Deeplearning basic

220603

고 우 영

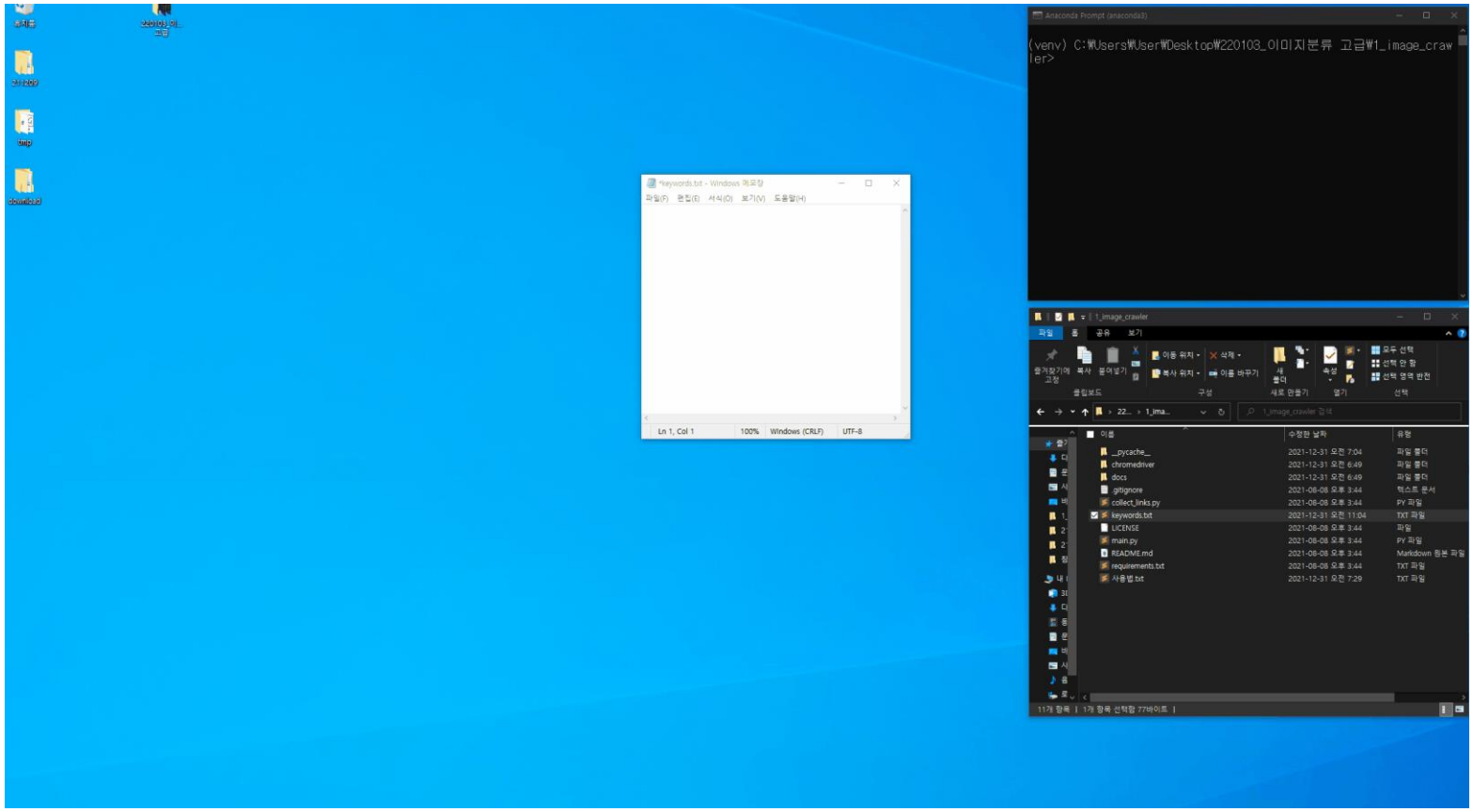
Contents

- **강의자료**
  - https://github.com/airobotlab/advanced_cnn

Contents

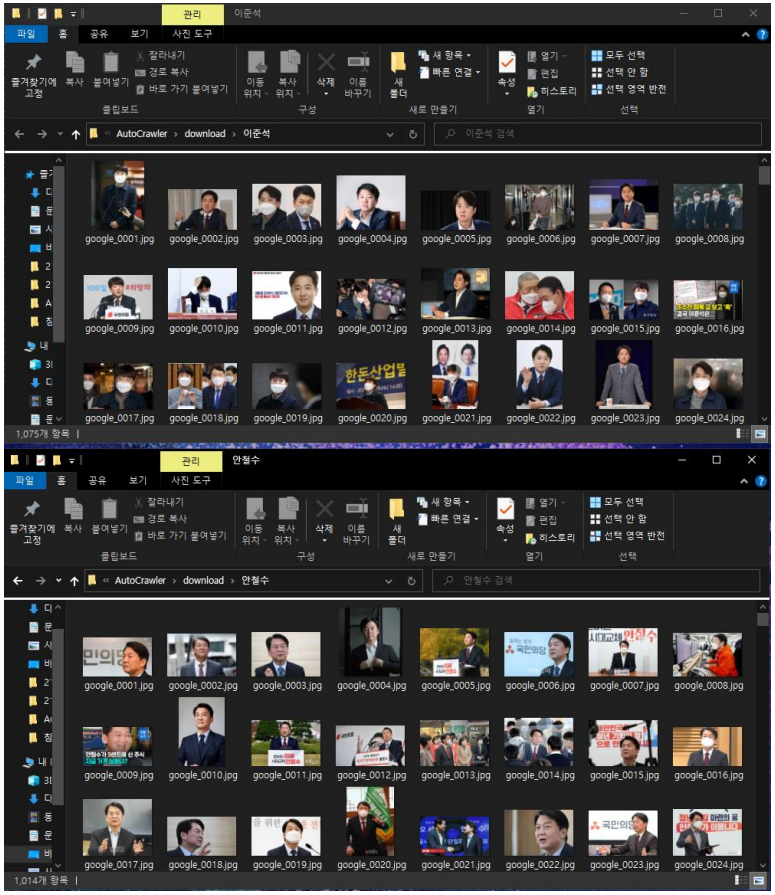
ACVC - 0) Crawling 실습

- 이미지 분류 고급 (ACVC, Advanced Computer Vision Classification)
  - https://keep-steady.tistory.com/29

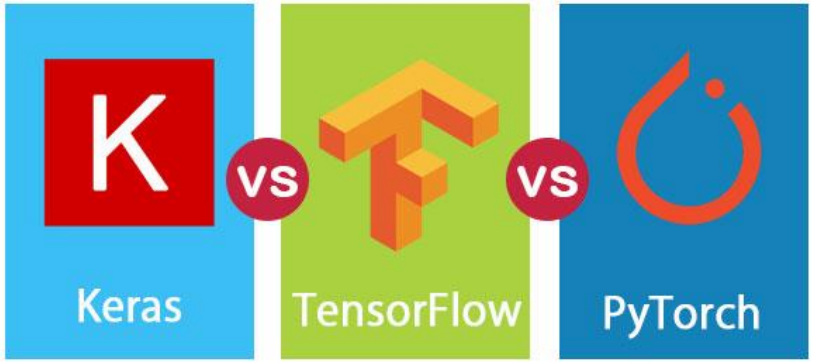


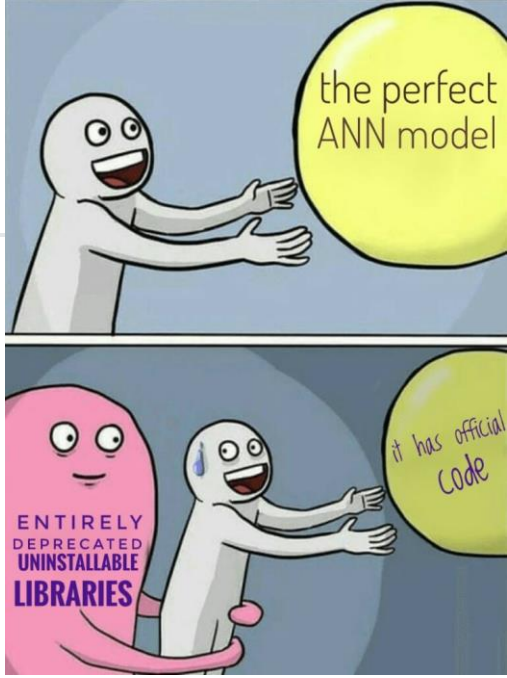
ACVC - 0) Crawling 실습

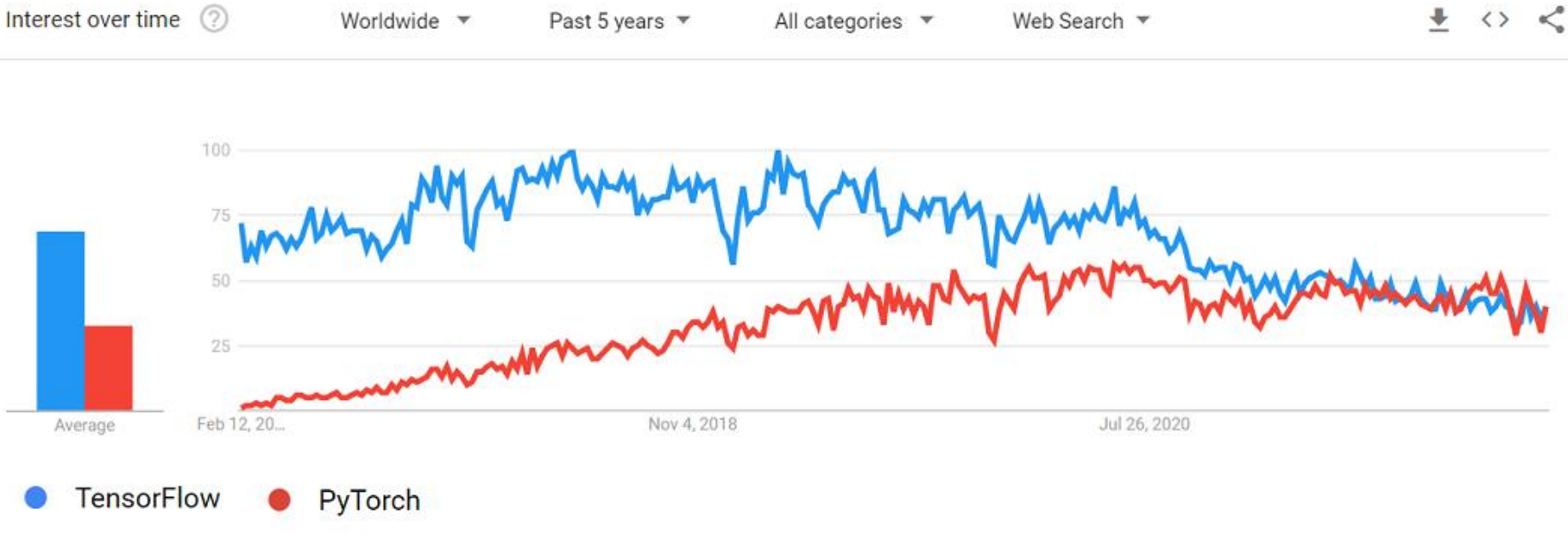
- 이미지 분류 고급 (ACVC, Advanced Computer Vision Classification)
  - https://keep-steady.tistory.com/29



Pytorch VS Tensorflow

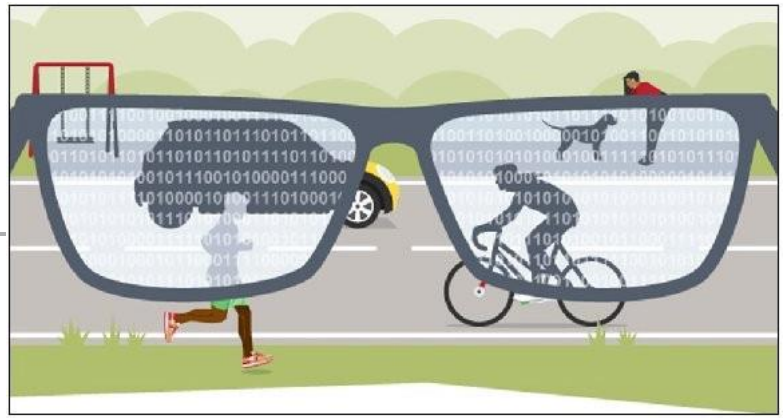




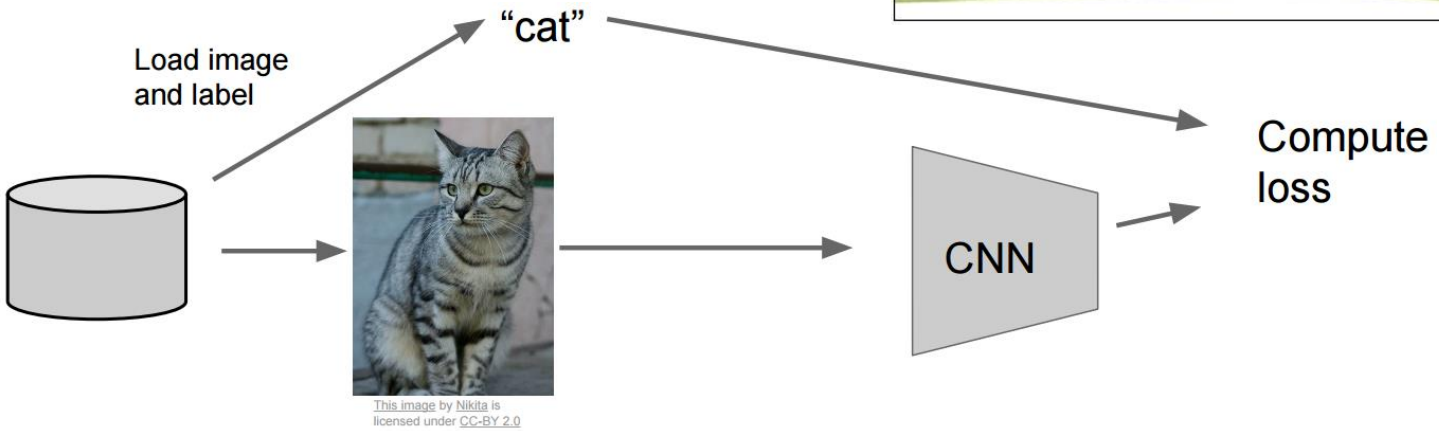


https://www.facebook.com/photo?fbid=10159796144121007&set=gm.[illegible]
https://buggyprogrammer.com/pytorch-vs-tensorflow-which-one-is-better/
https://www.educba.com/keras-vs-tensorflow-vs-pytorch/

ACVC – 1) CNN 복습



- 이미지 분류



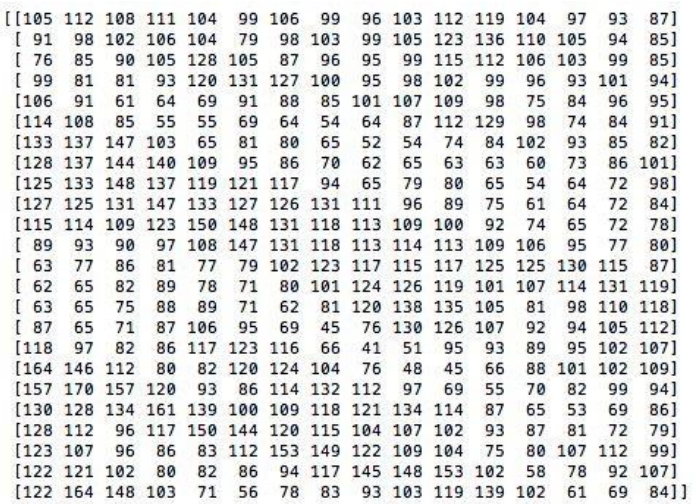
This image by Nikita is licensed under CC-BY 2.0

```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  85 101 107 109  98  75  84  96  95]
 [114 108  85  55  55  69  64  54  64  87 112 129  98  74  84  91]
 [133 137 147 103  65  81  80  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  70  62  65  63  63  60  73  86 101]
 [125 133 148 137 119 121 117  94  65  79  80  65  54  64  72  98]
 [127 125 131 147 133 127 126 131 111  96  89  75  61  64  72  84]
 [115 114 109 123 150 148 131 118 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 118 113 114 113 109 106  95  77  80]
 [ 63  77  86  81  77  79 102 123 117 115 117 125 125 130 115  87]
 [ 62  65  82  89  78  71  80 101 124 126 119 101 107 114 131 119]
 [ 63  65  75  88  89  71  62  81 120 138 135 105  81  98 110 118]
 [ 87  65  71  87 106  95  69  45  76 130 126 107  92  94 105 112]
 [118  97  82  86 117 123 116  66  41  51  95  93  89  95 102 107]
 [164 146 112  80  82 120 124 104  76  48  45  66  88 101 102 109]
 [157 170 157 120  93  86 114 132 112  97  69  55  70  82  99  94]
 [130 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [128 112  96 117 150 144 120 115 104 107 102  93  87  81  72  79]
 [123 107  96  86  83 112 153 149 122 109 104  75  80 107 112  99]
 [122 121 102  80  82  86  94 117 145 148 153 102  58  78  92 107]
 [122 164 148 103  71  56  78  83  93 103 119 139 102  61  69  84]]
```

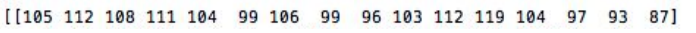

https://medium.com/@divyam.vashisht/image-segmentation-on-indian-traffic-dataset-587842fee63a

ACVC – 1) CNN 복습

- 이미지처리 vs 자연어처리

f()=고양이

```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  85 101 107 109  98  75  84  96  95]
 [114 108  85  55  55  69  64  54  64  87 112 129  98  74  84  91]
 [133 137 147 103  65  81  80  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  70  62  65  63  63  60  73  86 101]
 [125 133 148 137 119 121 117  94  65  79  80  65  54  64  72  98]
 [127 125 131 147 133 127 126 131 111  96  89  75  61  64  72  84]
 [115 114 109 123 150 148 131 118 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 118 113 114 113 109 106  95  77  80]
 [ 63  77  86  81  77  79 102 123 117 115 117 125 125 130 115  87]
 [ 62  65  82  89  78  71  80 101 124 126 119 101 107 114 131 119]
 [ 63  65  75  88  89  71  62  81 120 138 135 105  81  98 110 118]
 [ 87  65  71  87 106  95  69  45  76 130 126 107  92  94 105 112]
 [118  97  82  86 117 123 116  66  41  51  95  93  89  95 102 107]
 [164 146 112  80  82 120 124 104  76  48  45  66  88 101 102 109]
 [157 170 157 120  93  86 114 132 112  97  69  55  70  82  99  94]
 [130 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [128 112  96 117 150 144 120 115 104 107 102  93  87  81  72  79]
 [123 107  96  86  83 112 153 149 122 109 104  75  80 107 112  99]
 [122 121 102  80  82  86  94 117 145 148 153 102  58  78  92 107]
 [122 164 148 103  71  56  78  83  93 103 119 139 102  61  69  84]]
```

f()=부정

‘나는 지금 배가 많이 고파요’

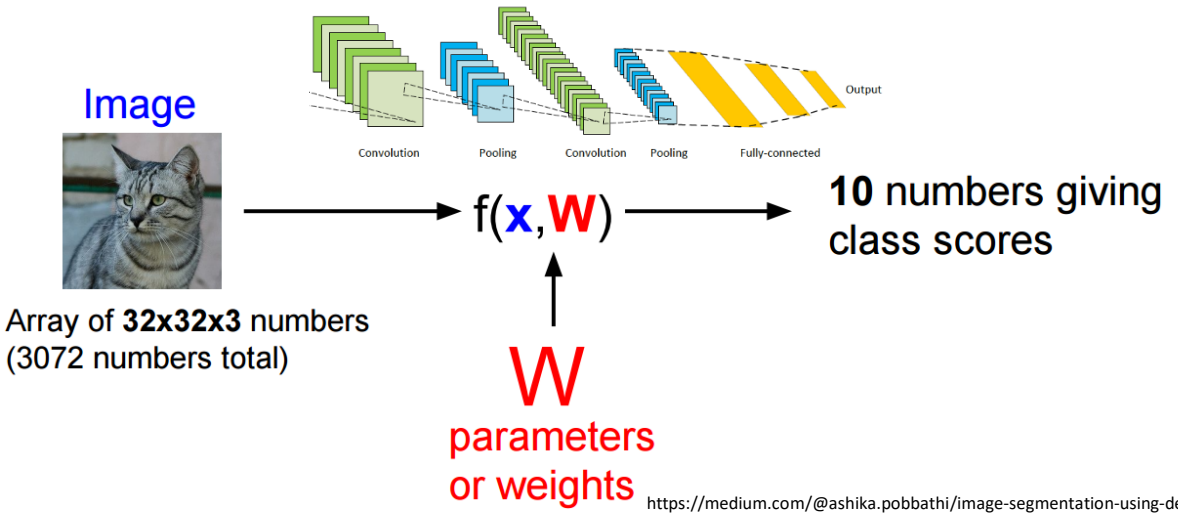
ACVC – 1) CNN 복습

- 이미지처리 vs 자연어처리

f(

```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  85 101 107 109  98  75  84  96  95]
 [114 108  85  55  55  69  64  54  64  87 112 129  98  74  84  91]
 [133 137 147 103  65  81  80  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  70  62  65  63  63  60  73  86 101]
 [125 133 148 137 119 121 117  94  65  79  80  65  54  64  72  98]
 [127 125 131 147 133 127 126 131 111  96  89  75  61  64  72  84]
 [115 114 109 123 150 148 131 118 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 118 113 114 113 109 106  95  77  80]
 [ 63  77  86  81  77  79 102 123 117 115 117 125 125 130 115  87]
 [ 62  65  82  89  78  71  80 101 124 126 119 101 107 114 131 119]
 [ 63  65  75  88  89  71  62  81 120 138 135 105  81  98 110 118]
 [ 87  65  71  87 106  95  69  45  76 130 126 107  92  94 105 112]
 [118  97  82  86 117 123 116  66  41  51  95  93  89  95 102 107]
 [164 146 112  80  82 120 124 104  76  48  45  66  88 101 102 109]
 [157 170 157 120  93  86 114 132 112  97  69  55  70  82  99  94]
 [130 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [128 112  96 117 150 144 120 115 104 107 102  93  87  81  72  79]
 [123 107  96  86  83 112 153 149 122 109 104  75  80 107 112  99]
 [122 121 102  80  82  86  94 117 145 148 153 102  58  78  92 107]
 [122 164 148 103  71  56  78  83  93 103 119 139 102  61  69  84]]
```

)=고양이



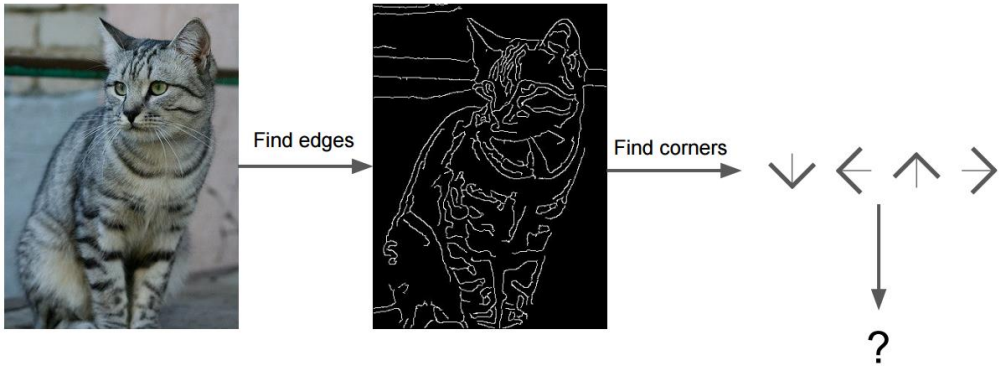
https://medium.com/@ashika.pobbathi/image-segmentation-using-deep-learning-models-and-datasets-98609c02a2b6

ACVC – 1) CNN 복습

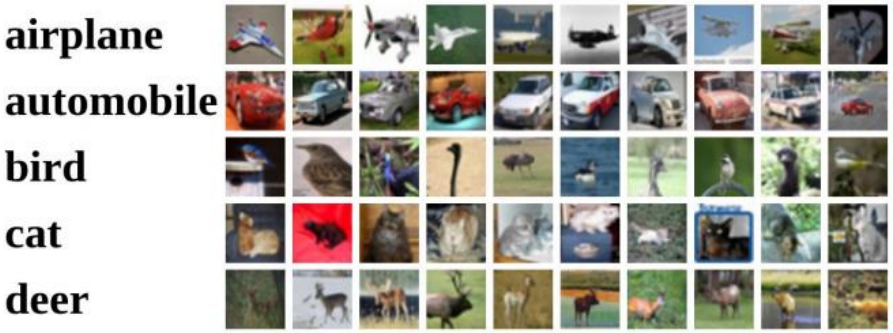
- 이미지처리의 어려움
  - Illumination
  - Deformation
  - Occlusion
  - Background Clutter
  - Intra-class variation

ACVC – 1) CNN 복습

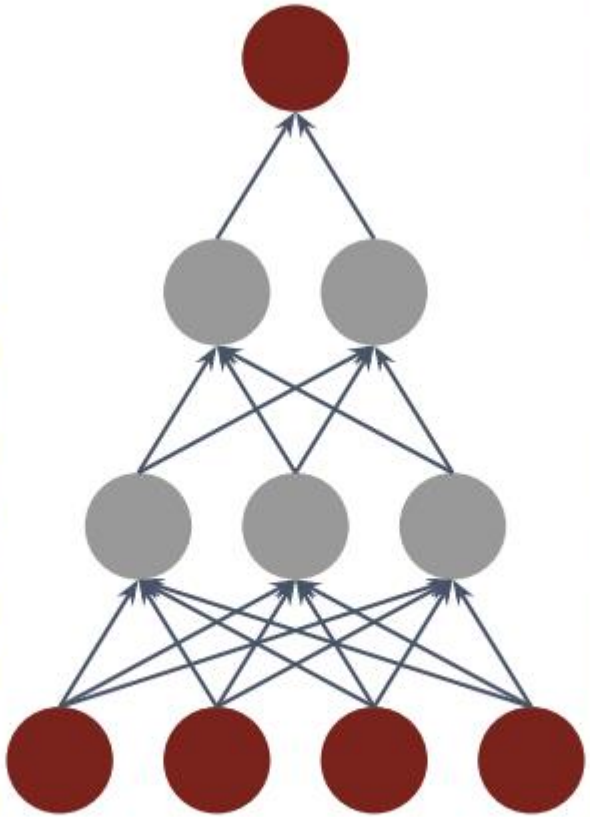
- Deeplearning 이전 방법



- Deeplearning, Data-Driven Approach
  - 데이터(image&label)를 왕창 수집
  - DL로 스스로 feature를 학습
  - Classifier로 분류

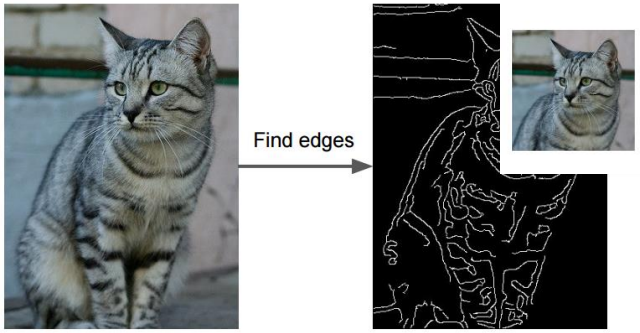


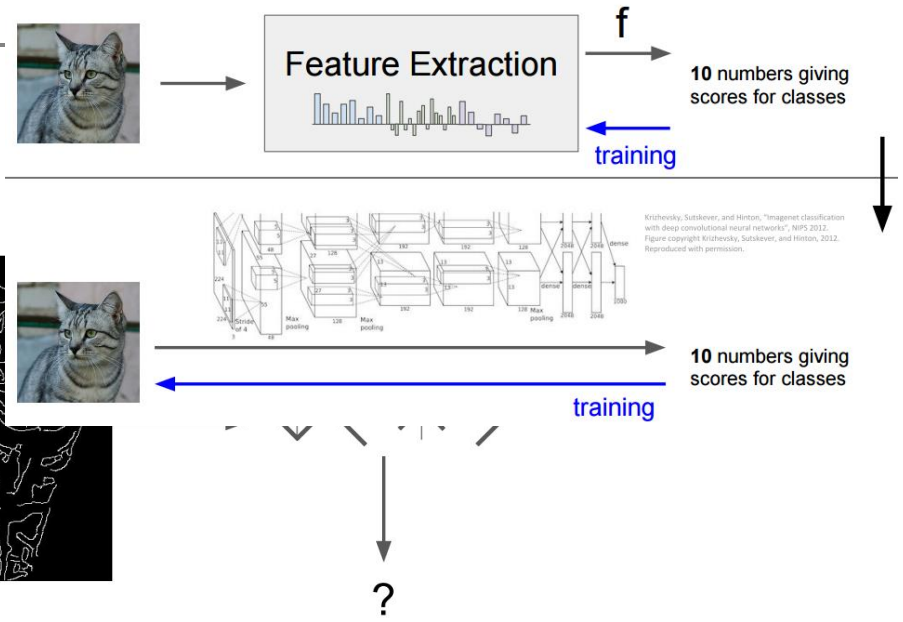
ACVC – 1) CNN 복습



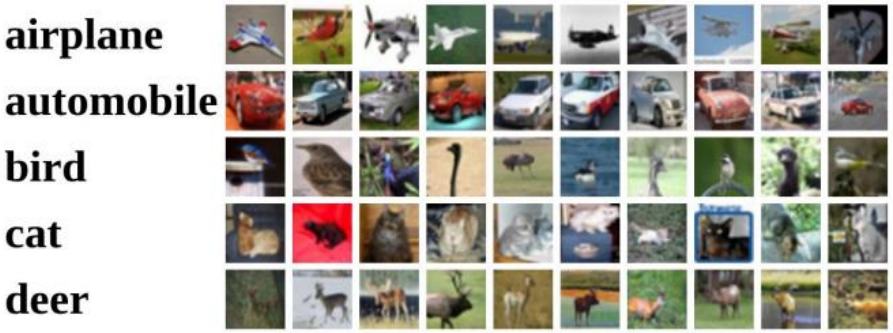
ACVC – 1) CNN 복습

- Deeplearning 이전 방법



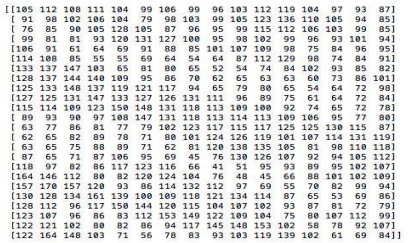


- Deeplearning, Data-Driven Approach
  - 데이터(image&label)를 왕창 수집
  - DL로 스스로 feature를 학습
  - Classifier로 분류



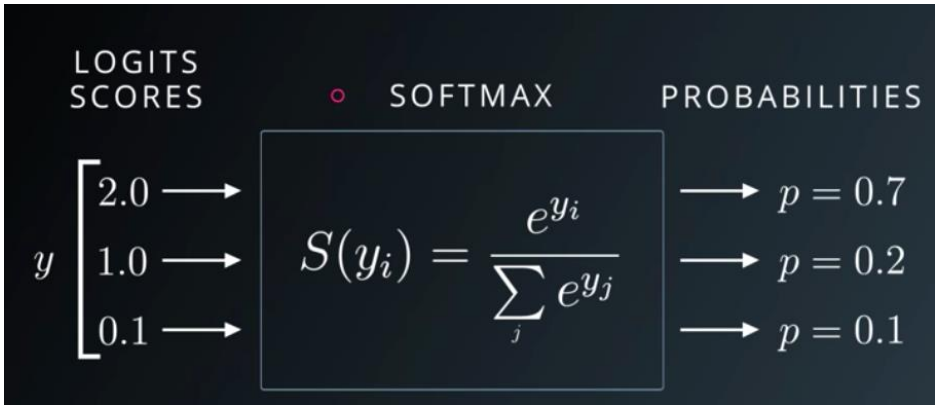
ACVC – 1) CNN 복습

- Multi-class classification



f( )=

cat
car
frog



1
0
0

Label

https://stackoverflow.com/questions/52658204/how-the-parameters-of-softmax-are-obtained-in-tensorflow-if-the-softmax-is-not-i

Jetbot Software-Collision Avoidance 1

Collision Avoidance - Train Model (ResNet18)

Welcome to this host side Jupyter Notebook! This should look familiar if you ran through the notebooks that run on the robot. In this notebook we'll train our image classifier to detect two classes `free` and `blocked`, which we'll use for avoiding collisions. For this, we'll use a popular deep learning library *PyTorch*

```
import torch
import torch.optim as optim
import torch.nn.functional as F
import torchvision
import torchvision.datasets as datasets
import torchvision.models as models
import torchvision.transforms as transforms
```

Upload and extract dataset

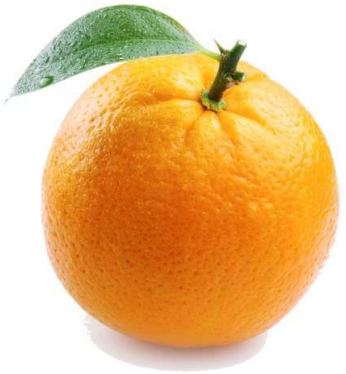
Before you start, you should upload the `dataset.zip` file that you created in the `data_collection.ipynb` notebook on the robot.

You should then extract this dataset by calling the command below

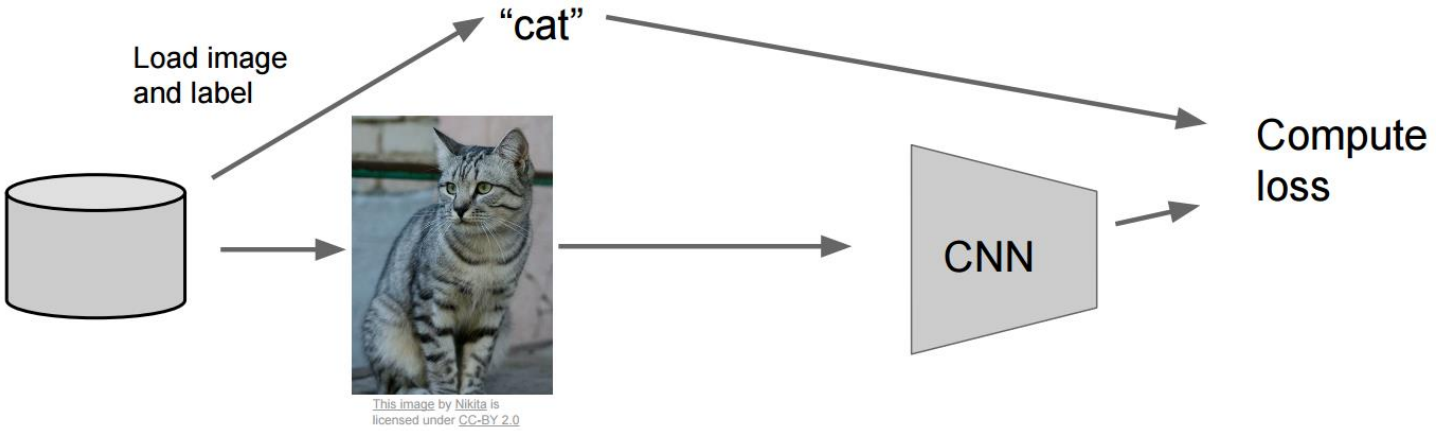
```
!unzip -q dataset.zip
```

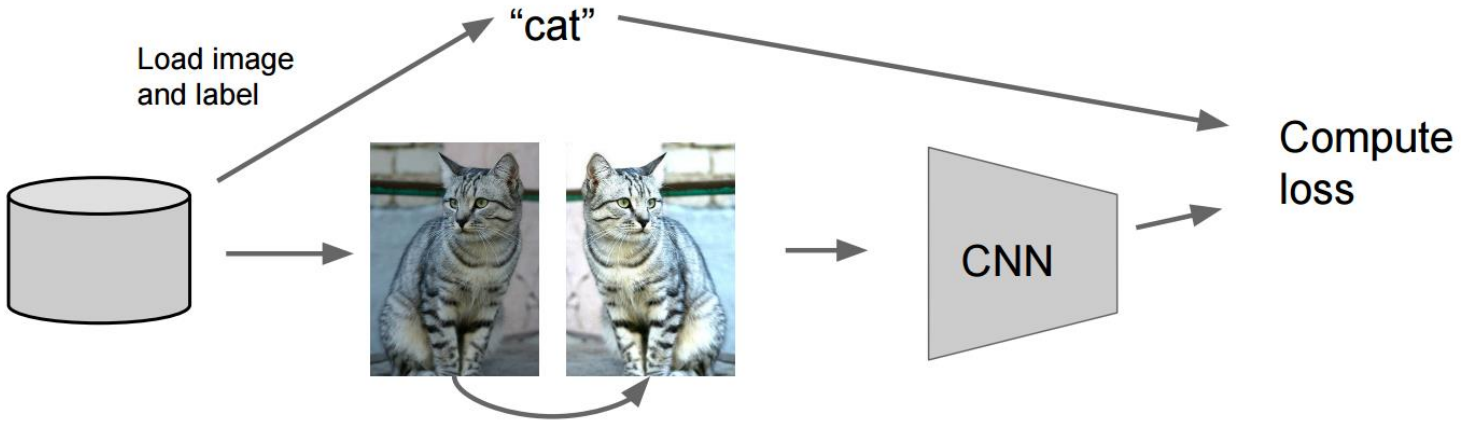
You should see a folder named `dataset` appear in the file browser.

이미지 분류모델 학습 -1



- **Convert data & augmentation**





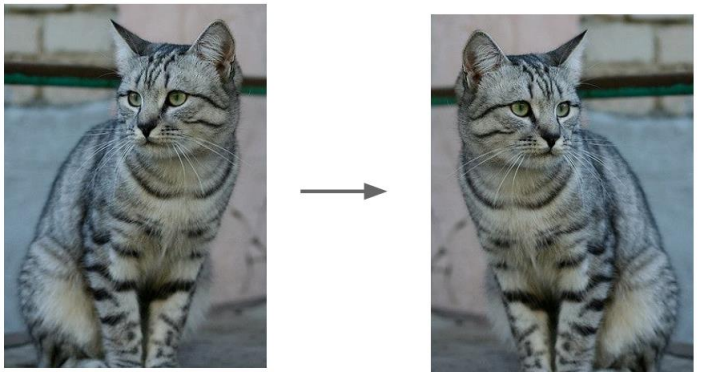
Jetbot Software-Collision Avoidance 1

Create dataset instance

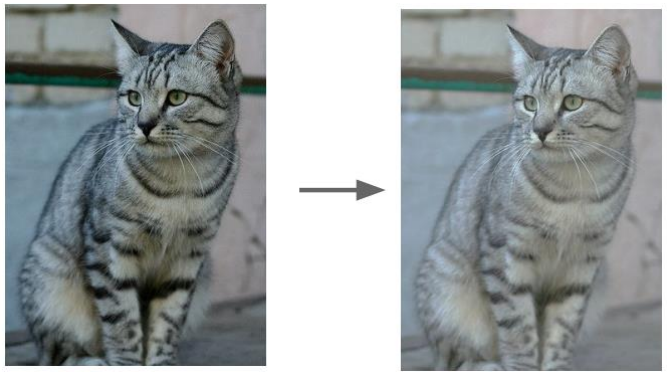
Now we use the ImageFolder dataset class available with the torchvision.datas
attach transforms from the torchvision.transforms package to prepare the data for training.

```
dataset = datasets.ImageFolder(
    'dataset',
    transforms.Compose([
        transforms.ColorJitter(0.1, 0.1, 0.1, 0.1),
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ])
)
```

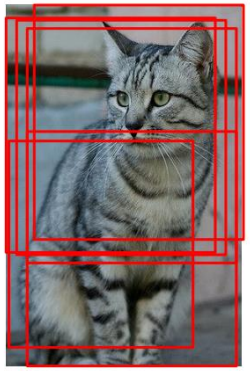
Random crops&scales: 임의로
자르고 크기변경



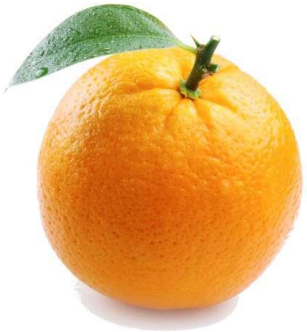
Flip: 수평/수직 회전



Color Jitter: 밝기 대비 변경



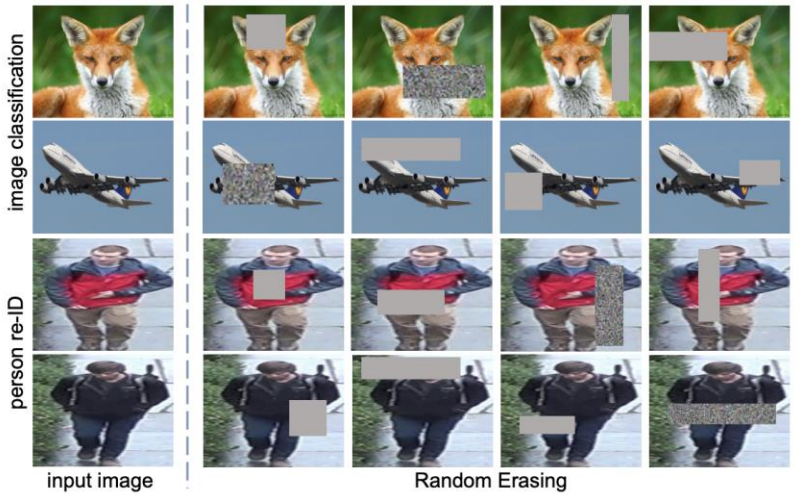
이미지 분류모델 학습 -1

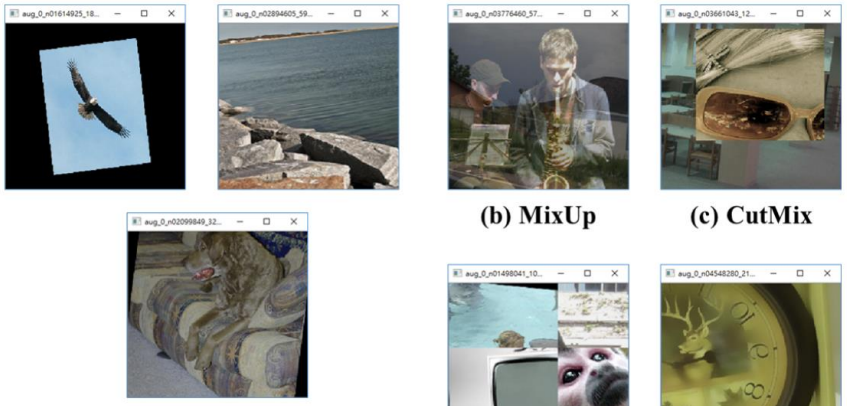


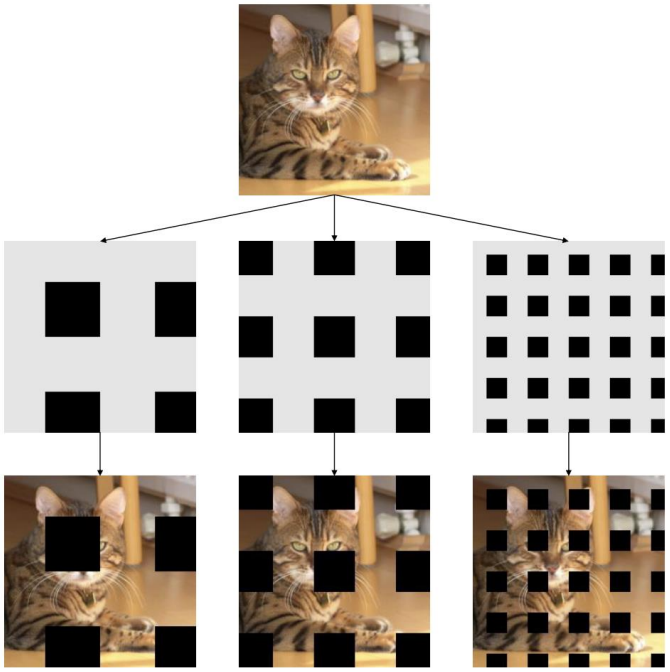
- **Convert data & augmentation**
  - transforms.ToPILImage() - csv 파일로 데이터셋을 받을 경우, PIL image로 바꿔준다.
  - transforms.CenterCrop(size) - 가운데 부분을 size 크기로 자른다.
  - transforms.Grayscale(num_output_channels=1) - grayscale로 변환한다.
  - transforms.RandomAffine(degrees) - 랜덤으로 affine 변형을 한다.
  - transforms.RandomCrop(size) -이미지를 랜덤으로 아무데나 잘라 size 크기로 출력한다.
  - transforms.RandomResizedCrop(size) - 이미지 사이즈를 size로 변경한다
  - transforms.Resize(size) - 이미지 사이즈를 size로 변경한다
  - transforms.RandomRotation(degrees) 이미지를 랜덤으로 degrees 각도로 회전한다.
  - transforms.RandomResizedCrop(size, scale=(0.08, 1.0), ratio=(0.75, 1.3333333333333333)) - 이미지를 랜덤으로 변형한다.
  - transforms.RandomVerticalFlip(p=0.5) - 이미지를 랜덤으로 수직으로 뒤집는다. p=0이면 뒤집지 않는다.
  - transforms.RandomHorizontalFlip(p=0.5) - 이미지를 랜덤으로 수평으로 뒤집는다.
  - transforms.ToTensor() - 이미지 데이터를 tensor로 바꿔준다.
  - transforms.Normalize(mean, std, inplace=False) - 이미지를 정규화한다.

이미지 분류모델 학습 -2

- 다양한 Augmentation 방법
  - Random erasing, Grid mask
  - Crop, Rotation, Flip, ...
  - MixUP, CutMix, Mosaic, Blur







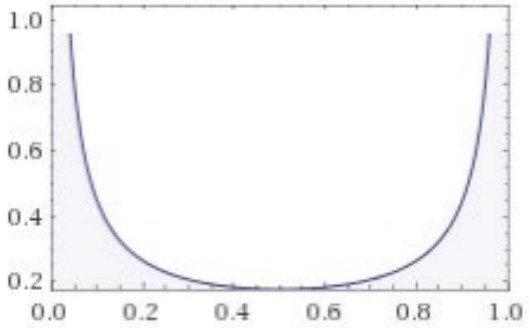
이미지 분류모델 학습 -2

- 다양한 Augmentation 방법
  - Random erasing, Grid mask
  - Crop, Rotation, Flip, ...
  - MixUP, CutMix, Mosaic, Blur

Randomly blend the pixels
of pairs of training images,
e.g. 40% cat, 60% dog

CNN

Target label:
cat: 0.4
dog: 0.6



Jetbot Software-Collision Avoidance 1

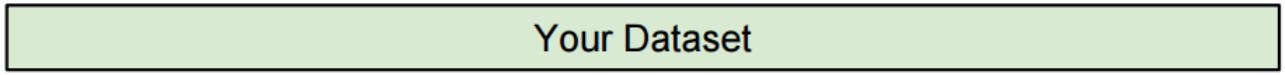
Split dataset into train and test sets

Next, we split the dataset into *training* and *test* sets. The test set will be used to verify the accuracy of the model we train.

```
train_dataset, test_dataset = torch.utils.data.random_split(dataset, [len(dataset) - 5
0, 50])
```

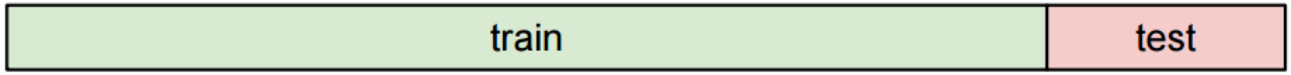
Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data



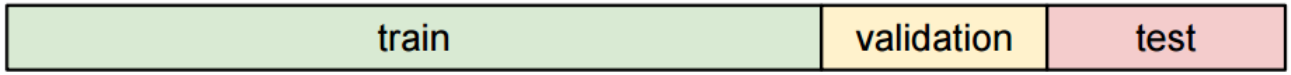
Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data



Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Better!

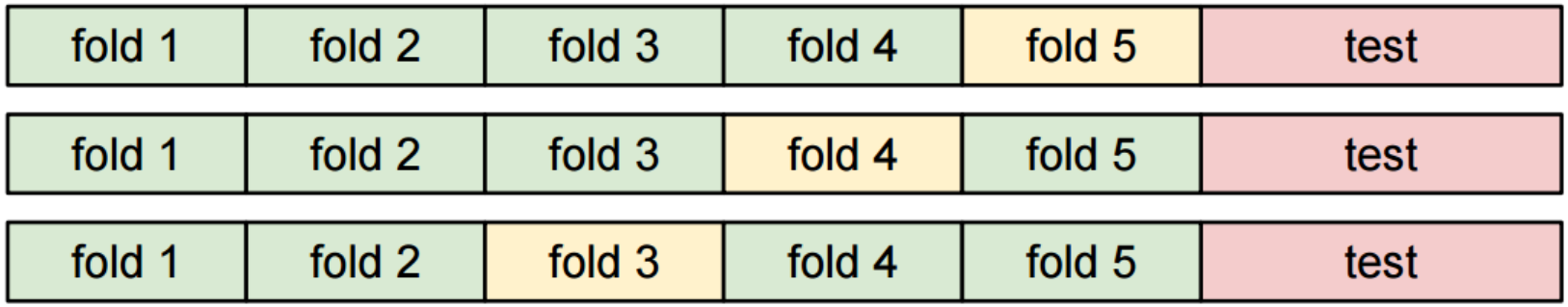


Jetbot Software-Collision Avoidance 1

Split dataset into train and test sets

Next, we split the dataset into *training* and *test* sets. The test set will be used to verify the accuracy of the model we train.

```
train_dataset, test_dataset = torch.utils.data.random_split(dataset, [len(dataset) - 5
0, 50])
```



Jetbot Software-Collision Avoidance 1

Create data loaders to load data in batches

We'll create two `DataLoader` instances, which provide utilities for shuffling data, producing *batches* of images, and loading the samples in parallel with multiple workers.

```
train_loader = torch.utils.data.DataLoader(
    train_dataset,
    batch_size=8,
    shuffle=True,
    num_workers=0,
)

test_loader = torch.utils.data.DataLoader(
    test_dataset,
    batch_size=8,
    shuffle=True,
    num_workers=0,
)
```

Jetbot Software-Collision Avoidance 1

Define the neural network ¶

Now, we define the neural network we'll be training. The *torchvision* package provides a collection of pre-trained models that we can use.

In a process called *transfer learning*, we can repurpose a pre-trained model (trained on millions of images) for a new task that has possibly much less data available.

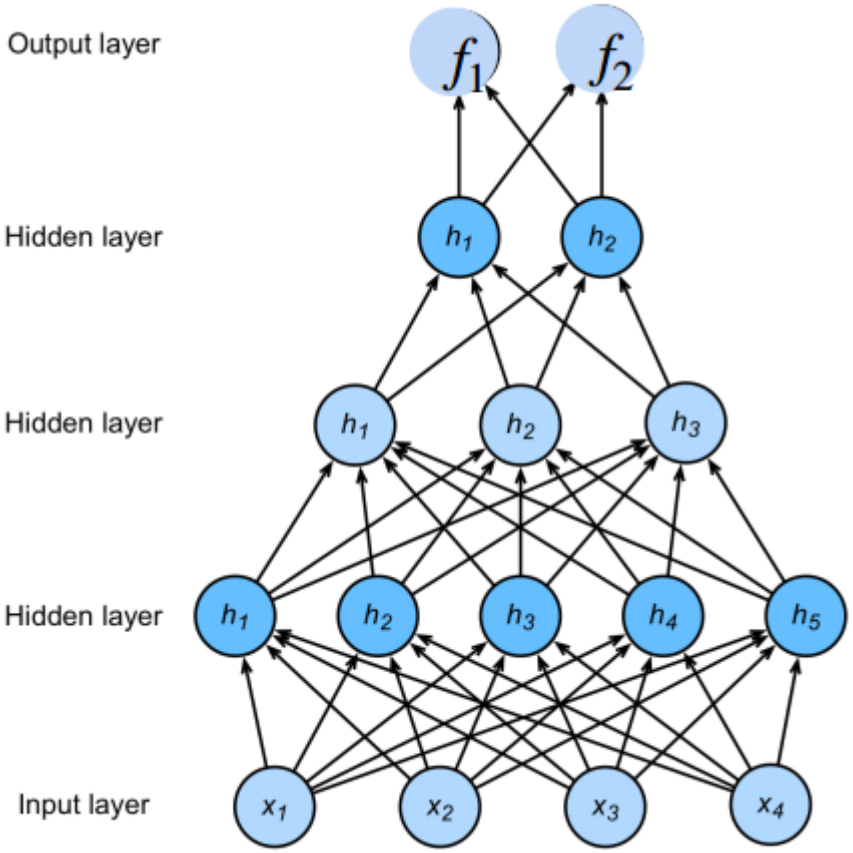
Important features that were learned in the original training of the pre-trained model are re-usable for the new task. We'll use the `resnet18` model.

```
model = models.resnet18(pretrained=True)
```

Convolutional Neural Network

CNN

- 기존 Deep Neural Network (DNN)

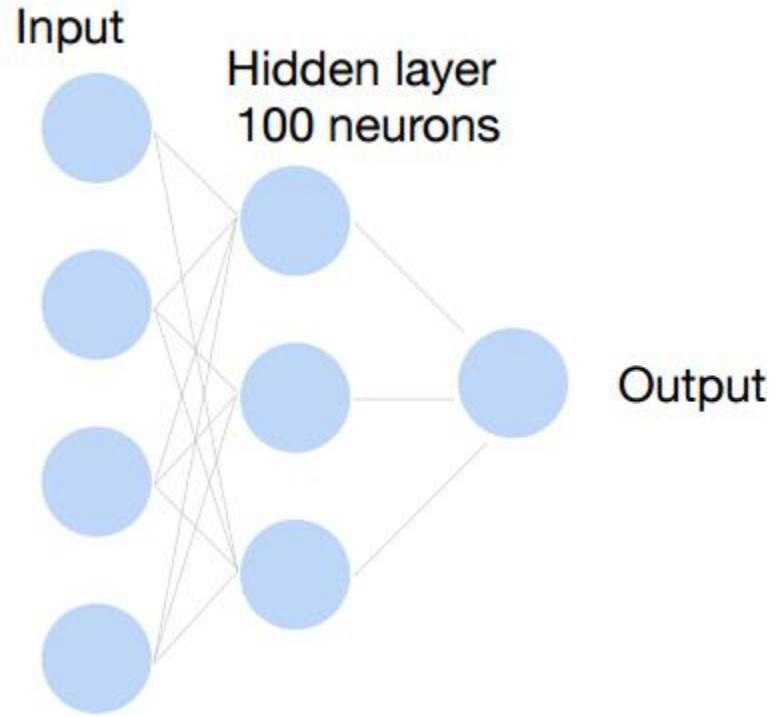


$$\mathbf{h}_1 = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$
$$\mathbf{h}_2 = \sigma(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2)$$
$$\mathbf{h}_3 = \sigma(\mathbf{W}_3\mathbf{h}_2 + \mathbf{b}_3)$$
$$\mathbf{f} = \mathbf{W}_4\mathbf{h}_3 + \mathbf{b}_4$$
$$\mathbf{y} = \text{softmax}(\mathbf{f})$$

NNs are composition of nonlinear functions

CNN

- 기존 Deep Neural Network (DNN)



CNN

- 기존 Deep Neural Network (DNN)

CNN

- **기존 Deep Neural Network (DNN)**

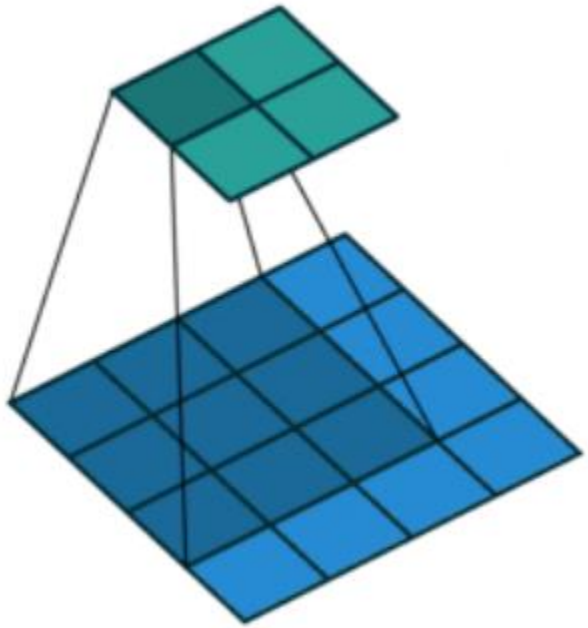
Why Convolution?

