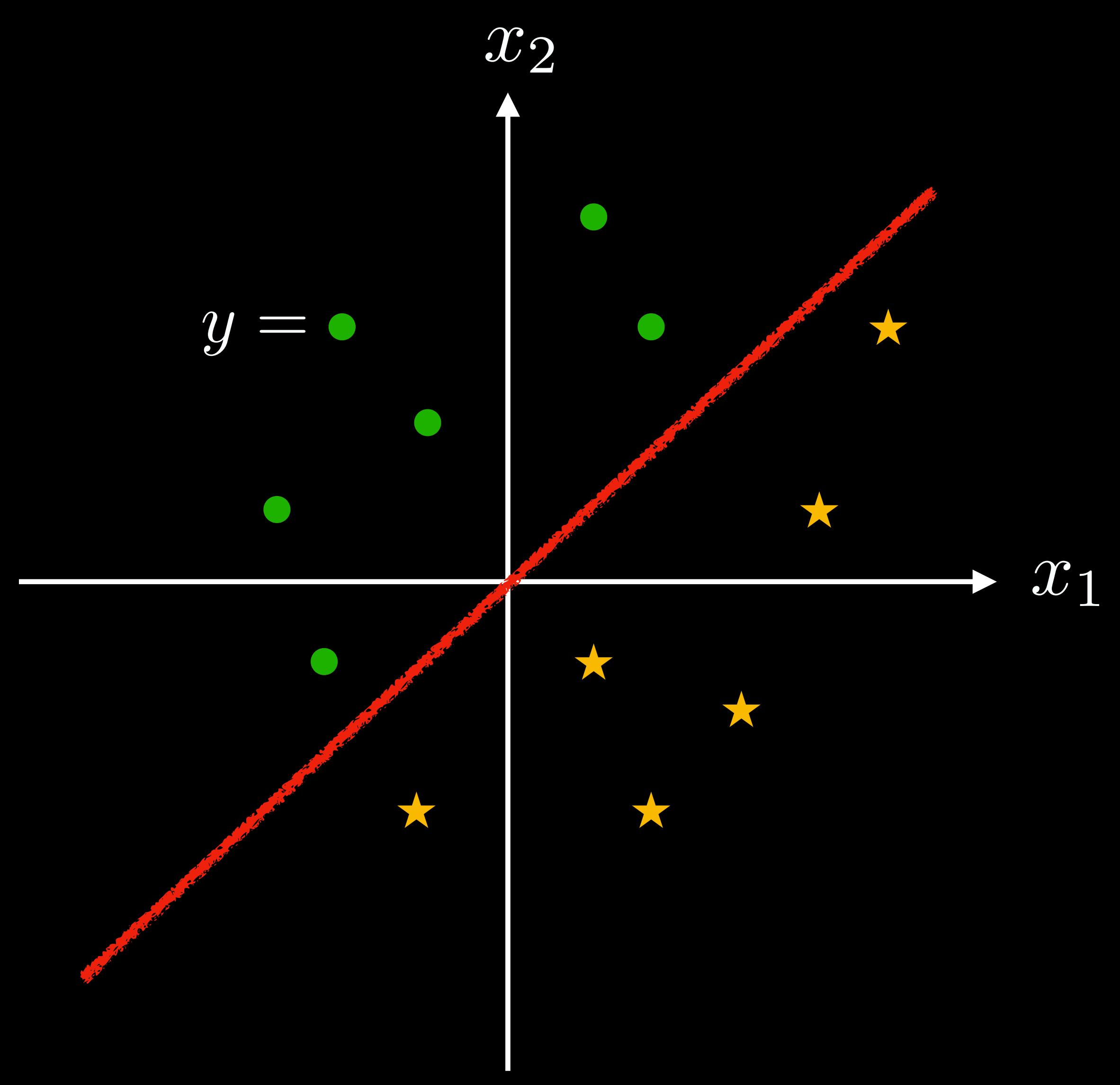
Deep Learning

Supervised Learning

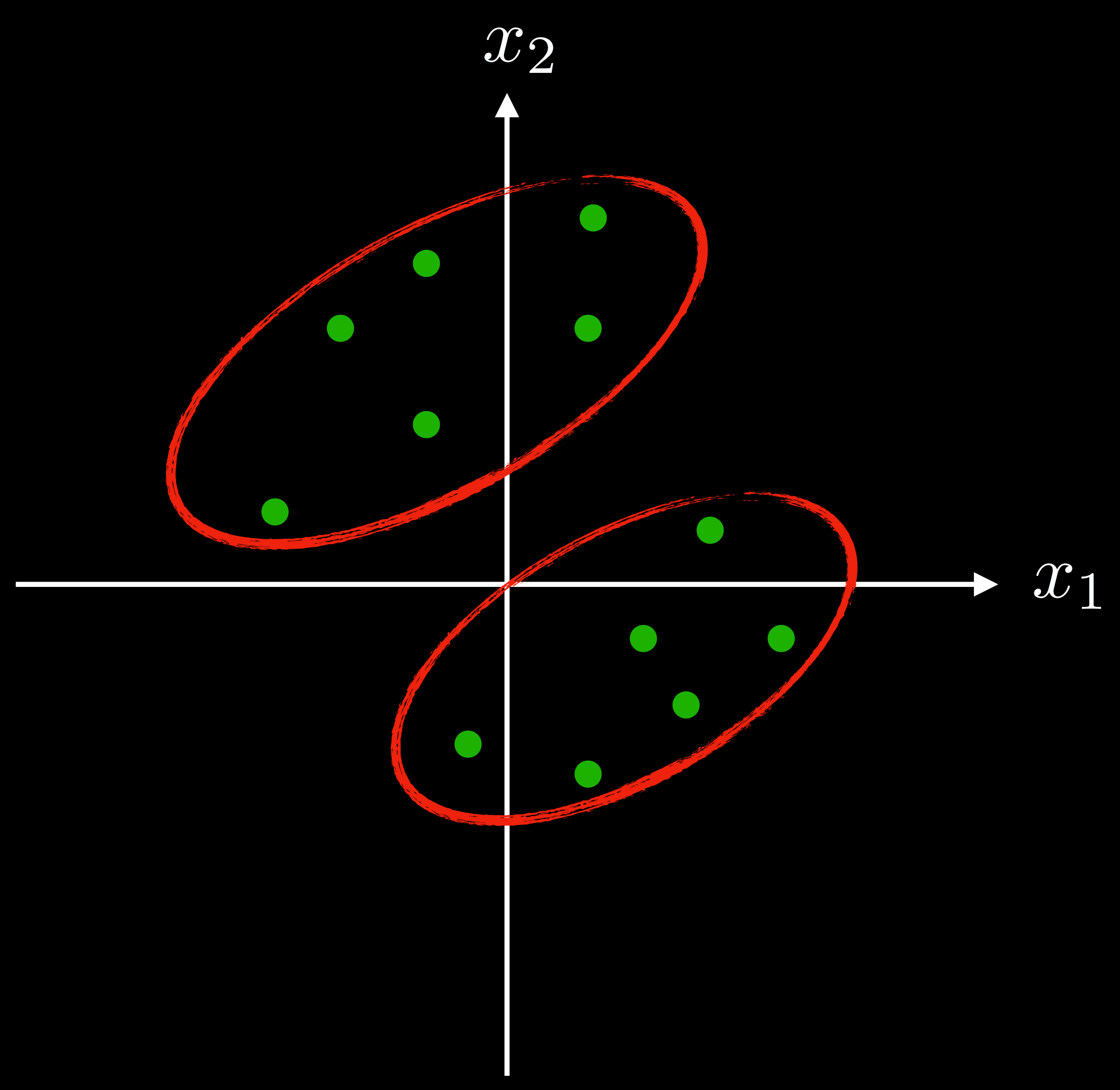
Unsupervised Learning

Given $(\mathbf{x}, \mathbf{y})$

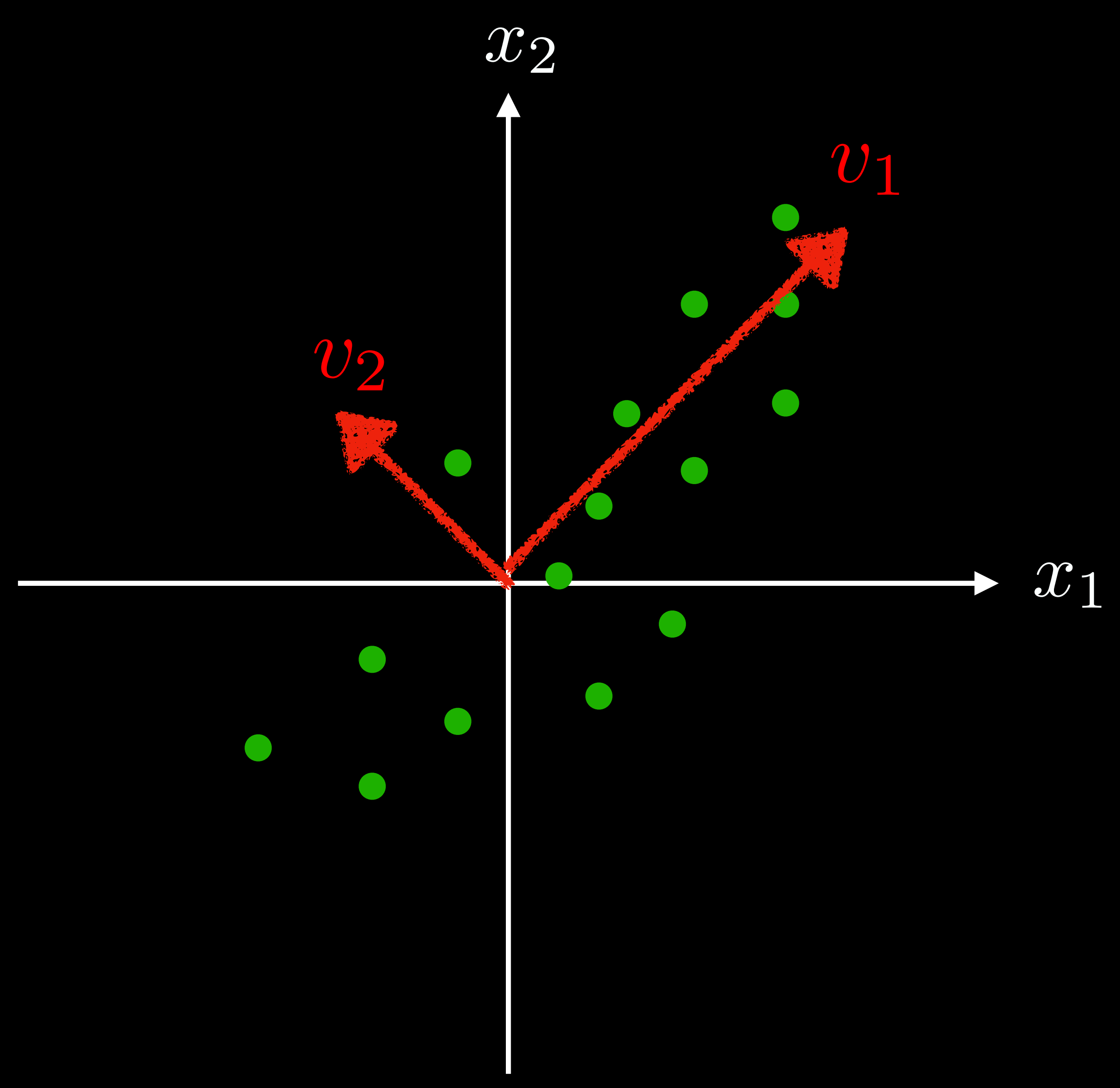


Regression

Given $\mathbf{x}$



Clustering

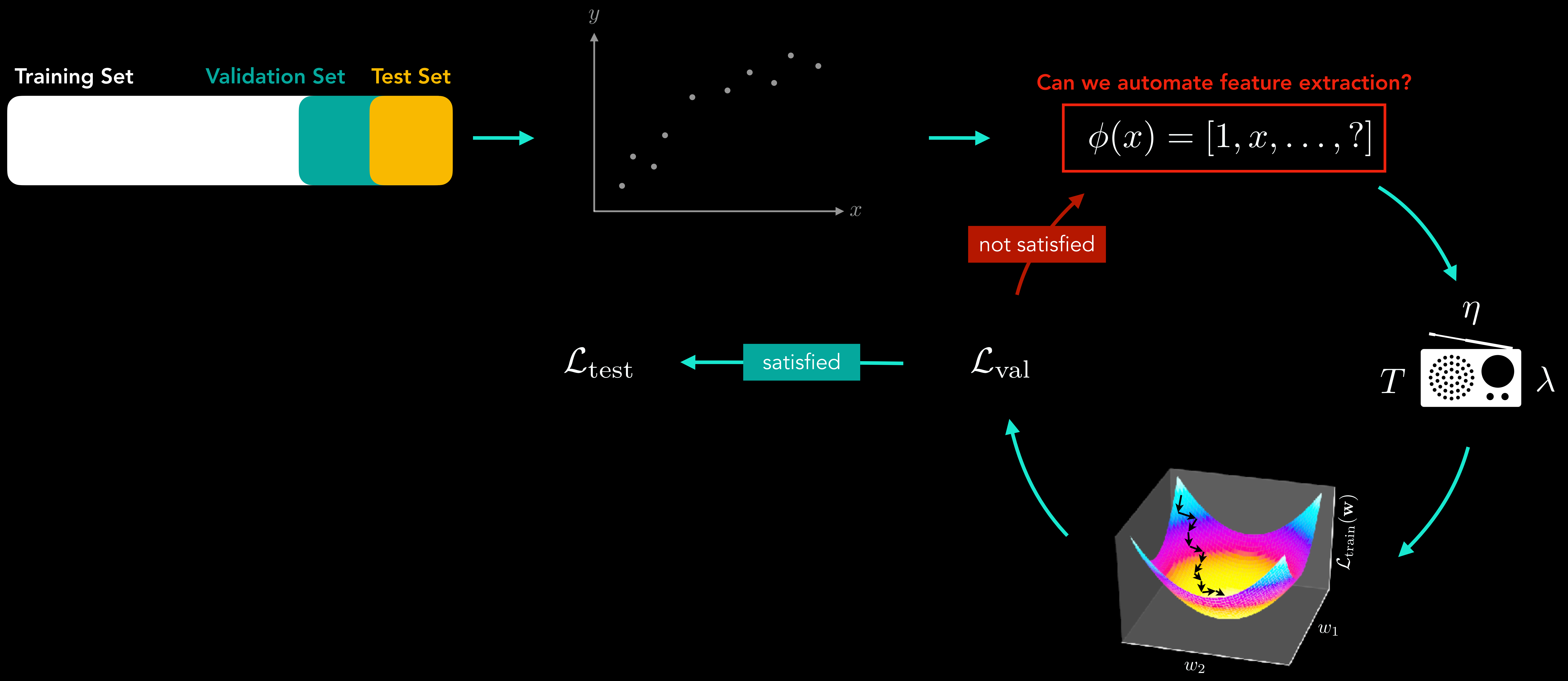


PCA

- Learn mapping given the labels
- Use to predict unseen labels

- Find patterns in high dimensional data
- Use for dimensionality reduction

The ML workflow



Linear Predictor

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

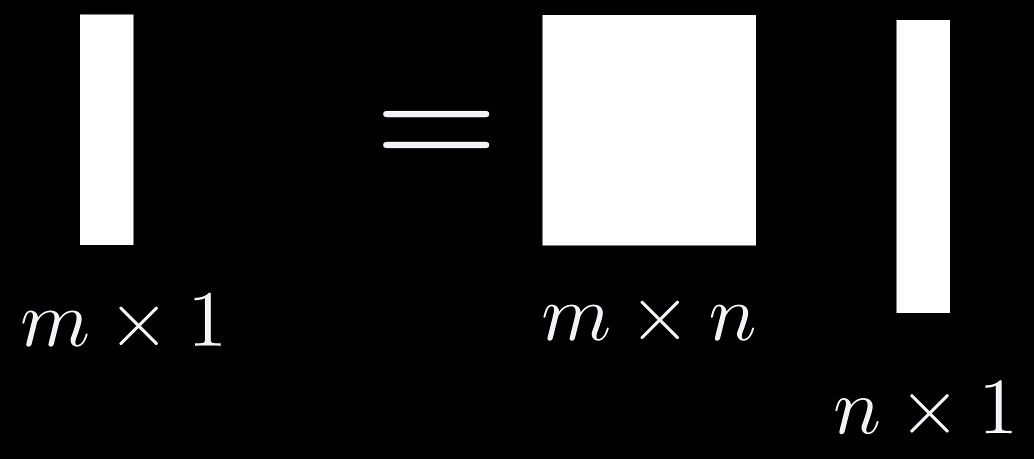
$$\phi(x) = [1,\ x]$$

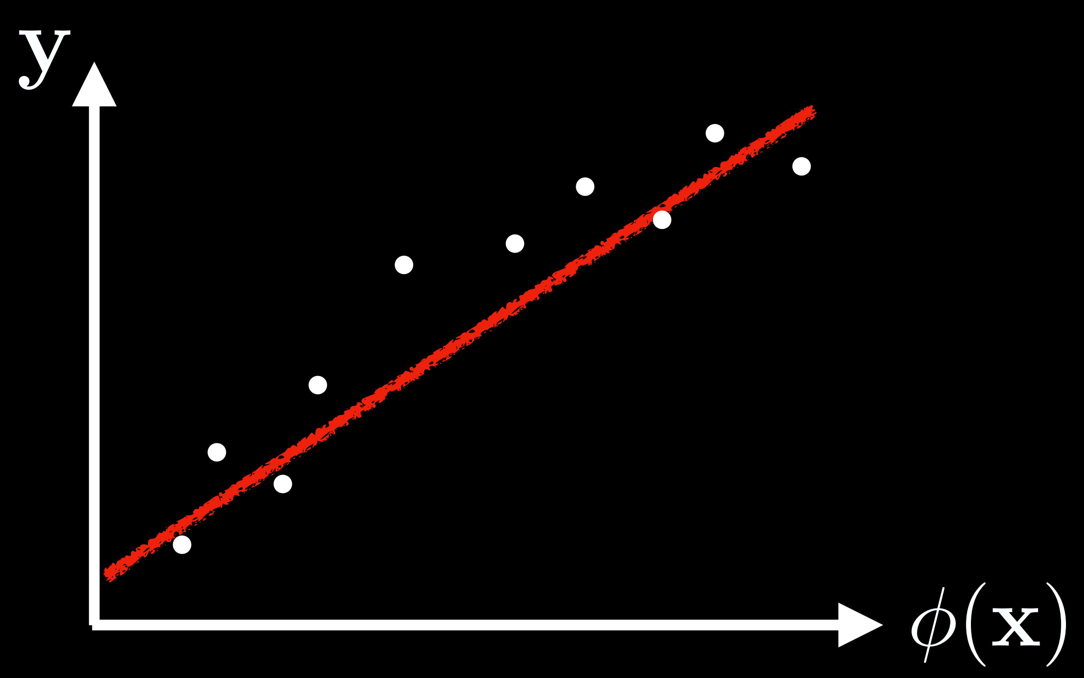
$$\phi(x) = [1, x, x^2, x^3]$$

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

Higher dimensions

$$\overset{\hat{\mathbf{y}}}{\boxed{f_{\mathbf{W}}(\mathbf{x})}} = \mathbf{W}\ \overset{\mathbf{x}}{\boxed{\phi(\mathbf{x})}}$$





Linear Predictor

Matrix multiplication

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

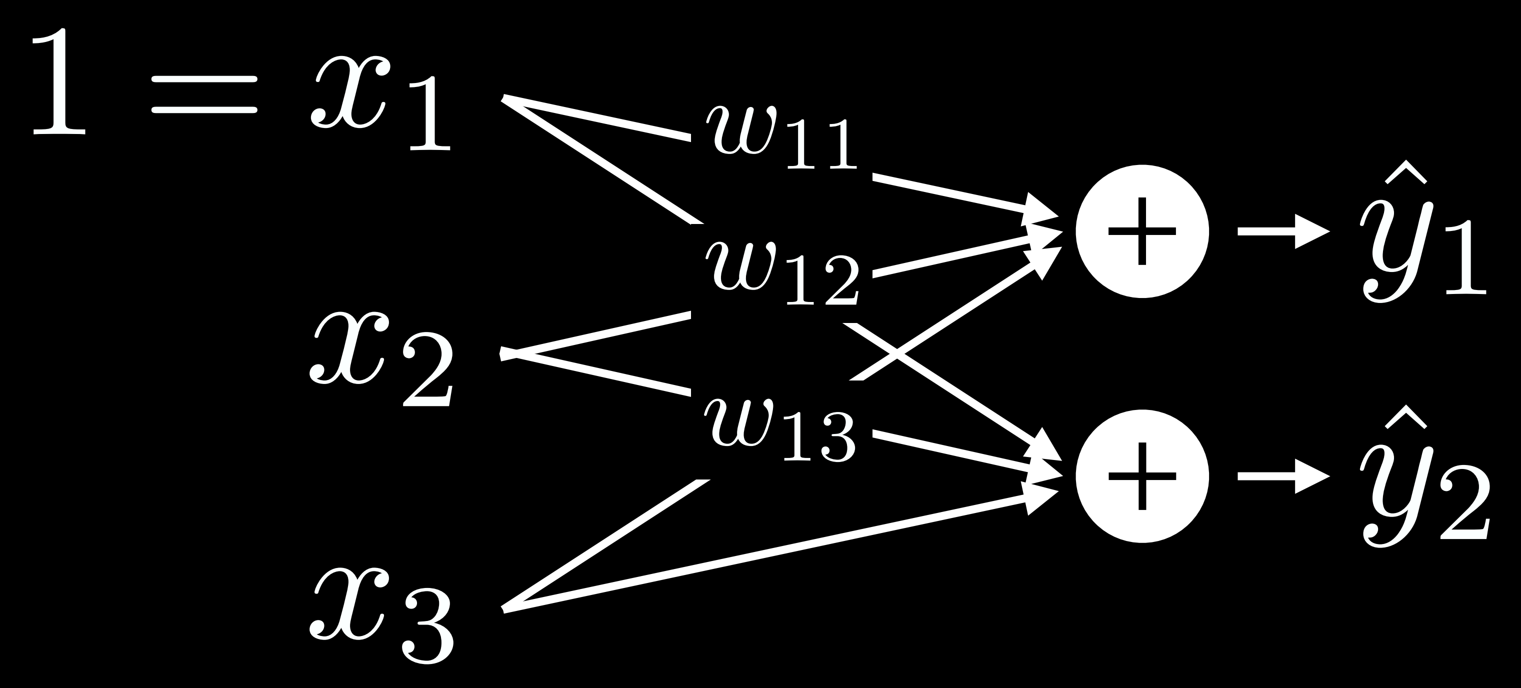
$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Index notation

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

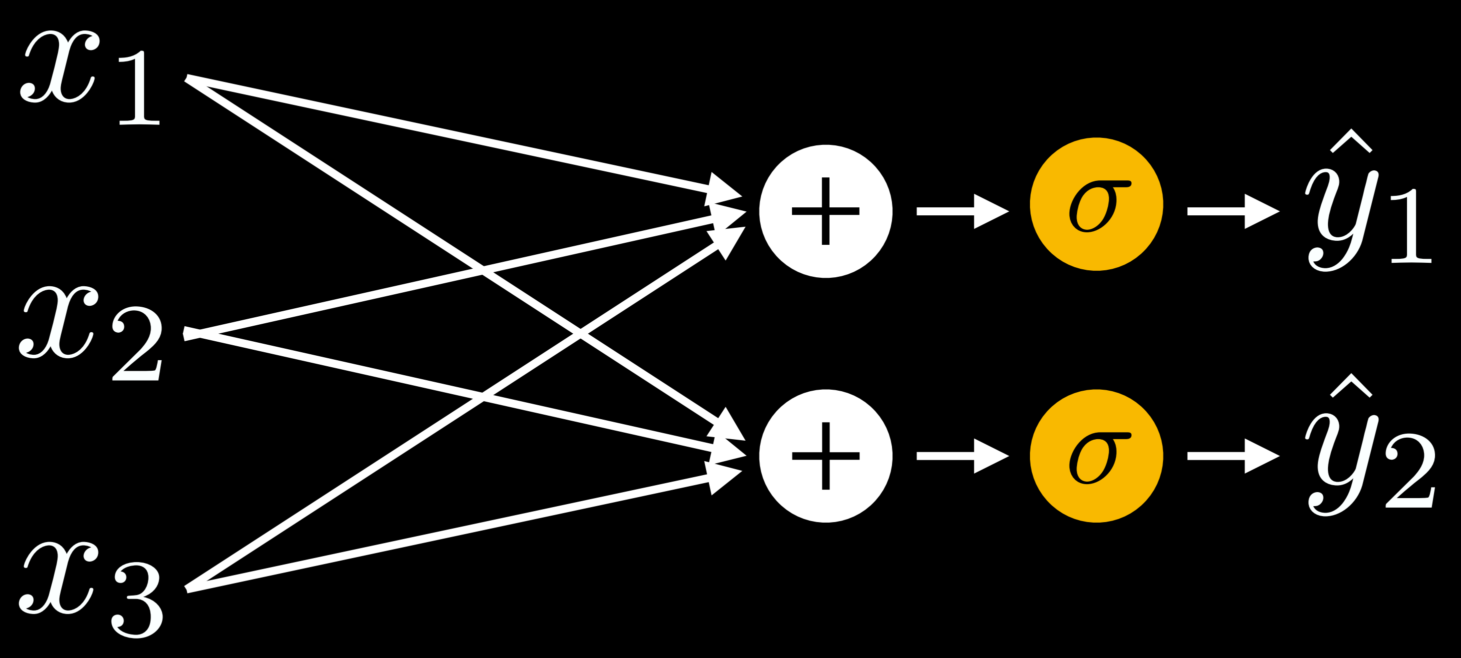
Define features such that
first component accounts for bias

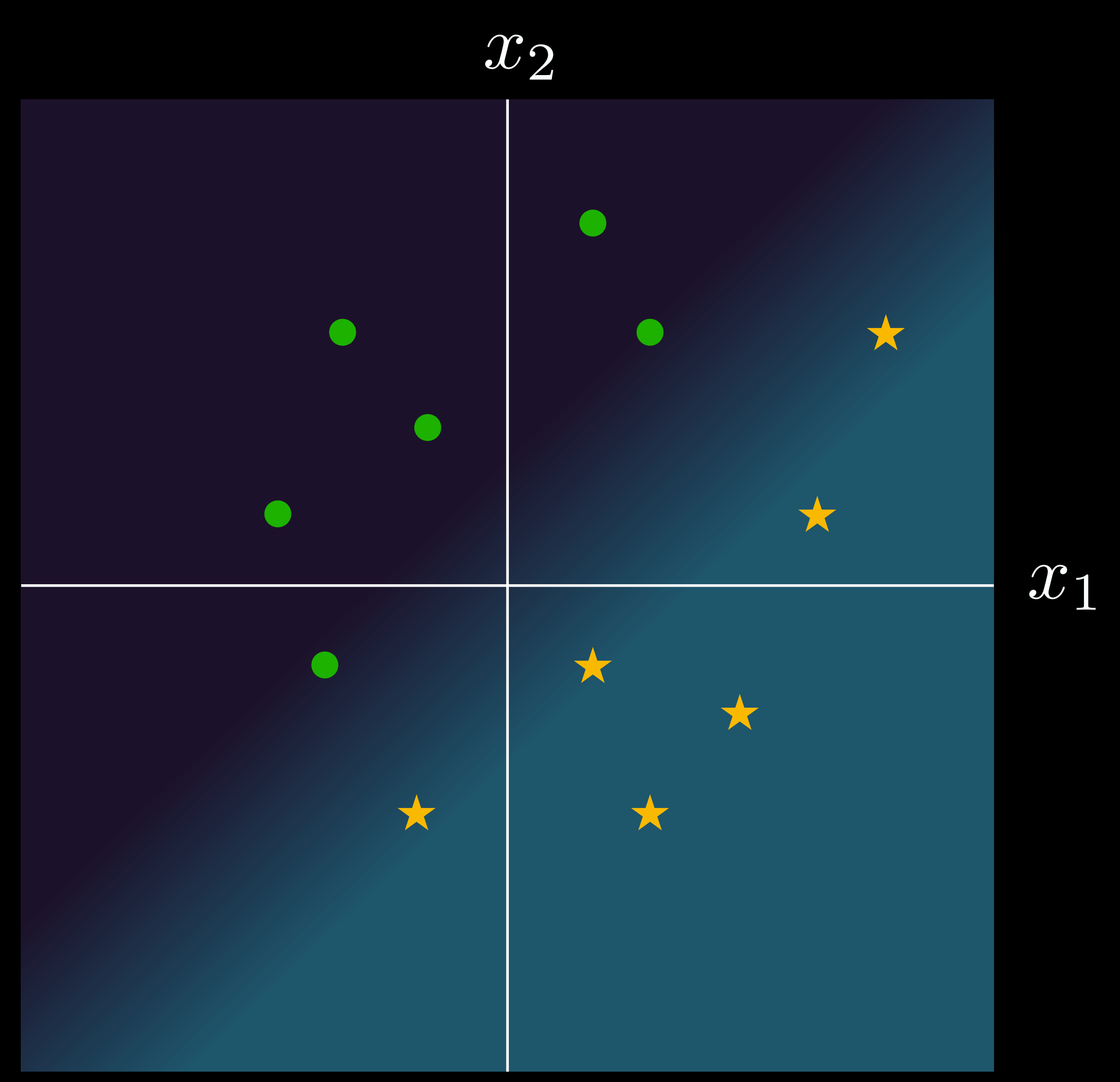
Network representation



Nonlinear Predictor

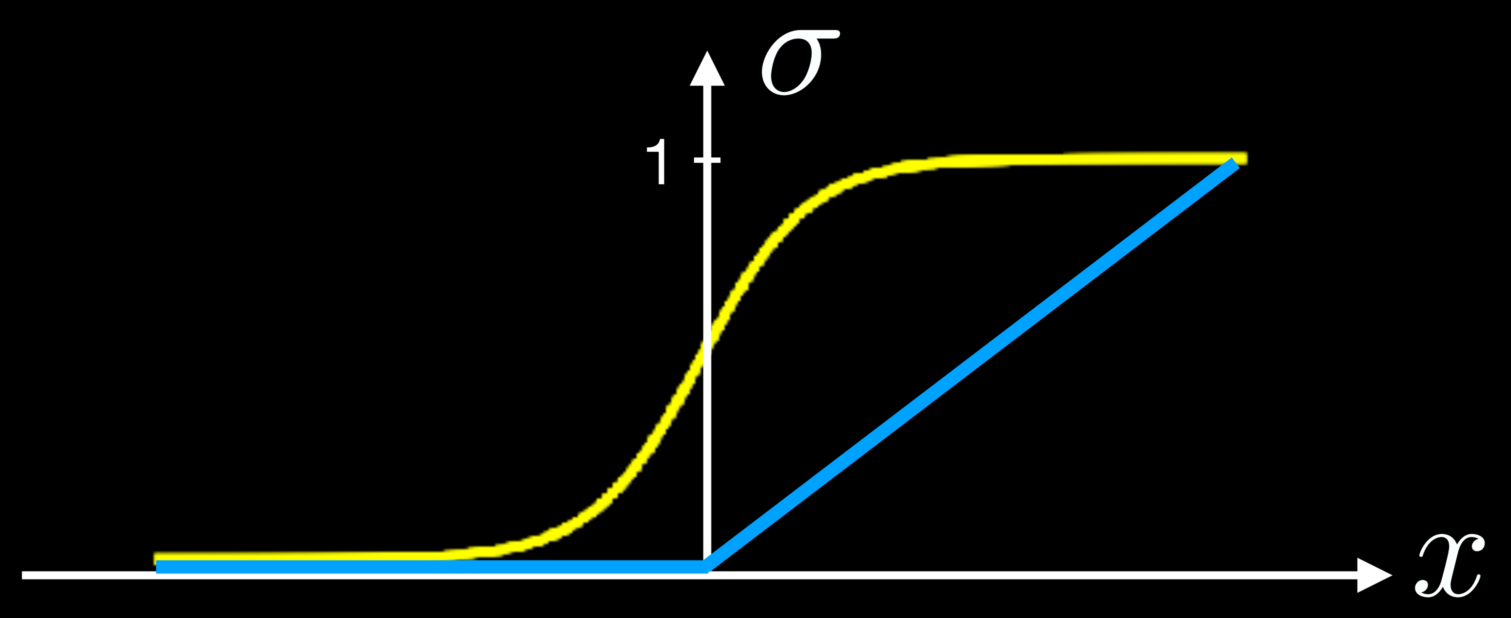
$$\hat{\mathbf{y}} = \sigma(\mathbf{W}\mathbf{x})$$





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

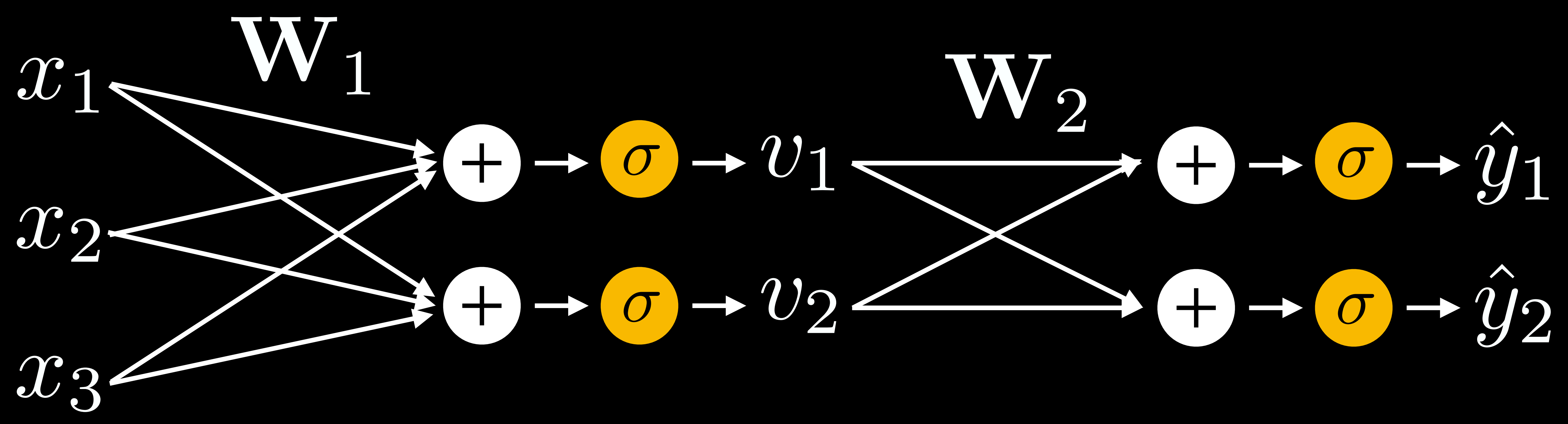
ReLU: $\sigma(x) = xH(x)$



Activation Function

Neural network

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



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

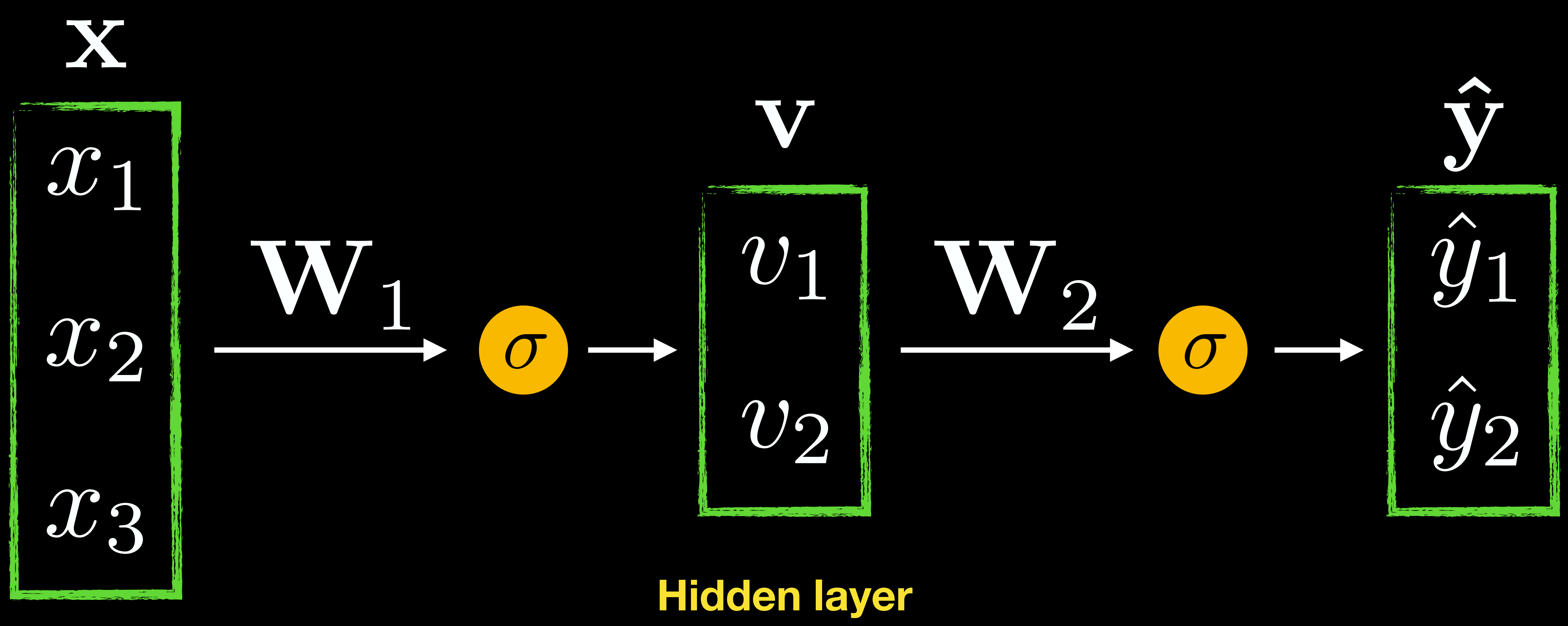
Nonlinear Linear Nonlinear Linear

Neural network

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

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

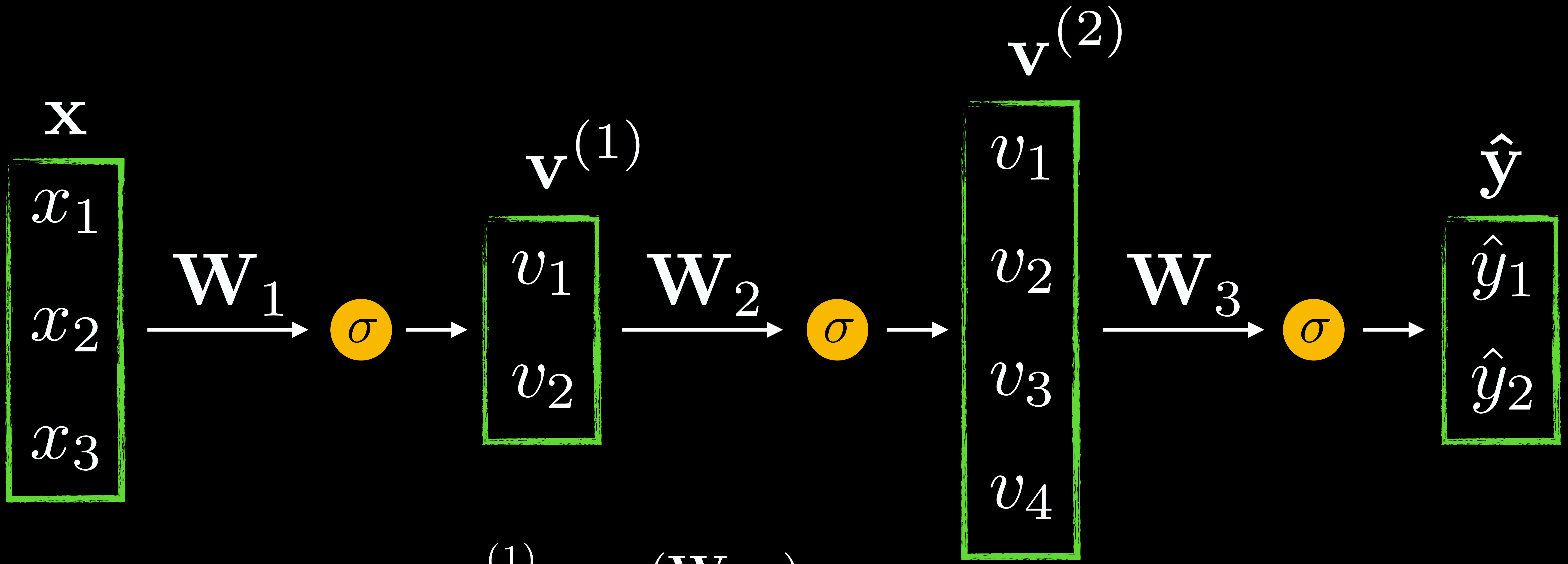
$$\hat{\mathbf{y}} = \sigma(\mathbf{W}_2 \mathbf{v})$$



