Introduction to Deep Learning (DL)

Patrick Pérez

Valeo valeo.ai

Peyresq, July 2018

Plan

Basics (2h)

- Artificial neural nets
- Supervised training, back-prop
- Regularization

Recurrent nets (1h)

- Sequential data and RNNs
- Gating, long-short time memory
- Applications

Convolutional nets (1h)

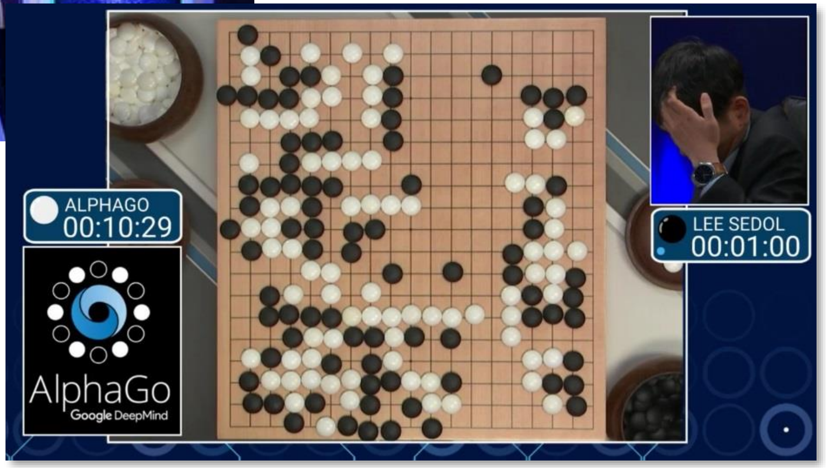
- Sequential data and RNNs
- Gating, long-short time memory
- Applications

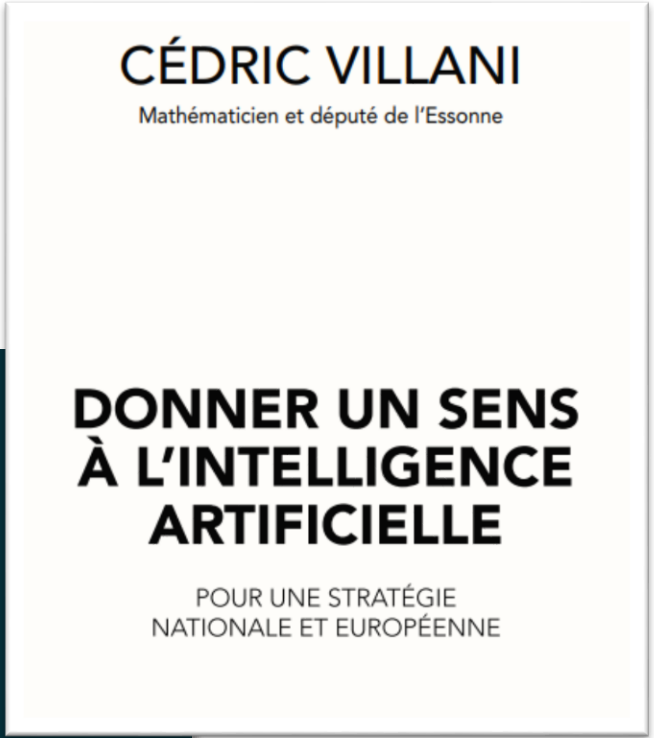
Unsupervised representations (1h)

- Unsupervised / self-supervised
- Auto-encoders
- Deep generative models

Basics

21st century Artificial Intelligence (AI)





21st century Artificial Intelligence (AI)

Deep Learning (DL)

Reinforcement Learning (RL)

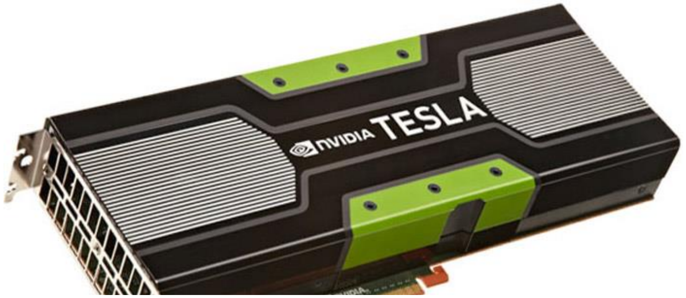
DL in a nutshell

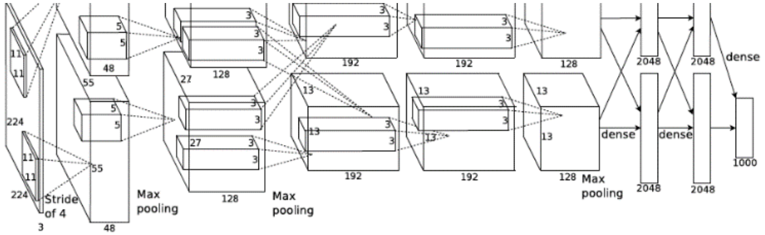
Branch of Machine Learning (ML)

- Dates back to the 40's, survived two winters

Spectacular revival in 2006-2012

- Boost of computational power (GPU)
- Massive amount of data (internet)
- Deeper architectures and improved algorithms





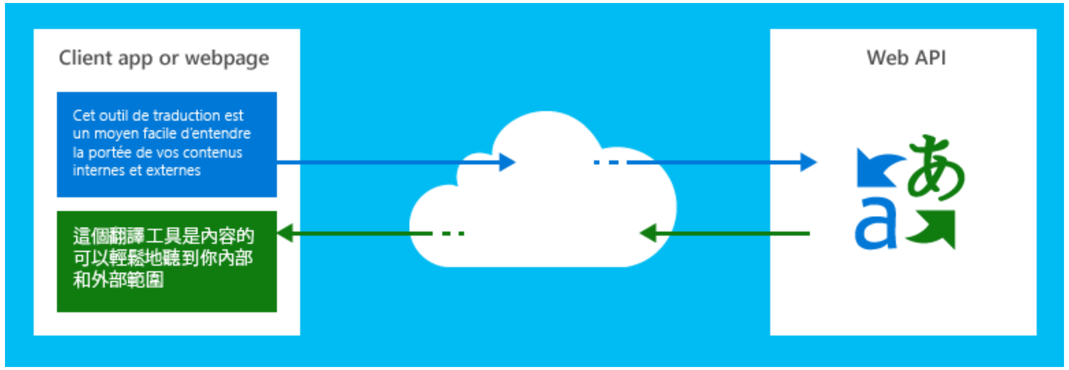
Learn rich representations

- From sequences of non-linear transforms
- For a specific task
- Agnostic to future use

Key applications

Recognition

- Optical character recognition (OCR)
- Automatic speech recognition (ASR)
- Natural language processing (NLP)
- Image understanding

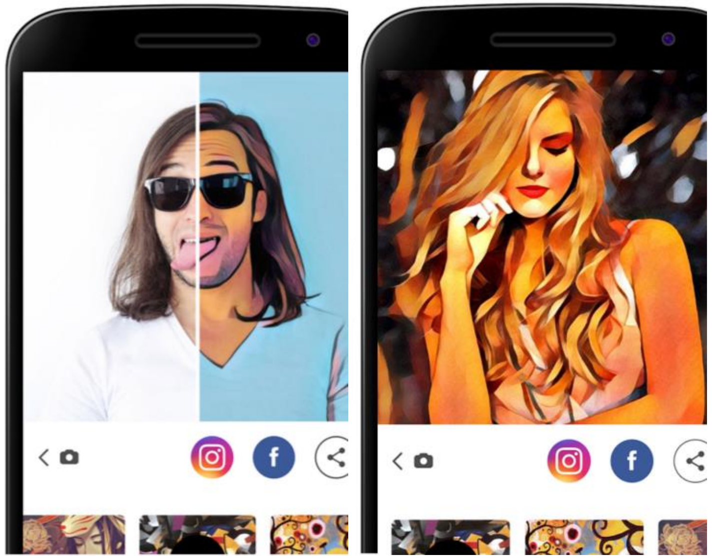


Prediction, decision and control

- Cars, autonomous robots, games, complex systems

Signal transformation

- Enhancement
- Example-based or interactive editing



Machine Learning

From training data, not prior rules

- Learn to solve a prediction task (usually under supervision)
- Learn to extract structure (possibly with no supervision)

$$\mathbf{x} \longrightarrow \boxed{\text{machine*}} \longrightarrow \hat{\mathbf{y}}$$

symbols	symbol(s)
numbers	number(s)

* program with **parameters** and internal **data representation(s)**

Images

$$\mathbf{x} \xrightarrow{\text{machine}} \hat{\mathrm{y}}$$

Images

$$\mathbf{x} \xrightarrow{\text{machine}} \hat{\mathbf{y}}$$

place: “indoor”
object: “person”
action: “eating”

(category from list)

Images

$\mathbf{x}$ → machine → $\hat{y}$

child | little | cute | fun | girl | indoors | baby | people
boy | toddler | innocence | toy | family | youth | birthday
preschool | enjoyment | hungry | one | kindergarten

Images

$\mathbf{x}$ → machine → $\hat{\mathbf{y}}$

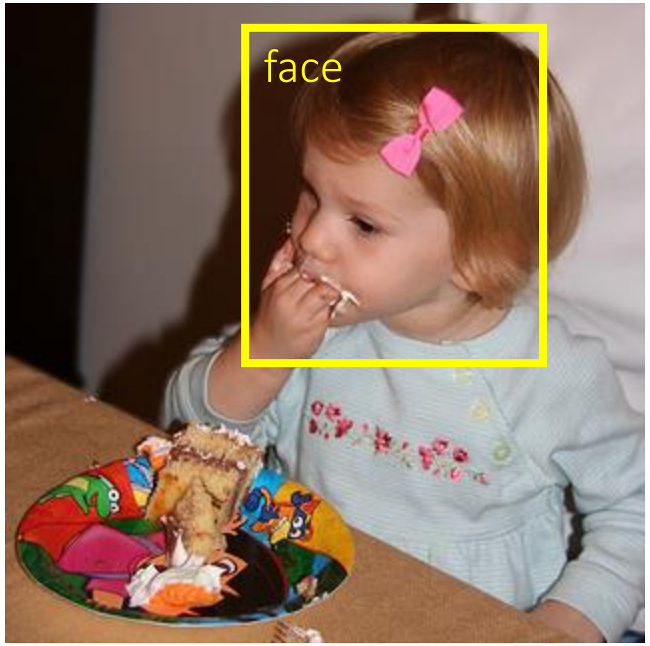
"a little girl is eating a piece of cake"

Images

$\mathbf{x}$ → machine → $\hat{\mathbf{y}}$

Images

$\mathbf{x} \longrightarrow$ machine $\longrightarrow \hat{\mathbf{y}}$

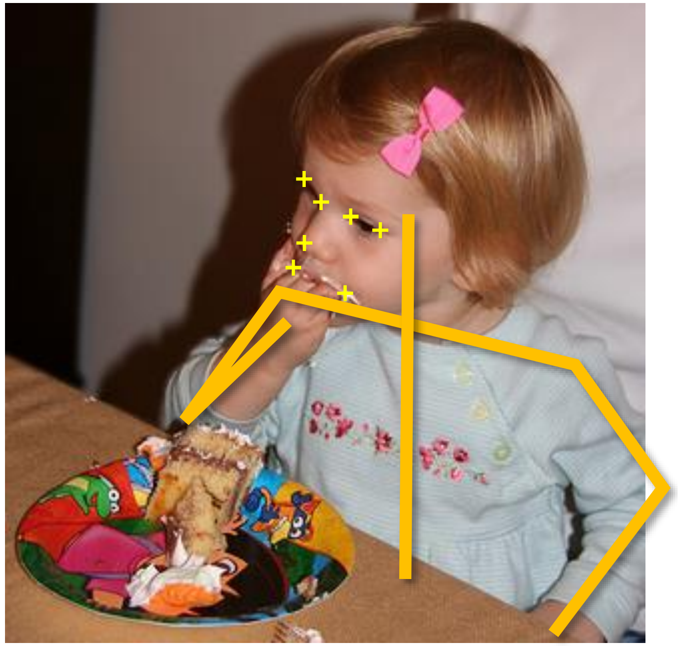


Images

$\mathbf{x}$ → machine → $\hat{\mathbf{y}}$

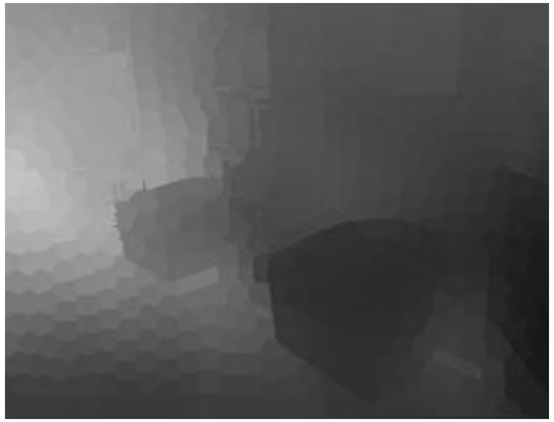
Images

$\mathbf{x}$ → machine → $\hat{\mathbf{y}}$



Images

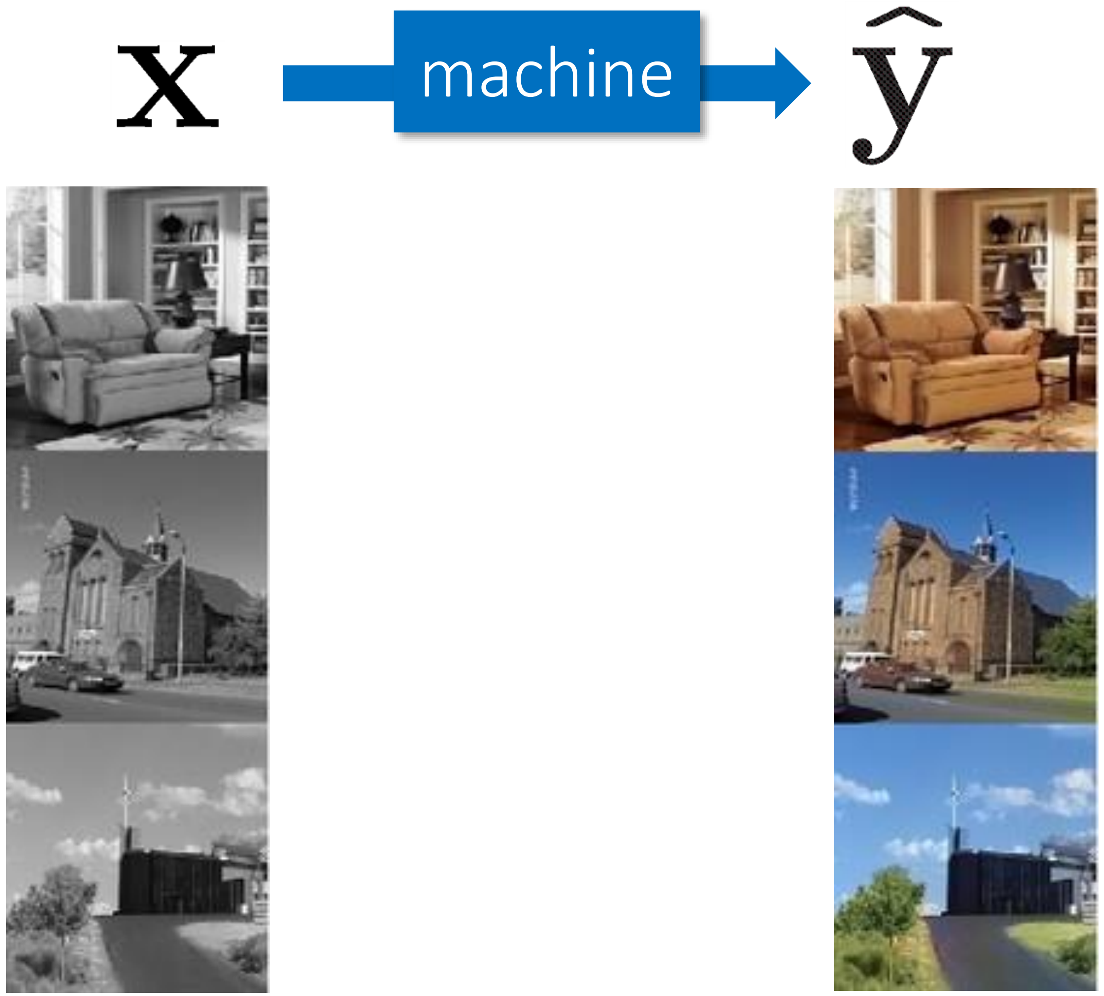
$\mathbf{x}$ → machine → $\hat{\mathbf{y}}$



Images

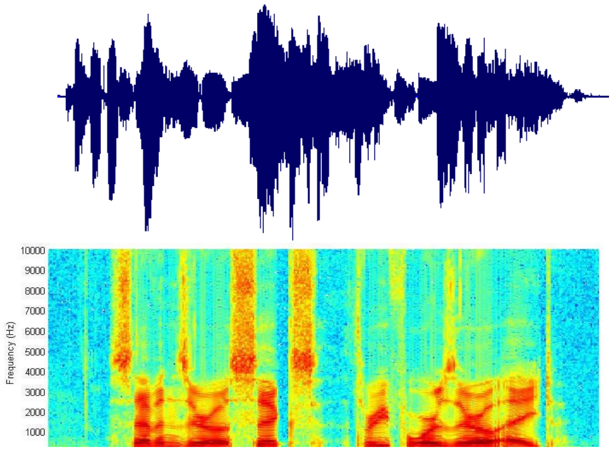
$\mathbf{x}$ → machine → $\hat{\mathbf{y}}$

Images



Audio

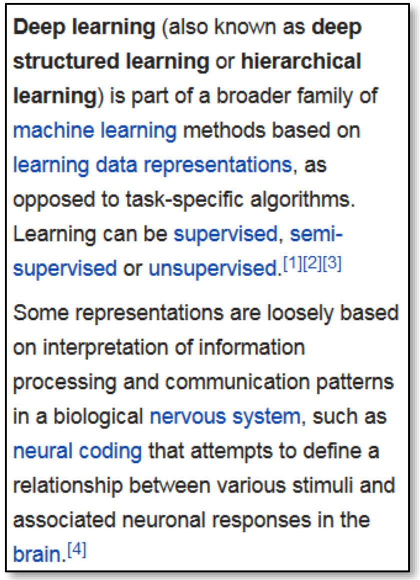
$\mathbf{x}$ → machine → $\hat{\mathbf{y}}$



recognition*: label
speech recog. : words
enhancement: audio
source separation: audio

*audio scene, speech, music – category or instance level

Text


Deep learning (also known as **deep structured learning** or **hierarchical learning**) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised.[1][2][3]

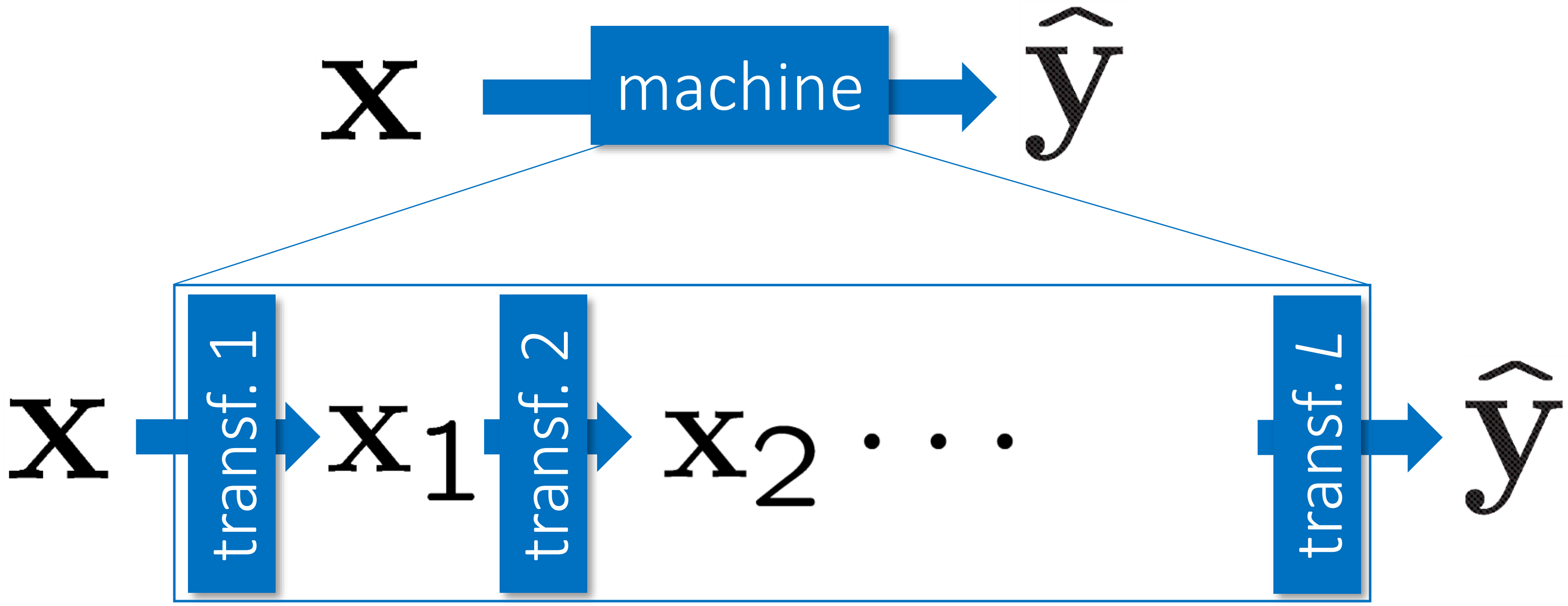
Some representations are loosely based on interpretation of information processing and communication patterns in a biological nervous system, such as neural coding that attempts to define a relationship between various stimuli and associated neuronal responses in the brain.[4]

categorization*: label
summarization: text
translation: text
q. answering: text
visual search: image

* nature, topic, spam

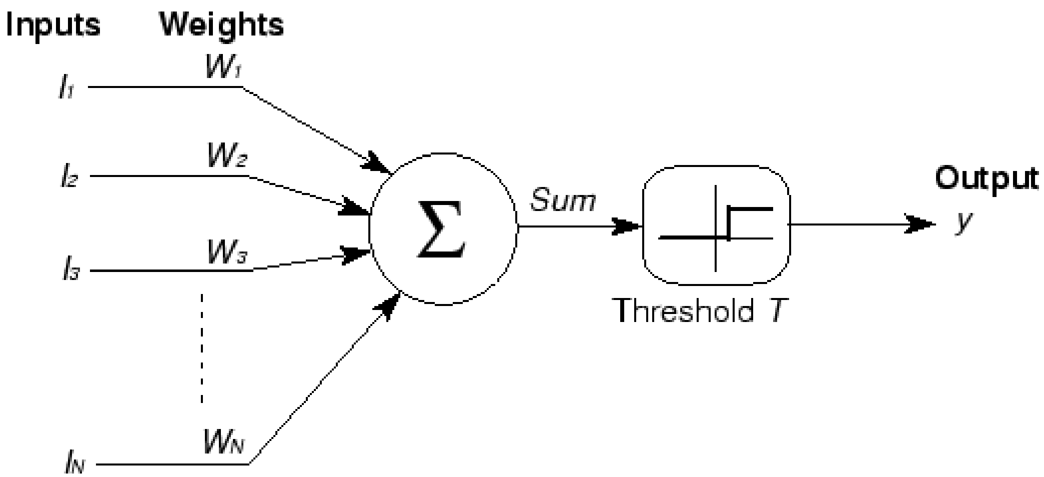
Feed-forward artificial neural net

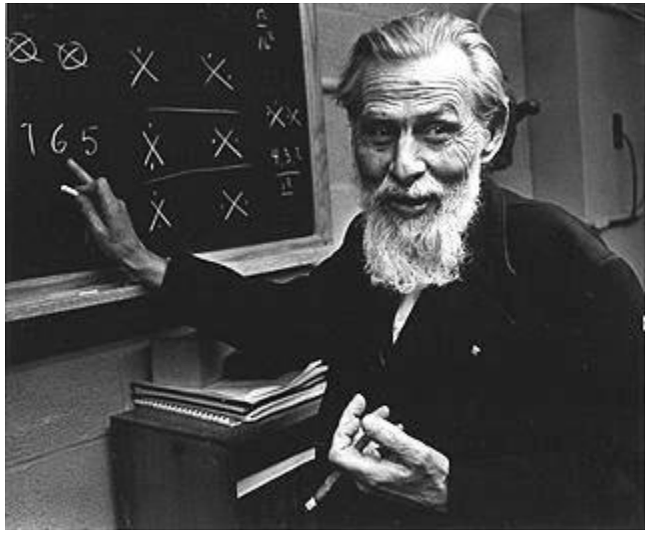
- A sequence of transforms based on formal neurons
- Producing a sequence of representations

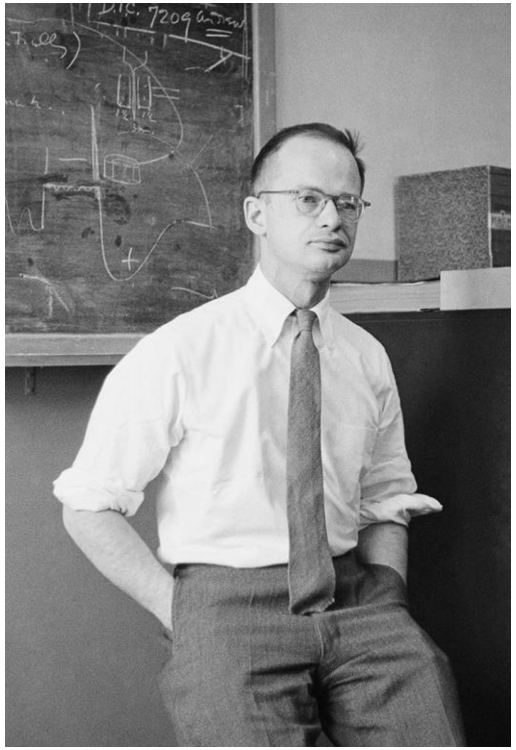


Origins

- McCulloch & Pitts (1943): Linear threshold unit

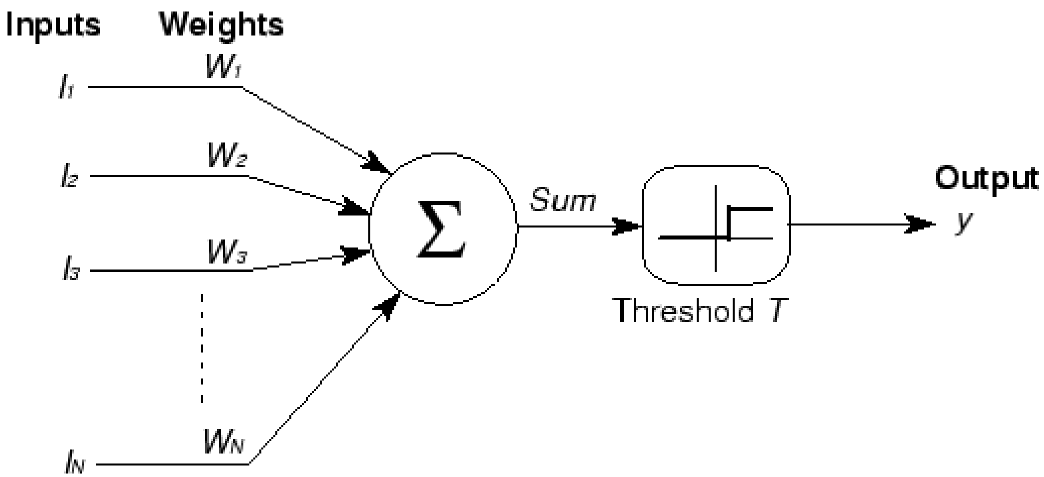


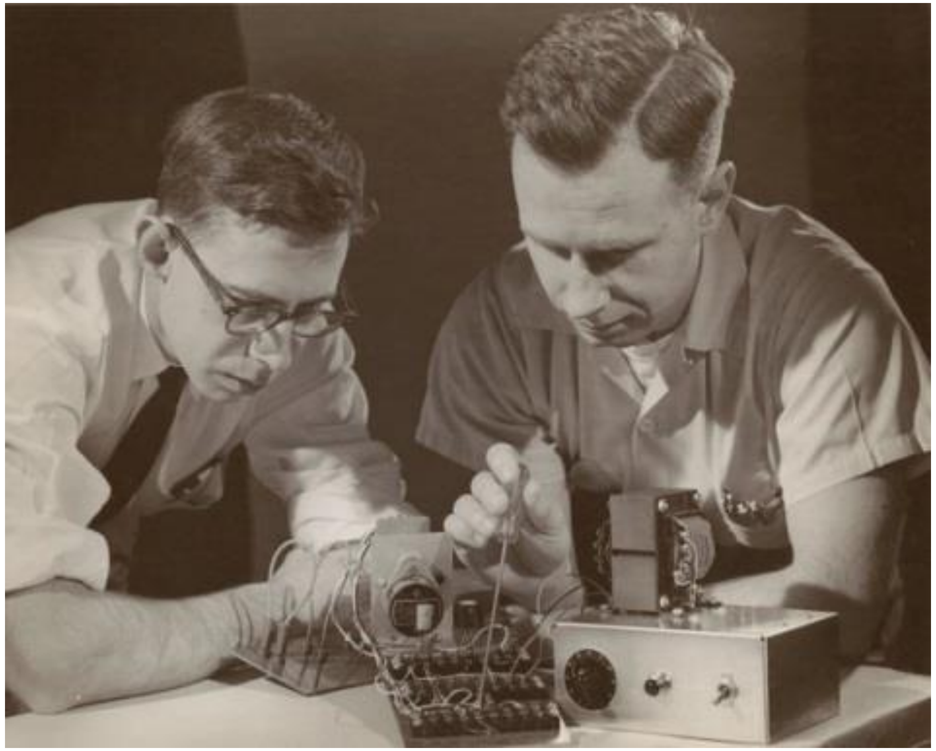




Origins

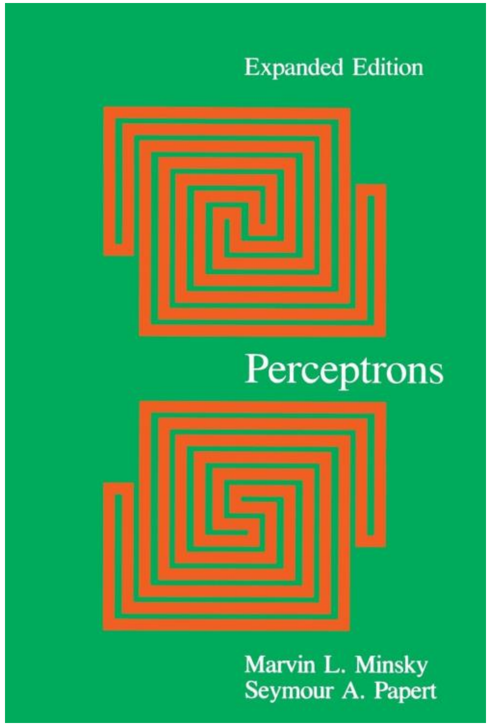
- McCulloch & Pitts (1943): Linear threshold unit
- Rosenblath (1957-61): Perceptron learning algorithm

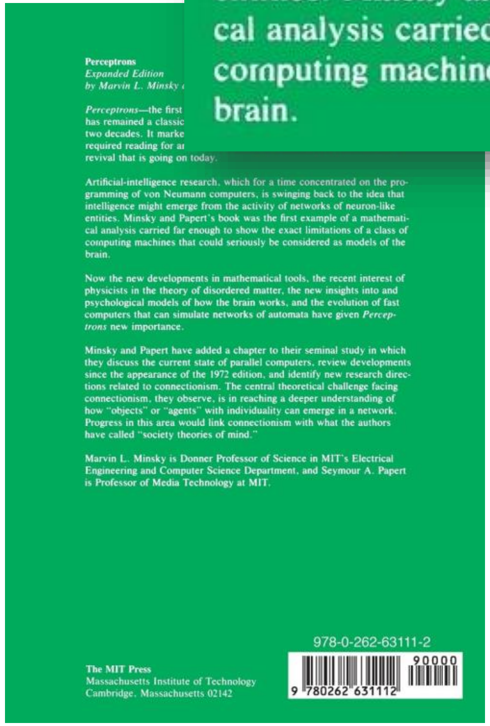




Origins

- McCulloch & Pitts (1943): Linear threshold unit
- Rosenblath (1957-61): Perceptron learning algorithm
- Minsky and Papert (1969): Analysis of (multilayer) perceptron





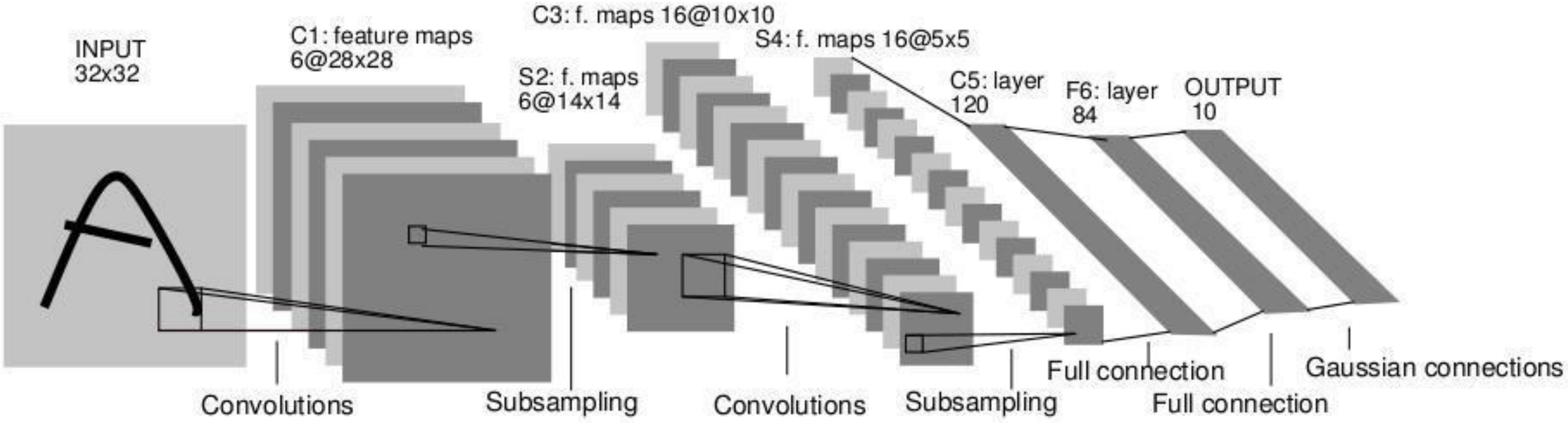
entities. Minsky and Papert's book was the first example of a mathematical analysis carried far enough to show the exact limitations of a class of computing machines that could seriously be considered as models of the brain.

