※본 수업자료는 “Lecture Slide of InfolabAI (https://github.com/InfolabAI/DeepLearning)” 기반으로 제작되었습니다.

의료와 데이터사이언스 10 주차

(비정형 데이터 생체신호 분석 2: 생체신호AI 모델링1)

Hyun-Lim Yang, PhD

Assistant Professor

Office of Hospital Information
Seoul National University Hospital

Adjunct Assistant Professor

Department of Medicine
Seoul National University

SNUH
서울대학교
SEOUL NATIONAL UNIVERSITY

CONTENTS

Recap: Pandas

SNUH

- **DataFrame**
  - R의 dataframe 데이터 타입과 유사
  - Tabular data type을 지원하여 정형 데이터를 분석하기 쉬움
  - Columns / Row (data) / Index

```
1 import pandas as pd
```

```
1 df_dic = pd.DataFrame({"A":[1,4,7], "B":[2,5,8], "C":[3,6,9]})
2 df_dic
```

	A	B	C
0	1	2	3
1	4	5	6
2	7	8	9

```
1 import numpy as np
```

```
1 df_np = pd.DataFrame(np.array([[1,2,3], [4,5,6], [7,8,9]]))
2 df_np
```

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

Recap: Pandas

- **Index by columns**
  - 컬럼 이름으로 바로 인덱싱 가능

```
df_dic['A']
```

```
0    1
1    4
2    7
Name: A, dtype: int64
```

```
df_dic[['A', 'B']]
```

	A	B
0	1	2
1	4	5
2	7	8

- **Index by condition**
  - 조건을 통해 row (data) 를 indexing 가능

```
row_index = np.array([True, False, True])
df_dic[row_index]
```

	A	B	C
0	1	2	3
2	7	8	9

```
df_dic[df_dic.index % 2 == 0]
```

	A	B	C
0	1	2	3
2	7	8	9

Recap: Pandas

SNUH

- **Get index of DataFrame**
  - DataFrame의 index를 반환함

```
df_np.index
```

```
RangeIndex(start=0, stop=3, step=1)
```

```
list(df_np.index)
```

```
[0, 1, 2]
```

- **Get value of DataFrame**
  - DataFrame의 값을 array 형태로 반환함

```
df_np.values
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

```
df_dic['A'].values
```

```
array([1, 4, 7])
```

```
df_dic[['A', 'C']].values
```

```
array([[1, 3],
       [4, 6],
       [7, 9]])
```

Recap: Pandas

- **notnull()**
  - na 혹은 NaN 값 비교를 하여 boolean을 반환함 (없으면 True)
- **fillna()**
  - na 혹은 NaN 값의 위치에 모두 특정 값을 대입함

```
df_na = pd.DataFrame({"A":[1,np.nan,7],
                      "B":[2,5,np.nan],
                      "C":[3,6,9]})
df_na
```

	A	B	C
0	1.0	2.0	3
1	NaN	5.0	6
2	7.0	NaN	9

```
df_na.notnull()
```

	A	B	C
0	True	True	True
1	False	True	True
2	True	False	True

```
df_na[df_na['A'].notnull() & df_na['B'].notnull()]
```

	A	B	C
0	1.0	2.0	3

```
df_fna = pd.DataFrame({"A":[1,np.nan,7],
                       "B":[2,5,np.nan],
                       "C":[3,6,9]})
df_fna
```

	A	B	C
0	1.0	2.0	3
1	NaN	5.0	6
2	7.0	NaN	9

```
df_fna.fillna(0)
```

	A	B	C
0	1.0	2.0	3
1	0.0	5.0	6
2	7.0	0.0	9

```
df_fna.fillna('not available')
```

	A	B	C
0	1	2	3
1	not available	5	6
2	7	not available	9

Keras Install

- **High-level deep learning API for python**
- **Originally support TensorFlow, Theano, and CNTK as backend engine**
- **Now officially supported by TensorFlow 2.0**

TensorFlow 2 설치

TensorFlow는 다음 64비트 시스템에서 테스트 및 지원됩니다.

- Python 3.6~3.9
- Ubuntu 16.04 이상
- Windows 7 이상(C++ 재배포 가능 패키지)
- macOS 10.12.6(Sierra) 이상(GPU 지원 없음)

패키지 다운로드

Python의 *pip* 패키지 관리자를 사용해 TensorFlow를 설치하세요.

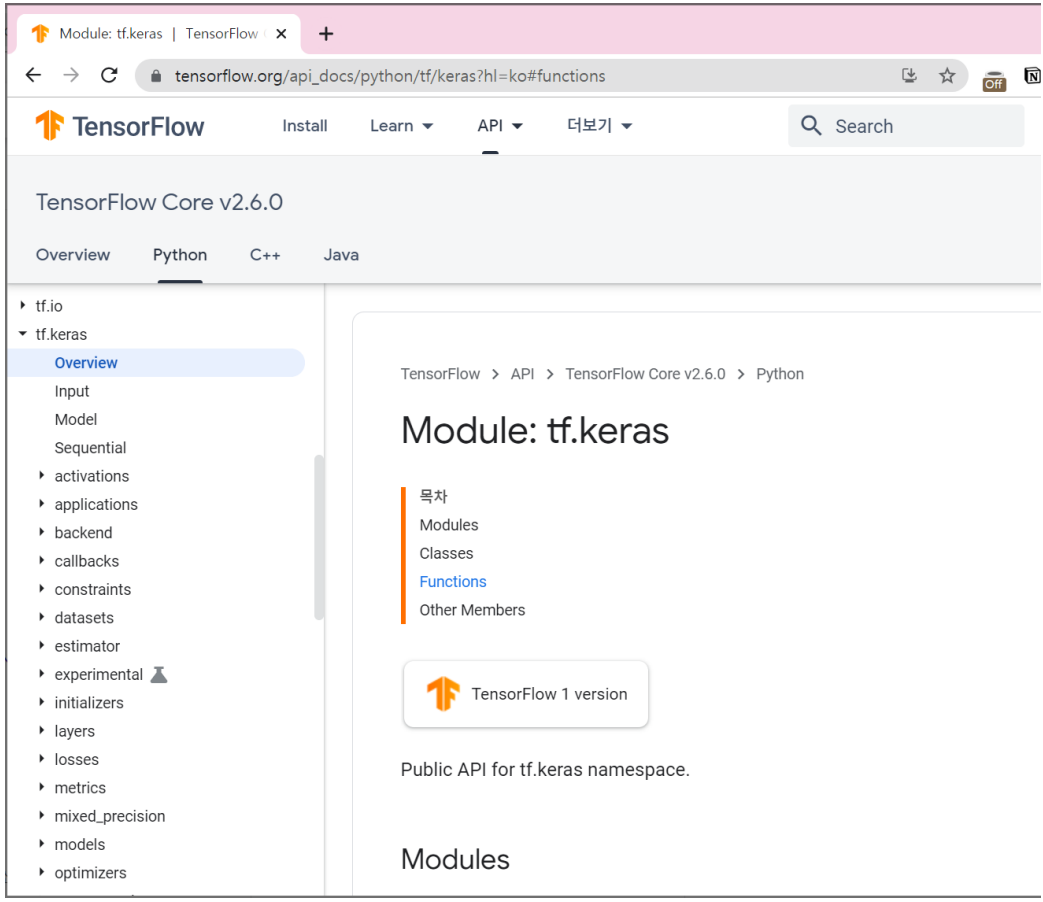
★ TensorFlow 2 패키지에는 **pip** 19.0가 넘는 버전(또는 macOS의 경우 20.3이 넘는 버전)가 필요합니다.

공식 패키지는 Ubuntu, Windows, macOS에서 사용할 수 있습니다.

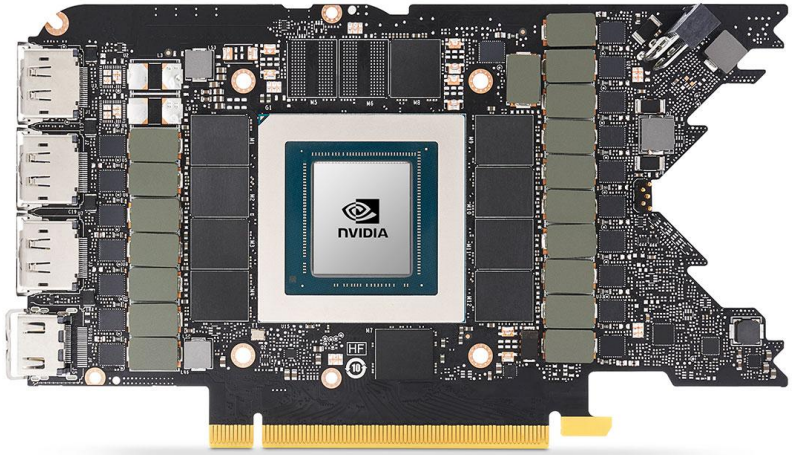
```
# Requires the latest pip
$ pip install --upgrade pip

