ML @ ICL

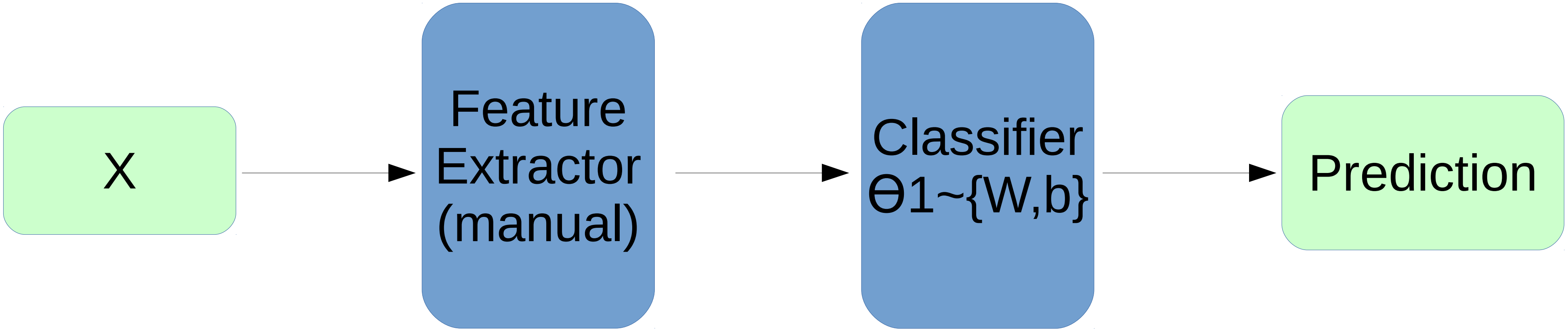
Episode 17

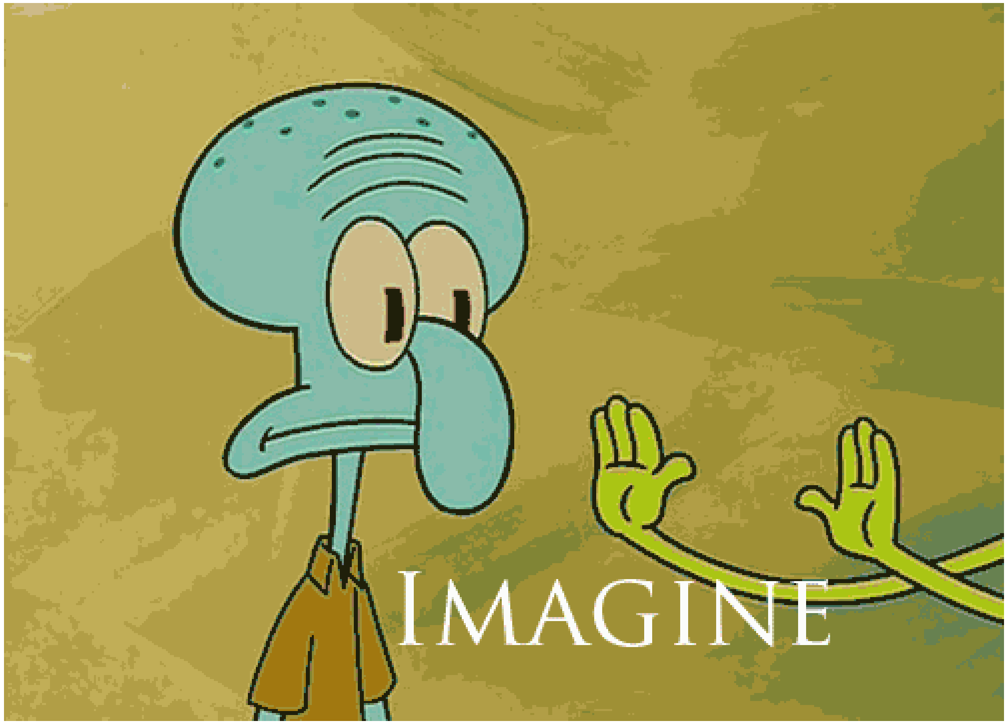
Deep learning whereabouts

A catch-all lecture in useless philosophy & practical tricks

Previously on deep learning...

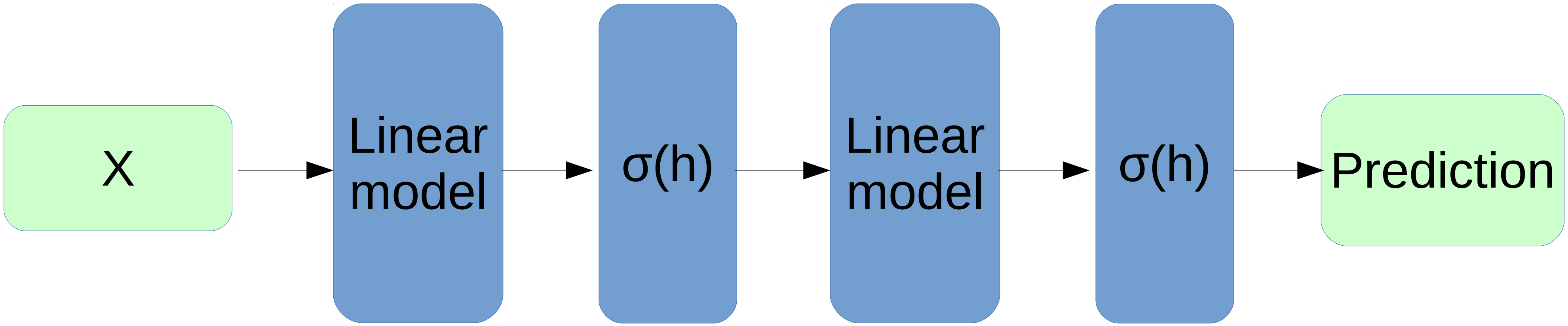
Feature extraction





Features would tune to your problem automatically!

Simple neural network



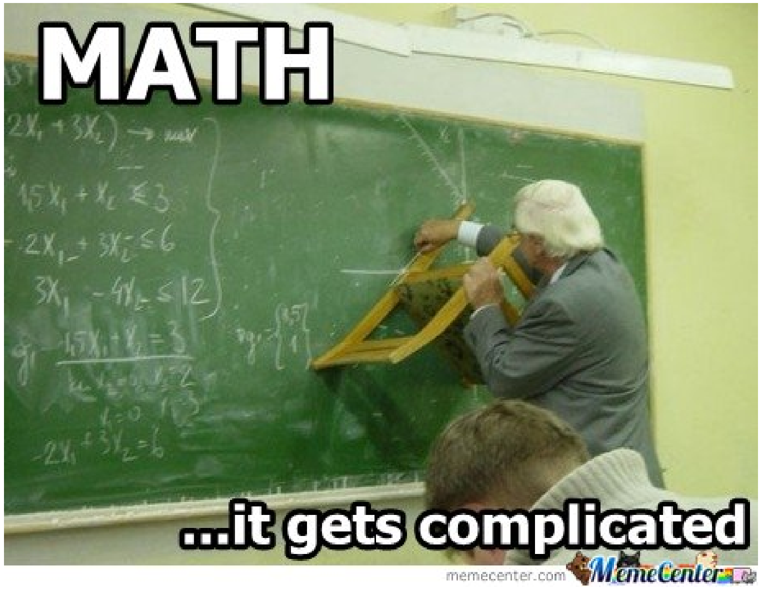
Trains with stochastic gradient descent!
or momentum/rmsprop/adam/...

