



# PyTorch

서울대학교 산업공학과  
계산금융 및 통계학습 연구실

This document is confidential and is intended solely for the use

# What is Pytorch and How to install?



Image Source: <https://github.com/pytorch/pytorch>

Prof. Jaewook Lee

Blockchain @ SNU

# PYTORCH

# PyTorch

## ■ Deep learning Framework

- FROM RESEARCH TO PRODUCTION
- An open source machine learning framework that accelerates the path from research prototyping to production deployment.
- <https://pytorch.org>, <https://github.com/pytorch/pytorch>, <https://pytorch.kr>

## ■ Facebook AI에서 개발됨.

- OS: Linux, macOS, Windows
- Community: Pytorch KR (<https://www.facebook.com/groups/PyTorchKR/>)
- Version released: v1.11

# PyTorch vs Tensorflow

- 최근 PyTorch가 딥러닝 개발 언어로 급부상 중.

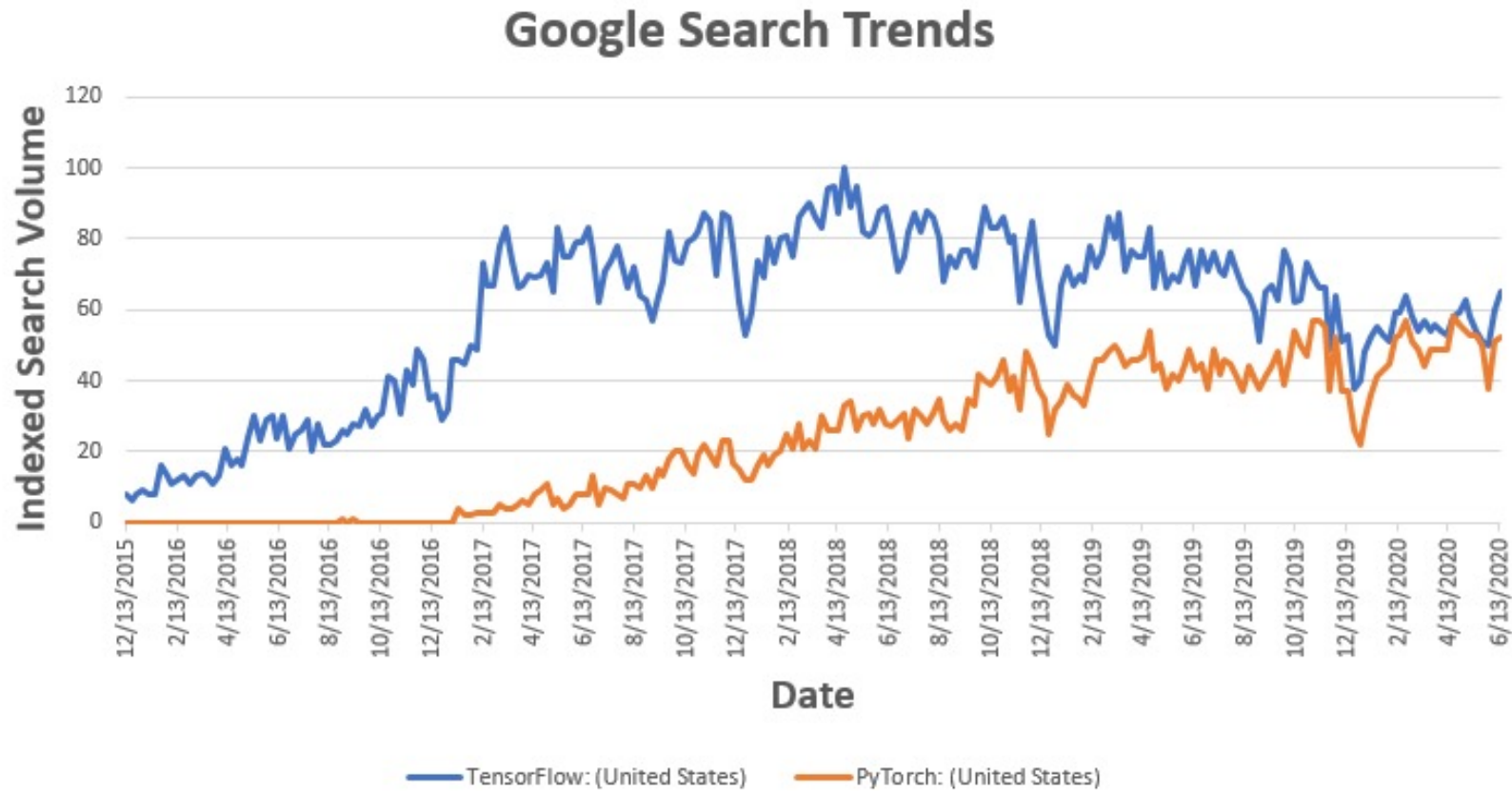


Image Source: [https://miro.medium.com/max/2400/1\\*DNpNE0pK-S0VlsbVg6XrDA.png](https://miro.medium.com/max/2400/1*DNpNE0pK-S0VlsbVg6XrDA.png)

# PyTorch vs Tensorflow


## ■ TensorFlow

- Python 기반의 라이브러리
- 데스크톱, 모바일 등 다양한 플랫폼에 적용 가능
- Keras에서 TensorFlow backend를 사용해 구동 가능
- Define-and-Run

## ■ PyTorch

- 기존 Lua 기반의 딥러닝 프레임워크(torch)를 python으로 이식
- Python과의 유연한 연동 (DataLoader, Numpy, DataFrame)
- 쉽고 간결한 코드
- Define-by-Run

