04.10.2024

Statistical Methods in AI (CS7.403)

Lecture-16: CNN/Deep Learning - 2

Ravi Kiran (ravi.kiran@iiit.ac.in)

https://ravika.github.io

@vikataravi

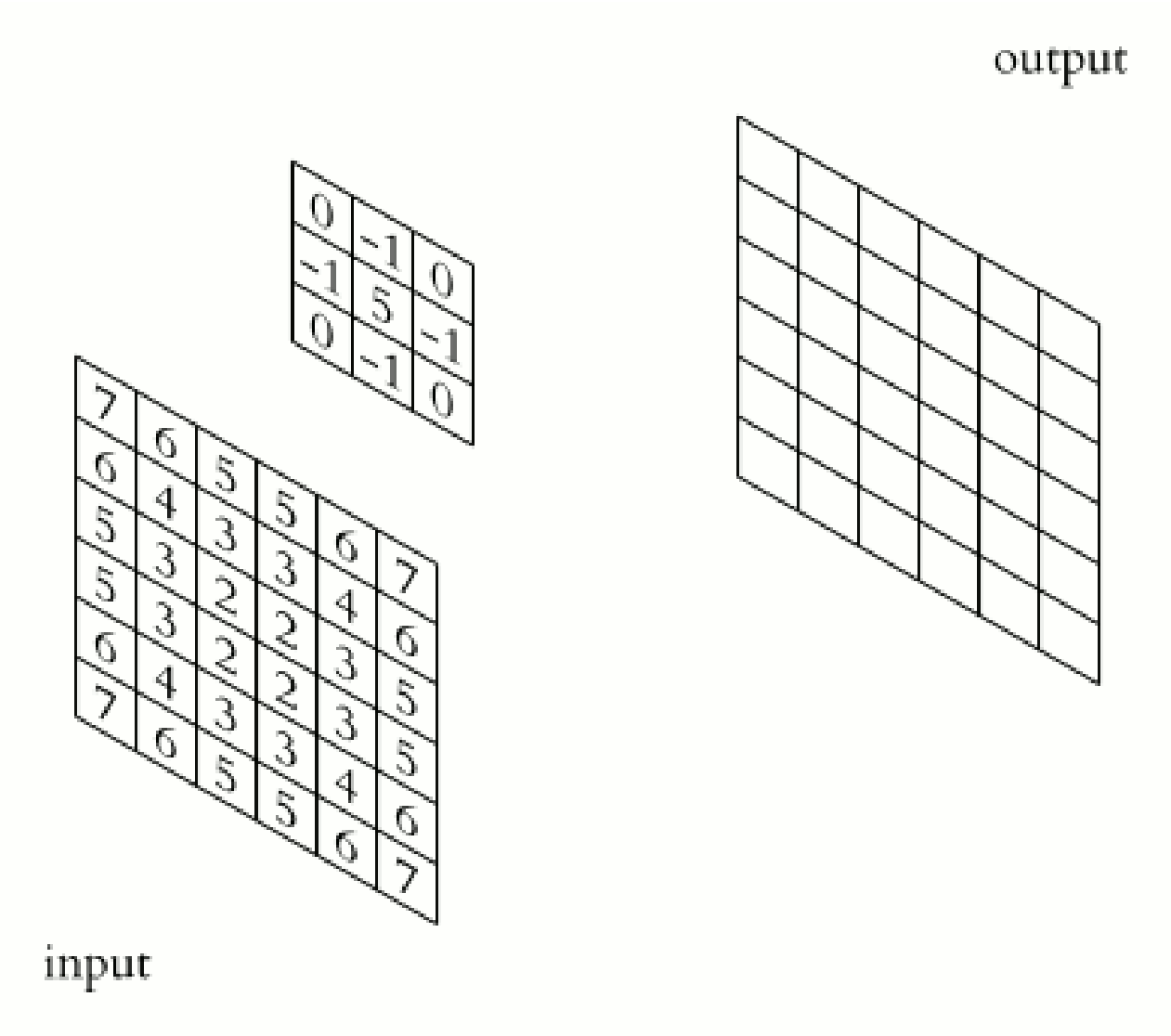
Center for Visual Information Technology (CVIT)
IIIT Hyderabad

Wouldn't it be nice if we could feed the data directly ?

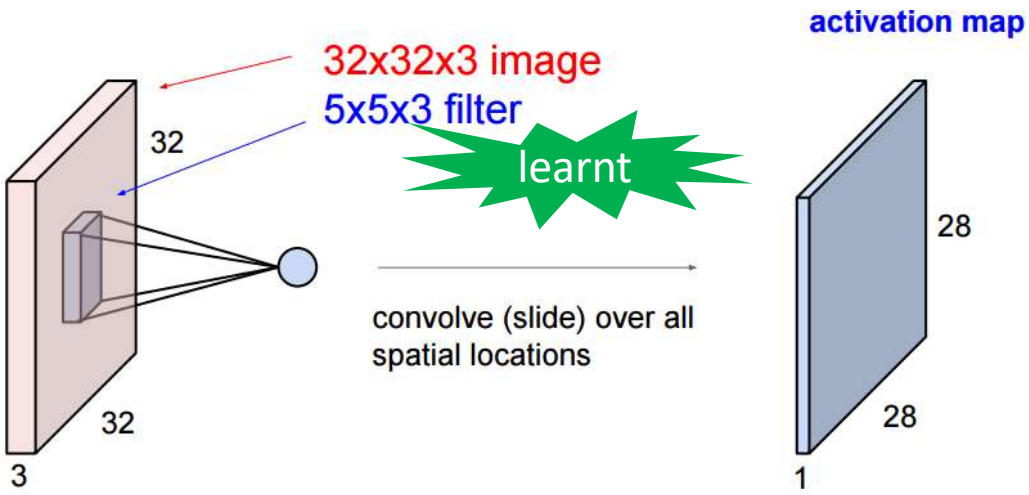
… and let the "learning process" figure out which features to extract ?

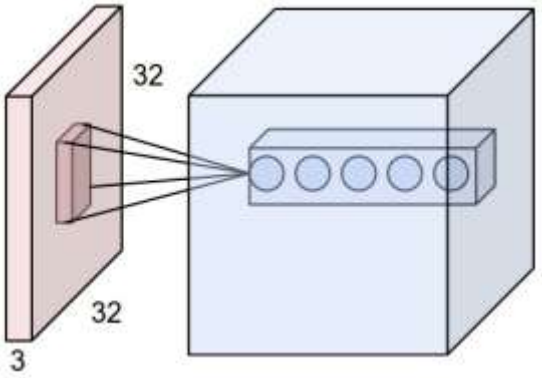
(which filters to construct)

Convolution (in general)

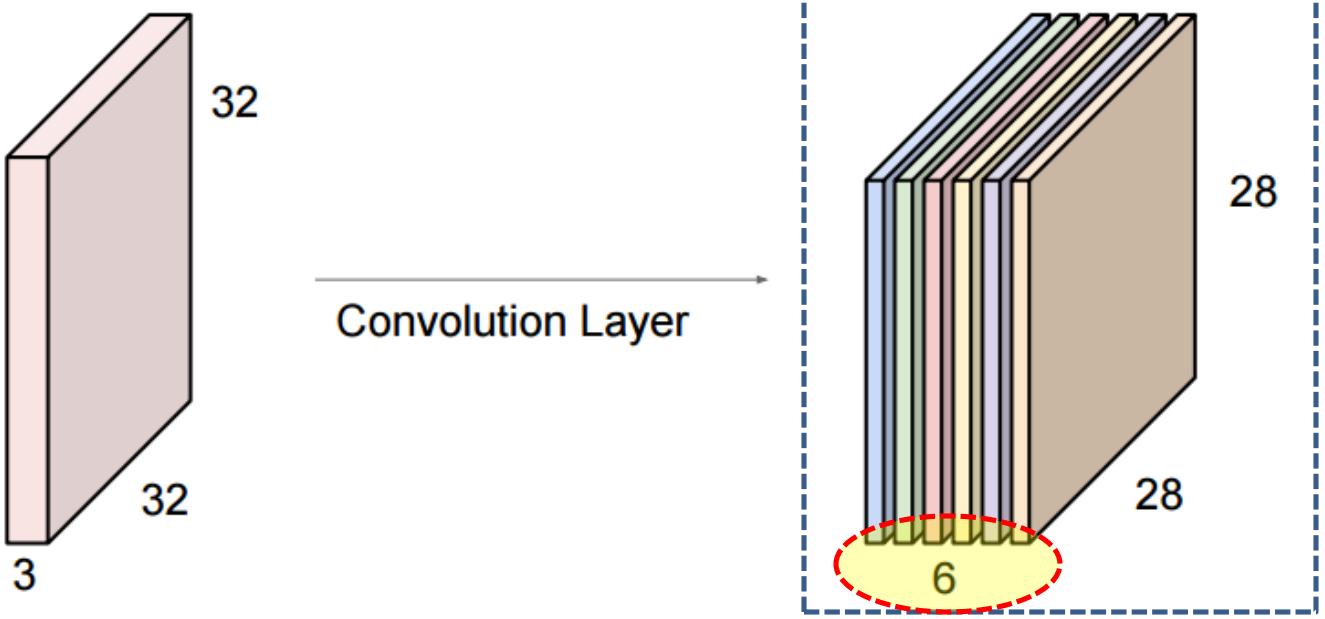


Convolution Layer

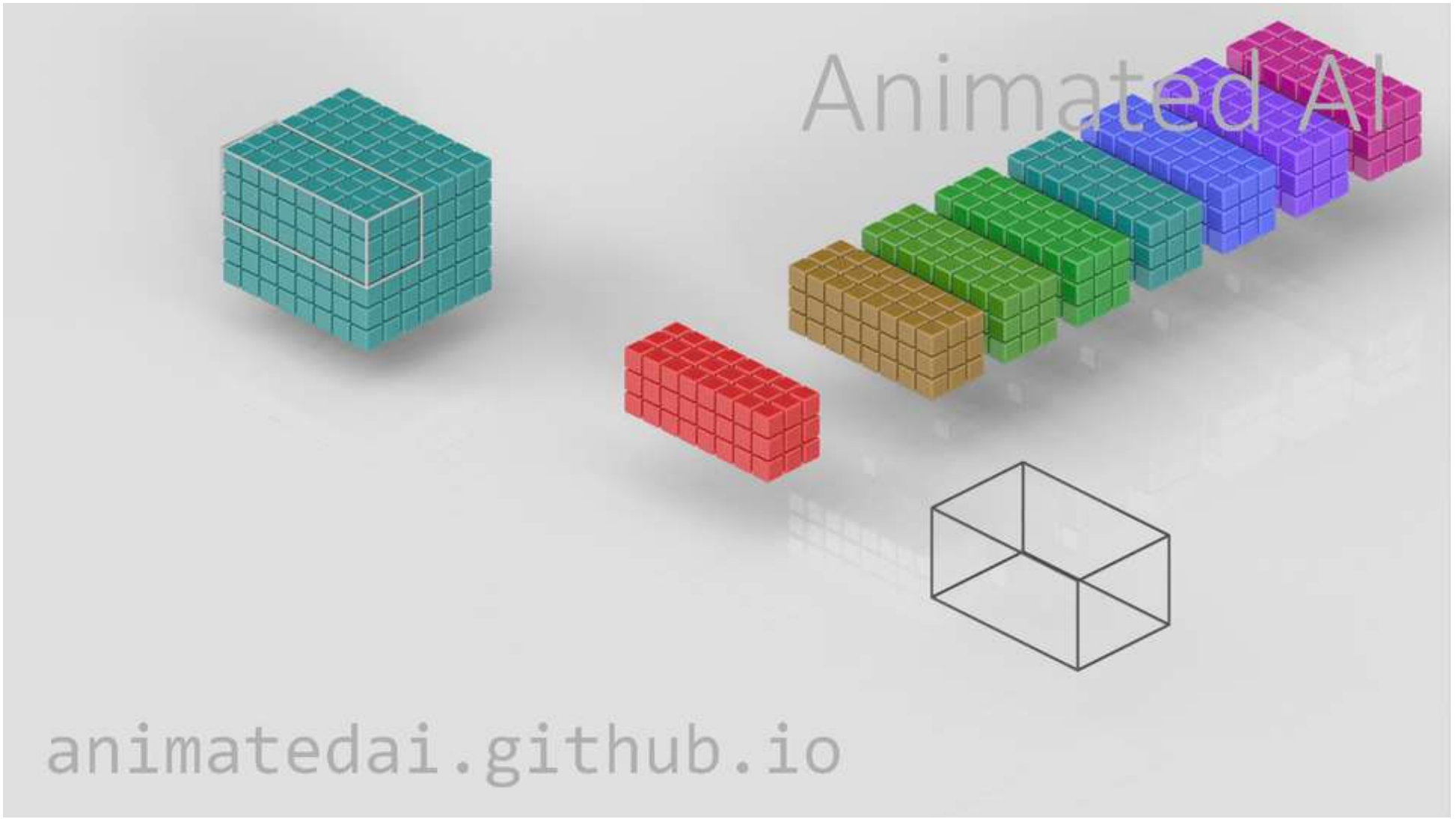




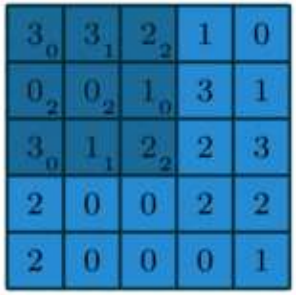
Input to next layer

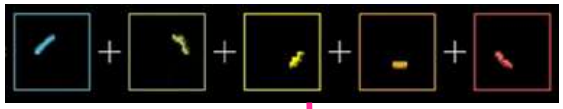


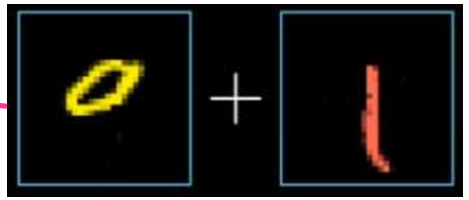
Convolution (in general)

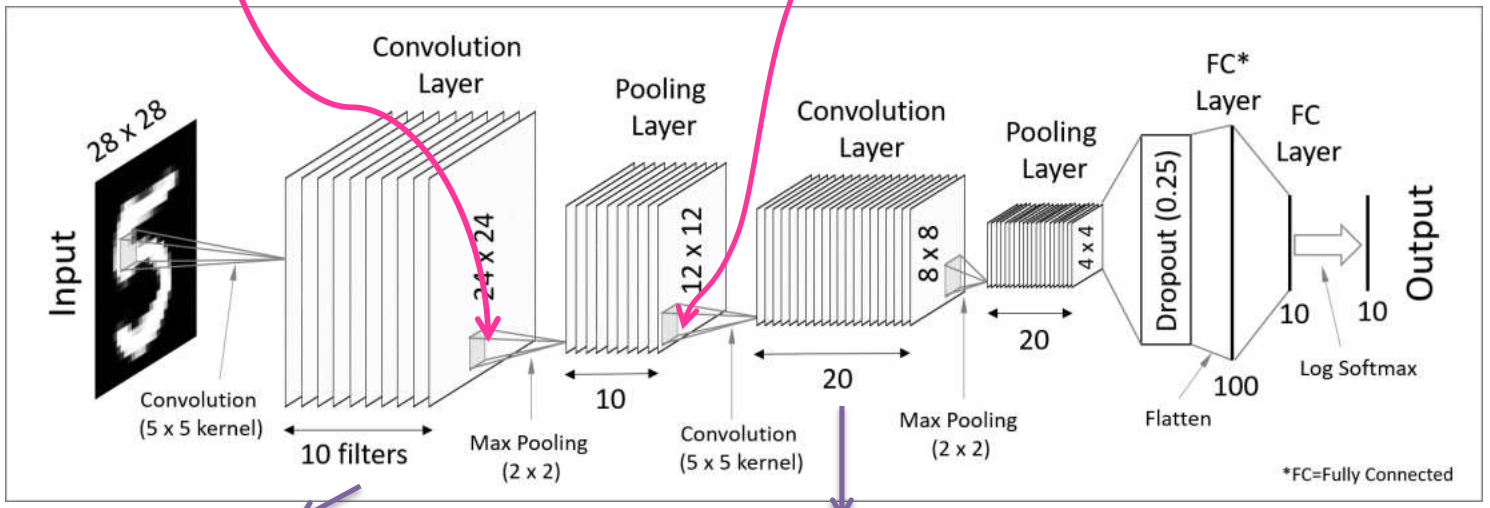


Design choices : Filter size, # of filters





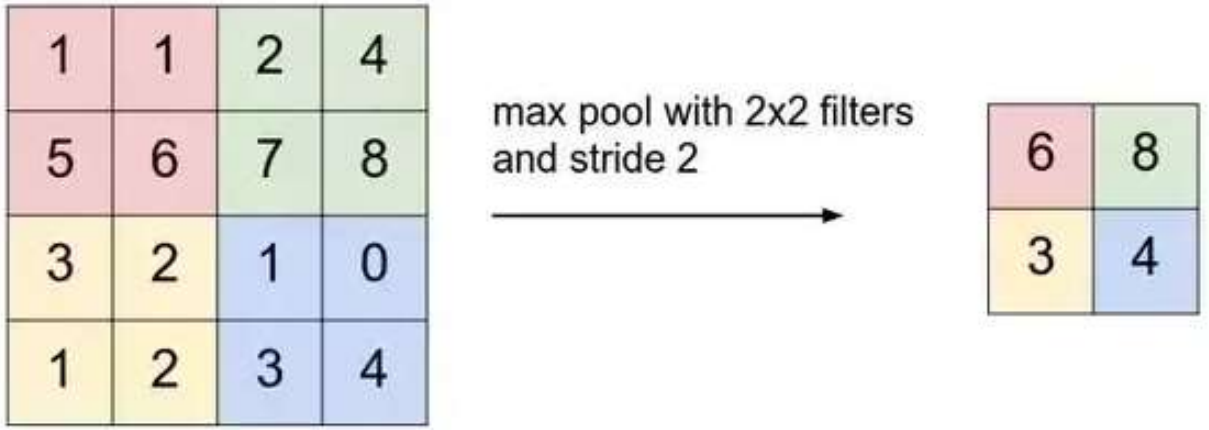




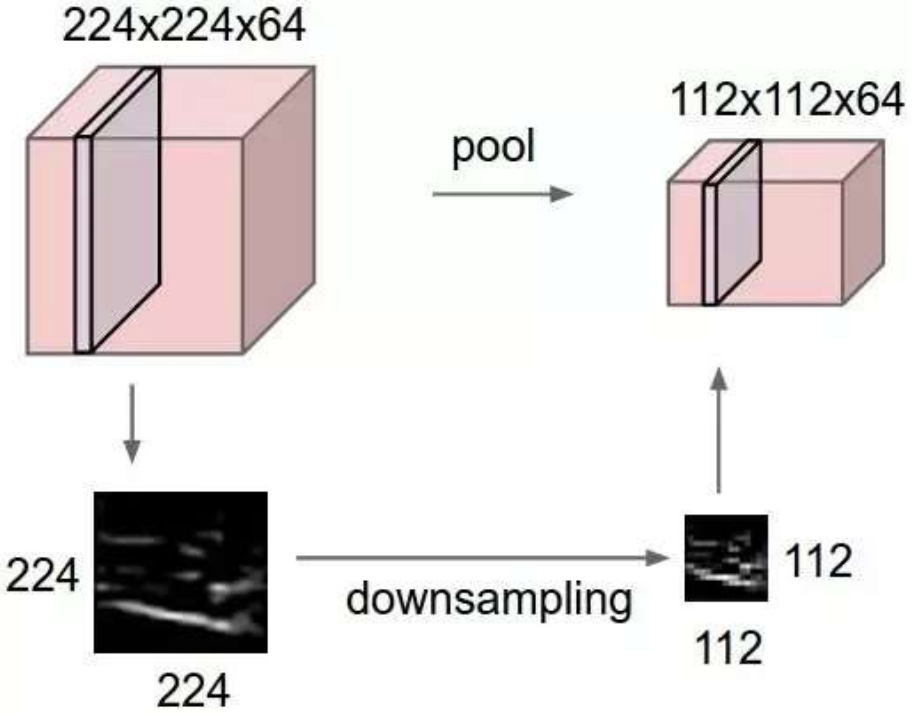
~ Alphabets

~ Words

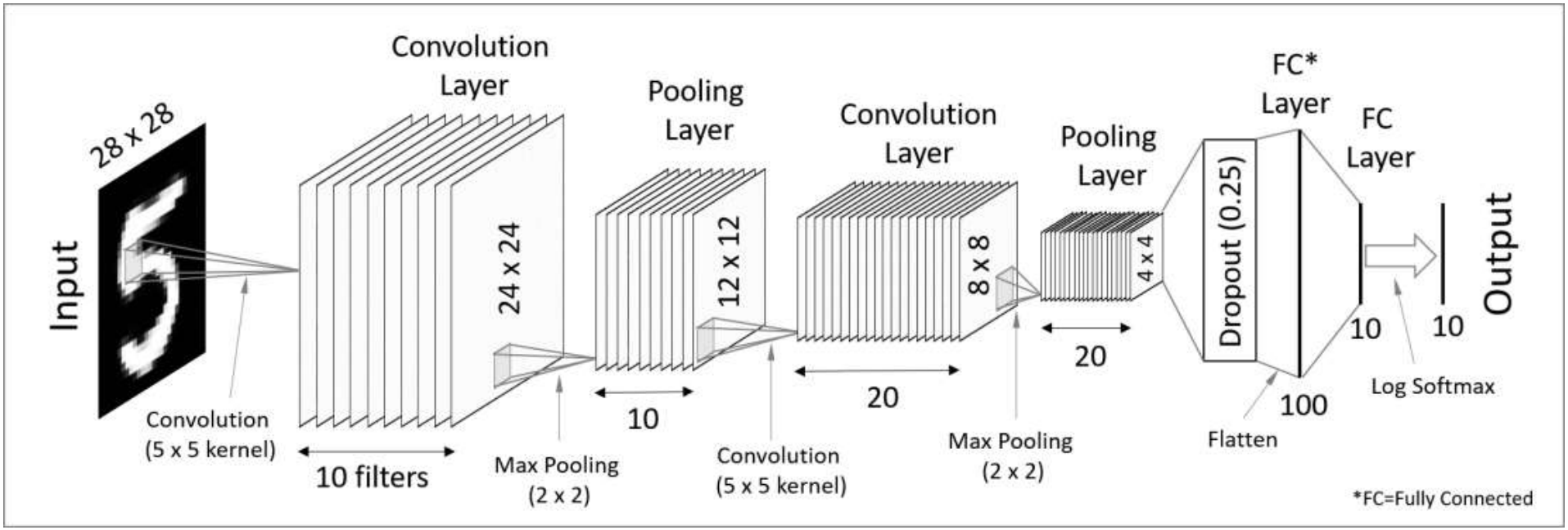
Pooling Layer



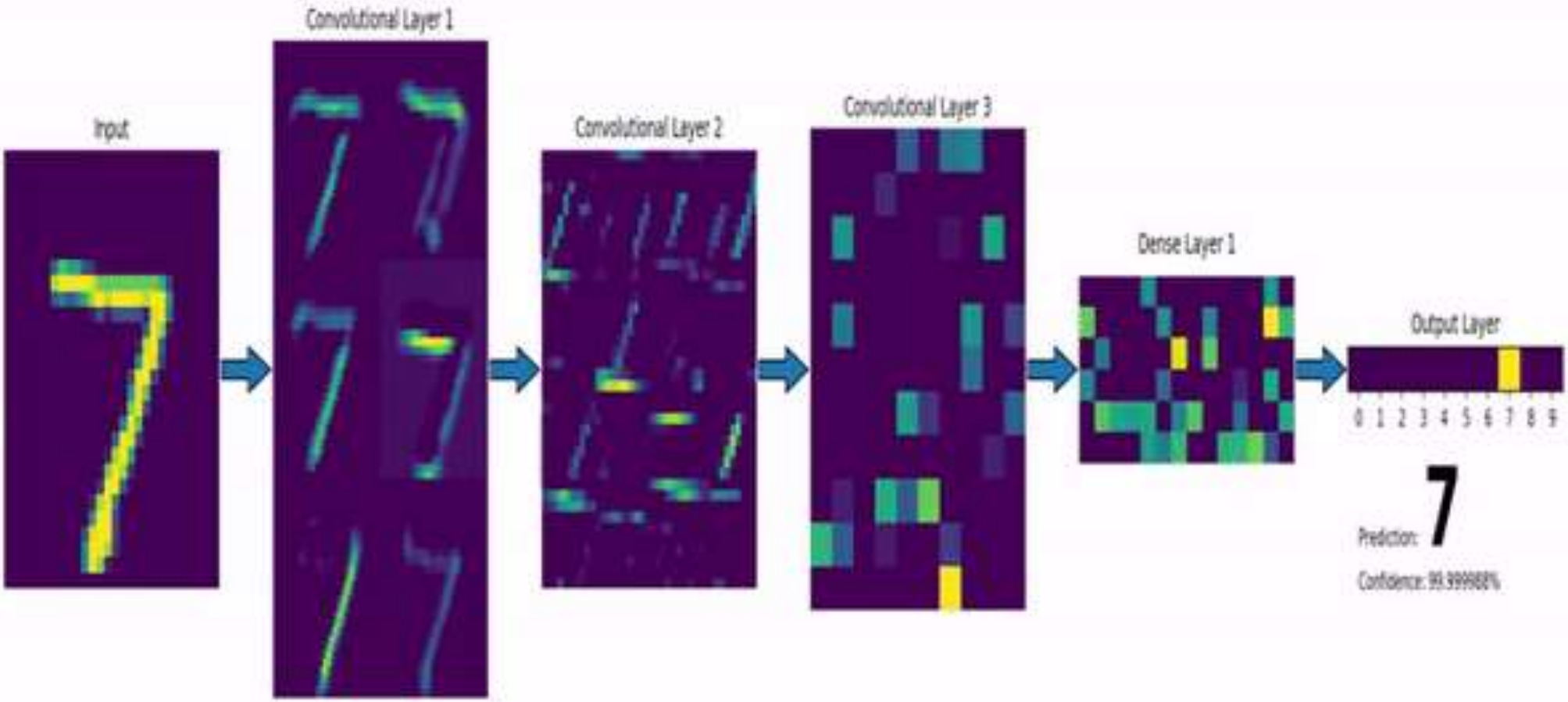
- **Motivation: We care about presence of features, not their exact location !**
- Dimensionality Reduction
- Prevents overfitting



