강의계획표

주	해당 장	주제
1	1장 2장, 3장	머신러닝이란
2		머신러닝을 위한 기초지식, 구현을 위한 도구
3	4장	선형 회귀로 이해하는 지도학습
4	5장	분류와 군집화로 이해하는 지도 학습과 비지도 학습
5	6장	다양한 머신러닝 기법들 - 다항 회귀, Logistic Regression - 정보이론, 결정트리 - SVM, Ensemble
6		
7		
8		중간고사 (04-20)
9	7장	인공 신경망 기초 - 문제와 돌파구
10	8장	고급 인공 신경망 구현
11	9장	신경망 부흥의 시작, 합성곱 신경망
12	10장	순환 신경망
13	11장	차원축소와 매니폴드 학습
14	12장	오토인코더와 잠재표현 학습
15		보강주
16		기말고사

9장 신경망 부흥의 시작, 합성곱 신경망

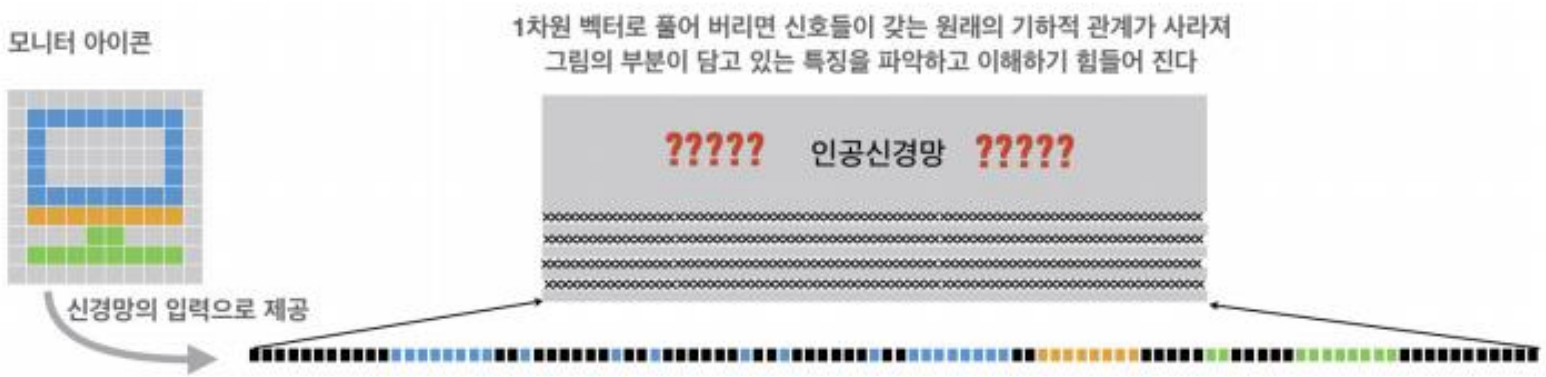
11 주차

- 이미지 인식에 신경망을 적용할 때의 문제는 무엇일까.
- 합성공 신경망은 어떤 원리로 동작하는가.
- 합성곱 신경망이 잘 동작하는 이유는 무엇인가.
- 패딩과 풀링은 신경망의 학습에 어떤 영향을 미치는가.

시각 정보 처리와 합성곱의 필요성

❖ 기존 MLP의 image 처리(MNIST)

- 2차원 배열을 1차원 벡터로 flatten(평탄화)함으로써 2차원 image의 특징(인접 pixel) 손실
- 특징추출 필요

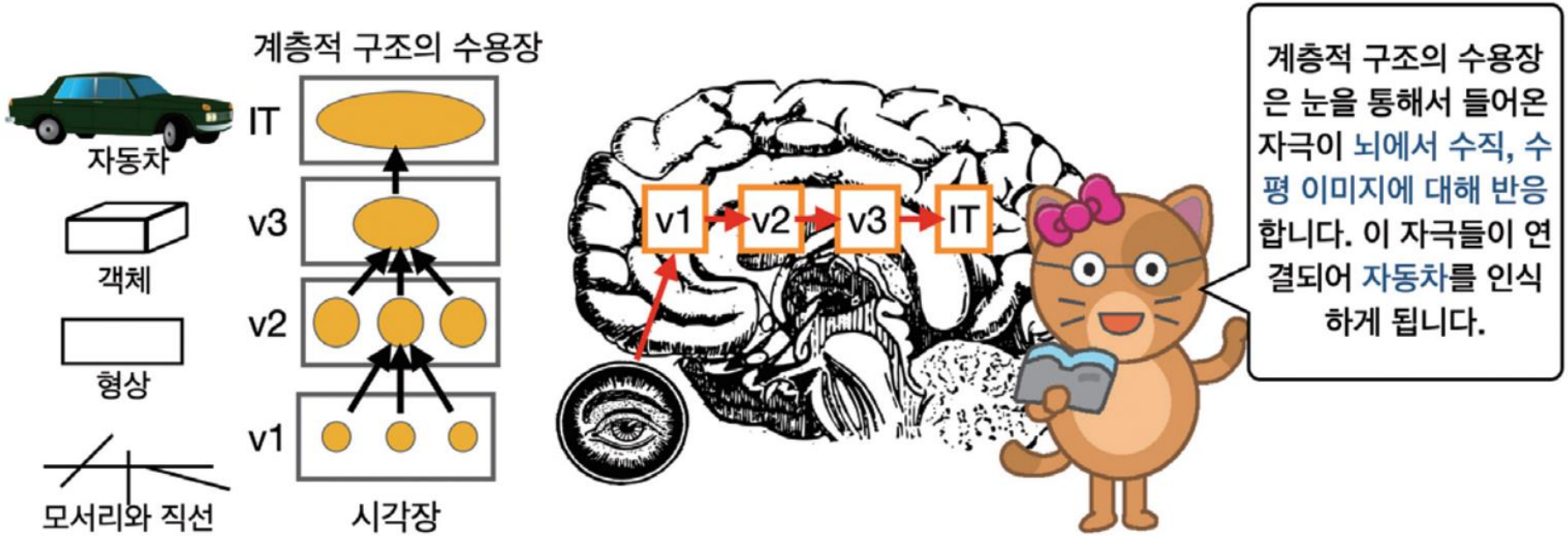


❖ 동물의 시각 인식 실험(1950년대 말)

- 데이비드 허벨David Hubel과 토르스텐 비셀Torsten Wiesel
- 시각 피질visual cortex에 있는 뉴런들이 시야의 일부 범위 안에 있는 영역에 대해 활성화되어 정보를 받아들임
- 수용장receptive field: 하나의 뉴런을 활성화시키는 데에 영향을 미치는 시각 정보의 영역

수용장과 합성곱 신경망

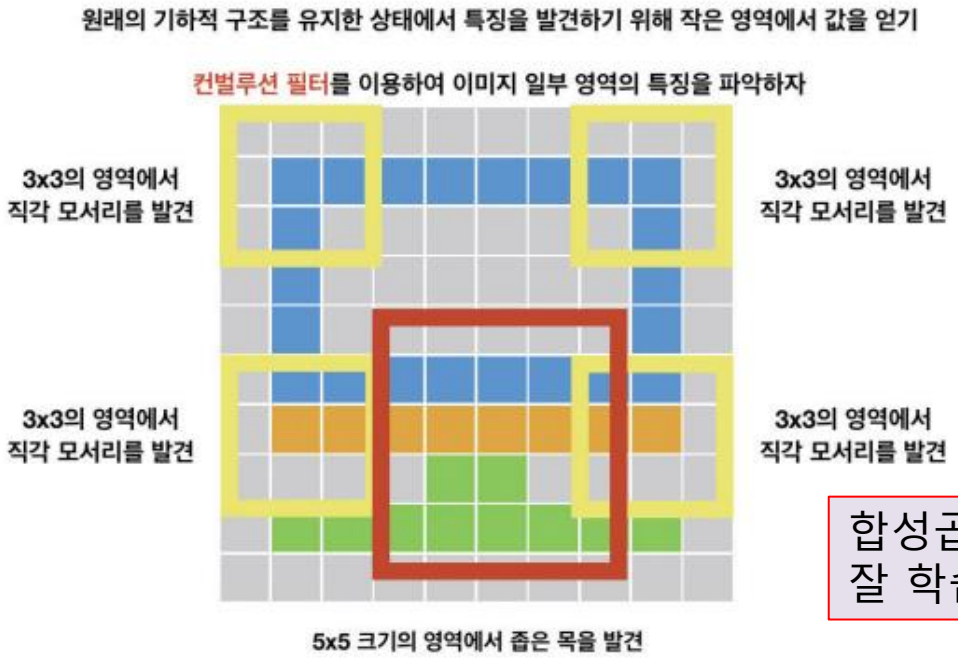
- 어떤 뉴런은 **수평선 이미지에 반응**하고, 어떤 뉴런은 **수직선 이미지에 반응**하며 이러한 낮은 수준의 이미지에 대한 반응을 조합하는 더 **큰 수용장이 패턴에 반응**한다는 결과였다.



