CNN Implementation

2022.08.25 / 이승연

Contents

Speed of Convergence

Speed of Convergence

Strategies to Handle Speed of Convergence

1. Momentum term

2. Activation Function

3. Weight Initialization

4. Learning Rate

5. Batch Normalization (BN)

Speed of Convergence

Strategies to Handle Speed of Convergence

1. Momentum term

2. Activation Function

3. **Weight Initialization**

4. **Learning Rate**

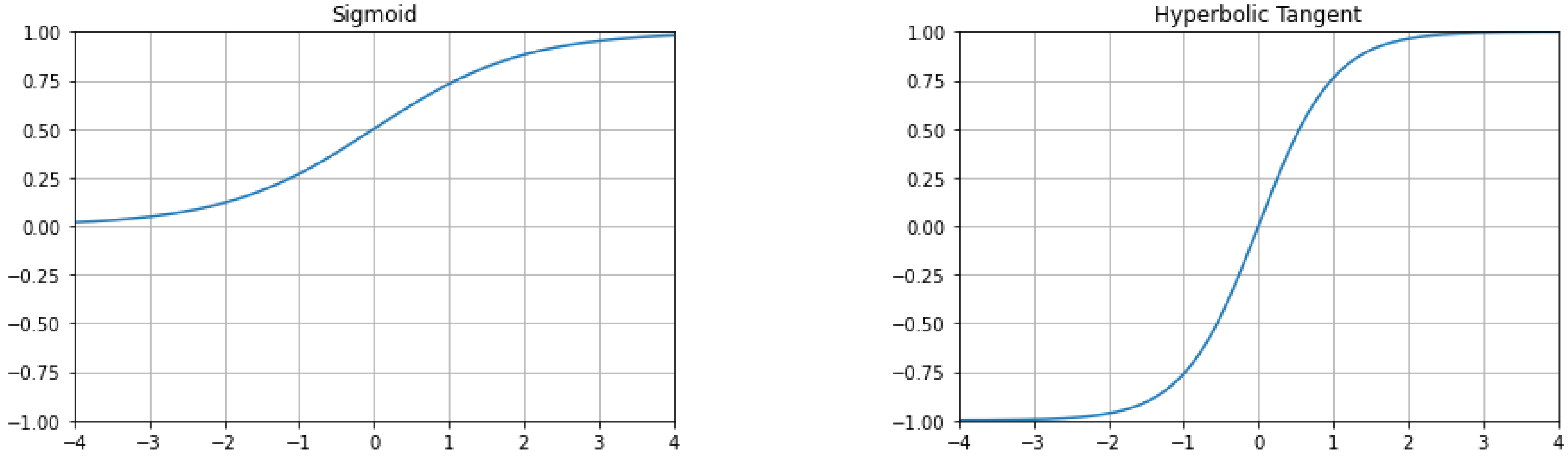
5. **Batch Normalization (BN)**

Activation Functions

Activation Function and Activation Values

Traditional Activation Function Sigmoid & Tanh Function

Sigmoid & Tanh Function

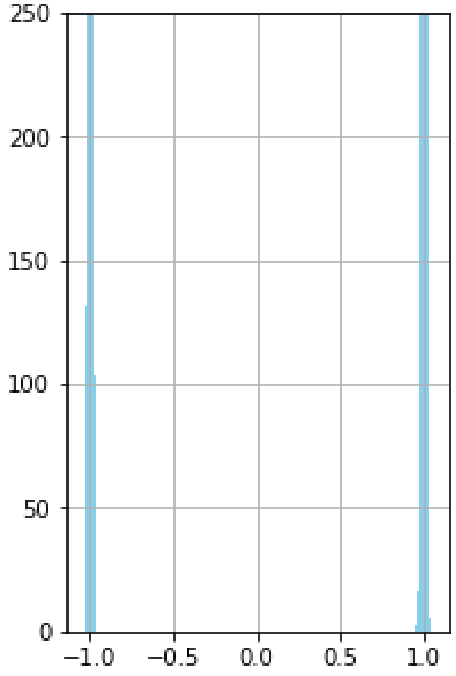


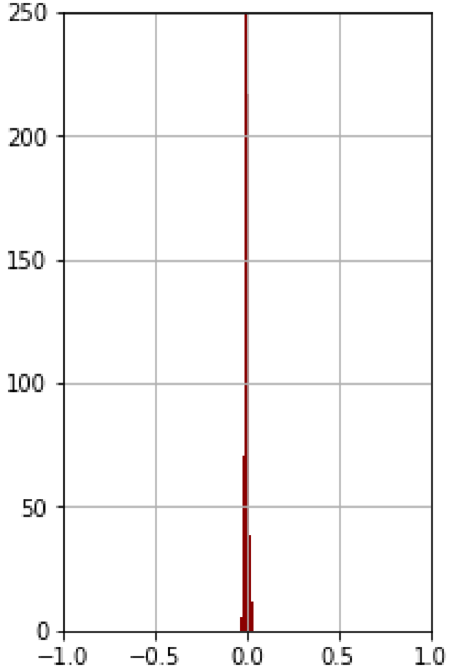
Activation Functions

Cases of Activation Value

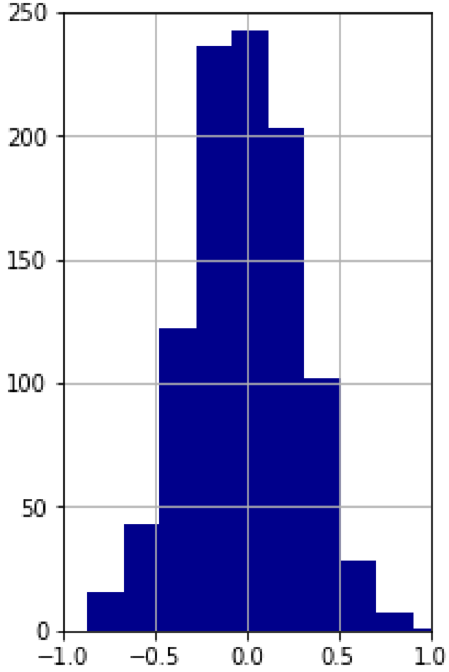
Most of activation values are either -1 or 1 only

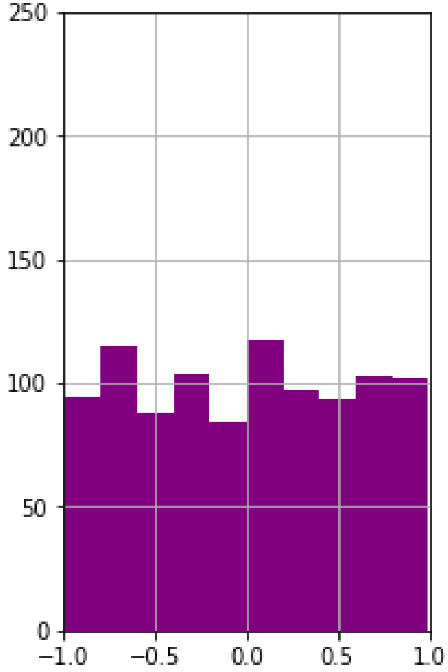
Most of activation values are 0





Activation values distributes from -1 to 1 although most are 0



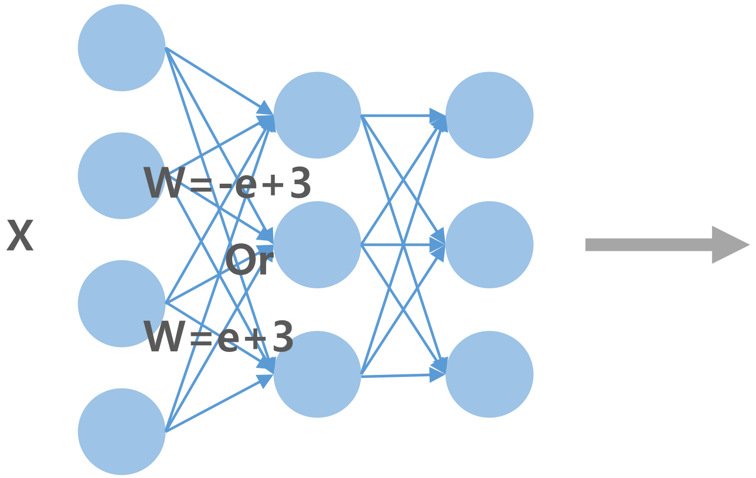


Weight Initialization

Poor Initialization Problem

Distribution of activation values of each node affects speed of convergence

Weight initialized with too Small or Big Number



Output value = WX^T = too small or big

Some calculated values are too small and vice versa.

Extremely high or low values drive activation value towards -1 or 1

Weight Initialization

How can we initialize weight values well?

1. Initialize with 0
2. Initialize with Uniform Distribution
3. Initialize with Standard Gaussian Distribution

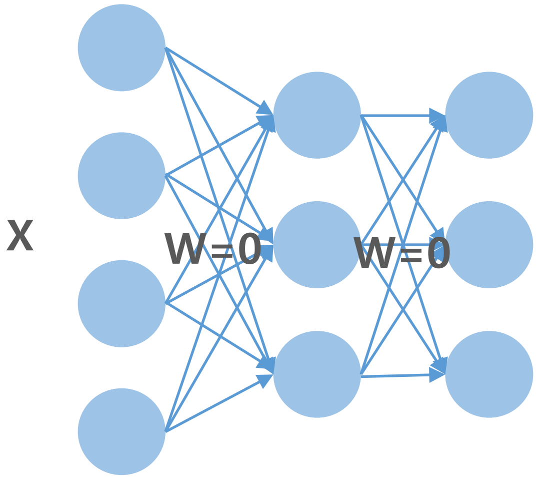
http://www.deeplearning.ai/ai-notes/initialization/#1

Weight Initialization

Initialize with Zero

NN's task is optimizing **weight** values. Then, how can we initialize the weight value?

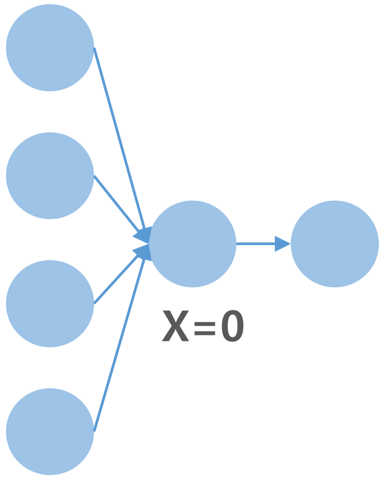
All weight initialized with 0



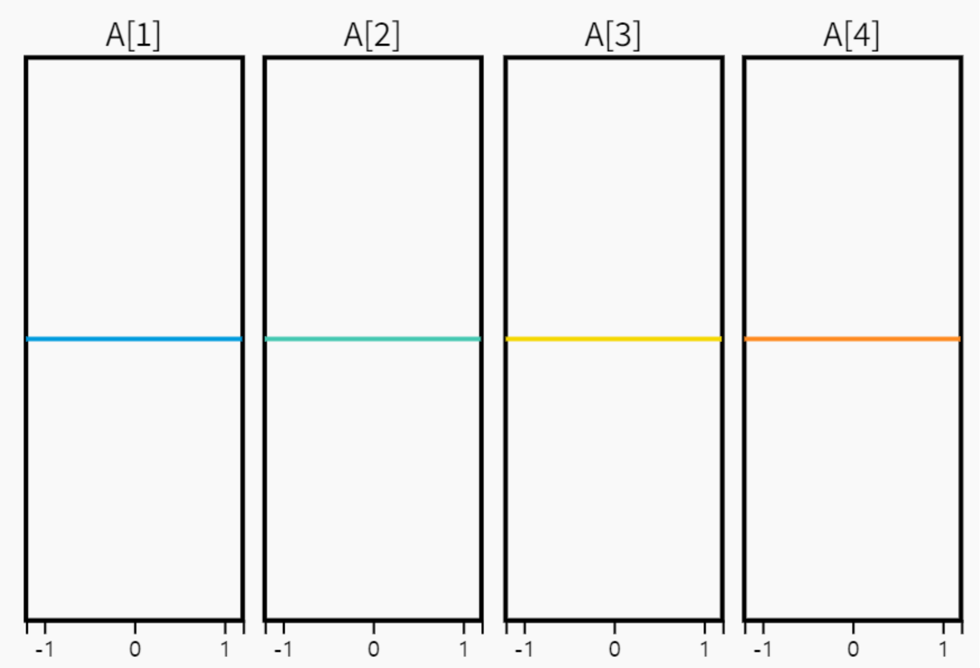
Output value = WX^T=0

All outputs and gradients are same as 0

So **entire neurons do the same task**

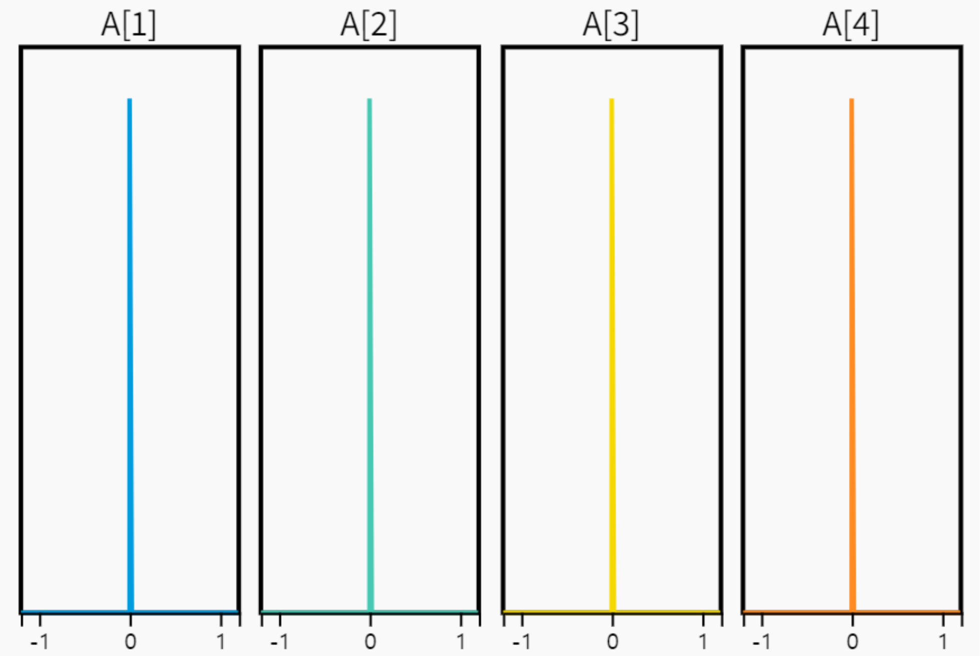


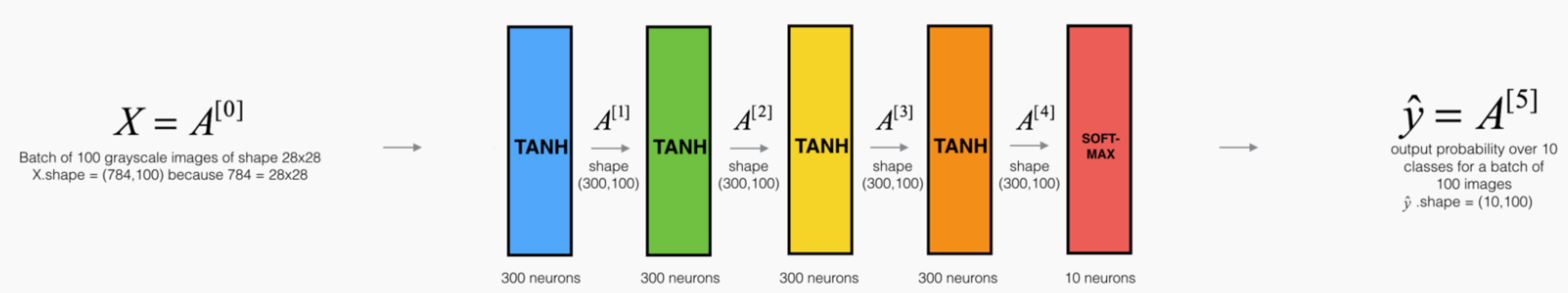
Initialize with Zero



Activation Values
Just after
1 batch
(1 iteration)

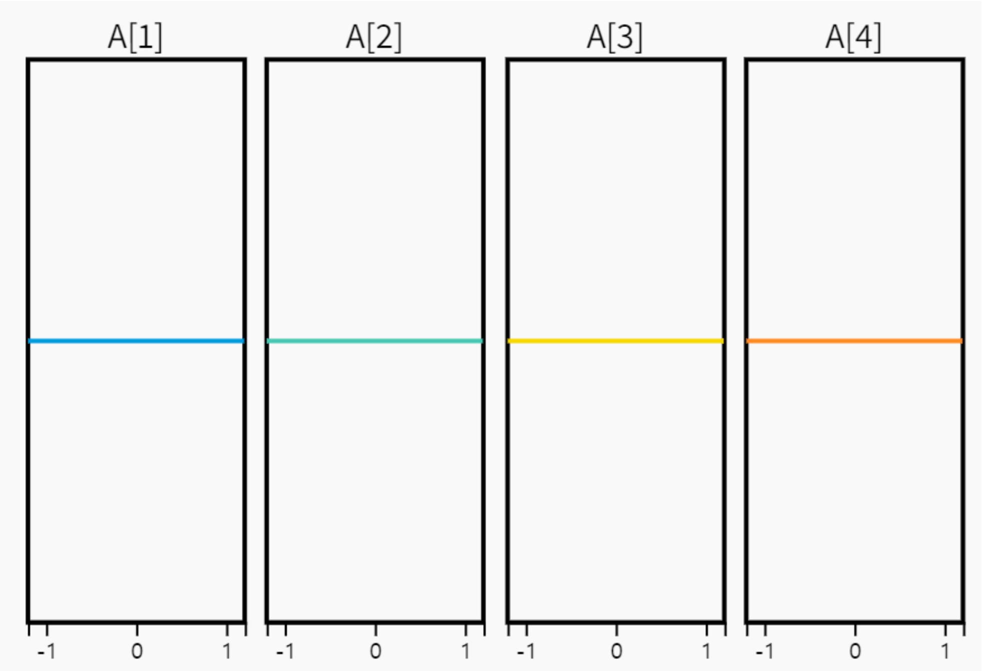
Using Tanh
Activation Func.





Weight Initialization

Initialize with Uniform Distribution



Activation Values

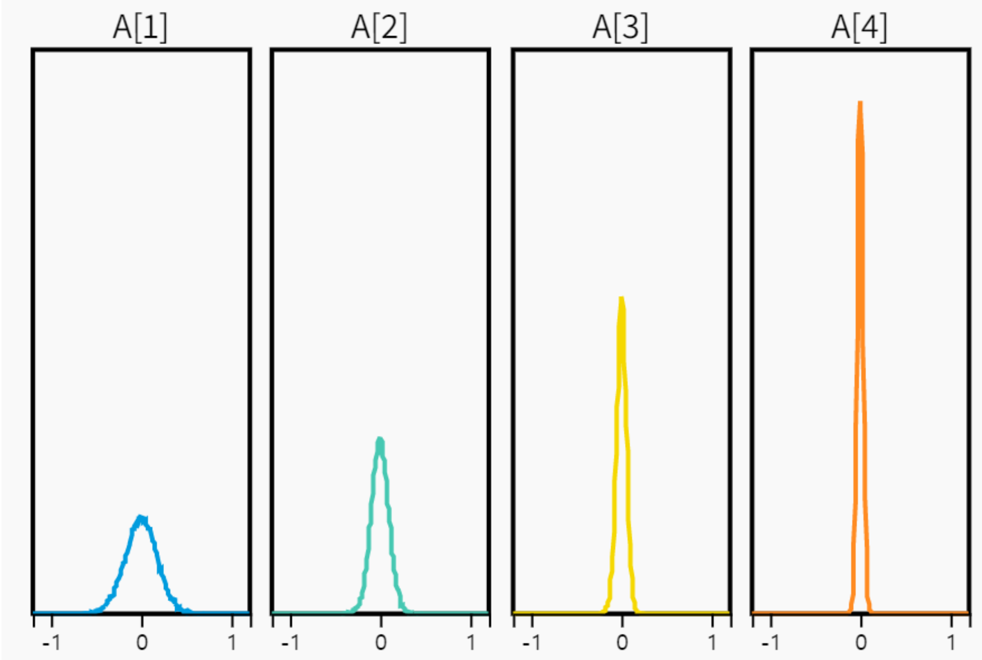
Just after

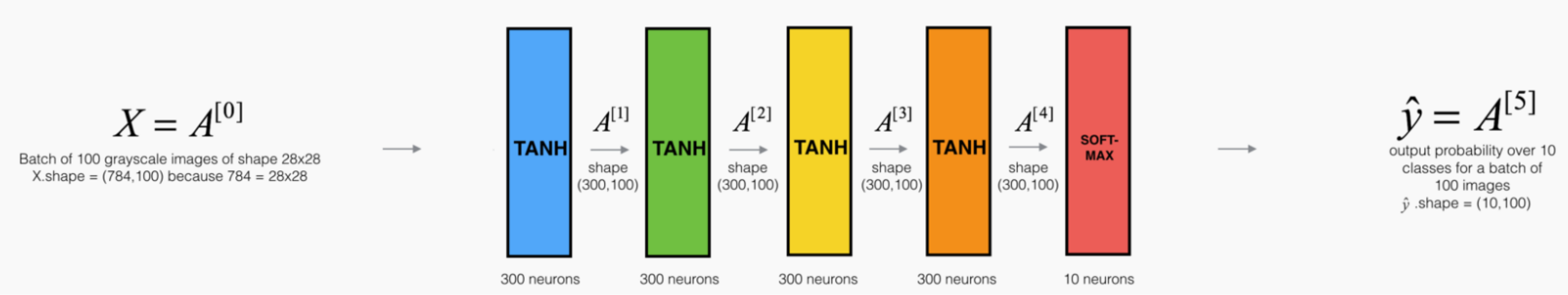
1 batch

(1 iteration)

Using Tanh

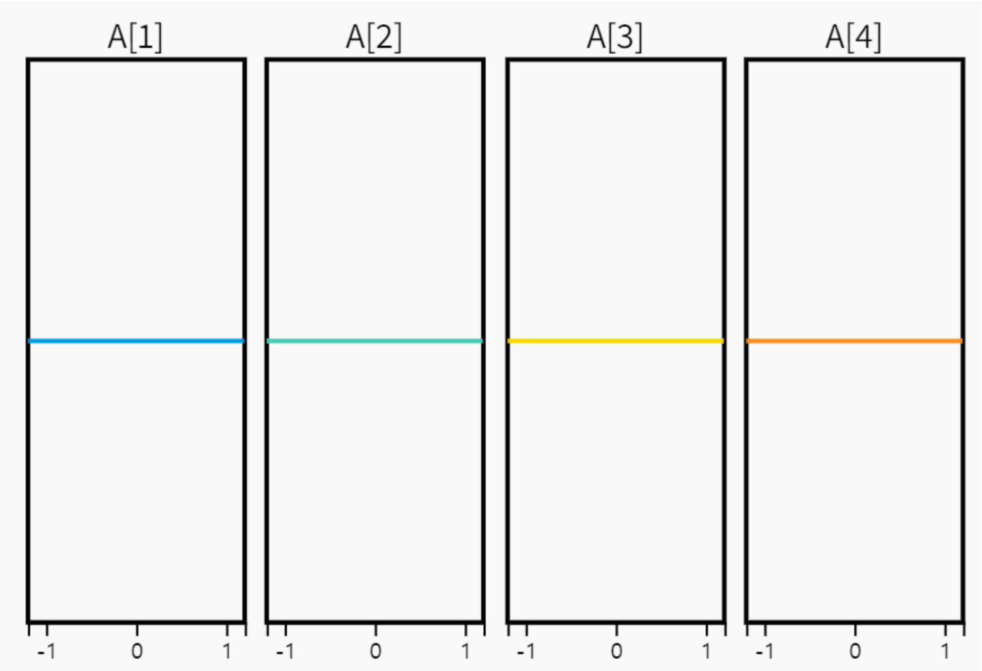
Activation Func.