# PyTorch vs Tensorflow

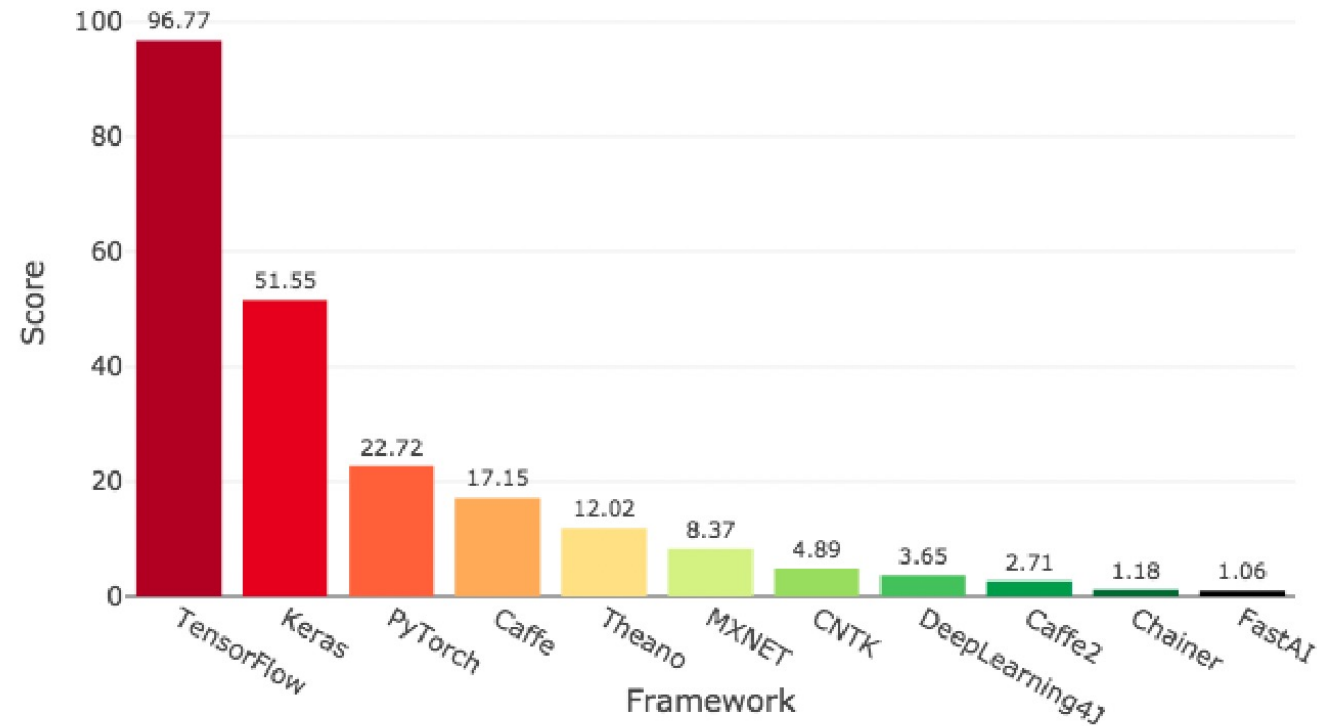
	Languages	Tutorials and training materials	CNN modeling capability	RNN modeling capability	Architecture: easy-to-use and modular front end	Speed	Multiple GPU support	Keras compatible
Theano	Python, C++	++	++	++	+	++	+	+
Tensor-Flow	Python	+++	+++	++	+++	++	++	+
Torch	Lua, Python (new)	+	+++	++	++	+++	++	
Caffe	C++	+	++		+	+	+	
MXNet	R, Python, Julia, Scala	++	++	+	++	++	+++	
Neon	Python	+	++	+	+	++	+	
CNTK	C++	+	+	+++	+	++	+	

<http://codedragon.tistory.com/5004>

# PyTorch vs Tensorflow

■ 2018년

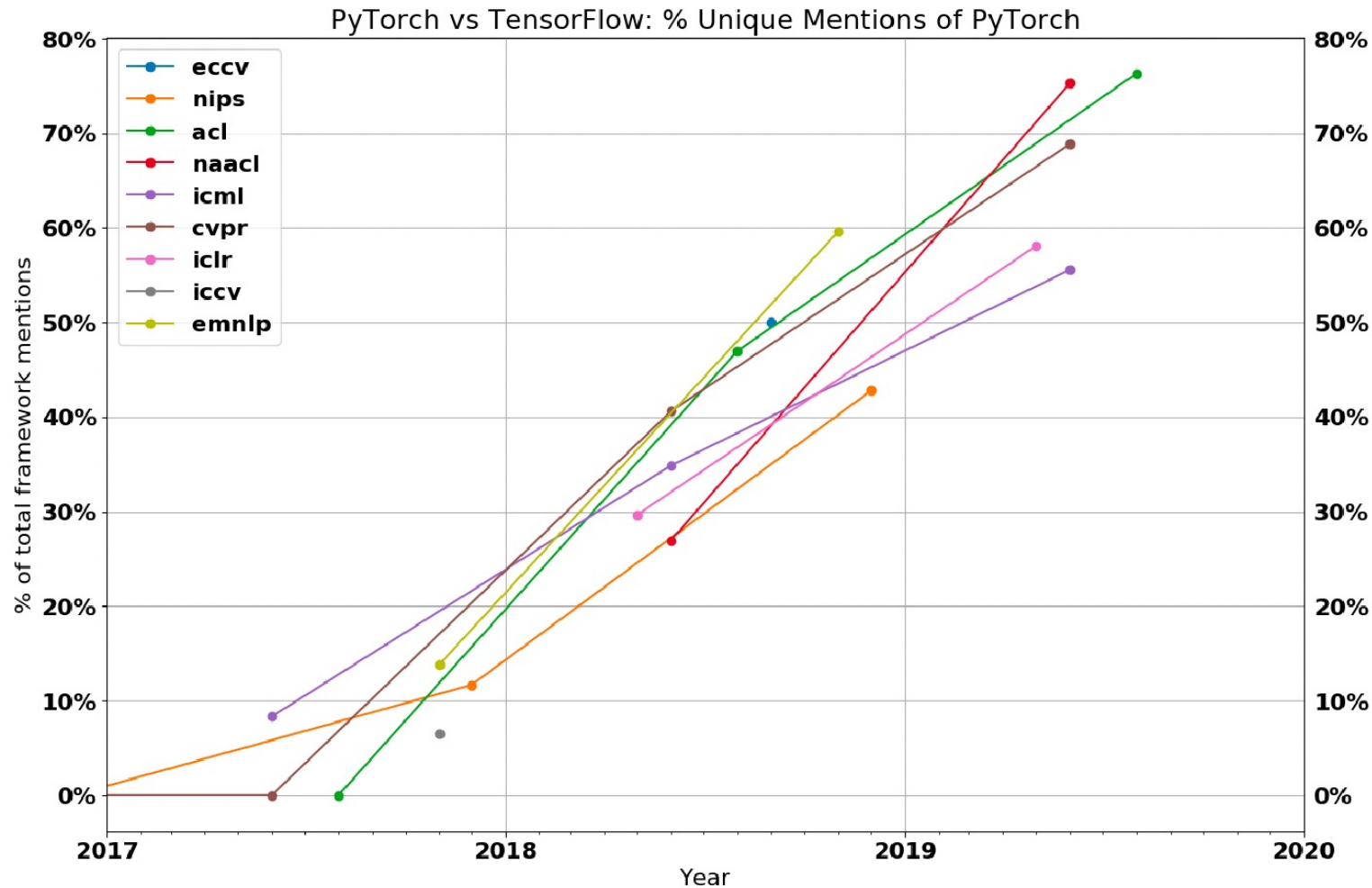
Deep Learning Framework Power Scores 2018





# PyTorch vs Tensorflow

■ 2020년



<https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry/>

# PyTorch vs Tensorflow

## ■ Pytorch이 사용하기 편한 언어다!

- 직장에서는 Tensorflow (이미 배포된 시스템이 Tensorflow 기반)
- 혼자 할 때는 Pytorch (Backtest는 편한 언어로)

↑ Posted by u/cjmcmurtrie 1 year ago

203 Discussion [D] So... Pytorch vs Tensorflow: what's the verdict on how they compare? What are their individual strong points?

