ML_8

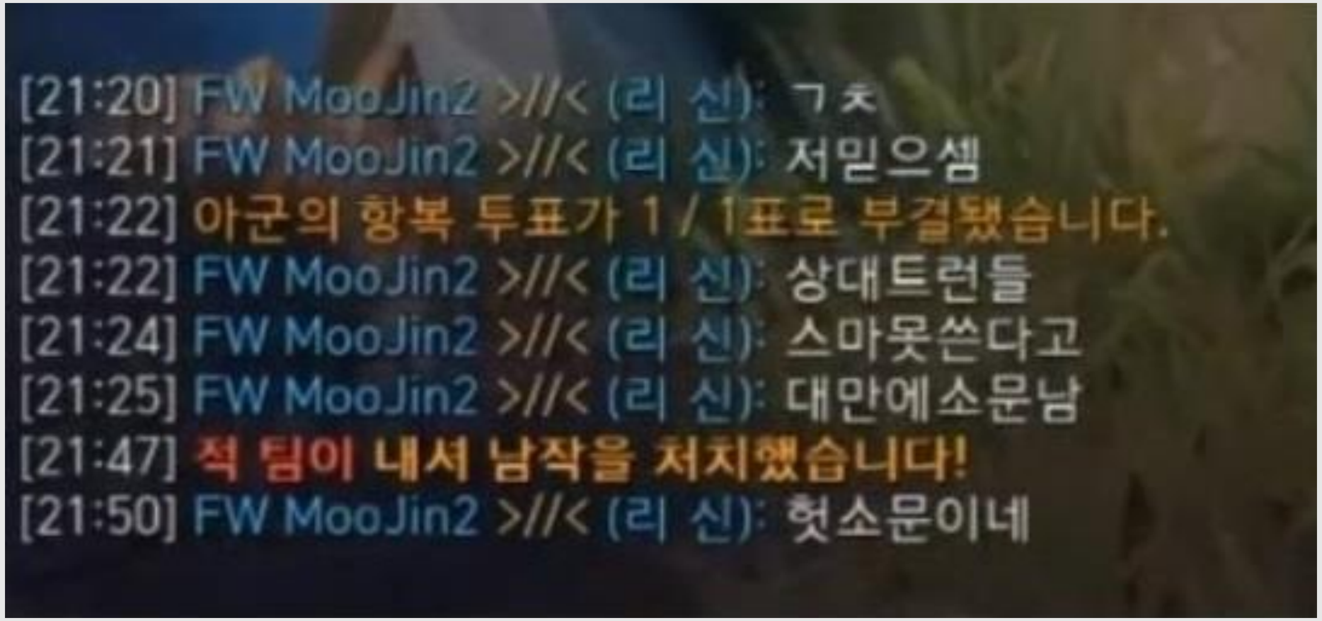
Team BMS

INDEX

ML Study
8 Week

Ice Breaking

Ice Breaking



〈http://www.inven.co.kr/board/lol/4625/2164143〉

Ice Breaking

<http://www.inven.co.kr/webzine/news/?news=150091&site=lol>

01 I. Ice Breaking

Before Start

테리의 딥러닝 토크님이 게시물을 공유했습니다.

3월 26일 ·

"딥러닝을 일찍 공부하는 게 좋지 않은 이유는 마치 이런 것이다. 2차 방정식을 공부하는데 함수 그래프를 그려보며 문제를 이해하기 이전에 근의 공식 먼저 외우는 그런 느낌?"

"'이거 딥러닝으로 꼭 풀어야 해'라고 주장할 수 있는 근거를 찾을 실력이 있는가? 그것도 보통 더 비싼 방법을 말이다."

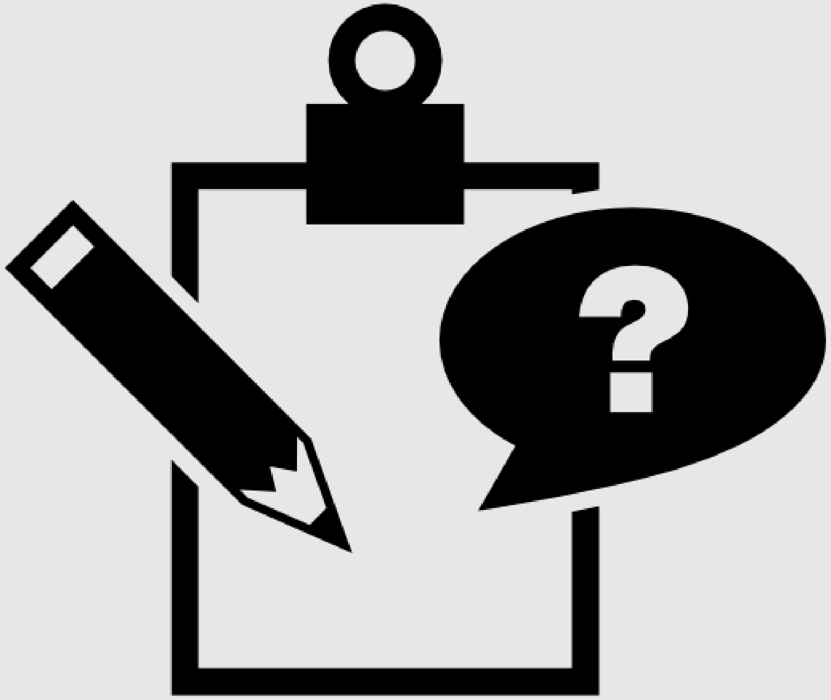
"그럼 딥러닝은 언제 어떻게 공부해야 하는가? 단순하게 답하기는 쉽지 않지만, 딥러닝을 비판적인 관점으로 바라볼 실력이 될 때 공부하는 것이 좋다. 각 단계를 공부하며 '이거 꼭 이렇게 해야 해?'라고 스스로 질문할 수 있어야 한다. 그리고 그렇게 하는 이유 또는 한계점을 직관적으로 이해하고 논리적으로 설명할 수 있어야 한다."

<https://www.facebook.com/deeplearningtalk/posts/600510300301265?__xts__[0]=68.ARCKMC9ayay7vk-OR7CdOEa7yhJZfy2lWW4VyJuTaoOYhF_OMLQOz87-iBLVthoWvC8_4dcWOIPH5iTLf59tNi3lnzWwLLoYhaB6SzyfelFlgZD4KZ-WqBE5qpr-o-rP5LxTQGqiiygpH00D9VxSrKjWlJZEAqjEBzGSxoPem_TaRtL2BNDqqwUgJ_4SpGt_EcFz48glecrtEFF6flp1ia9DGNYidyM_xeoqBBBnp9JTFw5Uv4EbRH92v5wMFZWee_AQNg7q0pc8XwWAJaJSVKOeNulBSMWbjzHu9j-Hn9La1wFXqsaW5zH4frxq3dQvekr4POI40vjhWYzbakw&__tn__=-R>

Before Start

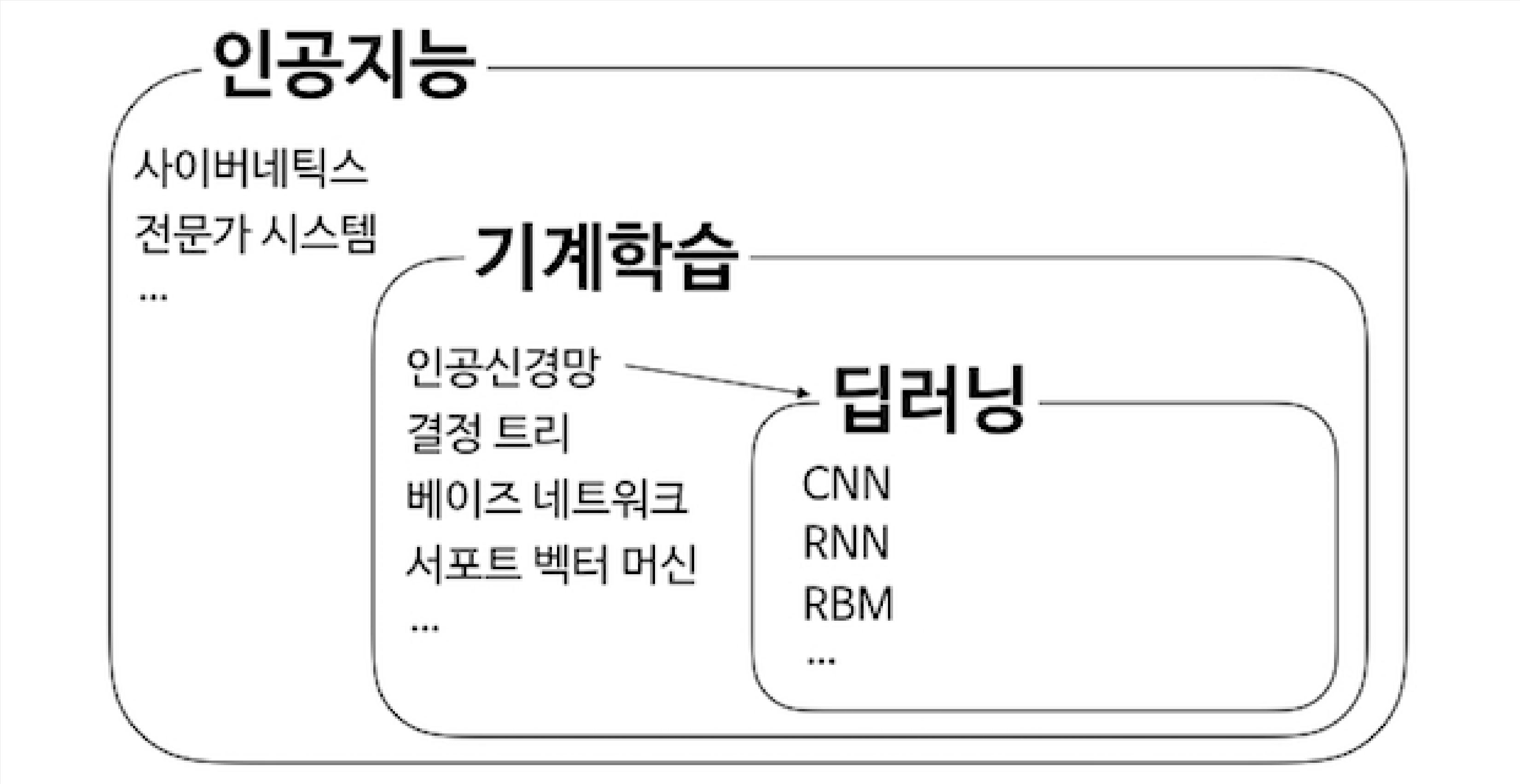
〈http://m.todayhumor.co.kr/view.php?table=humordata&no=1522219〉

