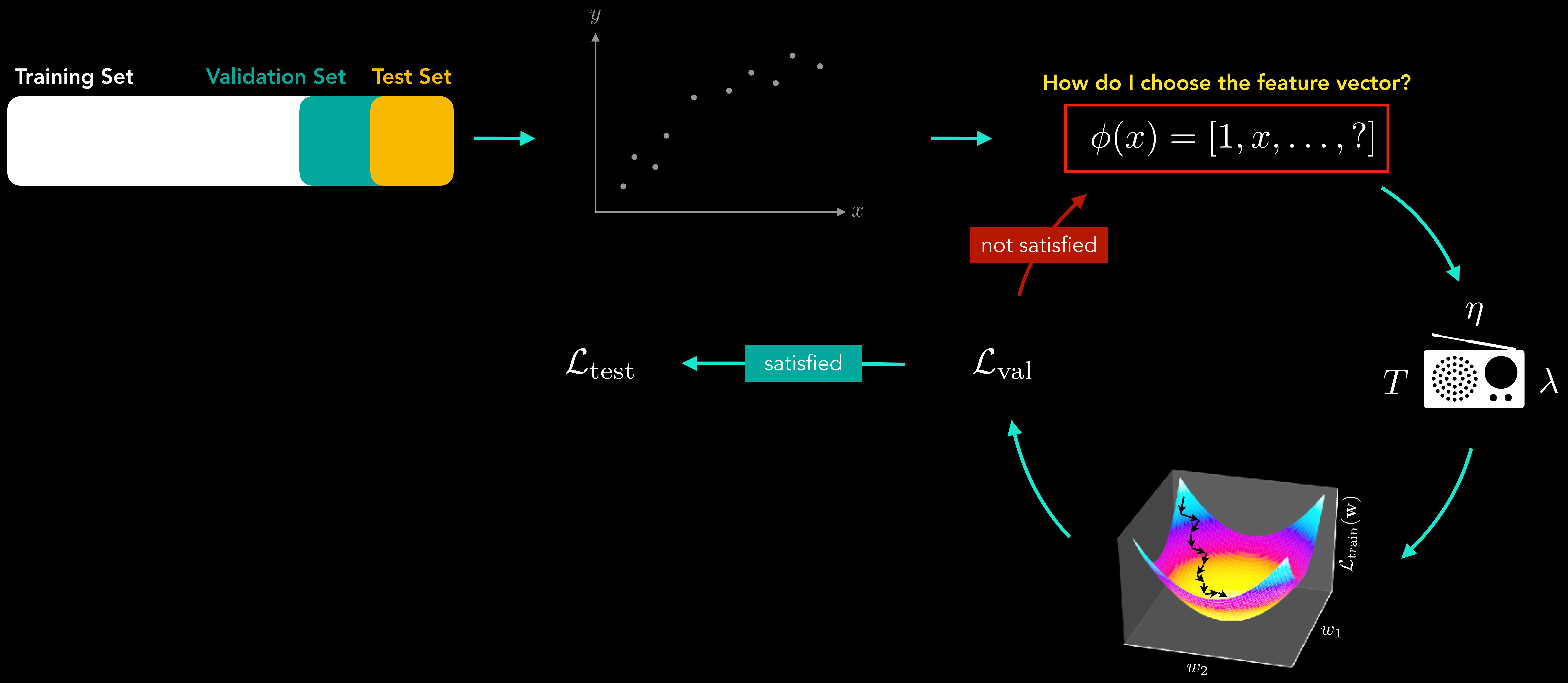
Deep Learning

The ML workflow



How do I choose the feature vector?

$$\phi(x) = [1, x, \ldots, ?]$$

$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$$

$\mathbf{x}$

$$\phi(x) = [1,\ x]$$

$$\phi(x) = [1, x, x^2, x^3]$$

$$\phi(x) = [1,\ x,\ \sin(3x)]$$

??????????????

How do I choose the feature vector?

$$\phi(x) = [1, x, \ldots, ?]$$

Decision Boundary

$$\phi(x) \cdot \mathbf{w} = 0$$

→ **Boat**

Linear Predictor

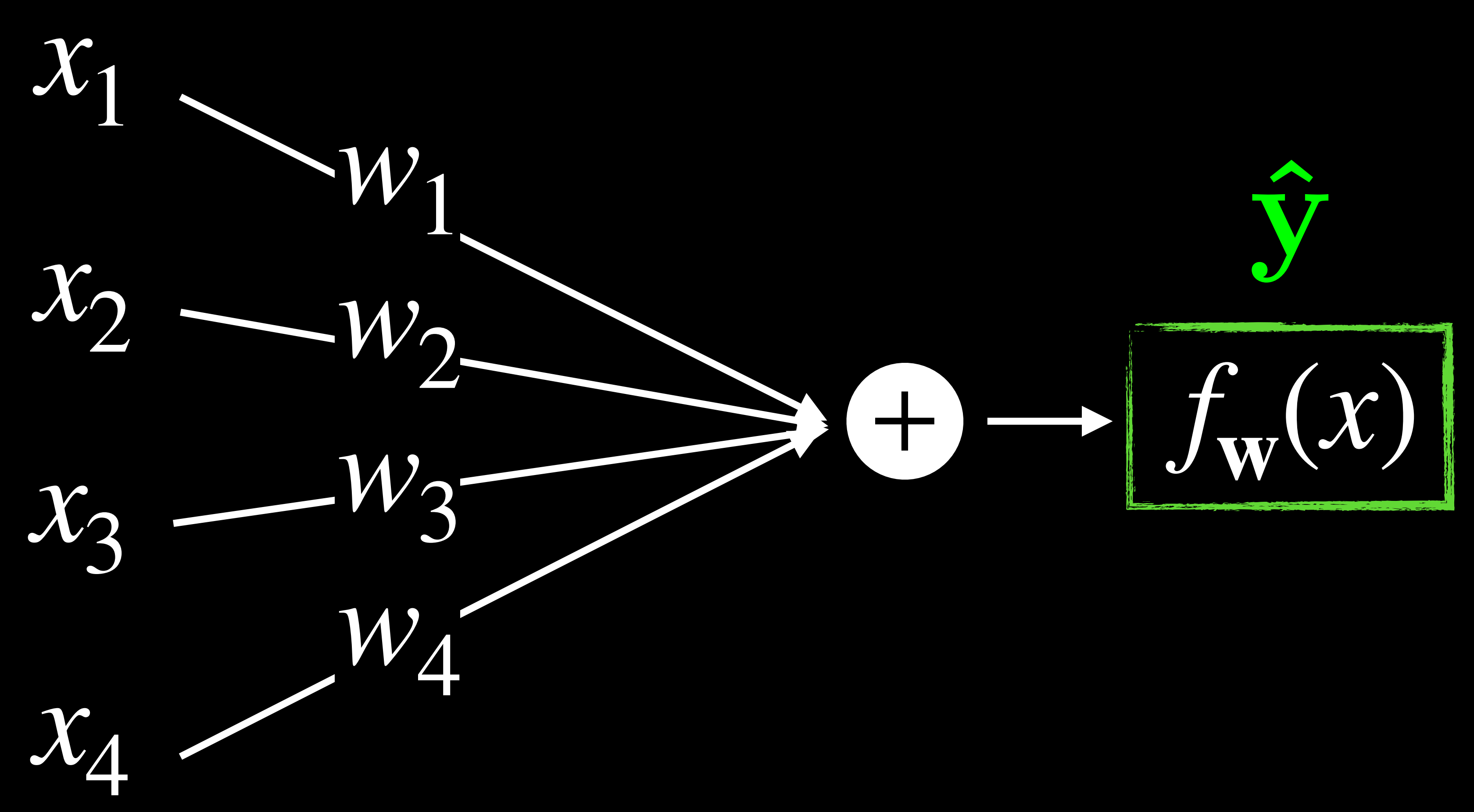
$$f_{\mathbf{w}}(x) = \mathbf{x} \cdot \mathbf{w}$$

$$\mathbf{w} = [w_1, w_2, w_3, w_4]$$

$$\mathbf{x} = [x_1, x_2, x_3, x_4]$$

$$f_{\mathbf{w}}(x) = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4$$

Network Representation



Linear Predictor

2 outputs?

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \longrightarrow \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \end{bmatrix}$$

$$\mathbf{x} \qquad\qquad \hat{\mathbf{y}}$$

3 * 2 fitting parameters

$$\hat{y}_1 = \mathbf{w}_1 \cdot \mathbf{x} = w_{11}x_1 + w_{12}x_2 + w_{13}x_3$$

$$\hat{y}_2 = \mathbf{w}_2 \cdot \mathbf{x} = w_{21}x_1 + w_{22}x_2 + w_{23}x_3$$

Matrix form

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\hat{\mathbf{y}} = \mathbf{W} \mathbf{x}$$

From Matrix to Network

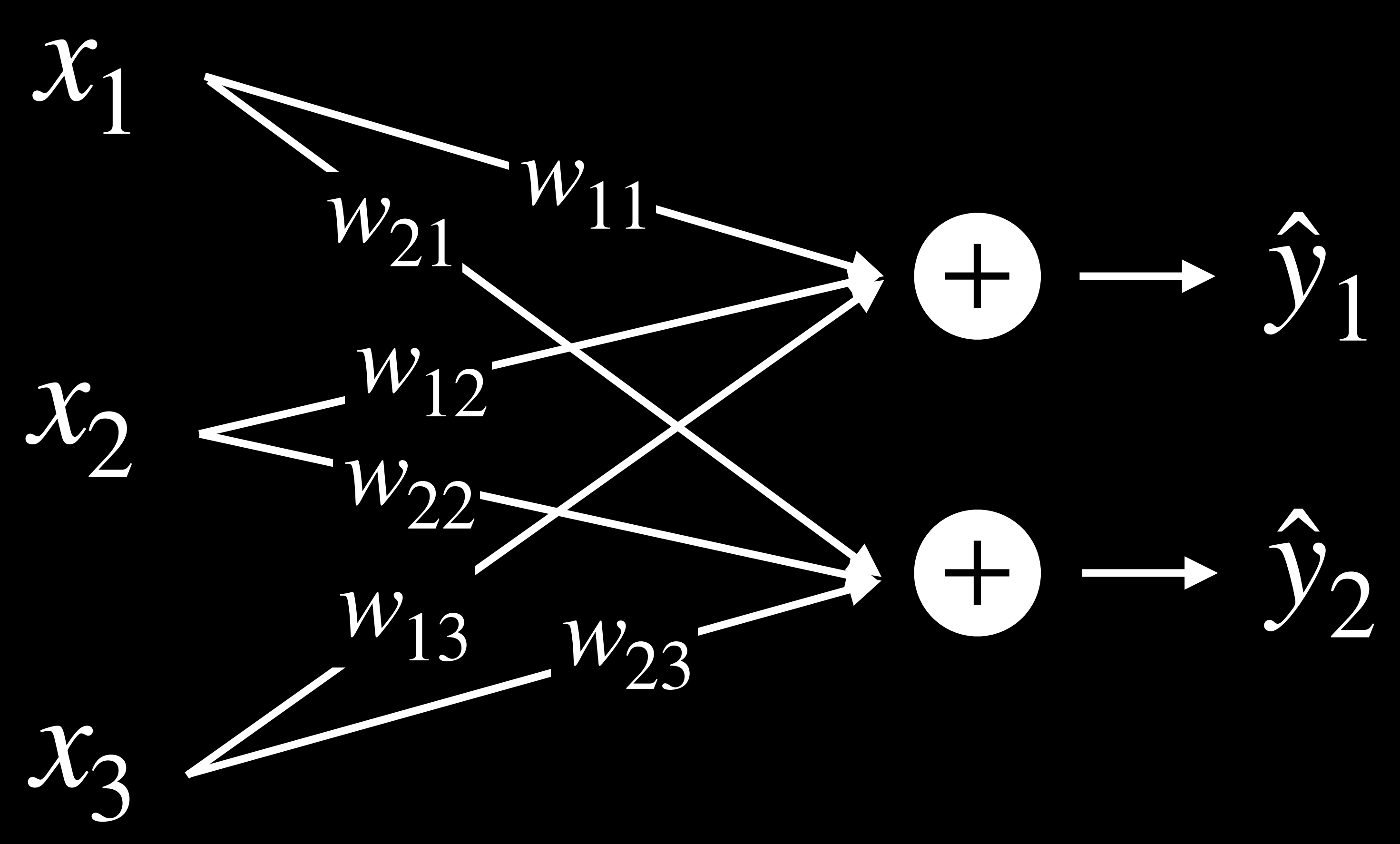
Matrix Representation

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\hat{\mathbf{y}} = \mathbf{W} \mathbf{x}$$

Network Representation

Account for Bias: 1 =



Index notation

$$\hat{y}_i = \sum_{j=1}^{n} w_{ij} x_j$$

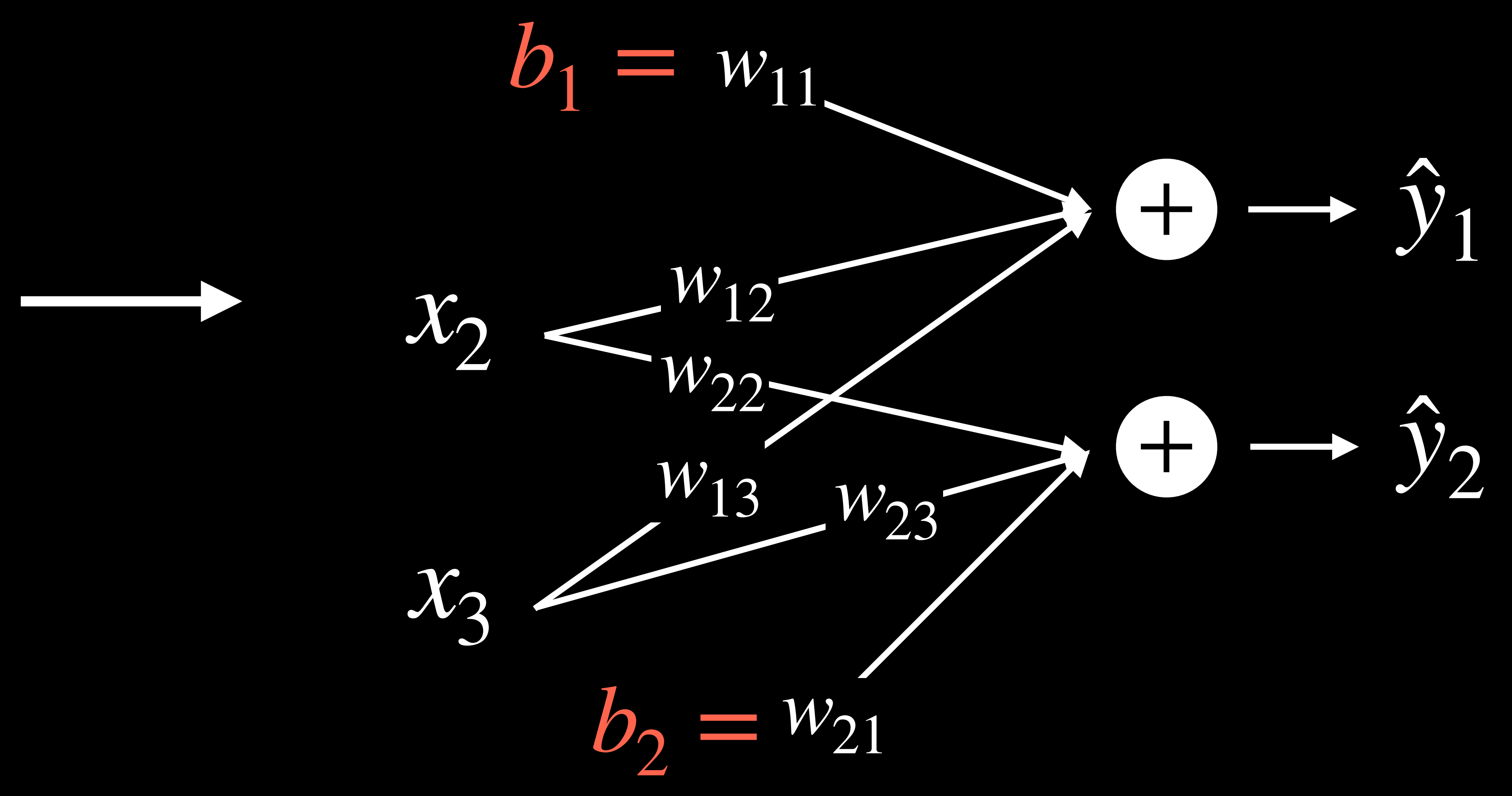
Linear Predictor - Explicit Bias

Matrix Representation

$$\begin{bmatrix}\hat{y}_1\\ \hat{y}_2\end{bmatrix} = \begin{bmatrix}w_{12} & w_{13}\\ w_{22} & w_{23}\end{bmatrix}\begin{bmatrix}x_2\\ x_3\end{bmatrix} + \begin{bmatrix}w_{11}\\ w_{21}\end{bmatrix}$$

$$\hat{\mathbf{y}} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

Network Representation



Linear Predictor

$$\hat{\mathbf{y}} = \mathbf{W} \mathbf{x} + \mathbf{b}$$

$m \times 1$ — Number of outputs

$m \times n$

$n \times 1$ — Number of inputs

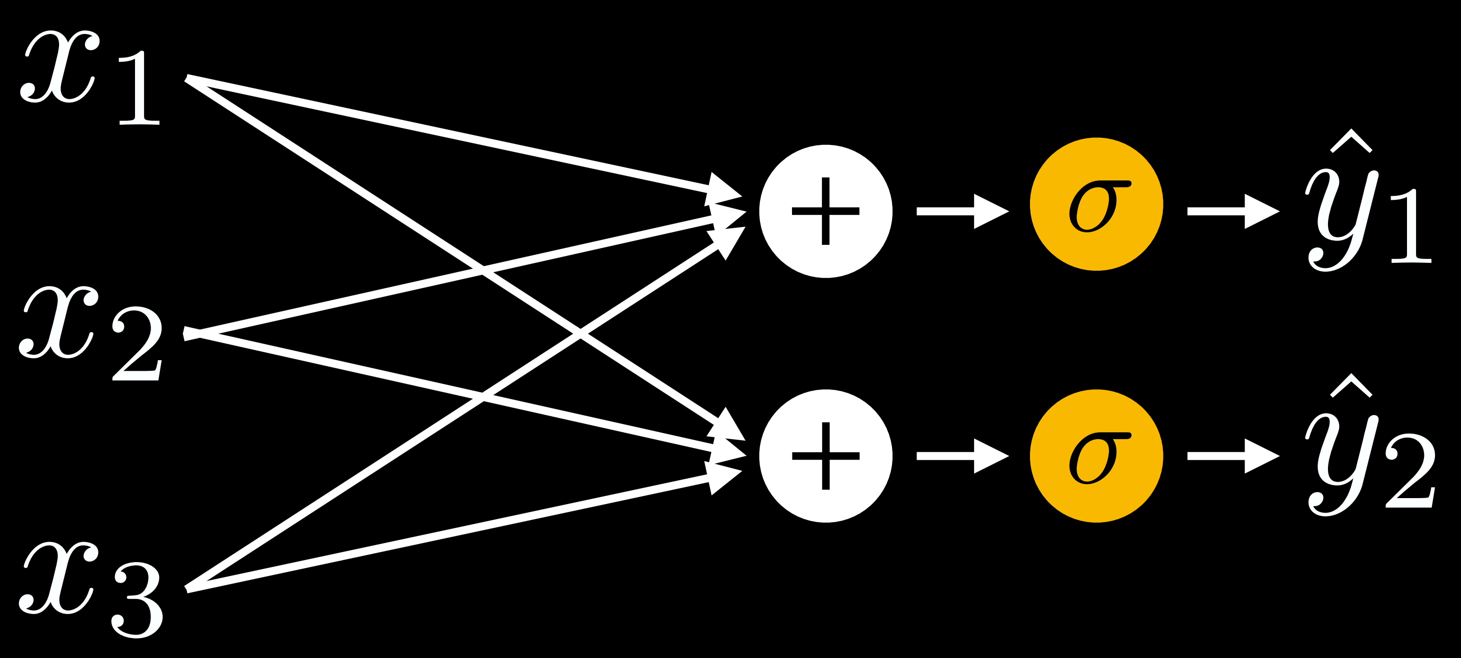
$m \times 1$

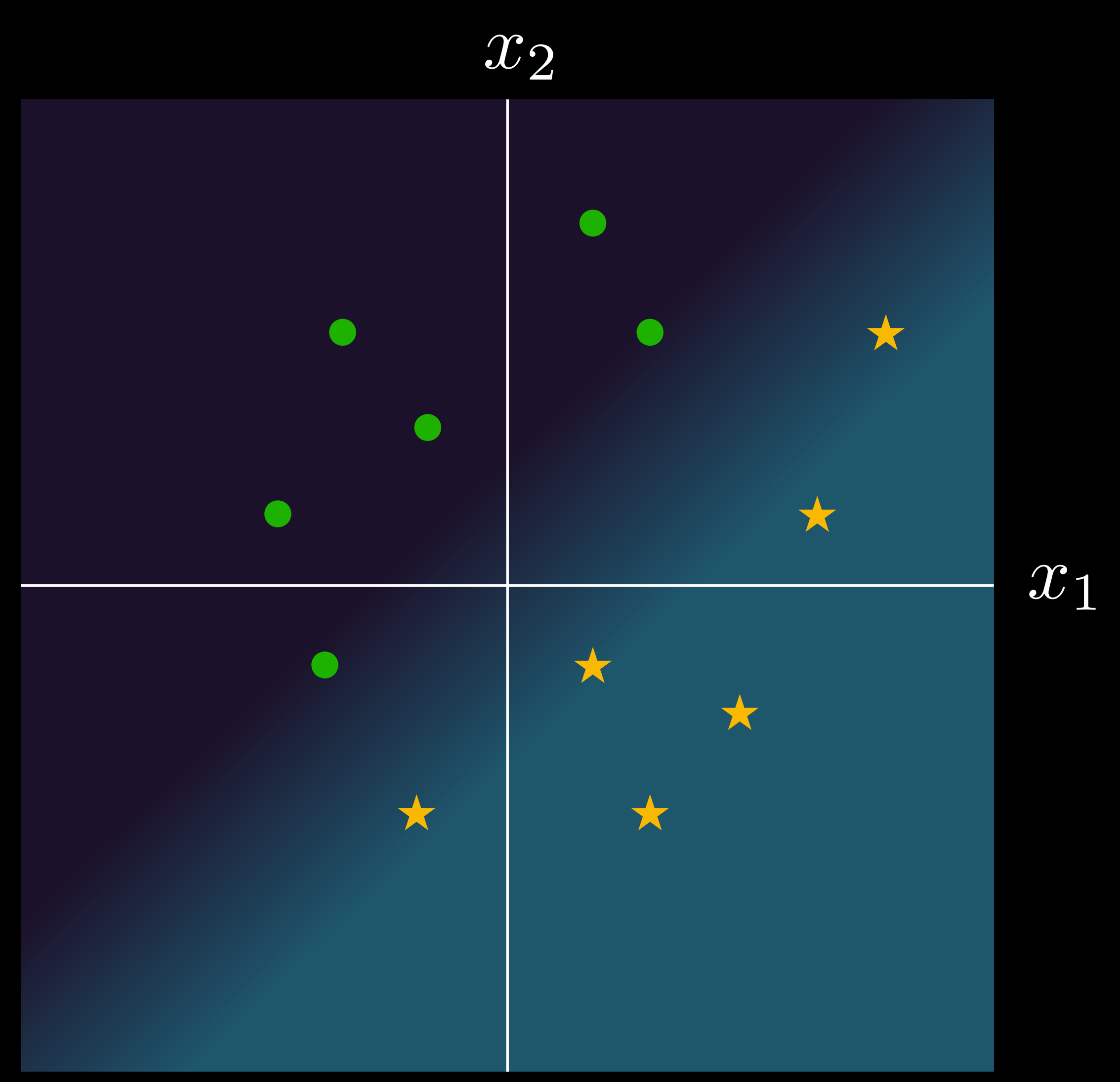
Some formulations explicitly account for $\mathbf{b}$, while others include the bias as part of $\mathbf{x}$

Here we omit $\mathbf{b}$ for simplicity of representation

Nonlinear Predictor

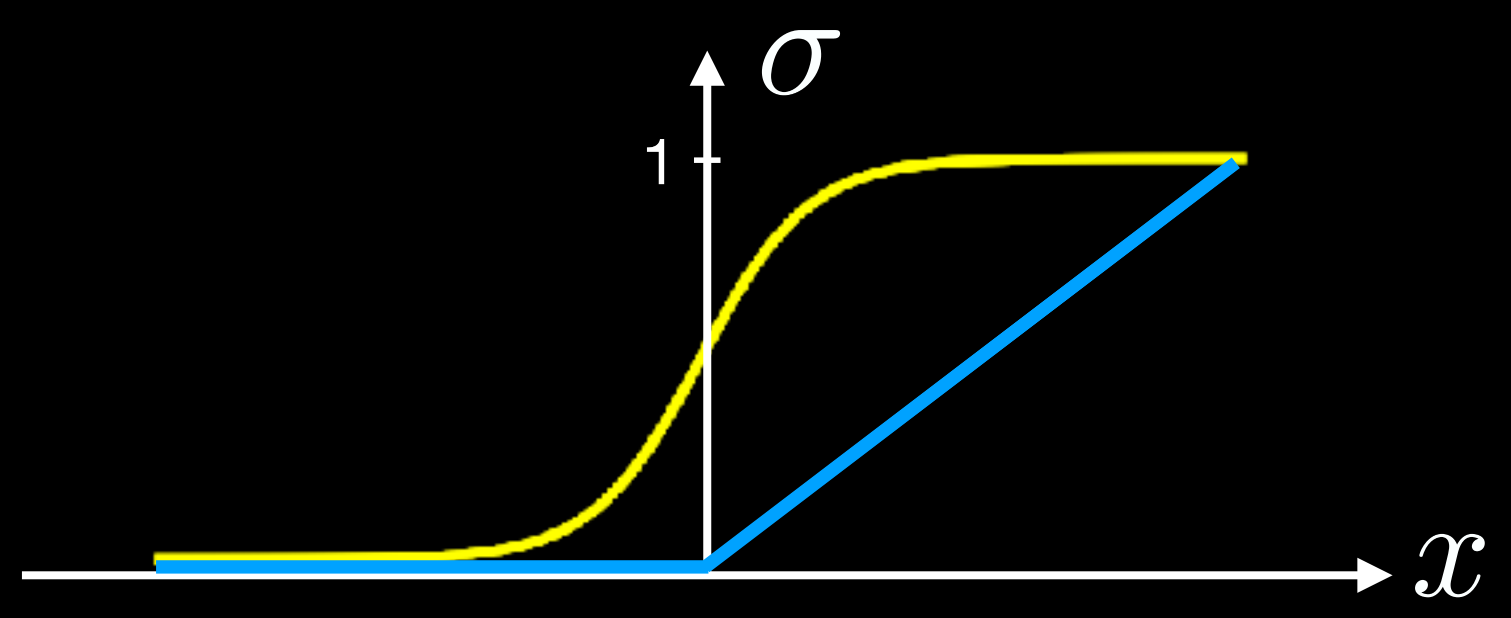
$$\hat{\mathbf{y}} = \sigma(\mathbf{W}\mathbf{x})$$





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

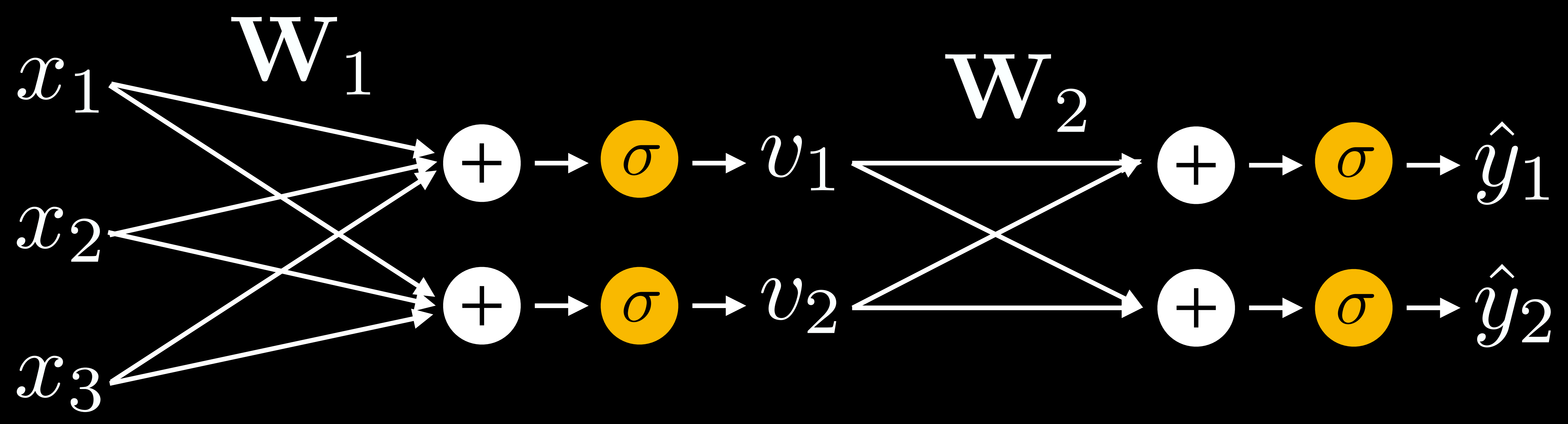
ReLU: $\sigma(x) = xH(x)$



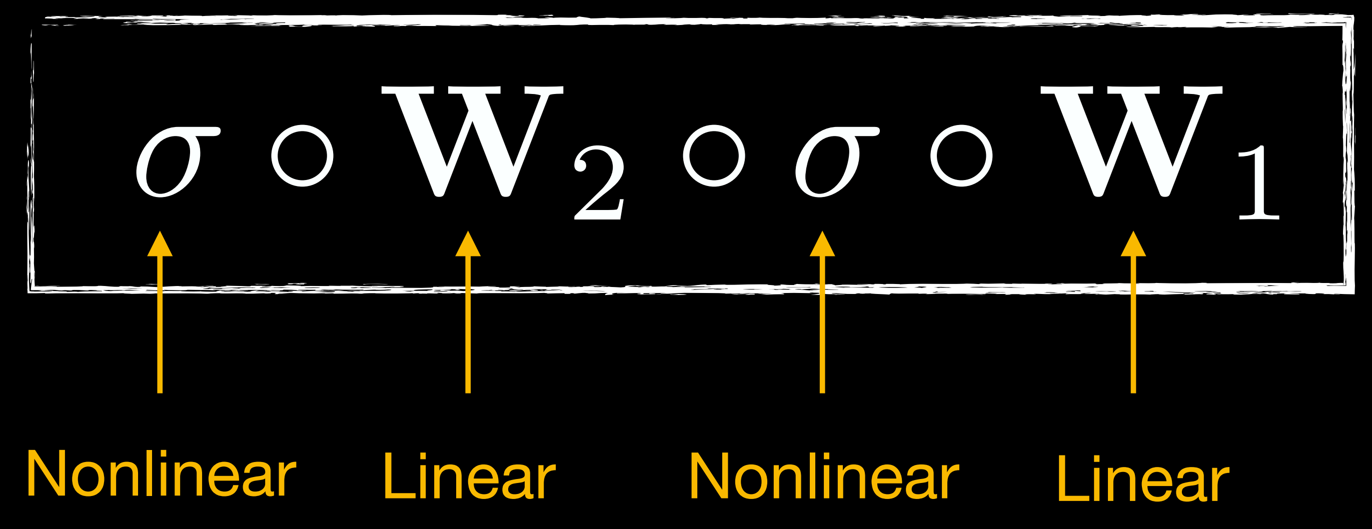
Activation Function

Neural network

$$\hat{\mathbf{y}} = \sigma\left(\mathbf{W}_2\, \sigma\left(\mathbf{W}_1 \mathbf{x}\right)\right)$$



$$\mathbf{x} \rightarrow \boxed{\sigma \circ \mathbf{W}_2 \circ \sigma \circ \mathbf{W}_1} \rightarrow \hat{\mathbf{y}}$$

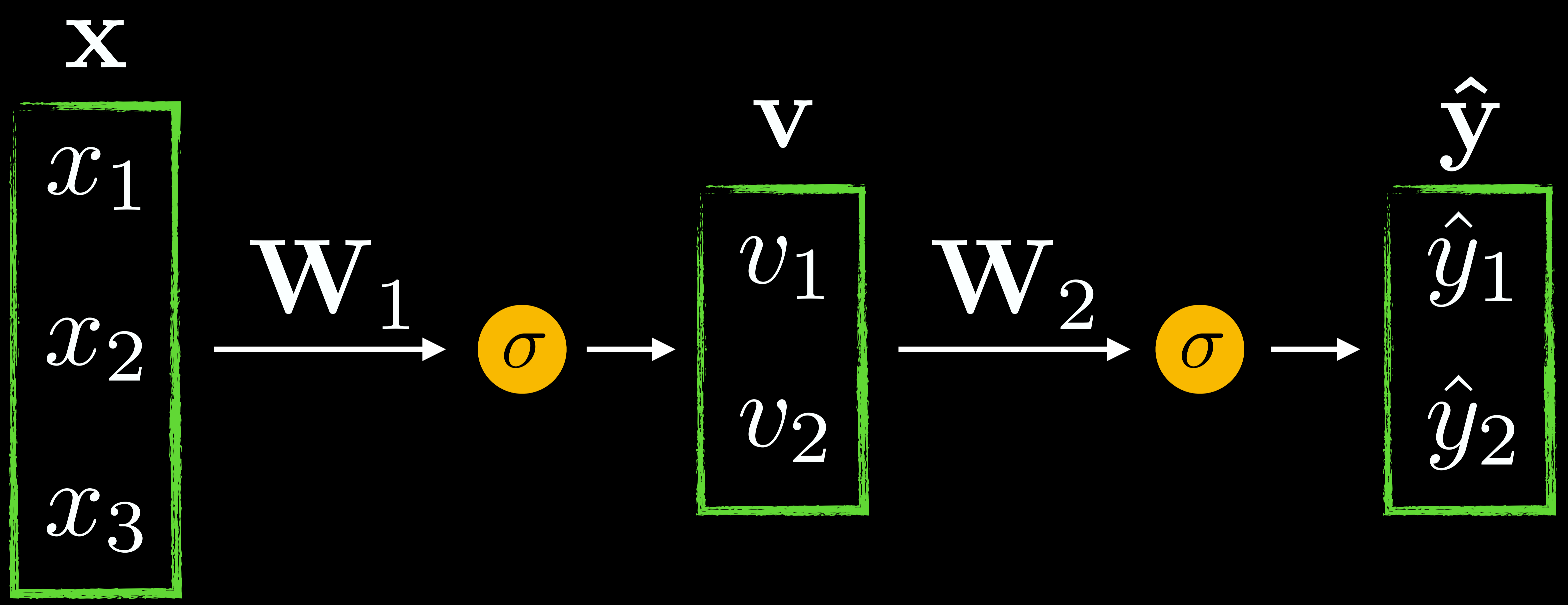


Neural network

$$\hat{\mathbf{y}} = \sigma\left(\mathbf{W}_2\, \sigma\left(\mathbf{W}_1 \mathbf{x}\right)\right)$$

$$\mathbf{v} = \sigma(\mathbf{W}_1 \mathbf{x})$$

$$\hat{\mathbf{y}} = \sigma(\mathbf{W}_2 \mathbf{v})$$

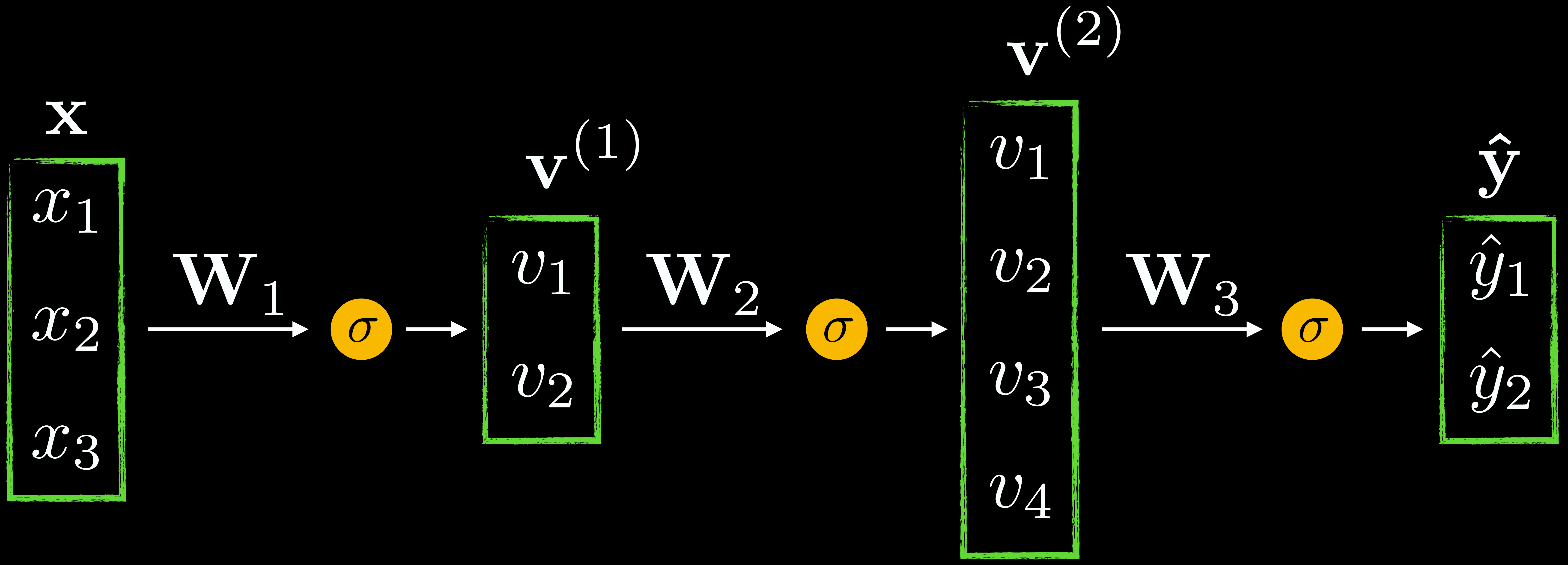


Hidden layer

Can be interpreted as a learned $\phi(\mathbf{x})$

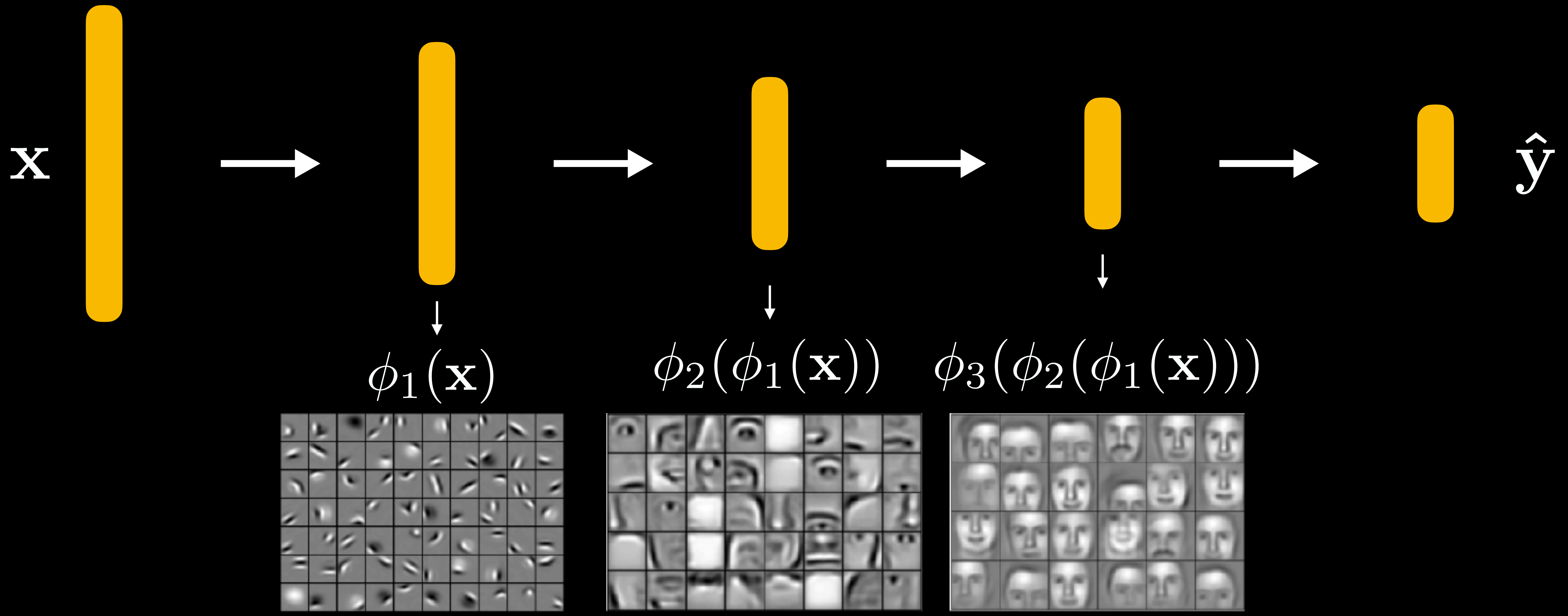
Deep network

$$\hat{\mathbf{y}} = \sigma\left(\mathbf{W}_3\, \sigma\left(\mathbf{W}_2\, \sigma\left(\mathbf{W}_1 \mathbf{x}\right)\right)\right)$$



$$\mathbf{v}^{(1)} = \sigma\left(\mathbf{W}_1 \mathbf{x}\right) \qquad \mathbf{v}^{(2)} = \sigma\left(\mathbf{W}_2 \mathbf{v}^{(1)}\right) \qquad \hat{\mathbf{y}} = \sigma\left(\mathbf{W}_3 \mathbf{v}^{(2)}\right)$$

Why *deep* learning?



