ML_5

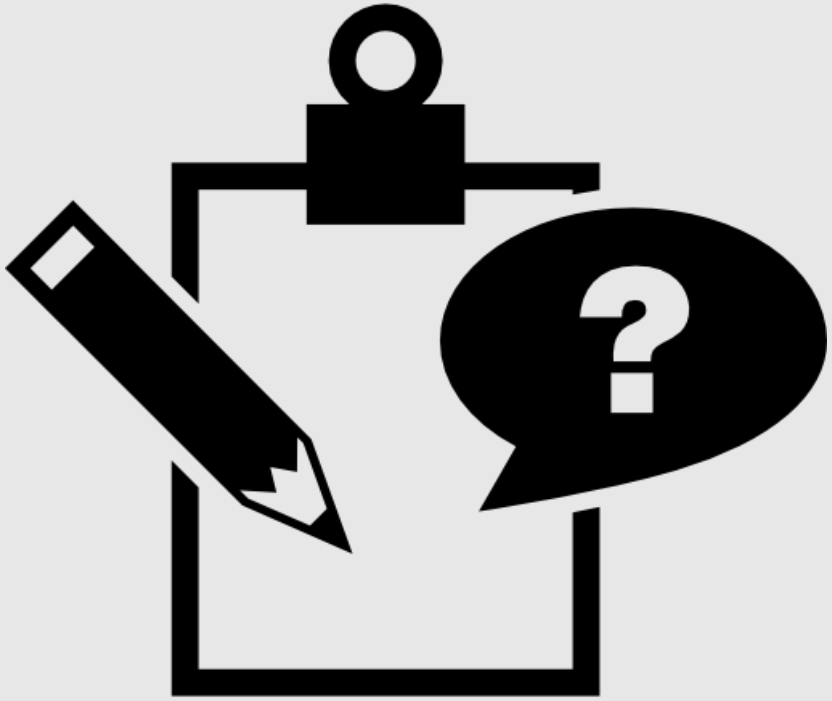
Team BMS

INDEX

ML Study
5 Week

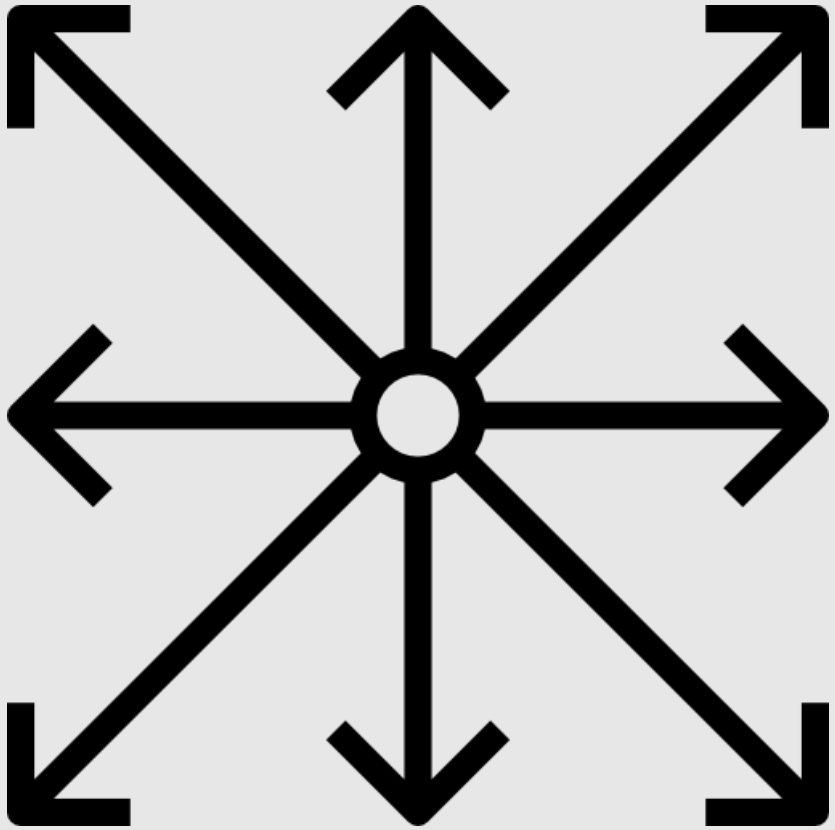
Ice Breaking

Ice Breaking



Dimensionality Reduction

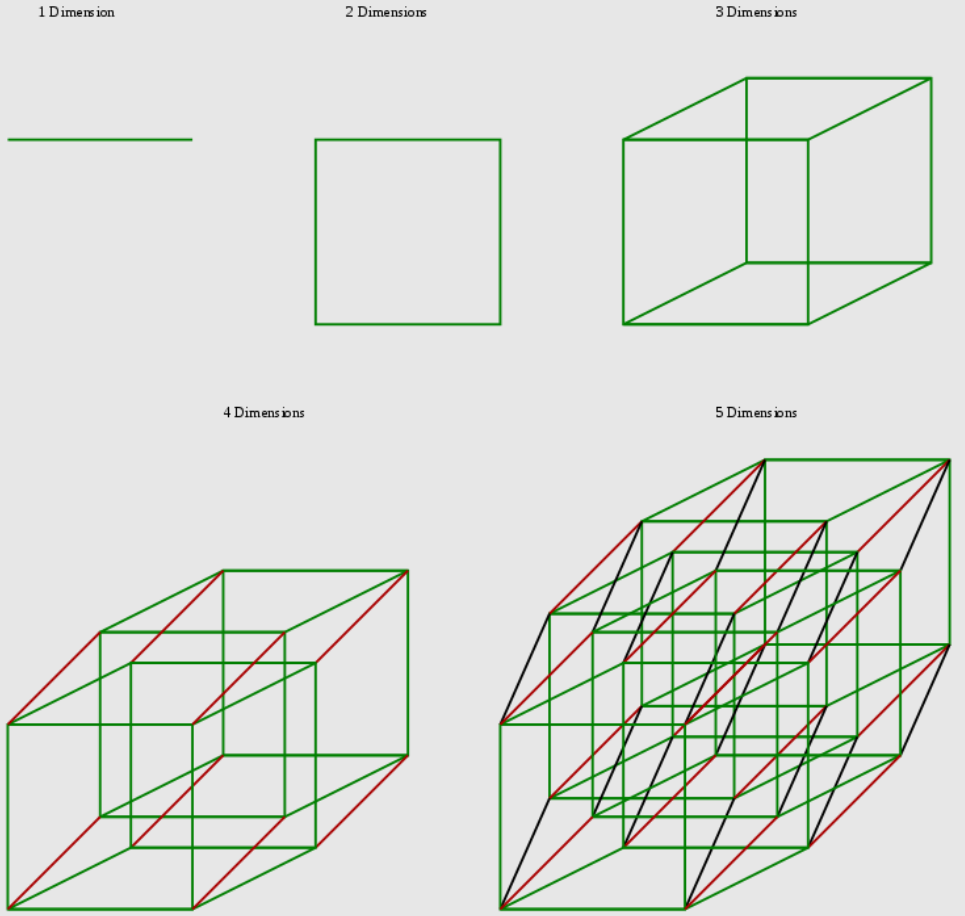
Dimensionality



차원이란?

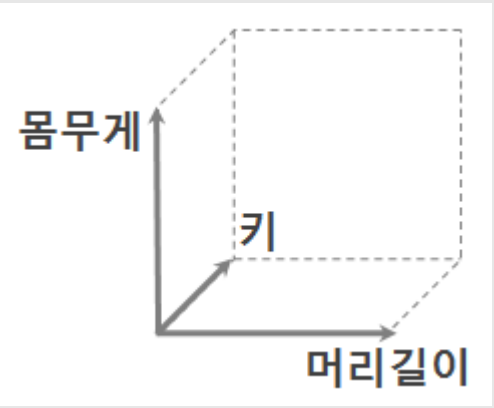
Dimensionality

- 수학적 차원
  - ➔ 하나의 정의가 여러 필요를 만족시키는 것은 불가능
  - ➔ 여러 개념들은 전부 n차원 유클리드 공간 E^n에서 유래
  - ➔ 벡터 공간:
    벡터 공간의 기저에 속하는 원소의 수
  - ➔ 다양체:
    연결 위상다양체는 부분적으로 n차원 유클리드 공간과 위상 동형이다.
  - ➔ 가환환의 크룰(Krull) 차원:
    소 아이디얼들의 강한 포함관계에 의한 사슬의 길이가 가질 수 있는 극대값



Dimensionality

키	몸무게	머리길이
168	58	10
162	55	30
159	49	25
165	45	40



- 내가 가진 데이터를 그래프로 표현하기 위한 축의 개수가 내가 가진 데이터 공간의 차원이므로 **'축의 개수=변수의 수=차원'**이 된다.
- **'변수의 수가 늘어난다=차원이 늘어난다=데이터 공간이 커진다'**라고도 할 수 있다.

Dimensionality & Accuracy

- 반지름의 길이가 $r = 1$ 인 구를 생각해보자. 물론 차원은 D 차원이다.
- 이 때의 구의 부피는? (이 때 입력 차원에 독립적인 일반식을 생각해보자.)

$$V_D(r) = K_D r^D \qquad (1.75)$$

- 참고로 3차원에서 구의 부피는 $(4/3)\pi r^3$ 이다. (기억들이 나시는지?)
- 그리고 거리가 $r = 1$ 인 경우의 구의 부피에서 $r = 1 - e$ 인 구의 부피를 빼는 것을 상상해보자.
  - 이 때 e 는 보통 매우 작은 값을 사용할 것이다.
  - 이러면 다차원 구의 부피는 표면 층의 부피가 된다.
  - 물론 e 값을 어느 정도의 작은 값으로 취하는 가에 따라 표피만 의미하는 것이 아니라 수박 껍질처럼 고려될수도 있다.
- 그럼 이제 e 값을 조절하면서 이 구간의 부피비를 생각해볼 수 있을 것이다.

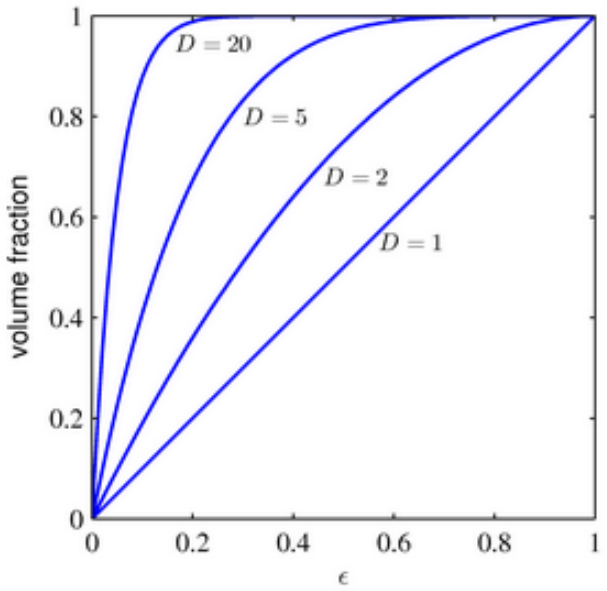
$$\frac{V_D(1) - V_D(1-e)}{V_D(1)} = 1 - (1-e)^D \qquad (1.76)$$

- 식은 매우 단순한데, 원래 구의 부피를 분모로 놓고 e 로 인해 결정되는 겉 껍질의 부피를 분자로 놓게 된다.
  - 즉, 원래의 부피와 겉면의 부피의 비를 확인하고 싶은 것이다.

〈http://norman3.github.io/prml/docs/chapter01/4.html〉

Dimensionality & Accuracy

- 이 때 e 값이 변화할 때 원래 부피와의 비율을 살펴보자.



- 그림을 잘 보면 차원이 증가할수록(D가 커질수록) e 값이 작더라도 원래 볼륨 크기와 근접하게 됨을 알 수 있다.
- 이걸 다른 관점에서 이야기하자면 차원이 증가할수록 전체 볼륨 크기의 대부분은 표면에 위치하게 된다는 것이다.
    - 3차원까지만 머릿속에 그릴 수 있는 우리로서는 약간 이해하기 어렵지만 그래도 상상을 해 볼수는 있다.

〈http://norman3.github.io/prml/docs/chapter01/4.html〉

Dimensionality & Accuracy

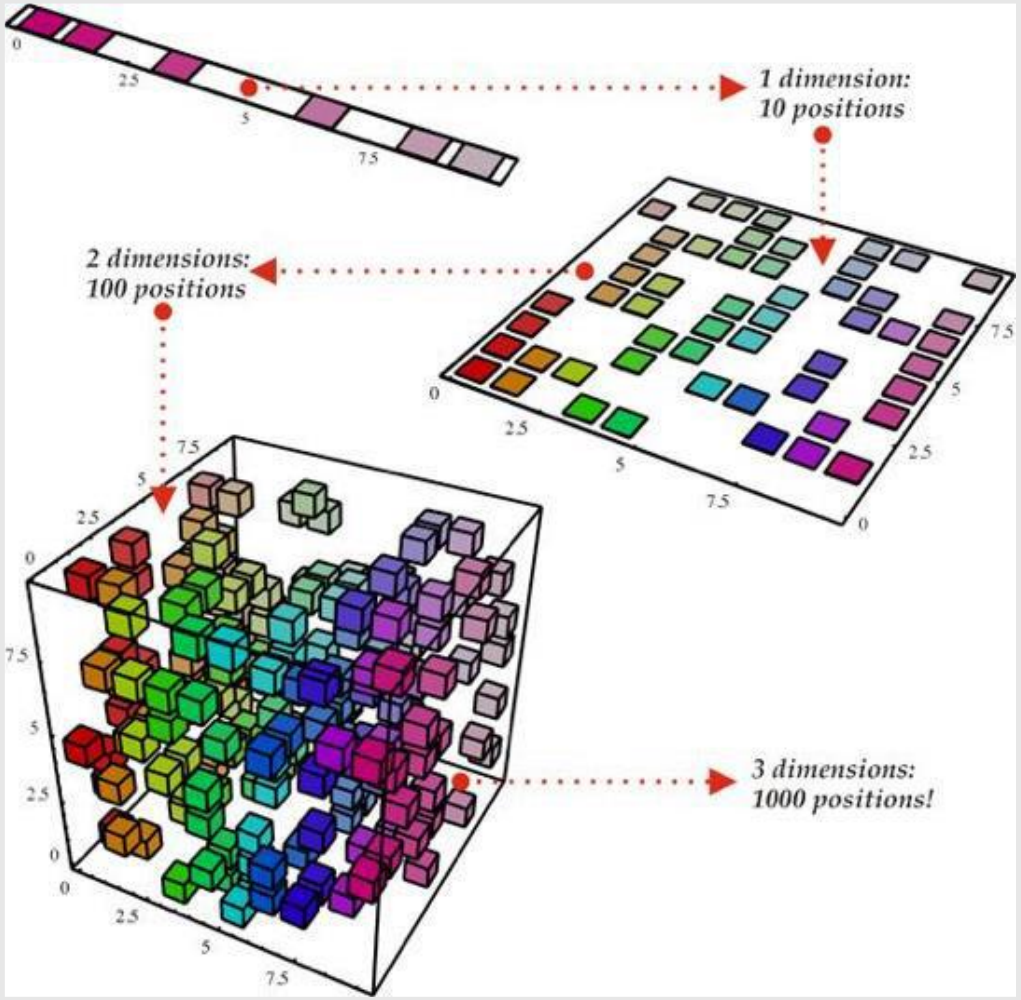
그러면 데이터가 무조건 많으면 되지 않을까?