# Current stable release for CPU and GPU
$ pip install tensorflow

# Or try the preview build (unstable)
$ pip install tf-nightly
```



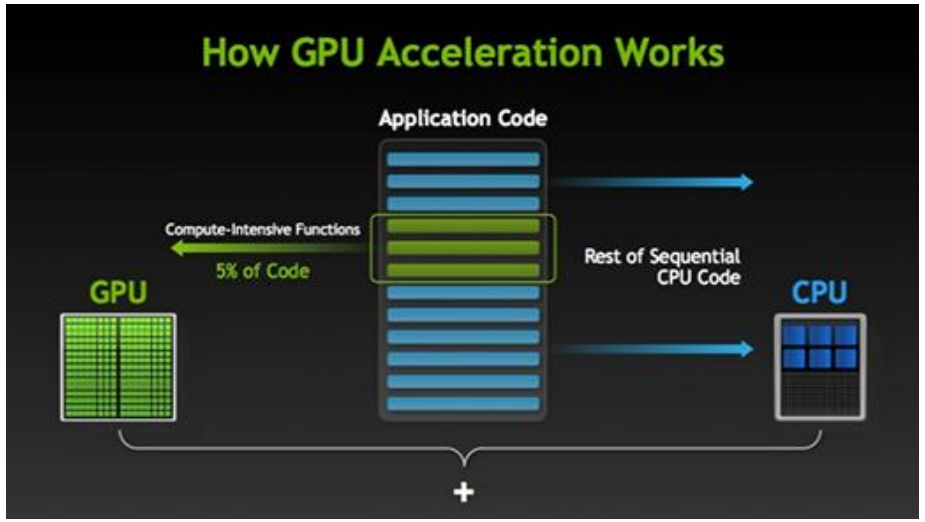
CPU & GPU

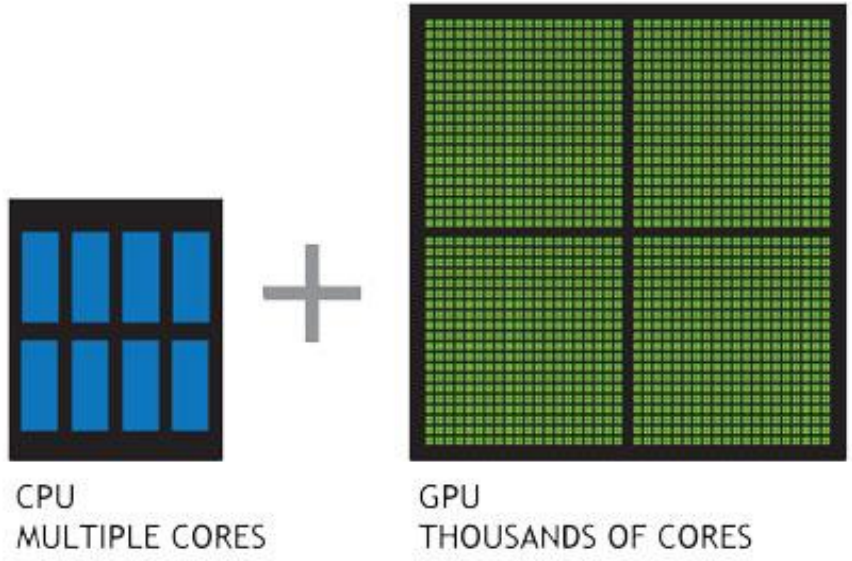


What is GPU computing?

SNUH

- Accelerate computing power with GPU + CPU
- Parallelize the algorithm and software according to the GPU architecture





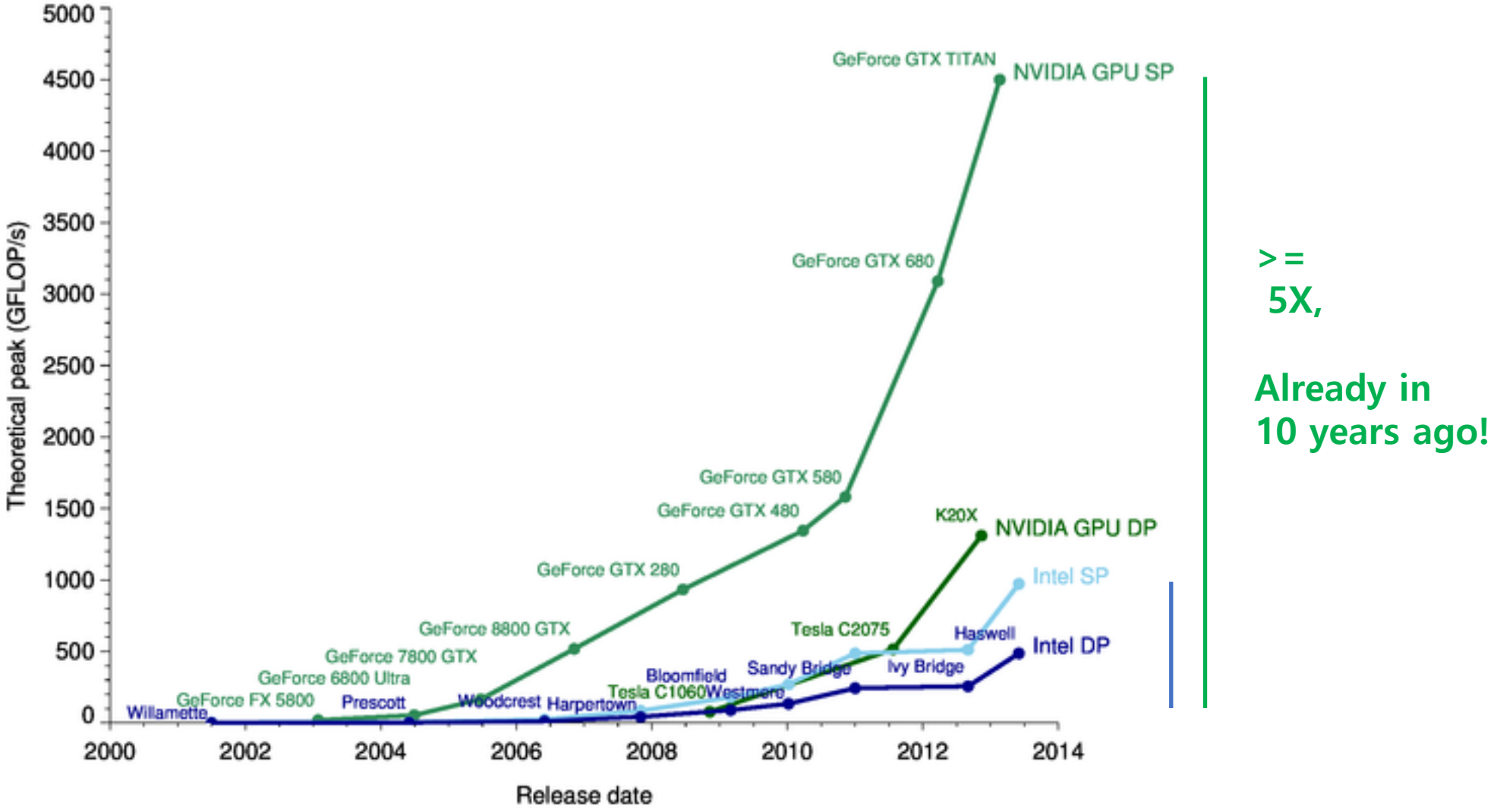
Why GPU computing?

SNUH

- **Required percormance of processor is dramatically increased**
- **Serial performance scaling is Over**
  - **cannot** continue to scale processor frequencies (no 10GHz chip)
  - **cannot** continue to increase power consumption
  - **can** continue to increase transistor density