Before Start



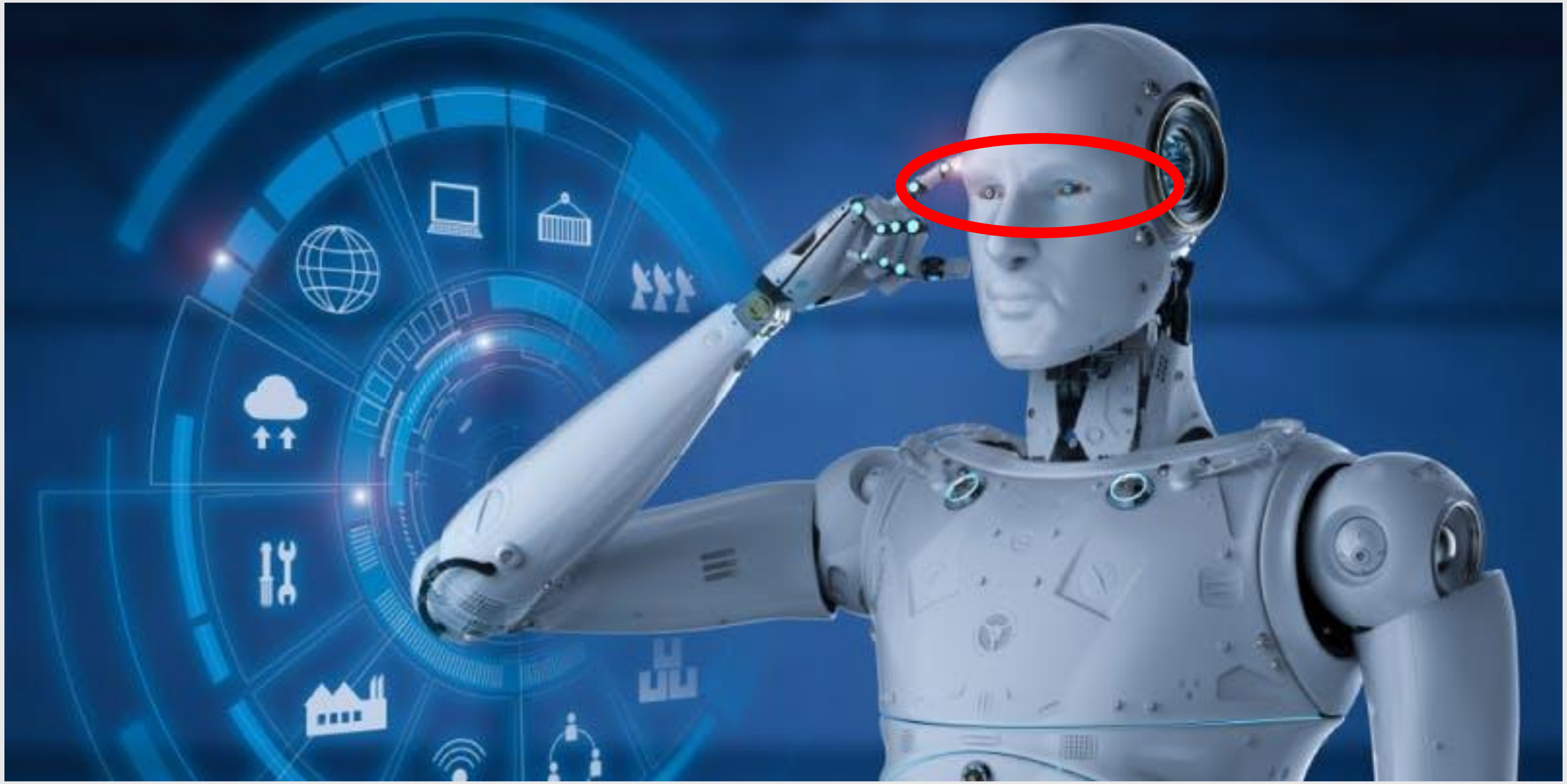
Convolutional Neural Network

Artificial Intelligence



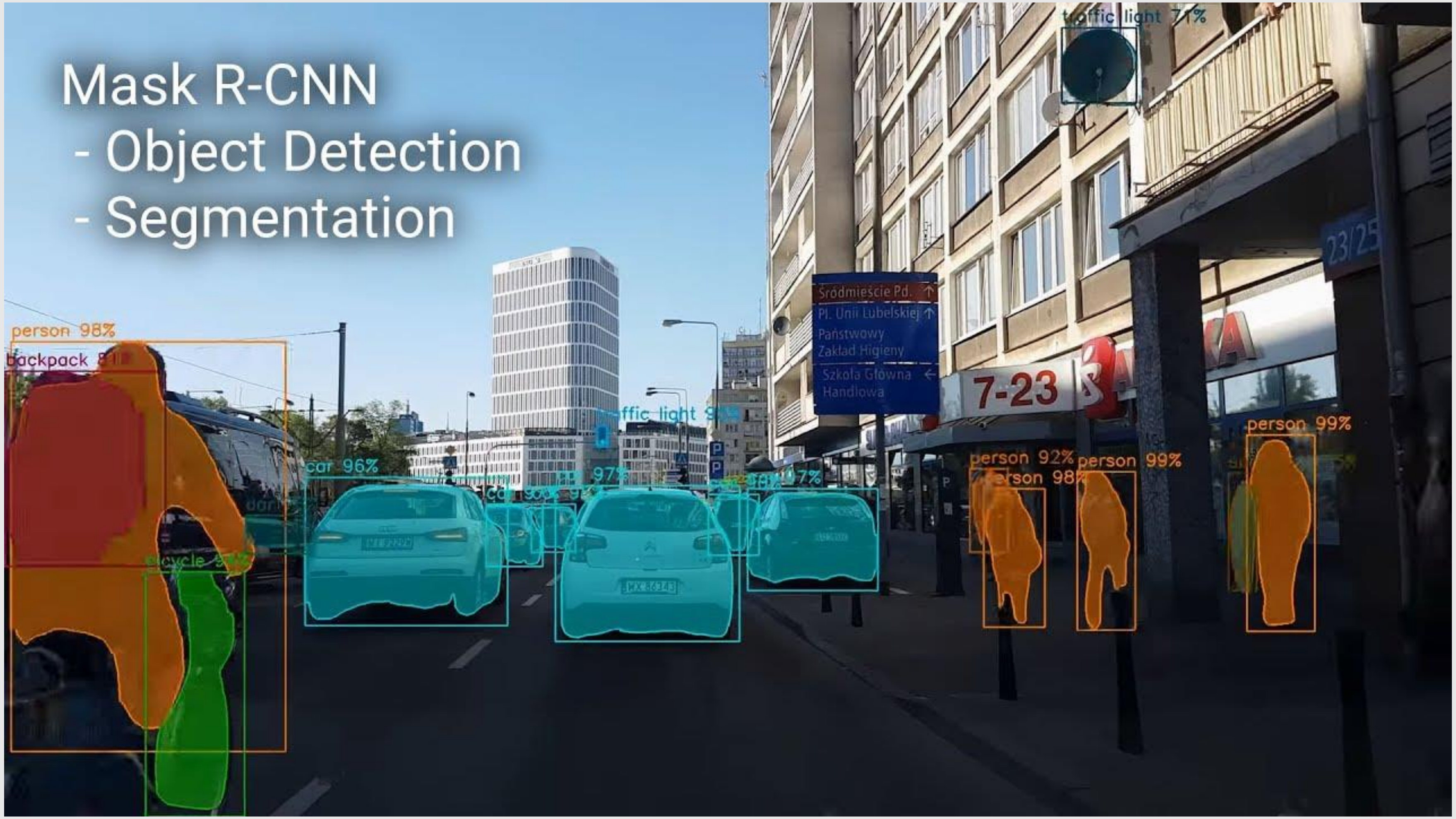
<http://www.yoonsupchoi.com/2017/08/08/ai-medicine-4/>

Artificial Intelligence's eye



<https://yourstory.com/2018/04/online-courses-to-robots-as-concierge-ai-all-the-way/>

Artificial Intelligence's eye



⟨https://www.youtube.com/watch?v=OOT3UIXZztE⟩

CNN History

1959

J. Physiol. (1959) 147, 226–238

SINGLE UNIT ACTIVITY IN STRIATE CORTEX OF UNRESTRAINED CATS

By D. H. HUBEL*
From the Department of Neurophysiology, Walter Reed Army Institute of Research, Walter Reed Army Medical Center, Washington 12, D.C., U.S.A.

(*Received* 15 *December* 1958)

A beginning has recently been made in recording single neurone activity from animals with chronically implanted electrodes (Hubel, 1957*a*; Gusel'nikov, 1957; Ricci, Doane & Jasper, 1957; Strumwasser, 1958). These methods eliminate anaesthetics, paralysing drugs, brain-stem lesions, and other acute experimental procedures. They make it possible to record electrical events in the higher central nervous system with the animal in a normal state, and to correlate these electrical events with such variables as waking state, attention, learning, and motor activity.

The present paper describes a method for unit recording from the cortex of unanaesthetized, unrestrained cats, and presents some observations from the striate cortex. The objectives have been (1) to observe maintained unit activity under various conditions such as sleep and wakefulness, and (2) to find for each unit the natural stimuli which most effectively influence firing. Of 400 units observed, some 200 are presented here because of their common characteristics. Since there is reason to believe that the remaining 200 units were afferent fibres from the lateral geniculate nucleus, these will be described in a separate paper. A preliminary account of some of this work has been given elsewhere (Hubel, 1958).

METHODS

Unit recordings in unrestrained animals were made with micro-electrodes held in a positioner which was anchored rigidly to the skull during recordings, and removed between recordings. A rigidly implanted peg adapted from the design of Ricci *et al.* (1957) held the micropositioner at the time of recording (Text-fig. 1). The peg was made of the plastic Kel-F (fluorocarbon polymer made by Minnesota Mining and Manufacturing Company, St Paul, Minn.). It was hollow and was

* Present address: Neurophysiological Laboratory, Department of Pharmacology, Harvard Medical School, Boston 15, Massachusetts, U.S.A.

1968

J. Physiol. (1968), **195**, *pp.* 215–243
With 3 plates and 14 text-figures
Printed in Great Britain

RECEPTIVE FIELDS AND FUNCTIONAL ARCHITECTURE OF MONKEY STRIATE CORTEX

By D. H. HUBEL and T. N. WIESEL
From the Department of Physiology, Harvard Medical School, Boston, Mass., U.S.A.

(*Received 6 October* 1967)

SUMMARY

1. The striate cortex was studied in lightly anaesthetized macaque and spider monkeys by recording extracellularly from single units and stimulating the retinas with spots or patterns of light. Most cells can be categorized as simple, complex, or hypercomplex, with response properties very similar to those previously described in the cat. On the average, however, receptive fields are smaller, and there is a greater sensitivity to changes in stimulus orientation. A small proportion of the cells are colour coded.
2. Evidence is presented for at least two independent systems of columns extending vertically from surface to white matter. Columns of the first type contain cells with common receptive-field orientations. They are similar to the orientation columns described in the cat, but are probably smaller in cross-sectional area. In the second system cells are aggregated into columns according to eye preference. The ocular dominance columns are larger than the orientation columns, and the two sets of boundaries seem to be independent.
3. There is a tendency for cells to be grouped according to symmetry of responses to movement; in some regions the cells respond equally well to the two opposite directions of movement of a line, but other regions contain a mixture of cells favouring one direction and cells favouring the other.
4. A horizontal organization corresponding to the cortical layering can also be discerned. The upper layers (II and the upper two-thirds of III) contain complex and hypercomplex cells, but simple cells are virtually absent. The cells are mostly binocularly driven. Simple cells are found deep in layer III, and in IV A and IV B. In layer IV B they form a large proportion of the population, whereas complex cells are rare. In layers IV A and IV B one finds units lacking orientation specificity; it is not clear whether these are cell bodies or axons of geniculate cells. In layer IV most cells are driven by one eye only; this layer consists of a mosaic with

1959

Biol. Cybernetics 36, 193–202 (1980)

Biological Cybernetics
© by Springer-Verlag 1980

Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position

Kunihiko Fukushima
NHK Broadcasting Science Research Laboratories, Kinuta, Setagaya, Tokyo, Japan

Abstract. A neural network model for a mechanism of visual pattern recognition is proposed in this paper. The network is self-organized by "learning without a teacher", and acquires an ability to recognize stimulus patterns based on the geometrical similarity (Gestalt) of their shapes without affected by their positions. This network is given a nickname "neocognitron". After completion of self-organization, the network has a structure similar to the hierarchy model of the visual nervous system proposed by Hubel and Wiesel. The network consists of an input layer (photoreceptor array) followed by a cascade connection of a number of modular structures, each of which is composed of two layers of cells connected in a cascade. The first layer of each module consists of "S-cells", which show characteristics similar to simple cells or lower order hypercomplex cells, and the second layer consists of "C-cells" similar to complex cells or higher order hypercomplex cells. The afferent synapses to each S-cell have plasticity and are modifiable. The network has an ability of unsupervised learning: We do not need any "teacher" during the process of self-organization, and it is only needed to present a set of stimulus patterns repeatedly to the input layer of the network. The network has been simulated on a digital computer. After repetitive presentation of a set of stimulus patterns, each stimulus pattern has become to elicit an output only from one of the C-cells of the last layer, and conversely, this C-cell has become selectively responsive only to that stimulus pattern. That is, none of the C-cells of the last layer responds to more than one stimulus pattern. The response of the C-cells of the last layer is not affected by the pattern's position at all. Neither is it affected by a small change in shape nor in size of the stimulus pattern.

1. Introduction

The mechanism of pattern recognition in the brain is little known, and it seems to be almost impossible to reveal it only by conventional physiological experiments. So, we take a slightly different approach to this problem. If we could make a neural network model which has the same capability for pattern recognition as a human being, it would give us a powerful clue to the understanding of the neural mechanism in the brain. In this paper, we discuss how to synthesize a neural network model in order to endow it an ability of pattern recognition like a human being.

Several models were proposed with this intention (Rosenblatt, 1962; Kabrisky, 1966; Giebel, 1971; Fukushima, 1975). The response of most of these models, however, was severely affected by the shift in position and/or by the distortion in shape of the input patterns. Hence, their ability for pattern recognition was not so high.

In this paper, we propose an improved neural network model. The structure of this network has been suggested by that of the visual nervous system of the vertebrate. This network is self-organized by "learning without a teacher", and acquires an ability to recognize stimulus patterns based on the geometrical similarity (Gestalt) of their shapes without affected by their position nor by small distortion of their shapes.

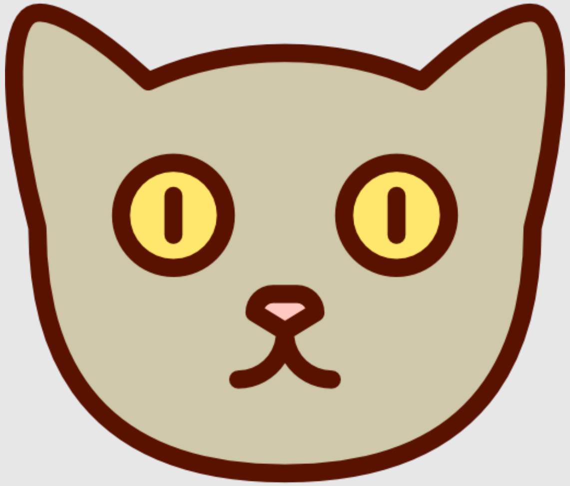
This network is given a nickname "neocognitron"[1], because it is a further extention of the "cognitron", which also is a self-organizing multilayered neural network model proposed by the author before (Fukushima, 1975). Incidentally, the conventional cognitron also had an ability to recognize patterns, but its response was dependent upon the position of the stimulus patterns. That is, the same patterns which were presented at different positions were taken as different patterns by the conventional cognitron. In the neocognitron proposed here, however, the response of the network is little affected by the position of the stimulus patterns.

1 Preliminary report of the neocognitron already appeared elsewhere (Fukushima, 1979a, b)

0340-1200/80/0036/0193/$02.00

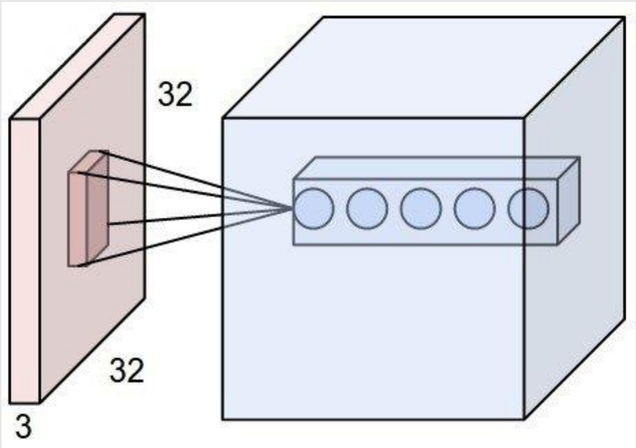
CNN History

1959



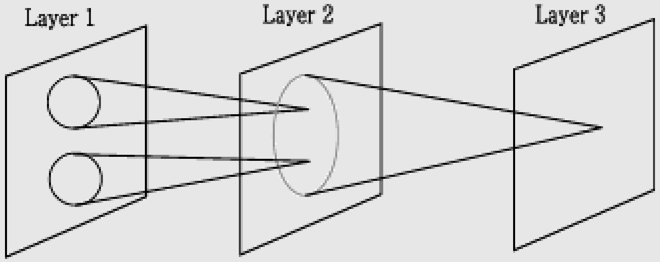
고양이 실험

1968



Local Receptive Field

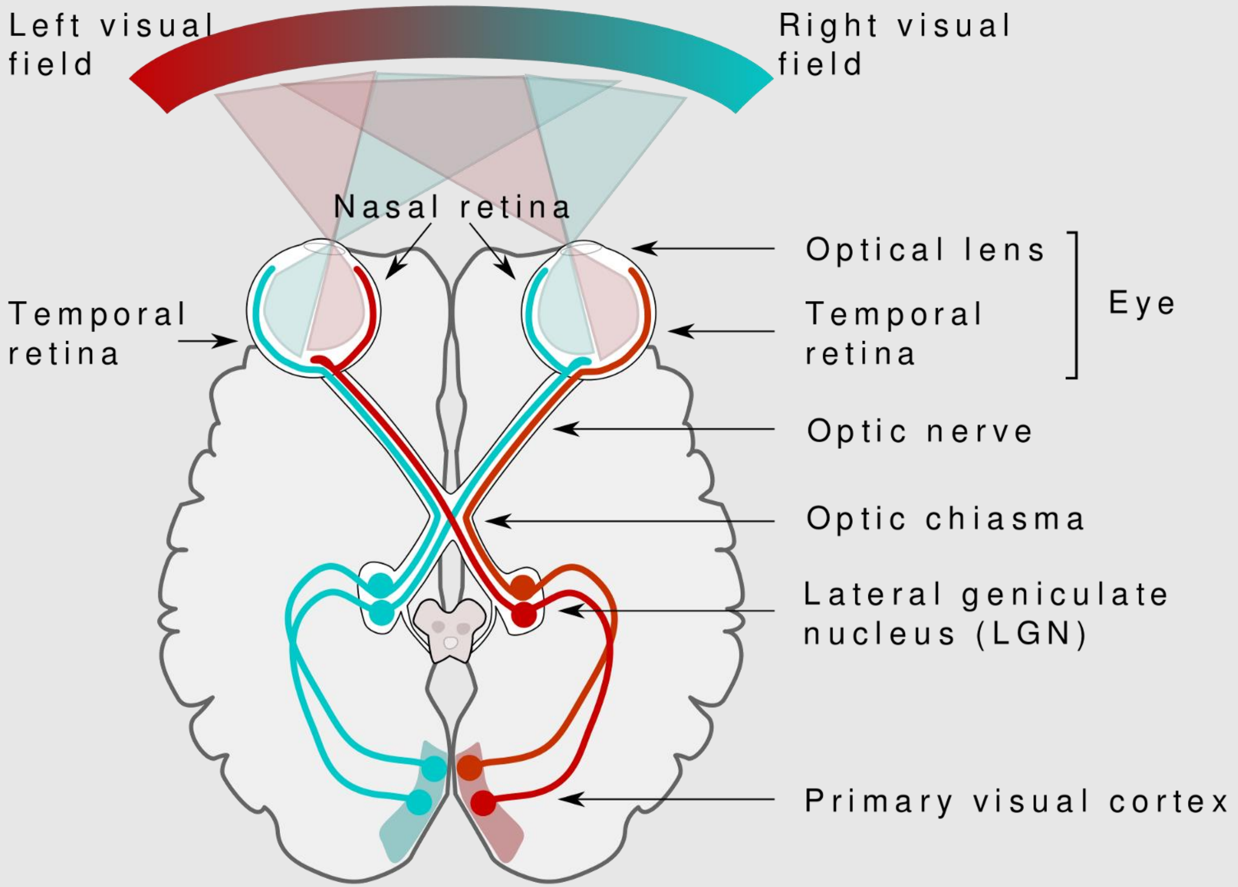
1980



<http://www.aistudy.co.kr/neural/cognitron_kim.htm>

Neocognition

Intuitive guide to CNN



<https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050>

Yann LeCun et al, 1998

Object Recognition with Gradient-Based Learning

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio

AT&T Shannon Lab, 100 Schulz Drive, Red Bank NJ 07701, USA,
yann@research.att.com
http://www.research.att.com/~yann

Abstract. Finding an appropriate set of features is an essential problem in the design of shape recognition systems. This paper attempts to show that for recognizing simple objects with high shape variability such as handwritten characters, it is possible, and even advantageous, to feed the system directly with minimally processed images and to rely on learning to extract the right set of features. Convolutional Neural Networks are shown to be particularly well suited to this task. We also show that these networks can be used to recognize multiple objects without requiring explicit segmentation of the objects from their surrounding. The second part of the paper presents the Graph Transformer Network model which extends the applicability of gradient-based learning to systems that use graphs to represents features, objects, and their combinations.