↓

Have any users here had extensive experience with both? What are your main concerns or delights with both libraries?

I never made a switch from Torch7 to Tensorflow. I played around with Tensorflow but I always found Torch7 more intuitive (maybe I didn't play around enough!). I also had a tip that Pytorch was on the way, so decided I would wait for that.

After a few weeks using Pytorch, I don't think I'll be moving to Tensorflow any time soon, at least for my passion projects. It's ridiculously simple to write custom modules in Pytorch, and the dynamic graph construction is giving me so many ideas for things that previously would've been achieved by late-night hacks (and possibly put on the wait list). I think Pytorch is an incredible toolset for a machine learning developer. I realise that the wealth of community resources is much stronger for Tensorflow, but when working on novel projects (instead of re-coding known architectures or reading tutorials) this isn't always much help.

Thoughts?

47 Comments Share ... 94% Upvoted

↑ thatguydr 12 points · 1 year ago

↓ This is the best reply I've seen in this subreddit in months. I've been using TF at work strictly because of its distributed capabilities, but I'm going to check out PyTorch at home. Thanks!

Share Save

# PyTorch vs Tensorflow

## ■ 코딩 시의 편리함

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()

        self.layer = nn.Sequential(
            nn.Conv2d(1,16,5),
            nn.ReLU(),
            nn.Conv2d(16,32,5),
            nn.ReLU(),
            nn.MaxPool2d(2,2),
            nn.Conv2d(32,64,5),
            nn.ReLU(),
            nn.MaxPool2d(2,2)
        )

        self.fc_layer = nn.Sequential(
            nn.Linear(64*3*3,100),
            nn.ReLU(),
            nn.Linear(100,10)
        )

    def forward(self,x):
        out = self.layer(x)
        out = out.view(batch_size,-1)
        out = self.fc_layer(out)

        return out
```

v.s.

```
X = tf.placeholder(tf.float32, [None, 28, 28, 1])
Y = tf.placeholder(tf.float32, [None, 10])
keep_prob = tf.placeholder(tf.float32)

W1 = tf.Variable(tf.random_normal([3, 3, 1, 32], stddev=0.01))
L1 = tf.nn.conv2d(X, W1, strides=[1, 1, 1, 1], padding='SAME')
L1 = tf.nn.relu(L1)
L1 = tf.nn.max_pool(L1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

W2 = tf.Variable(tf.random_normal([3, 3, 32, 64], stddev=0.01))
L2 = tf.nn.conv2d(L1, W2, strides=[1, 1, 1, 1], padding='SAME')
L2 = tf.nn.relu(L2)
L2 = tf.nn.max_pool(L2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

W3 = tf.Variable(tf.random_normal([7*7*64, 256], stddev = 0.01))
L3 = tf.reshape(L2, [-1, 7*7*64])
L3 = tf.matmul(L3, W3)
L3 = tf.nn.relu(L3)
L3 = tf.nn.dropout(L3, keep_prob)

W4 = tf.Variable(tf.random_normal([256, 10], stddev=0.01))
model = tf.matmul(L3, W4)
```

## ■ Jax

- Google이 만든 Python과 Numpy만을 결합한 머신러닝 라이브러리
- np.~ 를 jnp.~로 대체하여 사용할 수 있어서 편리함
- 하지만 아직은 개발 중

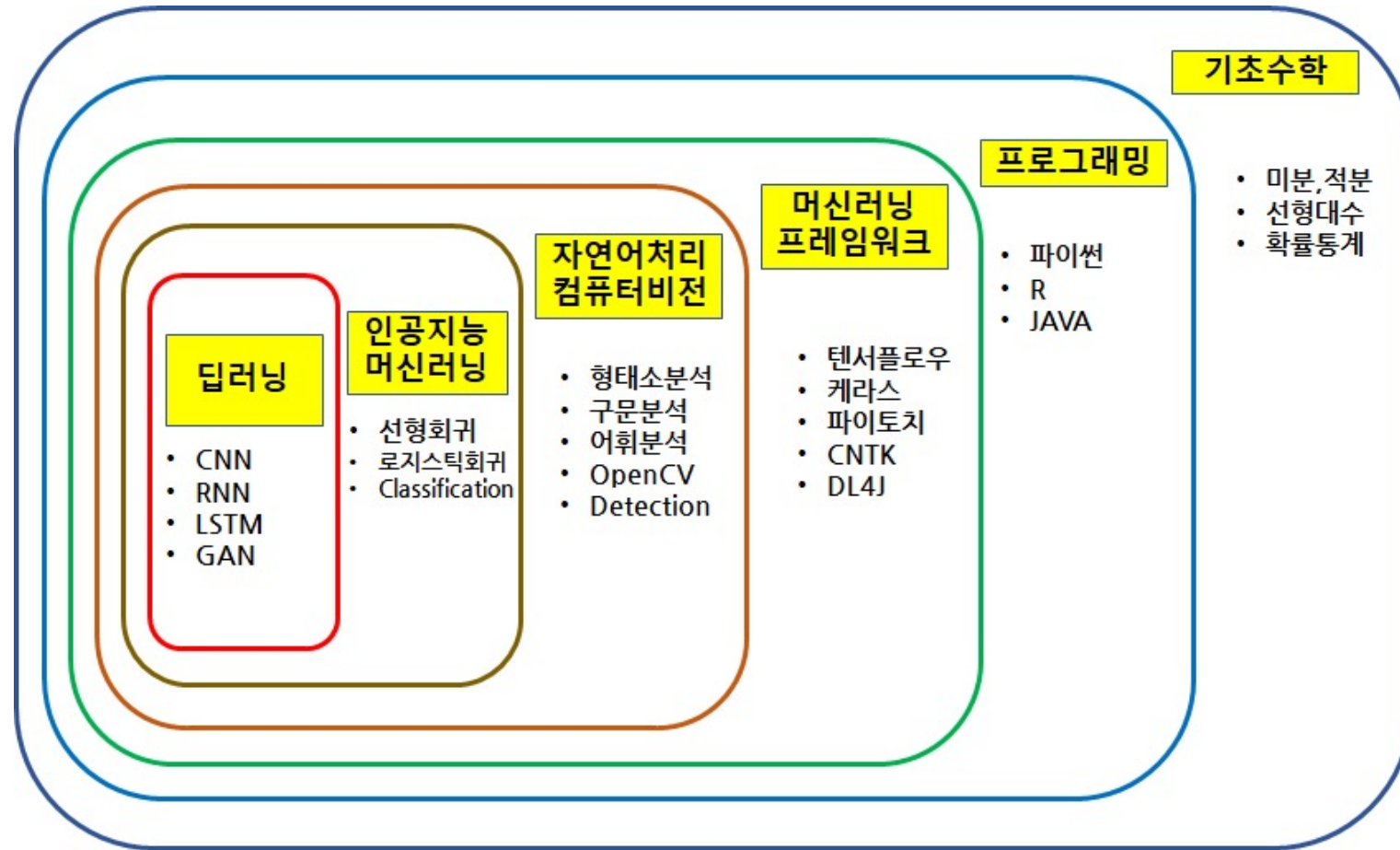
```
from jax import grad
import jax.numpy as jnp