- **Concurrency revolution**

CPU vs. GPU

SNUH



> =
5X,

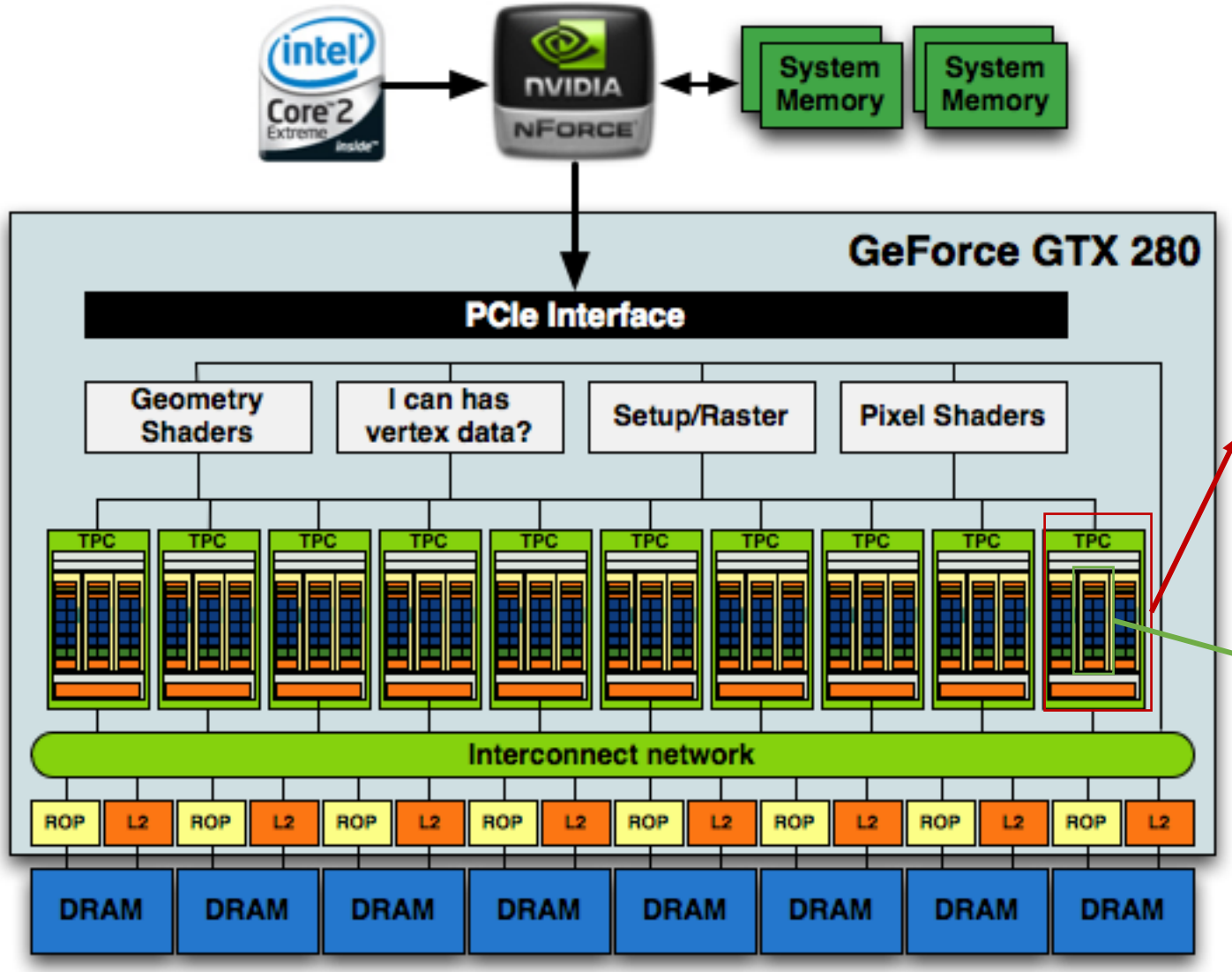
Already in
10 years ago!

New consideration

SNUH

- **Computers no longer get faster, just wider**
- **We must re-think our algorithm to be parallel**
- **How to be parallel?**
  - Instruction-level parallelism
    - e.g.) out-of-order execution
  - **Data-level parallelism – most scalable solution**
    - e.g.) vector unit operation
  - Thread-level parallelism
    - e.g.) multithreading

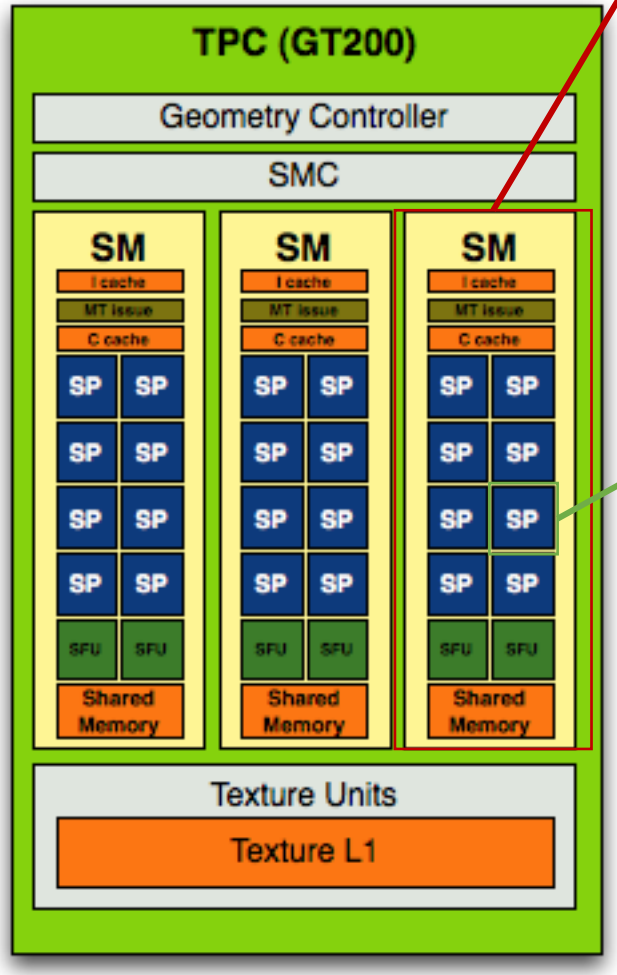
GPU architecture



TPC
(Texture Processor Cluster)

1 SM
(streaming multiprocessor)

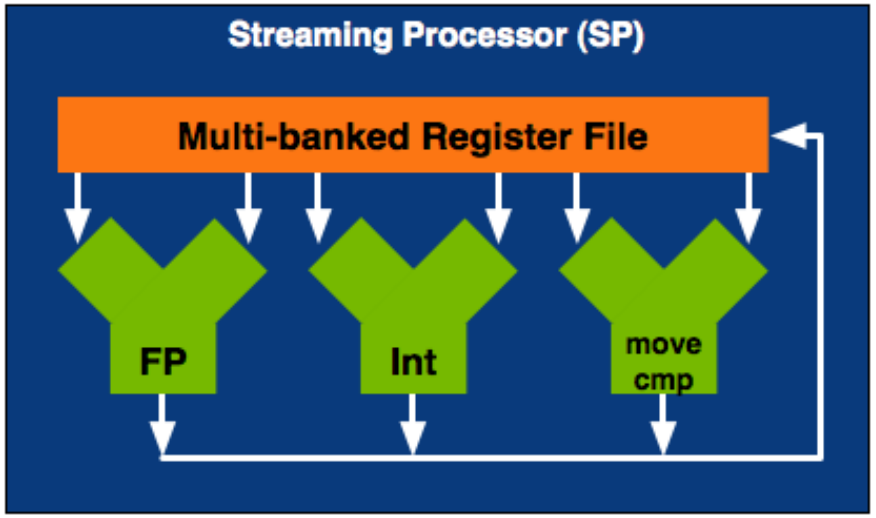
GPU architecture



1 SM (Streaming multiprocessor)