Feature learning

Lee, Honglak, et al. "Unsupervised learning of hierarchical representations with convolutional deep belief networks." Communications of the ACM 54.10 (2011)

Loss function

$$f_{\mathbf{W}_1\mathbf{W}_2}(\mathbf{x}) = \mathbf{W}_2\,\sigma(\mathbf{W}_1\mathbf{x})$$

$$\mathscr{L}(\mathbf{x}, \mathbf{y}, \mathbf{W}_1, \mathbf{W}_2) = \left\| f_{\mathbf{W}_1\mathbf{W}_2}(\mathbf{x}) - \mathbf{y} \right\|^2$$

Stochastic gradient descent update

$$\mathbf{W}_1 \leftarrow \mathbf{W}_1 - \alpha \nabla_{\mathbf{W}_1} \mathscr{L}(\mathbf{x}, \mathbf{y}, \mathbf{W}_1, \mathbf{W}_2)$$

$$\mathbf{W}_2 \leftarrow \mathbf{W}_2 - \alpha \nabla_{\mathbf{W}_2} \mathscr{L}(\mathbf{x}, \mathbf{y}, \mathbf{W}_1, \mathbf{W}_2)$$

How do we calculate the gradients?

Approach

Training loss

$$\mathscr{L}(\mathbf{W}_1, \mathbf{W}_2) = \frac{1}{n}\sum_{i=1}^{n} \mathscr{L}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{W}_1, \mathbf{W}_2)$$

Objective

$$\hat{\mathbf{W}}_1, \hat{\mathbf{W}}_2 = \arg\min_{\mathbf{W}_1, \mathbf{W}_2} \mathscr{L}(\mathbf{W}_1, \mathbf{W}_2)$$

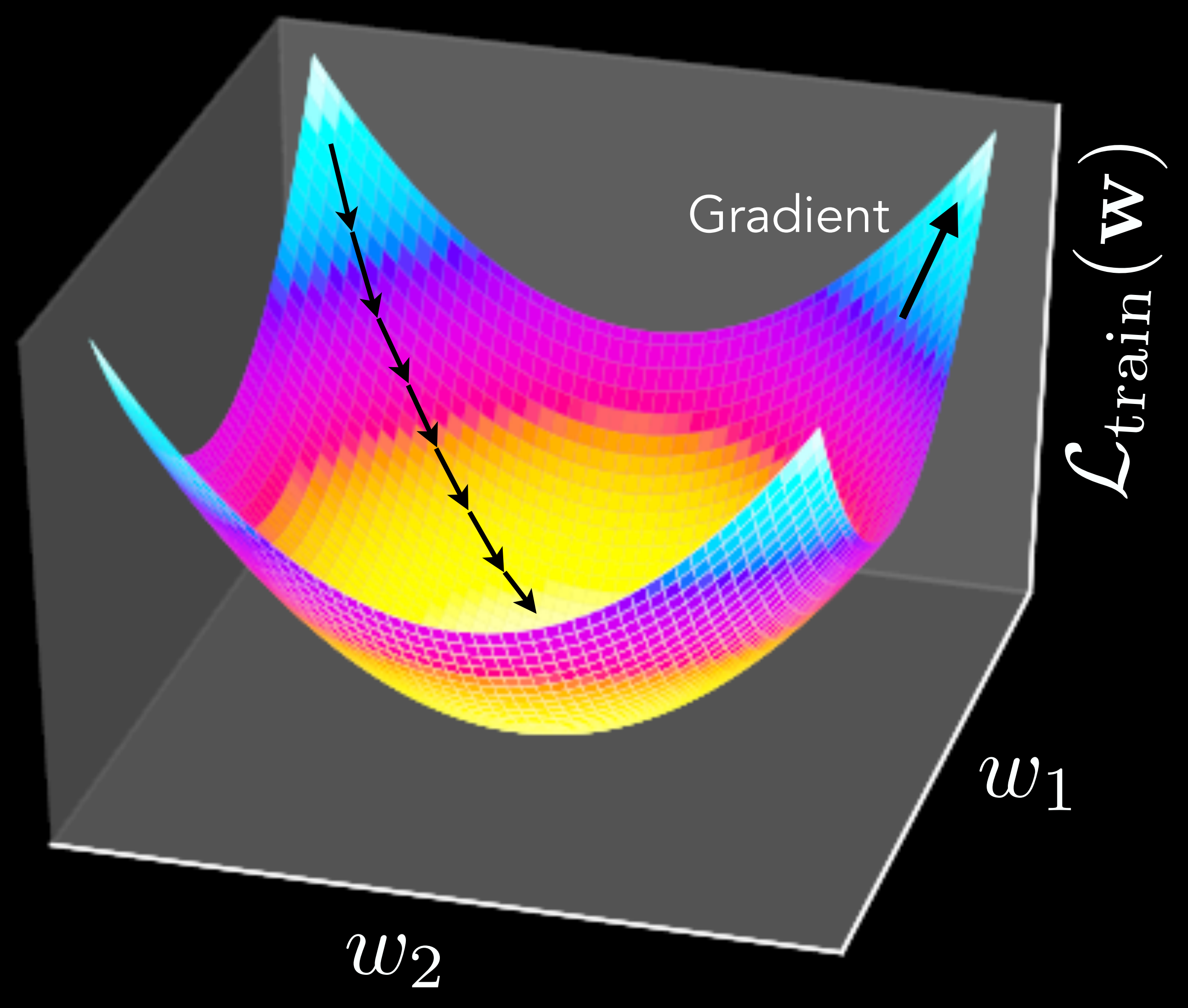
Optimal predictor

$$f_{\hat{\mathbf{W}}_1 \hat{\mathbf{W}}_2}(\mathbf{x}) = \hat{\mathbf{W}}_2 \sigma(\hat{\mathbf{W}}_1 \mathbf{x})$$

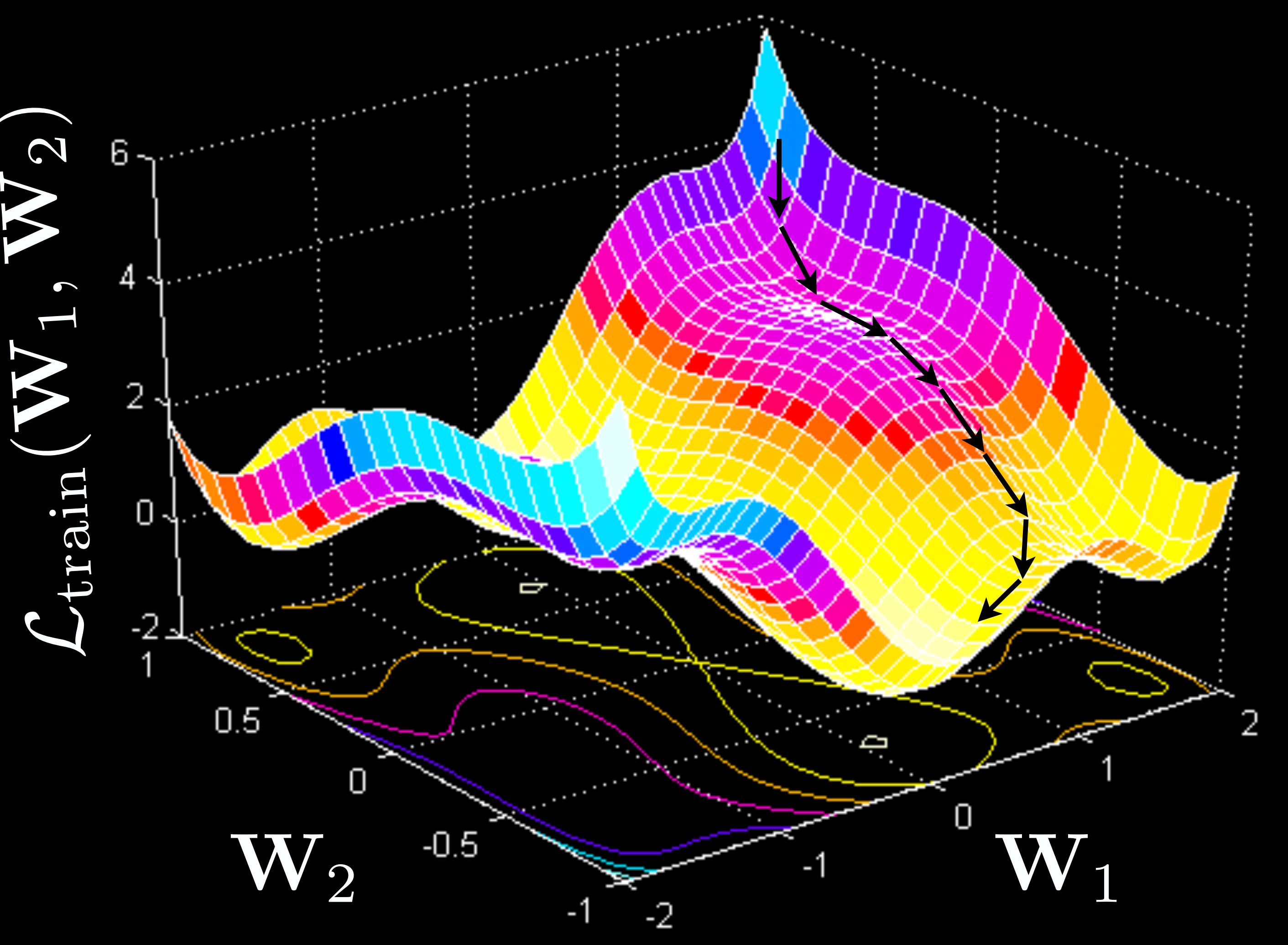
Non-convexity

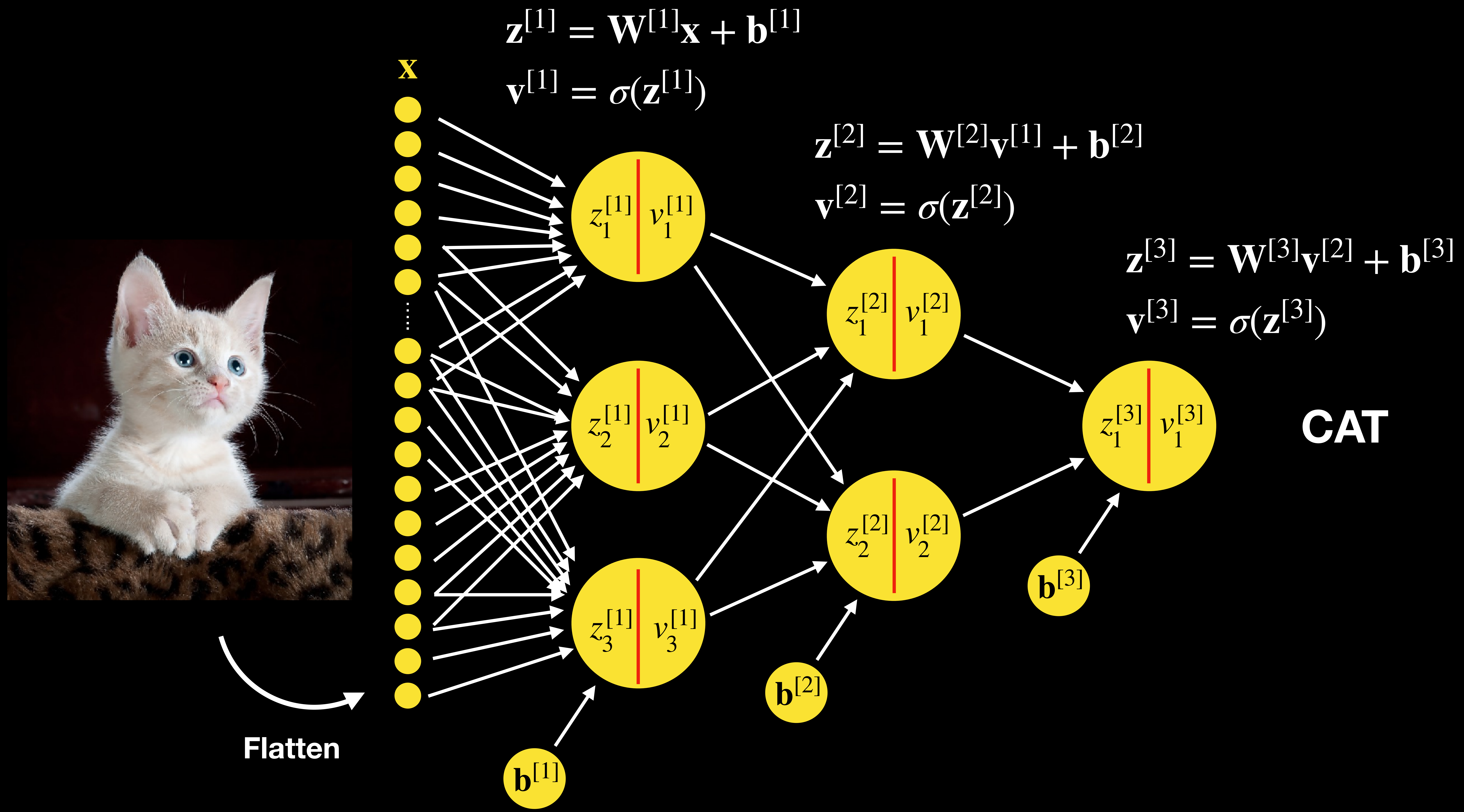
$$\hat{\mathbf{W}}_1, \hat{\mathbf{W}}_2 = \arg\min_{\mathbf{W}_1, \mathbf{W}_2} \mathscr{L}(\mathbf{W}_1, \mathbf{W}_2)$$

Linear predictor loss



Neural network loss





$\mathbf{z}^{[1]} = \mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]}$
$\mathbf{v}^{[1]} = \sigma(\mathbf{z}^{[1]})$
$\mathbf{z}^{[2]} = \mathbf{W}^{[2]}\mathbf{v}^{[1]} + \mathbf{b}^{[2]}$
$\mathbf{v}^{[2]} = \sigma(\mathbf{z}^{[2]})$
$\mathbf{z}^{[3]} = \mathbf{W}^{[3]}\mathbf{v}^{[2]} + \mathbf{b}^{[3]}$
$\mathbf{v}^{[3]} = \sigma(\mathbf{z}^{[3]})$
$\mathbf{x}$
$z_1^{[1]}$ $v_1^{[1]}$
$z_2^{[1]}$ $v_2^{[1]}$
$z_3^{[1]}$ $v_3^{[1]}$
$z_1^{[2]}$ $v_1^{[2]}$
$z_2^{[2]}$ $v_2^{[2]}$
$z_1^{[3]}$ $v_1^{[3]}$
$\mathbf{b}^{[1]}$
$\mathbf{b}^{[2]}$
$\mathbf{b}^{[3]}$
CAT
Flatten

Hypothesis

$$\mathbf{z}^{[1]} = \mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]}$$

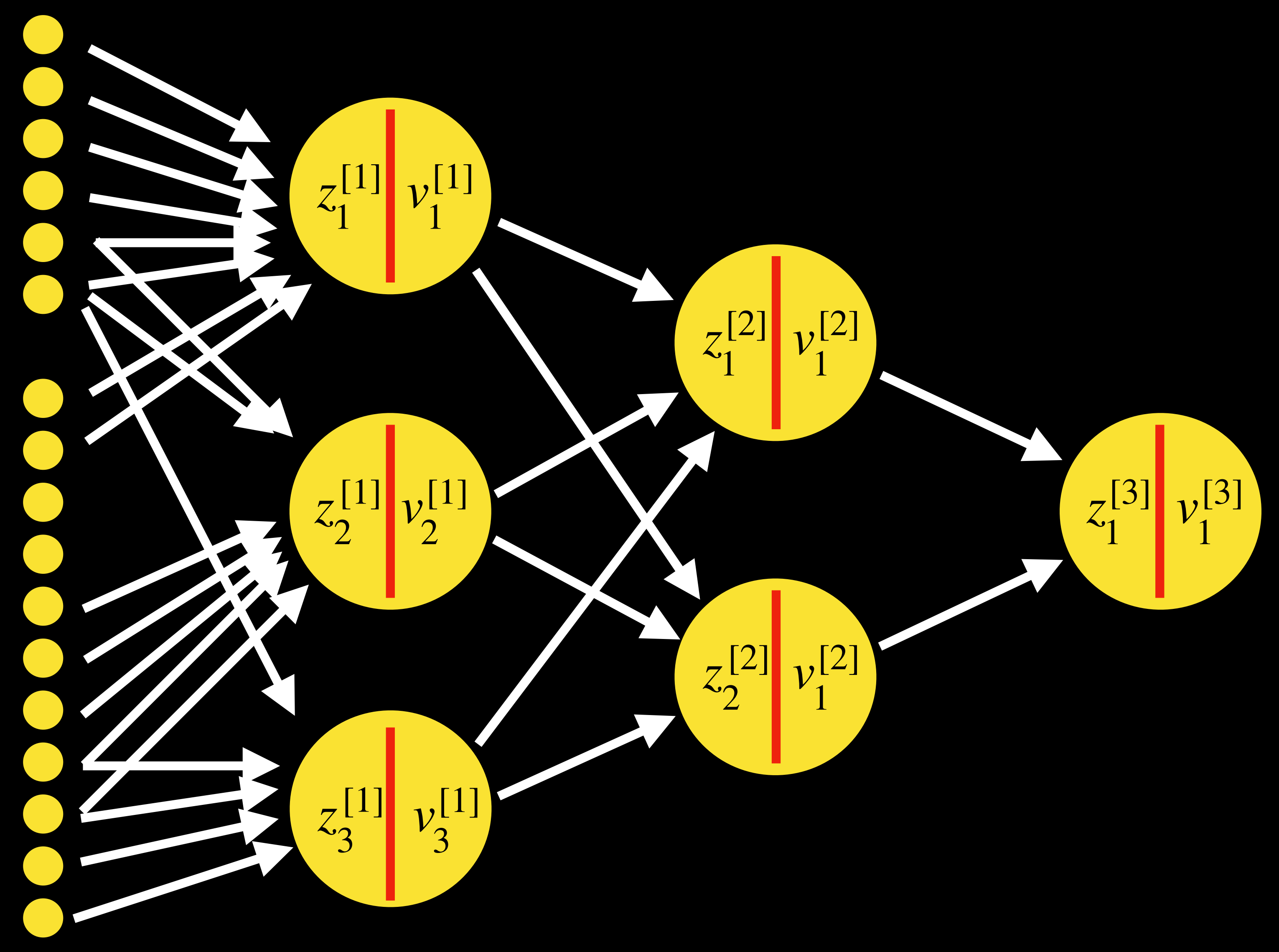
$$\mathbf{v}^{[1]} = \sigma(\mathbf{z}^{[1]})$$

$$\mathbf{z}^{[2]} = \mathbf{W}^{[2]}\mathbf{v}^{[1]} + \mathbf{b}^{[2]}$$

$$\mathbf{v}^{[2]} = \sigma(\mathbf{z}^{[2]})$$

$$\mathbf{z}^{[3]} = \mathbf{W}^{[3]}\mathbf{v}^{[2]} + \mathbf{b}^{[3]}$$

$$\mathbf{v}^{[3]} = \sigma(\mathbf{z}^{[3]})$$



$$\hat{y} \equiv \mathbf{v}^{[3]}$$

$$\hat{y} = f_{\mathbf{W}^{[1]}\mathbf{W}^{[2]}\mathbf{W}^{[3]}}(\mathbf{x})$$

Loss (Binary output): $$\mathscr{L}(\hat{y}, y) = -\sum_{i=1}^{n} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

Gradient Descent

Update the i^{th} layer:

$$\mathbf{W}^{[i]} = \mathbf{W}^{[i]} - \alpha \frac{\partial \mathscr{L}}{\partial \mathbf{W}^{[i]}}$$

$$\mathbf{b}^{[i]} = \mathbf{b}^{[i]} - \alpha \frac{\partial \mathscr{L}}{\partial \mathbf{b}^{[i]}}$$

What are the gradients?

Hypothesis

$$\mathbf{z}^{[1]} = \mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]}$$

$$\mathbf{v}^{[1]} = \sigma(\mathbf{z}^{[1]})$$

$$\mathbf{z}^{[2]} = \mathbf{W}^{[2]}\mathbf{v}^{[1]} + \mathbf{b}^{[2]}$$

$$\mathbf{v}^{[2]} = \sigma(\mathbf{z}^{[2]})$$

$$\mathbf{z}^{[3]} = \mathbf{W}^{[3]}\mathbf{v}^{[2]} + \mathbf{b}^{[3]}$$

$$\hat{y} = \sigma(\mathbf{z}^{[3]})$$

What are the gradients?

$$\mathscr{L}(\hat{y}, y) = -\sum_{i=1}^{n} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

$$\frac{\partial \mathscr{L}}{\partial \mathbf{W}^{[3]}} = \frac{\partial \mathscr{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{z}^{[3]}} \frac{\partial \mathbf{z}^{[3]}}{\partial \mathbf{W}^{[3]}}$$

$$\frac{\partial \mathscr{L}}{\partial \mathbf{W}^{[2]}} = \frac{\partial \mathscr{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{z}^{[3]}} \frac{\partial \mathbf{z}^{[3]}}{\partial \mathbf{v}^{[2]}} \frac{\partial \mathbf{v}^{[2]}}{\partial \mathbf{z}^{[2]}} \frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{W}^{[2]}}$$

$$\frac{\partial \mathscr{L}}{\partial \mathbf{b}^{[2]}} = \frac{\partial \mathscr{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{z}^{[3]}} \frac{\partial \mathbf{z}^{[3]}}{\partial \mathbf{v}^{[2]}} \frac{\partial \mathbf{v}^{[2]}}{\partial \mathbf{z}^{[2]}} \frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{b}^{[2]}}$$

Hypothesis

$$\mathbf{z}^{[1]} = \mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]}$$

$$\mathbf{v}^{[1]} = \sigma(\mathbf{z}^{[1]})$$

$$\mathbf{z}^{[2]} = \mathbf{W}^{[2]}\mathbf{v}^{[1]} + \mathbf{b}^{[2]}$$

$$\mathbf{v}^{[2]} = \sigma(\mathbf{z}^{[2]})$$

$$\mathbf{z}^{[3]} = \mathbf{W}^{[3]}\mathbf{v}^{[2]} + \mathbf{b}^{[3]}$$

$$\hat{y} = \sigma(\mathbf{z}^{[3]})$$

What are the gradients?

$$\mathscr{L}(\hat{y}, y) = -\sum_{i=1}^{n} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

$$\frac{\partial \mathscr{L}}{\partial \mathbf{W}^{[3]}} = \frac{\partial \mathscr{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{z}^{[3]}} \frac{\partial \mathbf{z}^{[3]}}{\partial \mathbf{W}^{[3]}}$$

$$\frac{\partial \mathscr{L}}{\partial \hat{y}} = -y^{(i)} \frac{1}{\hat{y}^{(i)}} + (1 - y^{(i)}) \frac{1}{1 - \hat{y}^{(i)}}$$

$$\frac{\partial \hat{y}}{\partial \mathbf{z}^{[3]}} = \sigma'(\mathbf{z}^{[3]}) = \sigma(\mathbf{z}^{[3]})(1 - \sigma(\mathbf{z}^{[3]}))$$

$$\frac{\partial \mathbf{z}^{[3]}}{\partial \mathbf{W}^{[3]}} = \mathbf{v}^{[2]^\top}$$

Hypothesis

$$\mathbf{z}^{[1]} = \mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]}$$

$$\mathbf{v}^{[1]} = \sigma(\mathbf{z}^{[1]})$$

$$\mathbf{z}^{[2]} = \mathbf{W}^{[2]}\mathbf{v}^{[1]} + \mathbf{b}^{[2]}$$

$$\mathbf{v}^{[2]} = \sigma(\mathbf{z}^{[2]})$$

$$\mathbf{z}^{[3]} = \mathbf{W}^{[3]}\mathbf{v}^{[2]} + \mathbf{b}^{[3]}$$

$$\hat{y} = \sigma(\mathbf{z}^{[3]})$$

$$\frac{\partial \mathscr{L}}{\partial \mathbf{W}^{[3]}} = \frac{\partial \mathscr{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{z}^{[3]}} \frac{\partial \mathbf{z}^{[3]}}{\partial \mathbf{W}^{[3]}}$$

$$\frac{\partial \mathscr{L}}{\partial \hat{y}} = -y^{(i)} \frac{1}{\hat{y}^{(i)}} + (1 - y^{(i)}) \frac{1}{1 - \hat{y}^{(i)}}$$

$$\frac{\partial \hat{y}}{\partial \mathbf{z}^{[3]}} = \sigma'(\mathbf{z}^{[3]}) = \sigma(\mathbf{z}^{[3]})(1 - \sigma(\mathbf{z}^{[3]}))$$

$$\frac{\partial \mathbf{z}^{[3]}}{\partial \mathbf{W}^{[3]}} = \mathbf{v}^{[2]^\top}$$

$$\vdots$$

$$\frac{\partial \mathscr{L}}{\partial \mathbf{W}^{[3]}} = (y^{(i)} - \hat{y}^{(i)}) \mathbf{v}^{[2]^\top}$$

SGD Update

$$\mathbf{W}^{[3]} = \mathbf{W}^{[3]} - \alpha \frac{\partial \mathscr{L}}{\partial \mathbf{W}^{[3]}}$$

Hypothesis

$$\mathbf{z}^{[1]} = \mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]}$$
$$\mathbf{v}^{[1]} = \sigma(\mathbf{z}^{[1]})$$

$$\mathbf{z}^{[2]} = \mathbf{W}^{[2]}\mathbf{v}^{[1]} + \mathbf{b}^{[2]}$$
$$\mathbf{v}^{[2]} = \sigma(\mathbf{z}^{[2]})$$

$$\mathbf{z}^{[3]} = \mathbf{W}^{[3]}\mathbf{v}^{[2]} + \mathbf{b}^{[3]}$$
$$\hat{y} = \sigma(\mathbf{z}^{[3]})$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[3]}} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{z}^{[3]}} \frac{\partial \mathbf{z}^{[3]}}{\partial \mathbf{W}^{[3]}} = (y^{(i)} - \hat{y}^{(i)})\mathbf{v}^{[2]^\top}$$

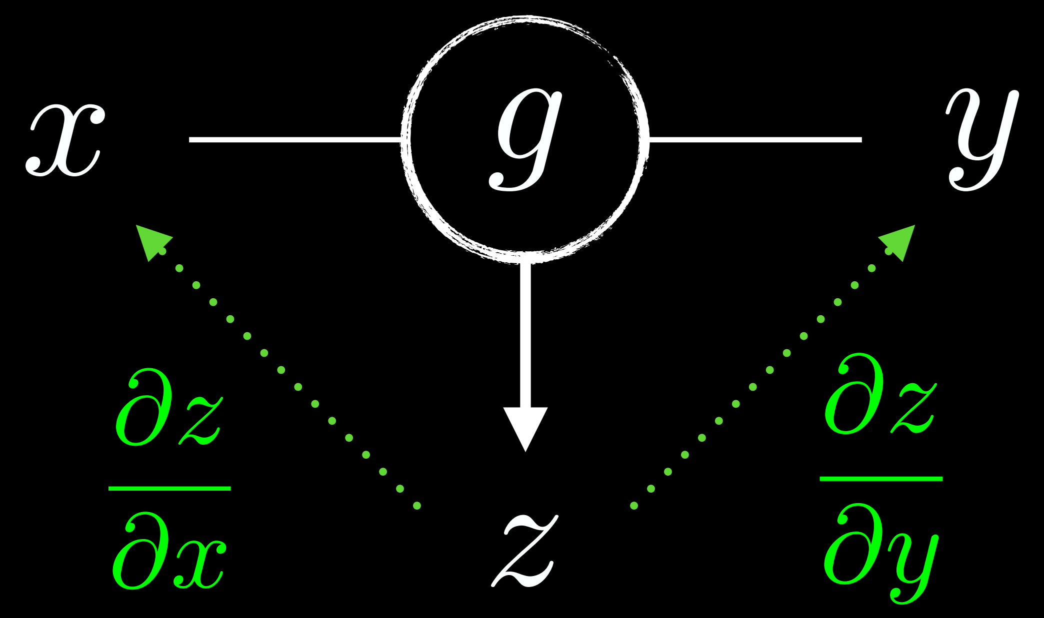
$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[2]}} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{z}^{[3]}} \frac{\partial \mathbf{z}^{[3]}}{\partial \mathbf{v}^{[2]}} \frac{\partial \mathbf{v}^{[2]}}{\partial \mathbf{z}^{[2]}} \frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{W}^{[2]}}$$

common terms across gradients

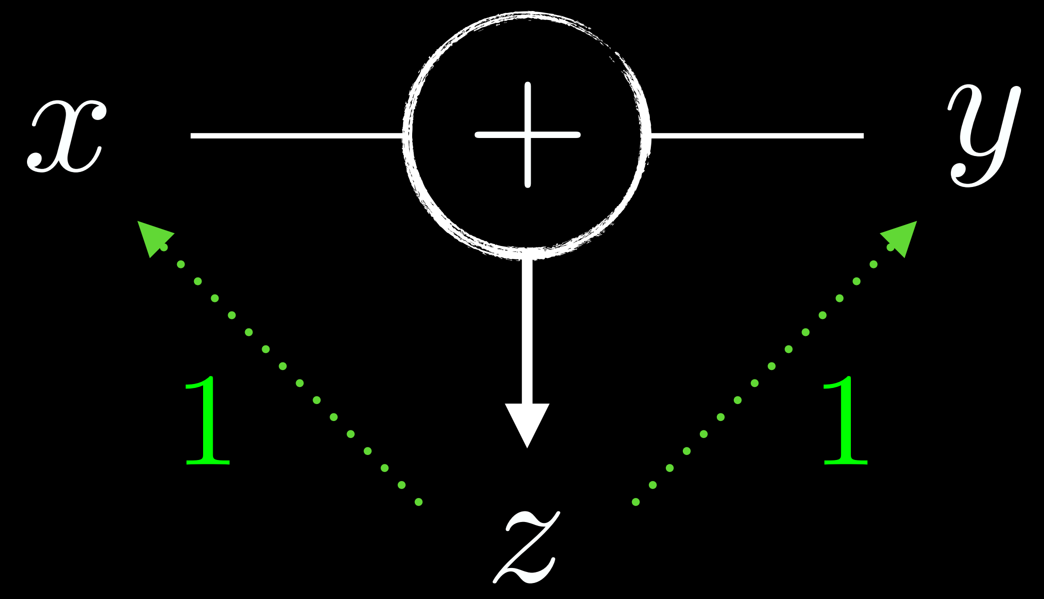
Can we save the gradients to be reused for computing other gradients?