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

Deep network

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

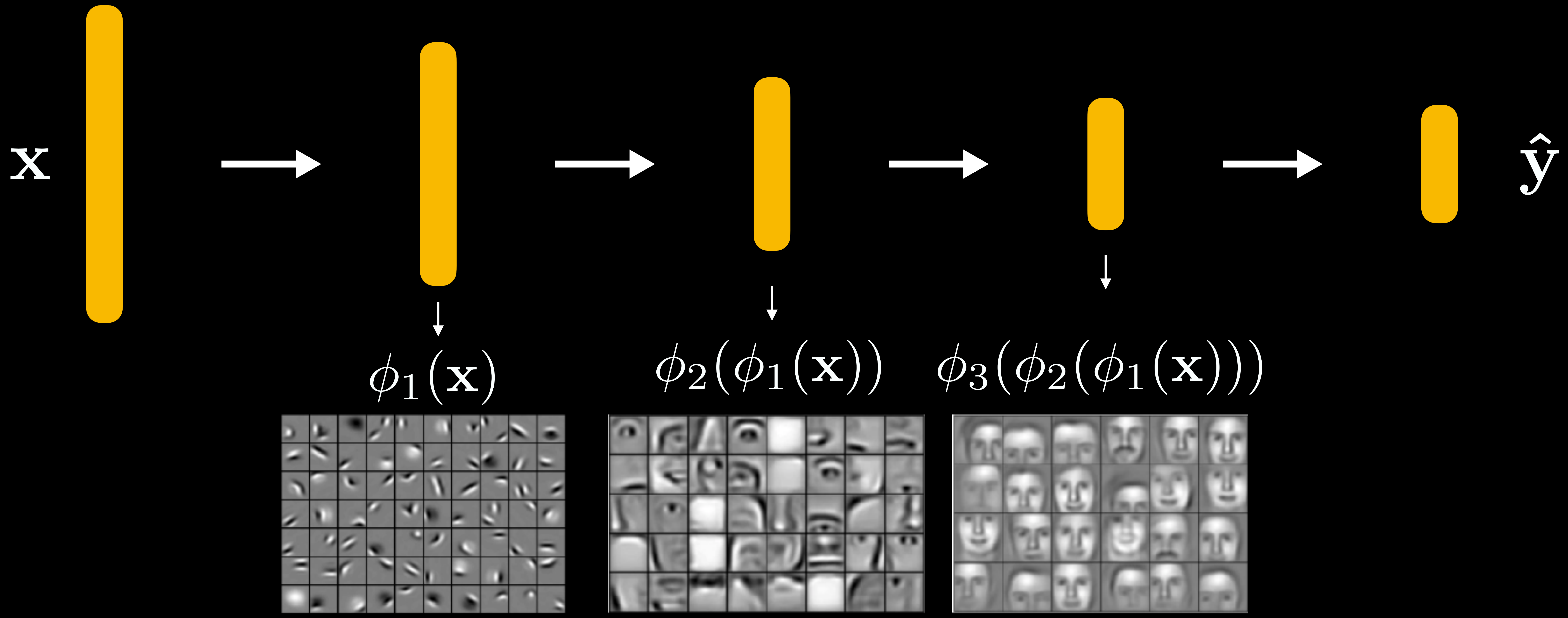


$$\mathbf{v}^{(1)} = \sigma\left(\mathbf{W}_1 \mathbf{x}\right)$$

$$\mathbf{v}^{(2)} = \sigma\left(\mathbf{W}_2 \mathbf{v}^{(1)}\right)$$

$$\hat{\mathbf{y}} = \sigma\left(\mathbf{W}_3 \mathbf{v}^{(2)}\right)$$

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

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

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

Stochastic gradient descent update

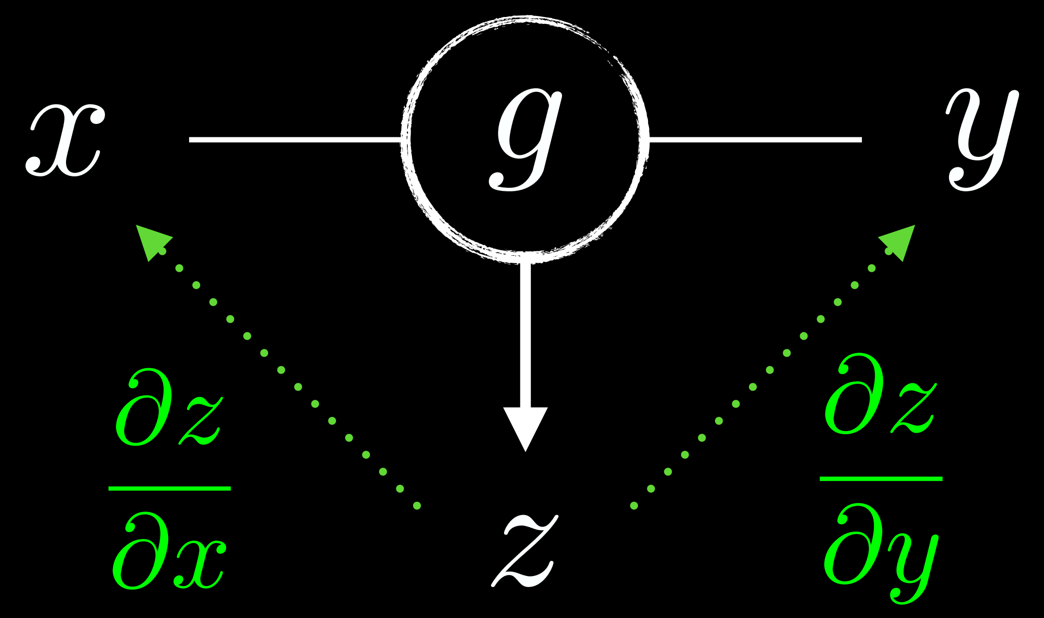
$$\mathbf{W}_1 \leftarrow \mathbf{W}_1 - \eta \nabla_{\mathbf{W}_1} \mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{W}_1, \mathbf{W}_2)$$

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

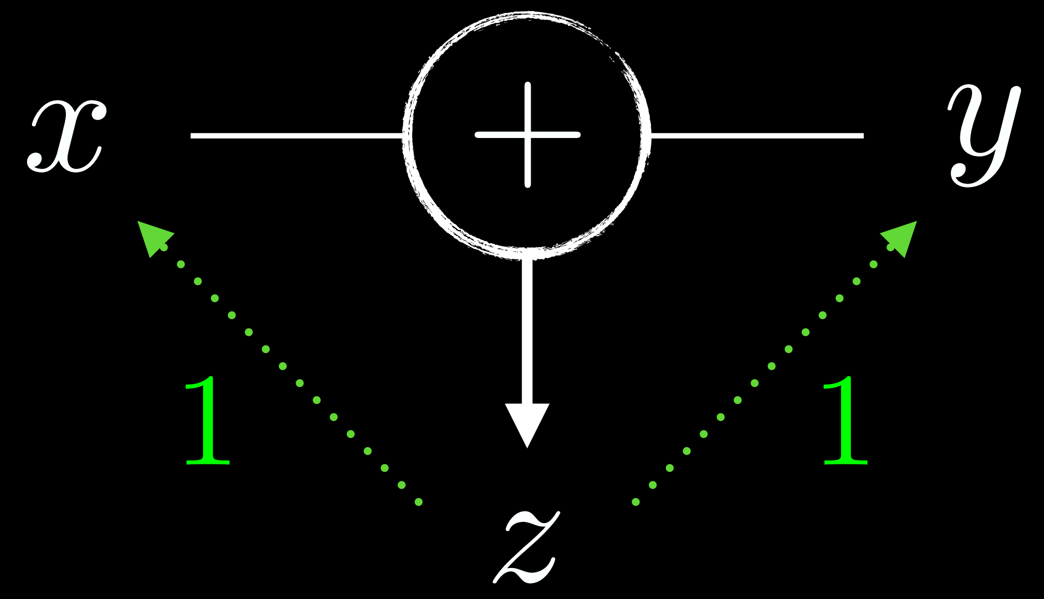
How do we calculate the gradients?

Computation graphs

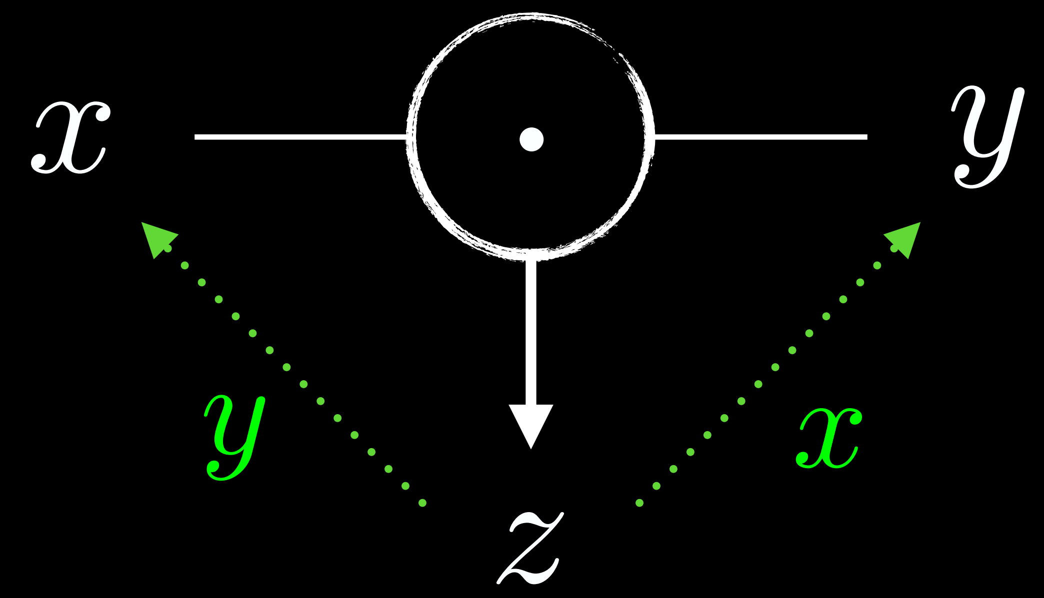
$$z = g(x, y)$$



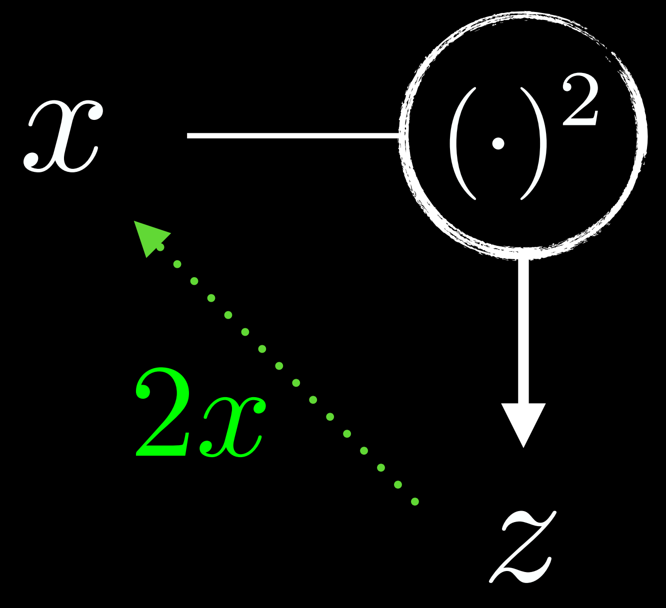
$$z = x + y$$



$$z = xy$$

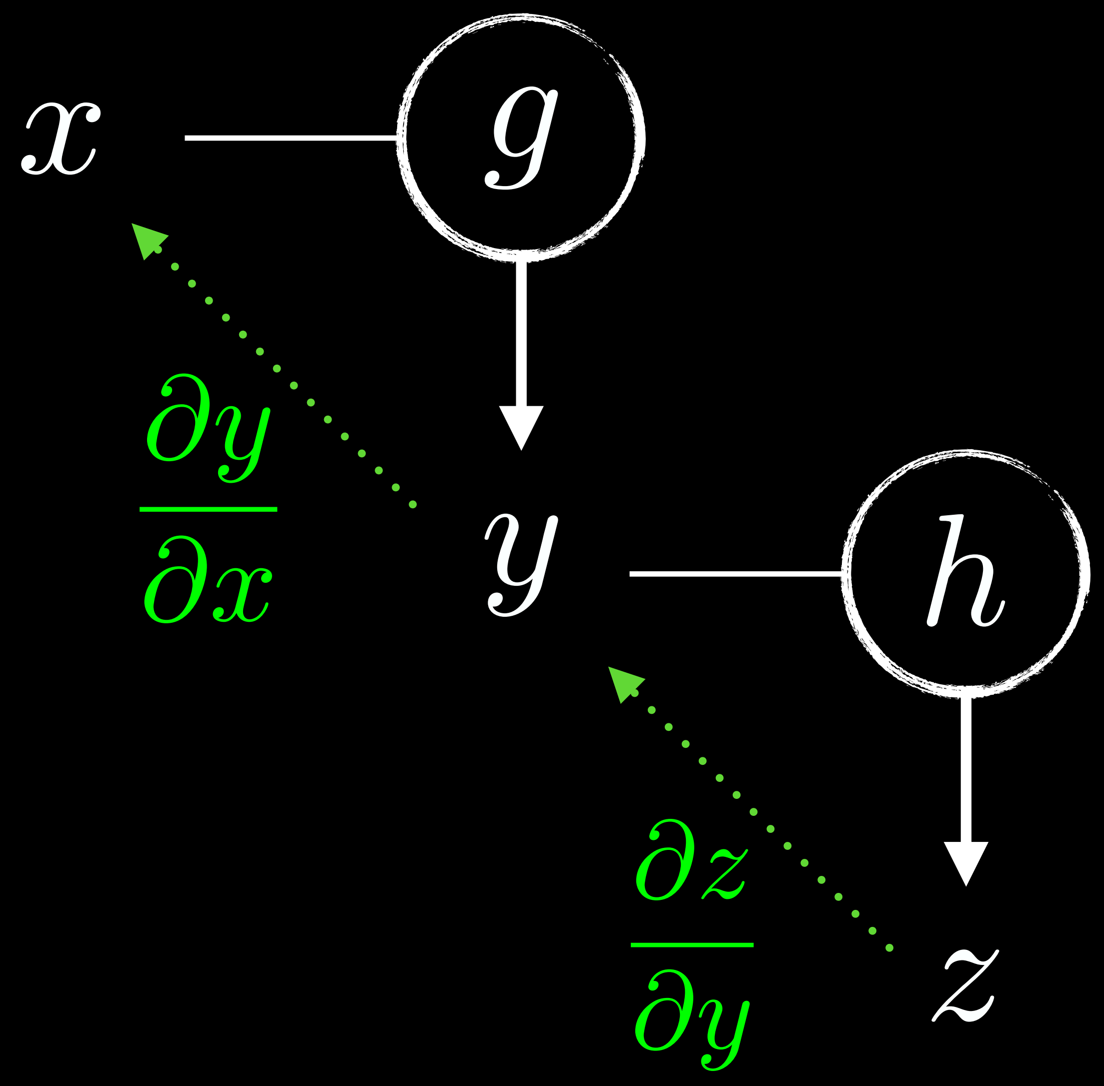


$$z = x^2$$



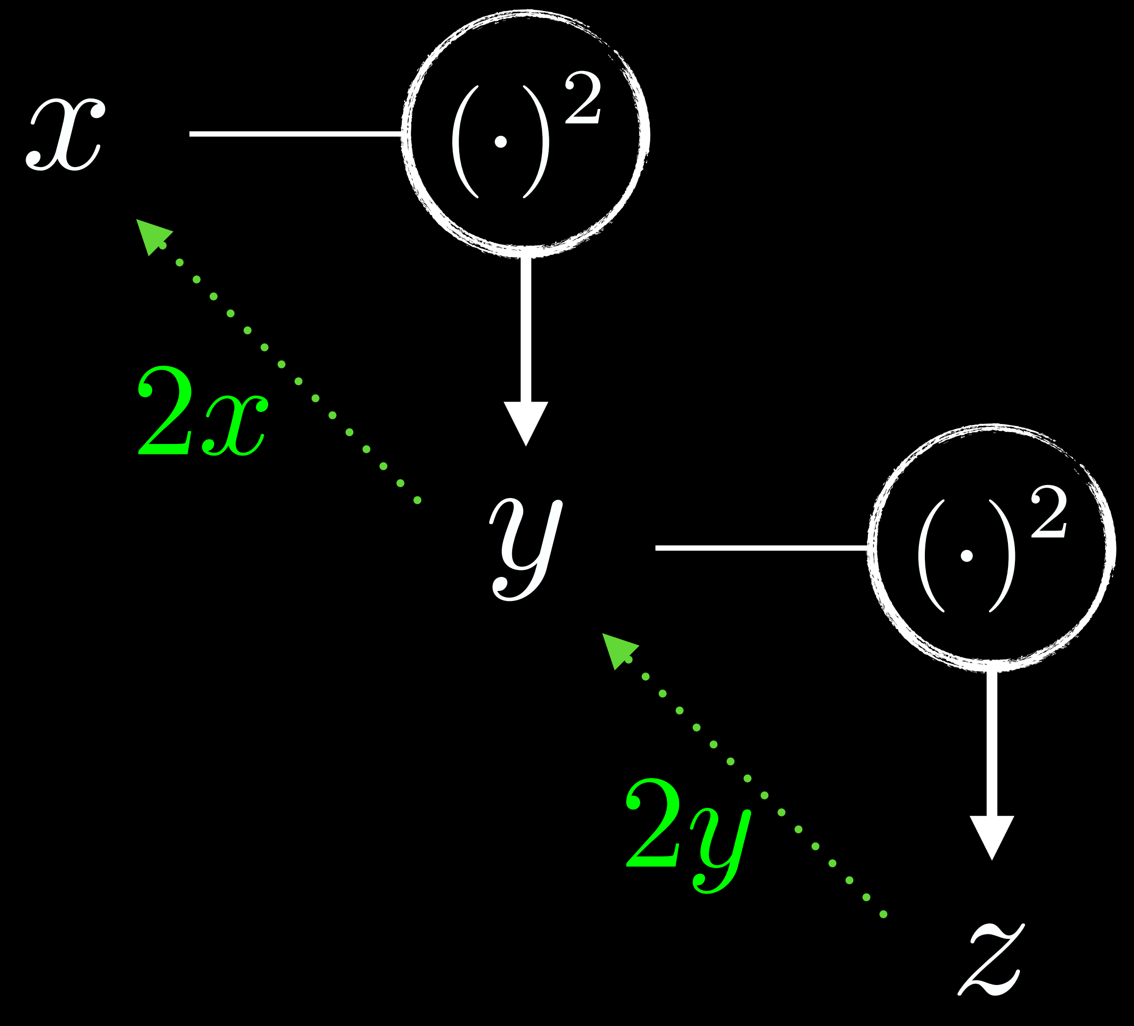
Chain rule

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



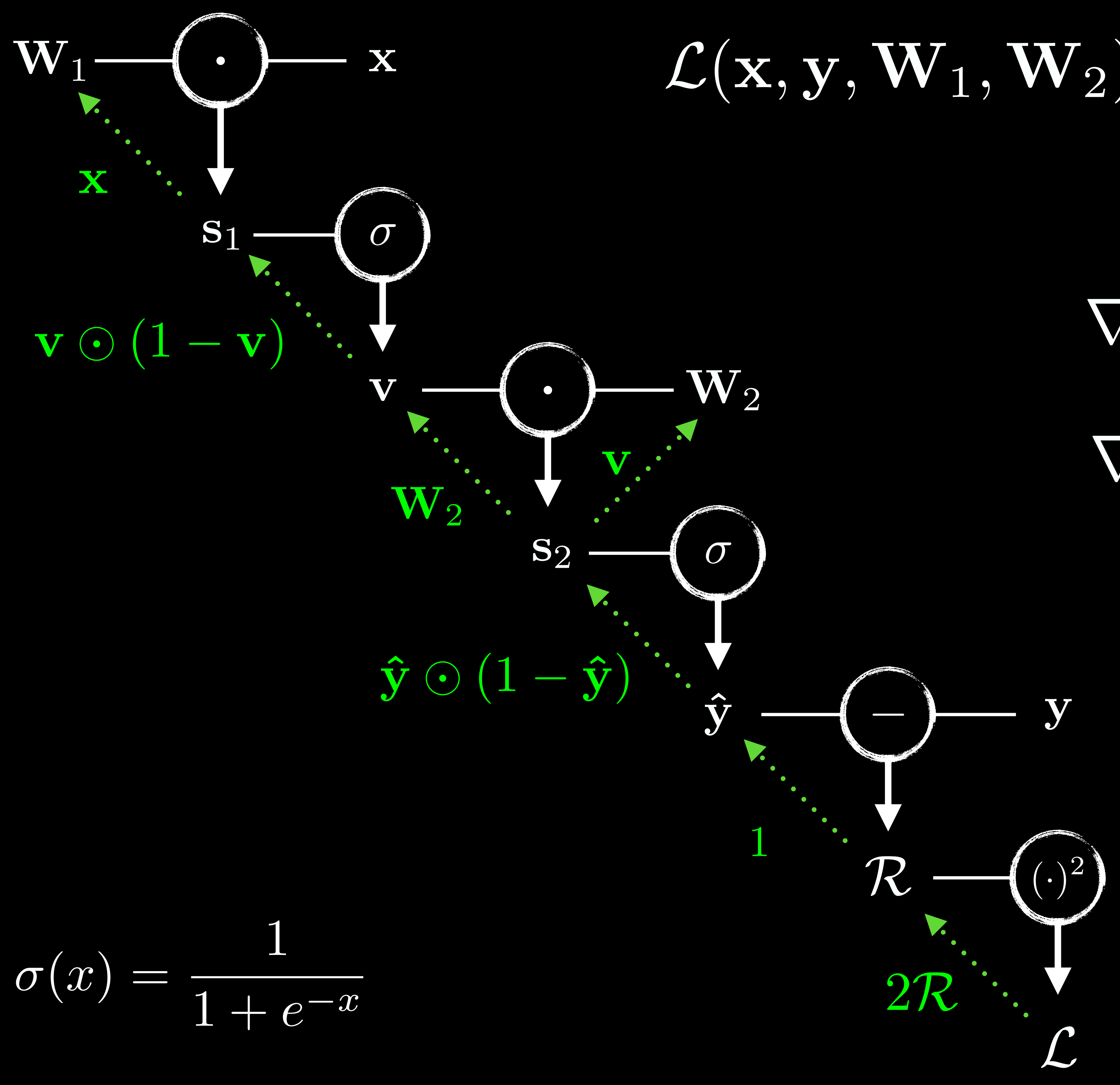
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

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



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

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

Optimization

Training loss

$$\mathcal{L}_{\text{train}}(\mathbf{W}_1, \mathbf{W}_2) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}_{\text{train}}} \mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{W}_1, \mathbf{W}_2)$$

Objective

$$\hat{\mathbf{W}}_1, \hat{\mathbf{W}}_2 = \underset{\mathbf{W}_1, \mathbf{W}_2}{\operatorname{argmin}} \; \mathcal{L}_{\text{train}}(\mathbf{W}_1, \mathbf{W}_2)$$

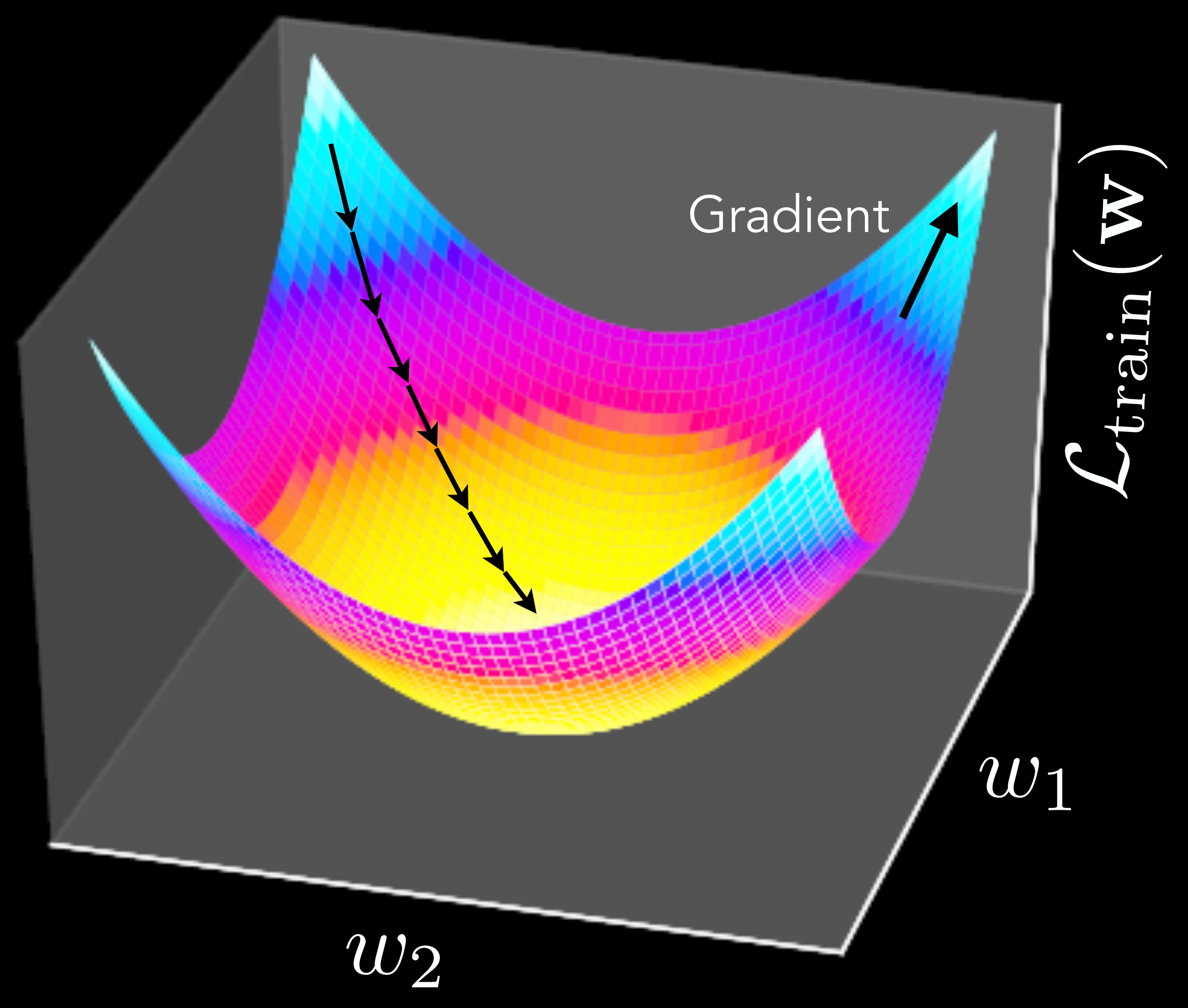
Optimal predictor

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

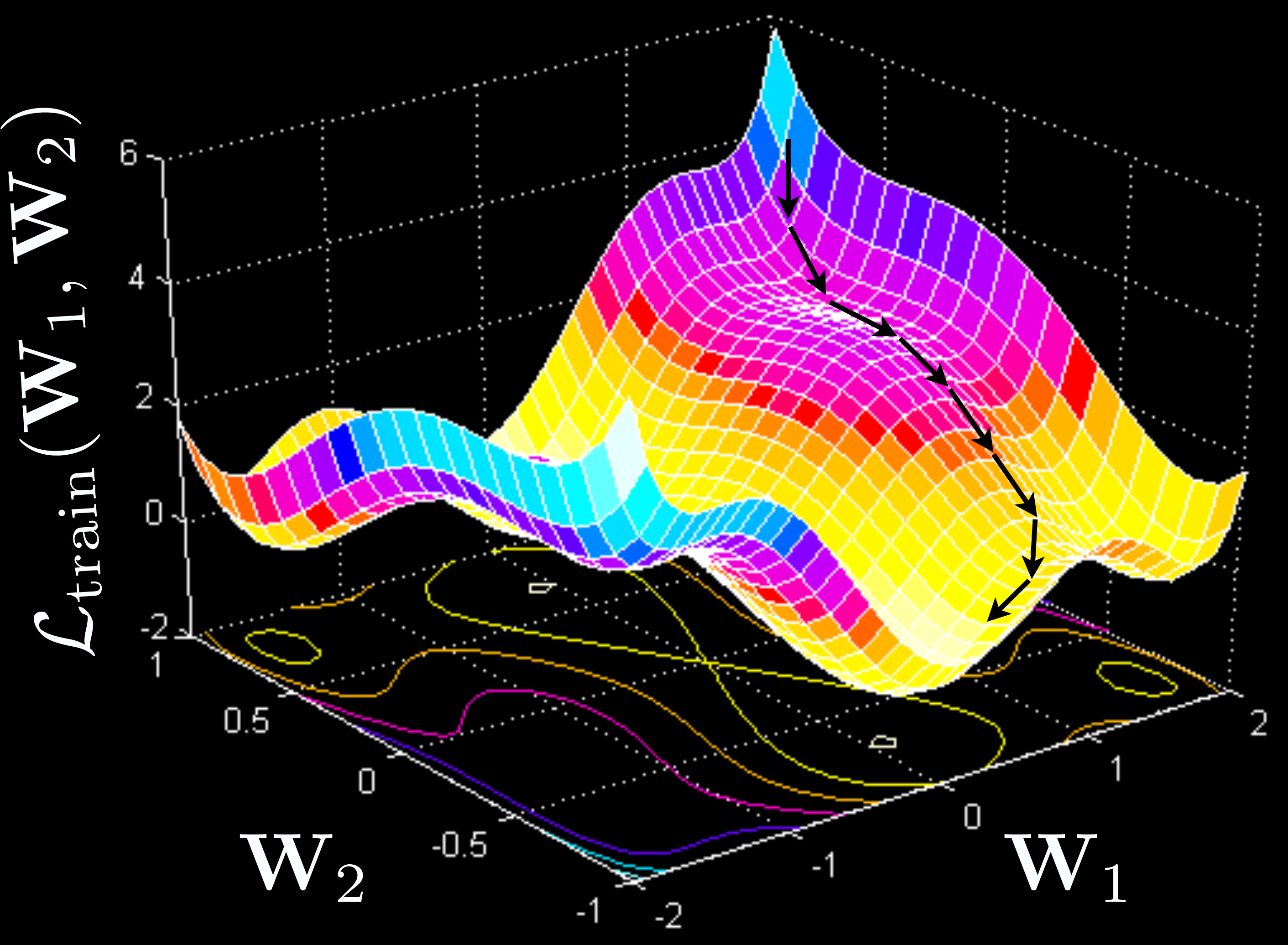
Non-convexity

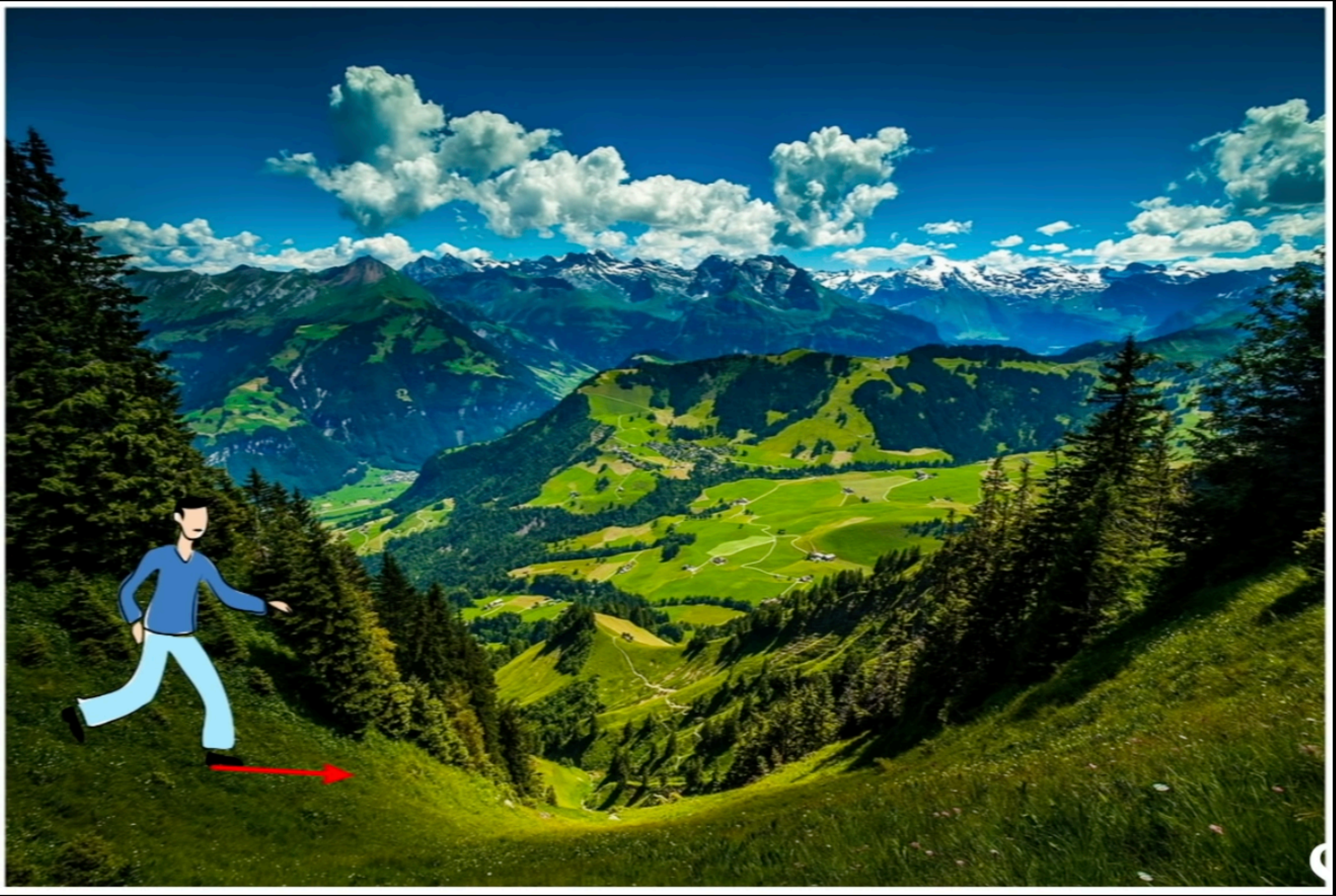
$$\hat{\mathbf{W}}_1, \hat{\mathbf{W}}_2 = \underset{\mathbf{W}_1, \mathbf{W}_2}{\text{argmin}} \ \mathcal{L}_{\text{train}}(\mathbf{W}_1, \mathbf{W}_2)$$

