Computer Vision (1주차)

Young-Gon Kim
DLI Instructor

DEEP LEARNING INSTITUTE

DLI Mission

Helping people solve challenging problems using AI and deep learning.

- Developers, data scientists and engineers
- Self-driving cars, healthcare and robotics
- Training, optimizing, and deploying deep neural networks

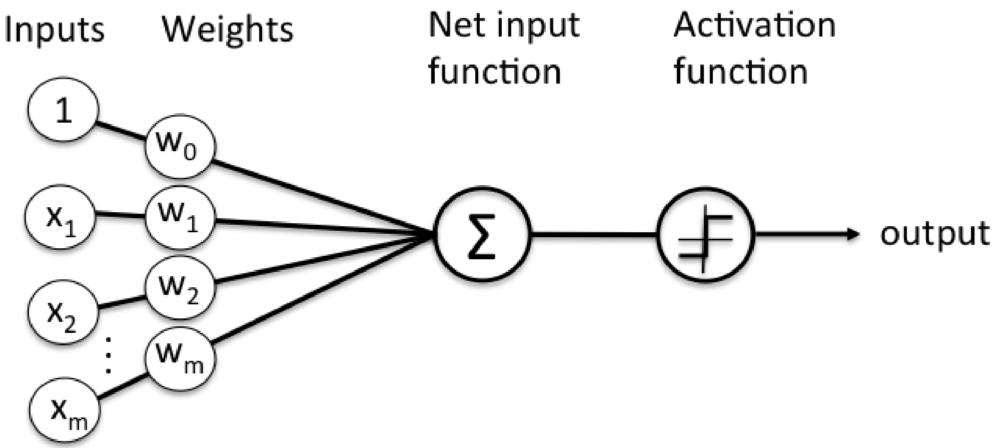
TOPICS

- History
- Application
- Image Processing
- Multi-Layer Perceptron (MLP)
- Convolutional Neural Networks (CNNs)
- Deep Learning Framework (Caffe)

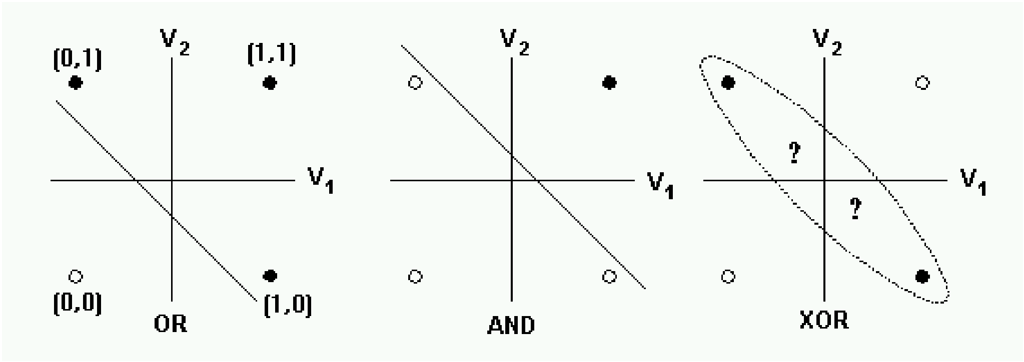
HISTORY

HISTORY

- Artificial Neural Networks (ANNs) & Perceptron (1943~1986년)
  - McCulloch와 Pitts가 최초의 인공신경망 제시
  - Frank Rosenblatt가 최초의 **Perceptron** 제시
  - 하지만, Minsky와 Papert가 **Perceptron으로 XOR문제를 해결할 수 없음을 제기**



최초의 Perceptron

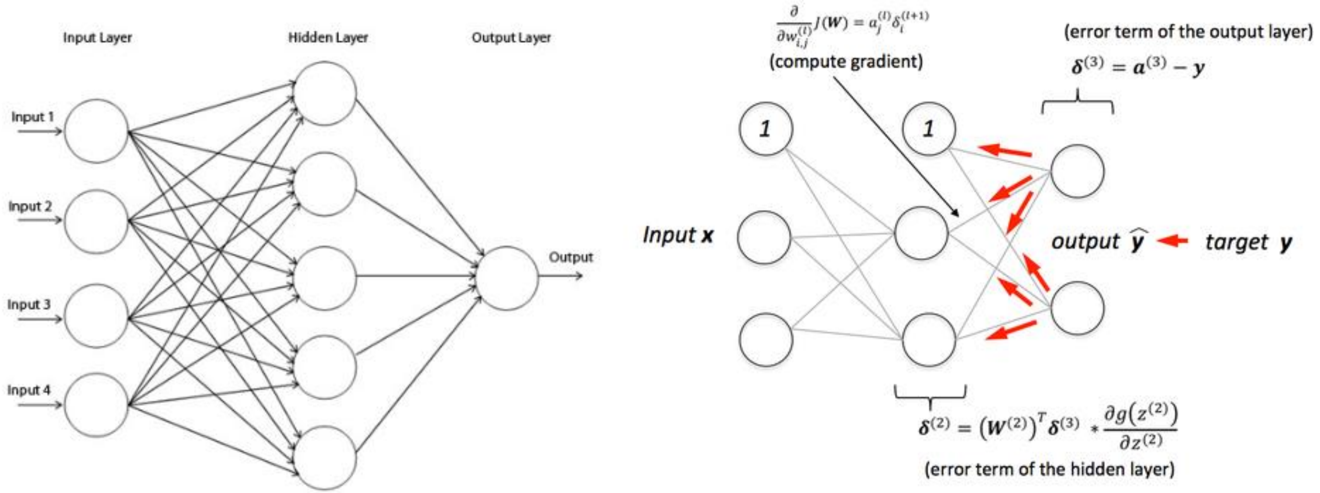


Perceptron은 XOR문제를 해결할 수 없음

http://solarisailab.com/archives/1206

HISTORY

- Multi-Layer Perceptron (MLP) & Backpropagation Algorithm (1986~2006년)
  - McClelland, James L., David E. Rumelhart, and Geoffrey E. Hinton
  - **Multi-Layer Perceptron**으로 XOR문제 해결
  - **Backpropagation Algorithm**으로 적절한 weight와 bias를 학습

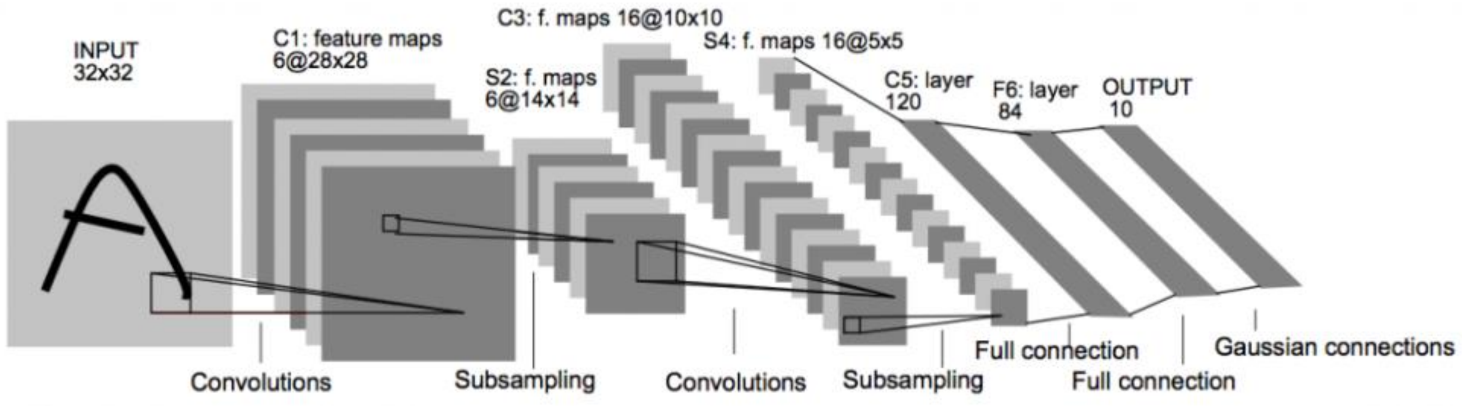


MLP 구조와 Backpropagation Algorithm

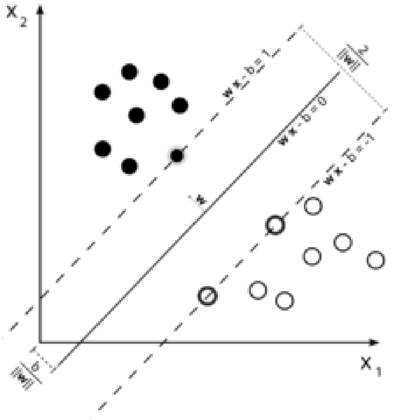
http://solarisailab.com/archives/1206

HISTORY

- Multi-Layer Perceptron (MLP) & Backpropagation Algorithm (1986~2006년)
  - Yann Lecun이 Convolutional Neural Networks (CNNs)의 시초가 되는 Neural Networks 구조인 **LeNet-5** 제안
  - **Vanishing Gradient Problem**
  - SVM (Support Vector Machine) 등의 성능 좋은 Machine Learning 기법의 등장



LeNet-5 (1998)



Support Vector Machine

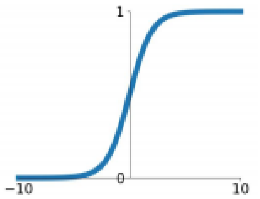
http://solarisailab.com/archives/1206

HISTORY

- ReLU (Rectified Linear Unit)
  - Hinton은 vanishing gradient의 원인이 activation function으로 사용한 sigmoid로 생각
    - Sigmoid는 미분을 거듭할 수록 0에 가까워짐
  - 새로운 activation인 **ReLU** 제안
  - Vanishing gradient 해결!