Weight Initialization

Initialize with Standard Gaussian Distribution



Activation Values

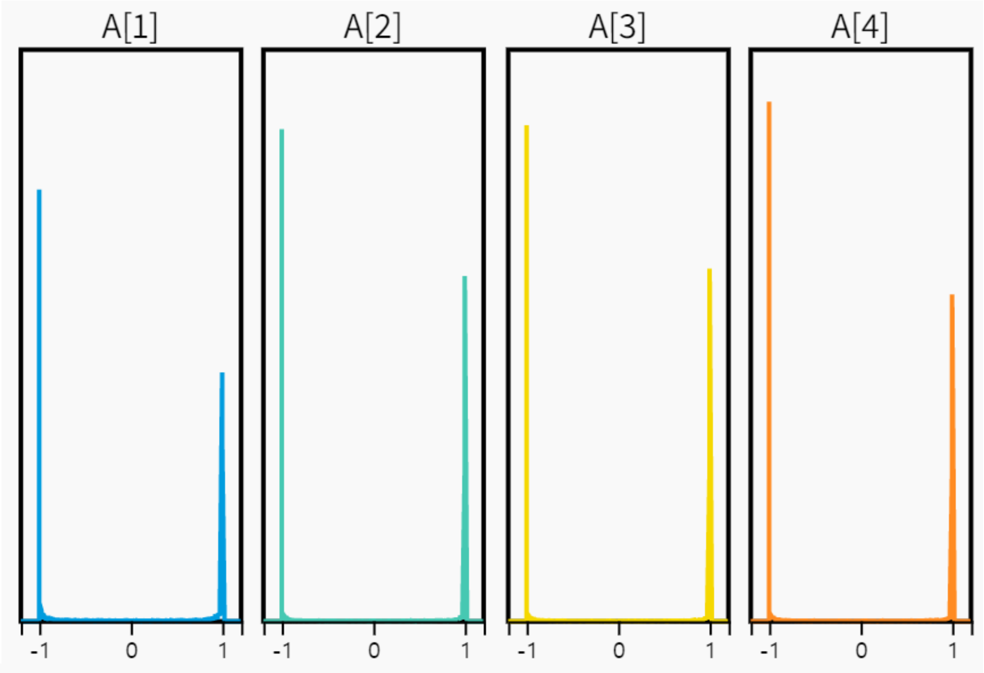
Just after

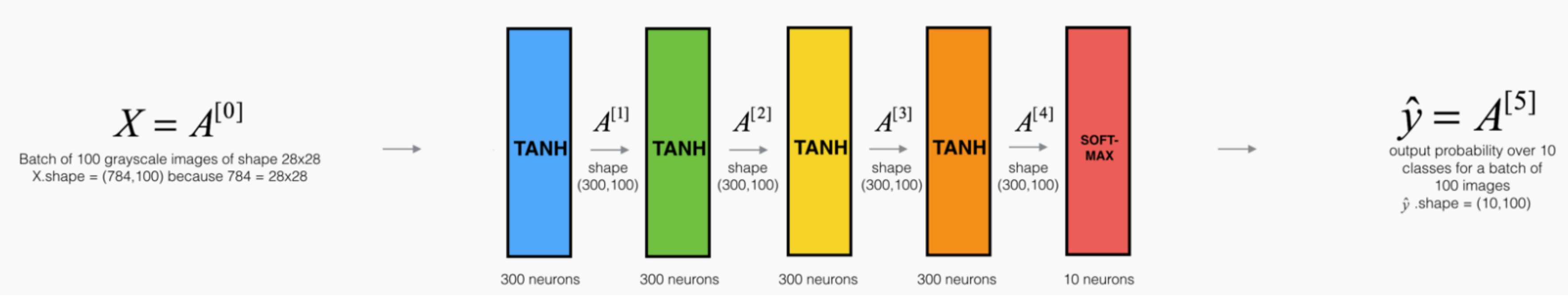
1 batch

(1 iteration)

Using Tanh

Activation Func.





Weight Initialization

Solution for Poor Initialization Problem

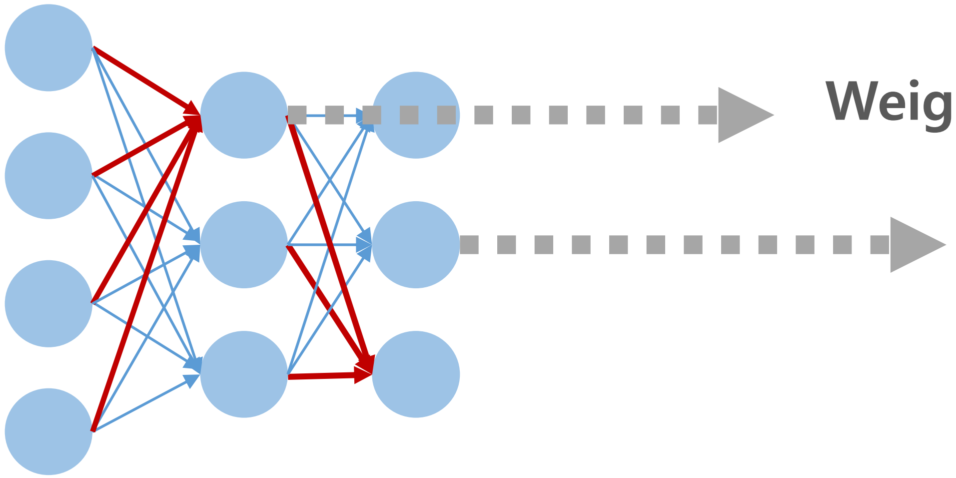
1. Xavier Initialization

2. Kaiming He Initialization

Weight Initialization

Xavier Initialization

Xavier weight $w_i \sim G(0, \frac{1}{N_{in}})$



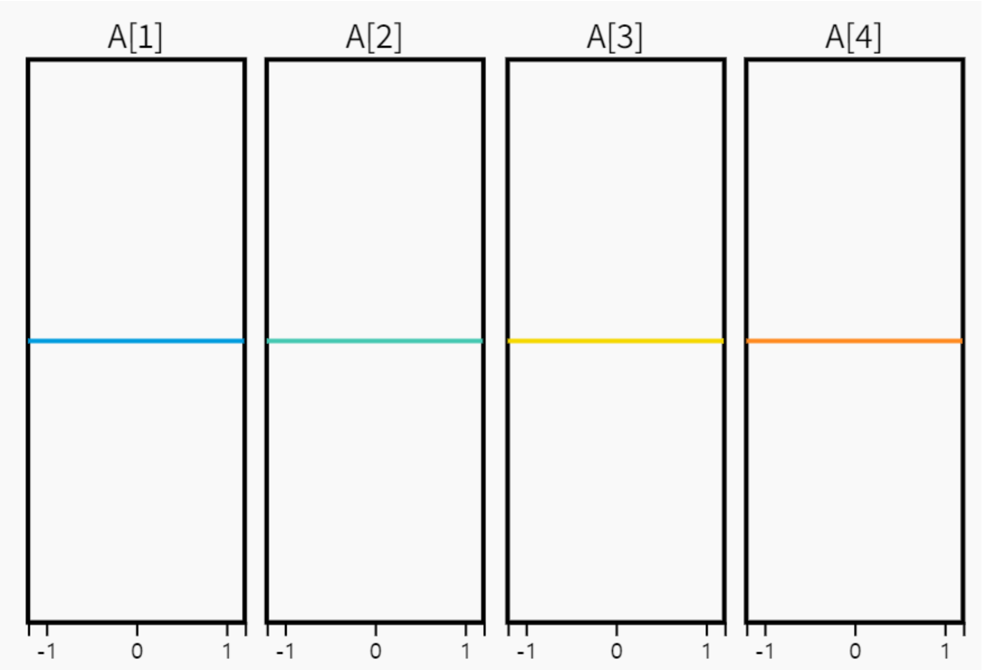
Weights connected with this node, $w_i \sim G(0, 1/4)$

Weights connected with this node, $w_i \sim G(0, 1/3)$

Ref : Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." Aistats. Vol. 9. 2010

Weight Initialization

Initialize with Xavier Method



Activation Values

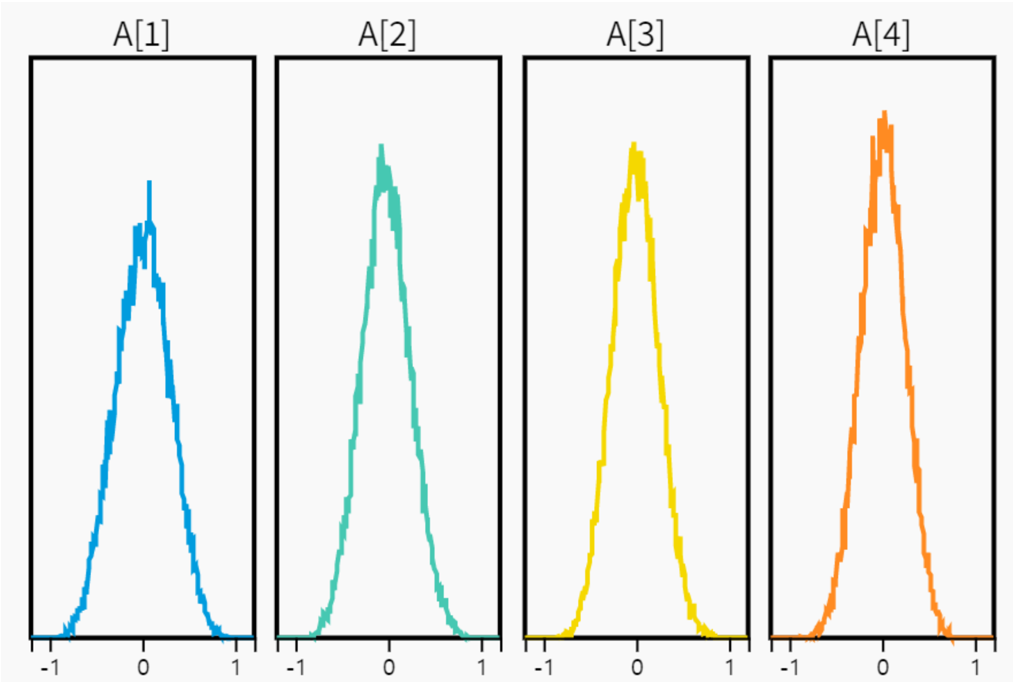
Just after

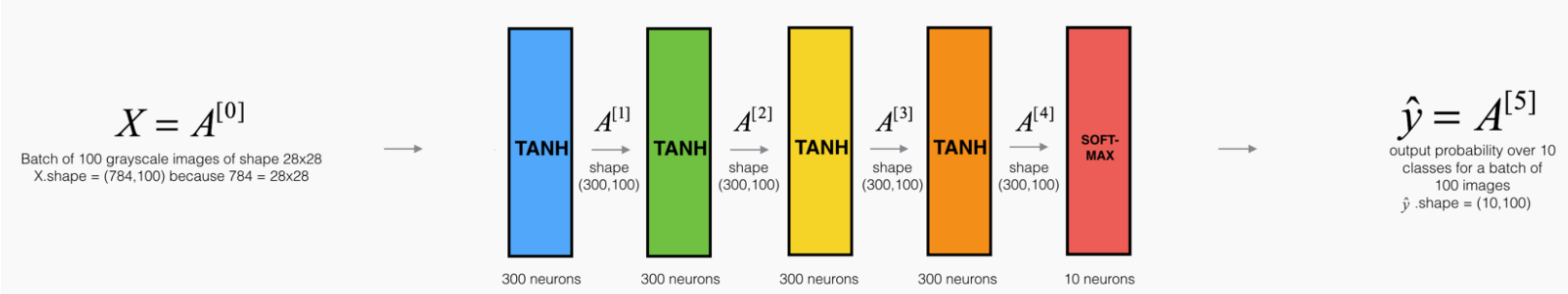
100 batch

(100 iteration)

Using Tanh

Activation Func.



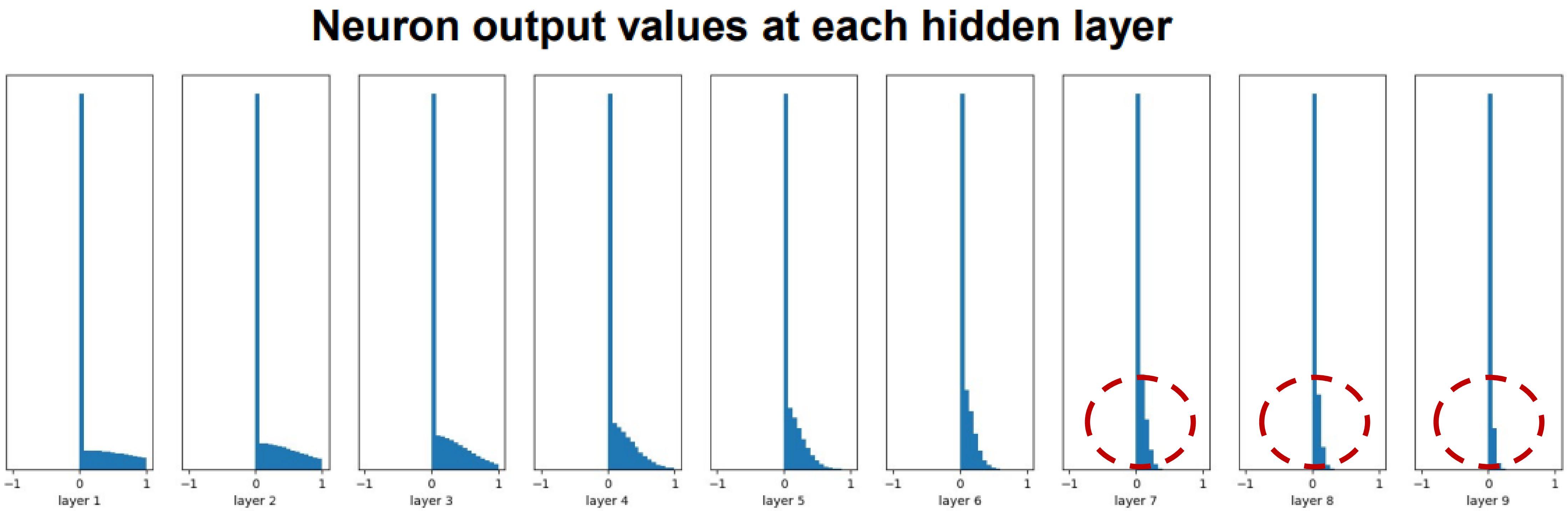


Weight Initialization

Xavier Initialization

But for ReLU Function, doesn't work well

It shows same problem (most of activation values converge to 0)

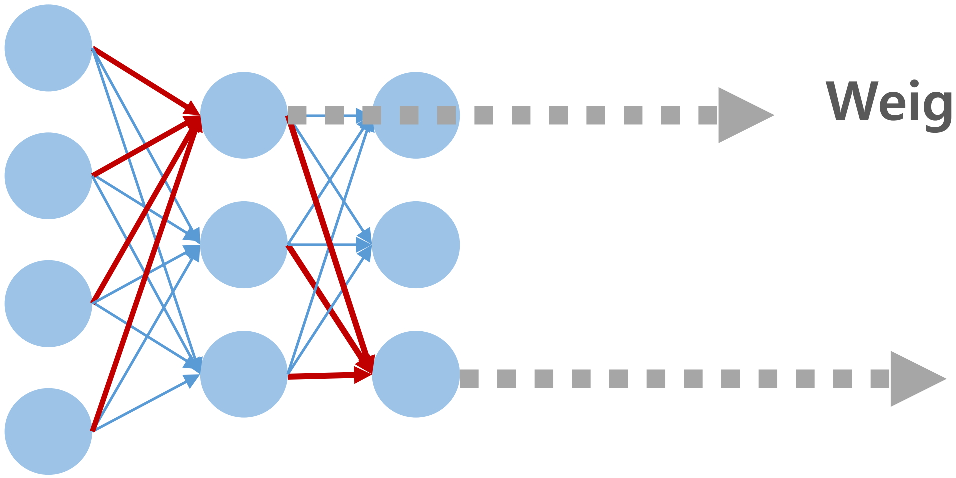


Ref : Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." Aistats. Vol. 9. 2010

Weight Initialization

Kaiming He Initialization

Xavier weight $w_i \sim G(0, \frac{2}{N_{in}})$



Weights connected with this node, $w_i \sim G(0, 2/4)$

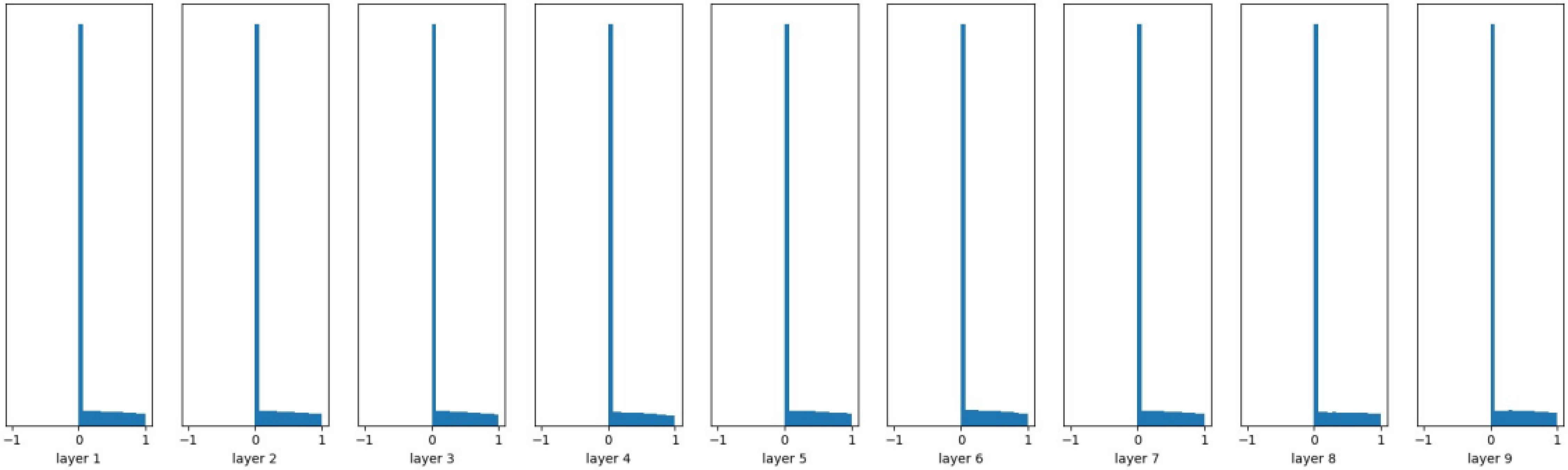
Weights connected with this node, $w_i \sim G(0, 2/3)$

Ref: Kaiming He et al., Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, 2015.d