def tanh(x): # Define a function
    y = jnp.exp(-2.0 * x)
    return (1.0 - y) / (1.0 + y)

grad_tanh = grad(tanh) # Obtain its gradient function
print(grad_tanh(1.0)) # Evaluate it at x = 1.0
# prints 0.4199743
```

# PREREQUISITE FOR DEEP LEARNING

# Programming and mathematics

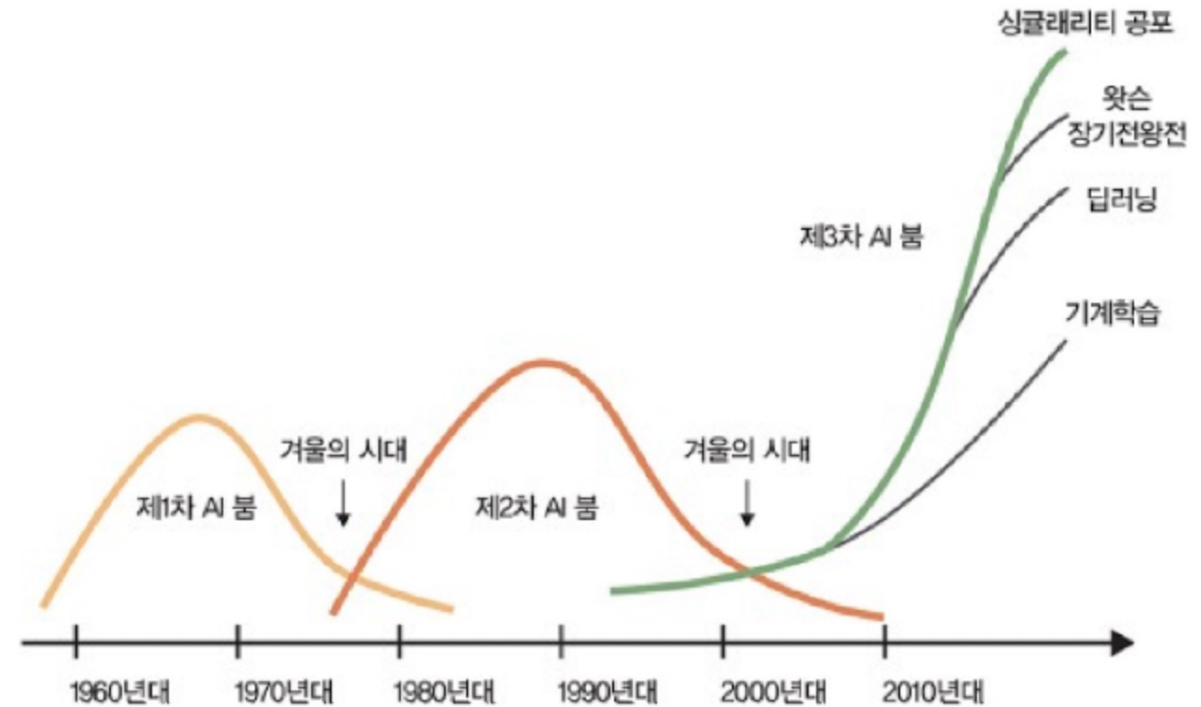


<https://sdc-james.gitbook.io/onebook/2.-1/2.6.-ai-3>

# Deep learning의 역사

## ■ Artificial Intelligence는 새로운 개념이 아니다!

- 놀랍게도 1960년대부터 사용되던 개념임.
- 제 1차 AI boom: Perceptron  
→ 학습시킬 방법의 부재: First AI winter
- 제 2차 AI boom: Backpropagation  
→ SVM의 등장과 학습의 한계:  
Second AI winter
- 제 3차 AI boom: Deep neural networks



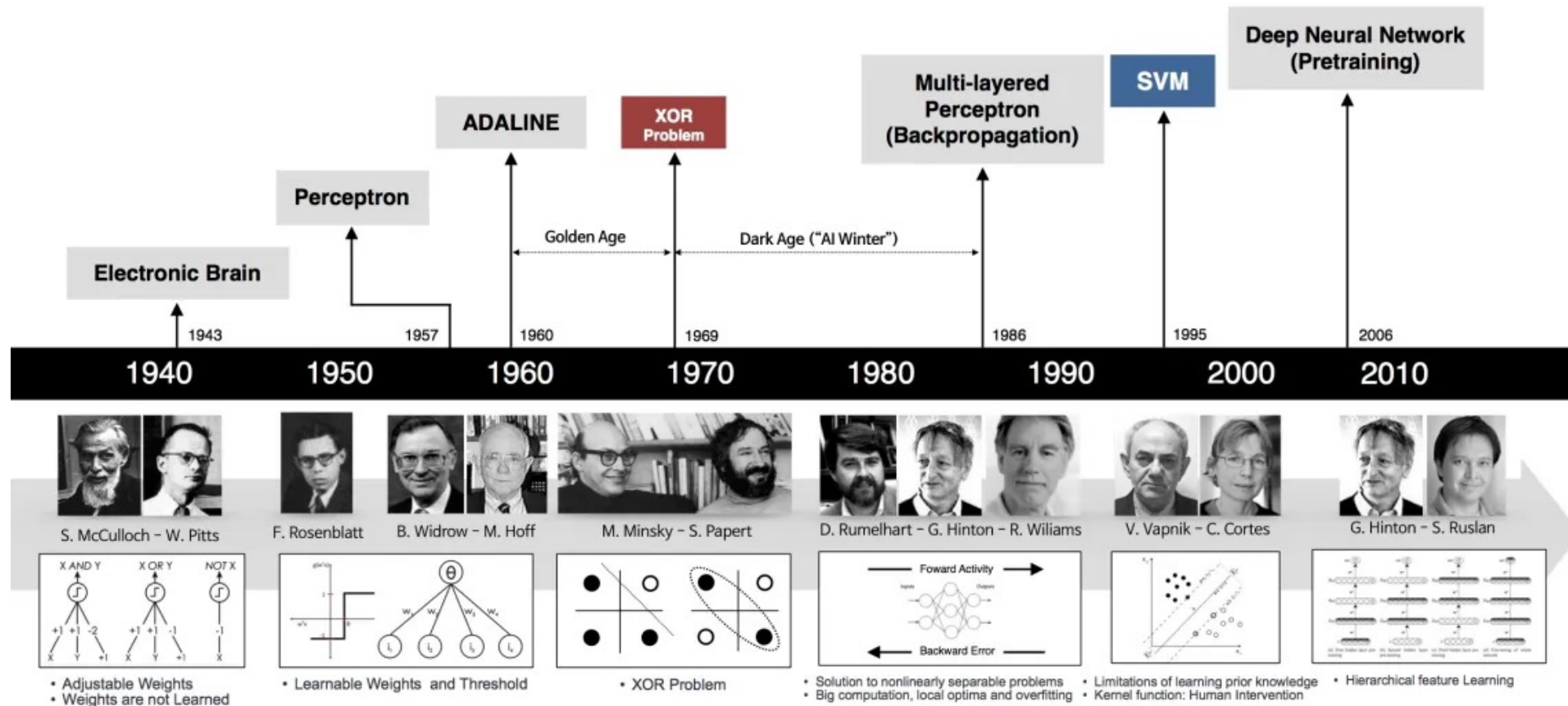
출처: 인공지능과 딥러닝, 마쓰오 유타카 지음

# Deep learning의 역사

## ■ 2018년 Turing Award:

- [요슈아 벤지오](#) / [제프리 힌턴](#) / [안 레쿤](#) (심층 학습)

<https://sefiks.com/2017/10/14/evolution-of-neural-networks/>

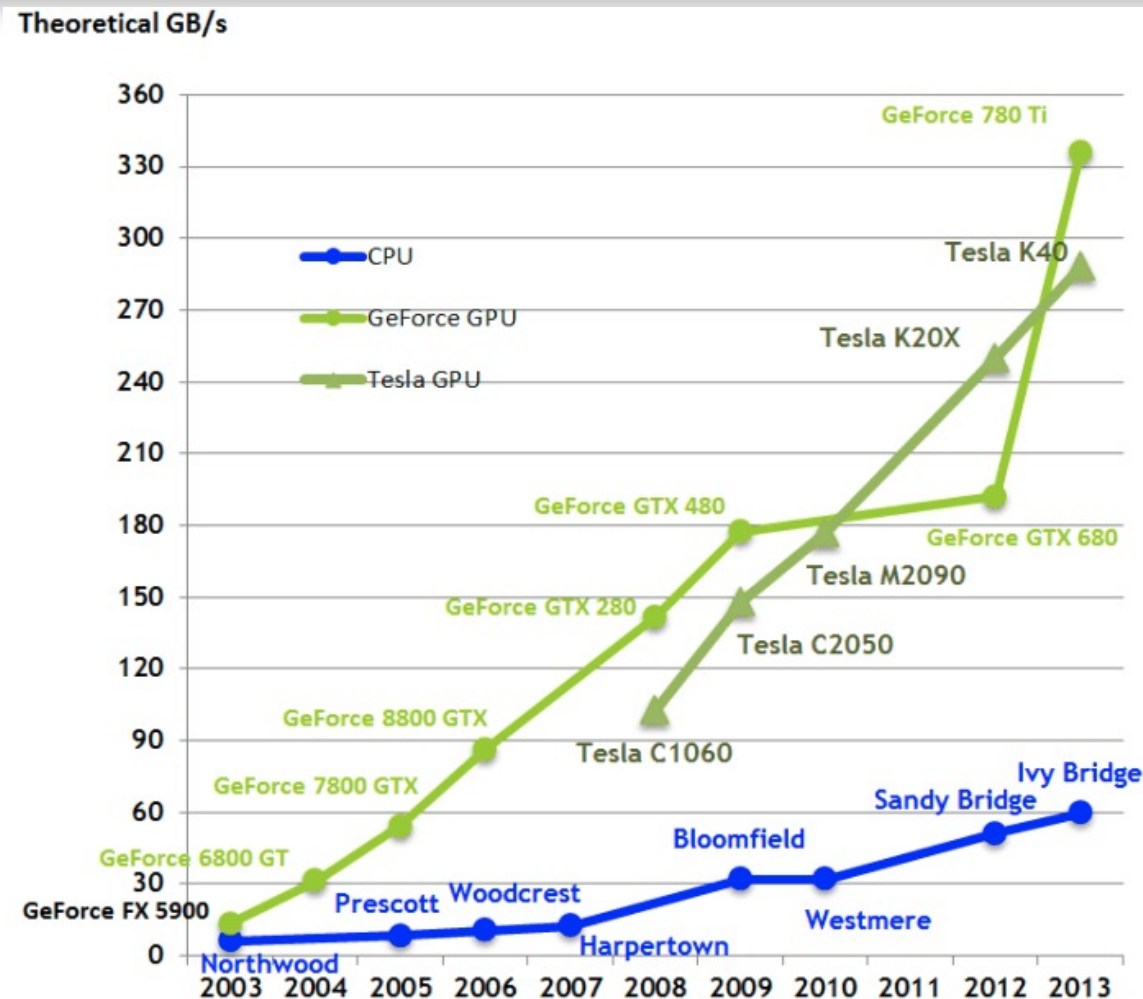




# 3차 AI boom이 가능했던 이유

## ■ 1. 하드웨어 성능 개선과 GPU의 등장

- 계산 속도가 급속도로 빨라짐.
- GPU의 등장으로 병렬 처리가 가능해짐. (NVIDIA-CUDA)



<https://medium.com/@shachishah.ce/do-we-really-need-gpu-for-deep-learning-47042c02efe2>