Computation graphs

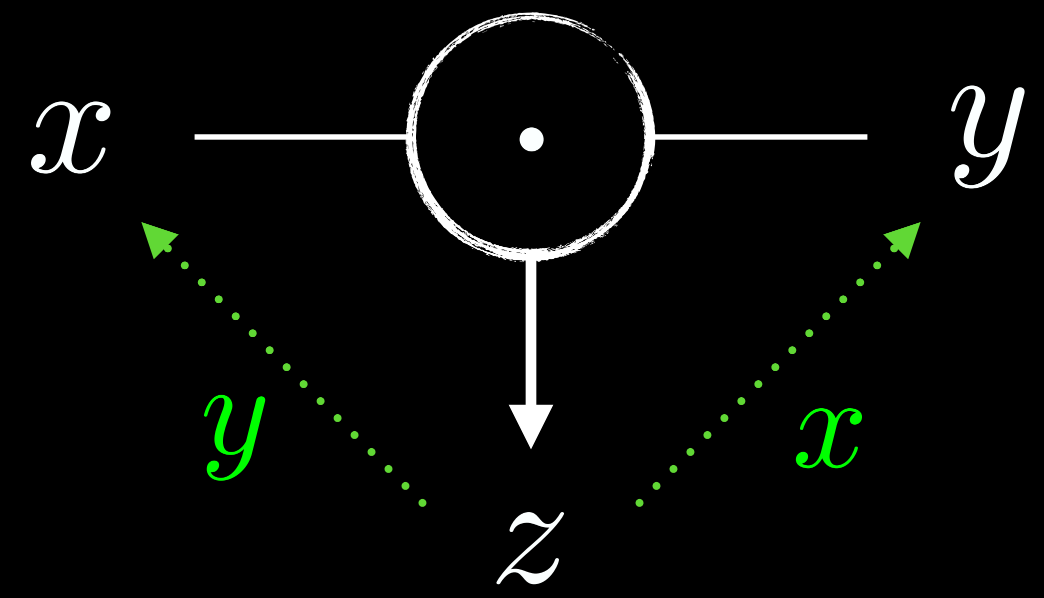
$$z = g(x, y)$$



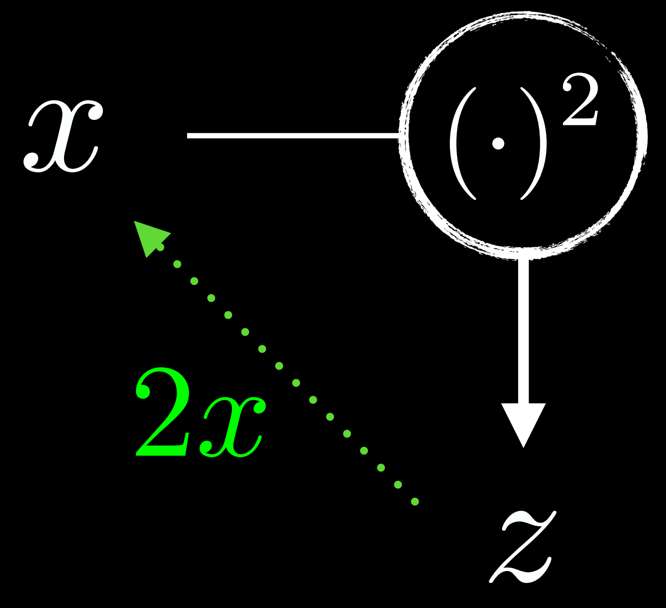
$$z = x + y$$



$$z = xy$$

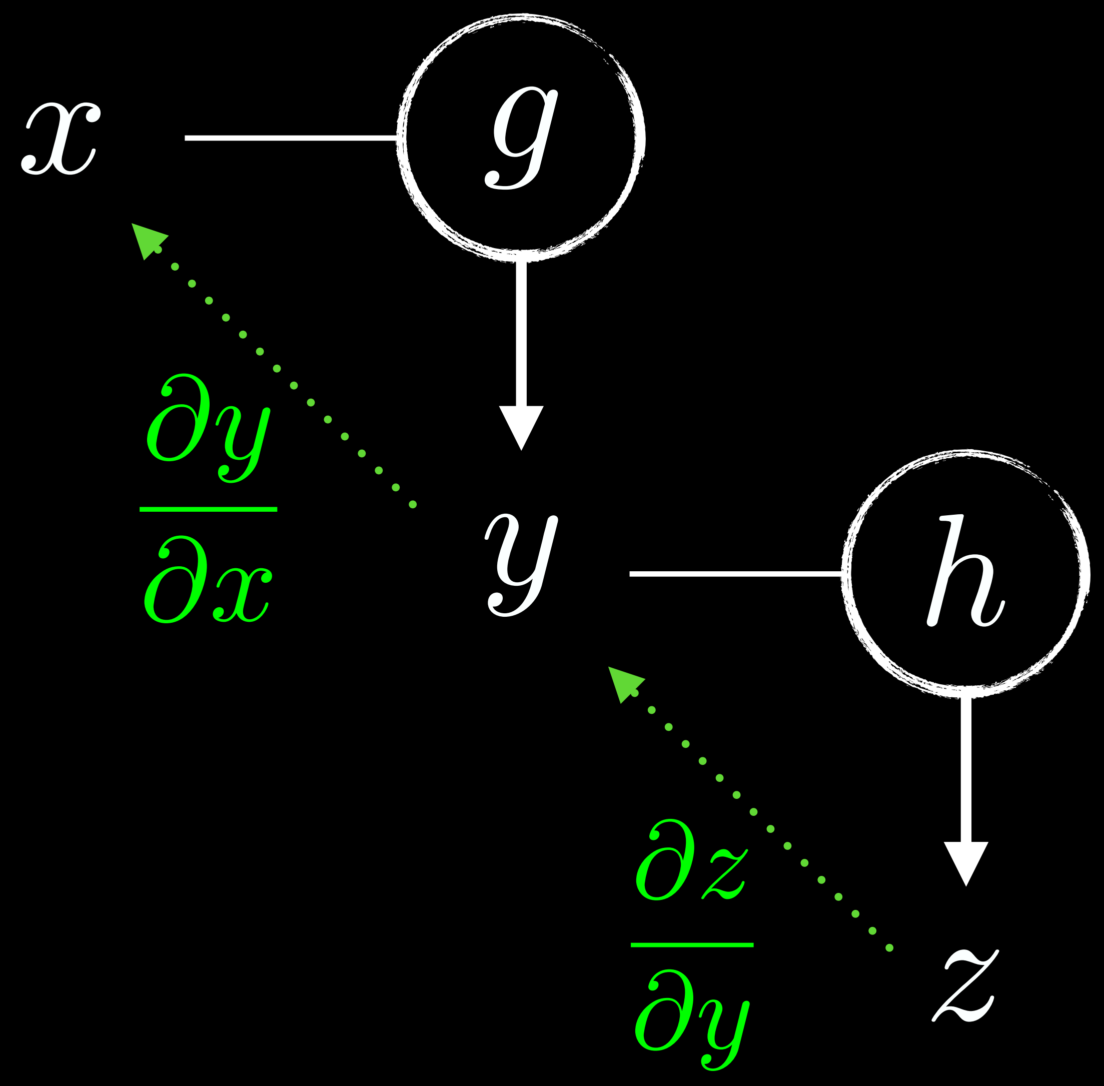


$$z = x^2$$



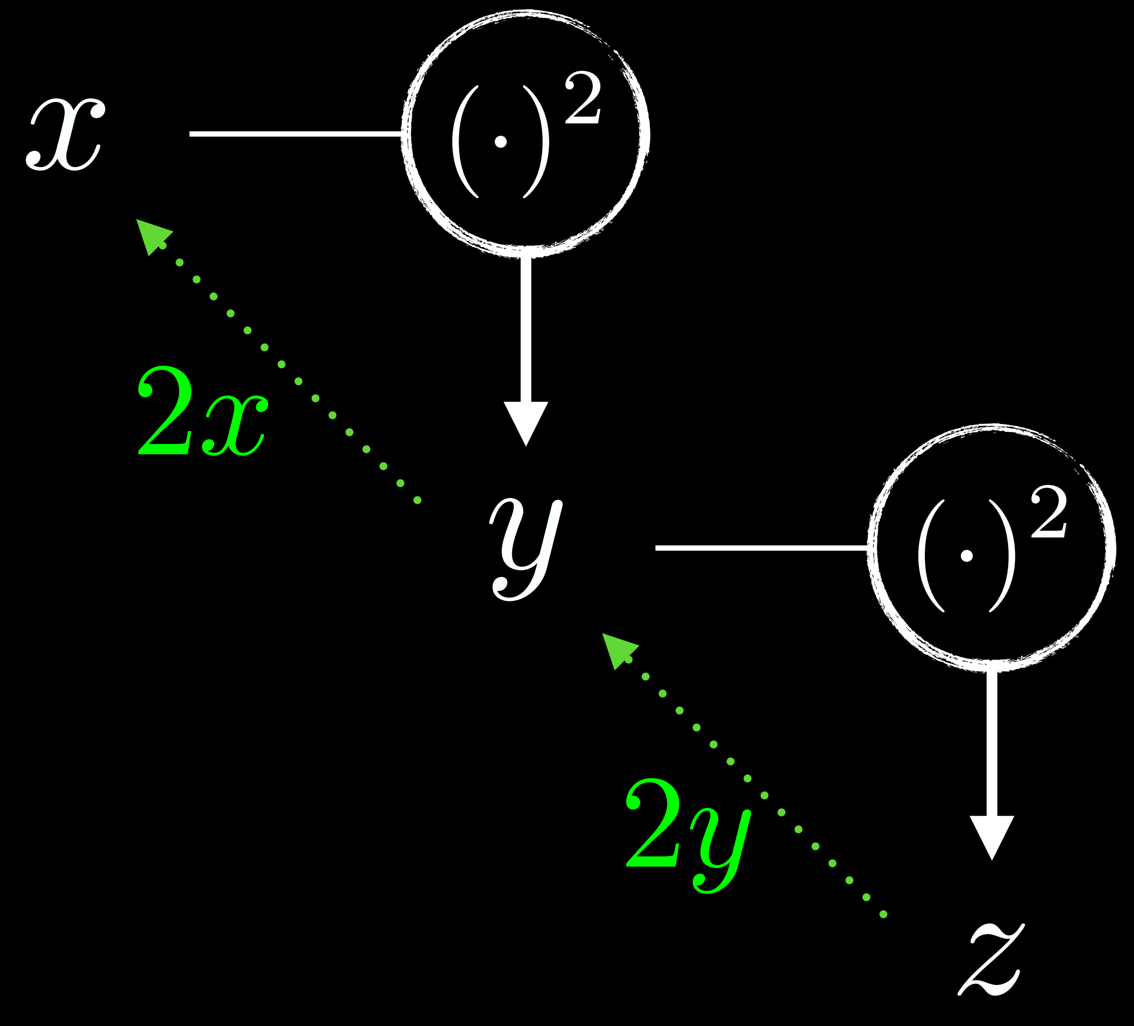
Chain rule

$$y = g(x)$$
$$z = h(y)$$



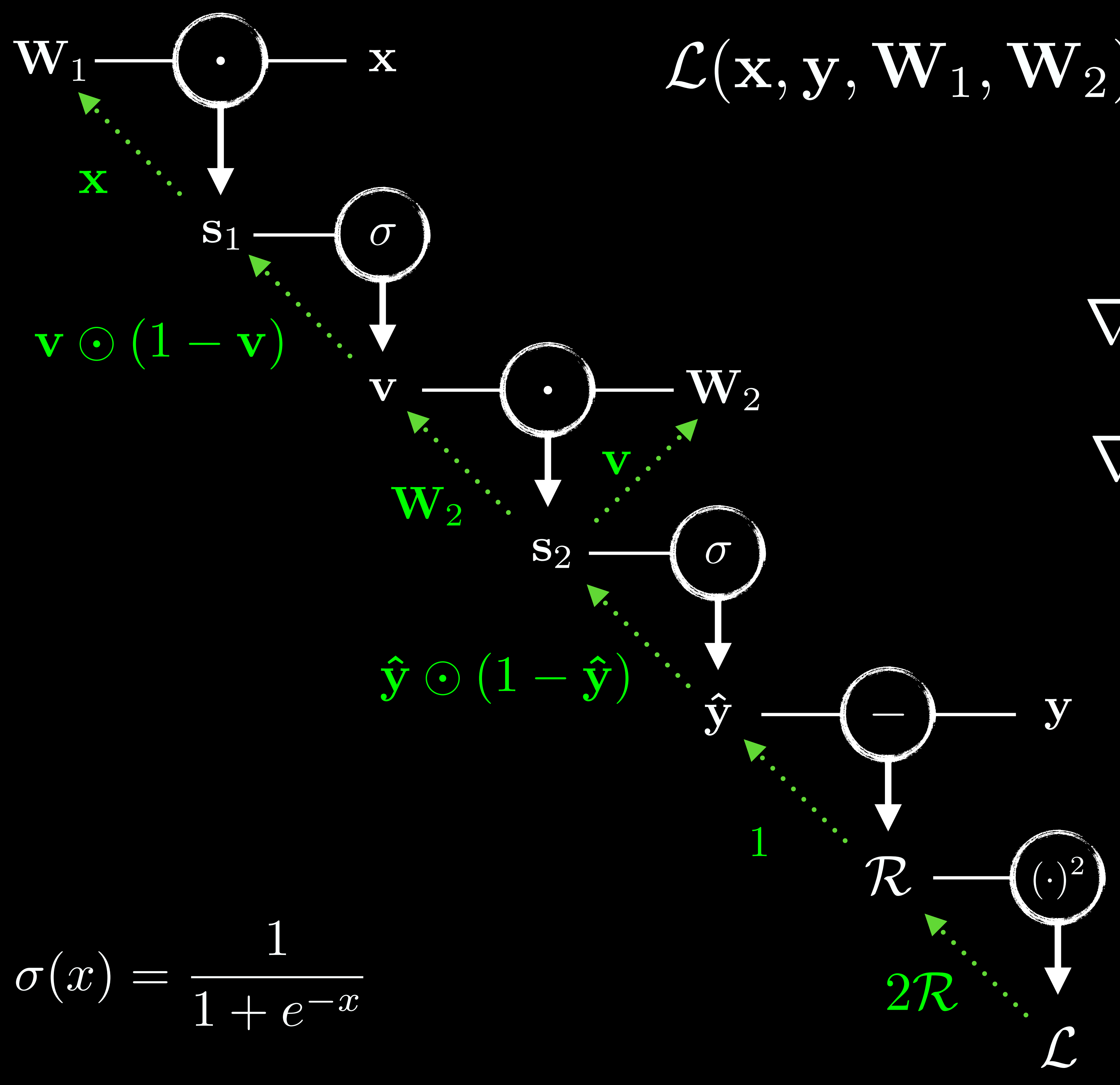
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

$$y = x^2$$
$$z = y^2$$



$$\frac{\partial z}{\partial x} = 4xy = 4x^3$$

Backpropagation



$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{W}_1, \mathbf{W}_2) = \|\sigma(\mathbf{W}_2\,\sigma(\mathbf{W}_1\mathbf{x})) - \mathbf{y}\|^2$$

$$\nabla_{\mathbf{W}_1}\mathcal{L} = 2\mathbf{W}_2^\top \mathcal{R} \odot \hat{\mathbf{y}} \odot (1 - \hat{\mathbf{y}}) \odot \mathbf{v} \odot (1 - \mathbf{v})\mathbf{x}^\top$$

$$\nabla_{\mathbf{W}_2}\mathcal{L} = 2\mathcal{R} \odot \hat{\mathbf{y}} \odot (1 - \hat{\mathbf{y}})\mathbf{v}^\top$$

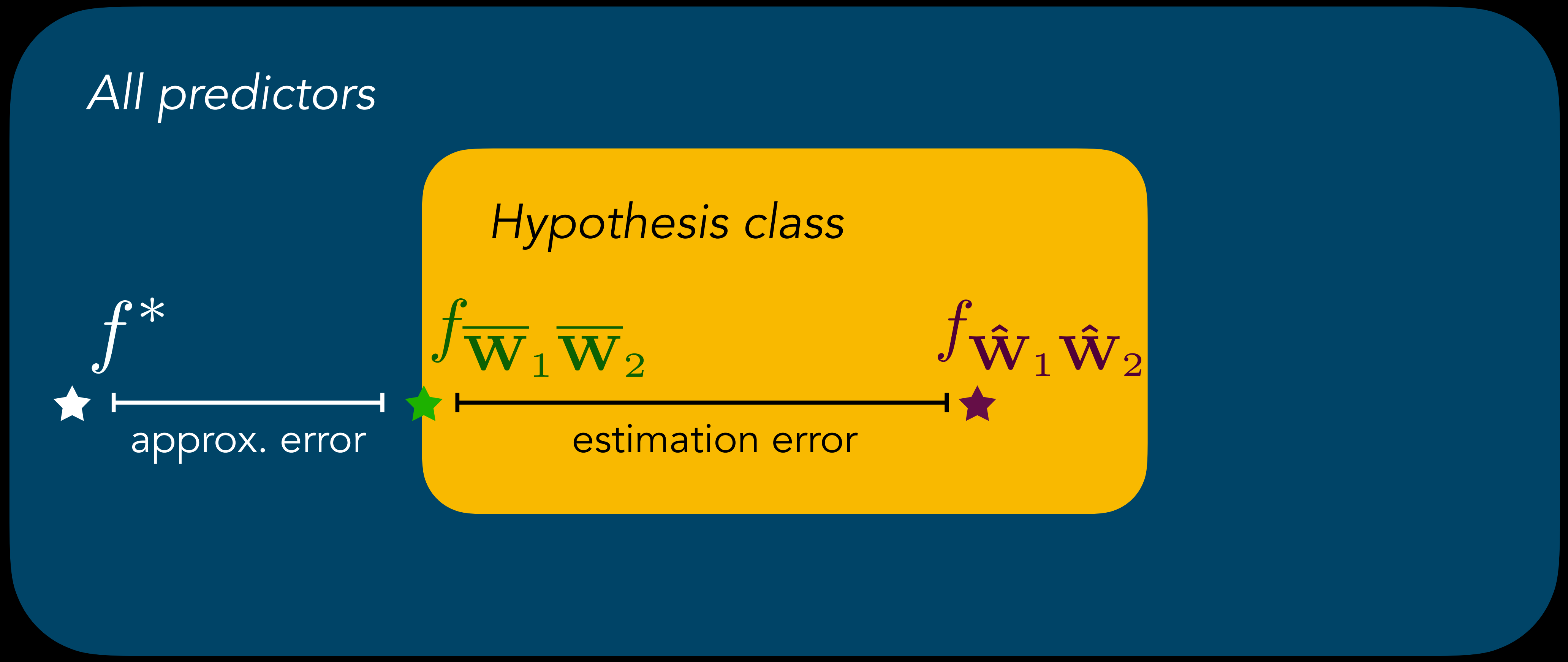
assuming $\sigma(x) = \dfrac{1}{1 + e^{-x}}$

Advanced Deep Learning

Hypothesis Class

$$f_{\mathbf{W}_1\mathbf{W}_2}(\mathbf{x}) = \sigma\left(\mathbf{W}_2\ \sigma\left(\mathbf{W}_1\mathbf{x}\right)\right)$$

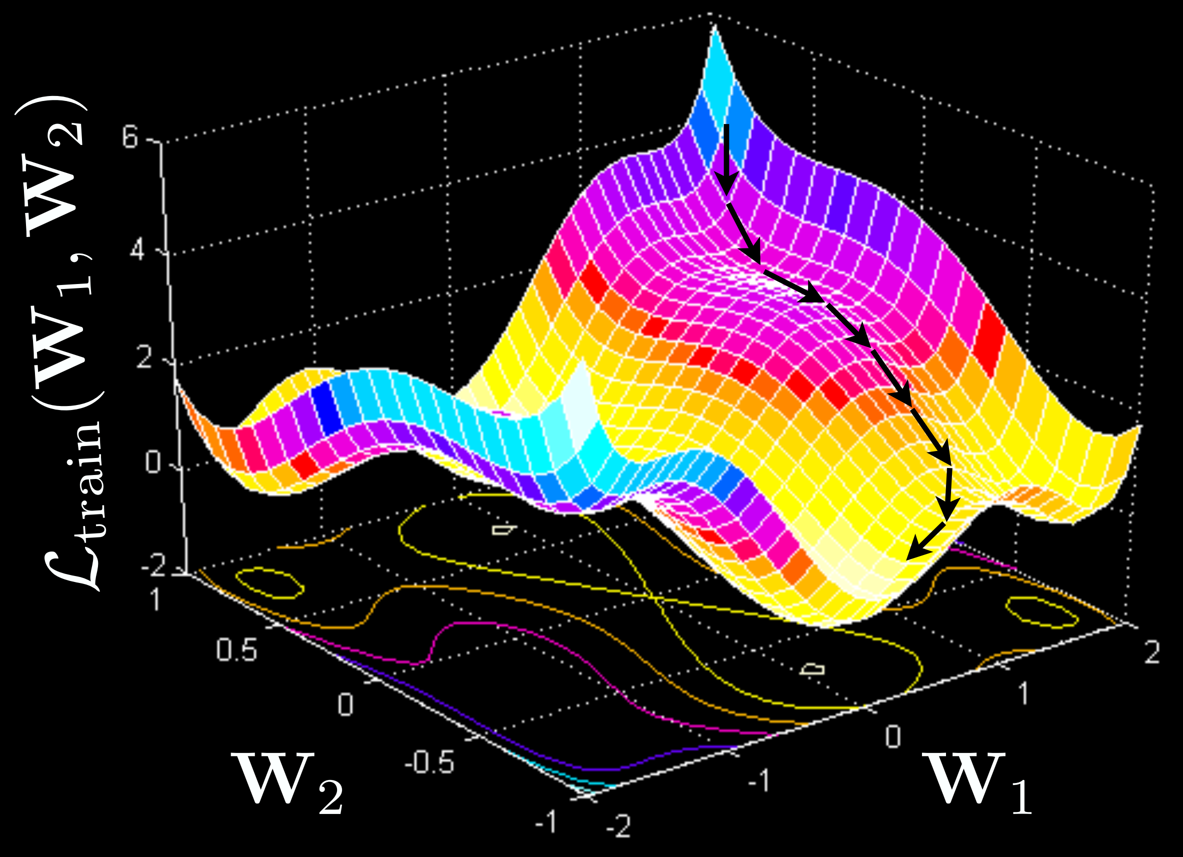
$$\mathcal{F} = \left\{ f_{\mathbf{W}_1\mathbf{W}_2}(\mathbf{x}) \mid \mathbf{W}_1 \in \mathbb{R}^{k\times n}, \mathbf{W}_2 \in \mathbb{R}^{m\times k} \right\}$$



Hyperparameters

How do you train a deep network?

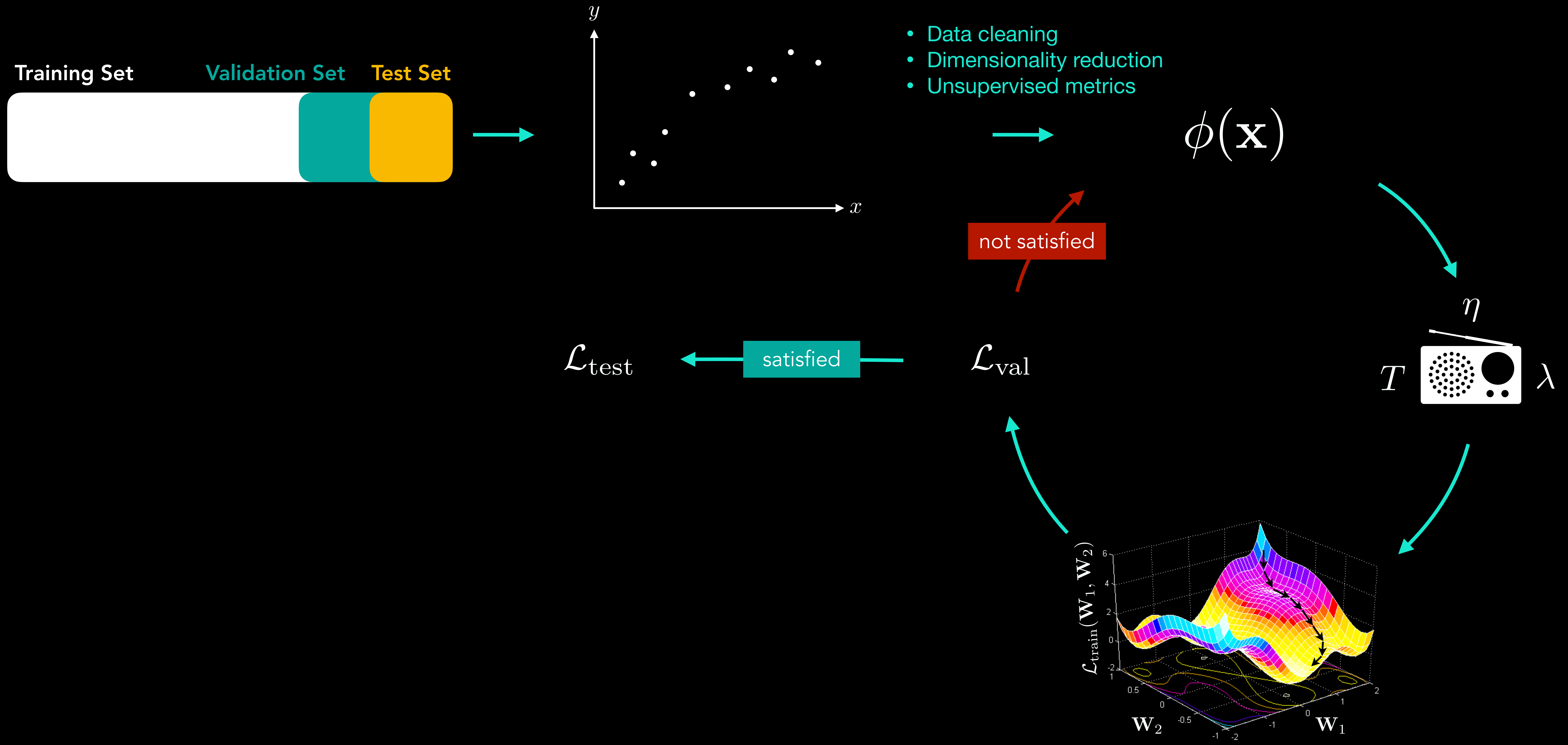
- Use many hidden layers for abstraction
- Use adaptive time steps
- Use hyper-parameter optimization



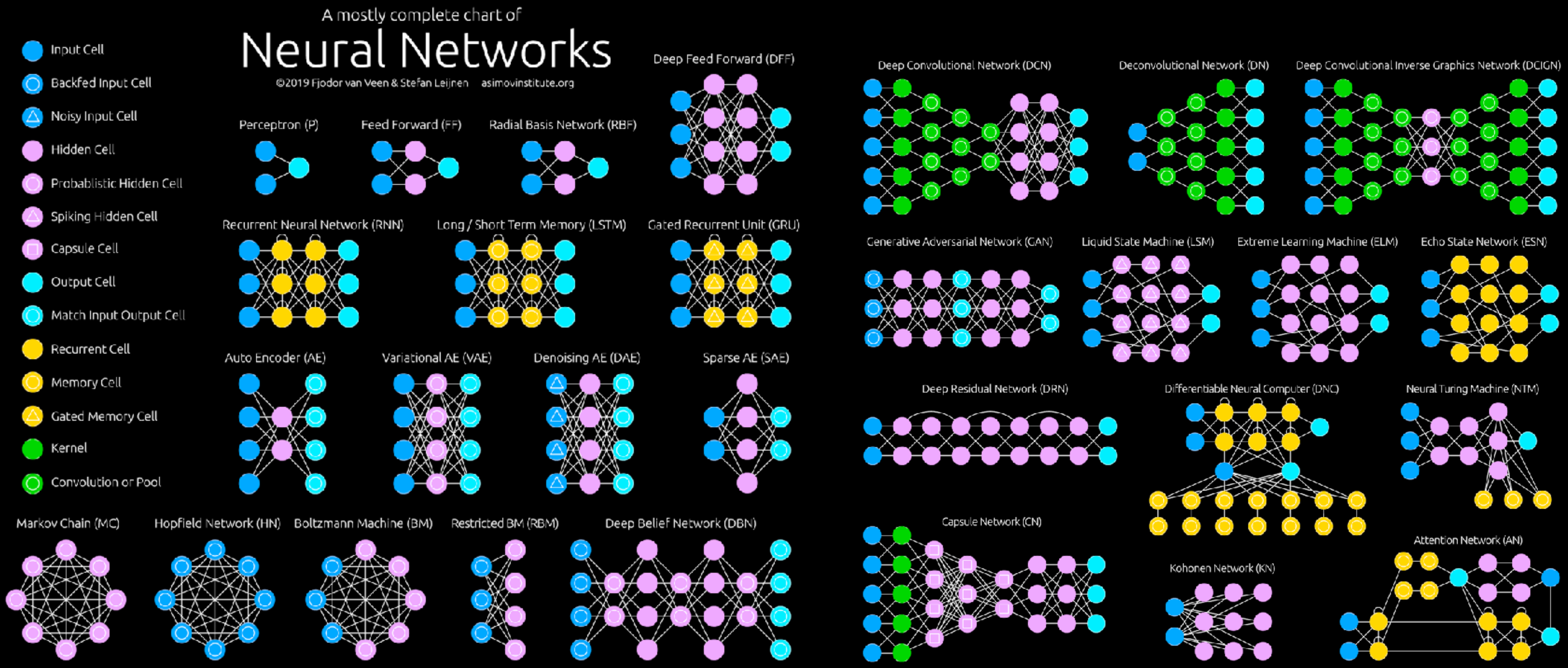
https://playground.tensorflow.org/

http://vision.stanford.edu/teaching/cs231n-demos/linear-classify/

The ML workflow

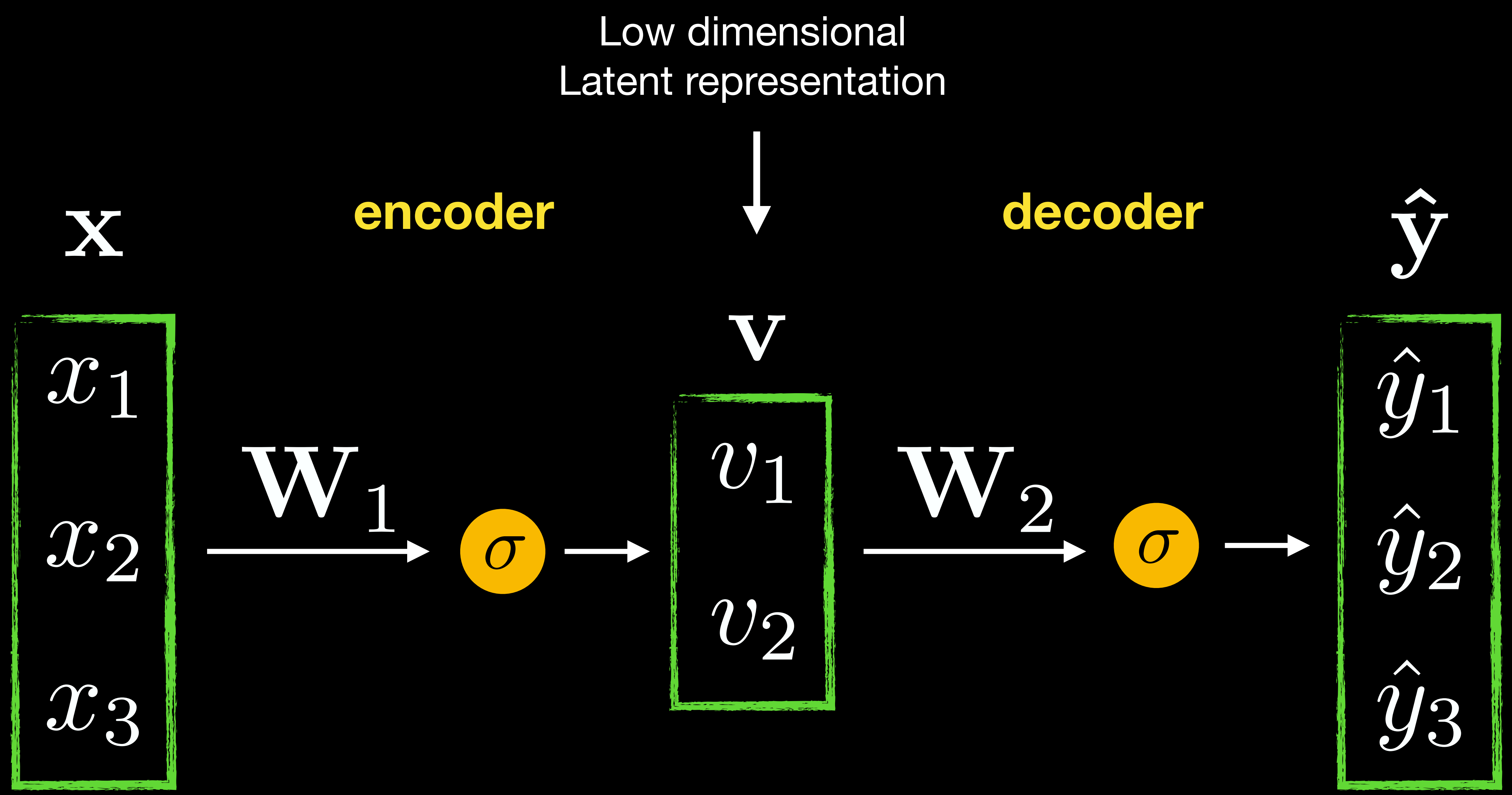


Neural network zoo



asimovinstitute.org

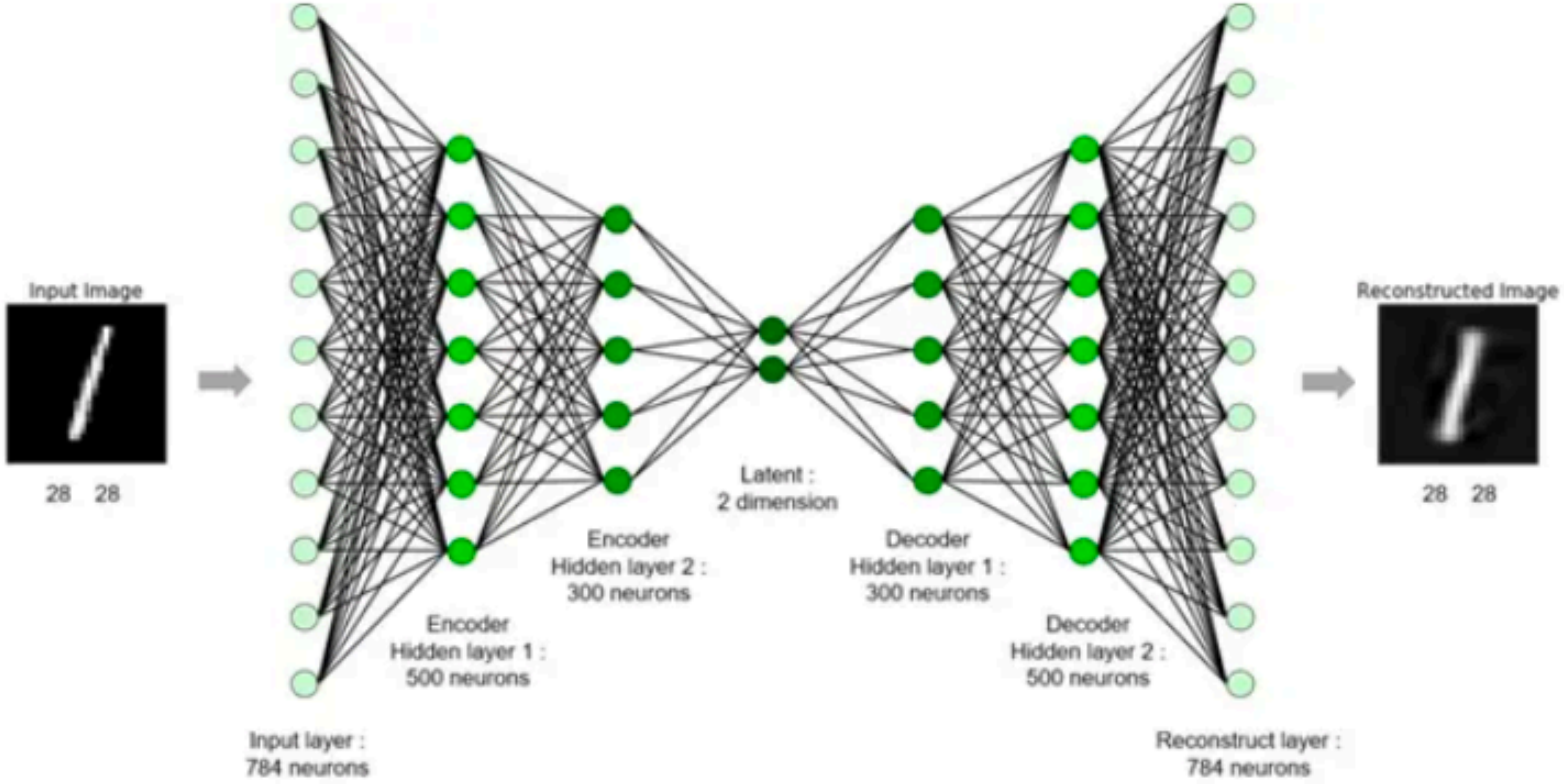
Auto-encoders

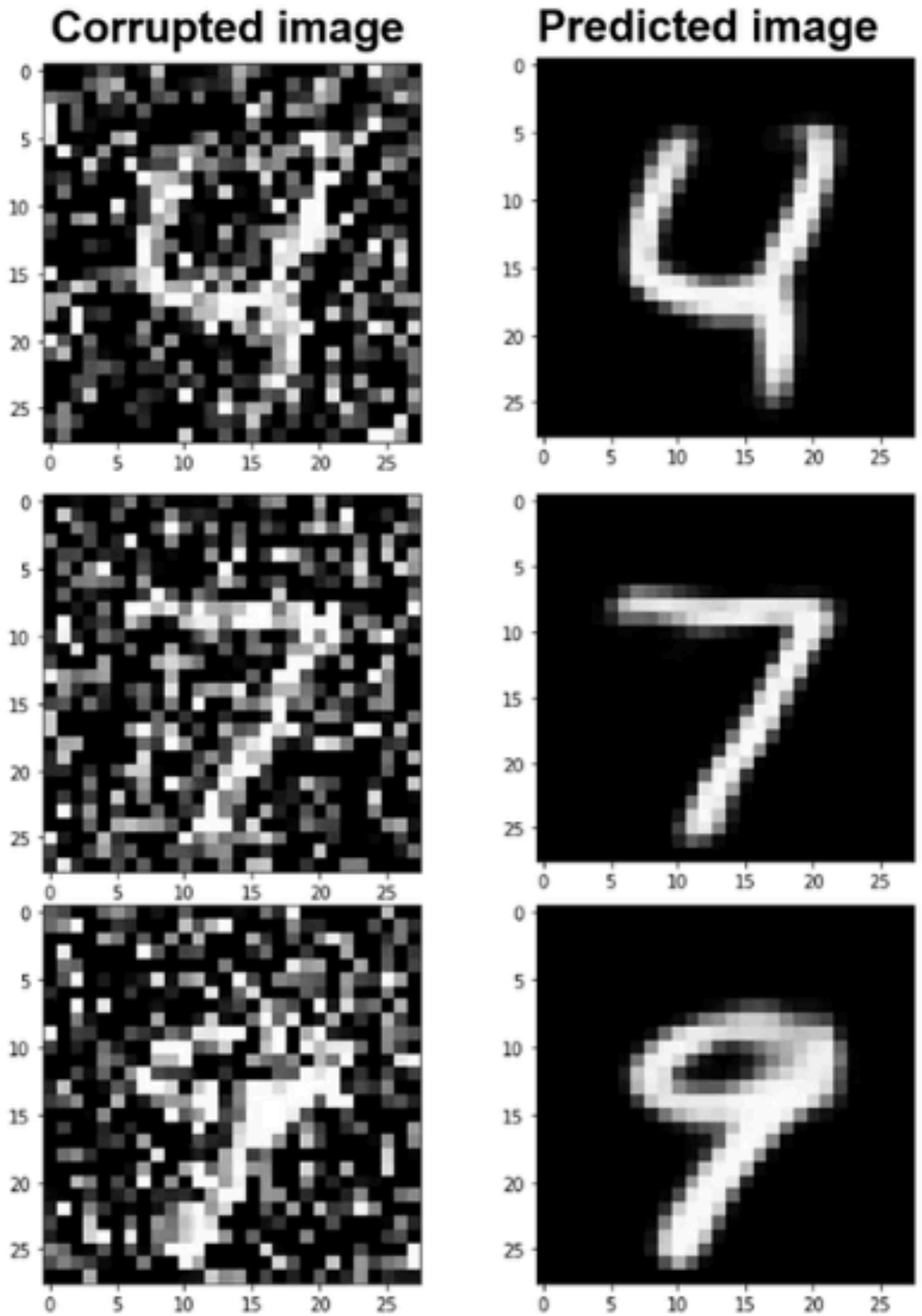


$$\mathcal{L} = \|f_{\mathbf{W}_1\mathbf{W}_2}(\mathbf{x}) - \mathbf{x}\|^2$$

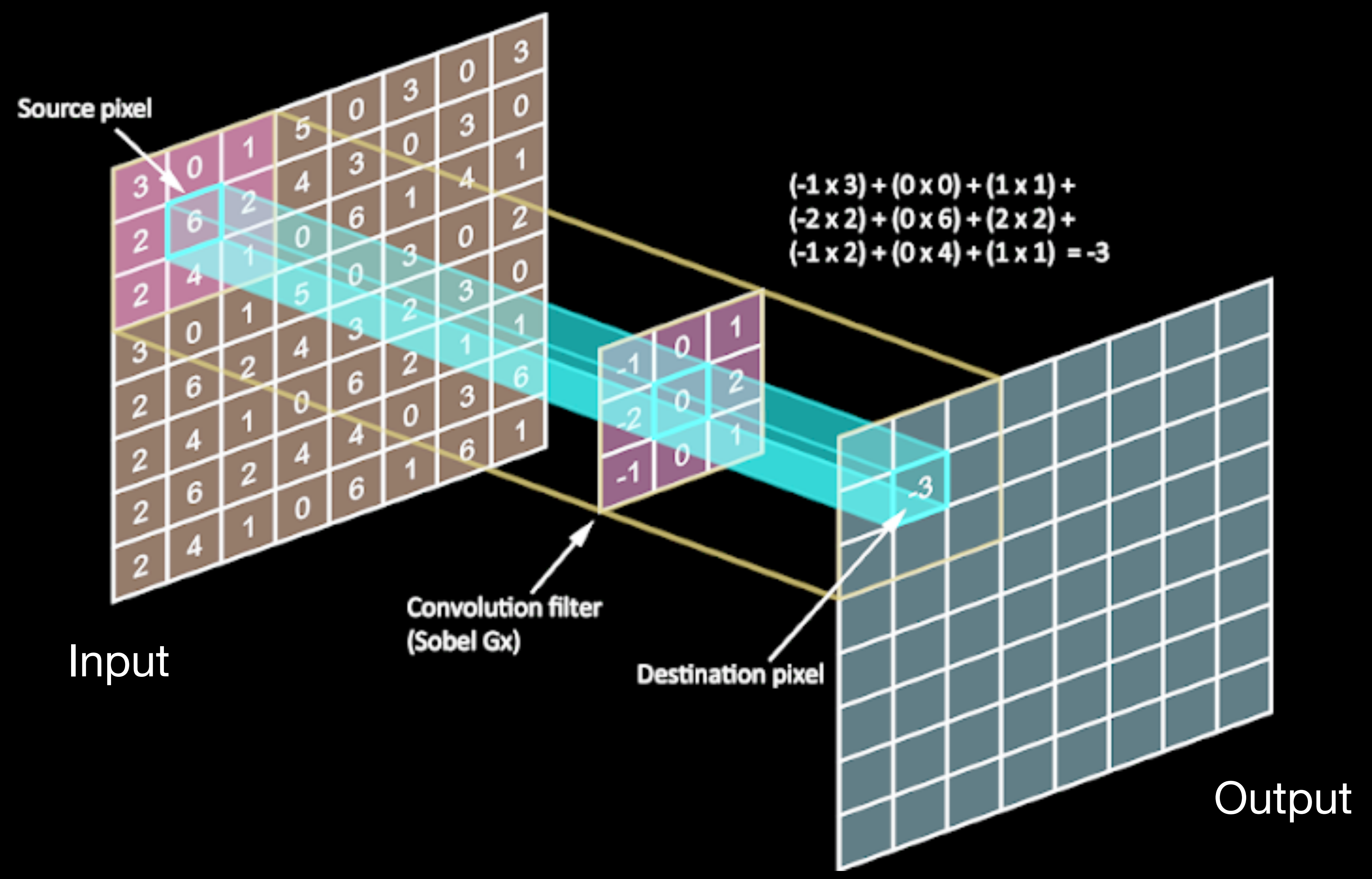
if $\sigma = I$, network performs SVD decomposition

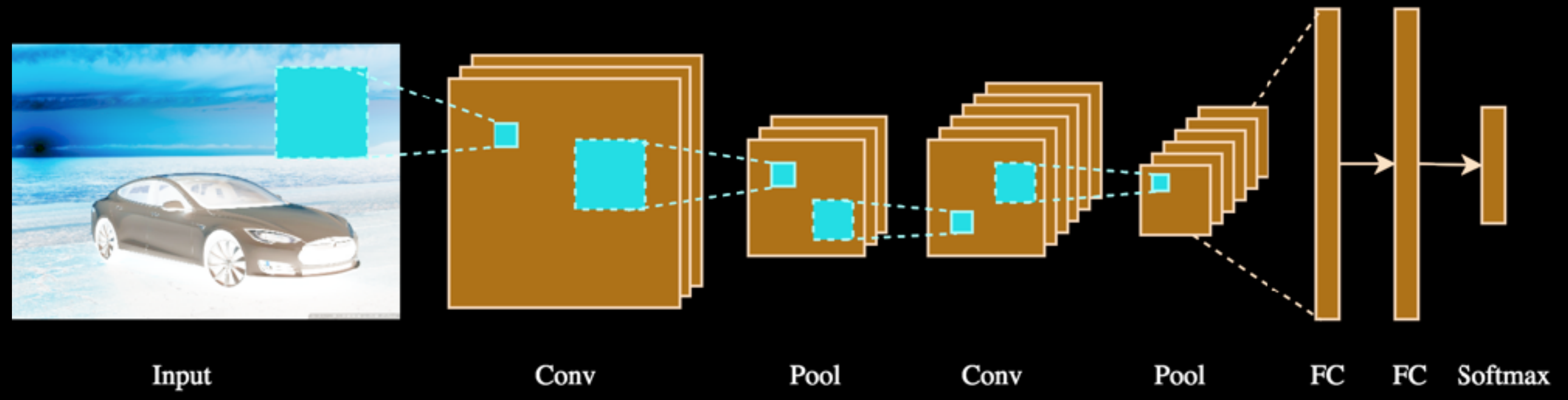
$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$$


Input Image
28 28
Input layer :
784 neurons
Encoder
Hidden layer 1 :
500 neurons
Encoder
Hidden layer 2 :
300 neurons
Latent :
2 dimension
Decoder
Hidden layer 1 :
300 neurons
Decoder
Hidden layer 2 :
500 neurons
Reconstruct layer :
784 neurons
Reconstructed Image
28 28