Kaiming He Initialization

He initialization also work well with ReLU function

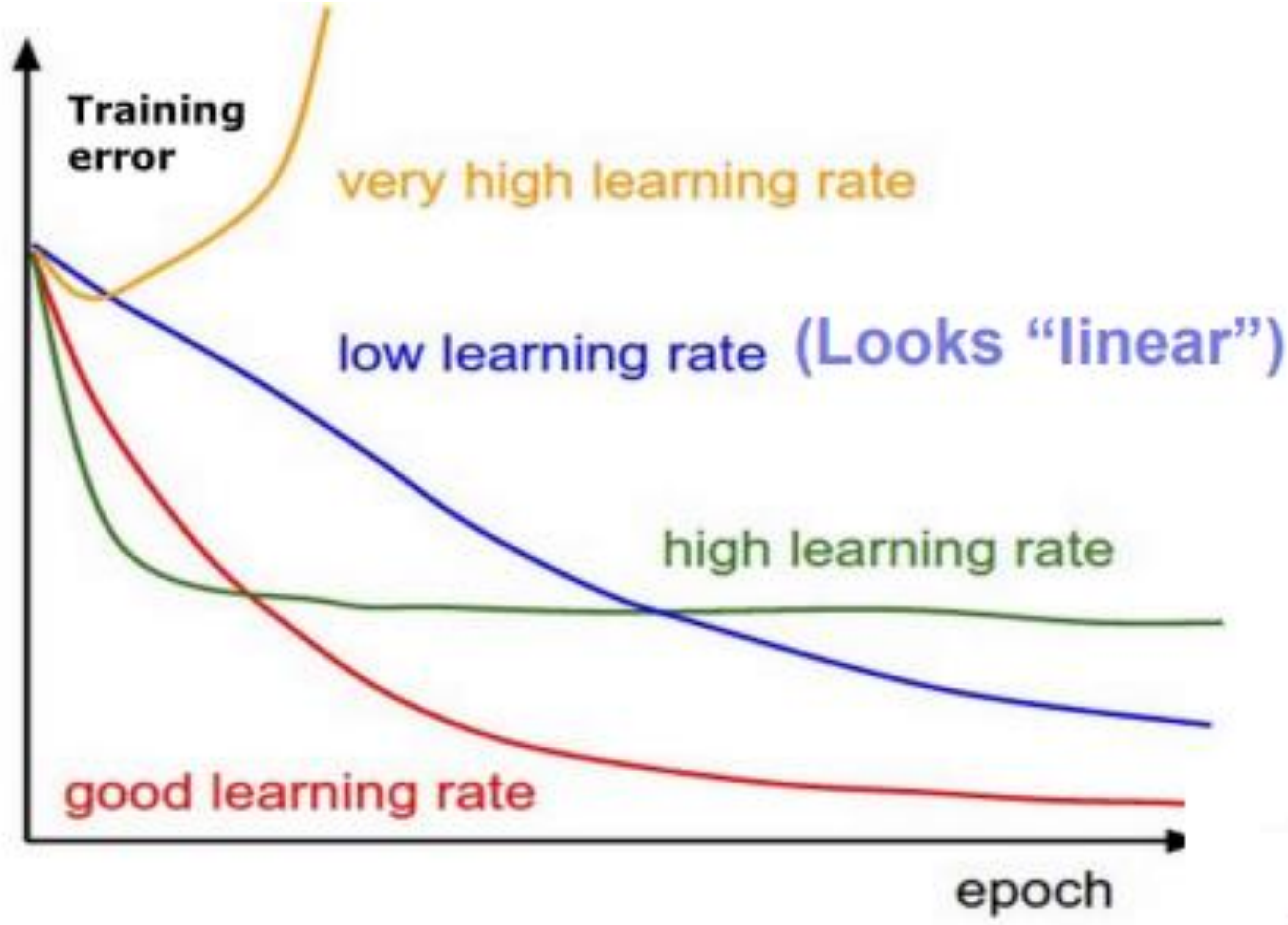
Neuron output values at each hidden layer

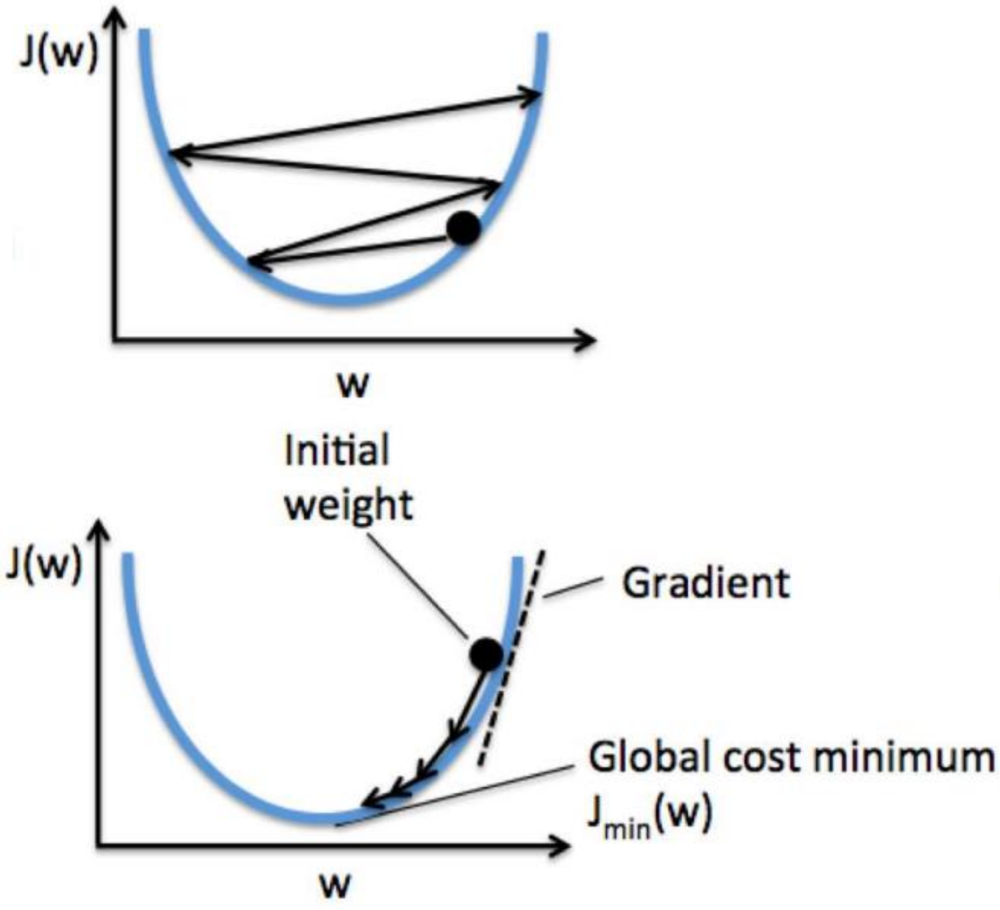


Ref: Kaiming He et al., Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, 2015.d

Learning Rate

What is the Best Learning Rate?





Learning Rate

Learning Rate Scheduling

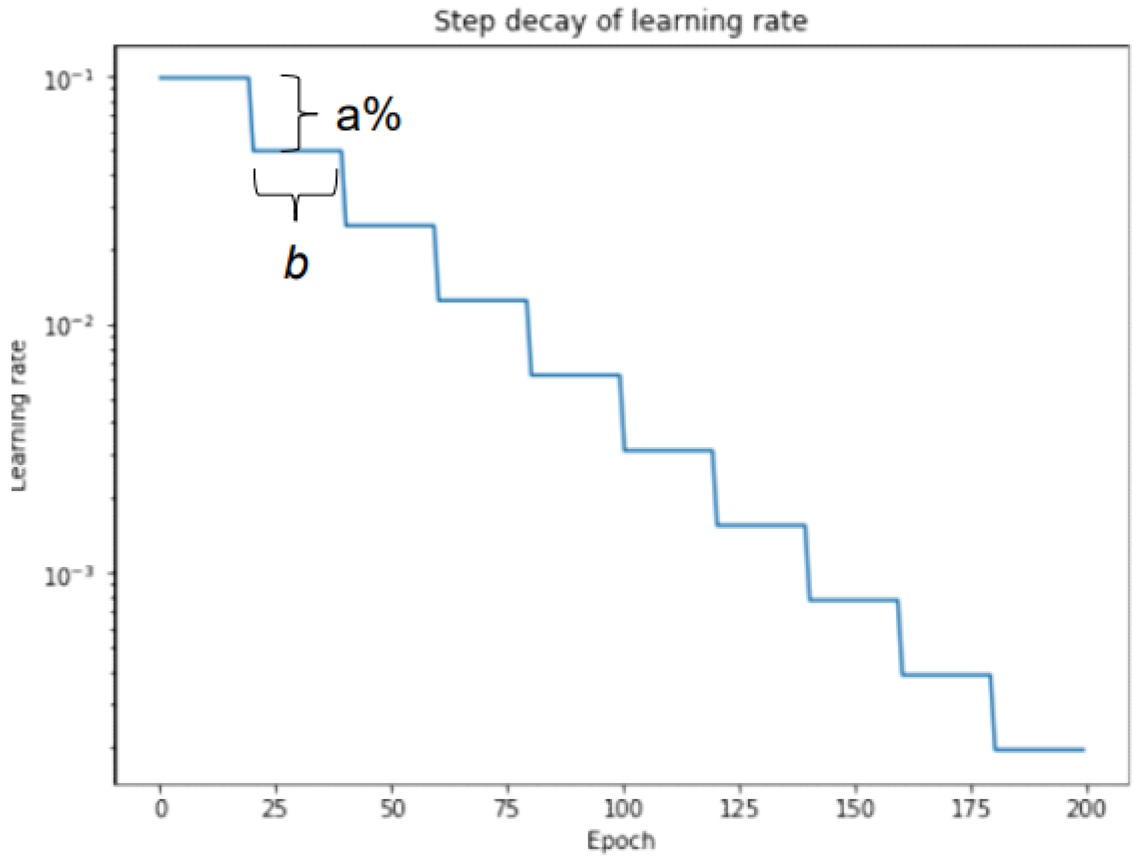
Seek to adjust the learning rate during training by reducing the learning rate according to a pre-defined schedule

- **Time-based decay:** $lr = \frac{lr_0}{1+kt}$
- **SquareRoot decay:** $lr = lr_0/\sqrt{t+1}$
- **Exponential decay:** $lr = lr_0 \exp(-kt)$

lr : Learning Rate, lr_0 : Initial Learning Rate, k : hyperparameter, t : iteration number

Learning Rate

Learning Rate Scheduling

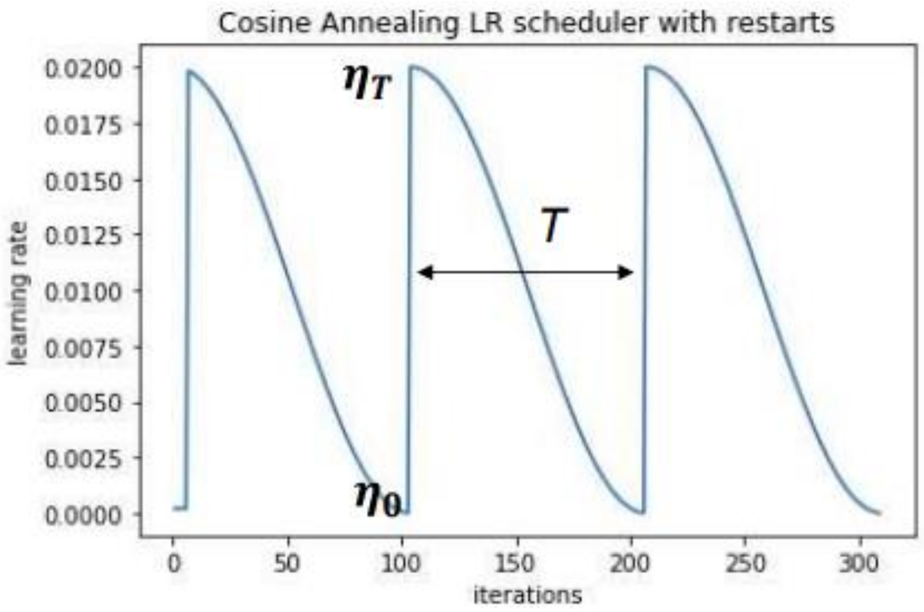


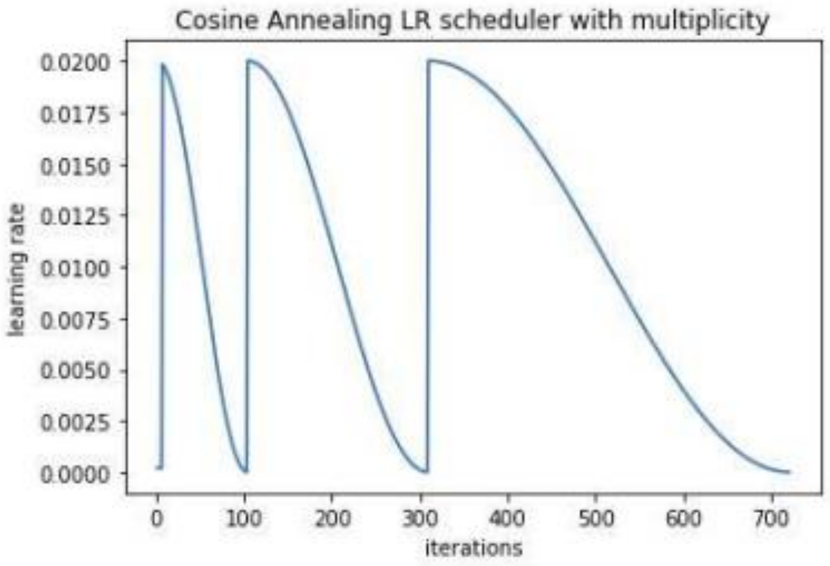
Step-based decay (lr annealing)

Reduce a% of learning rate after each or every b epoch

Learning Rate

Learning Rate Scheduling



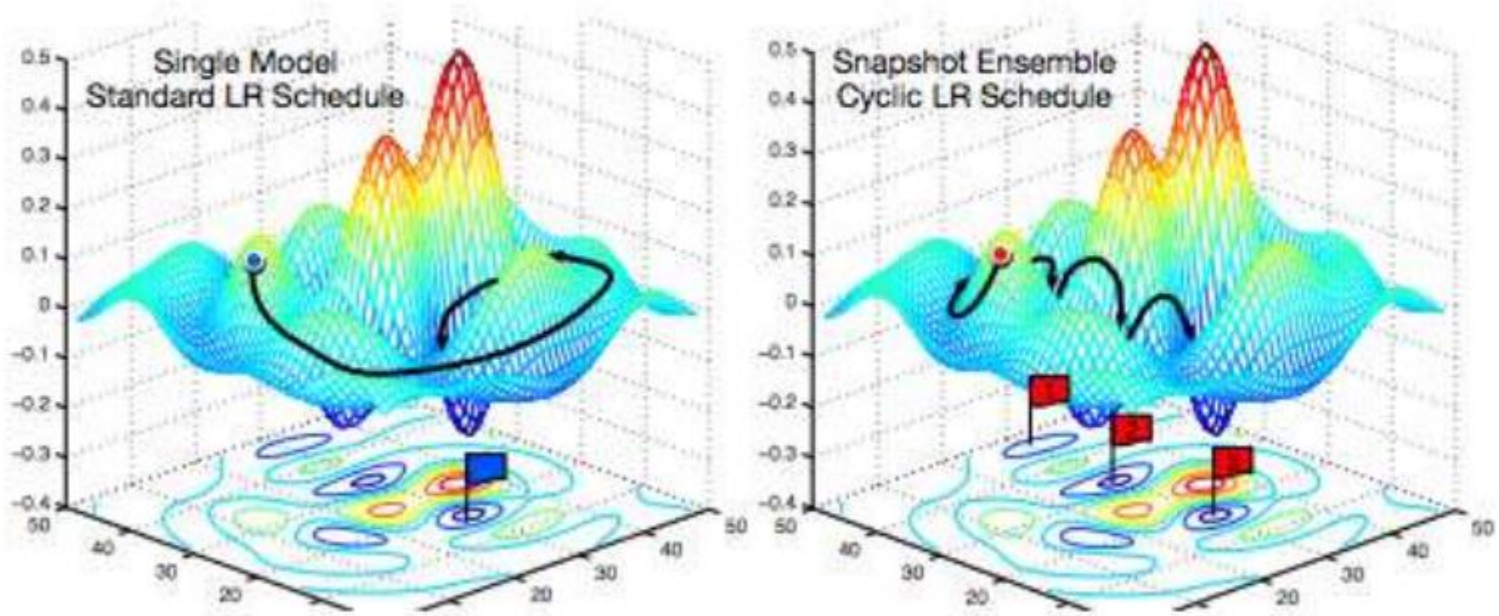


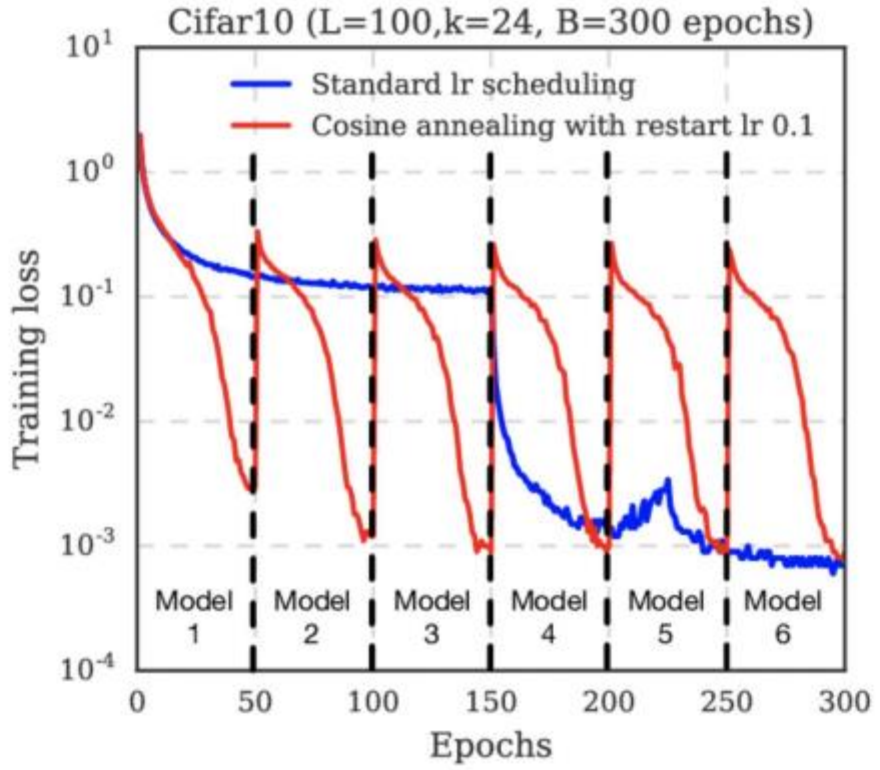
Cosine Scheduler

$$\eta_t = \eta_T + \frac{\eta_0 - \eta_T}{2}(1 + \cos(\pi t/T))$$

Epoch t<T, η_0 is the initial learning rate, η_T is the target rate at time. For t>T, we simply pin the value to η_T without increasing it again

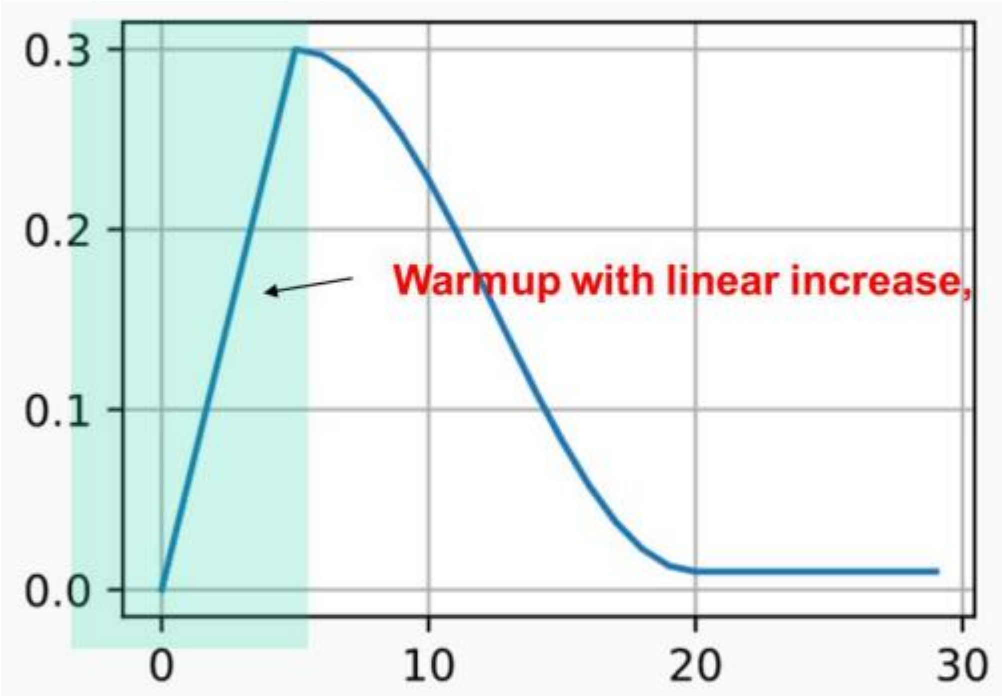
Learning Rate Scheduling





Learning Rate

Warmup



A warmup period before optimization can **prevent divergence**

Increase learning rate to its initial maximum and then cool down the rate until the end of the optimization process

