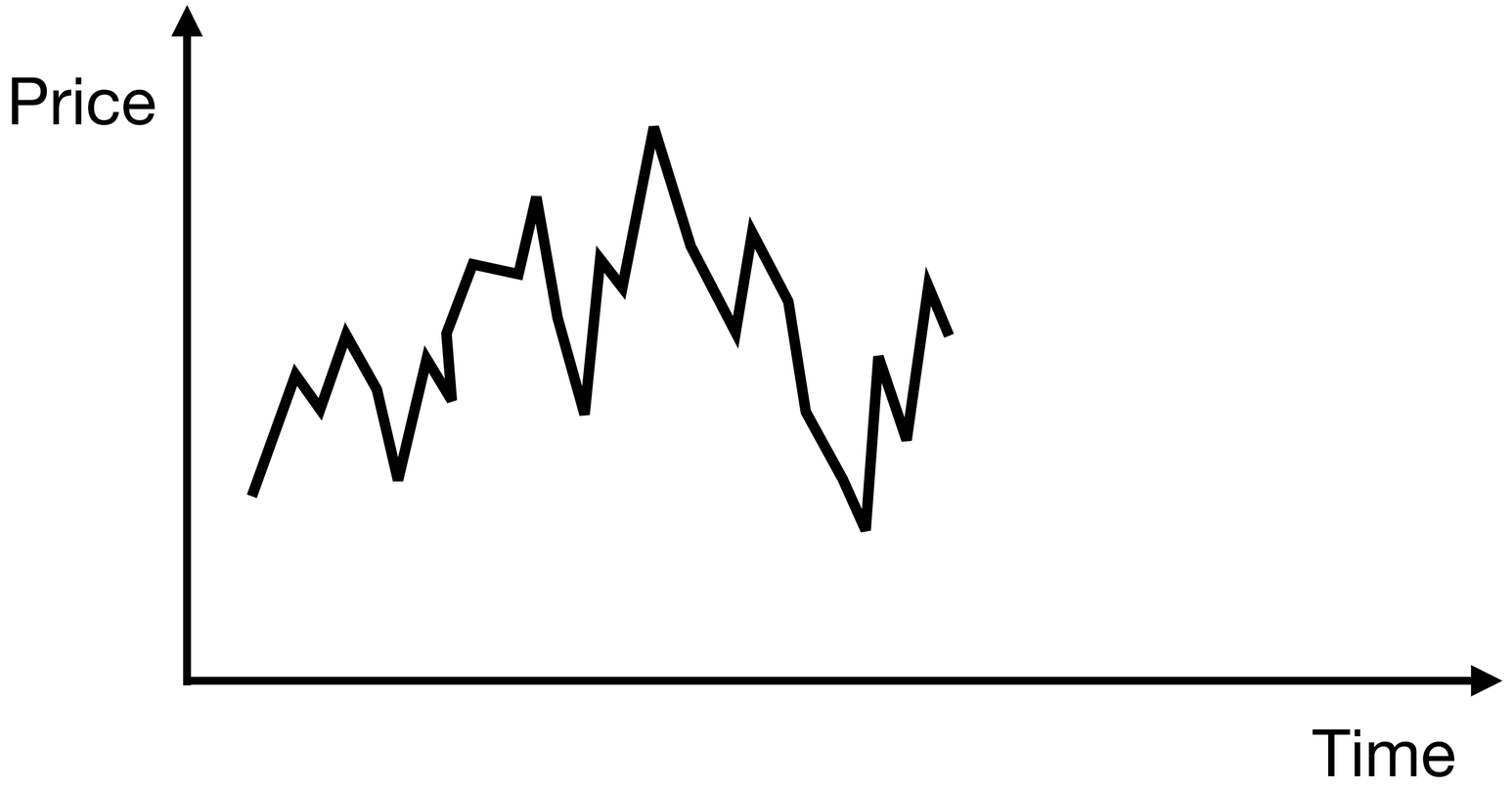
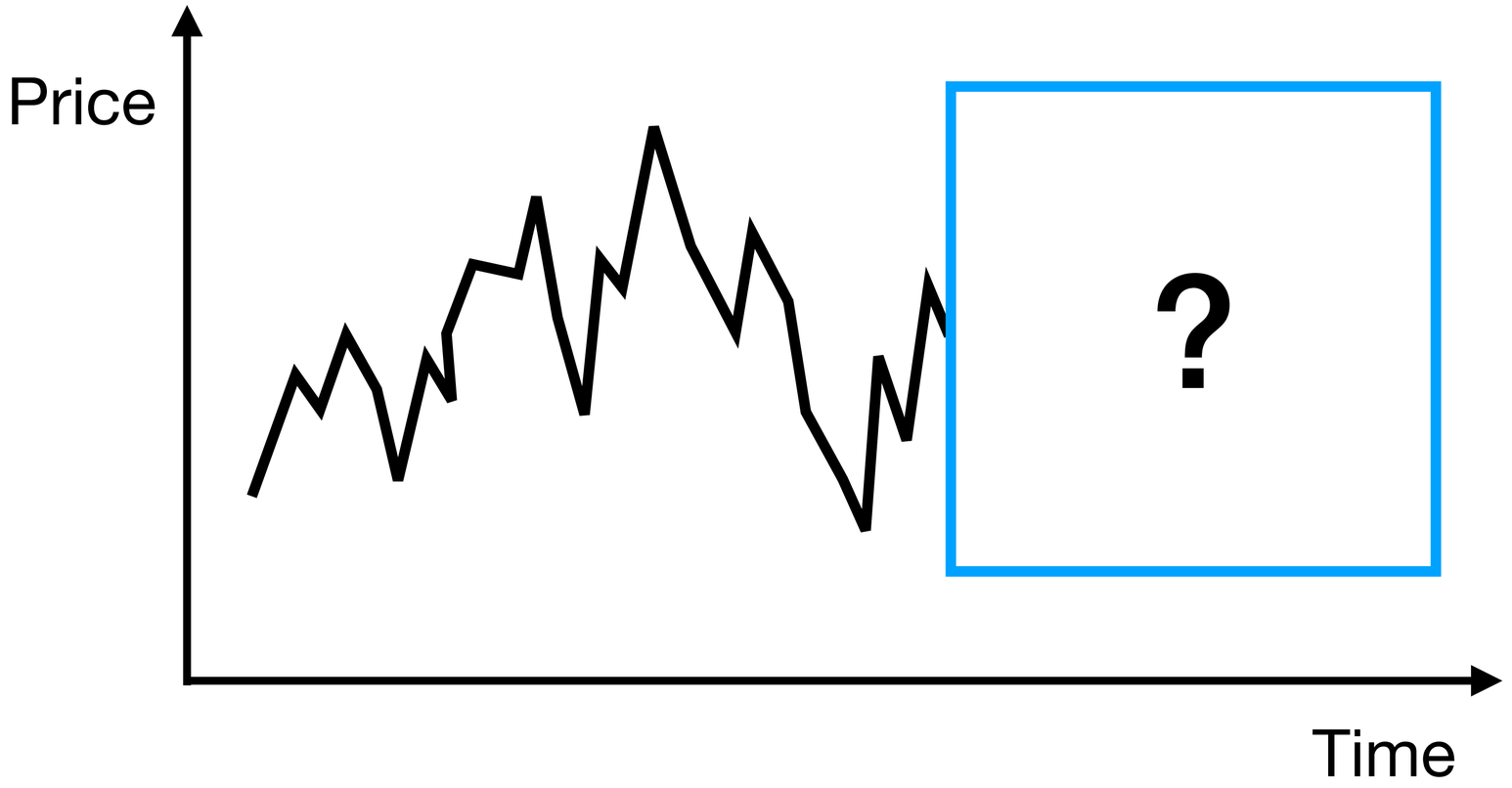
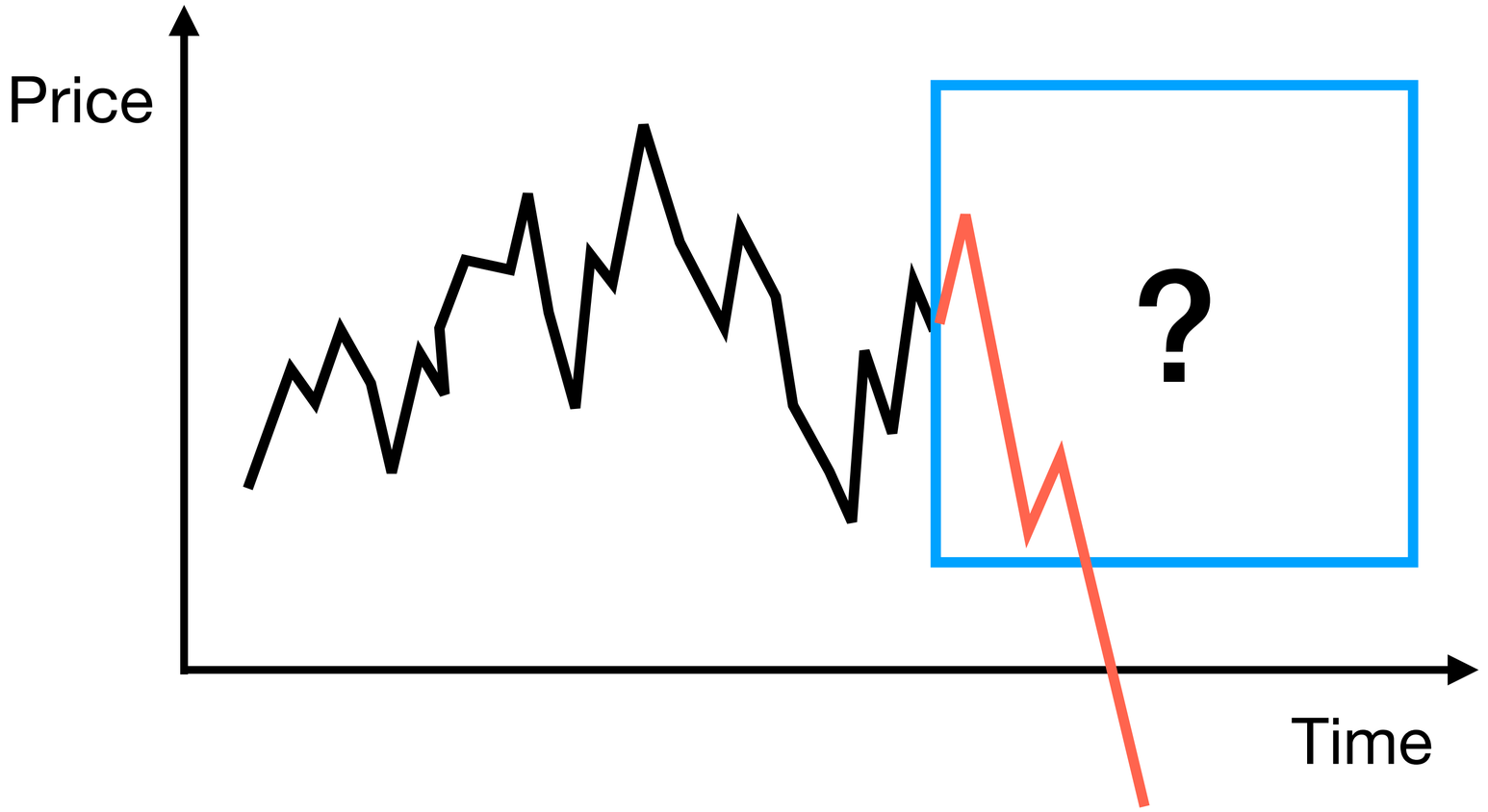
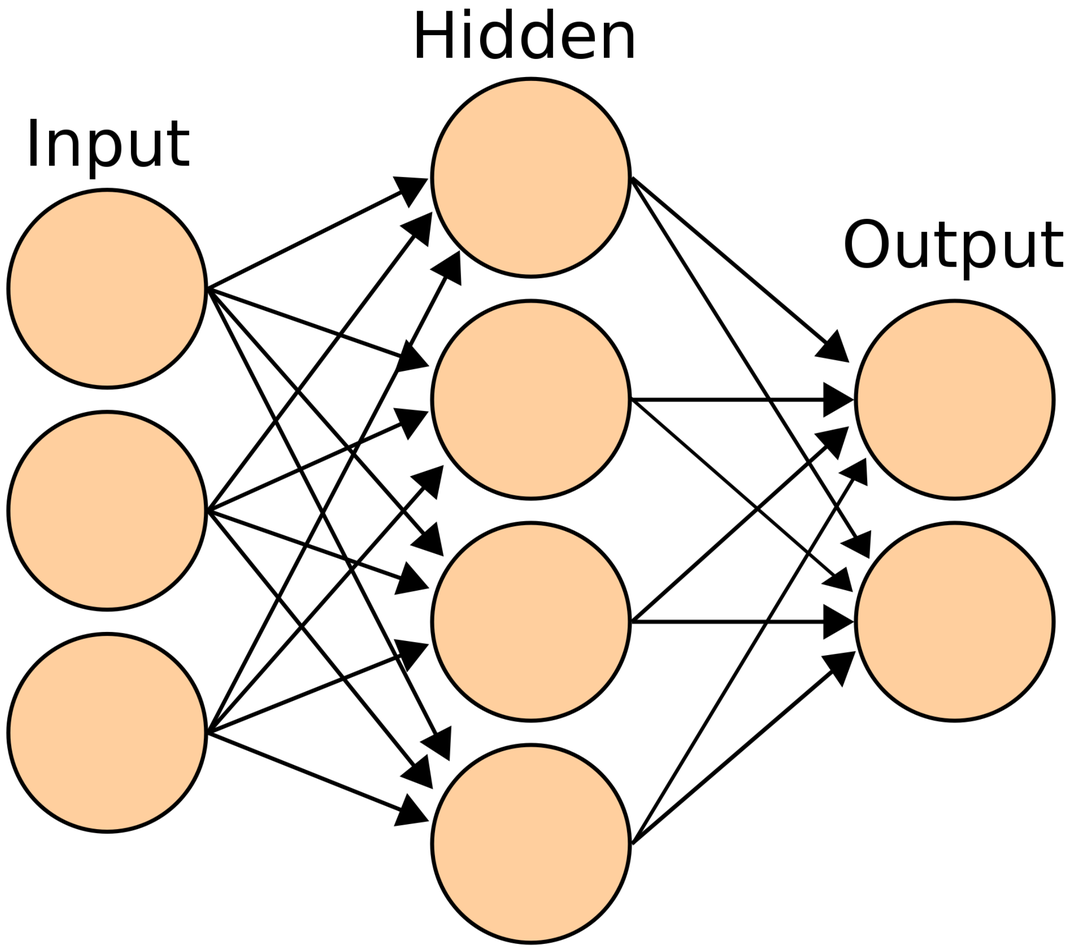
Understanding Deep Learning

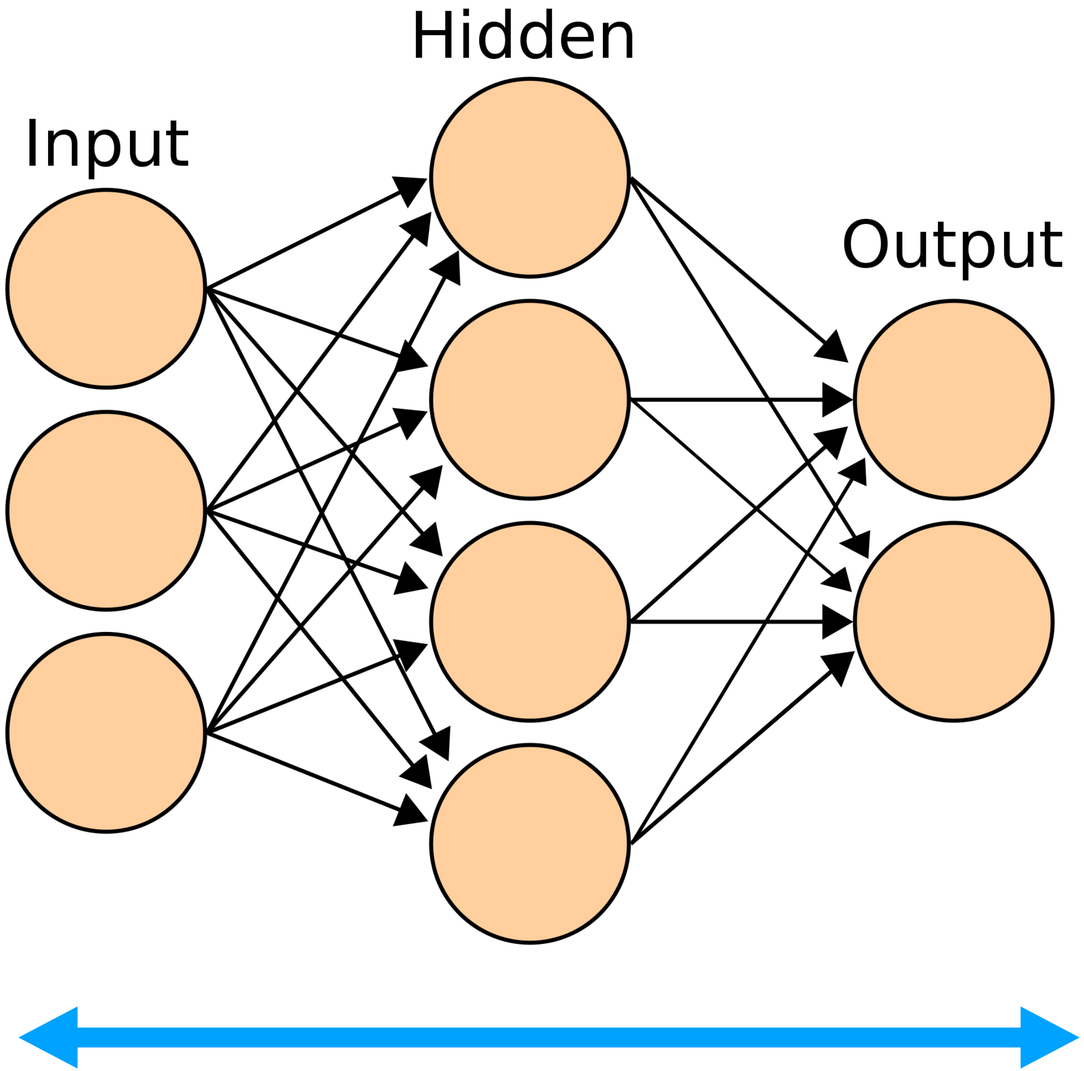
Jessica Yung

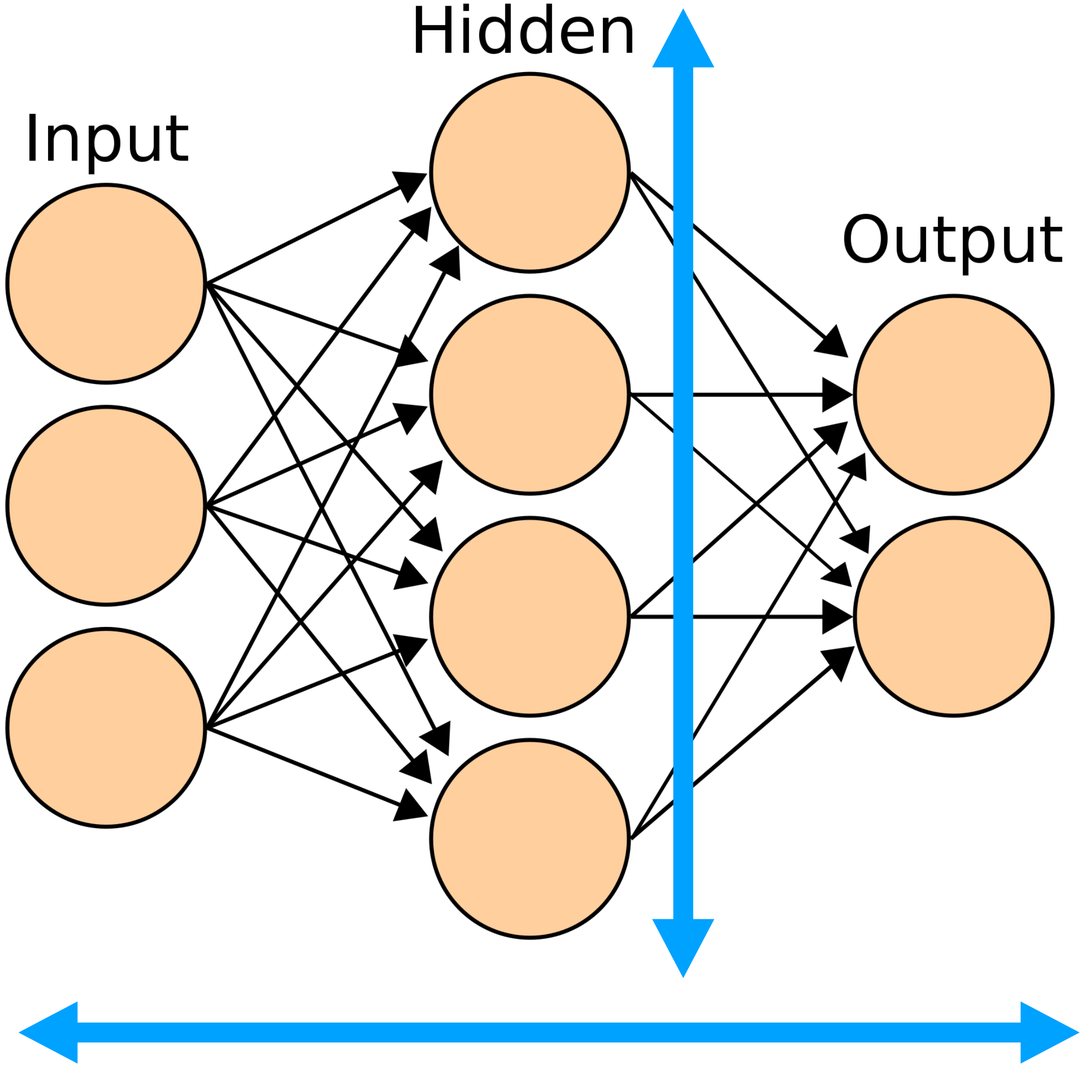

Price
Time

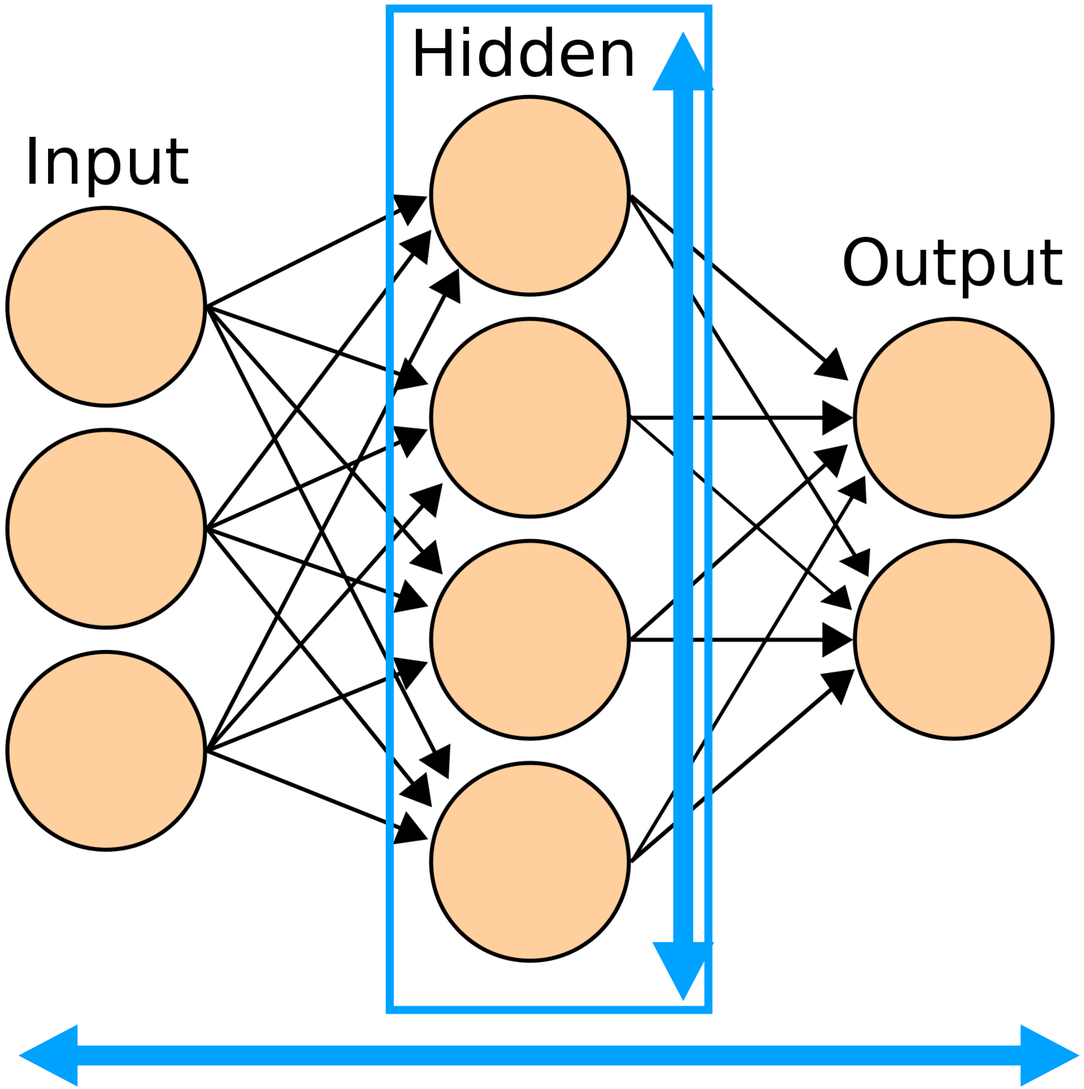

Price
?
Time

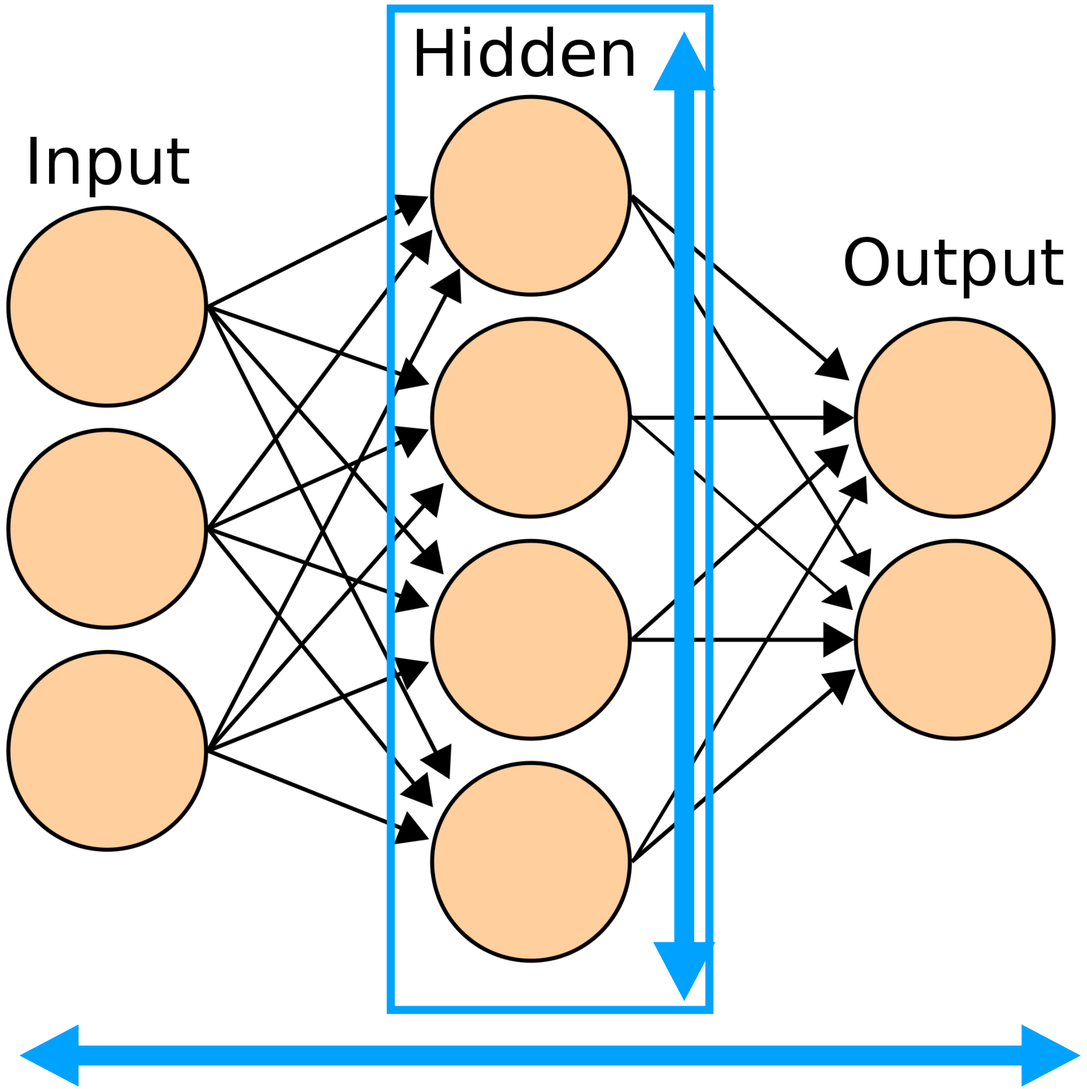

Price
?
Time


Hidden
Input
Output

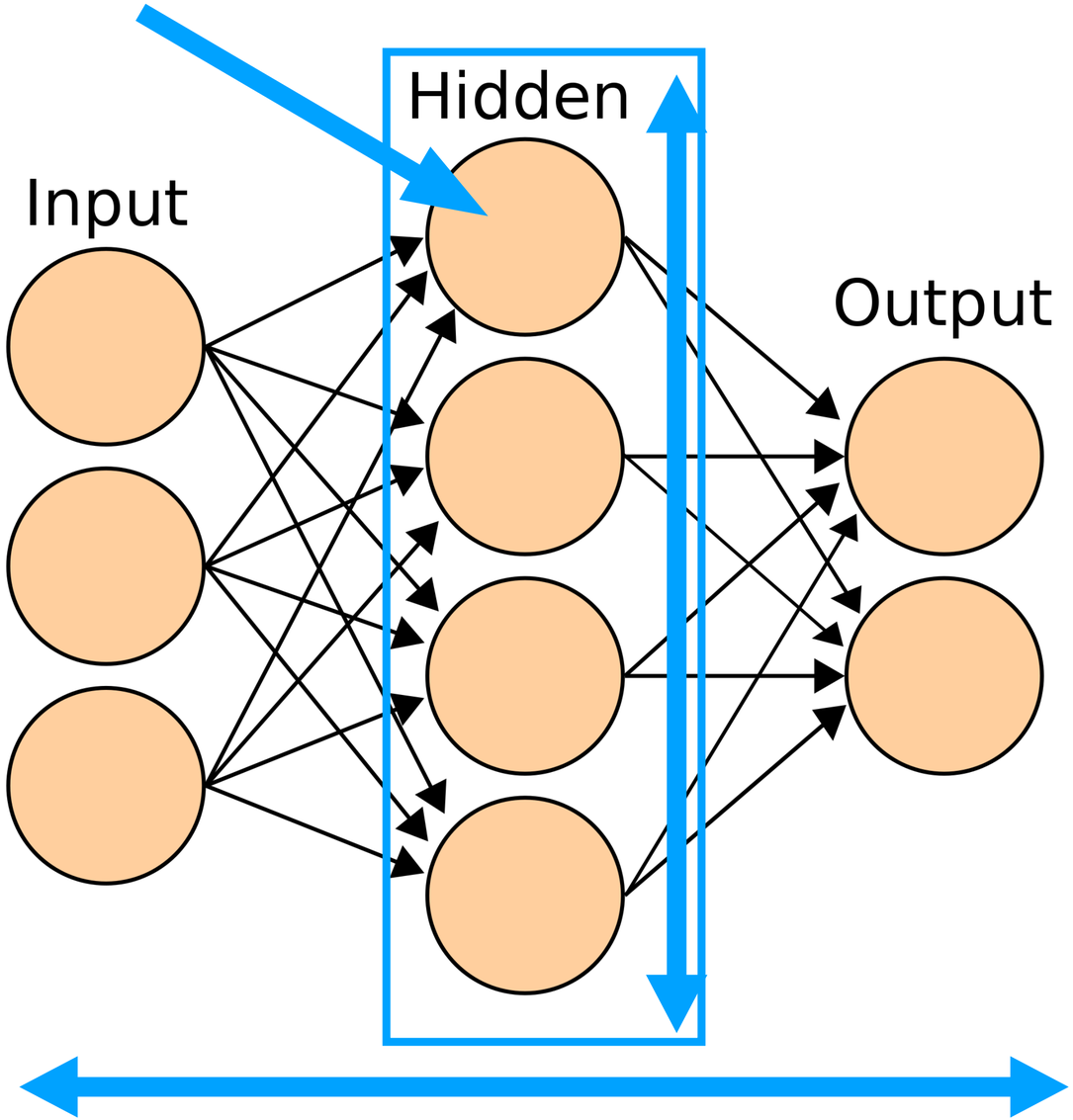

Hidden
Input
Output


Hidden
Input
Output

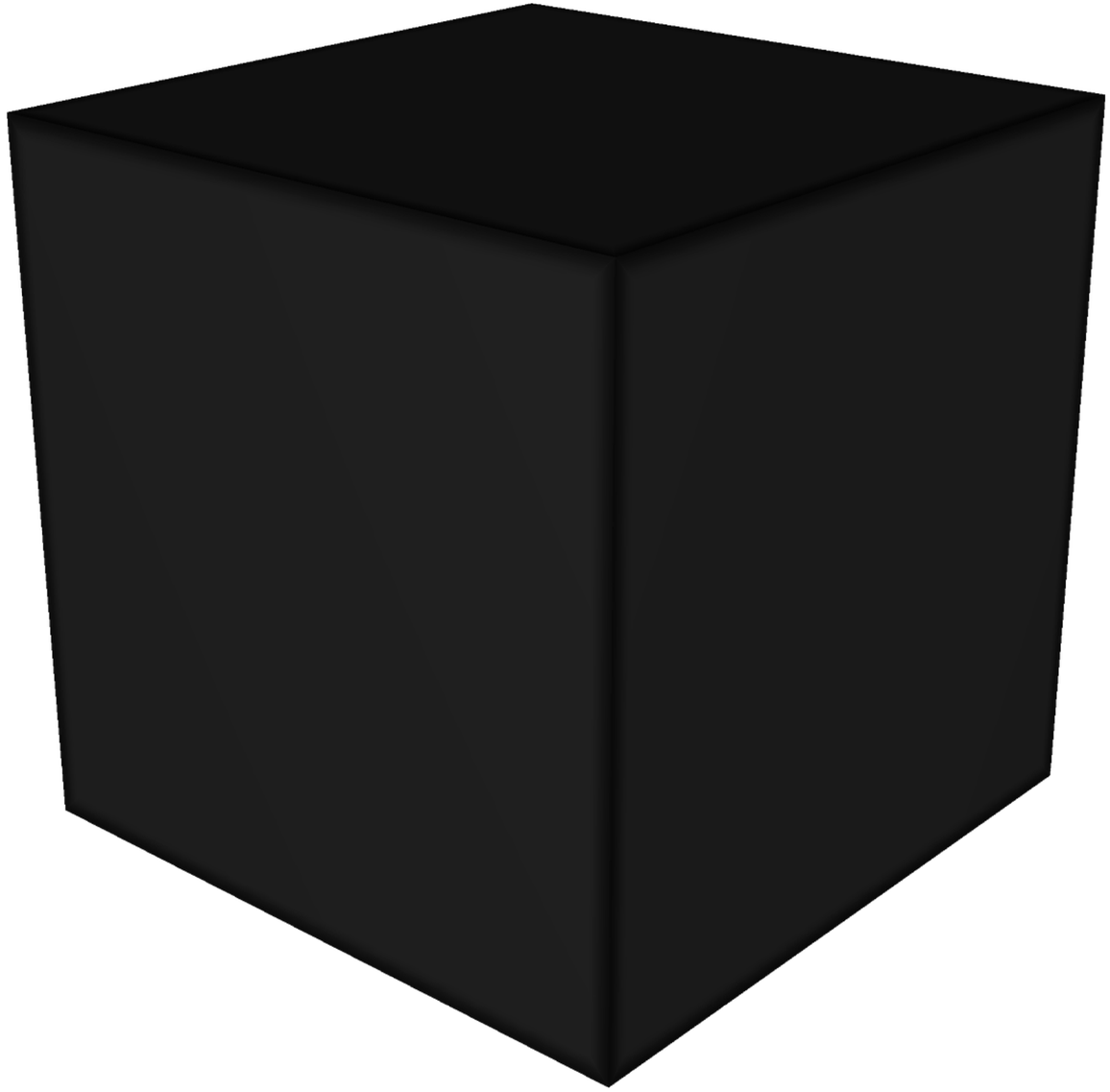

Hidden
Input
Output

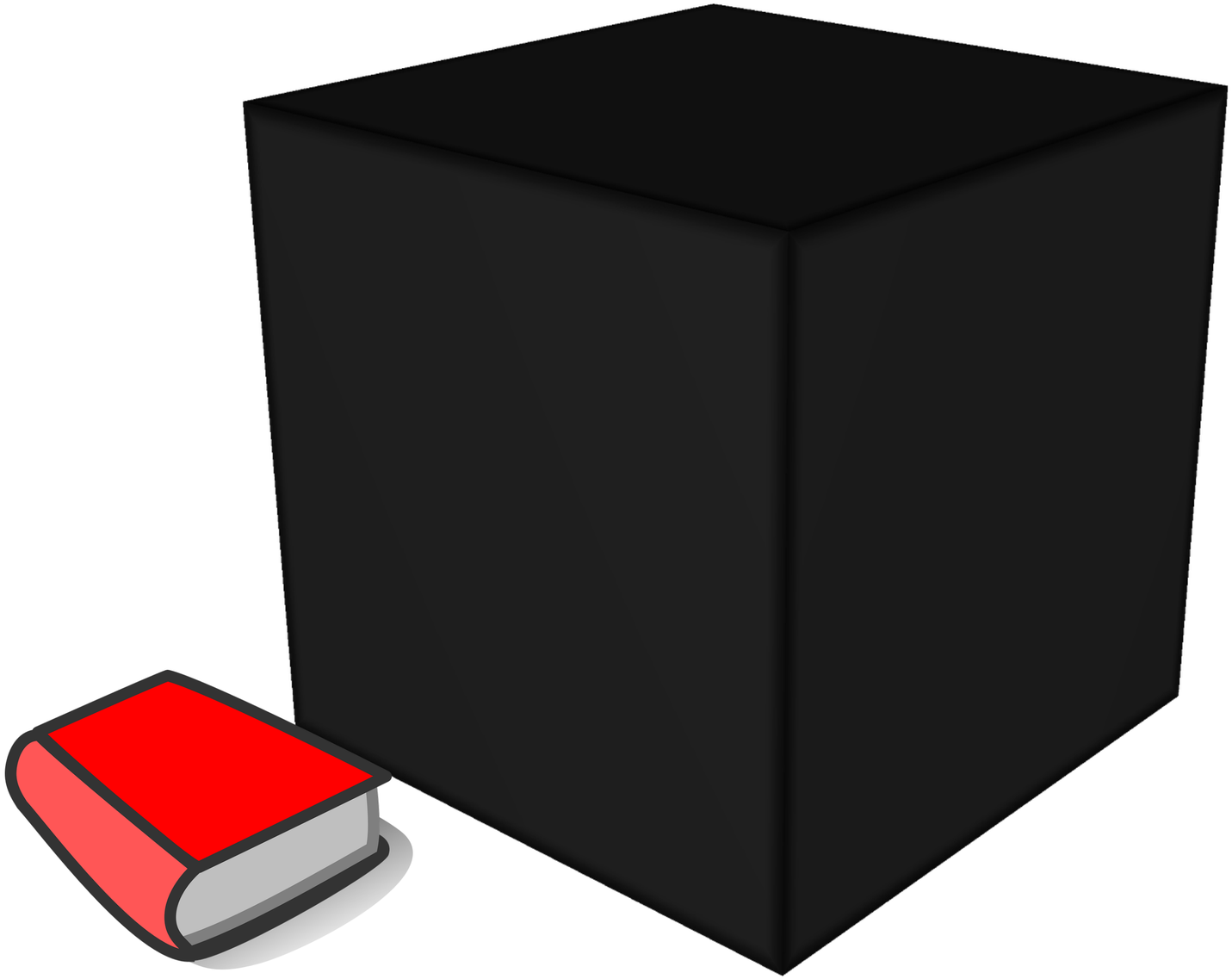


learning rate



learning rate

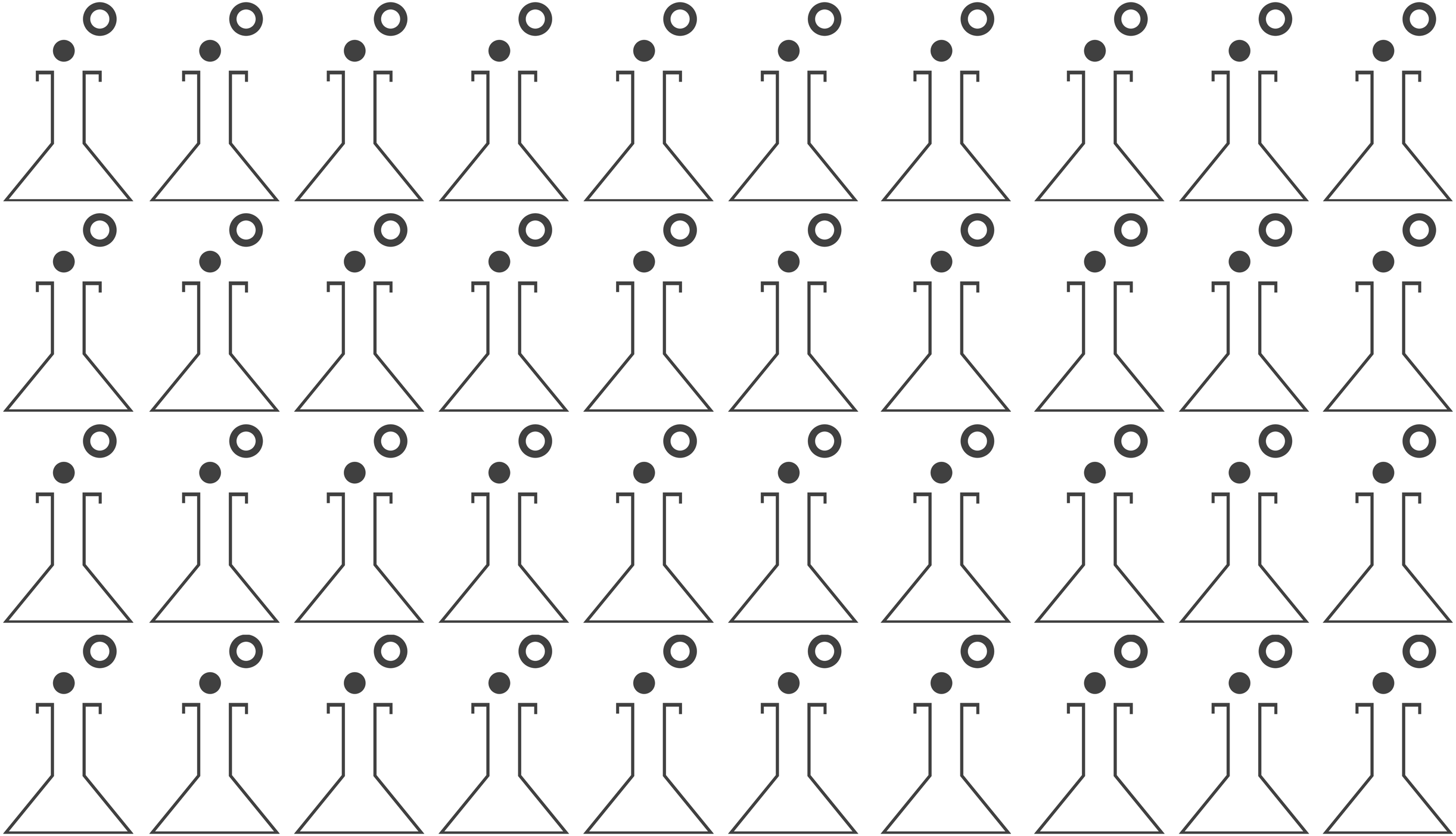


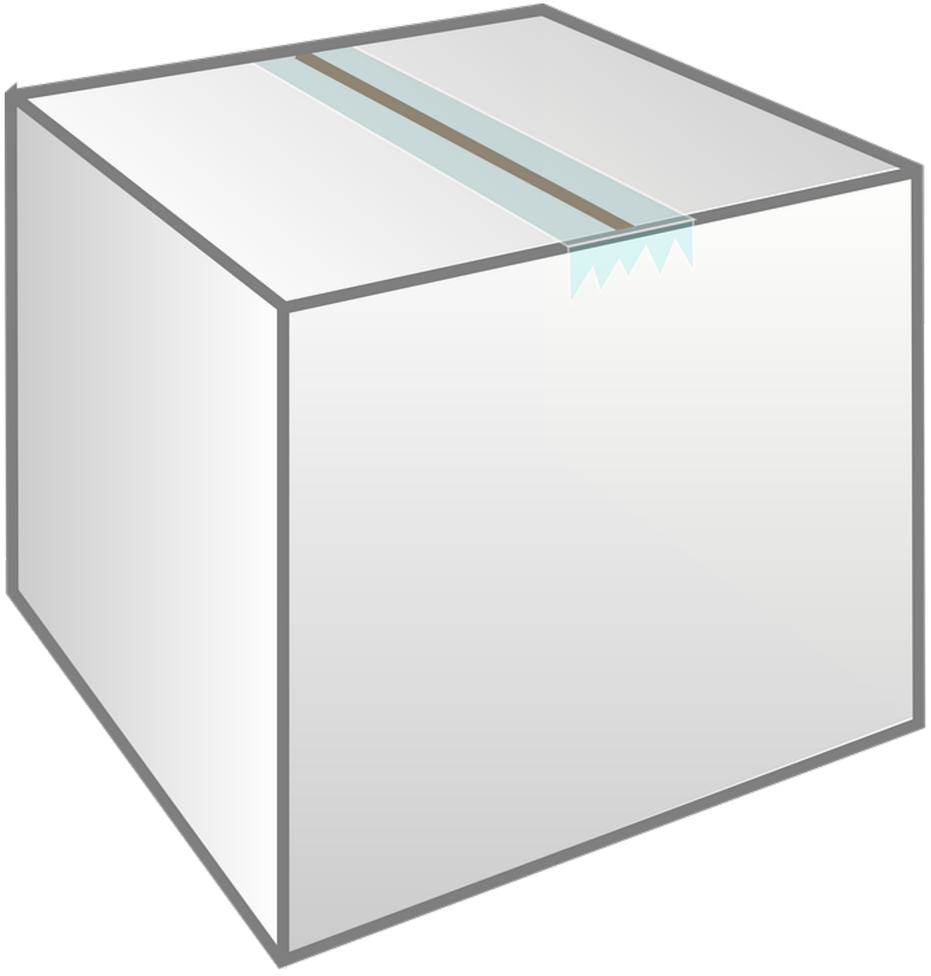


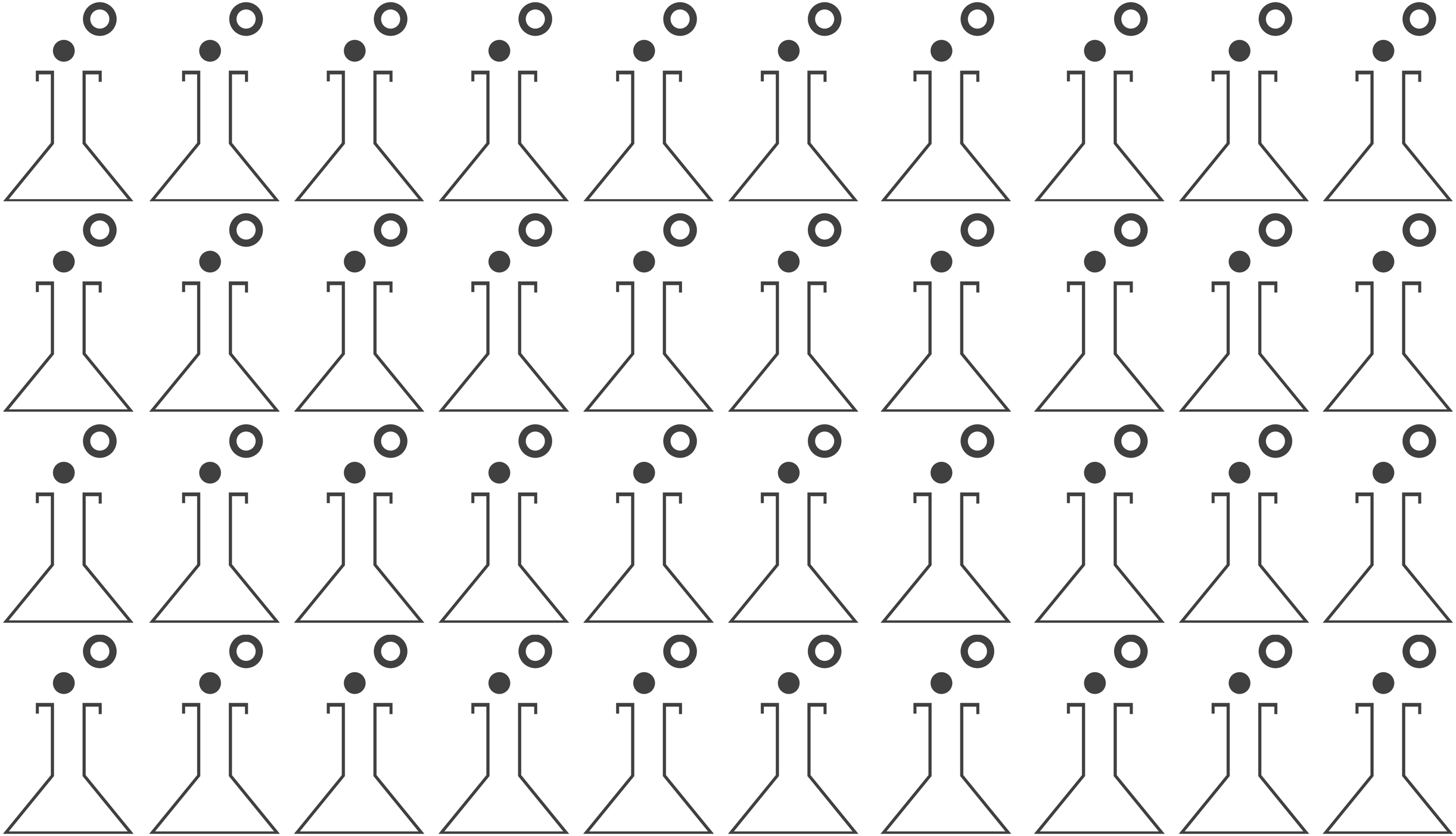
```
def debug():
    return 0 + “no”
```