- Translation Invariance
- Locality

CNN

- **2D Convolution**

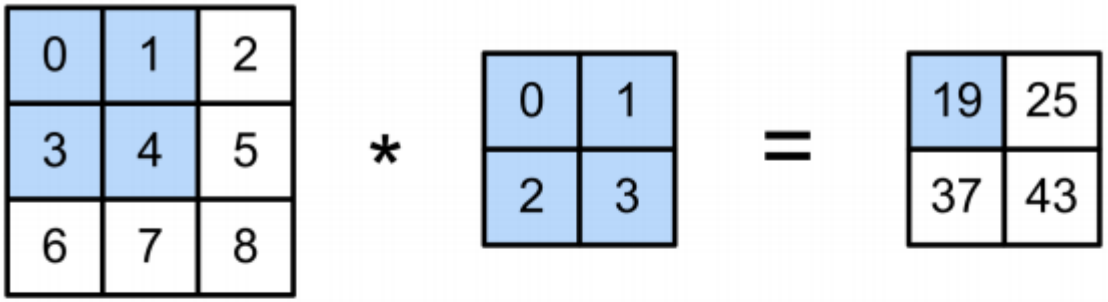
Input				Kernel			Output	
0	1	2	*	0	1	=	19	25
3	4	5		2	3		37	43
6	7	8						



$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19,$$
$$1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 = 25,$$
$$3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 = 37,$$
$$4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 = 43.$$

CNN

- **2D Convolution Layer**



- $\mathbf{X} : n_h \times n_w$ input matrix
- $\mathbf{W} : k_h \times k_w$ kernel matrix
- b: scalar bias
- $\mathbf{Y} : (n_h - k_h + 1) \times (n_w - k_w + 1)$ output matrix

$$\mathbf{Y} = \mathbf{X} \star \mathbf{W} + b$$

- **W** and *b* are learnable parameters

CNN

- **Kernel Examples**

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Edge Detection

Input

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

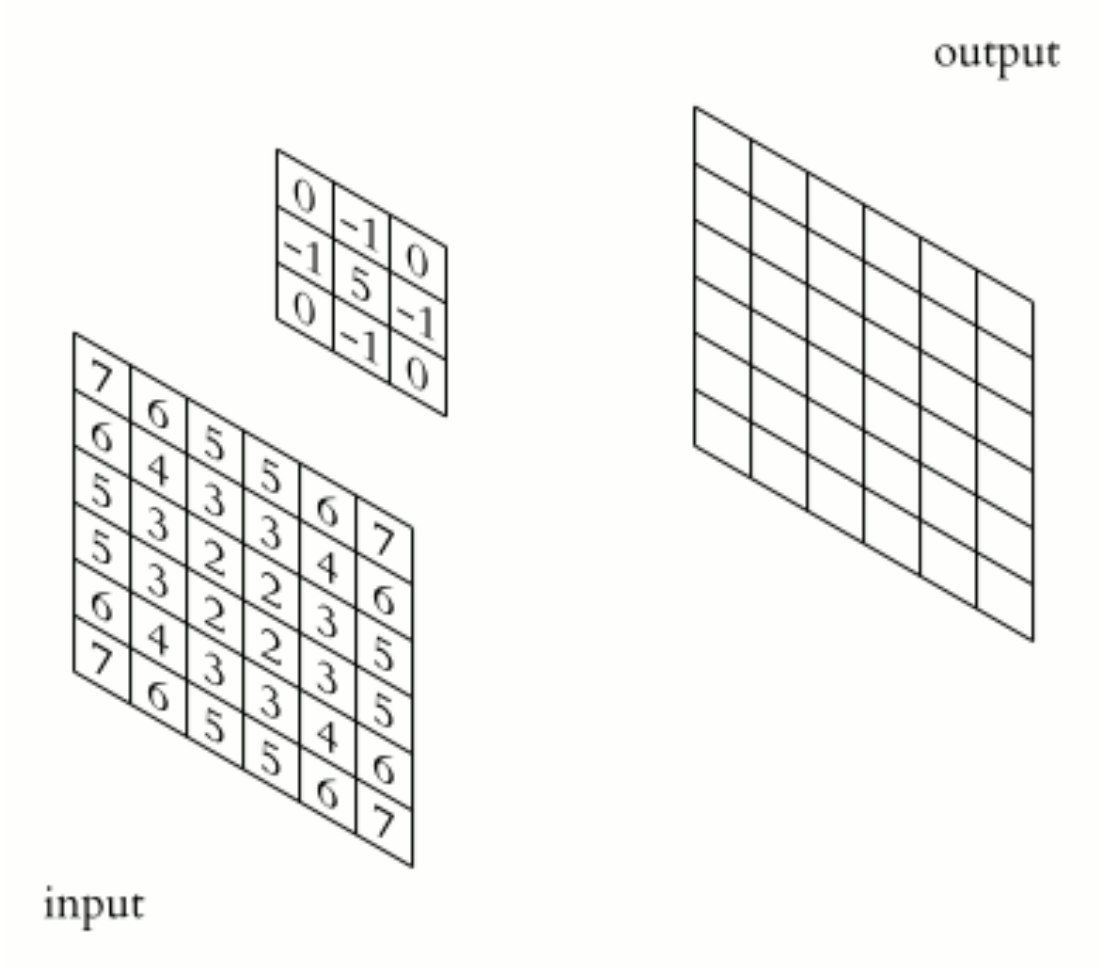
Sharpen

$$\frac{1}{16}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Gaussian Blur

CNN

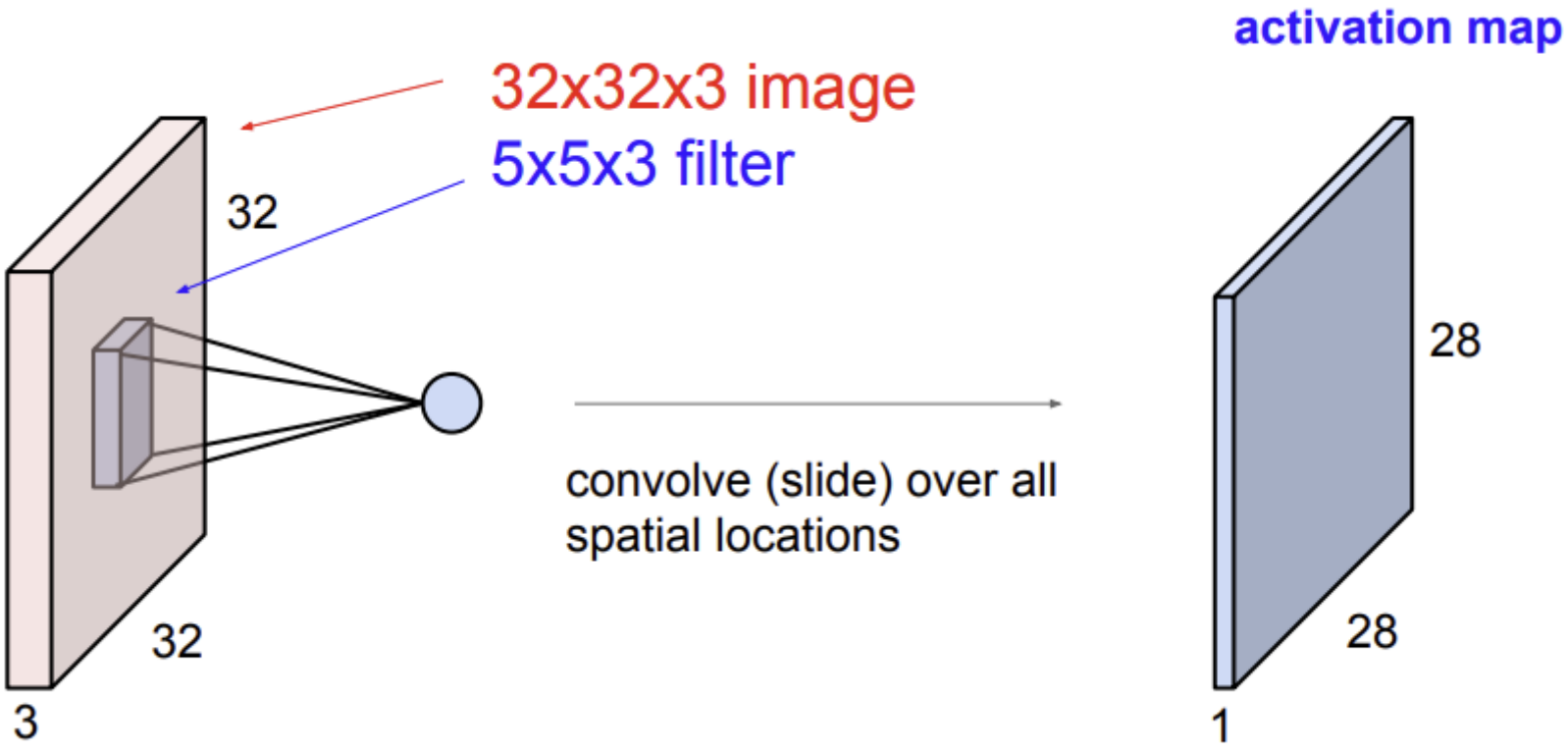
- **Kernel Examples**



https://miro.medium.com/max/780/1*k_Kw7RPCfMaIY7USoAedqA.gif

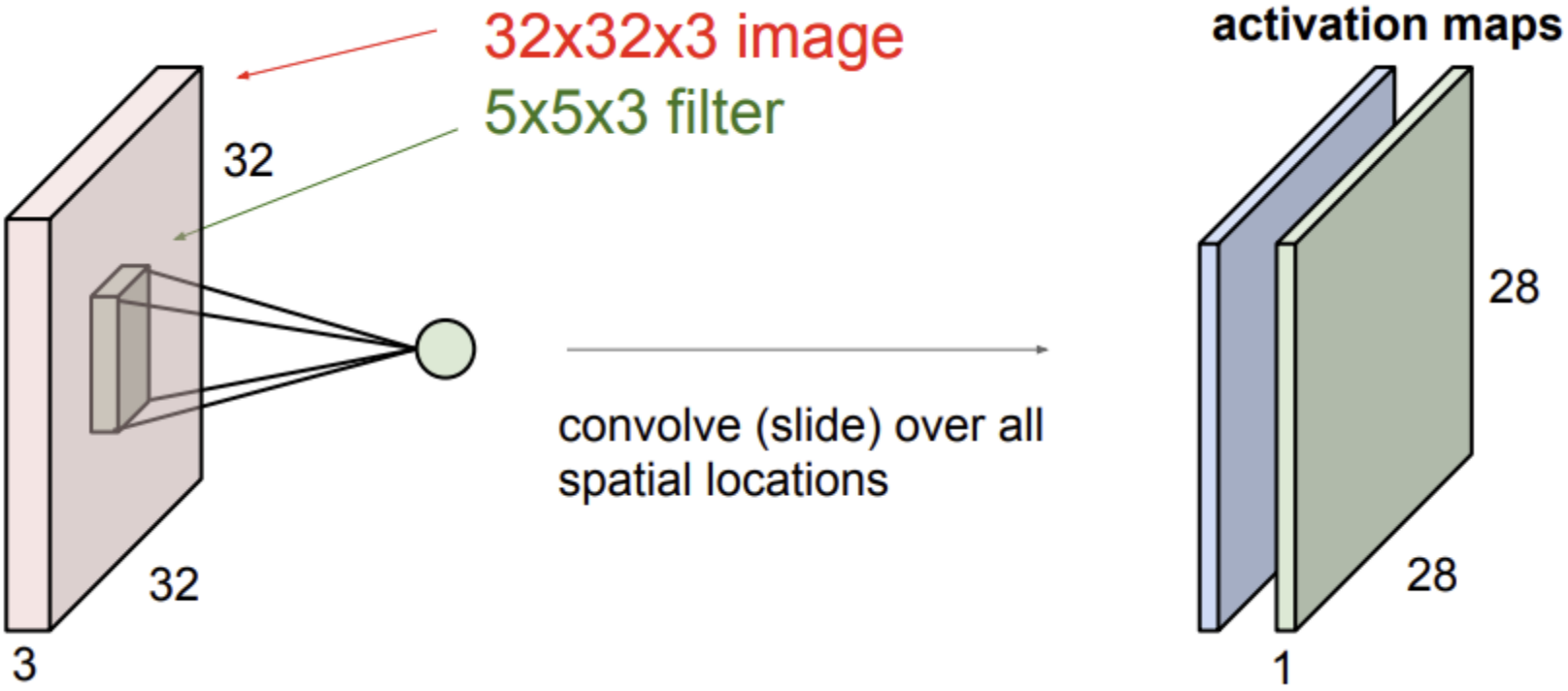
CNN

- **Convolutional layer Examples**



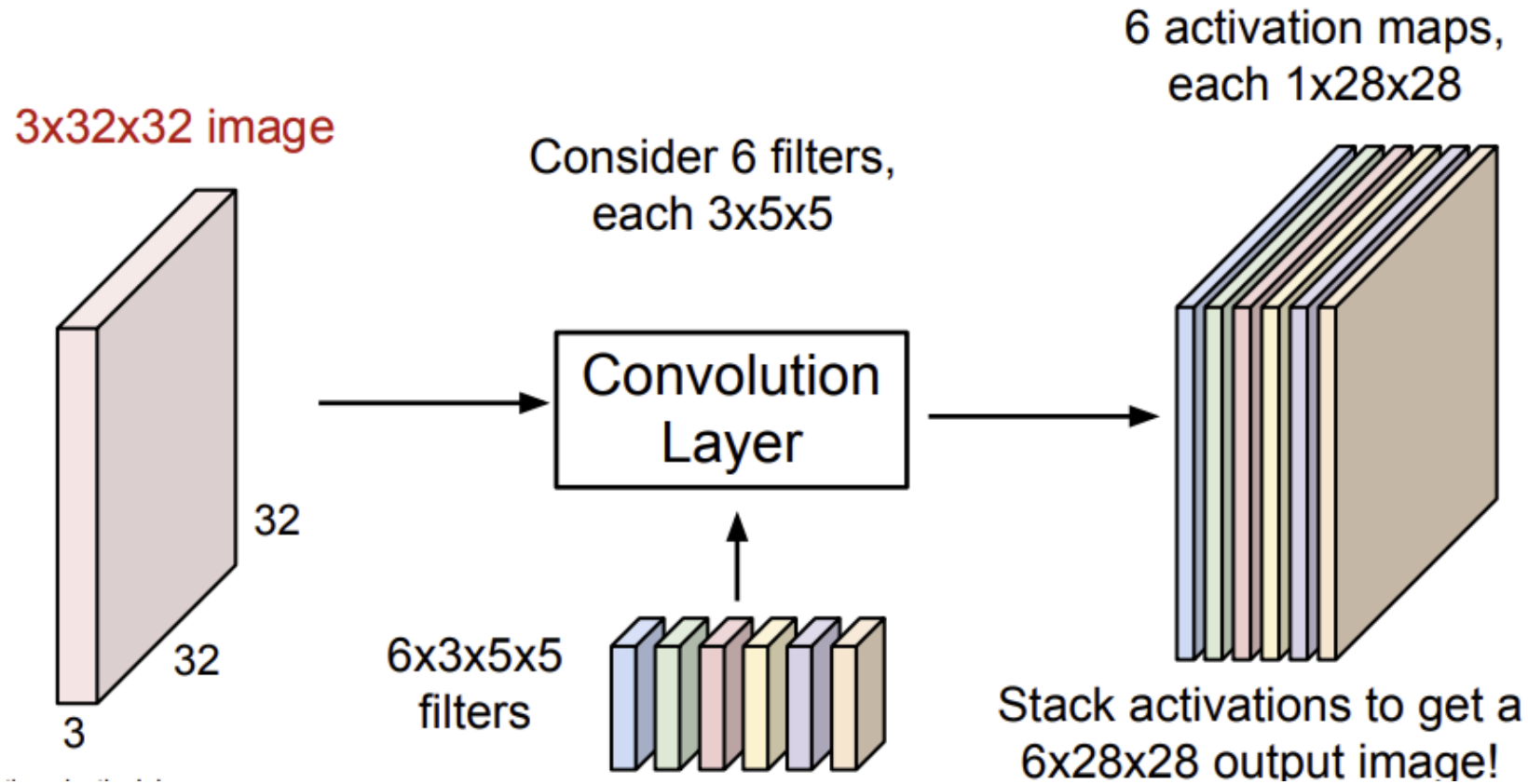
CNN

- **Convolutional layer Examples**



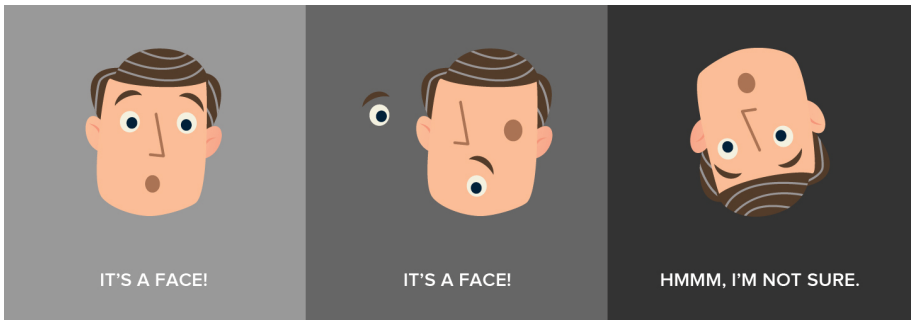
CNN

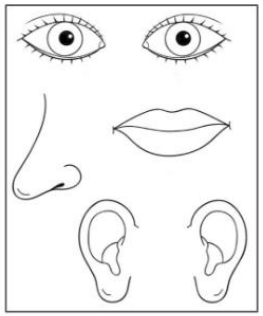
- **Convolutional layer Examples**



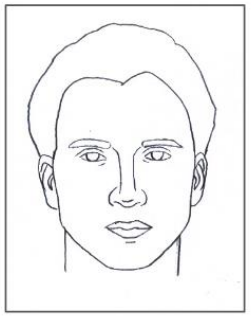
CNN

- Higher layer은 단순히 lower layer들의 가중합(weighted sum)
- Simple feature와 complex feature간의 위치 관계를 전혀 고려하지 않음
- CapsuleNet: 공간적인 계층 고려

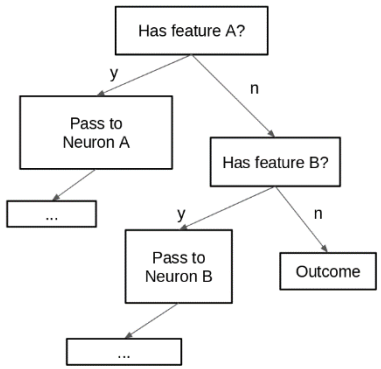




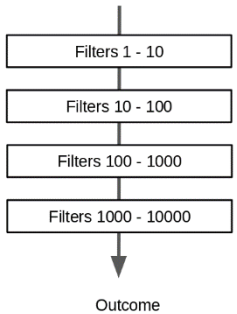
Face



Face



Humans

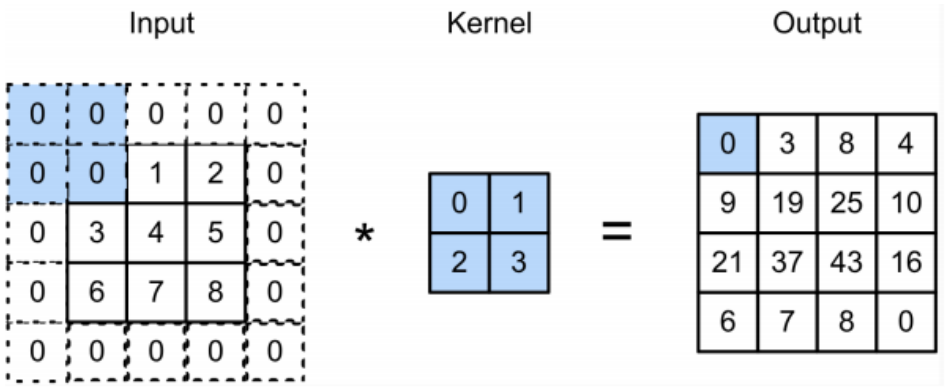


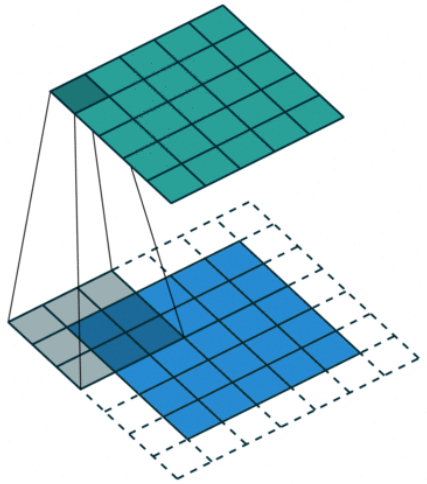
CNN

https://www.i2tutorials.com/what-is-the-difference-between-convolutional-neural-network-and-capsule-neural-network/

CNN

- **Padding**
  - **Adds rows/columns around input**

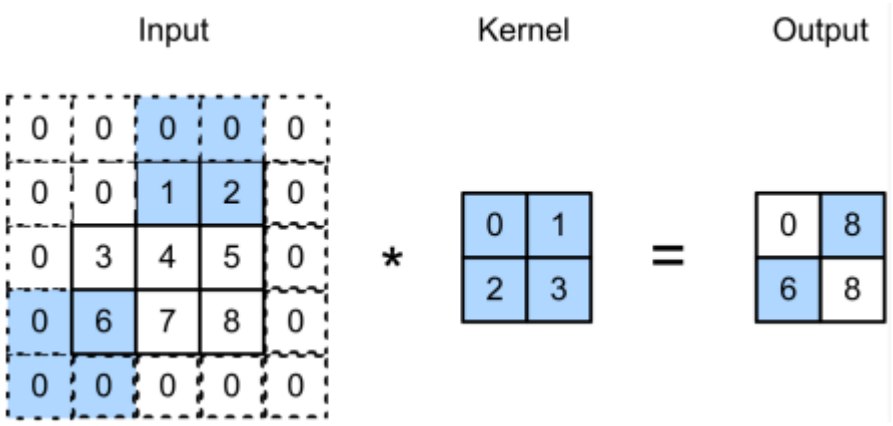




$$0 \times 0 + 0 \times 1 + 0 \times 2 + 0 \times 3 = 0$$

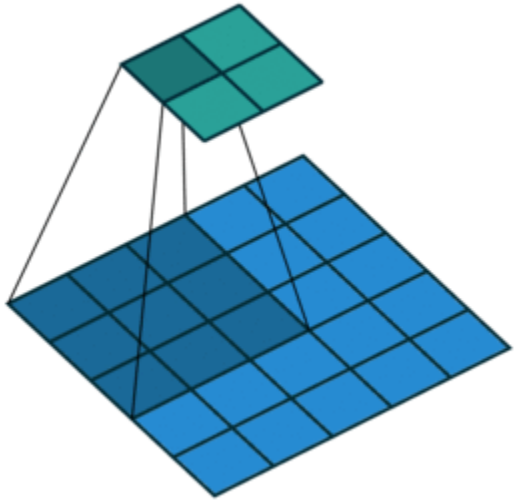
CNN

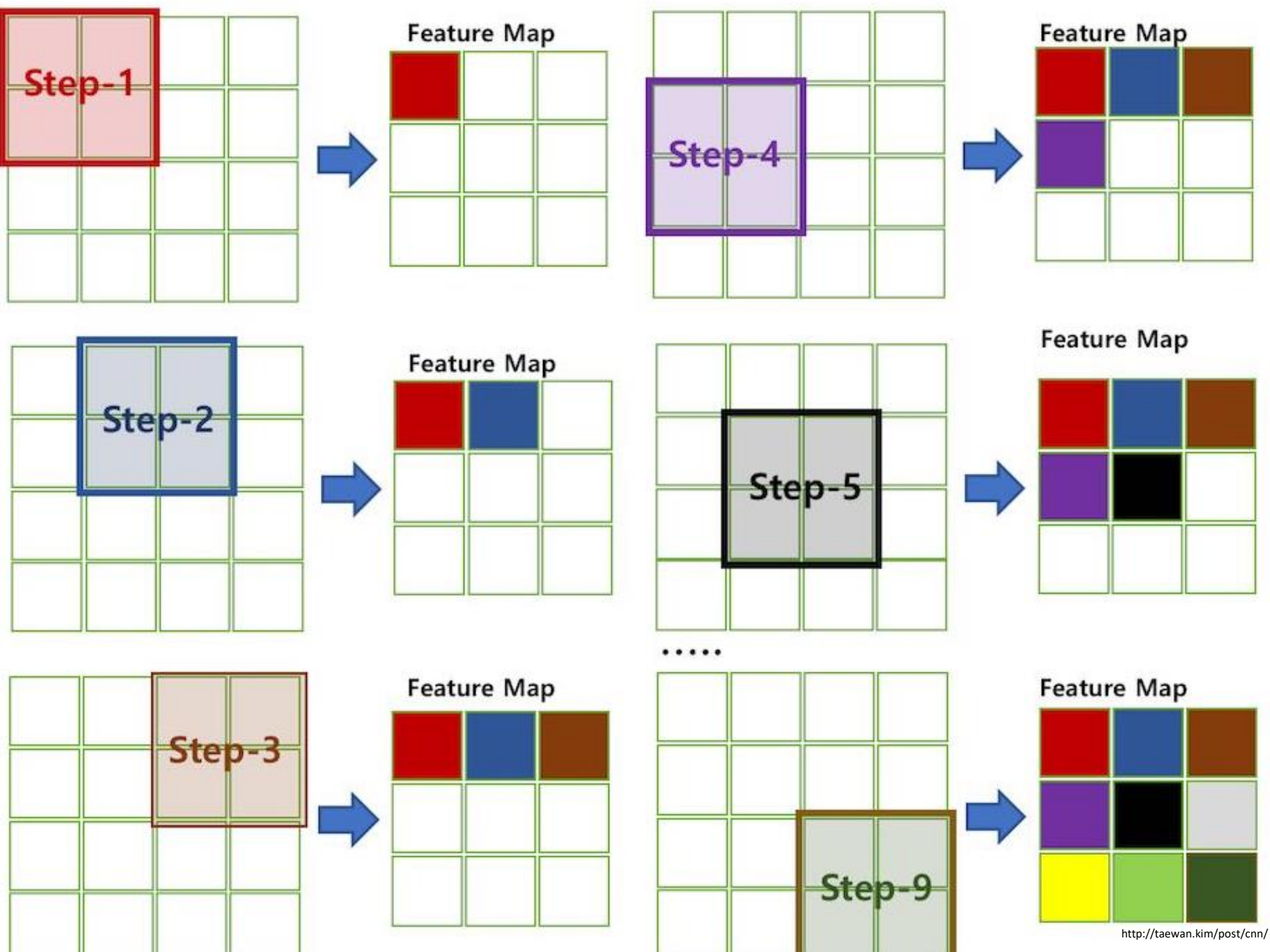
- **Padding**
  - **Adds rows/columns around input**
- **Stride**
  - **the #rows/#columns per slide**
  - **3 for height, 2 for width**



$0 \times 0 + 0 \times 1 + 1 \times 2 + 2 \times 3 = 8$

$0 \times 0 + 6 \times 1 + 0 \times 2 + 0 \times 3 = 6$

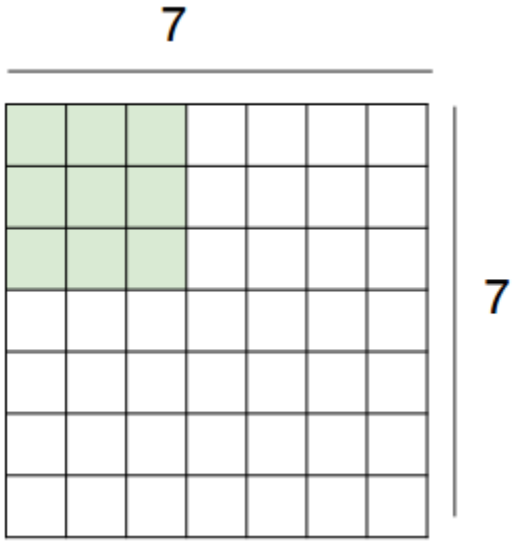




http://taewan.kim/post/cnn/

Quiz!!!

- **Input Image: 7x7**
- **No padding**
- **Kernel size: 3x3**
- **Stride: 2**
- **What is the dimension of the output?**
  - **A. 3X3**
  - **B. 7X7**
  - **C. 5X5**
  - **D. 2X2**

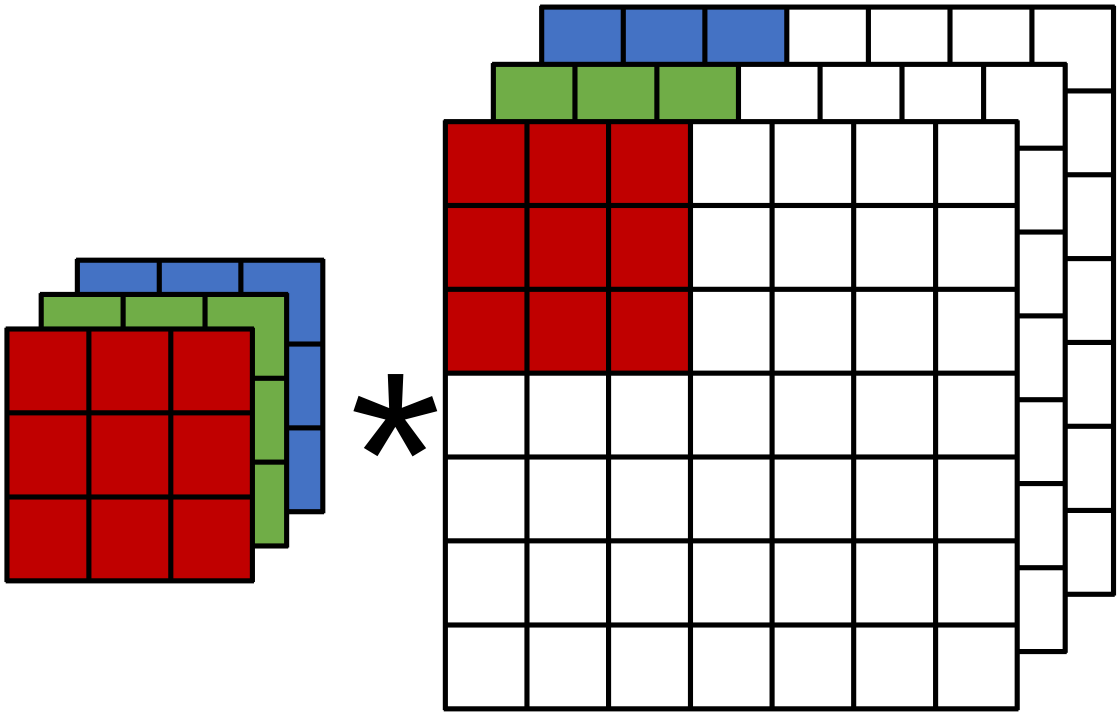


CNN

- **Multiple Input Channels**
  - **Color image may have three RGB channels**
  - **Converting to grayscale loses information**

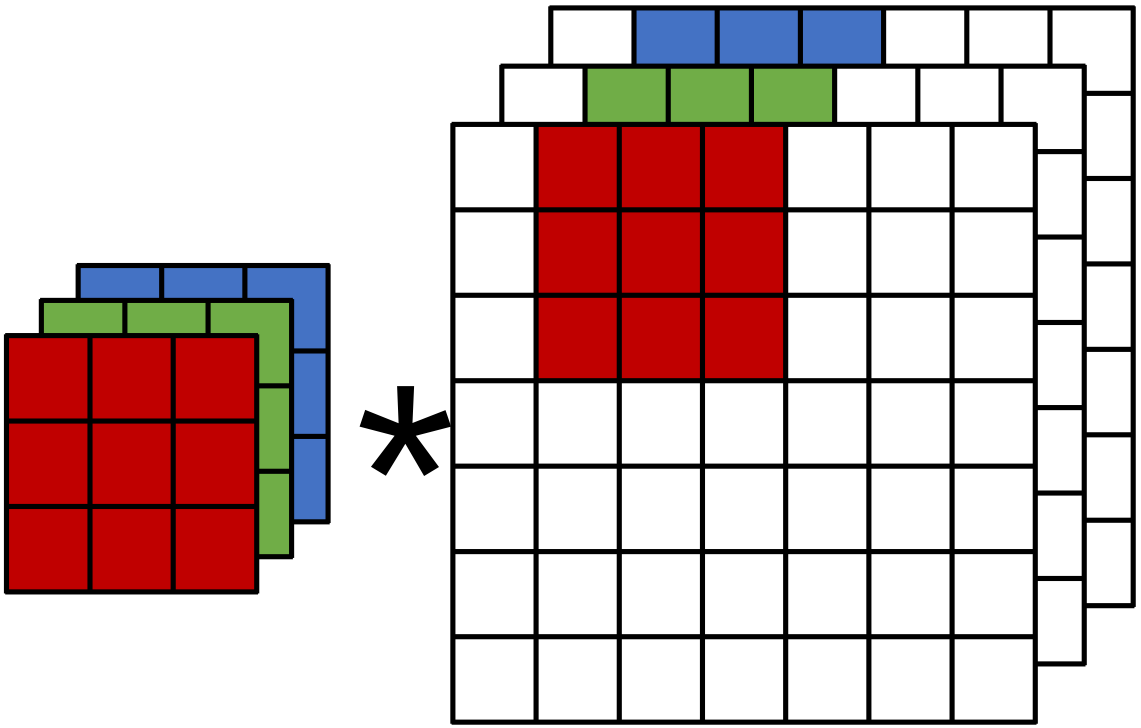
CNN

- **Multiple Input Channels**
  - **Color image may have three RGB channels**
  - **Converting to grayscale loses information**



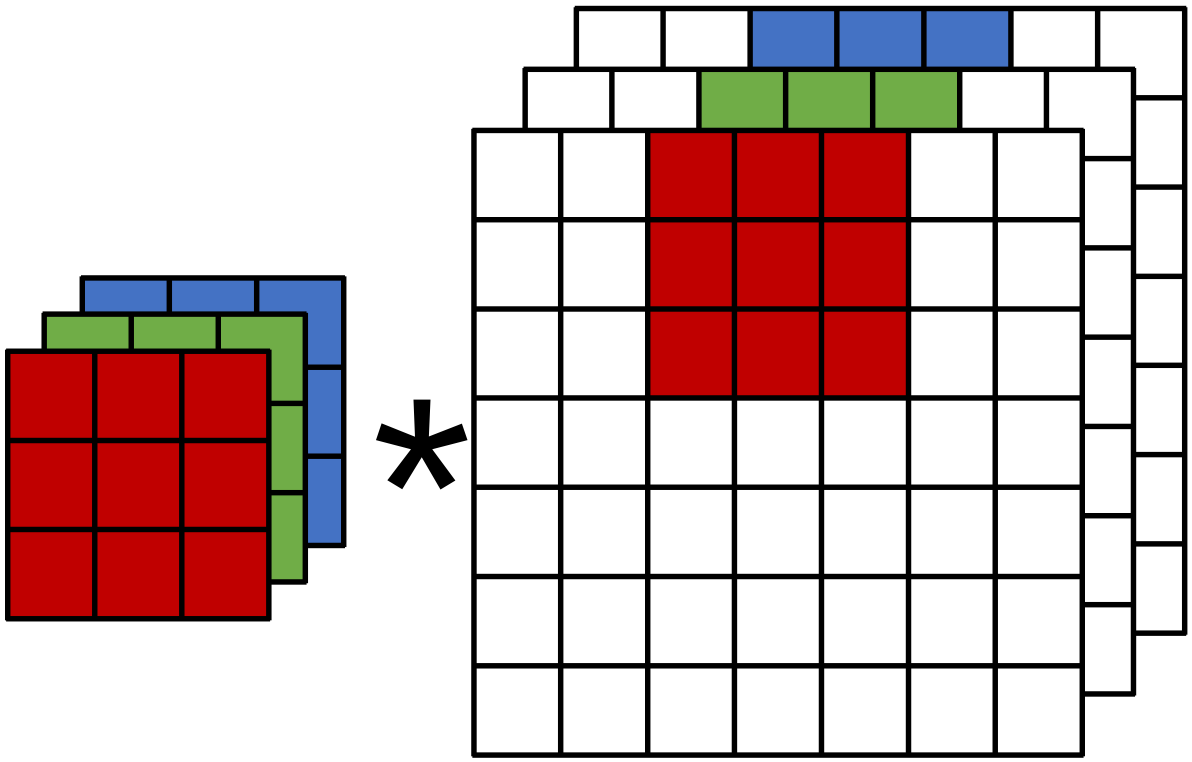
CNN

- **Multiple Input Channels**
  - **Color image may have three RGB channels**
  - **Converting to grayscale loses information**



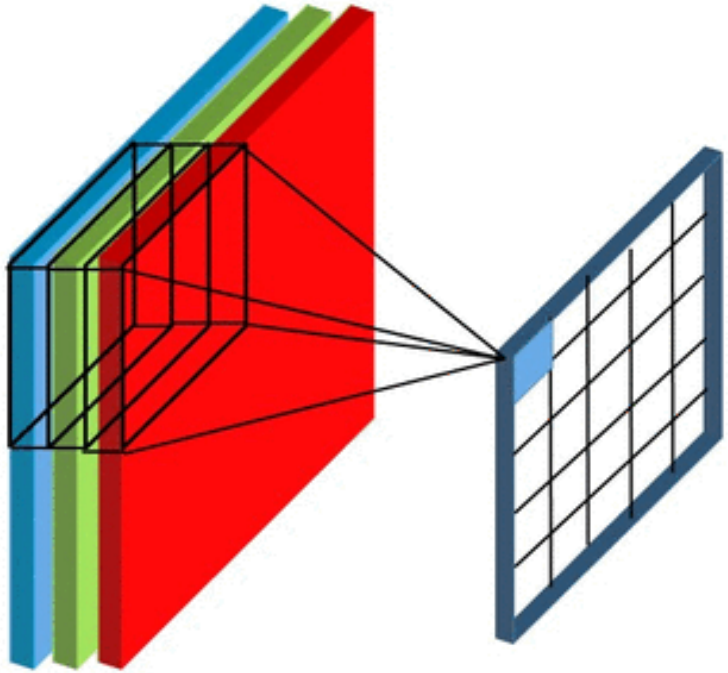
CNN

- **Multiple Input Channels**
  - **Color image may have three RGB channels**
  - **Converting to grayscale loses information**



CNN

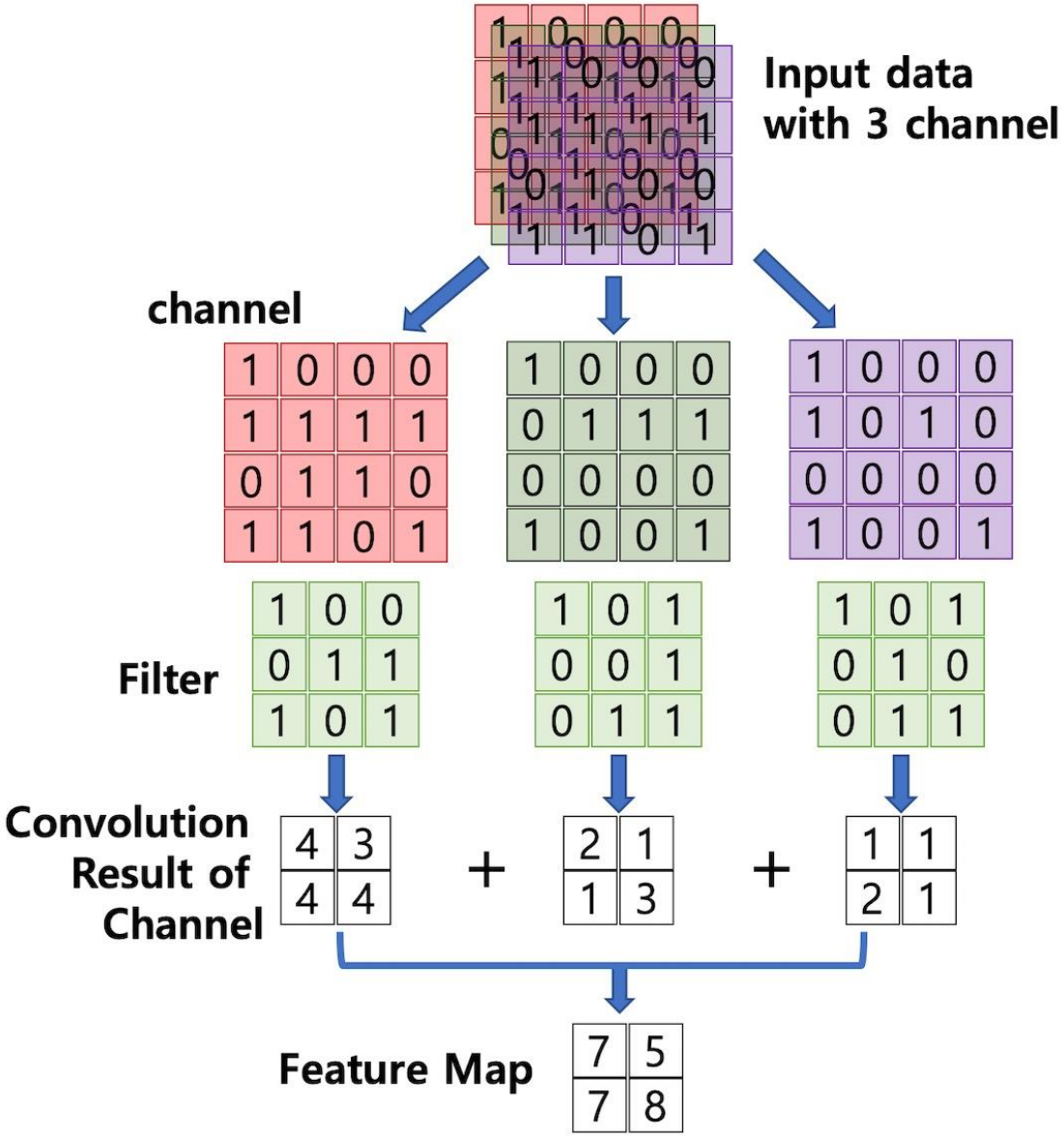
- **Multiple Input Channels**
  - **Color image may have three RGB channels**
  - **Converting to grayscale loses information**



https://miro.medium.com/max/738/1*tmKo_Dvlf3bHCMFx_PKZYA.gif

CNN

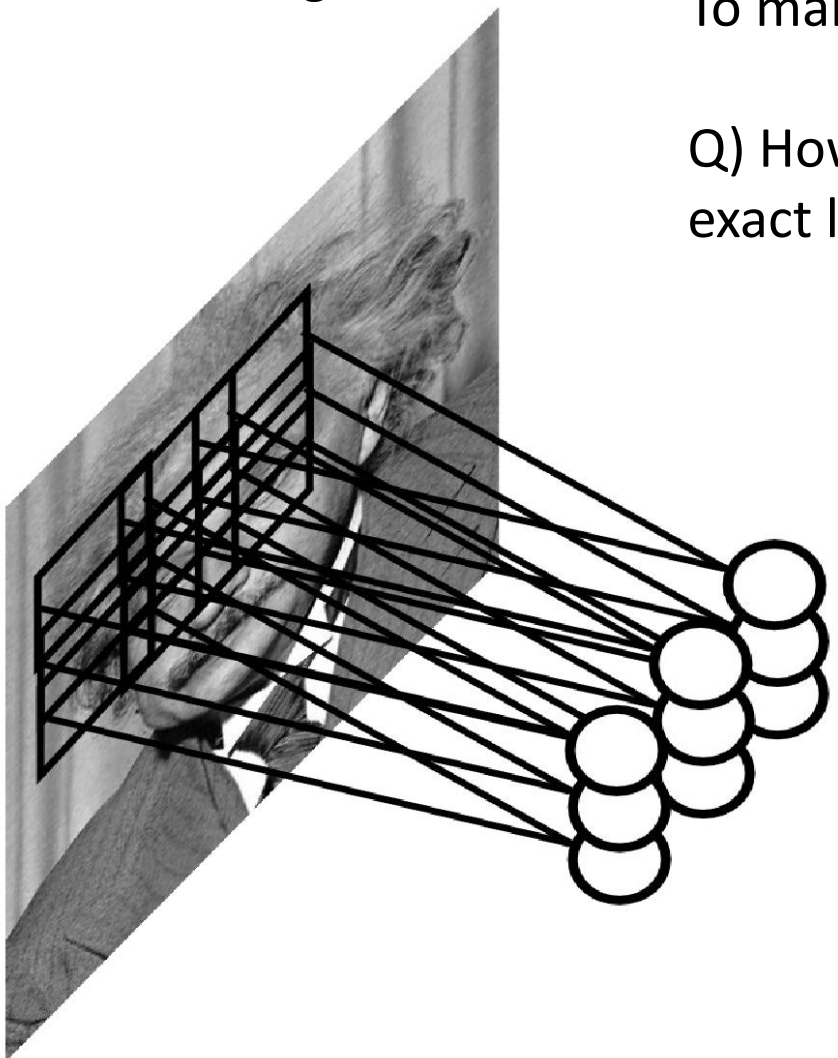
- Multiple
  - Colo
  - Conv



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

CNN

- **Pooling**

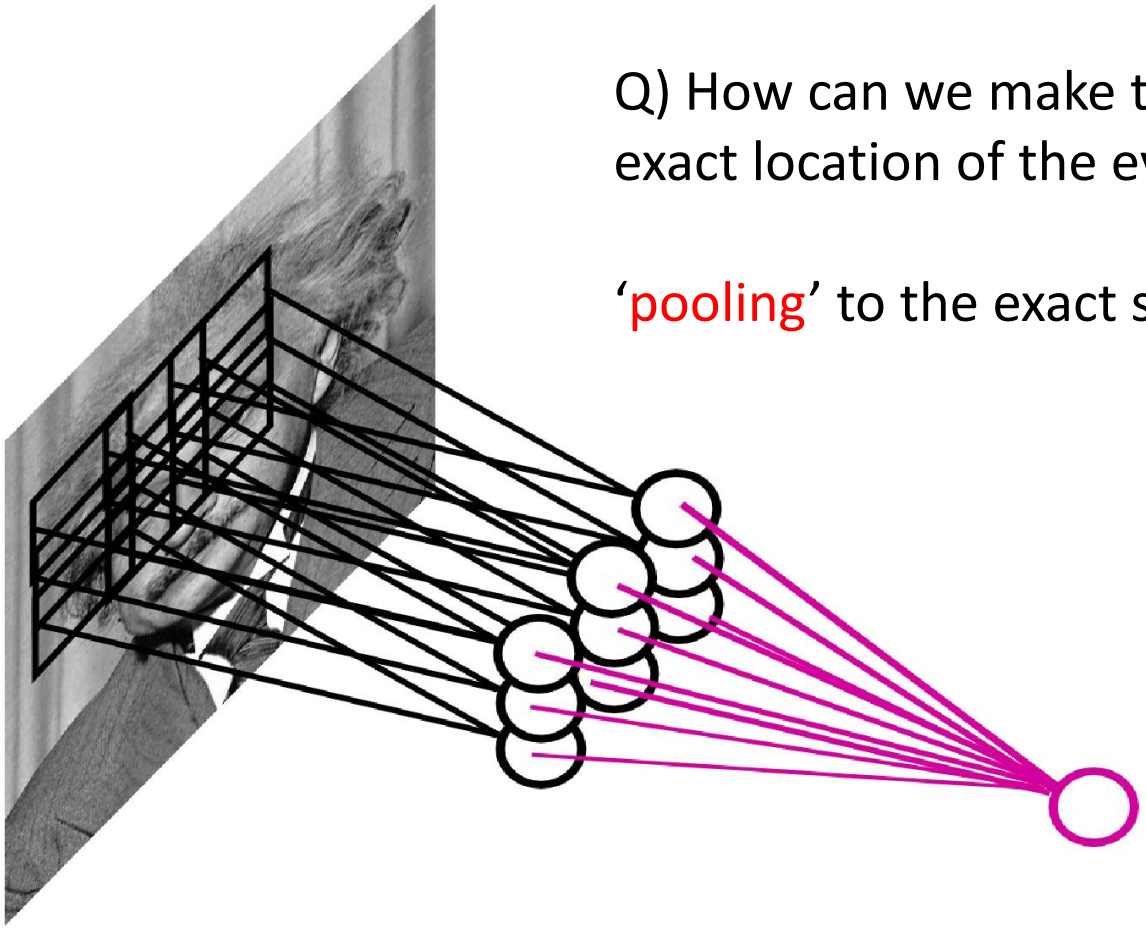


To make a ‘eye’ detector filter

Q) How can we make the detection robust to the exact location of the eyes??

CNN

- **Pooling**

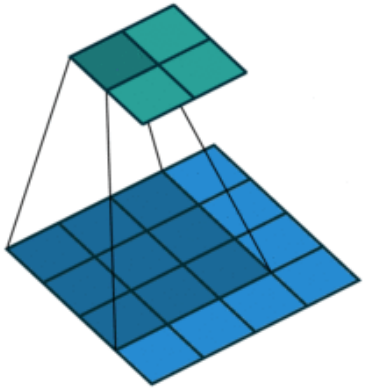


To make a ‘eye’ detector filter

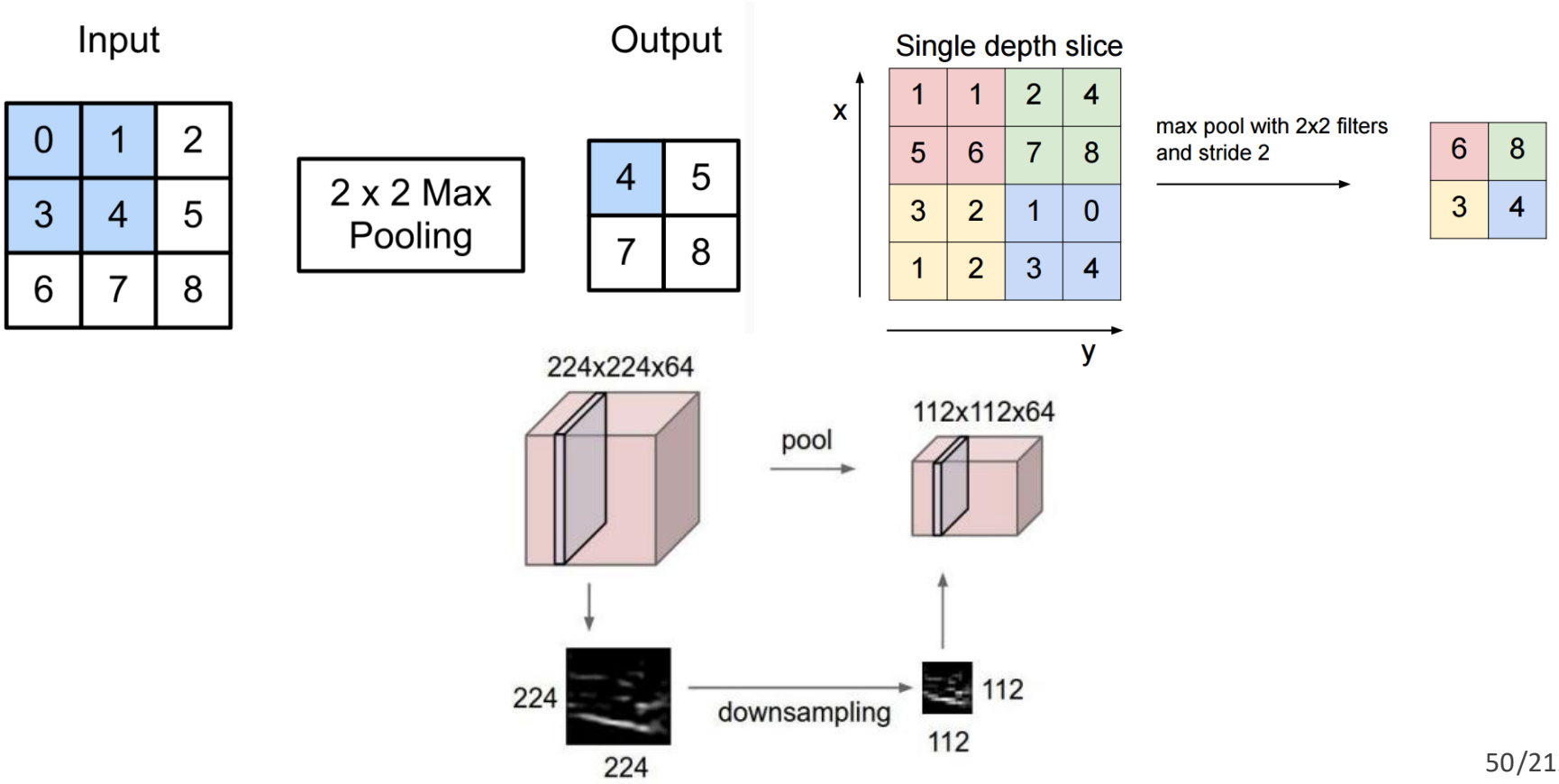
Q) How can we make the detection robust to the exact location of the eyes??