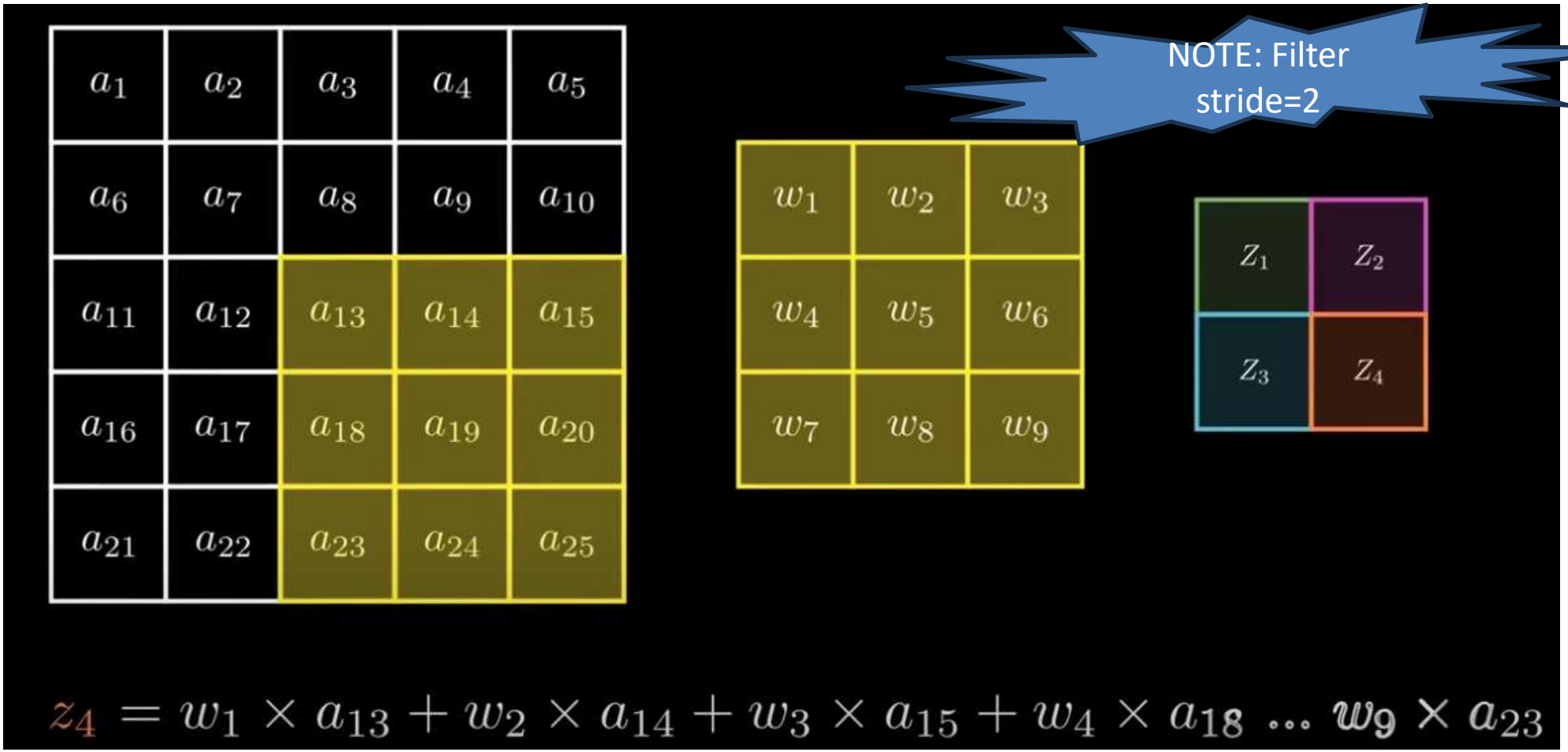
Convolutional Neural Network



Convolution (in general)

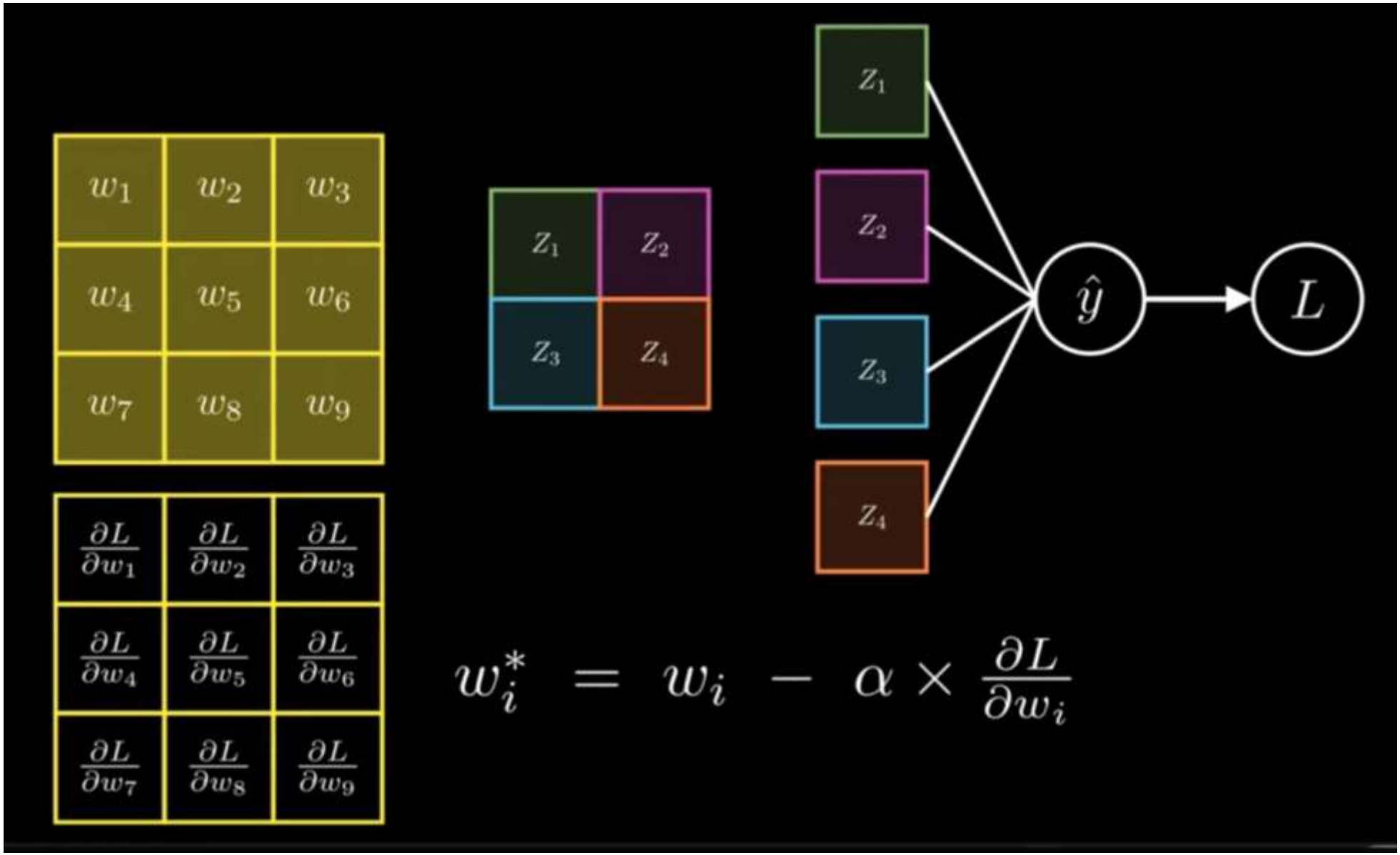


Backpropagation in CNNs

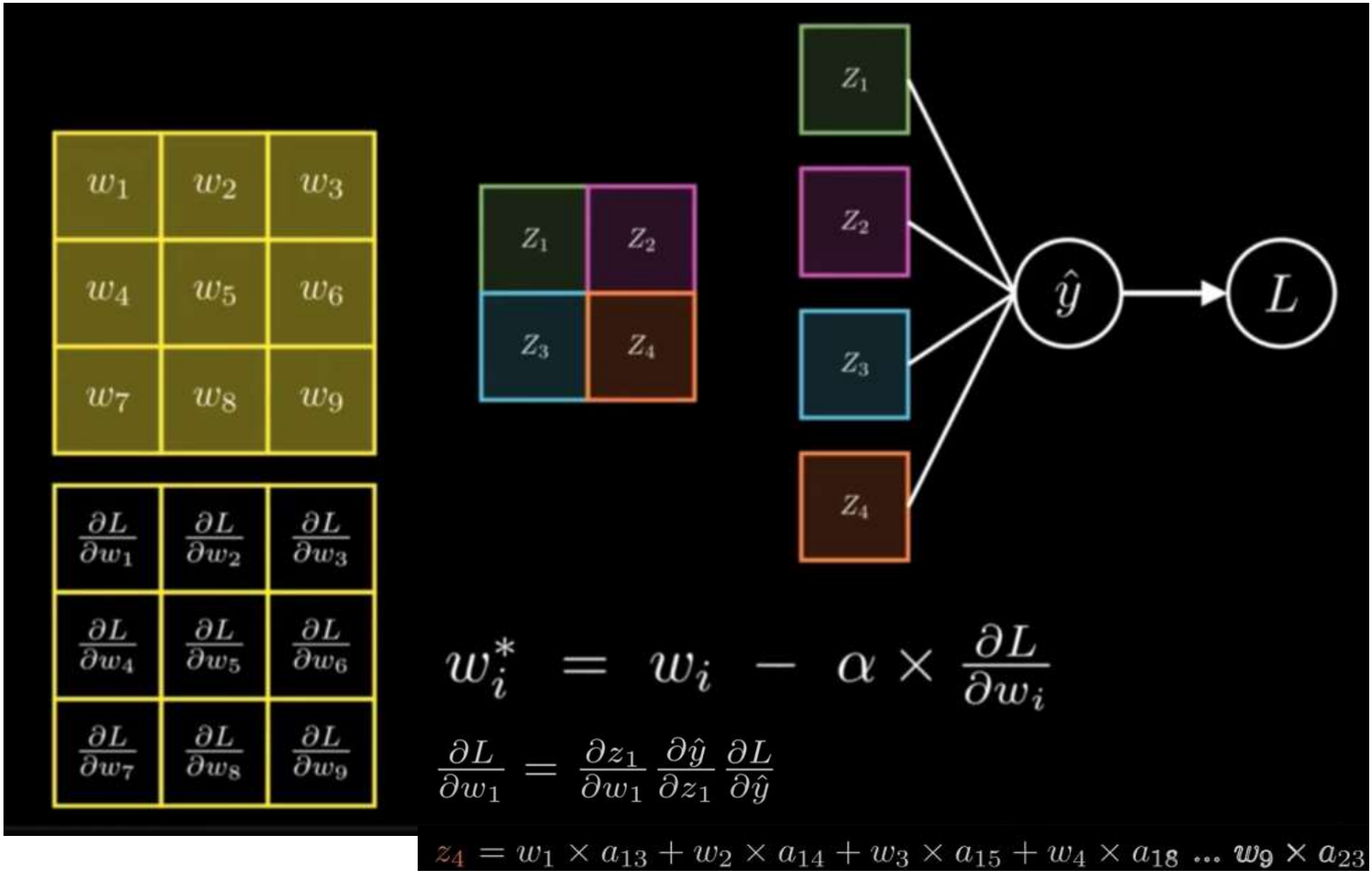


A * W = Z

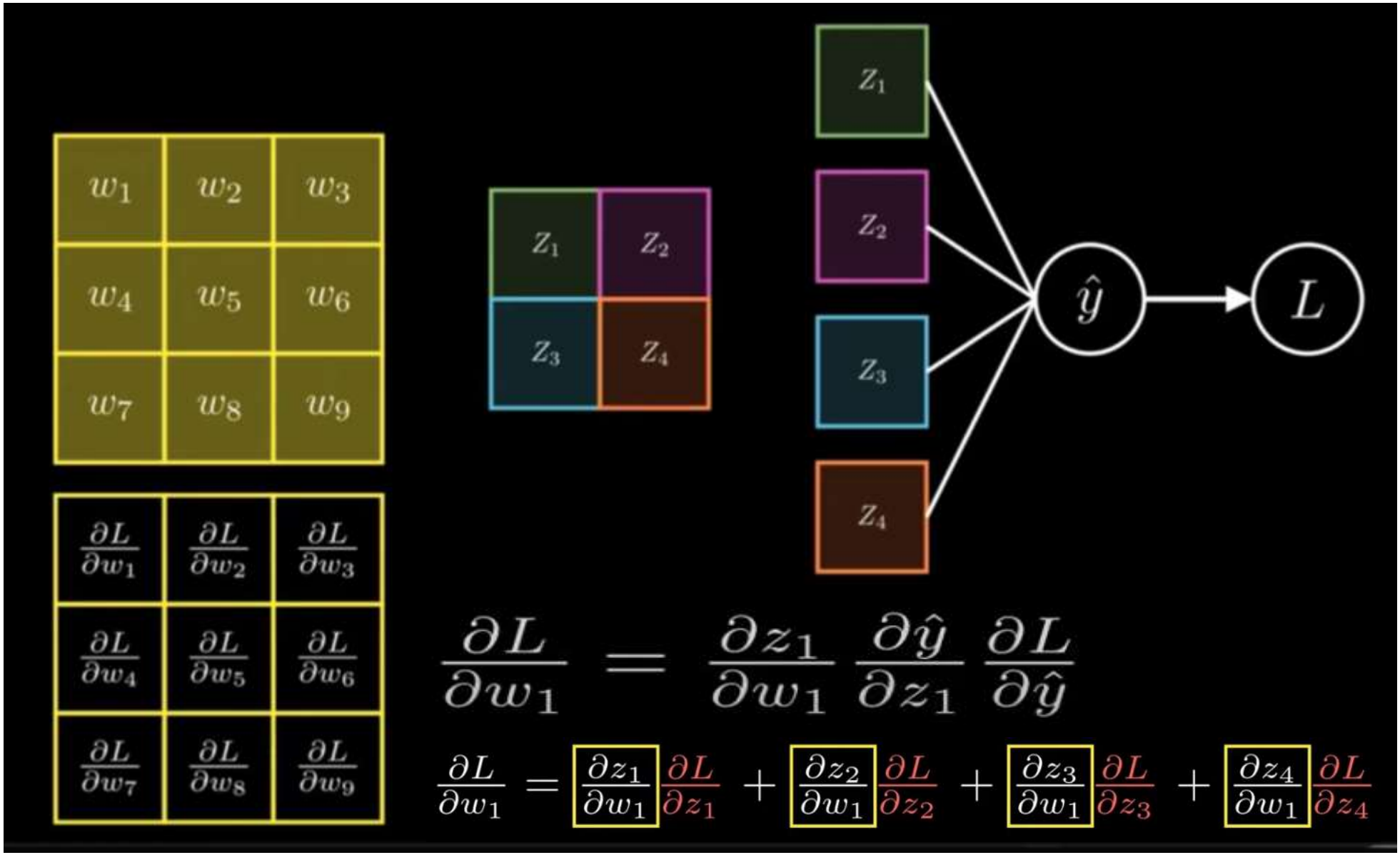
Backpropagation in CNNs



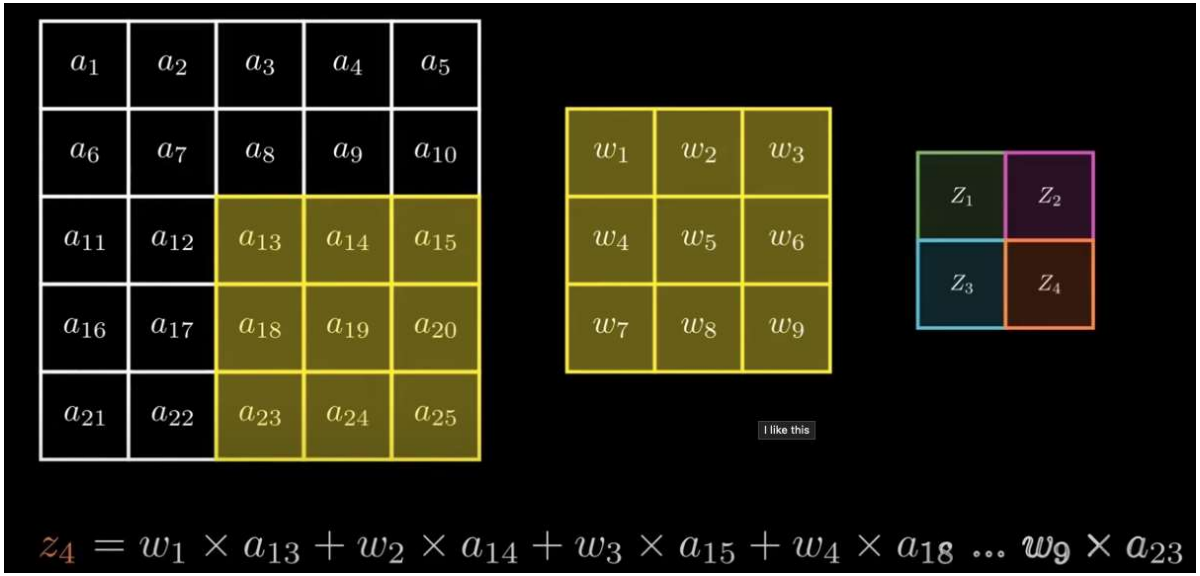
Backpropagation in CNNs



Backpropagation in CNNs



Backpropagation in CNNs



$$\frac{\partial L}{\partial w_1} = \boxed{\frac{\partial z_1}{\partial w_1}}\frac{\partial L}{\partial z_1} + \boxed{\frac{\partial z_2}{\partial w_1}}\frac{\partial L}{\partial z_2} + \boxed{\frac{\partial z_3}{\partial w_1}}\frac{\partial L}{\partial z_3} + \boxed{\frac{\partial z_4}{\partial w_1}}\frac{\partial L}{\partial z_4}$$

$$\frac{\partial L}{\partial w_2} = a_2\frac{\partial L}{\partial z_1} + a_4\frac{\partial L}{\partial z_2} + a_{12}\frac{\partial L}{\partial z_3} + a_{14}\frac{\partial L}{\partial z_4}$$

$$z_1 = w_1 \times a_1 + w_2 \times a_2 + w_3 \times a_3 + w_4 \times a_6 \ \ldots \ w_9 \times a_{13}$$

$$z_2 = w_1 \times a_3 + w_2 \times a_4 + w_3 \times a_5 + w_4 \times a_8 \ \ldots \ w_9 \times a_{15}$$

$$z_3 = w_1 \times a_{11} + w_2 \times a_{12} + w_3 \times a_{13} + w_4 \times a_{16} \ \ldots \ w_9 \times a_{23}$$

$$z_4 = w_1 \times a_{13} + w_2 \times a_{14} + w_3 \times a_{15} + w_4 \times a_{18} \ \ldots \ w_9 \times a_{25}$$

Backpropagation in CNNs

$$\frac{\partial L}{\partial w_1} = \boxed{\frac{\partial z_1}{\partial w_1}}\frac{\partial L}{\partial z_1} + \boxed{\frac{\partial z_2}{\partial w_1}}\frac{\partial L}{\partial z_2} + \boxed{\frac{\partial z_3}{\partial w_1}}\frac{\partial L}{\partial z_3} + \boxed{\frac{\partial z_4}{\partial w_1}}\frac{\partial L}{\partial z_4}$$

$$\frac{\partial L}{\partial w_2} = a_2\frac{\partial L}{\partial z_1} + a_4\frac{\partial L}{\partial z_2} + a_{12}\frac{\partial L}{\partial z_3} + a_{14}\frac{\partial L}{\partial z_4}$$

$$z_1 = w_1 \times a_1 + w_2 \times a_2 + w_3 \times a_3 + w_4 \times a_6 \ \ldots \ w_9 \times a_{13}$$

$$z_2 = w_1 \times a_3 + w_2 \times a_4 + w_3 \times a_5 + w_4 \times a_8 \ \ldots \ w_9 \times a_{15}$$

$$z_3 = w_1 \times a_{11} + w_2 \times a_{12} + w_3 \times a_{13} + w_4 \times a_{16} \ \ldots \ w_9 \times a_{23}$$

$$z_4 = w_1 \times a_{13} + w_2 \times a_{14} + w_3 \times a_{15} + w_4 \times a_{18} \ \ldots \ w_9 \times a_{25}$$

a_1	a_2	a_3	a_4	a_5
a_6	a_7	a_8	a_9	a_{10}
a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{16}	a_{17}	a_{18}	a_{19}	a_{20}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}