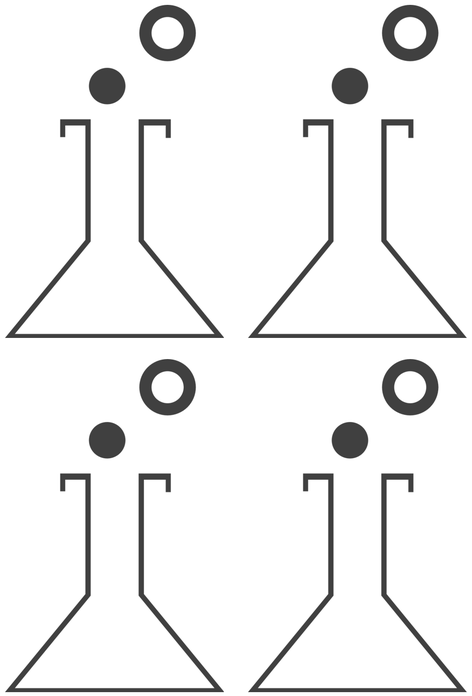
```
567 563m8dep

    j6lmj? _ g “?@”
```









Why deep learning?

(Super) human performance in

(Super) human performance in

(Super) human performance in

あ → A

(Super) human performance in

あ → A

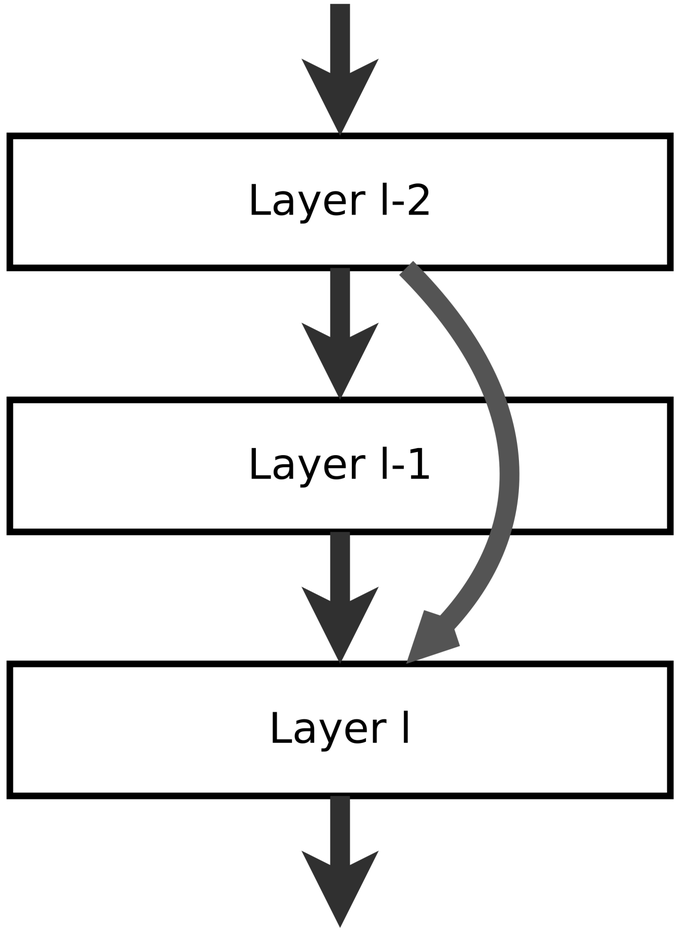
Outline

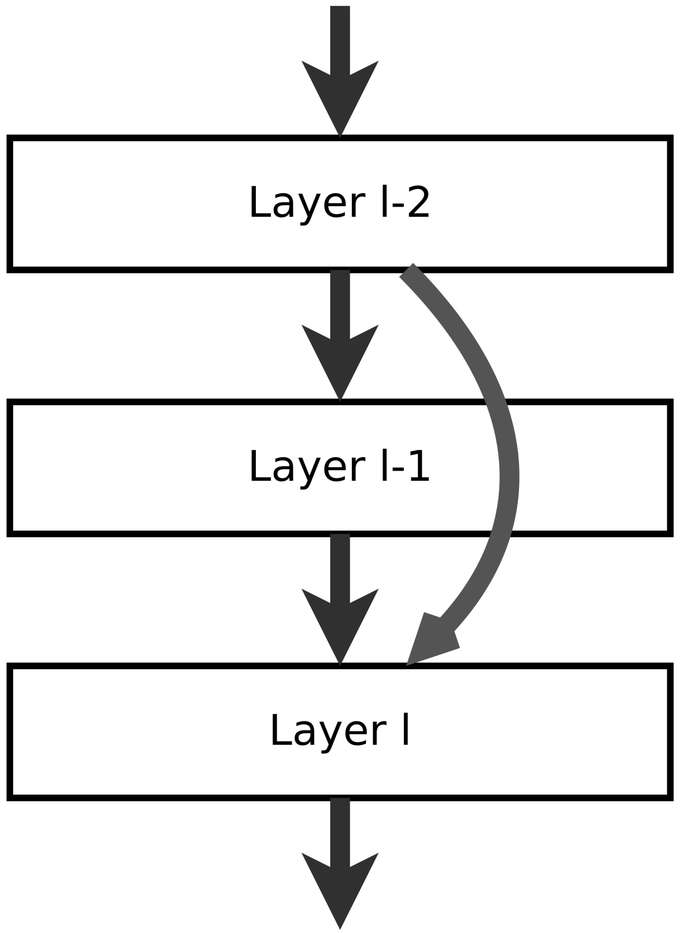
1. Single layer and putting layers together
2. How to train models
3. Deeper is better
4. Practical tips

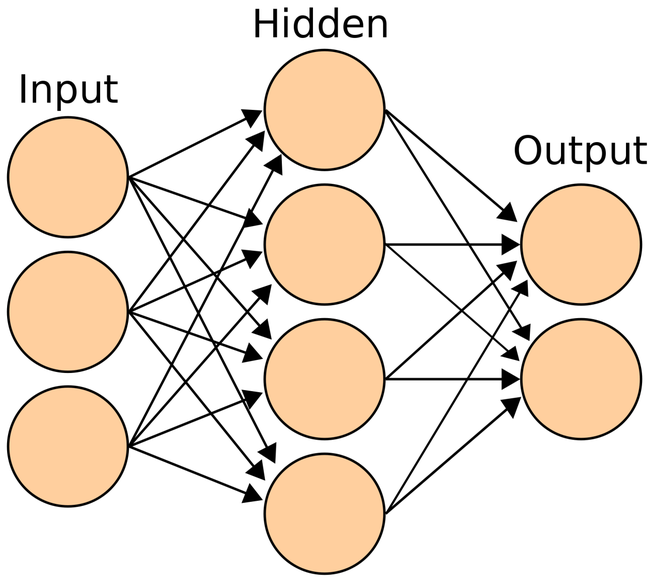
Understanding is not something you can Google.

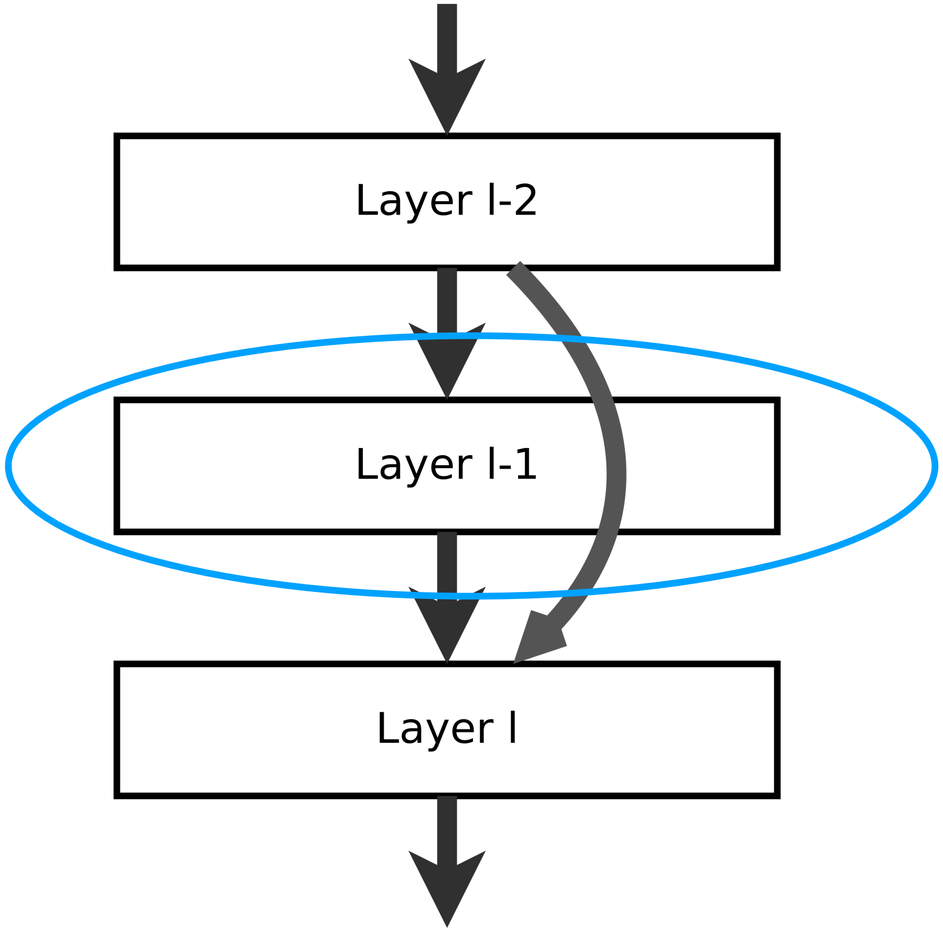
Machine learning is engineering-driven.