Backpropagation

TL;DR: backprop = chain rule*

$$\frac{\partial f(g(x))}{\partial x} = \frac{\partial f(g(x))}{\partial g(x)} \cdot \frac{\partial g(x)}{\partial x}$$

* g and x can be vectors/vectors/tensors



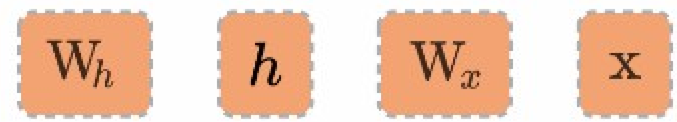
Dynamic graphs

Chainer, DyNet, Pytorch

A graph is created on the fly

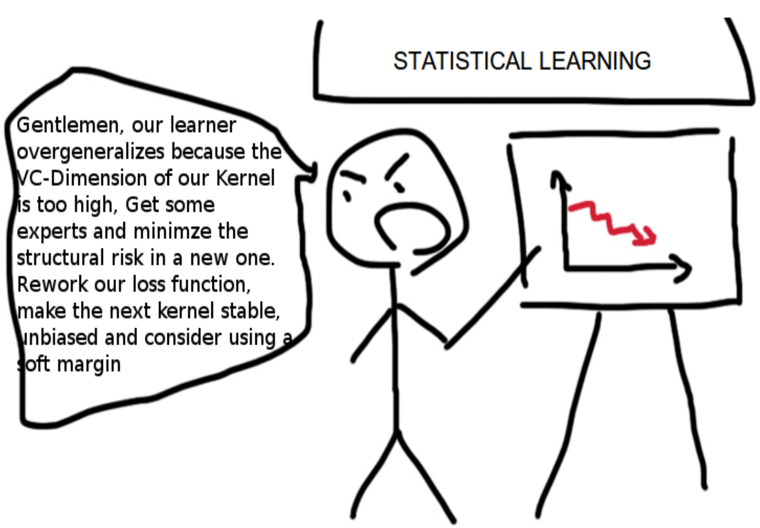
```
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))
```

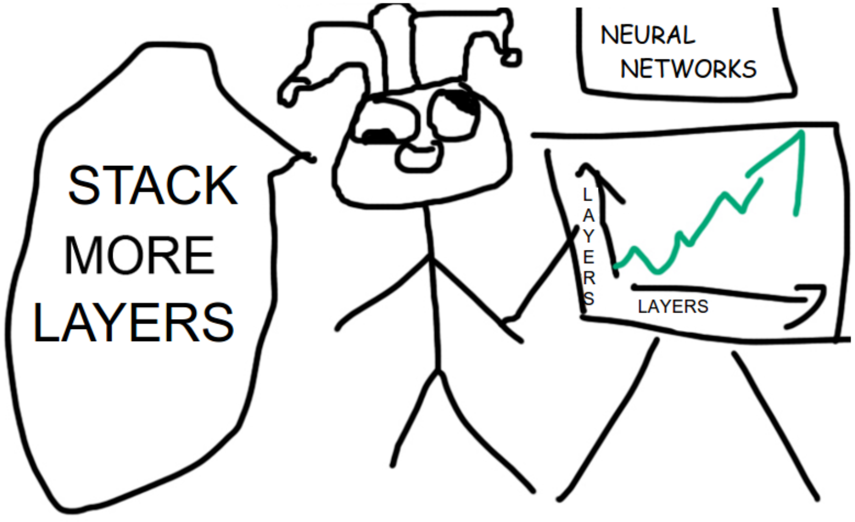


Not magic!

Don't expect deep learning to solve all your problems for free. For it won't.



vs



https://i.warosu.org/data/sci/img/0073/62/1435656449422.png

Not magic

Book of grudges

- No core theory
  - Relies on intuitive reasoning

Not magic

Book of grudges

- No core theory
  - Relies on intuitive reasoning

- Needs tons of data
  - You need either large dataset or heavy wizardry

Not magic

Book of grudges

- No core theory
  - Relies on intuitive reasoning

- Needs tons of data
  - You need either large dataset or heavy wizardry

- Computationally heavy
  - Running on mobiles/embedded is a challenge

Not magic

Book of grudges

- No core theory
  - Relies on intuitive reasoning
- Needs tons of data
  - You need either large dataset or heavy wizardry
- Computationally heavy
  - Running on mobiles/embedded is a challenge
- Pathologically overhyped
  - People expect of it to make wonders

Deep learning is a language

Deep learning is a language

in which you can hint your model
on what you want it to learn

Deep learning is a language

Say, you train classifier on two sets of features

Deep learning is a language

Say, you train classifier on two sets of features

Raw features

Car photo (image pixels)