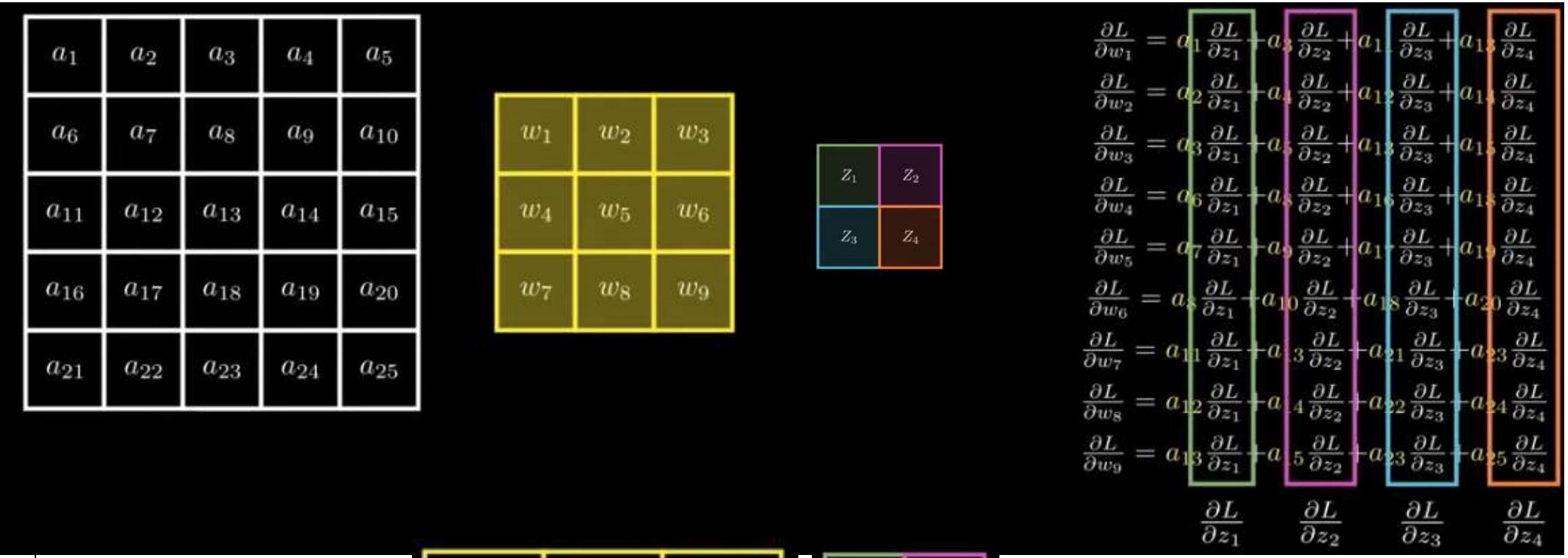
w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

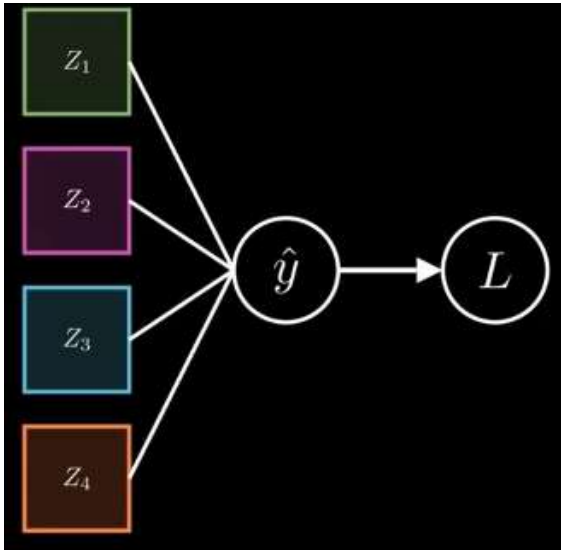
z_1	z_2
z_3	z_4

$$\frac{\partial L}{\partial w_1} = a_1\frac{\partial L}{\partial z_1} + a_3\frac{\partial L}{\partial z_2} + a_1\frac{\partial L}{\partial z_3} + a_{13}\frac{\partial L}{\partial z_4}$$
$$\frac{\partial L}{\partial w_2} = a_2\frac{\partial L}{\partial z_1} + a_4\frac{\partial L}{\partial z_2} + a_{12}\frac{\partial L}{\partial z_3} + a_{14}\frac{\partial L}{\partial z_4}$$
$$\frac{\partial L}{\partial w_3} = a_3\frac{\partial L}{\partial z_1} + a_5\frac{\partial L}{\partial z_2} + a_{13}\frac{\partial L}{\partial z_3} + a_{15}\frac{\partial L}{\partial z_4}$$
$$\frac{\partial L}{\partial w_4} = a_6\frac{\partial L}{\partial z_1} + a_8\frac{\partial L}{\partial z_2} + a_{16}\frac{\partial L}{\partial z_3} + a_{18}\frac{\partial L}{\partial z_4}$$
$$\frac{\partial L}{\partial w_5} = a_7\frac{\partial L}{\partial z_1} + a_9\frac{\partial L}{\partial z_2} + a_{17}\frac{\partial L}{\partial z_3} + a_{19}\frac{\partial L}{\partial z_4}$$
$$\frac{\partial L}{\partial w_6} = a_8\frac{\partial L}{\partial z_1} + a_{10}\frac{\partial L}{\partial z_2} + a_{18}\frac{\partial L}{\partial z_3} + a_{20}\frac{\partial L}{\partial z_4}$$
$$\frac{\partial L}{\partial w_7} = a_{11}\frac{\partial L}{\partial z_1} + a_{13}\frac{\partial L}{\partial z_2} + a_{21}\frac{\partial L}{\partial z_3} + a_{23}\frac{\partial L}{\partial z_4}$$
$$\frac{\partial L}{\partial w_8} = a_{12}\frac{\partial L}{\partial z_1} + a_{14}\frac{\partial L}{\partial z_2} + a_{22}\frac{\partial L}{\partial z_3} + a_{24}\frac{\partial L}{\partial z_4}$$
$$\frac{\partial L}{\partial w_9} = a_{13}\frac{\partial L}{\partial z_1} + a_{15}\frac{\partial L}{\partial z_2} + a_{23}\frac{\partial L}{\partial z_3} + a_{25}\frac{\partial L}{\partial z_4}$$

$\frac{\partial L}{\partial z_1}$ $\frac{\partial L}{\partial z_2}$ $\frac{\partial L}{\partial z_3}$ $\frac{\partial L}{\partial z_4}$

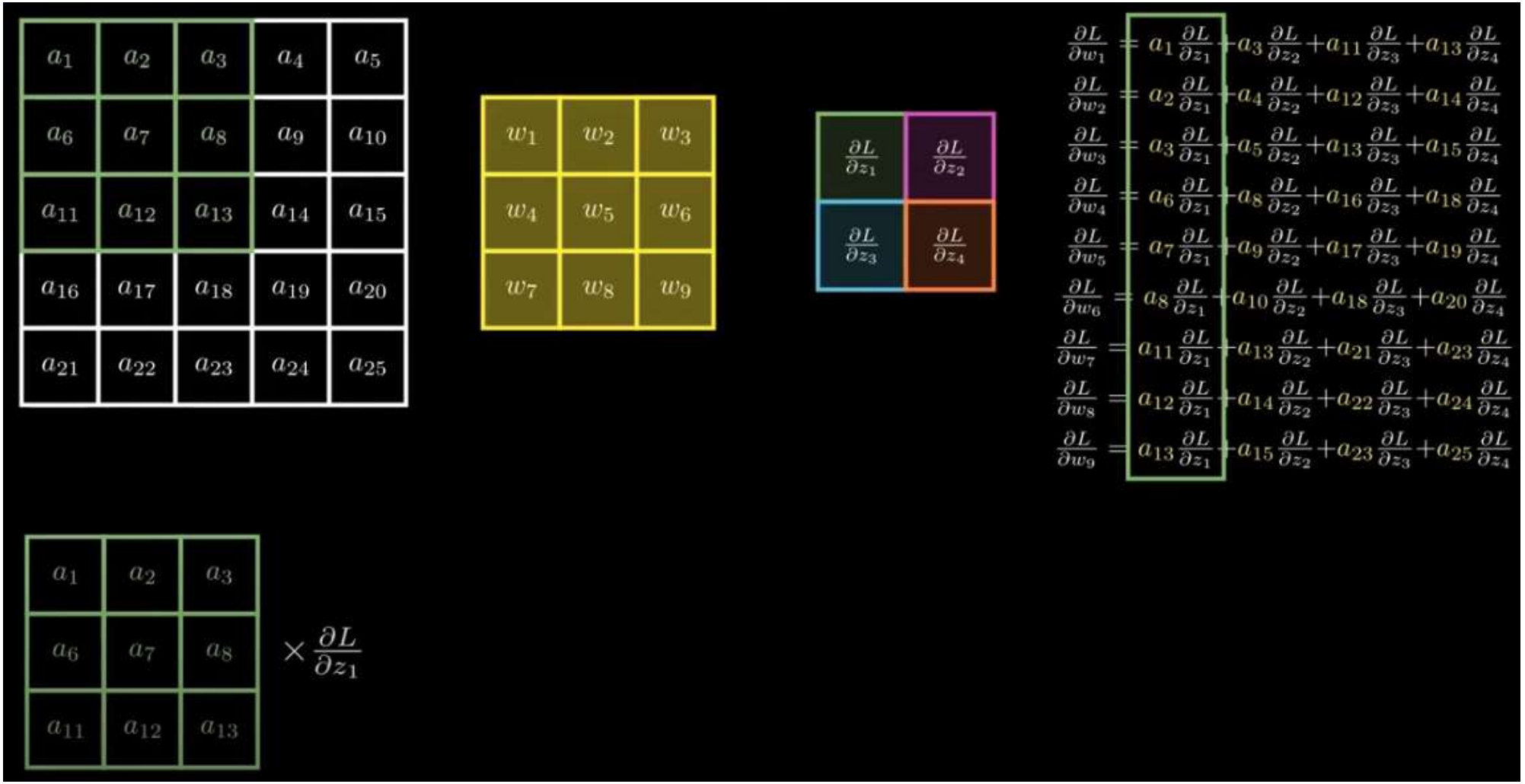
Backpropagation in CNNs



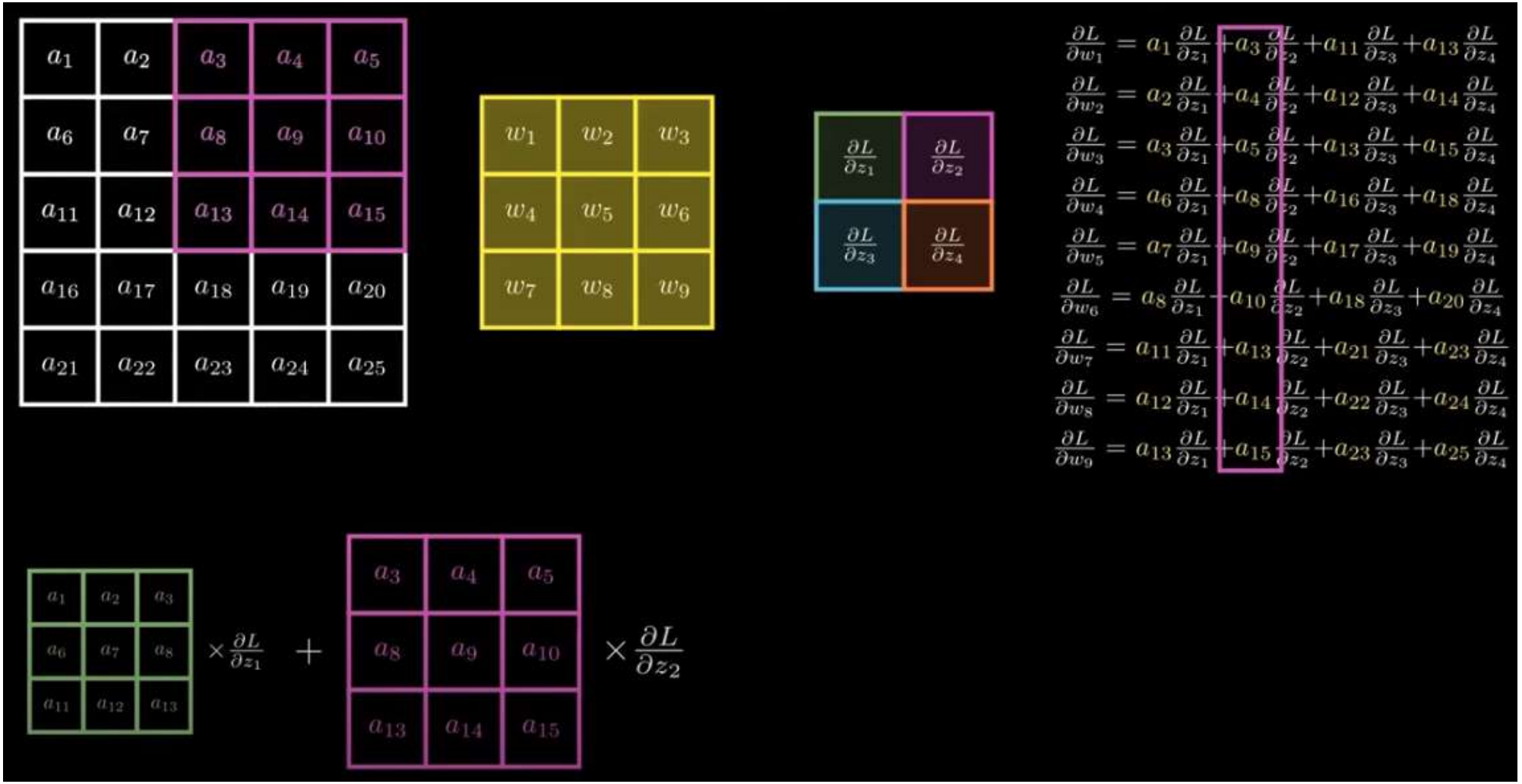


$$\frac{\partial L}{\partial w_1} = \frac{\partial z_1}{\partial w_1} \frac{\partial \hat{y}}{\partial z_1} \frac{\partial L}{\partial \hat{y}}$$

