1))
```

딥러닝  
모델 만들기



## Getting started: 30 seconds to Keras

```
from keras.models import Sequential
from keras.layers import Dense
```

라이브러리  
가져오기

```
model = Sequential()
model.add(Dense(1, input_dim=1))
```

딥러닝  
모델 만들기

```
model.compile(loss='mean_squared_error', optimizer='adam')
```

## Getting started: 30 seconds to Keras

```
from keras.models import Sequential
from keras.layers import Dense
```

라이브러리  
가져오기

```
model = Sequential()
model.add(Dense(1, input_dim=1))
```

딥러닝  
모델 만들기

```
model.compile(loss='mean_squared_error', optimizer='adam')
```

```
model.fit(data, outcome, epochs=1000, batch_size=5)
```

학습

## Getting started: 30 seconds to Keras

```
model = Sequential()
model.add(Dense(1, input_dim=1))
```

딥러닝  
모델 만들기

```
model.compile(loss='mean_squared_error', optimizer='adam')
```

```
model.fit(data, outcome, epochs=1000, batch_size=5)
```

학습

```
model.predict([20])
```

예측

## Keras 딥러닝: 모델 구성

```
model = Sequential()
model.add(Dense(1, input_dim=1))
```

딥러닝  
모델 만들기

model = Sequential()

> 딥러닝 모델을 입력 단에서 부터 “순차적”으로 연결하여 구성함!

참조

- <https://keras.io/layers/core/>
- [https://tykimos.github.io/2017/01/27/MLP\\_Layer\\_Talk/](https://tykimos.github.io/2017/01/27/MLP_Layer_Talk/)

## Keras 딥러닝: 모델 구성

```
model = Sequential()
model.add(Dense(1, input_dim=1))
```

딥러닝  
모델 만들기

model.add( Dense(1, input\_dim=1))

> 딥러닝 모델을 입력 단에서 부터 “순차적”으로 추가/add !

참조

- <https://keras.io/layers/core/>
- [https://tykimos.github.io/2017/01/27/MLP\\_Layer\\_Talk/](https://tykimos.github.io/2017/01/27/MLP_Layer_Talk/)

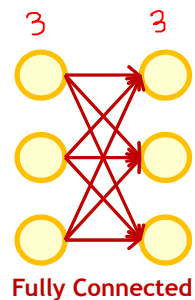
## Keras 딥러닝: 모델 구성

```
model = Sequential()
model.add(Dense(1, input_dim=1))
```

딥러닝  
모델 만들기

```
model.add( Dense(1, input_dim=1))
```

- > Dense는 층의 속성으로 Fully Connected를 의미
- > 첫번째 인자는 층에 포함된 뉴런의 수
- > “input\_dim=1”은 입력의 수를 의미



참조

- <https://keras.io/layers/core/>
- [https://tykimos.github.io/2017/01/27/MLP\\_Layer\\_Talk/](https://tykimos.github.io/2017/01/27/MLP_Layer_Talk/)

## Keras 딥러닝: 모델 컴파일

```
model.compile(loss='mean_squared_error', optimizer='adam')
```

```
model.compile( ... )
```

- > loss: 실제 출력값과 모델을 통해서 얻은 출력값의 상이 수준
  - mean\_squared\_error
  - ...
- > optimizer:
  - adam
  - ...

## Keras 딥러닝: 모델 컴파일

```
model.compile(loss='mean_squared_error', optimizer='adam')
```

loss = 정답과 예상치의 차이

면접관 : 당신의 강점은 무엇이죠?  
나 : 저는 머신러닝에 대해 잘 압니다.  
면접관 : 9 + 10은?  
나 : 3이요  
면접관 : 비슷하지도 않군요. 답은 19입니다.  
나 : 16이요  
면접관 : 아니오, 그냥 19라고요.  
나 : 18이요.  
면접관 : 아니, 19라고.  
나 : 19요.  
면접관 : 채용하겠습니다.



학습

loss(16) = 정답(19) - 예상치(3)

loss(3) = 정답(19) - 예상치(16)

loss(1) = 정답(19) - 예상치(18)

loss(0) = 정답(19) - 예상치(19)

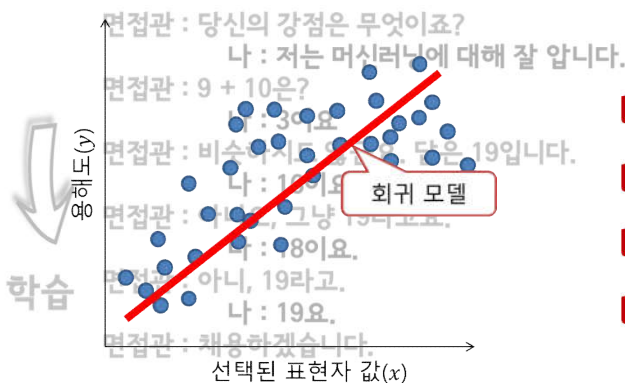
mean\_absolute\_error

> <https://keras.io/losses/>

## Keras 딥러닝: 모델 컴파일

```
model.compile(loss='mean_squared_error', optimizer='adam')
```

loss = 정답과 예상치의 차이



$$MSE = \frac{1}{n} \sum_{i=1}^n (\hat{Y}_i - Y_i)^2$$

loss(256) = (정답(19) - 예상치(3))<sup>2</sup>

loss(9) = (정답(19) - 예상치(16))<sup>2</sup>

loss(1) = (정답(19) - 예상치(18))<sup>2</sup>

loss(0) = (정답(19) - 예상치(19))<sup>2</sup>



## Keras 딥러닝: 모델 컴파일

```
model.compile(loss='mean_squared_error', optimizer='adam')
```

**model.compile( ... )**

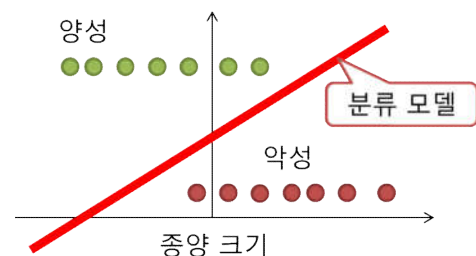
- > loss: 실제 출력값과 모델을 통해서 얻은 출력값의 상이 수준
  - mean\_squared\_error (연속적인 출력값의 예측)
  - binary\_crossentropy (이진 분류, YES/NO)
  - categorical\_crossentropy (다중 분류, 어디에 속하나?)
  - ...
- > optimizer:

## Keras 딥러닝: 모델 컴파일

```
model.compile(loss='binary_crossentropy', optimizer='adam')
```

**binary\_crossentropy**

- > logistic regression
- > 샘플을 True 또는 False로 분류









> <https://keras.io/losses/>

## Keras 딥러닝: 모델 컴파일

```
model.compile(loss='categorical_crossentropy', optimizer='adam')
```

### categorical\_crossentropy

Multi-Class

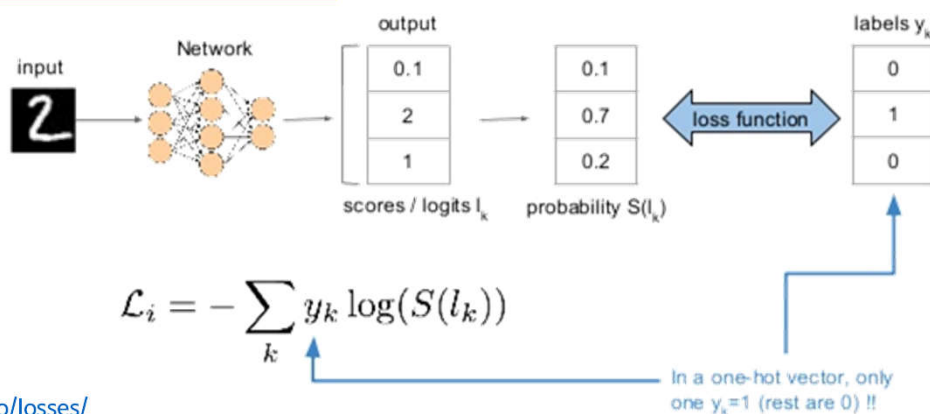
C = 3	Samples		
  			
	Labels (t)		
	[0 0 1]	[1 0 0]	[0 1 0]

> <https://keras.io/losses/>

## Keras 딥러닝: 모델 컴파일

```
model.compile(loss='categorical_crossentropy', optimizer='adam')
```

### categorical\_crossentropy



> <https://keras.io/losses/>

## Keras 딥러닝: 모델 컴파일

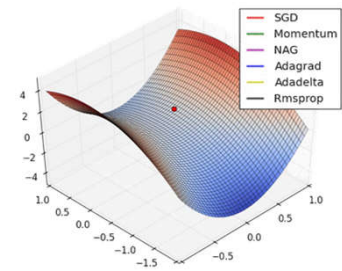
```
model.compile(loss='mean_squared_error', optimizer='adam')
```

**model.compile( ... )**

> **loss**: 실제 출력값과 모델을 통해서 얻은 출력값의 상이 수준

> **optimizer**: 최적의 weight를 빠르게 찾기

- adam
- sgd
- rmsprop
- adagrad
- ...



## Keras 딥러닝: 모델 학습

