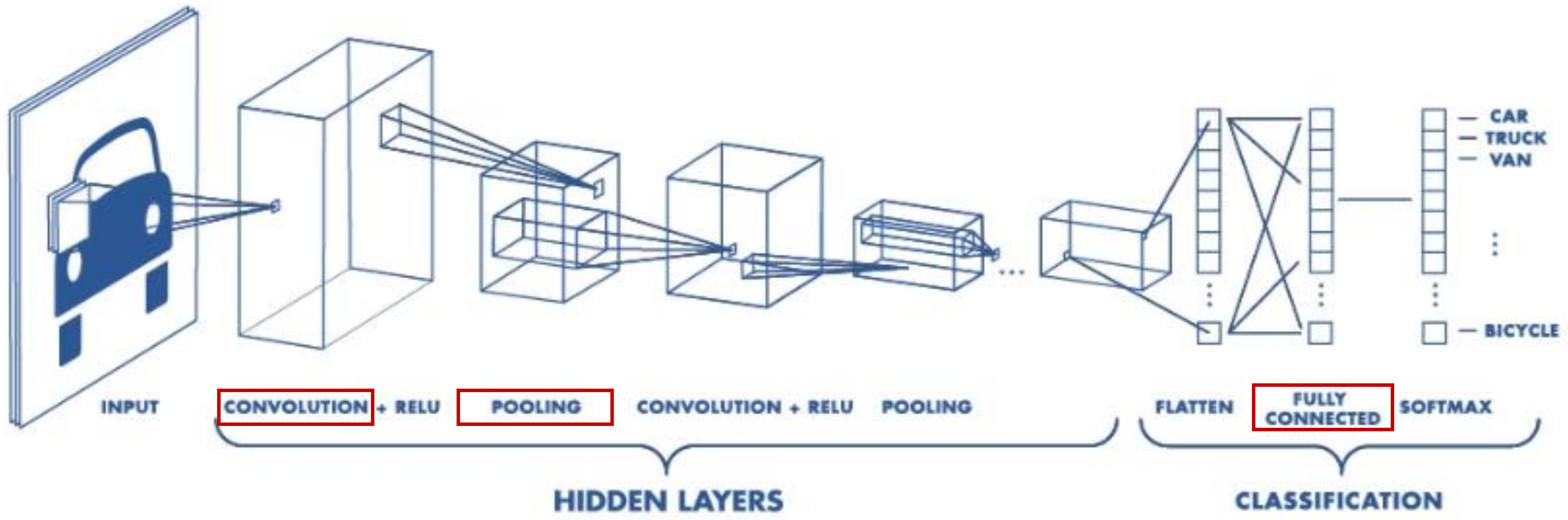
Convolutional Neural Network

TA. Dongjae Kim

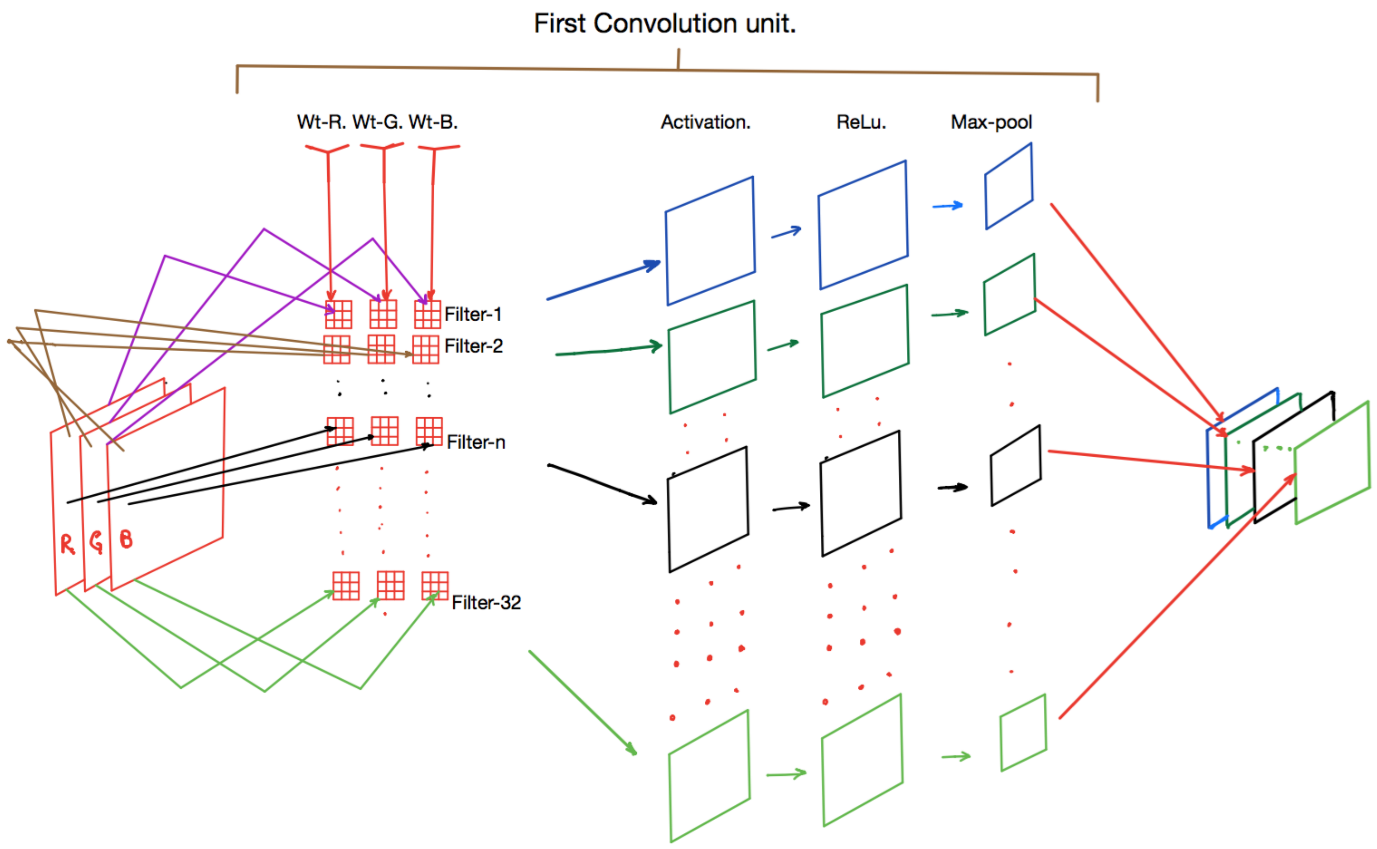
Contents

1. Convolutional Neural Network



- Main Components
  - Convolution (Shared weights, dimension reduction)
  - Nonlinearity (ReLU)
  - Max pooling (translation invariance, dimension reduction for FC layer)
  - Fully connected Layer (classification)

1-1. Convolution Layer



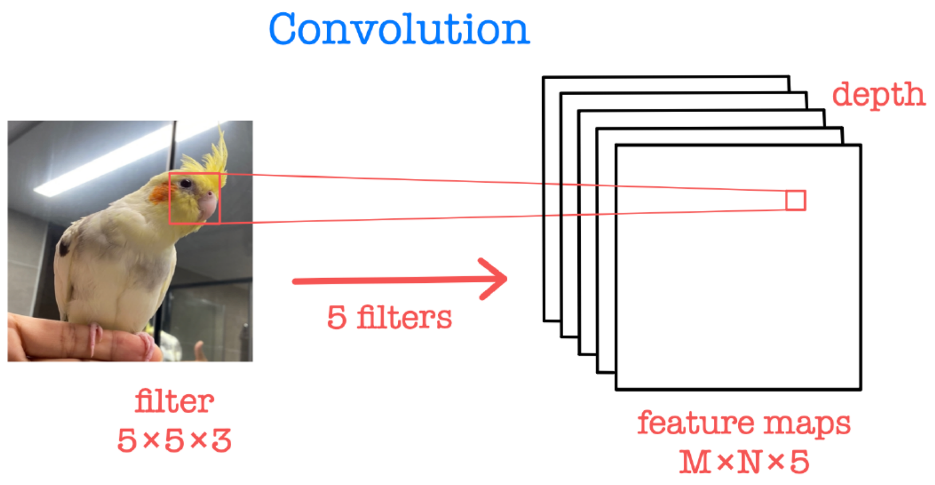
- The weights and the biases which are the components of the filters change when training the network.