High-level features

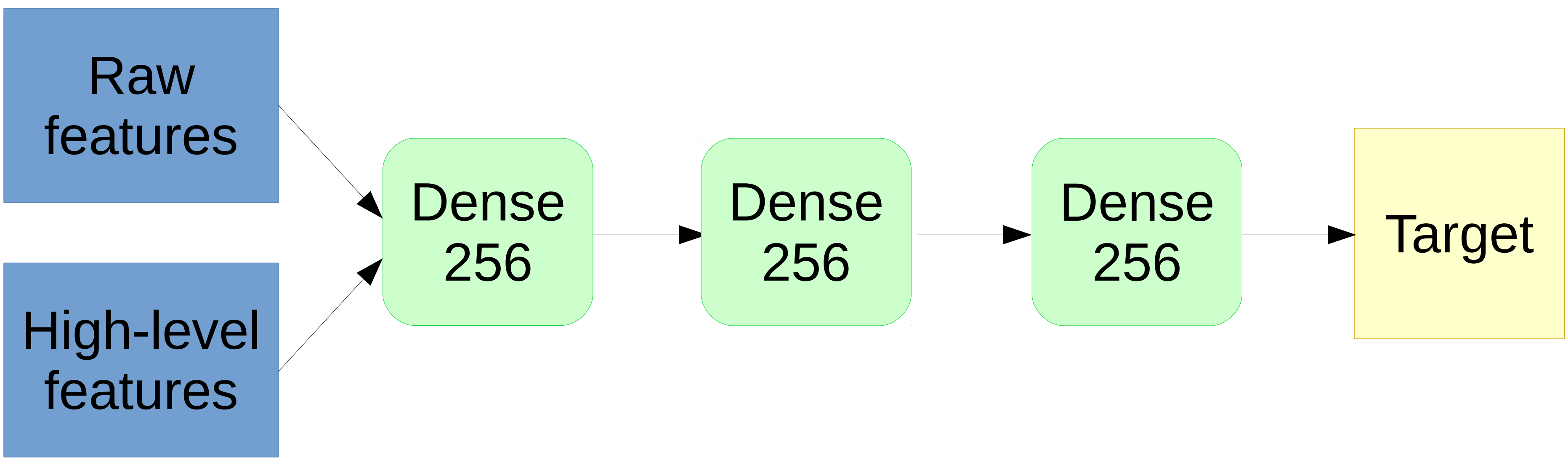
Car brand, model, age, blemishes

Car price

Target

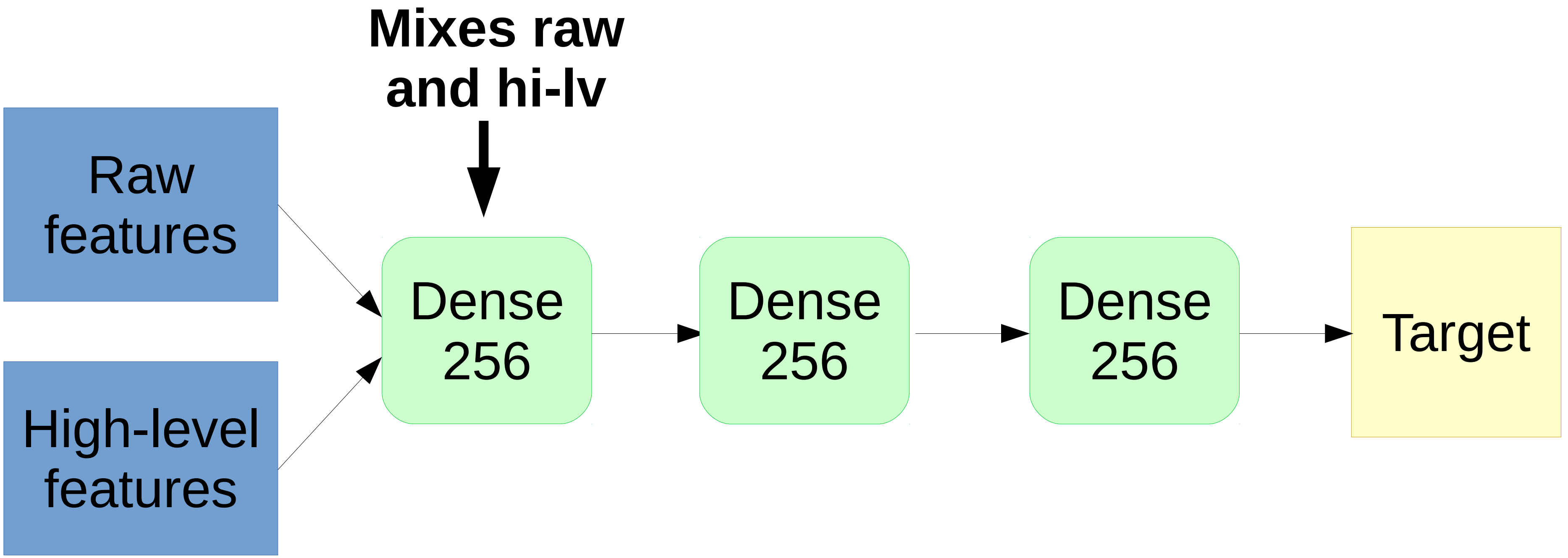
Deep learning is a language

Naive approach



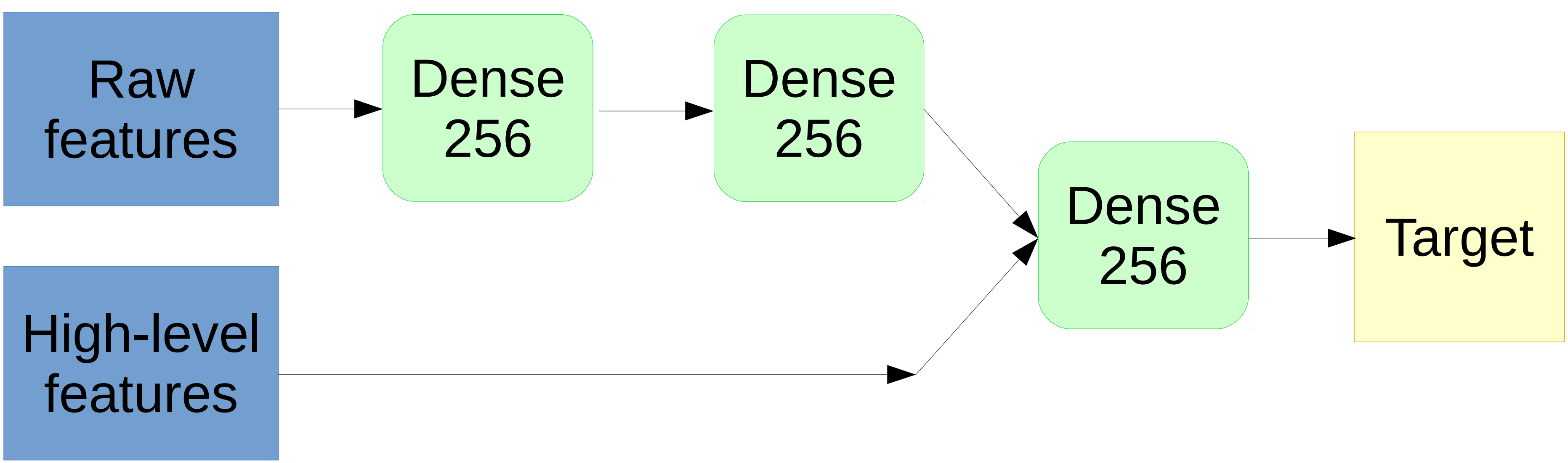
Deep learning is a language

Naive approach



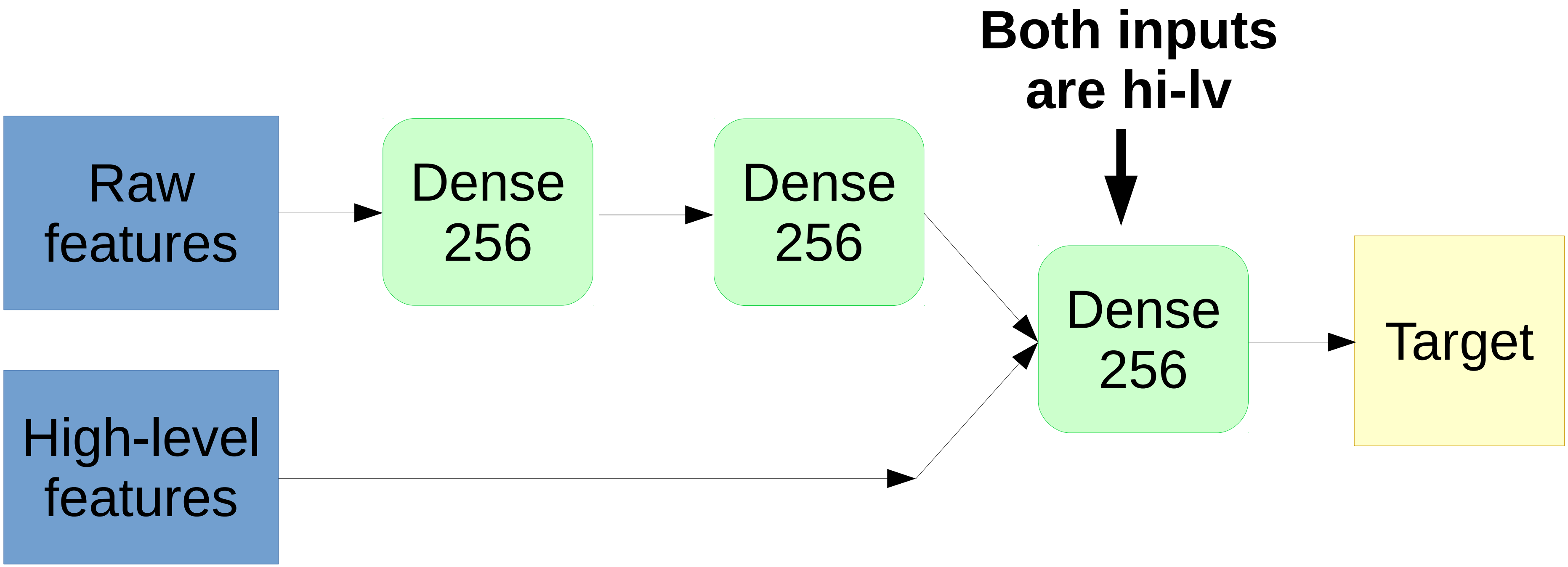
Deep learning is a language

Less naïve approach



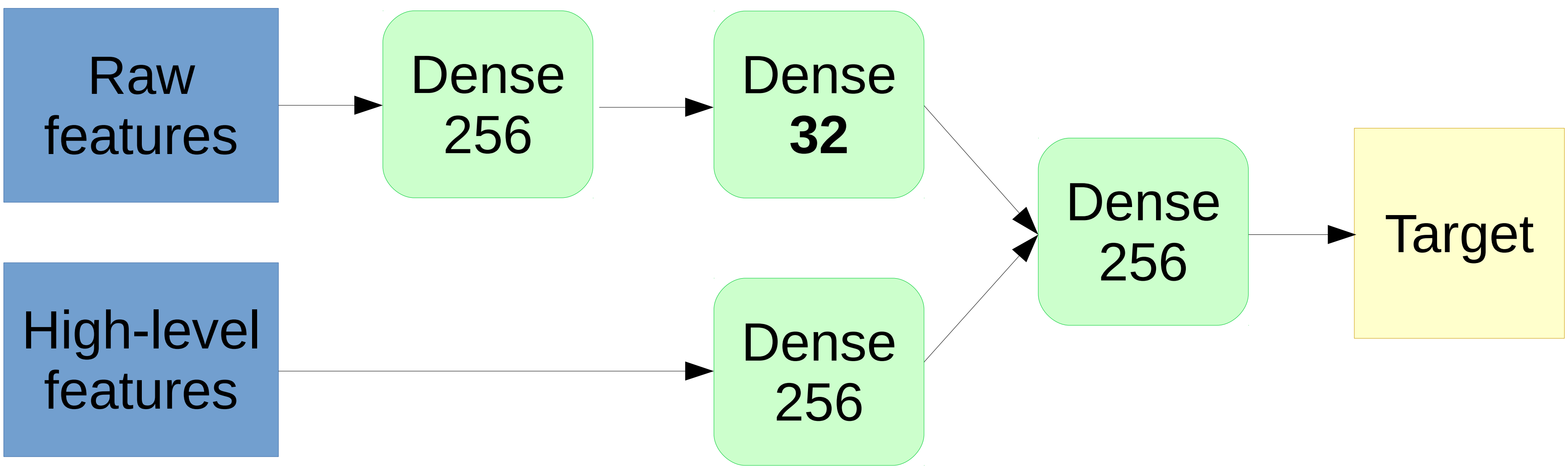
Deep learning is a language

Less naïve approach



Deep learning is a language

“Image features should be less important”
if that's what you want to say