Linear predictor loss



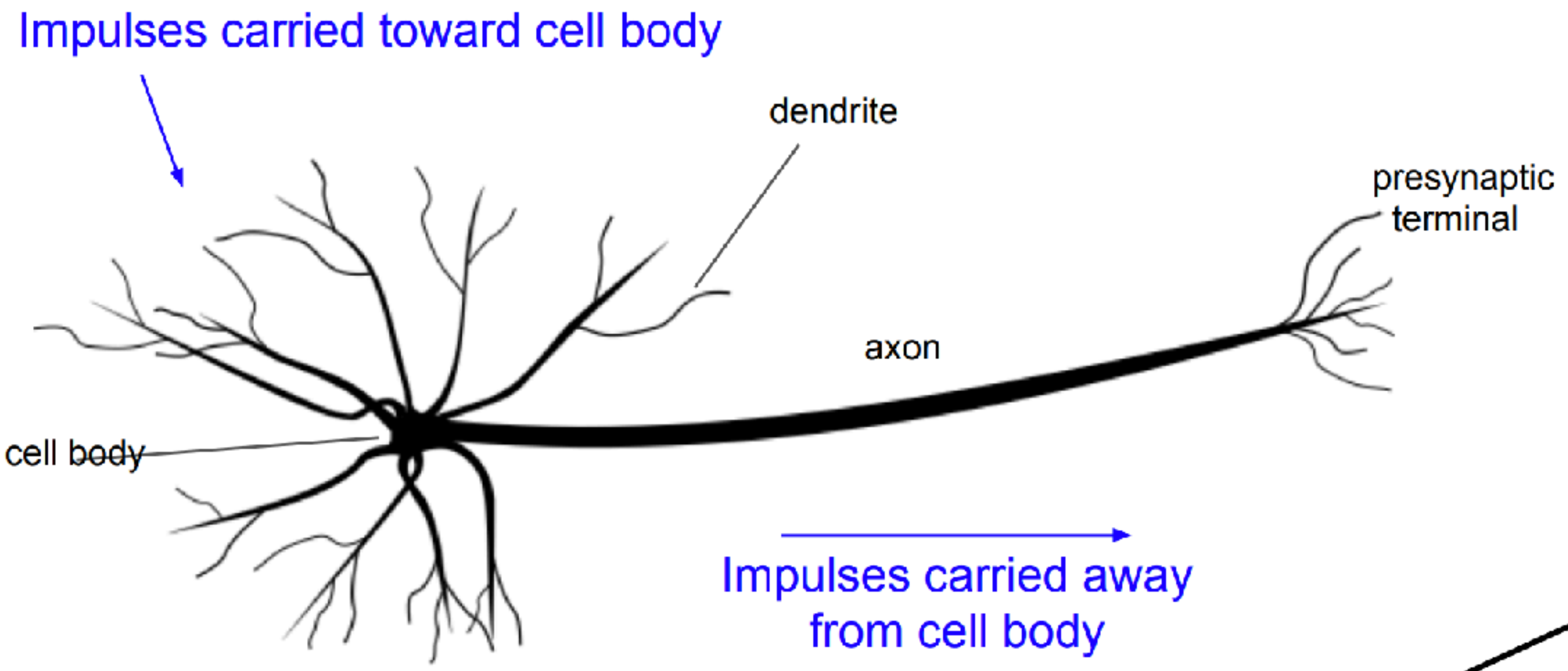
Neural network loss

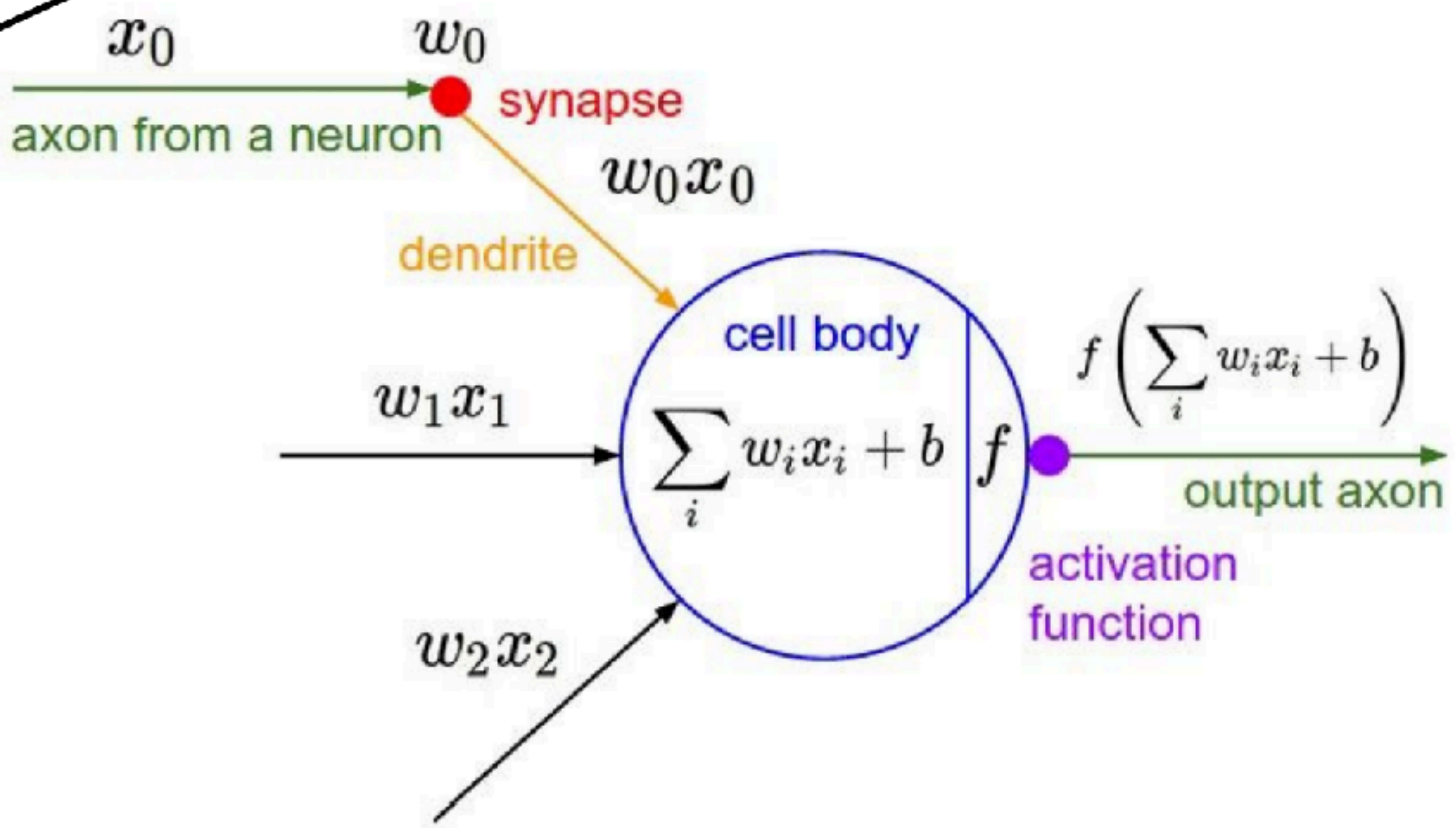




Full implementation of training a 2-layer Neural Network needs ~20 lines:

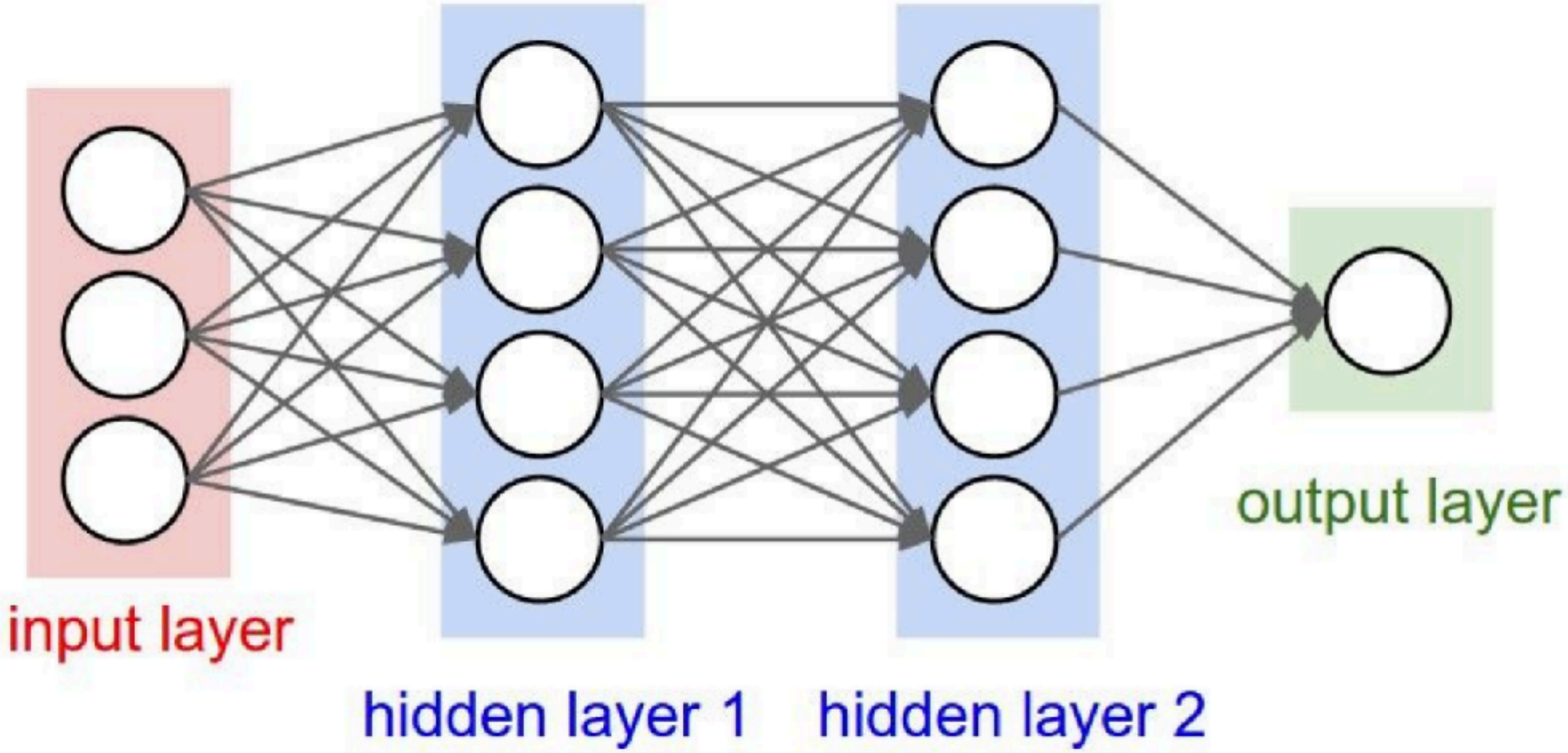
```python
import numpy as np
from numpy.random import randn

N, D_in, H, D_out = 64, 1000, 100, 10
x, y = randn(N, D_in), randn(N, D_out)
w1, w2 = randn(D_in, H), randn(H, D_out)

for t in range(2000):
  h = 1 / (1 + np.exp(-x.dot(w1)))
  y_pred = h.dot(w2)
  loss = np.square(y_pred - y).sum()
  print(t, loss)

  grad_y_pred = 2.0 * (y_pred - y)
  grad_w2 = h.T.dot(grad_y_pred)
  grad_h = grad_y_pred.dot(w2.T)
  grad_w1 = x.T.dot(grad_h * h * (1 - h))

  w1 -= 1e-4 * grad_w1
  w2 -= 1e-4 * grad_w2
```






```python
class Neuron:
  # ...
  def neuron_tick(inputs):
    """ assume inputs and weights are 1-D numpy arrays and bias is a number """
    cell_body_sum = np.sum(inputs * self.weights) + self.bias
    firing_rate = 1.0 / (1.0 + math.exp(-cell_body_sum)) # sigmoid activation func
    return firing_rate
```

Example feed-forward computation of a neural network

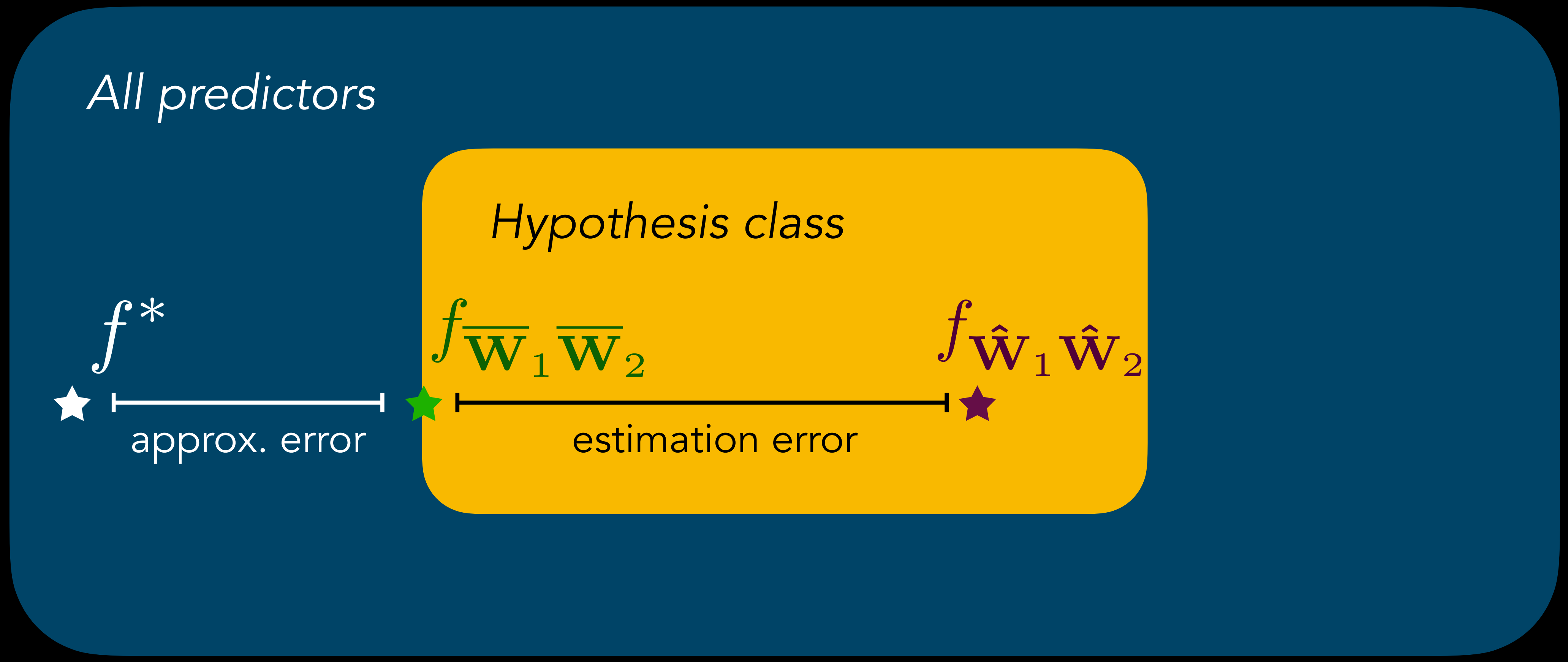



```
# forward-pass of a 3-layer neural network:
f = lambda x: 1.0/(1.0 + np.exp(-x)) # activation function (use sigmoid)
x = np.random.randn(3, 1) # random input vector of three numbers (3x1)
h1 = f(np.dot(W1, x) + b1) # calculate first hidden layer activations (4x1)
h2 = f(np.dot(W2, h1) + b2) # calculate second hidden layer activations (4x1)
out = np.dot(W3, h2) + b3 # output neuron (1x1)
```

Hypothesis Class

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

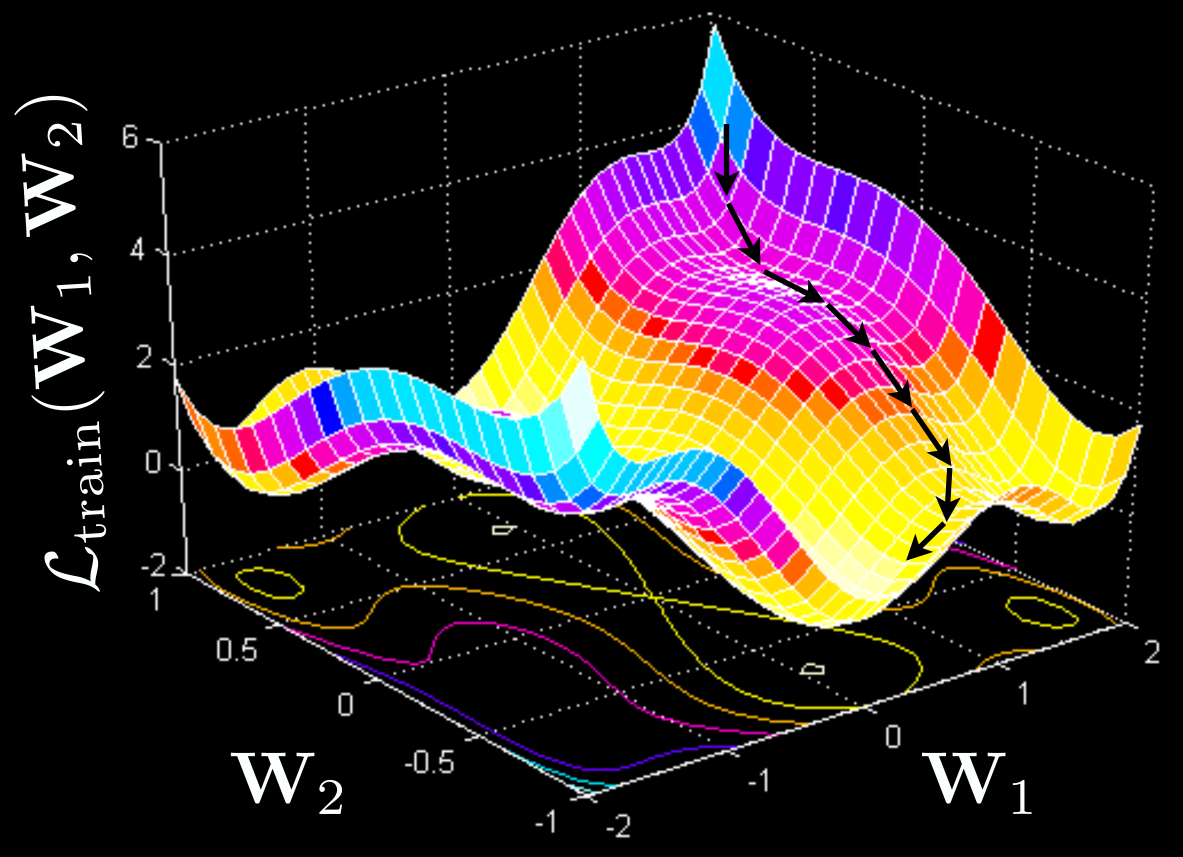
$$\mathcal{F} = \left\{f_{\mathbf{W}_1\mathbf{W}_2}(\mathbf{x}) \mid \mathbf{W}_1 \in \mathbb{R}^{k\times n}, \mathbf{W}_2 \in \mathbb{R}^{m\times k}\right\}$$



Hyperparameters

How do you train a deep network?

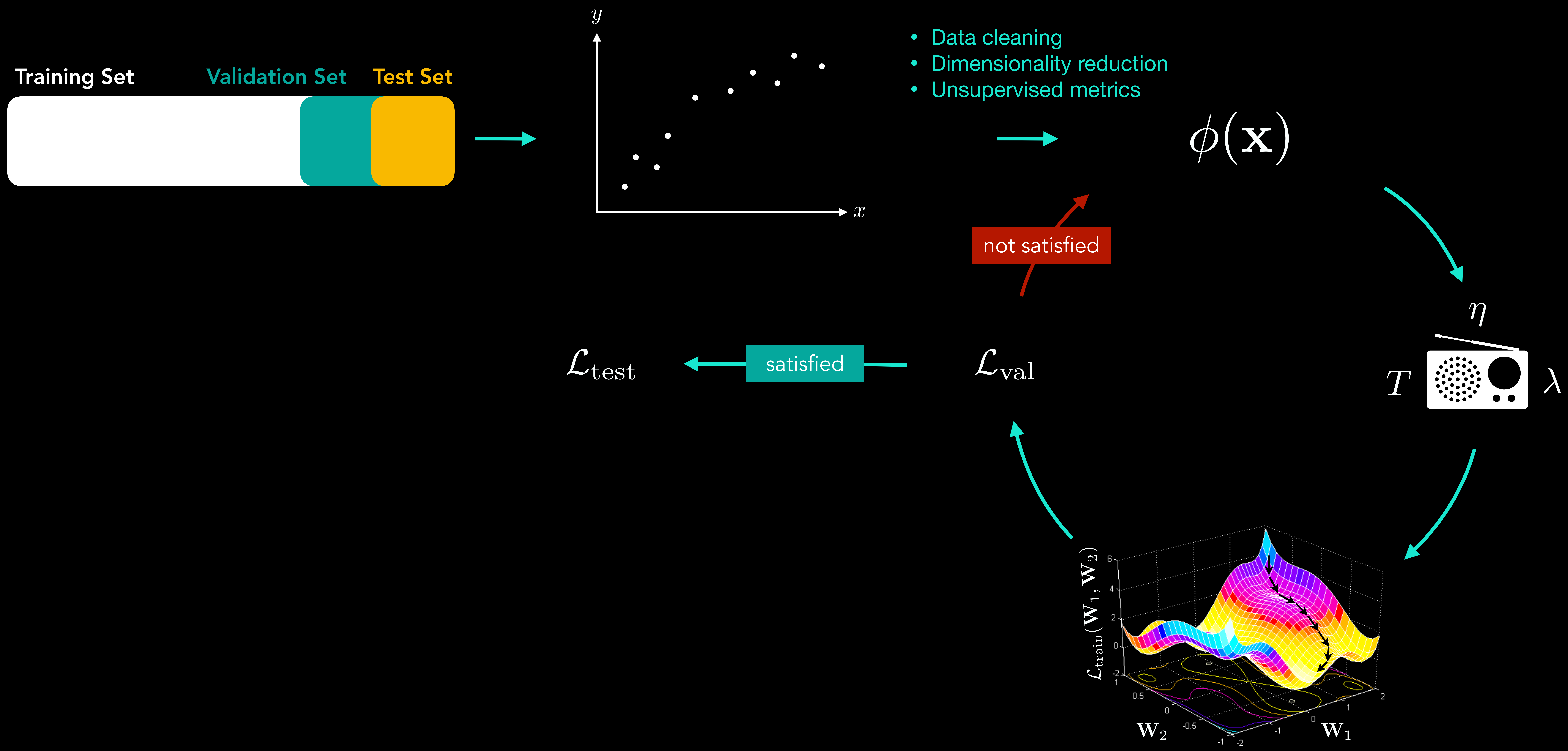
- Use many hidden layers for abstraction
- Use adaptive time steps
- Use hyper-parameter optimization



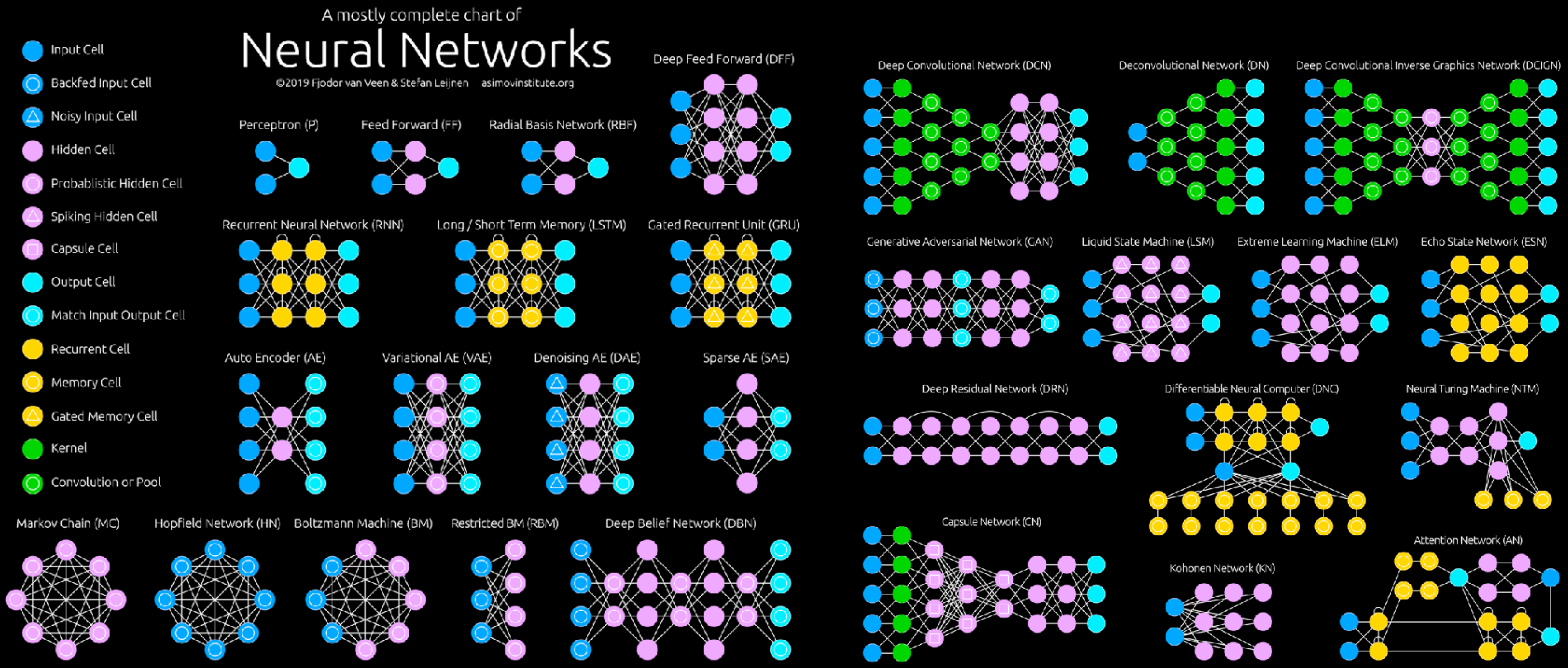
https://playground.tensorflow.org/

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

The ML workflow

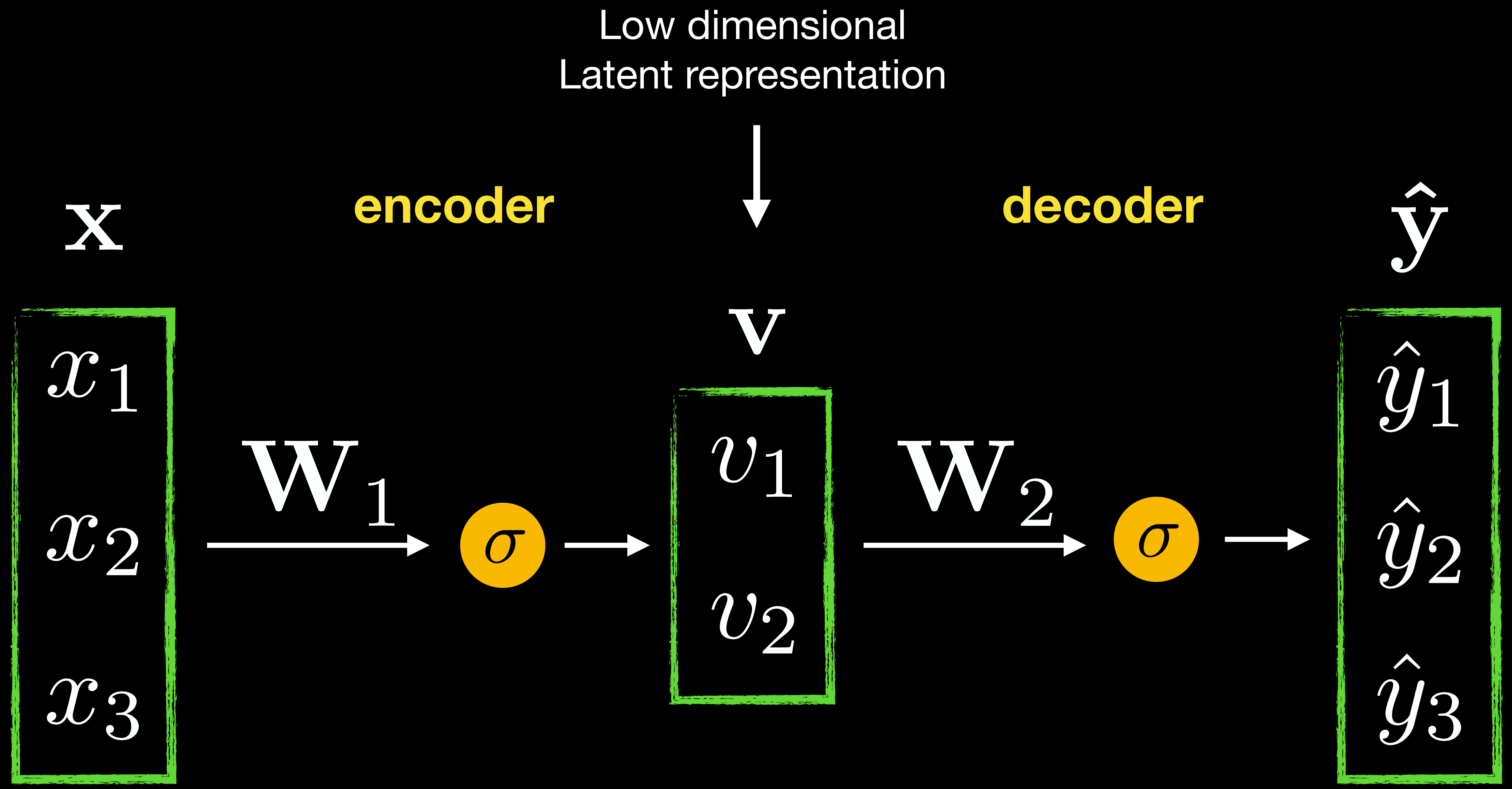


Neural network zoo



asimovinstitute.org

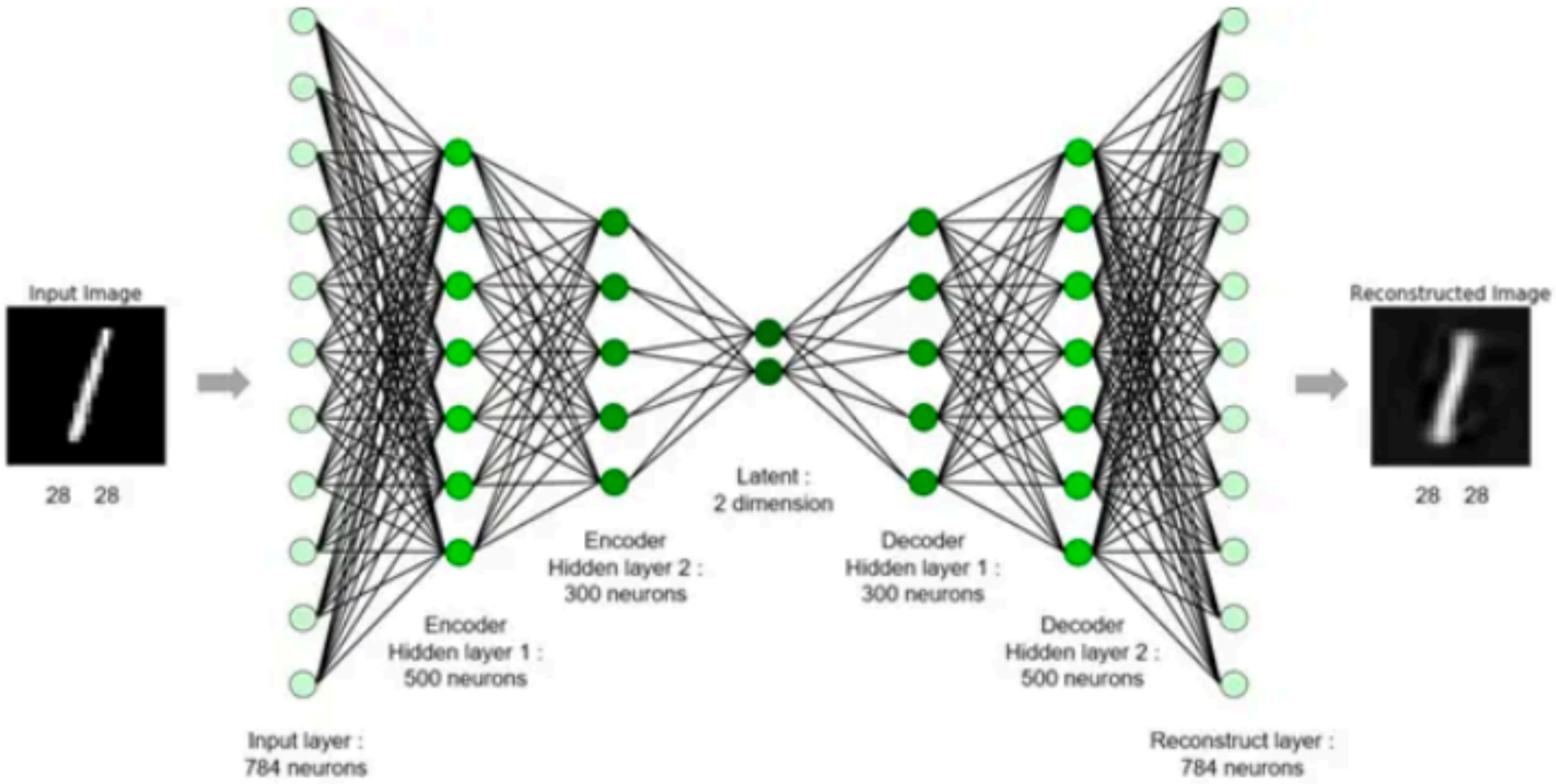
Auto-encoders

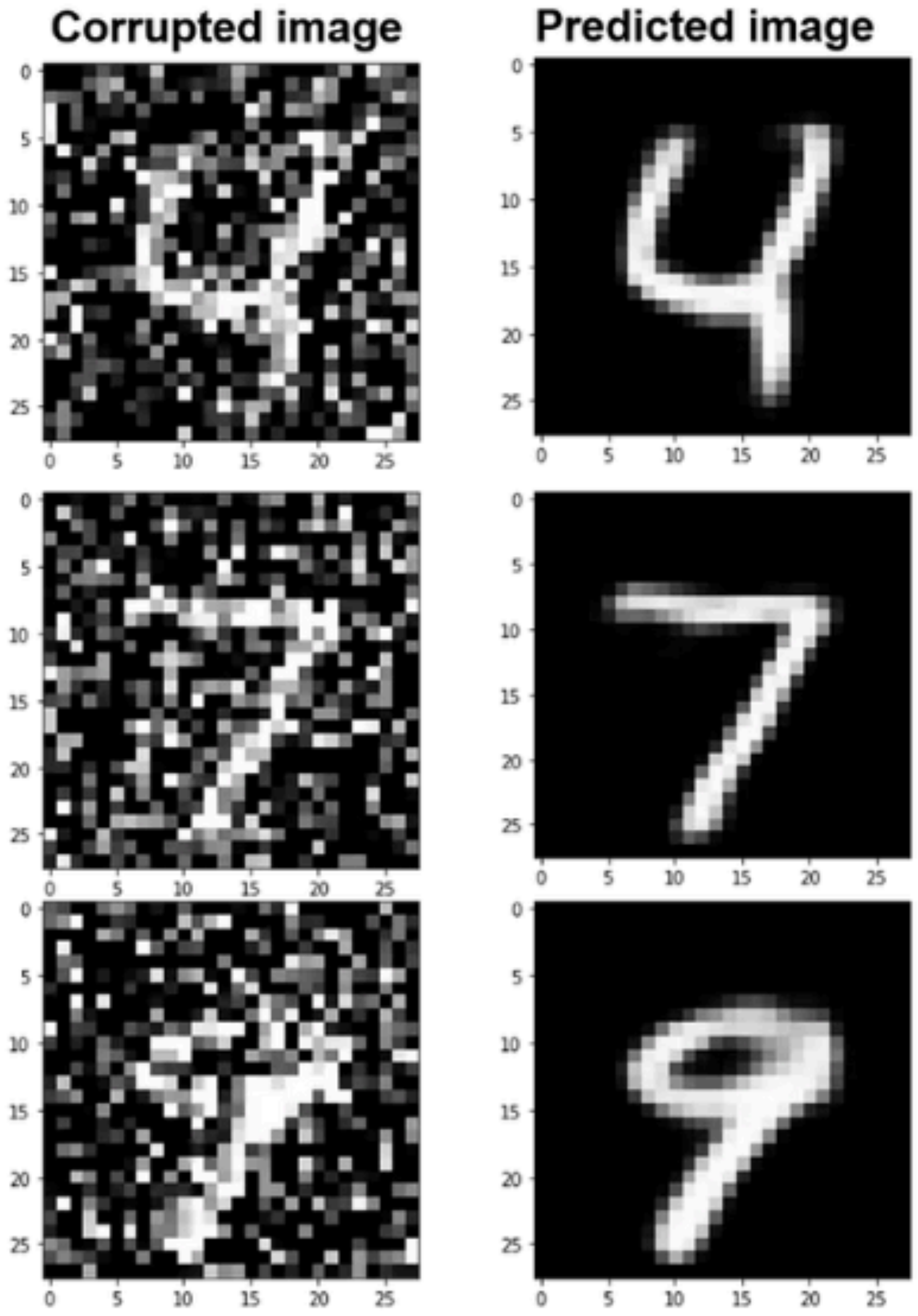


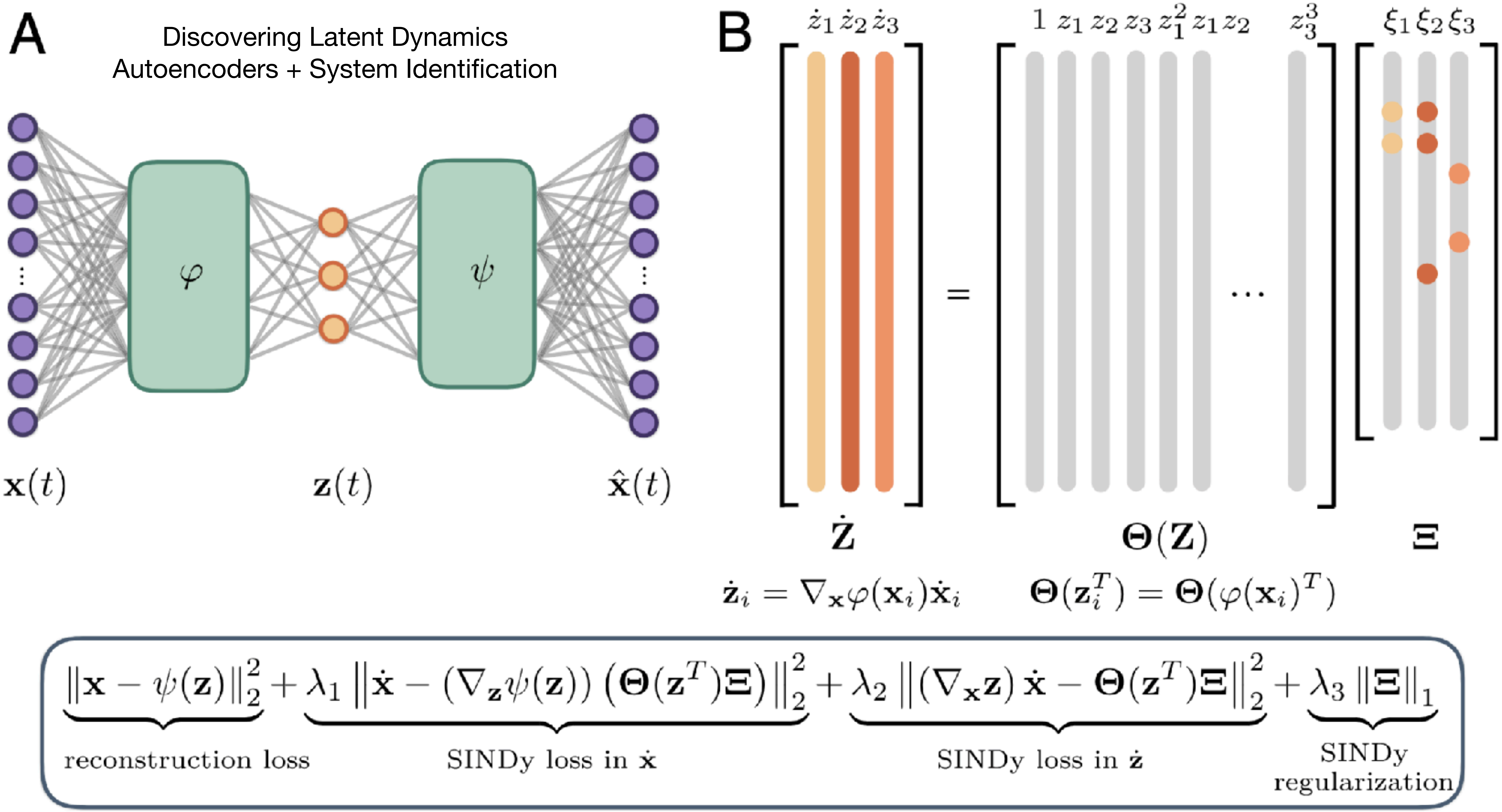
$$\mathcal{L} = \|f_{\mathbf{W}_1 \mathbf{W}_2}(\mathbf{x}) - \mathbf{x}\|^2$$

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

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


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

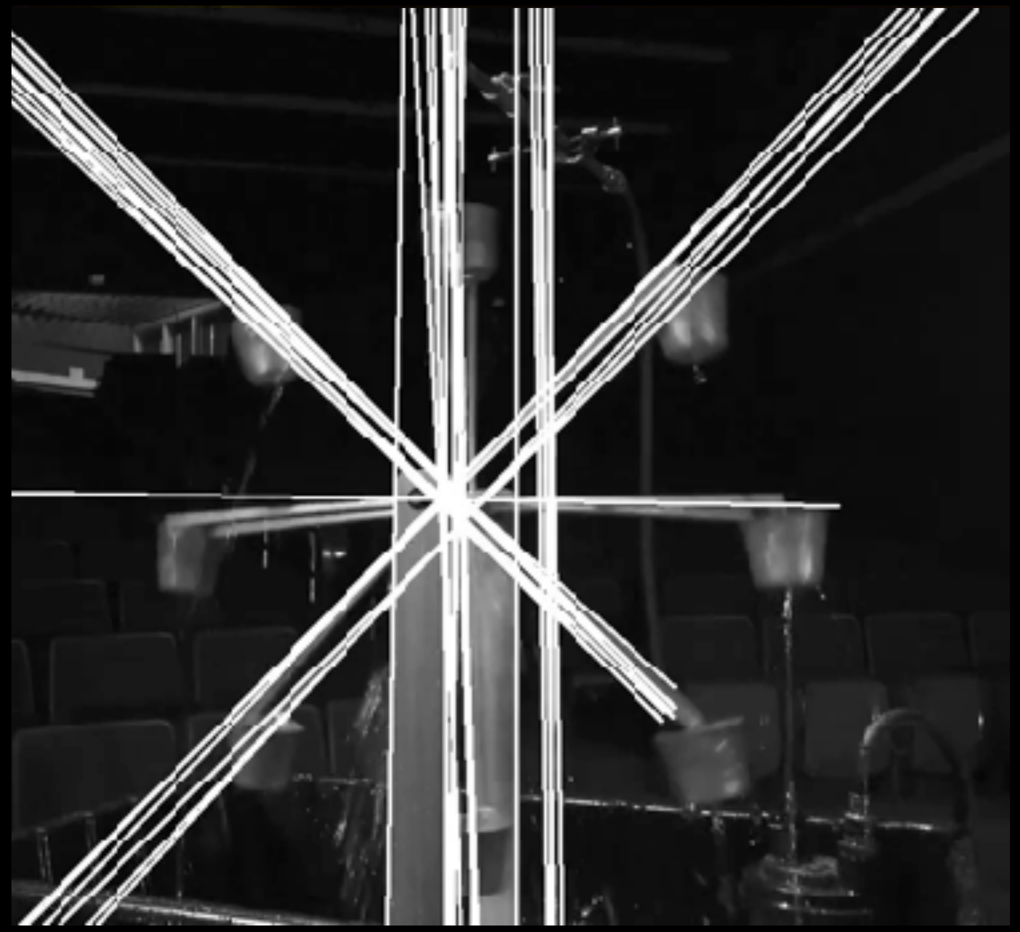

Corrupted image
Predicted image



Champion, Kathleen, et al. "Data-driven discovery of coordinates and governing equations." Proceedings of the National Academy of Sciences 116.45 (2019): 22445-22451.