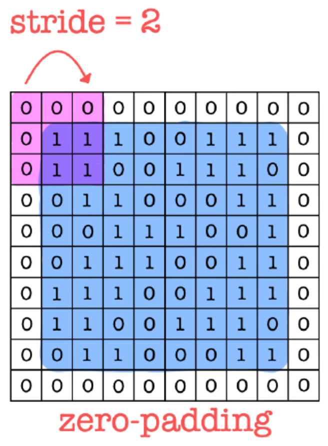
1-1. Convolution Layer

Hyperparameters on Convolution layer

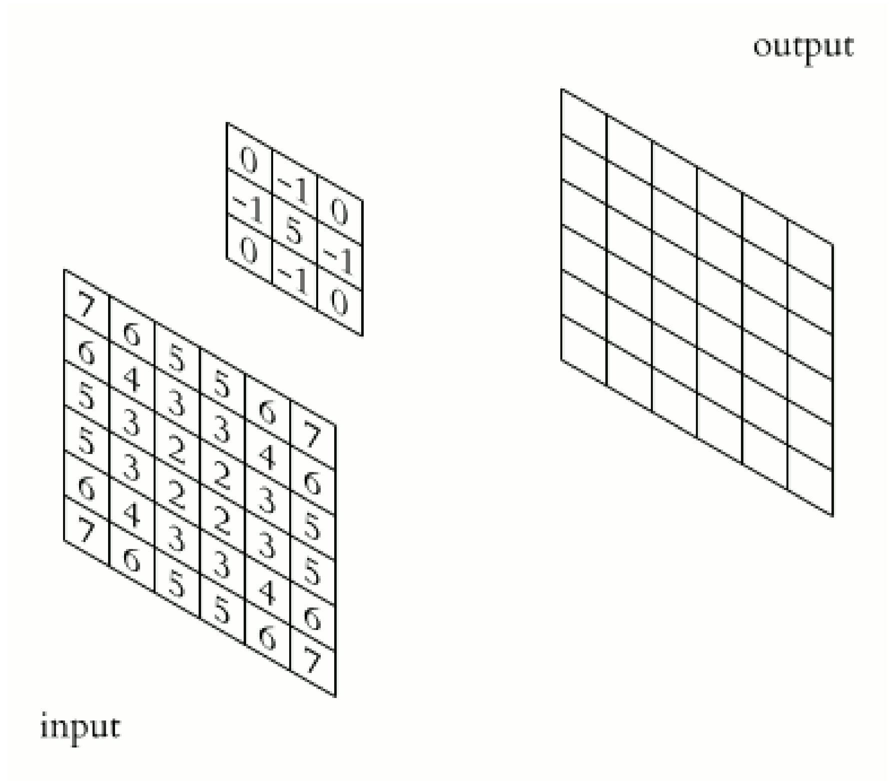
- filters: number of filters (output feature maps)
- kernel size: size of kernel (convolution window: height x width)
- strides: distance between two successive kernel positions
- padding: "valid" = no padding

  "same" = padding with zeros to make the output with the same size as the input



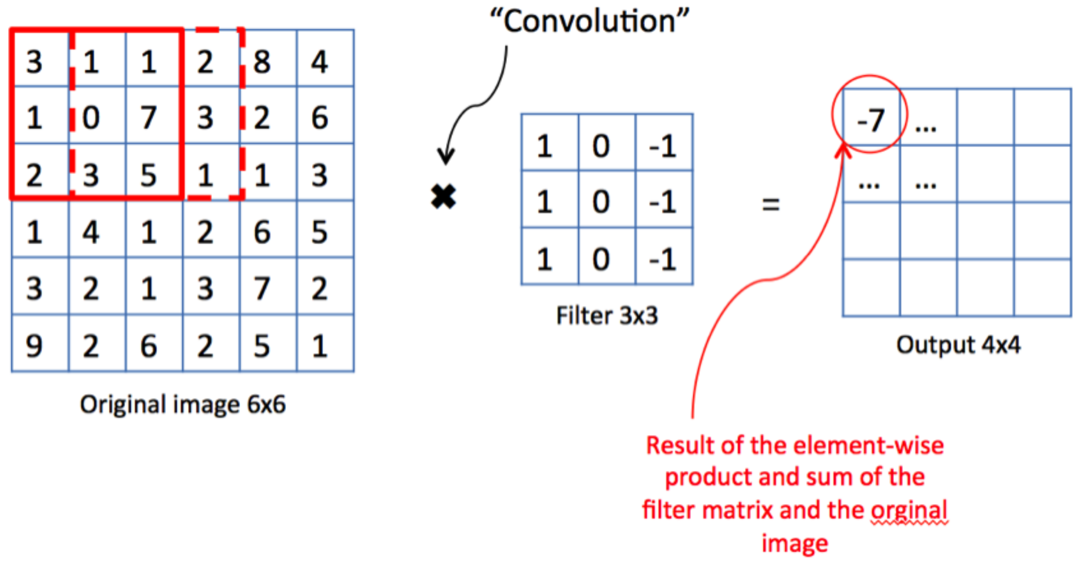


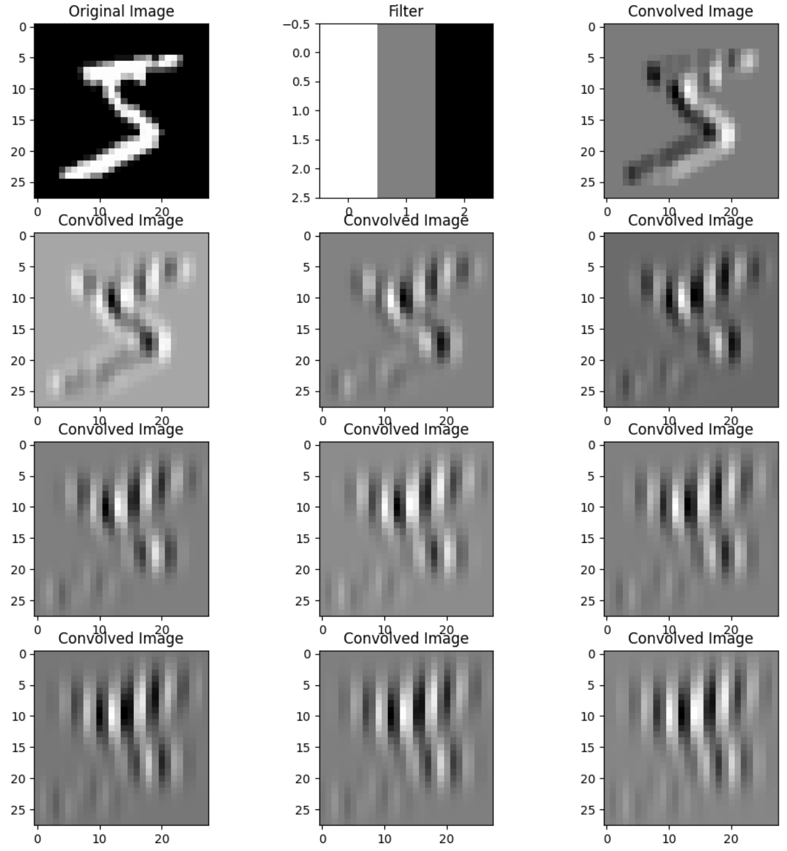
1-2. Convolution Operation



- $g(x,y) = w * f(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} w(i,j) f(x-i, y-j)$

1-2. Convolution Operation - Kernel





- Note that the size of the output may be different from the size of the input.
- The output of the filter will be large if the input has the feature that the filter can detect.