‘pooling’ to the exact spatial location of features

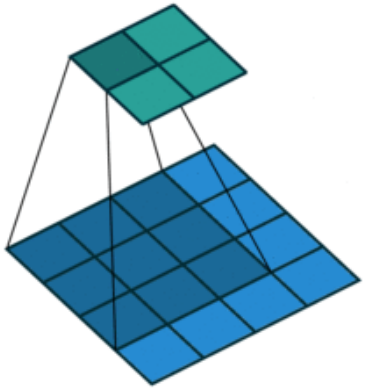
CNN



- **Pooling**
  - **Max/Average Pooling**



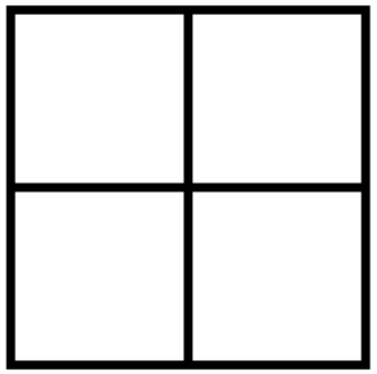
CNN



- Pooling
  - Max/Average Pooling

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

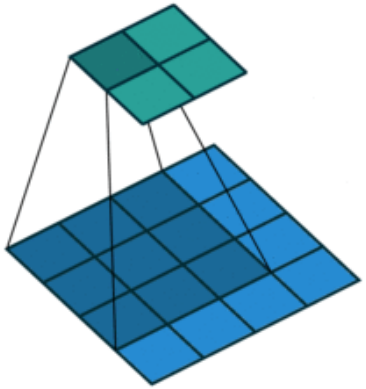
Feature map



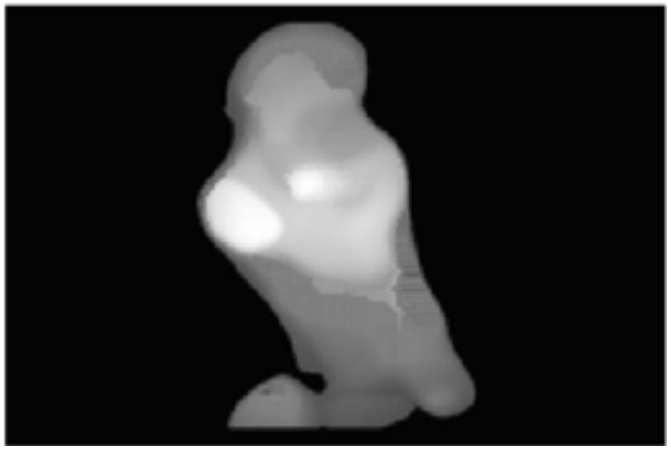
Pooled
Feature map

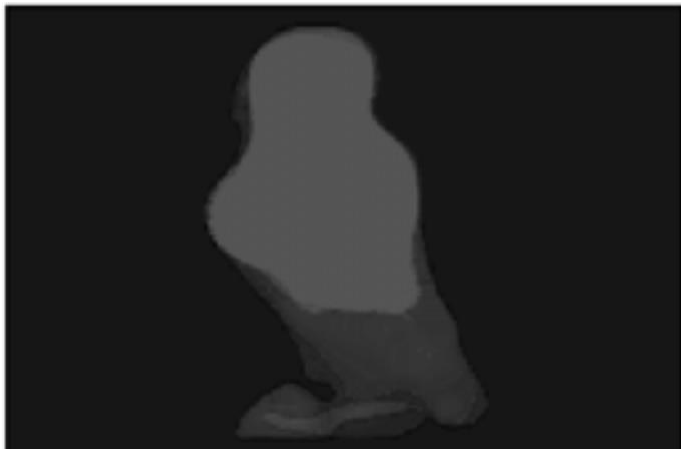
https://miro.medium.com/max/1400/1*vOxthD0FpBR6fJcpPxq6Hg.gif

CNN

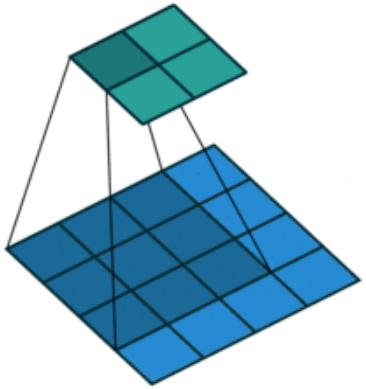


- **Pooling**
  - **Quiz!!**
  - **Which one is the result of Average Pooling??**





CNN



- **Pooling**
  - **Quiz!!**
  - **Which one is the result of Average Pooling??**
  - **What is the max/average pooling output??**
    - **Input image: 4x4**
    - **Max/Average pooling: 2x2**
    - **stride: 2**

Input Image: $\begin{bmatrix} 12 & 20 & 30 & 0 \\ 20 & 12 & 2 & 0 \\ 0 & 70 & 5 & 2 \\ 8 & 2 & 90 & 3 \end{bmatrix}$

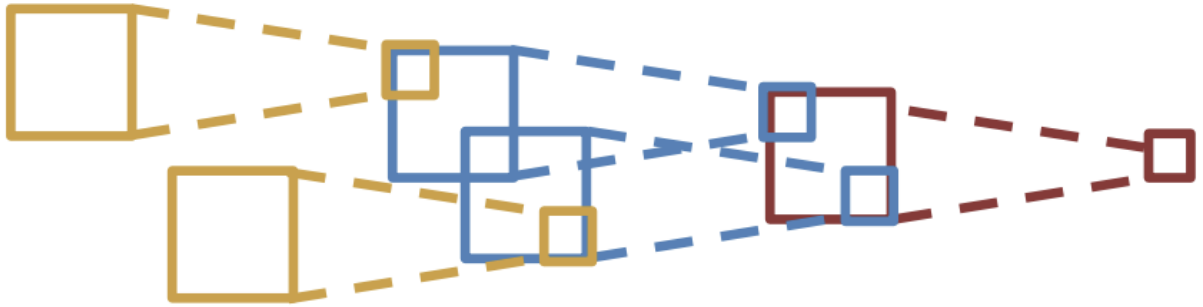
A. $\begin{bmatrix} 20 & 30 \\ 70 & 90 \end{bmatrix}$

B. $\begin{bmatrix} 16 & 8 \\ 20 & 25 \end{bmatrix}$

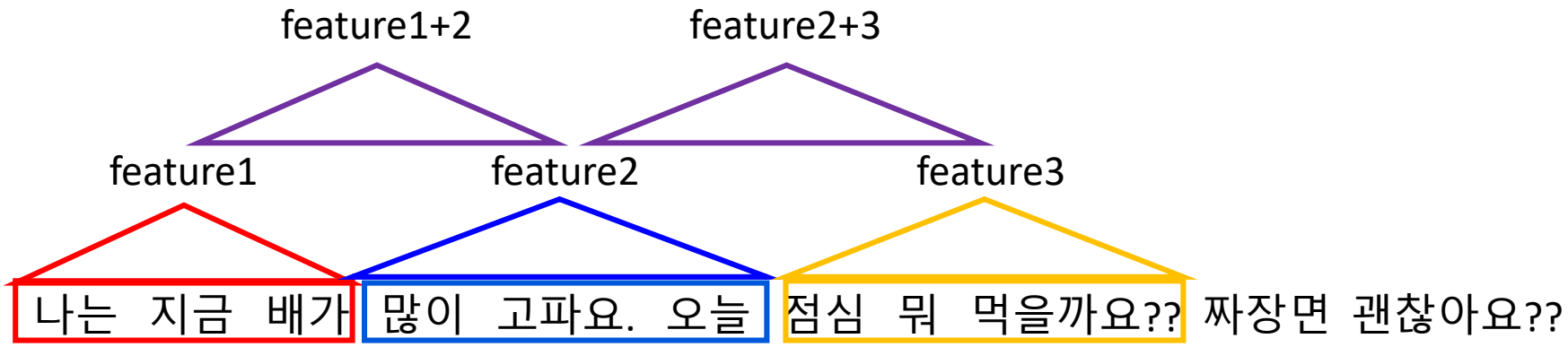
C. $\begin{bmatrix} 20 & 30 \\ 20 & 25 \end{bmatrix}$

D. $\begin{bmatrix} 12 & 2 \\ 70 & 5 \end{bmatrix}$

CNN

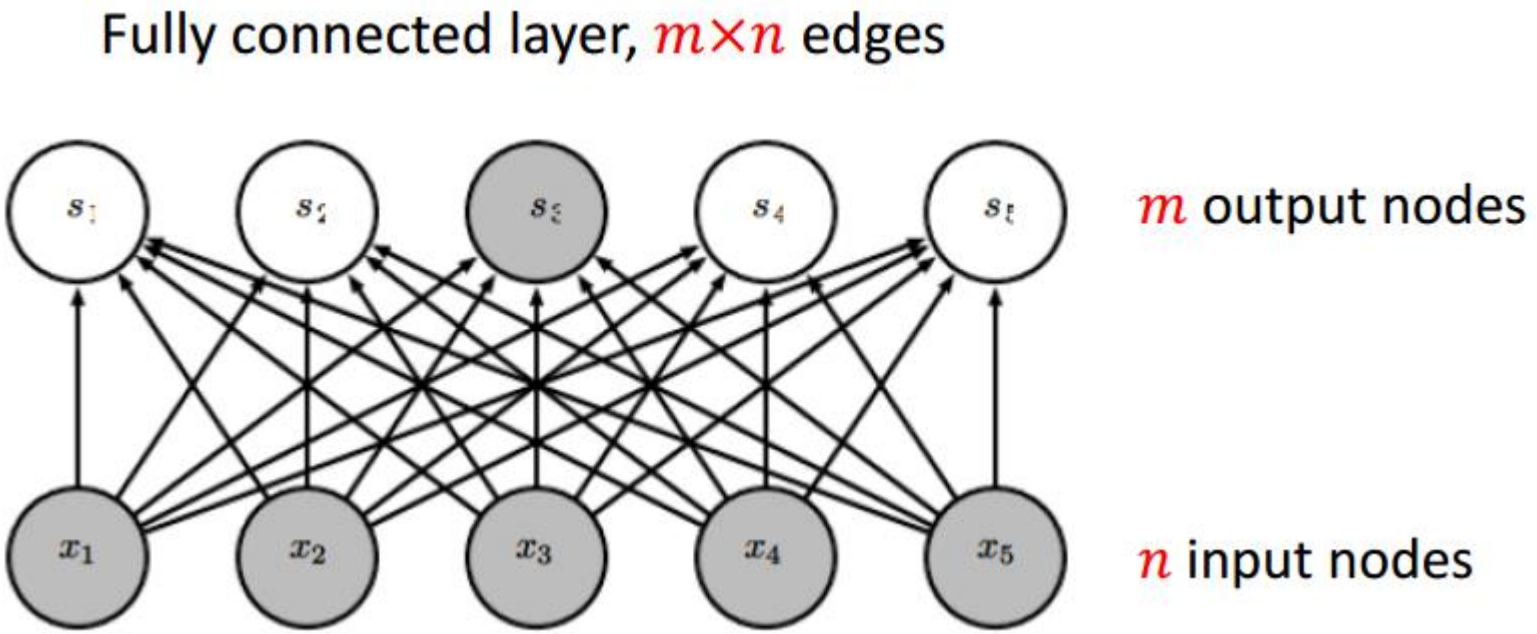


- **Receptive Fields**
  - k size convolution, output depends on a KxK receptive field in the input
  - Problem: For large images we need many layers for each output to "see" the whole image image
  - Solution: Downsample inside the network



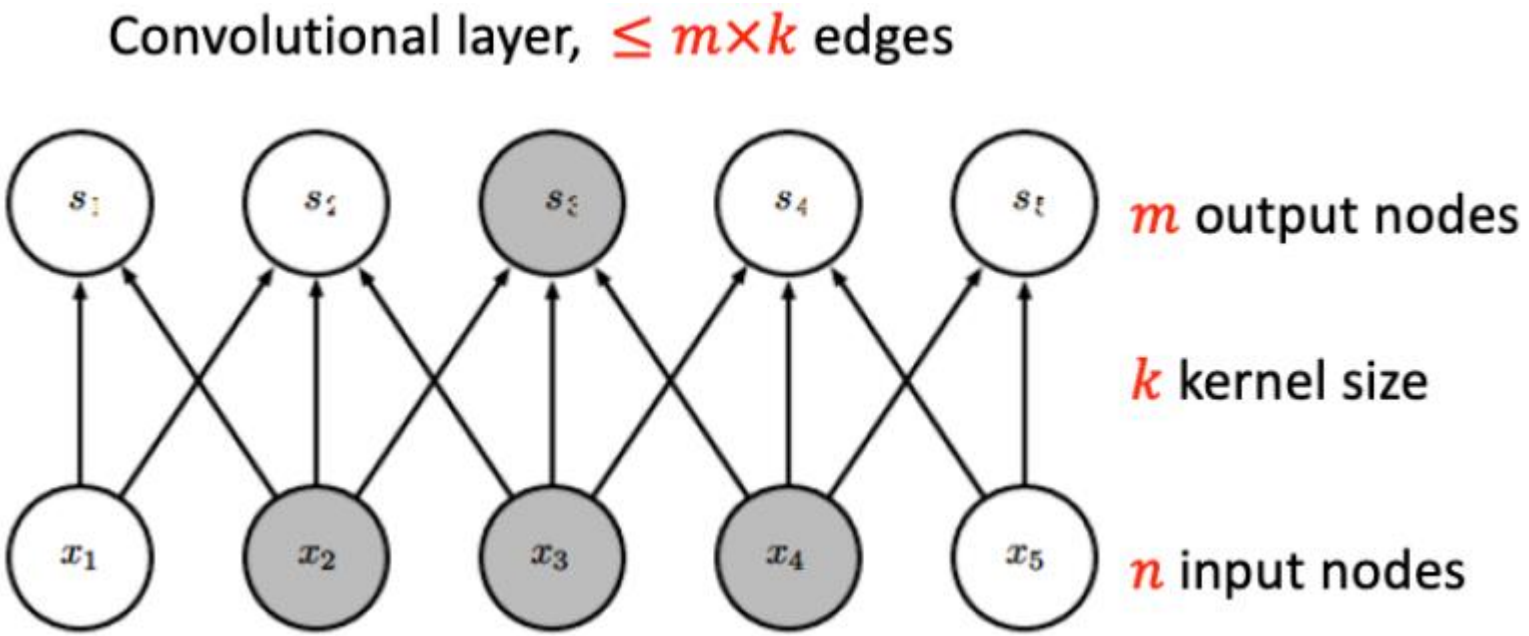
CNN

- **Advantage: sparse interaction**



CNN

- **Advantage: sparse interaction**



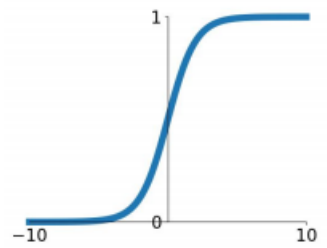
CNN

- **Pooling Example**

Activation Fn

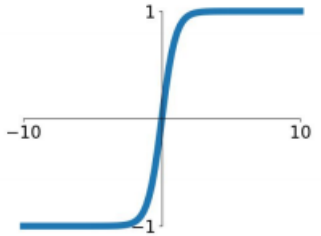
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



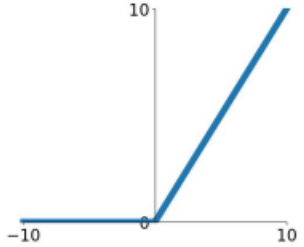
tanh

$$\tanh(x)$$



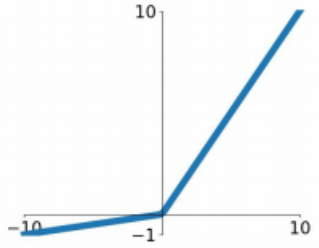
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

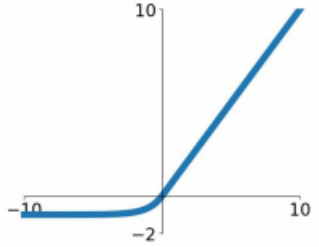


Maxout

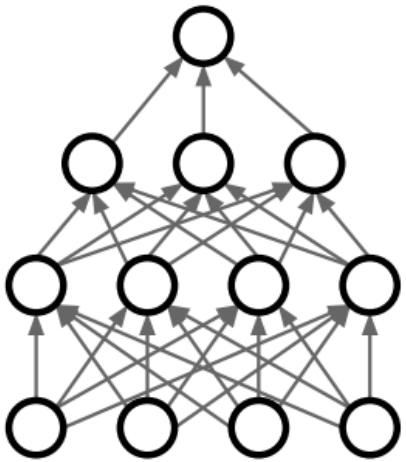
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

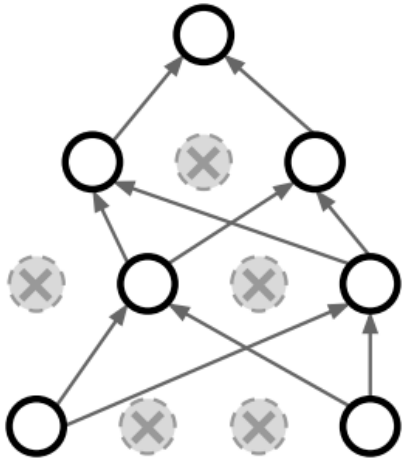
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



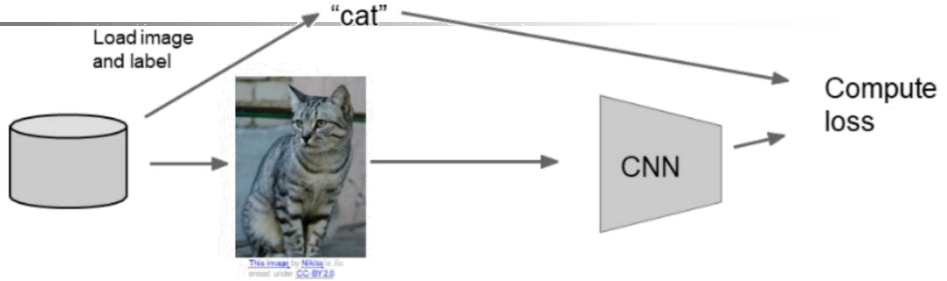
DropOut

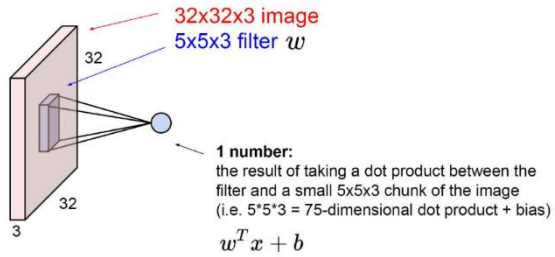


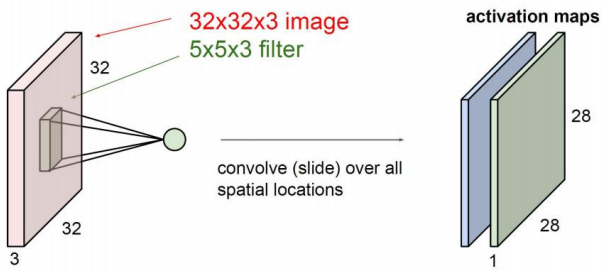


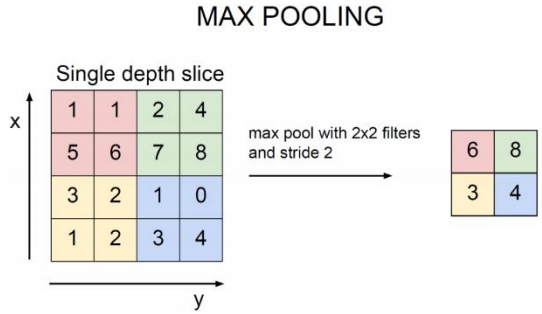
CNN

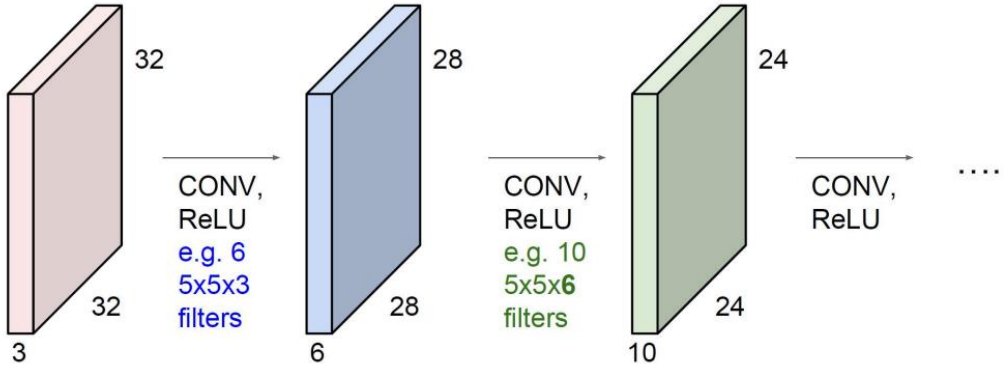
- 이미지 분류모델 – CNN

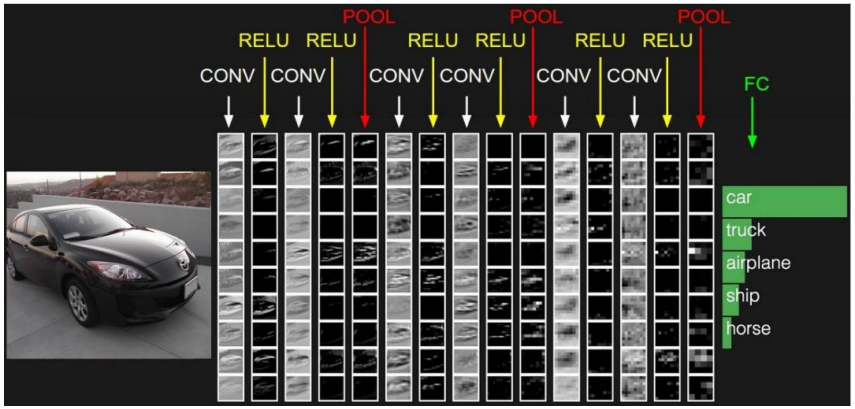






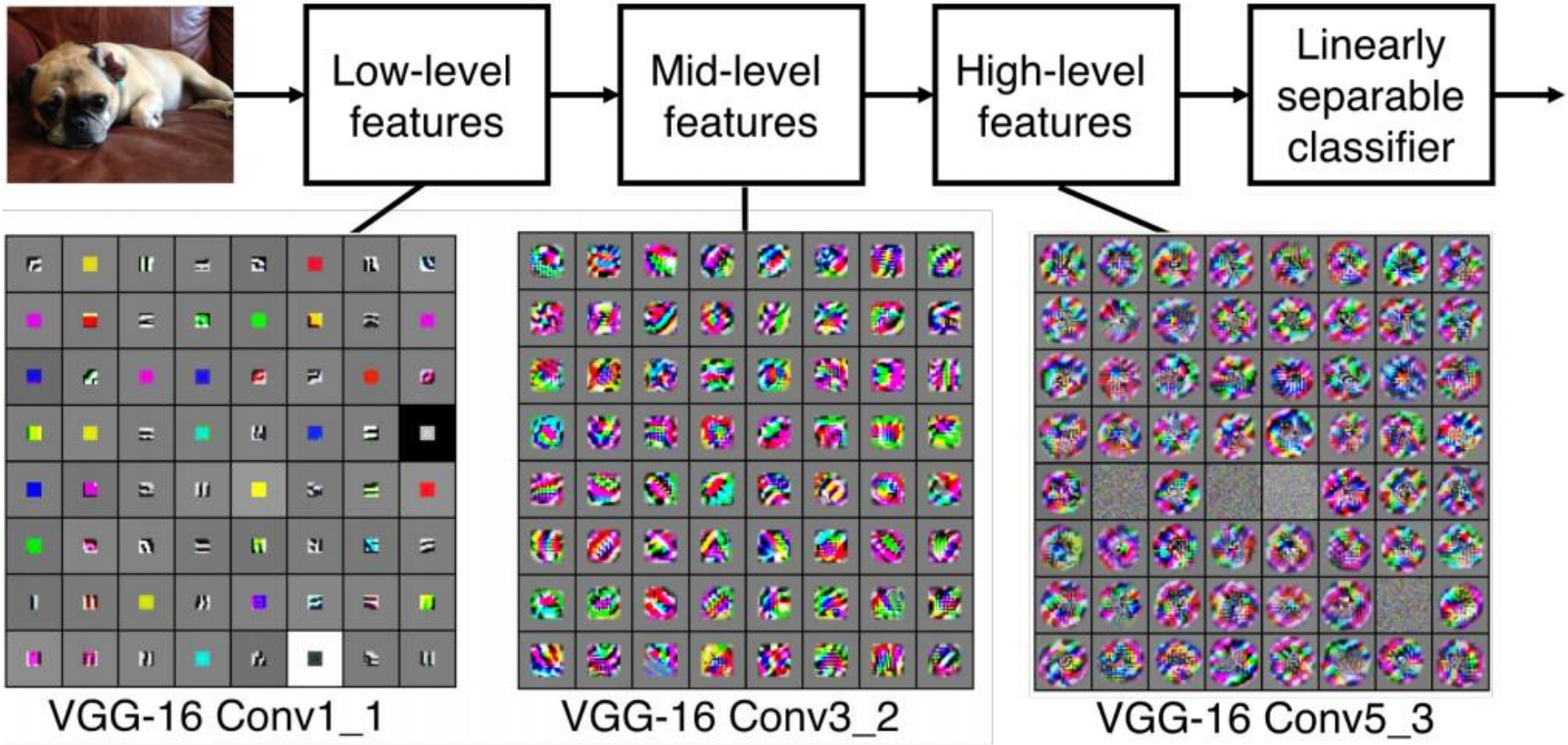






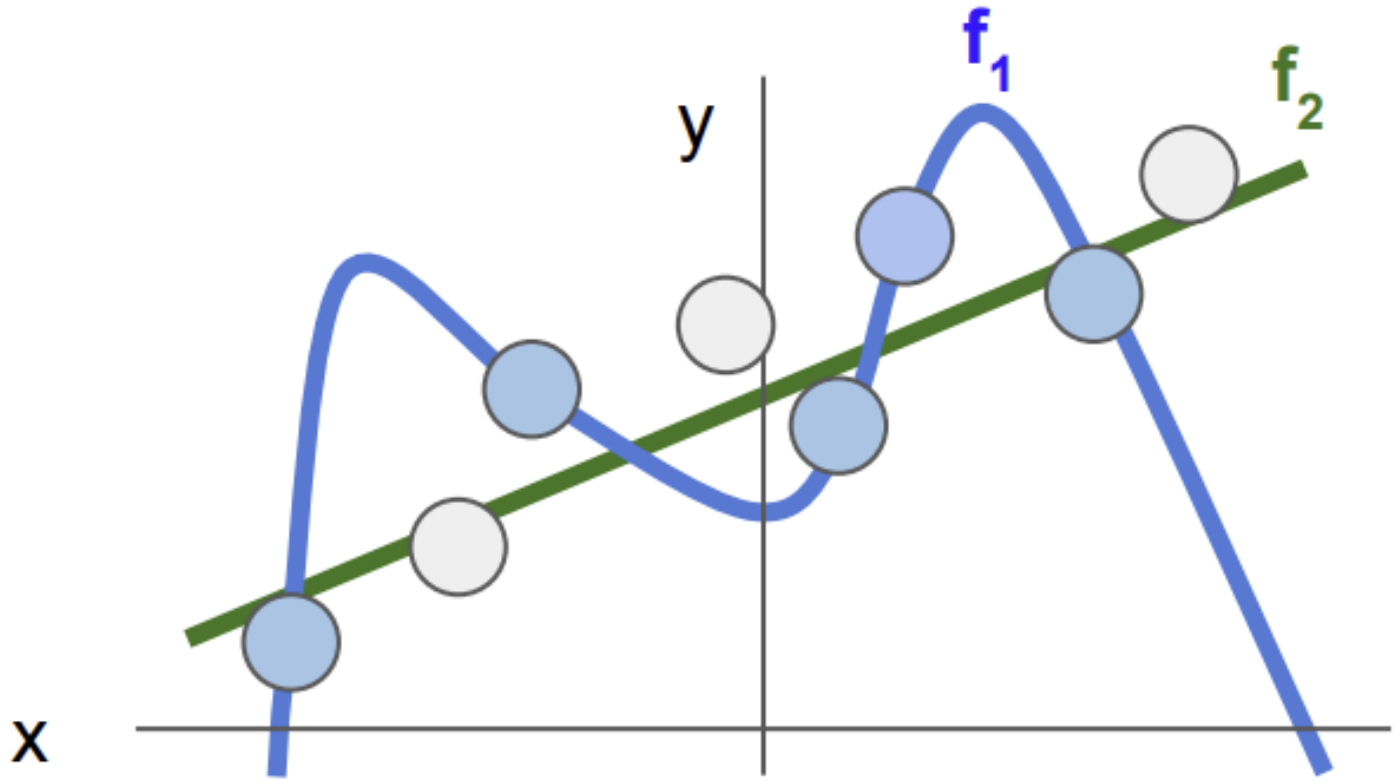
CNN

- 이미지 분류모델 – CNN



CNN

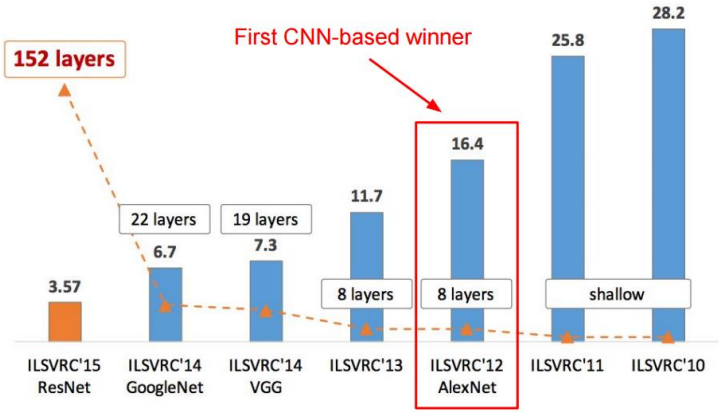
- 이미지 분류모델 – CNN



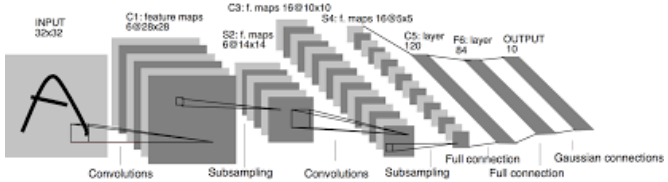
http://cs231n.stanford.edu/slides/2022/lecture_3_ruohan.pdf

CNN

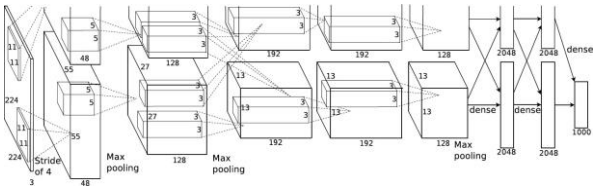
- Evolution of neural net architectures
  - Components: CNN, Pooling



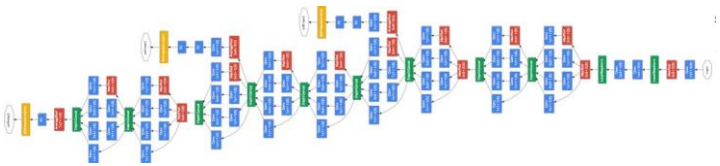
LeNet



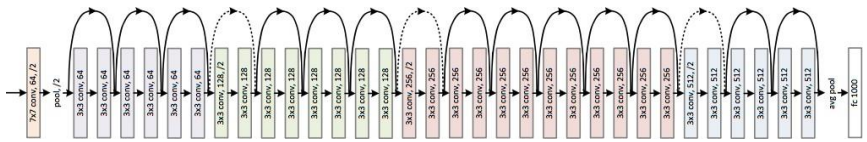
AlexNet



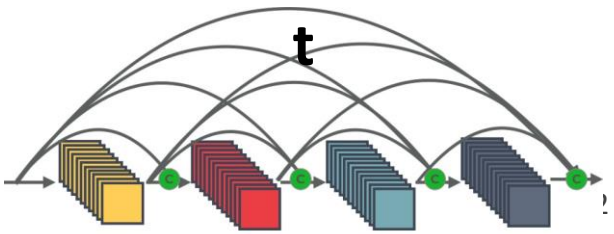
InceptionNet



ResNet



DenseNet



CNN

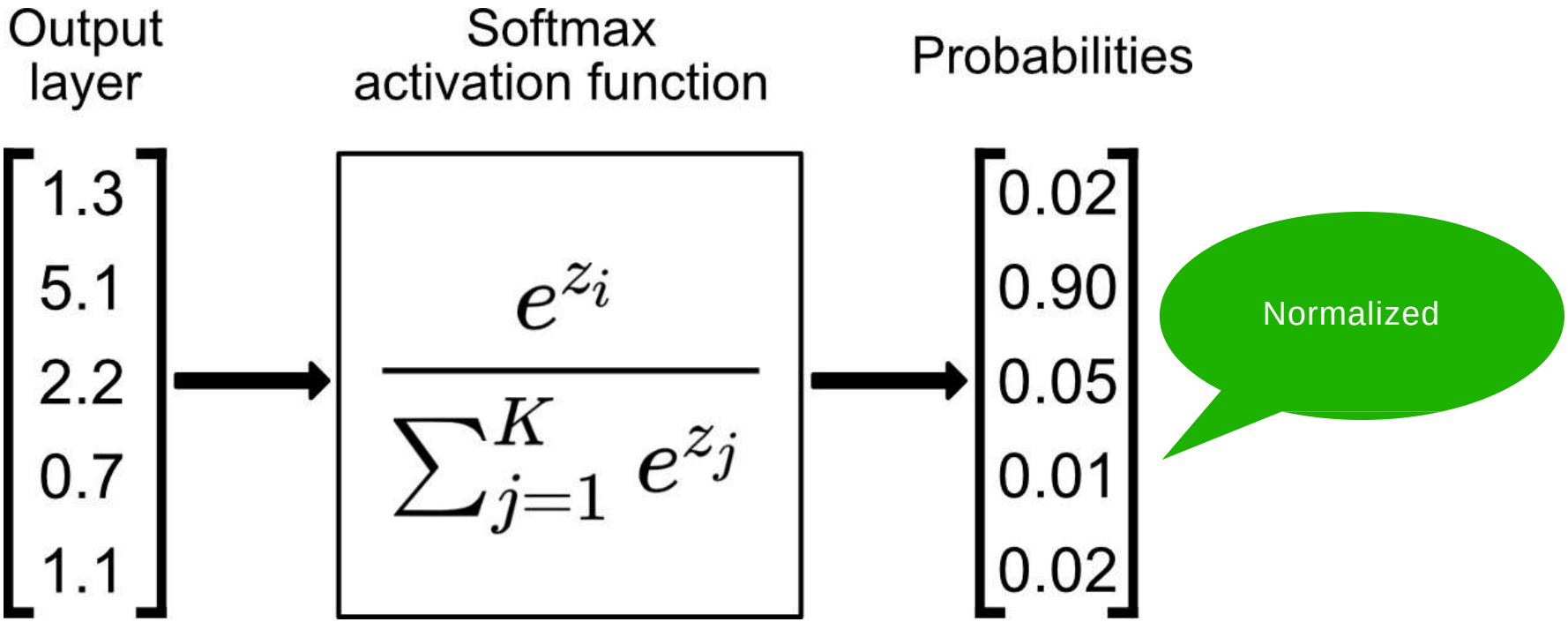
- **Multi-Class classification with Softmax**

Input

$$p_i(\boldsymbol{x}) = \frac{\exp\left(f_i(\boldsymbol{x})\right)}{\sum_{j=1}^{N} \exp\left(f_j(\boldsymbol{x})\right)}, \quad \text{softmax}$$

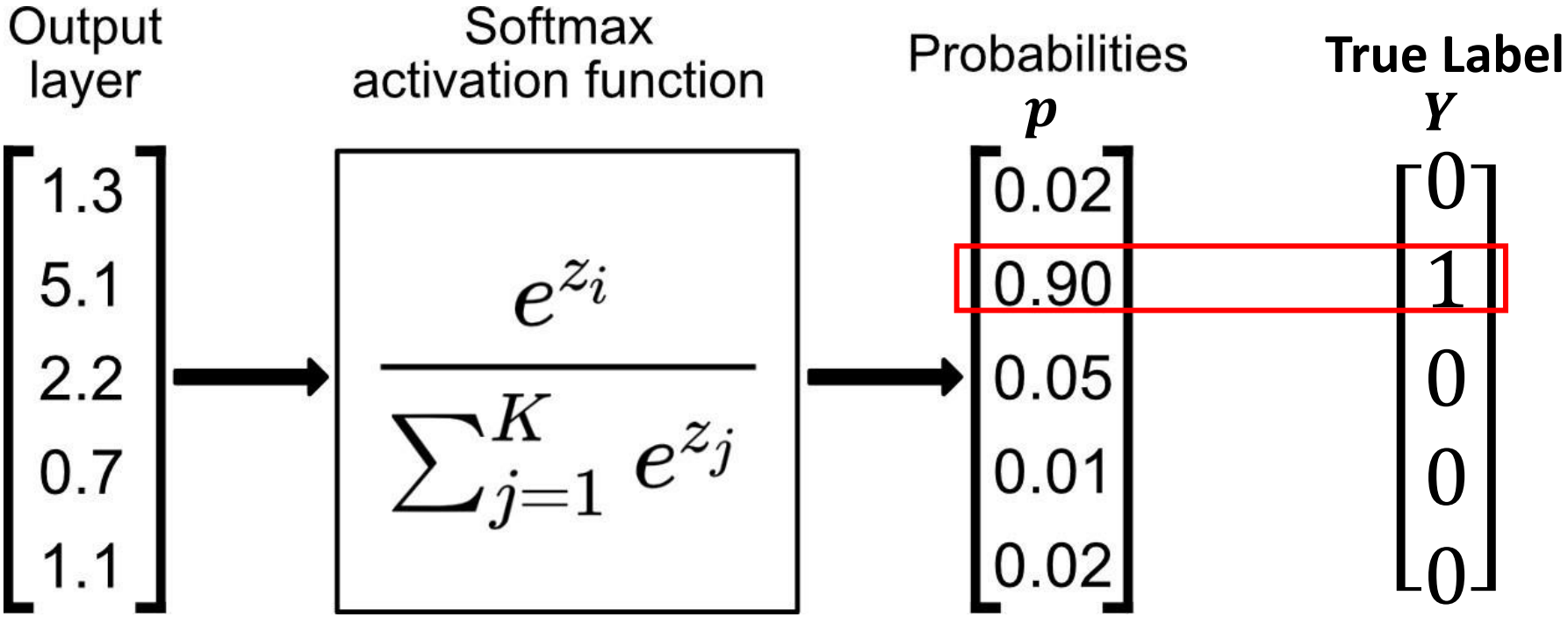
CNN

- **Multi-Class classification with Softmax**



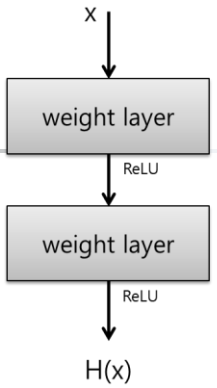
CNN

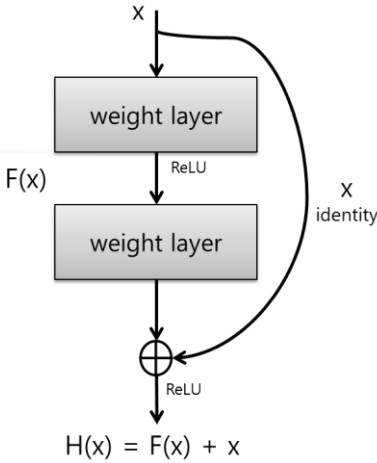
- **Multi-Class classification with Softmax**
- $\boldsymbol{Loss_{CrossEntrophy} = \sum_i -Y_i log(p_i) = -log(0.9)}$



CNN

- 이미지 분류모델 – Residual Network



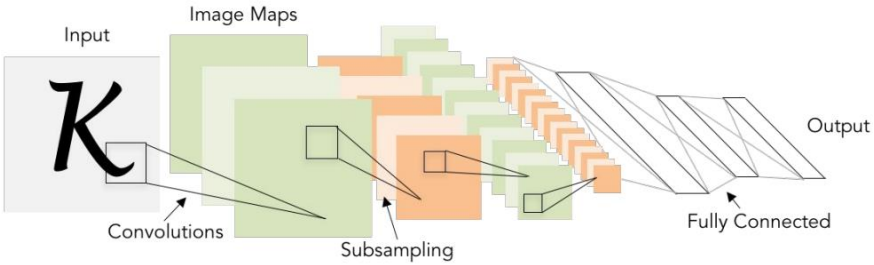


기존 방식

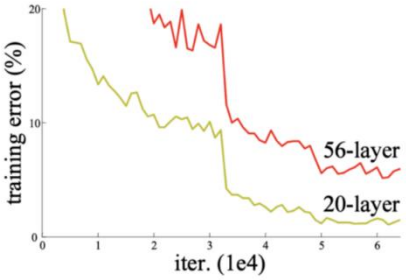
Residual block

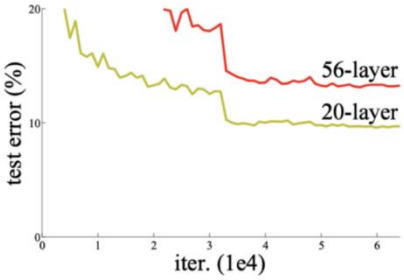
Review: LeNet-5

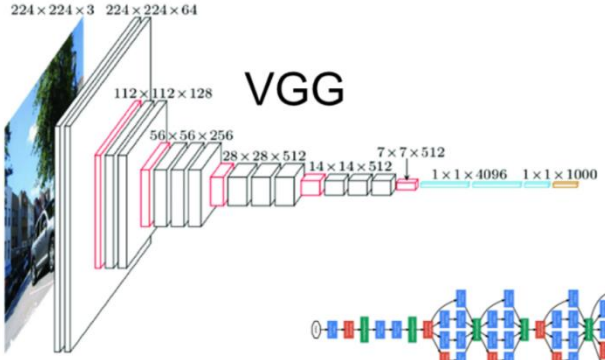
[LeCun et al., 1998]

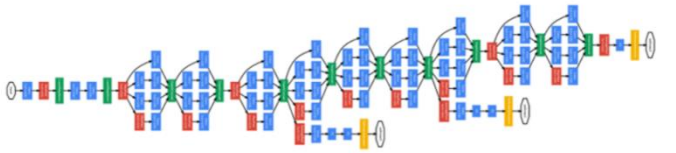


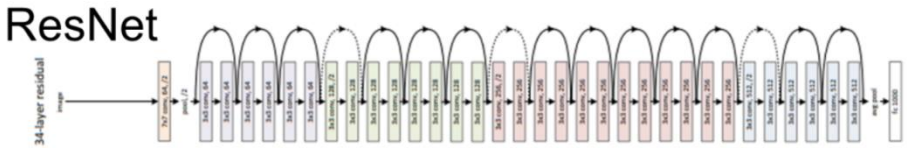
Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-FC-FC]



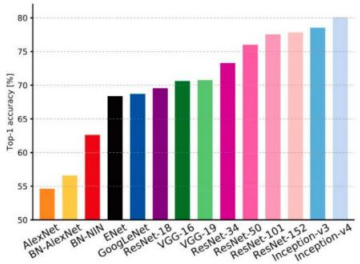


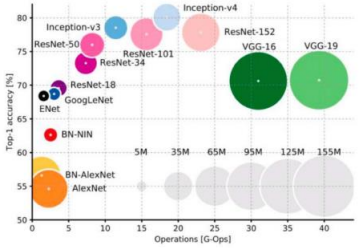






Comparing complexity...



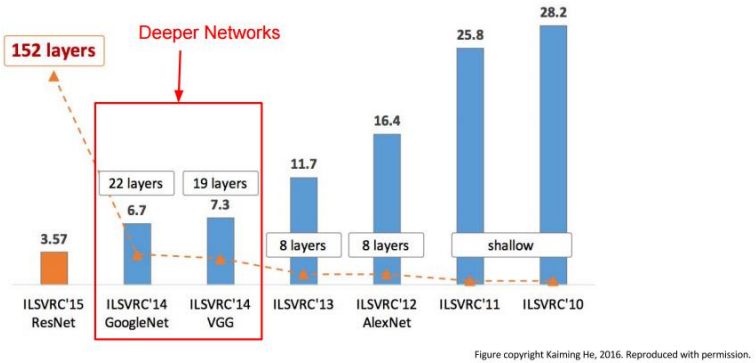


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

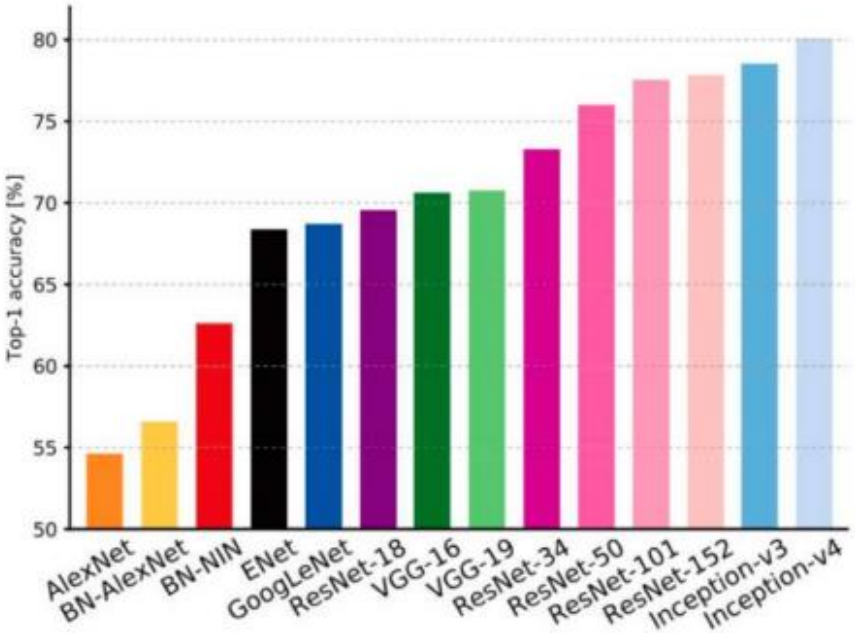
https://oi.readthedocs.io/en/latest/computer_vision/cnn/densenet.html
https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/
https://glassboxmedicine.com/2020/12/08/using-predefined-and-pretrained-cnns-in-pytorch-tutorial-with-code/

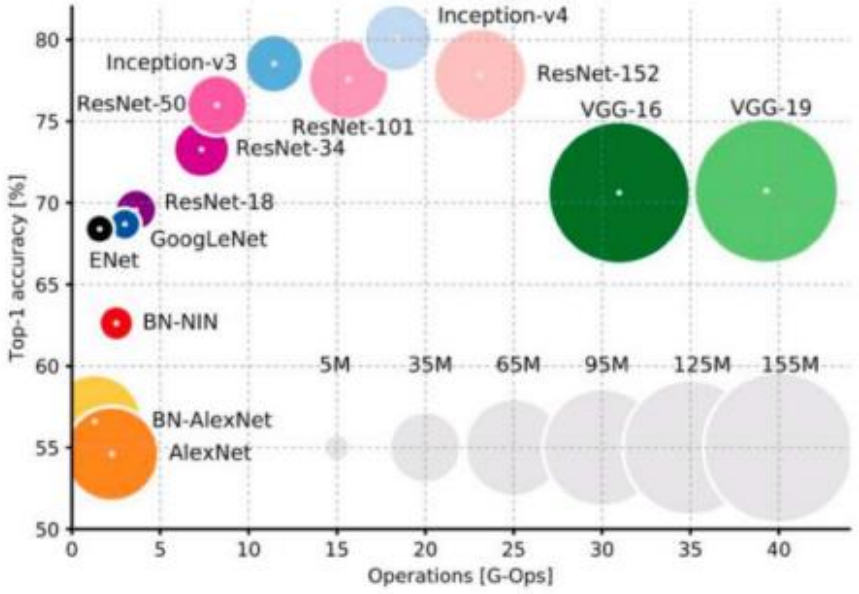
CNN

- 이미지 분류모델 – Residual Network



Comparing complexity...





An Analysis of Deep Neural Network Models for Practical Applications, 2017.

https://oi.readthedocs.io/en/latest/computer_vision/cnn/densenet.html
https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/
https://glassboxmedicine.com/2020/12/08/using-predefined-and-pretrained-cnns-in-pytorch-tutorial-with-code/

CNN

- 다양한 이미지 분류모델
  - AlexNet
  - VGG
  - ResNet
  - SqueezeNet
  - DenseNet
  - Inception v3
  - GoogLeNet
  - ShuffleNet v2
  - MobileNetV2
  - MobileNetV3
  - ResNeXt
  - Wide ResNet
  - MNASNet
  - **EfficientNet**
  - RegNet
  - VisionTransformer
  - ConvNeXt

Model	Acc@1	Acc@5
AlexNet	56.522	79.066
VGG-11	69.020	88.628
VGG-13	69.928	89.246
VGG-16	71.592	90.382
VGG-19	72.376	90.876
VGG-11 with batch normalization	70.370	89.810
VGG-13 with batch normalization	71.586	90.374
VGG-16 with batch normalization	73.360	91.516
VGG-19 with batch normalization	74.218	91.842
ResNet-18	69.758	89.078
ResNet-34	73.314	91.420
ResNet-50	76.130	92.862
ResNet-101	77.374	93.546
ResNet-152	78.312	94.046
SqueezeNet 1.0	58.092	80.420
SqueezeNet 1.1	58.178	80.624
Densenet-121	74.434	91.972
Densenet-169	75.600	92.806
Densenet-201	76.896	93.370
Densenet-161	77.138	93.560
Inception v3	77.294	93.450
GoogleNet	69.778	89.530
ShuffleNet V2 x1.0	69.362	88.316
ShuffleNet V2 x0.5	60.552	81.746
MobileNet V2	71.878	90.286
MobileNet V3 Large	74.042	91.340
MobileNet V3 Small	67.668	87.402
ResNeXt-50-32x4d	77.618	93.698
ResNeXt-101-32x8d	79.312	94.526
Wide ResNet-50-2	78.468	94.086
Wide ResNet-101-2	78.848	94.284
MNASNet 1.0	73.456	91.510
MNASNet 0.5	67.734	87.490

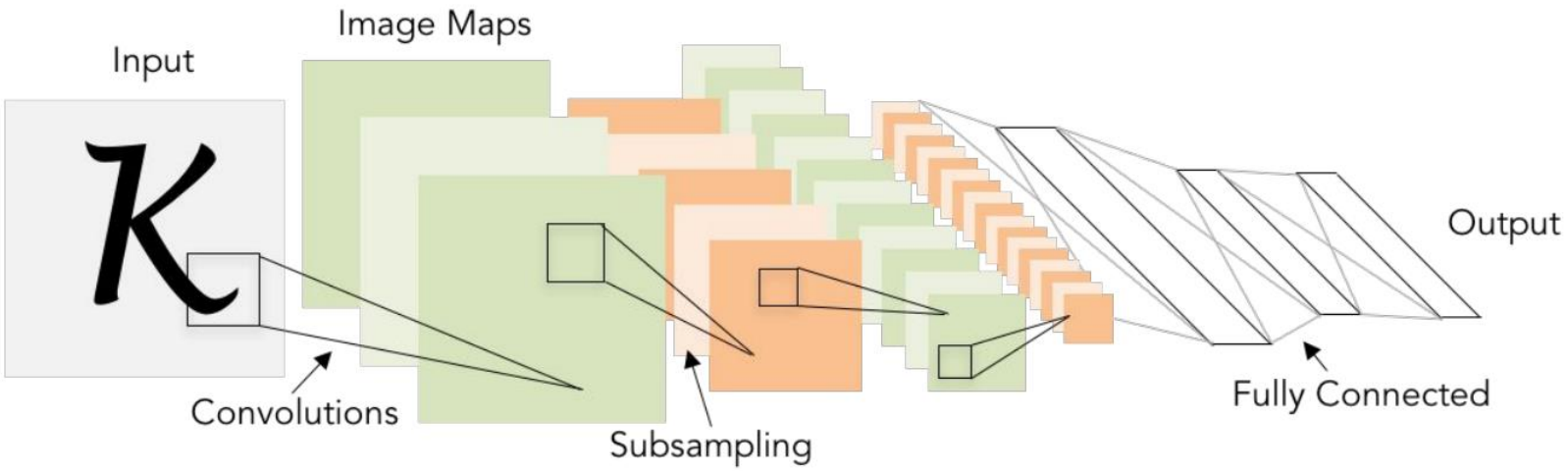
https://py

CNN

- **LeNet**

Review: LeNet-5

[LeCun et al., 1998]

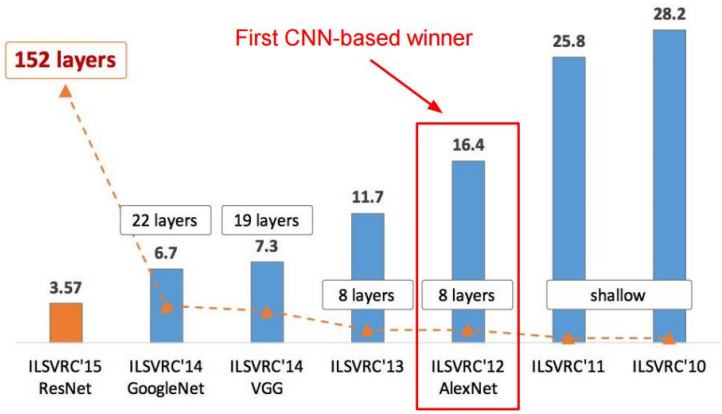


Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-FC-FC]

https://pytorch.org/vision/stable/models.html

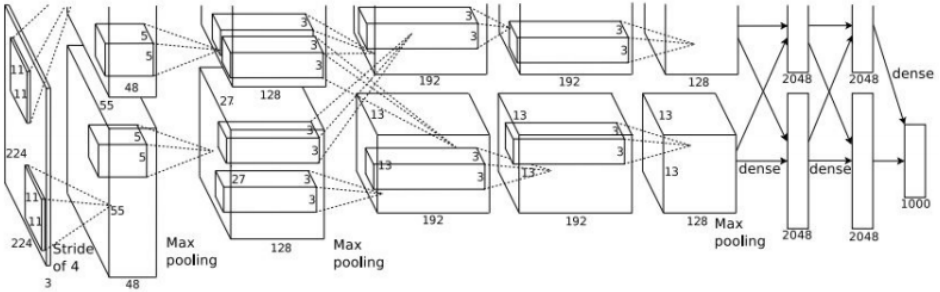
CNN

- **AlexNet**



Case Study: AlexNet

[Krizhevsky et al. 2012]

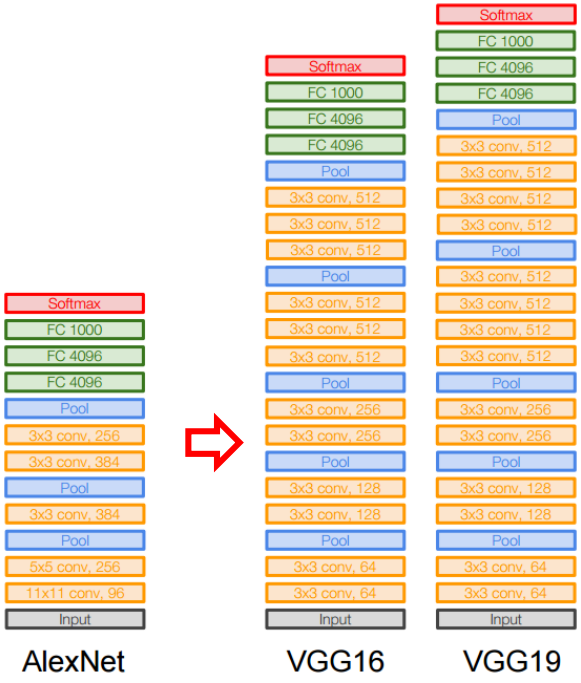


Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] 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

- **VGG Network, 2014**

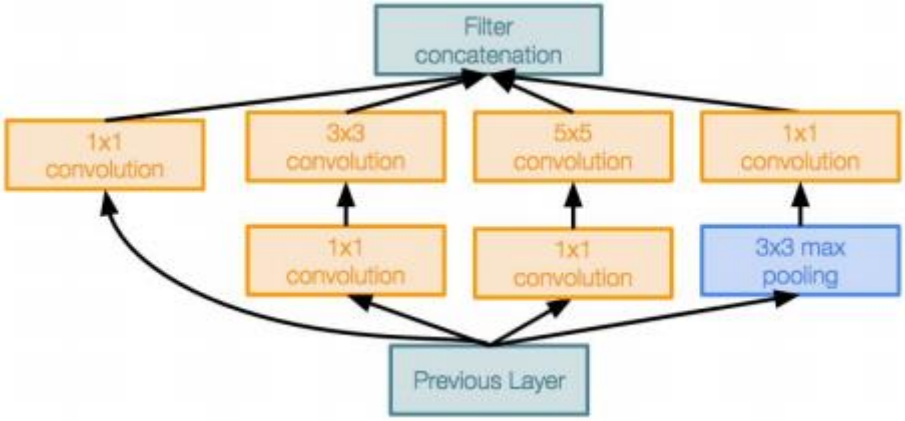


INPUT: [224x224x3] memory: 224*224*3=150K params: 0 (not counting biases)
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

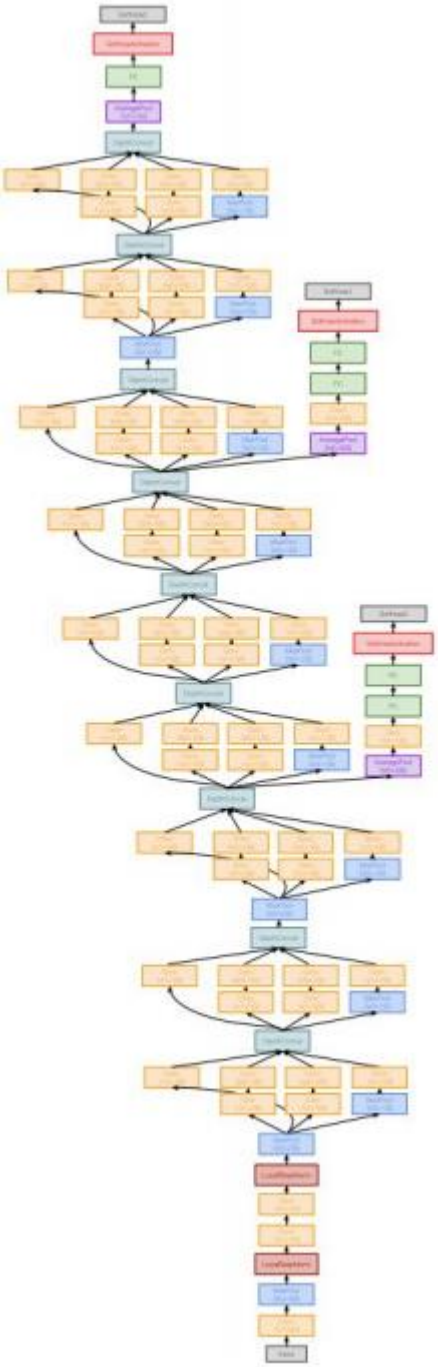
TOTAL memory: 24M * 4 bytes ~= 96MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

CNN

- **GoogleNet, 2014**

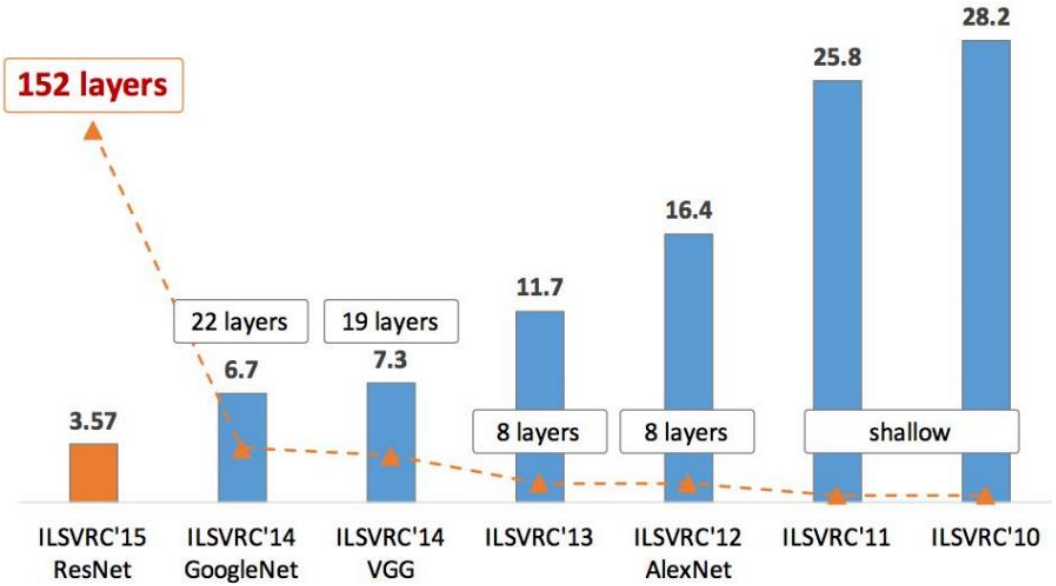


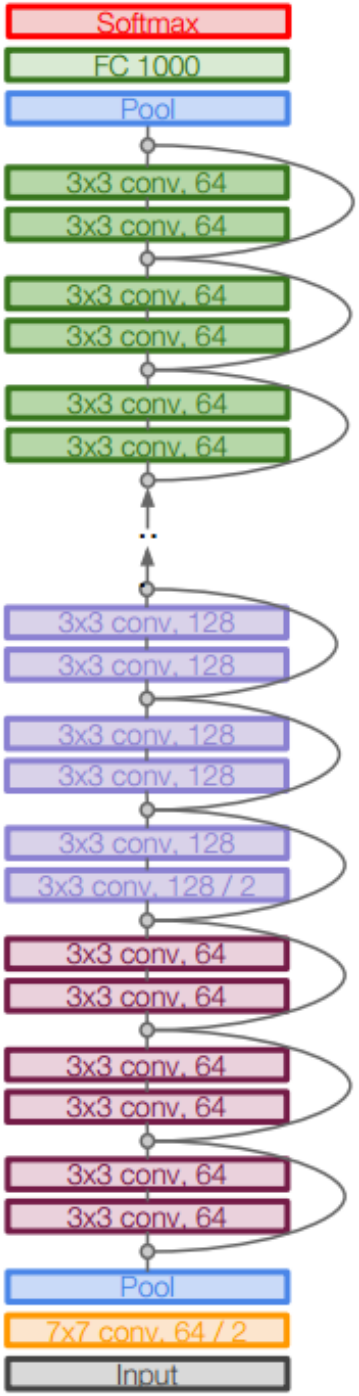
Inception Module



CNN

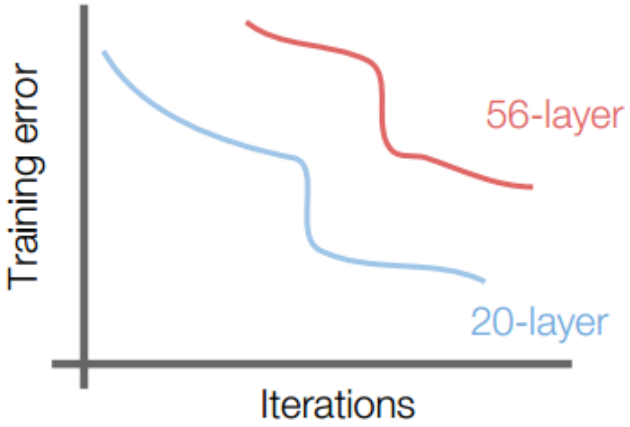
- ResNet, 2015
  - Very deep networks using residual connections
  - VGG: 19 layers. ResNet: 152 layers.

