Yandex

Machine Learning – Part 4

Nikita Kazeev

With acknowledgments to
Maxim Borisyak, Alexander Panin, Andrey Ustyuzhanin

Thank you for the feedback!

How does magic happen?

Same technique* for:
- Playing Go
- Computer vision
- Higgs boson detection
- Machine translation

...

Image source

How does magic happen?

Bricks aka layers

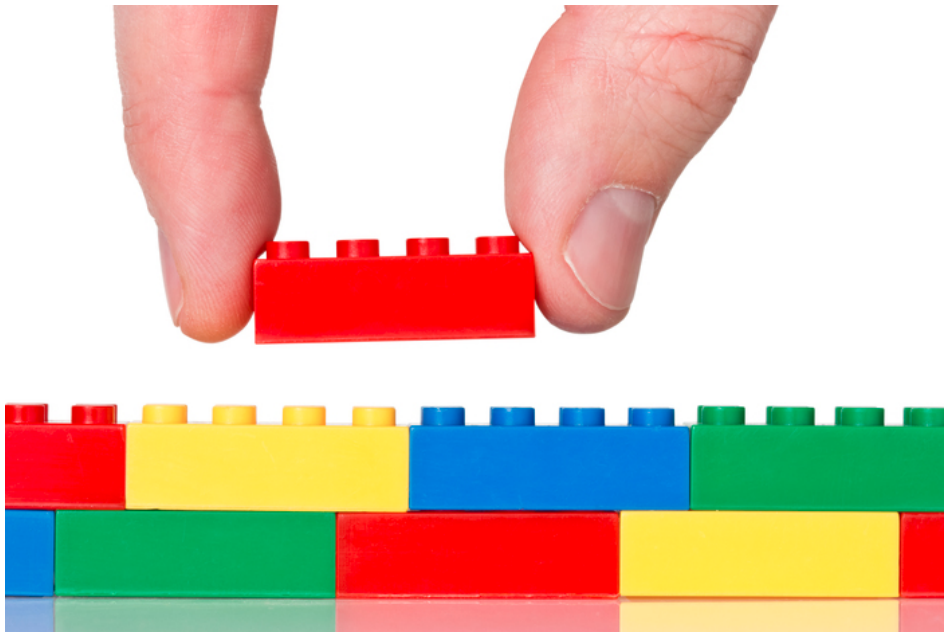
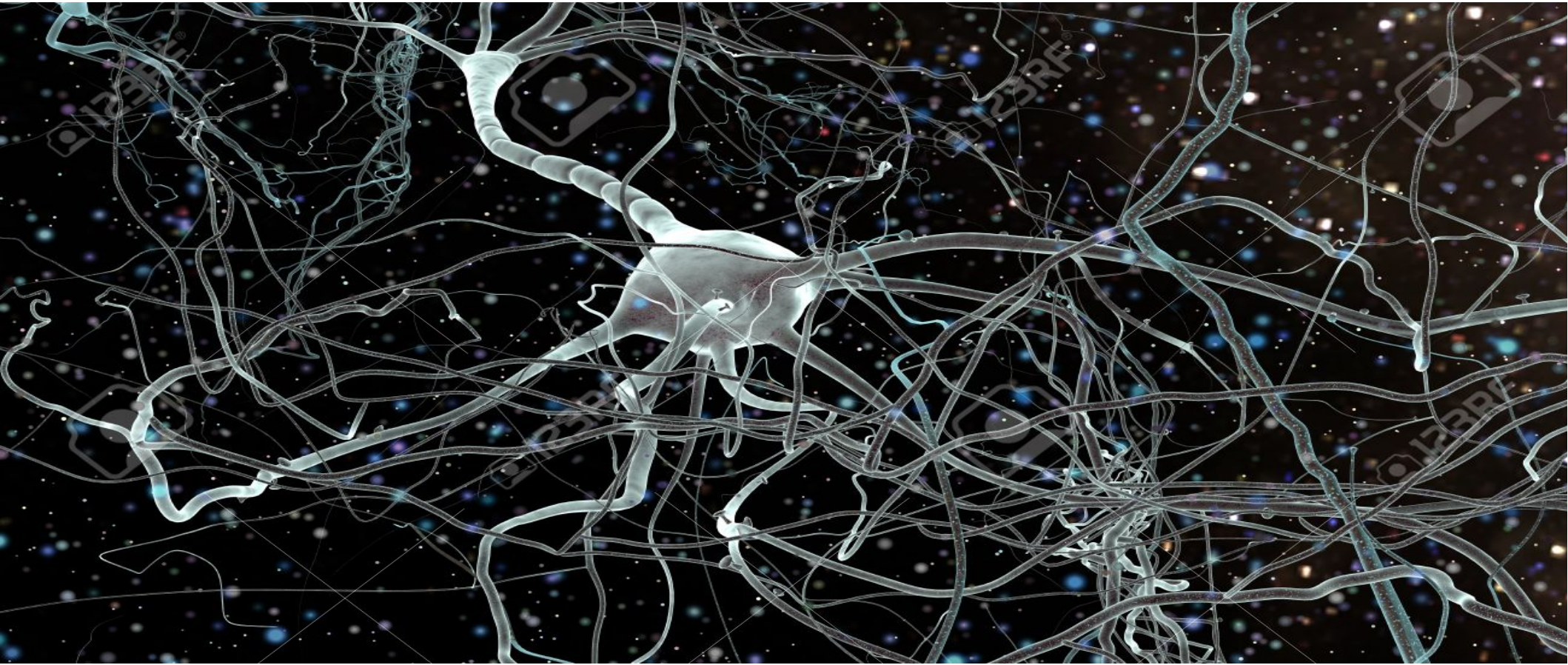
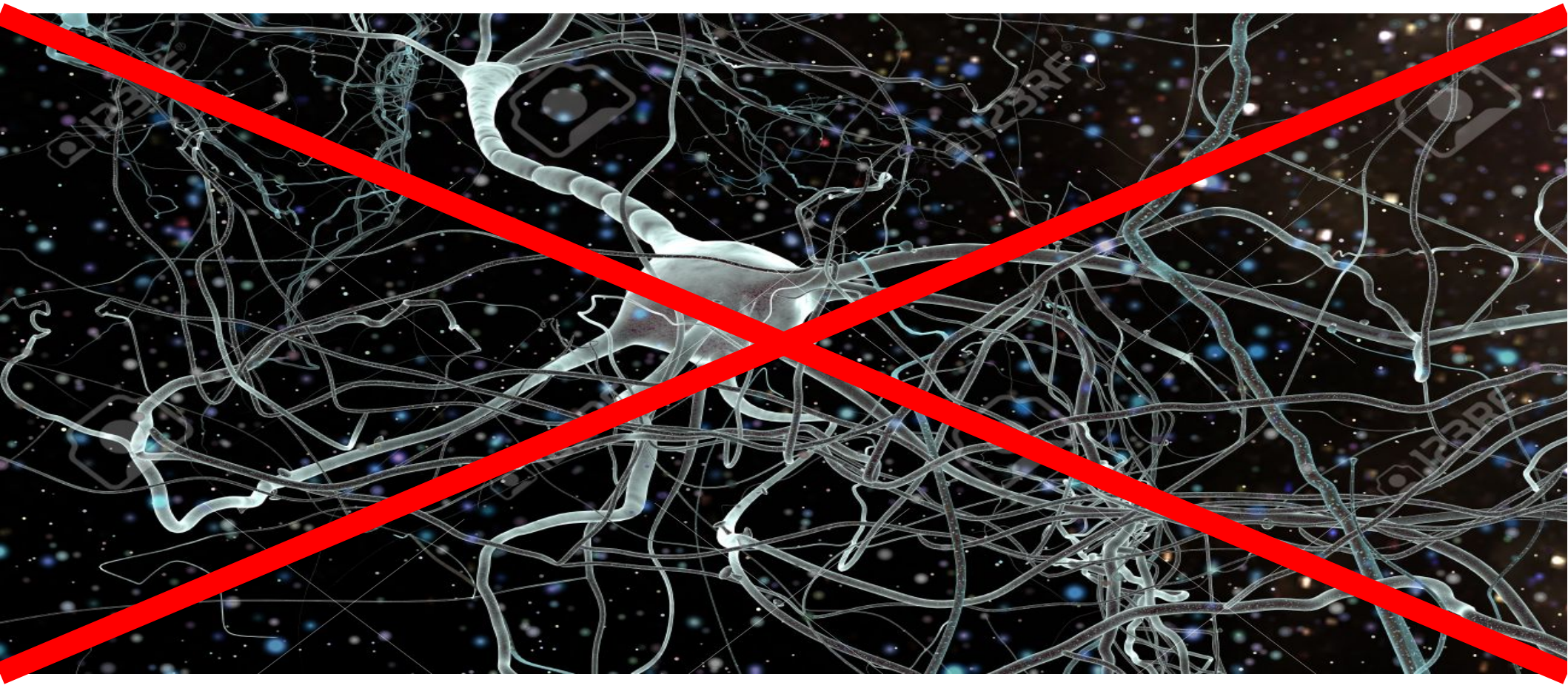


Image source

Biological inspiration



Biological inspiration

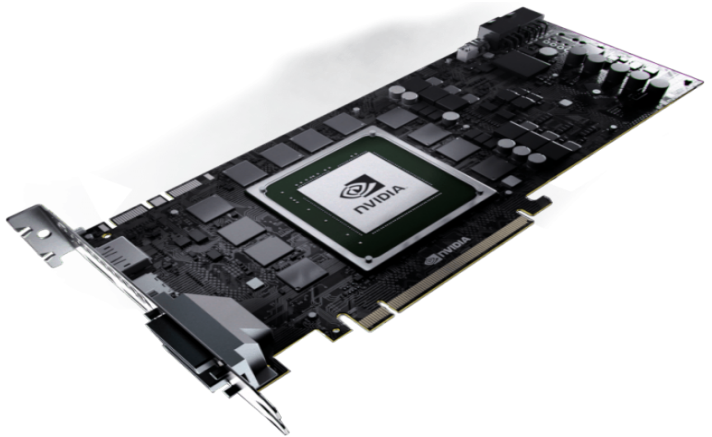


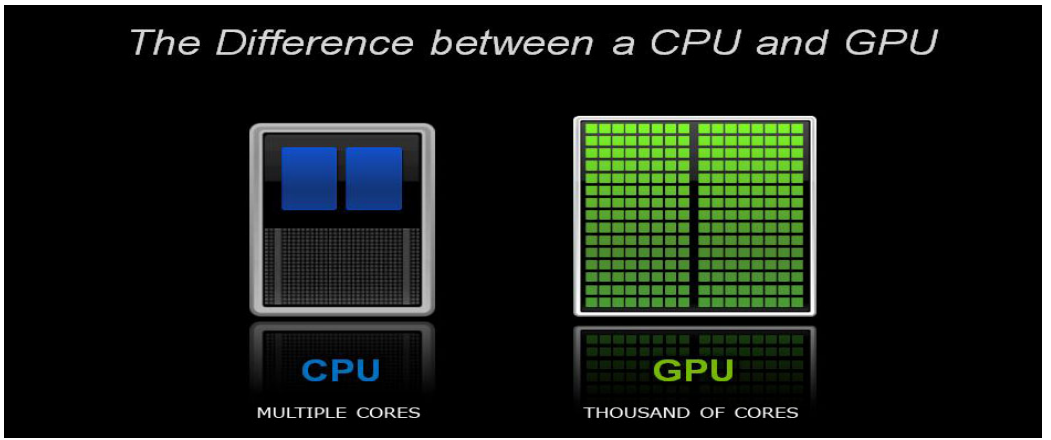
Ingredients of deep learning

- Lots of data
- Enough computing power

Image source

Computation



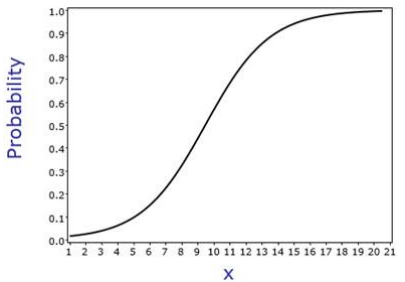


Plan for the lecture

- Representation learning
- Training deep models
- Convolutions
- <maybe> Unsupervised learning
- <maybe> Transfer learning
- <maybe> Sequential data processing (RNN)
- Practice with Keras

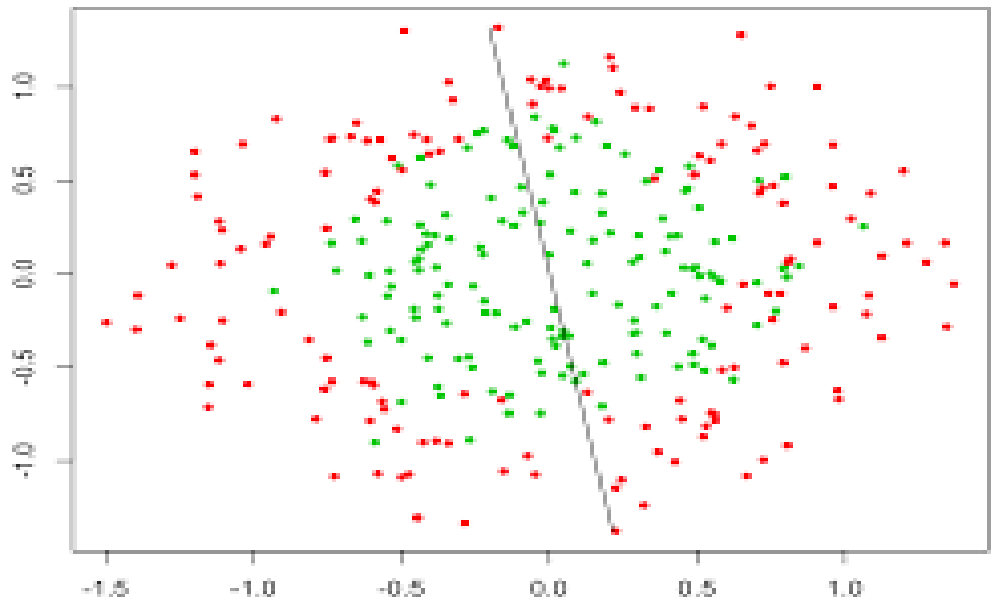
Representation learning

Recap: logistic regression

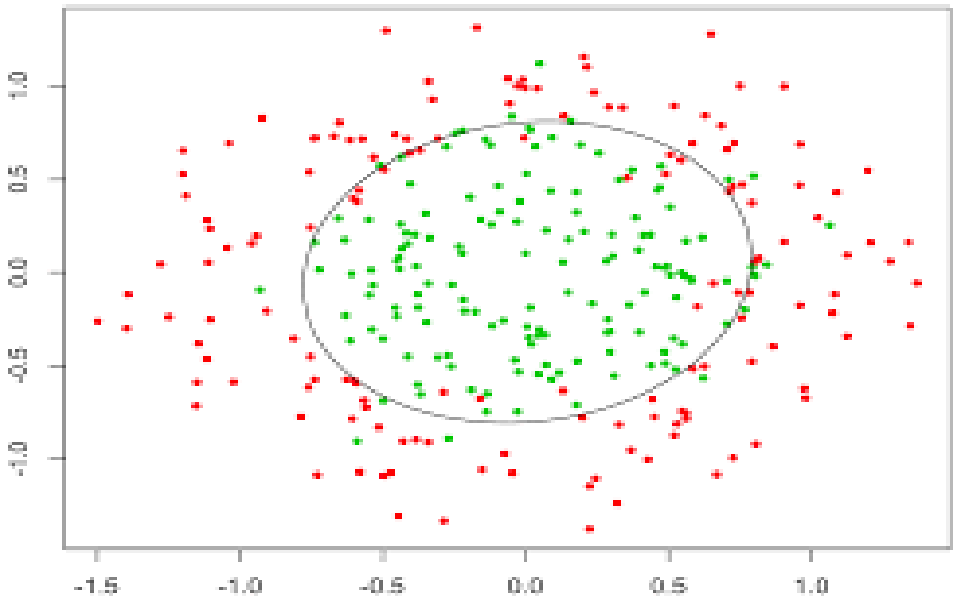
X → Wx +b → 

→ P(y)

Nonlinear dependencies



What we have



What we want

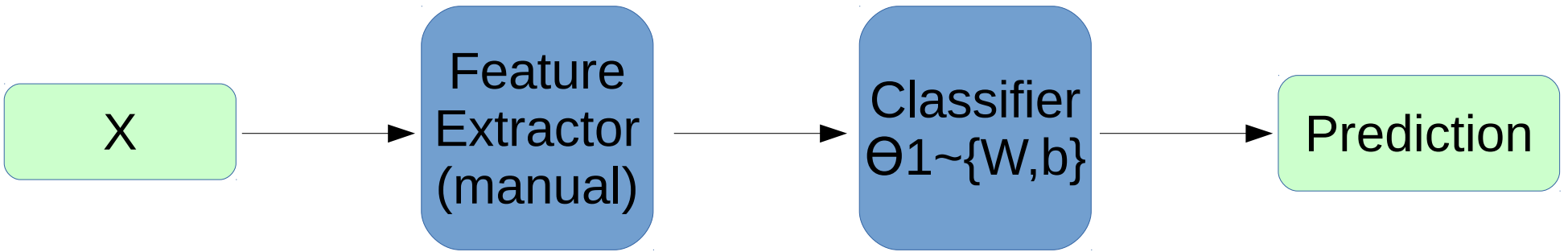
- How to get that?

Feature extraction

Loss, for example:

$$L = -\sum_i y_i \log P(y|x_i) + (1 - y_i) \log (1 - P(y|x_i))$$

Model:



Training:

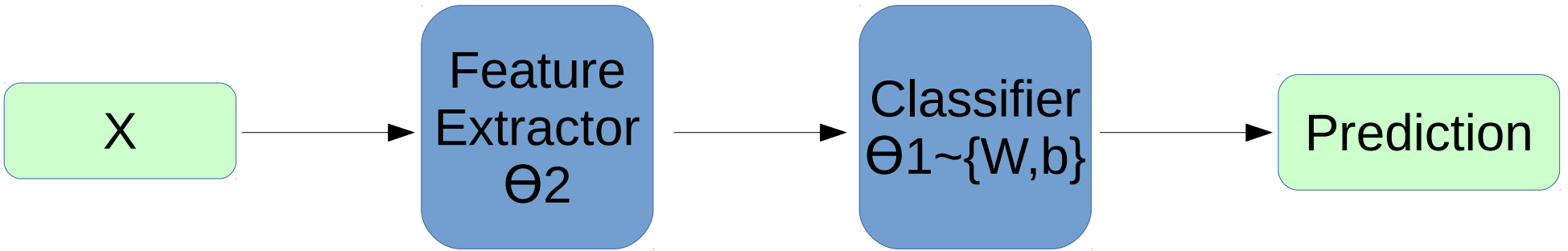
$$\underset{\theta_1}{argmin}\, L(y, P(y|x))$$

What do we want, exactly?

Loss, for example:

$$L = -\sum_i y_i \log P(y|x_i) + (1 - y_i) \log (1 - P(y|x_i))$$

Model:



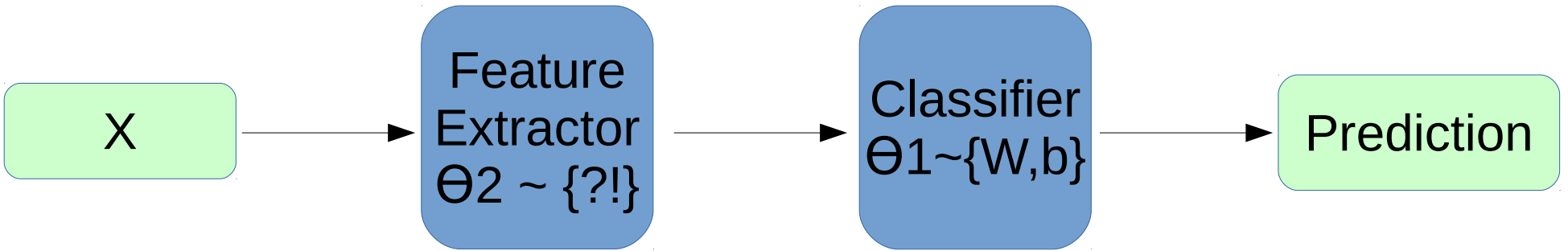
Training: ? $\underset{\theta_1}{argmin}\, L(y, P(y|x))$

What do we want, exactly?

Loss, for example:

$$L = -\sum_i y_i \log P(y|x_i) + (1 - y_i) \log (1 - P(y|x_i))$$

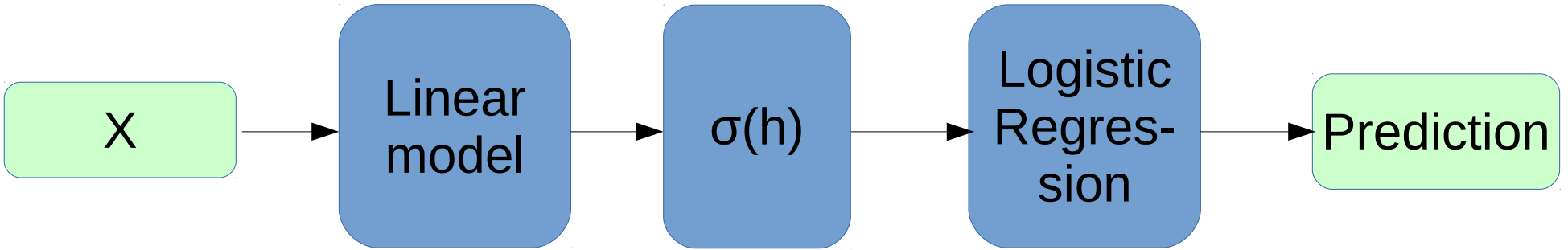
Model:



Gradients: $\underset{\theta_2}{argmin}\, L(y, P(y|x))$ $\underset{\theta_1}{argmin}\, L(y, P(y|x))$

Nonlinearity

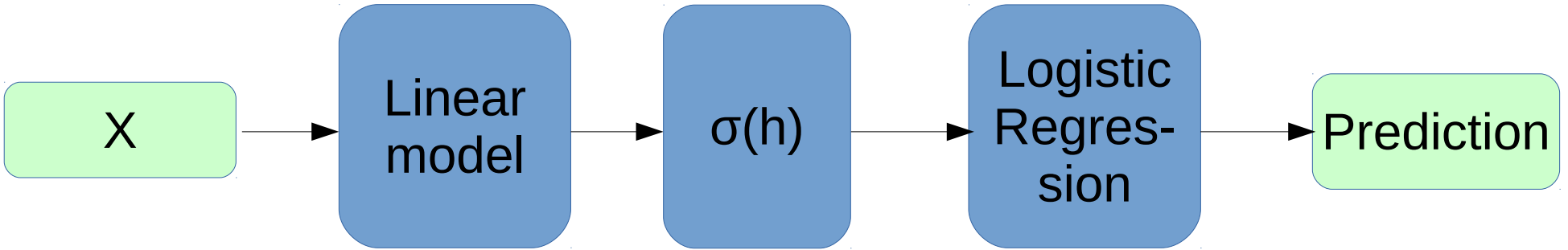
Model:



$$h_j = \sigma\left(\sum_{\substack{i \\ j \in \{1,2,\ldots,n\}}} w_{ij}^h x_i + b_j^h\right) \qquad y_{pred} = \sigma\left(\sum_j w_j^o h_j + b^o\right)$$

Nonlinearity

Model:



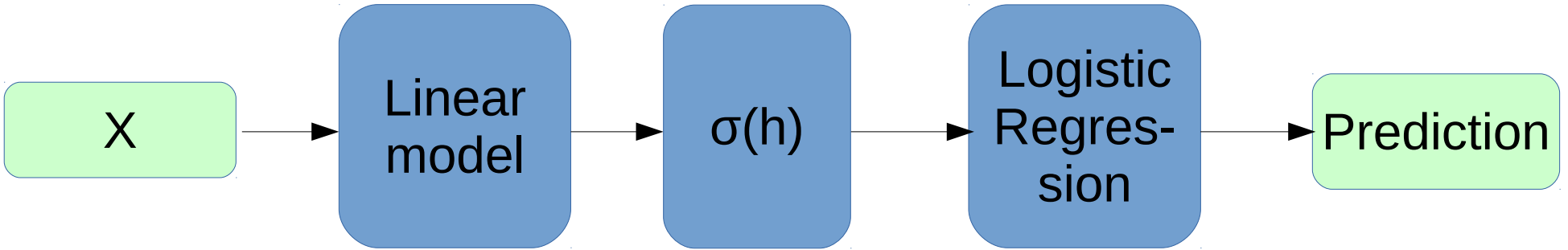
$$h_j = \sigma\left(\sum_{\substack{i \\ j \in \{1,2,\ldots,n\}}} w_{ij}^h x_i + b_j^h\right) \qquad y_{pred} = \sigma\left(\sum_j w_j^o h_j + b^o\right)$$

Output:

$$P(y|x) = \sigma\left(\sum_j w_j^o \sigma\left(\sum_i w_{ij}^h x_i + b_j^h\right) + b^o\right)$$

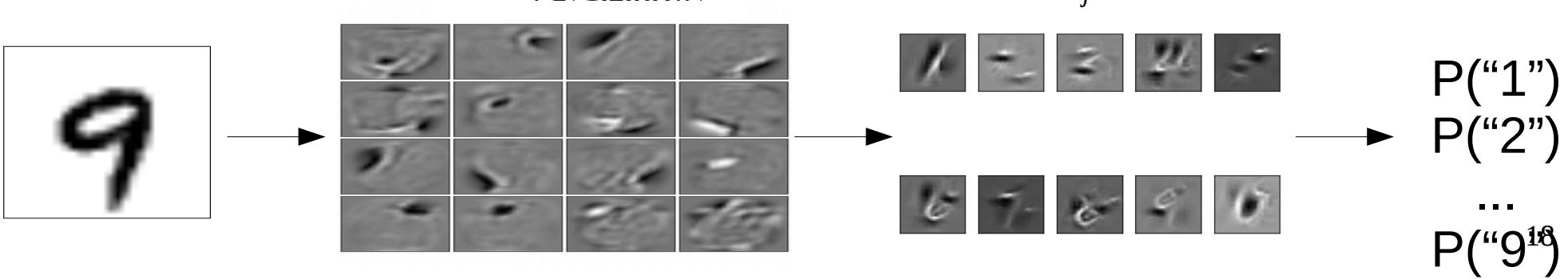
Nonlinearity

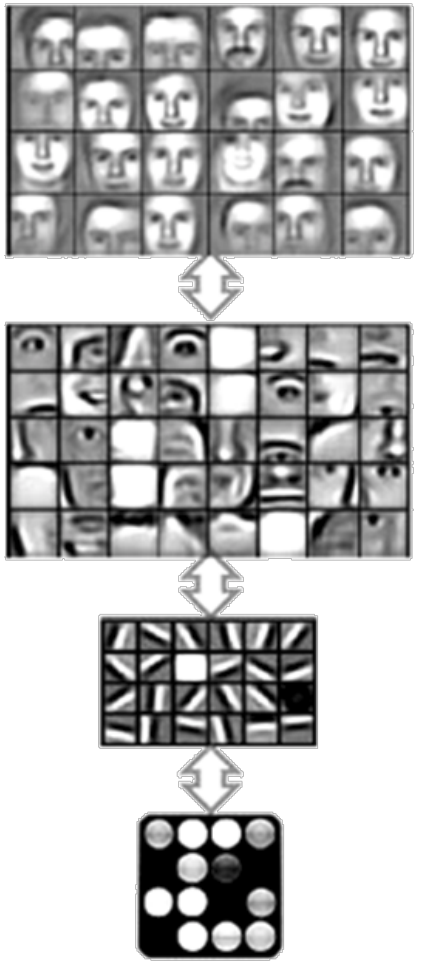
Model:



$$h_j = \sigma\left(\sum_{i \in \{1.2.....n\}} w_{ij}^h x_i + b_j^h\right)$$

$$y_{pred} = \sigma\left(\sum_j w_j^o h_j + b^o\right)$$





Discrete Choices

Layer 2 Features

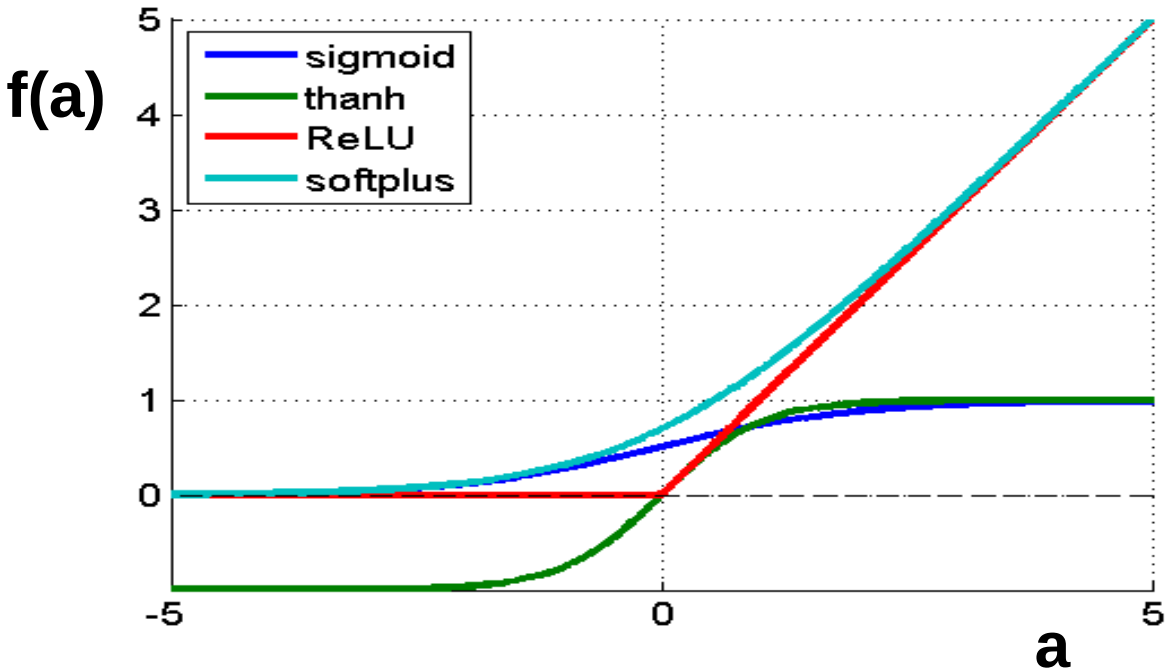
Layer 1 Features

Original Data

Nonlinearity

- f(a) = 1/(1+e^a)
- f(a) = tanh(a)

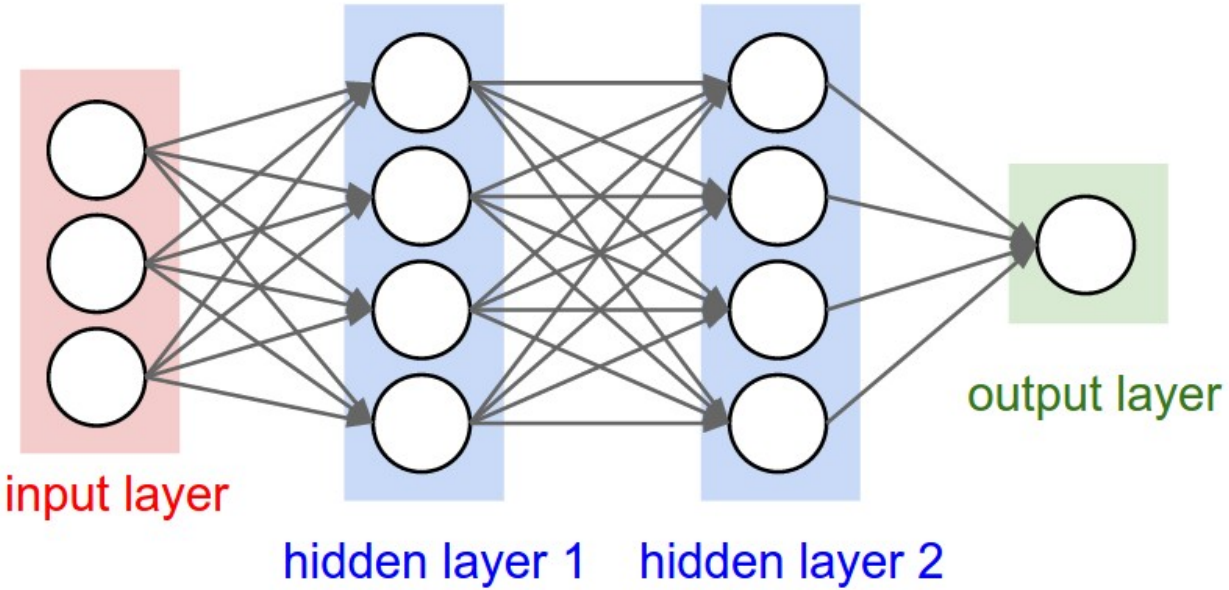
- f(a) = max(0,a)
- f(a) = log(1+e^a)



Connectionist phrasebook

- Layer – a building block for NNs :
    - “Dense layer”: f(x) = Wx+b
    - “Nonlinearity layer”: f(x) = σ(x)
    - Input layer, output layer
    - A few more we will cover later
- Activation – layer output
    - i.e. some intermediate signal in the NN

How do we train it?