Origins

- McCulloch & Pitts (1943): Linear threshold unit
- Rosenblath (1957-61): Perceptron learning algorithm
- Minsky and Papert (1969): Analysis of (multilayer) perceptron
- Werbos (1974): Gradient back-propagation
- Fukushima (1975-80): Unsupervised multilayer cognitron
- Hinton (1985-93): Auto-encoders

Origins

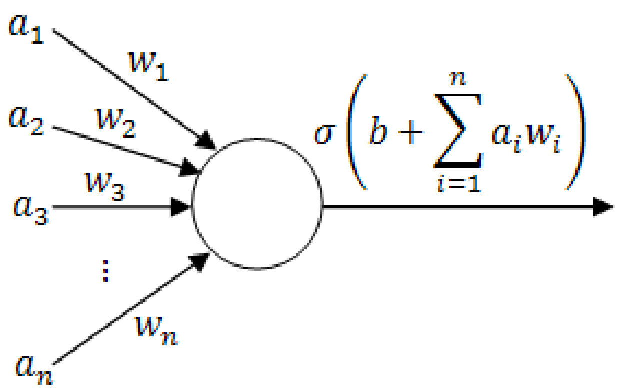
- McCulloch & Pitts (1943): Linear threshold unit
- Rosenblath (1957-61): Perceptron learning algorithm
- Minsky and Papert (1969): Analysis of (multilayer) perceptron
- Werbos (1974): Gradient back-propagation
- Fukushima (1975-80): Unsupervised multilayer cognitron
- Hinton (1985-93): Auto-encoders
- LeCun (1989-98): Modern convolutional nets

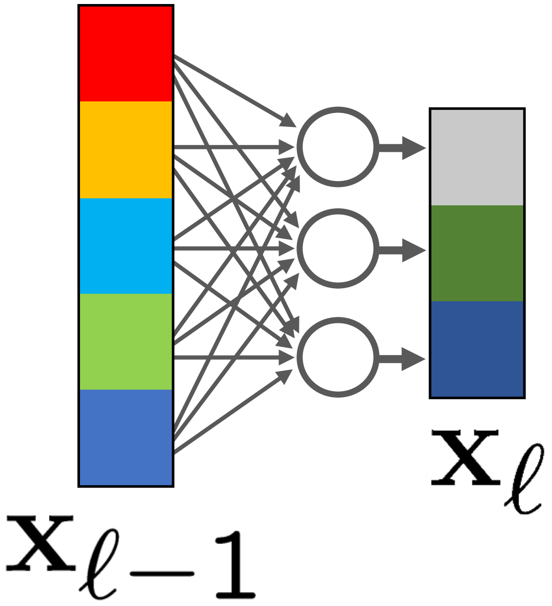


Feed-forward nets

depth: 1 — shallow; 10 — deep; 100 — very deep

transforms: built on ordered layers of artificial neurons

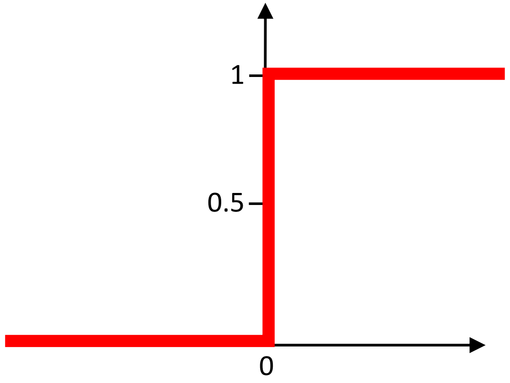


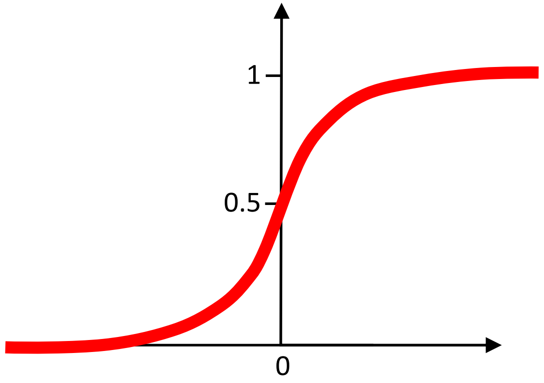


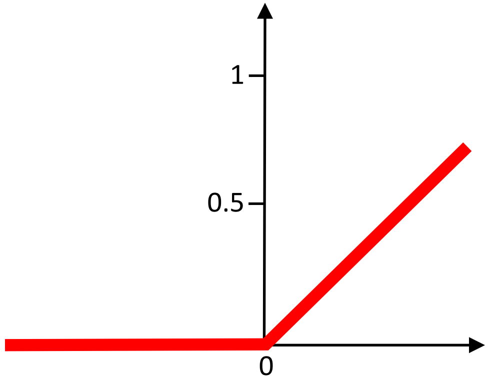
$$\mathbf{x}_\ell = \sigma\left(\mathrm{W}_\ell \mathbf{x}_{\ell-1} + \mathbf{b}_\ell\right)$$

Non-linear activation

Heavyside step	Sigmoid	Rectified Linear Unit (ReLU)
$\sigma(a) = [\![a > 0]\!]$	$\sigma(a) = \frac{1}{1+\exp(-a)}$	$\sigma(a) = \max(0, a)$

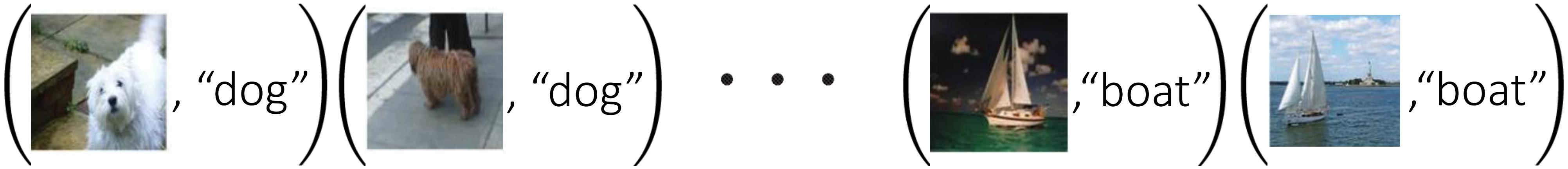




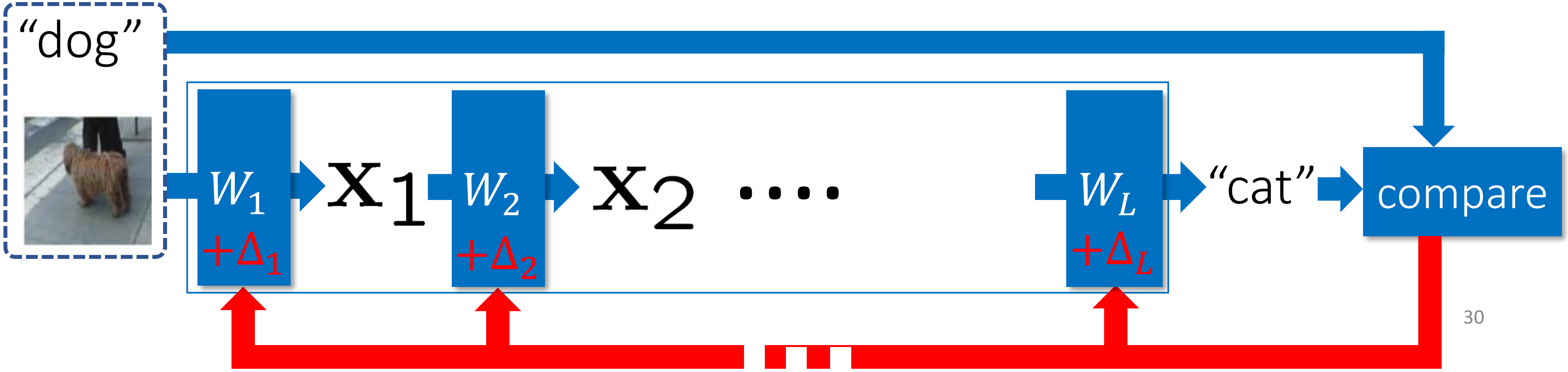


Supervised learning

- Training data: large collection of annotated examples



- Learn parameters (1-100M) by error back-propagation



Using trained models

- Same task, same domain: *reuse*
- Same task, different domain: *fine-tune*
- Similar data, different task: *adapt*, train/fine-tune

- Same data type: use off-the-shelf intermediate *deep representations*

Using trained models

- Same task, same domain: *reuse*
- Same task, different domain: *fine-tune*
- Similar data, different task: *adapt*, train/fine-tune

- Same data type: use off-the-shelf intermediate *deep representations*

Tools

Open source frameworks

- Torch (NYU/Facebook) [C++/Lua]
- Caffe (Berkeley) [C++/Python/Matlab]
- Keras (F. Chollet) [Theano/Tensorflow]
- CNTK (Microsoft) [C++]
- TensorFlow (Google)[C++/Python]
- Pytorch (Facebook)[Python]
- MXNet (AWS)[multi]

Hardware specific

- Nvidia SDK, CuDNN
- Qualcomm SDK

Take home message

- Learn from examples instead of knowledge-based rules
- Learn end-to-end pipeline
  - From input,
  - through intermediate, distributed representations of increasing abstraction,
  - to predicted output.
- Many generic bricks
- Trained architectures and quality codes available

Adapt them to your data and problems

Learning and representations

High level goal

- Turn input “signals” into “representations” (descriptors/features/embedding/codes)
- Learn to reason on codes

Two pipelines

- Classic: Engineer generic features, pool them into codes, learn predictors on codes
- Deep: Learn representation jointly with a task solver… or independently of tasks

Supervisions

• (Fully) supervised	all training examples fully annotated
• Weakly supervised	all training examples partially annotated
• Un-supervised	training examples with no annotation
• Semi-supervised	fraction of training examples fully annotated
• Self-supervised	automatic annotation of training examples
• Privileged	annotation richer than task requires

Supervised training

( , “dog”) ( , “dog”) $\cdots$ ( , “boat”) ( , “boat”)

training set: $\left(\mathbf{x}^{(n)}, \mathbf{y}^{(n)}\right),\ n = 1 \cdots N$

$$\begin{pmatrix} \mathbf{x}^{(n)} \\ \mathbf{y}^{(n)} \end{pmatrix} \xrightarrow{\text{machine}} f(\mathbf{x}^{(n)}; \boldsymbol{\theta}) \overset{?}{\approx} \mathbf{y}^{(n)}$$

find “best” $\boldsymbol{\theta}$ given training set

Risk and loss

- Training set $\mathcal{T} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^{N}$
- Predictor family $\mathcal{F} = \{f : \mathcal{X} \to \mathcal{Y}\}$
- Loss
$$E(\mathbf{y}, \mathbf{y}') \geqslant 0,\ \forall(\mathbf{y}, \mathbf{y}') \in \mathcal{Y} \times \mathcal{Y}$$
$$E(\mathbf{y}, \mathbf{y}) = 0,\ \forall \mathbf{y} \in \mathcal{Y}$$
- Expected and empirical risks

$$\mathcal{R}(f) = \mathbb{E}\Big[E\Big(f(\mathbf{X}), \mathbf{Y}\Big)\Big] \approx \frac{1}{N}\sum_{n=1}^{N} E\Big(f\Big(\mathbf{x}^{(n)}\Big), \mathbf{y}^{(n)}\Big)$$

Minimizing empirical risk

- Parametric predictors $\mathcal{F} = \{f(.;\theta),\ \theta \in \Theta\}$
- Learning problem

$$\min_{\theta} \frac{1}{N} \sum_{n=1}^{N} E\Big(f\big(\mathbf{x}^{(n)};\theta\big),\mathbf{y}^{(n)}\Big)$$

- Single framework: classic regression, SVM, kernels, DL
- DL: non-convex optimization, very large dimension (millions)
  - Iterative gradient descent

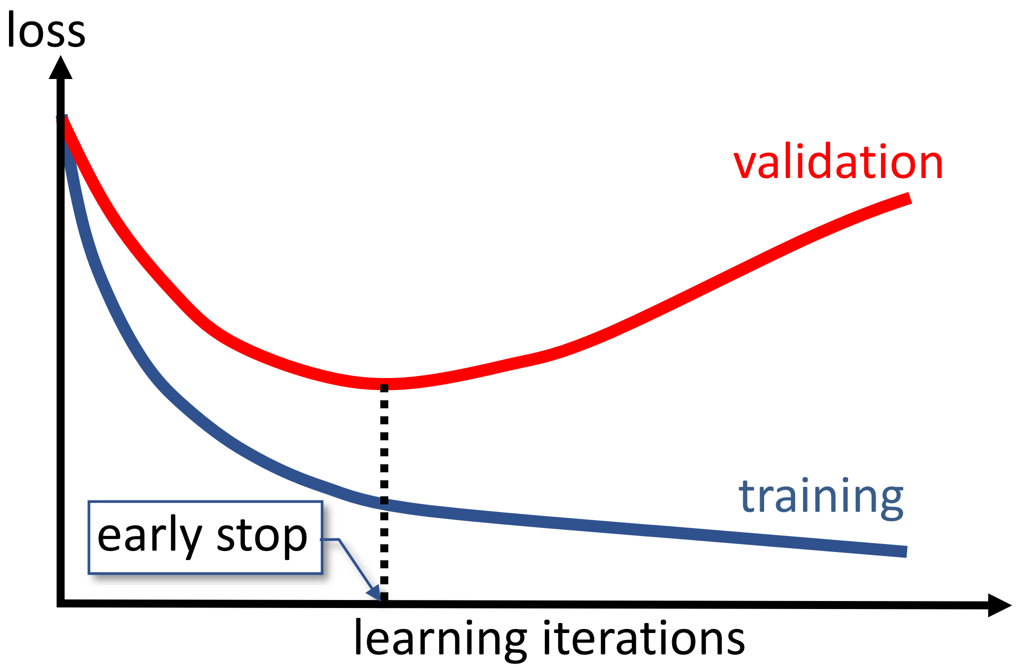
Training, validation and test

Iterative minimization of

$$\min_{\theta} \frac{1}{N} \sum_{n=1}^{N} E\Big(f\big(\mathbf{x}^{(n)}; \theta\big), \mathbf{y}^{(n)}\Big)$$

To avoid overfitting for good generalization

- Monitor error on validation set
- Use early stopping
- Balance model capacity vs. data amount
- Use priors (e.g., penalize large parameters)



Evaluate performance on test set

Gradient descents

$$\min_{\theta} \frac{1}{N} \sum_{n=1}^{N} E\Big(f\big(\mathbf{x}^{(n)}; \theta\big), \mathbf{y}^{(n)}\Big)$$

Gradient descent : follow steepest descent direction at current location

$$\theta^{(k+1)} = \theta^{(k)} - \frac{\alpha}{N} \sum_{n=1}^{N} \nabla_\theta E\Big(f(\mathbf{x}^{(n)}; \theta^{(k)}), \mathbf{y}^{(n)}\Big)$$

Stochastic gradient descent (SGD): process one sample at a time

$$\theta^{(k+1)} = \theta^{(k)} - \alpha \nabla_\theta E\Big(f(\mathbf{x}^{(n_k)}; \theta^{(k)}), \mathbf{y}^{(n)}\Big)$$

Minibatch SGD: process one small subset of samples at a time

$$\theta^{(k+1)} = \theta^{(k)} - \frac{\alpha}{|B_k|} \sum_{n \in B_k} \nabla_\theta E\Big(f(\mathbf{x}^{(n)}; \theta^{(k)}), \mathbf{y}^{(n)}\Big)$$

Epochs and learning rates

Epoch: a complete pass over all training examples

Learning rate: progressively slow down, *e.g.* after each epoch

Minibatch SGD: process one small subset of samples at a time

initialize weights to small random values

for $e = 1 \cdots E$

- update rate: α_e
- shuffle training set, divide in K batches
- for $k = 1 \cdots K$
  - update weights:
    $\theta^{(k+1)} = \theta^{(k)} - \frac{\alpha}{|B_k|} \sum_{n \in B_k} \nabla_\theta E\Big(f(\mathbf{x}^{(n)}; \theta^{(k)}), \mathbf{y}^{(n)}\Big)$

Case of feed-forward net

- Sequential functional definition

$$\hat{\mathbf{y}} = f_L \circ f_{L-1} \circ \cdots \circ f_1(\mathbf{x}) = f(\mathbf{x}; \underbrace{W_1 \cdots W_L}_{\theta})$$

- Backward parameter dependencies

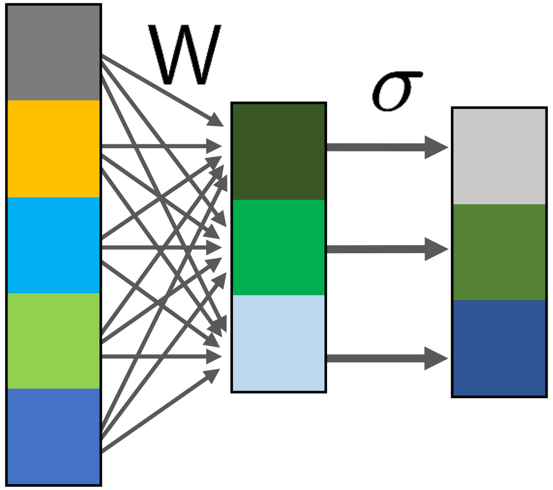
$$\hat{\mathbf{y}} = \mathsf{Fun}(\mathbf{x}_{\ell-1}, W_\ell, W_{\ell+1} \cdots W_L),\ \ell = 1 \cdots L-1$$

- Back-propagation: efficient gradient computation with chain-rule

Single-layer Perceptron

$$\hat{\mathbf{y}} = \sigma(\underbrace{\mathsf{W}\mathbf{x}}_{\mathbf{a}})$$

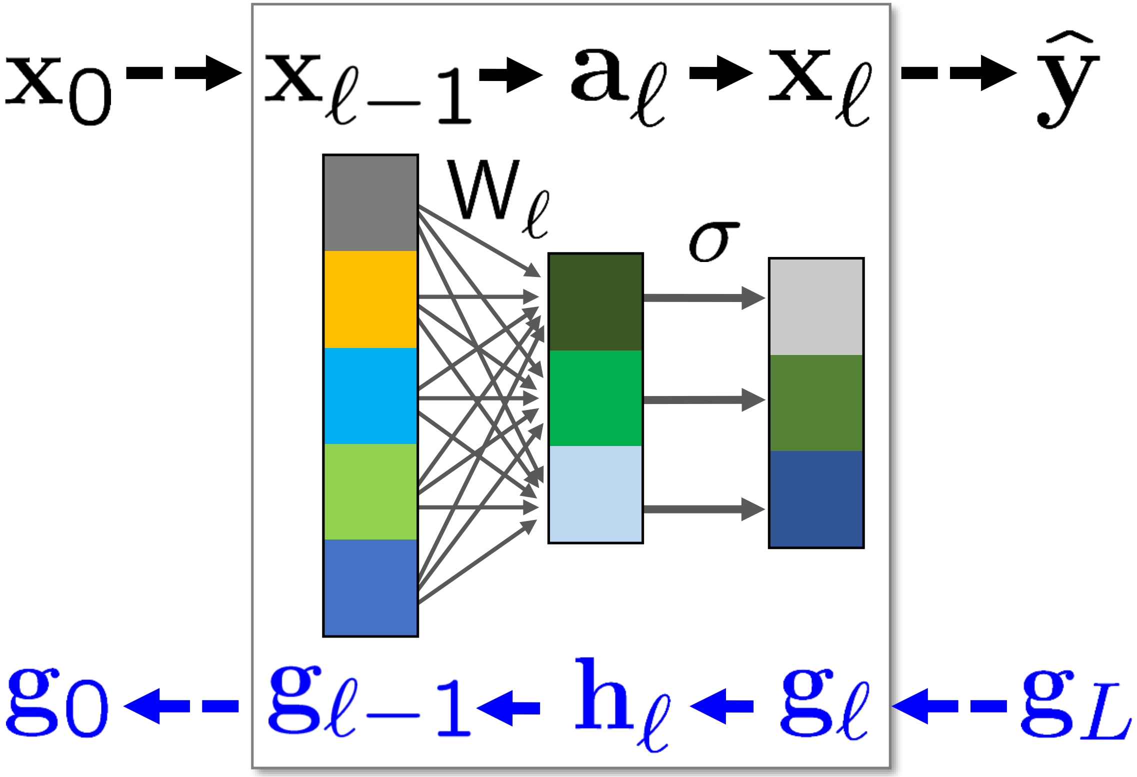
$$\mathbf{x} \rightarrow \mathbf{a} \rightarrow \hat{\mathbf{y}}$$



$$\frac{\partial E(\hat{\mathbf{y}}, \mathbf{y})}{\partial \mathsf{W}} = \left[\sigma'(\mathbf{a}) \odot \frac{\partial E(\hat{\mathbf{y}}, \mathbf{y})}{\partial \hat{\mathbf{y}}}\right] \mathbf{x}^\top$$

Multi-layer Perceptron (MLP)

$$\mathrm{W}_\ell \mathbf{x}_{\ell-1} = \mathbf{a}_\ell, \ \sigma(\mathbf{a}_\ell) = \mathbf{x}_\ell$$



$$\mathbf{h}_\ell = \frac{\partial E(\hat{\mathbf{y}},\mathbf{y})}{\partial \mathbf{a}_\ell} \qquad \mathbf{g}_\ell = \frac{\partial E(\hat{\mathbf{y}},\mathbf{y})}{\partial \mathbf{x}_\ell}$$

$$\mathbf{g}_L = \frac{\partial E(\hat{\mathbf{y}},\mathbf{y})}{\partial \hat{\mathbf{y}}}$$

for $\ell = L \cdots 1$

$$\mathbf{h}_\ell = \sigma'(\mathbf{a}_\ell) \odot \mathbf{g}_\ell$$

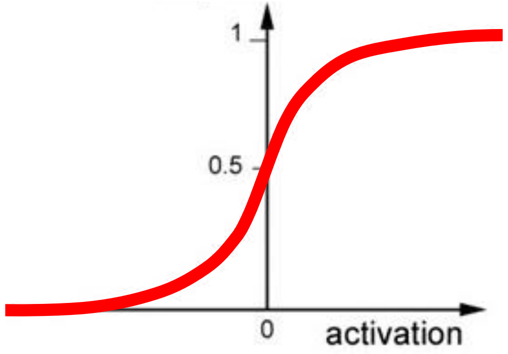
$$\mathbf{g}_{\ell-1} = \mathrm{W}_\ell^\top \mathbf{h}_\ell$$

$$\frac{\partial E(\hat{\mathbf{y}},\mathbf{y})}{\partial \mathrm{W}_\ell} = \mathbf{h}_\ell \mathbf{x}_{\ell-1}^\top$$

Non-linearities

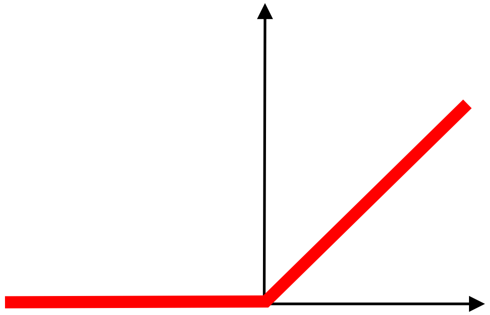
Sigmoid

$$\sigma(a) = \frac{1}{1+\exp(-a)}$$



Rectified Linear Unit (ReLU)

$$\sigma(a) = \max(0, a)$$



Classification

- Softmax: pseudo-probabilistic output

$$\hat{\mathbf{y}} = \frac{\exp(\mathbf{a}_L)}{\mathtt{sum}\Big(\exp(\mathbf{a}_L)\Big)} \in [0,1]^C, \ \|\hat{\mathbf{y}}\|_1 = 1$$

- "One-hot" ground-truth output (indicator)

$$\mathbf{y} \in \{0,1\}^C, \ \|\mathbf{y}\|_1 = 1$$

- Cross-entropy loss (maximum log-likelihood)

$$E(\hat{\mathbf{y}}, \mathbf{y}) = \mathbb{H}[\mathbf{y}, \hat{\mathbf{y}}] = -\sum_{c=1}^{C} y_c \ln \hat{y}_c = -\ln \hat{y}_{c^*}$$

One layer classifier

- Logistic regression

$$\hat{y}_c \propto \exp(\underbrace{\mathbf{w}_c^\top \mathbf{x}}_{a_c})$$

- Loss gradient for one training example

$$\frac{\partial E(\hat{\mathbf{y}}, \mathbf{y})}{\partial \mathbf{a}} = \hat{\mathbf{y}} - \mathbf{y} \quad \frac{\partial E(\hat{\mathbf{y}}, \mathbf{y})}{\partial \mathsf{W}} = (\hat{\mathbf{y}} - \mathbf{y})\mathbf{x}^\top$$

  - By increasing a class activation, the loss decreases if the class is correct, and increases otherwise

Regularization

- L2 penalty: Weight decay

$$\Omega(\theta) = \frac{\gamma}{2}||\theta||_2^2 = \frac{\gamma}{2}\sum_{\ell}||\mathrm{W}_\ell||_F^2$$

  - Gradient $$\frac{\partial E(\hat{\mathbf{y}}, \mathbf{y})}{\partial \mathrm{W}_\ell} = \mathbf{h}_\ell \mathbf{x}_{\ell-1}^\top + \gamma \mathrm{W}_\ell$$

- L1 penalty: Sparsity enforcing prior

$$\Omega(\theta) = \gamma||\theta||_1$$

  - *Lasso* special case

Regression

- Square loss

$$E(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{2}\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2$$

- From least square regression to deep regression

Learning rate issue

Towards better adaptation of learning rate

- Momentum $$g_k = \nabla_\theta E_{B_k}(\theta^{(k)}),$$
- AdaGrad $$G_k = \mathrm{diag}\Big(\sum_{m=1}^{k} g_m g_m^\top\Big),$$
- Adam $$m_{k+1} = \frac{\beta_1 m_k + (1-\beta_1) g_k}{1-\beta_1^k}$$ $$v_{k+1} = \frac{\beta_2 v_k + (1-\beta_2) g_k^2}{1-\beta_2^k}$$

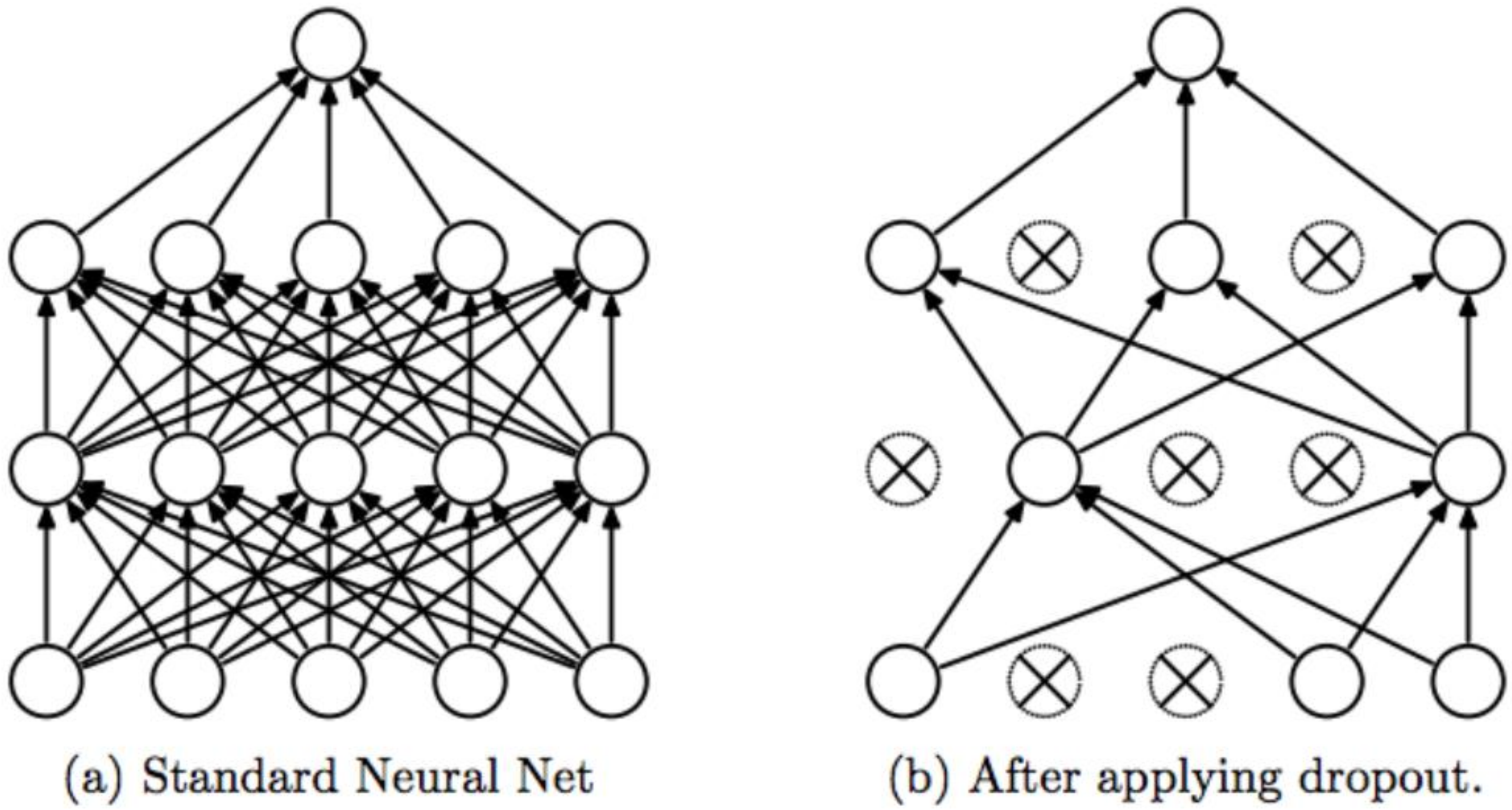
$$\theta^{(k+1)} = \theta^{(k)} - \alpha g_k + \eta \Delta\theta_k$$

$$\theta^{(k+1)} = \theta^{(k)} - \alpha G_k^{-1/2} g_k$$

$$\theta^{(k+1)} = \theta^{(k)} - \alpha \frac{m_{k+1}}{\sqrt{v_{k+1}} + \varepsilon}$$