# CPU vs GPU

## ■ CPU

- 컴퓨터의 두뇌를 담당함.
- 입출력장치, 기억장치, 연산장치를 비롯한 컴퓨터 리소스를 활용하는 중앙처리장치
- 코어의 수는 많지 않으나 (4~16개) 각 코어의 연산능력이 뛰어남.

## ■ GPU

- 픽셀로 이루어진 비디오 영상을 처리하는 용도로 탄생.
- CPU에 비해 반복적이고 비슷한 대량의 정보를 병렬처리할 때 사용.
- 코어의 연산 능력은 떨어지나 코어의 수가 많음 (2080TI – 4352개).

# CPU vs GPU

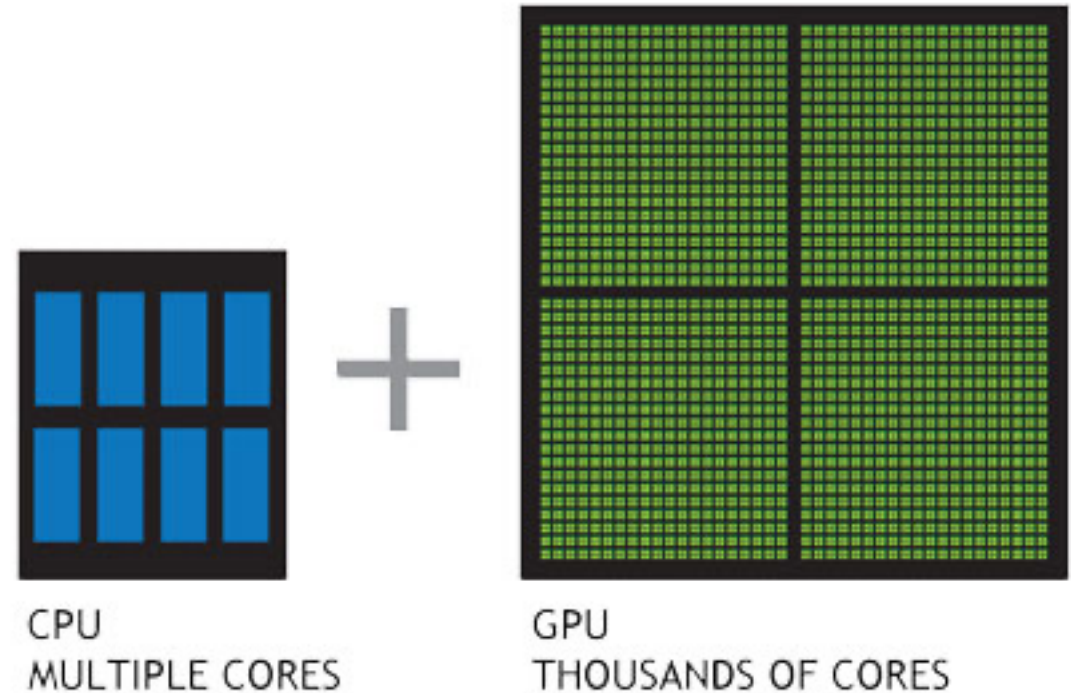
## ■ CPU

- 코어의 수는 적지만 각 코어의 컴퓨팅 파워가 강력 → Sequential task

## ■ GPU

- 각 코어의 컴퓨팅 파워는 약하지만 코어의 수가 매우 많음  
→ Parallel task

## ■ 딥러닝에서는 GPU를 사용한다!



<https://kr.nvidia.com/object/what-is-gpu-computing-kr.html>

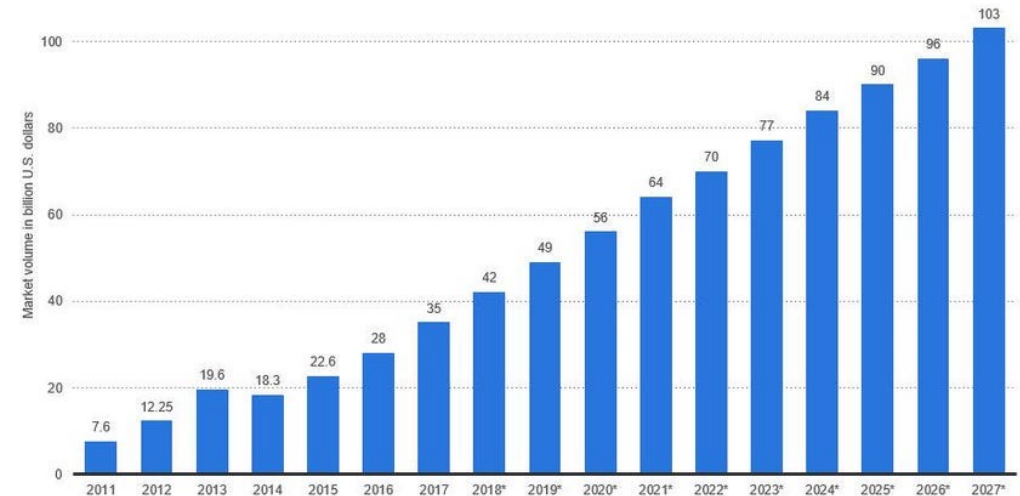
# 3차 AI boom이 가능했던 이유

## ■ 2. 빅데이터의 등장

- 데이터 양의 증가로 인해 모델 학습이 쉬워짐.
- 다양한 데이터의 수집으로 다양한 모델을 만들 수 있게 됨.

Forecast Revenue Big Data Market Worldwide 2011-2027

**Big Data Market Size Revenue Forecast Worldwide From 2011 To 2027**  
(in billion U.S. dollars)

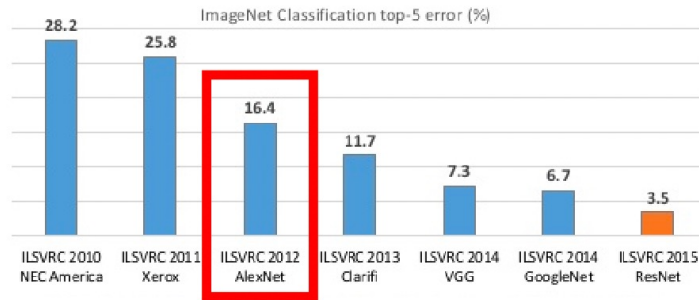


statista

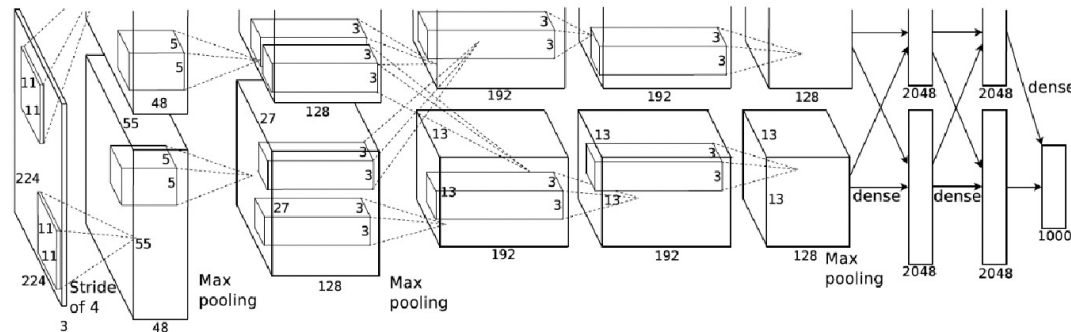
# 3차 AI boom이 가능했던 이유

## ■ 3. 다양한 학습 알고리즘의 등장

- 2012년 AlexNet을 기준으로 이미지 분류 대회에서 딥러닝 모델이 1위를 거둠.



AlexNet의 구조