1 Learning the Right Features

The most commonly accepted model of pattern recognition, is composed of a *segmenter* whose role is to extract objects of interest from their background, a hand-crafted *feature extractor* that gathers relevant information from the input and eliminates irrelevant variabilities, and a *classifier* which categorizes the resulting feature representations (generally vectors or strings of symbols) into categories. There are three major methods for classification: *template matching* matches the feature representation to a set of class templates; *generative methods* use a probability density model for each class, and pick the class with the highest likelihood of generating the feature representation; *discriminative models* compute a discriminant function that directly produces a score for each class. Generative and discriminative models are often estimated (learned) from training samples. In all of these approaches, the overall performance of the system is largely determined by the quality of the segmenter and the feature extractor.

Because they are hand-crafted, the segmenter and feature extractor often rely on simplifying assumptions about the input data and can rarely take into account all the variability of the real world. An ideal solution to this problem is to feed the entire system with minimally processed inputs (e.g. "raw" pixel images), and train it from data so as to minimize an overall loss function (which maximizes a given performance measure). Keeping the preprocessing to a minimum ensures that no unrealistic assumption is made about the data. Unfortunately, that also

〈Yann LeCun et al, 1999〉

- 1998년 Yann LeCun의 논문

➔Object Recognition
➔CNN의 시작

Yann LeCun et al, 1998

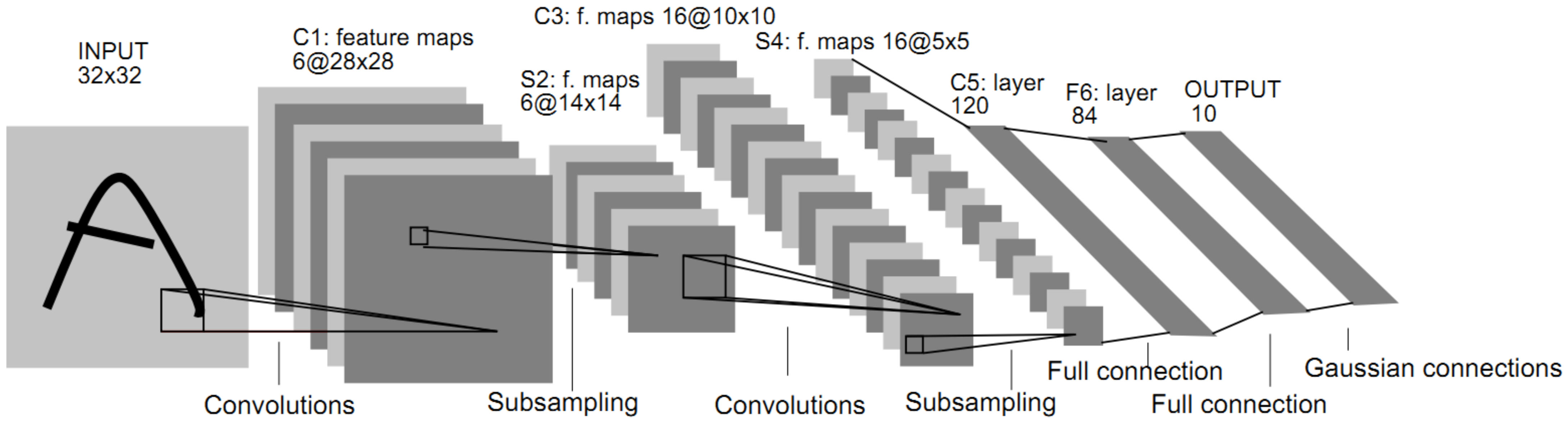
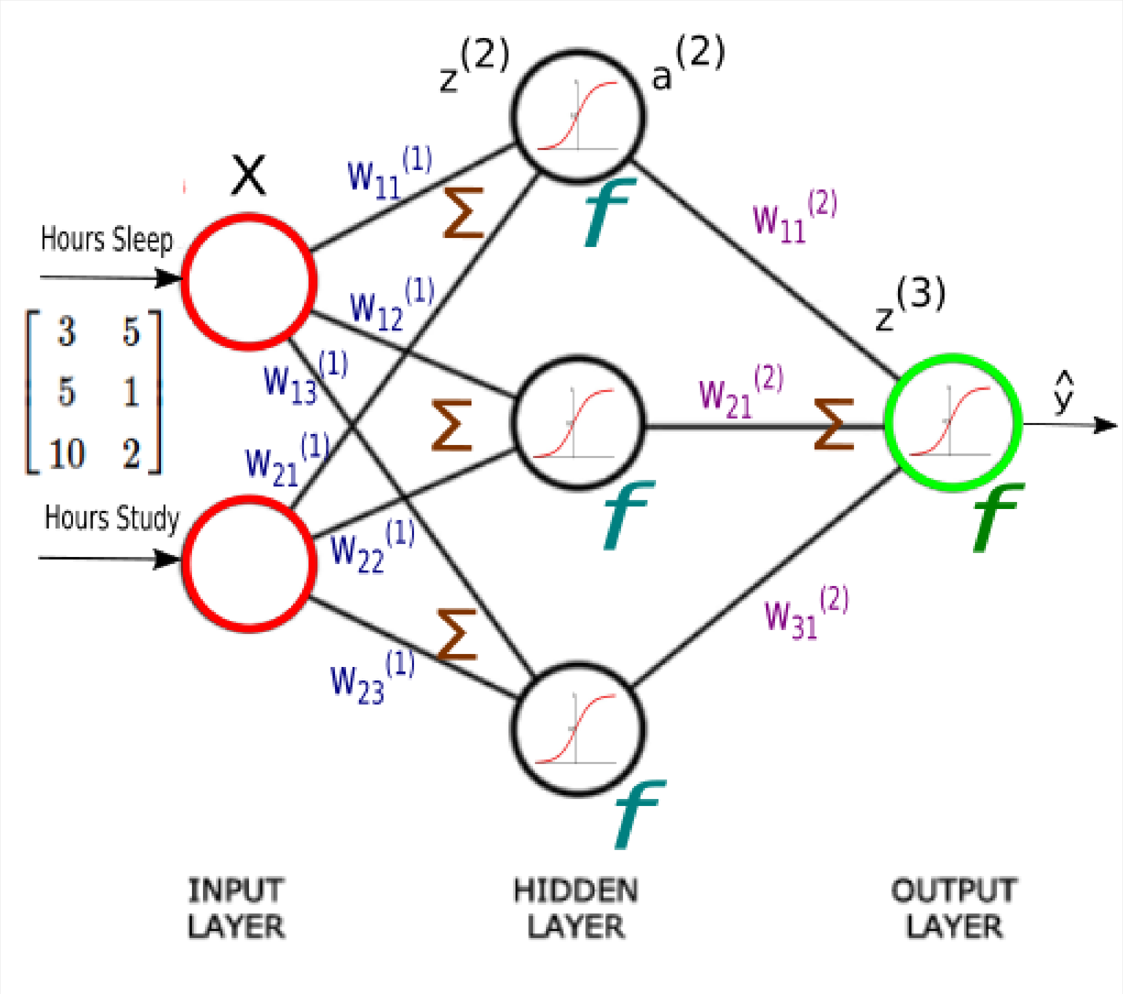


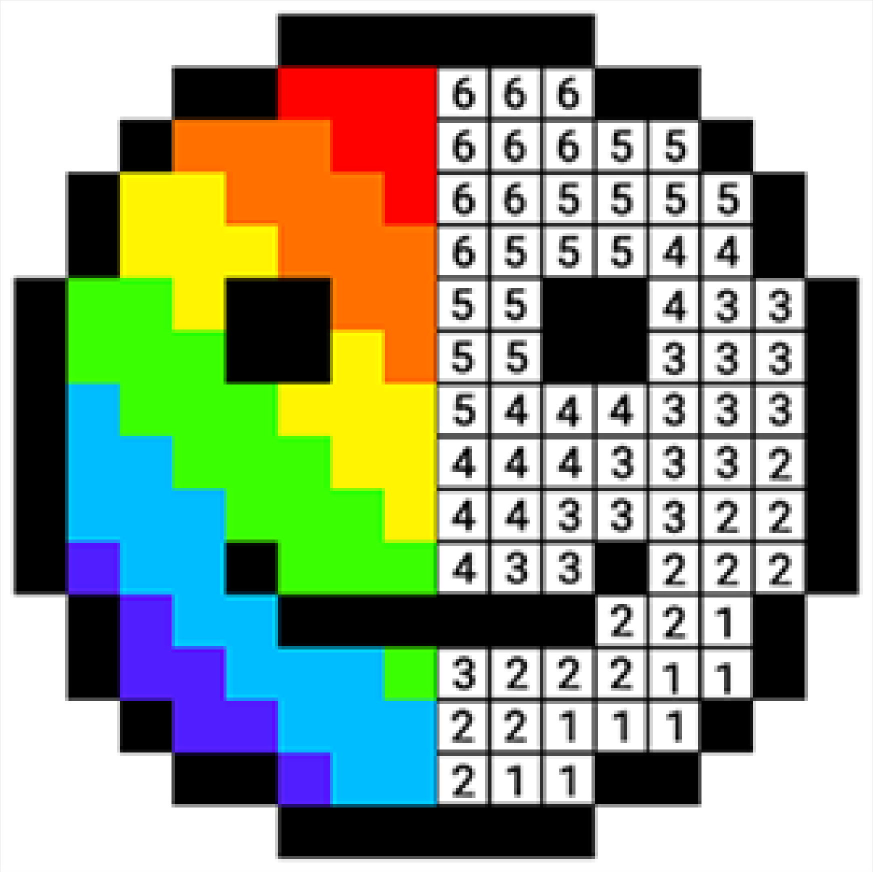
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Artificial Neural Network with Pixel



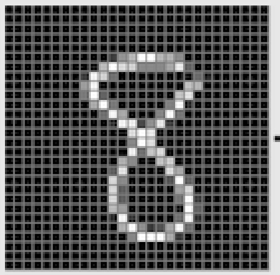
<https://www.bogotobogo.com/python/scikit-learn/Artificial-Neural-Network-ANN-1-Introduction.php>

Artificial Neural Network with Pixel



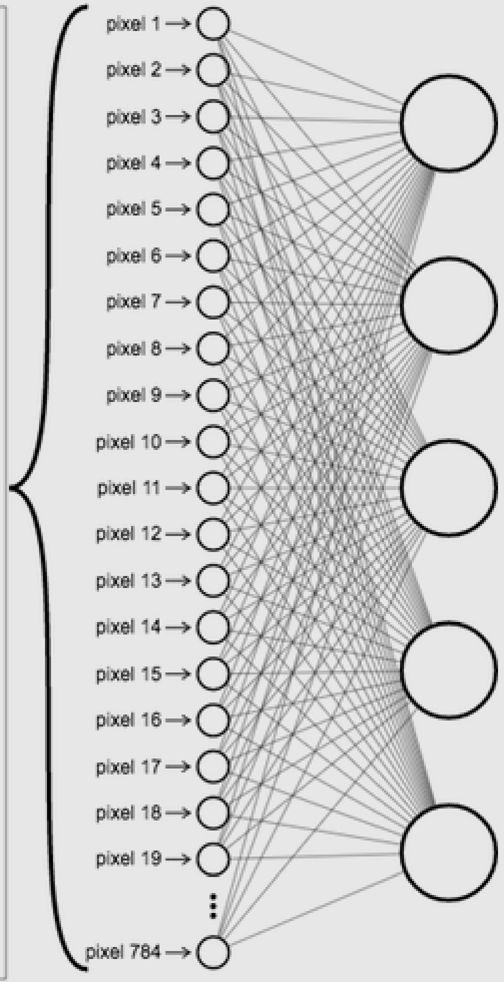
〈https://www.microsoft.com/en-us/p/pixel-art-color-by-number-sandbox/9n635qpcd7xh〉

Artificial Neural Network with Pixel



28 x 28
784 pixels

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	4	62	146	182	254	254	181	176	139	15	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	34	186	253	217	208	136	136	136	166	232	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	61	242	208	111	3	0	0	0	0	0	18	32	107	43	0	0	0	0	0	0
0	0	0	0	0	0	0	0	156	242	23	0	0	0	0	0	0	0	13	191	181	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	121	255	98	3	0	0	0	0	0	8	194	225	12	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	169	253	120	3	0	0	0	0	128	247	51	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	3	111	244	169	19	0	14	131	249	117	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	59	241	235	72	142	229	66	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	25	218	254	231	36	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	133	253	221	33	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	19	237	111	196	217	19	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	138	0	23	193	204	18	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	96	224	0	0	0	25	218	169	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	138	0	0	0	0	86	253	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	3	162	214	11	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	0	118	253	68	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	185	157	0	0	0	0	0	40	254	98	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	50	244	61	0	0	0	0	112	244	58	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	251	142	59	83	167	244	111	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	6	133	253	253	253	169	61	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



⟨https://ml4a.github.io/ml4a/ko/neural_networks/⟩

Random Forest with Pixel

```
rfc.fit(X_train, y_train)
```

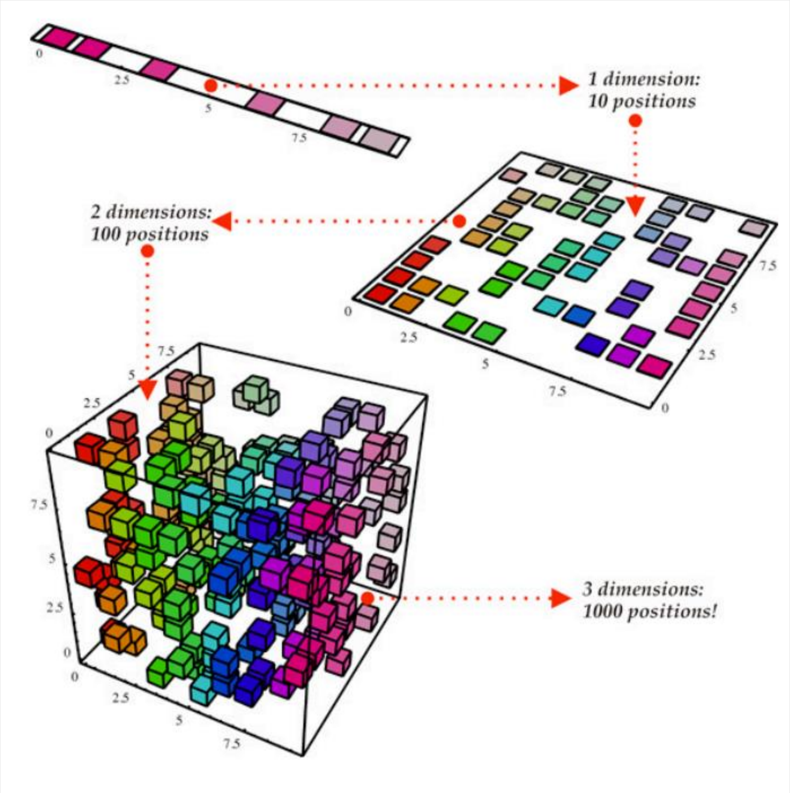
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=-1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

```
rfc.score(X_test, y_test)
```