Discovering Dynamics from Video

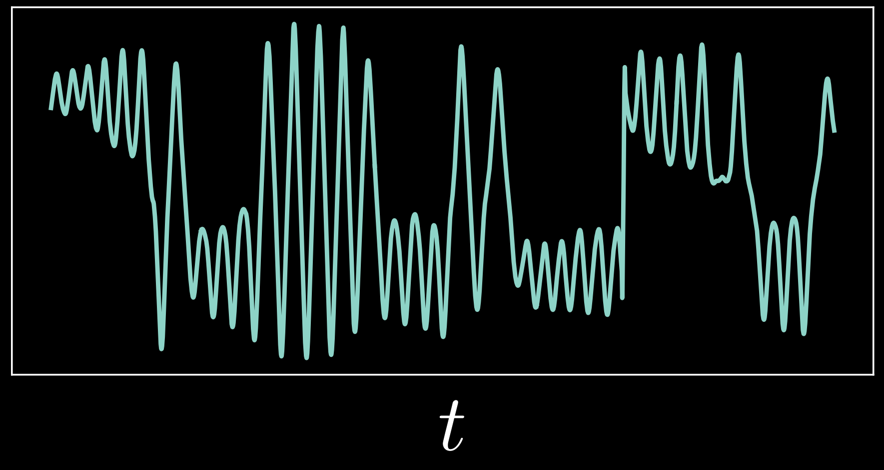
Lorenz waterwheel



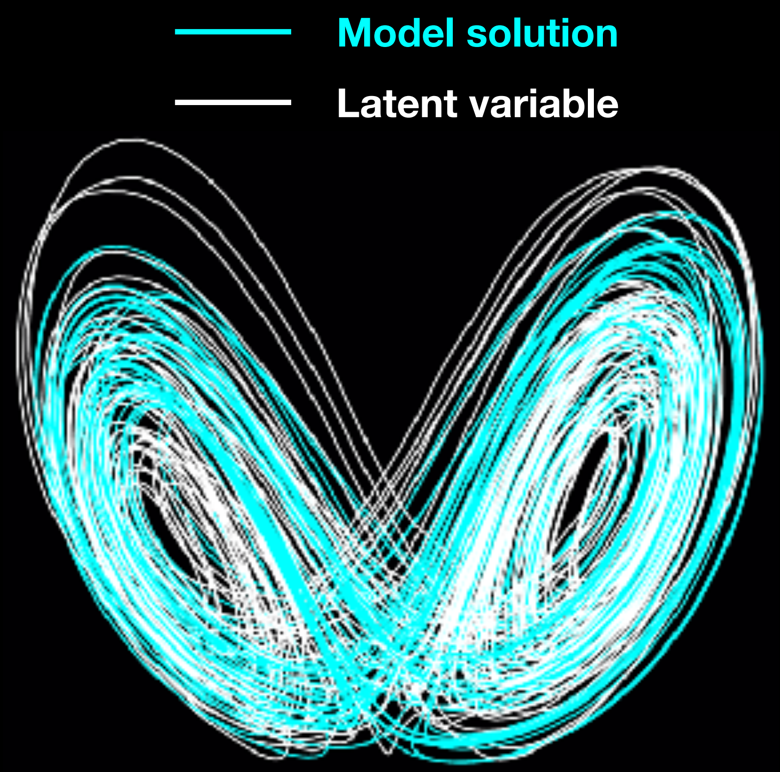
Harvard Natural Sciences Lecture Demonstrations
https://youtu.be/Lx8gMBJBlP8

Extract

Angular velocity



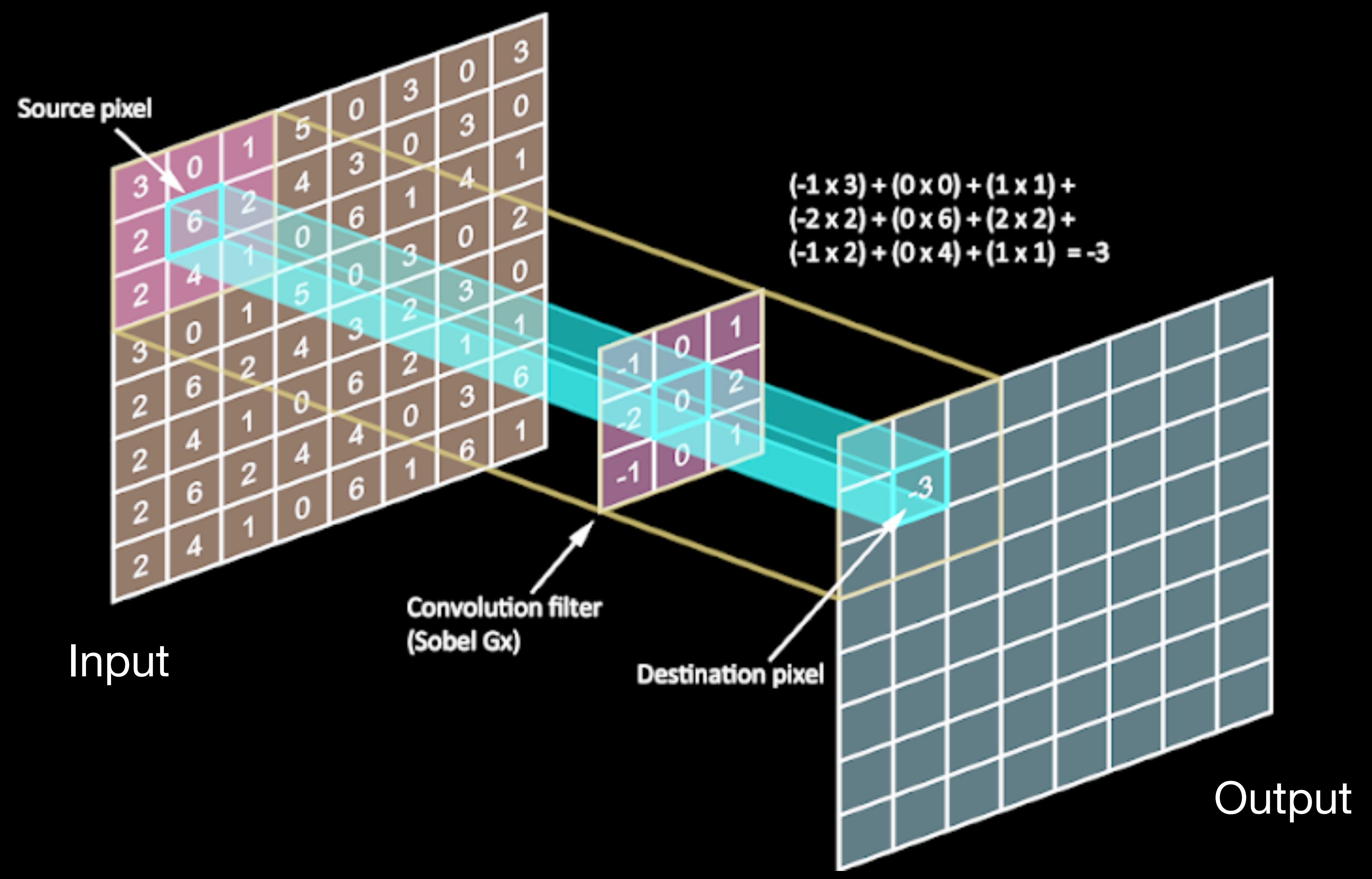
Delay SINDY Autoencoder

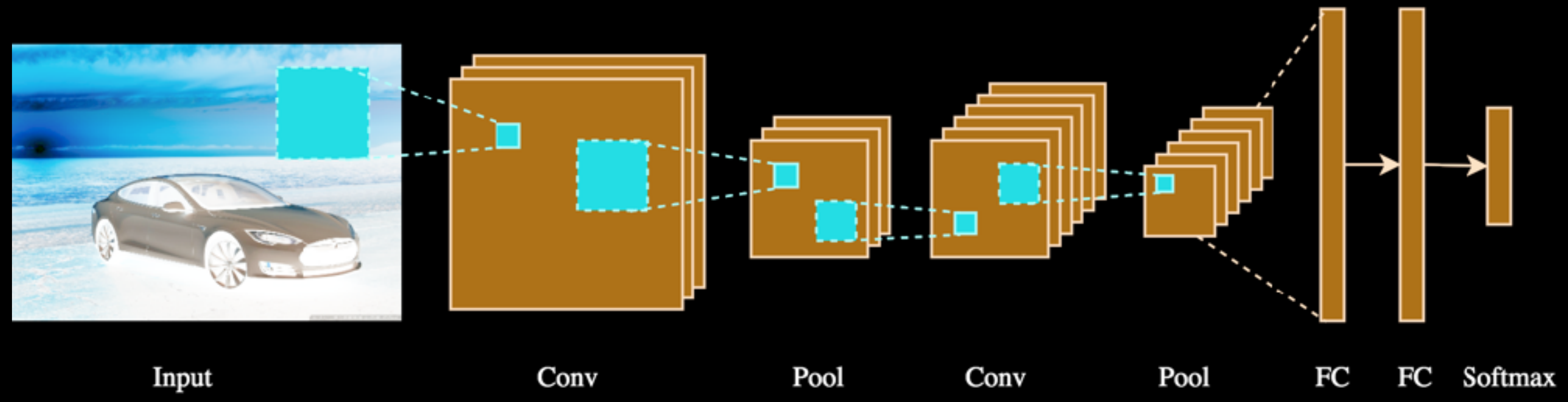


θ	1	z_1	z_2	z_3	z_1^2	$z_1 z_2$	$z_1 z_3$	z_2^2	$z_1 z_2$	z_3^2	z_1^3	$z_1^2 z_2$	$z_1^2 z_3$	$z_1 z_2^2$	$z_1 z_2 z_3$	$z_1 z_3^2$	z_2^3	$z_2^2 z_3$	$z_2 z_3^2$	z_3^3
$\dot{z}_1$	-0.12	1.70	0.04			-1.79		0.16									-0.02	0.03	-0.37	
$\dot{z}_2$	-1.05	0.05	-0.28		0.12		1.50	0.60							-0.10			0.18	-0.33	
$\dot{z}_3$	0.08	-0.12	0.30	0.07	0.08	-0.29	-0.22	0.35	-0.55			0.06		-0.07	0.20	0.03	-0.25	0.28		

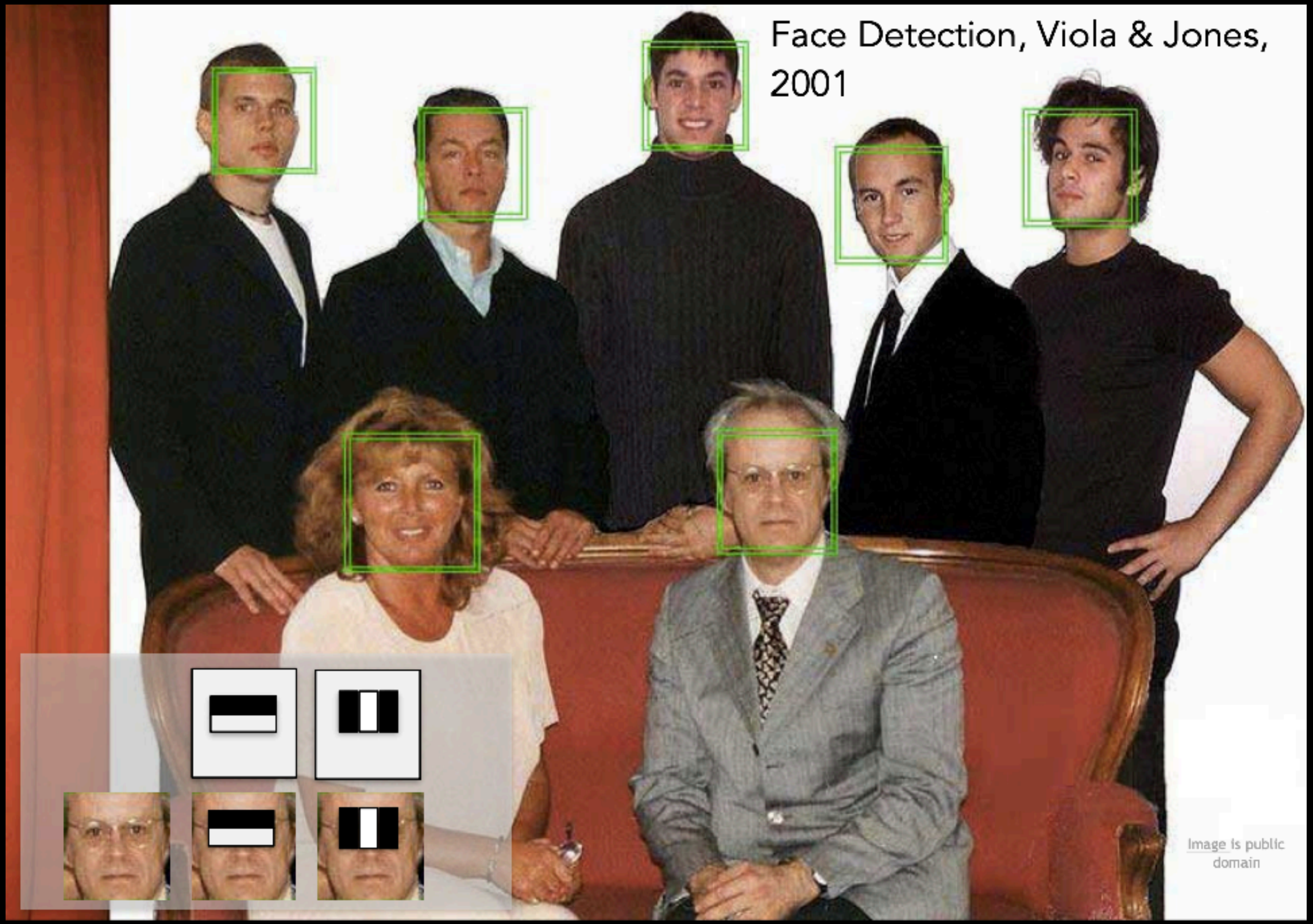
Bakarji, Joseph, et al. "Discovering governing equations from partial measurements with deep delay autoencoders." Proceedings of the Royal Society A 479.2276 (2023): 20230422.

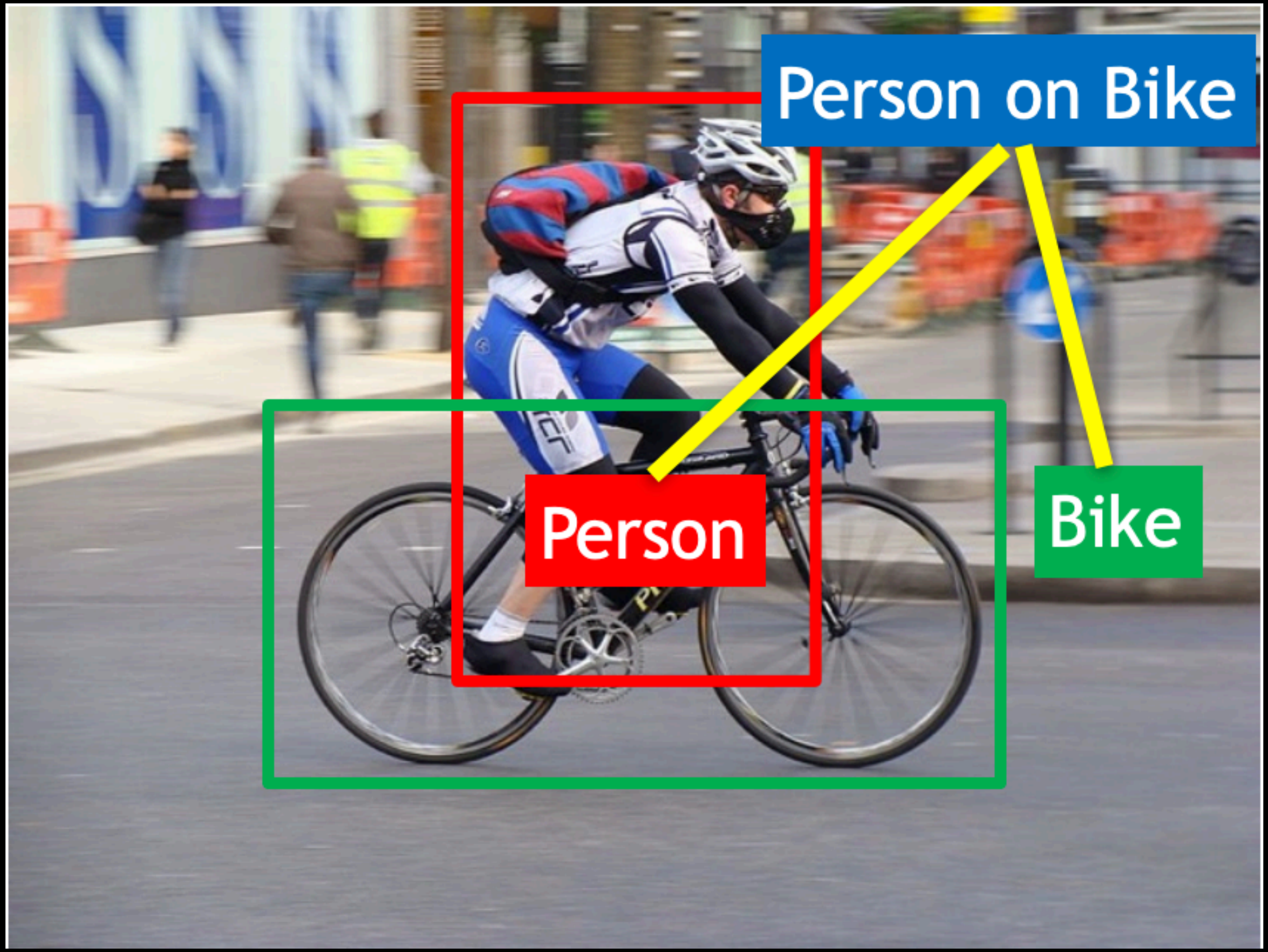
Convolutional neural networks



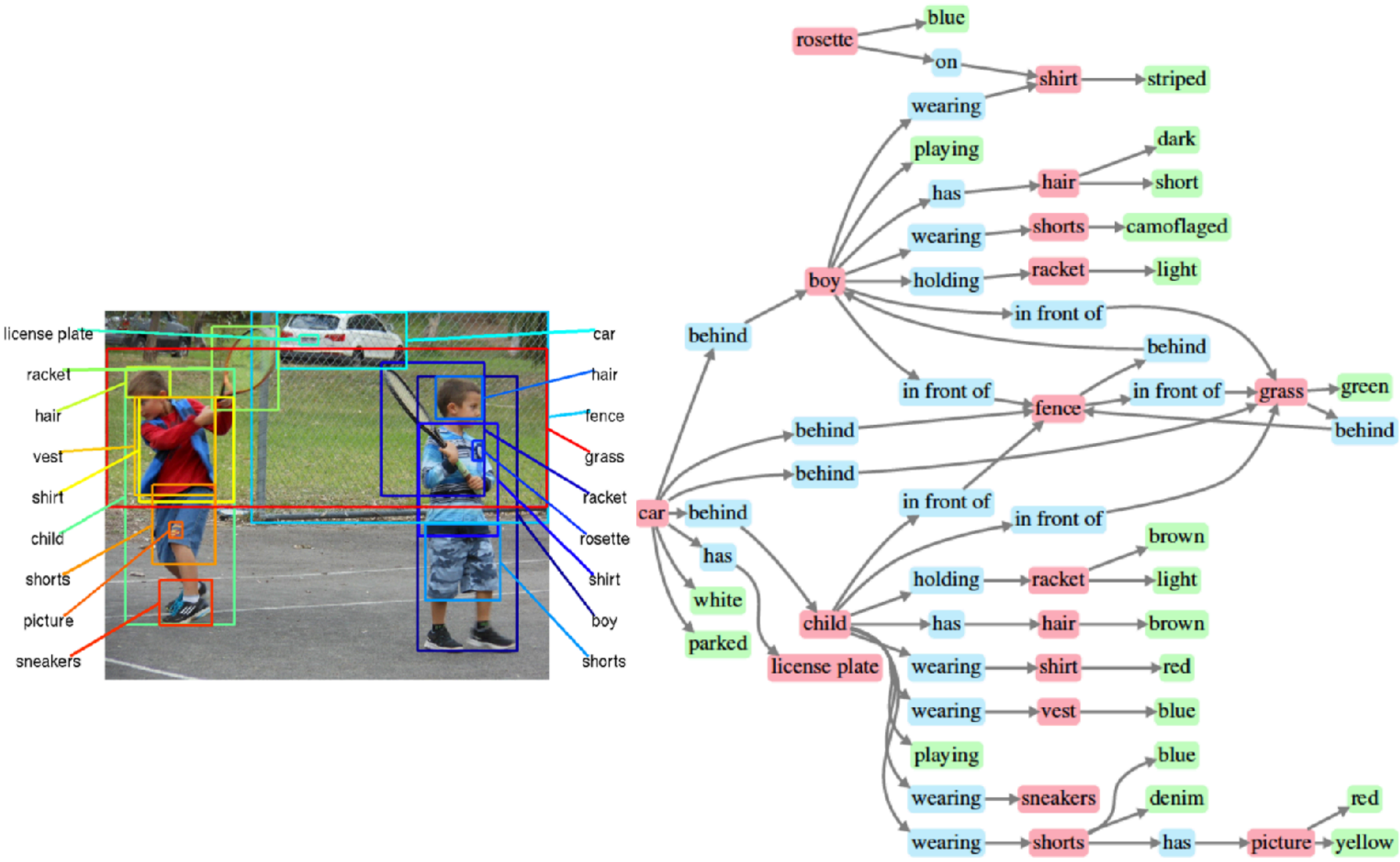


https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2



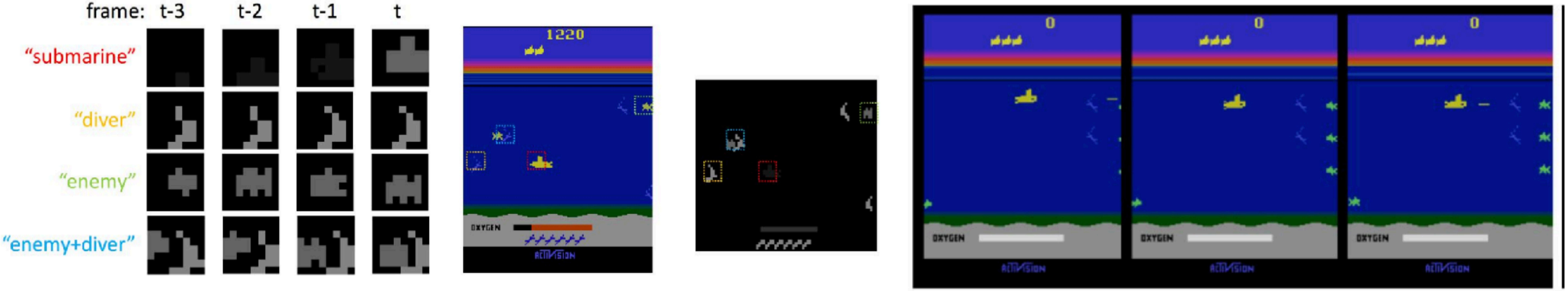


The following slides were taken from Stanford's CS231:
- https://cs231n.github.io/
- https://www.youtube.com/playlist?list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv



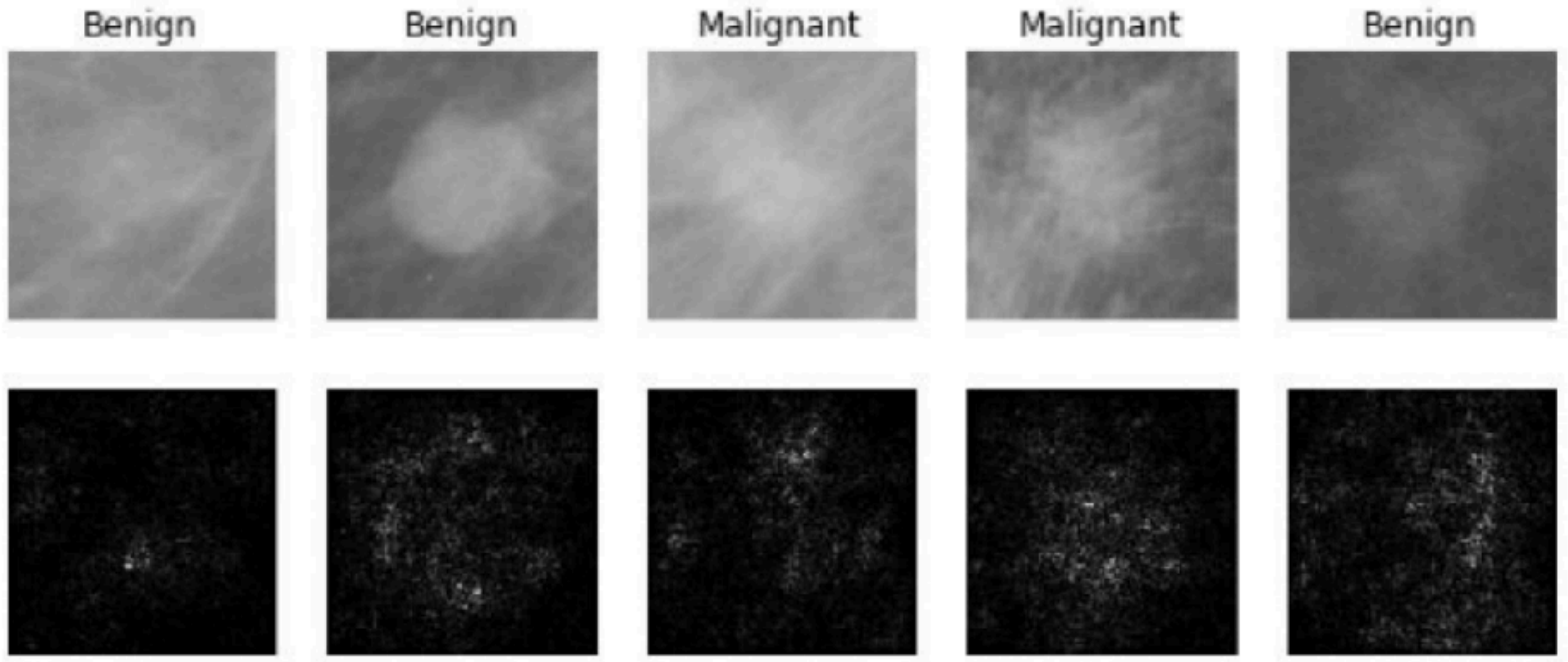
Johnson *et al.*, "Image Retrieval using Scene Graphs", CVPR 2015

[Toshev, Szegedy 2014]



[Guo et al. 2014]

Fast-forward to today: ConvNets are everywhere



[Levy et al. 2016]

[Dieleman et al. 2014]

[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

Minor errors

A man in a baseball uniform throwing a ball

Somewhat related

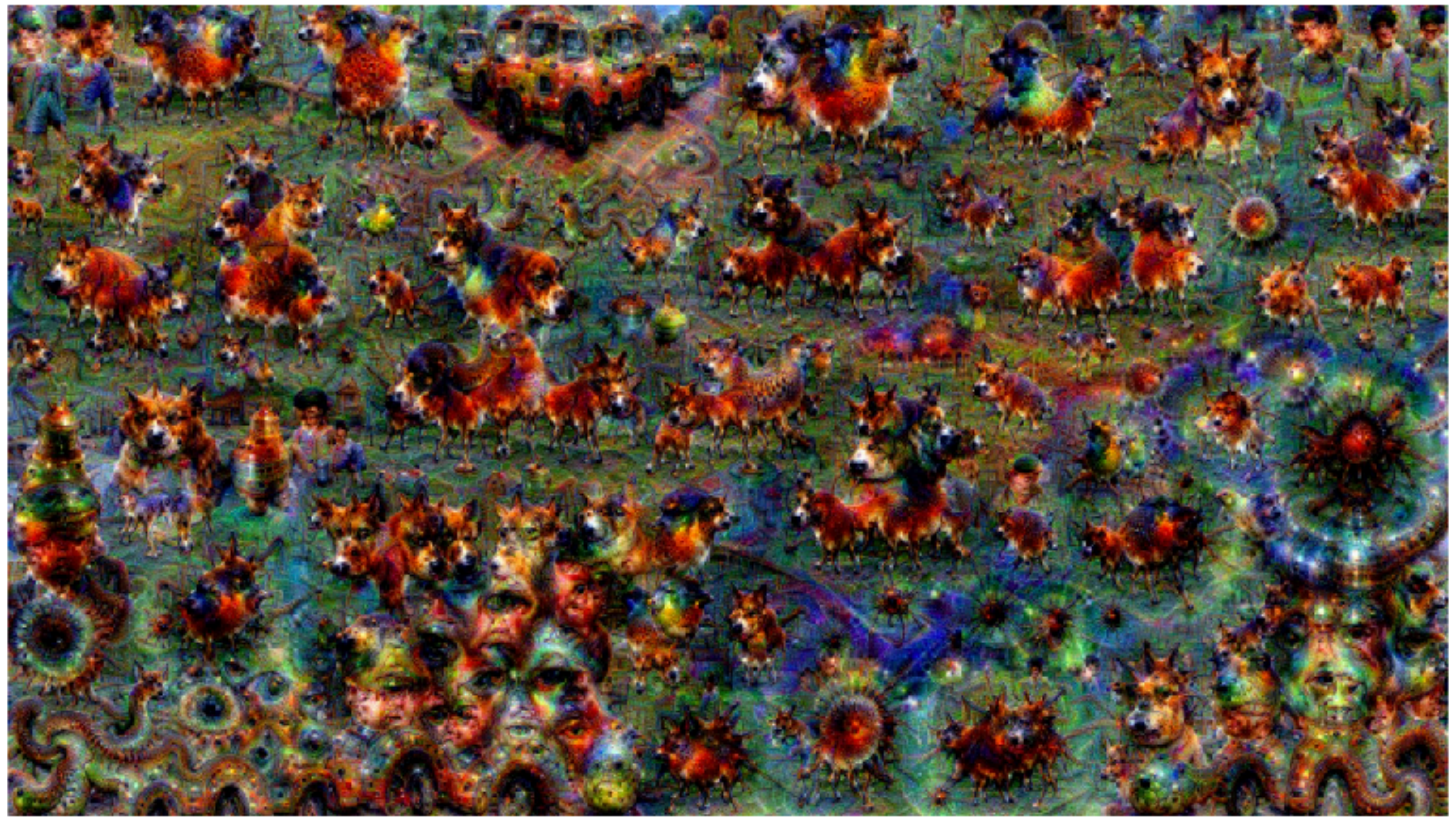
A woman is holding a cat in her hand

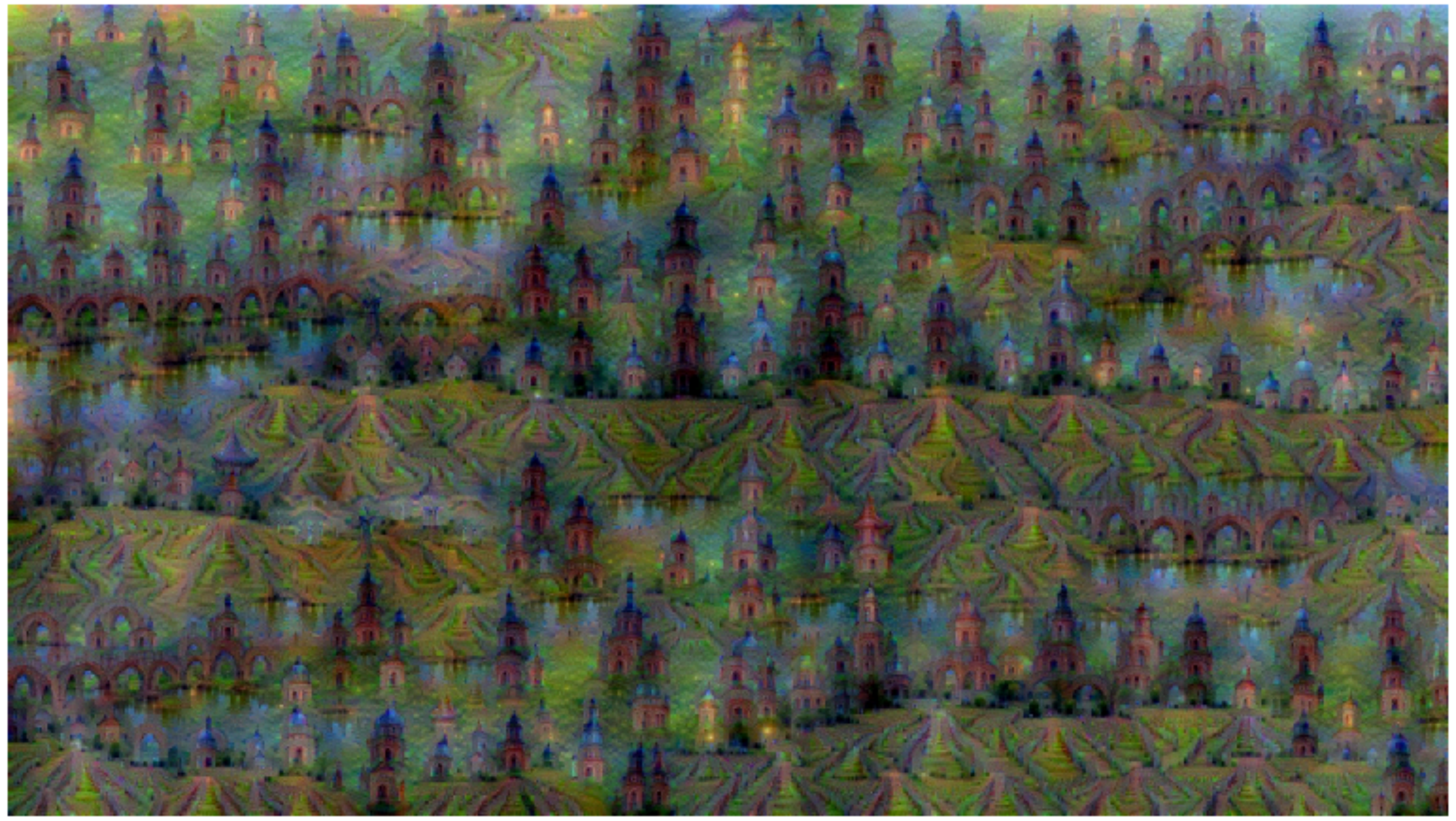
A man riding a wave on top of a surfboard