Backpropagation

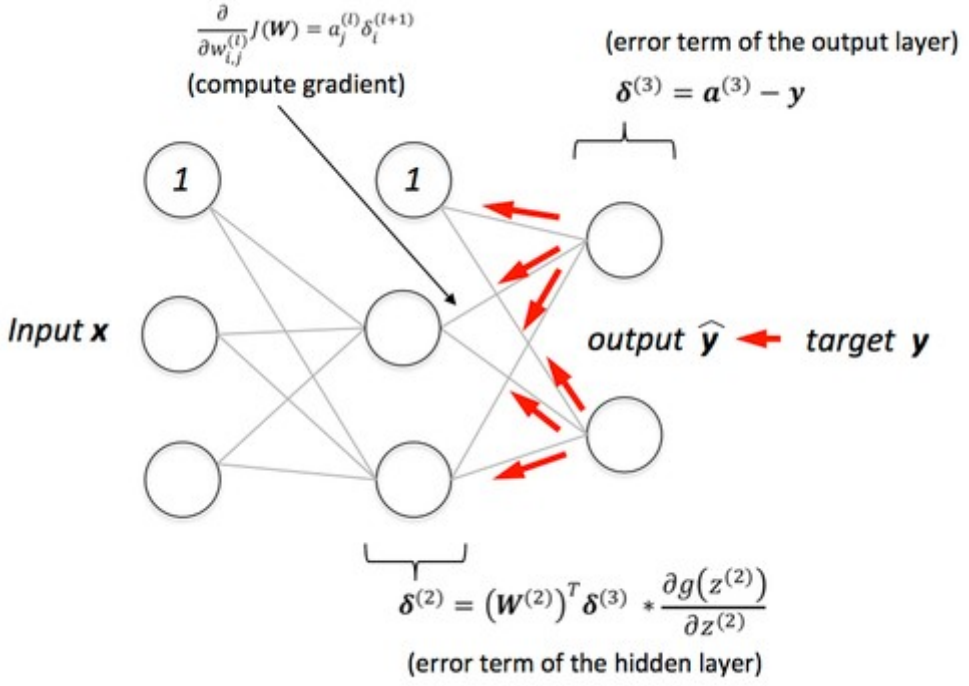


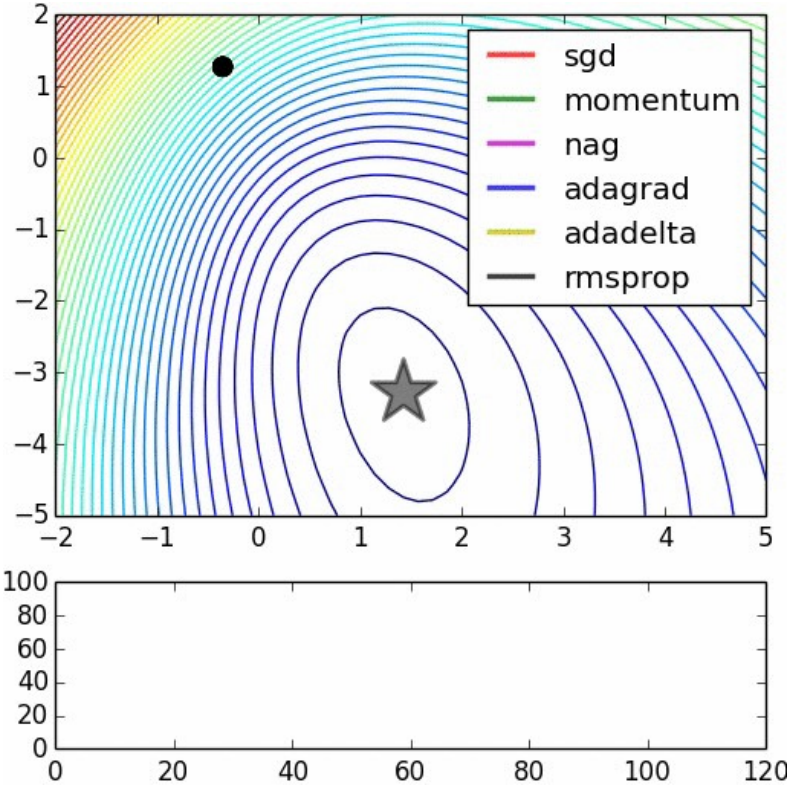
Image source

Why backpropagation?

Combine **any** differentiable functions into **any*** graph!

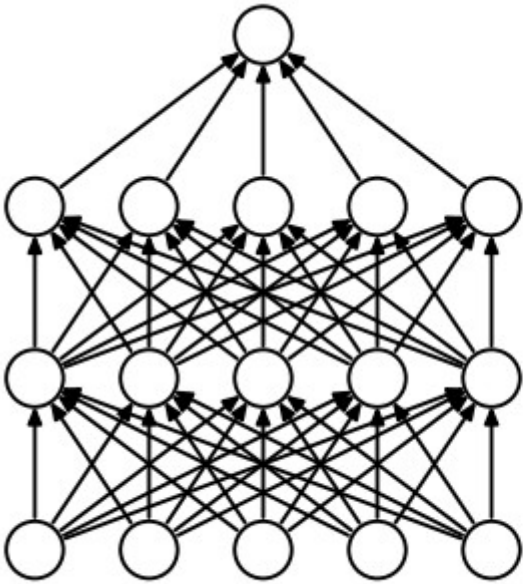
Image source

Apply your favorite optimization

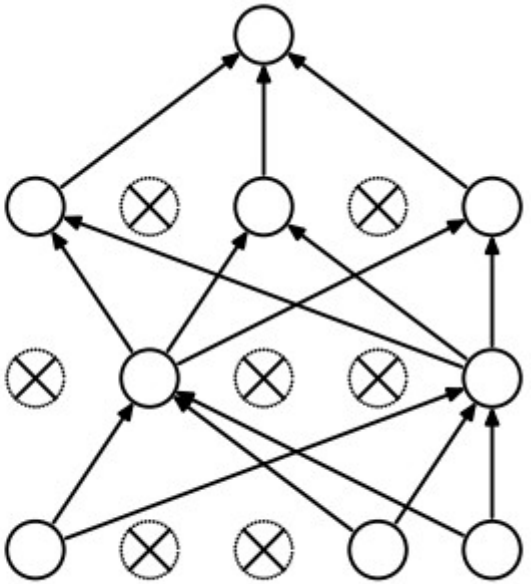


Regularization

- L1, L2, as usual
- Dropout

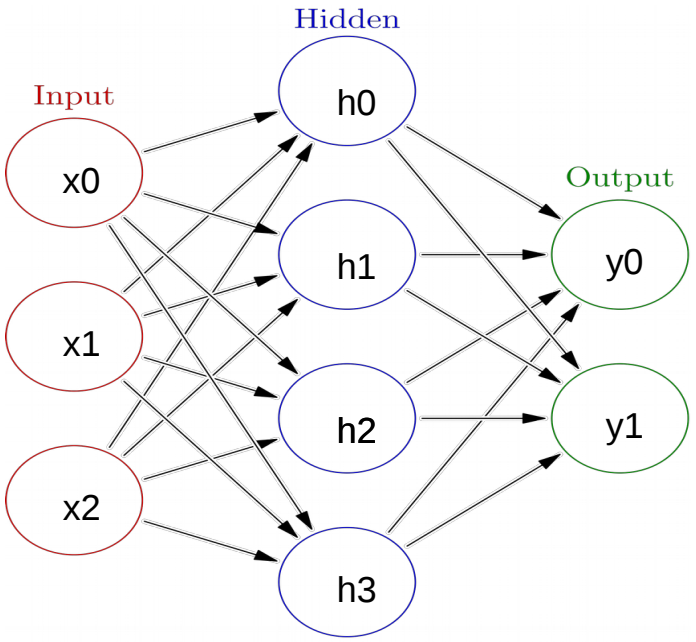


(a) Standard Neural Net



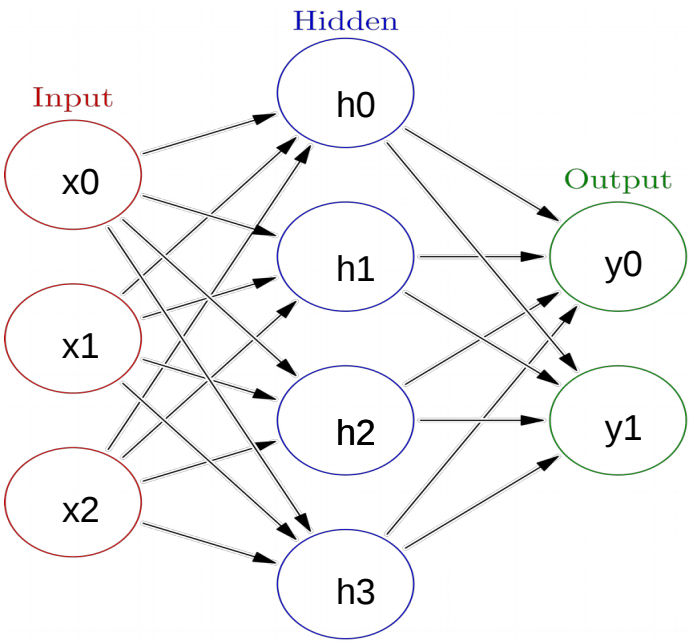
(b) After applying dropout.

Initialization, symmetry problem



- Initialize with zeros
  $W \leftarrow 0$
- What will the first step look like?

Initialization, symmetry problem



- Break the symmetry
- Initialize with random numbers!

Convolutional networks

Image recognition

“Dog”

Why bother?

We study methods for spatial data

- 1D: Time-series, spectrograms
- 2D: Calorimeters; triggers at LHC
- 3D: Hits; tracks; events
- >3D: Spatio-temporal; descriptor manifolds

Image recognition

“Gray wall”

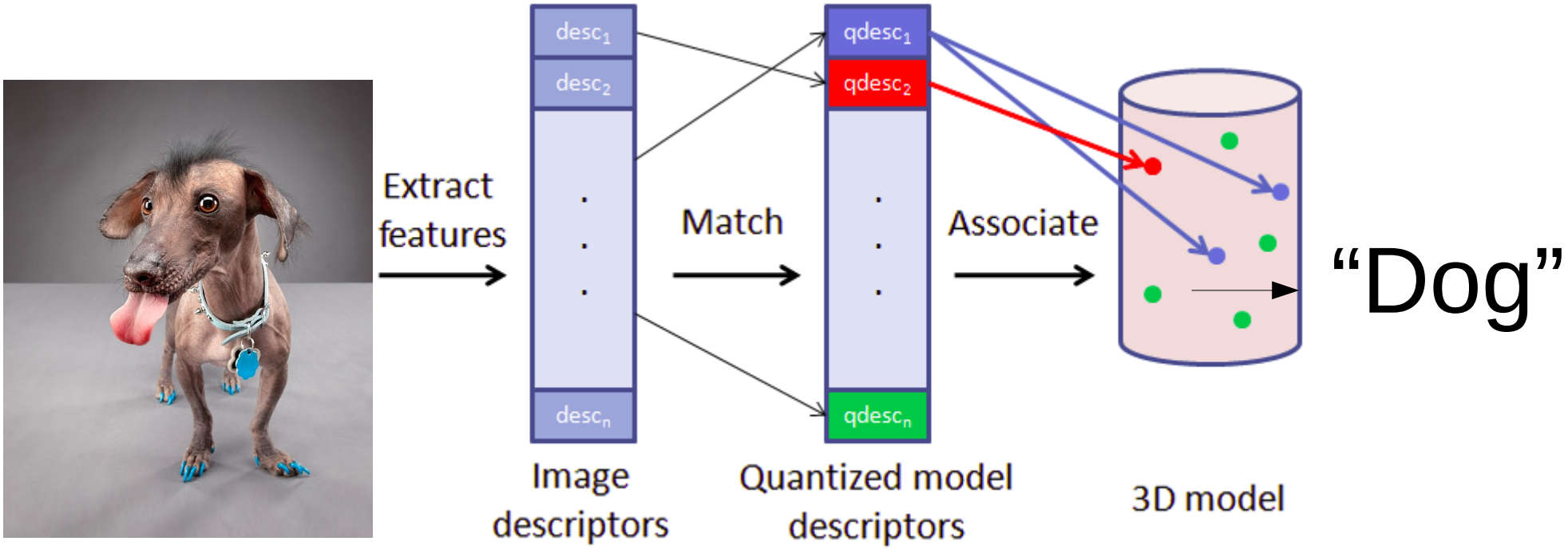
“Dog tongue”

“Dog”

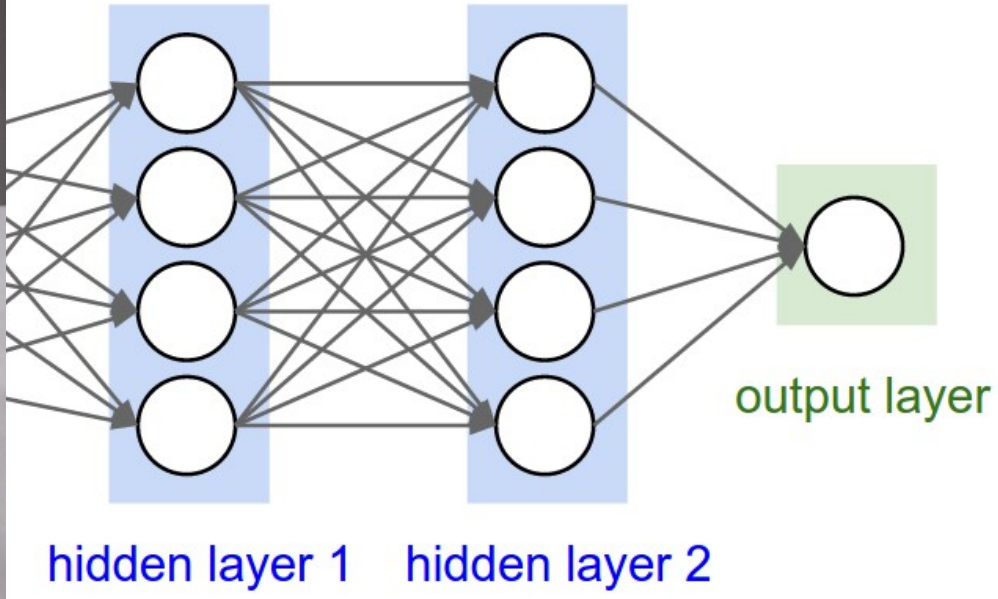
<a particular kind of dog>

“Animal sadism”

Classical approach



NN approach



P(Dog)

What features could NN learn this way?

Problem

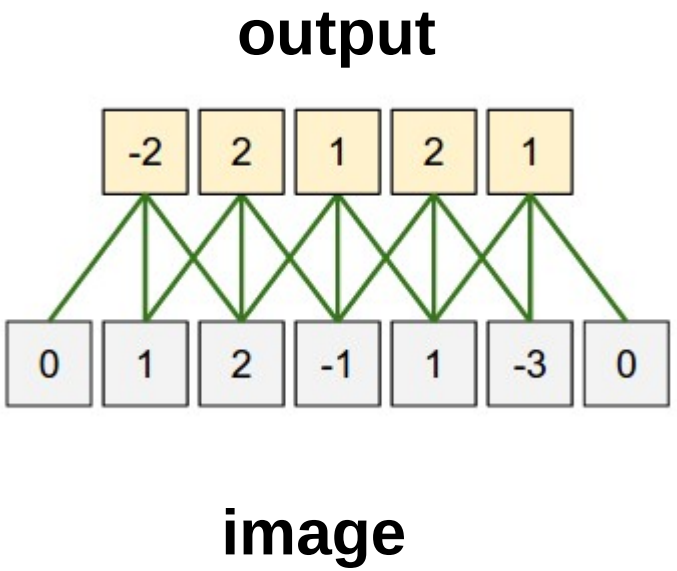
Should we require, say, “Dog ear” feature

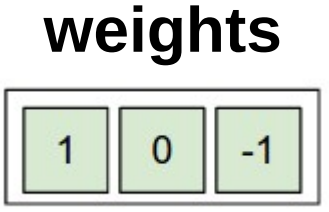
- Linear combination can only select dog ear at a one (or a few) positions.
- Need to learn independent features for each position
- Next layer needs to react on “dog ear 0,0 or dog ear 0,1 or … or dog ear 255,255”
- Introduce **a lot** of parameters and risk overfitting.

Idea: force all these “dog ear” features to use **exactly same weights**, shifting weight matrix each time.

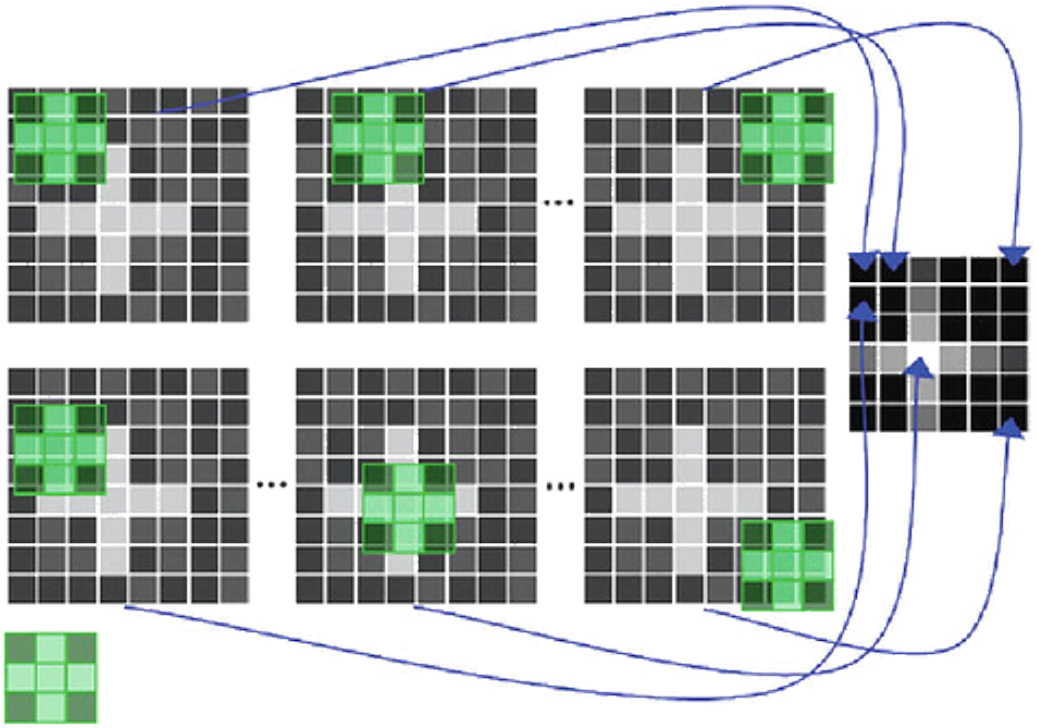
Convolution

- Apply same weights to all patches



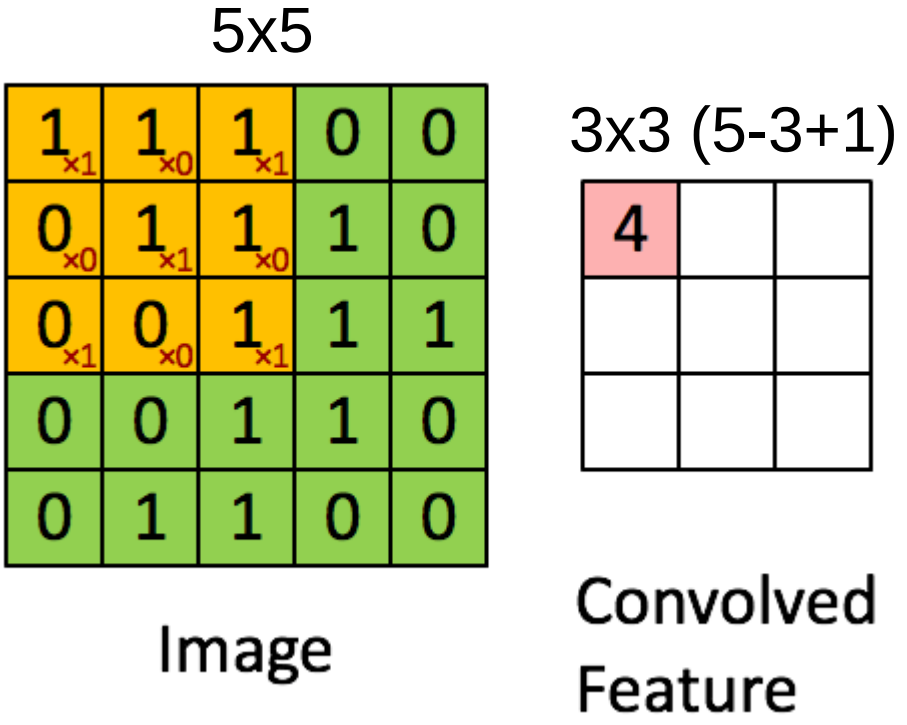


Convolution



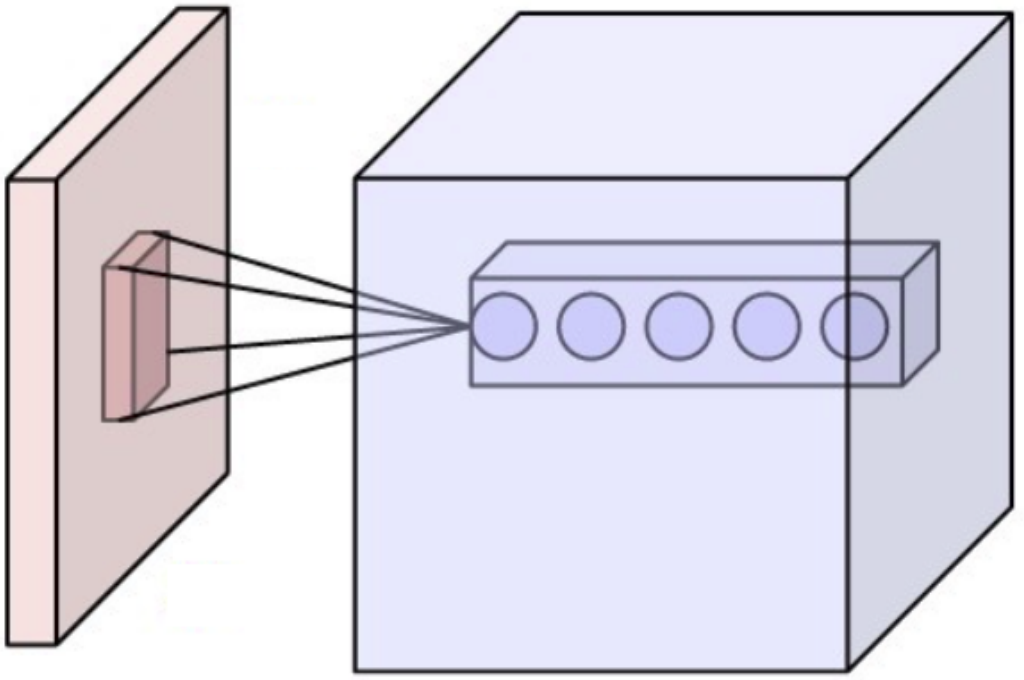
apply same filter to all patches

Convolution



Intuition: how cat-like is this square?

Convolution



Intuition: how cat-like is this square?

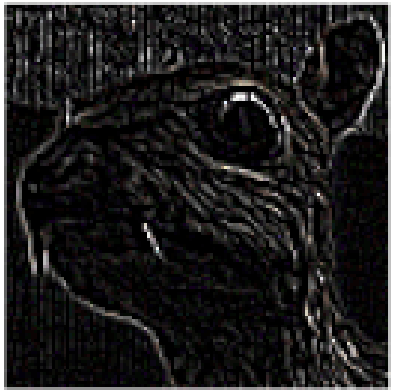
Convolution

Input image

Convolution Kernel

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

Feature map



Intuition: how **edge-like** is this square?