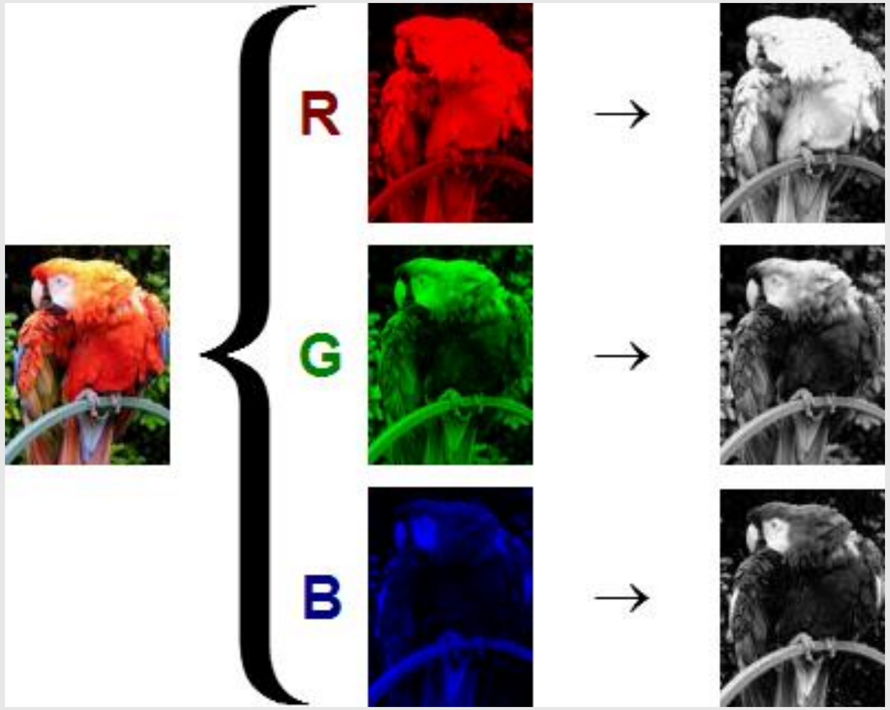
```
0.93885714285714283
```

<https://www.kaggle.com/atorin/mnist-digit-recognition-with-random-forests>

Pixel Based Modeling's Problem



〈https://towardsdatascience.com/that-cursing-dimensionality-ac317fb0fdcc〉



〈https://ko.wikipedia.org/wiki/%ED%8C%8C%EC%9D%BC:RGB_channels_separation.png〉

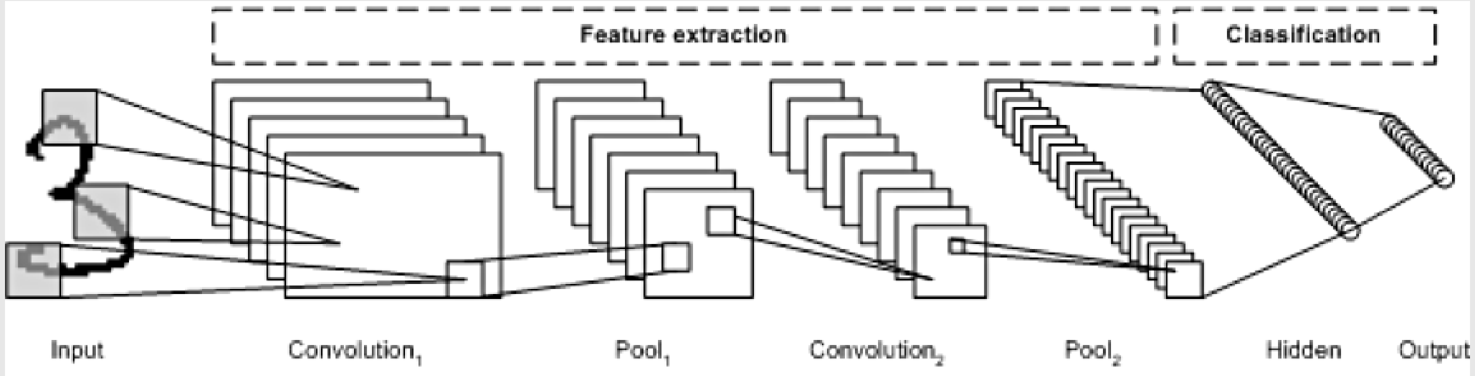
Pixel Based Modeling’s Problem

그림의 형태로 볼 수 없다.

⟨https://ko.wikipedia.org/wiki/%ED%95%B4%EB%B0%94%EB%9D%BC%EA%B8%B0_(%EA%B3%A0%ED%9D%90)⟩

CNN’s Benefit

- 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, 일반 인공 신경망에 비해 학습 파라미터가 매우 적음



<http://taewan.kim/post/cnn/>

CNN's Benefit

- **Convolution (합성곱)**
- Channel
- Filter
- **Stride**
- Padding
- Feature Map
- Activation Map
- **Pooling**

Convolution

In mathematics (and, in particular, functional analysis) convolution is a mathematical operation on two functions (f and g) to produce a third function that expresses how the shape of one is modified by the other. The term convolution refers to both the result function and to the process of computing it. Convolution is similar to cross-correlation. For discrete, real-valued functions, they differ only in a time reversal in one of the functions. For continuous functions, the cross-correlation operator is the adjoint of the convolution operator.

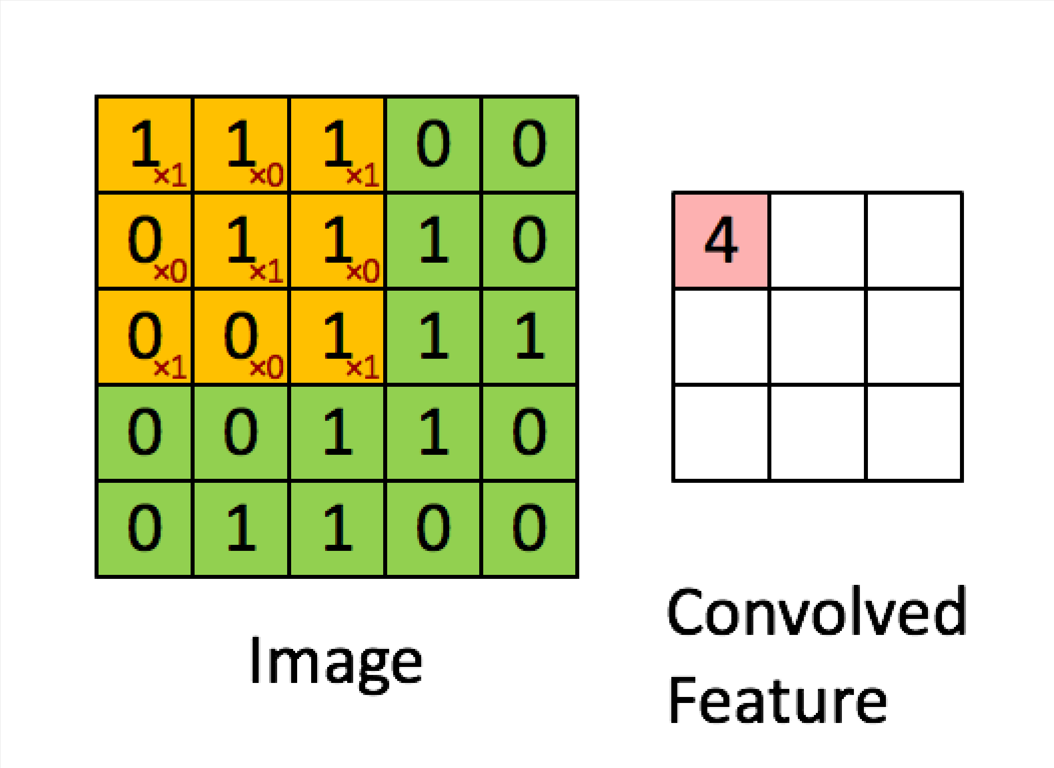
〈Wikipedia〉

합성곱(合成-, convolution, 콘벌루션)은 하나의 함수와 또 다른 함수를 반전 이동한 값을 곱한 다음, 구간에 대해 적분하여 새로운 함수를 구하는 수학 연산자이다.

〈위키백과〉

???

Convolution



Gif파일입니다.
<http://taewan.kim/post/cnn/>

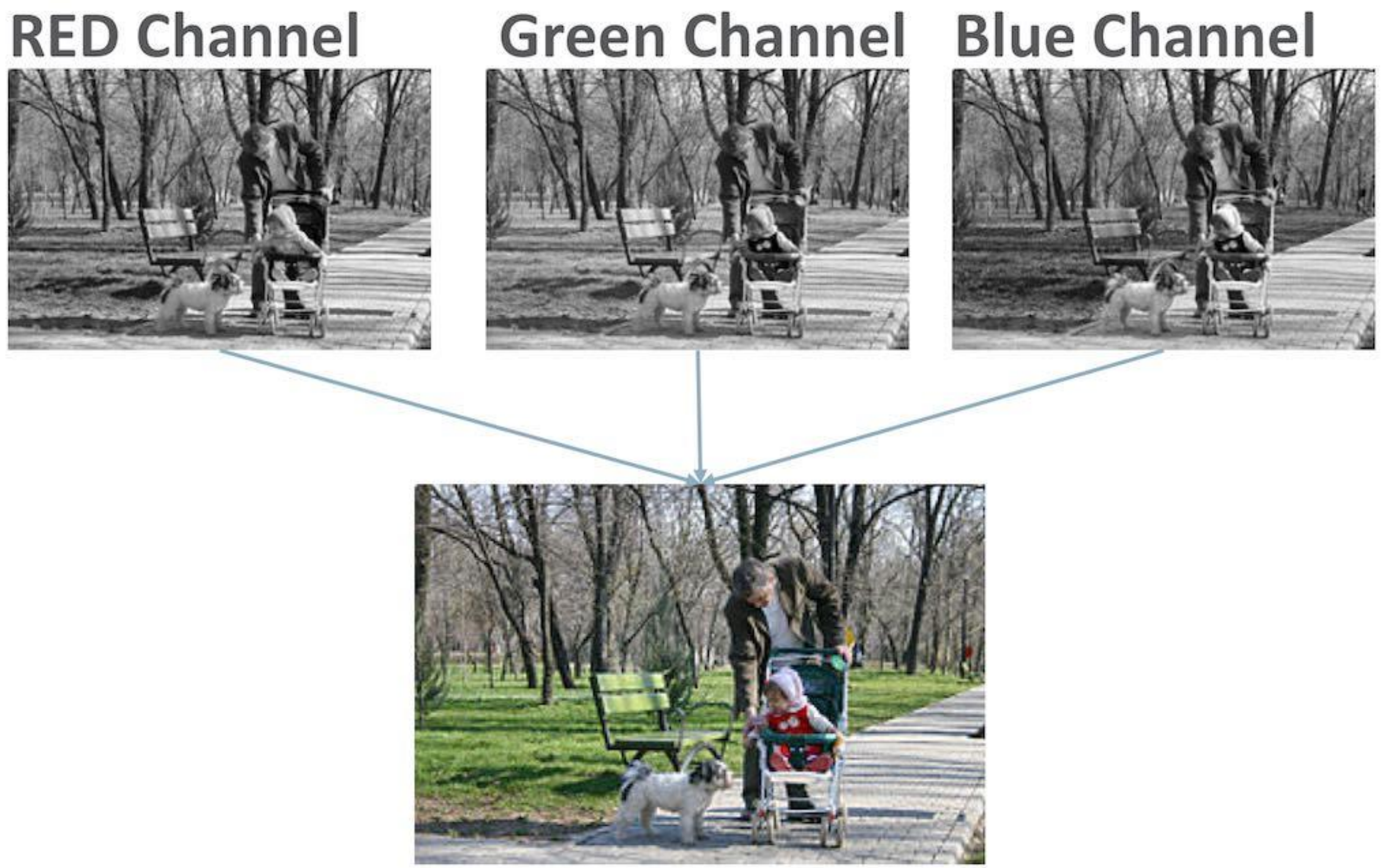
Convolution

이 신경망은 입력의 특정 위치의 특정 패턴에 대해 반응하는 (activate) 필터를 학습한다. 이런 액티베이션 맵 (activation map)을 깊이 (depth) 차원을 따라 쌓은 것이 곧 출력 볼륨이 된다. 그러므로 출력 볼륨의 각 요소들은 입력의 작은 영역만을 취급하고, 같은 액티베이션 맵 내의 뉴런들은 같은 모수들을 공유한다 (같은 필터를 적용한 결과이므로).

〈http://taewan.kim/post/cnn/〉

➔ Local Receptive Field (Local Connectivity)
[FC는 비효율적]

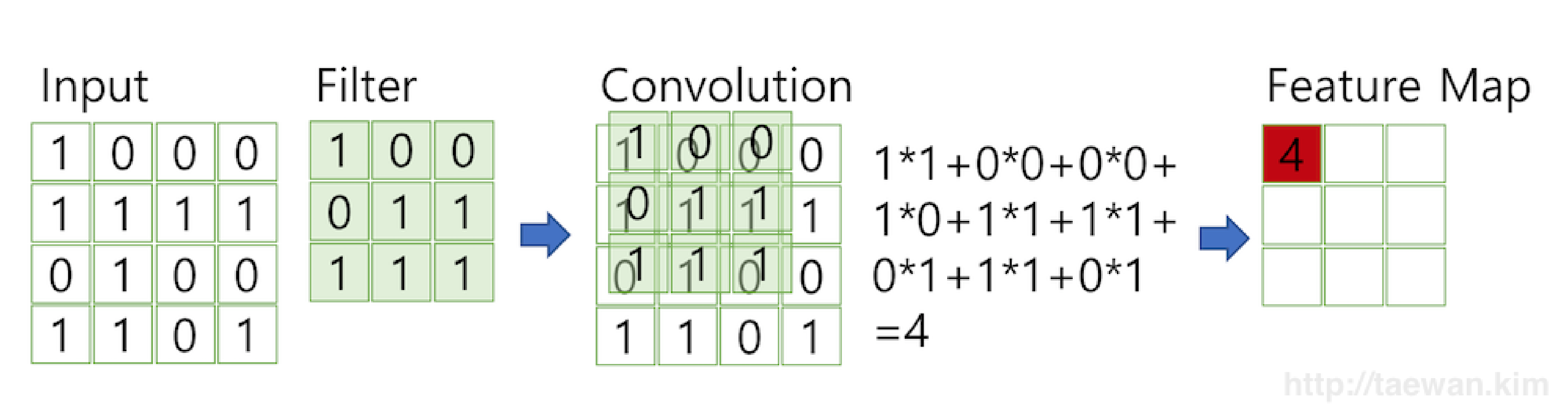
Channel



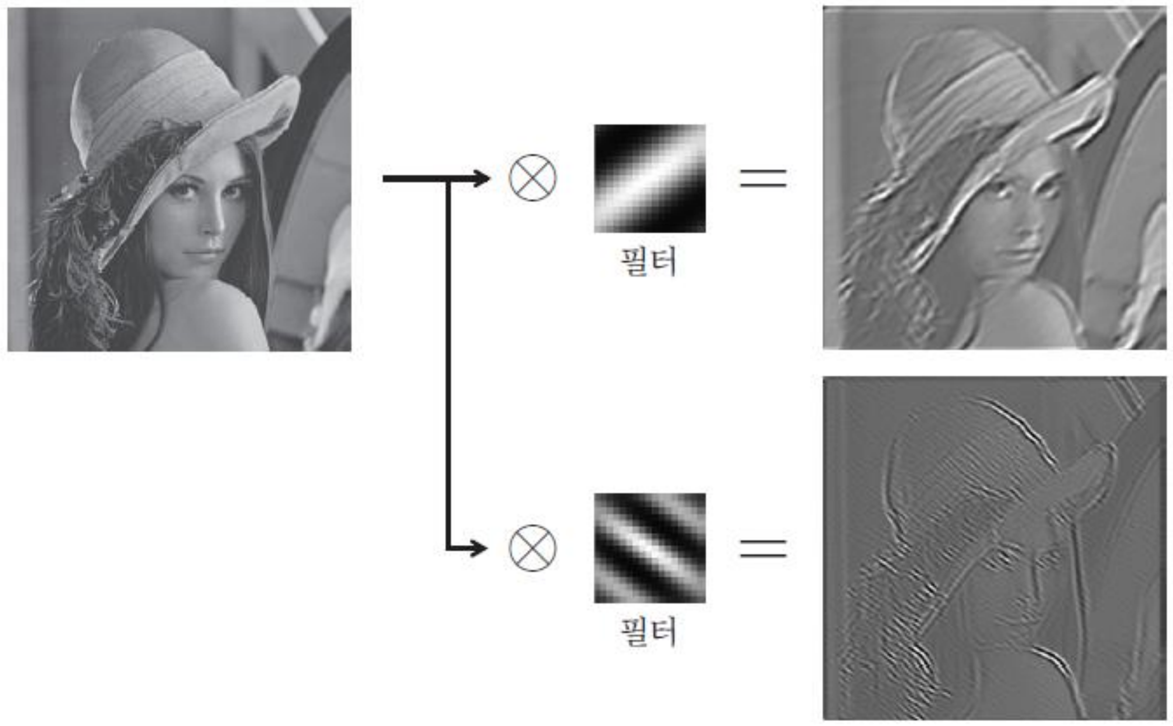
이미지 출처: https://en.wikipedia.org/wiki/Channel_(digital_image)