```
model.fit(data, outcome, epochs=1000, batch_size=5) 학습
```

**model.fit(입력, 출력, epochs, batch\_size ...)**

> **epochs** (시대, 한세대, ...):

- !

> **batch\_size**: ...

- !!



## Keras 딥러닝: 모델 학습

`model.fit(data, outcome, epochs=1000, batch_size=5)` **학습**


`model.fit(입력, 출력, epochs, batch_size ...)`

> epochs:


- !

> batch\_size:


- !!

**Epoch** 

An Epoch represents one iteration over the entire dataset.

**Batch** 

We cannot pass the entire dataset into the neural network at once. So, we divide the dataset into number of batches.

**Iteration** 

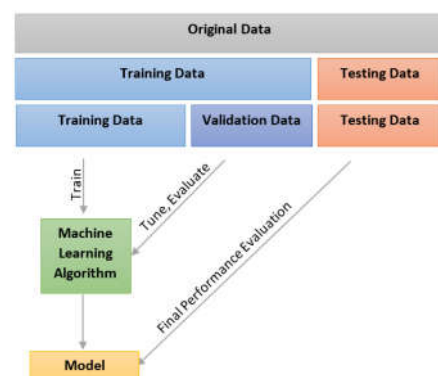
If we have 10,000 images as data and a batch size of 200, then an epoch should contain  $10,000/200 = 50$  iterations.

## Keras 딥러닝: 모델 기반 예측

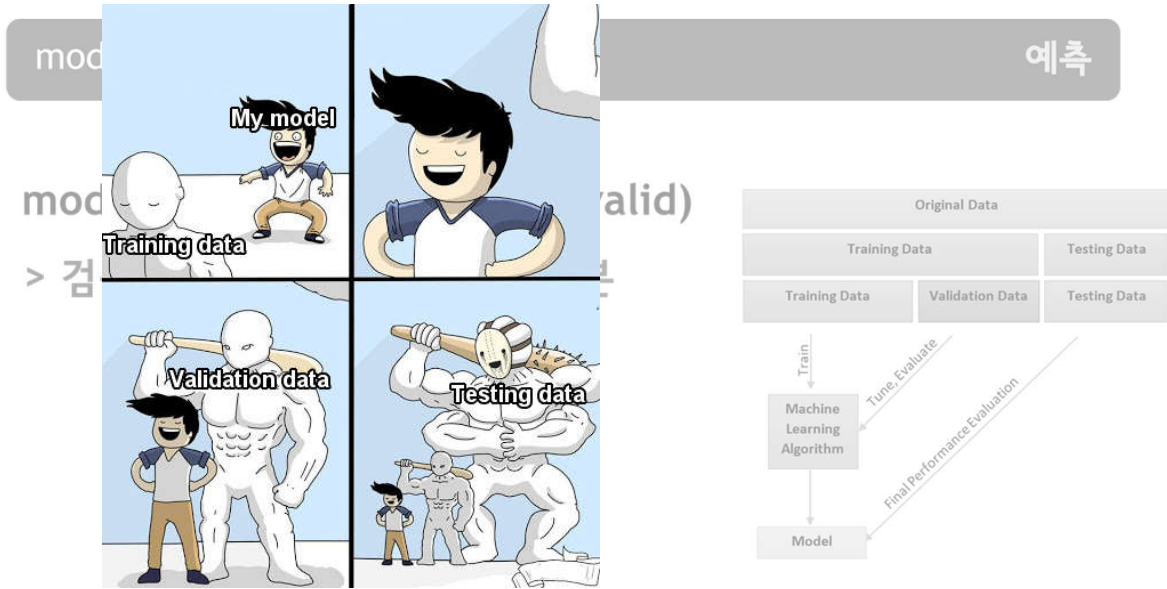
`model.predict([20])` **예측**

`model.evaluate(x_valid, y_valid)`

> 검증 데이터: 학습데이터와 구분



## Keras 딥러닝: 모델 기반 예측



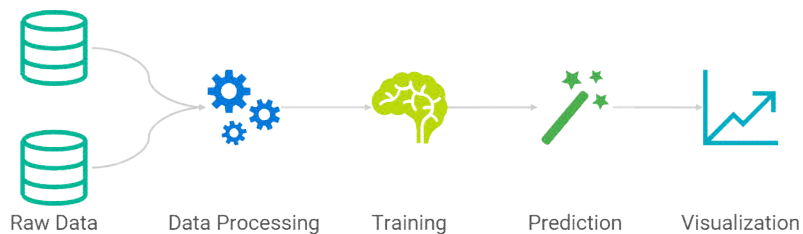
## Keras 딥러닝: 모델 기반 예측

```
model.predict([20])
```

예측

`model.predict( 테스트 데이터 )`

> 테스트 데이터: 학습에 사용하지 않은 데이터 사용



## Keras 딥러닝: AND 논리 함수

x1	x2	out
0	0	0
0	1	0
1	0	0
1	1	1

## Keras 딥러닝: AND 논리 함수

```
from keras.models import Sequential
from keras.layers import Dense
import numpy as np
```

```
data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
outcome = np.array([[0], [0], [0], [1]])
```

```
model = Sequential()
model.add(Dense(1, input_dim=2))
```



딥러닝  
모델

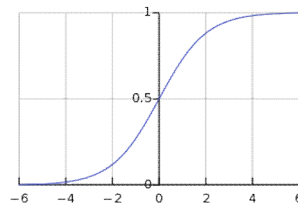
```
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(data, outcome, epochs=1000, batch_size=5)
```

## Keras 딥러닝: AND 논리 함수

```
model = Sequential()
model.add(Dense(1, input_dim=2))
```

### > Sigmoid 활성 함수 사용

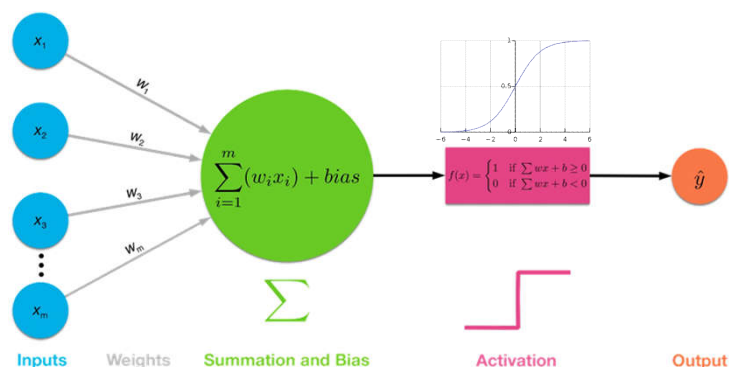
```
model = Sequential()
model.add(Dense(1, input_dim=2, activation='sigmoid'))
```



## Keras 딥러닝: AND 논리 함수

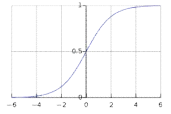
### > Sigmoid 활성 함수 사용

```
model = Sequential()
model.add(Dense(1, input_dim=2, activation='sigmoid'))
```





## Keras 딥러닝: AND 논리 함수



### > Sigmoid 활성화 함수 사용

```
model = Sequential()
model.add(Dense(1, input_dim=2, activation='sigmoid'))
```

### > Sigmoid 활성화 함수 활용

- Binary classification에 효과적 (0 or 1)

## Keras 딥러닝: AND 논리 함수

### > 다층 구조 활용

```
model = Sequential()
model.add(Dense(4, input_dim=2, activation='relu'))
model.add(Dense(1, init='uniform', activation='sigmoid'))
```

## Keras 딥러닝: AND 논리 함수

### > 심층 (Deep) 구조 활용

```
model = Sequential()
model.add(Dense(8, input_dim=2, activation='relu'))
model.add(Dense(4, init='uniform', activation='relu'))
model.add(Dense(1, init='uniform', activation='sigmoid'))
```

## Keras 딥러닝: AND 논리 함수

### > 심층 (Deep) 구조 활용 + Optimizer 변경

```
model = Sequential()
model.add(Dense(8, input_dim=2, activation='relu'))
model.add(Dense(4, init='uniform', activation='relu'))
model.add(Dense(1, init='uniform', activation='sigmoid'))

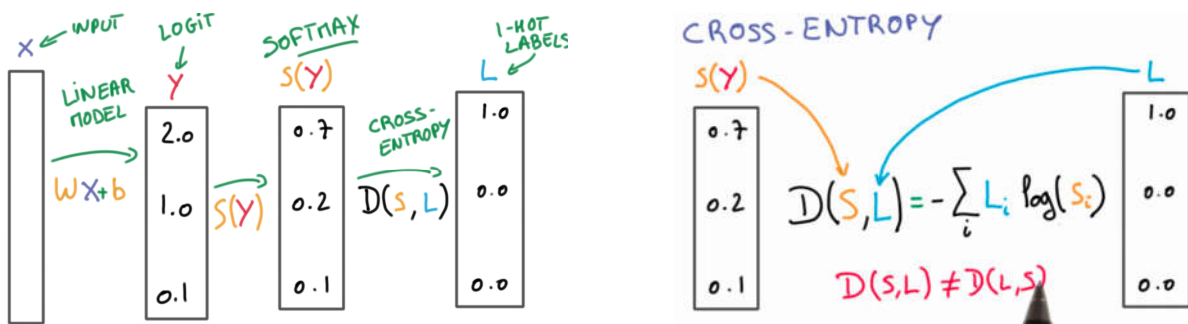
- model.compile(loss='mean_squared_error', optimizer='adam')
+ model.compile(loss='binary_crossentropy', optimizer='adam')
```

## Keras 딥러닝: 활성화 함수의 활용

**회귀** → 항등 함수 (identity function, linear activation)

**분류(0/1)** → 시그모이드 함수 (sigmoid function)

**분류(multiple)** → 소프트맥스 함수 (softmax function)



## Keras 딥러닝: AND 논리 함수

> 심층 (Deep) 구조 활용 + **Optimizer 변경**

```
model = Sequential()
model.add(Dense(16, input_dim=2, activation='relu'))
model.add(Dense(8, init='uniform', activation='relu'))
model.add(Dense(4, init='uniform', activation='softmax'))
```

```
- model.compile(loss='mean_squared_error', optimizer='adam')
+ model.compile(loss='categorical_crossentropy', ...)
```

# Keras 딥러닝 예제 실습

1. 영화 리뷰 분류: 이진 분류
2. 뉴스 기사 분류: 다중 분류



## Keras 딥러닝: 영화 리뷰 분류\*

- > 영화 리뷰를 긍정과 부정으로 분류
- > IMDB (internet movie database) 데이터셋
  - 양 극단의 리뷰 5만개로 이루어짐
  - 훈련데이터 2만5천개, 테스트데이터 2만5천개
  - 각각 50% 부정, 50% 긍정