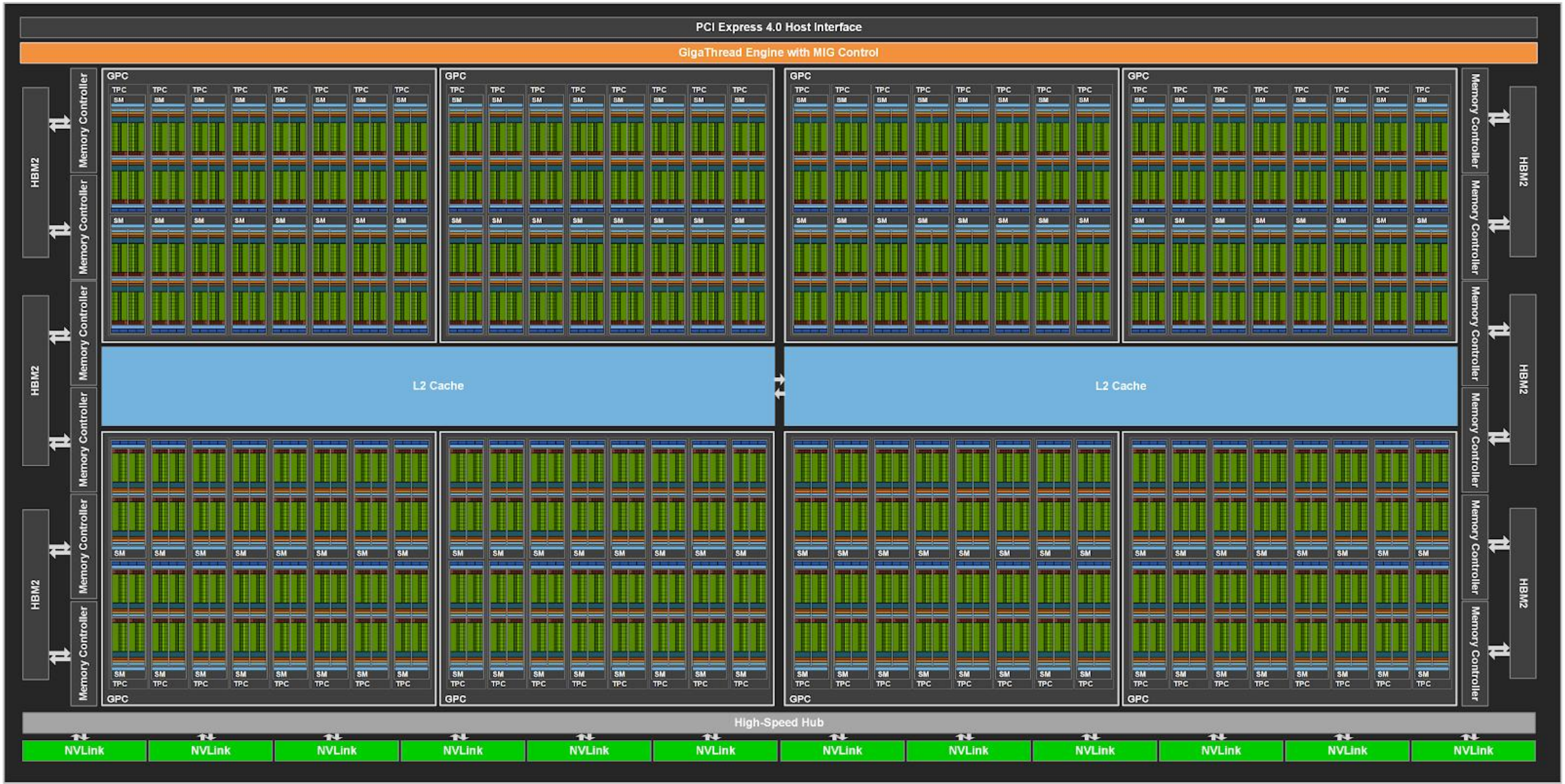
- Direct load/store to memory
- Usual linear sequence of bytes
- High Bandwidth

1 SP (streaming processor)



GPU architecture: A100 (in 2020)

SNUH



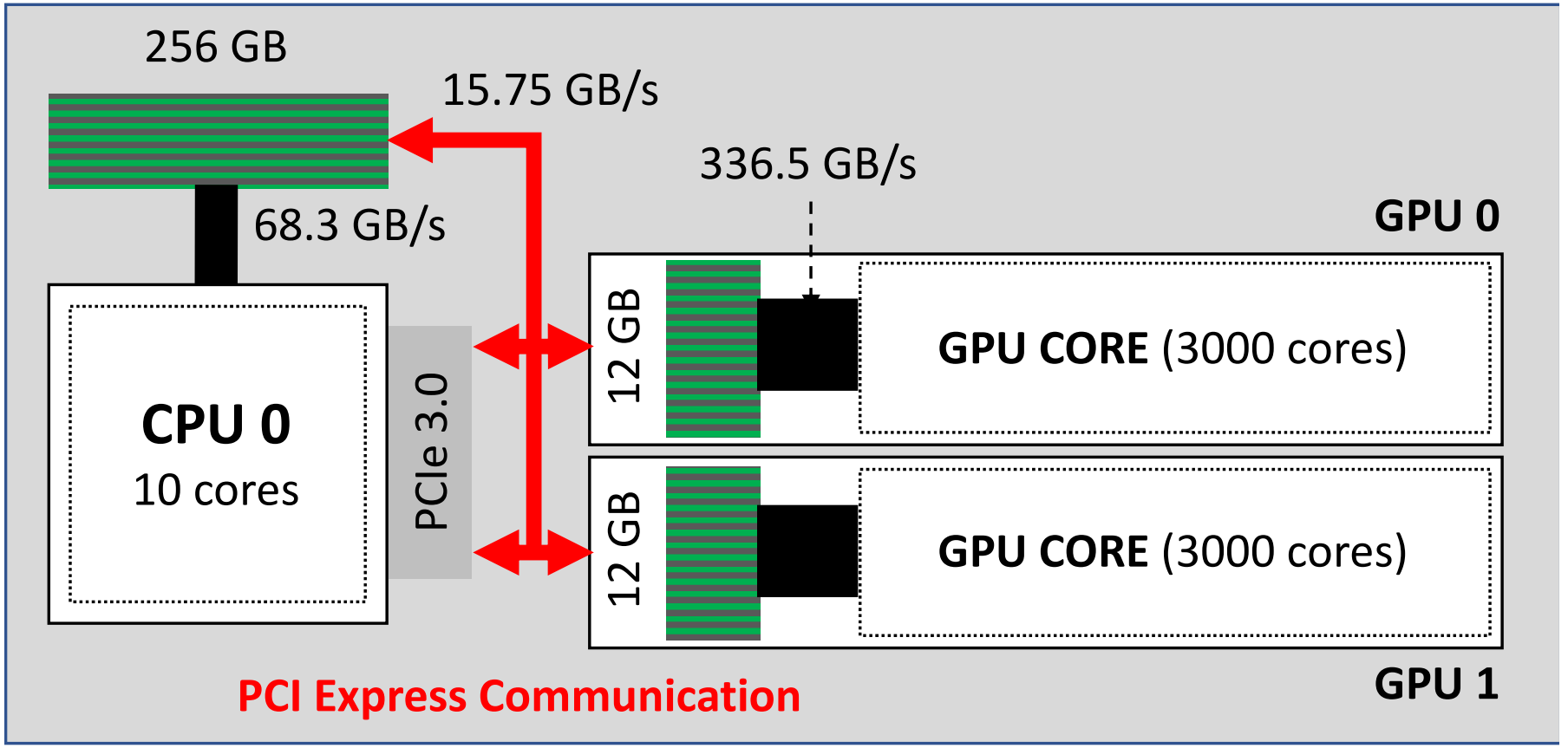
GPU architecture: A100 (in 2020)

→ **Streaming Multiprocessor (SM)**

Streaming Processor (SP)

Computation bottleneck in CPU-GPU

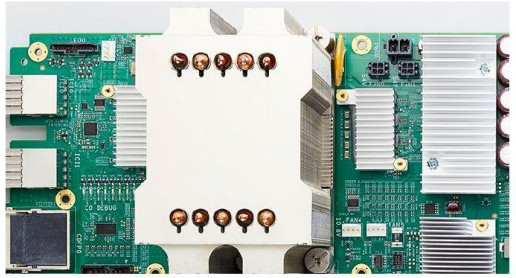
SNUH



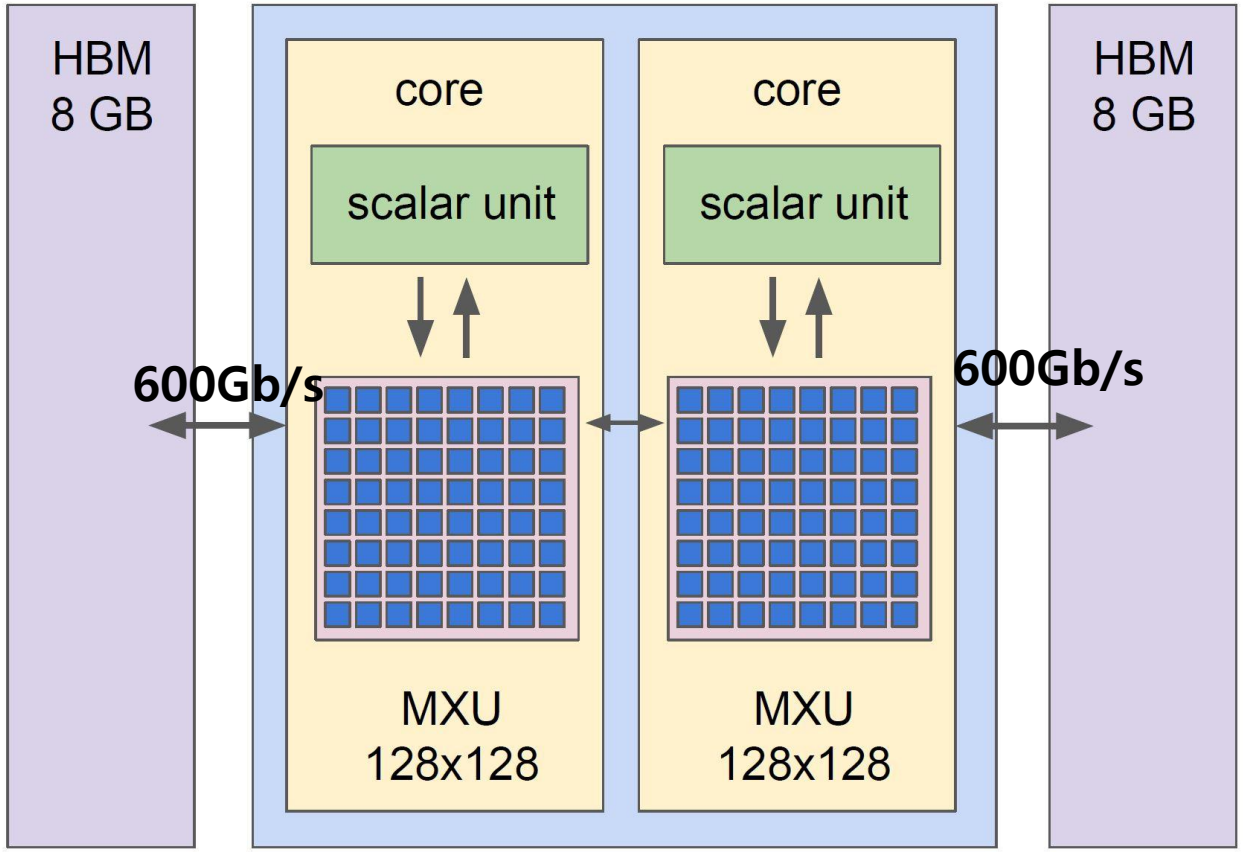
Cf. TPU architecture

SNUH

TPUv2 Chip



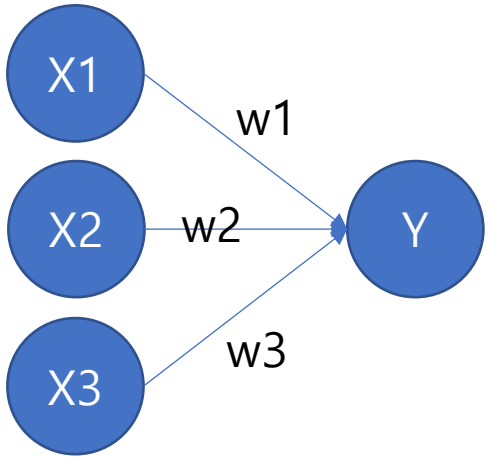
- 16 GB of HBM
- 600 GB/s mem BW
- Scalar unit: 32b float
- MXU: 32b float accumulation but reduced precision for multipliers
- 45 TFLOPS



GPGPU and Deep Learning

SNUH

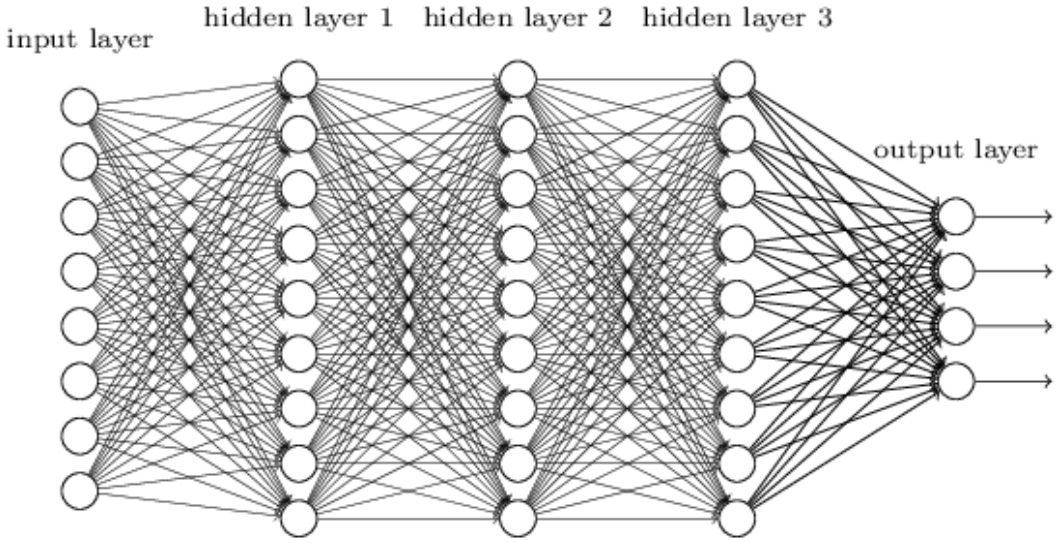