Curse of Dimensionality

〈http://thesciencelife.com/archives/1001〉

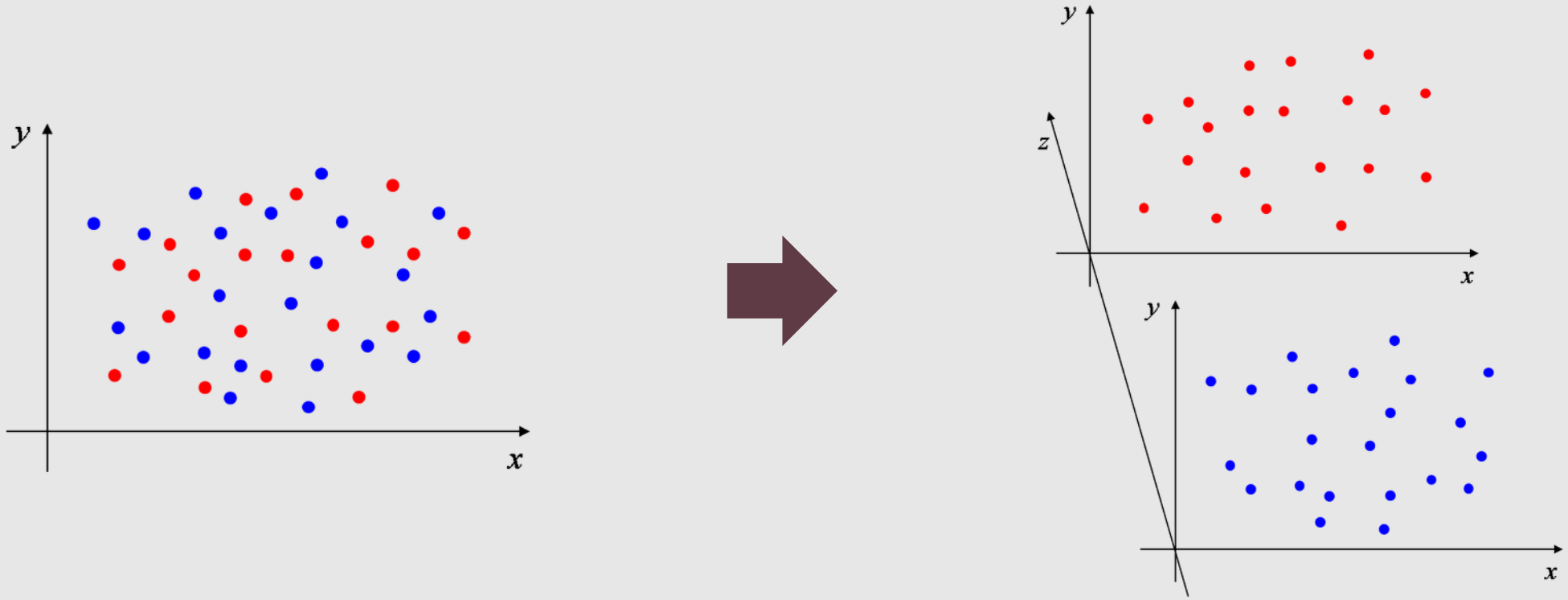
모델을 학습할 때 독립샘플이 많을수록 학습이 잘 되는 것과 달리
차원이 커질수록 학습이 더 어려워진다.

Curse of Dimensionality



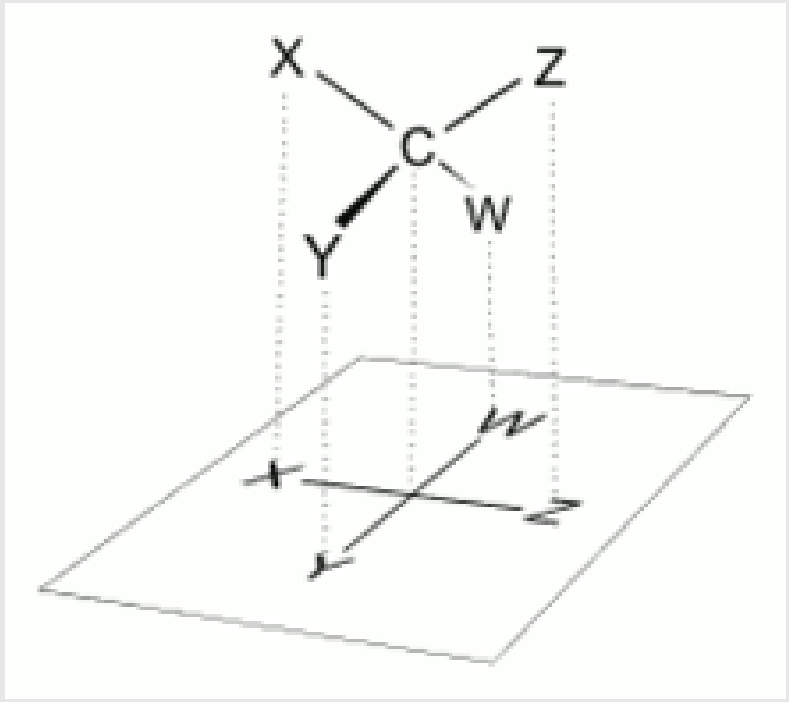
〈https://brunch.co.kr/@kakao-it/161〉

Curse of Dimensionality

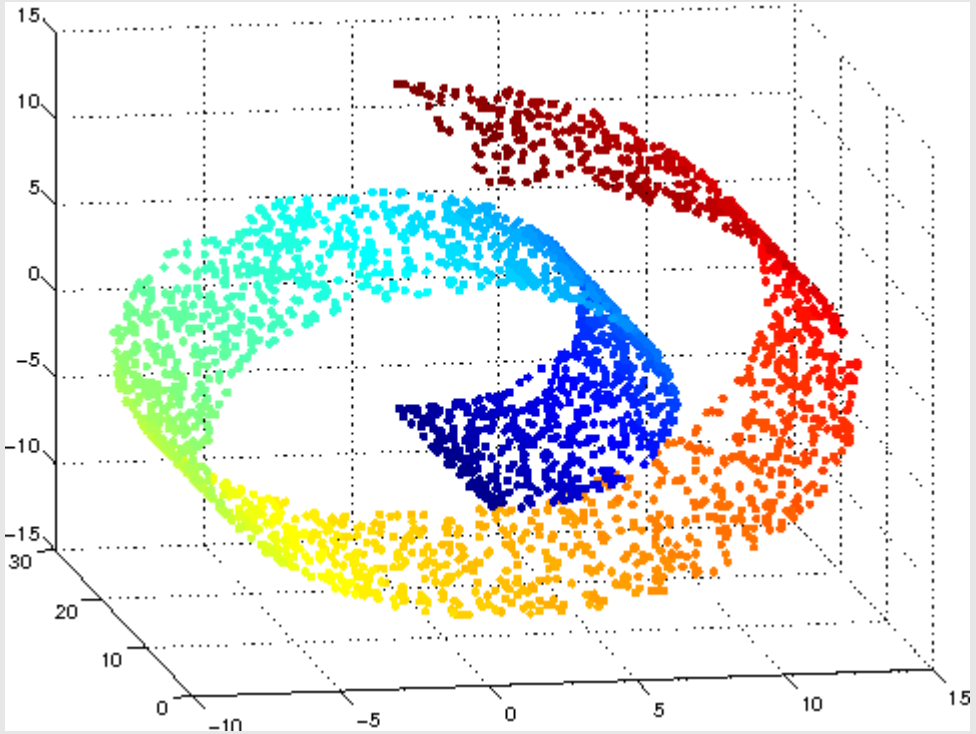


- 차원을 단순히 투영시키기에는 정보 손실이 우려됨
- 차원을 '잘' 축소할 방법이 필요

Dimensionality Reduction Base



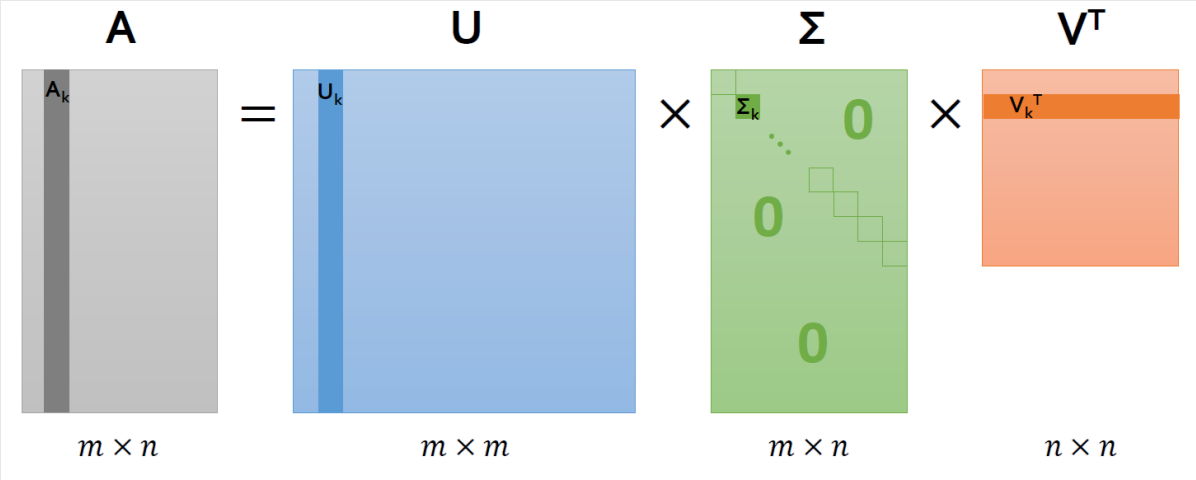
투영
(Projection)



매니폴드
(Manifold)

Dimensionality Reduction

- **특이값 분해 (Singular Value Decomposition; SVD)**
  - ➔ Thin SVD, Compact SVD, Truncated SVD, 등 으로 2차적 활용한 것들을 많이 사용
  - ➔ M x N크기의 데이터 행렬을 아래와 같이 분해하는 방법

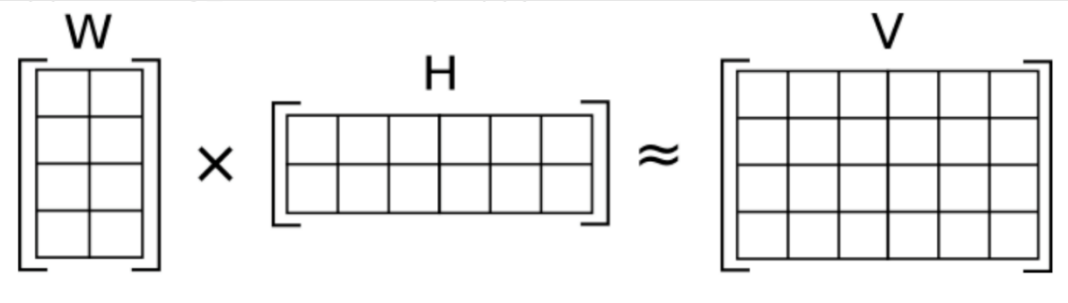


- **주성분 분석 (Principal Component Analysis; PCA)**
  - ➔ 가장 대중적으로 사용
  - ➔ 분산을 최대한 보존하며 직교하는 새 기저(축)을 찾아 고차원 공간의 표본들을 선형 연관성이 없는 저차원 공간으로 매핑하는 방법
  - ➔ SVD와 사실상 같은 것이다.

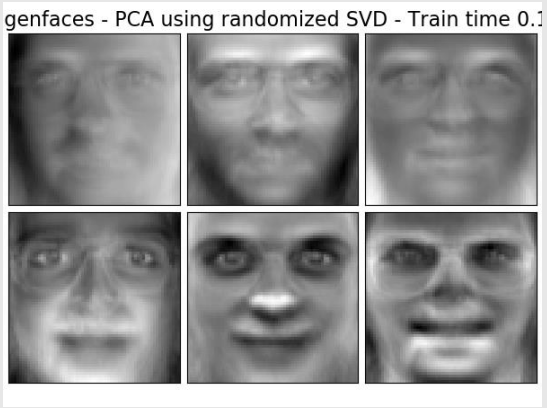
Dimensionality Reduction

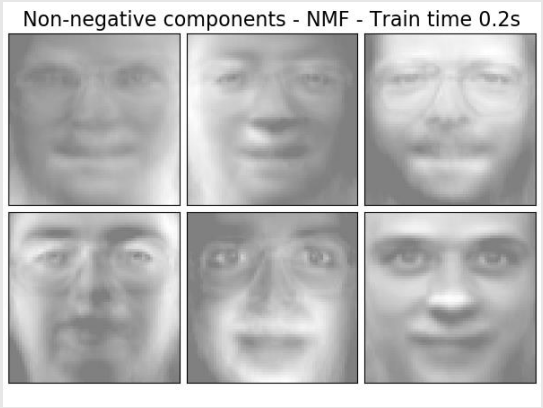
- **비음수 행렬 인수분해 (Non-negative Matrix Factorization; NMF)**

➔ 매트릭스의 거리를 최적화하여 샘플을 두 개의 매트릭스로 분해



➔ PCA와의 가장 큰 차이는 뺄셈이 없이 구성 요소를 겹쳐서 덧셈 방식만으로 계산을 하기 때문에 이미지 처리에 용이하다. (이미지가 변질되는 가능성이 줄어든다.)

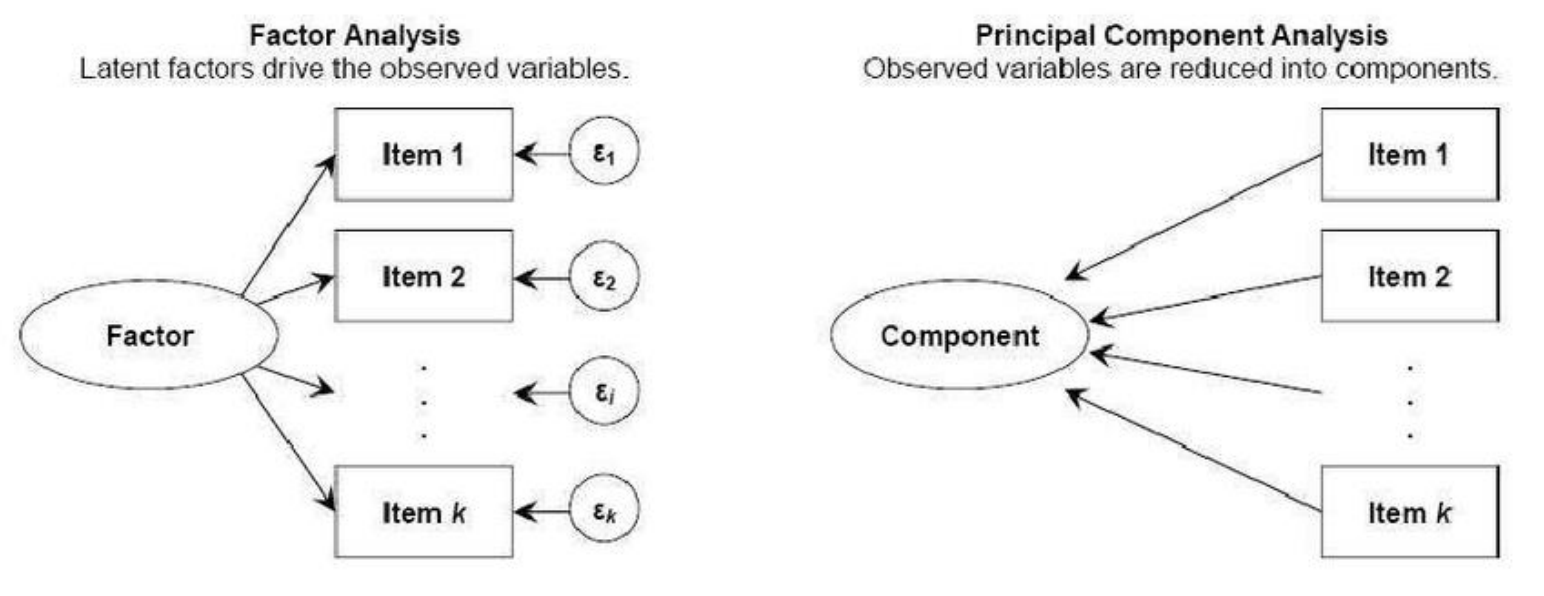




〈http://scikit-learn.org/stable/modules/decomposition.html#nmf〉

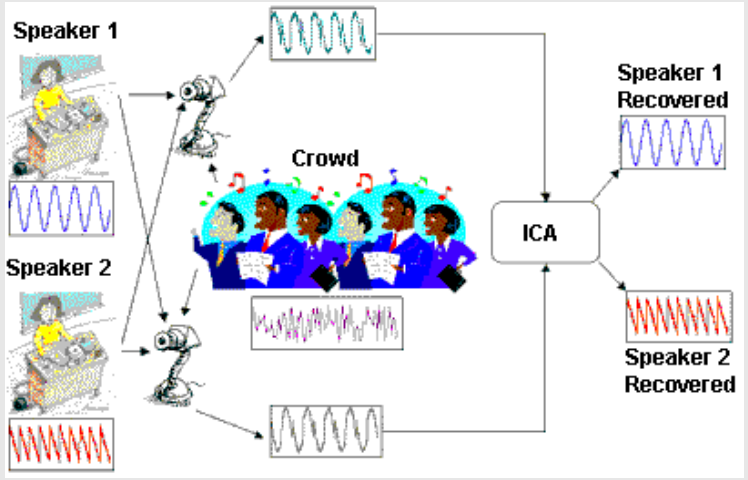
Dimensionality Reduction

- **요인/인자 분석(Factor Analysis)**
  - ➔ 변수들 간의 상관관계를 고려하여 저변에 내재된 개념인 요인들을 추출해내는 분석방법
  - ➔ 공통인자 방법과 주성분 방법으로 나뉨
  - ➔ 주성분 분석의 큰 범주로 요인/인자 분석을 생각하기도 하는데 이는 명백히 틀림
    [PCA = Data Reduction / FA = Structure Detection]