[Rumelhart *et al*. 1986 / Duchi *et al*. 2011 / Kingma and Jimmy 2014]

Drop-out



[Hinton *et al.* 2012]

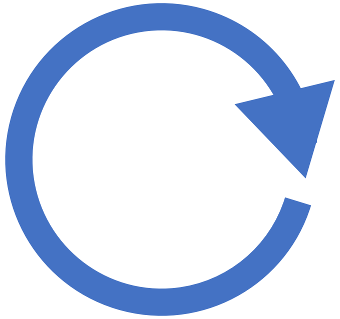
The Art of DL

Data gathering

- Size and quality, annotations
- Augmentation: synthetic transforms
- Real *vs.* synthetic

Loss definition

- Domain knowledge
- Differentiability
- Regularization



Architecture design

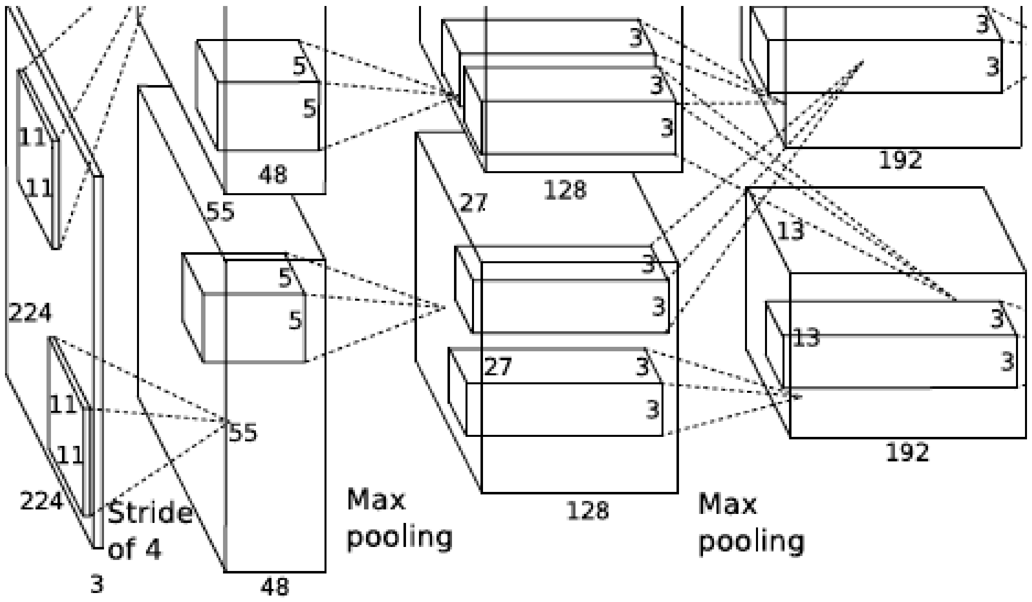
- Exploiting known structures
- Convolutional net, recurrent net
- Lego game and hyper-parameters

Training

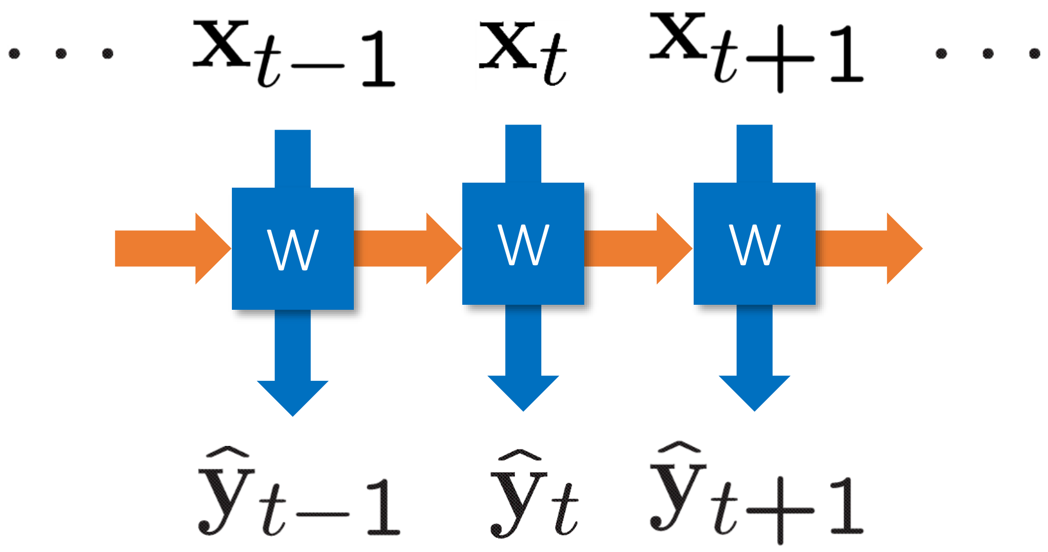
- Initialization
- SGD: batches, learning rate, misc.
- Monitoring, early stopping

Structured inputs and/or outputs

2D data: convolutional nets



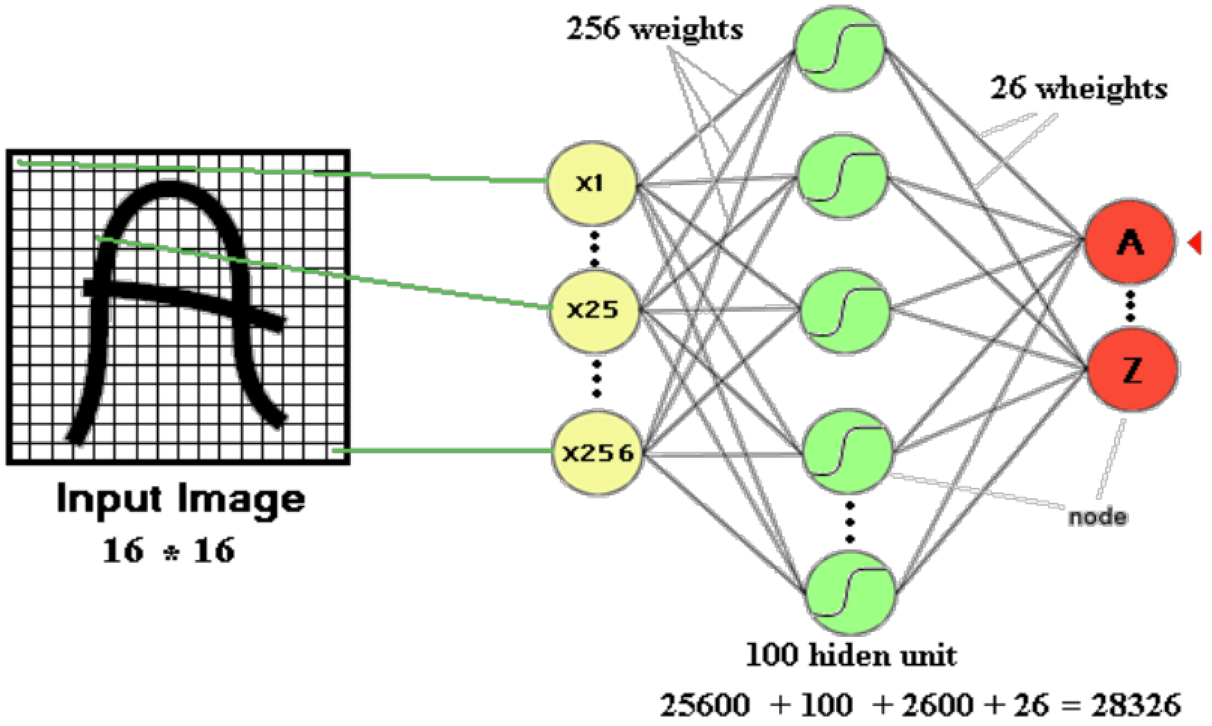
Sequences: recurrent nets



ConvNets

MLP for images?

Scalability

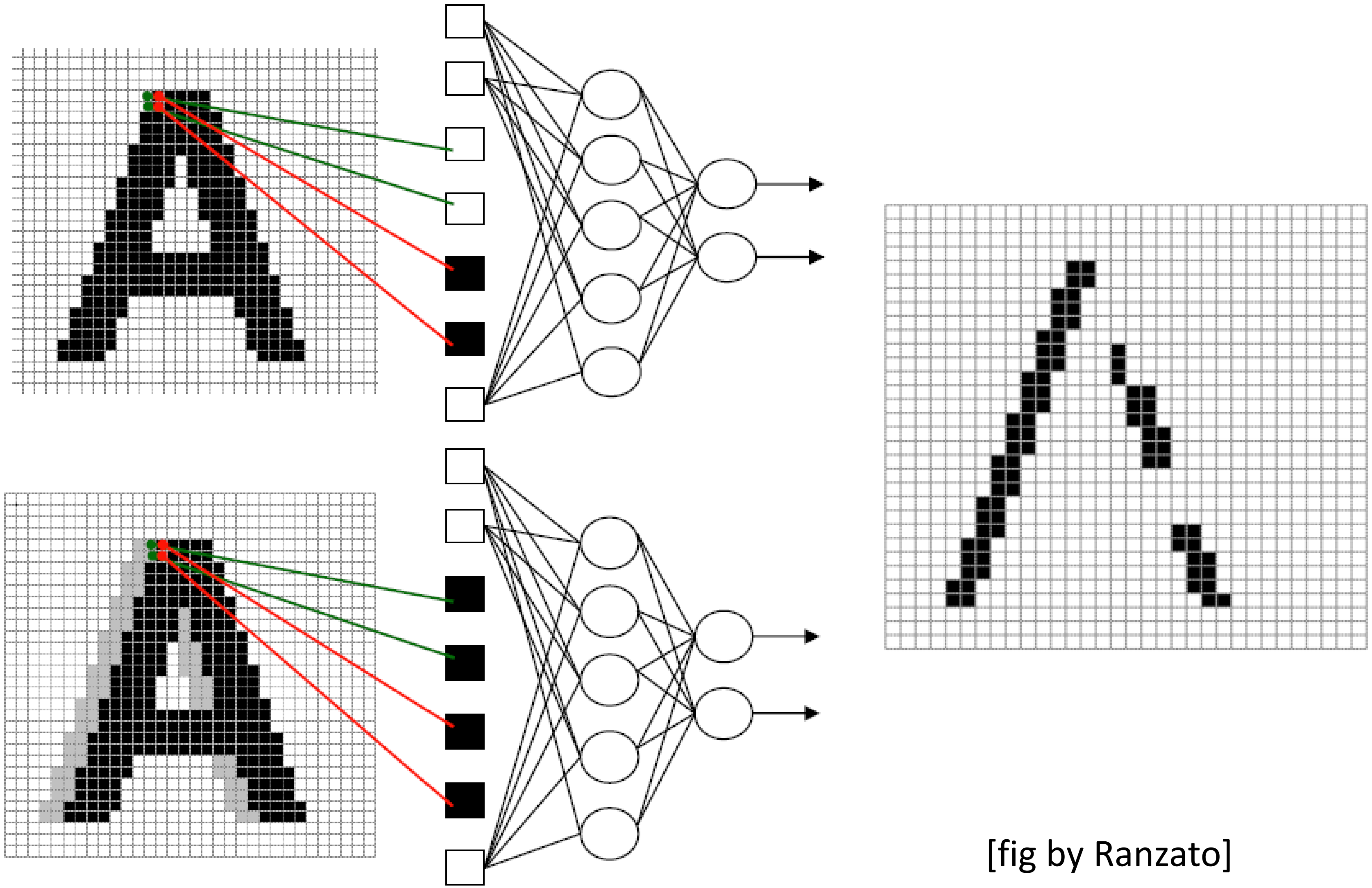


- 1Mpix image and 1M unit hidden layer: 10^{12} weights

[fig by Ranzato]

MLP for images?

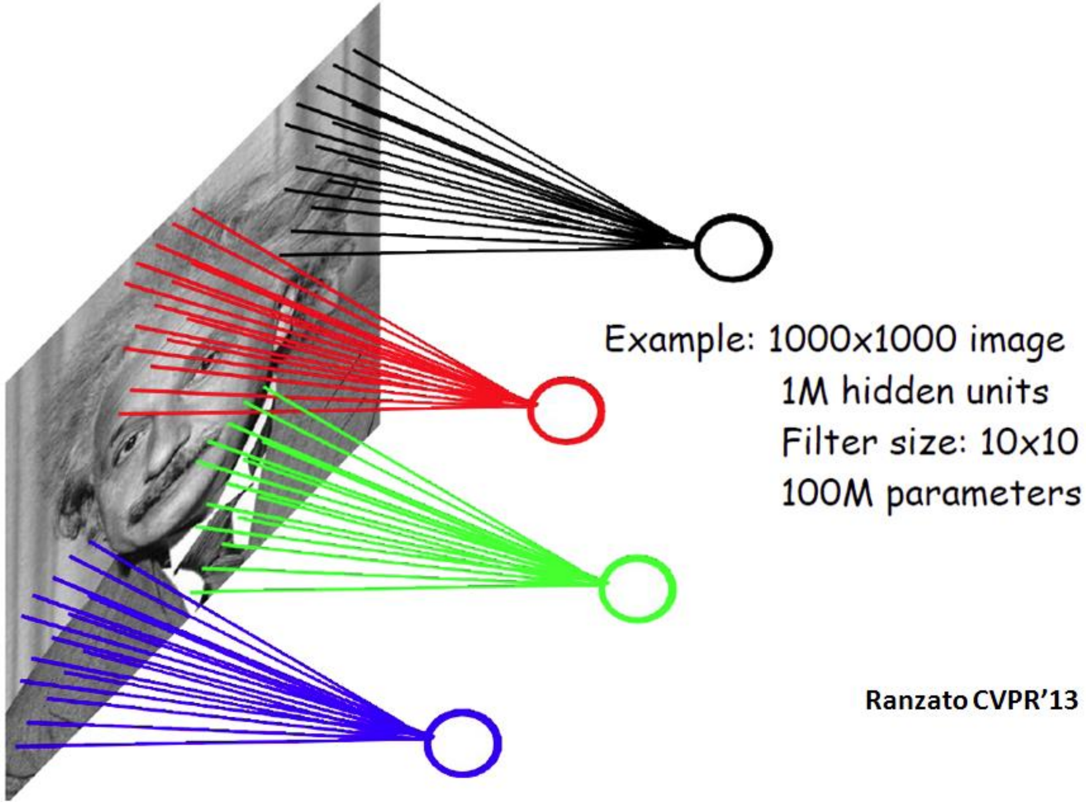
Stability: expect invariance to small input distortions



[fig by Ranzato]

Local linear transforms

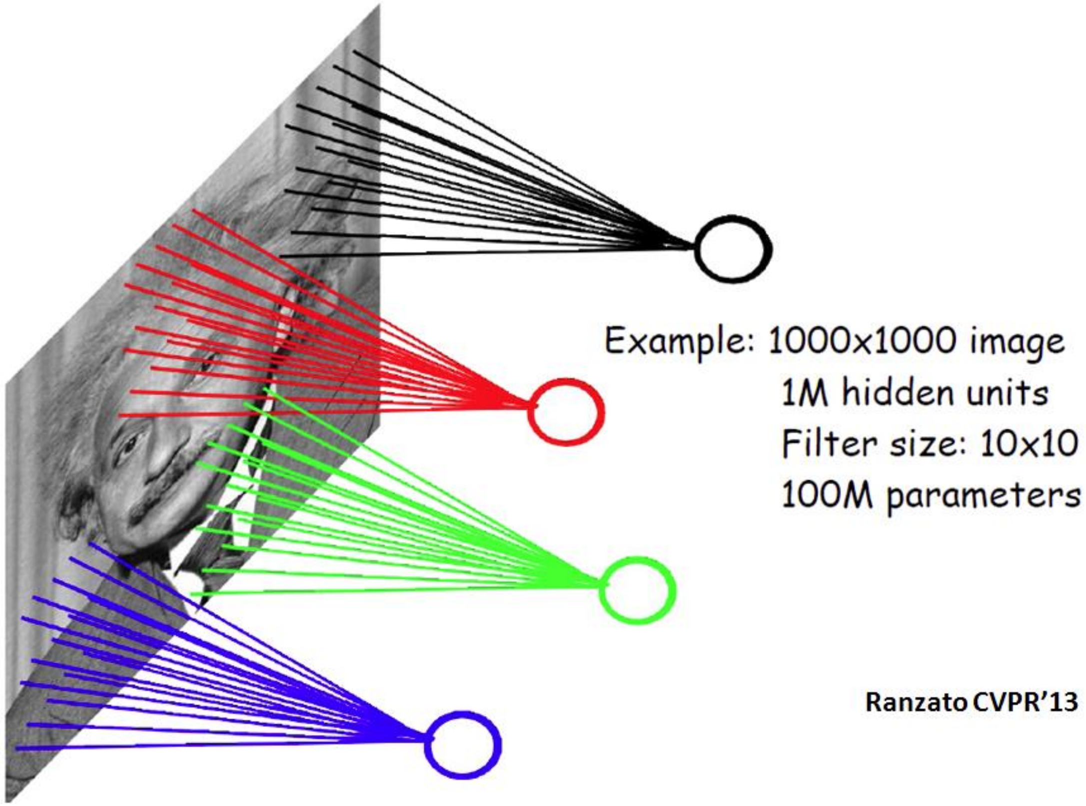
- Each unit connected to a “local patch”
- As in biological vision
  - A cell sensitive to a small region of input (receptive field)
  - Tiled to cover the whole field of view



Ranzato CVPR’13

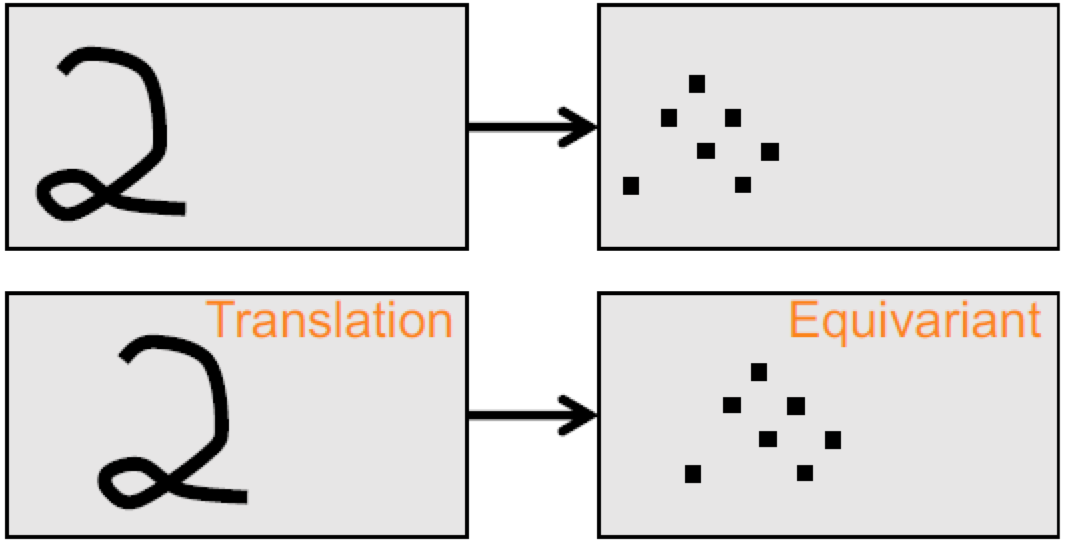
Shared weights

- Weights shared across spatial locations
- Translation-invariant (local) linear filter = convolution with small kernel
- One filter and non-linearity produces one "feature map" or "channel"



Convolutional layer (CL)

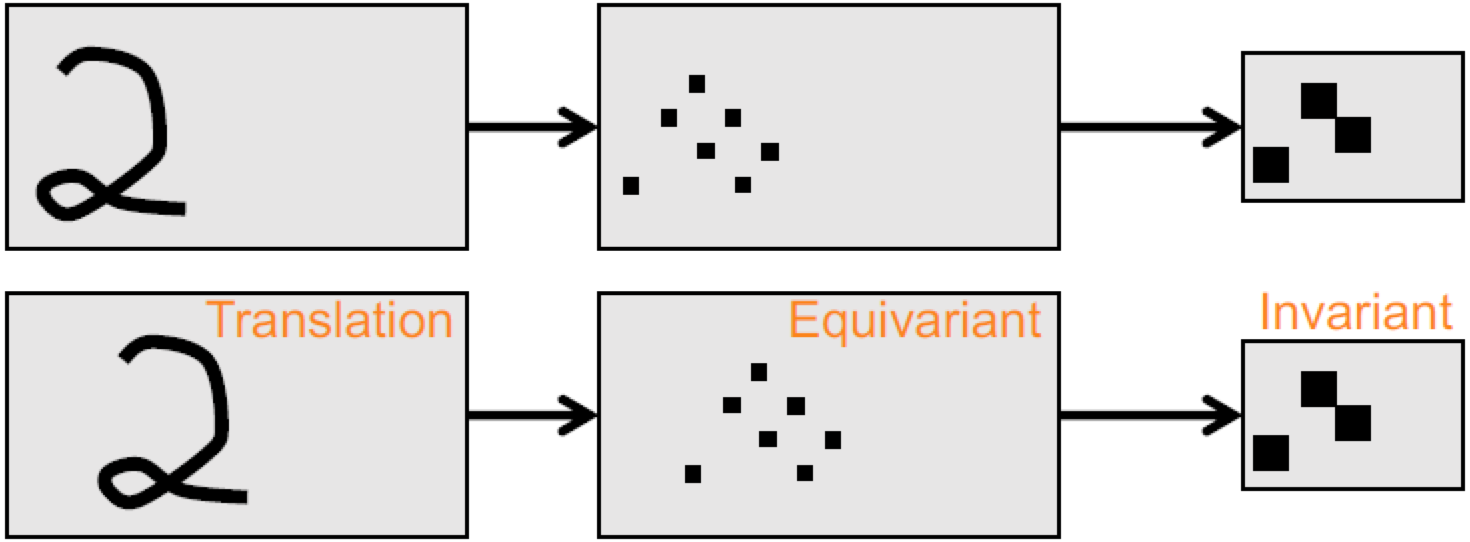
- Building bock of convolutional networks (ConvNet)
- Covariance (not invariance) of map with input



[fig by Ranzato]

Spatial pooling

- Local aggregation of activations (max or sum)
- Small amount of translation invariance
- Reduction of spatial resolution: increase receptive fields of next layers



[fig by Ranzato]

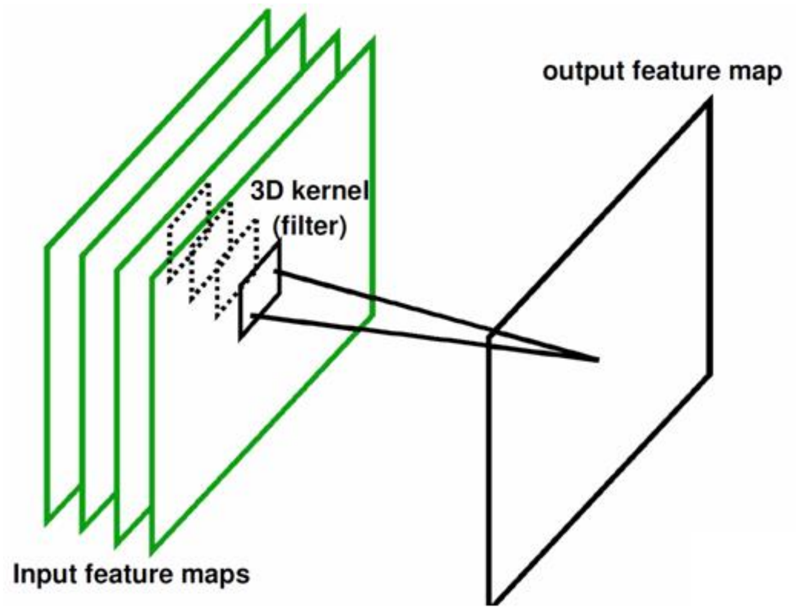
Multiple i/o maps

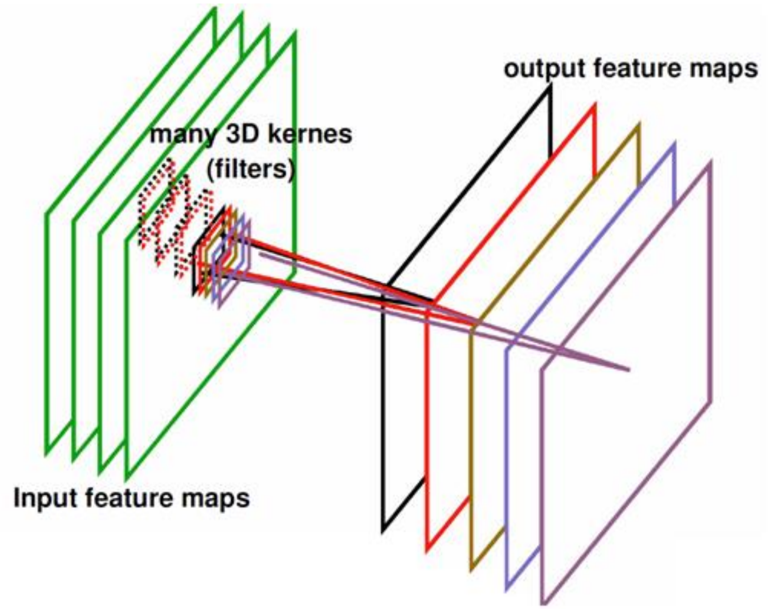
Multiple input maps (*e.g.* color image)

- One filter = stack of 2D filters = 3D filter (tensor)

Multiple output maps

- Number of maps = width of layer

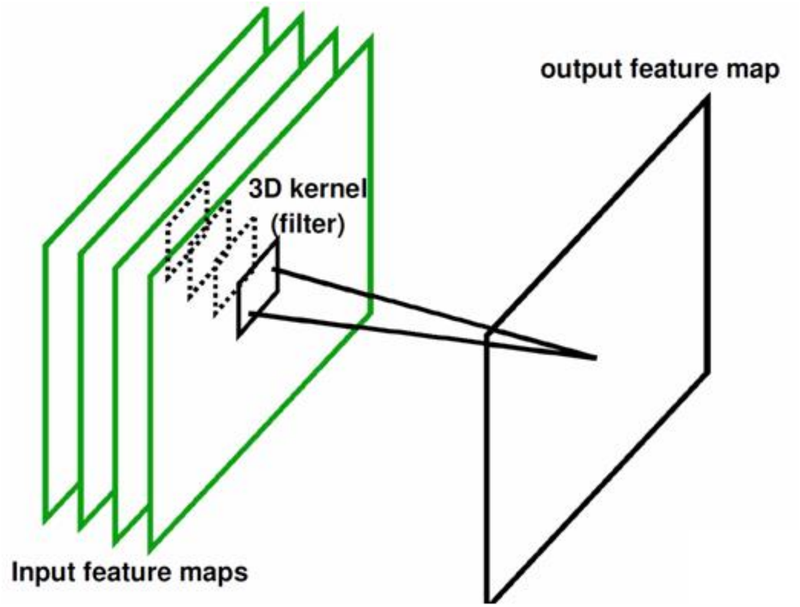




Ranzato CVPR'13

“1x1 convolution”

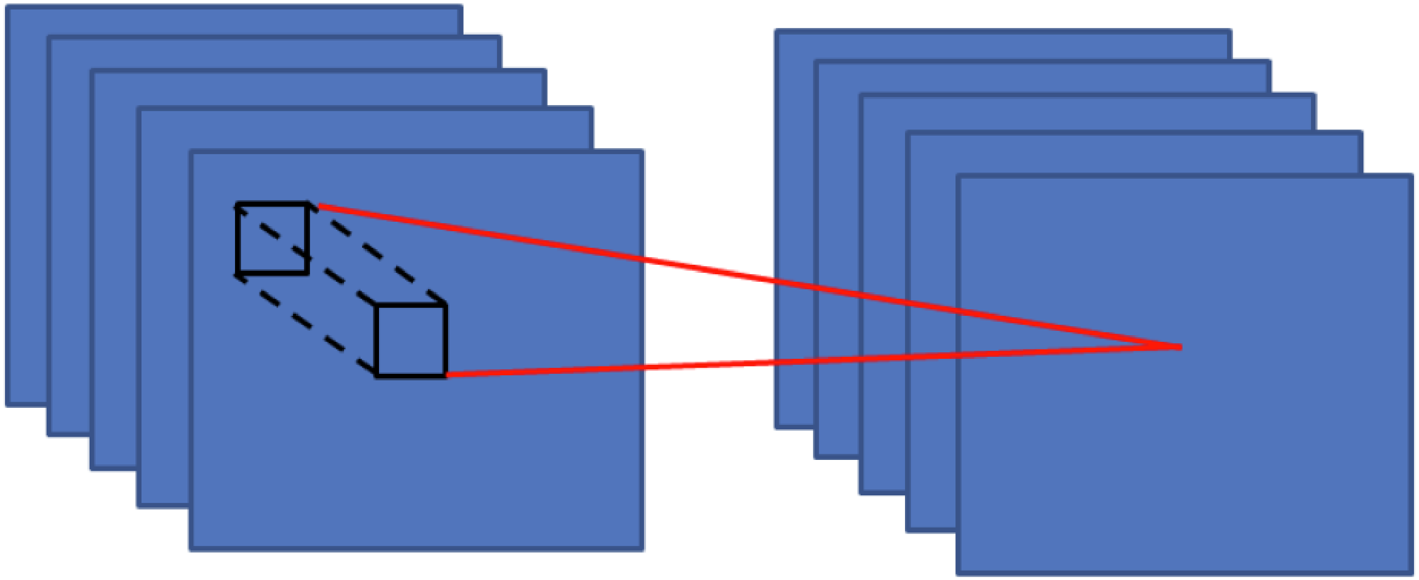
Linear combination of multiple input maps in a single one



[fig by Ranzato]

Normalizations

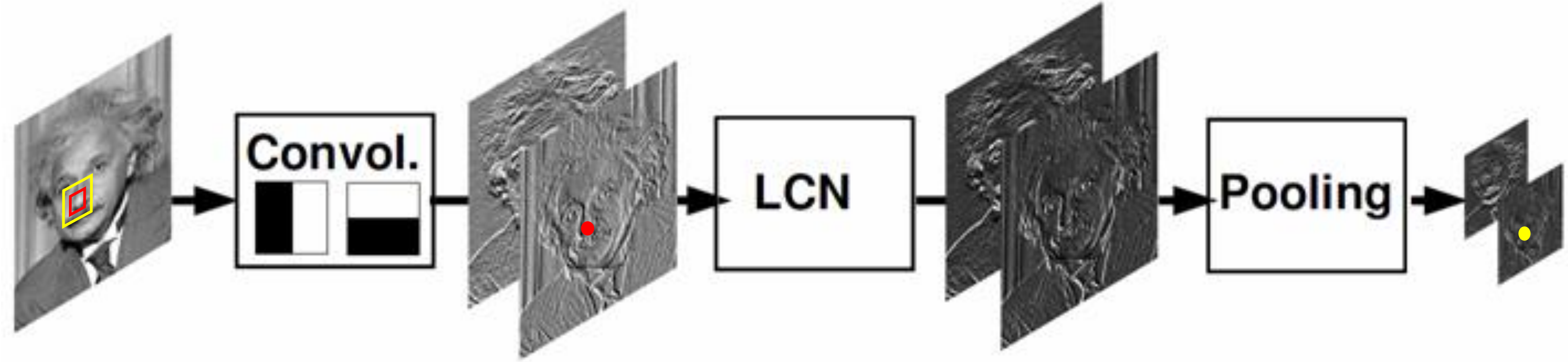
- Local contrast normalization (LCN): normalize activations across channels and local spatial neighborhood