합성곱의 기본 개념 이해 (1)

❖ 합성곱 신경망

- 합성곱: 특정한 영역(수용장) 내에 있는 모든 픽셀pixel 정보로 하나의 값(특징)으로 변환
- 시각 수용장의 역할을 합성곱 혹은 컨볼루션convolution 연산이 수행해 그 결과가 신경세포로 전달
- 입력에 대한 전처리(flatten, 특징 추출)가 불필요
- 학습 과정에 입력을 특징을 추출하는 방법도 함께 학습

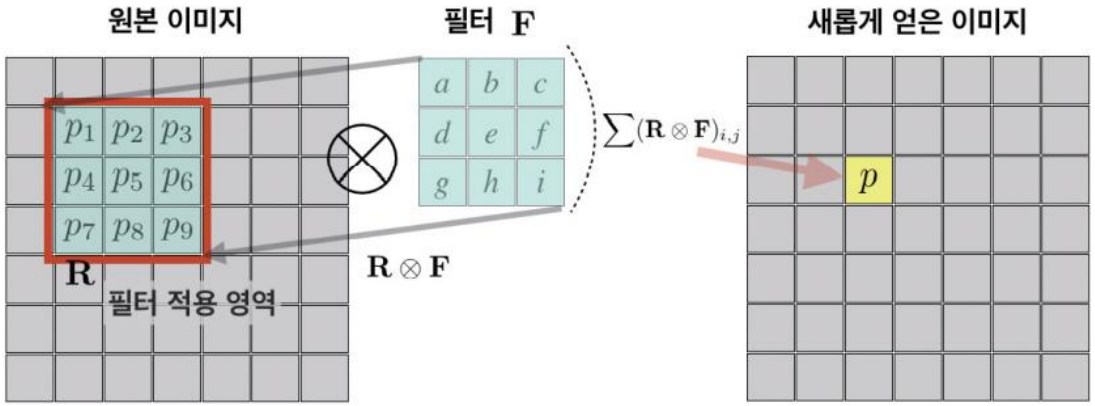


합성곱 신경망 : 2차원 필터를
잘 학습시키는 것이 더 나을 것이다

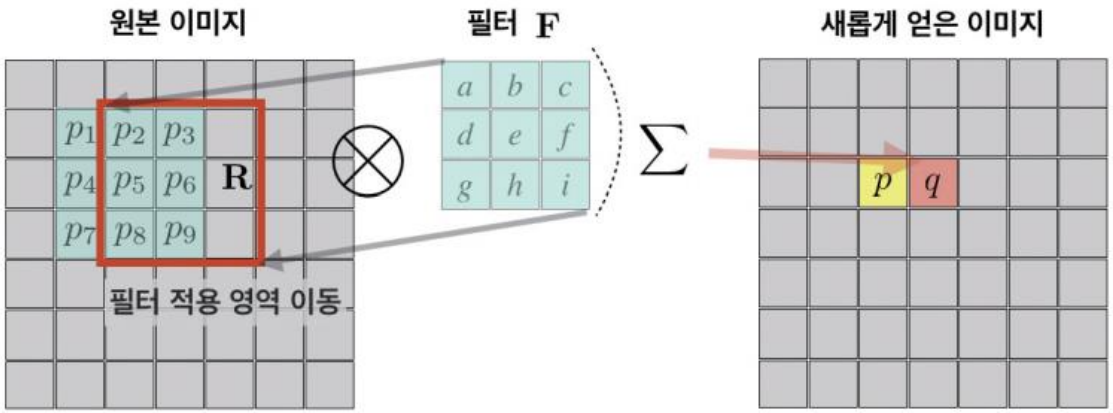
합성곱의 기본 개념 (1)

❖ 합성곱연산(컨볼루션)

- 필터filter, 커널kernel: 수용장
- 필터를 통한 이미지 변형



$$p = a \cdot p_1 + b \cdot p_2 + c \cdot p_3 + d \cdot p_4 + e \cdot p_5 + f \cdot p_6 + g \cdot p_7 + h \cdot p_8 + I \cdot p_9$$



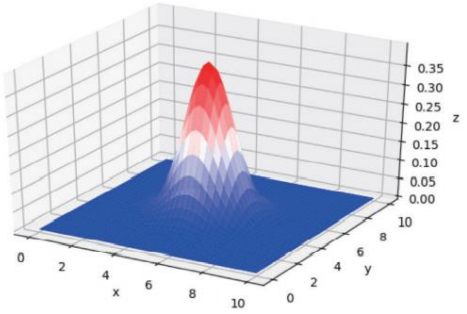
합성곱의 기본 개념 (2)

❖ Filtering

- 평균 필터average filter, **상자 필터**box filter

$$\mathbf{M} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- 가우스 필터
  - 중심 픽셀에는 더 높은 중요도(가중치)를 부여하고, 중심에서 멀어질수록 중요성을 낮게 함



원본 이미지 상자 필터를 이용한 합성곱 가우스 필터를 이용한 합성곱

이미지와 영상 처리 분야에서는 필터를 사용하여 이미지를 가공하는 것이 일반적이다

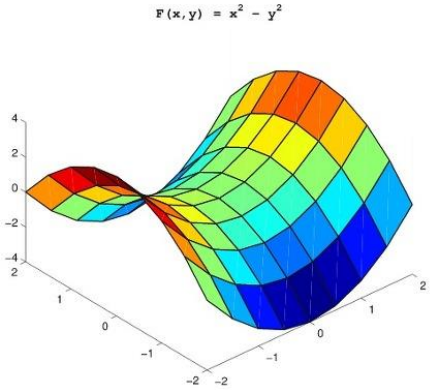
합성곱을 통한 특징 추출 (1)

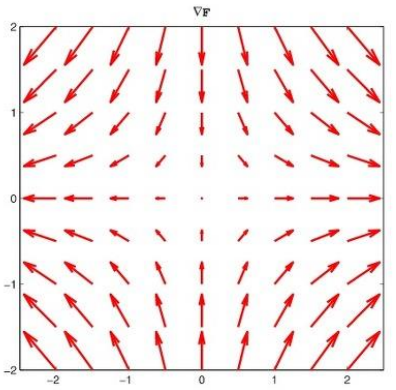
- 윤곽선 추출
  - 색이 연속적이지 않고 갑작스럽게 변하는 픽셀들을 찾아냄
  - 이미지 색상: 픽셀 위치를 정의역domain으로 하고 픽셀의 색상을 치역range으로 하는 함수, f(pixel)
  - 픽셀의 각 지점마다 색상 값이 변하는 방향으로 기울기(∇, 나블라)를 구하면, 값이 크게 변하는 곳에서는 이 기울기 벡터의 크기가 커짐
  - 라플라시안Laplacian 연산 Δ: 2차 미분 연산, 기울기의 발산Divergence of Gradient
    - https://micropilot.tistory.com/2970
    - Divergence 연산자($\nabla \cdot f$): 단위 단면적으로부터 퍼져 나가는 정도를 알려 줌. Divergence 가 0이라면 퍼져 나가는 정도와 유입되는 정도가 동일하므로 벡터의 흐름이 균일한 상태를 의미
    - 변화가 큰 윤곽선 지점에서 큰 값이 나옴

$$\Delta f = \nabla^2 f = \nabla \cdot \nabla f$$

$$\nabla = \left(\frac{\partial}{\partial x_1}, \cdots, \frac{\partial}{\partial x_n} \right)$$

$$\Delta f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2}$$





합성곱을 통한 특징 추출 (2)

- 이미지(2차원 공간)에 라플라스 연산
  - 각 픽셀에 대해 각 축으로 두 번 미분한 것을 합산
  - 이미지 생성함수를 찾기는 불가능하므로, 실제 함수 미분의 근사치가 되는 수치 미분을 적용
  - 픽셀 사이의 간격을 단위 길이 1이라고 가정하고 인접 픽셀의 차를 구하면 바로 1차 미분의 근사치
  - 1차 미분의 근사치의 행렬에서 인접 원소의 차를 구하면 2차 미분
  - 픽셀 $I_{i,j}$를 x_1축 방향으로 두 번 미분한 값의 근사치

$$\frac{\partial^2 I_{i,j}}{\partial x_1^2} = (I_{i+1,j} - I_{i,j}) - (I_{i,j} - I_{i-1,j}) = I_{i+1,j} - 2I_{i,j} + I_{i-1,j}$$

  - x_2축 방향으로 두 번 미분한 값의 근사치

$$\frac{\partial^2 I_{i,j}}{\partial x_2^2} = (I_{i,j+1} - I_{i,j}) - (I_{i,j} - I_{i,j-1}) = I_{i,j+1} - 2I_{i,j} + I_{i,j-1}$$

  - 특정 픽셀의 라플라시안 연산 결과

$$\Delta I_{i,j} = \frac{\partial^2 I_{i,j}}{\partial x_1^2} + \frac{\partial^2 I_{i,j}}{\partial x_2^2} = I_{i,j+1} + I_{i,j-1} + I_{i+1,j} + I_{i-1,j} - 4I_{i,j}$$

  - 라플라시안 필트

$$\mathbf{K}_{Laplacian} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

합성곱을 통한 특징 추출 (3)

- 파이썬 openCV 19. 에지검출 : 라플라시안(Laplacian), LoG ...

0	−1	0
−1	4	−1
0	−1	0

(a) Laplacian 1

0	1	0
1	−4	1
0	1	0

(b) Laplacian 2

−1	−1	−1
−1	8	−1
−1	−1	−1

(c) Lap. 1, 8 direction

원본 영상

라플라시안 필터 1

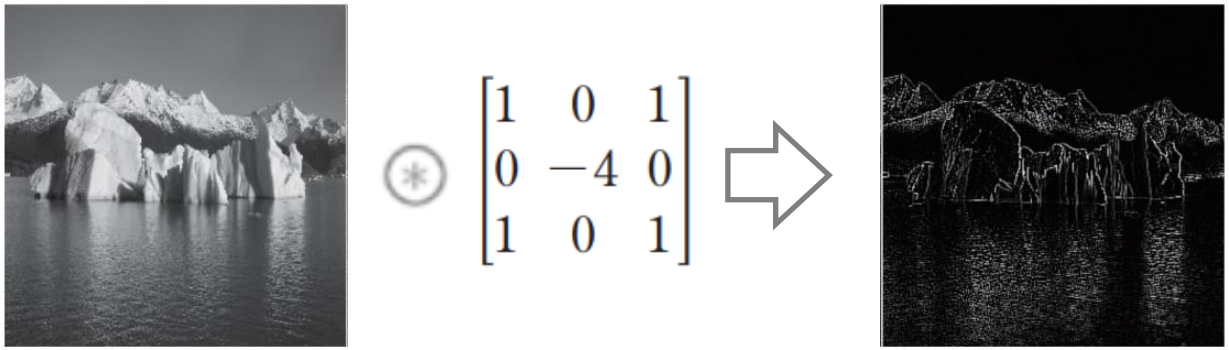
라플라시안 필터 2

라플라시안 필터 3

합성곱을 통한 특징 추출 (4)