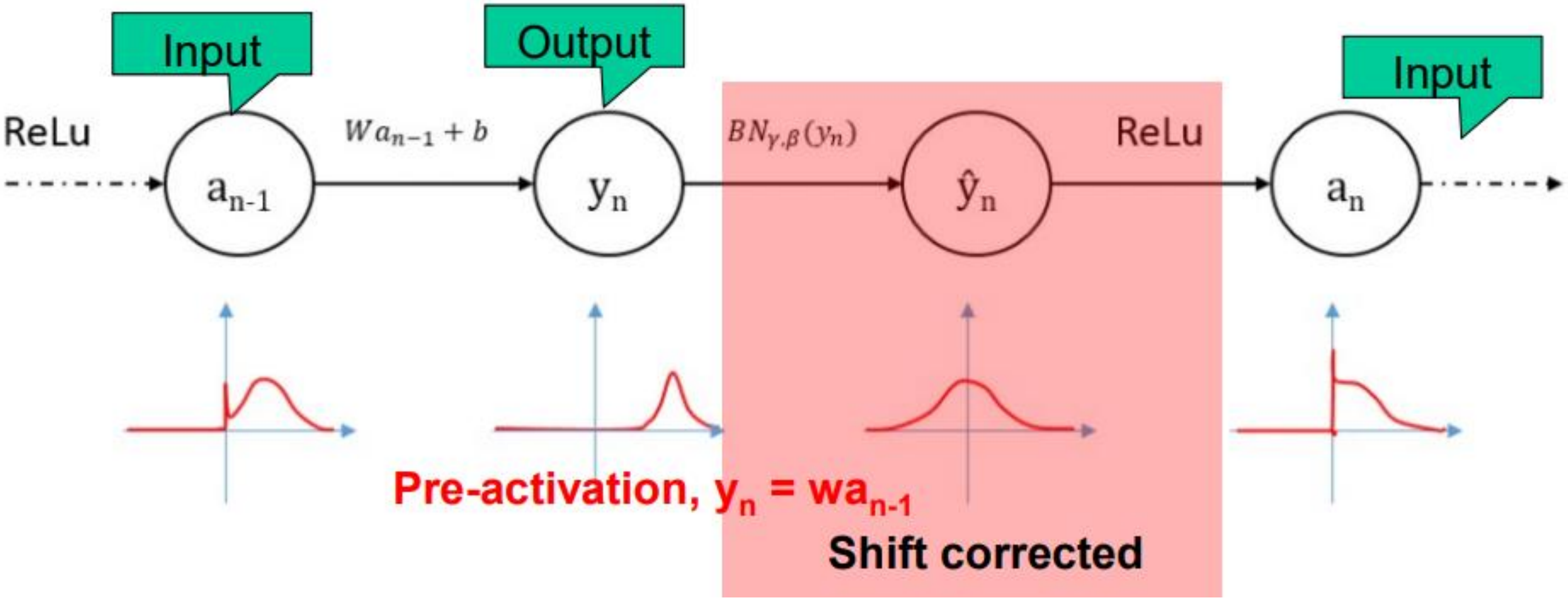
Warmup can be applied to any scheduler

Normalization

Normalizing the input value is crucial for gradient descent

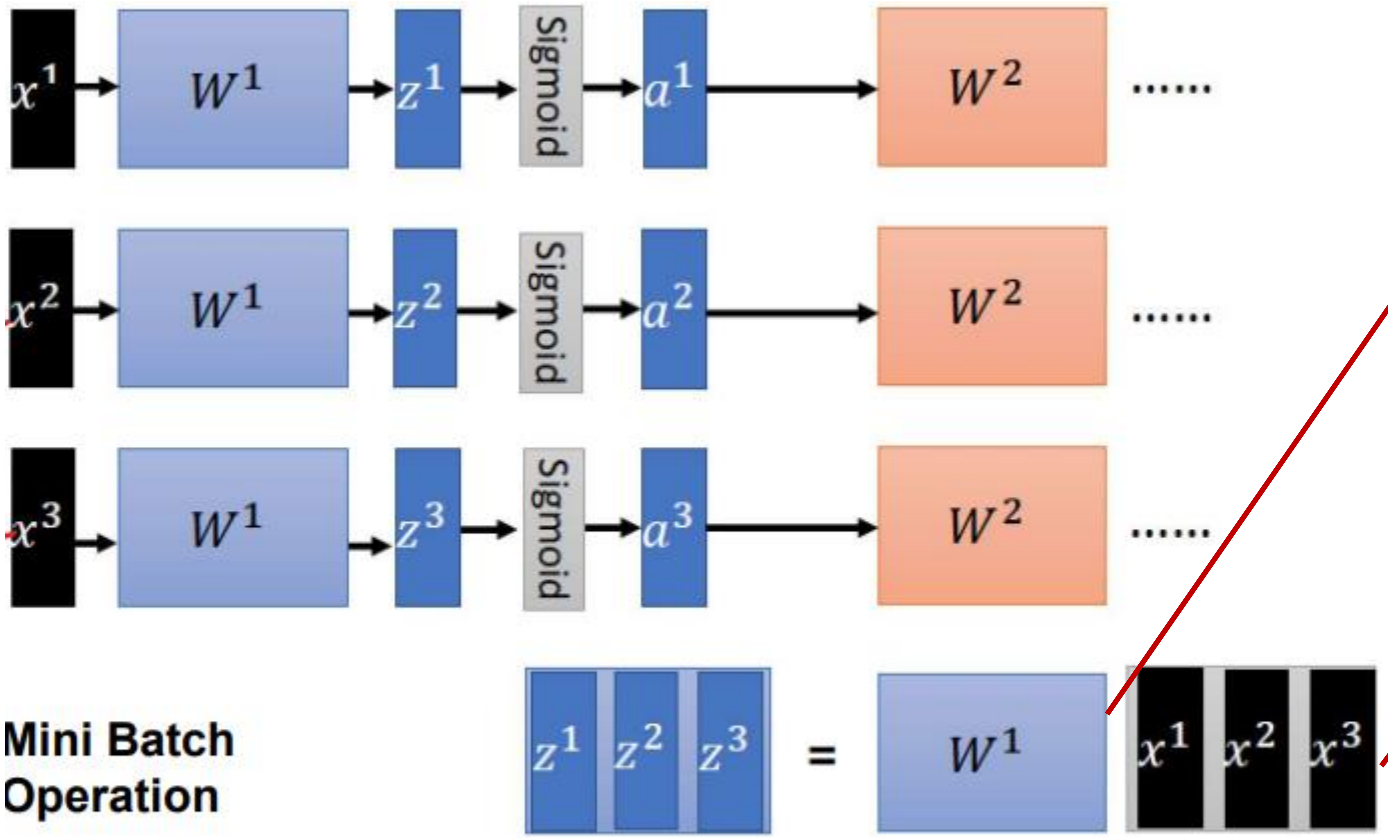
It also important for the **activation value between hidden layers**

Pre-activations or post-activations for each batch **(BN)** makes the learning much **faster** and **boost the accuracy**



Normalization

Remind the mini-batch operation, we divide the input into number of batches



Weight matrix is same for each batch (input)

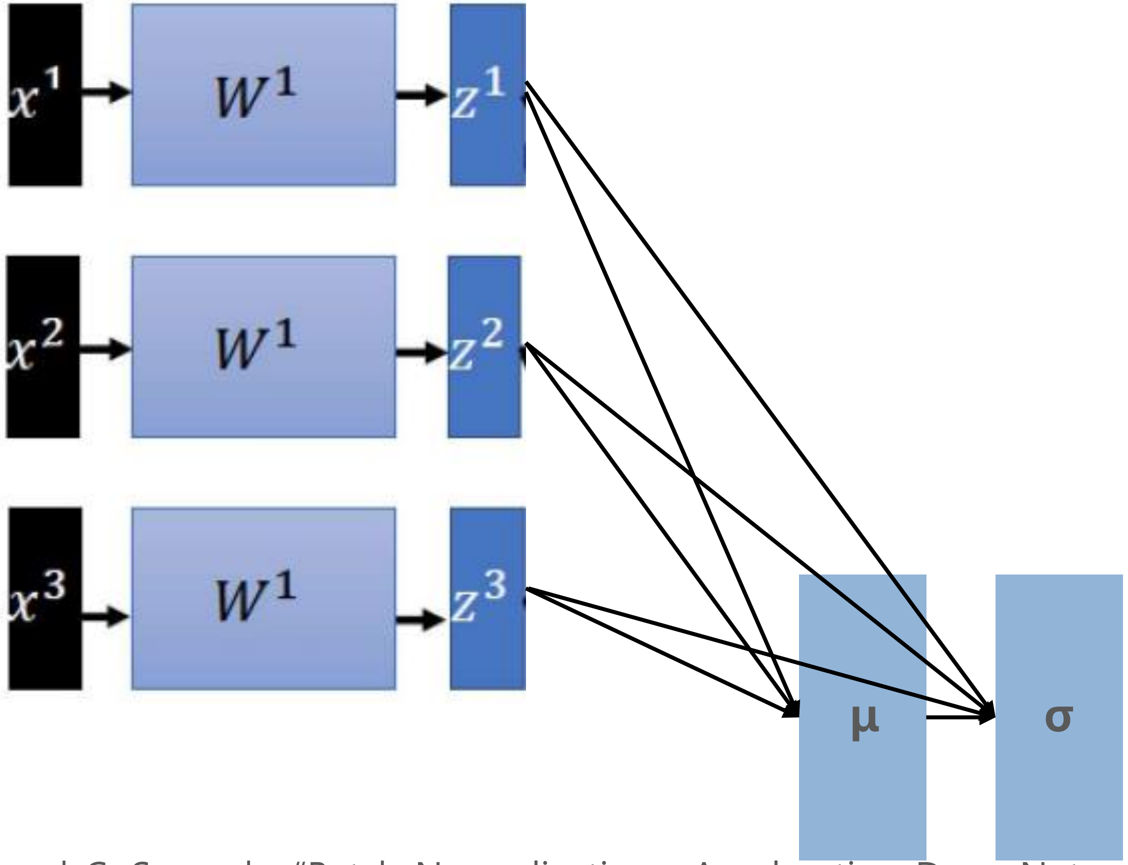
Input matrix is divided into 3 batches

Ref: S. Ioffe and C. Szegedy, "Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift," presented at ICML, 2015, pp. 448-456

Normalization

Ref: S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” presented at ICML, 2015, pp. 448–456.

Basic operation of BN is calculating the **mean** and **std** of result of input and weight calculation.



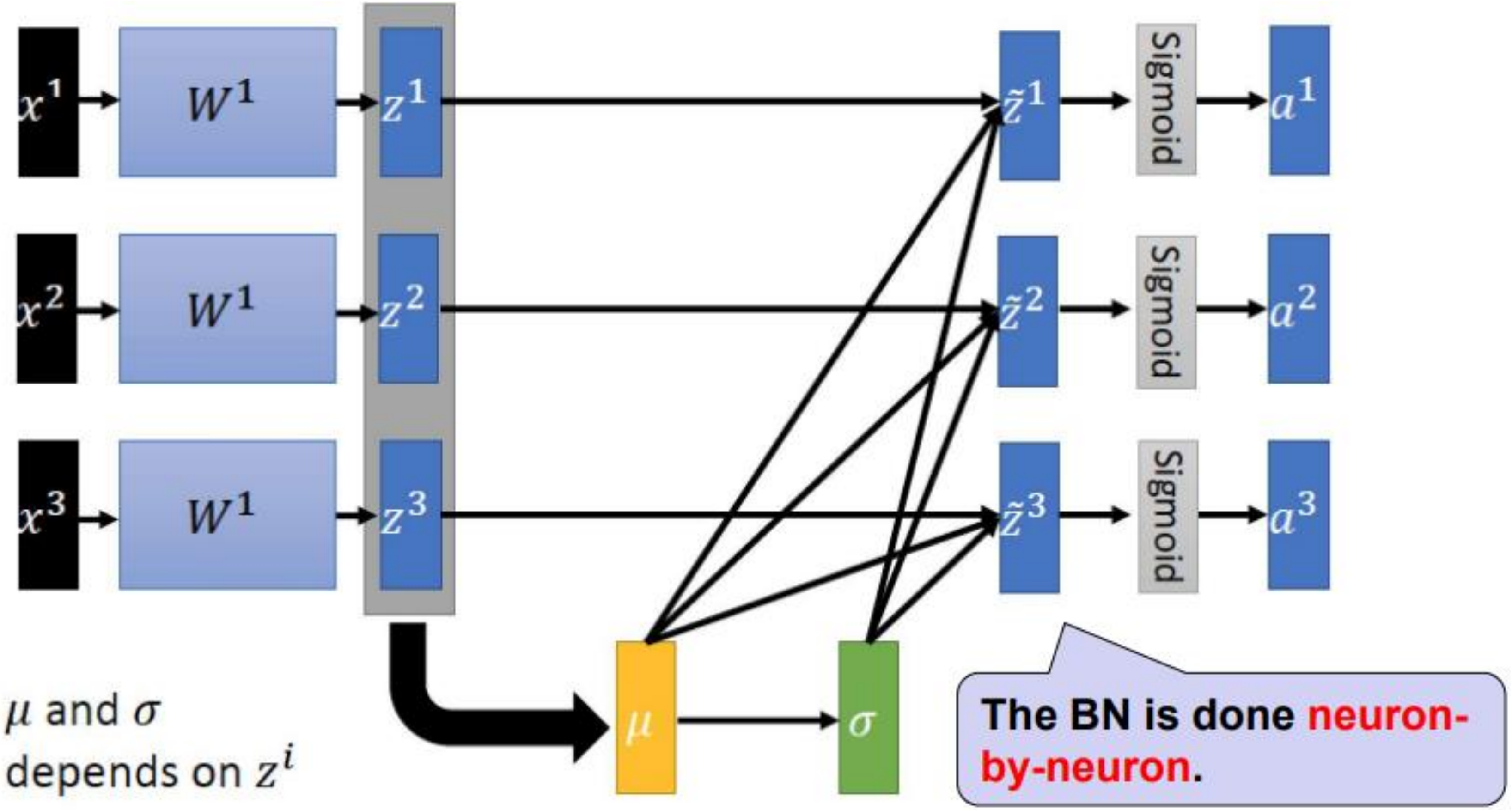
$$\mu = \frac{1}{3}\sum_{i}^{3} z^i$$

$$\sigma = \sqrt{\frac{1}{3}\sum_{i}^{3}(z^i - \mu)^2}$$

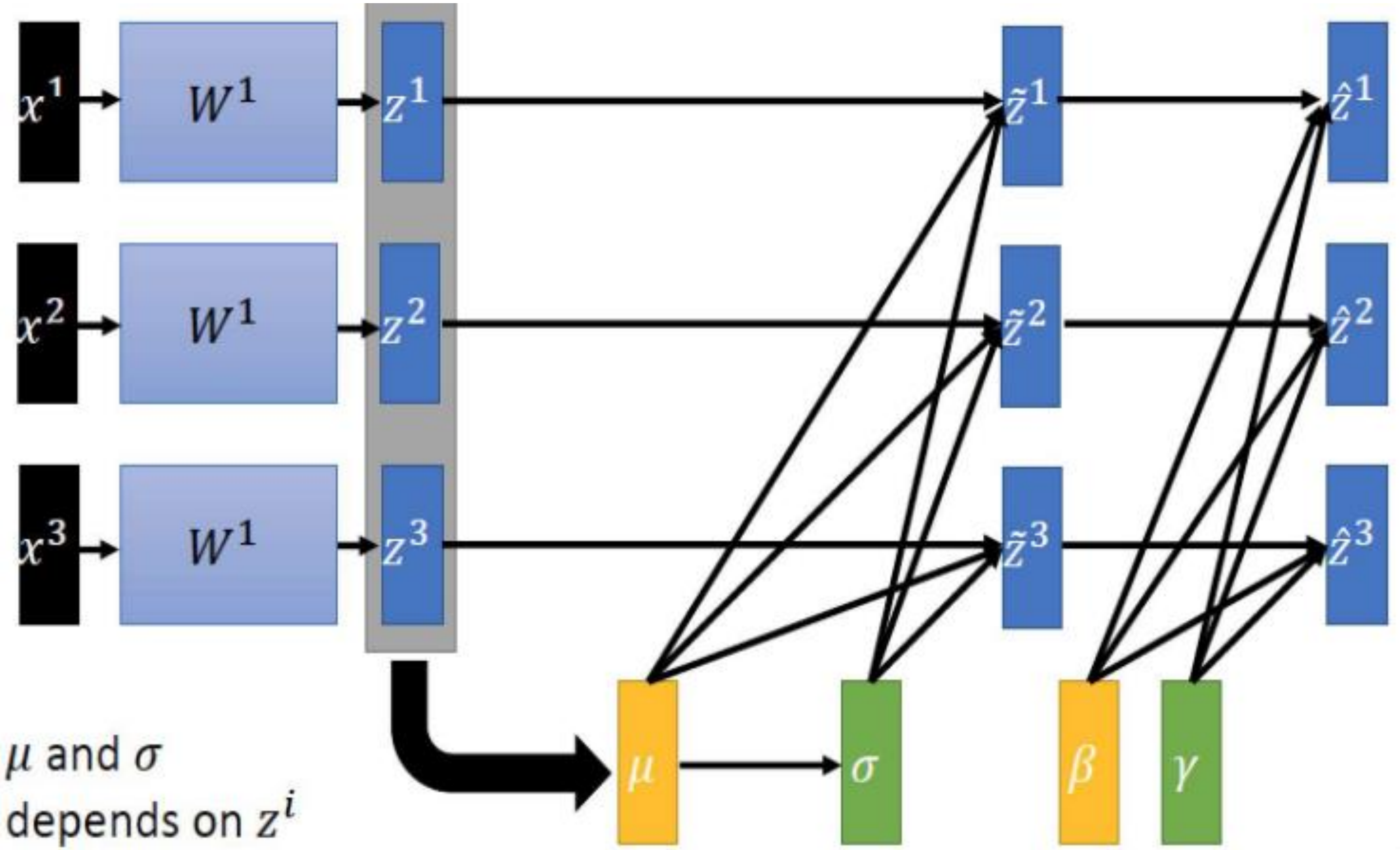
Ref: S. Ioffe and C. Szegedy, “Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift,” presented at ICML, 2015, pp. 448-456

Normalization

With using the mean and std value, **normalize** the z value **before they pass the activation function**



Normalization



$$\tilde{z}^i = \frac{z^i - \mu}{\sigma}$$

$$\hat{z}^i = \gamma \odot \tilde{z}^i + \beta$$

β and γ are

Network Parameters

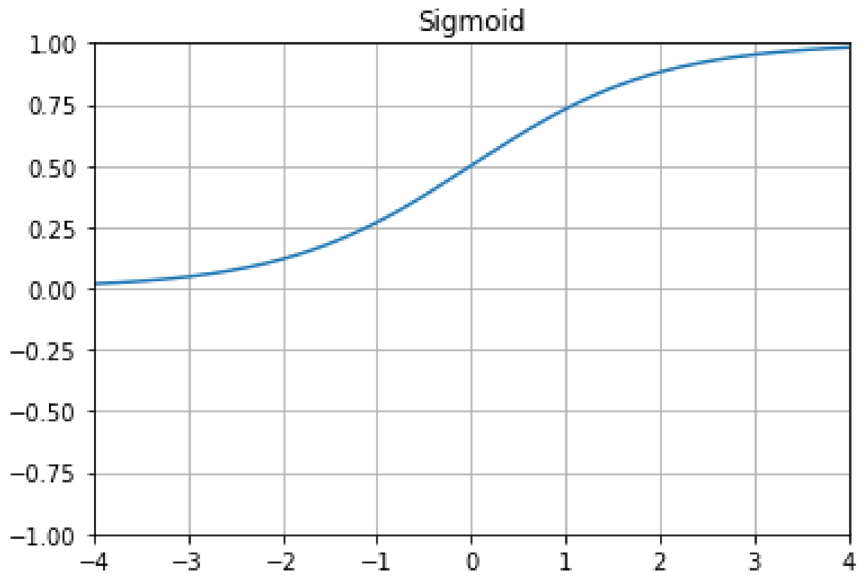
Ref: S. Ioffe and C. Szegedy, "Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift," presented at ICML, 2015, pp. 448-456

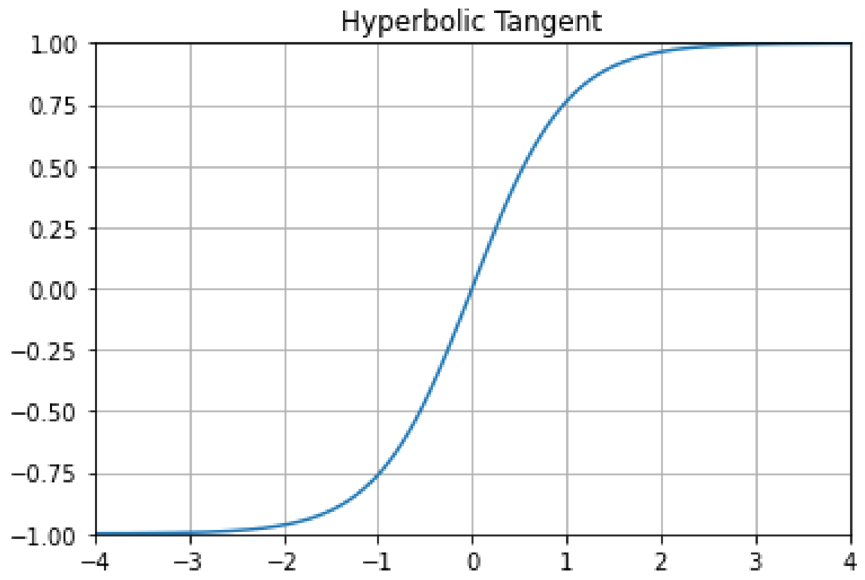
Why we need beta and gamma

After Normalization, 95% of activation values are in between -1.96 ~ 1.96

That range is linear-liked for these activation functions, so we should shift and rescale the activation values

Sigmoid & Tanh Function





Pros and Cons

Pros

• Previously we say high learning rate is a disaster, BN makes **high learning rate use is possible**

• Previously we say **sigmoid, tanh** functions don't work due to gradient vanishing, BN enables these functions **usable in the deep networks**

• BN makes weights **initialization not so critical**

• BN can **also increase the accuracy** of the model (make model generalize)