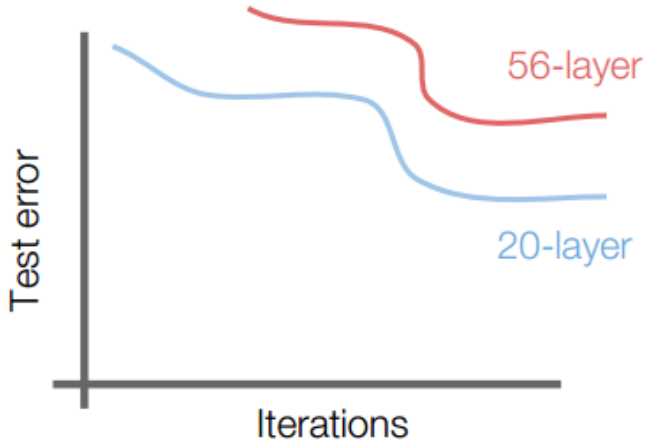


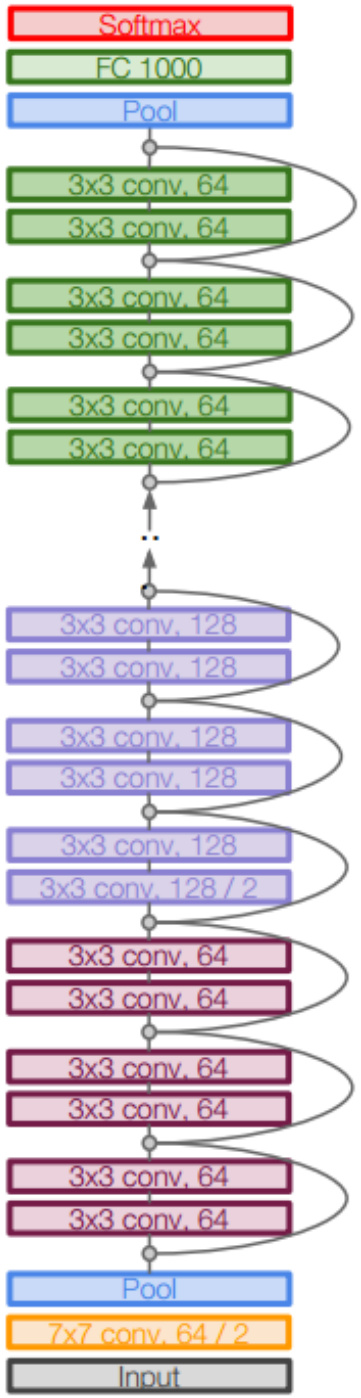


CNN

- ResNet, 2015
  - Very deep networks using residual connections
  - VGG: 19 layers. ResNet: 152 layers.
  - Add more layers… sufficient?
  - No! Some problems:
    1) Vanishing gradients: more layers ➔ more likely
    2) Instability: deeper models are harder to optimiz

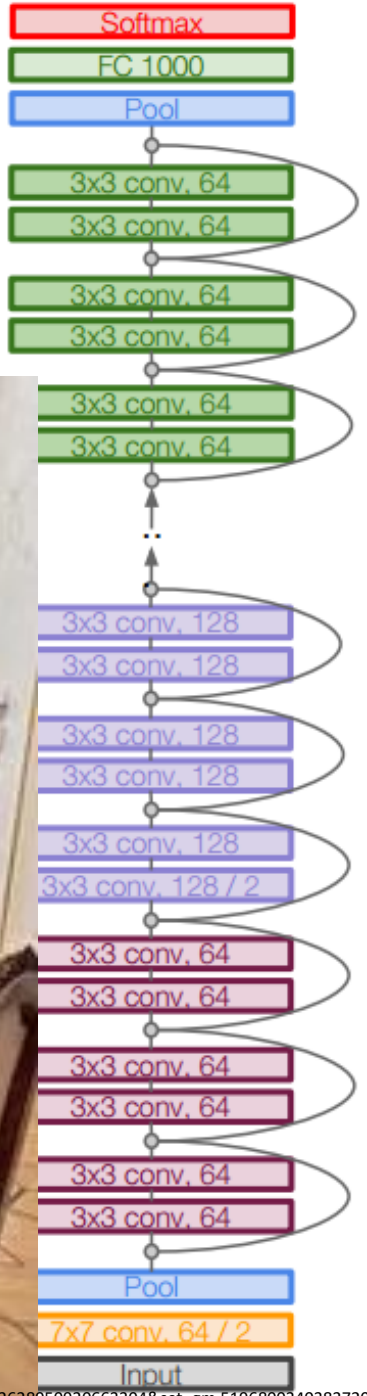






CNN

- **ResNet, 2015**



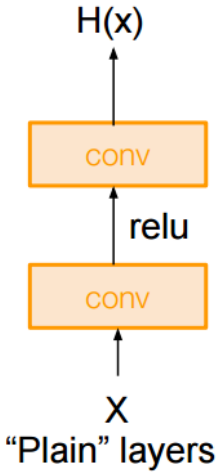
https://www.facebook.com/photo?fbid=2628050920663304&set=gm.519680034038372

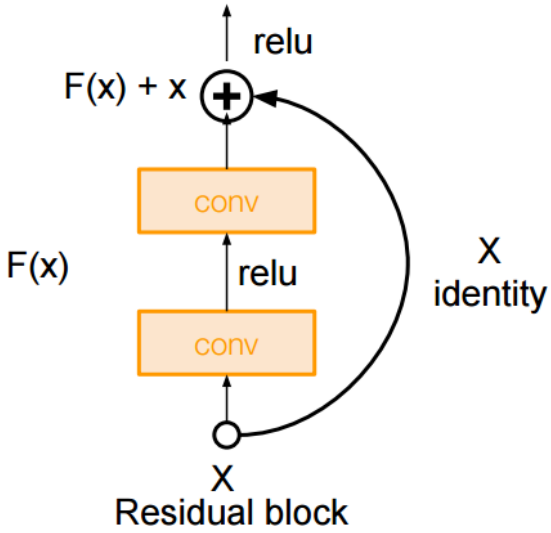
CNN

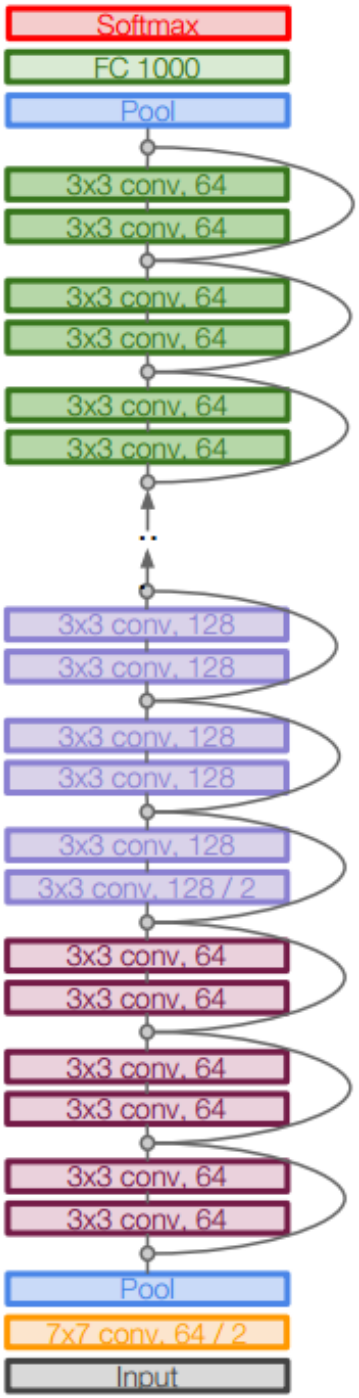
코드 실습!

https://github.com/airobotlab/advanced_cnn/blob/main/05_01_CNN_%EA%B8%B0%EC%B4%88_%EB%B3%B5%EC%8A%B5_%EC%99%84%EC%84%B1.ipynb

- ResNet, 2015
  - Very deep networks using residual connections
  - 152-layer







CNN

- **Quiz!!**
- **Which of the following is not true?**

  A. **Adding more layers can improve the performance of a neural network.**

  B. **Residual connections help deal with vanishing gradients.**

  C. **CNN architectures use no more than ~20 layers to avoid problems such as vanishing gradients.**

Transfer Learning

Transfer Learning

- 다양한 이미지 분류모델

```python
import torchvision.models as models
resnet18 = models.resnet18(pretrained=True)
alexnet = models.alexnet(pretrained=True)
squeezenet = models.squeezenet1_0(pretrained=True)
vgg16 = models.vgg16(pretrained=True)
densenet = models.densenet161(pretrained=True)
inception = models.inception_v3(pretrained=True)
googlenet = models.googlenet(pretrained=True)
shufflenet = models.shufflenet_v2_x1_0(pretrained=True)
mobilenet_v2 = models.mobilenet_v2(pretrained=True)
mobilenet_v3_large = models.mobilenet_v3_large(pretrained=True)
mobilenet_v3_small = models.mobilenet_v3_small(pretrained=True)
resnext50_32x4d = models.resnext50_32x4d(pretrained=True)
wide_resnet50_2 = models.wide_resnet50_2(pretrained=True)
mnasnet = models.mnasnet1_0(pretrained=True)
```

Model	Acc@1	Acc@5
AlexNet	56.522	79.066
VGG-11	69.020	88.628
VGG-13	69.928	89.246
VGG-16	71.592	90.382
VGG-19	72.376	90.876
VGG-11 with batch normalization	70.370	89.810
VGG-13 with batch normalization	71.586	90.374
VGG-16 with batch normalization	73.360	91.516
Wide ResNet-50-2	78.468	94.086
Wide ResNet-101-2	78.848	94.284
MNASNet 1.0	73.456	91.510
MNASNet 0.5	67.734	87.490

https://py

Transfer Learning

- 다양한 이미지 분류모델

Model	Acc@1	Acc@5
AlexNet	56.522	79.066
VGG-11	69.020	88.628
VGG-13	69.928	89.246
VGG-16	71.592	90.382
VGG-19	72.376	90.876
VGG-11 with batch normalization	70.370	89.810
VGG-13 with batch normalization	71.586	90.374
VGG-16 with batch normalization	73.360	91.516
Wide ResNet-50-2	78.468	94.086
Wide ResNet-101-2	78.848	94.284
MNASNet 1.0	73.456	91.510
MNASNet 0.5	67.734	87.490

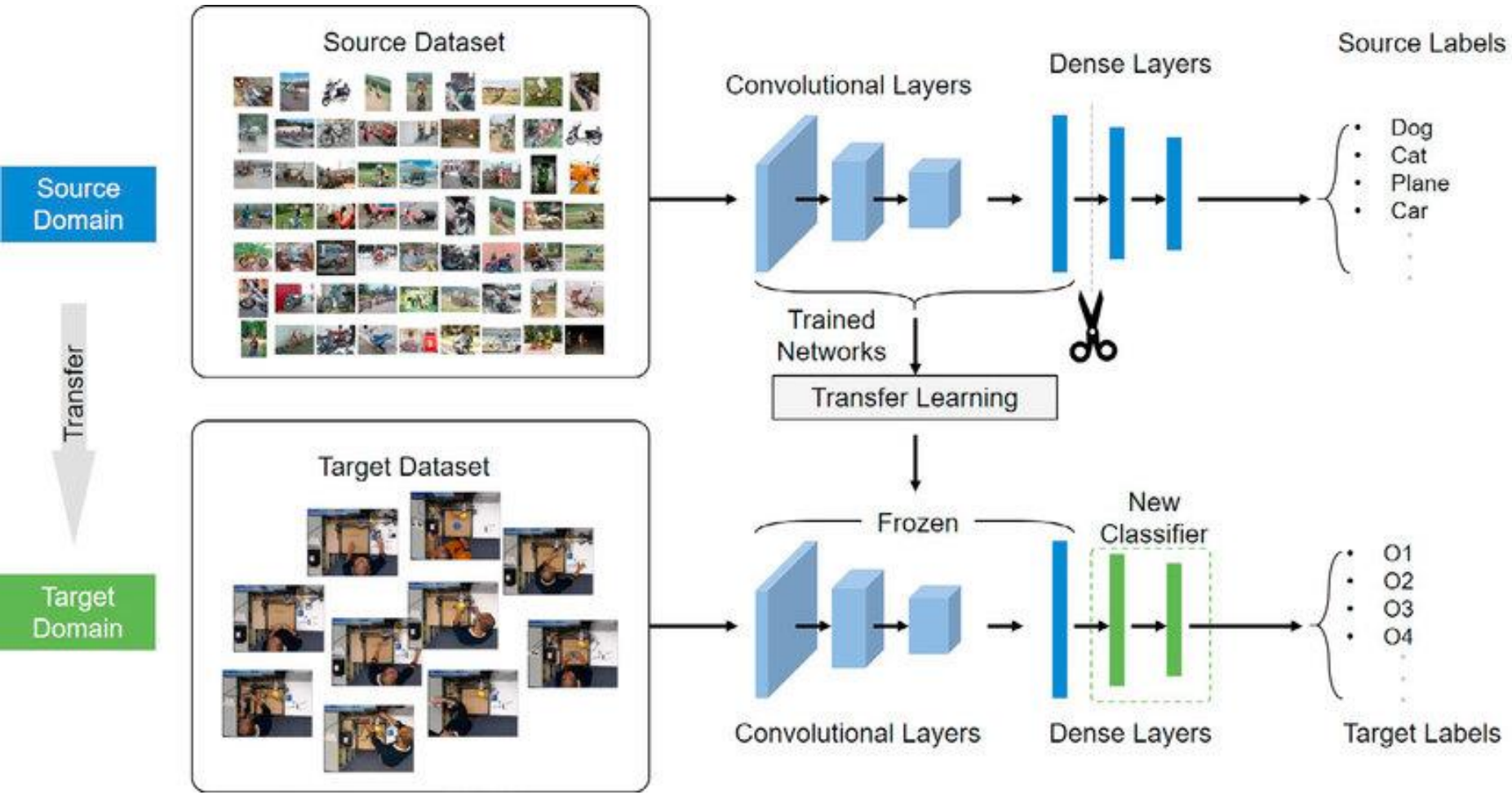
```
1  import torchvision.models as models
2  resnet18 = models.resnet18(pretrained=True)
3  alexnet
4  squeeze
5  vgg16 =
6  densene
7  incepti
8  googlen
9  shuffle                                                    rue)
10 mobilen
11 mobilen                                                    rained=True)
12 mobilenet_v3_small = models.mobilenet_v3_small(pretrained=True)
13 resnext50_32x4d = models.resnext50_32x4d(pretrained=True)
14 wide_resnet50_2 = models.wide_resnet50_2(pretrained=True)
15 mnasnet = models.mnasnet1_0(pretrained=True)
```

Transfer Learning

"You need a lot of a data if you want to train/use CNNs"

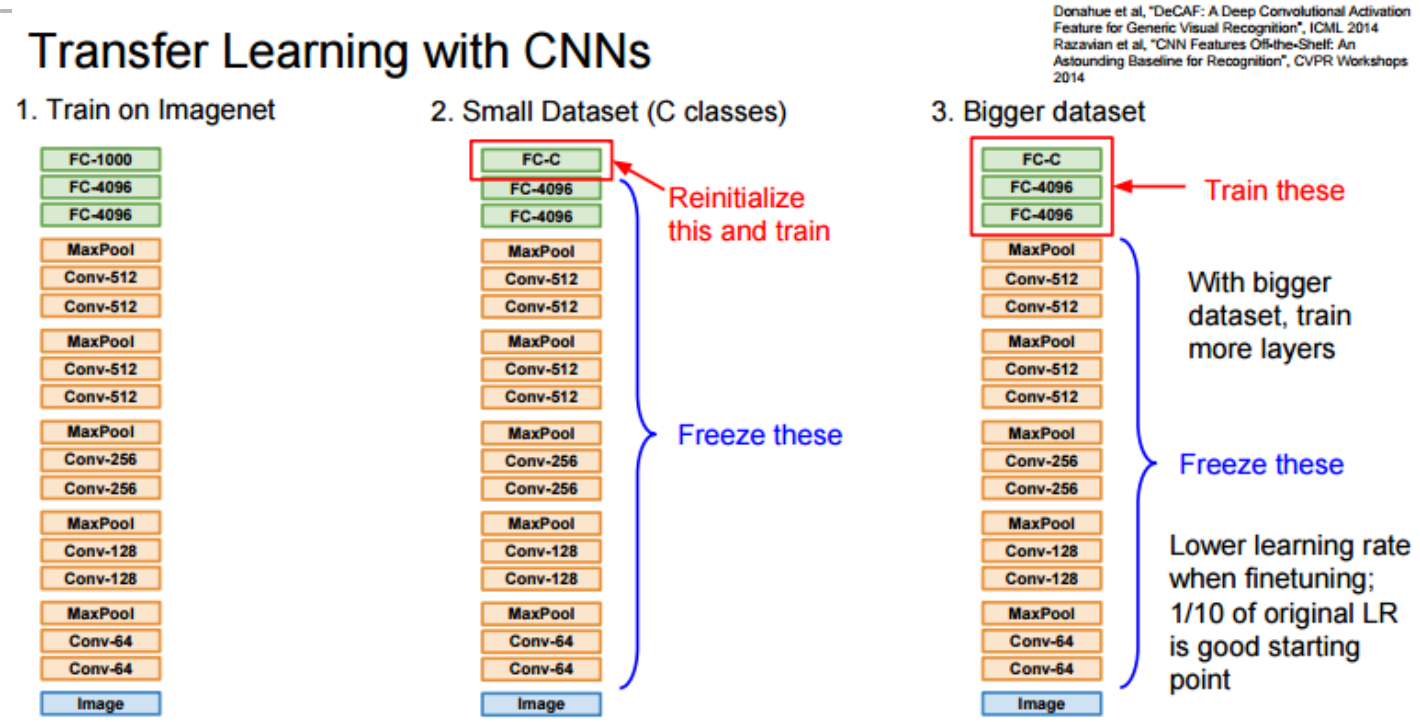
https://py

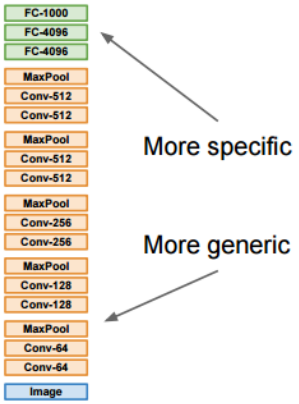
Transfer Learning



https://www.researchgate.net/figure/The-architecture-of-our-transfer-learning-model_fig4_342400905

Transfer Learning



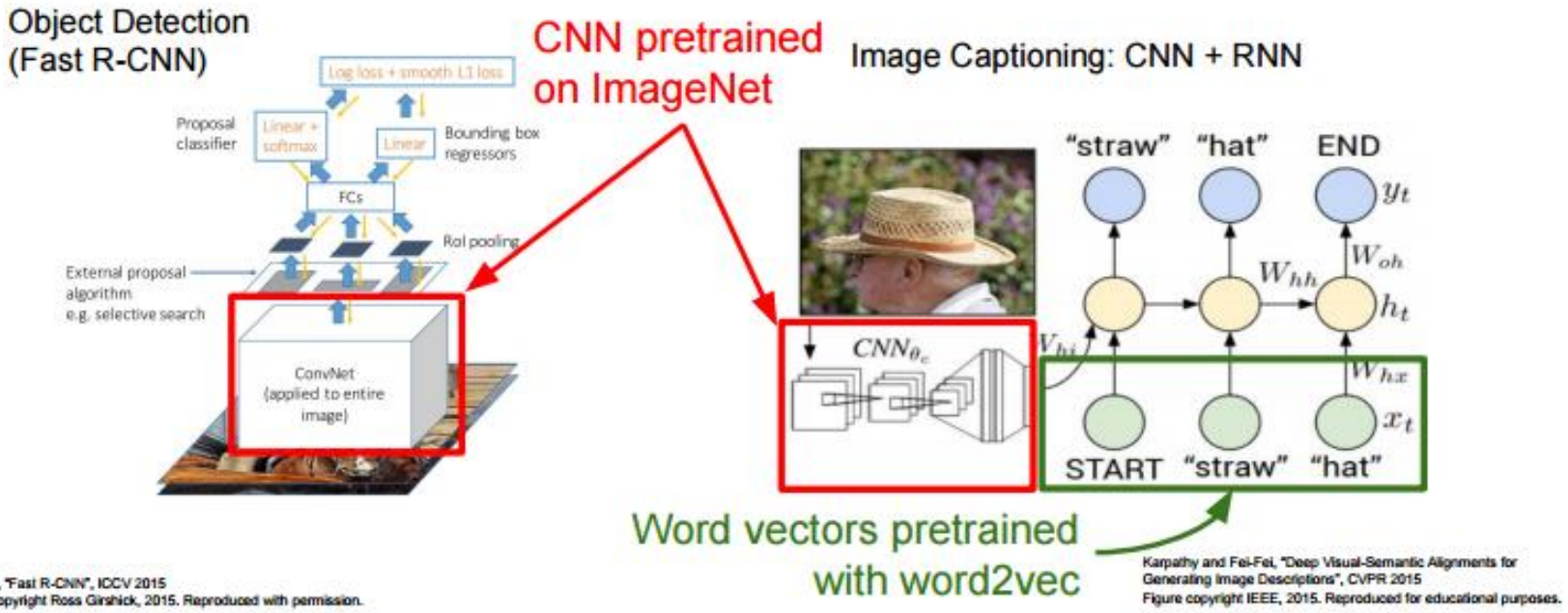


	very similar dataset	very different dataset
very little data	Use Linear Classifier on top layer	You're in trouble… Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

https://pytorch.org/vision/stable/models.html

Transfer Learning

Transfer learning with CNNs is pervasive…
(it's the norm, not an exception)



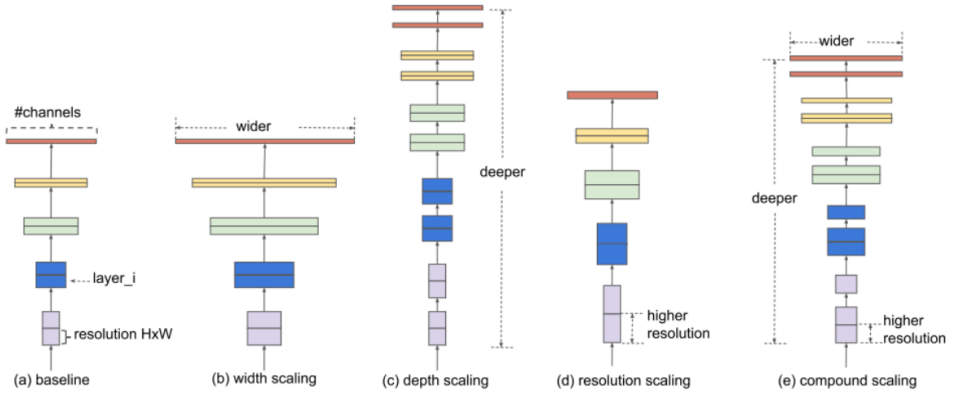
Girshick, "Fast R-CNN", ICCV 2015
Figure copyright Ross Girshick, 2015. Reproduced with permission.

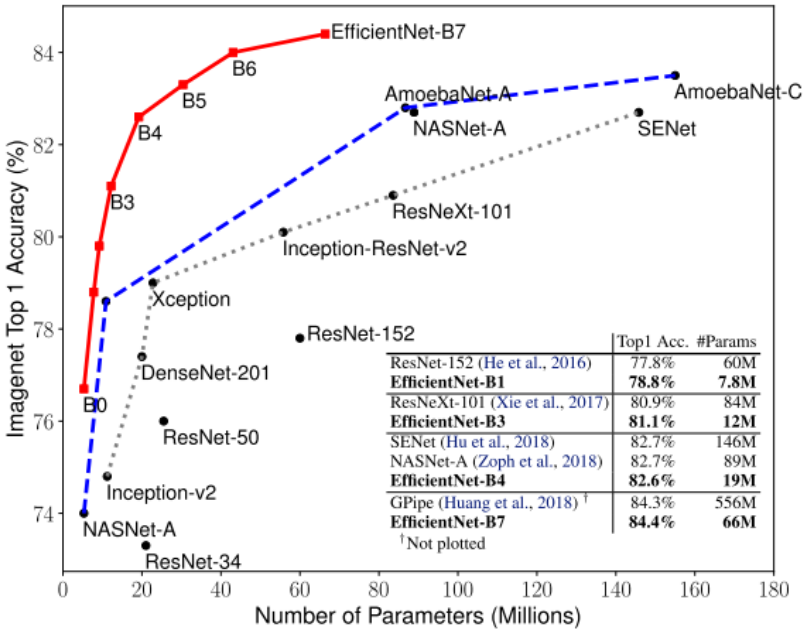
Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015
Figure copyright IEEE, 2015. Reproduced for educational purposes.

https://pytorch.org/vision/stable/models.html

Transfer Learning

- **SOTA 이미지 분류모델 : EfficientNet**
  - Find Small & Powerful model with AutoML
  - 참고자료 : https://keep-steady.tistory.com/35
  - Compound scaling 방법으로 최적화
    - Channel 수
    - Layer 수
    - 입력이미지의 해상도





https://arxiv.org/abs/1905.11946

Transfer Learning

- **SOTA 이미지 분류모델 : EfficientNet**
  - Find Small & Powerful model with AutoML
  - 참고자료 : https://keep-steady.tistory.com/35

```
model = models.efficientnet_b4(pretrained=True).to(device)

# print(model)

for param in model.parameters():
    param.requires_grad = False # 가져온 부분은 W, b를 업데이트하지 않는다

model.classifier = nn.Sequential(
    nn.Linear(1792, 512),
    nn.ReLU(),
    nn.Linear(512, 149)
).to(device)
```

1 for regression

```
Downloading: "https://download.pytorch.org/models/efficientnet_b4_rwightman-7eb33cd5.pth" to /root/.cache/torch/hub/checkpoints/efficientnet_b4_rwightman-7eb33cd5.pth

  0%|          | 0.00/74.5M [00:00<?, ?B/s]
```


https://arxiv.org/abs/1905.11946

Transfer Learning

- **SOTA 이미지 분류모델 : EfficientNet**
  - Find Small & Powerful model with AutoML
  - 참고자료 : https://keep-steady.tistory.com/35

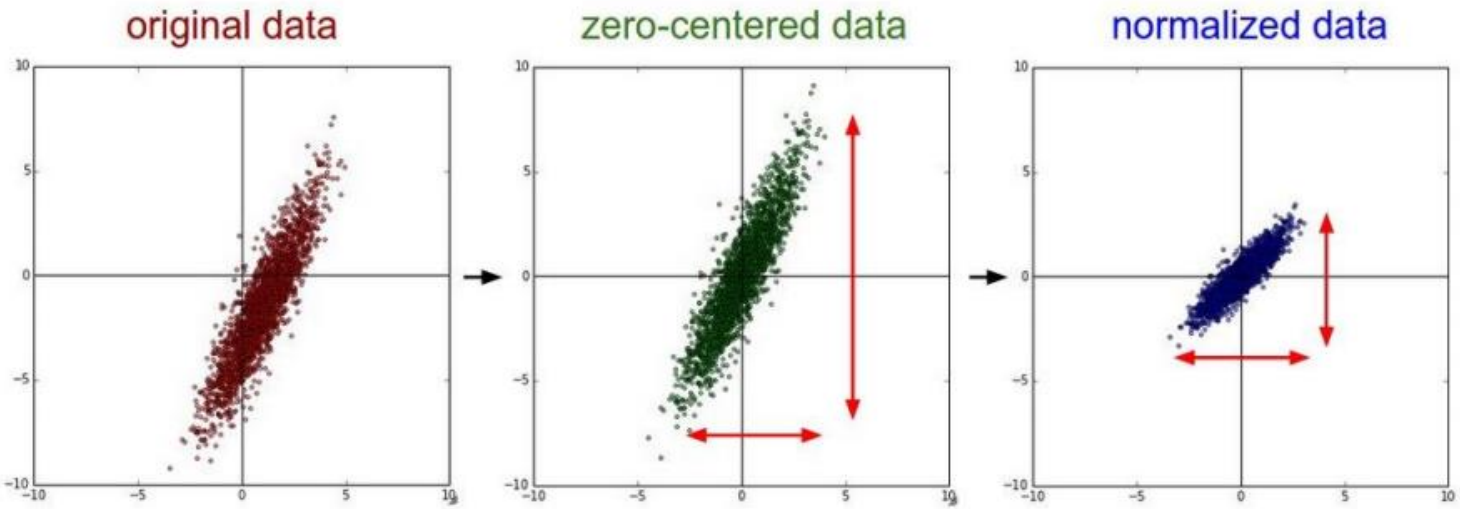
```
model = models.efficientnet_b4(pretrained=True).to(device)
```

```
import torchvision.models as models
resnet18 = models.resnet18(pretrained=True)
alexnet = models.alexnet(pretrained=True)
squeezenet = models.squeezenet1_0(pretrained=True)
vgg16 = models.vgg16(pretrained=True)
densenet = models.densenet161(pretrained=True)
inception = models.inception_v3(pretrained=True)
googlenet = models.googlenet(pretrained=True)
shufflenet = models.shufflenet_v2_x1_0(pretrained=True)
mobilenet_v2 = models.mobilenet_v2(pretrained=True)
mobilenet_v3_large = models.mobilenet_v3_large(pretrained=True)
mobilenet_v3_small = models.mobilenet_v3_small(pretrained=True)
resnext50_32x4d = models.resnext50_32x4d(pretrained=True)
wide_resnet50_2 = models.wide_resnet50_2(pretrained=True)
mnasnet = models.mnasnet1_0(pretrained=True)
```

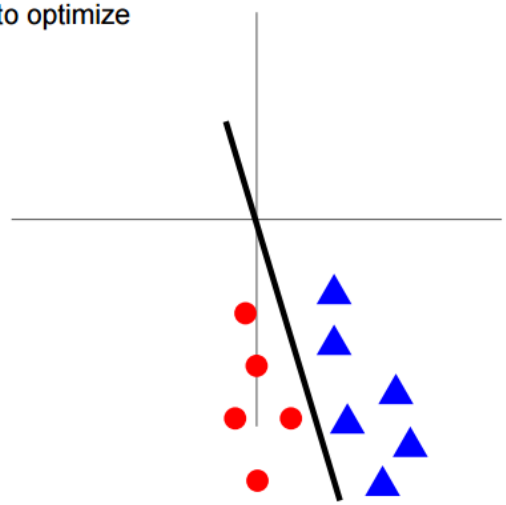

https://arxiv.org/abs/1905.11946

Deep Learning Pipeline

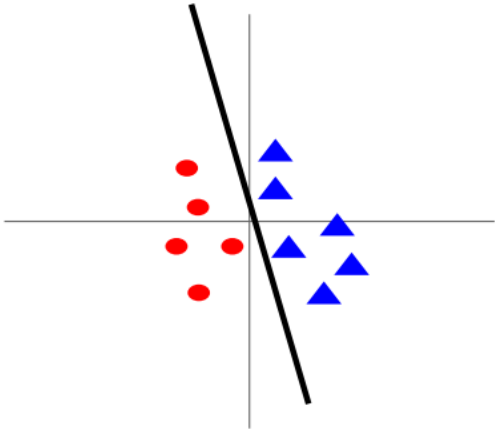
Data Preprocessing



Before normalization: classification loss very sensitive to changes in weight matrix; hard to optimize



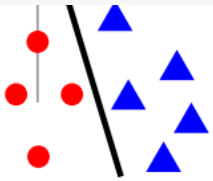
After normalization: less sensitive to small changes in weights; easier to optimize



https://pytorch.org/vision/stable/models.html

Data Preprocessing


```
data_transforms = {
    'train': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.RandomAffine(0, shear=10, scale=(0.8, 1.2)),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
    'validation': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
}
```



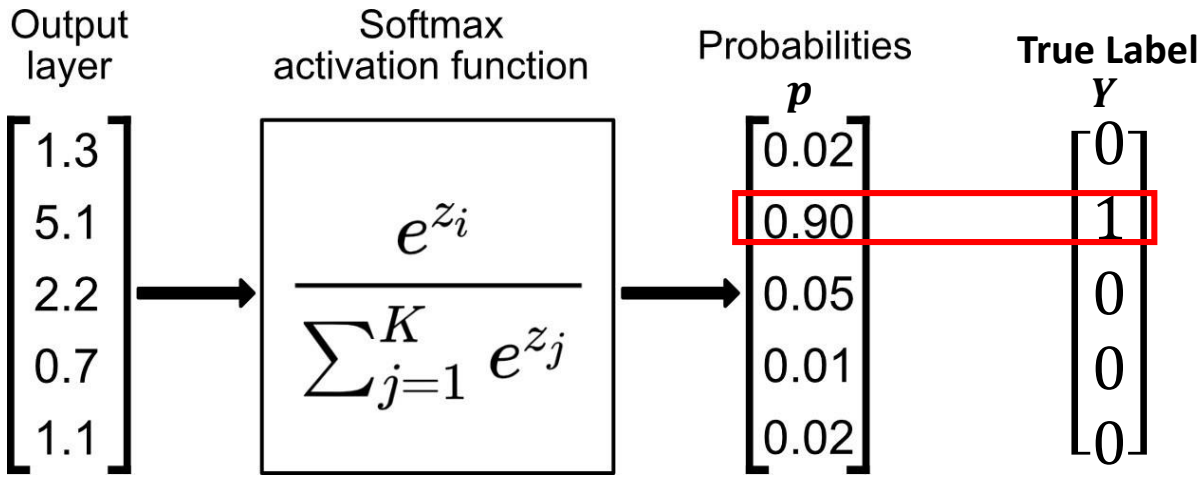
https://pytorch.org/vision/stable/models.html

CrossEntropy Loss

로스, 옵티마이저 정의

```
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.classifier.parameters(), lr=0.001)
```

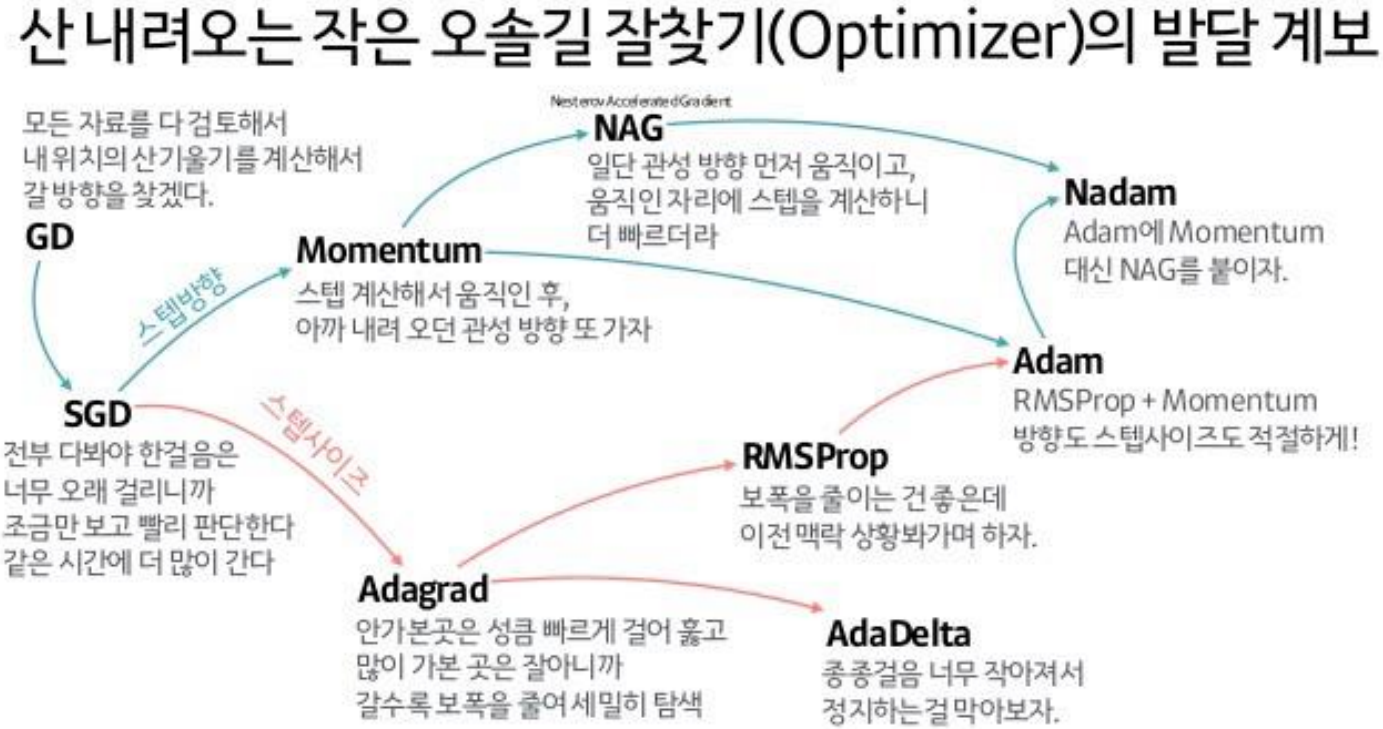
- **Multi-Class classification with Softmax**
- $Loss_{CrossEntrophy} = \sum_i -Y_i log(p_i) = -log(0.9)$



Optimizer

로스, 옵티마이저 정의

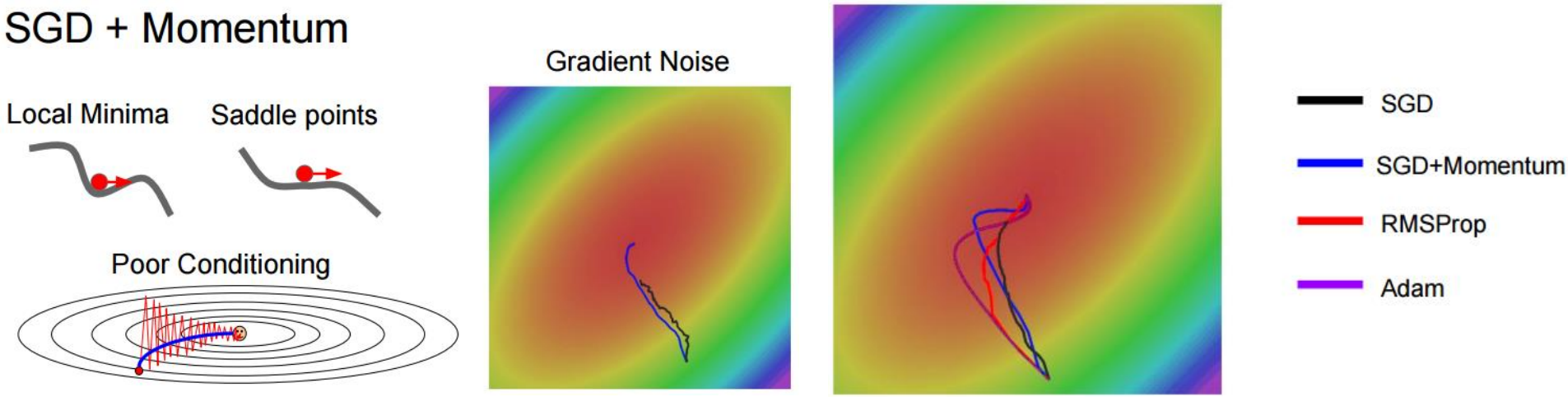
```
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.classifier.parameters(), lr=0.001)
```



Optimizer

로스, 옵티마이저 정의

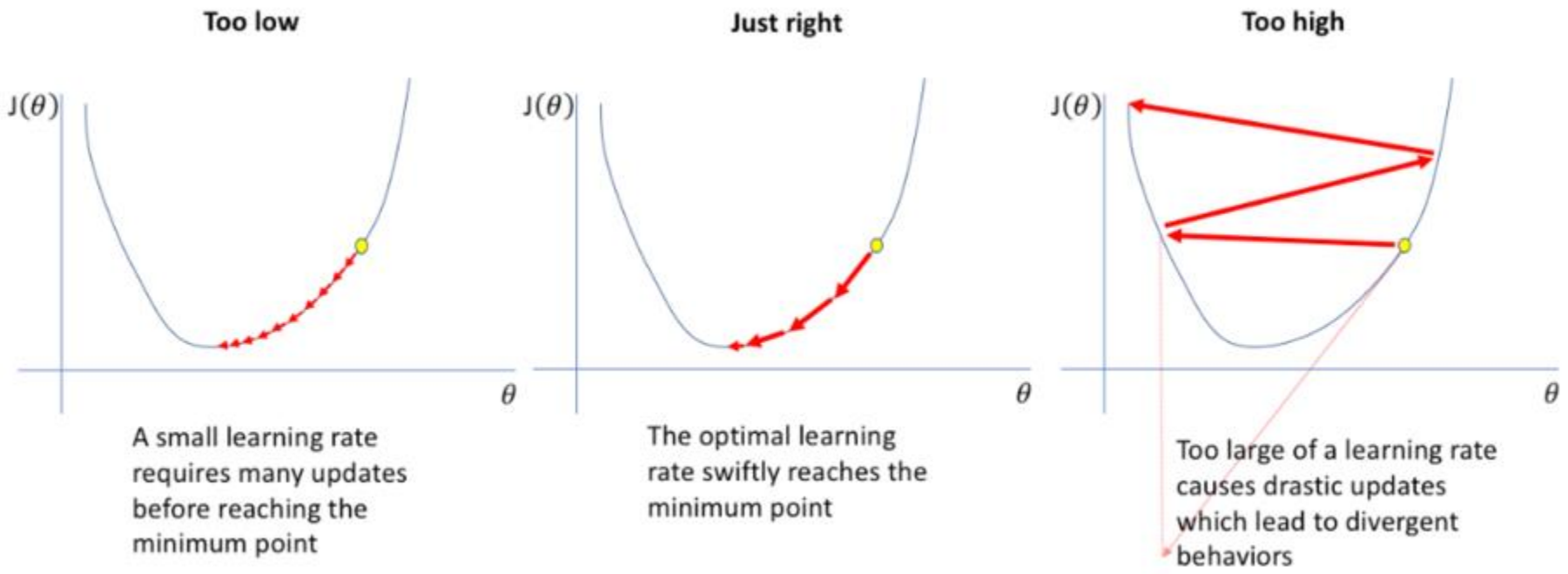
```
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.classifier.parameters(), lr=0.001)
```



HyperParameter

로스, 옵티마이저 정의

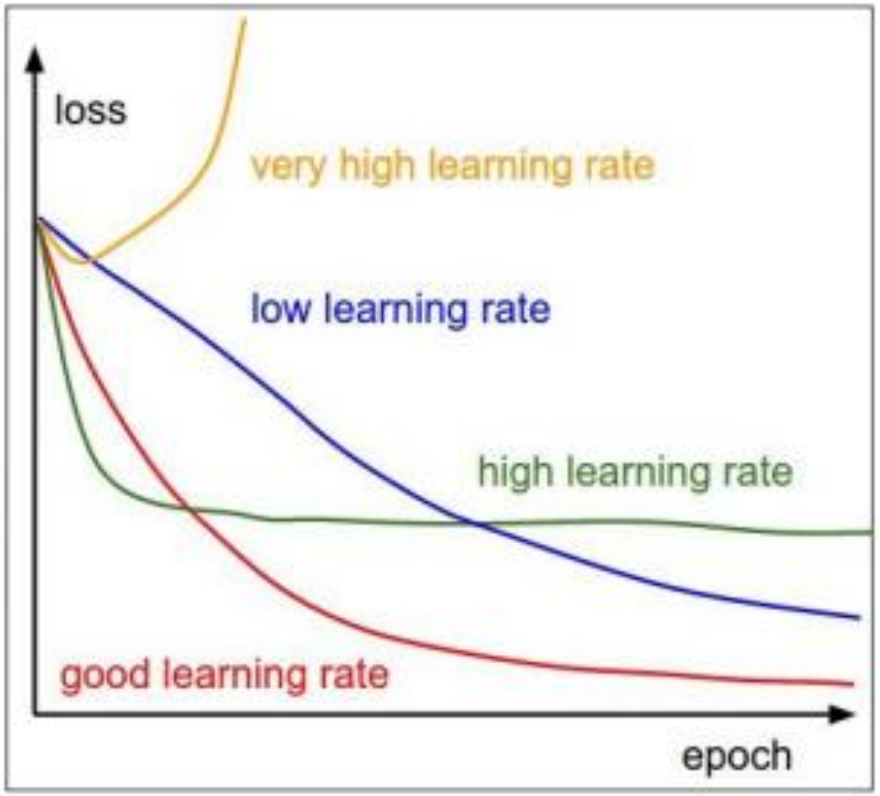
```
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.classifier.parameters(), lr=0.001)
```



HyperParameter

로스, 옵티마이저 정의

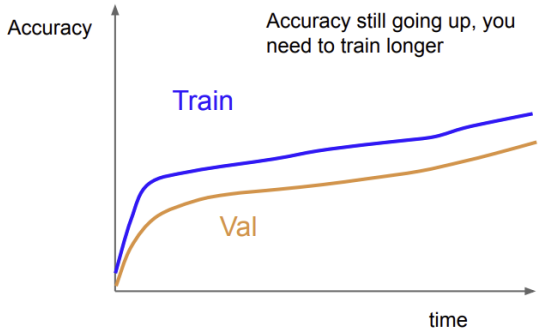
```
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.classifier.parameters(), lr=0.001)
```

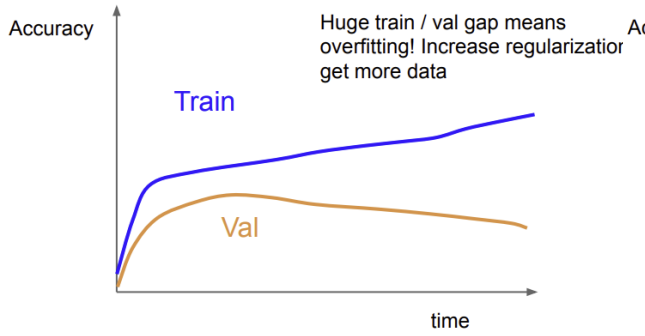


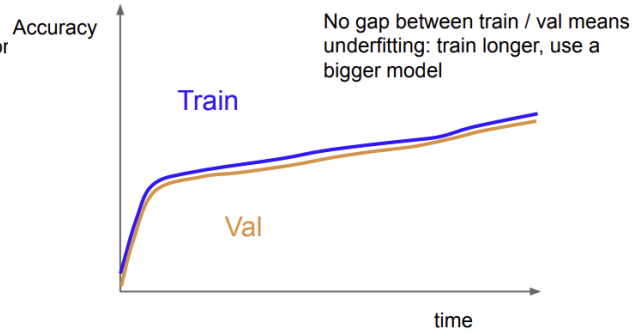
HyperParameter

로스, 옵티마이저 정의

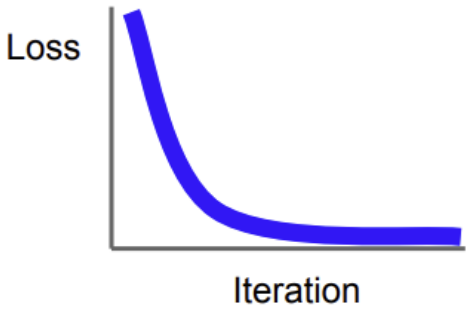
```
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.classifier.parameters(), lr=0.001)
```

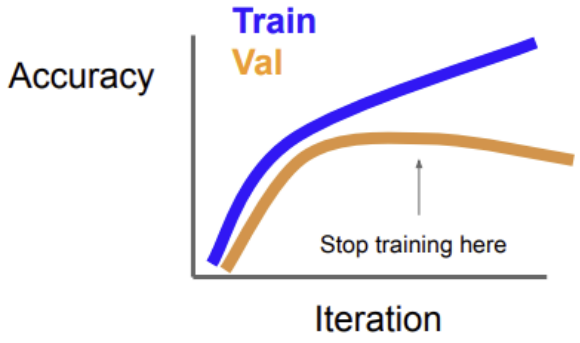






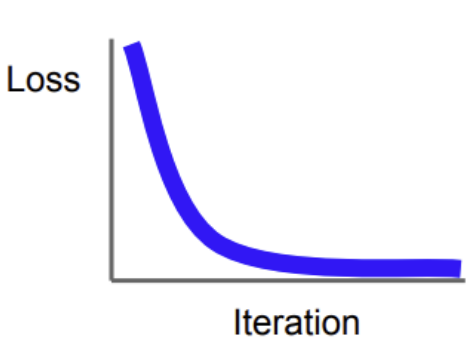
Early Stopping

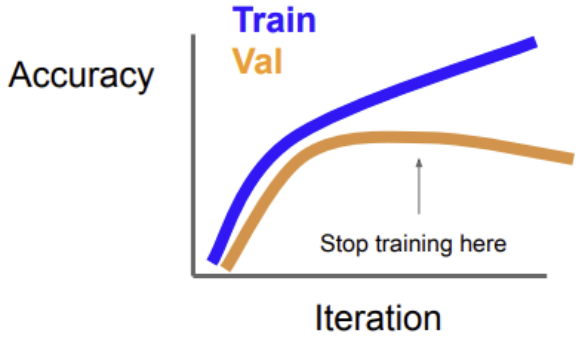




HyperParameter

Early Stopping



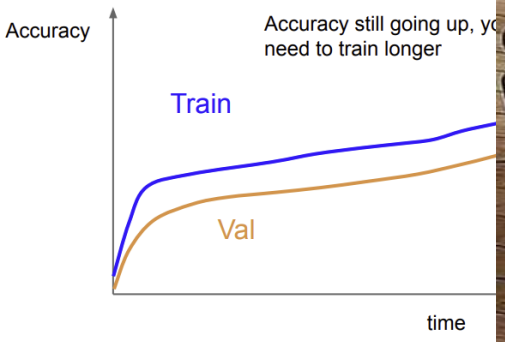


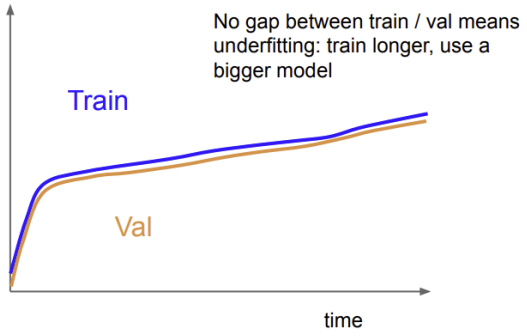
https://www.instagram.com/p/CZGjf_xr_Mh/

HyperParameter

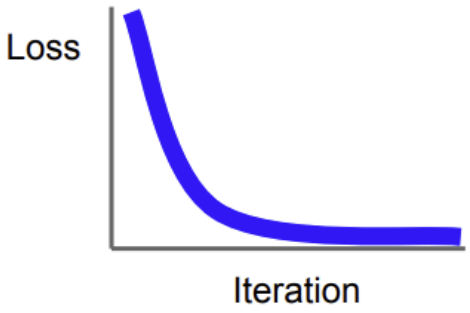
로스, 옵티마이

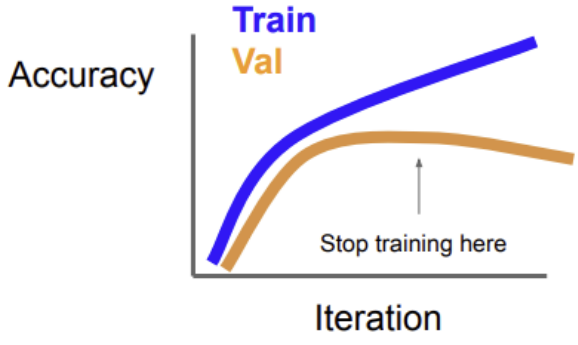
```
criterion = n
optimizer = o                                    ameters(), lr=0.001)
```