❖ 이미지(2차원 공간)에 라플라스 연산

- 시각 정보를 처리할 때 라플라스 필터를 적용하여 윤곽선 특징을 추출해 낼 수 있음

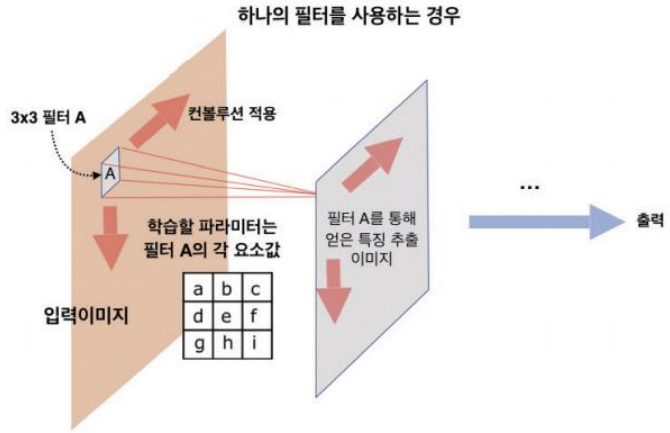


❖ 합성곱 신경망

- 합성곱 연산을 통해 원본 이미지에서 필터의 크기와 동일한 크기를 가진 부분 영역에 대해 해당 필터가 원하는 특징을 얻어냄
- 필터로 사용되는 행렬만 바꾸면 여러 가지 다른 결과를 얻을 수 있음
- 기계가 스스로 좋은 필터를 만들어냄 => **필터의 가중치를 모델의 파라미터로 학습**

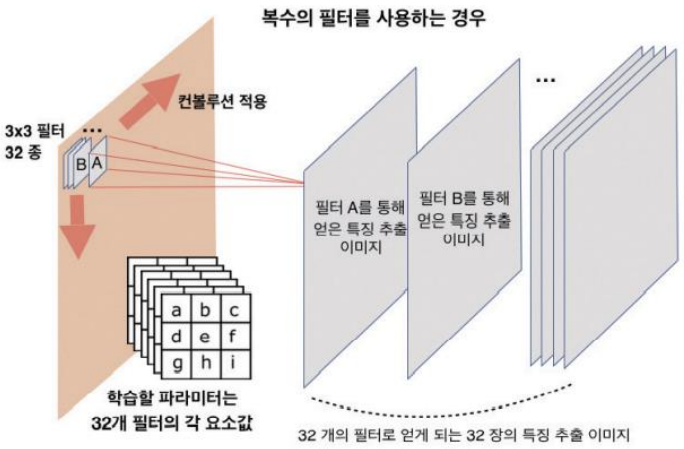
합성곱 수행 신경망의 기본 구조와 문제 (1)

❖ 필터를 학습하는 구조



- A필터: 3x3이면 9개의 요소
- 특징 이미지를 만드는 계층의 연결강도는 9개의 파라미터로 정의

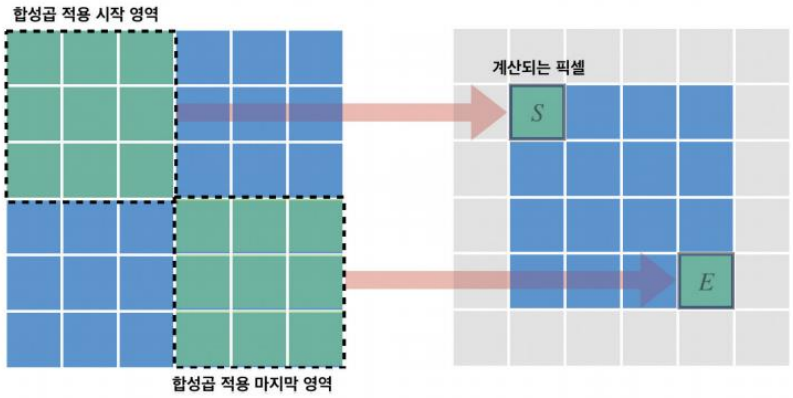
❖ 특성 맵feature map



- n개의 필터(예, 32)를 모델의 파라미터로 두고 학습한다고 가정
- 특성 맵: 하나의 이미지에 대해 모두 32장의 특징 추출 결과
- 학습을 통해 좋은 필터를 만듦

합성곱 수행 신경망의 기본 구조와 문제 (2)

❖ 문제점1: 점점 작아지는 이미지



- 3x3 필터에 대한 합성곱 결과 1개의 픽셀로 표현
- (n-1)x(n-1)의 합성곱 이미지 생성
- 연결망을 계속해서 쌓아 나가면 출력에 가까운 곳에서는 이미지가 지나치게 작은 크기로 줄어들게 됨
- Padding으로 해결

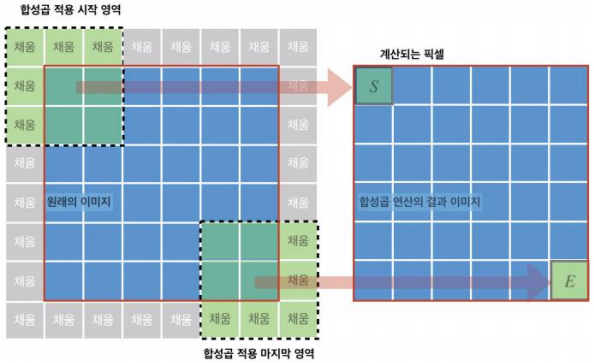
❖ 문제점2: 특징 추상화 능력 필요

- 합성곱을 거치면서 이미지의 크기가 작아지는 것이 이미지 전체의 특성을 요약한 것이 아님
- 이미지의 가장자리 정보들이 소실되고 중심부의 정보만 살아남는 것이기 때문에 정보를 추상화시키는 과정도 아님
- 정보를 요약하지 않으면 입력 단계의 작은 잡음이 가진 영향력도 계층을 거치며 사라지지 않고 계속해서 전파
- 풀링pooling: **정보를 요약하는 계층을 통해 이미지를 적절한 크기로 줄이는 과정**

패딩, 스트라이딩, 다중 채널

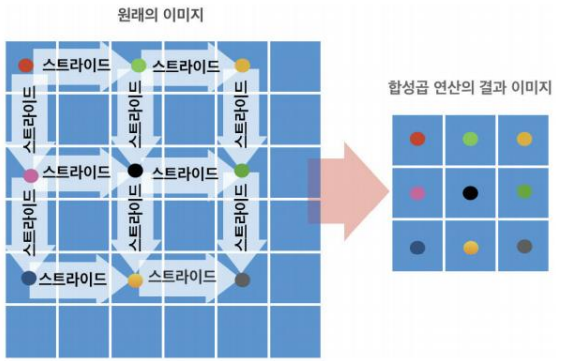
❖ 패딩padding

- 입력 이미지의 주변에 값을 덧대어 채워주는 일

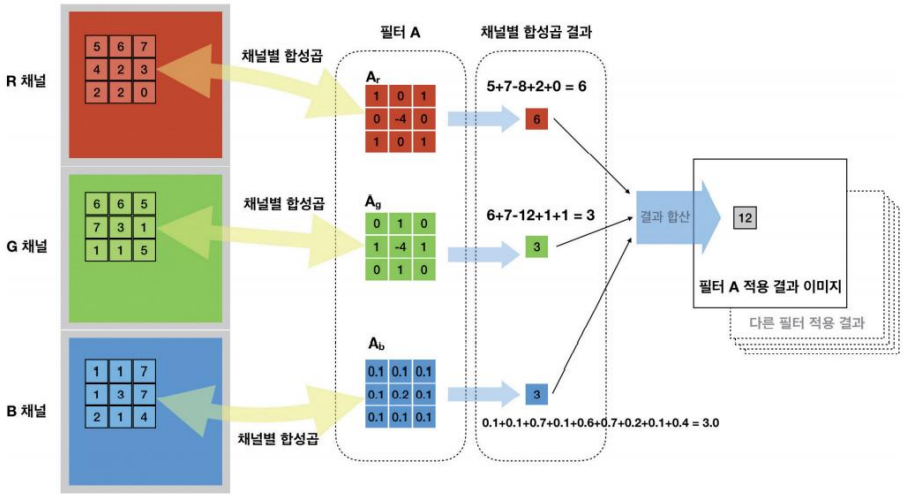


❖ 스트라이드stride

- Window(필터가 적용되는 영역)가 움직이는 보폭



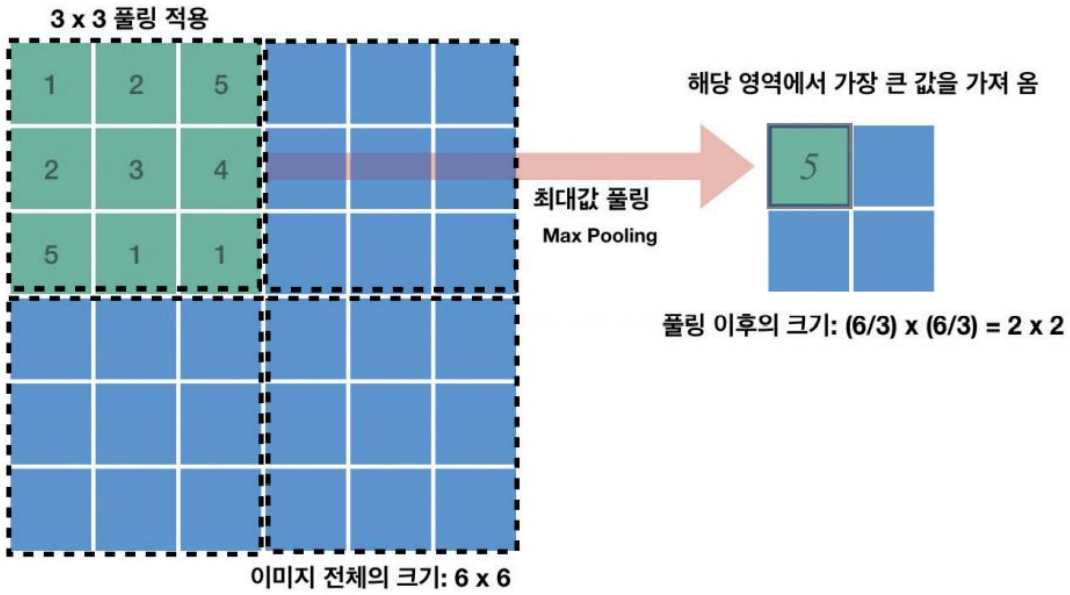
❖ 다중 채널multi-channel



강건한 모델을 만드는 풀링 (1)