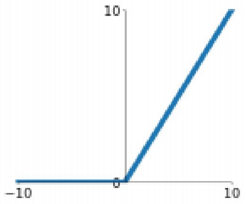
Sigmoid

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



ReLU

$\max(0, x)$



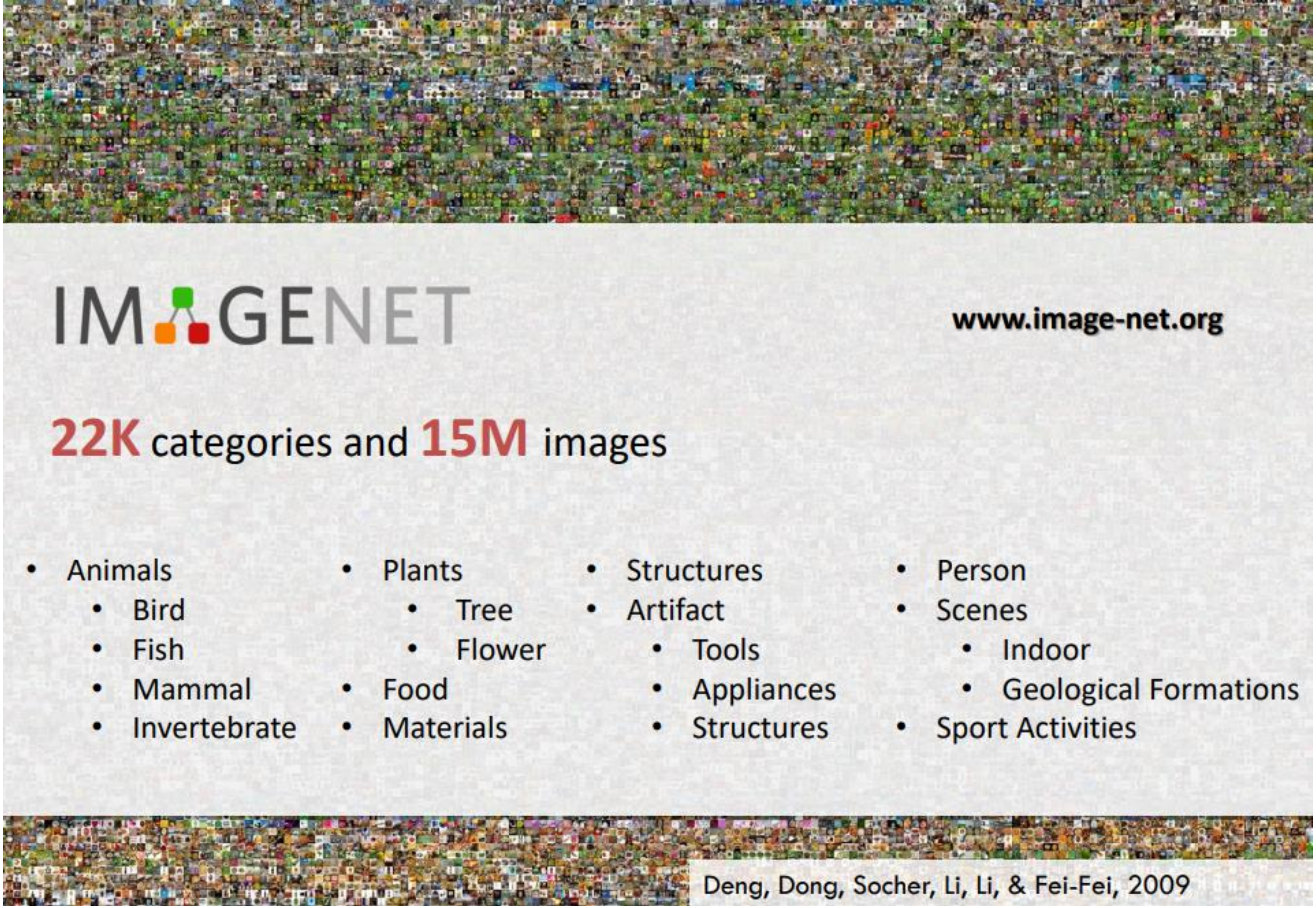
http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture03.pdf

HISTORY

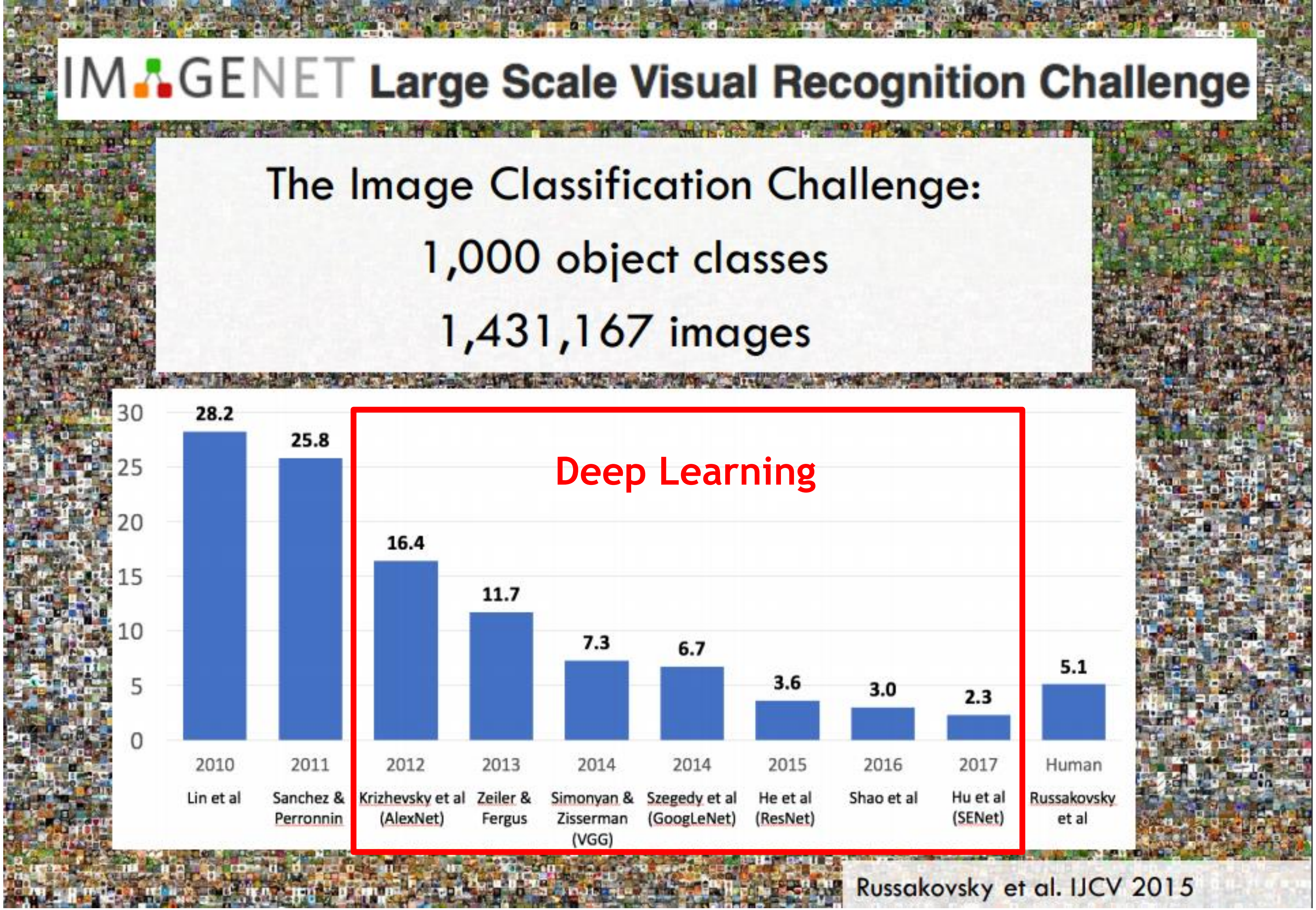
- 이 후 인공지능의 부흥
  - **GPU (Graphics Processing Unit)**의 사용으로 학습시간 개선
    - 엄청난 코어 수로 간단한 "병렬연산" 가능
  - **Big Data**
    - 적은 데이터는 overfitting만 발생

      → 엄청난 수의 데이터를 다룰 수 있는 기술로 딥러닝 발전

https://www.slideshare.net/HyungsooRyoo/ss-80388094



http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture01.pdf

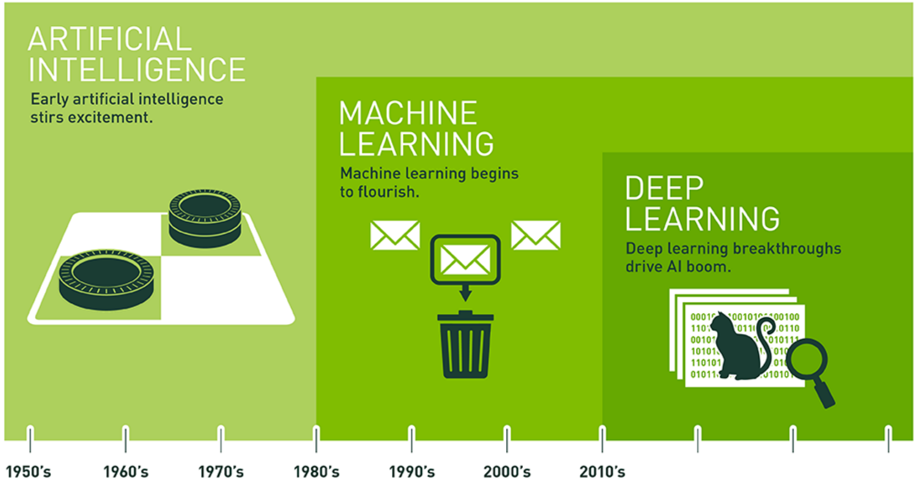


http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture01.pdf

APPLICATION

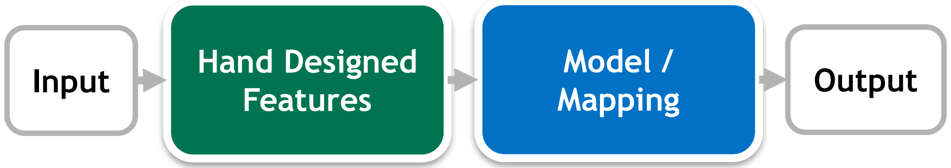
APPLICATION

- Machine Learning vs. Deep Learning



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Classic Machine Learning [1990 : now]



Deep/End-to-End Learning [2012 : now]