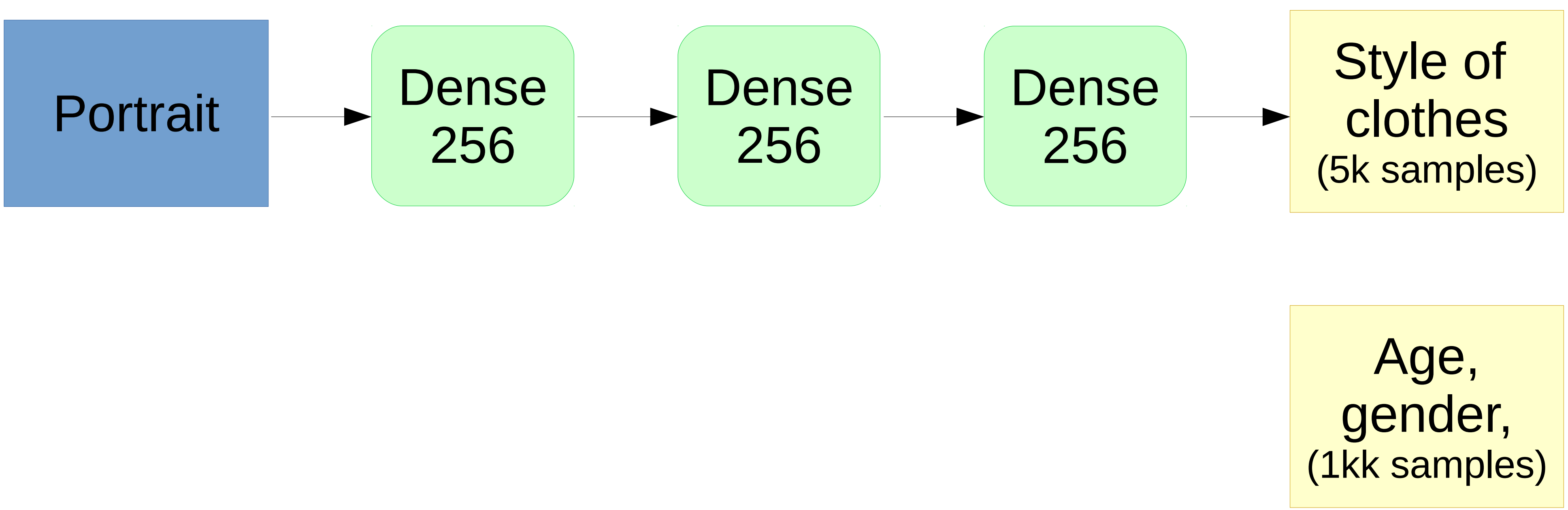


Deep learning is a language

You have a small dataset

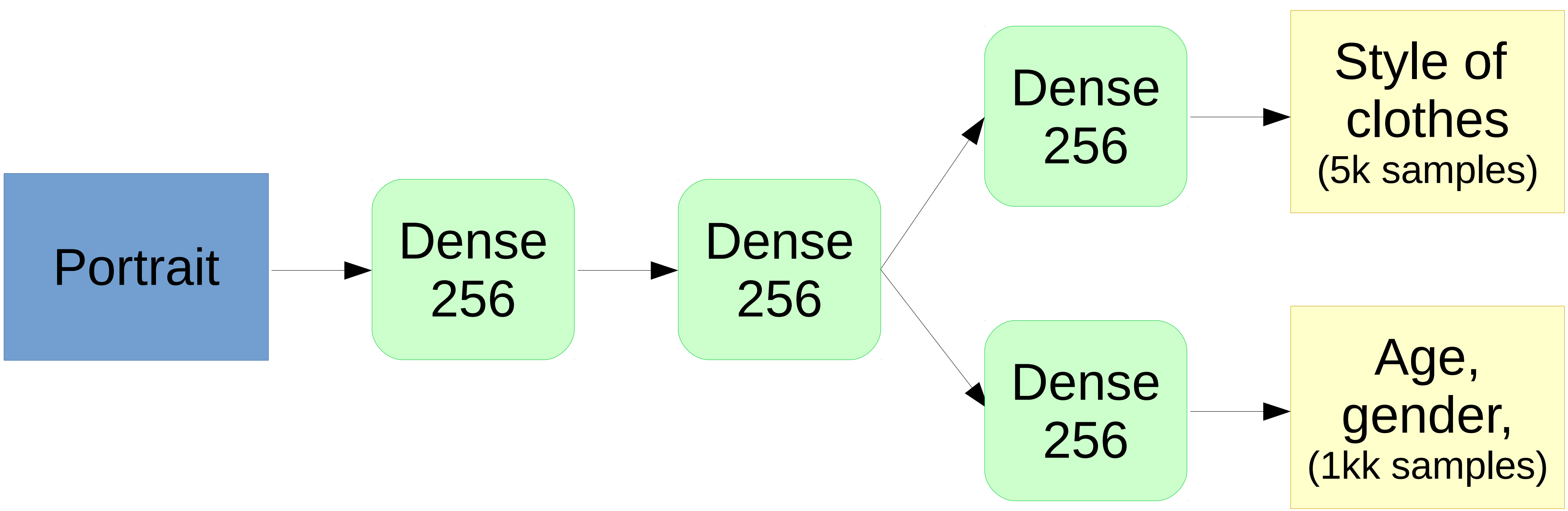
Deep learning is a language

You have a small dataset
and a larger dataset with similar task



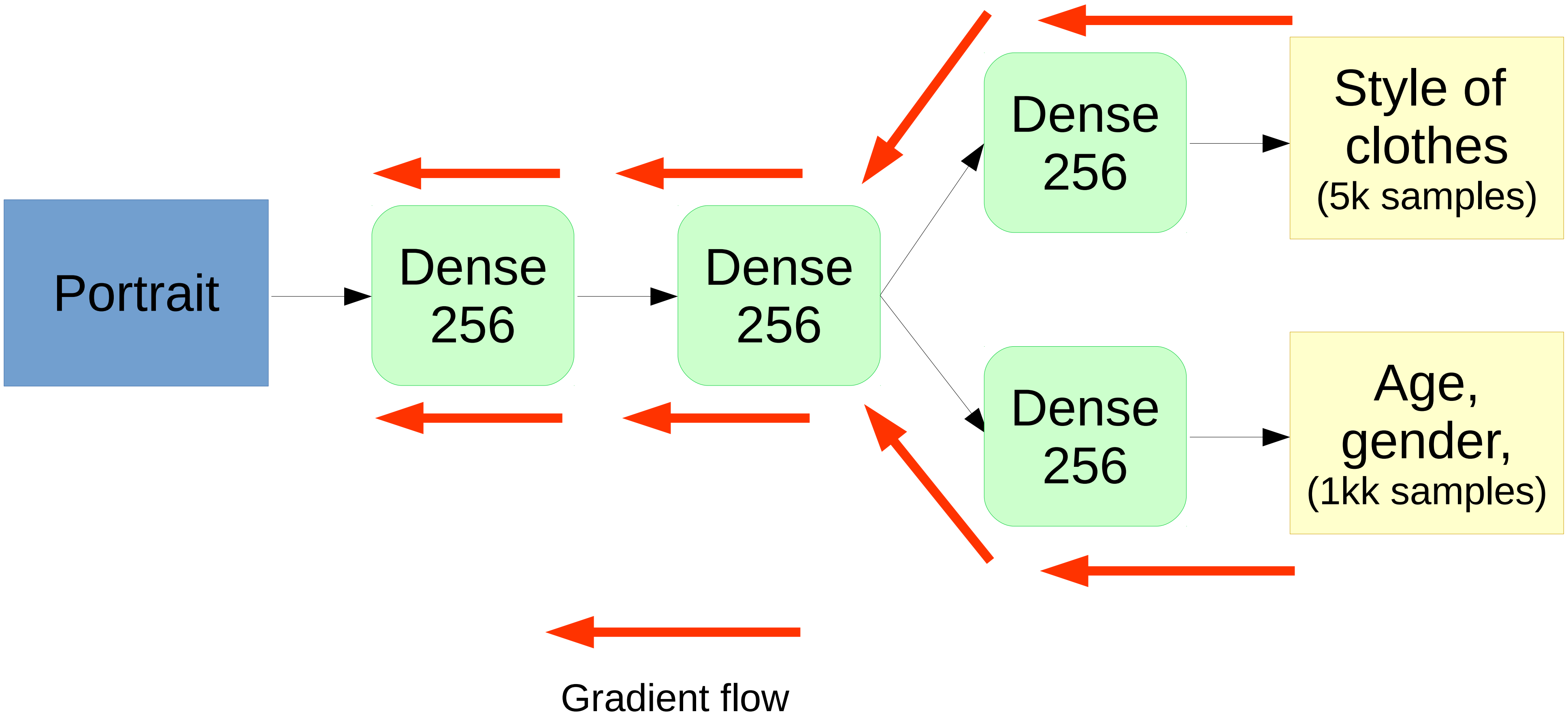
Deep learning is a language

You have a small dataset
and a larger dataset with similar task



Deep learning is a language

I want to learn features for style classification that also help determine age & gender



Deep learning is a language

For images:
- “I want to classify cats regardless where they are”