1-2. Convolution Operation - Some Examples of Kernel

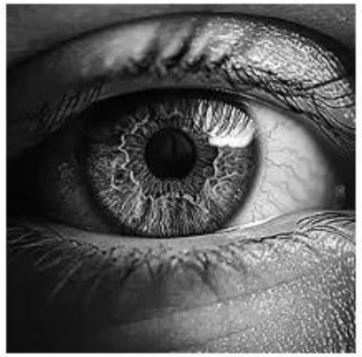
Identity Kernel

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Identity Kernel Matrix and Filtered Image

Sharpen

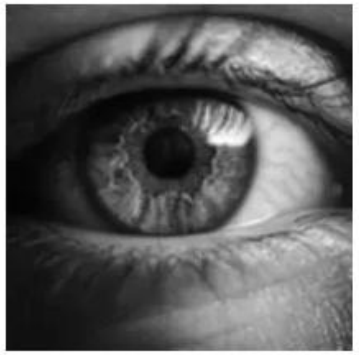
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



Sharpen Kernel Matrix and Filtered Image

Box Blur

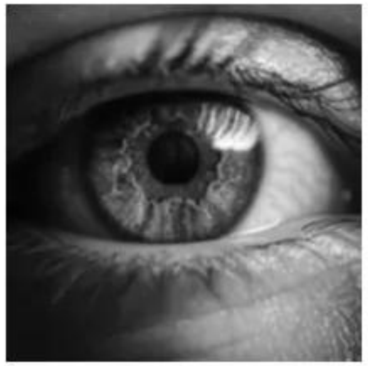
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



Box-blur Kernel Matrix and Filtered Image

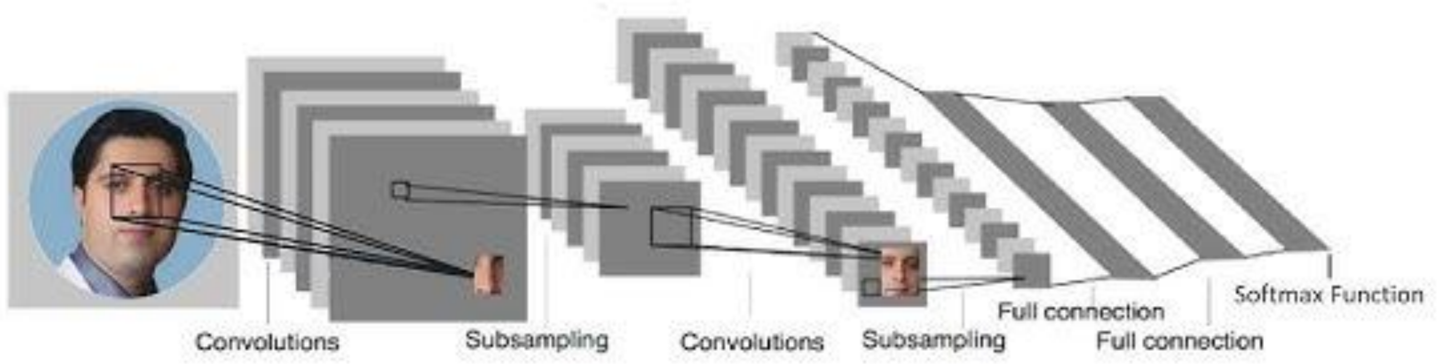
Gaussian Blur

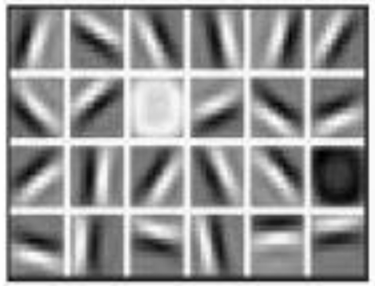
$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

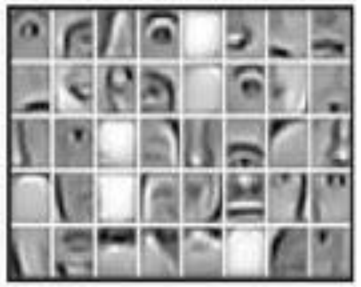


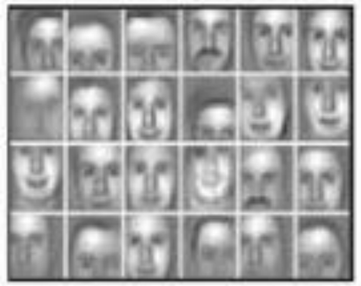
Gaussian Blur Kernel Matrix and Filtered Image

1-3. Learning Parameterized Filters



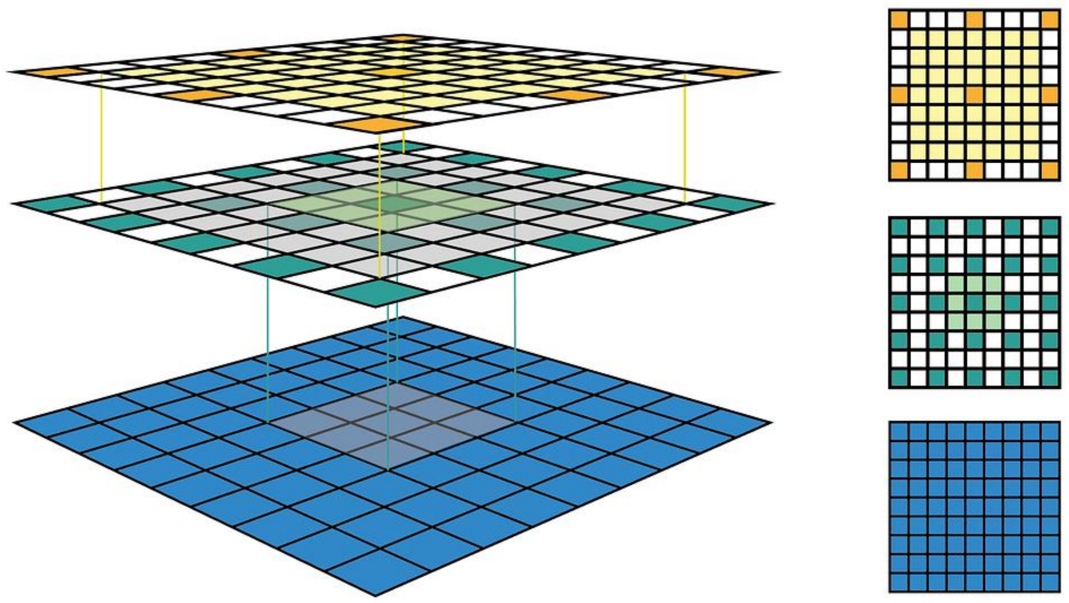






- From hand-crafted filters to learned(data-driven) filters.
- As layer goes deeper, features progress from simple(edges/textures) to complex(parts/objects).

1-4. Receptive Field



- After convolution (and pooling/striding), the feature map spatial size decreases
- However, the receptive field of each unit grows as depth increases.
- Receptive field: The region of the input image that influences one feature in a layer.