Filter

- 이미지의 특징을 찾아내기 위한 공용 파라미터
- CNN에서는 Filter와 Kernel은 같은 말
- CNN에서 학습의 대상은 필터 파라미터
- 지정된 간격으로 이동하면서 전체 입력 데이터와 합성곱하여 Feature Map생성



Filter

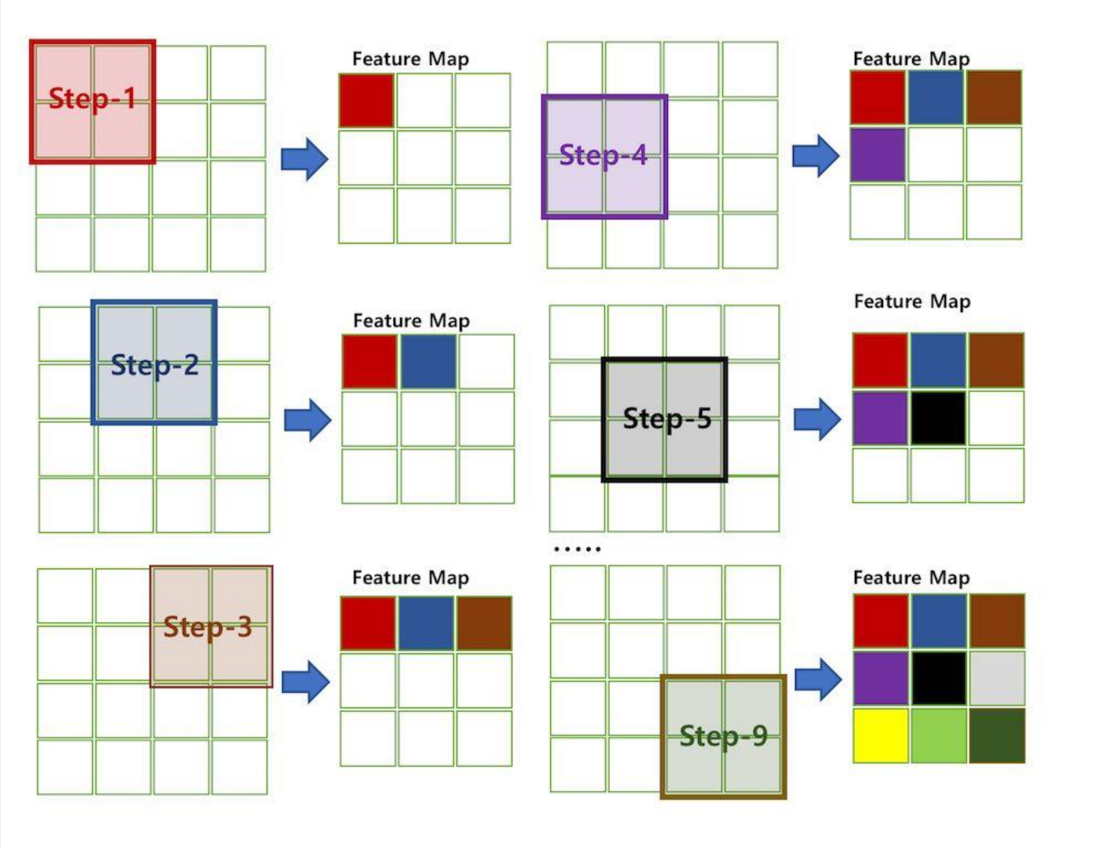


필터를 합성곱한 결과의 예. 512 × 512 크기의 이미지에 17 × 17 크기의 필터 두 종류를 합성곱하였다. 필터는 보기 쉽도록 이미지보다 상대적으로 크게 확대 표시하였다. 또, 필터와 합성곱 결과 이미지는 음의 값을 갖는 픽셀이 검게, 양의 값을 갖는 픽셀은 희게 나타나도록 명암값을 조정하였다.

〈딥러닝 제대로 시작하기〉

Stride

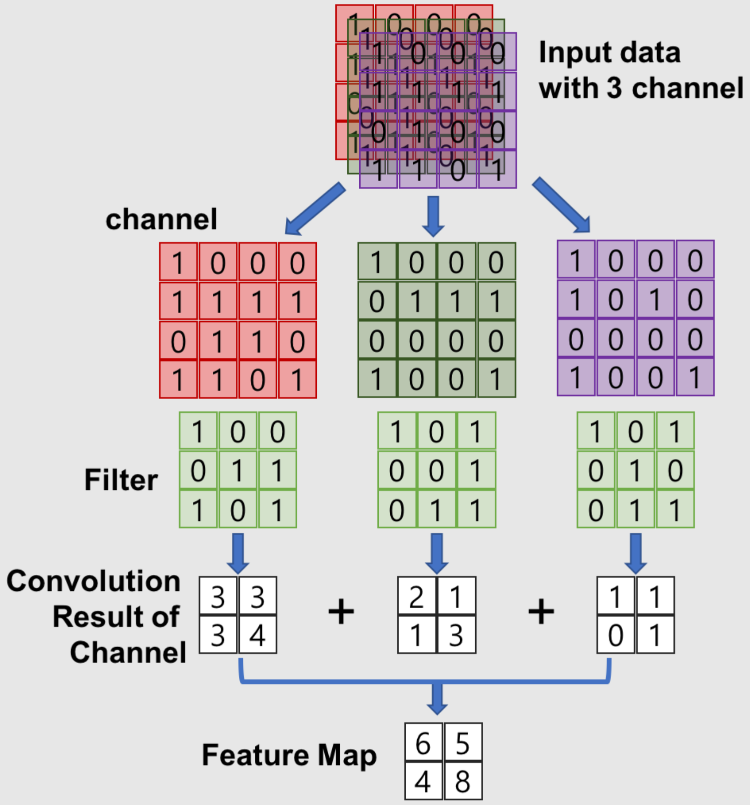
필터는 입력 데이터를 지정한 간격으로 순회하면서 합성곱을 계산합니다. 여기서 지정된 간격으로 필터를 순회하는 간격을 Stride라고 합니다. <그림 4>는 stride가 1로 필터를 입력 데이터에 순회하는 예제입니다. stride가 2로 설정되면 필터는 2칸씩 이동하면서 합성곱을 계산합니다.



<http://taewan.kim/post/cnn/>

Stride

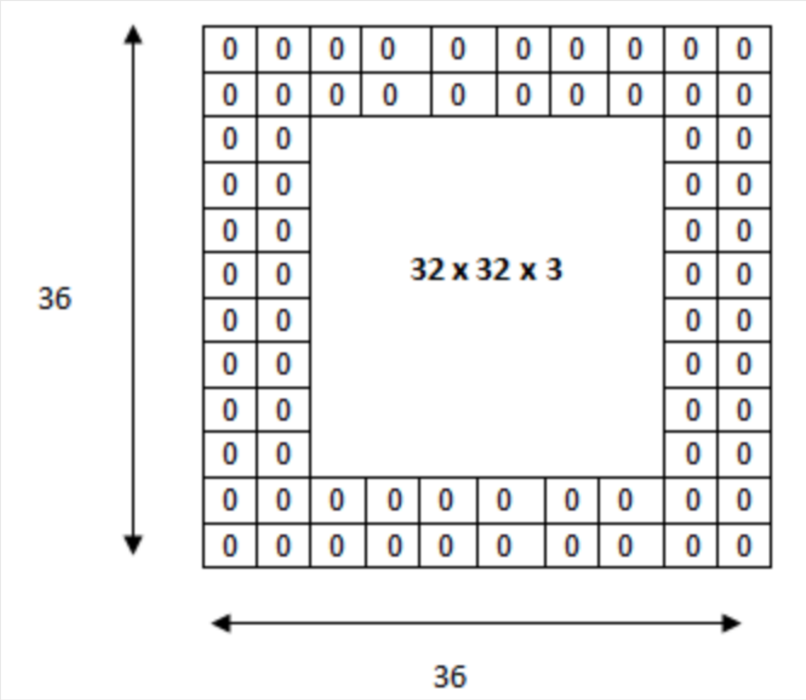
입력 데이터가 여러 채널을 갖을 경우 필터는 각 채널을 순회하며 합성곱을 계산한 후, 채널별 피처 맵을 만듭니다. 그리고 각 채널의 피처 맵을 합산하여 최종 피처 맵으로 반환합니다. 입력 데이터는 채널 수와 상관없이 필터 별로 1개의 피처 맵이 만들어 집니다.



<http://taewan.kim/post/cnn/>

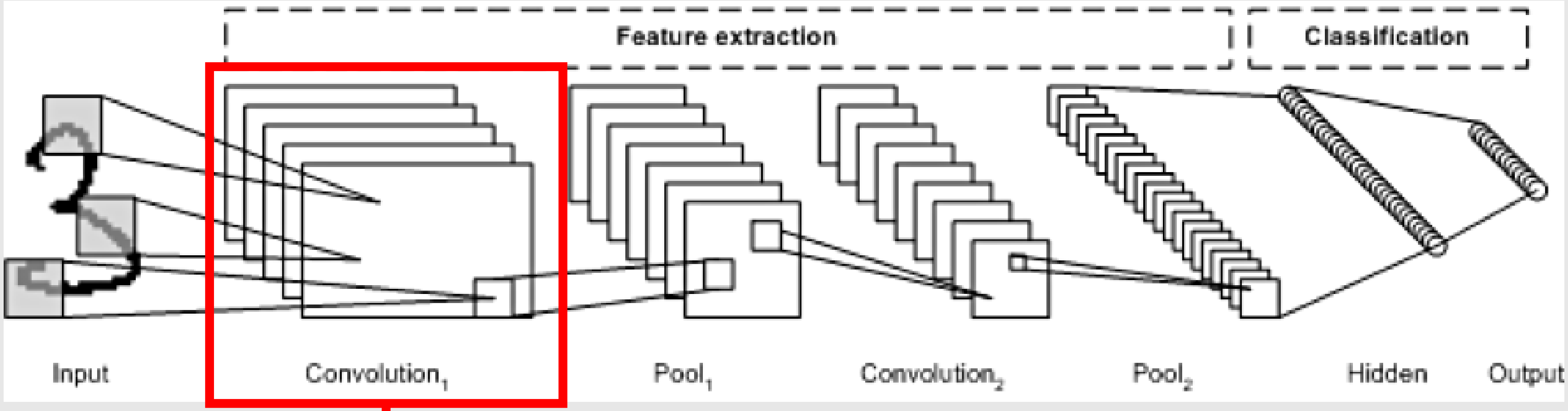
Padding

Convolution 레이어에서 Filter와 Stride에 작용으로 Feature Map 크기는 입력데이터 보다 작습니다.
Convolution 레이어의 출력 데이터가 줄어드는 것을 방지하는 방법이 패딩입니다. 패딩은 입력 데이터의 외각에 지정된 픽셀만큼 특정 값으로 채워 넣는 것을 의미합니다. 보통 패딩 값으로 0으로 채워 넣습니다.



