Practical Reinforcement Learning
Episode 3.5

Deep Learning 101

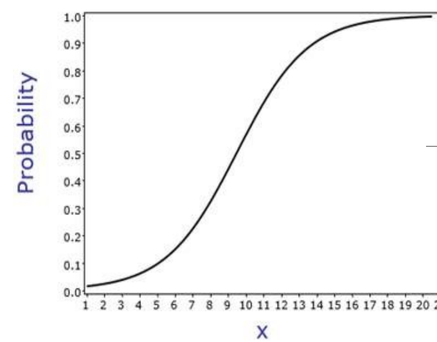
Yandex
Data Factory

LAMBDA

British Hedgehog
Preservation Society

Recap: logistic regression

X → Wx +b →



→ P(y)

Recap: Gradient descent

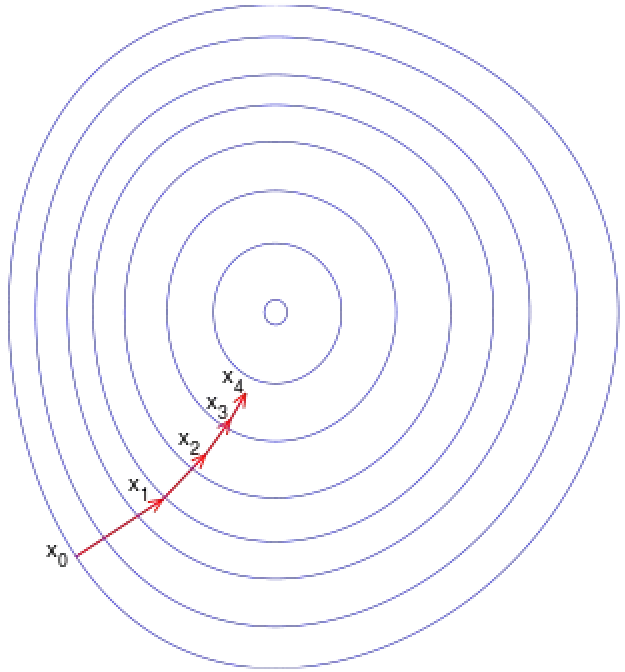
$$P(y|x)=\sigma(w\cdot x+b)$$

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

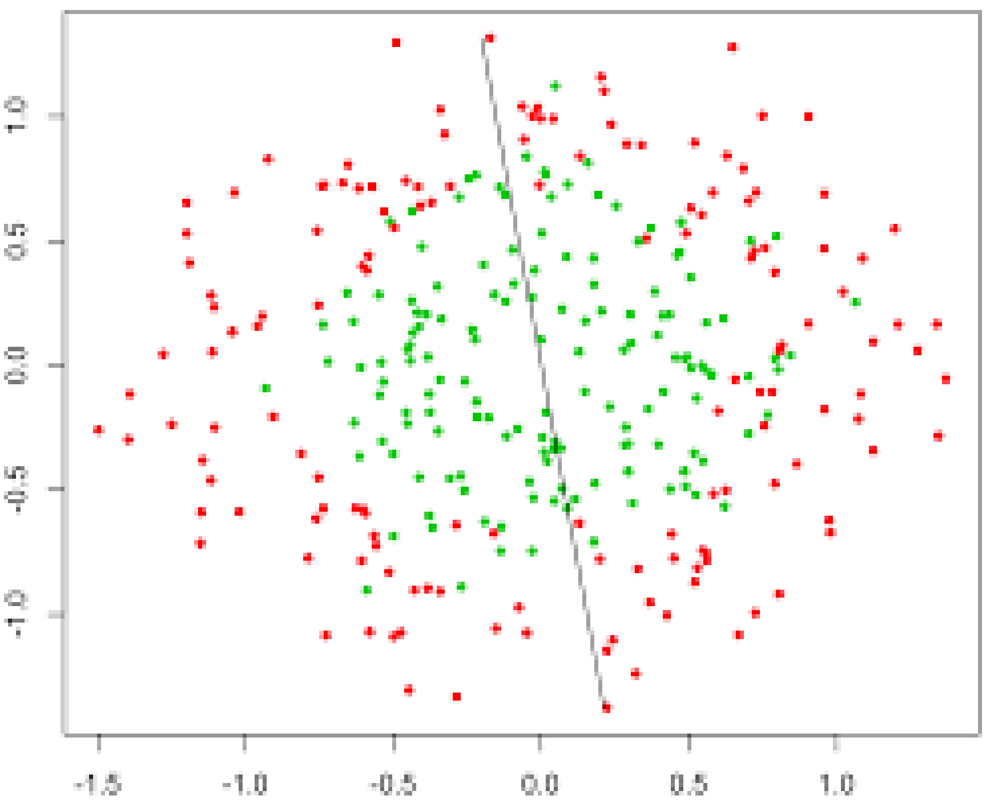
Repeat until convergence

$$\theta_j:=\theta_j-\alpha\cdot\frac{\partial L(y,y_{pred})}{\partial\theta_j}$$

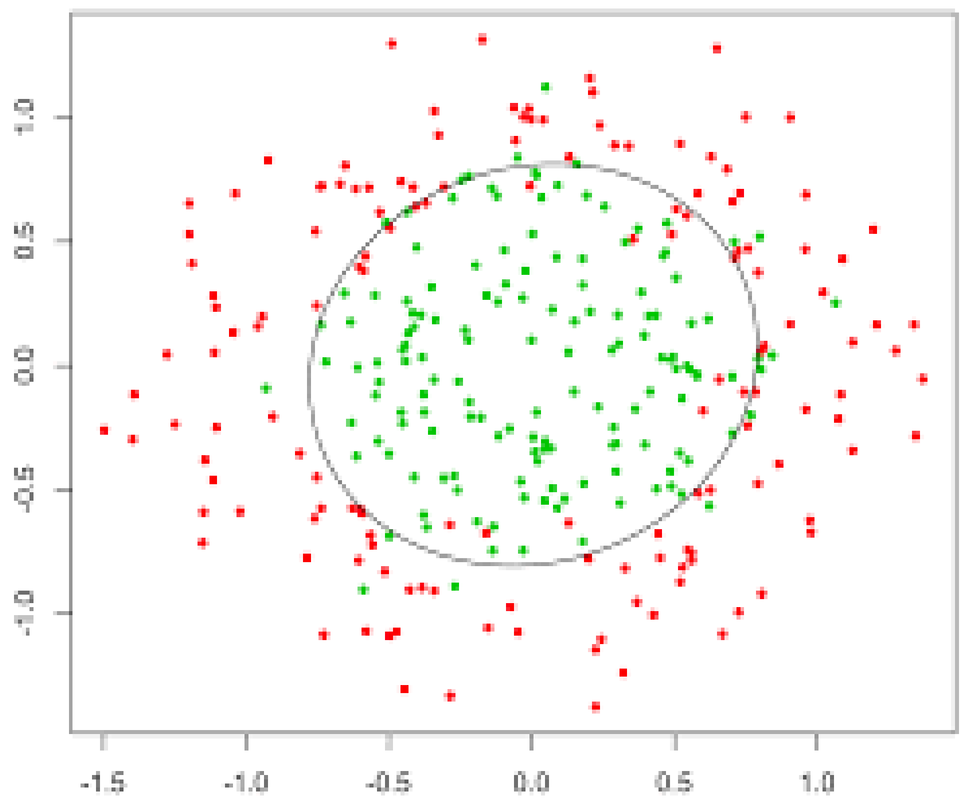
ϴ~{W,b}



Problem: Nonlinear dependencies



What we have



What we want

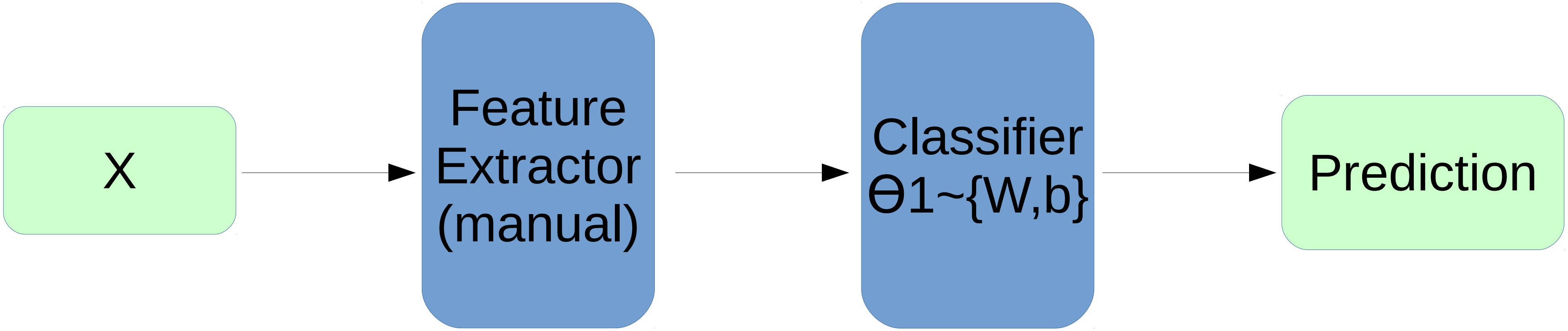
- **Trivia:** how do we solve that with logistic regression?

Feature extraction

Loss, for example:

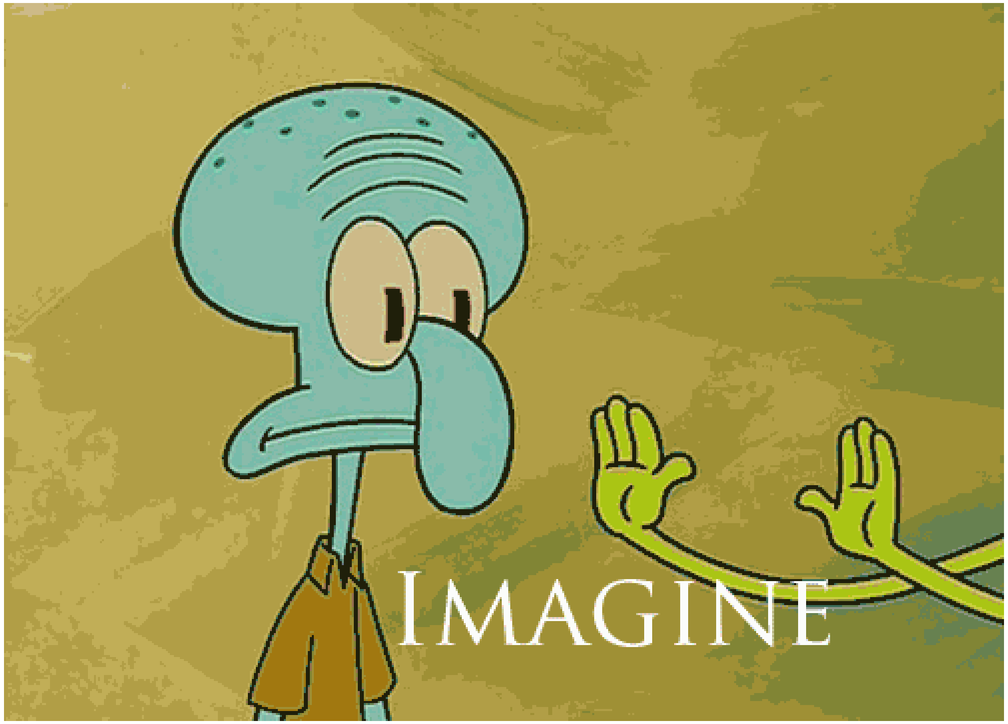
$$L = -\sum_i y_i \log P(y|x_i) + (1 - y_i) \log(1 - P(y|x_i))$$

Model:



Training:

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



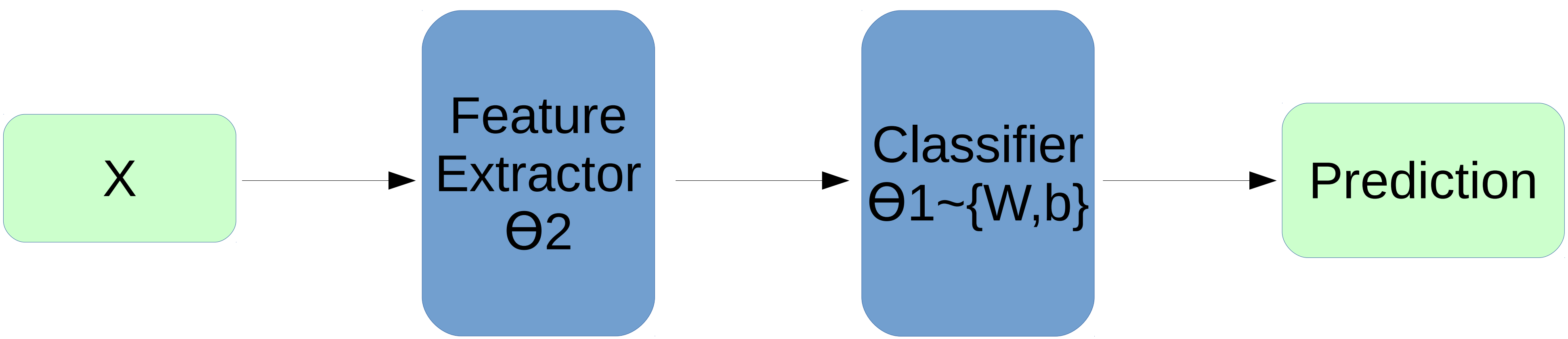
Features would tune to your problem automatically!