Earl

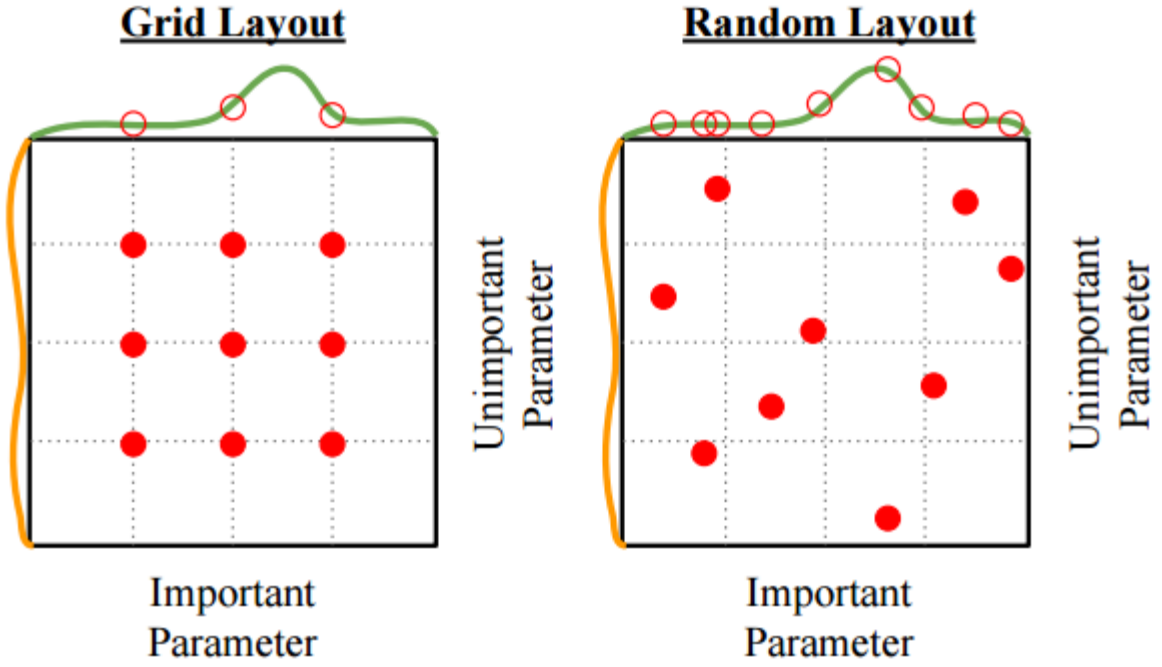




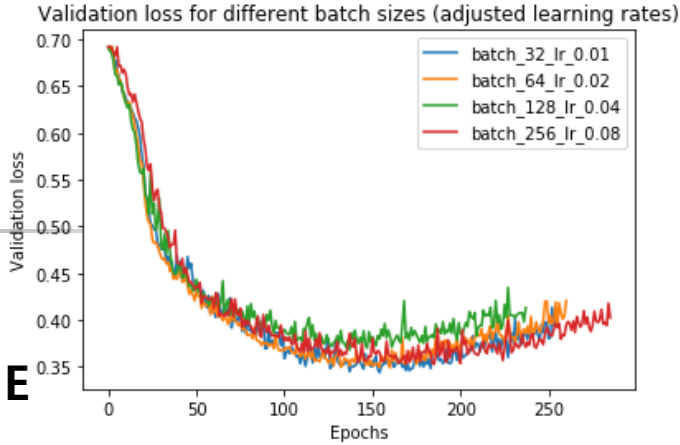
HyperParameter

로스, 옵티마이저 정의

```
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.classifier.parameters(), lr=0.001)
```



HyperParameter tuning



- **HYPERPARAMETER TUNING WITH RAY TUNE**

```
net = Net(config["l1"], config["l2"])

device = "cpu"
if torch.cuda.is_available():
    device = "cuda:0"
    if torch.cuda.device_count() > 1:
        net = nn.DataParallel(net)
net.to(device)

criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=config["lr"], momentum=0.9)

if checkpoint_dir:
    model_state, optimizer_state = torch.load(
        os.path.join(checkpoint_dir, "checkpoint"))
    net.load_state_dict(model_state)
    optimizer.load_state_dict(optimizer_state)

trainset, testset = load_data(data_dir)

test_abs = int(len(trainset) * 0.8)
train_subset, val_subset = random_split(
    trainset, [test_abs, len(trainset) - test_abs])

trainloader = torch.utils.data.DataLoader(
    train_subset,
    batch_size=int(config["batch_size"]),
    shuffle=True,
    num_workers=8)
```

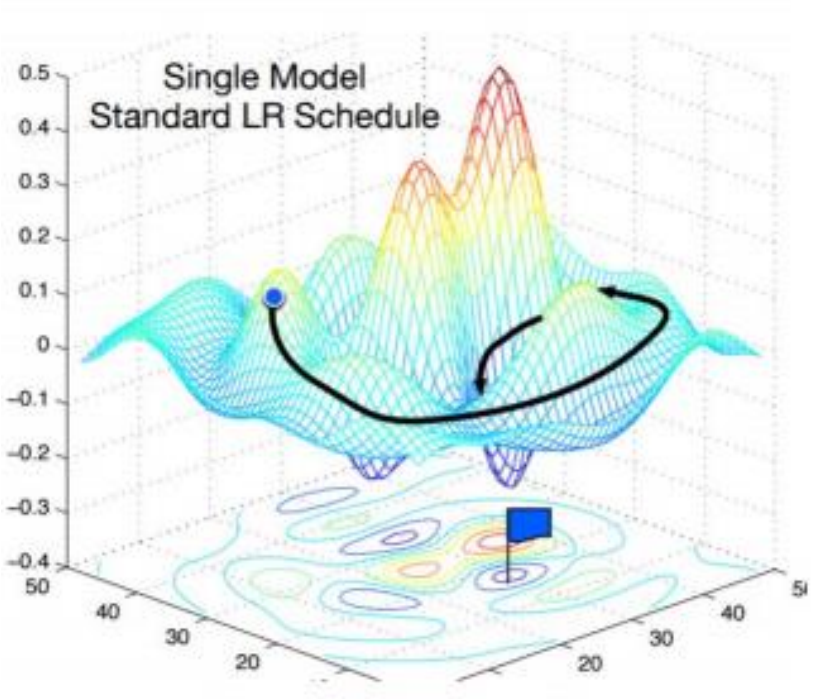
```
Number of trials: 10 (10 TERMINATED)
+-----+------+------+-------------+--------------+---------+------------+--------------------
+
| ... |   l1 |   l2 |          lr |   batch_size |    loss |   accuracy | training_iteration
|
|-----+------+------+-------------+--------------+---------+------------+--------------------
|
| ... |   64 |    4 | 0.00011629  |            2 | 1.87273 |     0.244  |                  2
|
| ... |   32 |   64 | 0.000339763 |            8 | 1.23603 |     0.567  |                  8
|
| ... |    8 |   16 | 0.00276249  |           16 | 1.1815  |     0.5836 |                 10
|
| ... |    4 |   64 | 0.000648721 |            4 | 1.31131 |     0.5224 |                  8
|
| ... |   32 |   16 | 0.000340753 |            8 | 1.26454 |     0.5444 |                  8
|
| ... |    8 |    4 | 0.000699775 |            8 | 1.99594 |     0.1983 |                  2
|
| ... |  256 |    8 | 0.0839654   |           16 | 2.3119  |     0.0993 |                  1
|
| ... |   16 |  128 | 0.0758154   |           16 | 2.33575 |     0.1327 |                  1
|
| ... |   16 |    8 | 0.0763312   |           16 | 2.31129 |     0.1042 |                  4
|
| ... |  128 |   16 | 0.000124903 |            4 | 2.26917 |     0.1945 |                  1
|
+-----+------+------+-------------+--------------+---------+------------+--------------------
+


Best trial config: {'l1': 8, 'l2': 16, 'lr': 0.00276249, 'batch_size': 16, 'data_dir': '...'}
Best trial final validation loss: 1.181501
Best trial final validation accuracy: 0.5836
Best trial test set accuracy: 0.5806
```


https://pytorch.org/tutorials/beginner/hyperparameter_tuning_tutorial.html
https://medium.com/geekculture/how-does-batch-size-impact-your-model-learning-2dd34d9fb1fa

Learning Rate Scheduler

로스, 옵티마이저 정의

```
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.classifier.parameters(), lr=0.001)
```

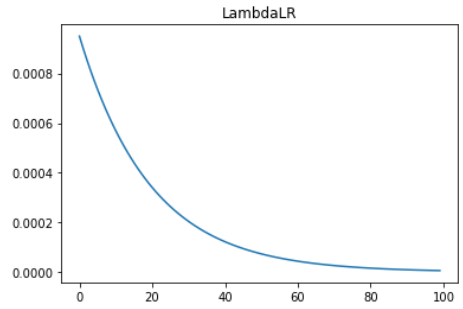


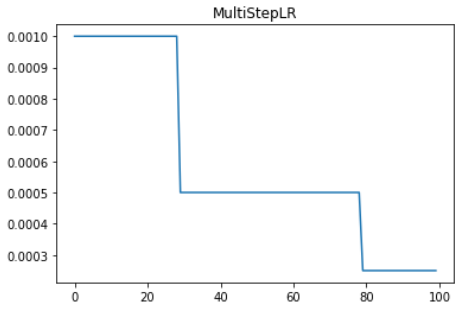
Learning Rate Scheduler

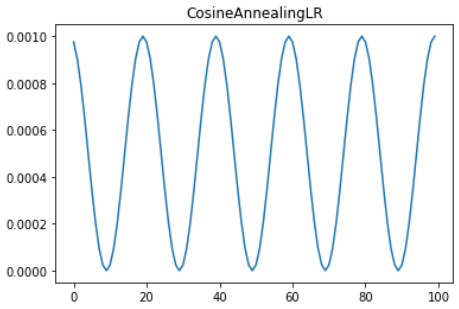
로스, 옵티마이저 정의

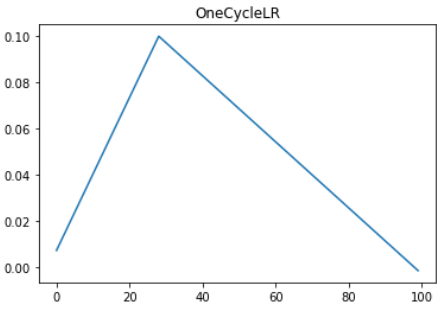
```
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.classifier.parameters(), lr=0.001)
```

```
optimizer = torch.optim.SGD(model.parameters(), lr=0.001)
scheduler = optim.lr_scheduler.LambdaLR(optimizer=optimizer,
                                        lr_lambda=lambda epoch: 0.95 ** epoch)
```









```
for epoch in range(num_epochs):
    for phase in ['train', 'validation']:
        if phase == 'train':
            model.train()
        else:
            model.eval()

        running_loss = 0.0
        running_corrects = 0

        for inputs, labels in dataloaders[phase]:
            inputs = inputs.to(device)
            labels = labels.to(device)
            one_hot = F.one_hot(labels, num_classes=149).float()

            outputs = model(inputs)
            loss = criterion(outputs, one_hot)

            if phase == 'train':
                optimizer.zero_grad()
                loss.backward()
                optimizer.step()

            _, preds = torch.max(outputs, 1)
            running_loss += loss.item() * inputs.size(0)
            running_corrects += torch.sum(preds == labels.data)

        epoch_loss = running_loss / len(image_datasets[phase])
        epoch_acc = running_corrects.double() / len(image_datasets[phase])

        print('{}, Epoch {}/{}, loss: {:.4f}, acc: {:.4f}'.format(
            phase, epoch+1, num_epochs, epoch_loss, epoch_acc))
```

batch 단위 입출력 데이터 불러오기

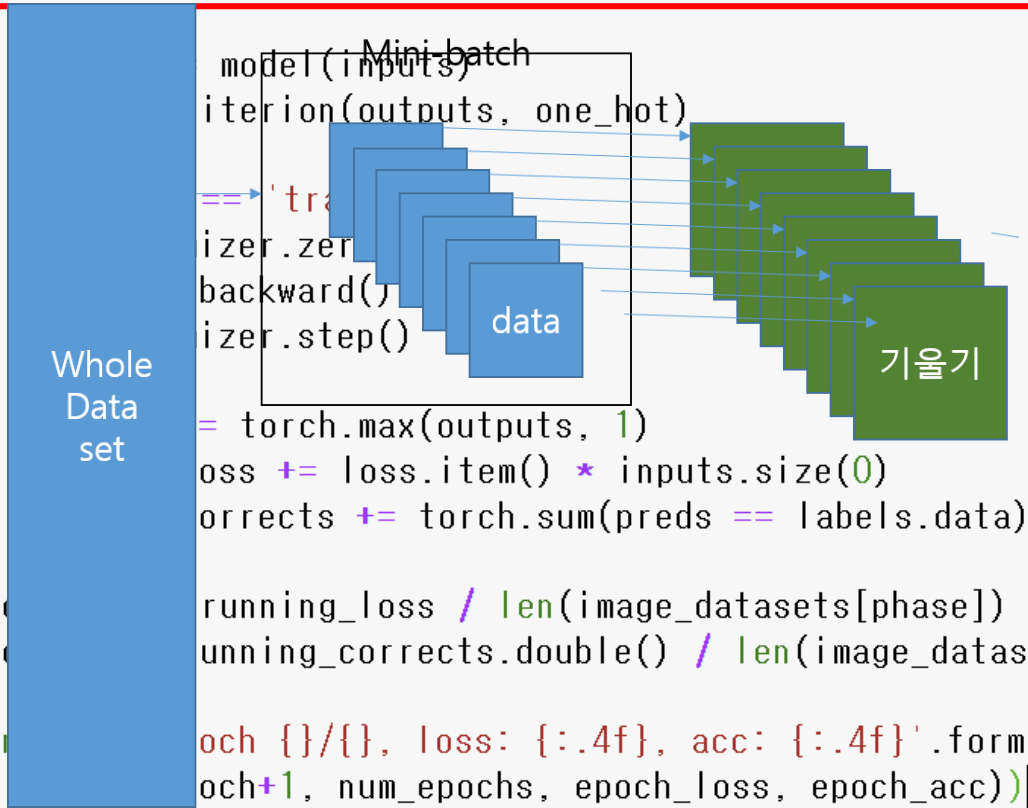
EfficientNet에 입력이미지 feeding

모델의 predict와 true label 비교

loss를 줄이는 방향으로 backpropagation

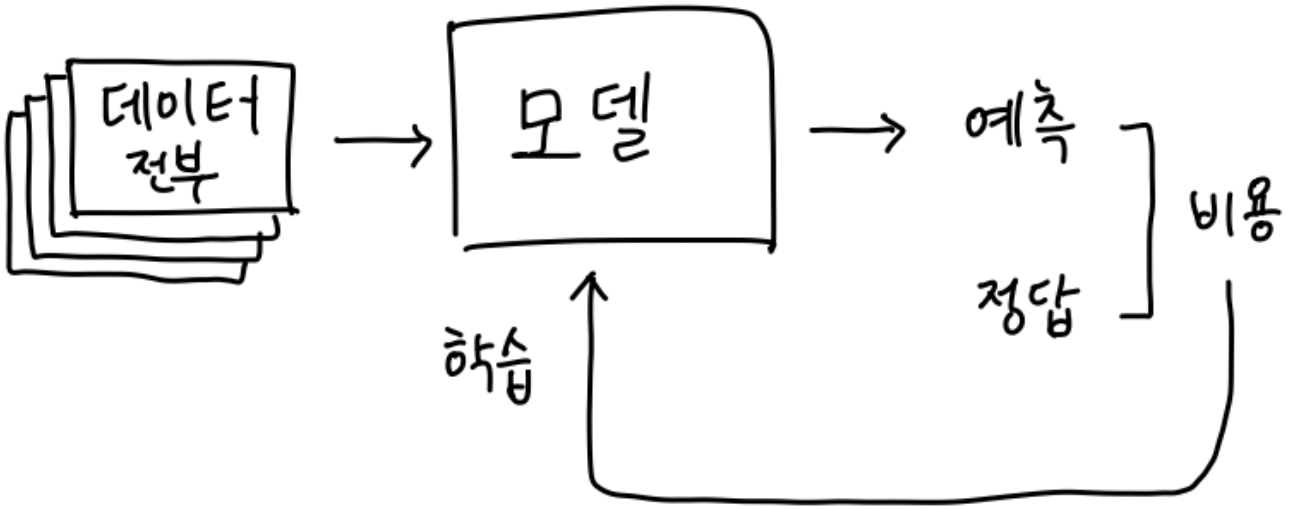
```
for epoch in range(num_epochs):
    for phase in ['train', 'validation']:
        if phase == 'train':
            model.train()
        else:
            model.eval()

        running_loss = 0.0
        running_corrects = 0

        for inputs, labels in dataloaders[phase]:
            inputs = inputs.to(device)
            labels = labels.to(device)
            one_hot = F.one_hot(labels, num_classes=149).float()
```

batch 단위 입출력 데이터 불러오기

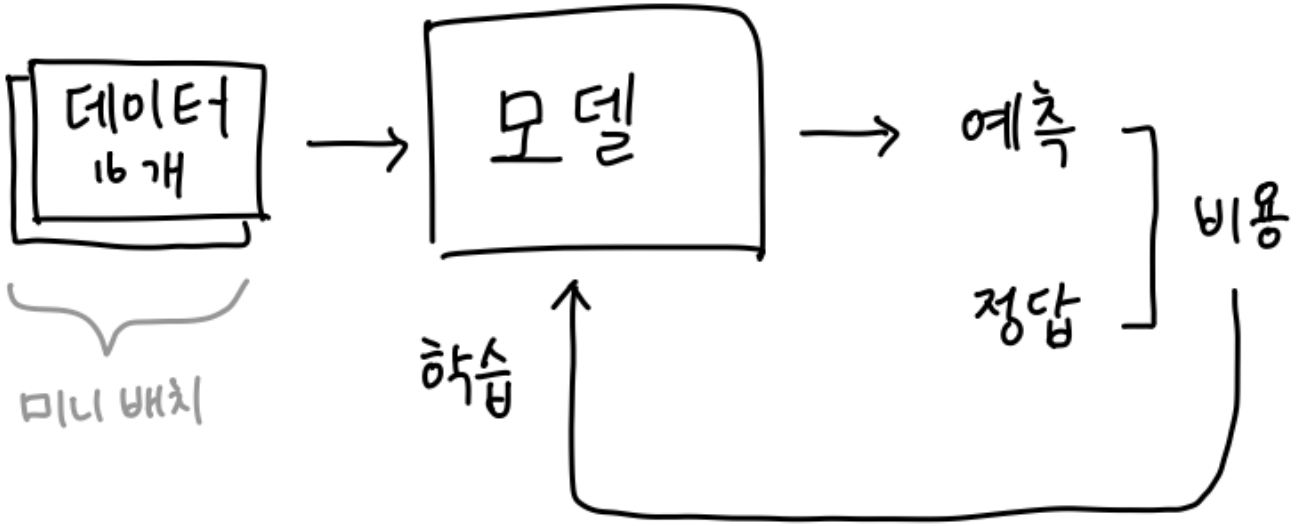



```
            model(inputs)
            iterion(outputs, one_hot)

        ==='tra
            izer.zer
            backward()
            izer.step()

            = torch.max(outputs, 1)
            oss += loss.item() * inputs.size(0)
            orrects += torch.sum(preds == labels.data)

    epoc        running_loss / len(image_datasets[phase])
    epoc        unning_corrects.double() / len(image_datasets[phase])

    prin        och {}/{}, loss: {:.4f}, acc: {:.4f}'.format(
                och+1, num_epochs, epoch_loss, epoch_acc))
```

for




```
        loss = criterion(outputs, one_hot)
```



Tr

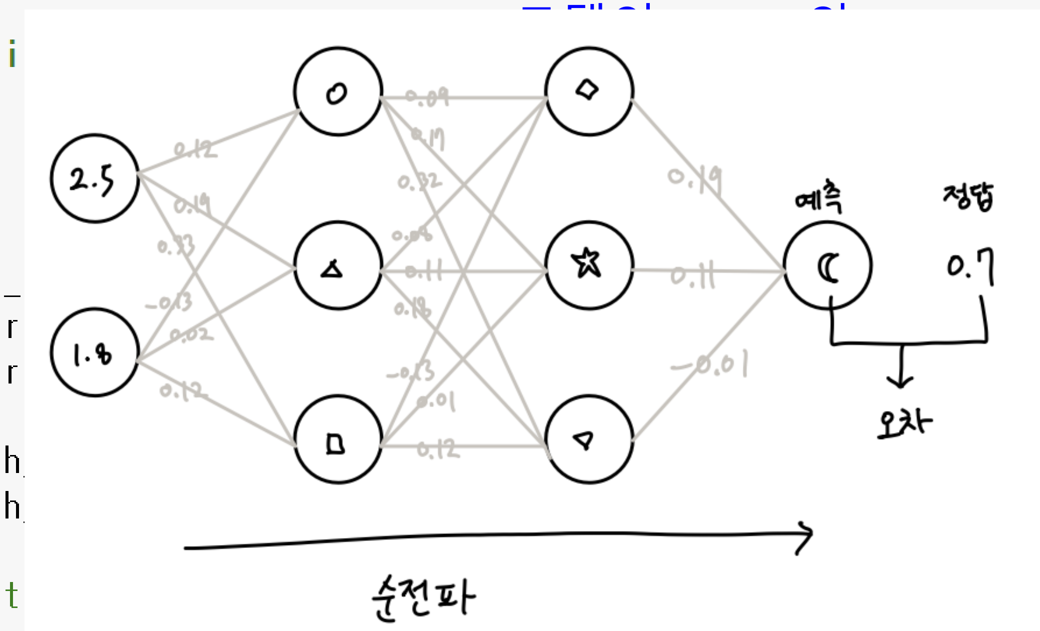
```
for epoch in range(num_epochs):
    for phase in ['train', 'validation']:
        if phase == 'train':
            model.train()
        else:
            model.eval()

        running_loss = 0.0
        running_corrects = 0

        for inputs, labels in dataloaders[phase]:
            inputs = inputs.to(device)
            labels = labels.to(device)
            one_hot = F.one_hot(labels, num_classes=149).float()

            outputs = model(inputs)
            loss = criterion(outputs, one_hot)

            i
            _
            r
            r

        epoch
        epoch                                              s[phase])

        print                                              (
            phase, epoch+1, num_epochs, epoch_loss, epoch_acc))
```

batch 단위 입출력 데이터 불러오기

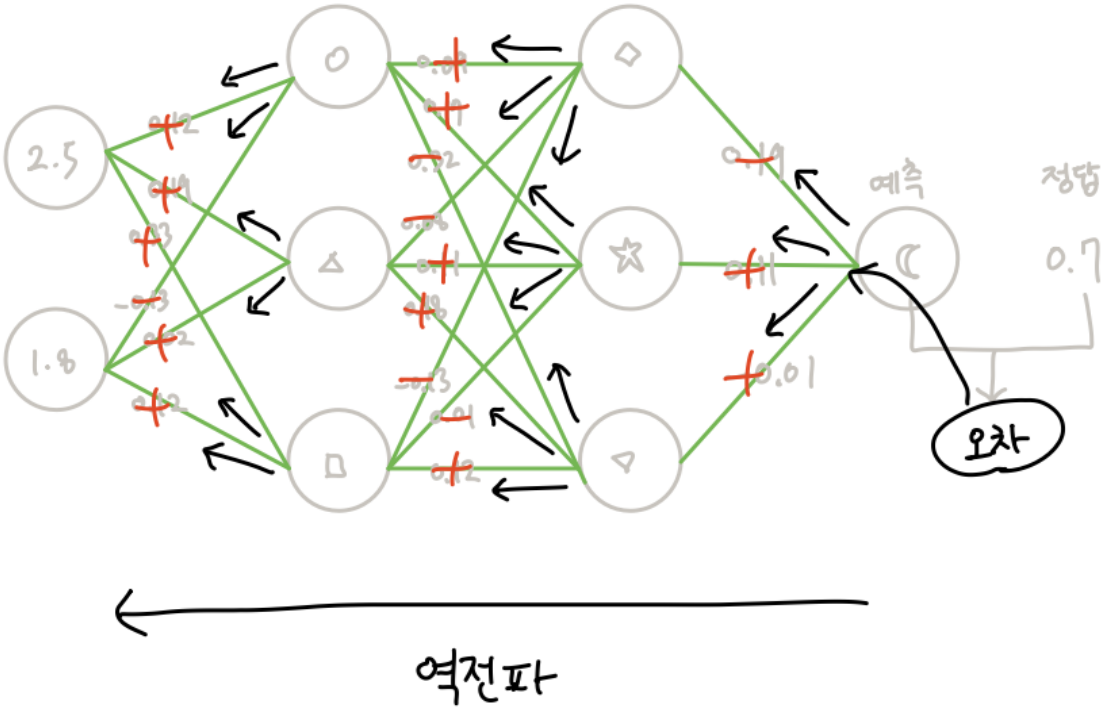
EfficientNet에 입력이미지 feeding

비교

ckpropagation



https://www.philgineer.com/2021/09/27-5.html



거오기

eding

모델의 predict와 true label 비교

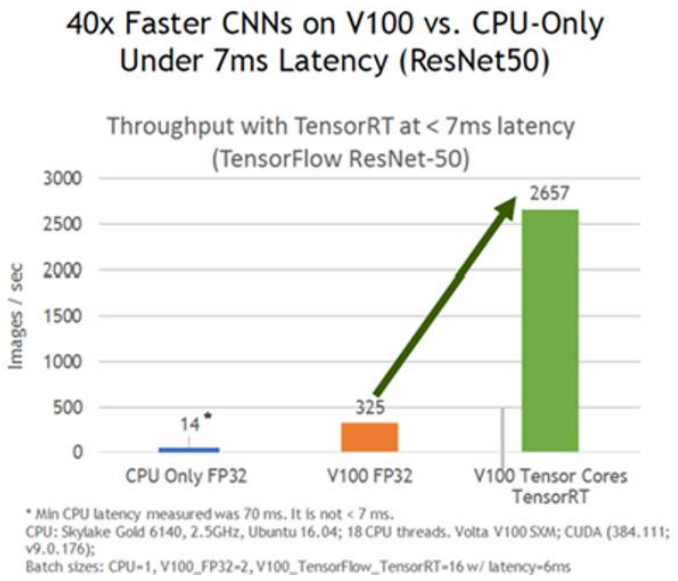
loss를 줄이는 방향으로 backpropagation

```
for ep
    fo
        if phase == 'train':
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

        _, preds = torch.max(outputs, 1)
        running_loss += loss.item() * inputs.size(0)
        running_corrects += torch.sum(preds == labels.data)

    epoch_loss = running_loss / len(image_datasets[phase])
    epoch_acc = running_corrects.double() / len(image_datasets[phase])

    print('{}, Epoch {}/{}, loss: {:.4f}, acc: {:.4f}'.format(
        phase, epoch+1, num_epochs, epoch_loss, epoch_acc))
```

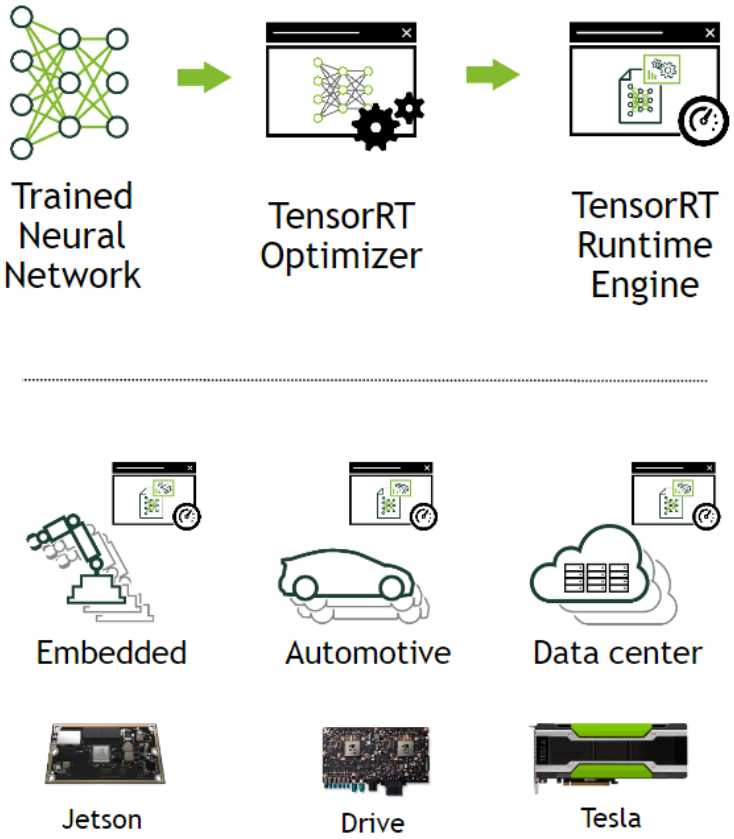

https://www.philgineer.com/2021/09/27-5.html

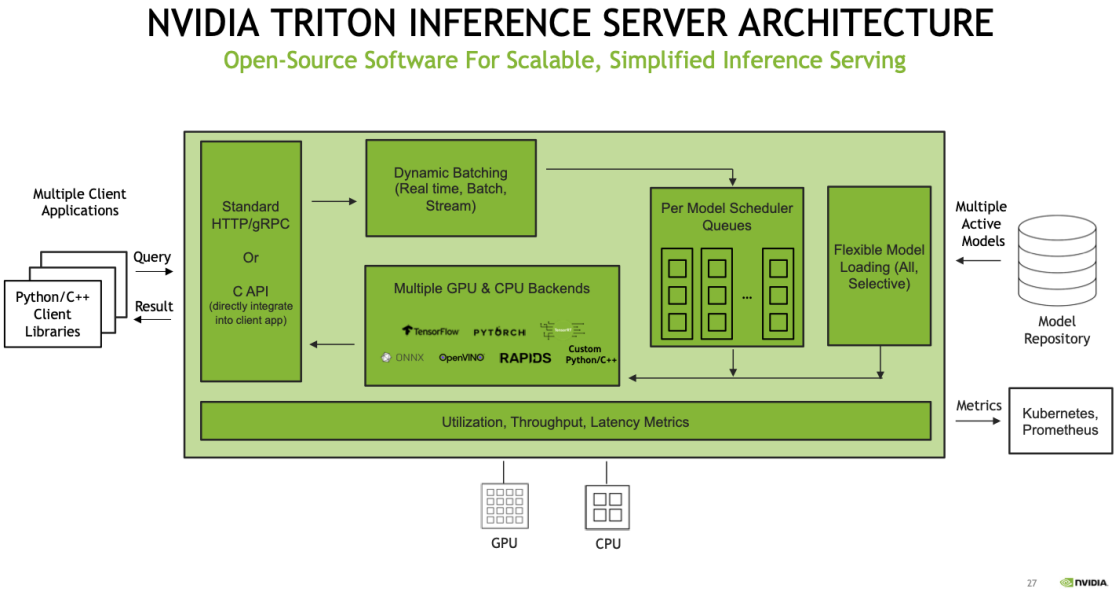
TensorRT

- **TensorRT**
  - 학습된 딥러닝 모델을 최적화
  - NVIDIA GPU 상에서의 추론 속도를 수배 ~ 수십배 까지 향상
  - 딥러닝 서비스를 개선하는데 도움을 줄 수 있는 모델 최적화 엔진
  - Caffe, Pytorch, TensorFlow 모델을 NVIDIA GPU 플랫폼(TESLA T4 , JETSON TX2, TESLA V100)에 아름답게 싣는 것
  - NVIDIA GPU 연산에 적합한 최적화 기법들을 이용하여 모델을 최적화하는 Optimizer 와 다양한 GPU에서 모델 연산을 수행하는 Runtime Engine 을 포함
    - 양자화 및 정밀도 캘리브레이션
    - 그래프 최적화
    - 커널 자동 튜닝
    - 동적 텐서 메모리 및 멀티 스트림 실행



TensorRT





https://dataintegration.info/deploy-fast-and-scalable-ai-with-nvidia-triton-inference-server-in-amazon-

Collision avoidance - TensorRT

```
import torch
import torchvision

model = torchvision.models.resnet18(pretrained=False)
model.fc = torch.nn.Linear(512, 2)
model = model.cuda().eval().half()
```

Next, load the trained weights from the best_model_resnet18.pth file that you uploaded

```
model.load_state_dict(torch.load('best_model_resnet18.pth'))
```

Currently, the model weights are located on the CPU memory execute the code below to transfer to the GPU device.

```
device = torch.device('cuda')
```

```
from torch2trt import torch2trt

data = torch.zeros((1, 3, 224, 224)).cuda().half()

model_trt = torch2trt(model, [data], fp16_mode=True)
```

Save the optimized model using the cell below

```
torch.save(model_trt.state_dict(), 'best_model_trt.pth')
```

CNN

- **Quiz!!**
- **True for success of deep models??**
  - **Better design of the neural networks**
  - **Large scale training dataset**
  - **Available computing power**
  - **All of the above**

Multi-class classification & XAI(GradCAM) 실습

- **https://keep-steady.tistory.com/35**
- **https://github.com/airobotlab/advanced_cnn/05_03_advanced_cnn_CAM_AA_220412_final.ipynb**

XAI

설명 가능한 인공지능(Expainable AI)

XAI

- **In the future**

https://www.scoopnest.com/user/NewYorker/684741246443778048-a-cartoon-by-paul-noth-find-more-cartoons-from- this-week39s-issue-here

XAI

- **Interpretability**
  - **The degree to which a human can understand the cause of model's decision**
  - **DL have achieved high performance in various application**
  - **Challenges: Black box model**
    - **Despite the high performance, we still cannot fully trust DL**
    - **Limited interpretability is a major challenge for practical use of current DL approaches**

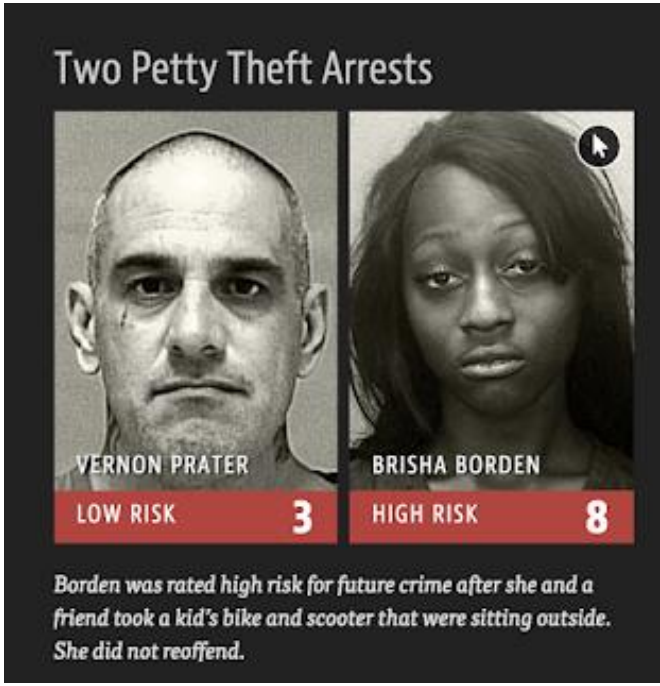
https://www.scoopnest.com/user/NewYorker/684741246443778048-a-cartoon-by-paul-noth-find-more-cartoons-from- this-week39s-issue-here

XAI

- Biad in AI Model
  - COMPAS (Correctional Offender Management Profiling for Alternative Sanctions)
  - 범죄를 저지른 후 2년내 다시 범죄를 저지를 위험도

“
- 위험인물로 지목되었지만 범죄를 저지르지 않은
  - 백인 비율: 23.5%
  - 흑인 비율: 44.9%

“
- 덜 위험한 인물로 지목되었지만 범죄를 저지른
  - 백인 비율: 47.7%
  - 흑인 비율: 28.0%



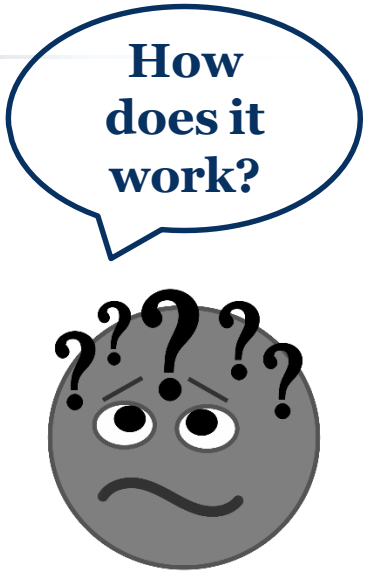
https://medium.com/@angellin880221/machine-bias-3cd8052d1356
http://news.karts.ac.kr/?p=8543

XAI

- **ML Model Categories**
  - **White Box Models**
    - Easy to interpret
    - Sometimes can not learn the patterns in the data well (low accuracy) due to their simplicity
  - **Black Box Models**
    - Hard (or impossible) to interpret
    - Mostly more powerful and effective compared to white box models
    - e.g. neural networks

XAI

- 설명 가능한 AI(eXplainable AI)
- AI 스스로 사람이 이해할 수 있도록 설명
- 현재 Deeplearning의 약점
  - Why?? - BlackBox



Black Box AI

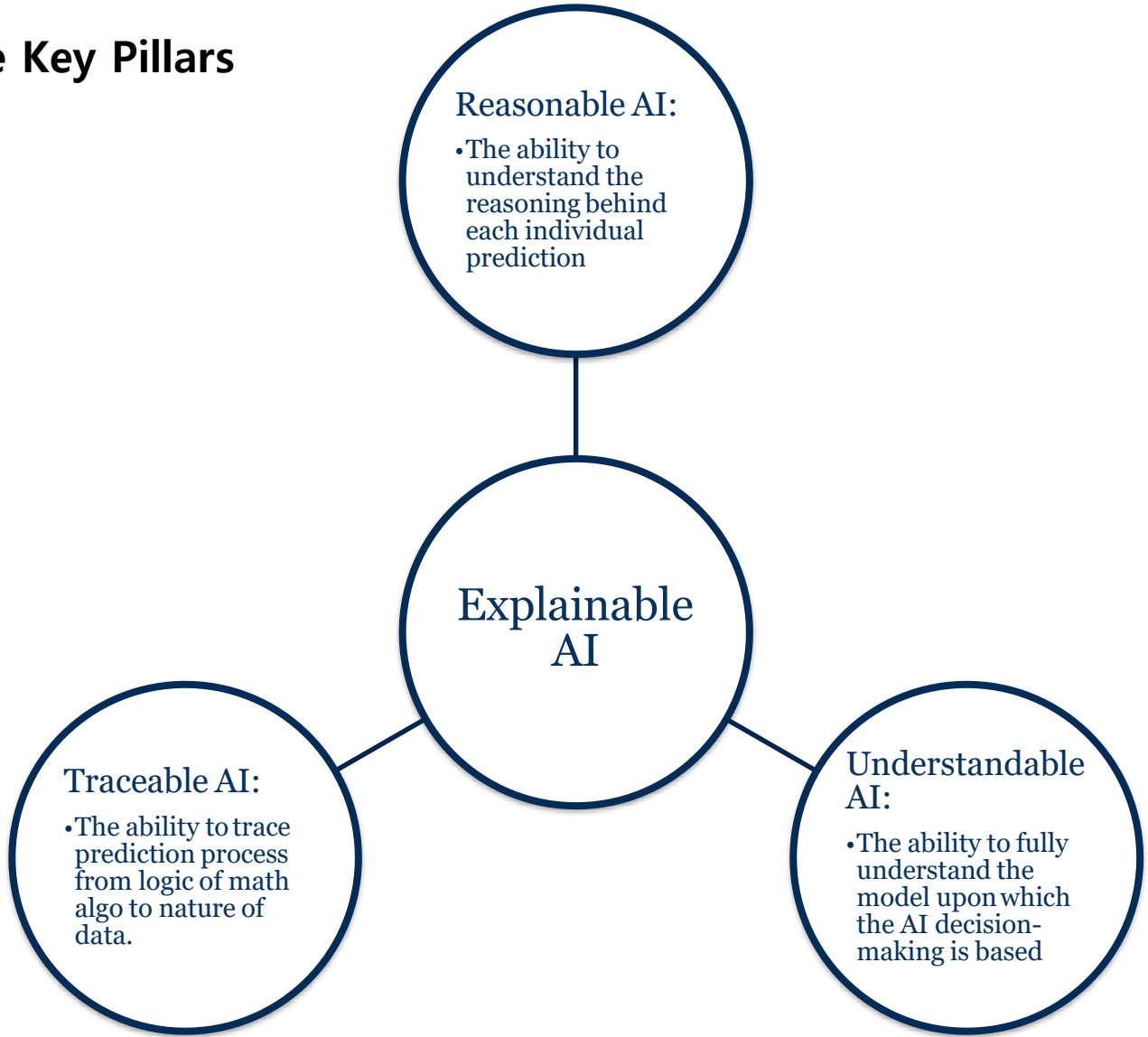
- Why did the AI system do that?
- Why didn't the AI system do something else?
- When did the AI system succeed?
- When did the AI system fail?
- When does the AI system give enough confidence in the decision that you can trust it?
- How can the AI system correct an error?

XAI

- 설명 가능한 AI(eXplainable AI)
- AI 스스로 사람이 이해할 수 있도록 설명
- 현재 Deeplearning의 약점
  - Why??
  - Adversarial Attack

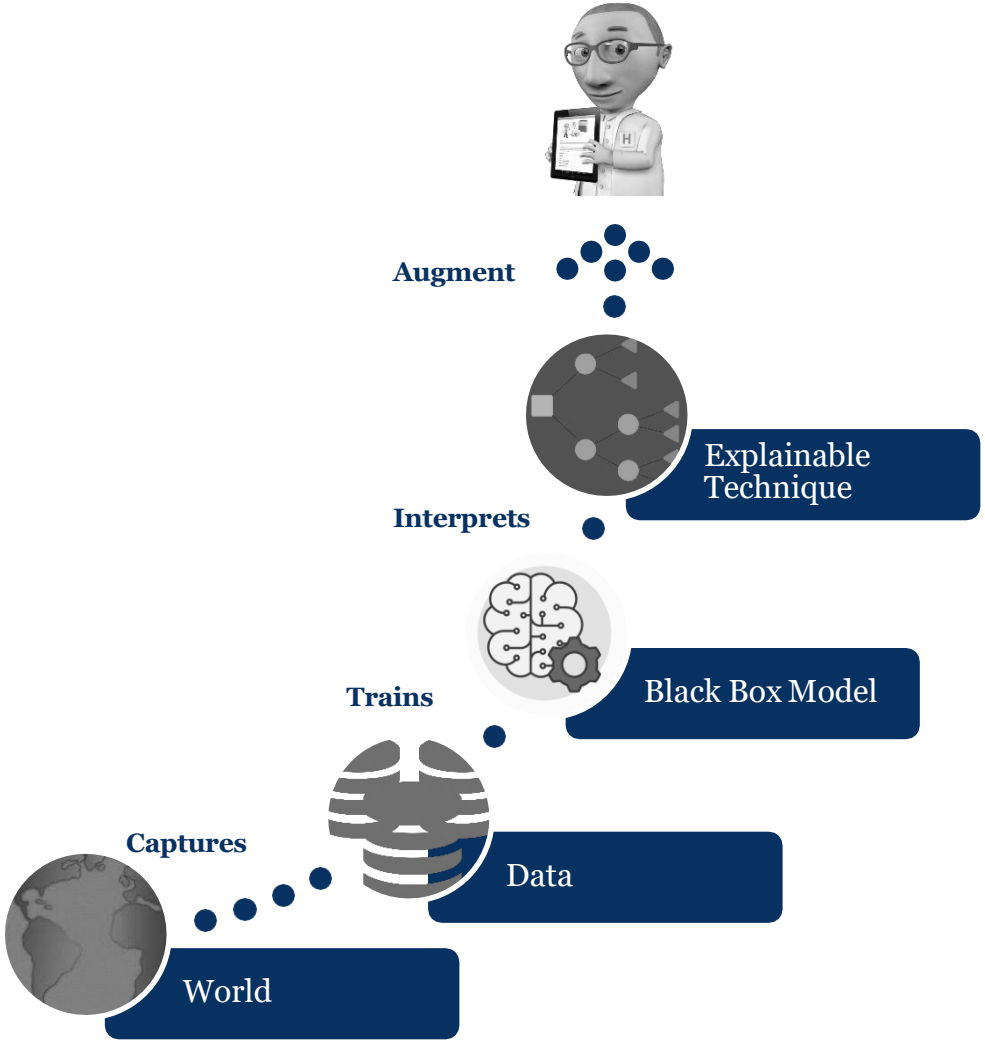
XAI

- **Three Key Pillars**



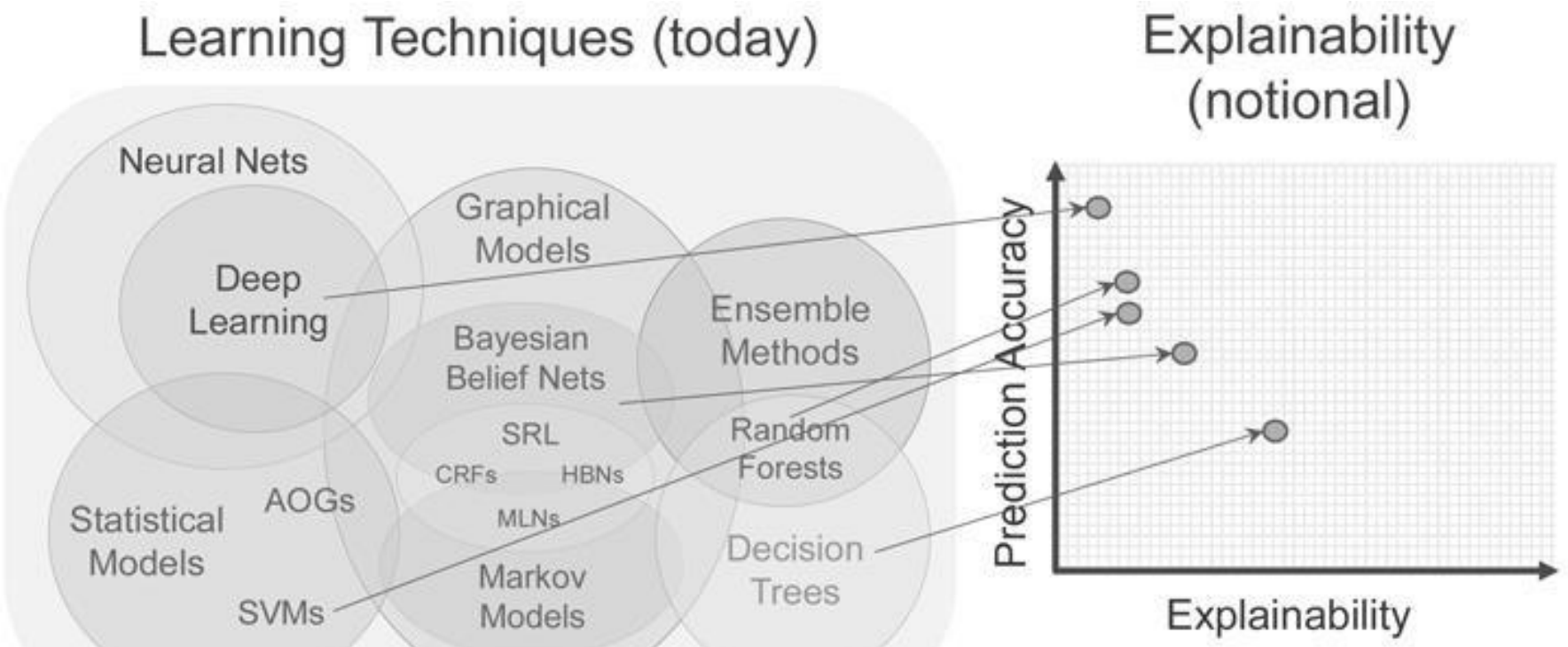
XAI

- **Full Circle of Life**



XAI

- **Current State of AI Models w.r.t. XAI vs Performance**



- Regression – Multinomial Equation can be messy
- Random Forrest – Multiple Tree and their Data Set and Voting
- SVM – Kernel and Data Partition effect on Feature
- K Mean – Nature of Centroid don't describe cluster well.
- NN – Hidden Nodes and their way of creating features

XAI

- **Accuracy VS Interpretability**

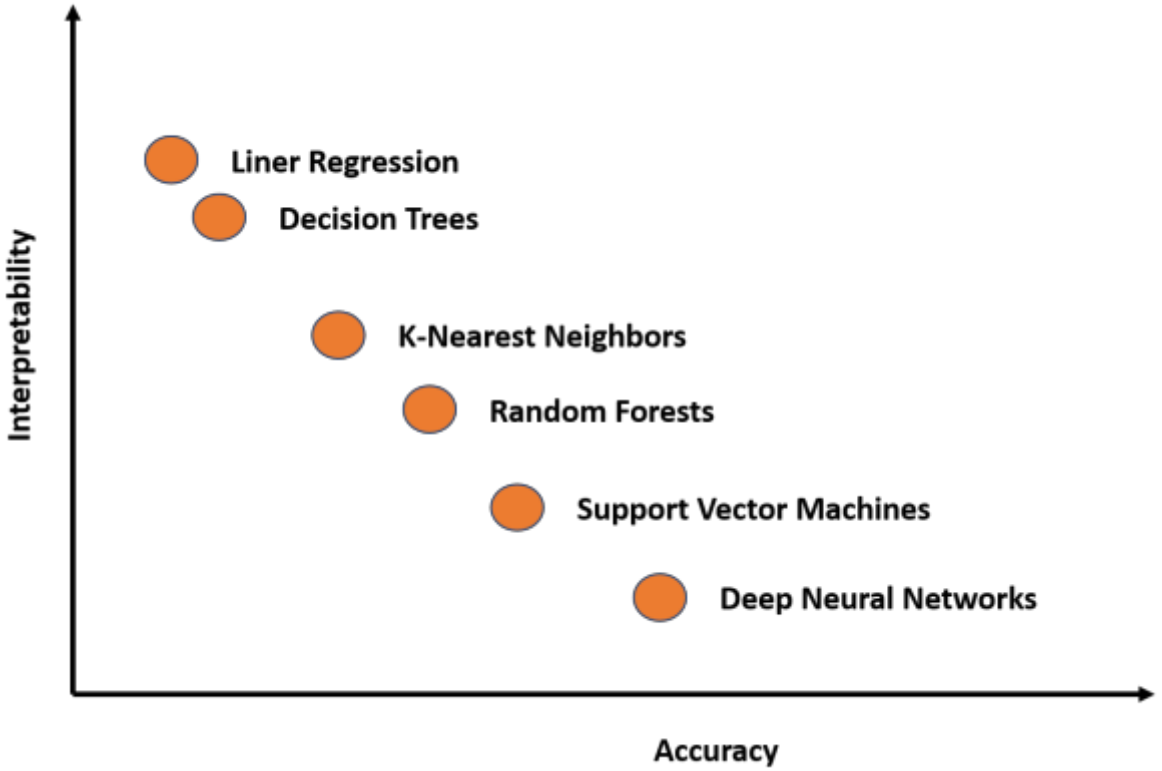


Image source: https://towardsdatascience.com/interpretability-vs-accuracy-the-friction-that-defines-deep-learning- dae16c84db5c

XAI

- **Some XAI Use Cases**

Medicine

Finance

Legal

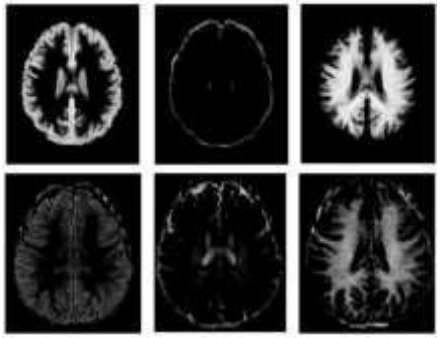
Autonomous Cars

Military

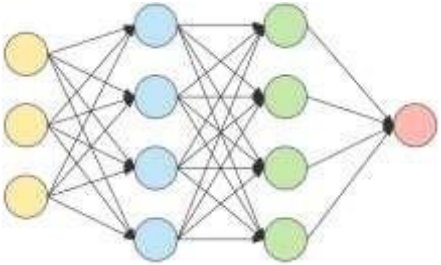
Image source: https://towardsdatascience.com/interpretability-vs-accuracy-the-friction-that-defines-deep-learning- dae16c84db5c

XAI

- **Medical Application**



Brain MRI data



Complex ML model

Report:

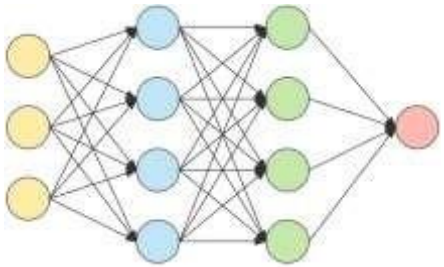
Patient is diagnosed with **Epilepsy** with **%85 confidence**.

Image sources: https://msaffarm.github.io/projects/ml- mri/Sahebzamani.Saffar.HSZ.Jan.2019.pdf https://artemisinccouk.wordpress.com/author/torak289/

XAI

- **Finance Application**

Financial and Demographic Data



Complex ML model

Image sources: https://msaffarm.github.io/projects/ml- mri/Sahebzamani.Saffar.HSZ.Jan.2019.pdf https://artemisinccouk.wordpress.com/author/torak289/

XAI

- General Data Protection Regulation (GDPR)
- GDPR imposes companies to provide exaplanations of their ML models to their customers
- Legal implication of wrong diagnosis in medical applications can be tough
- How can the doctor trust the ML model predictions?

Image sources: https://msaffarm.github.io/projects/ml- mri/Sahebzamani.Saffar.HSZ.Jan.2019.pdf https://artemisinccouk.wordpress.com/author/torak289/

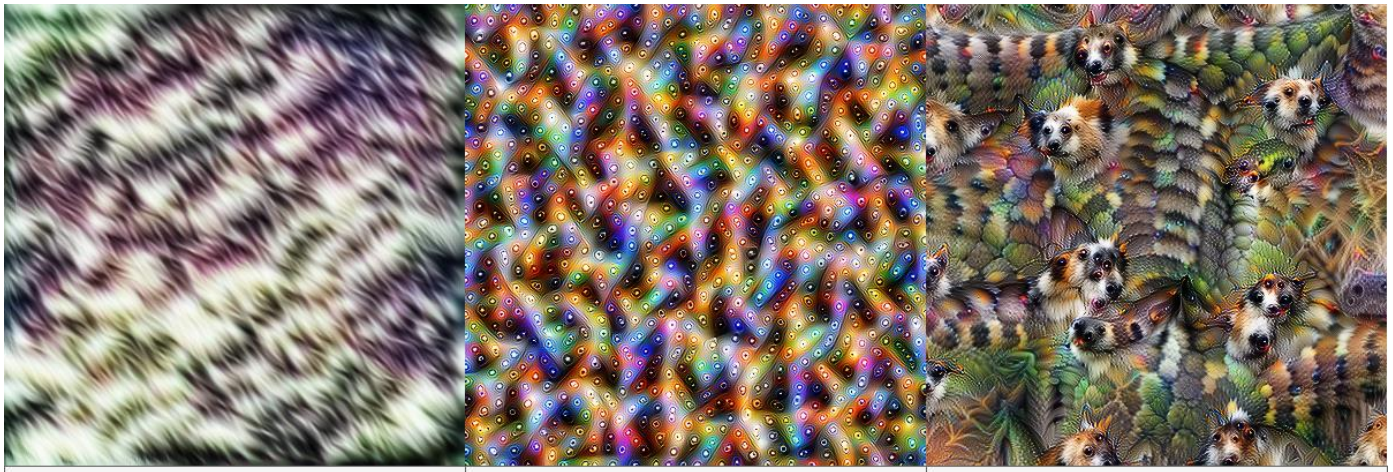
XAI

- **TRUST**
  - Understand decision making process
  - Make sure model is looking at the right features
- **BIAS & FAIRNESS**
  - features are taken into account
  - Detect biased patterns in data
- **EXPLAINABILITY**
  - Explain the decision making process

Image sources: https://msaffarm.github.io/projects/ml- mri/Sahebzamani.Saffar.HSZ.Jan.2019.pdf https://artemisinccouk.wordpress.com/author/torak289/

XAI

- **1) Feature Understanding**
  - DL 내부(neuron)에 어떤 특징이 학습되었지?

