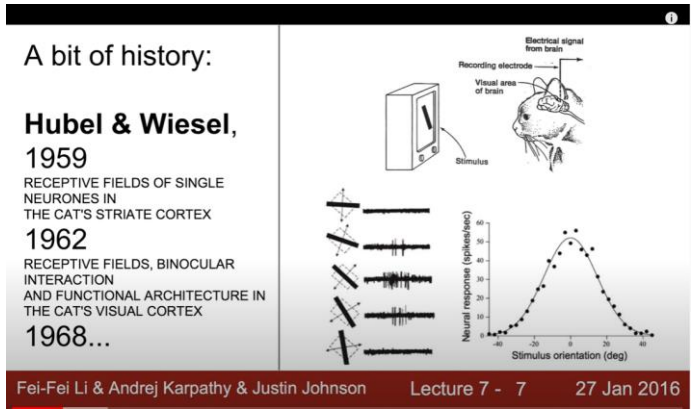
Convolutional Neural Network (CNN)

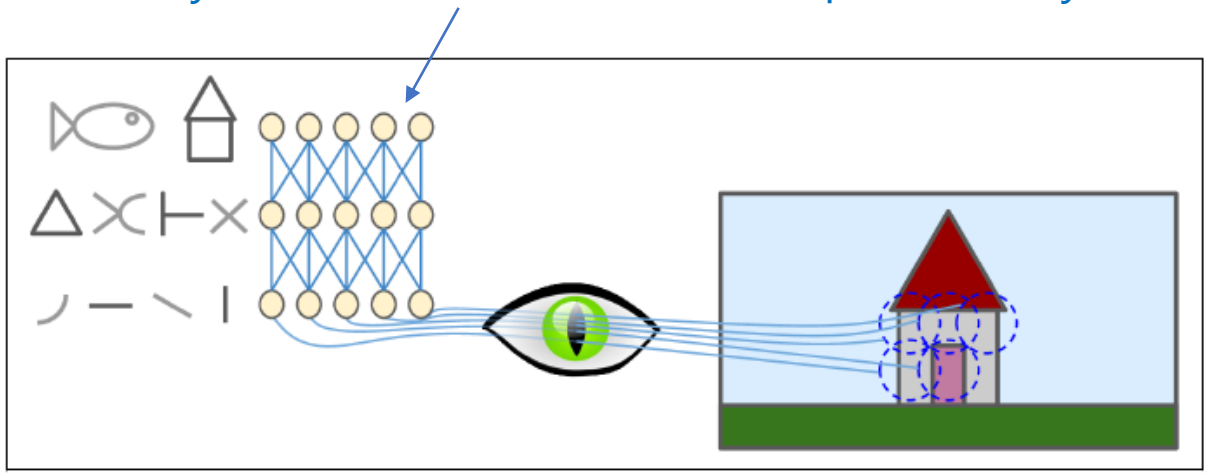
2022. 8

Yongjin Jeong, KwangWoon University

Motivation

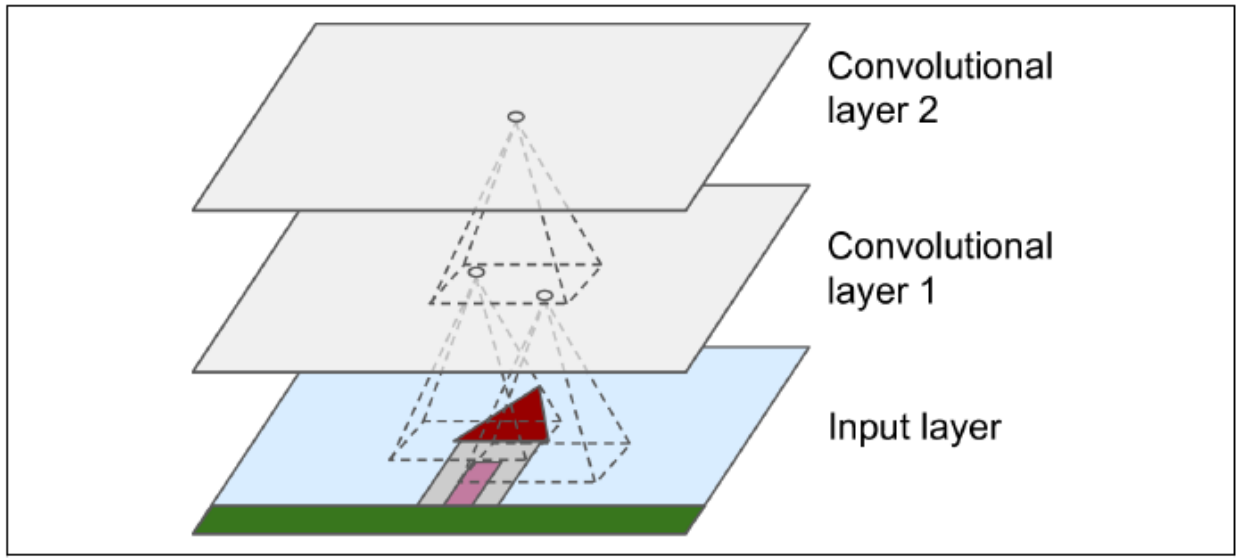
- **Hubel and Wiesel observed that:**
  - Many neurons react only to visual stimuli located in a limited region.
  - Some neurons react only to specific patterns (horizontal lines, lines with different orientations (two neurons may have the same receptive field but react to different line orientations).
  - Some neurons have larger receptive fields, and they react to more complex patterns that are combinations of the lower-level patterns.
- **These observation led to the Idea:**
  - Higher-level neurons are based on the outputs of neighboring lower-level neurons.
  - Notice that each neuron is connected only to a few neurons from the previous layer.





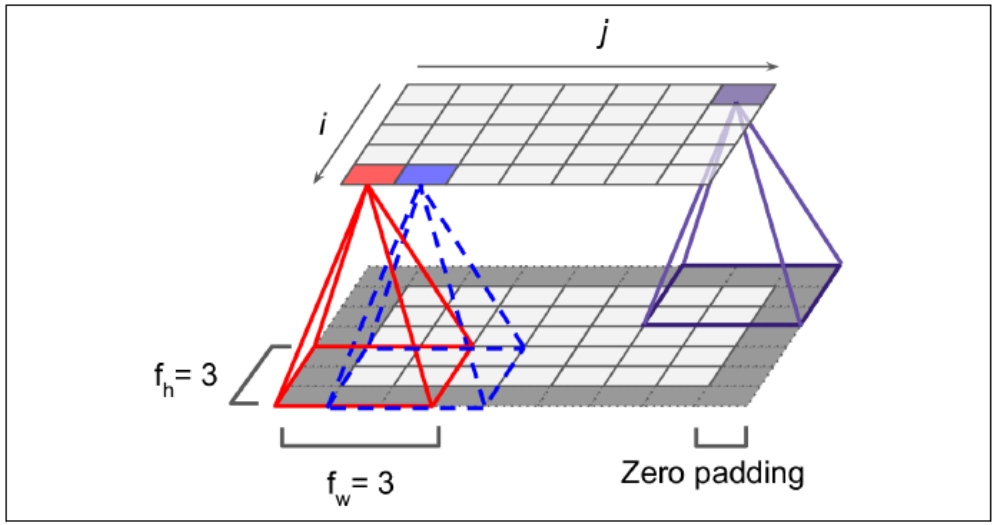
Convolutional Layer

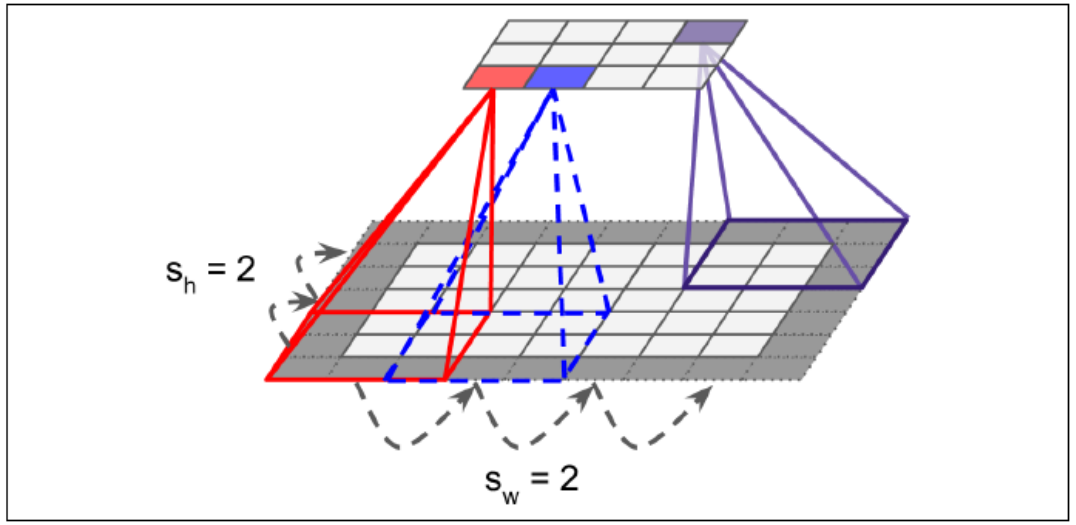
- **The most important building block of CNN**
  - Neurons are connected only to their receptive fields of the previous layer
  - Allowing the network to concentrate on small low-level features
  - Then, assembled into larger higher-level features in the next layer
  - This hierarchical architecture is very common in real-world images.



Convolutional Layer

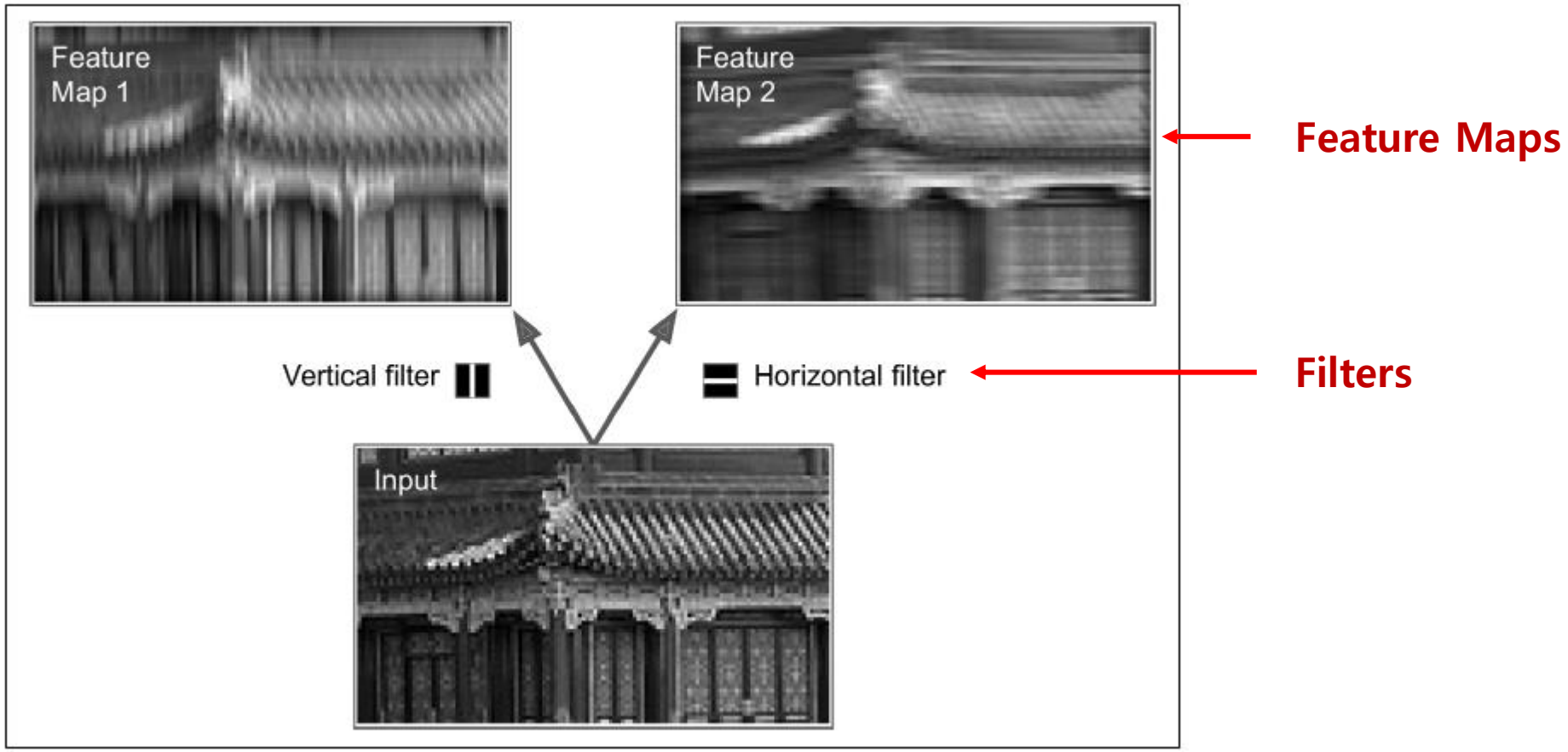
- **Connection between layers and zero padding**
- **Reducing dimensionality using Stride of 2**