Convolution

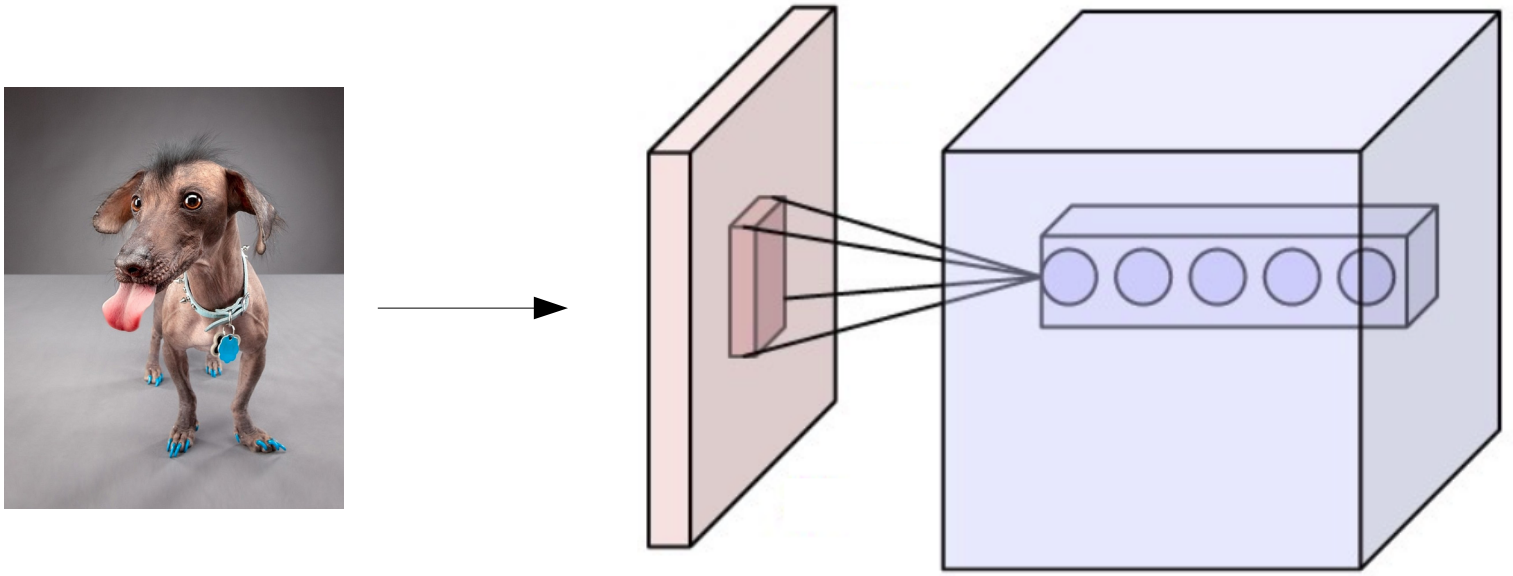
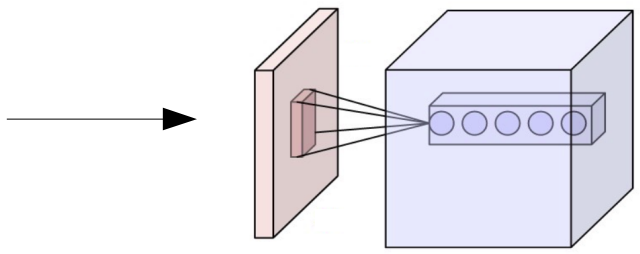


Image : 3 (RGB) x 100 px x 100 px

Convolution



100x96x96

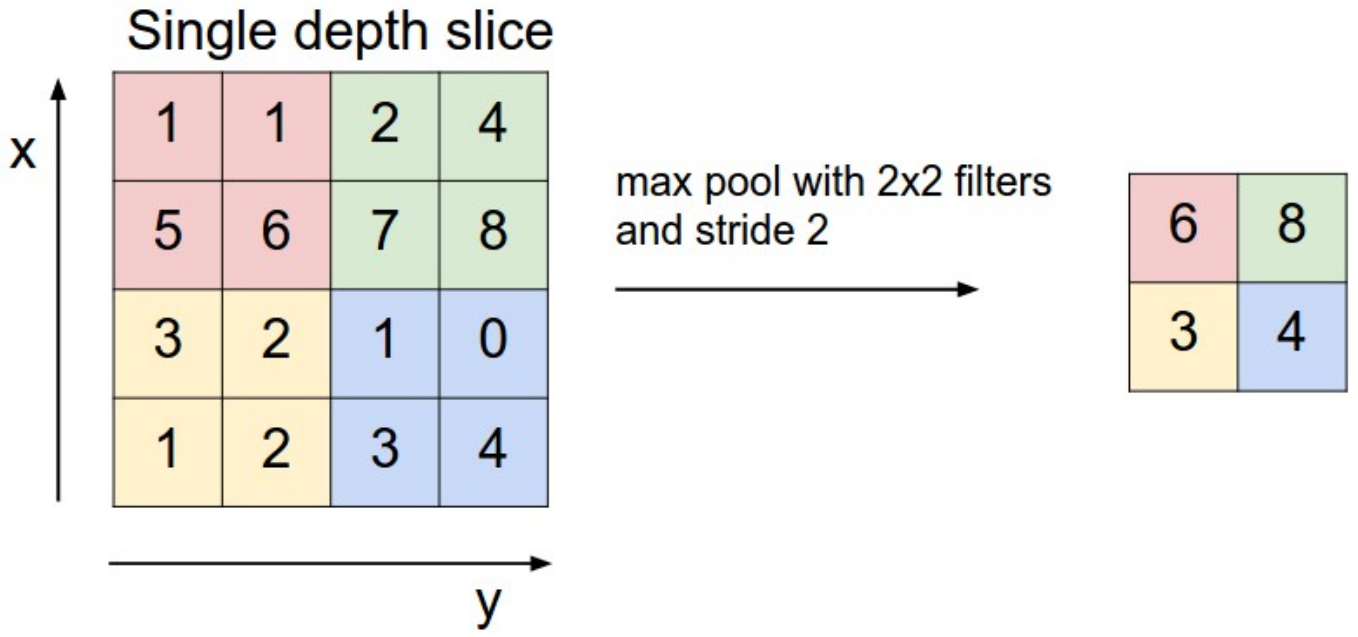
~10^6

Filters: 100x(3x5x5)

Image : 3 (RGB) x 100 px x 100 px

Somewhat too many!

Pooling



Intuition: What is the max **cat**ness in this area?

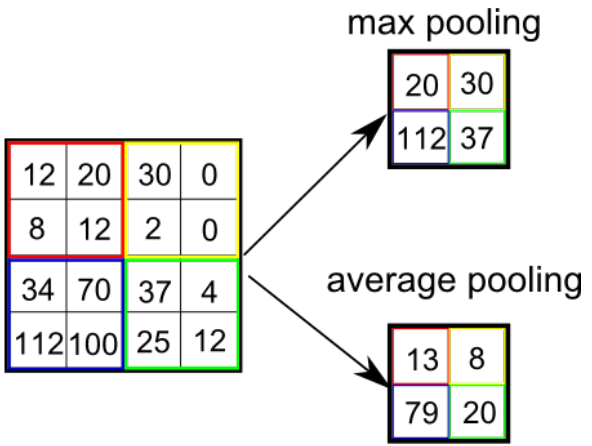
Pooling

Motivation:

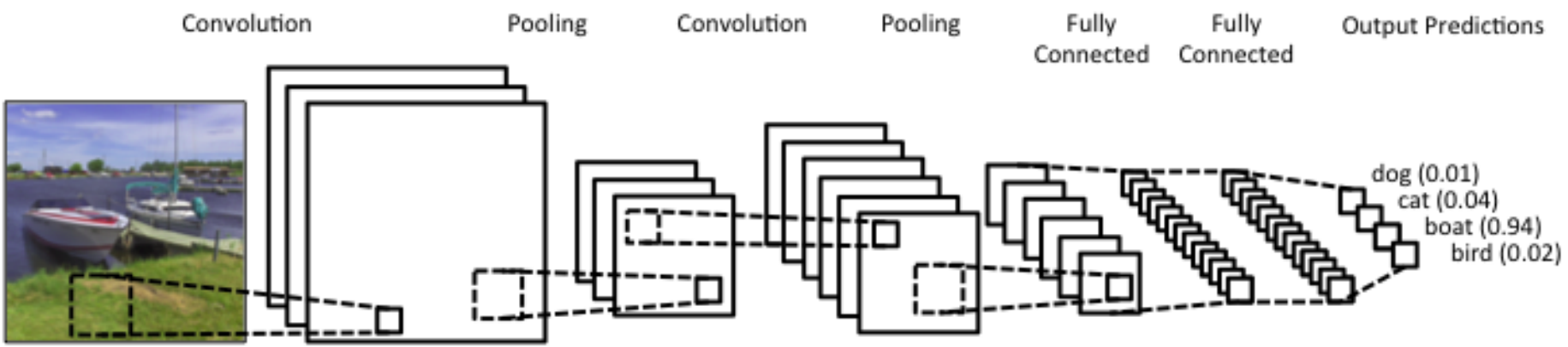
- Reduce layer size by a factor
- Make NN less sensitive to small image shifts

Popular types:

- Max
- Mean(average)

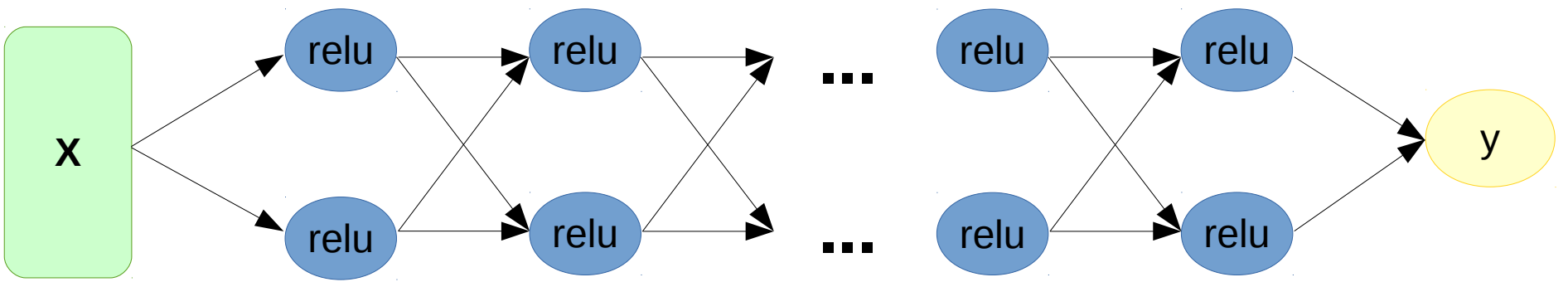


Convolutional NNs



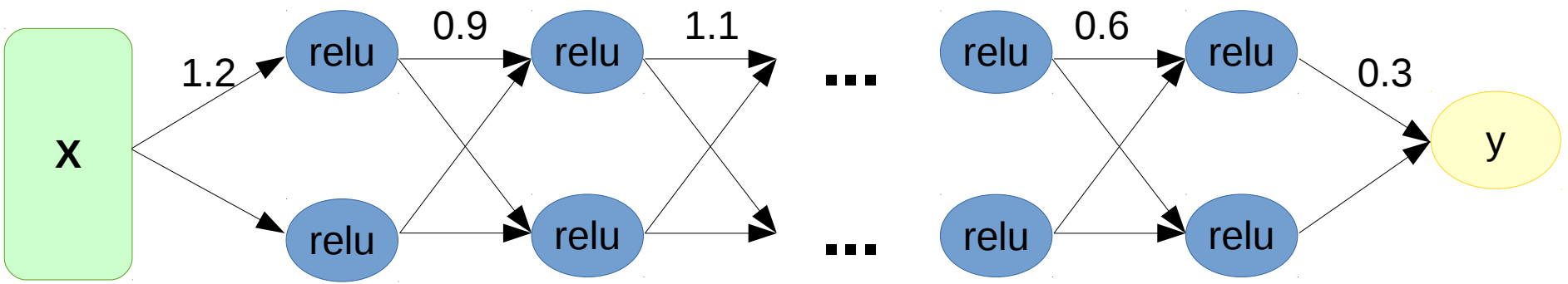
The problem with deep networks

- Imagine a 100-layer network with ReLU



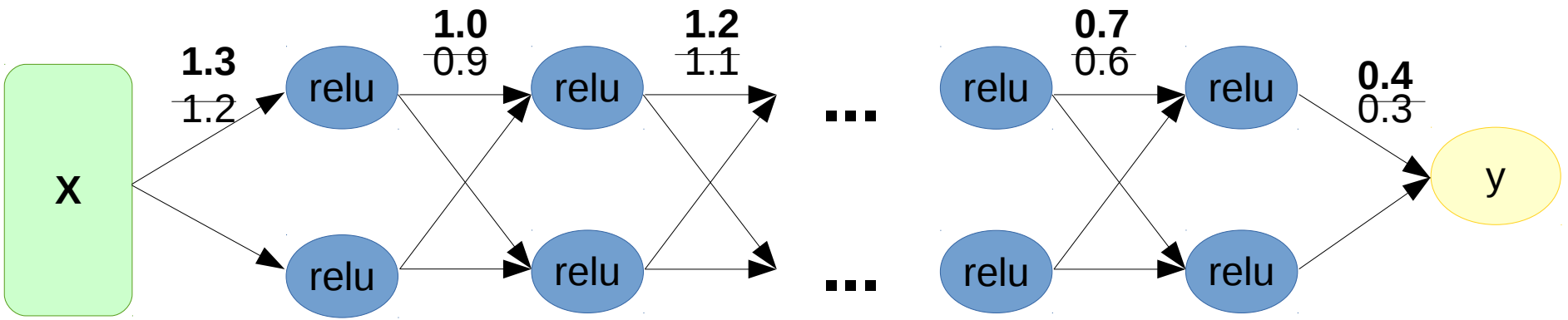
The problem with deep networks

- Imagine a 100-layer network with ReLU



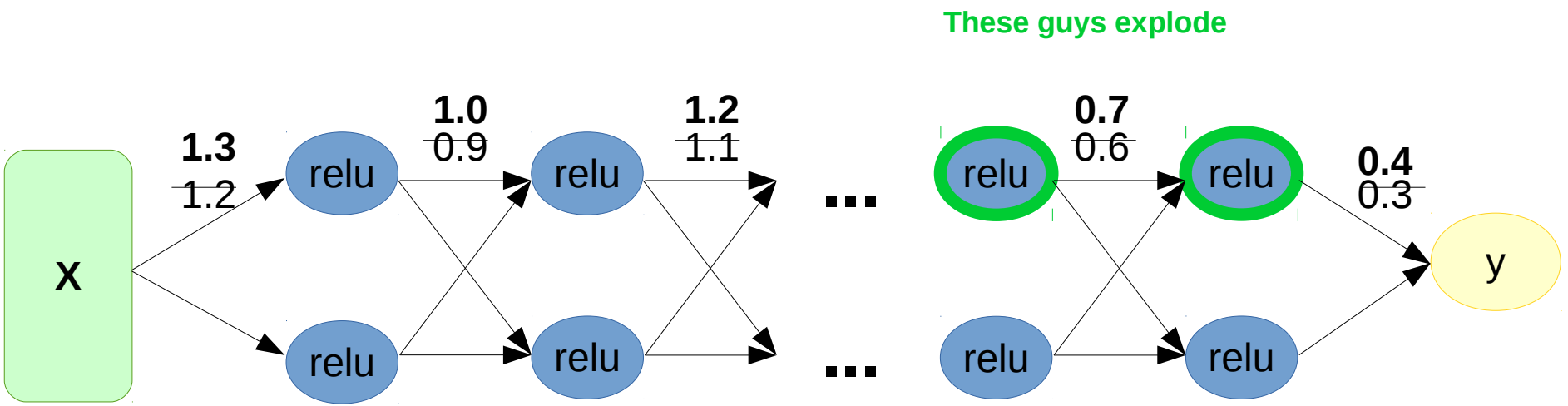
The problem with deep networks

- Imagine a 100-layer network with ReLU
- Single gradient step...



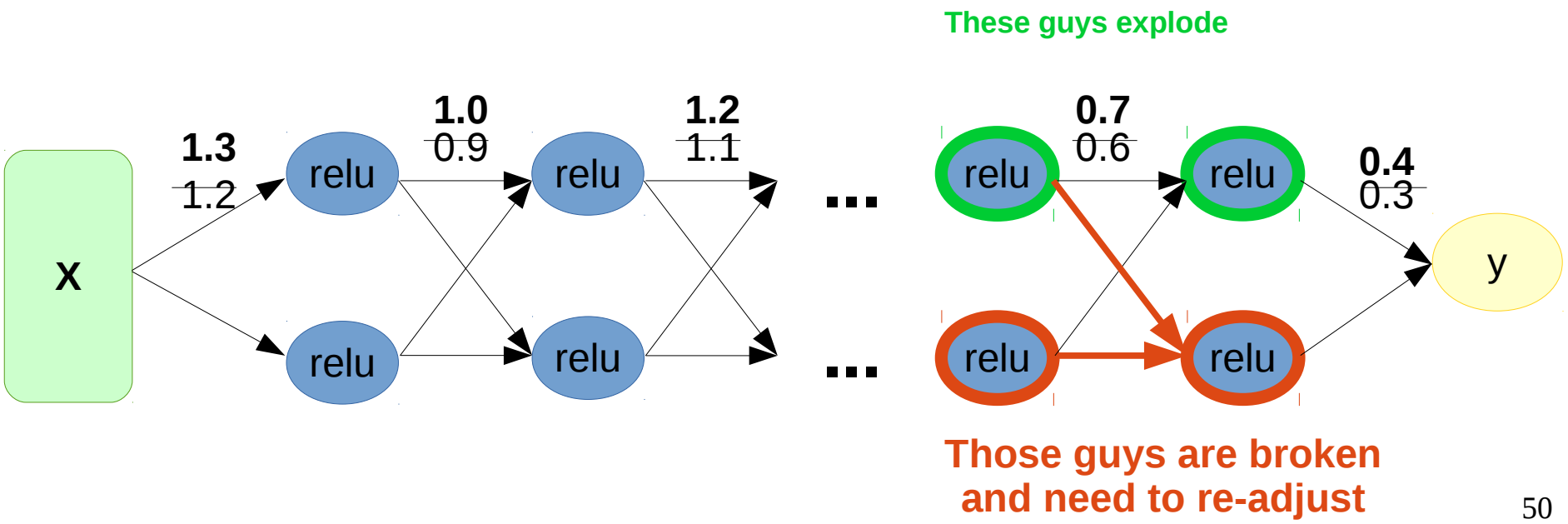
The problem with deep networks

- Imagine a 100-layer network with ReLU
- Single gradient step...



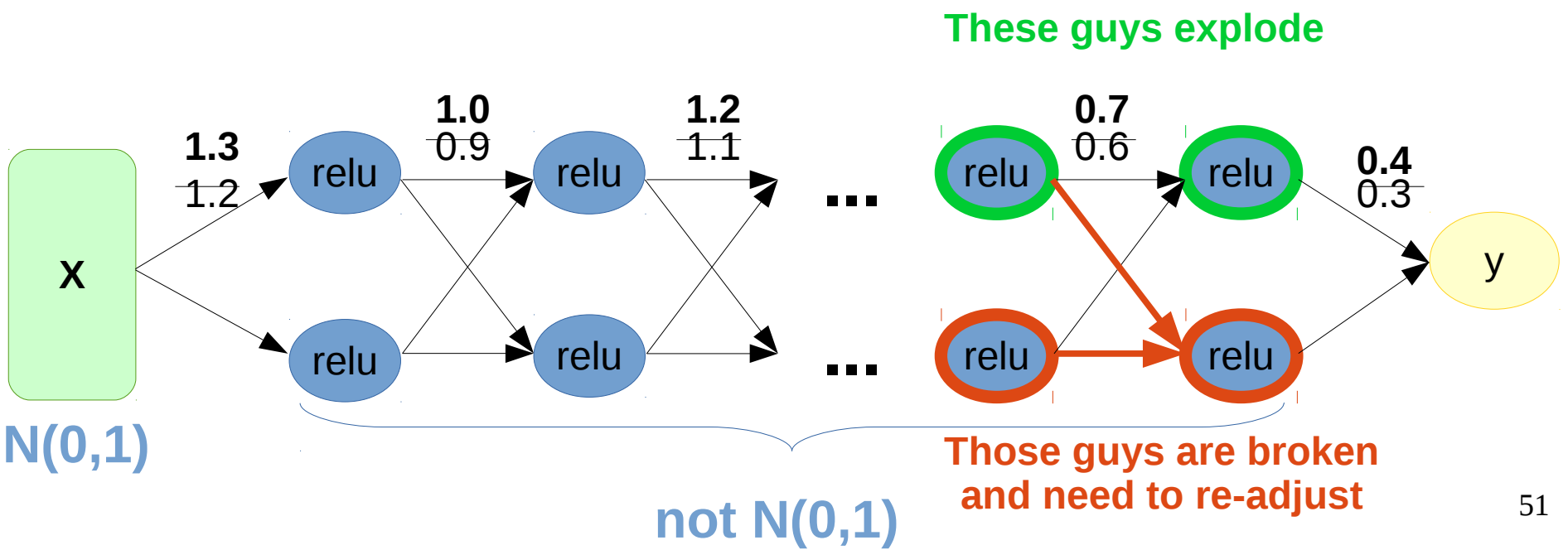
The problem with deep networks

- Imagine a 100-layer network with ReLU
- Single gradient step...



The problem with deep networks

- Imagine a 100-layer network with ReLU
- Single gradient step...



Batch normalization

TL;DR:

- It's usually a good idea to normalize linear model inputs

(c) Every machine learning lecturer, ever

Batch normalization

Idea:

- We normalize activation of a hidden layer

  (zero mean unit variance)

$$h_i = \frac{h_i - \mu_i}{\sqrt{\sigma_i^2}}$$

i stands for i-th neuron

- Update μ_i, σ_i^2 with moving average while training

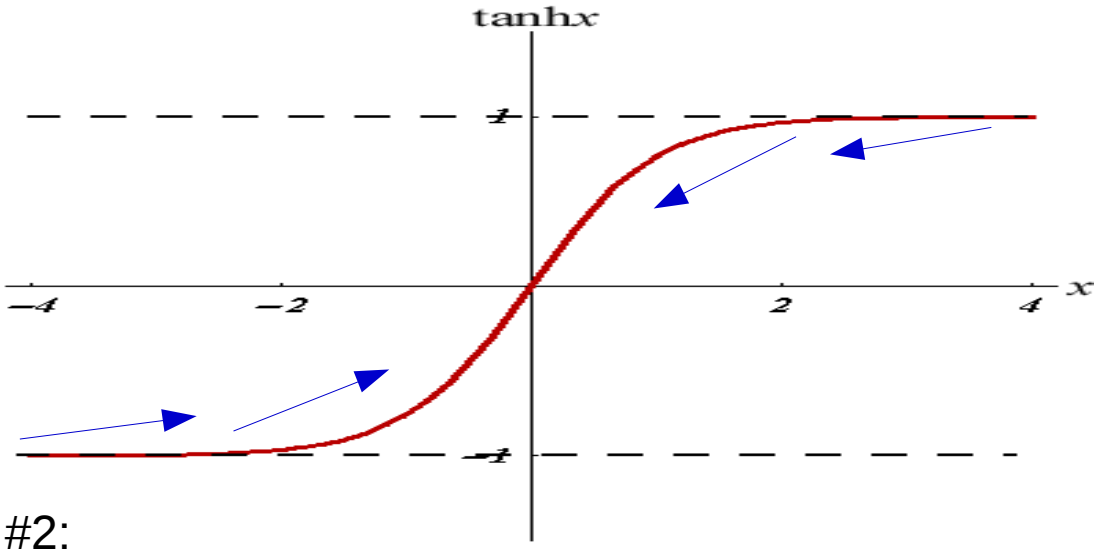
$$\mu_i := \alpha \cdot mean_{batch} + (1-\alpha) \cdot \mu_i$$

$$\sigma_i^2 := \alpha \cdot variance_{batch} + (1-\alpha) \cdot \sigma_i^2$$

Batch normalization

Good side effect #1:

- Vanishing gradient less a problem for sigmoid-like nonlinearities



Good side effect #2:

- We no longer need to train bias (+b term in Wx+b)

Weight normalization

Same problem, different solution

- Learn separate “direction” w and “length” l

$$\hat{w} \stackrel{\text{def}}{=} \frac{w}{||w||} \cdot l$$

- Much simpler, but requires good init

Data augmentation

- Idea: we can get N times more data by tweaking images.
- A cat rotated 15° clockwise is still a cat

- Rotate, crop, zoom, flip horizontally, add noise, etc.
- Sound data: add background noises

Other CV applications

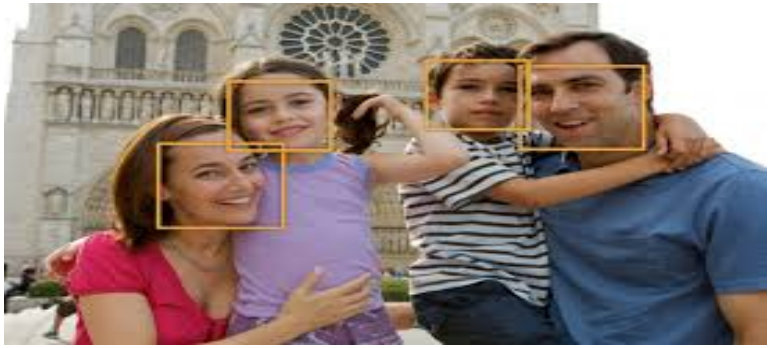
Real computer vision starts when
image classification is no longer enough.

Bounding box regression

Predict object bounding box

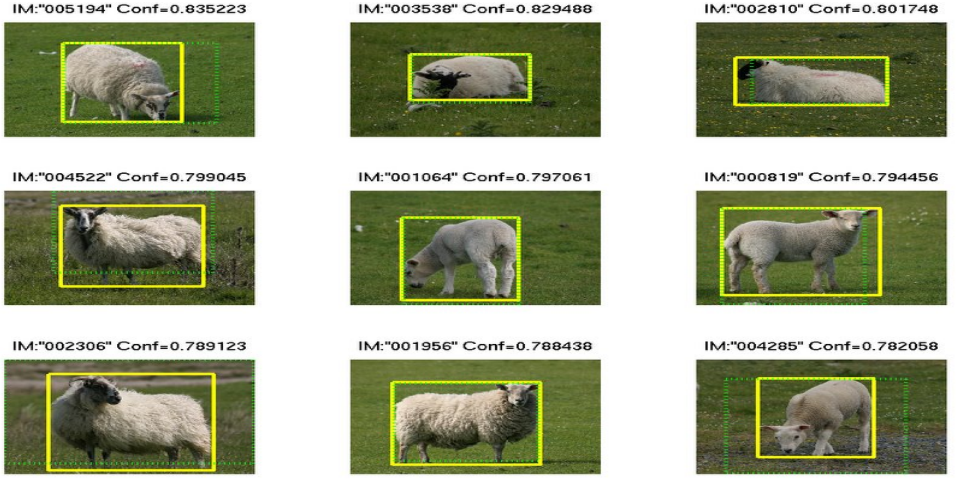
(x0,y0,w,h)

or several bounding boxes for multiple objects.



Applications examples:

- Face detection @ cameras
- Surveillance cameras
- Self-driving cars



Segmentation

Predict class for each pixel

(fully-convolutional networks)

Applications examples:

- More surveillance
- Brain scan labeling
- Map labeling

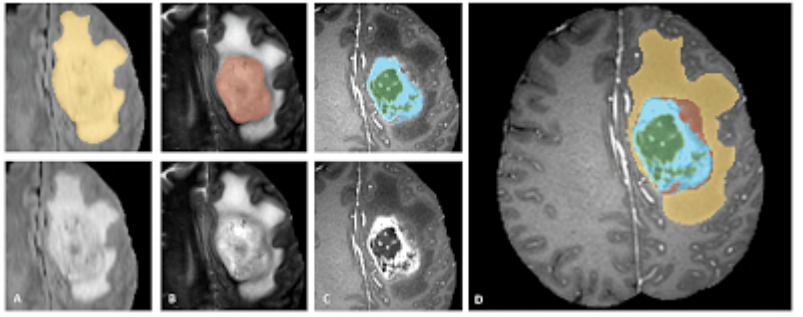
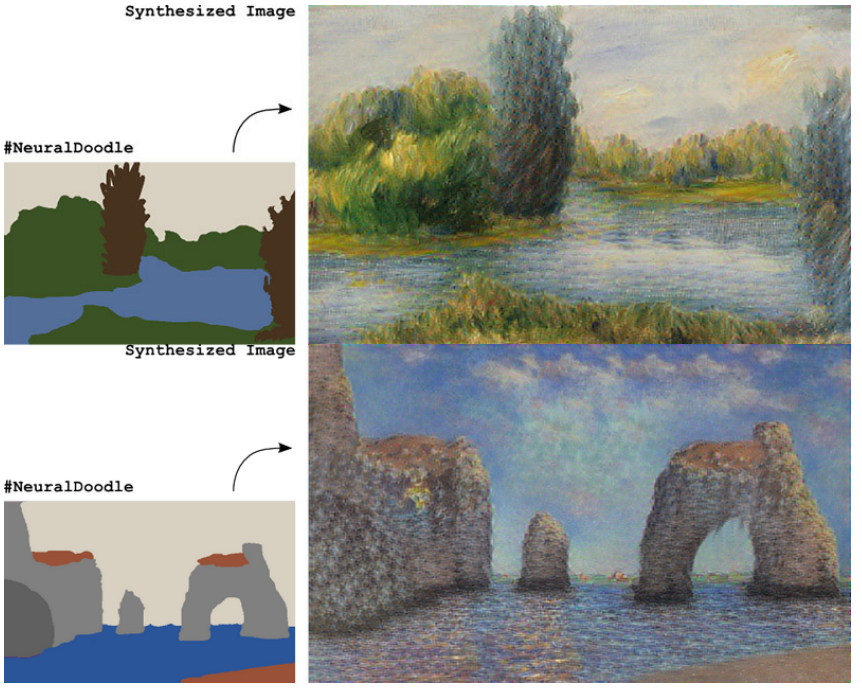


Image generation/transformation

- **Generation:** Given a set of reference images, learn to generate new images, resembling those you were given.

- **Transformation:** Given a set of reference images, learn to convert other images into ones resembling the reference set.



Neural Doodle
(D. Ulyanov et al.)