- “I don't want model to be indifferent to small shifts”

For texts:
- “Model should reconstruct the underlying process”

In general:
- “I don't want model to trust single feature too much”
- “I want my features to be sparse”

Let's see a few more “words”

Regularization

- Neural networks overfit like nothing else.

Gotta regularize!

- We can use L1/L2 like usual, but there's more!

Regularization

- Dropout:

“I don't my network to trust
any single neuron too much”

- Idea:

At training time, with probability **p**

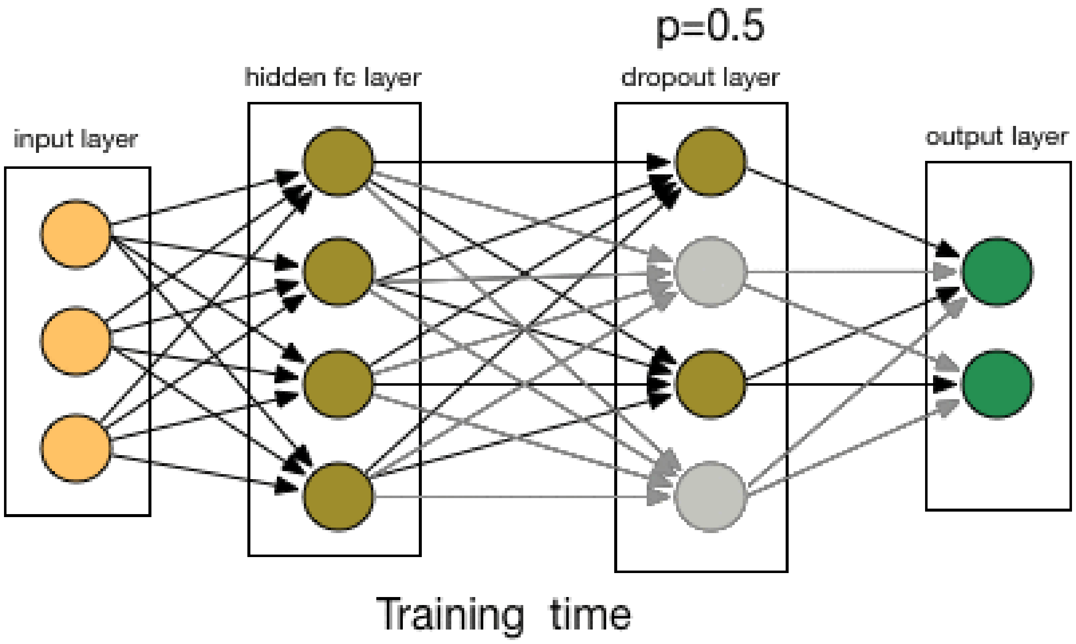
multiply neurons by zero!