Filters

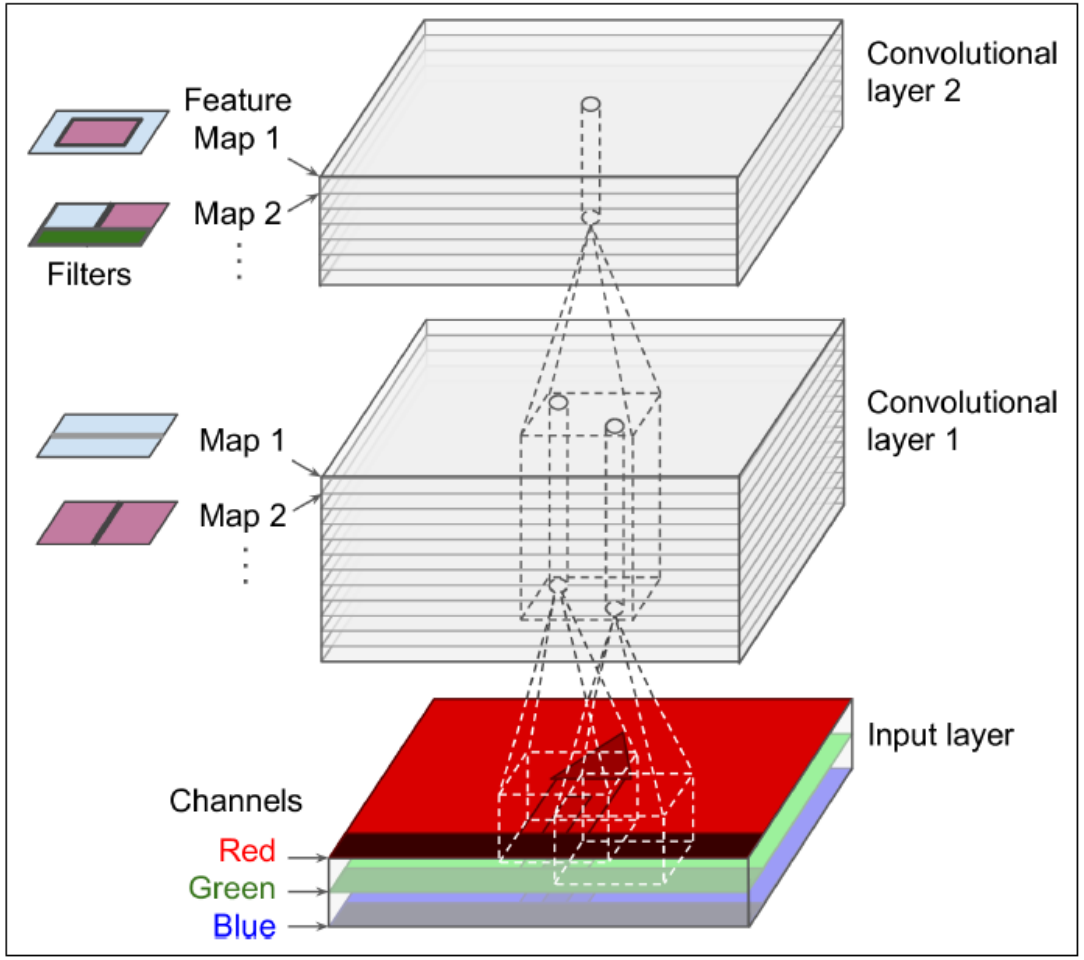
- **Filters and Feature Maps**



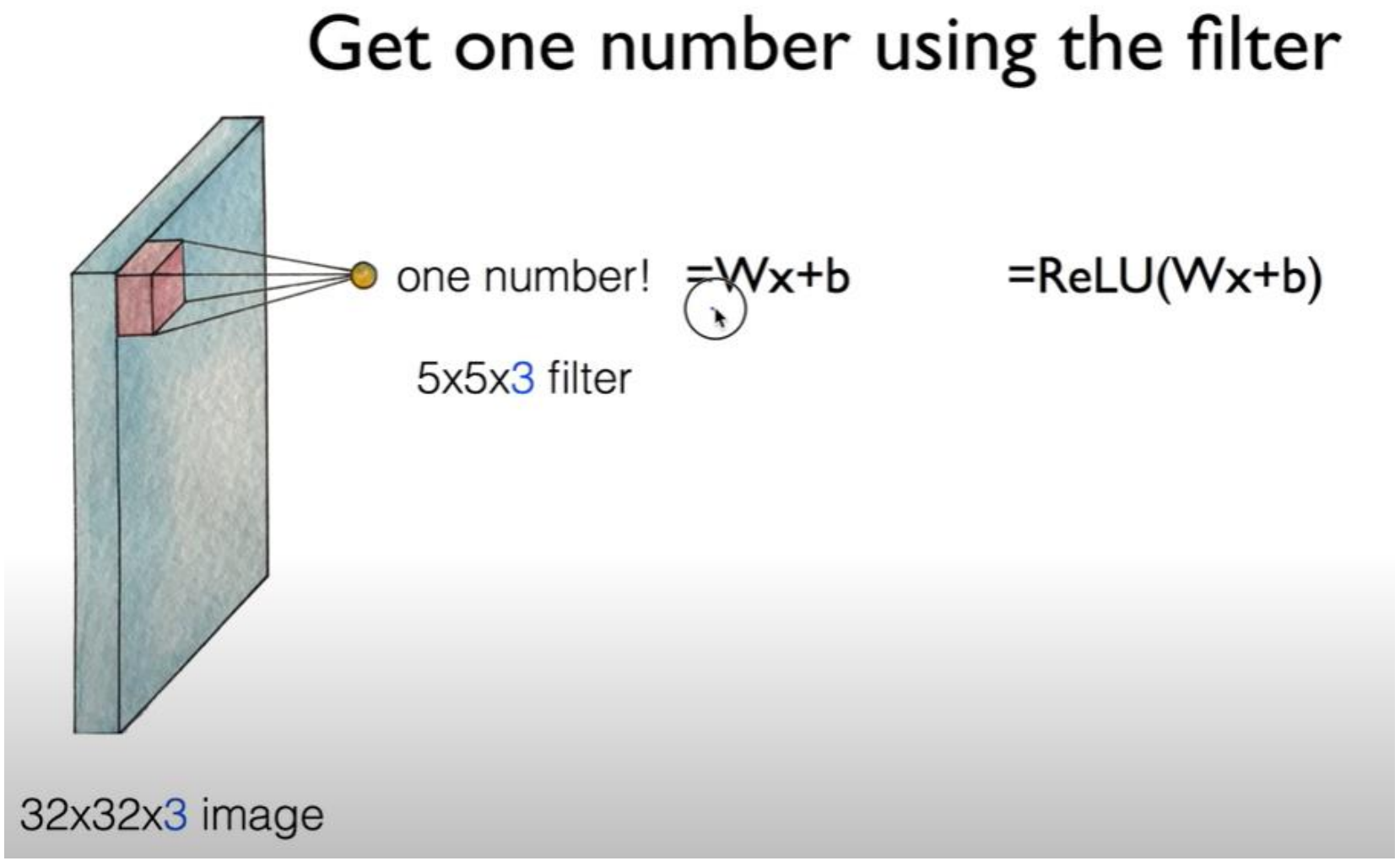
Applying two different filters to get two feature maps