1-4. Receptive Field

n :	number of features
r :	receptive field size
j :	jump (distance between two consecutive features)
$start$:	center coordinate of the first feature

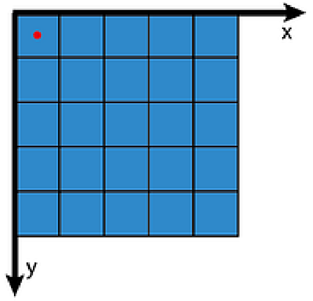
k : convolution kernel size
p : convolution padding size
s : convolution stride size

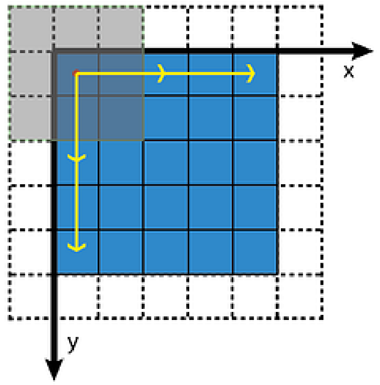
$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$
$$j_{out} = j_{in} * s$$
$$r_{out} = r_{in} + (k-1) * j_{in}$$
$$start_{out} = start_{in} + \left(\frac{k-1}{2} - p\right) * j_{in}$$

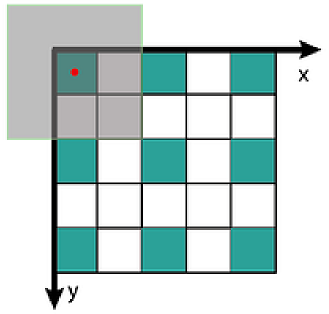
Layer 0: $n_0 = 5;\ r_0 = 1;\ j_0 = 1;$ $start_0 = 0.5$

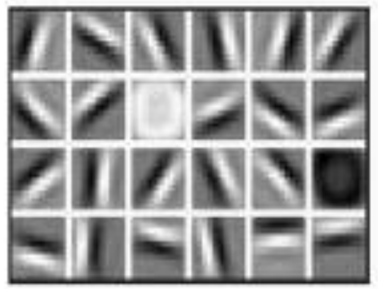
Conv1: $k_1 = 3;\ p_1 = 1;\ s_1 = 2$

Layer 1: $n_1 = 3;\ r_1 = 3;\ j_1 = 2;$ $start_1 = 0.5$

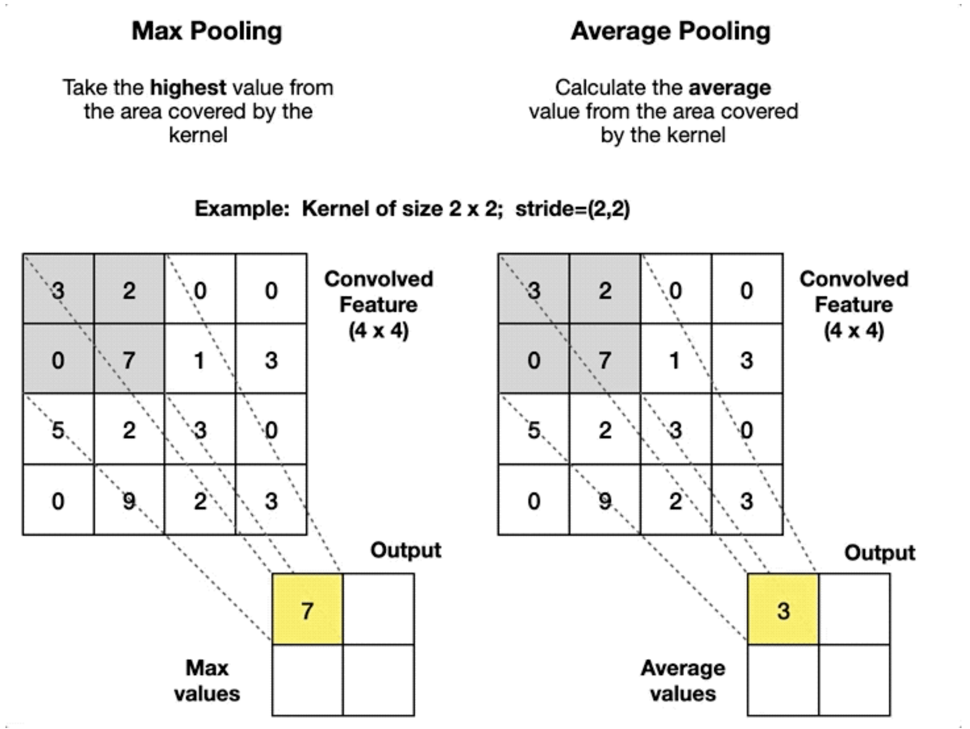




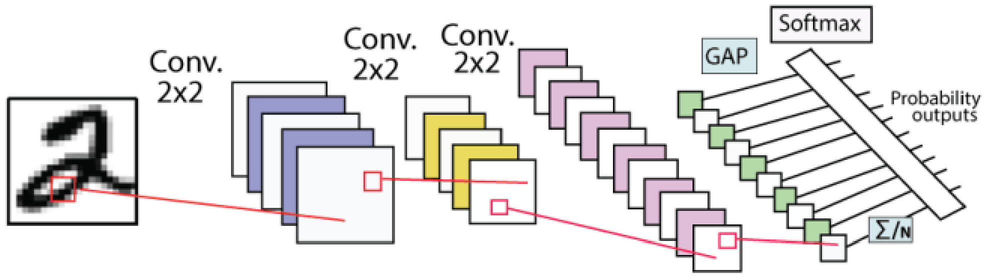




2. Pooling Layer



Max Pooling
Take the **highest** value from the area covered by the kernel
Average Pooling
Calculate the **average** value from the area covered by the kernel
Example: Kernel of size 2 x 2; stride=(2,2)
Convolved Feature (4 x 4)
3 2 0 0
0 7 1 3
5 2 3 0
0 9 2 3
Output
7
Max values
Convolved Feature (4 x 4)
3 2 0 0
0 7 1 3
5 2 3 0
0 9 2 3
Output
3
Average values

