CPH 100A: Modeling Images and Volumes: Convolutional Neural Networks

Instructor: Adam Yala, PhD (yala@berkeley.edu)

Computational
PRECISION HEALTH

Berkeley | UCSF

Agenda

Recap

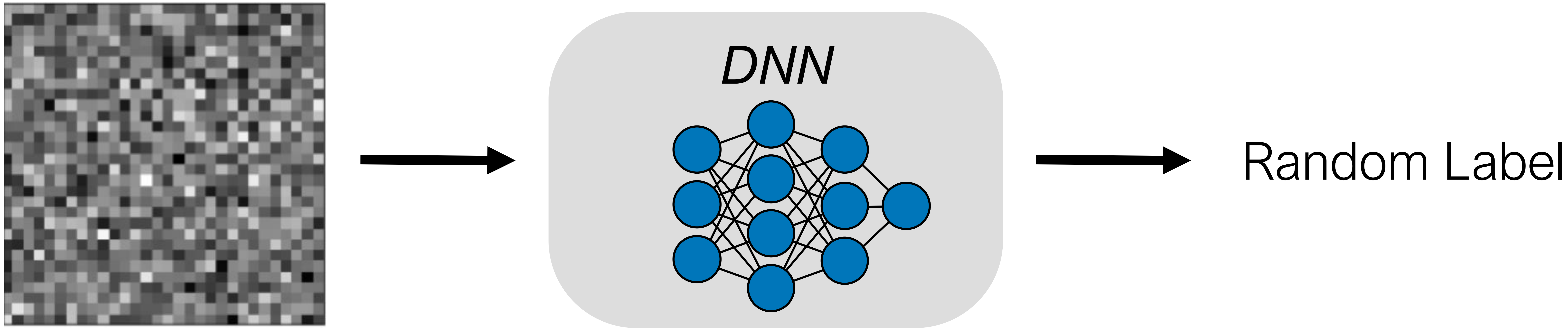
Failure modes of fully-connected neural networks

Convolutions

Pooling

CNNs across modalities

Well-trained Networks can learn anything

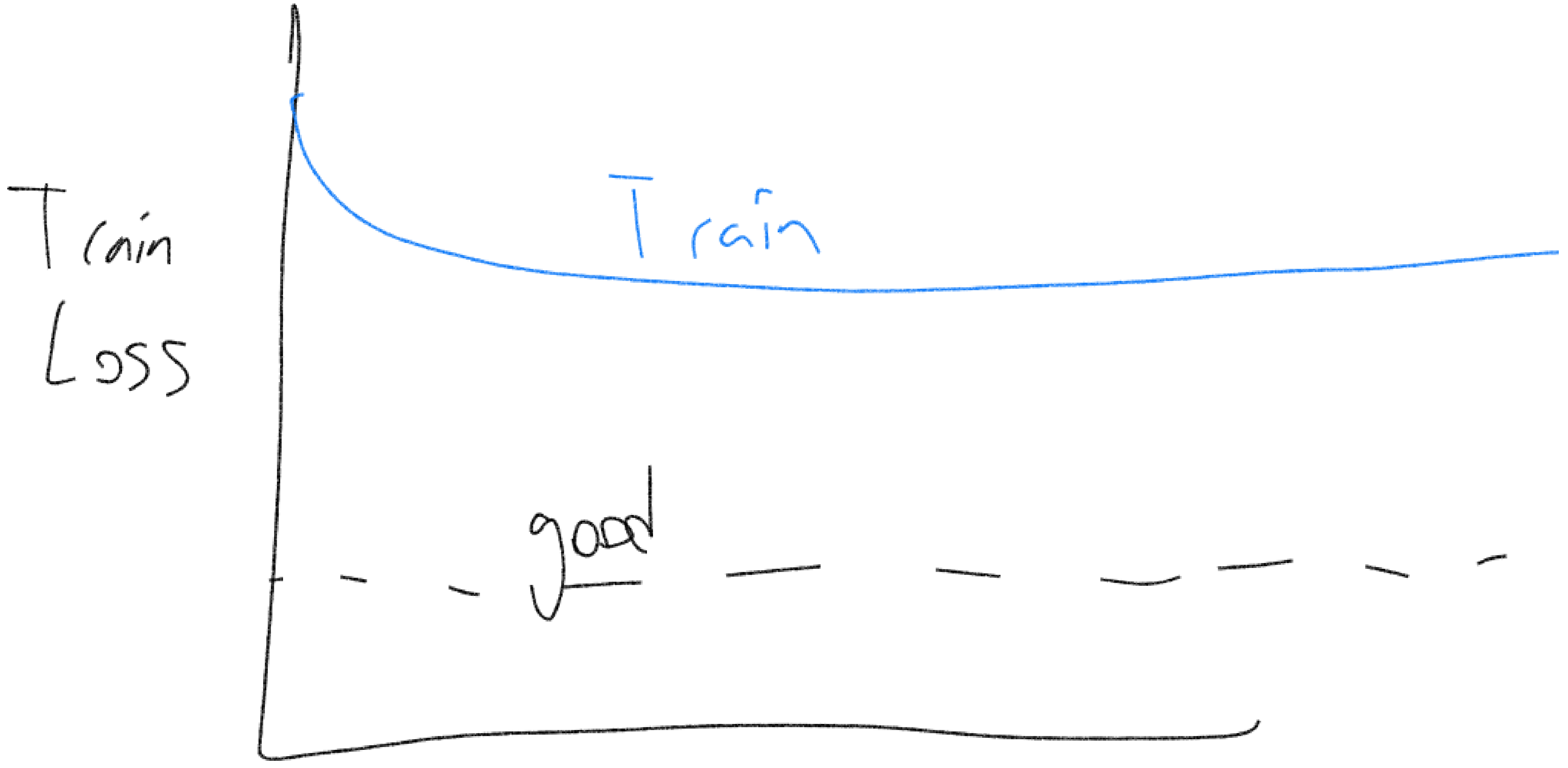


Understanding deep learning (still) requires rethinking generalization

Authors: Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, Oriol Vinyals Authors Info & Claims

Communications of the ACM, Volume 64, Issue 3 • March 2021 • pp 107–115 • https://doi.org/10.1145/3446776

Failure modes of Neural Network Optimization



Underfitting

Overfitting

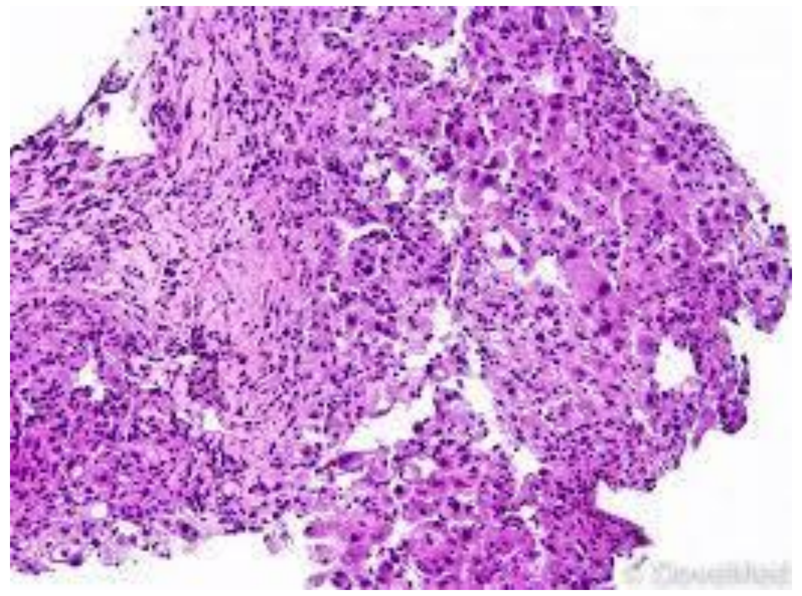
Why did the neural networks fail to train?

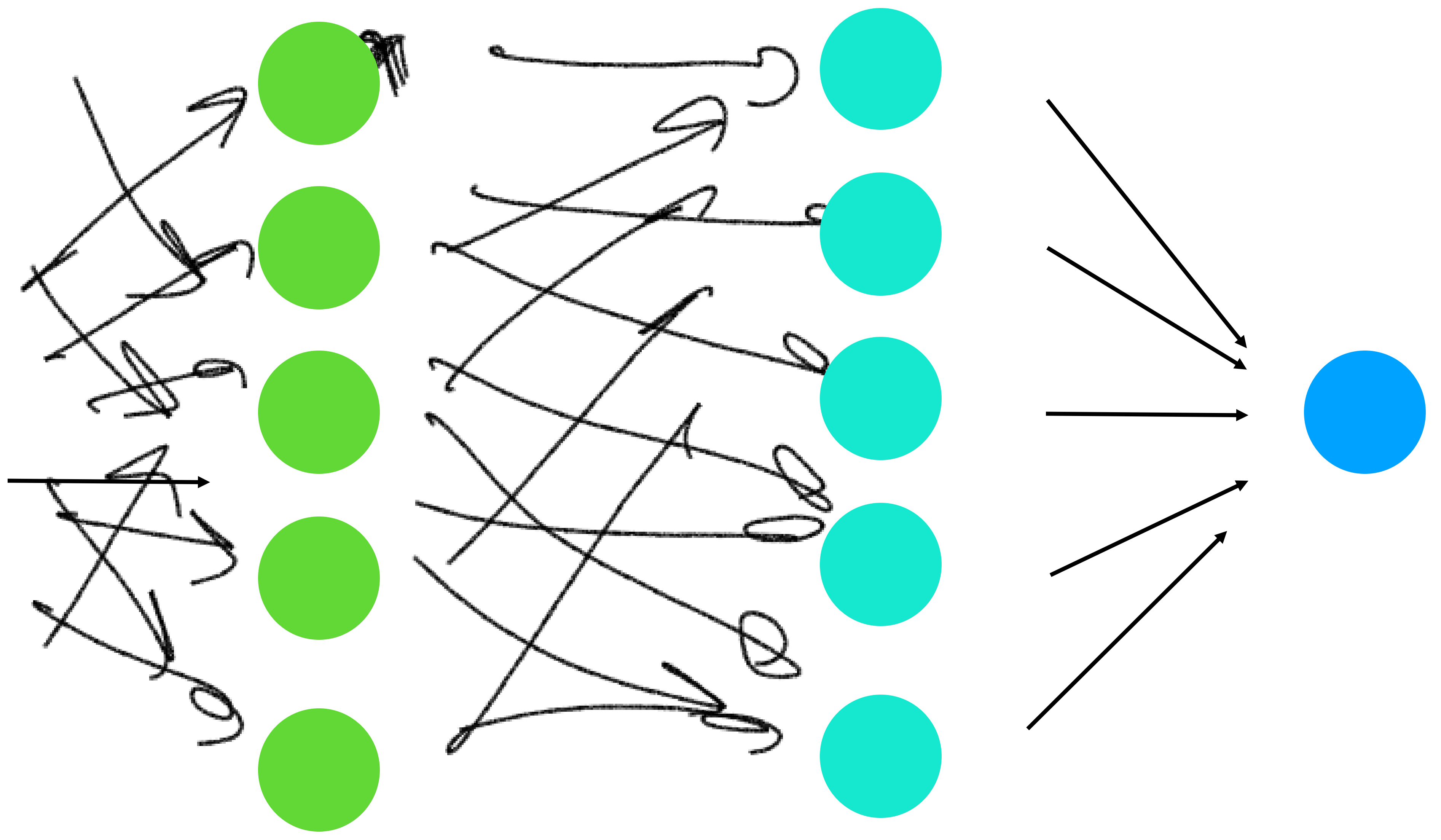
Complex Interaction between:

- Initialization
- Hypothesis Class
- Optimizer
- Learning Objective
- Data

Choosing scales of random init

$D^1 \times D^0$ Θ^1 $D^2 \times D^1$ Θ^2 $1 \times D^2$ Θ^P





$z^1_{D^1 \times 1}$ $z^2_{D^1 \times 1}$

$L(p,y) = \frac{1}{2}(p-y)^2$

Control $\|z^\ell\|$

Control $\left\|\frac{\partial L}{\partial \theta}\right\|$

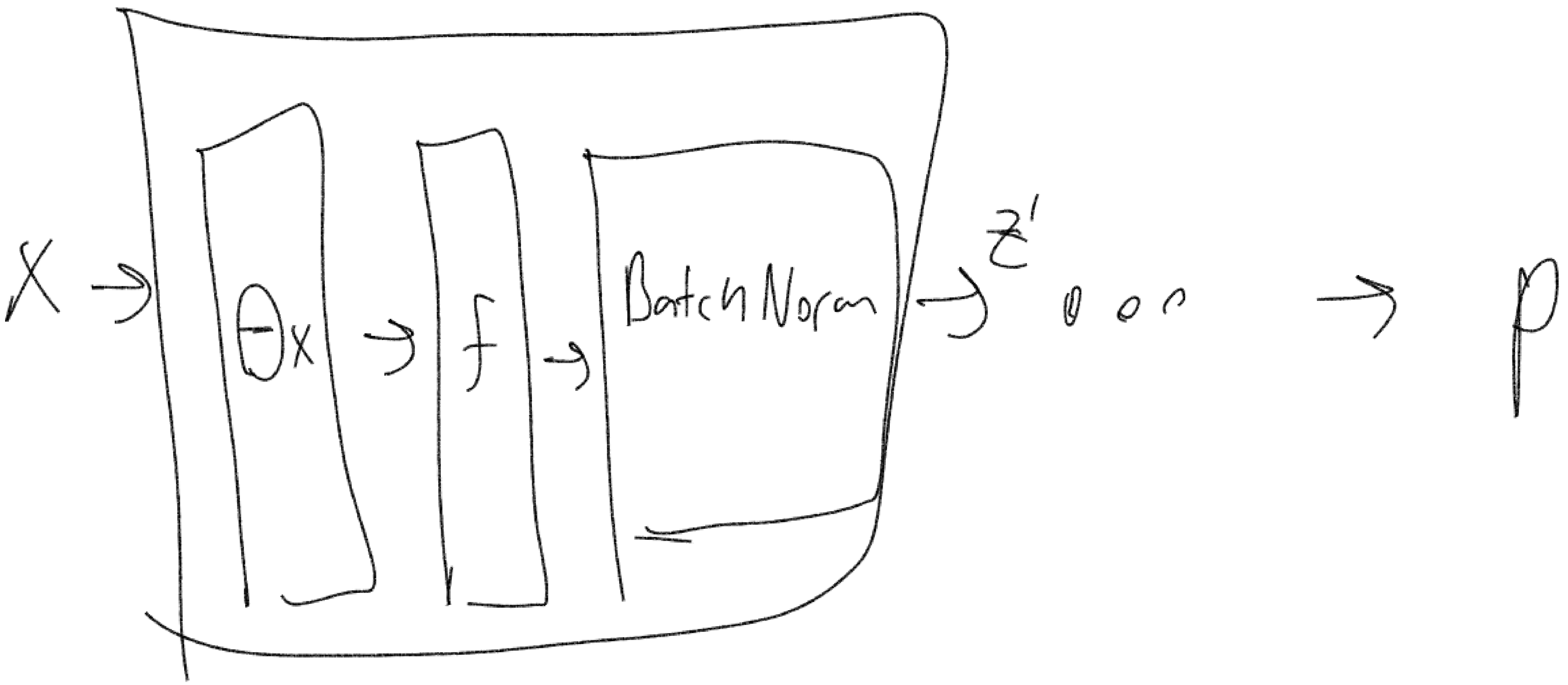
– depend on f

He init: ← ReLU

$\Theta \sim N(0, \sqrt{2/D})$

BatchNorm: Preventing activation collapse

$$\text{Batchnorm}(X) = \frac{X - \mu_x}{\sqrt{\sigma_x^2 + \epsilon}}$$



Use $\forall z^\ell$!

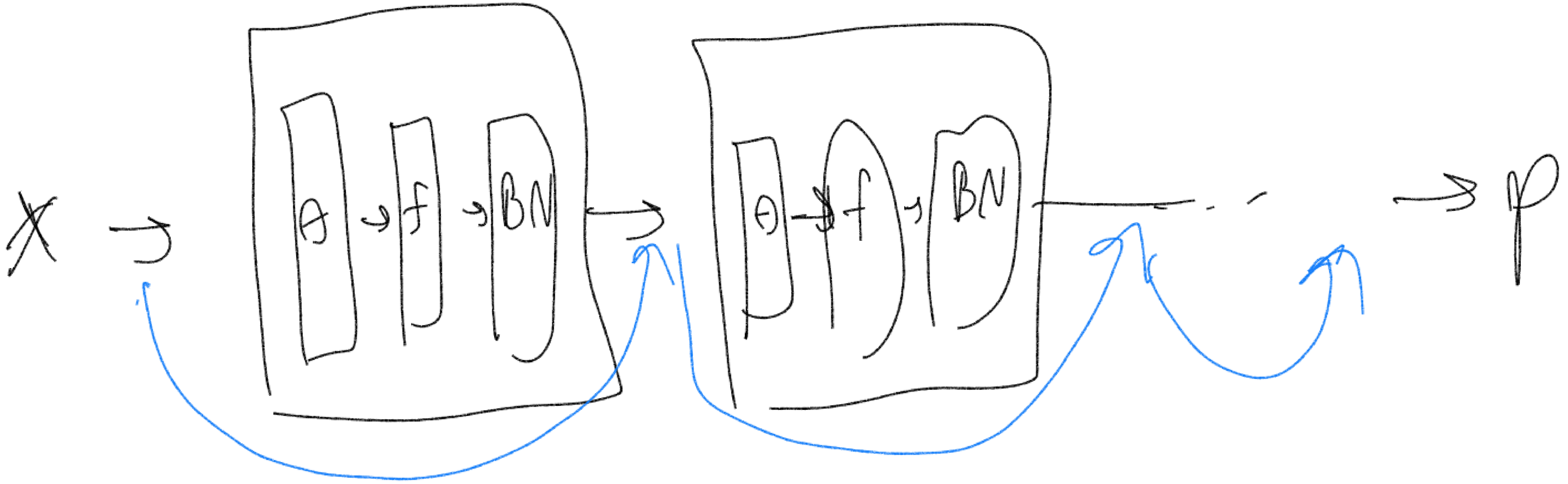
Now z^ℓ can't explode!

Residual connections: Skipping training bottlenecks

Add Skip / Residual

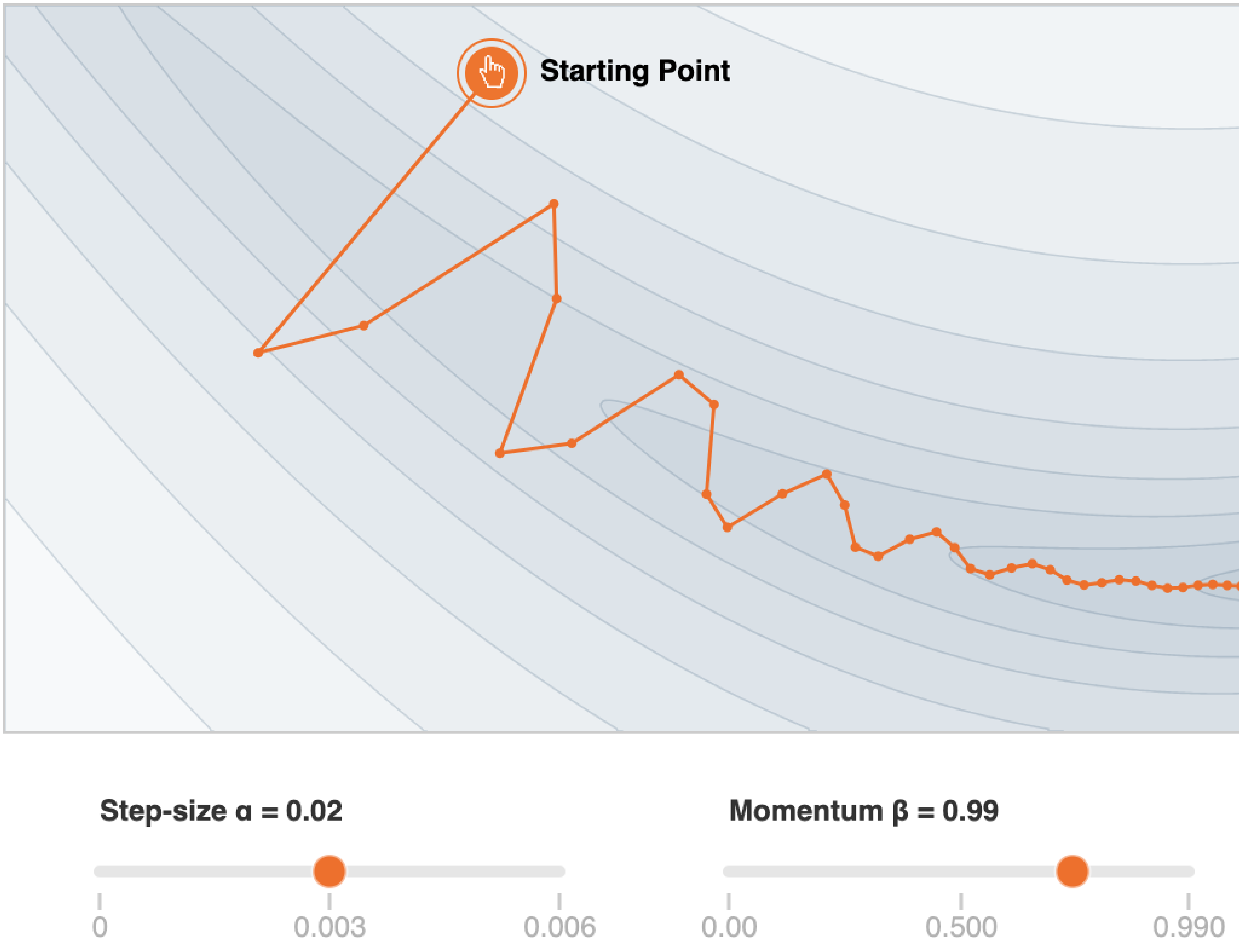
$$z^{\ell} = f(\underbrace{\Theta z^{\ell-1} + b}_{a^{\ell}}) + z^{\ell-1}$$

$$\frac{\partial z^{\ell}}{\partial z^{\ell-1}} = f'(a^{\ell})\,\Theta^{\ell} \boxed{+1}$$



Allows training deep NNs!