Corrupted image
Predicted image

Convolutional neural networks





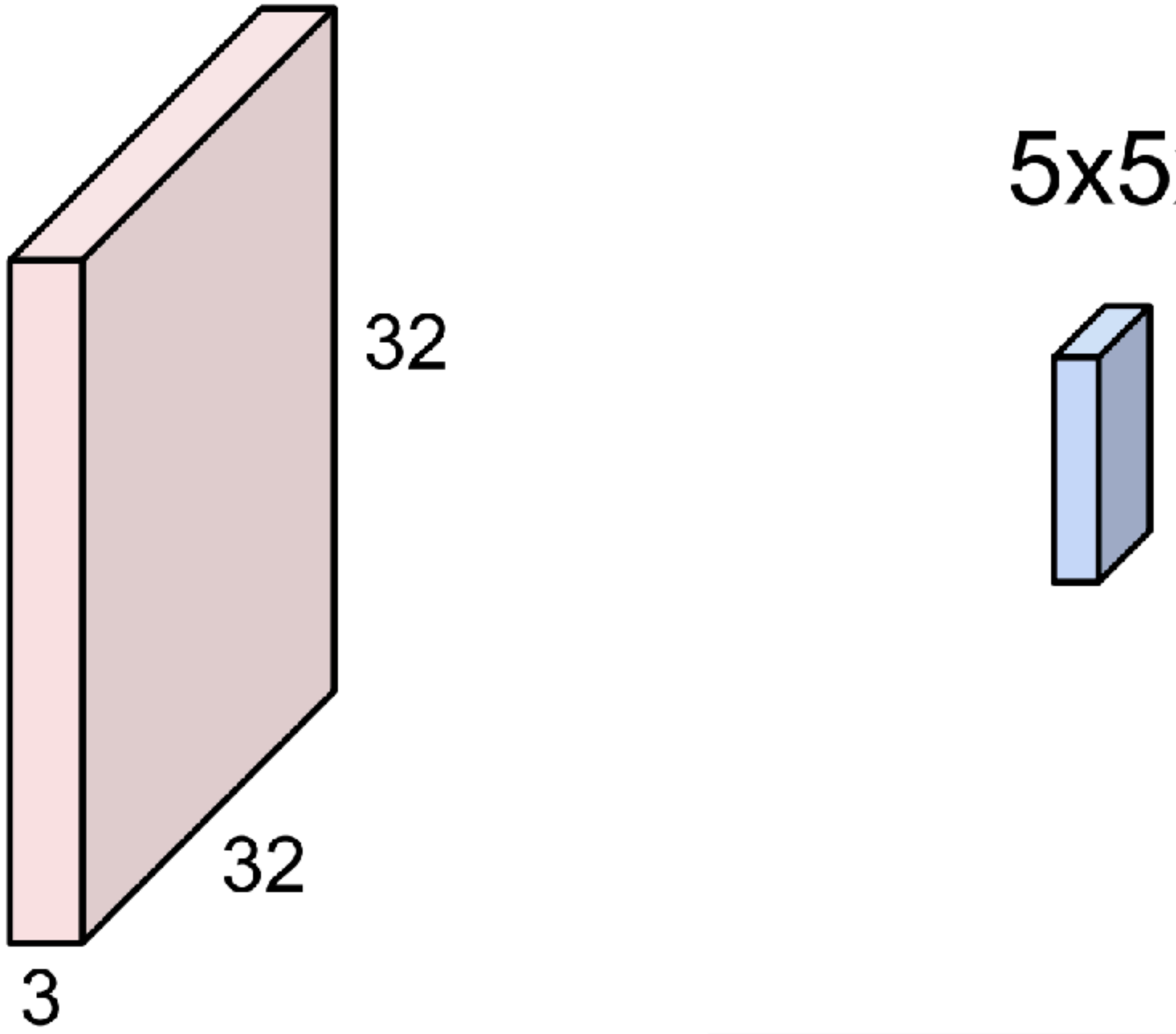
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Convolution Layer

Filters always extend the full depth of the input volume

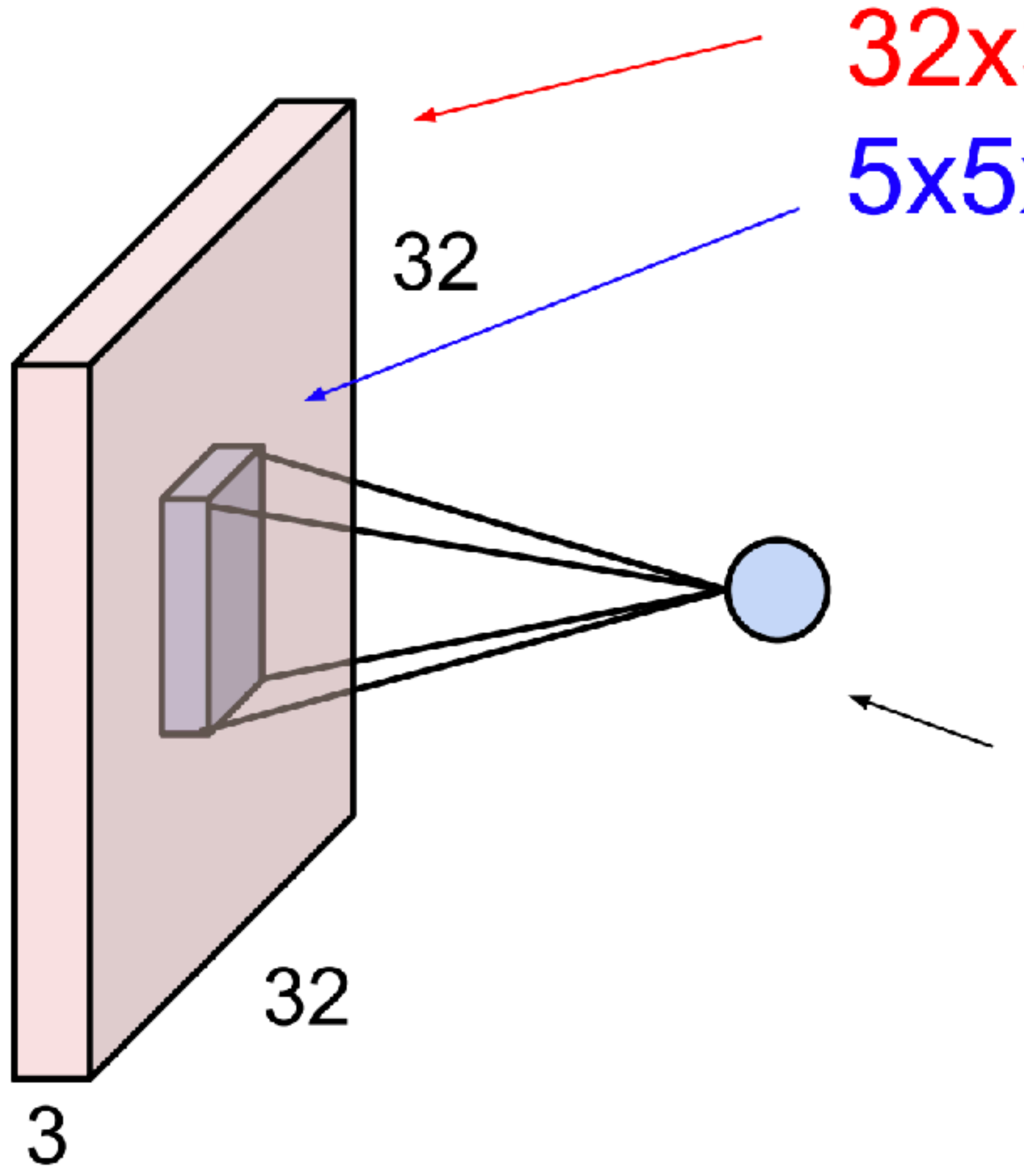
32x32x3 image

5x5x3 filter



Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

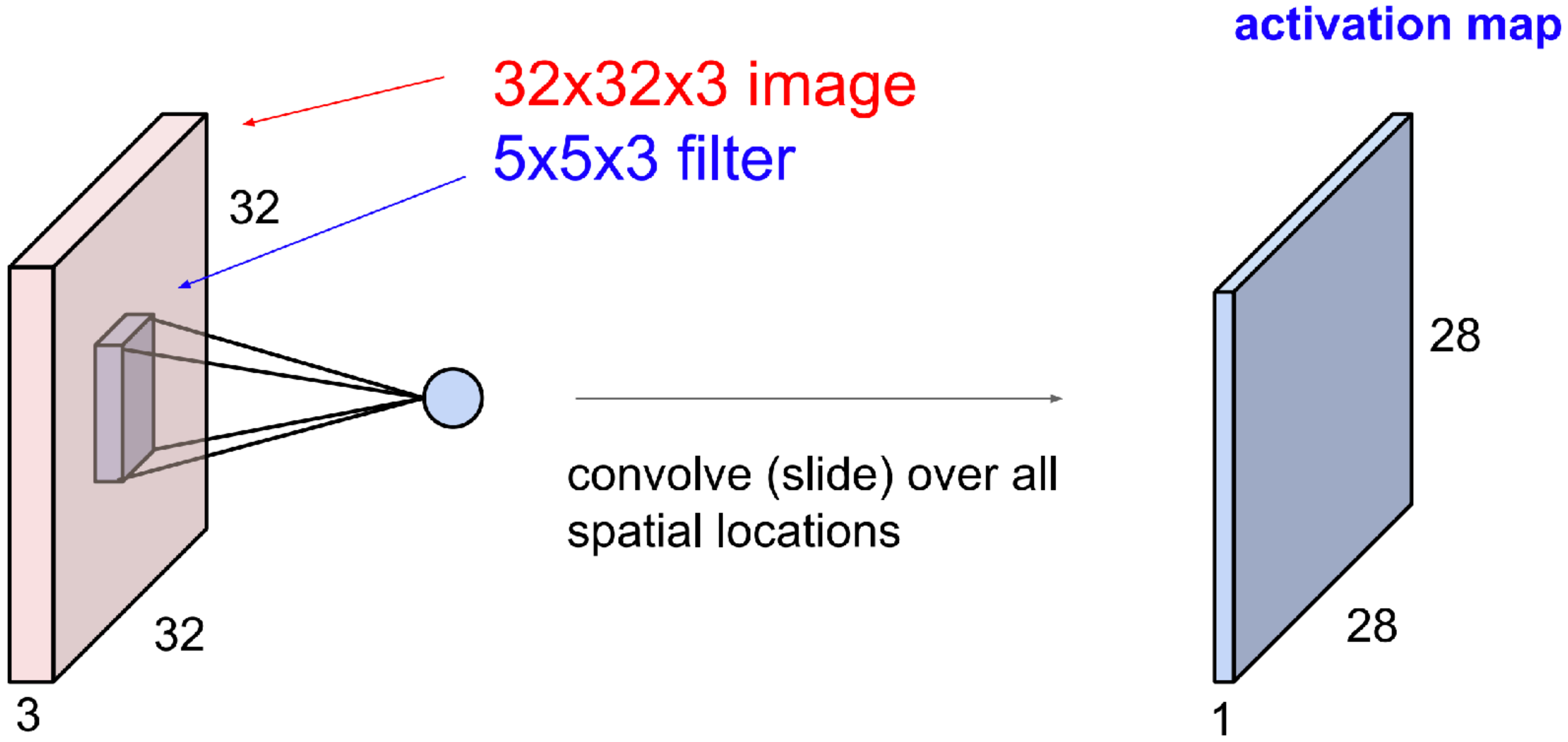
Convolution Layer



1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

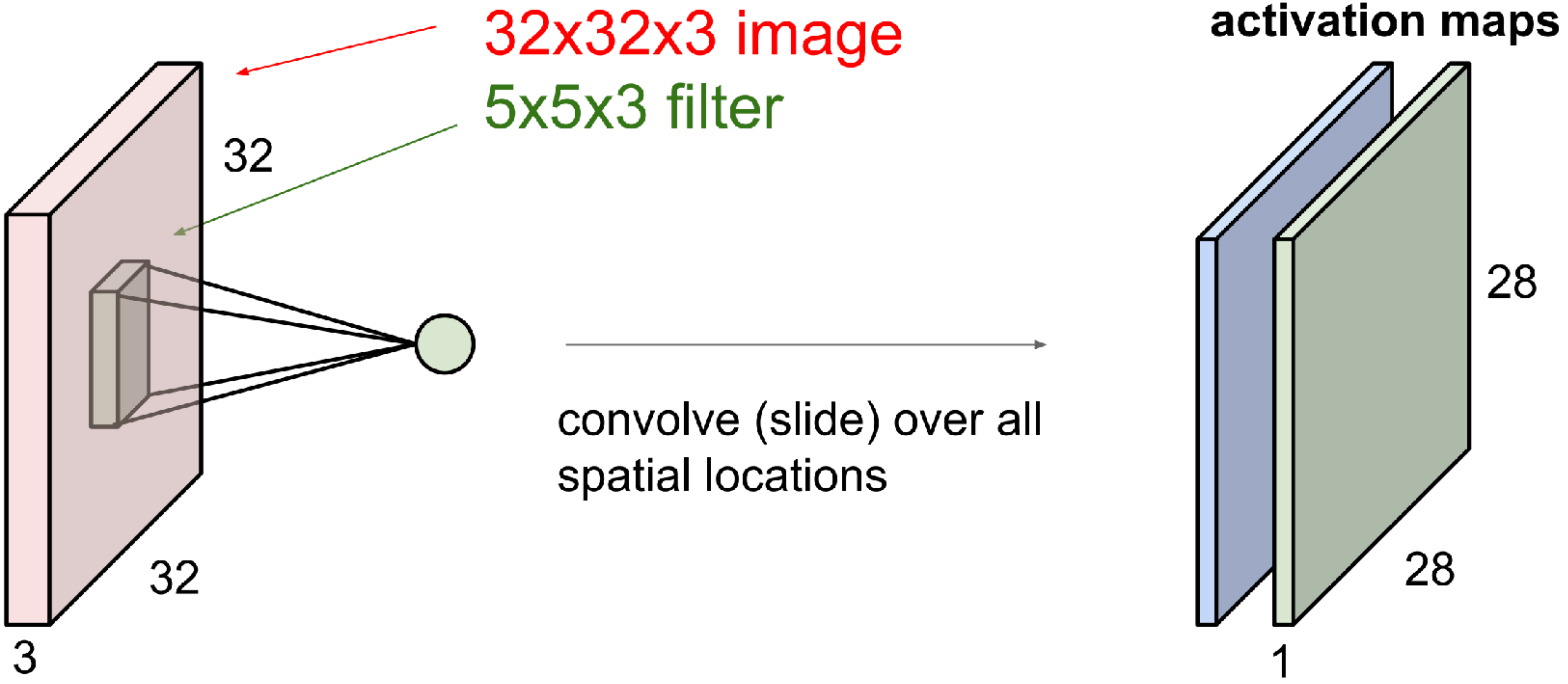
$$w^T x + b$$

Convolution Layer

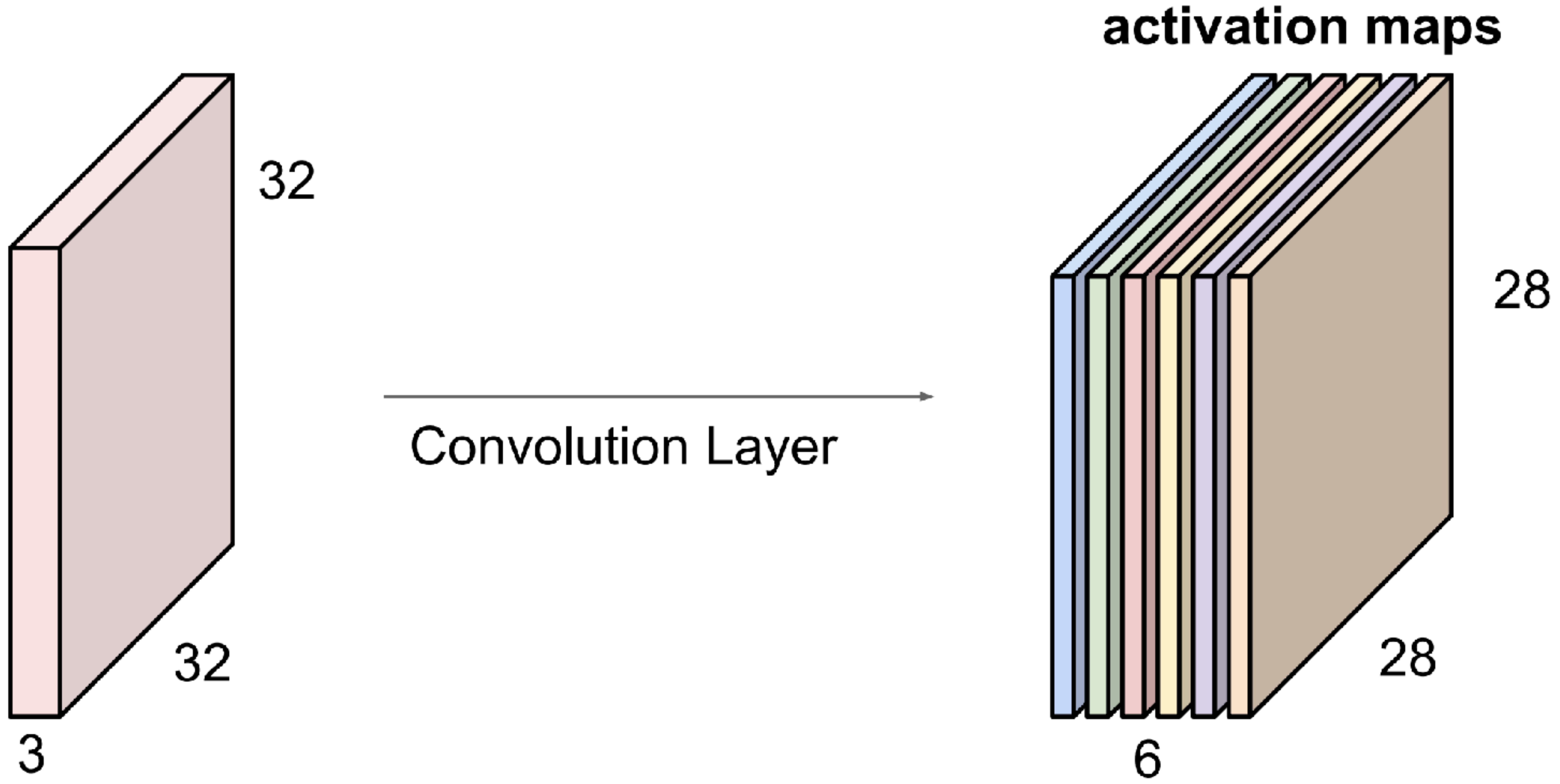


Convolution Layer

consider a second, green filter

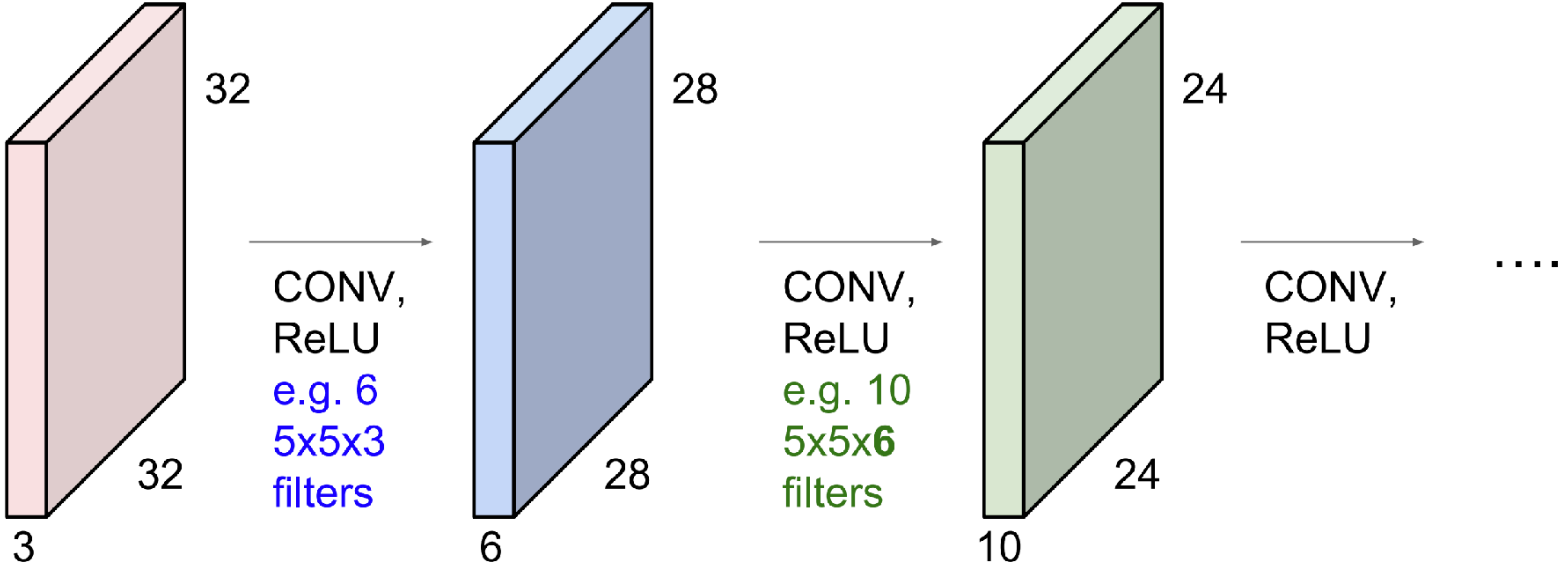


For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



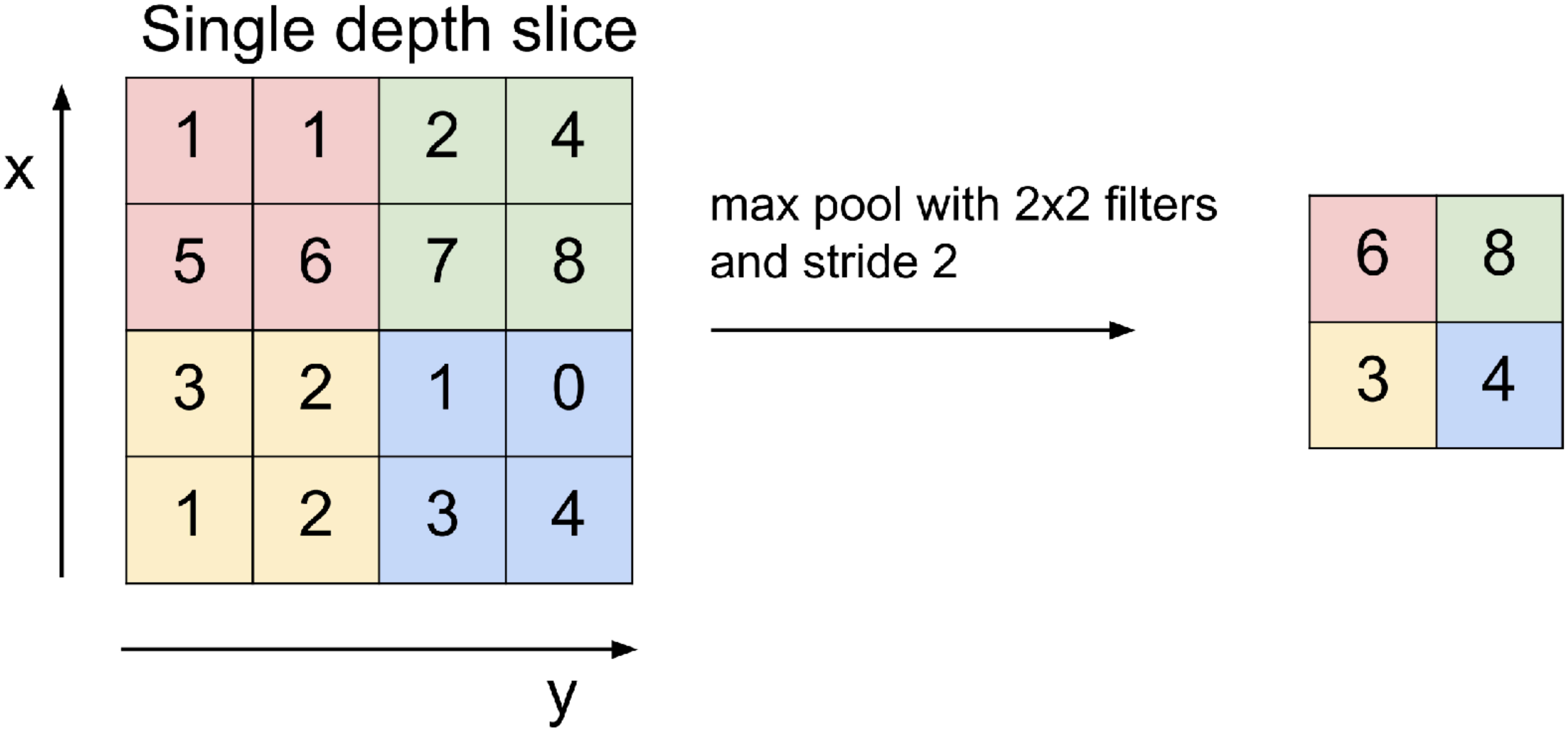
We stack these up to get a “new image” of size 28x28x6!

Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



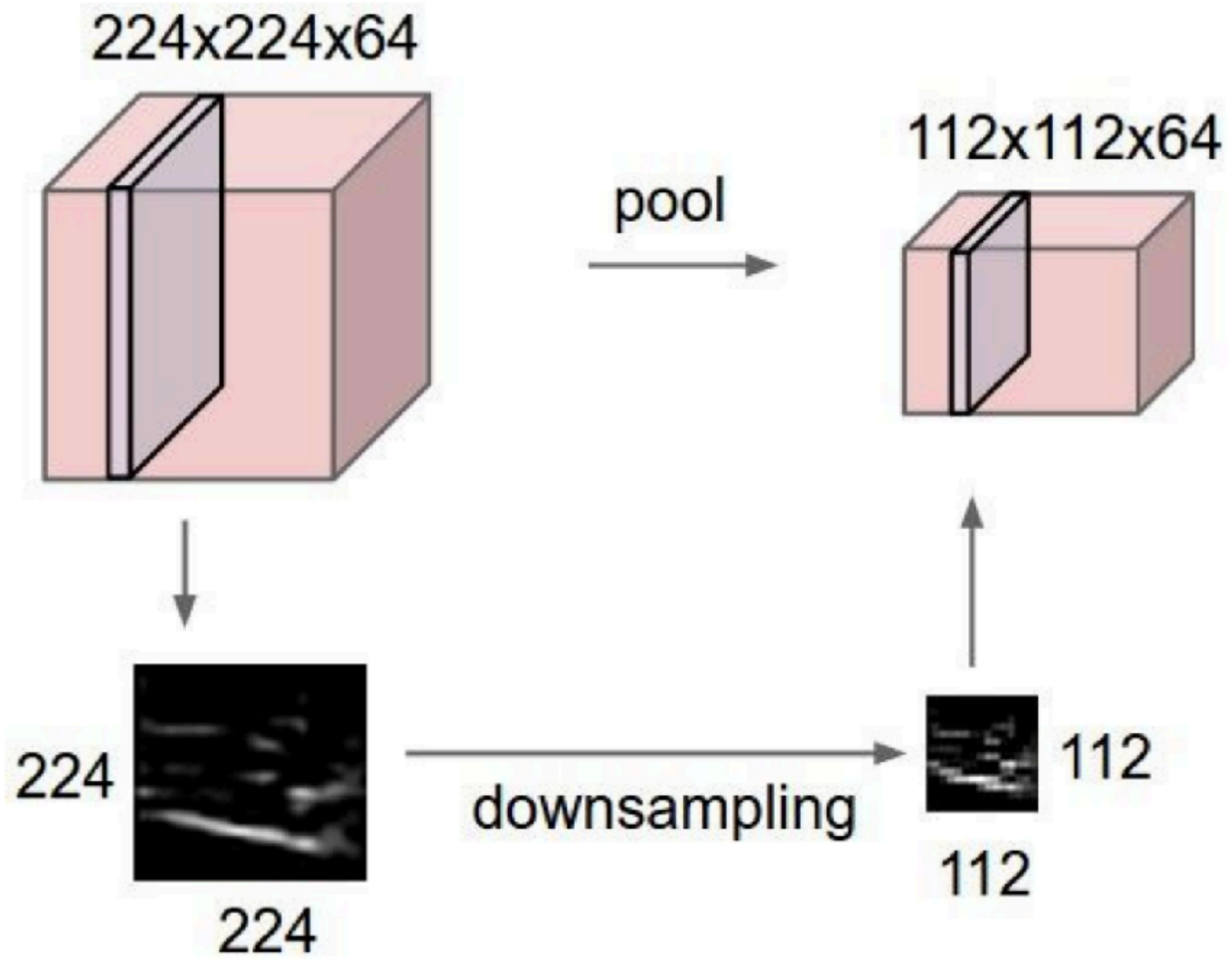
For more details + an animated demo see:
https://cs231n.github.io/convolutional-networks/

MAX POOLING


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

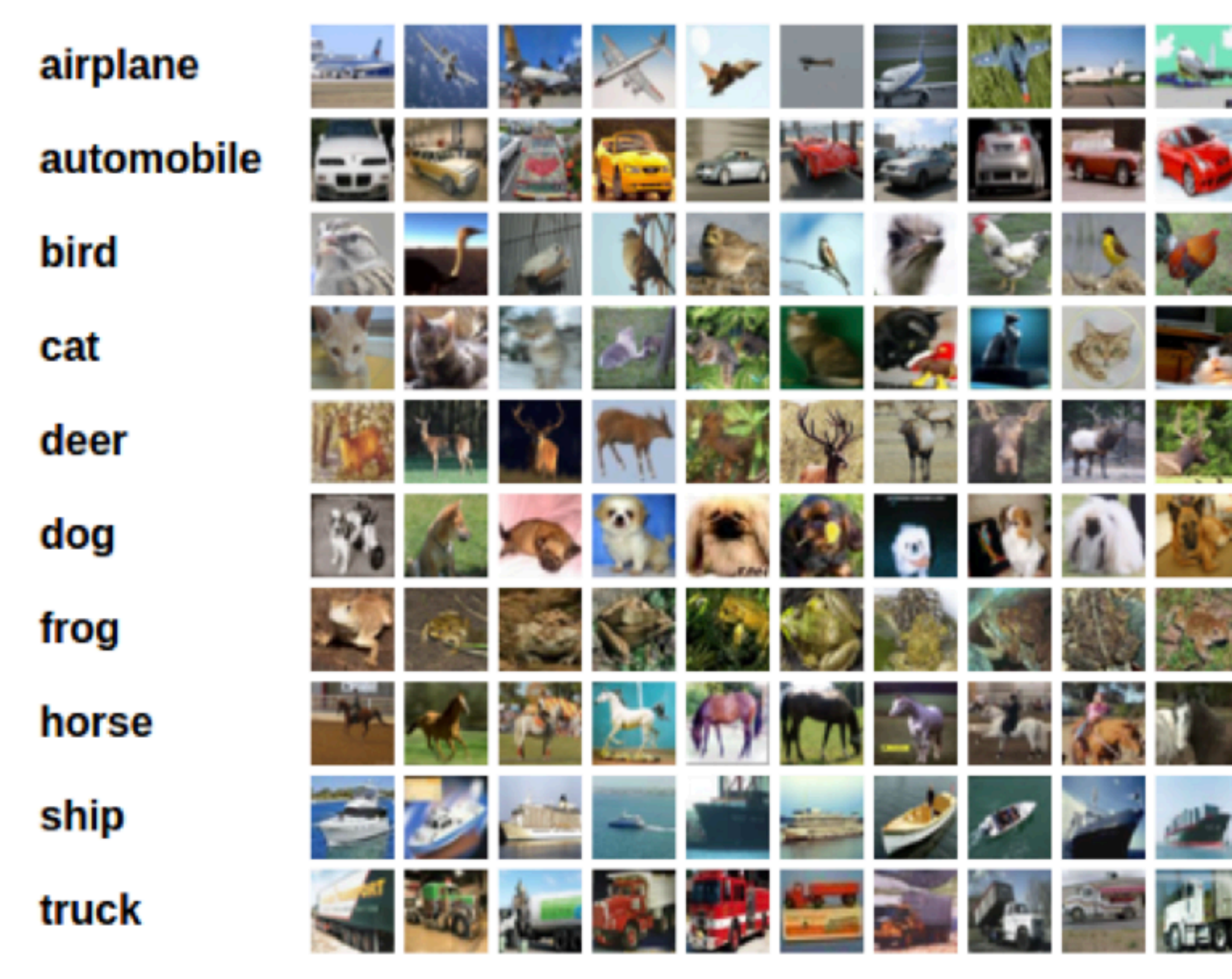
Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:

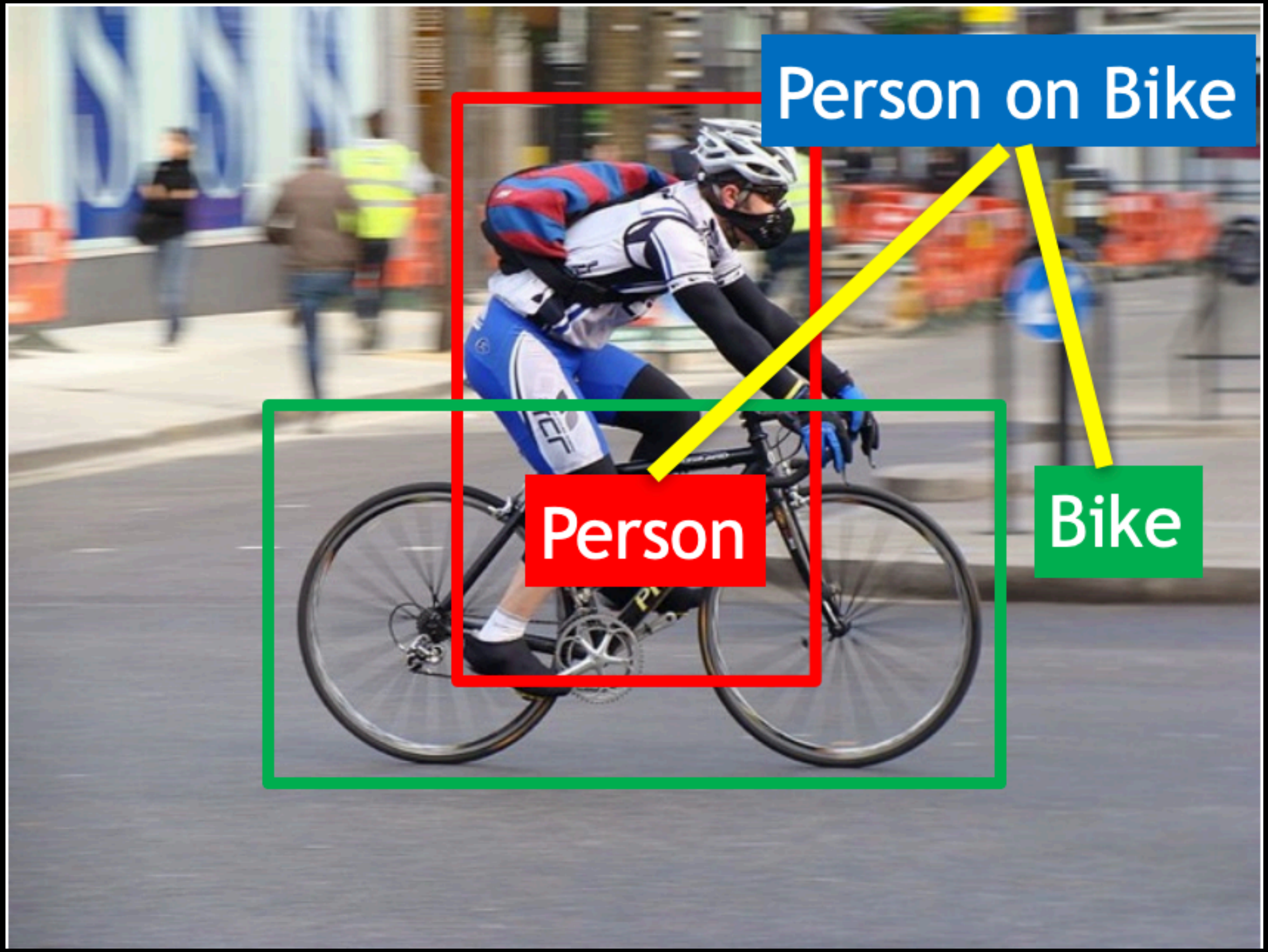



POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

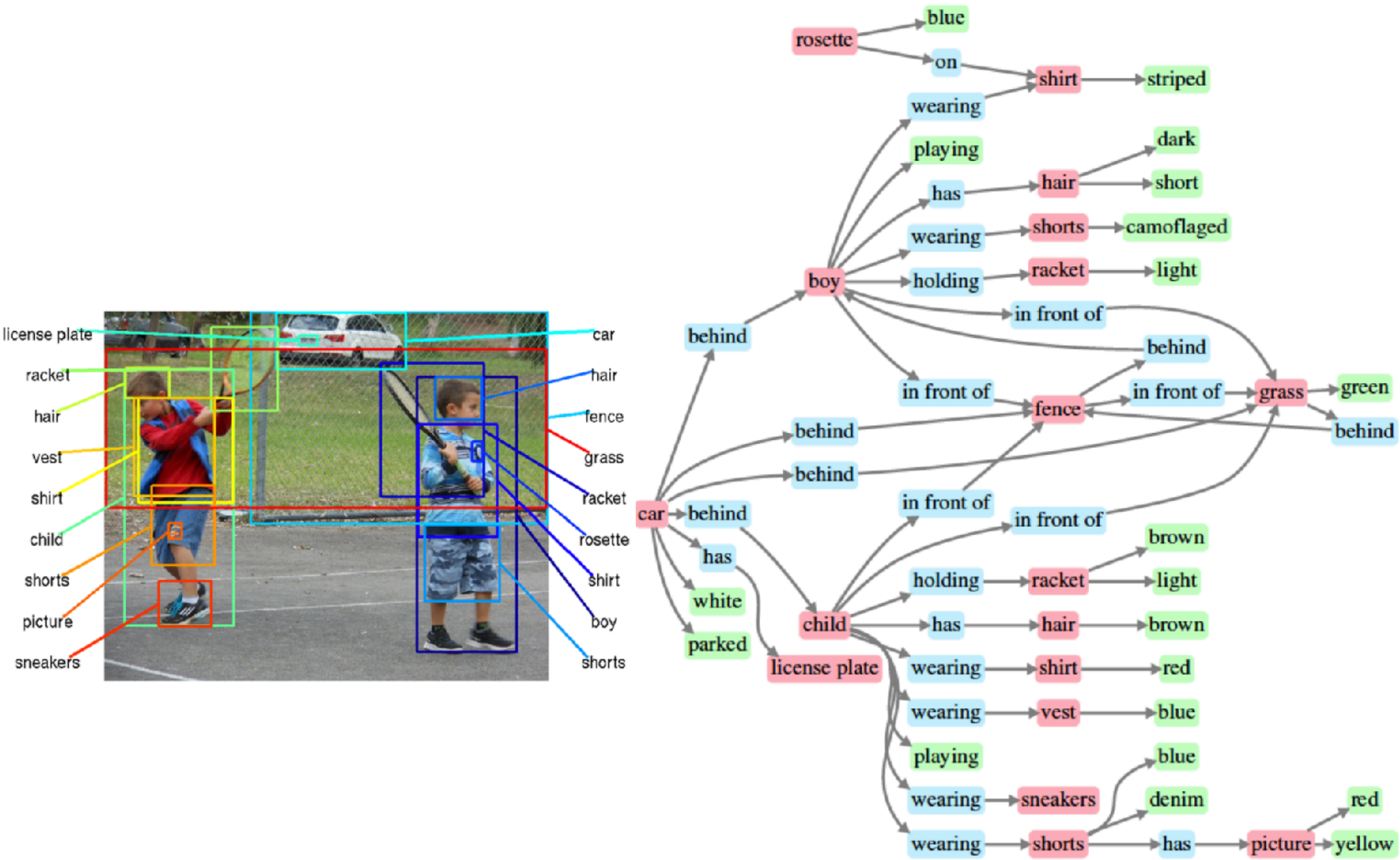
```python
import torch.nn as nn
import torch.nn.functional as F


class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1) # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x


net = Net()
```



https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html



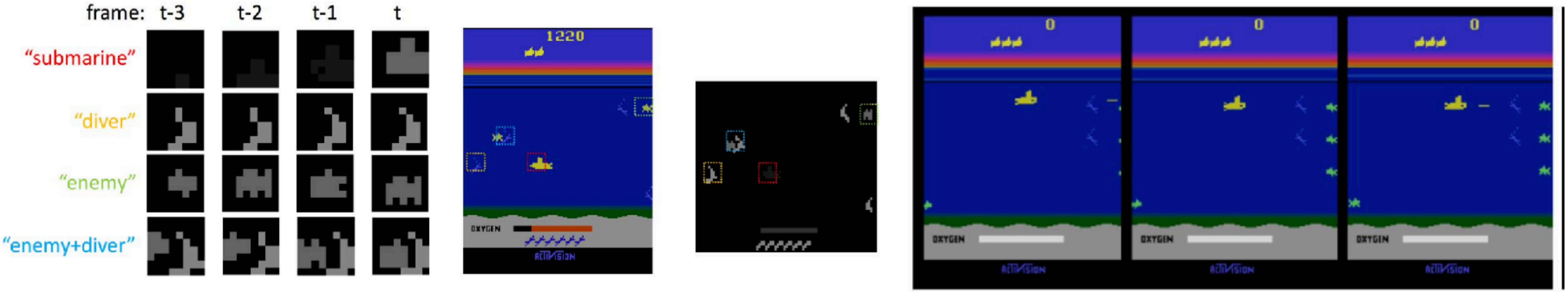
The following slides were taken from Stanford's CS231:
- https://cs231n.github.io/
- https://www.youtube.com/playlist?list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv



Johnson *et al.*, "Image Retrieval using Scene Graphs", CVPR 2015

Images are examples of pose estimation, not actually from Toshev & Szegedy 2014. Copyright Lane McIntosh.