Dimensionality Reduction

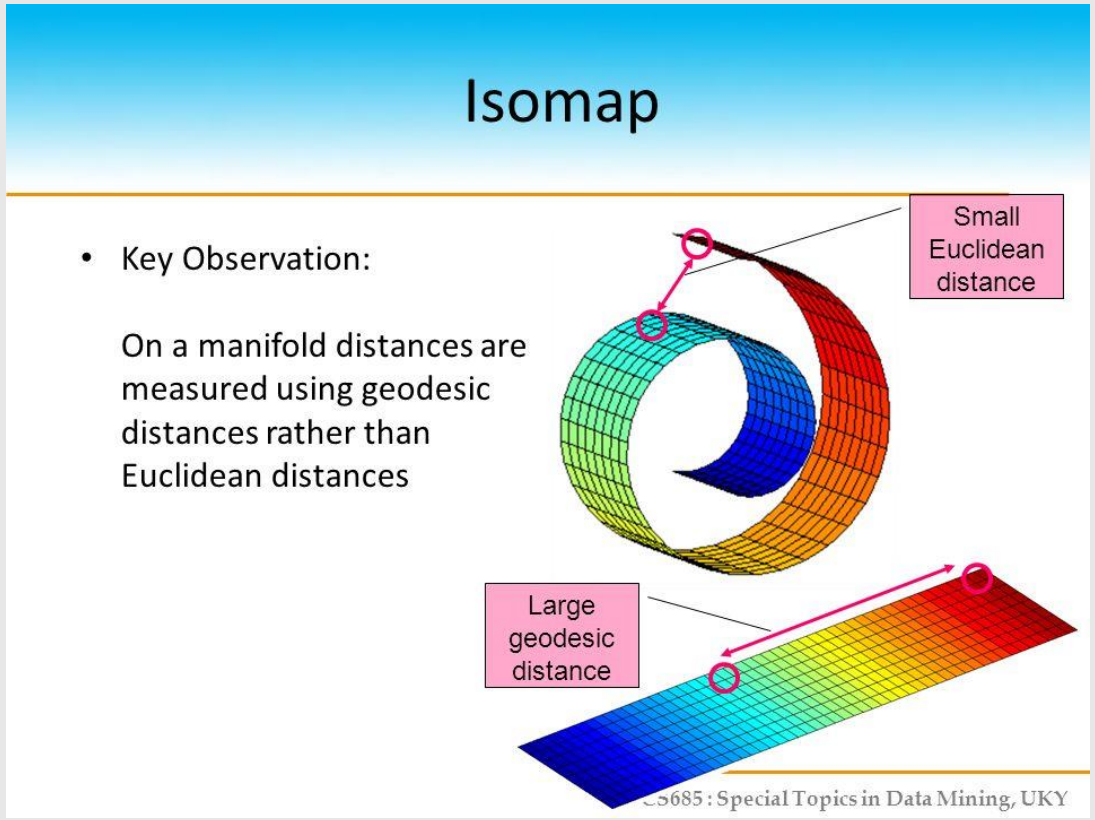
- **독립 성분 분석(Independent Component Analysis)**
  - ➔ 다변량의 신호를 통계적으로 독립적인 하부 성분으로 분리하는 계산 방법
  - ➔ 각 성분은 비 가우스 성 신호로서 서로 통계적 독립을 이루는 성분으로 구성



- **다차원 척도법(Multidimensional Scaling; MDS)**
  - ➔ 대상들을 유사성 지각에 관한 정보를 토대로 시각적으로 나타냄
  - ➔ 그 위치들로부터 유사성 지각의 토대가 된 차원들을 추정
  - ➔ 추정거리에 가장 많이 사용되는 것은 Kruscal의 Stress방식

Dimensionality Reduction

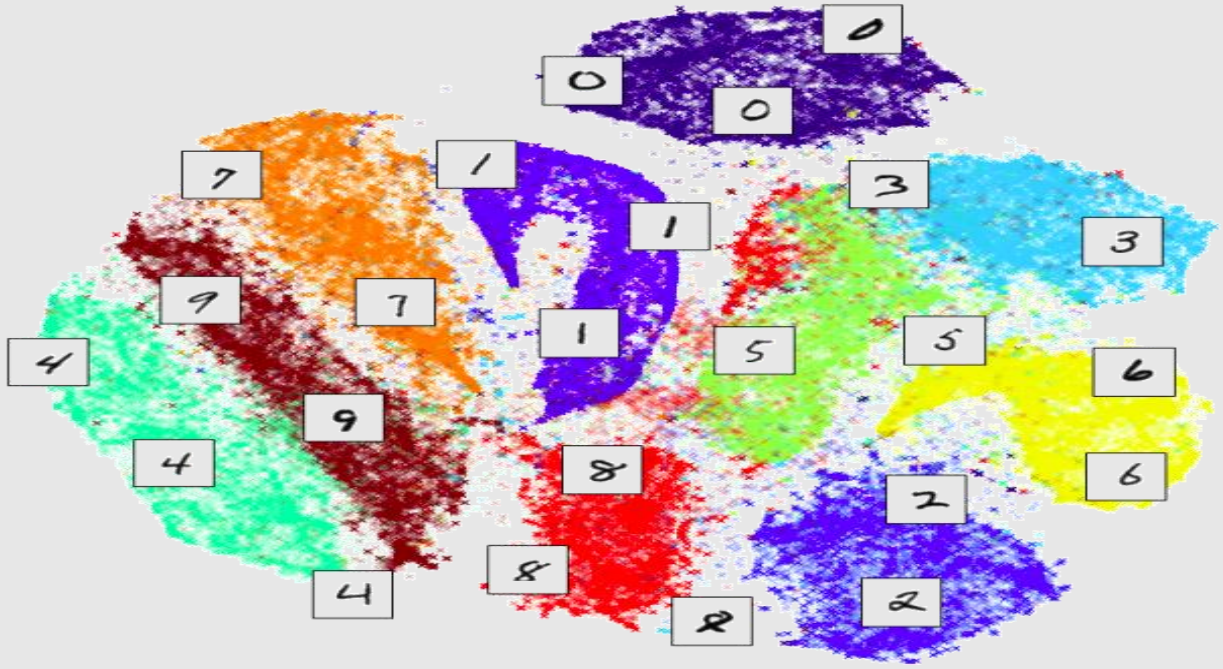
- **Isomap**
  - ➔ 각 포인트들의 이웃을 찾아서 이웃을 기반으로 가중치가 적용된 그래프 형성
  - ➔ 모든 포인트들 사이의 가장 짧은 거리 계산 후 MDS 실행



〈https://slideplayer.com/slide/8705933/〉

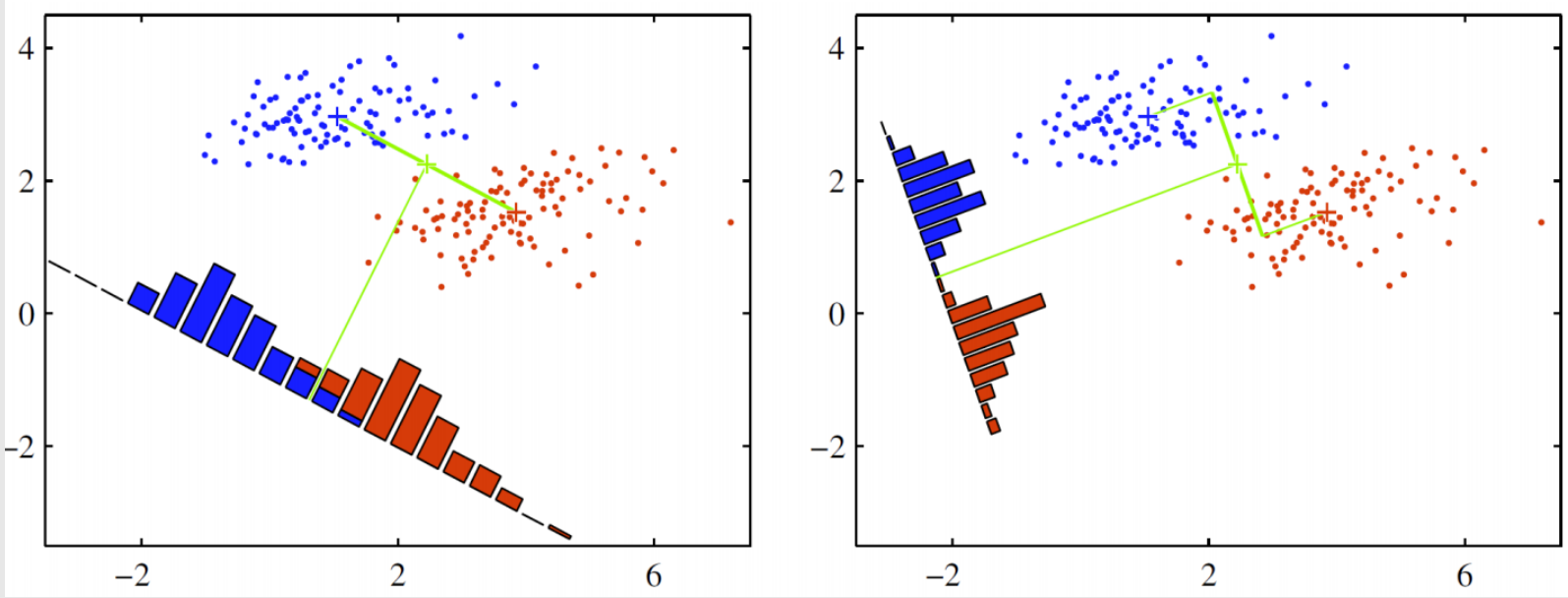
Dimensionality Reduction

- **t-SNE(t-Distributed Stochastic Neighbor Embedding)**
  - ➔ 비선형 차원 축소 기법 중 하나로 Geoffrey Hinton이 만들었다.
  - ➔ 각 데이터 쌍에 대한 결합분포를 만든 뒤, 비슷한 데이터는 선택될 확률이 높게 하는 분포를 생성



Dimensionality Reduction

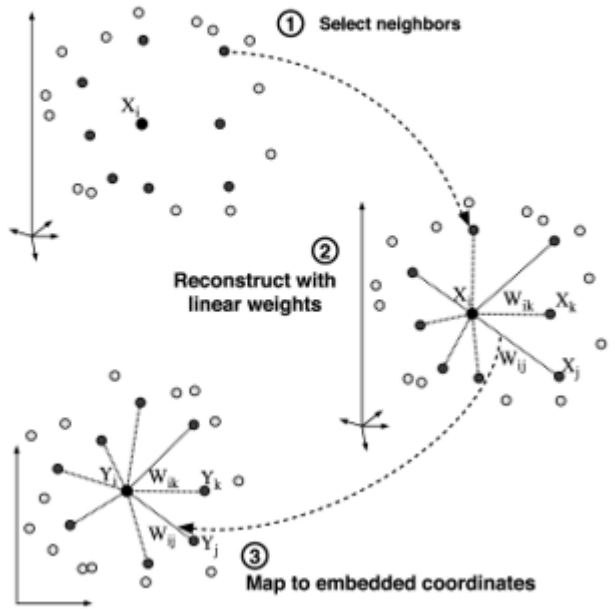
- **선형 판별 분석 (Linear Discriminant Analysis; LDA)**
  - ➔ 특정한 축에 사영한 후 두 범주를 잘 구분할 수 있는 직선을 찾는 것이 목적
  - ➔ 차원 축소에서 쓰이는 경우는 이 특정한 축을 활용하는 것



Dimensionality Reduction

- **지역 선형 임베딩 (Locally Linear Embedding; LLE)**
  - ➔ 고차원의 공간에서 인접해 있는 데이터들의 선형적 구조를 보존하면서 저차원으로 Embedding하는 것
  - ➔ 다른 비선형 방법보다 다루기 쉽다 (논문에서 말한 내용)

LLE Algorithm Step



LLE Algorithm Step

Step 1: Compute the neighbors of each data point $\vec{X_i}$

Step 2: Compute the wieght W_{ij} that best reconstruct each data point from its neighbors, minimizing the cost function by constrained linear fits

$$\varepsilon(W) = \sum_i \left| \vec{X_i} - \sum_j W_{ij} \vec{X_j} \right|^2$$

$s.t\ W_{ij} = 0$ *if* $\vec{X_j}$ *does not belong to the neighbors of* $\vec{X_i}$

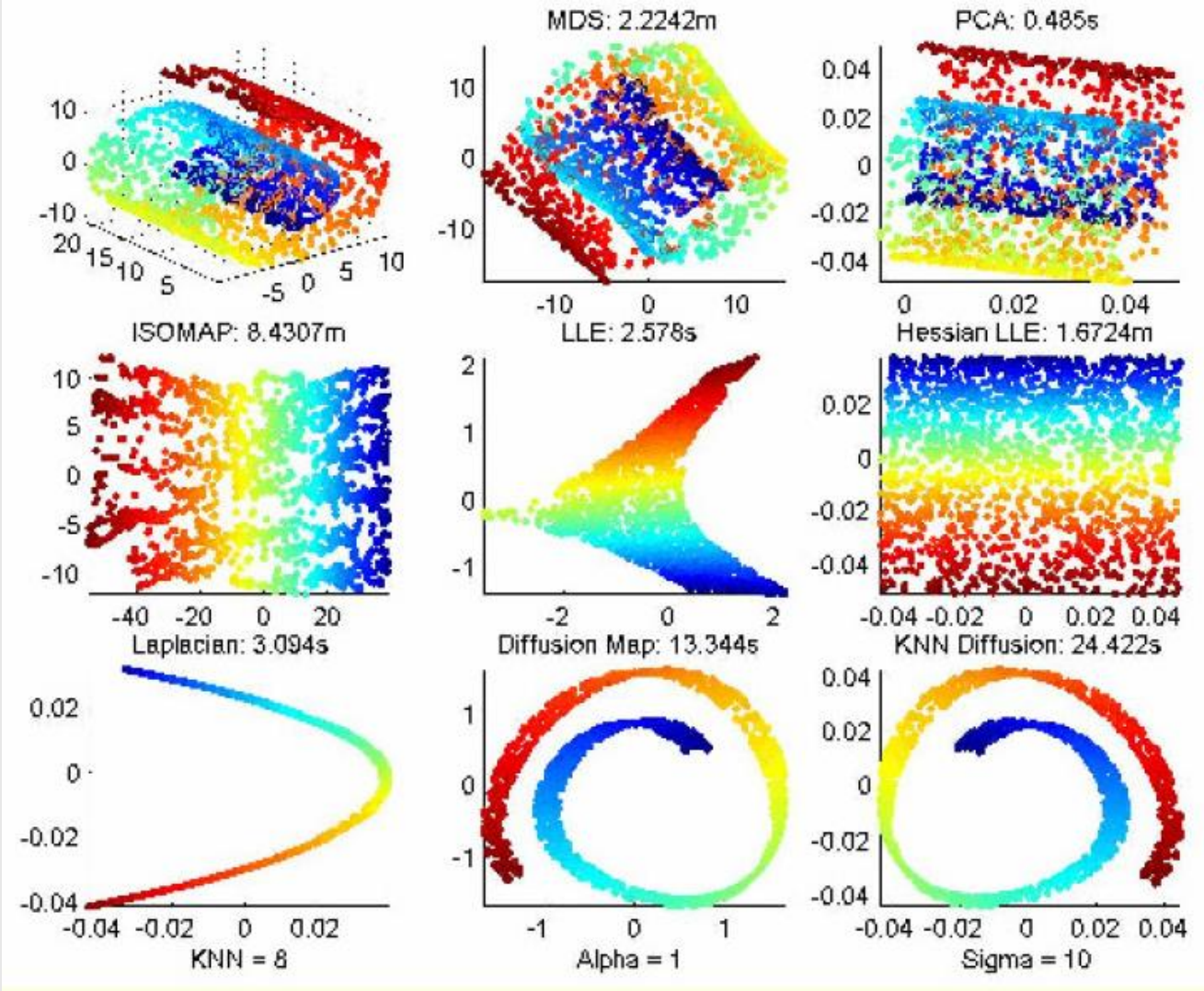
$$\sum_j W_{ij} = 1, \textit{for all } i$$