- **Artificial Neuron Operation**



$$y = f(w_1x_1 + w_2x_2 + w_3x_3)$$

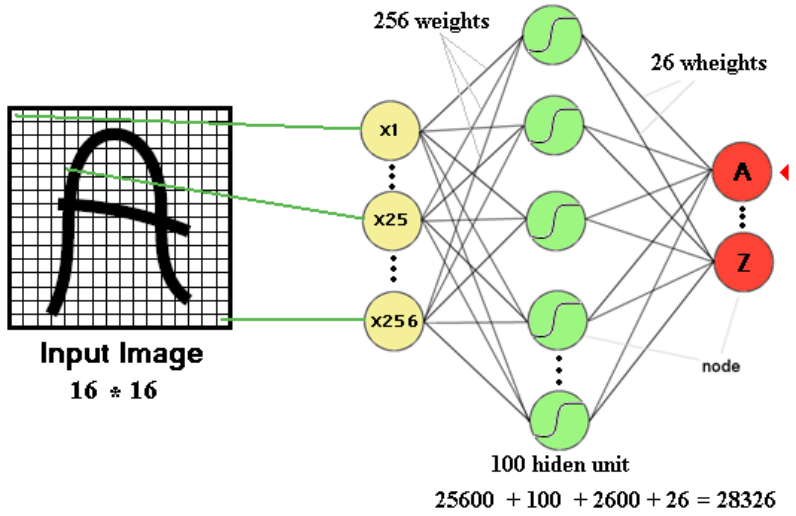
$$f(x) : activation\ function$$

- **Deep Neural Network**



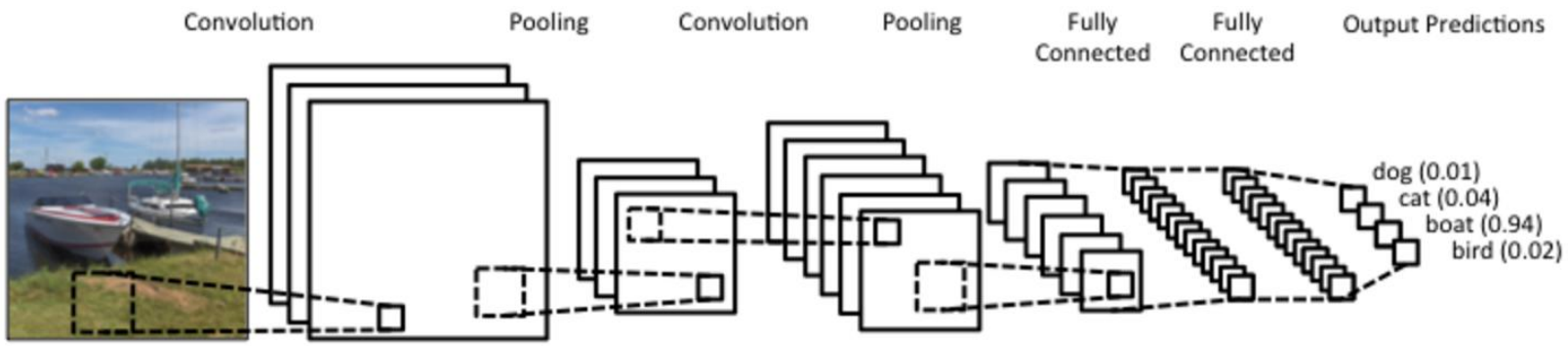
Multi-Layer Perceptron (MLP) for image recognition task

- Use intensity of each pixel as 1D input vector
- There are several problems such as:
  - Required extremely large number of parameter
  - No invariance to shifting, scaling
  - Unable to deal with variable size input data



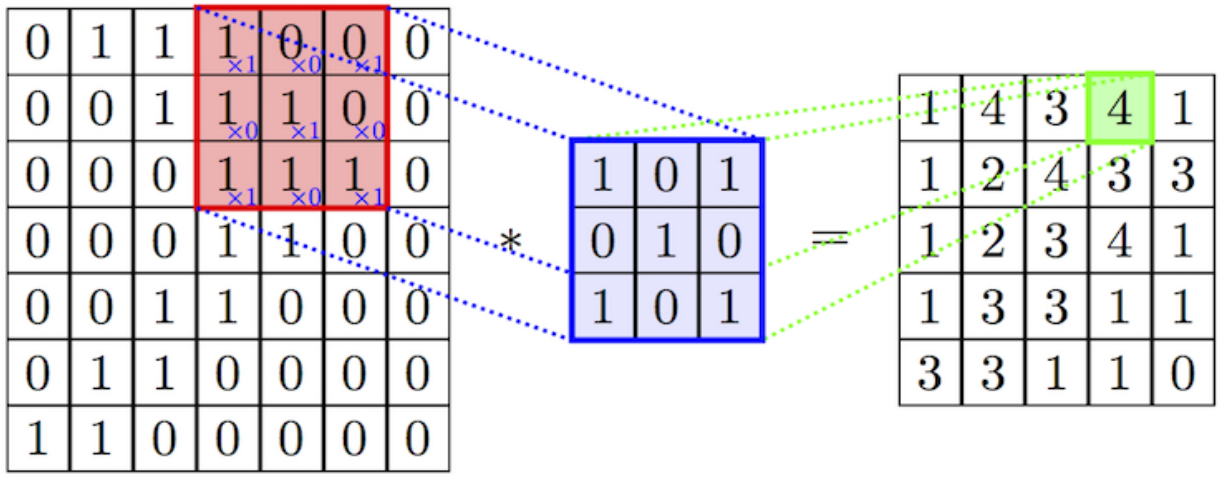
Convolutional Neural Network (CNN)

- CNN consists of:
  - Convolutional layer
  - Pooling layer
  - Fully connected layer
- CNN learns convolutional layer's kernel