- Batch normalization (BatchNorm): and across batches

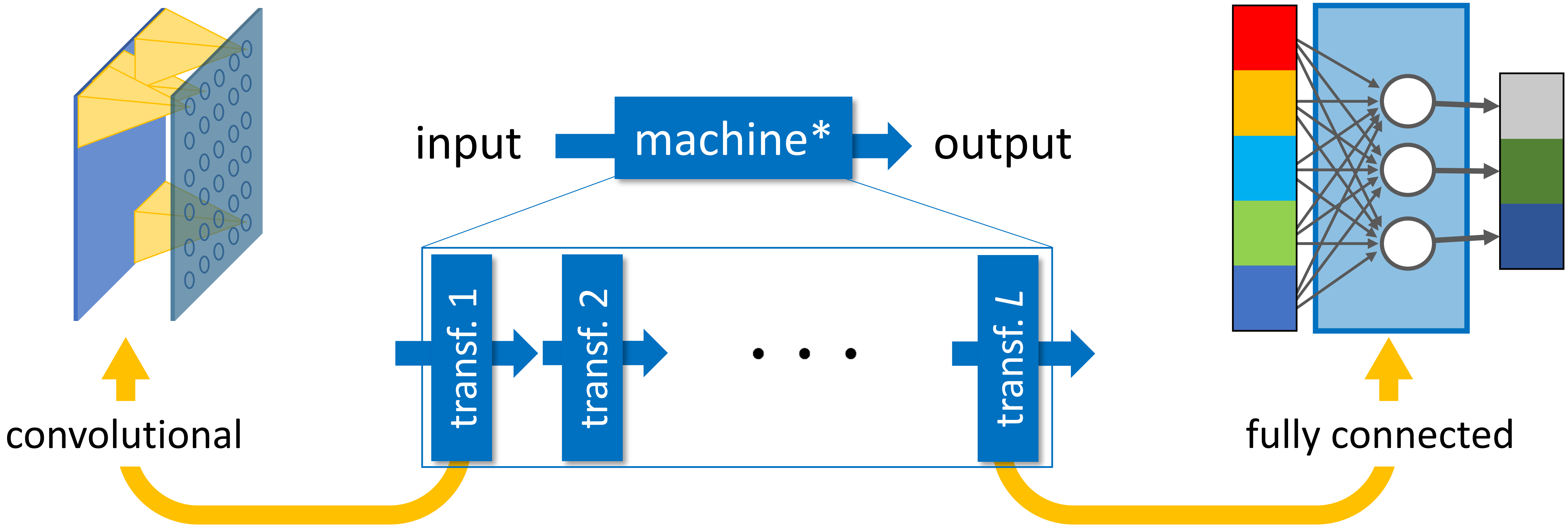
A simple example

- Example with two ad-hoc edge filters



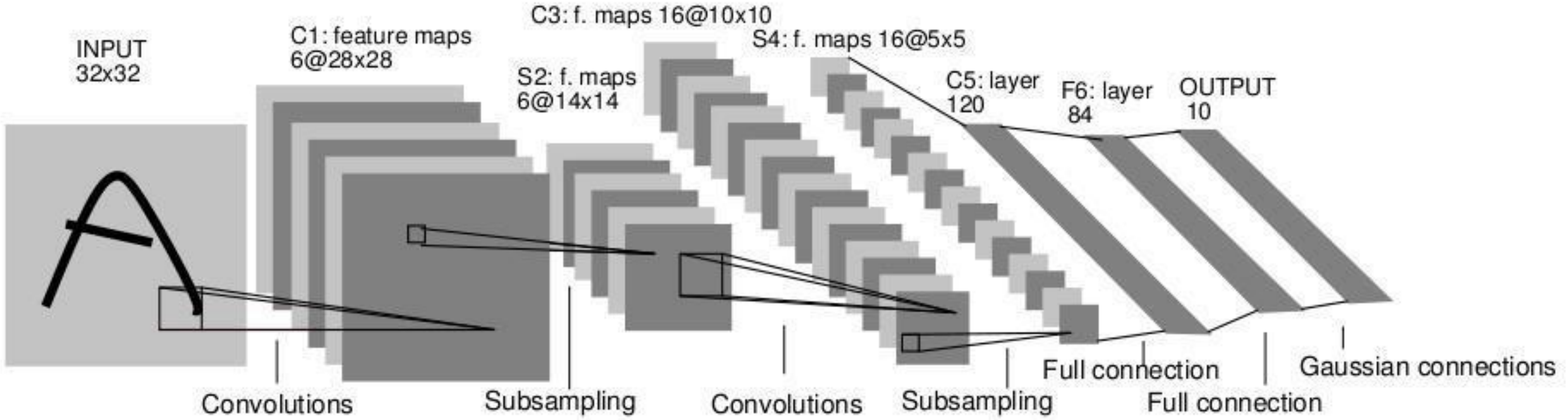
[fig by Ranzato]

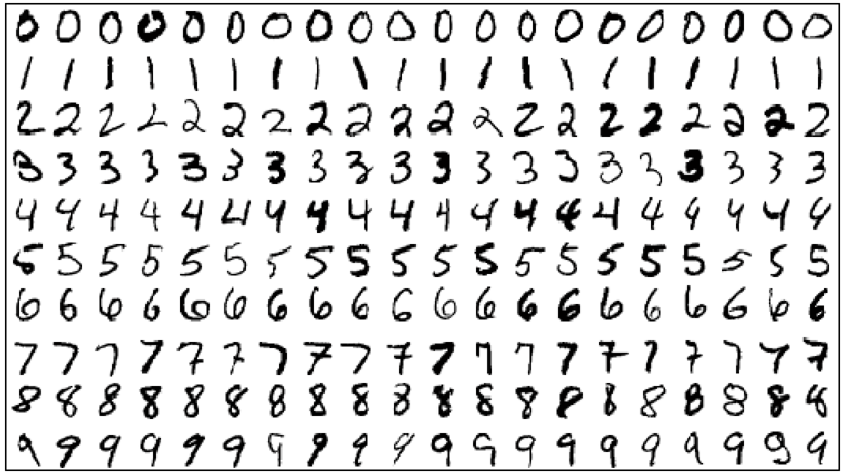
Convolutional networks



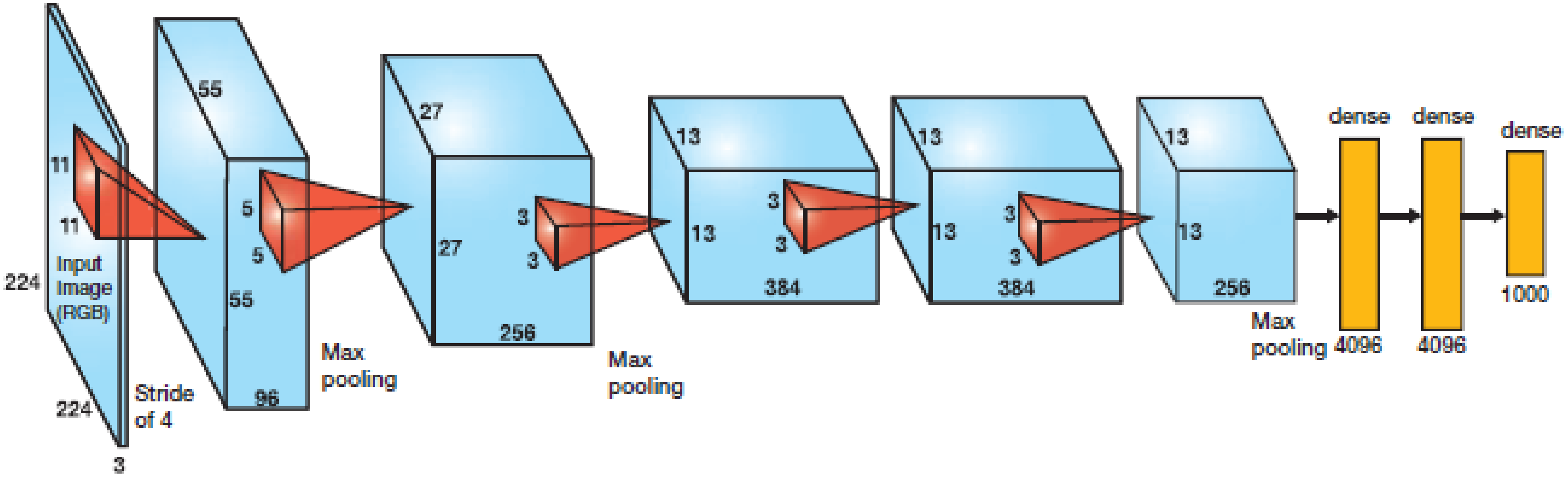
- Leverage spatial stationarity
- Share parameters
- Key for spatial data (inc. images at large)

LeCun 1998: "LeNet5"

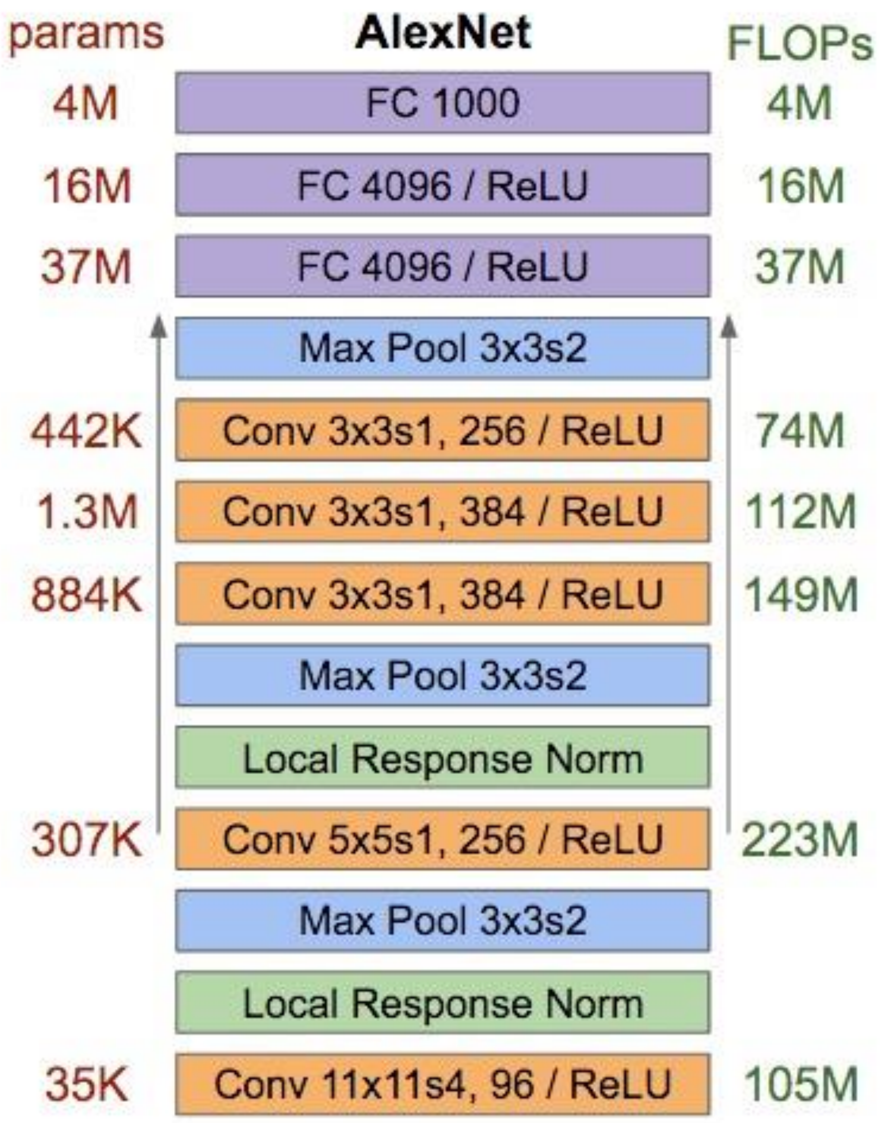




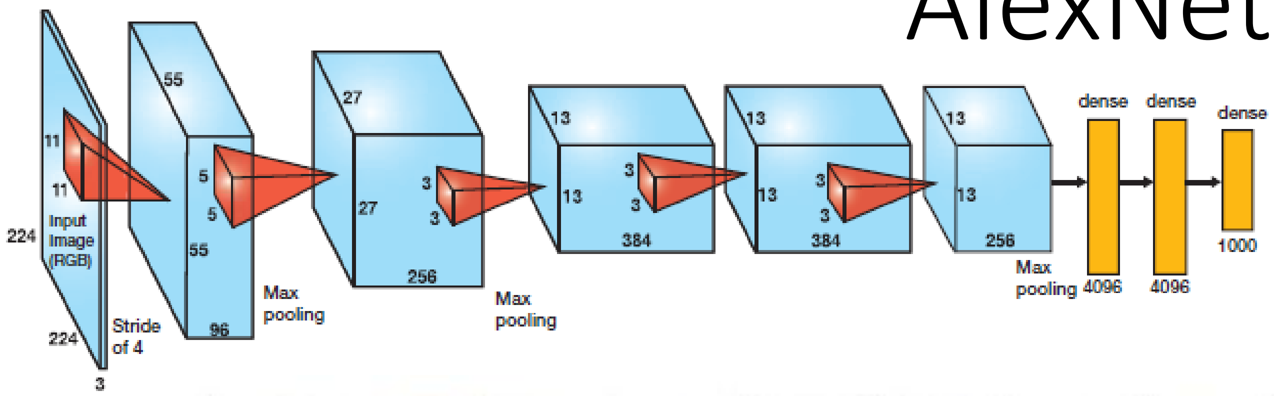
Krizhevsky *et al*. 2012: "AlexNet"



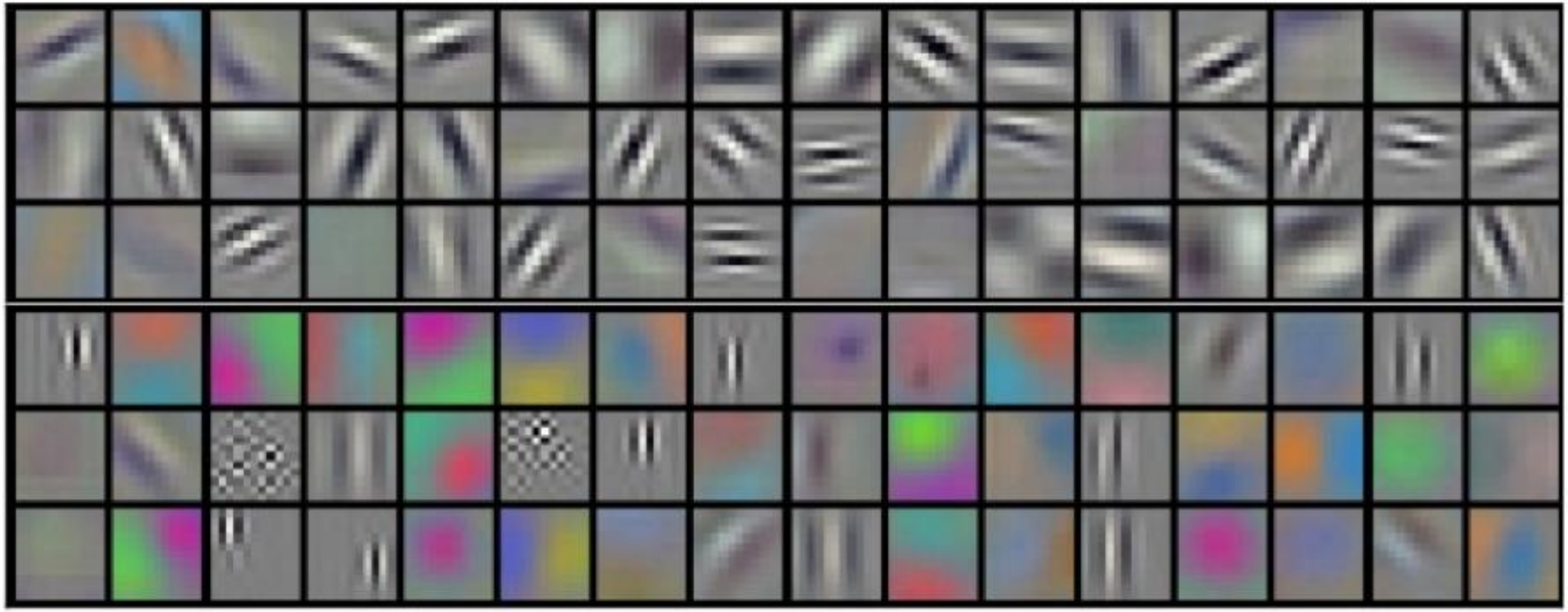
Krizhevsky *et al*. 2012: ”AlexNet”



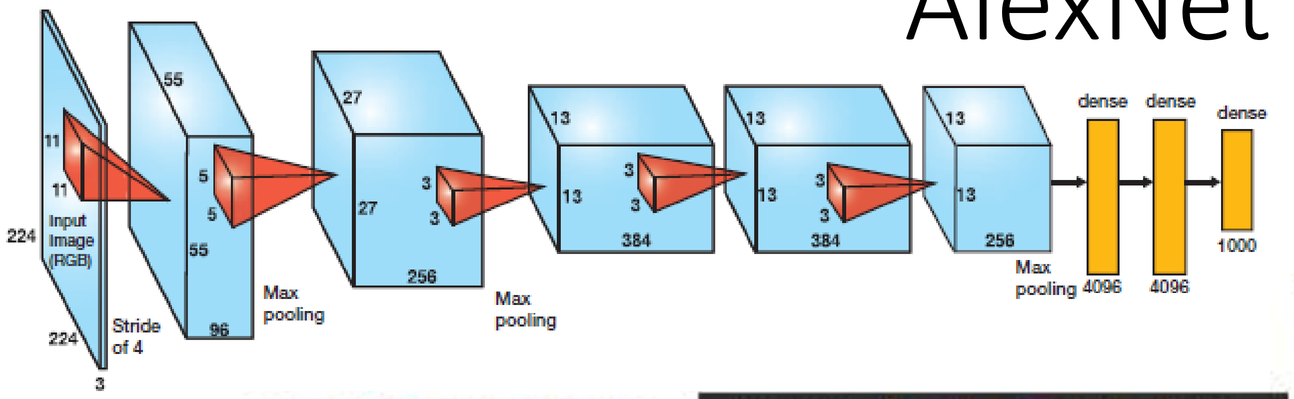
AlexNet's filters



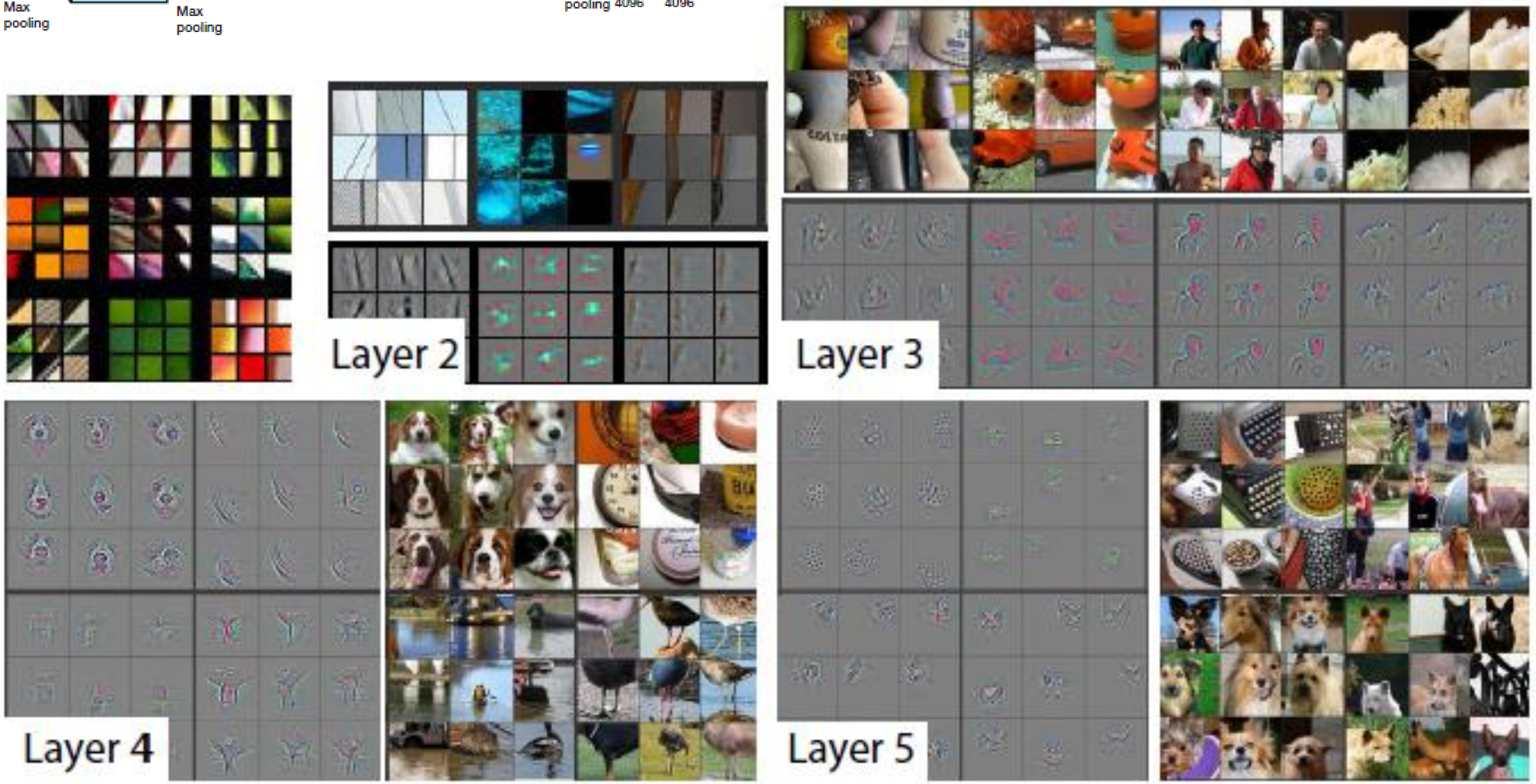
[Zeiller 2013]



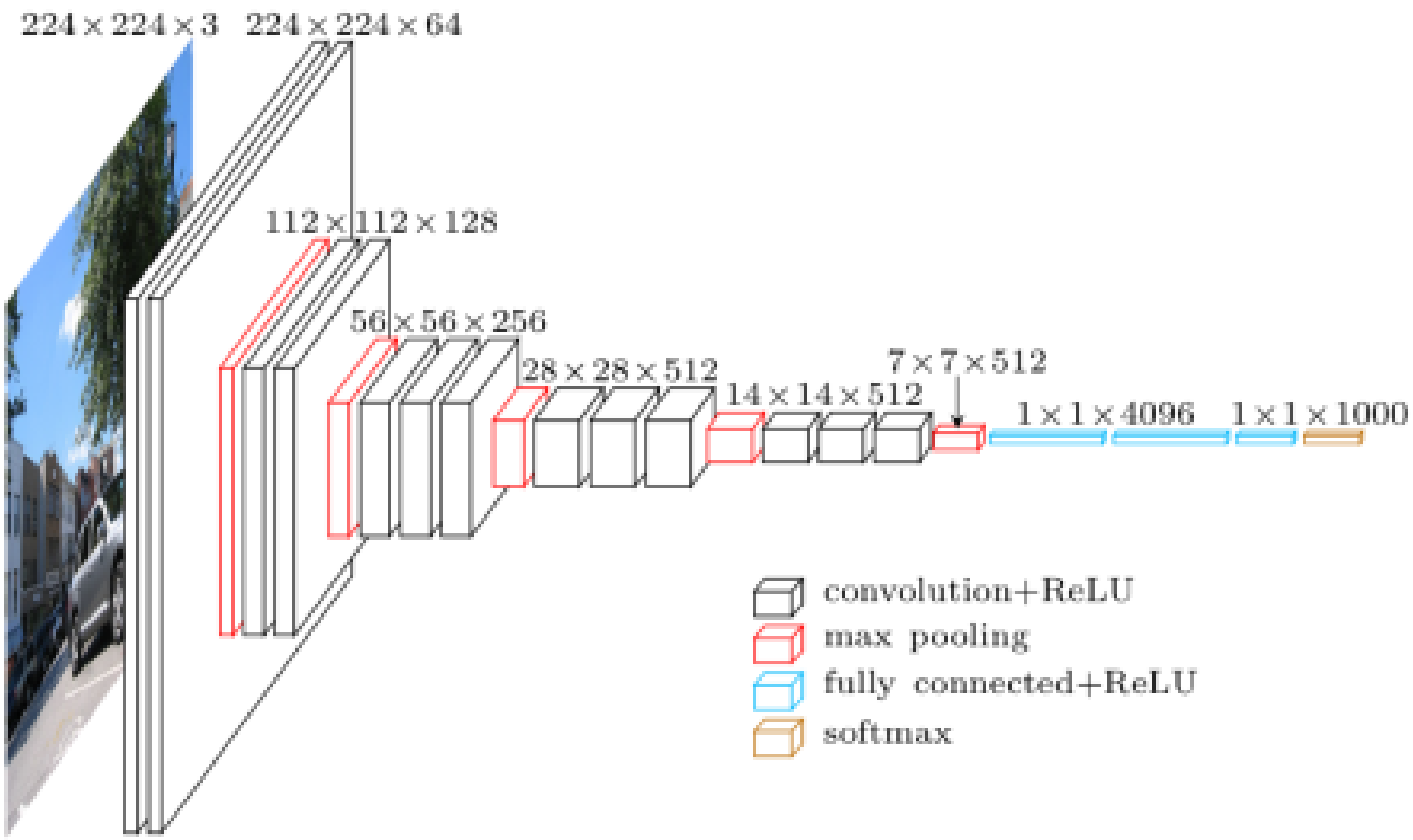
AlexNet's filters



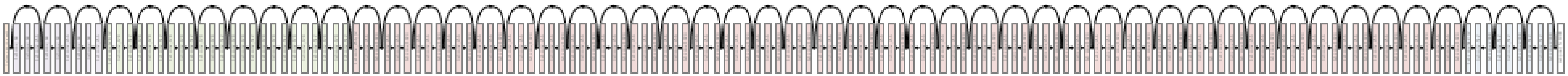
[Zeiller 2013]

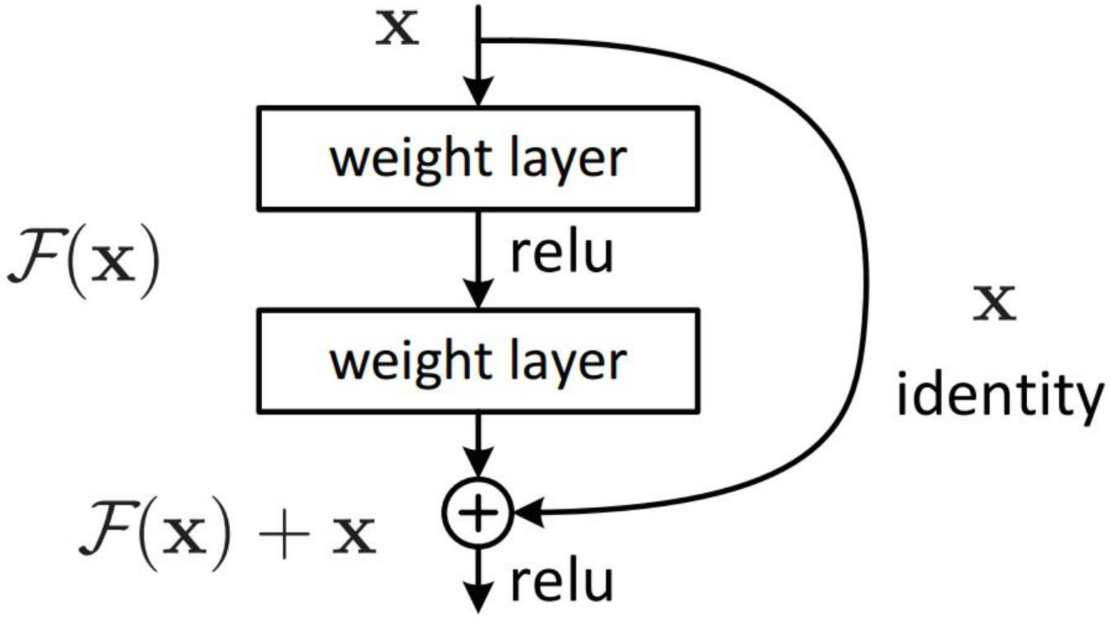


Simonyan *et al*. 2014: ”VGG”

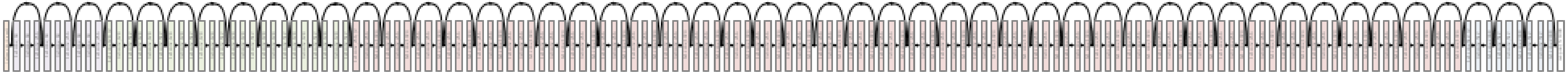


He *et al.* 2014: “ResNet”

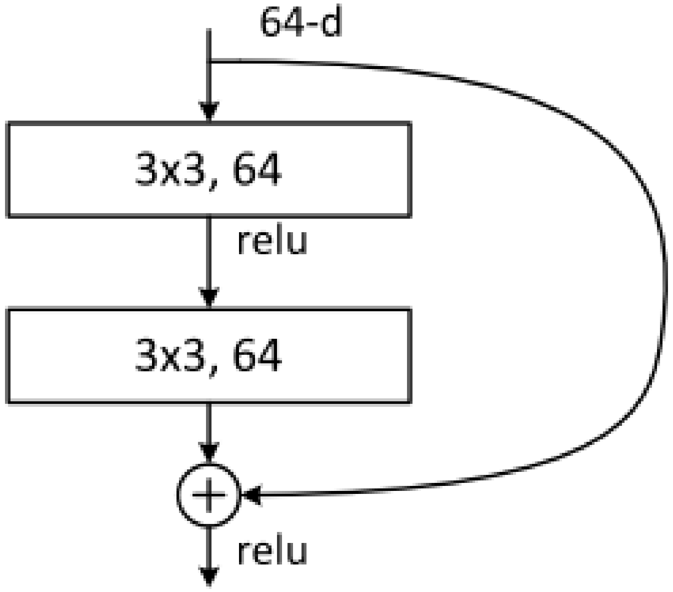


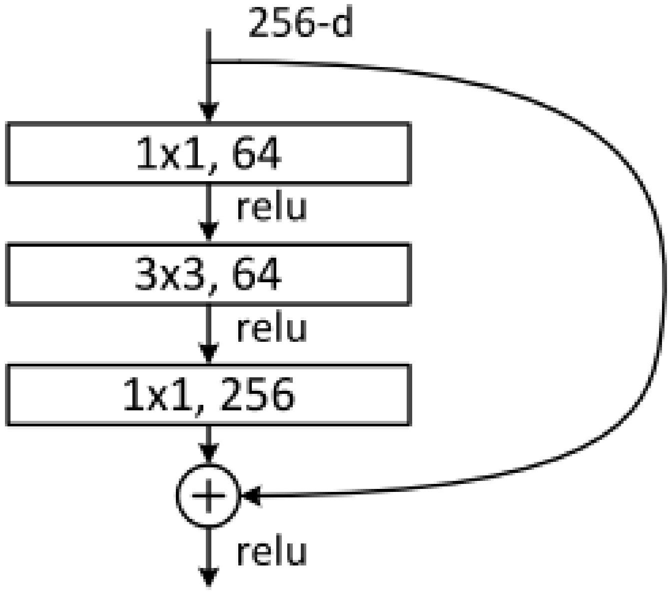


He *et al.* 2014: “ResNet”