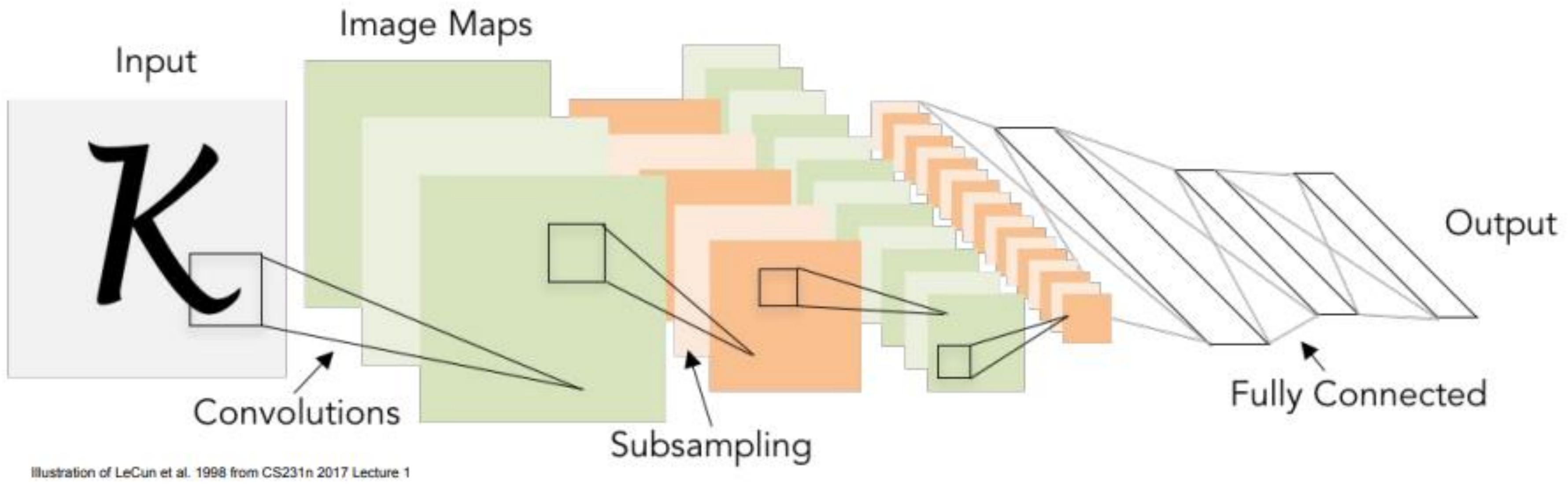
https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

APPLICATION

- Machine Learning
  - **Supervised Learning**
    - KNN (K-Nearest Neighbors)
    - Linear & Logistic Regression
    - Decision Tree, Random Forest
    - Support Vector Machine
    - Neural Network
  - **Unsupervised Learning**
    - Clustering
      - K-Means, HCA, EM
    - Dimensionality Reduction
      - PCA, LLE, t-SNE
  - **Reinforcement Learning**

APPLICATION

- Deep Learning
  - **Convolutional Neural Networks (CNNs)**

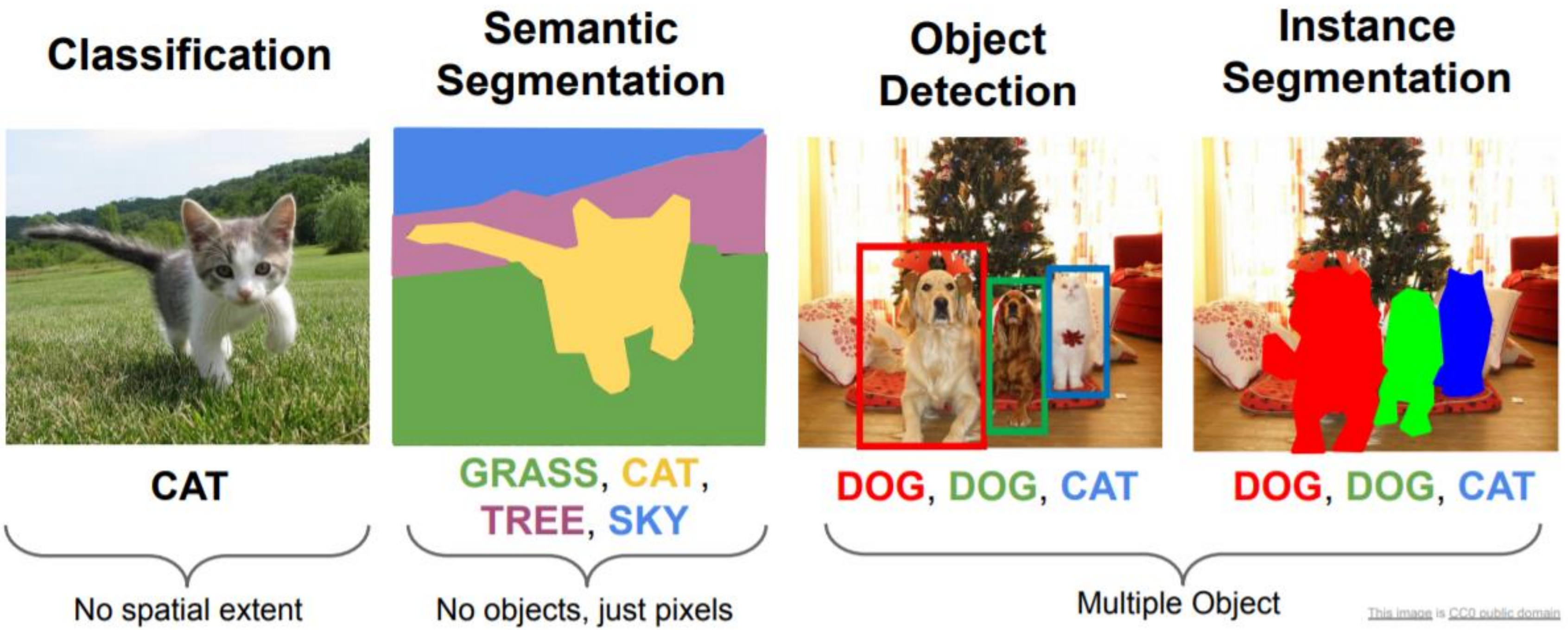


CNN 구조

http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture05.pdf

APPLICATION

- Deep Learning
  - **Convolutional Neural Networks (CNNs)**

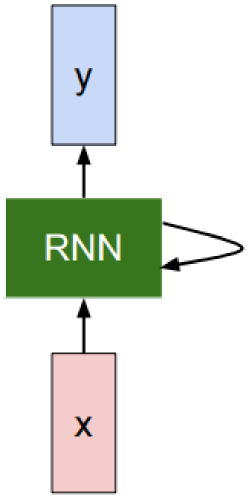


Applications using CNNs

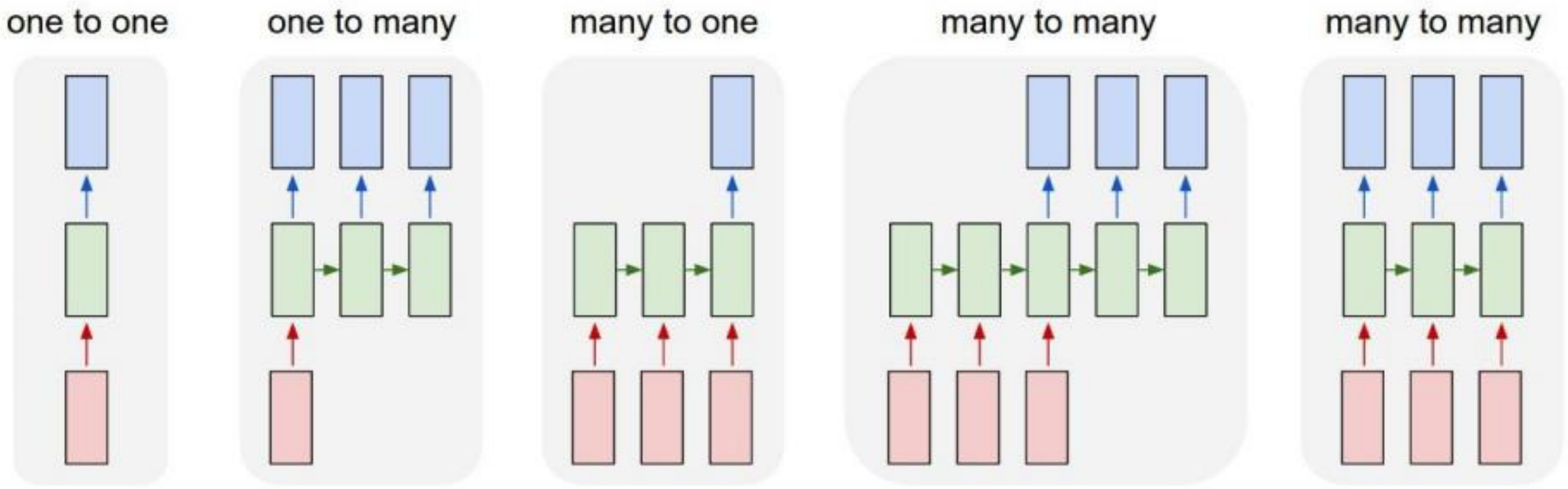
http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture12.pdf

APPLICATION

- Deep Learning
  - **Recurrent Neural Networks (RNNs)**



RNN 구조



RNN Process Sequence

http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture10.pdf

APPLICATION

- Deep Learning
  - **Recurrent Neural Networks (RNNs)**

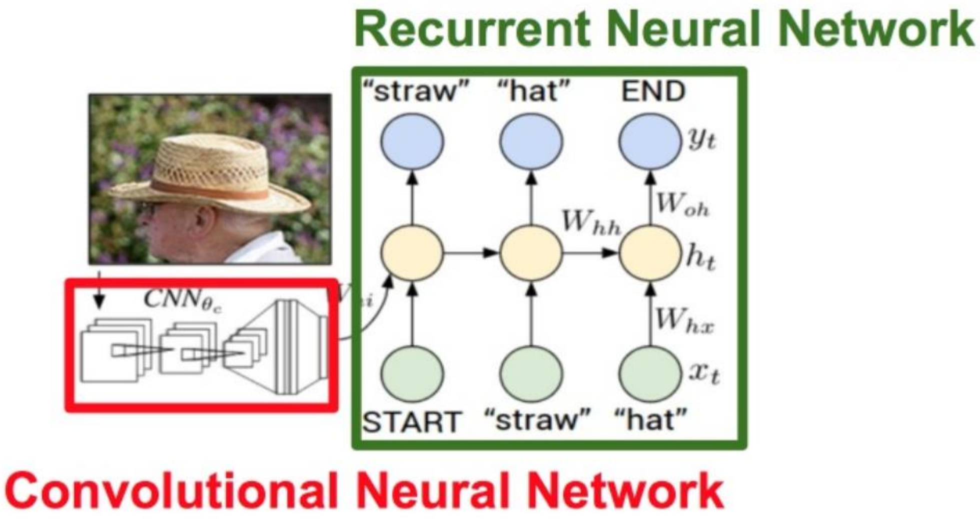
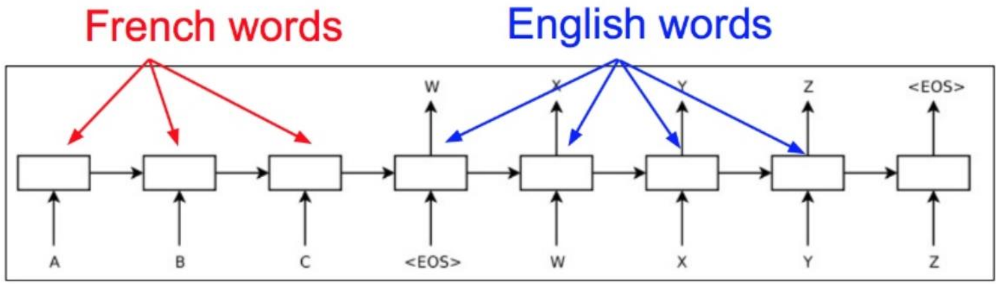