```
from keras.datasets import imdb
(train_data, train_labels), (test_data, test_labels) =
    imdb.load_data(num_words=10000)
```

\* <https://wikidocs.net/24586>

\*\* num\_words=10000 : 훈련데이터에서 가장 자주 나타나는 단어 1만개만 사용.

\*\*\* <https://github.com/rickiepark/deep-learning-with-python-notebooks>

## Keras 딥러닝: 영화 리뷰 분류\*

[Open in Colab](#)

한림대학교 소프트웨어 융합 대학 특강

누구나 즐기는 딥러닝: 오픈소스 Keras를 활용하여!!!

이정근 교수

빅데이터전공주임/오픈소스소프트웨어센터장 소프트웨어융합대학

jeonggun.lee@hallim.ac.kr 2019년 5월

### IMDB 리뷰 감성 분류하기

imdb의 데이터 로딩 확일이 이전 버전의 numpy를 활용하기 때문에, 이전 numpy 버전 (1.16.2)을 새롭게 설치함

이를 위하여 다음 명령어를 수행

`!pip install numpy==1.16.2`

In [1]: `!pip install numpy==1.16.2`

Requirement already satisfied: numpy==1.16.2 in /usr/local/lib/python3.6/dist-packages (1.16.2)

In [2]: `import keras`

`keras.__version__`

Using TensorFlow backend.

Out [2]: '2.2.4'

[https://github.com/jeonggunlee/OpenSourceKeras/blob/master/04\\_classifying\\_movie\\_reviews.ipynb](https://github.com/jeonggunlee/OpenSourceKeras/blob/master/04_classifying_movie_reviews.ipynb)

\* <https://github.com/rickiepark/deep-learning-with-python-notebooks>

## Keras 딥러닝 예제 실습

1. 영화 리뷰 분류: 이진 분류
2. 뉴스 기사 분류: 다중 분류



## Keras 딥러닝: 뉴스 기사 분류

- > 로이터 뉴스를 **46개의 상호 배타적인 토픽**으로 분류
- > 1986년에 로이터에서 공개한 짧은 뉴스 기사와 토픽의 집합인 로이터 데이터셋을 사용



```
from keras.datasets import reuters
(train_data, train_labels), (test_data, test_labels) =
    imdb.load_data(num_words=10000)
```

[https://github.com/jeonggunlee/OpenSourceKeras/blob/master/05\\_classifying\\_newswires.ipynb](https://github.com/jeonggunlee/OpenSourceKeras/blob/master/05_classifying_newswires.ipynb)  
 \* <https://github.com/rickiepark/deep-learning-with-python-notebooks>

## Keras 딥러닝: 뉴스 기사 분류

[Open in Colab](#)

한림대학교 소프트웨어 융합 대학 특강

누구나 즐기는 딥러닝: 오픈소스 Keras를 활용하여!!!

이정근 교수

빅데이터전공주임/오픈소스소프트웨어센터장 소프트웨어융합대학

jeonggun.lee@hallym.ac.kr 2019년 5월

**로이터 뉴스 기사 분류: 다중 분류 문제**

로이터 뉴스를 46개의 상호 배타적인 토픽으로 분류!

```
"reuters":
['cocoa', 'grain', 'veg-oil', 'earn', 'acq', 'wheat',
'copper', 'housing', 'money-supply', 'coffee', 'sugar',
'trade', 'reserves', 'ship', 'cotton', 'carcass', 'crude',
'nat-gas', 'cpi', 'money-fx', 'interest', 'gnp', 'meal-
feed', 'alum', 'oilseed', 'gold', 'tin', 'strategic-metal',
'livestock', 'retail', 'ipi', 'iron-steel', 'rubber', 'heat',
'jobs', 'lei', 'bop', 'zinc', 'orange', 'pet-chem', 'dlr',
'gas', 'silver', 'wpi', 'hog', 'lead'],
```

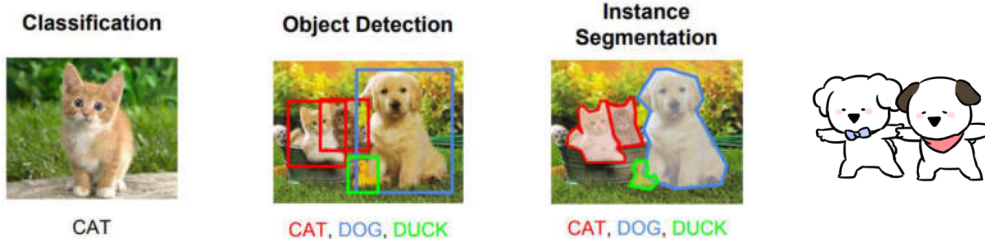
```
In [0]: import keras
keras.__version__
Using TensorFlow backend.
```

```
Out [0]: '2.2.4'
```

[https://github.com/jeonggunlee/OpenSourceKeras/blob/master/05\\_classifying\\_newswires.ipynb](https://github.com/jeonggunlee/OpenSourceKeras/blob/master/05_classifying_newswires.ipynb)  
 \* <https://github.com/rickiepark/deep-learning-with-python-notebooks>

# Keras 딥러닝 예제 실습

## 3. 이미지 인식



## Keras 딥러닝: MNIST 숫자 이미지 인식~!

[Open in Colab](#)

한림대학교 소프트웨어 융합 대학 특강

누구나 즐기는 딥러닝: 오픈소스 Keras를 활용하여!!!

이정근 교수

빅데이터전공주임/오픈소스소프트웨어센터장 소프트웨어융합대학

jeonggun.lee@hailym.ac.kr 2019년 5월