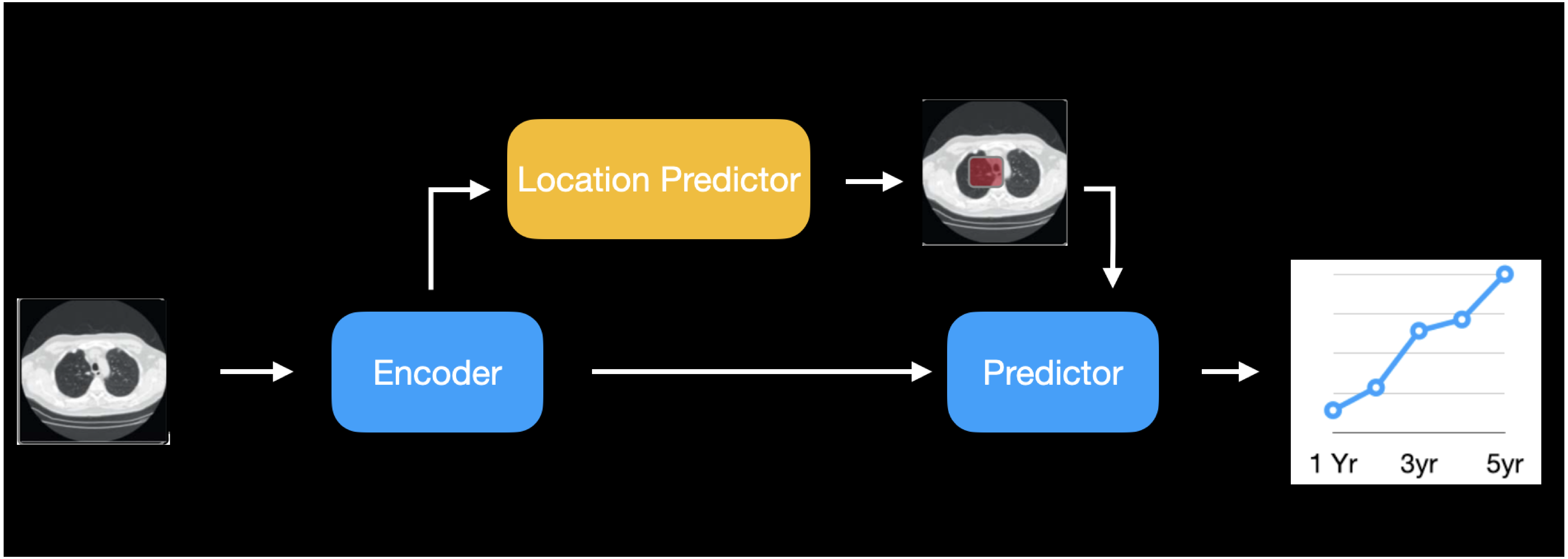
Momentum: Accelerating SGD



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?

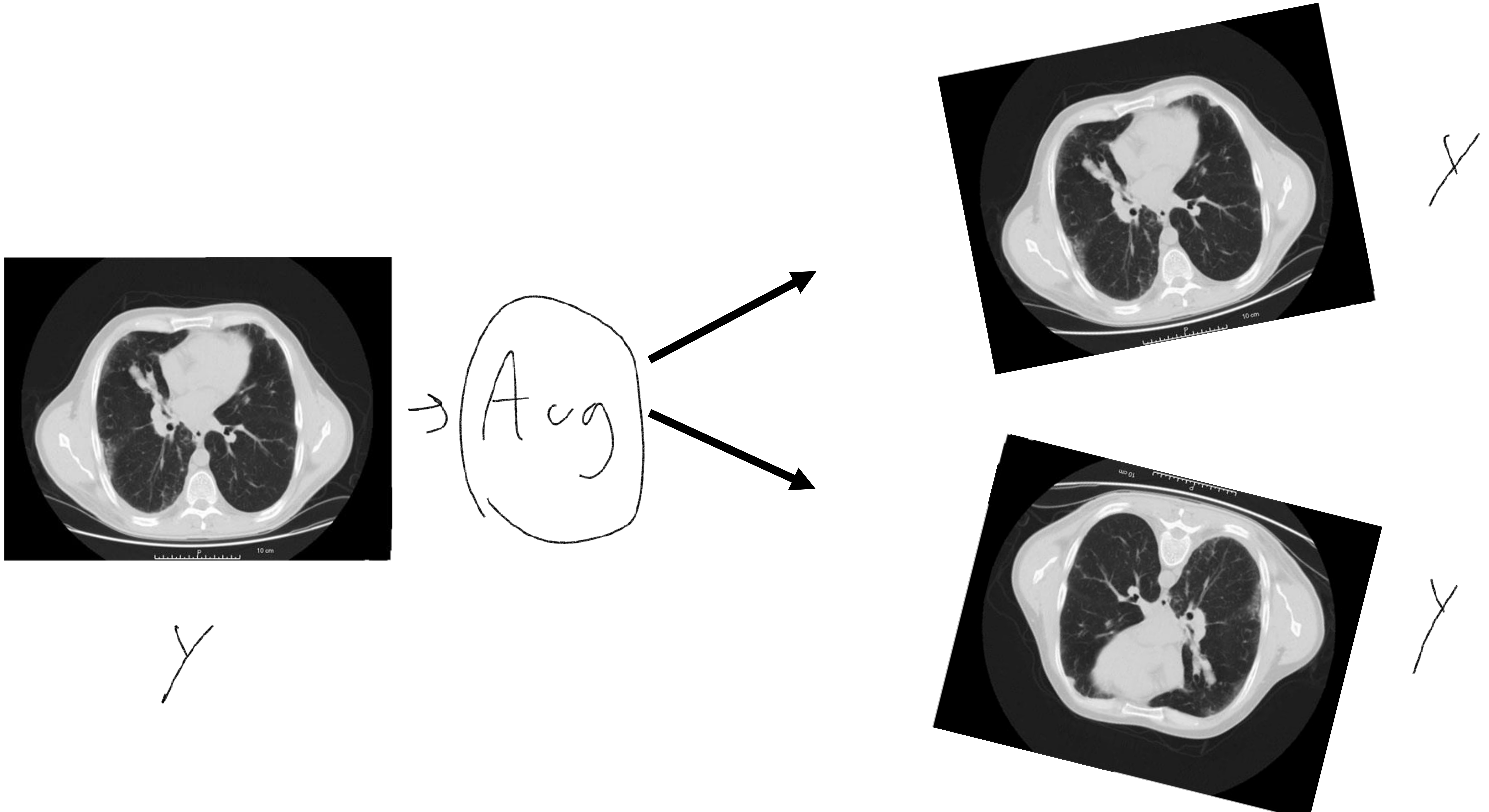
https://distill.pub/2017/momentum/

Managing overfitting: Adding additional losses



$$\mathcal{L} = L_{Risk} + L_{Location} + L_{Weight\ Decay} \leftarrow \|\theta\|^2$$

Managing overfitting: Data Augmentation



Agenda

Recap

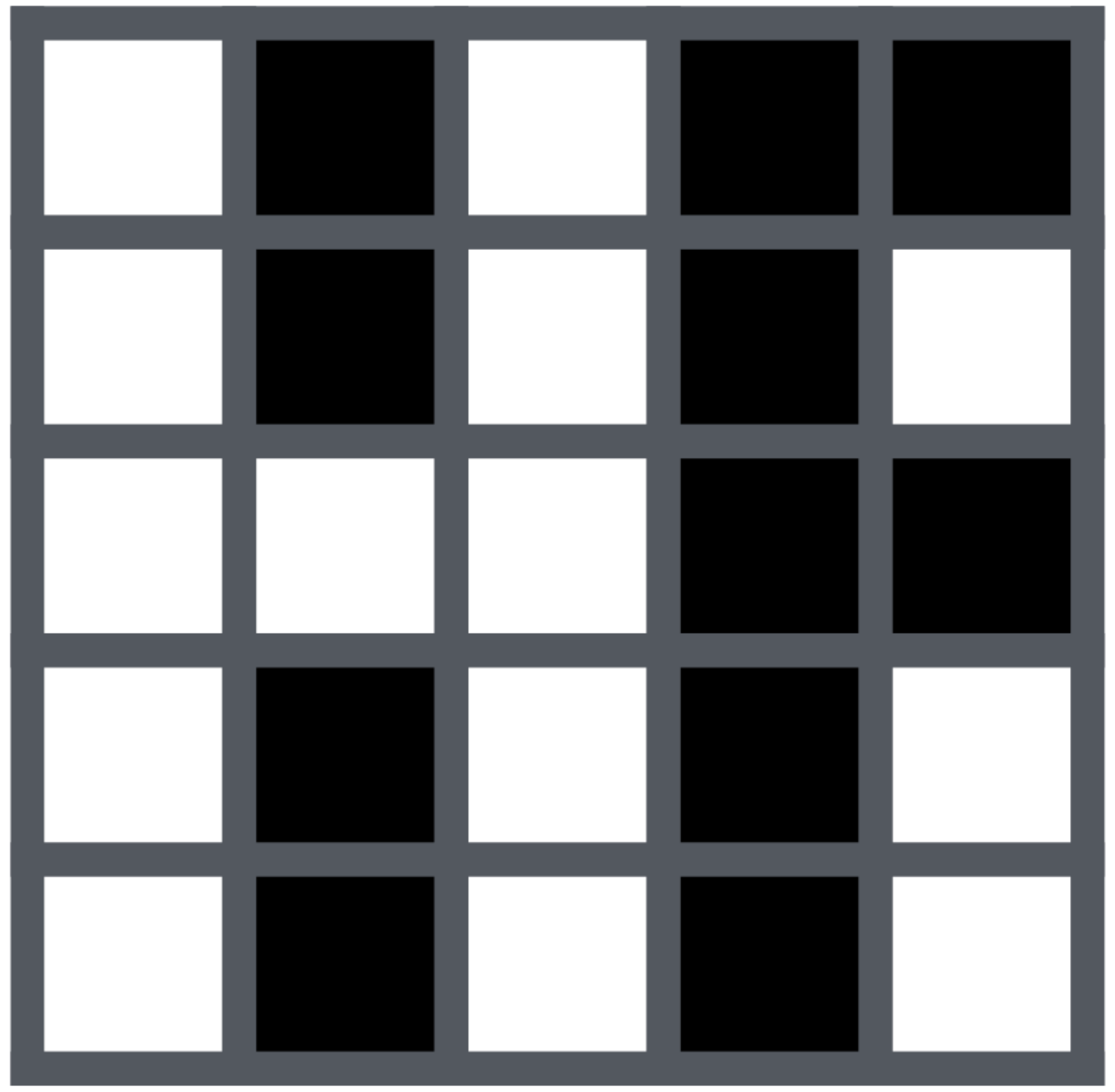
Failure modes of fully-connected neural networks

Convolutions

Pooling

CNNs across modalities

Images are tensors



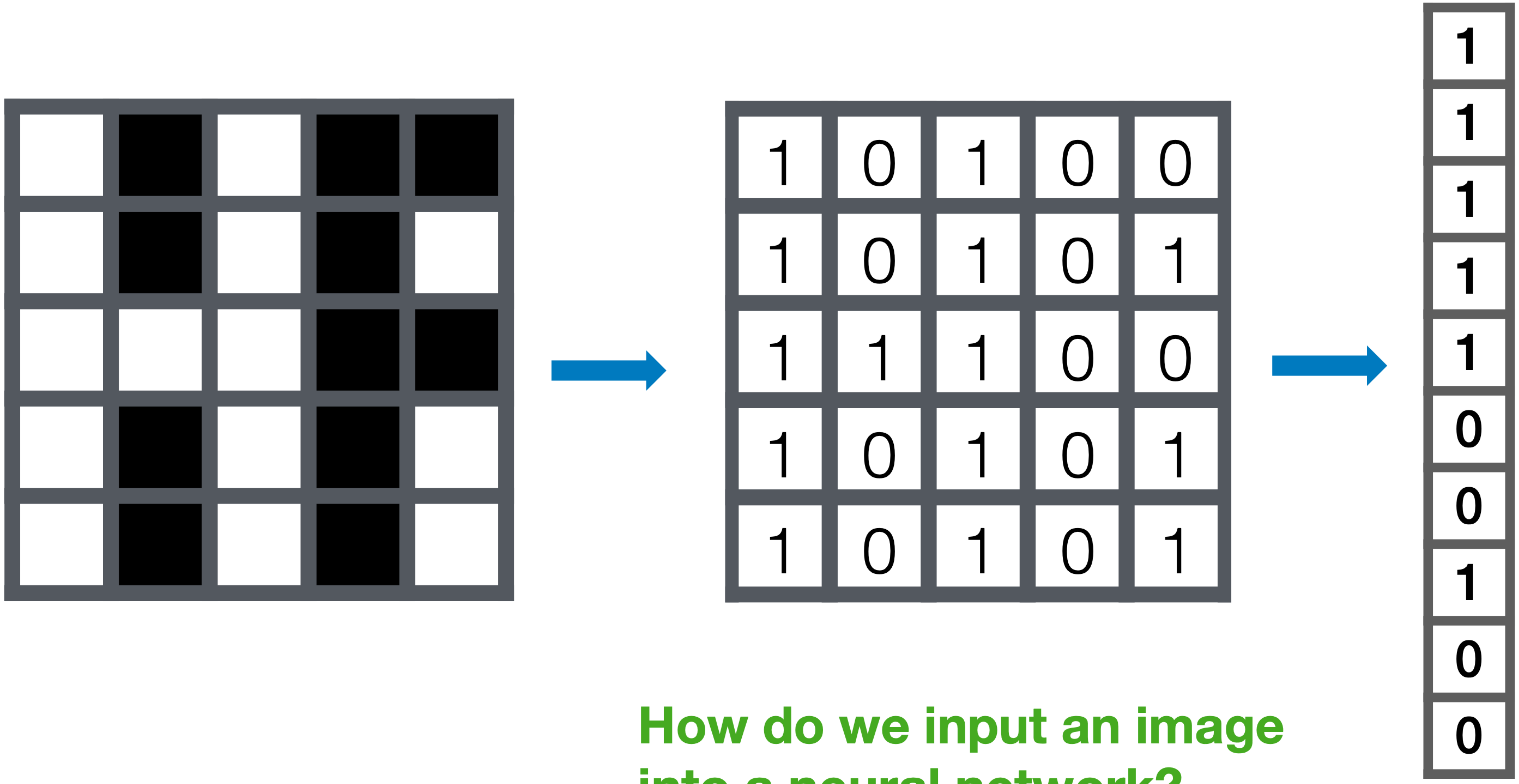
1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

How do we input an image into a neural network?

...

Image credit: Tamara Broderick

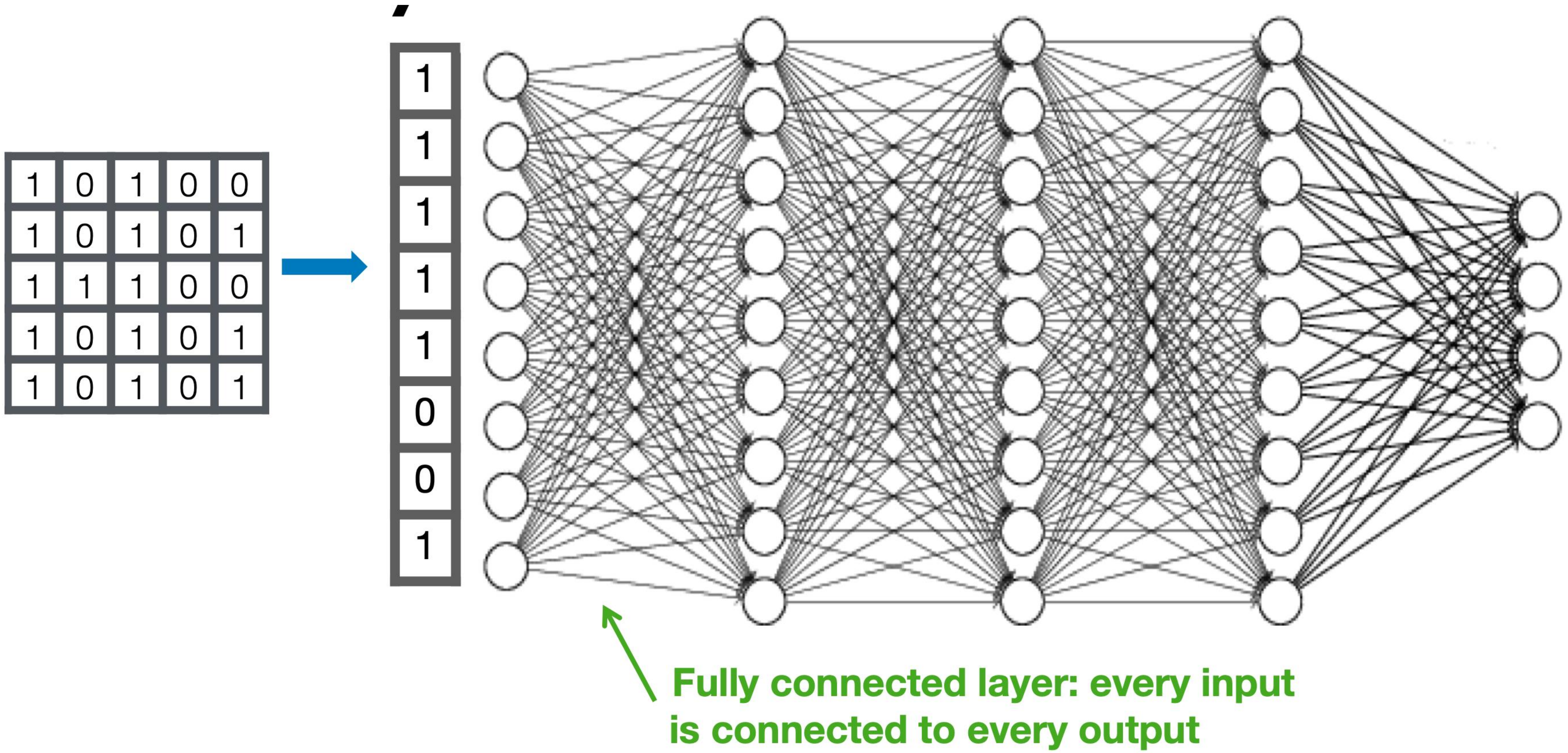
Images in our class so far..



How do we input an image into a neural network?

Idea: make it a vector and use FNN!

Using FNNs on tiny images



Example:

512*512 image

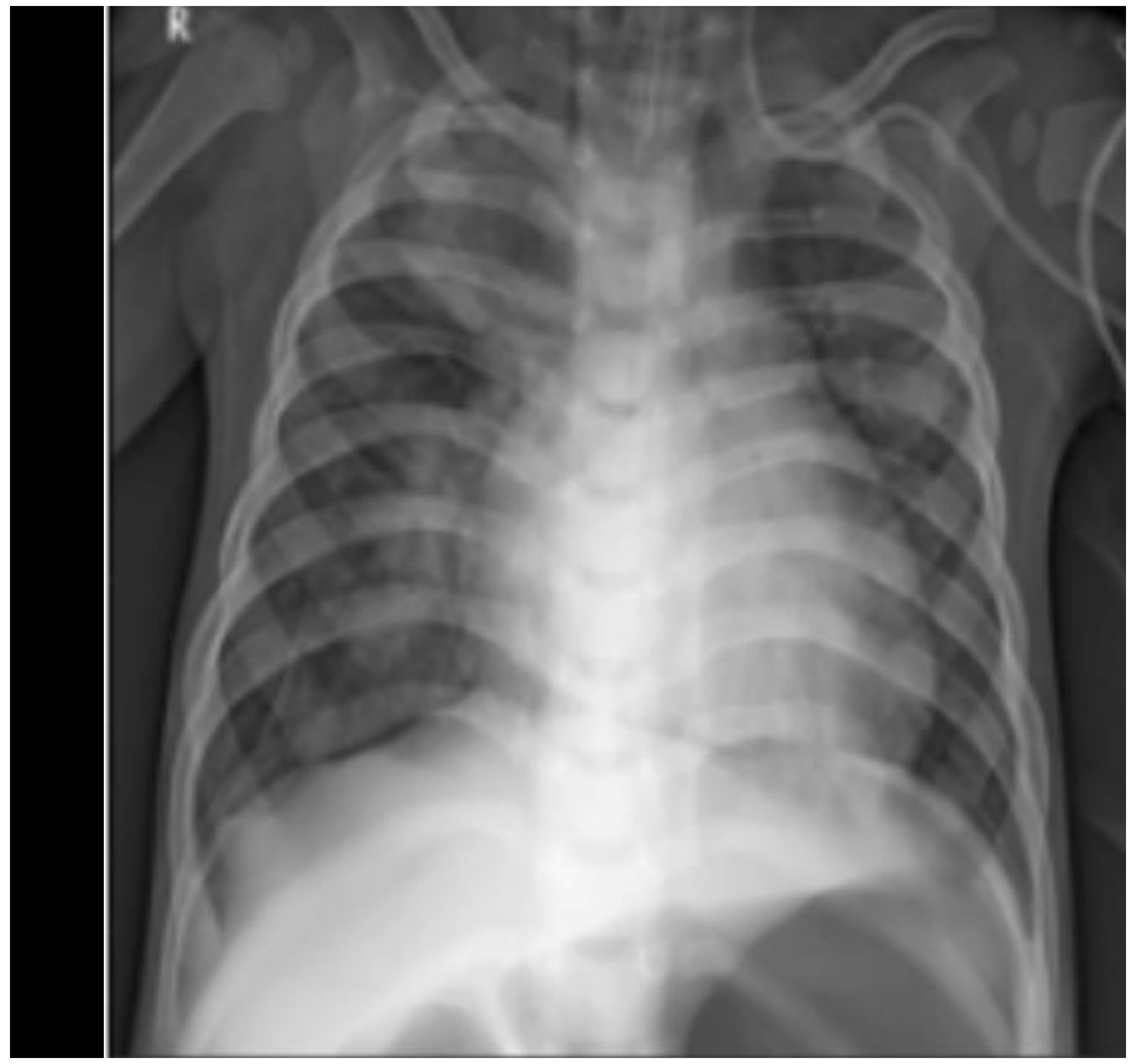
~250K hidden units

62.5B parameters per layer <- Bigger than GPT3…

What's wrong with FNNs?