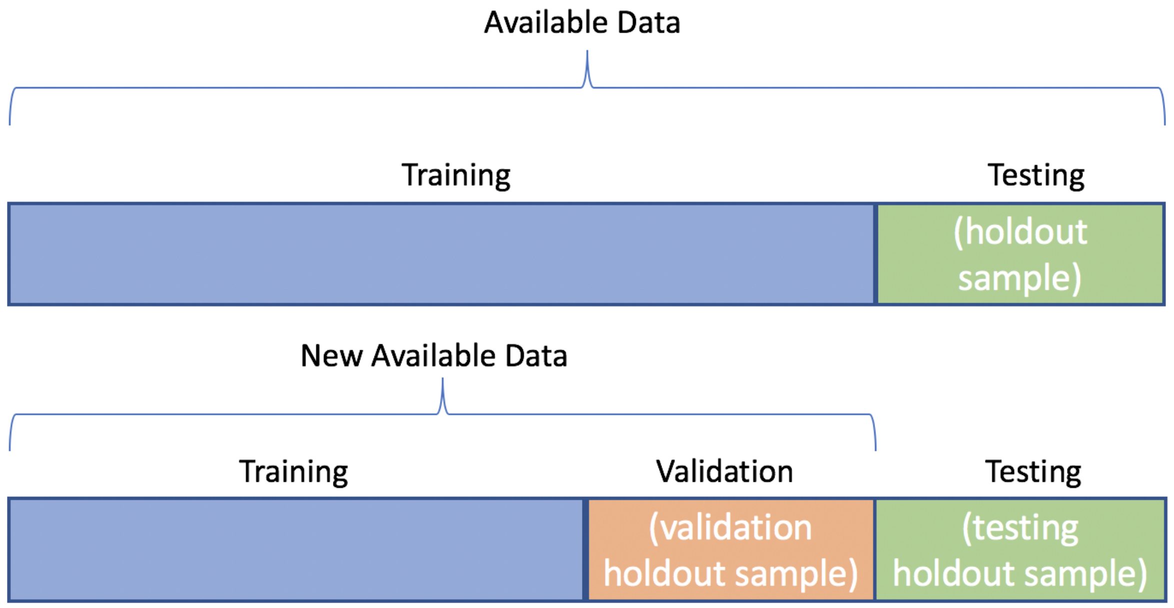


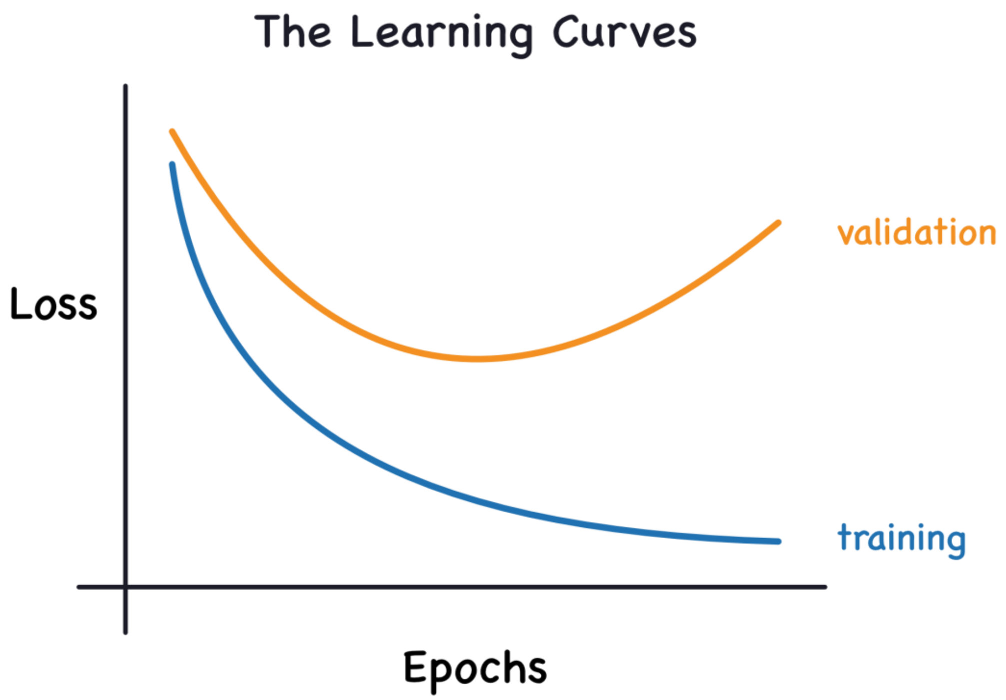
- Pooling layers reduce the dimensions of data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer.
- Max pooling: Take the highest value from the area covered by the kernel
- Average pooling layer: Calculate the average value from the area covered by the kernel
- Global Average Pooling (GAP): (C,H,W) -> (C,) fixes the input shape before passing into fully connected layer.

Overfitting

Overfitting

- What is a good model?

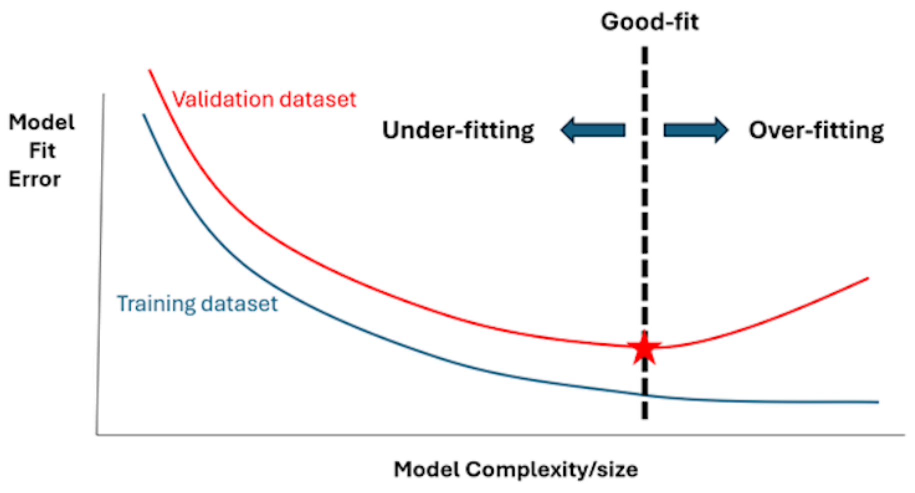


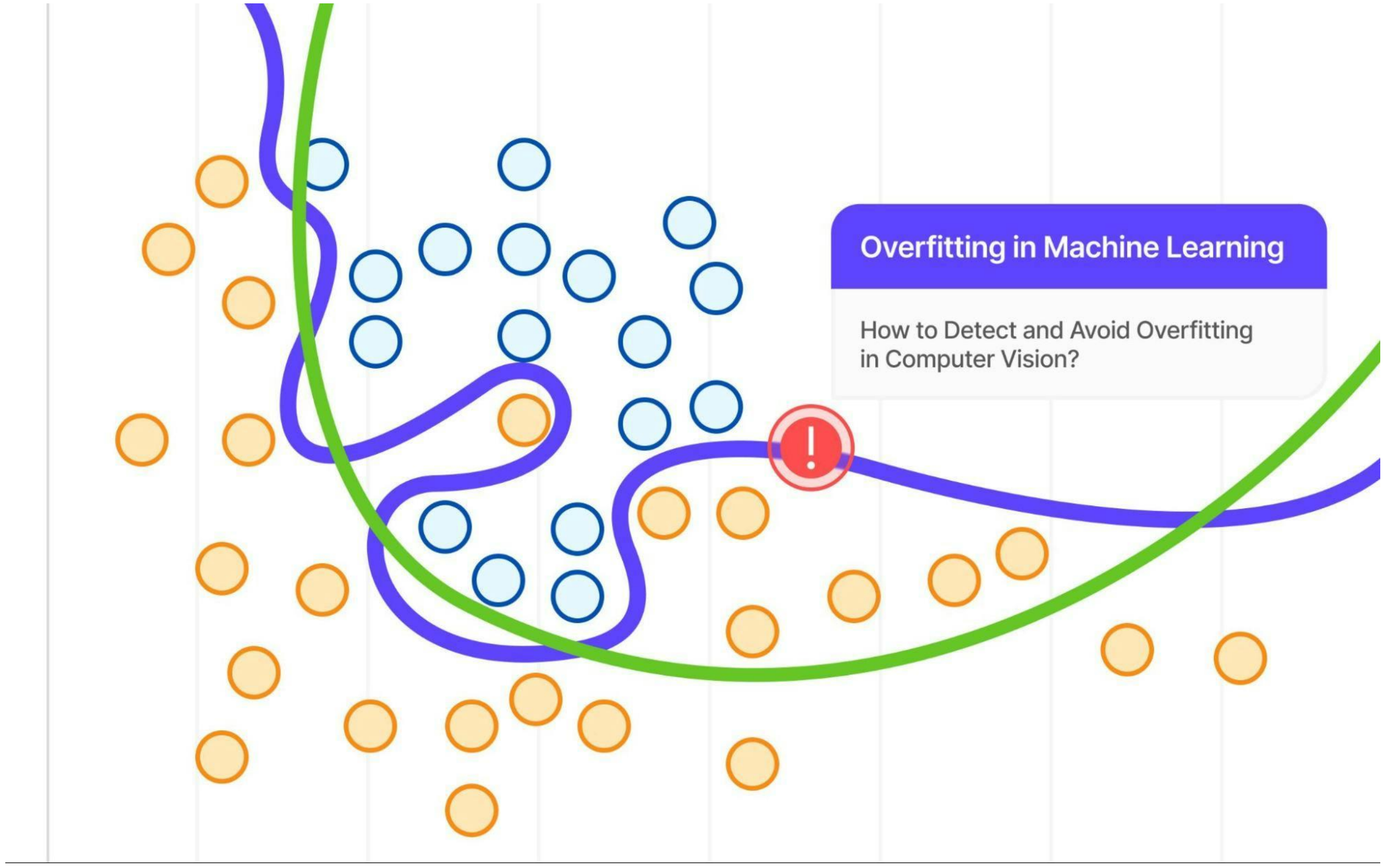


- Overfitting is when a model learns the training data too closely (capturing noise and specific patterns) so it performs well on training data but poorly on unseen data.