Filters

- **Stacking Multiple Feature Maps**

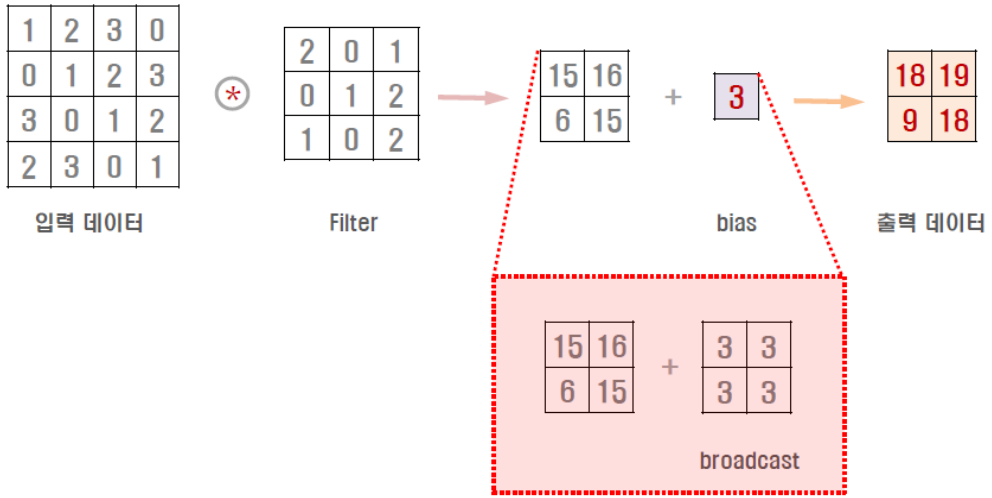


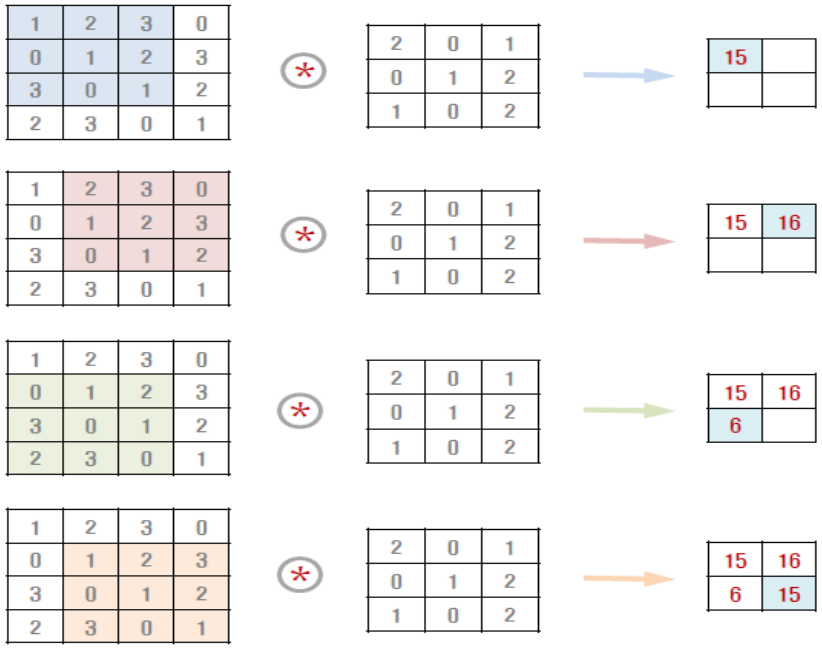
CNN operation



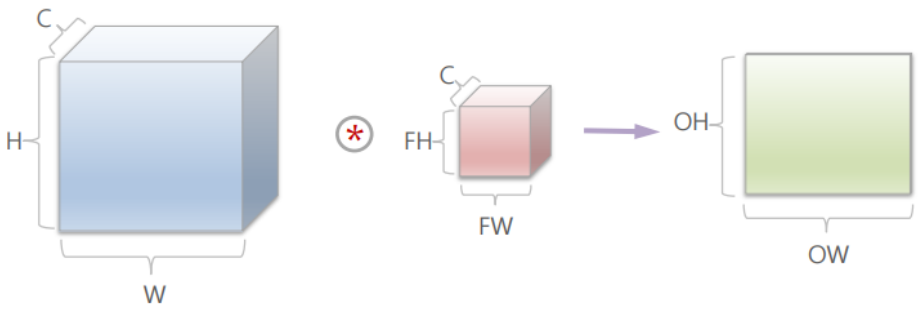
Convolution

입력 데이터(height, width)에 대해 **필터(커널)**을
일정 간격(**Stride**) 만큼 이동해 가며 행렬 곱셈 연산 수행





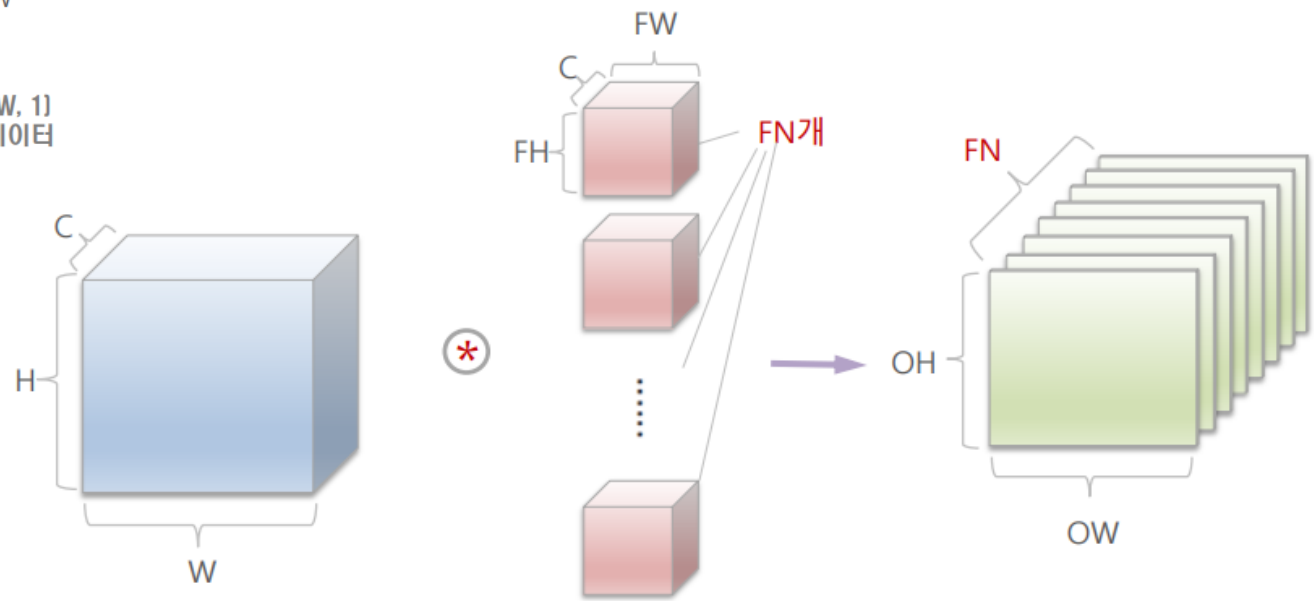
3-dimensional data convolution



(H, W, C)
입력 데이터

(FH, FW, C)
필터

(OH, OW, 1)
출력 데이터

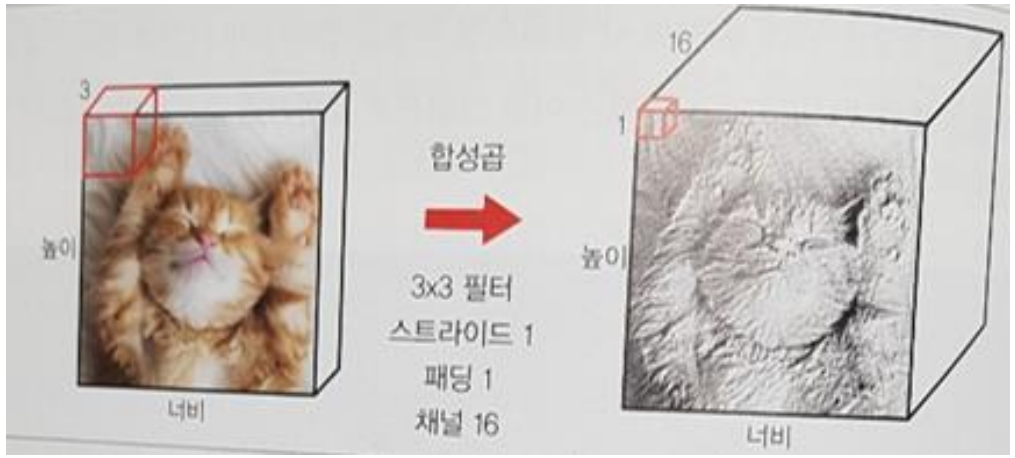


(H, W, C)
입력 데이터

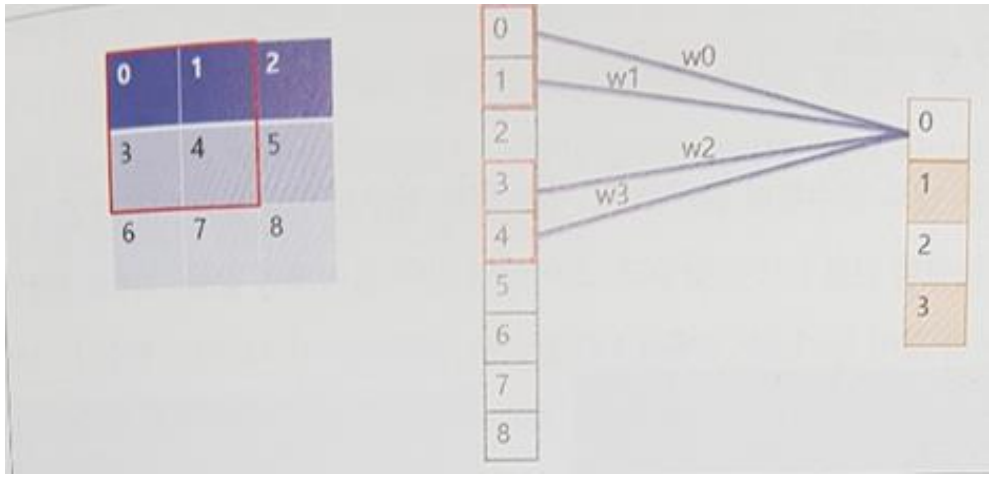
(FH, FW, C, **FN**)
필터

(OH, OW, **FN**)
출력 데이터

Different views of Convolution



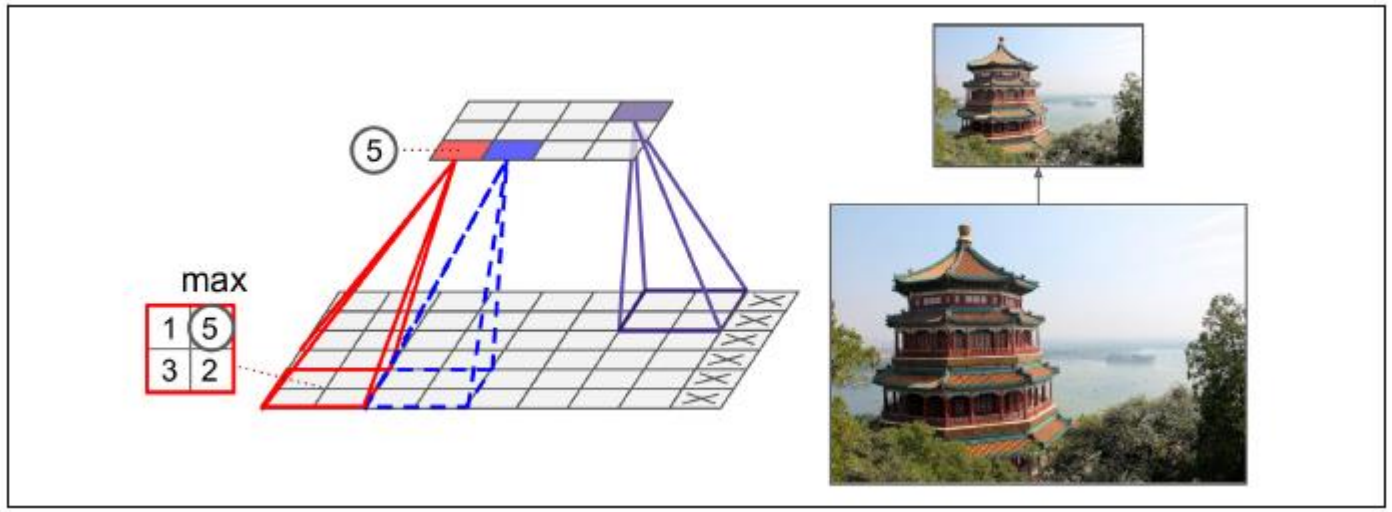
Convolution 층 연산의 3차원적 이해



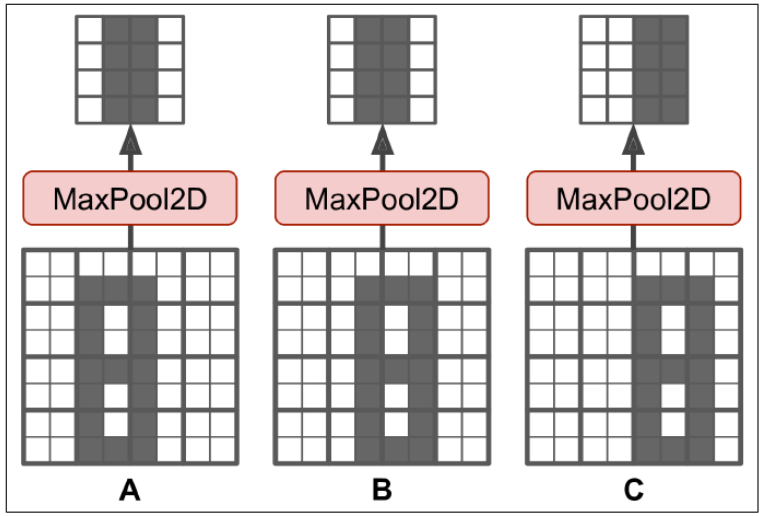
NLP 형태로 본 Convolution 층 연산

Pooling Layer

- **Max Pooling layer**
  - The goal is to subsample (i.e. shrink) the input image (to reduce computational load, the memory usage, and the number of parameters) limiting the risk of overfitting
  - Noise suppression
  - Makes it invariant to translation movement (shifting or rotational) to some extent
  - Helps capture essential structural features of the represented images



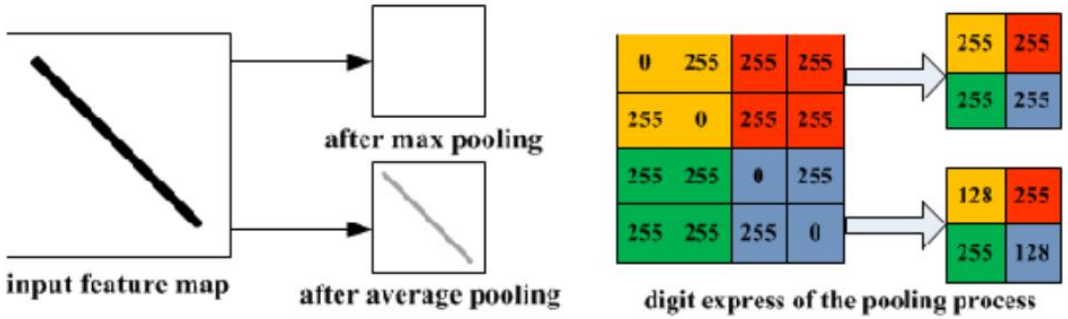
Max Pooling layer (2 x 2 pooling kernel, stride 2, no padding)

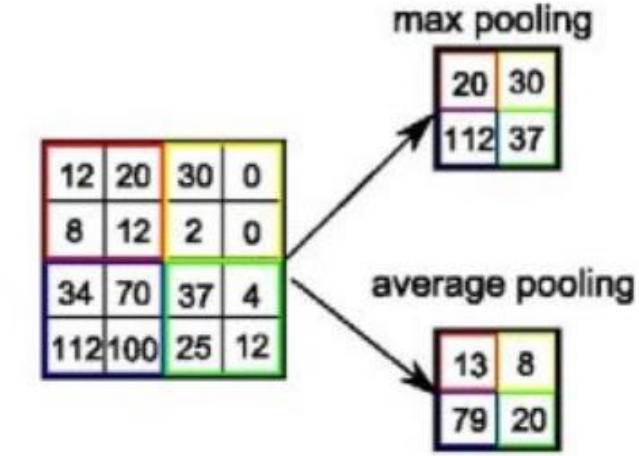


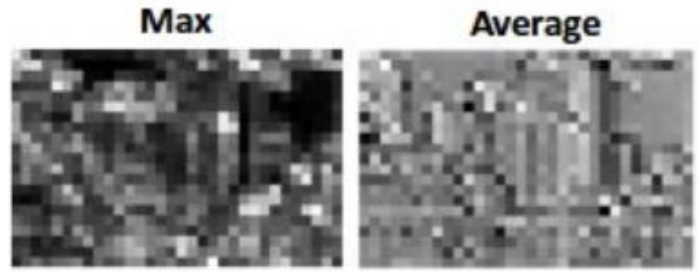
Invariance to small translations

Pooling Layer

- **Maxpooling and AvgPooling**
  - data dependent, but in general Maxpooling is preferred







- Max Pooling extracts the most important features like edges.

https://www.quora.com/What-is-the-benefit-of-using-average-pooling-rather-than-max-pooling

Implementing CNN

- 컨볼루션 연산
  - 컨볼루션은 해당하는 요소끼리 곱하고 결과를 모두 더하는 선형 연산
  - 식 (4.10)과 식 (4.11)에서 u는 커널, z는 입력, s는 출력(특징 맵)

$$s(i) = z \circledast u = \sum_{x=-(h-1)/2}^{(h-1)/2} z(i+x)u(x) \qquad (4.10)$$

← 1차원 입력

$$s(j,i) = z \circledast u = \sum_{y=-(h-1)/2}^{(h-1)/2} \sum_{x=-(h-1)/2}^{(h-1)/2} z(j+y, i+x)u(y,x) \qquad (4.11)$$

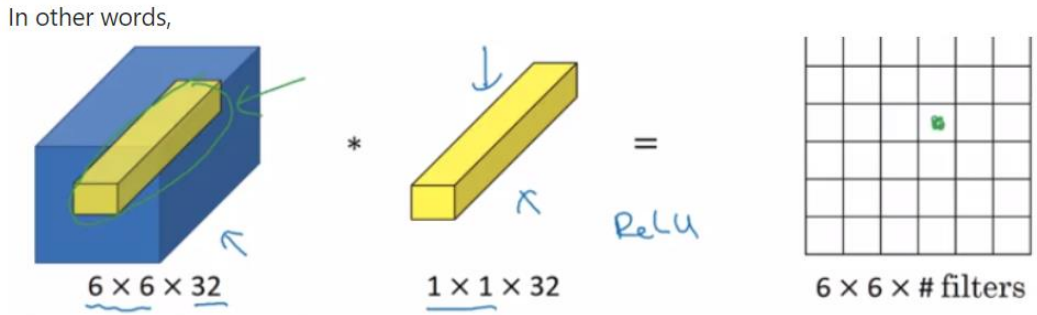
← 2차원 입력

(a) 1차원 컨볼루션 (b) 2차원 컨볼루션

Ref: 오일석, 기계학습

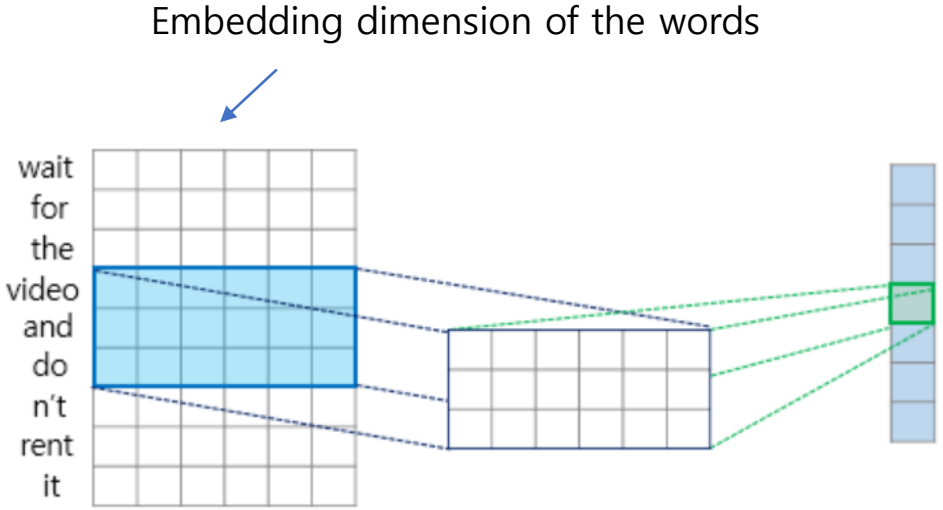
Implementing CNN

- **1-D convolutions in Image Processing (1x1 convolution: pointwise convolution)**



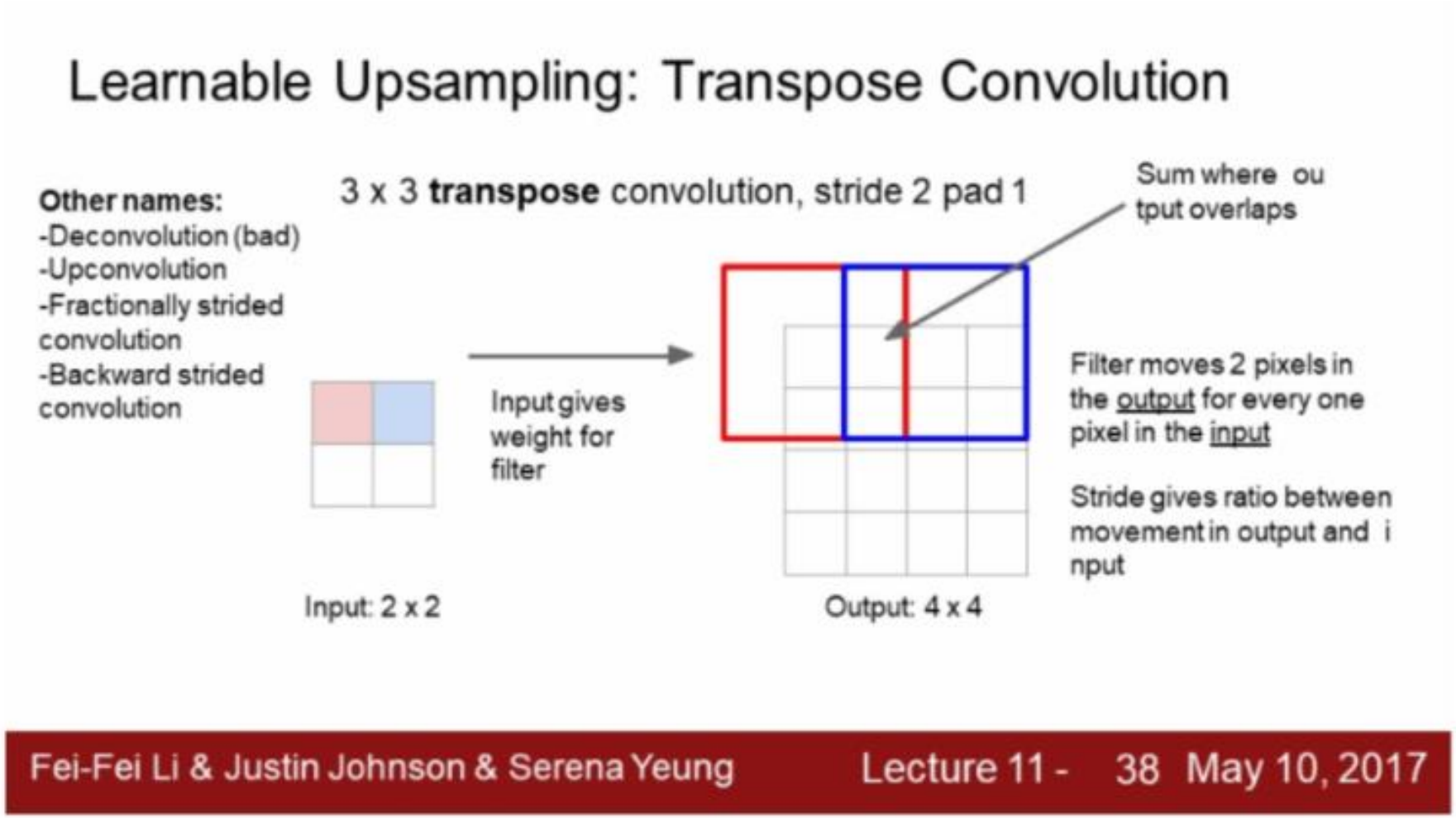
- To reduce no of channels while keeping the same width and height. In other words, you can resize the feature map. (e.g. (k,k,128) -> (k,k,32))
- Used in Inception module (GoogleNet, 2014)