Normalization

Pros and Cons

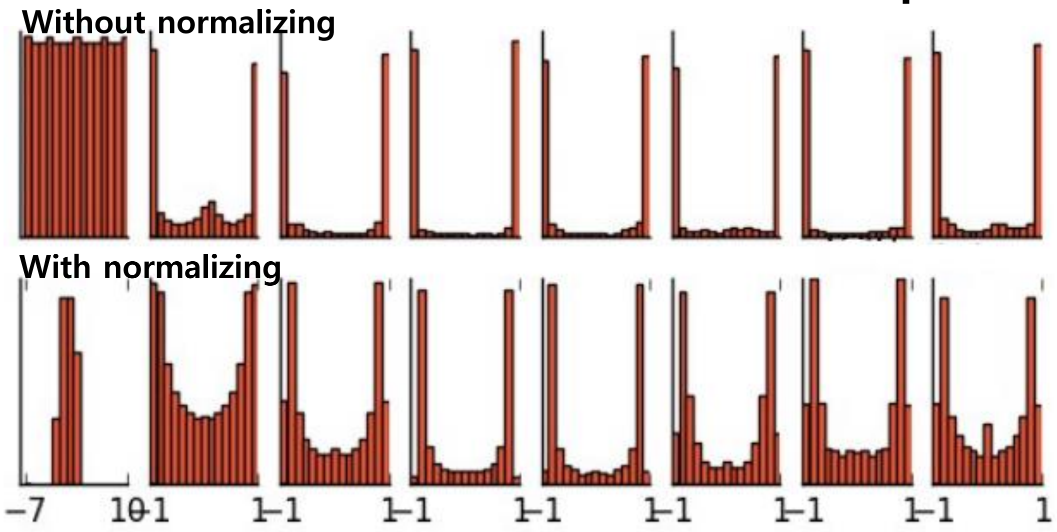
Cons

- Batch number (B) should be large to estimate reliable statistics
- Not suitable for dynamic network structure and recurrent network

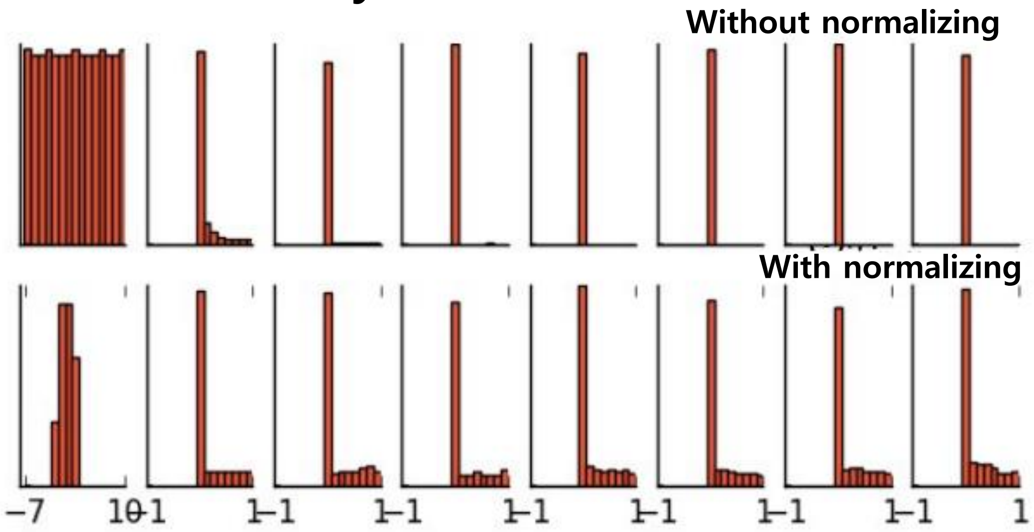
Normalization

Without using any initialization, BN makes very nice activation value distribution

Neuron output values at each hidden layer



Activation function : tanh



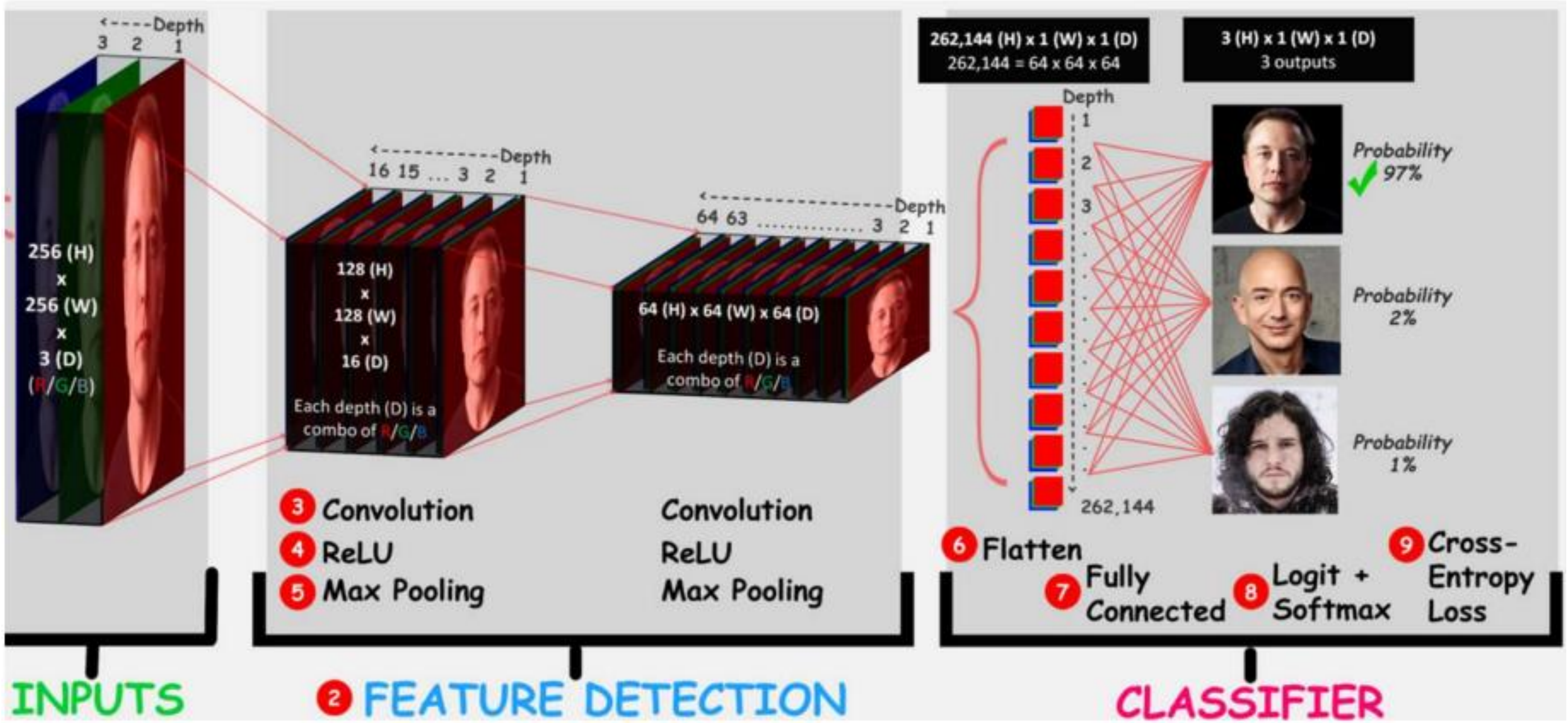
Activation function : ReLU

How CNN Works?

How CNN Works?

Why CNN works well for image data?

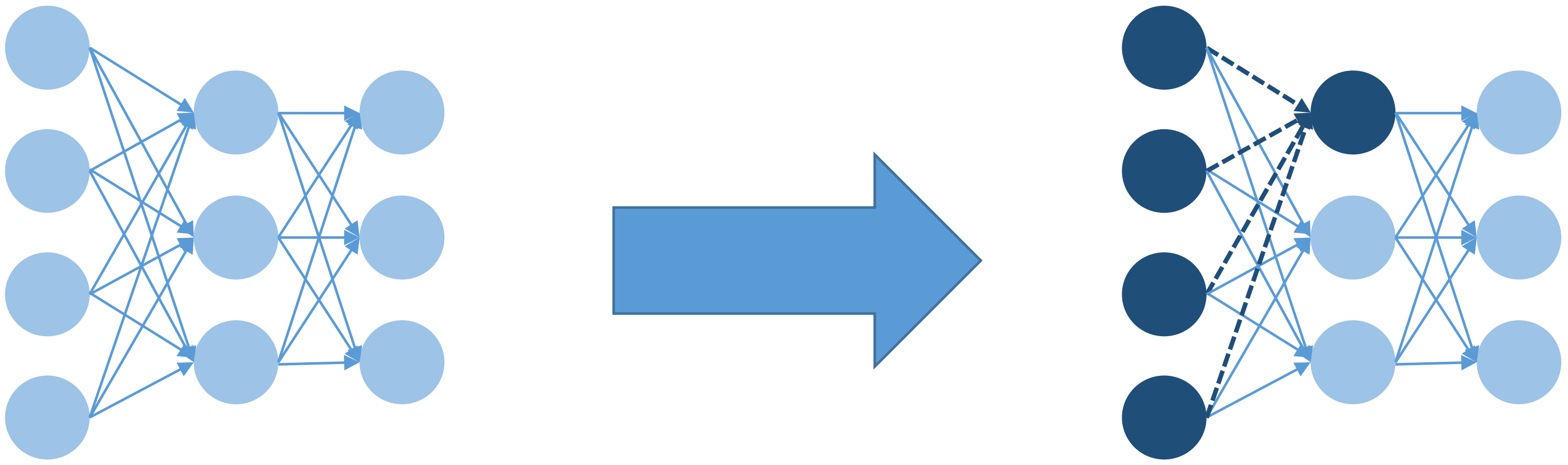
And why we use CNN for several task?



How CNN Works?

CNN is a special case of MLP

What is a difference of CNN and MLP?

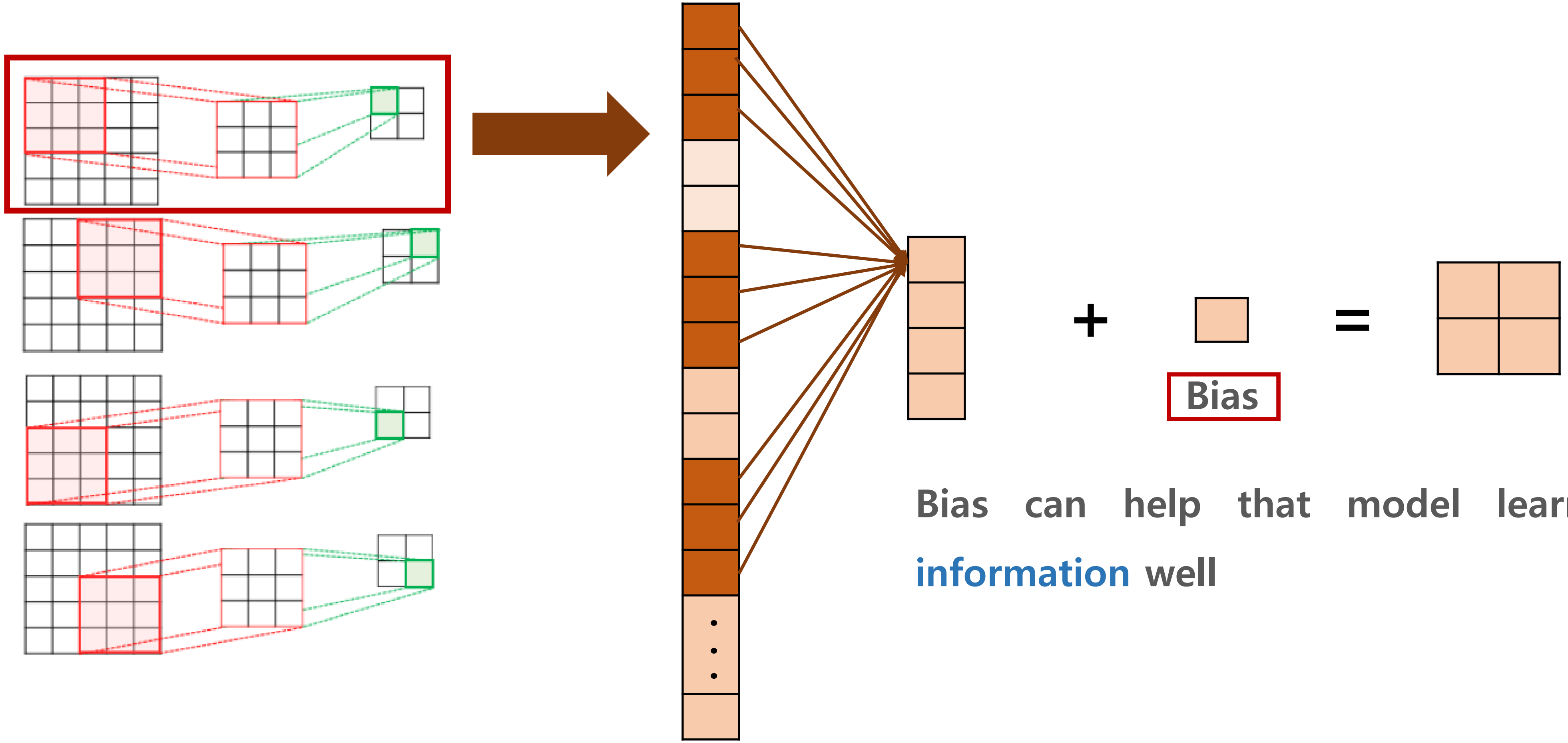


Original MLP (Fully connected layers) considers **all input feature**. They flatten the all-input features and feed them to next neuron. In fact, next neuron always **reflects entire features** from current input.

How CNN Works?

Two characters of CNN

1. Local Connectivity

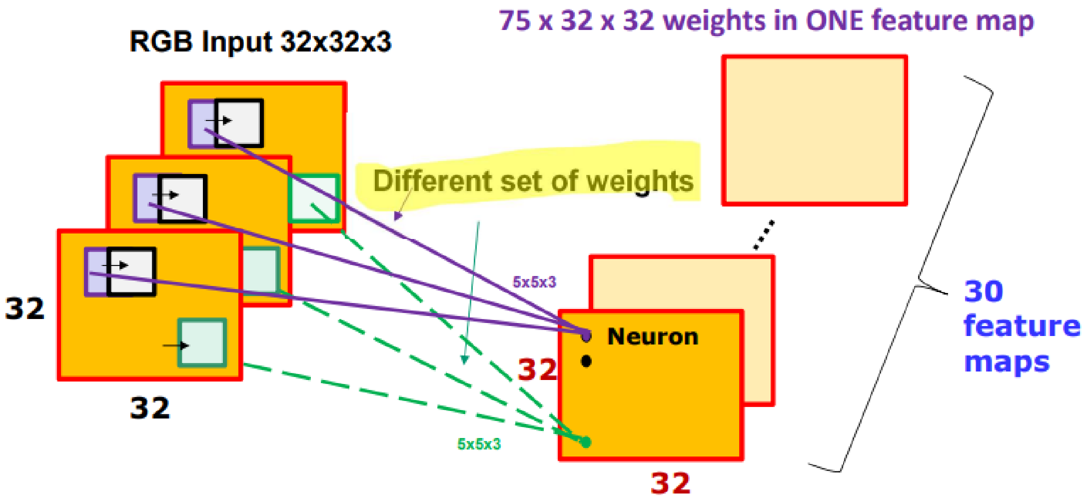


Bias can help that model learn the local information well

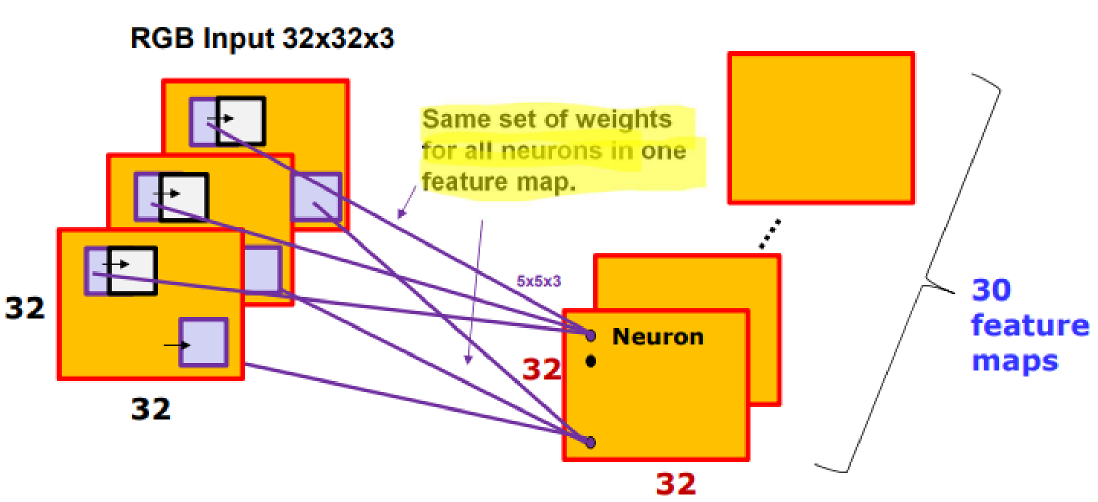
How CNN Works?

Two characters of CNN

2. Weight Sharing



With MLP



With CNN

Different weights for every single neuron

Total # weights = (5*5*3)*(32*32*30)

= **2.3 millions**

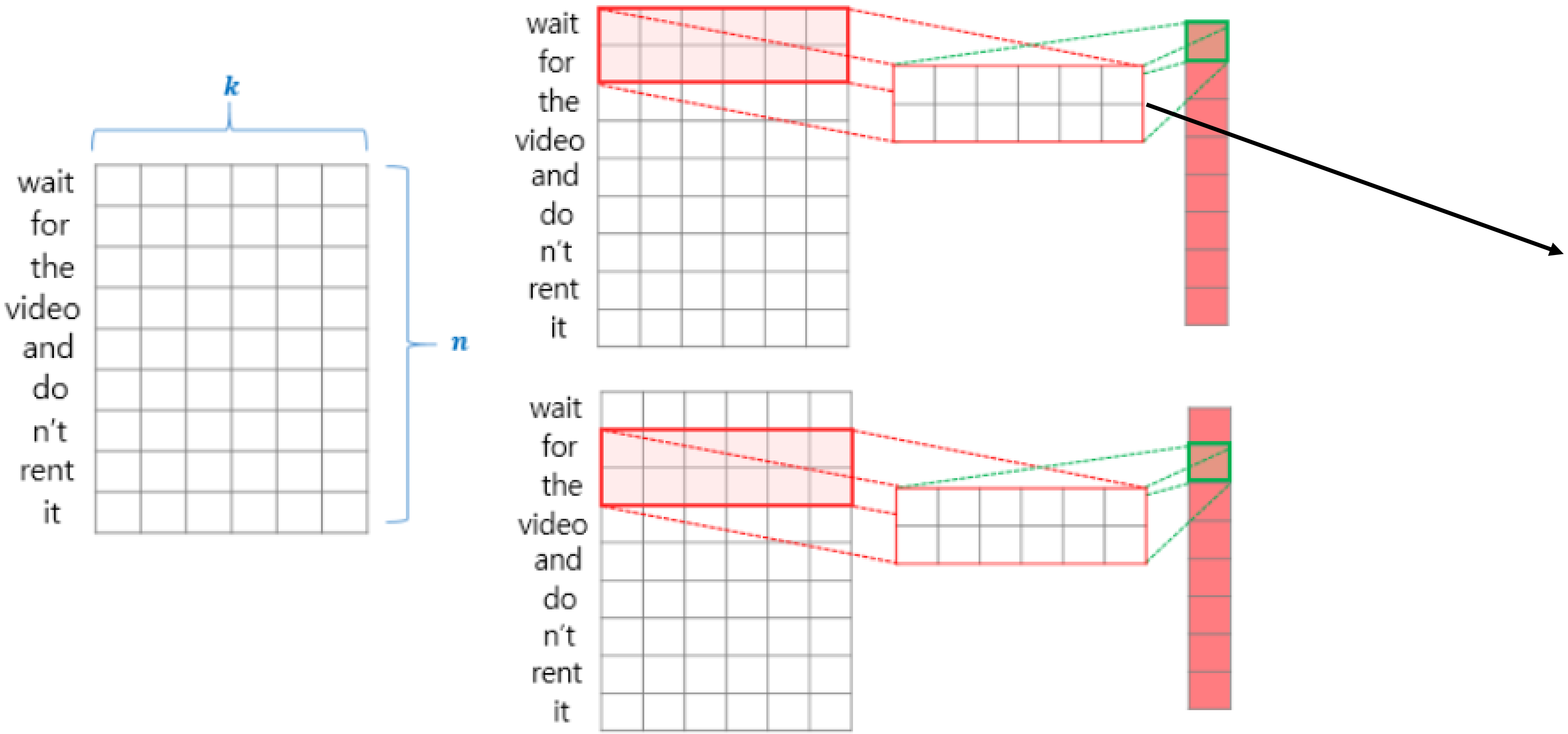
Same weight for every single feature map

Total # weights = (5*5*3)*30

= **2250**

1D CNN

With using the local connectivity of CNN, we can adjust it for NLP field.