❖ Pooling

- 이미지의 일정한 영역 내의 픽셀들이 가진 값을 하나로 축소하는 연산
- 이미지의 크기 $n \times m$, 풀링 영역의 크기: $n_p \times m_p$ ⇨ $\frac{n}{n_p} \times \frac{m}{m_p}$



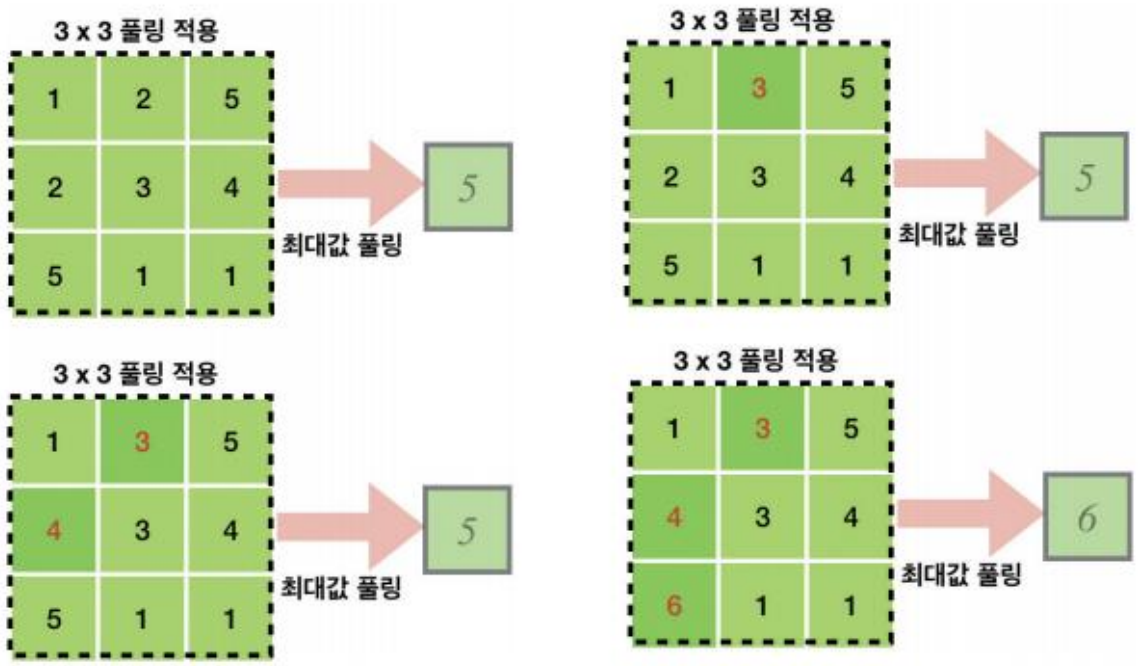
- 최대값 풀링max pooling
- 평균값 풀링mean pooling

- 다수의 픽셀 정보를 통합하여 하나로 만드는 것으로 원래 신호에 존재하는 잡음noise 요소를 제거 => 정보를 요약
- 미리 정해진 기능(최대값, 평균)을 수행하므로 학습 단계에서 파라미터를 최적화할 필요가 없음(원래 이미지가 가지고 있던 채널을 그대로 유지하면서 공간만 줄임)

강건한 모델을 만드는 풀링 (2)

❖ 이동 불변성translation invariance

- 풀링 과정에서 입력의 변화에 대해 덜 민감한 신호 전달(과적합 방지)
- 강건한robust 모델
- 과적합을 피할 수 있는 능력이 크다(모델의 복잡도가 줄어듦)



Program(1)

❖ LAB[9-1] 합성곱을 만들어보자

실습 목표

필터를 다양하게 생성해서 이미지와 합성곱을 구현해보자. 실습을 통해 합성곱 연산의 원리에 대해 깊이 이해해 보자.

힌트

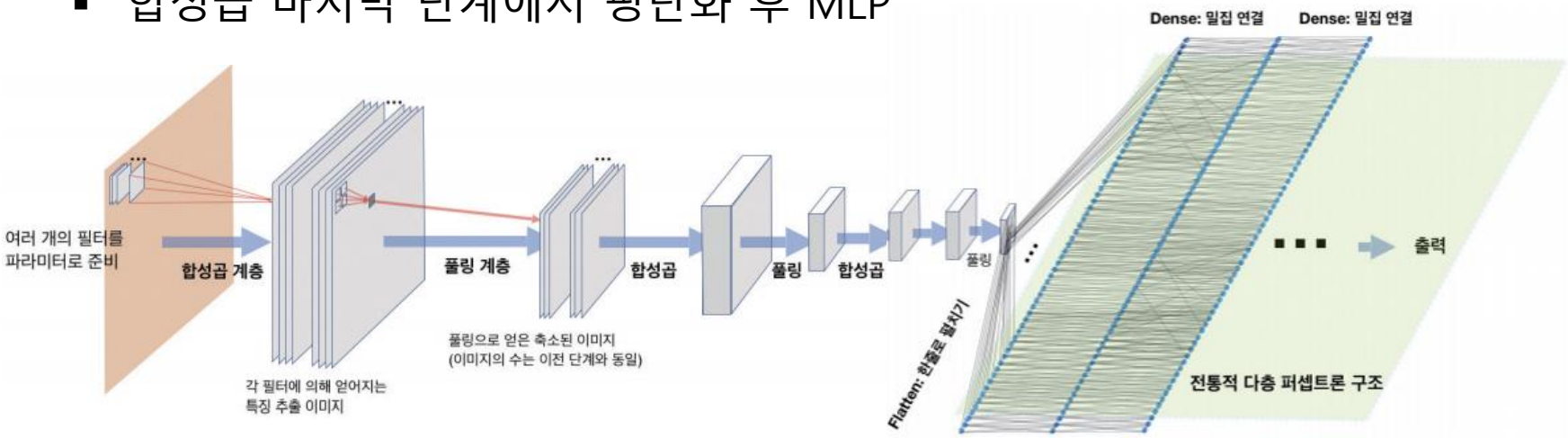
맷플롯립은 png 파일을 읽을 수 있는 기능을 제공한다. 이것을 이용하여 이미지를 읽으면 넘파이 배열 형태로 변환이 되는데, 넘파이 배열로 필터를 만들어 이 필터가 이미지를 표현하는 배열에 곱해지도록 한다. 이 책에서는 다양한 이미지 포맷을 다루거나 복잡한 이미지 처리를 하는 것이 목표가 아니므로 맷플롯립만을 사용하지만, 더욱 고급 이미지 처리를 원할 경우에는 Pillow나 OpenCV 패키지를 활용하도록 하라.

- https://colab.research.google.com/drive/14JBJrNtQSFQN51CT8V_GZIbGMrBTPisY
  - 구글드라이브 마운트

```
from google.colab import drive
drive.mount('/content/drive')
url = '/content/drive/MyDrive/images/david.png'
```

합성곱 신경망 모델의 구성

❖ 합성곱 신경망의 구성
- 학습 파라미터: 합성곱 필터의 가중치
- 풀링은 정보를 줄이기만 할 뿐 학습 가능한 파라미터가 없음
- 파라미터 계산: 커널크기 x #입력채널 x #출력채널 + 출력채널_bias
- 합성곱 마지막 단계에서 평탄화 후 MLP



❖ Keras API

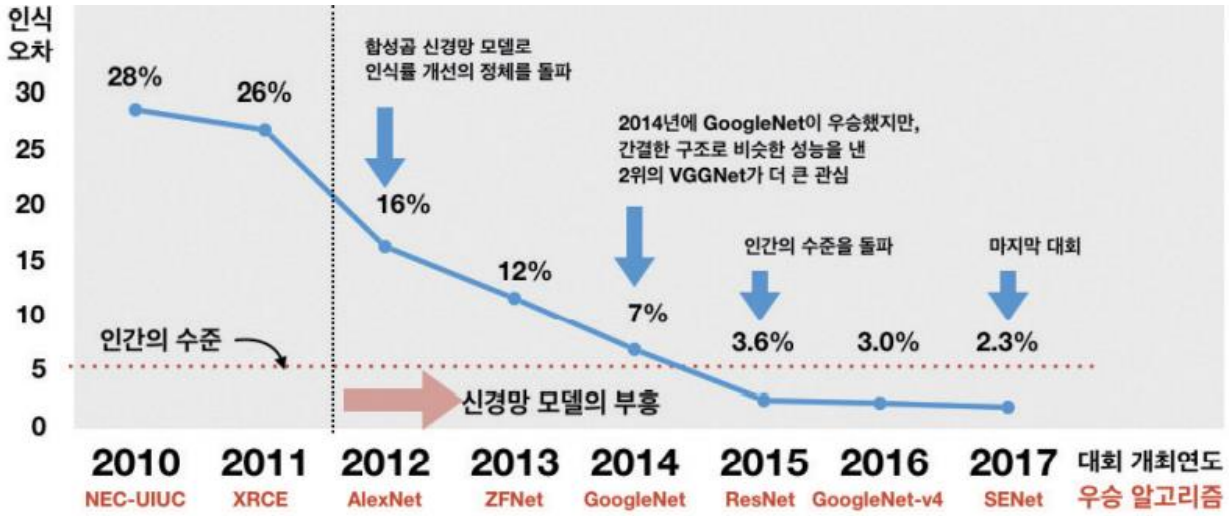
```
model = keras.models.Sequential( [
    keras.layers.Conv2D(input_shape = (64, 64, 3),
                        kernel_size = (3,3), filters = 32),
    keras.layers.MaxPooling2D((2, 2), strides=2),
    keras.layers.Conv2D(kernel_size = (3,3), padding='same' filters = 64),
    ... ])
```