# 3차 AI boom이 가능했던 이유

## ■ 3. 다양한 학습 알고리즘의 등장

- Batchnorm, dropout 등 학습에 도움이 되는 다양한 알고리즘 개발.

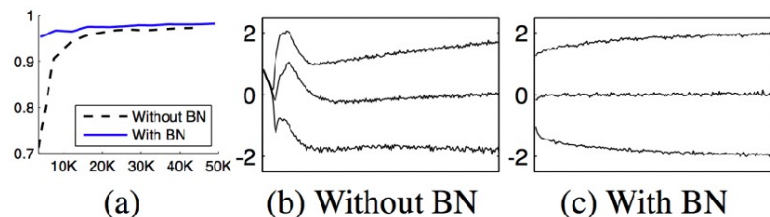
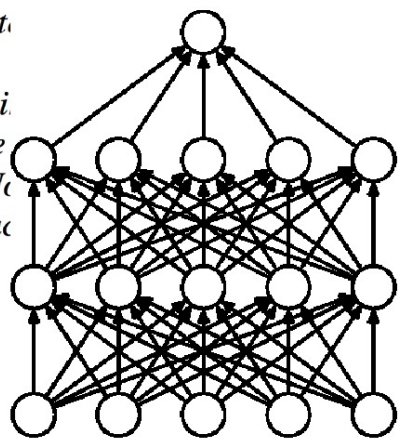
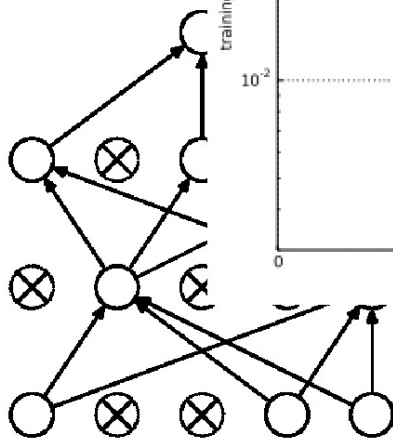


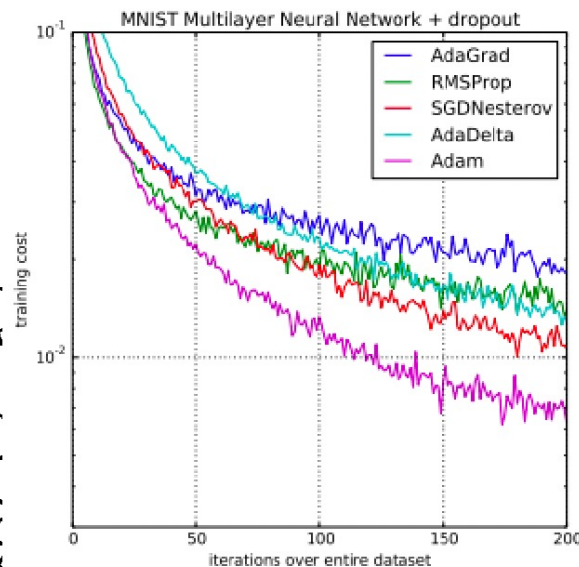
Figure 1: (a) The test accuracy of the MNIST network trained with and without Batch Normalization, vs. the number of training steps. (b) The evolution of i-moid, over the course percentiles. Batch Normalization is more stable and reduces the variance.



(a) Standard Neural Net

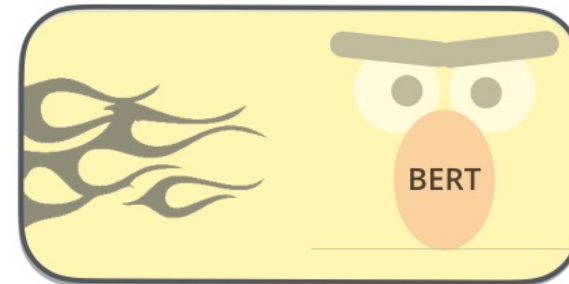
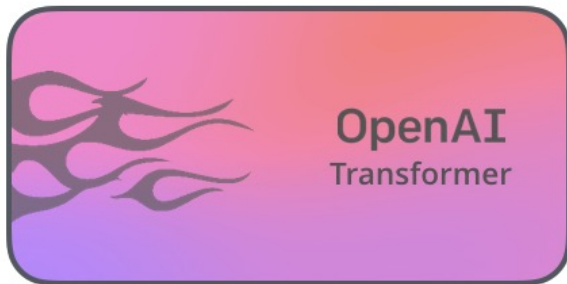


(b) After applying dropout.



# 3차 AI boom이 가능했던 이유

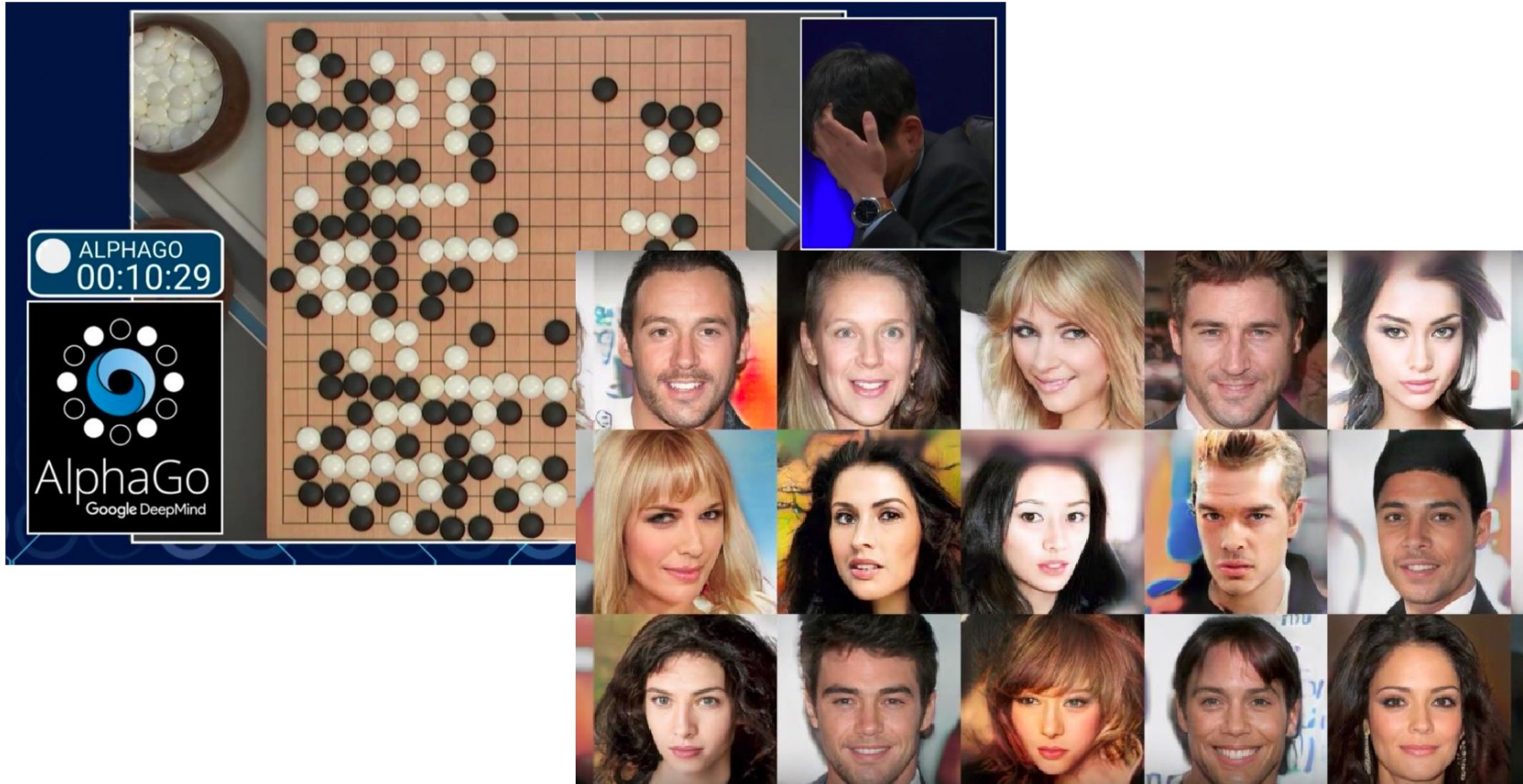
- 3. 다양한 학습 알고리즘의 등장
  - Transformer와 같은 최신 모델의 등장.





# 3차 AI boom이 가능했던 이유

- 4. 사람들의 인식 변화
  - AlphaGo와 GAN을 활용한 이미지 생성.





# INSTALL PYTORCH

# PyTorch 설치하기

## ■ Conda 활용

- <https://pytorch.org>에 들어가서 사용하고 싶은 Pytorch version과 본인 컴퓨터의 OS를 골라주기만 하면 아래 conda 실행 명령어가 나옴!
- 이를 Terminal에 복사해서 입력하기.