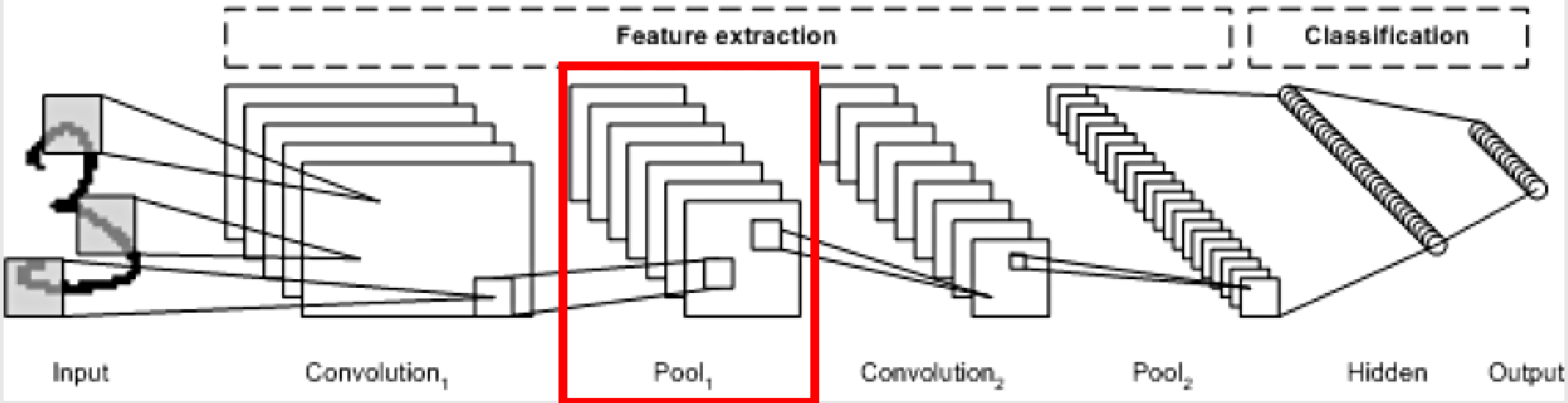
<http://taewan.kim/post/cnn/>

Convolution Layer



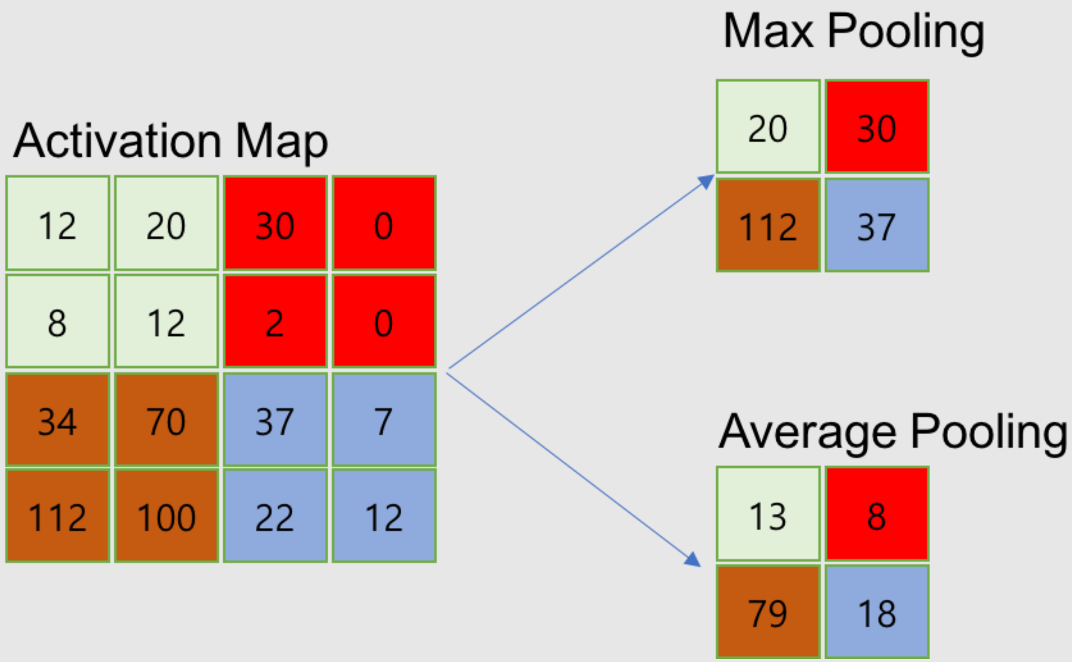
특징을 파악하고 학습하는 층

Pooling Layer



배운 특징을 강화하는 층

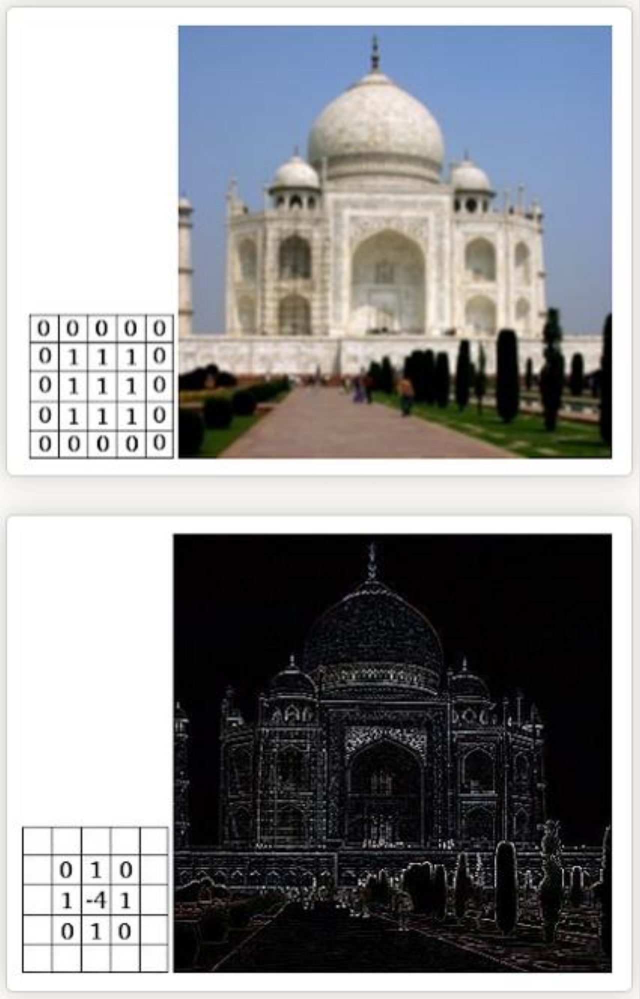
Pooling Layer



〈http://taewan.kim/post/cnn/〉

- 학습대상 파라미터가 없음
- Pooling레이어를 통과하면 행렬의 크기 감소
- 채널 수의 변화 또한 없음

Pooling Layer

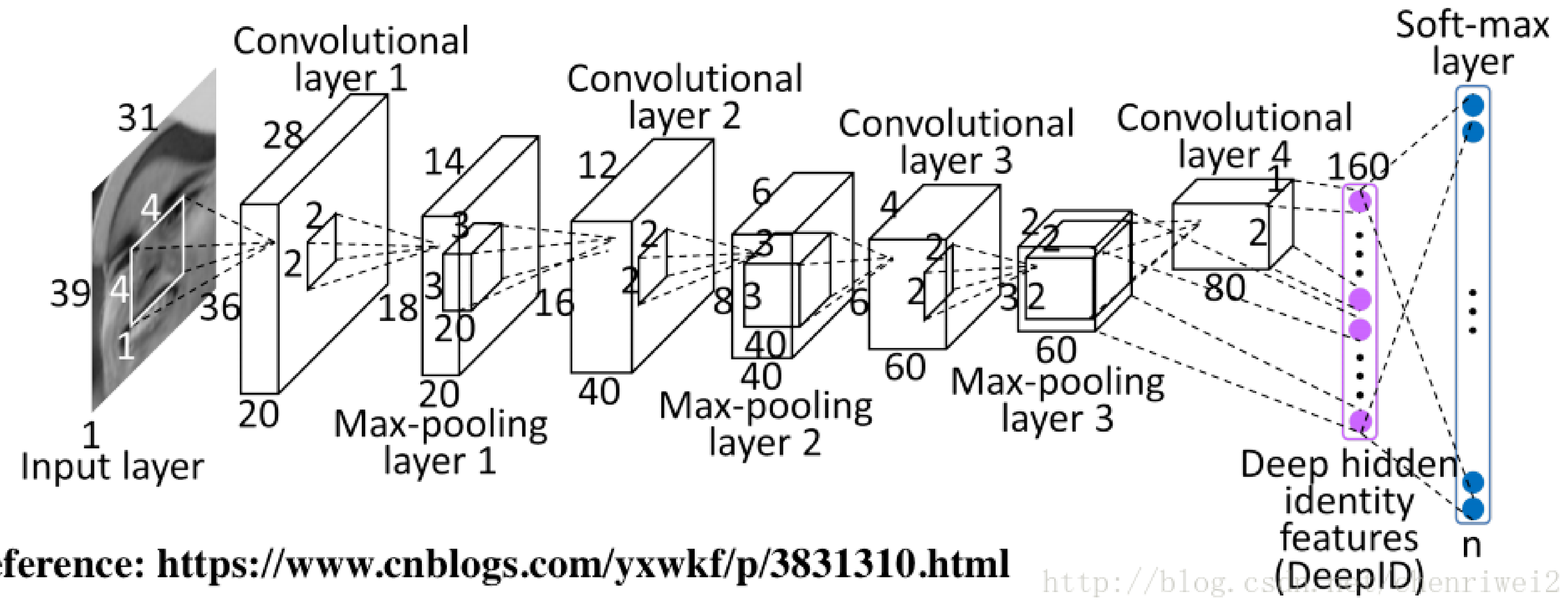


⟨https://brunch.co.kr/@chris-song/24⟩

학습해야 할
파라미터 수 ↓

특징 강화로
성능 ↑

Convolution Neural Network Architecture



Reference: https://www.cnblogs.com/yxwkf/p/3831310.html

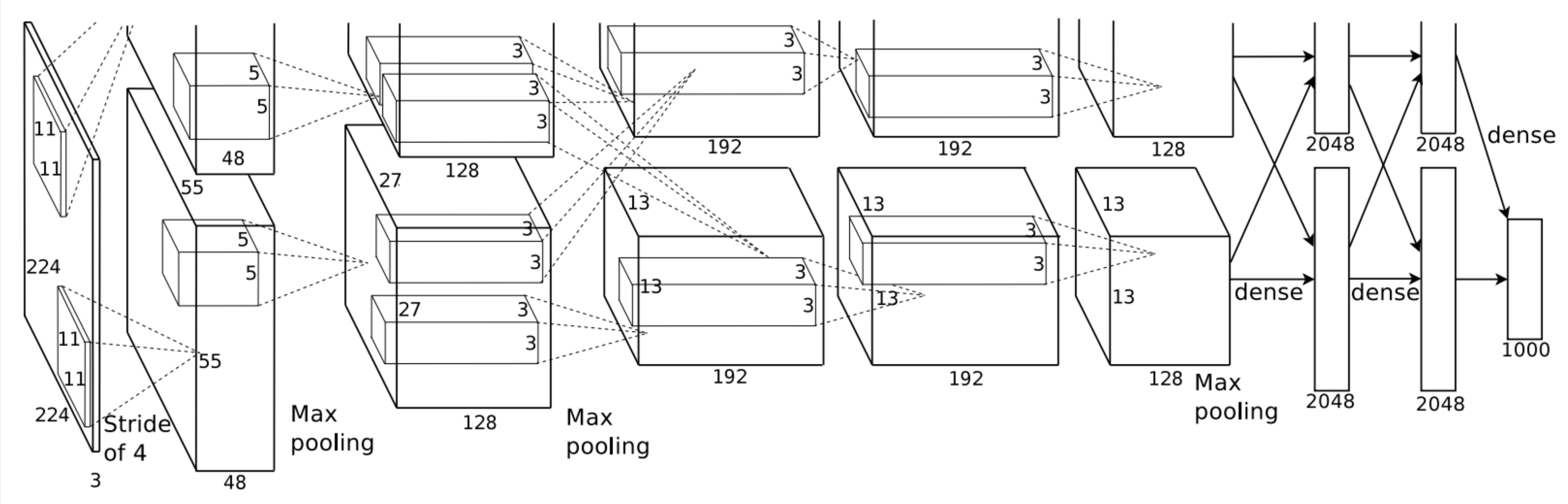
Parameter Architecture

- 입력데이터 Shape: (39, 31, 1)
- 분류 클래스: 100

layer	Filter	Stride	Max Pooling	activation function
Convolution Layer 1	(4, 4, 20)	1	X	relu
Max Pooling Lyaer 1	X	2	(2, 2)	X
Convolution Layer 2	(3, 3, 40)	1	X	relu
Max Pooling Lyaer 2	X	2	(2, 2)	X
Convolution Layer 3	(3, 3, 60)	1	1	relu
Max Pooling Lyaer 3	X	2	(2, 2)	X
Convolution Layer 4	(2, 2, 80)	1	1	relu
Flatten	X	X	X	X
fully connected Layer	X	X	X	softmax

〈http://taewan.kim/post/cnn/〉

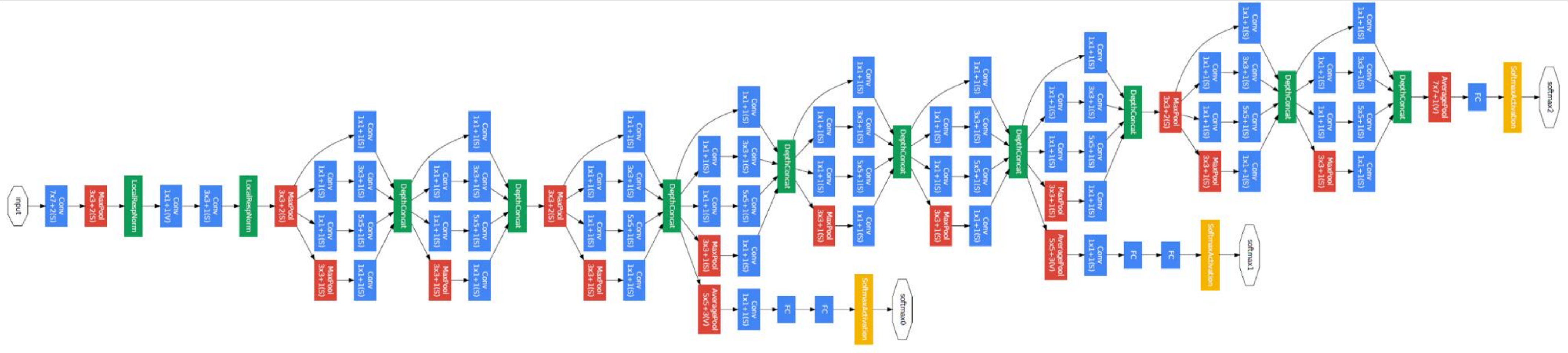
AlexNet

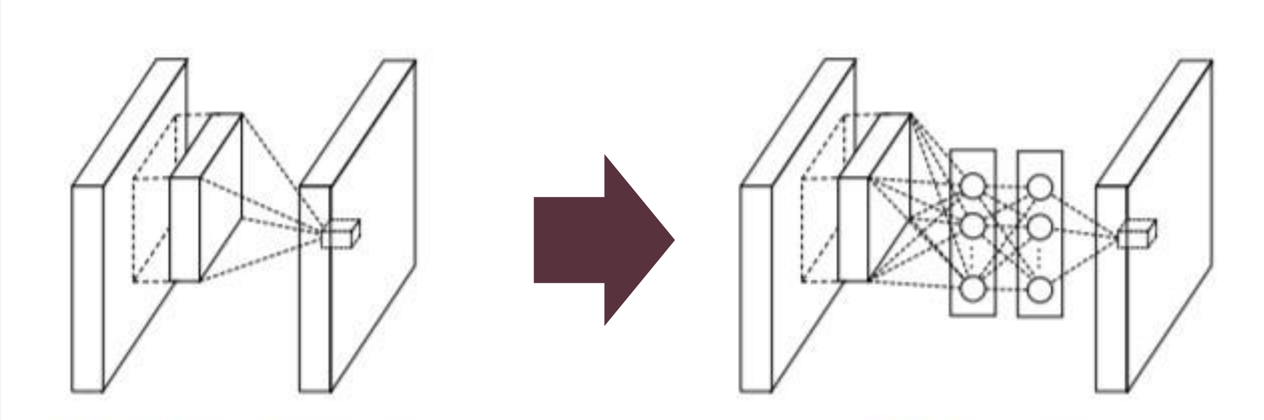


- Conv Layer, Max-Pooling Layer, Dropout Layer 5개
- Fully Connected Layer 3개
- Nonlinearity Function: ReLU
- Batch Stochastic Gradient Descent