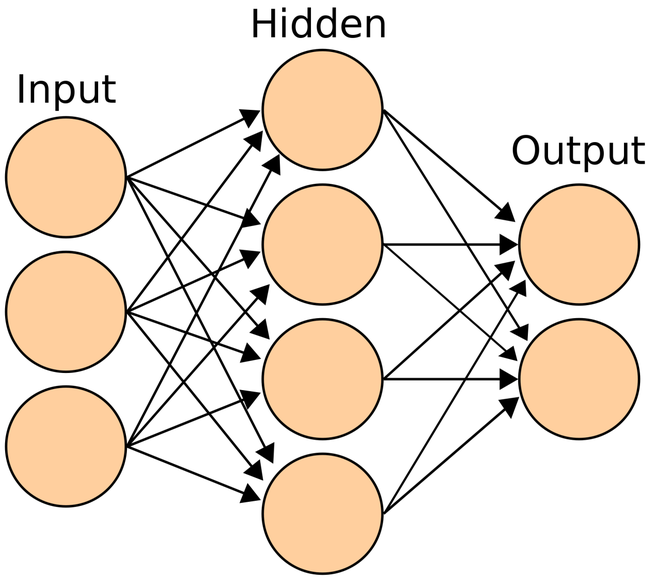
I. Single Layer

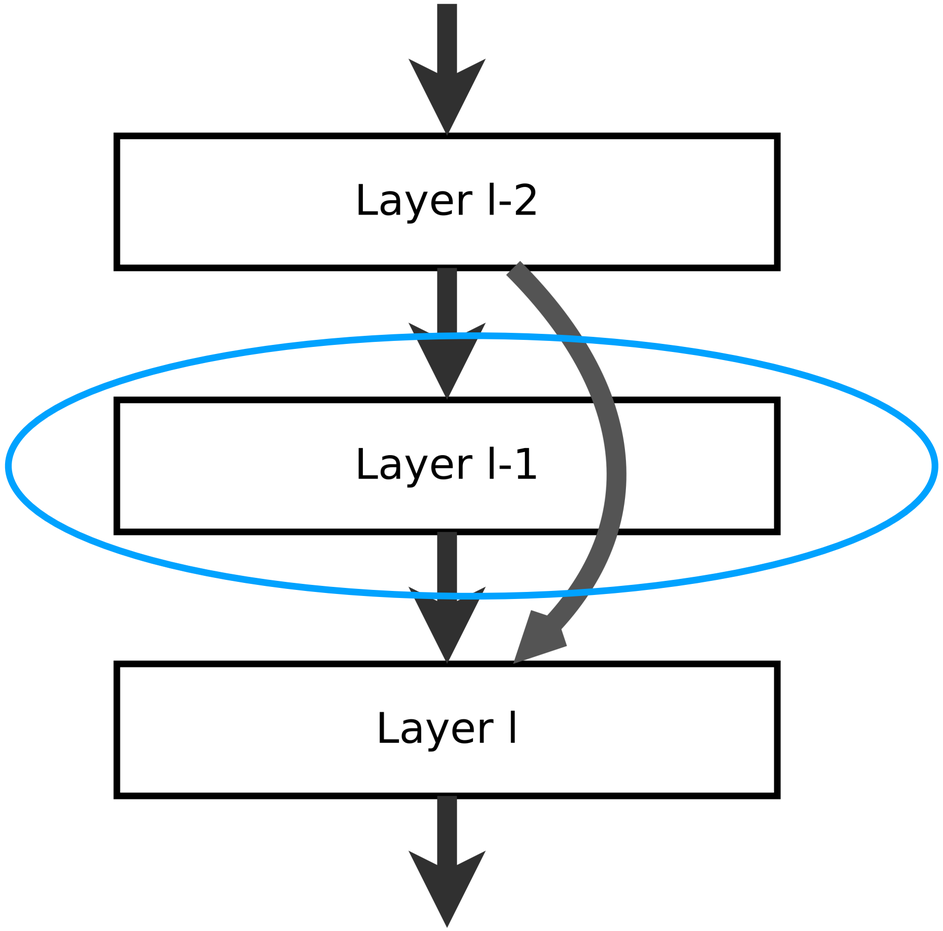

Layer l-2
Layer l-1
Layer l

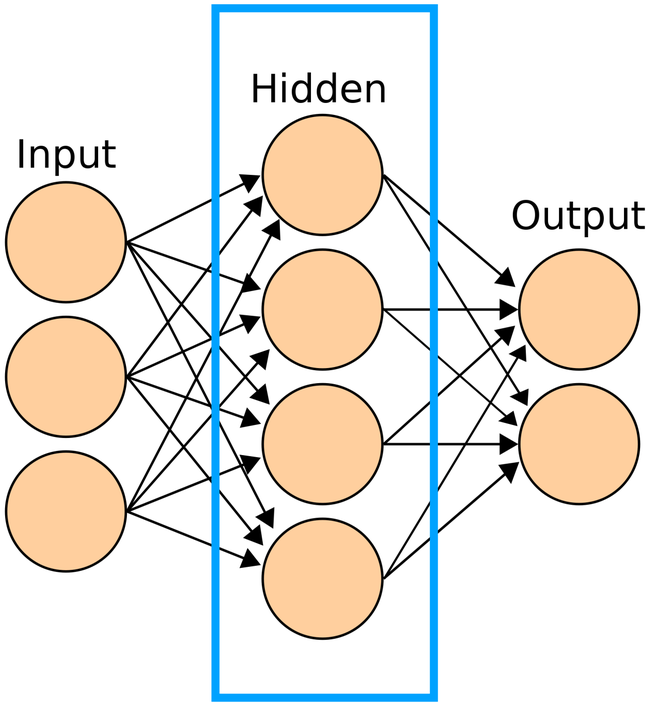

Layer l-2
Layer l-1
Layer l


Input
Hidden
Output

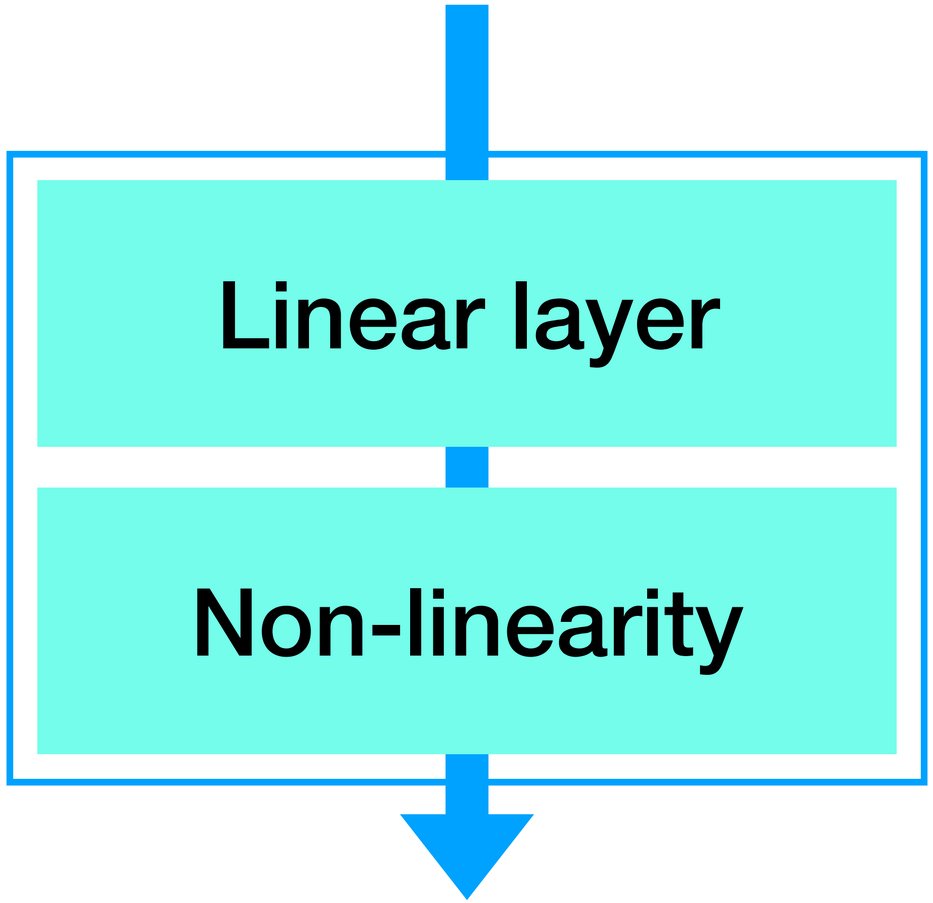

Layer l-2
Layer l-1
Layer l


Input
Hidden
Output


Layer l-2
Layer l-1
Layer l

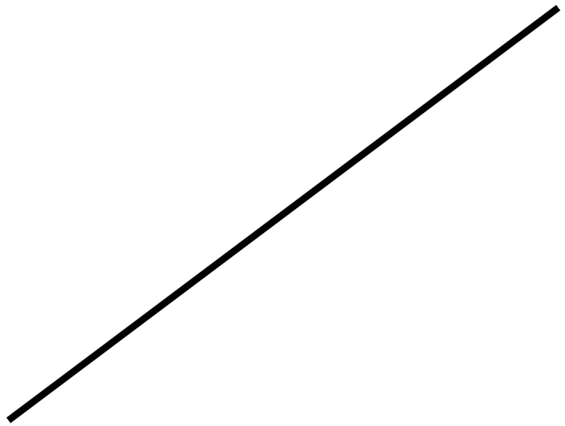

Input
Hidden
Output

Fully connected layer

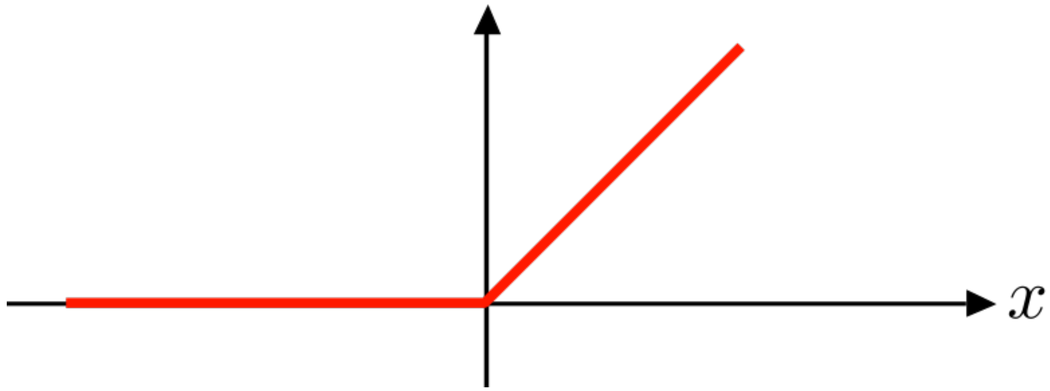


A single layer

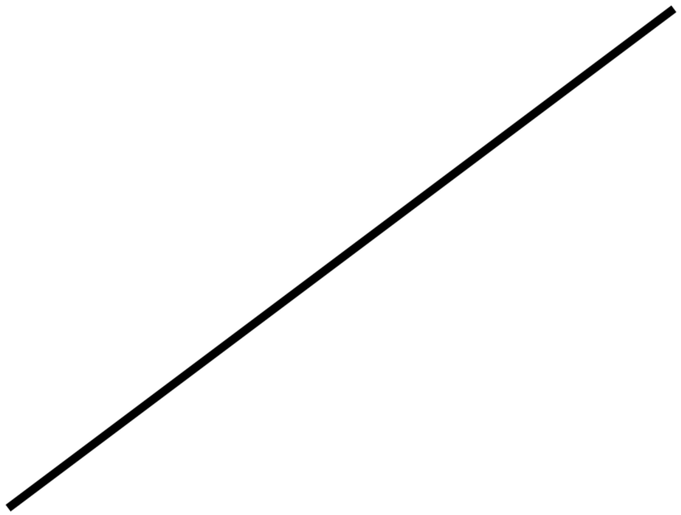
Linear layer



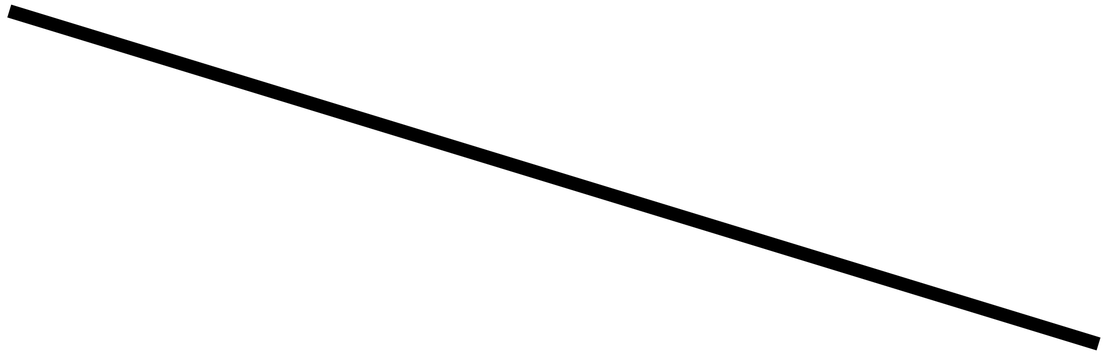
Non-linearity

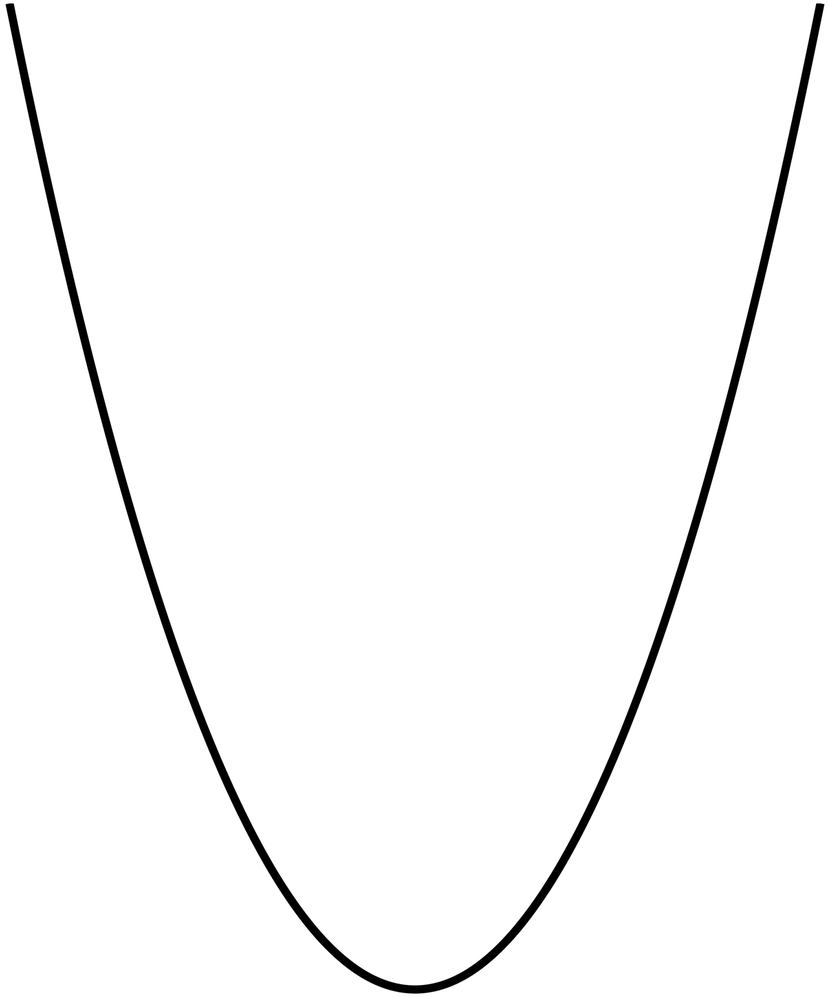


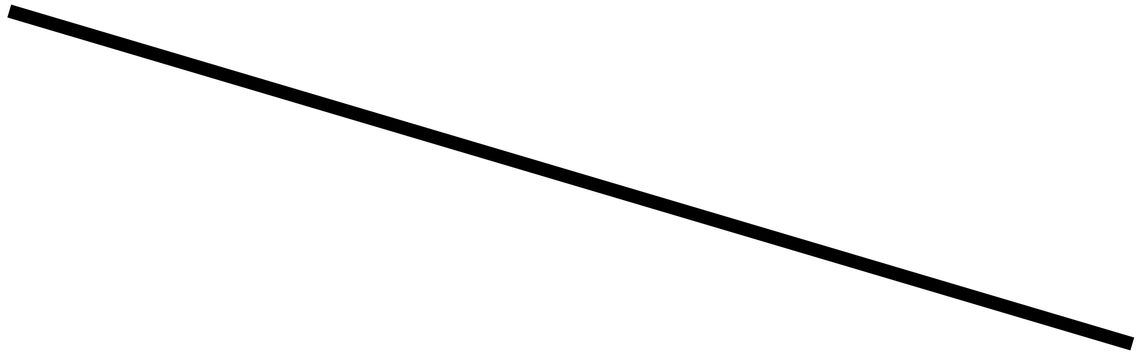
Linear layer

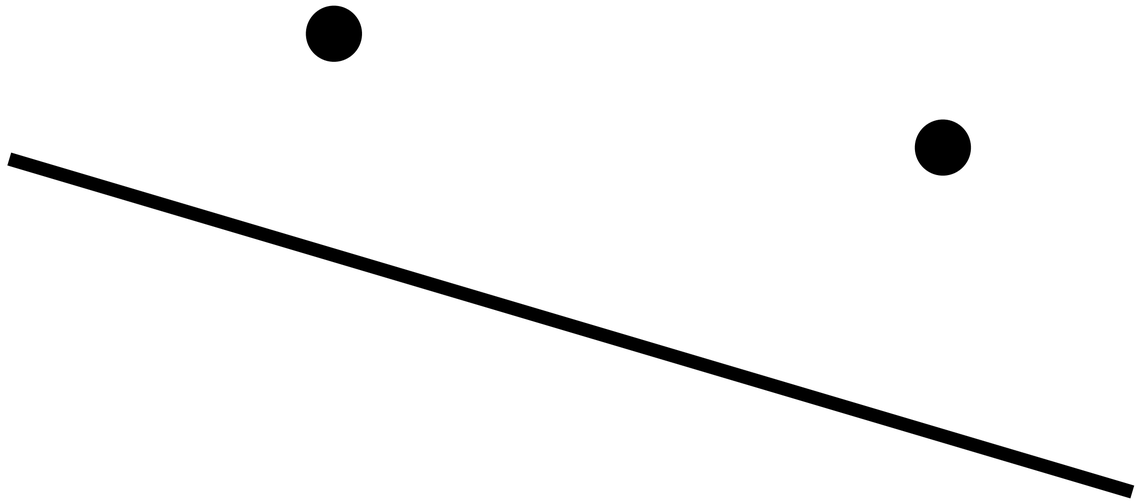


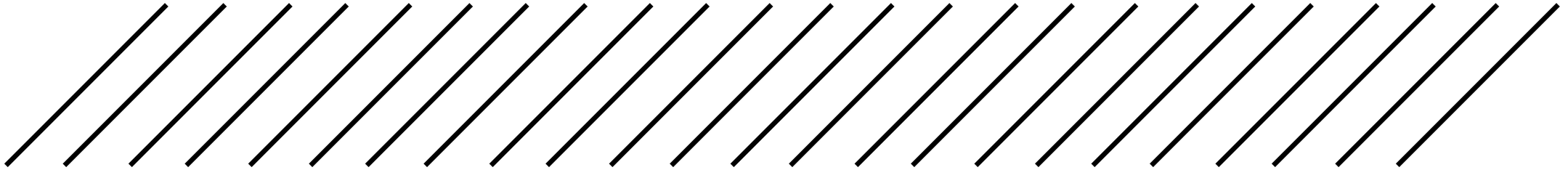
$$\mathbf{W}\mathbf{x} + \mathbf{b}$$

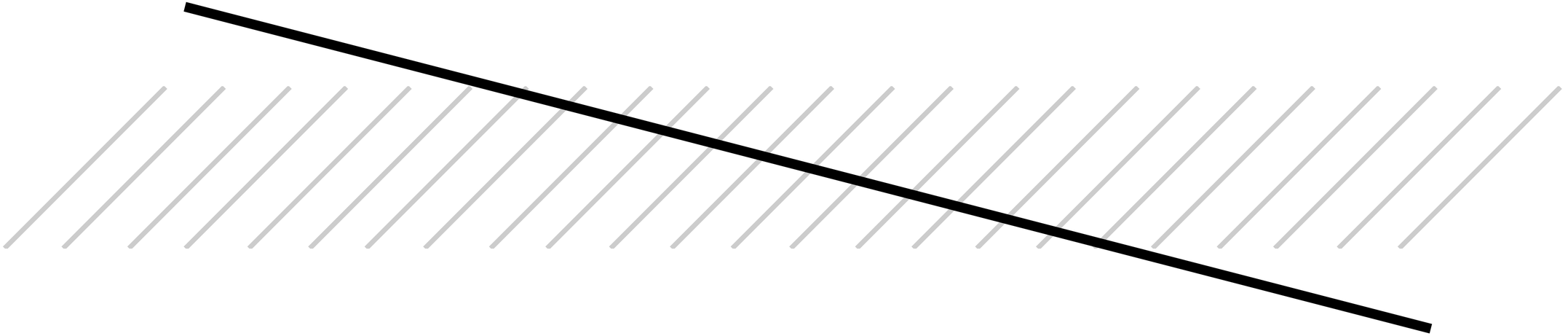


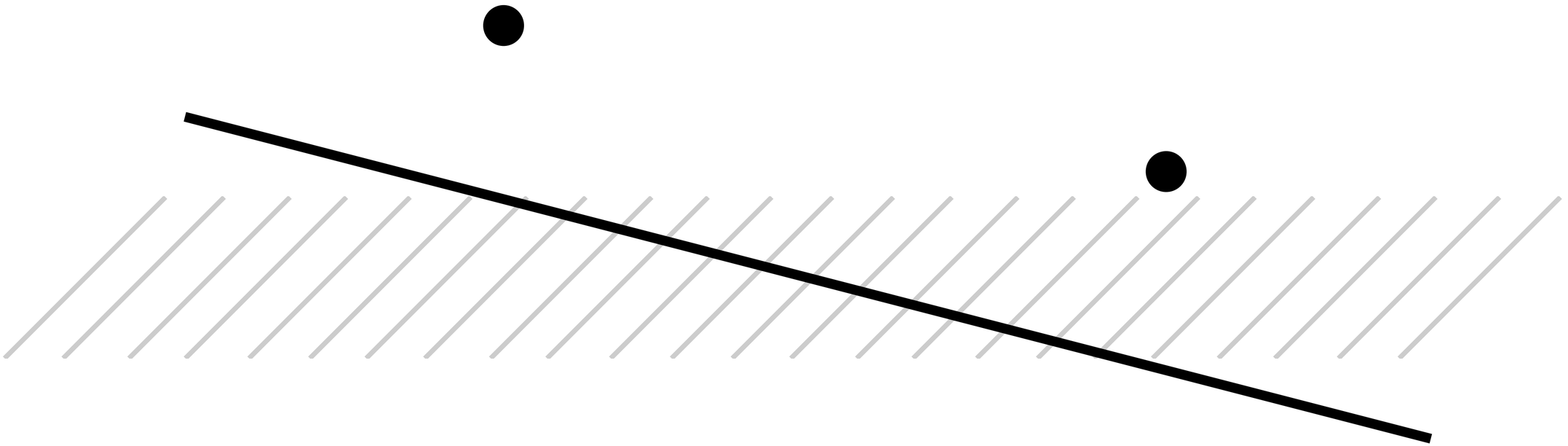


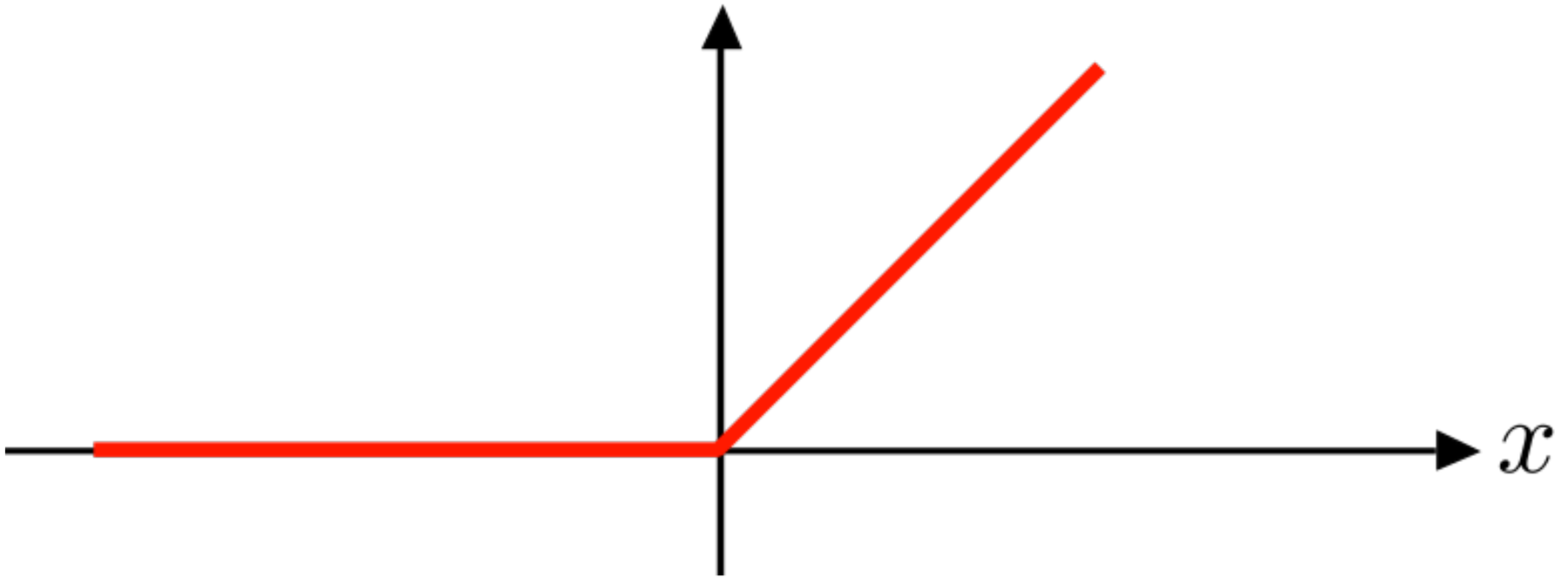


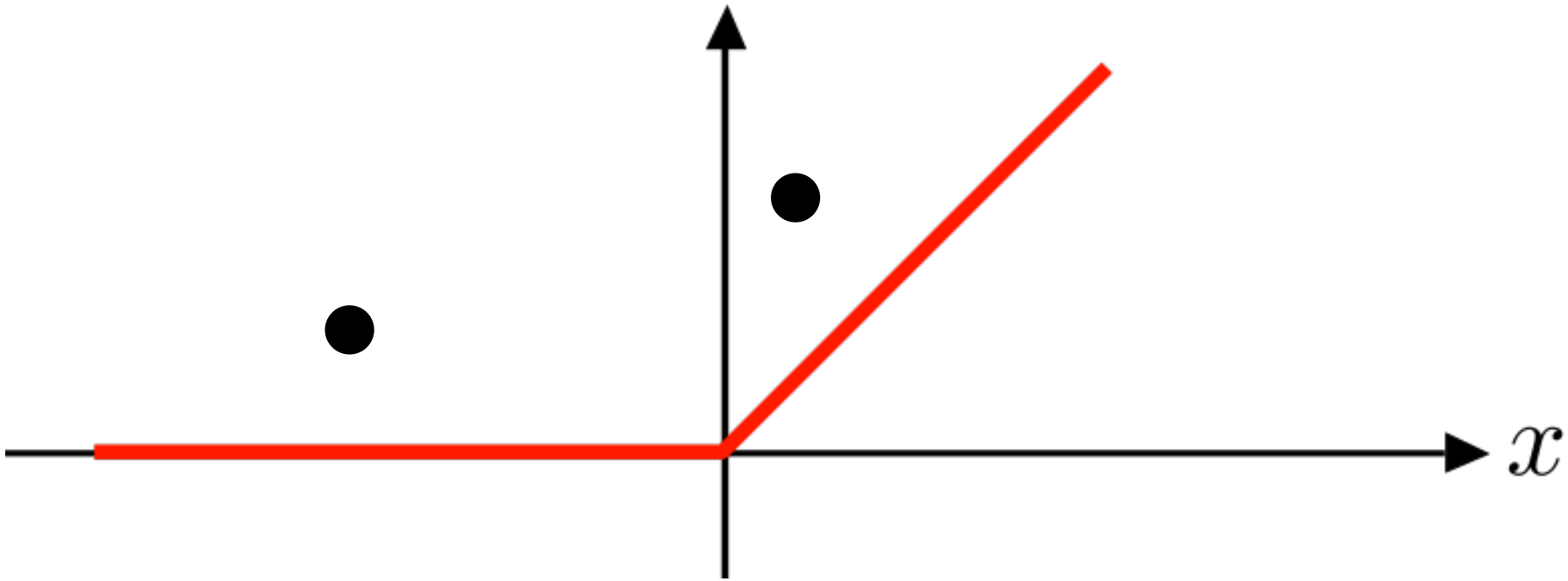






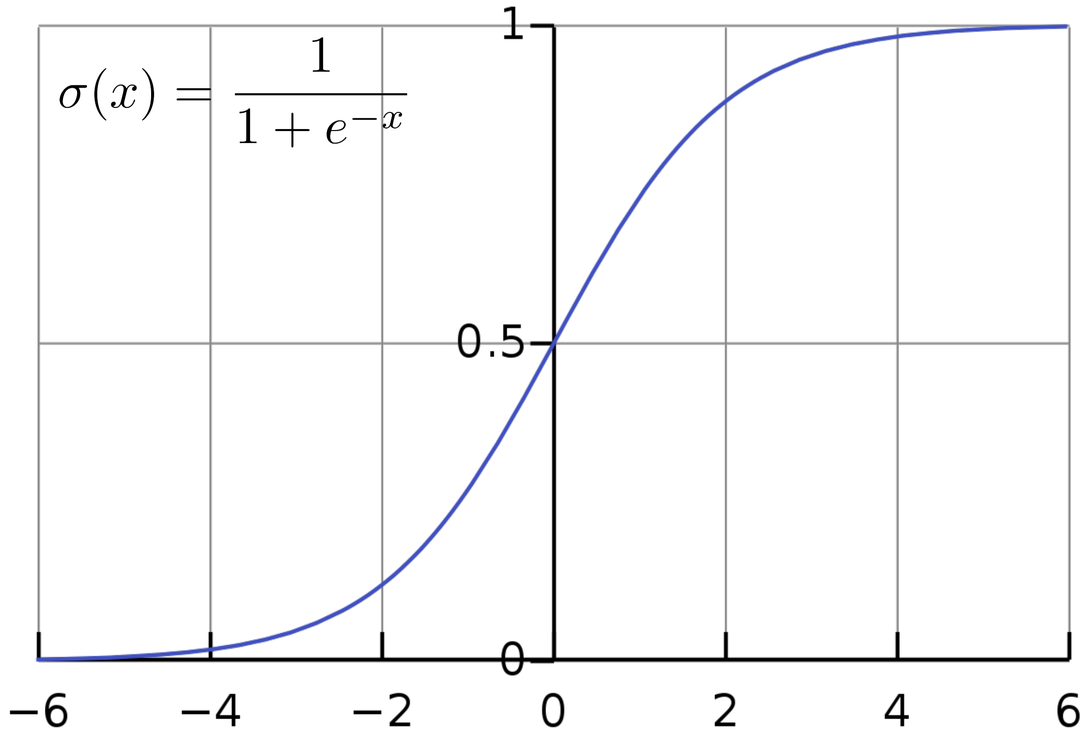



x

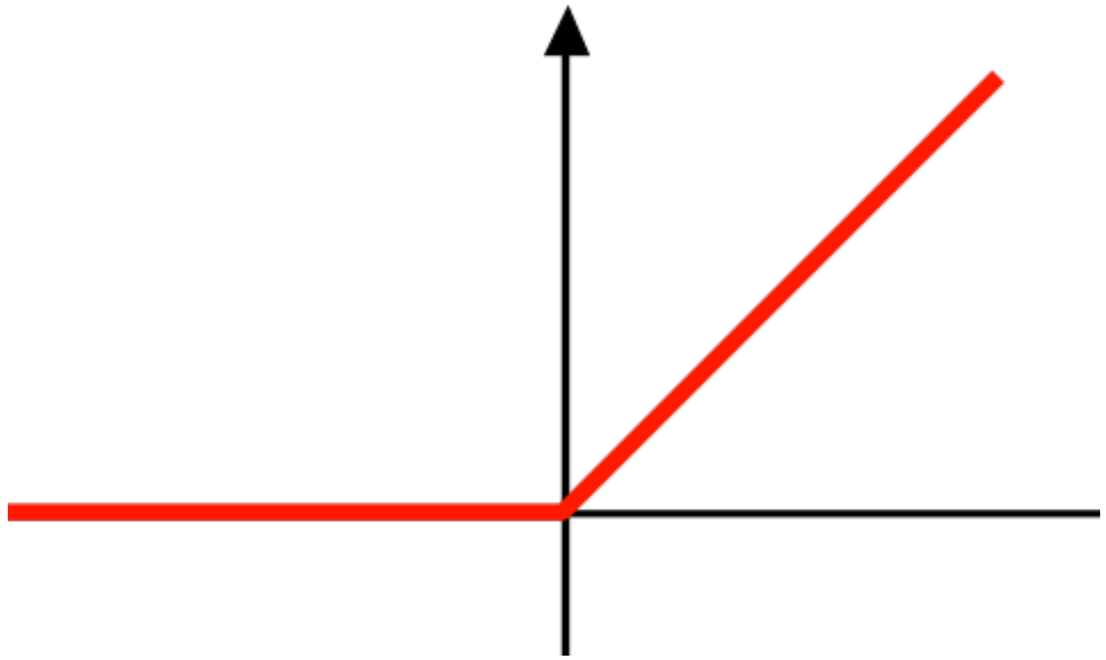

x

Non-linearities

Sigmoid

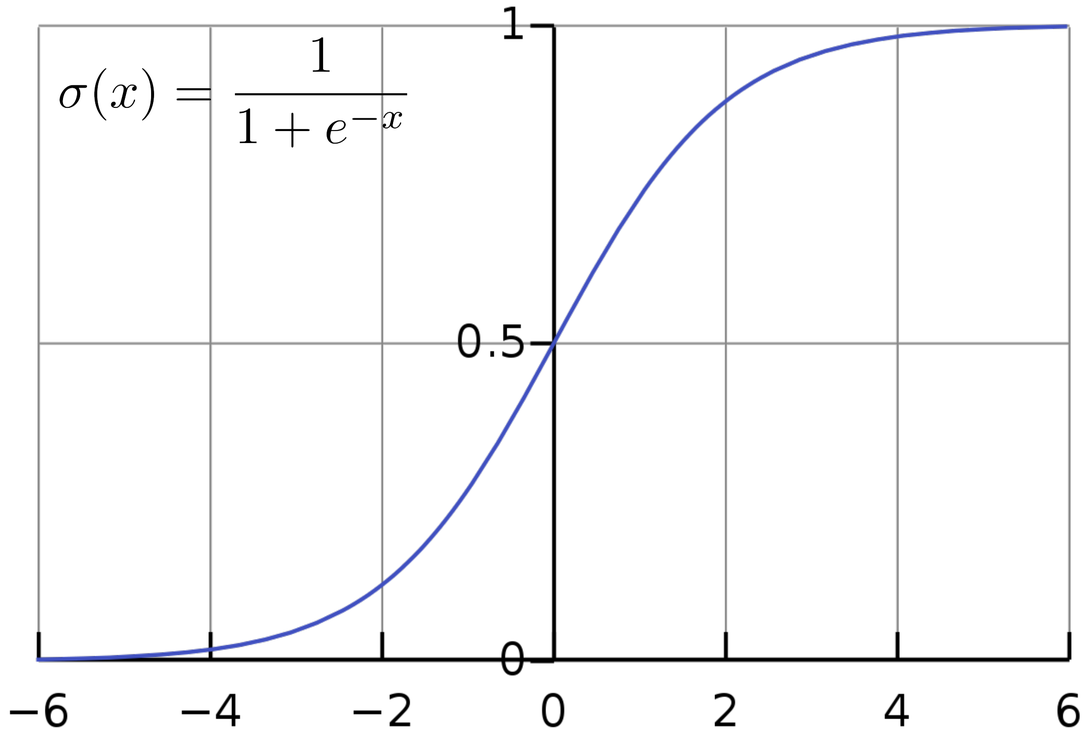


ReLU

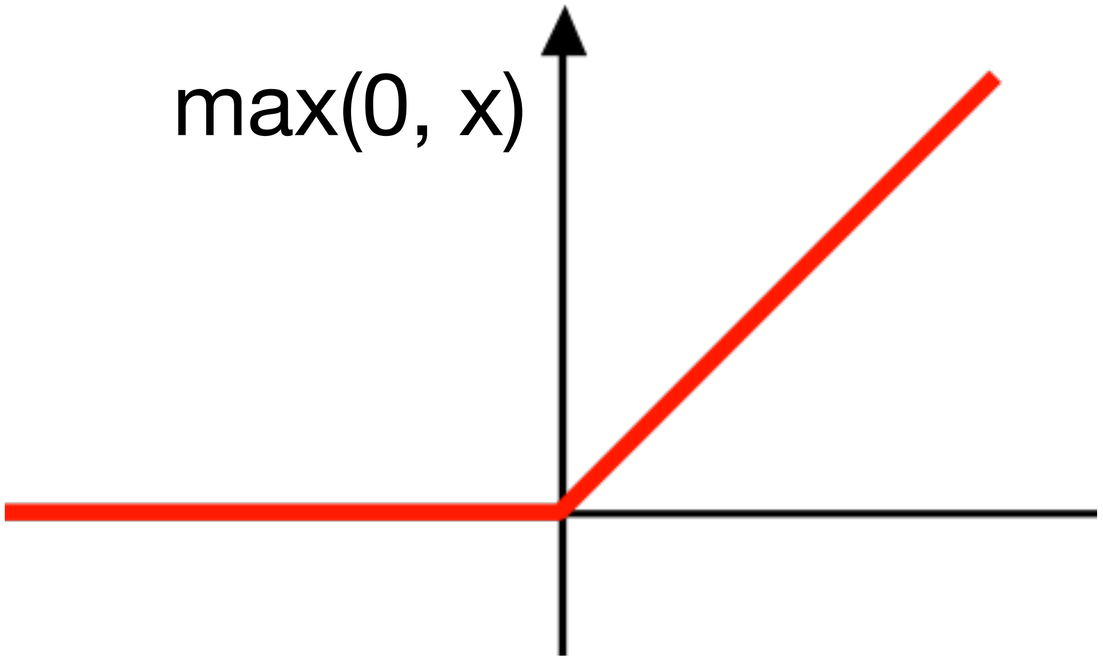


Non-linearities

Sigmoid

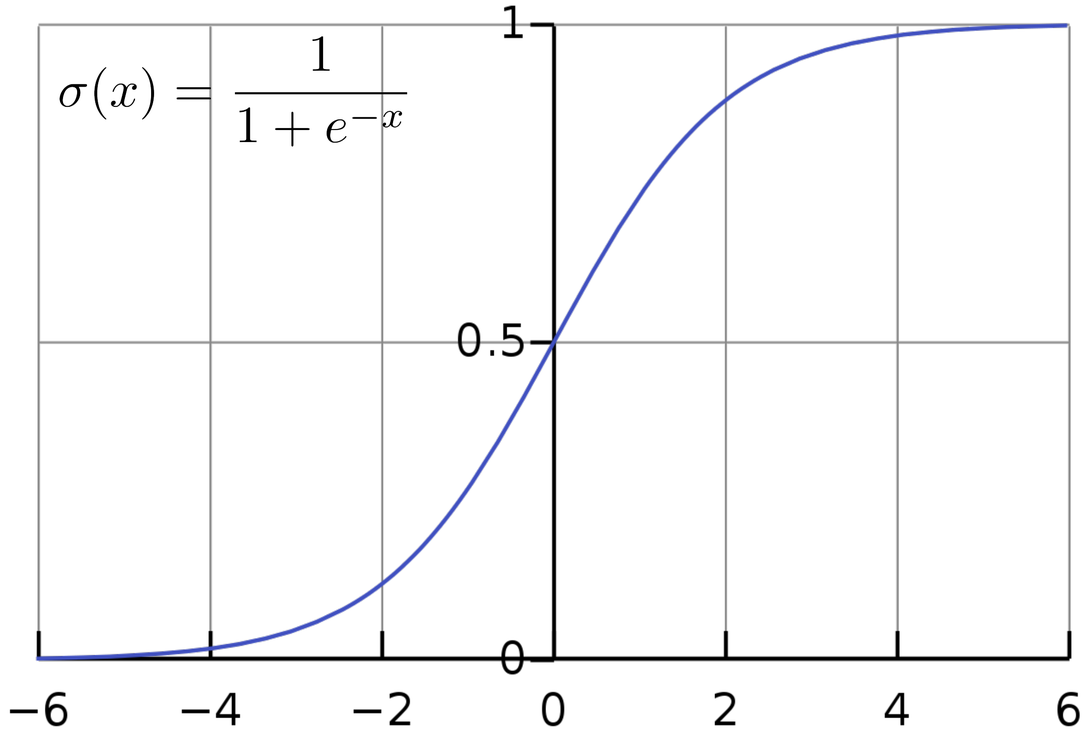


ReLU

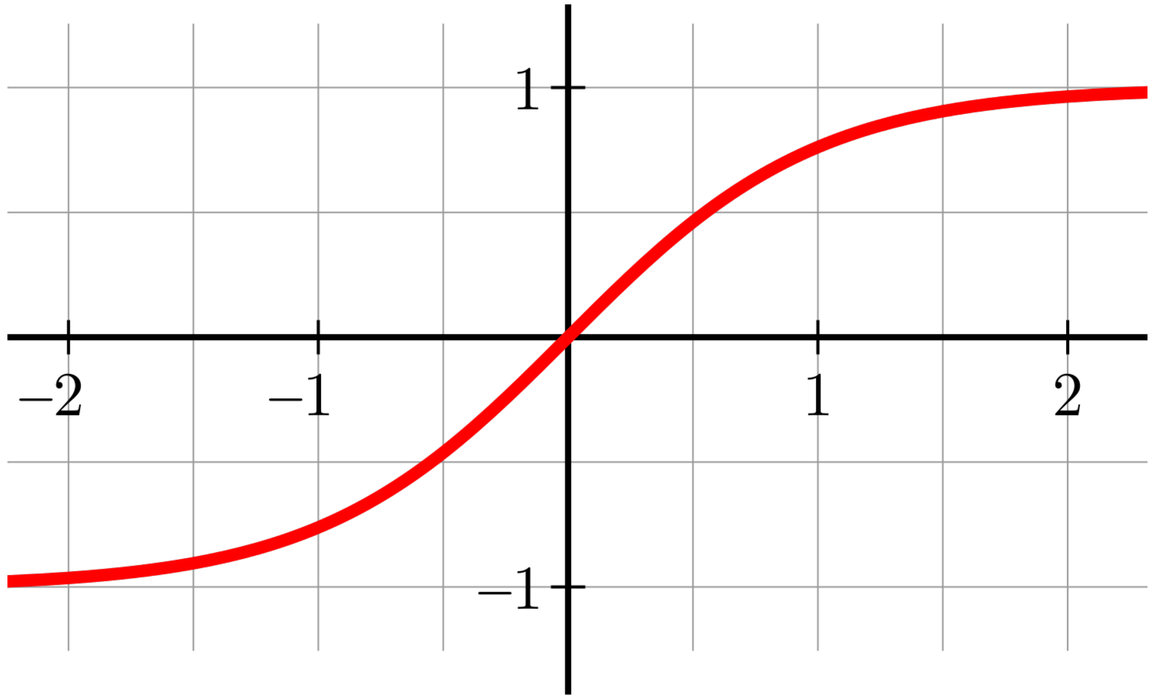


Output layer

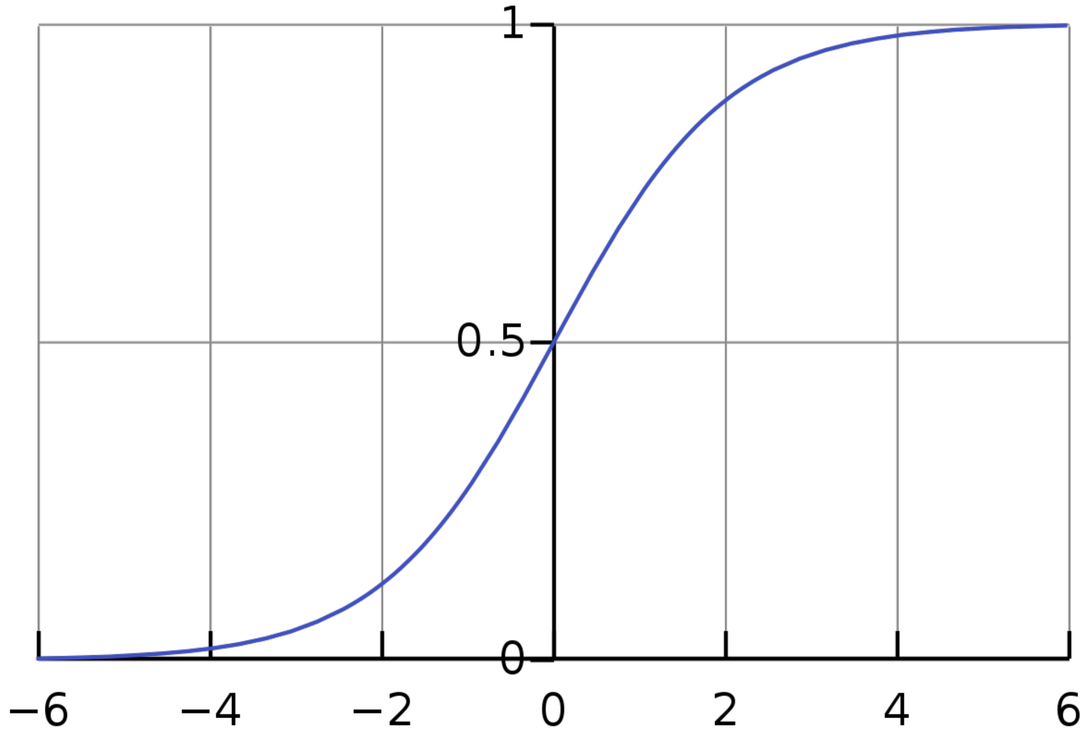
Sigmoid



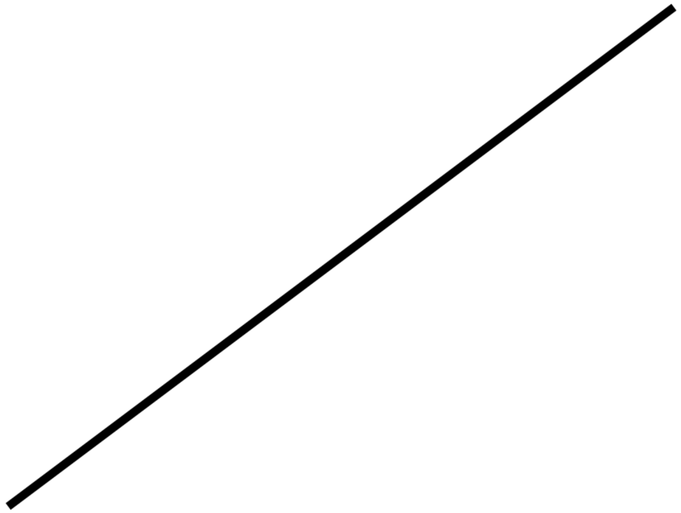
tanh



Sigmoid

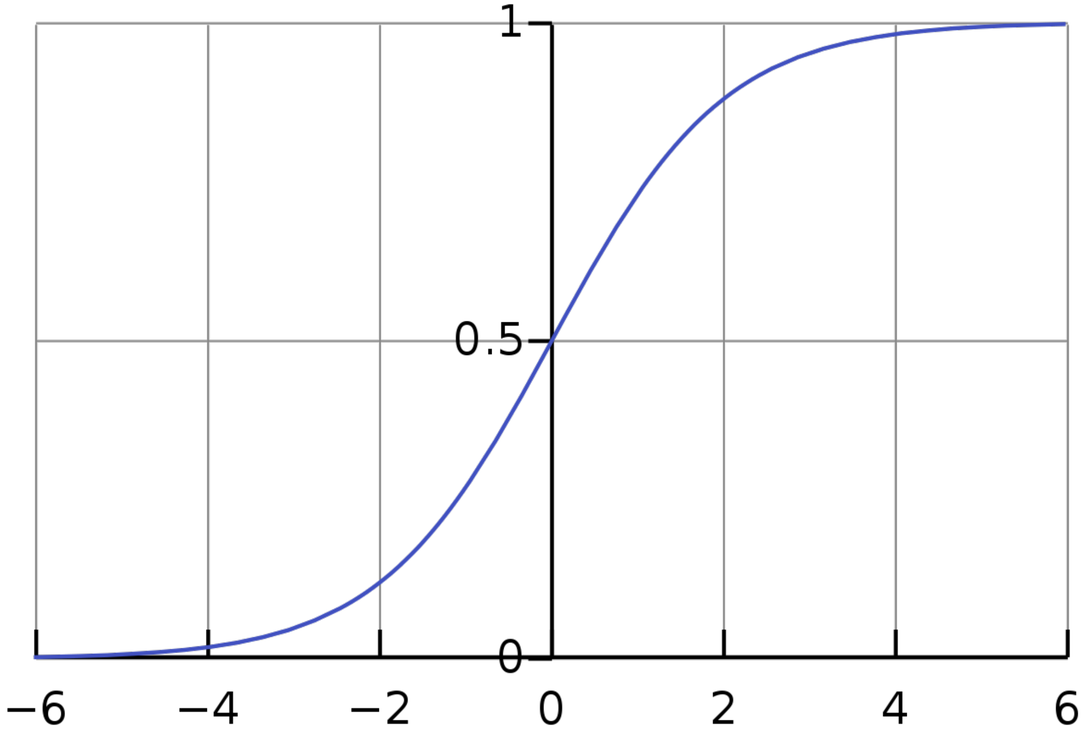


Linear layer

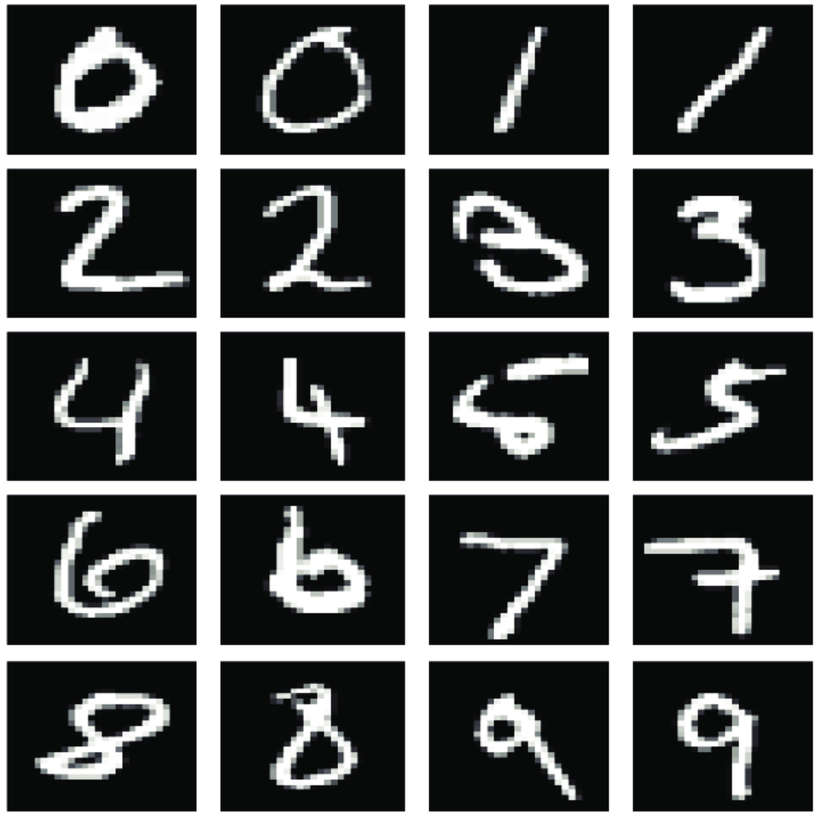


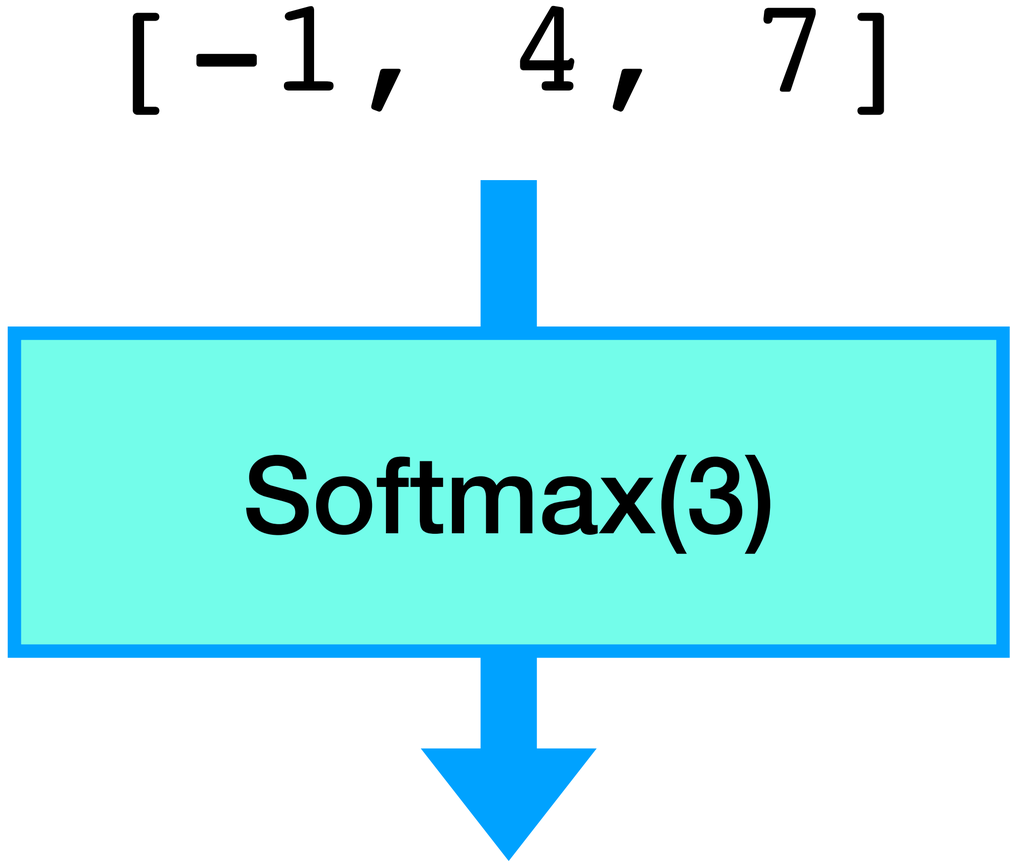
$$\mathbf{W}\mathbf{x} + \mathbf{b}$$

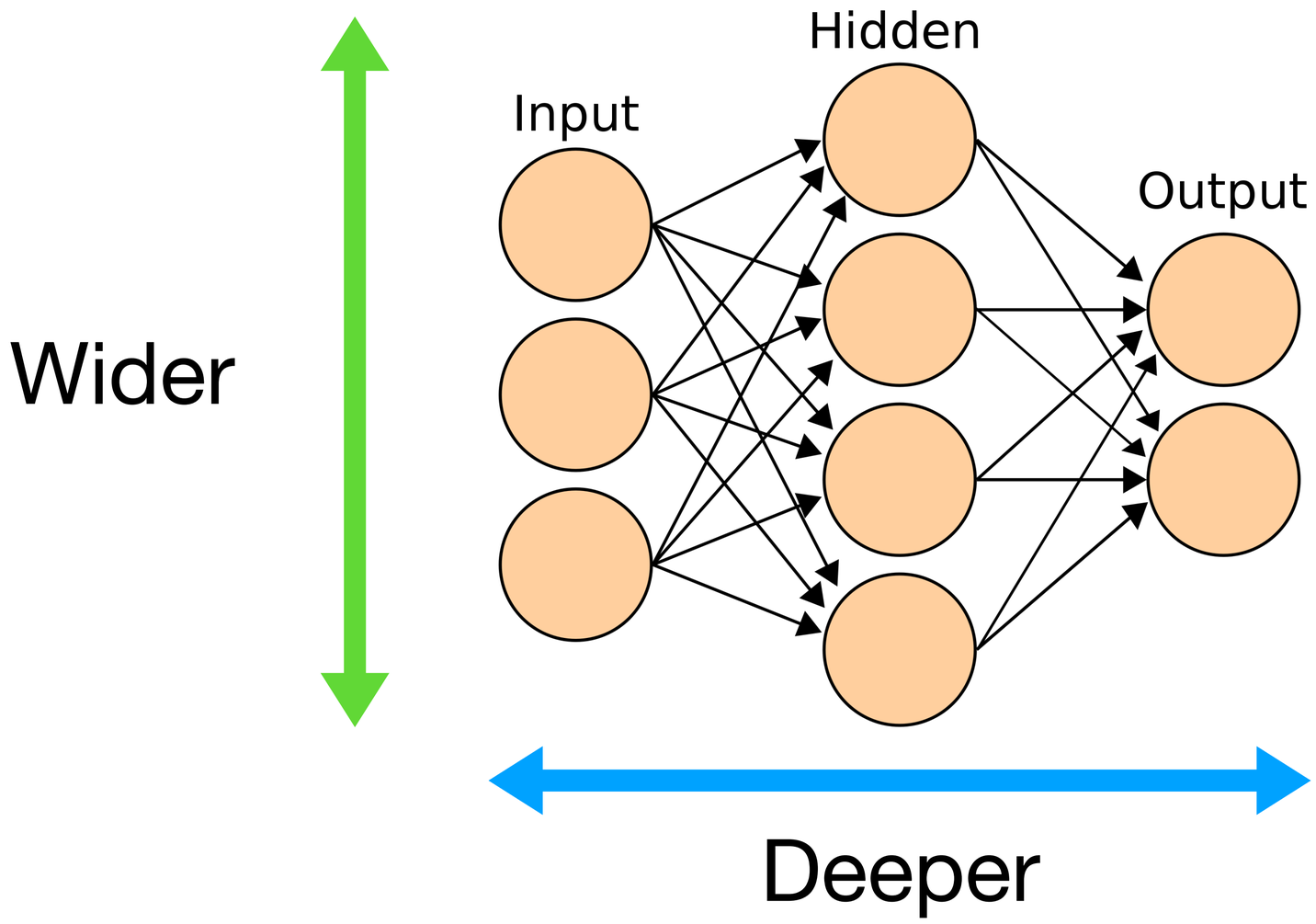
Sigmoid

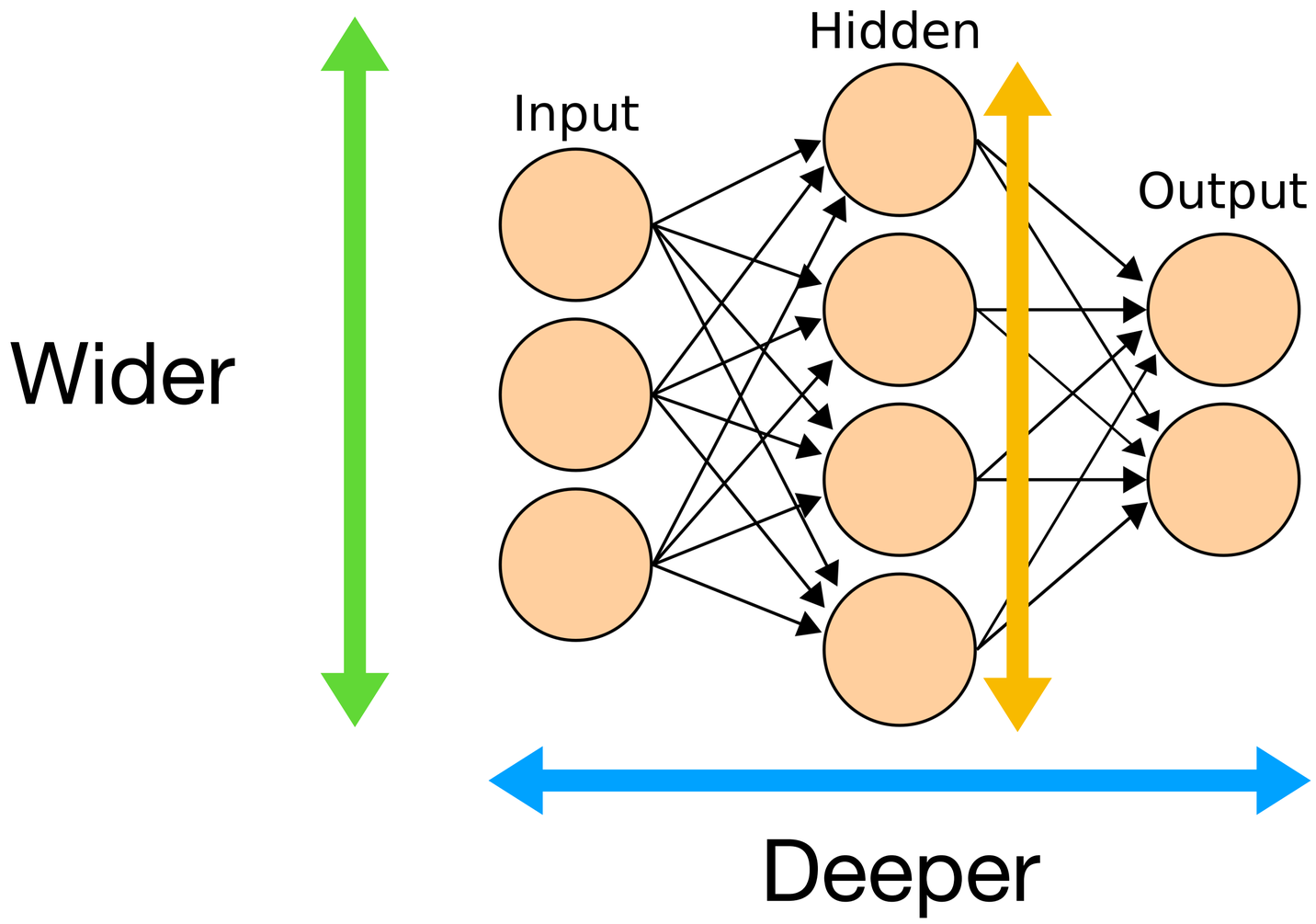


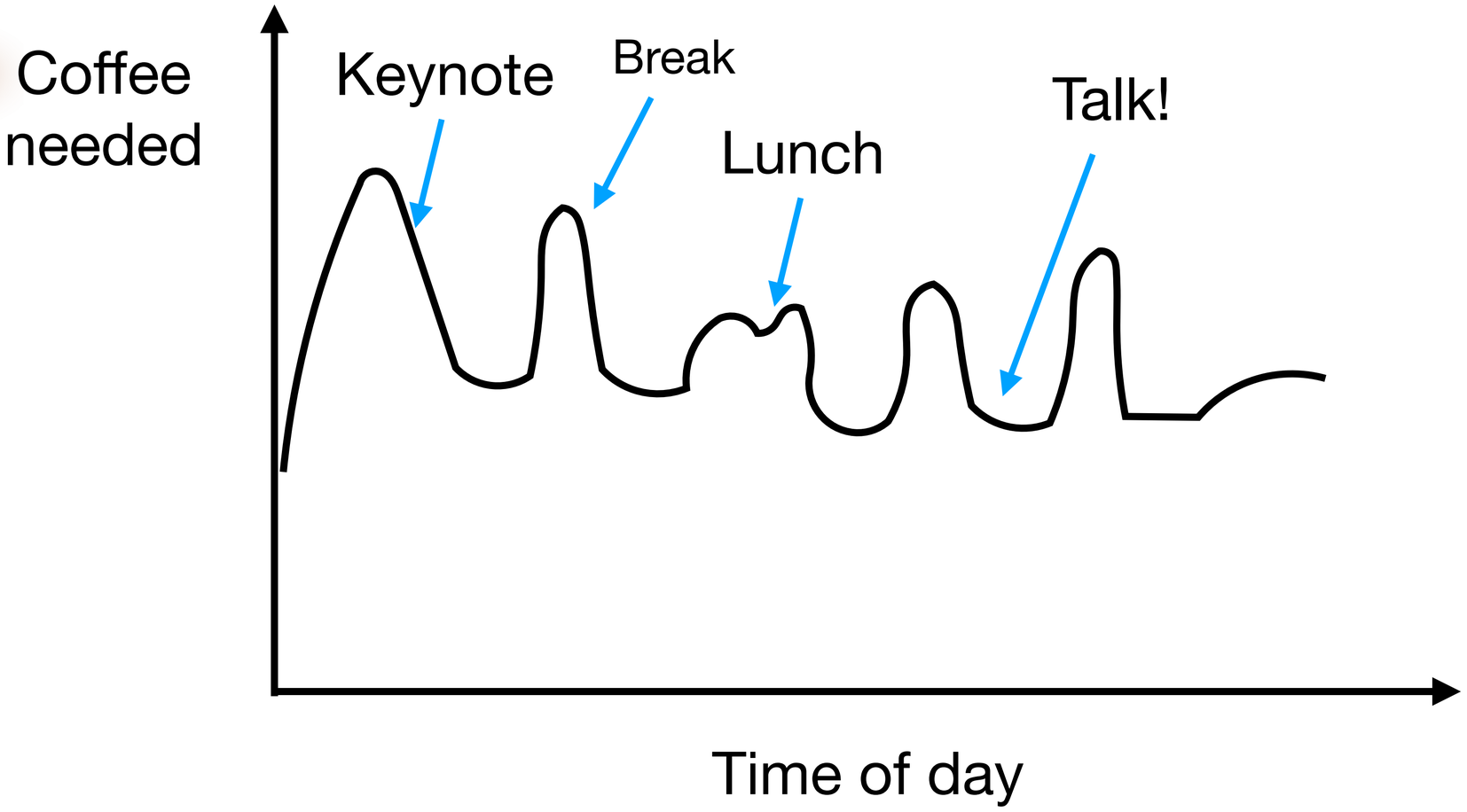
MNIST dataset

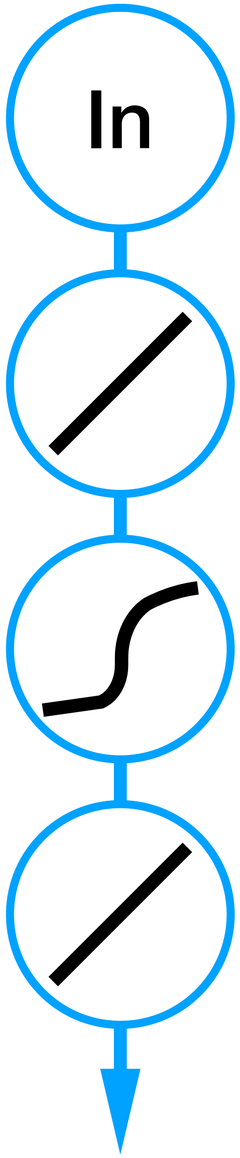


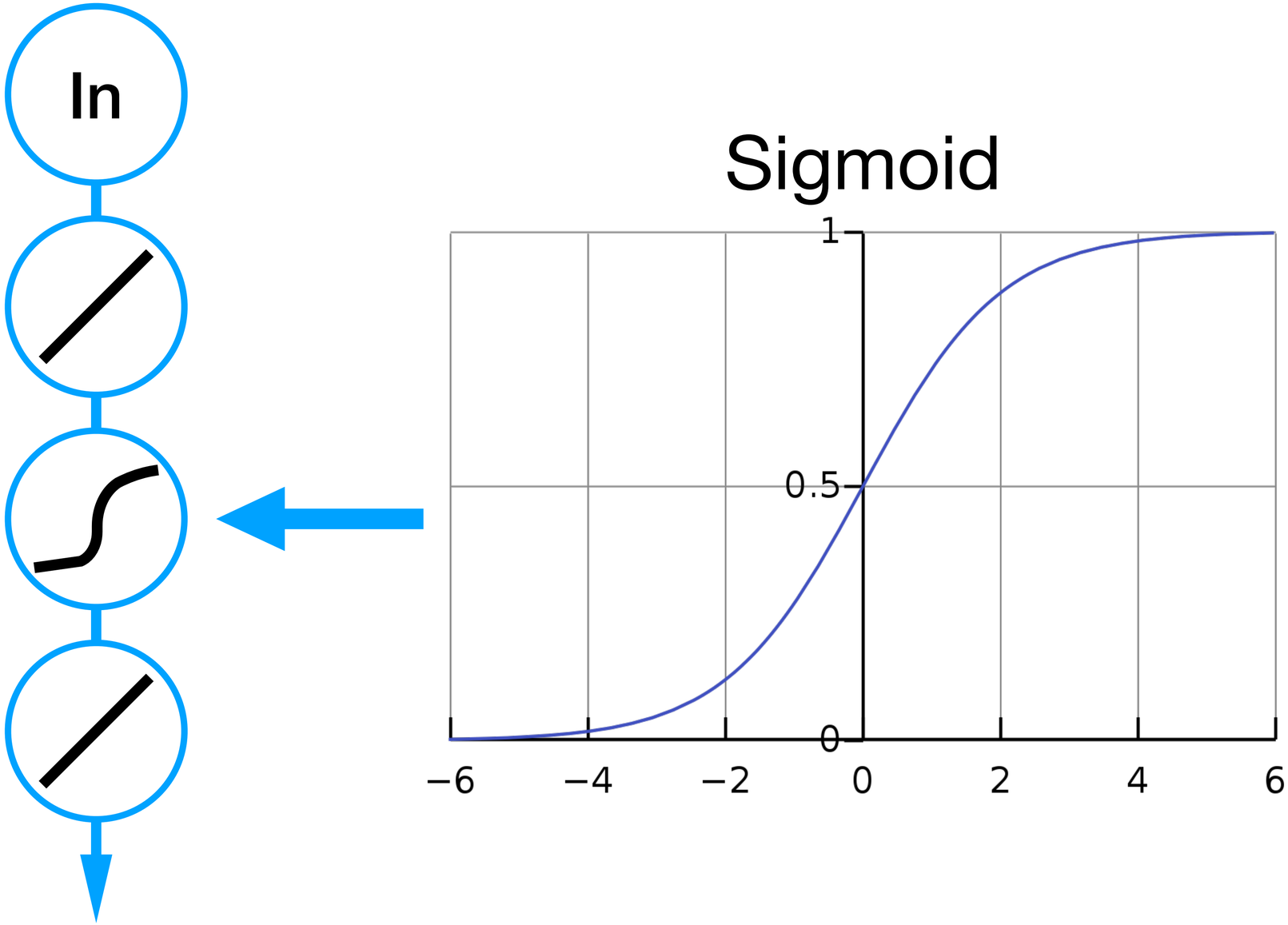

[-1, 4, 7]
Softmax(3)
[0.0003 0.047 0.95]