Image tagging
Image captioning
Image retrieval
Image encoding
Image morphing
Image encoding
Image upscaling
Object tracking on video
Video processing
Video interpolation

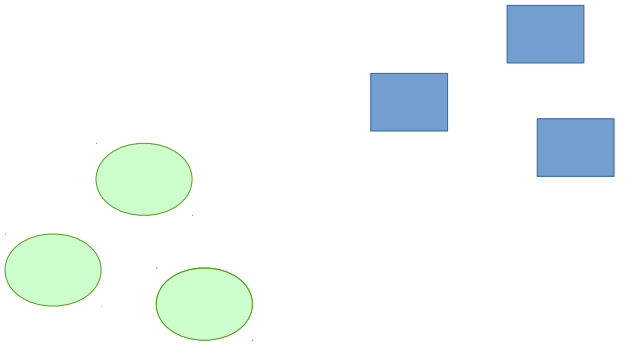
Fine-tuning
Adversarial Networks
Variational Autoencoders
Knowledge transfer
Domain adaptation
Online learning
Explaining predictions
Soft targets
Scene reconstruction
3D object retrieval
Classifier optimization

Unsupervised learning

Supervised vs Unsupervised

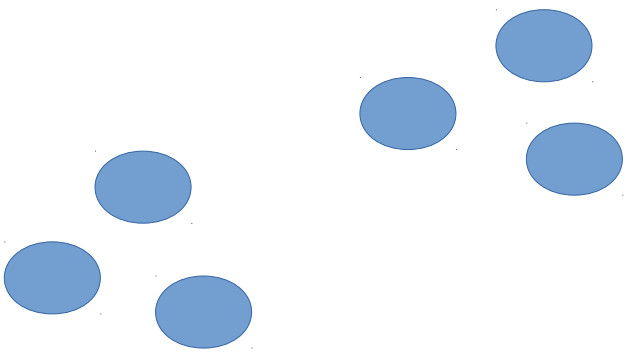
Supervised learning

- Take (x,y) pairs



Unsupervised learning

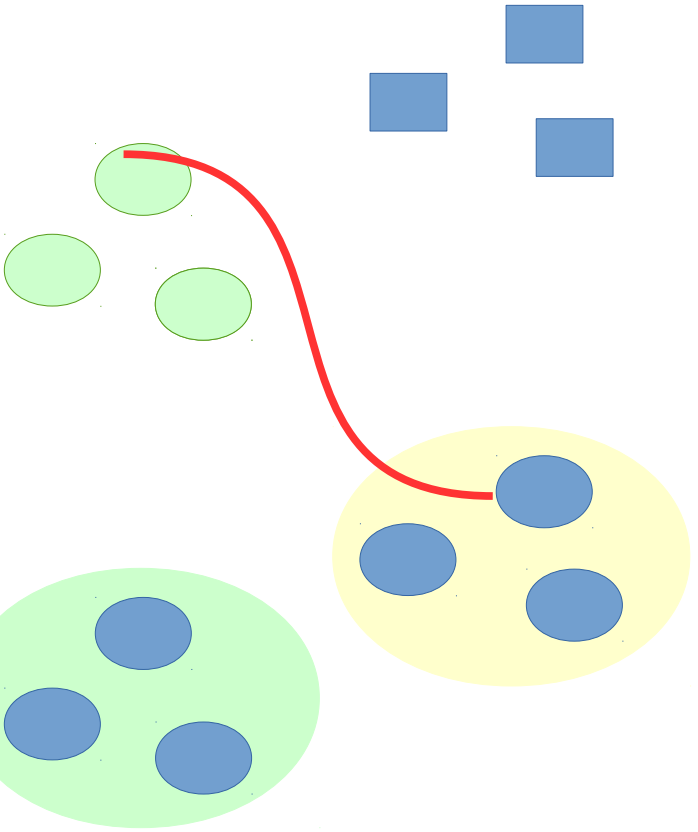
- Take x alone



Supervised vs Unsupervised

Supervised learning

- Take (x,y) pairs
- Learn mapping x → y



Unsupervised learning

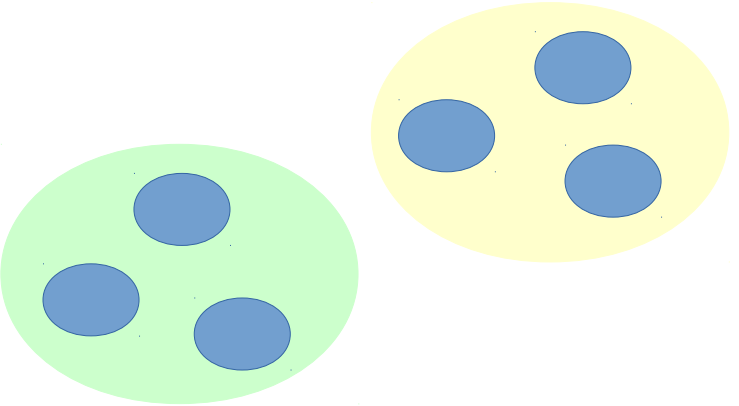
- Take unlabeled x
- Learn hidden structure

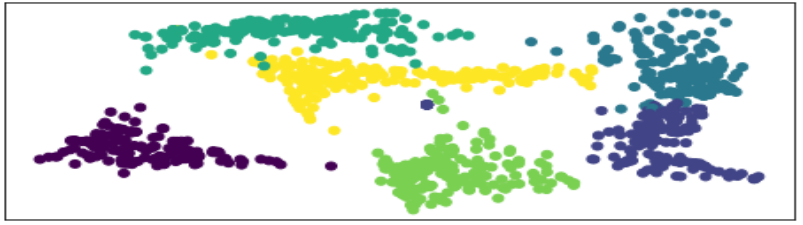
  behind the data

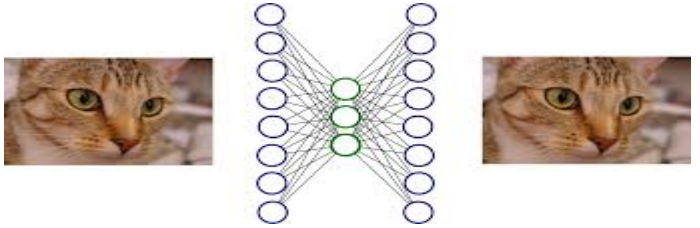
Why bother?

Unsupervised learning:

- Dimensionality reduction
- Find great features
- Explore high-dim data
- Generate new samples



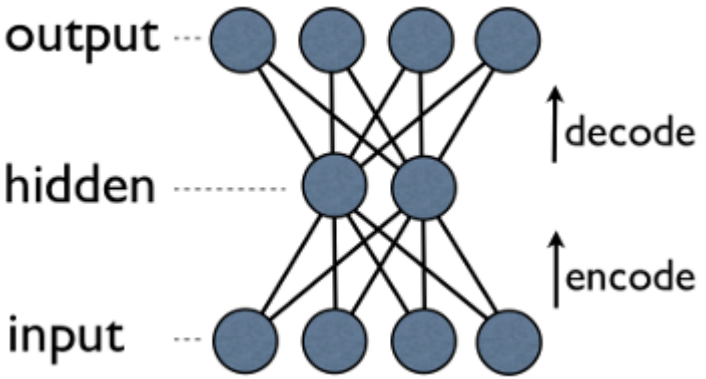




Autoencoders 101

Main idea:

- Take data in some original (high-dimensional) space;
- Project data into a new space **from which it can then be accurately restored;**
- Encoder = data to hidden
- Decoder = hidden to data
- Decoder(Encoder(x)) ~ x



Why do we ever need that?

- Dimensionality reduction
  - |code| << |data|

<to be continued>

Matrix decompositions

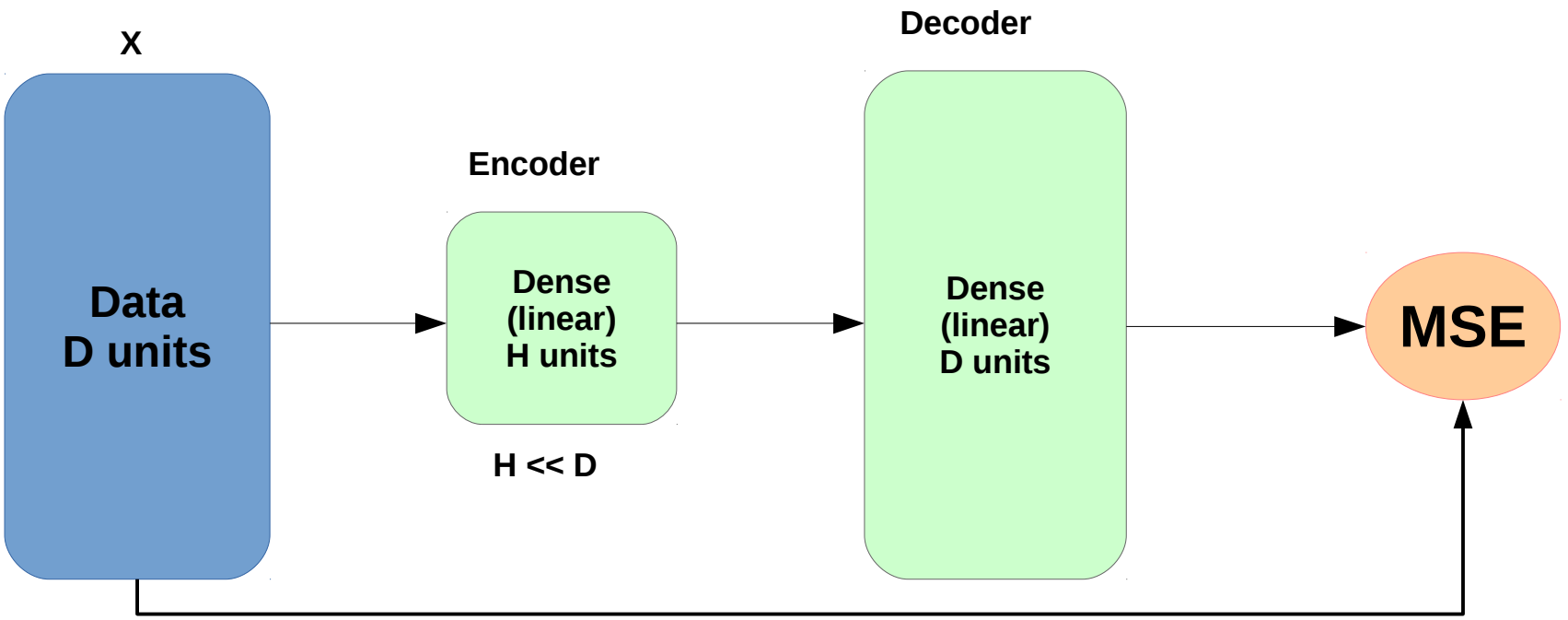
Example: matrix factorization (PCA, SVD)

$$\|X - U \cdot V^T\| \rightarrow \min_{U,V}$$

Minimizing reconstruction error

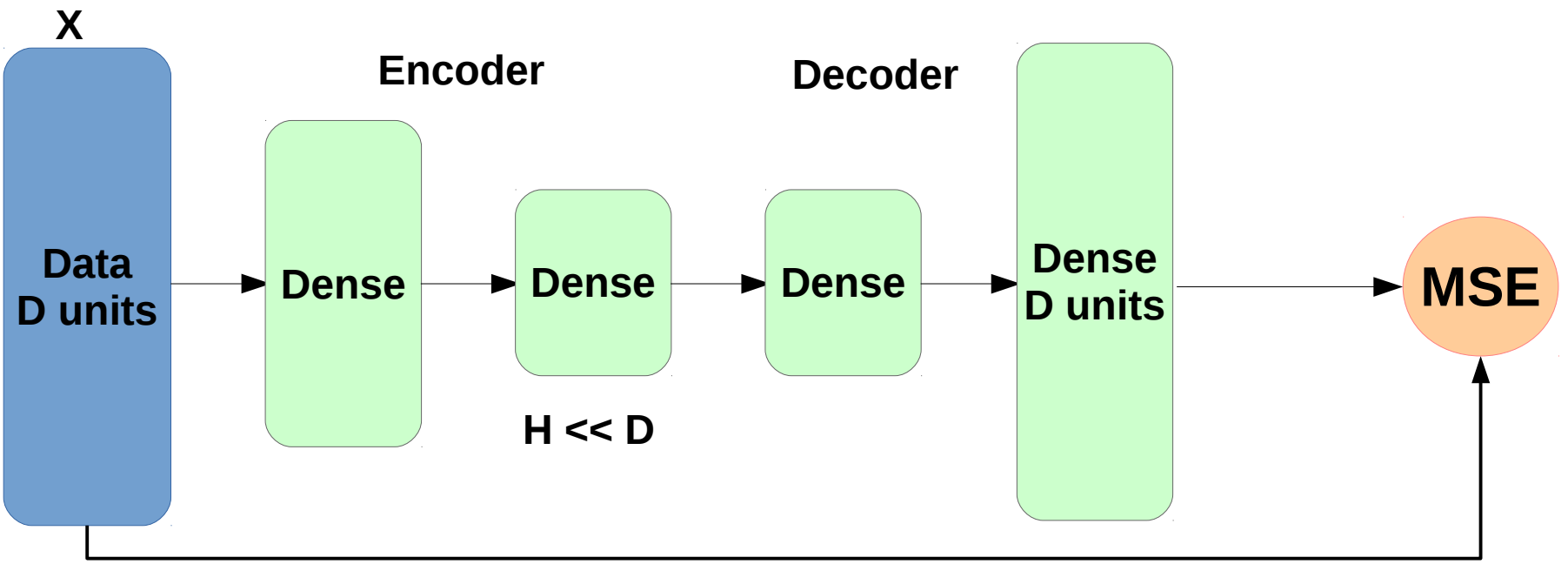
Matrix decomposition

- A different perspective



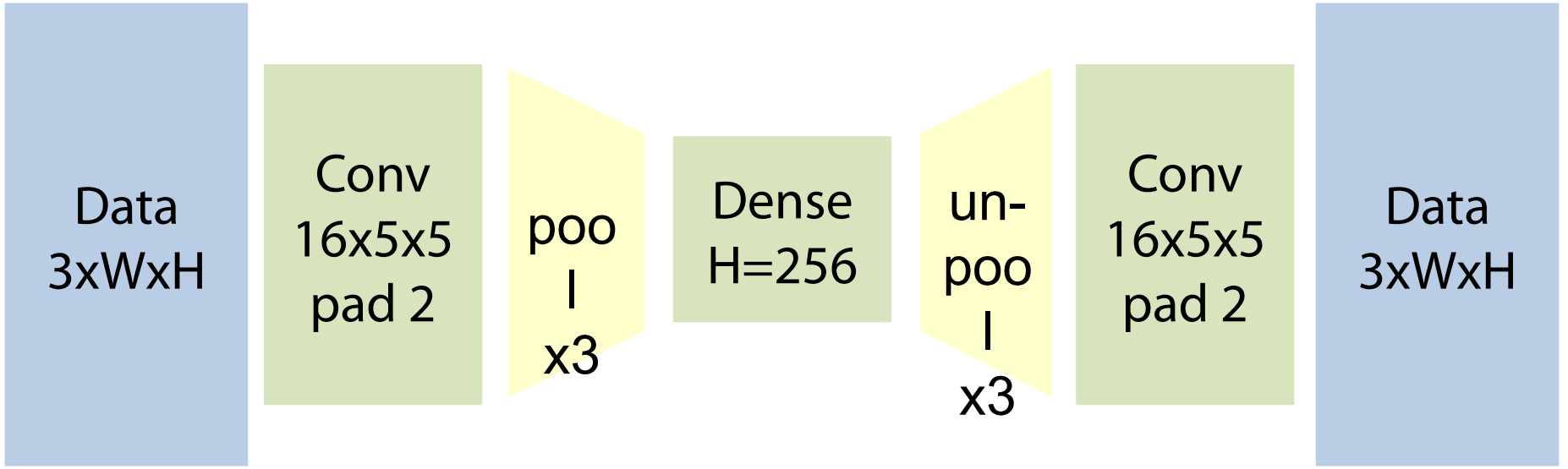
(kinda) Deep autoencoder

- Stack more layers!



Quiz: What if data is an image?

Convolutional autoencoders

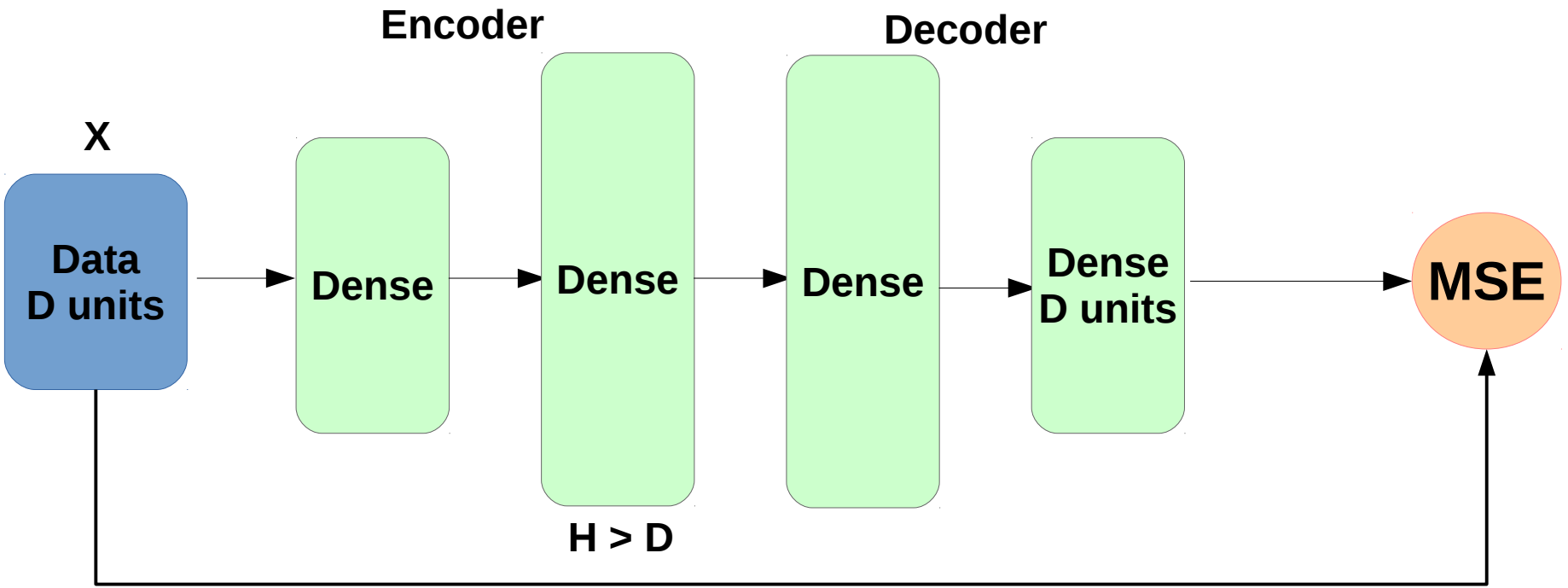


Why do we ever need that?

- Dimensionality reduction
  - |code| << |data|
- **Learn some great features!**
  - Before feeding data to your XGBoost

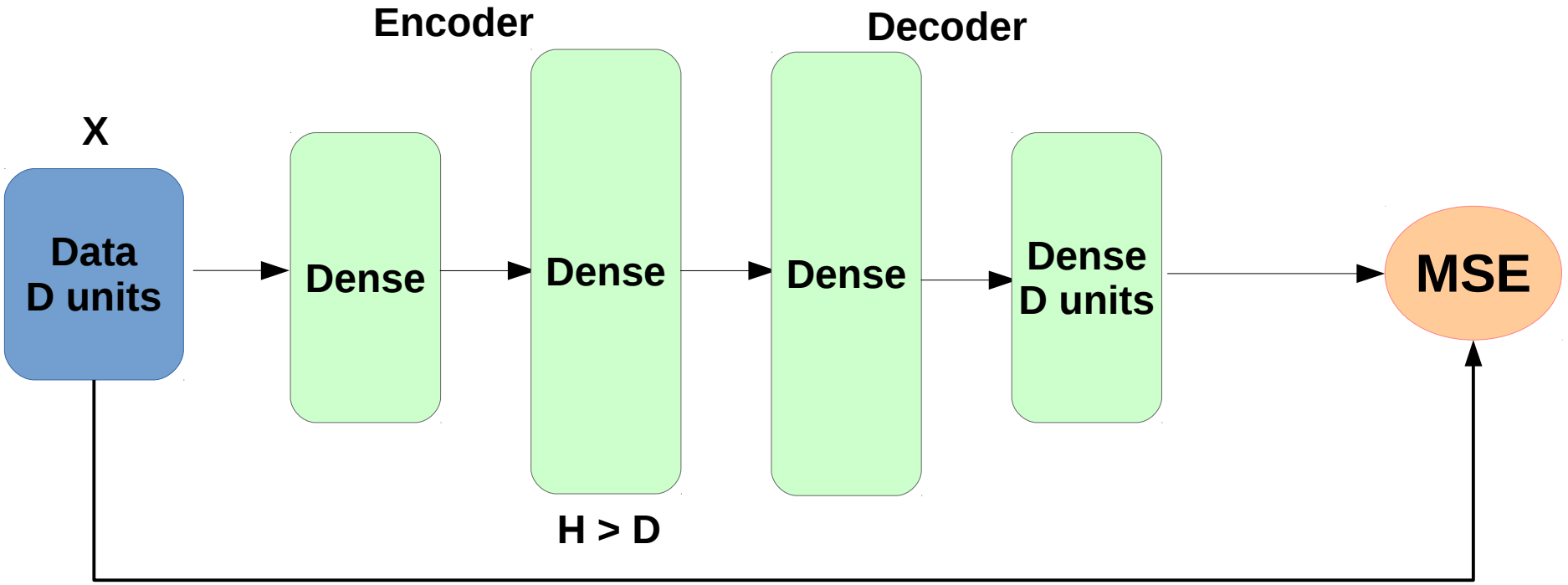
Expanding autoencoder

- Bigger/richer representation



Expanding autoencoder

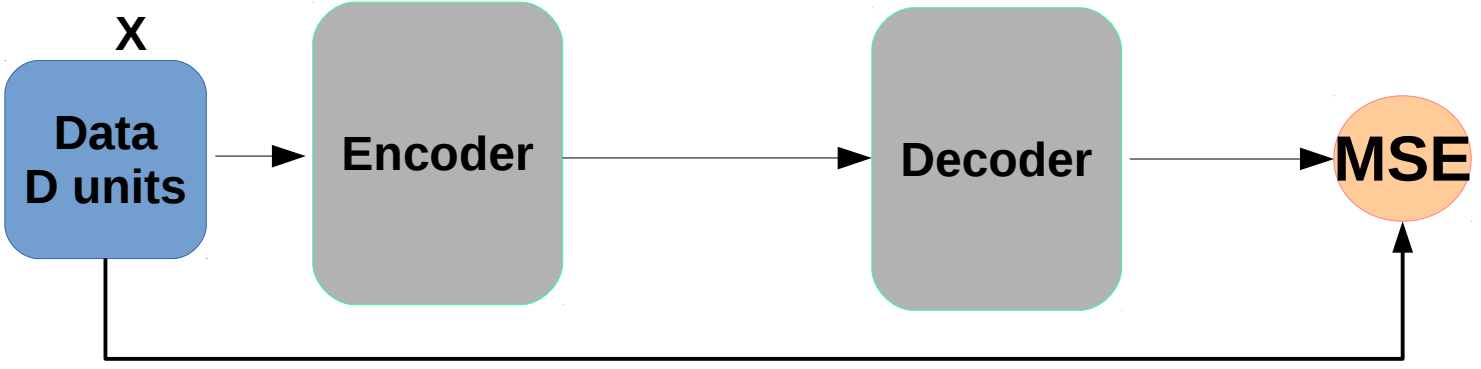
- Bigger/richer representation



Something's wrong with this guy. **Ideas?**

Expanding autoencoder

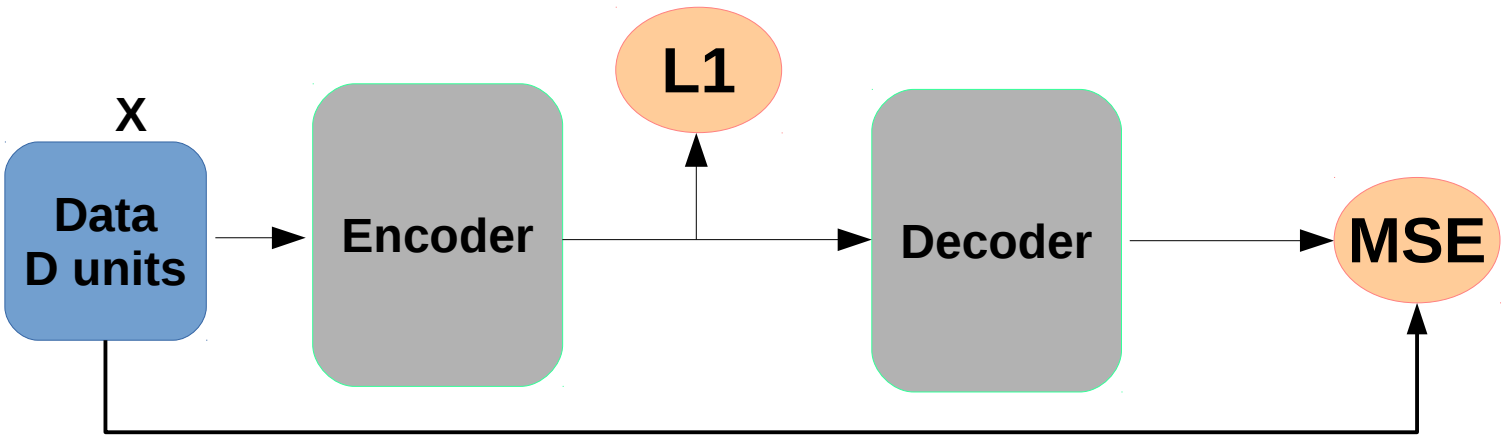
- Naive approach will learn identity function!
- Gotta regularize!



$$L = \| X - Dec\left(Enc\left(X\right)\right) \|$$

Sparse autoencoder

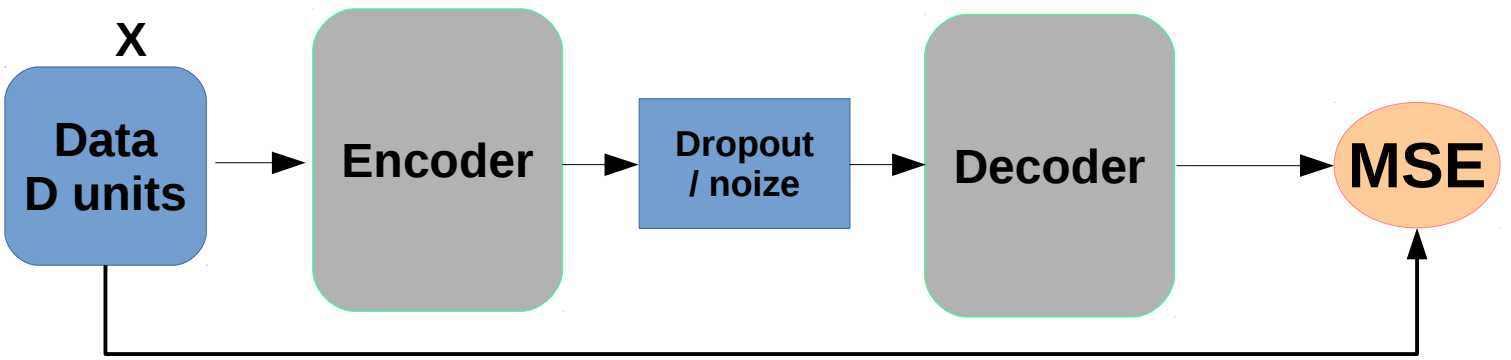
- Naive approach will learn identity function!
- Idea 1: L1 on **activations**, sparse code



$$L = \|X - Dec(Enc(X))\| + \sum_i |Enc_i(X)|$$

Redundant autoencoder

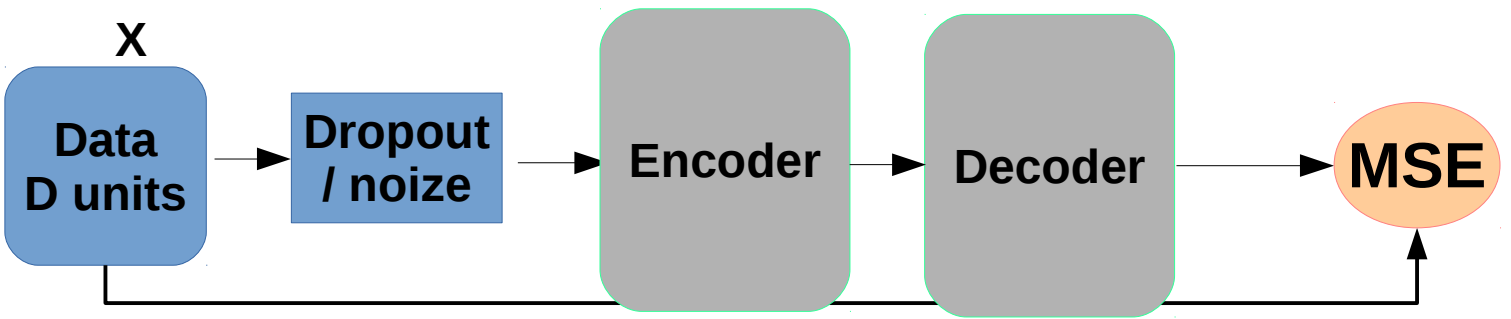
- Naive approach will learn identity function!
- Idea 2: noize/dropout, redundant code



$$L = \| X - Enc(Noize(Dec(X))) \|$$

Denoising autoencoder

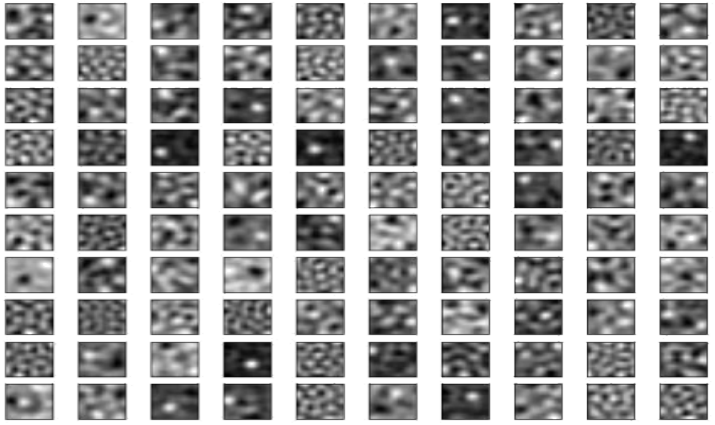
- Naive approach will learn identity function!
- Idea 3: distort input, learn to undo distortion



$$L = \| X - Enc(Dec(Noize(X))) \|$$

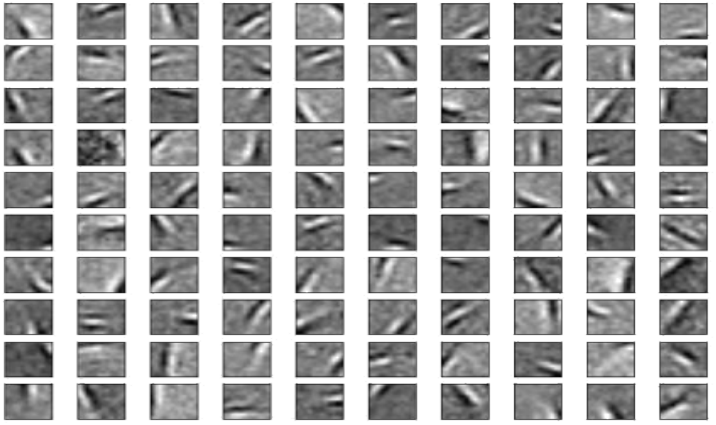
Sparse Vs Denoising

- Filter weights, 12x12 patches



Sparse AE

Actually meaningless :)



Denoizing AE

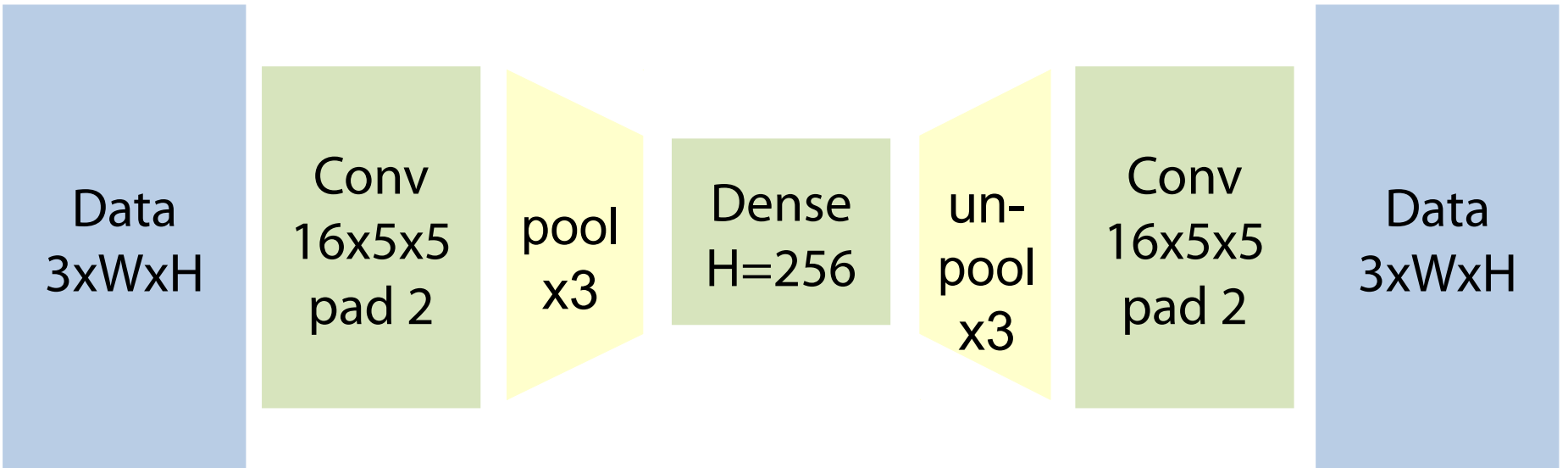
[Vincent et al. 2010]

Why do we ever need that?