```
In [1]: import keras
keras.__version__
Using TensorFlow backend.
Out [1]: 2.2.4
```

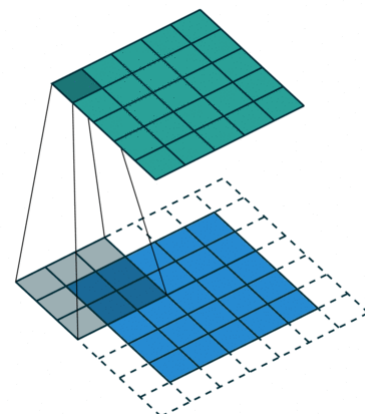
이하 자료는 모두 다음 사이트의 내용에서 가져온 자료입니다.

<https://github.com/rickiepark/deep-learning-with-python-notebooks>

원작자: François Chollet, <https://github.com/fchollet>

한글화: Haesun Park (rickiepark) <https://github.com/rickiepark>

### > Convolution Filter/Layer



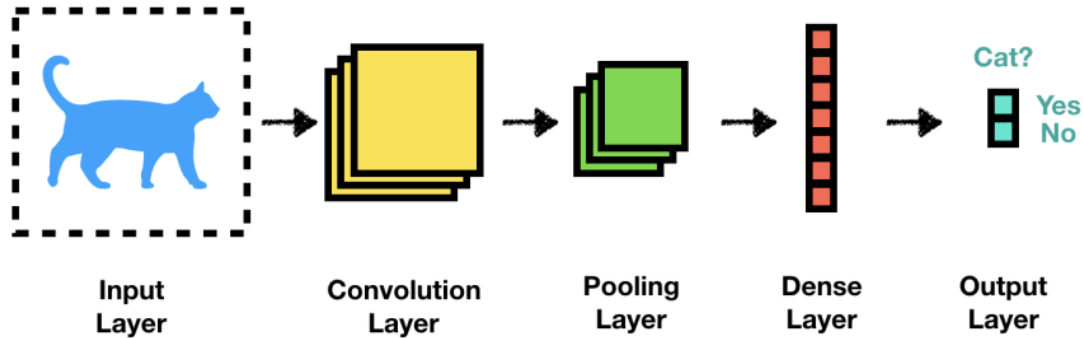
### 5.1 - 합성곱 신경망 소개

이 노트북은 [케라스: 딥러닝을 배우는 딥러닝](#) 책의 5장 1절의 코드 예제입니다. 책에는 더 많은 내용과 그림이 있습니다. 이 노트북에는 소스 코드에 관련된 설명만 포함합니다. 이 노트북의 설명은 케라스 버전 2.2.2에 맞추어져 있습니다. 케라스 최신 버전이 릴리스되면 노트북을 다시 테스트하기 때문에 설명과 코드의 결과가 조금 다를 수 있습니다.

[https://github.com/jeonggunlee/OpenSourceKeras/blob/master/06\\_introduction\\_to\\_convnets.ipynb](https://github.com/jeonggunlee/OpenSourceKeras/blob/master/06_introduction_to_convnets.ipynb)

\* <https://github.com/rickiepark/deep-learning-with-python-notebooks>

## Keras 딥러닝: MNIST 숫자 이미지 인식~!



[https://github.com/jeonggunlee/OpenSourceKeras/blob/master/06\\_introduction\\_to\\_convnets.ipynb](https://github.com/jeonggunlee/OpenSourceKeras/blob/master/06_introduction_to_convnets.ipynb)  
 \* <https://github.com/rickiepark/deep-learning-with-python-notebooks>

## Keras 딥러닝: MNIST 숫자 이미지 인식~!

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved  
Feature

`pe=(28, 28, 1))`

`)))`

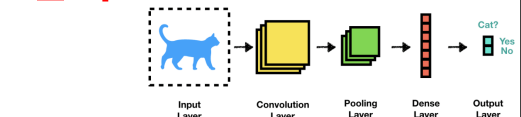
`activ`

`)))`

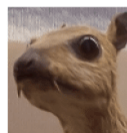
`activ`

`n='relu'))`

`n='softmax'))`



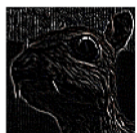
Input image



Convolution  
Kernel

-1	-1	-1
-1	8	-1
-1	-1	-1

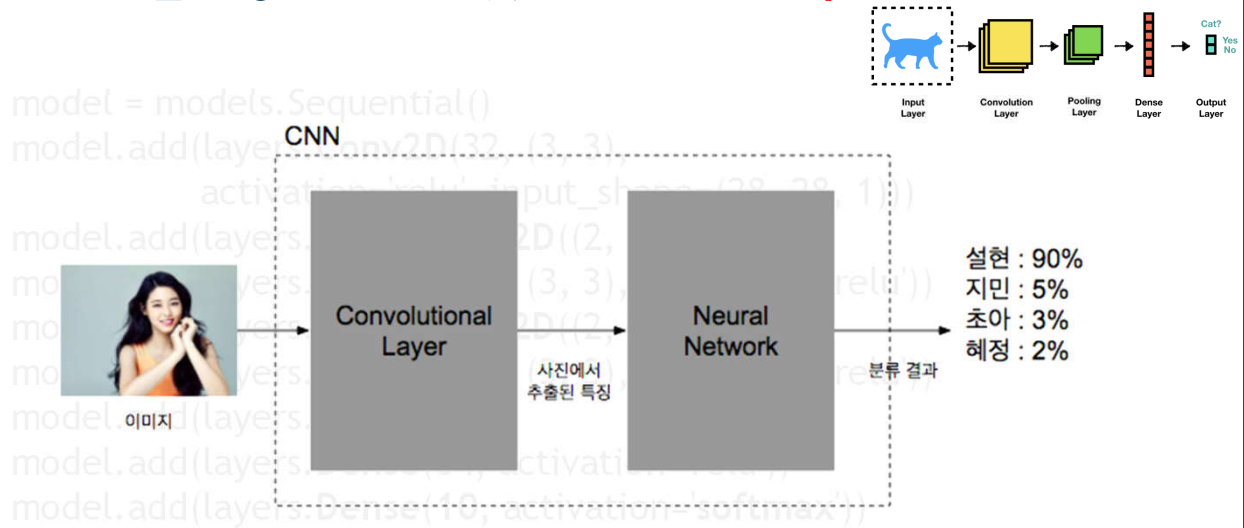
Feature map



[https://github.com/jeonggunlee/OpenSourceKeras/blob/master/06\\_introduction\\_to\\_convnets.ipynb](https://github.com/jeonggunlee/OpenSourceKeras/blob/master/06_introduction_to_convnets.ipynb)  
 \* <https://github.com/rickiepark/deep-learning-with-python-notebooks>



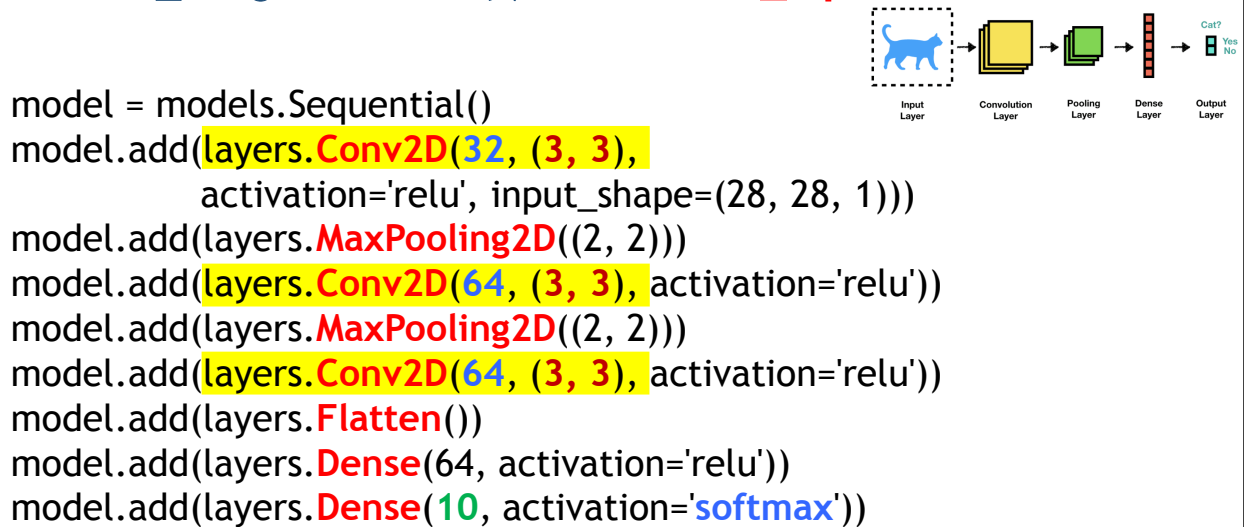
## Keras 딥러닝: MNIST 숫자 이미지 인식~!



[https://github.com/jeonggunlee/OpenSourceKeras/blob/master/06\\_introduction\\_to\\_convnets.ipynb](https://github.com/jeonggunlee/OpenSourceKeras/blob/master/06_introduction_to_convnets.ipynb)

\* <https://github.com/rickiepark/deep-learning-with-python-notebooks>

## Keras 딥러닝: MNIST 숫자 이미지 인식~!



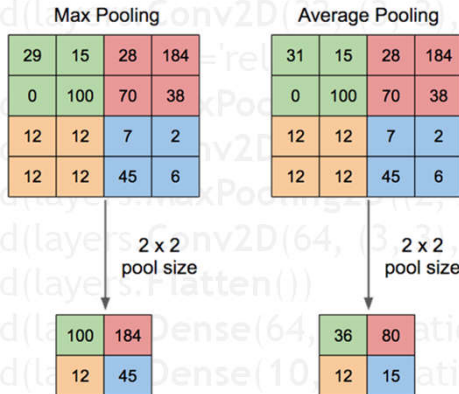
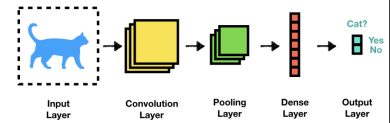
```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3),
                        activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

[https://github.com/jeonggunlee/OpenSourceKeras/blob/master/06\\_introduction\\_to\\_convnets.ipynb](https://github.com/jeonggunlee/OpenSourceKeras/blob/master/06_introduction_to_convnets.ipynb)

\* <https://github.com/rickiepark/deep-learning-with-python-notebooks>

## Keras 딥러닝: MNIST 숫자 이미지 인식~!

```
model.add(layers.MaxPooling2D((2, 2)))
```

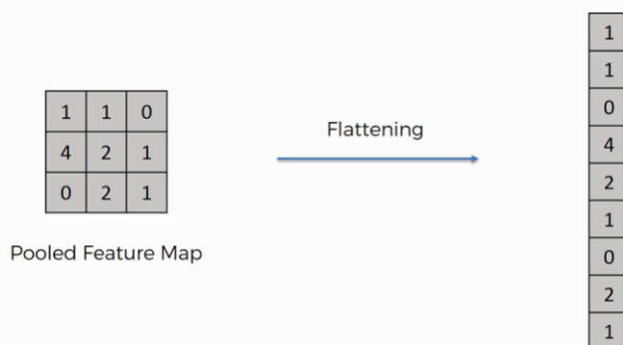
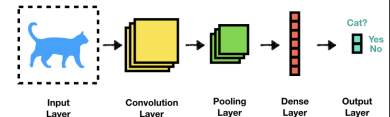


[https://github.com/jeonggunlee/OpenSourceKeras/blob/master/06\\_introduction\\_to\\_convnets.ipynb](https://github.com/jeonggunlee/OpenSourceKeras/blob/master/06_introduction_to_convnets.ipynb)

\* <https://github.com/rickiepark/deep-learning-with-python-notebooks>

## Keras 딥러닝: MNIST 숫자 이미지 인식~!

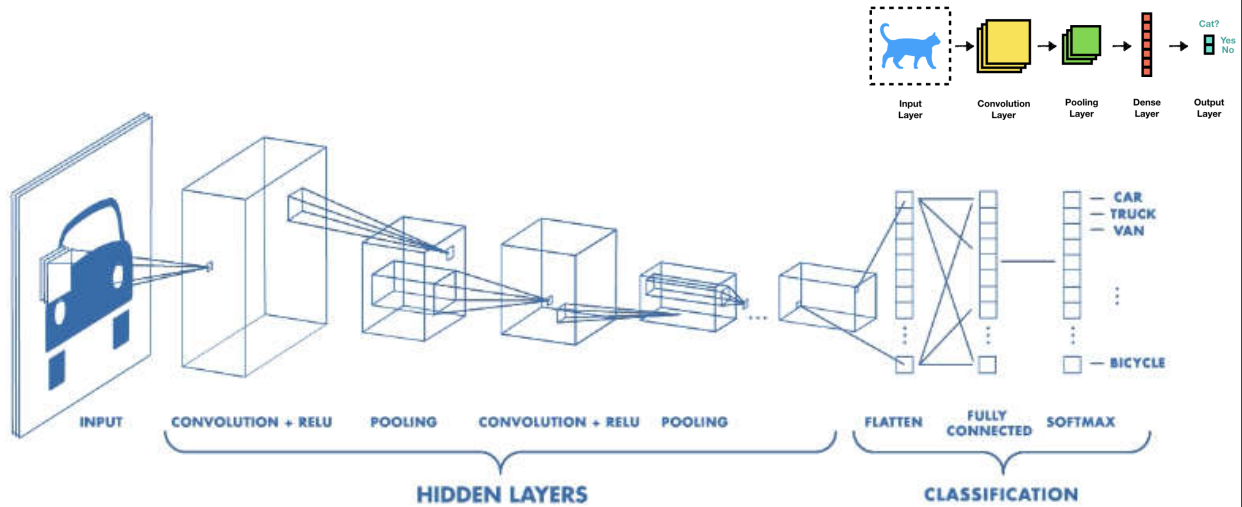
```
model.add(layers.Flatten())
```



[https://github.com/jeonggunlee/OpenSourceKeras/blob/master/06\\_introduction\\_to\\_convnets.ipynb](https://github.com/jeonggunlee/OpenSourceKeras/blob/master/06_introduction_to_convnets.ipynb)

\* <https://github.com/rickiepark/deep-learning-with-python-notebooks>

## Keras 딥러닝: MNIST 숫자 이미지 인식~!



[https://github.com/jeonggunlee/OpenSourceKeras/blob/master/06\\_introduction\\_to\\_convnets.ipynb](https://github.com/jeonggunlee/OpenSourceKeras/blob/master/06_introduction_to_convnets.ipynb)  
 \* <https://github.com/rickiepark/deep-learning-with-python-notebooks>