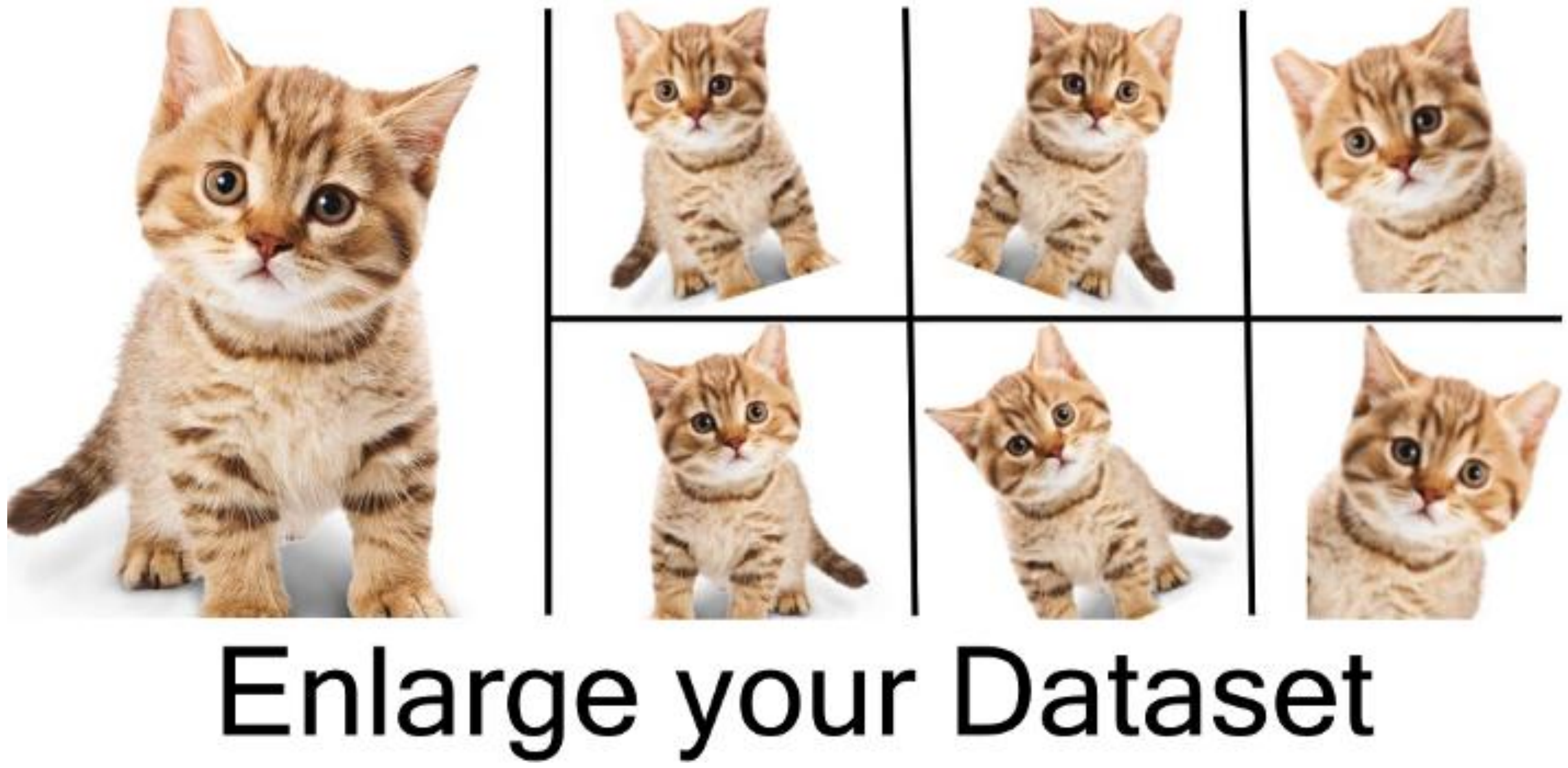
1. Reasons for Overfitting

- Limited training data (⇒ **data augmentation**)
- Noisy or incorrect labels (⇒ **label smoothing** or etc)
- Model too complex (⇒ **decrease parameters**)
- Training for too long (⇒ **early stopping**)
- Weak regularization (⇒ **weight decay**, **dropout**)



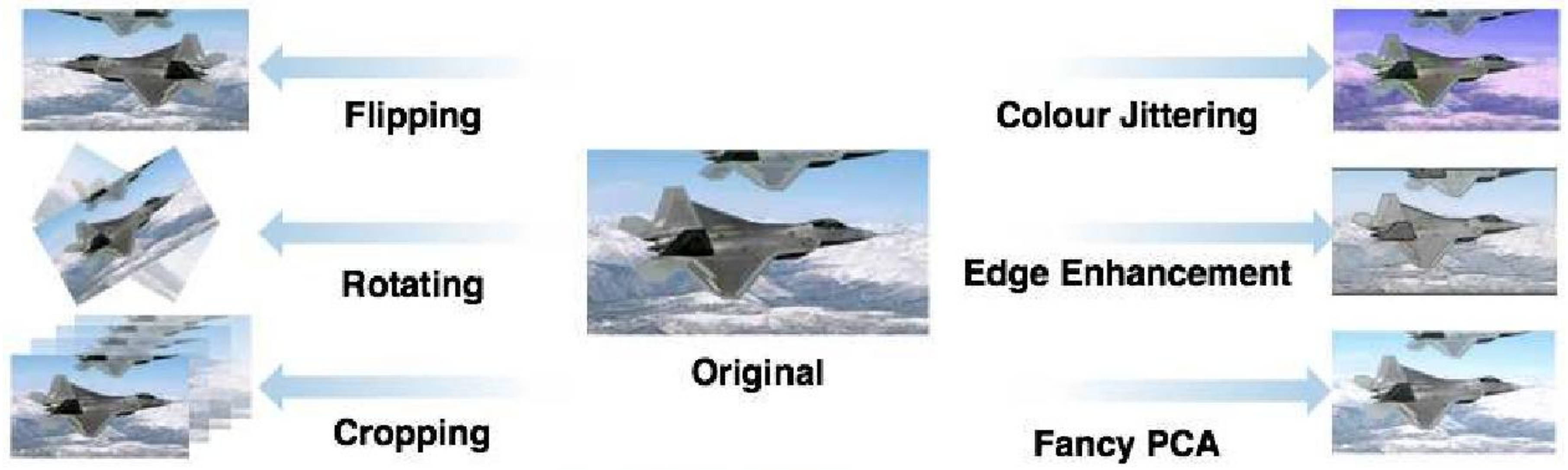

Overfitting in Machine Learning
How to Detect and Avoid Overfitting in Computer Vision?