- Dimensionality reduction
  - |code| << |data|
- Learn some great features!
  - Before feeding data to your XGBoost
- **Unsupervised pretraining**
  - Exploit unlabeled data to improve classifier

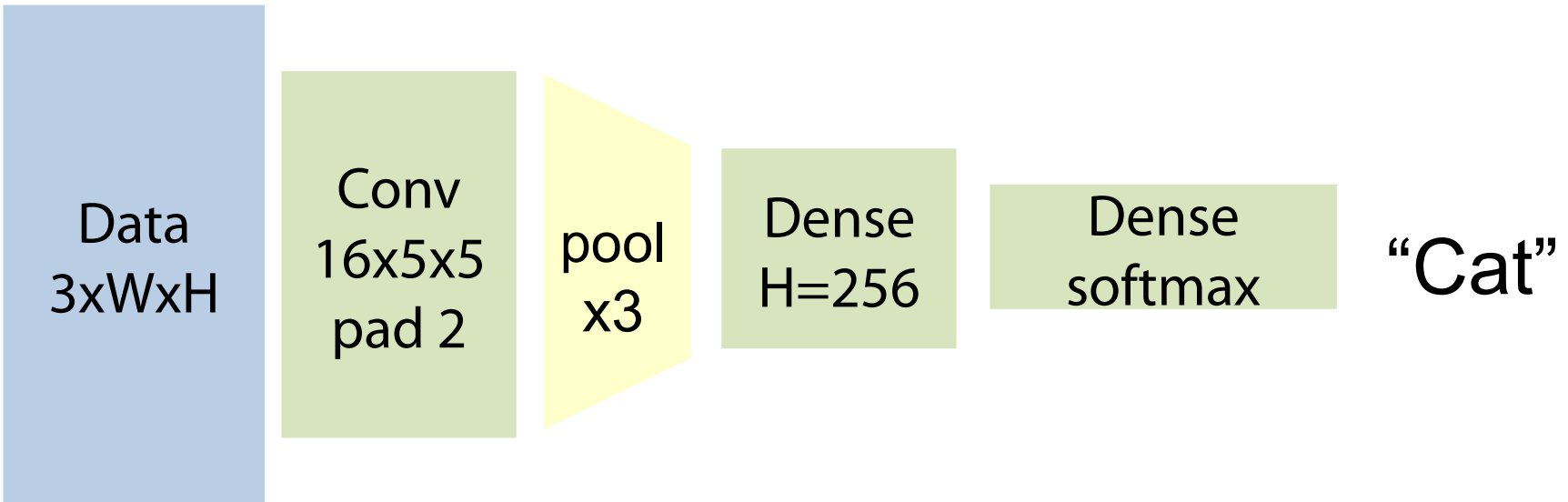
Unsupervised pre-training

Step 1: train autoencoder



Unsupervised pre-training

Use autoencoder as initialization

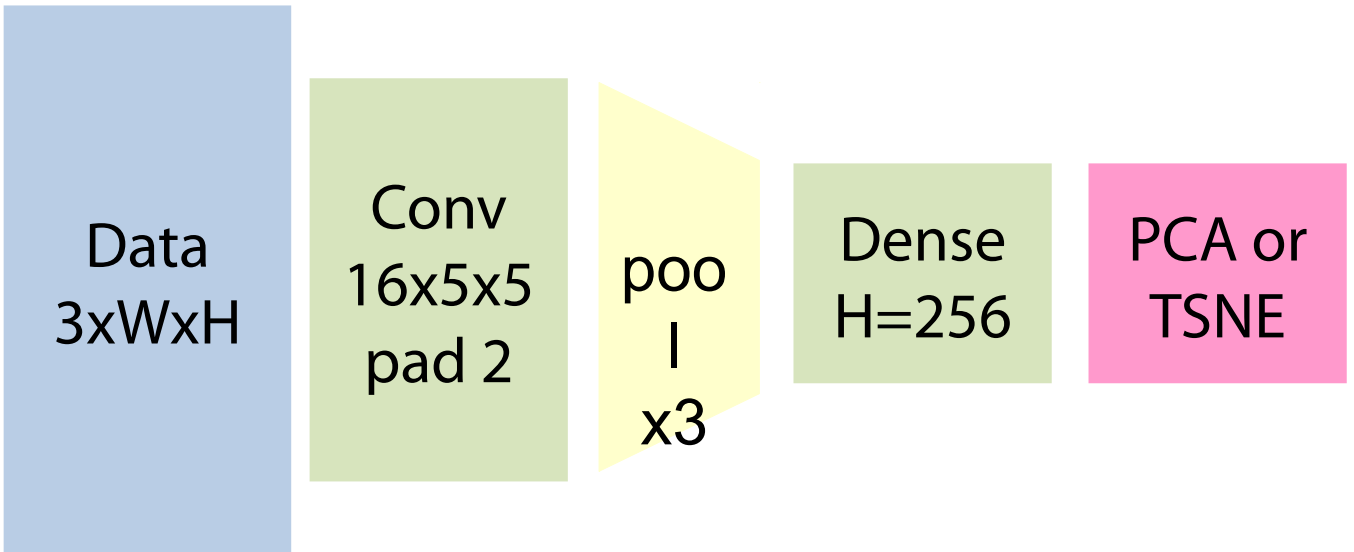


Why do we ever need that?

- Dimensionality reduction
  - |code| << |data|
- Learn some great features!
  - Before feeding data to your XGBoost
- Unsupervised pretraining
  - Exploit unlabeled data to improve classifier
- Visualizing data structure

Exploratory analysis

Visualize data in hidden space



Exploratory analysis

Visualize data in hidden space

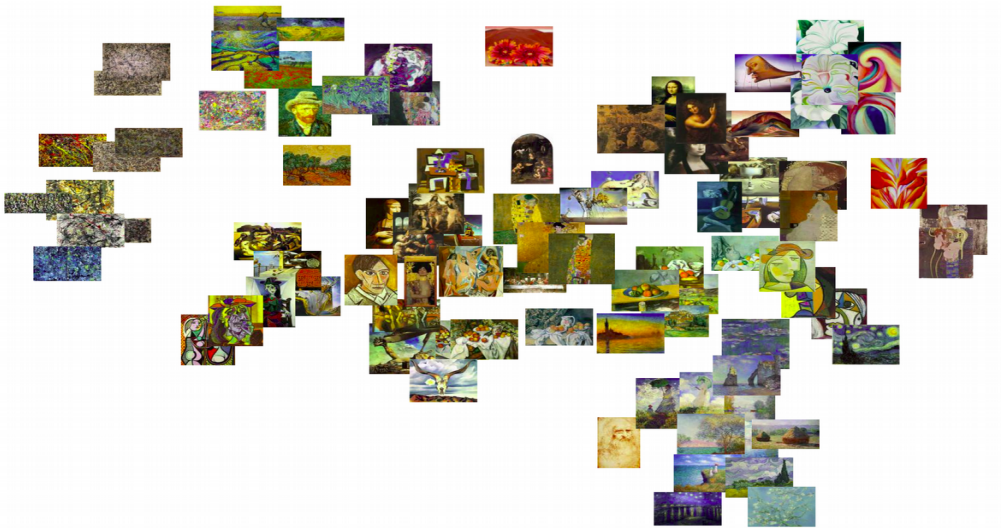


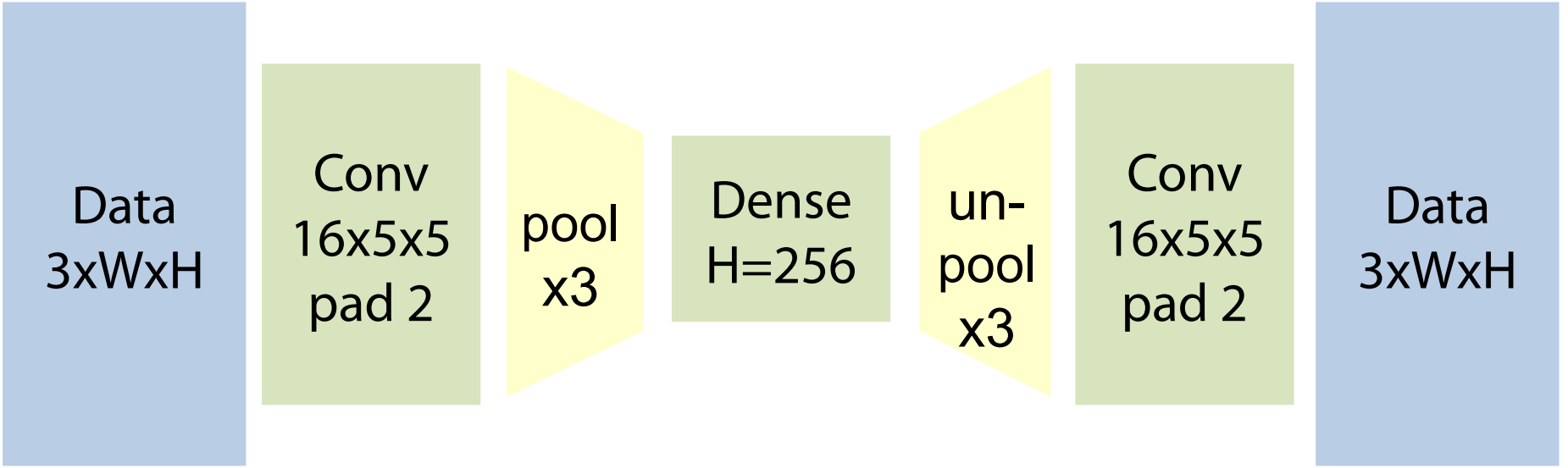
Image: https://razi.xyz/vgg2vec/picasso

Why do we ever need that?

- Dimensionality reduction
  - |code| << |data|
- Learn some great features!
  - Before feeding data to your XGBoost
- Unsupervised pretraining
  - Exploit unlabeled data to improve classifier
- Visualizing data structure
- **Generating new data**
  - Your trainable monte-carlo

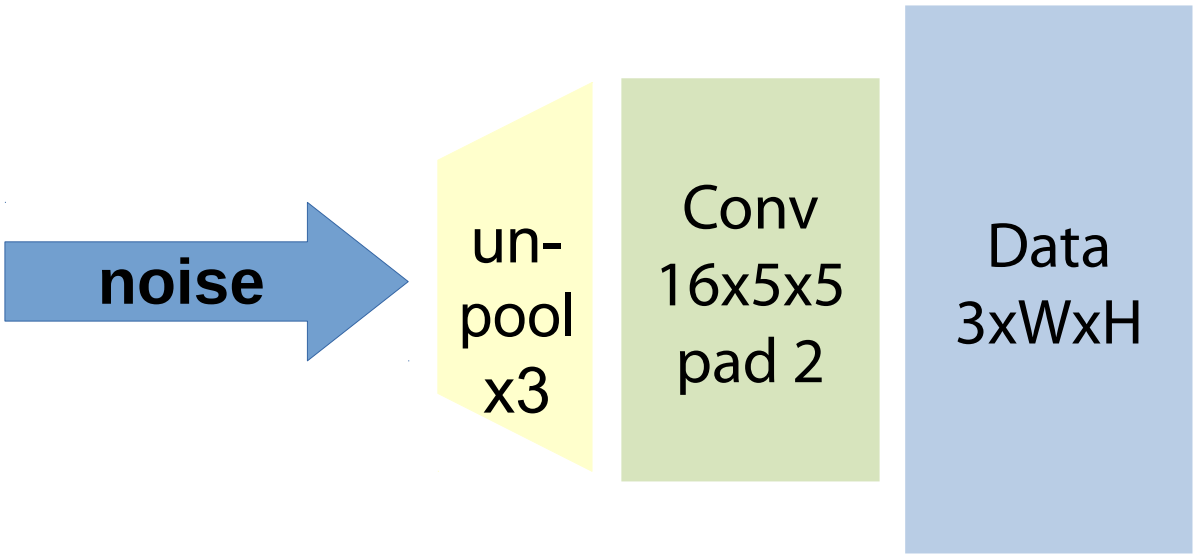
Generating images

Step 1: train autoencoder



Generating images

Step 2: use decoder to generate data



Disclaimer: this isn't the state of the art approach

Generating images

Step 2: use decoder to generate data

Img: decoded trajectories from hidden space

Image morphing with AE

Idea:

- If Enc(image1) = c1

  Enc(image2) = c2
- Than maybe (c1+c2)/2 is a semantic average of the two images

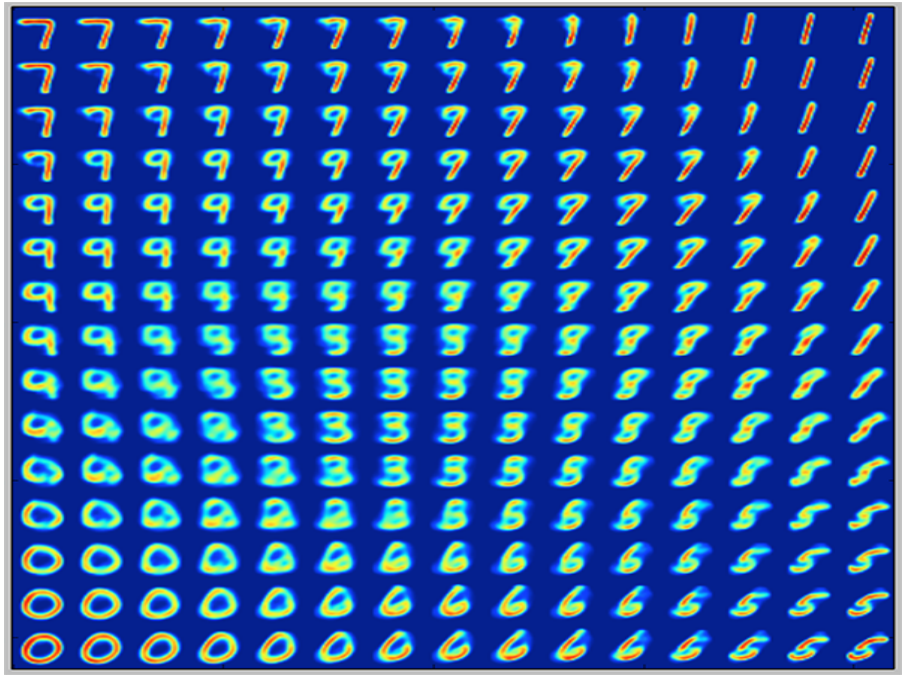
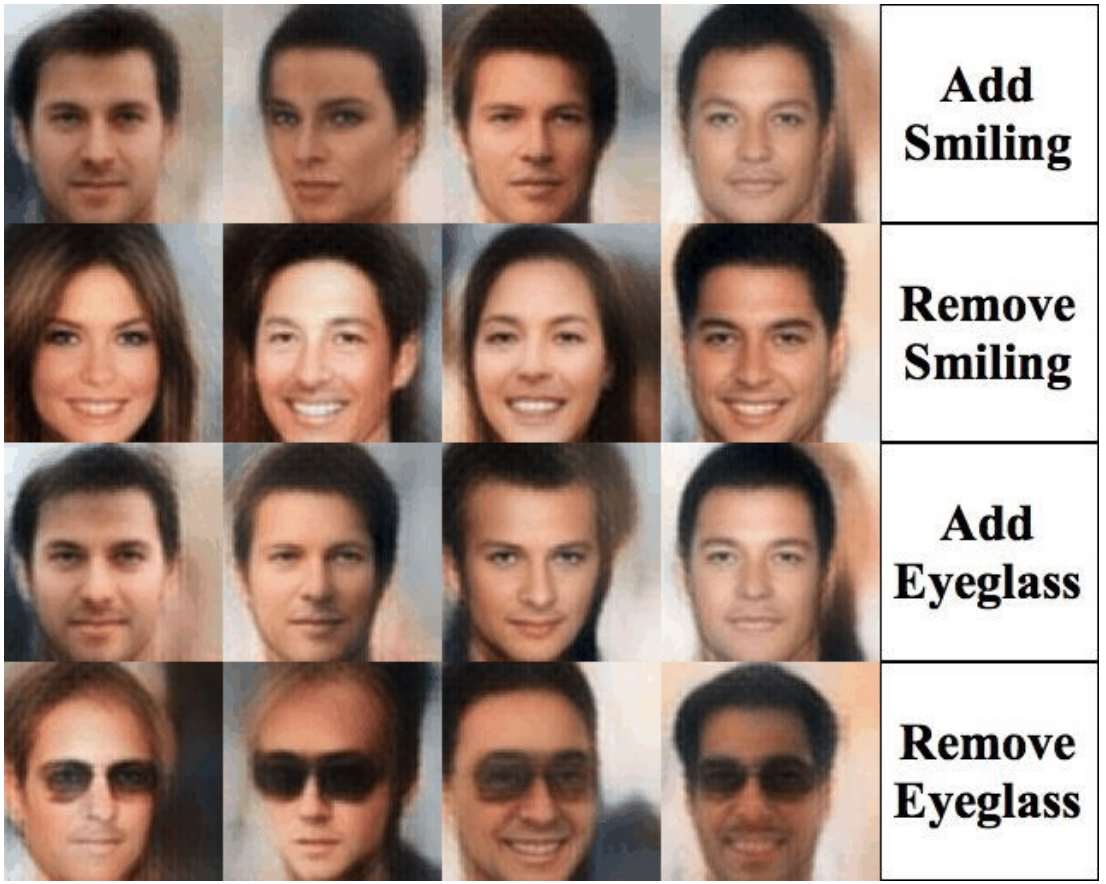


Image morphing with AE

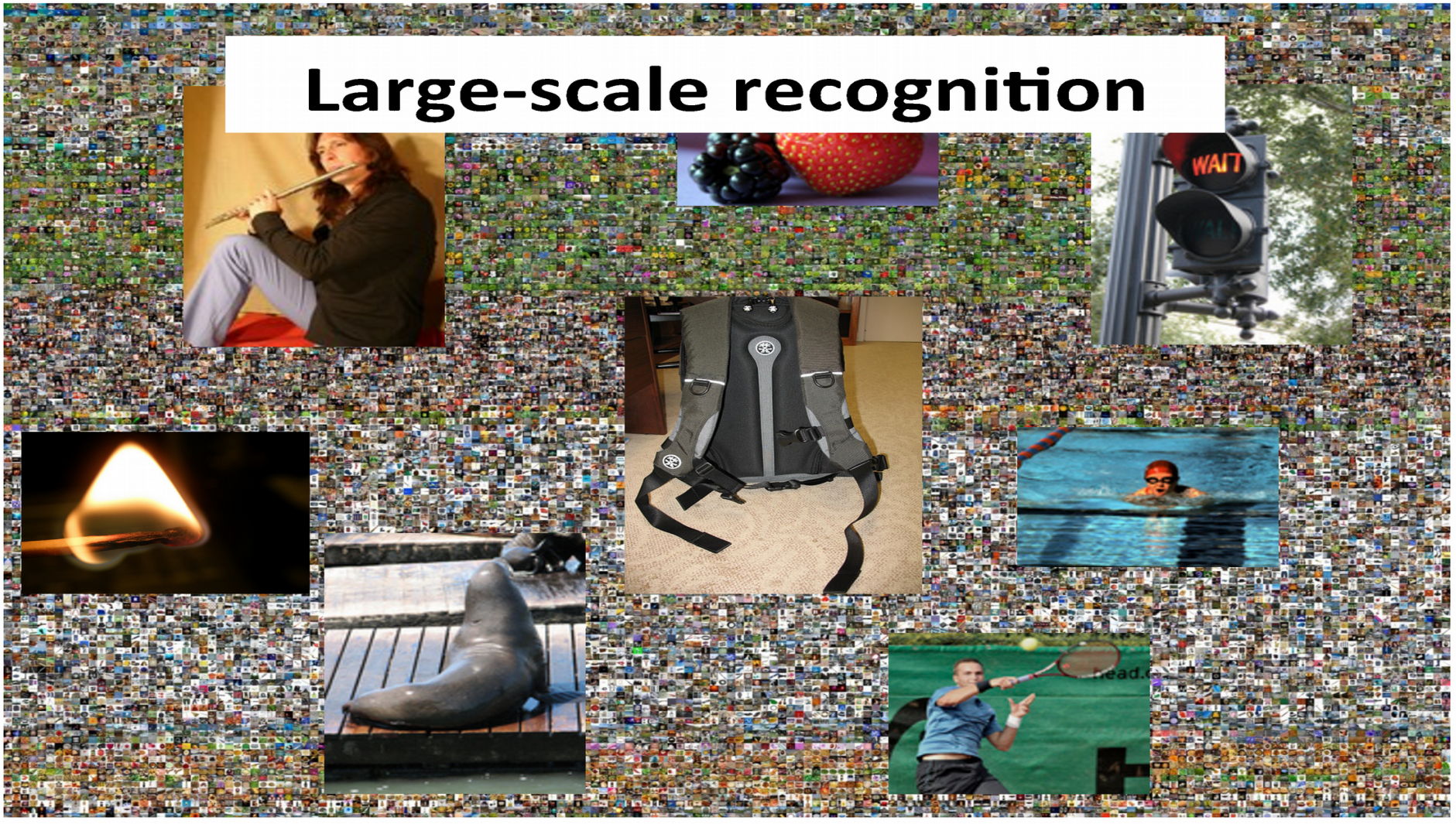
Idea:

- Look for a common direction vector for “add mustache” or “add age” changes.
- Apply to new images

Image morphing with AE



Large-scale recognition



CIFAR

- 60k images
- 10 classes (left)
- 32x32 RGB
- subclasses

aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips

The only problem: you only have 5k images

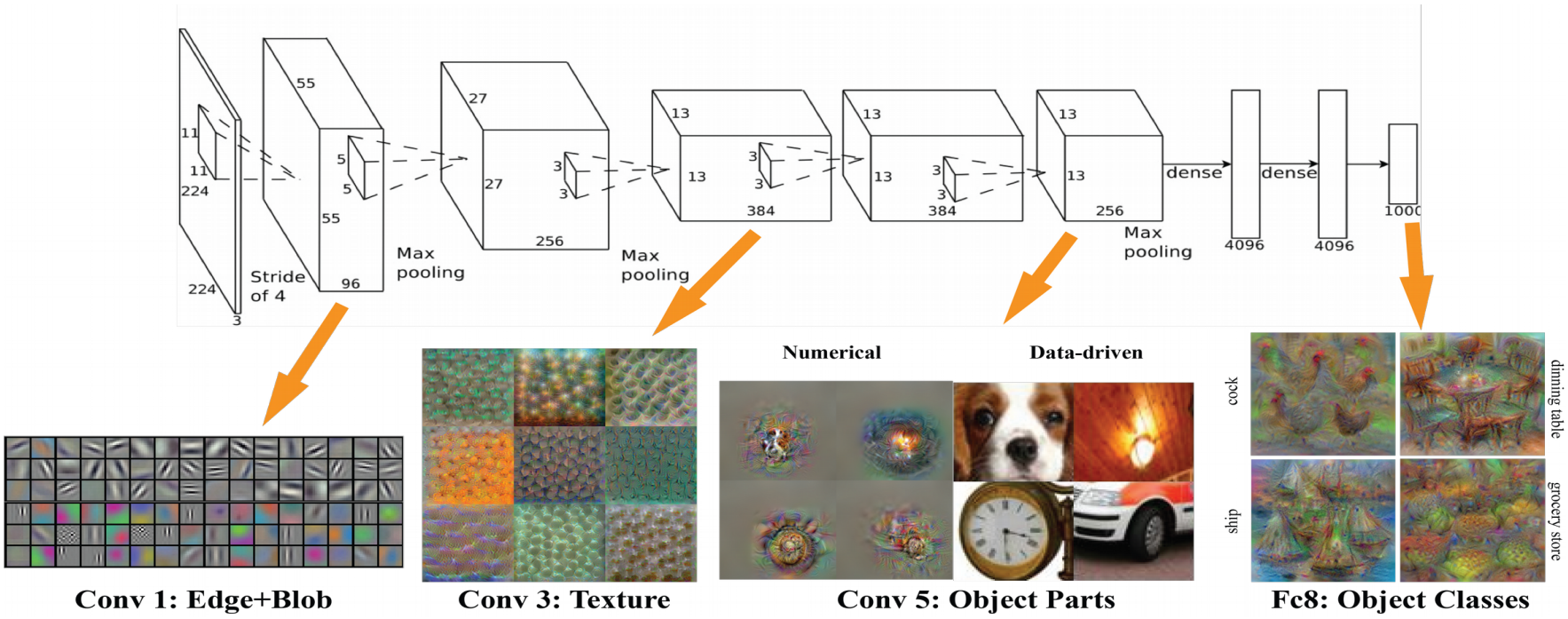
e.g. transport vs military aircrafts

Ideas?