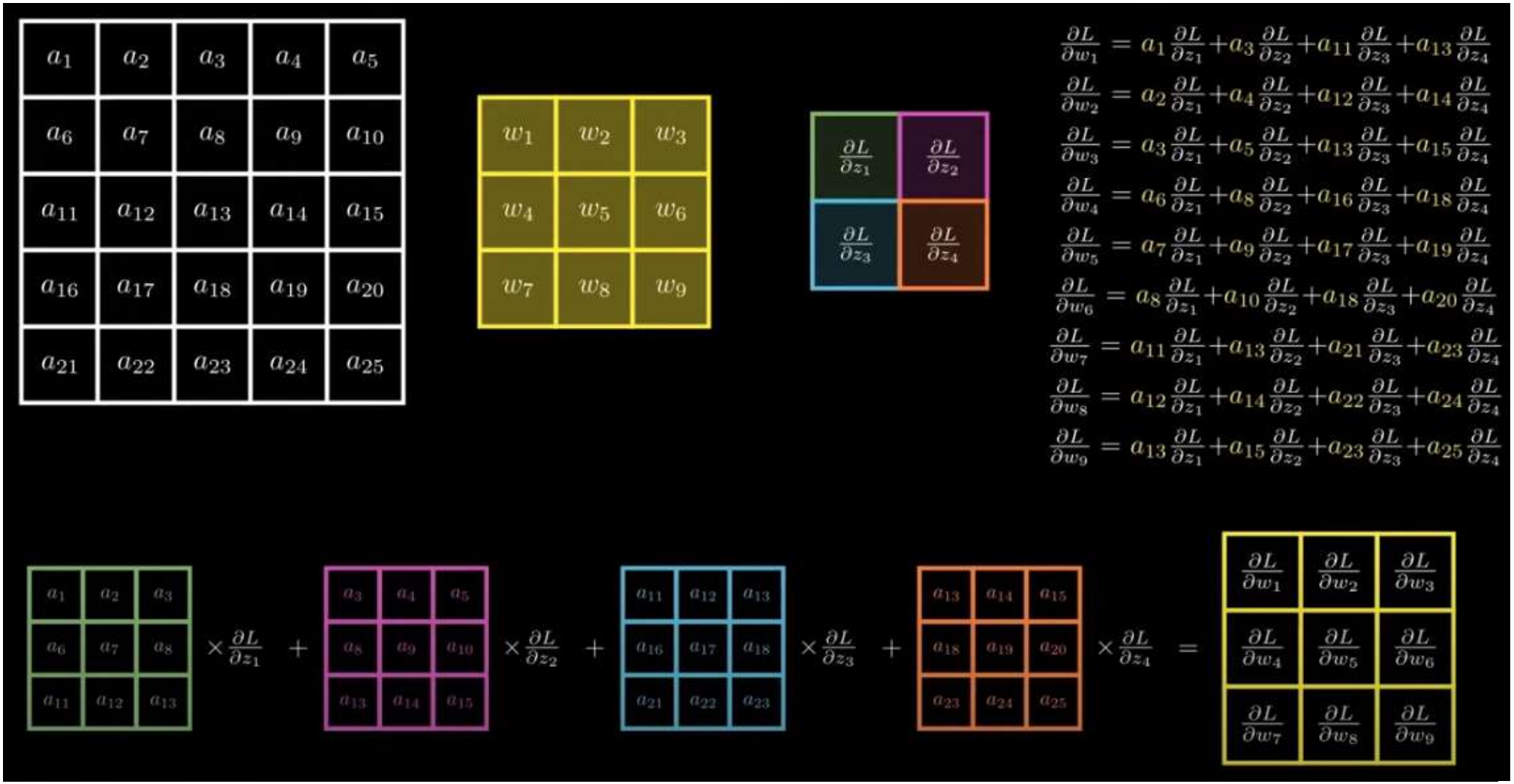
Backpropagation in CNNs

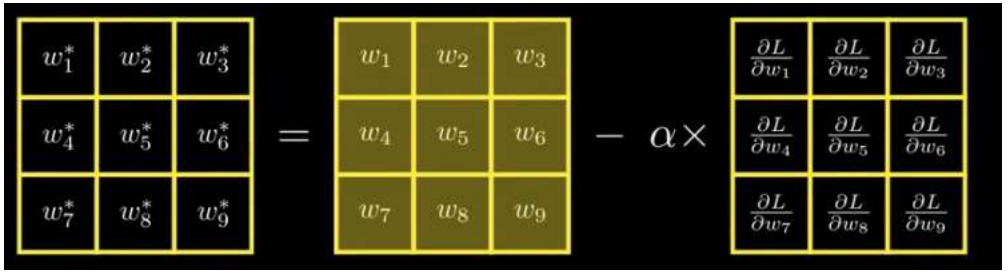


Backpropagation in CNNs

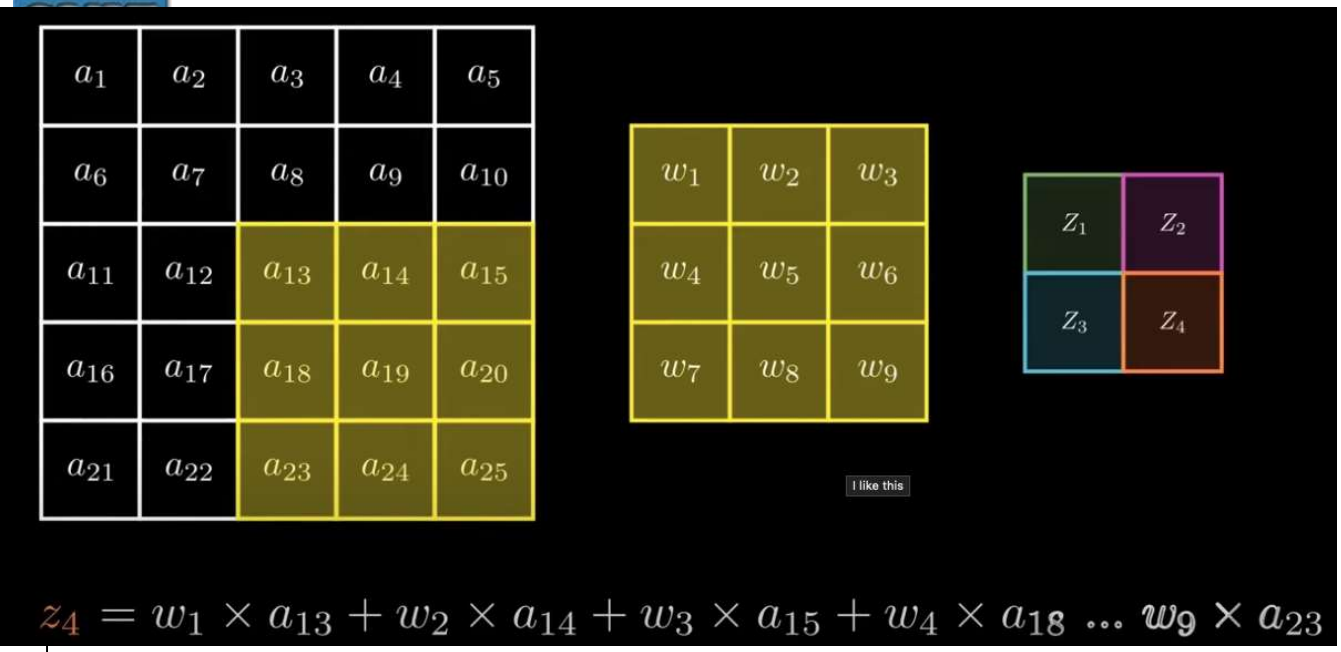


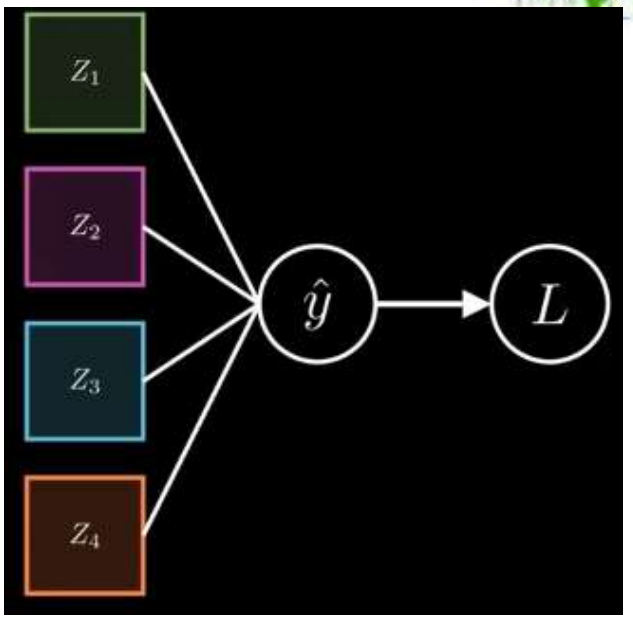
Backpropagation in CNNs

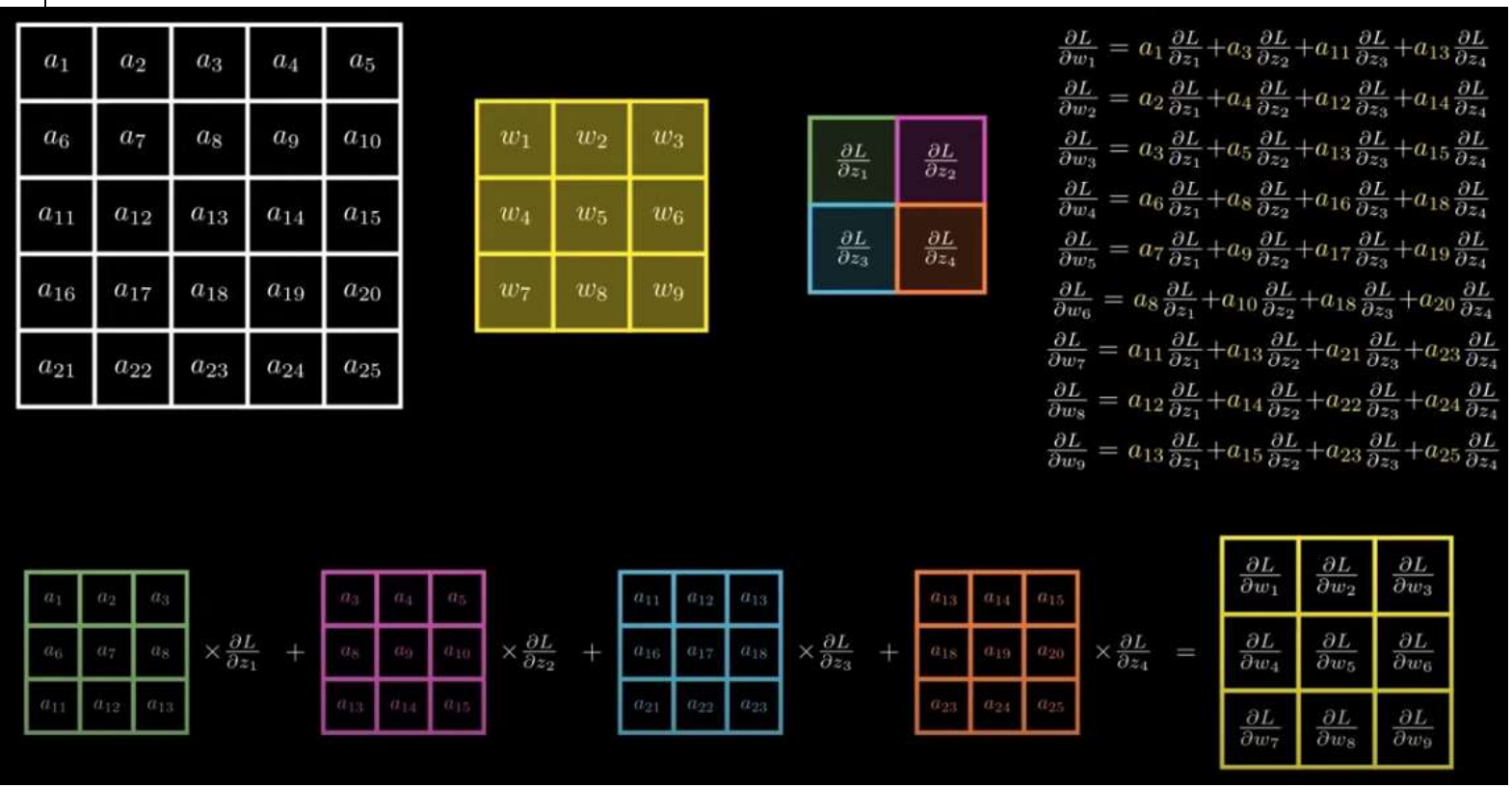




CNN – forward & backpropagation



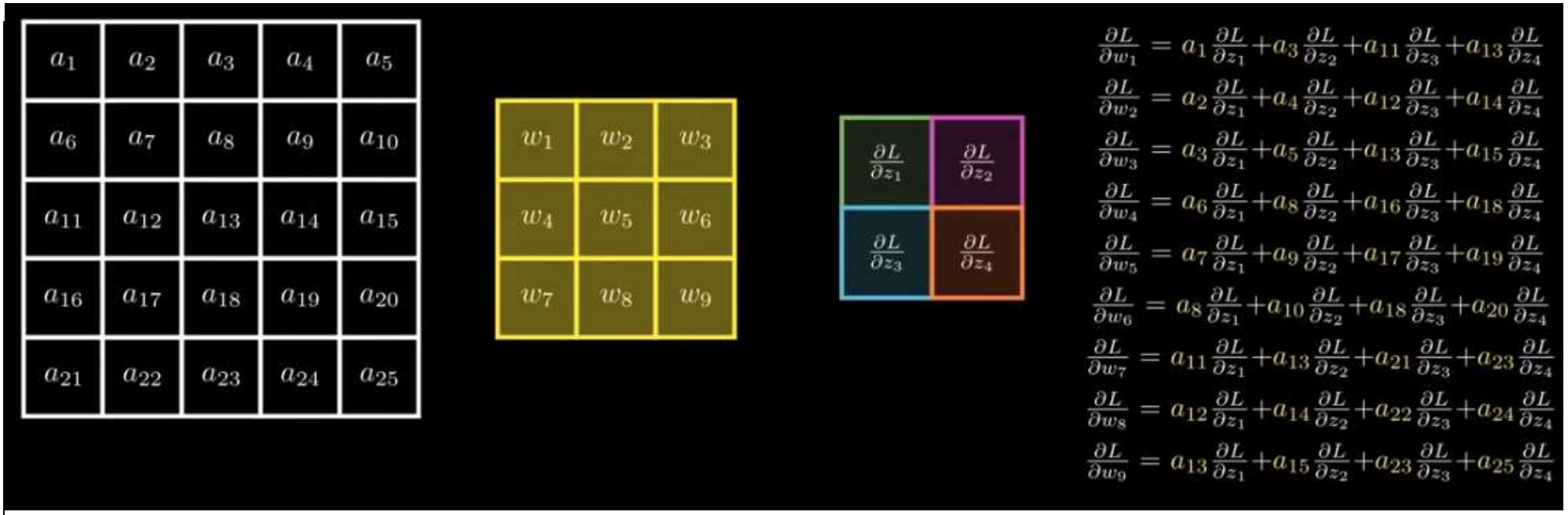




$$\frac{\partial L}{\partial w_1} = \frac{\partial z_1}{\partial w_1} \frac{\partial \hat{y}}{\partial z_1} \frac{\partial L}{\partial \hat{y}}$$

Convolution

CNN – forward & backpropagation



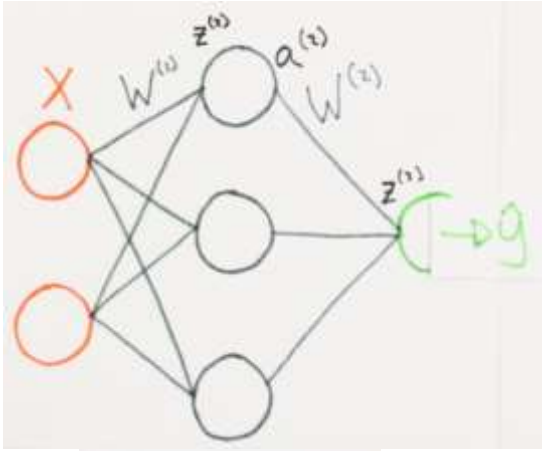
$$\frac{\partial L}{\partial w_1} = \frac{\partial z_1}{\partial w_1}\frac{\partial L}{\partial z_1} + \frac{\partial z_2}{\partial w_1}\frac{\partial L}{\partial z_2} + \frac{\partial z_3}{\partial w_1}\frac{\partial L}{\partial z_3} + \frac{\partial z_4}{\partial w_1}\frac{\partial L}{\partial z_4}$$

$$z_1 = w_1 \times a_1 + w_2 \times a_2 + w_3 \times a_3 + w_4 \times a_6 \ \ldots \ w_9 \times a_{13}$$

$$z_2 = w_1 \times a_3 + w_2 \times a_4 + w_3 \times a_5 + w_4 \times a_8 \ \ldots \ w_9 \times a_{15}$$

$$z_3 = w_1 \times a_{11} + w_2 \times a_{12} + w_3 \times a_{13} + w_4 \times a_{16} \ \ldots \ w_9 \times a_{23}$$

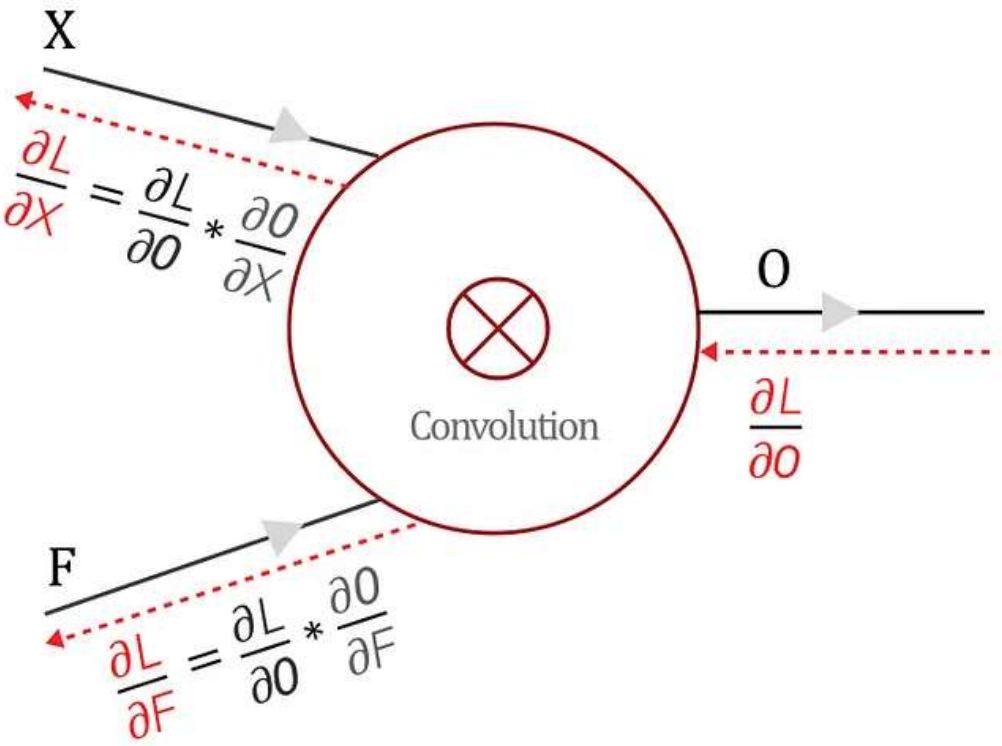
$$z_4 = w_1 \times a_{13} + w_2 \times a_{14} + w_3 \times a_{15} + w_4 \times a_{18} \ \ldots \ w_9 \times a_{25}$$



$$z^{(2)} = XW^{(1)} \quad (1)$$
$$a^{(2)} = f(z^{(2)}) \quad (2)$$
$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$
$$\hat{y} = f(z^{(3)}) \quad (4)$$

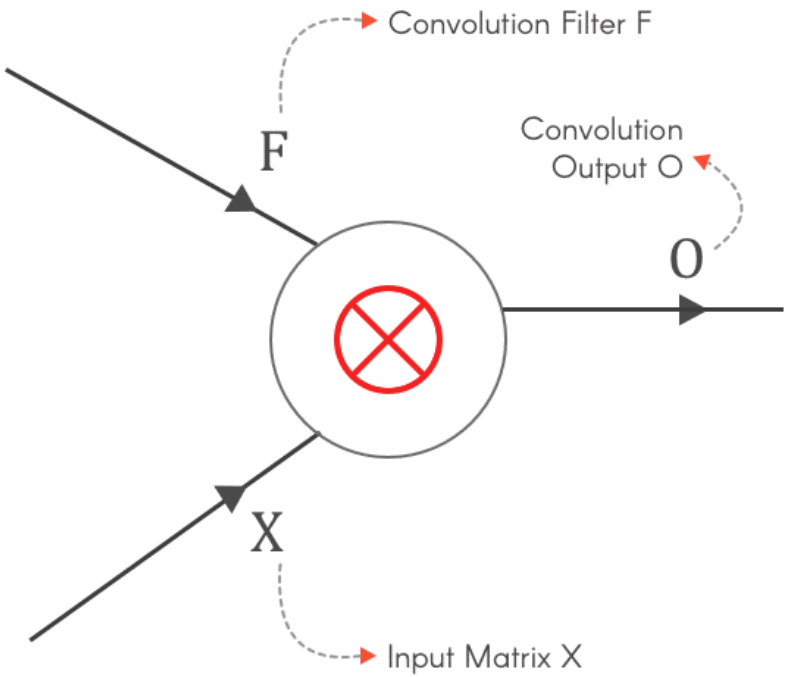
$$\frac{\partial J}{\partial W^{(2)}} = \underbrace{\frac{\partial J}{\partial z^{(3)}}}_{\text{loss from previous layer}} \times \underbrace{\frac{\partial z^{(3)}}{\partial W^{(2)}}}_{\text{loss from local gradient}}$$

$$\frac{\partial J}{\partial W^{(1)}} = \underbrace{\frac{\partial J}{\partial z^{(2)}}}_{\text{loss from previous layer}} \times \underbrace{\frac{\partial z^{(2)}}{\partial W^{(1)}}}_{\text{loss from local gradient}}$$



$\frac{\partial O}{\partial X}$ & $\frac{\partial O}{\partial F}$ *are local gradients*

$\frac{\partial L}{\partial z}$ *is the loss from the previous layer which has to be backpropagated to other layers*



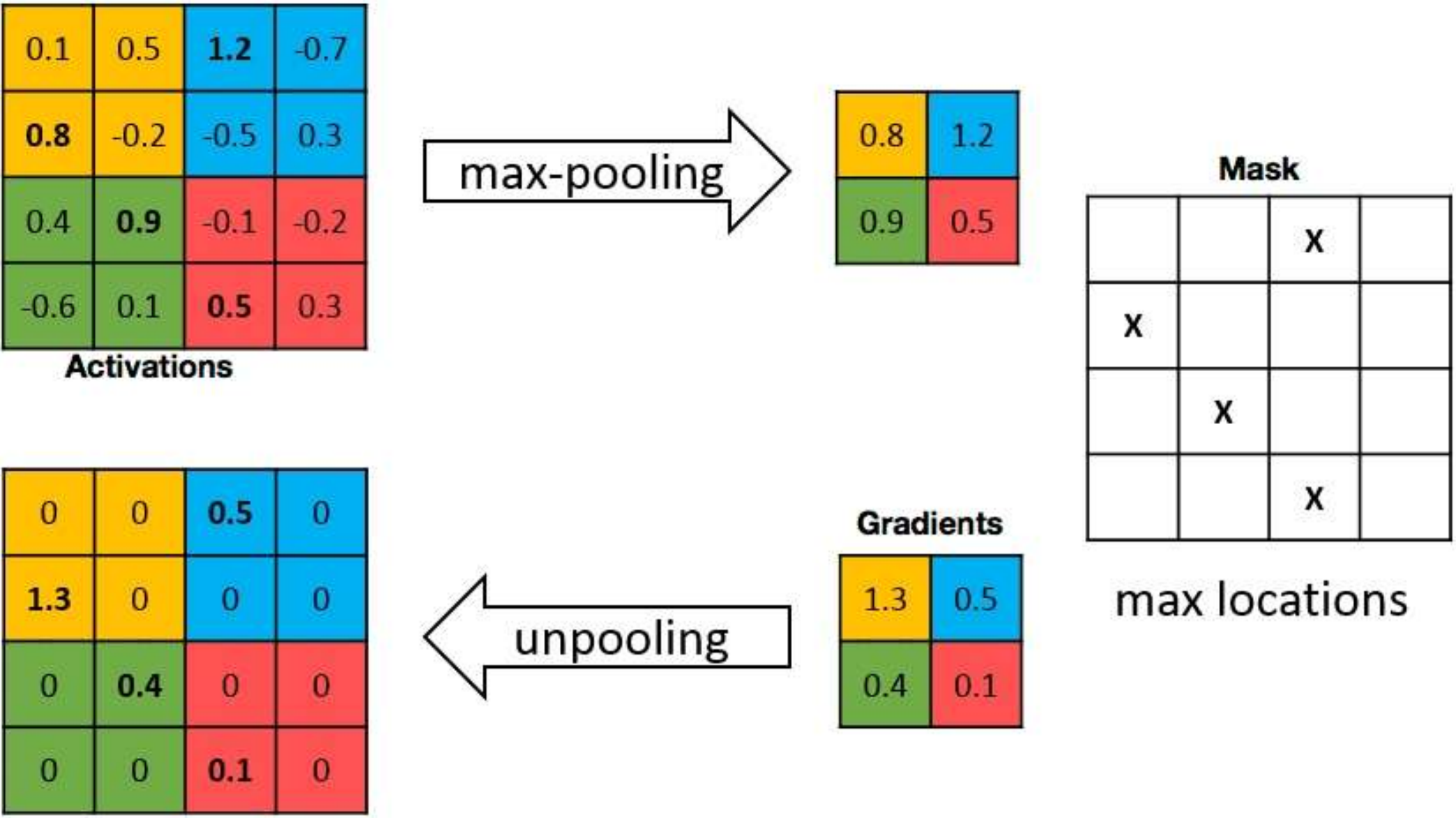
Backpropagation in a Convolutional Layer of a CNN

Finding the gradients:

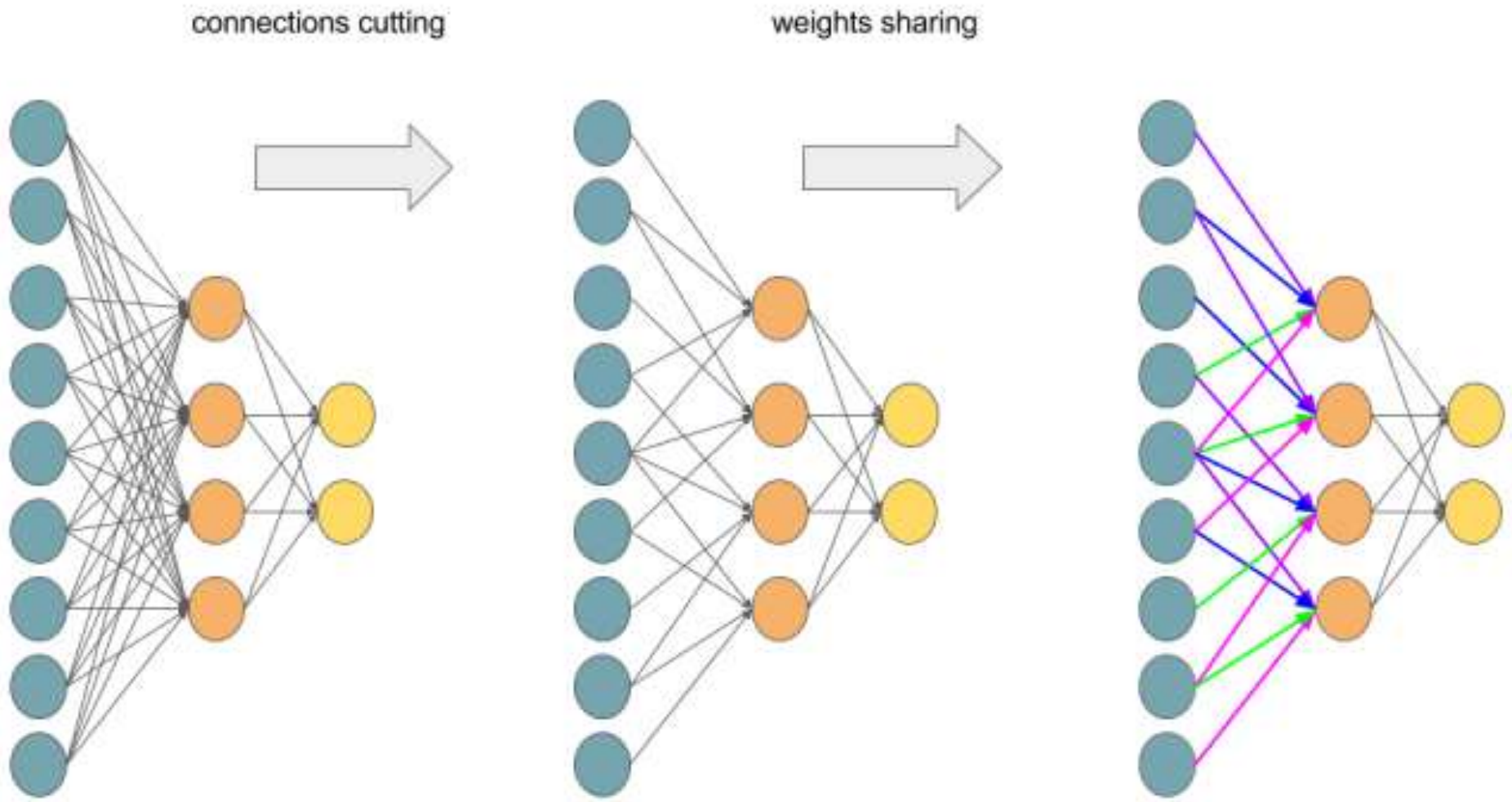
$$\frac{\partial L}{\partial F} = \text{Convolution}\left(\text{Input } X\text{, Loss gradient } \frac{\partial L}{\partial O}\right)$$

$$\frac{\partial L}{\partial X} = \text{Full Convolution}\left(180^{\circ}\text{rotated Filter } F\text{, Loss Gradient } \frac{\partial L}{\partial O}\right)$$