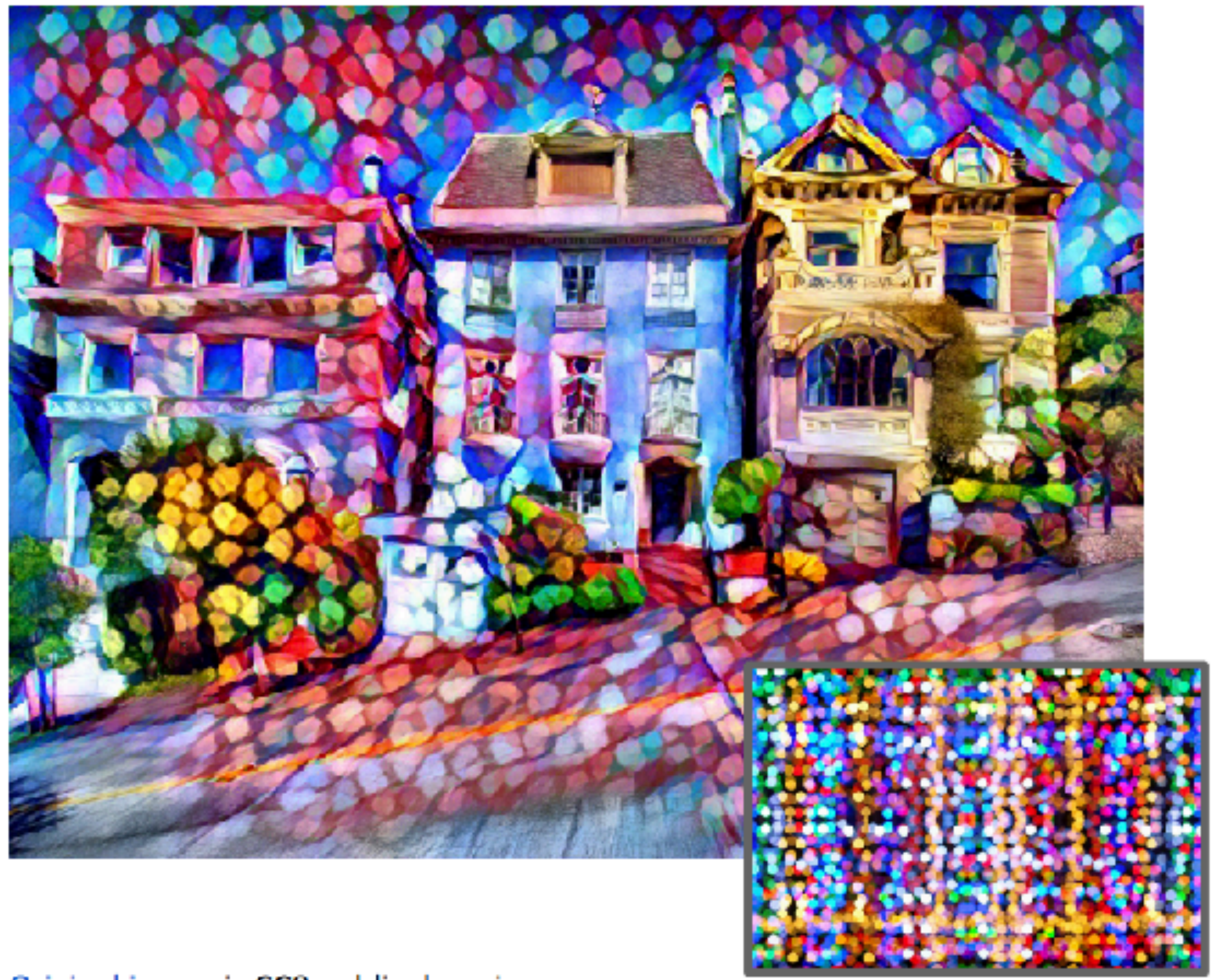
A cat sitting on a suitcase on the floor

A woman standing on a beach holding a surfboard

Captions generated by Justin Johnson using Neuraltalk2







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
<u>Steel drum</u>
Drumstick
Mud turtle

Output:
Scale
T-shirt
Giant panda
Drumstick
Mud turtle

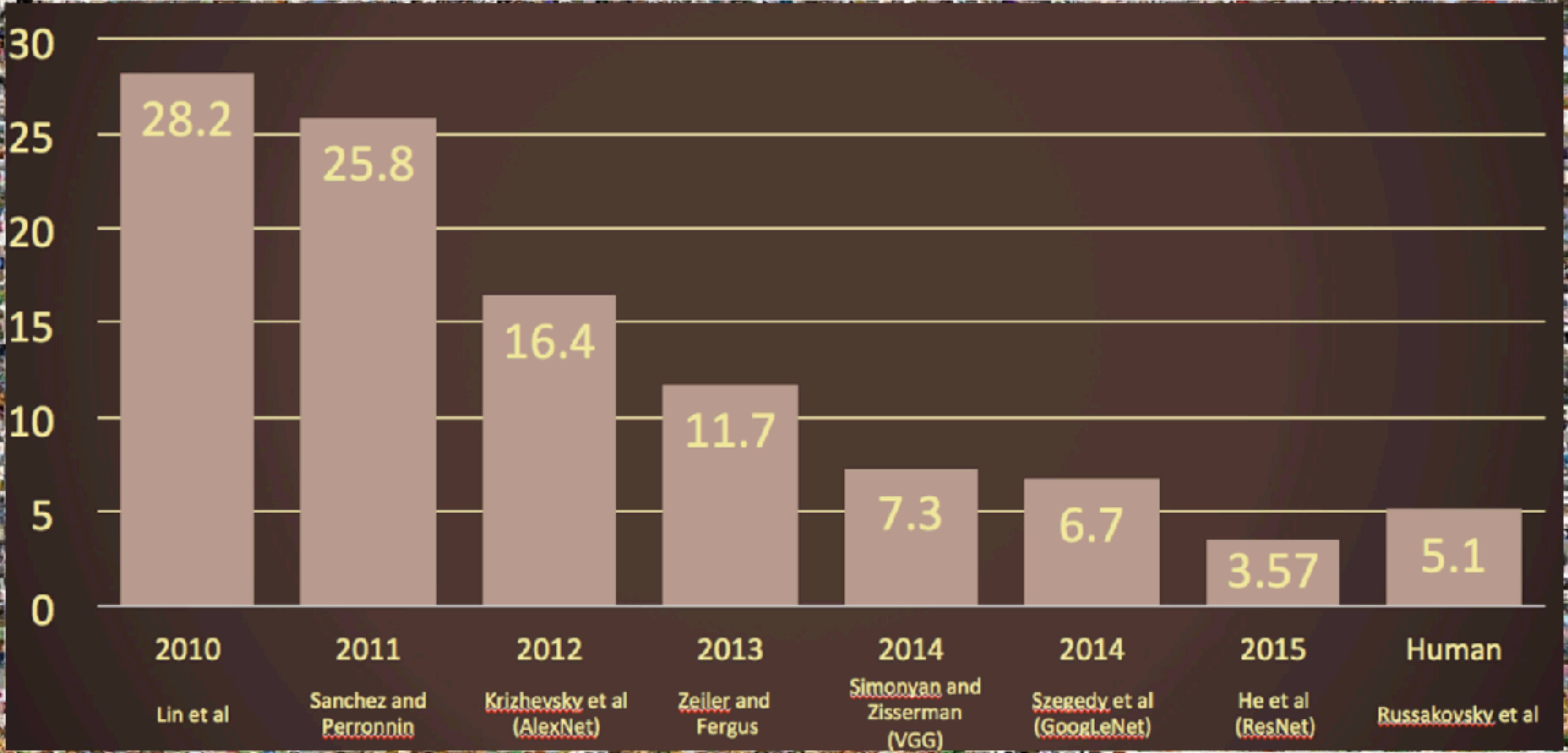
Russakovsky et al. arXiv, 2014

IMAGENET Large Scale Visual Recognition Challenge

The Image Classification Challenge:

1,000 object classes

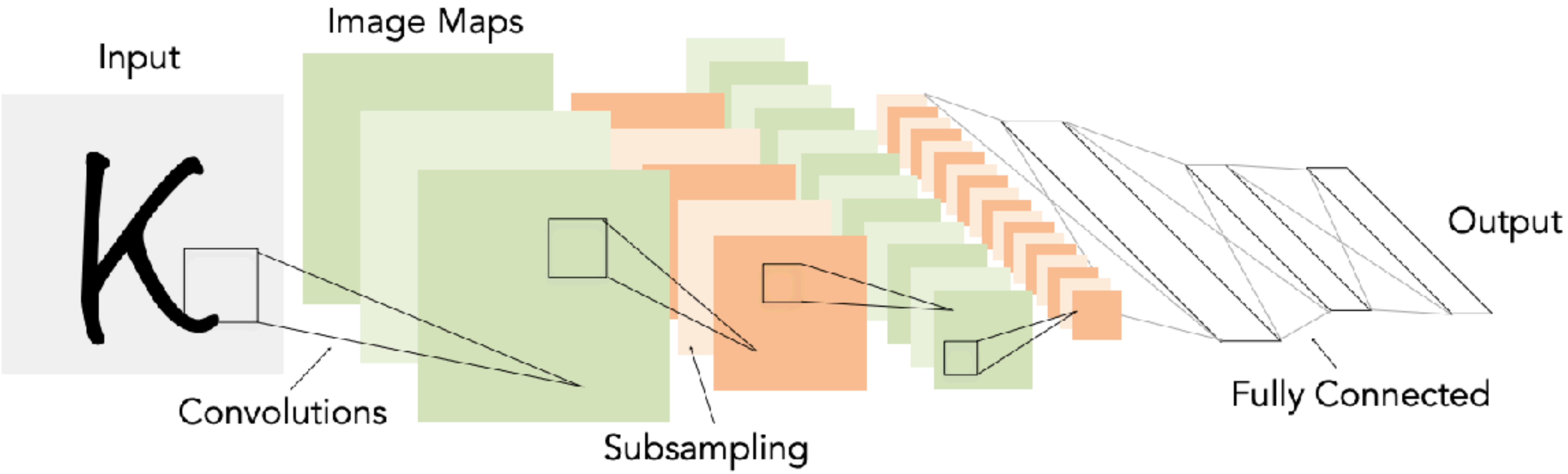
1,431,167 images



Russakovsky et al. arXiv, 2014

1998

LeCun et al.



# of transistors

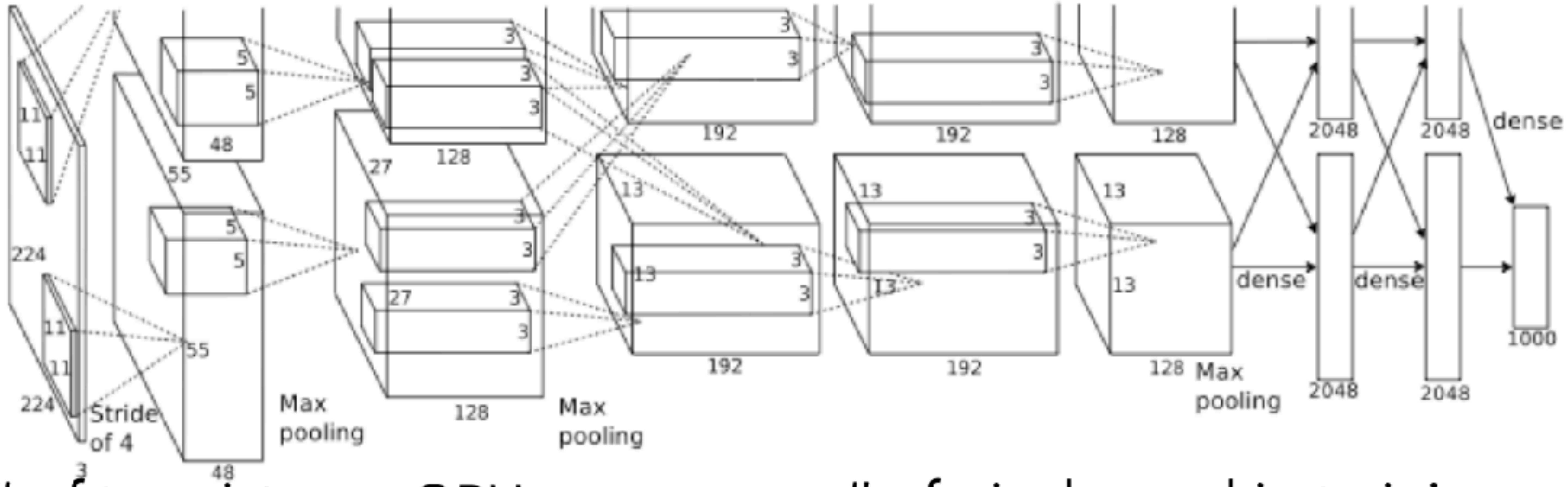
10^6

# of pixels used in training

10^7 NIST

2012

Krizhevsky et al.



# of transistors

10^9

GPUs

# of pixels used in training

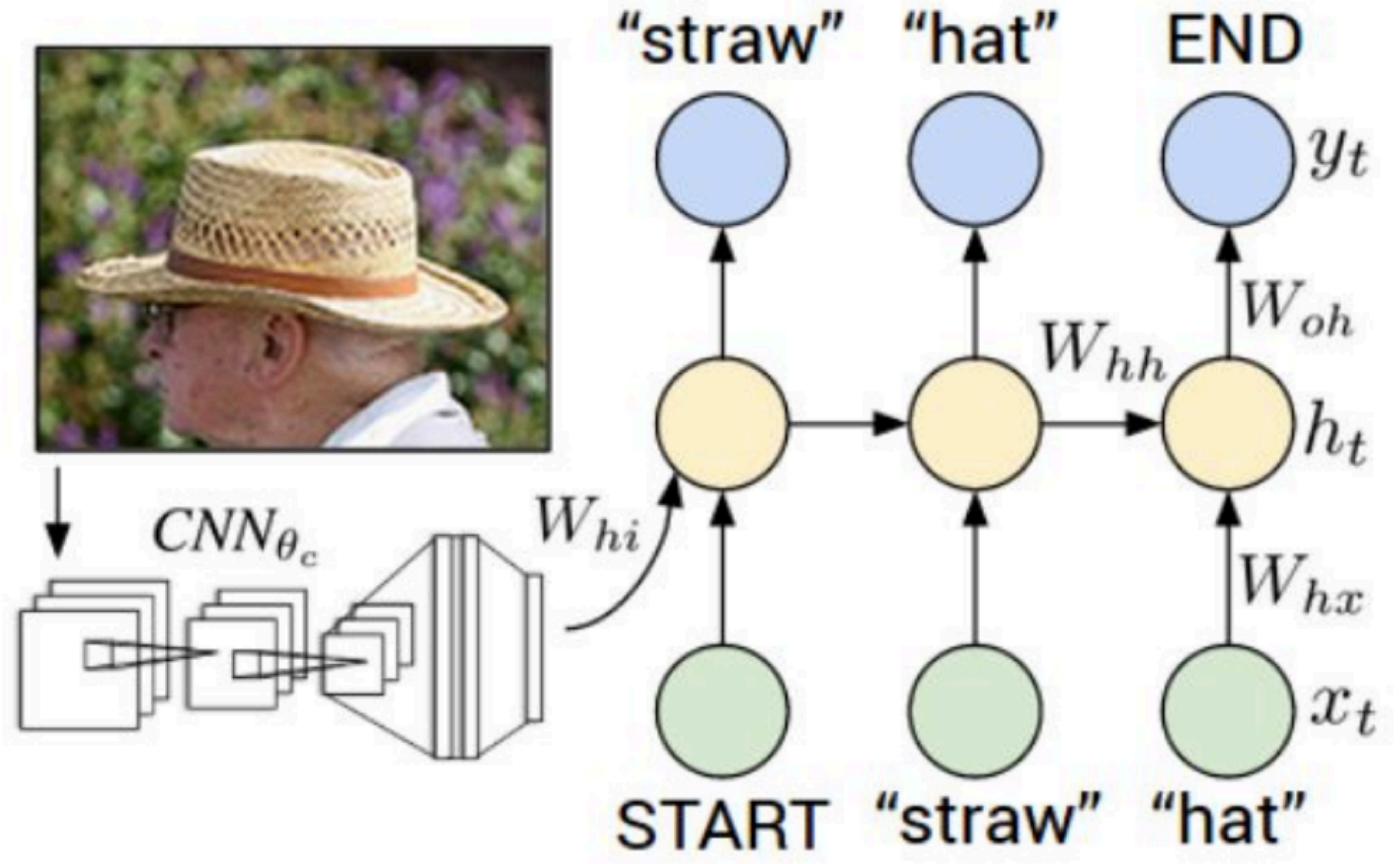
10^{14} IMAGENET

Two young girls are playing with lego toy.

Boy is doing backflip on wakeboard

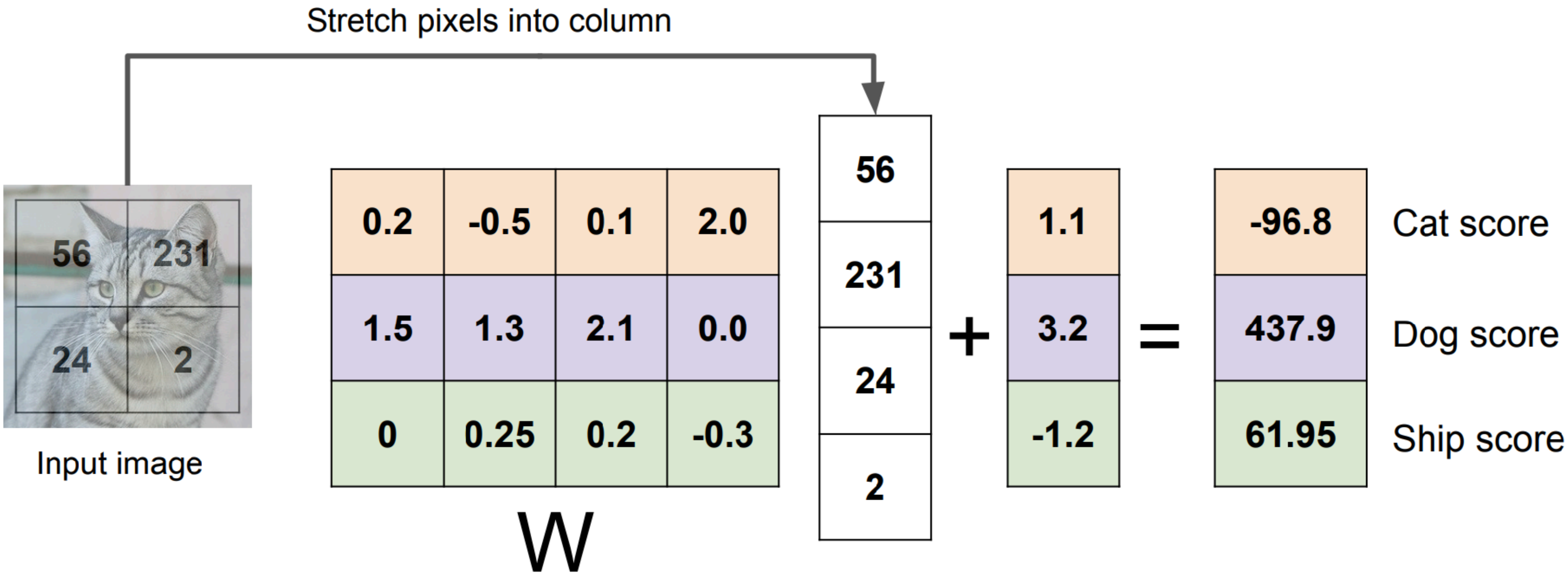
Man in black shirt is playing guitar.

Construction worker in orange safety vest is working on road.



Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)



Interpreting a Linear Classifier

$$f(x,W) = Wx + b$$

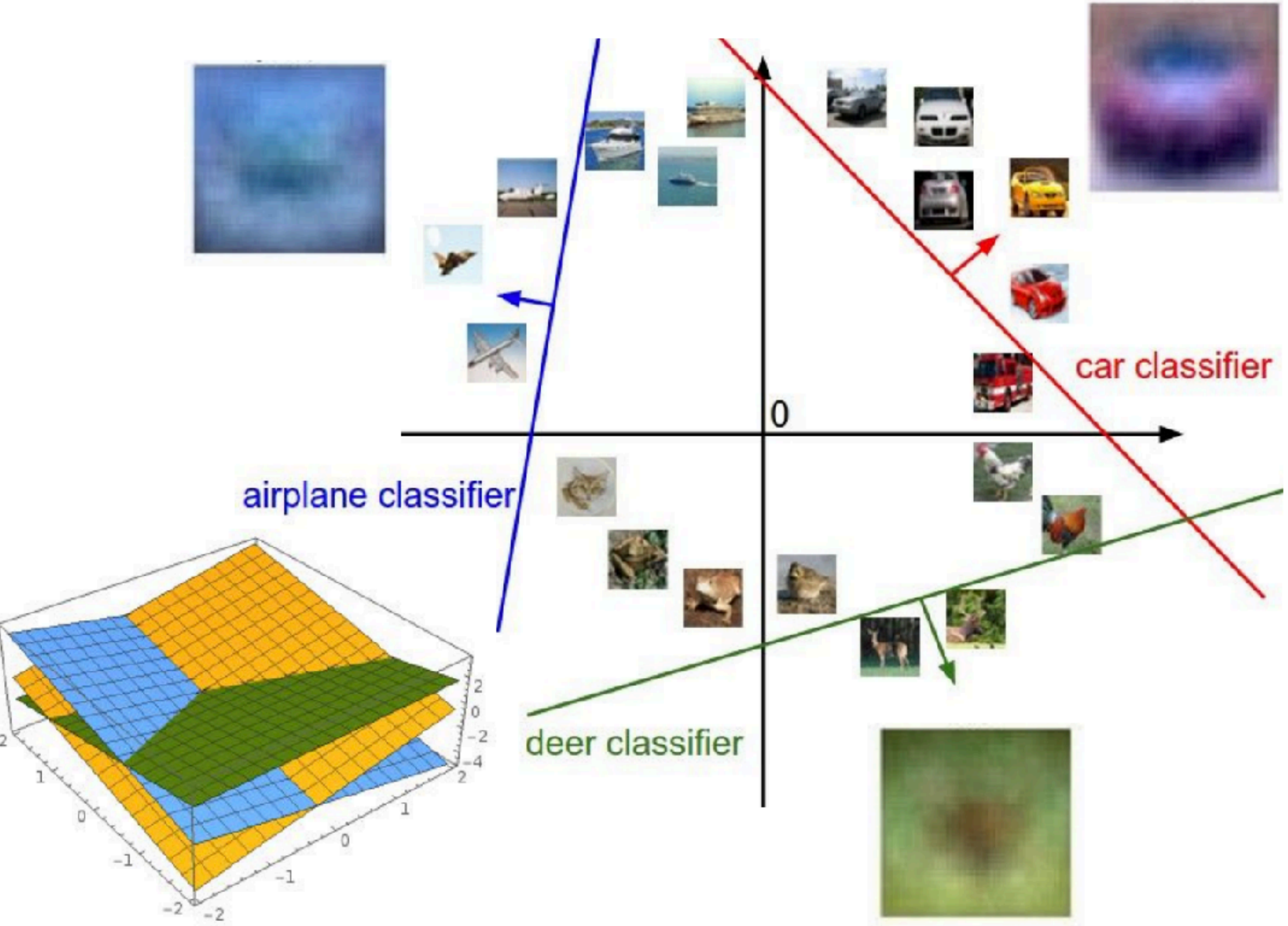
What is this thing doing?

Interpreting a Linear Classifier

$$f(x,W) = Wx + b$$

Example trained weights of a linear classifier trained on CIFAR-10:

Interpreting a Linear Classifier



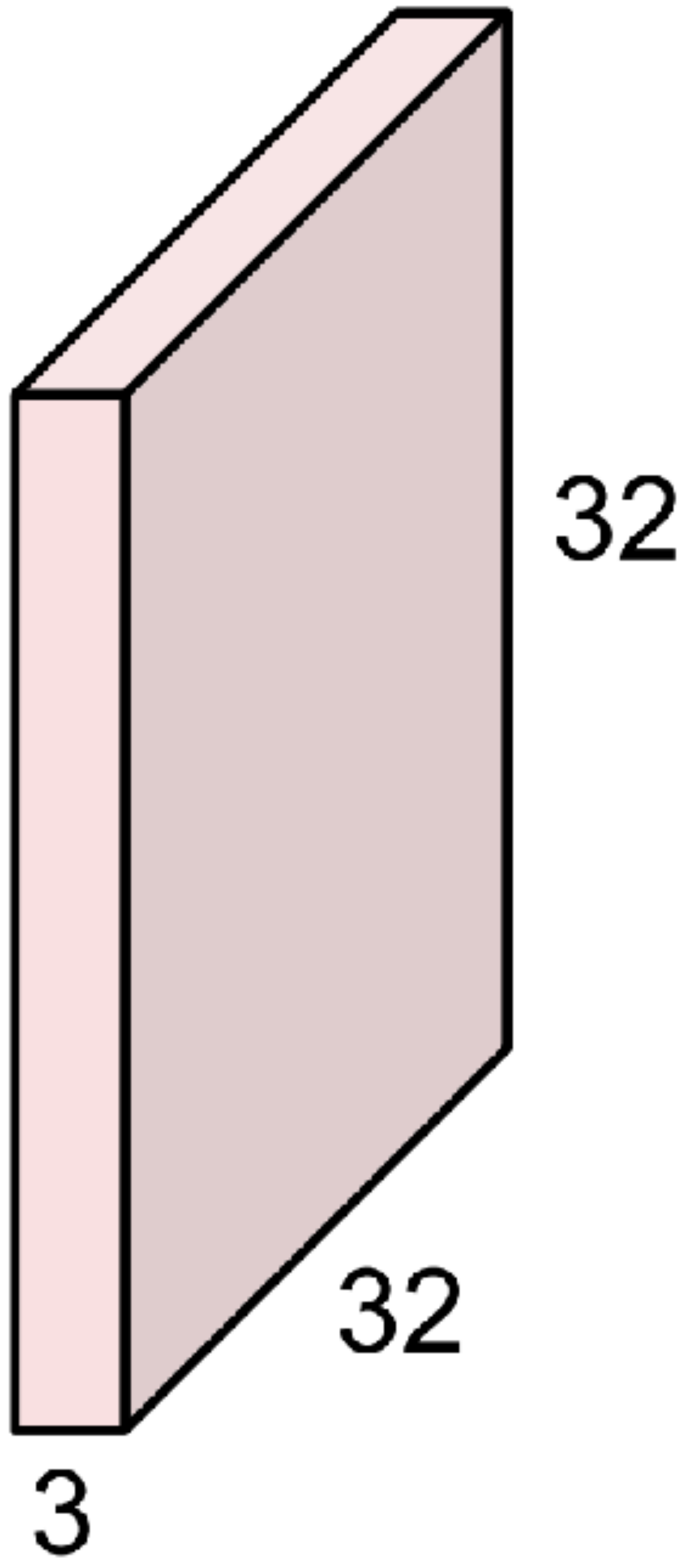
$$f(x,W) = Wx + b$$

Array of **32x32x3** numbers
(3072 numbers total)

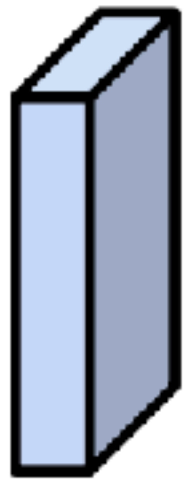
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

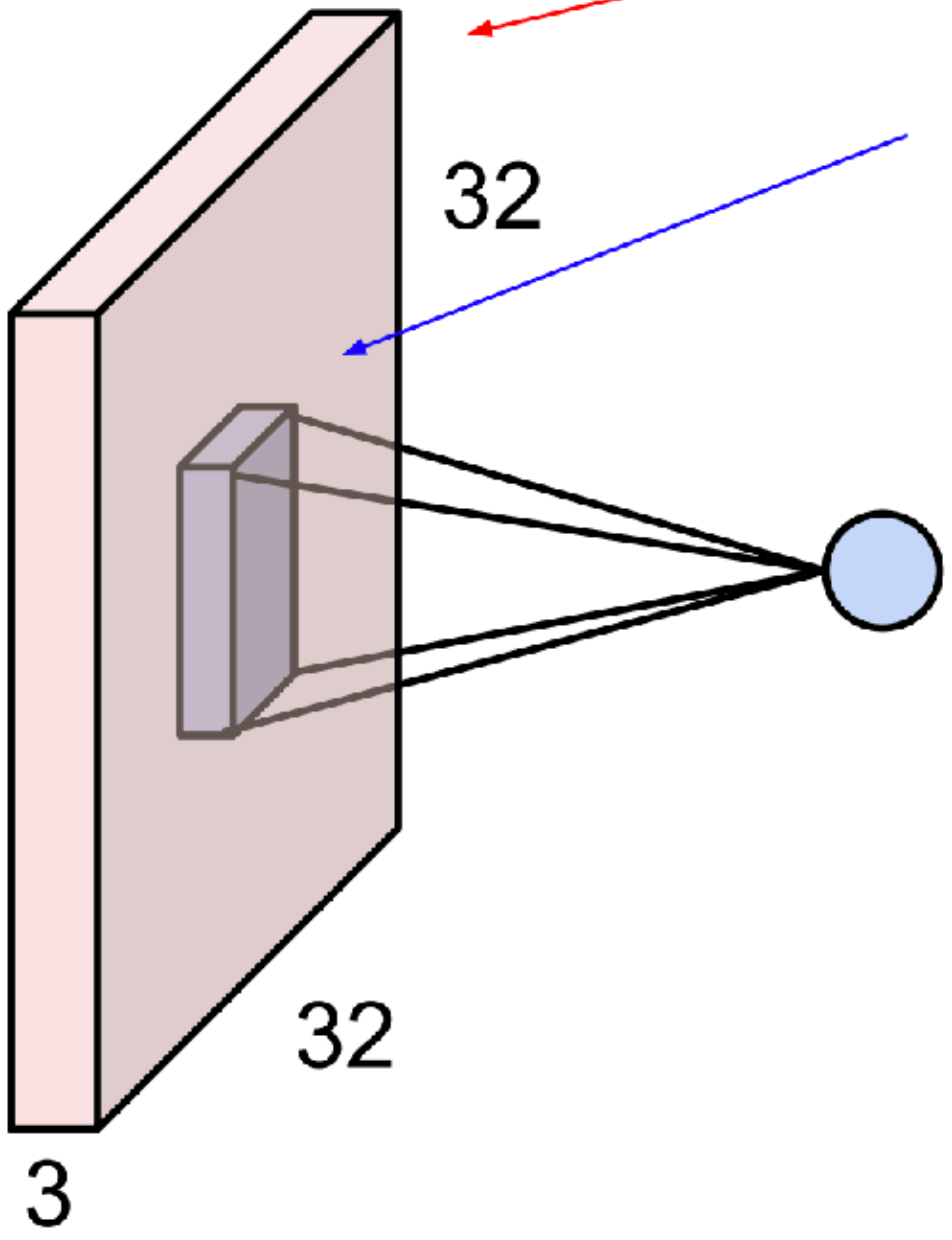


5x5x3 filter



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

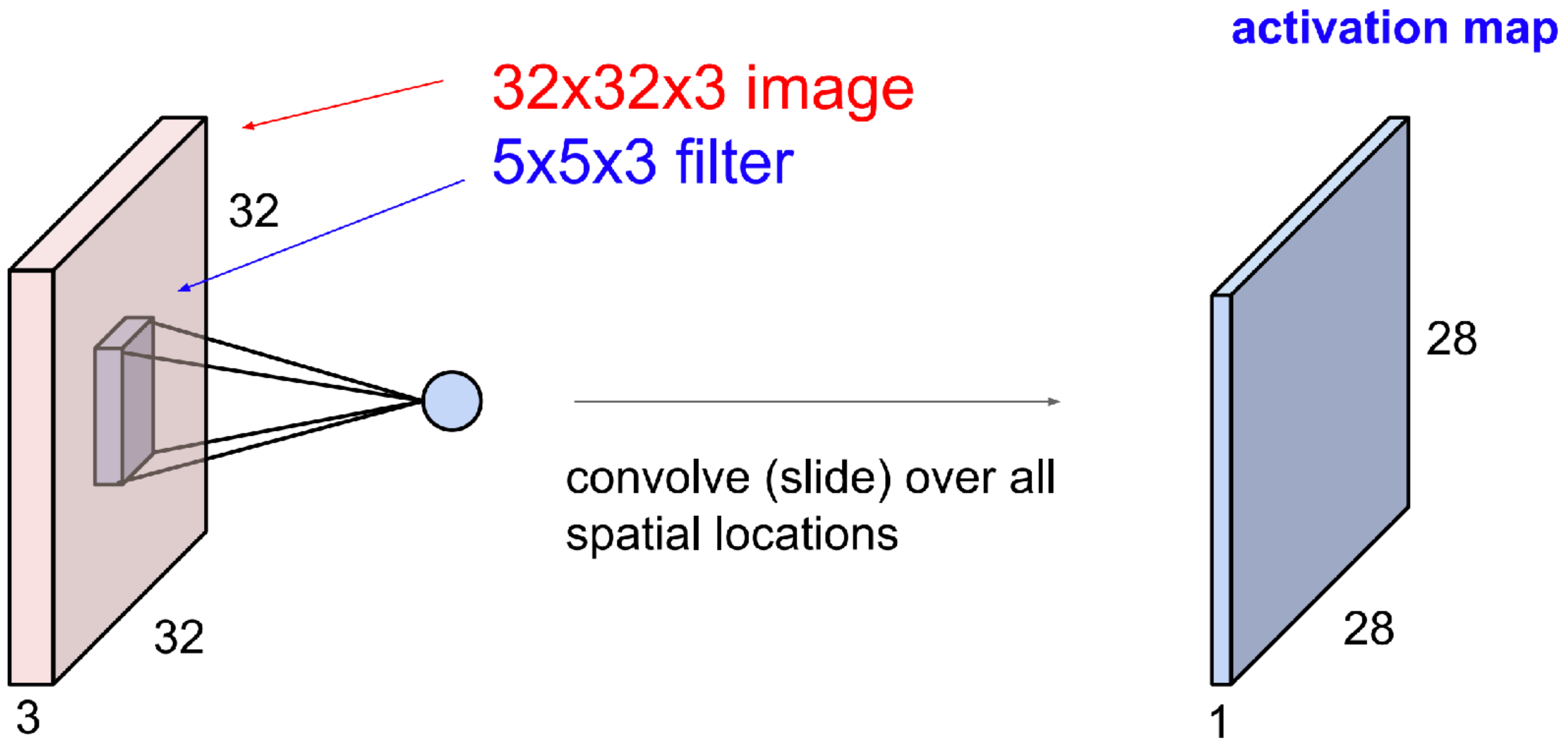
Convolution Layer



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)