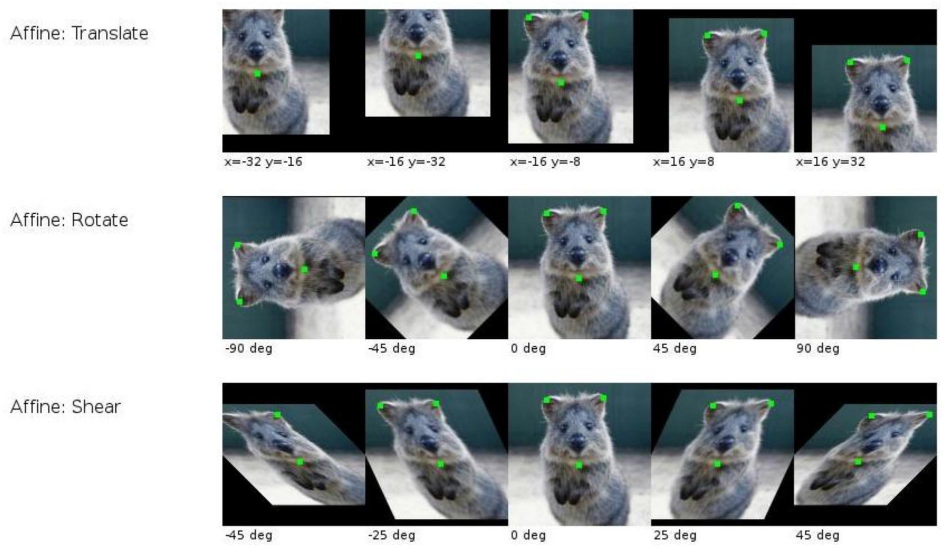
2. Solutions - Data Augmentation



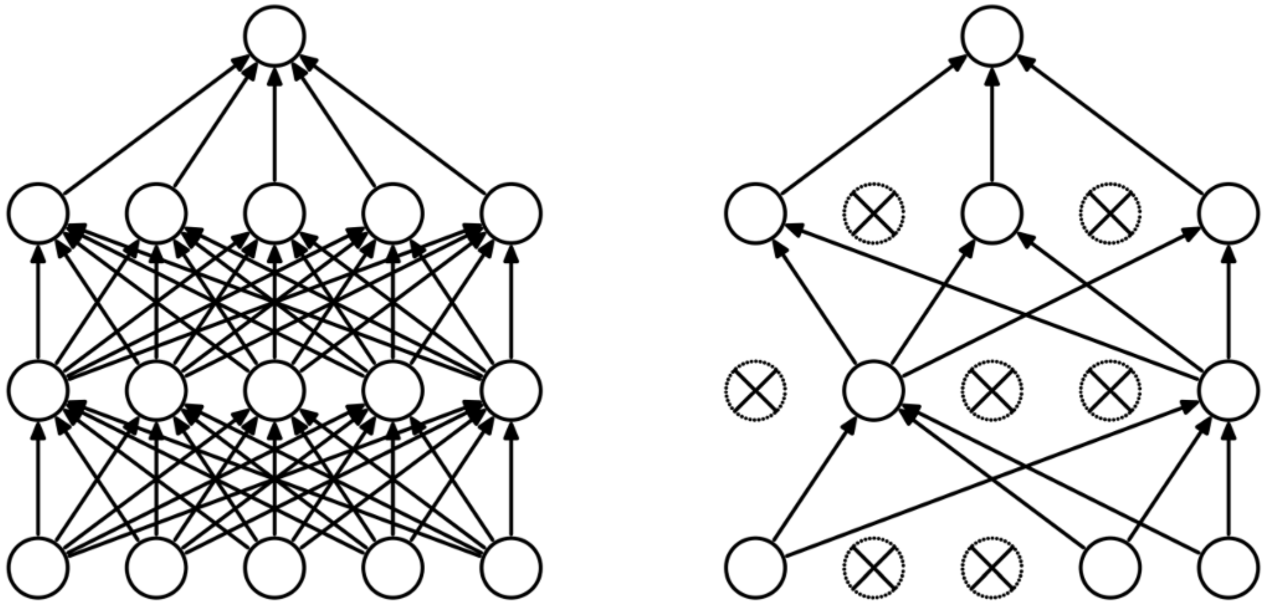
- One way of reducing overfitting.

2. Solutions - Data Augmentation





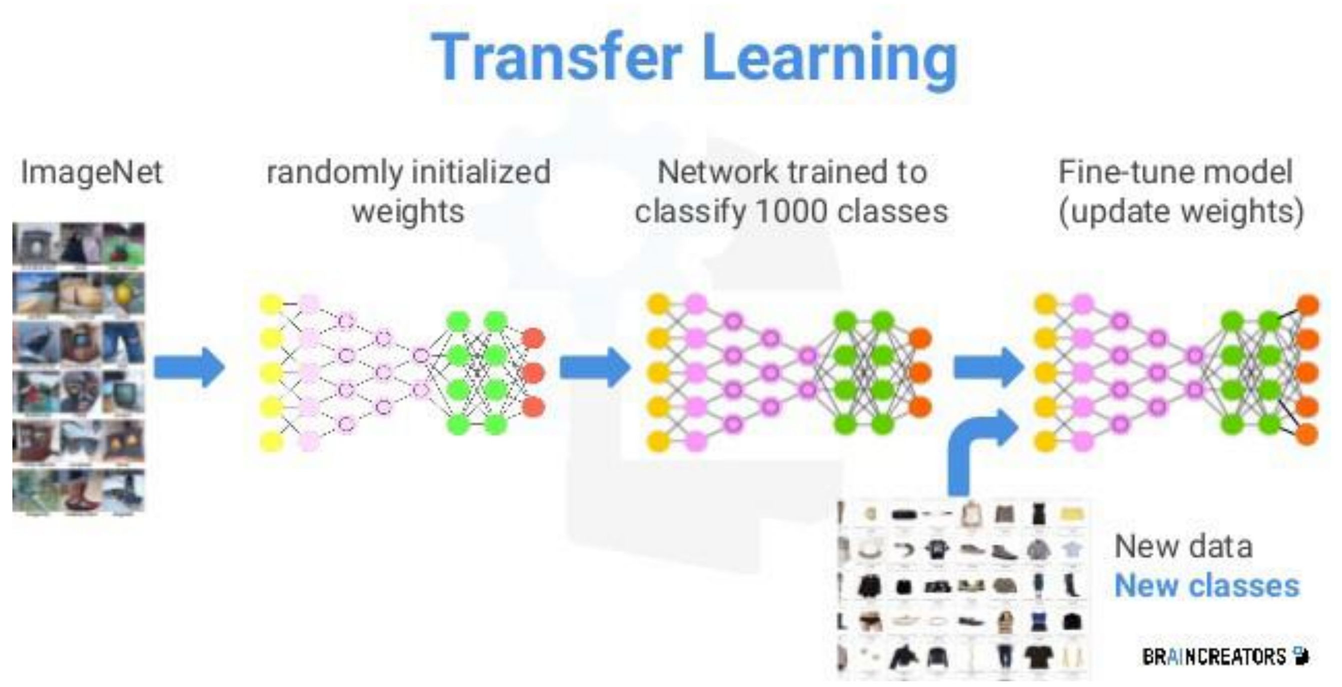
2. Solutions - Dropout


(a) Standard Neural Net (b) After applying dropout.

- Adds noise/regularization effect, which lowers effective model capacity and helps prevent memorization
- At inference, using the full network with scaled activations approximates averaging those subnetworks, improving generalization

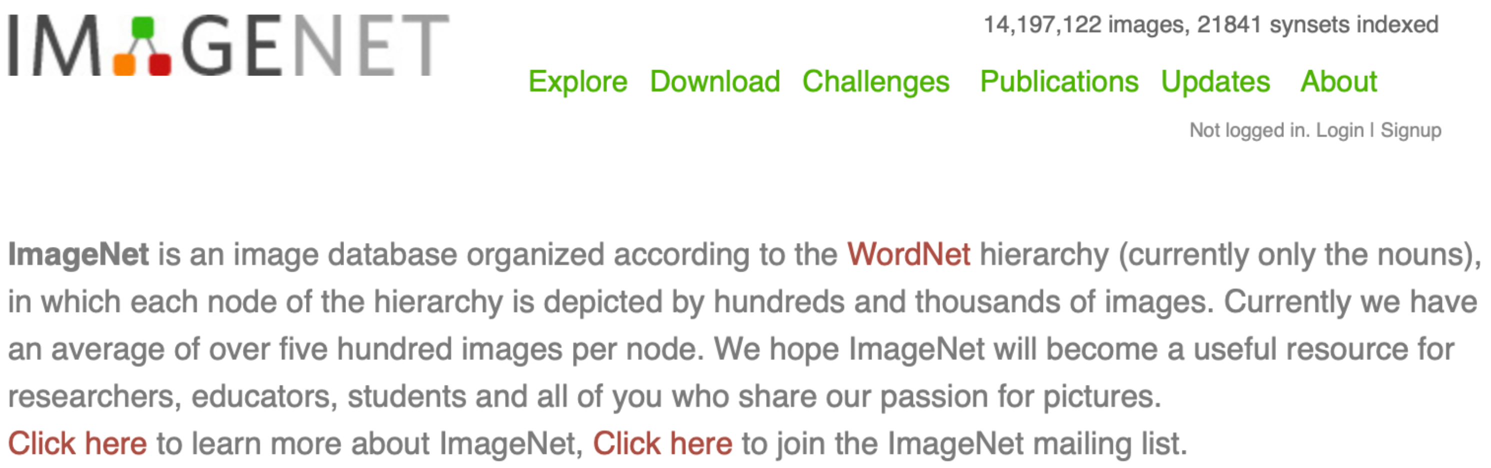
Transfer Learning

2. Transfer Learning


Transfer Learning
ImageNet
randomly initialized weights
Network trained to classify 1000 classes
Fine-tune model (update weights)
New data
New classes
BRAINCREATORS