Step 3: Compute the vectors best reconstructed by the weights W_{ij}, minimizing the quadratic form by its bottom nonzero eigenvectors

$$\Phi(Y) = \sum_i \left| \vec{Y_i} - \sum_j W_{ij} \vec{Y_j} \right|^2$$

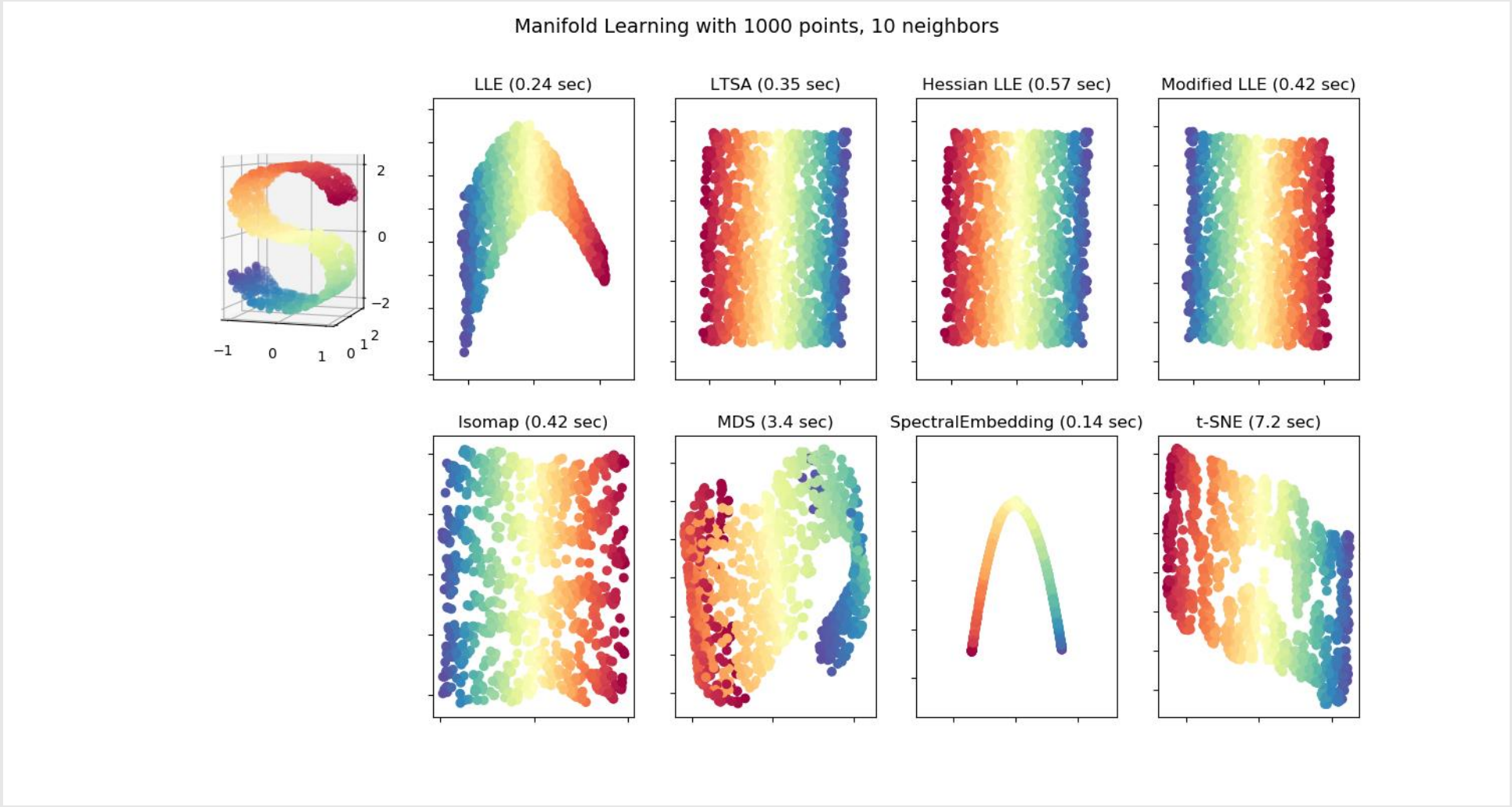
〈http://t-lab.tistory.com/26〉

Swiss Roll Example

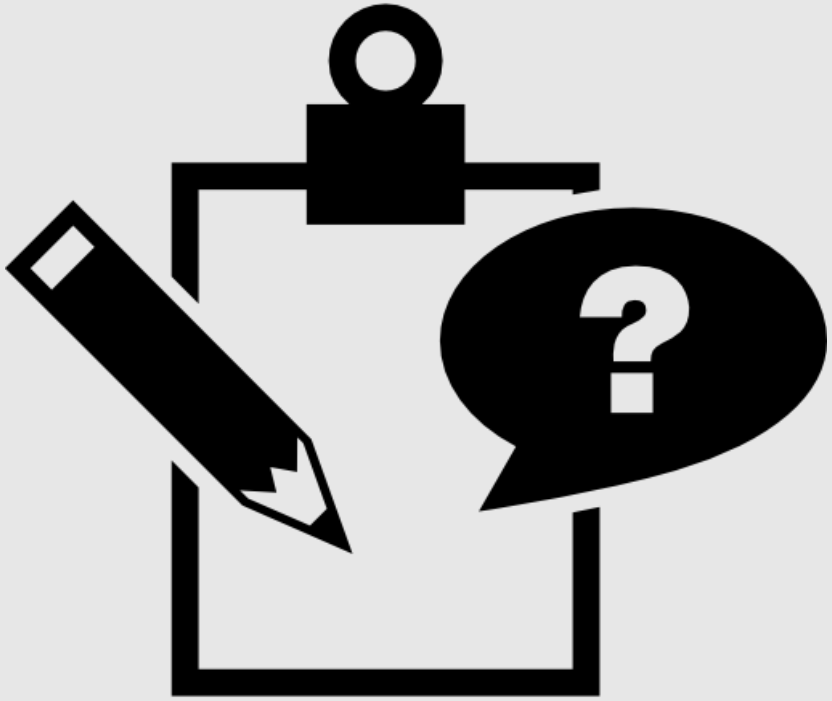


〈http://www.cs.haifa.ac.il/~rita/uml_course/add_mat/mani_presentation.pdf〉

Swiss Roll Example



〈http://scikit-learn.org/stable/modules/manifold.html〉



Up and Running with TensorFlow

Tensorflow

TensorFlow

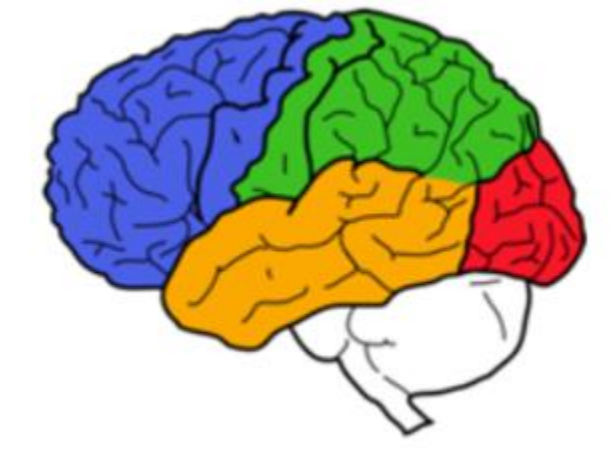
Tensorflow

"We saw that the only difference between cows that produce 30 liters of milk a day and those that produce 10 liters was the animal's health. Could technology make cows healthier, and in doing so, help farmers grow their businesses?"
– Yasir Khokar

Tensorflow

Yasir's story is an inspiring example of how machine learning can help tackle all kinds of problems. It also reminds us of the origins of TensorFlow itself—a product designed to help people everywhere make the most of AI.

〈https://ai.google/stories/tensorflow/〉



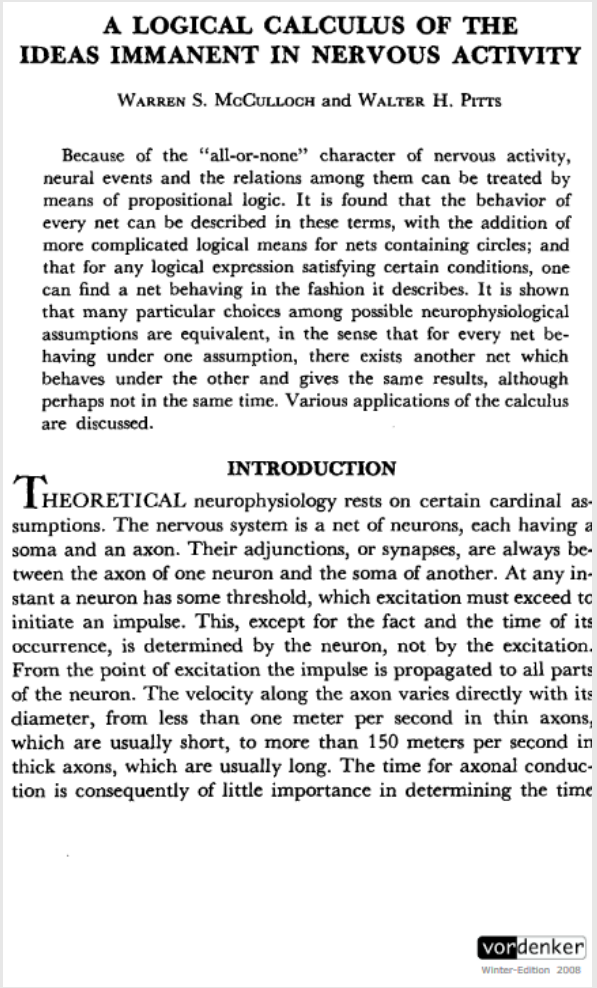
〈DistBelief〉

TensorFlow

Deeplearning History

1943년
역사적인 논문
Pitts & McCulloh

‘A Logical Calculus of The Ideas Immanent In Nervous Activity’



A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY

WARREN S. MCCULLOCH and WALTER H. PITTS

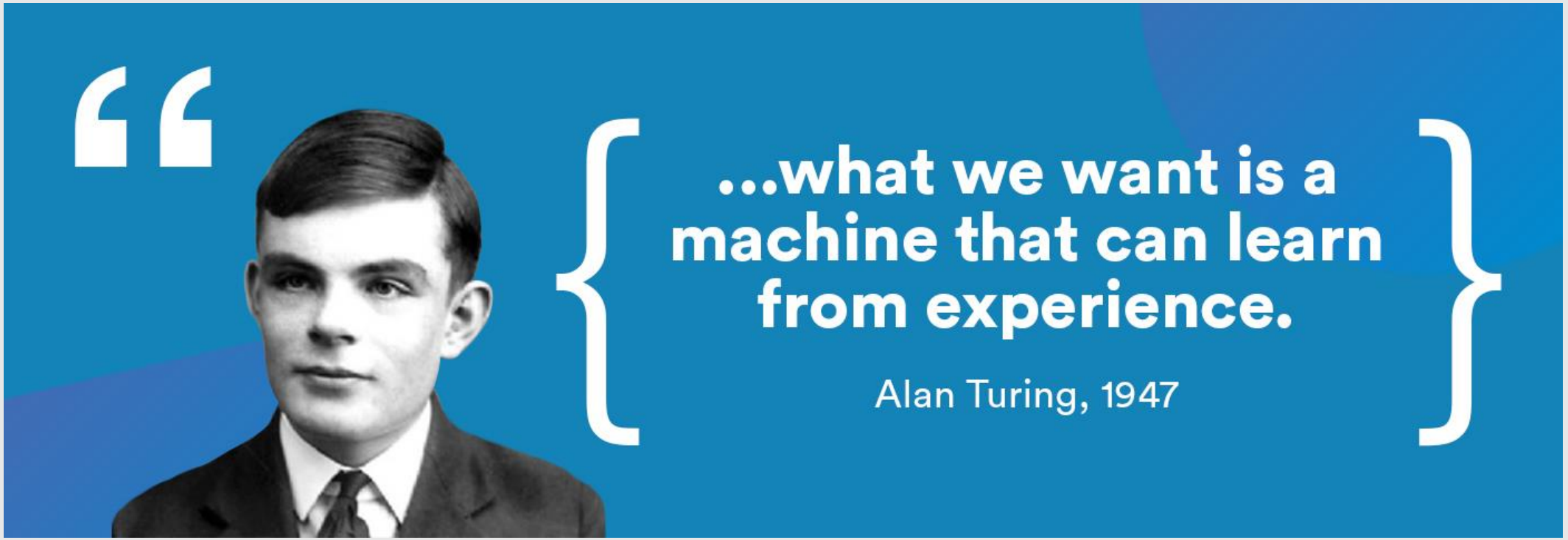
Because of the "all-or-none" character of nervous activity, neural events and the relations among them can be treated by means of propositional logic. It is found that the behavior of every net can be described in these terms, with the addition of more complicated logical means for nets containing circles; and that for any logical expression satisfying certain conditions, one can find a net behaving in the fashion it describes. It is shown that many particular choices among possible neurophysiological assumptions are equivalent, in the sense that for every net behaving under one assumption, there exists another net which behaves under the other and gives the same results, although perhaps not in the same time. Various applications of the calculus are discussed.