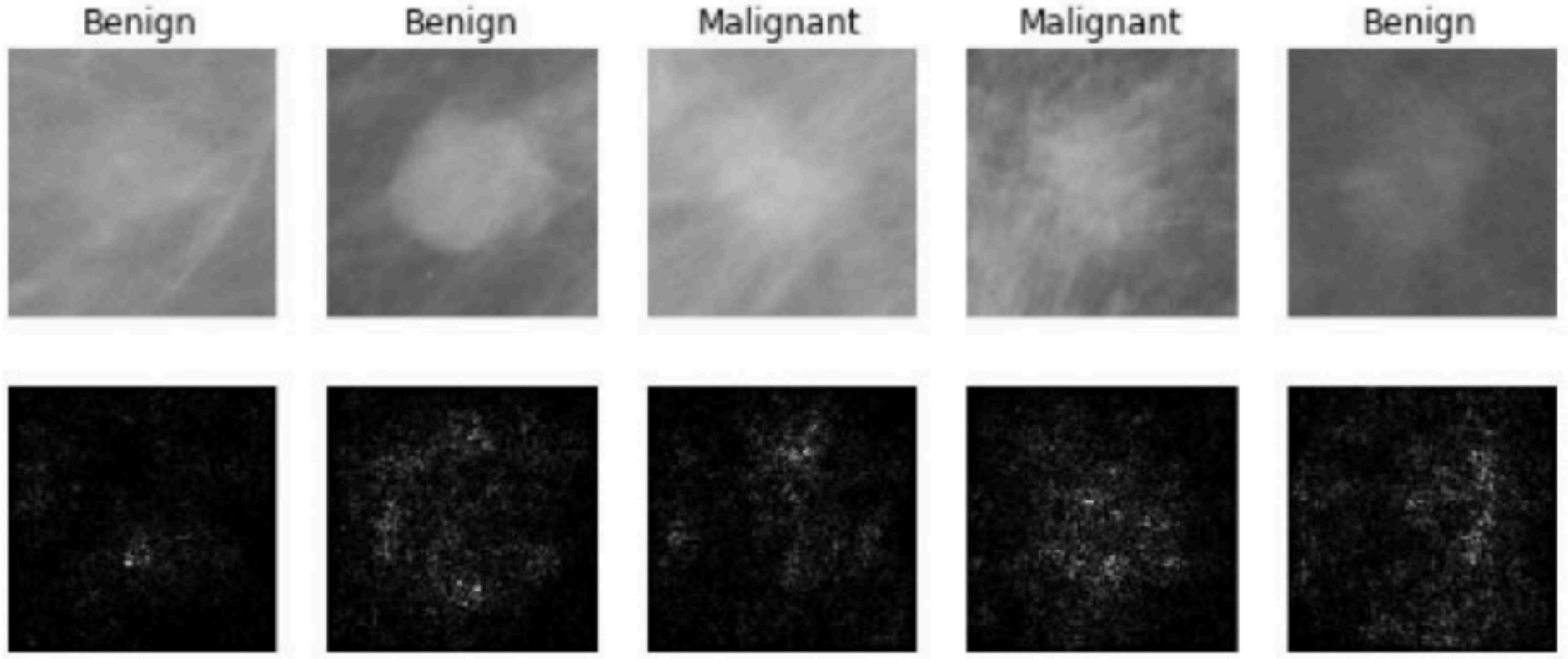
[Toshev, Szegedy 2014]



[Guo et al. 2014]

Figures copyright Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard Lewis, and Xiaoshi Wang, 2014. Reproduced with permission.

Fast-forward to today: ConvNets are everywhere



[Levy et al. 2016]

[Dieleman et al. 2014]

From left to right: public domain by NASA, usage permitted by ESA/Hubble, public domain by NASA, and public domain.

[Sermanet et al. 2011]
[Ciresan et al.]

This image by Christin Khan is in the public domain and originally came from the U.S. NOAA.

Whale recognition, Kaggle Challenge

Photo and figure by Lane McIntosh; not actual example from Mnih and Hinton, 2010 paper.

Mnih and Hinton, 2010

Image Captioning

[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]

No errors

A white teddy bear sitting in the grass

A man riding a wave on top of a surfboard

Minor errors

A man in a baseball uniform throwing a ball

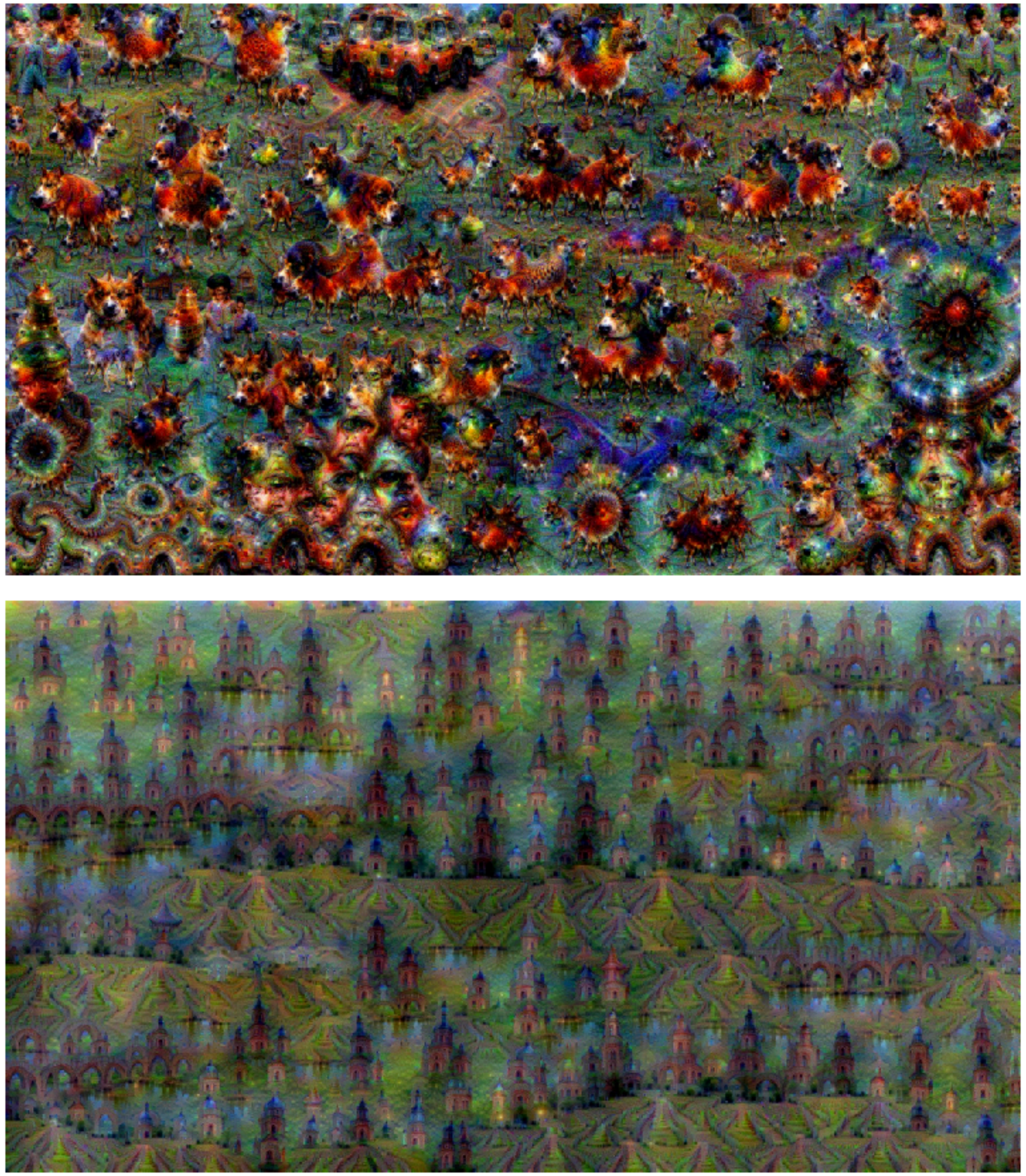
A cat sitting on a suitcase on the floor

Somewhat related

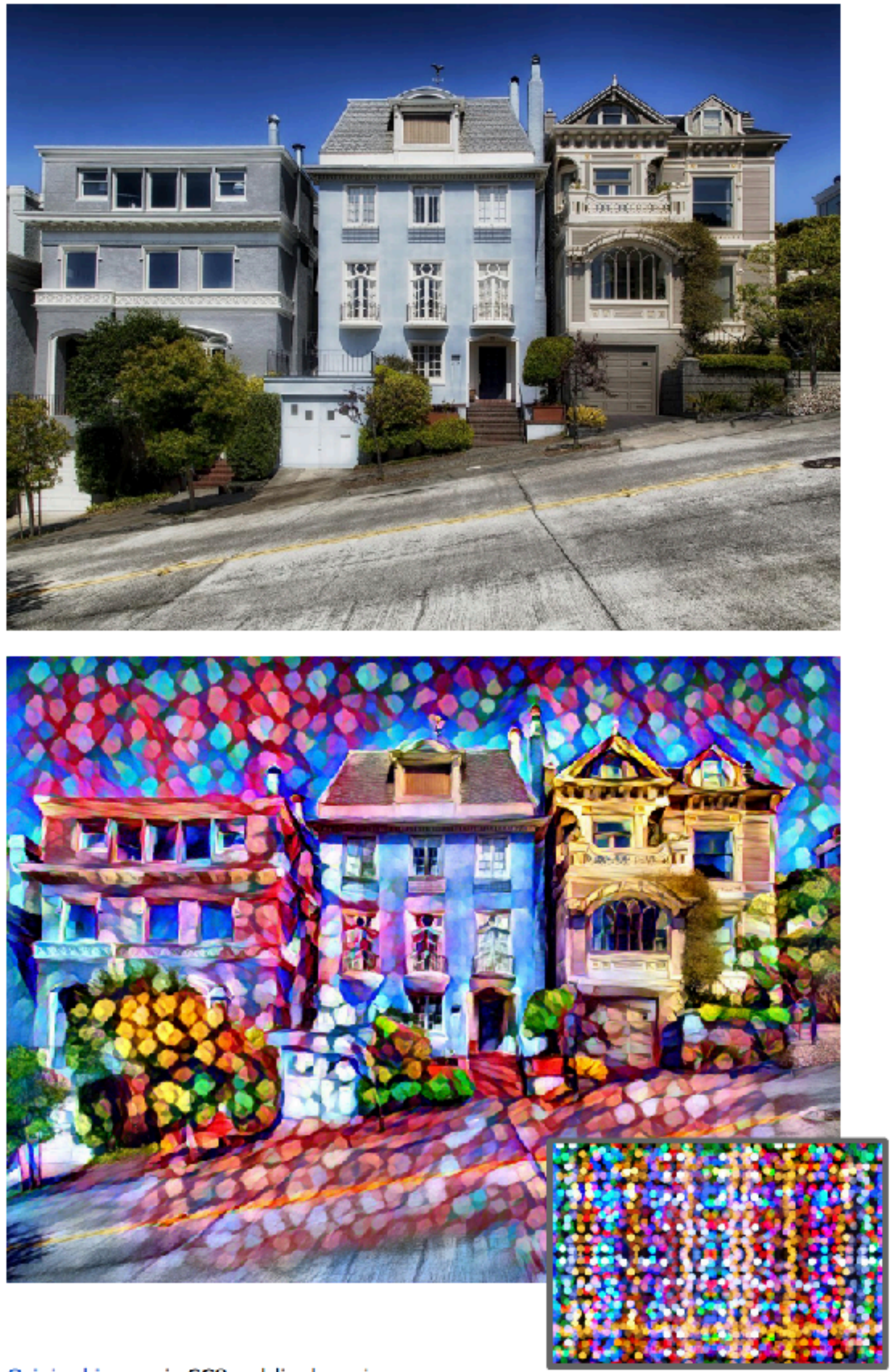
A woman is holding a cat in her hand

A woman standing on a beach holding a surfboard

Captions generated by Justin Johnson using Neuraltalk2



Figures copyright Justin Johnson, 2015. Reproduced with permission. Generated using the Inceptionism approach from a blog post by Google Research.



Original image is CC0 public domain
Starry Night and Tree Roots by Van Gogh are in the public domain
Bokeh image is in the public domain
Stylized images copyright Justin Johnson, 2017;
reproduced with permission

Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016
Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017

IMAGENET Large Scale Visual Recognition Challenge

Steel drum

The Image Classification Challenge:

1,000 object classes

1,431,167 images

Output:
Scale
T-shirt
Steel drum
Drumstick
Mud turtle

✔

Output:
Scale
T-shirt
Giant panda
Drumstick
Mud turtle

✗

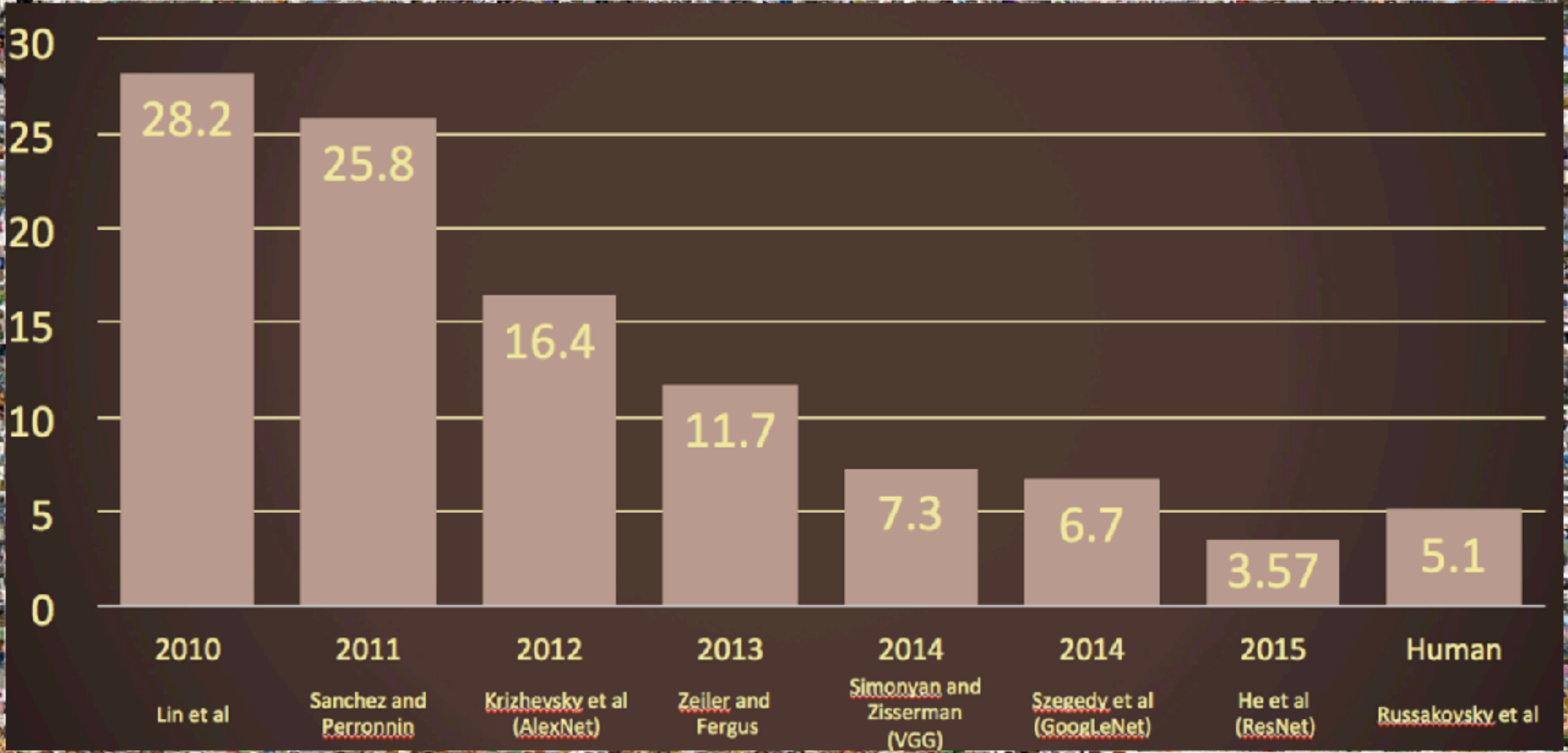
Russakovsky et al. arXiv, 2014

IMAGENET Large Scale Visual Recognition Challenge

The Image Classification Challenge:

1,000 object classes

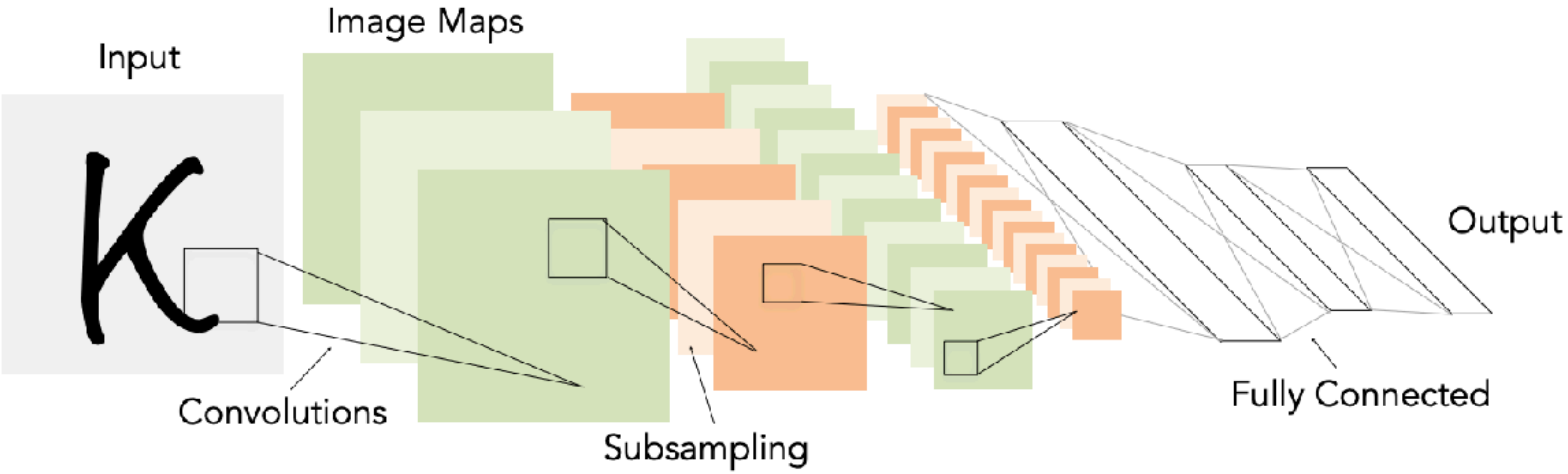
1,431,167 images



Russakovsky et al. arXiv, 2014

1998

LeCun et al.



# of transistors

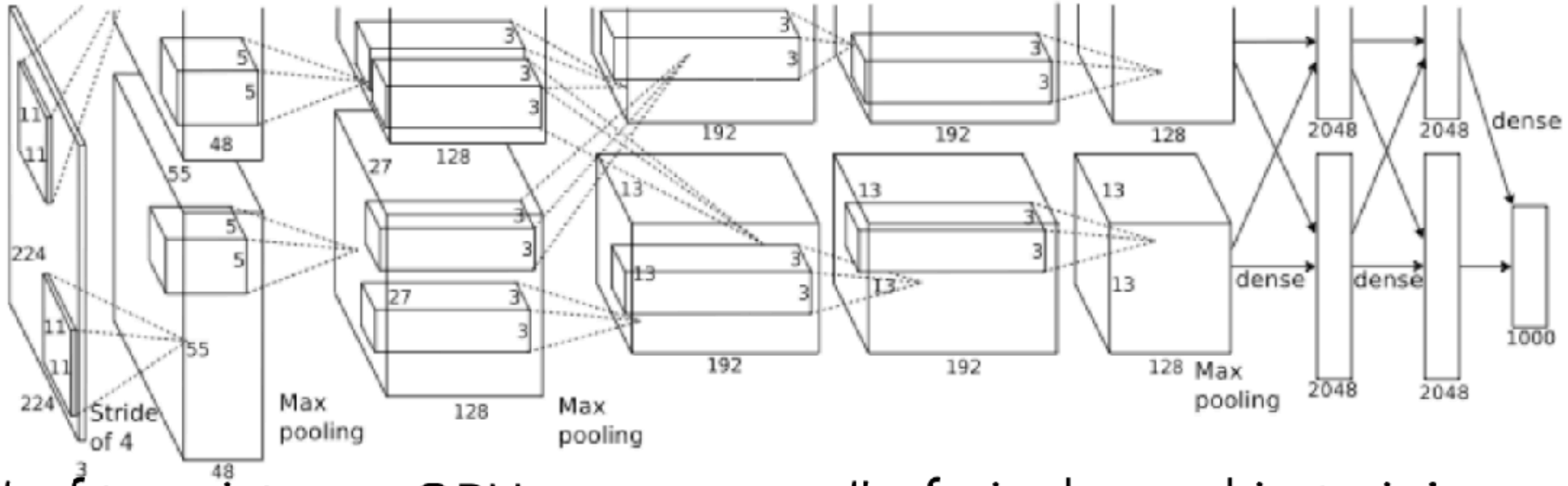
10^6

# of pixels used in training

10^7 NIST

2012

Krizhevsky et al.



# of transistors

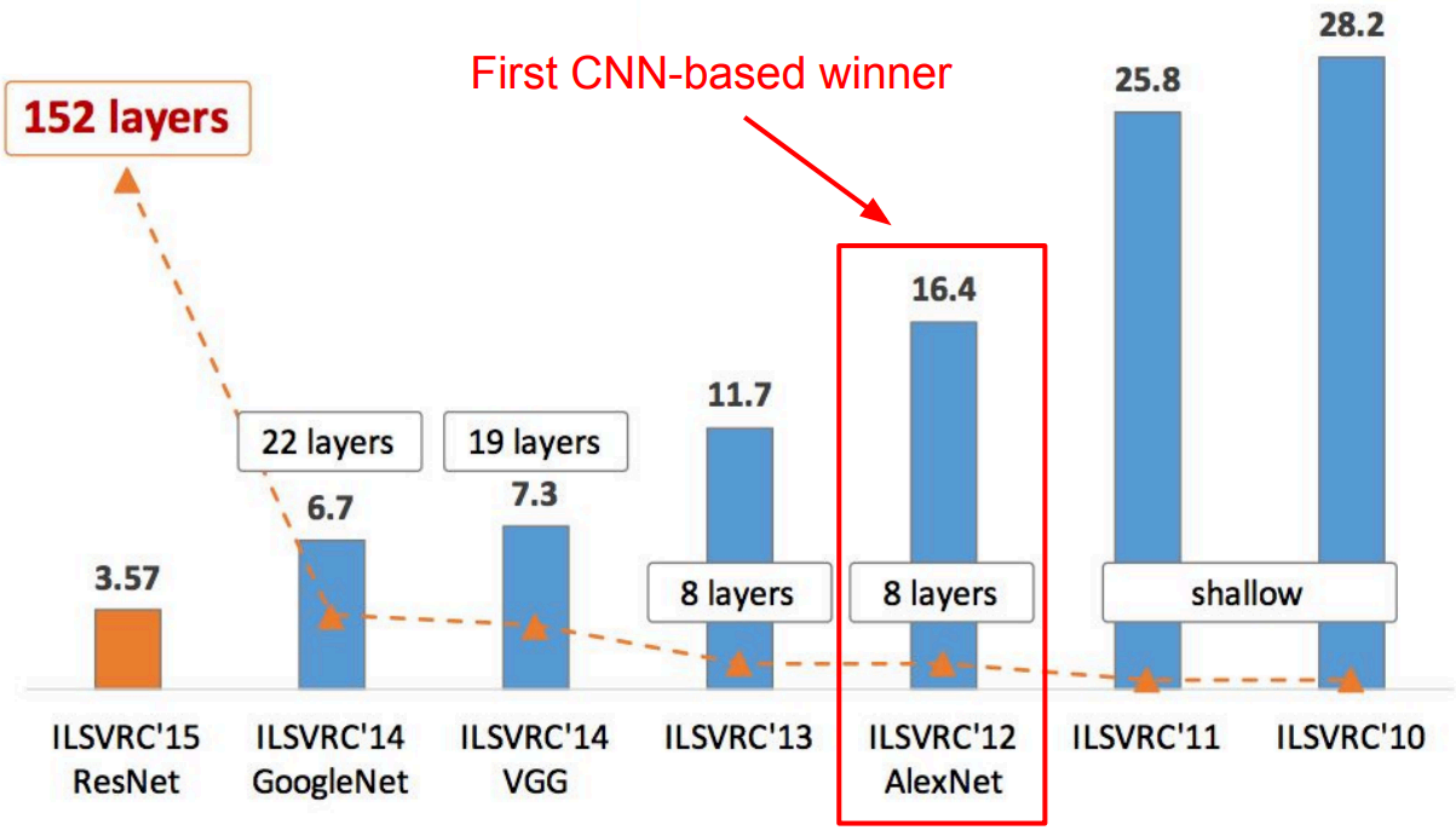
10^9

GPUs

# of pixels used in training

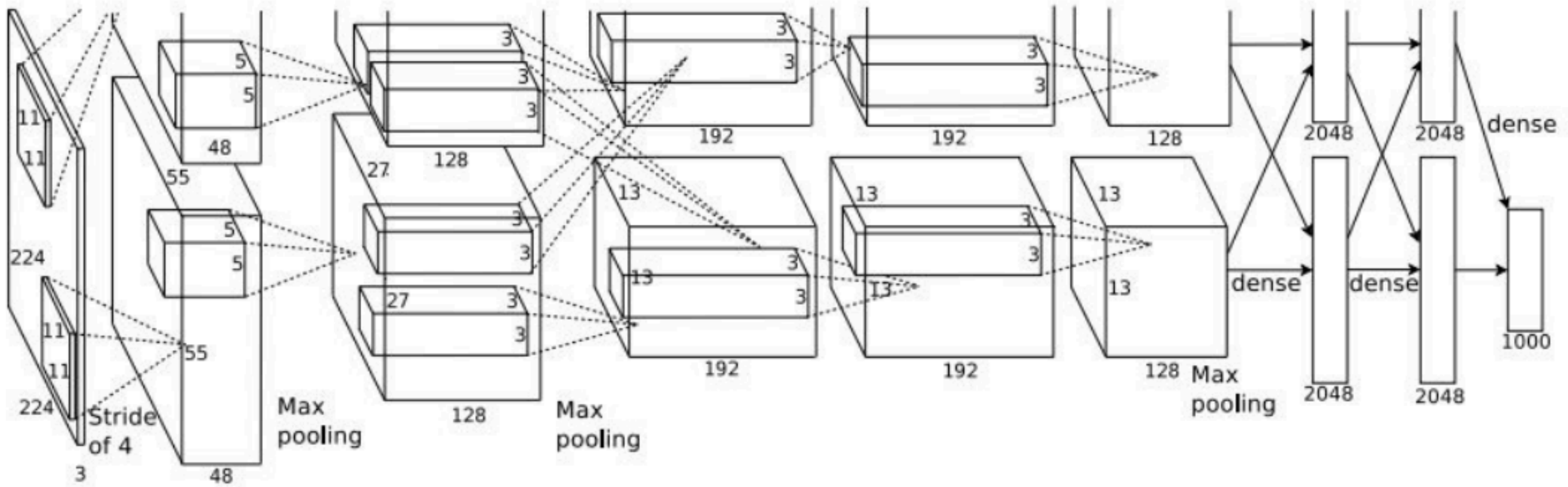
10^{14} IMAGENET

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Case Study: AlexNet

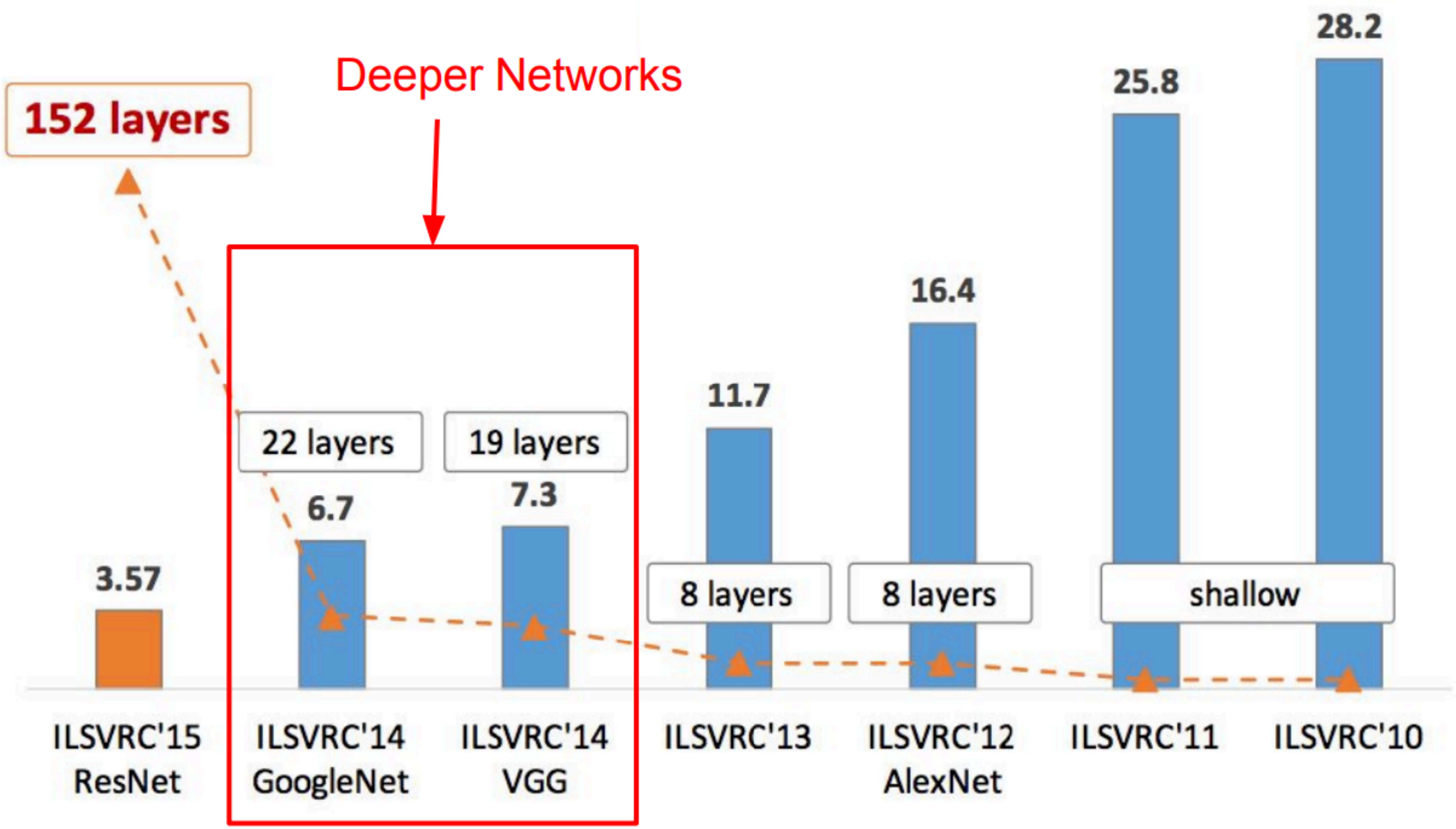
[Krizhevsky et al. 2012]



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%

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Case Study: VGGNet

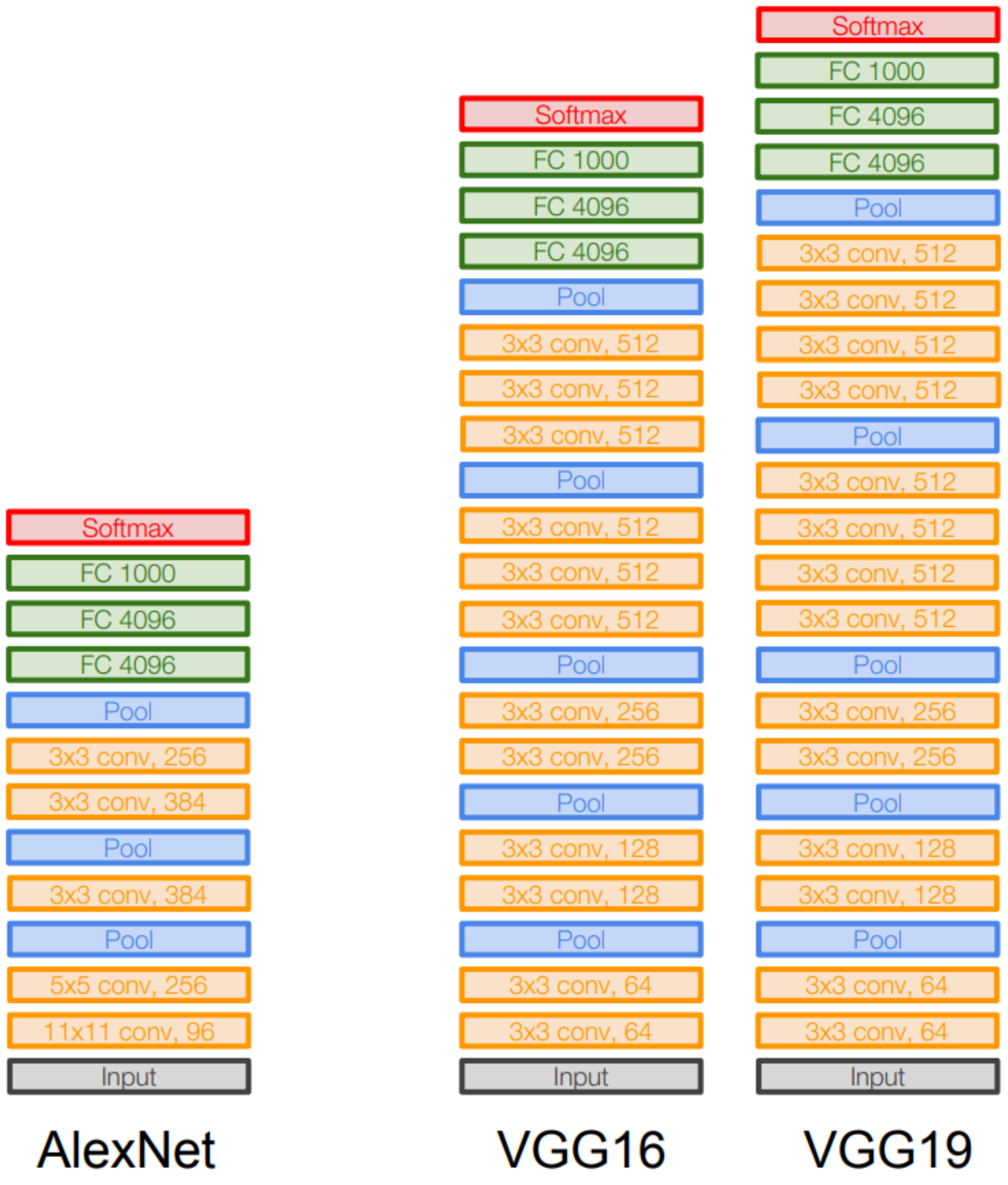
[Simonyan and Zisserman, 2014]

Small filters, Deeper networks

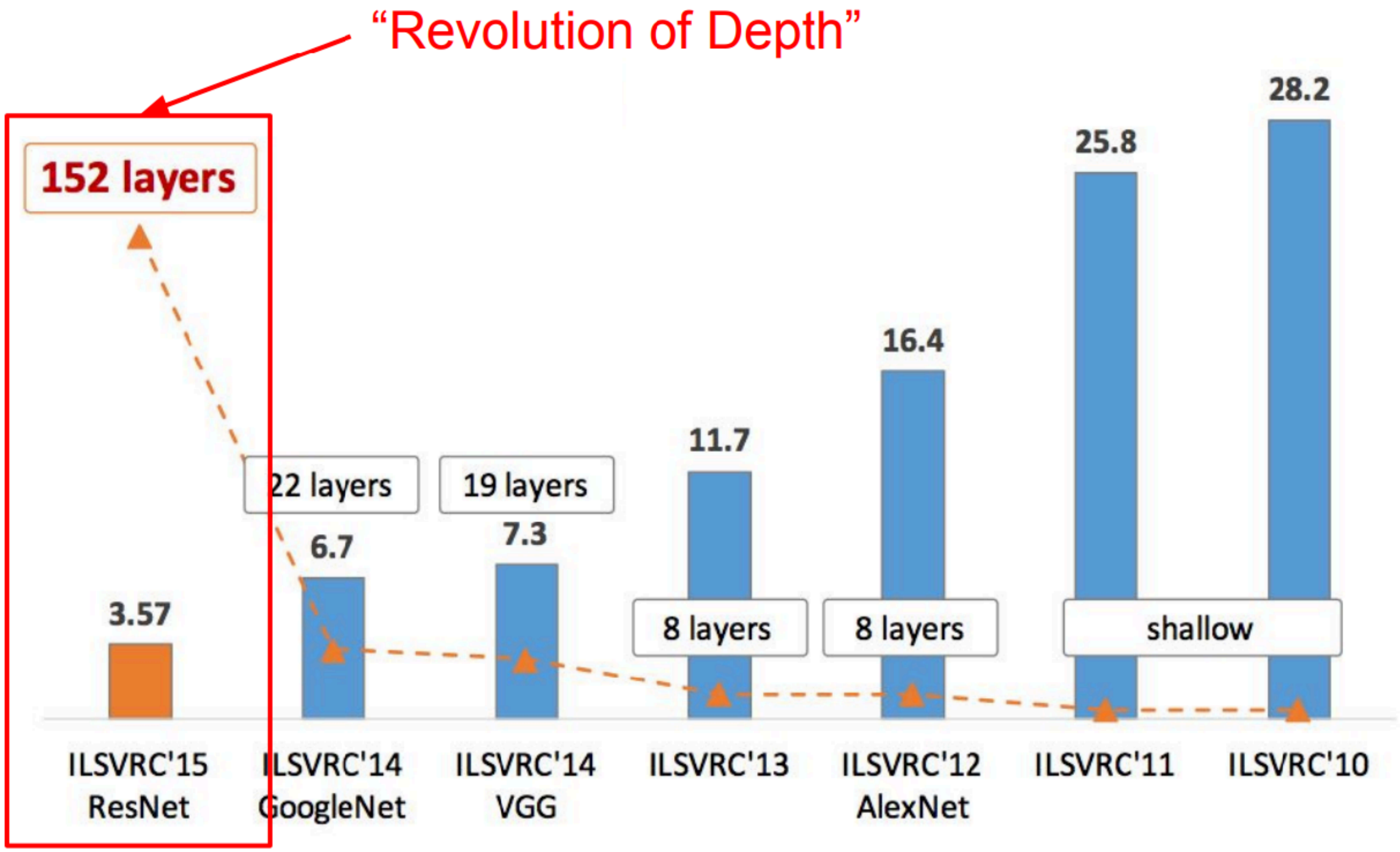
8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

11.7% top 5 error in ILSVRC'13
(ZFNet)
-> 7.3% top 5 error in ILSVRC'14



ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

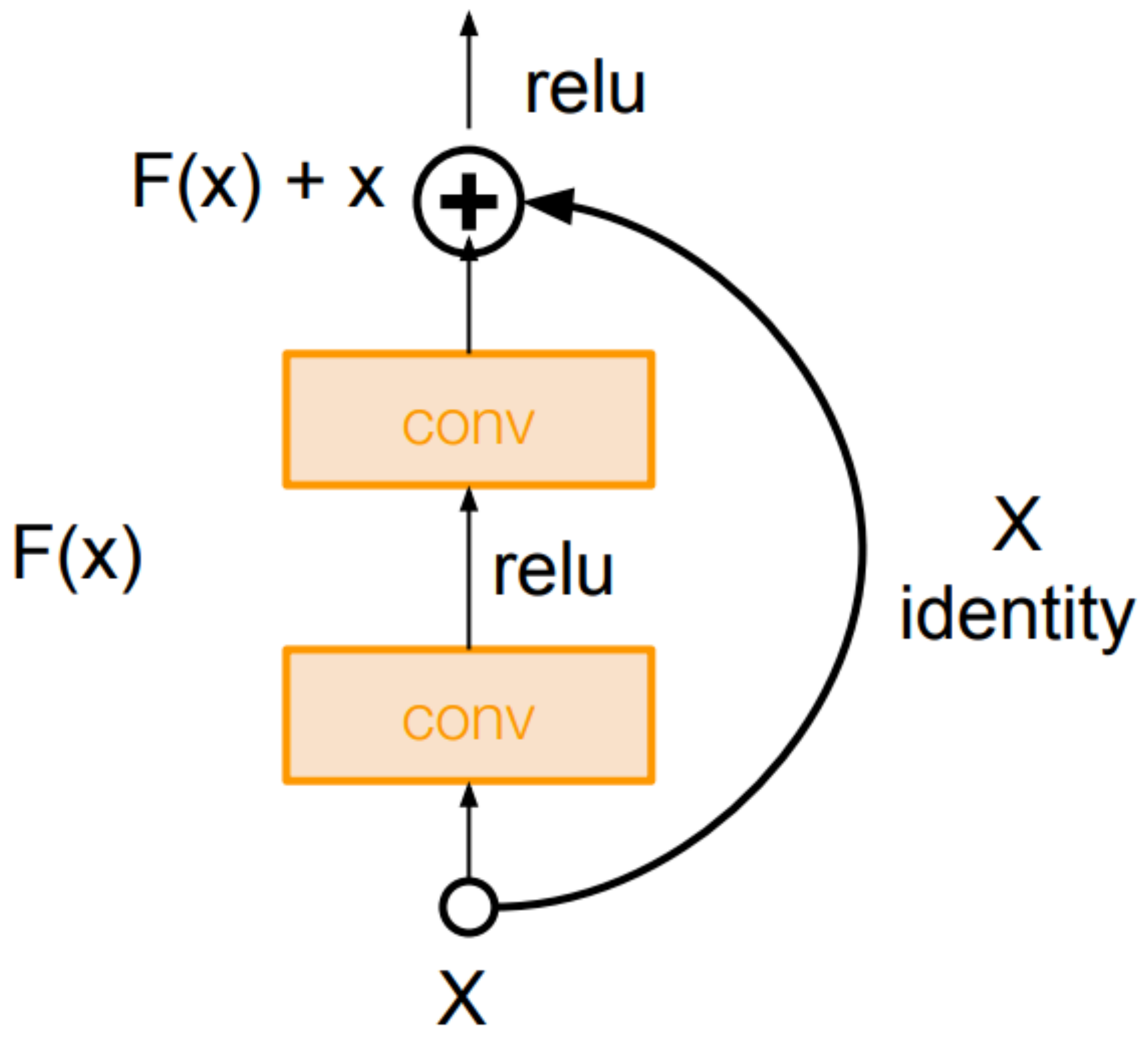


Case Study: ResNet

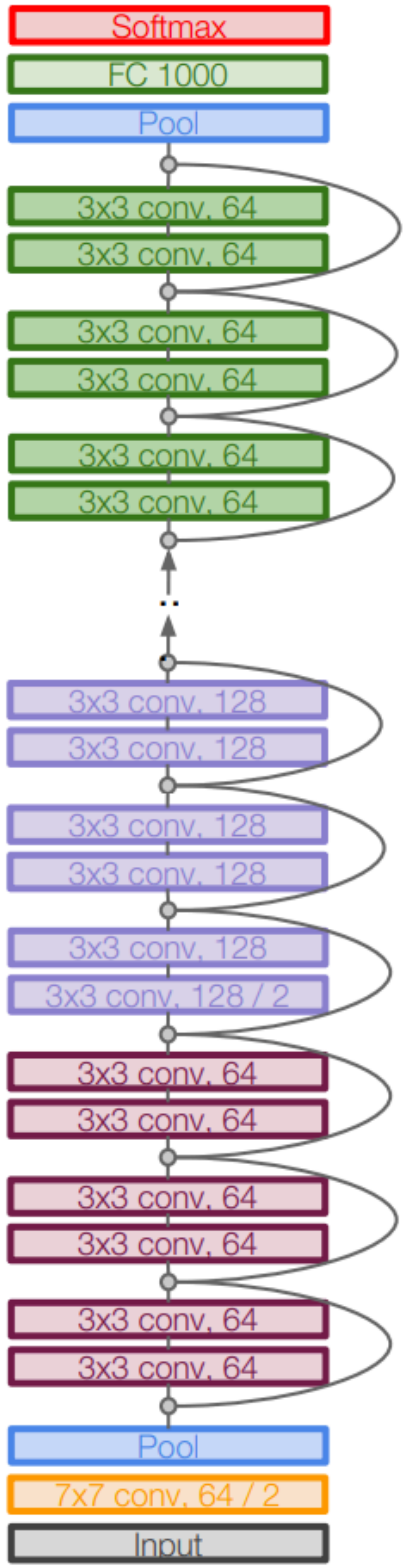
[He et al., 2015]

Very deep networks using residual connections

- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!