$\downarrow$

$x^0 = [0,1,1, \dots, 0]$



$\downarrow$

$x^1 = [0,0,1, \dots, 0]$

Small padding, very different feature vectors!

Can it learn?

Yes

Will it be easy?

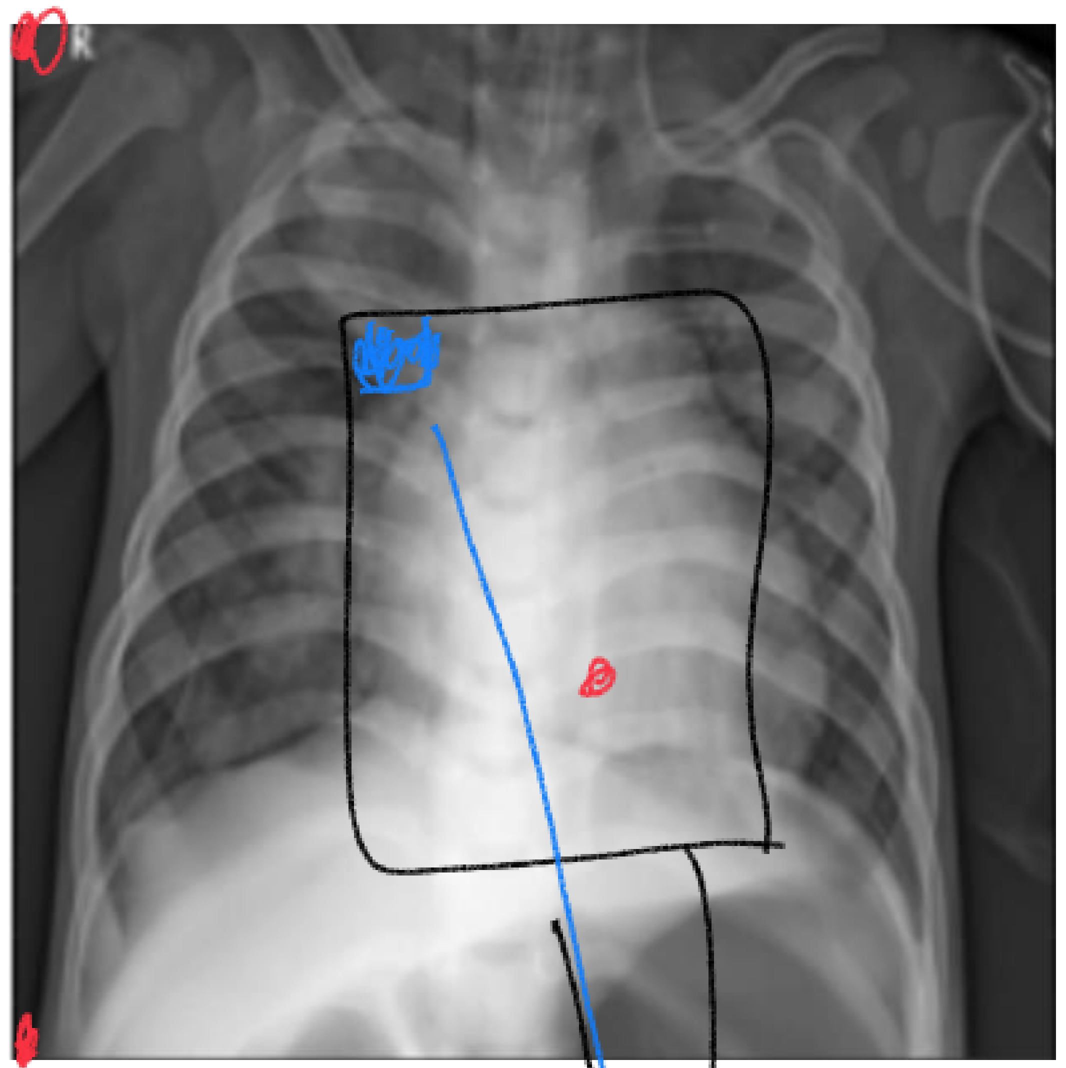
No.

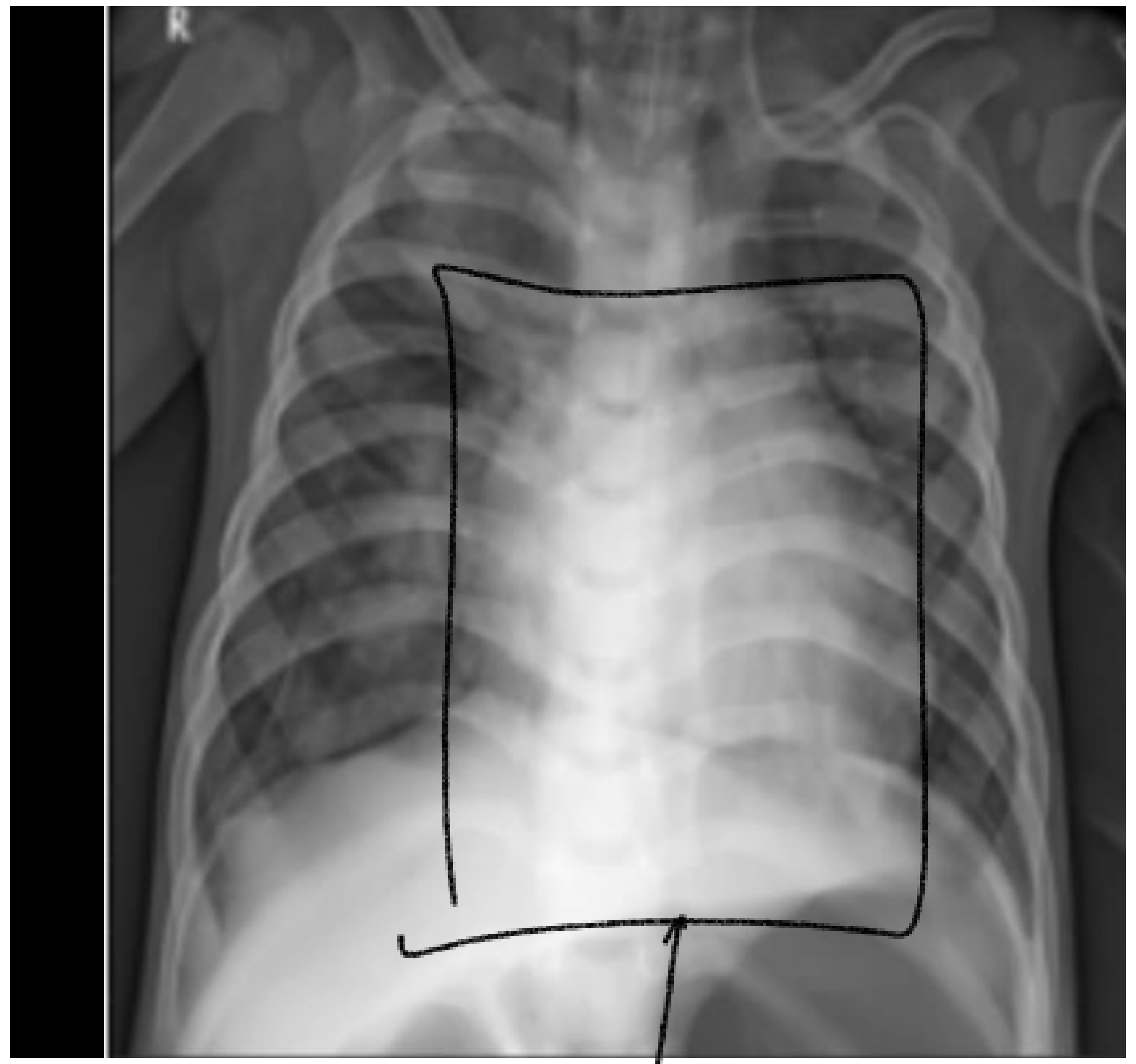
→ Learn invariance is hard.

→ Memory prohibitive

Conceptual role of Hypothesis class:
Choose your Mountain range

What do we know?





Locality

Translations don't matter!

Desired properties for a good Hypothesis Class

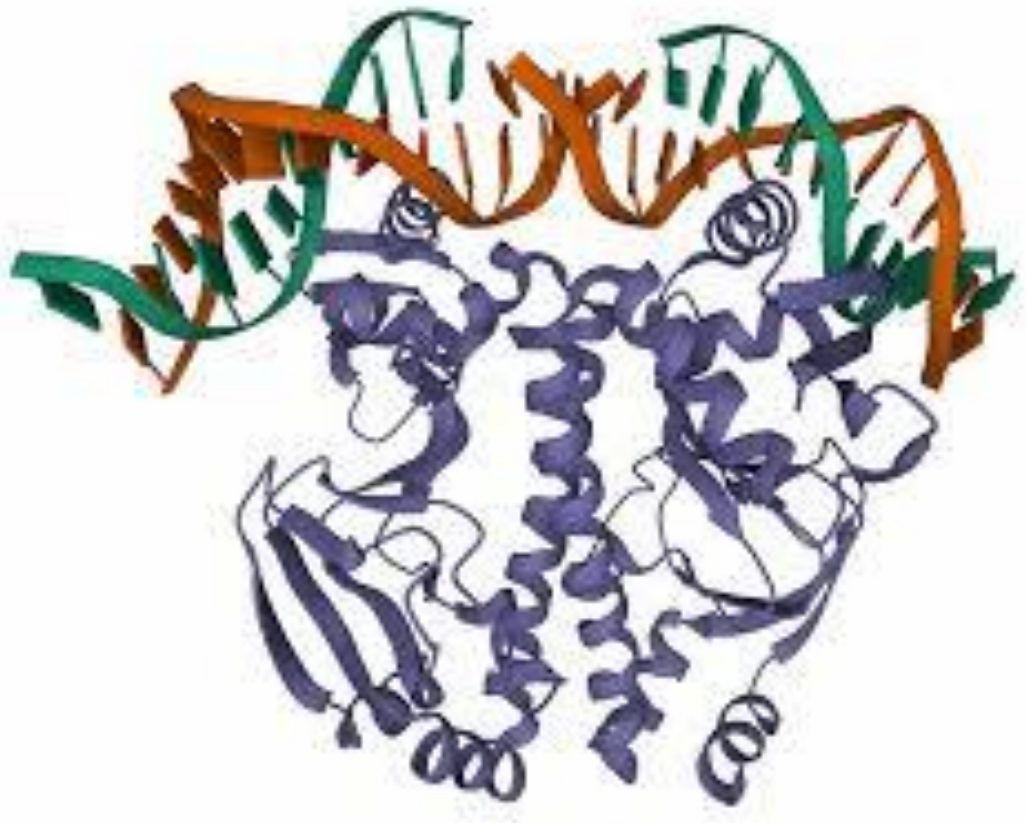
Capture spatial dependencies: Pixel positions and locality matter!

Handle Translations / Nuissance variations: Objects of interest can appear anywhere

Scale: Allow efficient computation for large inputs

Images/ Volumes

Text



Graphs

Agenda

Recap

Failure modes of fully-connected neural networks

Convolutions: Capturing locality and positional-equivariance

Pooling

CNNs across modalities

Convolution: Definition

$$\mathrm{Conv}(x, \theta) = (x * \theta)[i] = \sum_{j=0}^{k} x[i-j]\,\theta[j]$$

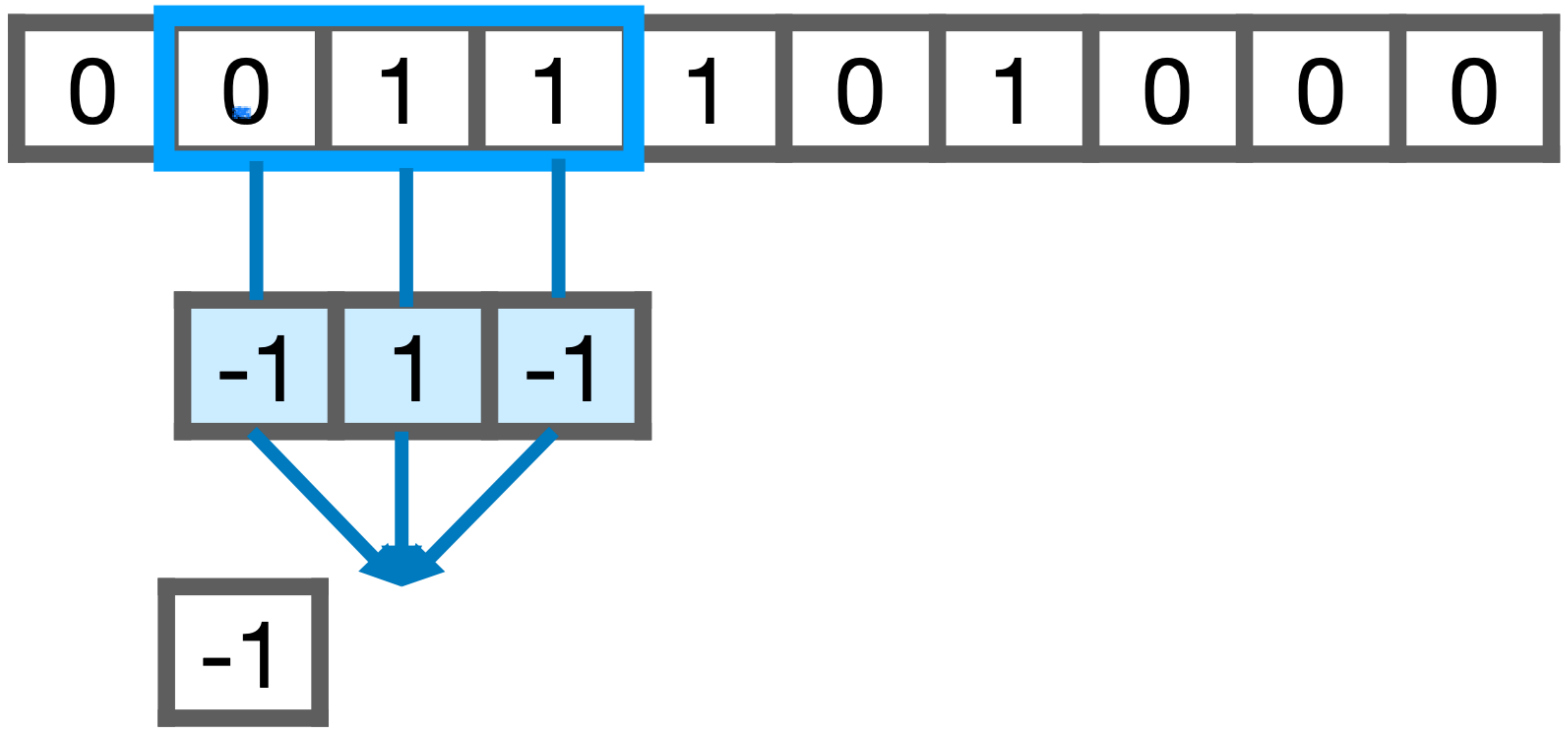
$$x \in \mathbb{R}^N$$

$$\theta \in \mathbb{R}^k$$

Sliding window dot product!

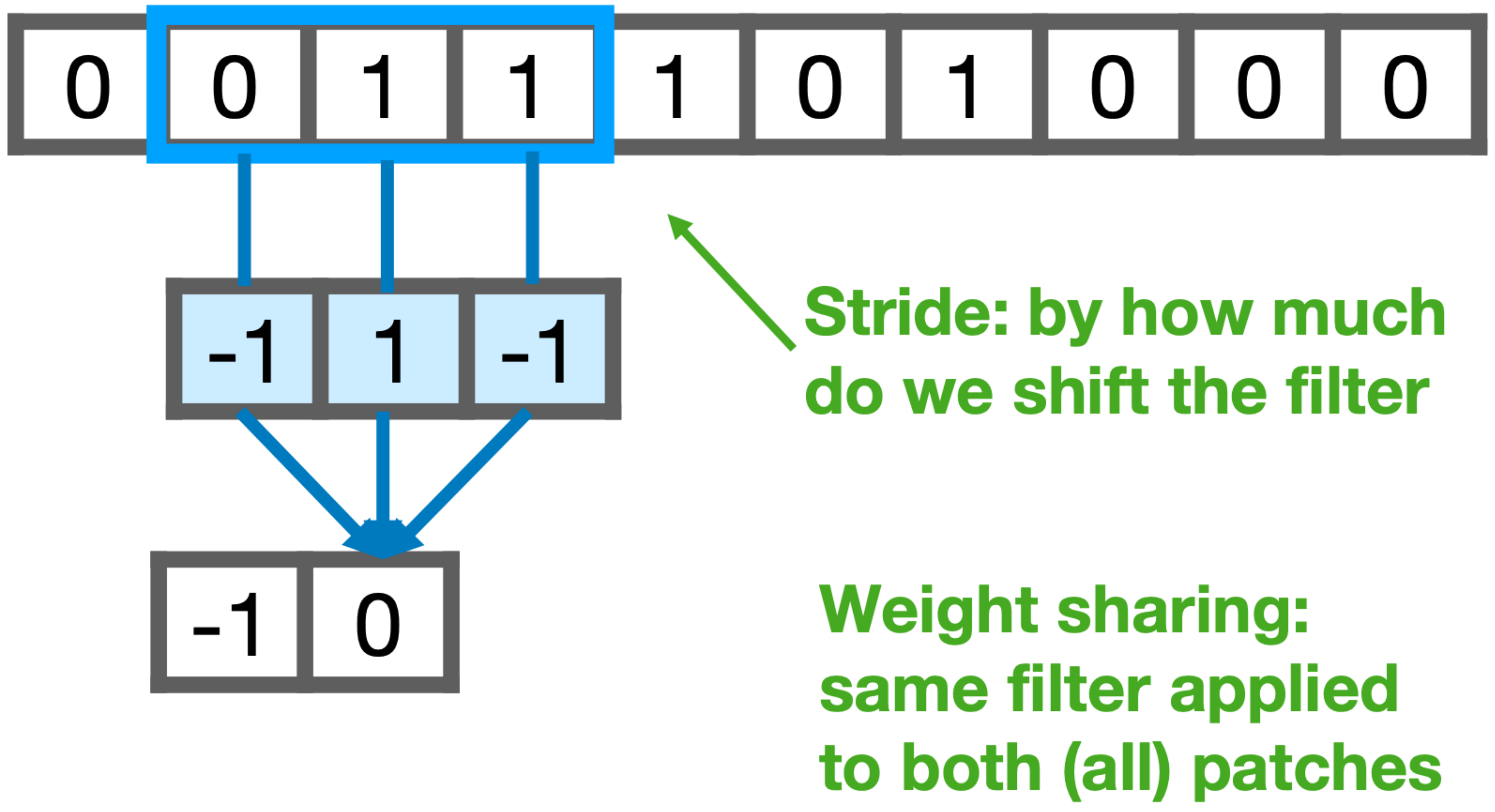
Convolutional Layer: 1D example

- 1D image
- Filter
- After convolution



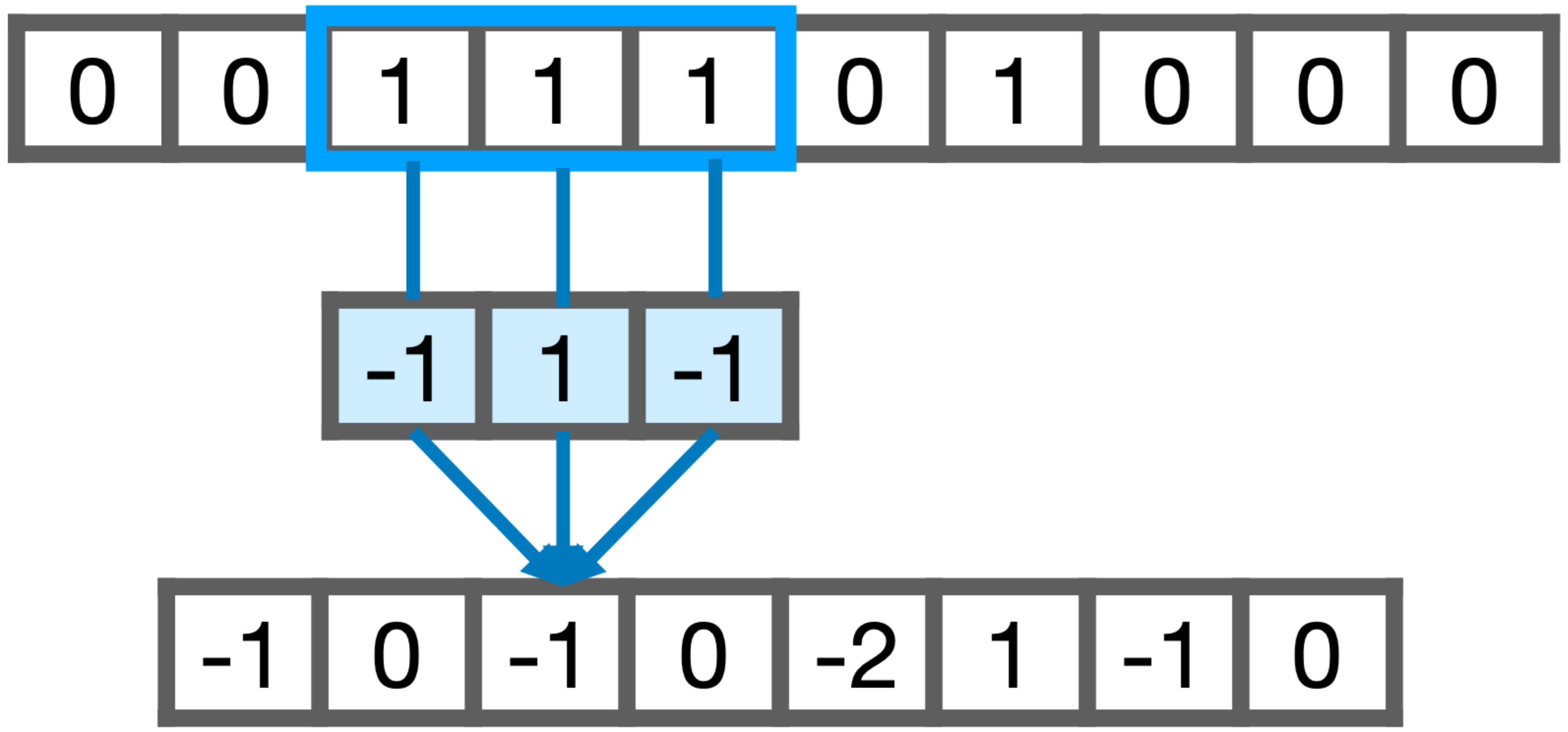
Convolutional Layer: 1D example

- 1D image
- Filter
- After convolution



Convolutional Layer: 1D example

- 1D image
- Filter
- After convolution



Big advantage: due to weight sharing, needs much fewer weights than a fully connected network

Translation equivariance!