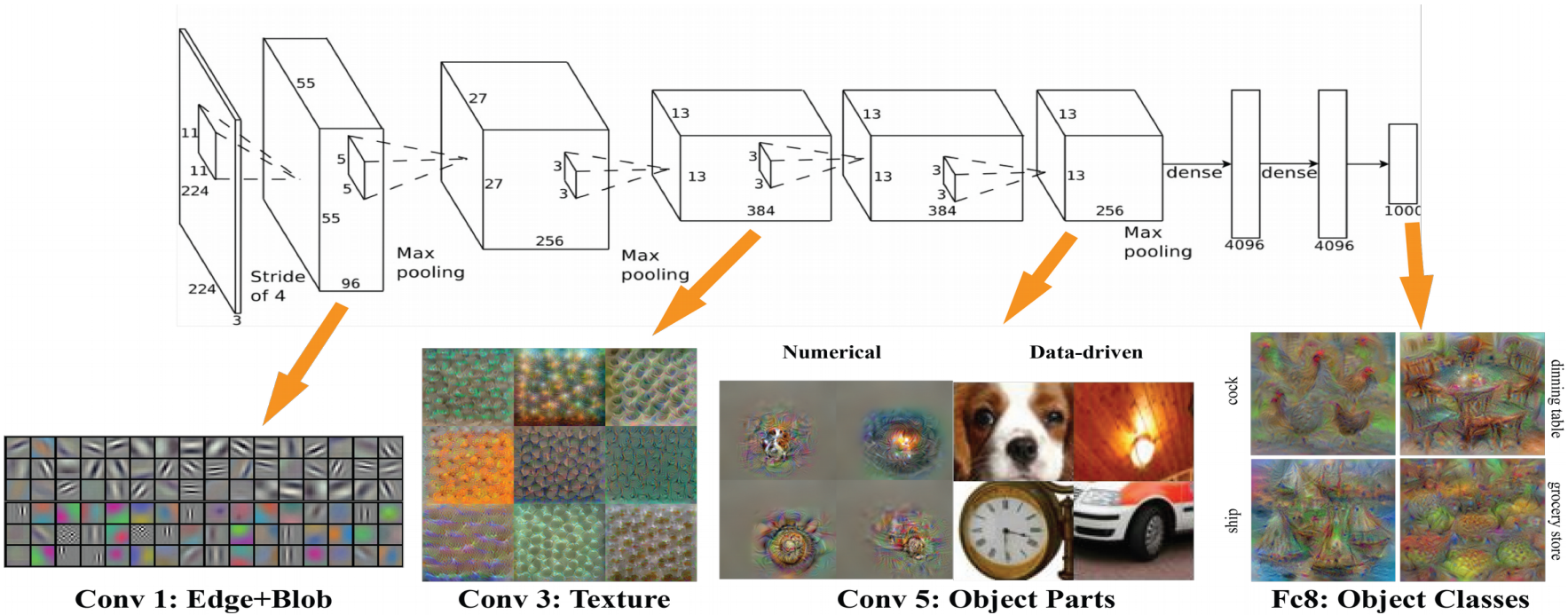
Regular solutions

- Regularize really hard
- Super small network
- Data augmentation
- Whatever

Feature learning



Feature learning

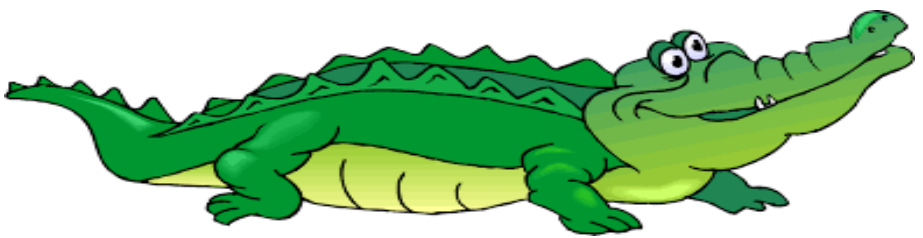


Idea: let's pre-train network on a larger dataset

Pre-training

- 1. Train a network on large dataset

cifar X



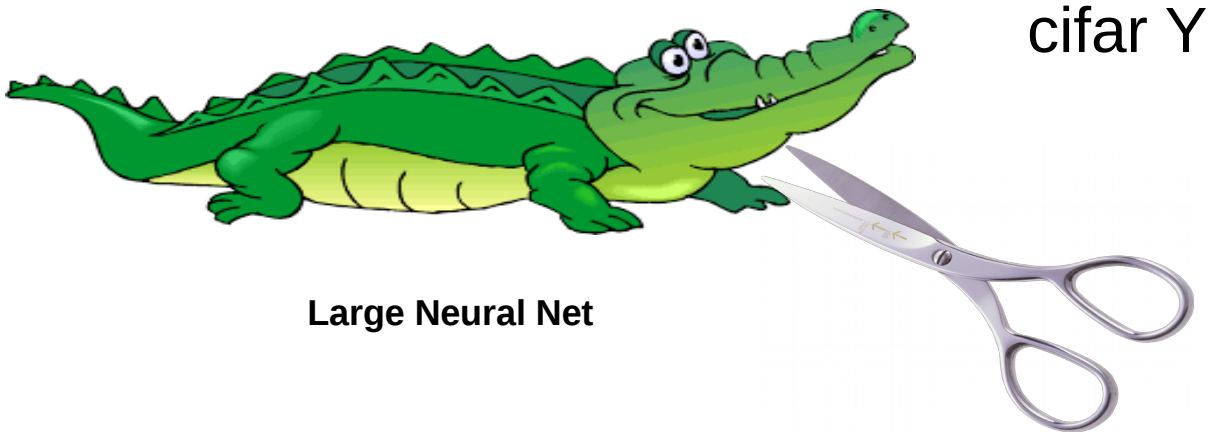
Large Neural Net

cifar Y

Pre-training

- 1. Train a network on large dataset
- 2. Take some intermediate layer

cifar X



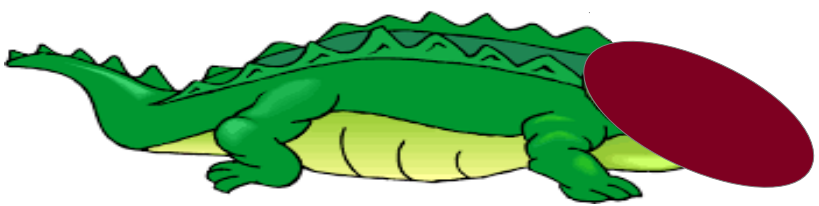
cifar Y

Pre-training

- 1. Train a network on large dataset
- 2. Take some intermediate layer

cifar Y

cifar X

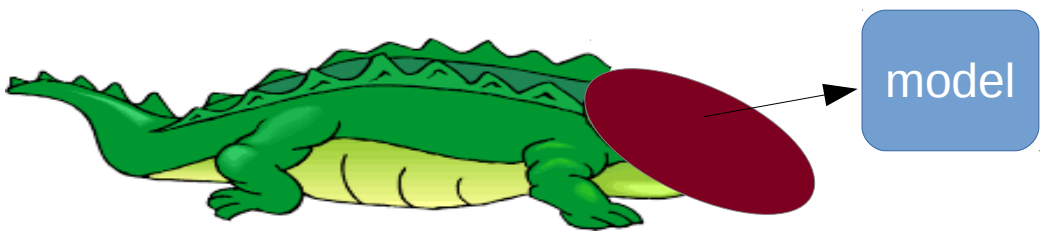


Large Neural Net

Pre-training

- 1. Train a network on large dataset
- 2. Take some intermediate layer
- 3. Build model on top of it

cifar X

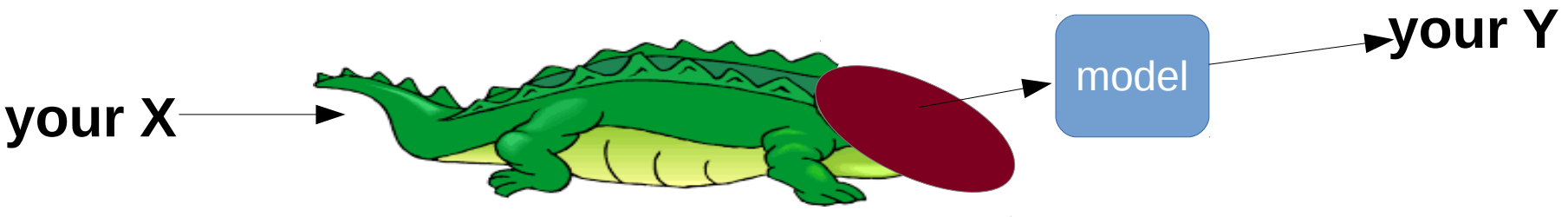


cifar Y

Large Neural Net

Pre-training

- 1. Train a network on large dataset
- 2. Take some intermediate layer
- 3. Build model on top of it
- 4. Train model for your objective

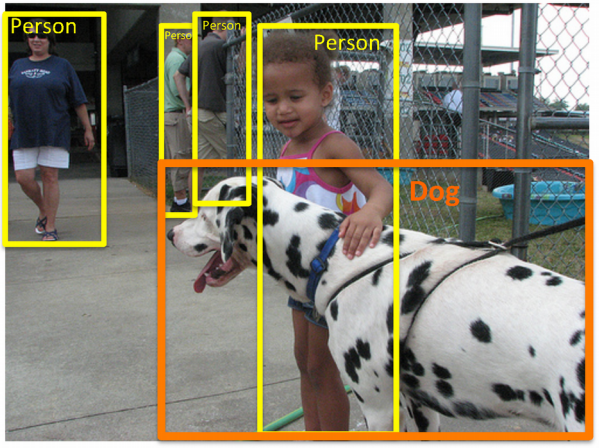


Large Neural Net

Pfft... weakling

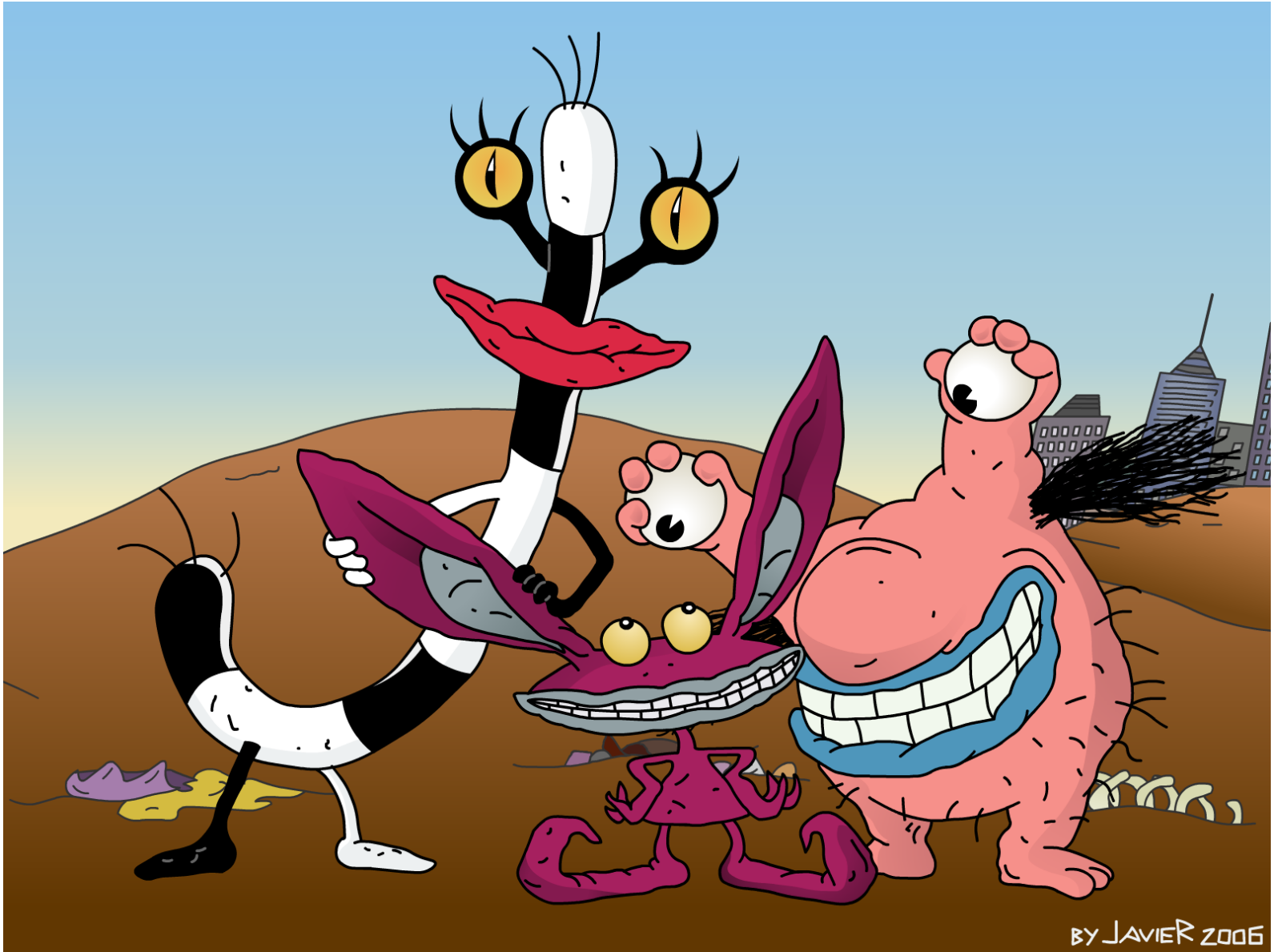
IMAGENET Large Scale Visual Recognition Challenge (ILSVRC) 2010-2014

200 object classes	**456,567 images**	**DET**
1000 object classes	**1,431,167 images**	**CLS-LOC**

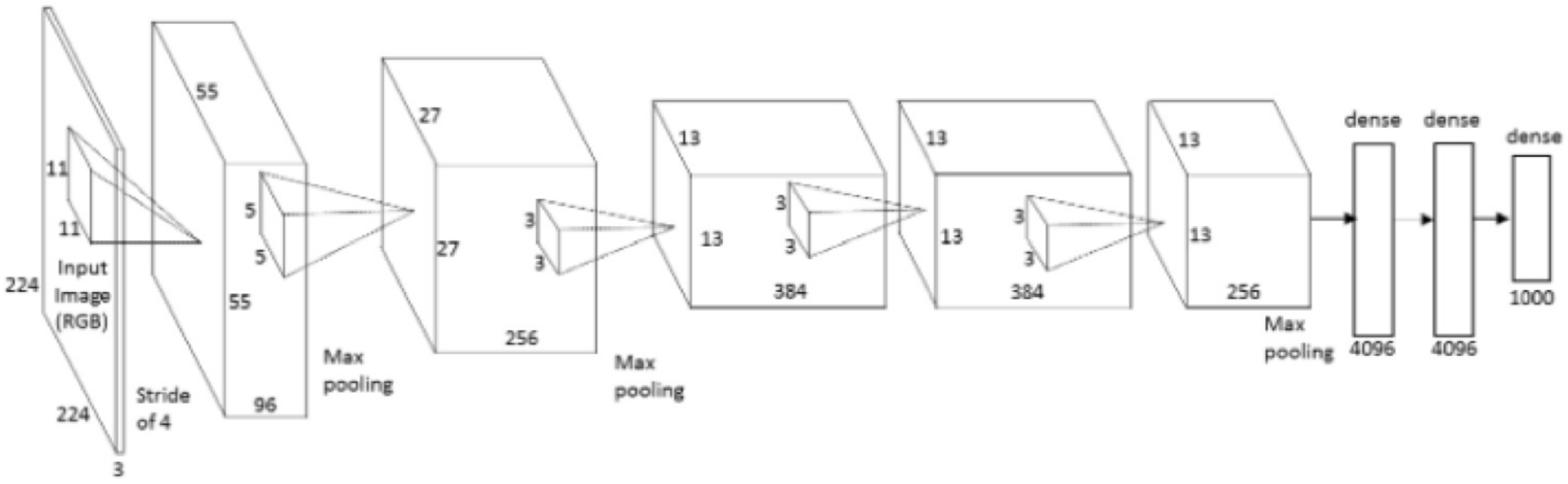


http://image-net.org/challenges/LSVRC/

What kind of network will handle that?

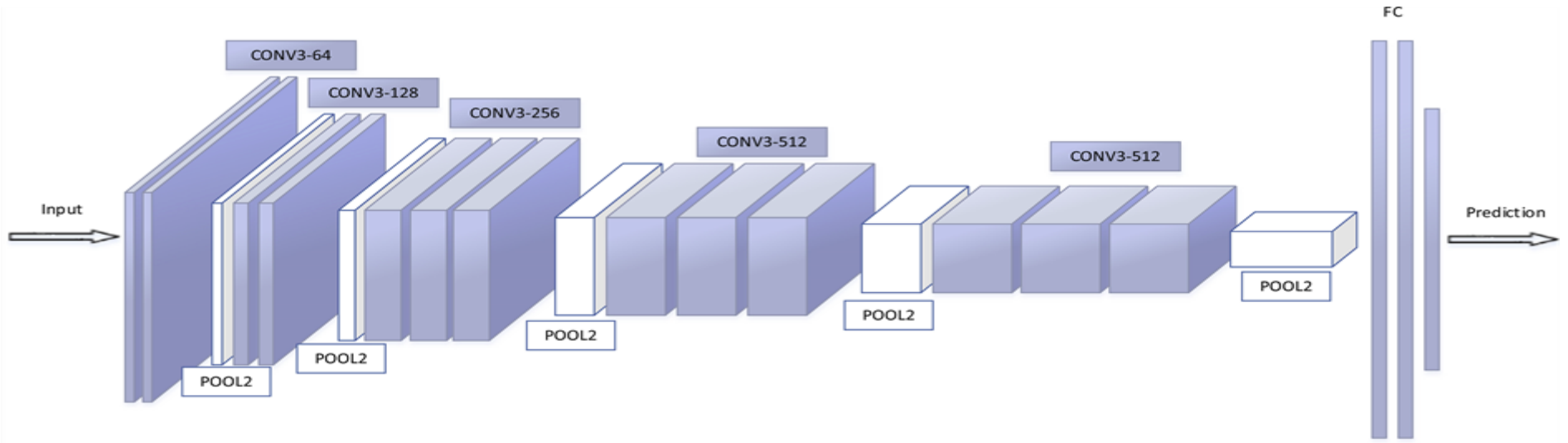

BY JAVIER 2006

Alexnet



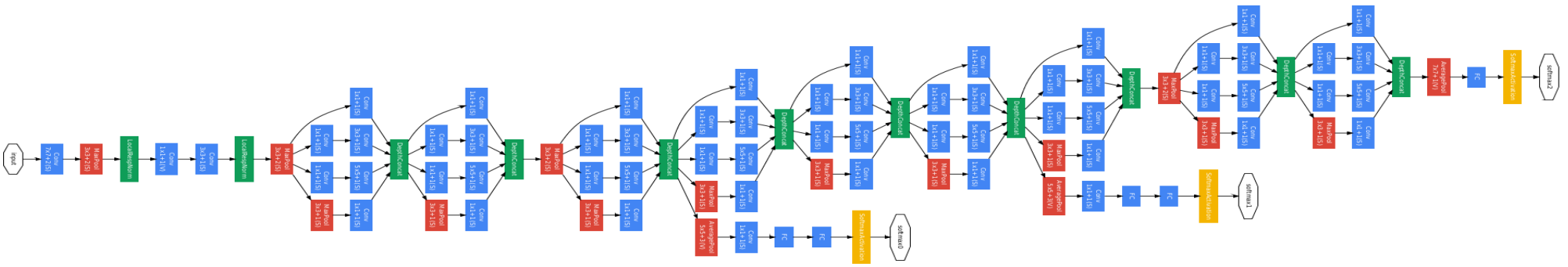
(Krizhevsky et al.)

VGG



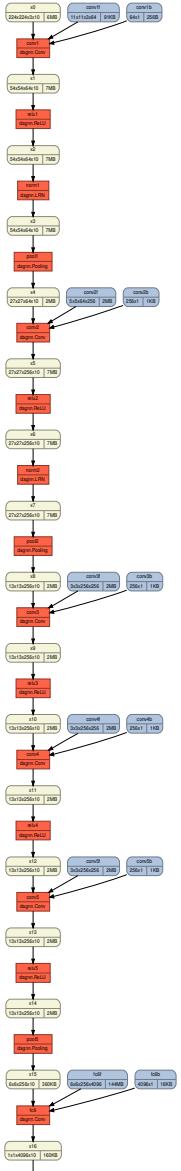
Karen Simonyan and Andrew Zisserman

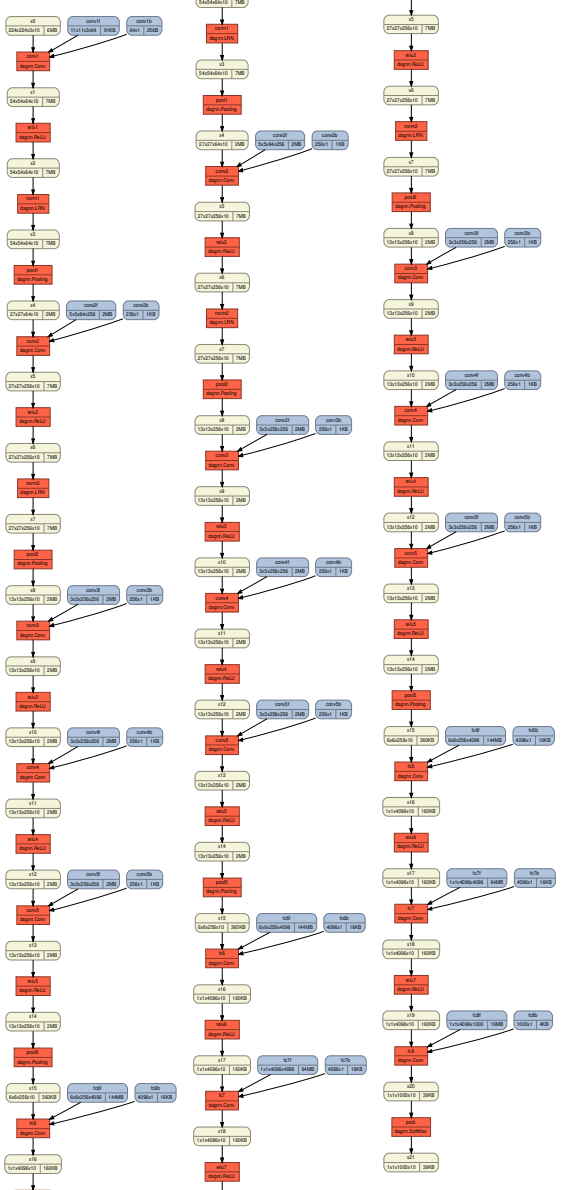
Googlenet (inception)



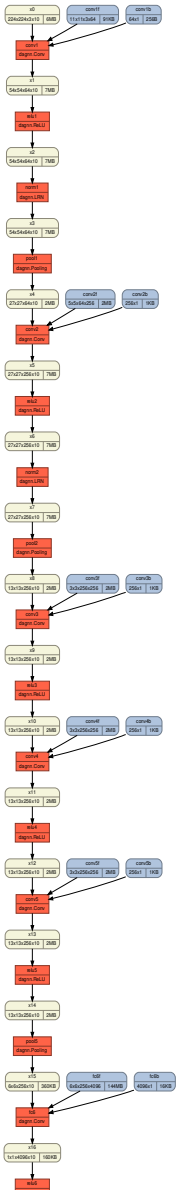
Szegedy et al

ResNet-152

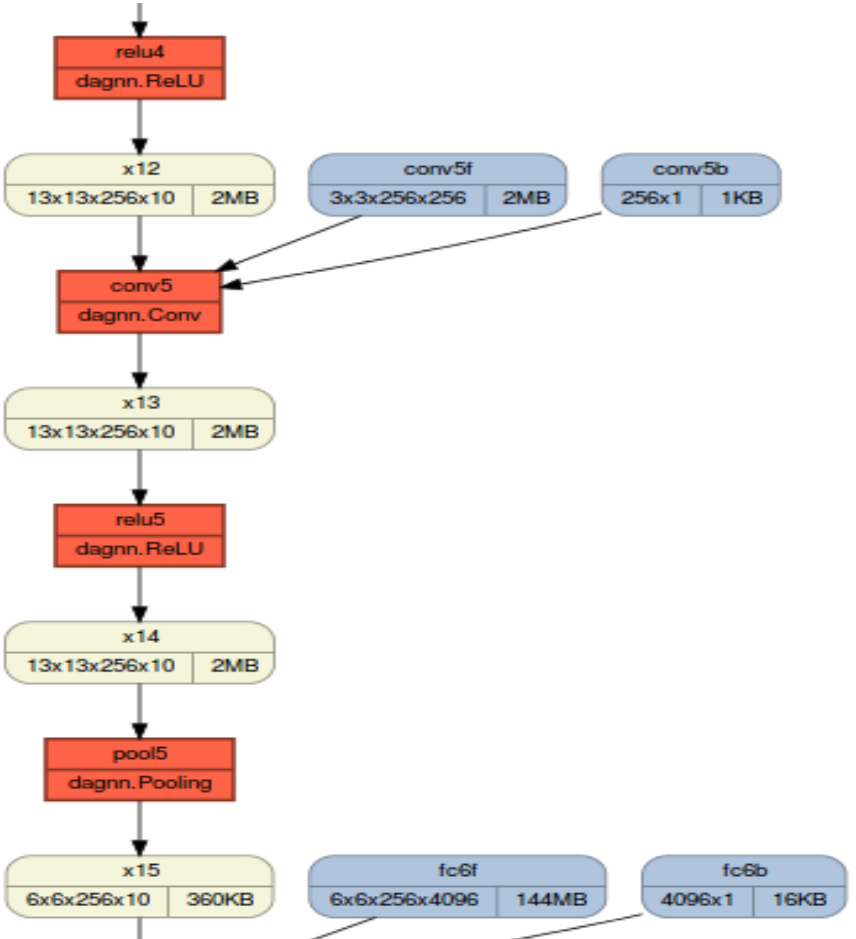




ResNet-152



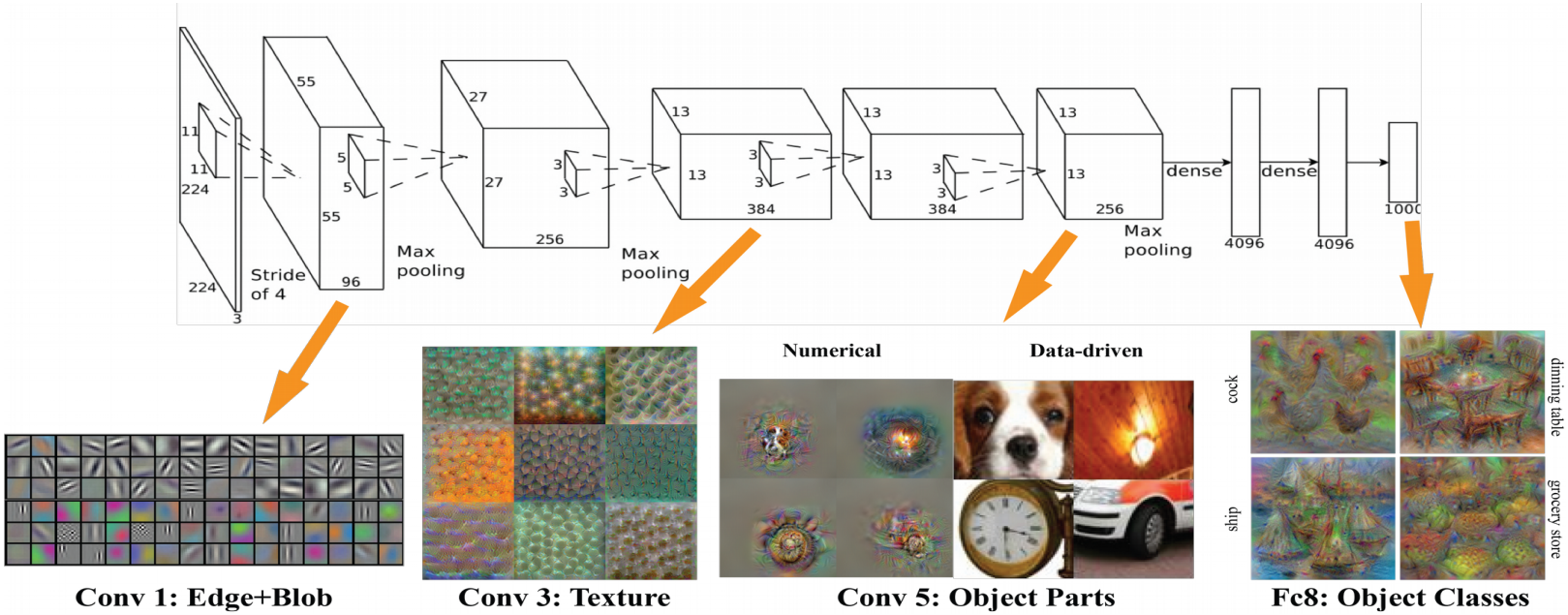
ResNet-152


relu4
dagnn.ReLU
x12
13x13x256x10
2MB
conv5f
3x3x256x256
2MB
conv5b
256x1
1KB
conv5
dagnn.Conv
x13
13x13x256x10
2MB
relu5
dagnn.ReLU
x14
13x13x256x10
2MB
pool5
dagnn.Pooling
x15
6x6x256x10
360KB
fc6f
6x6x256x4096
144MB
fc6b
4096x1
16KB

Can we use them?*

*for aircraft problem

Can we use them?