Image Captioning

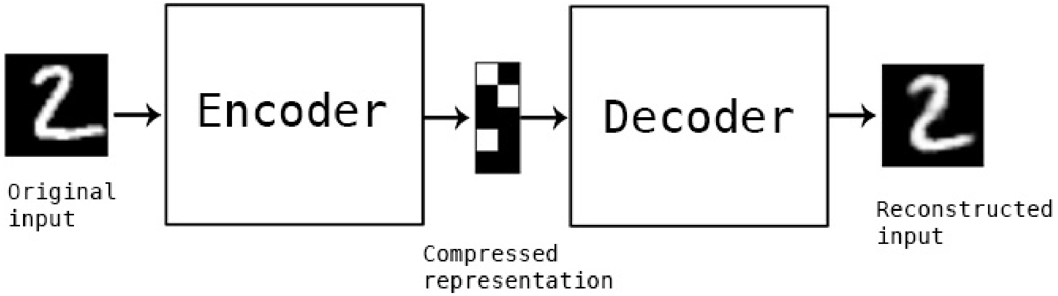


Machine Translation Model

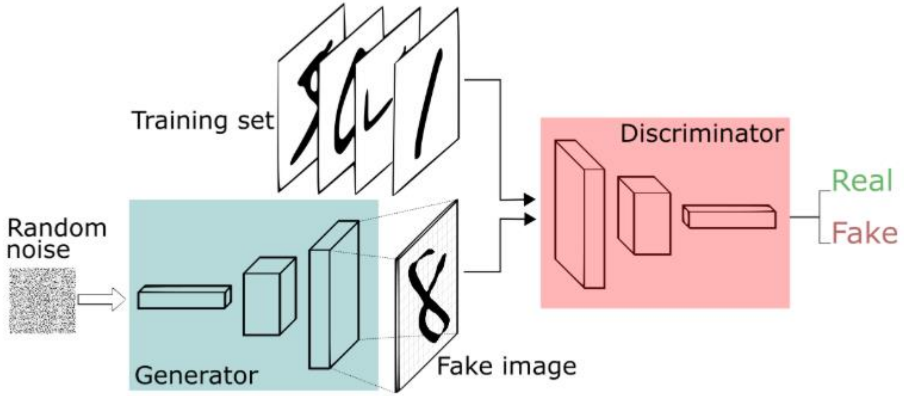
https://www.slideshare.net/samchoi7/rnnerica

APPLICATION

- Deep Learning
  - **Generative Adversarial Networks (GAN)**



Auto Encoder (AE)

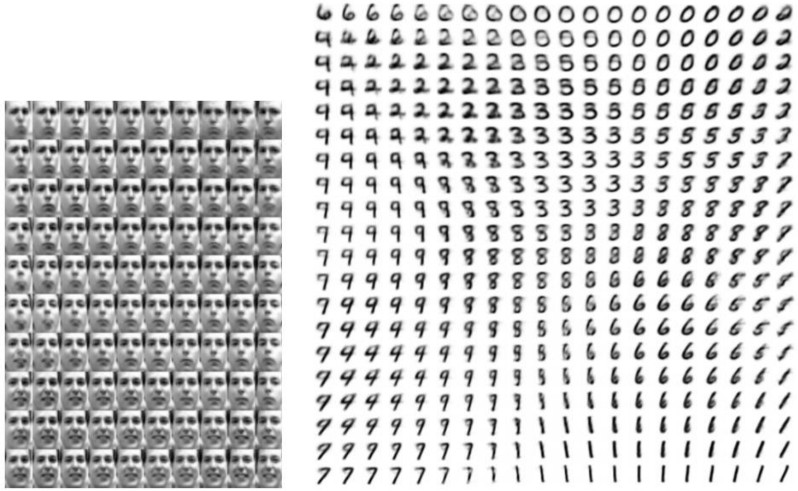


Generative Adversarial Networks (GANs)

http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture11.pdf

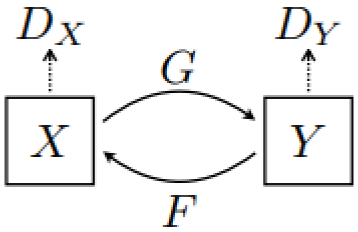
APPLICATION

- Deep Learning
  - **Generative Adversarial Networks (GAN)**

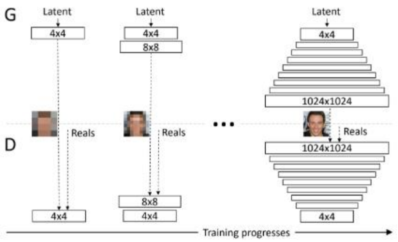


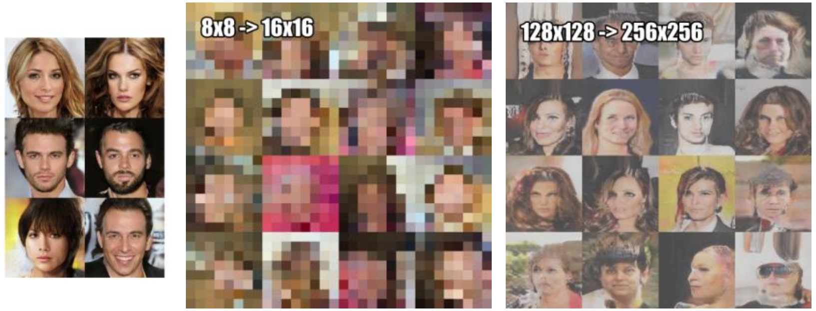
(a) Learned Frey Face manifold (b) Learned MNIST manifold

Latent vector space 변화에 따른 VAE 결과



CycleGAN (J.Y. Zhu et al., ICCV 2017.)



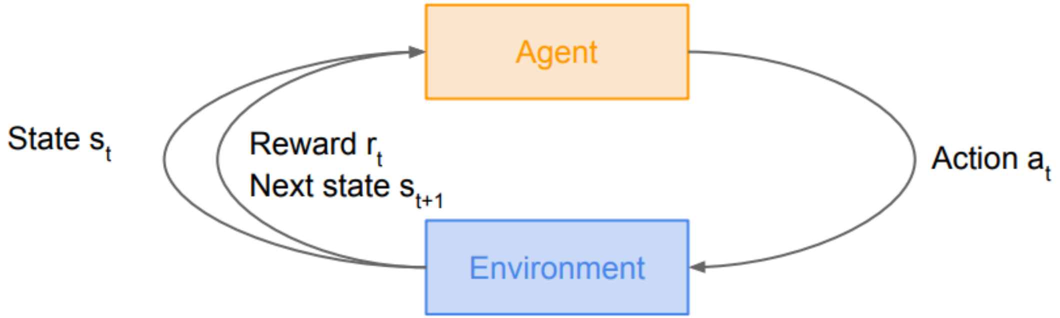


PGGAN (T. Karras et al., ICLR 2018.)

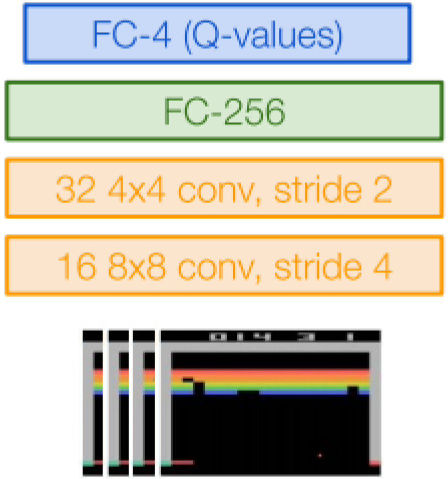
https://arxiv.org/abs/1312.6114
https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix
https://arxiv.org/abs/1710.10196
https://github.com/nashory/pggan-pytorch

APPLICATION

- Deep Learning
  - **Deep Reinforcement Learning**



Reinforcement Learning



Current state s_t: 84x84x4 stack of last 4 frames
(after RGB->grayscale conversion, downsampling, and cropping)

Q-network Architecture

http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture14.pdf

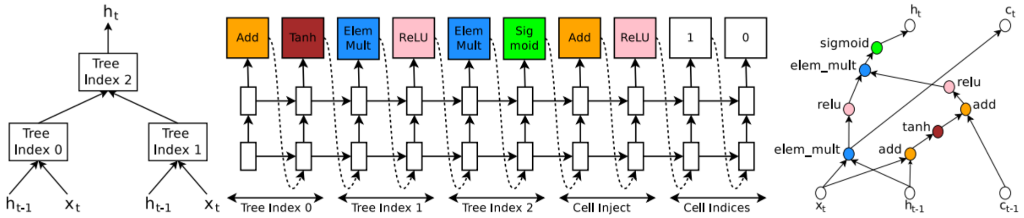
APPLICATION

- Deep Learning
  - **Deep Reinforcement Learning**

Boston Dynamics

AlphaGo

AlphaGo (Google Deepmind)



Neural Architecture Search (NAS)

https://youtu.be/_sBBaNYex3E
https://en.wikipedia.org/wiki/AlphaGo
https://arxiv.org/pdf/1611.01578.pdf

IMAGE PROCESSING

IMAGE PROCESSING

8 bit ?

RGB ?

Image format ?