- **1-D convolutions for Text Processing**

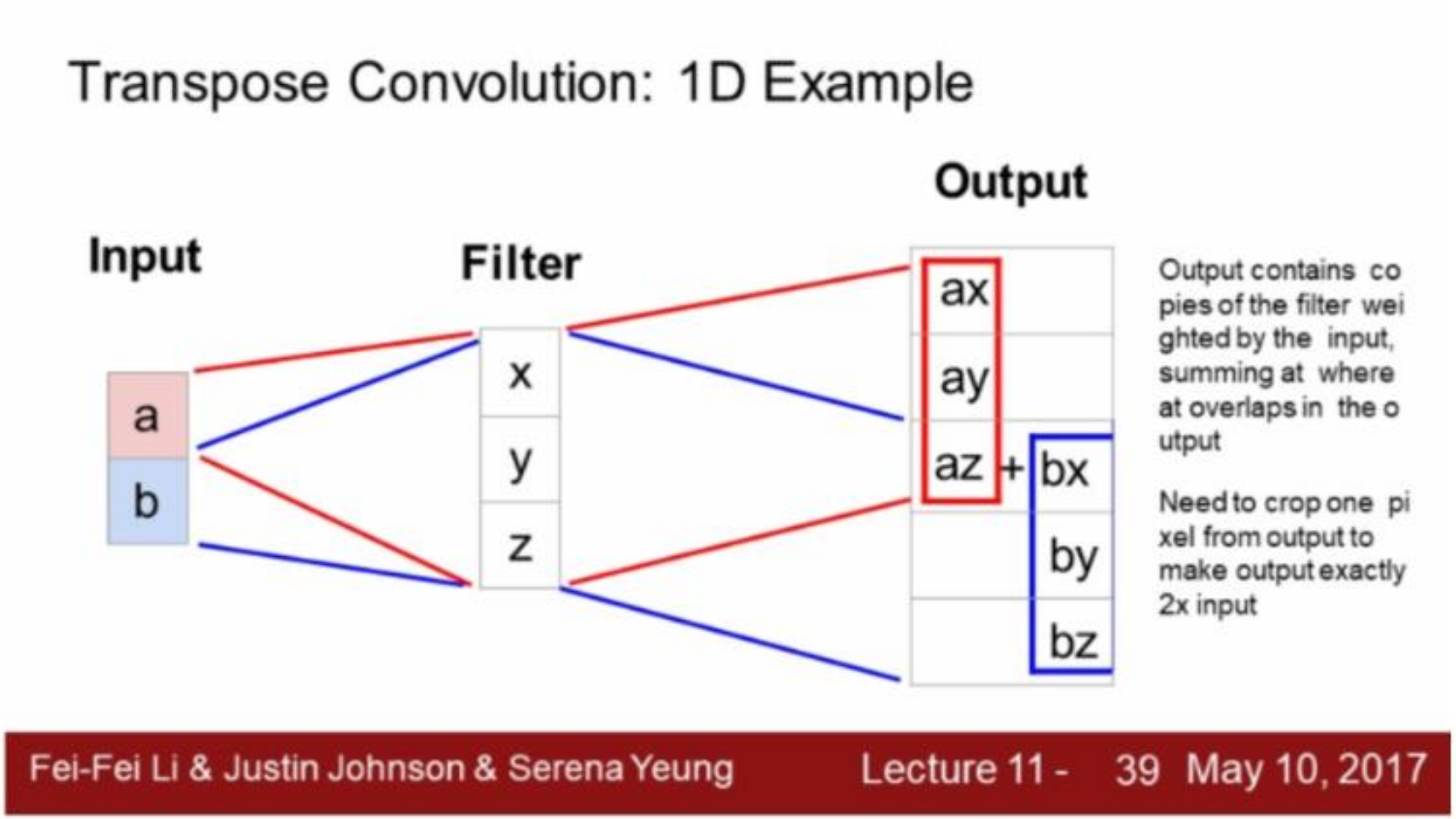


- kernel size 3 -> referencing 3-gram

Up-sampling - Transposed Convolution

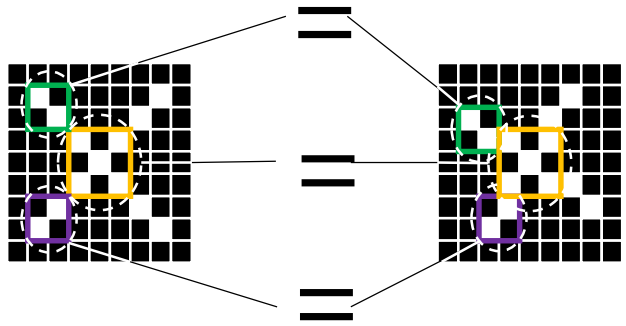


Transposed Convolution – 1D



Detecting patterns in CNN

- 공간상의 위치를 픽셀 단위로 일 대 일 비교하면
  - 이 두 그림에서 흰 점이 일치하는 부분은 거의 없다.
  - 따라서 이미지를 벡터로 환산(즉, 1차원 벡터로 변환)하여 비교하면 서로 다르다고 판단
- 3x3 픽셀 단위로 나누어 관찰하면
  - 절대적인 위치는 다르지만 일치하는 패턴이 여러 곳에 나타나므로
  - 다른 위치에 있는 같은 패턴을 찾아낼 수 있다
- CNN의 필터는 바로 이러한 패턴의 활성화 성분 크기 즉, 활성값을 찾아준다.
- 이러한 원리를 이용하여 CNN은 어떤 패턴의 크기, 절대적 위치, 두께, 회전 각도 등이 다르더라도 이를 찾아서 다음 계층으로 전달할 수 있다.

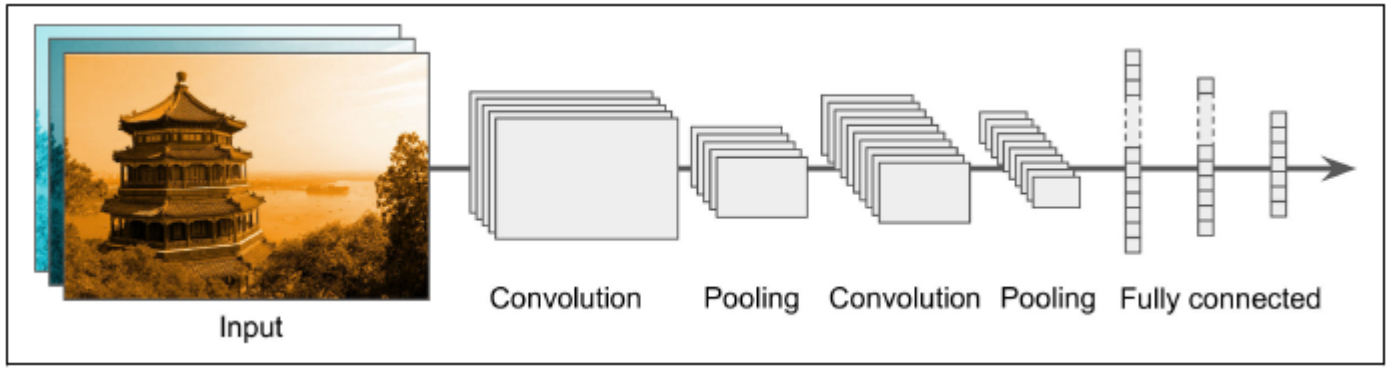


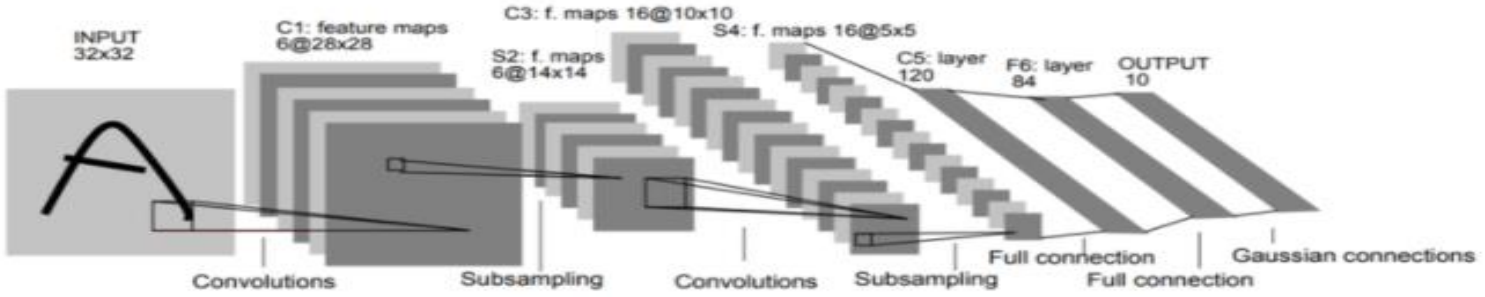
Detecting patterns in CNN

- CNN에서는 특성맵을 한가지만 만드는 것이 아니라 수십~수백 가지를 만든다. (처음에는 3원색인 경우 3에서 출발)
- 왜냐하면 찾아내야 할 패턴이 가로 성분, 세로 성분, 대각 성분, X-형 성분, 색상 정보, 엣지 등 여러 가지가 있기 때문
- 어떤 계층의 CNN 필터의 종류 수가 32라면 이 계층에서 생산되는 특성맵은 32개가 된다.
- CNN 필터를 커널(kernel)이라고 부르는데 커널 수가 많을수록 다양한 패턴을 찾아낼 수 있다
- 그러나 모델의 복잡도가 너무 커지면 과대적합의 원인
  - 내부 parameter 수가 많아 학습 데이터를 너무 정교하게 모델링하기 때문

CNN Architecture

- **Typical CNN architecture**





(ex) LeNet-5

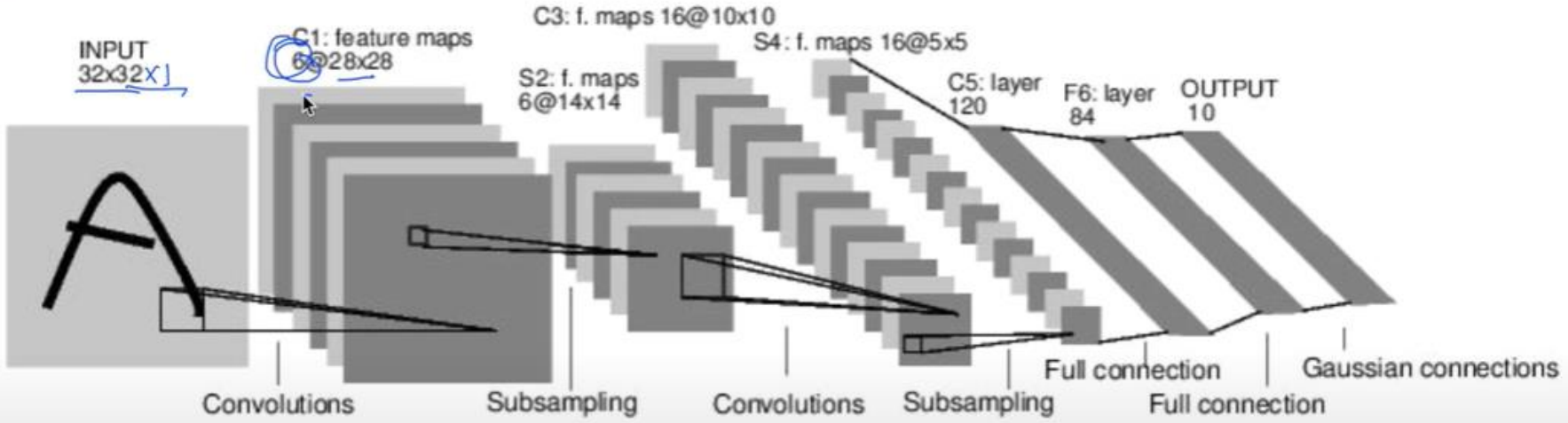
CNN Architecture

- CNN은 일반적으로 커널 필터링과 활성화 함수 그리고 최대 풀링을 묶어서 하나의 계층을 형성한다.
  - 참고) MLP에서는 전결합망과 활성화함수를 묶어서 한 계층을 형성
- CNN에서도 다양한 구조의 활성화 함수를 사용하고 필요하면 전결합망을 사용한다.
- 분류를 수행하는 경우 최종 계층에서는 전결합망을 만들고 그 결과에 소프트맥스 함수 (또는 시그모이드함수)를 적용

CNN example

Case Study: LeNet-5

[LeCun et al., 1998]

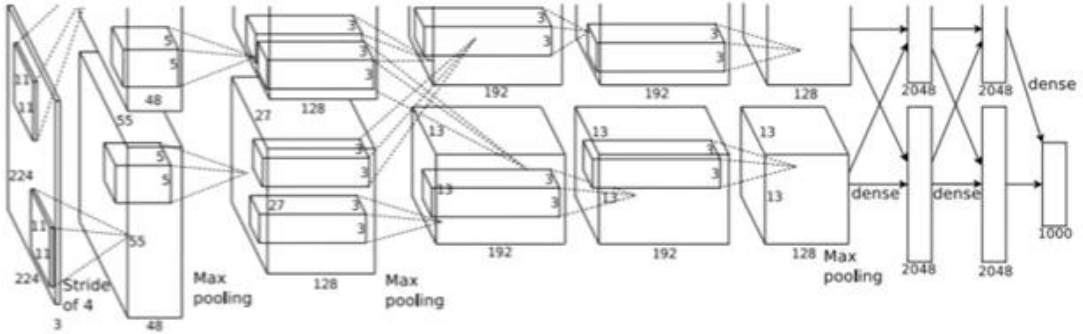


Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC]

CNN example

Case Study: AlexNet

[Krizhevsky et al. 2012]

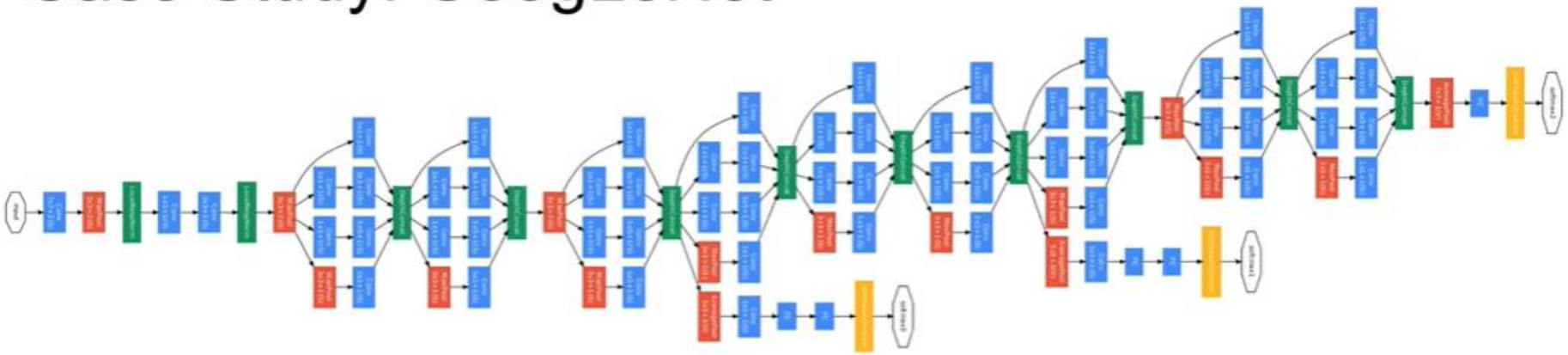


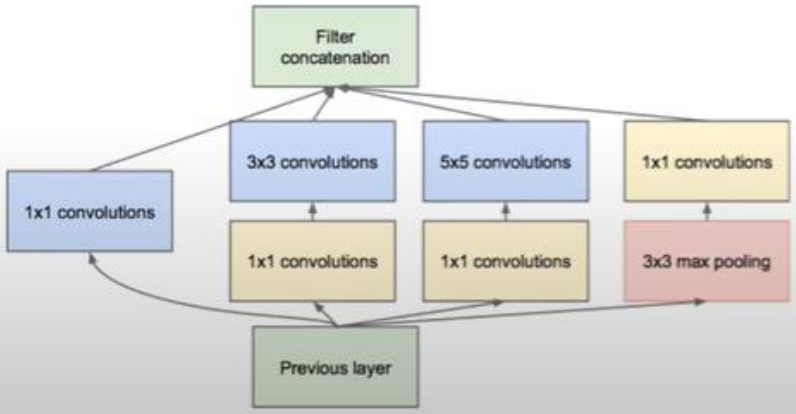
Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

CNN example

Case Study: GoogLeNet [Szegedy et al., 2014]