~ anywhere
even bran scans

photos
any kinds

casual photos
i.e. not satelite maps

highly similar tasks

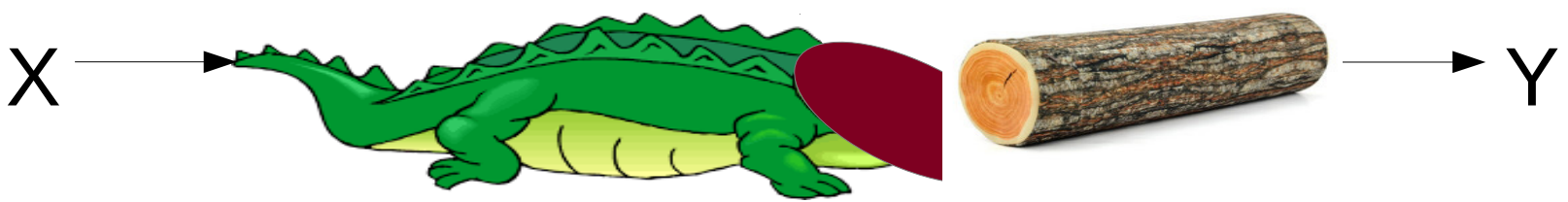
Fine-tuning

- Chop off “head”

Large Neural Net

Reusing features

- Chop off “head”
- use “neck” as feature extractor
- Train ANY classifier
  - even random forest will do



Large Neural Net

Fine-tuning

- Chop off “head”
- “freeze” body (consider constant)
- Build new neural network in it's place
- Train “head” only for several iteratons
- Un-freeze body and train full network

Large Neural Net

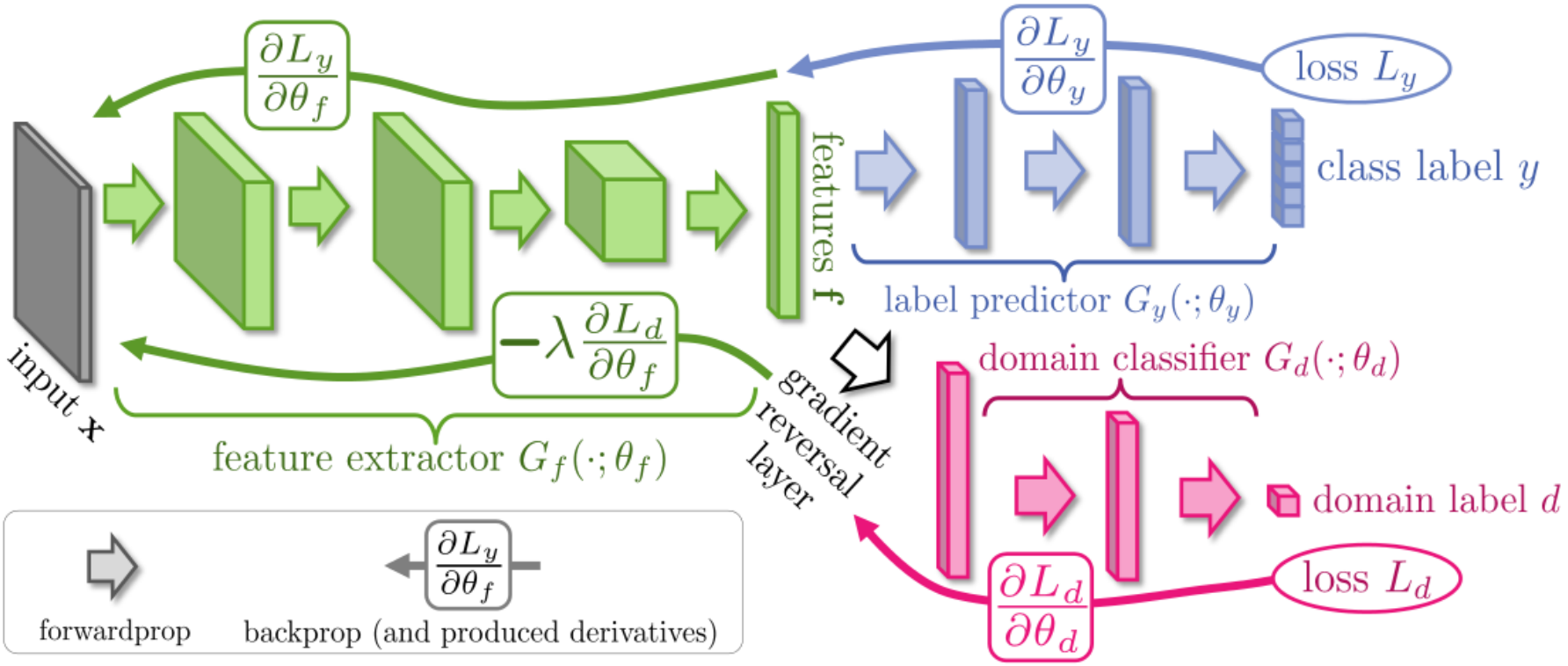
Domain adaptation (WIP)

Image source

Image: MNIST

Image source

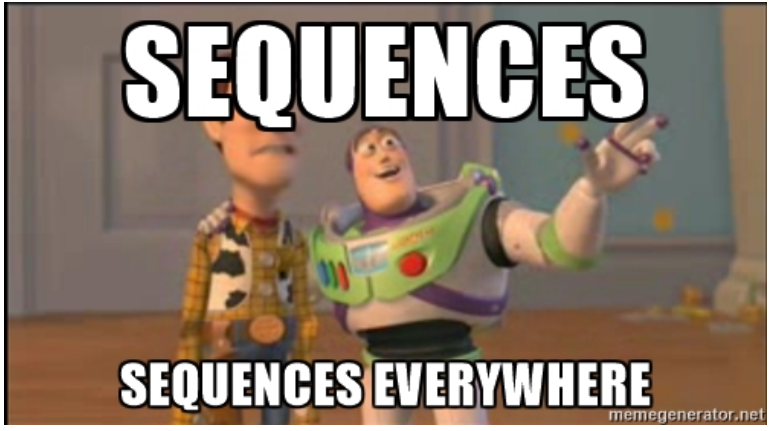
Gradient reversal



Recurrent Neural Networks

Sequential data

- Time series
  - Financial data analysis
  - Demand prediction
  - Predict vehicle breakdown using sensor data
  - Medical sensors
- Text
  - Generating tweets, poetry
  - Sentiment analysis
- Spatio-temporal
  - Video
  - Precipitation maps
- Etc
  - Speech, music
  - Molecules (SMILES)
  - Decay trees
  - …

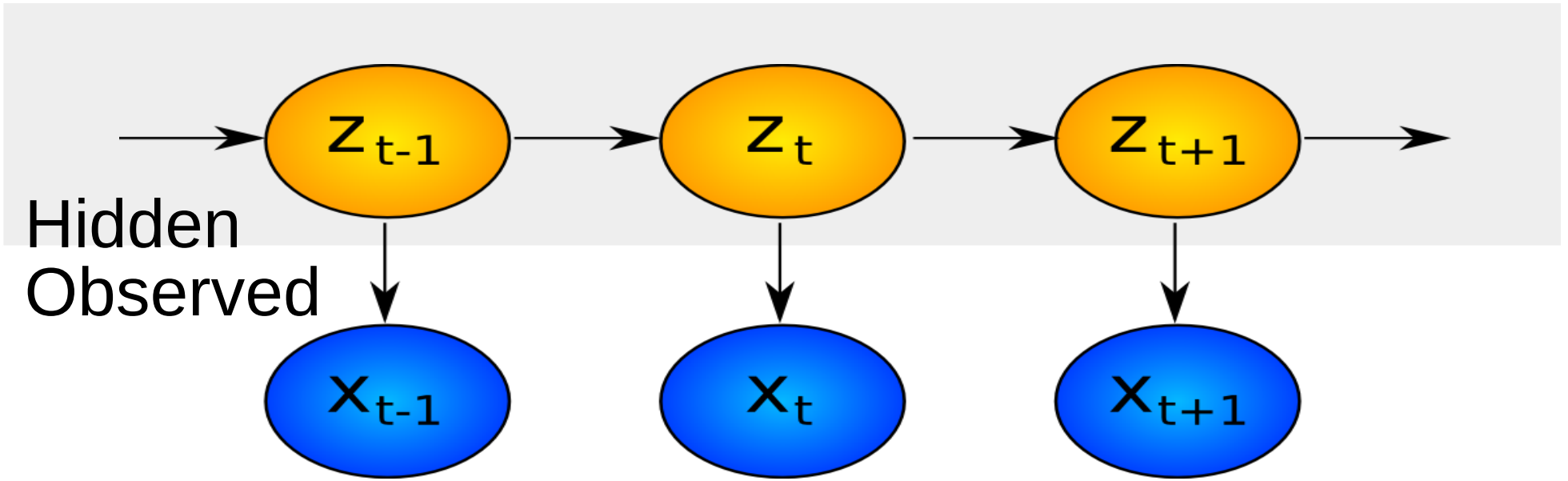


Could go on all day

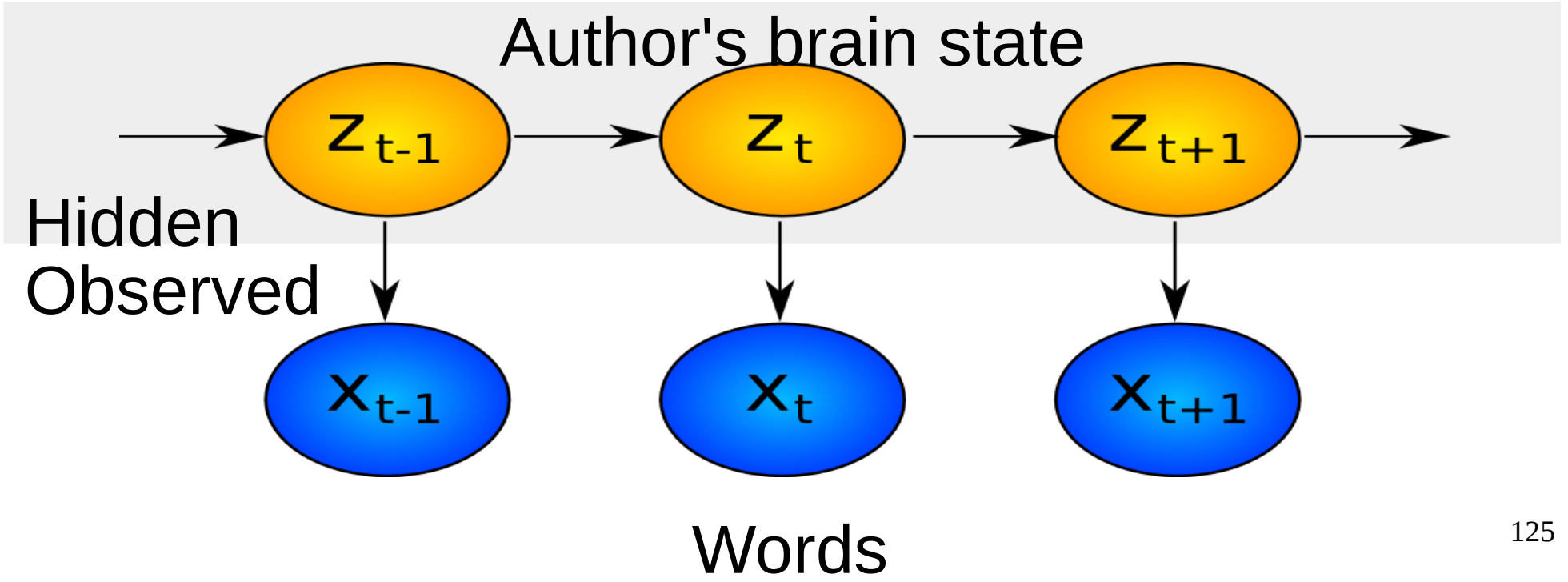
Can we learn* arbitrarily long dependencies?

* without infinitely many parameters

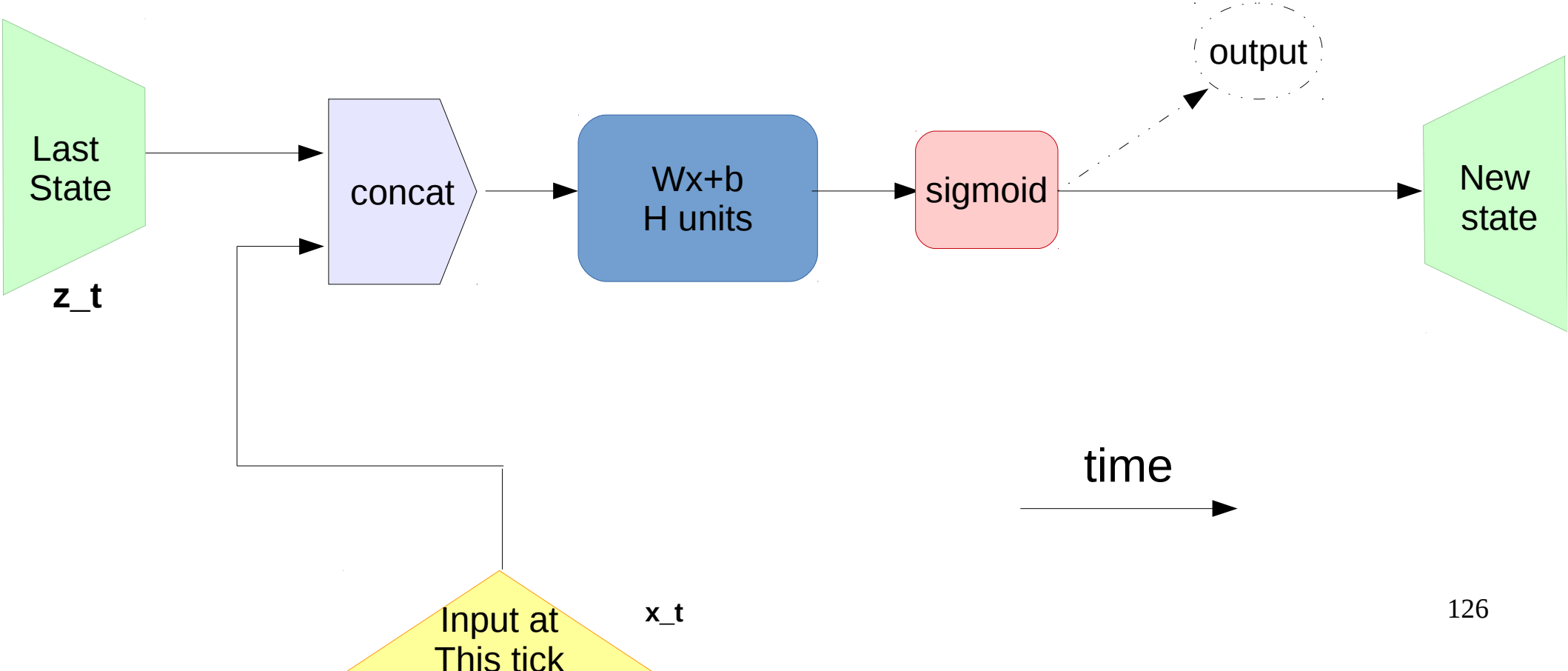
Hidden Markov Models: what's hidden



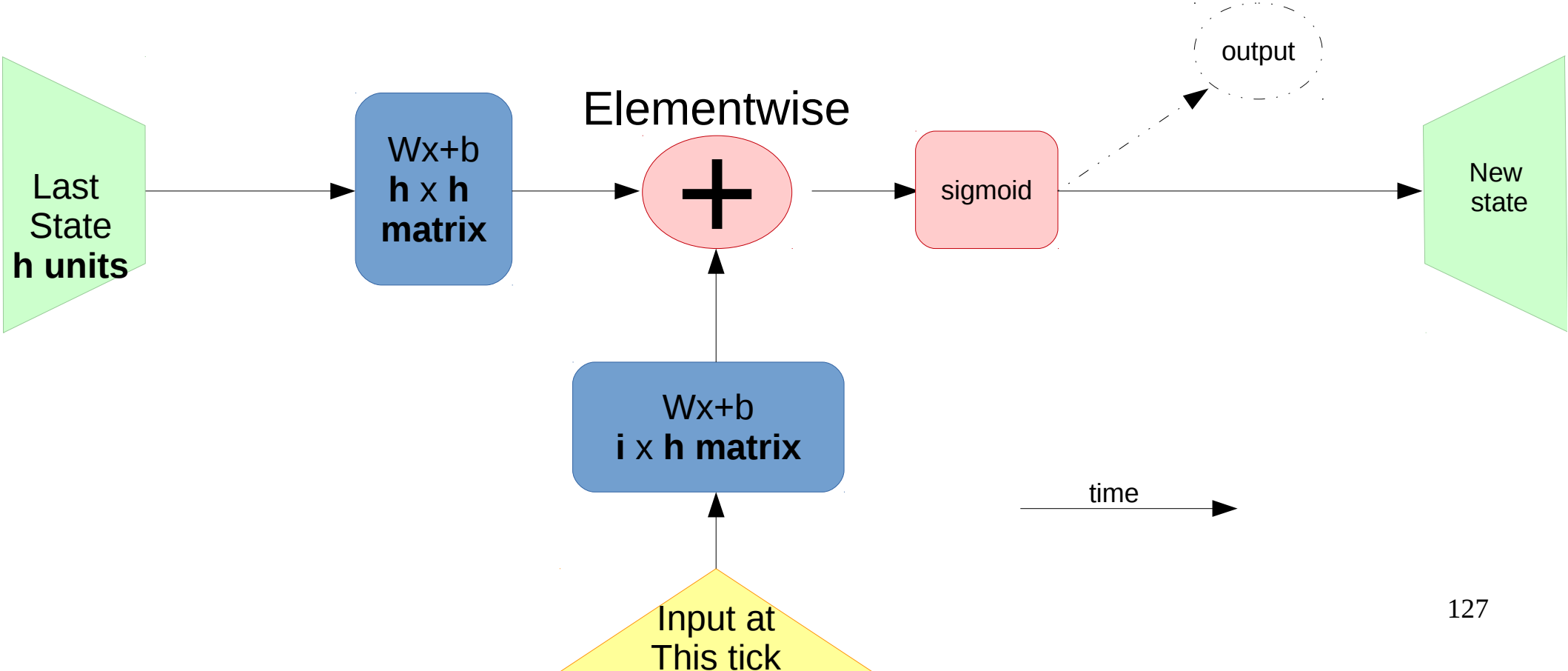
Hidden Markov Models: what's hidden



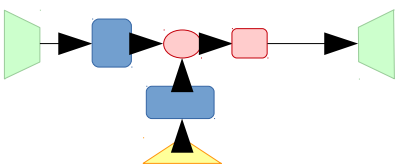
Recurrent neural network: one step



Recurrent neural network: one step

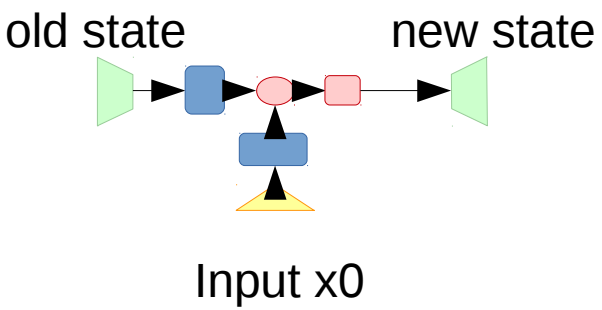


Recurrent neural network

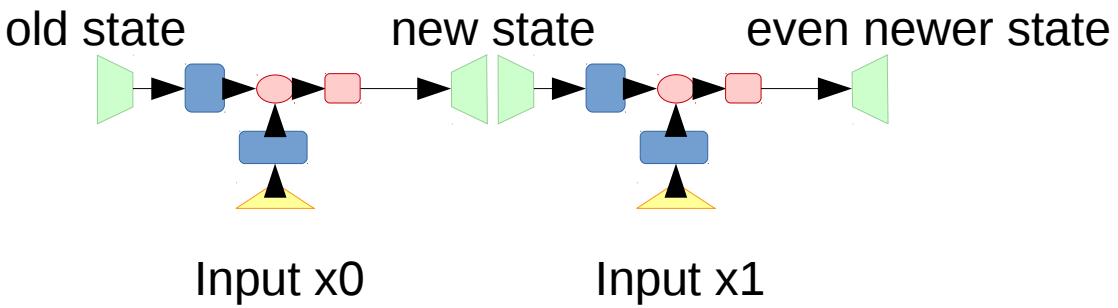


Zoom-out
of previous slide

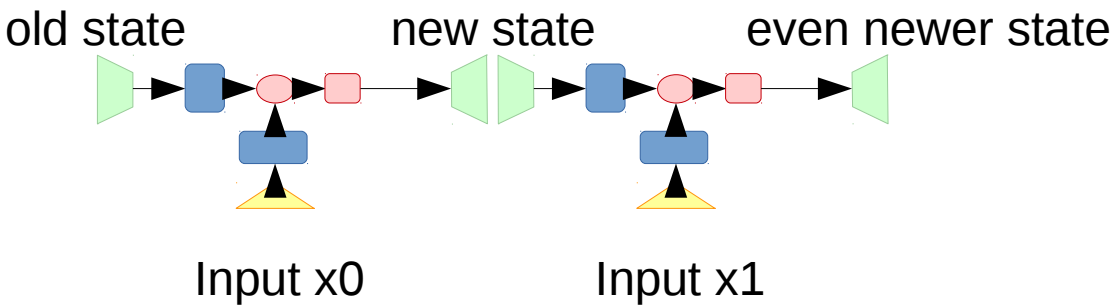
Recurrent neural network



Recurrent neural network

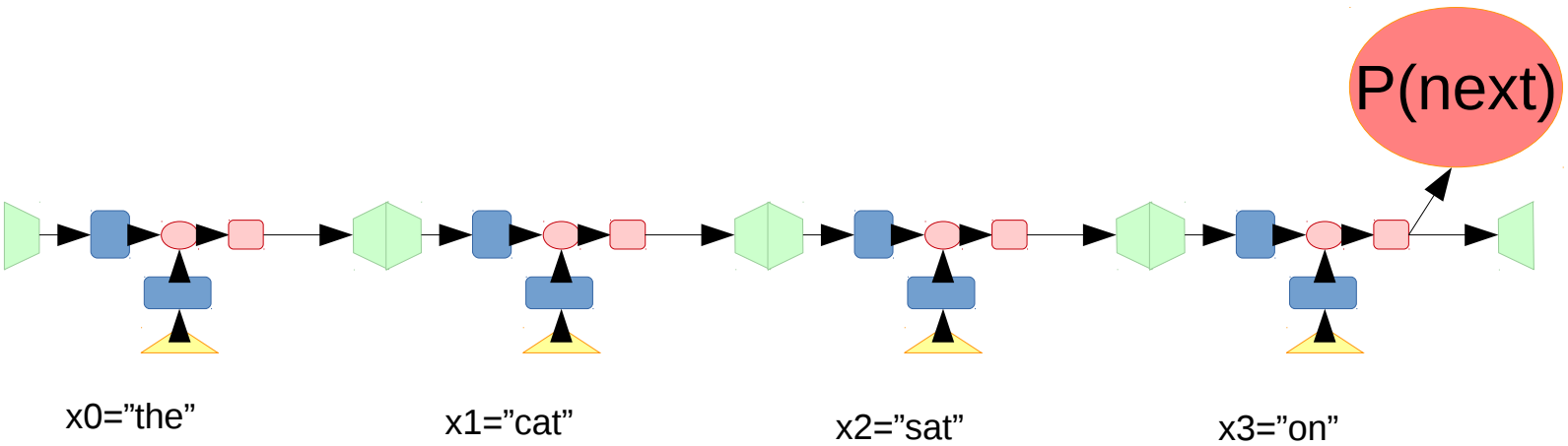


Recurrent neural network

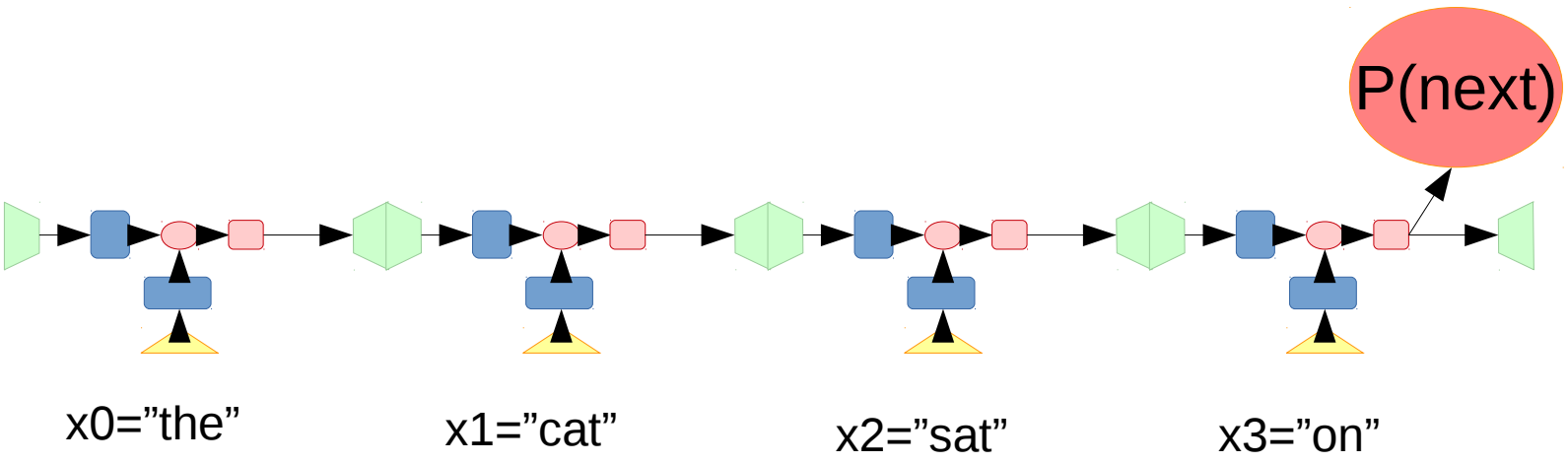


We use **same weight matrices** for all steps

Recurrent neural network

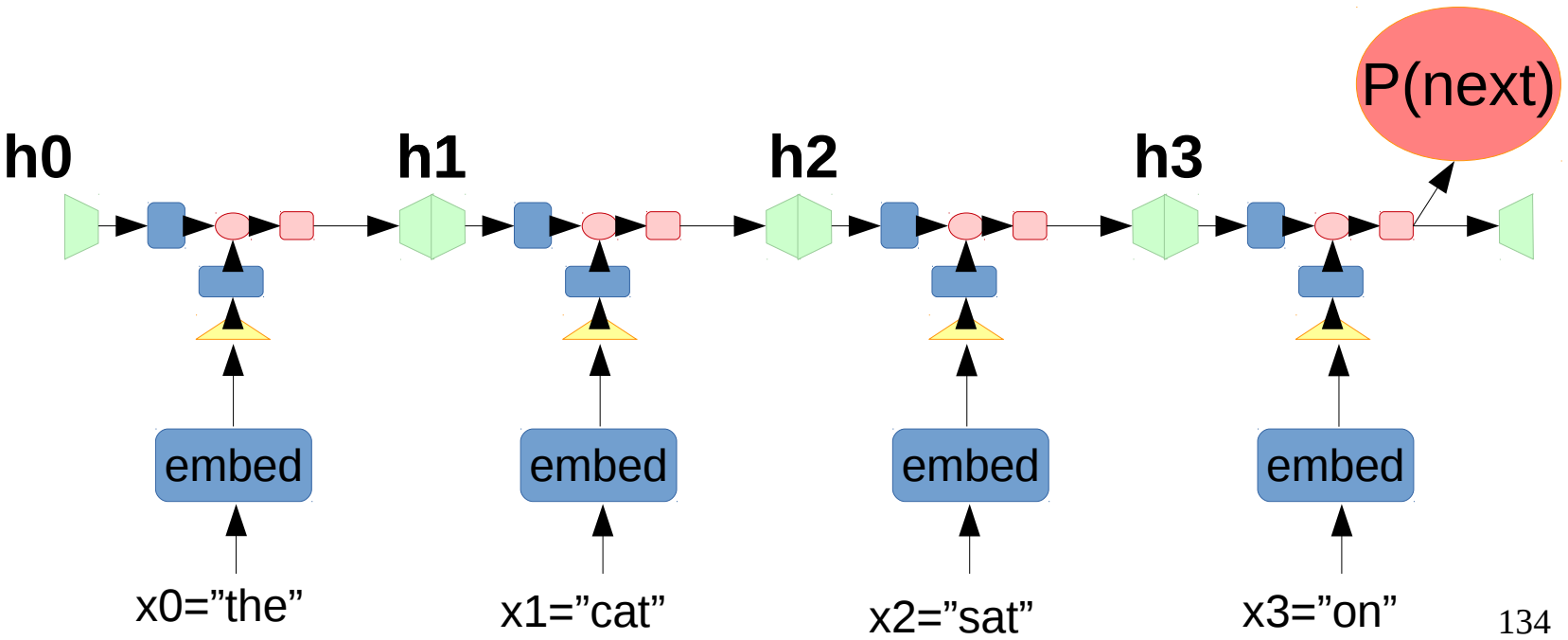


Recurrent neural network



How can we represent words?