### PYTORCH 설치하기

사용 환경을 선택하고 설치 명령을 복사해서 실행해 보세요. Stable 버전은 테스트 및 지원되고 있는 가장 최근의 PyTorch 버전으로, 대부분의 사용자에게 적합합니다. Preview 버전은 아직 완전히 테스트되지 않은 최신 1.11 버전으로 매일 업데이트됩니다. 사용 중인 패키지 매니저에 따라 **아래의 사전 요구사항(예: numpy)**이 충족되었는지 확인해 주세요. 모든 의존성을 설치할 수 있는 Anaconda를 패키지 매니저로 추천합니다. **이전 버전의 PyTorch도 설치할 수 있습니다.** LibTorch는 C++에서만 지원합니다.

PyTorch Enterprise Support Program을 통해 Stable 및 LTS 바이너리에 대한 추가적인 지원 / 보증이 가능합니다.

PyTorch 빌드	Stable (1.10)	Preview (Nightly)	LTS (1.8.2)	
OS 종류	Linux	Mac	Windows	
패키지 매니저	Conda	Pip	LibTorch	Source
언어	Python	C++ / Java		
플랫폼	CUDA 10.2	CUDA 11.3	ROCm 4.2 (beta)	CPU
이 명령을 실행하세요:	conda install pytorch torchvision torchaudio cudatoolkit=11.3 -c pytorch			

# PyTorch 설치하기

## ■ Conda 활용

- <https://pytorch.org>에 들어가서 사용하고 싶은 Pytorch version과 본인 컴퓨터의 OS를 골라주기만 하면 아래 conda 실행 명령어가 나옴!
- 이를 Terminal에 복사해서 입력하기.

### PYTORCH 설치하기

사용 환경을 선택하고 설치 명령을 복사해서 실행해 보세요. Stable 버전은 테스트 및 지원되고 있는 가장 최근의 PyTorch 버전으로, 대부분의 사용자에게 적합합니다. Preview 버전은 아직 완전히 테스트되지 않은 최신 1.11 버전으로 매일 업데이트됩니다. 사용 중인 패키지 매니저에 따라 **아래의 사전 요구사항(예: numpy)**이 충족되었는지 확인해 주세요. 모든 의존성을 설치할 수 있는 Anaconda를 패키지 매니저로 추천합니다. **이전 버전의 PyTorch도 설치할 수 있습니다.** LibTorch는 C++에서만 지원합니다.

PyTorch Enterprise Support Program을 통해 Stable 및 LTS 바이너리에 대한 추가적인 지원 / 보증이 가능합니다.