<https://www.cs.toronto.edu/~kriz/cifar.html>  
<https://en.wikipedia.org/wiki/CIFAR-10>

## Keras 딥러닝 예제 실습

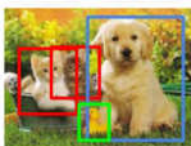
### 3. 이미지 인식 – One more ~

Classification



CAT

Object Detection



CAT, DOG, DUCK

Instance Segmentation



CAT, DOG, DUCK



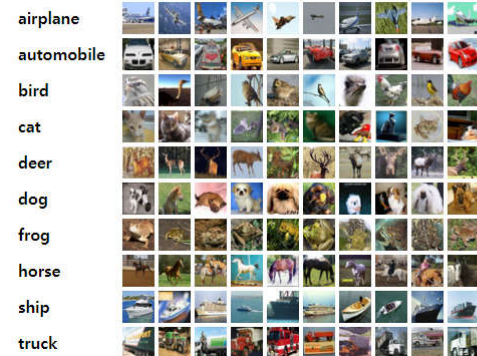
<https://github.com/jeonggunlee/OpenSourceKeras/blob/master/Cifar10.ipynb>

#### The CIFAR-10 dataset

The **CIFAR-10** dataset consists of **60000 32x32** colour images in **10 classes**, with **6000 images per class**. There are **50000 training images** and **10000 test images**.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

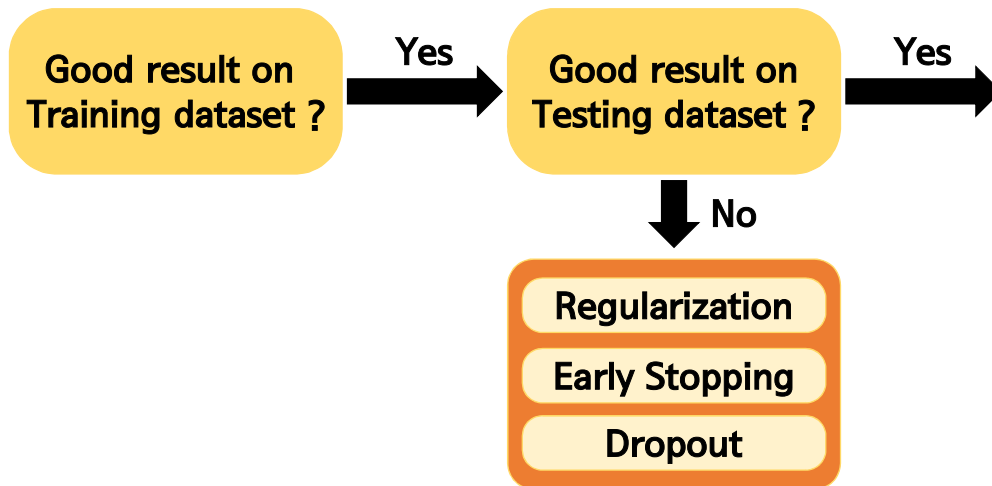
Here are the classes in the dataset, as well as 10 random images from each:



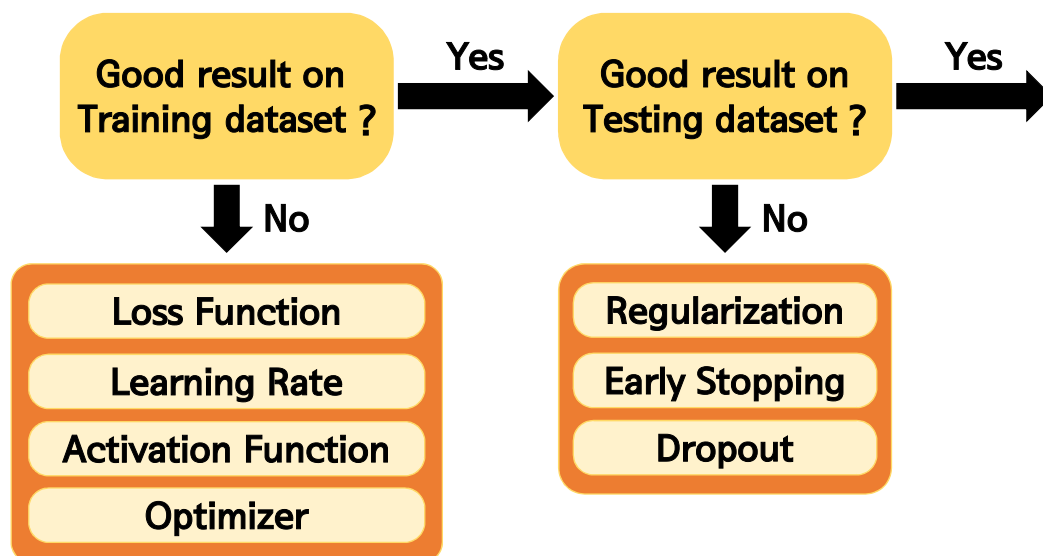
## Keras 딥러닝 최적화



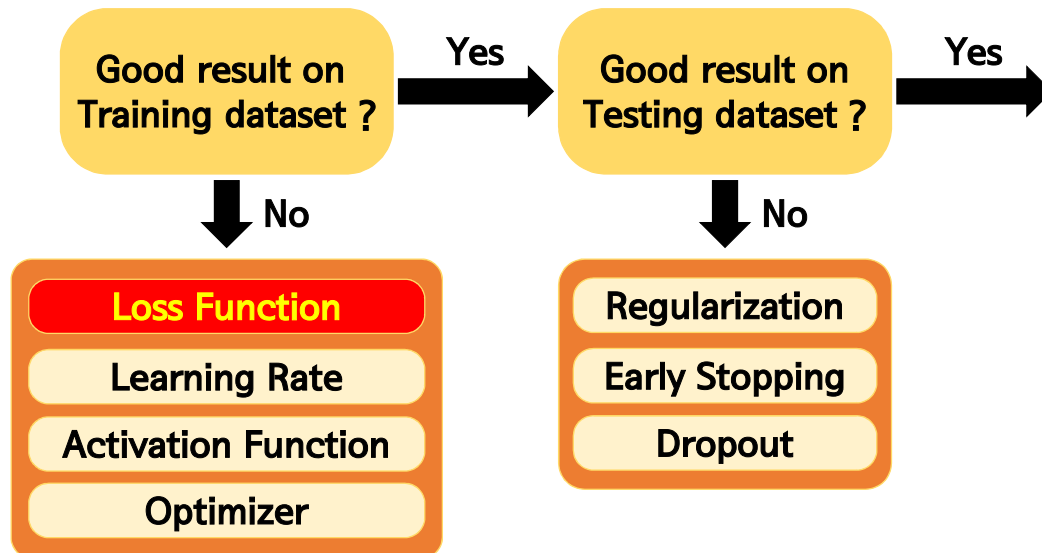
## 딥러닝 최적화



## 딥러닝 최적화



## 딥러닝 최적화



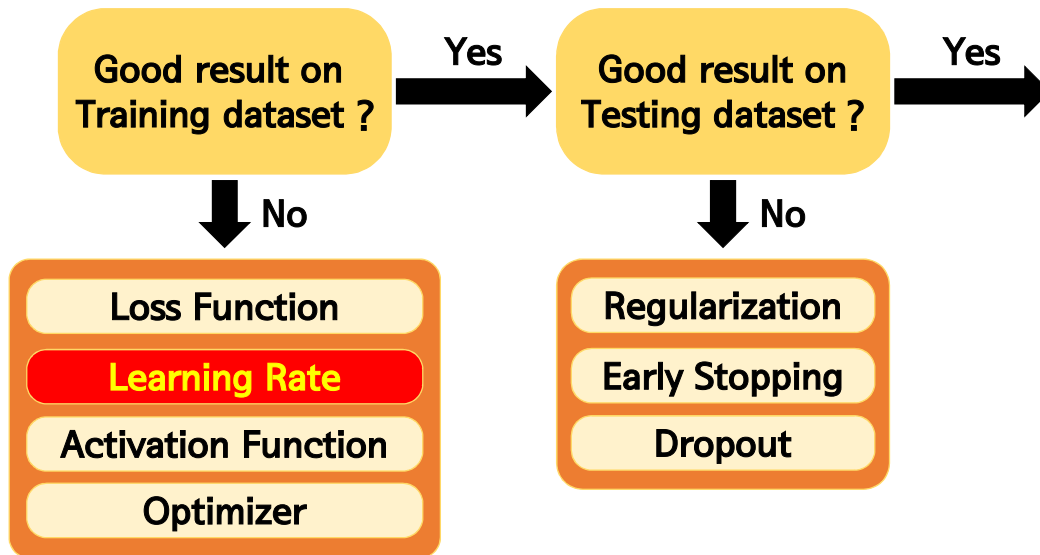
## Keras 최적화 : 모델 컴파일

```
model.compile(loss='mean_squared_error', optimizer='adam')
```

`model.compile( ... )`

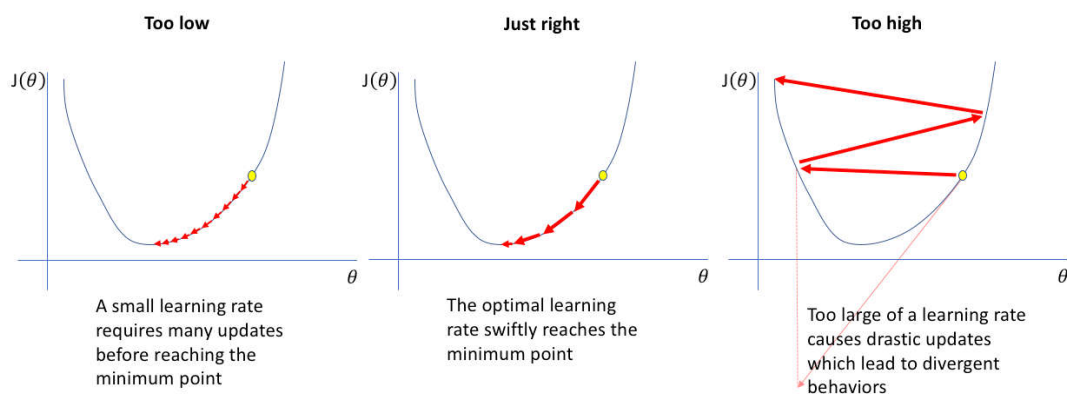
- > **loss**: 실제 출력값과 모델을 통해서 얻은 출력값의 상이 수준
  - `mean_squared_error` (연속적인 출력값의 예측)
  - `binary_crossentropy` (이진 분류, YES/NO)
  - `categorical_crossentropy` (다중 분류, 어디에 속하나?)
  - ...
- > **optimizer**:

## 딥러닝 최적화

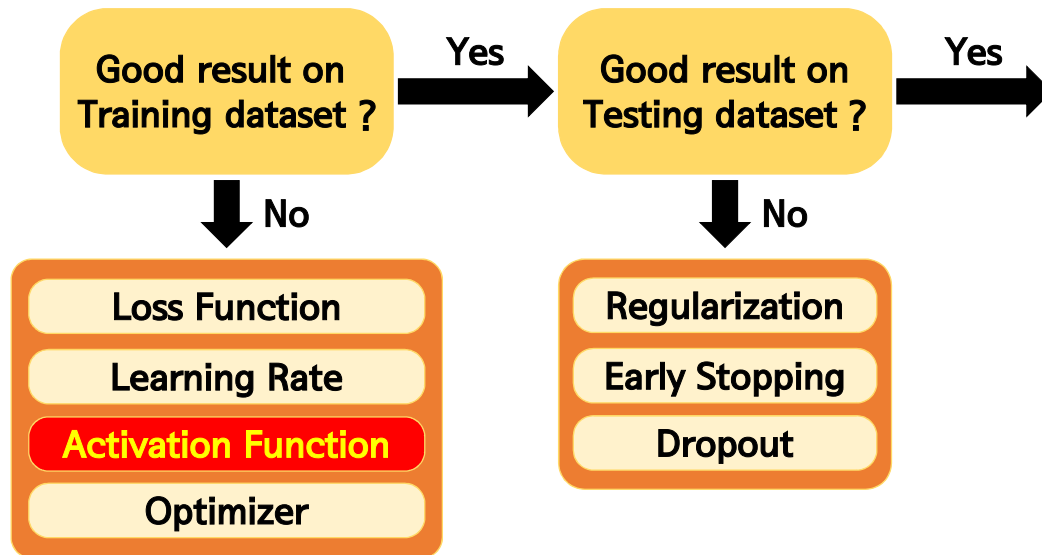


## 딥러닝 최적화

```
adam = optimizers.Adam(lr=0.01) # learning rate
model.compile(loss='mean_squared_error', optimizer=adam)
```



## 딥러닝 최적화



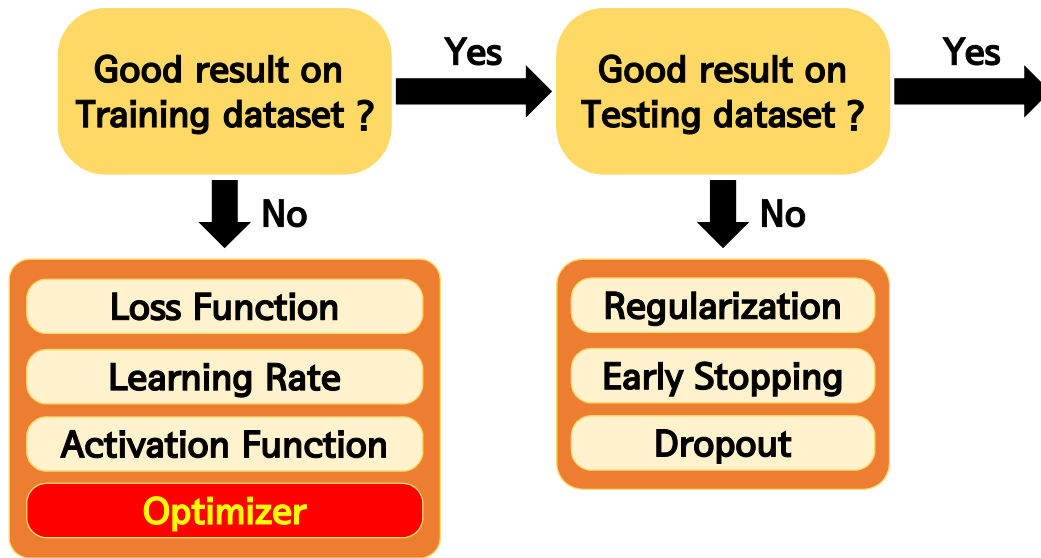
## 딥러닝 최적화

다양한  
활성화 함수

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parametric Rectified Linear Unit (PReLU) (2)		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) (3)		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