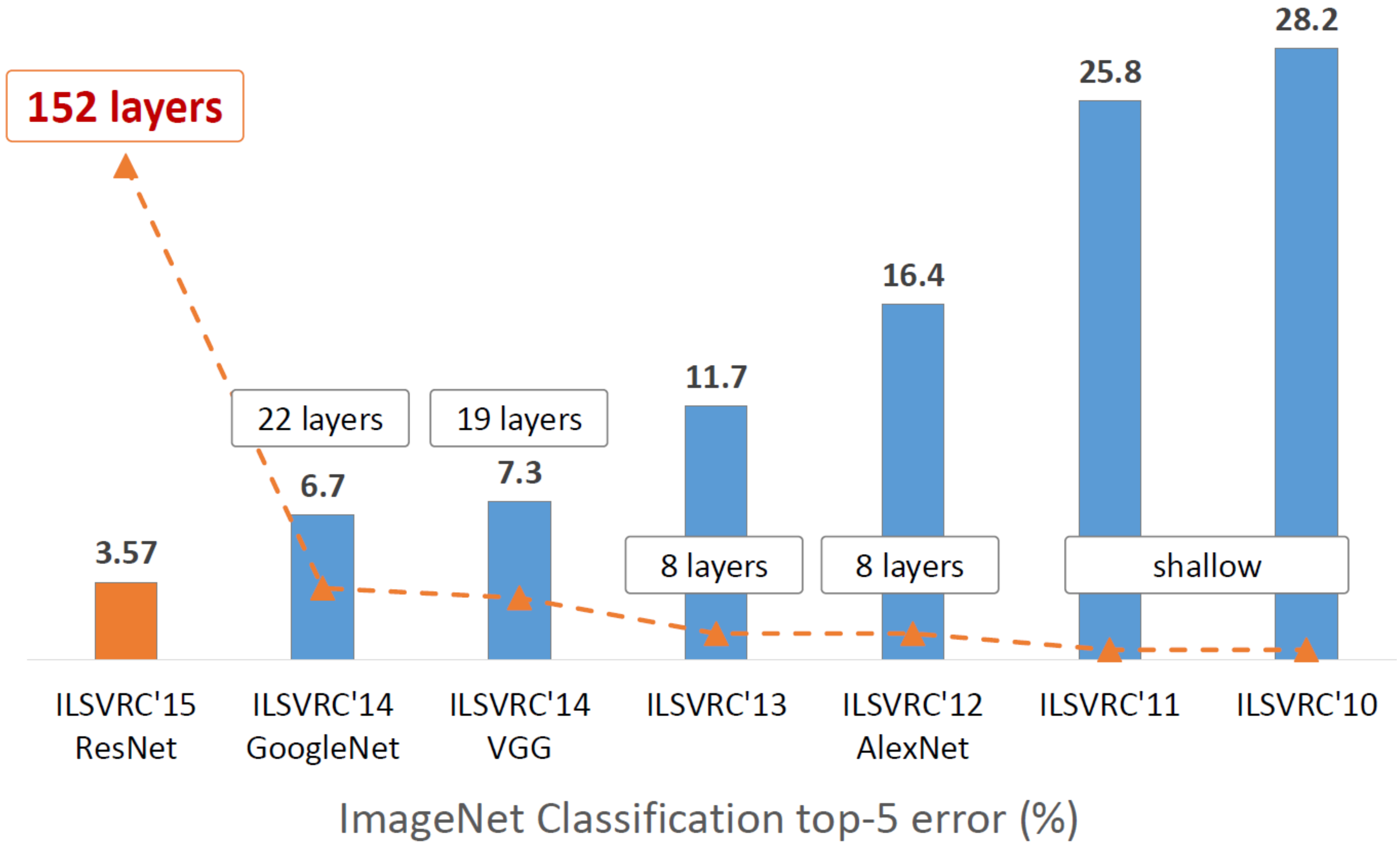


Residual blocks



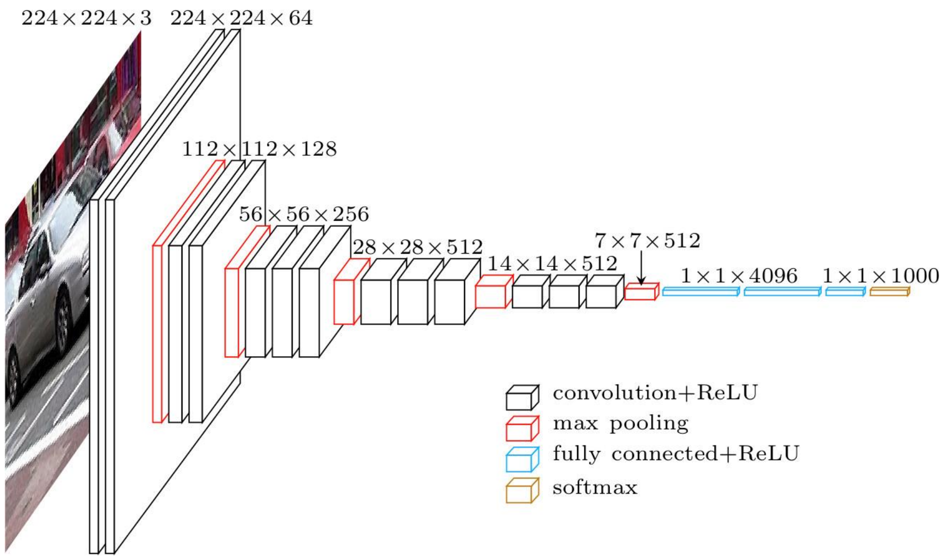


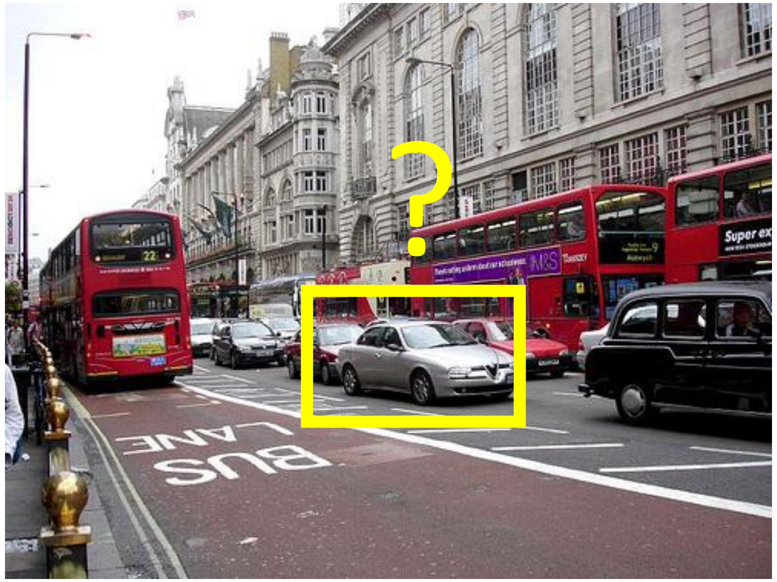
Deeper and deeper



Larger or more complex scenes

- Training on ImageNet: recognize fixed-sized cropped objects
- Extend to:
  - Different input image size?
  - More complex scenes with multiple images?
  - Object localization?

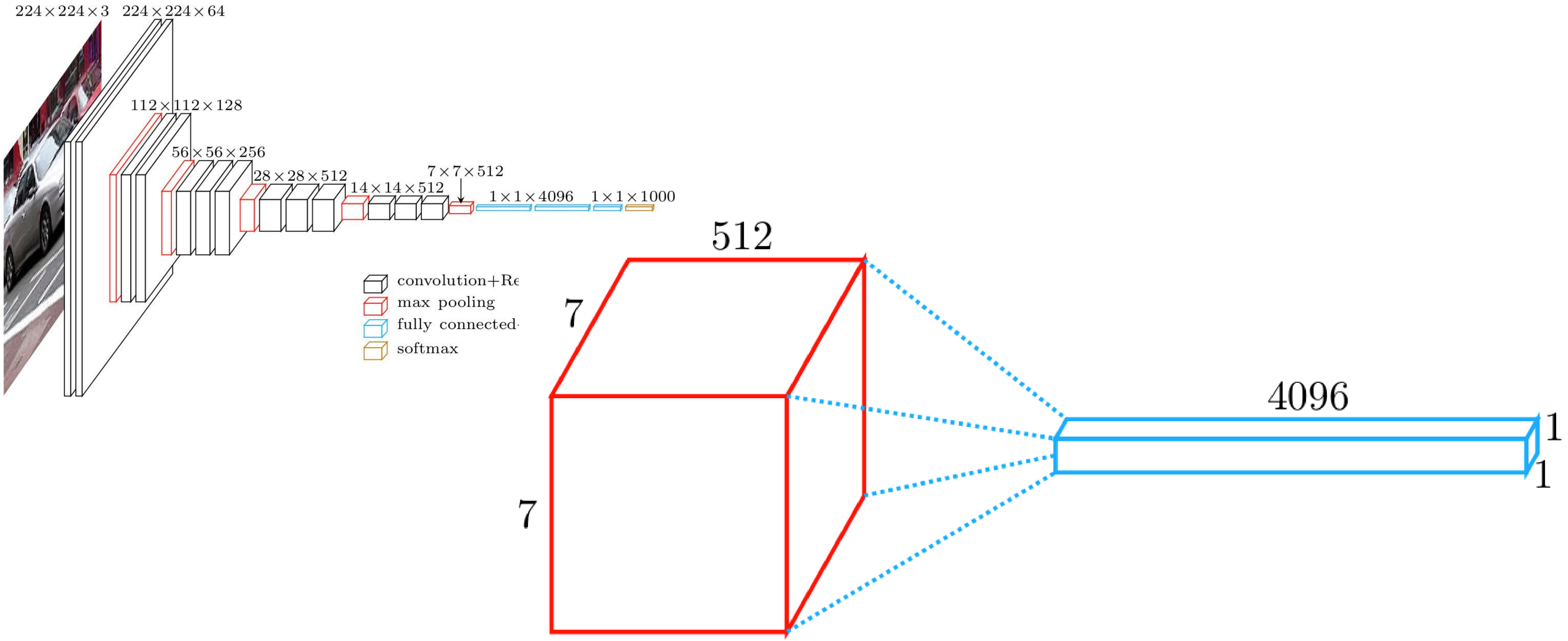




[fig by Cord]

Fully convolutional nets

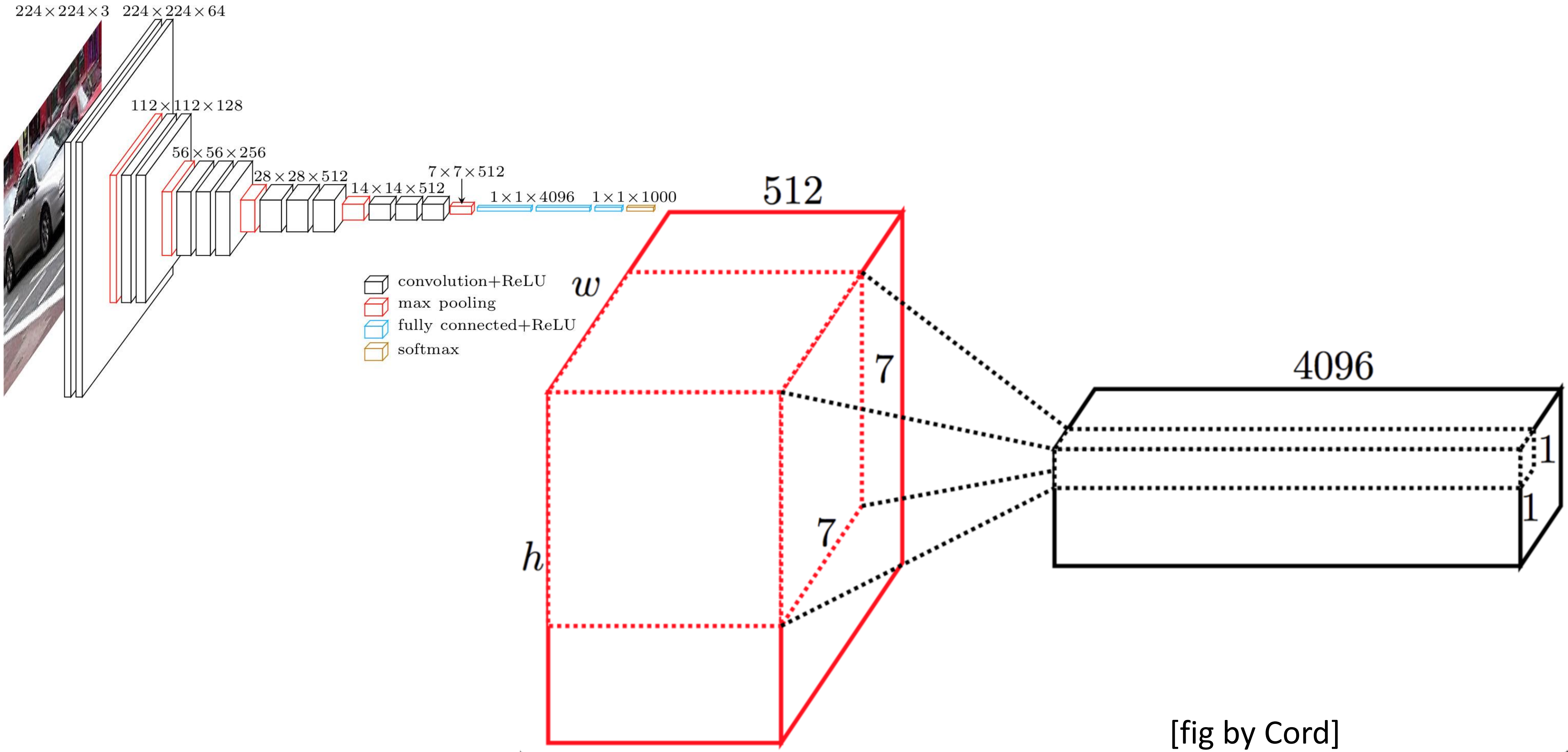
- Sliding window: Variable size input & loosely spatialized output



[fig by Cord]

Fully convolutional nets

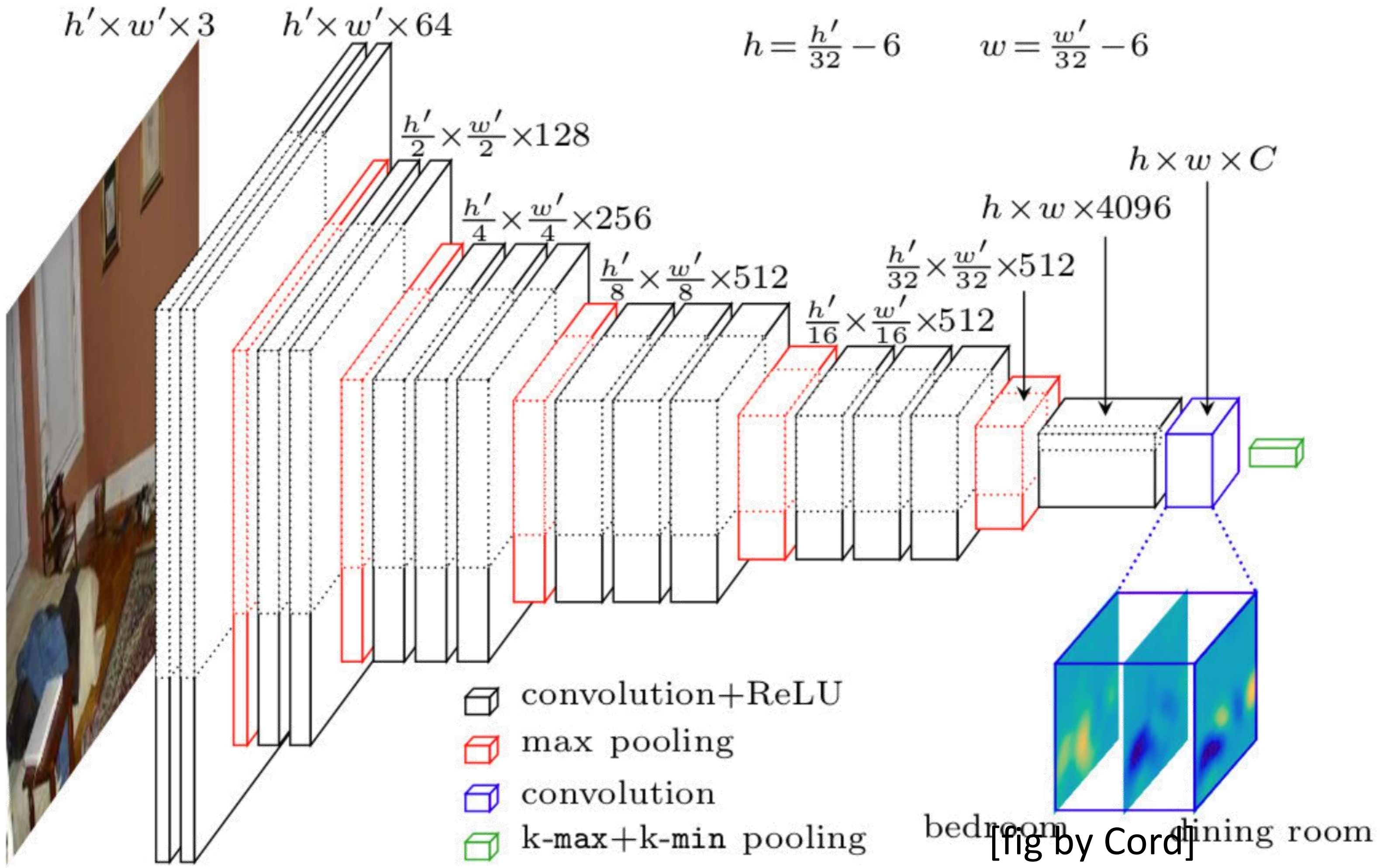
- Sliding window: FC as convolutions



[fig by Cord]

Feature map pooling

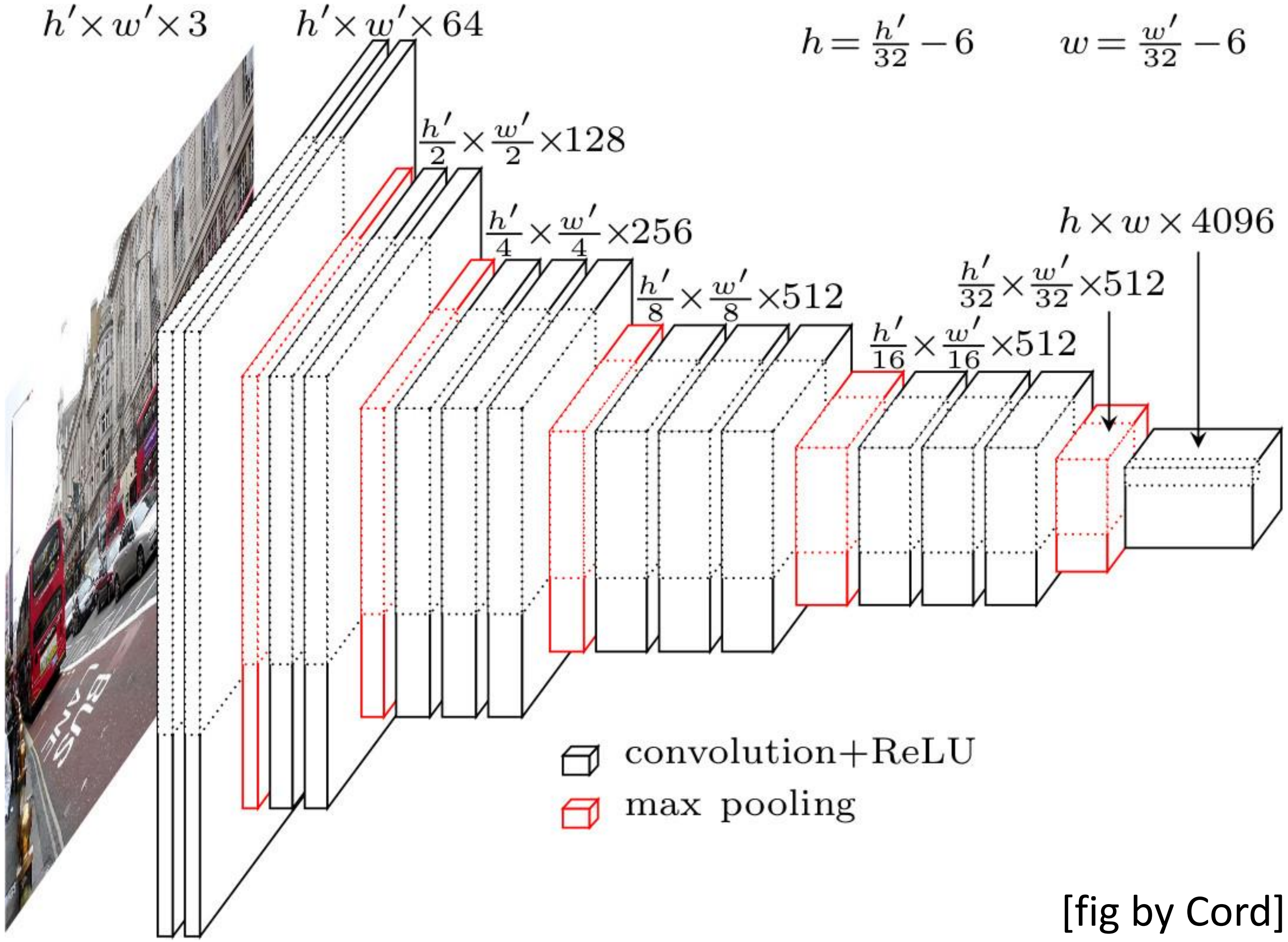
- Variable size input & loosely spatialized output



[fig by Cord]

Fully convolutional nets

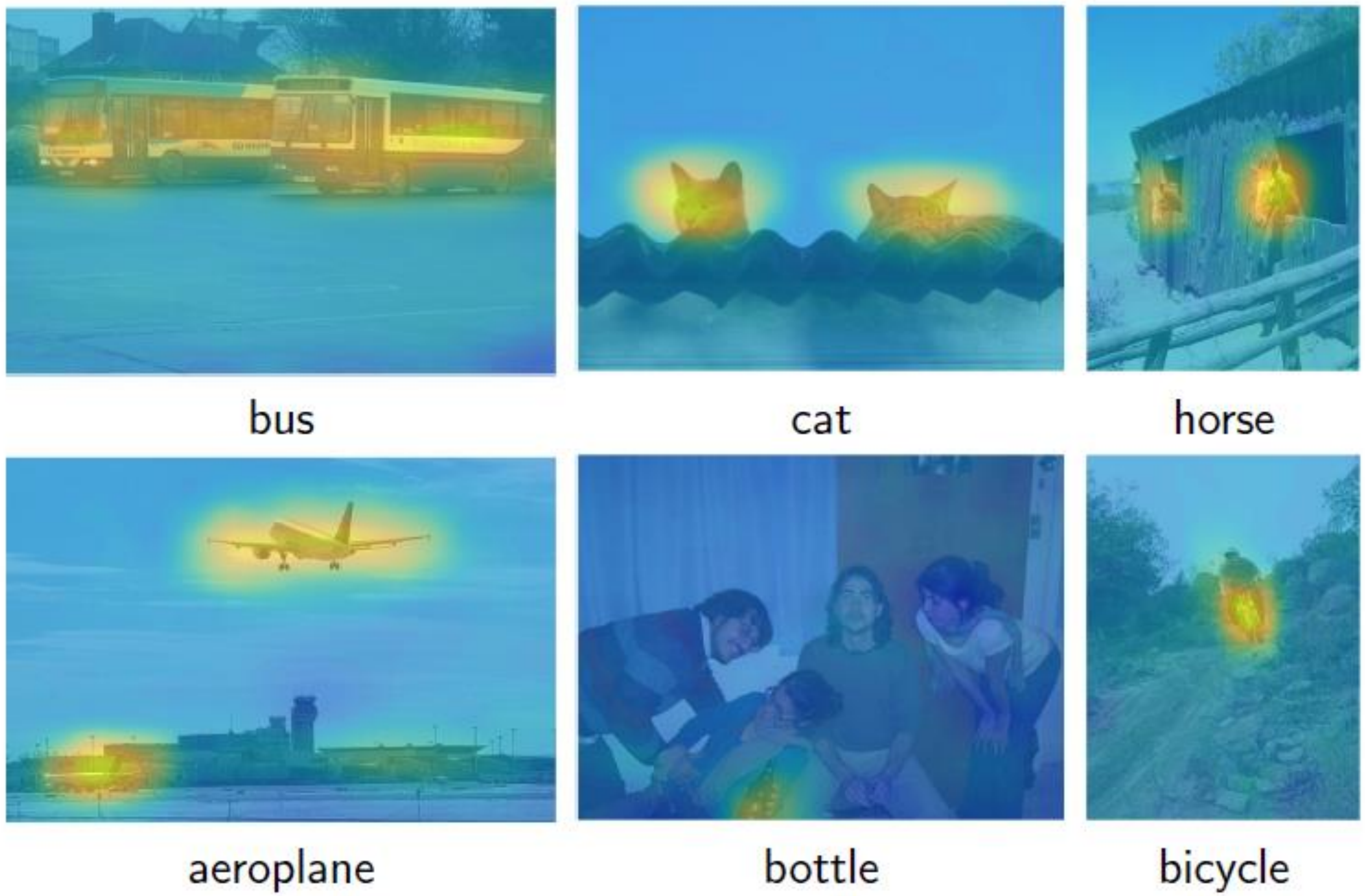
- Variable size input & loosely spatialized output



[fig by Cord]

Class activation maps

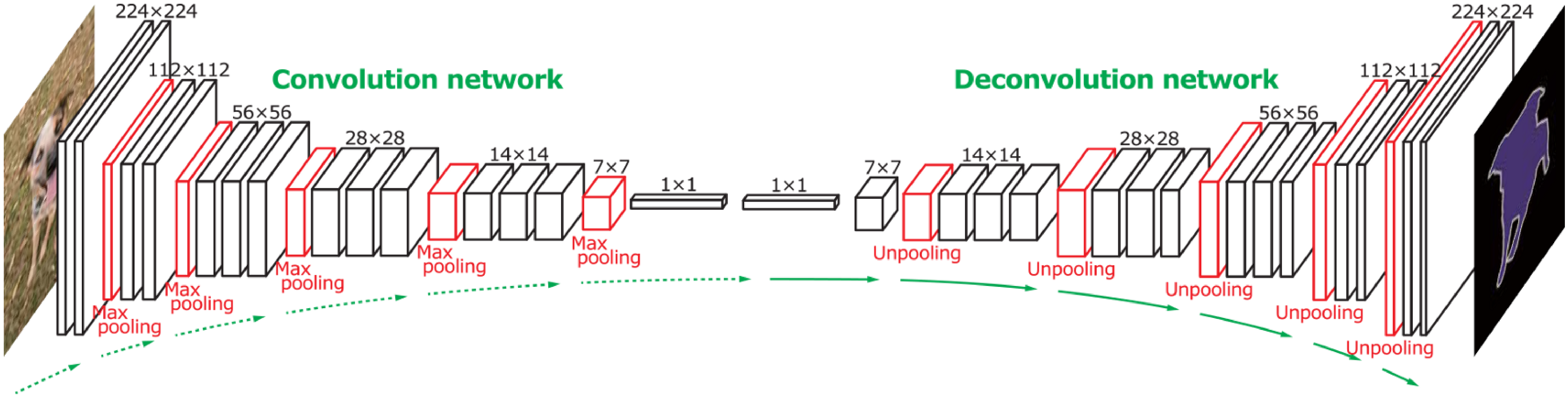
- Towards localization and (low-res) semantic segmentation



[fig by Cord]

Getting spatial resolution back

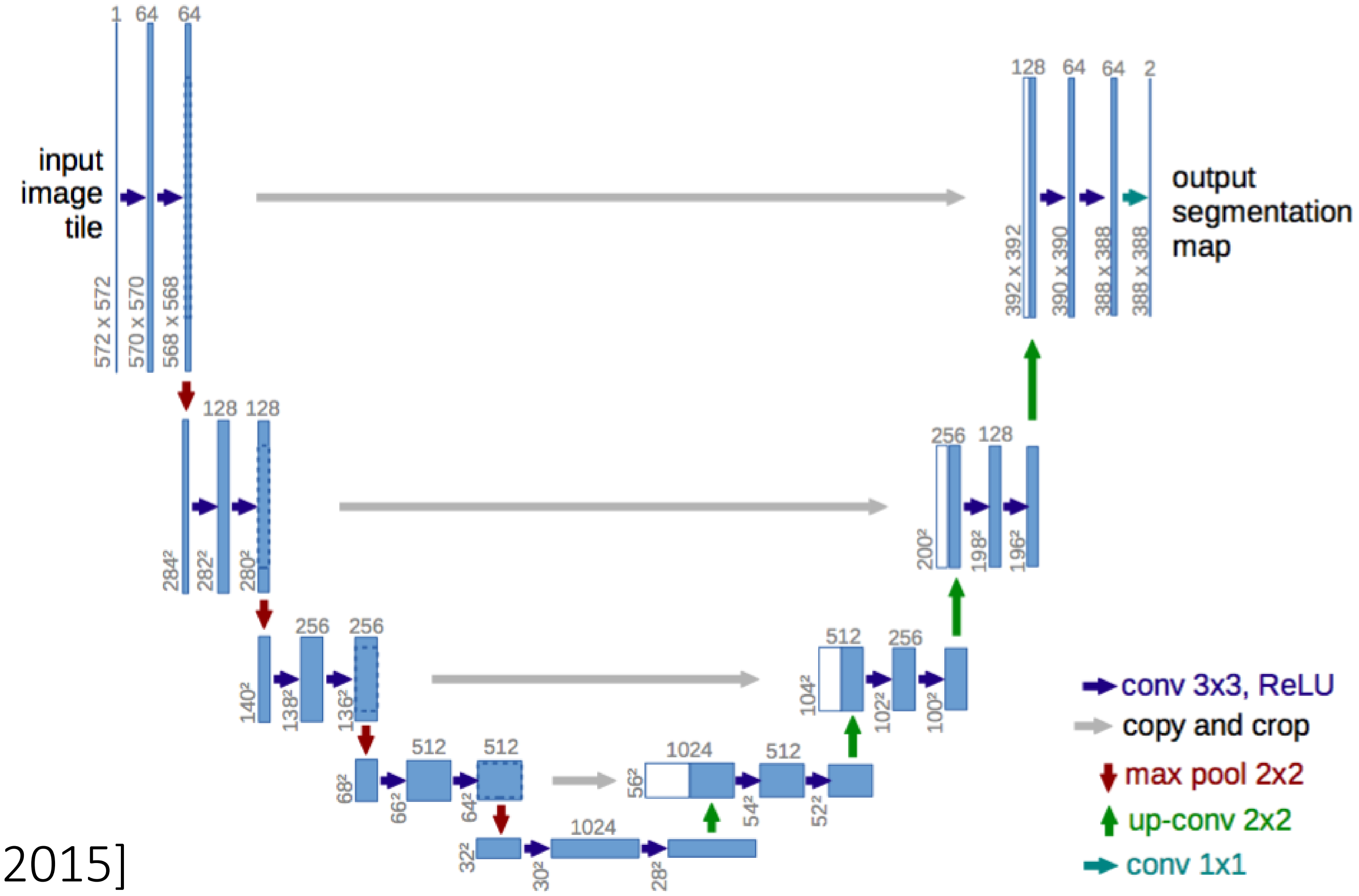
- Mirroring ConvNet: unpooling and convolution
- Upsampling and convolution
- Fractional-stride convolution



[Noh *et al.* 2015]

U-Net and skip-connections

- Passing features maps across through stacking



[Ronneberger 2015]