INTRODUCTION

THEORETICAL neurophysiology rests on certain cardinal assumptions. The nervous system is a net of neurons, each having a soma and an axon. Their adjunctions, or synapses, are always between the axon of one neuron and the soma of another. At any instant a neuron has some threshold, which excitation must exceed to initiate an impulse. This, except for the fact and the time of its occurrence, is determined by the neuron, not by the excitation. From the point of excitation the impulse is propagated to all parts of the neuron. The velocity along the axon varies directly with its diameter, from less than one meter per second in thin axons, which are usually short, to more than 150 meters per second in thick axons, which are usually long. The time for axonal conduction is consequently of little importance in determining the time

vordenker
Winter-Edition 2008

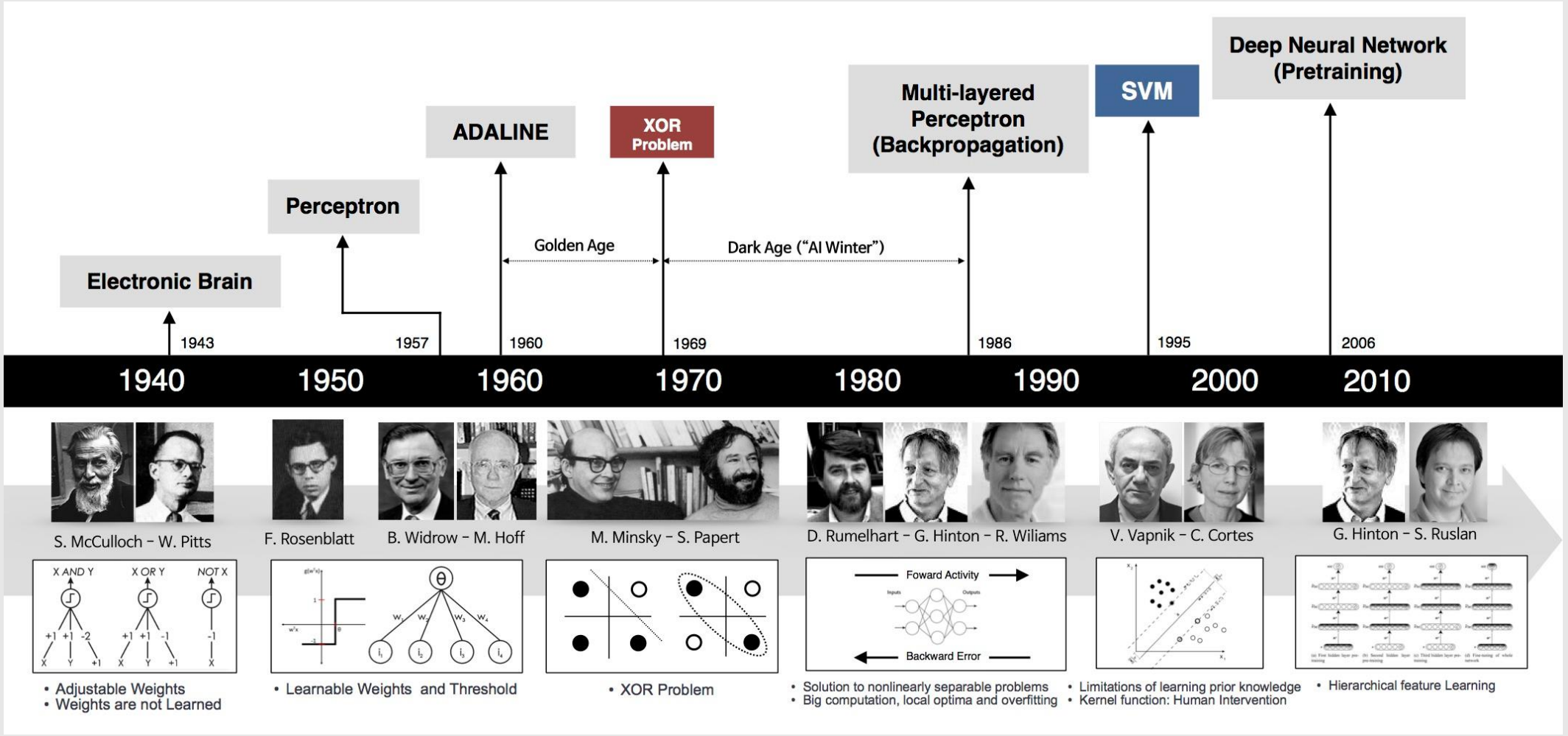
Deeplearning History



〈https://www.import.io/post/history-of-deep-learning/〉

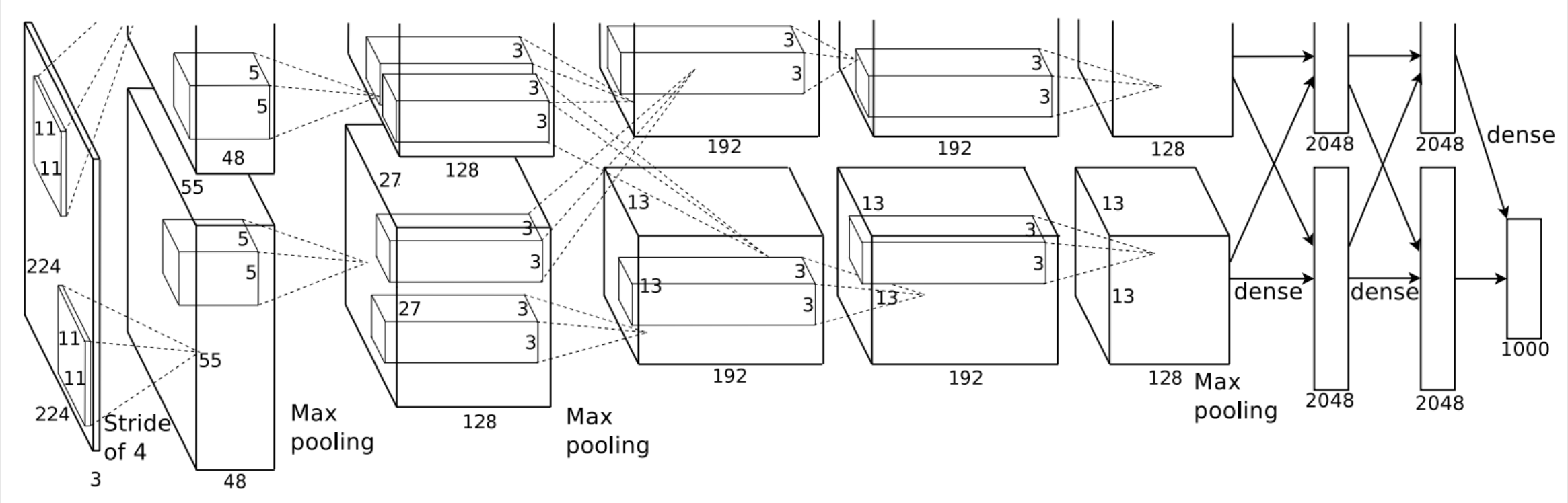
➔ 1950년

Deeplearning History



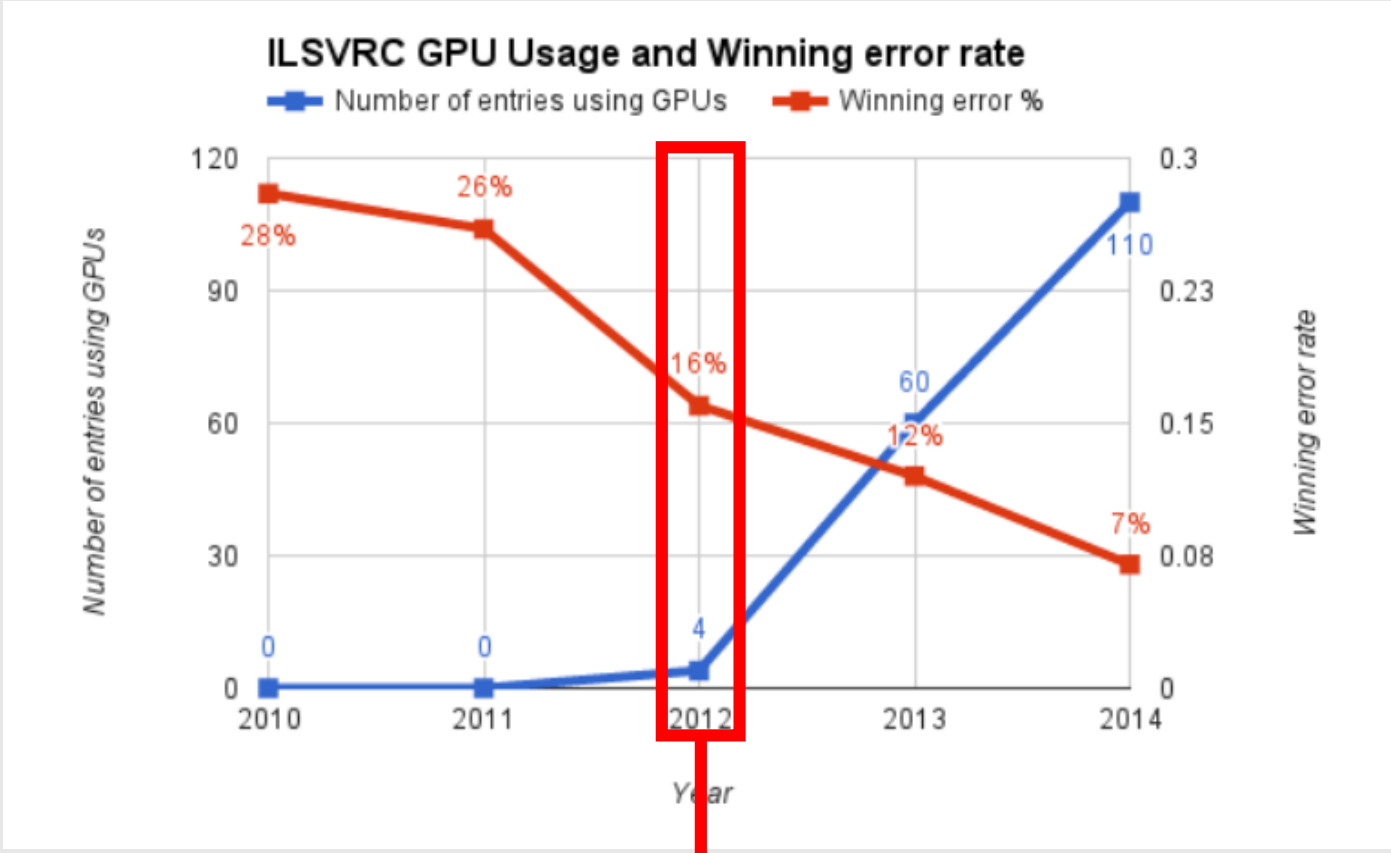
〈https://beamandrew.github.io/deeplearning/2017/02/23/deep_learning_101_part1.html〉

Deeplearning History



〈https://ratsgo.github.io/deep%20learning/2017/10/09/CNNs/〉

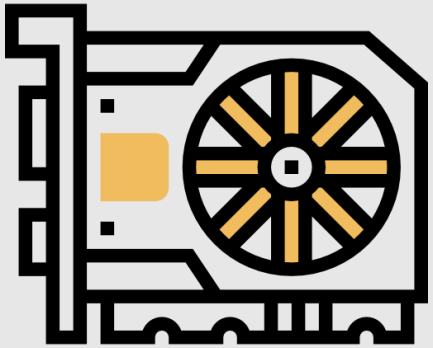
Deeplearning History

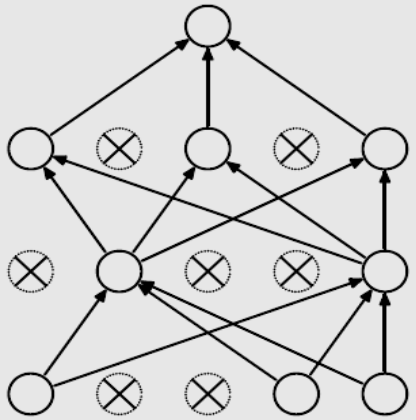


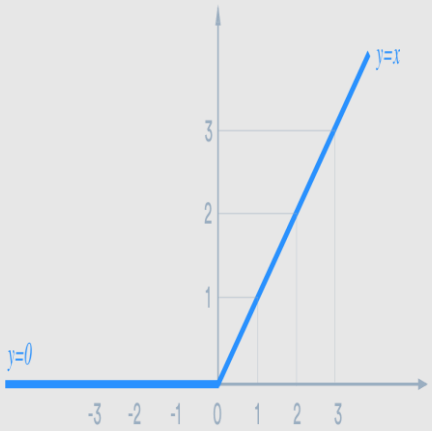
<https://beamandrew.github.io/deeplearning/2017/02/23/deep_learning_101_part1.html>