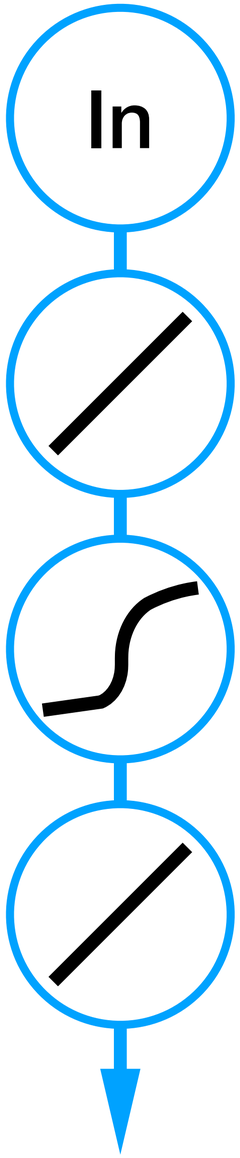

Wider
Input
Hidden
Output
Deeper

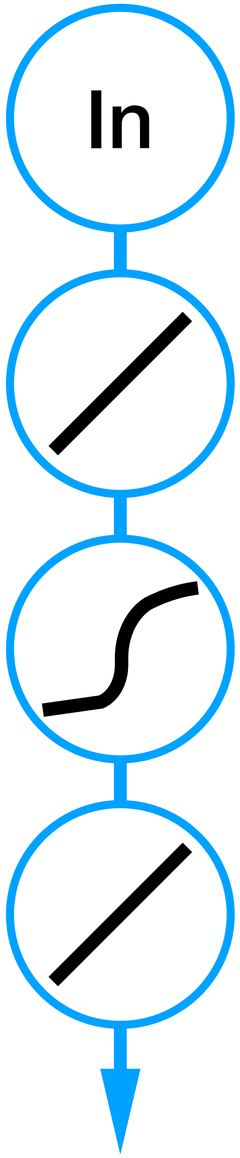

Wider
Input
Hidden
Output
Deeper

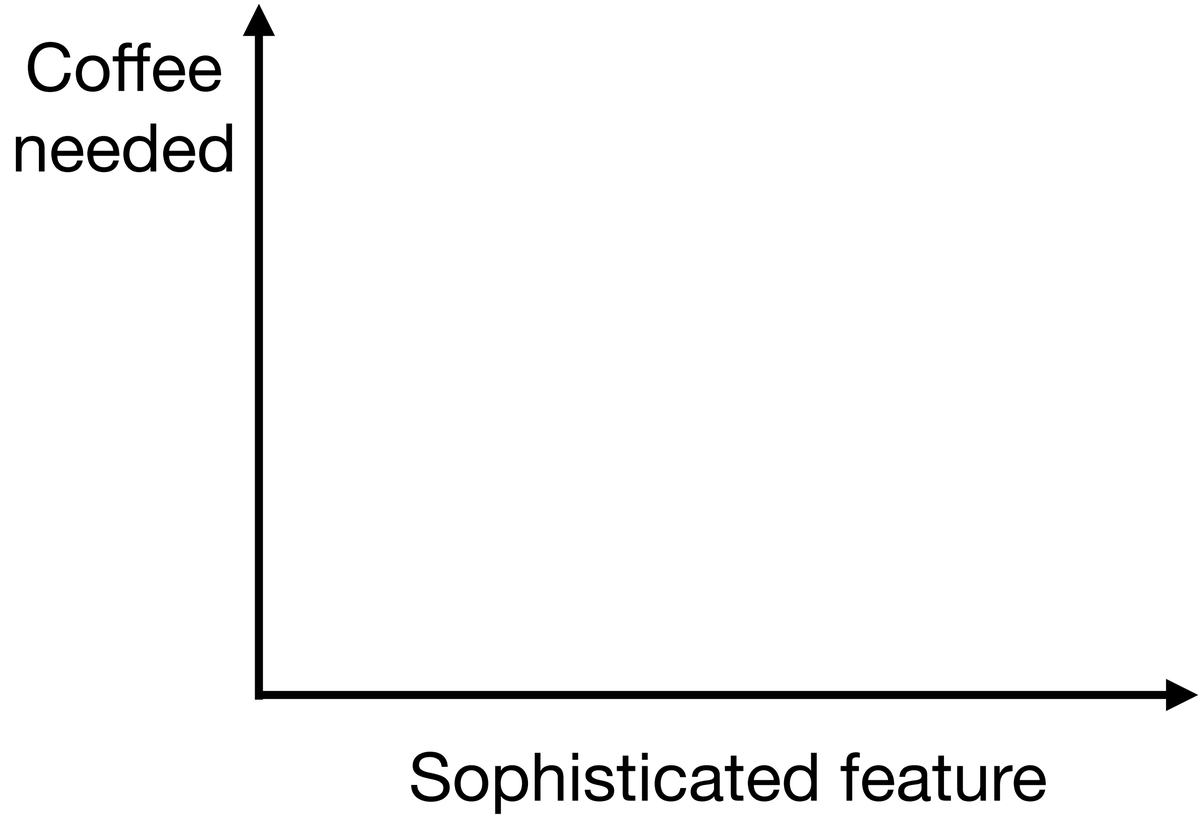

Coffee needed
Keynote
Break
Lunch
Talk!
Time of day

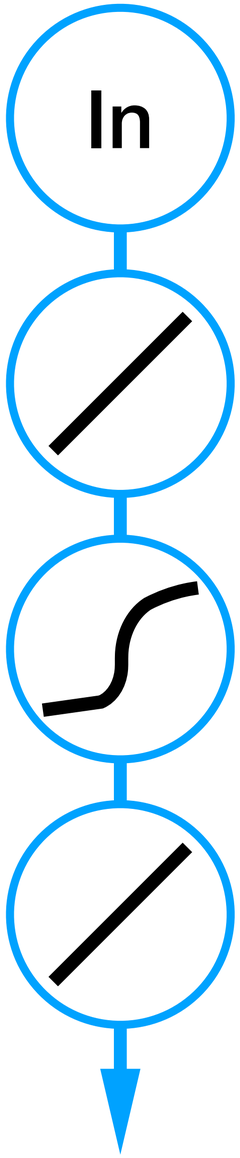

In

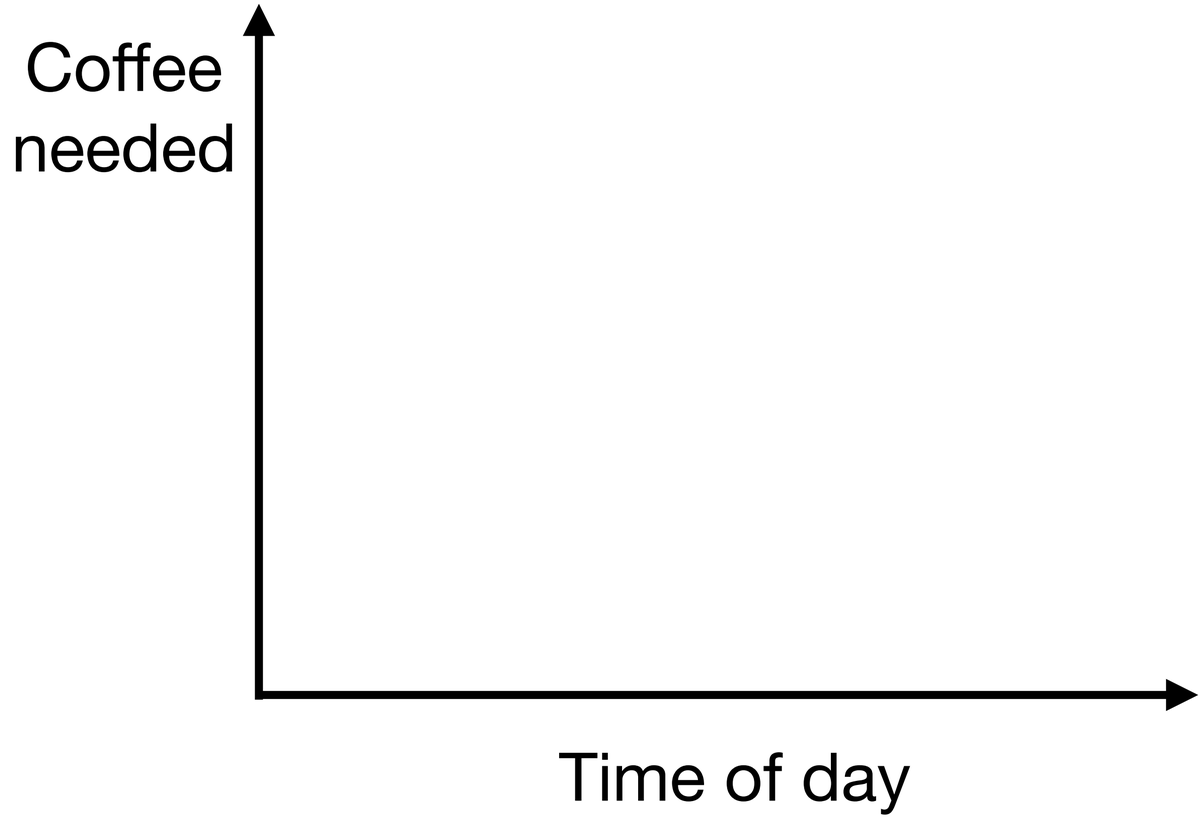

In
Sigmoid
1
0.5
0
−6
−4
−2
0
2
4
6

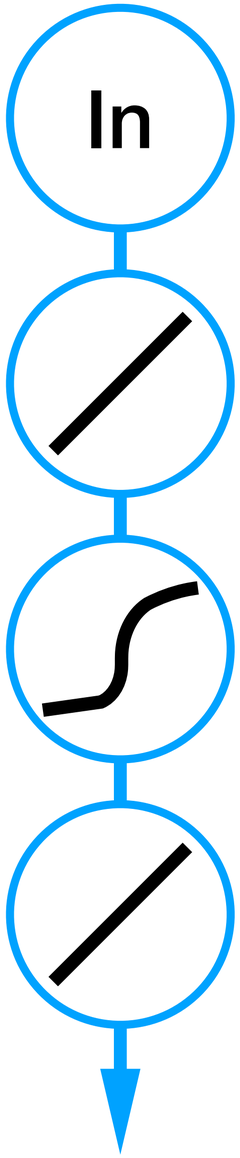

In

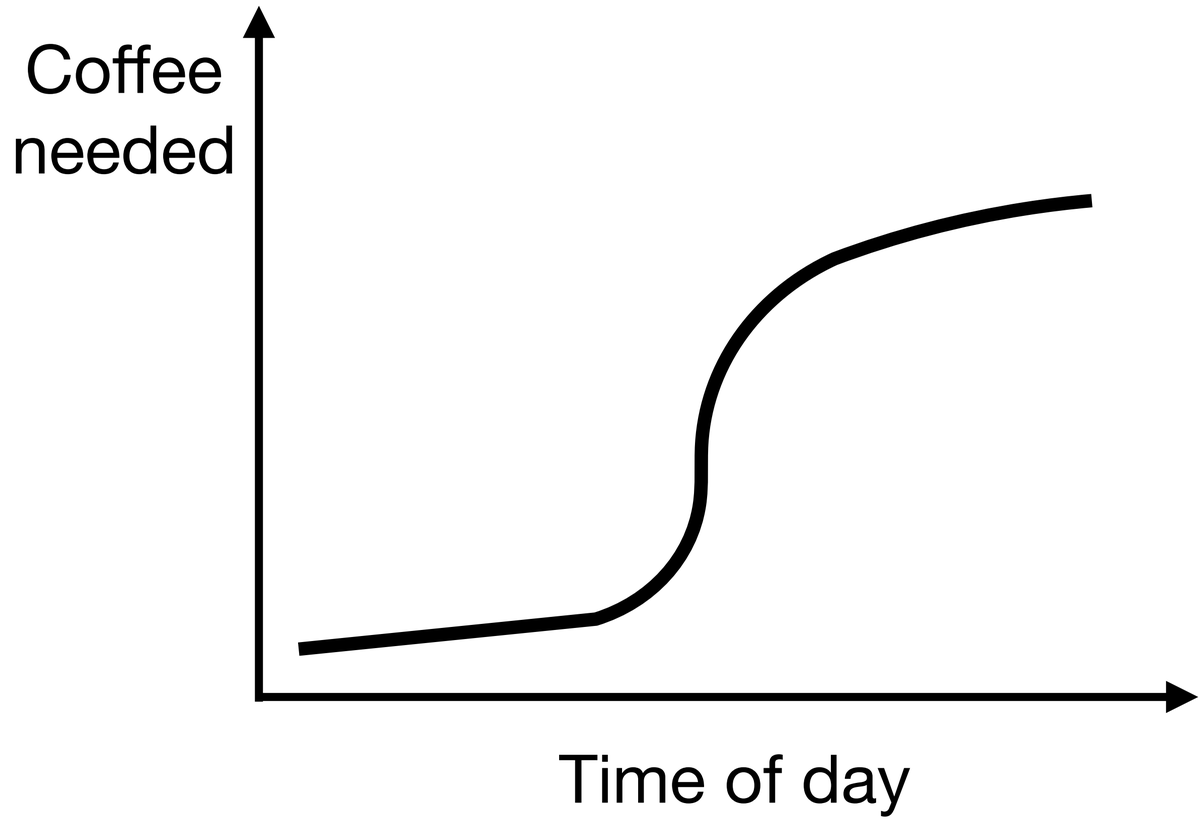

In

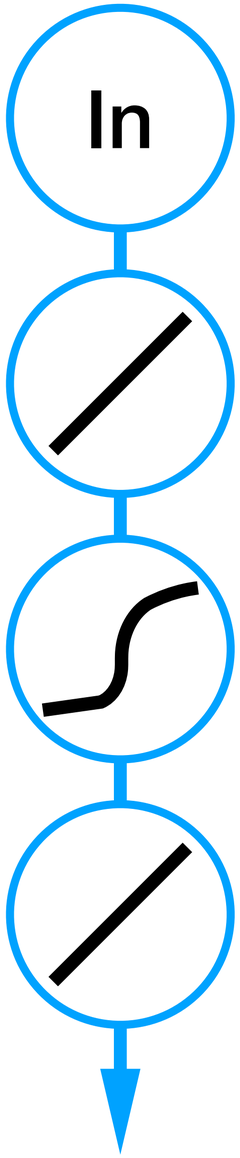

Coffee
needed
Sophisticated feature

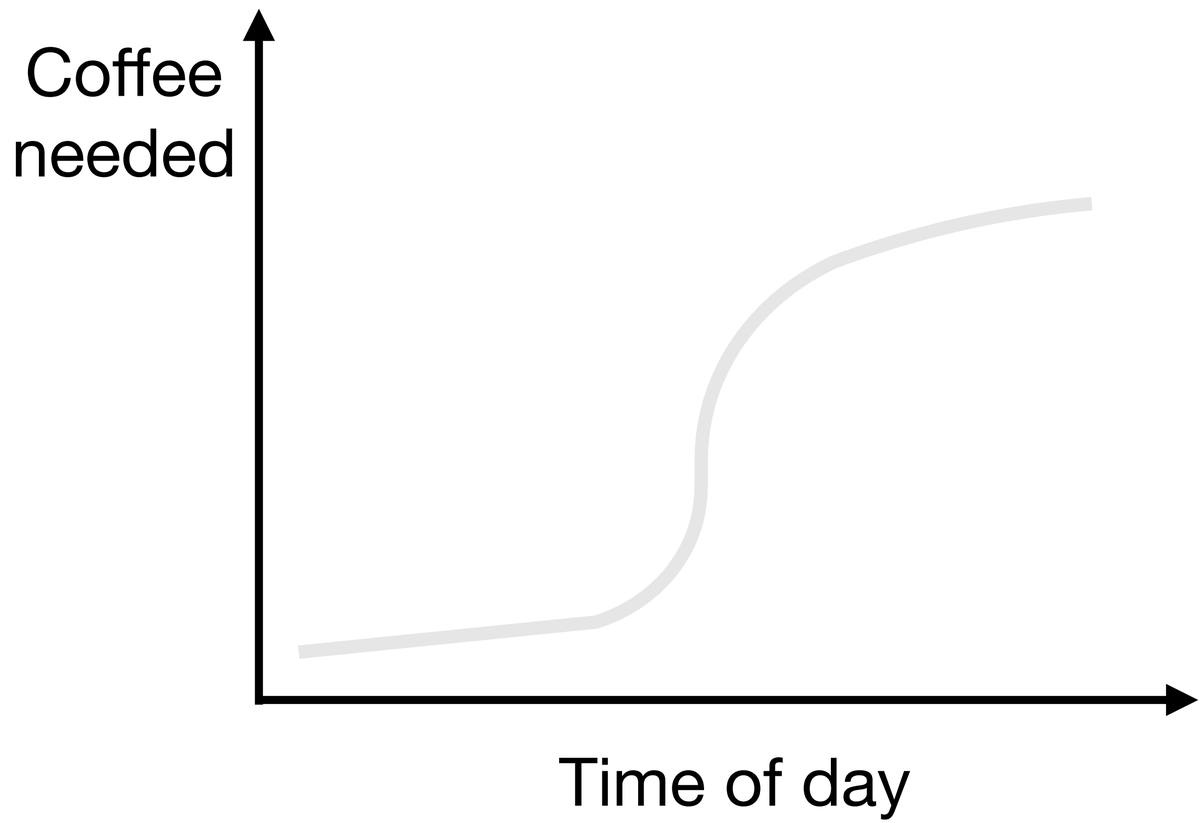

In

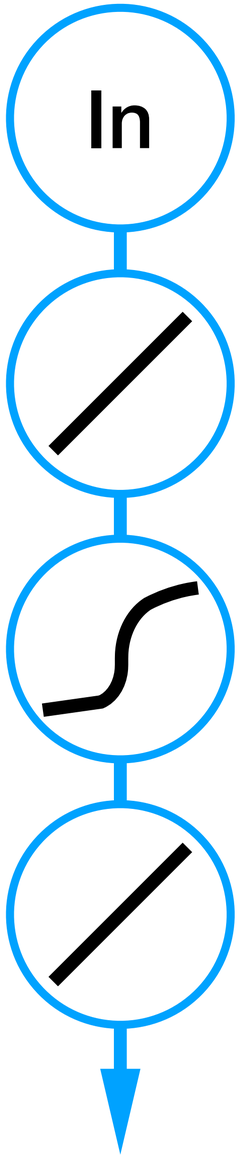

Coffee
needed
Time of day

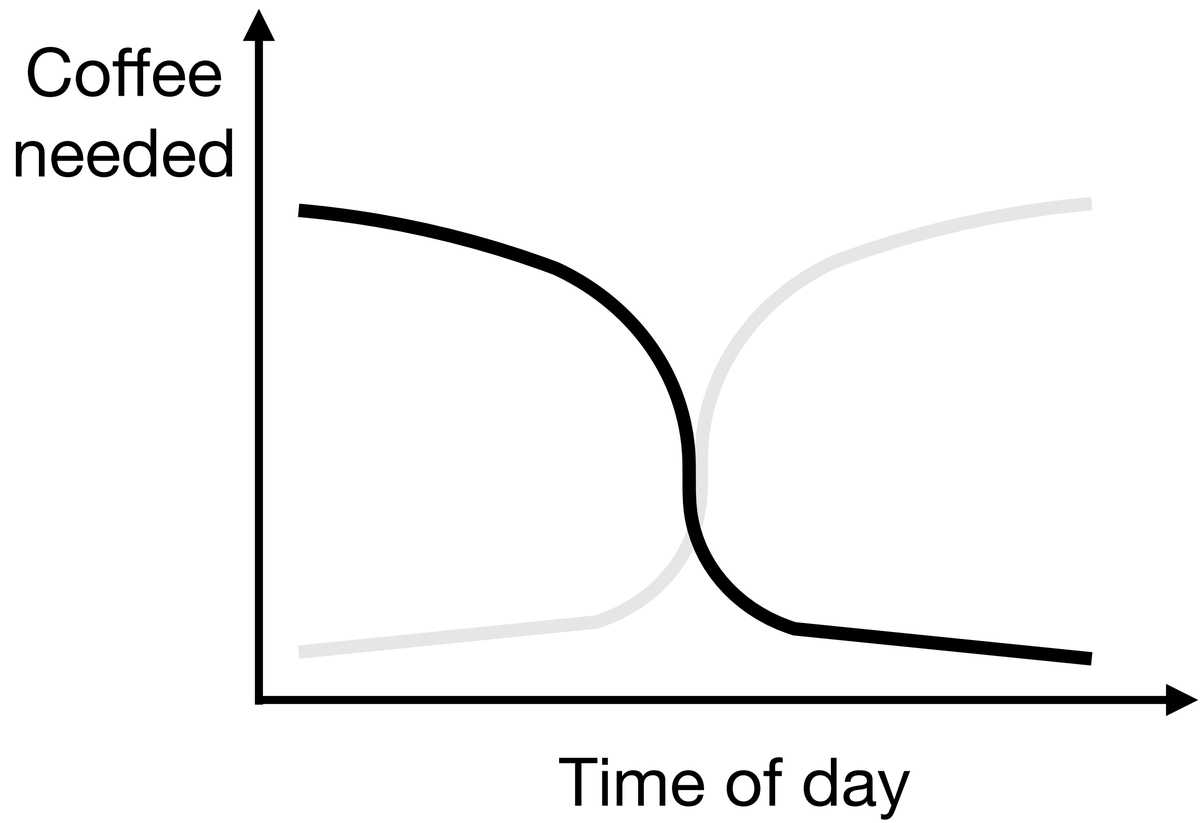

In

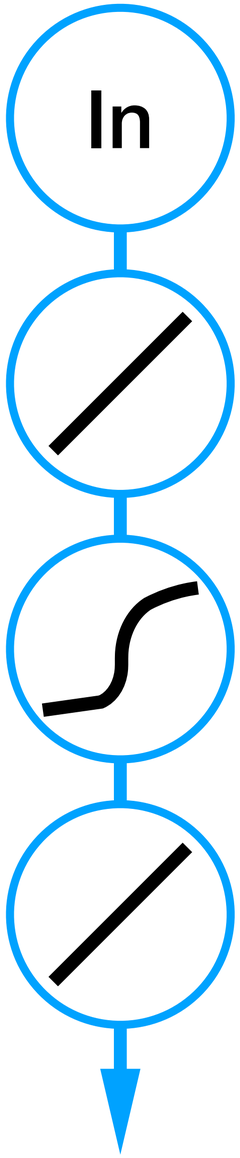

Coffee
needed
Time of day

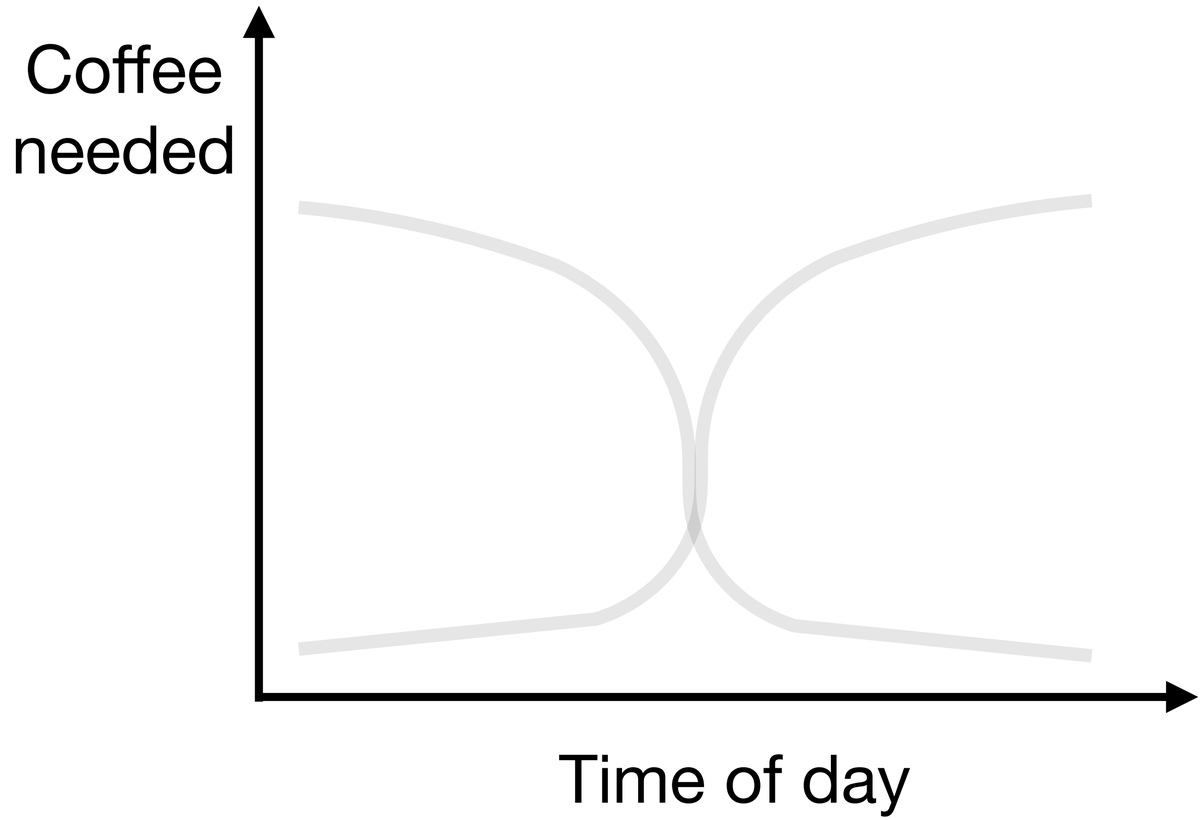

In

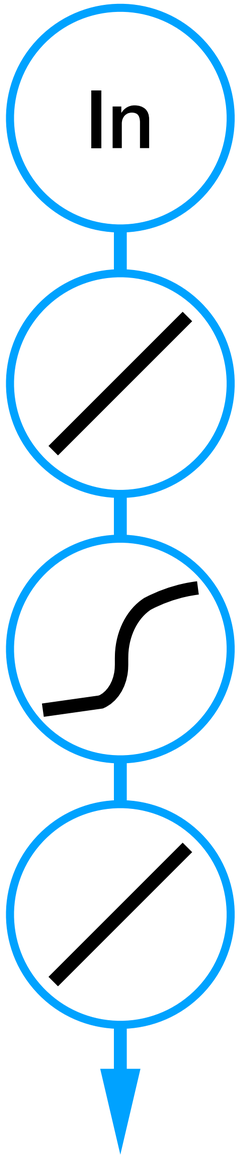

Coffee
needed
Time of day

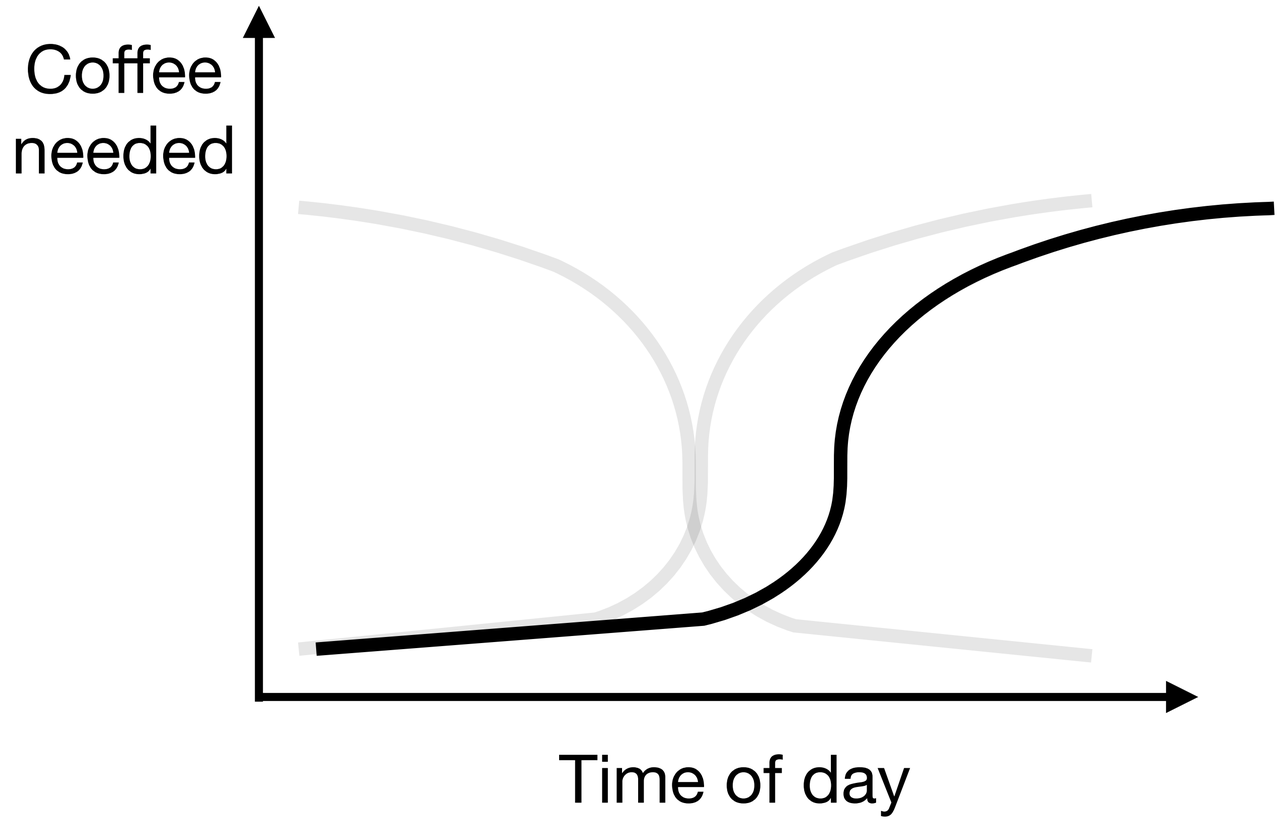

In

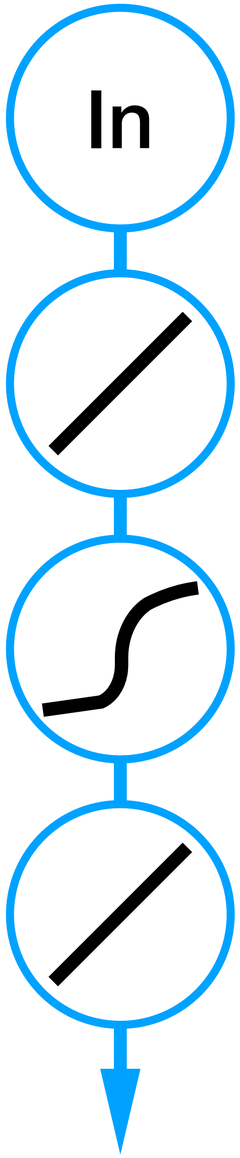

Coffee
needed
Time of day