Recurrent Neural Nets

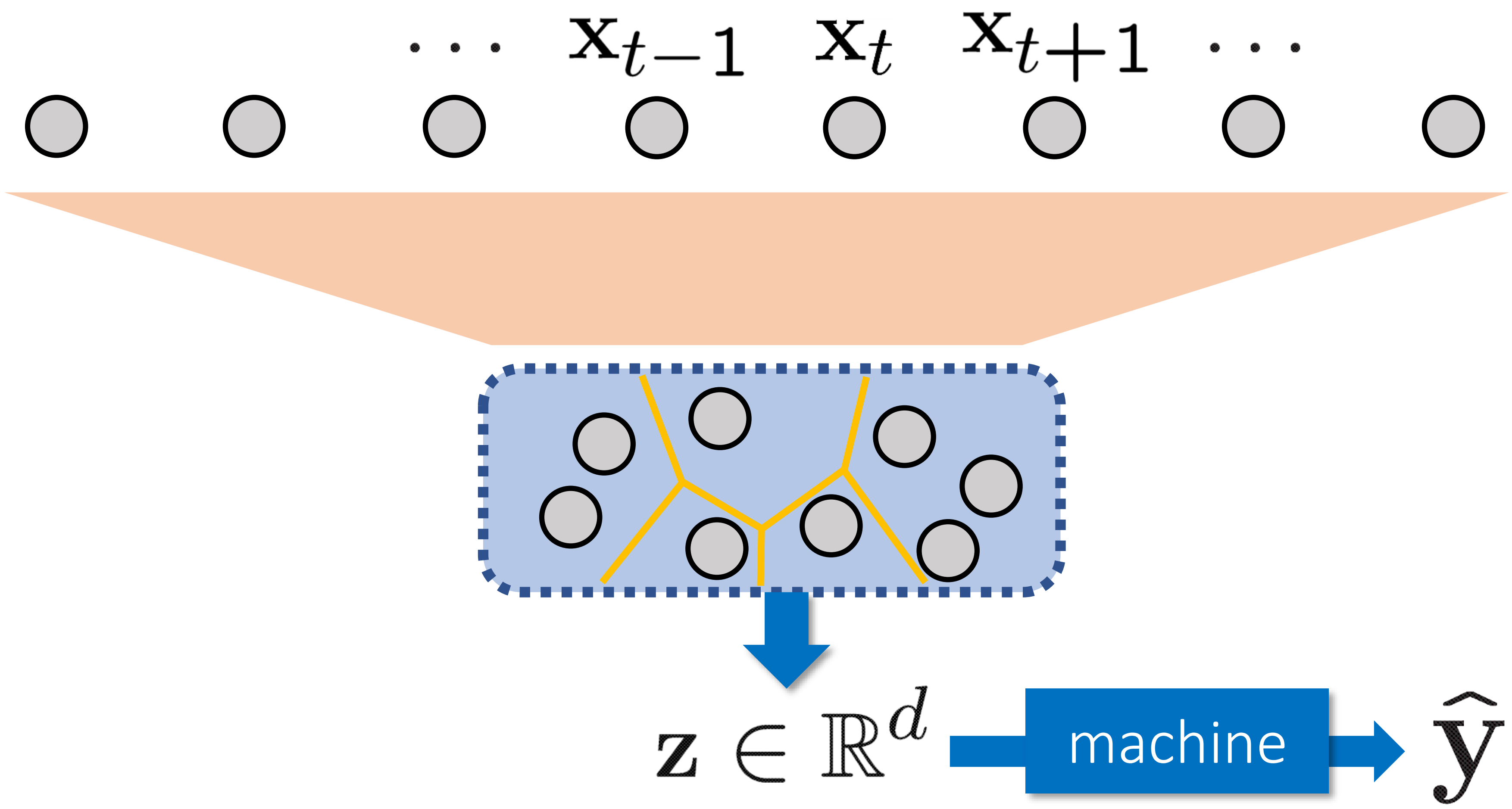
Sequential data

Data with natural total ordering, and variable length

- Text
- Speech
- Audio
- Video
- Animation
- EEG, body signals
- DNA sequences
- Streams and sequences at large

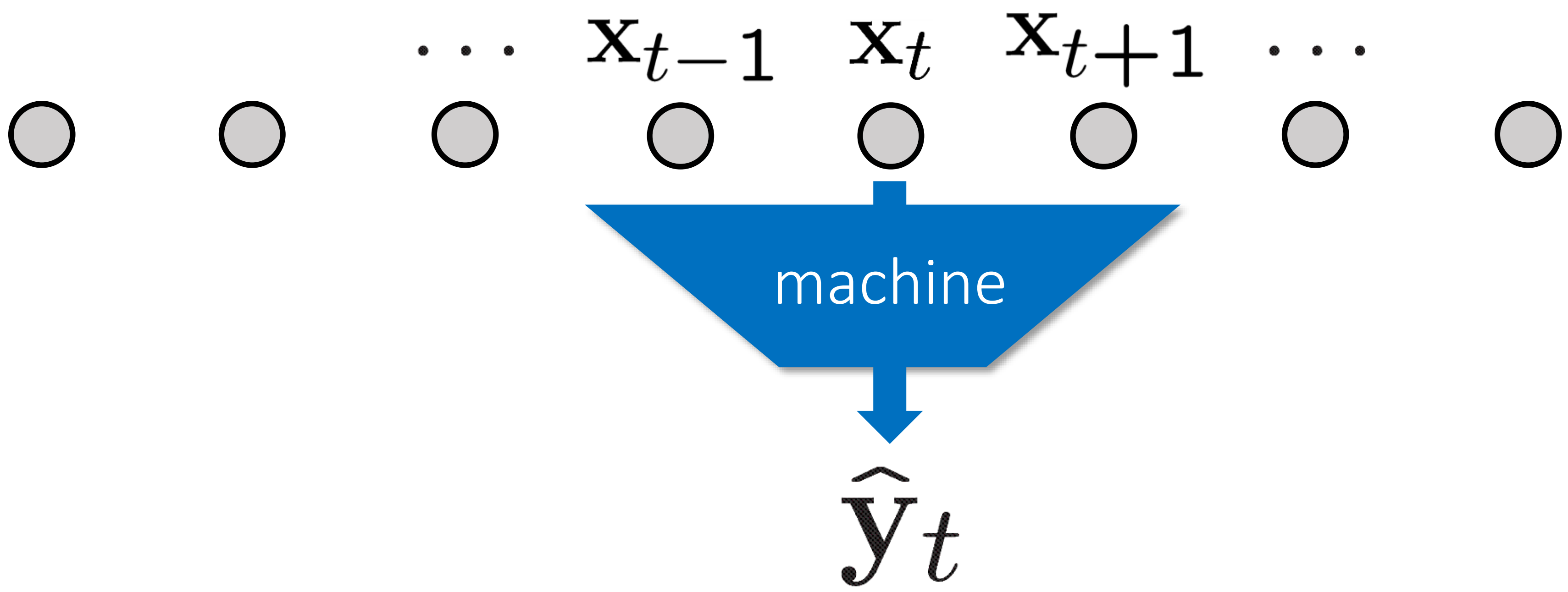
Handling sequences?

Order-less pooling, e.g. "bag-of-words"



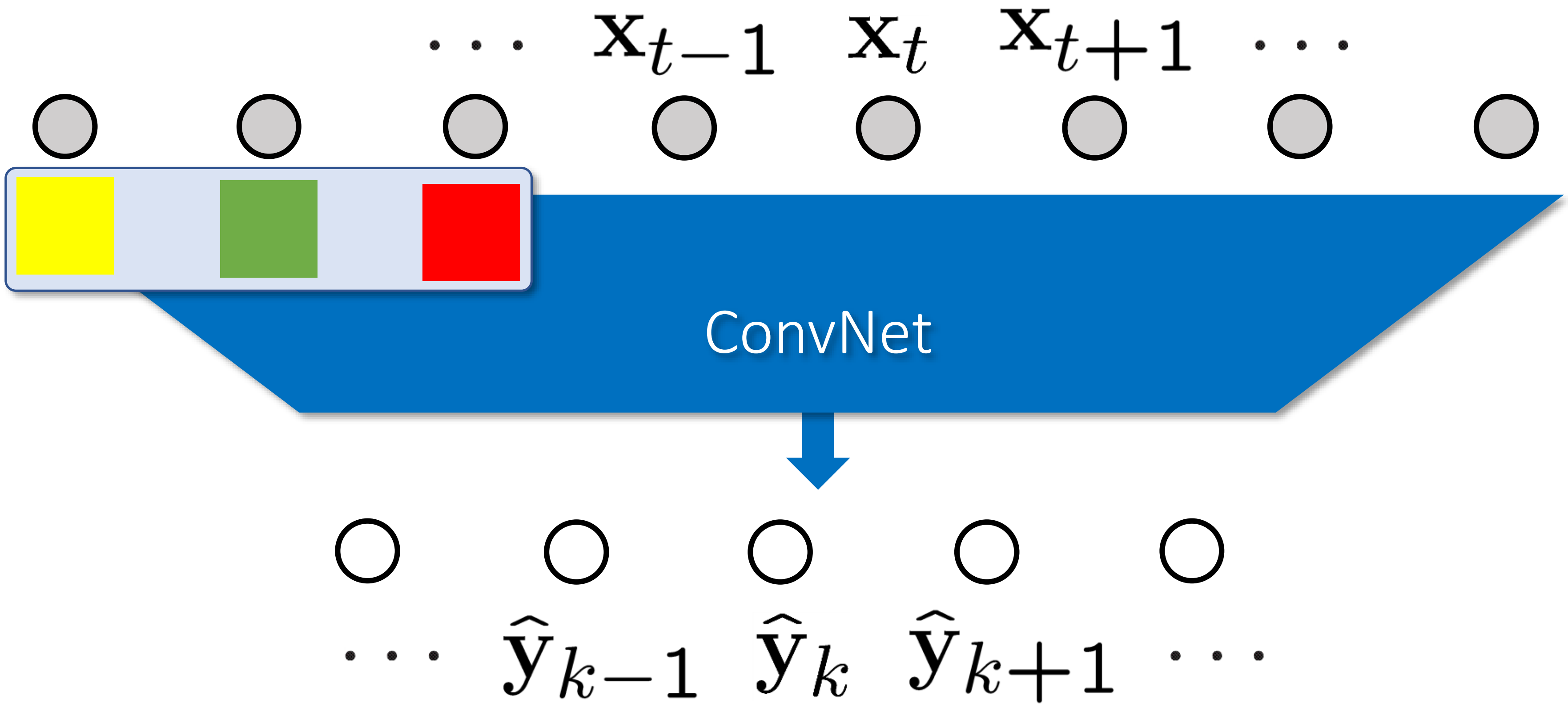
Handling sequences?

Sliding window (fixed-sized context), e.g., “word2vec”



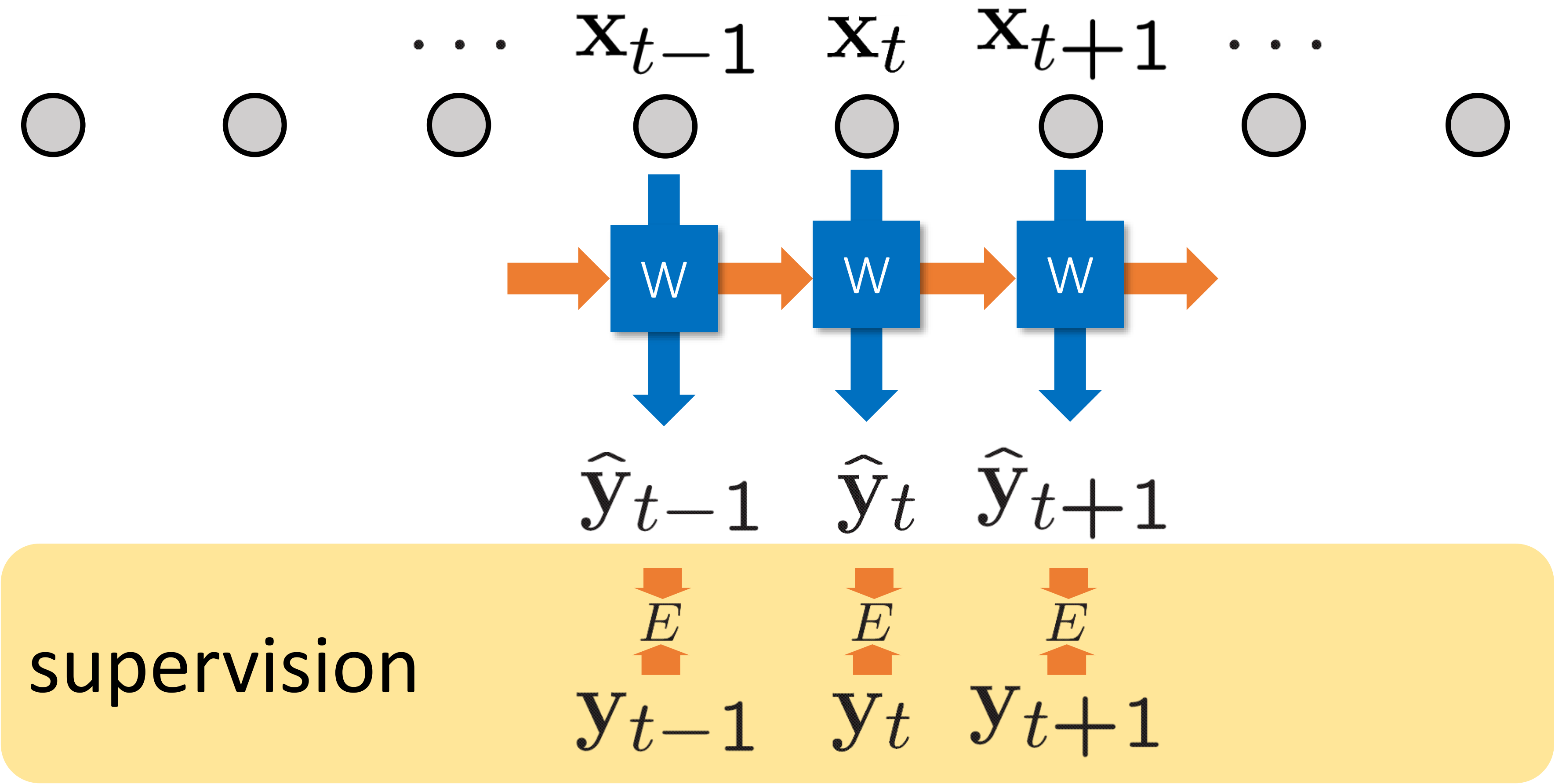
Handling sequences?

1D-convolution, e.g., Holden's motion embedding



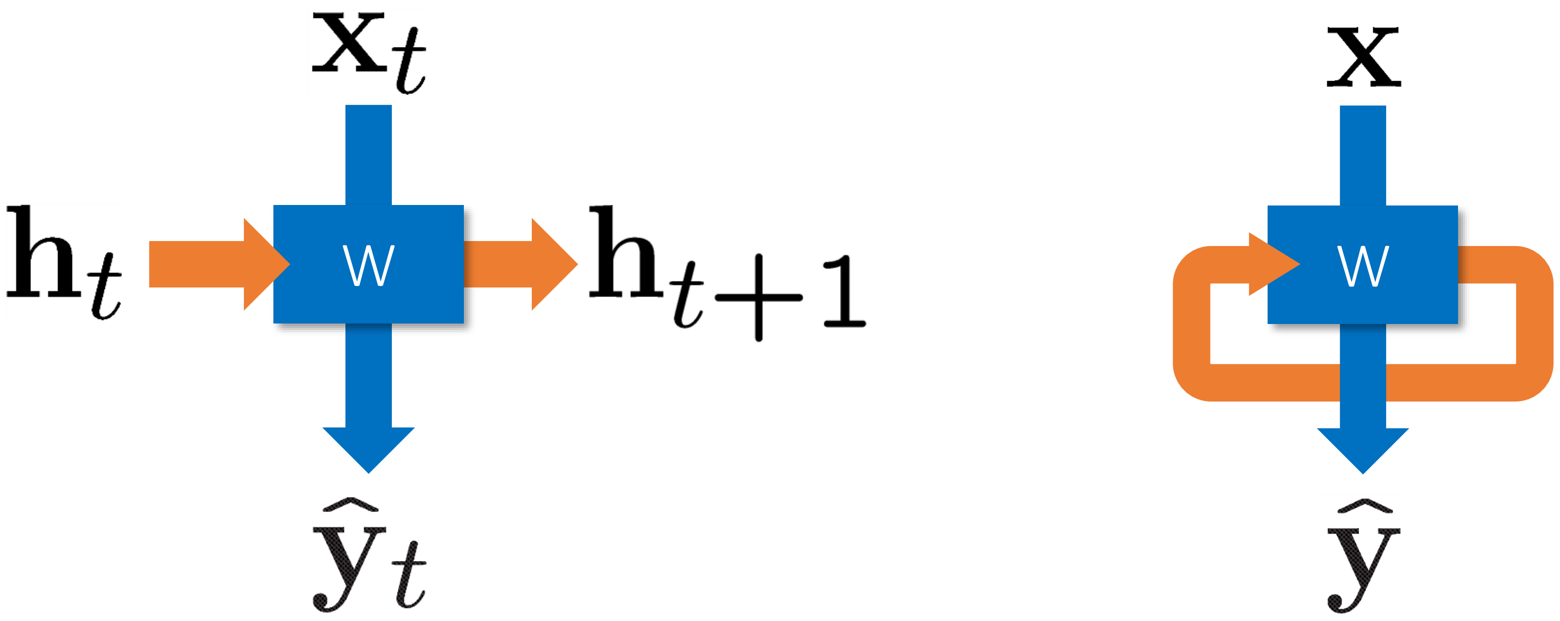
Handling sequences?

Recurrent machines, e.g., RNNs



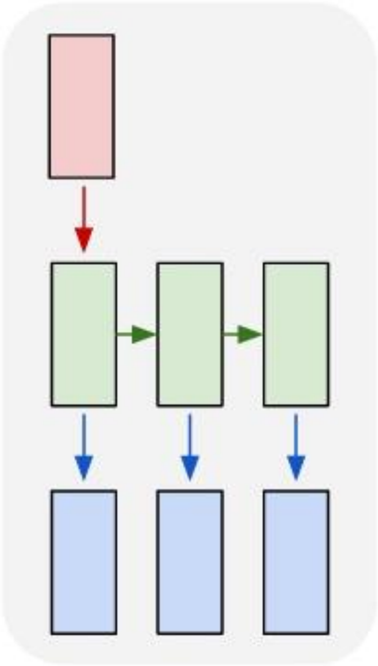
Recurrent Neural Nets (RNNs)

- Recurrent link through a dedicated “hidden state”
- Captures short term memory

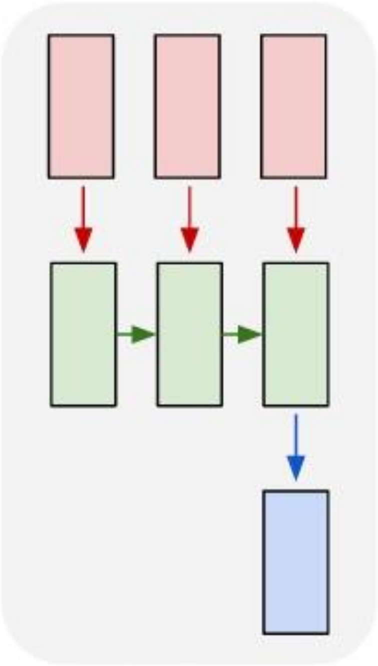


RNNs

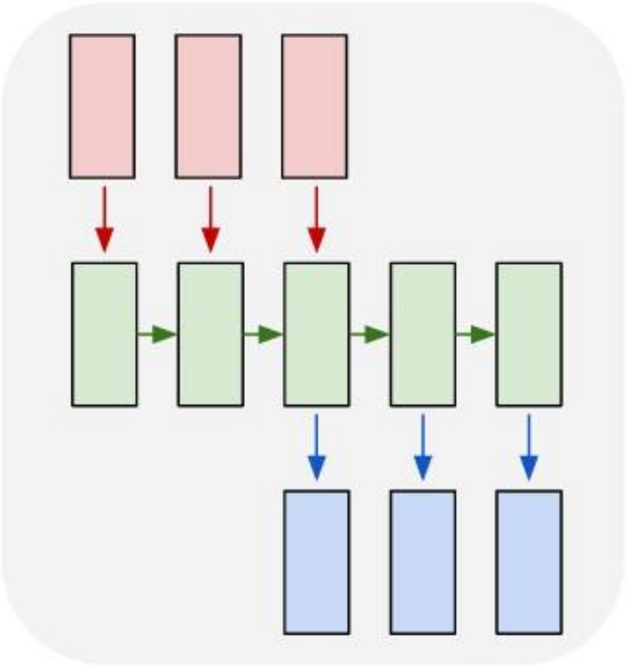
- Variations



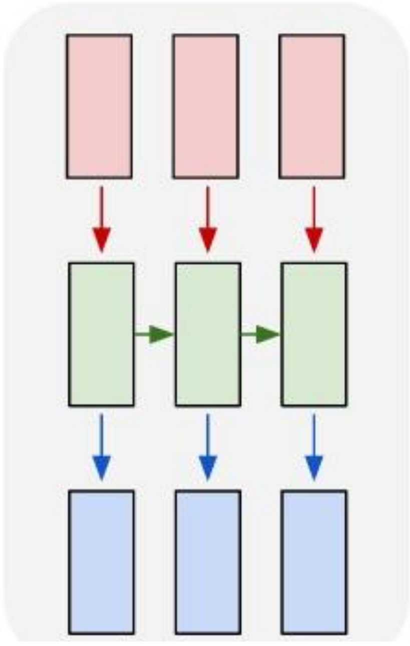
one-to-many



many-to-one



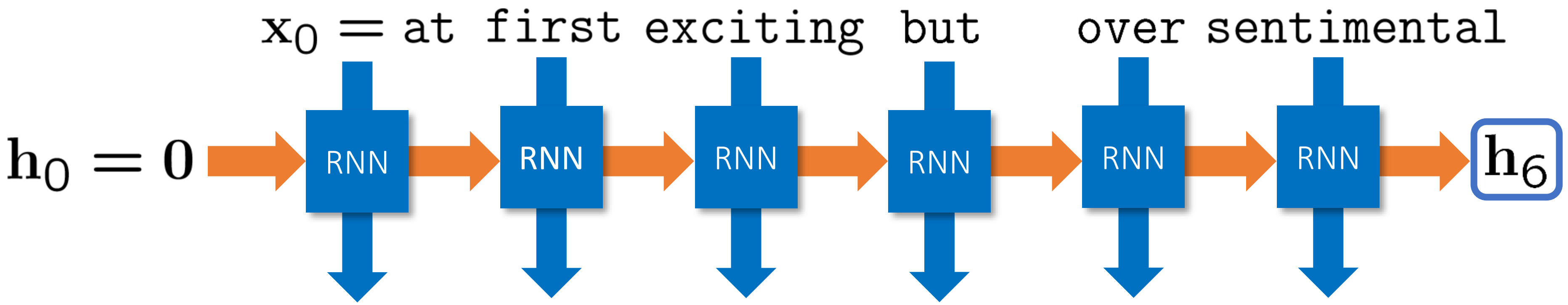
many-to-many



many-to-many
parallel

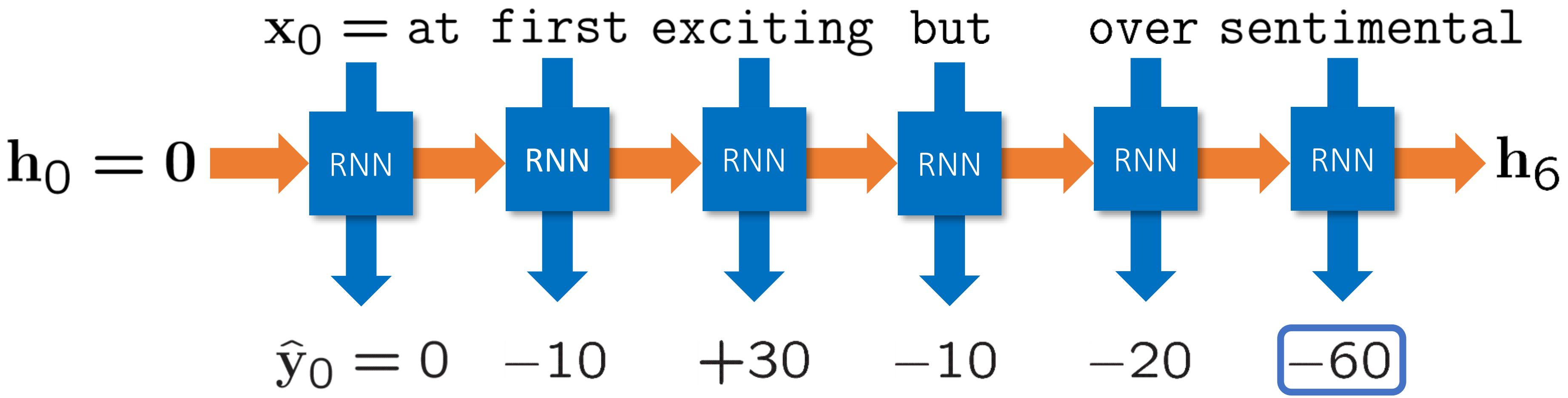
Many-to-one

- Sentence encoding (embedding)



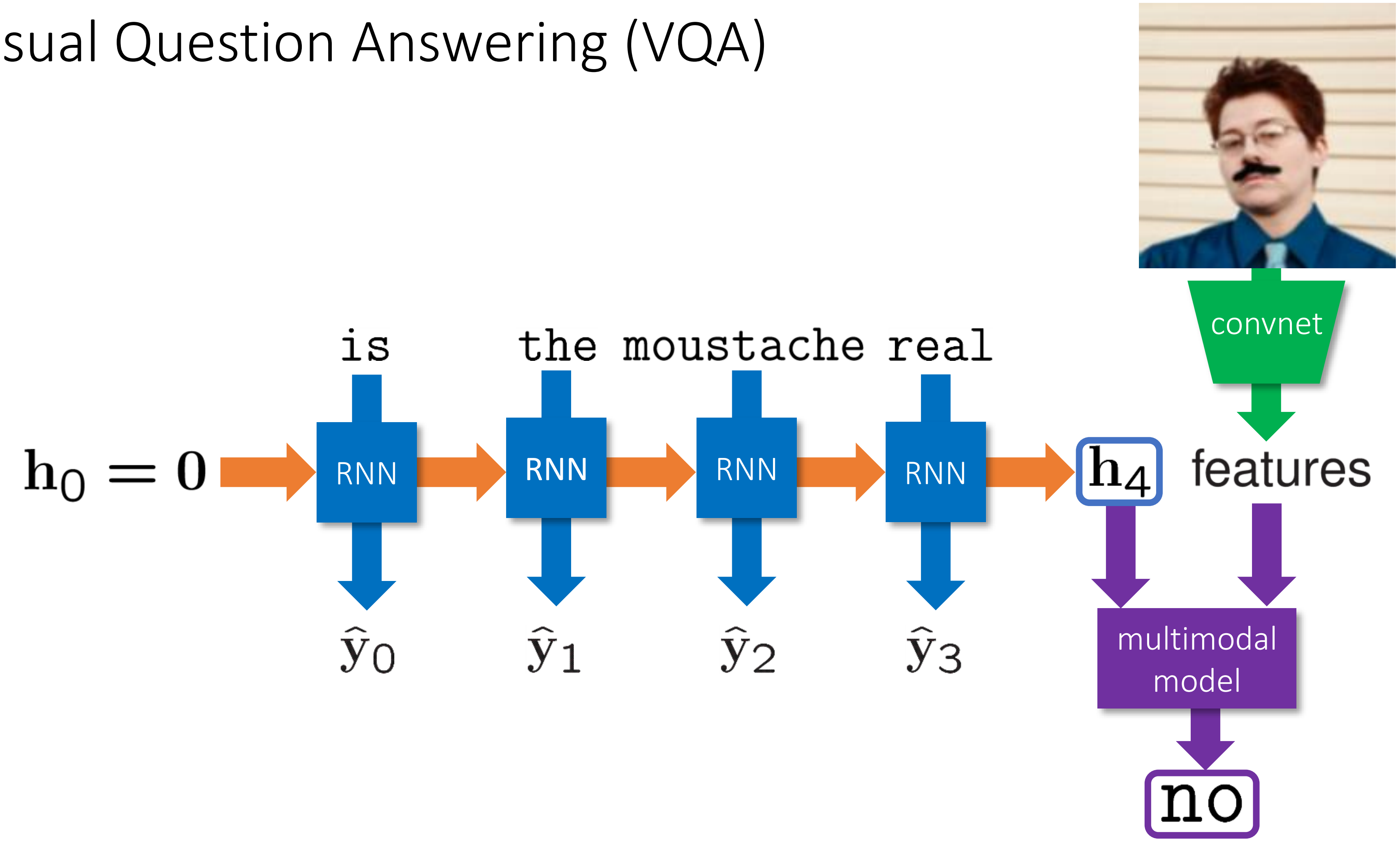
Many-to-one

- Sentiment analysis



Many-to-one

- Visual Question Answering (VQA)



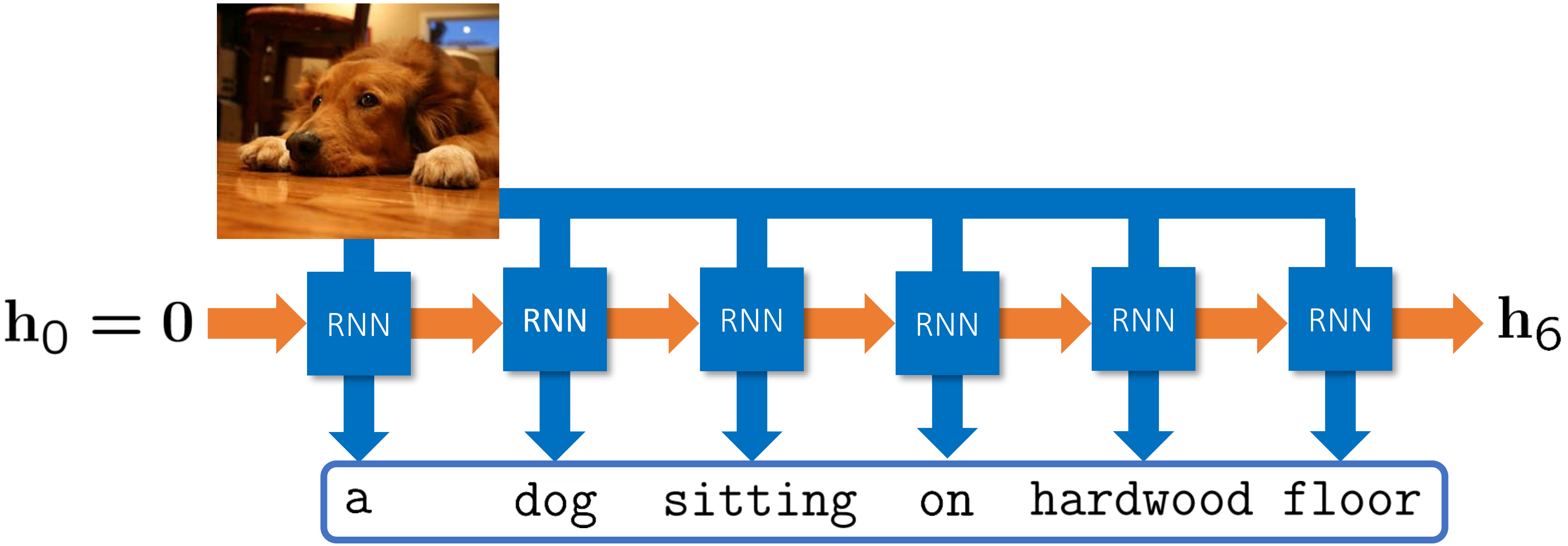
One-to-many

- Image captioning

[Xu et al. 2015]

One-to-many

- Image captioning

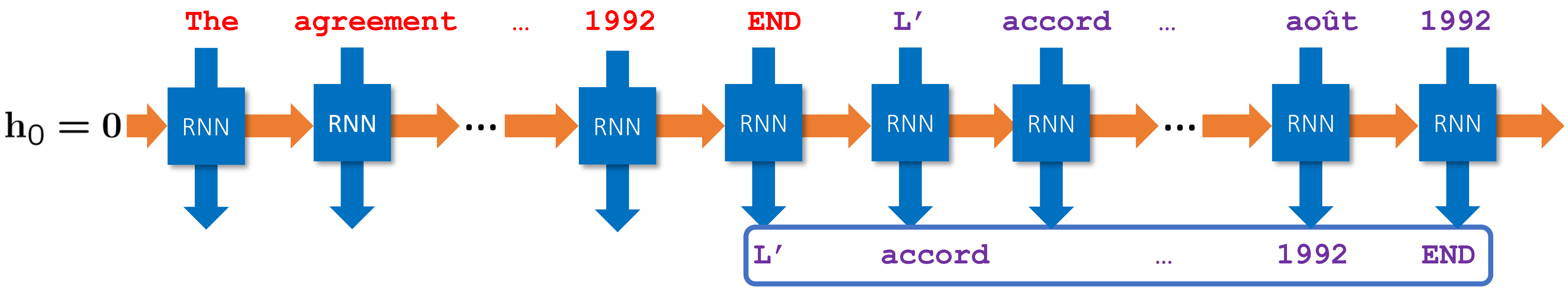


Many-to-many

- Machine translation text2text

"The agreement on the European Economic Area was signed in August 1992."

"L'accord sur la zone économique européenne a été signé en août 1992."