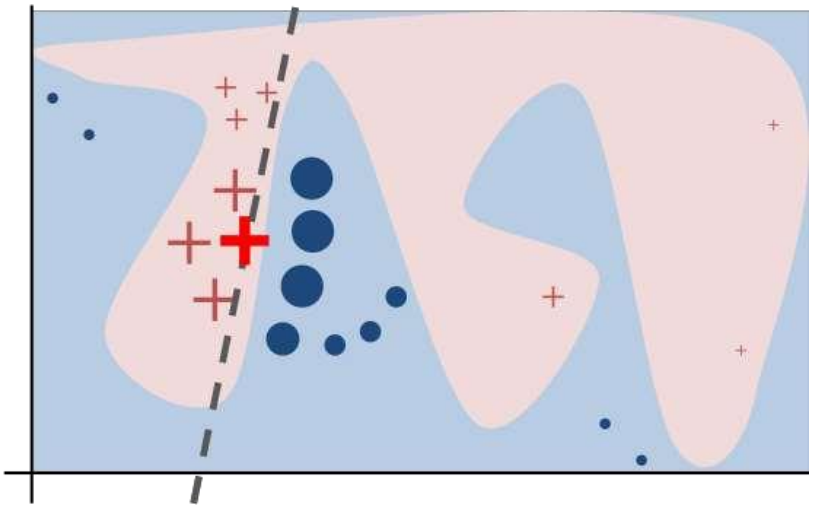


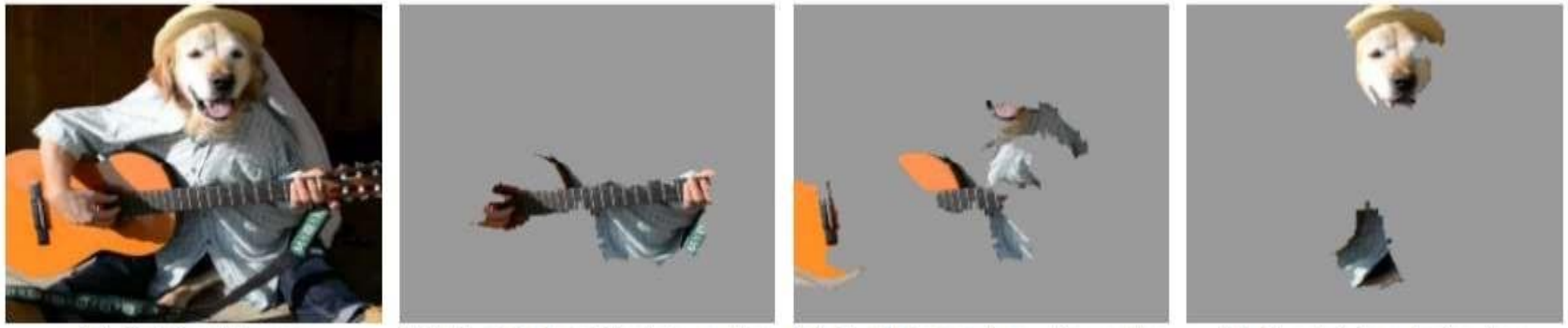
- **2) Feature Attribution**
  - 모델 예측의 근거 시각적 설명 방법
  - 모델의 예측에 어떤 특징(픽셀)이 중요하게 사용되었지?

https://towardsdatascience.com/visualizing-change-in-neural-networks-eea86529a9f3

XAI

- **Locally Interpretable Model-agnostic Explanations (LIME)**





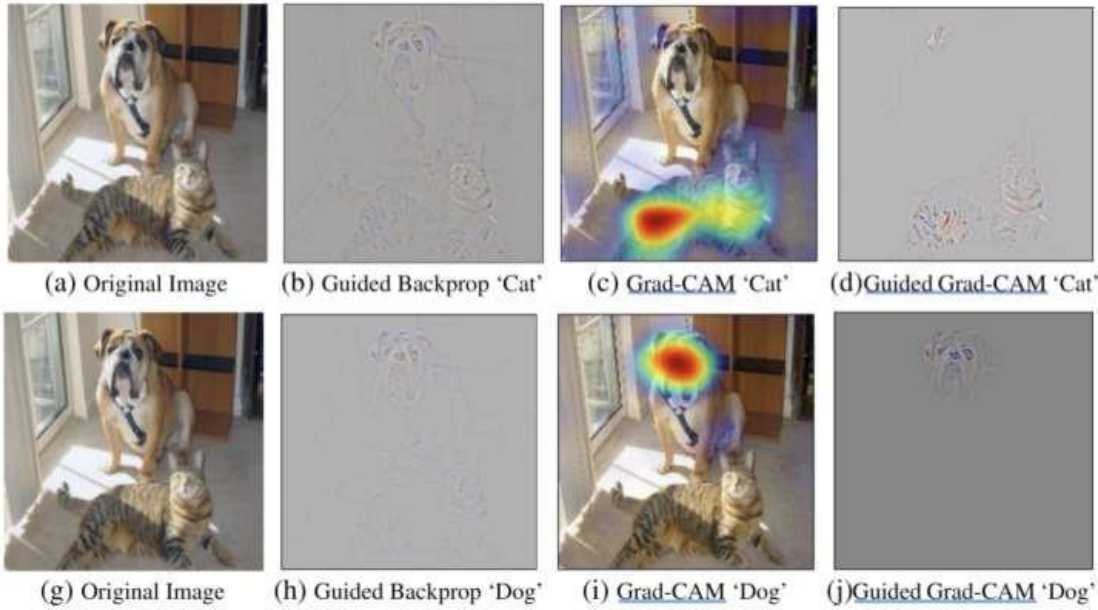
(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Class Activation Map (CAM) - Visual Explanation Method

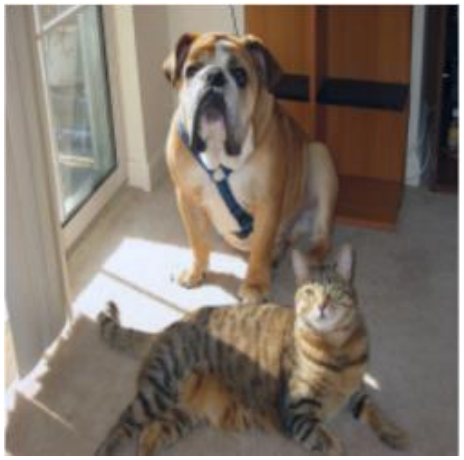
- Concern: Model Transparency & Interpretability
  - Despite unprecedented breakthroughs of CNN in a variety of computer vision tasks, their lack of decomposability into individually intuitive components makes them hard to interpret
- Purpose:
  - Visualizing CNNs
    - visualized CNN predictions by highlighting 'important' pixels
  - Help Users to Build Trust to AI
    - we must build 'transparent' models that have the ability to explain why they predict what they predict.

Class Activation Map (CAM) - Visual Explanation Method

- What makes a good visual explanation?
  - Class Discriminative – localize the category in the image
    - Class Activation Mapping (CAM)
    - Gradient-weighted Class Activation Mapping (Grad-CAM)
  - High-Resolution – capture fine-grained detail
    - Guided Back propagation
    - Deconvolution
  - Both
    - Guided Grad-CAM



Class Activation Map (CAM) - Visual Explanation Method



Grad-CAM for "Dog"

Grad-CAM for "Cat"

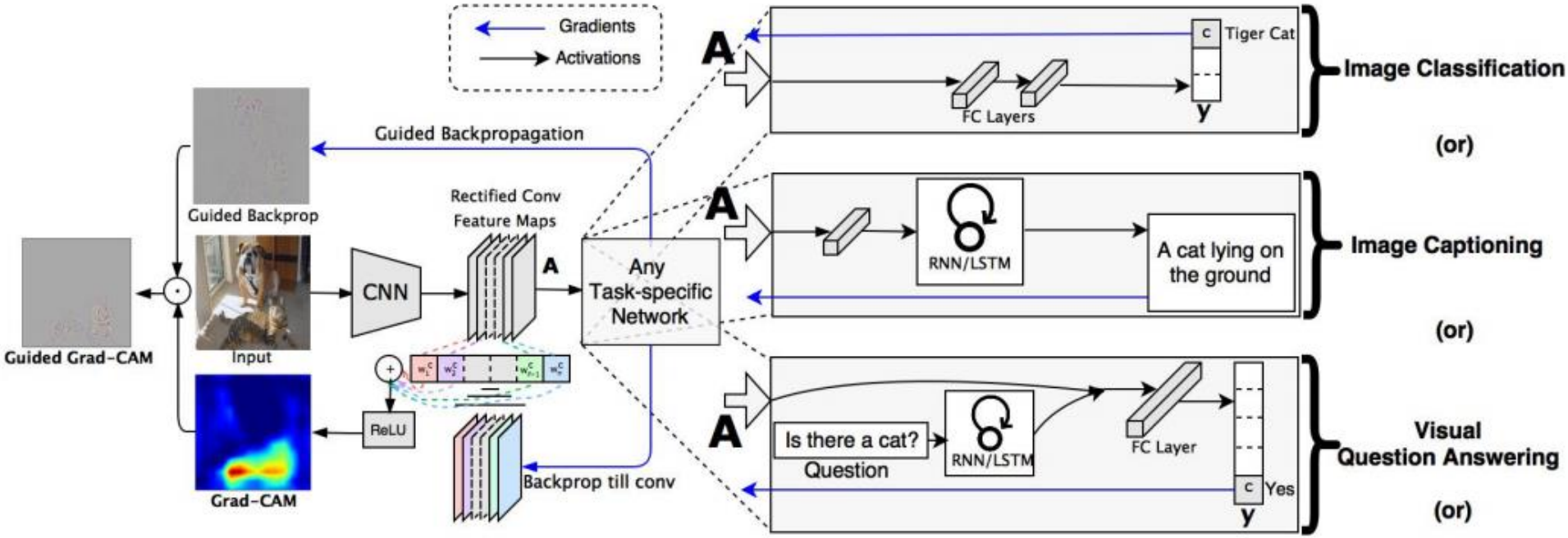
discriminative regions

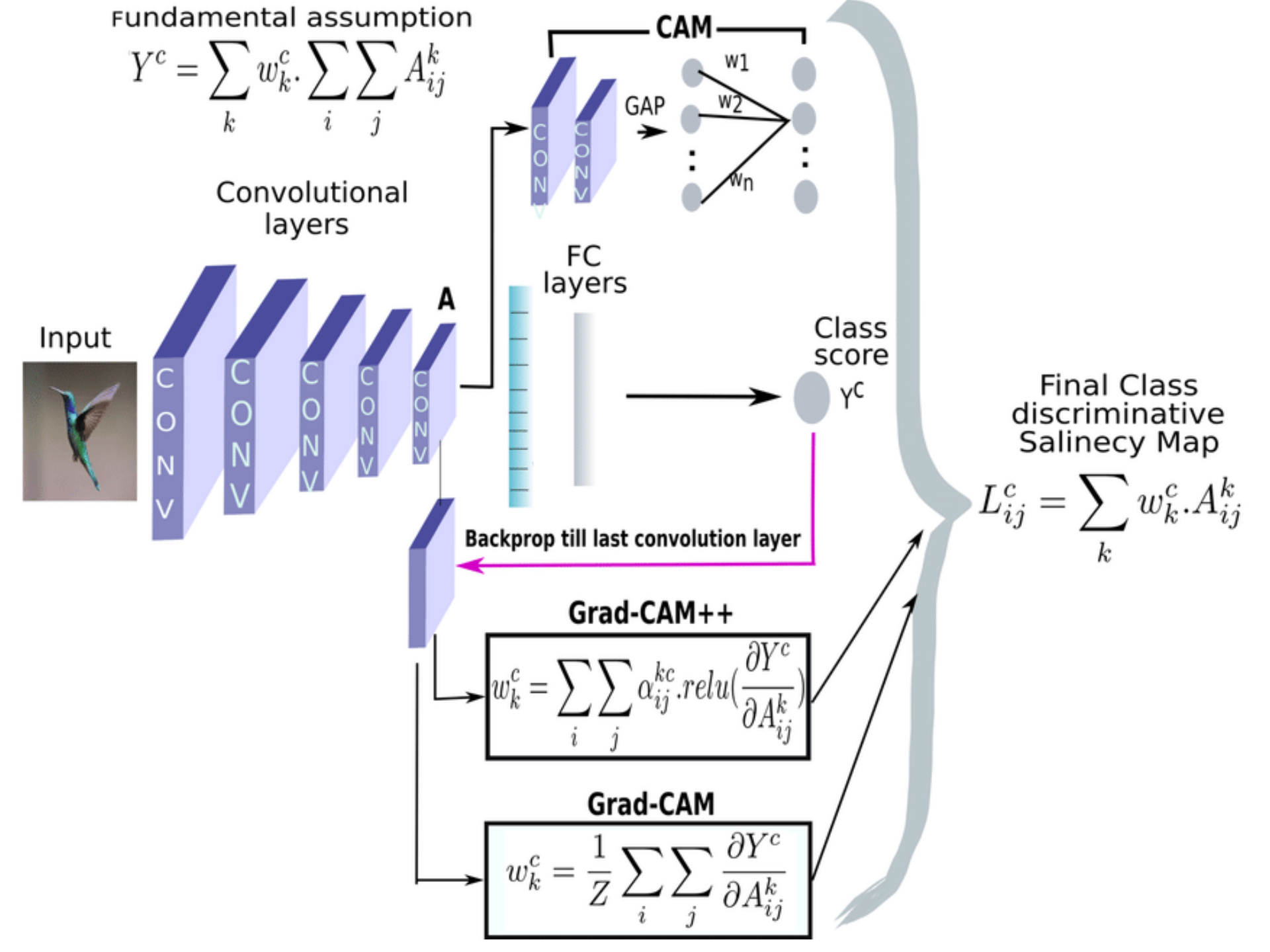
Original Image - Doctor

Predicted : Nurse
Grad-CAM for biased model

Predicted : Doctor
Grad-CAM for unbiased model

Class Activation Map (CAM) - Visual Explanation Method




Fundamental assumption
$Y^c = \sum_k w_k^c . \sum_i \sum_j A_{ij}^k$
CAM
GAP
w1
w2
wn
Convolutional layers
FC layers
A
Input
Class score
Y^c
Backprop till last convolution layer
Grad-CAM++
$w_k^c = \sum_i \sum_j \alpha_{ij}^{kc} . relu(\frac{\partial Y^c}{\partial A_{ij}^k})$
Grad-CAM
$w_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial Y^c}{\partial A_{ij}^k}$
Final Class discriminative Salinecy Map
$L_{ij}^c = \sum_k w_k^c . A_{ij}^k$

Class Activation Map (CAM) - Visual Explanation Method

- 높은 test accuracy

BUT!

- 넥타이 색깔만으로 판단 (데이타 바이어스 문제)
- 두명의 주자가 동시에 나오거나
- 아무도 나오지 않는 경우는….

https://jsideas.net/class_activation_map/

XAI

- **평가 방법**
  - 어떤 설명이 좋은 설명이지?
  - 1) Real-world User Evaluation
  - 2) Human Subjective Evaluation
  - 3) Functional Evaluation

Image sources: https://msaffarm.github.io/projects/ml- mri/Sahebzamani.Saffar.HSZ.Jan.2019.pdf https://artemisinccouk.wordpress.com/author/torak289/

올바른 분류를 할지라도

높은 Metrics (accuracy, precision, recall) 만을 가지고

딥러닝 모델을 믿을수 있을까??

이 답을 얻기위해

네트워크 침입탐지 딥러닝 모델을 구현하고

믿을수 있는가

분석해봤습니다

딥러닝 모델 구현 & 신뢰도 평가

- **Data**
  - Real network traffic based CSIC 2010 HTTP dataset
  - HTTP packets to detect web attacks
  - SQL injection, Buffer overflow, Information gathering, File disclosure
  - Normal : 18,640개, Anomalous : 15,878개
  - Example of anomalous http request - Server-Side Includes(SSI) Injection

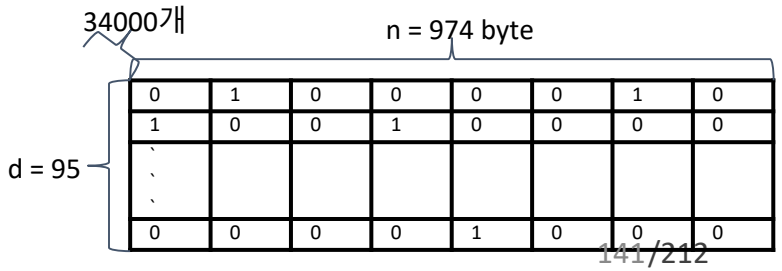
Server-Side Includes(SSI) Injection

Start-line discriminative region

Non discriminative region

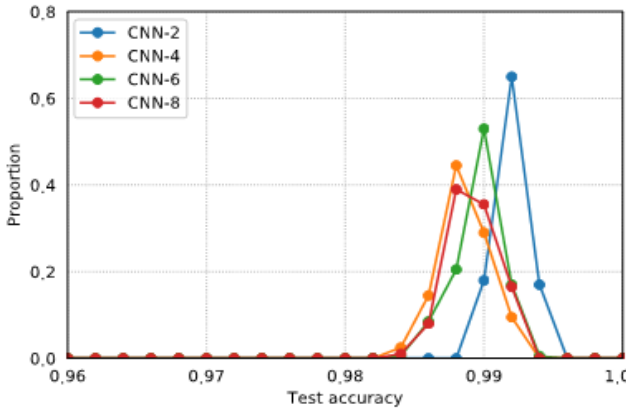
Body discriminative region

```
GET http://localhost:8080/tienda1/publico/autenticar.jsp?modo=entrar&login=espinosa&pwd=candado&remember=<!--#EXEC+cmd="ls
+/"-->&B1=Entrar HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=669BC99B1A18BD98674F606B40D7DA0F
Connection: close
```

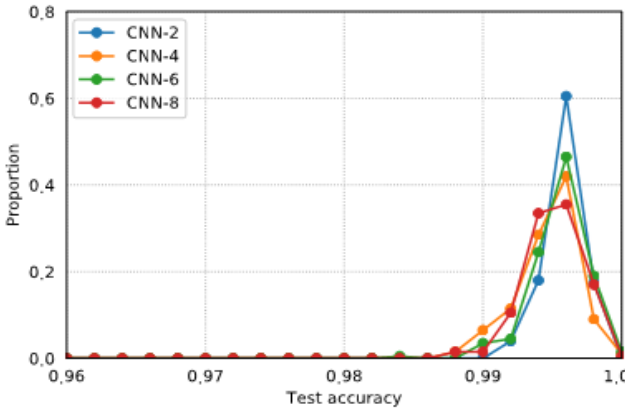


딥러닝 모델 구현 & 신뢰도 평가

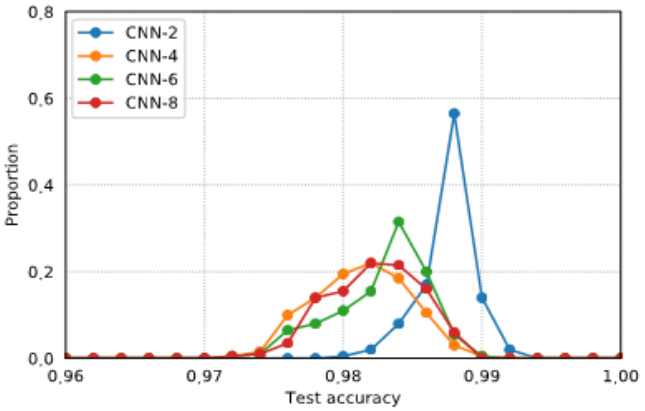
- **CNN 기반 네트워크 침입탐지 분류 모델 구현**
  - 모든 모델 모두 10 epochs 만에 **98% 이상** validation accuracy 이룸
  - 의외로 **가장 얕은** CNN-2의 성능이 **제일 좋음**
  - 다른 모델에 비해 smaller receptive field, **short pattern 이용**하는것이 CSIC 데이타셋에 적절



Total accuracy



Normal accuracy



Anomalous accuracy

1. Normal http request 분석

Start-line discriminative region

Non discriminative region

Body discriminative region

```
POST http://localhost:8080/tienda1/publico/pagar.jsp HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=F2ED4F8EB0F64C52F21B54ABF622B10D
Content-Type: application/x-www-form-urlencoded
Connection: close
Content-Length: 38

modo=insertar&precio=1525&B1=Confirmar
```

Normal acceptable http request

```
POST http://localhost:8080/tienda1/publico/pagar.jsp HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=F2ED4F8EB0F64C52F21B54ABF622B10D
Content-Type: application/x-www-form-urlencoded
Connection: close
Content-Length: 38

modo=insertar&precio=1525&B1=Confirmar
```

Normal suspicious http request

1. Normal http request 분석

Start-line discriminative region

Non discriminative region

Body discriminative region

```
POST http://localhost:8080/tienda1/publico/pagar.jsp HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (l
ike Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,te
xt/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=F2ED4F8EB0F64C52F21B54ABF622B10D
Content-Type: application/x-www-form-urlencoded
Connection: close
Content-Length: 38

modo=insertar&precio=1525&B1=Confirmar
```

Normal acceptable http request

```
POST http://localhost:8080/tienda1/publico/pagar.jsp HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (l
ike Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,te
xt/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=F2ED4F8EB0F64C52F21B54ABF622B10D
Content-Type: application/x-www-form-urlencoded
Connection: close
Content-Length: 38

modo=insertar&precio=1525&B1=Confirmar
```

Normal acceptable http request

단지 Ng은 올바른 답을 냈을지라도

항상 discriminative part를 보고 찾진않는다

2. Anomalous http request 분석, Web proxy Tool

Start-line
discriminative
region

Non
discriminative
region

```
GET http://localhost:8080/tienda1/publico/anadir.jsp?id=1&nombre=Queso+Manchego&precio=paros"+style="background:url(javascript:alert('Paros'))&id=2&cantidad=24&B1=Añadir+al+carrito HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=525D7B82C20954E61D3D561DE39E7BDC
Connection: close
```

Anomalous acceptable http request

```
GET http://localhost:8080/tienda1/publico/anadir.jsp?id=1&nombre=Queso+Manchego&precio=paros"+style="background:url(javascript:alert('Paros'))&id=2&cantidad=24&B1=Añadir+al+carrito HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=525D7B82C20954E61D3D561DE39E7BDC
Connection: close
```

Anomalous suspicious http request

3. Anomalous http request 분석, Server-Side Includes(SSI) Injection

Start-line discriminative region

Non discriminative region

```
GET http://localhost:8080/tienda1/publico/autenticar.jsp?modo=<!--#exec+cmd="rm+-rf+/;cat+/etc/passwd"+-->&login=darb2&pwd=h4_Bi651A2&remember=on&B1=Entrar HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=E202D64495248B3E18500148A76CFE12
Connection: close
```

Anomalous acceptable http request

```
GET http://localhost:8080/tienda1/publico/autenticar.jsp?modo=<!--#exec+cmd="rm+-rf+/;cat+/etc/passwd"+-->&login=darb2&pwd=h4_Bi651A2&remember=on&B1=Entrar HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=E202D64495248B3E18500148A76CFE12
Connection: close
```

Anomalous suspicious http request

3. Anomalous http request 분석, Server-Side Includes(SSI) Injection

Start-line discriminative region

Non discriminative region

```
GET http://localhost:8080/tienda1/publico/autenticar.jsp?modo=<!--#exec+cmd="rm+-rf+/;cat+/etc/passwd"+-->&login=darb2&pwd=h4_Bi651A2&remember=on&B1=Entrar HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=E202D64495248B3E18500148A76CFE12
Connection: close
```

Anomalous acceptable http request

```
GET http://localhost:8080/tienda1/publico/autenticar.jsp?modo=<!--#exec+cmd="rm+-rf+/;cat+/etc/passwd"+-->&login=darb2&pwd=h4_Bi651A2&remember=on&B1=Entrar HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=E202D64495248B3E18500148A76CFE12
Connection: close
```

Anomalous suspicious http request

딥러닝은 같은 데이타, 모델구조, 파라미터 일지라도 매번 분류 기준이 다르다

1. 딥러닝은 올바른 답을 냈을지라도

항상 discriminative part를 보고 찾진않는다

2. 같은 데이타, 모델구조, 파라미터 일지라도

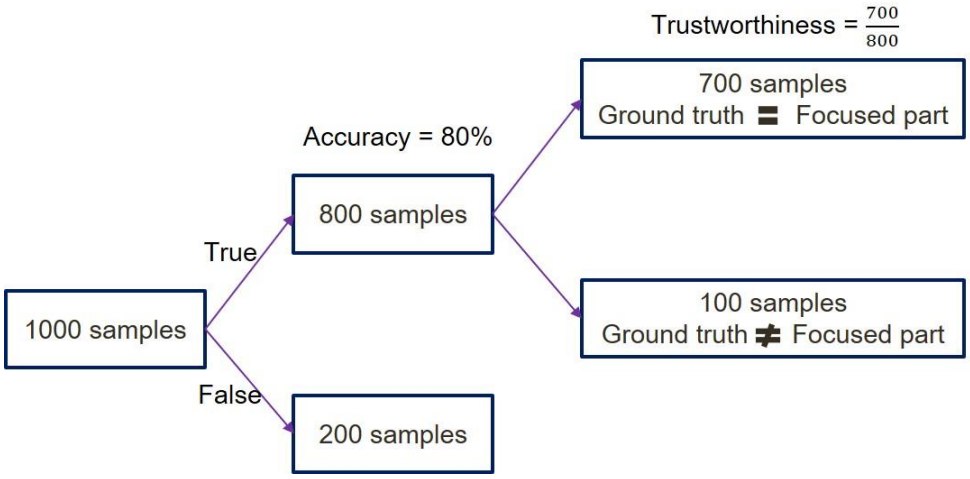
매번 분류 기준이 다르다

그럼, 다른 상황, 다른 아이디, 다른 데이타인 경우

99% 정확도를 믿을 수 있는가??

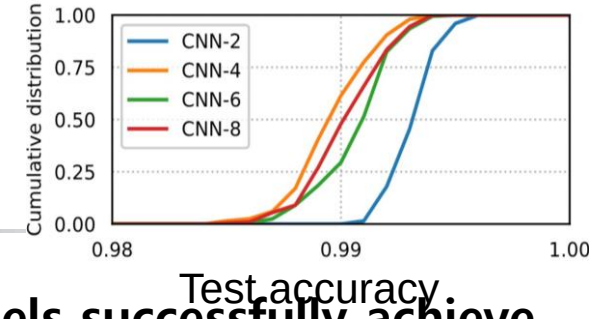
Trustworthiness

- To check whether the classification criteria of a model are trustworthy or not in case of “true positive” test samples.
- The classification criteria of a model are trustworthy when the focused regions of “true positive” test samples overlap the ground truth of the semantically discriminative regions.
- Trustworthiness can be low because of overfitting or data bias.

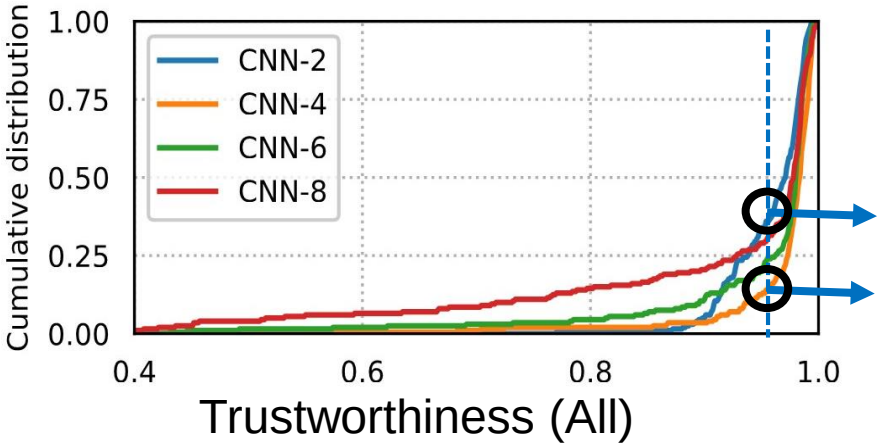


[Trustworthiness example]

Trustworthiness

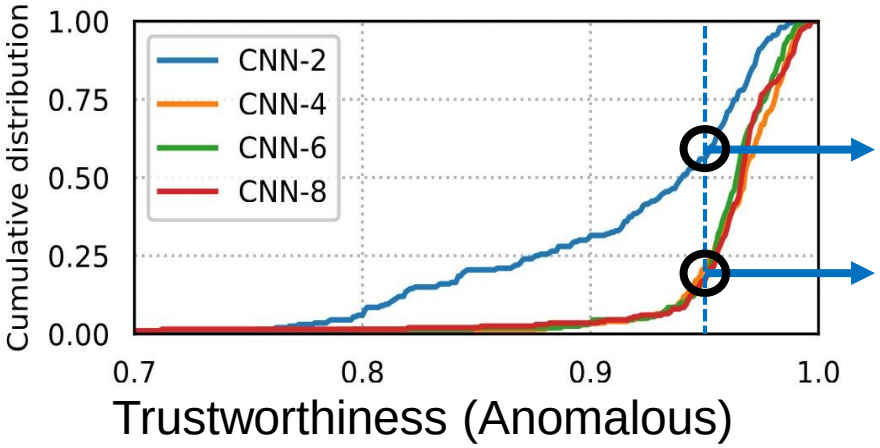


- **During the test performed in 200 times, all models successfully achieve more than 98% of test accuracy.**



CNN-2 : Trustworthiness : 64

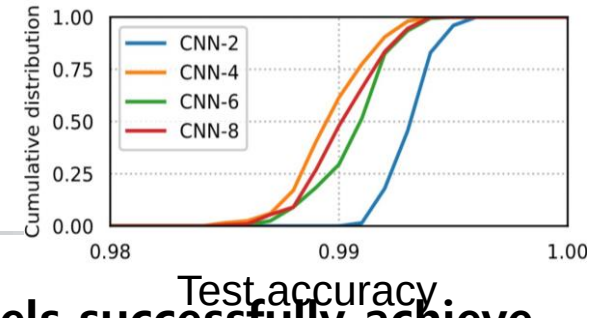
CNN-4 : Trustworthiness : 87.5



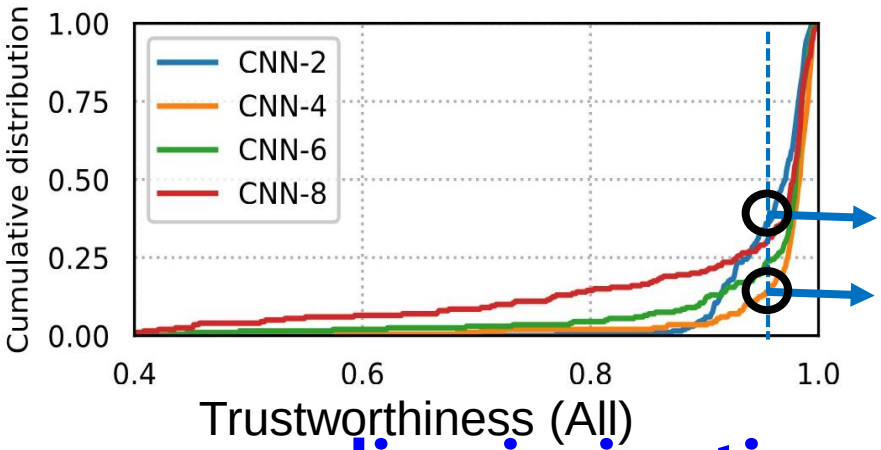
CNN-2 : Trustworthiness : 42

CNN-8 : Trustworthiness : 83

Trustworthiness



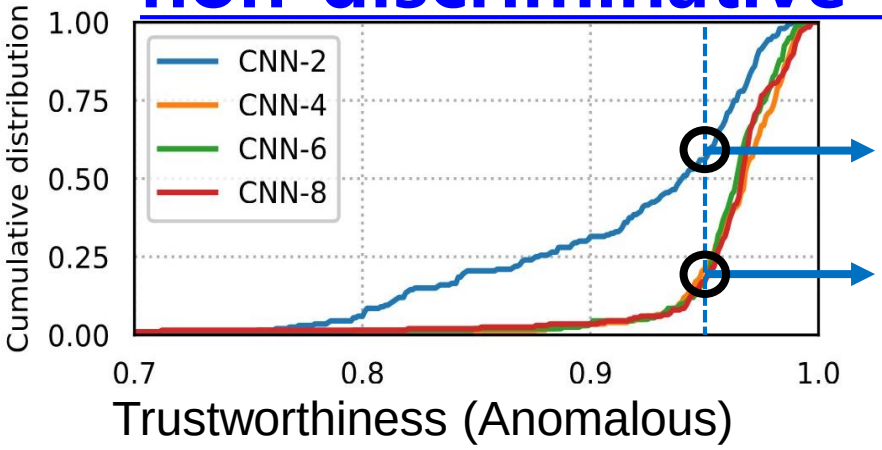
- **During the test performed in 200 times, all models successfully achieve more than 98% of test accuracy.**



CNN-2 : Trustworthiness : 64

CNN-4 : Trustworthiness : 87.5

non-discriminative part를 보고 분류하는 딥러닝 모델 신뢰할수 있는가??



CNN-2 : Trustworthiness : 42

CNN-8 : Trustworthiness : 83

신뢰도(Trustworthiness) 분석

- **신뢰도 있는 딥러닝 모델을 만들기 위해**

- **추가적인 Annotation 정보가 필요하다**
  - Test data with annotation
  - 모델의 분류 기준이 올바른지 판단

- **학습 시 Annotation 정보 활용**
  - Train data with annotation(일부)
  - CNN : Attention correction, CAM loss
  - RNN : Guided attention
  - get to know where they need to improve
  - 신뢰도 있는 분류 모델 학습 가능
  - Adversarial attack에 강인할 수 있음

https://arxiv.org/pdf/1605.09553.pdf
https://arxiv.org/pdf/1907.10830.pdf
https://arxiv.org/pdf/1710.08969.pdf

신뢰도(Trustworthiness) 분석

- **학습 시 Annotation 정보 활용**
  - Train data with annotation(일부)
  - **CNN : Attention correction, CAM loss**
  - RNN : Guided attention
  - get to know where they need to improve

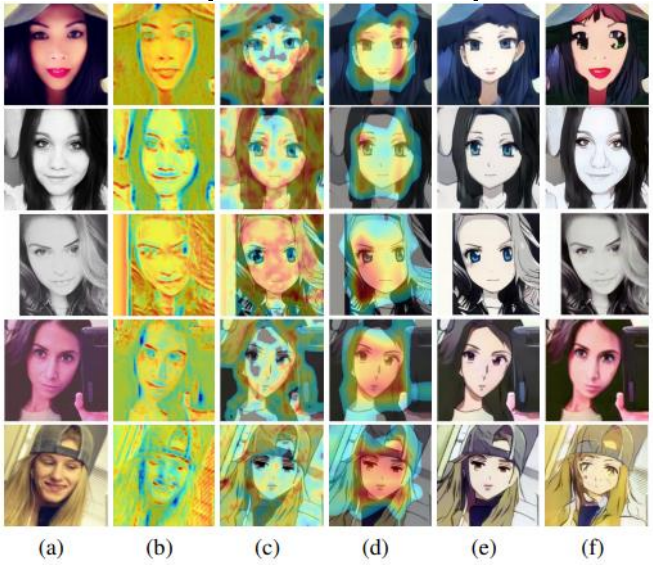


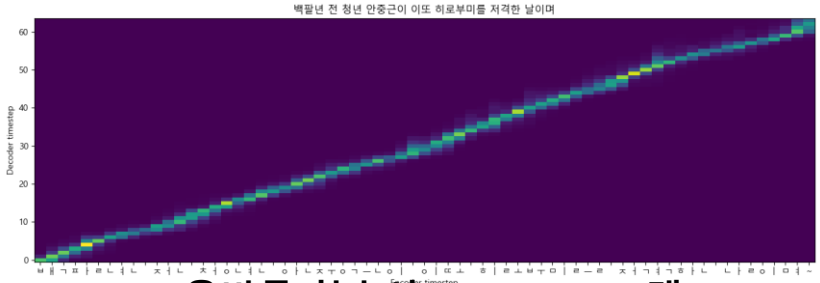
Figure 2: Visualization of the attention maps and their effects shown in the ablation experiments: (a) Source images, (b) Attention map of the generator, (c-d) Local and global attention maps of the discriminator, respectively. (e) Our results with CAM, (f) Results without CAM.

[U-GOT-IT CAM loss 적용]

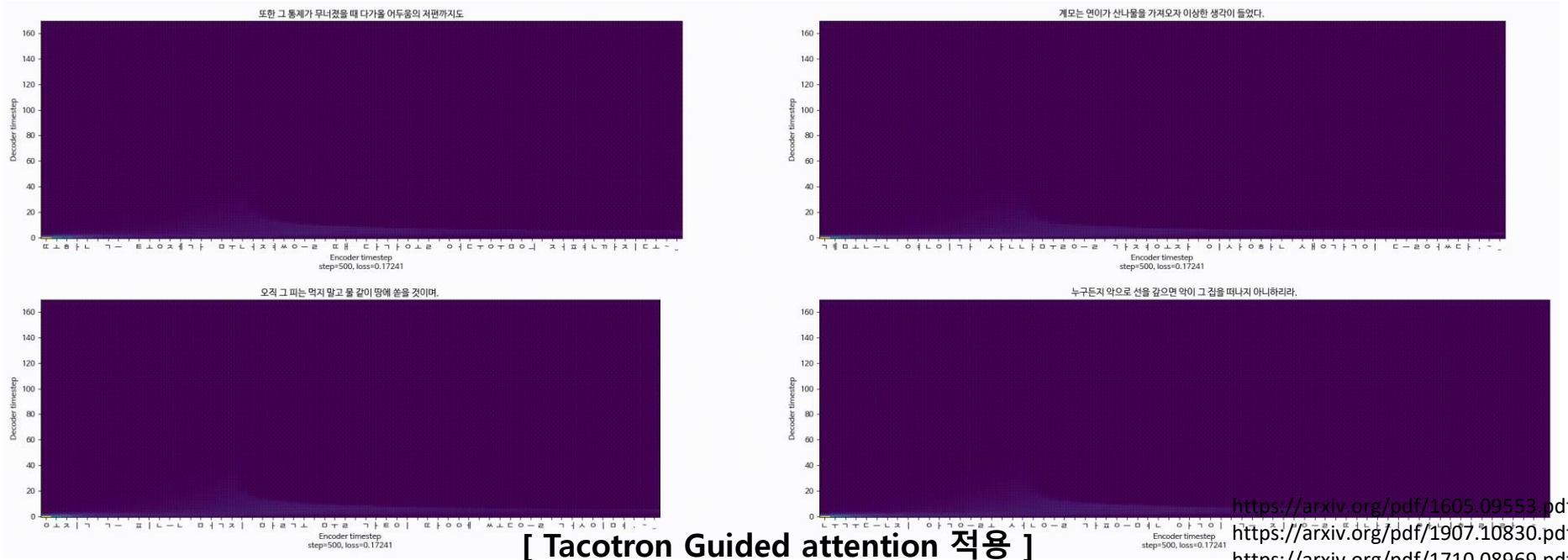
https://arxiv.org/pdf/1605.09553.pdf
https://arxiv.org/pdf/1907.10830.pdf
https://arxiv.org/pdf/1710.08969.pdf

신뢰도(Trustworthiness) 분석

- 학습 시 Annotation 정보 활용
  - Train data with annotation(일부)
  - CNN : Attention correction, CAM loss
  - **RNN : Guided attention**
  - get to know where they need to improve



[올바른 학습시 attention 그래프]



[Tacotron Guided attention 적용]

https://arxiv.org/pdf/1605.09553.pdf
https://arxiv.org/pdf/1907.10830.pdf
https://arxiv.org/pdf/1710.08969.pdf

Trustworthiness 결론

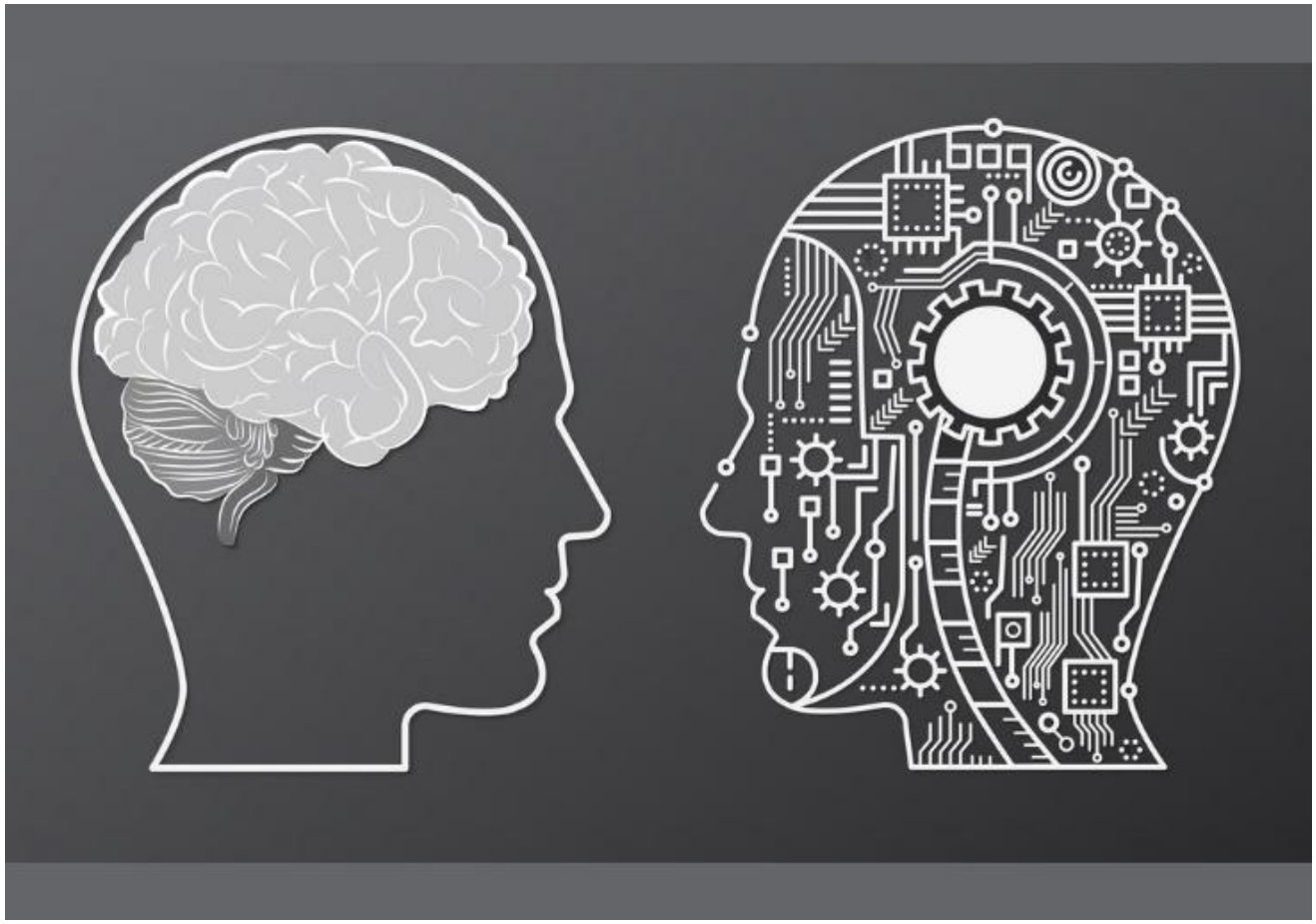
- 딥러닝 모델의 판단 근거도 신뢰할 수 있어야 한다
- 레이블 만으로는 신뢰할 수 있는 분류기준을 갖도록 학습하기 어렵다
- Metric 만으로 모델을 신뢰할 수 없다
- 추가적인 Annotation 정보가 필요하다
  - Test data with annotation
- 모델의 분류 기준이 올바른지 판단할 수 있다
  - Train data with annotation
- 학습과정에서 모델에게 입력의 어느 부분이 중요하고 의미 있는지를 가이드 할 수 있다

Adversarial Attack

White/Black box, Adversarial Training

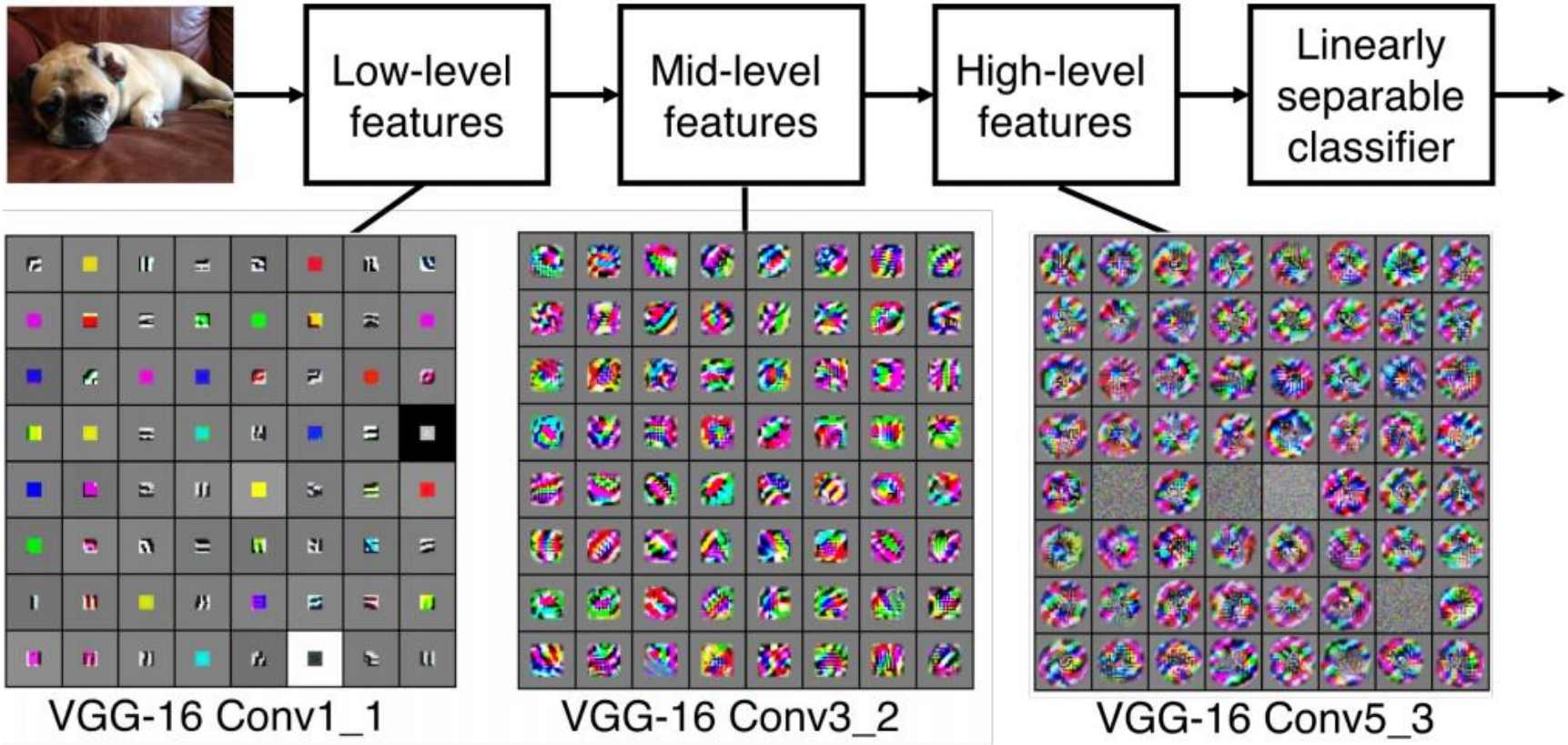
AA

- AI는 사람의 모방인가?
- AI는 사람과 같이 동작하는가?



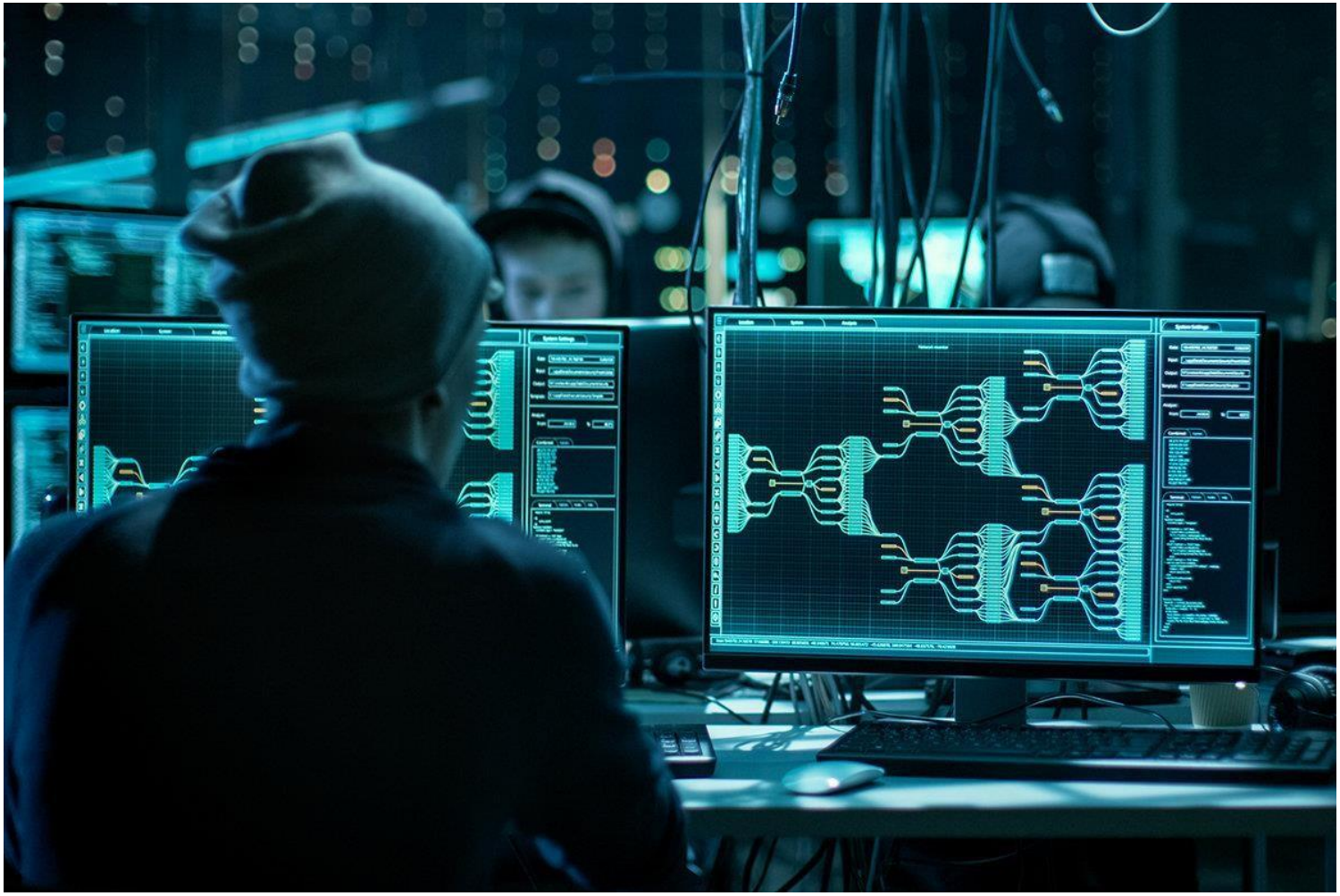
CNN

- 이미지 분류모델 – CNN



AA-Definition

- 사람 눈에 띄지 않으면서, Network가 원본과 다르게 판단하도록 하는 기술

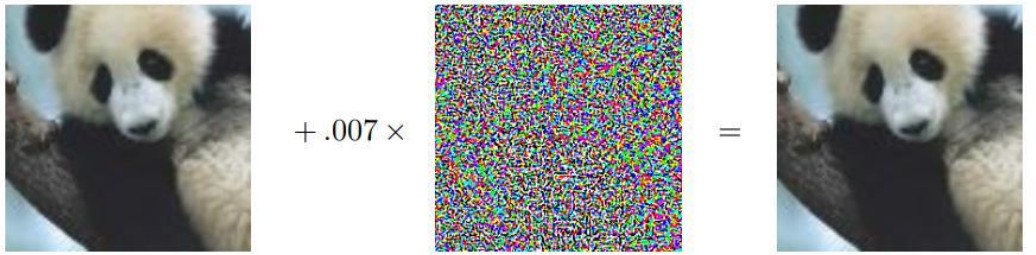


AA-Definition

- **사람 눈에 띄지 않으면서, Network가 원본과 다르게 판단하도록 하는 기술**

- **Adversarial Attack 기술: Adversarial Example 생성 기술**

- **Adversarial Example**
  - 사람의 눈에 띄지 않으면서, Network가 원본과 다른 판단을 하게 하는 영상
  - Adversarial Attack을 생성할 수 있는 정보가 다를 수 있음



$\boldsymbol{x}$

"panda"
57.7% confidence

$+ .007 \times$

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$=$

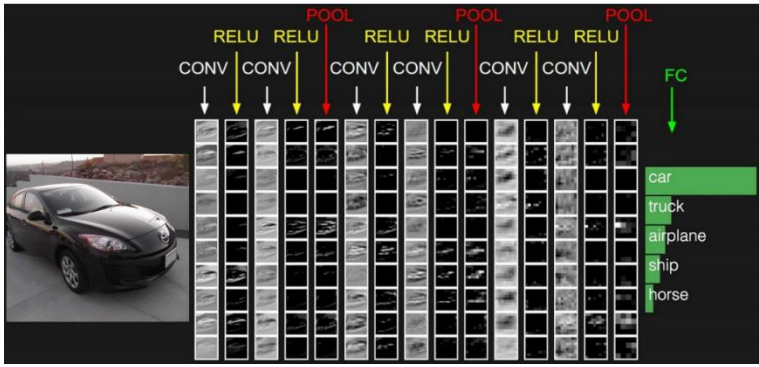
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

https://www.facebook.com/photo/?fbid=6493137367378939&set=gm.1181641385593713

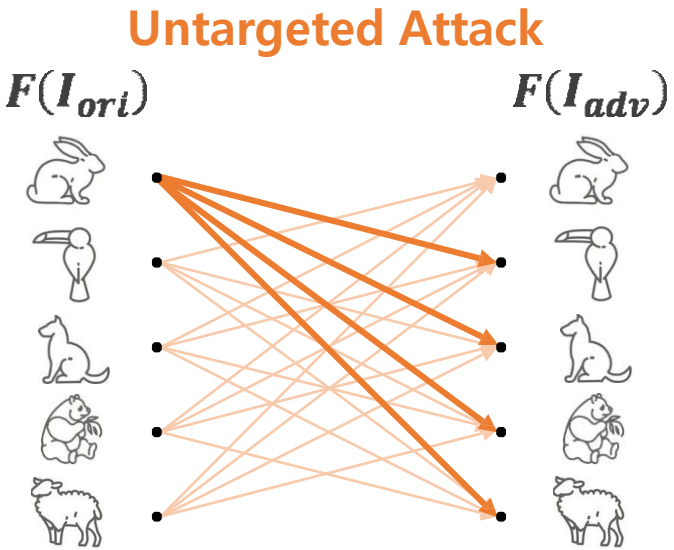
AA-종류1) Scenario

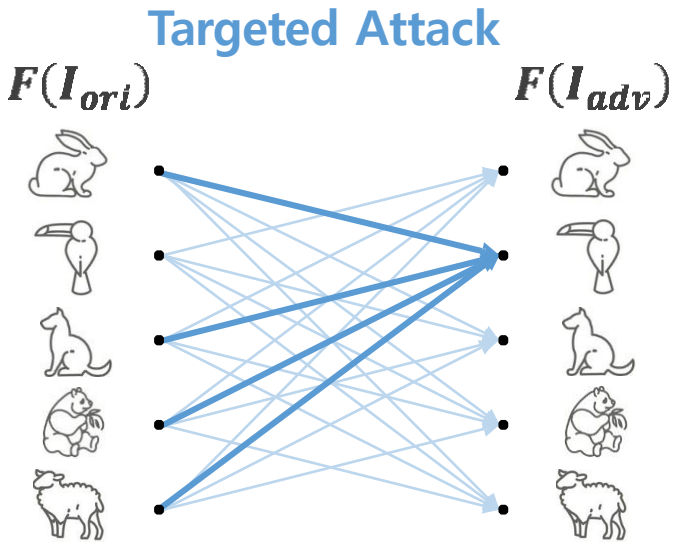
- **공격자는 얼마나 많은 정보를 가질 수 있는가?**
- **White Box Attack**
  - 공격자가 모델 정보에 접근 가능한 상태(접근 정보에 제한 없음)
  - Attacker가 Model & Output Logit의 모든 정보를 갖고 있음
  - Gradient를 직접 계산할 수 있음
- **Black Box Attack**
  - Attacker가 Weight Parameter는 모름
  - Attacker가 Model 구조를 알수도/모를수도
  - 분류 결과값만? Logit값도?
  - Test data loss?
  - Query 횟수가 무한정 허용? 제한허용?