IMAGE PROCESSING

8 bit

0	0	0	0	0	0	0	0

$$0\ (=0\text{x}2^7+0\text{x}2^6+0\text{x}2^5+0\text{x}2^4+0\text{x}2^3+0\text{x}2^2+0\text{x}2^1+0\text{x}2^0)$$

0	0	0	0	0	0	0	1

$$1\ (=0\text{x}2^7+0\text{x}2^6+0\text{x}2^5+0\text{x}2^4+0\text{x}2^3+0\text{x}2^2+0\text{x}2^1+1\text{x}2^0)$$

~

1	1	1	1	1	1	1	1

$$255\ (=1\text{x}2^7+1\text{x}2^6+1\text{x}2^5+1\text{x}2^4+1\text{x}2^3+1\text{x}2^2+1\text{x}2^1+1\text{x}2^0)$$

IMAGE PROCESSING

RGB

24bit (RGB) = 8bit (R) + 8bit (G) + 8bit (B)

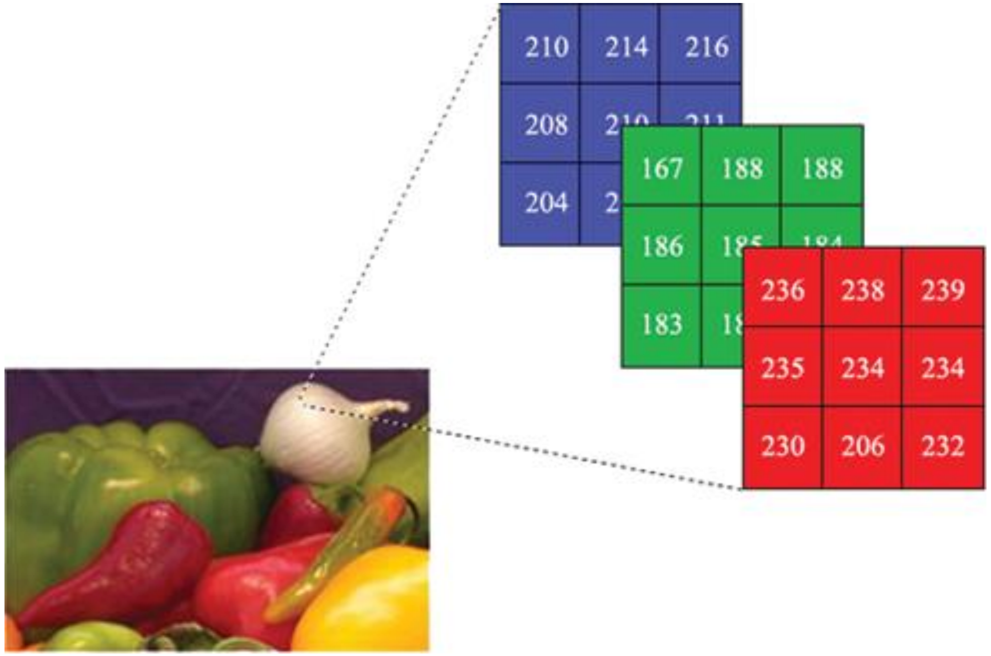


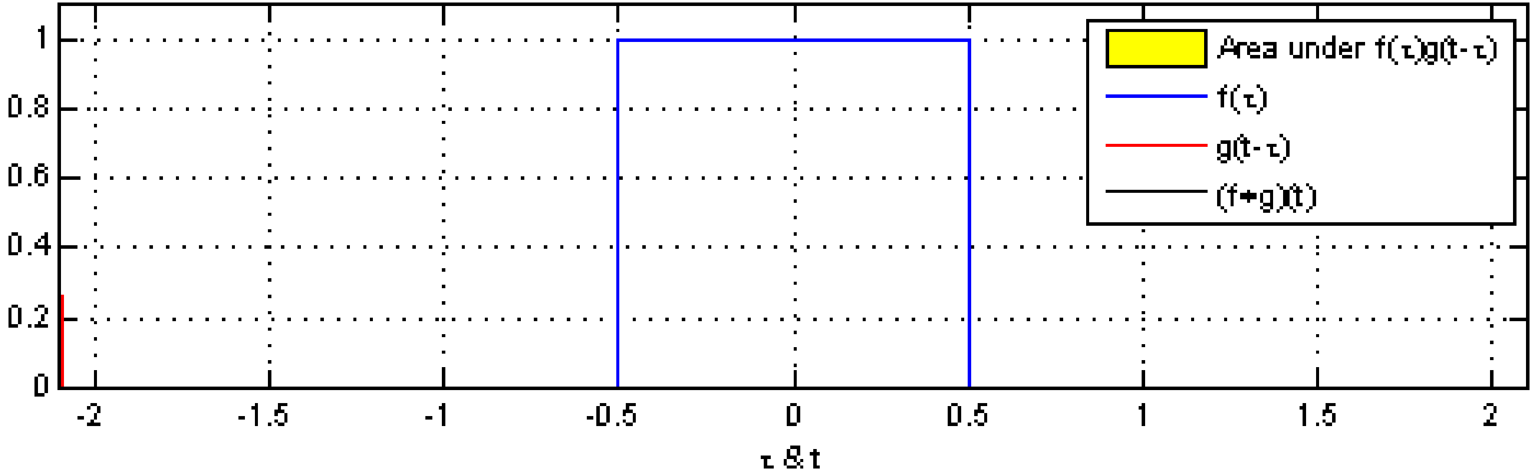
IMAGE PROCESSING

Image format

Type			Compression
JPEG			Loss
BMP			Lossless
PNG			Lossless
DICOM			Loss

IMAGE PROCESSING

- Convolution (1D-dimension)
  - $f(t) * g(t) \triangleq \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$



https://en.wikipedia.org/wiki/Convolution

IMAGE PROCESSING

- Convolution (2D-dimension)
  - $\boldsymbol{f(t) * g(t) \triangleq \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau}$

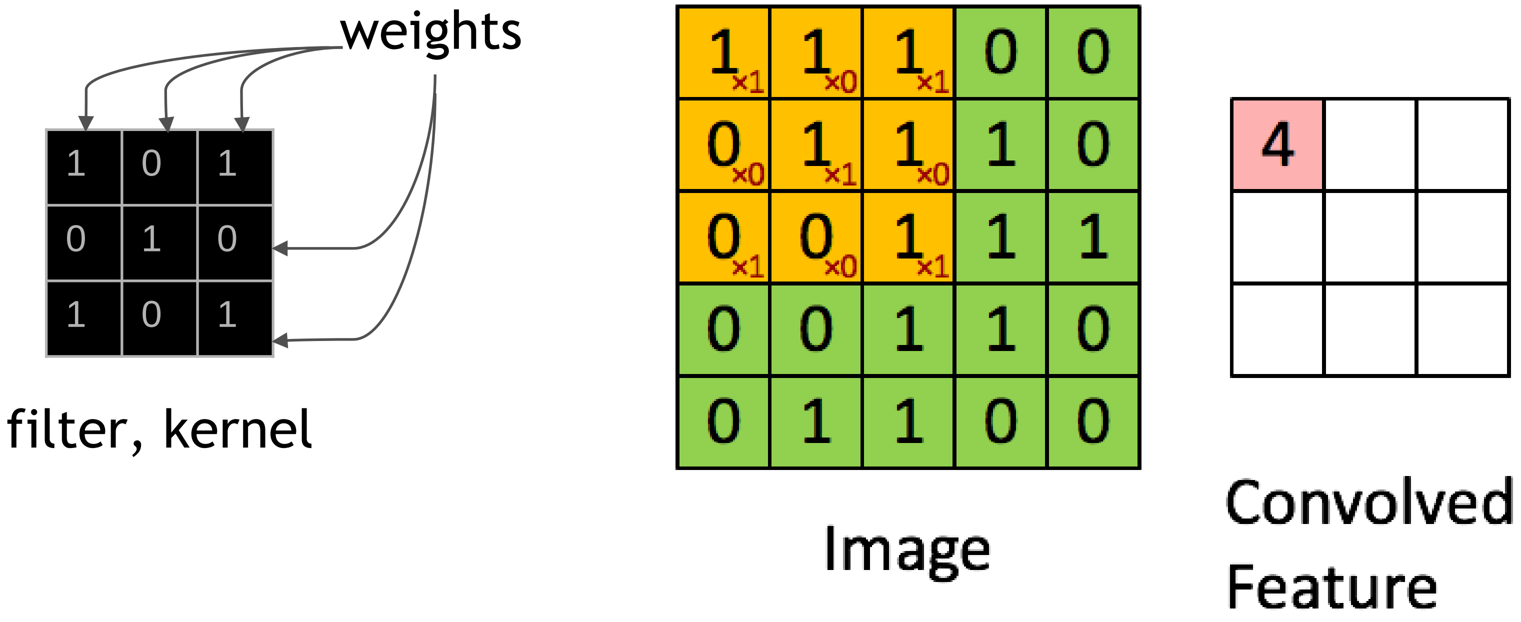
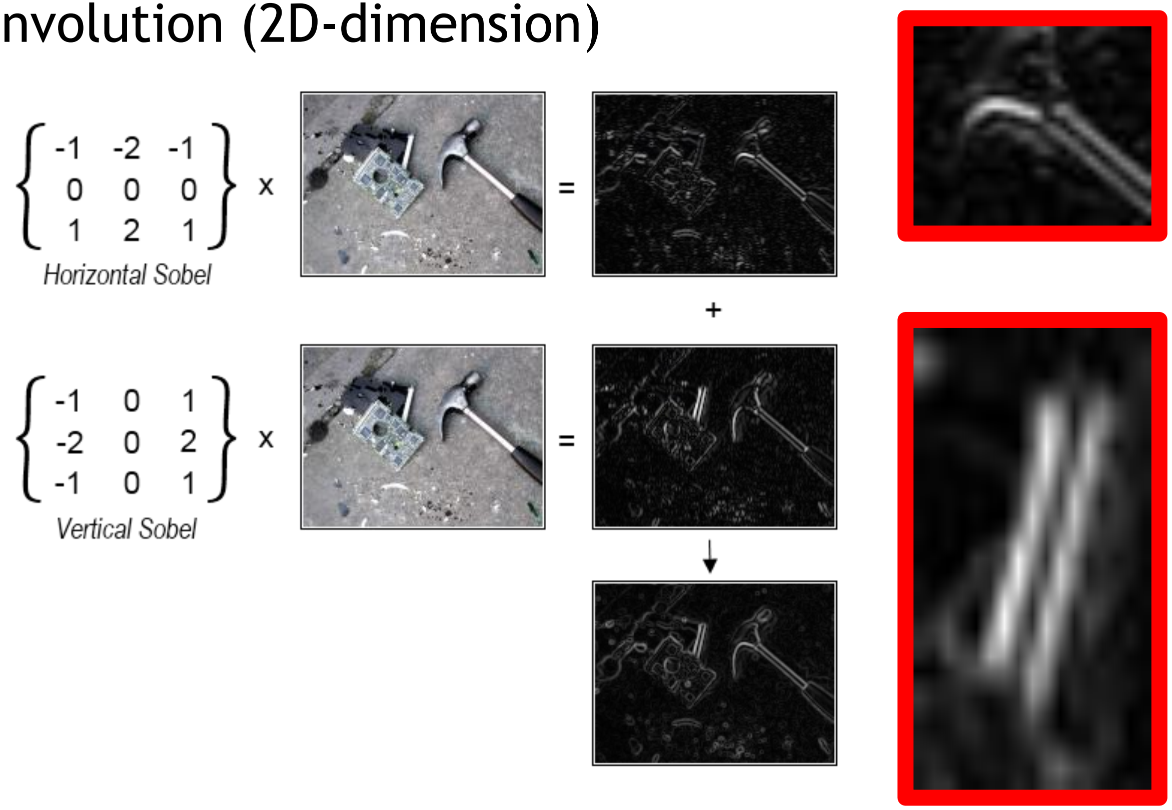