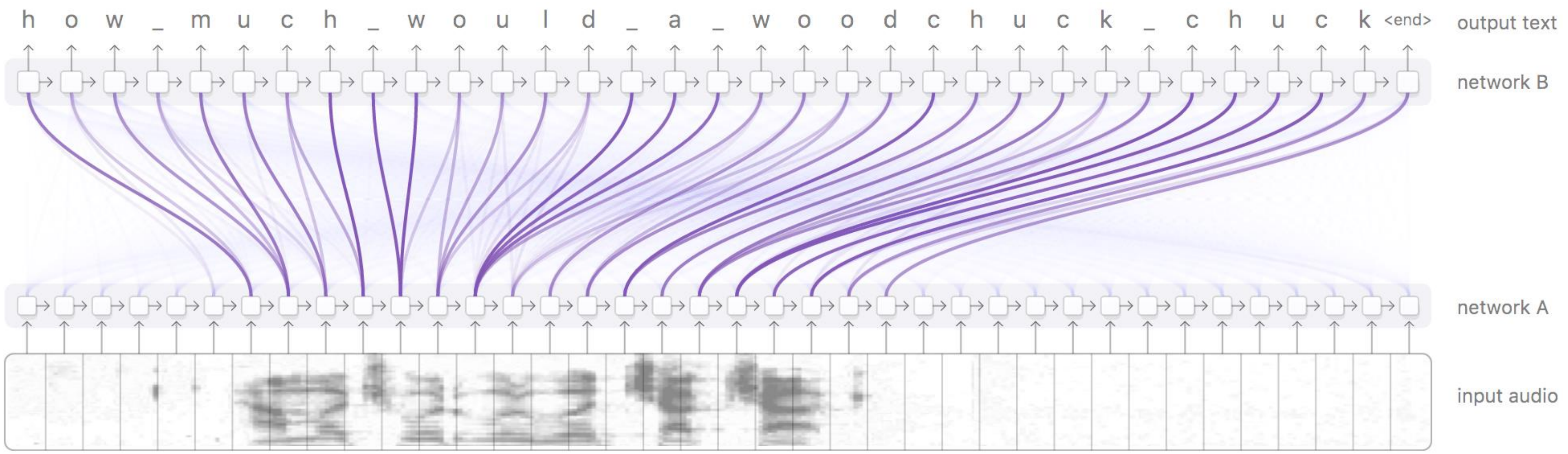


Many-to-many

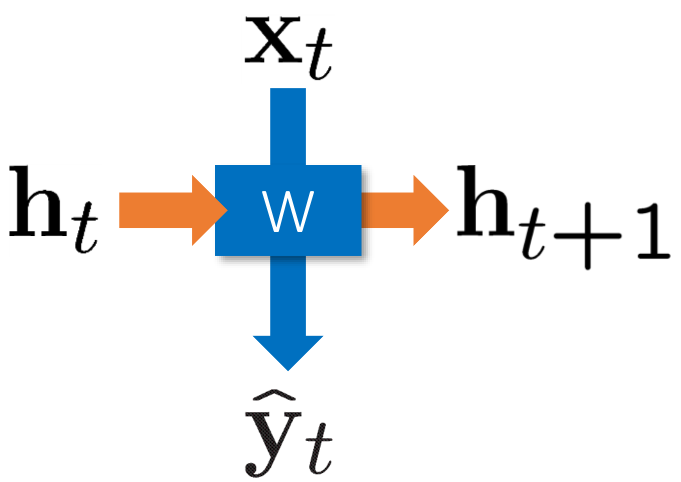
- Machine translation speech2text

audio.mp3

"How much would a woodchuck chuck"



Vanilla RNN



- Update state: $$\mathbf{h}_{t+1} = \tanh\left(\mathsf{W}_{\mathsf{xh}}\mathbf{x}_t + \mathsf{W}_{\mathsf{hh}}\mathbf{h}_t\right)$$
- Update output: $$\hat{\mathbf{y}}_t = \mathsf{W}_{\mathsf{hy}}\mathbf{h}_{t+1}$$
- Classification: $$\hat{\mathbf{p}}_t = \mathsf{SoftMax}\left(\hat{\mathbf{y}}_t\right)$$

Issue with long sequences

Vanishing / exploding gradients

- Gradient clipping

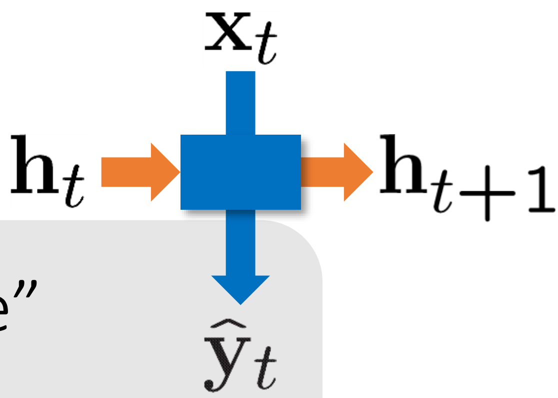
$$g_k = \nabla_\theta E_{B_k}(\theta^{(k)})$$

$$\text{if } \|g_k\| > \tau, \; g_k \leftarrow \frac{\tau}{\|g_k\|} g_k$$

Difficulty with long-term "memory"

- Gated unites

Cho *et al.* 2014: Gated Recurrent Unit (GRU)



- Update gates:

$$\mathbf{z}_t = \sigma\left(\mathrm{W}_{\mathrm{xz}}\mathbf{x}_t + \mathrm{W}_{\mathrm{hz}}\mathbf{h}_t\right) \quad \text{“update”}$$

$$\mathbf{r}_t = \sigma\left(\mathrm{W}_{\mathrm{xr}}\mathbf{x}_t + \mathrm{W}_{\mathrm{hr}}\mathbf{h}_t\right) \quad \text{“reset”}$$

- Update state:

$$\mathbf{h}_{t+1} = \mathbf{z}_t \odot \mathbf{h}_t + \bar{\mathbf{z}}_t \odot \tanh\left(\mathrm{W}_{\mathrm{xh}}\mathbf{x}_t + \left(\mathbf{r}_t \odot \mathrm{W}_{\mathrm{hh}}\mathbf{h}_t\right)\right)$$

- Update output:

$$\hat{\mathbf{y}}_t = \mathrm{W}_{\mathrm{yh}}\mathbf{h}_{t+1}$$

Hochreiter and Schmidhuber 1997
Long-Short Term Memory (LSTM)

“remember”
$$\mathbf{r}_t = \sigma\left(\mathrm{W}_{\mathrm{xr}}\mathbf{x}_t + \mathrm{W}_{\mathrm{wr}}\mathbf{w}_t\right)$$

“save”
$$\mathbf{s}_t = \sigma\left(\mathrm{W}_{\mathrm{xs}}\mathbf{x}_t + \mathrm{W}_{\mathrm{ws}}\mathbf{w}_t\right)$$

“focus”
$$\mathbf{f}_t = \sigma\left(\mathrm{W}_{\mathrm{xf}}\mathbf{x}_t + \mathrm{W}_{\mathrm{wf}}\mathbf{w}_t\right)$$

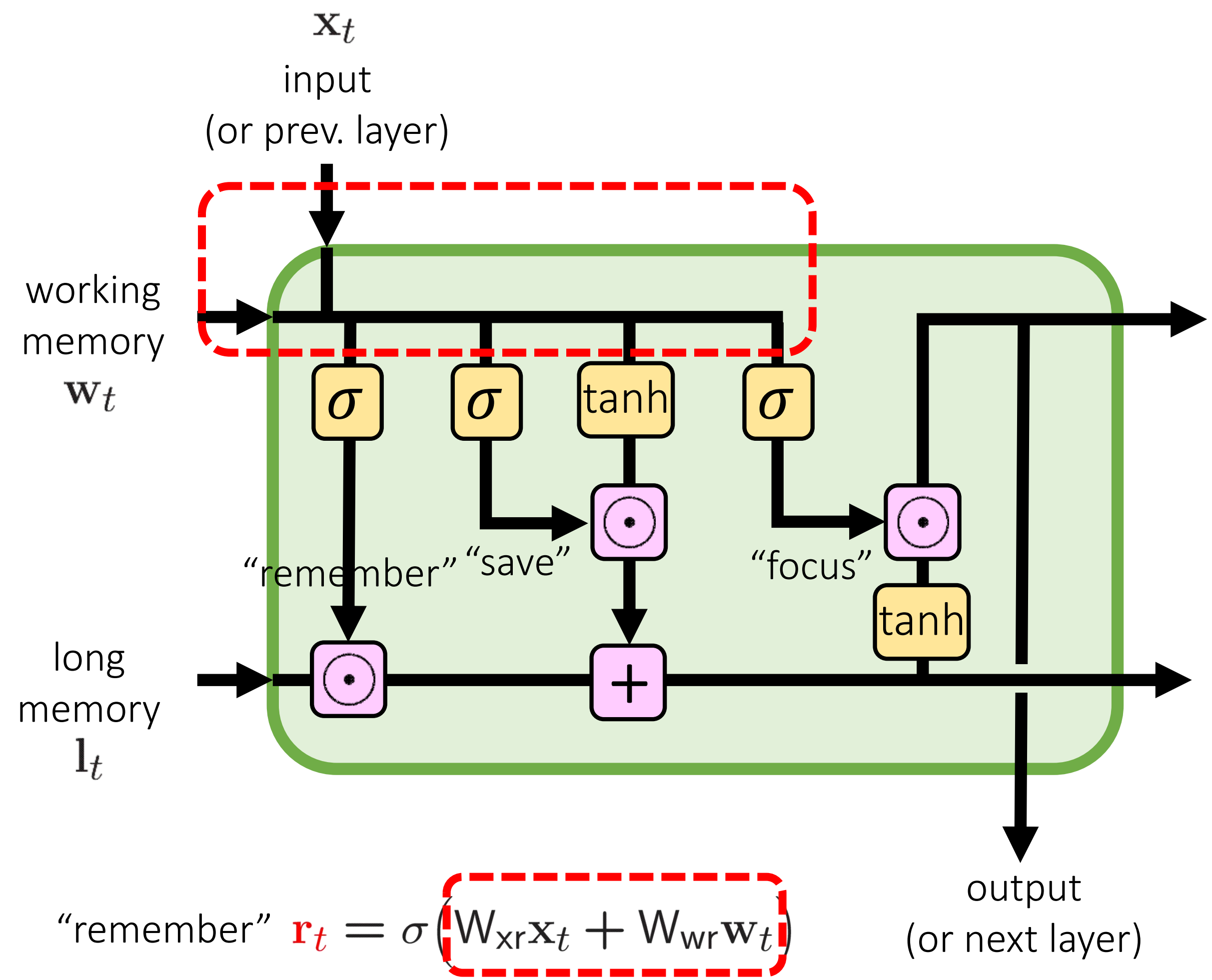
s

long memory
$$\mathbf{l}_{t+1} = \mathbf{r}_t \odot \mathbf{l}_t + \mathbf{s}_t \odot \tanh\left(\mathrm{W}_{\mathrm{xl}}\mathbf{x}_t + \mathrm{W}_{\mathrm{wl}}\mathbf{w}_t\right)$$

working memory
$$\mathbf{w}_{t+1} = \mathbf{f}_t \odot \tanh\left(\mathbf{l}_{t+1}\right)$$

output
$$\hat{\mathbf{y}}_t = \mathrm{W}_{\mathrm{yh}}\mathbf{w}_{t+1}$$

Long-Short Term Memory (LSTM)



$$\mathbf{w}_{t+1} = \mathbf{f}_t \odot \tanh\left(\mathbf{l}_{t+1}\right)$$

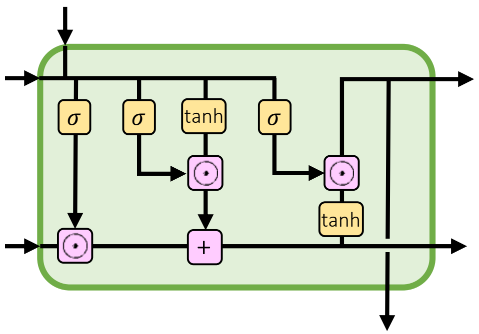
$$\mathbf{l}_{t+1} = \mathbf{r}_t \odot \mathbf{l}_t + \mathbf{s}_t \odot \tanh\left(\boxed{\mathsf{W}_{\mathsf{xl}}\mathbf{x}_t + \mathsf{W}_{\mathsf{wl}}\mathbf{w}_t}\right)$$

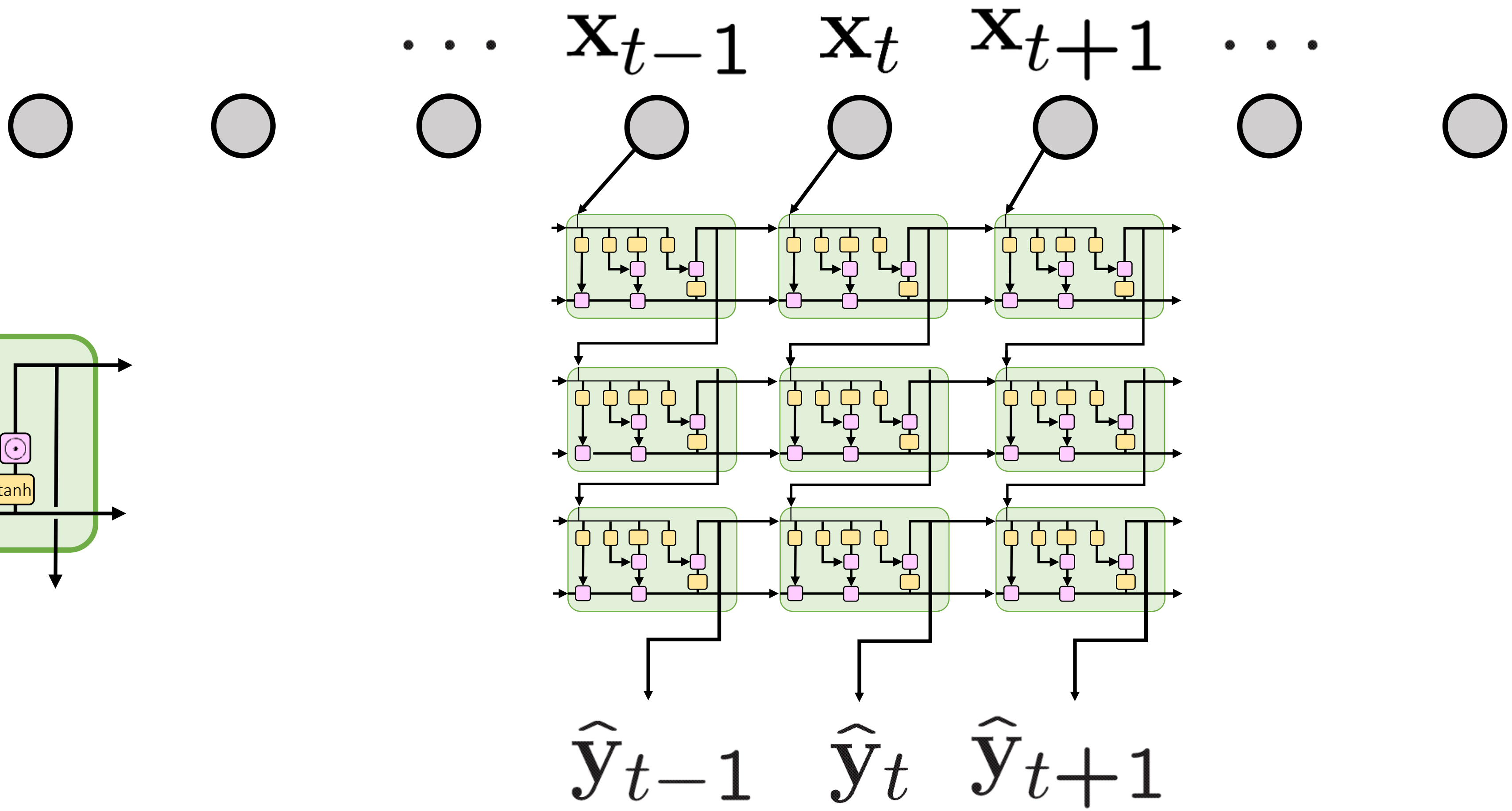
"remember" $\mathbf{r}_t = \sigma\left(\boxed{\mathsf{W}_{\mathsf{xr}}\mathbf{x}_t + \mathsf{W}_{\mathsf{wr}}\mathbf{w}_t}\right)$

"save" $\mathbf{s}_t = \sigma\left(\boxed{\mathsf{W}_{\mathsf{xs}}\mathbf{x}_t + \mathsf{W}_{\mathsf{ws}}\mathbf{w}_t}\right)$

"focus" $\mathbf{f}_t = \sigma\left(\boxed{\mathsf{W}_{\mathsf{xf}}\mathbf{x}_t + \mathsf{W}_{\mathsf{wf}}\mathbf{w}_t}\right)$

Deep RNN with LTSM units



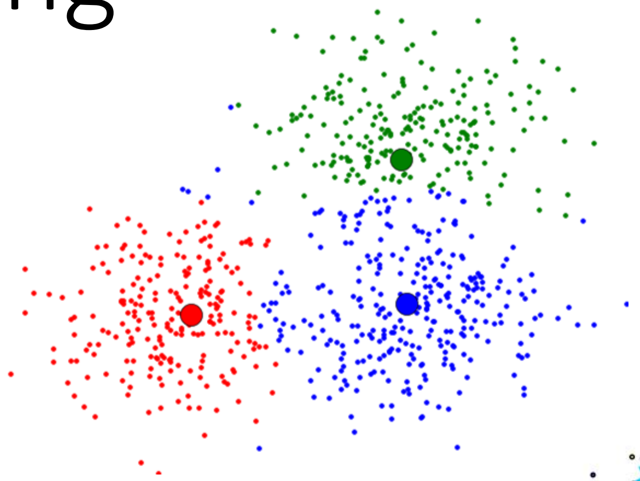


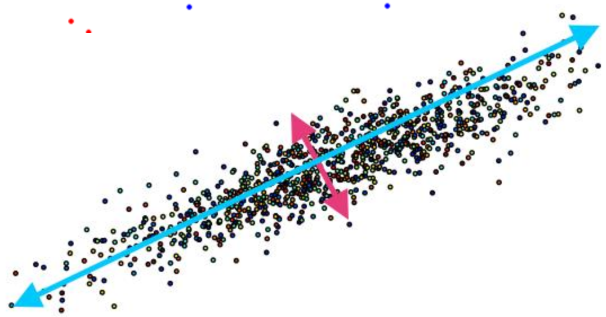
Unsupervised representations

Unsupervised learning

Discover useful structures in raw data

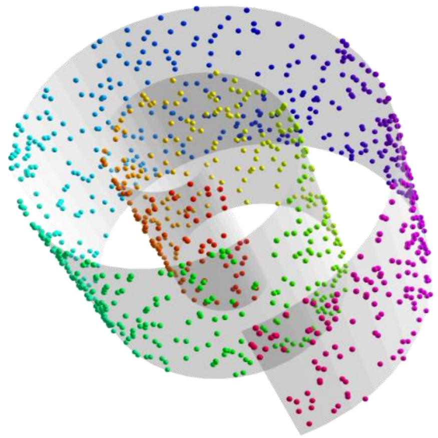
- Clusters, low-dim sub-space(s), manifold
- Probabilistic counterparts





Desirable properties

- "Related" inputs should have similar embedding
  - Semantic, context or geometric similarities
- Structure should be "economic"
  - Finite, low-dimensional or sparse
- Encoding should be approx. decodable



Unsupervised neural nets?

- One of current challenges
- Descriptive/predictive
  - Self-organizing (Kohonen) map [1982]
  - Auto-encoder (AE) [1988]
  - Restricted Boltzman machines (RBM) [2002-2006]
  - t-distributed stochastic neighbor embedding (t-SNE) [2008]
  - Skip-gram for word embedding [2013]
- Generative
  - Variational auto-encoder (VAE) [2013]
  - Generative adversarial net (GAN) [2014]

Auto-Encoder (AE)

Learning to reconstruct input from code

- Encoder: one layer perceptron

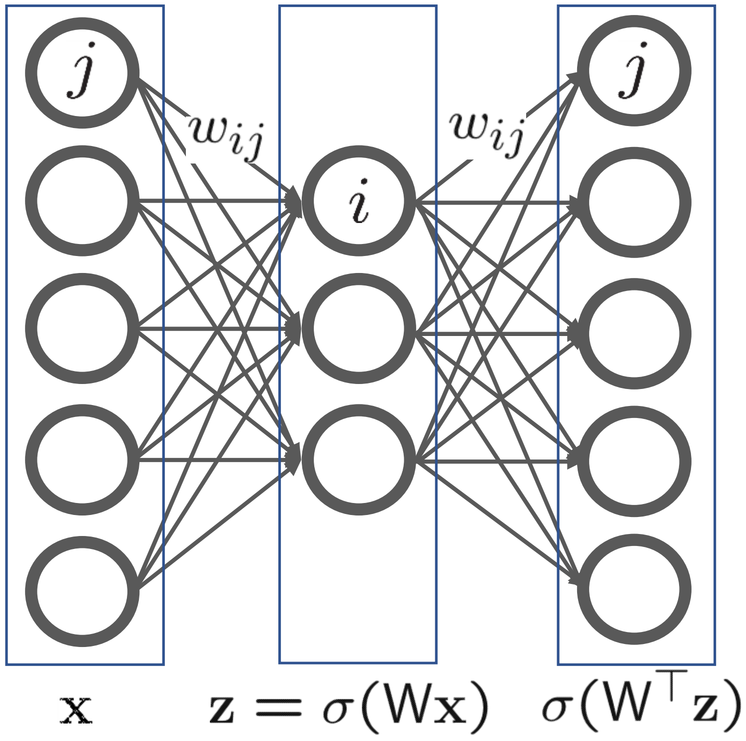
$$f(\mathbf{x}) = \sigma(\mathrm{W}\mathbf{x})$$

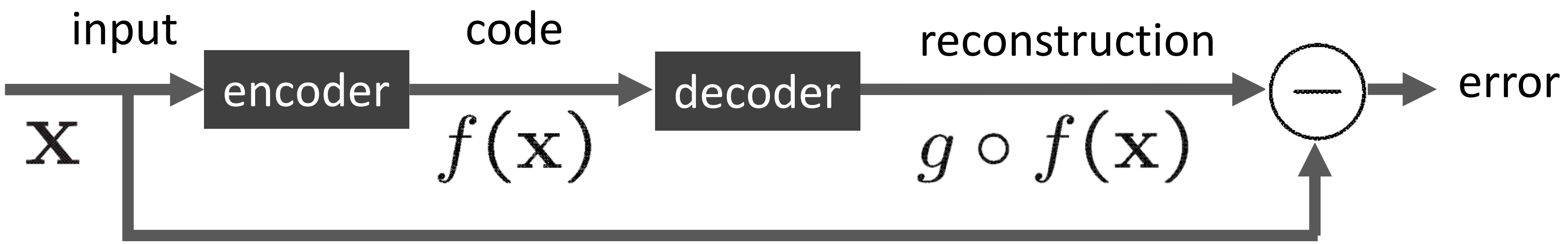
- Associated mirror decoder

$$g(\mathbf{z}) = \sigma(\mathrm{W}^\top \mathbf{z})$$

- Aiming at good reconstruction *on relevant data*

$$g \circ f \approx \mathrm{Id}$$





Auto-Encoder (AE)

Learning to reconstruct input from code

- Encoder: one layer perceptron

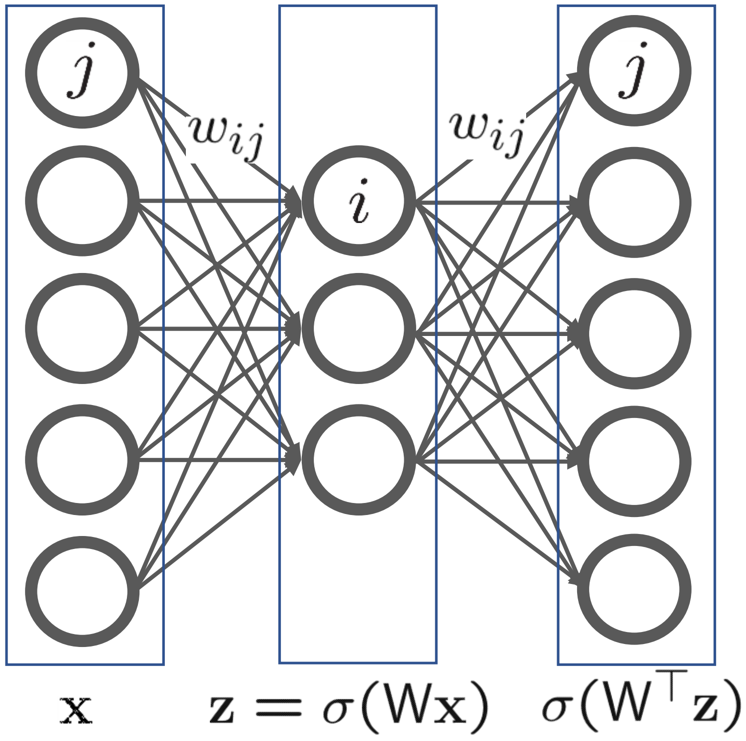
$$f(\mathbf{x}) = \sigma(\mathrm{W}\mathbf{x})$$

- Associated mirror decoder

$$g(\mathbf{z}) = \sigma(\mathrm{W}^\top \mathbf{z})$$

- Aiming at good reconstruction *on relevant data*

$$g \circ f \approx \mathrm{Id}$$



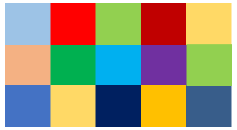
$$\min_{\boldsymbol{\theta}} \left[\frac{1}{N} \sum_{n=1}^{N} \left\| g \circ f(\mathbf{x}^{(n)}, \boldsymbol{\theta}) - \mathbf{x}^{(n)} \right\|^2 + \Omega(\boldsymbol{\theta}) \right]$$

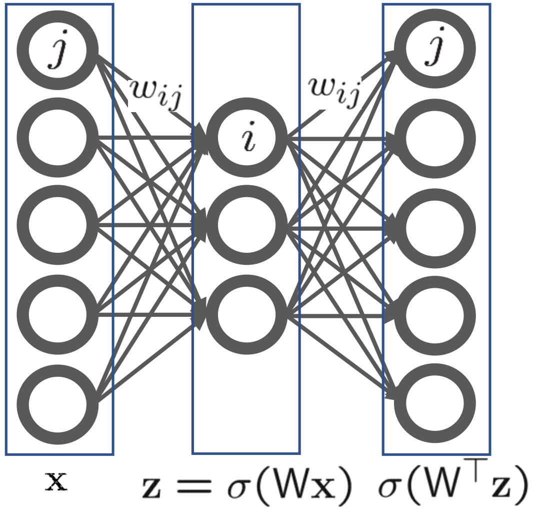
Auto-Encoder (AE)

AE with bottleneck

- Related to PCA

W =

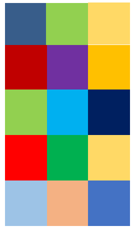


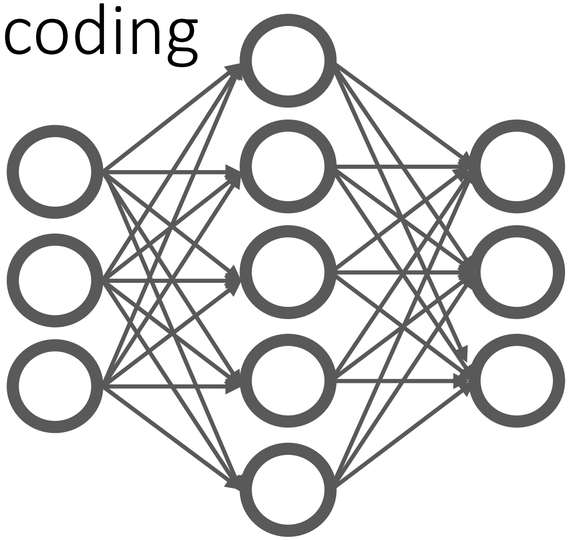


AE with sparsity prior

- Related to sparse coding

W =





Denoising AE

Corrupt inputs with noise

$$\mathcal{T} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$$

$$\min_{\boldsymbol{\theta}} \frac{1}{NE} \sum_{n=1}^{N} \sum_{e=1}^{E} \left\| g \circ f(\mathbf{x}^{(n)} + \boldsymbol{\varepsilon}^{(e,n)}, \boldsymbol{\theta}) - \mathbf{x}^{(n)} \right\|^2$$

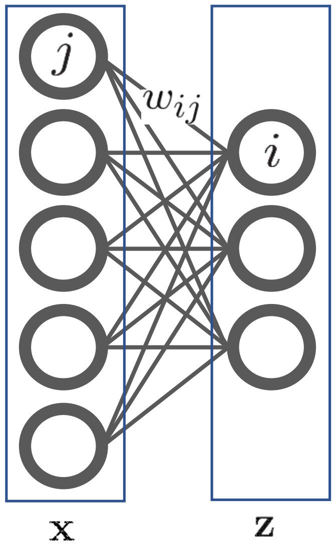
- Data augmentation
- Robustness, regularization, generalization
- Actual task!