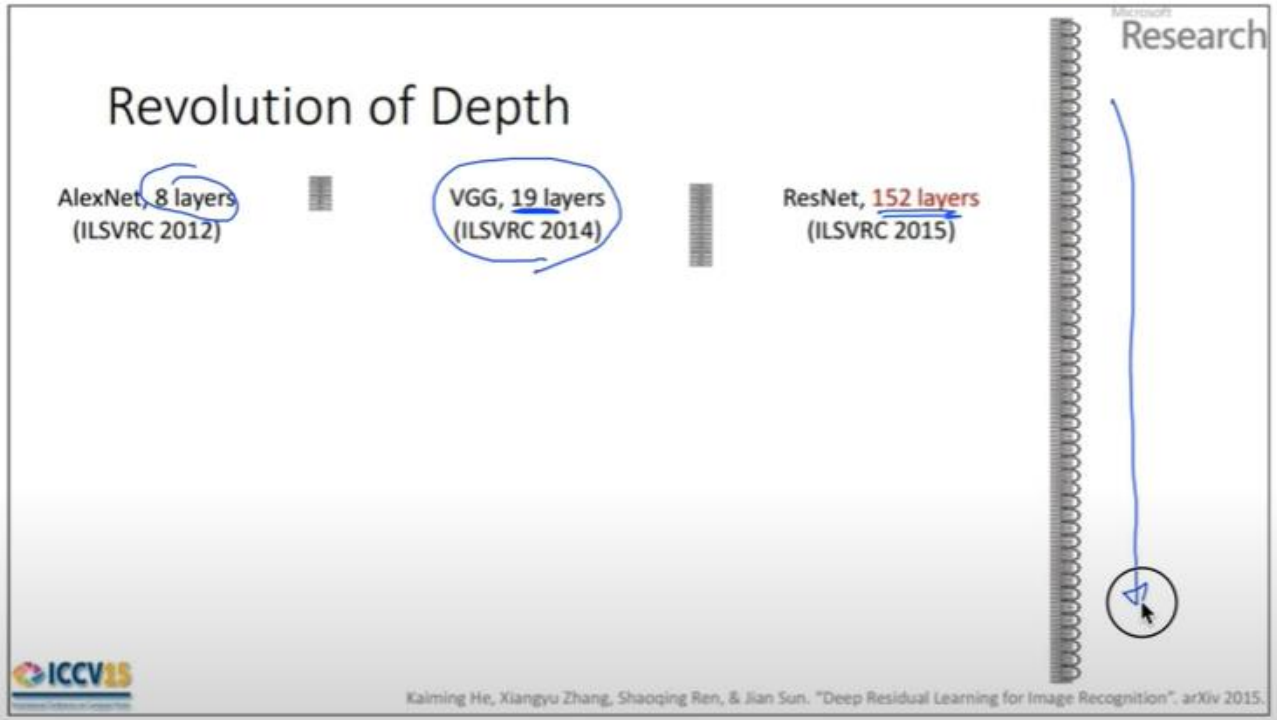
Inception module

ILSVRC 2014 winner (6.7% top 5 error)

CNN example

Case Study: ResNet *[He et al., 2015]*

ILSVRC 2015 winner (3.6% top 5 error)



2-3 weeks of training on 8 GPU machine

at runtime: faster than a VGGNet! (even though it has 8x more layers)

(slide from Kaiming He's recent presentation)

CNN 성능 개선

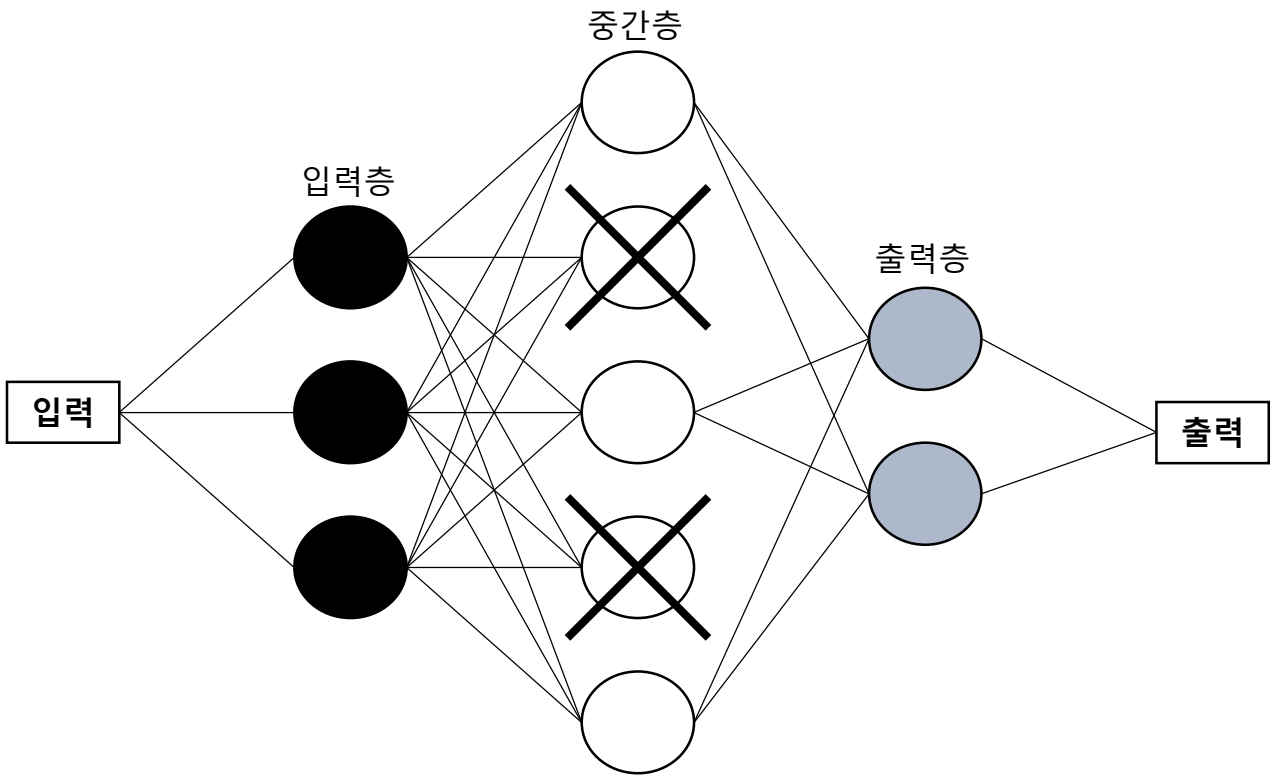
- 신경망은 파라미터 갯수가 많아서 과대적합 가능성이 항상 높다. 훈련 데이터 양이 많지 않을 때에는 특히 주의해야
  - 과대적합이 발생하면 신경망의 구조를 단순하게 만들어야 한다.
- 과대적합을 줄이는 작업을 일반화(generalize)를 위한 규제화(Regularize) 라고 부르는데,
  - 대표적인 규제화로 L2규제를 적용 - 네트워크를 구성하는 파라미터 값이 가능한 균등하게 분포하도록 제한
  - 케라스에서는 `regularizers.l2()` 함수를 사용

Drop Out

- 신호가 계층을 통과할 때 랜덤하게 유닛을 선택하여 신호를 전달하지 않도록 하는 방법
- 즉, 모든 신호를 다 사용하지 않고 고의적으로 신호의 일부를 누락시킨다
- 앙상블 효과를 얻어서 과대적합을 효과적으로 줄일 수 있다.
- 드롭아웃을 하면 입력과 출력간의 특정한 관계를 기억하지 못하게 하는 효과가 있어서 신경망이 보다 일반적인 학습을 할 수 있게 한다
- 네트워크의 기억을 랜덤하게 지우는 것이라고 볼 수 있다. 입출력의 특별한 관계를 평준화시키고 다양한 신경망 구조를 이용한 효과(즉, 앙상블 효과를 얻어 성능을 개선하는 방법)

Drop Out

- 드롭아웃은 학습을 하는 동안에만 적용
- 학습이 종료된 후 예측을 하는 단계에서는 모든 유닛을 사용하여 예측



How to find Good CNN architecture

- 최적의 신경망을 구성하는 계층의 수, 유닛의 수, 배치 크기, 학습률의 설정 등이 어려운 과제
- 기본 전략 : 처음에는 구조를 간단히 출발
- 일단 동작을 확인하고 성능을 개선한다.
  - 계층이 2~3만으로도 동작하는지를 확인
  - 만일 2~3개의 계층으로 모델이 동작하지 않으면 계층 수를 늘려도 동작하지 않는다고 알려져 있다.
- 입력 데이터를 간단히 만들어 보는 것도 필요
  - 예를 들어 10가지 동물 이미지를 구분하는 모델이 필요하다고 하여도 우선 몇 가지 대표적인 동물들을 구분하는 모델을 먼저 만들어보는 것

Batch Size

- 배치 크기 : 신경망이 한번에 학습하는 입력 데이터 수
- 배치 크기가 클수록 학습이 정교하고, 기울기를 정확히 구할 수 있으나 계산량이 많아진다.
  - 필요한 메모리 사용량이 많아 메모리 오류가 날 가능성이 높다(특히 GPU를 사용할 때). 처음에는 배치 크기를 작게 16~32정도로 적게 잡고 시작
- 배치 크기가 작을 때에는 기울기가 상대적으로 정확하게 계산되지 못하므로 학습률도 작게 잡아야 한다.
  - 일단 작은 값의 학습률로 동작하는 것을 확인하고 학습률을 조금씩 크게 한다

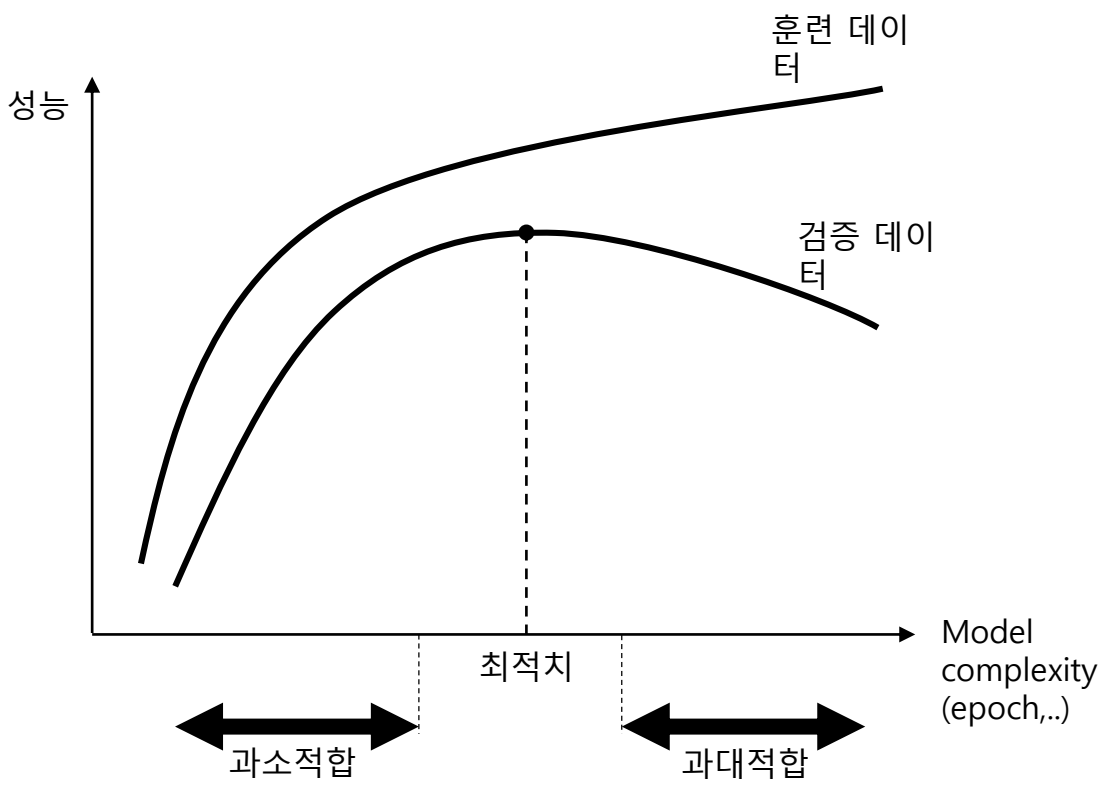
Batch Normalization (배치 정규화)

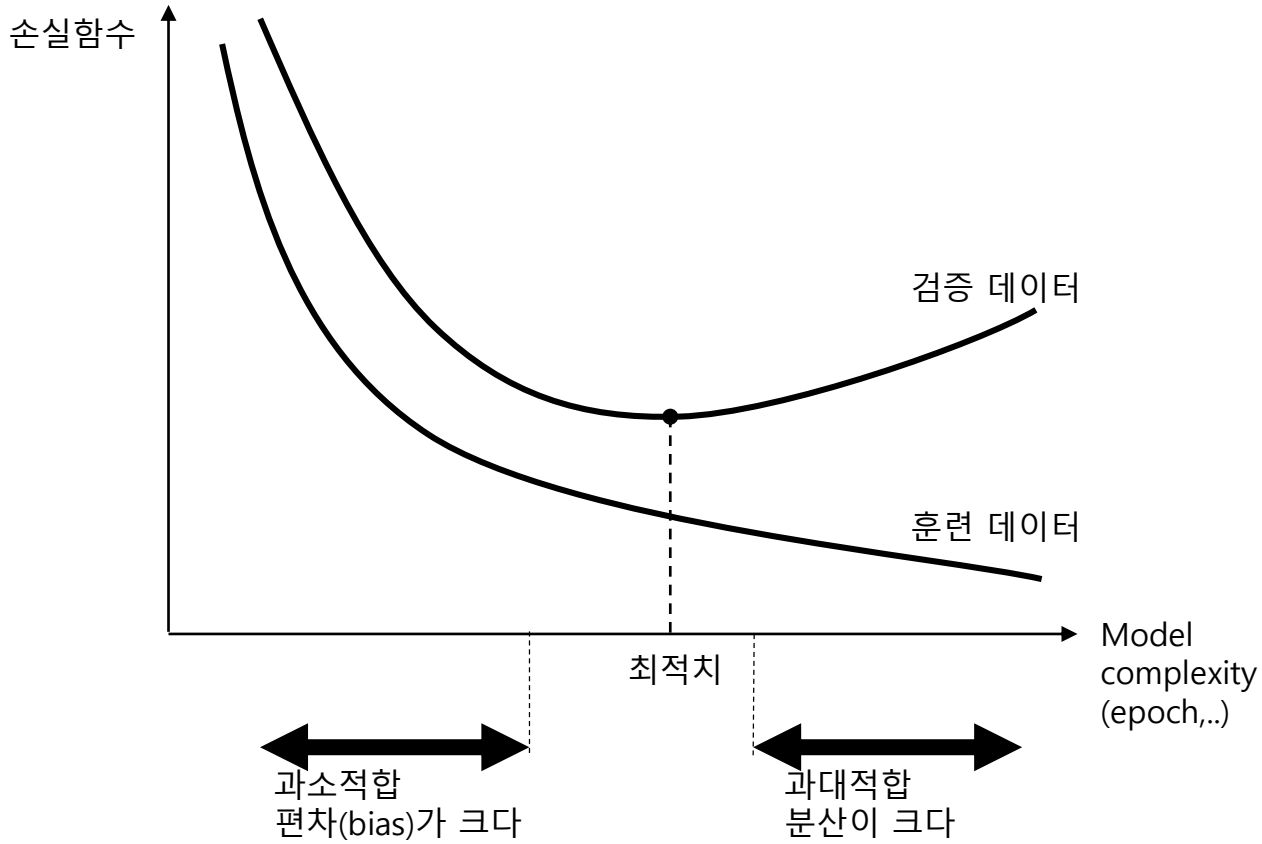
- 딥러닝 학습에서 해결해야할 어려운 문제 중 하나
  - Vanishing Gradient Problem
- 이는 기울기 값에 비례하여 학습시킬 때 기울기 즉, 미분값이 0에 가까워지면 변화량이 매우 적어지고 이것이 앞단의 계층으로 전파되는 양이 급속히 줄어들어 학습이 잘 되지 않는 현상
- 이러한 문제를 해결하기 위해 배치 정규화가 제시됨
  - 이는 "계층별로", 주어진 배치 데이터를 대상으로 정규화를 다시 수행하여 **데이터의 분포**가 너무 작거나 너무 커지지 않게 하는 방식
  - **학습 시 미니배치 단위로 정규화 : (평균 0, 분산 1)**

Overfitting (과대적합)