- padding='valid': 패딩 않음
- padding='same': 패딩 수행

합성곱 신경망 모델의 성공 (1)

❖ ImageNet 데이터베이스

- 스탠퍼드 대학교의 **페이페이리**Fei-Fei Li 교수
- 2만2천개의 범주로 구분된 1,500만장의 방대한 이미지
- ILSVRC: 이미지넷 대규모 시각 인지 경진대회ImageNet large scale visual recognition challenge



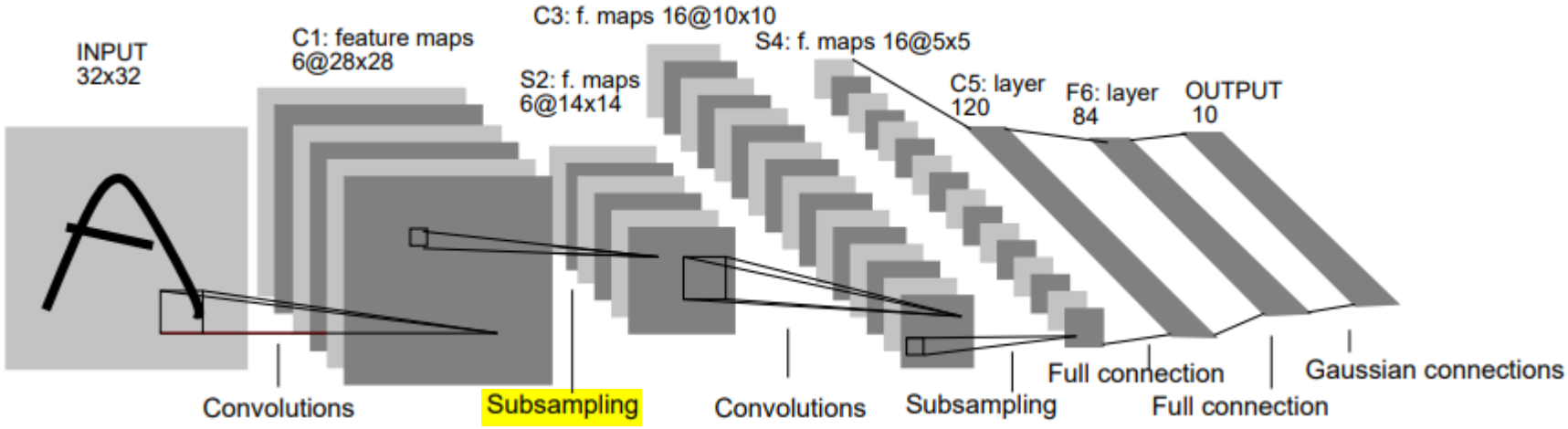
합성곱 신경망 모델의 성공 (2)

❖ 합성곱 신경망의 성공 요인

- **희소 상호작용**sparse interaction
  - 국소적인 영역에 작은 필터를 적용하여 특징을 추출
  - 입력된 이미지의 모든 픽셀들 사이의 관계를 다룰 필요가 없어 입력 신호들 사이의 연결 개수를 크게 줄일 수 있음
- **파라미터 공유**parameter sharing
  - 합성곱 필터는 많은 수의 데이터를 처리하여도 작은 크기로 유지될 수 있고, 파라미터의 수도 제한적임
  - 파라미터 공유는 연산량 감소에 큰 역할을 하고, 더 깊은 층을 쌓을 수 있도록 함
- **등변성 표현**equivariant representation
  - **등변 함수**equivariant function : $f(g(x)) = g(f(x))$의 성질을 갖는 함수 f
  - 등변성은 어떤 함수의 입력에 특정한 변경을 적용하면 출력도 같은 방식으로 변하는 것을 의미.
  - 합성곱 신경망은 이미지의 일부가 이동하면, 그 결과가 같은 방식으로 이동하여 나타남. 외곽선 추출과 같은 국소적인 함수가 어디서나 효과를 발휘한다는 것을 의미.
  - 이미지의 크기 변화나 회전 등에 대해서는 합성곱 연산이 등변성을 보이지 못함

다양한 CNN Architectures (1)

❖ LeNet-5

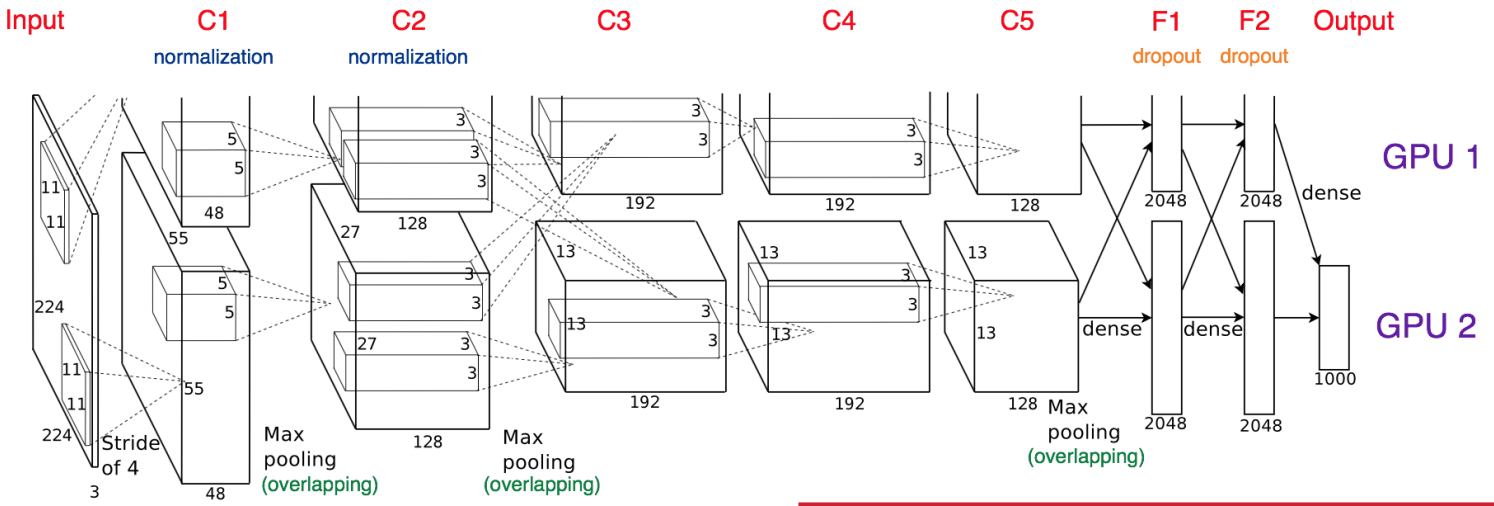


- Yann LeCun in 1998
- handwritten digit recognition

Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully Connected	–	10	–	–	RBF
F6	Fully Connected	–	84	–	–	tanh
C5	Convolution	120	1 × 1	5 × 5	1	tanh
S4	Avg Pooling	16	5 × 5	2 × 2	2	tanh
C3	Convolution	16	10 × 10	5 × 5	1	tanh
S2	Avg Pooling	6	14 × 14	2 × 2	2	tanh
C1	Convolution	6	28 × 28	5 × 5	1	tanh
In	Input	1	32 × 32	–	–	–

다양한 CNN Architectures (2)

❖ AlexNet

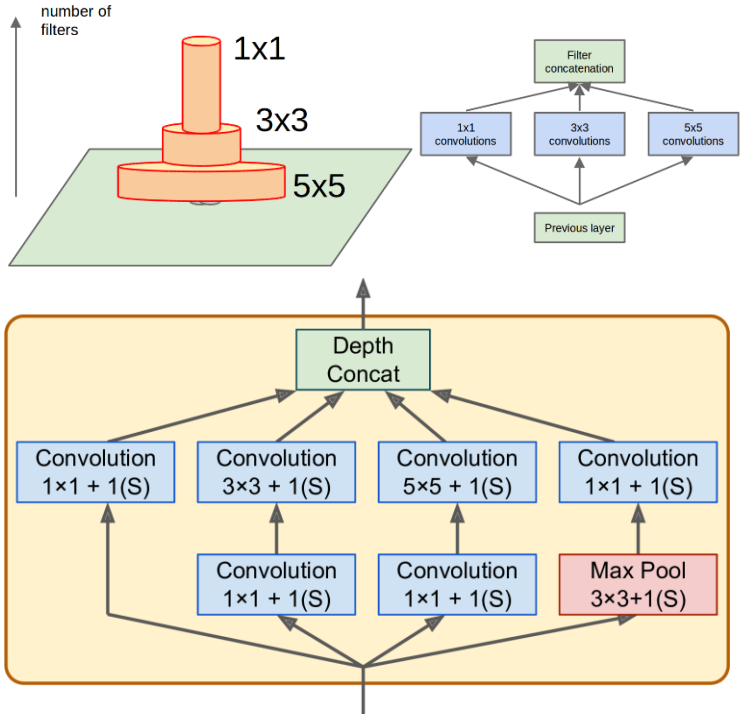


- Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton
- achieved 17% top-5 error rate in ImageNet ILSVRC challenge in 2012
- apply dropout with a 50% dropout rate.
- *data augmentation*
- *local response normalization*