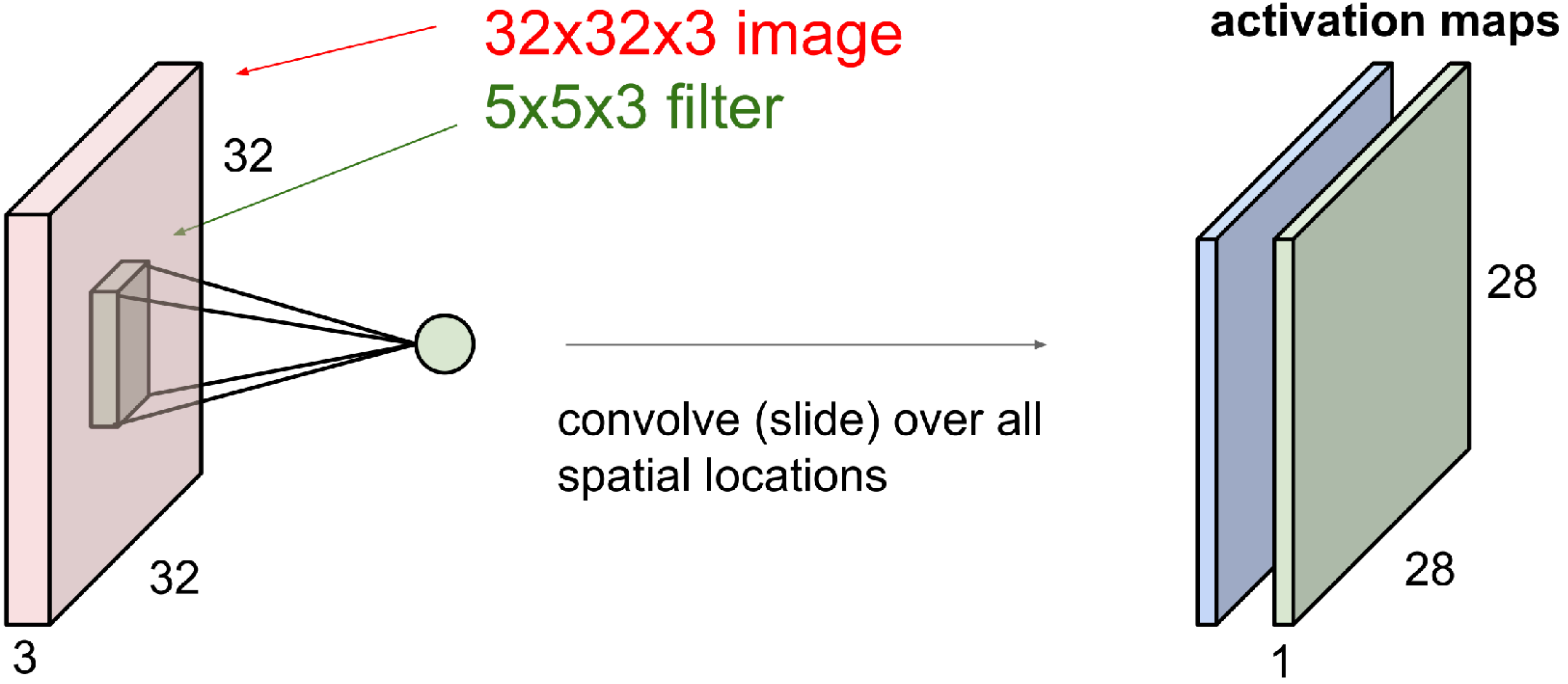
$$w^T x + b$$

Convolution Layer

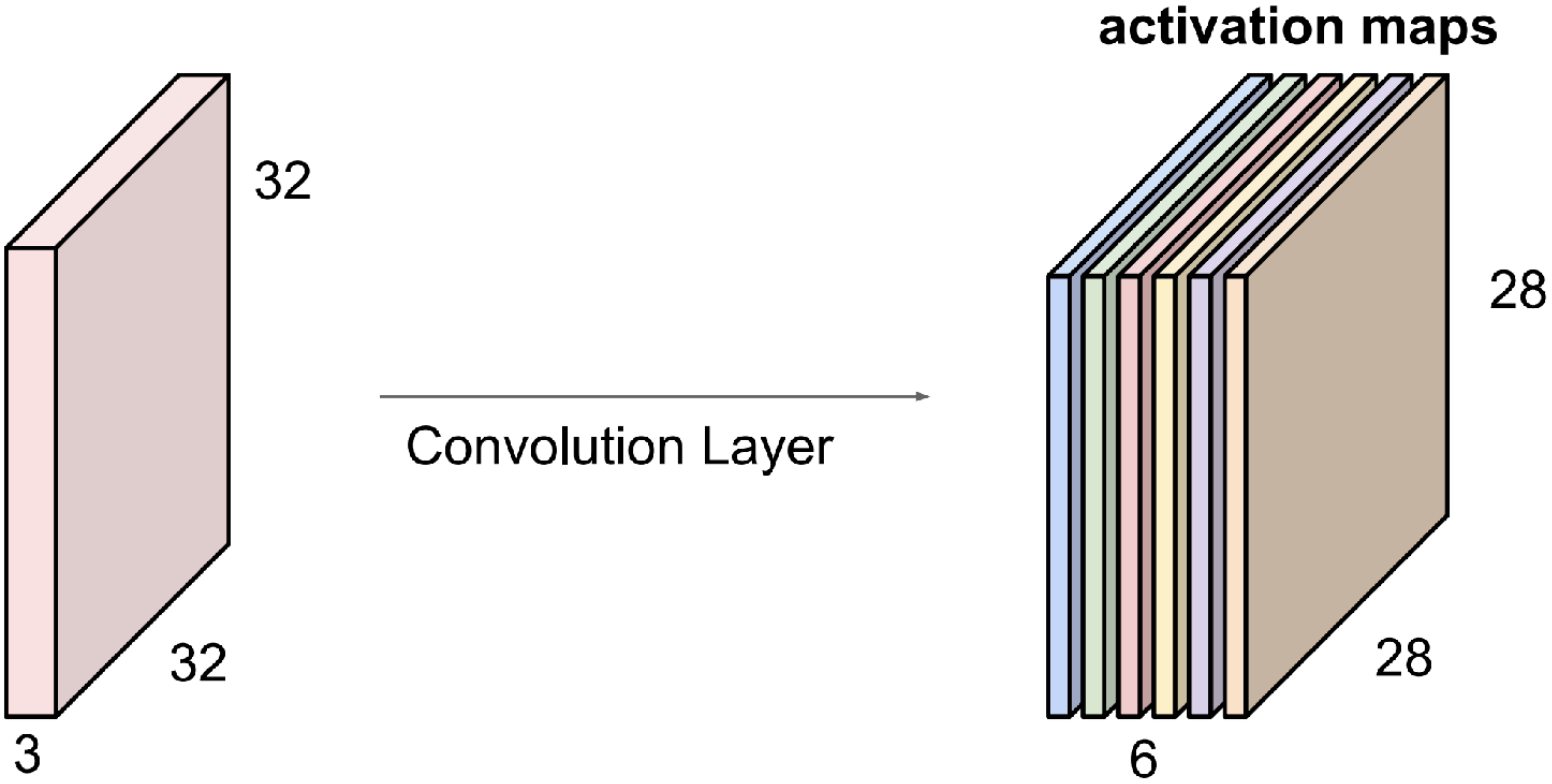


Convolution Layer

consider a second, green filter

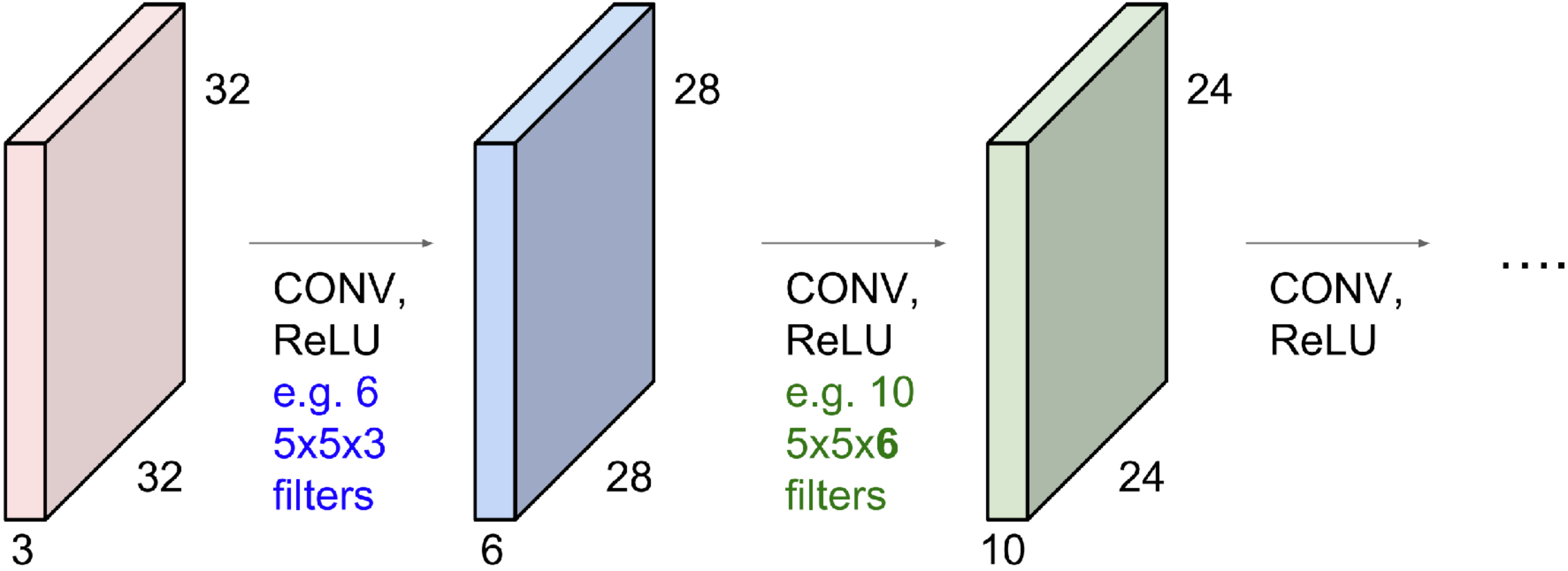


For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



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

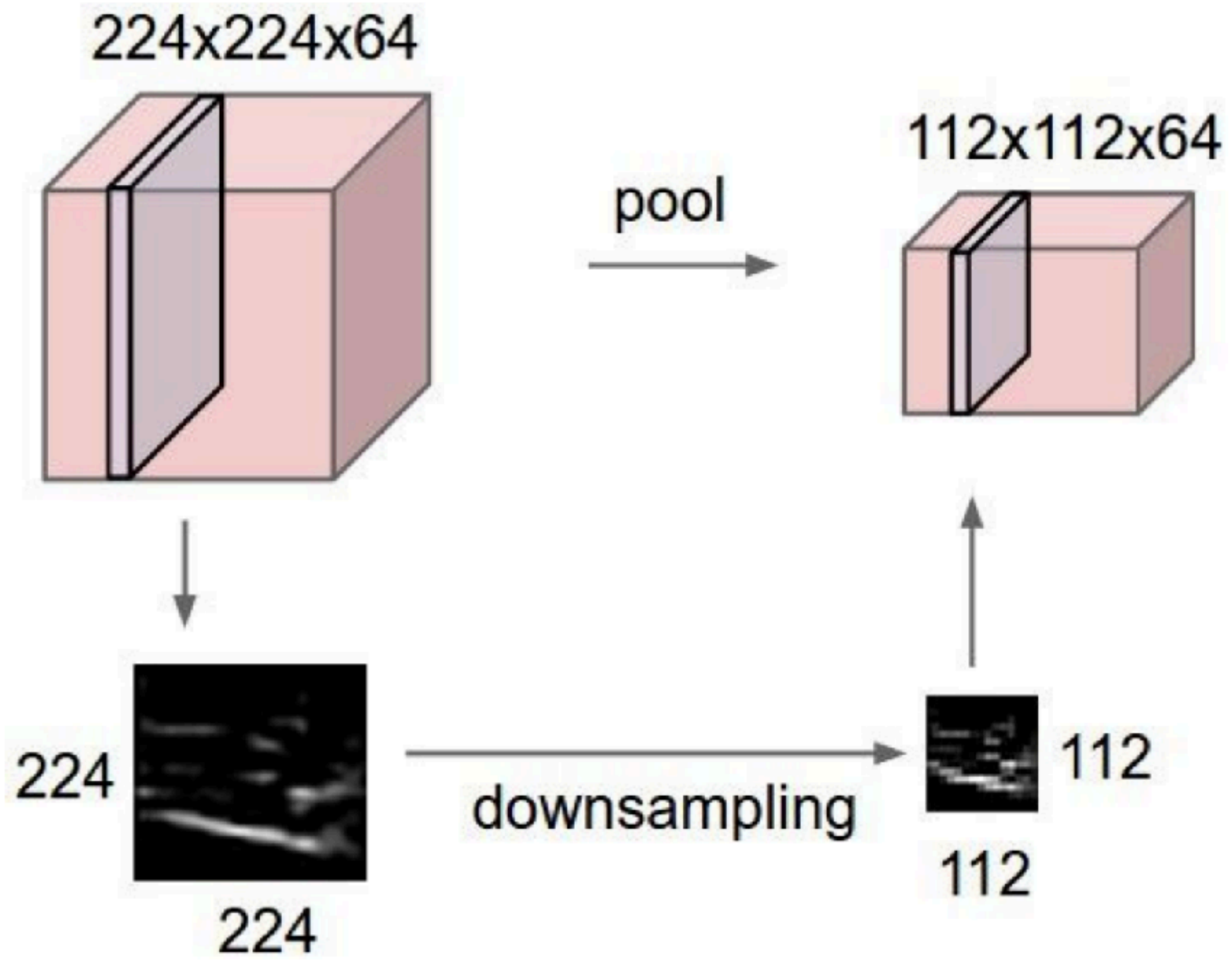
Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



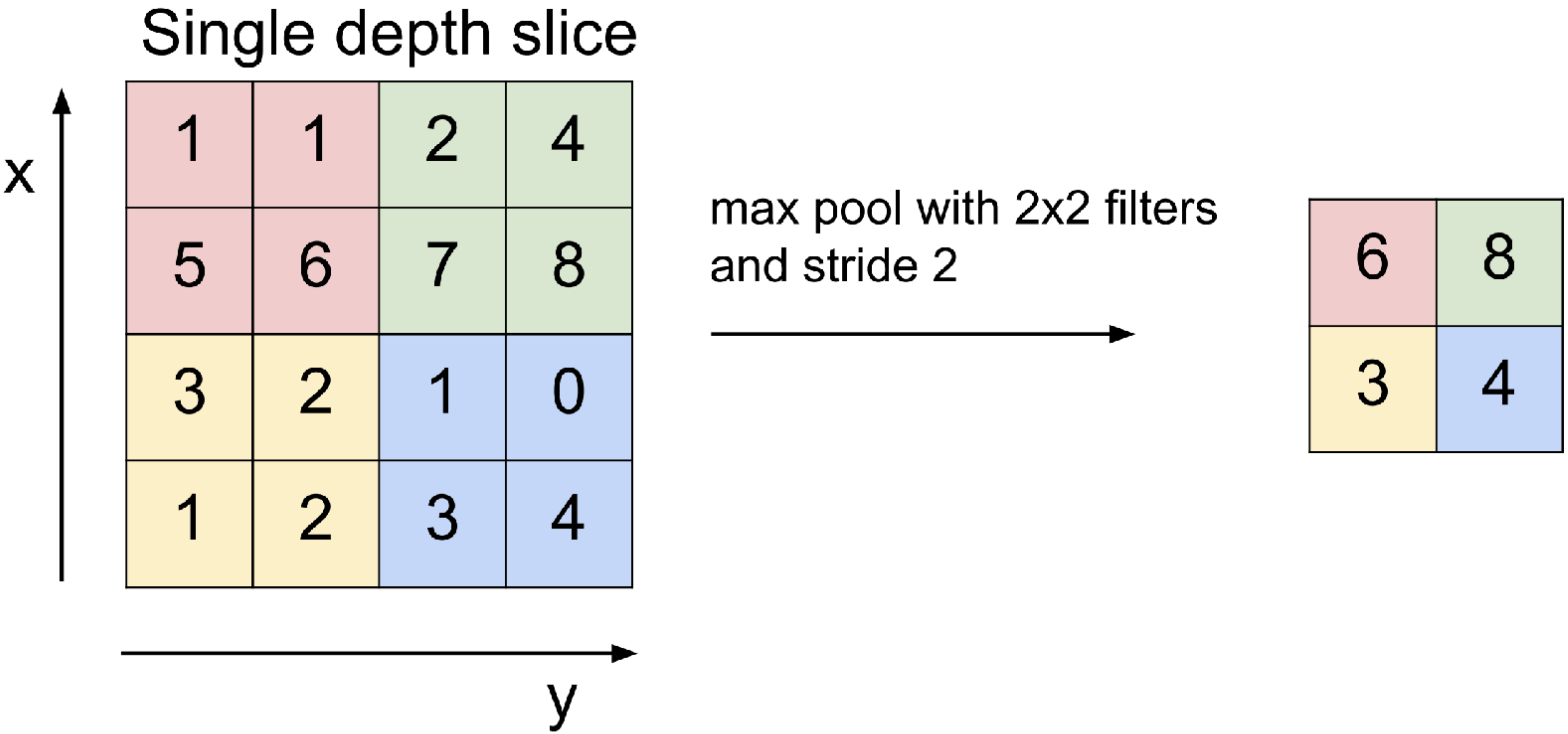
For more details + an animated demo see:
https://cs231n.github.io/convolutional-networks/

Pooling layer

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

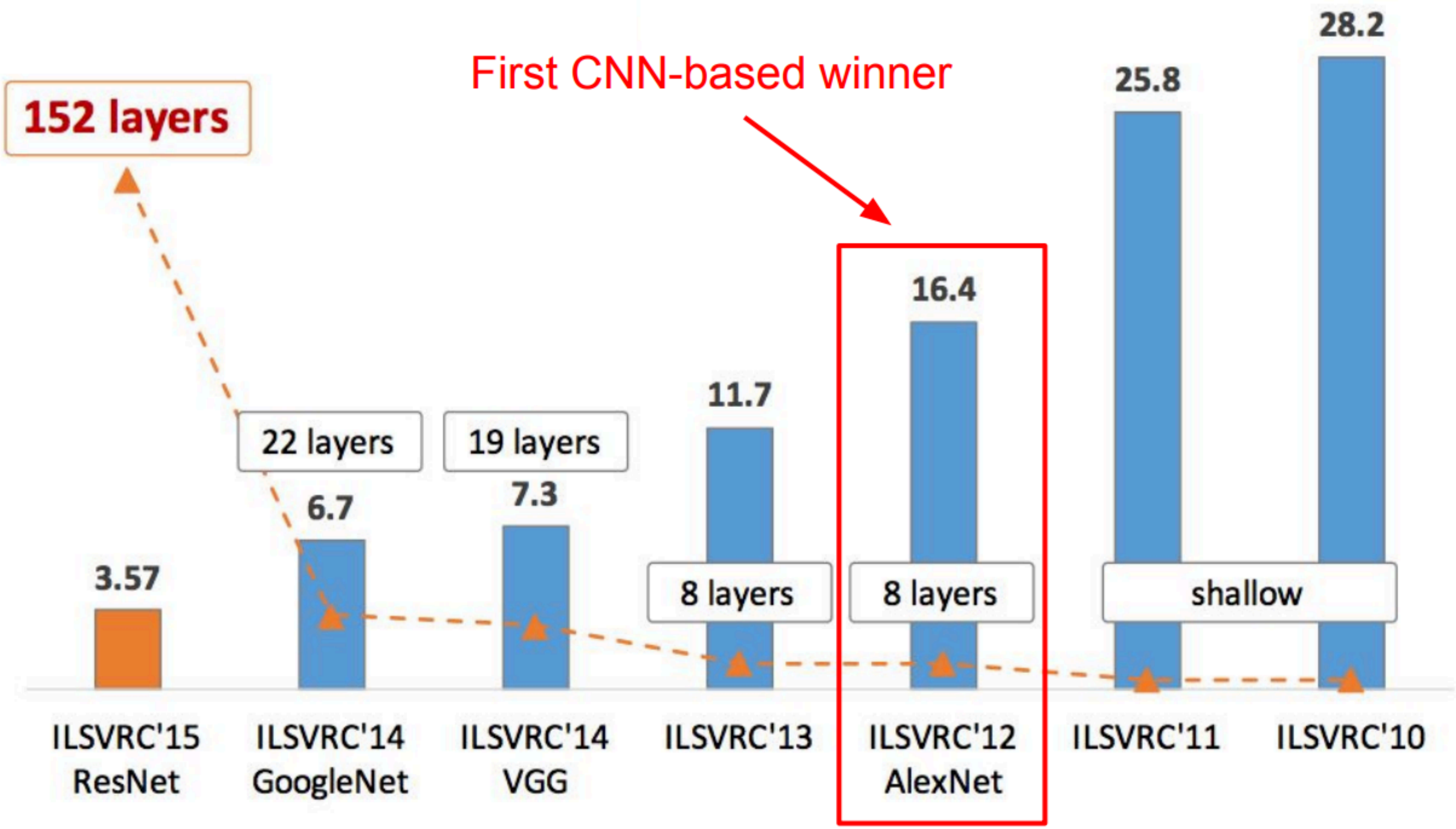


MAX POOLING


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

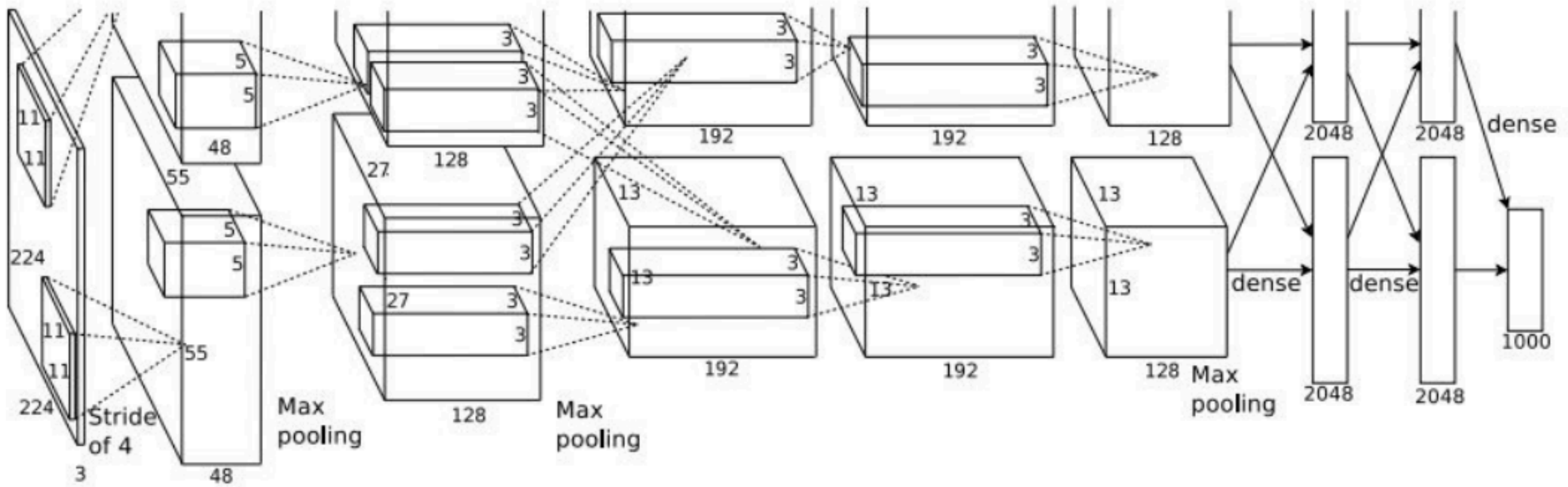

POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Case Study: AlexNet

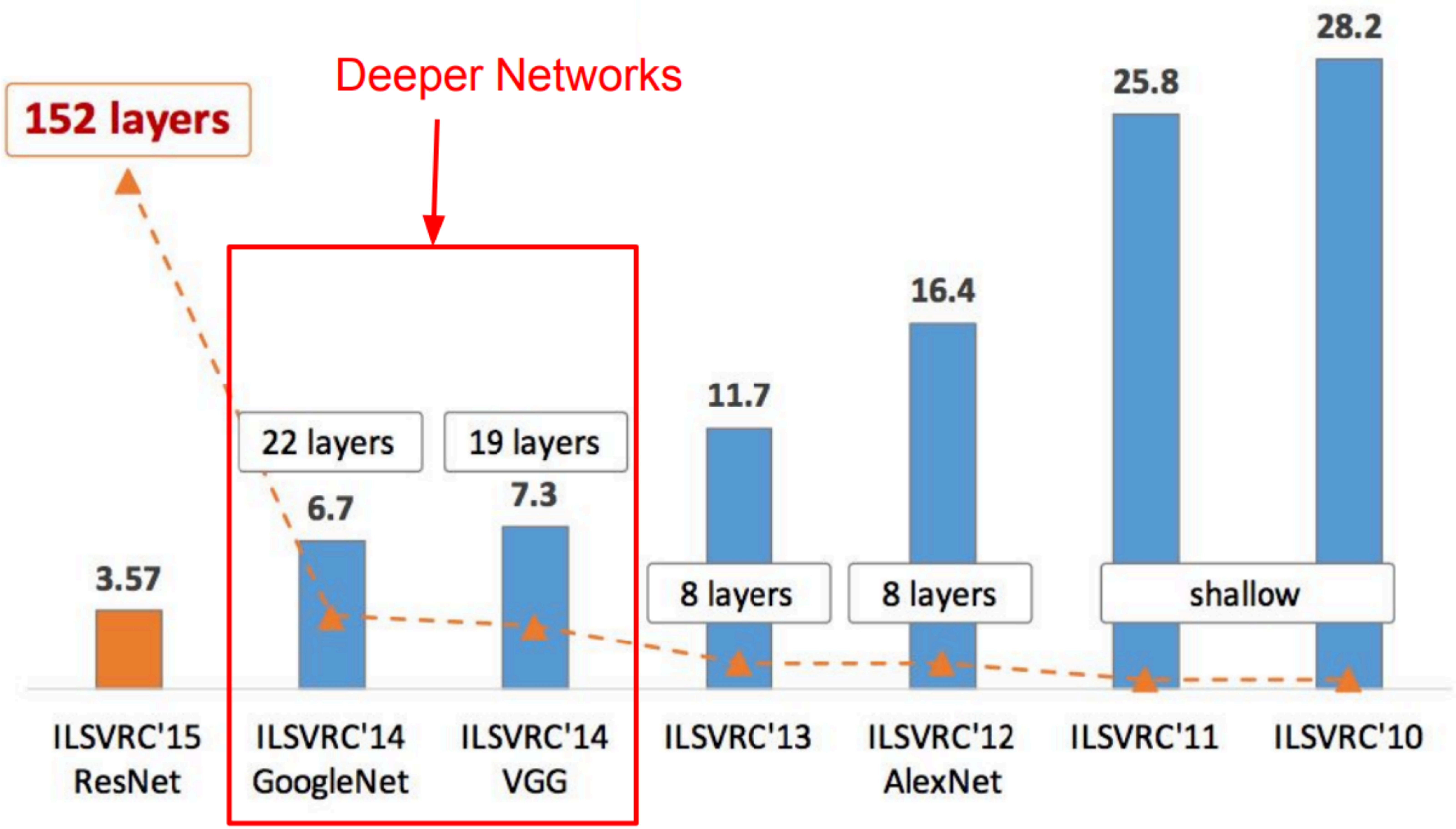
[Krizhevsky et al. 2012]



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

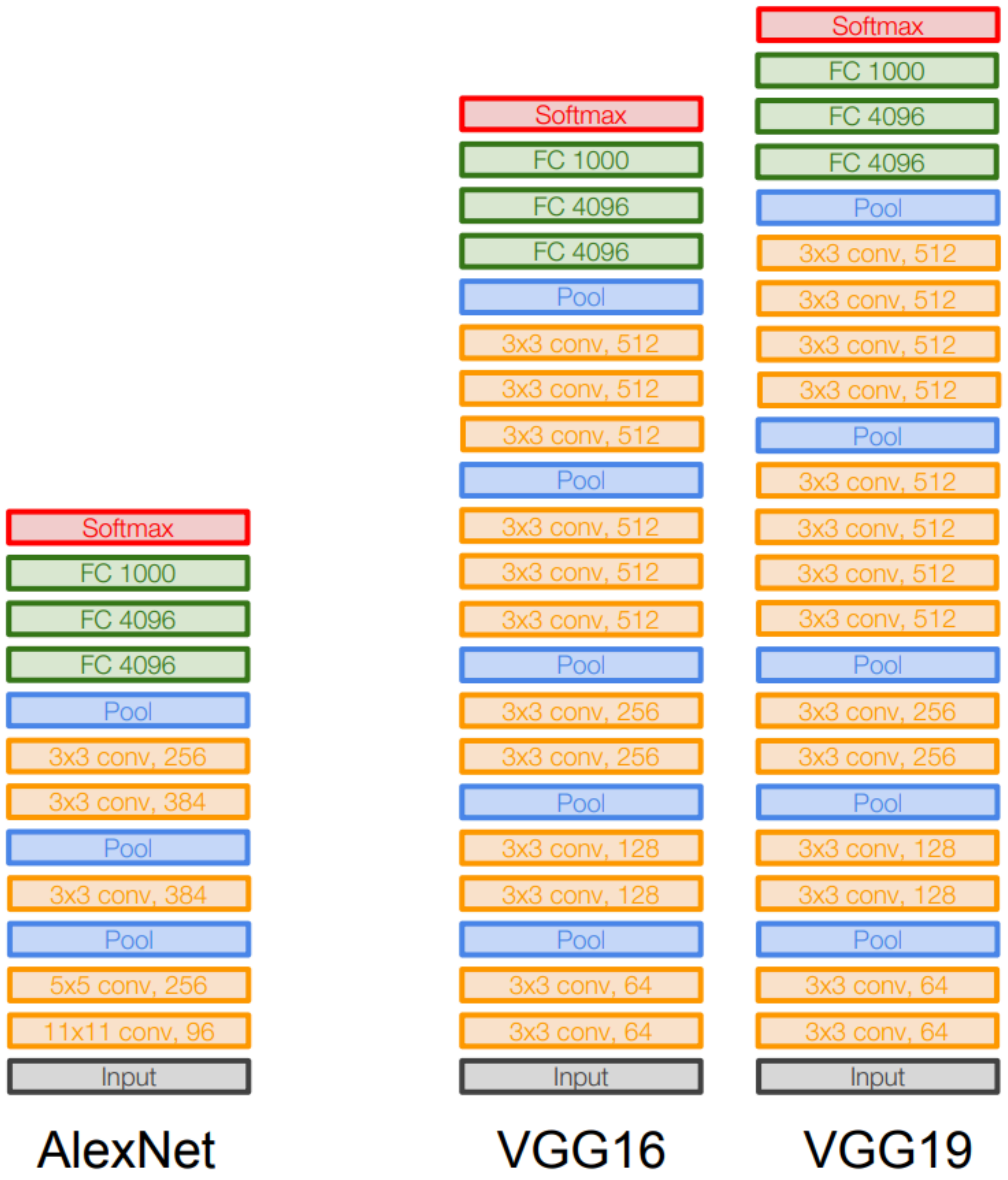
[Simonyan and Zisserman, 2014]

Small filters, Deeper networks

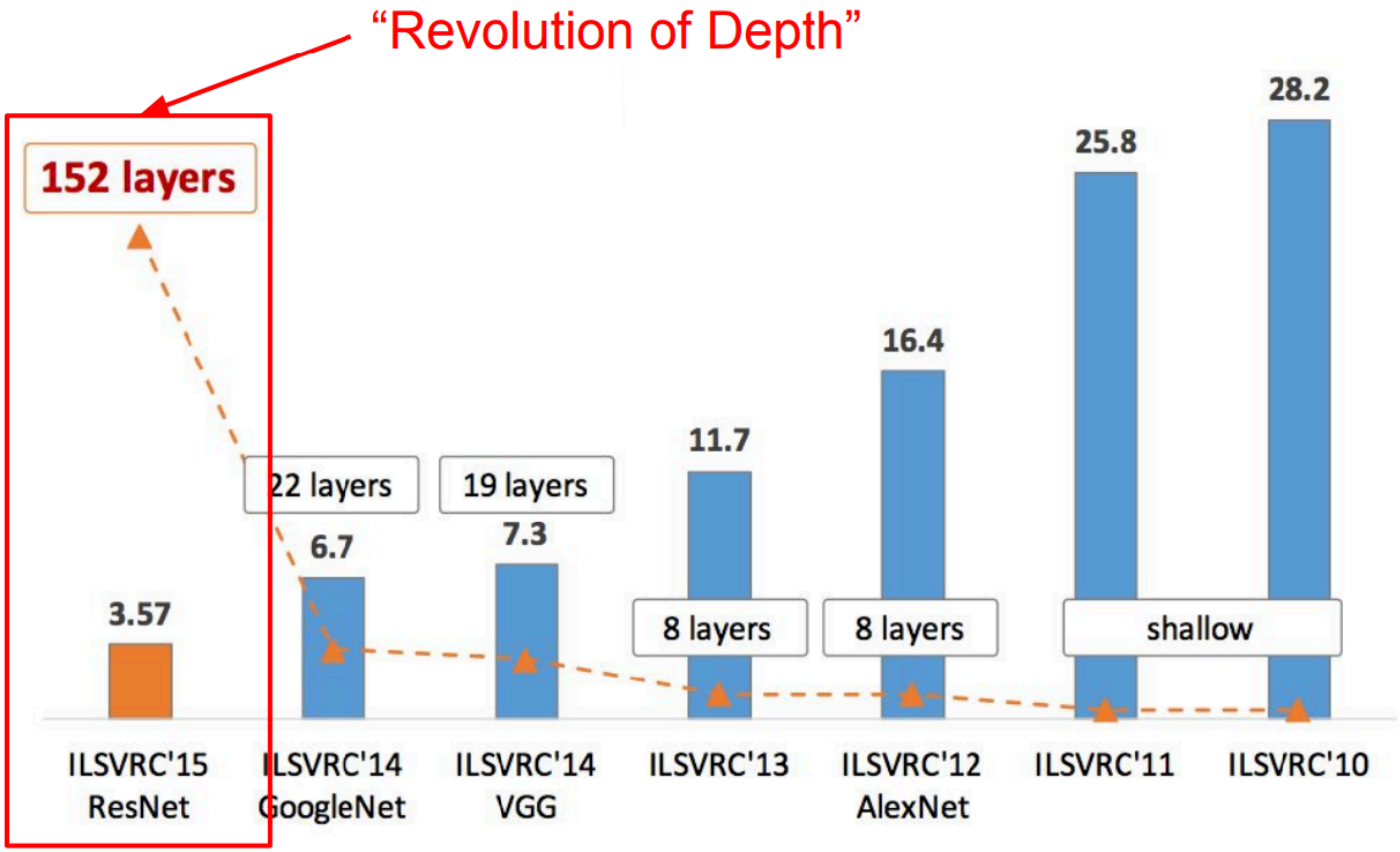
8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

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

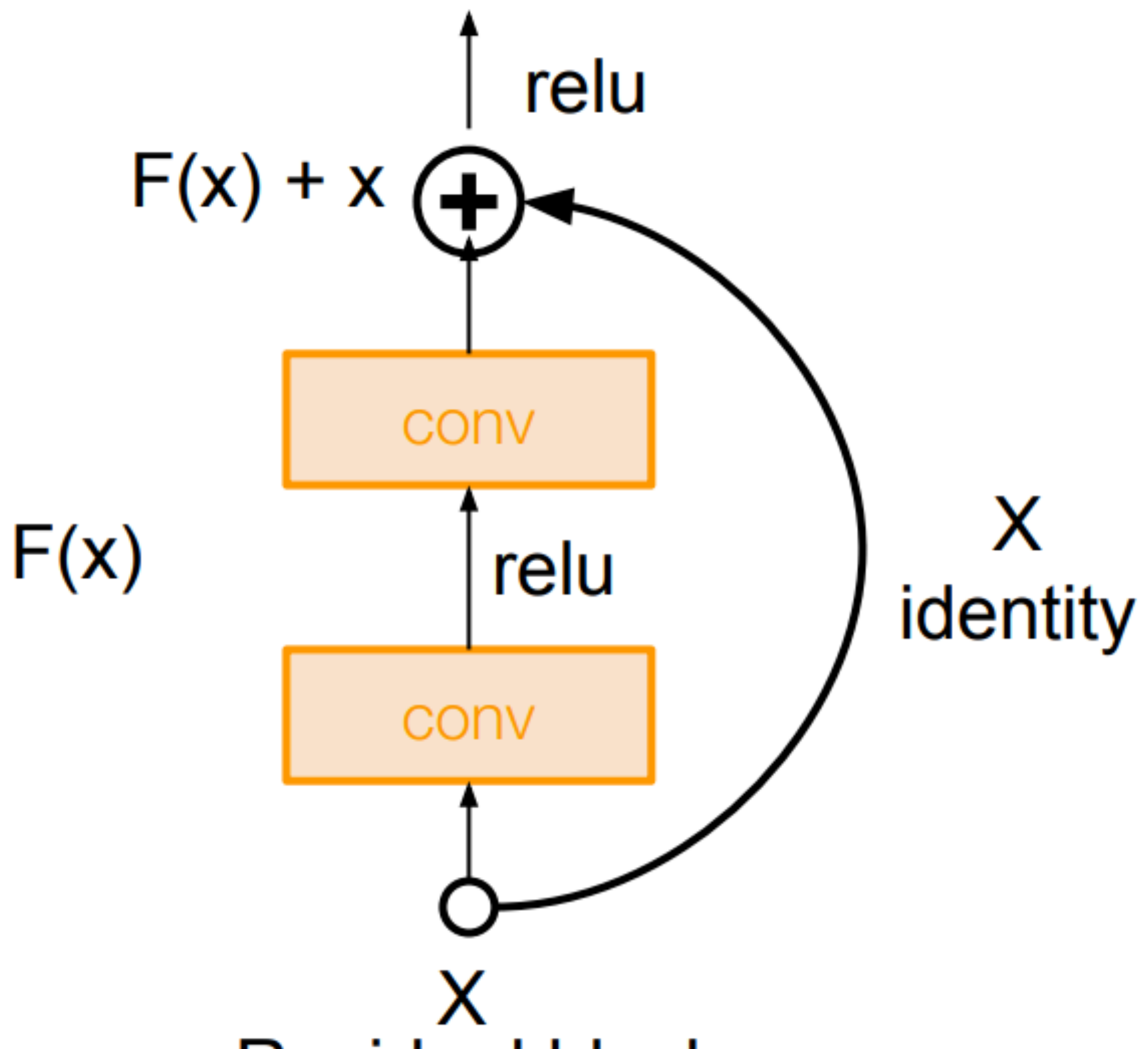


Case Study: ResNet

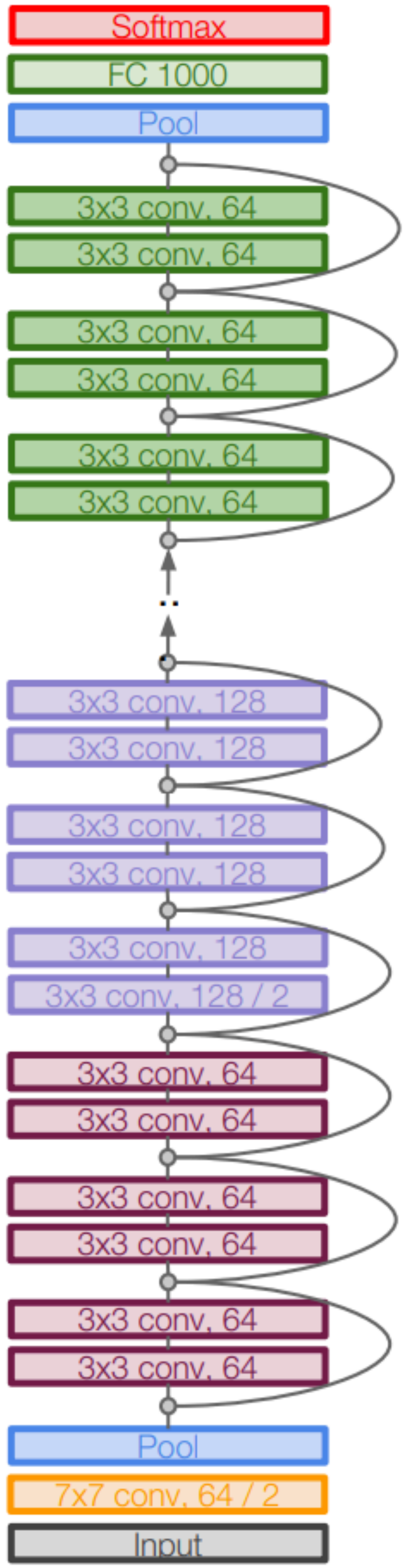
[He et al., 2015]

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!