Convolutional Layer: 1D example

- 1D image

0	0	1	1	1	0	1	0	0	0

- Filter

-1	1	-1

- After convolution

-1	0	-1	0	-2	1	-1	0

- After ReLu

Padding

- 1D image

0	0	1	1	1	0	1	0	0	0

- Filter

-1	1	-1

- After convolution

-1	0	-1	0	-2	1	-1	0

- After ReLu

0	0	0	0	0	1	0	0

Output is smaller! (why?)

Remedy: pad input with zeros

Padding

- 1D image

0	0	0	1	1	1	0	1	0	0	0	0

- Filter

-1	1	-1

- After convolution

0	-1	0	-1	0	-2	1	-1	0	0

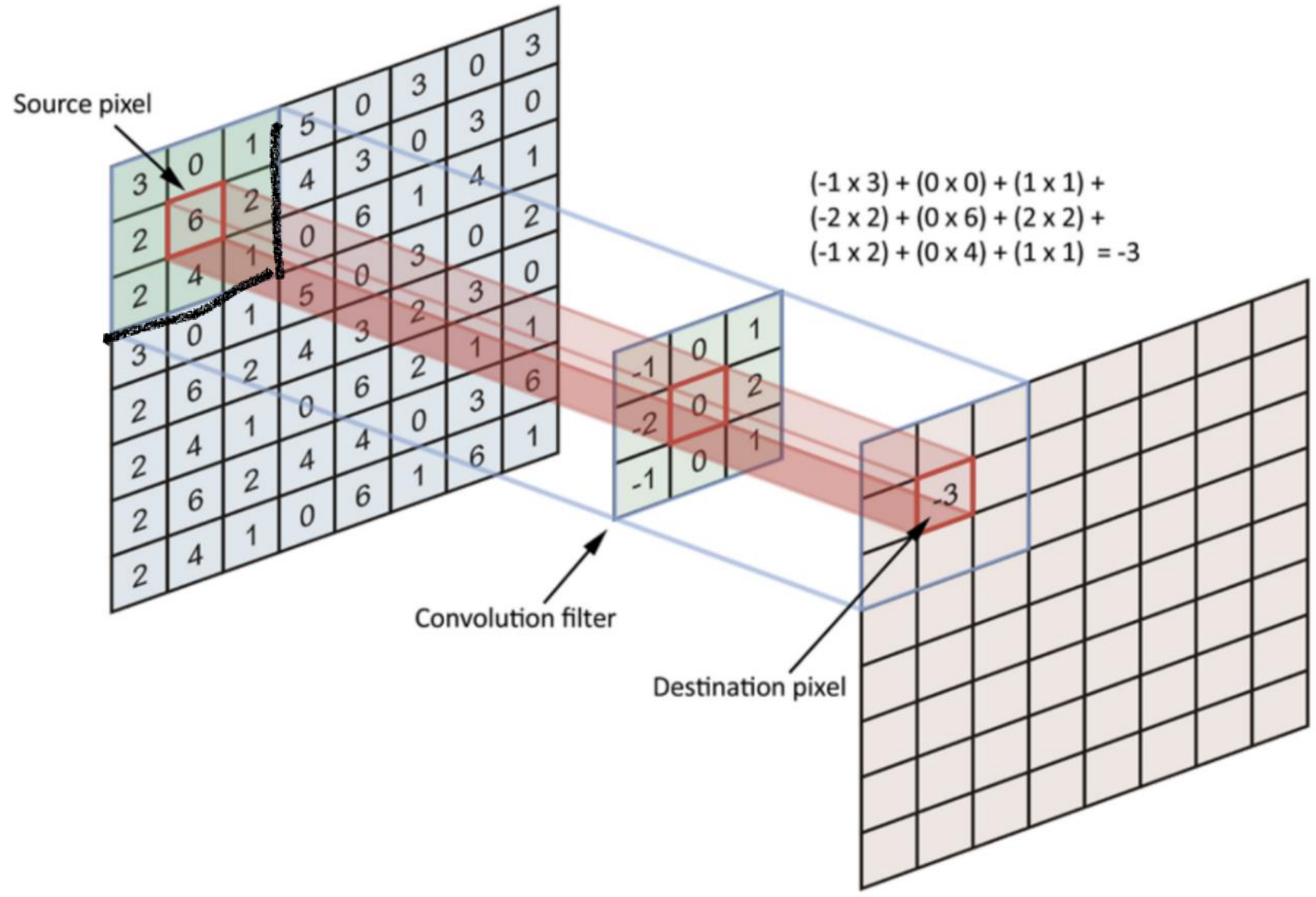
- After ReLu

0	0	0	0	0	0	1	0	0	0

Output is smaller! (why?)

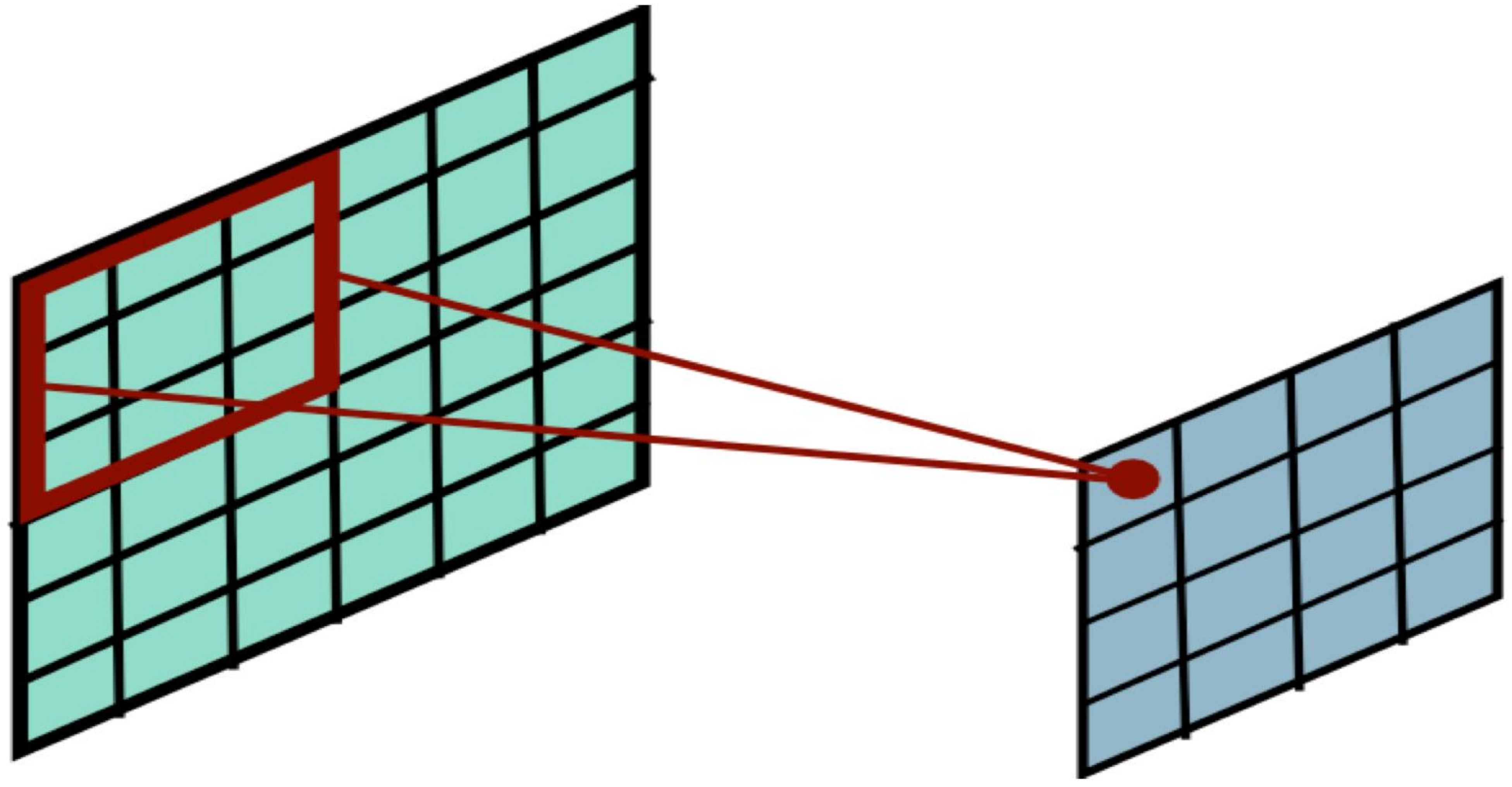
Remedy: pad input with zeros

2D Convolutions



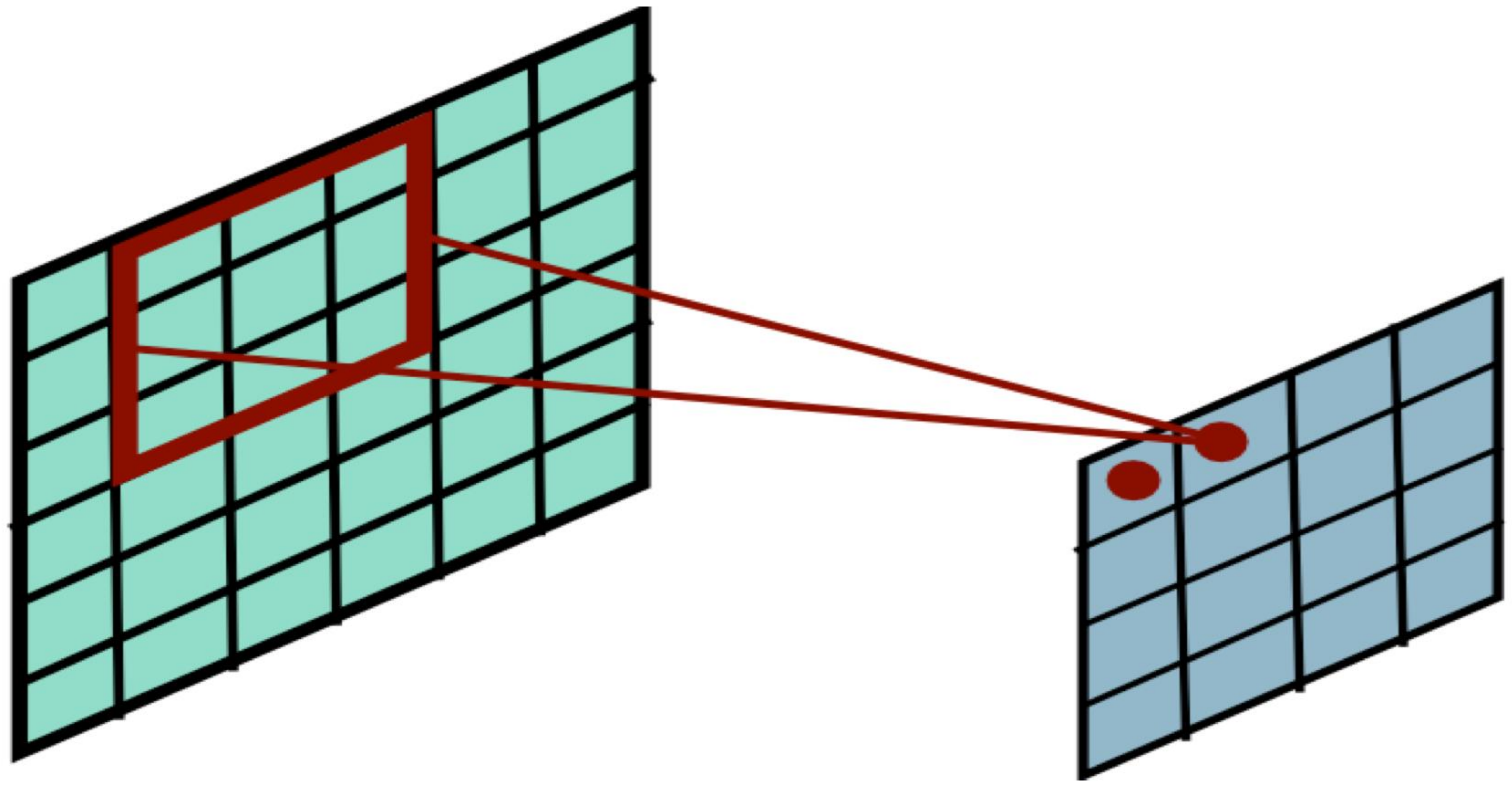
The convolution operation.

Convolutional Layer

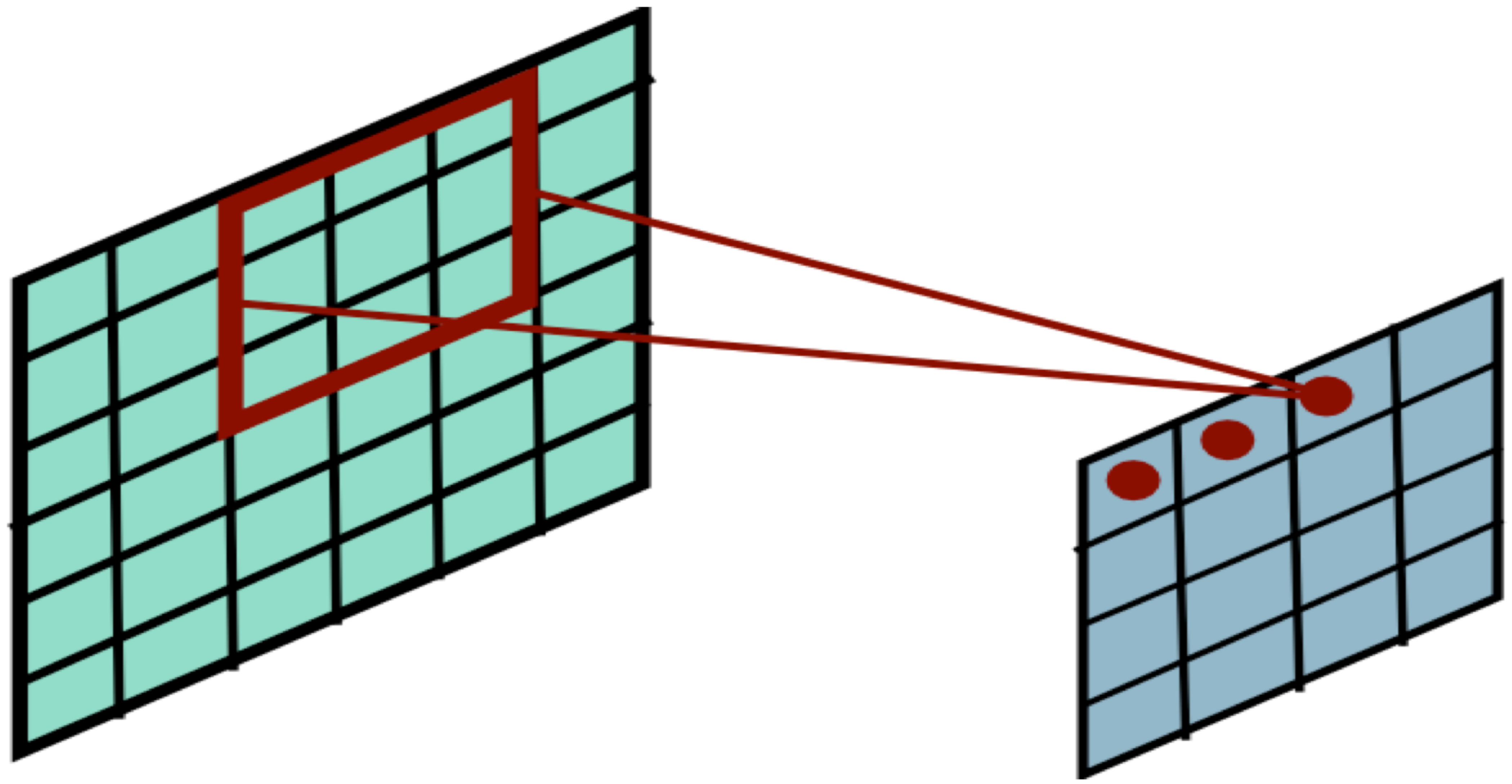


} activation
map.