- The core idea of transfer learning is to get better initial weights to extract good visual features

1. ImageNet


14,197,122 images, 21841 synsets indexed

IMAGENET
Explore Download Challenges Publications Updates About
Not logged in. Login I Signup

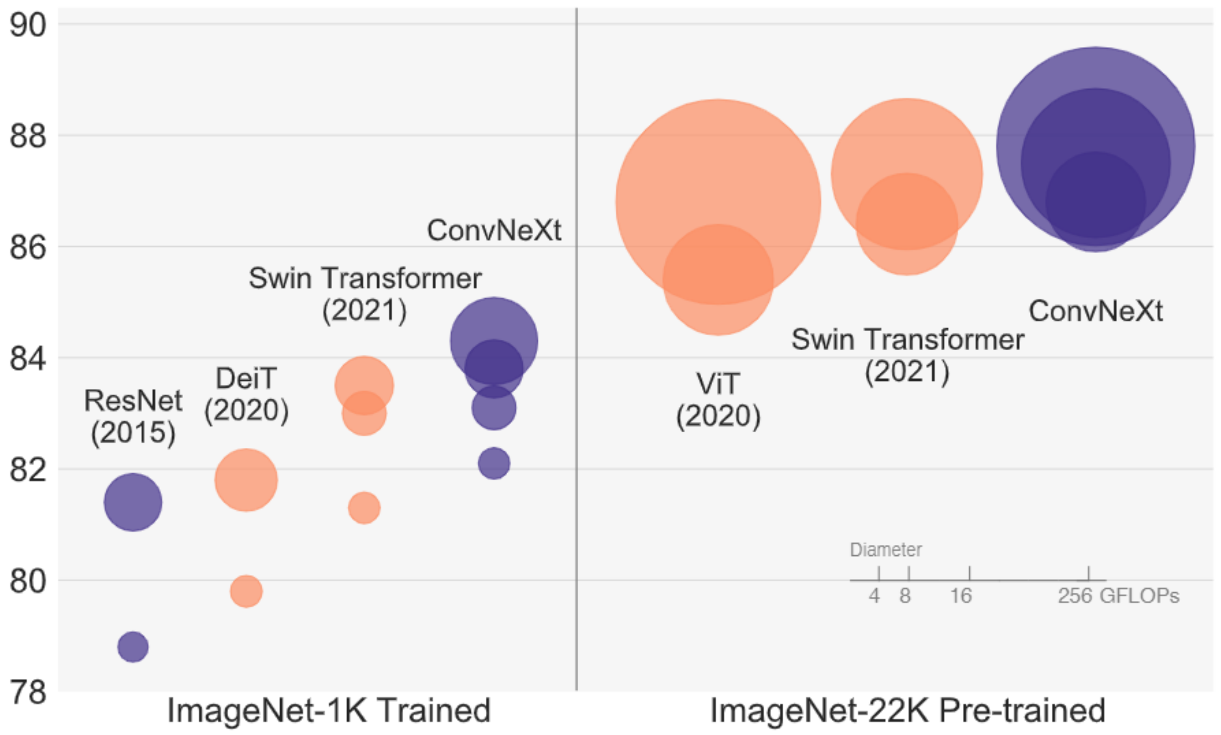
ImageNet is an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. Currently we have an average of over five hundred images per node. We hope ImageNet will become a useful resource for researchers, educators, students and all of you who share our passion for pictures.
Click here to learn more about ImageNet, Click here to join the ImageNet mailing list.

3. CNN SOTA models

Classification

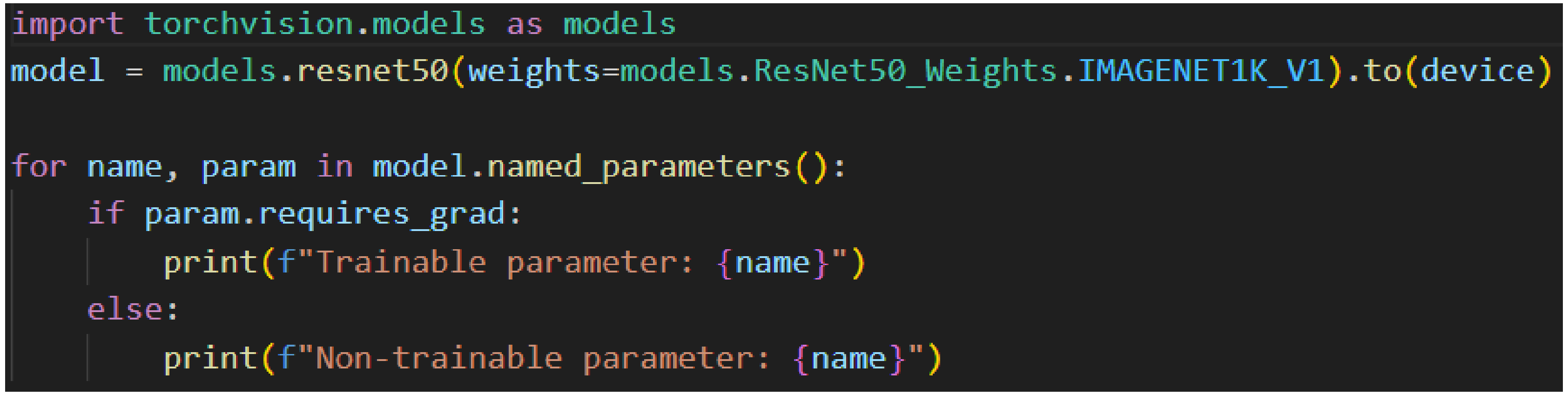
The following classification models are available, with or without pre-trained weights:

- AlexNet
- ConvNeXt
- DenseNet
- EfficientNet
- EfficientNetV2
- GoogLeNet
- Inception V3
- MaxVit
- MNASNet
- MobileNet V2
- MobileNet V3
- RegNet
- ResNet
- ResNeXt
- ShuffleNet V2
- SqueezeNet
- SwinTransformer
- VGG
- VisionTransformer
- Wide ResNet



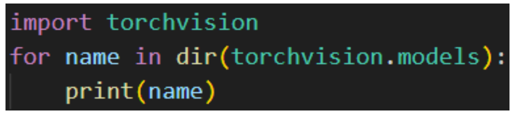
https://docs.pytorch.org/vision/main/models.html
A ConvNet for the 2020s

3. Transfer Learning using PyTorch



```python
import torchvision.models as models
model = models.resnet50(weights=models.ResNet50_Weights.IMAGENET1K_V1).to(device)

for name, param in model.named_parameters():
    if param.requires_grad:
        print(f"Trainable parameter: {name}")
    else:
        print(f"Non-trainable parameter: {name}")
```



```python
import torchvision
for name in dir(torchvision.models):
    print(name)
```

Check pretrained models in torchvision without resnet50

- Load pretrained model at torchvision.models
- Check model's layer name & trainable / non-trainable parameters

Question

- GitHub에 공유된 코드는 직접 제작한 모델입니다. 바로 위 페이지의 코드를 참고하여 transfer learning을 직접 구현해보고, 결과를 notion에 공유해주세요.