AA-종류2) Target

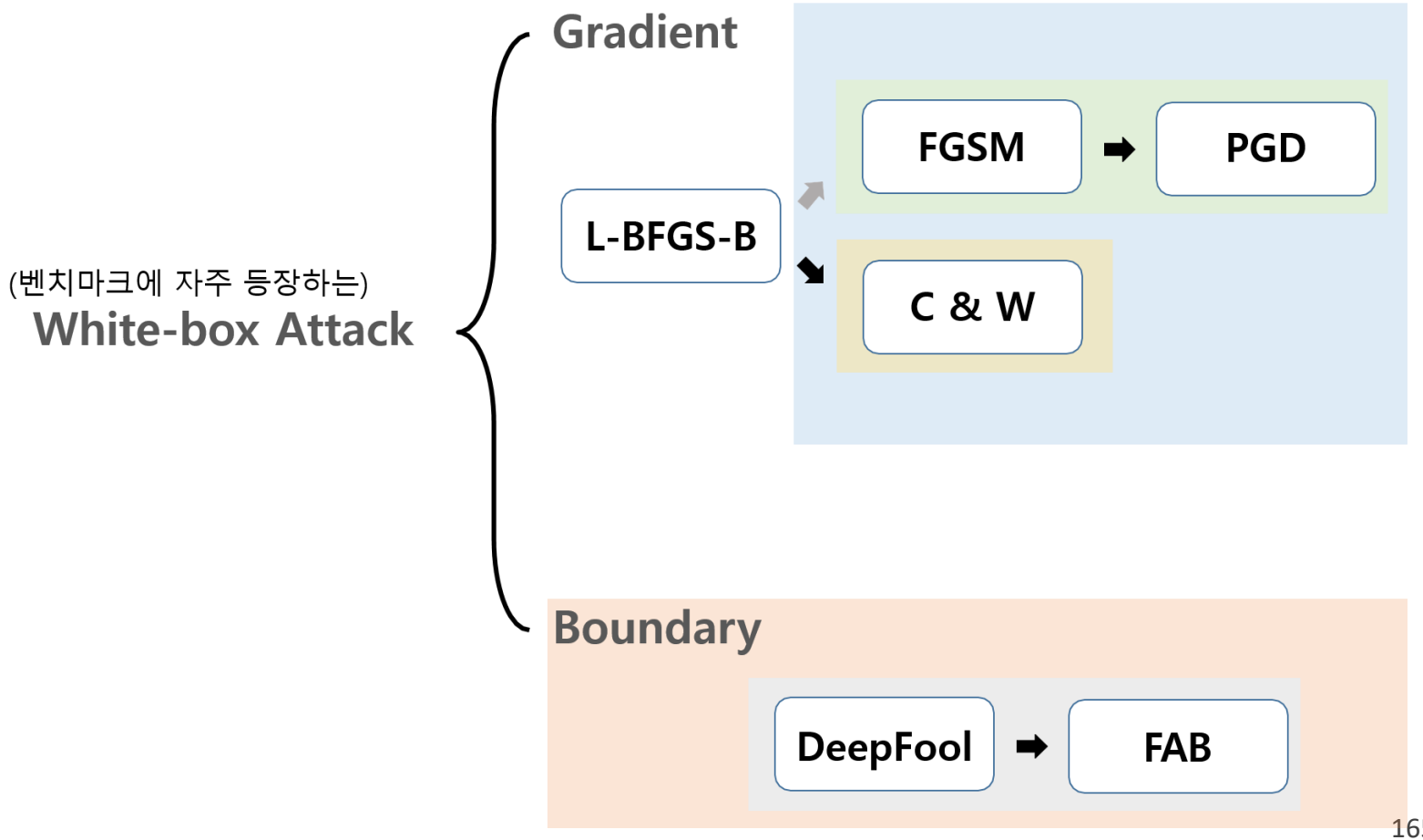
- The way network are fooled :



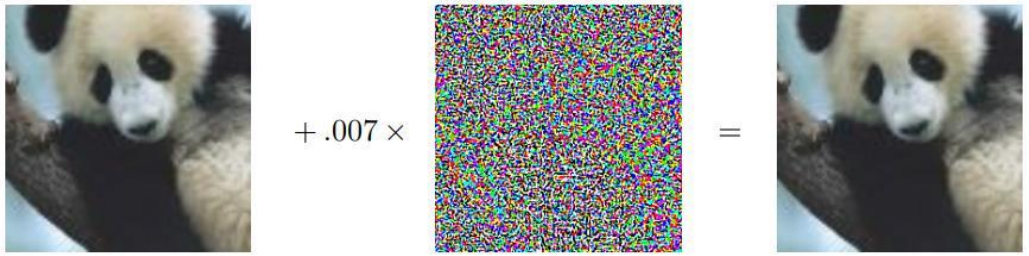


White-box Attack

- 공격자가 모든 정보에 접근 가능



White-box Attack 1



$\boldsymbol{x}$
"panda"
57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence

$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

- **FGSM (Fast Gradient Sign Method)**
  - Adversarial Example을 만드는 방식

$$I_{adv} = I_{ori} + \varepsilon\, sign\,(\nabla_I\, Loss(F(I_{ori}))) \qquad \|\boldsymbol{I_{adv}} - \boldsymbol{I_{ori}}\|_\infty \leq \varepsilon$$

Network Training

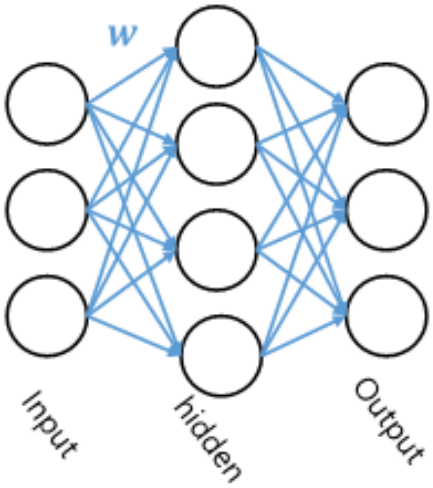
$$argmin_w\ Loss(F(I, w), l_{GT})$$

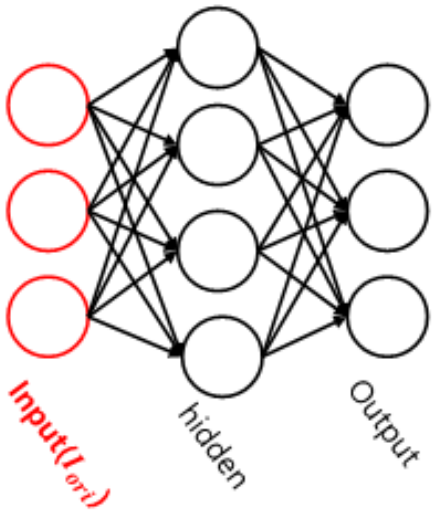
$$w_{i+1} = w_i - \alpha\nabla_w Loss(F(I, w_i), l_{GT})$$

FGSM

$$argmax_I\ Loss(F(I, w), l_{GT})$$

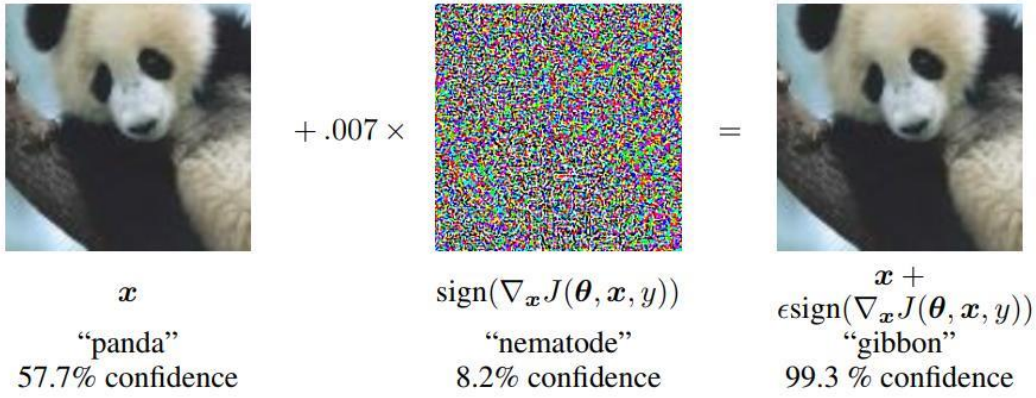
$$I_{adv} = I_{ori} + \varepsilon\nabla_I Loss(F(I_{ori}, w), l_{GT})$$





I. Goodfellow et al. "Explaining and Harnessing Adversarial Example", ICRL 2014

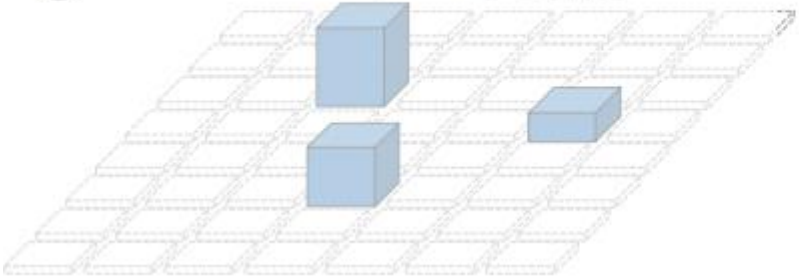
White-box Attack 1



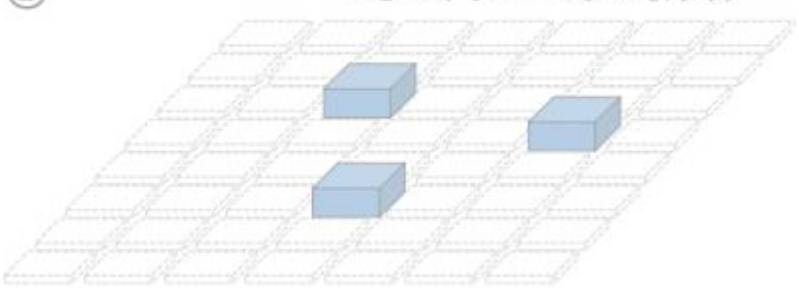
- **FGSM (Fast Gradient Sign Method)**
  - Adversarial Example을 만드는 방식

$$I_{adv} = I_{ori} + \varepsilon\, sign\,(\nabla_I\, Loss(F(I_{ori}))) \qquad \|I_{adv} - I_{ori}\|_\infty \le \varepsilon$$

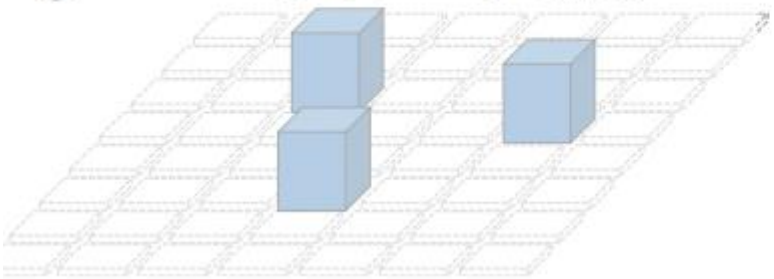
① $\nabla_I\, Loss(F(I_{ori}))$



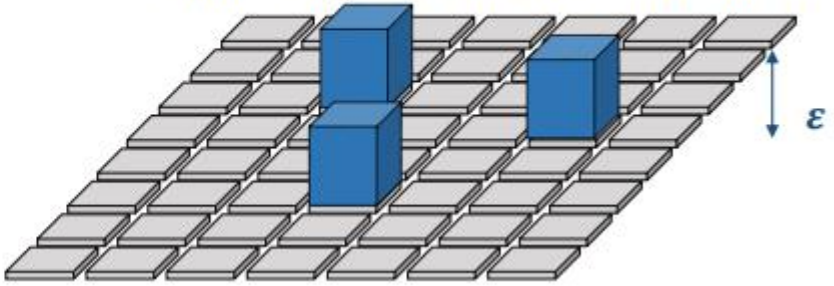
② $sign\,(\nabla_I\, Loss(F(I_{ori})))$



③ $\varepsilon\, sign\,(\nabla_I\, Loss(F(I_{ori})))$



④ $I_{adv} = I_{ori} + \varepsilon\, sign\,(\nabla_I\, Loss(F(I_{ori})))$



I. Goodfellow et al. "Explaining and Harnessing Adversarial Example", ICRL 2014

White-box Attack 2

- **PGD (Project Gradient Descent)**
  - FGSM을 iterative 하게 변형

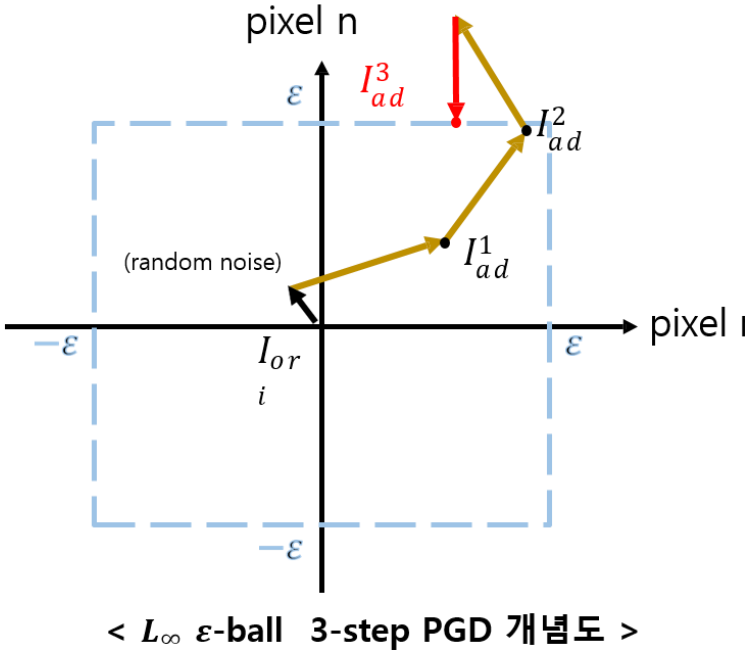
FGSM

$$I_{adv} = I_{ori} + \varepsilon\, sign\,(\nabla_I\, Loss_F(I_{ori}))$$ ← ... and clip $\max(\min(I_{adv}, 1), 0)$

PGD

$$I_{t_{temp}} = I_{t-1} + \alpha\, sign\,(\nabla_I\, Loss(F(I_{t-1}))) \leftarrow \text{... and clip } \max(\min(I_{adv}, 1), 0)$$

$$I_t = \min(\max(I_{t_{temp}}, I_{ori} - \varepsilon), I_{ori} + \varepsilon)$$



< L_∞ ε-ball 3-step PGD 개념도 >

[Pseudo Code (PGD)]

$I_{adv} = I_{ori} + Random\_noise$

$I_{adv} = \text{project}\,(I_{adv}, [I_{ori} + \varepsilon,\ I_{ori} - \varepsilon])$

for range(n_steps):

$I_{adv} = I_{adv} + \alpha\, sign\,(\nabla_I Loss_F\,(I_{ad}\,))$

$I_{adv} = \text{project}\,(I_{adv}, [I_{ori} + \varepsilon,\ I_{ori} - \varepsilon])$

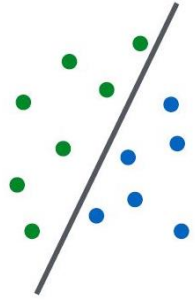
* restart, early stop can be added

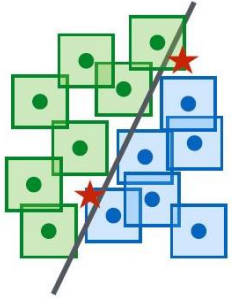
- *Recall FGSM* :

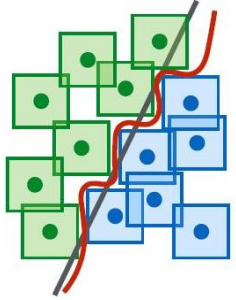
$$I_{adv} = I_{ori} + \varepsilon sign\,(\nabla_I Loss_F\ (I_{or}\,))$$

A. Madry et al. "Toward Deep Learning Models Resistant to Adversarial Attacks", ICRL 2018
B. 적대적 공격 및 방어 연구 소개, ETRI 황중언

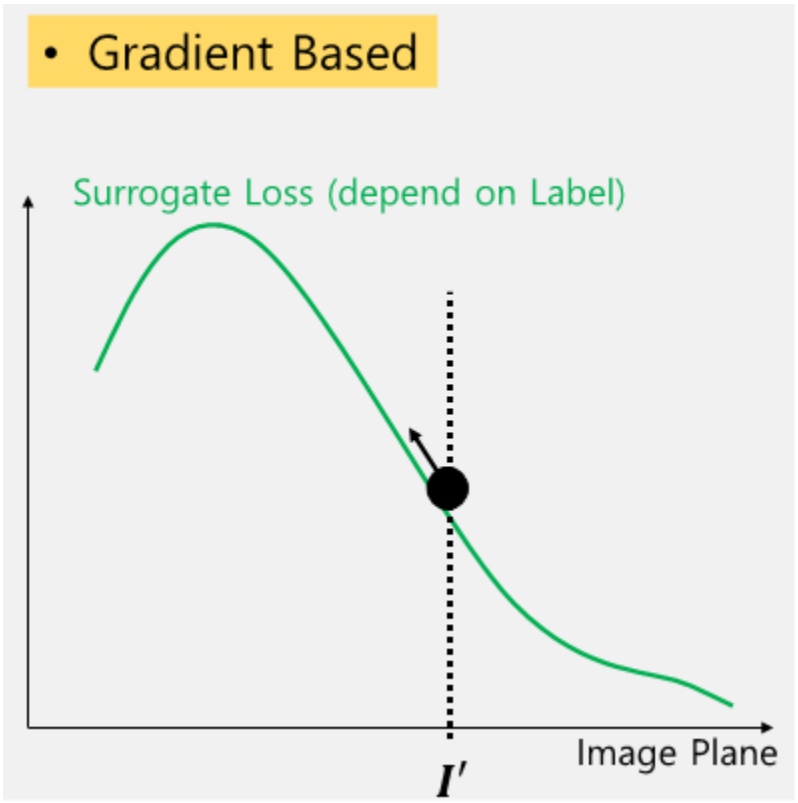
White-box Attack 3

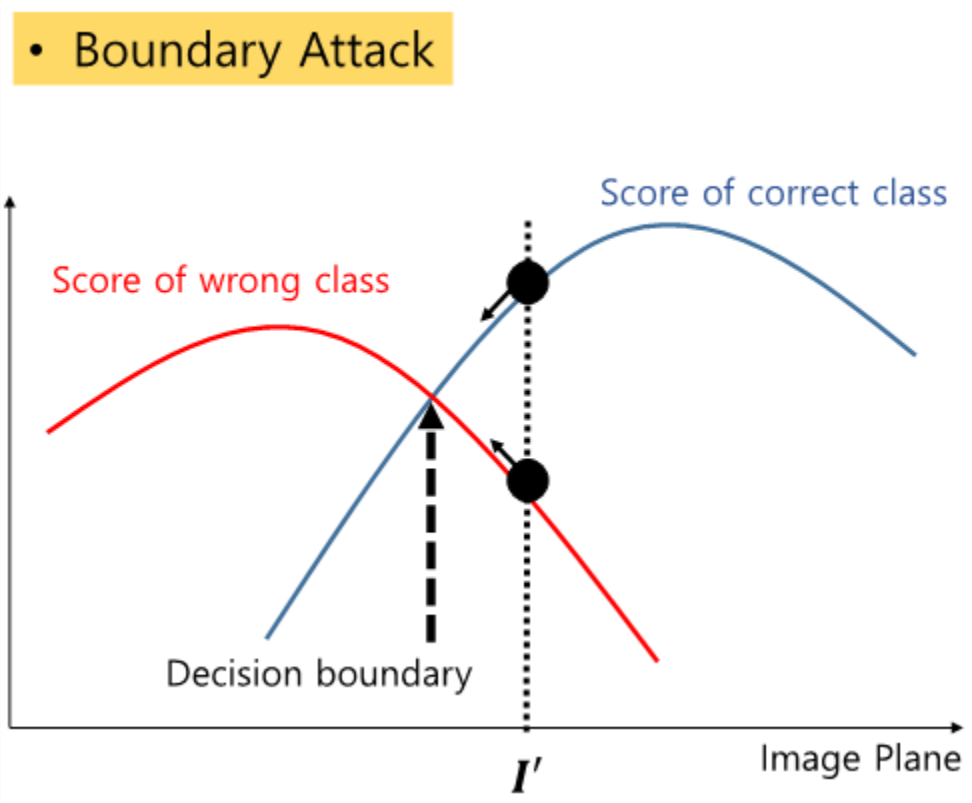






- **Boundary Attack, DeepFool**





S. Moosavi-Dezfooli et al. "DeepFooL: a simple and accurate method to fool deep neural networks", CVPR 2016

White-box Attack 실습

$$I_{adv} = I_{ori} + Random\_noise$$
$$I_{adv} = \text{project}\,(I_{adv}, [I_{ori} + \varepsilon,\ I_{ori} - \varepsilon])$$

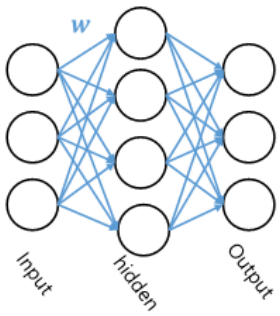
for range(n_steps):

$$I_{adv} = I_{adv} + \alpha\ sign\left(\nabla_I Loss_F\,(I_{ad}\ )\right)$$
$$I_{adv} = \text{project}\,(I_{adv}, [I_{ori} + \varepsilon,\ I_{ori} - \varepsilon])$$

- **Normal Case**
  - 1) 데이터 준비
  - 2) 모델 선언
  - 3) 학습
  - 4) 추론
    - input image to model, output result

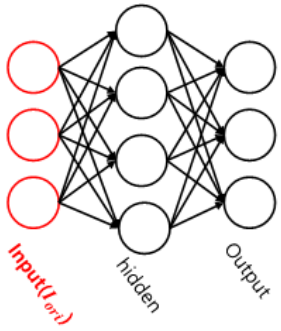
Network Training

$$argmin_w\ Loss(F(I,w), l_{GT})$$
$$w_{i+1} = w_i - \alpha\nabla_w Loss(F(I,w_i)\,, l_{GT})$$



FGSM

$$argmax_I\ Loss(F(I,w), l_{GT})$$
$$I_{adv} = I_{ori} + \varepsilon\nabla_I Loss(F(I_{ori}, w)\,, l_{GT})$$

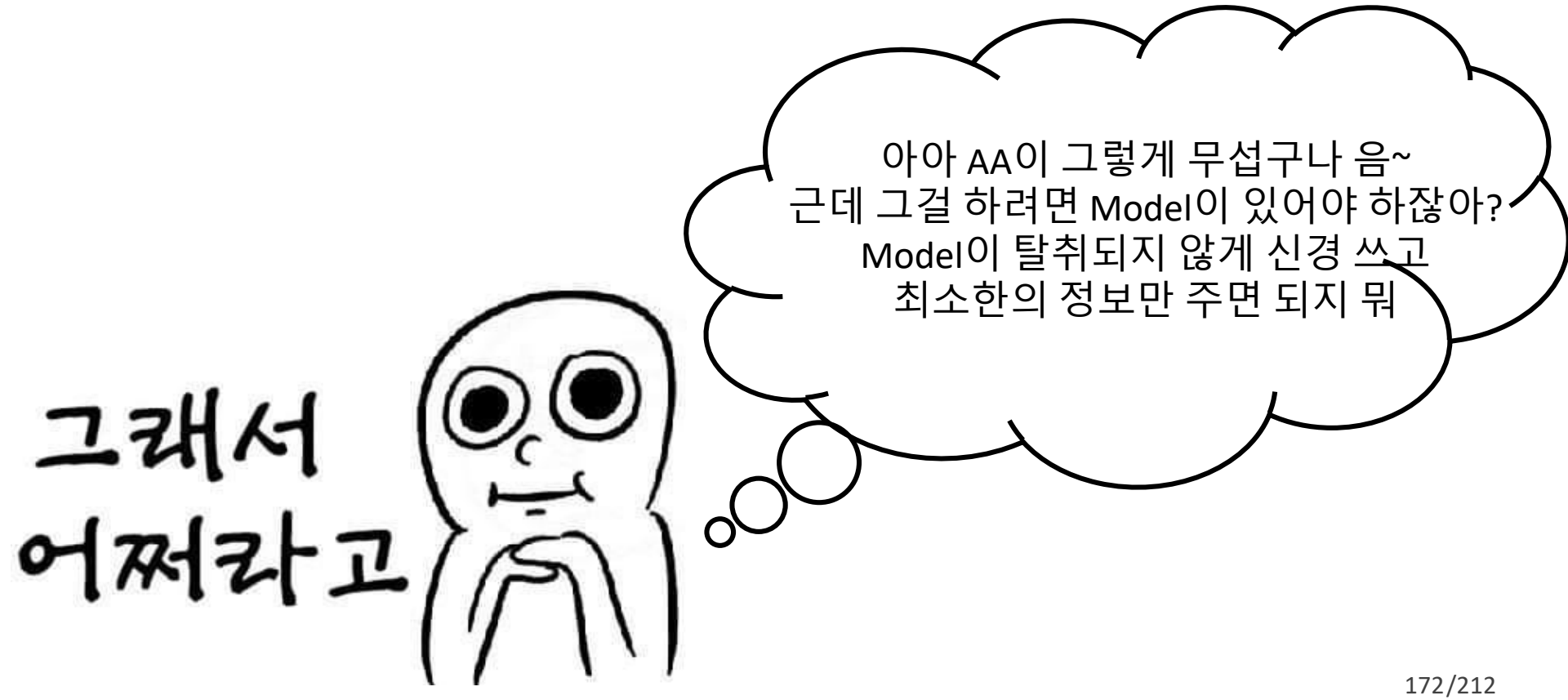


- **Adversarial Attack Case**
  - 1) 데이터 준비
  - 2) 모델 선언
  - 3) 학습
  - **4) Adversarial Attack**
    - 입력 이미지와 모델로 Adversarial Example을 만든다
    - 출력이 틀리는 방향으로 Gradient를 구한다
    - 들키지 않게 크기를 조절한다
    - n번 반복한다
  - 5) 추론
    - input Adversavial image to model, output result

White-box Attack

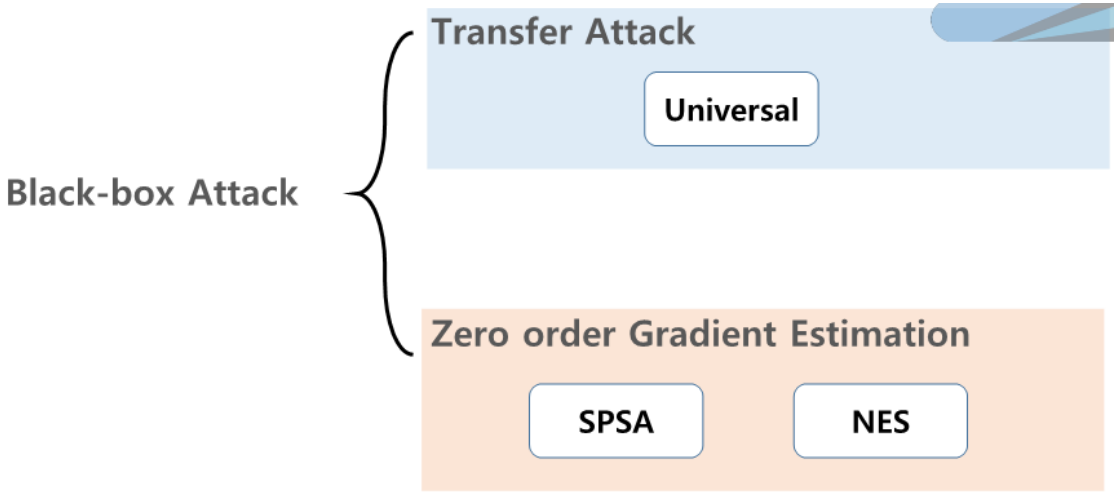
https://m.blog.naver.com/wnghmom/221769051372
https://jjalbot.com/jjals/Sy-FOP6UC

White-box Attack



Black-box Attack

- **Black-box Attack?**
  - Target model의 Gradient 정보에 접근할 수 없는 경우
  - Attacker가 Weight Parameter는 모름
  - Attacker가 Model 구조를 알수도/모를수도
  - 분류 결과값만? Logit값도?
  - Test data loss?
  - Query 횟수가 무한정 허용? 제한허용?



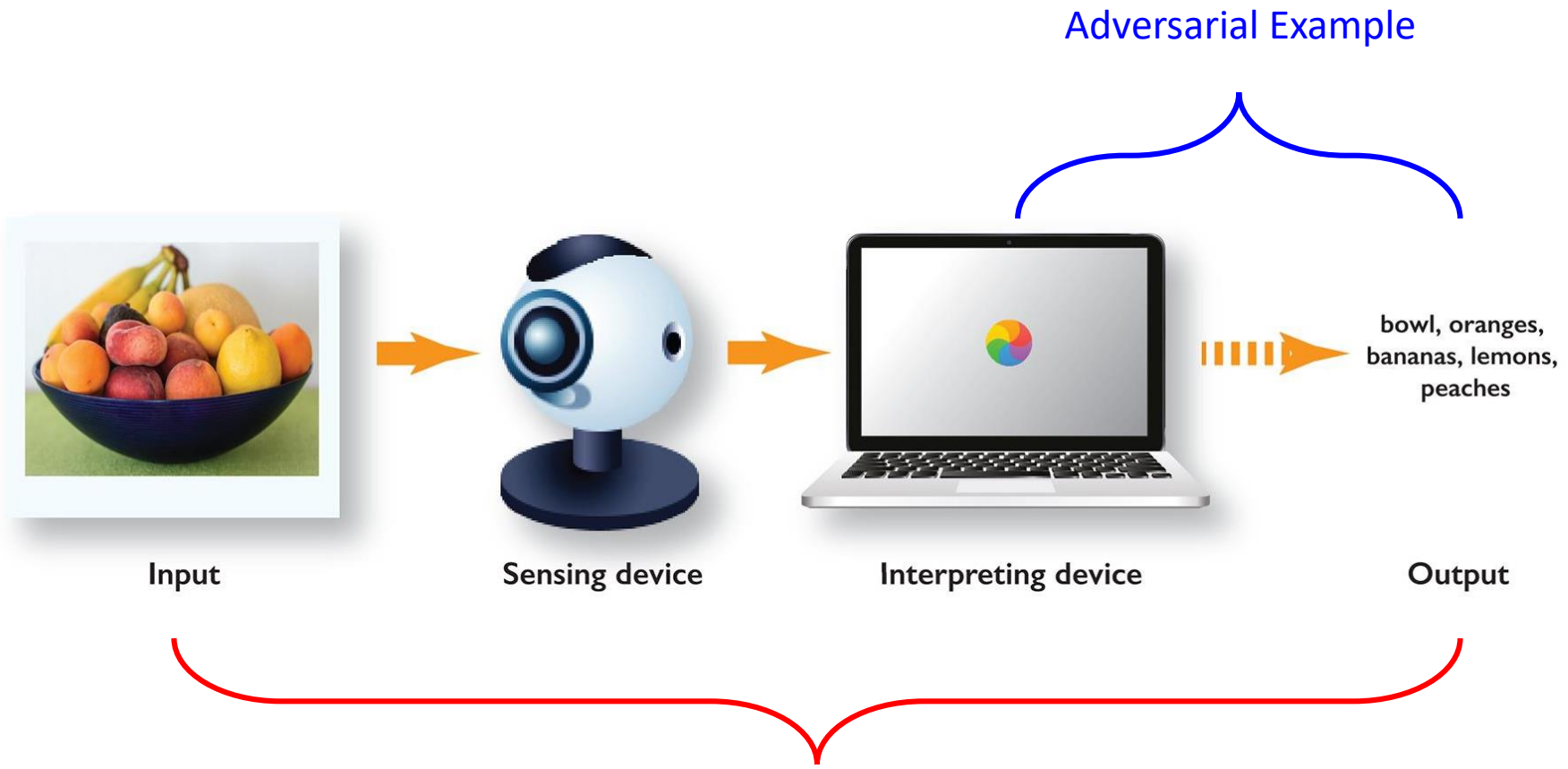
Black-box Attack

- **Transfer Attack**
  - AE는 혹시 그냥 어려운 문제가 아닐까?
  - 다른 Architecture, 다른 학습 데이터로 만들어도 Universal하게 작동하지 않을까?
  - A model에 학습한 Adversarial Example이, 다른 B model에도 Adversarial할 가능성 높음
  - Adversarial vulnerabilit도 Generalized. 즉 Attack이 Transferable

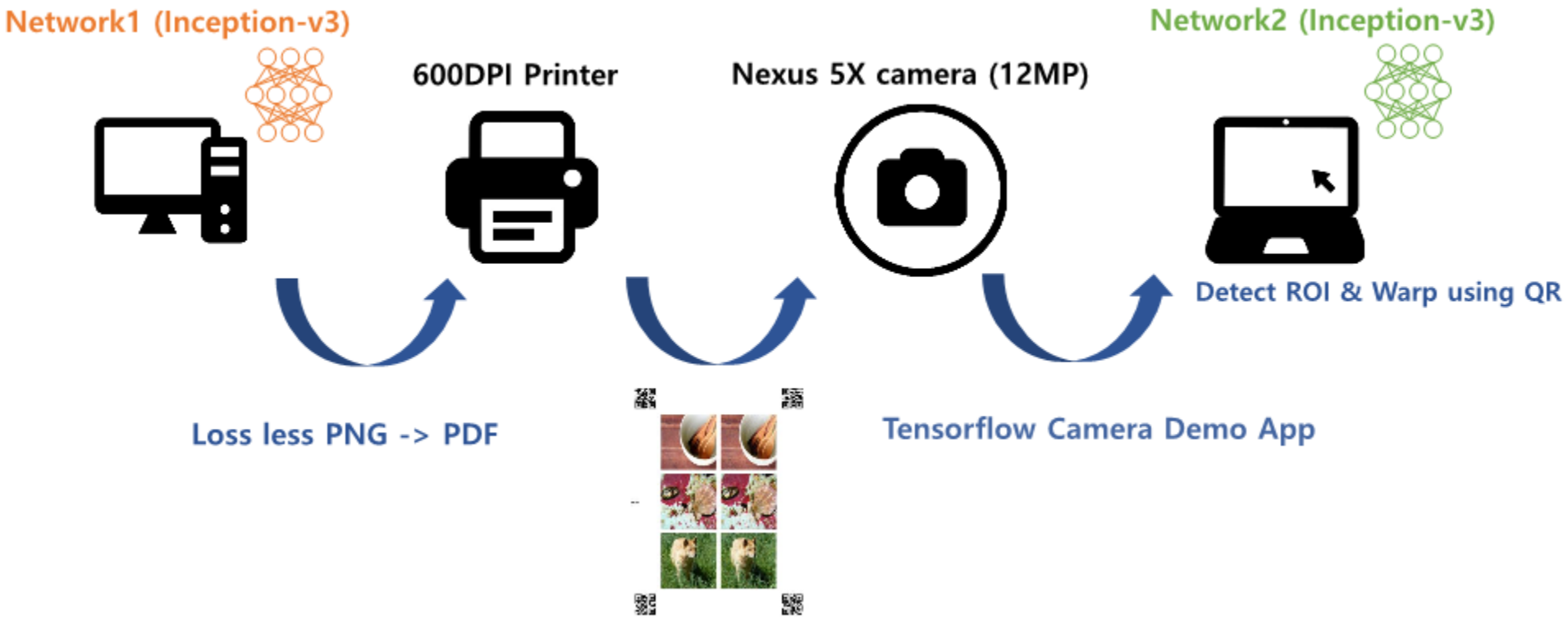
	VGG-F	CaffeNet	GoogLeNet	VGG-16	VGG-19	ResNet-152
VGG-F	**93.7%**	71.8%	48.4%	42.1%	42.1%	47.4 %
CaffeNet	74.0%	**93.3%**	47.7%	39.9%	39.9%	48.0%
GoogLeNet	46.2%	43.8%	**78.9%**	39.2%	39.8%	45.5%
VGG-16	63.4%	55.8%	56.5%	**78.3%**	73.1%	63.4%
VGG-19	64.0%	57.2%	53.6%	73.5%	**77.8%**	58.0%
ResNet-152	46.3%	46.3%	50.5%	47.0%	45.5%	**84.0%**

S Moosavi-Dezfooli et al. "Universal Adversarial Pertubation", CVPR 2017

Adversarial Attack in real world?



Adversarial Attack in real world?



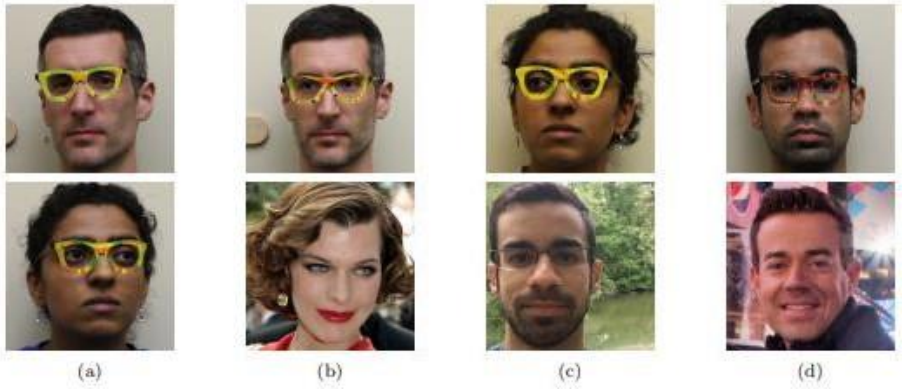
A. Kurakin et al. "Adversarial examples in the physical world", arXiv 2016

Adversarial Attack in real world?

Demo:

https://www.youtube.com/watch?v=zQ_uMenoBCk
https://www.youtube.com/watch?v=i1sp4X57TL4
https://www.youtube.com/watch?v=MIbFvK2S9g8

- **Real world에서도 AA이 가능**
- **Real world는 Black-box Attack**
- **Pertubation이 큰 AA가 잘 동작**
- **현실에서도 충분히 위협적**



accessorize to a crime real and stealthy attacks on state-of-the-art face recognition

Adversarial Attack in real world?

- 꼭 Adversarial Example이어야 할까?

https://www.facebook.com/photo/?fbid=3898548626850548&set=gm.4055581467838952

Adversarial Defense

- **Adversarial Training**
  - AA 이미지도 학습데이터로 이용

Training Data = **Original Image** + **Adversarial Image**

http://proceedings.mlr.press/v97/zhang19p/zhang19p.pdf
https://arxiv.org/pdf/1706.06083.pdf

Adversarial Defense

- **Adversarial Training**
  - AA 이미지도 학습데이터로 이용

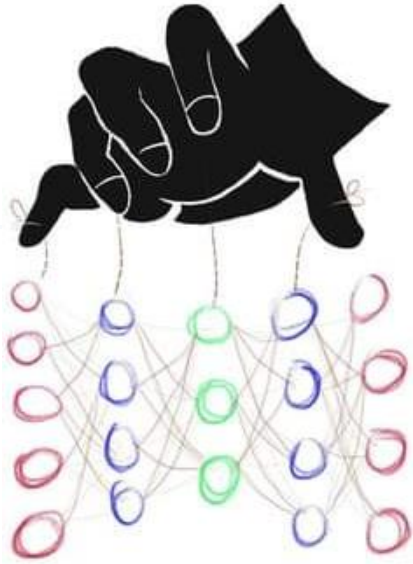
CIFAR10

	Simple	Wide	Simple	Wide	Simple	Wide	Simple	Wide
Natural	92.7%	95.2%	87.4%	90.3%	79.4%	87.3%	0.00357	0.00371
FGSM	27.5%	32.7%	90.9%	95.1%	51.7%	56.1%	0.0115	0.00557
PGD	0.8%	3.5%	0.0%	0.0%	43.7%	45.8%	1.11	0.0218
	(a) Standard training		(b) FGSM training		(c) PGD training		(d) Training Loss	

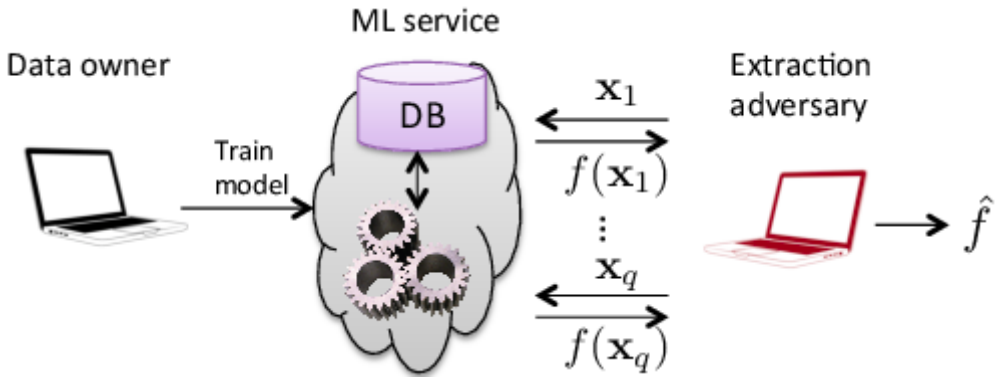
Defense	Defense type	Under which attack	Dataset	Distance	$\mathcal{A}_{\text{nat}}(f)$	$\mathcal{A}_{\text{rob}}(f)$
Buckman et al. (2018)	gradient mask	Athalye et al. (2018)	CIFAR10	0.031 (ℓ_∞)	-	0%
Ma et al. (2018)	gradient mask	Athalye et al. (2018)	CIFAR10	0.031 (ℓ_∞)	-	5%
Dhillon et al. (2018)	gradient mask	Athalye et al. (2018)	CIFAR10	0.031 (ℓ_∞)	-	0%
Song et al. (2018)	gradient mask	Athalye et al. (2018)	CIFAR10	0.031 (ℓ_∞)	-	9%
Na et al. (2017)	gradient mask	Athalye et al. (2018)	CIFAR10	0.015 (ℓ_∞)	-	15%
Wong et al. (2018)	robust opt.	FGSM^{20} (PGD)	CIFAR10	0.031 (ℓ_∞)	27.07%	23.54%
Madry et al. (2018)	robust opt.	FGSM^{20} (PGD)	CIFAR10	0.031 (ℓ_∞)	87.30%	**47.04%**
Zheng et al. (2016)	regularization	FGSM^{20} (PGD)	CIFAR10	0.031 (ℓ_∞)	94.64%	0.15%
Kurakin et al. (2017)	regularization	FGSM^{20} (PGD)	CIFAR10	0.031 (ℓ_∞)	85.25%	45.89%
Ross & Doshi-Velez (2017)	regularization	FGSM^{20} (PGD)	CIFAR10	0.031 (ℓ_∞)	95.34%	0%
TRADES ($1/\lambda = 1.0$)	regularization	FGSM^{20} (PGD)	CIFAR10	0.031 (ℓ_∞)	88.64%	49.14%
TRADES ($1/\lambda = 6.0$)	regularization	FGSM^{20} (PGD)	CIFAR10	0.031 (ℓ_∞)	84.92%	**56.61%**

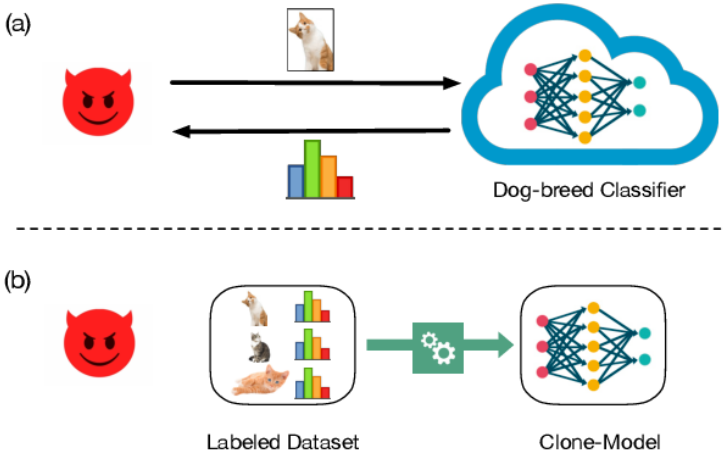
http://proceedings.mlr.press/v97/zhang19p/zhang19p.pdf
https://arxiv.org/pdf/1706.06083.pdf

Model Stealing



- 임의의 데이터를 API로 전송하여 모델을 유추하는 공격

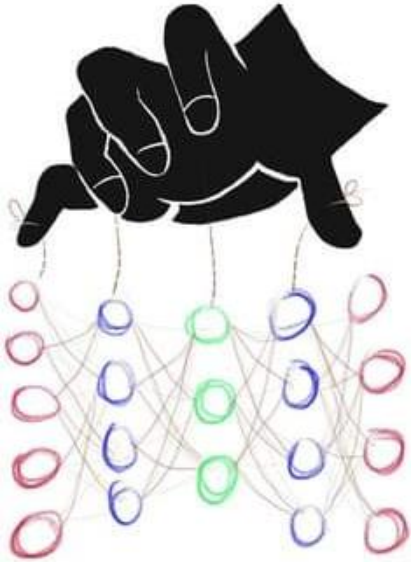




https://m.facebook.com/payatutechnologies/photos/blog-model-skewing-attacks-on-machine-learning-models-this-blog-covers-the-follo/1892459551141473/

S. Kariyappa, "Defending Against Model Stealing Attacks With Adaptive Misinformation", CVPR19

Model Stealing



- **임의의 데이터를 API로 전송하여 모델을 유추하는 공격**
- **However**
  - 무한정 API를 보낼 수 없다
  - 모델의 구조를 알 수 없다
  - 획득해야 하는 파라미터의 숫자가 너무 많다(BERT는 1억개)
- **대응방법**
  - API 숫자 제한
  - 앙상블 모델 병합 방식 사용
  - 일반적인 모델 사용 지양
    - CrossEntrophy 대신 MSE
    - Sigmoid 대신 Reverse Sigmoid

https://m.facebook.com/payatutechnologies/photos/blog-model-skewing-attacks-on-machine-learning-models-this-blog-covers-the-follo/1892459551141473/

S. Kariyappa, "Defending Against Model Stealing Attacks With Adaptive Misinformation", CVPR19

Video Recognition

Spatio-temporal, CNN+RNN, 3D CNN

Video Recognition

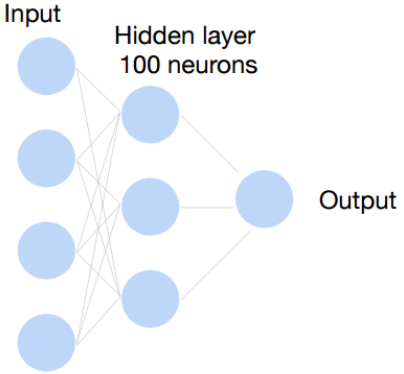
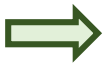


Image Classification

Cat?
Dog?



Video Classification

Kick?
Backflipping?

https://bleedai.com/human-activity-recognition-using-tensorflow-cnn-lstm/

Video Recognition

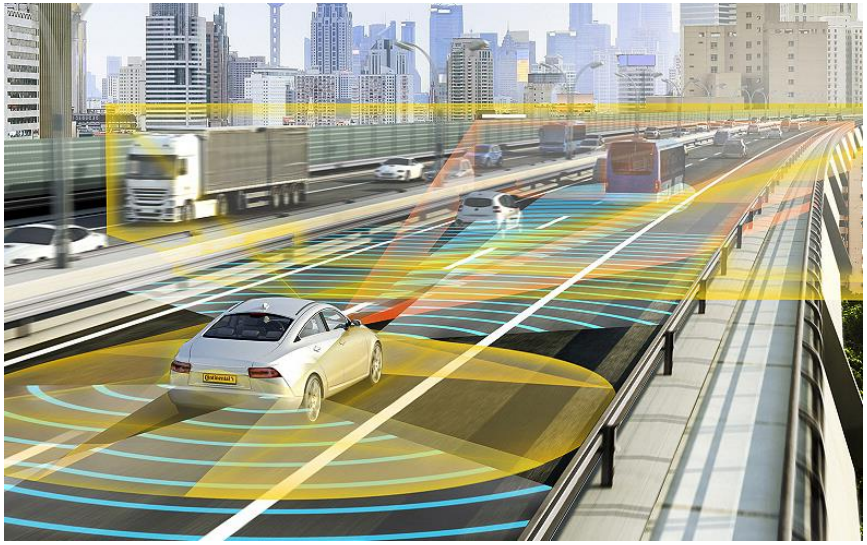
- **Video is a sequence of multiple image (frame)**

Video Classification

Kick?
Backflipping?

https://bleedai.com/human-activity-recognition-using-tensorflow-cnn-lstm/

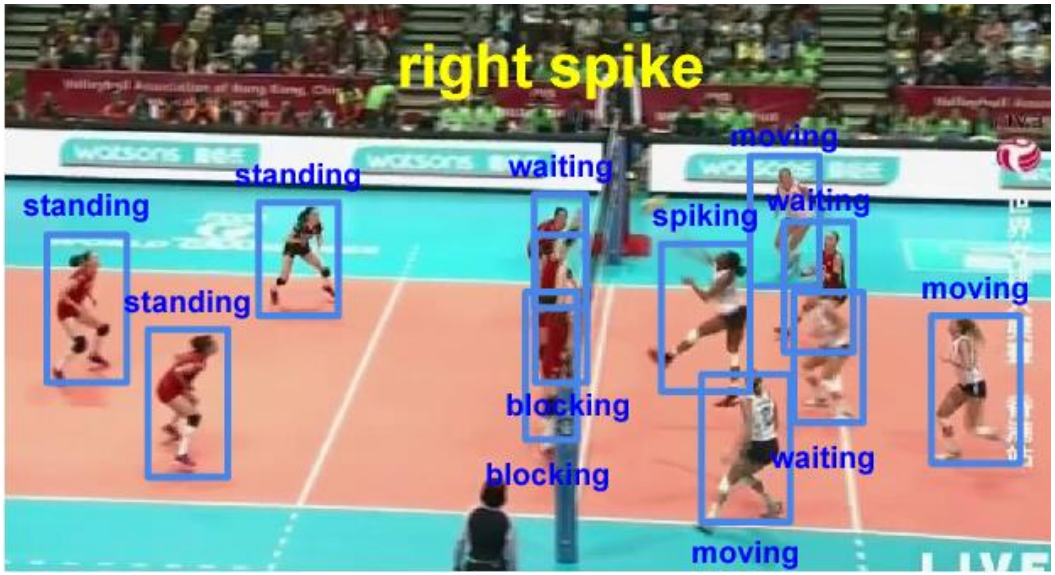
Videos are Everywhere

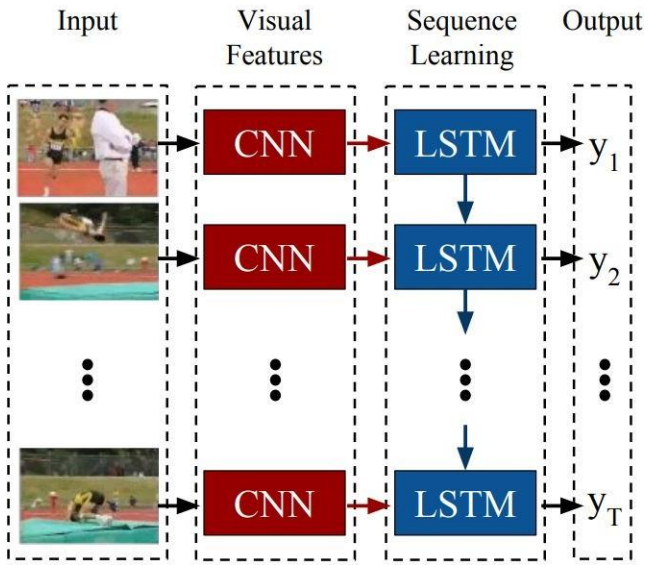


node=view
p-for-2022
vice-robot/
https://www.aitimes.kr/news/articleView.html?idxno=92372
https://www.outsourcing.co.kr/news/articleView.html?idxno=81322
https://news.kotra.or.kr/user/globalBbs/kotranews/782/globalBbsDataView.do?setIdx=242&dataIdx=179934
https://m.post.naver.com/viewer/postView.nhn?volumeNo=28947306&memberNo=162261848
https://www.aitimes.kr/news/articleView.html?idxno=9729

Video Recognition

- Video Understanding
- Video Captioning
- Video QA
- VR



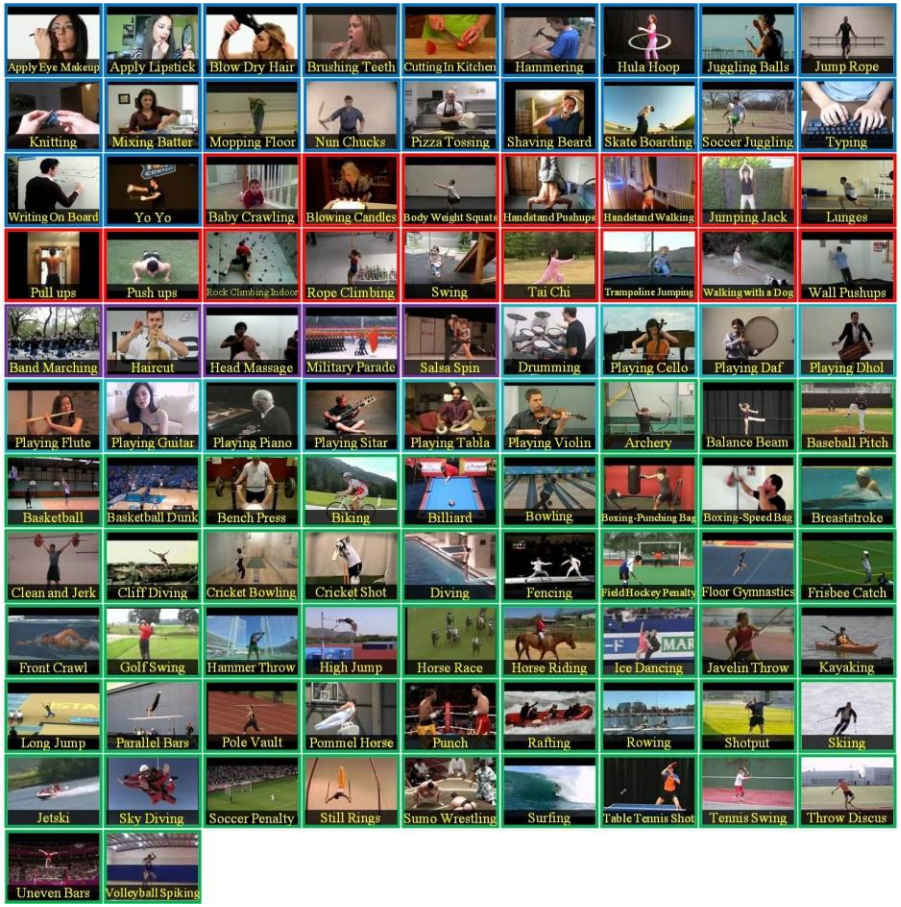


Timur M. Bagautdlnov, "Social Scene Understanding: End-to-End Multi-person Action Localization and Collective Activity Recognition", CVPR 2017
Donahue et al., "Long-term Recurrent Convolutional Networks for Visual Recognition and Description", CVPR, 2015

Video Dataset

- **Video Classification**
- **Atomic Actions**
- **Video Retrieval, …**

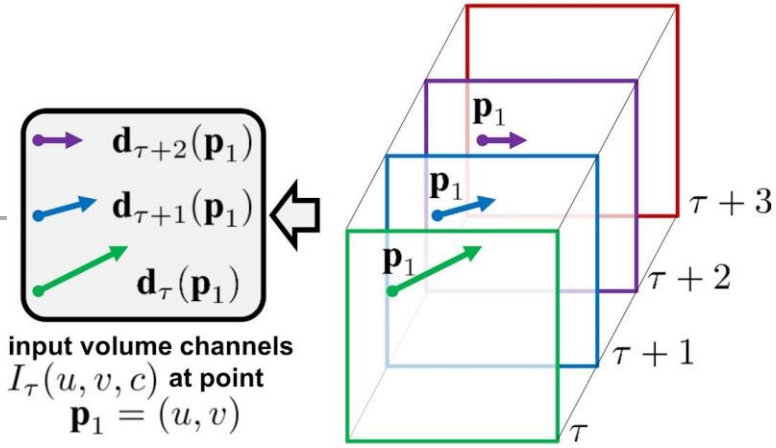
- **Video Classification**
  - UCF101: 101 actions
  - Sports-1M: 487 actions
  - YouTube 8M: 3,862 classes