➔ 의미있는 성능을 낸 첫 번째 CNN Architecture
➔ Dropout이 표준으로 자리잡음
➔ 병렬처리가 선풍적인 인기를 얻음

GoogleNet

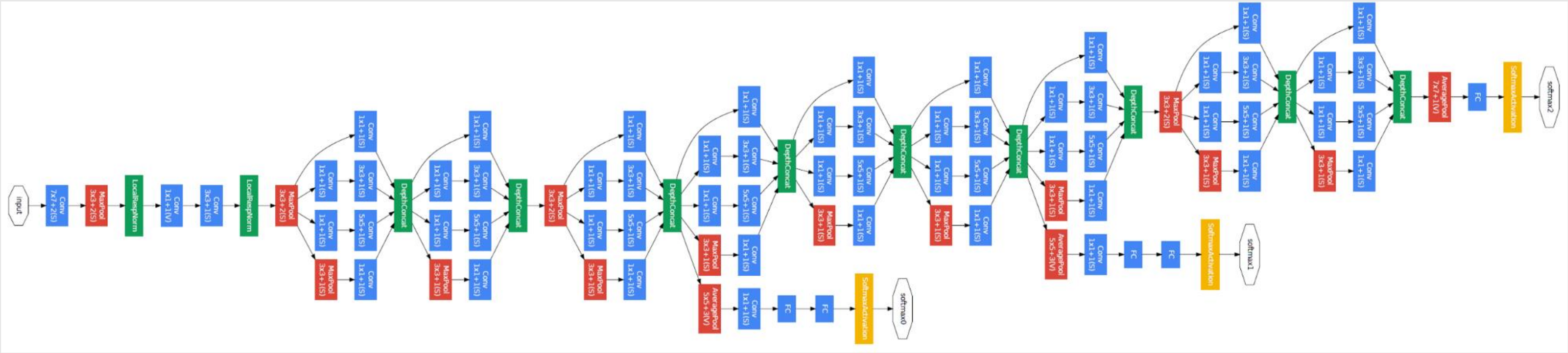




(a) Linear convolution layer (b) Mlpconv layer

〈http://nmhkahn.github.io/Casestudy-CNN〉

GoogleNet

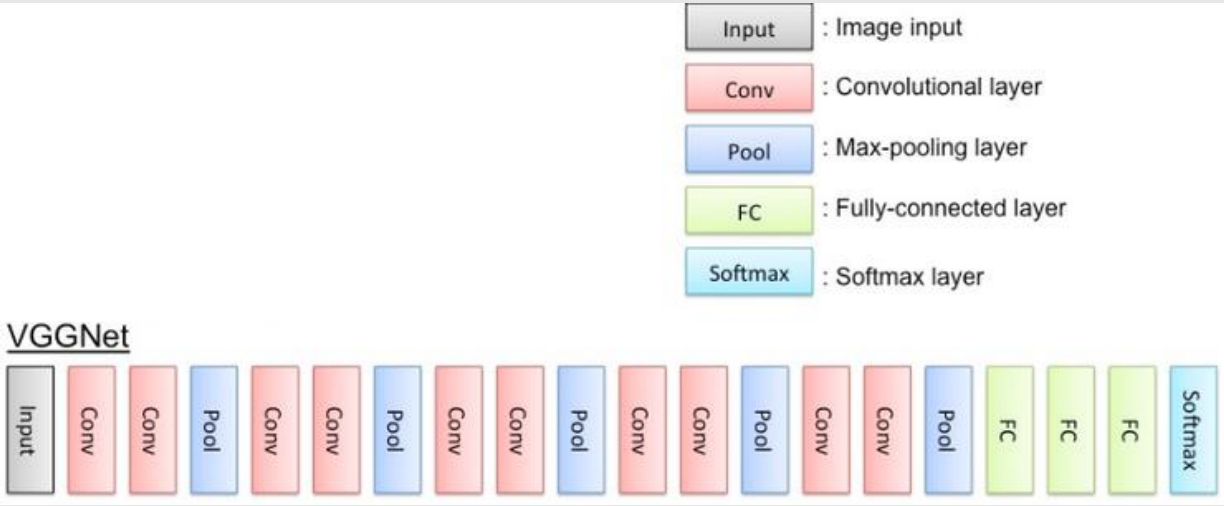


- Global Average Pooling
- 1x1 Convolution Layer
- 22Layer

➔ 2014년 IRSVRC 우승
➔ 구현을 위한 실질적인 문제들을 잘 해결
➔ 첫 Inception Module사용

https://norman3.github.io/papers/docs/google_inception.html 초강추

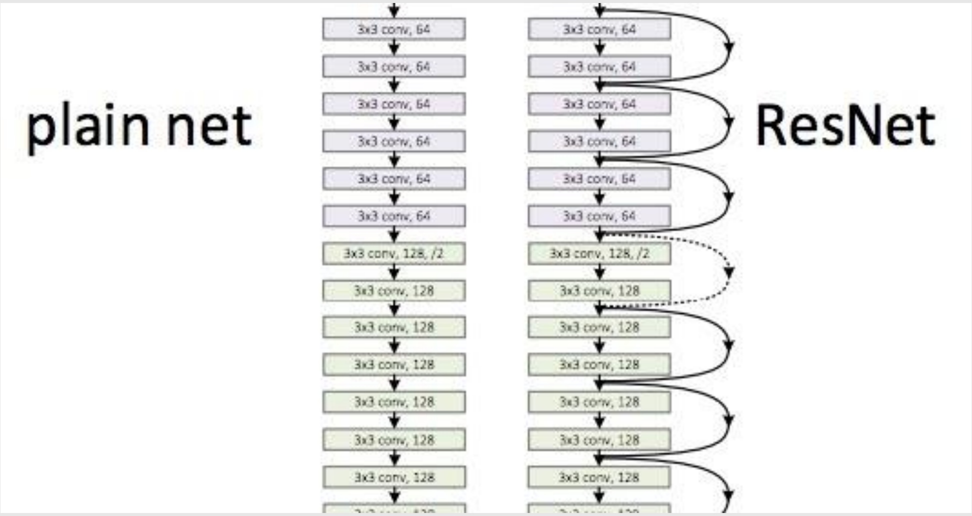
VGGnet

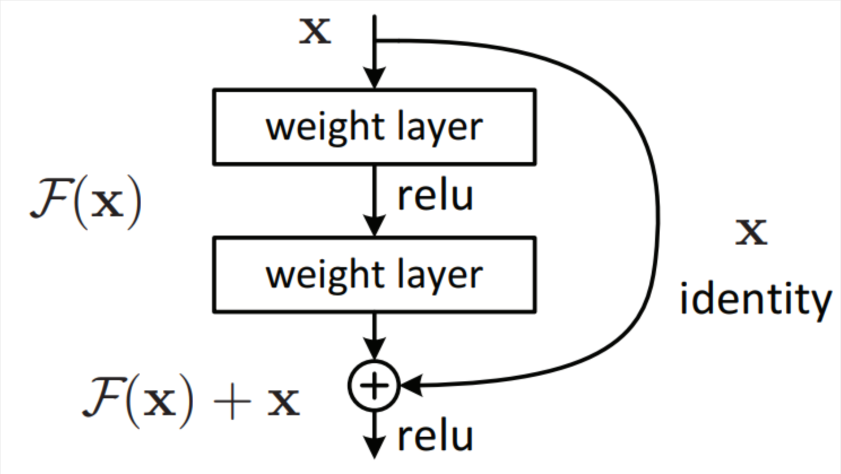


- AlexNet의 진화버전
- Image Rescaling사용 (Data Augmentation)
- 19 Layer

➔ 2014년 IRSVRC 준우승
➔ 가장 대중적으로 많이 사용됨
➔ VGG16과 VGG19가 존재

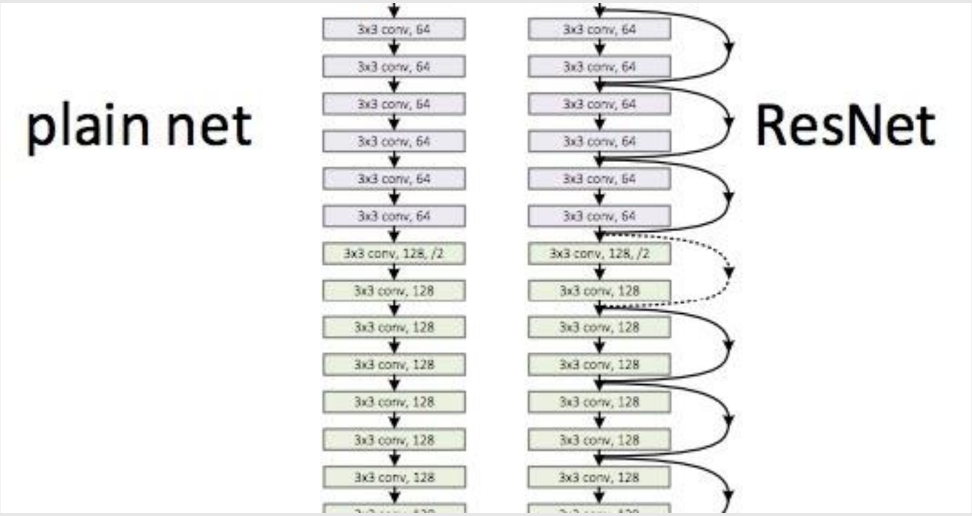
ResNet





<https://ratsgo.github.io/deep%20learning/2017/10/09/CNNs/>

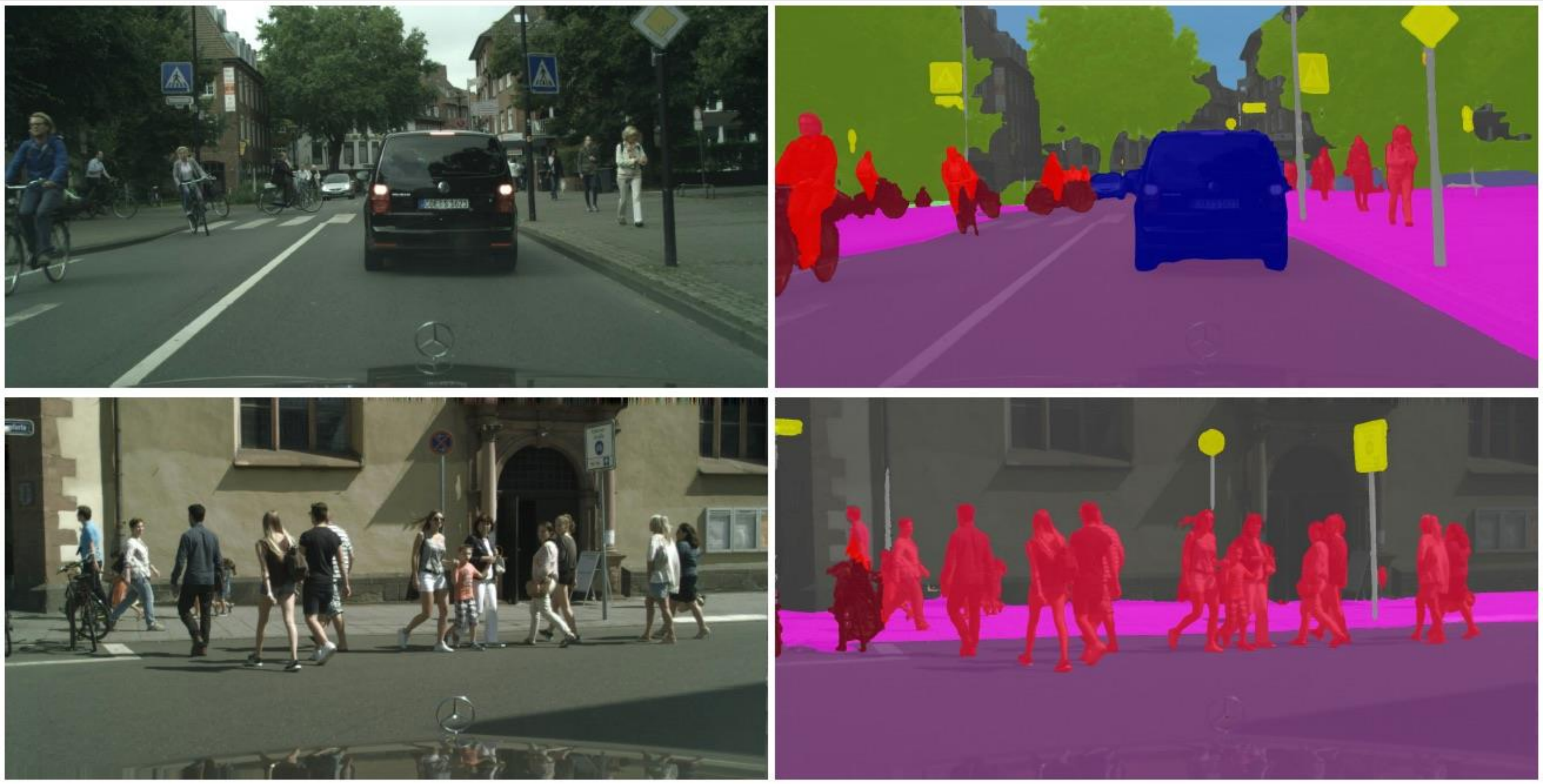
ResNet



- Residual Block
- Only 5layer

➔ Residual Block을 통한 Gradient Vanishing완화
➔ 또한 일종의 지름길(Skip Connection)역할
➔ 오류율 3.6% (5% 돌파) ▶ 인간의 분류 오차가 5~10%

Object Detection



R-CNN

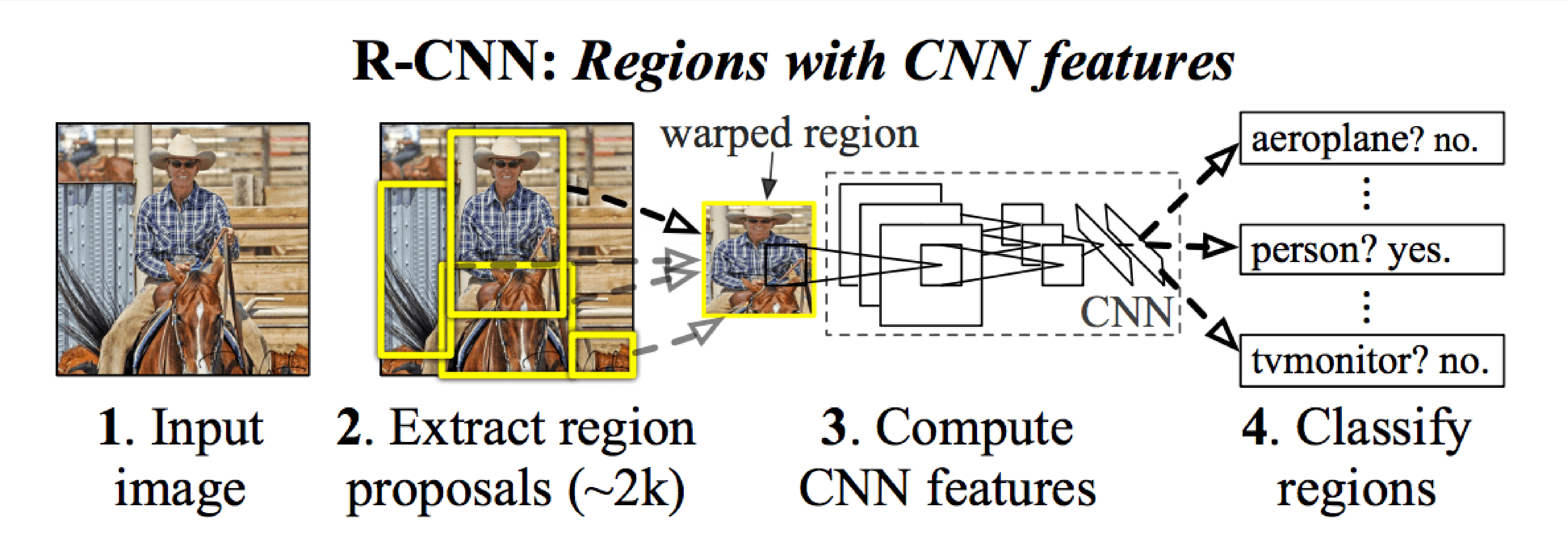
Rich feature hierarchies for accurate object detection and semantic segmentation

Tech report (v5)

Ross Girshick Jeff Donahue Trevor Darrell Jitendra Malik
UC Berkeley
{rbg,jdonahue,trevor,malik}@eecs.berkeley.edu

2013년 11월

R-CNN


R-CNN: Regions with CNN features
warped region
aeroplane? no.
person? yes.
tvmonitor? no.
CNN
1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

R-CNN

1. 인식을 시킬 대상이 될 만한 객체를 담는 Bounding Box생성한다.
2. AlexNet에 각각의 Box를 넣어본다.
3. SVM을 통해 각 박스가 어떤 객체인지 분류
4. 객체가 인식되면 회귀를 통해 박스 크기 조절

R-CNN

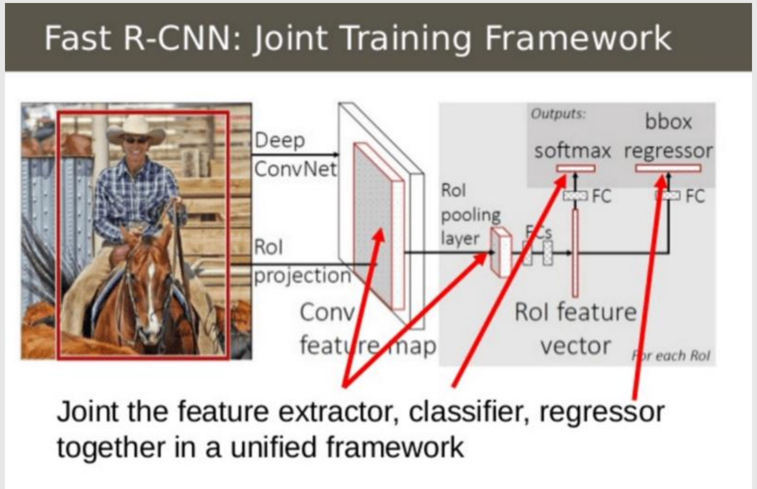
R-CNN은 잘 작동했지만, 몇가지 이유로 꽤 느렸다:

1. 매번 하나 이미지에서 나오는 모든 각각의 제안된 영역을 각각 CNN(AlexNet)을 통과시켜야 하는데, 이는 하나의 이미지당 2000번의 forward pass를 거쳐야 한다는 이야기이다.
2. 3개의 다른 모델을 학습시켜야 했다. Image feature를 생성하는 것, classifier가 class를 예측하는 것, regression model이 bouding box를 찾아낸 것. 이것이 전체적인 pipeline을 학습시키기 어렵게 하였다.