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:



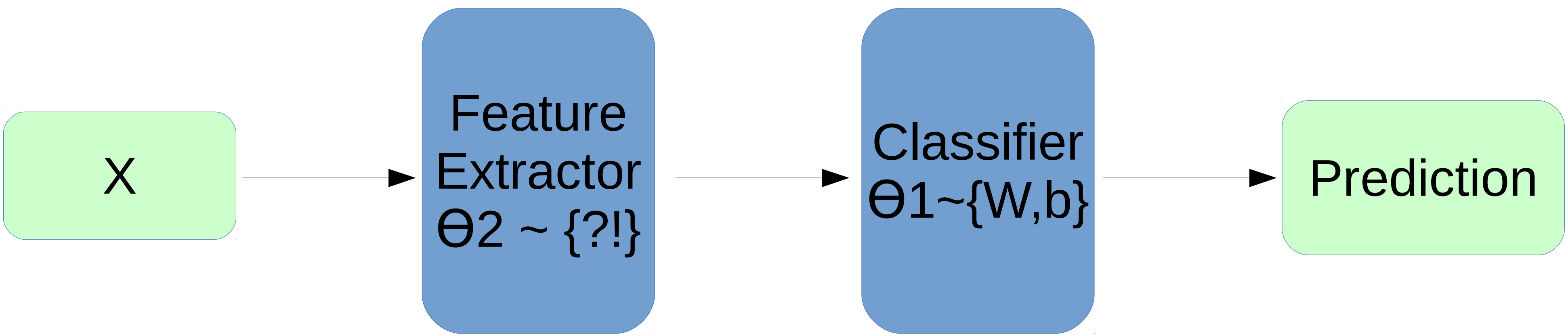
Training: ? $\underset{\theta_1}{argmin}\, L(y, P(y|x))$

What do we want, exactly?

Loss, for example:

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

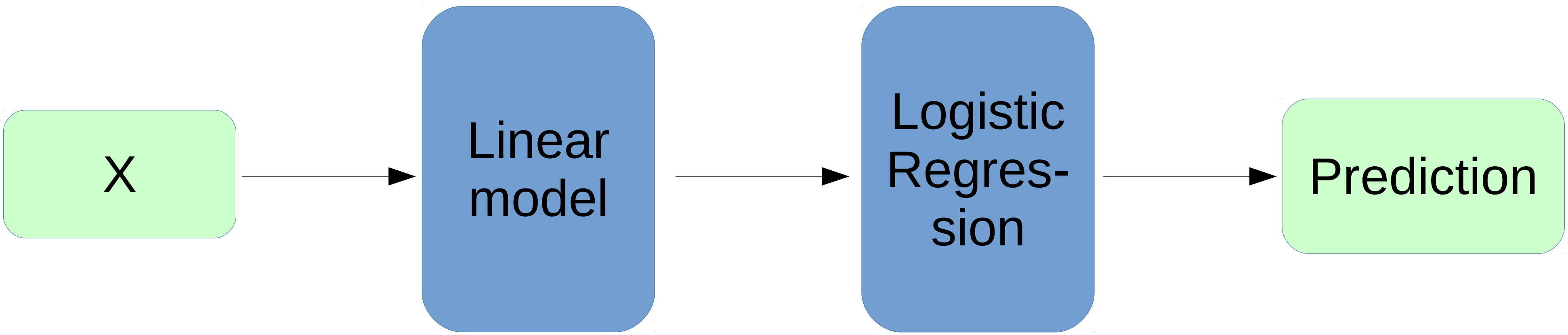
Model:



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

Try linear

Model:

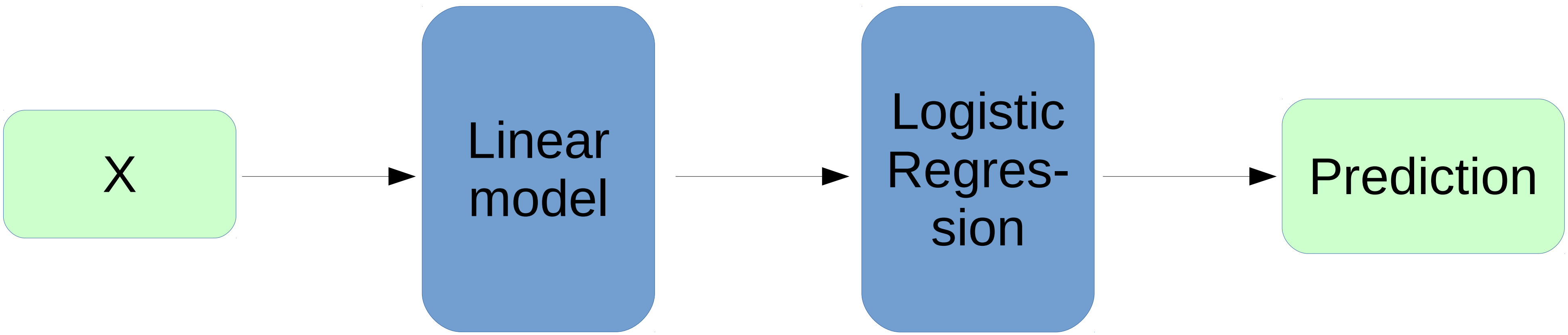


$$h_j = \sum_{i} w_{ij}^h x_i + b_j^h \qquad y_{pred} = \sigma\left(\sum_{j} w_j^o h_j + b^o\right)$$

$$j \in \{1, 2, \ldots, n\}$$

Try linear

Model:



$$h_j = \sum_{i} w^h_{ij} x_i + b^h_j \qquad j \in \{1,2,\ldots,n\}$$

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

Output:

$$P(y|x) = \sigma\left(\sum_j w^o_j \left(\sum_i w^h_{ij} x_i + b^h_j\right) + b^o\right)$$

Is it any better than logistic regression?

Try linear

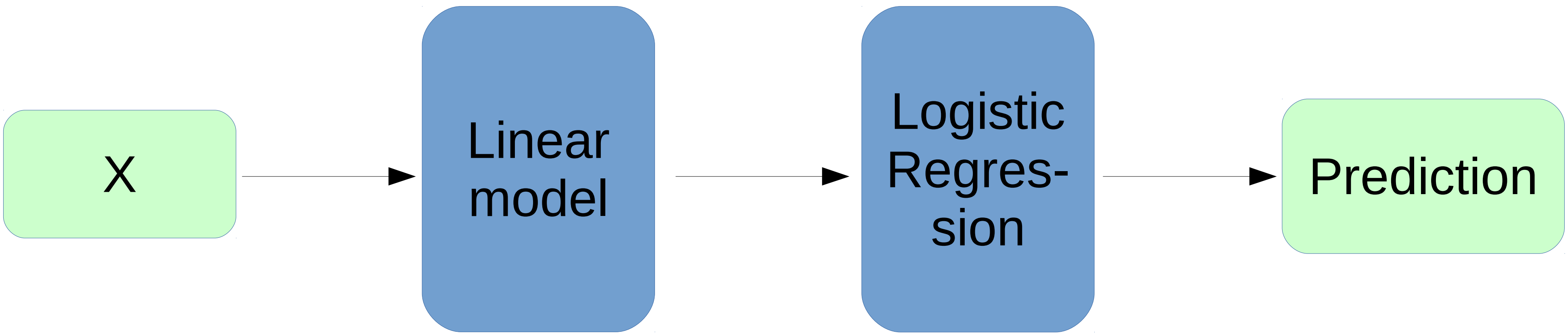
$$P(y|x)=\sigma\left(\sum_j w_j^o\left(\sum_i w_{ij}^h x_i+b_j^h\right)+b^o\right)$$

$$w'_i=\sum_j w_j^o w_{ij}^h \qquad b'=\sum_j w_j^o b_j^h+b^o$$

$$P(y|x)=\sigma\left(\sum_i w'_i x_i+b'\right)$$

Try linear

Model:



$$h_j = \sum_{i} w_{ij}^{h} x_i + b_j^{h} \qquad y_{pred} = \sigma\left(\sum_{j} w_j^{o} h_j + b^{o}\right)$$

$$j \in \{1,2,\ldots,n\}$$

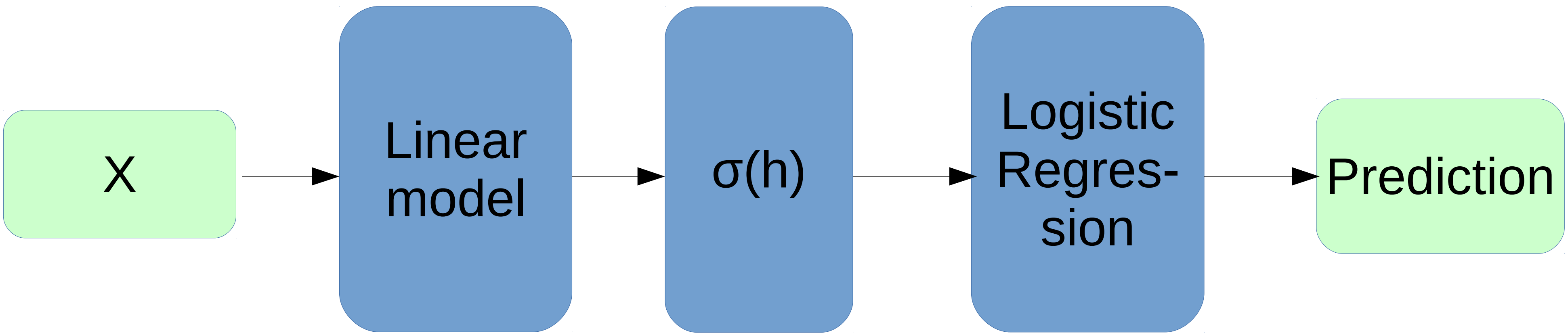
Output:

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

Is it any better than logistic regression?

Nonlinearity

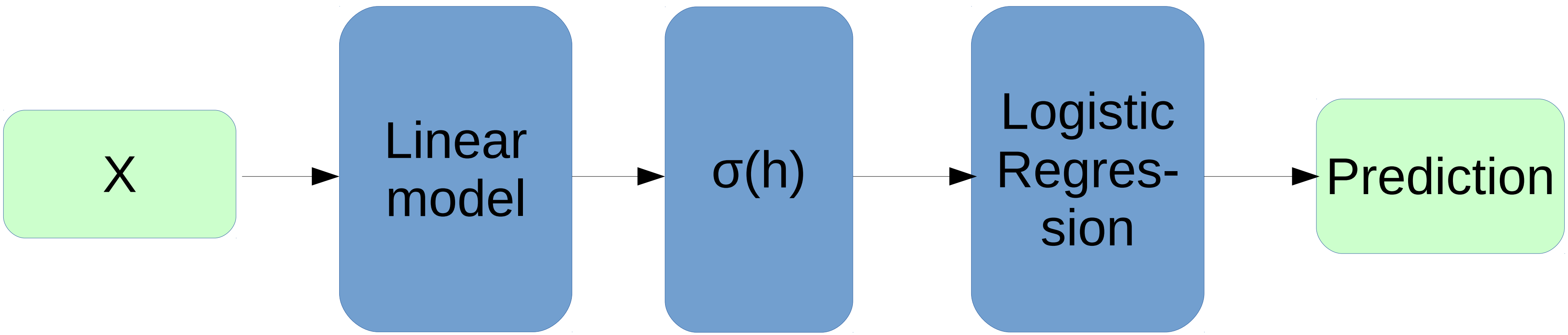
Model:



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

Nonlinearity

Model:



$$h_j = \sigma\left(\sum_{i} w_{ij}^{h} x_i + b_j^{h}\right) \qquad y_{pred} = \sigma\left(\sum_{j} w_j^{o} h_j + b^{o}\right)$$

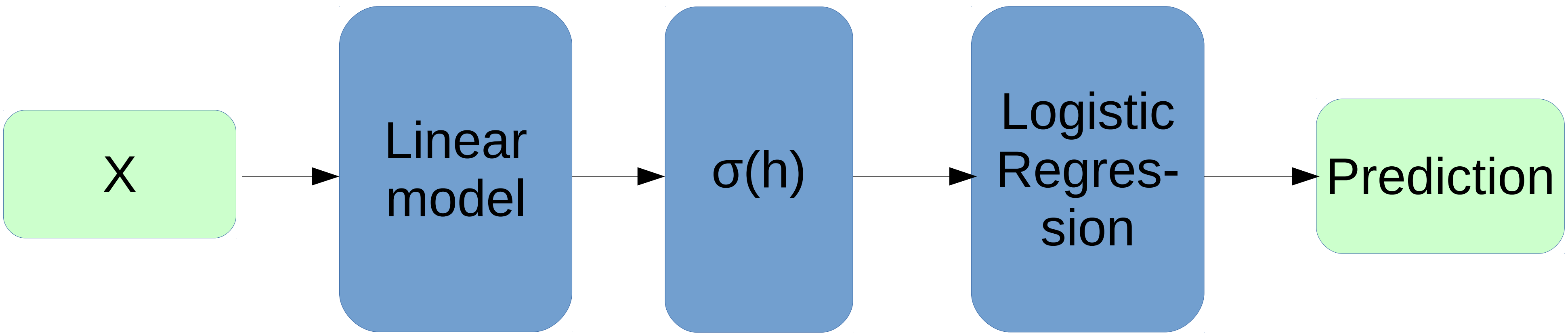
$$j \in \{1, 2, \ldots, n\}$$

Output:

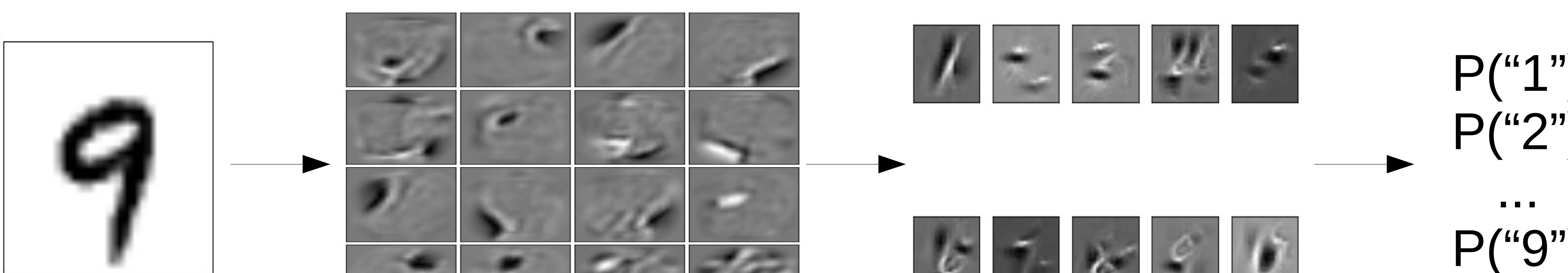
$$P(y|x) = \sigma\left(\sum_{j} w_j^{o} \sigma\left(\sum_{i} w_{ij}^{h} x_i + b_j^{h}\right) + b^{o}\right)$$