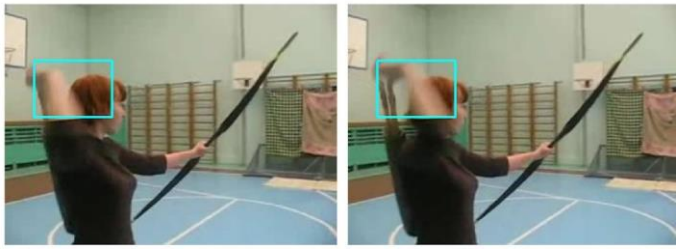


Khurram Soomro, Amir Roshan Zamir and Mubarak Shah, UCF101: A Dataset of 101 Human Action Classes From Videos in The Wild, CRCV-TR-12-01, 2012
Karpathy et al., Large-scale Video Classification with Convolutional Neural Networks, CVPR 2014
Karpathy et al., Large-scale Video Classification with Convolutional Neural Networks, CVPR 2014

Video Model

- **Old Method: Optical Flow**



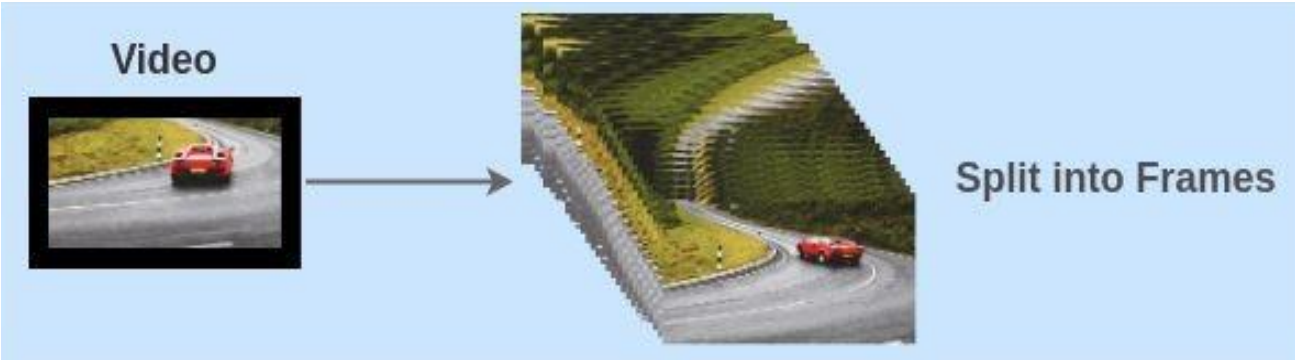


Simonyan and Zisserman, Two-Stream Convolutional Networks for Action Recognition in Videos, NeurIPS 2014

Video Model Approach

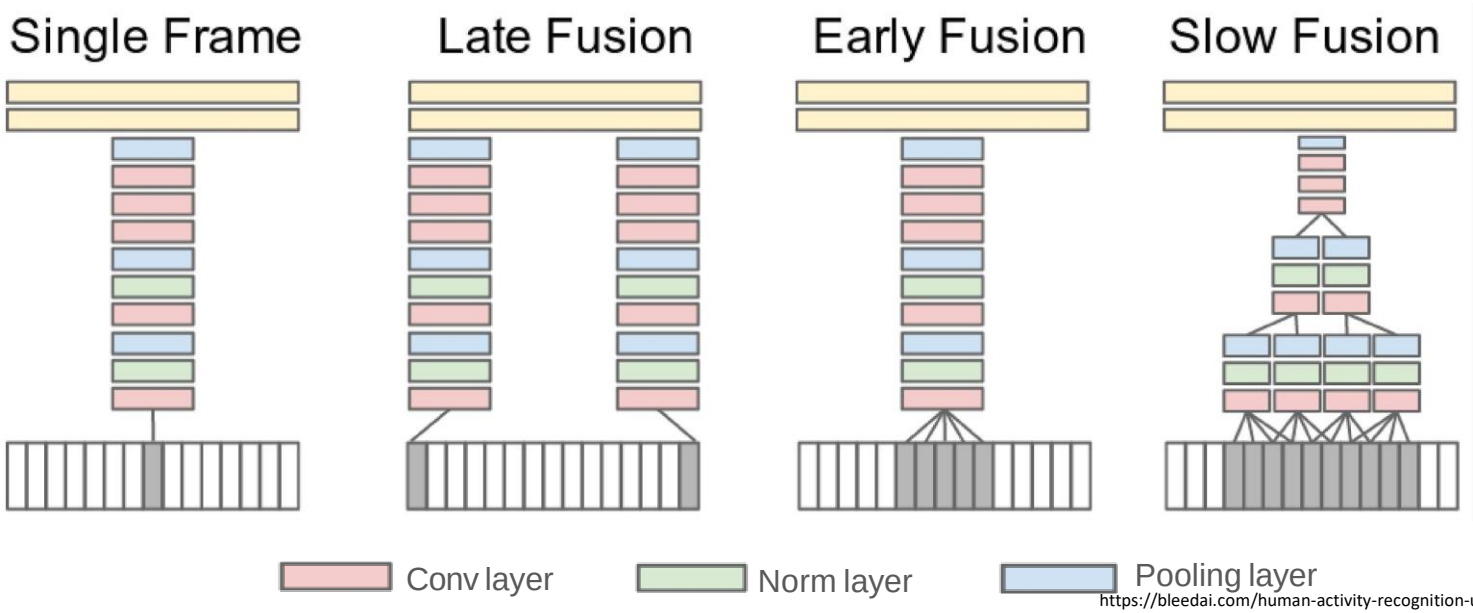
- **Series of frames**
- **Frames: spatial information (2D CNN)**
- **Series: temporal information (1D CNN, LSTM)**

- **Computation power**
- **Complexity of the data**
- **Difficulty in handling the data**



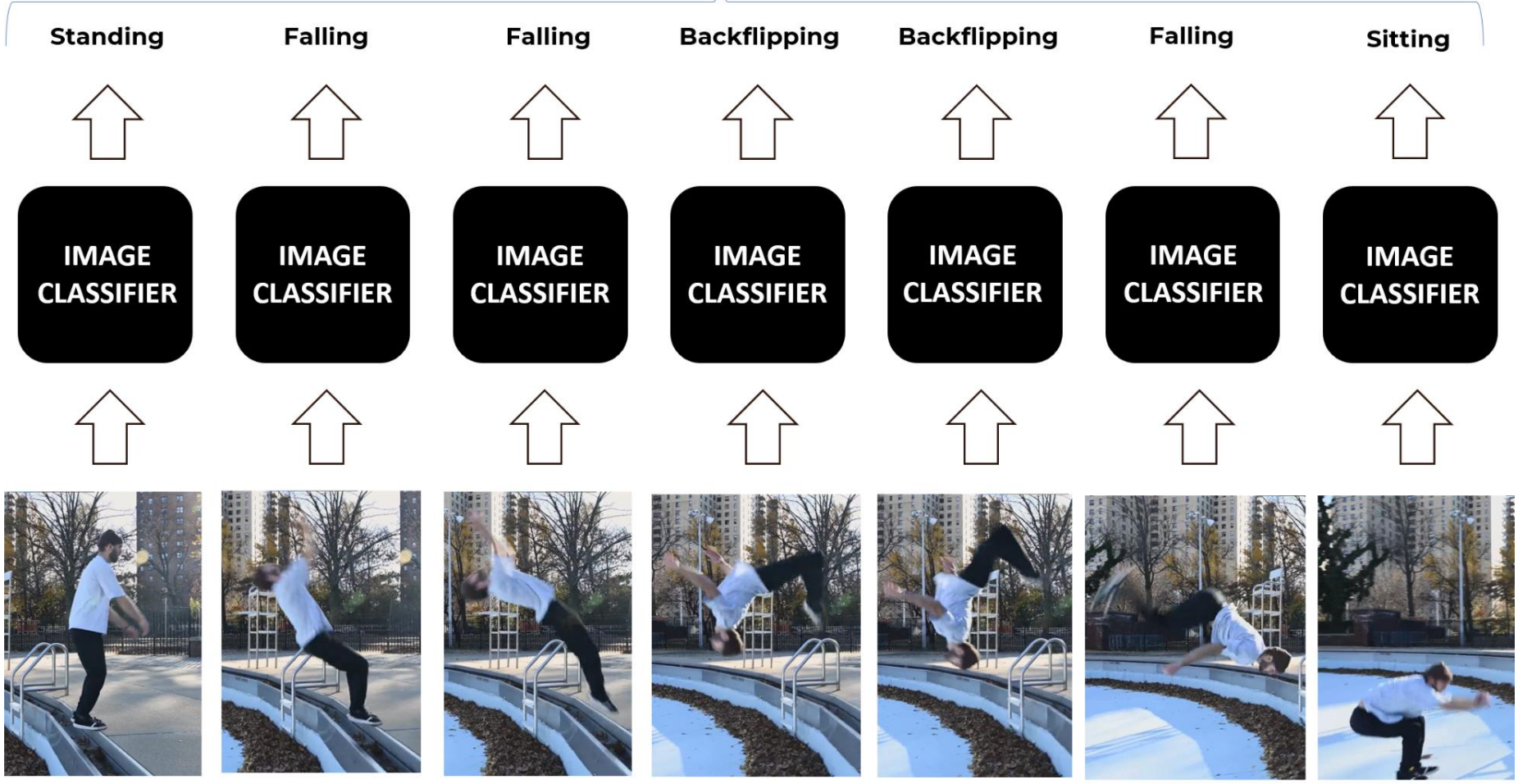
Video Model Approach

1) Single-Frame Classification
2) Late Fusion
3) Early Fusion
4) Using 3D CNN's (aka. Slow Fusion)
5) Using Pose Detection and LSTM
6) CNN + LSTM



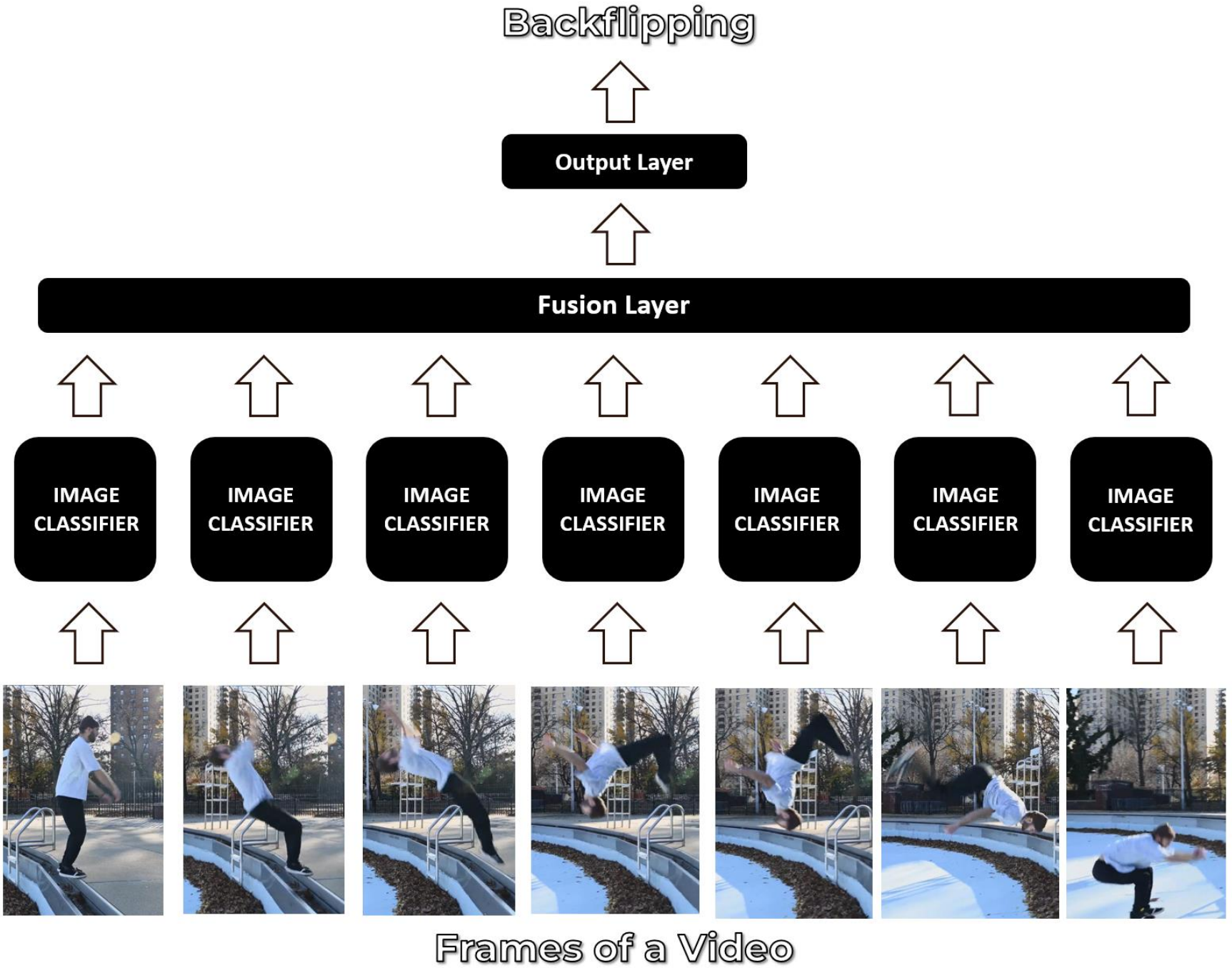
https://bleedai.com/human-activity-recognition-using-tensorflow-cnn-lstm/

Average Of prediction probabilities



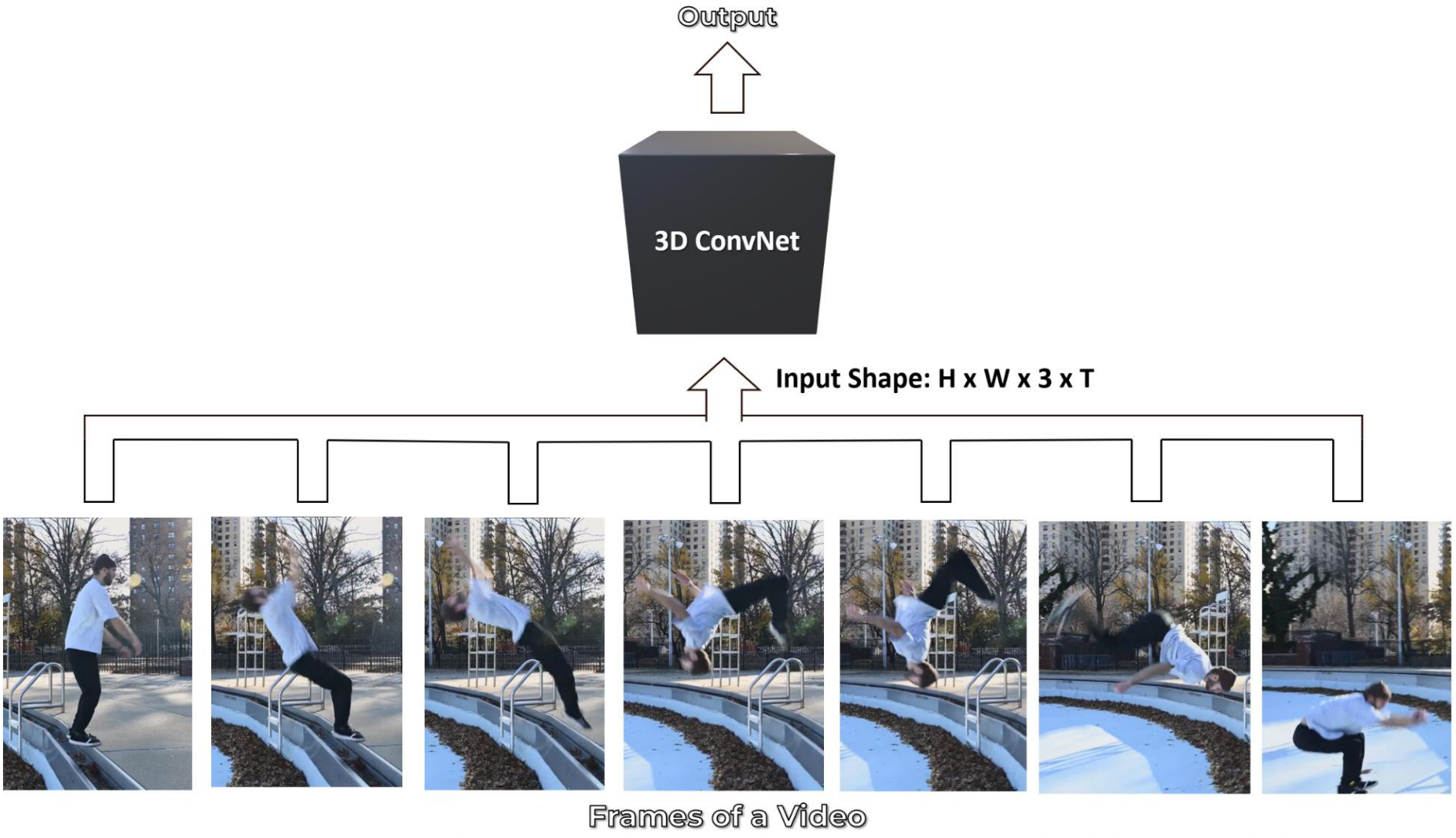
Conv layer Norm layer Pooling layer

https://bleedai.com/human-activity-recognition-using-tensorflow-cnn-lstm/


Backflipping
Output Layer
Fusion Layer
IMAGE CLASSIFIER
IMAGE CLASSIFIER
IMAGE CLASSIFIER
IMAGE CLASSIFIER
IMAGE CLASSIFIER
IMAGE CLASSIFIER
IMAGE CLASSIFIER
Frames of a Video

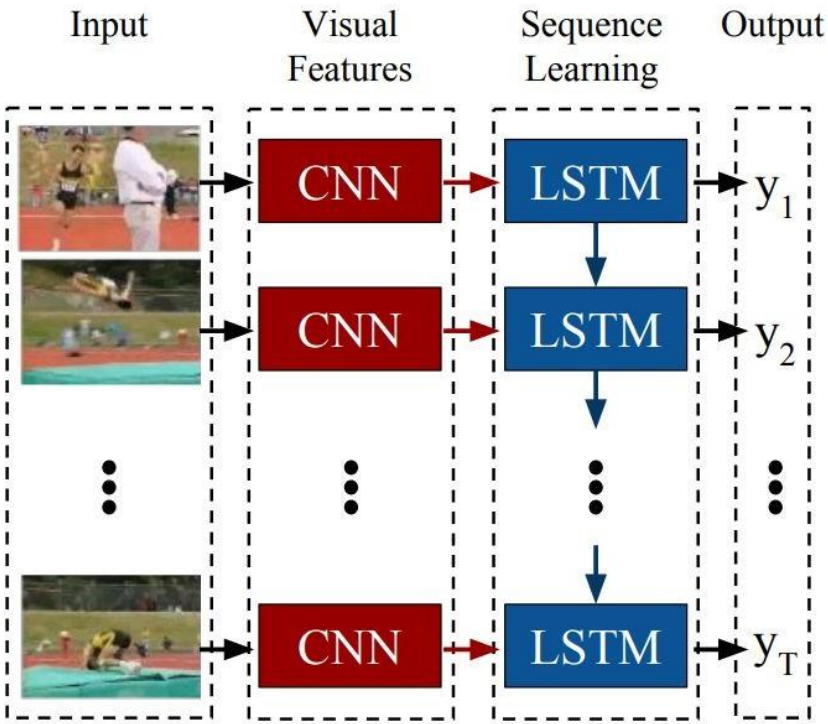
-lstm/

Video Model Approach



https://bleedai.com/human-activity-recognition-using-tensorflow-cnn-lstm/

Video Model Approach



https://bleedai.com/human-activity-recognition-using-tensorflow-cnn-lstm/

Video Classification 실습

- **Code: https://github.com/airobotlab/lecture_4_video_recognition**

- **run Video_Classification_colab.ipynb on colab**

Image Search

Advanced Face Search&Recommendation with faiss

Image Search

- **Code: https://github.com/airobotlab/lecture_2_advanced_face_search_system**
- **DeepFace-based Face recommendation practice**

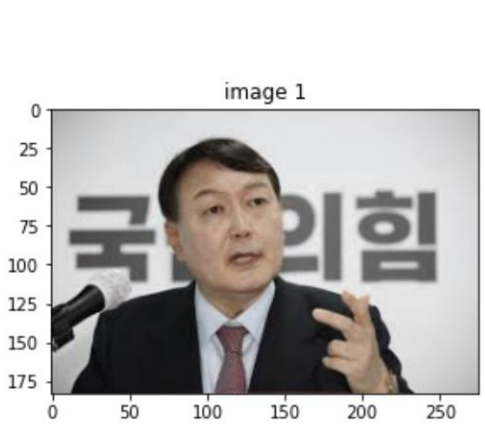
- **run 06_04_search_face.ipynb on colab**

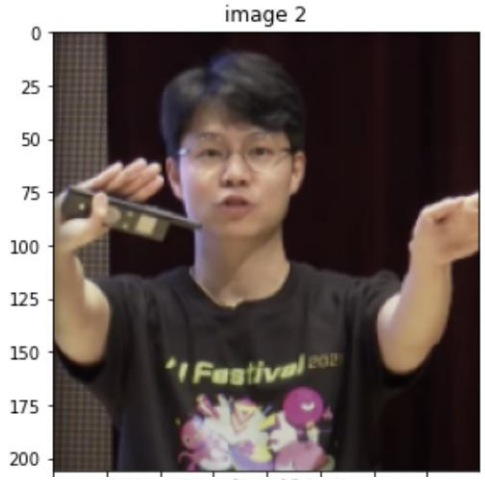
- **Face Detection&Recognition 실습**
- **Faiss 검색시스템 기초 예시**
- **Face search system**
- **K-means clustering**
- **Clustering practice**
- **t-SNE plot**

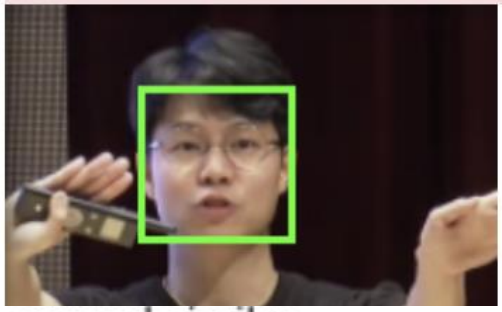
Image Search


```
{'age': 33,
 'dominant_emotion': 'sad',
 'dominant_race': 'asian',
 'emotion': {'angry': 1.867866888642311,
             'disgust': 5.353508747418 8e-07,
             'fear': 9.24883857369423,
             'happy': 0.00040961122067528777,
             'neutral': 43.256571888923645,
             'sad': 45.61874270439148,
             'surprise': 0.0075654141255654395},
 'gender': 'Man',
 'race': {'asian': 99.7871994972229,
          'black': 2.502049198938039e-05,
          'indian': 0.11423757532611489,
          'latino hispanic': 0.08078439277596772,
          'middle eastern': 1.35076696494707e-06,
          'white': 0.017756159650161862},
 'region': {'h': 63, 'w': 63, 'x': 54, 'y': 33}}
```

```
{'detector_backend': 'opencv',
 'distance': 0.3431723621100845,
 'model': 'VGG-Face',
 'similarity_metric': 'cosine',
 'threshold': 0.4,
 'verified': True}
```





Ima

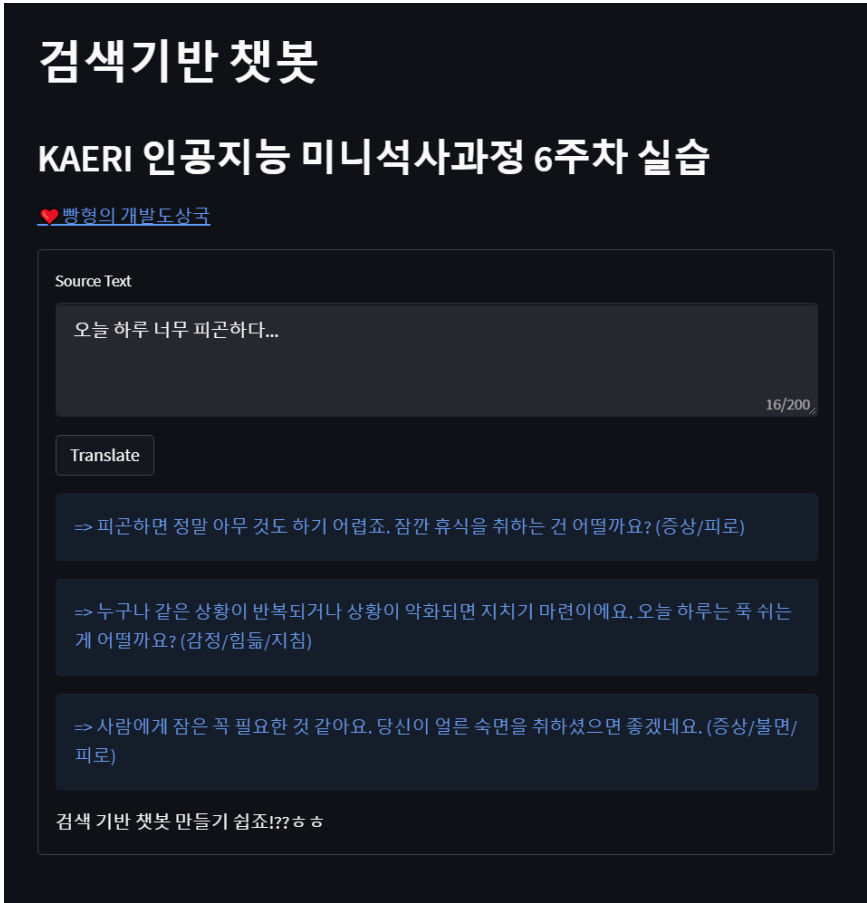
Text Search

Advanced Face Search&Recommendation with faiss

Text Search

- Code: https://github.com/airobotlab/lecture_3_advanced_text_search_system
- DeepFace-based Face recommendation practice

- run 0605_nlp_search_chatbot.ipynb on colab

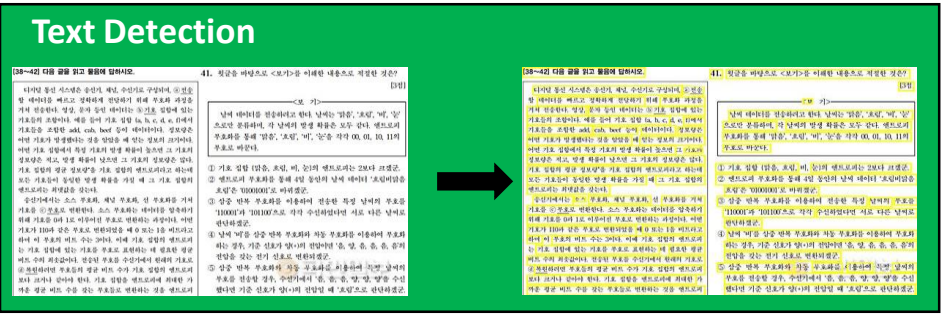


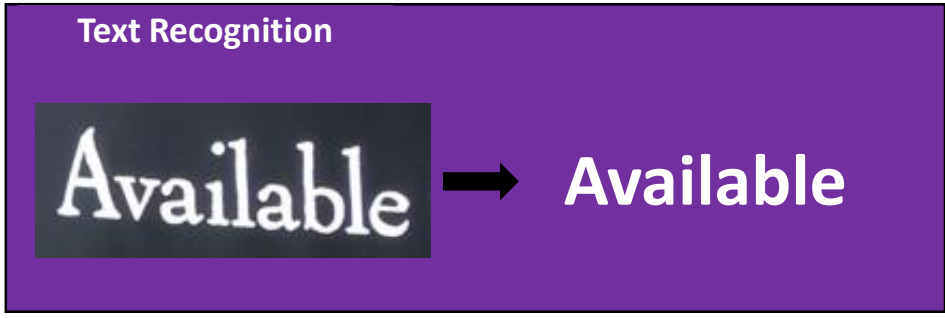
OCR

Optical Character Recognition

OCR

Image → Pre-Processing → Text Detection → Mid-Processing → Text Recognition → Output





OCR

Dataset Images

Ground Truth (text files)

```
11, 15, 42, 28, "How"
41, 16, 61, 28, "to"
8, 33, 36, 46, "Find"
41, 32, 64, 46, "the"
11, 50, 61, 65, "Perfect"
16, 69, 56, 82, "HDTV"
```

```
22 249 113 286 "The"
142 249 287 286 "Photo"
326 245 620 297 "Specialists"
```

```
0, 1, 177, 32, "\"HIGHER"
11, 35, 182, 63, "SAVINGS"
12, 67, 183, 103, "RATES\""
50, 114, 56, 120, "A"
60, 114, 91, 120, "reward"
96, 114, 108, 120, "for"
112, 116, 126, 120, "our"
130, 114, 164, 120, "current"
...
```

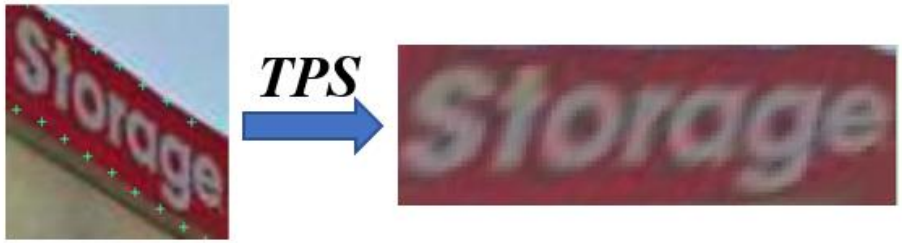
Ground Truth Visualisation

OCR

Irregular text recognition methods

Multi-directional feature-based method[1]

Segmentation-based method[2]



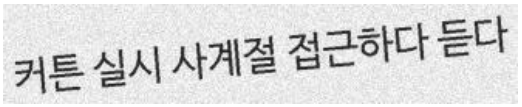
Rectification-based method[3]

2D-attention based method[4]

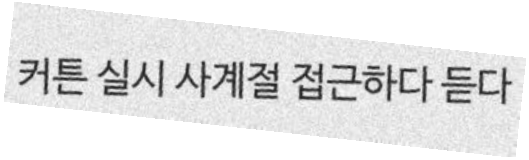
OCR

Image → Pre-Processing → Text Detection → Mid-Processing → Text Recognition → Output

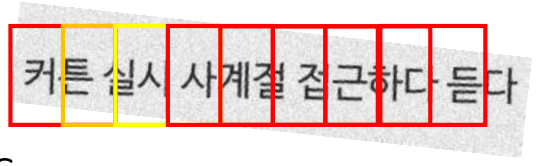
Text Recognition

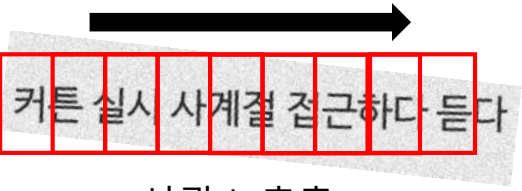


변환
TPS



특징추출
Resnet/VGG
Efficientnet



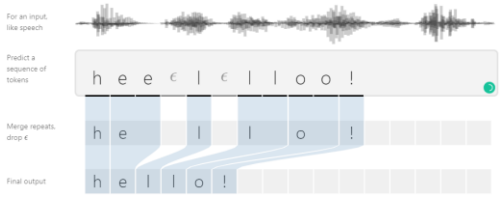


시퀀스 추출
BiLSTM

Prediction
CTC/Attn

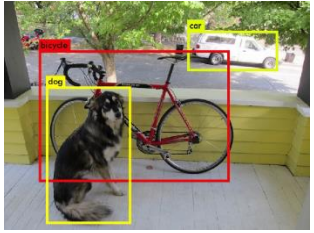
a-----v--a-i-l-a-bb-l-ee---

available

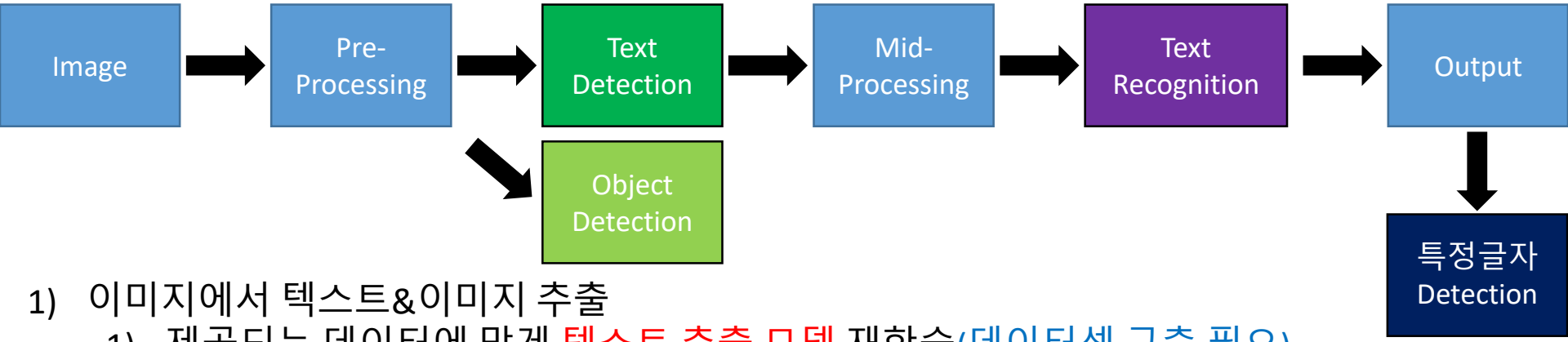


Connectionist Temporal Classification 예시

OCR



→ Available



1) 이미지에서 텍스트&이미지 추출
   1) 제공되는 데이터에 맞게 텍스트 추출 모델 재학습(데이터셋 구축 필요)
   2) 이미지 추출 모델 구축 및 학습(데이터셋 구축 필요)
      1) 물체검출 최고성능 알고리즘 구현 및 학습

2) OCR 구현(한국어)
   1) 제공되는 데이터의 텍스트 글꼴 예측
   2) 해당 글꼴 학습데이터 생성
   3) 실제 데이터 학습데이터 구축
   4) 학습 12가지 케이스, best 선정
   5) 영어 : 26자, 한글 11,172자(가~힣)

- 한국어/영어 Text recognition 모델 학습을 위한 학습데이터 생성기술

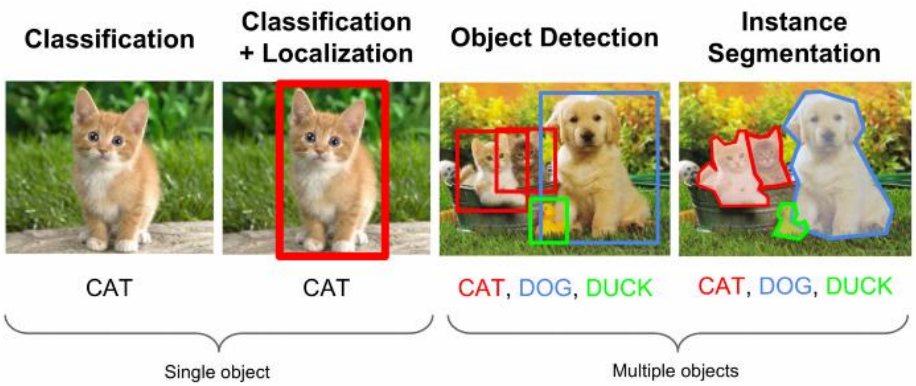
3) 특정글자 블러처리
   1) 특정 글자 위치 찾기
   2) 숫자는 쉽다(10개), 한글은 11,172 가지 CASE

Project

Image/NLP/Speach/Time series, ..

Project

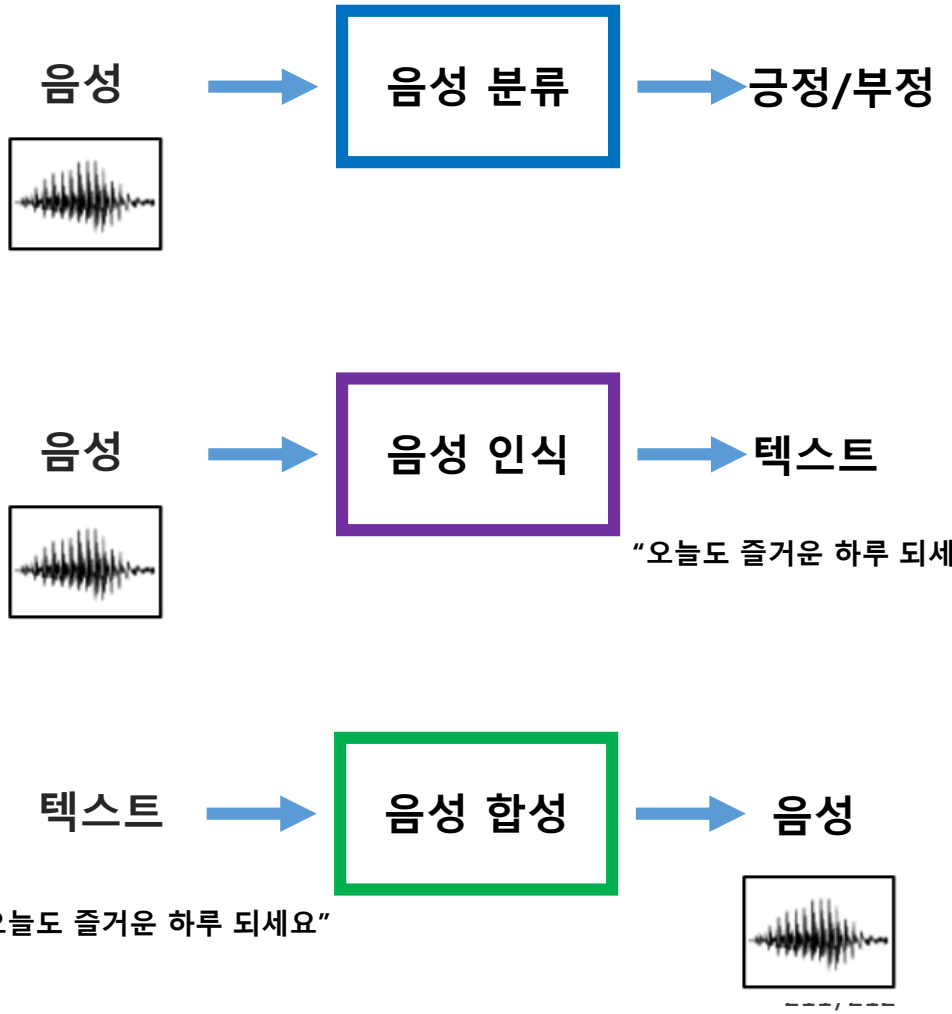
- 1) 보유한 데이터는?
  - 데이터는 어떤 종류인가?
    - 이미지?
    - 음성?
    - 텍스트?
  - Label은 어떤 종류인가? (이미지)
    - 분류?
    - 회기?
    - Object detection?
    - Segmentation?
    - OCR?



https://medium.datadriveninvestor.com/deep-learning-for-image-segmentation-d10d19131113

Project

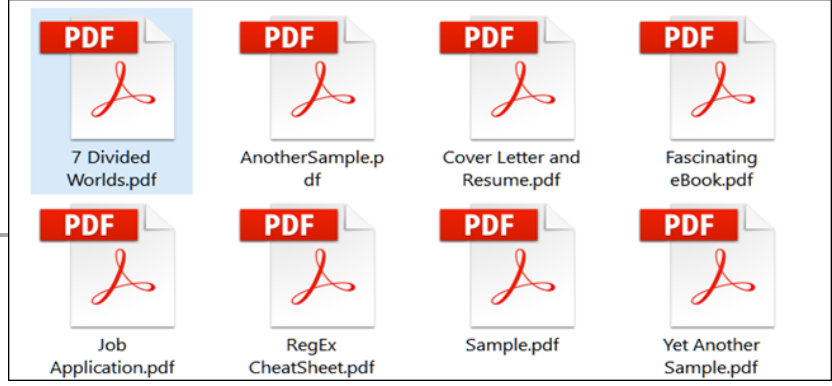
- 2) 보유한 데이터는?
  - 데이터는 어떤 종류인가?
    - 이미지?
    - 음성?
    - 텍스트?
  - Label은 어떤 종류인가? (음성)
    - 음성 분류?
    - 음성 인식?
    - 음성 합성?



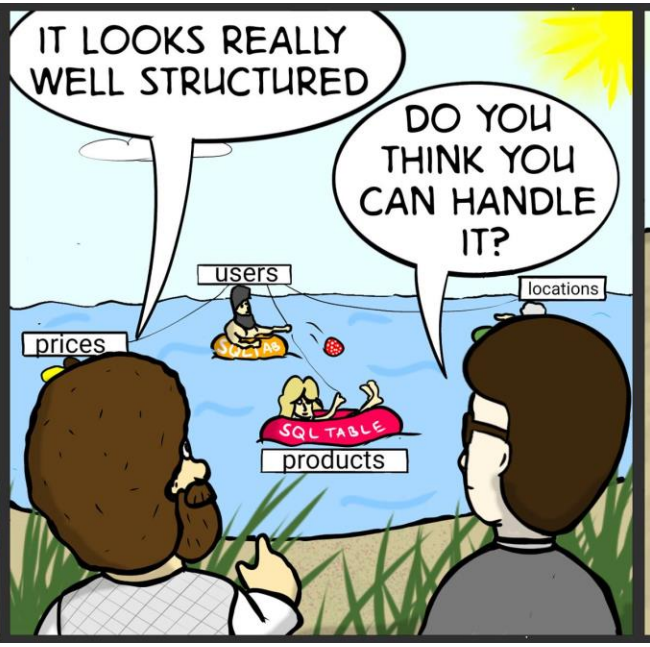
Project

- 3) 보유한 데이터는?
  - 데이터는 어떤 종류인가?
    - 이미지?
    - 음성?
    - 텍스트?
  - Label은 어떤 종류인가? (텍스트)
    - 분류?
    - 번역/요약/챗봇?
    - 생성?

Project

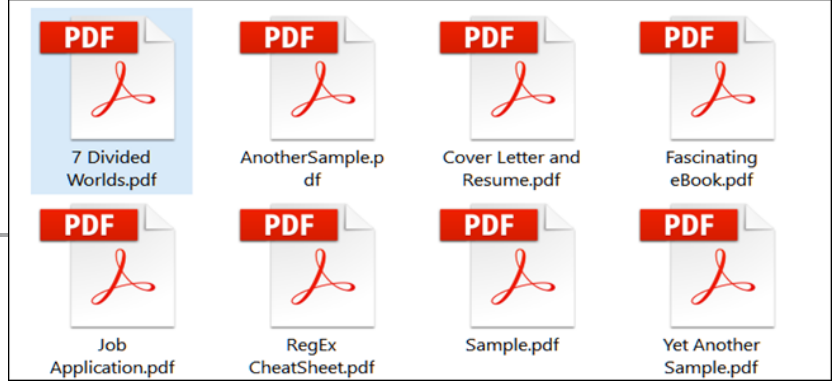


- 3) 우리사가 보유한 데이터는?
  - 데이터가 충분한가?
    - 서버에 가득 쌓여는 있다. 한 1000개?
    - **PDF문서는 데이터가 아니다!!** 파싱이 되어있는가? 안됐다면 없는 거다!

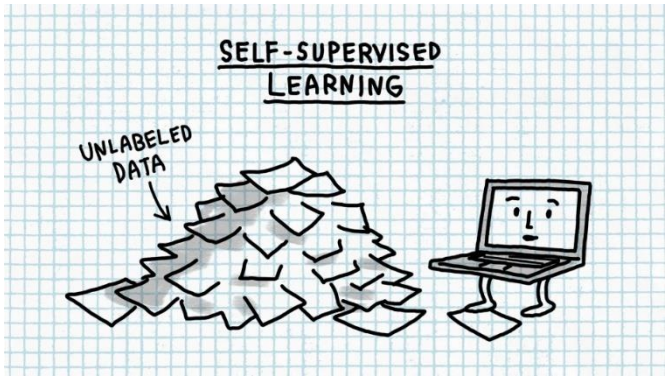


http://choesin.com/pdf-%ED%8C%8C%EC%9D%BC%EC%9D%B4%EB%9E%80-%EB%AC%B4%EC%97%87%EC%9D%B4%EB%A9%B0-%EC%96%B4%EB%96%BB%EA%B2%8C-%EC%97%B4-%EC%88%98-%EC%9E%88%EC%8A%B5%EB%8B%88%EA%B9%8C
https://www.facebook.com/tddcomics/photos/a.129014105680514/306002297981693/

Project



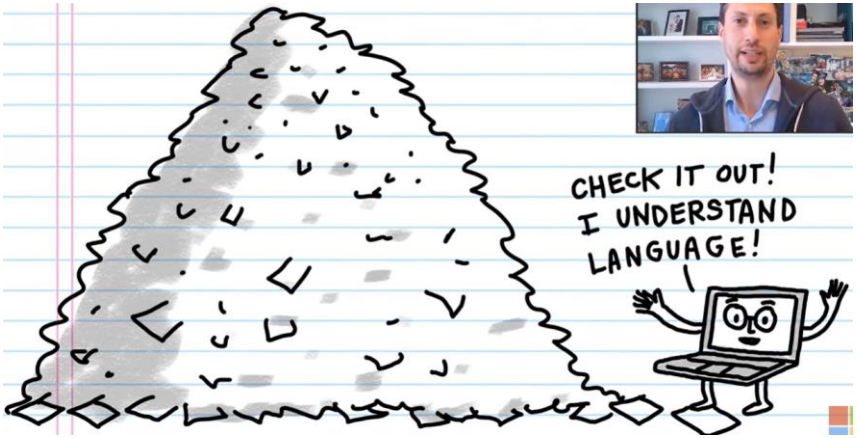
- 3) 우리사가 보유한 데이터는?
  - 데이터가 충분한가?
    - 서버에 가득 쌓여는 있다. 한 1000개?
    - **PDF문서는 데이터가 아니다!!** 파싱이 되어있는가? 안됐다면 없는 거다!
  - Label이 있는가?
    - 없다. 그 데이터는 레이블이 없는 데이터다. 없다!



http://choesin.com/pdf-%ED%8C%8C%EC%9D%BC%EC%9D%B4%EB%9E%80-%EB%AC%B4%EC%97%87%EC%9D%B4%EB%A9%B0-%EC%96%B4%EB%96%BB%EA%B2%8C-%EC%97%B4-%EC%88%98-%EC%9E%88%EC%8A%B5%EB%8B%88%EA%B9%8C

Project

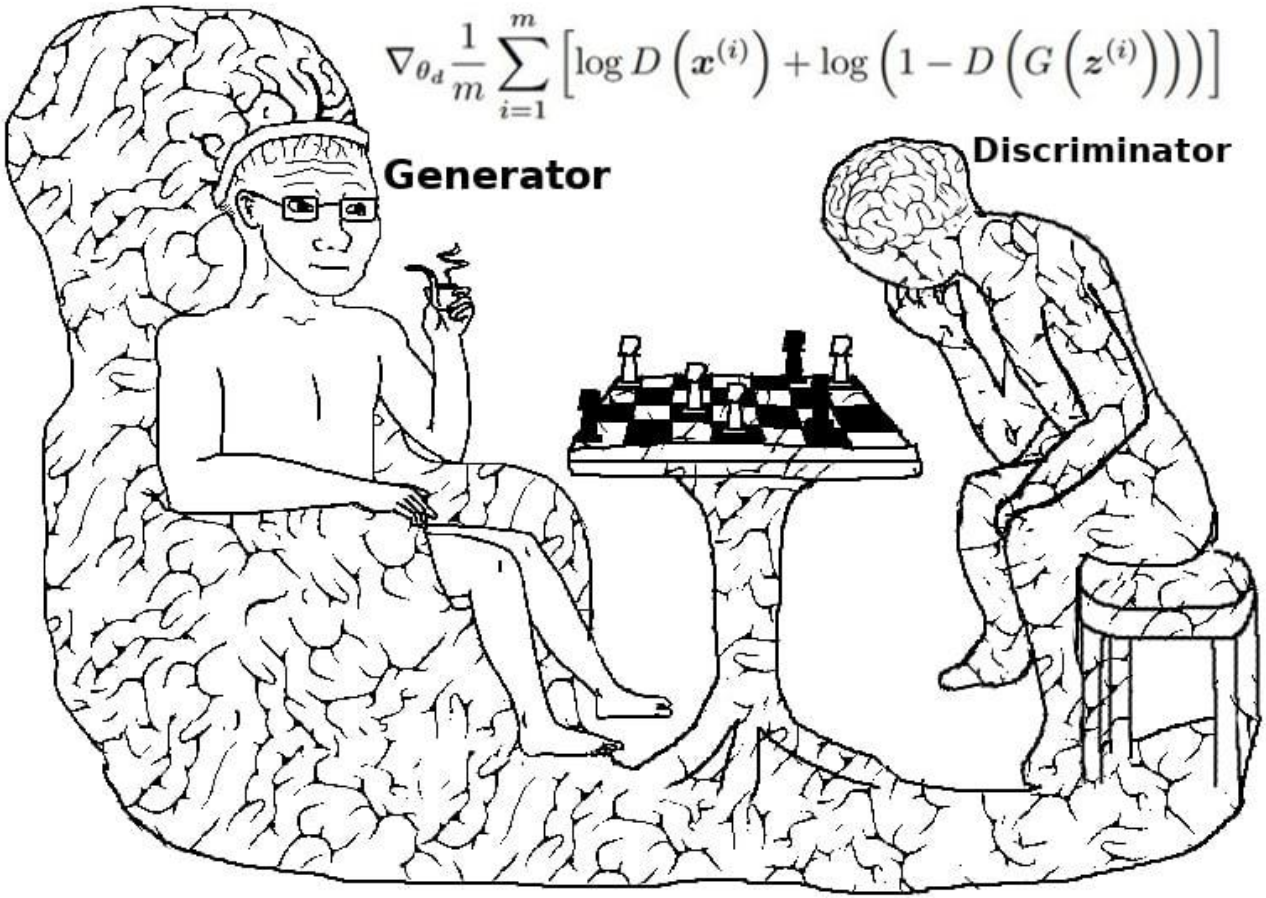
- 4) 우리사가 보유한 데이터는?
  - Label이 있는가?
    - 없다. 그 데이터는 레이블이 없는 데이터다. 없다!
    - Label이 없어도 학습할 수 있는 Semi/Self-Supervised learning이 있다는데??
      - Not possible for application



https://www.youtube.com/watch?v=8LZZg7YaZKE
https://blogs.microsoft.com/ai/openai-azure-supercomputer/
https://www.facebook.com/photo/?fbid=1965963656885547&set=gm.4475043012559460

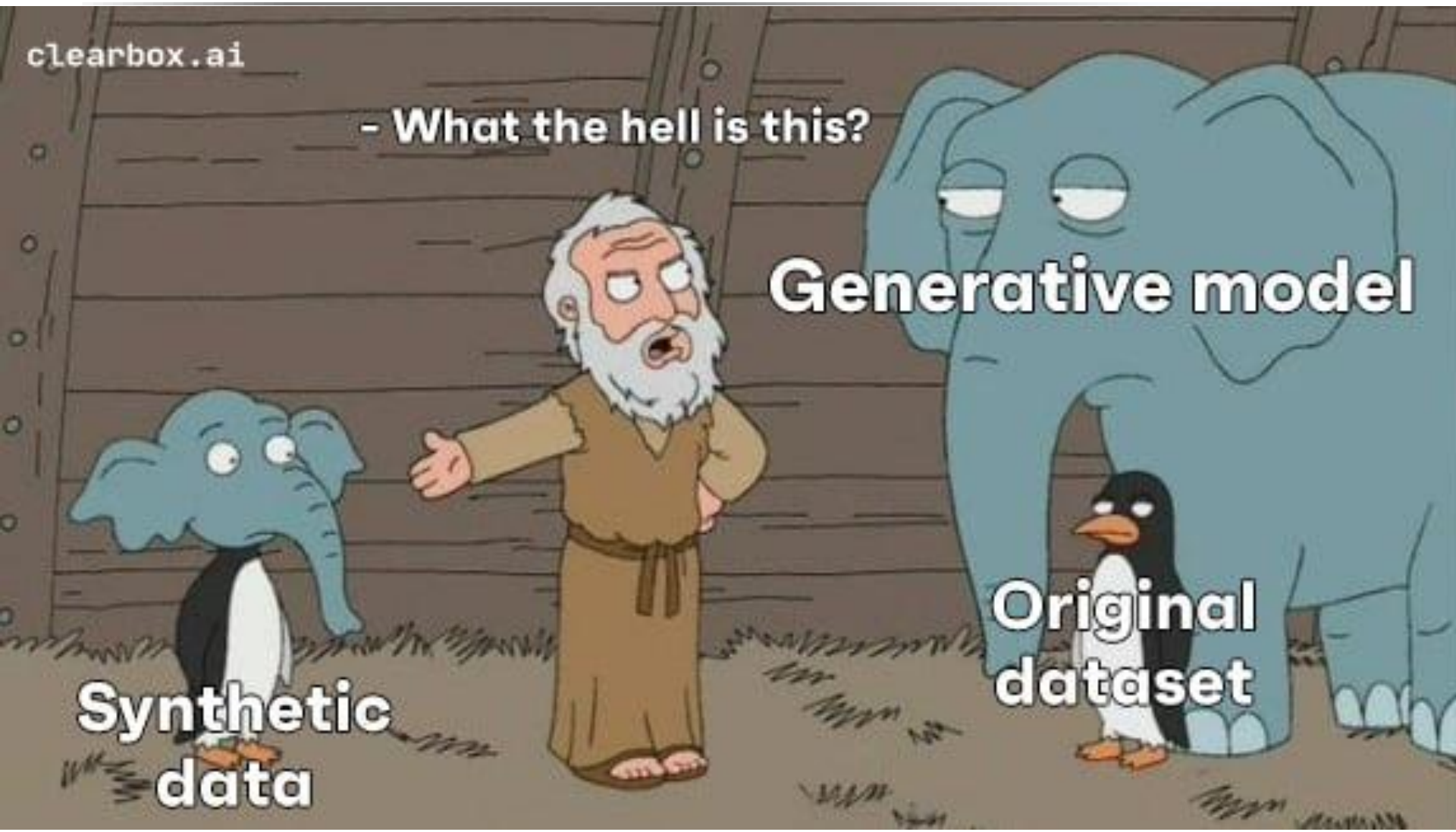
Project

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right]$$



https://www.facebook.com/photo?fbid=457118409623694 8&set=gm.4611557665574660

Project



https://www.facebook.com/photo/?fbid=1022513908713458 3&set=gm.4710931342303958

Project

- **Image??**
- **Text??**
- **Sound??**
- **Time series??**

Reference

- http://dsba.korea.ac.kr/seminar/?mod=document&uid=1784
- https://jihyeonryu.github.io/2021-04-02-survey-paper1/
- https://arxiv.org/pdf/2010.11929.pdf
- CS540, University of Wisconsin-Madison, Spring 2022
- https://pages.cs.wisc.edu/~sharonli/courses/cs540_spring2022/schedule.html
- https://www.slideshare.net/saurabhkaushikin/explainable-ai-xai-a-perspective?qid=aafedda6-8b8e-40db-9d8b-efbf480d57d3&v=&b=&from_search=7
- https://www.slideshare.net/MansourSaffarMehrjar/an-introduction-to-xai-towards-trusting-your-ml-models?qid=aafedda6-8b8e-40db-9d8b-efbf480d57d3&v=&b=&from_search=12
- 적대적 공격 및 방어 연구 소개, ETRI 황중언
- http://cs231n.stanford.edu/slides/2022/
- https://www.youtube.com/watch?v=HZg9E1sGcSM
- CS 231n, Adversarial Examples and Adversarial Training
- https://bleedai.com/human-activity-recognition-using-tensorflow-cnn-lstm/
- CS231N, Video Understanding

감사합니다