  - To make computation easier , CNN uses cross-correlation instead of convolution



2D convolution

- Convolutional Neural Network (CNN) learns optimized weight values of kernel

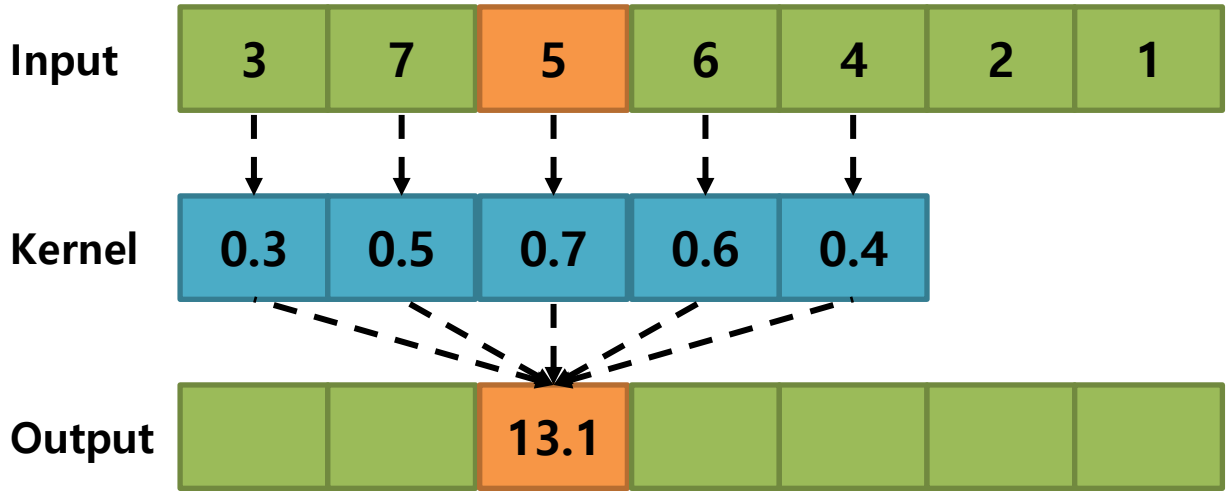


source : https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html

Convolution in Deep Learning (Cross-correlation)

SNUH

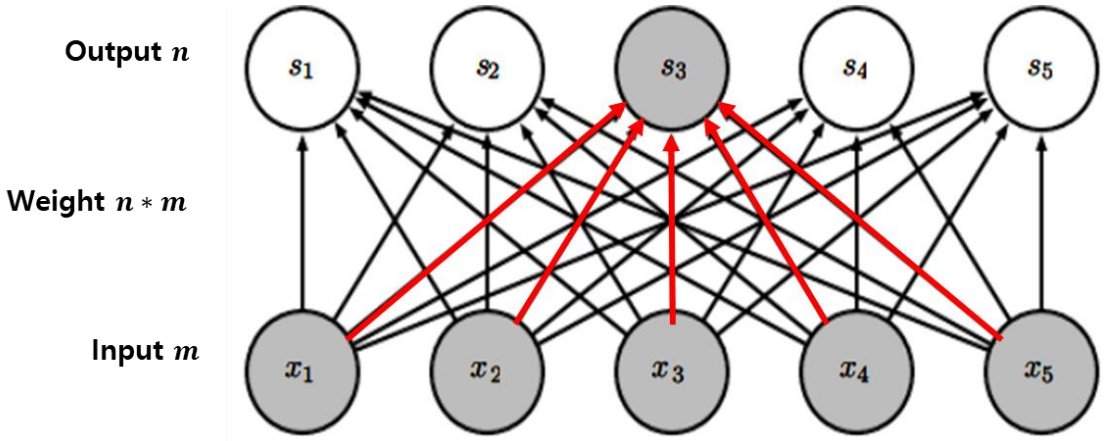
- **Convolutional operations**

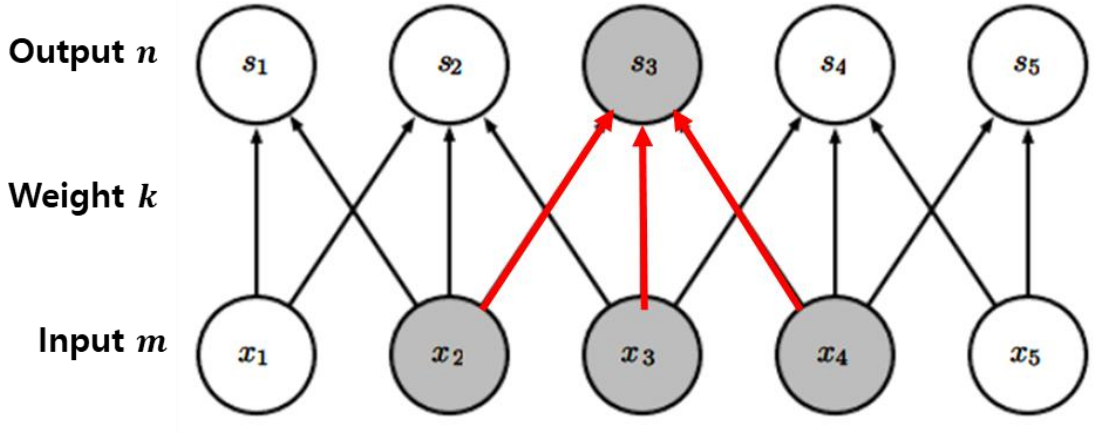


Sparse interaction

- Neurons of input layer and hidden layer are “fully connected”
- Complexity is $O(n * m)$

- To make the kernel smaller than the input causes sparsity
- Complexity is $O(n * k)$
- Fewer parameters are required to be stored and computed