Nonlinearity

Model:

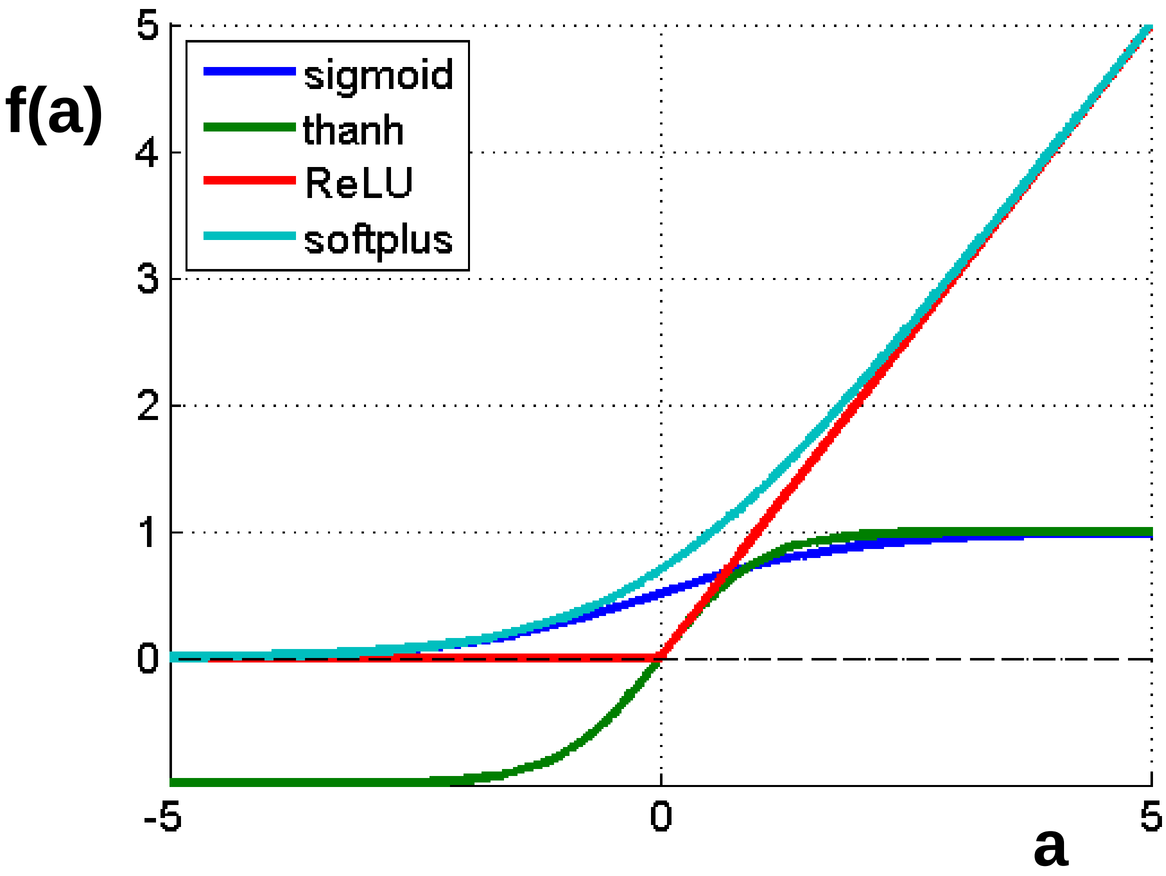


$$h_j = \sigma\left(\sum_{i \in \{1,2,\ldots,n\}} w_{ij}^h x_i + b_j^h\right) \qquad y_{pred} = \sigma\left(\sum_j w_j^o h_j + b^o\right)$$

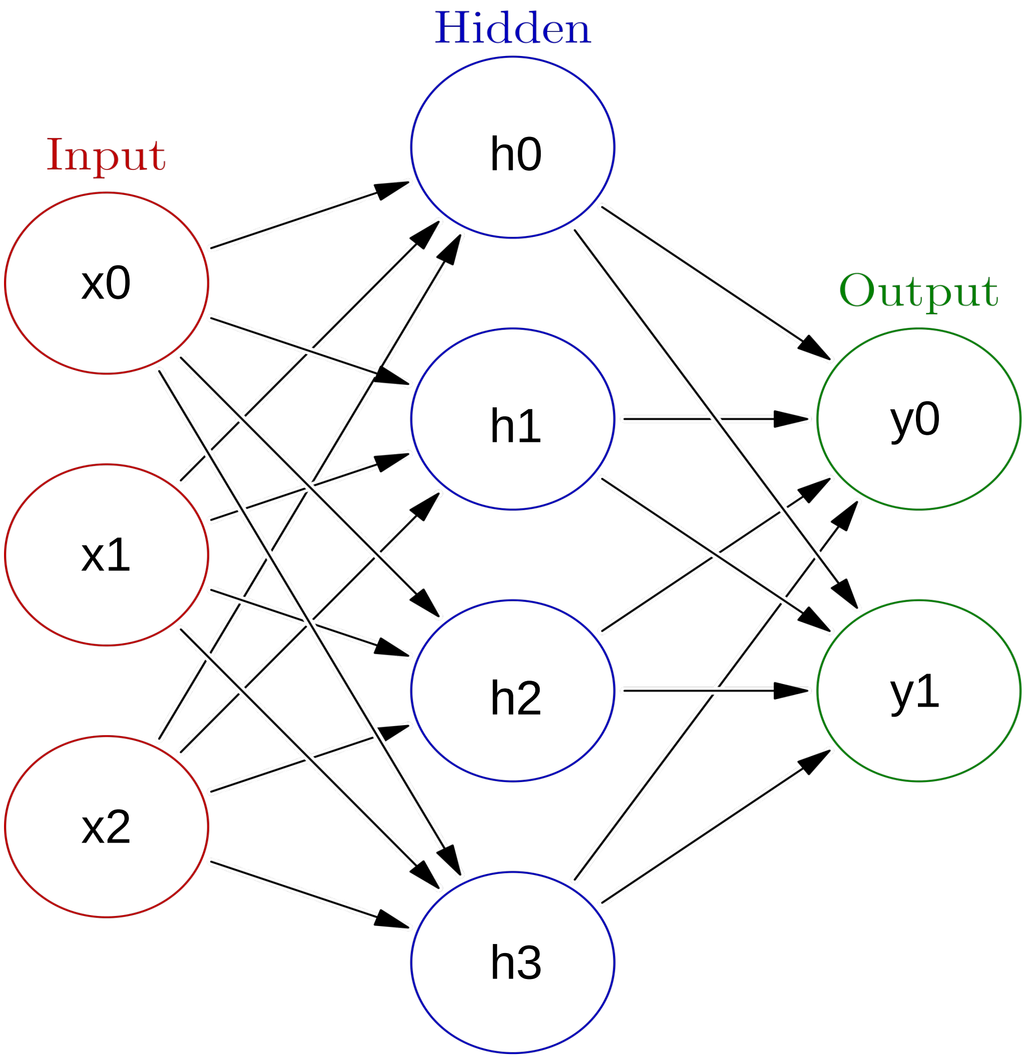


Nonlinearity

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



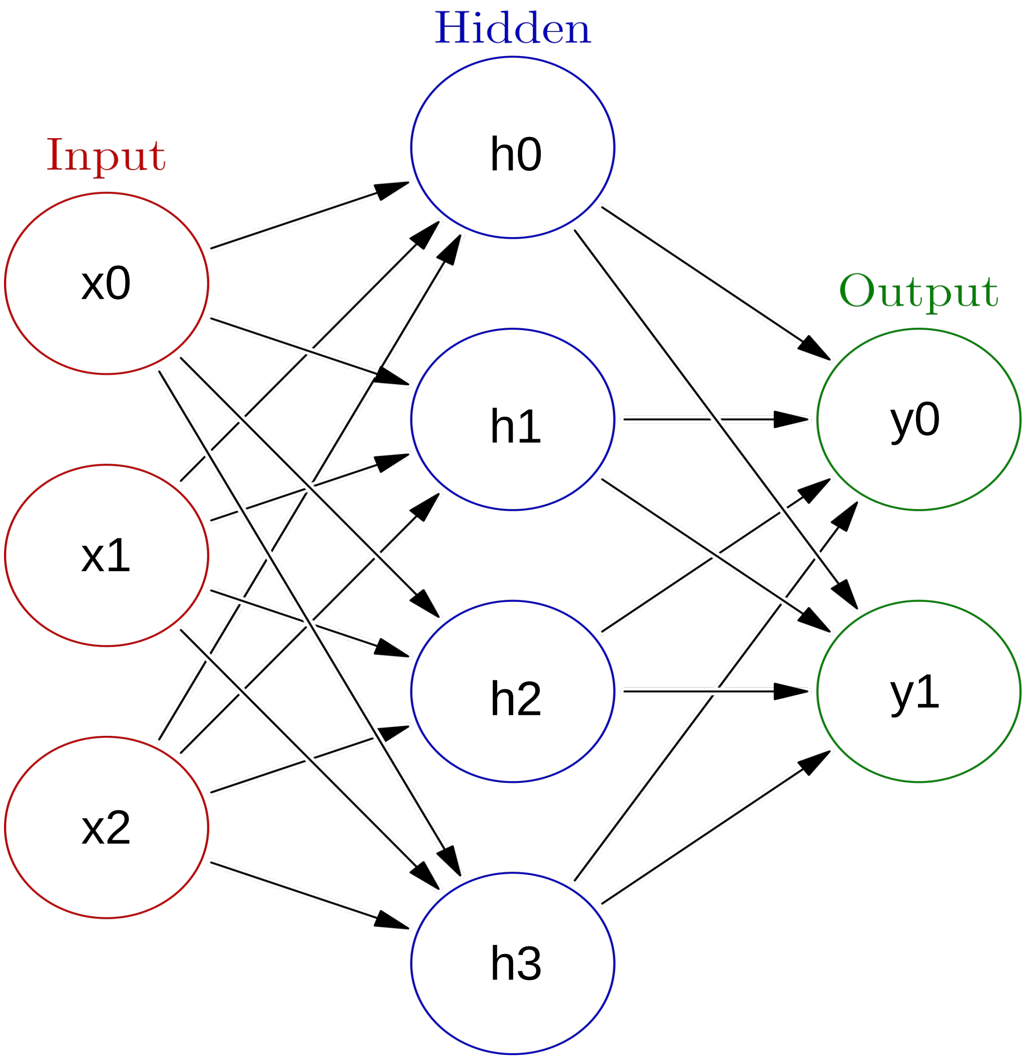
Initialization, symmetry problem



- Initialize with zeros
  $W \leftarrow 0$

- What will the first step look like?

Initialization, symmetry problem



- Break the symmetry!
- Initialize with random numbers!
  $W \leftarrow N(0,0.01)$?
  $W \leftarrow U(0,0.1)$?
- Can get a bit better for deep NNs

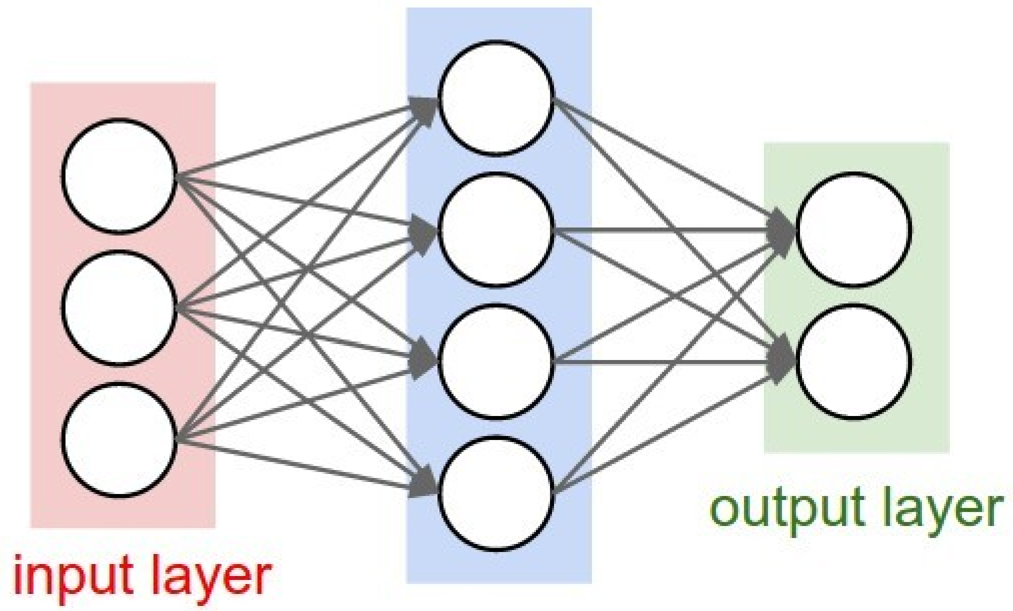
Connectionist phrasebook

- Layer – a building block for NNs :
  - “Dense layer”: $f(x) = Wx+b$
  - “Nonlinearity layer”: $f(x) = \sigma(x)$
  - Input layer, output layer
  - A few more we gonna cover later

- Activation – layer output
  - i.e. some intermediate signal in the NN

- Backpropagation – a fancy word for “chain rule”