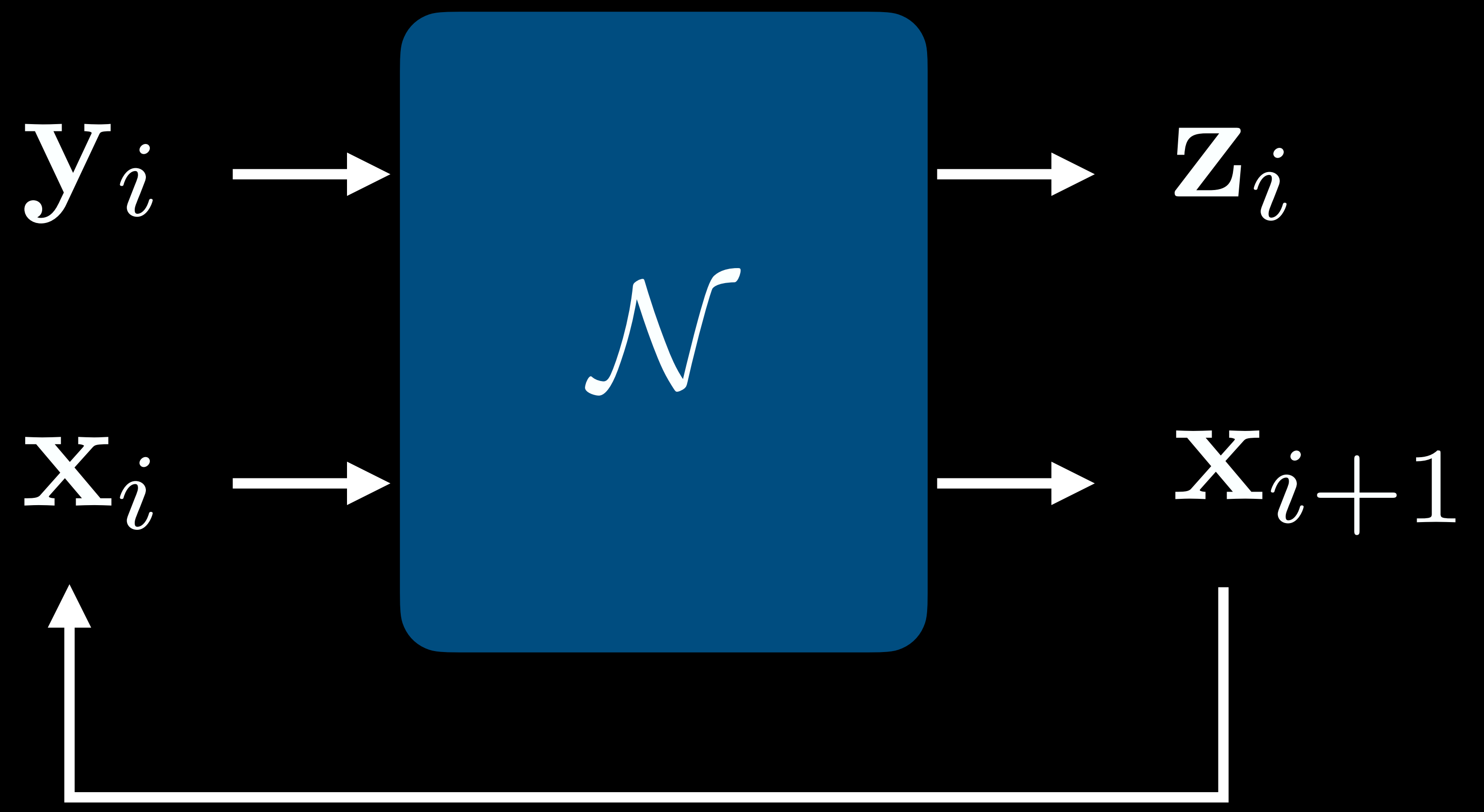
Residual block



Recurrent neural networks

$$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_n$$

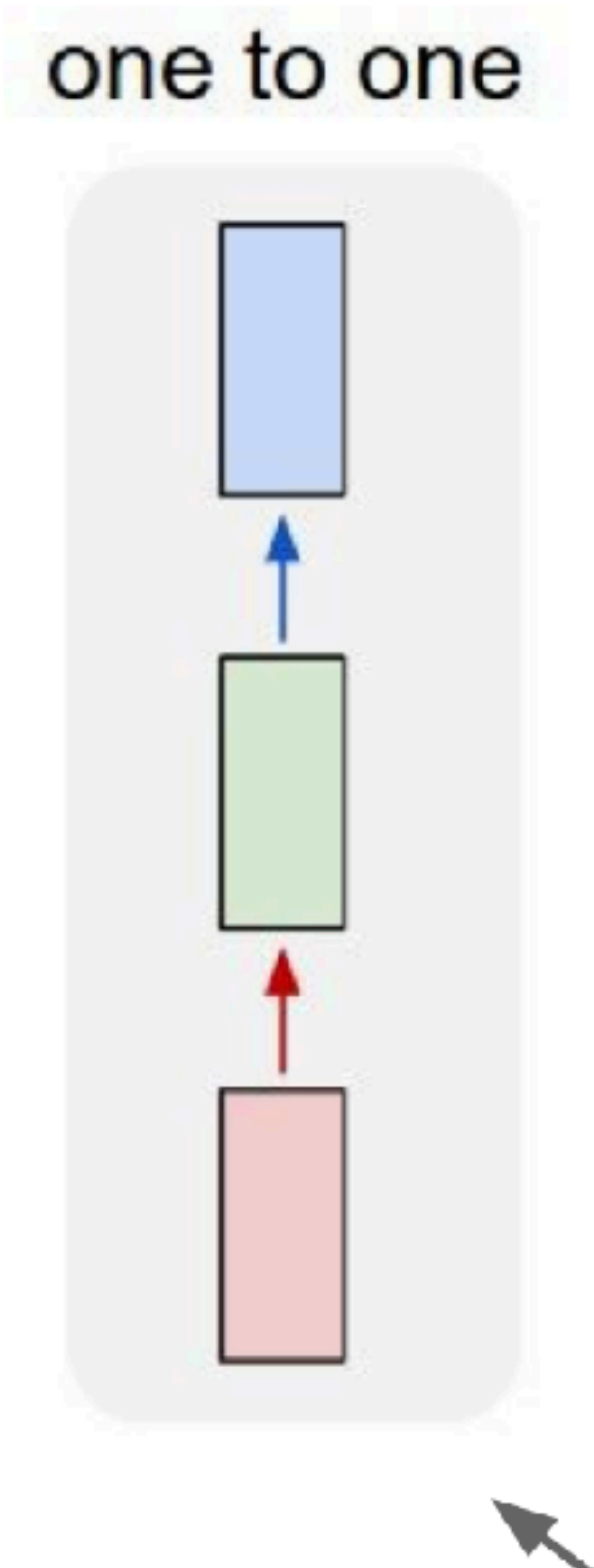
$$[\mathbf{x}_{i+1}, \mathbf{z}_i] = \mathcal{N}(\mathbf{x}_i, \mathbf{y}_i)$$



Feedback output to input

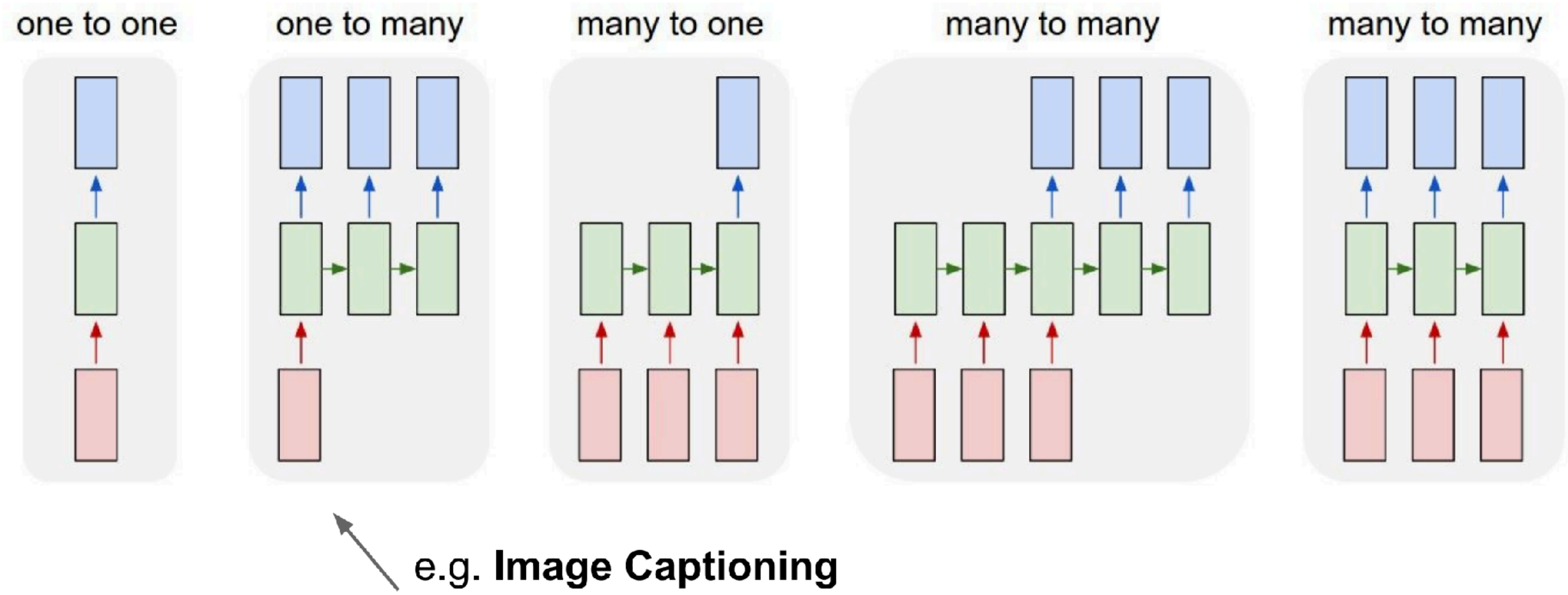
Numerical solvers are recurrence relations!

“Vanilla” Neural Network



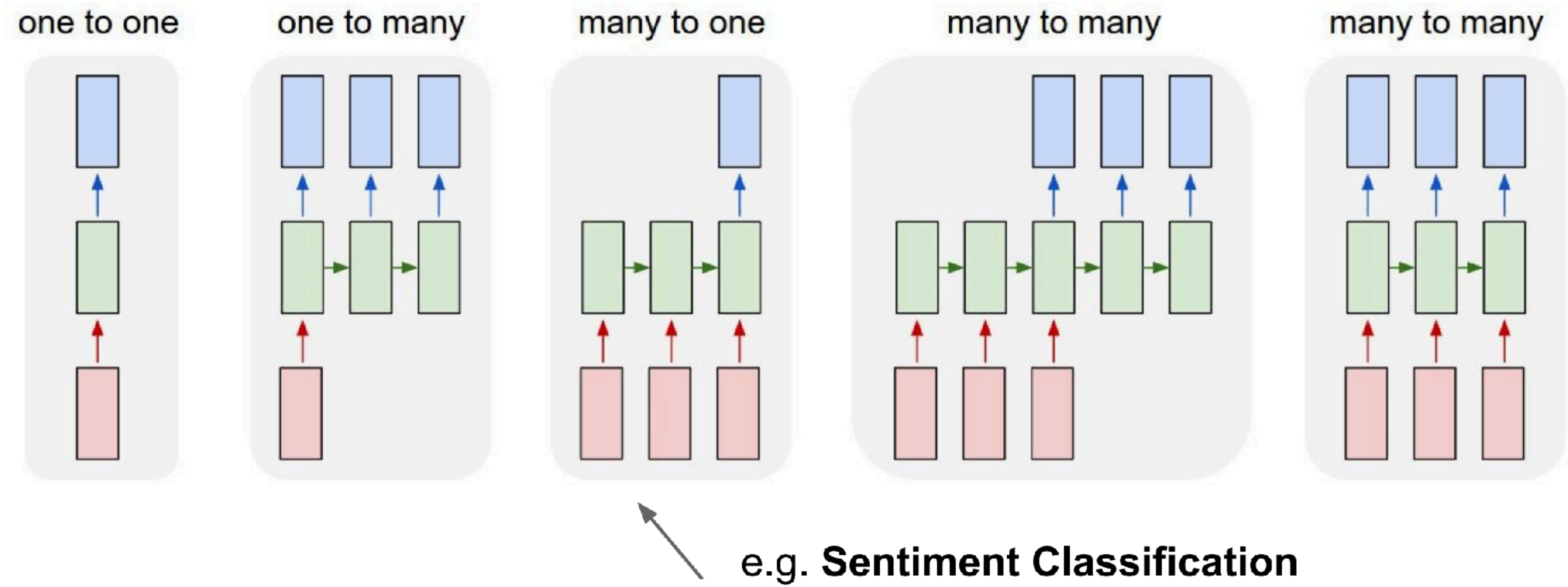
Vanilla Neural Networks

Recurrent Neural Networks: Process Sequences



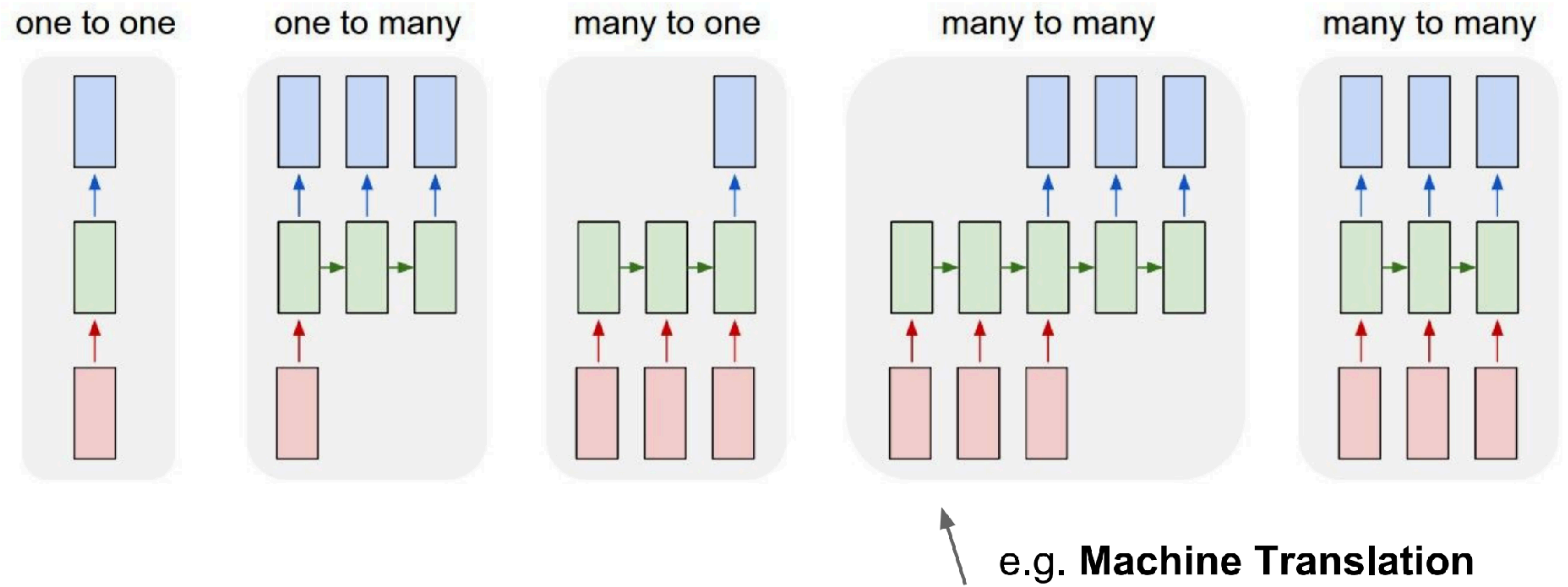
e.g. **Image Captioning**
image -> sequence of words

Recurrent Neural Networks: Process Sequences



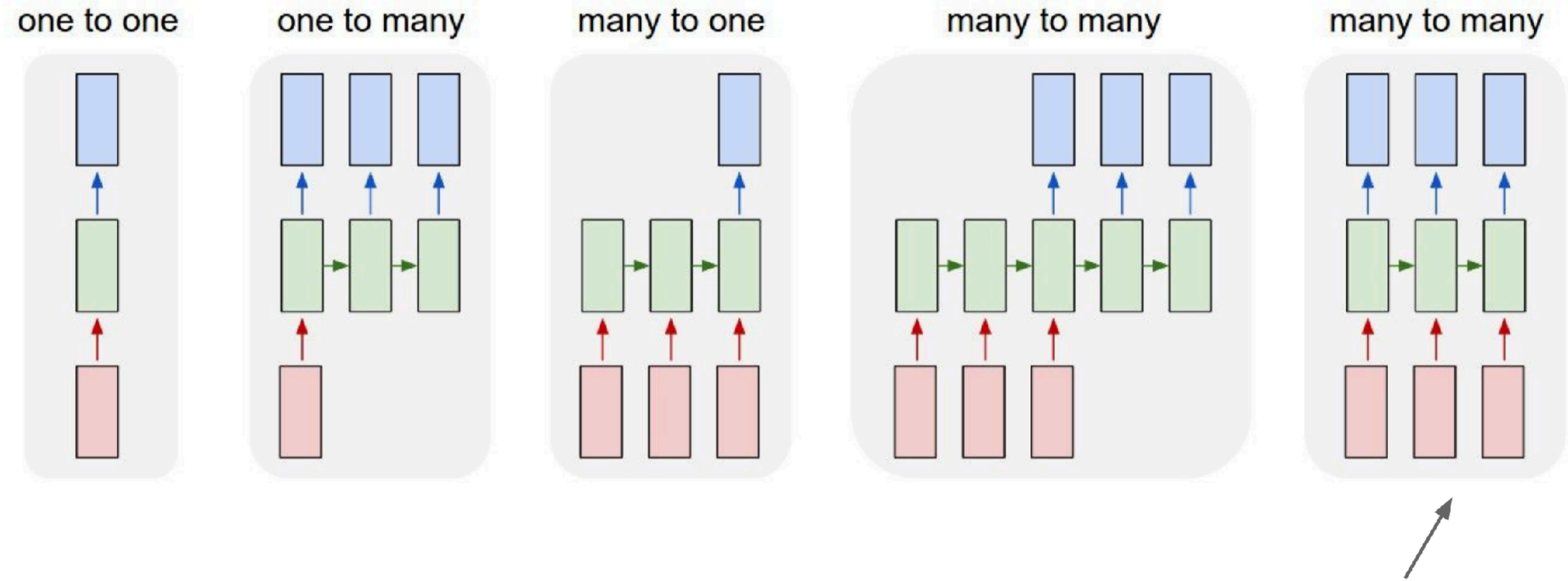
e.g. **Sentiment Classification**
sequence of words -> sentiment

Recurrent Neural Networks: Process Sequences



e.g. **Machine Translation**
seq of words -> seq of words

Recurrent Neural Networks: Process Sequences



e.g. **Video classification on frame level**

Recurrent Neural Network

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

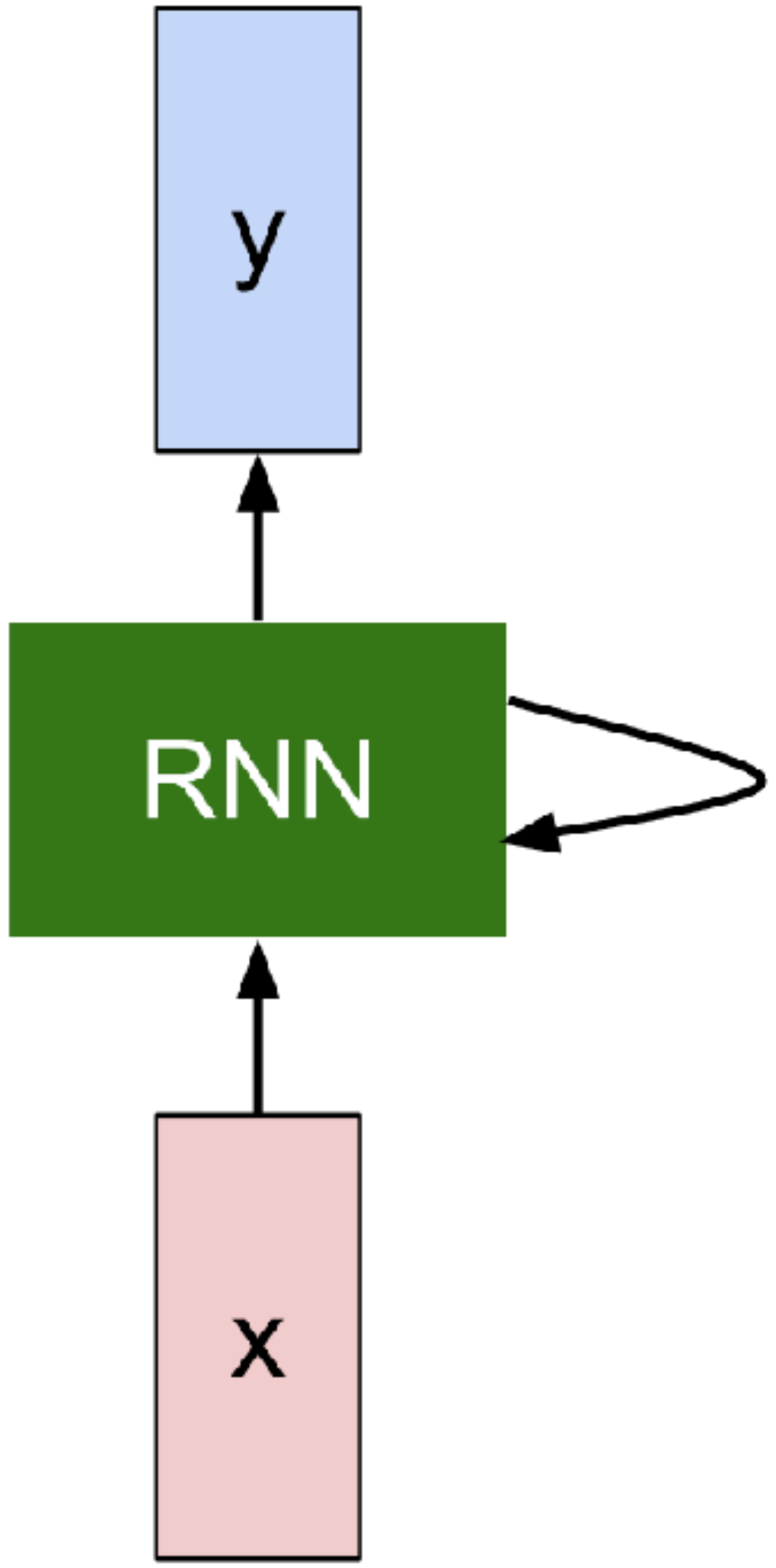
$$h_t = f_W(h_{t-1}, x_t)$$

new state

some function with parameters W

old state

input vector at some time step

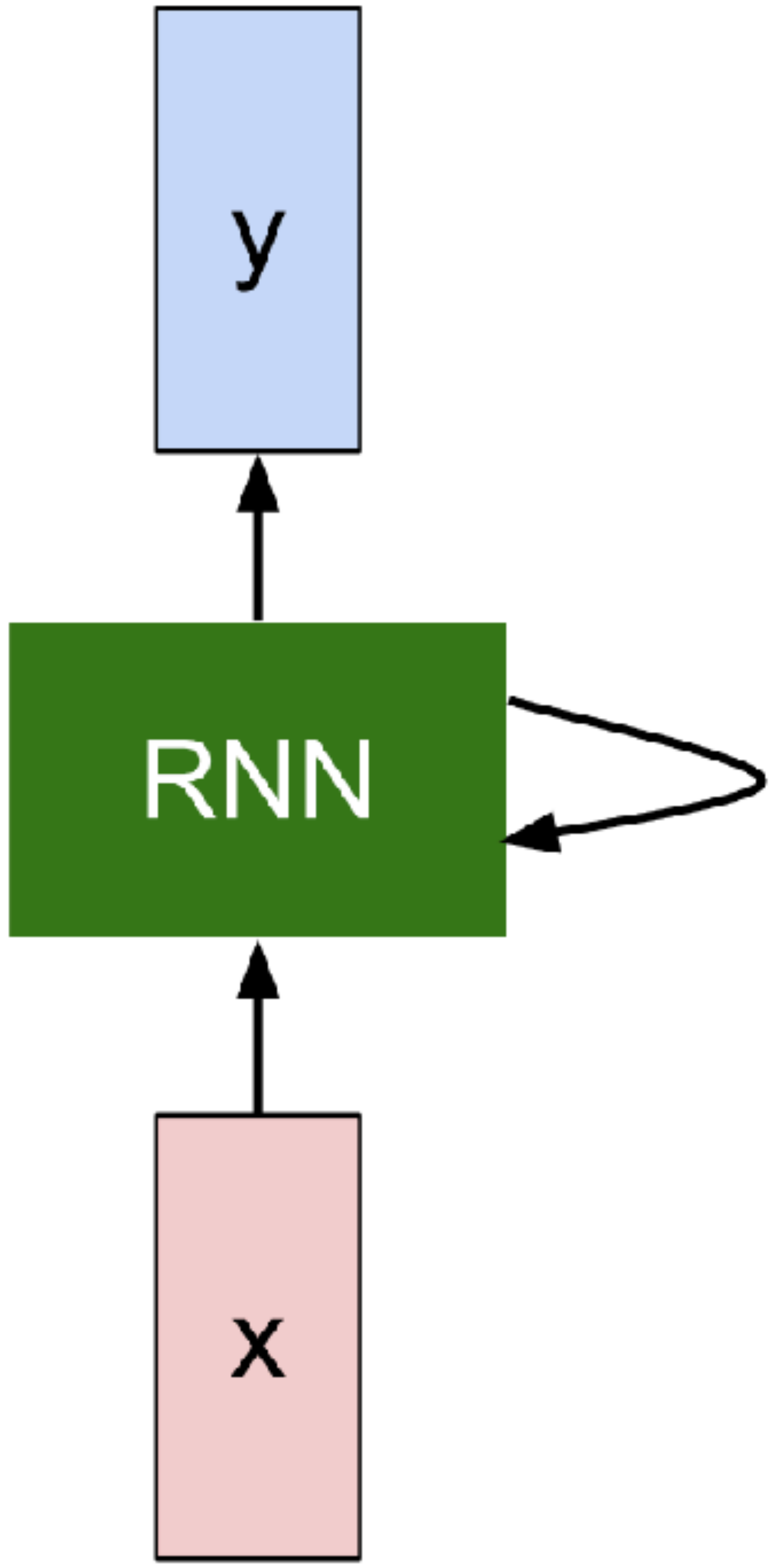


Recurrent Neural Network

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

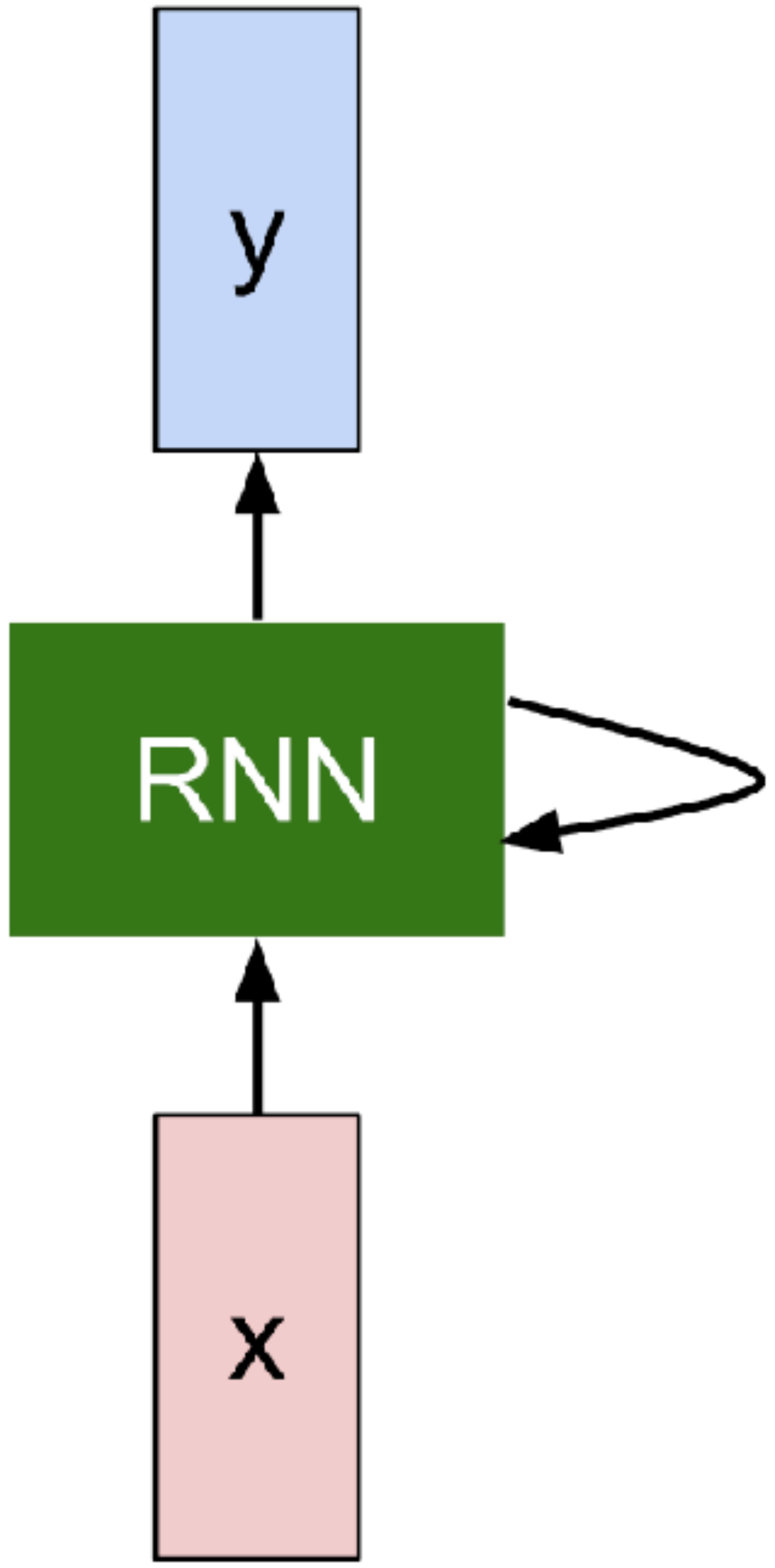
```
class RNN:
  # ...
  def step(self, x):
    # update the hidden state
    self.h = np.tanh(np.dot(self.W_hh, self.h) + np.dot(self.W_xh, x))
    # compute the output vector
    y = np.dot(self.W_hy, self.h)
    return y
```

with parameters W



(Vanilla) Recurrent Neural Network

The state consists of a single *"hidden"* vector **h**:

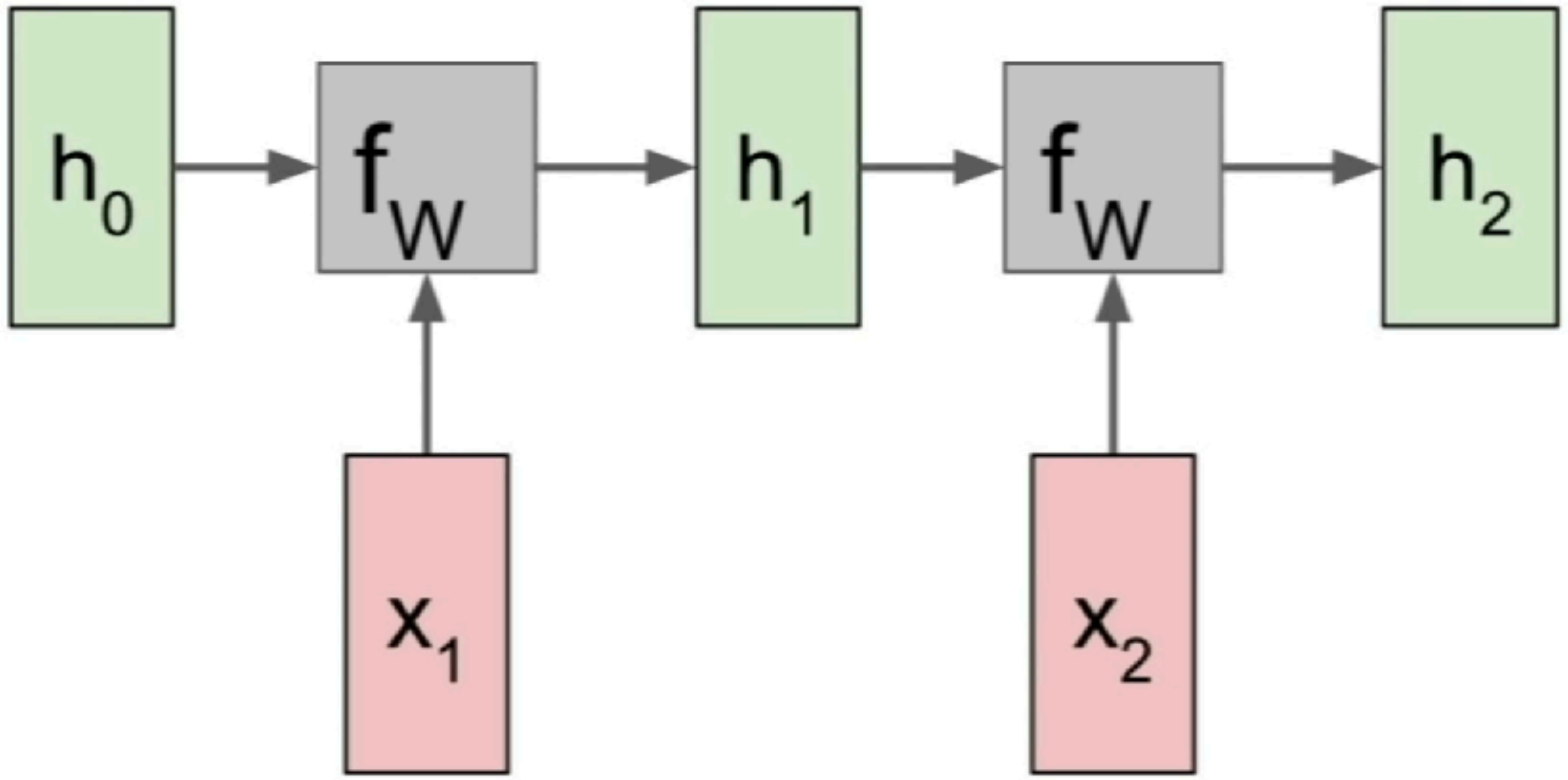


$$h_t = f_W(h_{t-1}, x_t)$$

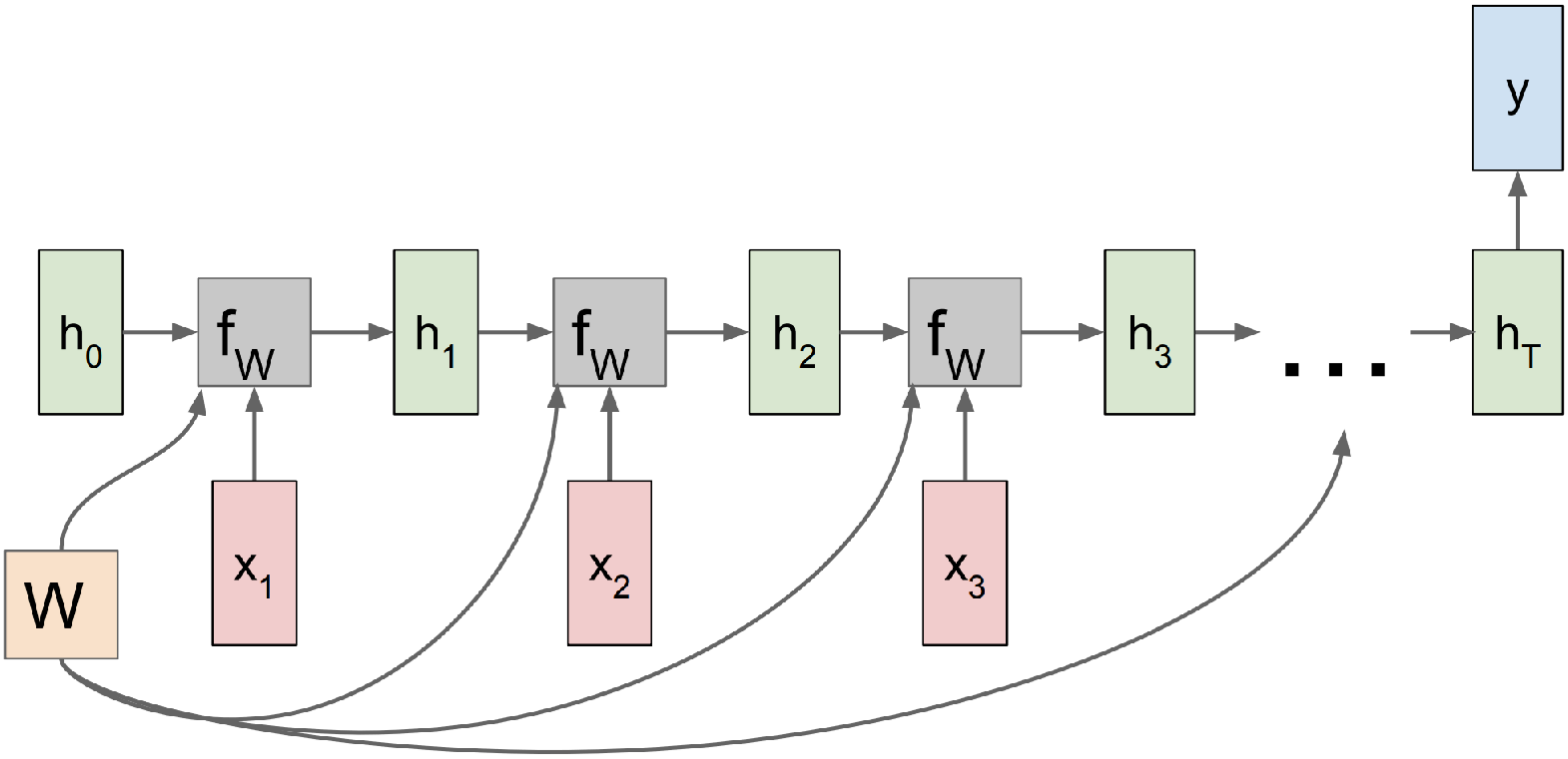
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