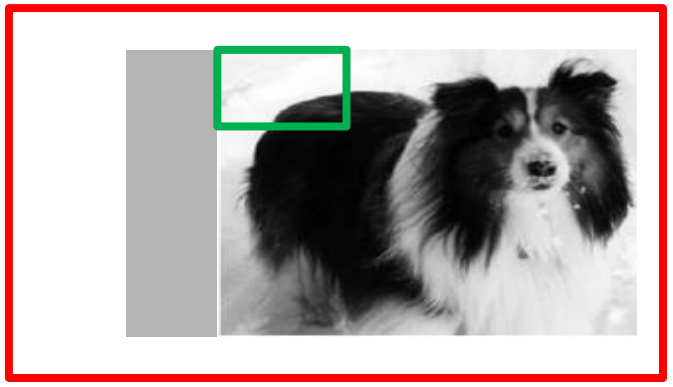


Equivariant representation

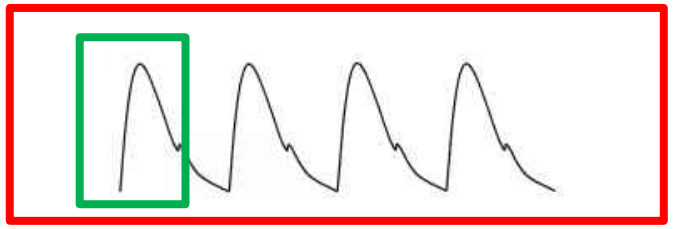
- **Convolution is equivariant operation about translate because it exploits shared kernel and stride from the scratch to the end**

Kernel Data

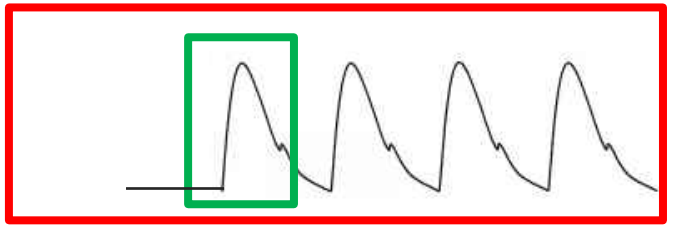
Dog in the center of the image



Dog in the right side of the image

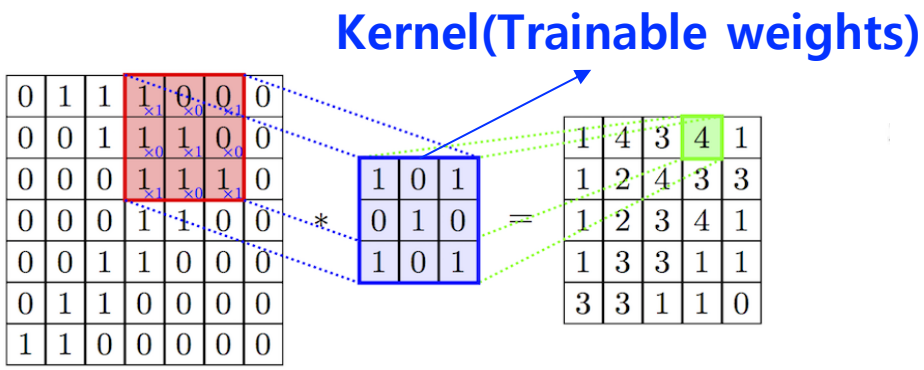


Beats in the center of the segments

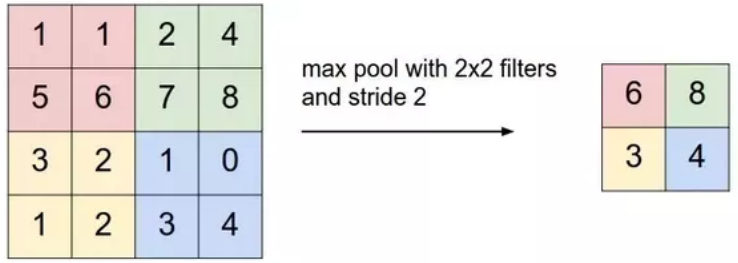


Beats in the aft-time of the segments

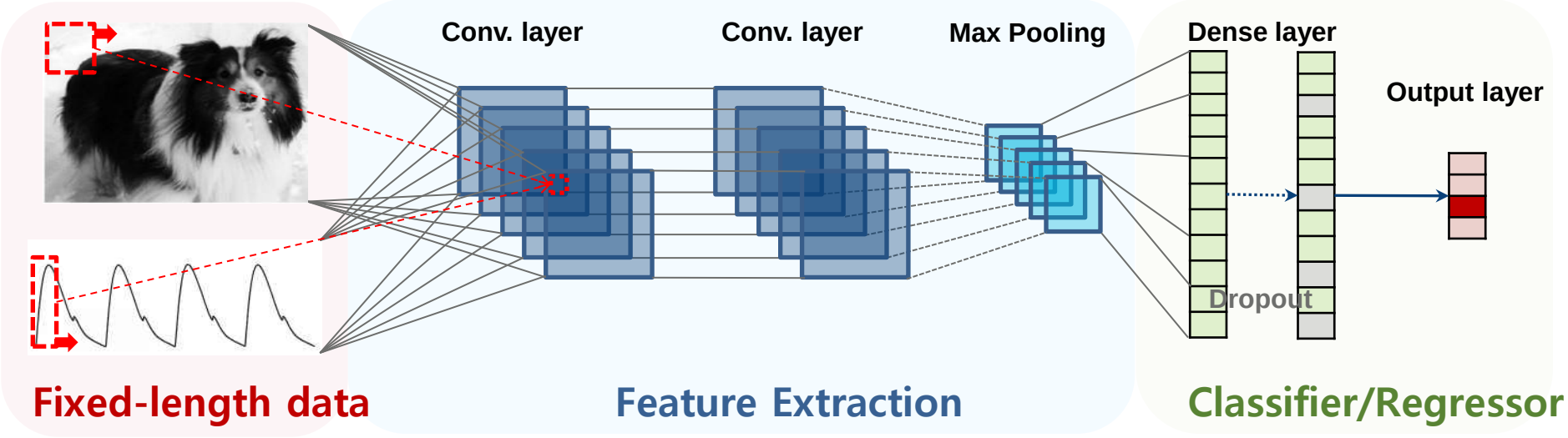
CNN overall



< Convolution operation >



< Max-pooling operation >



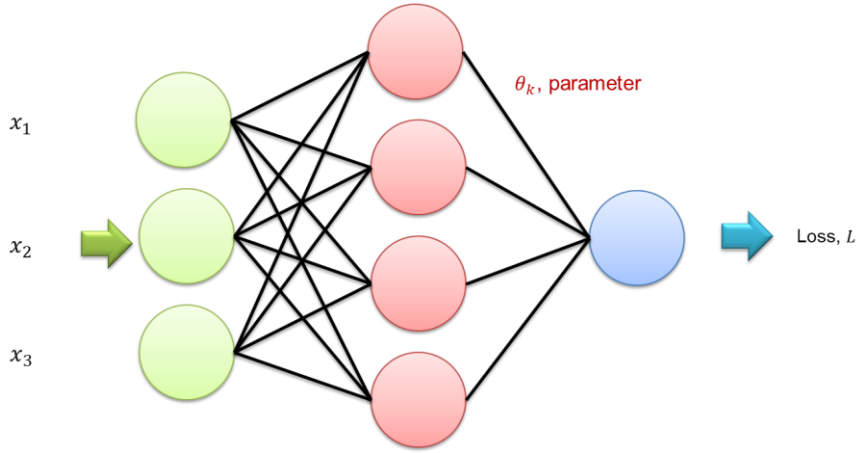
Model Training (dataset split)

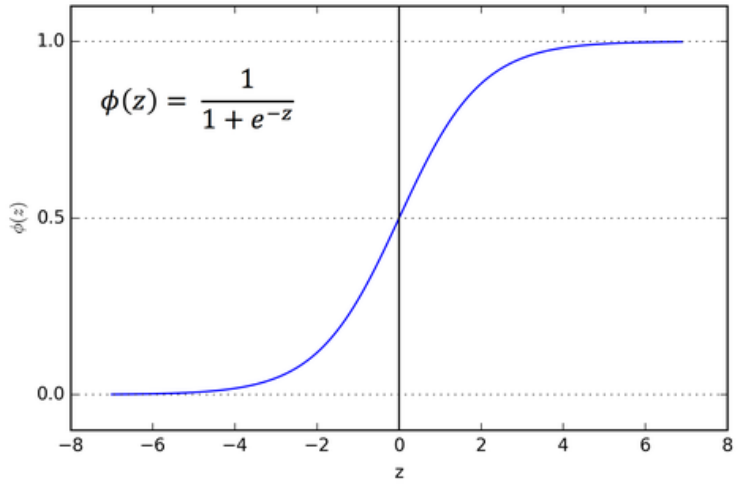