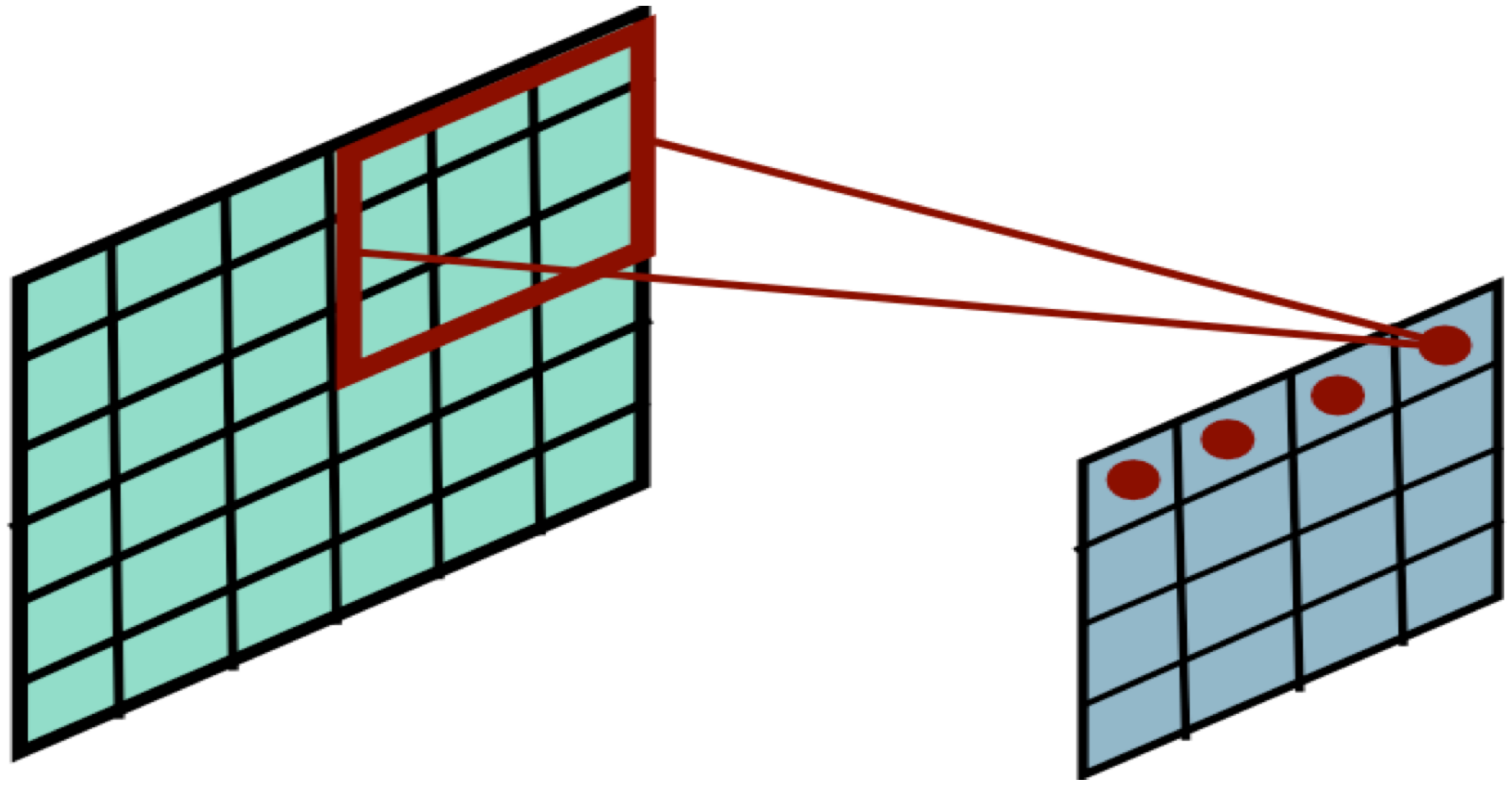
Convolutional Layer



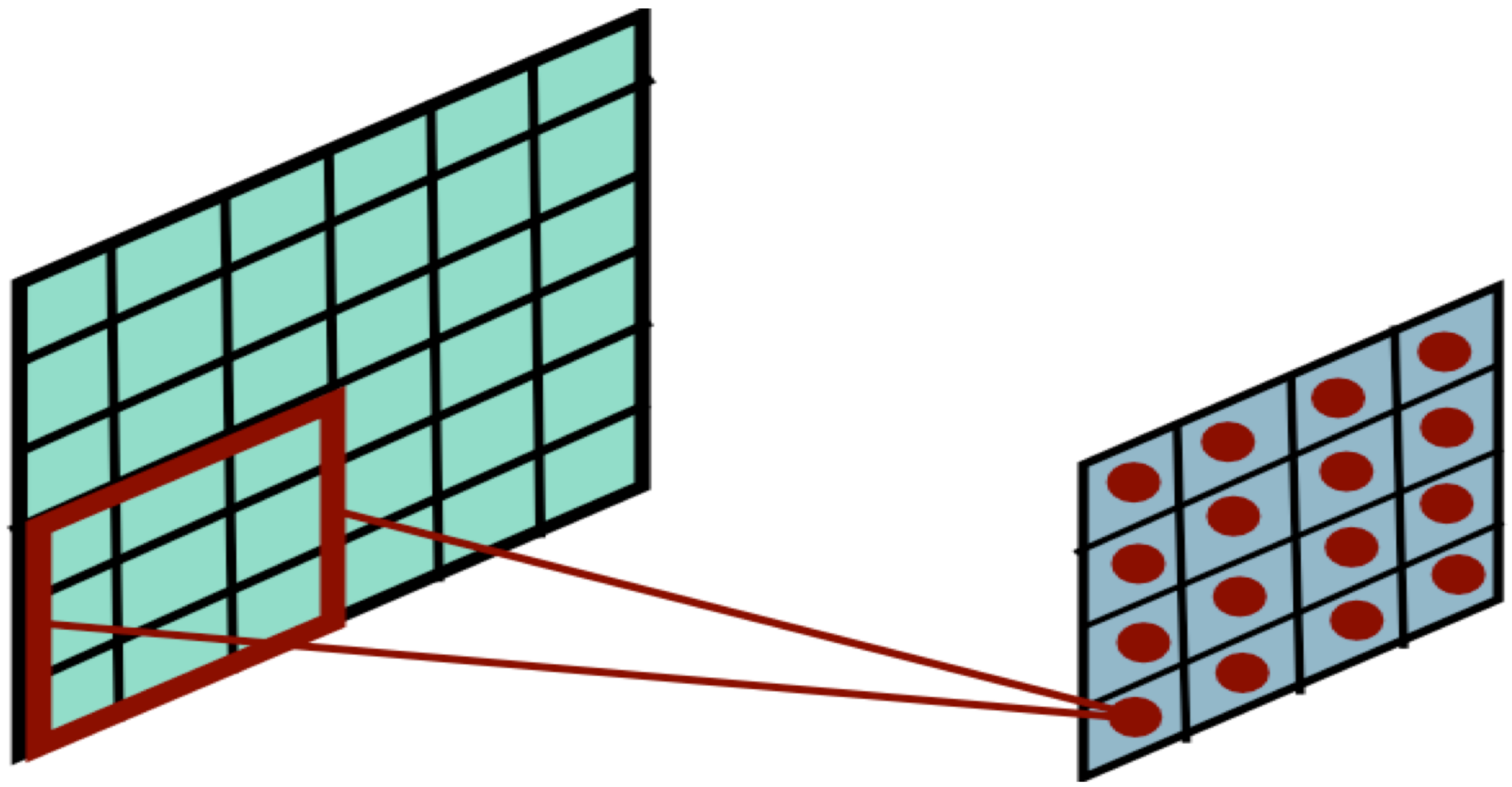
Convolutional Layer



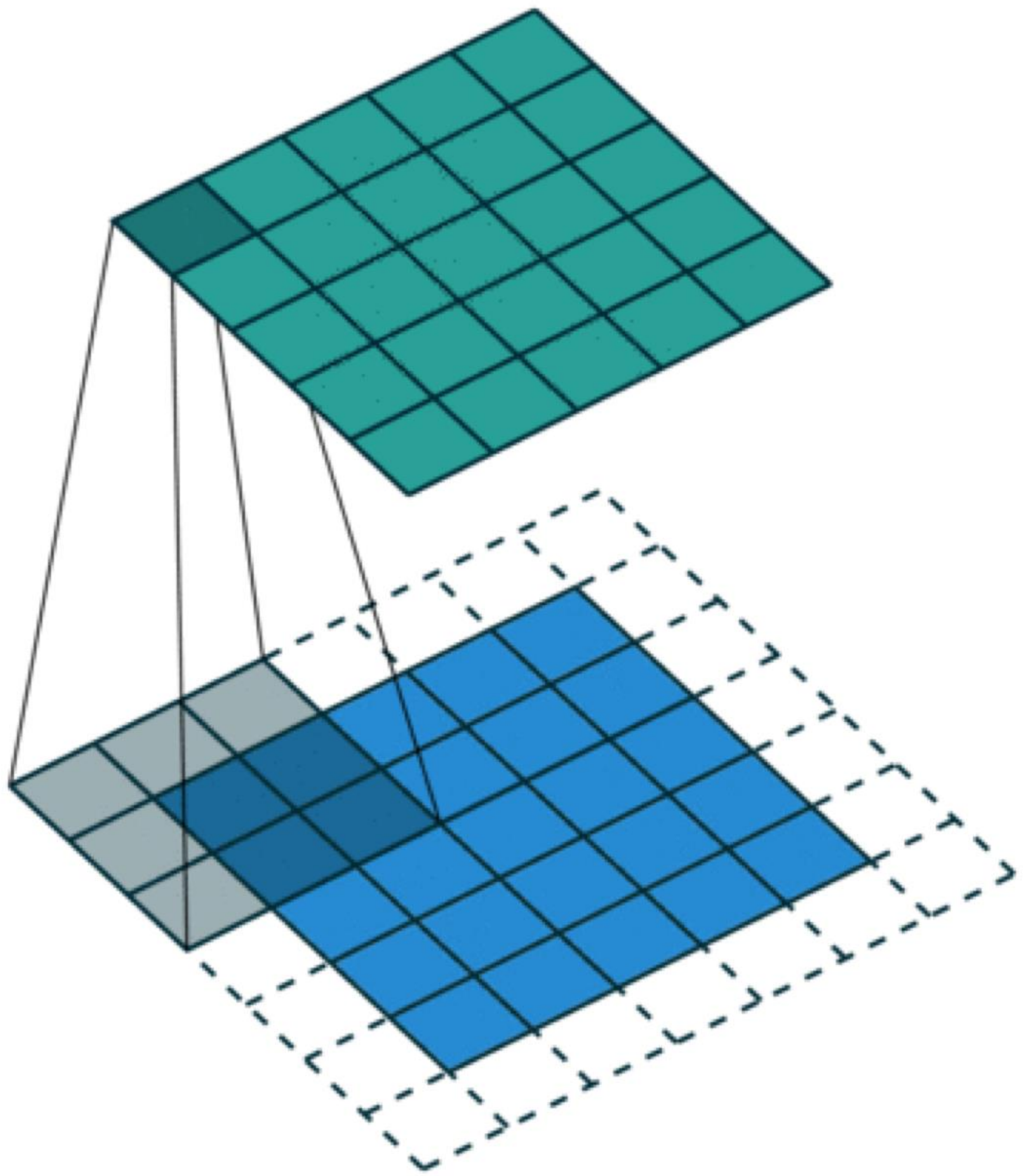
Convolutional Layer



Convolutional Layer

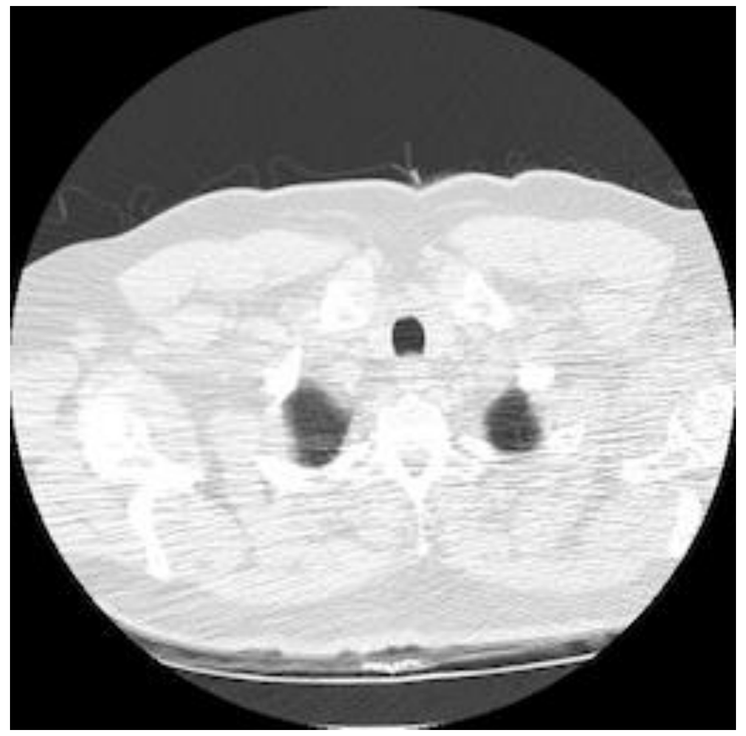


Convolution with Padding

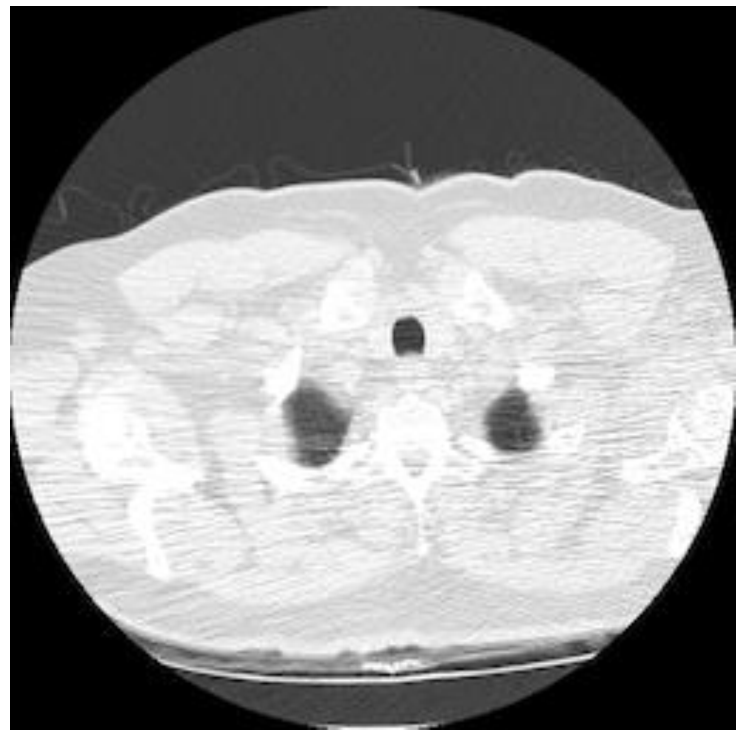


source: Theano, deeplearning.net

Question: How would you apply this idea to a CT-scan?

Question: How would you apply this idea to a CT-scan?

3D Convolutions

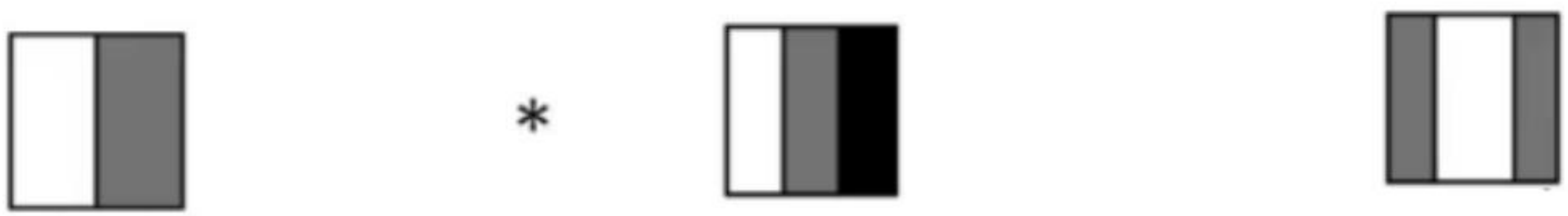
Examples of Convolutions

filter

mask

$$
\begin{array}{|c|c|c|c|c|c|}
\hline
10 & 10 & 10 & 0 & 0 & 0 \\ \hline
10 & 10 & 10 & 0 & 0 & 0 \\ \hline
10 & 10 & 10 & 0 & 0 & 0 \\ \hline
10 & 10 & 10 & 0 & 0 & 0 \\ \hline
10 & 10 & 10 & 0 & 0 & 0 \\ \hline
10 & 10 & 10 & 0 & 0 & 0 \\ \hline
\end{array}
\;*\;
\begin{array}{|c|c|c|}
\hline
1 & 0 & -1 \\ \hline
1 & 0 & -1 \\ \hline
1 & 0 & -1 \\ \hline
\end{array}
\;=\;
\begin{array}{|c|c|c|c|}
\hline
0 & 30 & 30 & 0 \\ \hline
0 & 30 & 30 & 0 \\ \hline
0 & 30 & 30 & 0 \\ \hline
0 & 30 & 30 & 0 \\ \hline
\end{array}
$$

input

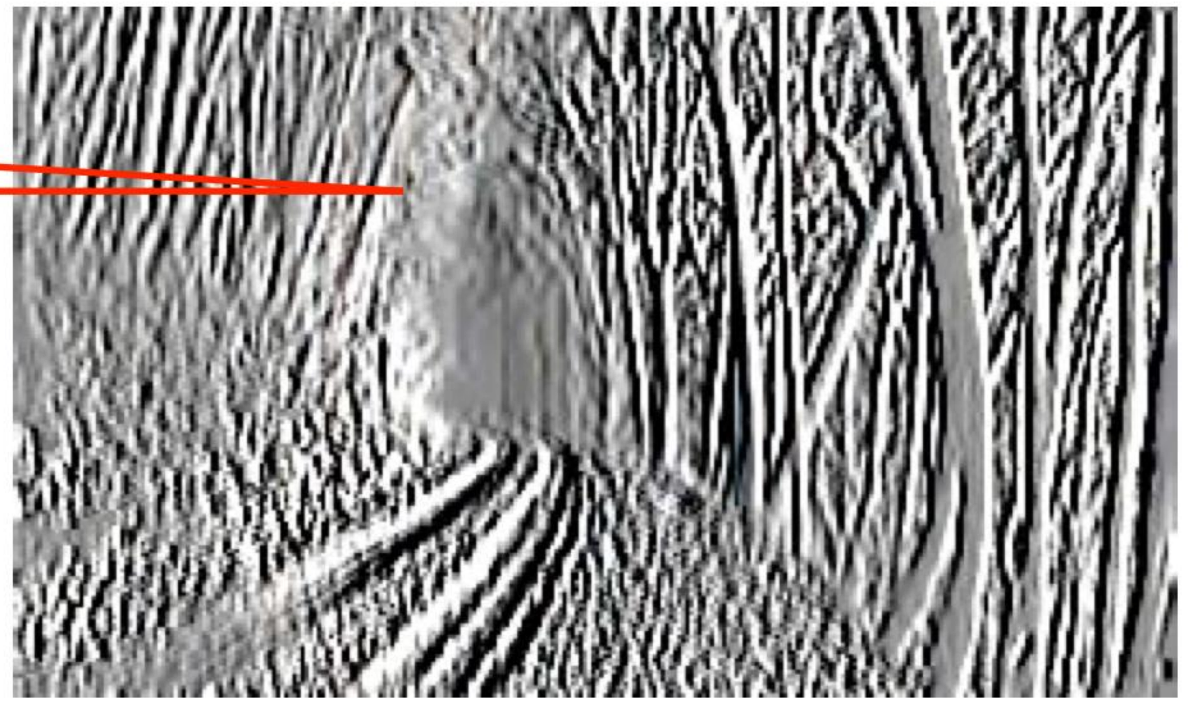


Examples of Convolutions

$$* \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} =$$

mask

(vector, NOT a matrix!)



Examples of Convolutions

Kernel

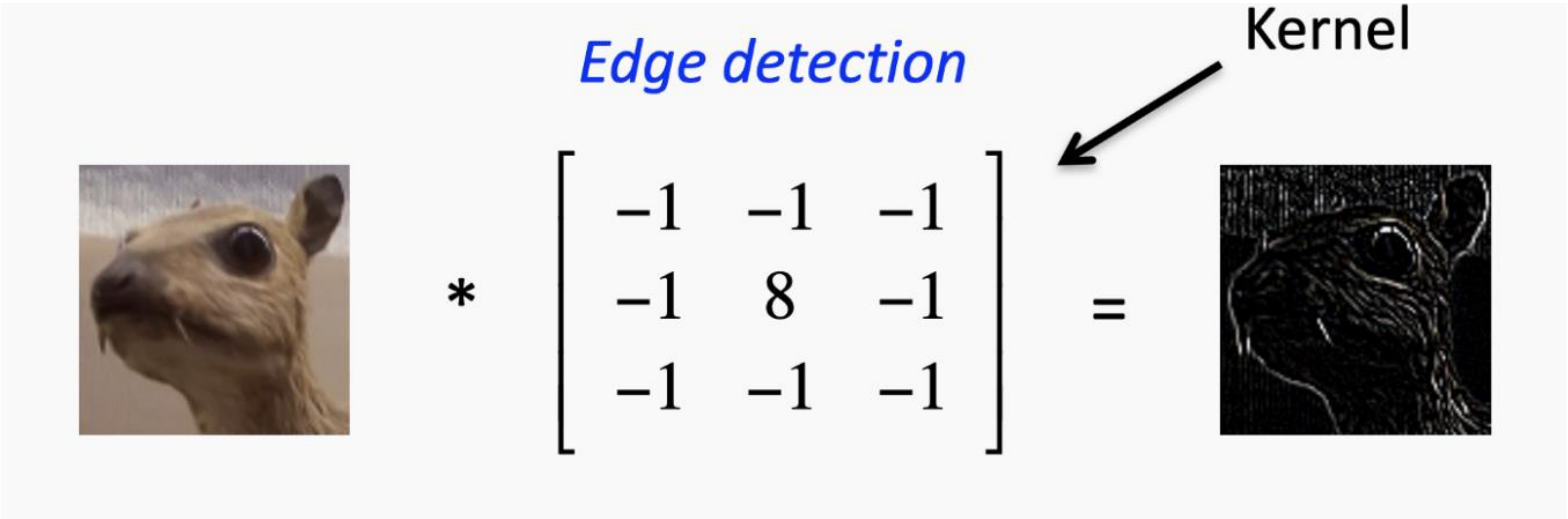
Edge detection

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

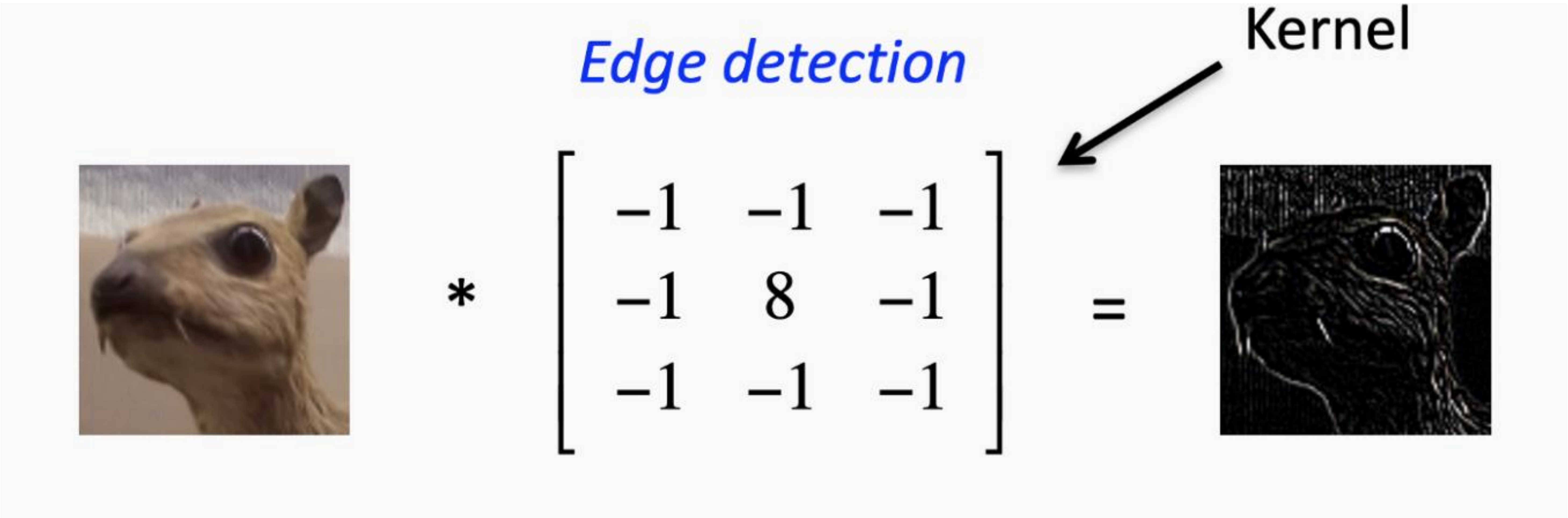
Sharpen

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

Question: How can we make convolutions more expressive?



Question: How can we make convolutions more expressive?