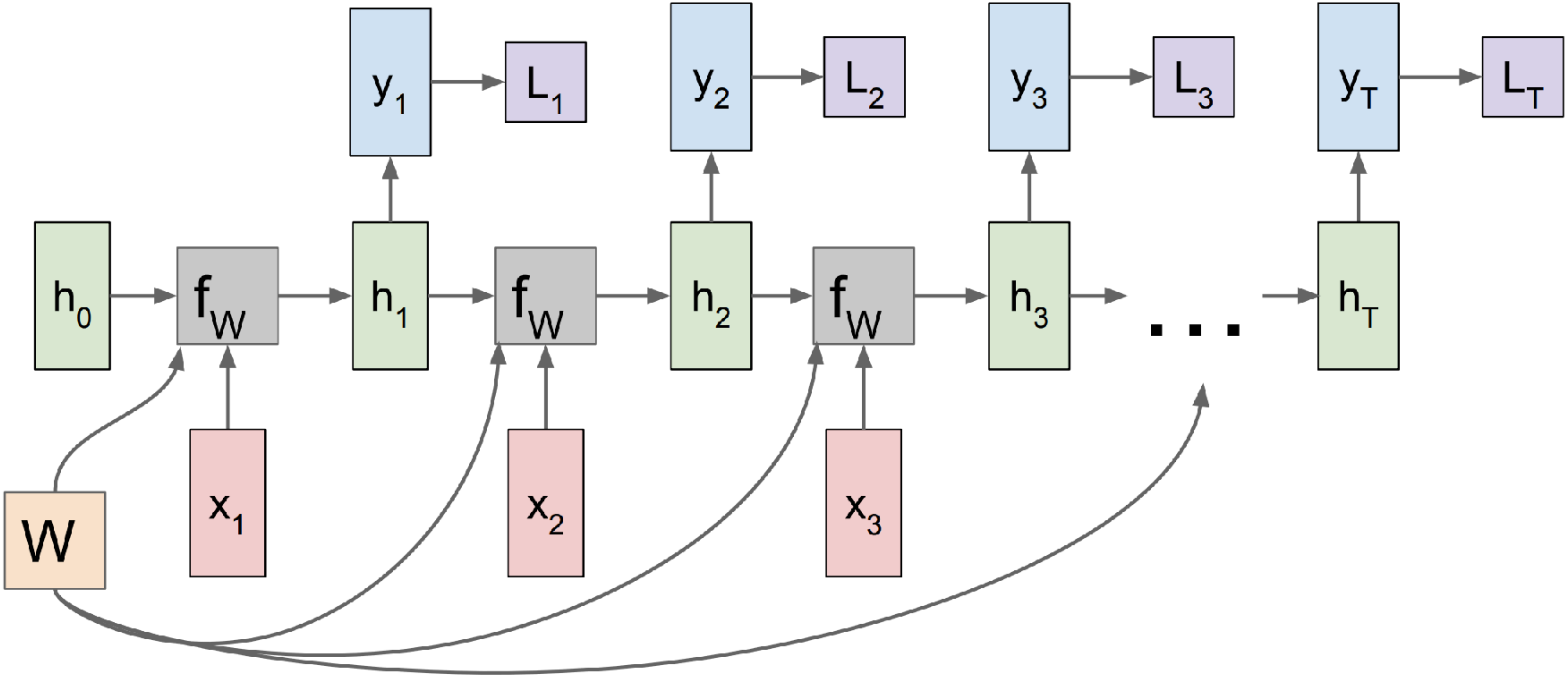
RNN: Computational Graph



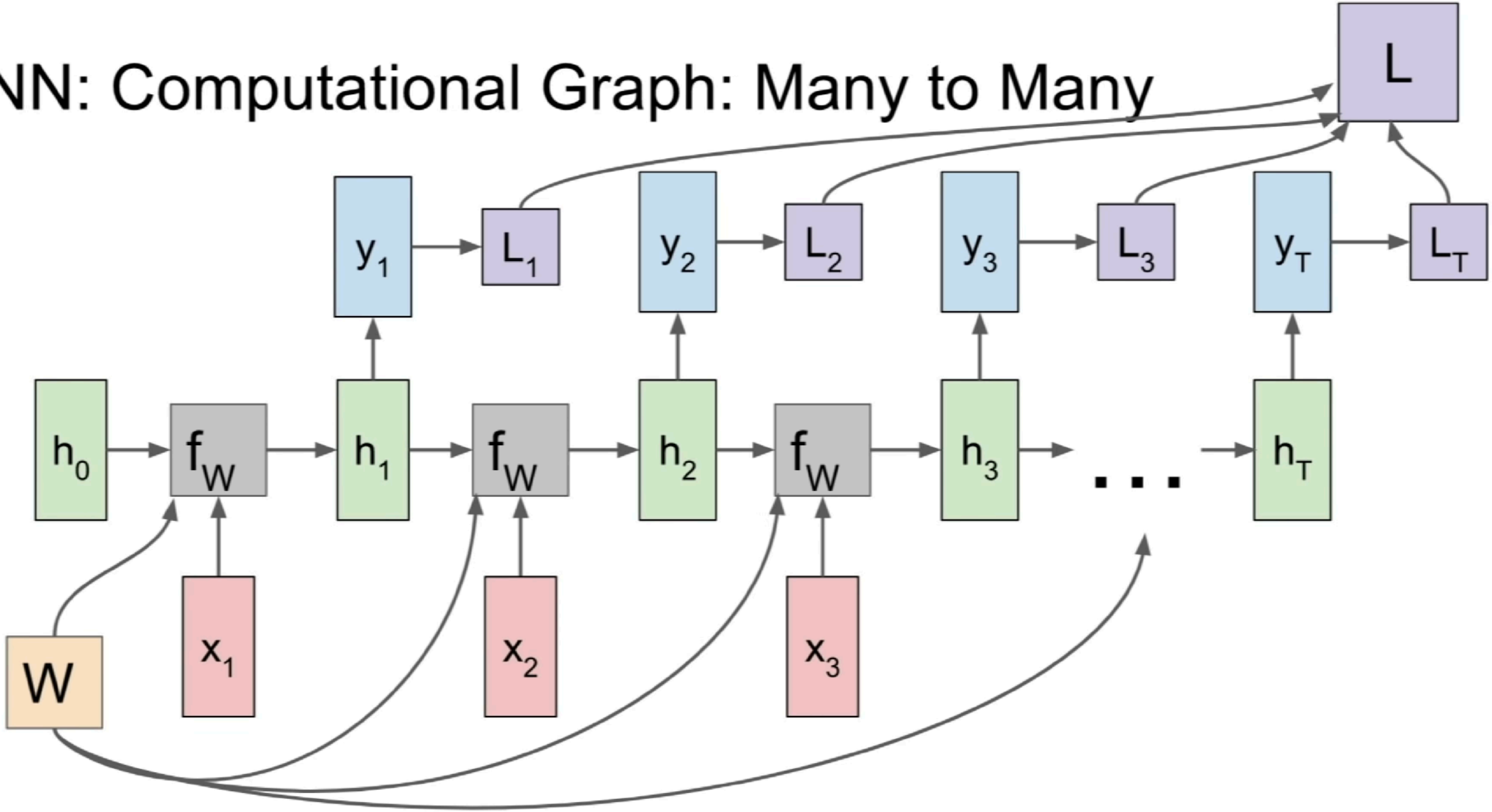
RNN: Computational Graph: Many to One



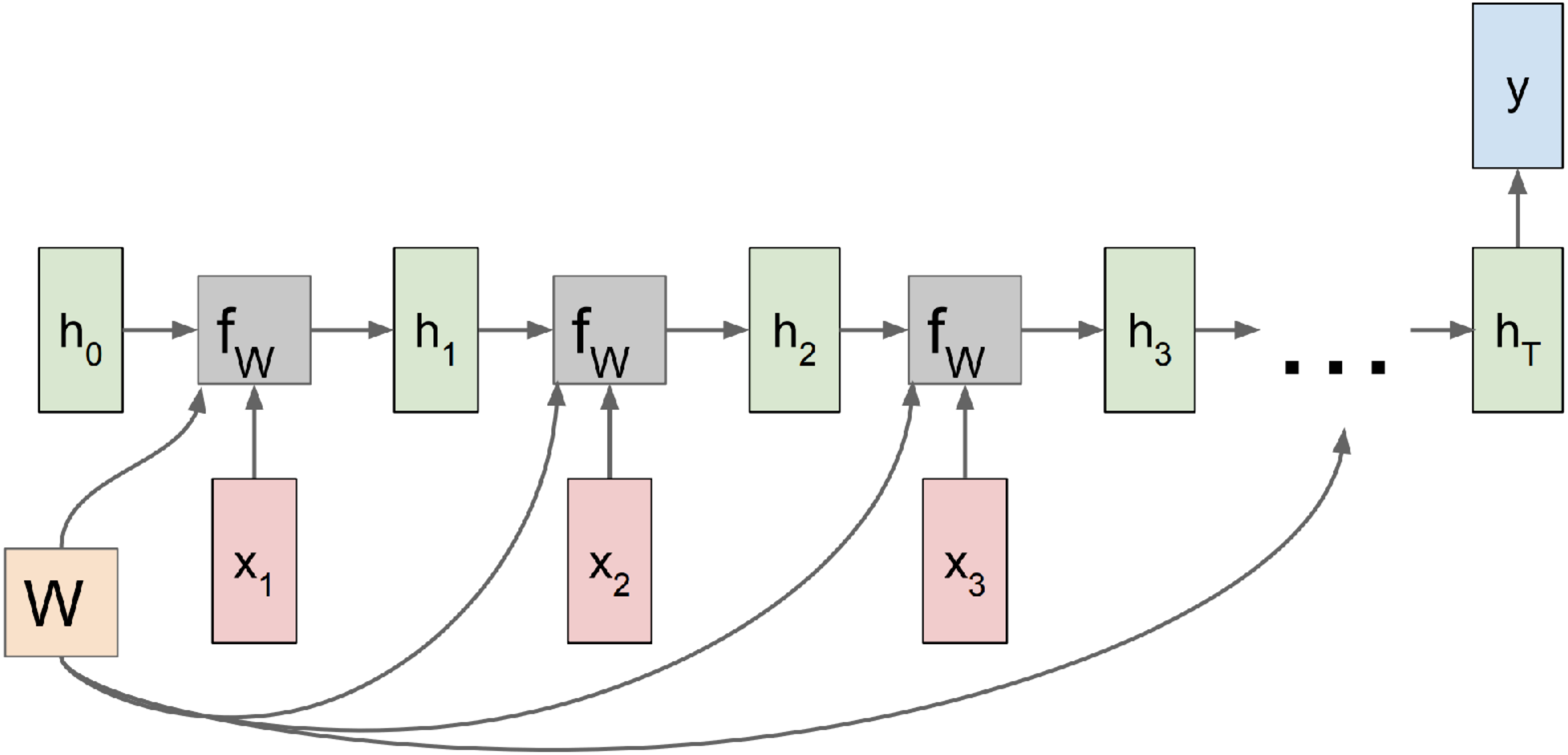
RNN: Computational Graph: Many to Many



RNN: Computational Graph: Many to Many



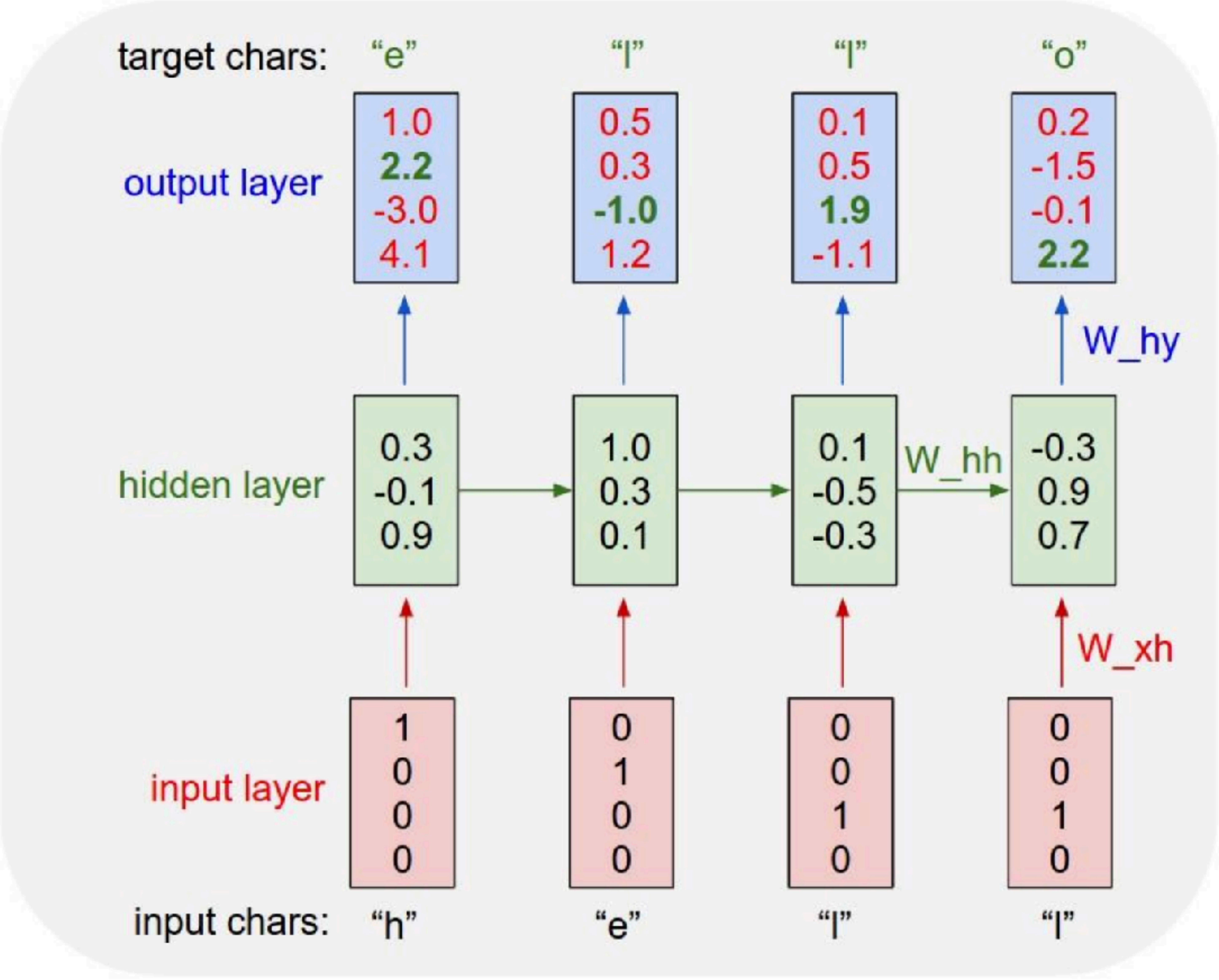
RNN: Computational Graph: Many to One



Example: Character-level Language Model

Vocabulary: [h,e,l,o]

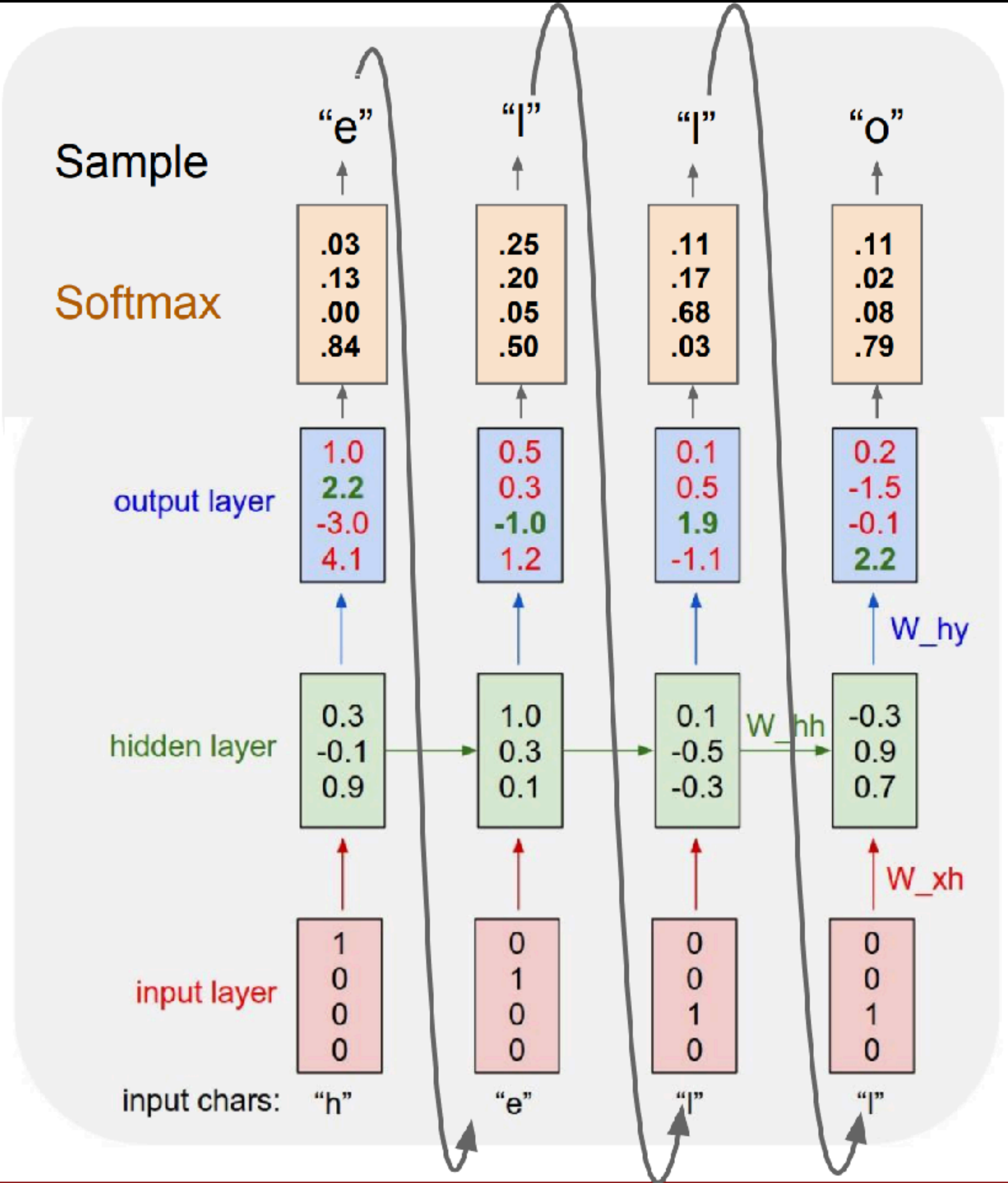
Example training sequence: **“hello”**



Example: Character-level Language Model Sampling

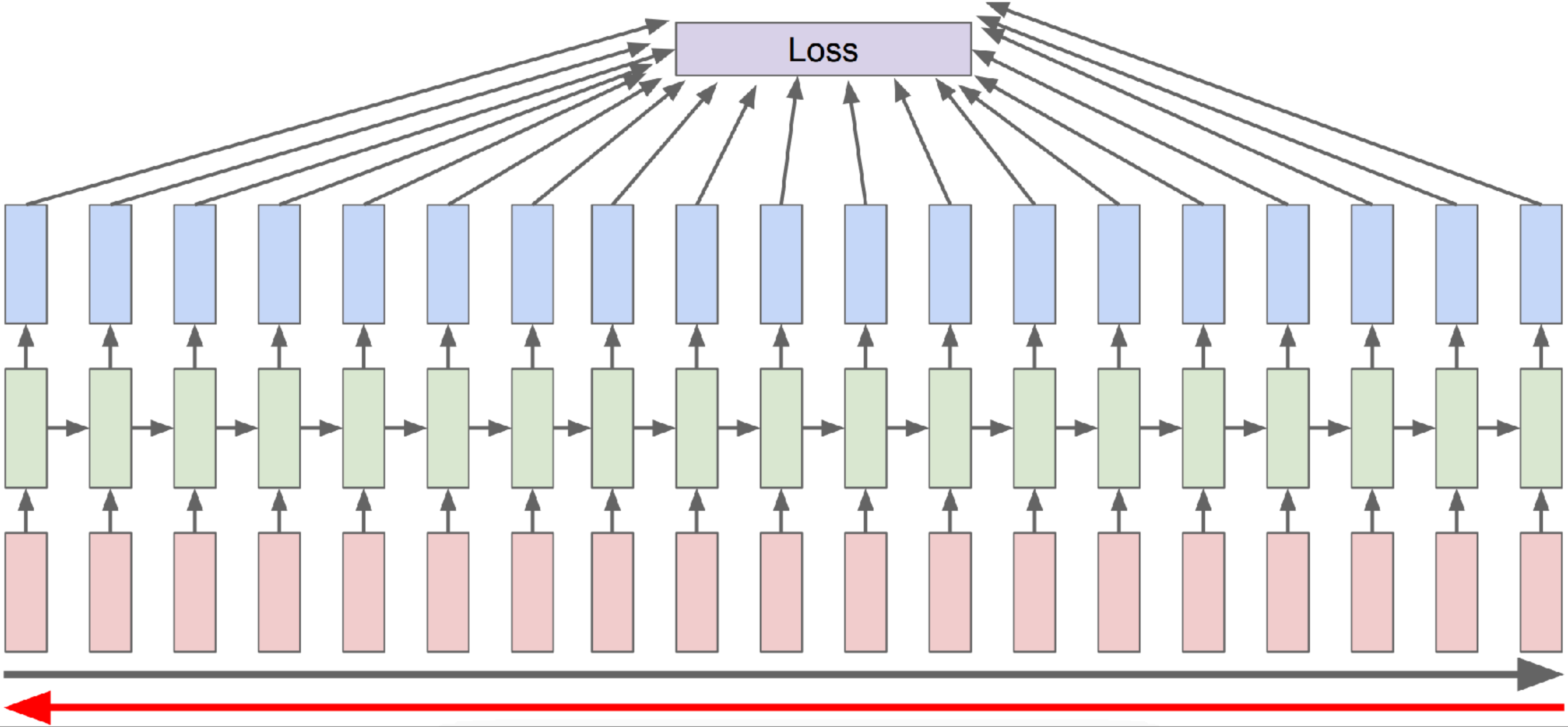
Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model

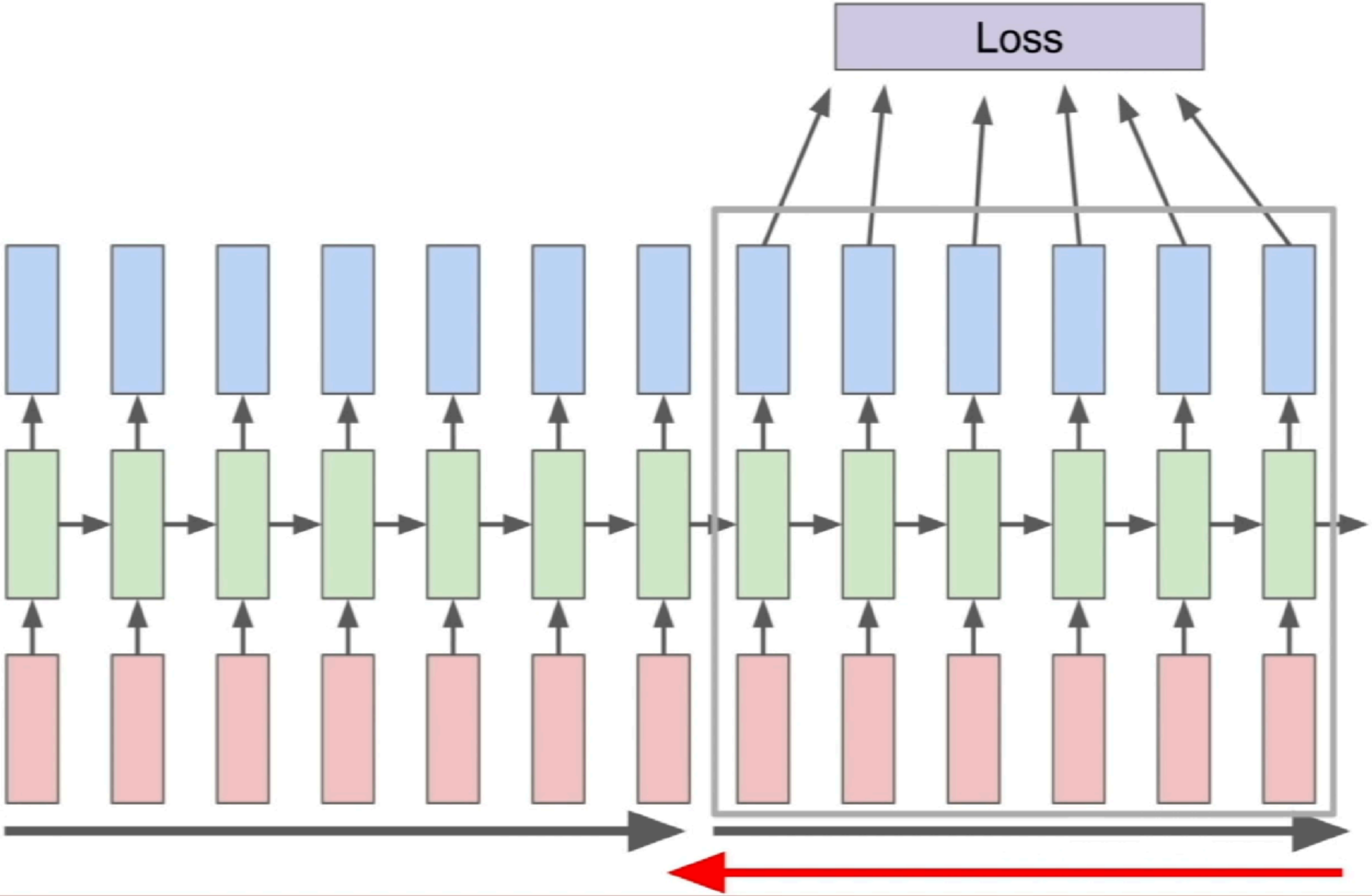


Backpropagation through time

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient



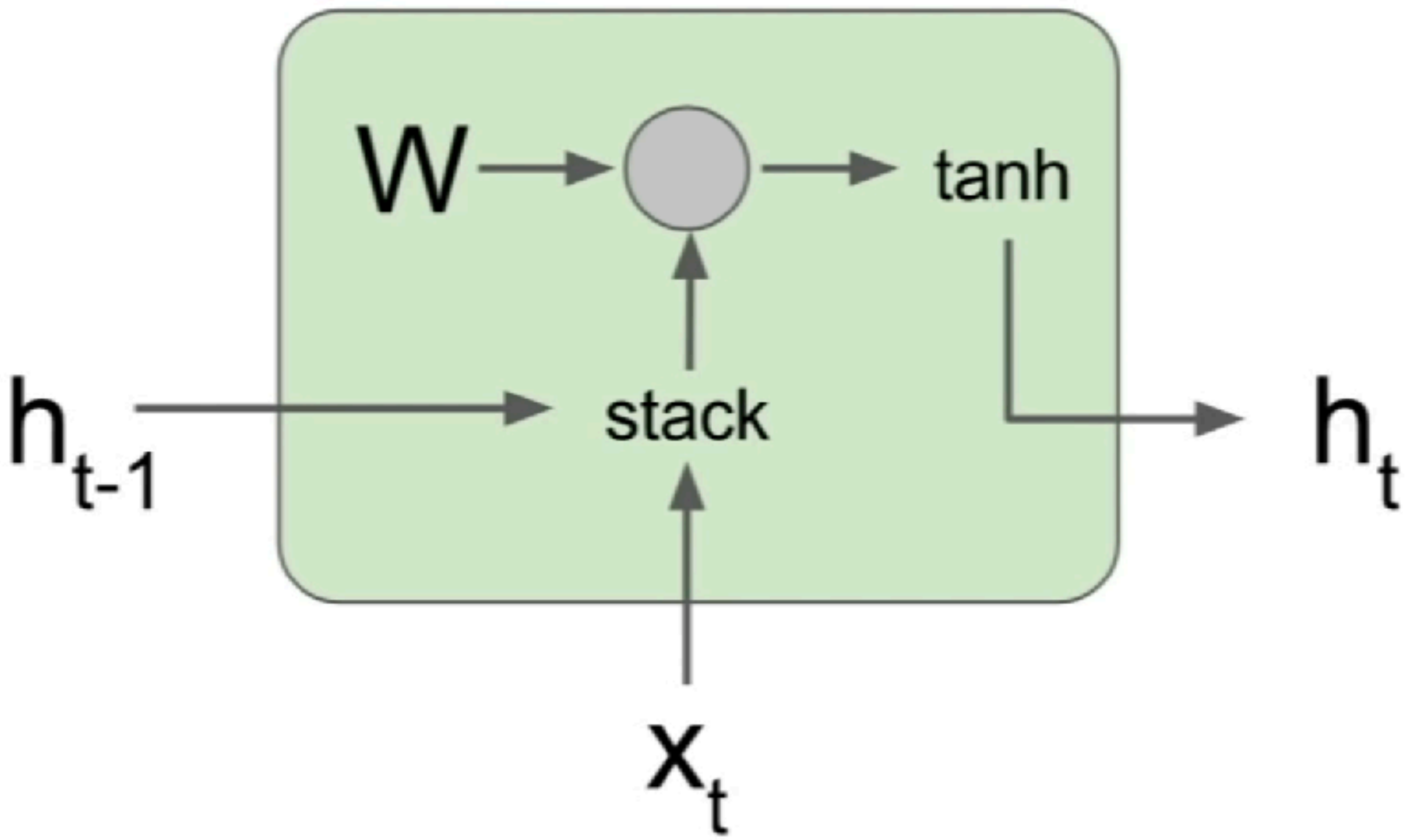
Truncated Backpropagation through time



Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013

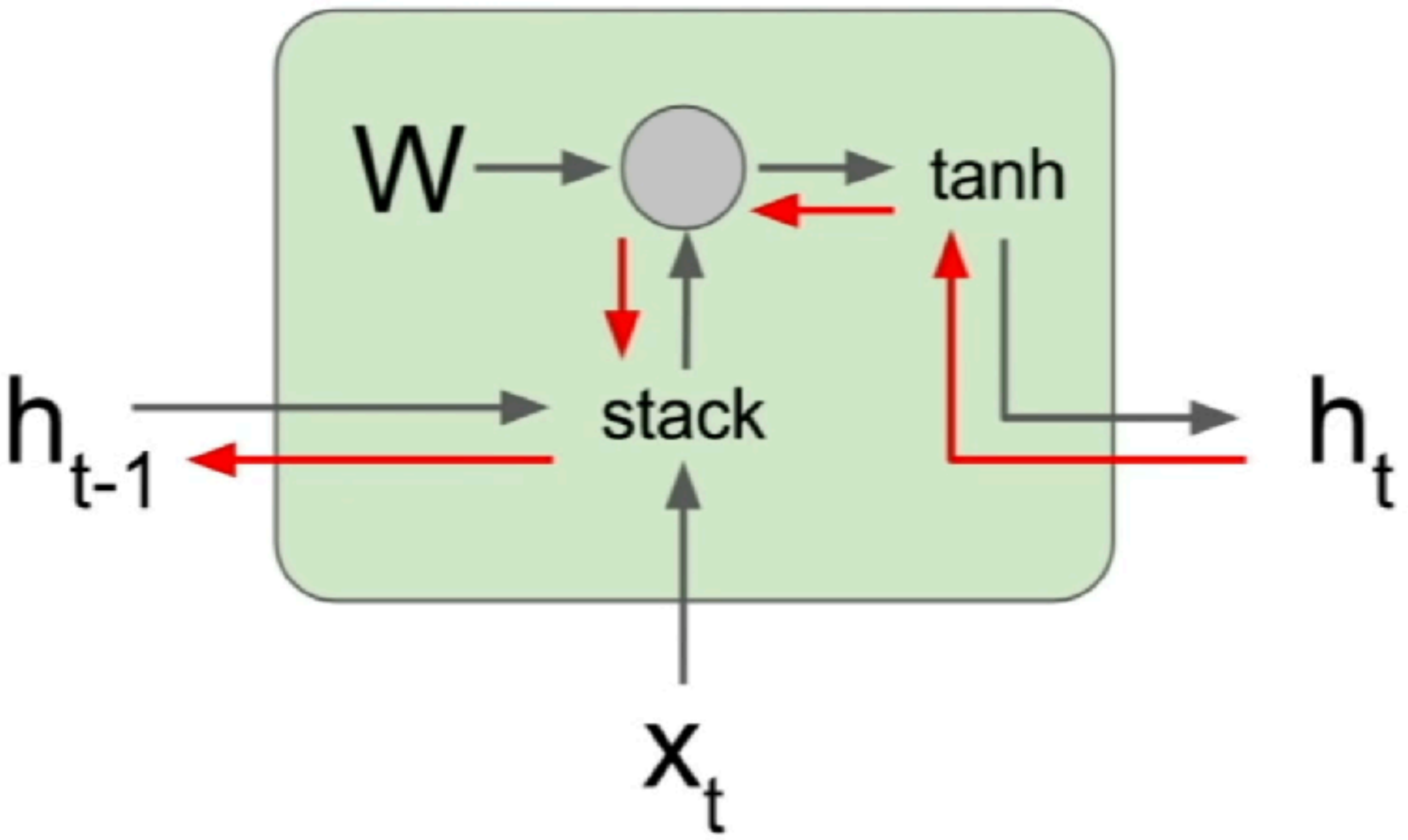


$$
\begin{aligned}
h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\
&= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right) \\
&= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)
\end{aligned}
$$

Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013

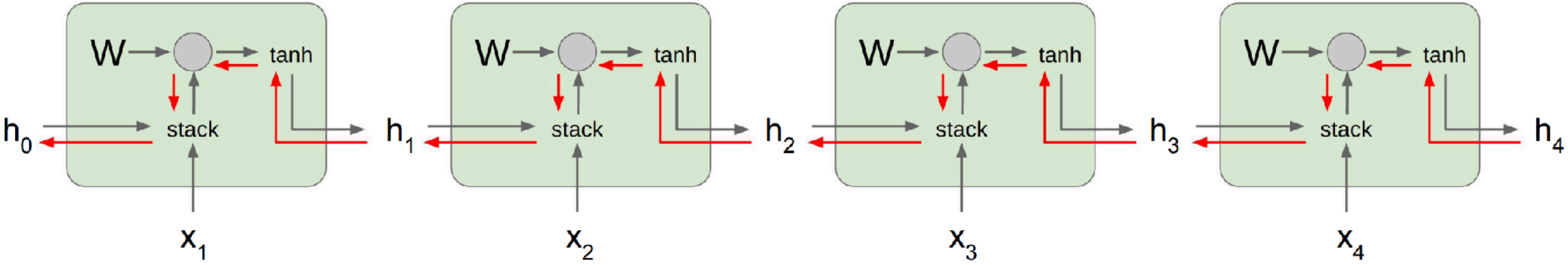
Backpropagation from h_t to h_{t-1} multiplies by W (actually W_{hh}^T)



$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$
$$= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013



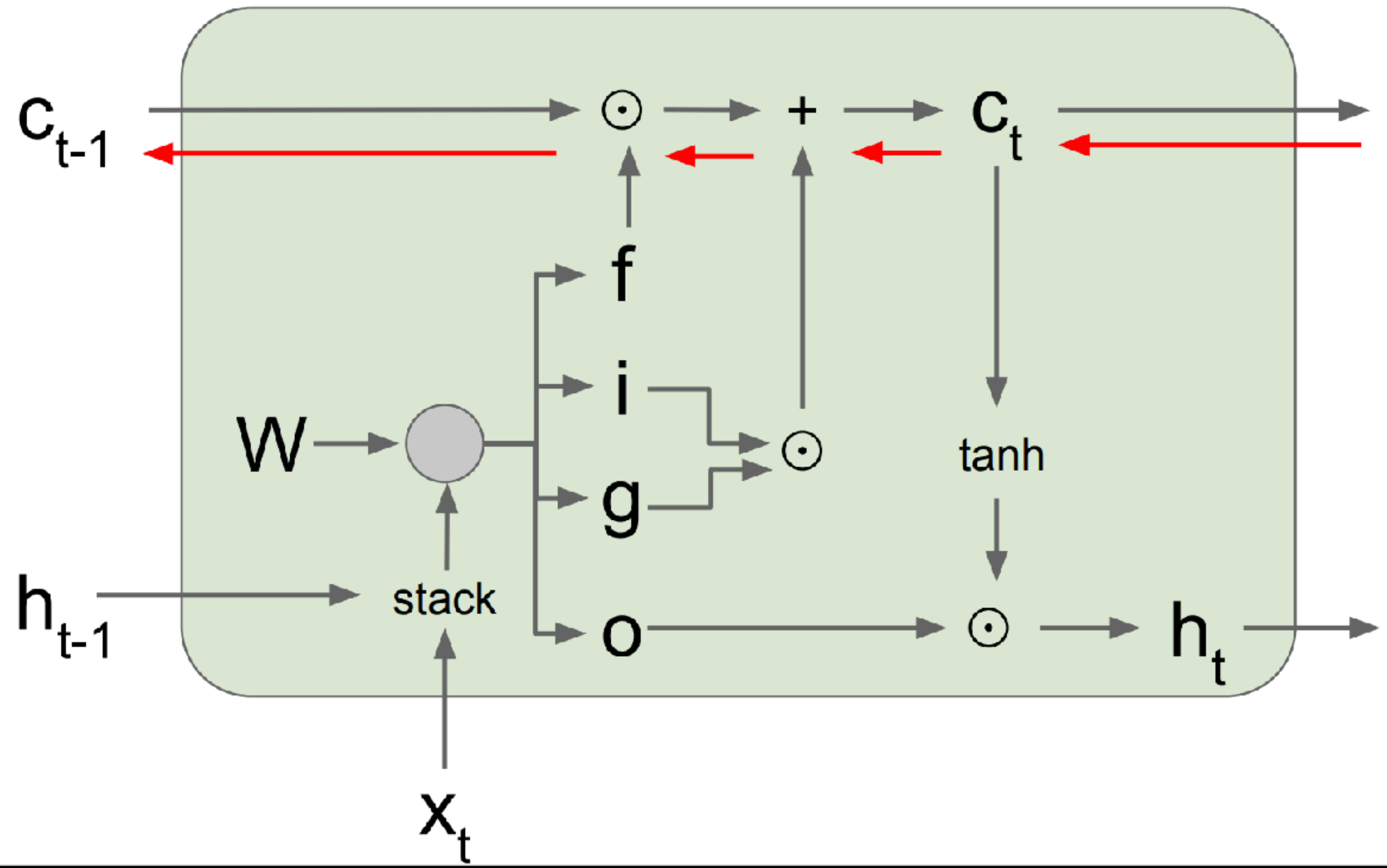
Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1:
Exploding gradients