PyTorch 빌드	Stable (1.10)	Preview (Nightly)	LTS (1.8.2)	
OS 종류	Linux	Mac	Windows	
패키지 매니저	Conda	Pip	LibTorch	Source
언어	Python	C++ / Java		
플랫폼	CUDA 10.2	CUDA 11.3	ROCm 4.2 (beta)	CPU
이 명령을 실행하세요:	conda install pytorch torchvision torchaudio cudatoolkit=11.3 -c pytorch			

# PyTorch 설치하기

## ■ Colab

- 별도의 설치 없이 import torch만으로 활용 가능.

✓ [1] 1 import torch  
5초

✓ [2] 1 torch.\_\_version\_\_  
0초  
  
'1.10.0+cu111'

## ■ 런타임 유형 변경

- 런타임>런타임 유형 변경>하드웨어 가속기 GPU



# PyTorch 설치하기

## ■ Colab

- 파이썬 버전
  - !python -version
- 패키지 버전
  - !pip list
  - !pip freeze
- GPU 정보 (nvidia 드라이버, CUDA)
  - !nvidia-smi

1 !nvidia-smi

Mon Mar 21 08:33:29 2022

NVIDIA-SMI 460.32.03				Driver Version: 460.32.03		CUDA Version: 11.2	
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr.	ECC
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute	M. MIG M.
0	Tesla K80	Off	00000000:00:04.0	Off	0%	Default	0
N/A	33C	P8	27W / 149W	0MiB / 11441MiB		N/A	

Processes:							
GPU	GI	CI	PID	Type	Process name	GPU Memory	Usage
	ID	ID					
No running processes found							