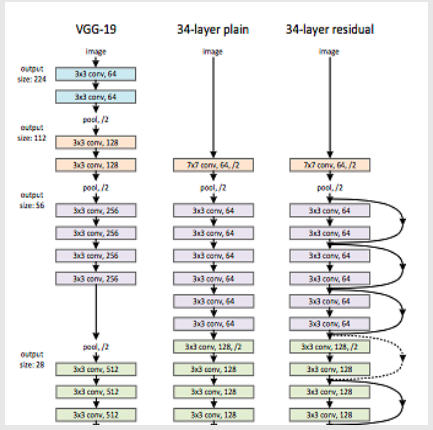
AlexNet

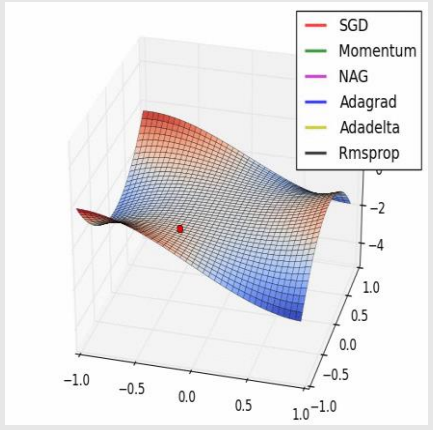
Deeplearning History

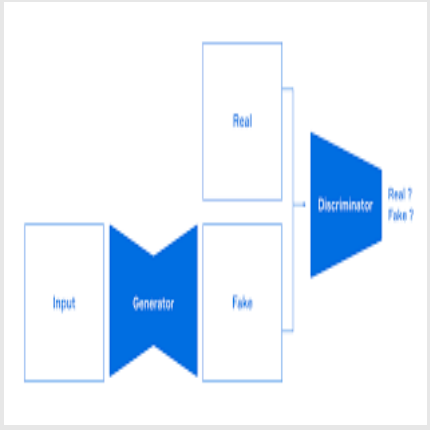








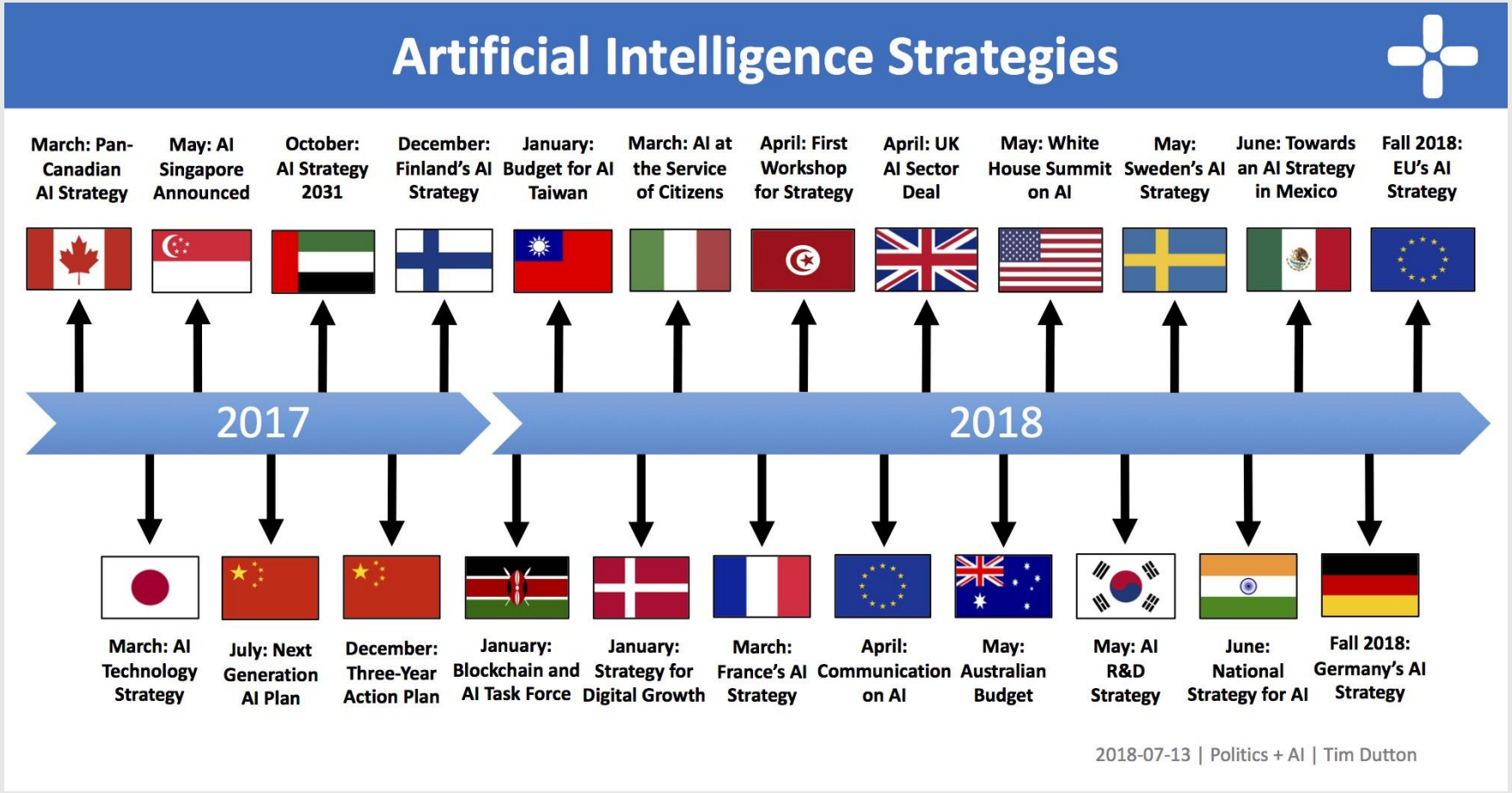




Korean Deeplearning History

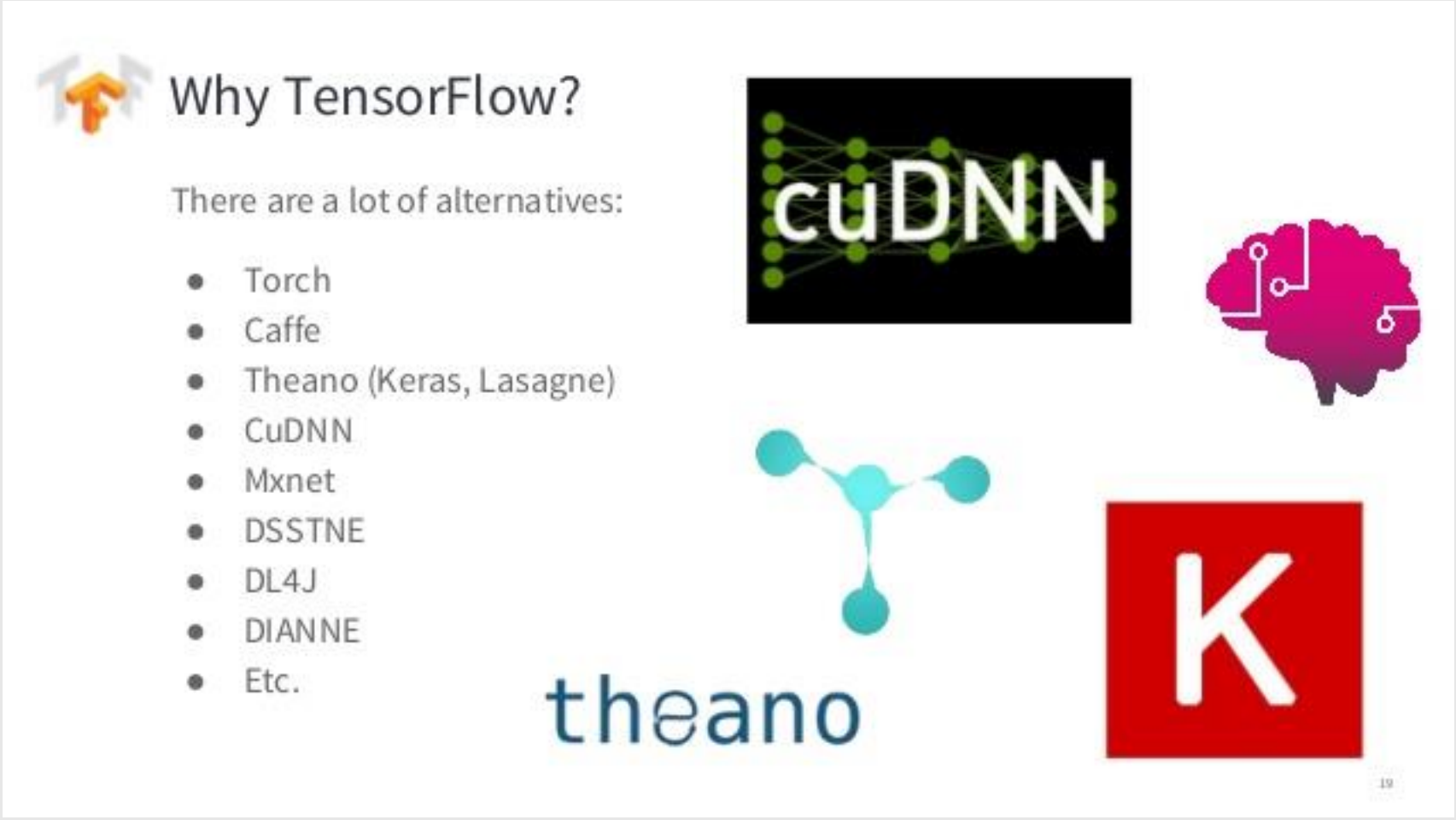
〈https://becominghuman.ai/alphagos-mastery-continues-3cd01493f539〉

Korean Deeplearning History



〈https://medium.com/politics-ai/an-overview-of-national-ai-strategies-2a70ec6edfd〉

Why Tensorflow?



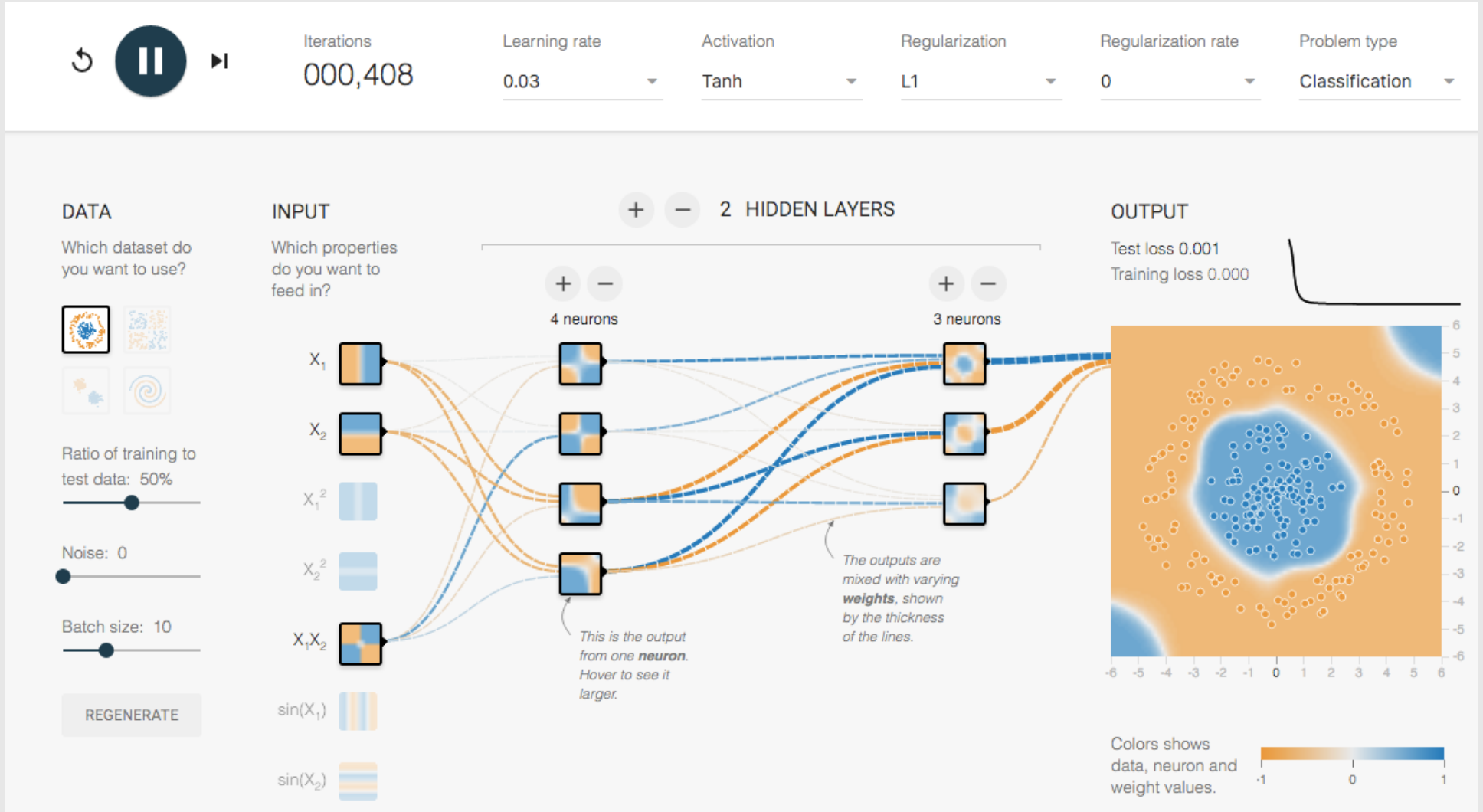
〈https://www.slideshare.net/matthiasfeys/introduction-to-tensorflow-66591270〉

Why Tensorflow?

"제가 생각하는 답은 **커뮤니티**입니다. 특히 저 같은 엔지니어에게 있어서 라이브러리를 선택할 때 가장 중요한 기준은 커뮤니티라고 생각합니다. 실무에 적용했을 때 생기는 문제점들을 해결하거나, 라이브러리 자체에 버그가 있을 때 얼마나 빠르게 수정되는가 하는 그런 것들. 바로 그런 요인들이 실무를 하는 엔지니어에게는 가장 중요한 부분이라고 할 수 있을 것입니다."

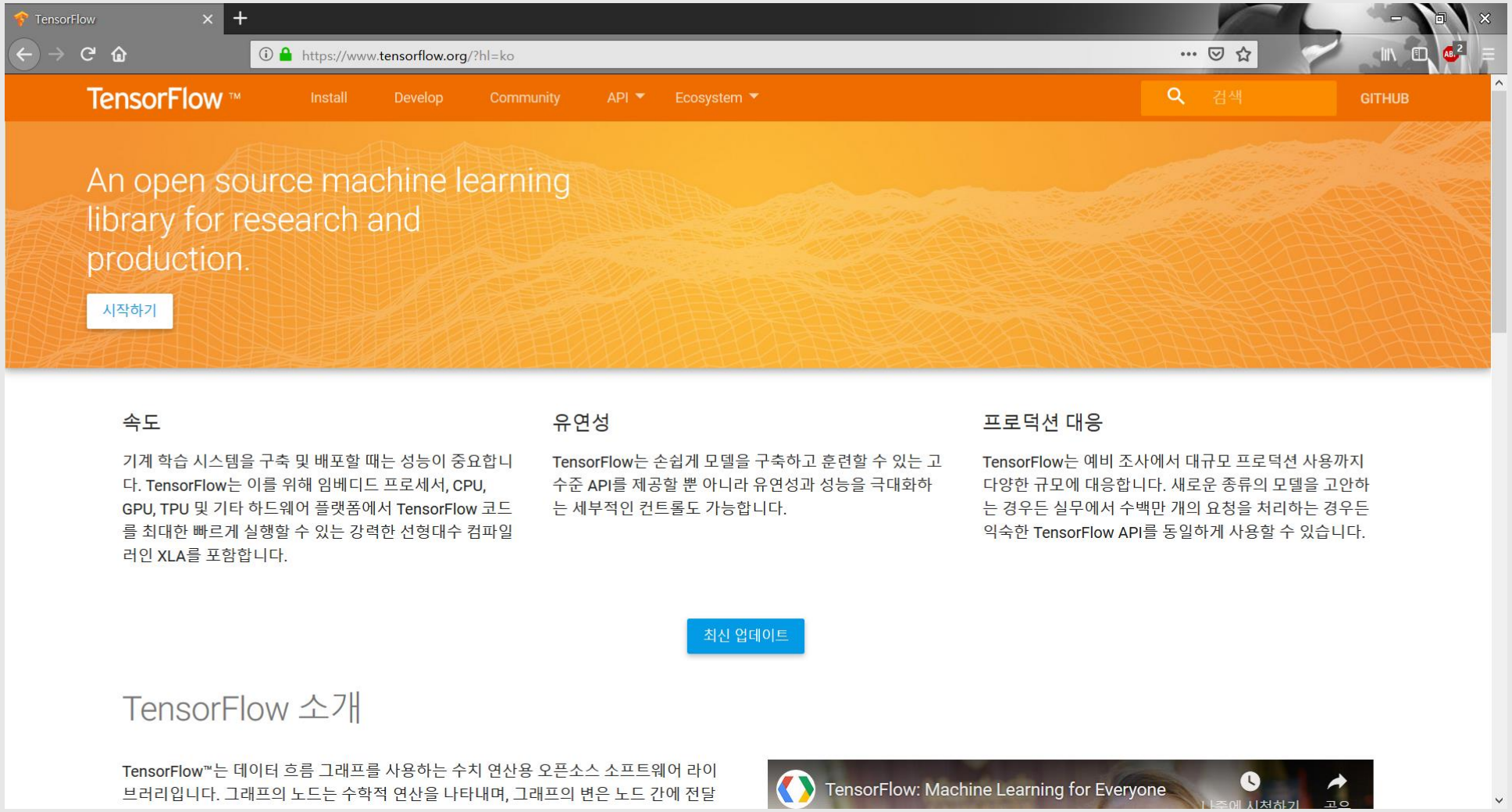
"당연하게도 구글 역시 텐서플로 커뮤니티를 상당히 적극적으로 지원하며, 한국에서는 페이스북의 TensorFlow KR 커뮤니티가 매우 활발하게 활동하고 있습니다. 그러므로 한국에 계신 연구자분들, 특히 머신러닝과 딥러닝을 처음 접하는 엔지니어라면 텐서플로로 시작하는 것이 딥러닝을 더 수월하게 익히는 길이라고 생각합니다."

Tensorflow Mechanism



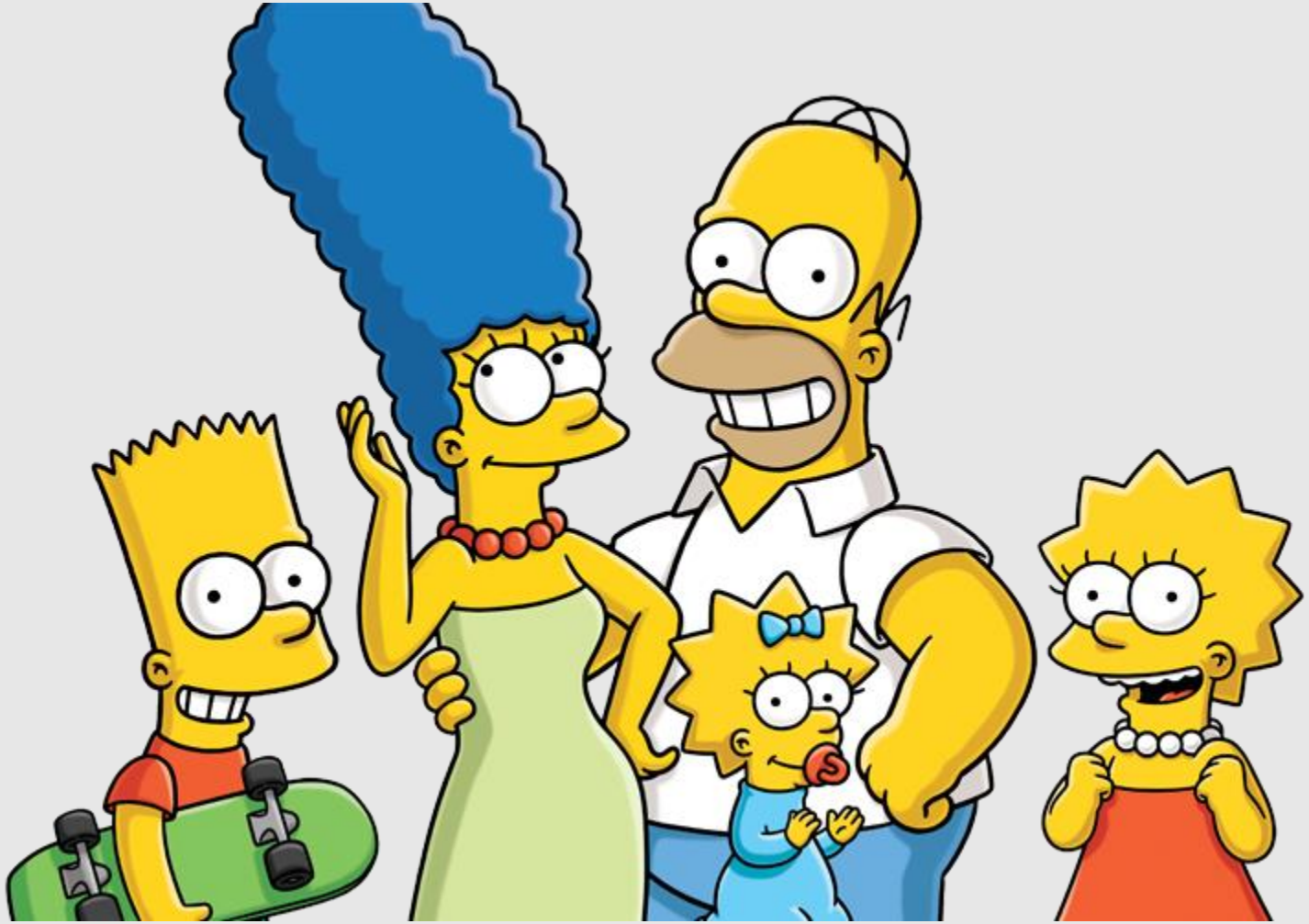
<https://techcrunch.com/2016/04/13/google-launches-distributed-version-of-its-tensorflow-machine-learning-library/>