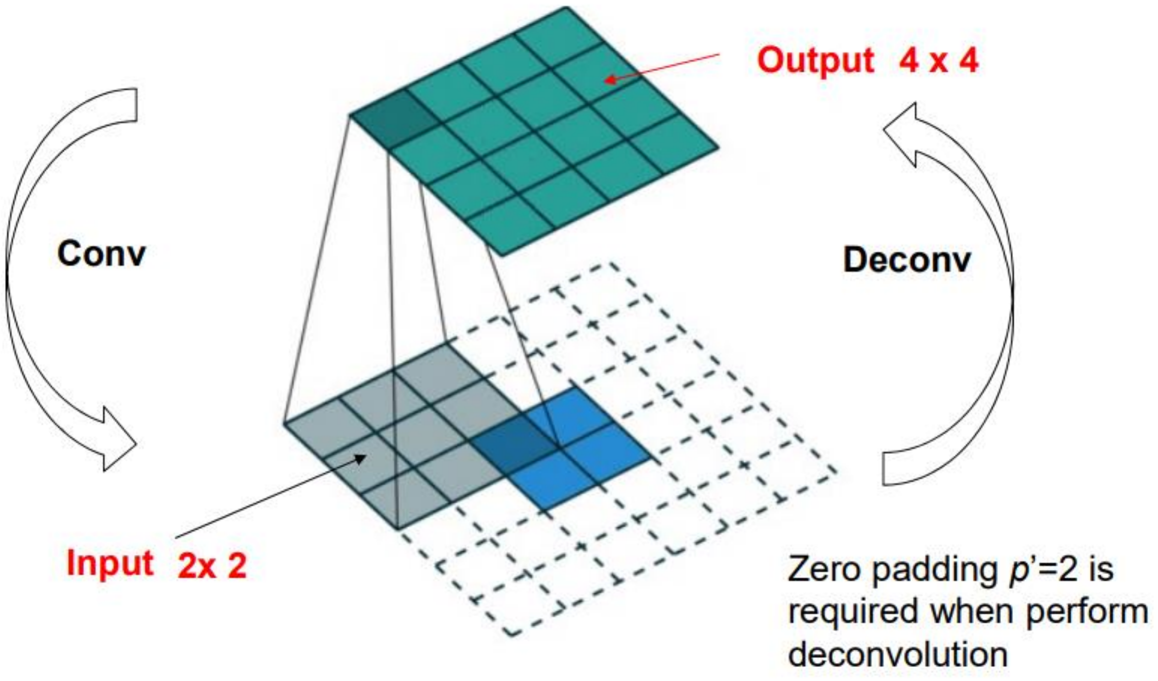


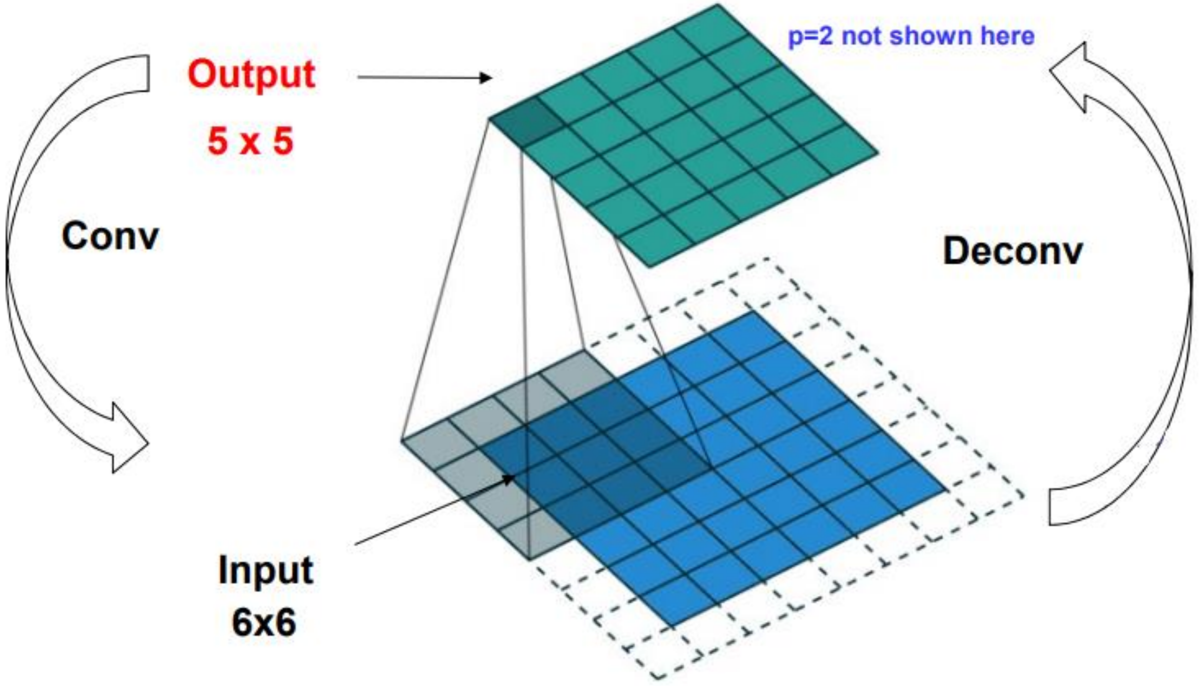
Kernel size can be considered as window size

Deconvolution

CNN also can be used for generate the novel image.

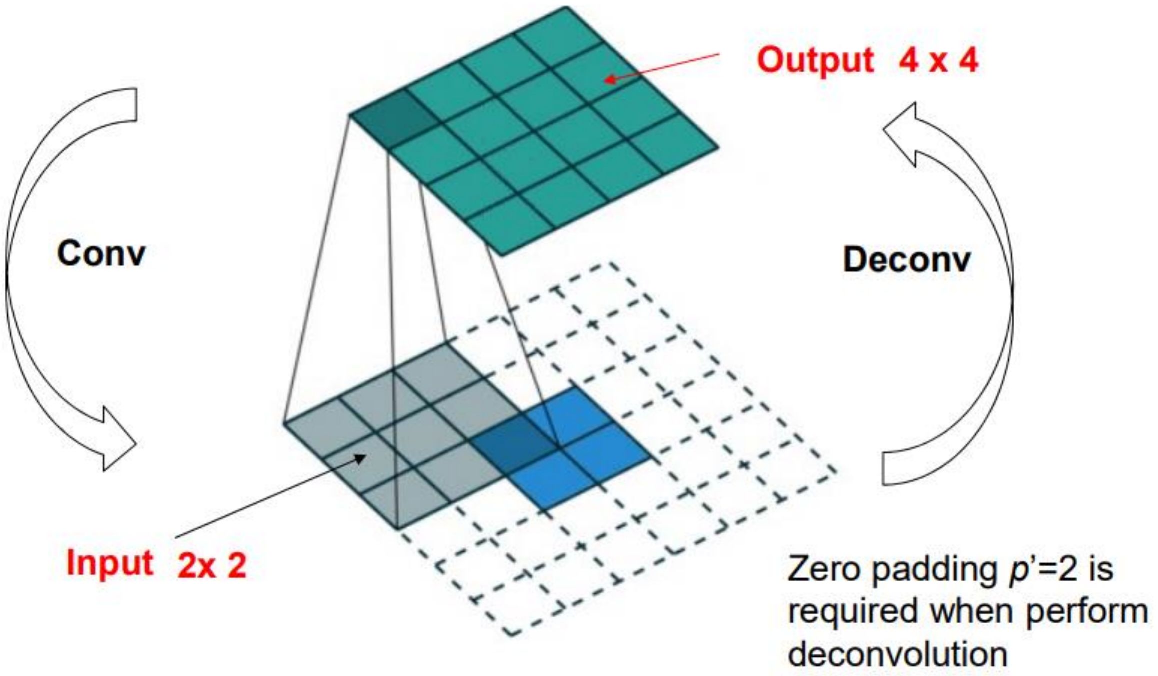
We can use **Deconvolution** to do **Image Segmentation** and **Super-Resolution**





Deconvolution

Deconvolution for non-padded feature map



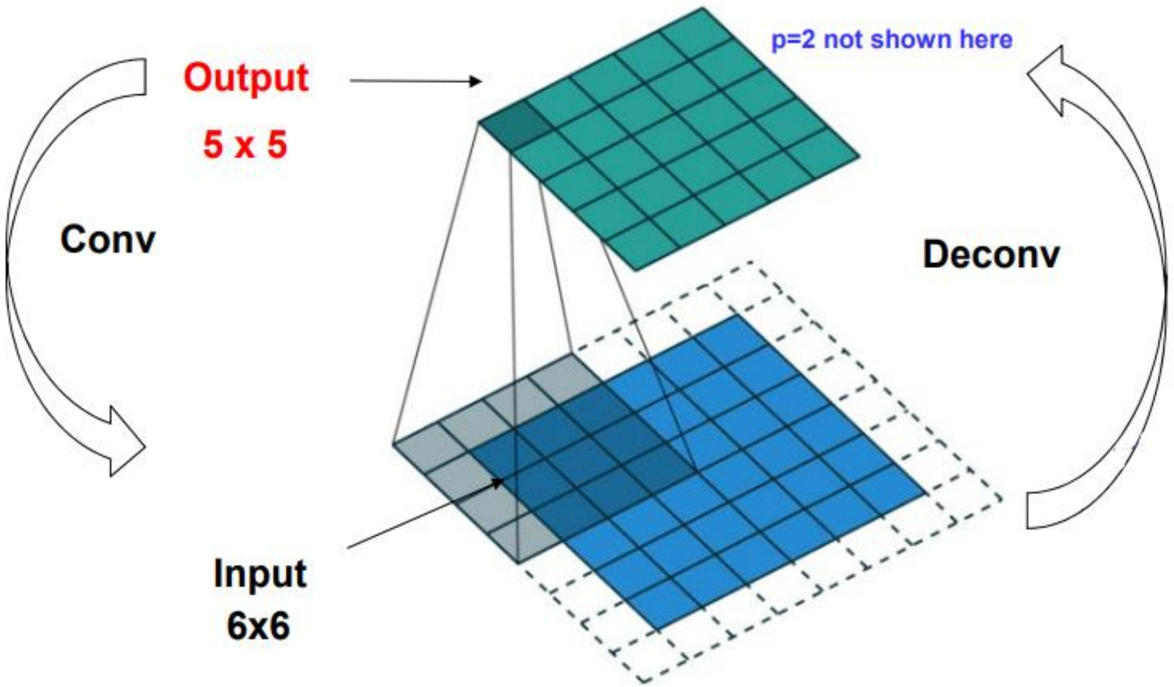
Recall from CNN calculation : $o = \left\lfloor \frac{i + 2p - k}{s} \right\rfloor + 1$

If $\mathrm{p} = 0$, $p' = k - 1$, $o' = s(i' - 1) + k$

$\boldsymbol{o'} = \boldsymbol{s}(\boldsymbol{i'} - \mathbf{1}) + \boldsymbol{k} = 1(2 - 1) + 3 = 4$

Deconvolution

Deconvolution for padded feature map



Recall from CNN calculation : $o = \left\lfloor \frac{i + 2p - k}{s} \right\rfloor + 1$

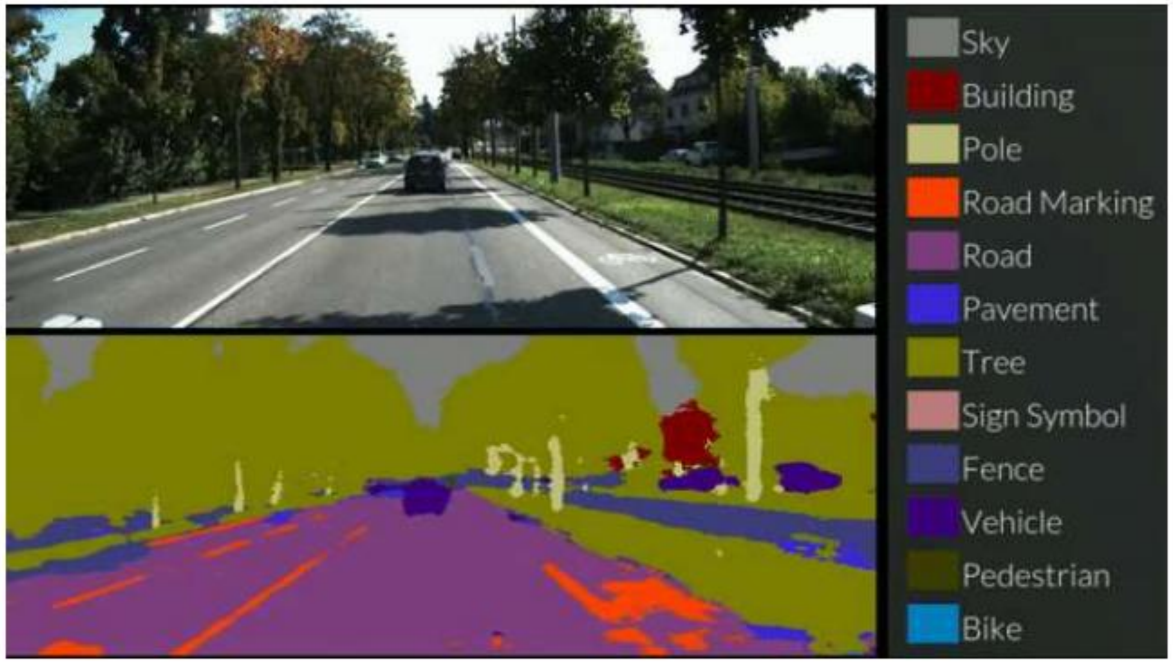
If $\mathrm{p} = 2$, $p' = k - \mathrm{p} - 1$, $o' = s(i' - 1) + k$

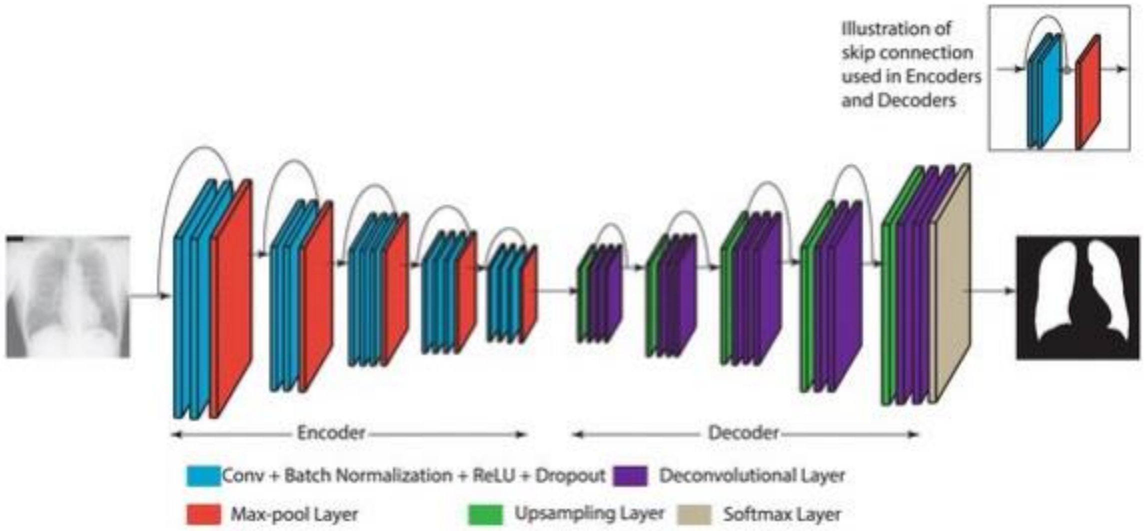
$\boldsymbol{o' = s(i' - 1) + k - 2p} = 1(6 - 1) + 4 - 4 = 5$

Deconvolution

Application

Dense pixel-wise predictions such as semantic segmentation, computing optical flow and disparity maps, contour detection etc.

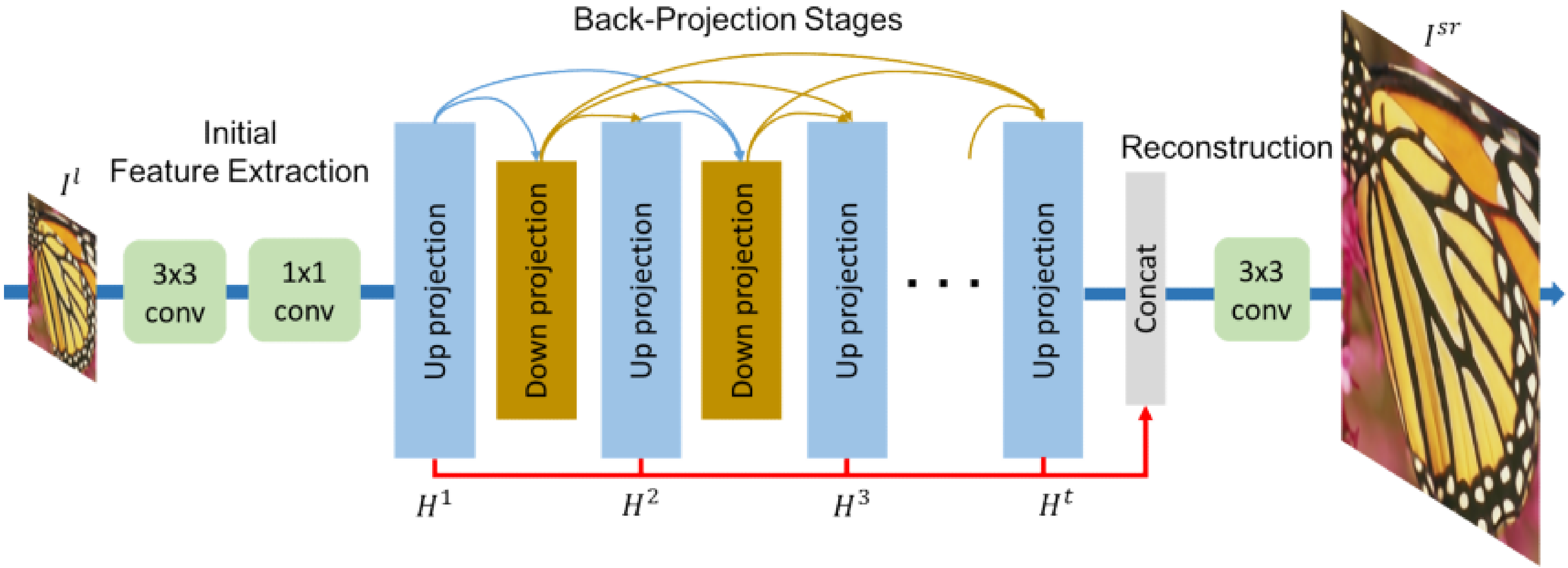




Deconvolution

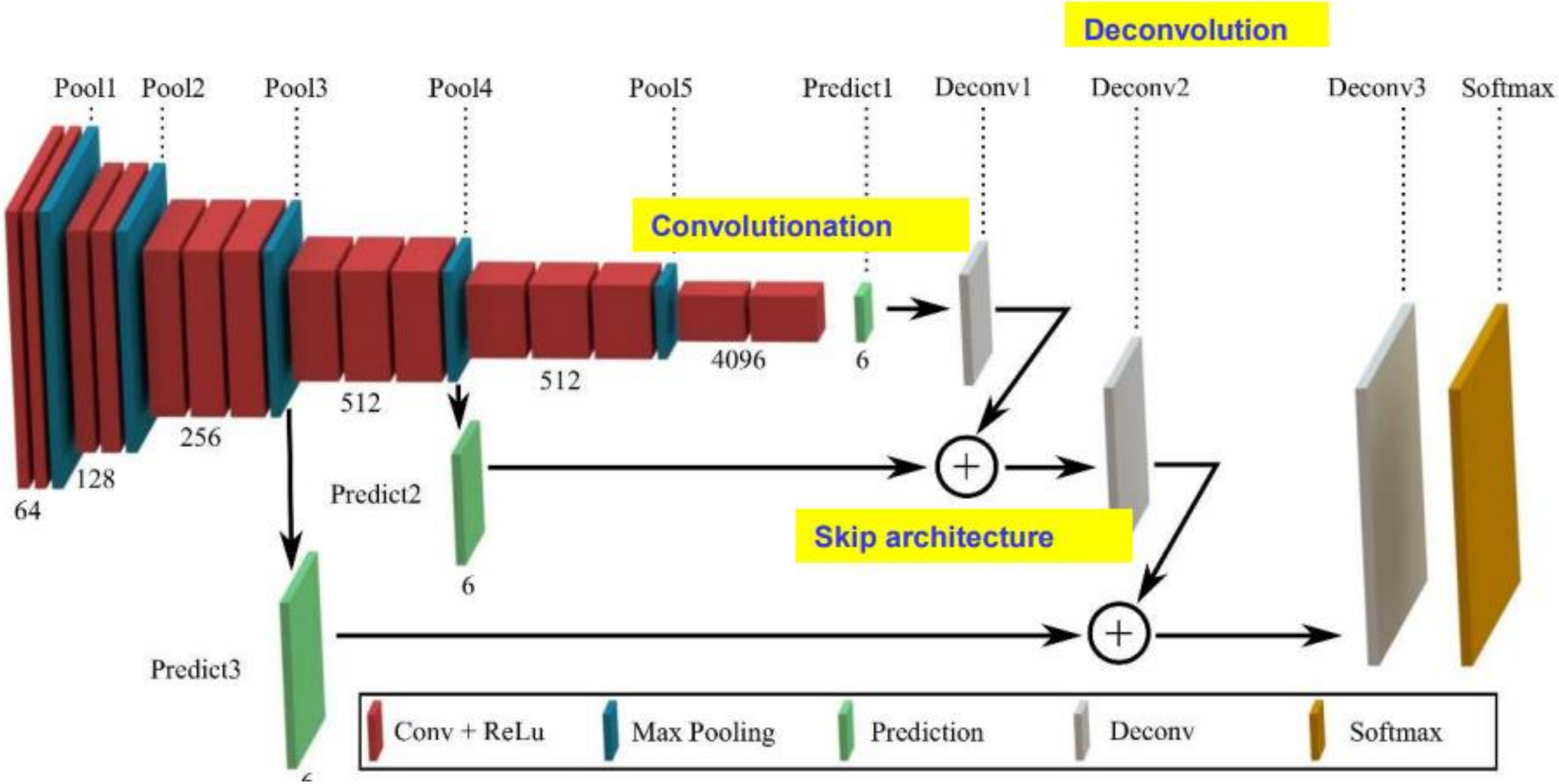
Application

It also can make a high-resolution image from low resolution image



Deconvolution

FCN (Fully Convolution Network)