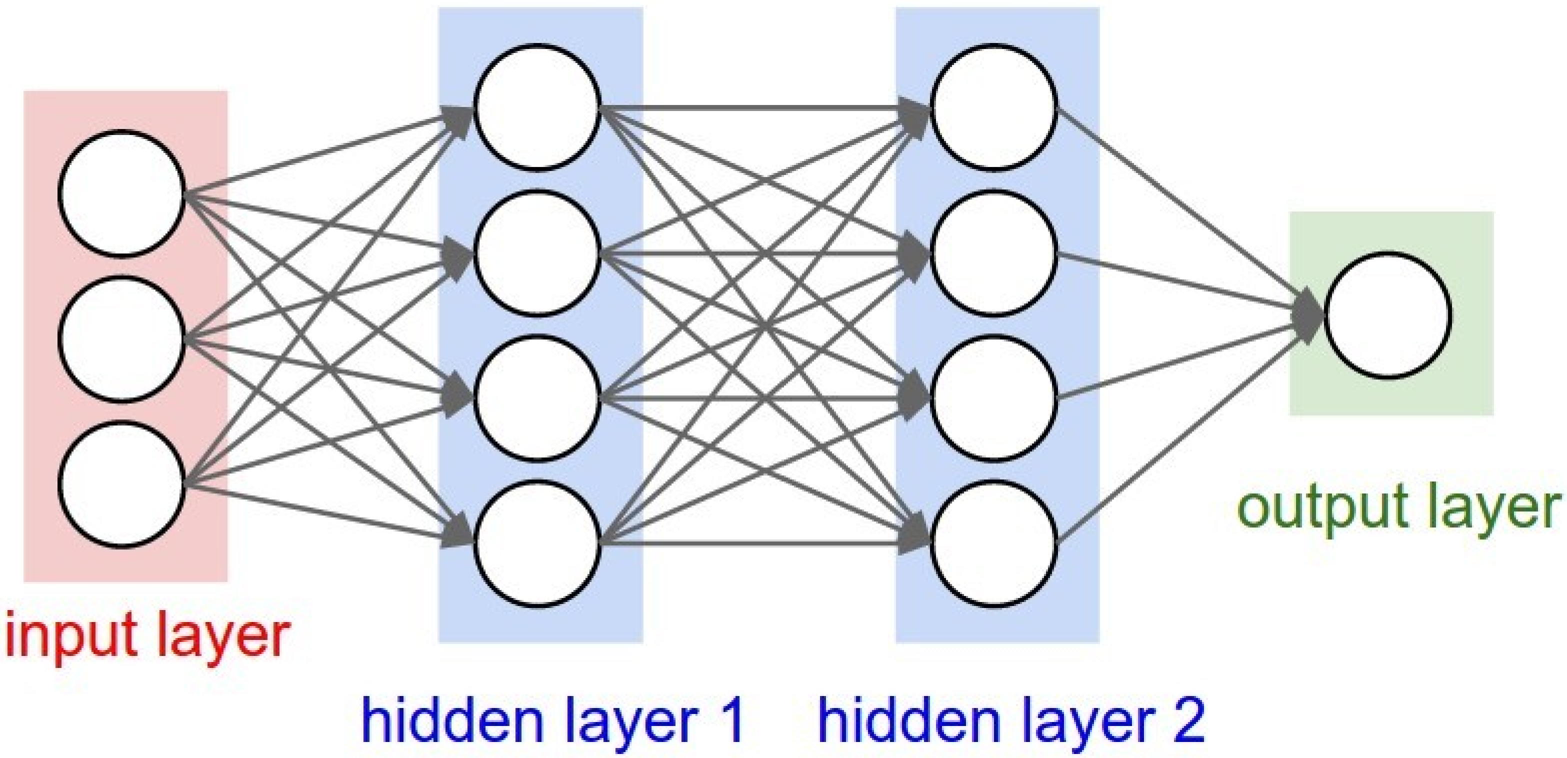
Connectionist phrasebook



Dense layer
with a nonlinearity

- “Train it via backprop!”

Connectionist phrasebook



How do we train it?

Image recognition

“Dog”

Image recognition

“Gray wall”

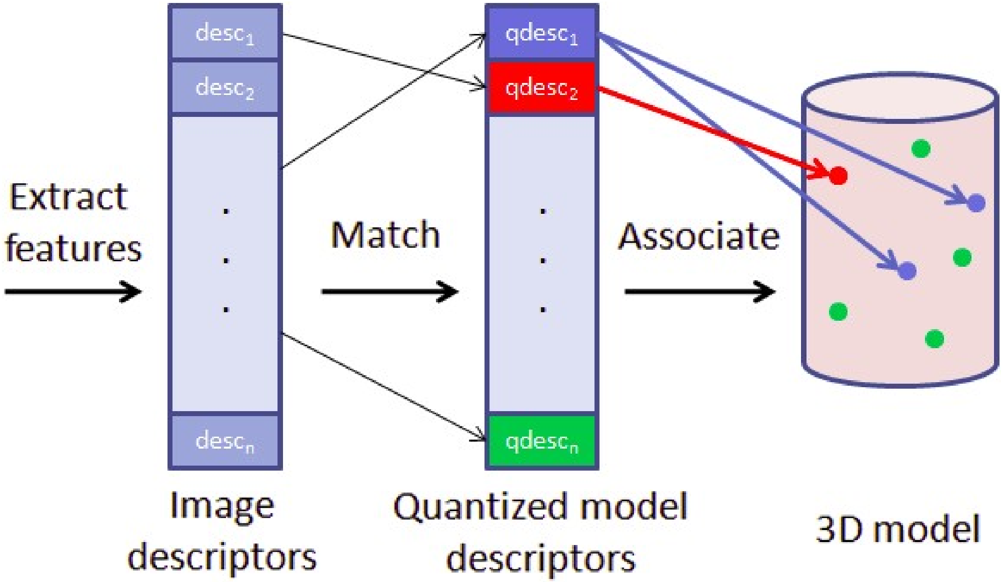
“Dog tongue”

“Dog”

<a particular kind of dog>

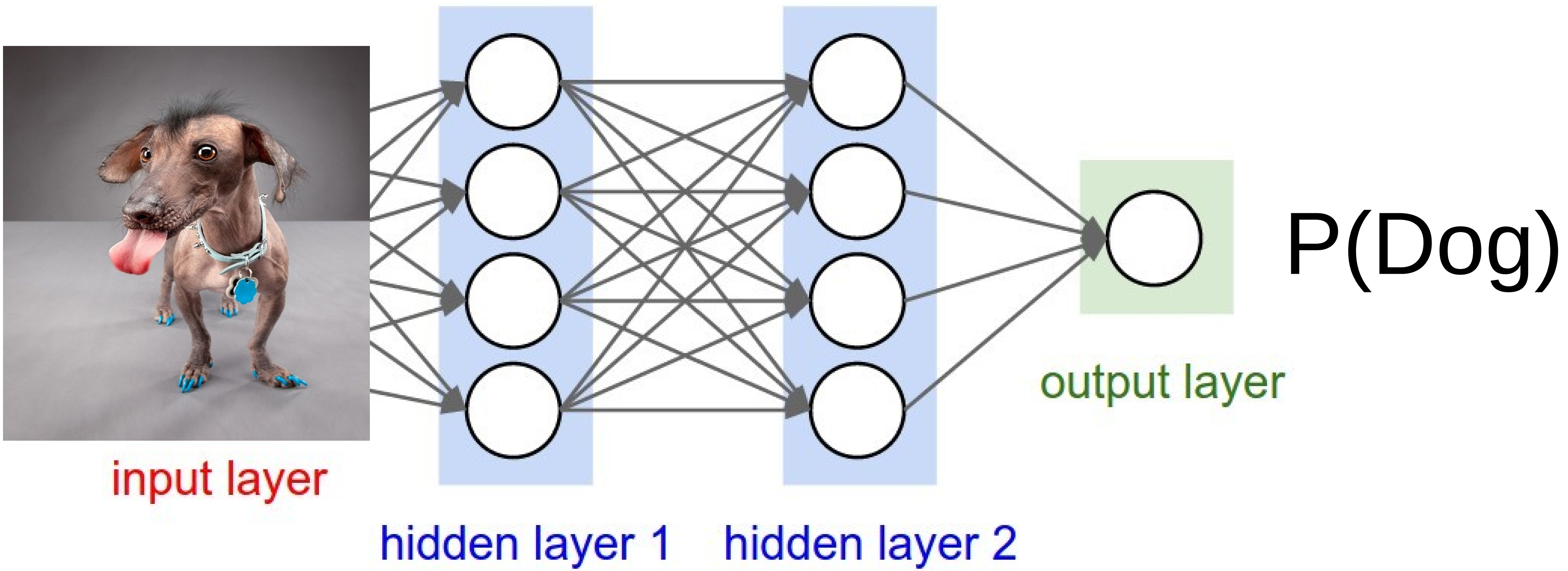
“Animal sadism”

Classical approach

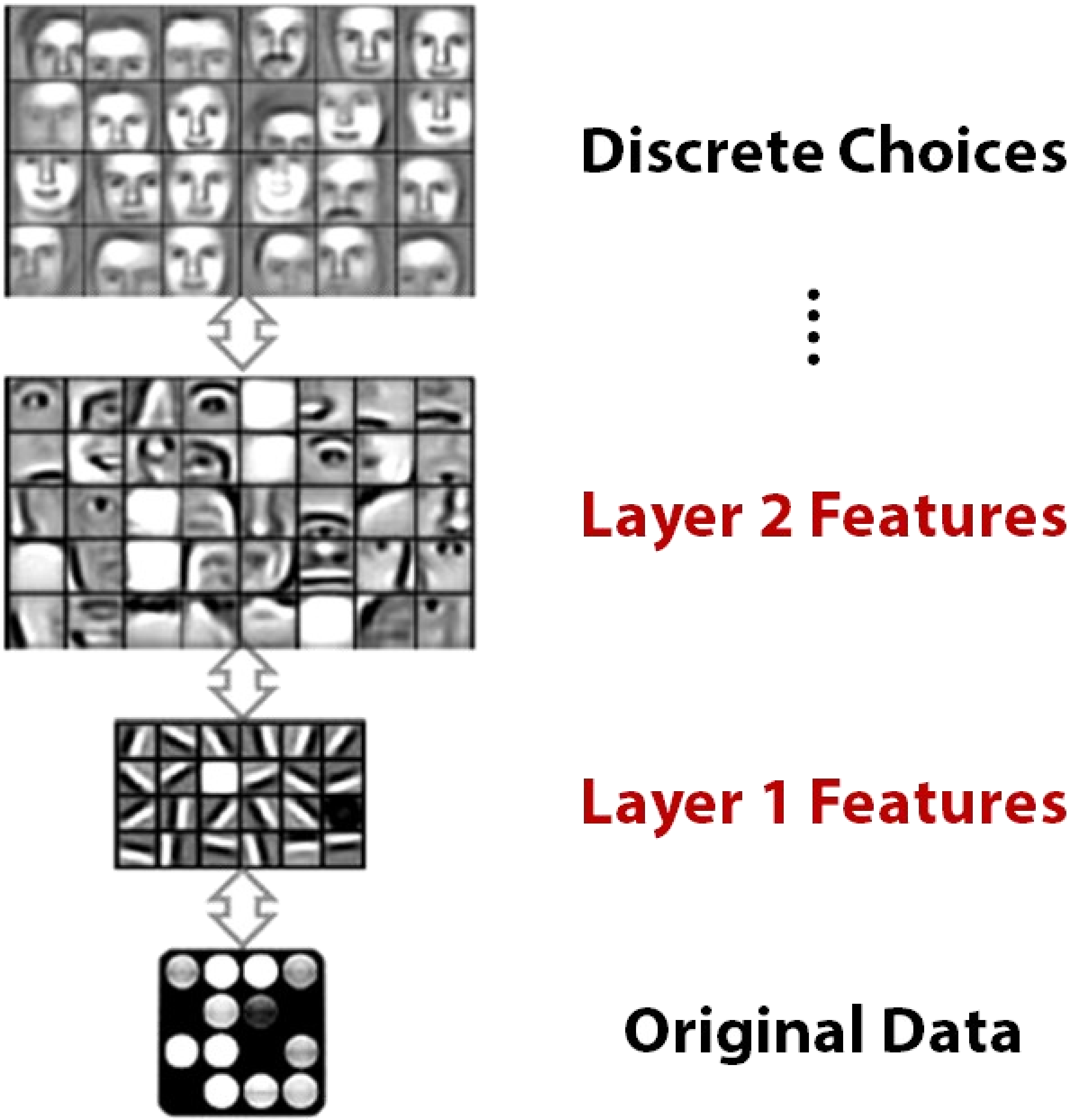


“Dog”

NN approach



What features could NN learn this way?


Discrete Choices
Layer 2 Features
Layer 1 Features
Original Data

Problem

Should we require, say, “Dog ear” feature

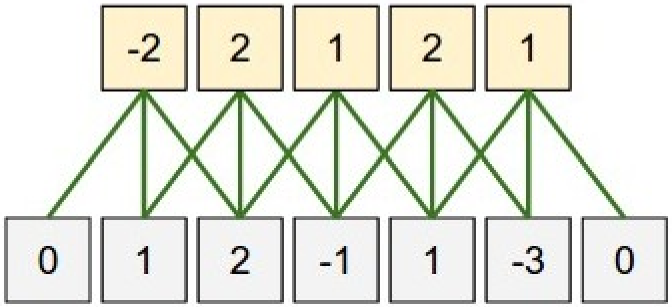
- Linear combination can only select dog ear at a one (or a few) positions.
- Need to learn independent features for each position
- Next layer needs to react on “dog ear 0,0 or dog ear 0,1 or … or dog ear 255,255”
- Introduce **a lot** of parameters and risk overfitting.