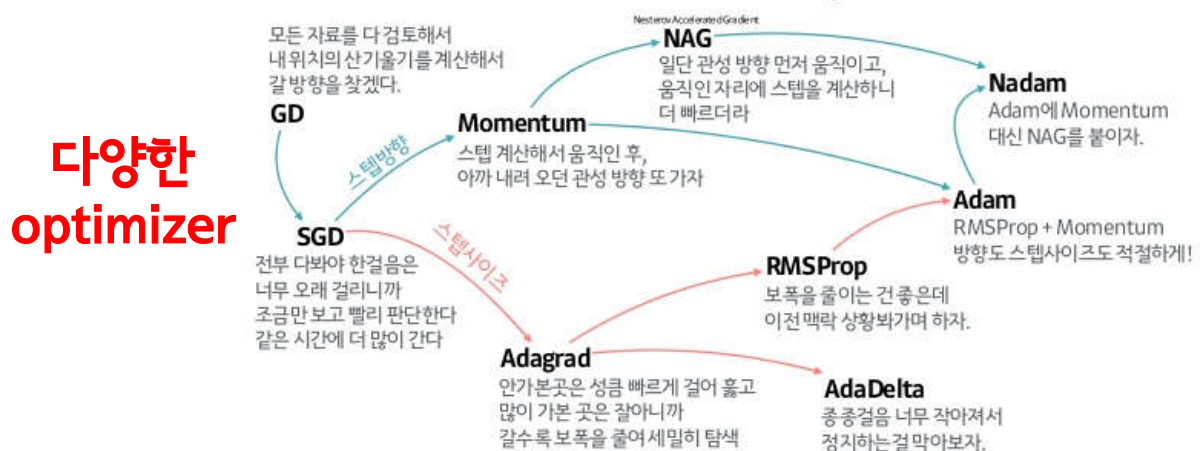


## 딥러닝 최적화

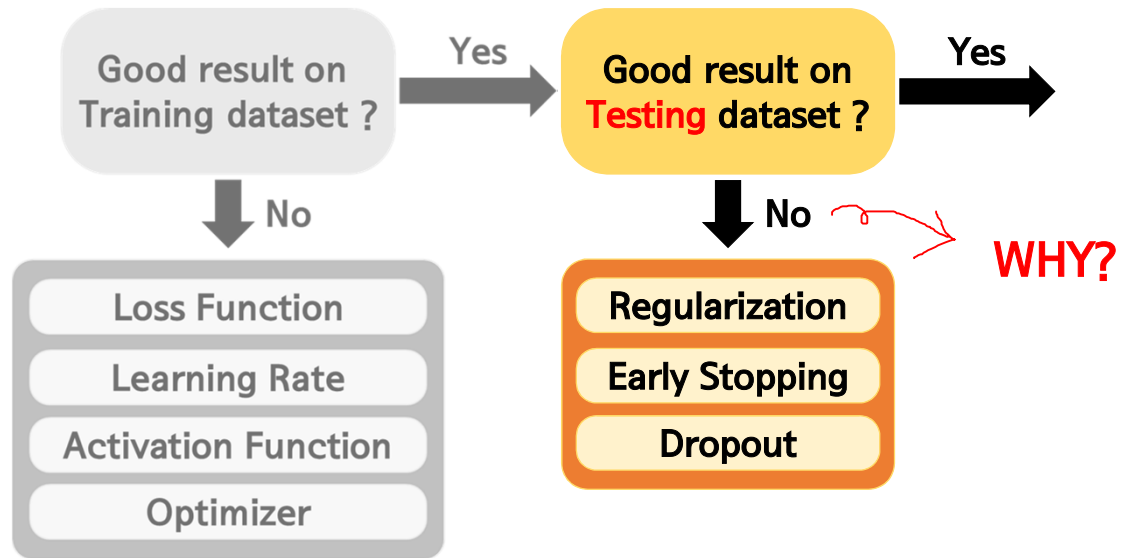


## 딥러닝 최적화

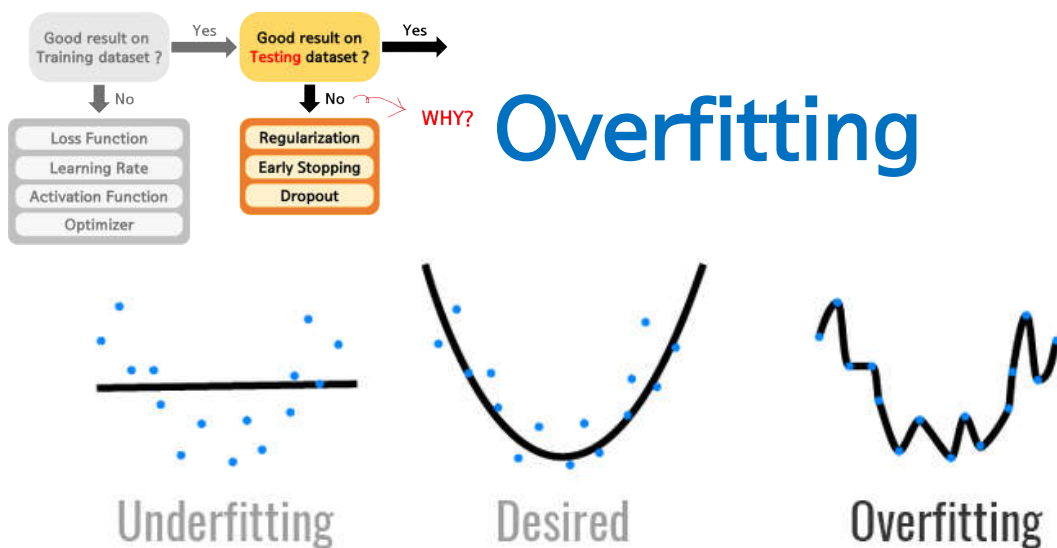
### 산 내려오는 작은 오솔길 잘찾기(Optimizer)의 발달 계보



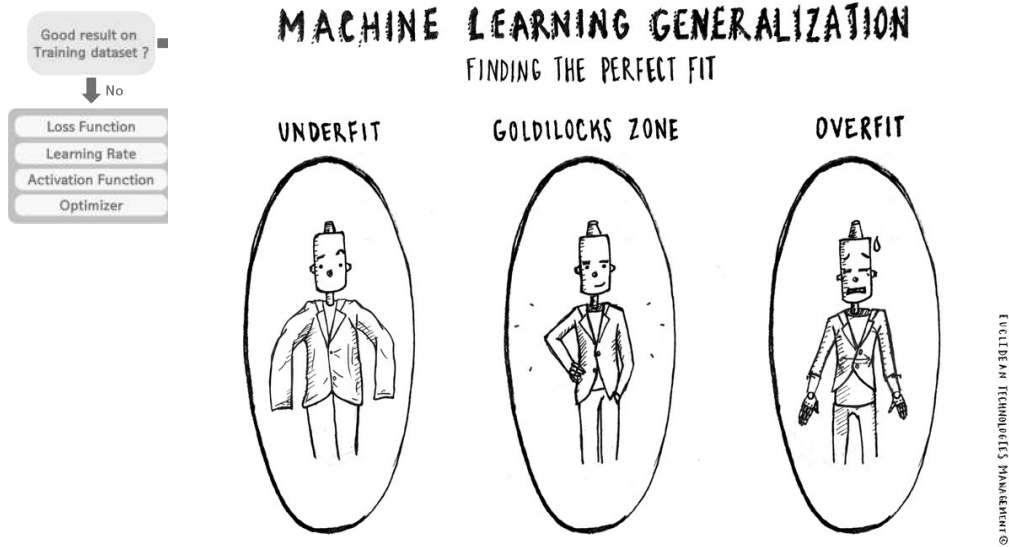
## 딥러닝 최적화



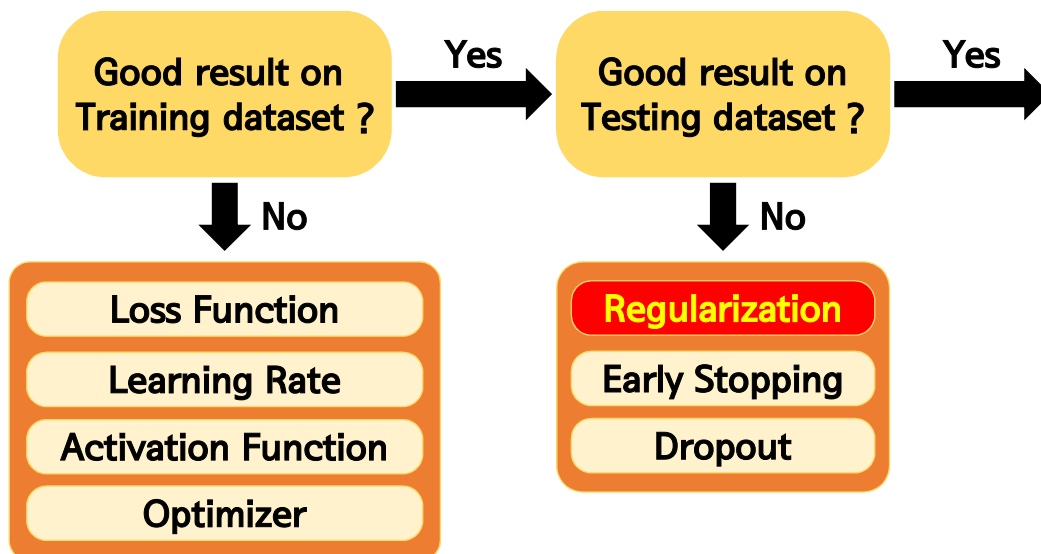
## 딥러닝 최적화



## 딥러닝 최적화



## 딥러닝 최적화

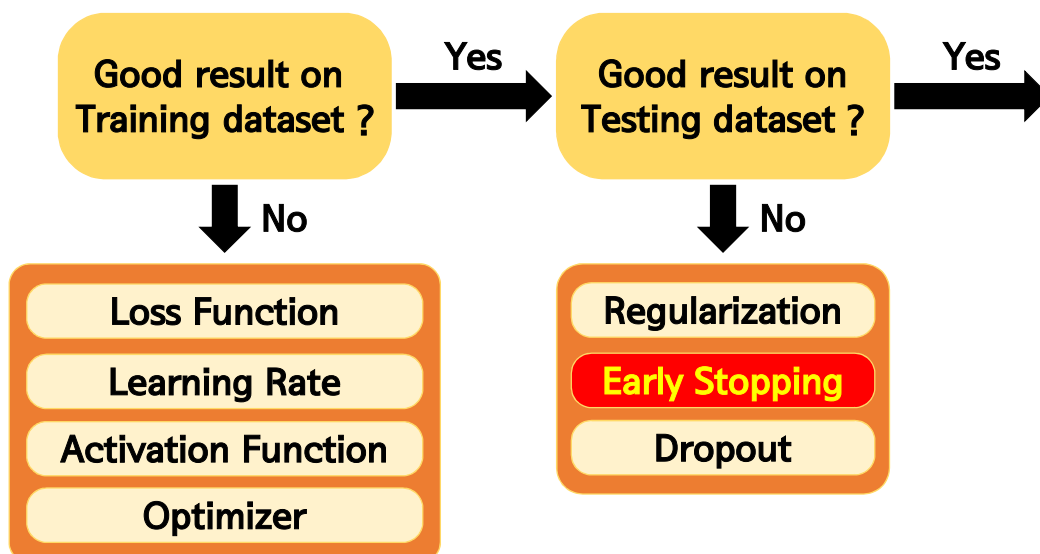


## 딥러닝 최적화 : Regularization

```
from keras import regularizers
model.add(Dense(64, input_dim=64,
kernel_regularizer=regularizers.l2(0.01),
activity_regularizer=regularizers.l1(0.01)))
```

정규화?  $\mathcal{L}' = \mathcal{L} + \beta \underbrace{\frac{1}{2} \|w\|_2^2}_{\frac{1}{2}(w_1^2 + w_2^2 + \dots + w_N^2)}$

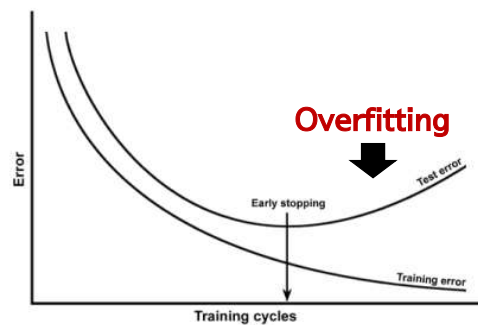
## 딥러닝 최적화



## 딥러닝 최적화 : Early Stopping

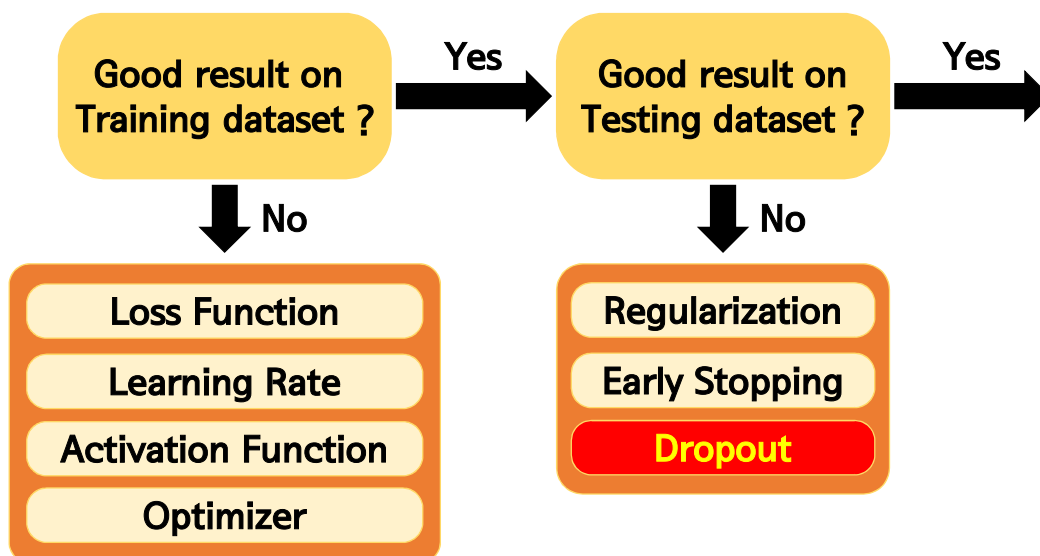
```
early_stopping = EarlyStopping()
model.fit(X_train, Y_train, nb_epoch= 1000, callbacks =
[early_stopping])
```

**빨리  
끝내기!**



<https://wikidocs.net/28147>

## 딥러닝 최적화

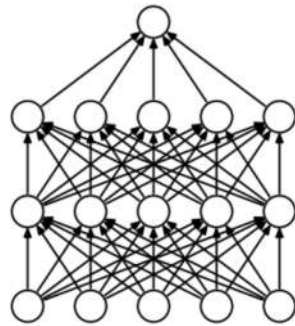


## 딥러닝 최적화 : Dropout

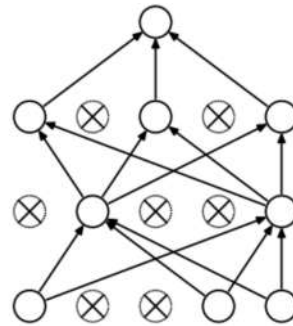
...