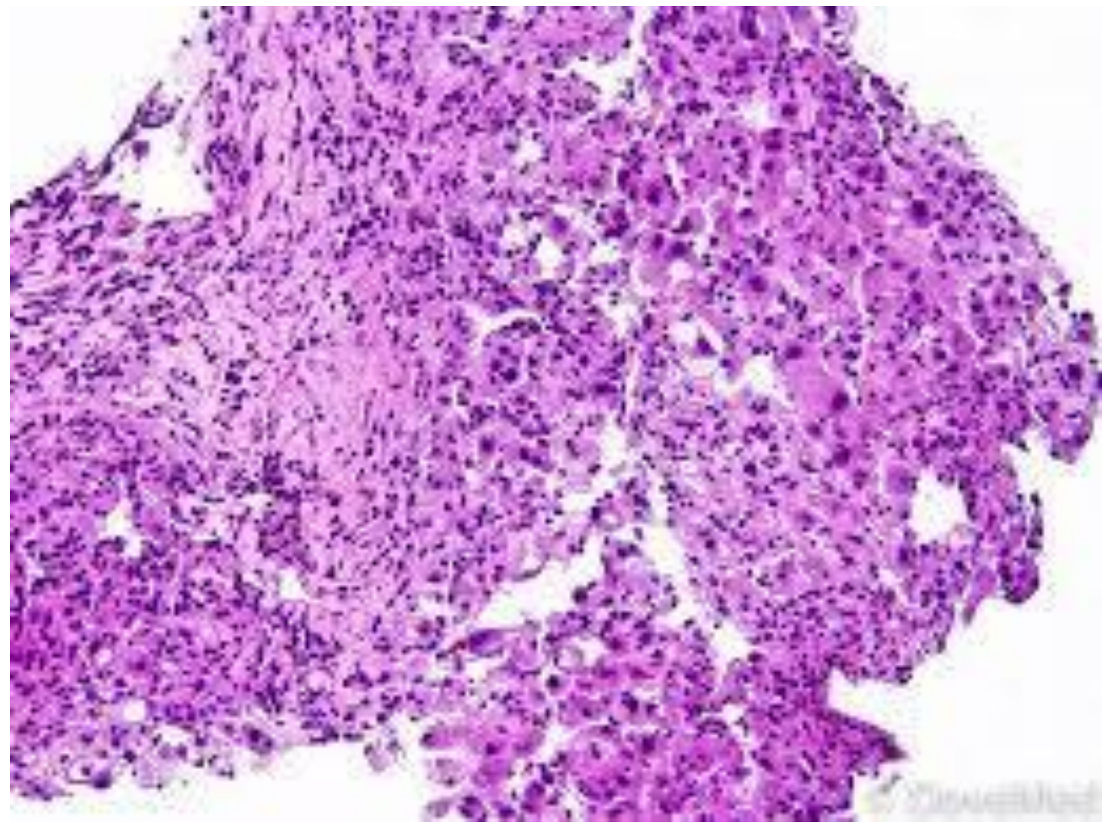


Width: Many kernels in parallel
Depth: Composing kernels

Motivating example: linear in a different basis

$\mathbb{R}^{D_1}$

$\mathbb{R}^{D_2}$



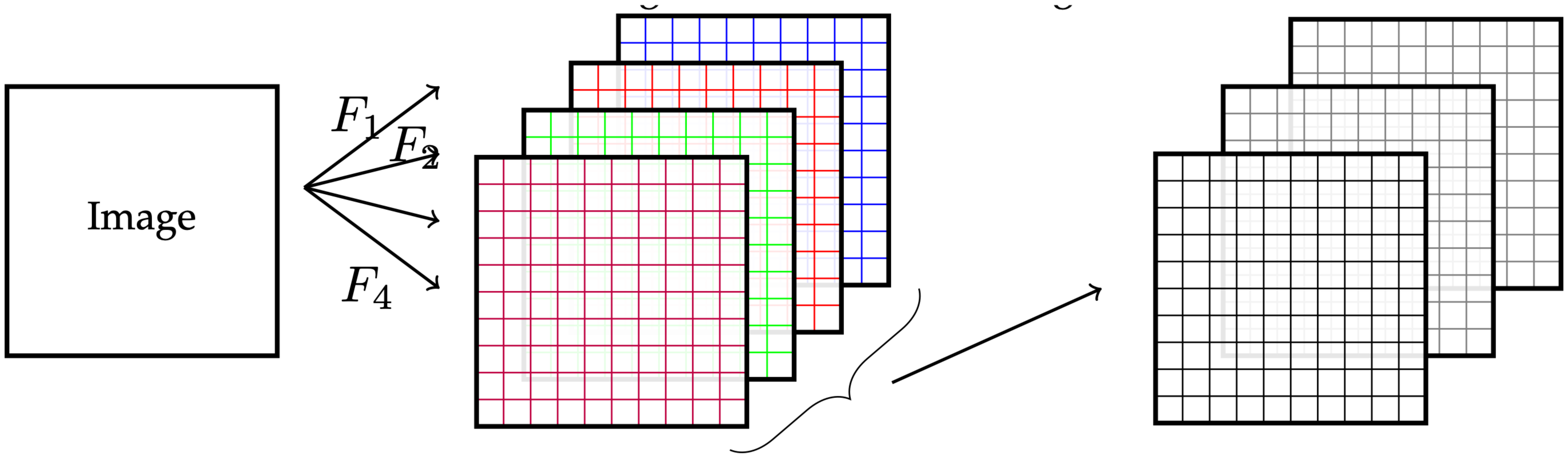
$\longrightarrow z \longrightarrow$ h $\longrightarrow$ response

Linear in z!

$$\sigma(\theta z^T) = p$$

$$\left[\begin{array}{l} \text{PD1 expression} \\ \text{Dead cells} \\ \text{TIL density} \\ \text{Cancer grade} \\ \vdots \end{array}\right] D_2$$

Multiple channels/filters

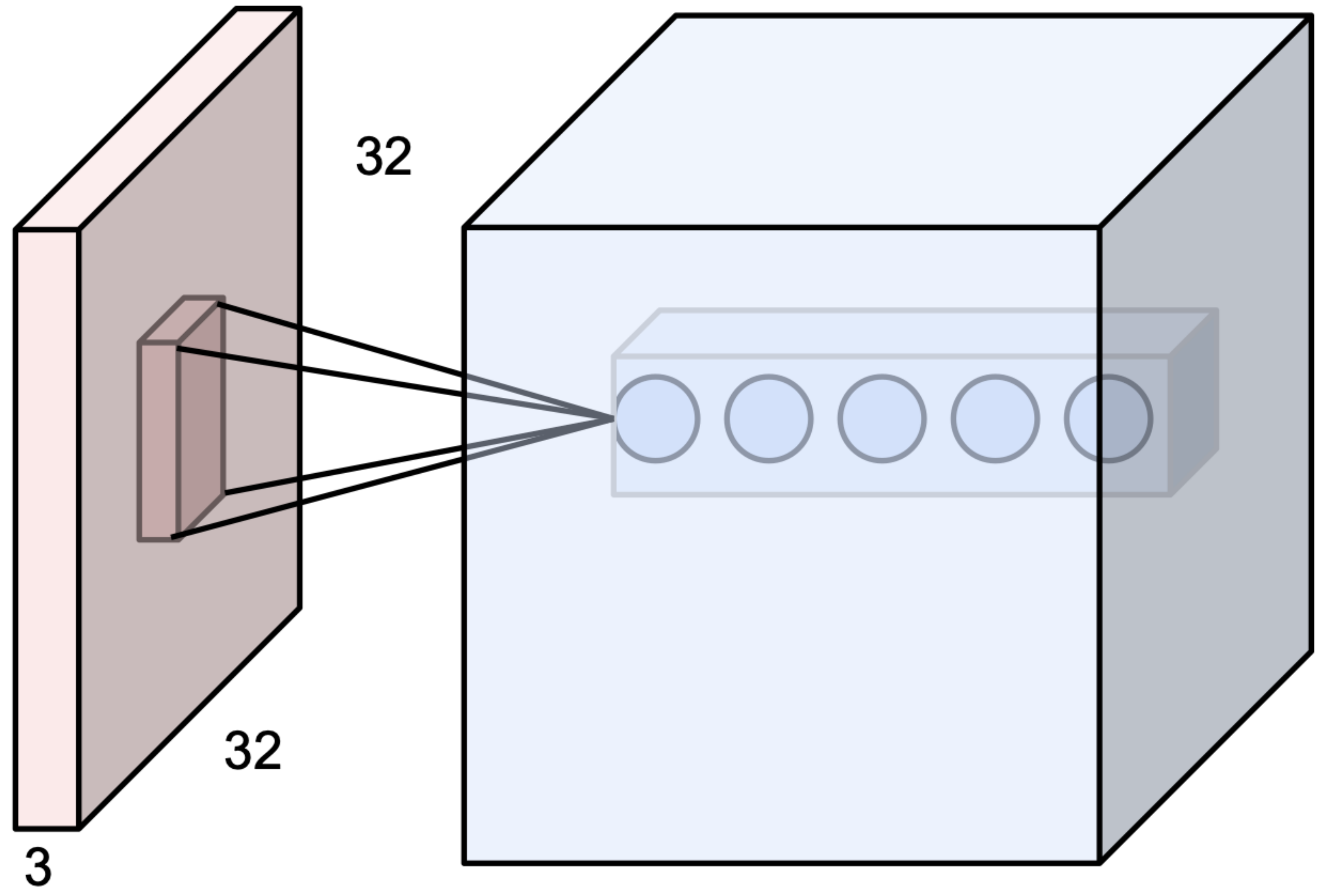


Channels: outputs of convolution

Filter bank: Collection of filters in a layer

$F_1 \to \theta_1 \in \mathbb{R}^{(c, w, h)} \leftarrow$ kernel size

Naming Conventions



Hidden layer of “depth” 5: five neurons all looking at the same patch; five different masks.

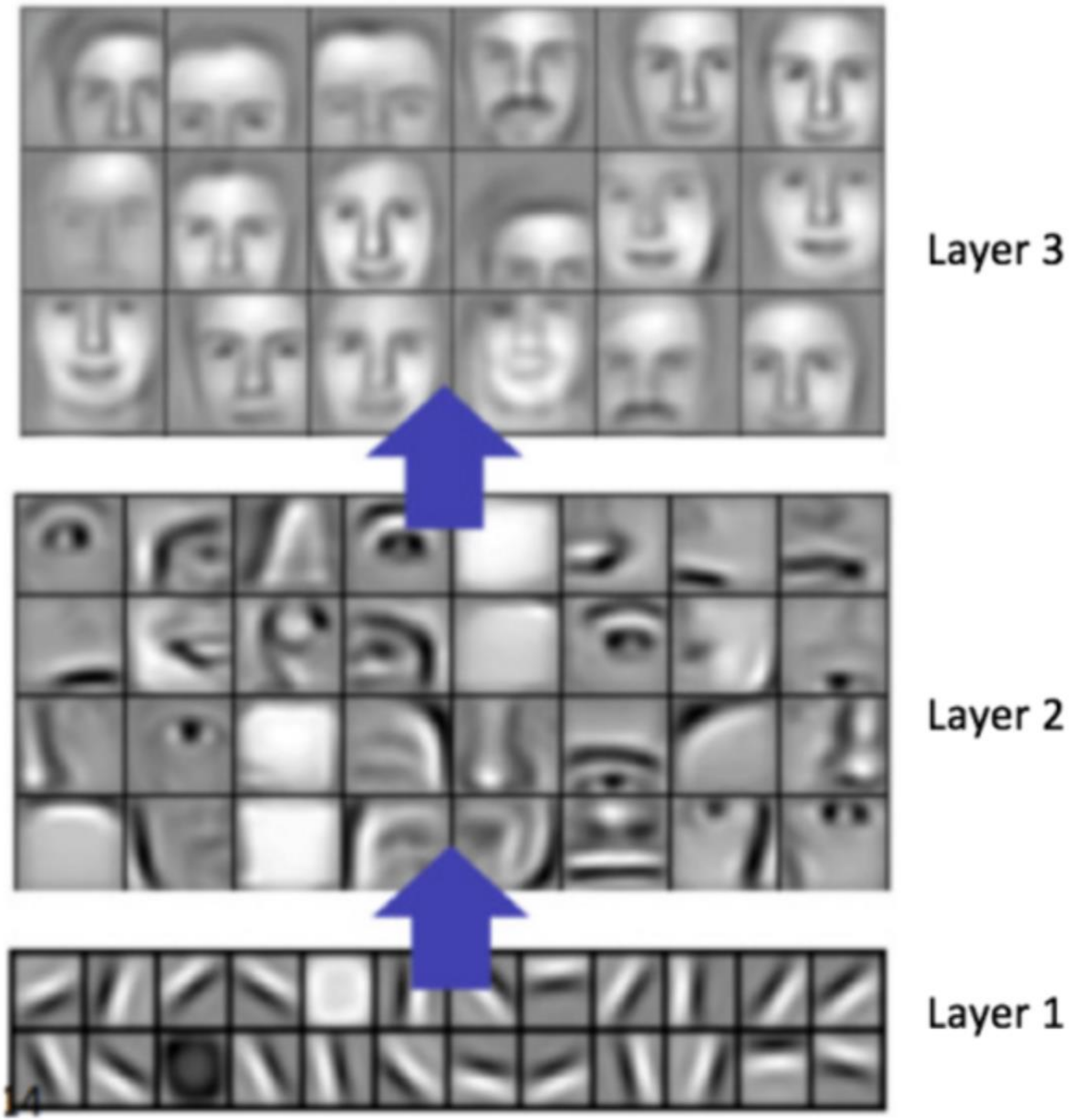
Apply the same 5 masks to each patch. Five neurons per patch.

Conv w param Θ

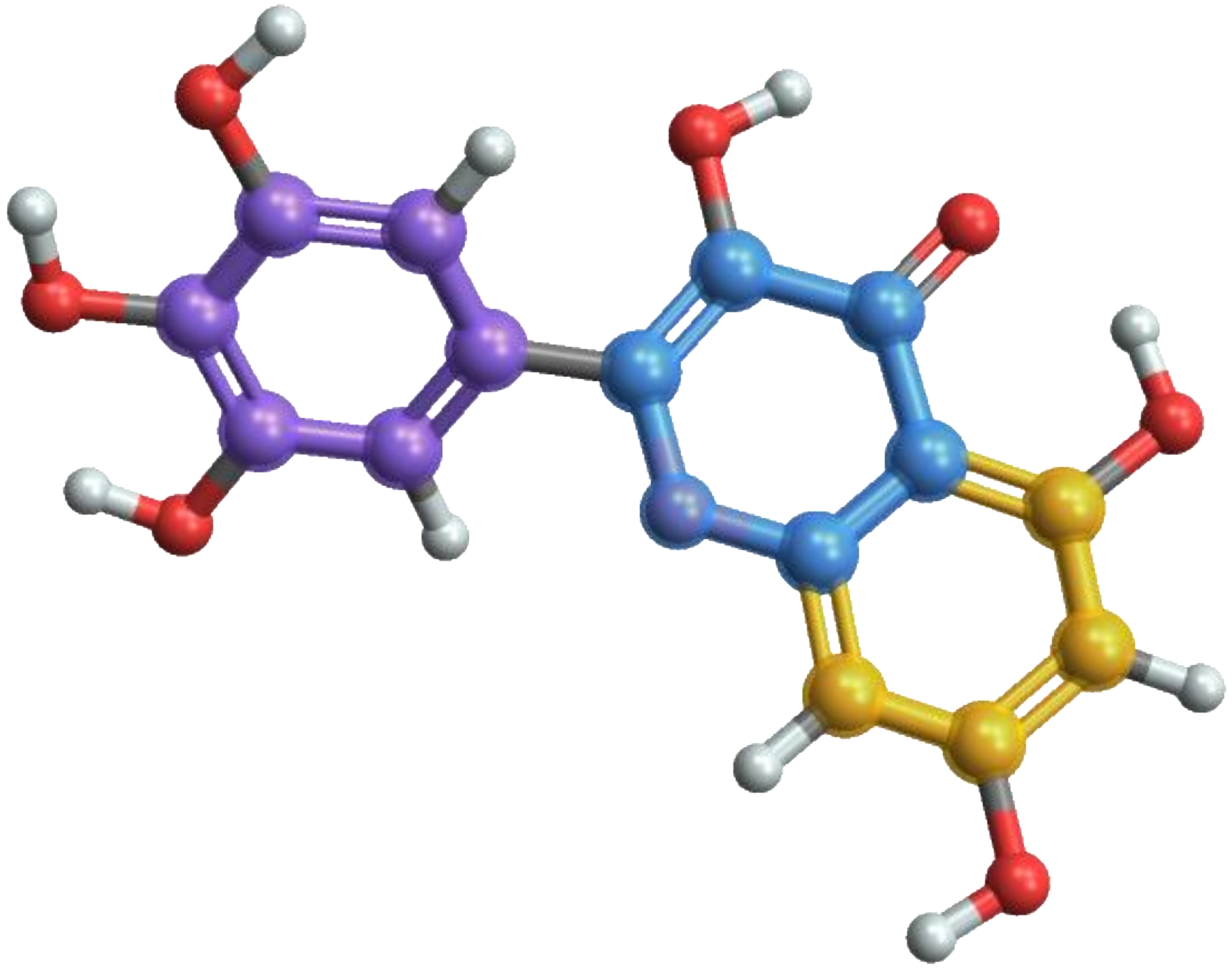
$$\dim(\Theta) \to [C_{in}, w, h, C_{out}]$$

$$\begin{array}{cccc} \uparrow & \uparrow & \uparrow & \uparrow \\ 1 & 3 & 3 & 128 \end{array}$$

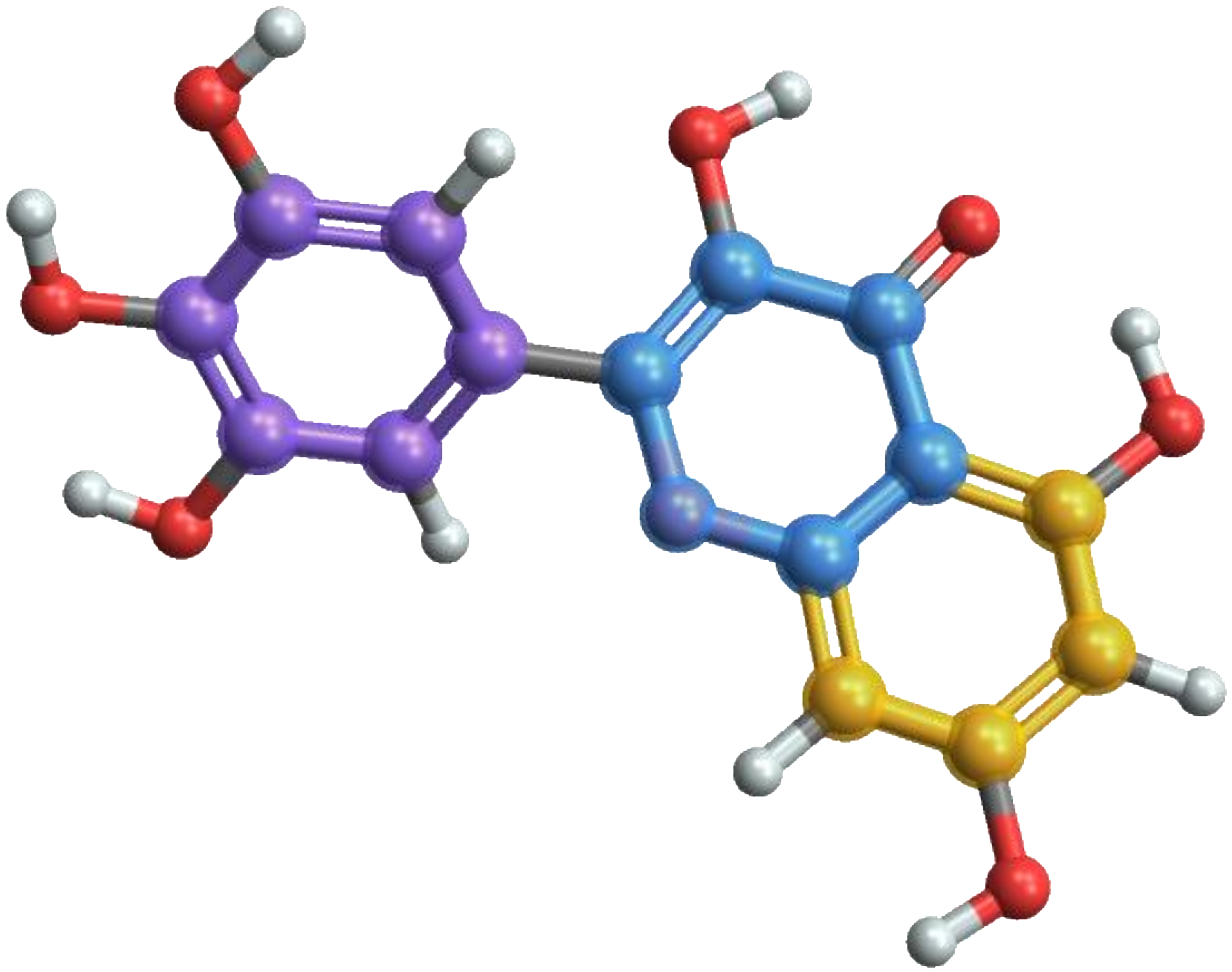
Examples of composition in the wild!



Question: How would you apply this idea to a graph?

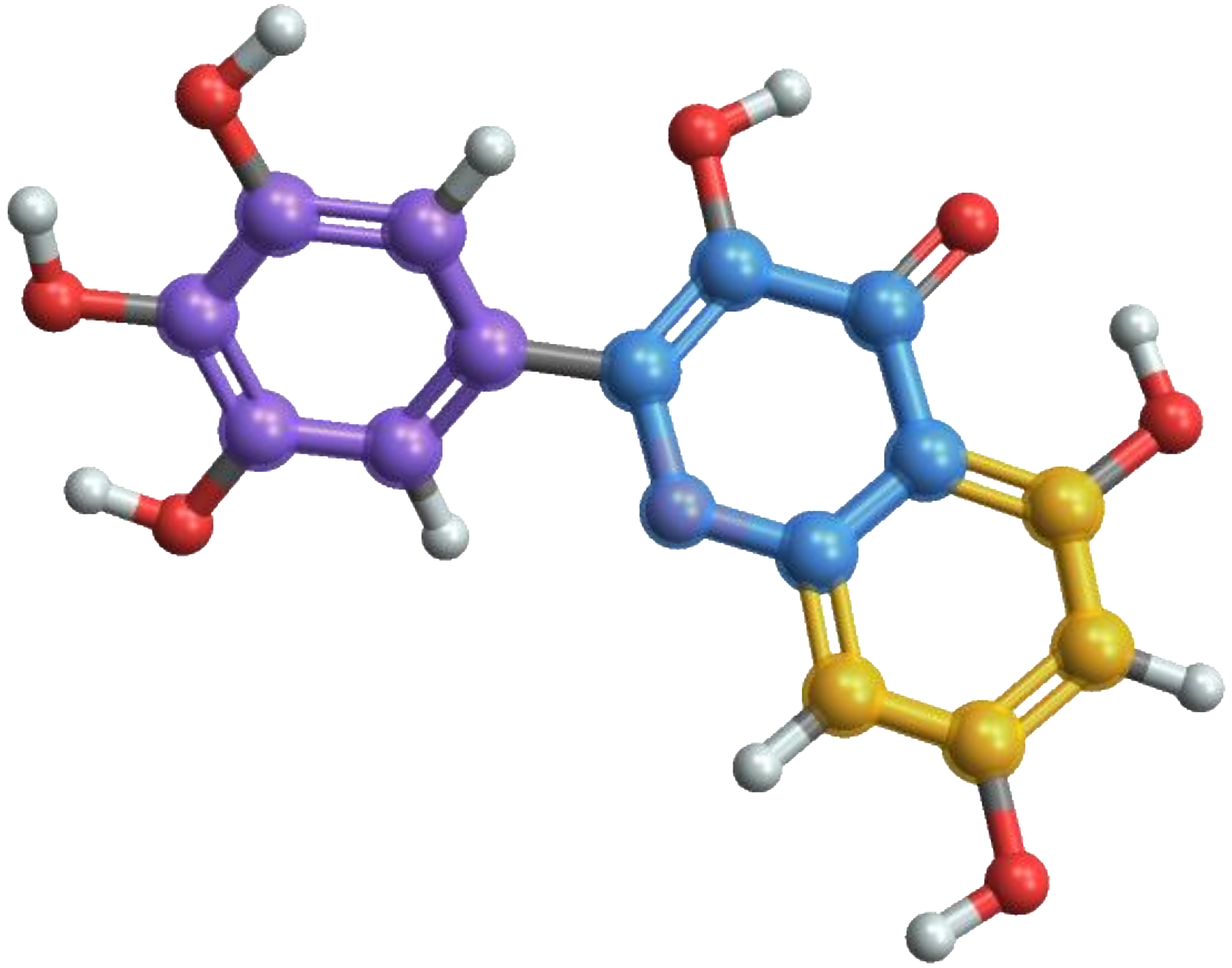


Question: How would you apply this idea to a graph?