Restricted Boltzman Machine (RBM)

- Bi-partite binary Markov random field from exponential family

$$\mathbb{P}(\mathbf{x},\mathbf{z}) \propto \exp - \underbrace{(\mathbf{z}^\top \mathsf{W}\mathbf{x} + \mathbf{x}^\top \mathbf{c} + \mathbf{z}^\top \mathbf{b})}_{E(\mathbf{x},\mathbf{z};\mathsf{W},\mathbf{b},\mathbf{c})}$$

$$\mathbb{P}(z_i = 1|\mathbf{x}) = [1 + \exp -(\textstyle\sum_j w_{ij}x_j + b_i)]^{-1}$$



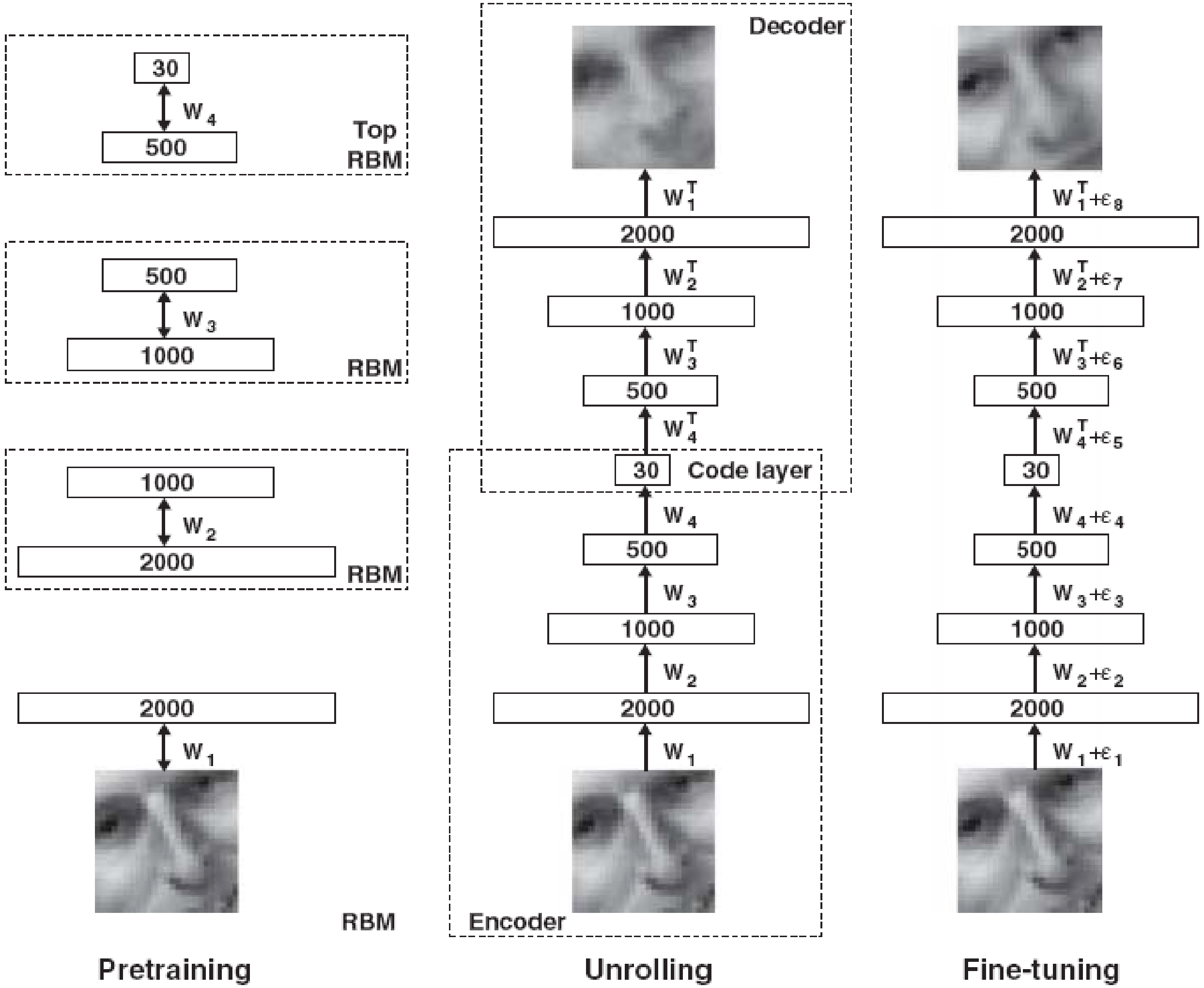
- Maximum likelihood training with SGD

$$\frac{\partial \log \mathbb{P}(\mathbf{x};\mathsf{W})}{\partial w_{ij}} = \underbrace{\mathbb{E}[x_j z_i|\mathsf{W}]}_{\text{untractable}} - \underbrace{\mathbb{E}[x_j z_i|\mathbf{x},\mathsf{W}]}_{x_j \mathrm{sigm}(\mathbf{w}_i^\top \mathbf{x} + b_i)}$$

  - Stochastic Monte Carlo approximation

Deepening

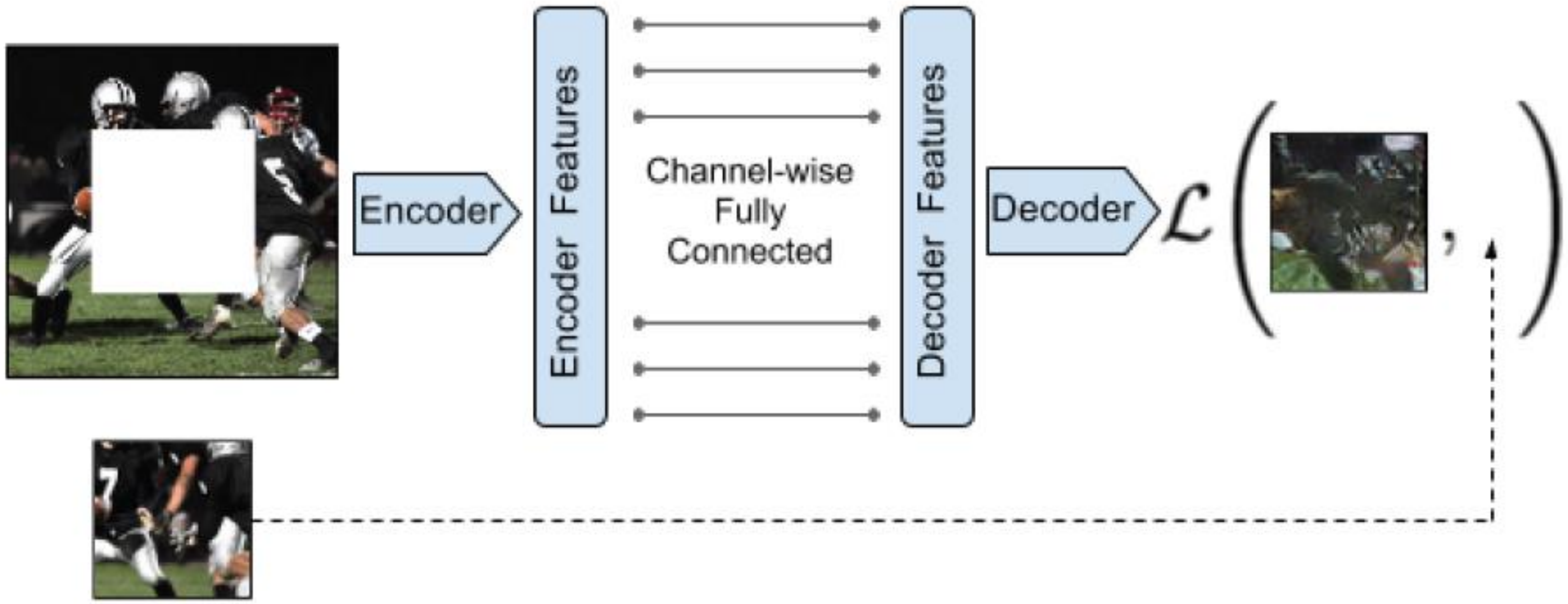
- Stack BMs/AEs, learned one at time + end-to-end fine tuning



[Hinton 2006]

Representation learning through prediction

- Learn to predict: surrounding letter, word, movement, image patch, video frame
- Trivial self-supervision from real raw data



[Pathak 2016]

Word-to-Vec

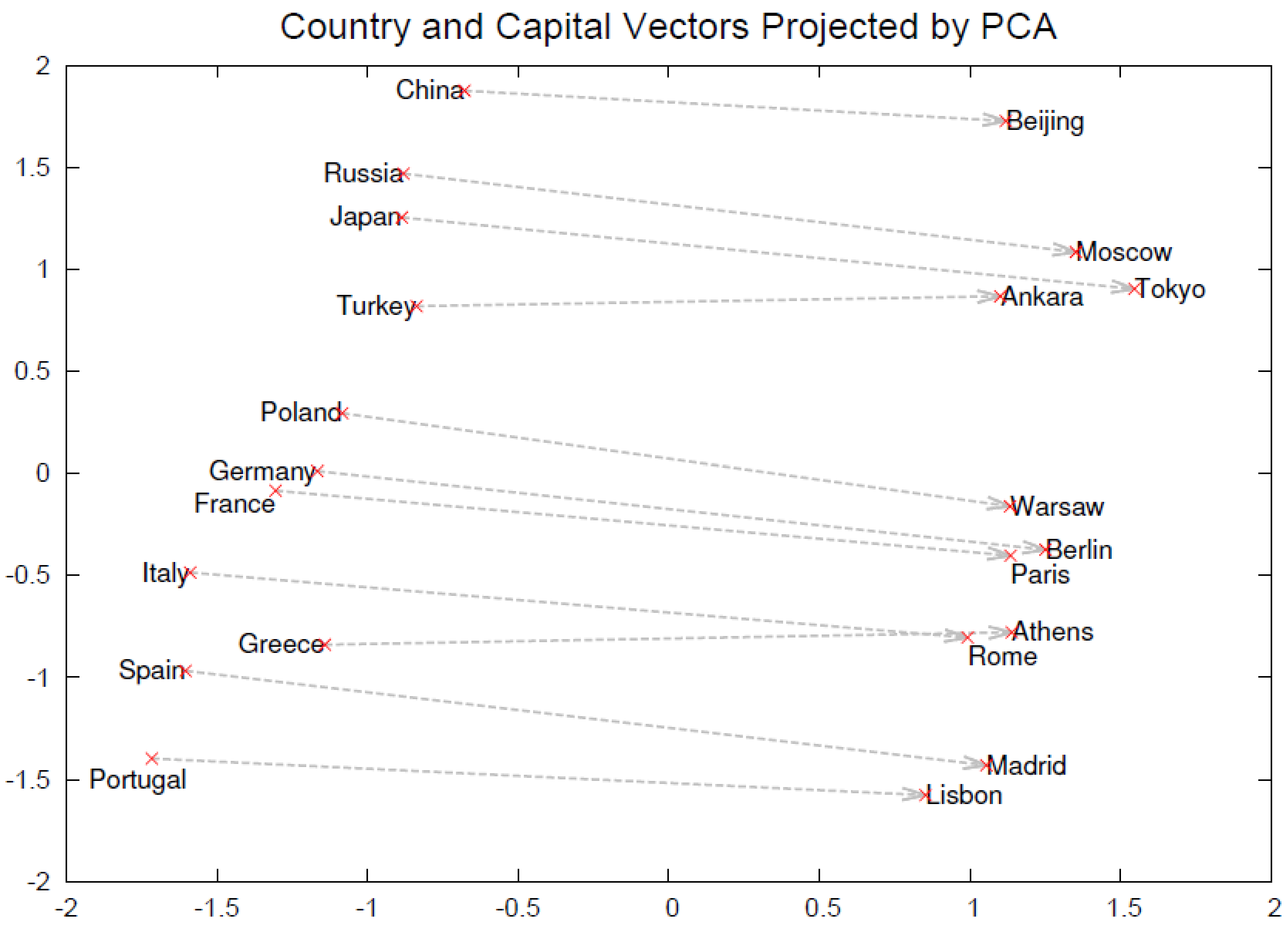
Word embedding based on "context"

- Learn explicit embedding allowing logistic regression of word given surrounding ones

$$\mathbb{P}[\mathbf{w}|\mathbf{c} \in \text{context}] \approx \text{SoftMax}\Big((\mathrm{D}\mathbf{w})^\top \sum_{\text{context}} \mathrm{D}\mathbf{c})\Big)$$

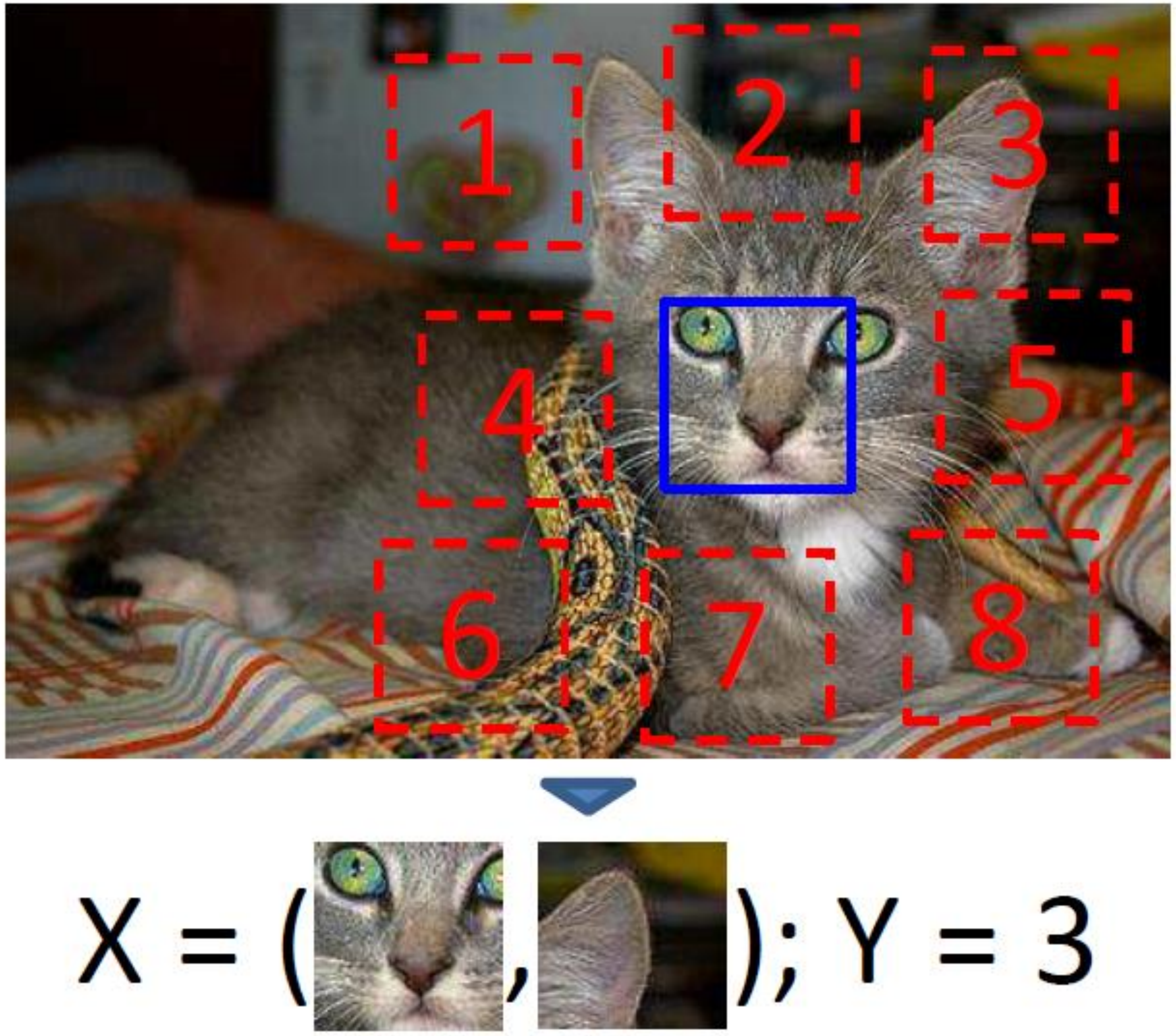
- Words frequently sharing context are mapped nearby
- Typical dimension: 100-600
- Surprising geometric relationships

Word-to-Vec



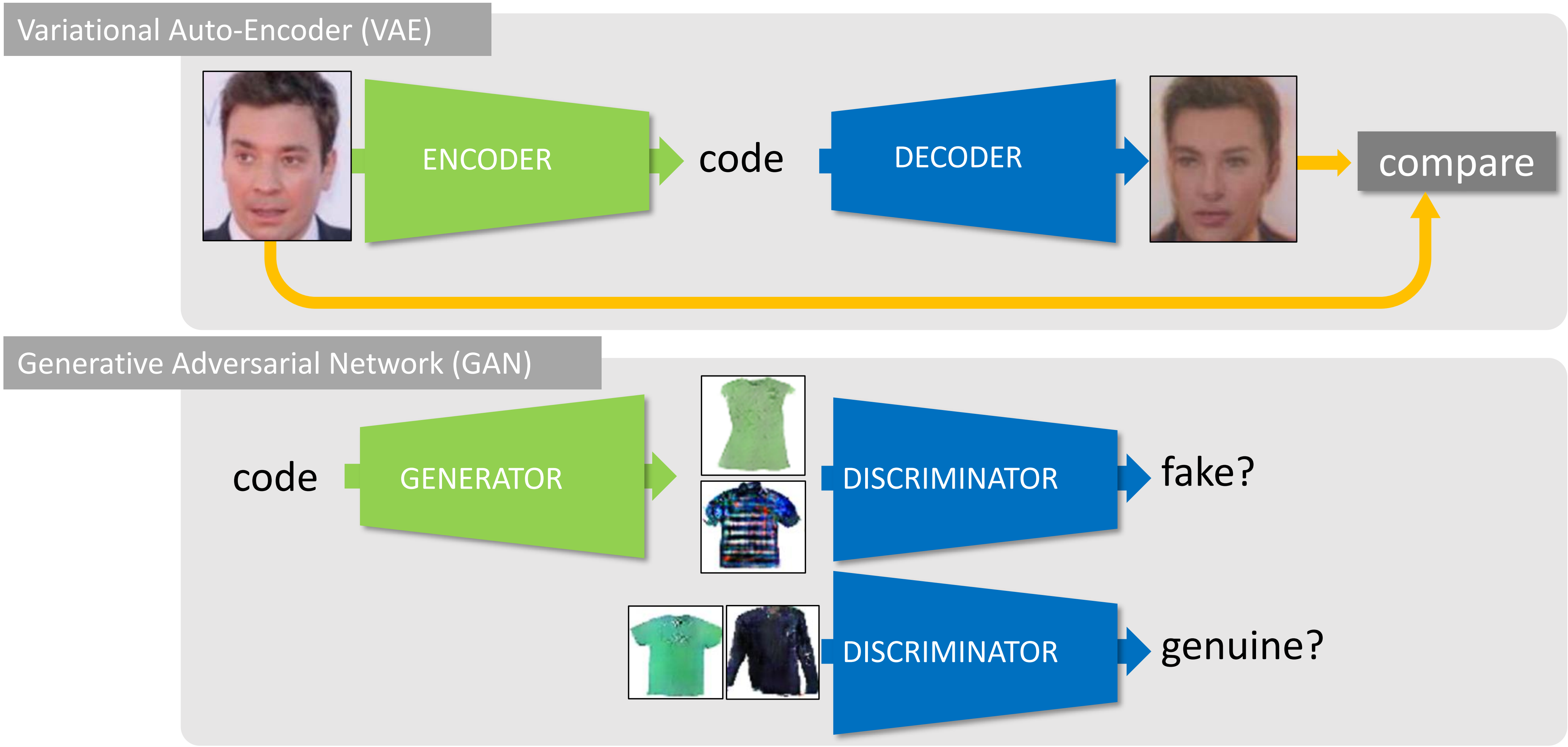
[Mikolov 2013]

Other self-supervised tasks



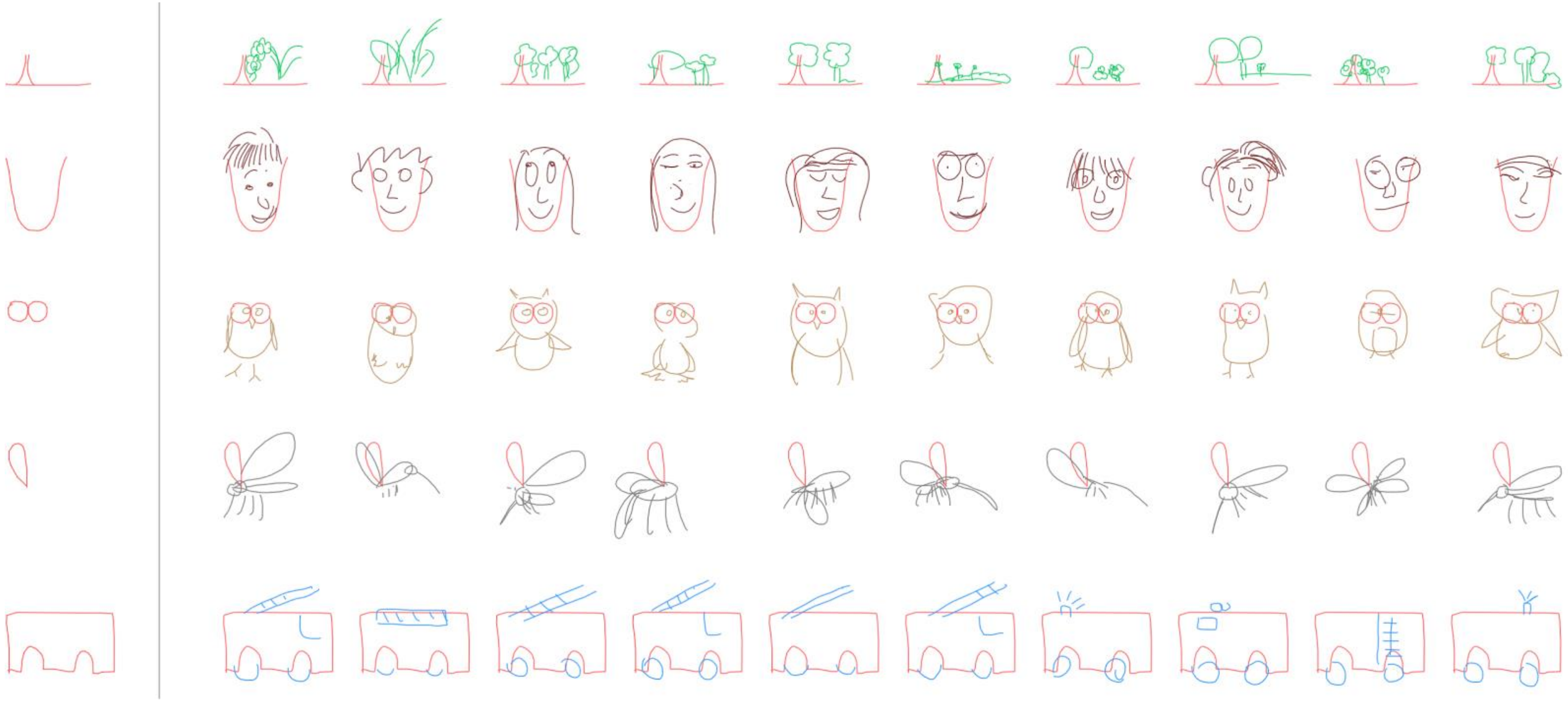
[Doersch 2015]

Generative models



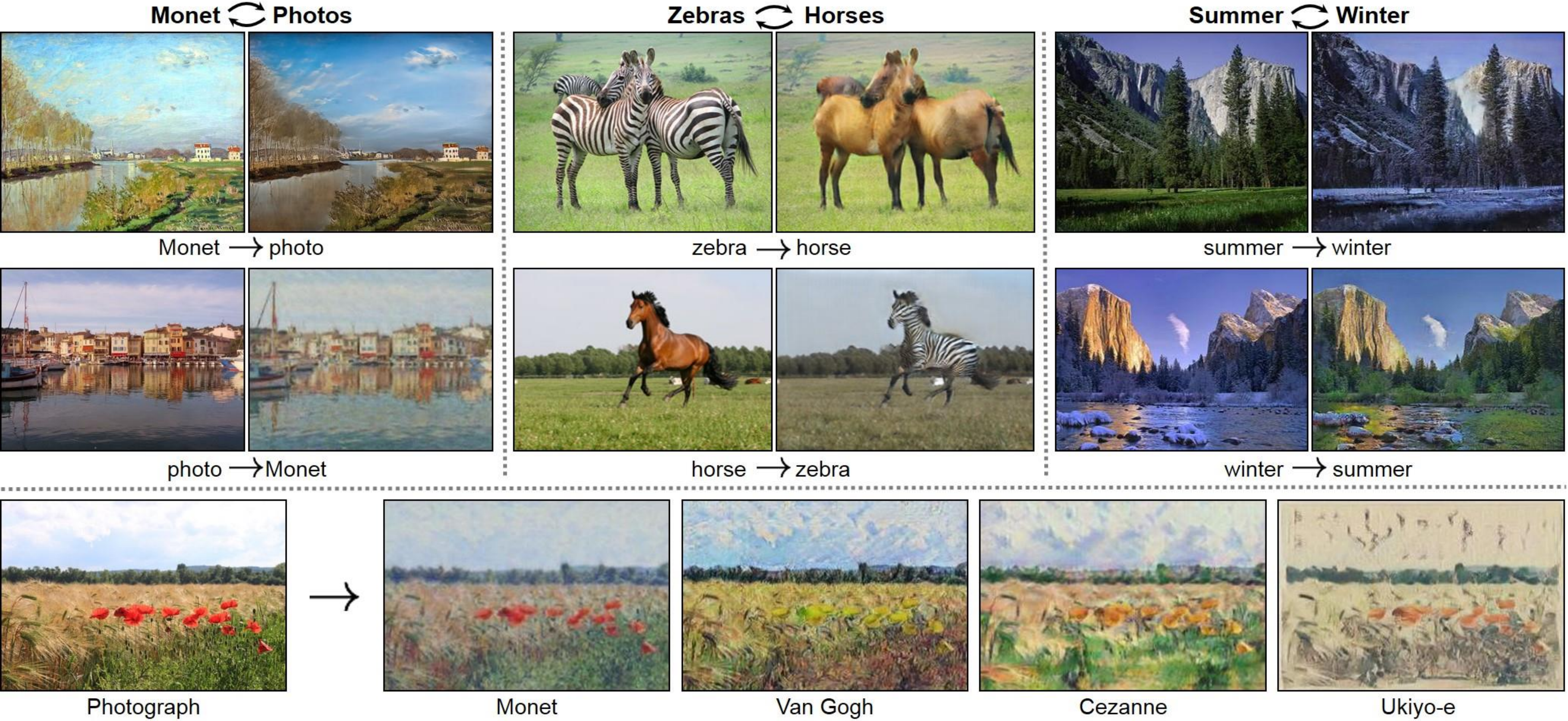
[Kingma and Welling 2013 / Goodfellow *et al*. 2014]

Ha and Eck 2017: Sketch VAE-RNN



Hou *et al.* 2016: Face VAE

Zhou *et al*. 2016: cGAN image translation



Karras *et al.* 2017: High-res GANs

Misc

Optimizing trained net w.r.t input

$$\arg\min_{\mathbf{x}_0} G(\mathbf{x}_{0:L}), \text{ with } \mathbf{x}_\ell = f_\ell \circ \cdots \circ f_1(\mathbf{x}_0)$$

- Network weights are frozen
- Random or specific initialization
- Iterative minimization with gradient back-prop

Application

- Visual inspection of the net, network inversion
- Create "adversarial perturbation"
- Modify or create new inputs with desirable properties

Visual inspection

- Input that maximizes a certain activation, e.g., class output

$$\arg\min_{\mathbf{x}_0} \|f(\mathbf{x}_0) - \mathbf{1}_c\|^2 + R(\mathbf{x}_0)$$

[Simonyan 2014]

Adversarial perturbation

- Small computed alteration of input to change output drastically

$$\arg\min_{\mathbf{x}_0} \|\mathbf{x}_0 - \mathbf{x}_0^*\| \text{ sb.t.} f(\mathbf{x}_0) \neq f(\mathbf{x}_0^*)$$

Its a bunny! I'm sure!

[Szegedy 2013]

Adversarial perturbation

- Small computed alteration of input to change output drastically

$$\arg\min_{\mathbf{x}_0} \|\mathbf{x}_0 - \mathbf{x}_0^*\| \text{ sb.t.} f(\mathbf{x}_0) \neq f(\mathbf{x}_0^*)$$

Its not a bunny! I'm sure!

[Szegedy 2013]

Adversarial perturbation

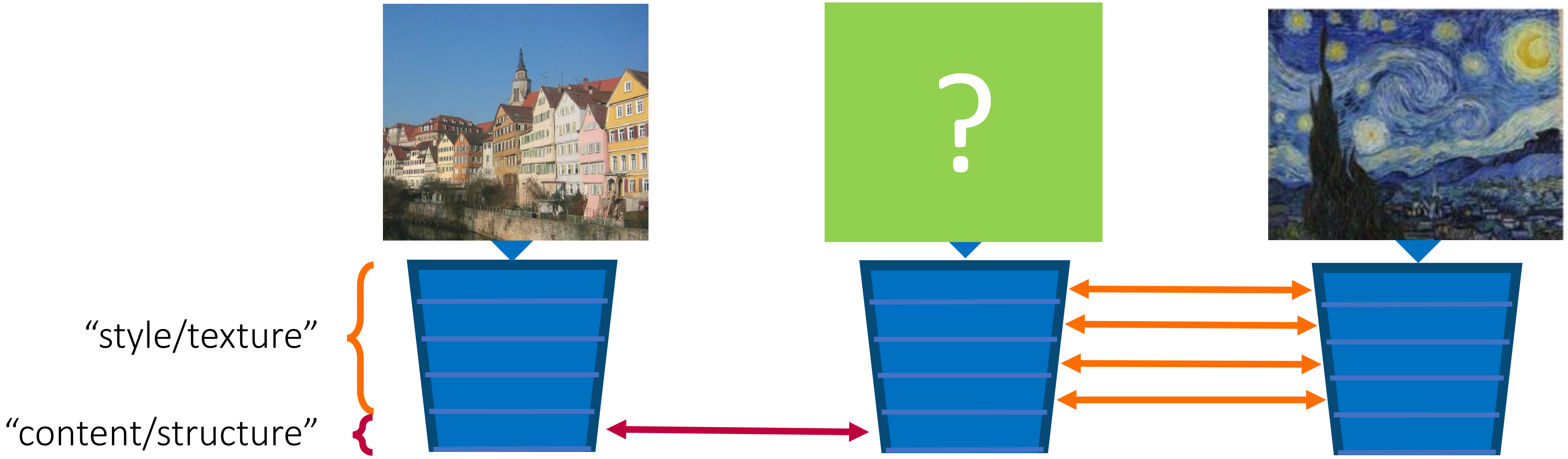
- Small computed change of input to change output drastically

$$\arg\min_{\mathbf{x}_0} \|\mathbf{x}_0 - \mathbf{x}_0^*\| \text{ sb.t. } f(\mathbf{x}_0) = \text{some target}$$

Its not a bunny! I'm sure!

[Szegedy 2013]

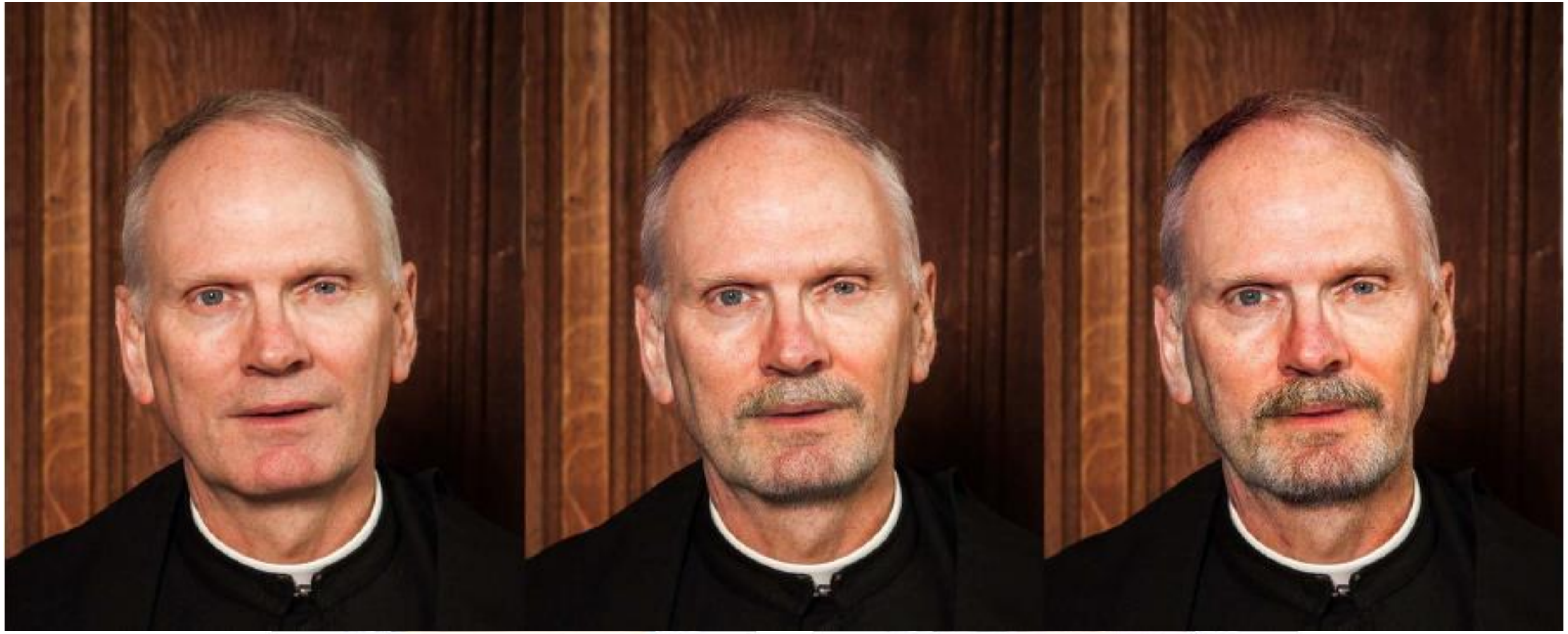
Gatys *et al.* 2015: Style transfer



$$\arg\min_{\mathbf{x}_0} \sum_{\ell \in L_s} \sum_{m,n} \left(\langle \mathbf{x}_{\ell,m}, \mathbf{x}_{\ell,n} \rangle - \langle \mathbf{x}^{S}_{\ell,m}, \mathbf{x}^{S}_{\ell,n} \rangle\right)^2 + \sum_{\ell \in L_c} \sum_{m} \|\mathbf{x}_{\ell,m} - \mathbf{x}^{*}_{\ell,m}\|^2$$

Gatys *et al.* 2015: Style transfer

Upchurch *et al*. 2017
Deep Feature Interpolation



Advanced topics

- Multi-tasking
- Meta learning (Auto ML)
- Efficient deep learning (on budget)
- Explainable deep learning
- Provable deep learning
- Trainable logic
- Neural solvers
- Bayesian neural nets
- Long term memory
- Regularization and generalization
- Transfer learning