Layer	Type	Maps	Size	Kernel size	Stride	Padding	Activation
Out	Fully Connected	–	1,000	–	–	–	Softmax
F9	Fully Connected	–	4,096	–	–	–	ReLU
F8	Fully Connected	–	4,096	–	–	–	ReLU
C7	Convolution	256	13 × 13	3 × 3	1	SAME	ReLU
C6	Convolution	384	13 × 13	3 × 3	1	SAME	ReLU
C5	Convolution	384	13 × 13	3 × 3	1	SAME	ReLU
S4	Max Pooling	256	13 × 13	3 × 3	2	VALID	–
C3	Convolution	256	27 × 27	5 × 5	1	SAME	ReLU
S2	Max Pooling	96	27 × 27	3 × 3	2	VALID	–
C1	Convolution	96	55 × 55	11 × 11	4	VALID	ReLU
In	Input	3 (RGB)	227 × 227	–	–	–	–

다양한 CNN Architectures (3)

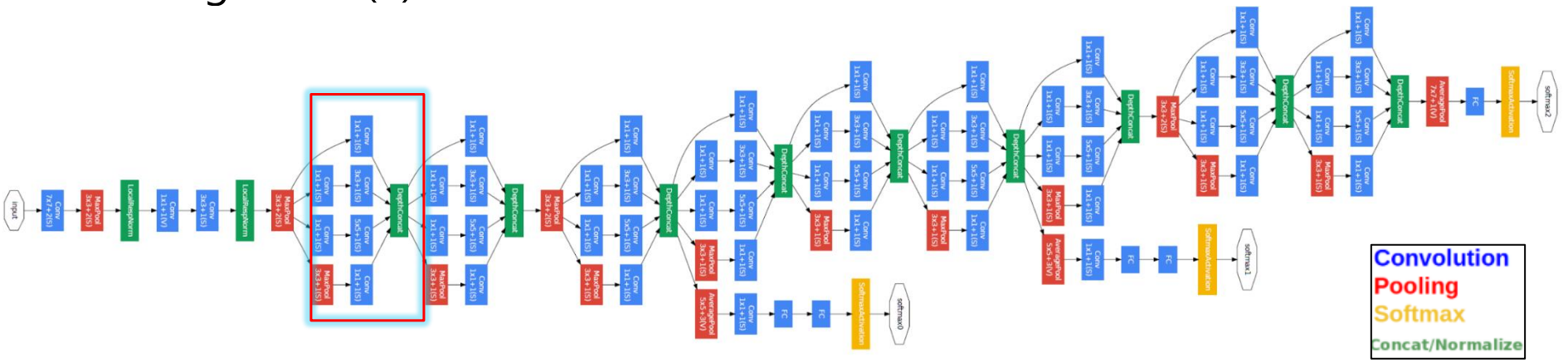
- GoogLeNet (1)
  - Christian Szegedy et al in 2014
  - *inception modules*
    - $3 \times 3 + 1(S)$: 3 × 3 kernel, stride 1, and SAME padding
    - every single layer uses a stride of 1 and SAME padding, so their outputs all have the same height and width as their inputs.
    - This makes it possible to concatenate all the outputs along the depth dimension in the final *depth concatenation layer.*



  - 1×1 *kernel*
    - can capture patterns along the depth dimension.
    - serve as *bottleneck layers,* meaning they reduce dimensionality. This cuts the computational cost and the number of parameters, speeding up training and improving generalization.
    - each pair of convolutional layers ([1×1, 3×3] and [1×1, 5×5]) acts like a single, capable of capturing more complex patterns.

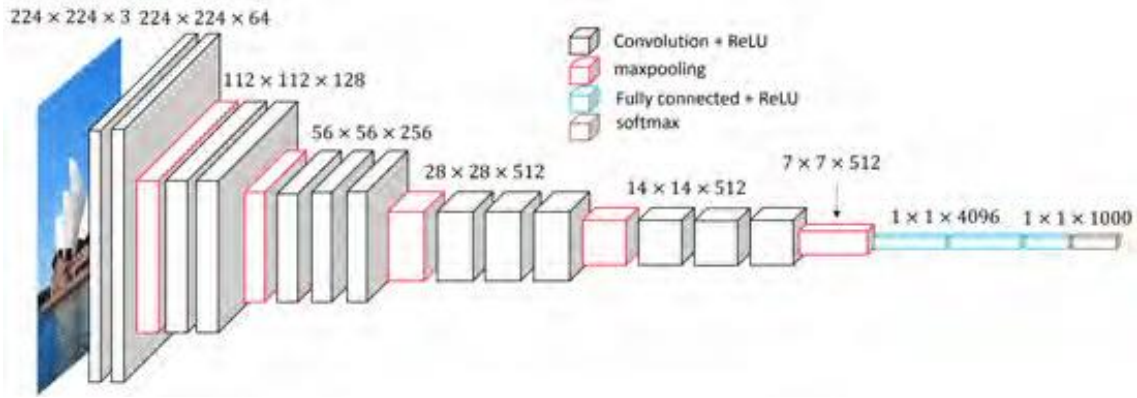
다양한 CNN Architectures (4)

❖ GoogLeNet (2)



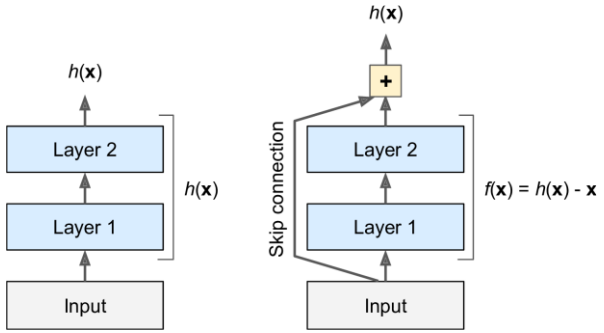
❖ VGGNet

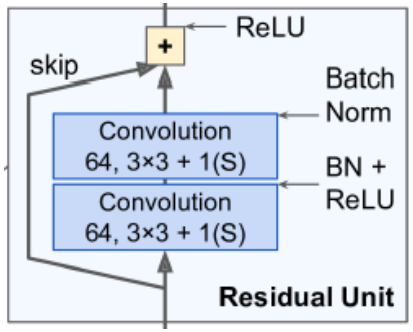
- K. Simonyan and A. Zisserman in 2014
- 2-3 convolutional layer and pooling
- 3 × 3 filter

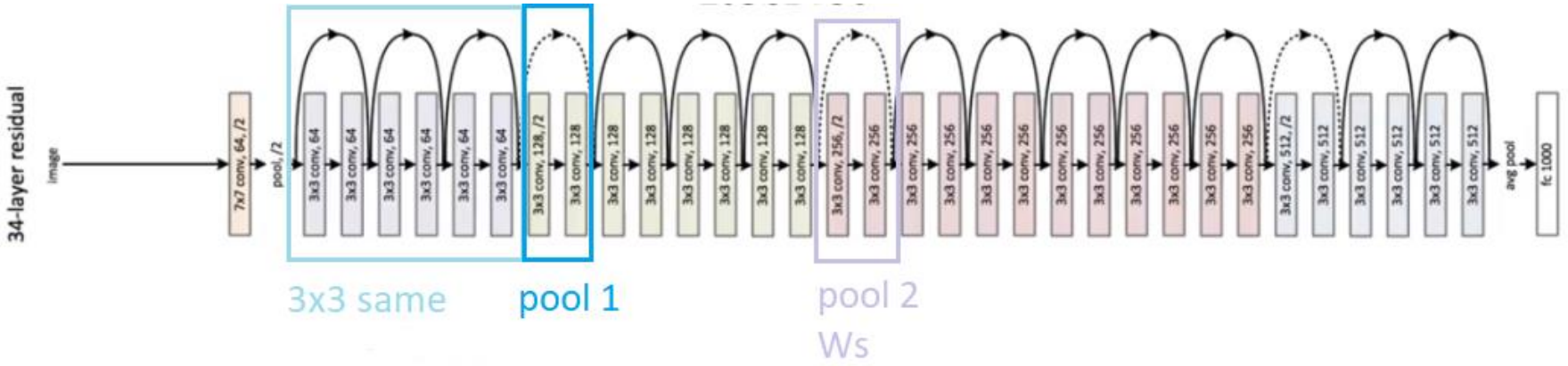


다양한 CNN Architectures (5)

- ResNet
  - Kaiming He et al. in 2015
  - 152 layers
  - Skip(shortcut) connection
    - the resulting network just outputs a copy of its inputs.
    - the network can start making progress even if several layers have not started learning yet.
    - the signal can easily make its way across the whole network.
  - Residual Unit