SNUH

- **Training set** is used to train the model.
- **Validation set** is used to measure the performance of the model during training. It is used **repeatedly**.
- **Testing set** is used to measure the expected performance of the model after training. It is **only** used **once**.

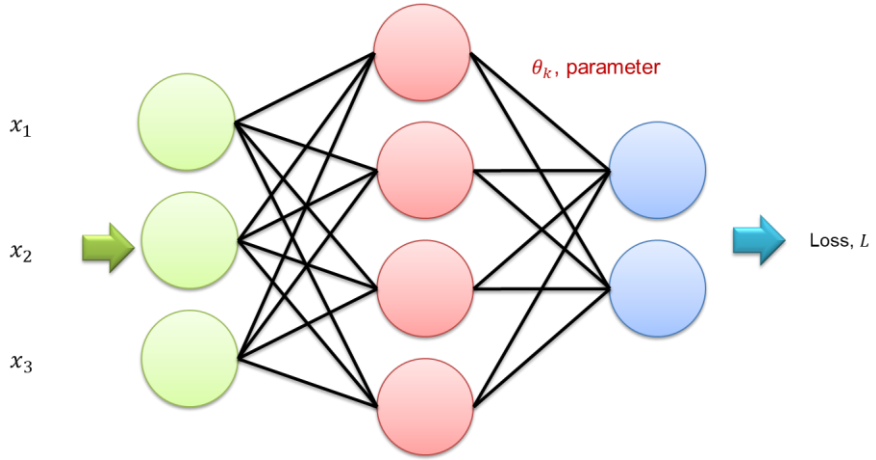
Model Training (output layers)

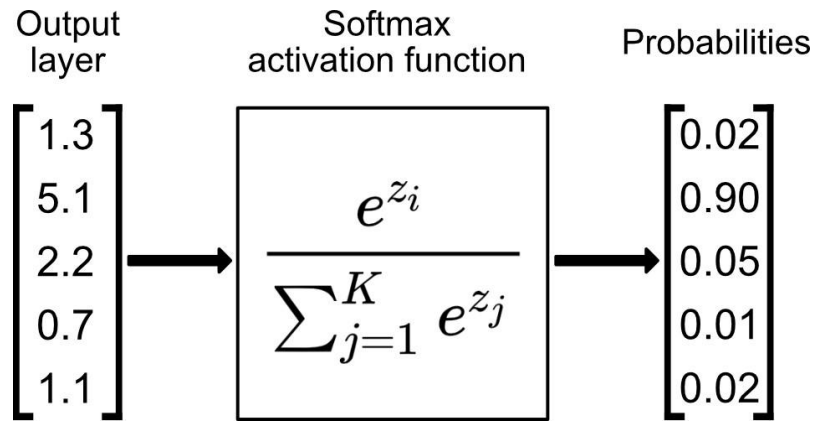
- **Sigmoid function** is used for binary classification





- **Softmax function** is used for multiclass classification

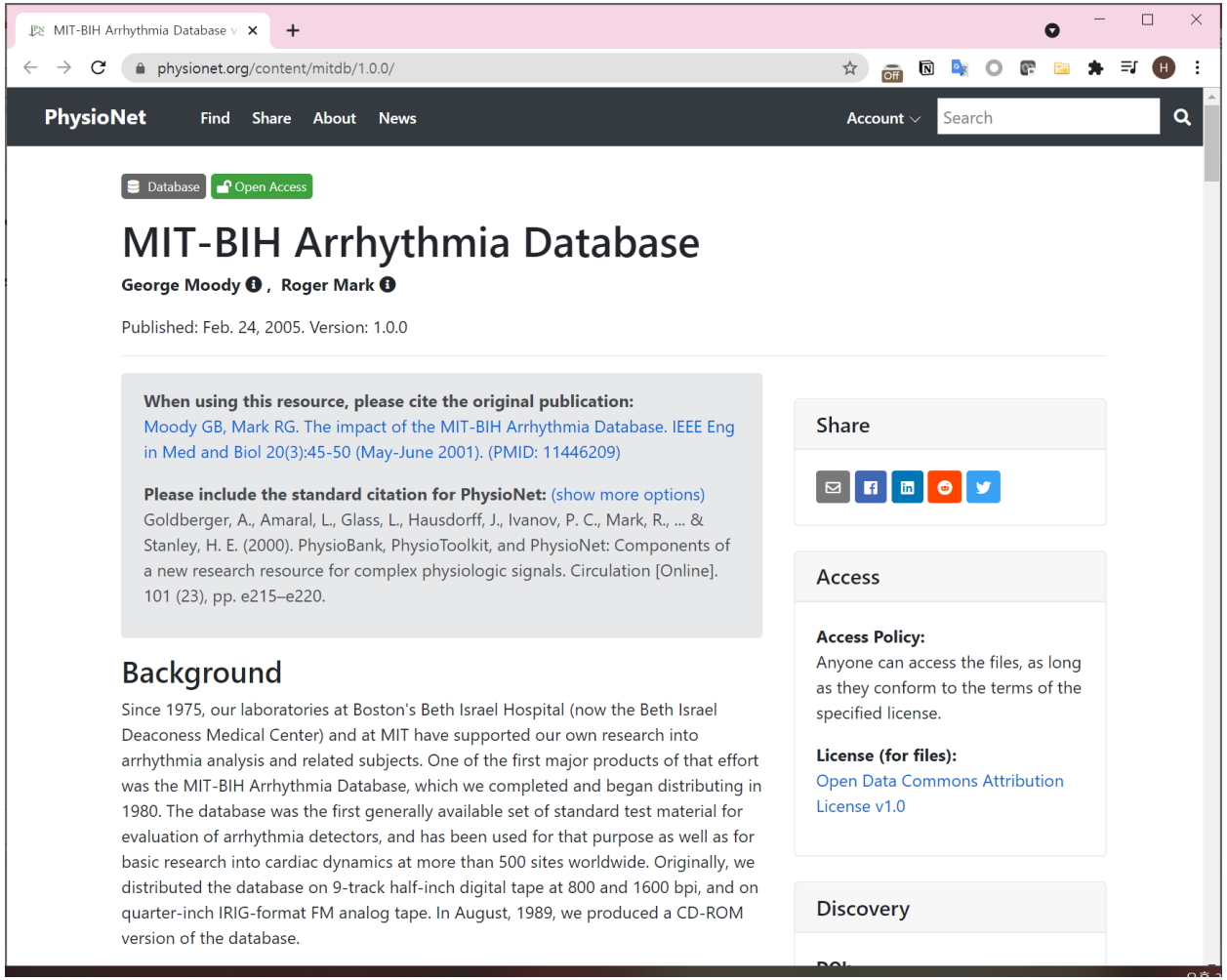


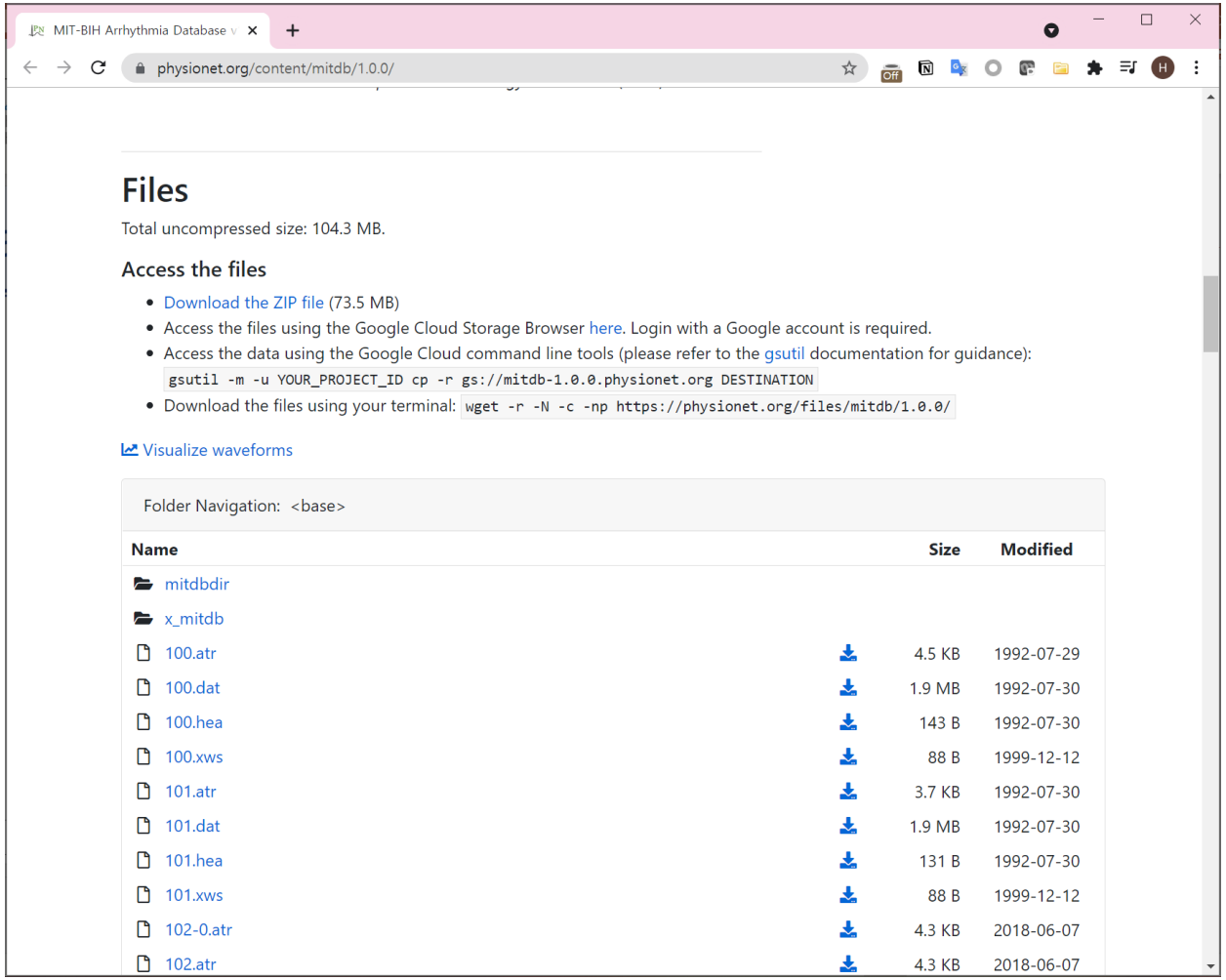


MIT-BIH Arrhythmia Database

SNUH

https://physionet.org/content/mitdb/1.0.0/





Hands-on: Arrhythmia detection