Largest singular value < 1:
Vanishing gradients

Long Short Term Memory (LSTM): Gradient Flow

[Hochreiter et al., 1997]



Backpropagation from c_t to c_{t-1} only elementwise multiplication by f, no matrix multiply by W

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

Long Short Term Memory (LSTM)

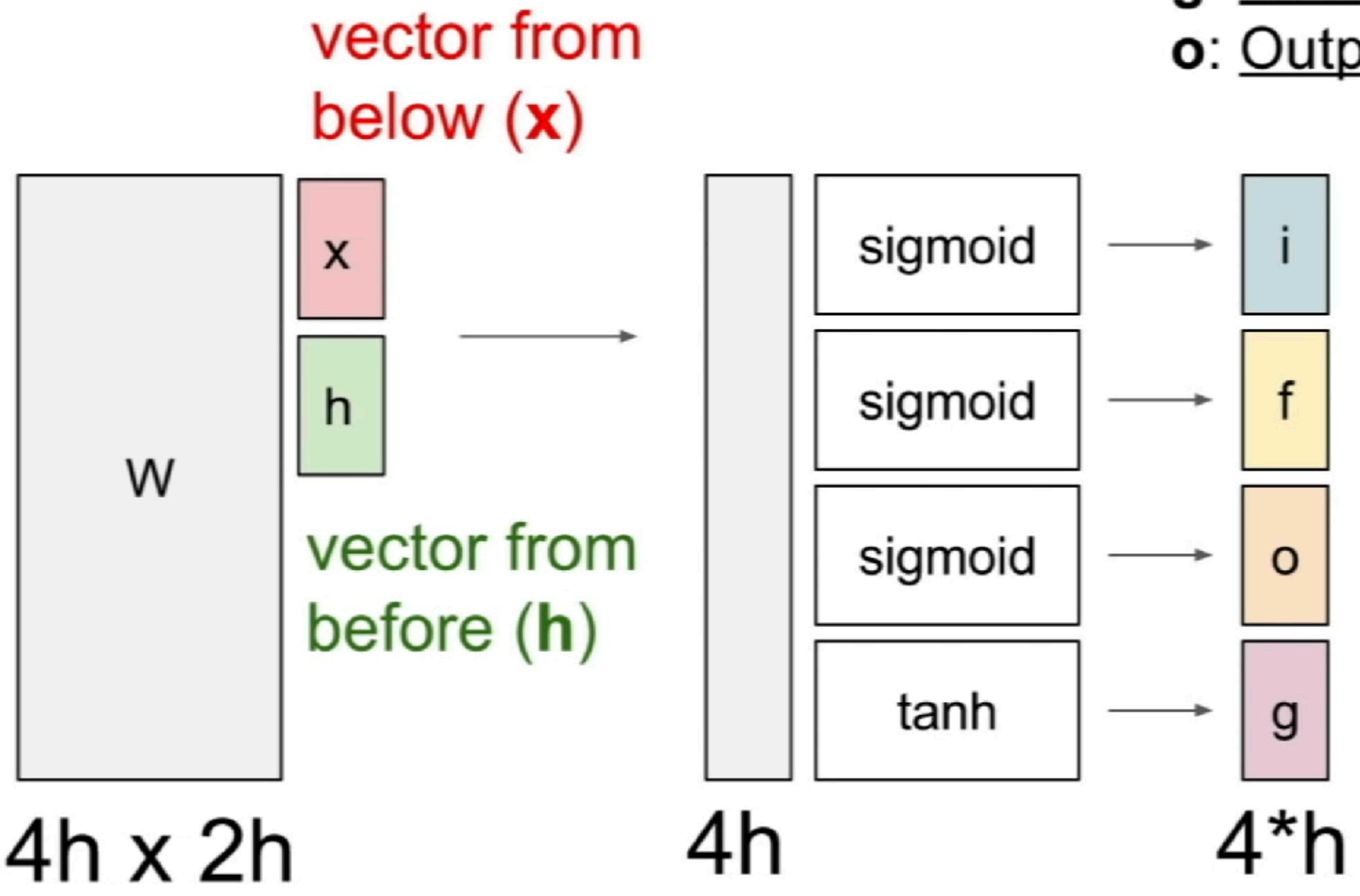
[Hochreiter et al., 1997]

f: Forget gate, Whether to erase cell

i: Input gate, whether to write to cell

g: Gate gate (?), How much to write to cell

o: Output gate, How much to reveal cell



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

Long Short Term Memory (LSTM): Gradient Flow

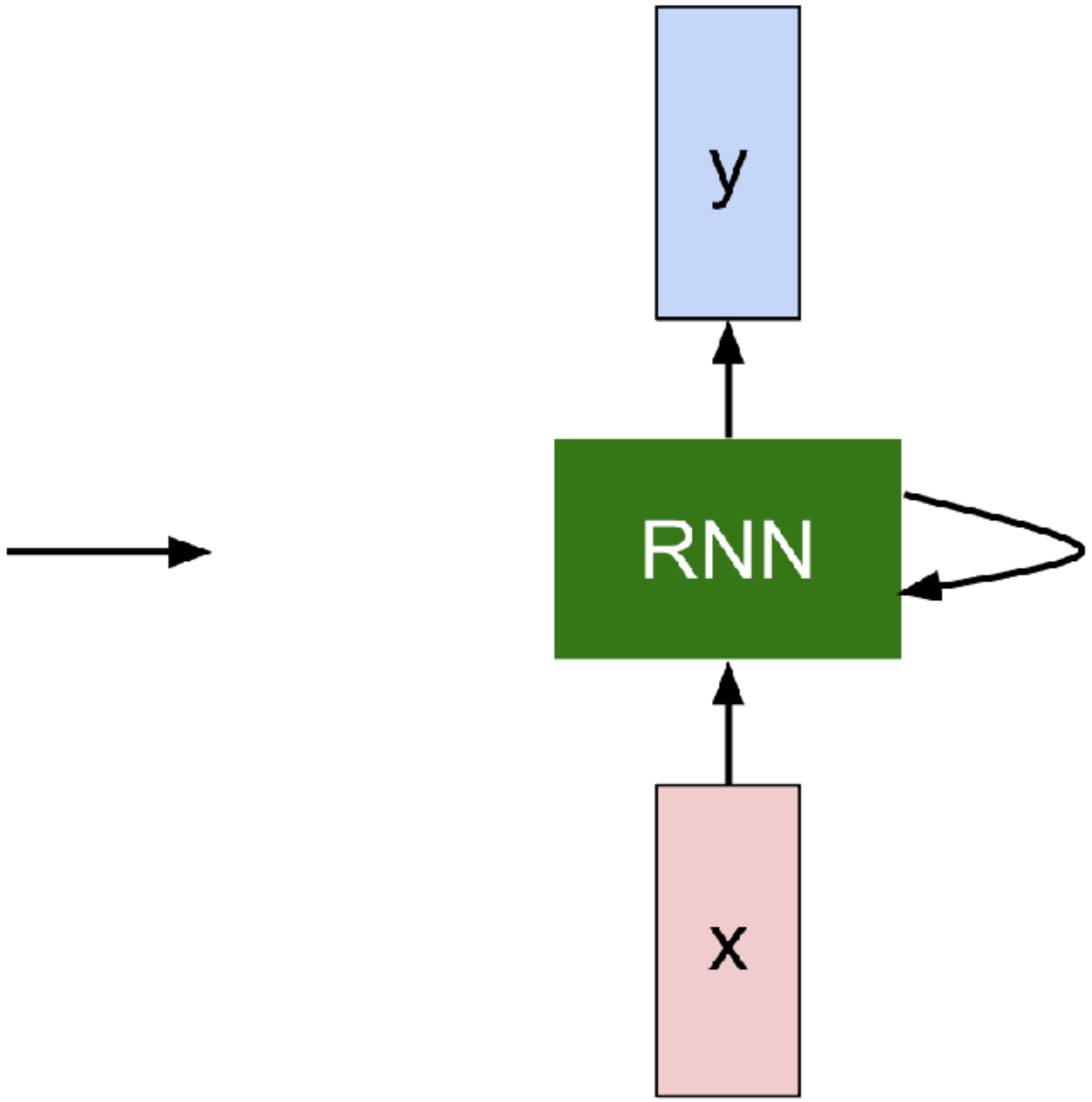
[Hochreiter et al., 1997]

THE SONNETS

by William Shakespeare

From fairest creatures we desire increase,
That thereby beauty's rose might never die,
But as the riper should by time decease,
His tender heir might bear his memory:
But thou, contracted to thine own bright eyes,
Feed'st thy light's flame with self-substantial fuel,
Making a famine where abundance lies,
Thyself thy foe, to thy sweet self too cruel:
Thou that art now the world's fresh ornament,
And only herald to the gaudy spring,
Within thine own bud buriest thy content,
And tender churl mak'st waste in niggarding:
  Pity the world, or else this glutton be,
  To eat the world's due, by the grave and thee.

When forty winters shall besiege thy brow,
And dig deep trenches in thy beauty's field,
Thy youth's proud livery so gazed on now,
Will be a tatter'd weed of small worth held:
Then being asked, where all thy beauty lies,
Where all the treasure of thy lusty days;
To say, within thine own deep sunken eyes,
Were an all-eating shame, and thriftless praise.
How much more praise deserv'd thy beauty's use,
If thou couldst answer 'This fair child of mine
Shall sum my count, and make my old excuse,'
Proving his beauty by succession thine!
  This were to be new made when thou art old,
  And see thy blood warm when thou feel'st it cold.



at first:

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

↓ train more

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

↓ train more

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

↓ train more

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m,\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \text{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, **??** and the fact that any U affine, see Morphisms, Lemma **??**. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\text{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma **??** we can define a map of complexes $\text{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\text{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\text{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \text{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example **??**. It may replace S by $X_{spaces,\acute{e}tale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma **??**. Namely, by Lemma **??** we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\underline{Proj}_X(\mathcal{A}) = \text{Spec}(B)$ over U compatible with the complex

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition **??** *(without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S. Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem*

(1) f is locally of finite type. Since $S = \text{Spec}(R)$ and $Y = \text{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{X,\ldots,0}$.

Lemma 0.2. *Let X be a locally Noetherian scheme over S, $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq \mathfrak{p}$ is a subset of $\mathcal{J}_{n,0} \circ \overline{A}_2$ works.*

Lemma 0.3. *In Situation* **??**. *Hence we may assume $\mathfrak{q}' = 0$.*

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (**??**). On the other hand, by Lemma **??** we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

```c
static void do_command(struct seq_file *m, void *v)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
    cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
  else
    seq = 1;
  for (i = 0; i < 16; i++) {
    if (k & (1 << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
        ((count & 0x00000000fffffff8) & 0x000000f) << 8;
    if (count == 0)
      sub(pid, ppc_md.kexec_handle, 0x20000000);
    pipe_set_bytes(i, 0);
  }
  /* Free our user pages pointer to place camera if all dash */
  subsystem_info = &of_changes[PAGE_SIZE];
  rek_controls(offset, idx, &soffset);
  /* Now we want to deliberately put it to device */
  control_check_polarity(&context, val, 0);
  for (i = 0; i < COUNTER; i++)
    seq_puts(s, "policy ");
}
```

Generated C code

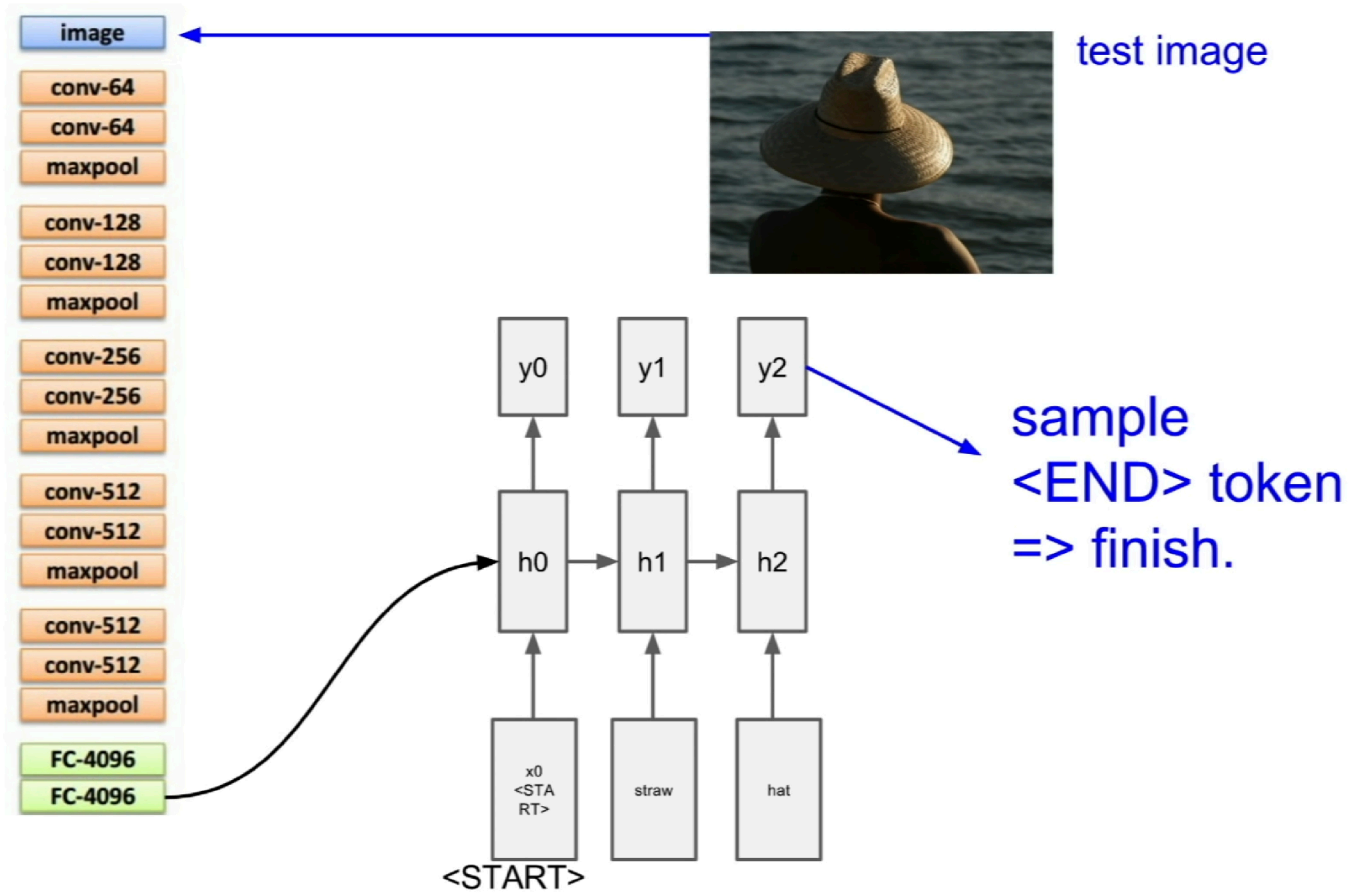


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
y2
h0
h1
h2
x0 <STA RT>
straw
hat
<START>
sample <END> token => finish.

Image Captioning: Example Results

Captions generated using neuraltalk2
All images are CC0 Public domain: cat suitcase, cat tree, dog, bear, surfers, tennis, giraffe, motorcycle

A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

Two people walking on the beach with surfboards

A tennis player in action on the court

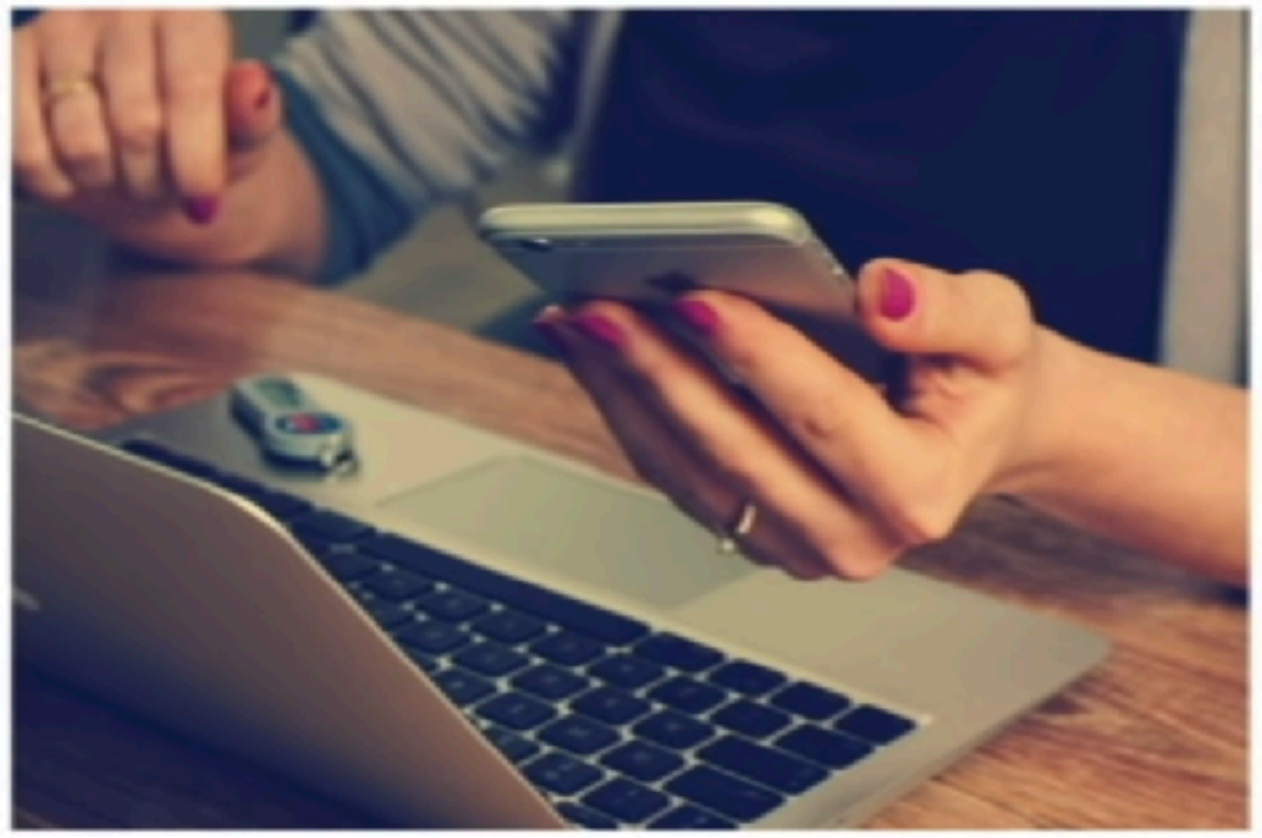
Two giraffes standing in a grassy field

A man riding a dirt bike on a dirt track

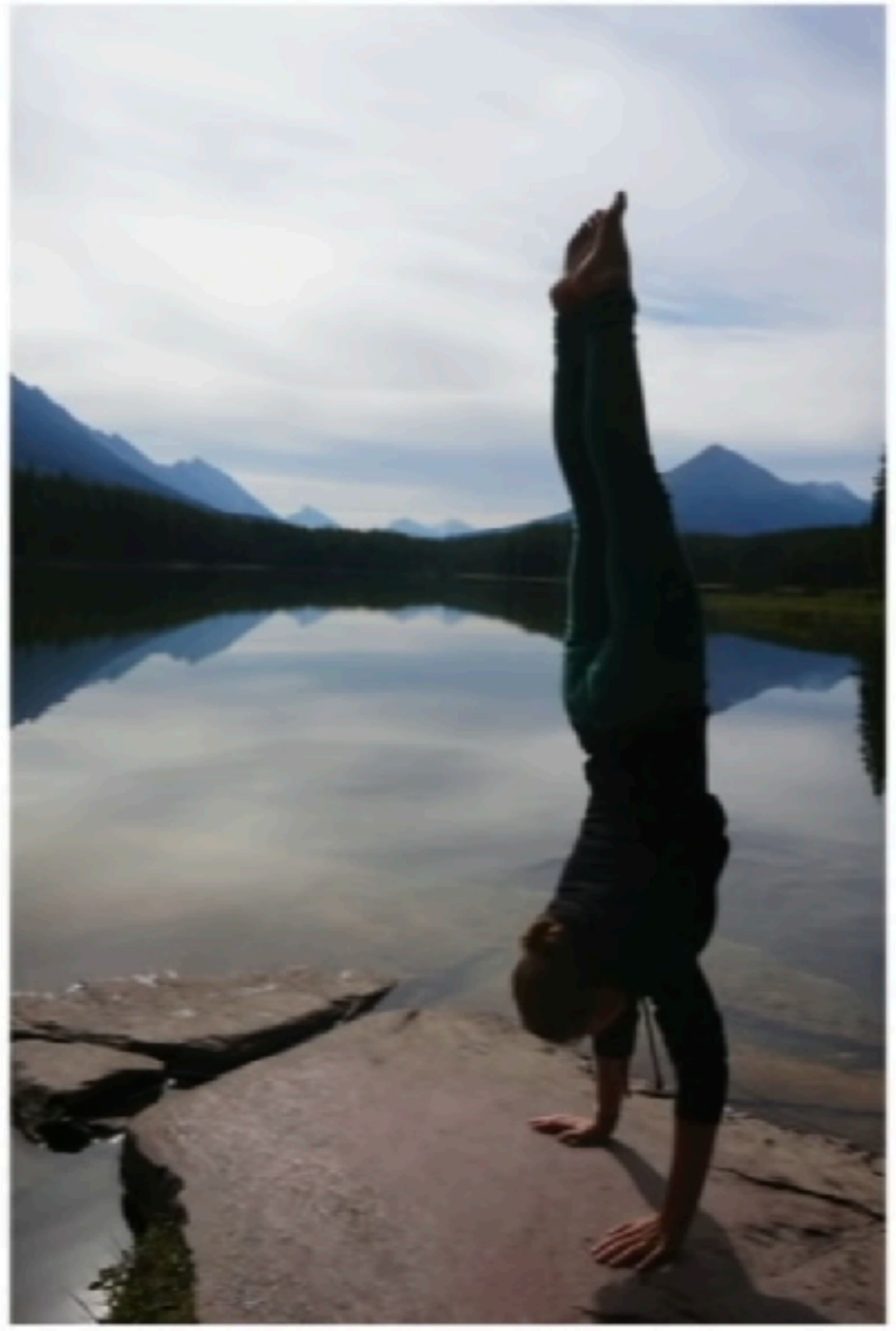
Image Captioning: Failure Cases

Captions generated using neuraltalk2
All images are CC0 Public domain: fur coat, handstand, spider web, baseball

A woman is holding a cat in her hand



A person holding a computer mouse on a desk

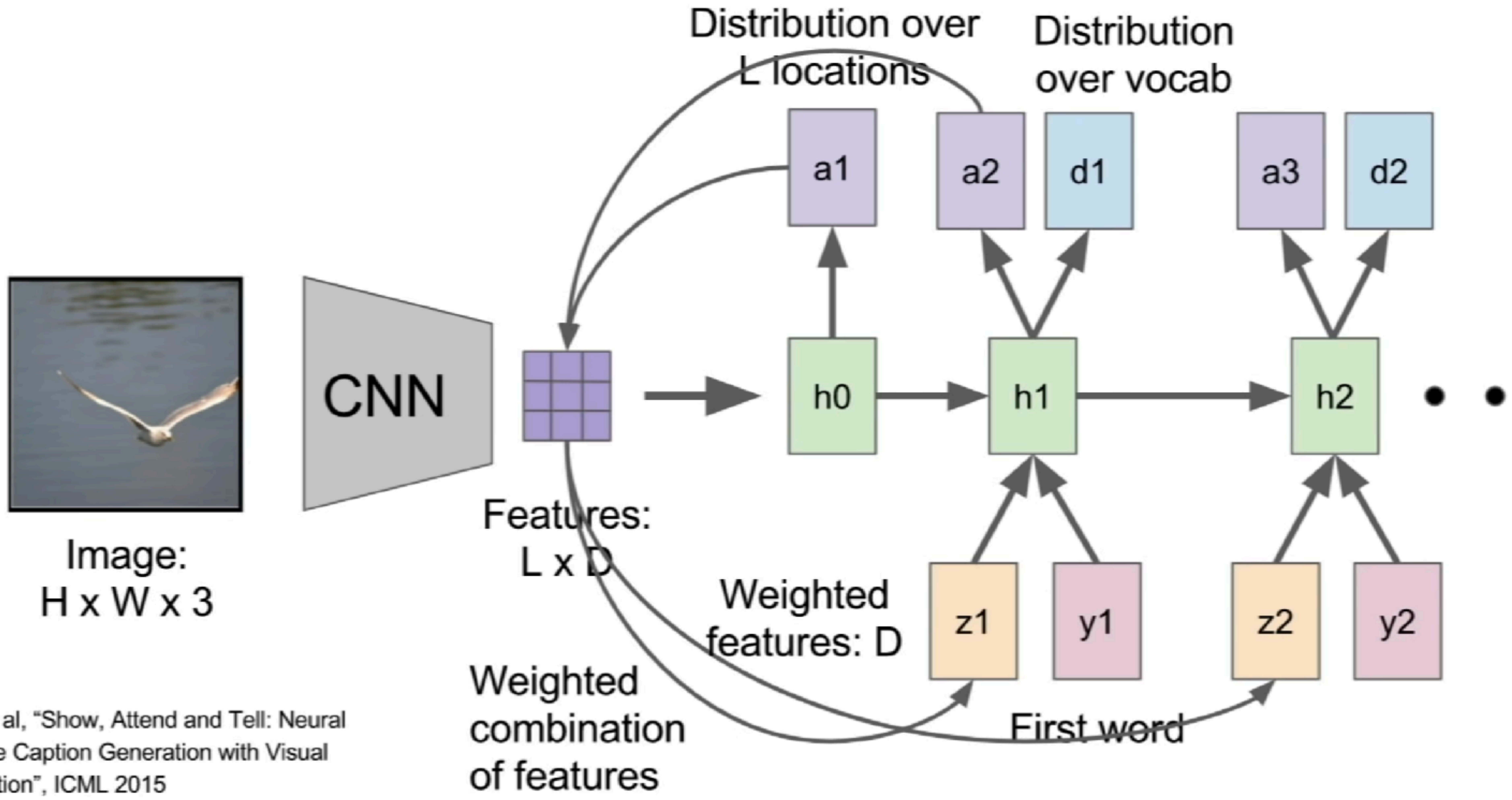


A woman standing on a beach holding a surfboard

A bird is perched on a tree branch

A man in a baseball uniform throwing a ball

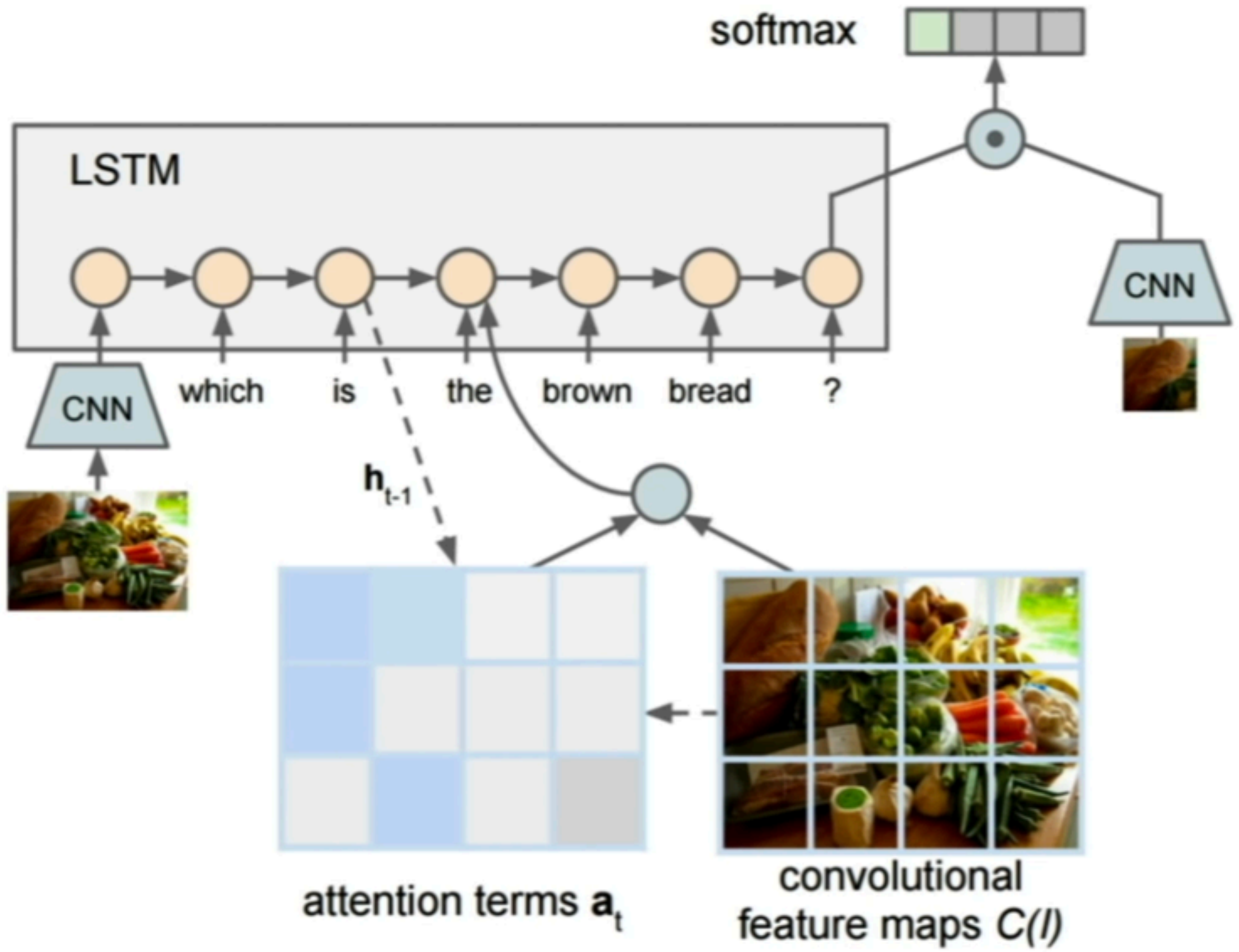
Image Captioning with Attention

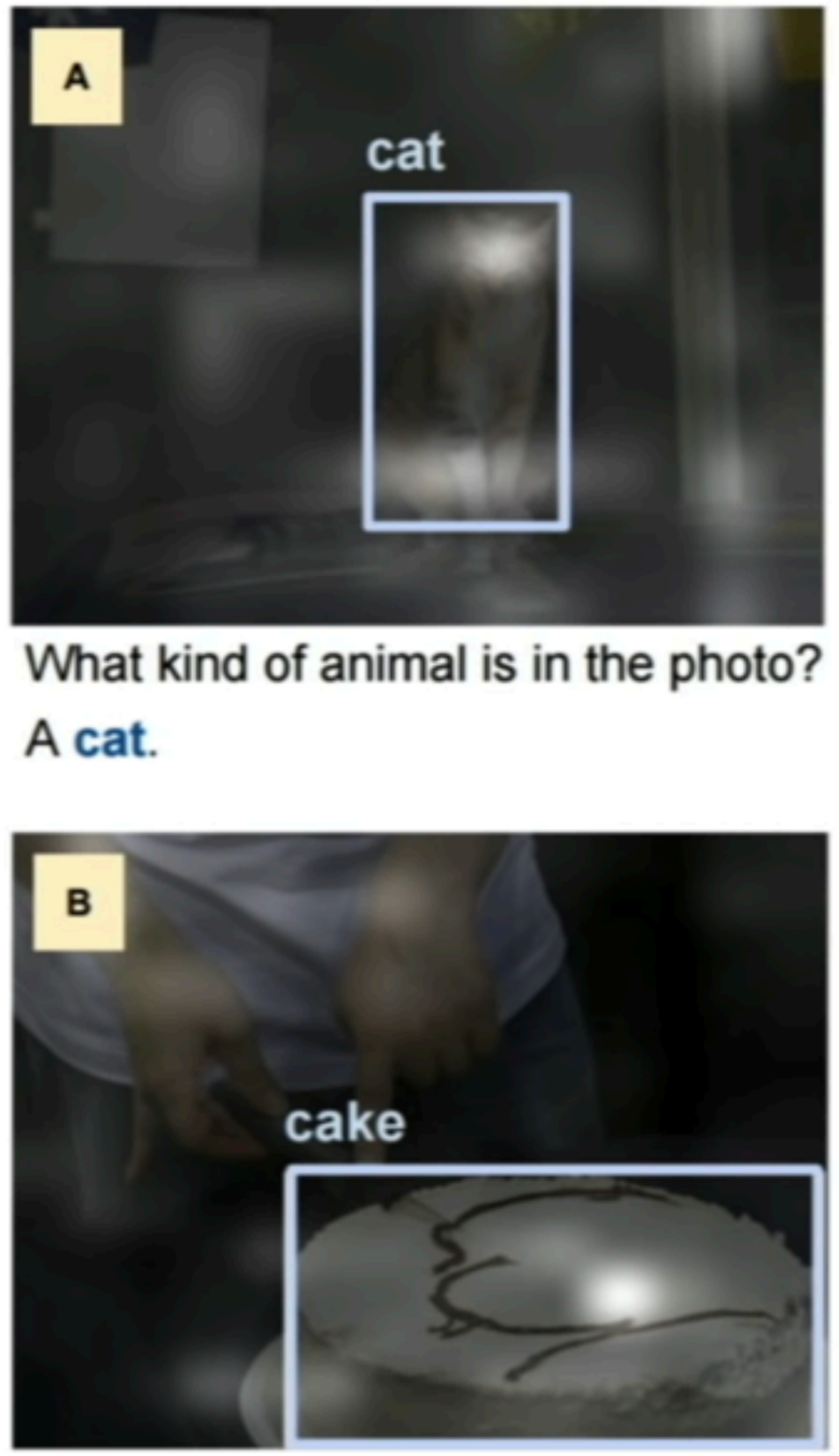


Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Image Captioning with Attention

Visual Question Answering: RNNs with Attention





Zhu et al, "Visual 7W: Grounded Question Answering in Images", CVPR 2016

Xu et al, "Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Multilayer RNNs

$$h_t^l = \tanh W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

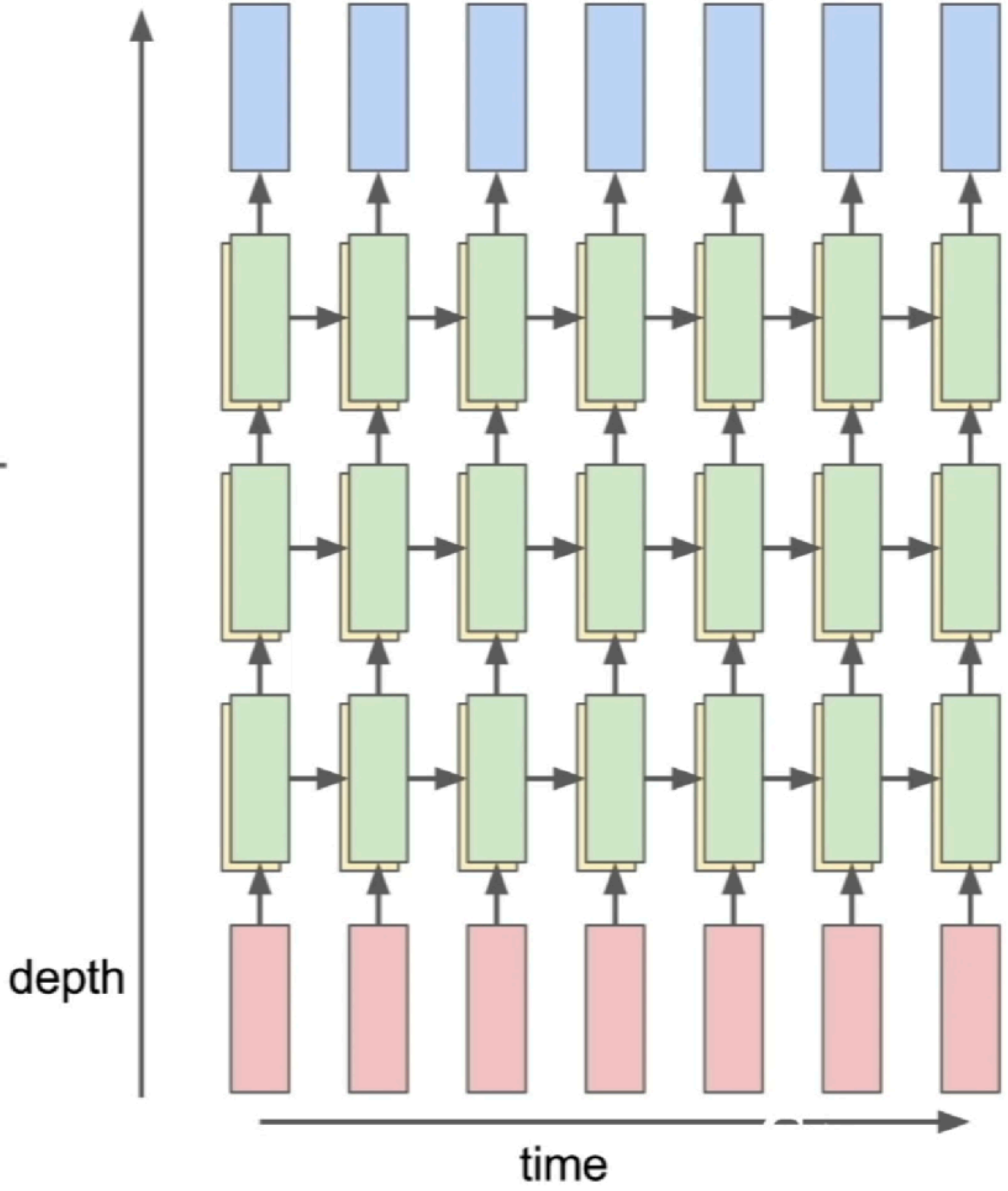
$h \in \mathbb{R}^n$ $\quad W^l \ [n \times 2n]$

LSTM:

$W^l \ [4n \times 2n]$

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

$$c_t^l = f \odot c_{t-1}^l + i \odot g$$
$$h_t^l = o \odot \tanh(c_t^l)$$



Other RNN Variants

GRU [*Learning phrase representations using rnn encoder-decoder for statistical machine translation*, Cho et al. 2014]

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)$$
$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)$$
$$\tilde{h}_t = \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h)$$
$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t$$

[*LSTM: A Search Space Odyssey*, Greff et al., 2015]

[*An Empirical Exploration of Recurrent Network Architectures*, Jozefowicz et al., 2015]

MUT1:

$$\begin{aligned} z &= \mathrm{sigm}(W_{\mathrm{xz}}x_t + b_{\mathrm{z}}) \\ r &= \mathrm{sigm}(W_{\mathrm{xr}}x_t + W_{\mathrm{hr}}h_t + b_{\mathrm{r}}) \\ h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + \tanh(x_t) + b_{\mathrm{h}}) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

MUT2:

$$\begin{aligned} z &= \mathrm{sigm}(W_{\mathrm{xz}}x_t + W_{\mathrm{hz}}h_t + b_{\mathrm{z}}) \\ r &= \mathrm{sigm}(x_t + W_{\mathrm{hr}}h_t + b_{\mathrm{r}}) \\ h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + W_{xh}x_t + b_{\mathrm{h}}) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

MUT3:

$$\begin{aligned} z &= \mathrm{sigm}(W_{\mathrm{xz}}x_t + W_{\mathrm{hz}}\tanh(h_t) + b_{\mathrm{z}}) \\ r &= \mathrm{sigm}(W_{\mathrm{xr}}x_t + W_{\mathrm{hr}}h_t + b_{\mathrm{r}}) \\ h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + W_{xh}x_t + b_{\mathrm{h}}) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$