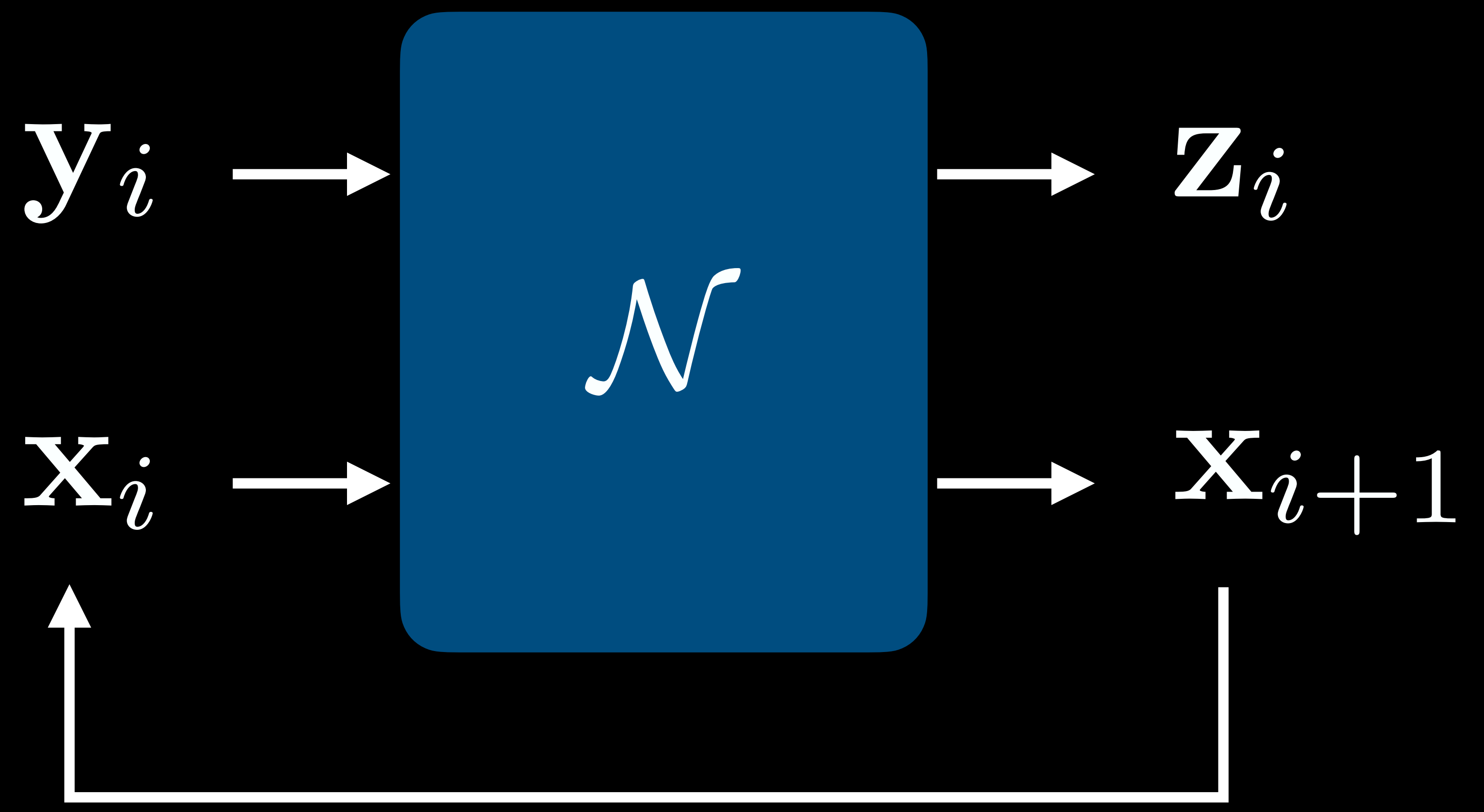
Residual block



Recurrent neural networks

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

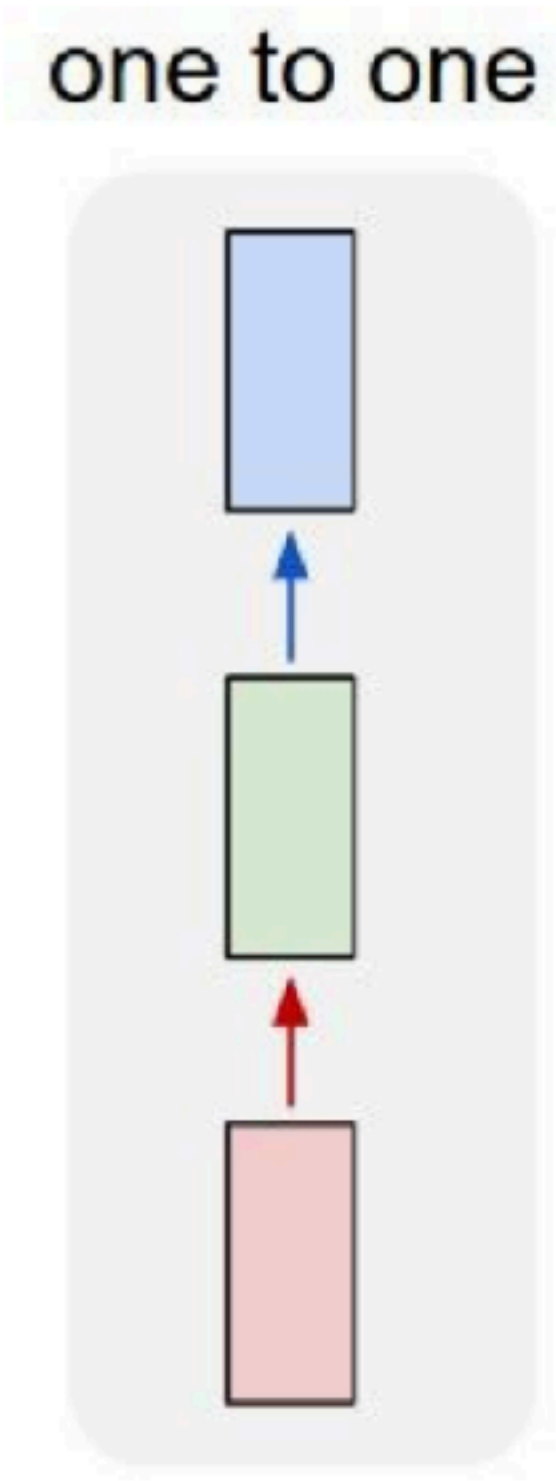
$$[\mathbf{x}_{i+1}, \mathbf{z}_i] = \mathcal{N}(\mathbf{x}_i, \mathbf{y}_i)$$



Feedback output to input

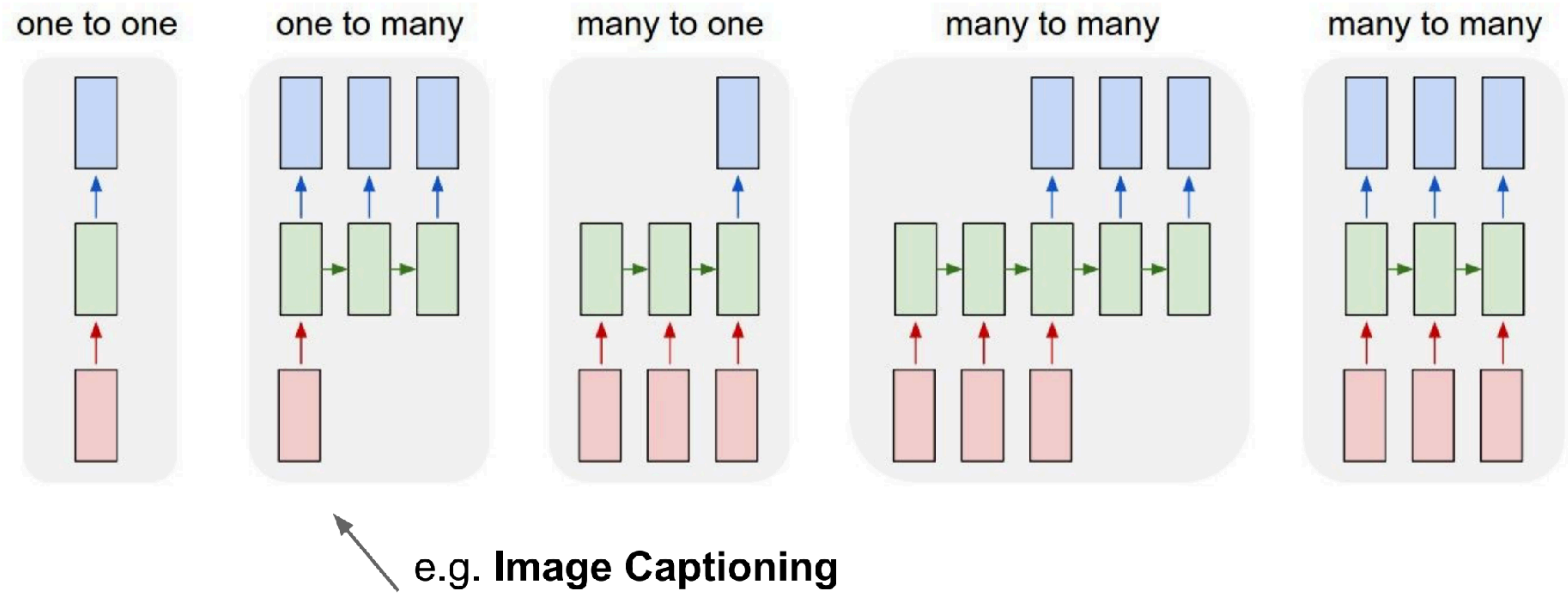
Numerical solvers are recurrence relations!

"Vanilla" Neural Network



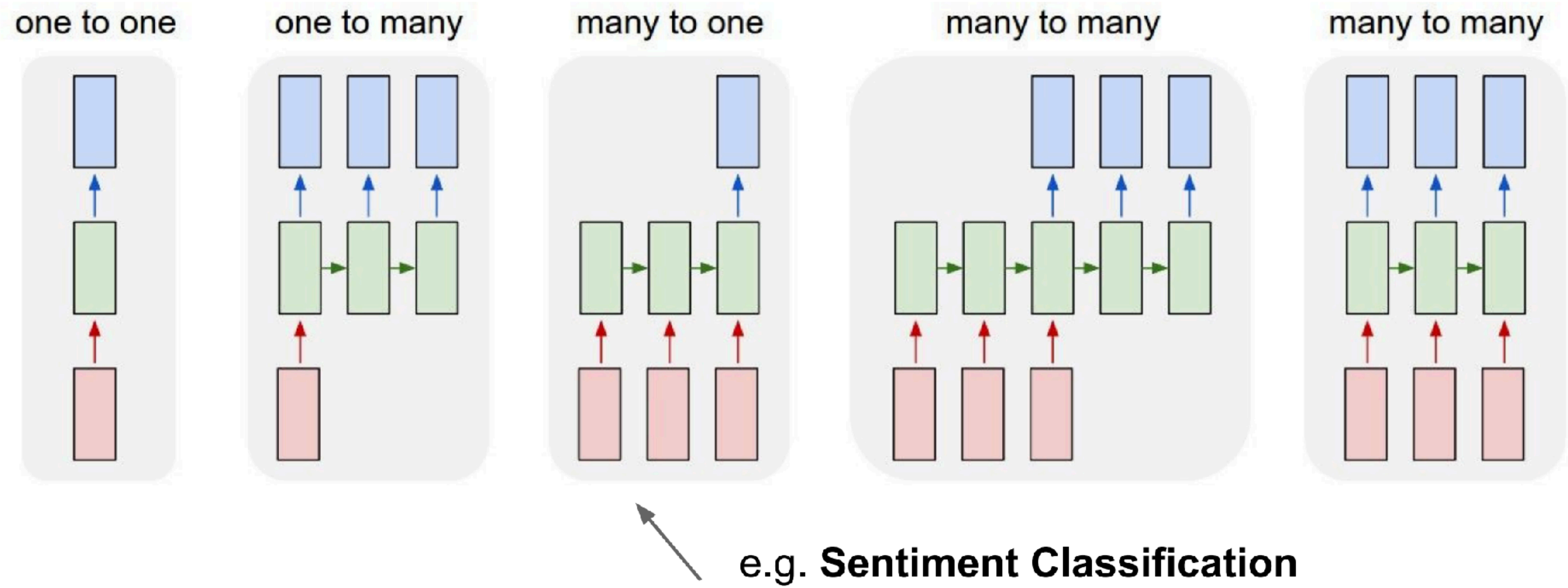
Vanilla Neural Networks

Recurrent Neural Networks: Process Sequences



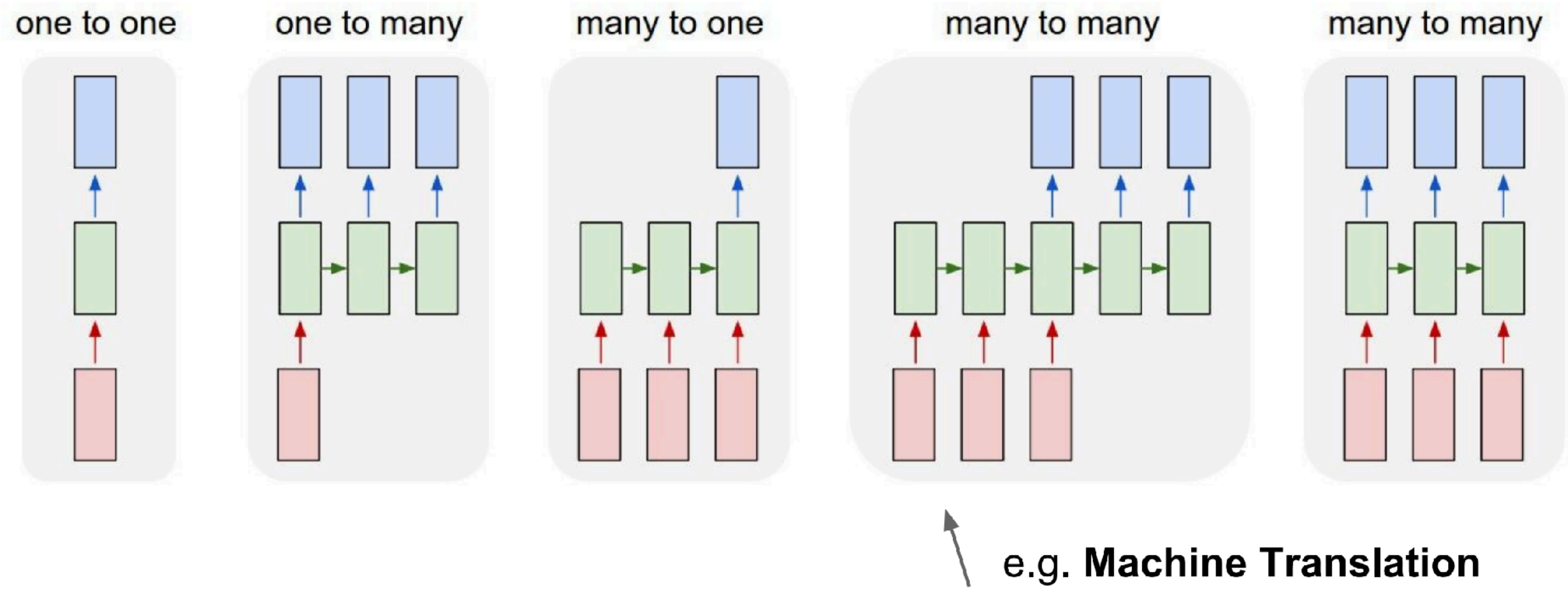
e.g. **Image Captioning**
image -> sequence of words

Recurrent Neural Networks: Process Sequences



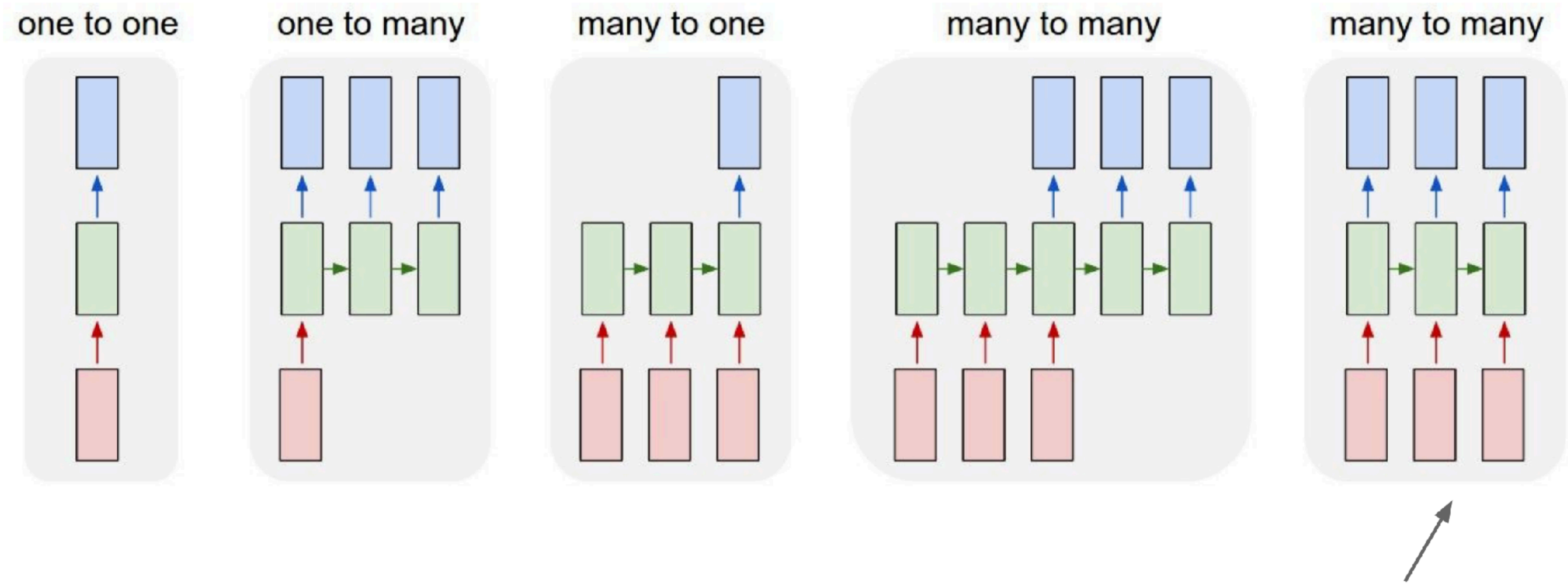
e.g. **Sentiment Classification**
sequence of words -> sentiment

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:

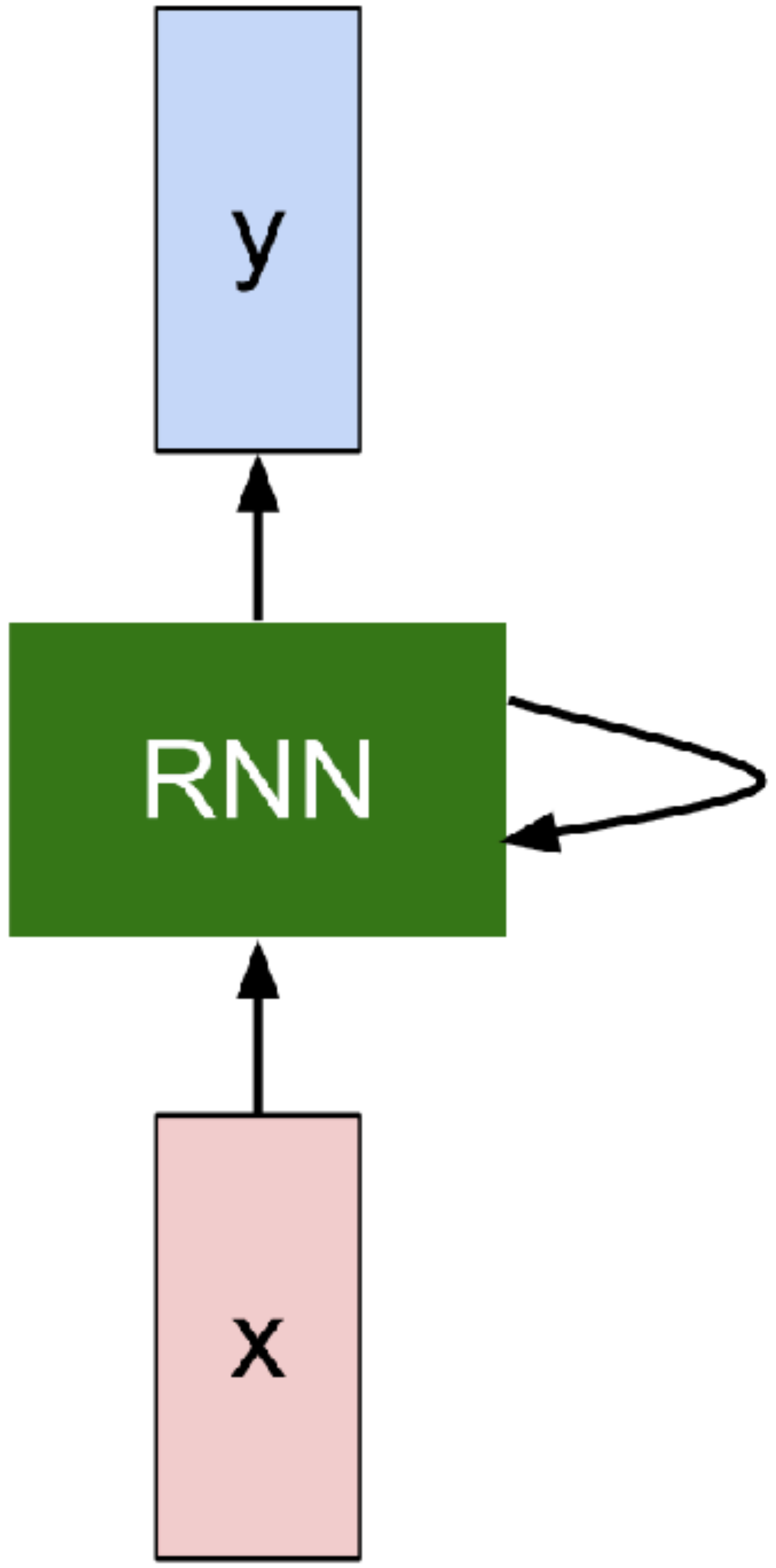
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

some function with parameters W

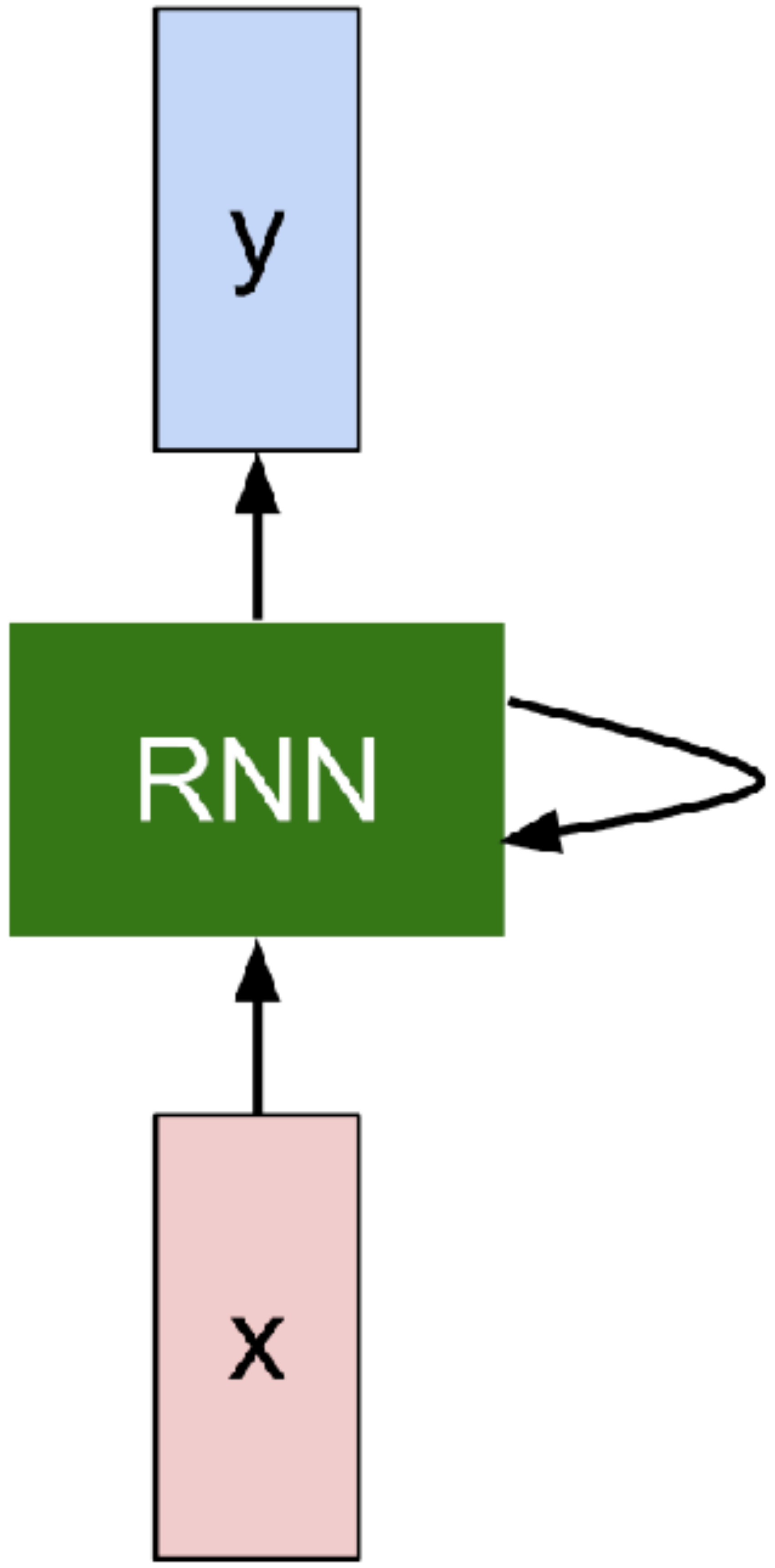
old state

input vector at some time step



(Vanilla) Recurrent Neural Network

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

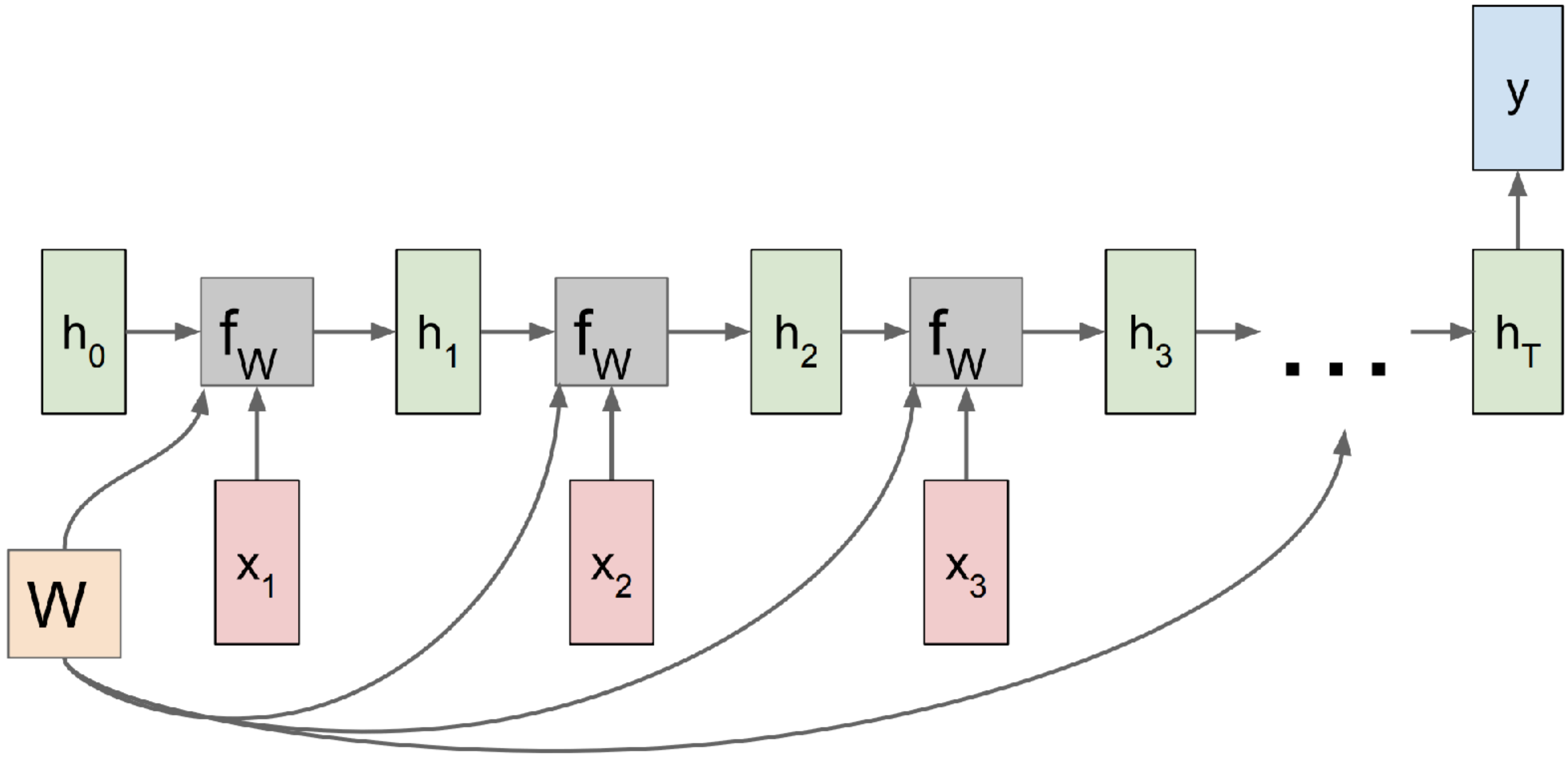


$$h_t = f_W(h_{t-1}, x_t)$$

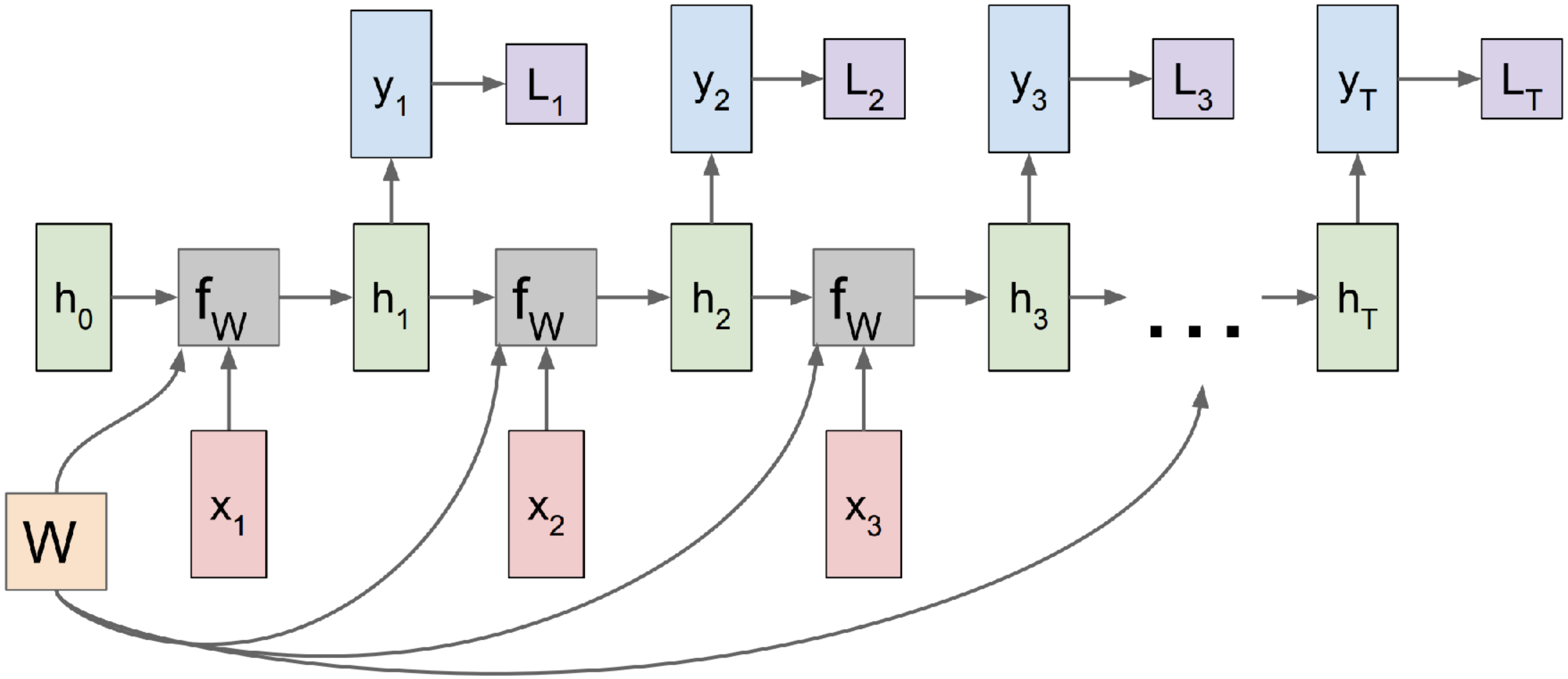
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

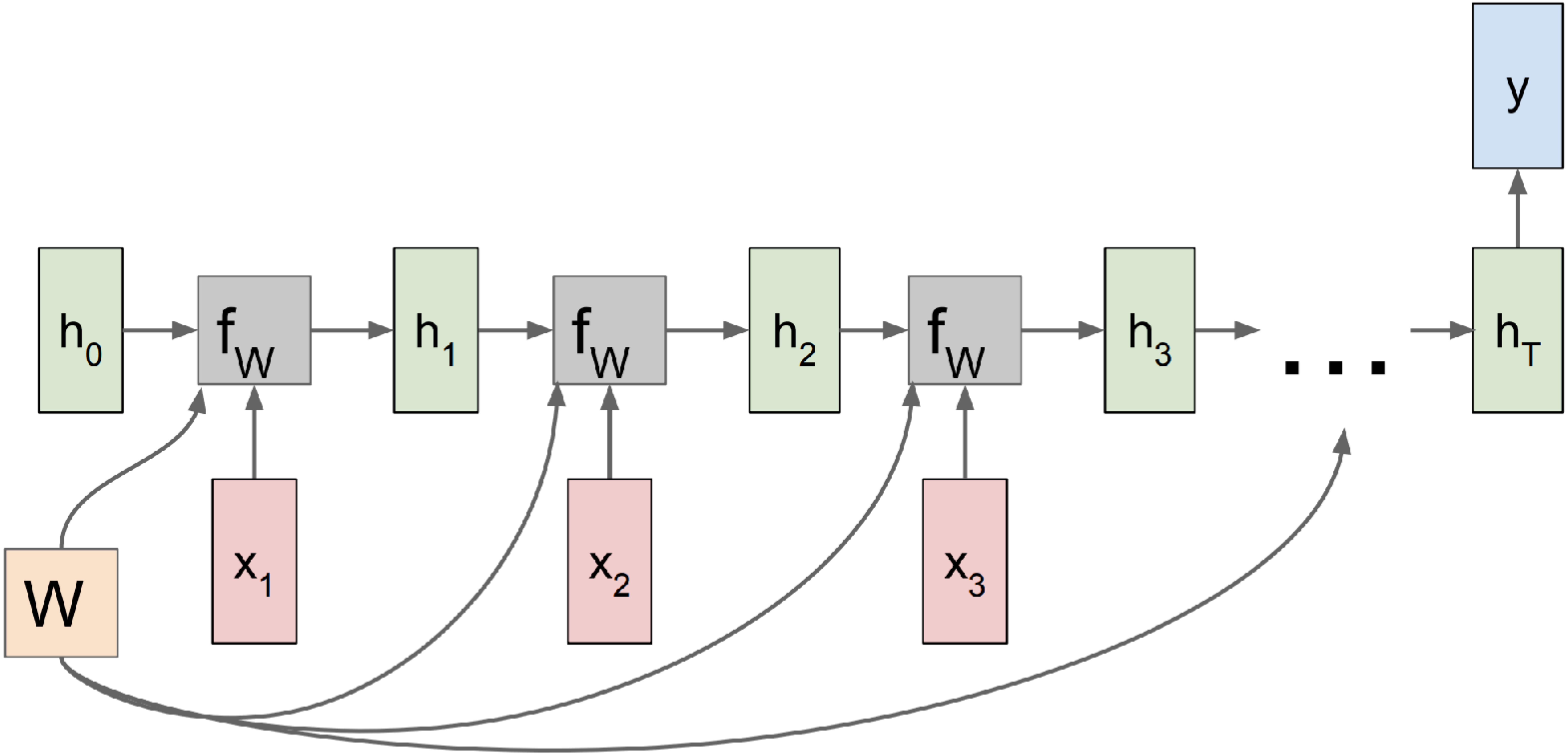
RNN: Computational Graph: Many to One



RNN: Computational Graph: Many to Many



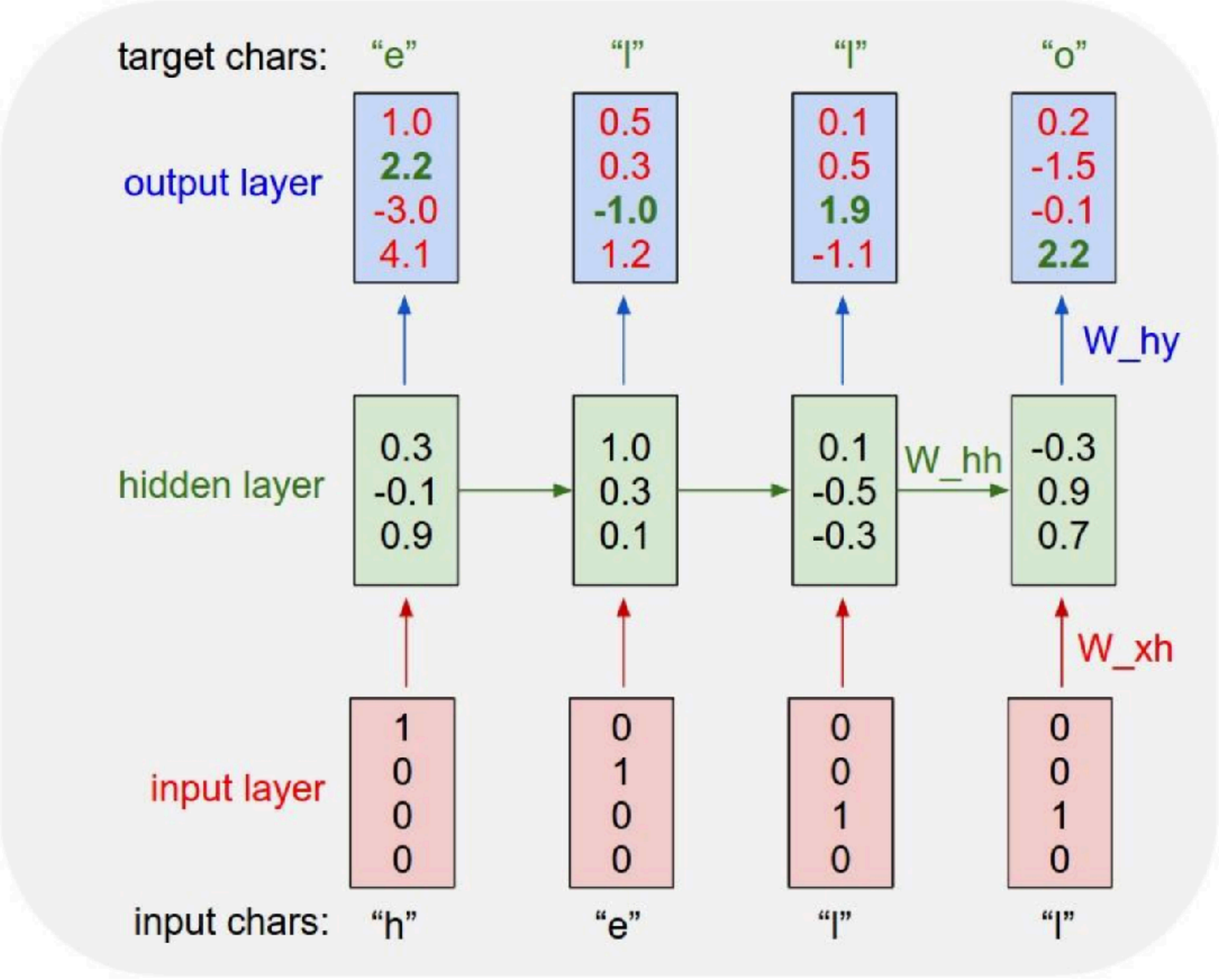
RNN: Computational Graph: Many to One



Example:
Character-level
Language Model

Vocabulary:
[h,e,l,o]