- 신경망은 파라미터가 많아서 과대적합되기 쉽다. 즉, 상세한 모델링이 가능하여, 훈련 데이터 수가 적으면 이 훈련 데이터의 속성을 모두 기억할 수 있어 과대적합되기 쉬운 것이다.
- 훈련데이터에 대해서는 계속 성능이 좋아지지만 검증 데이터에 대해서는 오히려 성능이 나빠진다면 과대적합한 것이다.
- 성능의 개선되는지를 측정하는 방법
  - 정확도 등 최종 성능 지표를 관찰
  - 손실 함수가 줄어드는지를 관찰

Overfitting





How to avoid or reduce Overfitting

- 학습조기중단 (Early Stopping)
- L1 이나 L2 규제 (Regularization)
- 드롭아웃(Dropout)
- 데이터 확장(Data Augmentation)
  - 훈련 데이터가 많은 것처럼 보이는 효과

Data Augmentation

- 과대적합이 일어나는 이유 중 하나
  - 훈련데이터가 부족하기 때문
- 훈련 데이터가 충분히 많다면 과대적합을 줄일 수 있다.
- 데이터 확장이라 훈련 데이터를 다양하게 변형하여 변형된 새로운 훈련 데이터처럼 사용함으로써 마치 훈련 데이터 수가 늘어난 효과를 얻는 것이다.
- 데이터 확장을 사용하면 여러 이포크를 수행해도 똑같은 데이터를 가지고 학습하지 않게 된다.
- Affine Transform
  - Rotation, shifting, rescaling, flipping, shearing, stretching

Data Augmentation (데이터 확장)

- 과대적합이 일어나는 이유 중 하나
  - 훈련데이터가 부족하기 때문
- 훈련 데이터가 충분히 많다면 과대적합을 줄일 수 있다.
- 데이터 확장이라 훈련 데이터를 다양하게 변형하여 변형된 새로운 훈련 데이터처럼 사용함으로써 마치 훈련 데이터 수가 늘어난 효과를 얻는 것이다.
- 데이터 확장을 사용하면 여러 이포크를 수행해도 똑같은 데이터를 가지고 학습하지 않게 된다.
- Affine Transform
  - Rotation, shifting, rescaling, flipping, shearing, stretching
- Synthetic Data Generation using GAN

Data Augmentation (example)

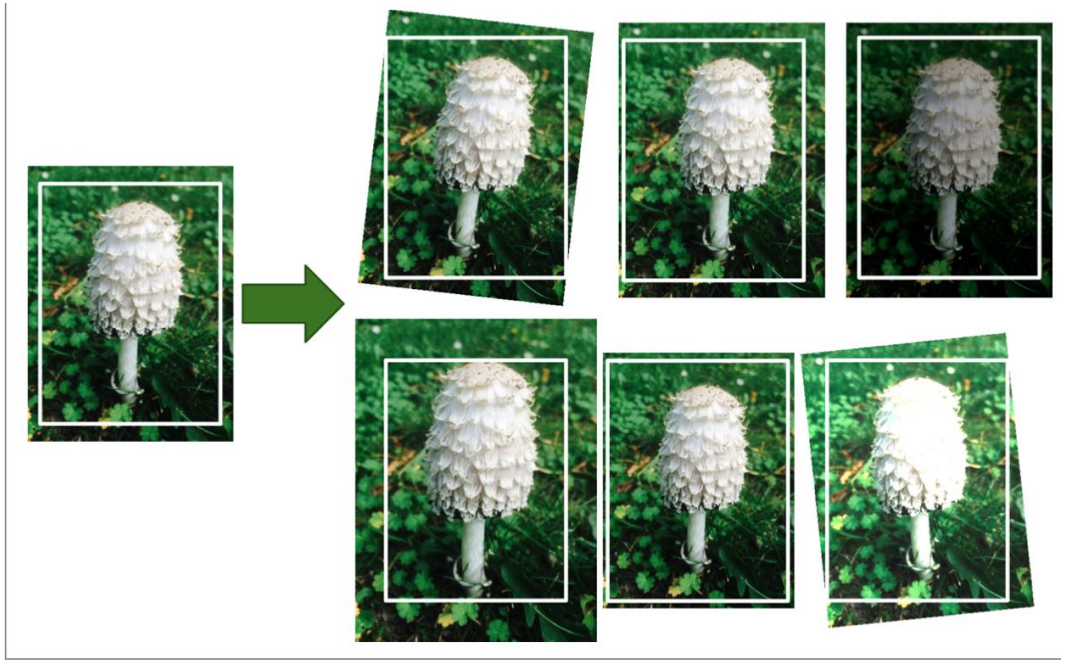
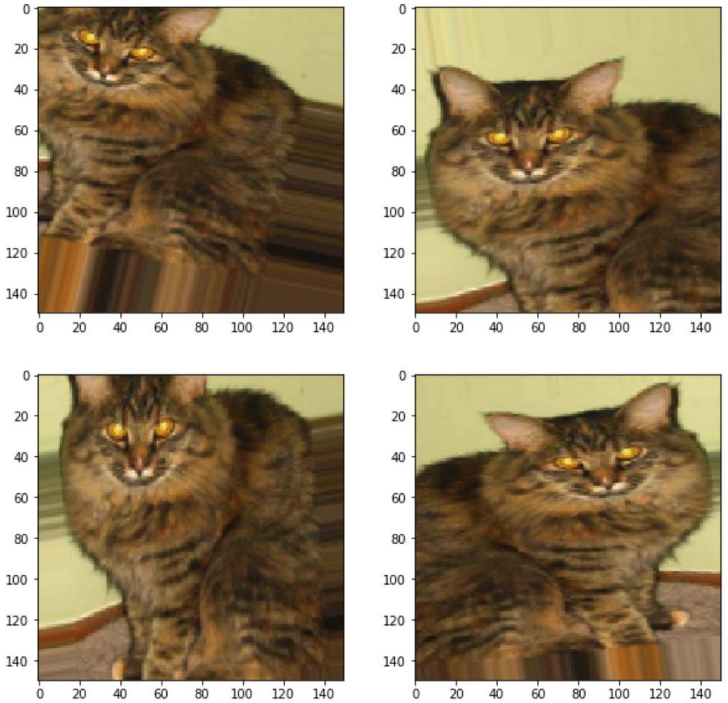
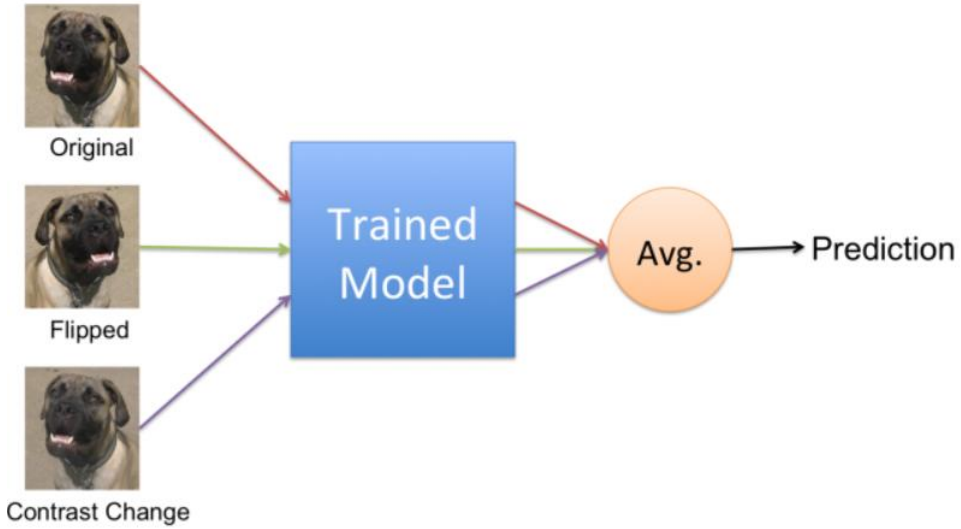


Figure 11-10. Generating new training instances from existing ones



Test Time Augmentation (TTA)

- Augmentation in Training:
  - 훈련 데이터가 부족할 때 데이터 셋을 회전/반전/뒤집기/늘이기/줄이기/노이즈 등 다양한 방법을 사용
- TTA (Test Time Augmentation):
  - 학습할 때 augmentation하는게 아닌, 테스트 셋으로 모델을 테스트하거나, 실제 운영(deploy)할 때 augmentation을 수행
  - 각 확장된 데이터에 대해 Model 을 적용(Test) 한 후 평균하여 최종 결과 도출 (다양한 관점의 테스트 이미지에 대해 테스트하므로 성능이 향상, but not always !)



Transfer Learning (전이학습)

- 전이학습 이란 다른 데이터 셋을 사용하여 이미 학습한 모델을 유사한 다른 데이터를 인식하는데 사용하는 기법이다.

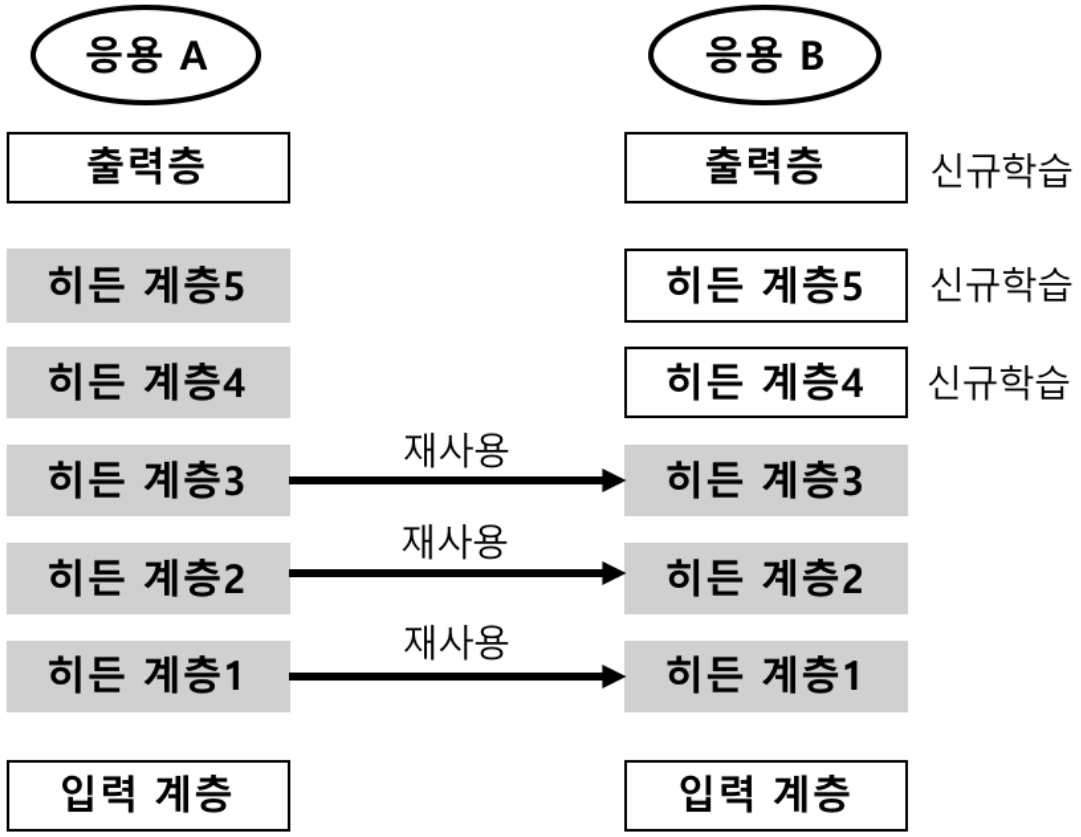
- 이 방법은 특히 새로 훈련시킬 데이터가 충분히 확보되지 못한 경우에 학습 효율을 높여준다.

- 이미지넷(imagenet)에는 동물이나 일상생활의 물건들을 주로 포함하여 1000종의 이미지를 갖고 있으며 140만장의 사진이 있다.

- 이미지넷에는 고양이 강아지를 포함한 많은 동물 이미지도 들어 있으며, 이를 강아지 고양이 분류 문제에 사용할 수 있다.
  - 예제코드에서는 2014년에 소개된 VGG16 모델을 사용하겠다.

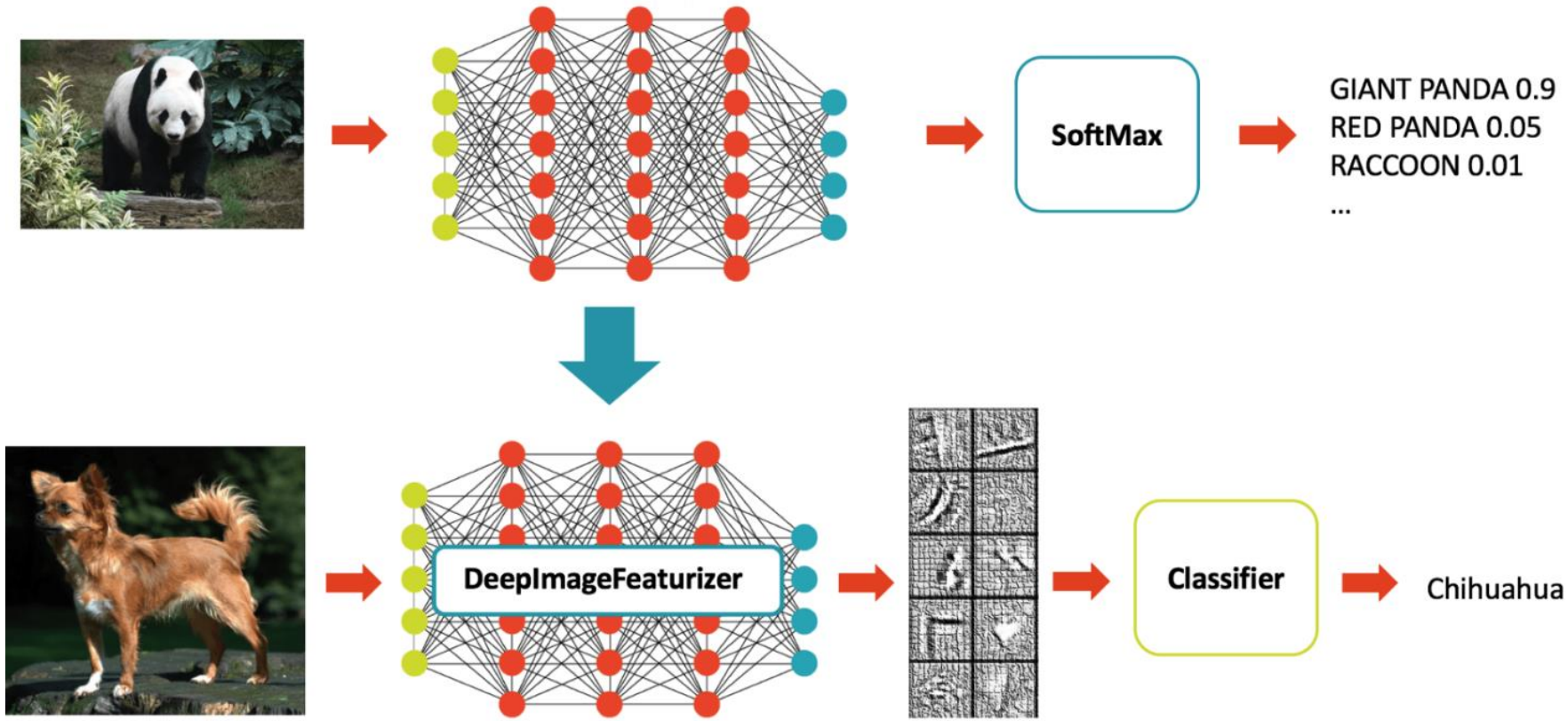
- 사전학습모델을 이용하는 방법은 특성 추출(feature extraction) 방식과 미세조정(fine-tuning) 방식이 있다.