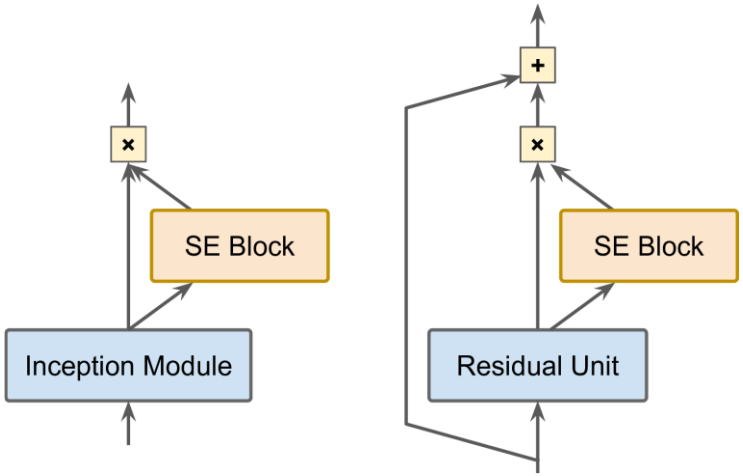


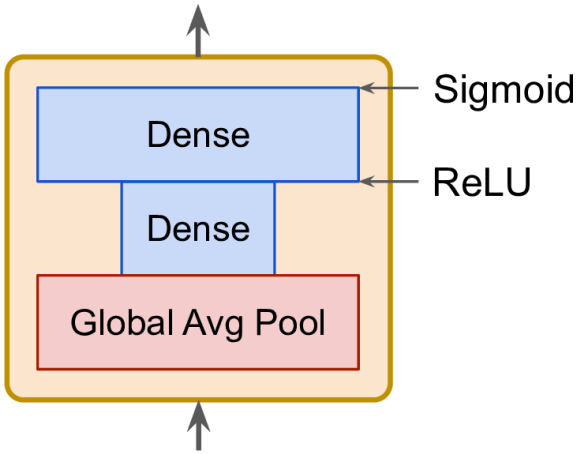


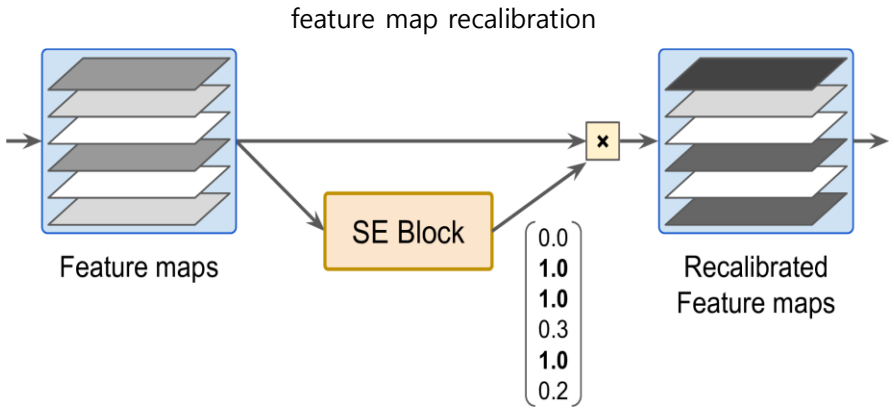
다양한 CNN Architectures (6)

❖ SENet

- Squeeze-and-Excitation Network
- *SE-Inception* extends inception networks
- *SE-ResNet* extends ResNet
- An SE block analyzes the output of the unit it is attached to, focusing exclusively on the depth dimension, and it learns **which features are usually most active together**.
- It then uses this information to recalibrate the feature maps.
- SE block architecture







Program(2): 패션 MNIST 분류(1)

❖ **LAB8-1** MLP 이용

- https://colab.research.google.com/drive/1bmzbi5JC-r3AYRp4SfdotyG6t8SzrDAX

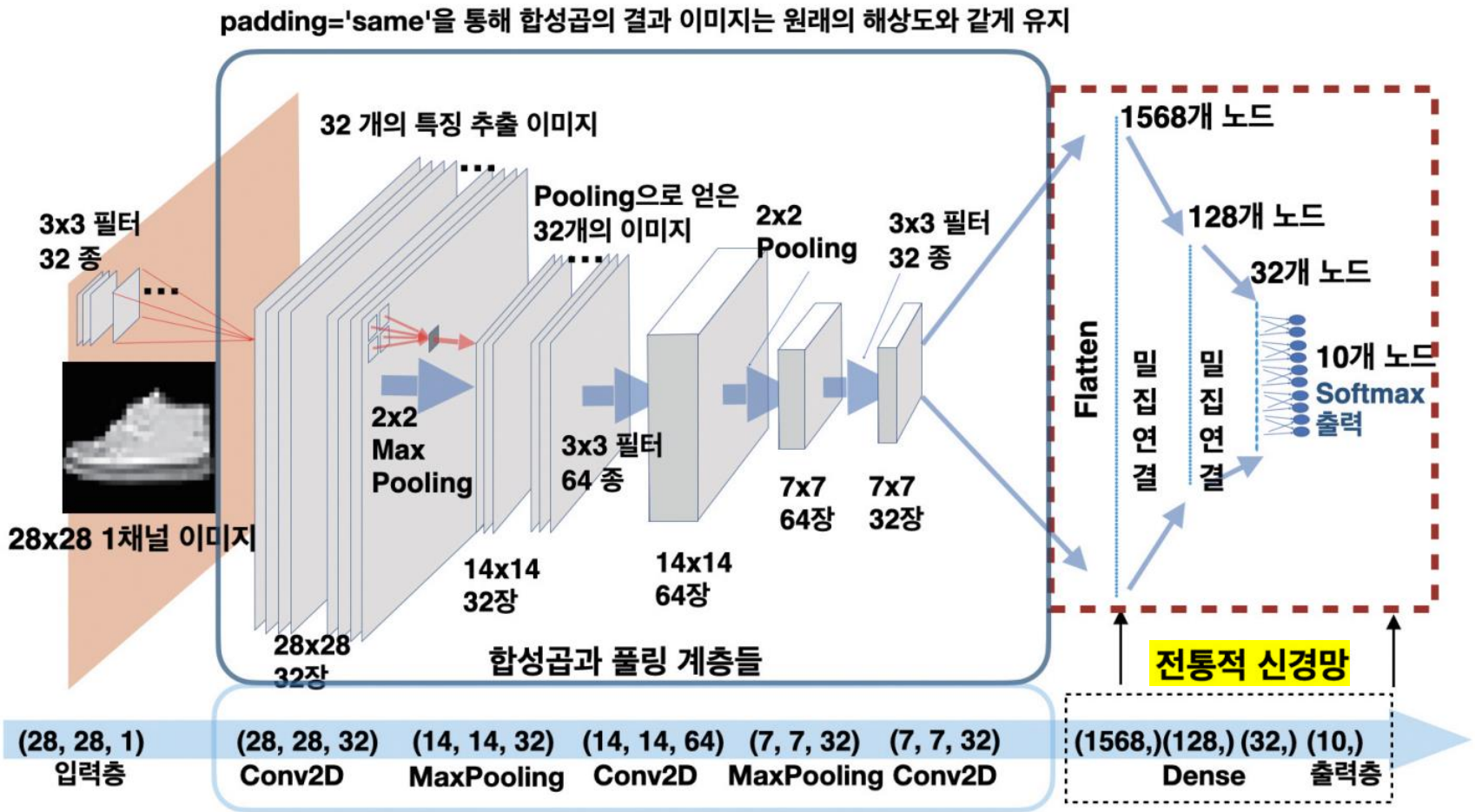
```
model = keras.models.Sequential( [
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(128, activation= 'relu'),
    keras.layers.Dense(32, activation= 'relu'),
    keras.layers.Dense(10, activation= 'softmax'),
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
history = model.fit(train_images, train_labels, batch_size=64,
                    epochs=10, validation_split=0.25)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dropout (Dropout)           (None, 784)               0

 dense (Dense)               (None, 128)               100480

 dense_1 (Dense)             (None, 32)                4128

 dense_2 (Dense)             (None, 10)                330

=================================================================
Total params: 104,938
Trainable params: 104,938
Non-trainable params: 0
```

Program(2): 패션 MNIST 분류(2)