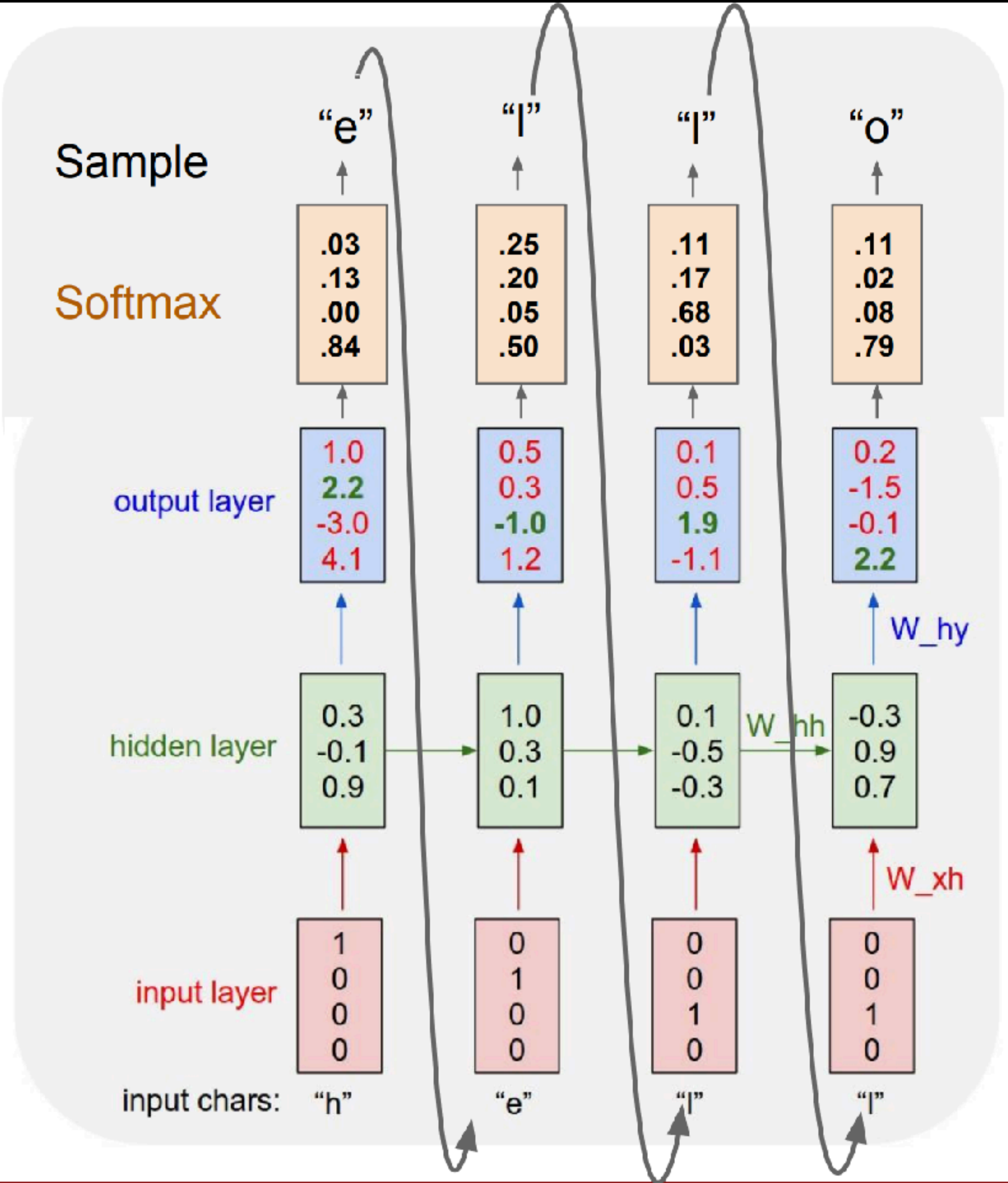
Example training sequence:
“hello”



Example: Character-level Language Model Sampling

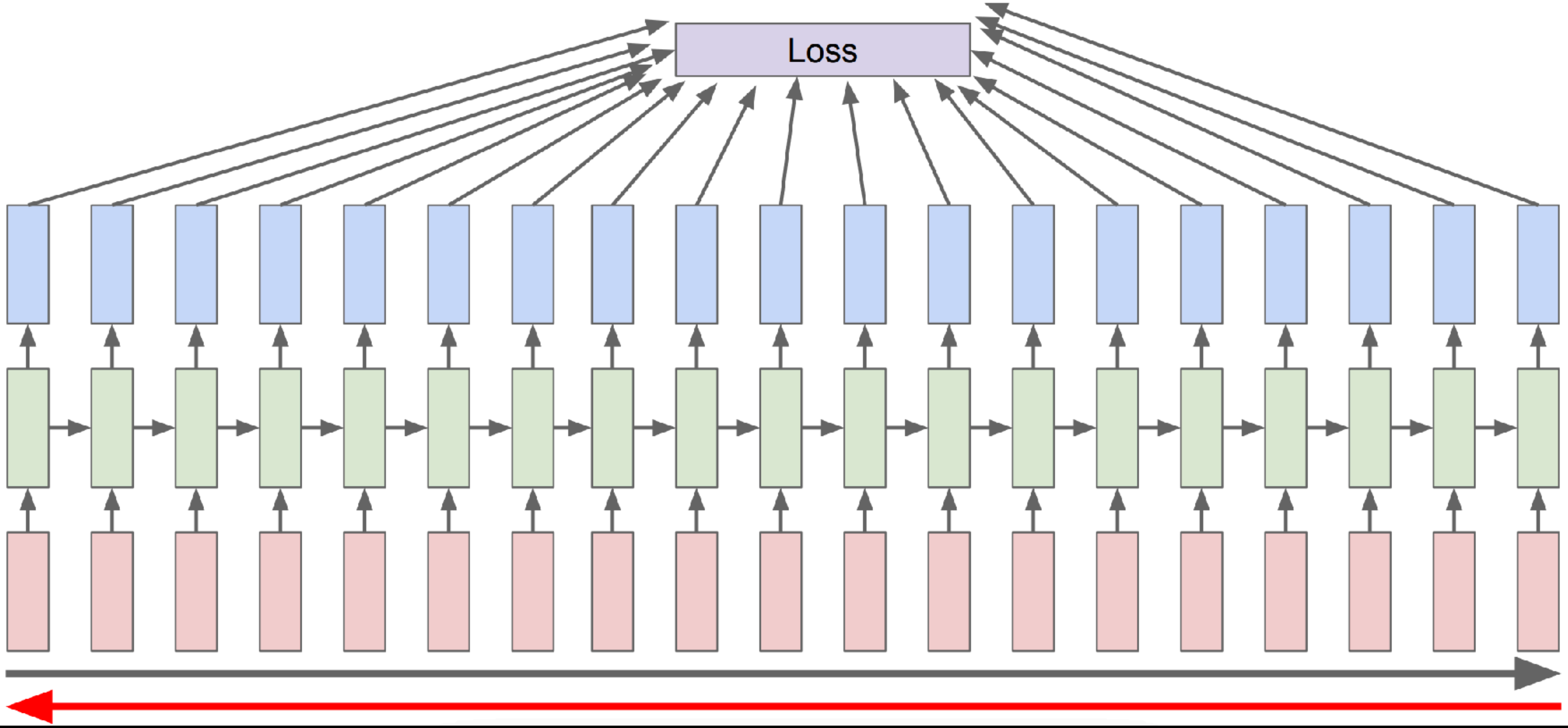
Vocabulary: [h,e,l,o]

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



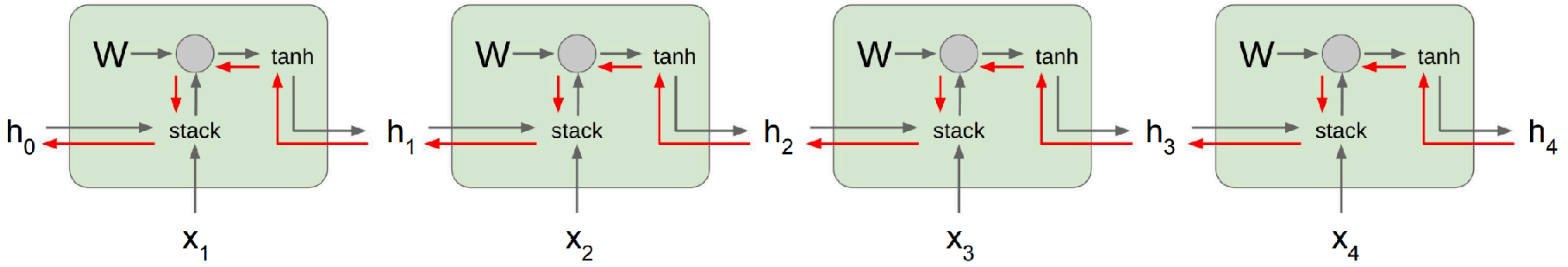
Backpropagation through time

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



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



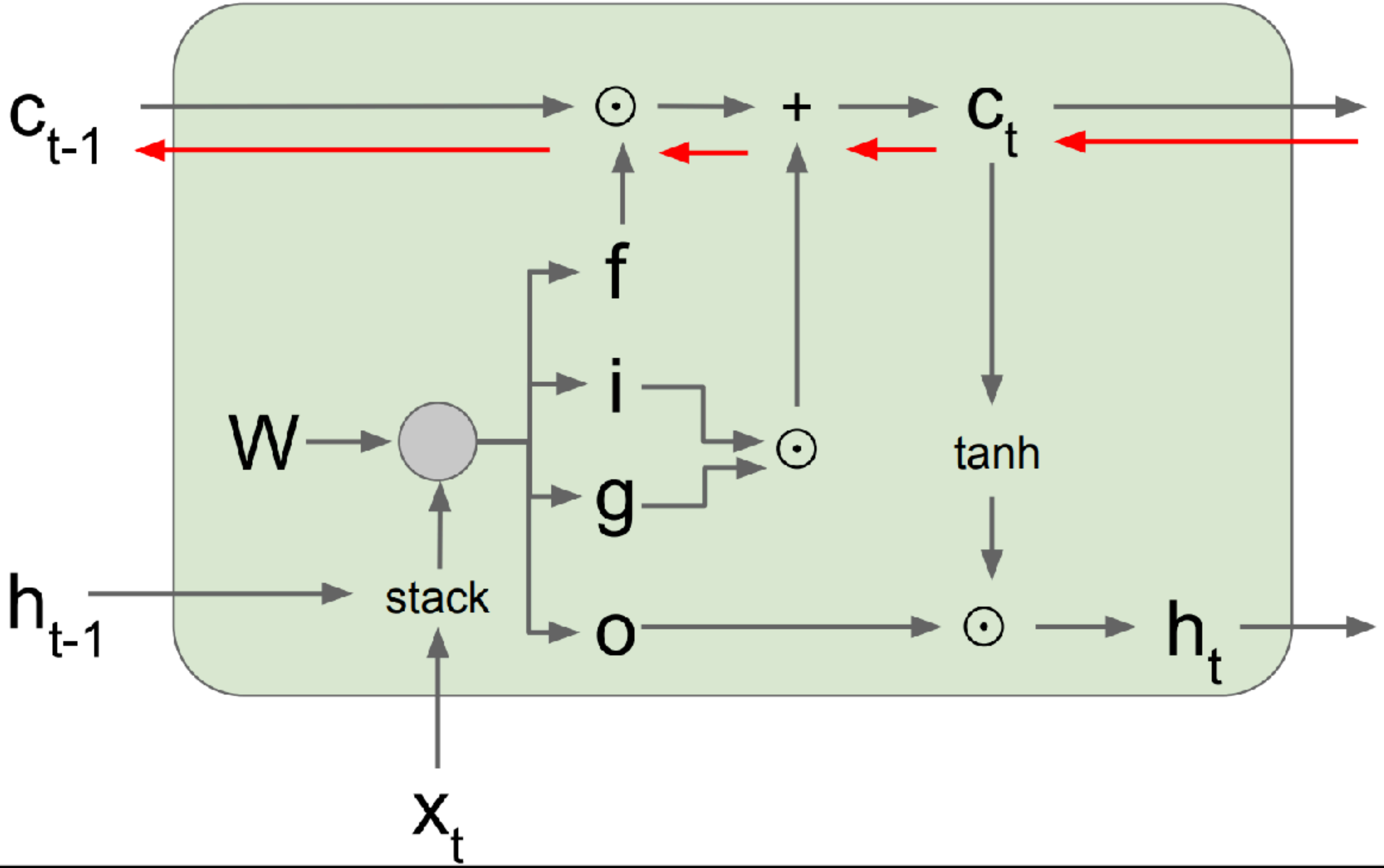
Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1:
Exploding gradients

Largest singular value < 1:
Vanishing gradients

Long Short Term Memory (LSTM): Gradient Flow

[Hochreiter et al., 1997]

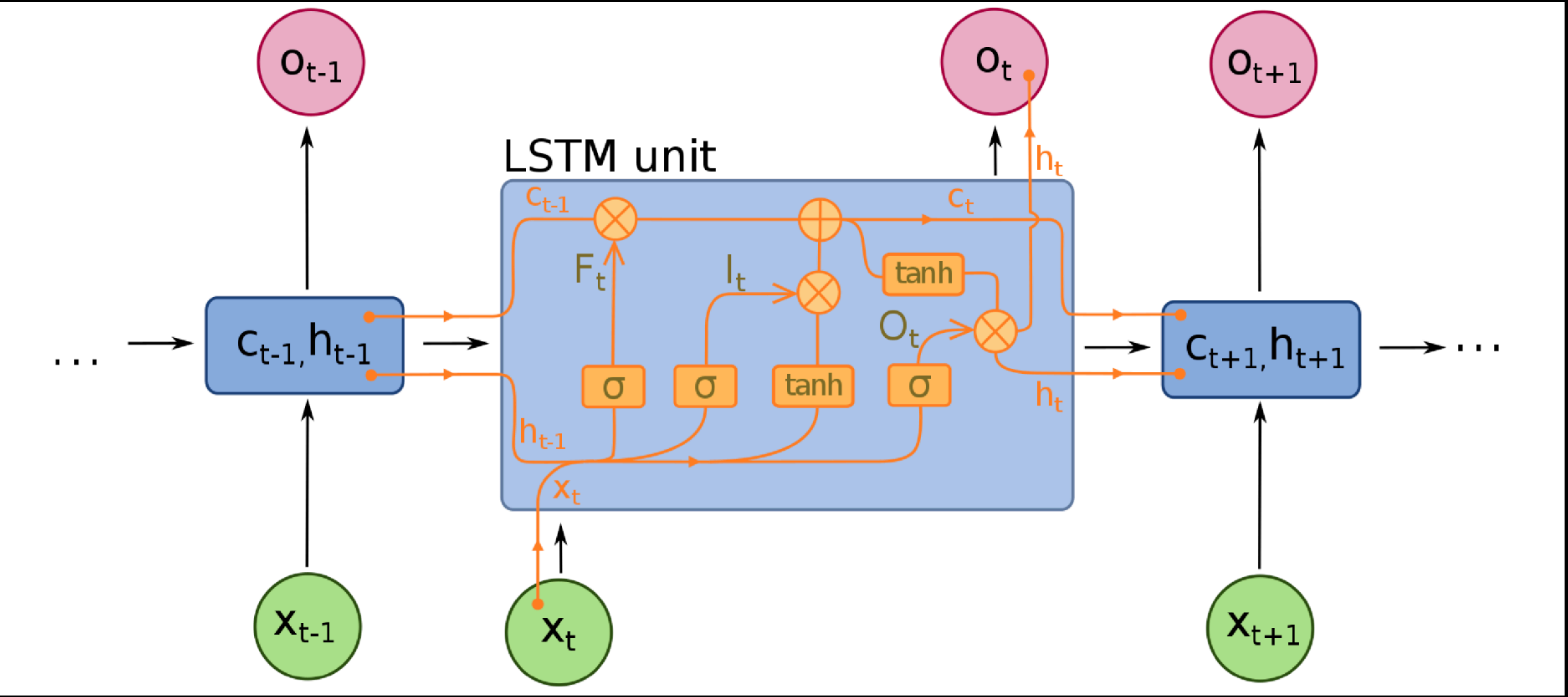


Backpropagation from c_t to c_{t-1} only elementwise multiplication by f, no matrix multiply by W

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

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

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



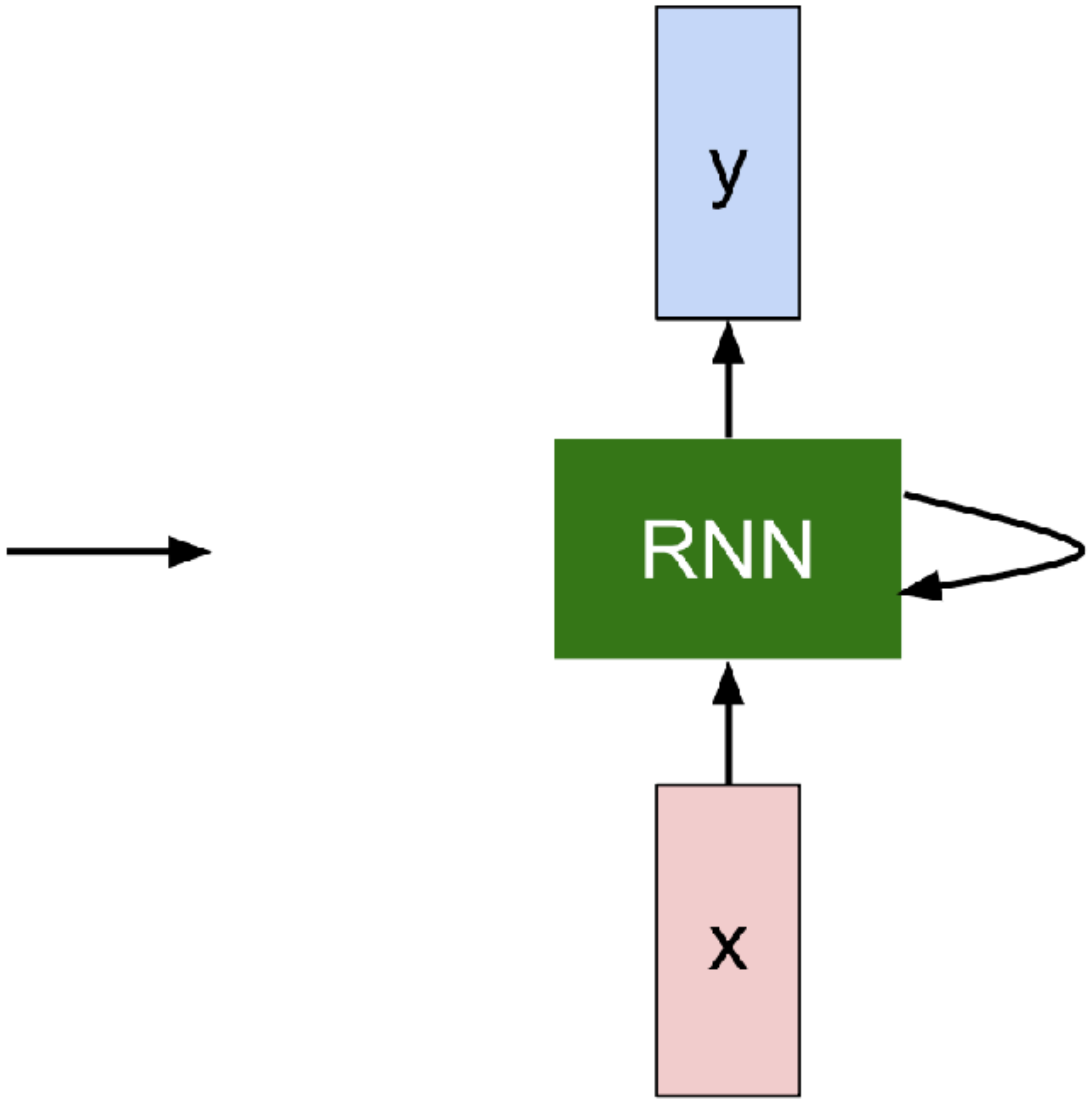
LSTM unit
o_{t-1}
o_t
o_{t+1}
c_{t-1}, h_{t-1}
c_{t+1}, h_{t+1}
x_{t-1}
x_t
x_{t+1}
c_{t-1}
c_t
h_t
h_{t-1}
F_t
I_t
O_t
σ
σ
tanh
σ
tanh

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 $\bigoplus_{n=1,\dots,m}$ 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

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

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 $\mathrm{Spec}(R') \to S$ 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 $\mathrm{GL}_{S'}(x'/S'')$ 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

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

is a unique morphism of algebraic stacks. Note that

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

and

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

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 $X_{spaces,étale}$ 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 $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ 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 $S = \mathrm{Spec}(R)$ and $Y = \mathrm{Spec}(R)$.*

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\dots,n} U_i$ 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 $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{X,\dots,0}$.

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

Derivative features

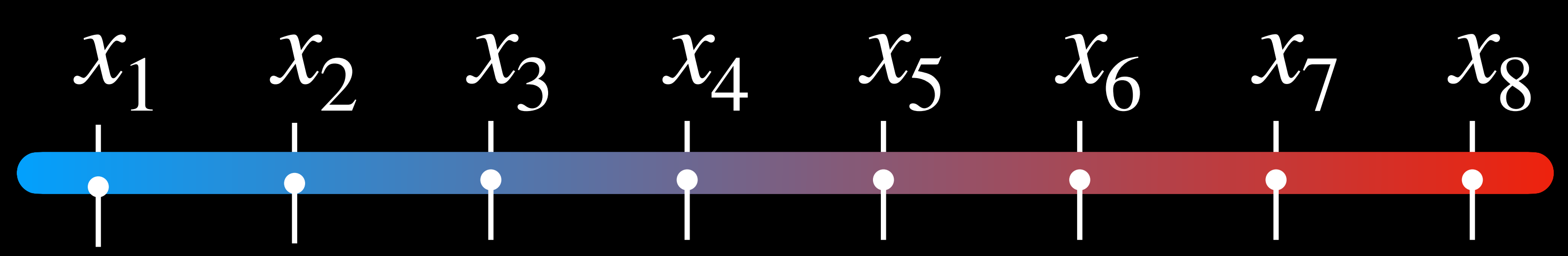
$$T_i^j = T(x_i, t_j)$$

$x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_8$

t_0	T_1^0	T_2^0	T_3^0	T_4^0	T_5^0	T_6^0	T_7^0	T_8^0
t_1	T_1^1	T_2^1	T_3^1	T_4^1	T_5^1	T_6^1	T_7^1	T_8^1

Derivative features

$$T_i^j = T(x_i, t_j)$$



$$\begin{array}{c|cccccccc}
t_0 & T_1^0 & T_2^0 & T_3^0 & T_4^0 & T_5^0 & T_6^0 & T_7^0 & T_8^0 \\
t_1 & T_1^1 & T_2^1 & T_3^1 & T_4^1 & T_5^1 & T_6^1 & T_7^1 & T_8^1 \\
t_2 & T_1^2 & T_2^2 & T_3^2 & T_4^2 & T_5^2 & T_6^2 & T_7^2 & T_8^2
\end{array}$$

Derivative features

$$T_i^j = T(x_i, t_j)$$

$$\begin{array}{ccccccccc}
 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\
t_0 & T_1^0 & T_2^0 & T_3^0 & T_4^0 & T_5^0 & T_6^0 & T_7^0 & T_8^0 \\
t_1 & T_1^1 & T_2^1 & T_3^1 & T_4^1 & T_5^1 & T_6^1 & T_7^1 & T_8^1 \\
t_2 & T_1^2 & T_2^2 & T_3^2 & T_4^2 & T_5^2 & T_6^2 & T_7^2 & T_8^2 \\
t_3 & T_1^3 & T_2^3 & T_3^3 & T_4^3 & T_5^3 & T_6^3 & T_7^3 & T_8^3 \\
 & & & & \vdots & & & & \\
t_n & T_1^n & T_2^n & T_3^n & T_4^n & T_5^n & T_6^n & T_7^n & T_8^n
\end{array}$$

Derivative features

$$T_i^j = T(x_i, t_j)$$

$$\begin{array}{c|cccccccc}
 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\
\hline
t_0 & T_1^0 & T_2^0 & T_3^0 & T_4^0 & T_5^0 & T_6^0 & T_7^0 & T_8^0 \\
t_1 & T_1^1 & T_2^1 & T_3^1 & T_4^1 & T_5^1 & T_6^1 & T_7^1 & T_8^1 \\
t_2 & T_1^2 & T_2^2 & T_3^2 & T_4^2 & T_5^2 & T_6^2 & T_7^2 & T_8^2 \\
t_3 & T_1^3 & T_2^3 & T_3^3 & T_4^3 & T_5^3 & T_6^3 & T_7^3 & T_8^3 \\
 & & & & \vdots & & & & \\
t_n & T_1^n & T_2^n & T_3^n & T_4^n & T_5^n & T_6^n & T_7^n & T_8^n
\end{array}$$

$$\xrightarrow{\phi}$$

$$\frac{T_i^{j+1} - T_i^j}{t_{j+1} - t_j} \qquad \frac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{x_{i+1} - 2x_i + x_{i-1}}$$

$$\left(\frac{\Delta T}{\Delta t}\right)_1 \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_1$$

Derivative features

$$T_i^j = T(x_i, t_j)$$

$$\begin{array}{c|cccccccc}
 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\
\hline
t_0 & T_1^0 & T_2^0 & T_3^0 & T_4^0 & T_5^0 & T_6^0 & T_7^0 & T_8^0 \\
t_1 & T_1^1 & T_2^1 & T_3^1 & T_4^1 & T_5^1 & T_6^1 & T_7^1 & T_8^1 \\
t_2 & T_1^2 & T_2^2 & T_3^2 & T_4^2 & T_5^2 & T_6^2 & T_7^2 & T_8^2 \\
t_3 & T_1^3 & T_2^3 & T_3^3 & T_4^3 & T_5^3 & T_6^3 & T_7^3 & T_8^3 \\
 & & & & \vdots & & & & \\
t_n & T_1^n & T_2^n & T_3^n & T_4^n & T_5^n & T_6^n & T_7^n & T_8^n
\end{array}$$

$$\xrightarrow{\phi}$$

$$\frac{T_i^{j+1} - T_i^j}{t_{j+1} - t_j} \qquad \frac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{x_{i+1} - 2x_i + x_{i-1}}$$

$$\left(\frac{\Delta T}{\Delta t}\right)_1 \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_1$$

$$\left(\frac{\Delta T}{\Delta t}\right)_2 \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_2$$

Derivative features

$$T_i^j = T(x_i, t_j)$$

$$\begin{array}{c|cccccccc}
 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\
\hline
t_0 & T_1^0 & T_2^0 & T_3^0 & T_4^0 & T_5^0 & T_6^0 & T_7^0 & T_8^0 \\
t_1 & T_1^1 & T_2^1 & T_3^1 & T_4^1 & T_5^1 & T_6^1 & T_7^1 & T_8^1 \\
t_2 & T_1^2 & T_2^2 & T_3^2 & T_4^2 & T_5^2 & T_6^2 & T_7^2 & T_8^2 \\
t_3 & T_1^3 & T_2^3 & T_3^3 & T_4^3 & T_5^3 & T_6^3 & T_7^3 & T_8^3 \\
 & & & & \vdots & & & & \\
t_n & T_1^n & T_2^n & T_3^n & T_4^n & T_5^n & T_6^n & T_7^n & T_8^n
\end{array}$$

$$\xrightarrow{\phi}$$

$$\frac{T_i^{j+1} - T_i^j}{t_{j+1} - t_j} \qquad \frac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{x_{i+1} - 2x_i + x_{i-1}}$$

$$\left(\frac{\Delta T}{\Delta t}\right)_1 \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_1$$

$$\left(\frac{\Delta T}{\Delta t}\right)_2 \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_2$$

$$\vdots \qquad \vdots$$

Derivative features

$$T_i^j = T(x_i, t_j)$$

$$
\begin{array}{c|cccccccc}
 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\
\hline
t_0 & T_1^0 & T_2^0 & T_3^0 & T_4^0 & T_5^0 & T_6^0 & T_7^0 & T_8^0 \\
t_1 & T_1^1 & T_2^1 & T_3^1 & T_4^1 & T_5^1 & T_6^1 & T_7^1 & T_8^1 \\
t_2 & T_1^2 & T_2^2 & T_3^2 & T_4^2 & T_5^2 & T_6^2 & T_7^2 & T_8^2 \\
t_3 & T_1^3 & T_2^3 & T_3^3 & T_4^3 & T_5^3 & T_6^3 & T_7^3 & T_8^3 \\
 & & & & \vdots & & & & \\
t_n & T_1^n & T_2^n & T_3^n & T_4^n & T_5^n & T_6^n & T_7^n & T_8^n
\end{array}
\xrightarrow{\ \phi\ }
\begin{array}{cc}
\dfrac{T_i^{j+1} - T_i^j}{t_{j+1} - t_j} & \dfrac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{x_{i+1} - 2x_i + x_{i-1}} \\
\left(\dfrac{\Delta T}{\Delta t}\right)_1 & \left(\dfrac{\Delta^2 T}{\Delta x^2}\right)_1 \\
\left(\dfrac{\Delta T}{\Delta t}\right)_2 & \left(\dfrac{\Delta^2 T}{\Delta x^2}\right)_2 \\
\vdots & \vdots
\end{array}
$$

Derivative features

$$T_i^j = T(x_i, t_j)$$

$$\begin{array}{c|cccccccc}
 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\
\hline
t_0 & T_1^0 & T_2^0 & T_3^0 & T_4^0 & T_5^0 & T_6^0 & T_7^0 & T_8^0 \\
t_1 & T_1^1 & T_2^1 & T_3^1 & T_4^1 & T_5^1 & T_6^1 & T_7^1 & T_8^1 \\
t_2 & T_1^2 & T_2^2 & T_3^2 & T_4^2 & T_5^2 & T_6^2 & T_7^2 & T_8^2 \\
t_3 & T_1^3 & T_2^3 & T_3^3 & T_4^3 & T_5^3 & T_6^3 & T_7^3 & T_8^3 \\
 & & & & \vdots & & & & \\
t_n & T_1^n & T_2^n & T_3^n & T_4^n & T_5^n & T_6^n & T_7^n & T_8^n
\end{array}$$

$$\xrightarrow{\phi}$$

$$\frac{T_i^{j+1} - T_i^j}{t_{j+1} - t_j} \qquad \frac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{x_{i+1} - 2x_i + x_{i-1}}$$

$$\left(\frac{\Delta T}{\Delta t}\right)_1 \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_1$$

$$\left(\frac{\Delta T}{\Delta t}\right)_2 \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_2$$

$$\vdots \qquad \vdots$$

Derivative features

$$T_i^j = T(x_i, t_j)$$

$$\begin{array}{c|cccccccc}
 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\
\hline
t_0 & T_1^0 & T_2^0 & T_3^0 & T_4^0 & T_5^0 & T_6^0 & T_7^0 & T_8^0 \\
t_1 & T_1^1 & T_2^1 & T_3^1 & T_4^1 & T_5^1 & T_6^1 & T_7^1 & T_8^1 \\
t_2 & T_1^2 & T_2^2 & T_3^2 & T_4^2 & T_5^2 & T_6^2 & T_7^2 & T_8^2 \\
t_3 & T_1^3 & T_2^3 & T_3^3 & T_4^3 & T_5^3 & T_6^3 & T_7^3 & T_8^3 \\
 & & & & \vdots & & & & \\
t_n & T_1^n & T_2^n & T_3^n & T_4^n & T_5^n & T_6^n & T_7^n & T_8^n
\end{array}
\quad \xrightarrow{\phi} \quad
\begin{array}{cc}
\dfrac{T_i^{j+1} - T_i^j}{t_{j+1} - t_j} & \dfrac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{x_{i+1} - 2x_i + x_{i-1}} \\
\left(\dfrac{\Delta T}{\Delta t}\right)_1 & \left(\dfrac{\Delta^2 T}{\Delta x^2}\right)_1 \\
\left(\dfrac{\Delta T}{\Delta t}\right)_2 & \left(\dfrac{\Delta^2 T}{\Delta x^2}\right)_2 \\
\vdots & \vdots \\
\left(\dfrac{\Delta T}{\Delta t}\right)_n & \left(\dfrac{\Delta^2 T}{\Delta x^2}\right)_n
\end{array}$$

Derivative features

$$T_i^j = T(x_i, t_j)$$

$$x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_8$$

$$
\begin{array}{c|cccccccc}
t_0 & T_1^0 & T_2^0 & T_3^0 & T_4^0 & T_5^0 & T_6^0 & T_7^0 & T_8^0 \\
t_1 & T_1^1 & T_2^1 & T_3^1 & T_4^1 & T_5^1 & T_6^1 & T_7^1 & T_8^1 \\
t_2 & T_1^2 & T_2^2 & T_3^2 & T_4^2 & T_5^2 & T_6^2 & T_7^2 & T_8^2 \\
t_3 & T_1^3 & T_2^3 & T_3^3 & T_4^3 & T_5^3 & T_6^3 & T_7^3 & T_8^3 \\
 & & & & \vdots & & & & \\
t_n & T_1^n & T_2^n & T_3^n & T_4^n & T_5^n & T_6^n & T_7^n & T_8^n
\end{array}
$$

$$\xrightarrow{\phi}$$

$$\frac{T_i^{j+1} - T_i^j}{t_{j+1} - t_j} \qquad \frac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{x_{i+1} - 2x_i + x_{i-1}}$$

$$\left(\frac{\Delta T}{\Delta t}\right)_1 \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_1$$

$$\left(\frac{\Delta T}{\Delta t}\right)_2 \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_2$$

$$\vdots \qquad \vdots$$

$$\left(\frac{\Delta T}{\Delta t}\right)_n \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_n$$

Derivative features

$$T_i^j = T(x_i, t_j)$$

$$\begin{array}{c|cccccccc}
 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\
\hline
t_0 & T_1^0 & T_2^0 & T_3^0 & T_4^0 & T_5^0 & T_6^0 & T_7^0 & T_8^0 \\
t_1 & T_1^1 & T_2^1 & T_3^1 & T_4^1 & T_5^1 & T_6^1 & T_7^1 & T_8^1 \\
t_2 & T_1^2 & T_2^2 & T_3^2 & T_4^2 & T_5^2 & T_6^2 & T_7^2 & T_8^2 \\
t_3 & T_1^3 & T_2^3 & T_3^3 & T_4^3 & T_5^3 & T_6^3 & T_7^3 & T_8^3 \\
 & & & & \vdots & & & & \\
t_n & T_1^n & T_2^n & T_3^n & T_4^n & T_5^n & T_6^n & T_7^n & T_8^n
\end{array}
\xrightarrow{\phi}$$

$$\frac{T_i^{j+1} - T_i^j}{t_{j+1} - t_j} \qquad \frac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{x_{i+1} - 2x_i + x_{i-1}}$$

$$\left(\frac{\Delta T}{\Delta t}\right)_1 \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_1$$

$$\left(\frac{\Delta T}{\Delta t}\right)_2 \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_2$$

$$\vdots \qquad \vdots$$

$$\left(\frac{\Delta T}{\Delta t}\right)_n \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_n$$

Find w in: $$\frac{\Delta T}{\Delta t} = w \frac{\Delta^2 T}{\Delta x^2}$$