- Scale up the remaining neurons to keep average the same

Regularization

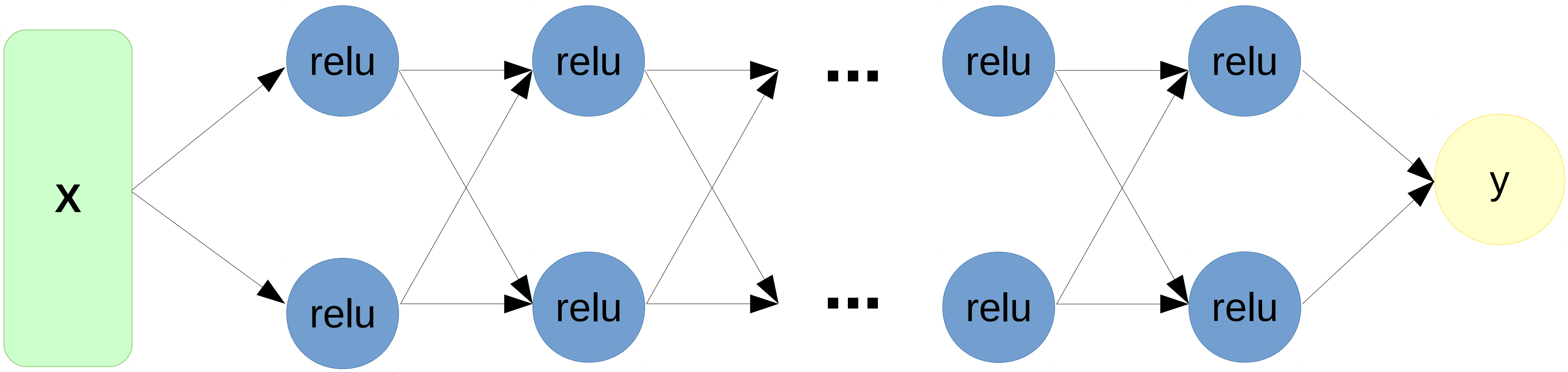
- Dropout:

"I don't my network to trust any single neuron too much"