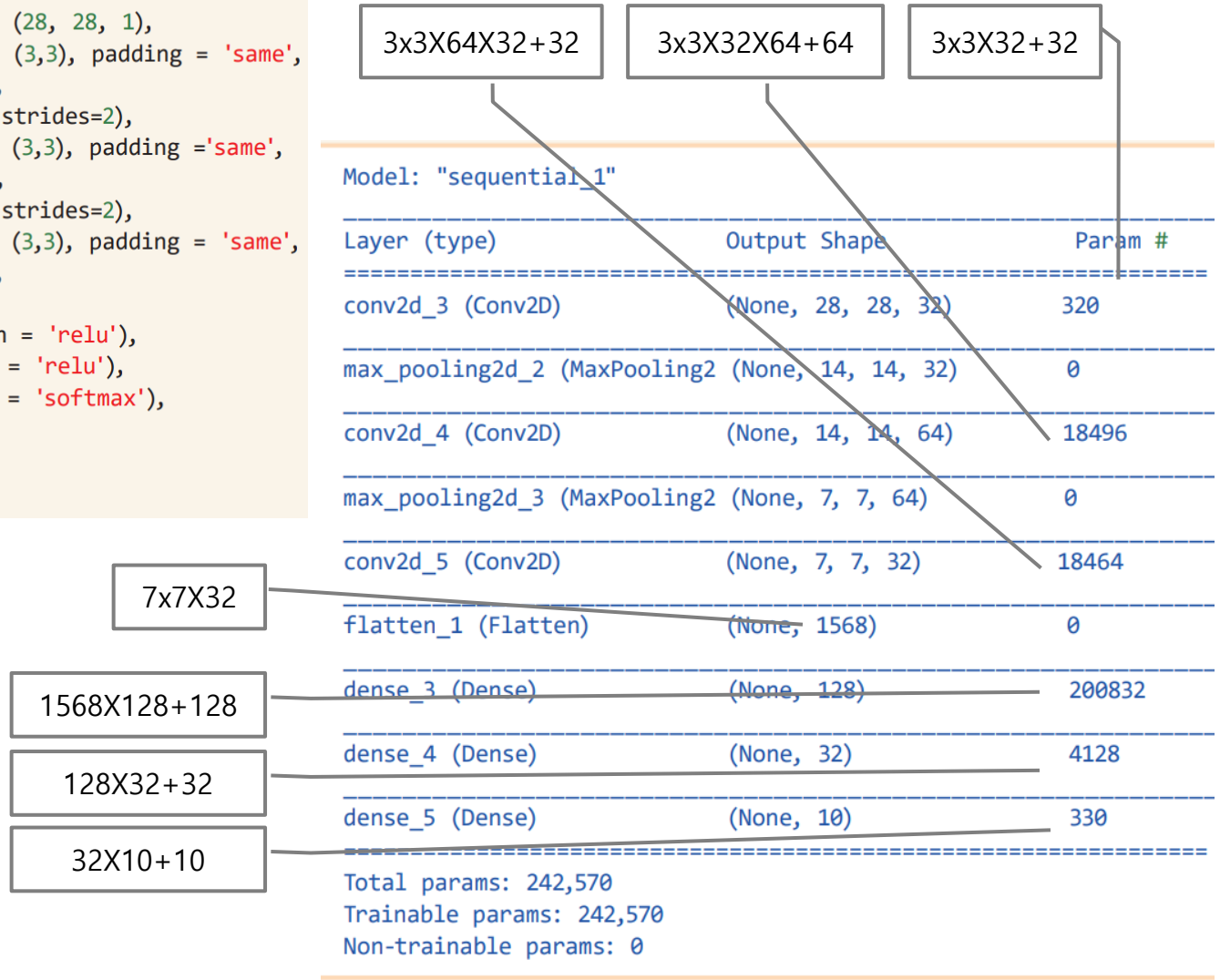
❖ LAB9-2 합성곱 이용

- https://colab.research.google.com/drive/14JBJrNtQSFQN51CT8V_GZIbGMrBTPisY



Program(2): 패션 MNIST 분류(3)

```
model = keras.models.Sequential( [
    keras.layers.Conv2D(input_shape = (28, 28, 1),
                        kernel_size = (3,3), padding = 'same',
                        filters = 32),
    keras.layers.MaxPooling2D((2, 2), strides=2),
    keras.layers.Conv2D(kernel_size = (3,3), padding ='same',
                        filters = 64),
    keras.layers.MaxPooling2D((2, 2), strides=2),
    keras.layers.Conv2D(kernel_size = (3,3), padding = 'same',
                        filters = 32),
    keras.layers.Flatten(),
    keras.layers.Dense(128, activation = 'relu'),
    keras.layers.Dense(32, activation = 'relu'),
    keras.layers.Dense(10, activation = 'softmax'),
])

model.summary()
```

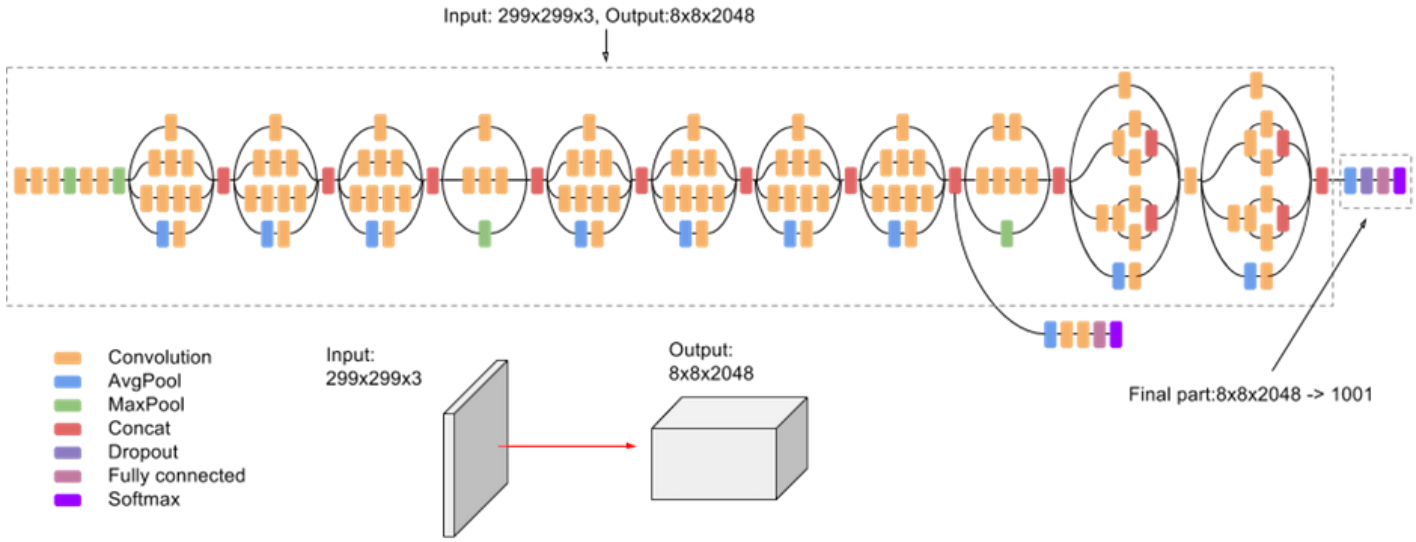



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 28, 28, 32)        320
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 14, 14, 64)        18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 7, 7, 32)          18464
_________________________________________________________________
flatten_1 (Flatten)          (None, 1568)              0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               200832
_________________________________________________________________
dense_4 (Dense)              (None, 32)                4128
_________________________________________________________________
dense_5 (Dense)              (None, 10)                330
=================================================================
Total params: 242,570
Trainable params: 242,570
Non-trainable params: 0
```

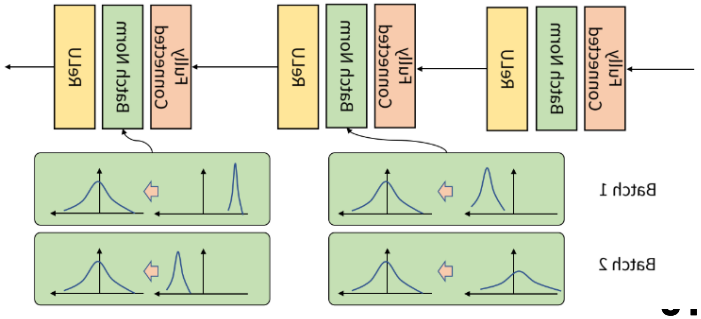
Program(3)

❖ LAB[9-3] 성능 좋은 구글 Inception V3 이용

- Inception v3 고급 가이드 | Cloud TPU
- Google Inception Model.



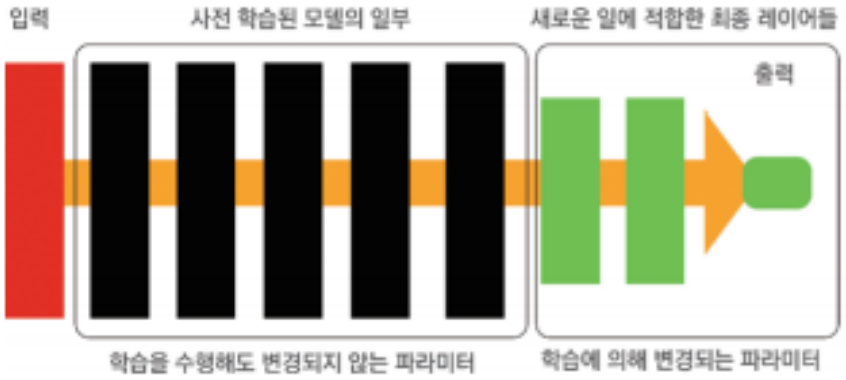
- 배치 정규화(Batch Normalization) - gaussian37
  - 학습 과정에서 각 배치 단위 별로 데이터가 다양한 분포를 가지더라도 **각 배치별로 평균과 분산을 이용해 정규화**하는 것

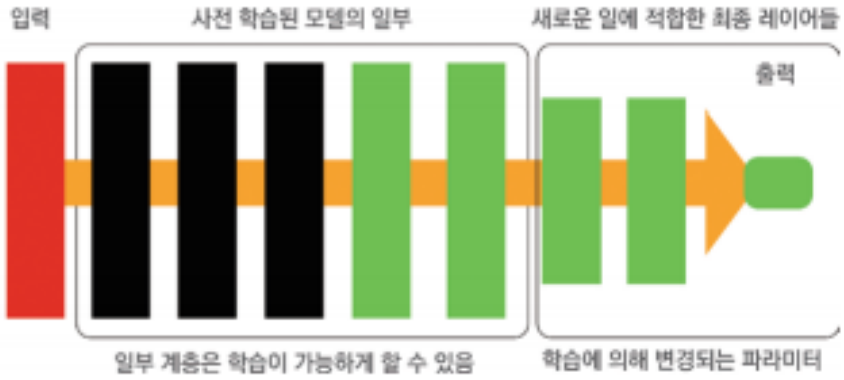


전이학습

❖ 전이학습transfer learning

- 이미 학습이 완료된 모델을 다른 목적에 맞춰 조금만 고쳐서 사용하는 것
- 이미 학습된 모델은 특징 추출 용도로 사용하고, 이 특징 추출기의 출력을 이용하여 다른 데이터에 대한 작업을 수행하는 구조
- 사전 학습된 모델의 일부 계층을 잘라서 가져오고, 이 계층들이 가지고 있는 파라미터들은 변경되지 않도록 만들어야 함





❖ https://huggingface.co/

Mini Project

❖ 얼굴찾기: SVM로 분류하기

목표

SVM을 이용하여 주어진 이미지가 사람의 얼굴인지 아닌지를 구분하는 프로그램을 만들어 보자.

- https://colab.research.google.com/drive/1UrZjeFCtwcLHqefXhZBa2mzeltx9_SSi

❖ 얼굴찾기: CNN 활용하기

목표

SVM을 이용했던 6장 미니 프로젝트 B1의 결과를 개선하기 위해 CNN을 적용해 보자. SVM은 입력을 바로 처리하기가 힘들어 HOG, 즉 이미지 각 픽셀의 기울기를 구해 그 히스토그램을 분석을 위한 특징으로 활용했는데, CNN은 이러한 과정 없이 스스로 특징을 찾아낼 것이다.

❖ 얼굴찾기: 전이학습 활용하기

목표

SVM을 이용했던 미니 프로젝트 B1의 결과를 개선하기 위해 미리 만들어 놓은 CNN을 적용해 보자. 텐서플로우는 이미검증된 훌륭한 CNN 모델을 함께 제공하고 있다. 우리는 구글에서 만든 Inception V3라는 뛰어난 CNN을 모델과 옥스퍼드 대학의 연구자들이 만든 VGG16을 사용할 것이다.

- https://colab.research.google.com/drive/14JBJrNtQSFQN51CT8V_GZIbGMrBTPisY