Deconvolution

FCN (Fully Convolution Network)

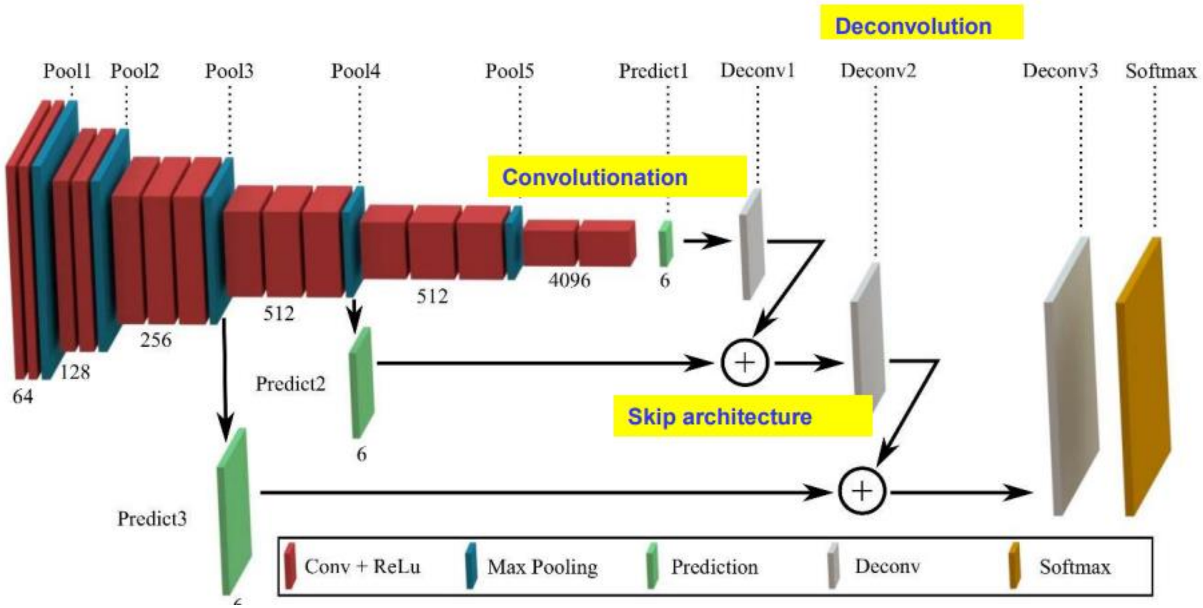
First End-to-End CNN architecture for semantic segmentation

Build on top of VGG backbone

Convolutionation

Deconvolution

Skip Architecture



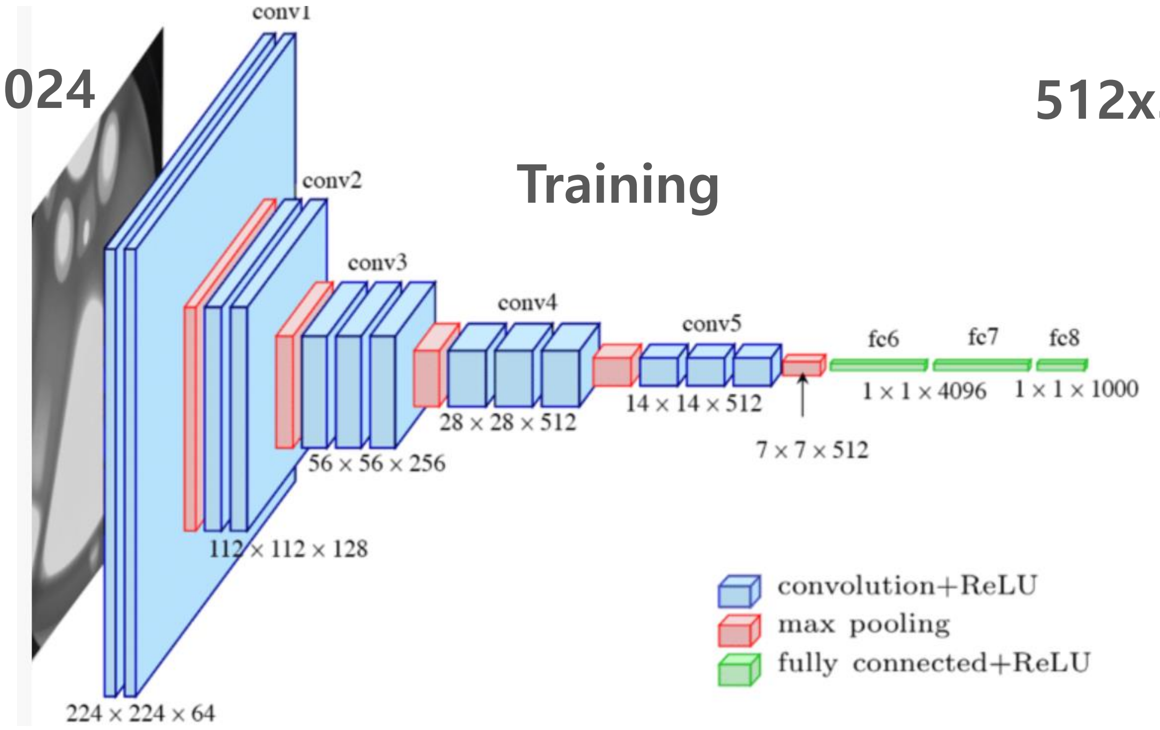
Deconvolution

Convolutionation

We **only** store the **'Weights value'** after training

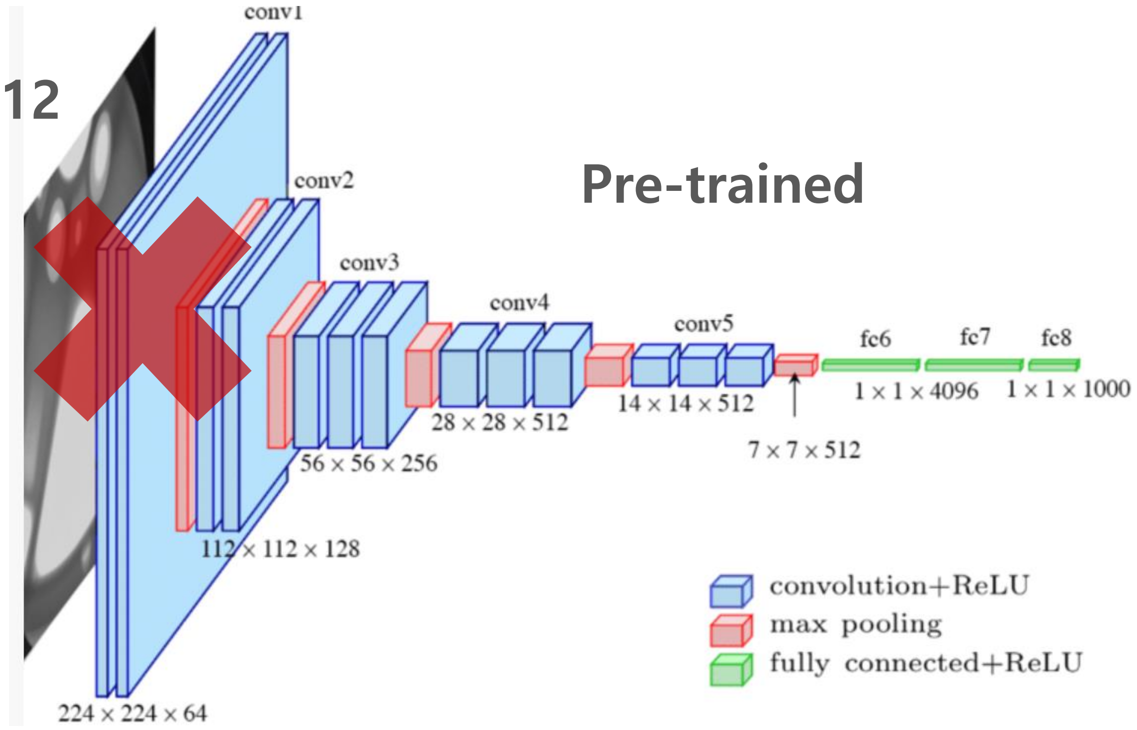
So, if we feed different size image to pre-trained model, there are some **dimension issue**

1024x1024



Training

512x512



Pre-trained

Deconvolution

Convolutionation

Training

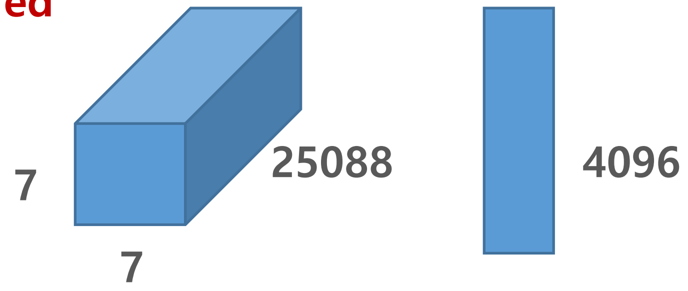
Training Input	250*250*3
Feature Map Size (last layer)	7*7*512 = 25088
FC Layer	4096
# of weights	4096*25088

Inference

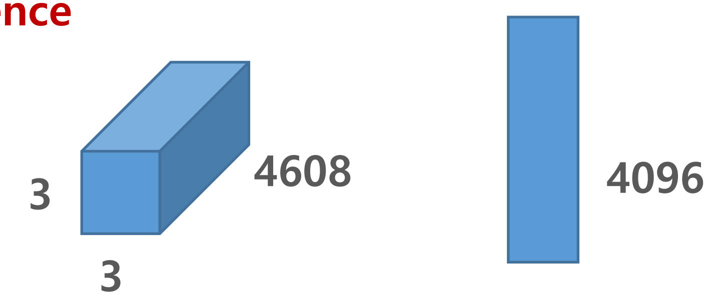
Training Input	150*150*3
Feature Map Size (last layer)	3*3*512 = 4608
FC Layer	4096
# of weights	4096*4096

Doesn't Matched !!

Stored



Inference



Deconvolution

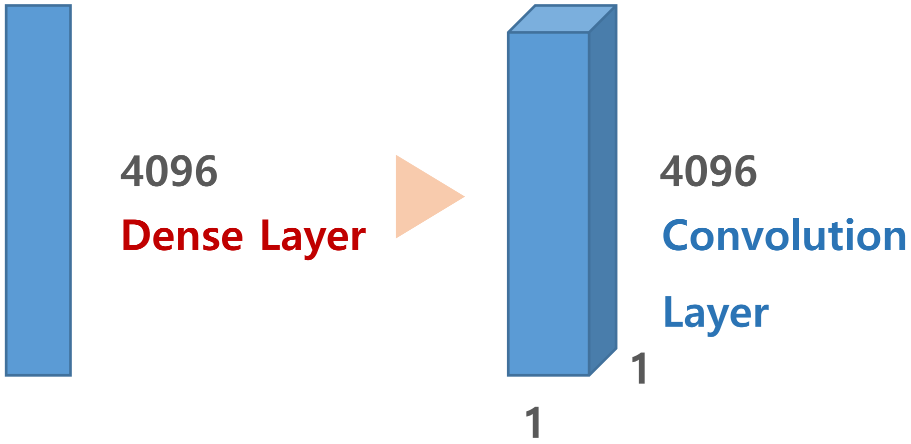
Convolutionation

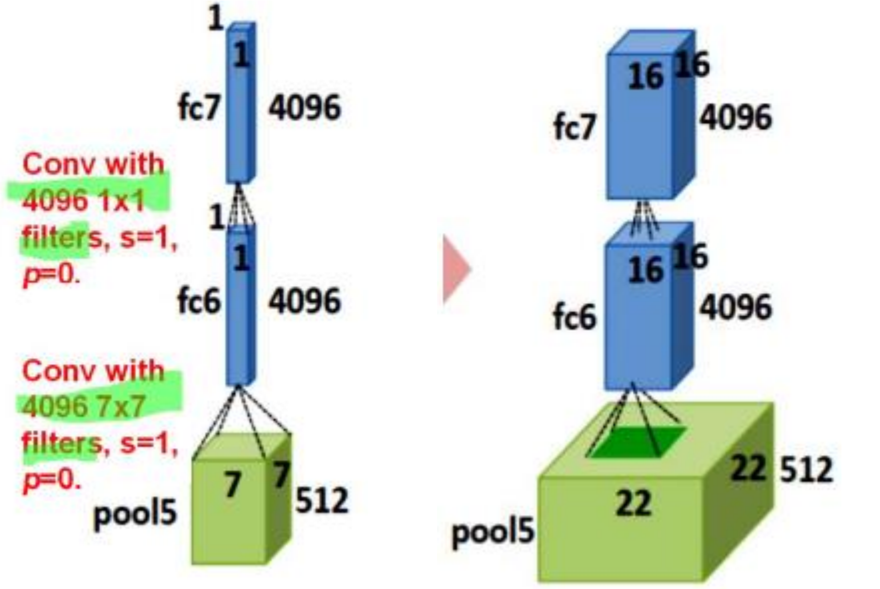
The weights of **FC layer vary with input vector size**

So, if the size of an input vector change, pre-trained model cannot work

But weights of **Convolution layer doesn't change** with various input sizes

Only the **size of kernel** determine the # of weights





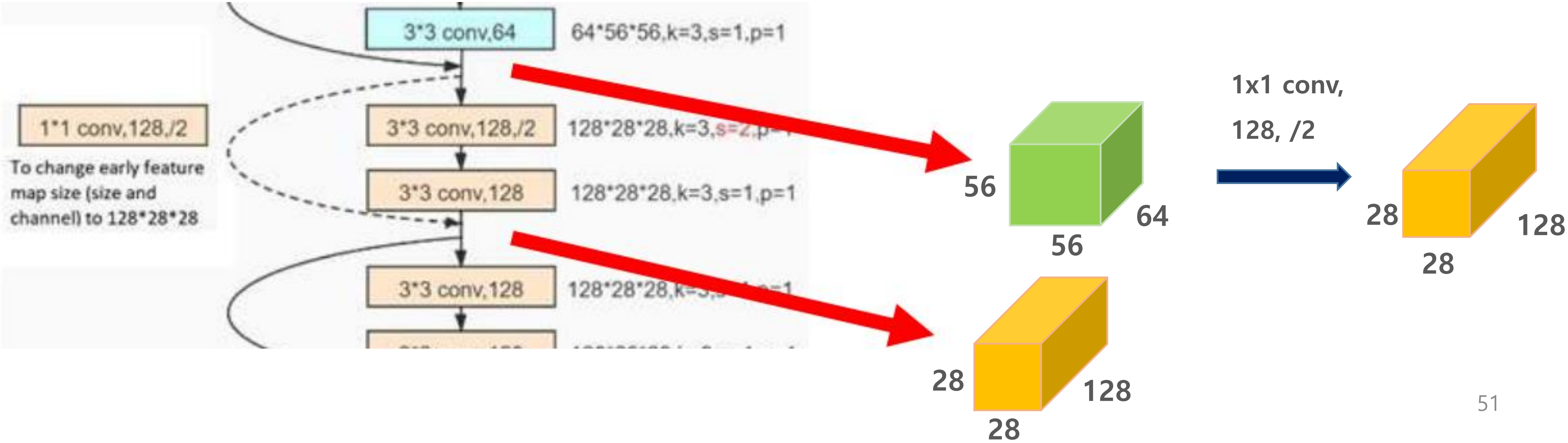
Convolution layers For larger feature map

Deconvolution

1x1 Convolution for Skip-Connection

For skip-connection or the other **skip-architecture** has one problem

They conduct the elementwise calculation, but the **size of feature map** for skipped one and pass the conv. layer one **do not matched**.



Transfer Learning

Concept of Transfer Learning

Myth in Deep Learning

You can't do deep learning unless you have a million labeled examples for your problem

Reality

You can learn useful representations from unlabeled data

You can train on a nearby surrogate objective for which it is easy to generate labels

You can **transfer** learned representations from a related task

→ **Transfer Learning, Few-shots learning**

Ref : Neural Network, Lecture Note, Yonsei University Electrical & Electronic Engineering, Andrew Theo Bengjin

Concept of Transfer Learning

Definition

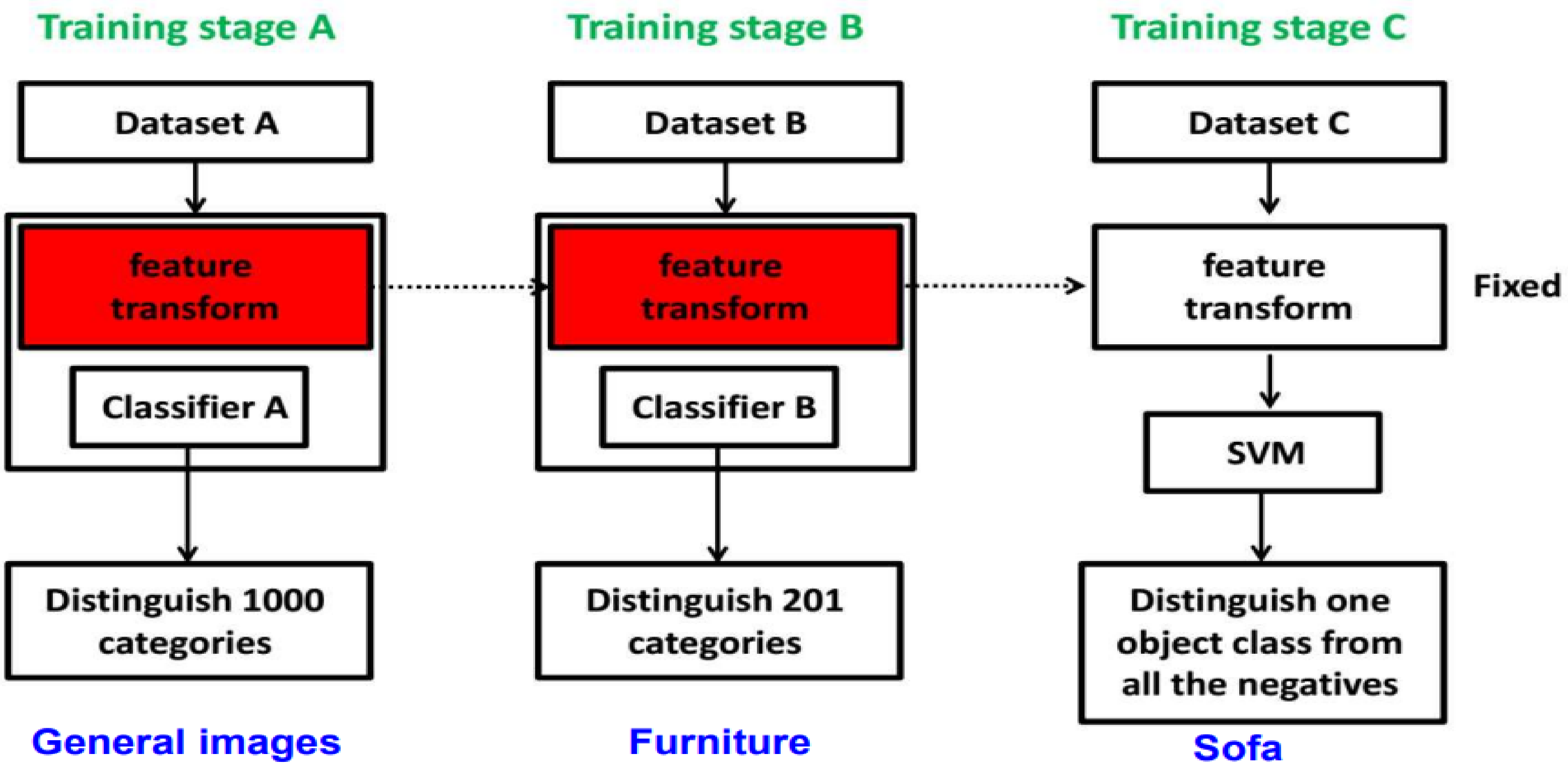
The ability of a system to recognize and apply knowledge and skills learned in previous task to novel tasks (in new domains).