Recurrent neural network

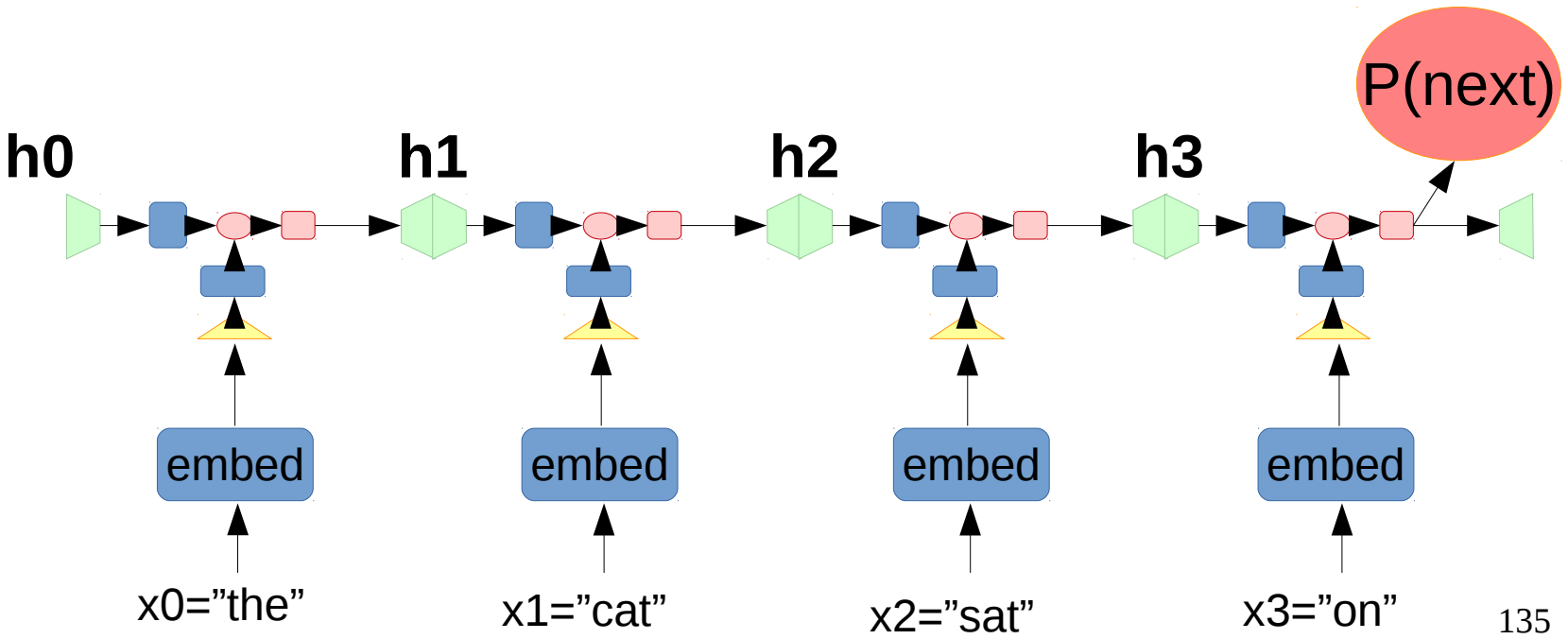


Recurrent neural network

$$h_0 = \overline{0}$$

$$h_1 = \sigma\left(W_{hid} \cdot h_0 + W_{inp} \cdot x_0 + b\right)$$

$$h_2 = ?$$



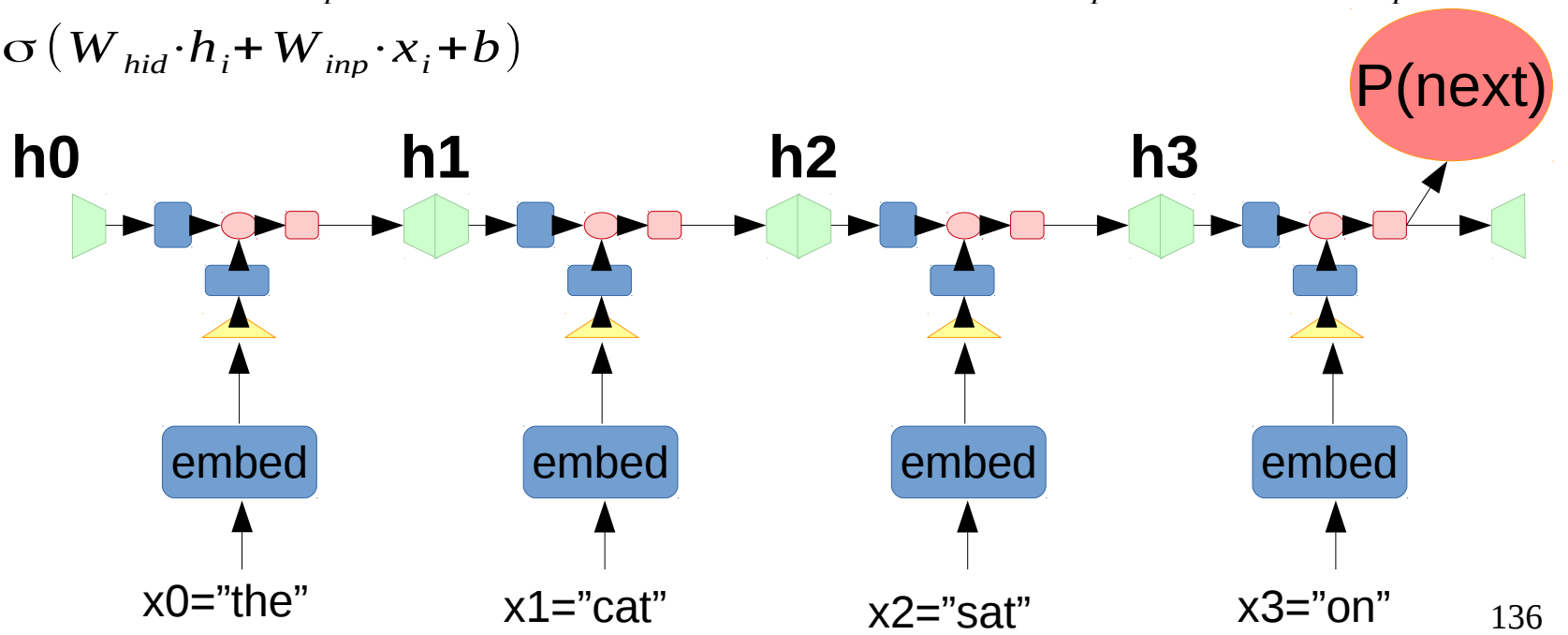
Recurrent neural network

$$h_0 = \bar{0}$$

$$h_1 = \sigma(W_{hid} \cdot h_0 + W_{inp} \cdot x_0 + b)$$

$$h_2 = \sigma(W_{hid} \cdot h_1 + W_{inp} \cdot x_1 + b) = \sigma(W_{hid} \cdot \sigma(W_{hid} \cdot h_0 + W_{inp} \cdot x_0 + b) + W_{inp} \cdot x_1 + b)$$

$$h_{i+1} = \sigma(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b)$$



Recurrent neural network

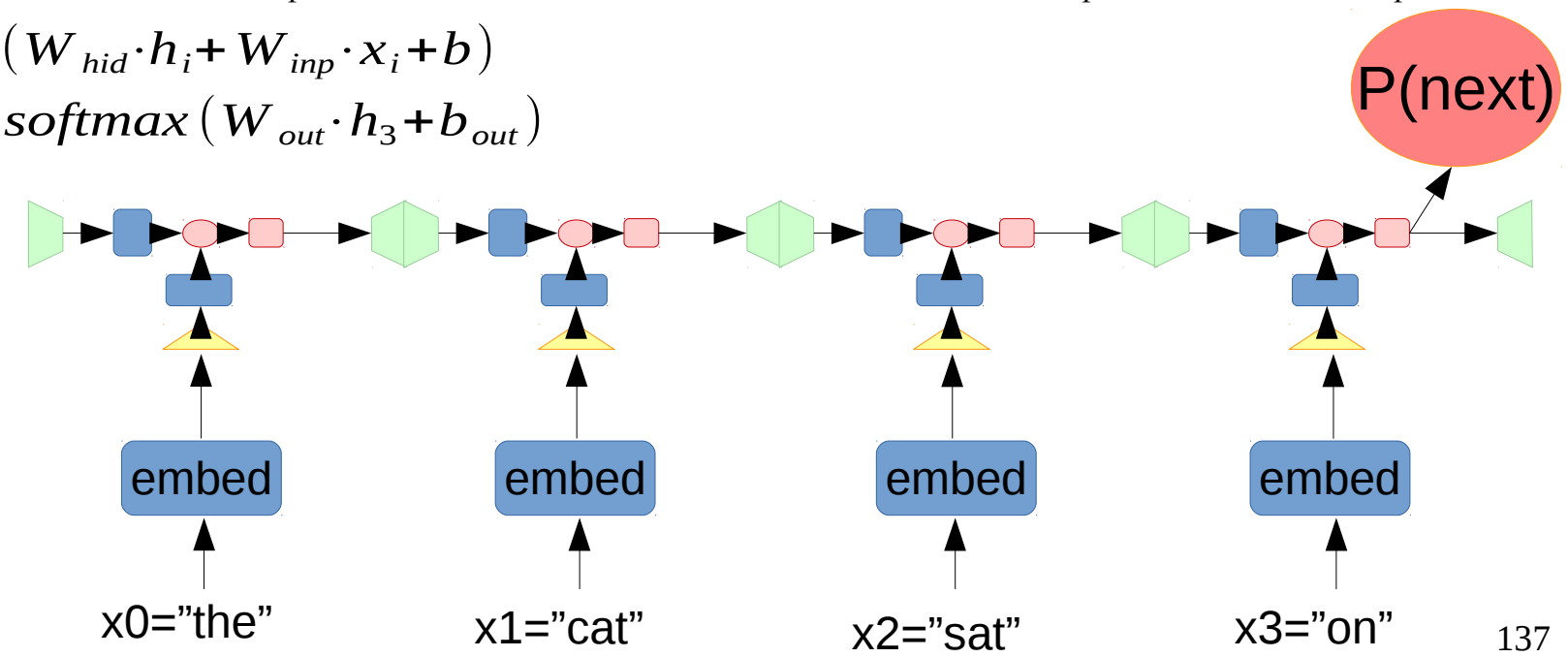
$$h_0 = \bar{0}$$

$$h_1 = \sigma(W_{hid} \cdot h_0 + W_{inp} \cdot x_0 + b)$$

$$h_2 = \sigma(W_{hid} \cdot h_1 + W_{inp} \cdot x_1 + b) = \sigma(W_{hid} \cdot \sigma(W_{hid} \cdot h_0 + W_{inp} \cdot x_0 + b) + W_{inp} \cdot x_1 + b)$$

$$h_{i+1} = \sigma(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b)$$

$$P(t_4) = softmax(W_{out} \cdot h_3 + b_{out})$$

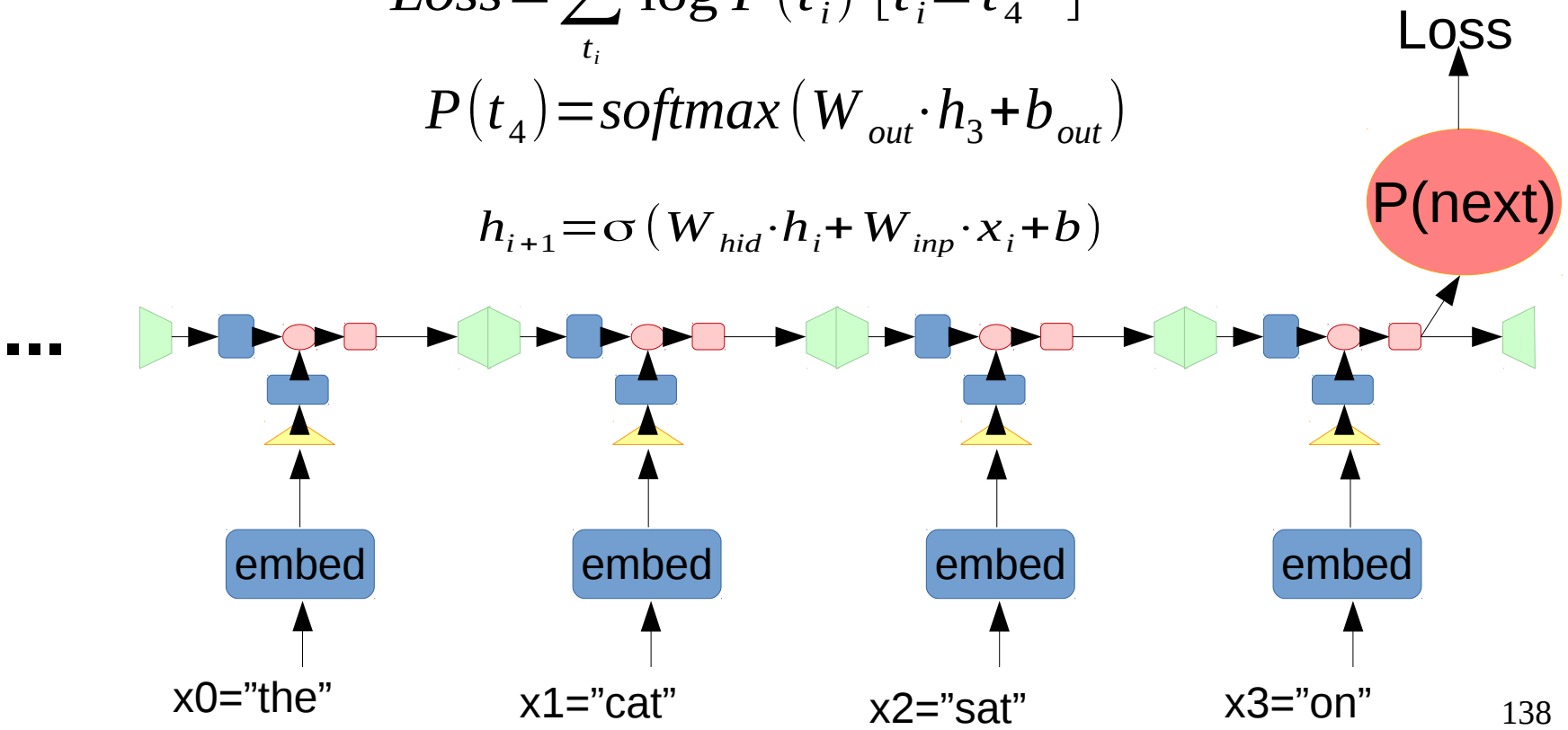


Recurrent neural network

$$Loss = \sum_{t_i} \log P(t_i) \cdot [t_i = t_4^{true}]$$

$$P(t_4) = softmax(W_{out} \cdot h_3 + b_{out})$$

$$h_{i+1} = \sigma(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b)$$



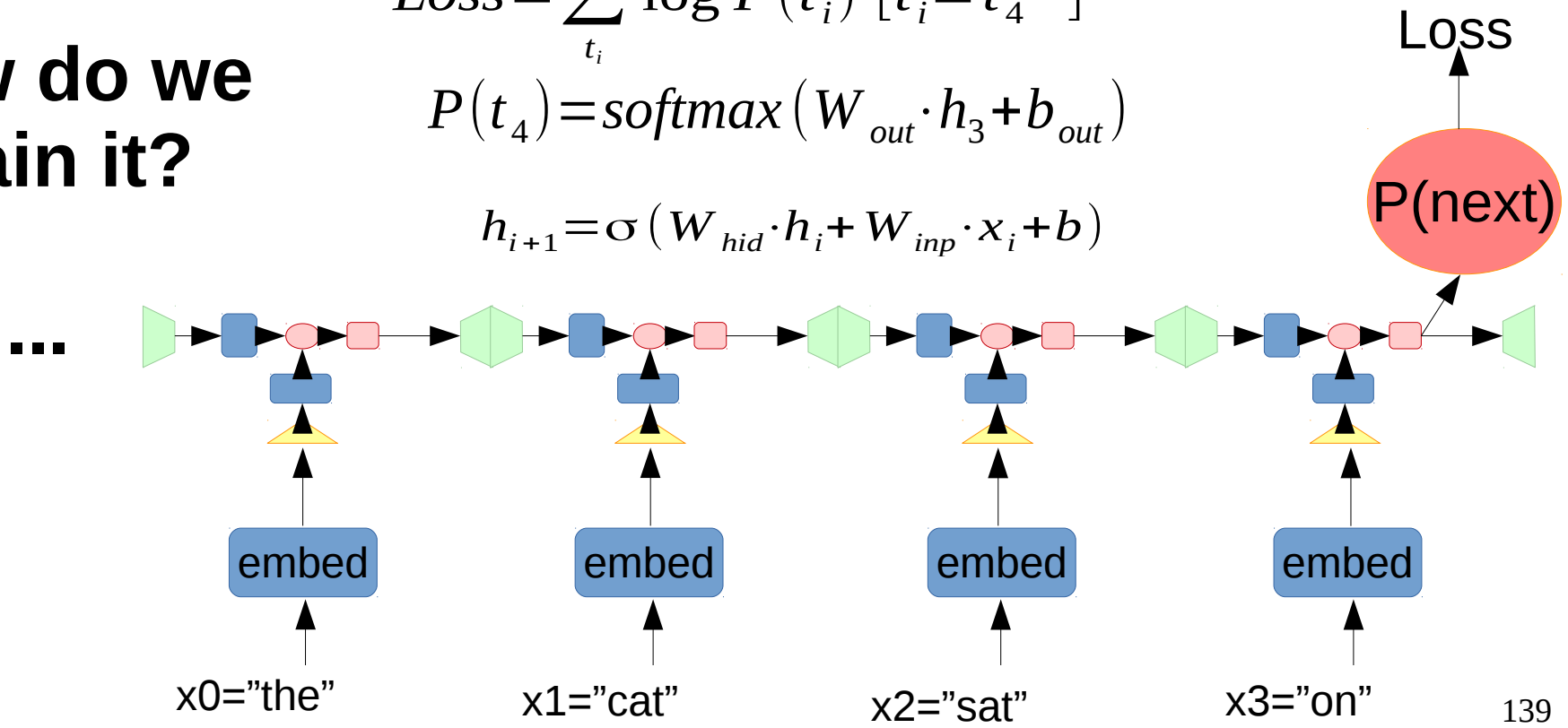
Recurrent neural network

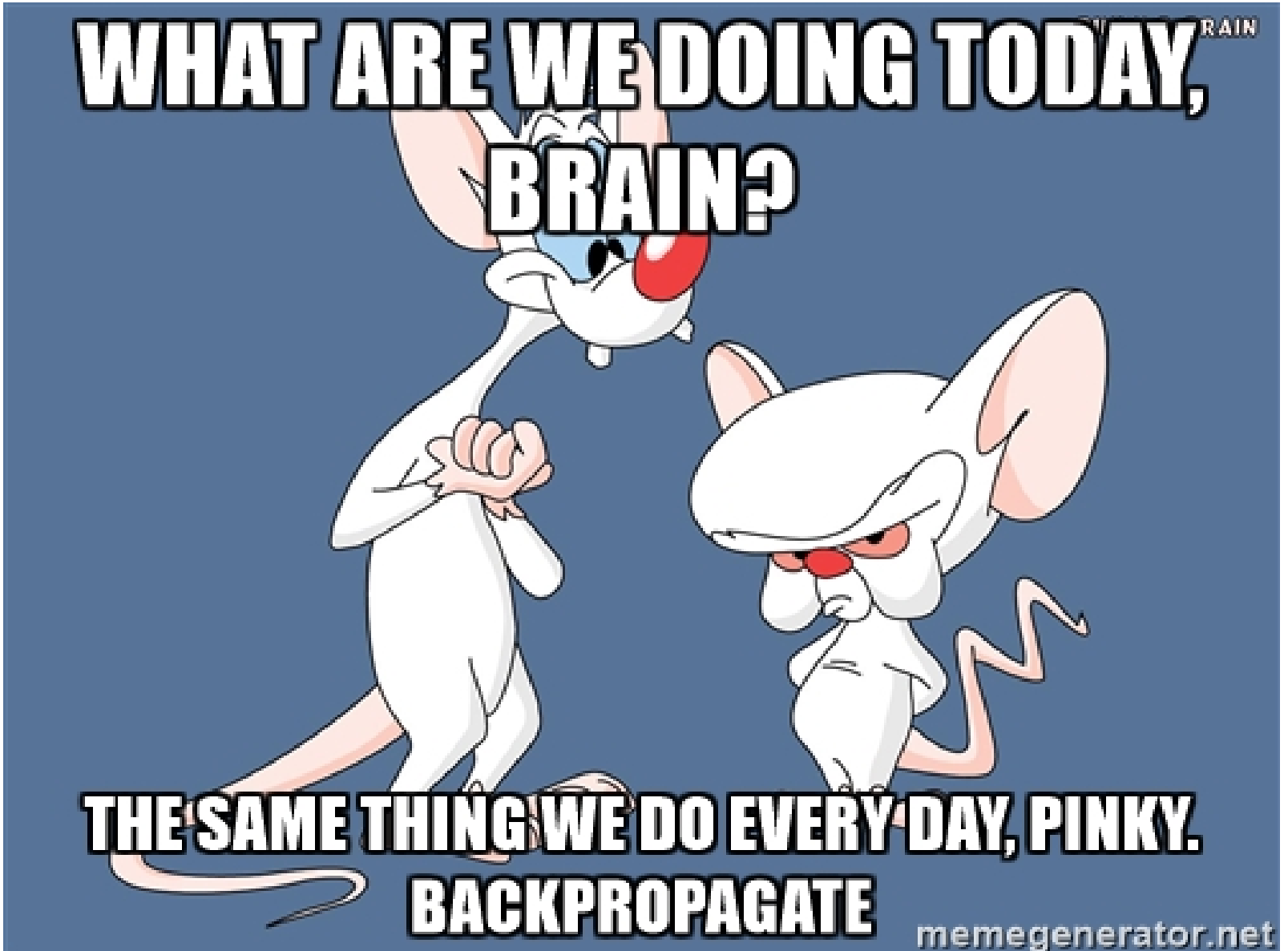
How do we train it?

$$Loss = \sum_{t_i} \log P(t_i) \cdot [t_i = t_4^{true}]$$

$$P(t_4) = softmax(W_{out} \cdot h_3 + b_{out})$$

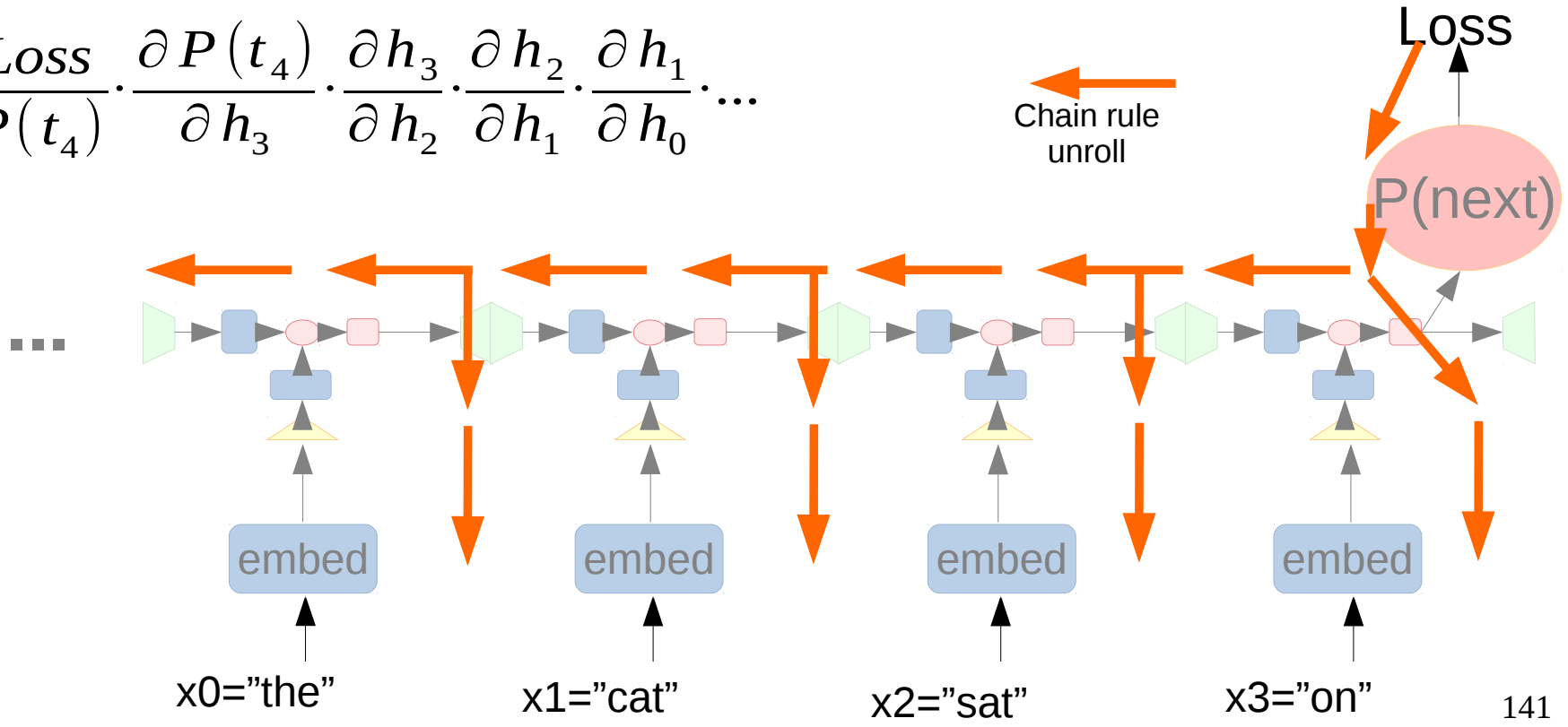
$$h_{i+1} = \sigma(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b)$$




WHAT ARE WE DOING TODAY, BRAIN?
THE SAME THING WE DO EVERY DAY, PINKY.
BACKPROPAGATE
memegenerator.net

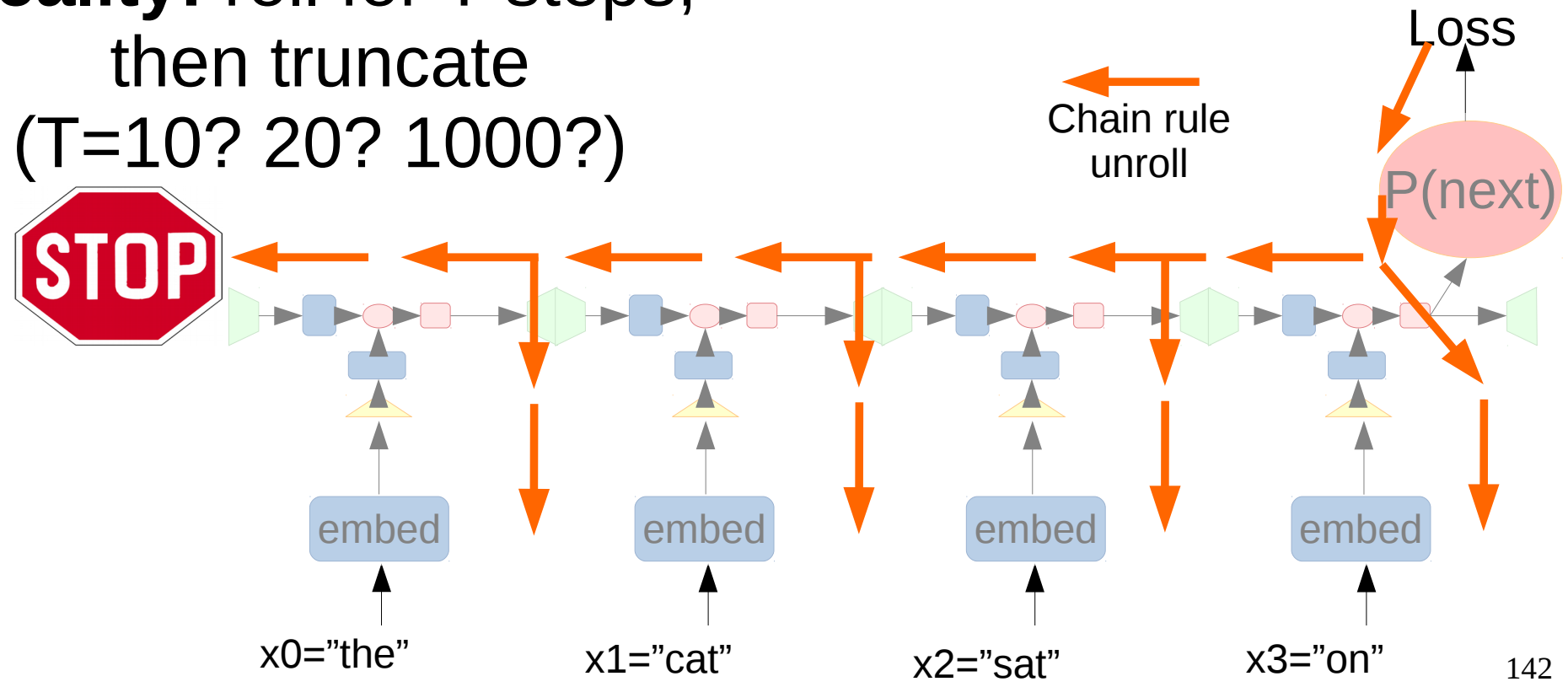
Backpropagation through time

$$\frac{\partial Loss}{\partial P(t_4)} \cdot \frac{\partial P(t_4)}{\partial h_3} \cdot \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial h_0} \cdot \ldots$$



Truncated BPTT

Reality: roll for T steps, then truncate (T=10? 20? 1000?)



Summary

- Deep learning has the hype
  - Some of it is deserved
- You need a GPU
- Use for
  - “Ordinary” classification & regression
  - Spatial data (e. g. images)
  - Sequential data (e. g. speech)
  - Physics?
    - Jet Substructure Classification
    - A Convolutional Neural Network Neutrino Event Classifier