IMAGE PROCESSING

- Convolution (2D-dimension)



Edge Detection

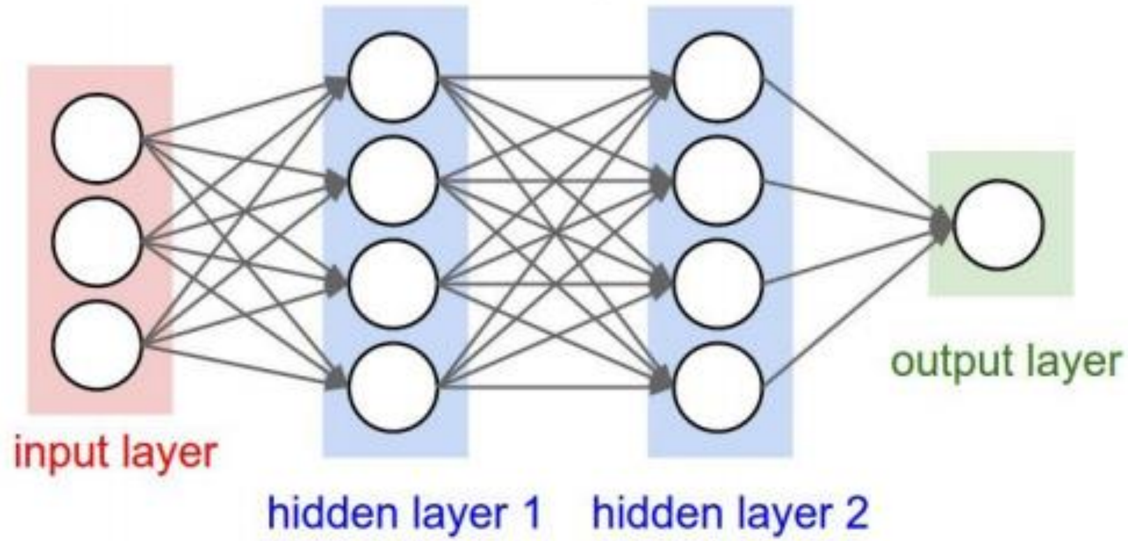
Gaussian Blur

https://en.wikipedia.org/wiki/Gaussian_blur

MULTI-LAYER PERCEPTRON (MLP)

Multi-Layer Perceptron (MLP)

- **Architecture**
  - 기존 perceptron은 hidden layer 없이 그대로 output 도출 → 선형 연산
  - MLP는 hidden layer 존재 → **비선형 연산**



Activation Function

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

or 3-layer Neural Network

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

$$x \in \mathbb{R}^D, W_1 \in \mathbb{R}^{H_1 \times D}, W_2 \in \mathbb{R}^{H_2 \times H_1}, W_3 \in \mathbb{R}^{C \times H_2}$$

Multi-Layer Perceptron

http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture04.pdf

Multi-Layer Perceptron (MLP)

- **Backpropagation**

Backpropagation: a simple example

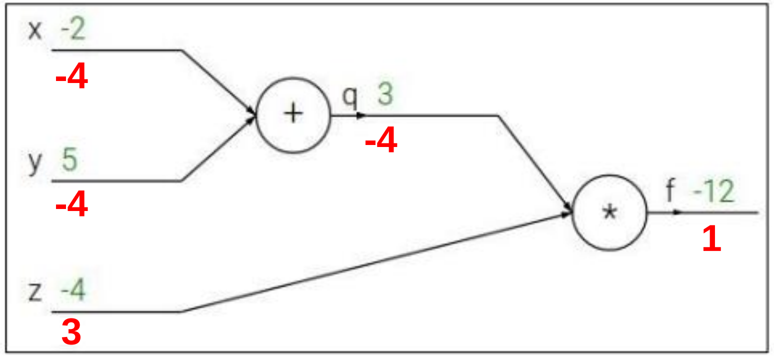
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$\boxed{q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1}$$

$$\boxed{f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Chain rule:

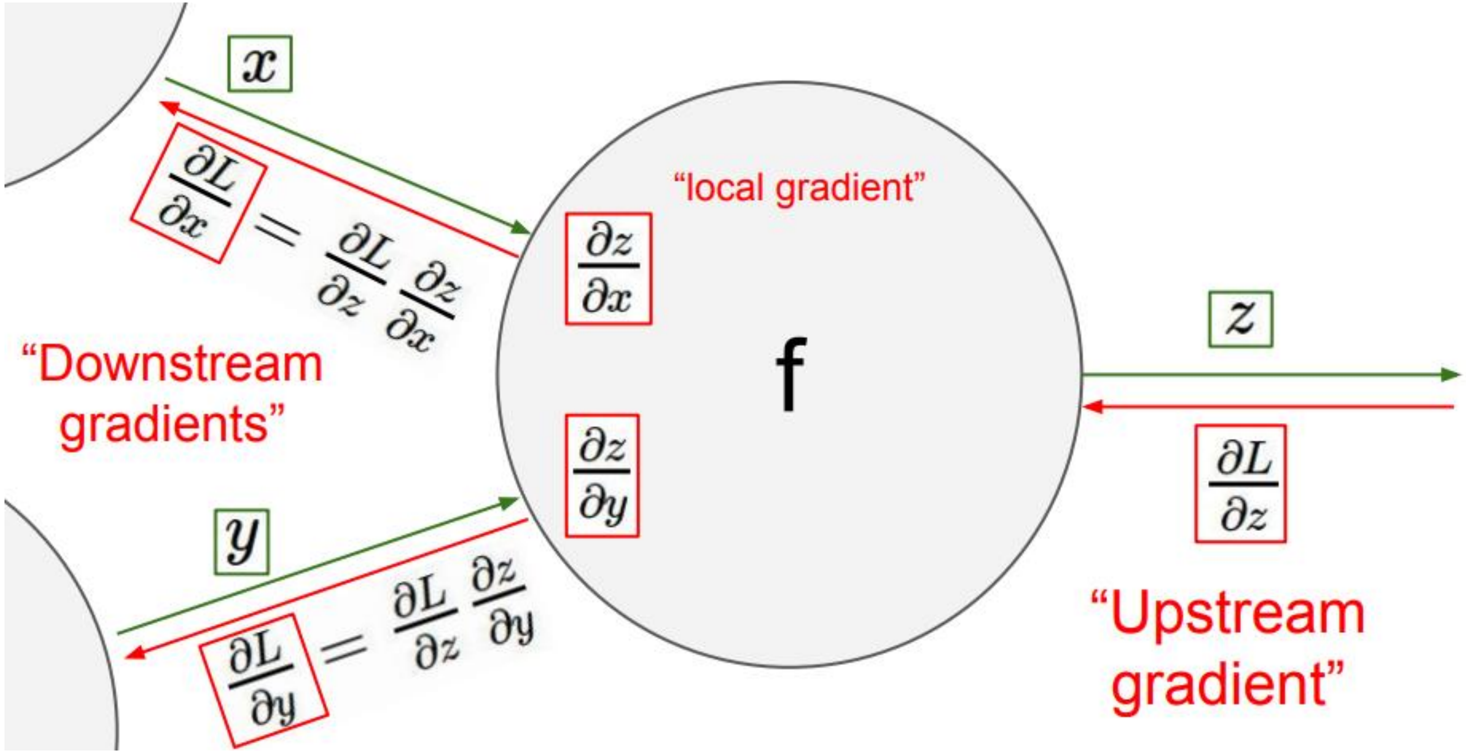
$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}}$$

Upstream gradient   Local gradient

http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture04.pdf

Multi-Layer Perceptron (MLP)

- **Backpropagation**



http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture04.pdf

CONVOLUTIONAL NEURAL NETWORKS (CNNs)

CONVOLUTIONAL NEURAL NETWORKS (CNNs)

- **Terminology**
  - **Hyper parameter**
    - Loss function
      - MSE / MAE / Cross-entropy / Soft Dice Loss / …
    - Optimizer
      - GD / SGD / Momentum / Adagrad / Adam / …
    - Measure
      - Accuracy / Sensitivity / Specificity / AUC / Dice coefficient / IoU / …
    - Learning rate
    - Epoch / Step

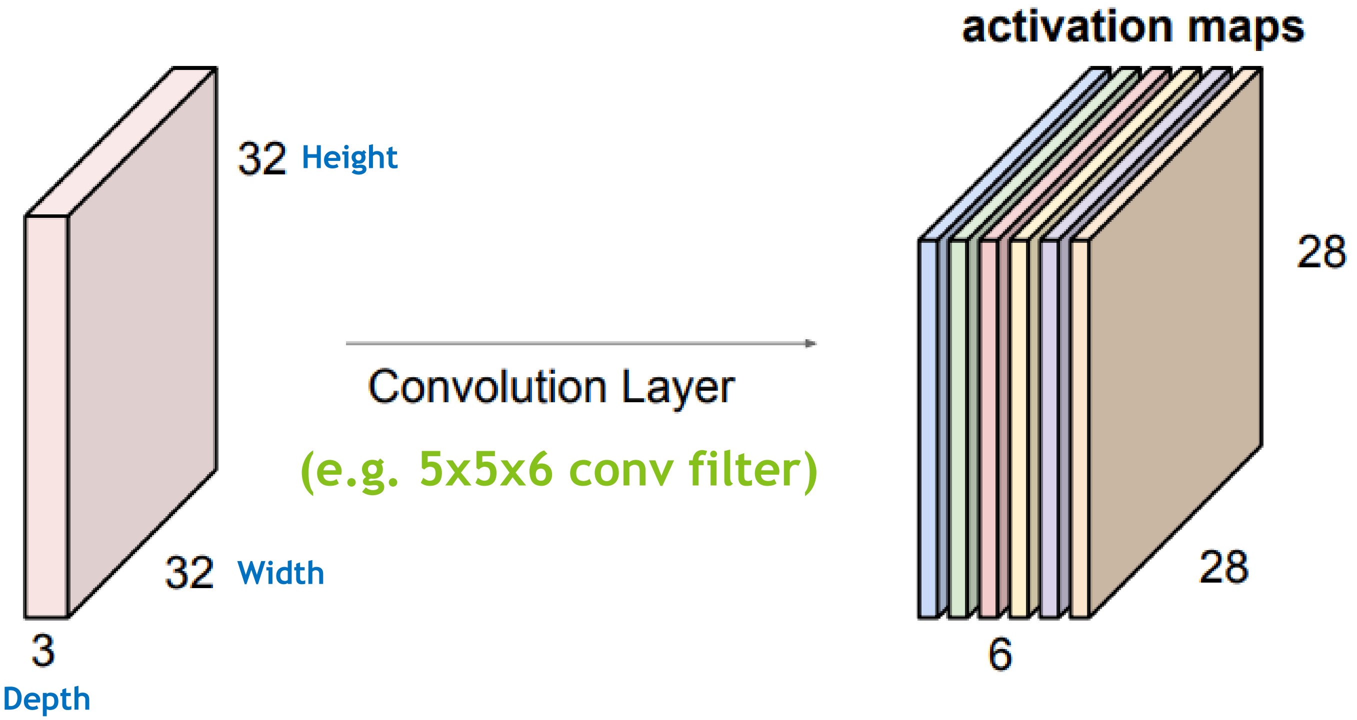
CONVOLUTIONAL NEURAL NETWORKS (CNNs)