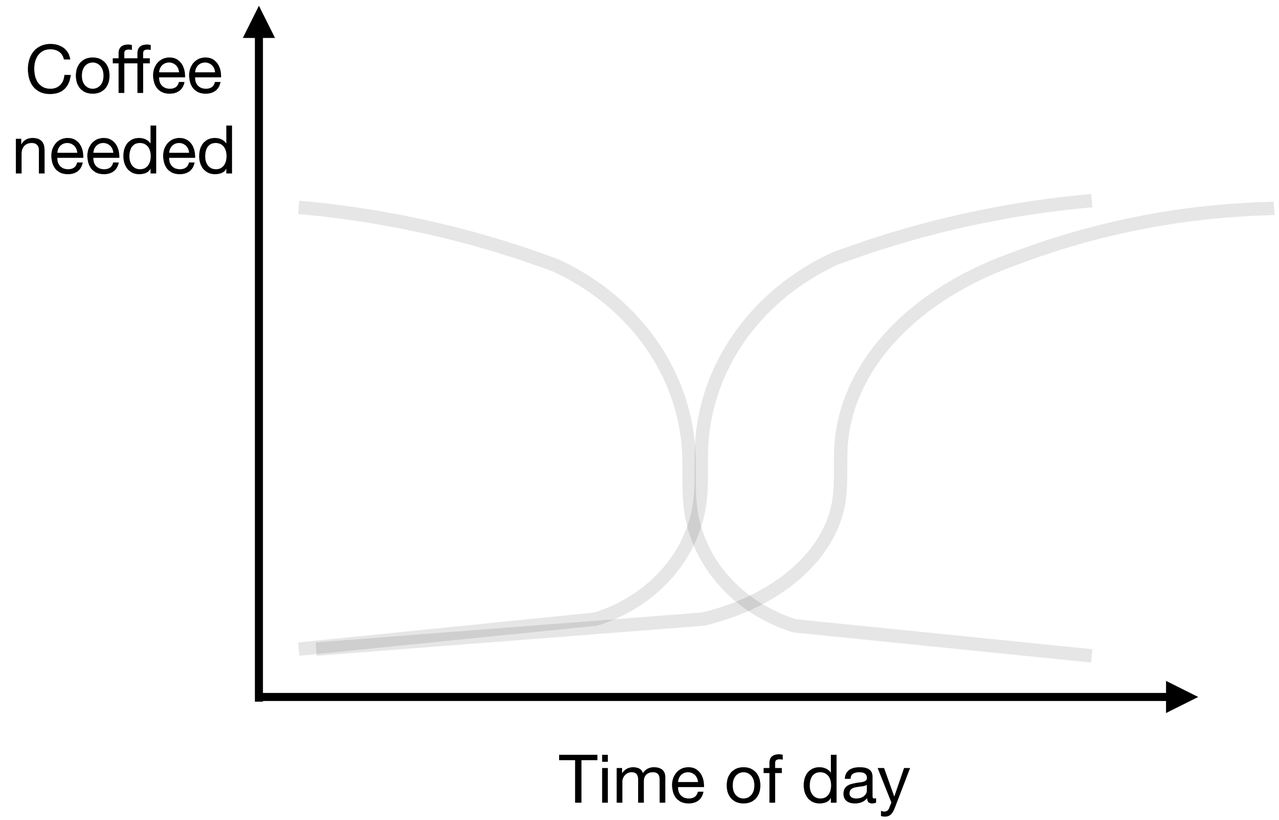

In

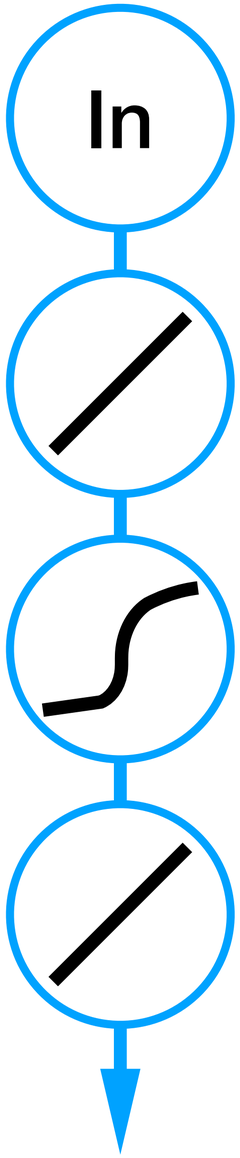

Coffee
needed
Time of day

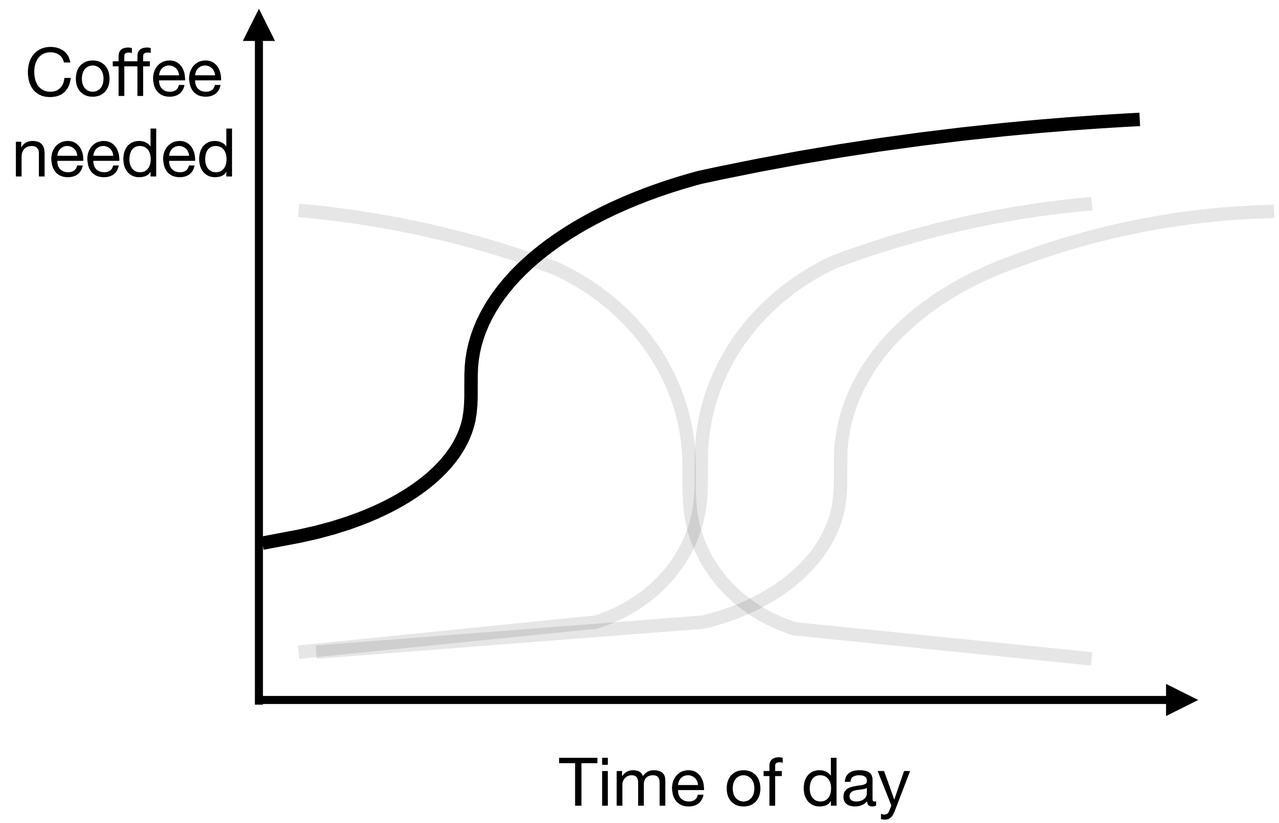

In

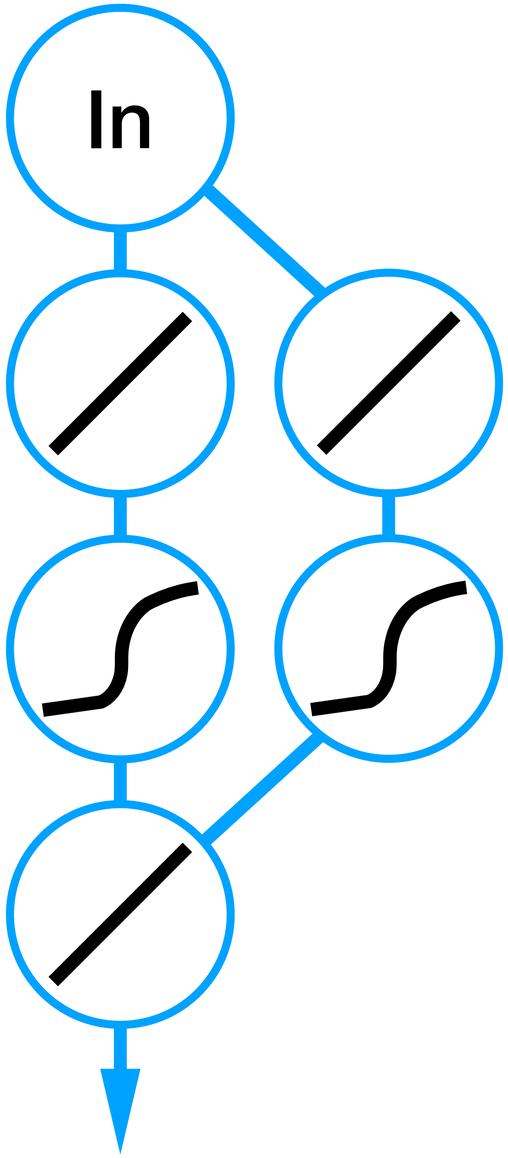

Coffee
needed
Time of day

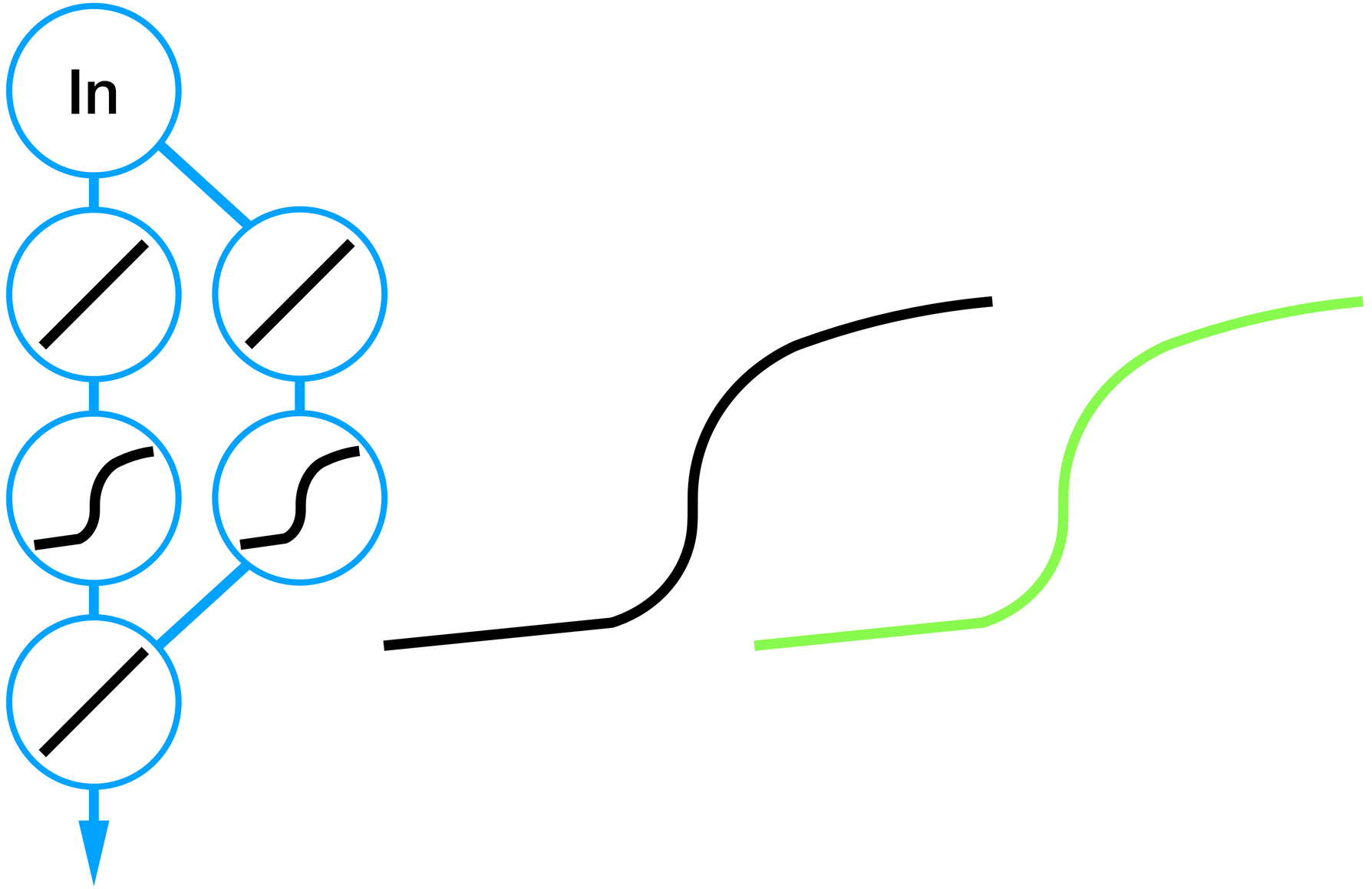

In

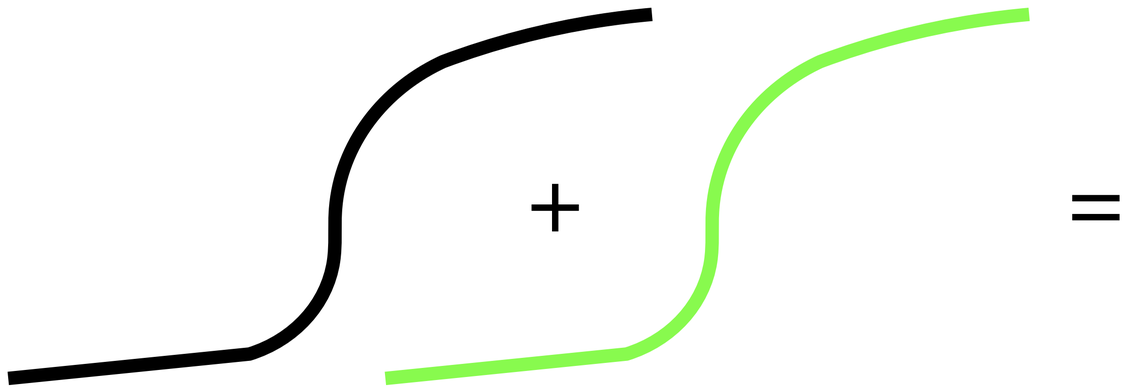

Coffee
needed
Time of day

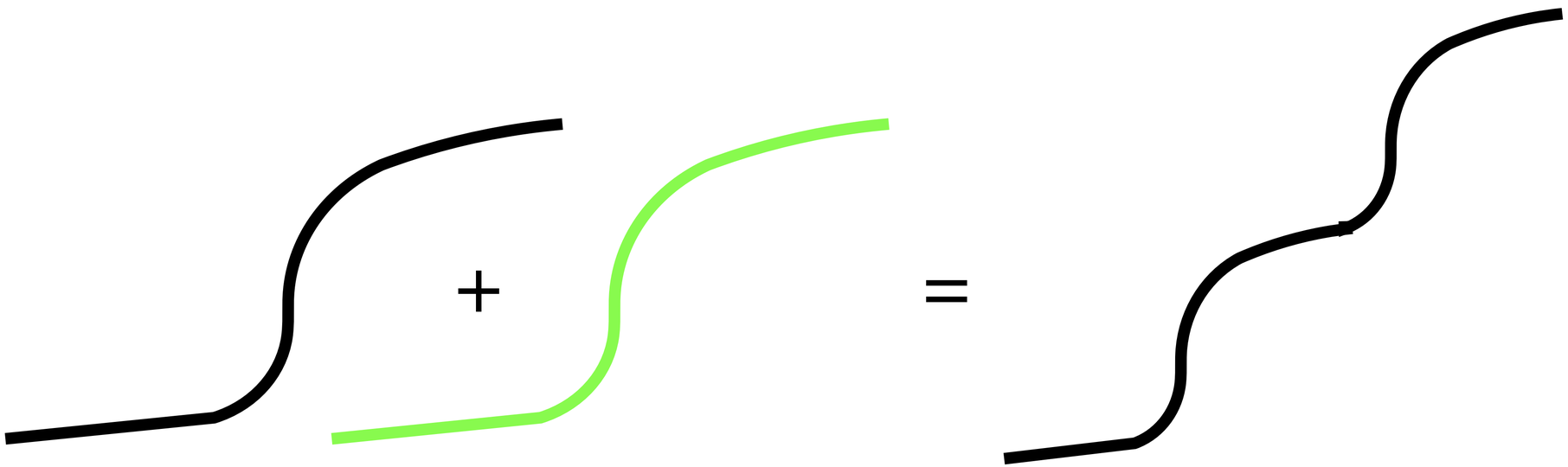

In

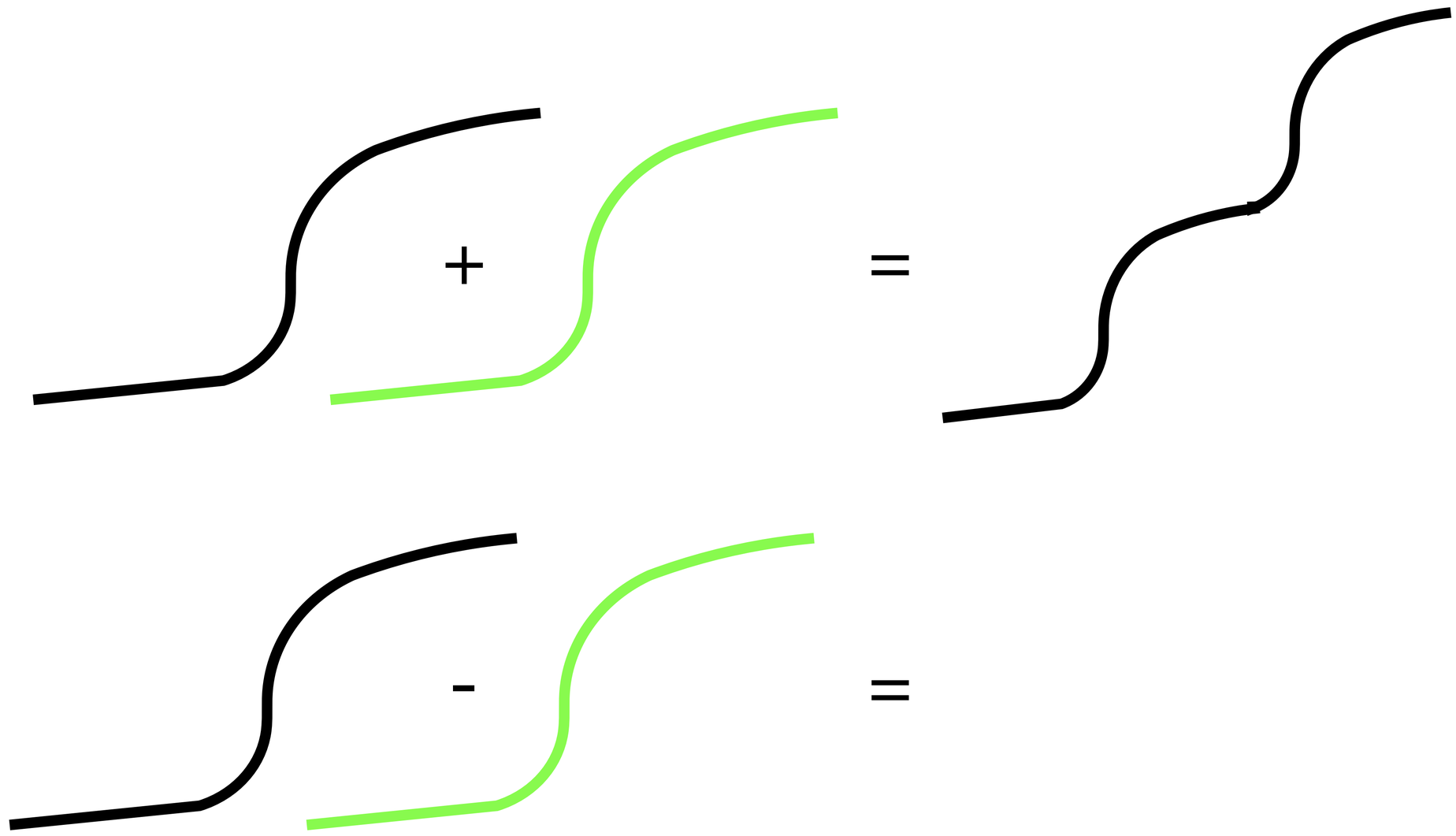

Coffee
needed
Time of day

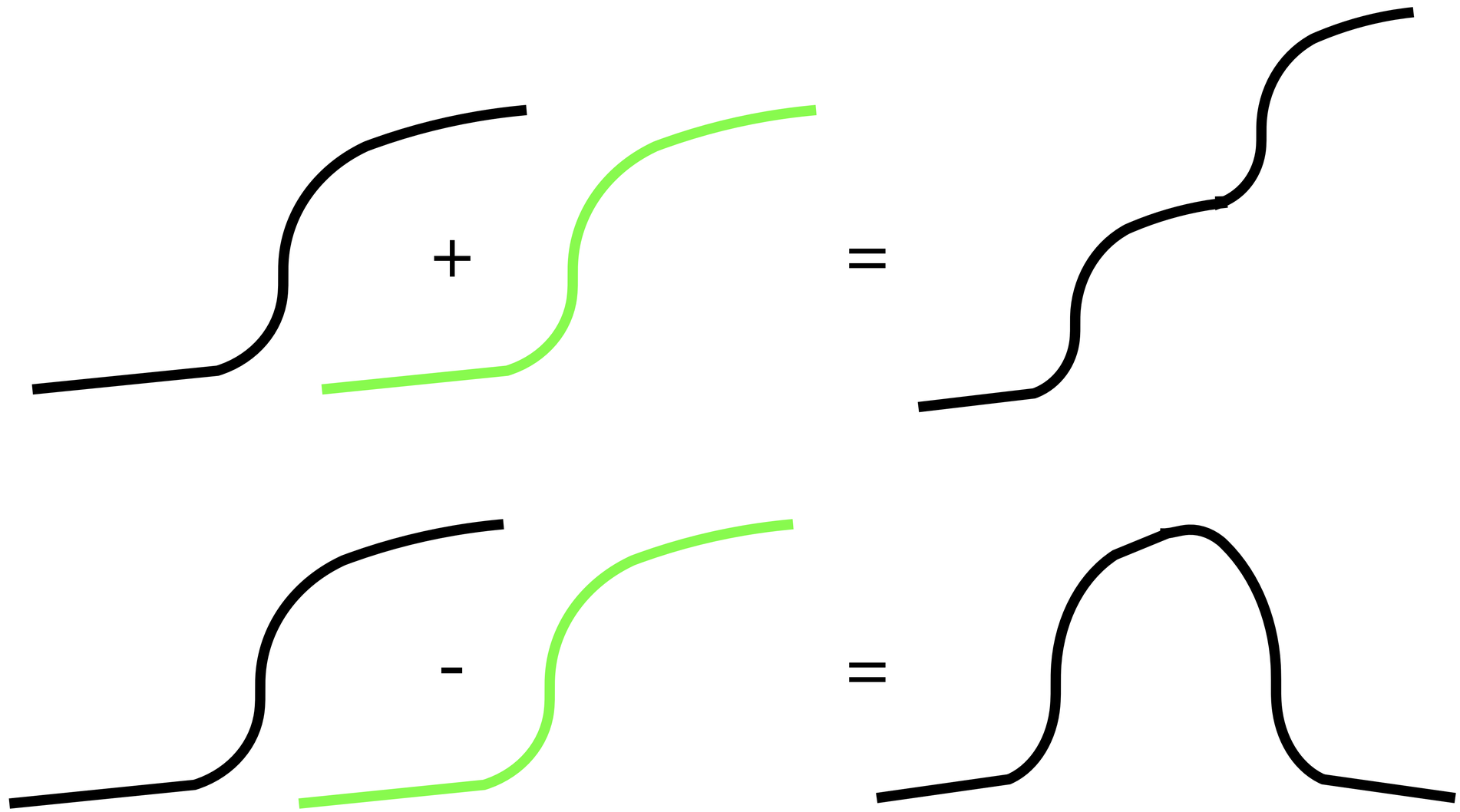

In

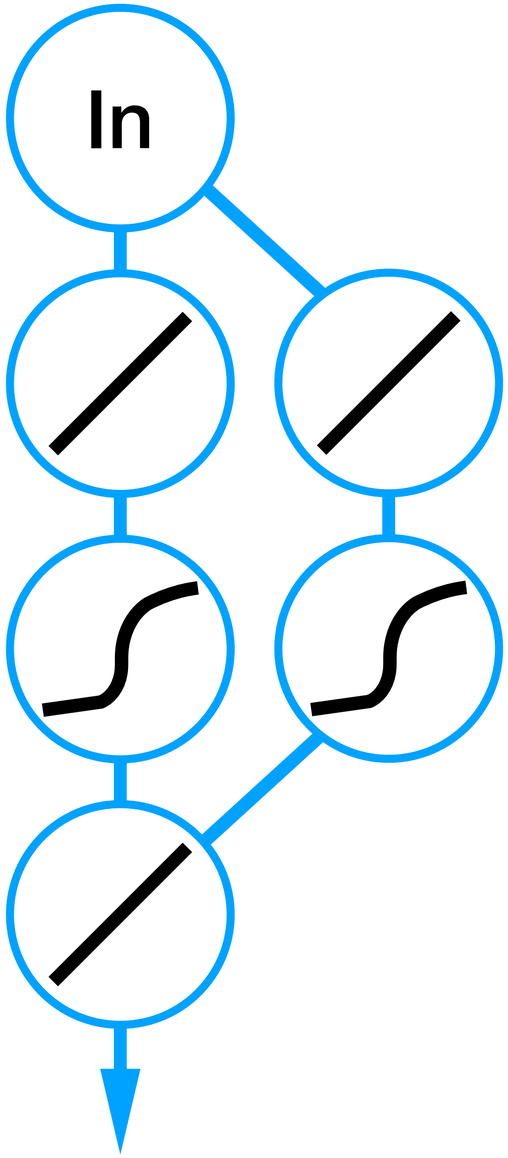

In

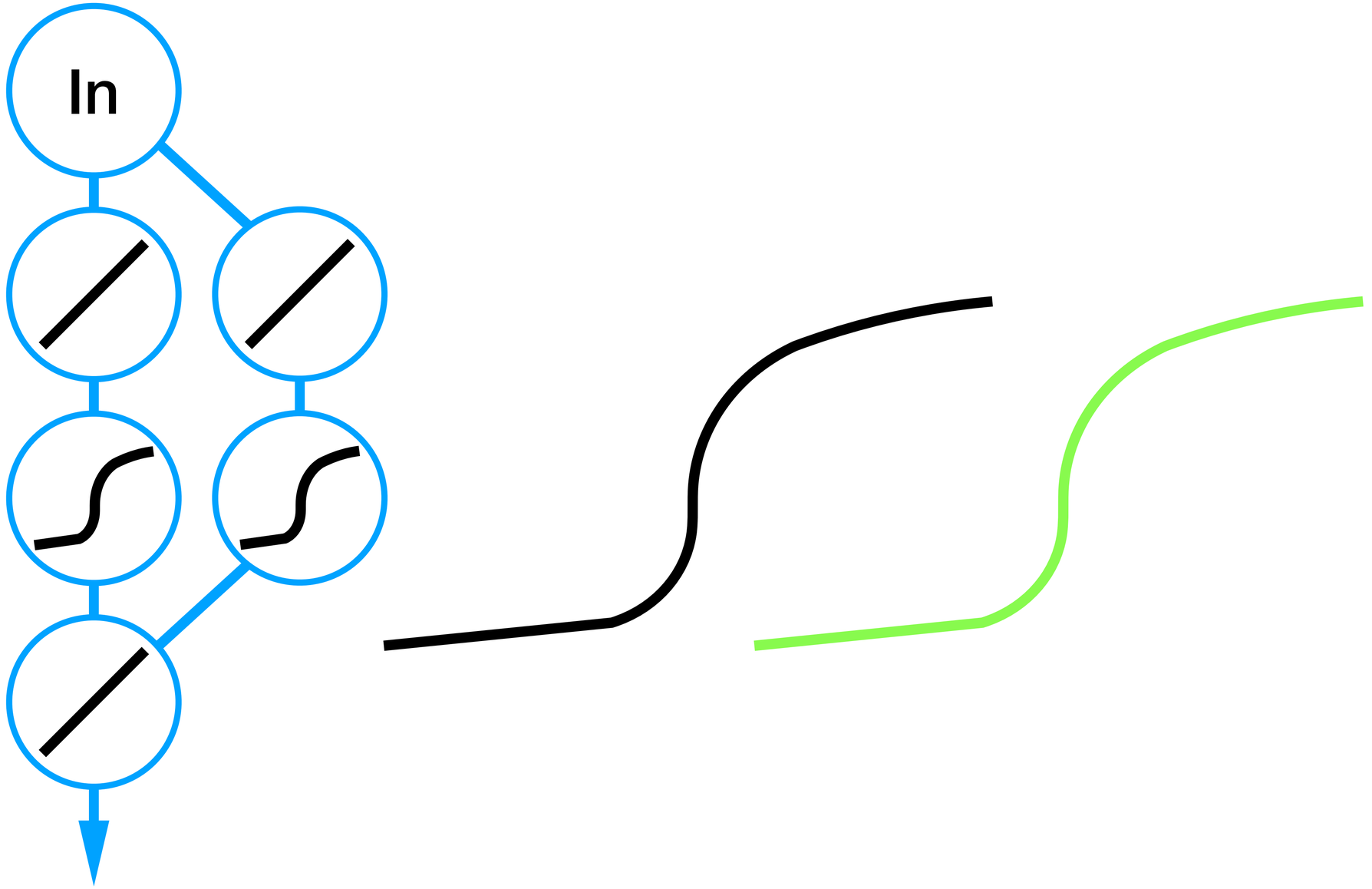


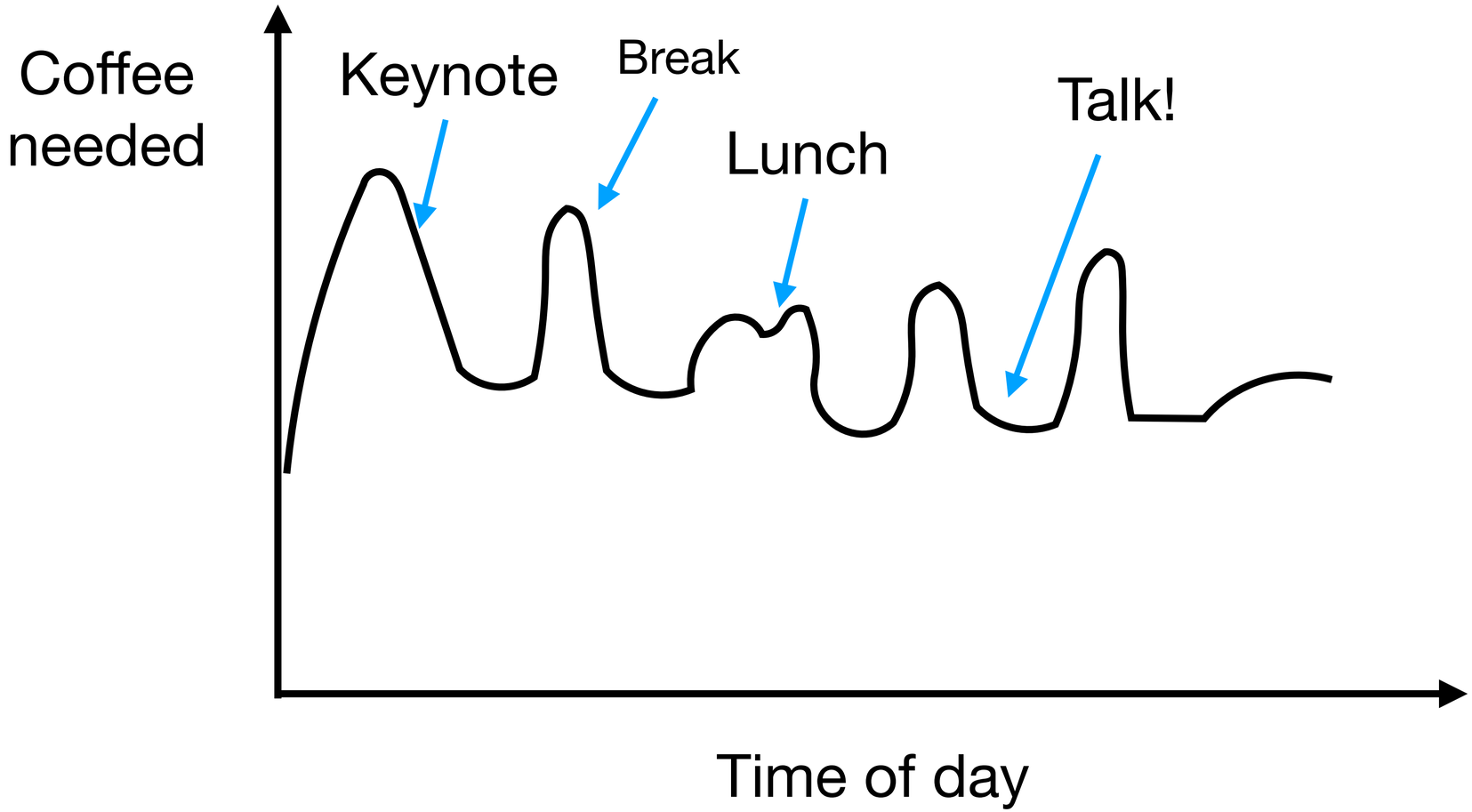


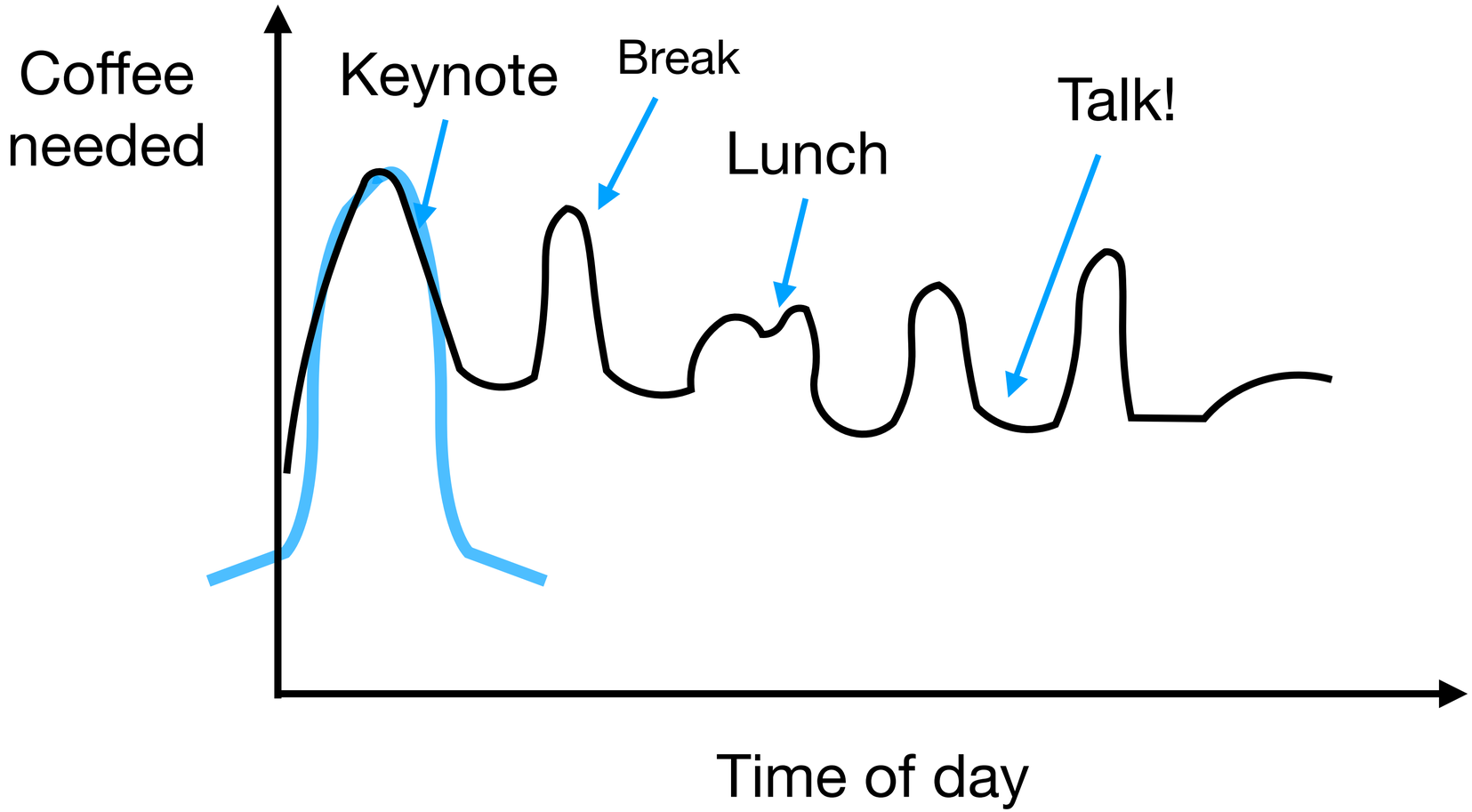


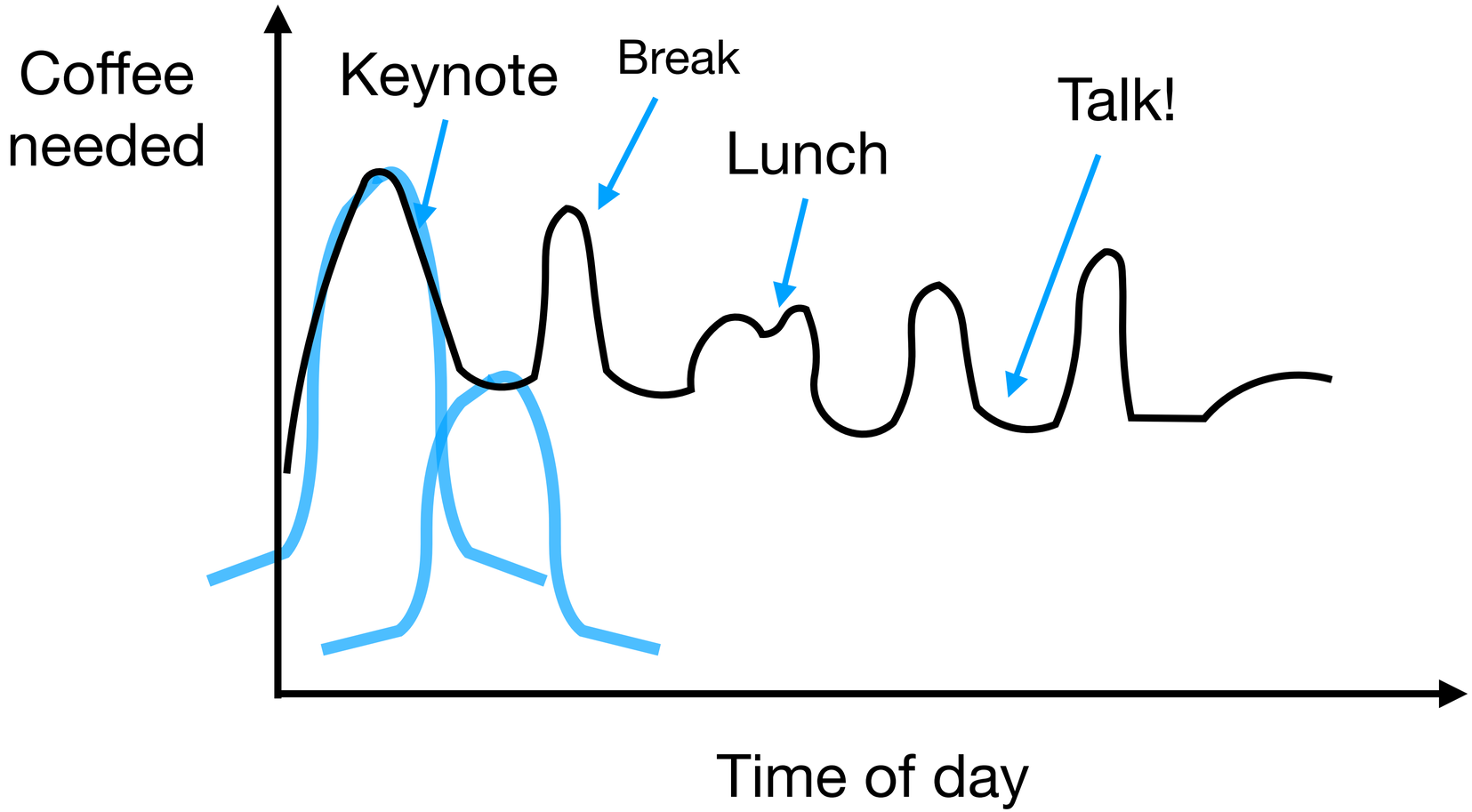


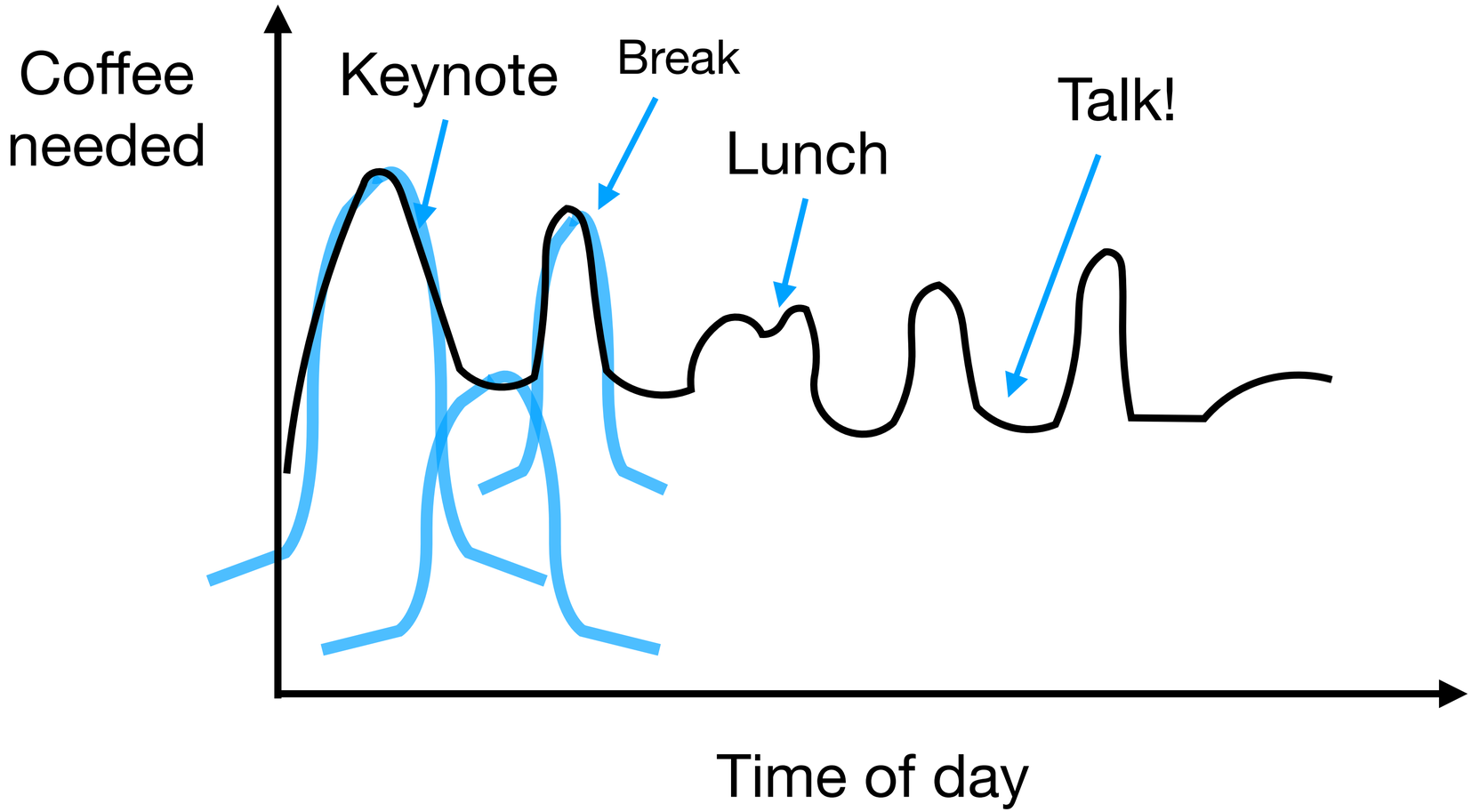

In

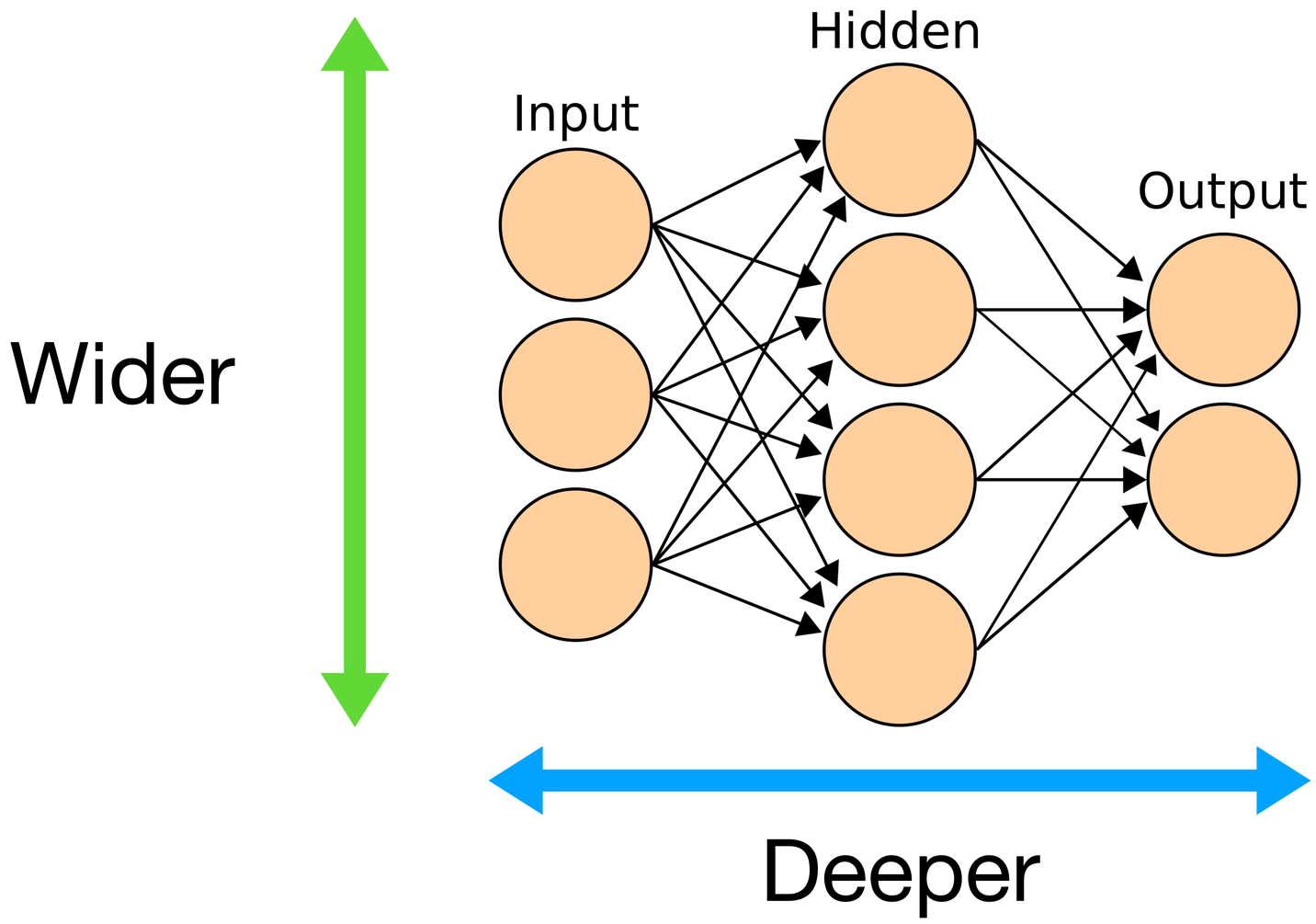

In


Coffee needed
Keynote
Break
Lunch
Talk!
Time of day


Coffee
needed
Keynote
Break
Lunch
Talk!
Time of day


Coffee needed
Keynote
Break
Lunch
Talk!
Time of day

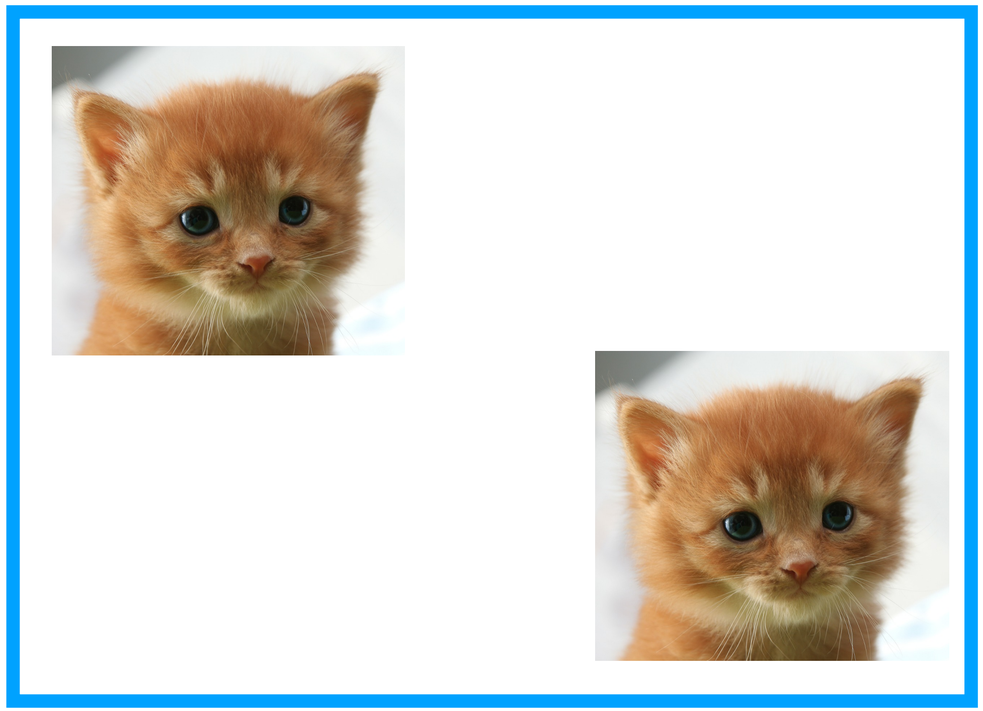