Tensorflow API

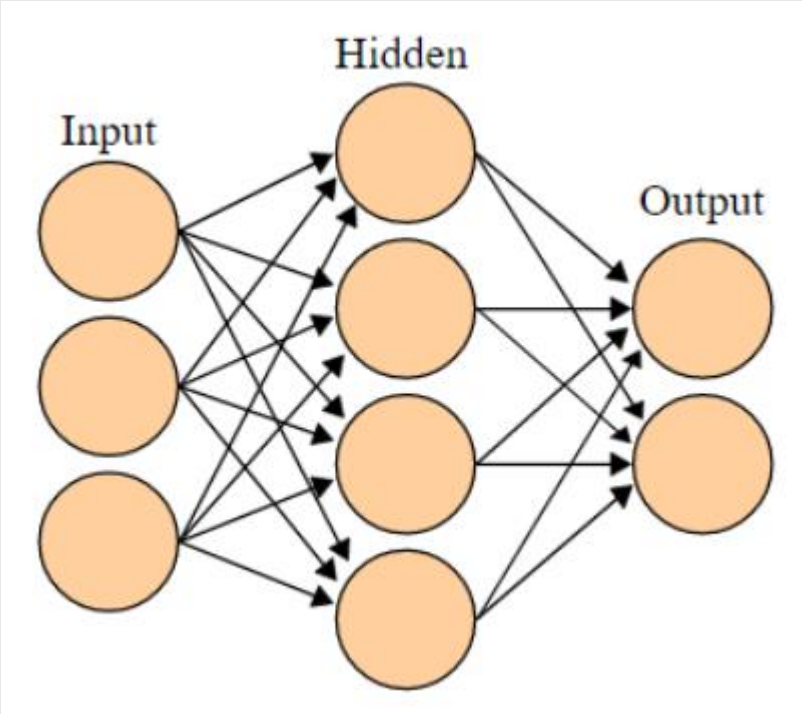


Tensorflow API

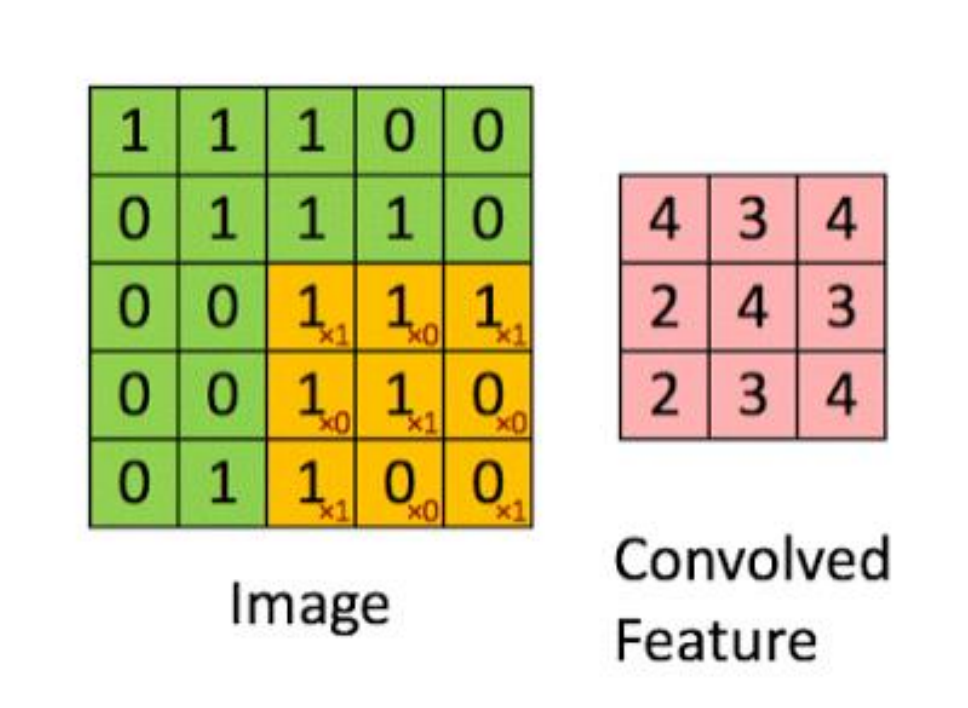
The Simpson's Classifier



About CNN

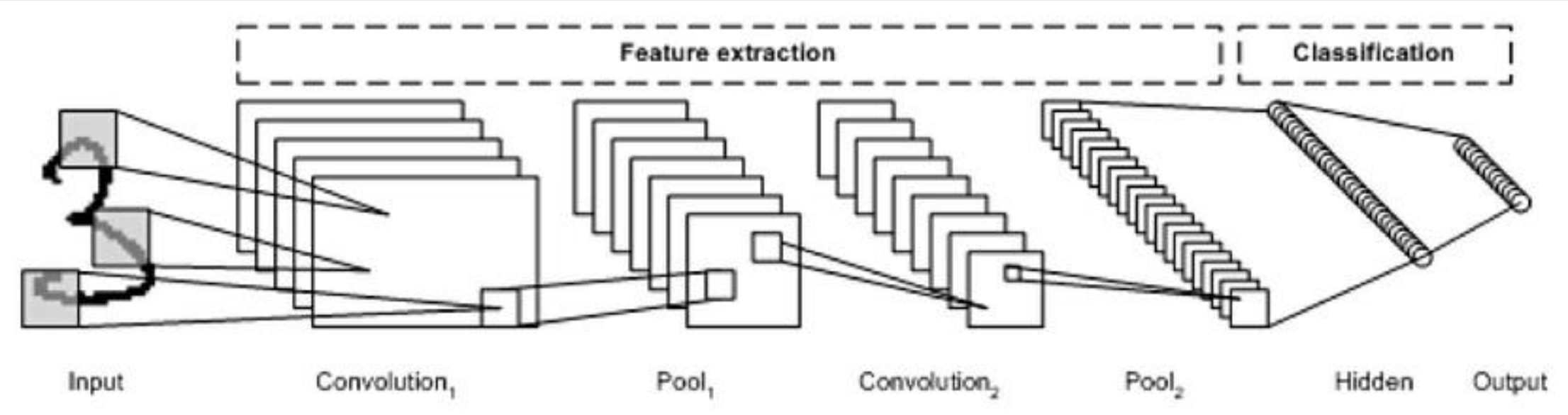


ANN



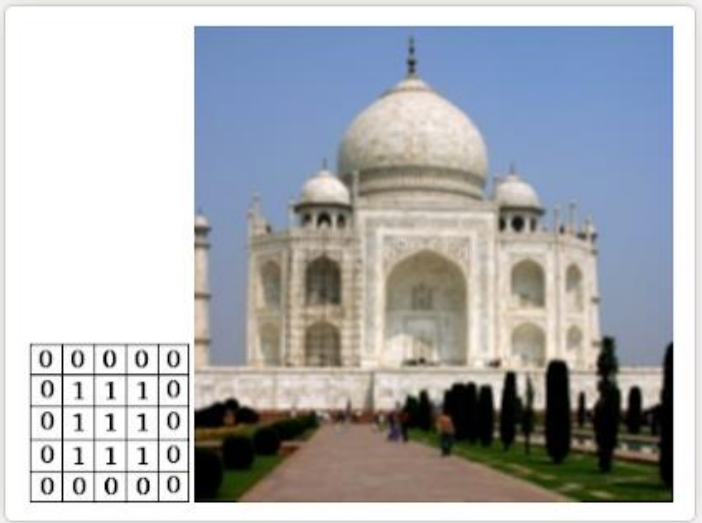
CNN

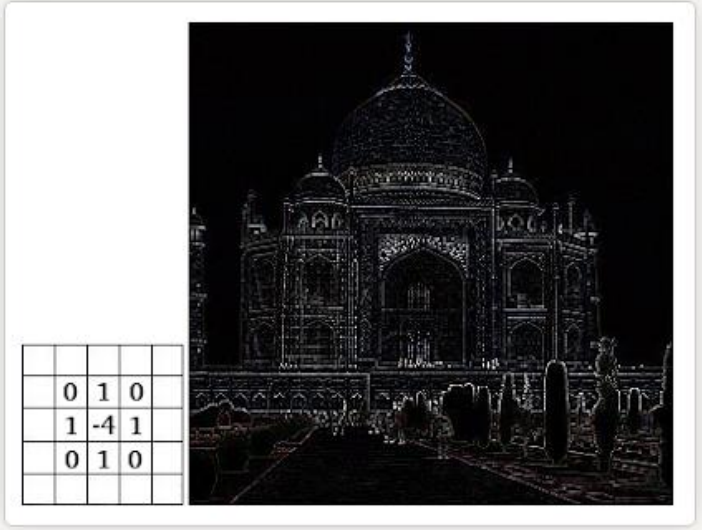
About CNN



〈http://taewan.kim/post/cnn/〉

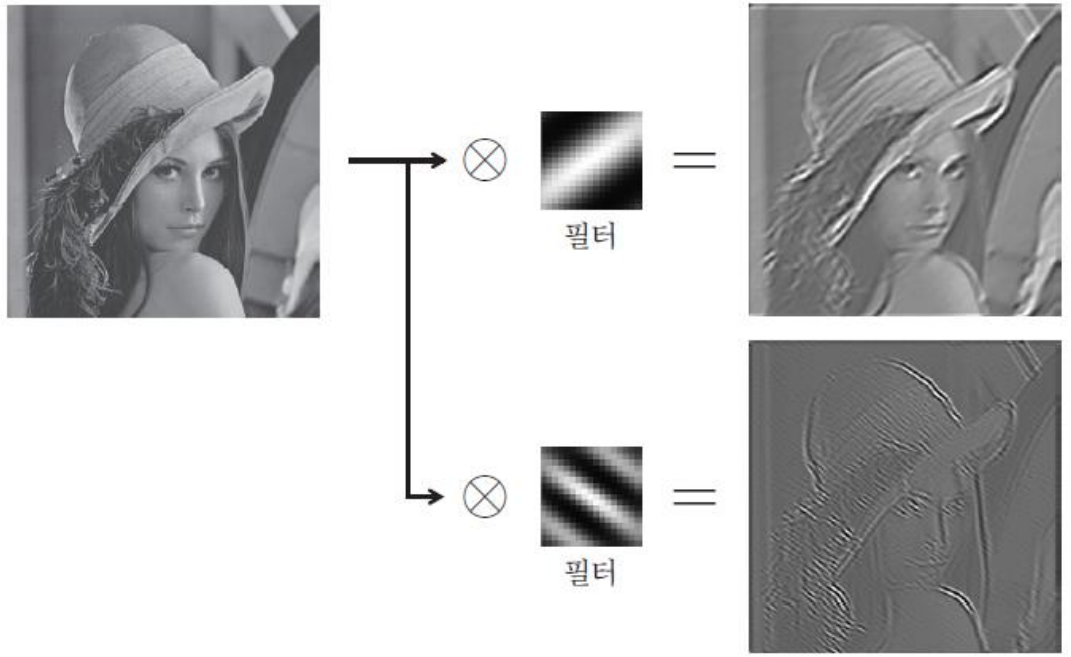
About CNN





- 각 레이어의 입출력 데이터의 형상 유지
- 이미지의 공간 정보를 유지하면서 인접 이미지와의 특징을 효과적으로 인식
- 복수의 필터로 이미지의 특징 추출 및 학습
- 추출한 이미지의 특징을 모으고 강화하는 Pooling레이어
- 필터를 공유 파라미터로 사용하기 때문에, 일반적인 ANN보다 학습 파라미터의 수가 적음

About CNN



필터를 합성곱한 결과의 예. 512 × 512 크기의 이미지에 17 × 17 크기의 필터 두 종류를 합성곱하였다. 필터는 보기 쉽도록 이미지보다 상대적으로 크게 확대 표시하였다. 또, 필터와 합성곱 결과 이미지는 음의 값을 갖는 픽셀이 검게, 양의 값을 갖는 픽셀은 희게 나타나도록 명암값을 조정하였다.

〈딥러닝 제대로 시작하기, 2016, 제이펍〉

About CNN

Examples

```
>>> from tempfile import TemporaryFile
>>> outfile = TemporaryFile()
>>> x = np.arange(10)
>>> y = np.sin(x)
```

Using **savez** with *args, the arrays are saved with default names.

```
>>> np.savez(outfile, x, y)
>>> outfile.seek(0) # Only needed here to simulate closing & reopening file
>>> npzfile = np.load(outfile)
>>> npzfile.files
['arr_1', 'arr_0']
>>> npzfile['arr_0']
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

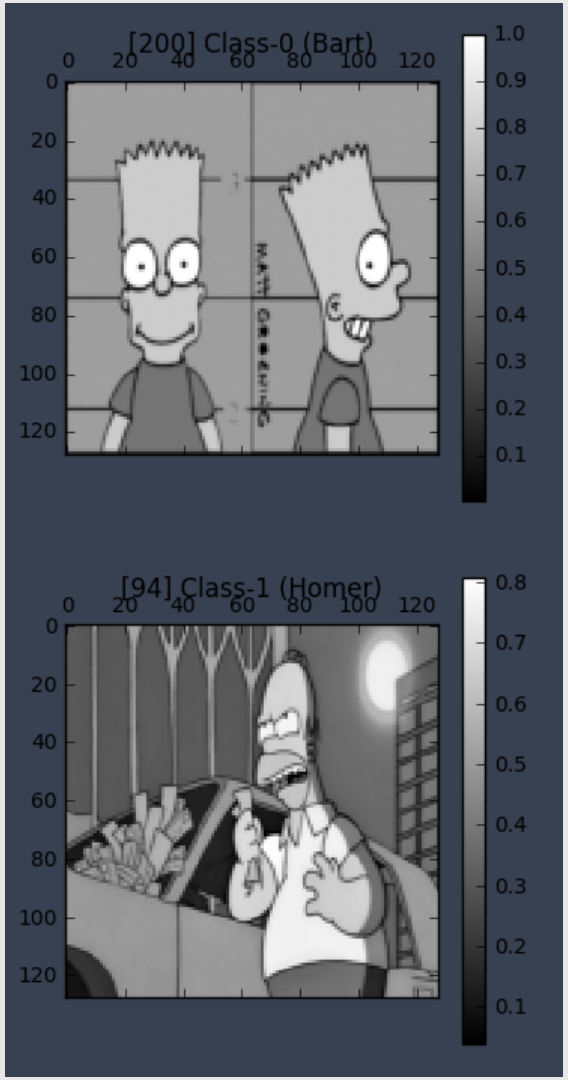
Using **savez** with **kwds, the arrays are saved with the keyword names.