Transfer Learning (전이학습)



전이 학습

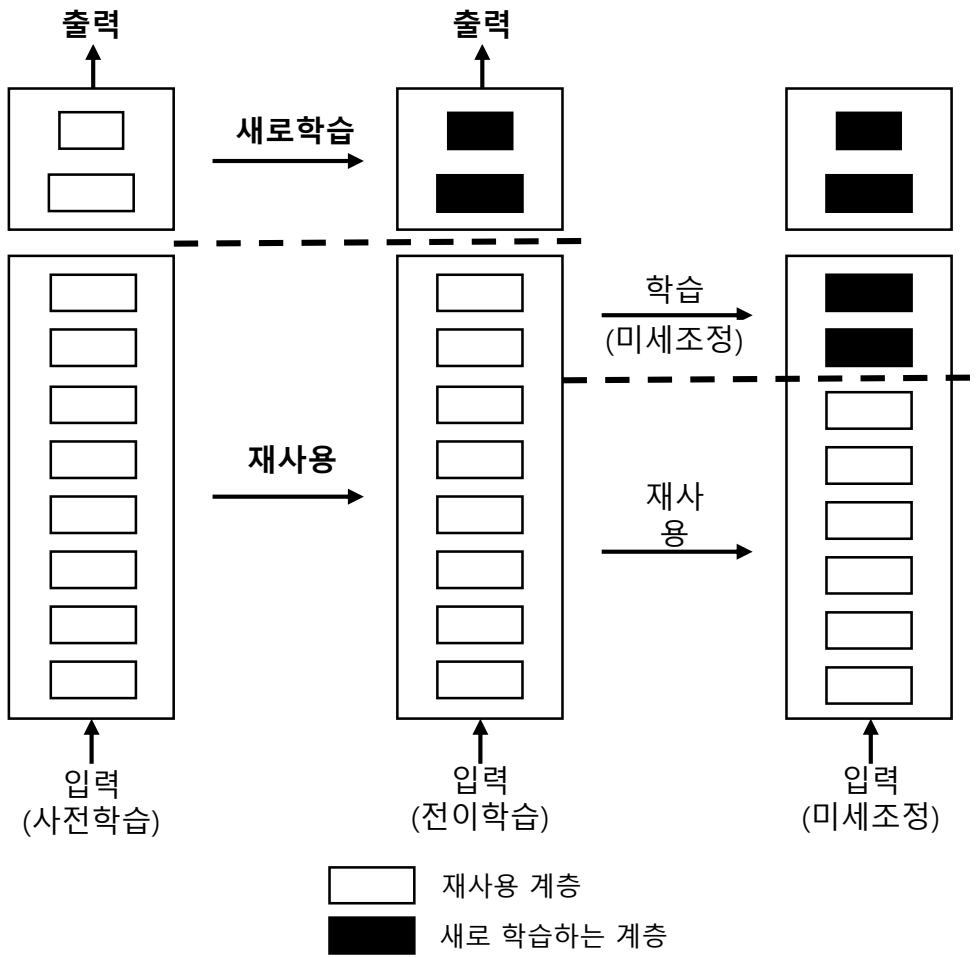
Transfer Learning (전이학습)



Transfer Learning – 특성 추출 방식

- 특성 추출이란 이전의 네트워크로부터 배운 표현 방식을 사용하여 새로운 데이터 샘플에서 추가로 흥미로운 특성들을 추출하는 것
- 이렇게 얻은 특성들을 사용하여 새로운 분류기를 작동시키고 학습을 수행한다.
- 이미지 분류기는 특성추출 부분(컨볼류션 네트워트와 풀링 계층의 조합)과 분류기 부분(전결합망) 으로 크게 두 부분으로 구성된다.
- 컨볼류션과 풀링으로 구성된 부분을 컨볼류션 베이스라고한다
- 특성 추출 방식은 컨볼류션 베이스를 그대로 사용하고, 새로운 데이터를 여기에 적용하여 훈련시키는 방식이다.

Transfer Learning – 특성 추출 방식



Transfer Learning – 특성 추출 방식

- 컨볼류션 베이스 부분만 재사용하는 이유는 이 부분은 상당히 일반적인 학습정보를 포함하고 있기 때문이다.
- 컨볼류션 계층에서도 재사용할 수 있는 정보의 수준은 몇 번째 계층인지에 따라 다르다. 모델의 앞단의 계층일수록 에지, 색상, 텍스처 등 일반적인 정보를 담는다.
- 반면에 뒷 부분의 깊은 계층일수록 추상적인 정보를 담는다 (예를 들어 고양이 귀, 강아지 귀 등)
- 새롭게 분류할 클래스의 종류가 사전 학습에 사용된 데이터와 특성이 매우 다르면, 컨볼류션 베이스 전체를 재사용해서는 안되고 앞단의 일부 계층만을 재사용해야 한다.

Transfer Learning – 미세 조정 방식

- 모델 베이스 중 상위 몇개의 계층은 전결합층 분류기와 함께 새로 학습시키는 방식이다.
- 최종 분류기의 계수가 랜덤하게 초기화 되어 있으므로 이를 먼저 학습시키며 이때 VGG16 모델의 컨볼류션 베이스를 초기에는 고정해야 한다.
- 먼저 분류기를 계수를 학습시킨 다음에 (즉, 이 동안은 미세조정을 하지 않도록 상위계층의 계수를 고정시켜 두고), 그 이후에 미세조정을 해야 한다.
- 처음부터 베이스 상위계층의 계수를 같이 훈련시키면 분류기에서 발생하는 큰 에러 값으로 인해, 사전 학습된 정보가 많이 손실된다.

Transfer Learning – 미세 조정 절차

1. 사전 학습된 기본 네트워크 상단에 새로운 네트워크를 추가한다.
2. 기본 네트워크를 고정시킨다.
3. 새로 추가한 부분을 학습시킨다.
4. 기본 계층 중에 학습시킬 상위 부분의 고정을 푼다
5. 고정을 푼 계층과 새로 추가한 계층을 함께 훈련시킨다.
6. 미세 조정을 천천히 수행하기 위해서 느린 학습 속도를 선택한다. 갑자기 큰 변화를 주면 사전 학습된 내용이 훼손되기 때문이다.

When does Transfer Learning make sense?

- Task A and B have the same input x.
- You have a lot more data for Task A than Task B.
- Low level features from A could be helpful for learning B.

Callback (콜백)

- 케라스, 콜백 함수
  - 모델 학습을 하는 동안 중간 결과를 저장하거나
  - 조기 종료를 시키거나 할 수 있는 기능
- 콜백은 여러 가지를 동시에 지정할 수 있으며 이를 fit 함수의 callbacks 인자로 넘겨줄 수 있다.
- History()
  - 훈련중에 발생하는 여러 이벤트를 History 객체에 저장
  - keras.callbacks.History()

Callback (콜백)

- **모델 저장**: 매 이포크마다 모델을 저장
  - 저장하는 주기(period)와 베스트 모델만 저장 등을 지정
  - keras.callbacks.ModelCheckpoint(filepath, monitor='val_loss', verbose=0, save_best_only=False, save_weights_only=False, mode='auto', period=1)

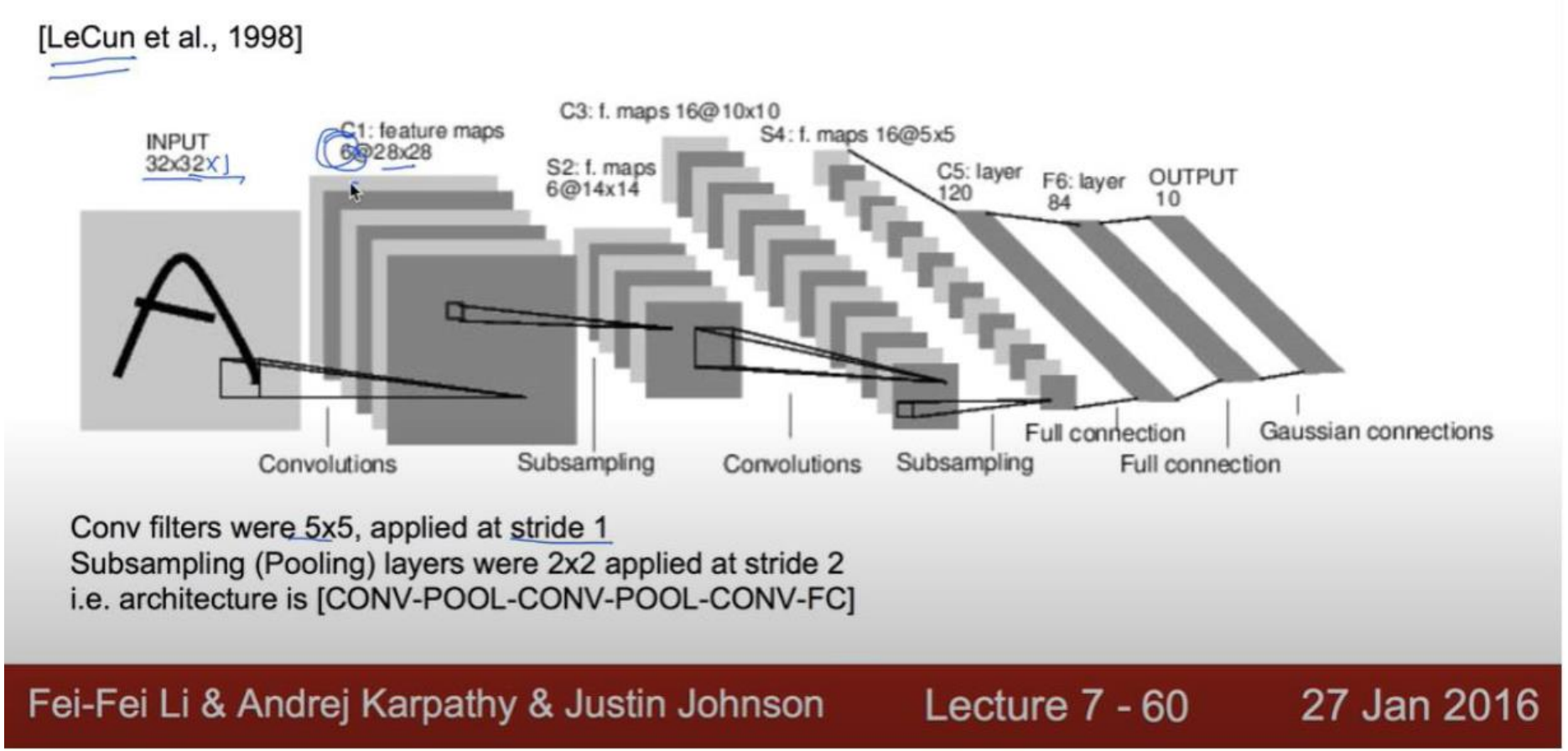
- **조기 종료 (Early Stopping)**
  - 손실값, 성능 등 관찰 중인 지표가 어떤 조건을 만족하면 이포크 실행을 종료
  - 조기 종료 조건들을 인자로 지정
  - keras.callbacks.EarlyStopping(monitor='val_loss', min_delta=0, patience=0, verbose=0, mode='auto', baseline=None, restore_best_weights=False)

- 참고) https://keras.io/callbacks/#modelcheckpoint

Datasets for Image Processing (CNN)

- **ImageNet**
  - largest image dataset for computer vision, used in ILSVRC(ImageNet Large Scale Visual Recognition Challenge) for image classification and object detection (150 GB)
  - 469*387 resolution in average (usually cropped to 256*256 or 224*224 before usage)
  - More than 14 million images with more than 21,000 groups(classes)
  - More than 1 million images have bounding box annotations
- **ILSVRC**
  - To evaluate algorithms for object detection and image classification (and localization) at large scale
  - To measure the progress of computer vision for large scale image indexing for retrieval and annotation
  - Uses smaller portion of the ImageNet (1,000 categories with 1.3 million train images, 50,000 validation images, and 1,00,000 test images
  - Available in Kaggle
  - 2010 ~ 2017

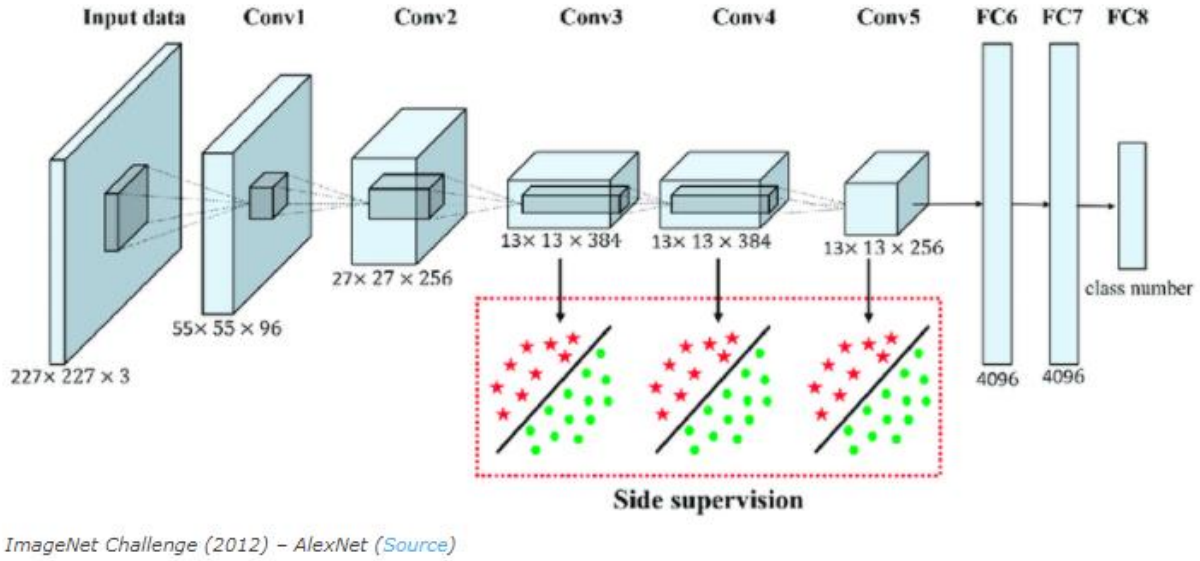
Lenet-5



- First architecture for CNN (excellent on MNIST dataset)
- Small and easy to understand
- Uses **tanh** activation function

ILSVRC Winners

- **ALexNet – 2012 Winner (top-5 error rate 15.3%)**
  - Use CNN (prior to 2012, the classification model error was 25%)
  - Regarded as a Pioneer of CNN and starting point of the Deep learning
  - 60 million parameters
  - Used **ReLU** activation function, heavy data augmentation, dropout, and overlapped pooling layers

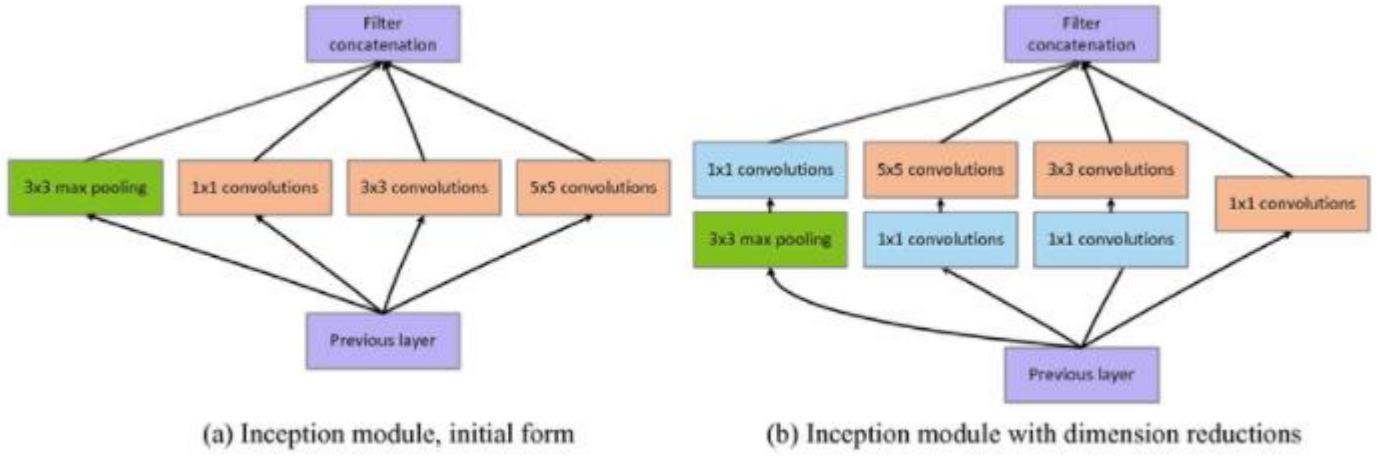


ImageNet Challenge (2012) – AlexNet (Source)