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Dropout(0.25))
```



(a) Standard Neural Net



(b) After applying dropout.

TiP! 💡

## Keras 딥러닝 Tip



## 딥러닝 Tips: Save & Load

### # SAVE MODEL

```
model.save('my_model.h5')  
del model
```

### # RELOAD MODEL

```
from keras.models import load_model  
model = load_model('my_model.h5')  
model.summary()  
yFit = model.predict(xVal, batch_size=10, verbose=1)
```

## 딥러닝 Tips: 가중치 정보 읽기

### # get weights

```
myweight = model.get_weights()
```

### # set weights

```
model.layers[1].set_weights(myweights[0:2])  
model.summary()
```

## 딥러닝 Tips: 층의 출력 값 알아내기

### # get weights

```
model_layer1 = Sequential()  
model_layer1.add(Dense(128,input_dim=200,  
                        weights=model_test.layers[0].get_weights()))  
model_layer1.add(Activation('relu'))
```

### # predict

```
model_layer1.predict(X_train[0:1])
```

# END!

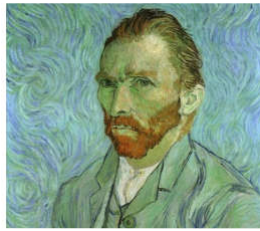
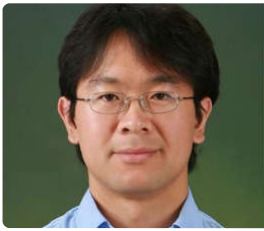






**질문 ?**

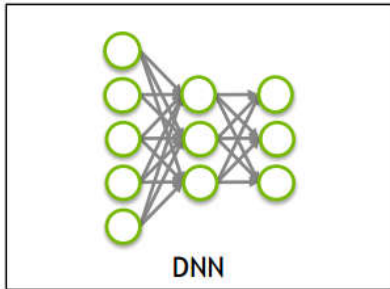
**Jeonggun.Lee@gmail.com**



**왜 지금 인가 ?**

## 배경

# The Big Bang in Machine Learning



새로운 알고리즘의  
효과성 증명



데이터의  
폭발적인 증가



컴퓨터기술의  
비약적 발전



인공지능 활용에 대한 사회적 관심증가

## 배경: 새로운 알고리즘의 효과



많이도 불렀군요 → 인공지능 연구자

마침내 **1998년** 제프리 힌튼 교수 밑에서  
박사학과정을 밟고 있던 얀 레쿰이 요슈아 벤지오와 함께  
블츠만 머신에 **백프로파게이션**을 결합하여  
**CNN**을 완성함으로써 딥러닝의 획기적인  
전환점을 마련하였습니다.



제프리 힌튼

간단한 것  
같지만 결코  
간단하지 않아



얀 레쿰



요슈아 벤지오

맞아  
맞아

배경: 고성능 컴퓨터 기술의 비약적 발전



GPU: Graphics Processing Unit

배경: 고성능 컴퓨터 기술의 비약적 발전



~~GPU: Graphics Processing Unit~~

DPU: Deep-learning Processing Unit

배경: 데이터의 폭발적인 증가



배경: 데이터의 폭발적인 증가



데이터의 세상이라는 지금!  
 얼마나 많은 데이터가  
 하루에  
 만들어지고 있을까요 ?

배경: 데이터의 폭발적인 증가



하루에 만들어지는  
데이터의 크기?



배경: 데이터의 폭발적인 증가



하루에 만들어지는  
데이터의 크기?

**$2.5 \times 10^{18}$  바이트**

배경: 데이터의 폭발적인 증가



하루에 만들어지는  
데이터의 크기?

$2.5 \times 10^{18}$  바이트

250 경 바이트

배경: 데이터의 폭발적인 증가



21세기 원유

"지난 100년간 석유가 세계 산업을 이끌었다면 앞으로는 데이터가 세계 산업을 이끌 것"





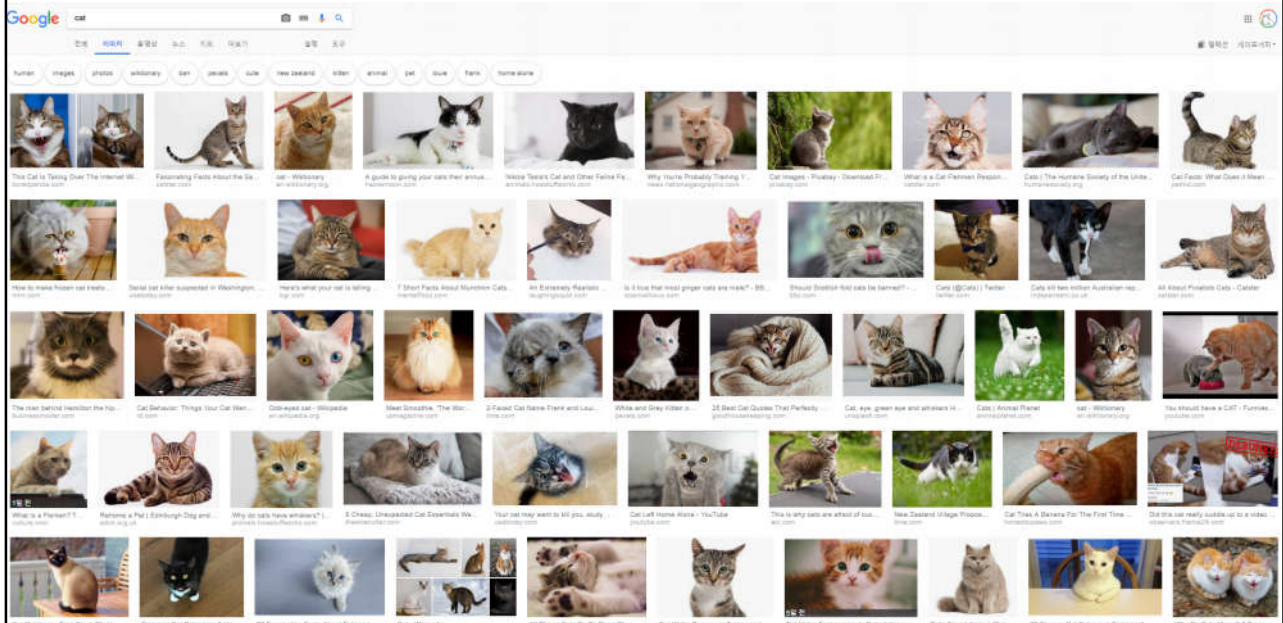
## 빅데이터 분석 및 빅데이터 기반 인공지능



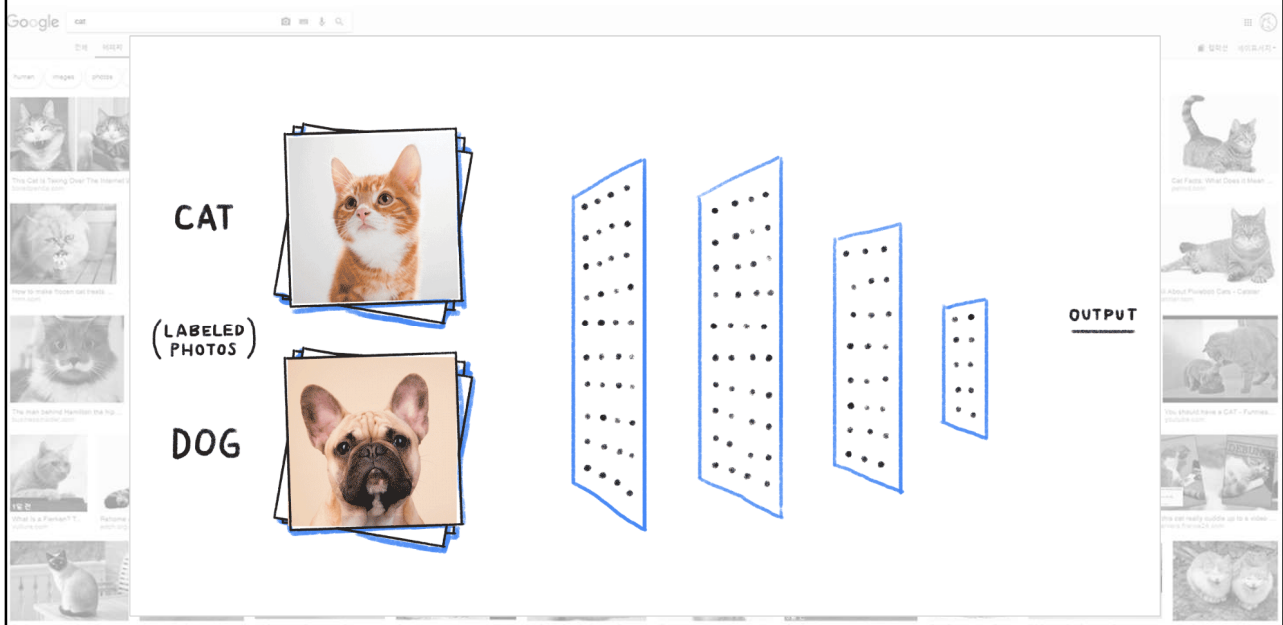
## 빅데이터 분석 및 빅데이터 기반 인공지능



# Google CAT



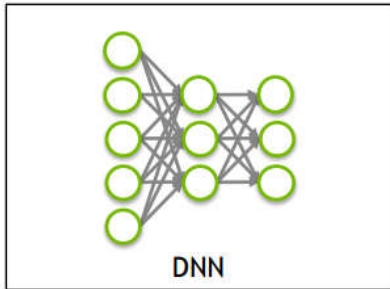
## Google "cat" & 고양이 빅데이터 & 인공지능





배경

# The Big Bang in Machine Learning



새로운 알고리즘의  
효과성 증명



데이터의  
폭발적인 증가



컴퓨터기술의  
비약적 발전



인공지능 활용에 대한 사회적 관심증가