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

Convolution

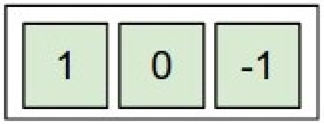
- Apply same weights to all patches

output

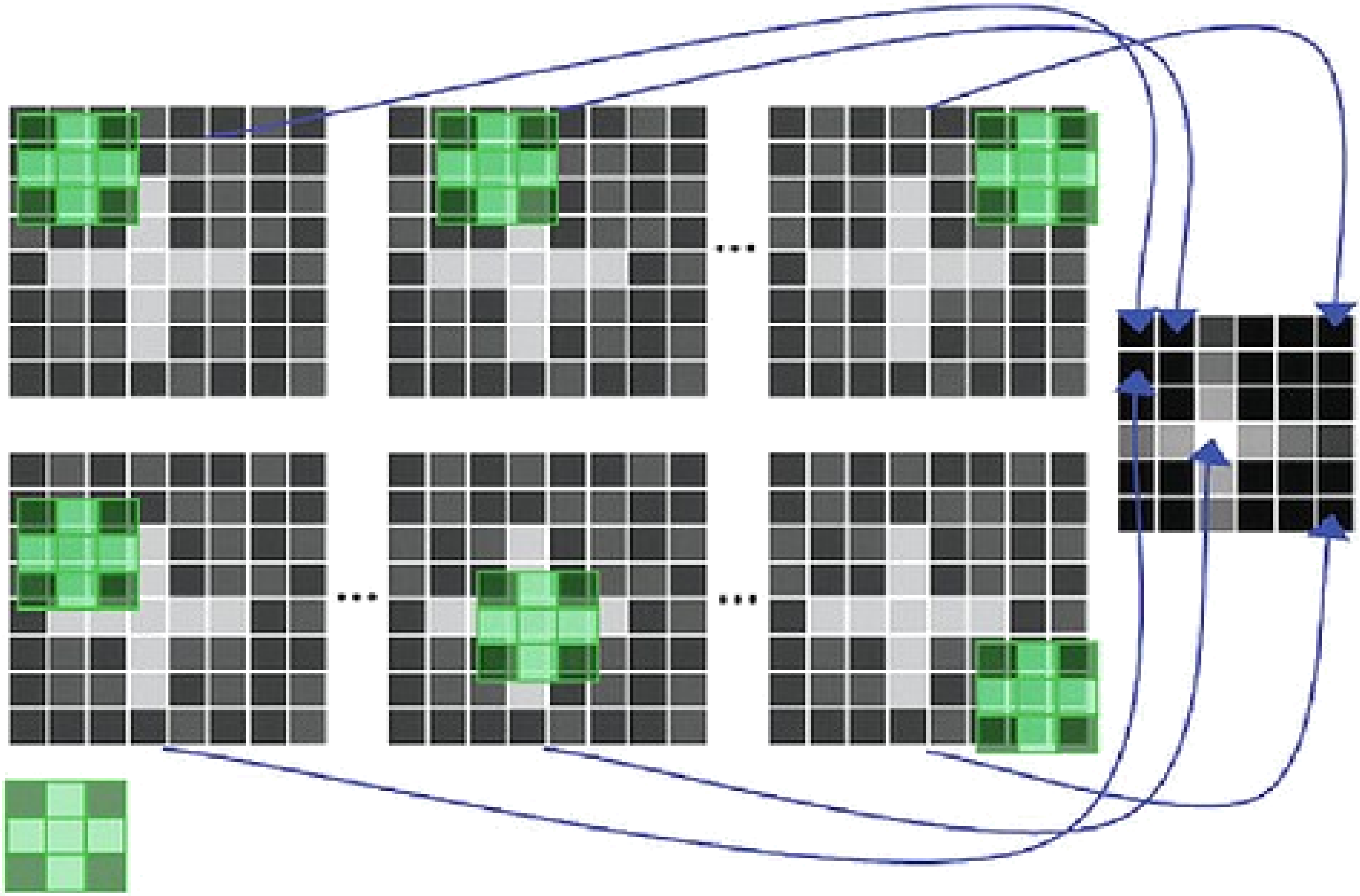


image

weights



Convolution

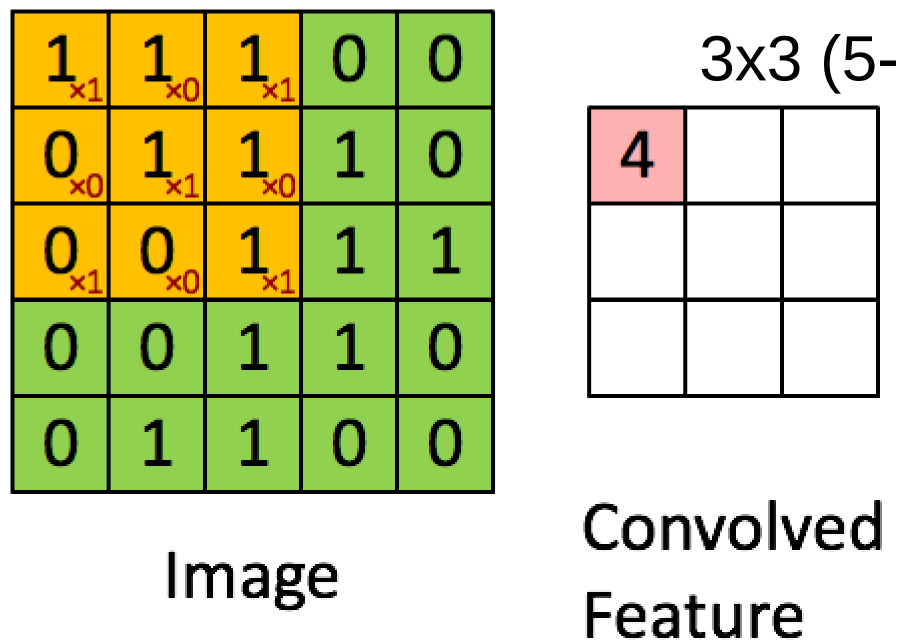


apply same filter to all patches

Convolution

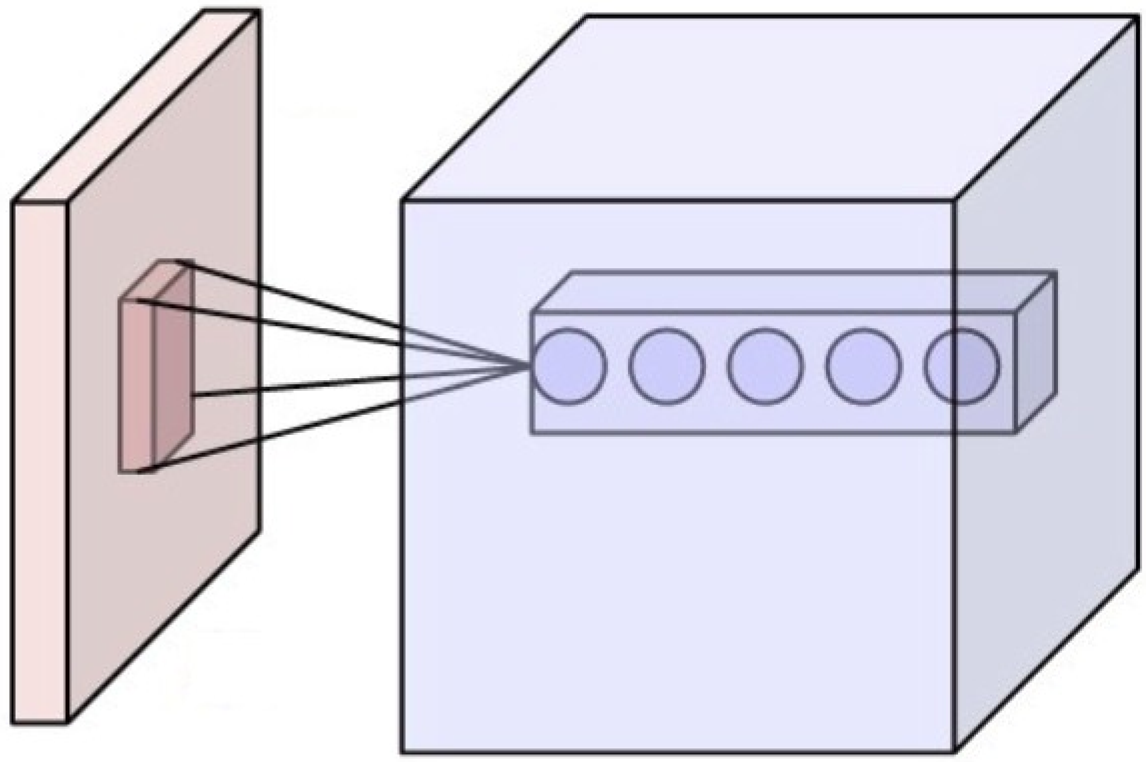
5x5

3x3 (5-3+1)



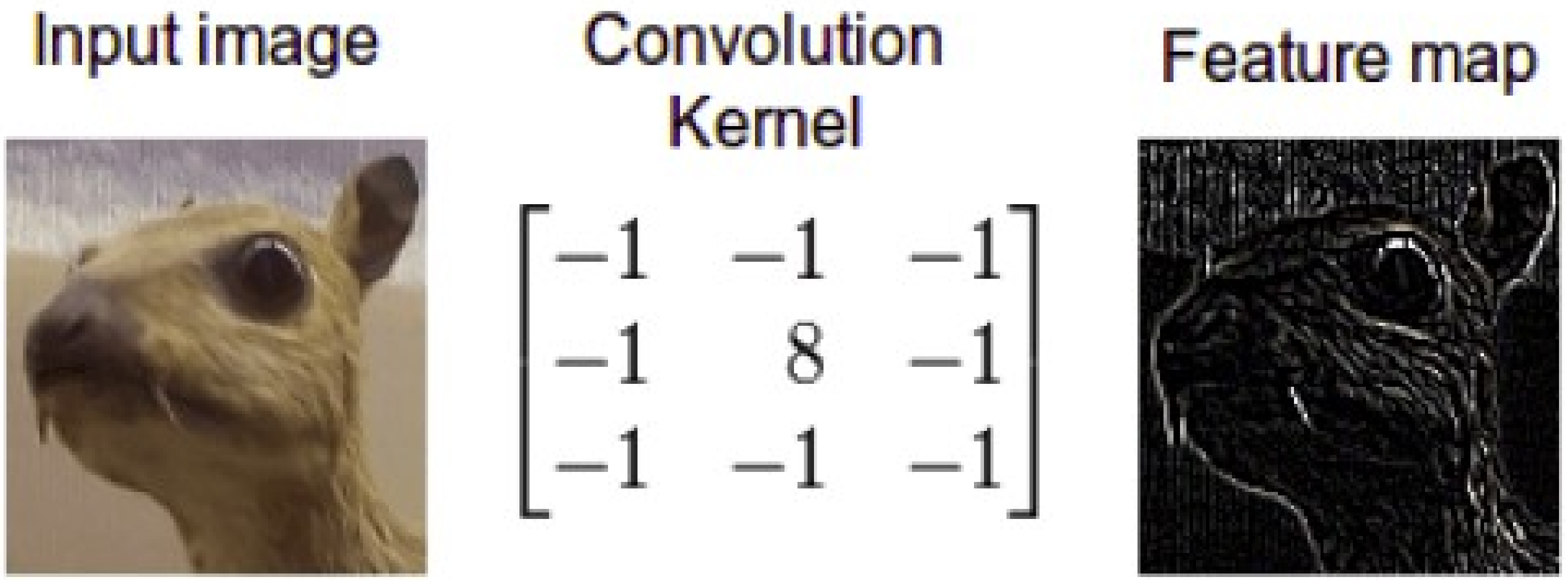
Intuition: how cat-like is this square?

Convolution



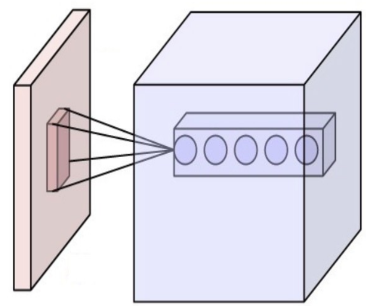
Intuition: how cat-like is this square?

Convolution



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

Convolution

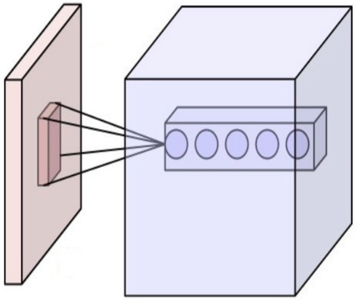


?!