Coffee needed
Keynote
Break
Lunch
Talk!
Time of day


Wider
Input
Hidden
Output
Deeper

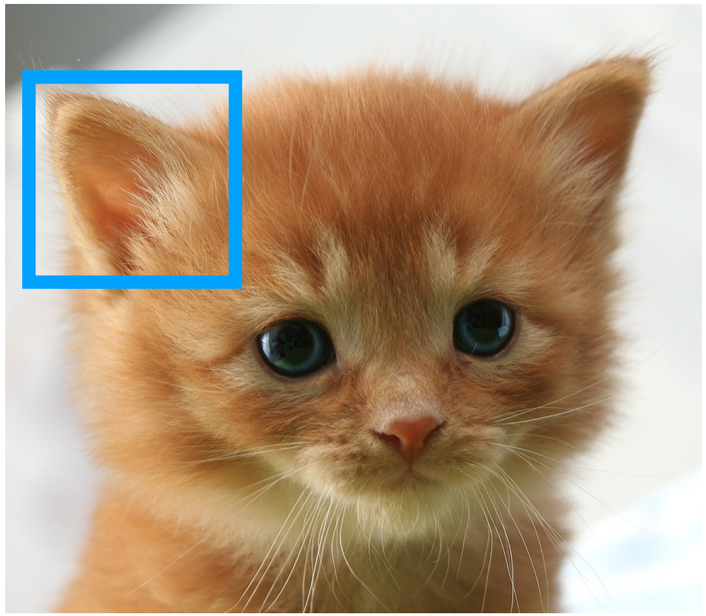
Foundational for CNNs and RNNs

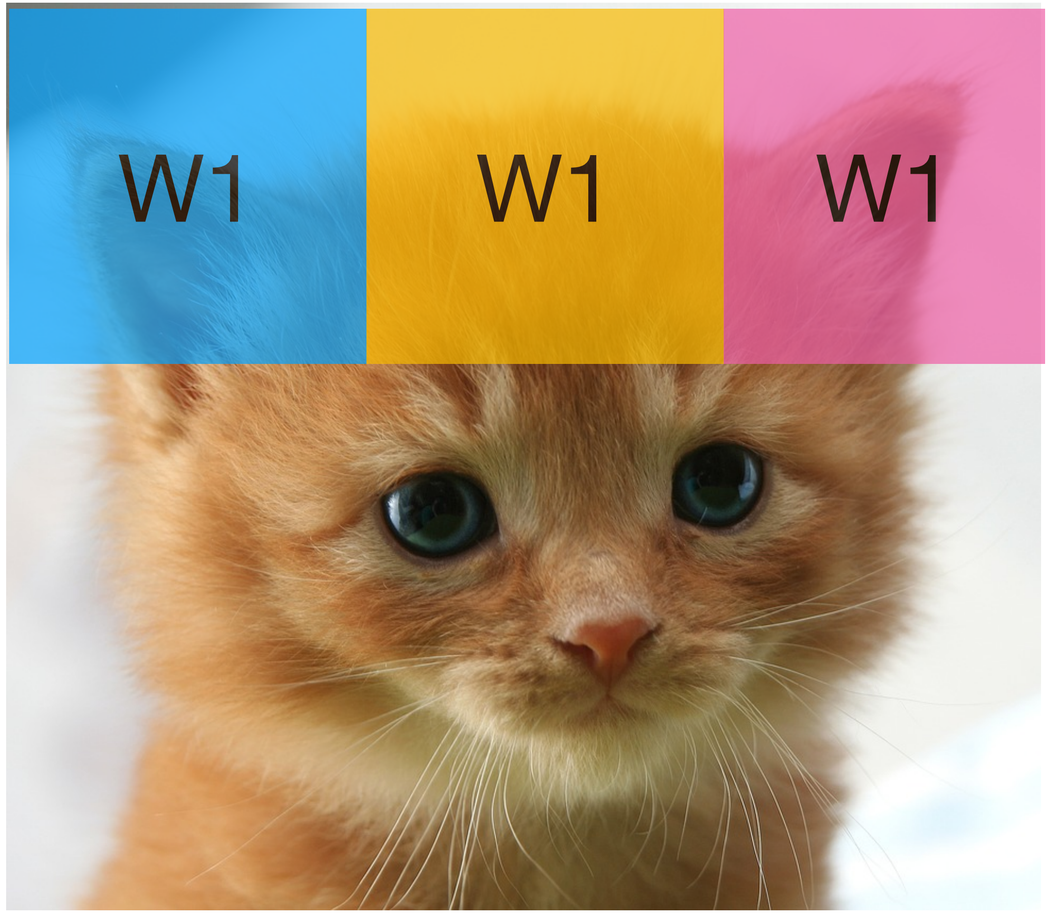
Convolutional layers

1. Translation invariance

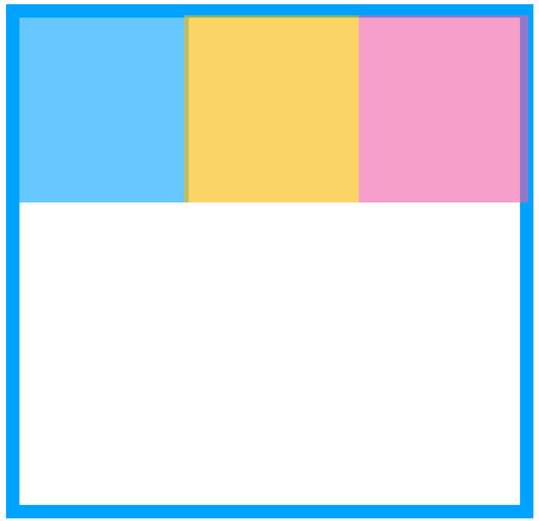


2. Locality

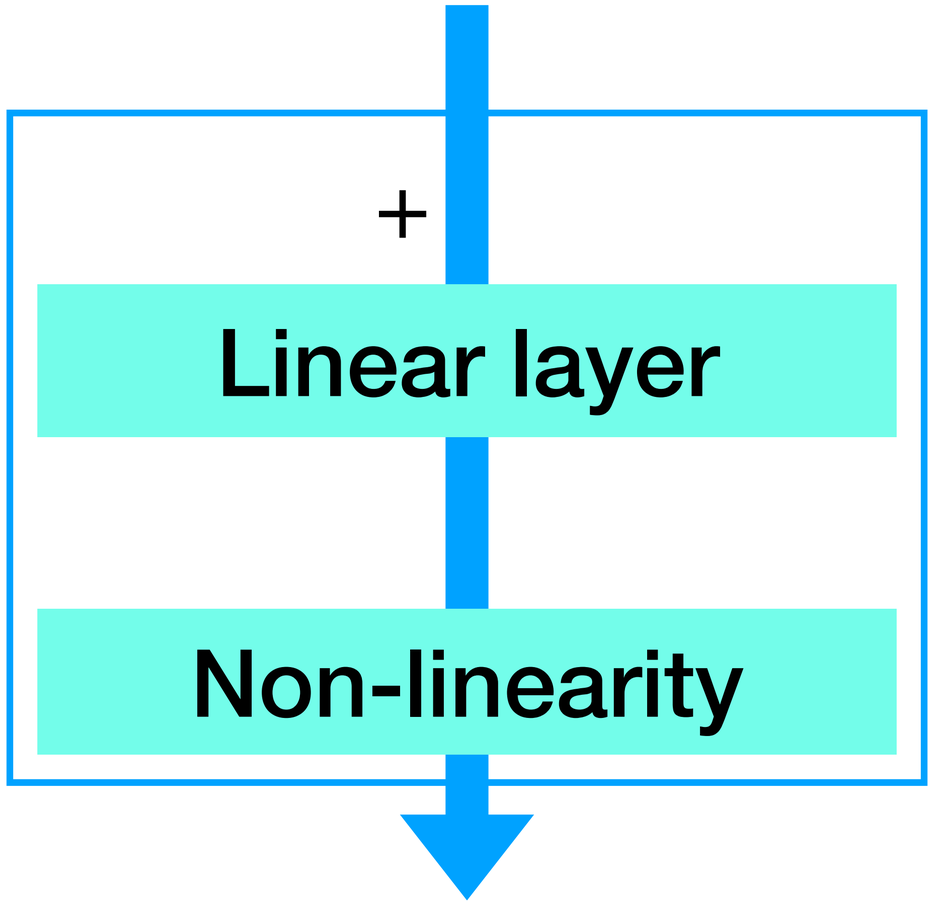




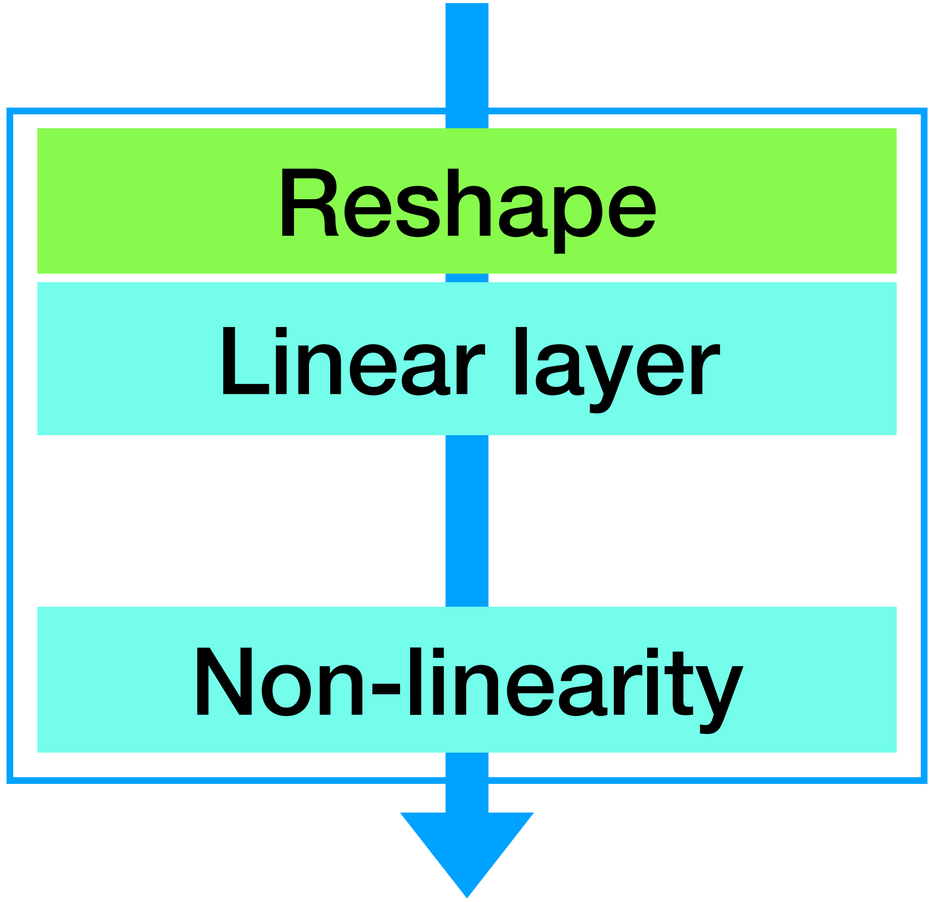
Output



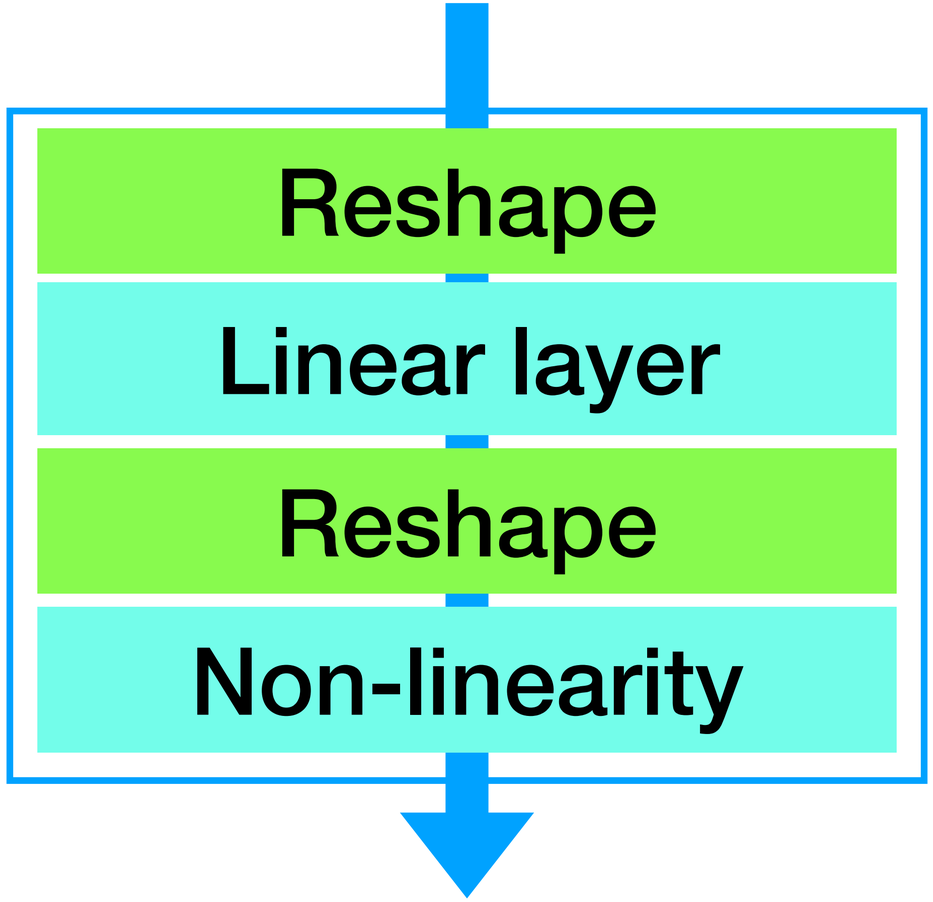
Convolutional layers



Convolutional layers

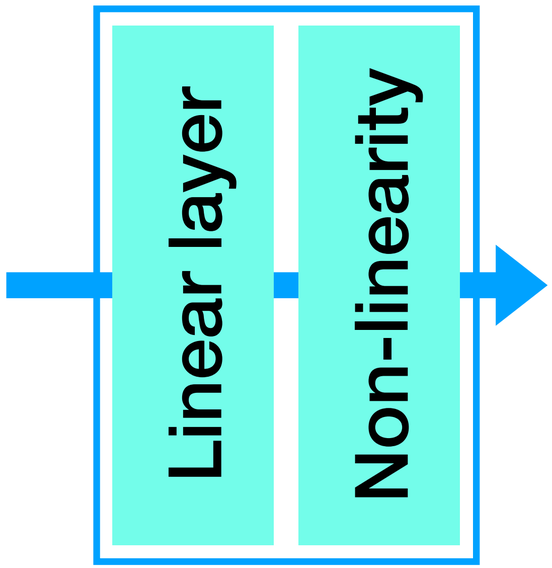


Convolutional layers

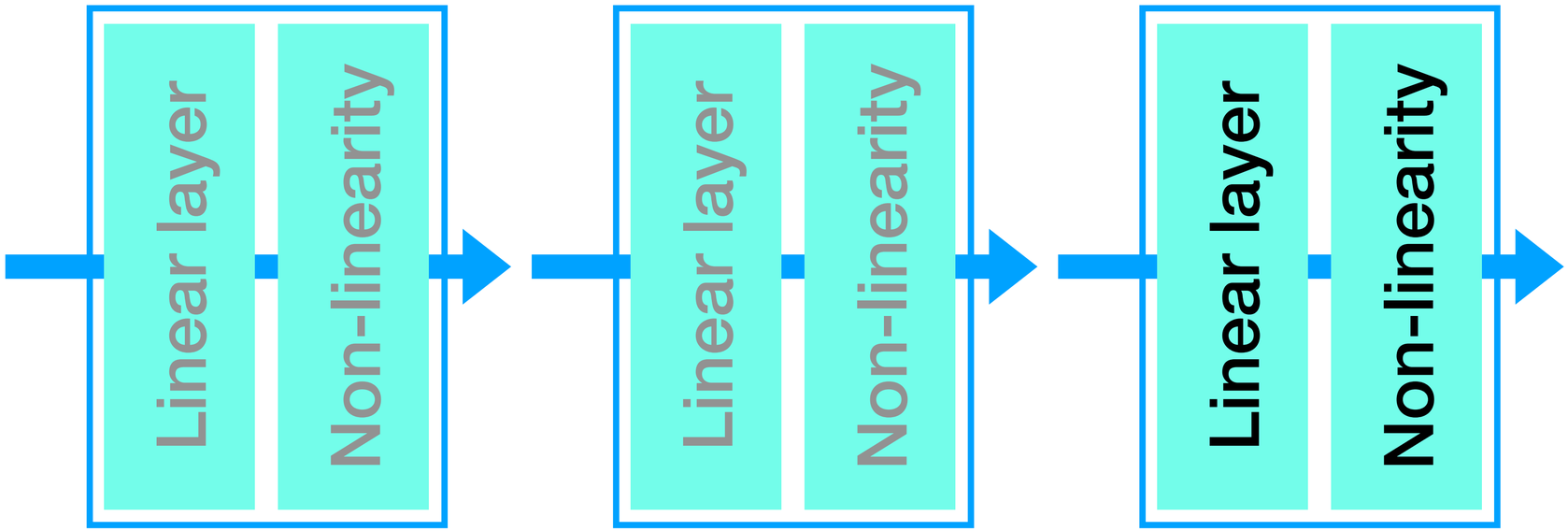


Recurrent layers

Recurrent layers

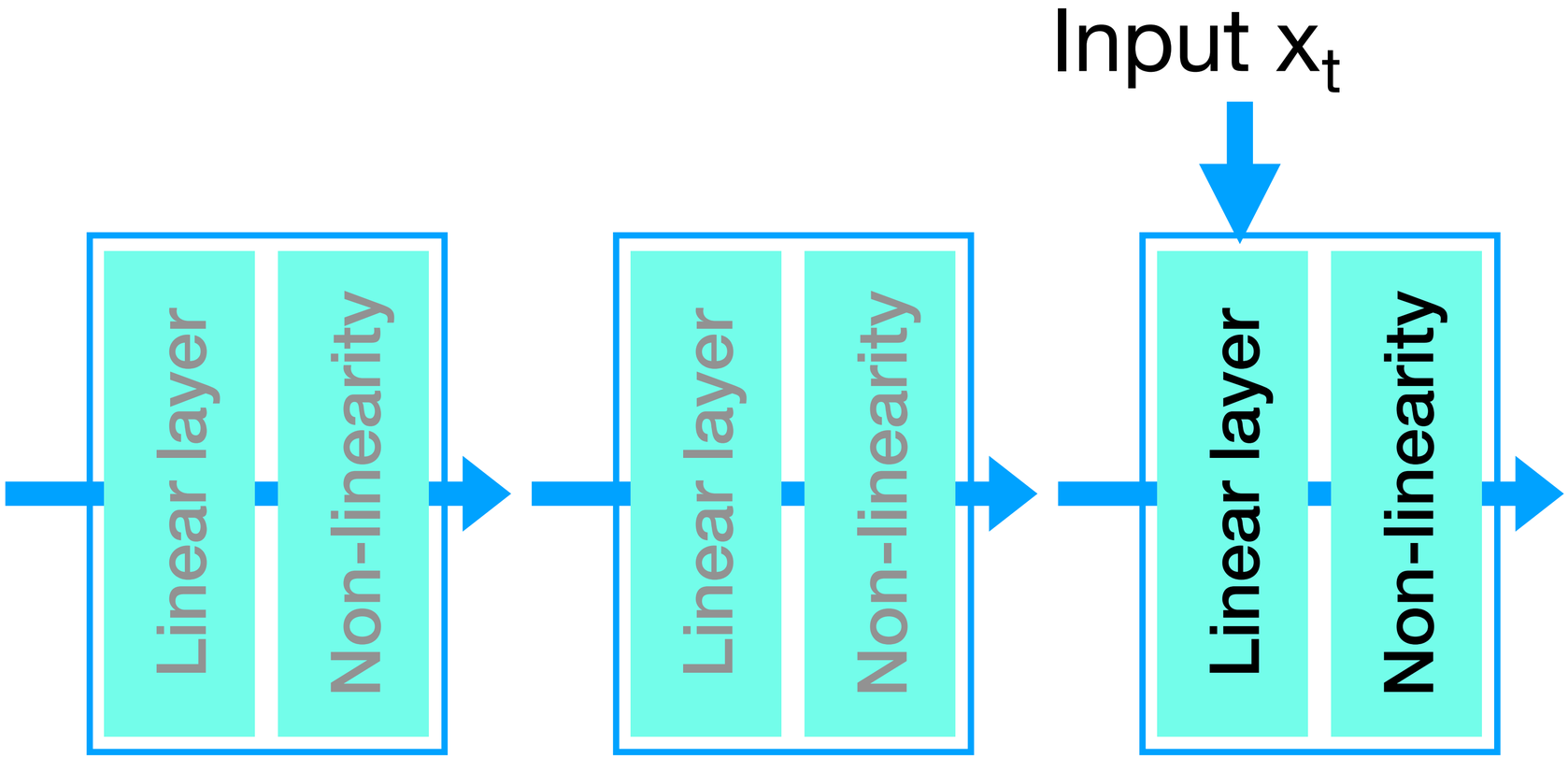


Recurrent layers

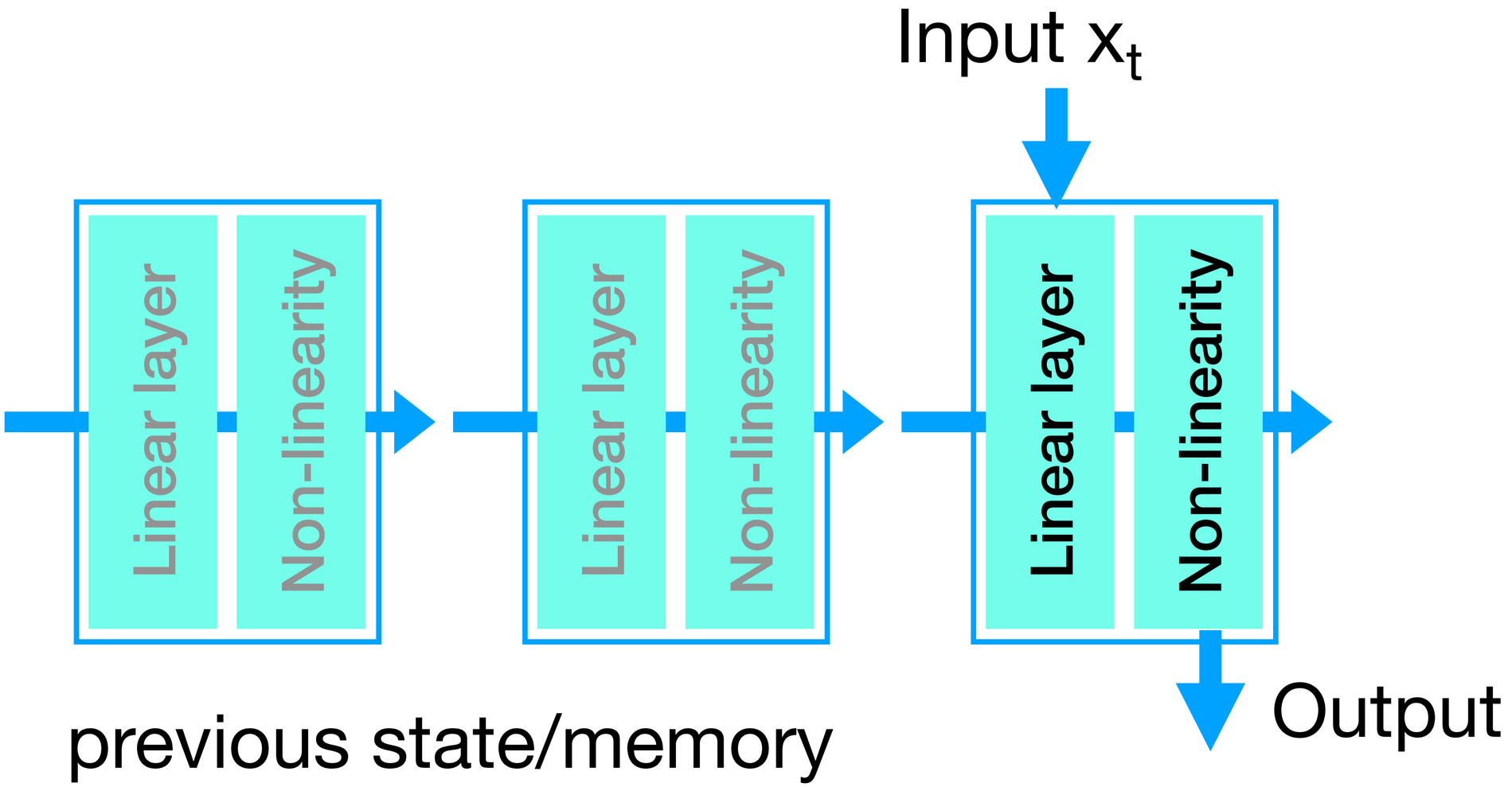


previous state/memory

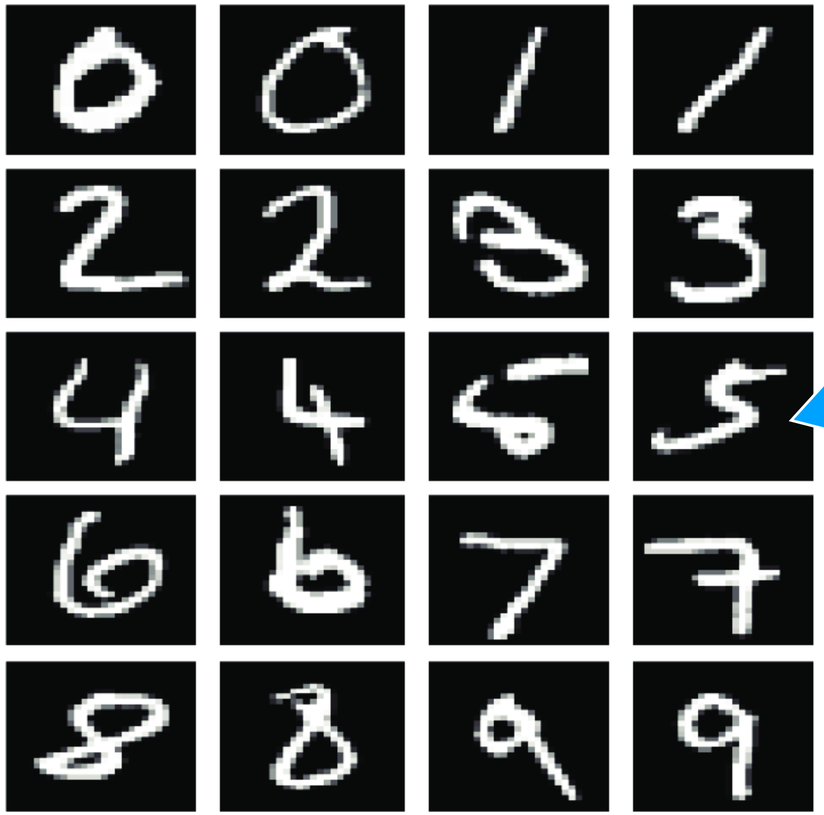
Recurrent layers



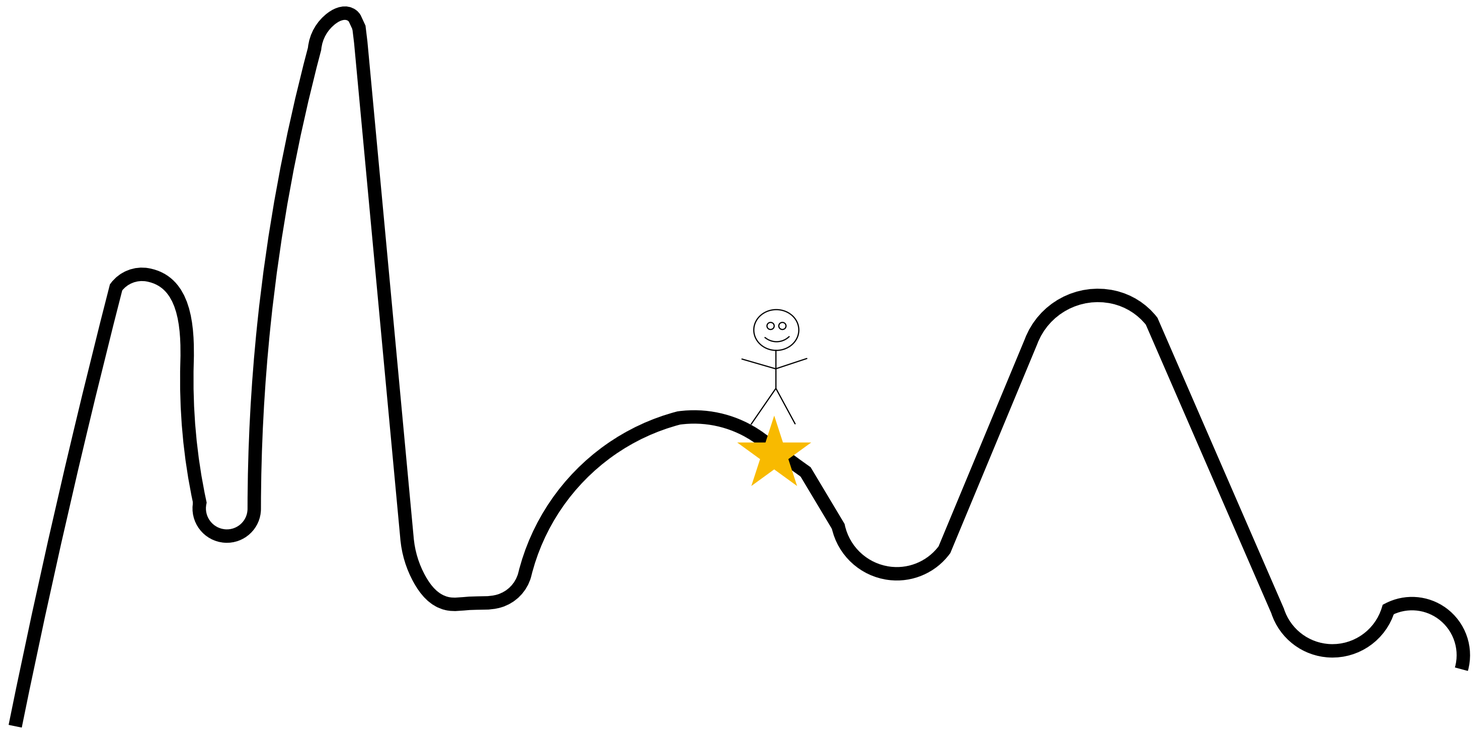
Recurrent layers

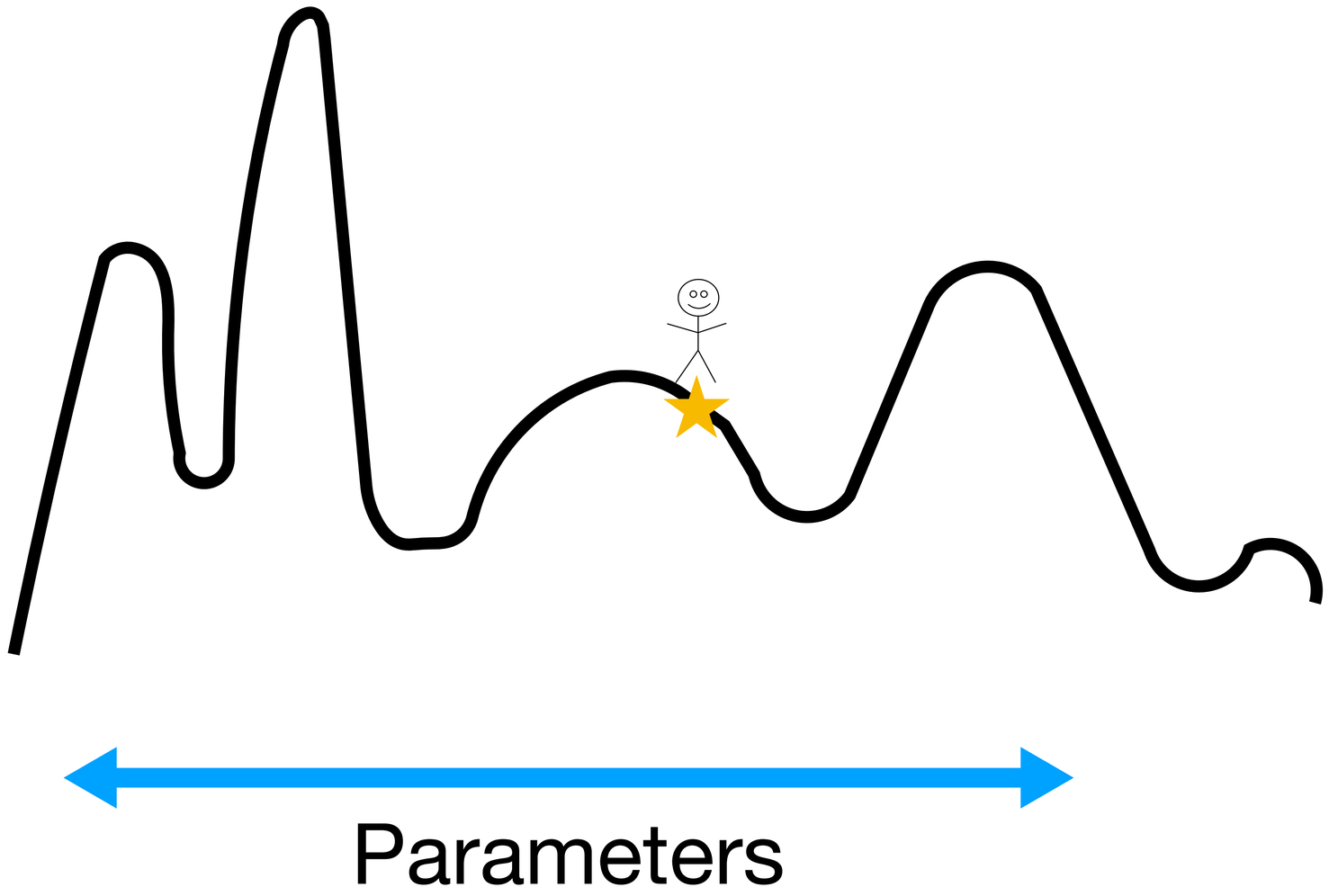


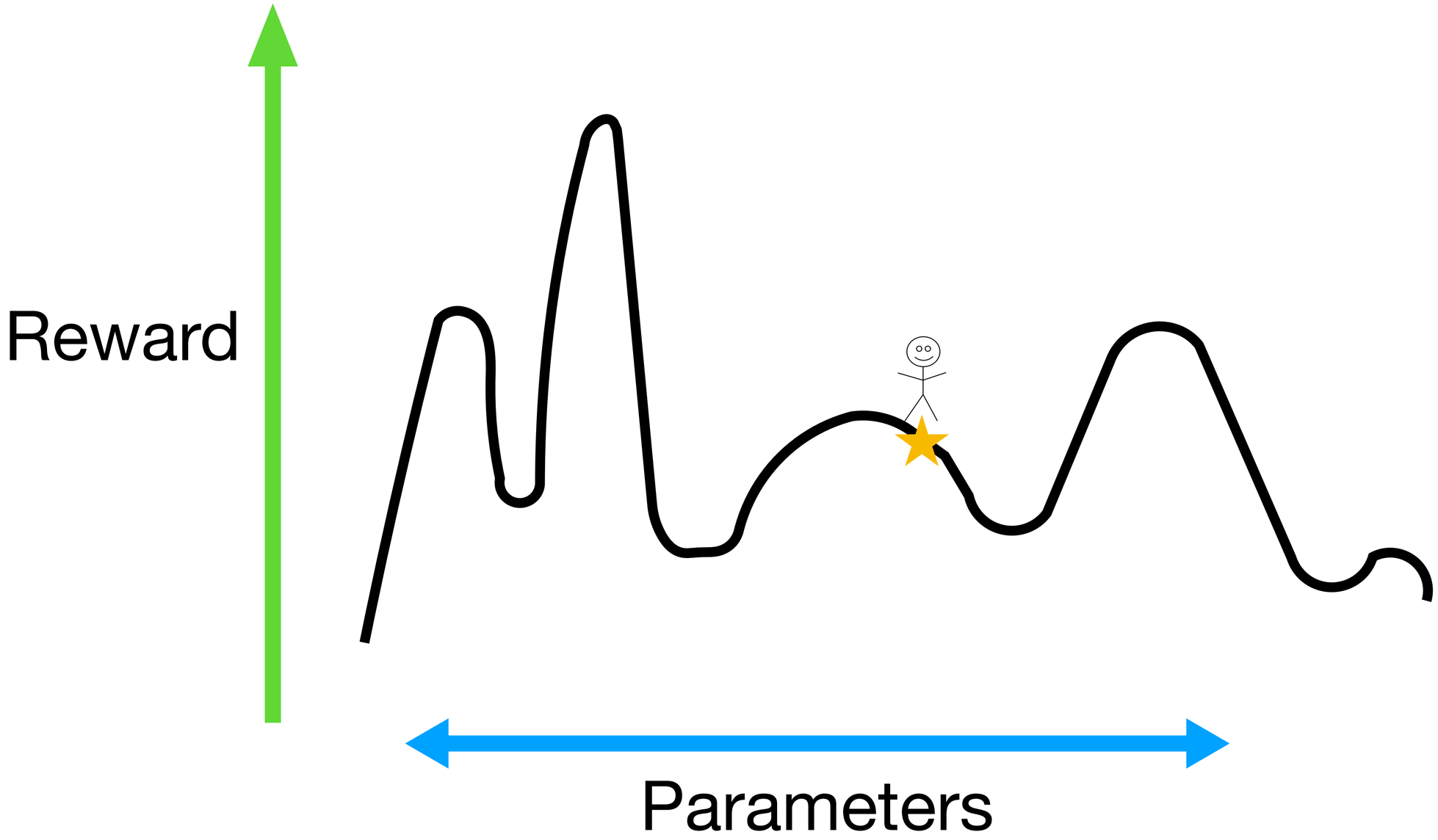
II. How train?