- **Terminology**
  - **Tricks to improve the model's performance (T. He et al., CVPR 2018.)**
    - Early stopping
    - Learning rate
      - Decay (Linear, Cosine, …)
      - Warm up
    - Model Tweaks
    - Augmentation
    - Label smoothing
    - Knowledge Distillation
    - Mix-up Training
    - Transfer Learning
    - Large-batch training
    - …

https://arxiv.org/pdf/1812.01187.pdf

CONVOLUTIONAL NEURAL NETWORKS (CNNs)

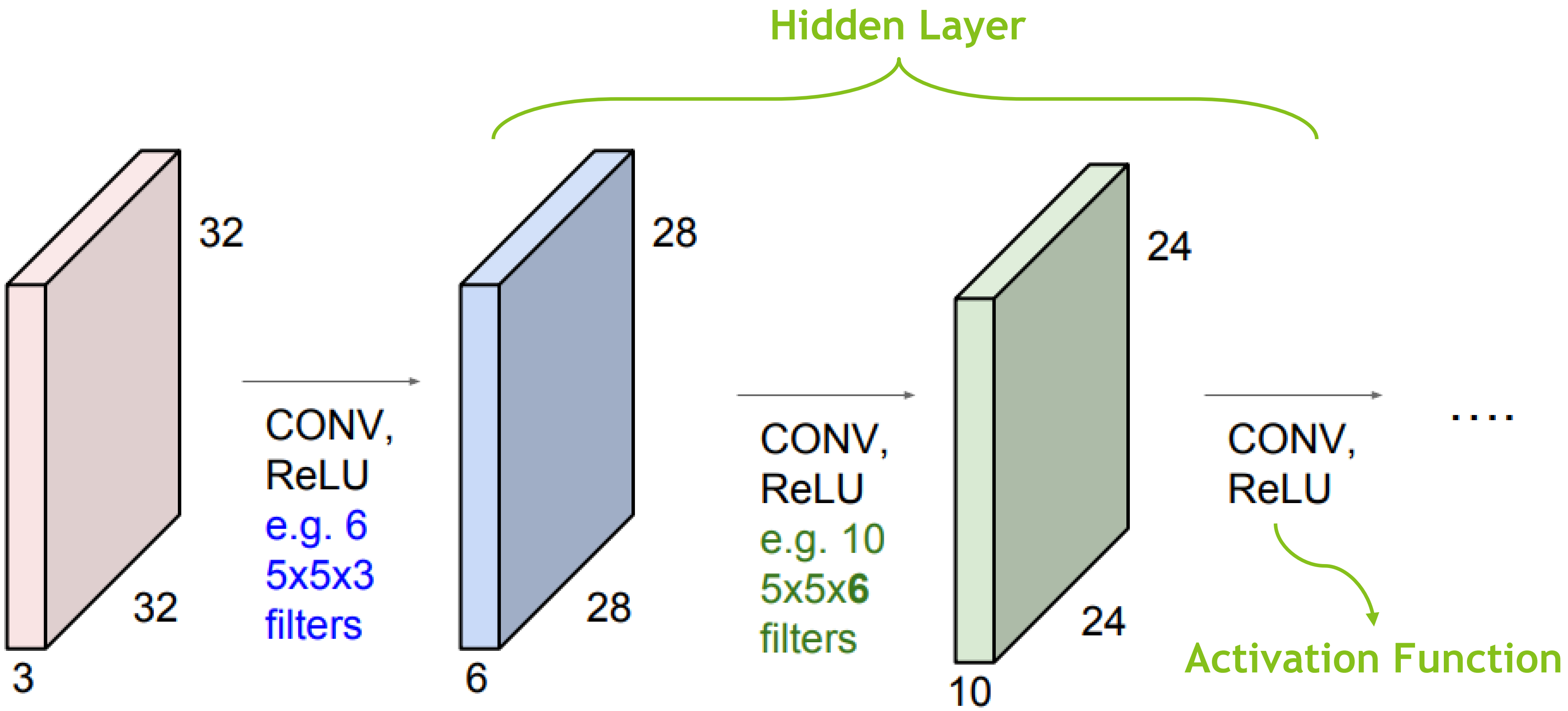
- Architecture



http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture05.pdf

CONVOLUTIONAL NEURAL NETWORKS (CNNs)

- **Architecture**



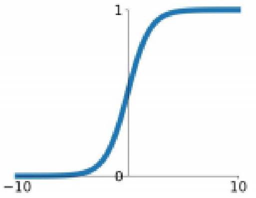
http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture05.pdf

CONVOLUTIONAL NEURAL NETWORKS (CNNs)

- **Activation Function**

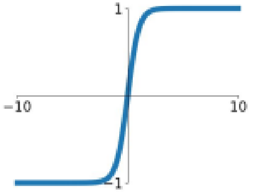
Sigmoid

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



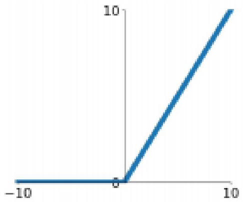
tanh

$\tanh(x)$



ReLU

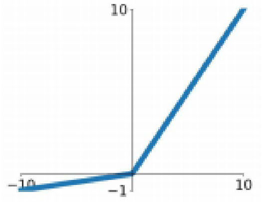
$\max(0, x)$



ReLU is a good default choice for most problems!

Leaky ReLU

$\max(0.1x, x)$

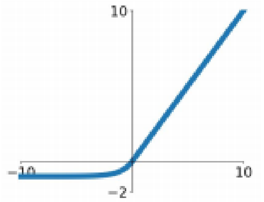


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture03.pdf

CONVOLUTIONAL NEURAL NETWORKS (CNNs)

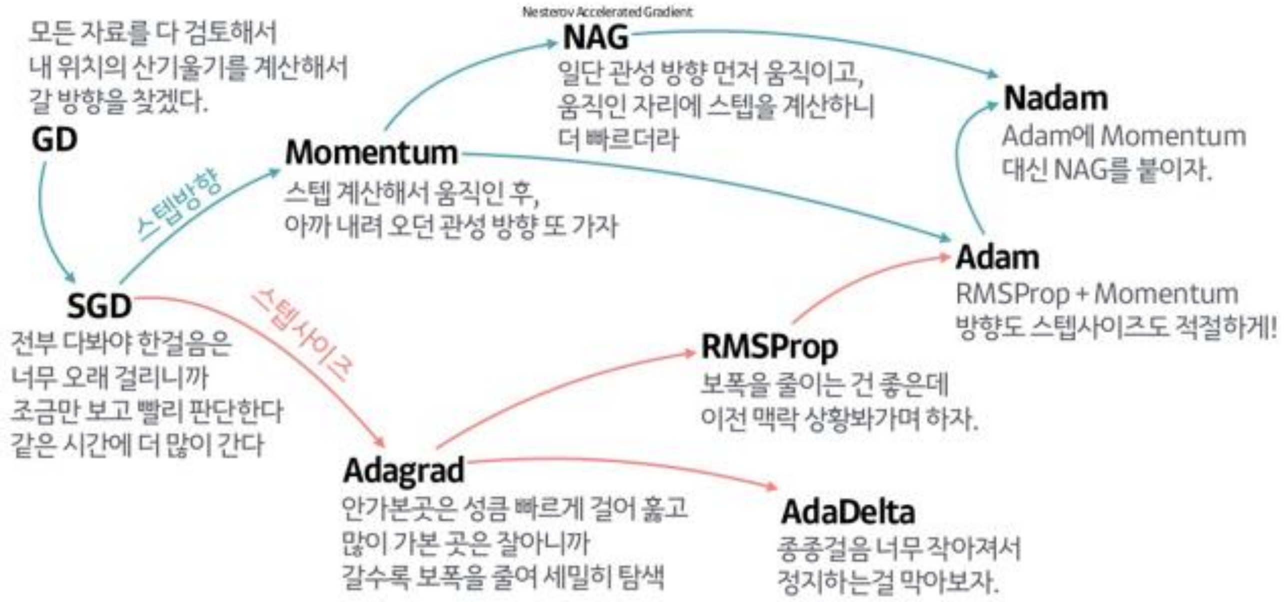
- **Loss Function**

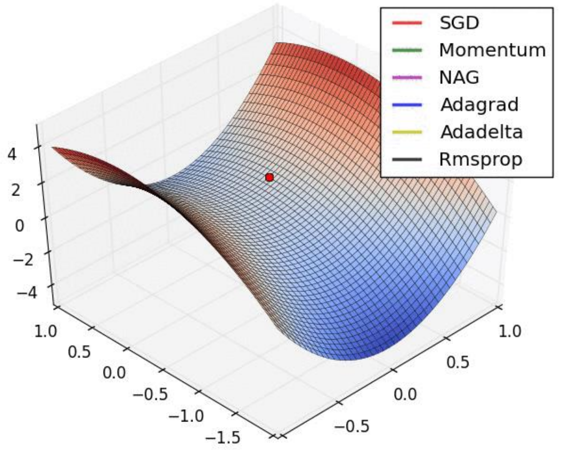
Loss function	Equation
Mean Squared Error (MSE)	$L = \frac{1}{N}\sum_{i}^{N}(x_i - y_i)^2$
Mean Absolute Error (MAE)	$L = \frac{1}{N}\sum_{i}^{N}\lvert x_i - y_i\rvert$
Cross-entropy	$L = -\sum y \log x$
Soft Dice Loss	$L = 1 - \frac{2\,\lvert X \cap Y\rvert}{\lvert X\rvert + \lvert Y\rvert}$

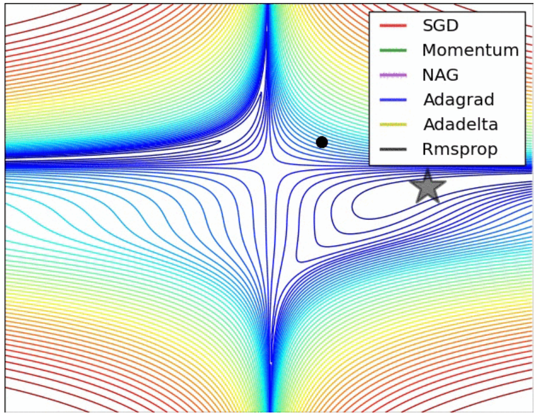
http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture03.pdf