Filters: 100x(3x5x5)

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

Convolution



100x96x96

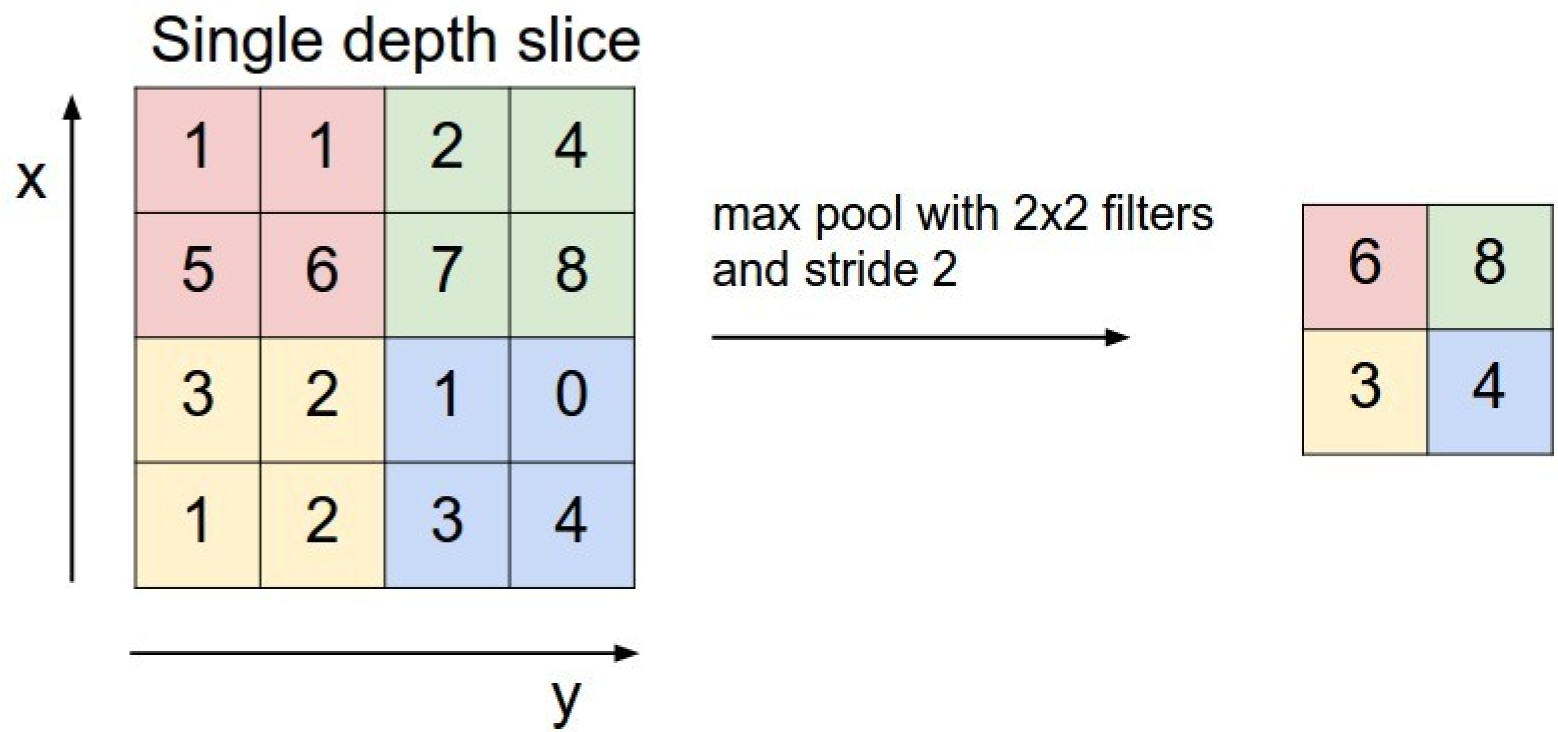
~10^6

Filters: 100x(3x5x5)

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

Somewhat too many!

Pooling



Intuition: What is the max cat-likelihood over this area?

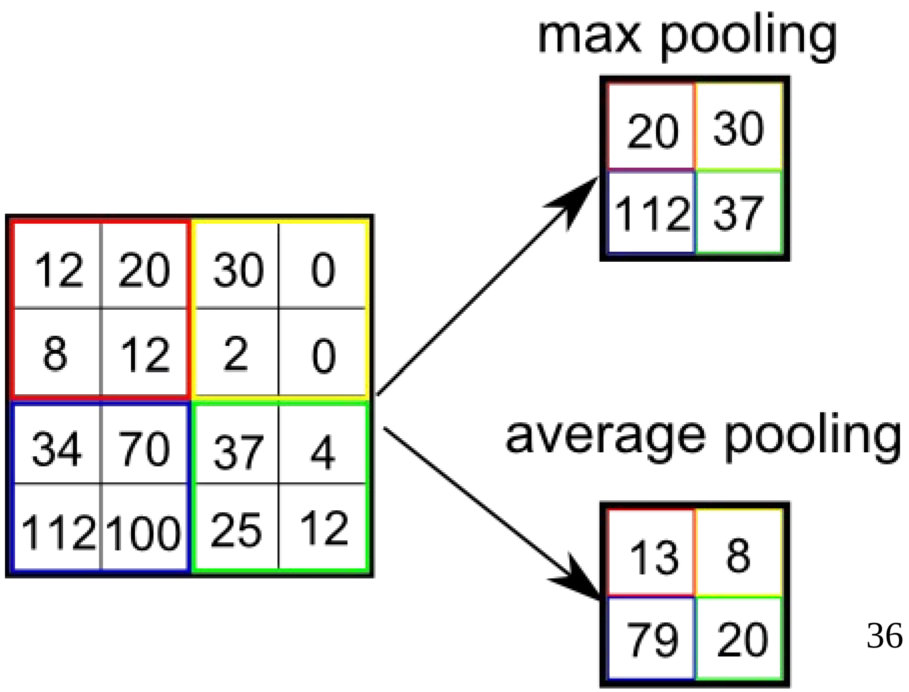
Pooling

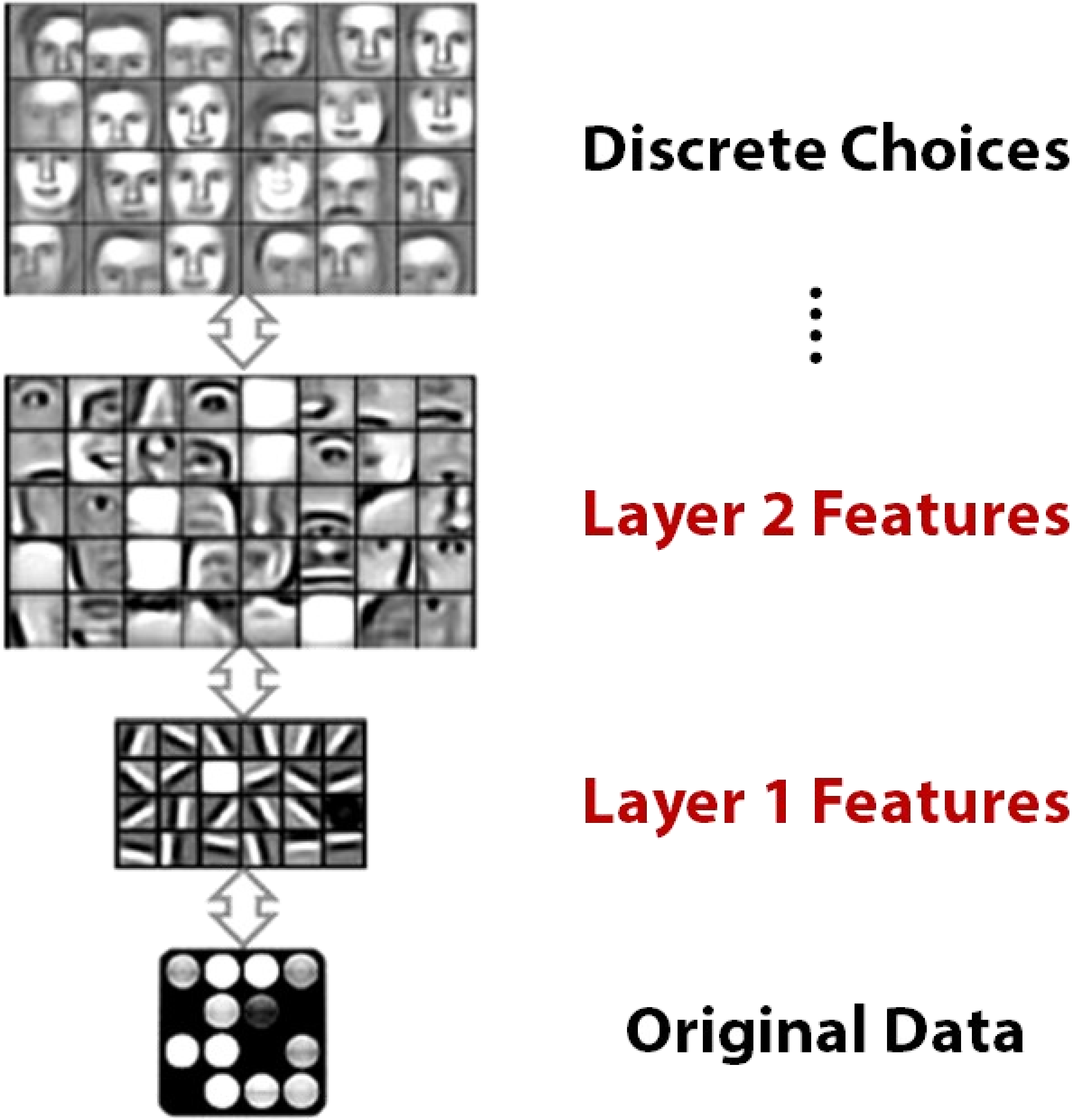
Motivation:

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

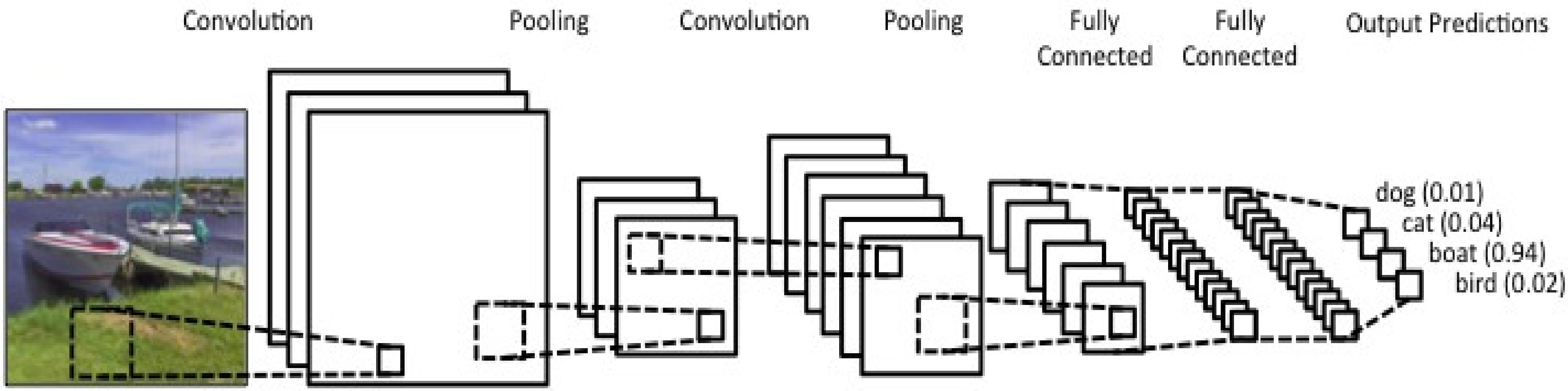
Popular types:

- Max
- Mean(average)

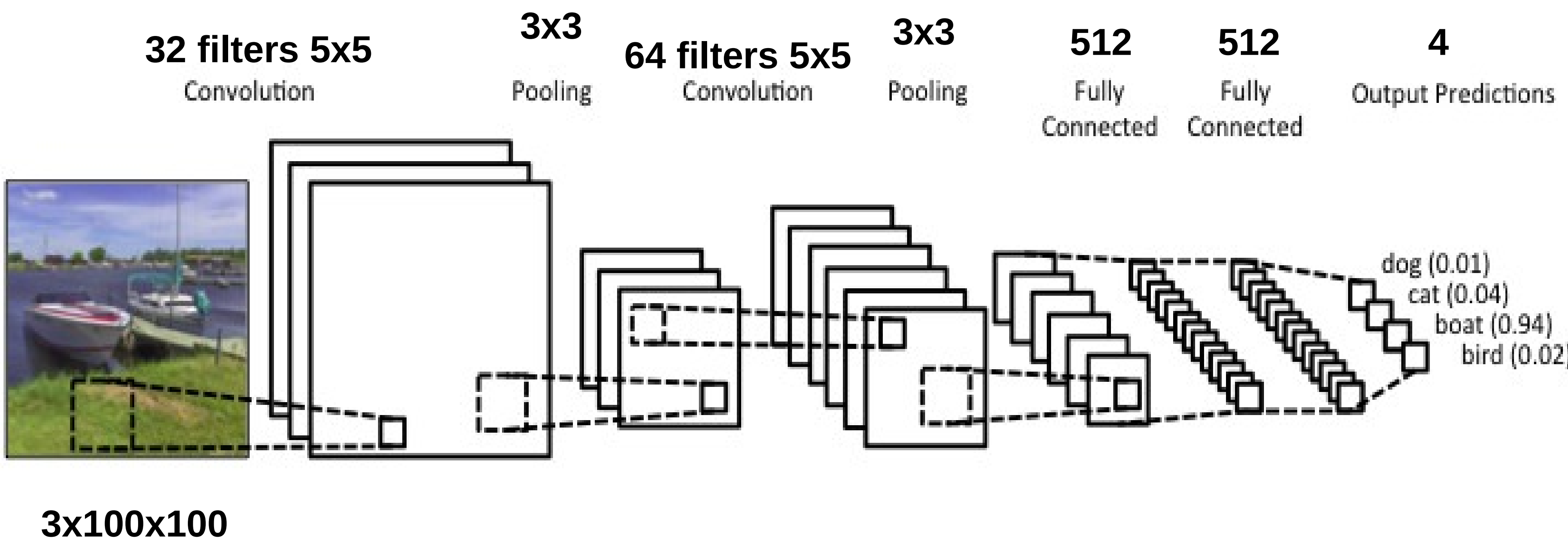



Discrete Choices
⋮
Layer 2 Features
Layer 1 Features
Original Data

Convolutional NNs



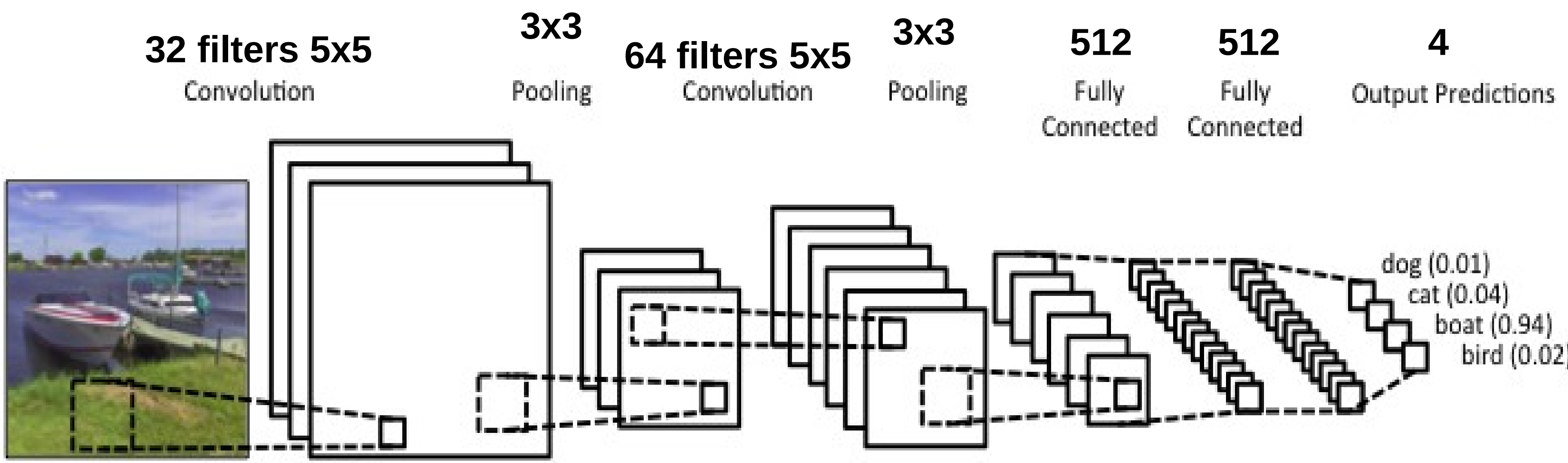
Convolutional NNs



Quiz:

1) What is the blob size **after second pooling**

Convolutional NNs

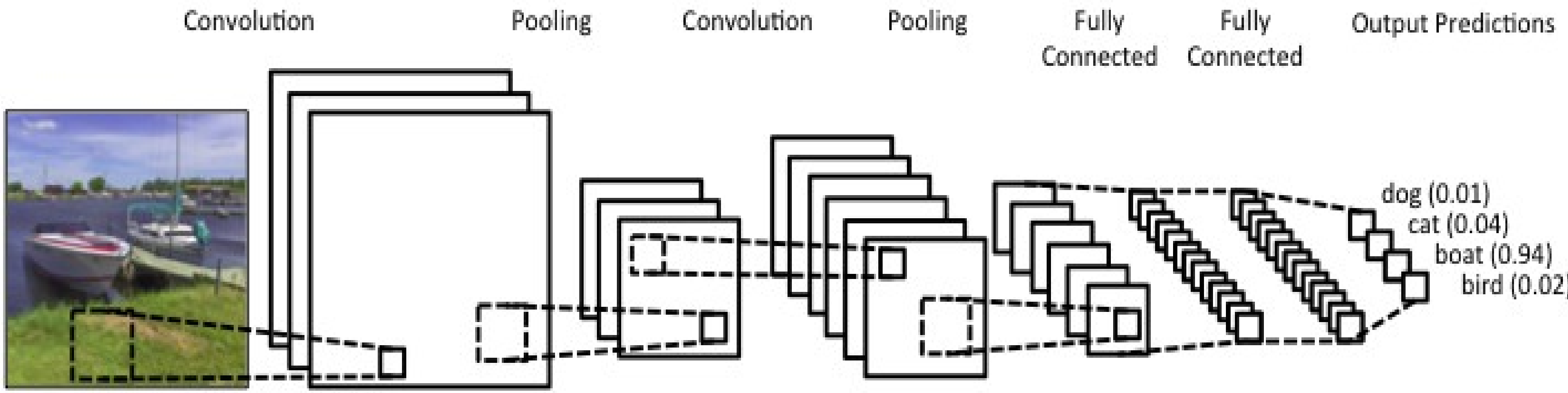


3x100x100

Quiz:
2) How many image pixels does **one cell** after **second convolution** depend on?

Convolutional NNs

32 filters 5x5 **3x3** **64 filters 5x5** **3x3** **512** **512** **4**



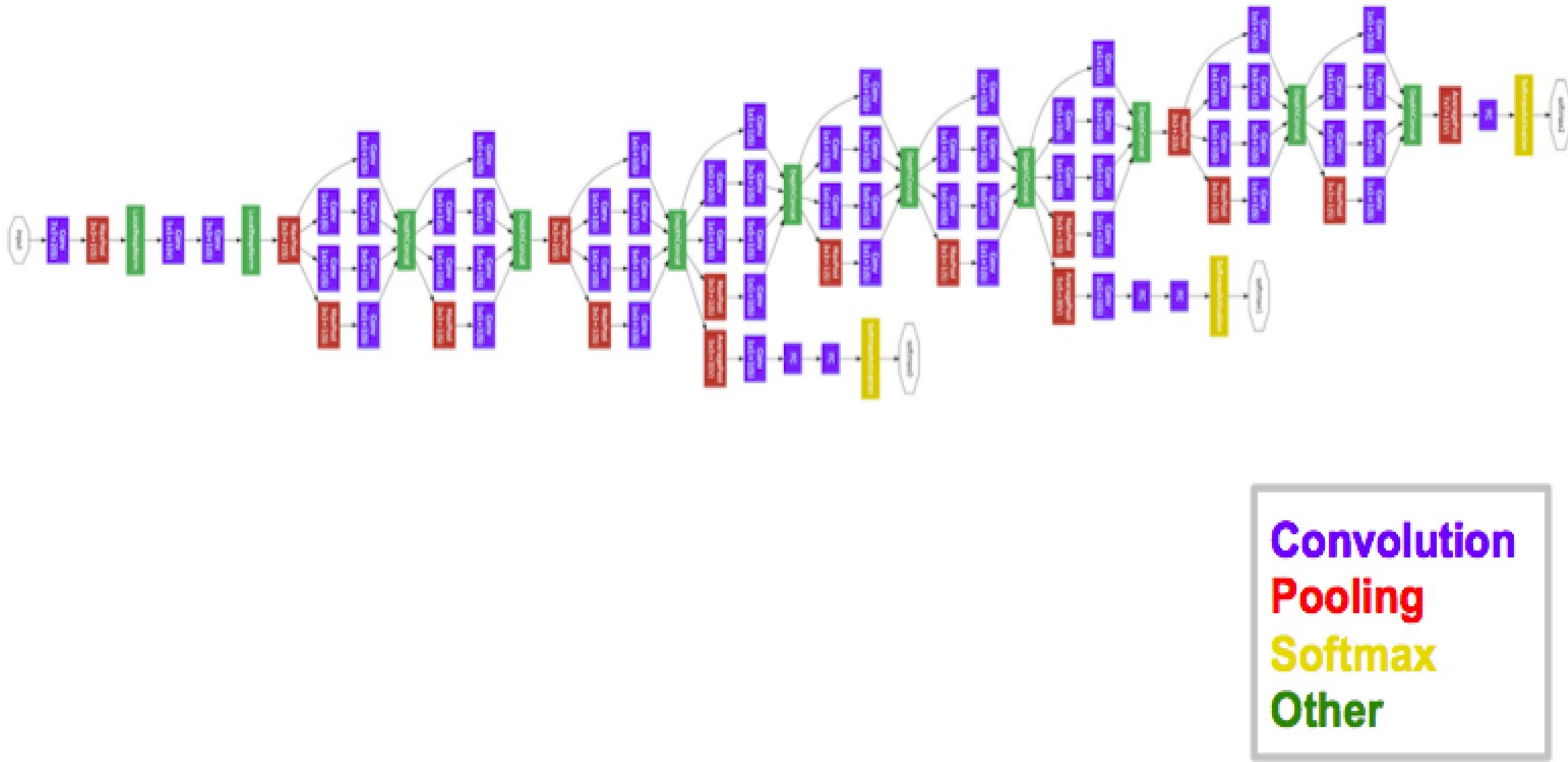
3x100x100

Quiz:

3) Which layer is hardest to compute?

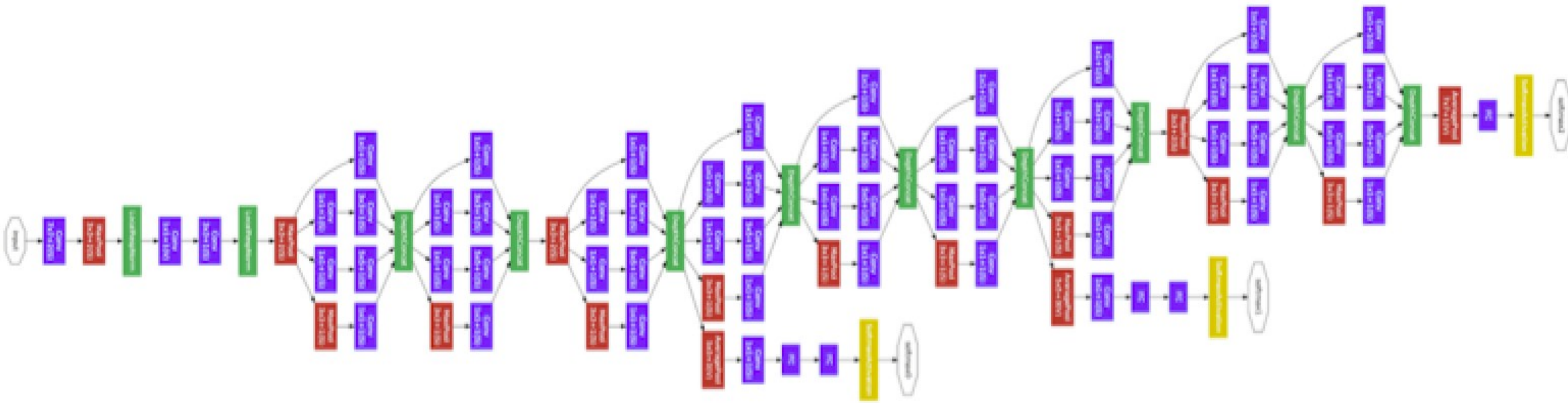
4) Which layer has most independent parameters?

Inception-GoogleNet

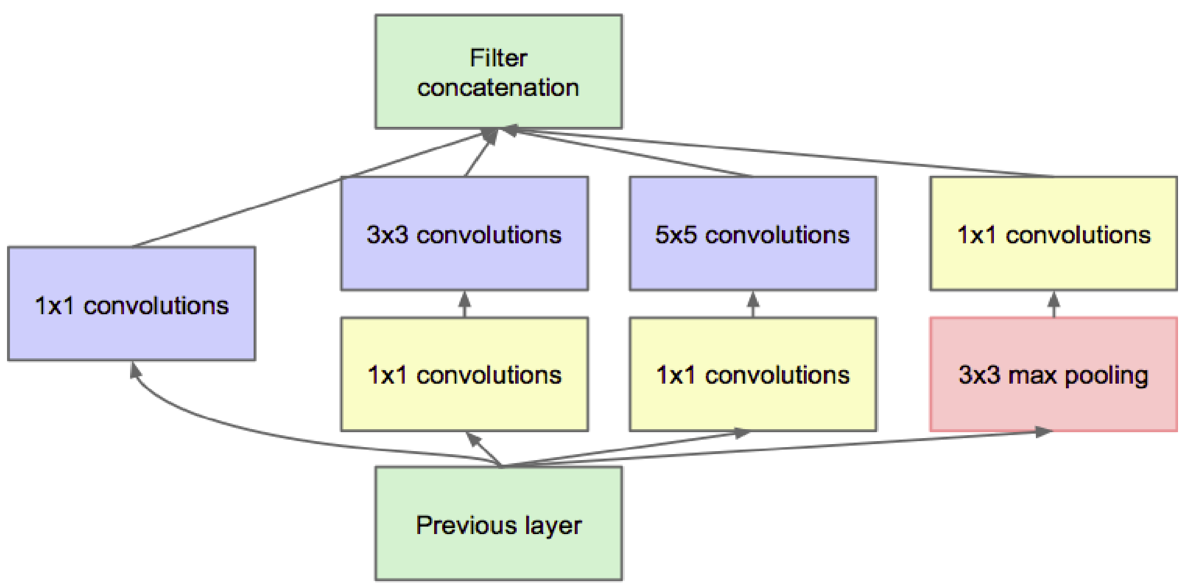


It is not a moon. It is a space station (c)

Inception-GoogleNet



Convolution
Pooling
Softmax
Other



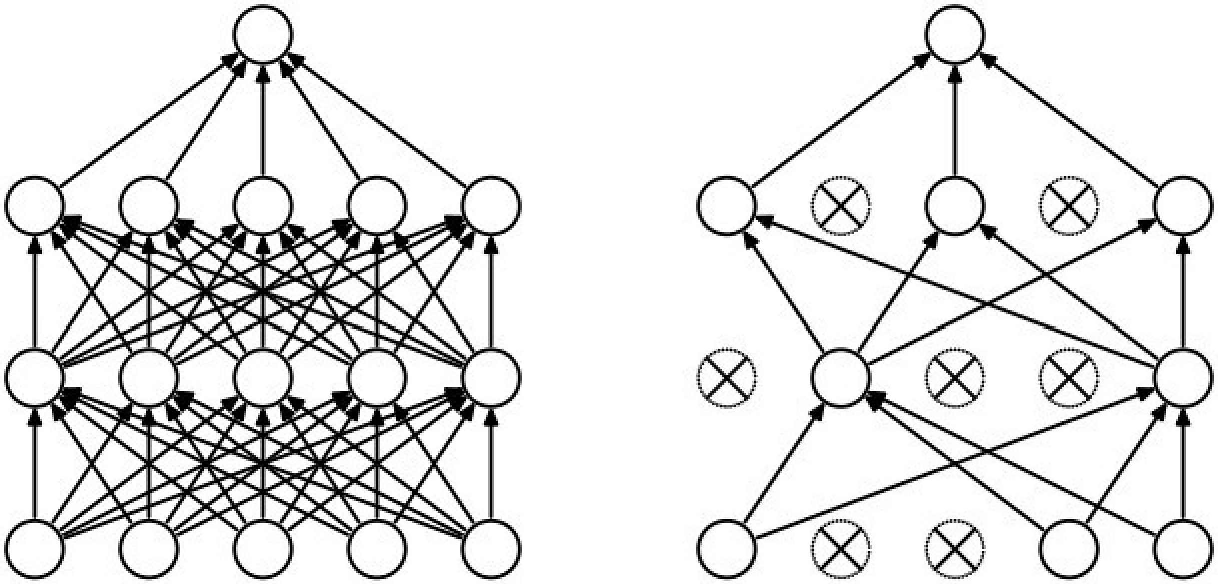
Potential caveats?