(*) top-5 error: The model is considered to have classified a given image correctly if the target label is one of the model's top 5 predictions. (image classification)

ILSVRC Winners

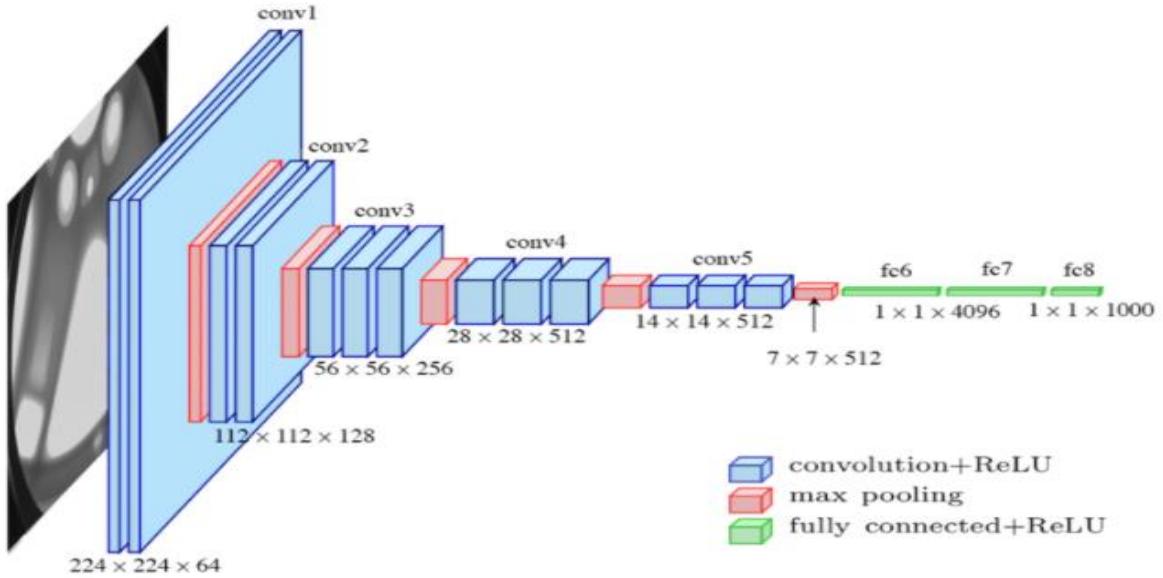
- **Inception V1 (GoogleNet) – 2014 Winner (top-5 error rate 6.67%)**
  - The v1 stands for 1st version and later there were further versions v2, v3, etc. It is also popularly known as GoogLeNet.
  - Deep with 22 layers.
  - Used multiple types of filter size, instead of being restricted to a single filter size, in a single image block, which we then concatenate and pass onto the next layer.
  - Used 1x1 convolutional with ReLU to reduce dimensions and number of operations.



ImageNet Challenge (2014)- Inception-V1 (GoogLeNet) Source

ILSVRC Winners

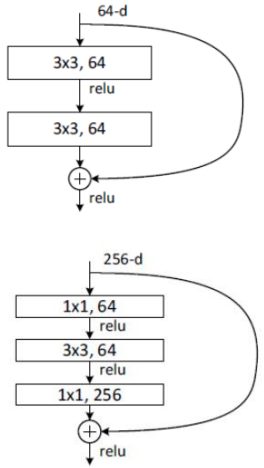
- **VGG-16 (University of Oxford) – 2014 Runners-Up (top-5 error rate 7.3%)**
  - Despite not winning the competition, VGG-16 architecture was appreciated and went on to become one of the most popular image classification models.
  - instead of using large-sized filters like AlexNet, it uses several 3×3 kernel-sized filters consecutively. The hidden layers of the network leverage ReLU activation functions.
  - VGG-16 is however very slow to train and the network weights, when saved on disk, occupy a large space.



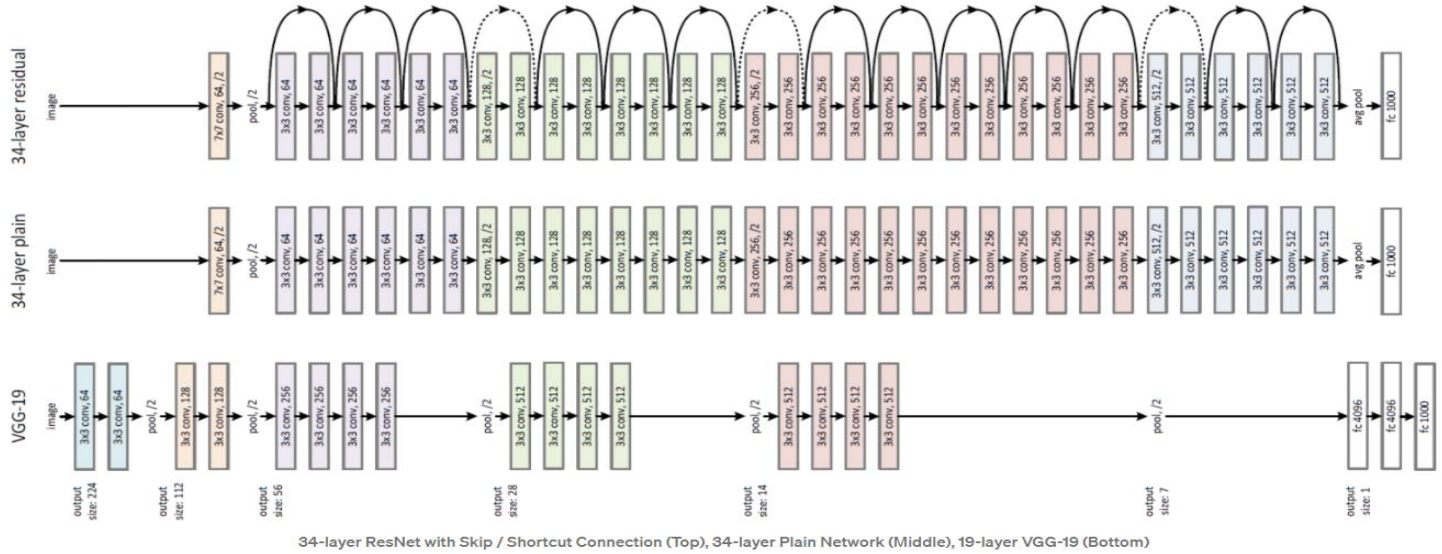
ImageNet Challenge (2014) – VGG-16 (Source)

ILSVRC Winners

- **ResNet – 2015 Winner (top-5 error rate 3.57%)**
  - ResNet (Residual Network) was created by the Microsoft Research team.
  - To solve the problem of vanishing/exploding gradients, a skip/shortcut connection is added to add the input x to the output after few weight layers as below:
  - 1×1 Conv can reduce the number of connections (parameters) while not degrading the performance of the network so much. (as in Inception-V1)
  - ResNet-18/34/50/101/152 has 1.8/3.6/3.8/7.6/11.3 GFLOPs (lower than VGG-16/19 with 15.3/19.6 GFLOPS)



The Basic block (top) and the Proposed Bottleneck design (bottom)

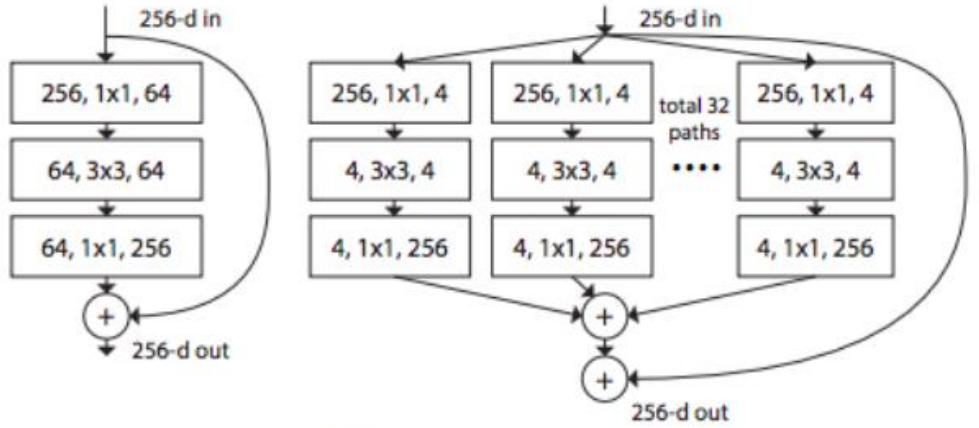


34-layer ResNet with Skip / Shortcut Connection (Top), 34-layer Plain Network (Middle), 19-layer VGG-19 (Bottom)

(*) VGG-19 (bottom): state-of-the-art approach in 2014
middle: deeper network of VGG-19 (i.e. more Conv layers)

ILSVRC Winners

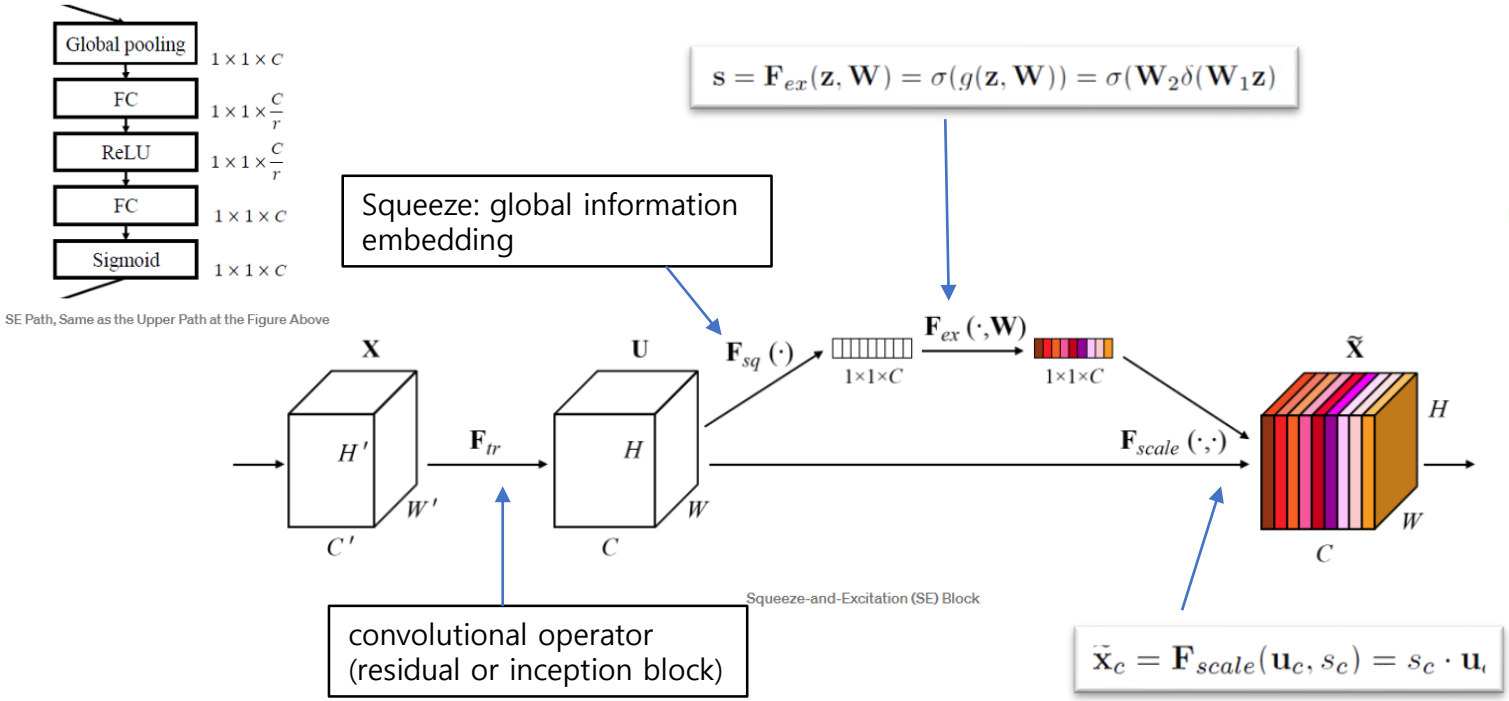
- **ResNeXt – 2016 Runners-Up (top-5 error rate 4.1%)**
  - Developed in the collaboration of the Researchers from UC San Diego and Facebook AI Research.
  - Inspired by (ResNet + VGG + Inception).
  - Still it became a popular model.
  - stacks the blocks and then use the ResNet approach of residual blocks. Here the hyper-parameters such as width and filters were also shared.

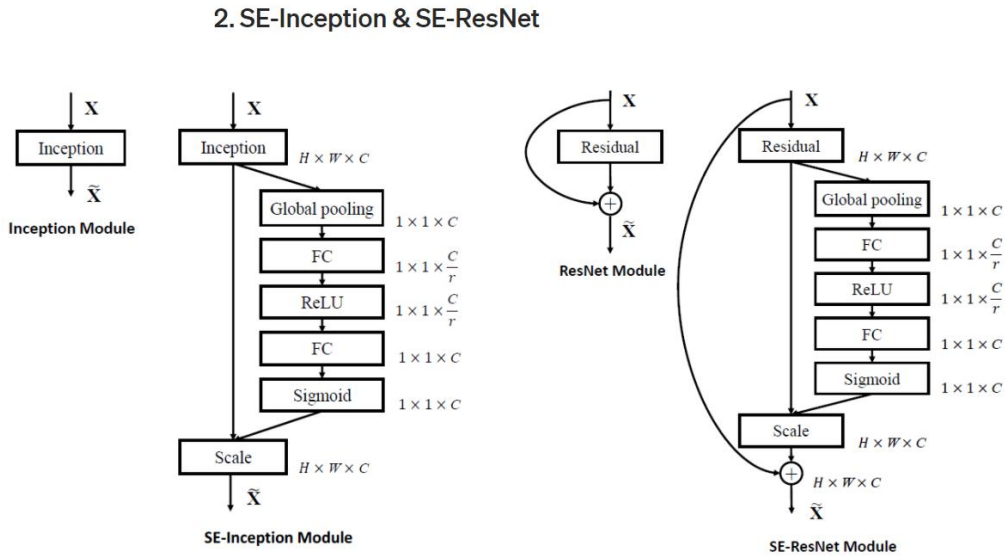


ImageNet Challenge (2016)- ResNeXt (Source)

ILSVRC Winners

- **SENet(Squeeze-and-Excitation Network) – 2017 Winner (top-5 error rate 2.251%)**
  - Developed in University of Oxford.
  - With "Squeeze-and-Excitation" (SE) block that **adaptively recalibrates channel-wise feature responses by explicitly modelling interdependencies between channels**, SENet is constructed.
  - SE block can be added to both Inception and ResNet block easily as **SE-Inception** and **SE-ResNet**. (achieved the best 2.25%).





(*) Human Beings – Top-5 Error Rate – 5.1%