CONVOLUTIONAL NEURAL NETWORKS (CNNs)

- **Optimizer**







https://www.slideshare.net/yongho/ss-79607172/49
http://shuuki4.github.io/deep%20learning/2016/05/20/Gradient-Descent-Algorithm-Overview.html

CONVOLUTIONAL NEURAL NETWORKS (CNNs)

- **Measure**

		True condition	
	Total	Condition Positive	Condition Negative
Predicted condition	Predicted Condition Positive	True Positive	False Positive
	Predicted Condition Negative	False Negative	True Negative

Accuracy

Precision

Sensitivity (Recall)

Specificity

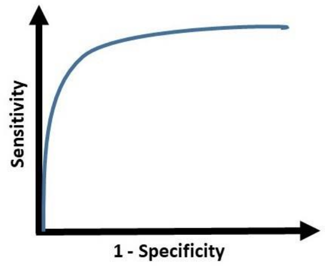
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \qquad \text{Specificity} = \frac{TN}{FP + TN}$$

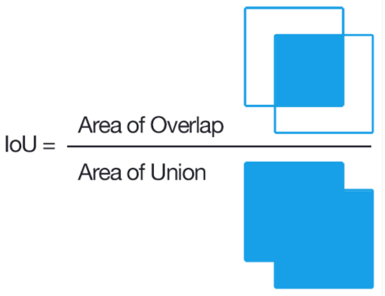
$$\text{Sensitivity (Recall)} = \frac{TP}{TP + FN} \qquad \text{Precision} = \frac{TP}{TP + FP}$$

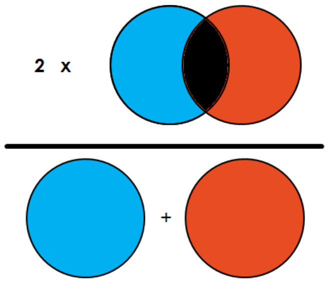
$$\text{AUC (Area Under the Curve)} = \frac{TP}{TP + FP}$$

$$\text{IOU (Intersection Over Union)} = \frac{TP}{TP + FN}$$

$$\text{Dice coefficient} = \frac{2 \times TP}{(TP+FP) + (TP+FN)}$$



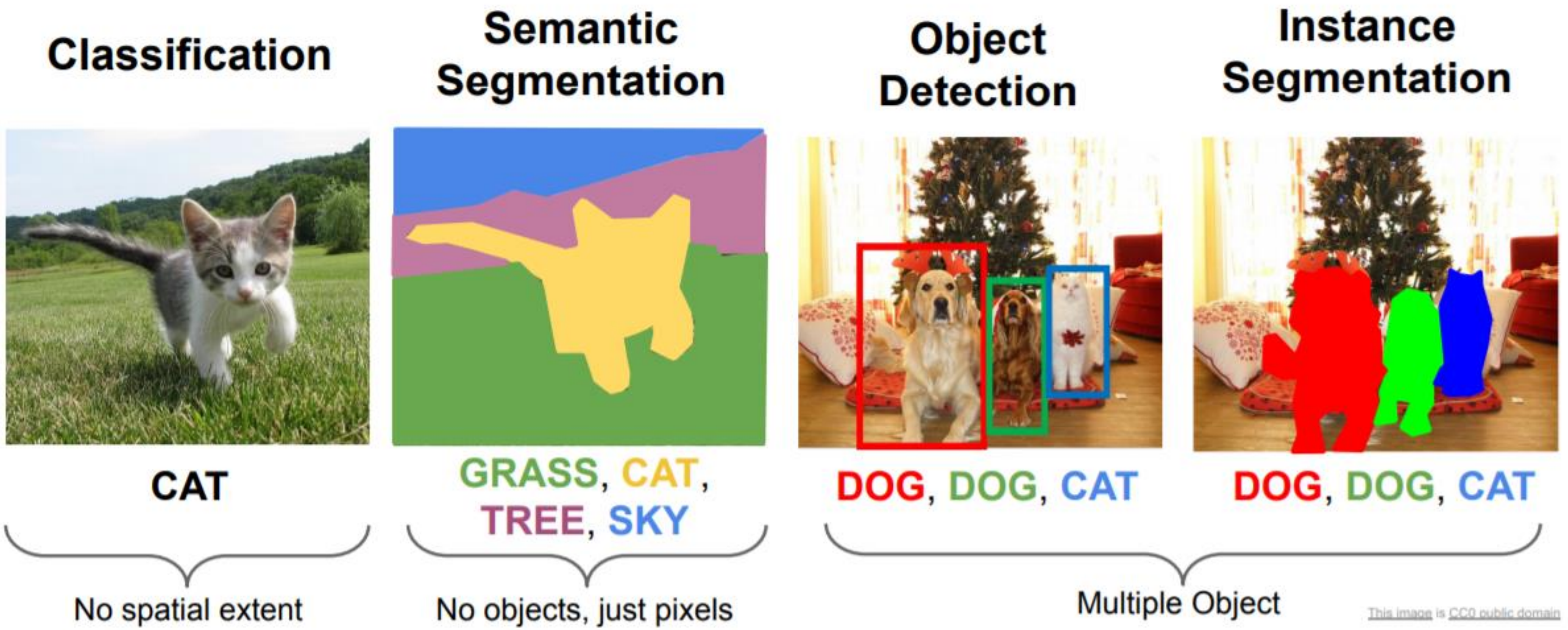




https://en.wikipedia.org/wiki/Sensitivity_and_specificity

CONVOLUTIONAL NEURAL NETWORKS (CNNs)

- **Application**
  - **Computer Vision Tasks**



http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture12.pdf

DEEP LEARNING FRAMEWORK (Caffe)

DEEP LEARNING FRAMEWORK (Caffe)

- Python
  - 문법이 쉽다
  - 오픈 소스
  - 간결하다
  - 개발 속도가 빠르다
- Deep Learning Framework
  - Tensorflow
  - Keras
  - Pytorch
  - Caffe2
  - …

https://wikidocs.net/6

DEEP LEARNING FRAMEWORK (Caffe)

- Caffe

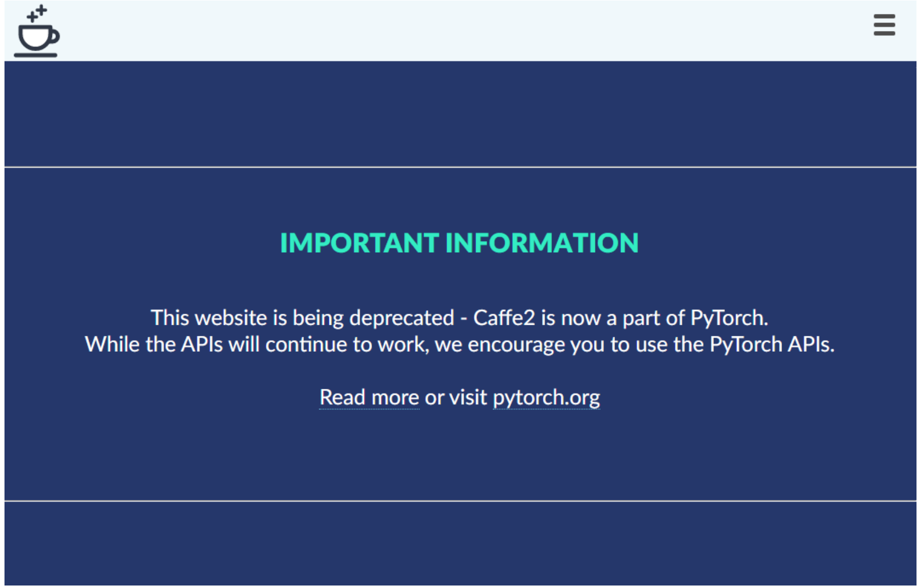
Protobuf model format

- Strongly typed format
- Human readable
- Auto-generates and checks Caffe code
- Developed by Google, currently managed by Facebook
- Used to define network architecture and training parameters
- No coding required!

```
name: "conv1"
type: "Convolution"
bottom: "data"
top: "conv1"
convolution_param {
    num_output: 20
    kernel_size: 3
    stride: 1
    weight_filler {
        type: "xavier"
    }
}
```

DEEP LEARNING FRAMEWORK (Caffe)

- **Caffe2**
  - Made in Facebook
  - 모바일 + 대용량 스케일의 사용 제품을 위한 Framework
  - Protocol buffer를 생성 → C++로 만들어진 backend에 입력



https://tensorflow.blog/tag/caffe2/
https://caffe2.ai/

DEEP LEARNING FRAMEWORK (Caffe)

- **Caffe2**
  - Made in Facebook
  - 모바일 + 대용량 스케일의 사용 제품을 위한 Framework
  - Protocol buffer를 생성 → C++로 만들어진 backend에 입력

```
name: "my first net"
op {
  input: "data"
  input: "fc_w"
  input: "fc_b"
  output: "fc1"
  name: ""
  type: "FC"
}
op {
  input: "fc1"
  output: "pred"
  name: ""
  type: "Sigmoid"
}
op {
  input: "pred"
  input: "label"
  output: "softmax"
  output: "loss"
  name: ""
  type: "SoftmaxWithLoss"
}
external_input: "data"
external_input: "fc_w"
external_input: "fc_b"
external_input: "label"
```

Reference

- http://solarisailab.com/archives/1206
    - https://pdfs.semanticscholar.org/5272/8a99829792c3272043842455f3a110e841b1.pdf
    - http://sebastianraschka.com/Articles/2015_singlelayer_neurons.html
    - http://ecee.colorado.edu/~ecen4831/lectures/NNet3.html
    - https://github.com/cazala/synaptic
    - https://sebastianraschka.com/faq/docs/visual-backpropagation.html
    - http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf
- http://cs231n.stanford.edu/
- https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/
- https://pathmind.com/kr/wiki/generative-adversarial-network-gan
- https://en.wikipedia.org/wiki/Convolution
- https://en.wikipedia.org/wiki/Gaussian_blur
- https://www.slideshare.net/samchoi7/rnnerica
- VAE (Variational Auto-Encoder)
    - Paper : https://arxiv.org/abs/1312.6114
    - Code : https://github.com/pytorch/examples/tree/master/vae
- CycleGAN
    - Paper : https://arxiv.org/abs/1703.10593
    - Code : https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

DEEP
LEARNING
INSTITUTE

www.nvidia.com/dli