- Hardcore overfitting
- No “golden standard” for architecture
- Computationally heavy

Regularization

- L1, L2, as usual

- Dropout



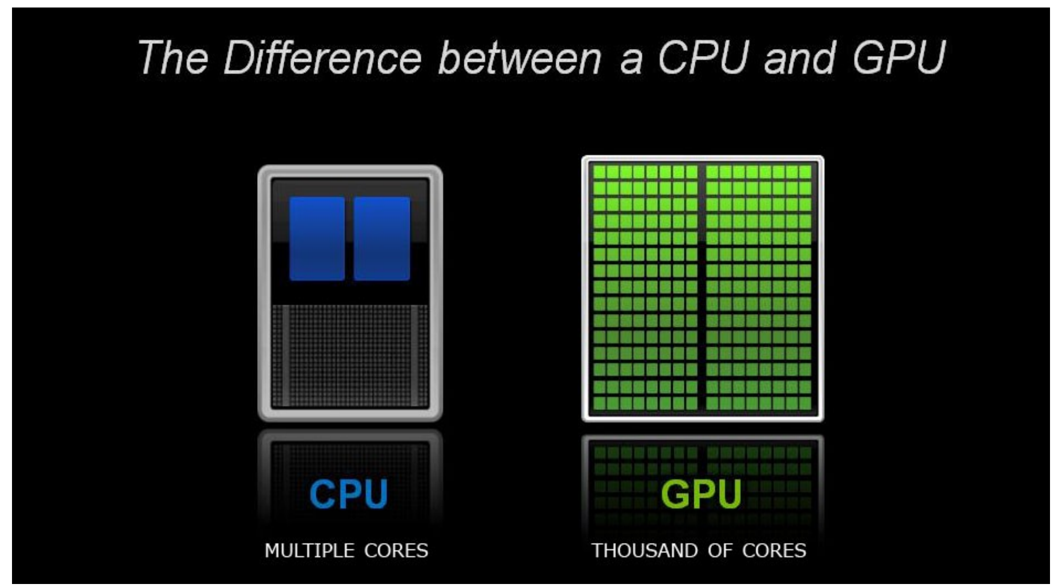
(a) Standard Neural Net (b) After applying dropout.

Data augmentation

- Idea: we can get N times more data by tweaking images.
- If you rotate cat image by 15°, it's still a cat

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

Computation



Batch normalization

Problem:

- Consider a neuron in any layer beyond first
- At each iteration we tune it's weights towards better loss function
- But we also tune it's inputs. Some of them become larger, some – smaller
- Now the neuron needs to be re-tuned for it's new inputs

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)

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

- 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

Idea:

– We normalize activation of a hidden layer

(zero mean unit variance)

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

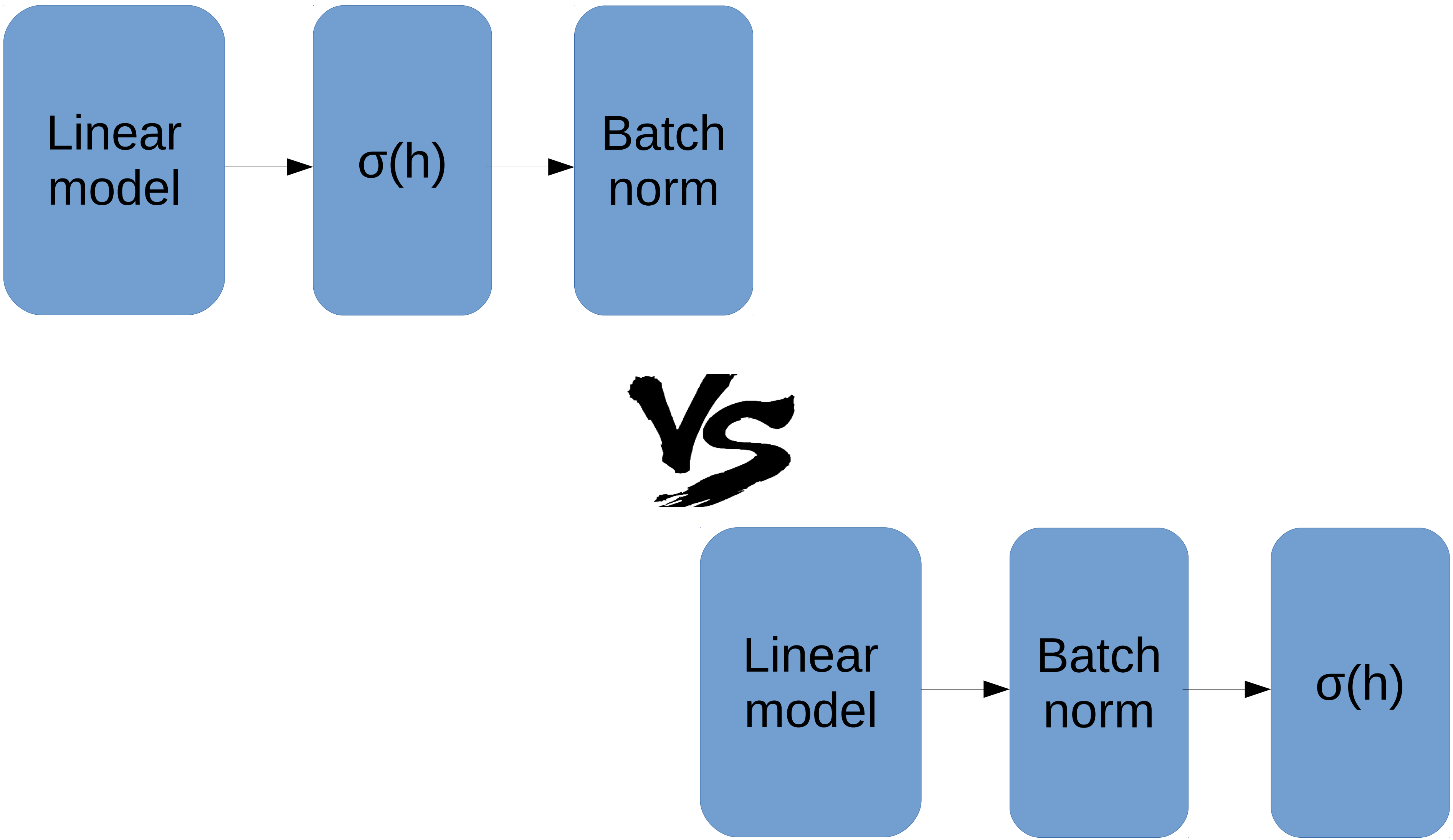
i stands for i-th neuron

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

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

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

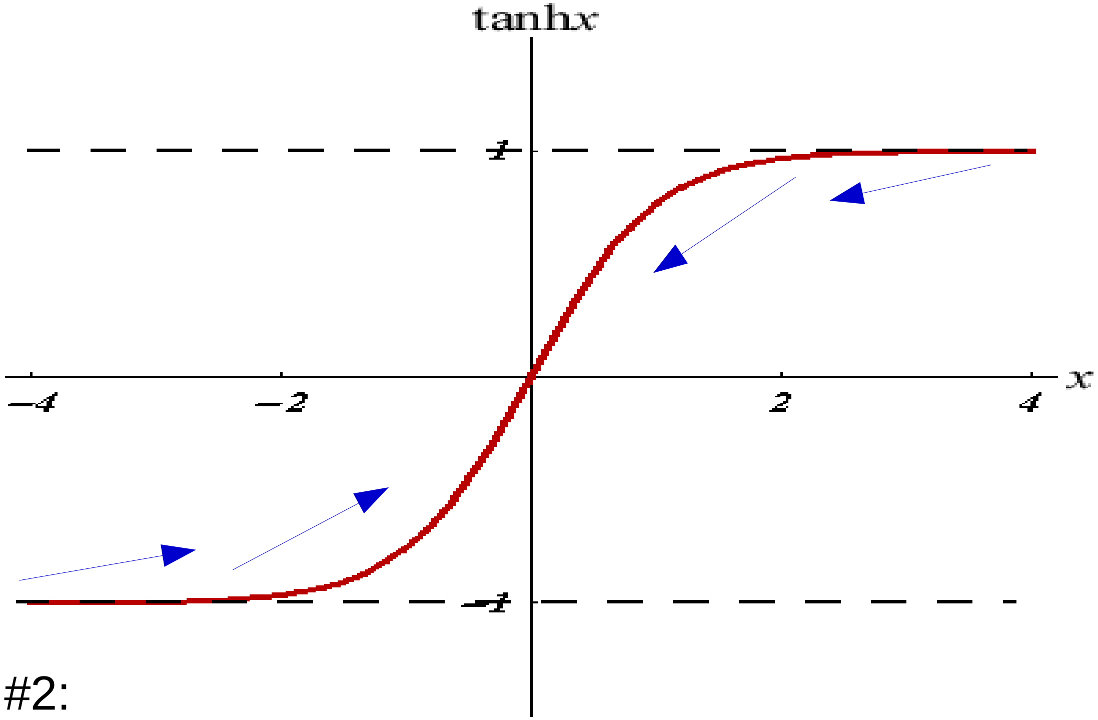
Batch normalization



Batch normalization

Good side effect #1:

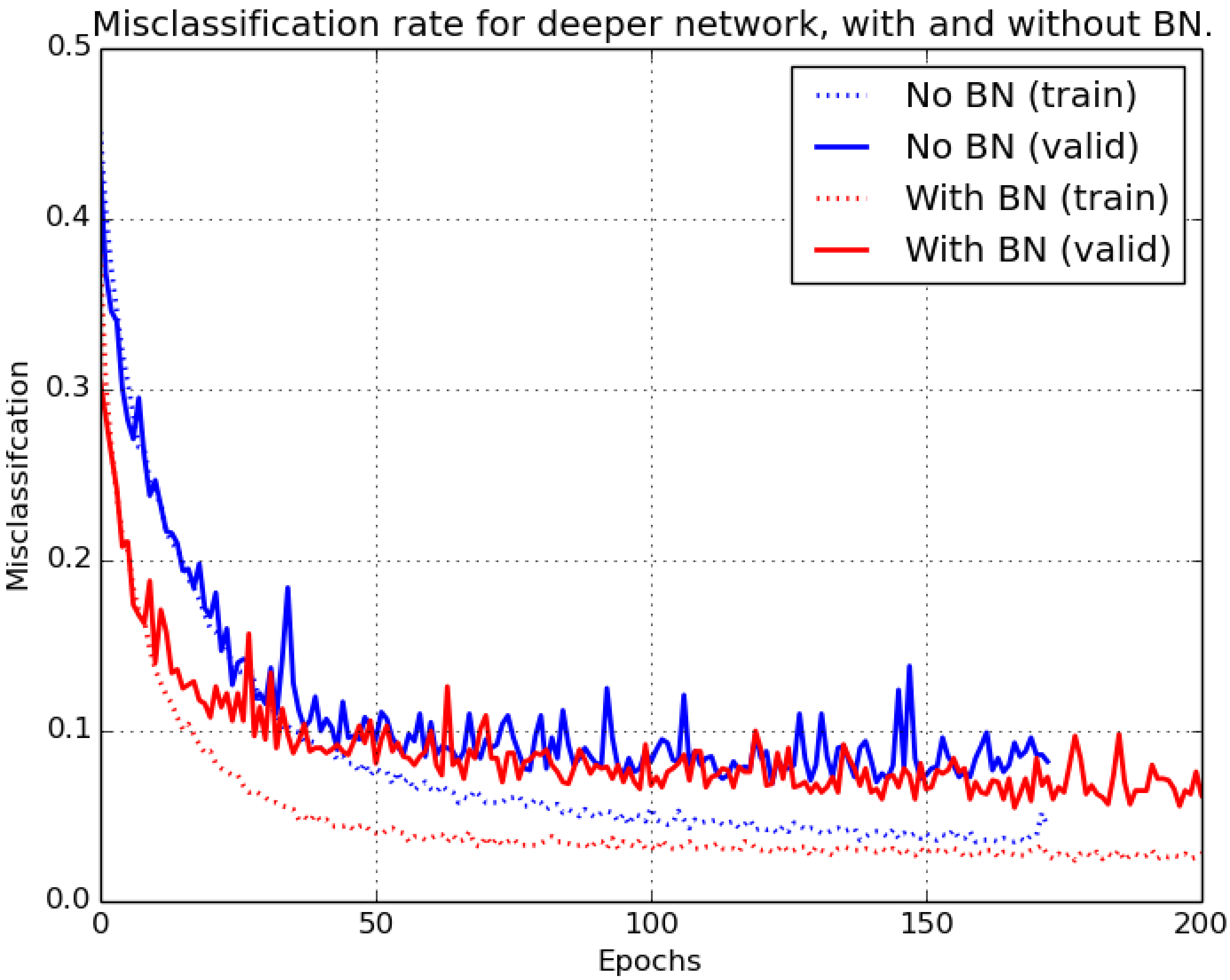
– Vanishing gradient less a problem for sigmoid-like nonlinearities



Good side effect #2:

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

Batch Normalization



More

Regularization:

– dropconnect, variational dropout, ...

Normalization:

– weight normalization,

– normalization propagation,

– layer normalization,

Initlialization:

– data-aware initialization, pre-training, ...

Nuff

Let's code some neural networks!