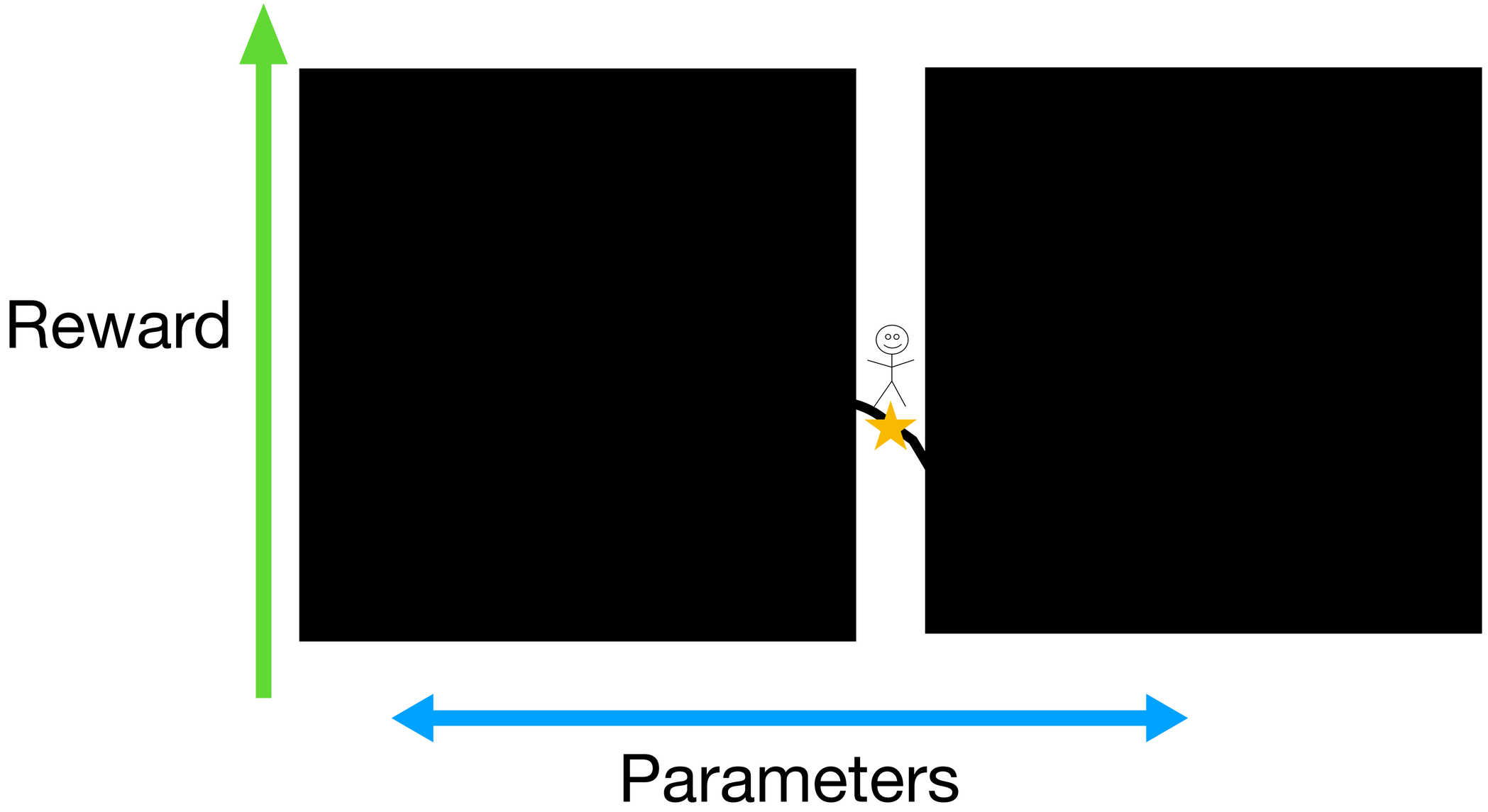


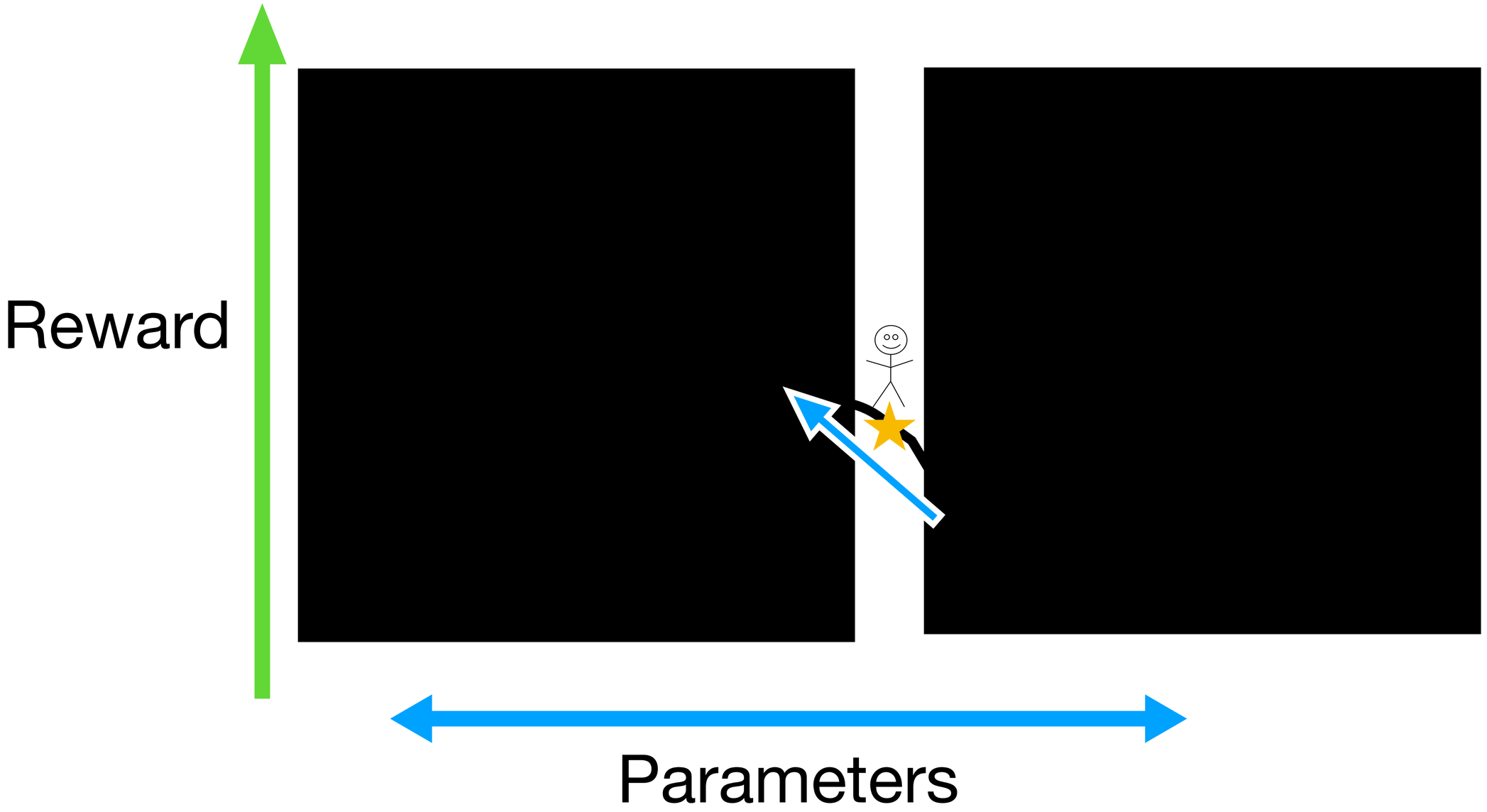
Metric:
~Prediction error

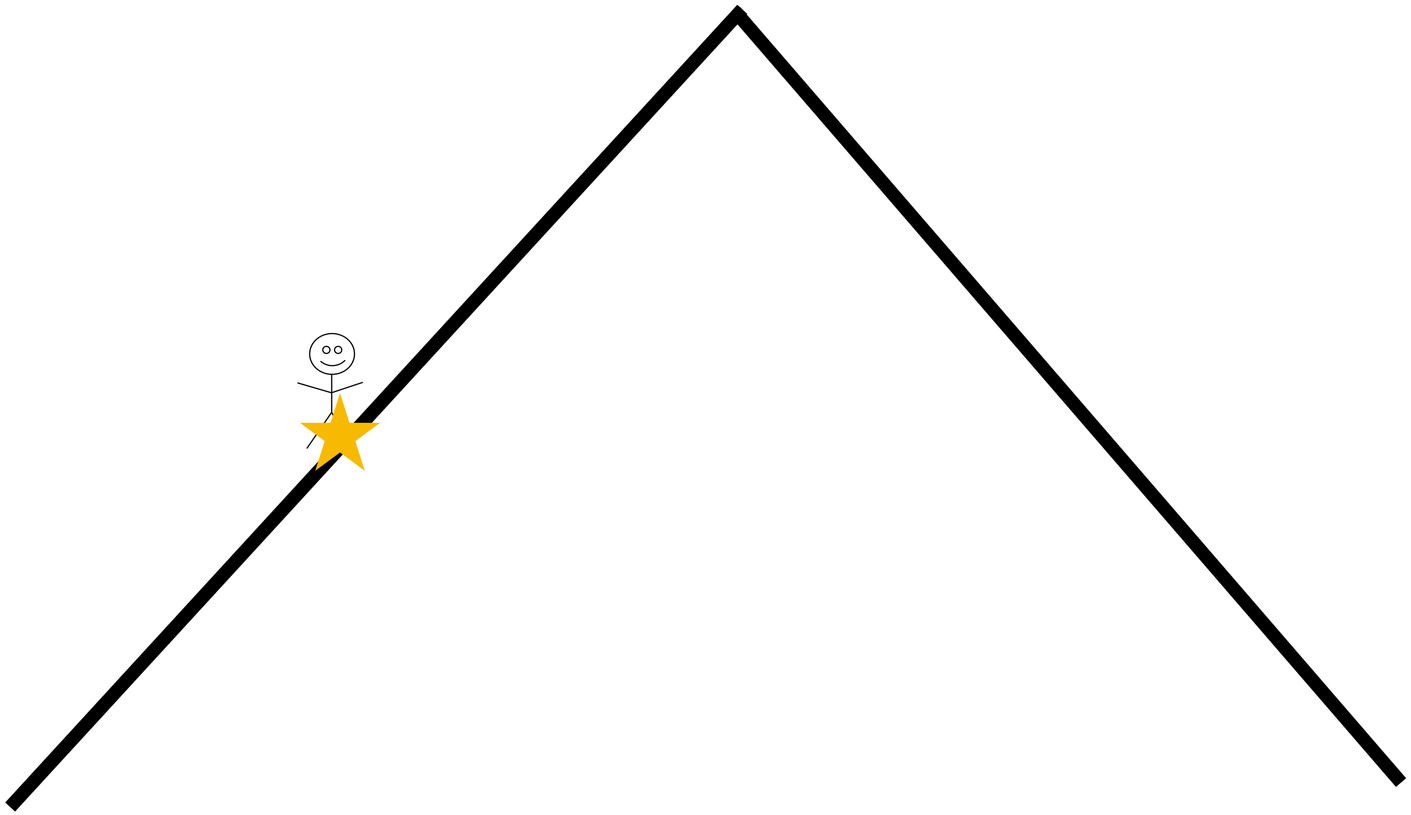


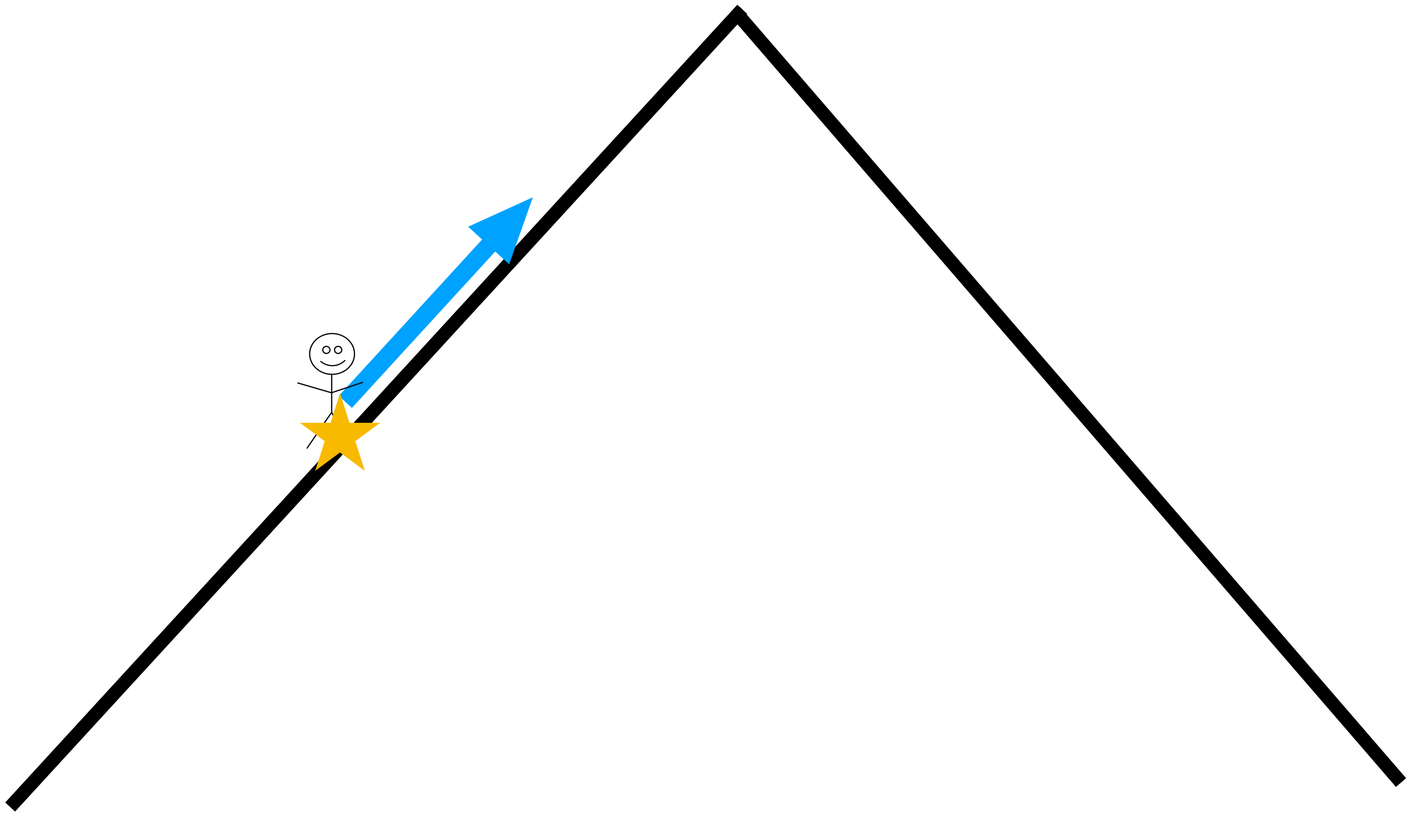

Parameters

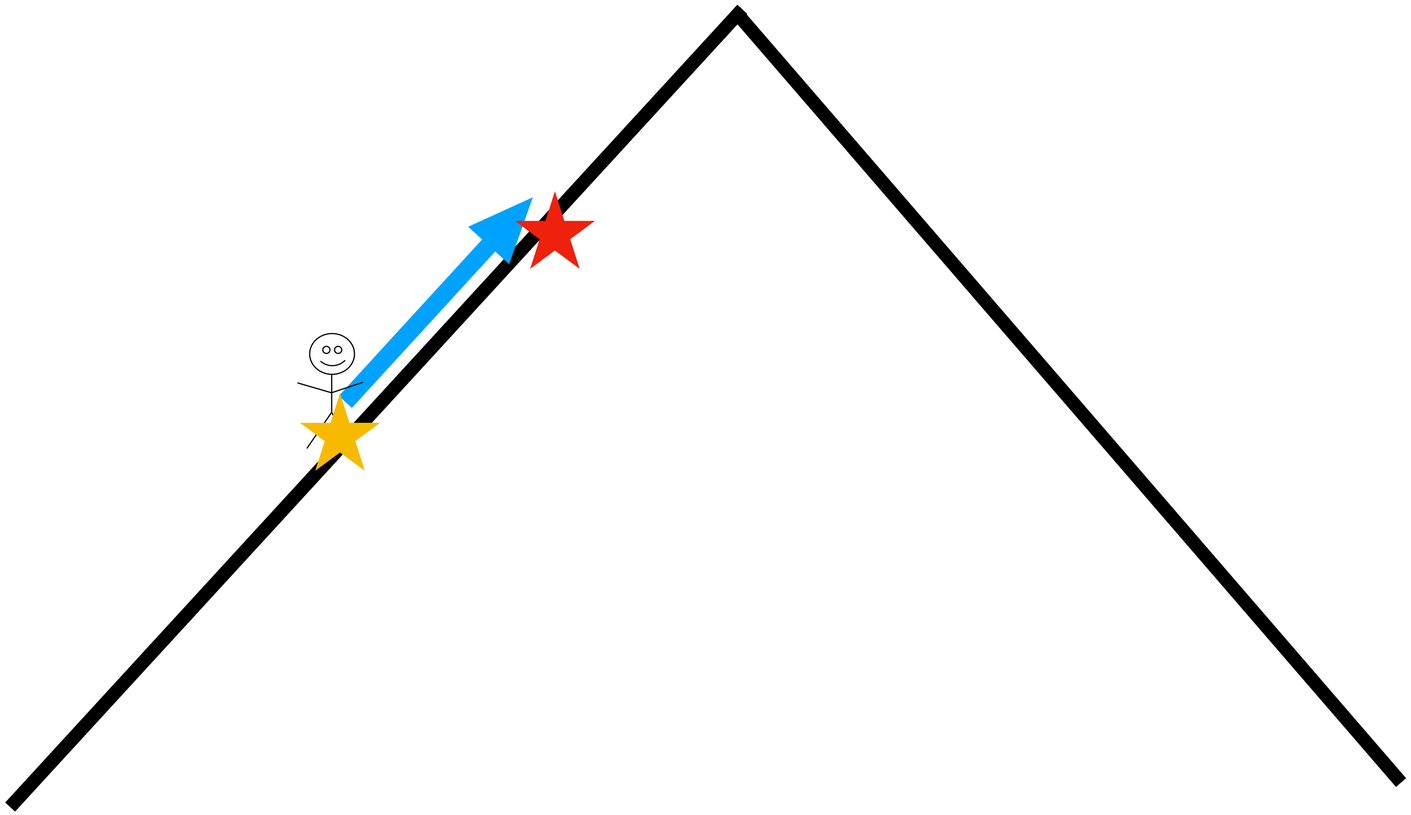

Reward
Parameters

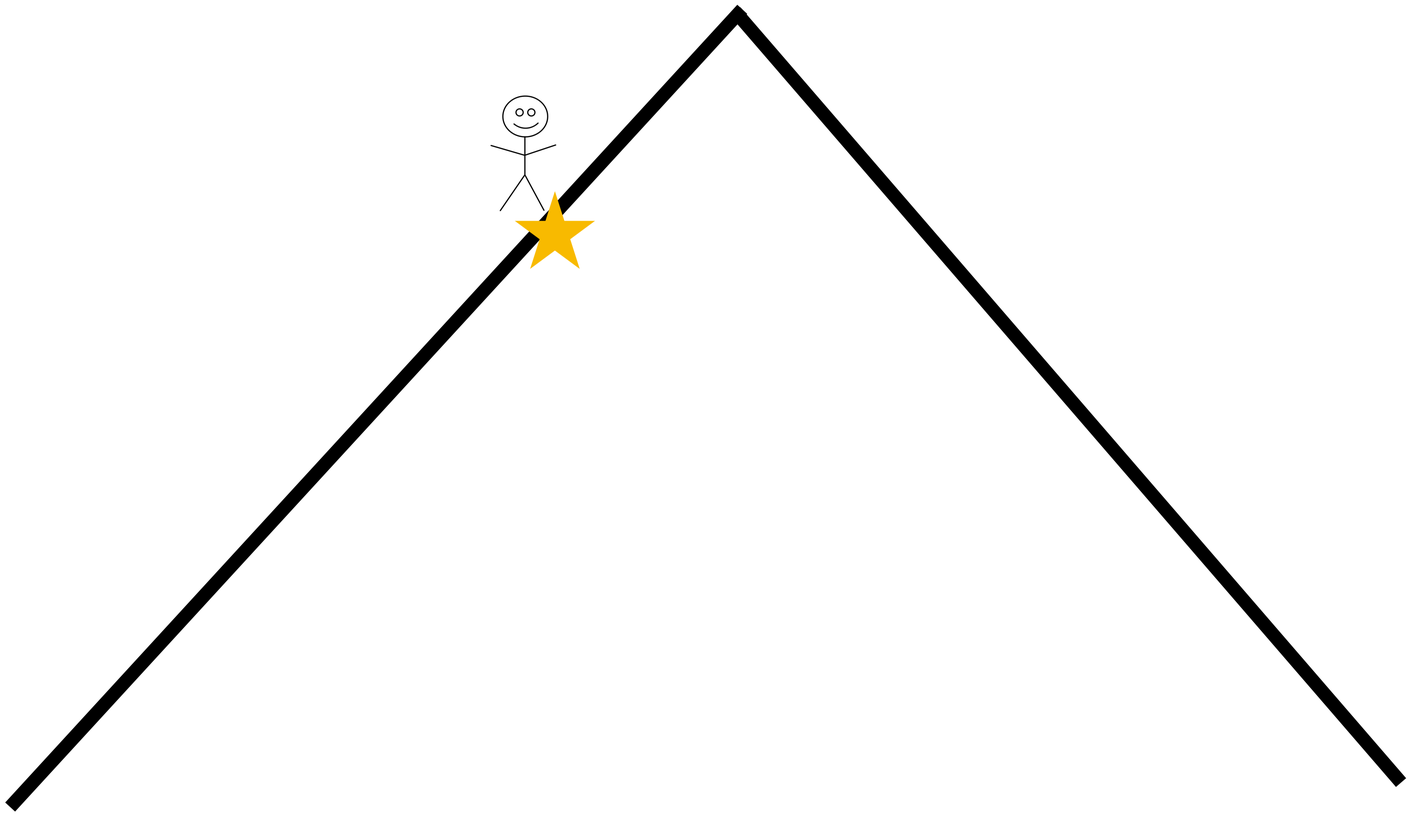

Reward
Parameters

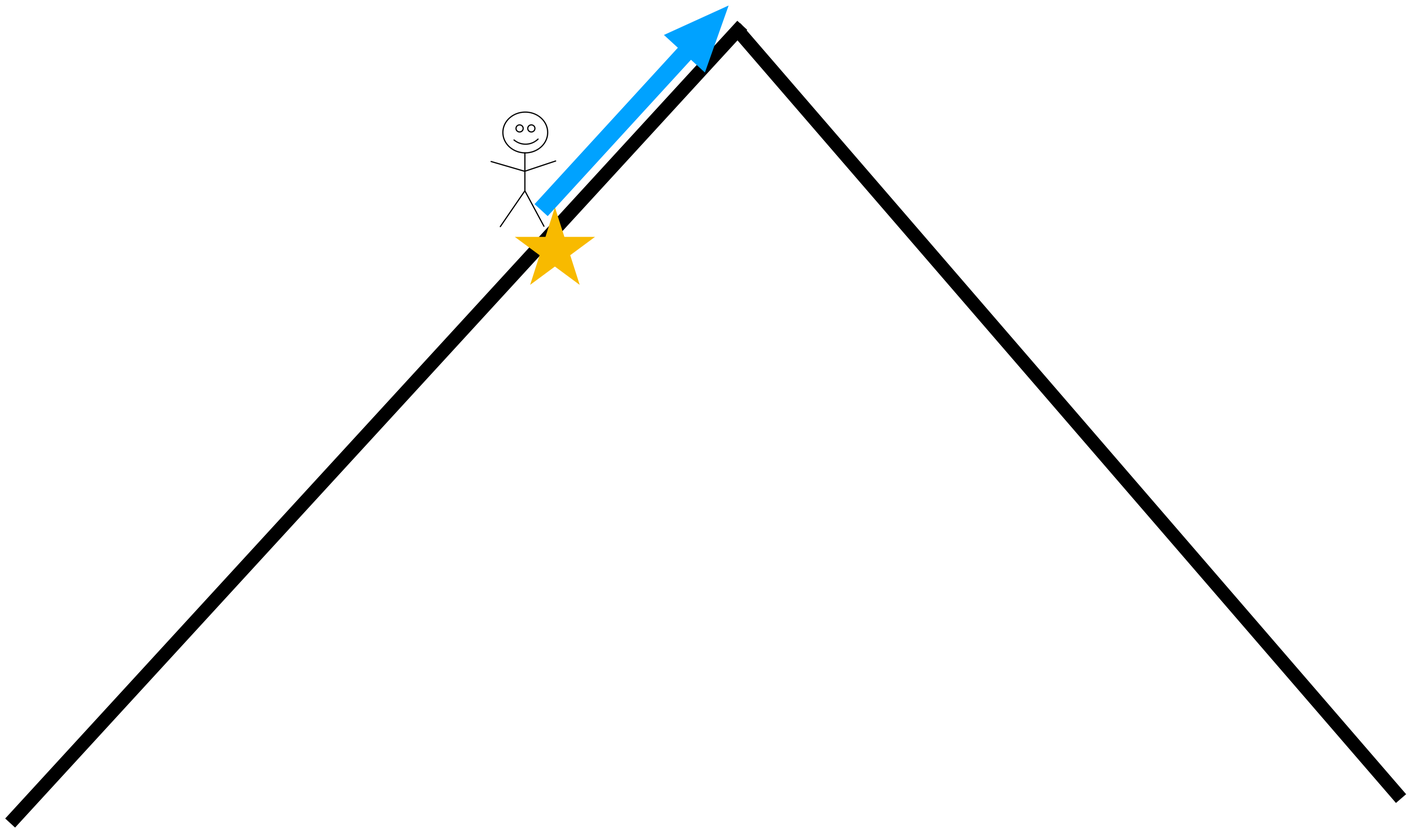

Reward
Parameters

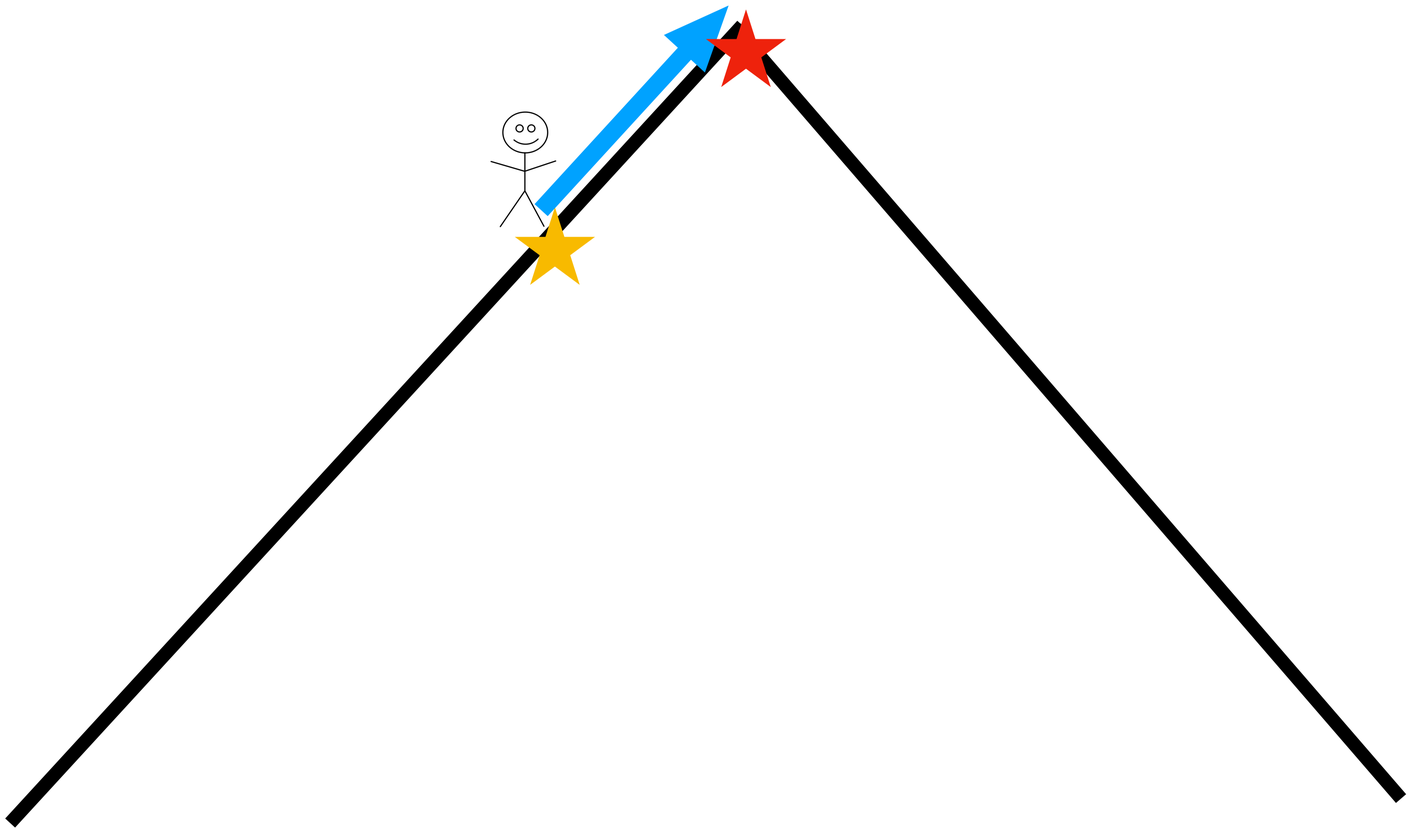


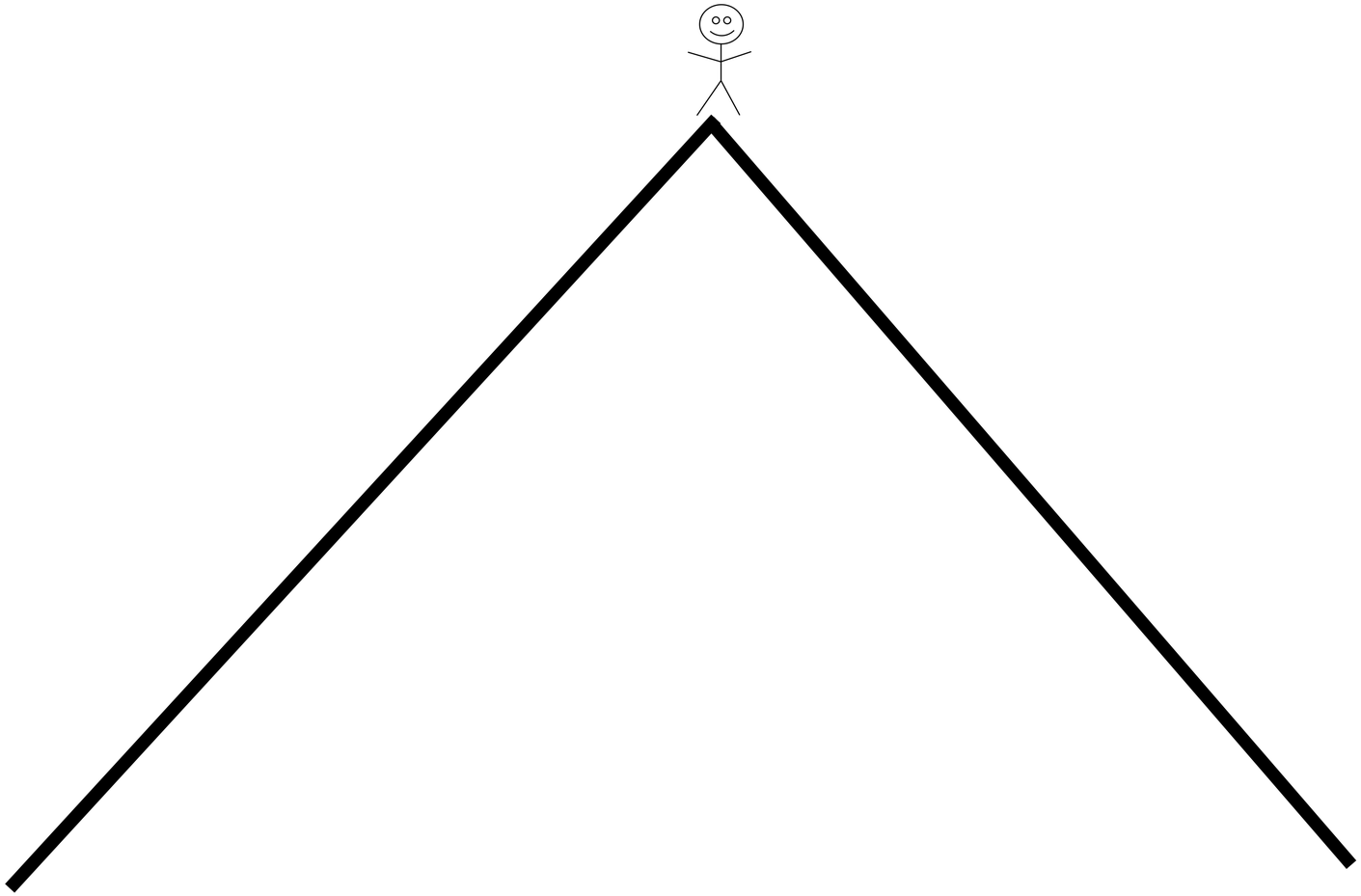


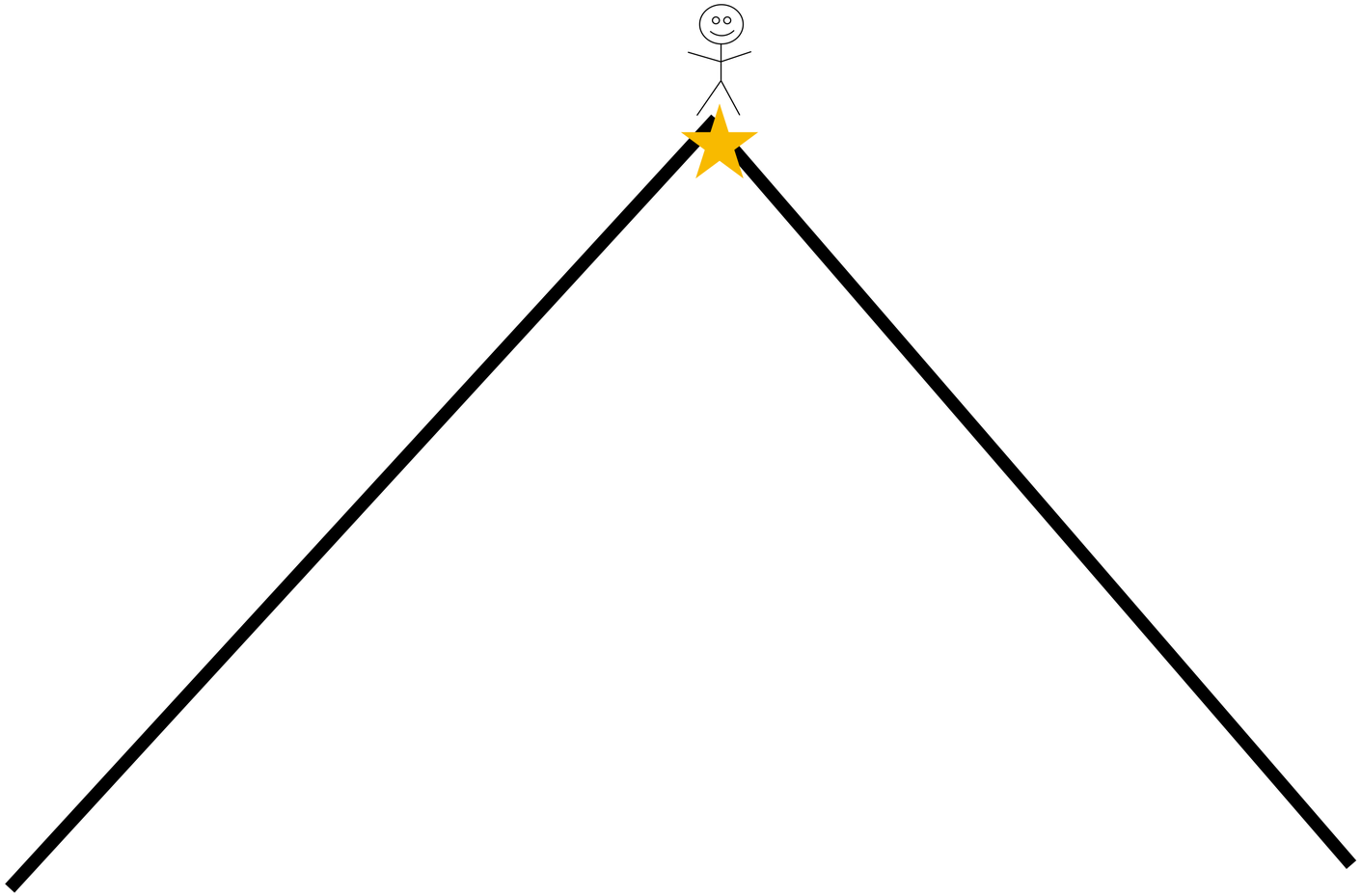


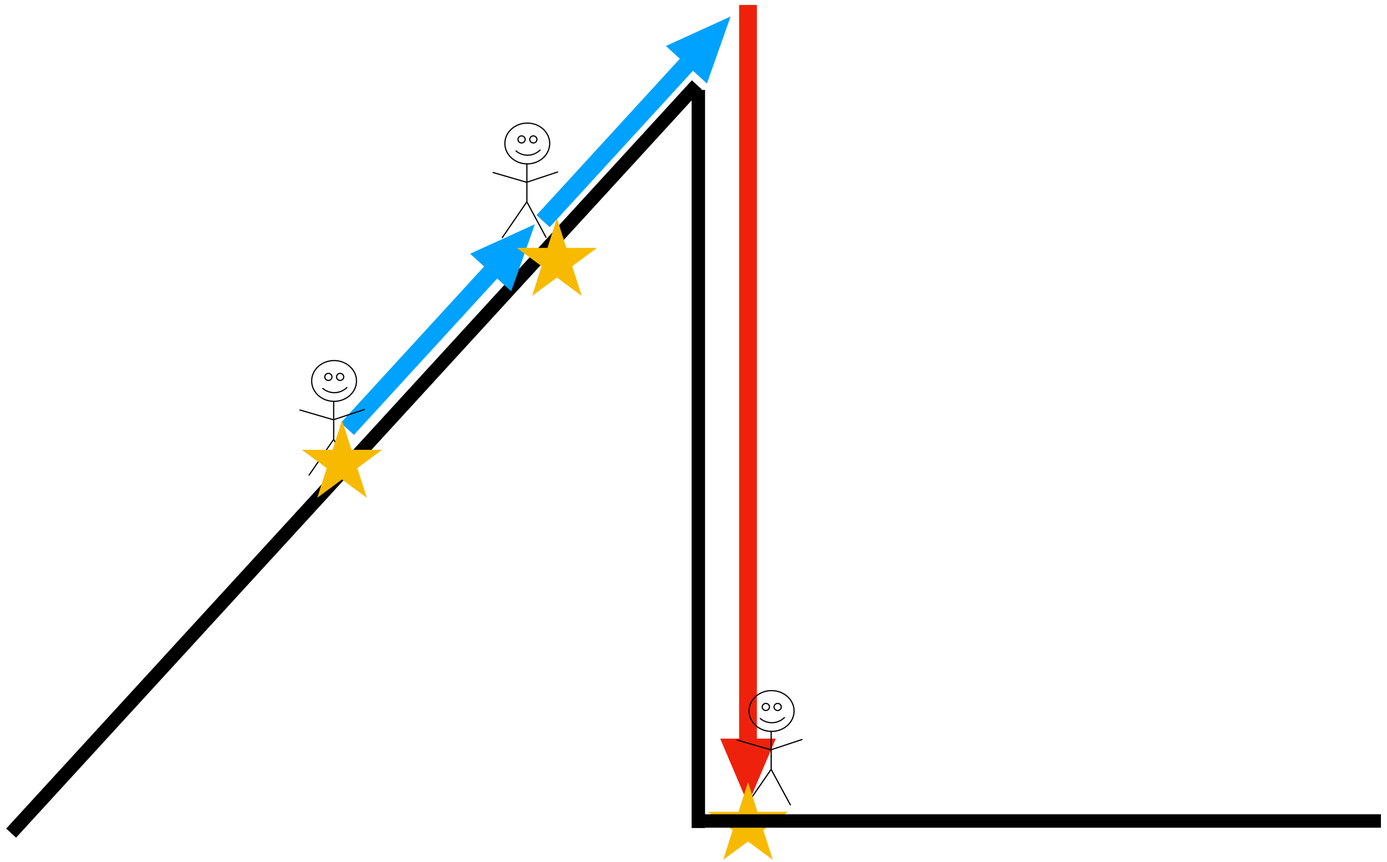


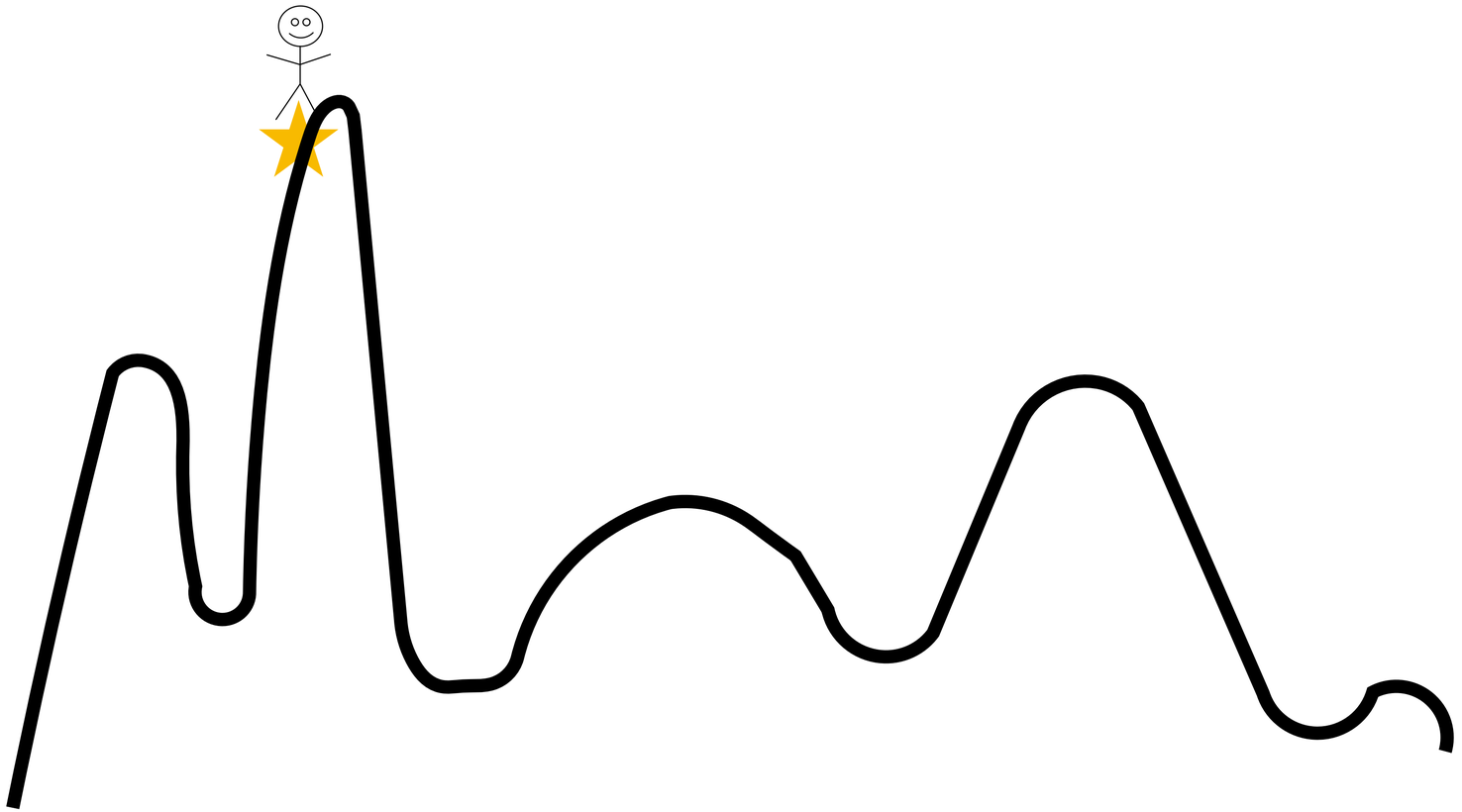


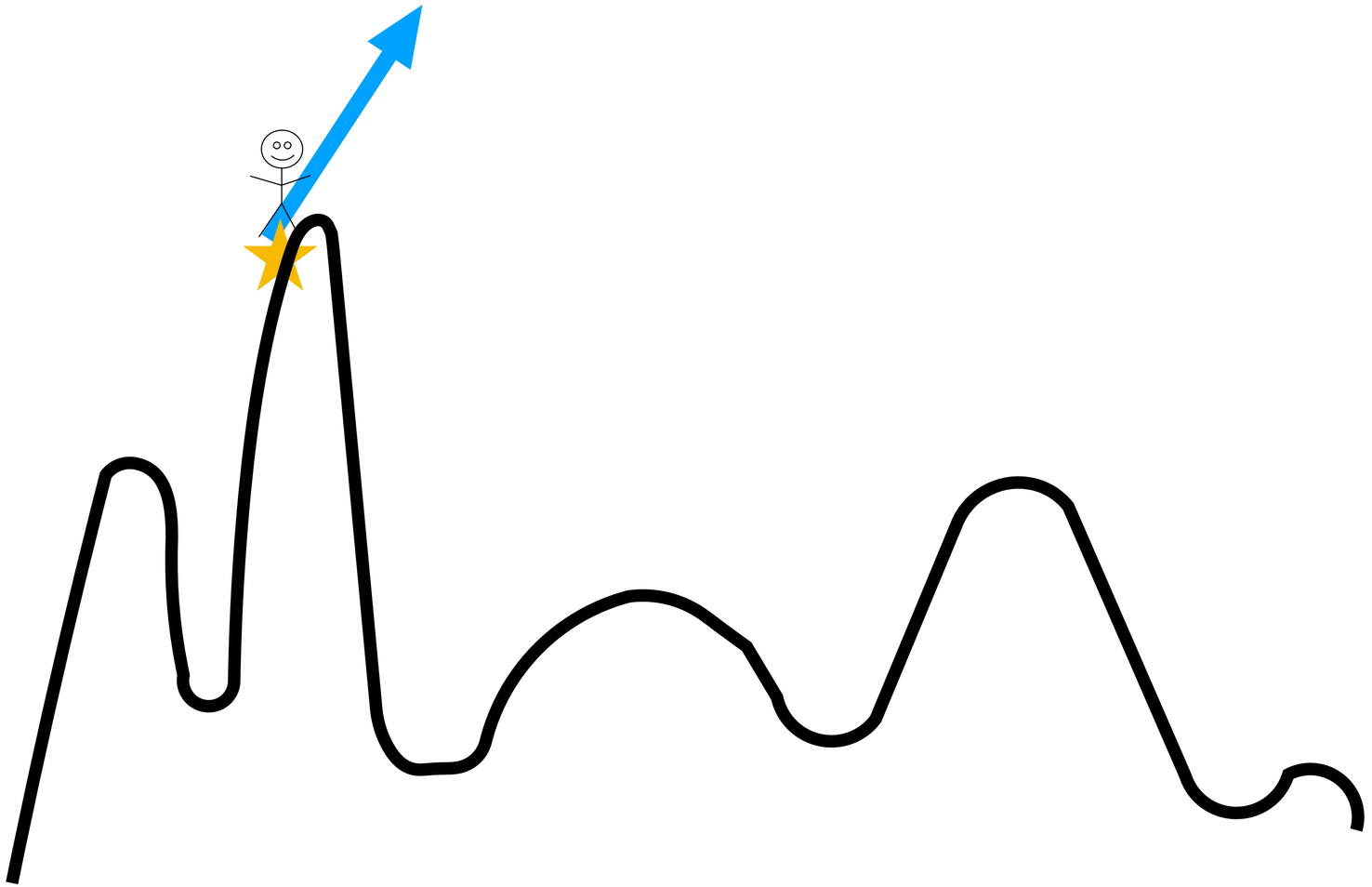


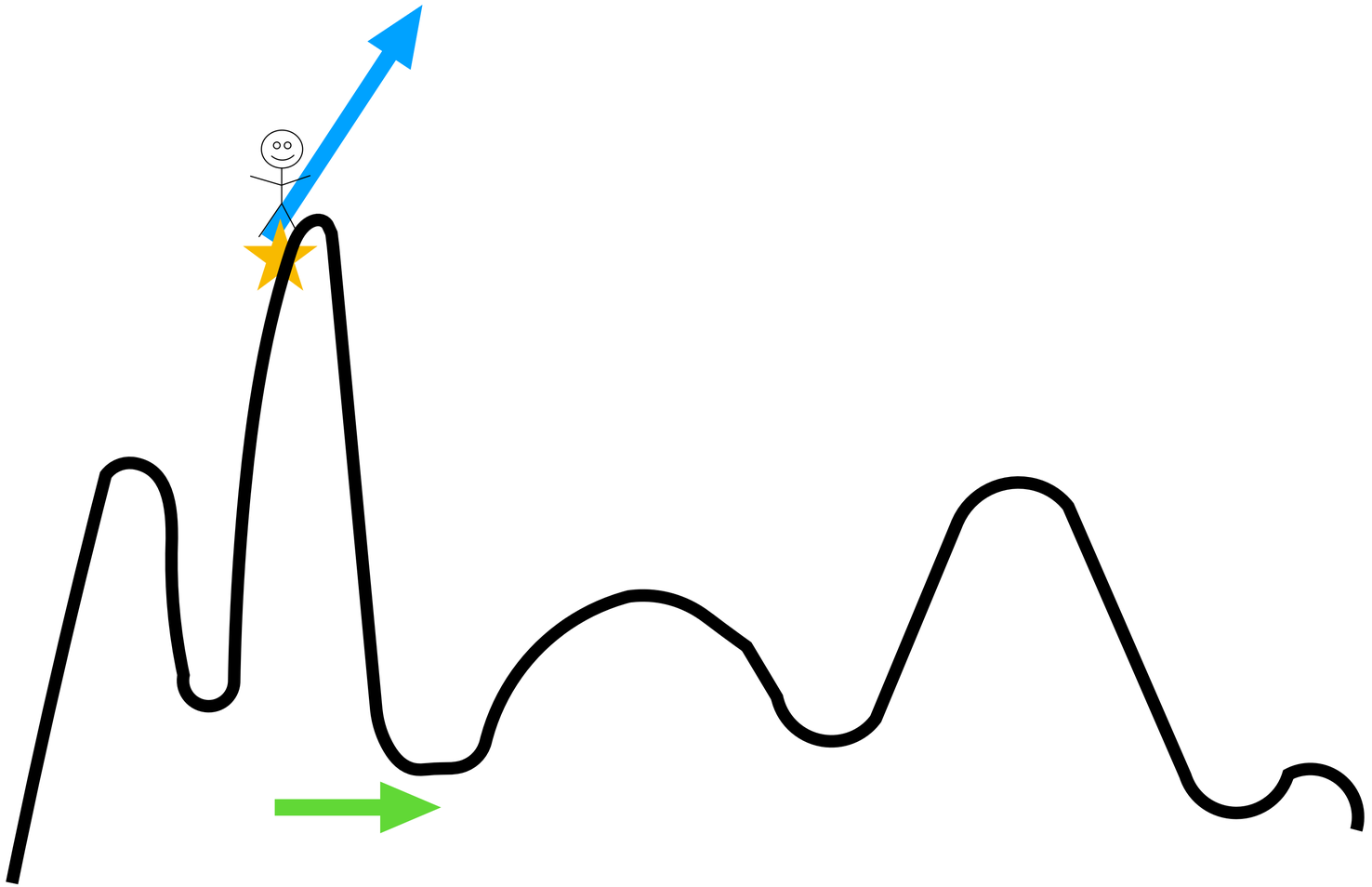


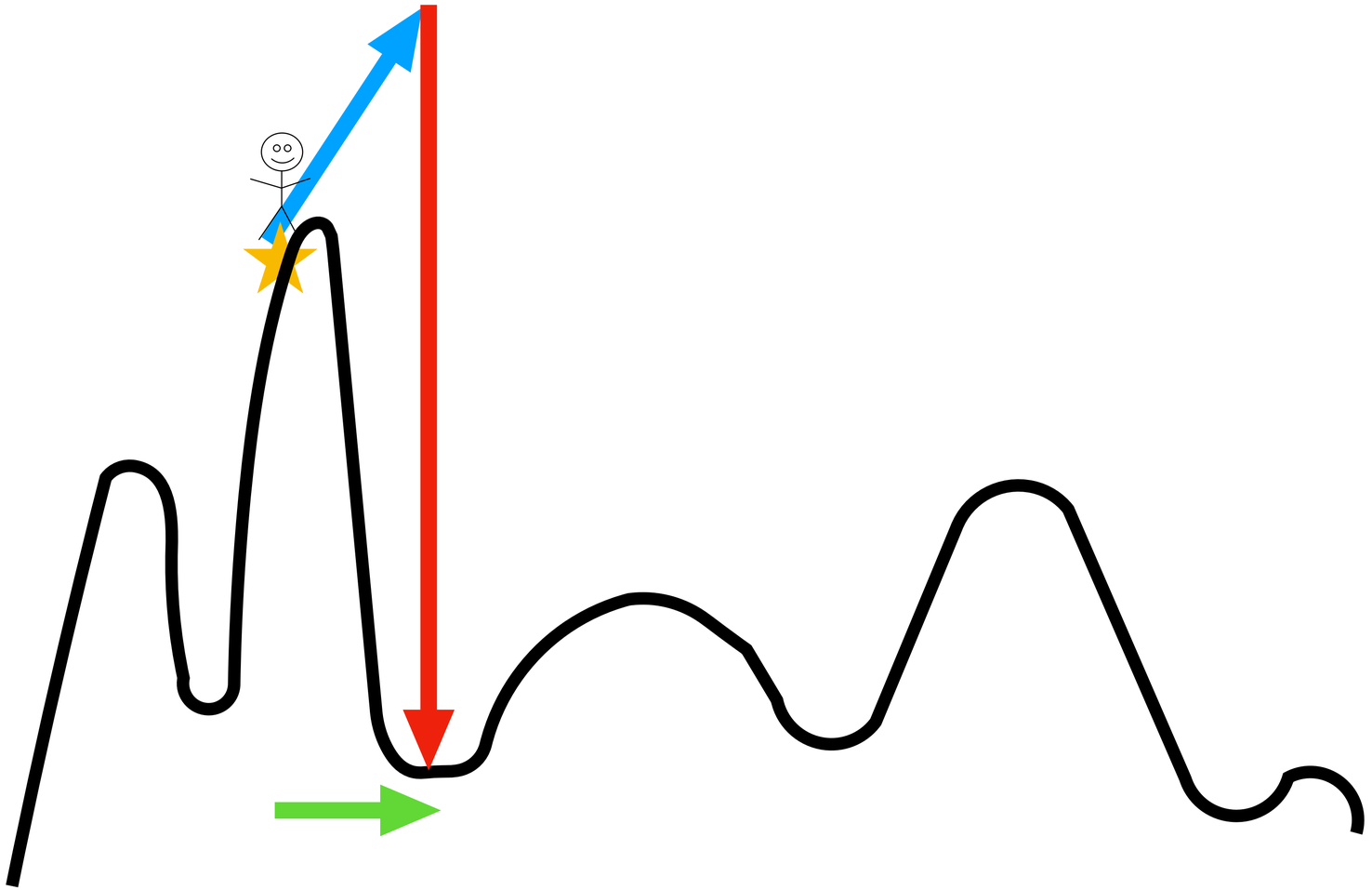


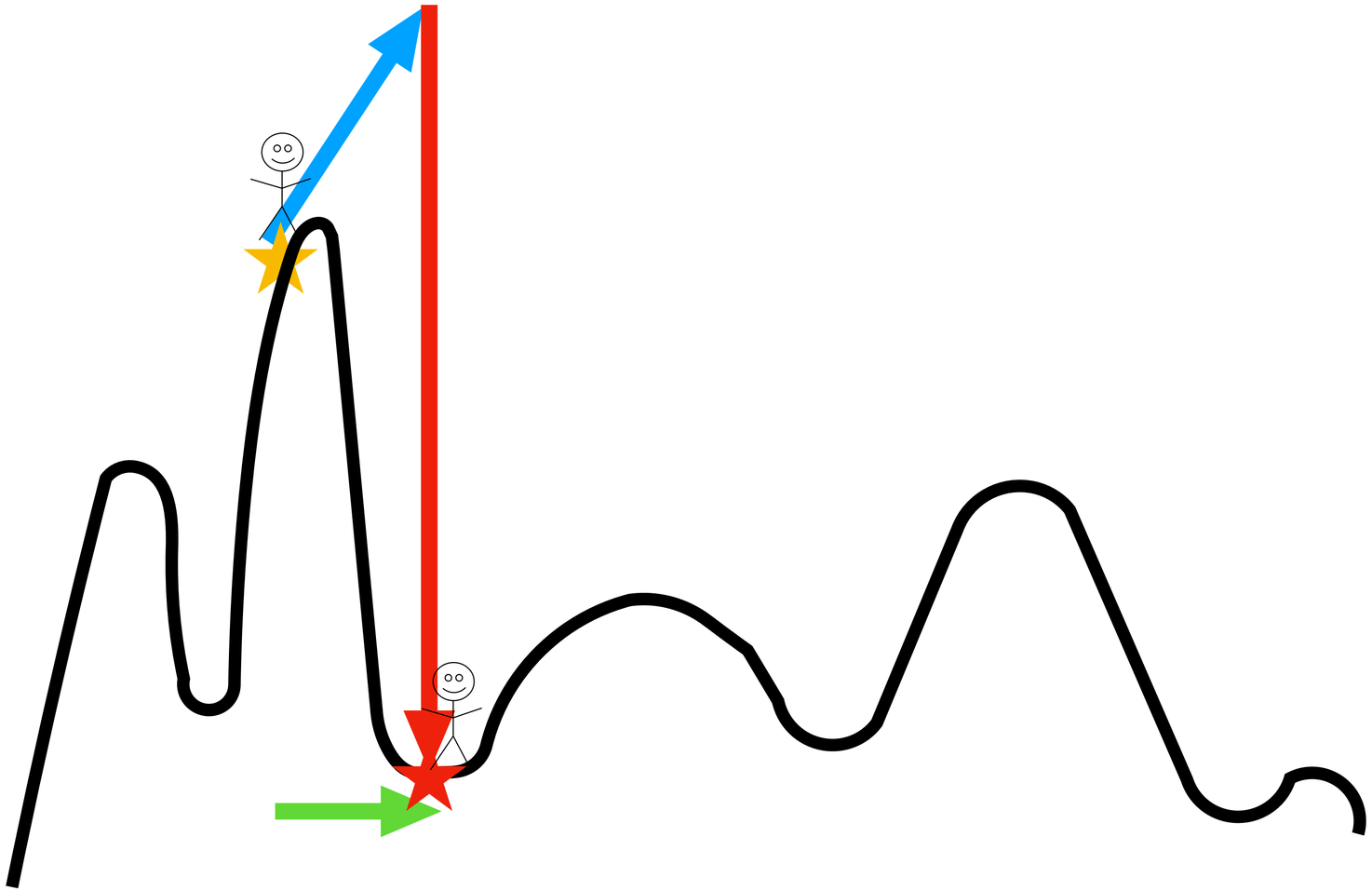


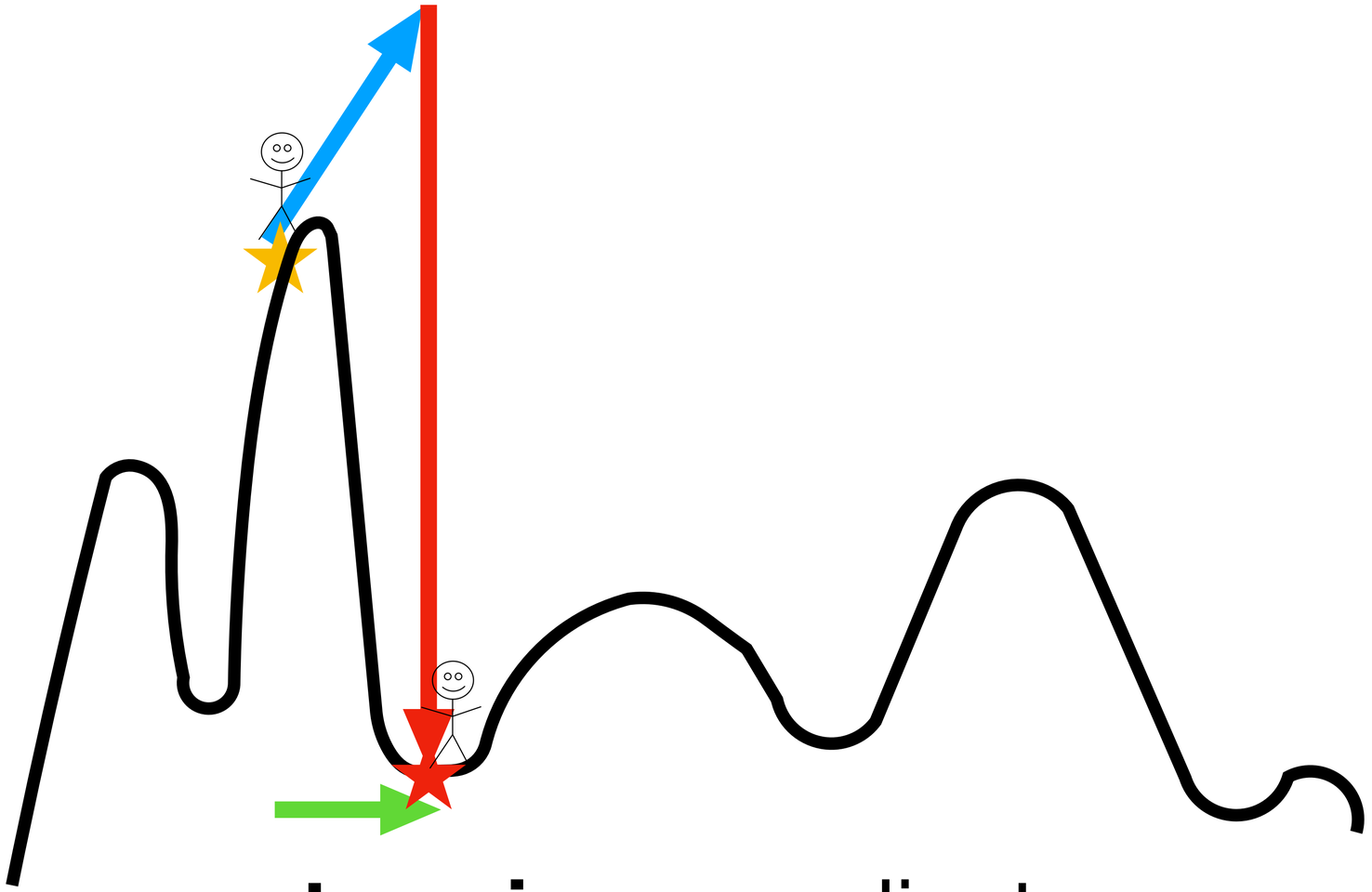












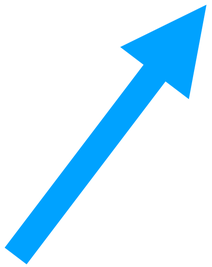
step size x gradient

Gradient descent

$$W = W - \alpha \nabla_W f$$

Gradient descent

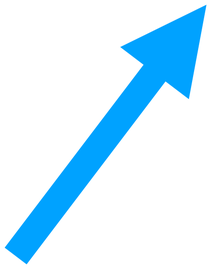
$$W = W - \alpha \nabla_W f$$



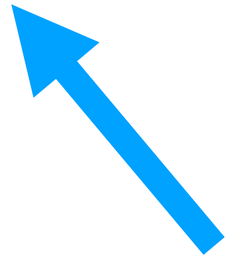
Parameters

Gradient descent

$$W = W - \alpha \nabla_W f$$



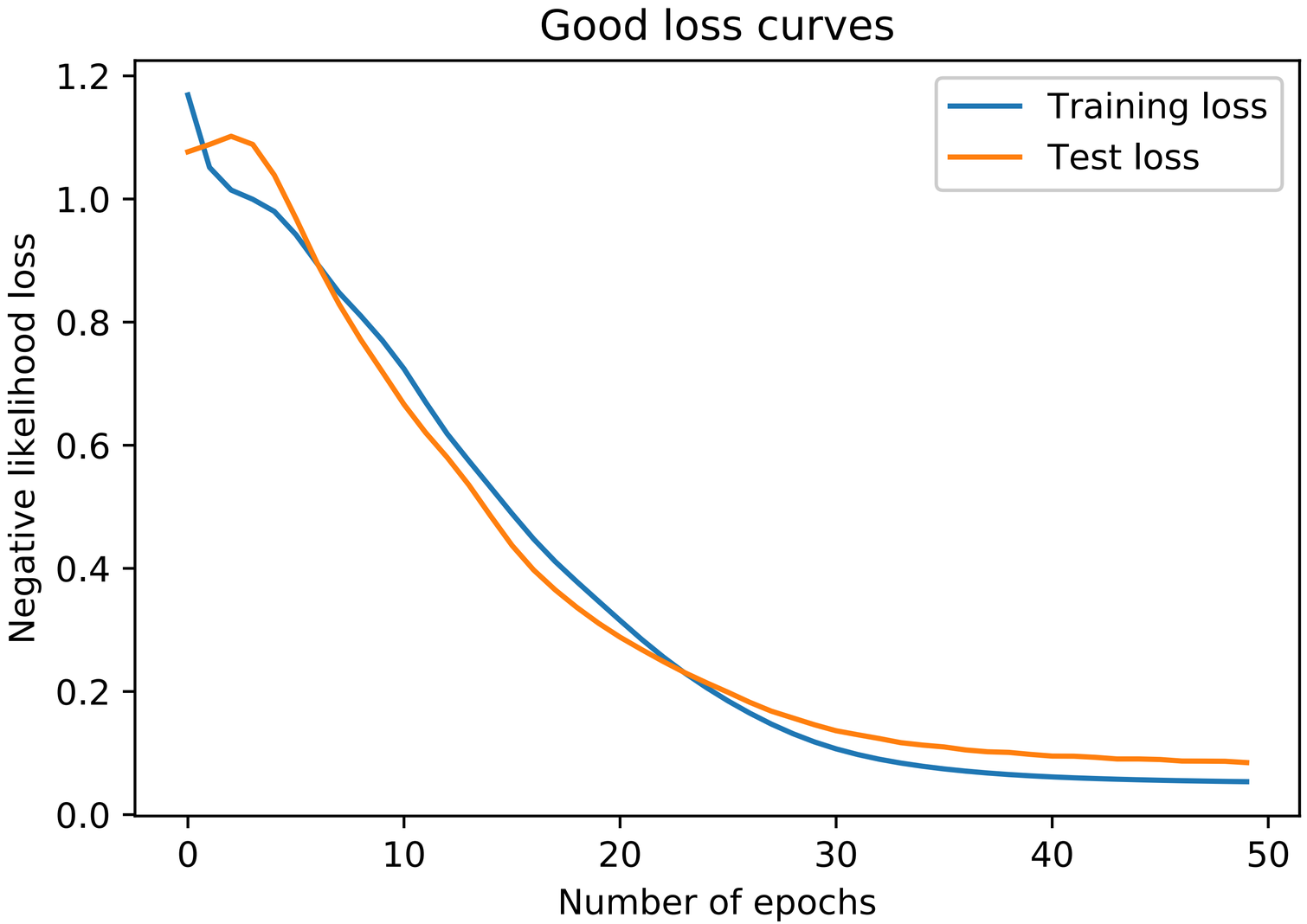
Parameters

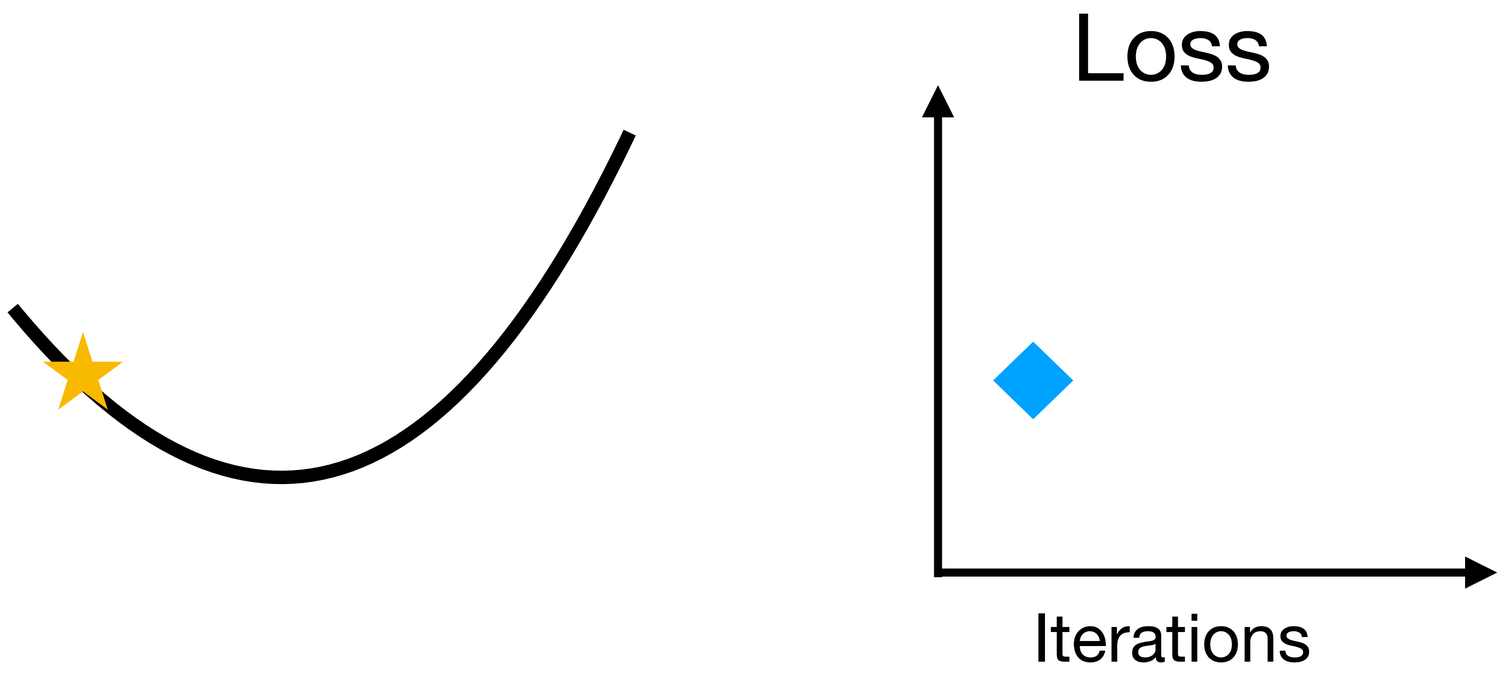


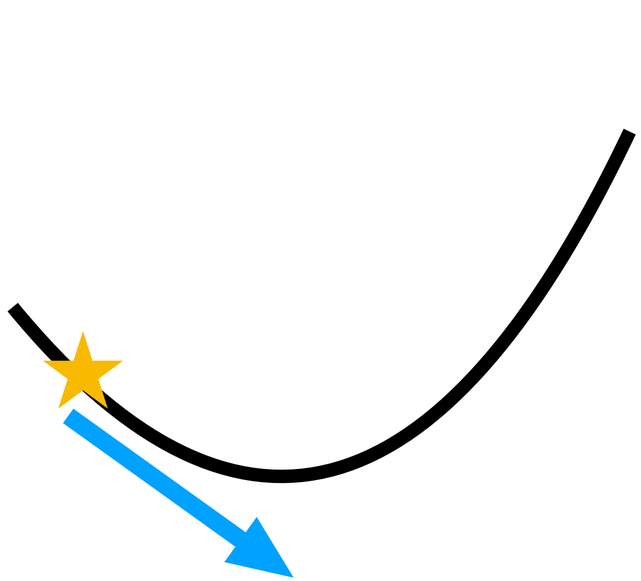
Gradient

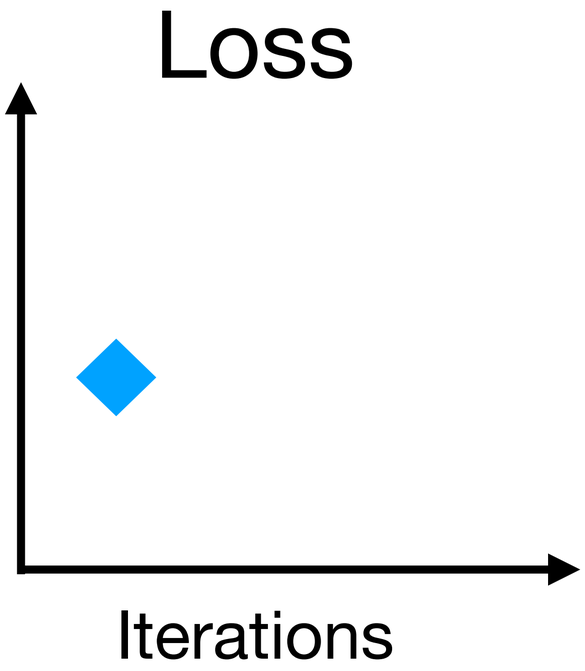
Gradient descent

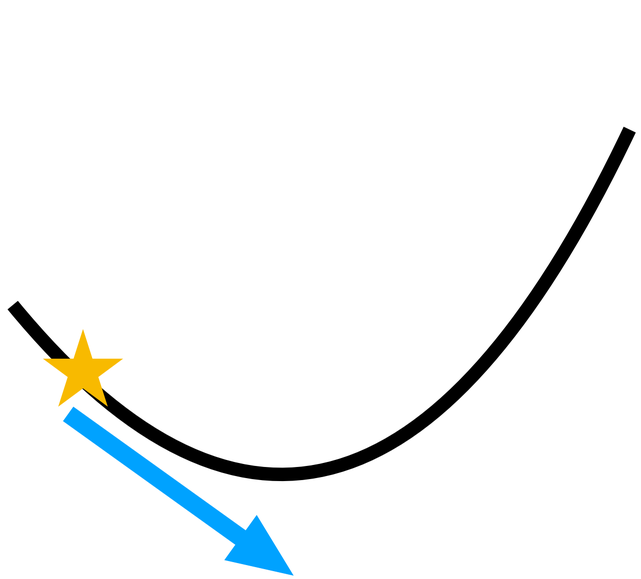