```
>>> outfile = TemporaryFile()
>>> np.savez(outfile, x=x, y=y)
>>> outfile.seek(0)
>>> npzfile = np.load(outfile)
>>> npzfile.files
['y', 'x']
>>> npzfile['x']
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

〈https://docs.scipy.org/doc/numpy-1.15.1/reference/generated/numpy.savez.html〉

<http://www.numpy.org/_static/numpy_logo.png>

Implementation: Data Read



SciPy

imread,
imresize

```
cwd = os.getcwd()
loadpath = cwd + "/data/" + data_name + ".npz"
l = np.load(loadpath)
print (l.files)

#Parse data
trainimg_loaded = l['trainimg']
trainlabel_loaded = l['trainlabel']
testimg_loaded = l['testimg']
testlabel_loaded = l['testlabel']
categories_loaded = l['categories']

print ("[%d] Training images" % (trainimg_loaded.shape[0]))
print ("[%d] Test Images" % (testimg_loaded.shape[0]))
print ("Loaded From [%s]" % (savepath))
```

Implementation: Function Definition

```python
def weight_variable(shape):
    initial = tf.truncated_normal(shape, stddev = 0.1)
    return tf.Variable(initial)

def bias_variable(shape):
    initial = tf.constant(0.1, shape = shape)
    return tf.Variable(initial)

def conv2d(x, W):
    return tf.nn.conv2d(x, W, strides = [1, 1, 1, 1], padding = "SAME")

def max_pool_2x2(x):
    return tf.nn.max_pool(x, ksize = [1, 2, 2, 1], strides = [1, 2, 2, 1], padding = "SAME")

def conv_layer(inputs, shape):
    W = weight_variable(shape)
    b = bias_variable([shape[3]])
    return tf.nn.relu(conv2d(inputs, W) + b)

def full_layer(inputs, size):
    in_size = int(inputs.get_shape()[1])
    W = weight_variable([in_size, size])
    b = bias_variable([size])
    return tf.matmul(inputs, W) + b
```

Implementation: Function Definition

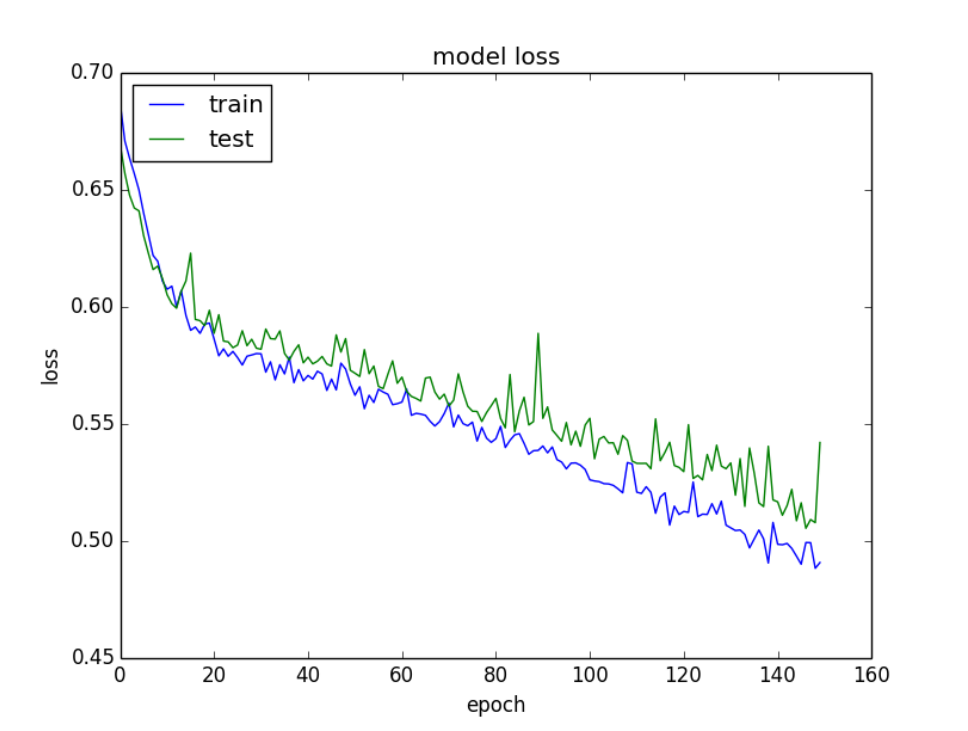
```
x = tf.placeholder(tf.float32, shape = [None, 128 * 128])
y_ = tf.placeholder(tf.float32, shape = [None, 5])

x_image = tf.reshape(x, [-1, 128, 128, 1])
conv1 = conv_layer(x_image, shape = [2, 2, 1, 16])
conv1_pool = max_pool_2x2(conv1)

conv2 = conv_layer(conv1_pool, shape = [2, 2, 16, 32])
conv2_pool = max_pool_2x2(conv2)

conv3 = conv_layer(conv2_pool, shape = [2, 2, 32, 32])
conv3_pool = max_pool_2x2(conv3)

conv3_flat = tf.reshape(conv3_pool, [-1, 16 * 16 * 32])
full_1 = tf.nn.relu(full_layer(conv3_flat, 1024))

keep_prob = tf.placeholder(tf.float32)
full1_drop = tf.nn.dropout(full_1, keep_prob = keep_prob)

y_conv = full_layer(full1_drop, 5)
```

Implementation: Model Training

```
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits = y_conv, labels = y_))
train_step = tf.train.AdamOptimizer(0.0001).minimize(cost)

correct_prediction = tf.equal(tf.argmax(y_conv, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
pred = y_conv

sess = tf.Session()
sess.run(tf.global_variables_initializer())
before_time = time.time()

for i in range(1001):
    randix = random.sample(range(len(trainimg_loaded)), 333)
    batch_xs = trainimg_loaded[randix]
    batch_ys = trainlabel_loaded[randix]

    if i % 100 == 0:
        train_accuracy = sess.run(accuracy, feed_dict = {x : batch_xs, y_ : batch_ys, keep_prob : 1.0})
        print("step {}, training accuracy {}".format(i, train_accuracy))
        spend_min = round((time.time() - before_time)/ 60)
        print("{}min spend".format(spend_min))
        before_time = time.time()

    sess.run(train_step, feed_dict = {x : batch_xs, y_ : batch_ys, keep_prob : 0.7})
    test_accuracy = sess.run(accuracy, feed_dict = {x : testimg_loaded, y_ : testlabel_loaded, keep_prob: 1.0})

print("test accuracy: {}".format(test_accuracy))
```

Implementation: Model Training

```
step 0, training accuracy 0.20720720291137695
0min spend
step 100, training accuracy 0.44744744896888733
1min spend
step 200, training accuracy 0.48948949575424194
1min spend
step 300, training accuracy 0.4834834933280945
1min spend
step 400, training accuracy 0.6036036014556885
1min spend
step 500, training accuracy 0.6276276111602783
1min spend
step 600, training accuracy 0.6516516804695129
1min spend
step 700, training accuracy 0.7297297120094299
1min spend
step 800, training accuracy 0.7627627849578857
1min spend
step 900, training accuracy 0.792792797088623
1min spend
step 1000, training accuracy 0.8258258104324341
1min spend
test accuracy: 0.7257353067398071
```



예시
〈https://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/〉

Implementation: Model Evaluate

```
from sklearn.metrics import confusion_matrix

cf = confusion_matrix(y_true = real, y_pred = pred)
print(cf)
print("Accuray is {}".format(round((1 - (len(false_index) / len(pred))) * 100, 2)))
```

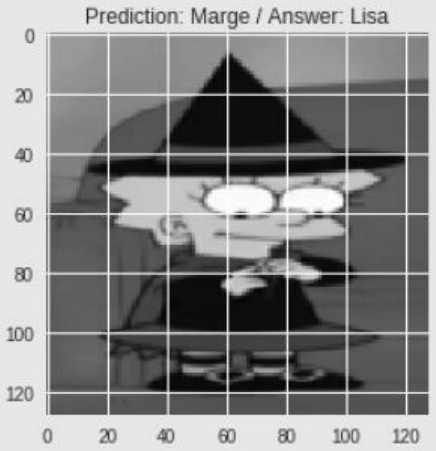
```
[[244  33  17  17   0]
 [ 24 417  17  11   0]
 [ 19  21 245   5   0]
 [ 36  27   8 191   0]
 [  2   1   3   1  21]]
Accuray is 82.21
```

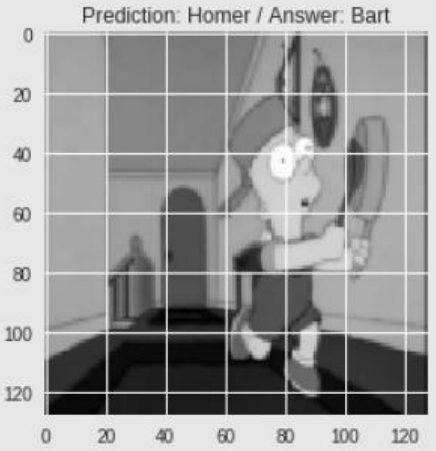
```
from sklearn.metrics import *

print(classification_report(real, pred, target_names = categories_loaded))
```

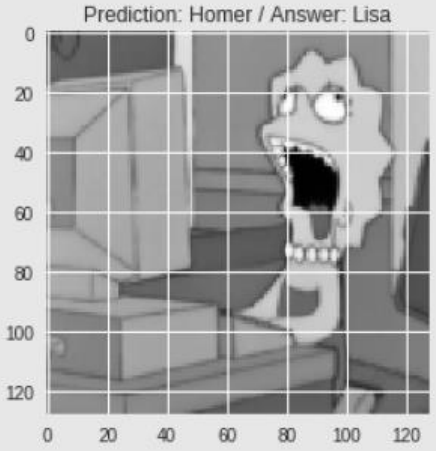
	precision	recall	f1-score	support
Bart	0.75	0.78	0.77	311
Homer	0.84	0.89	0.86	469
Marge	0.84	0.84	0.84	290
Lisa	0.85	0.73	0.78	262
Maggie	1.00	0.75	0.86	28
avg / total	0.82	0.82	0.82	1360

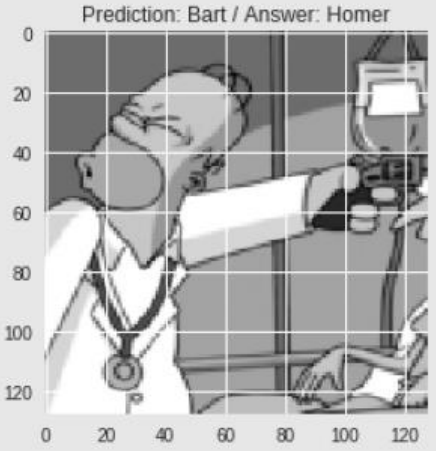
Implementation: Model Evaluate

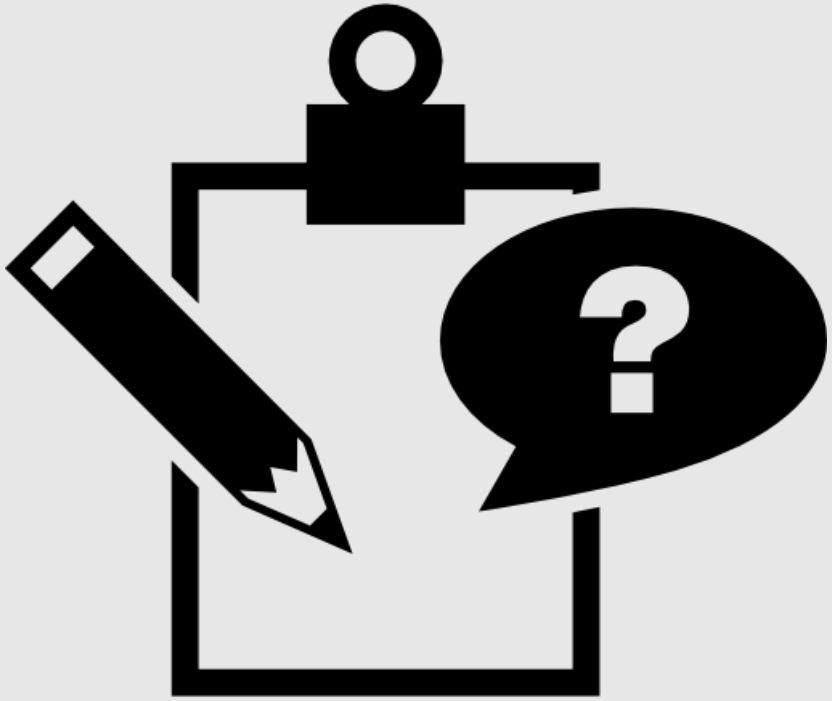
```
import random

figure_count = 5
randidx = random.sample(false_index, figure_count)
for i in randidx:
  currimg = np.reshape(testimg_loaded[i], (64, 64))
  plt.imshow(currimg, cmap = "gray")
  pred_a = categories_loaded[pred[i]]
  real_a = categories_loaded[real[i]]
  plt.title("Prediction: {} / Answer: {}".format(pred_a, real_a))
  plt.show()
```











Before End Class

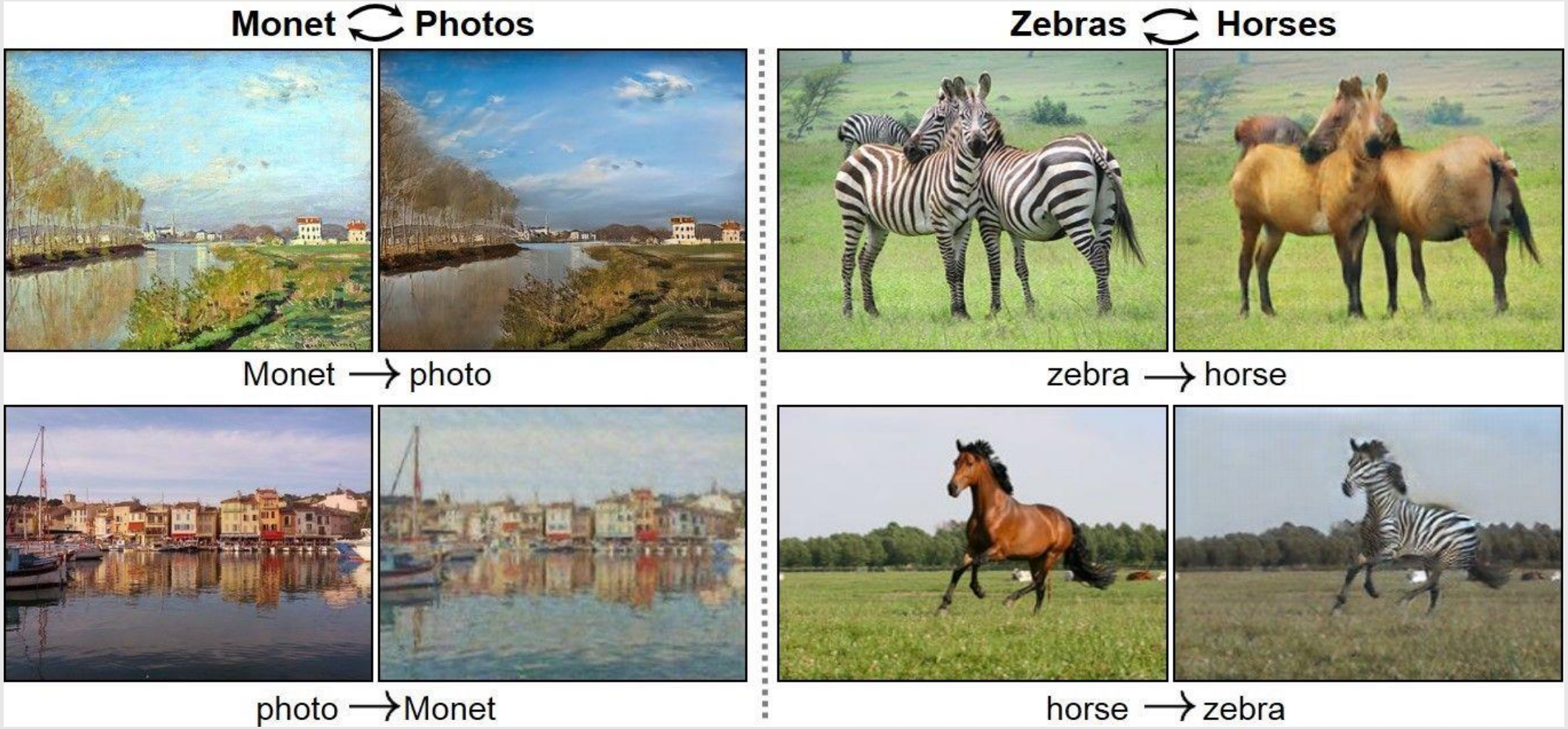
TED

〈https://www.ted.com/talks/blaise_aguera_y_arcas_how_computers_are_learning_to_be_creative#t-962754〉

Deep Dream

〈https://psmag.com/environment/googles-deep-dream-is-future-kitsch〉

GAN



〈http://cachestocaches.com/2017/12/favorite-deep-learning-2017/〉

AI Art

〈https://www.youtube.com/watch?v=HAfLCTRuh7U〉

Next Plan

11월 09일: 10장 (인공 신경망 소개)

11월 11일: 진교훈 생일

11월 15일: 11장 (심층 신경망 훈련)

11월 16일: 학술제


Ideas worth spreading
– TED Talks

고생하셨습니다.