Backprop in pooling layer

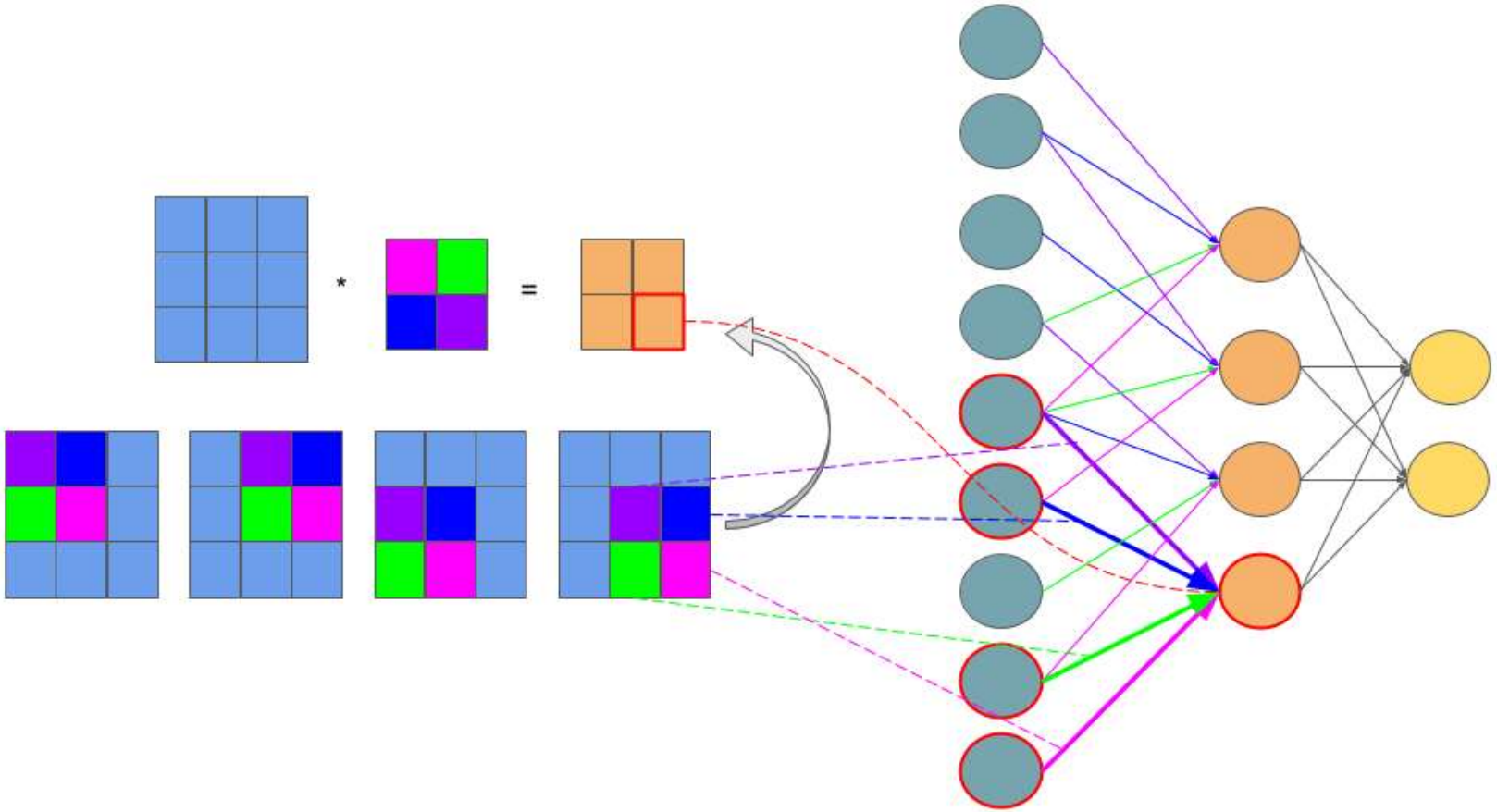


CNN = MLP with sparse connectivity and weight sharing

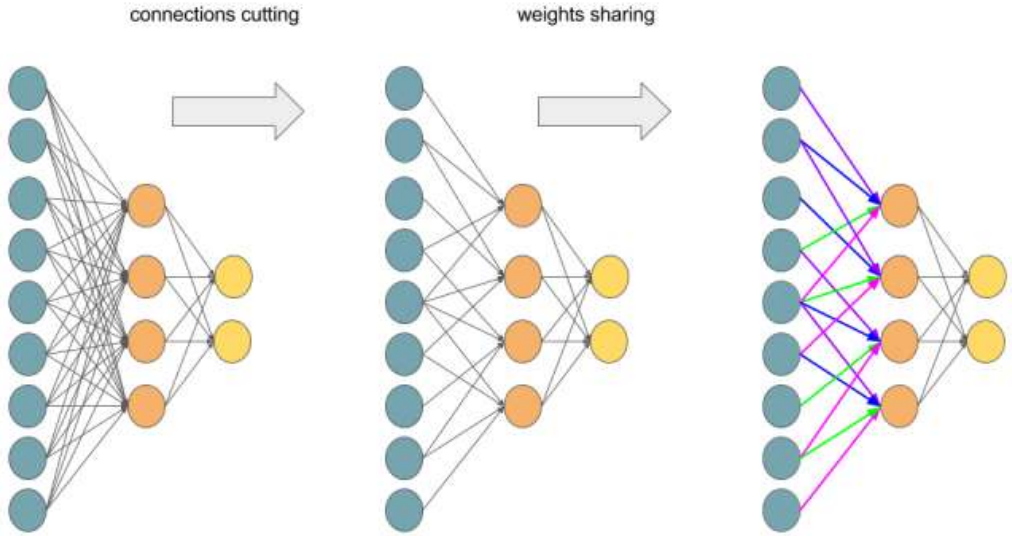


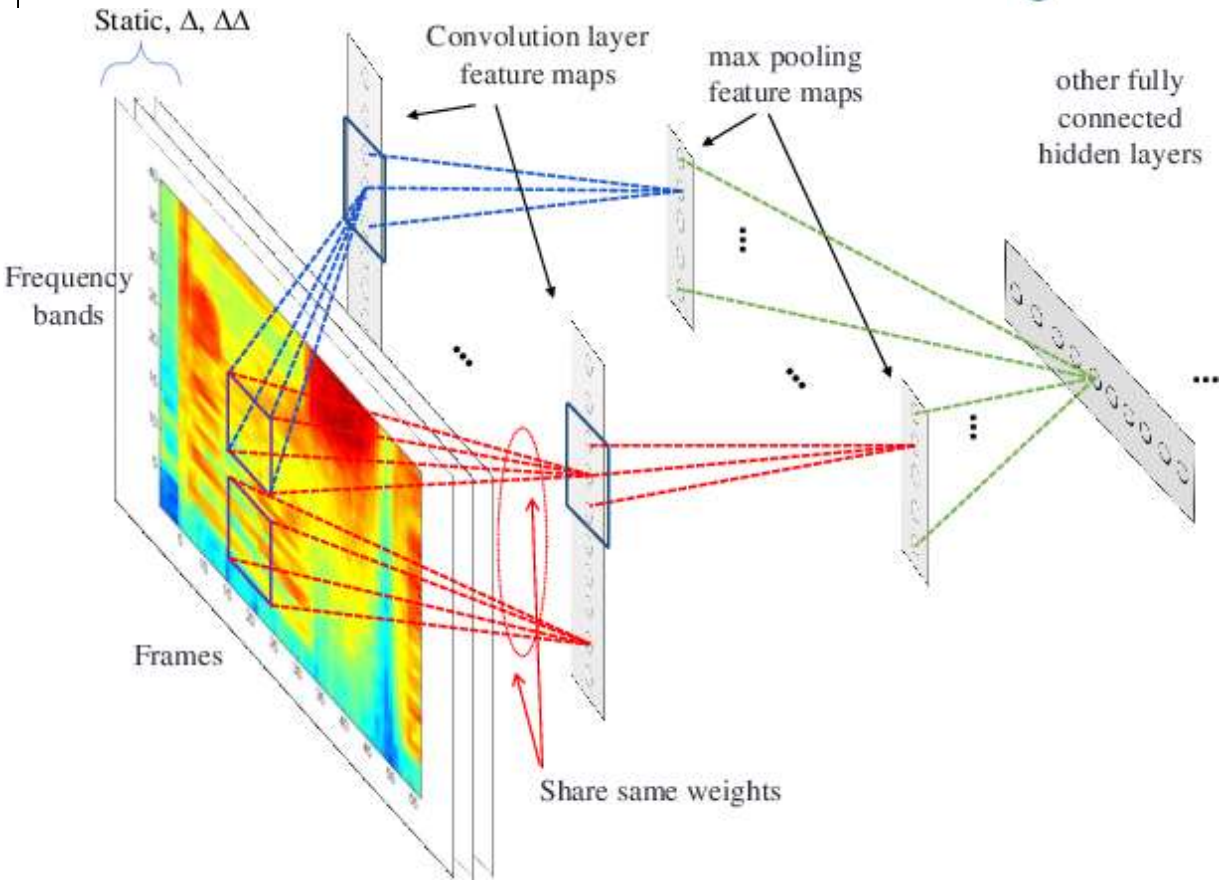
https://grzegorzgwardys.wordpress.com/2016/04/22/8/

CNN v/s MLP



https://grzegorzgwardys.wordpress.com/2016/04/22/8/


connections cutting
weights sharing

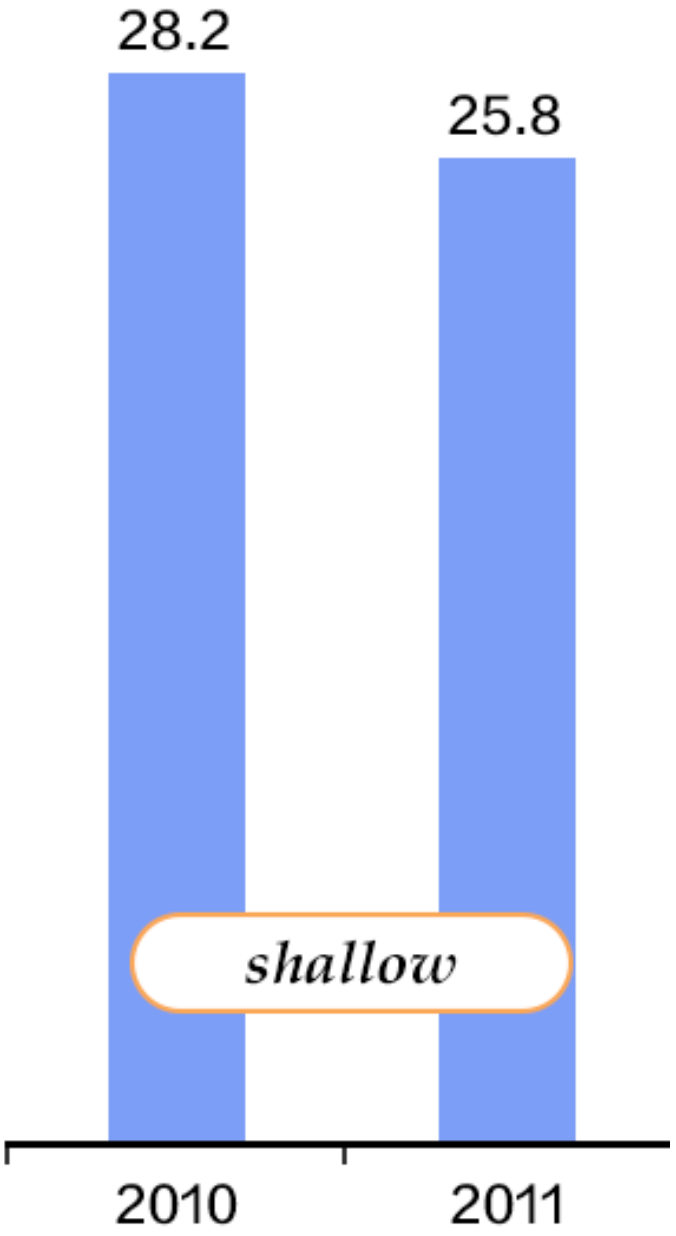

Static, Δ, ΔΔ
Convolution layer
feature maps
max pooling
feature maps
other fully
connected
hidden layers
Frequency
bands
Frames
Share same weights

ImageNet Challenge

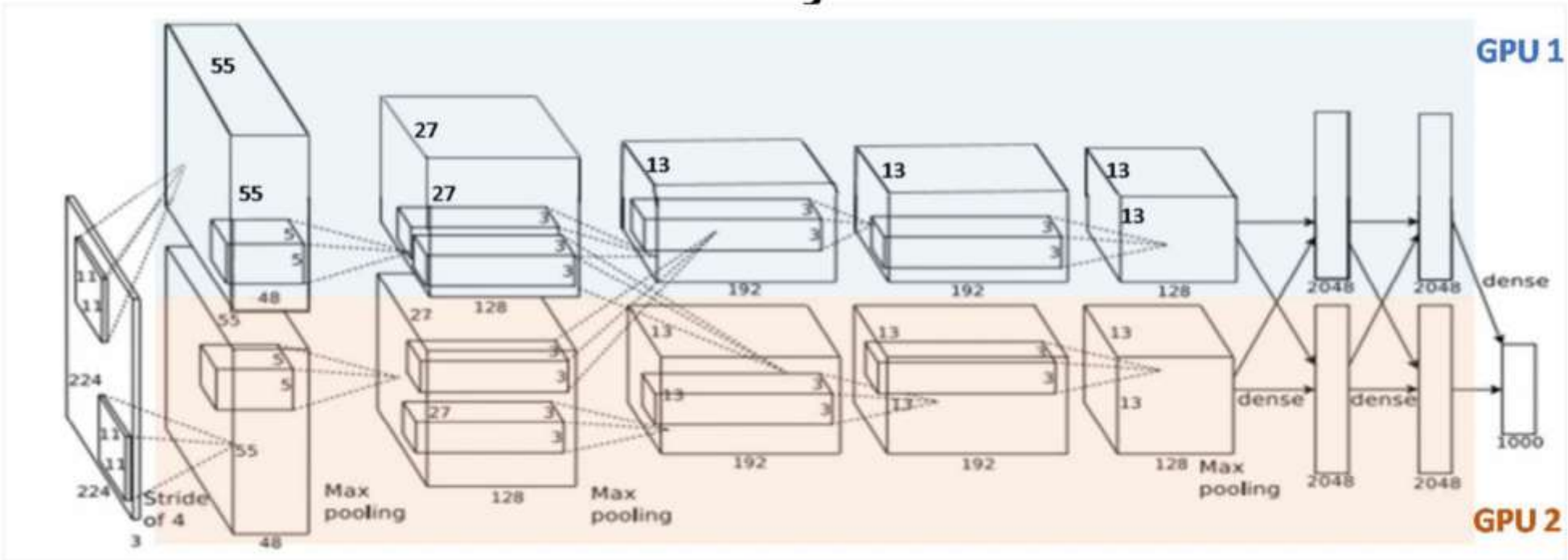
IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

Top-5 misclassification error rate



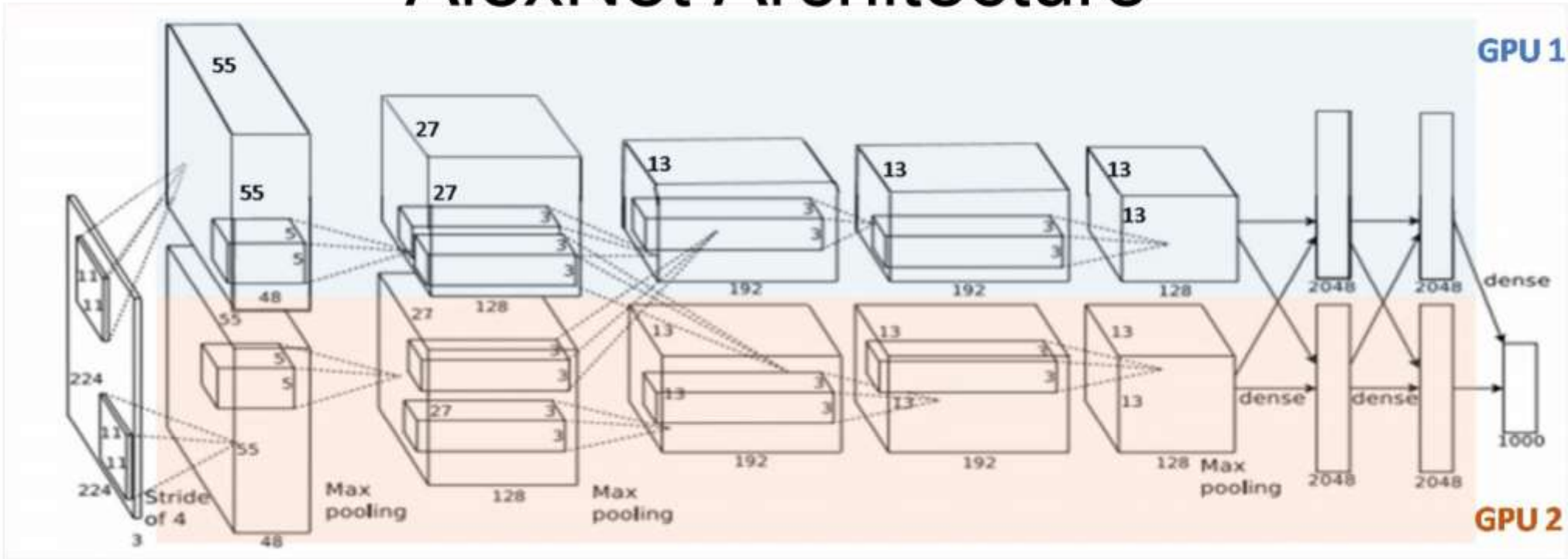
Case Study: AlexNet

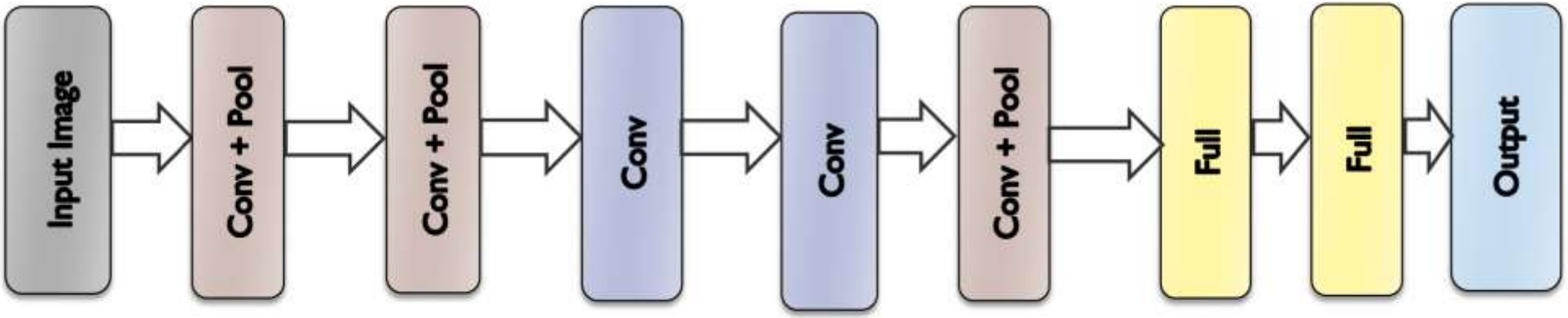


- Winner of ImageNet LSVRC-2010.
- Trained over 1.2M images using SGD with regularization.
- Deep architecture (60M parameters.)
- Optimized GPU implementation (cuda-convnet)

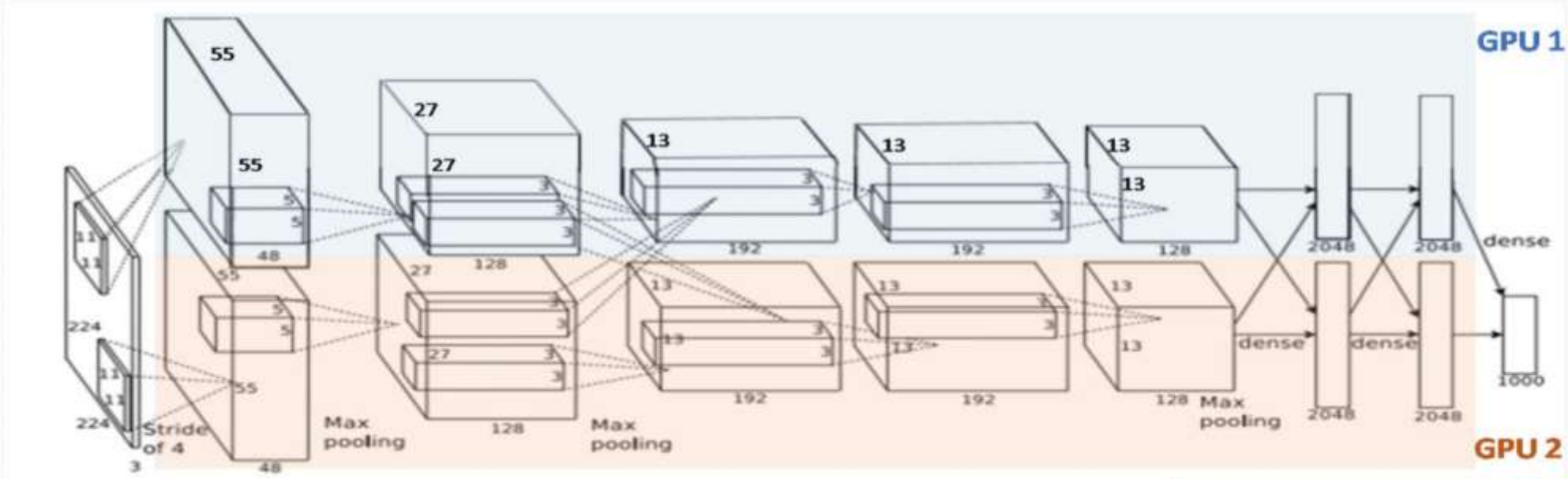
Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *NIPS* 2012.

AlexNet Architecture





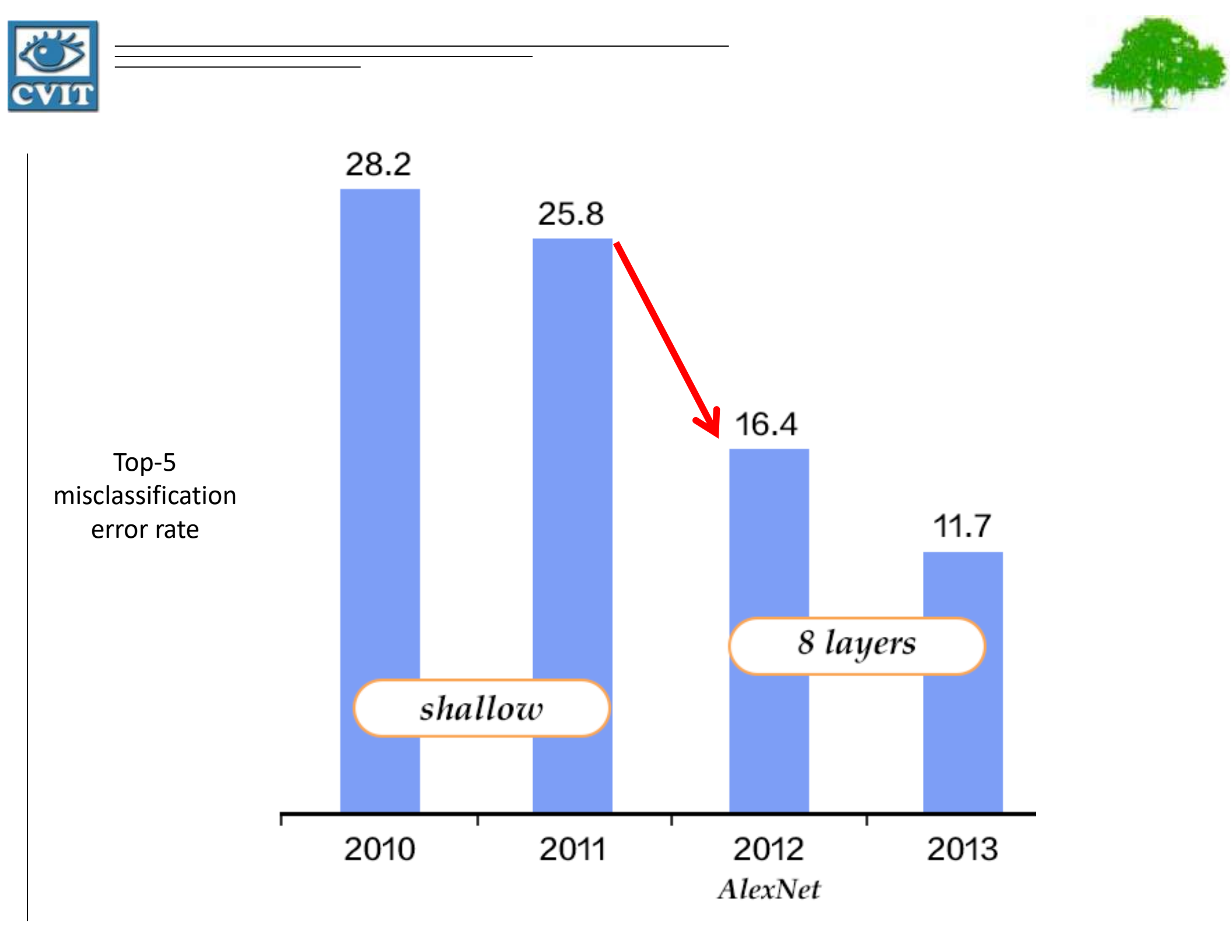
AlexNet Architecture



About 57 M parameters are in the fully connected layers

									Total
Parameters :	[(11 x 11 x 3) + 1] x 96 = 35 K	[5 x 5 x 48] x 256 = 307 K	[3 x 3 x 256] x 384 = 884 K	663 K	442 K	37 M	16 M	4 M	60 M
Neurons :	253,440	27 x 27 x 256 = 186,624	13 x 13 x 384 = 64,896	13 x 13 x 384 = 64,896	13 x 13 x 256 = 43,264	4096	4096	1000	0.63 M

- Convolutional layers cumulatively contain about 90-95% of computation, only about 5% of the parameters
- Fully-connected layers contain about 95% of parameters.