Graph Convolutions: Leverage neighboring nodes

Question: How would you apply this idea to a graph?



connected neighbors

$$\mathrm{GraphConv}(X)[i] = \Theta_b X_i + \sum_j \Theta X[j]$$

Graph Convolutions: Leverage neighboring nodes

Agenda

Recap

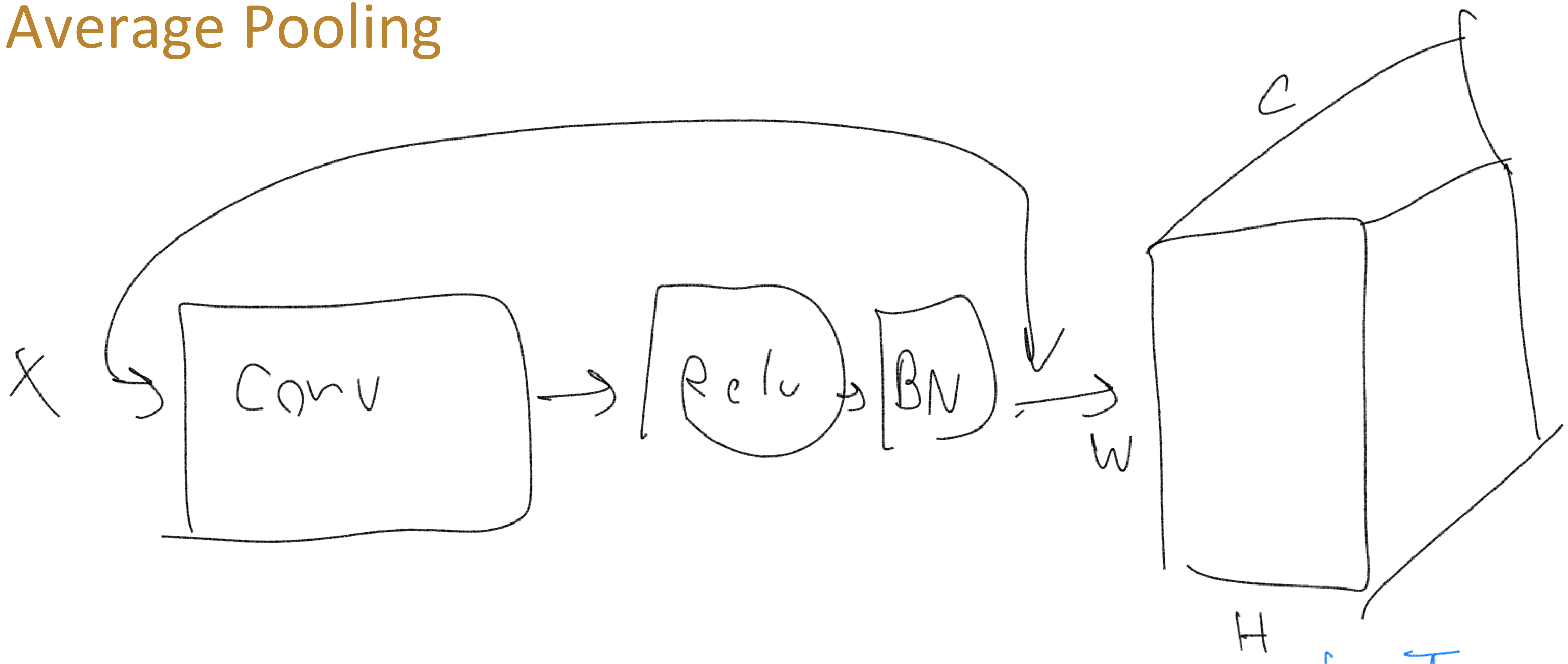
Failure modes of fully-connected neural networks

Convolutions

Pooling: Aggregating features and location invariance

CNNs across modalities

Average Pooling

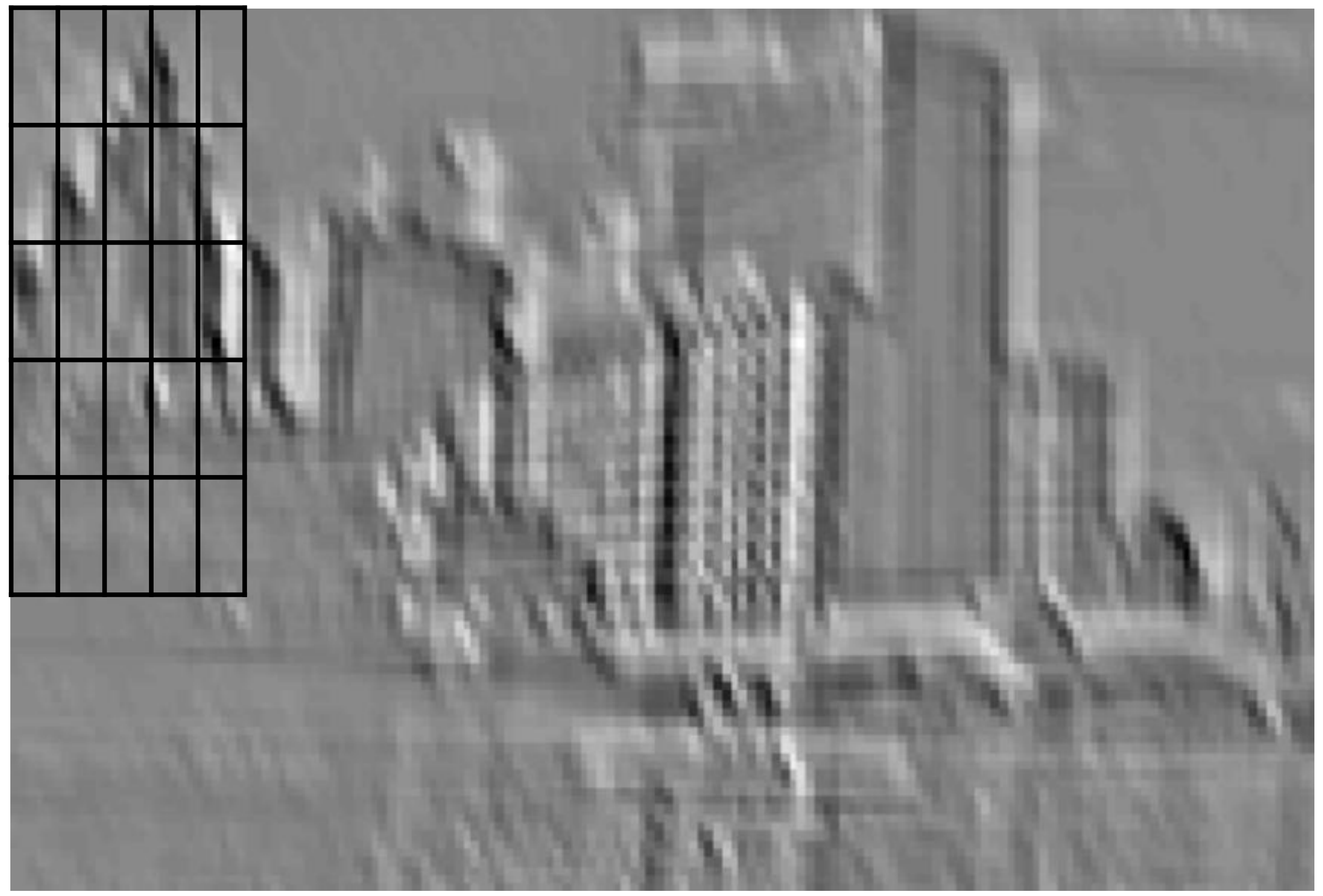


$$\mathrm{AvgPool}(z) = \frac{1}{HW} \sum_{i}^{H} \sum_{j}^{W} z[:, i, j]$$

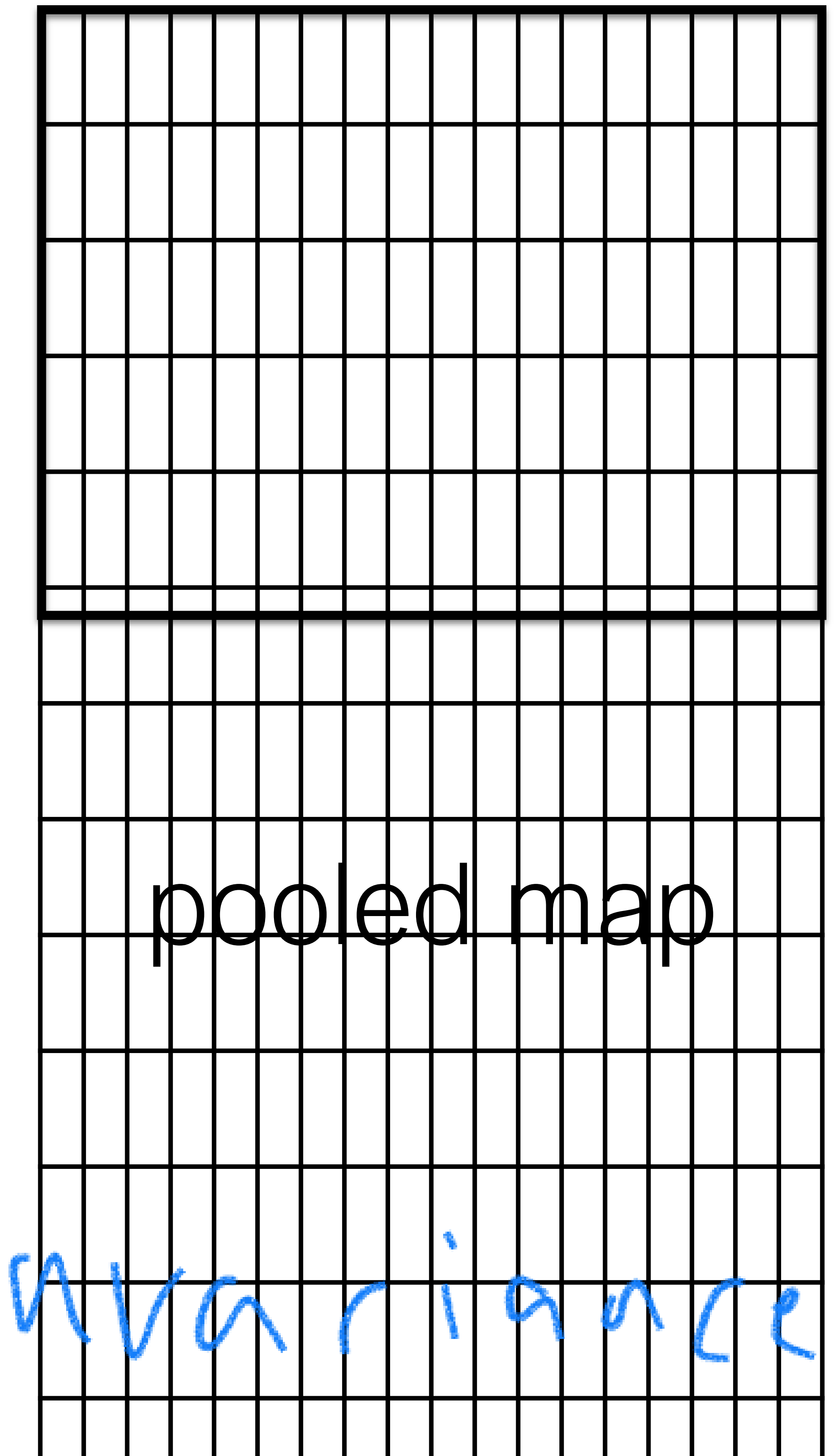
Translation invariance!

Pooling

- We wish to know whether a feature was there but not exactly where it was



feature map

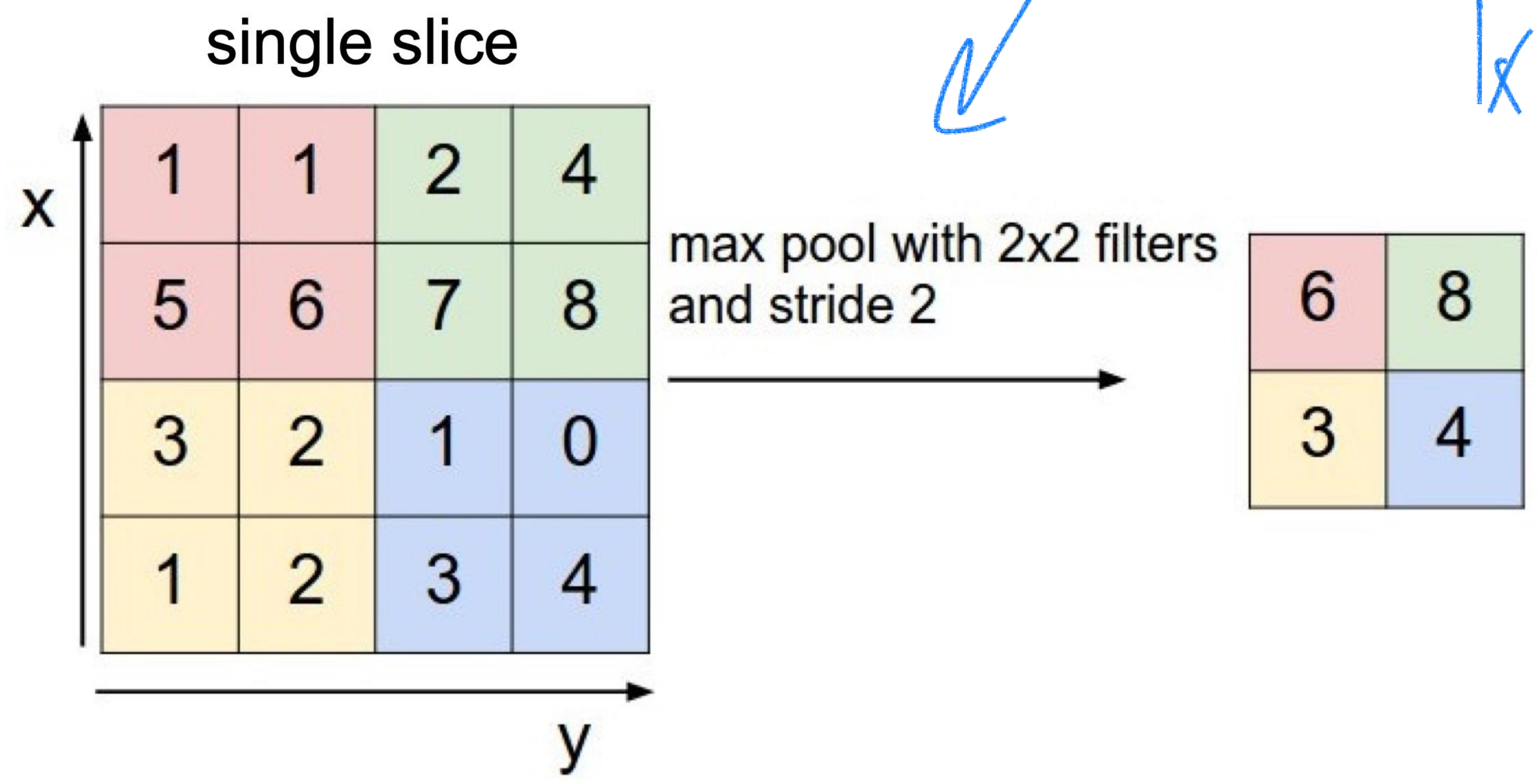


pooled map

Translation invariance!

Max Pooling

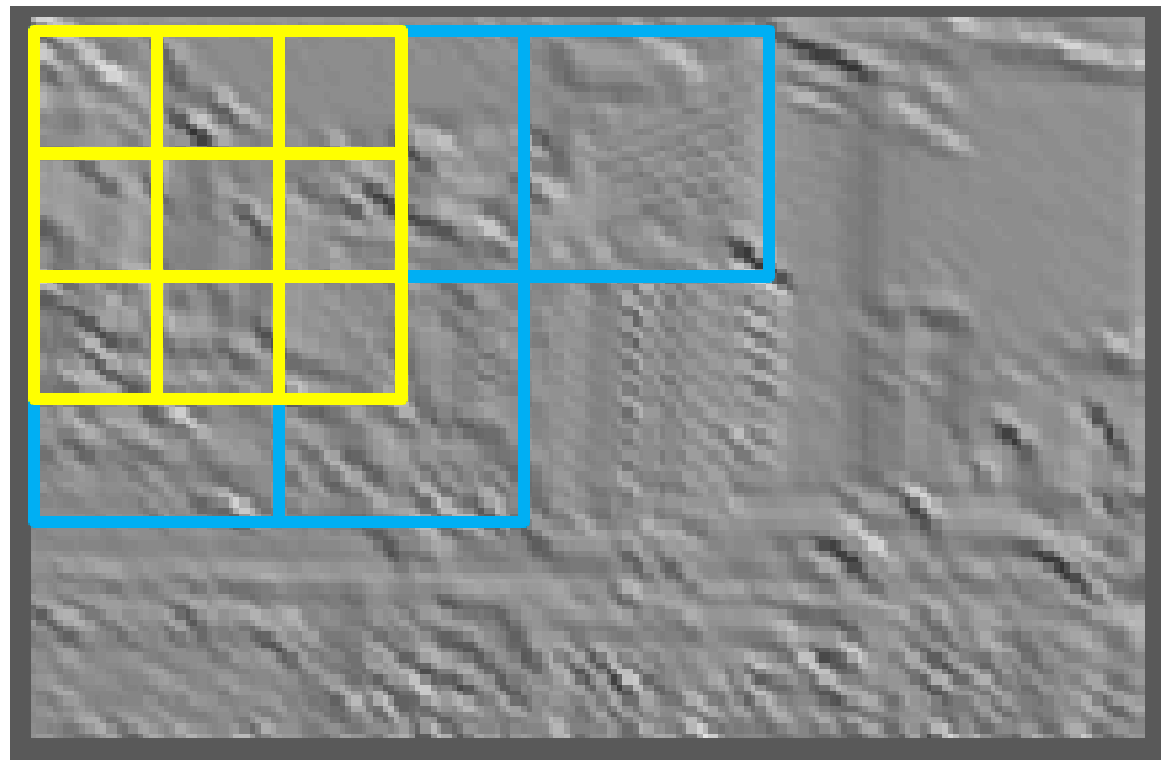
Pooling as conv kernel!