〈https://junn.in/archives/2517〉

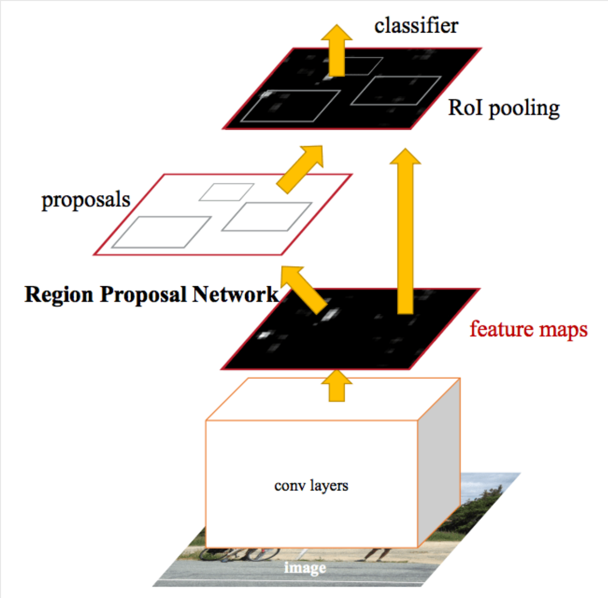
R-CNN History

2015년 4월:
Fast R-CNN



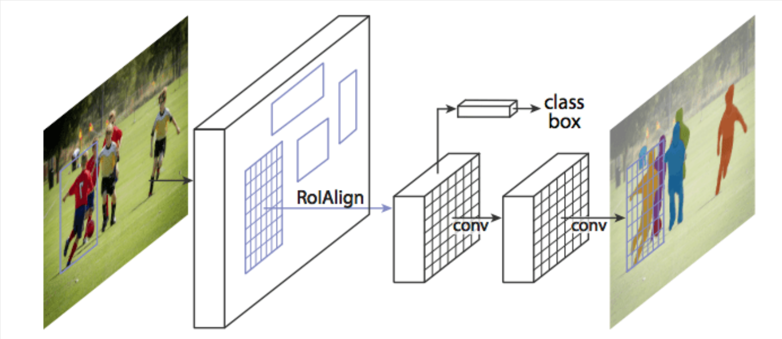
<https://junn.in/archives/2517>

2015년 6월:
Faster R-CNN



<https://arxiv.org/abs/1506.01497>

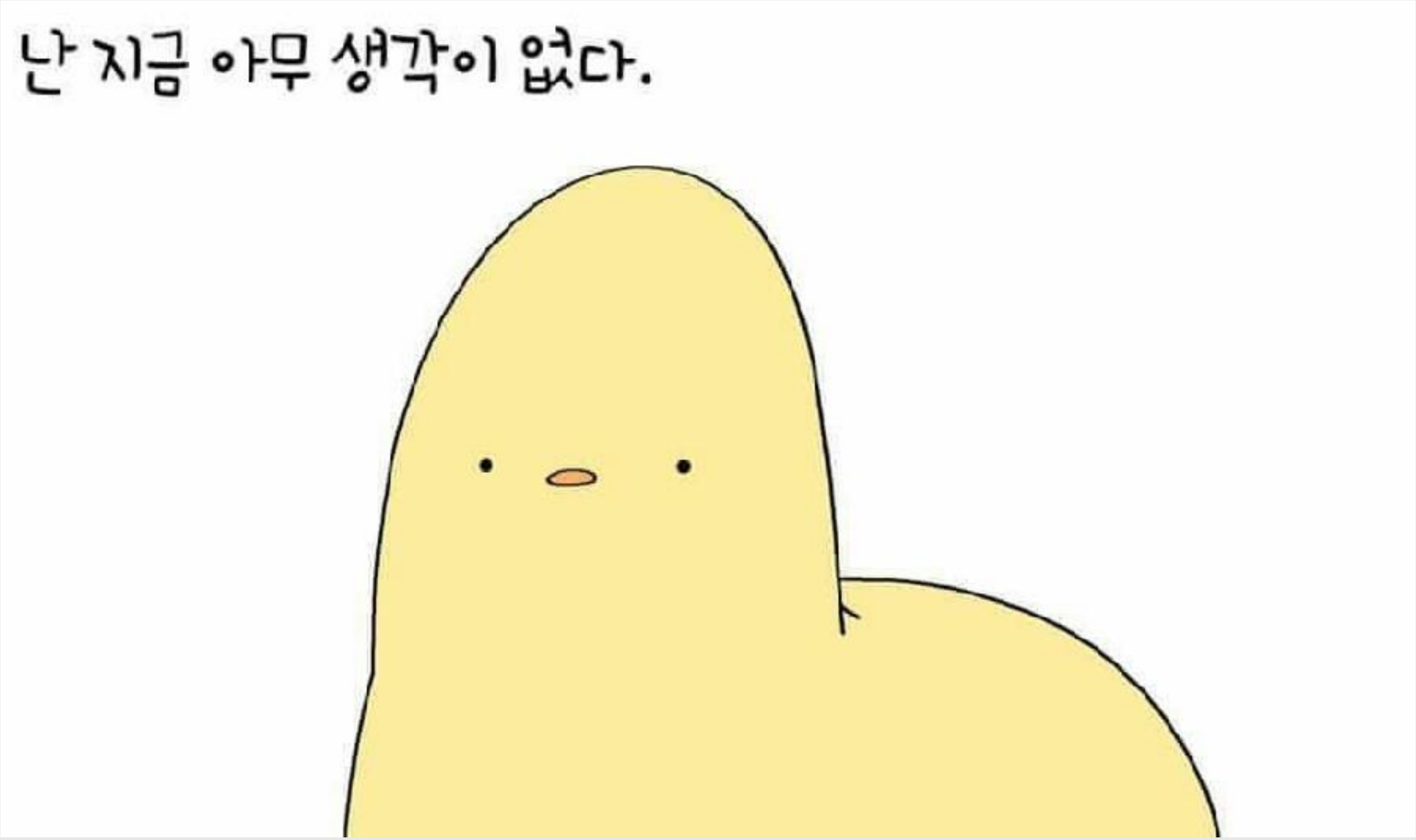
2017년 4월:
Mask R-CNN



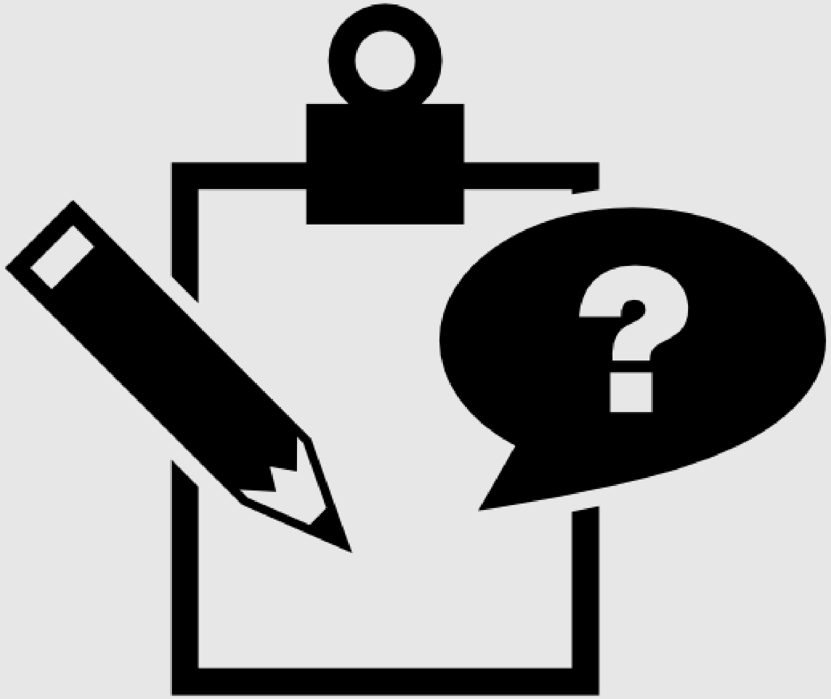
<https://junn.in/archives/2517>

2016년 5월: **YOLO**

R-CNN History



<http://www.maybugs.com/news/articleView.html?idxno=571516>

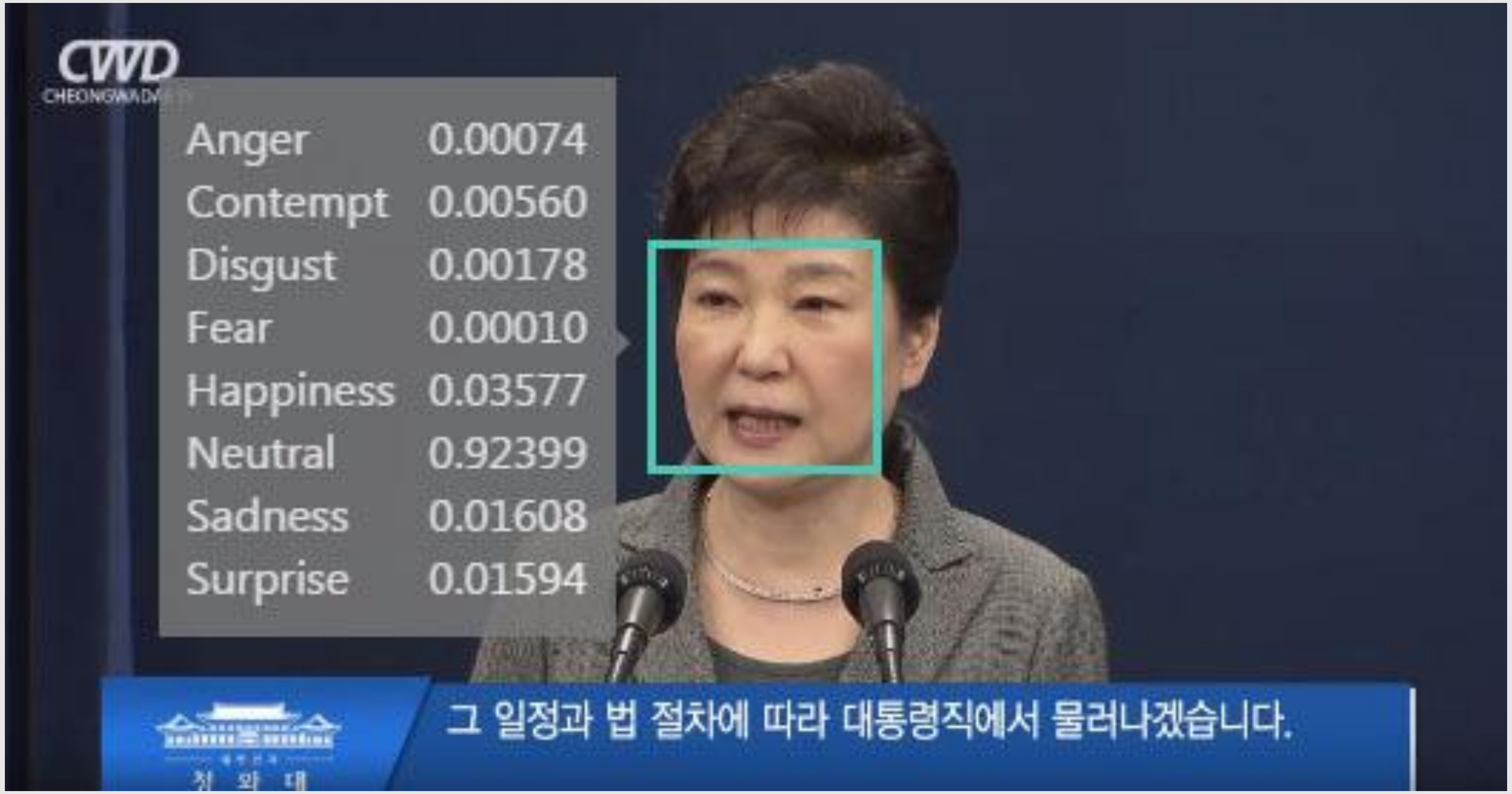


Before
Finish Class

503

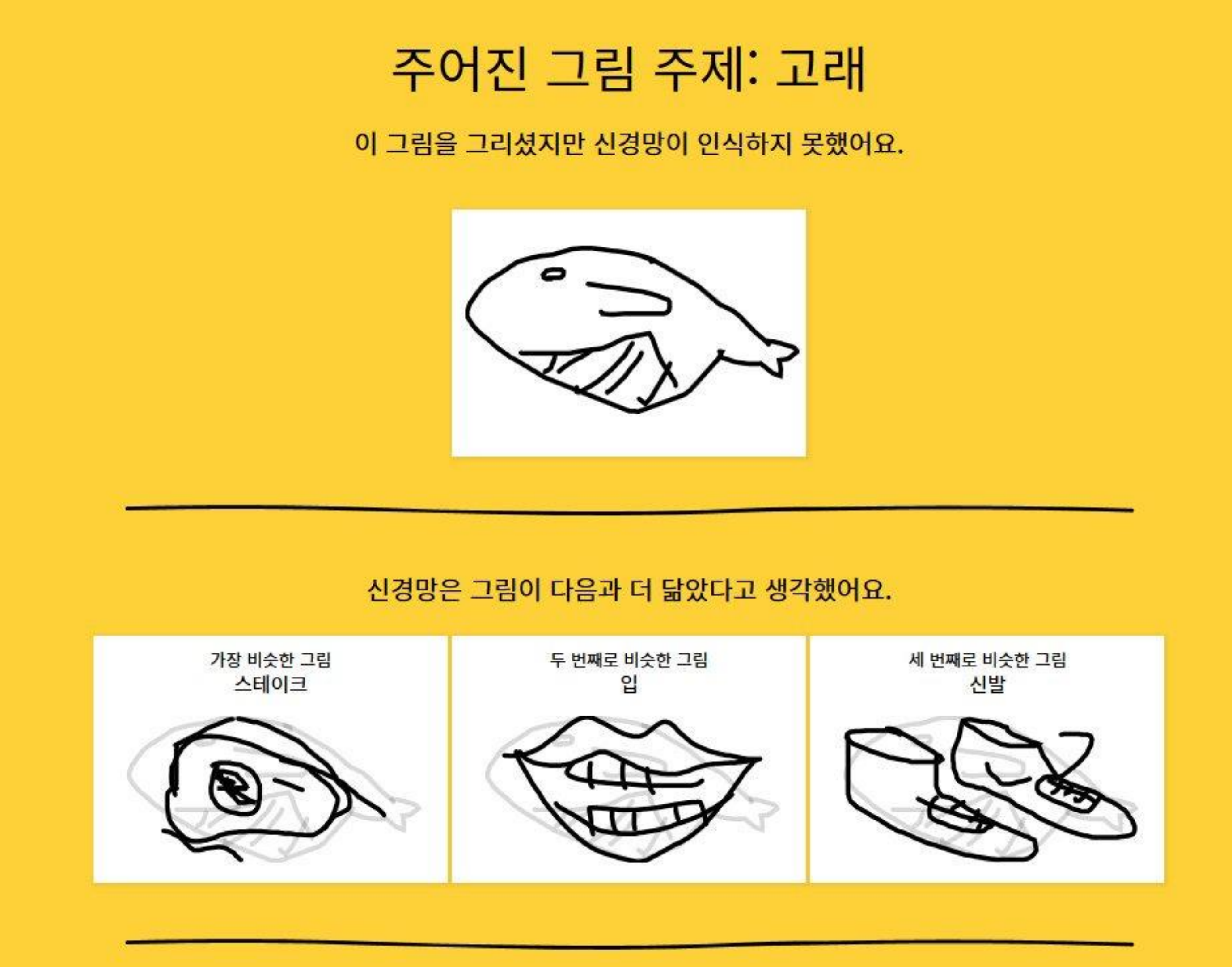
〈http://hub.zum.com/heraldcorp/6479〉

503



〈http://hub.zum.com/heraldcorp/6479〉

Quick Draw



〈https://twitter.com/cow_cow_cow/status/1013206865230958592〉

Schedule

12월 07일: 14장 (RNN)

12월 14일: 15장 (Autoencoder)

12월 21일: 총 정리 + 시험 + 망년회 **[방어회!]**


Ideas worth spreading
- TED Talks

고생하셨습니다.