Top-5 misclassification error rate
28.2
25.8
16.4
11.7
shallow
8 layers
2010
2011
2012
2013
AlexNet

Top-5 misclassification error rate

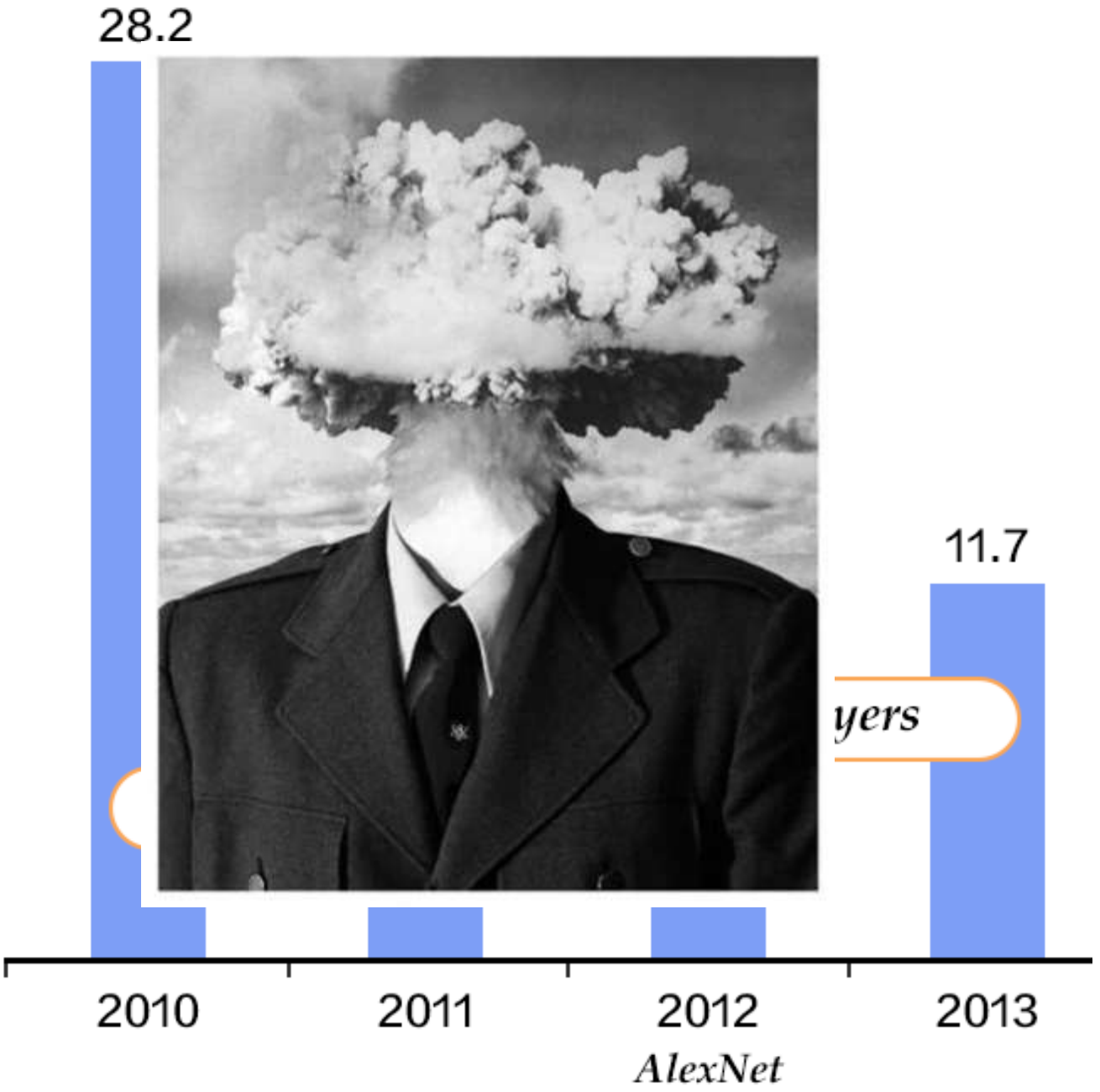
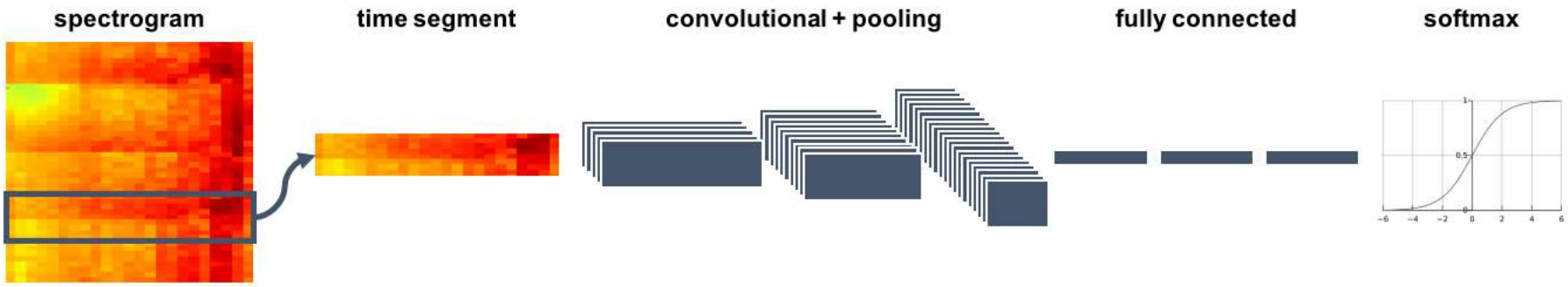


Image CNNs for non-image data

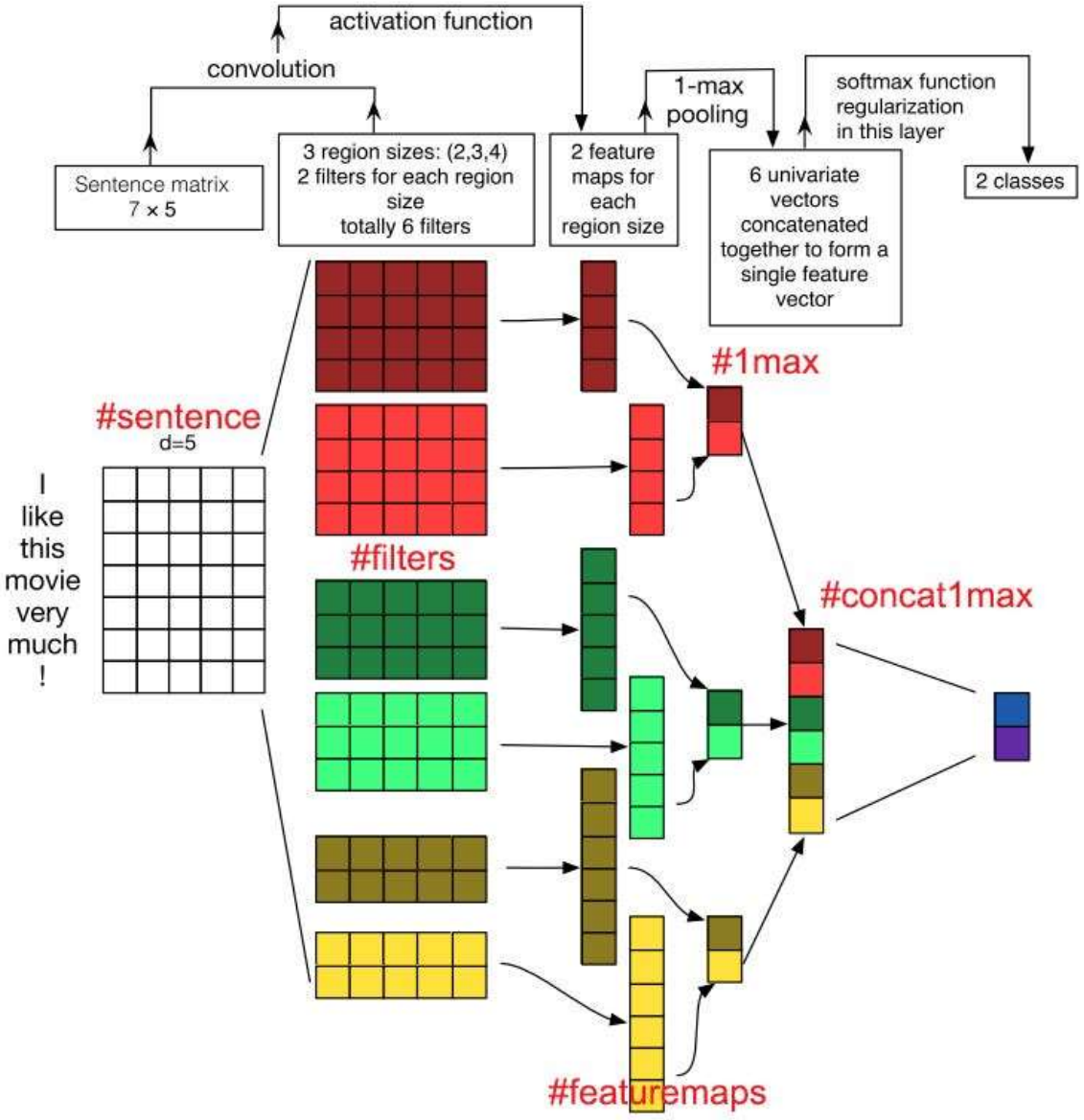
Audio Beat Detection



http://tommymullaney.com/projects/rhythm-games-neural-networks

Image CNNs for non-image data

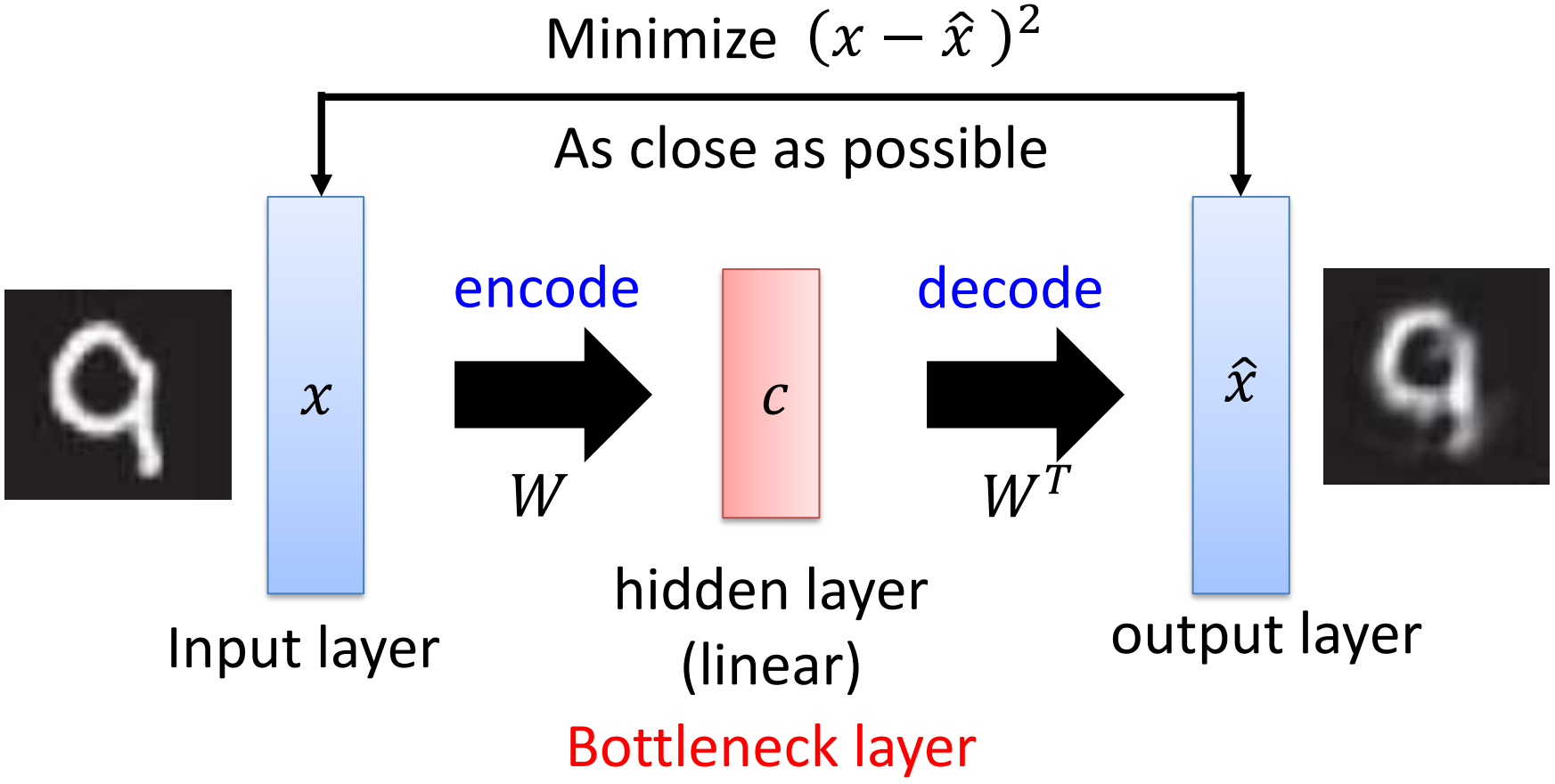
Sentiment Classification from Text



Unsupervised Learning:

Deep Auto-encoder

Recap: PCA



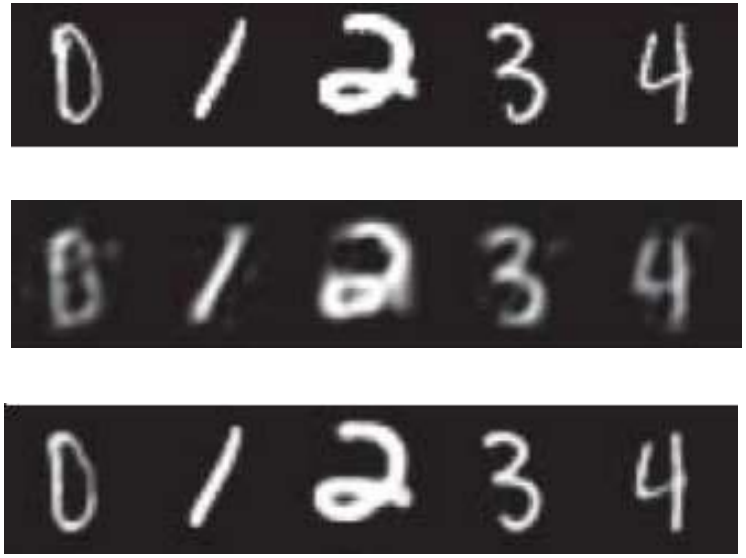
Output of the hidden layer is the code

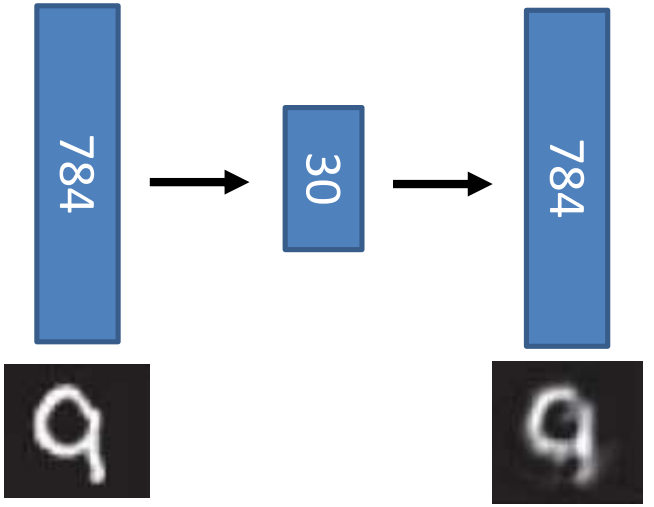
Deep Auto-encoder

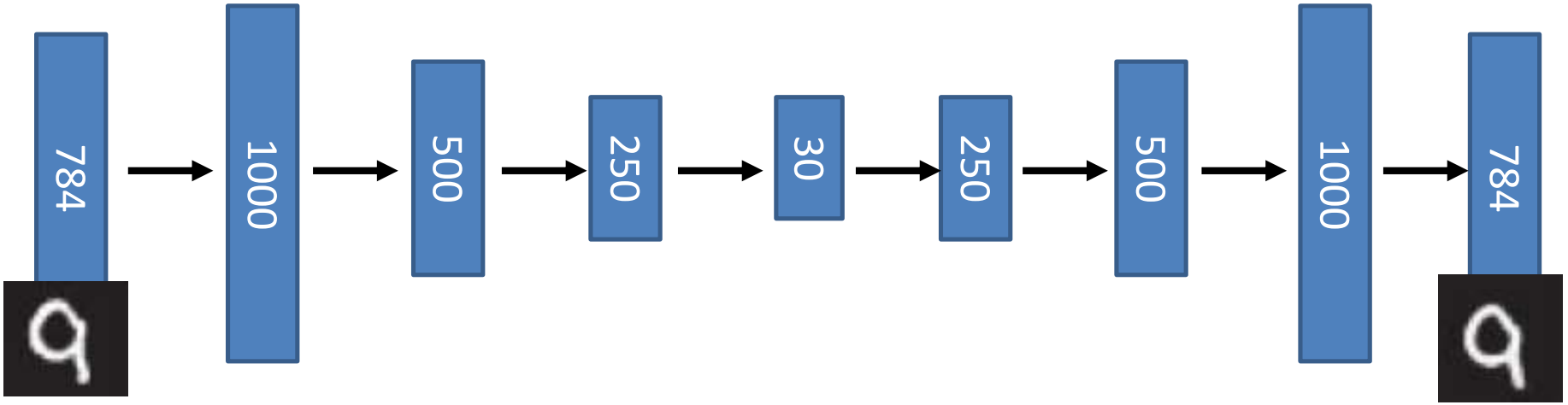
Original Image

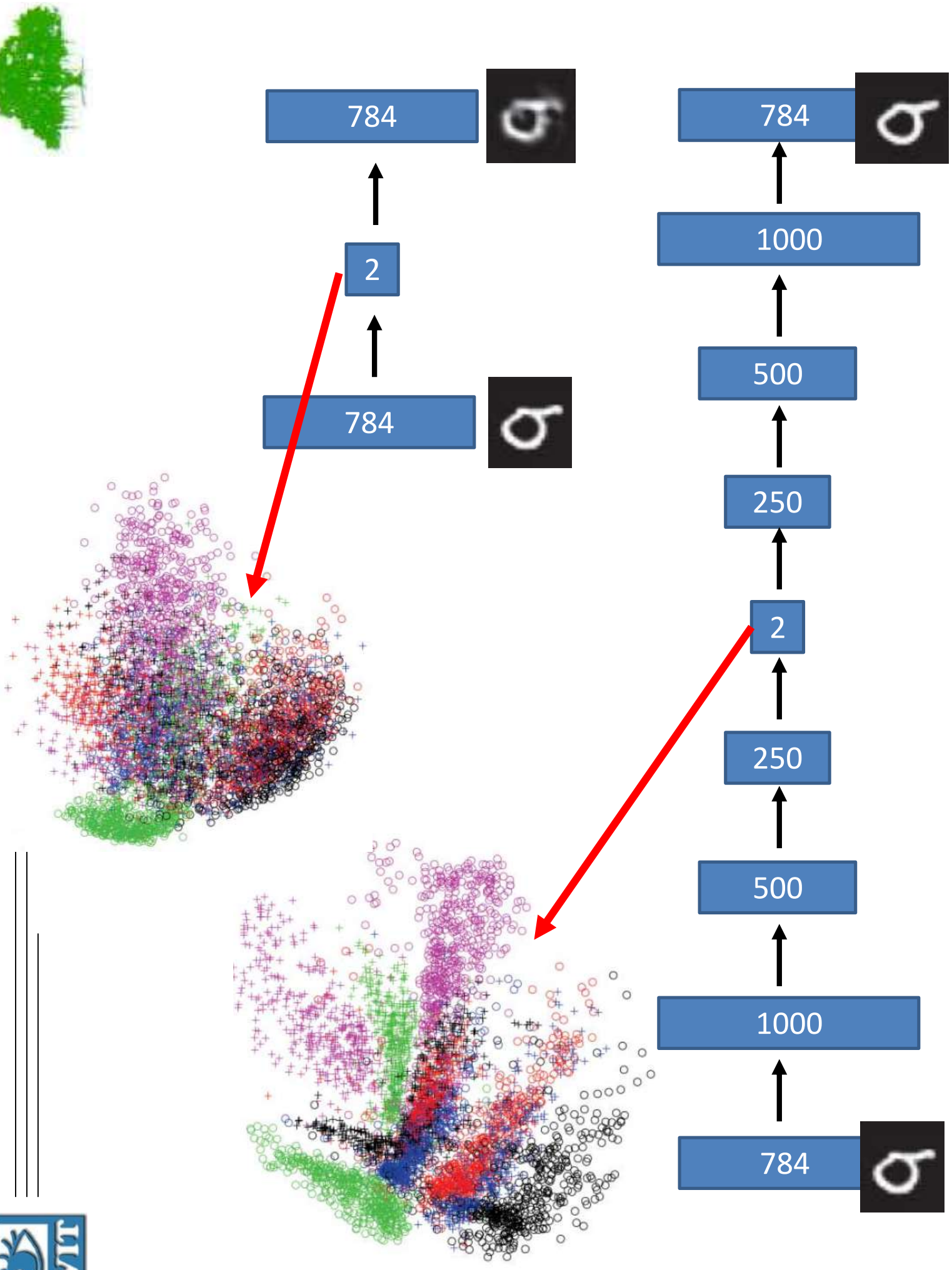
PCA

Deep Auto-encoder





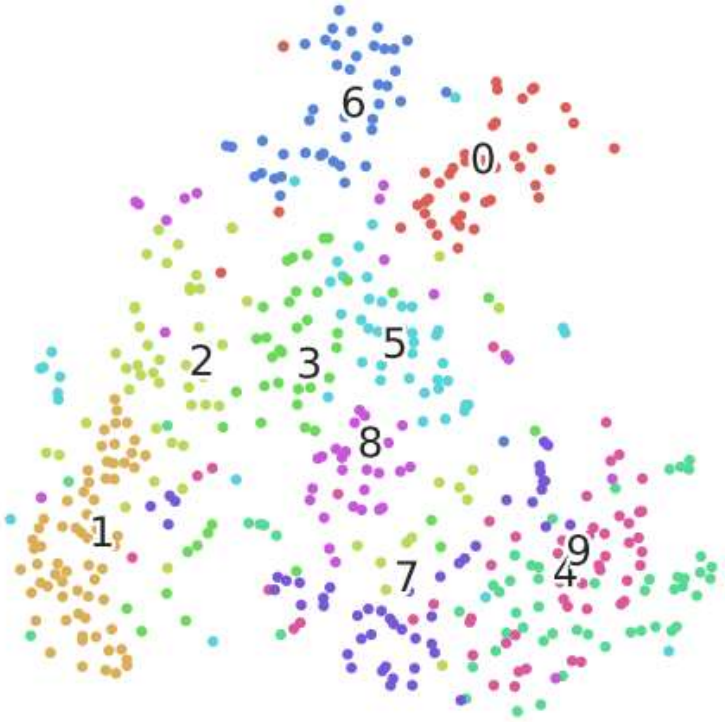



784
2
784
784
1000
500
250
2
250
500
1000
784

Deep Auto-encoder - Example

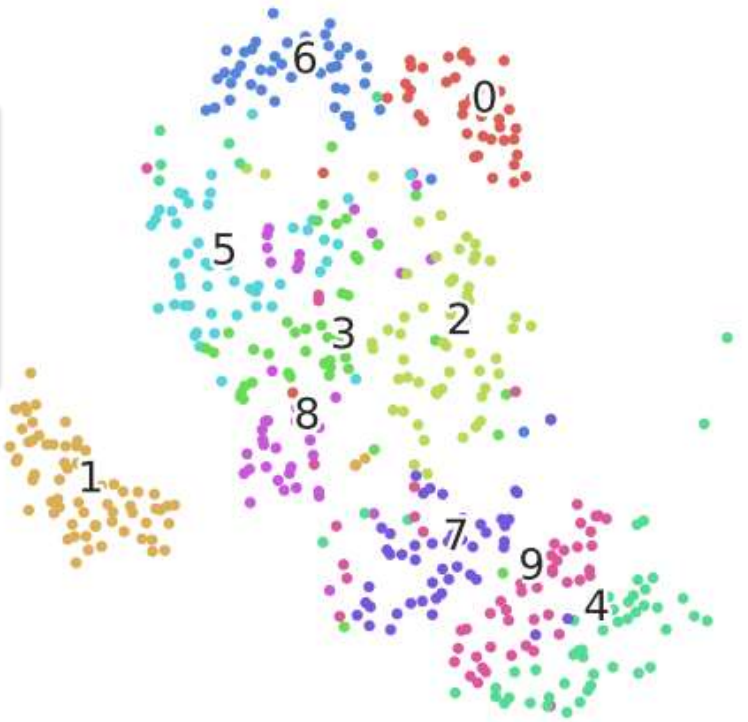
NN Encoder

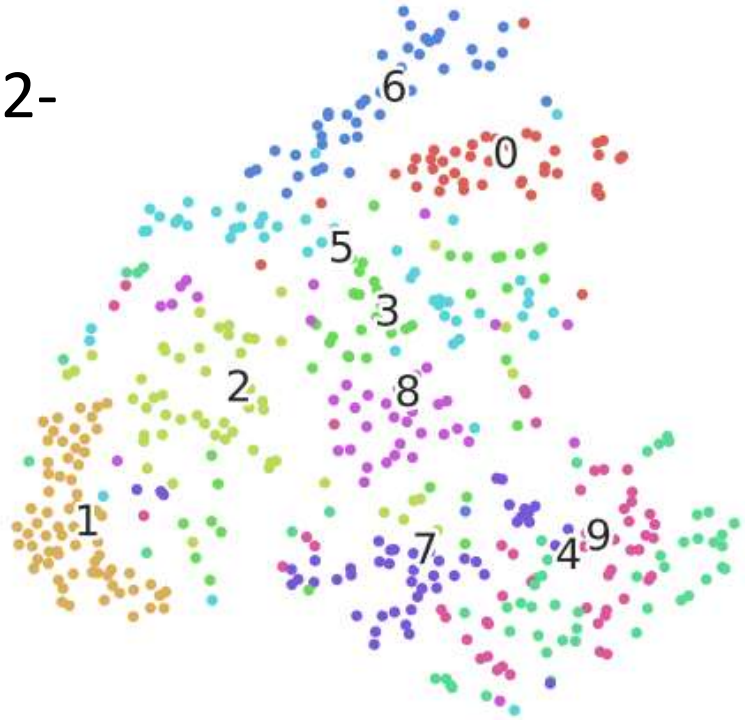
c



Pixel -> tSNE

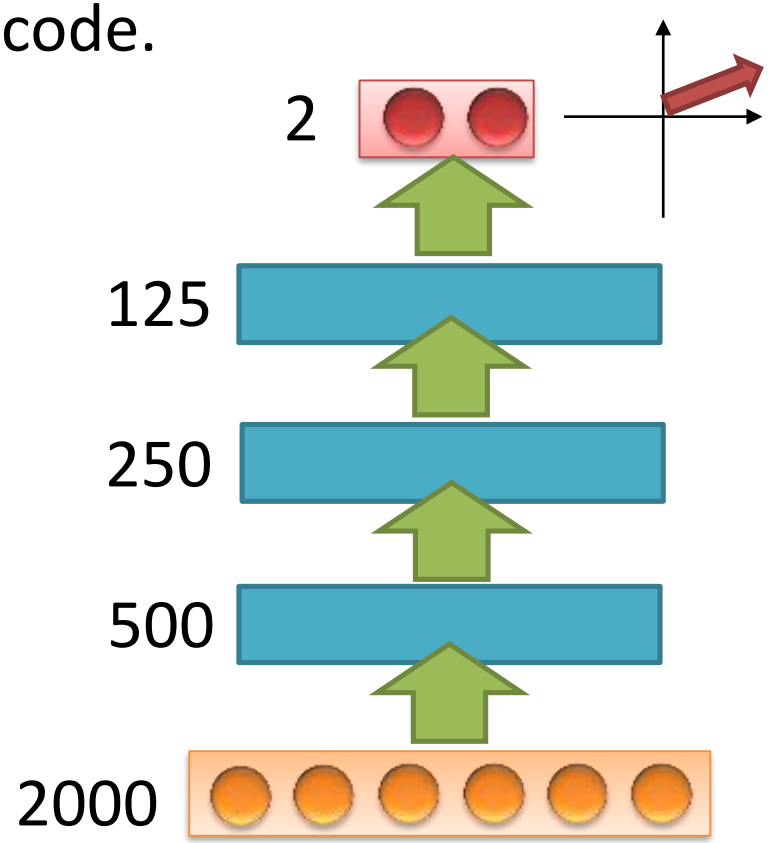
PCA 32-dim



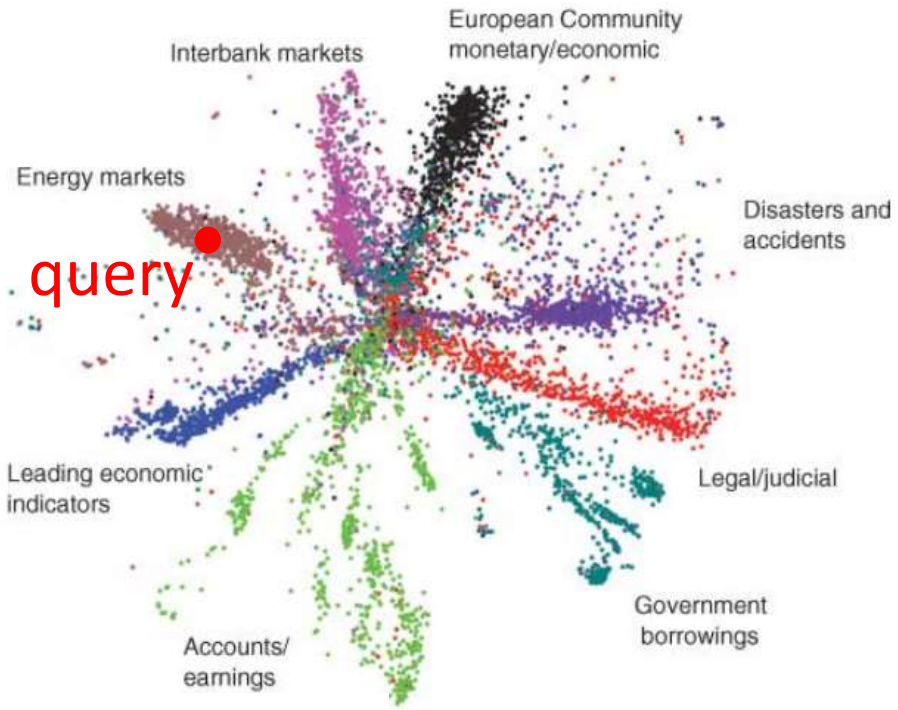


Auto-encoder – Text Retrieval

The documents talking about the same thing will have close code.



Bag-of-word
(document or query)



LSA: project documents to 2 latent topics

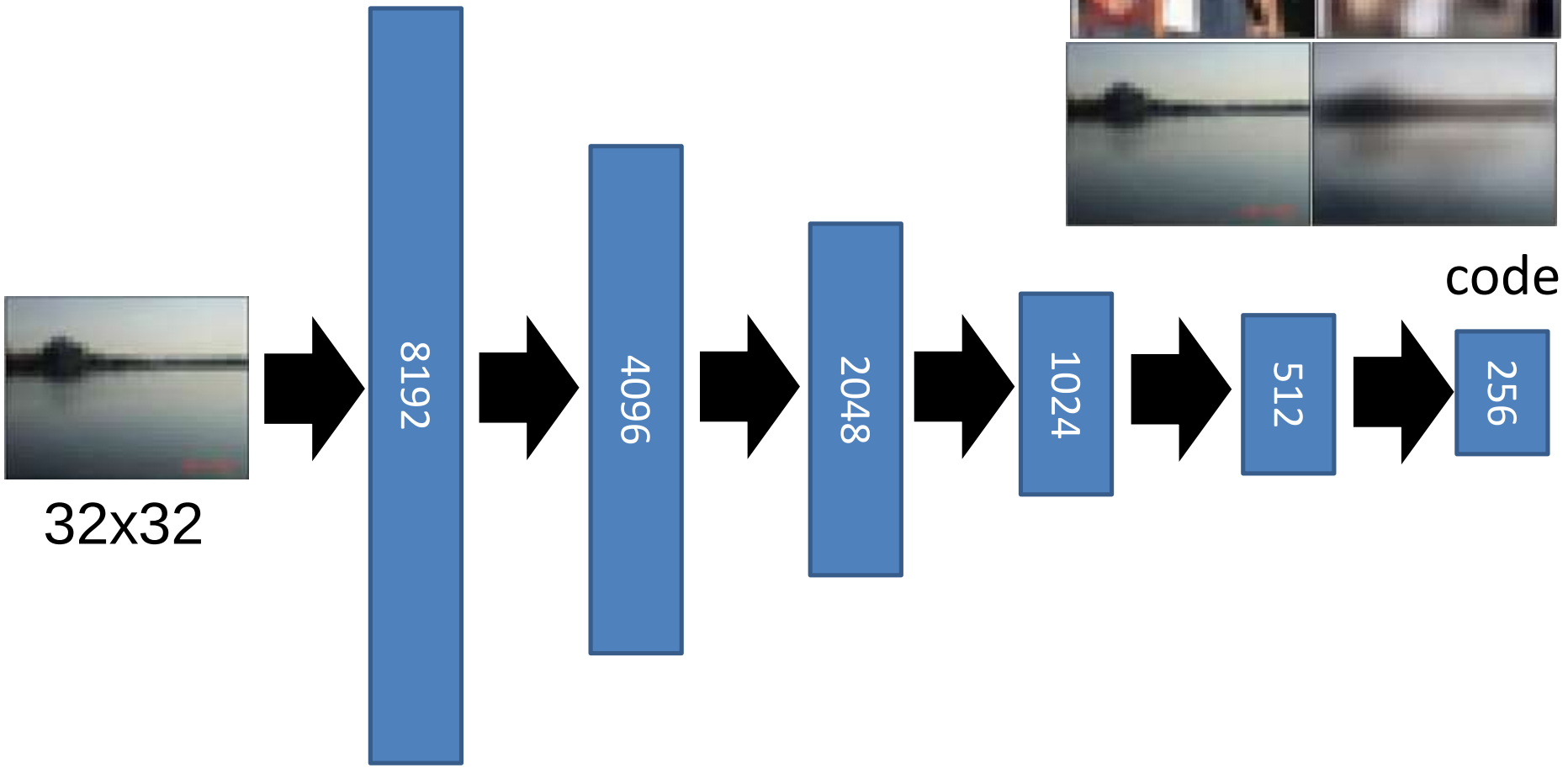
Auto-encoder – Similar Image Search

Retrieved using Euclidean distance in pixel intensity space

(Images from Hinton's slides on Coursera)

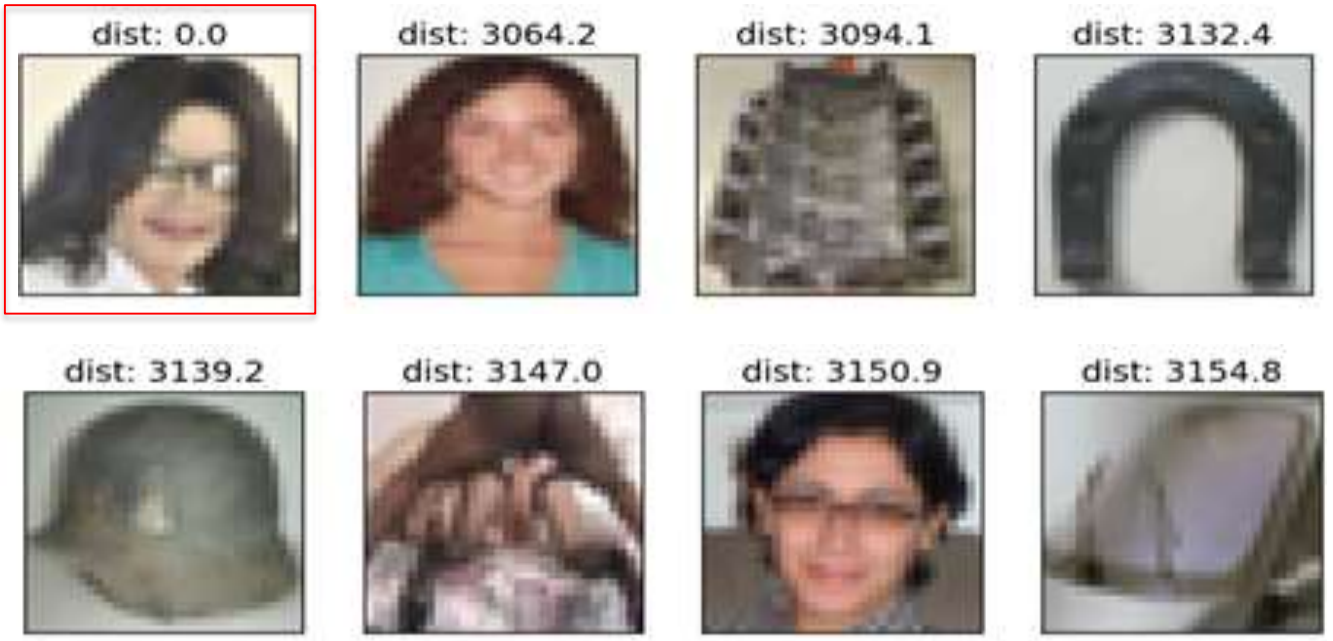
Reference: Krizhevsky, Alex, and Geoffrey E. Hinton. "Using very deep autoencoders for content-based image retrieval." *ESANN*. 2011.

Auto-encoder – Similar Image Search



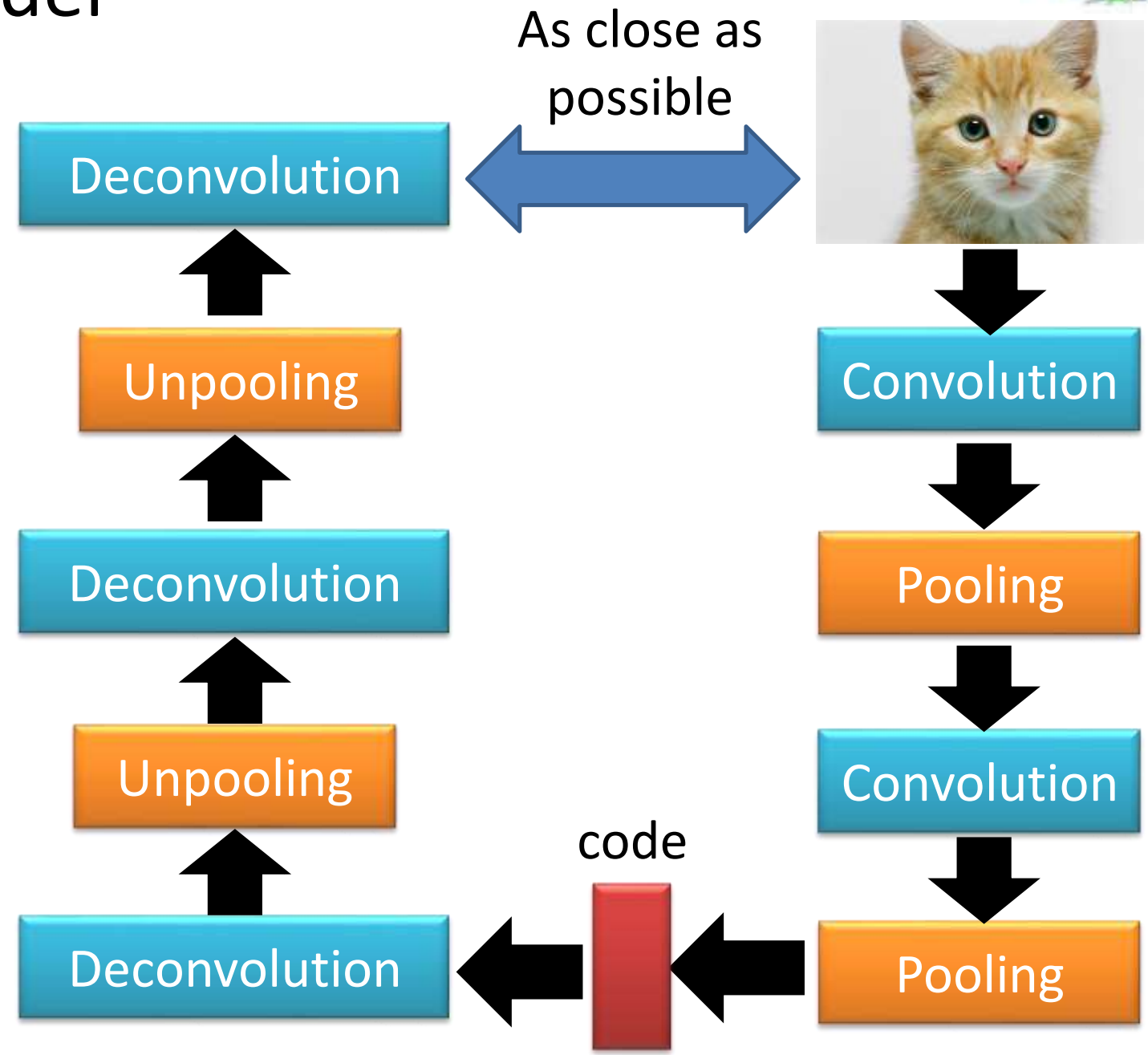
(crawl millions of images from the Internet)

Retrieved using Euclidean distance in pixel intensity space

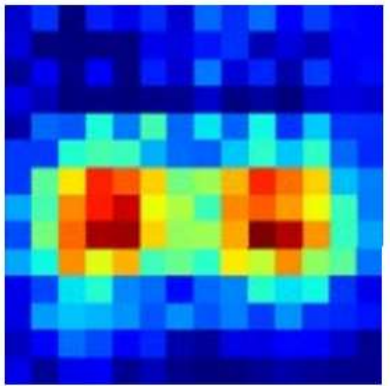


retrieved using 256 codes

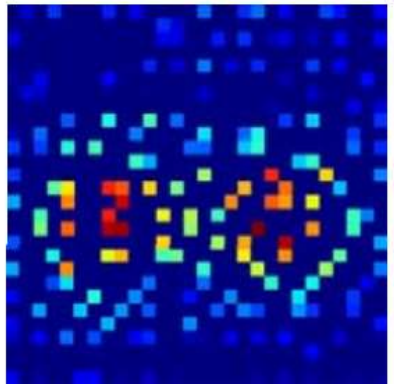
Auto-encoder for CNN



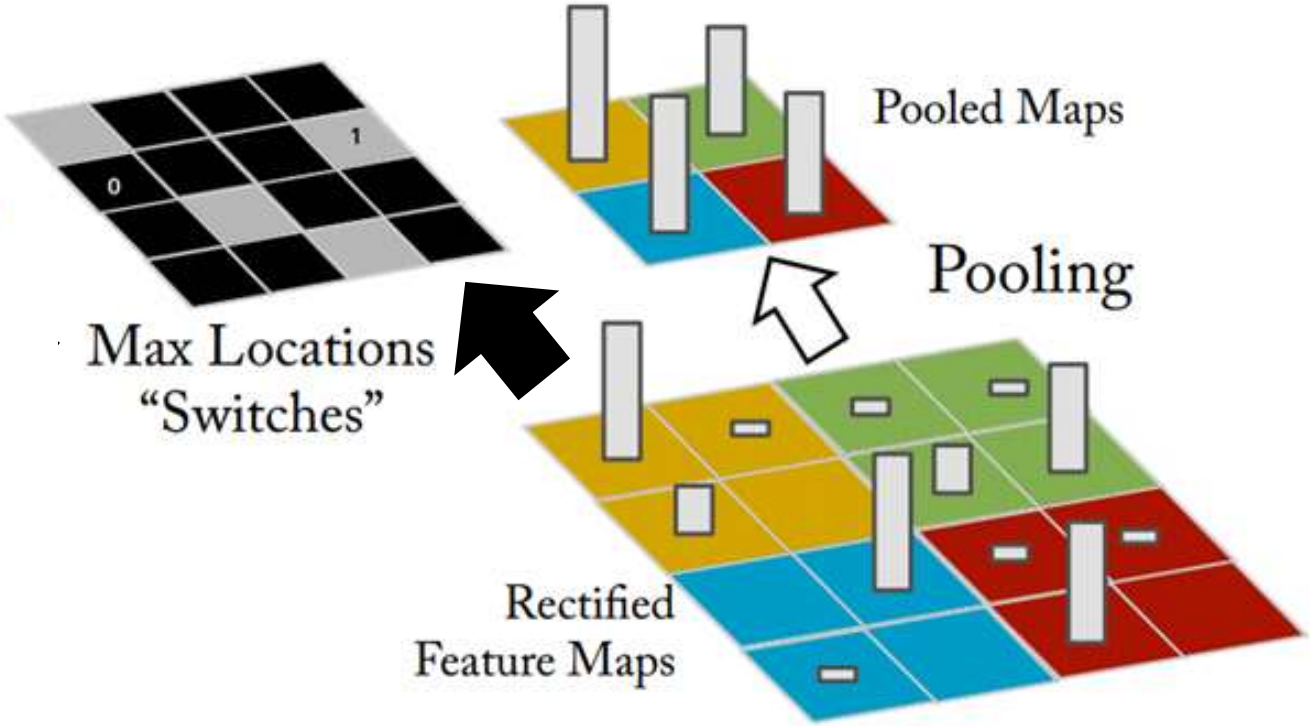
CNN -Unpooling



14 x 14



28 x 28

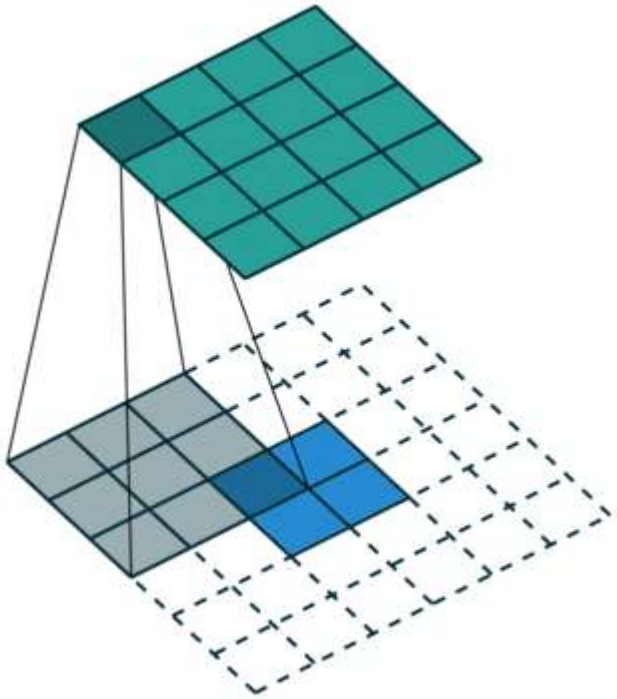


Alternative: simply repeat the values

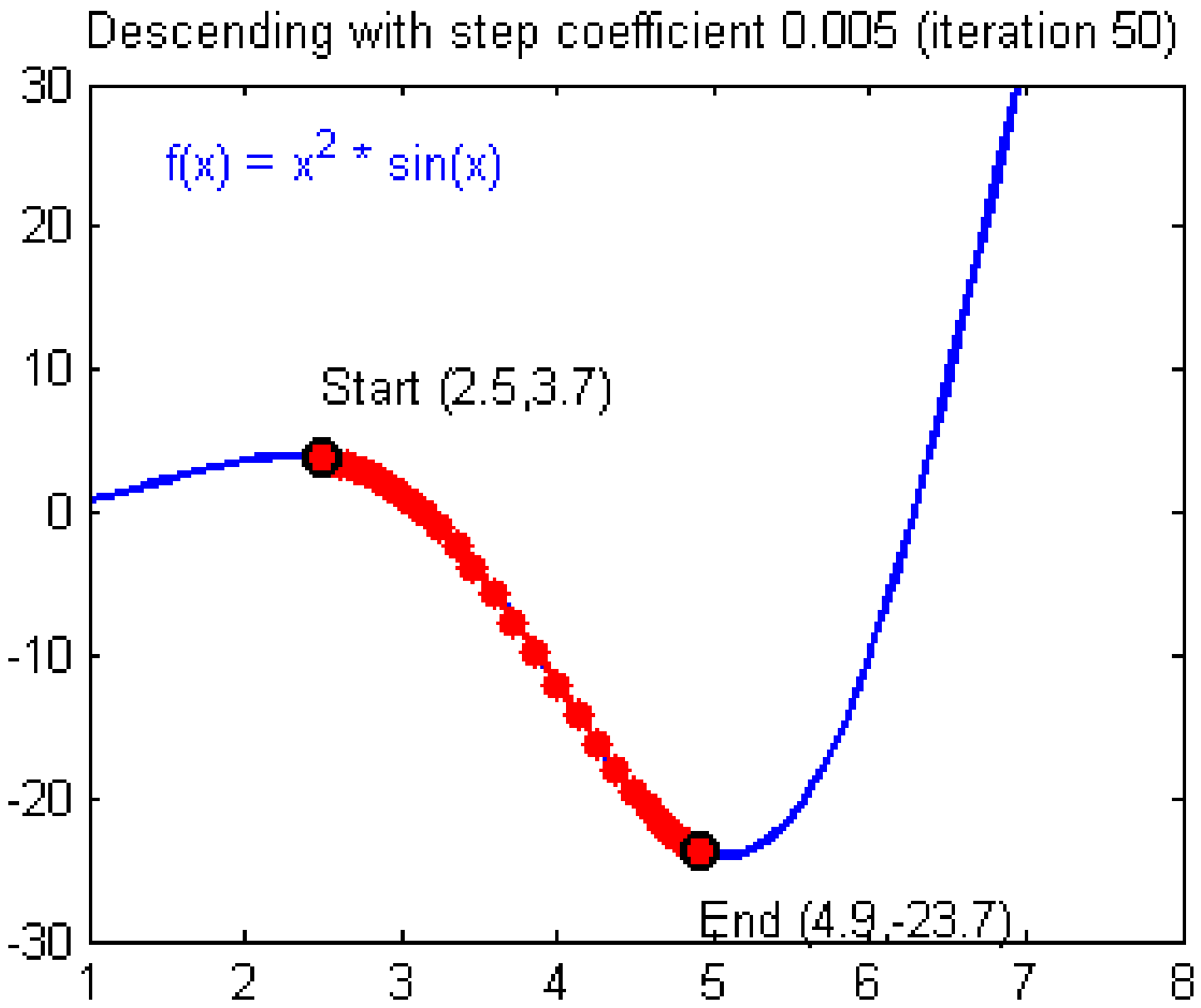
Source of image : https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/image_segmentation.html