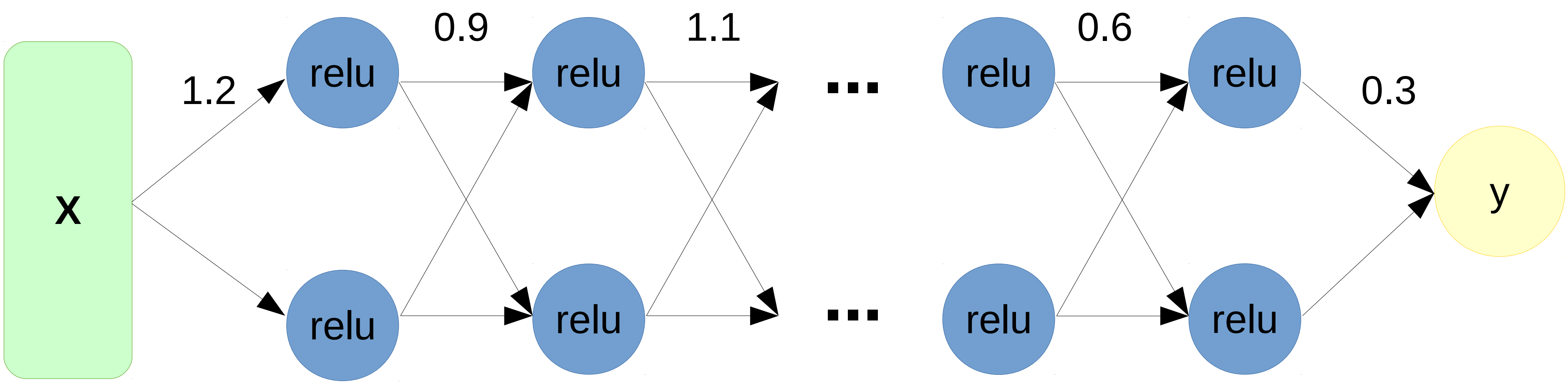
The problem with deep networks

- Imagine a 100-layer network with ReLU



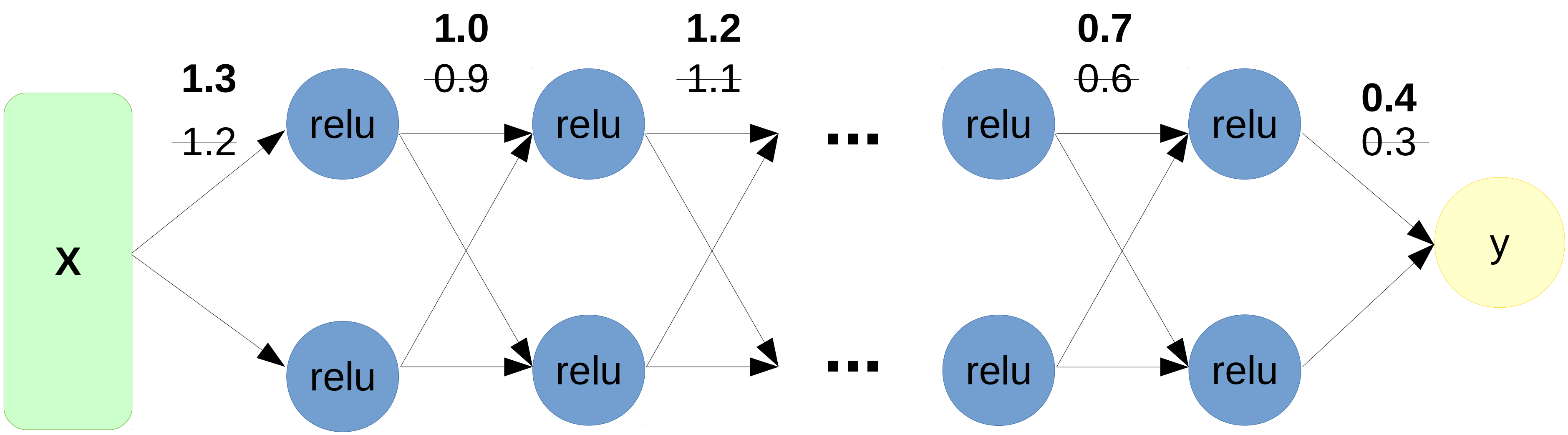
The problem with deep networks

- Imagine a 100-layer network with ReLU



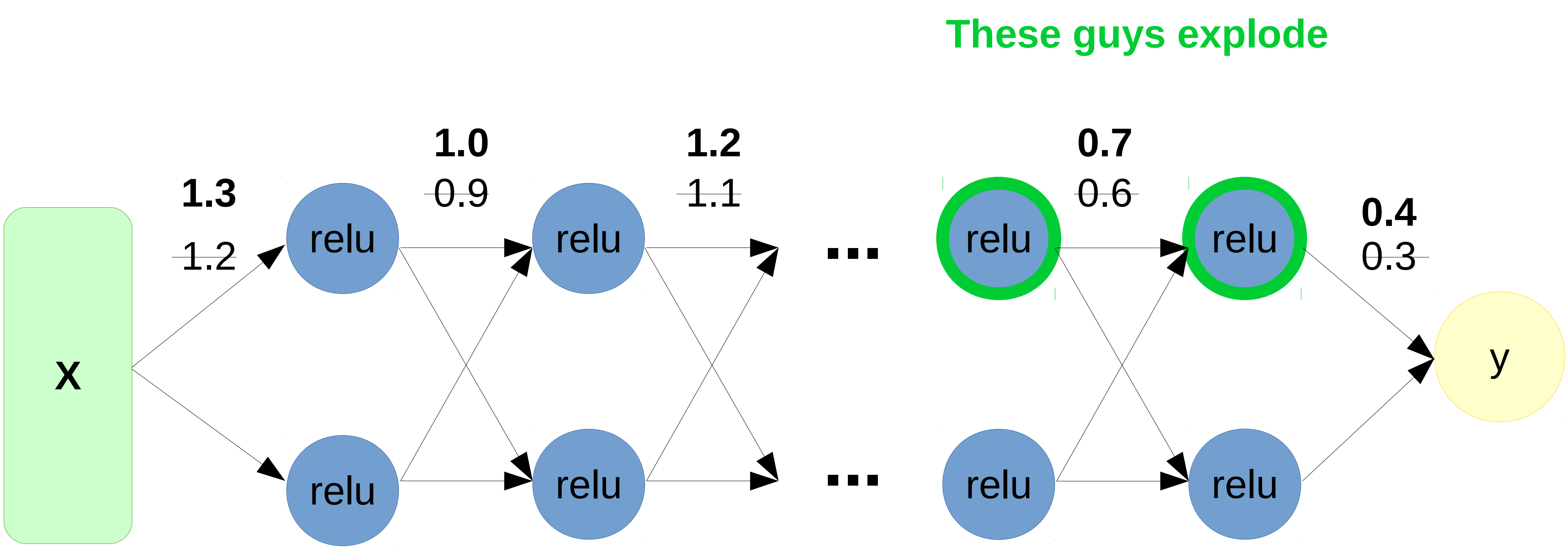
The problem with deep networks

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



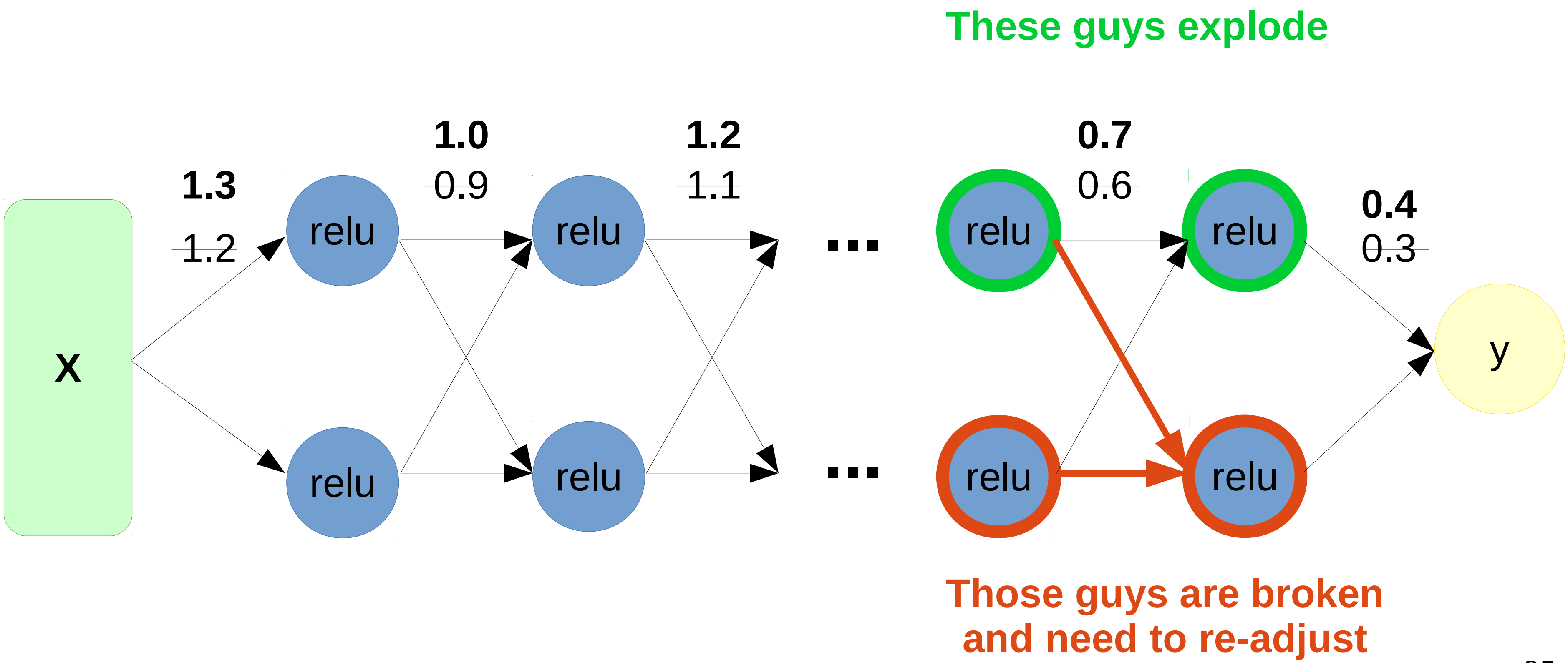
The problem with deep networks

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



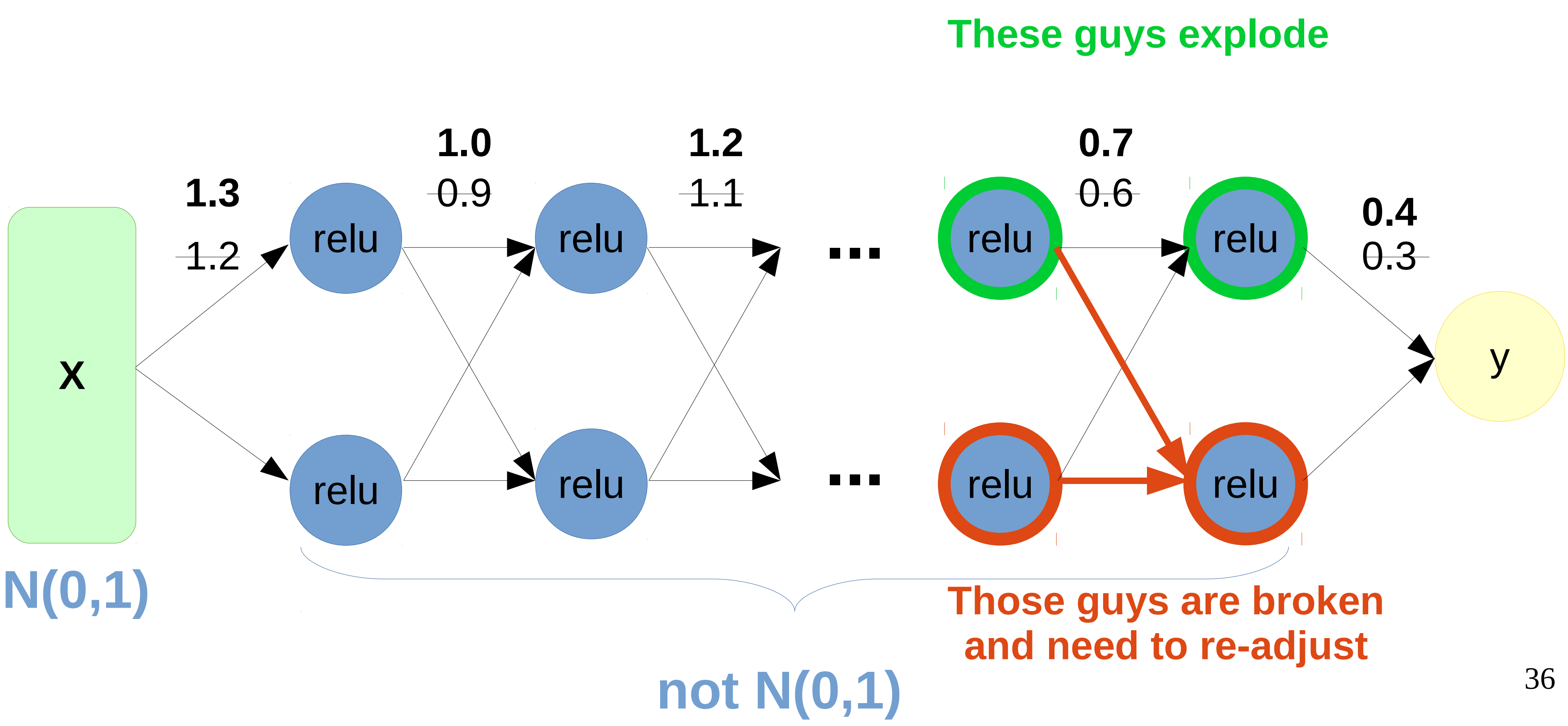
The problem with deep networks

- 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)