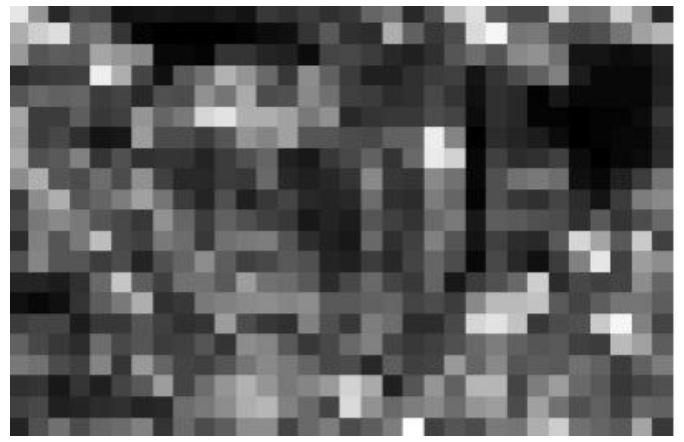
- Similar to filtering, but output the maximum entry instead of a weighted sum

Pooling (max)

- Pooling region and "stride" may vary
  - pooling induces translation invariance at the cost of spatial resolution
  - stride reduces the size of the resulting feature map

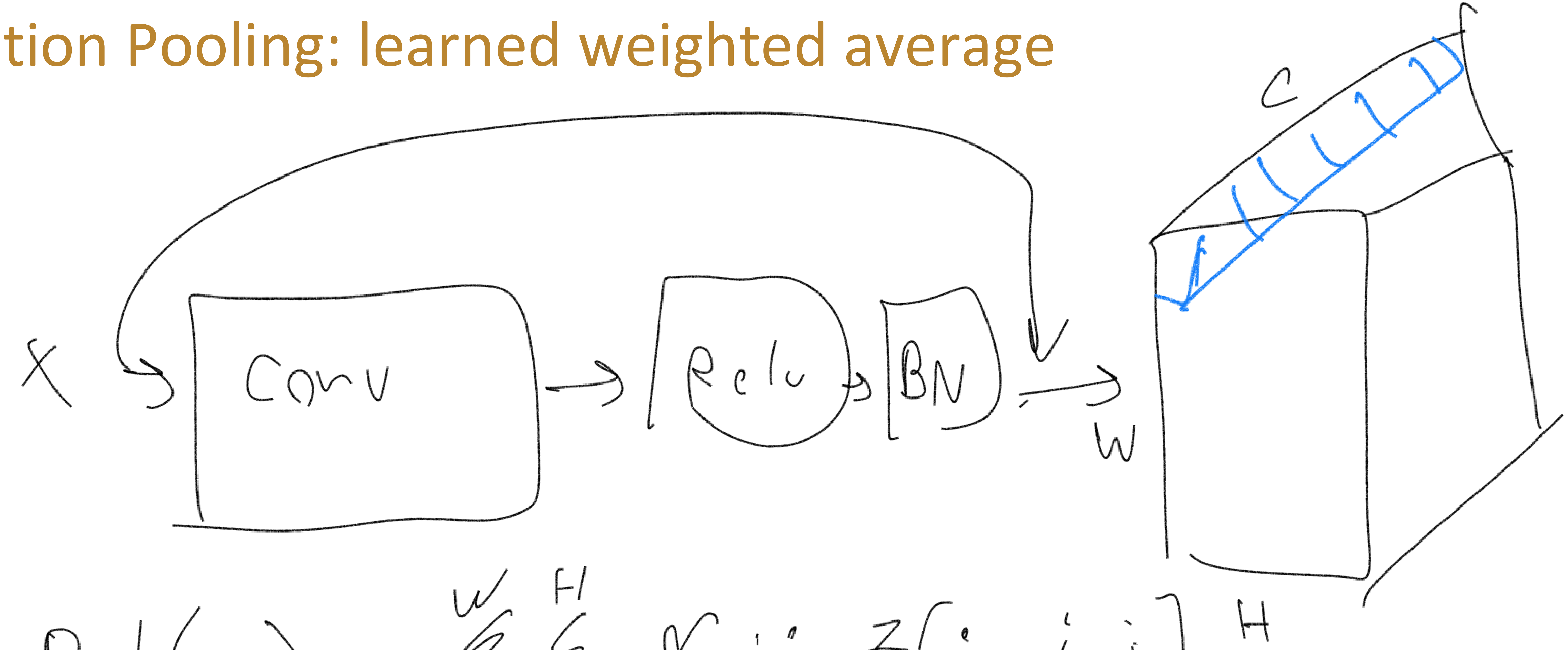


feature map



feature map
after max pooling

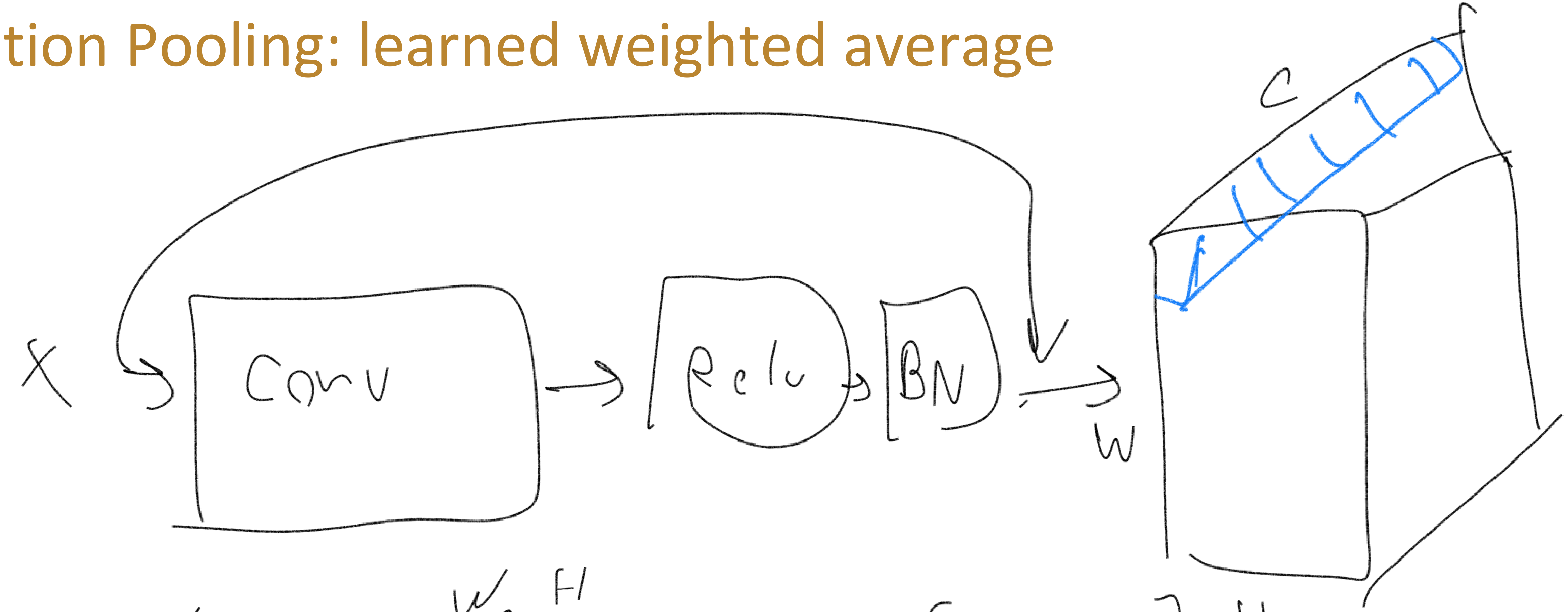
Attention Pooling: learned weighted average



$$\mathrm{AttnPool}(z) = \sum_{i}^{W} \sum_{j}^{H} \alpha_{ij}\, z[:, i, j]$$

$$\alpha = \mathrm{Softmax}(f(z)) \qquad |f(z)| \to (\cdot, W, H, 1)$$

Attention Pooling: learned weighted average



$$\mathrm{AttnPool}(z) = \sum_i^W \sum_j^H \alpha_{ij}\, z[:, i, j]$$

$$\alpha = \mathrm{Softmax}(f(z)) \qquad \mathrm{Softmax}(a) = \frac{e^{a}}{\sum_i e^{a_i}}$$

Multi-Head Attention Pooling: more shots on goal

$$\text{AttnPool}(z) = \sum_{i}^{W} \sum_{j}^{F} \alpha_{ij}\, z[:, i, j]$$

$$\text{MultiHeadAttn}(z) = [\underbrace{\text{AttnPool}(z) \,;\, \text{AttnPool}(z)}_{h \text{ unique } \alpha\text{s!}} \ldots]$$

Focus on different things

Agenda

Recap

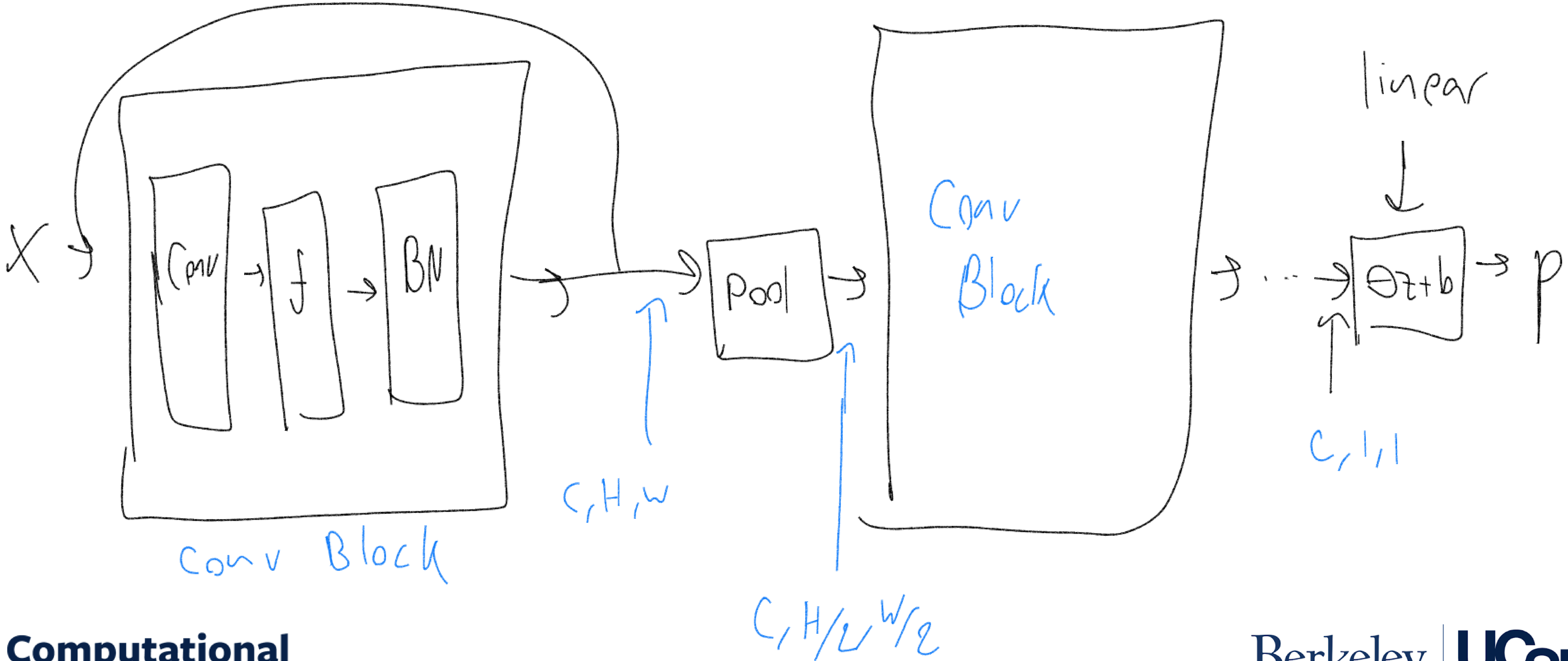
Failure modes of fully-connected neural networks

Convolutions

Pooling

CNNs across modalities

Putting it together: CNNs



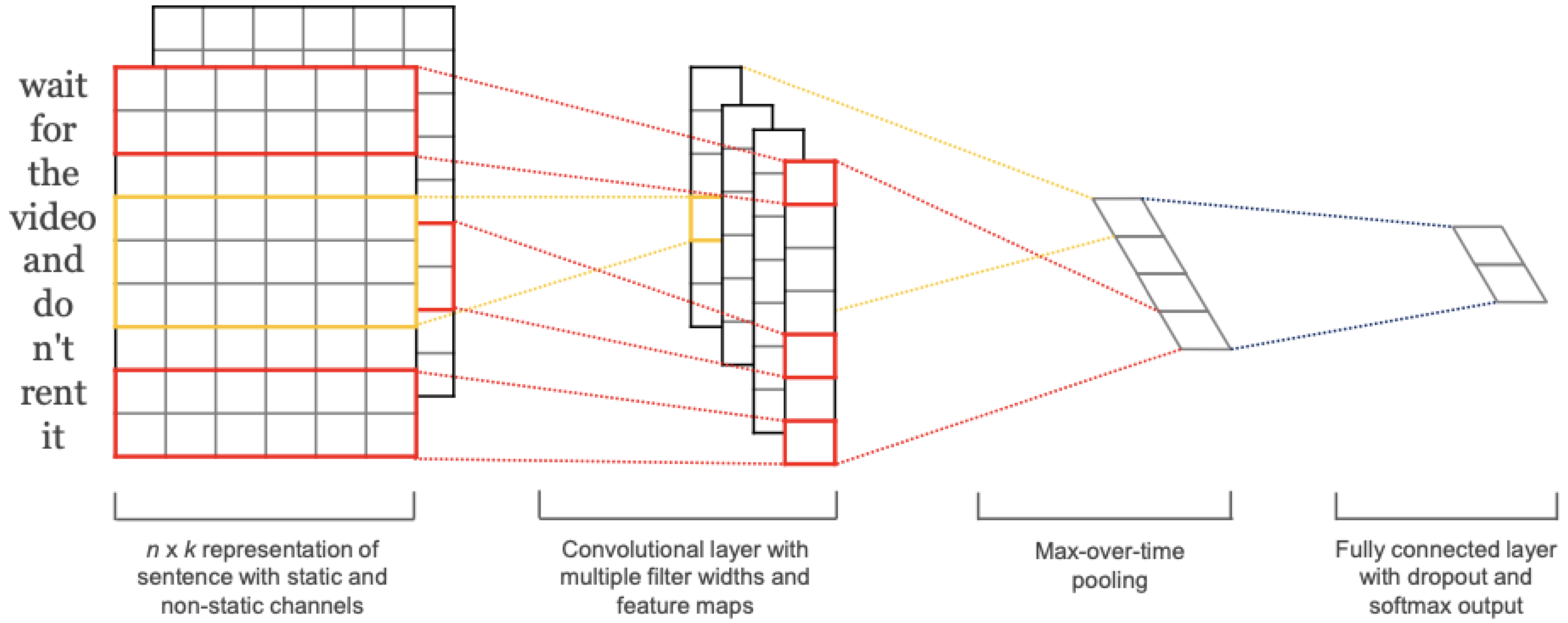
Computational
PRECISION HEALTH

Berkeley | UCSF

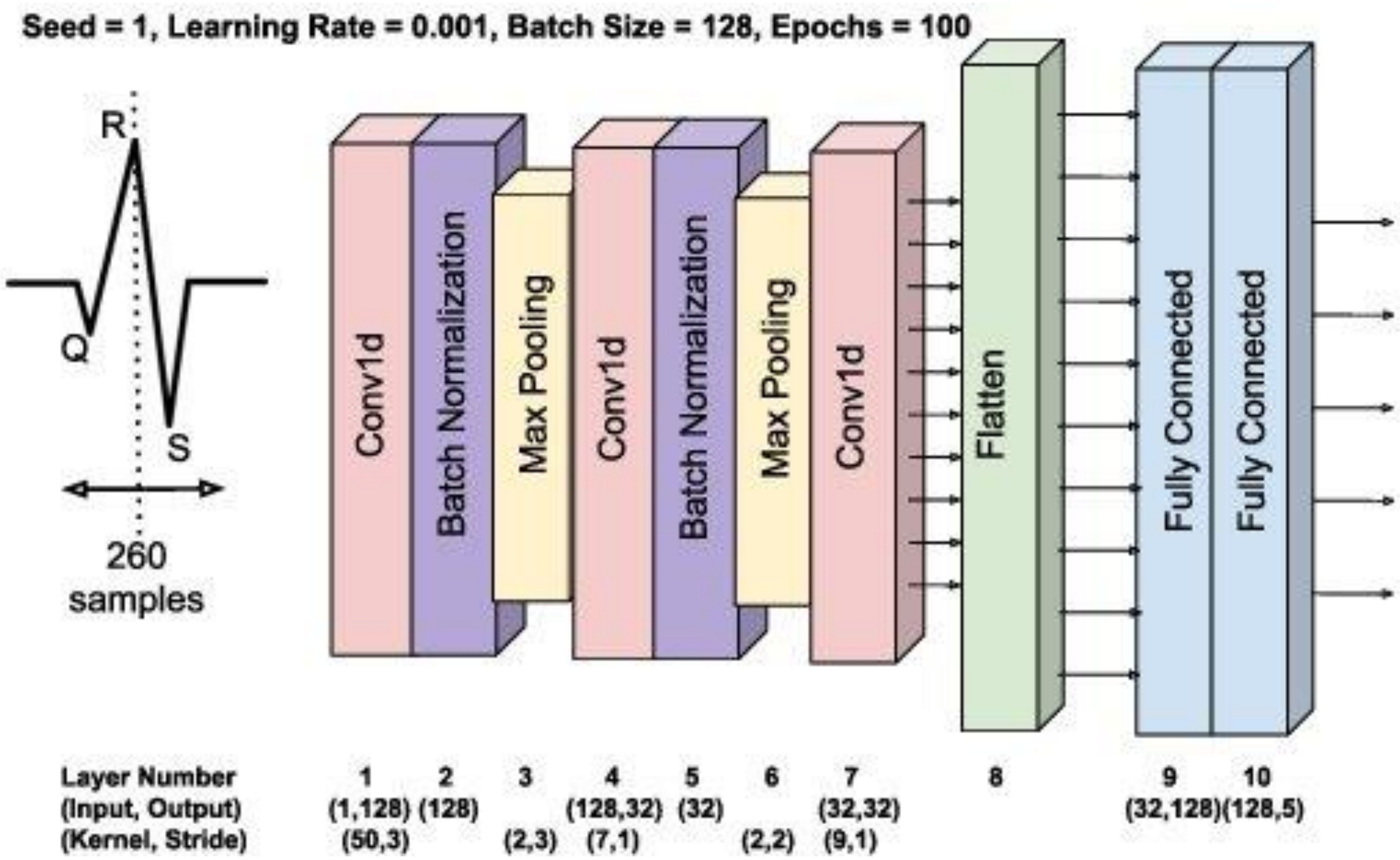
Popular 1D CNNs: Text

Convolutional Neural Networks for Sentence Classification

Yoon Kim
New York University
yhk255@nyu.edu

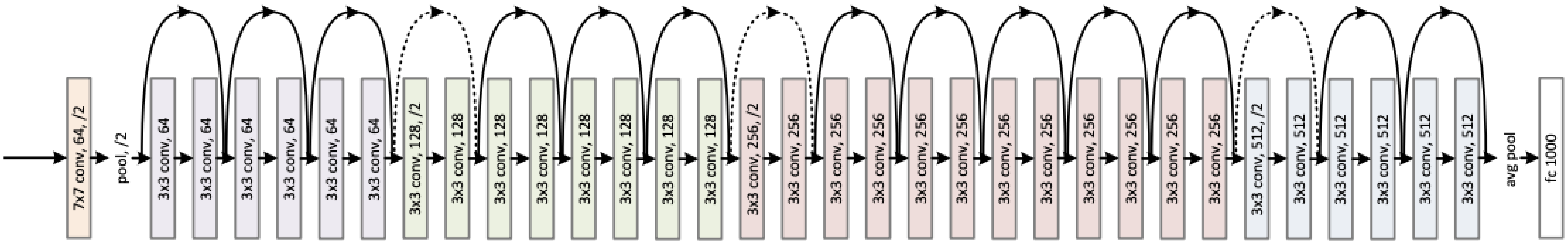


Popular 1D CNNs: Wave Forms



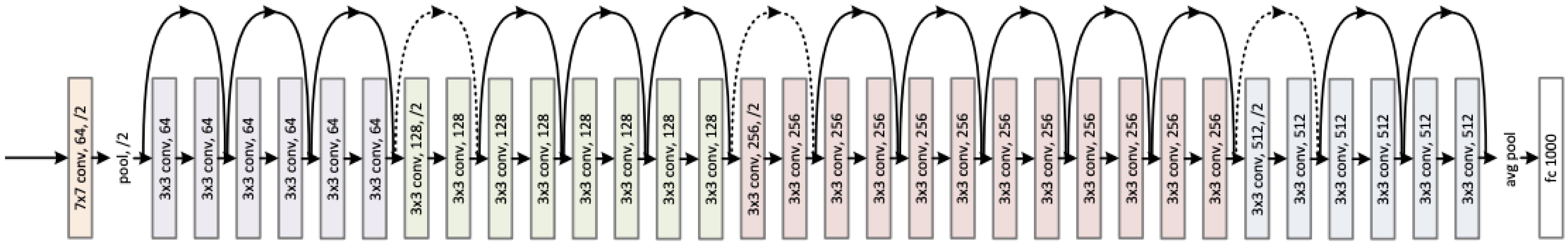
Xiaolin, Li, Barry Cardiff, and Deepu John. "A 1d convolutional neural network for heartbeat classification from single lead ecg." *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2020.

Popular 2D CNNs: ResNets



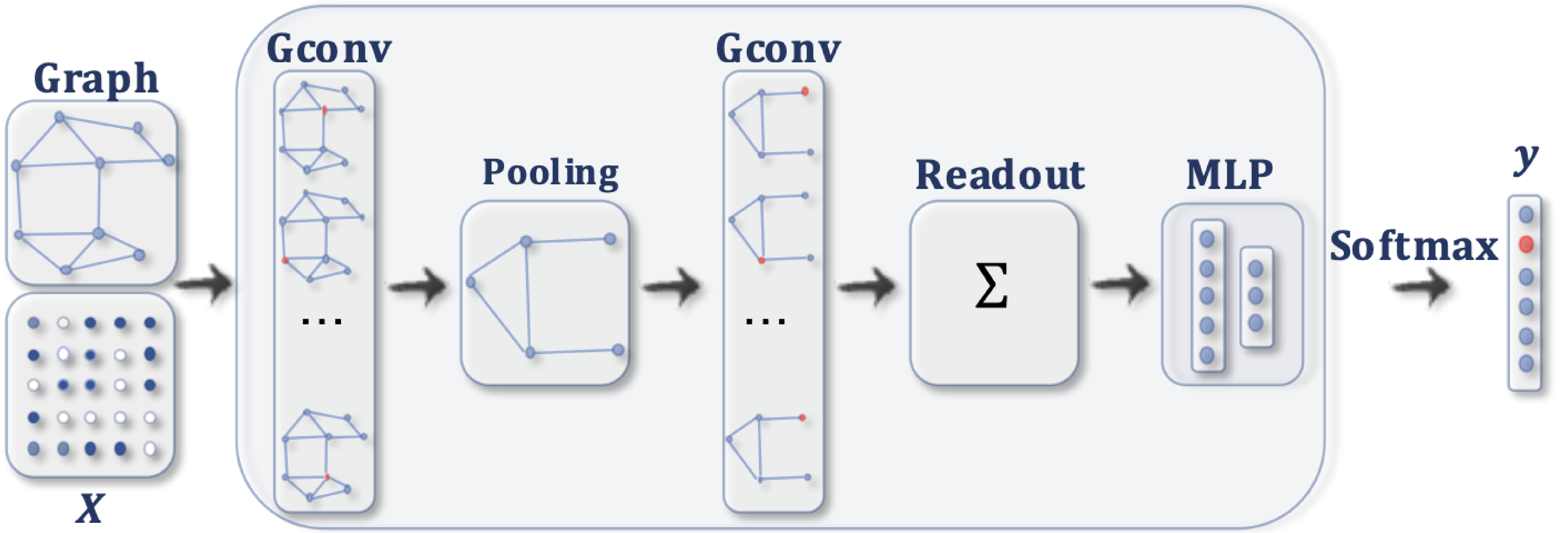
He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

Popular 3D CNNs: ResNet3D



Just make the kernels all 3D. Used in Sybil and many other 3D models.

Popular GNNs: Convolutions on graphs



(b) A ConvGNN with pooling and readout layers for graph classification [21]. A graph convolutional layer is followed by a pooling layer to coarsen a graph into sub-graphs so that node representations on coarsened graphs represent higher graph-level representations. A readout layer summarizes the final graph representation by taking the sum/mean of hidden representations of sub-graphs.

Wu, Zonghan, et al. "A comprehensive survey on graph neural networks." *IEEE transactions on neural networks and learning systems* 32.1 (2020): 4-24.

Summary

FFNs are **wildly** inefficient

Convolutions: Capture **local patterns** data

Pooling: **Spatial invariant** method to summarize features

Attention Pooling: Parameterized Weighted Averages

CNNs: NNs with Conv and Pooling building blocks

Applications to text, images, graphs, and more

Questions?