Deconvolution

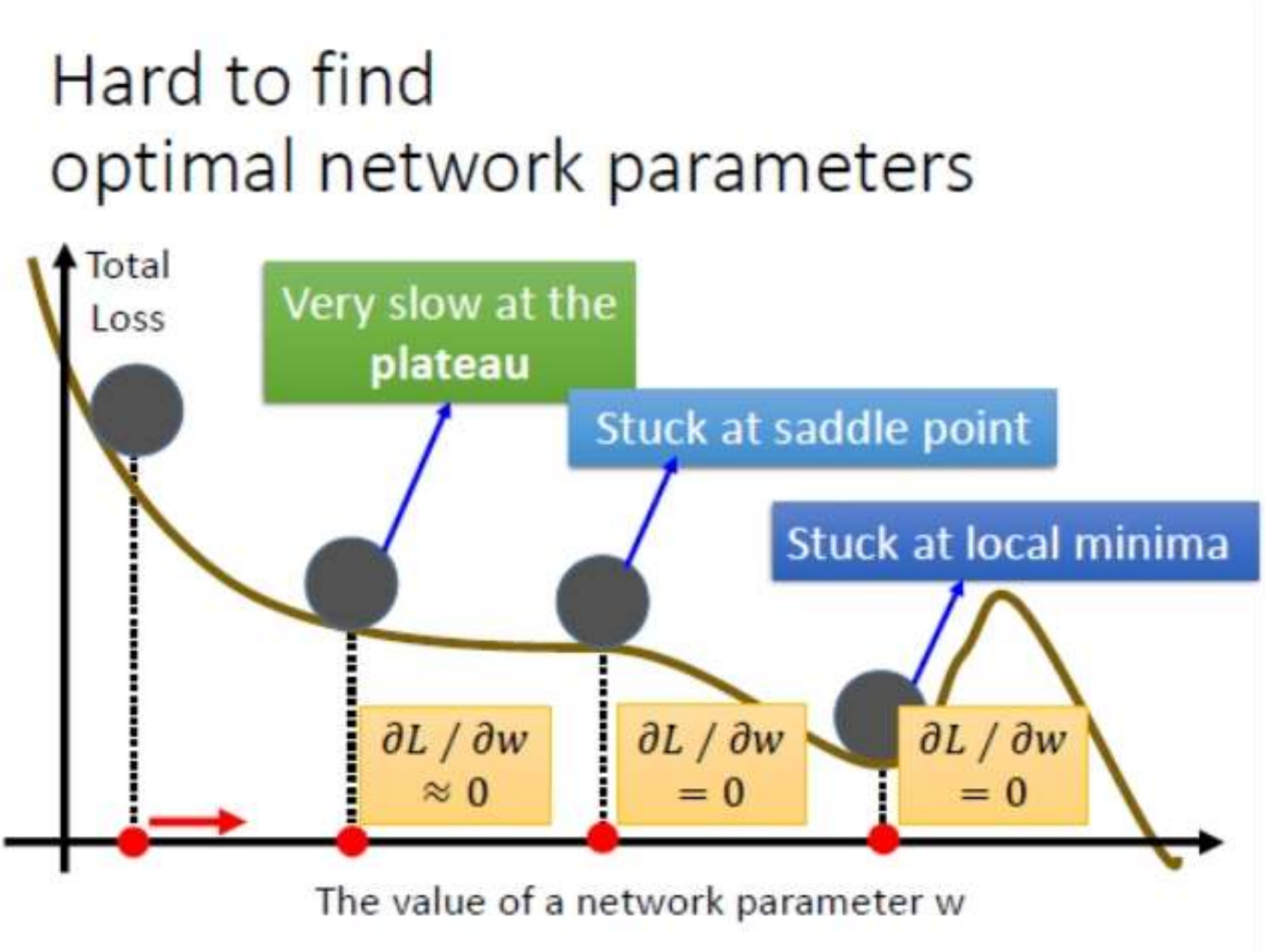
Actually, deconvolution is convolution.



SGD visualized



Learning – Suboptimal Modes



Momentum

-

$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}})$$

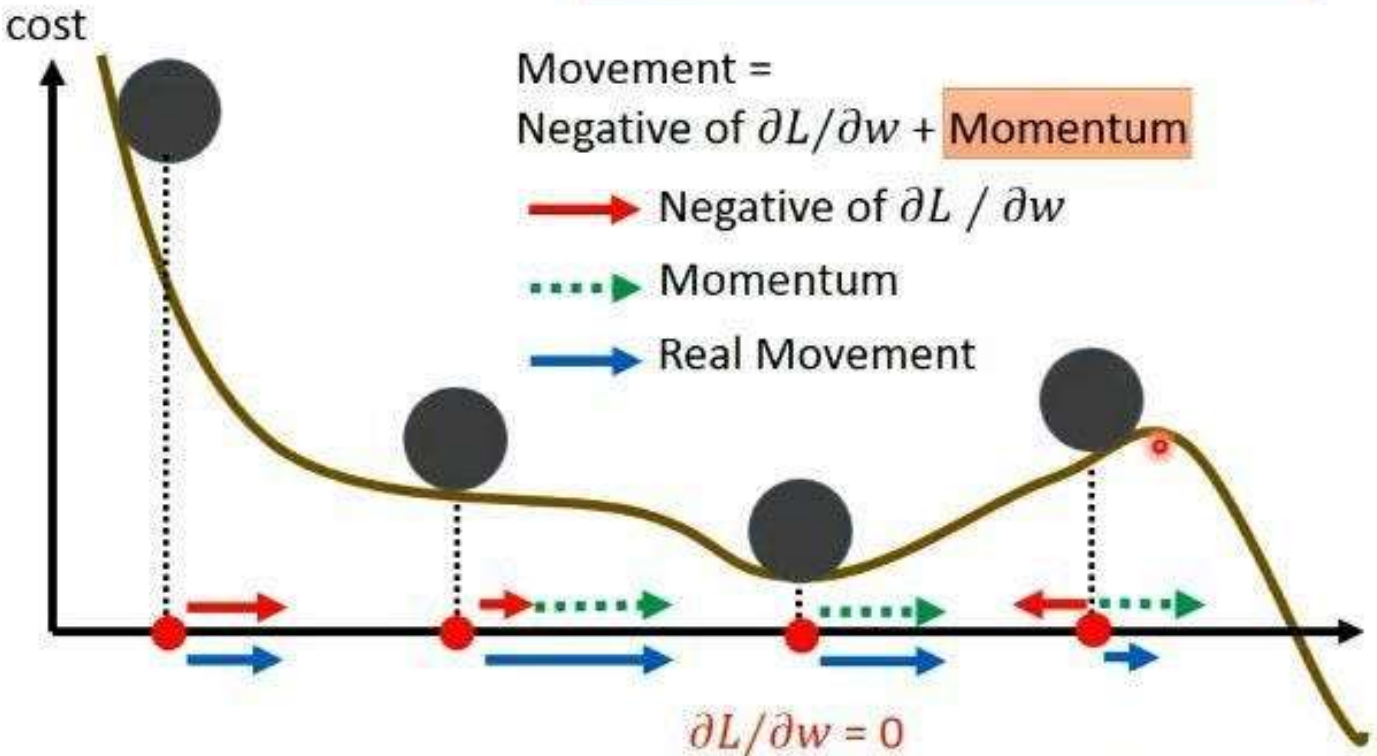
weight increment — learning rate — weight gradient

$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}}) + (\gamma * \Delta w_{ij}^{t-1})$$

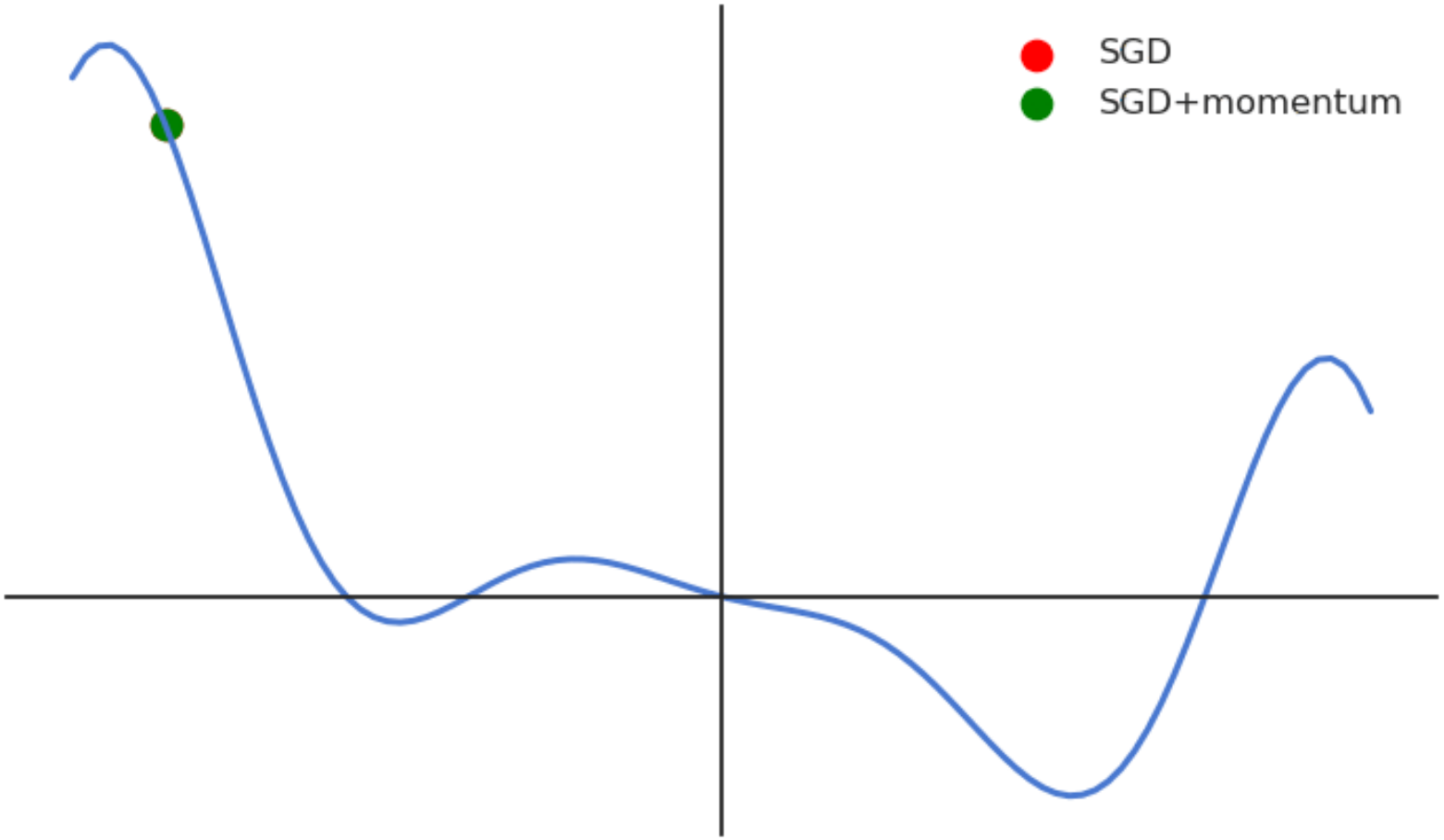
momentum factor — weight increment, previous iteration

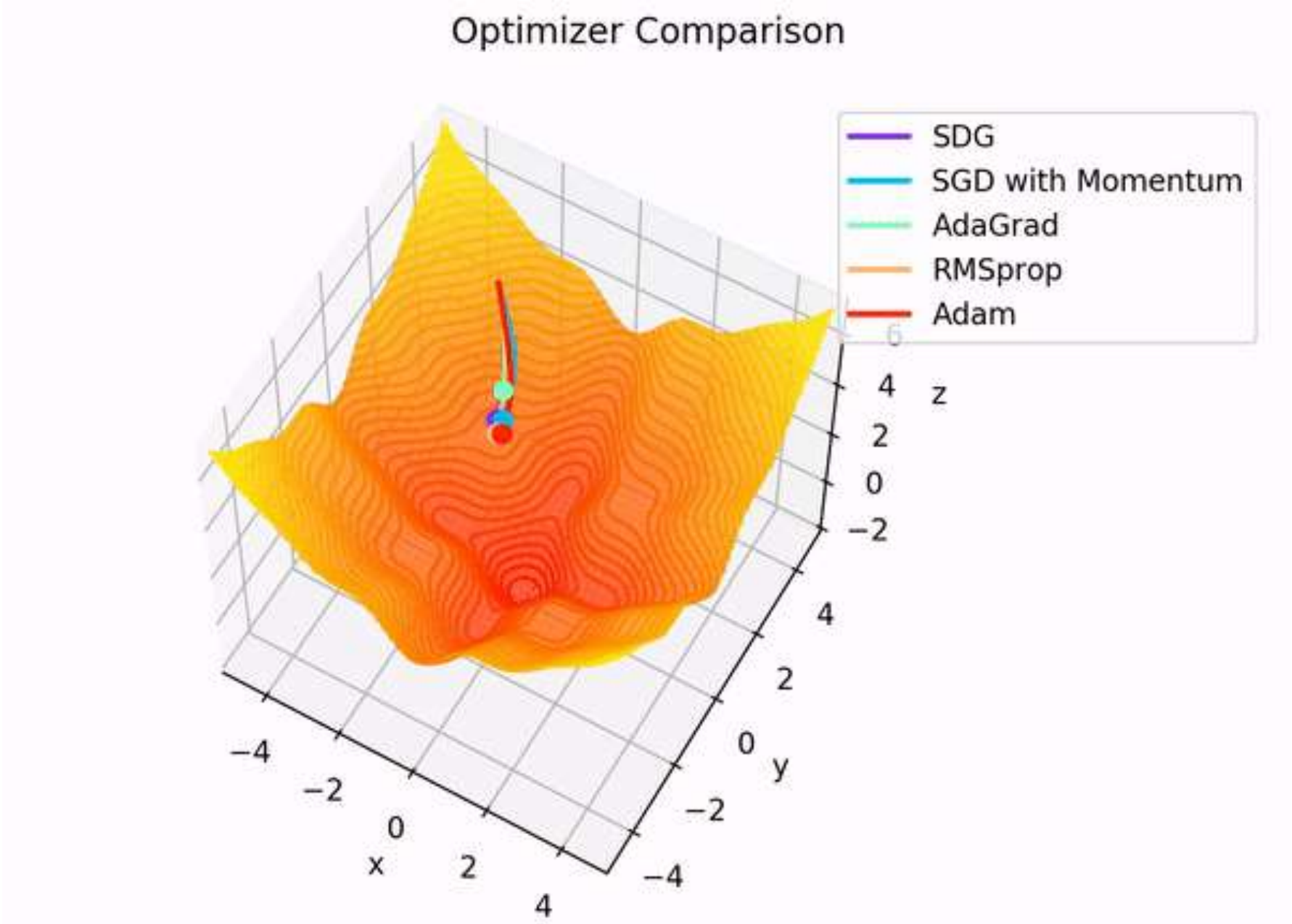
Momentum

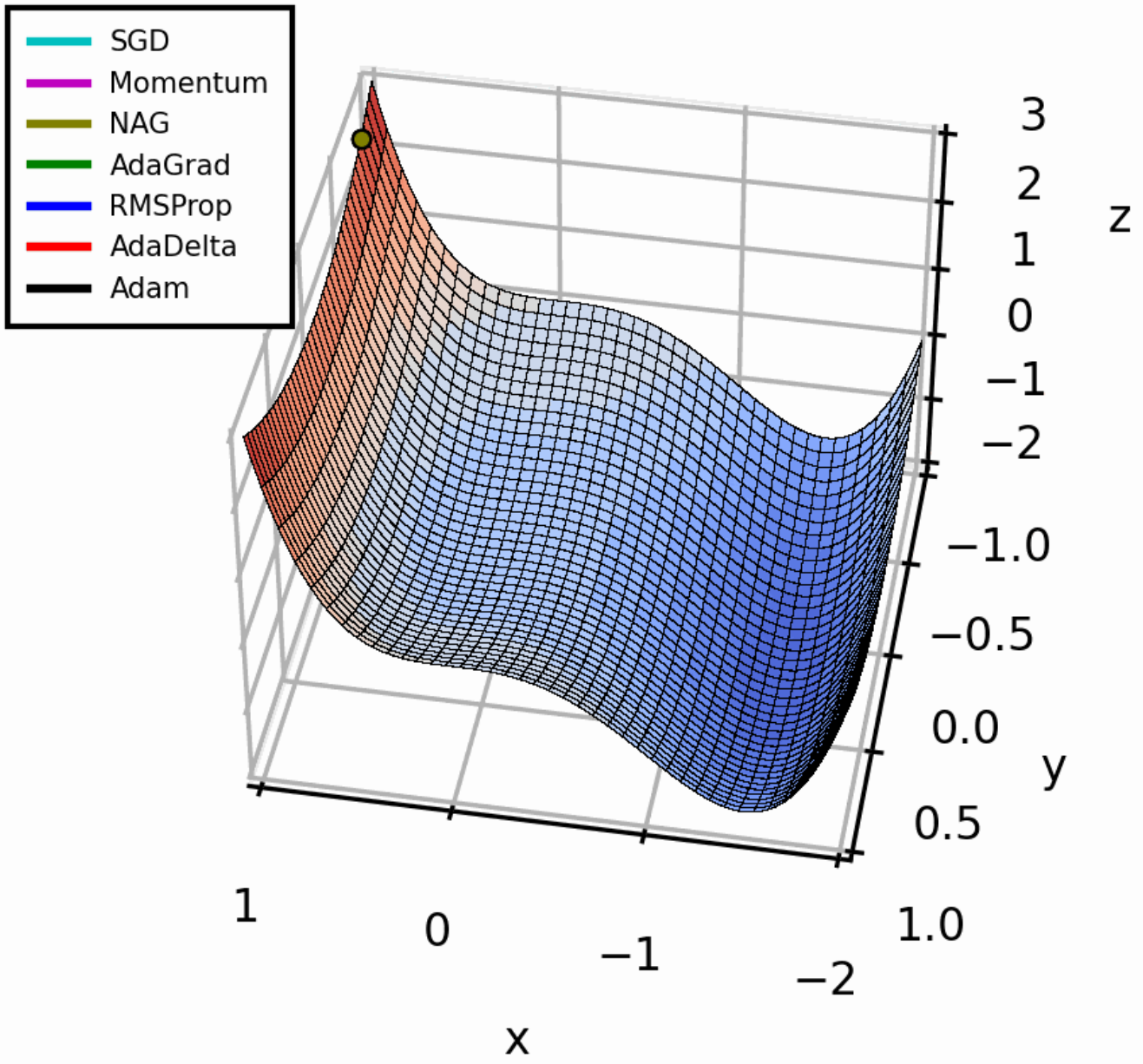
Still not guarantee reaching global minima, but give some hope .....

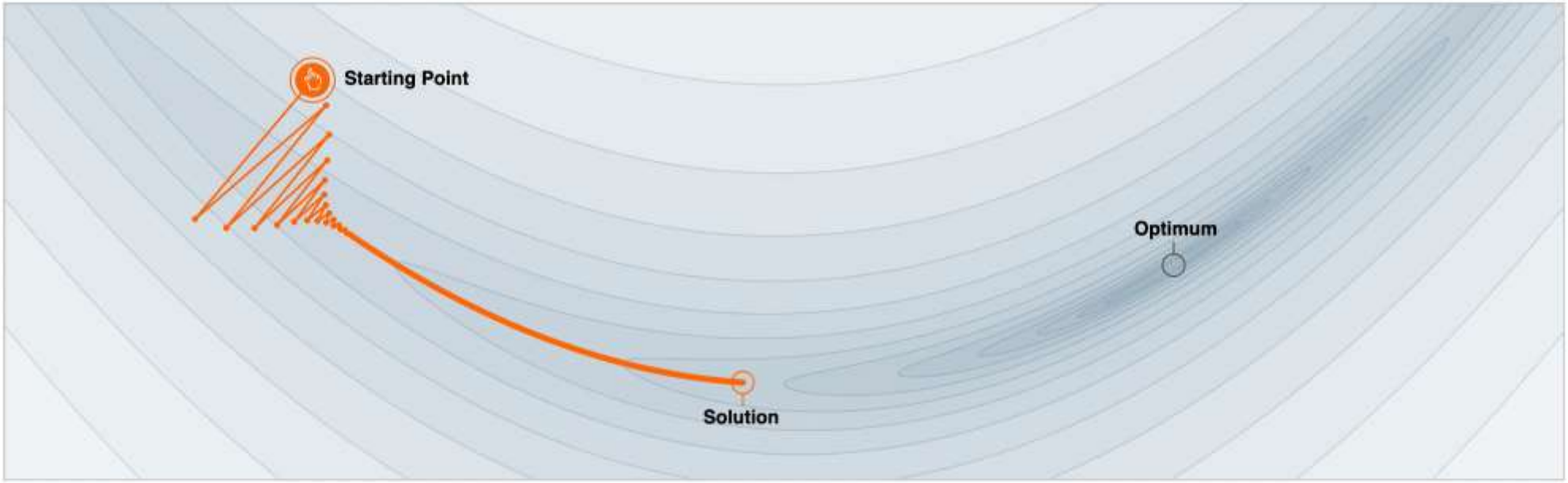


Momentum - visualized




Optimizer Comparison
SDG
SGD with Momentum
AdaGrad
RMSprop
Adam
z
6
4
2
0
−2
x
−4
−2
0
2
4
y
4
2
0
−2
−4


SGD
Momentum
NAG
AdaGrad
RMSProp
AdaDelta
Adam
3
2
1
0
−1
−2
z
−1.0
−0.5
0.0
0.5
1.0
y
1
0
−1
−2
x



Step-size α = 0.0030

0 0.003 0.006

Momentum β = 0.0

0.00 0.500 0.990

We often think of Momentum as a means of dampening oscillations and speeding up the iterations, leading to faster convergence. But it has other interesting behavior. It allows a larger range of step-sizes to be used, and creates its own oscillations. What is going on?

$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}})$$

weight increment — learning rate — weight gradient

$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}}) + (\gamma * \Delta w_{ij}^{t-1})$$

momentum factor — weight increment, previous iteration

Summary of learning methods for neural networks

For small datasets (e.g. 10,000 cases) or bigger datasets without much redundancy, use a full-batch method.

- Conjugate gradient, LBFGS ...
- adaptive learning rates, rprop ...

For big, redundant datasets use mini-batches.

- Try gradient descent with momentum.
- Try rmsprop (with momentum ?)

Why there is no simple recipe:

Neural nets differ a lot:

- Very deep nets (especially ones with narrow bottlenecks).
- Recurrent nets.
- Wide shallow nets.

Tasks differ a lot:

- Some require very accurate weights, some don't.

Optimization parameters

CLASS `torch.optim.SGD(params, lr=<required parameter>, momentum=0, dampening=0, weight_decay=0, nesterov=False)` [SOURCE]

Implements stochastic gradient descent (optionally with momentum).

Nesterov momentum is based on the formula from On the importance of initialization and momentum in deep learning.

Parameters

- **params** (*iterable*) – iterable of parameters to optimize or dicts defining parameter groups
- **lr** (*float*) – learning rate
- **momentum** (*float, optional*) – momentum factor (default: 0)
- **weight_decay** (*float, optional*) – weight decay (L2 penalty) (default: 0)
- **dampening** (*float, optional*) – dampening for momentum (default: 0)
- **nesterov** (*bool, optional*) – enables Nesterov momentum (default: False)

L2 Regularization

$$\text{Modified loss function} = \text{Loss function} + \lambda \sum_{i=1}^{n} W_i^2$$