It is motivated by **human learning**. People can often transfer knowledge learnt previously to novel situations

- Know how to ride a motorbike → Learn how to ride a car
- Know how to play classic piano → Learn how to play jazz piano

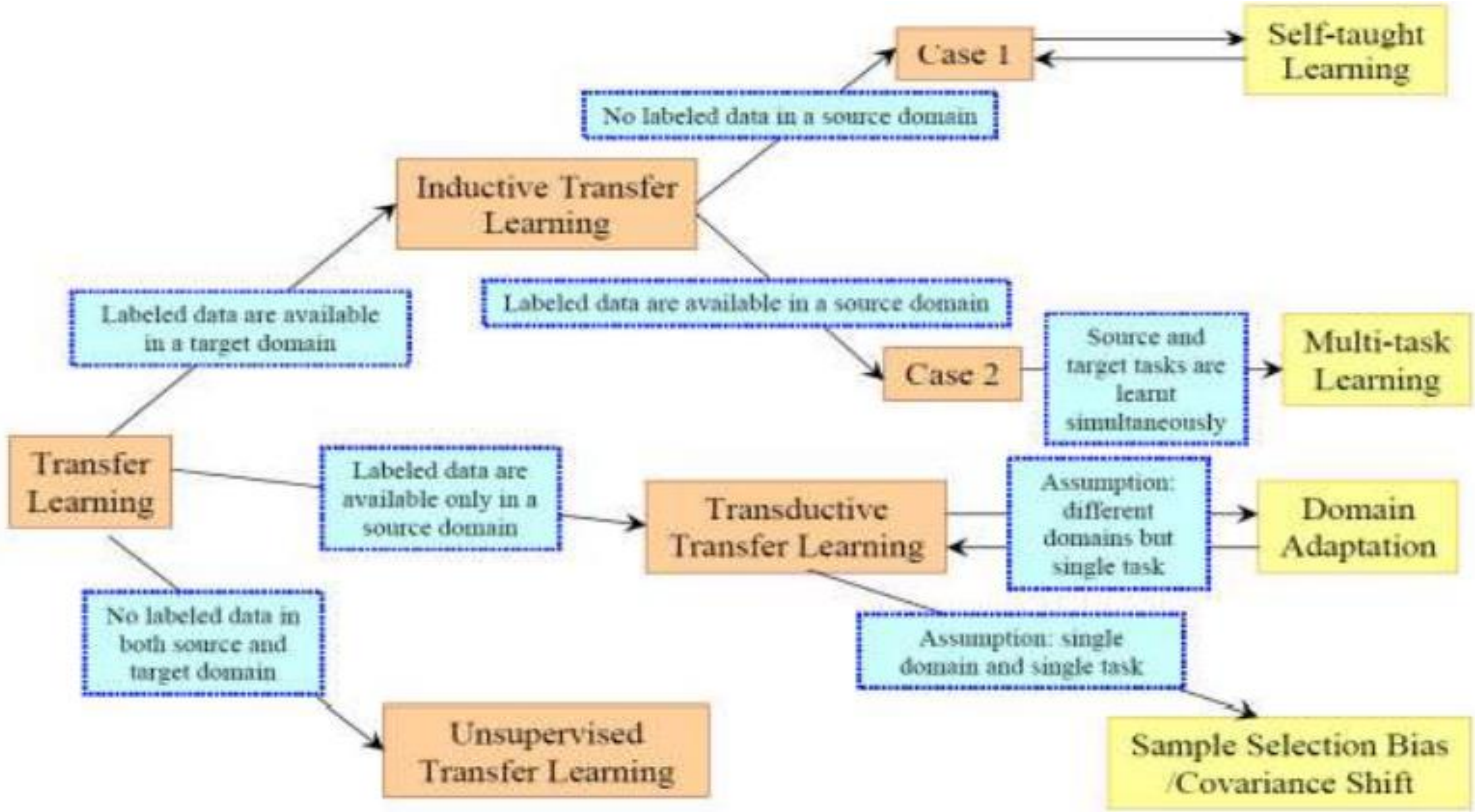
Ref : Neural Network, Lecture Note, Yonsei University Electrical & Electronic Engineering, Andrew Theo Bengjin

Schematic Overview of TL



Transfer Learning

Way for Transfer Learning



Ref : J. Pan and Q. Yang, "A survey on Transfer Learning", IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345-1359, Oct.2010

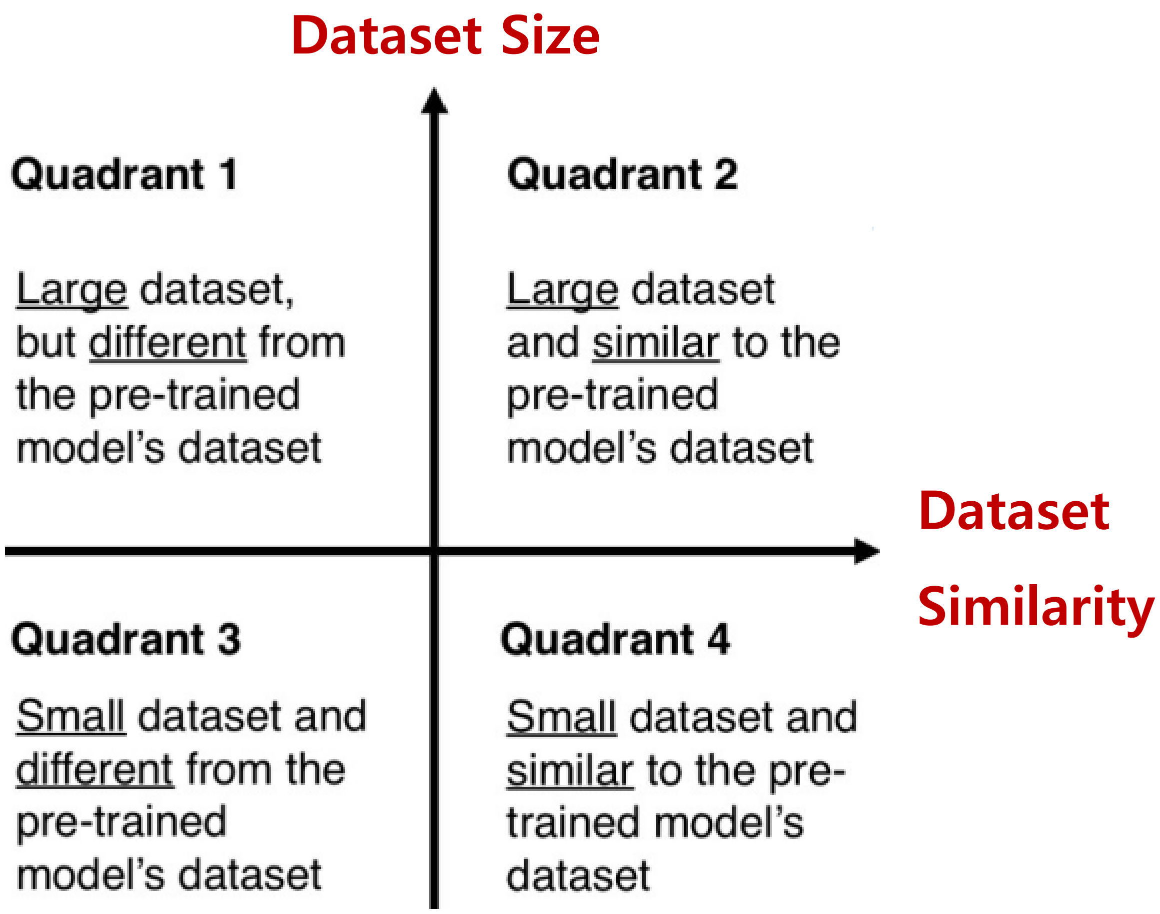
Transfer Learning

Size Similarity Matrix

The Guideline of basic Transfer Learning

X axis is Dataset Similarity

Y axis is Dataset Size



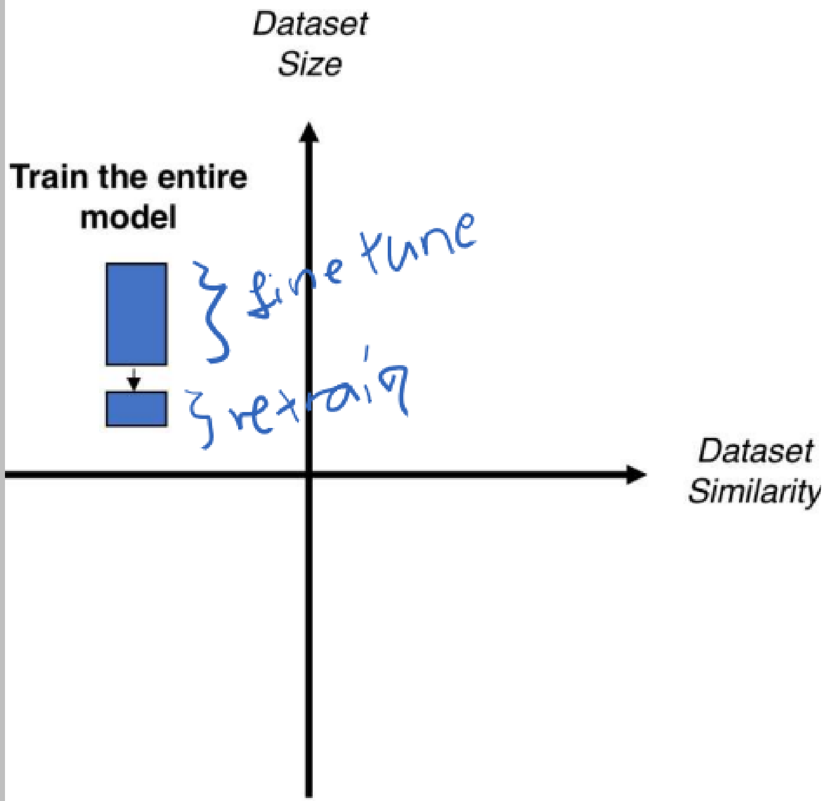
Size-Similarity Matrix

Ref : Neural Network, Lecture Note, Yonsei University Electrical & Electronic Engineering, Andrew Theo Bengjin

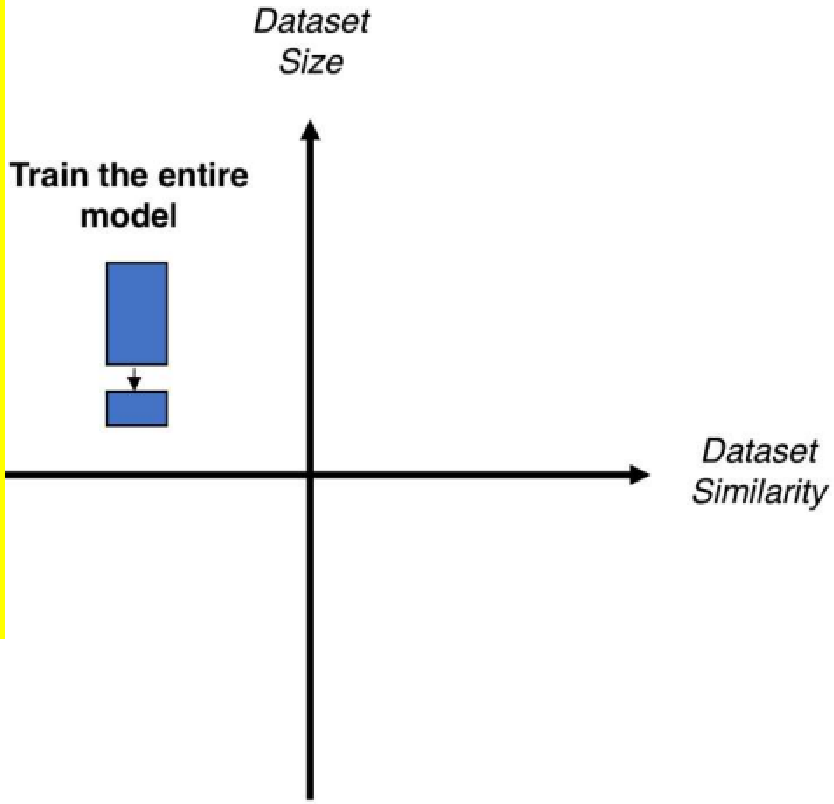
Transfer Learning

Transfer Learning Process

- Since the dataset is very large, we may expect that we can afford to train a ConvNet from **scratch.**
- However, in practice it is very often still beneficial to initialize with weights from a pretrained model.
- In this case, we would have enough data and confidence to fine-tune through the entire network.



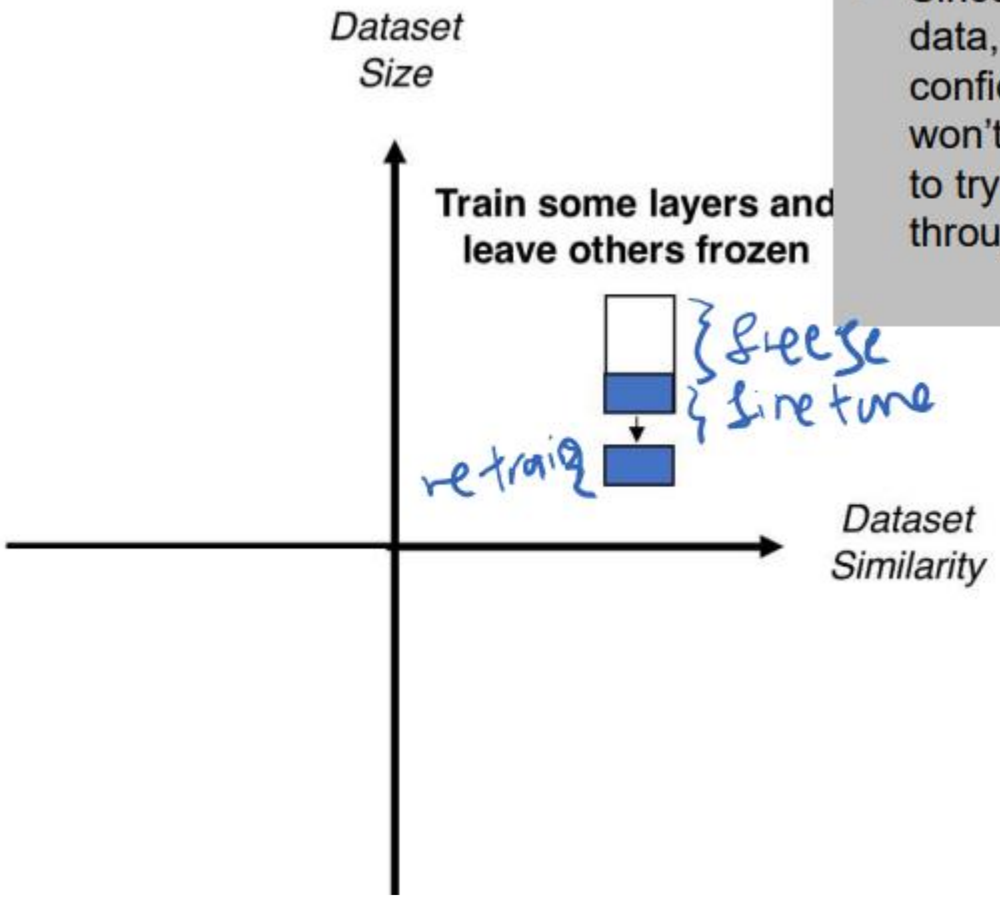
- Another perspective, this practice also can be seen as an another “initialization” method.
- Pre-trained models to provide a “good initial” point.
- Then fine tune + retrain is used to train the network with training data of interest.



Ref : Neural Network, Lecture Note, Yonsei University Electrical & Electronic Engineering, Andrew Theo Bengjin

Transfer Learning

Transfer Learning Process

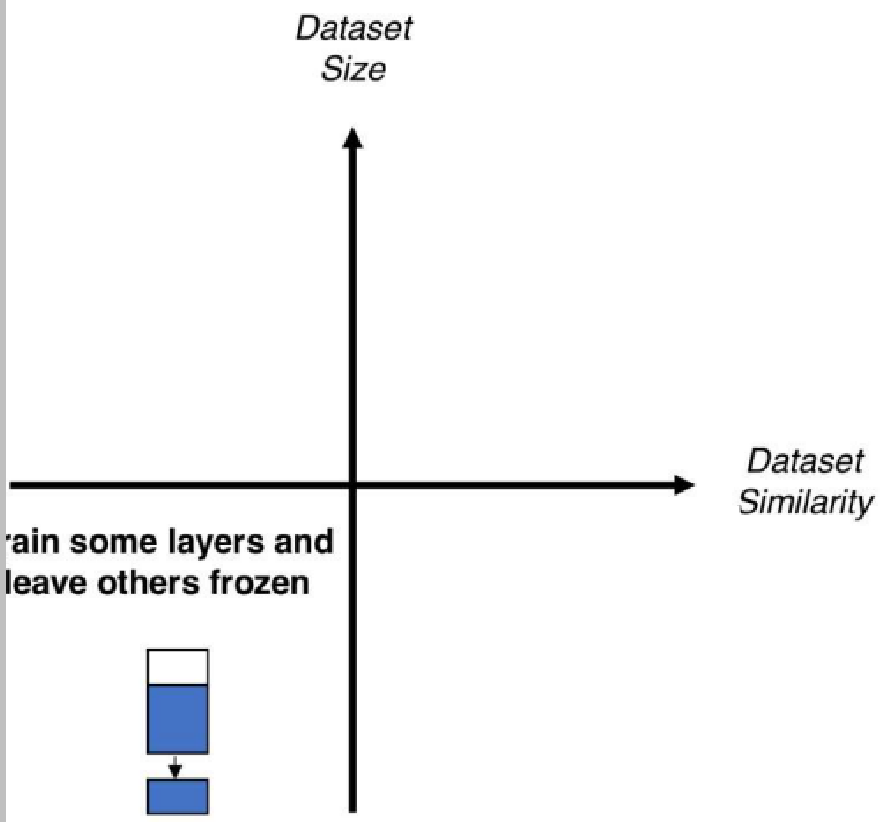


- Since we have more data, we can have more confidence that we won't overfit if we were to try to fine-tune through the full network.

Ref : Neural Network, Lecture Note, Yonsei University Electrical & Electronic Engineering, Andrew Theo Bengjin

Transfer Learning Process

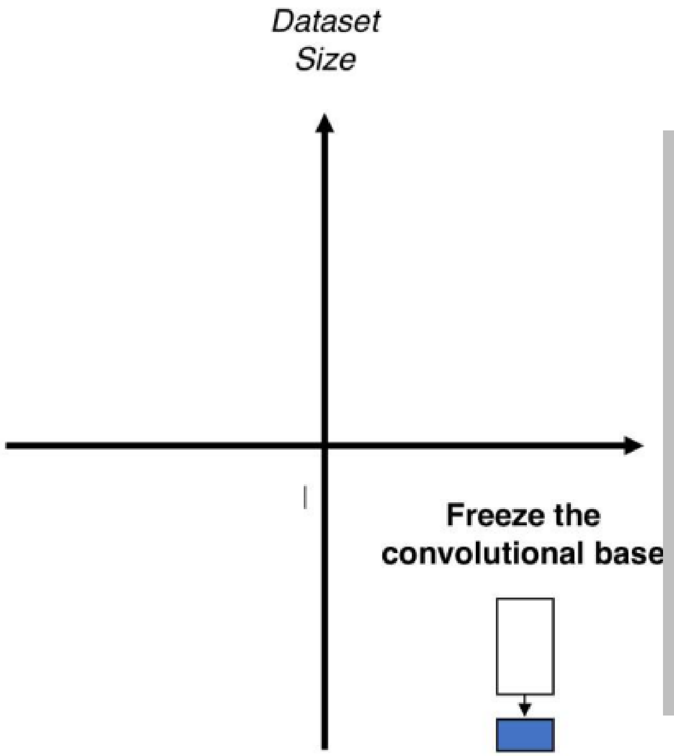
- Since the data is small, it is likely best to only train a linear classifier.
- Since the dataset is very different,, we also have to remove several late convolution layers and retain a few early convolution layers only.
- Alternatively, consider retrain many late conv layers & consider data augmentation.



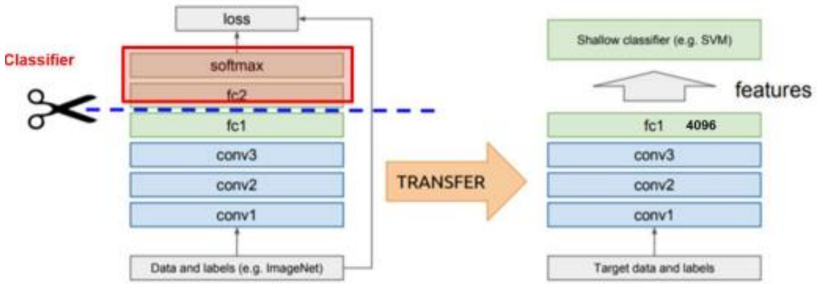
Ref : Neural Network, Lecture Note, Yonsei University Electrical & Electronic Engineering, Andrew Theo Bengjin

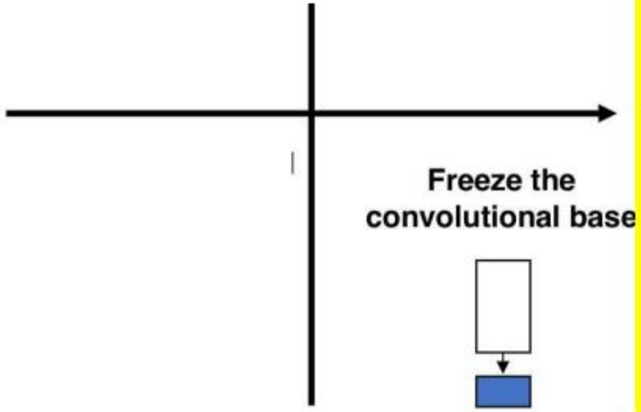
Transfer Learning

Way for Transfer Learning



- Since the data is small, it is not a good idea to fine-tune the ConvNet due to overfitting concerns.
- Since the data is similar to the original data, we expect higher-level features in the ConvNet to be relevant to this dataset as well. Hence, the best idea might be to train a linear classifier on the CNN codes.





- This case also can be seen like we use a pre-trained CNN as a feature extractor without finetuning the convolution layers.
- Classifier still has to be retrained on the target data.
- Example: prebuilt model trained on ImageNet and use to classify small size target dataset..

Ref : Neural Network, Lecture Note, Yonsei University Electrical & Electronic Engineering, Andrew Theo Bengjin

3. Summary

Speed of Convergence

Good convergence is the first thing to consider and followed by generalization issue.

Good convergence doesn't imply good generalization.

First Ladder (most important) : Learning Rate

Second Ladder : Hidden units, mini-batch size, momentum coefficient

Third Ladder (least important) : Number of layers, Learning rate decay, other optimizer hyperparameters

3. Summary

CNN

CNN extracts local information from the data (using certain size kernel and bias) and extract the global information by combining them

Based on this basic principle, it can be used not only for image processing but also for **other areas**

Transfer Learning

TL is a powerful tool when we want to train with a small amount of data, and when we want to slightly change the guaranteed models.

Thank you