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

- 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)

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

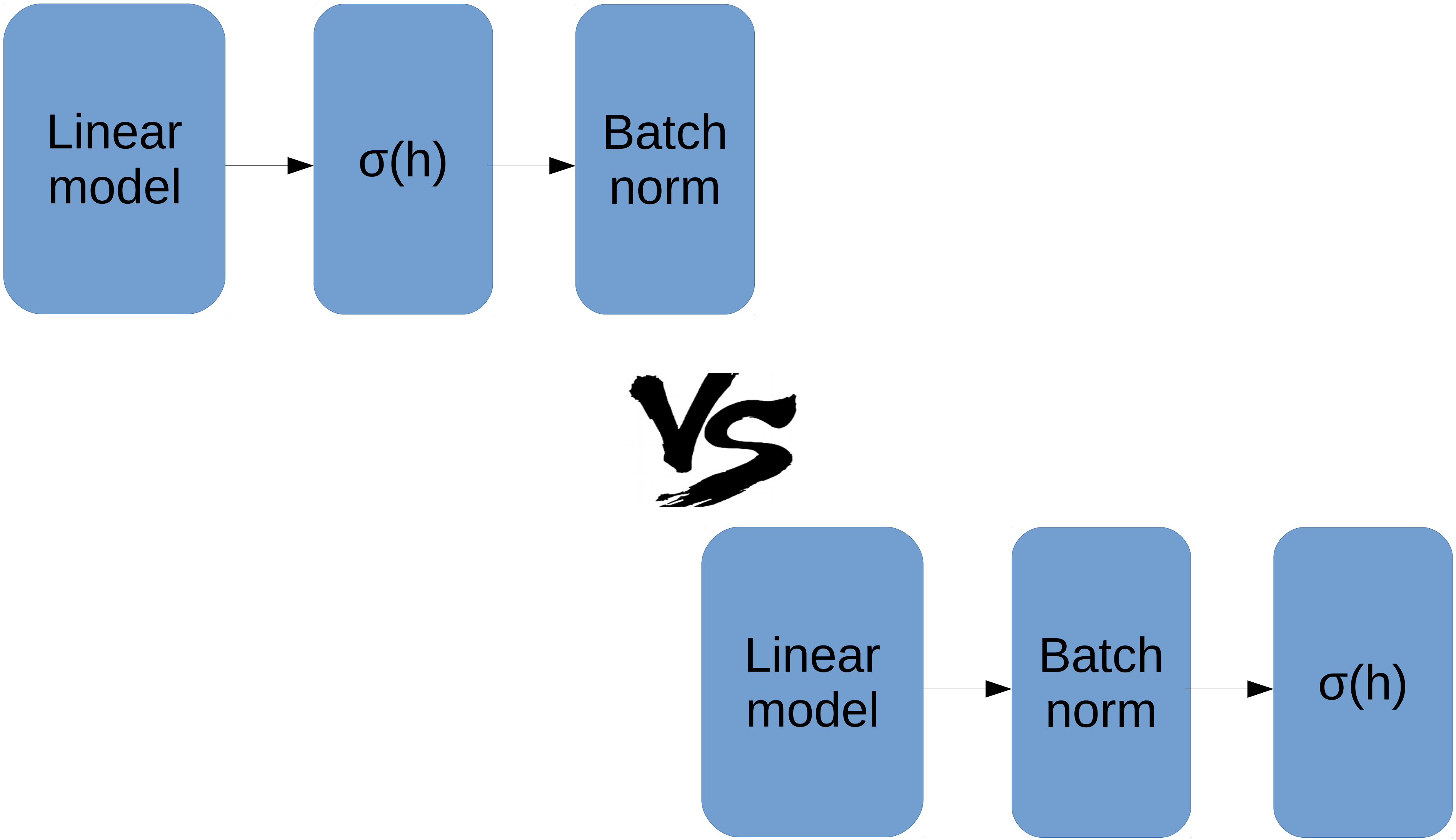
i stands for i-th neuron

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

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

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

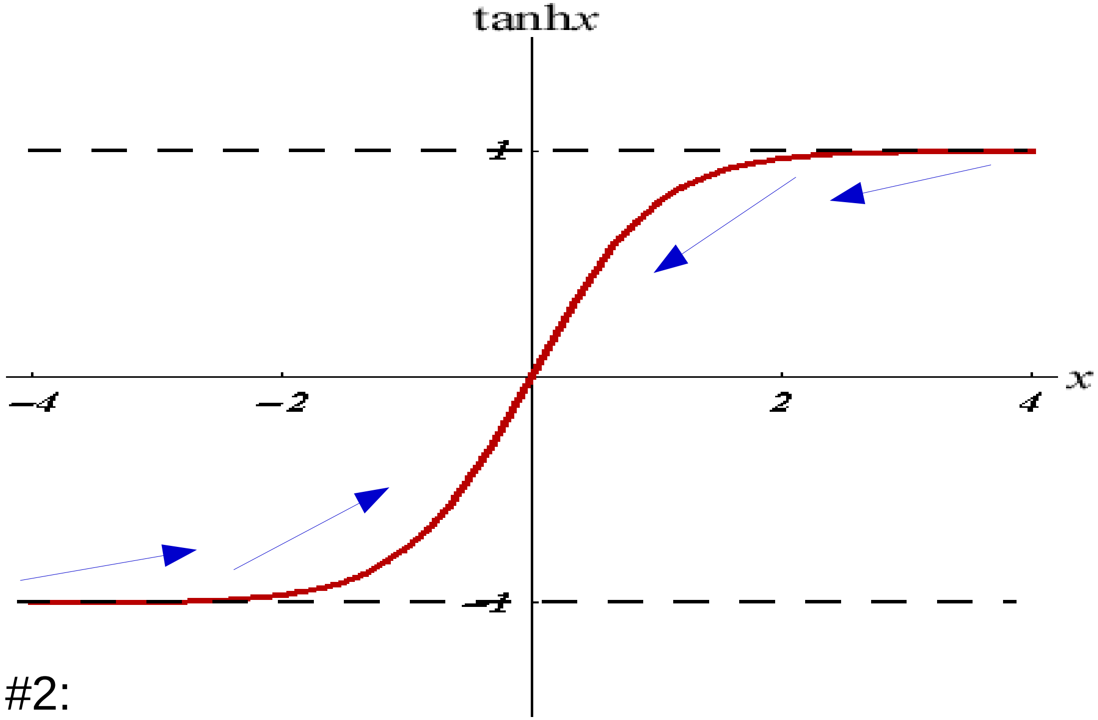
Batch normalization



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)

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

More normalization

Layer/Instance normalization

- Like batchnorm, but normalizes over different axes

Normprop

- A special training algorithm

Self-normalizing neural networks (SELU)

Nuff

Coding time!