$$W = W - \alpha \nabla_W f$$


Good loss curves
Training loss
Test loss
Negative likelihood loss
Number of epochs
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50

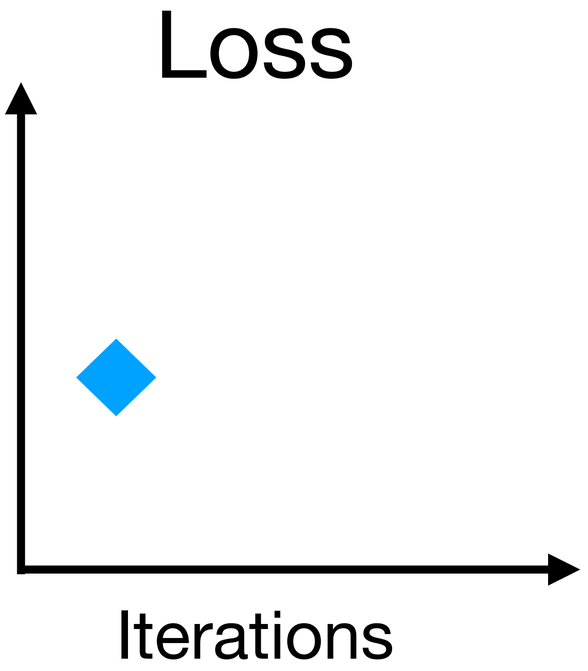

Loss
Iterations

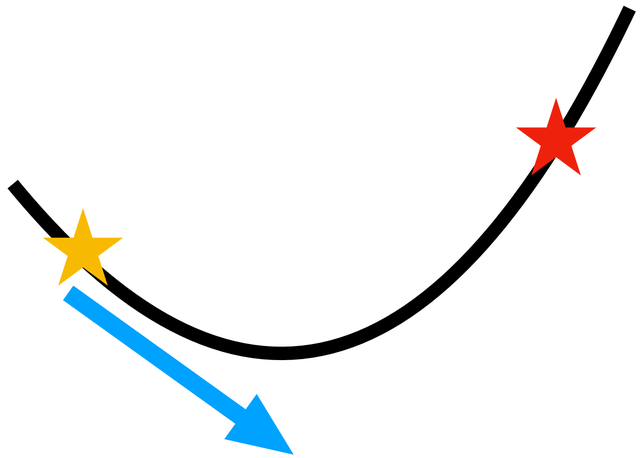



Loss
Iterations

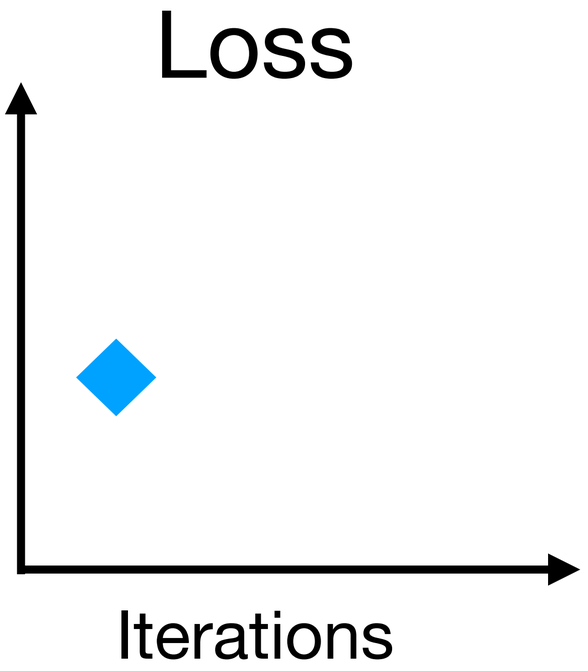


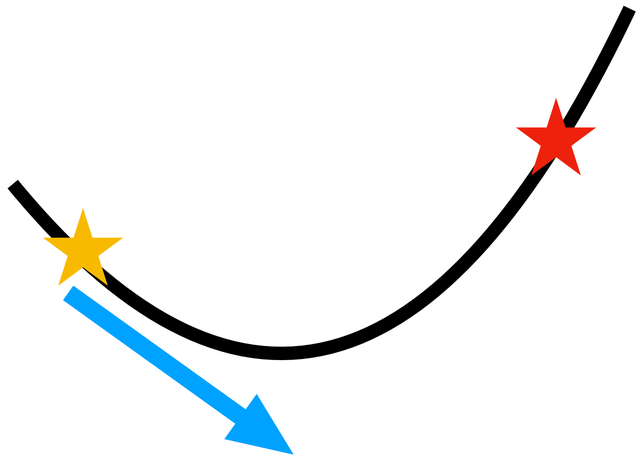
gradient x step size

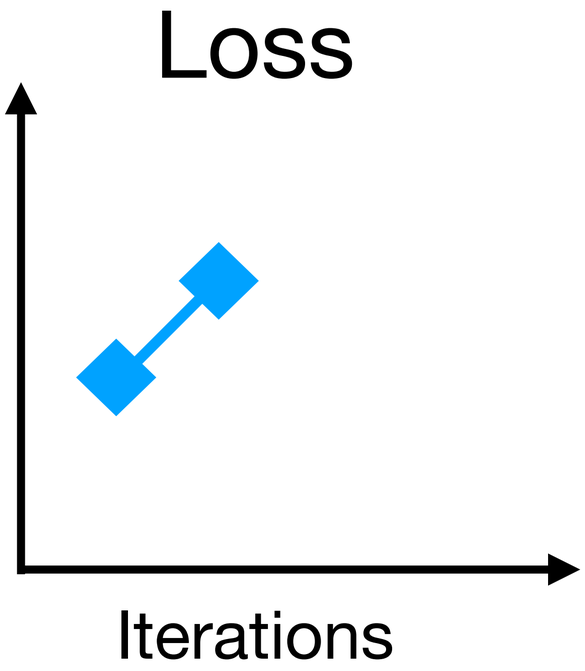




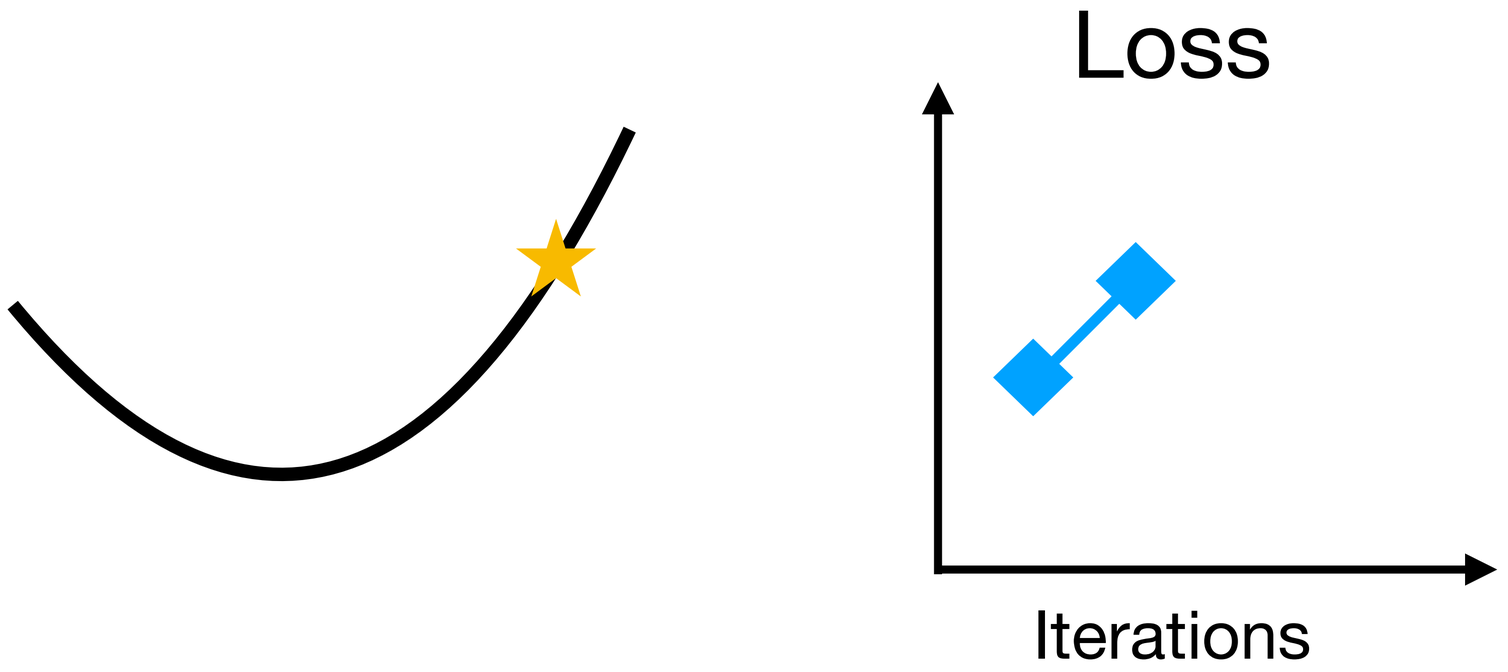
gradient x step size

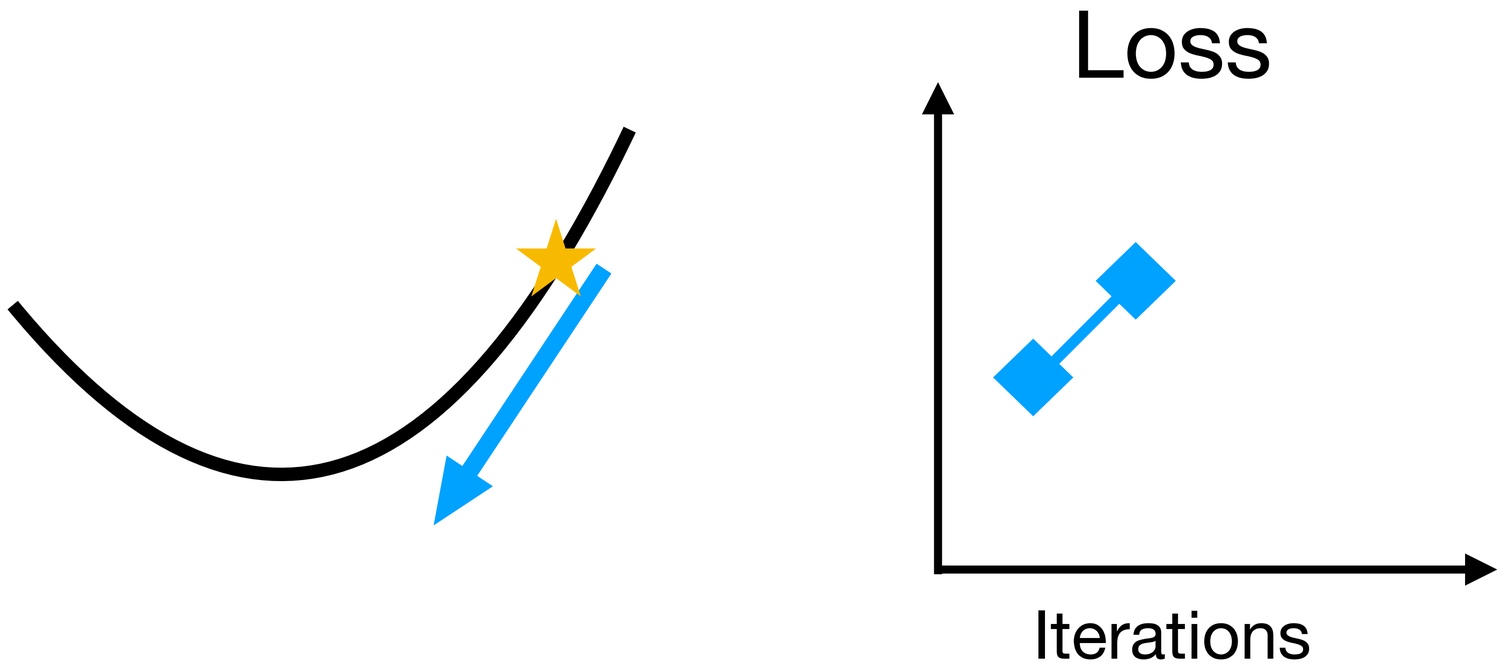


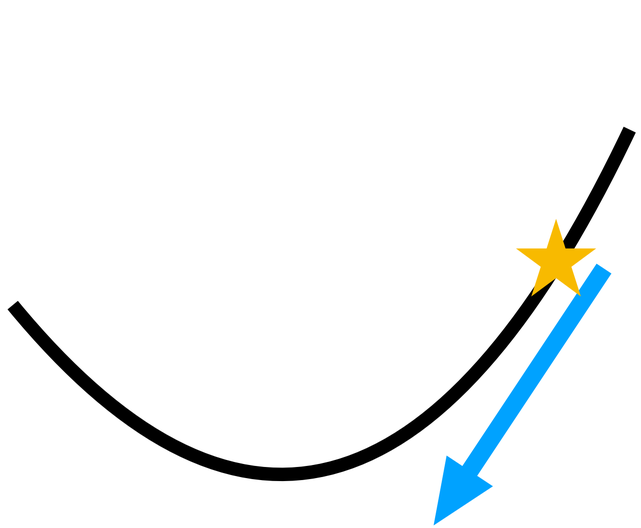


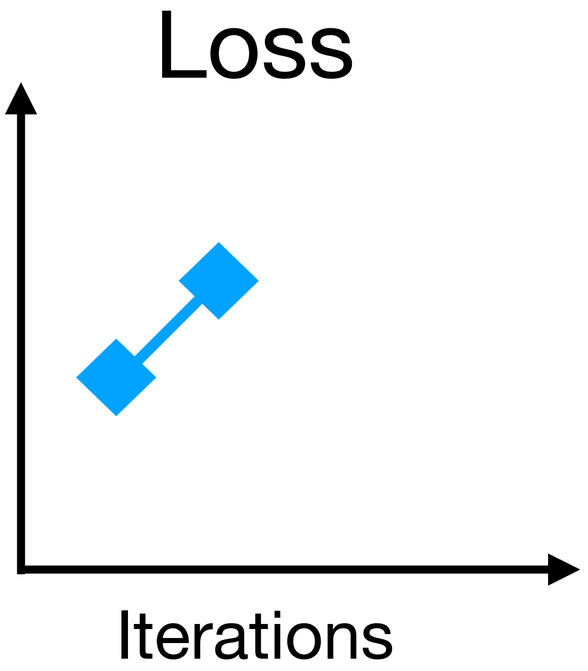


gradient x step size

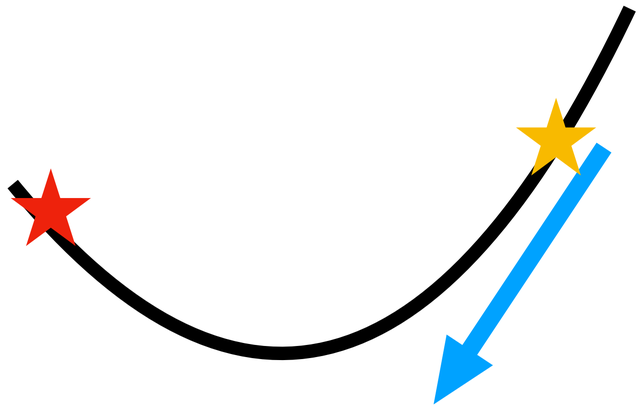

Loss
Iterations

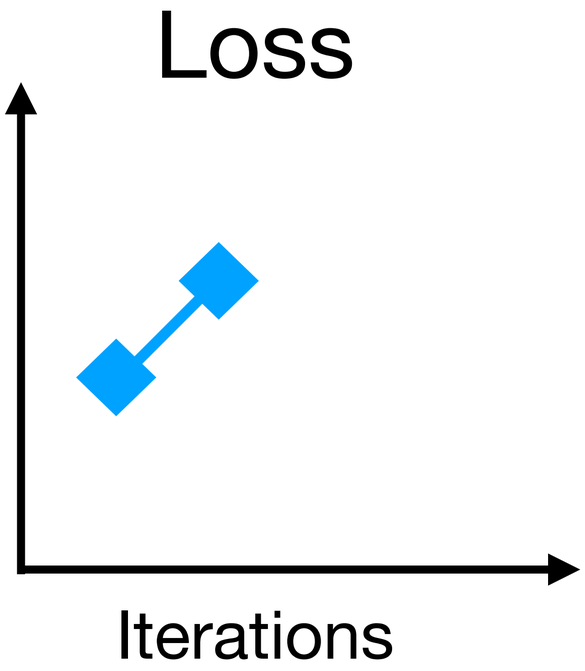

Loss
Iterations



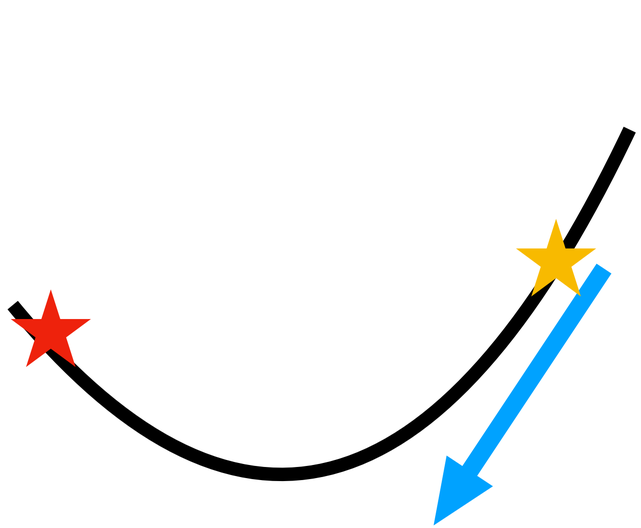


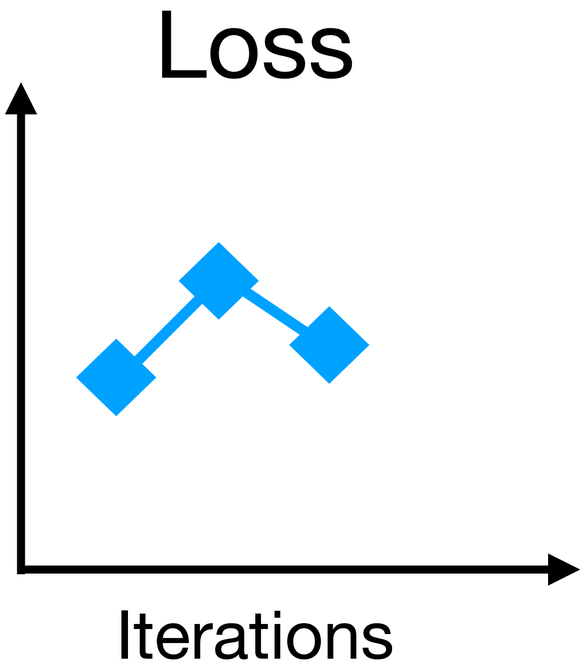
gradient x step size



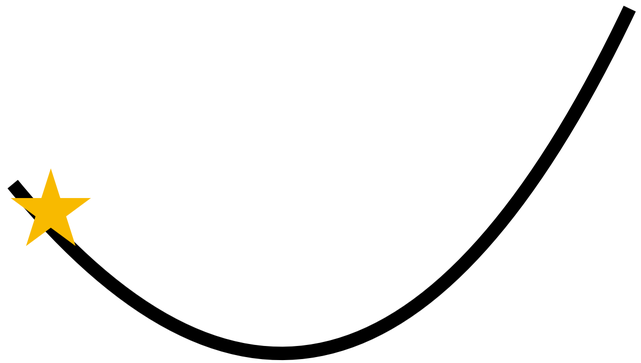


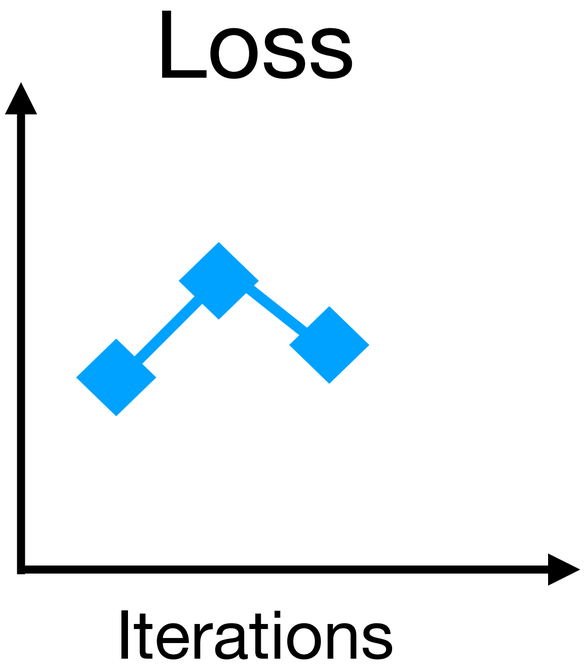
gradient x step size

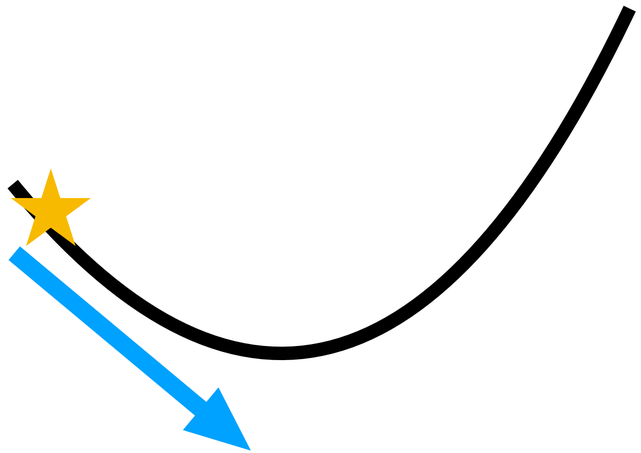


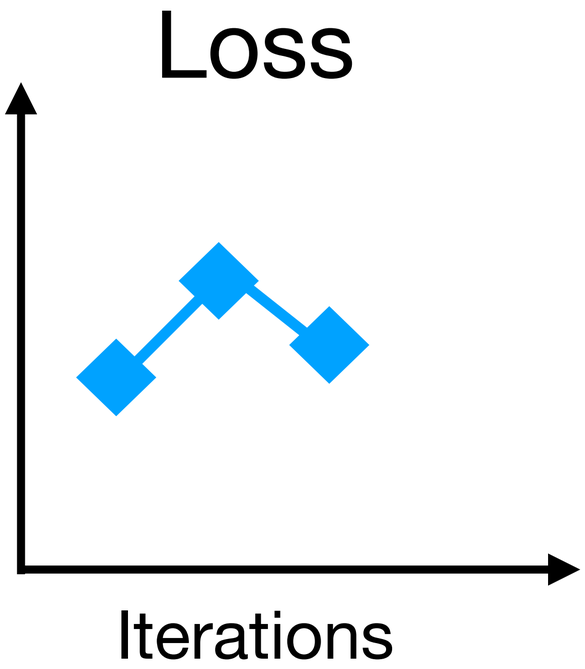


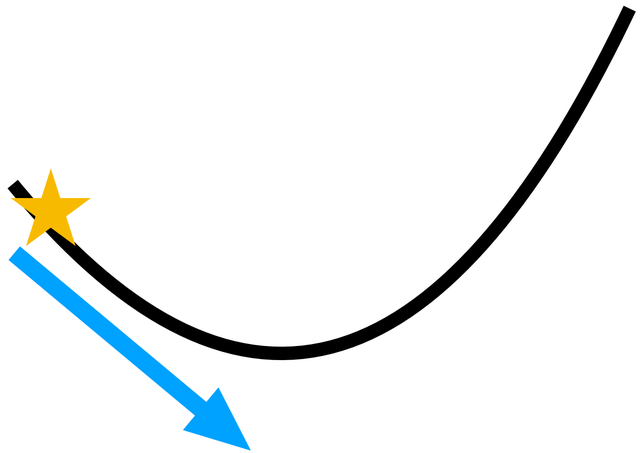
gradient x step size

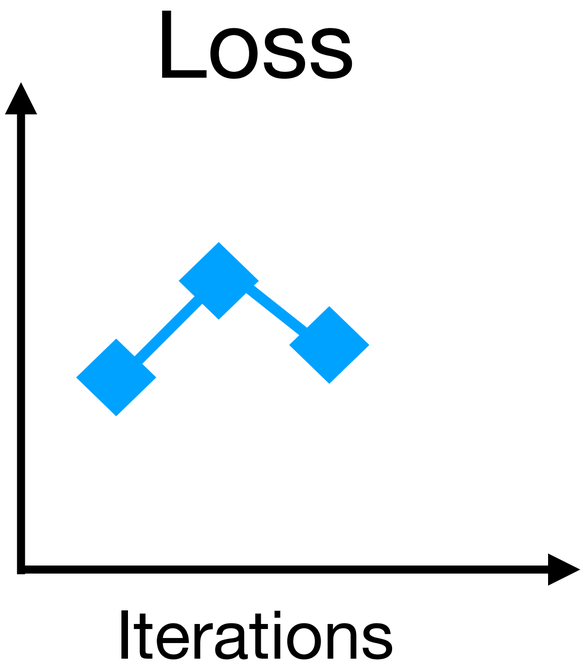



Loss
Iterations

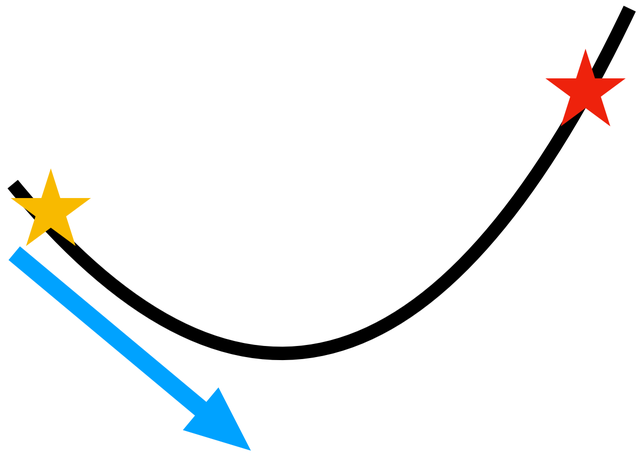


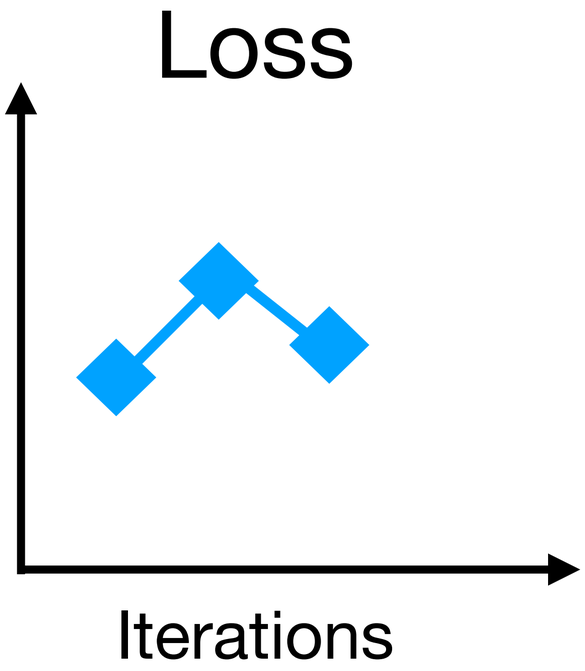

Loss
Iterations



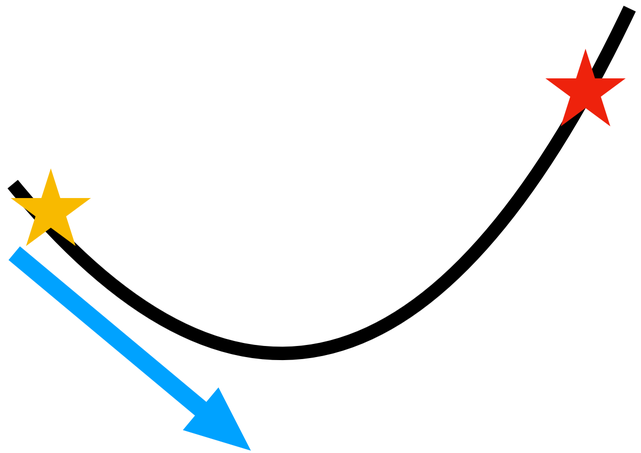


gradient x step size

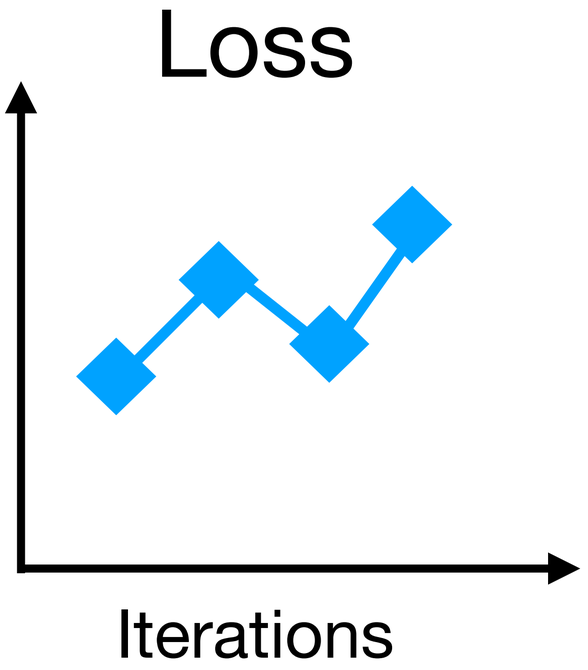


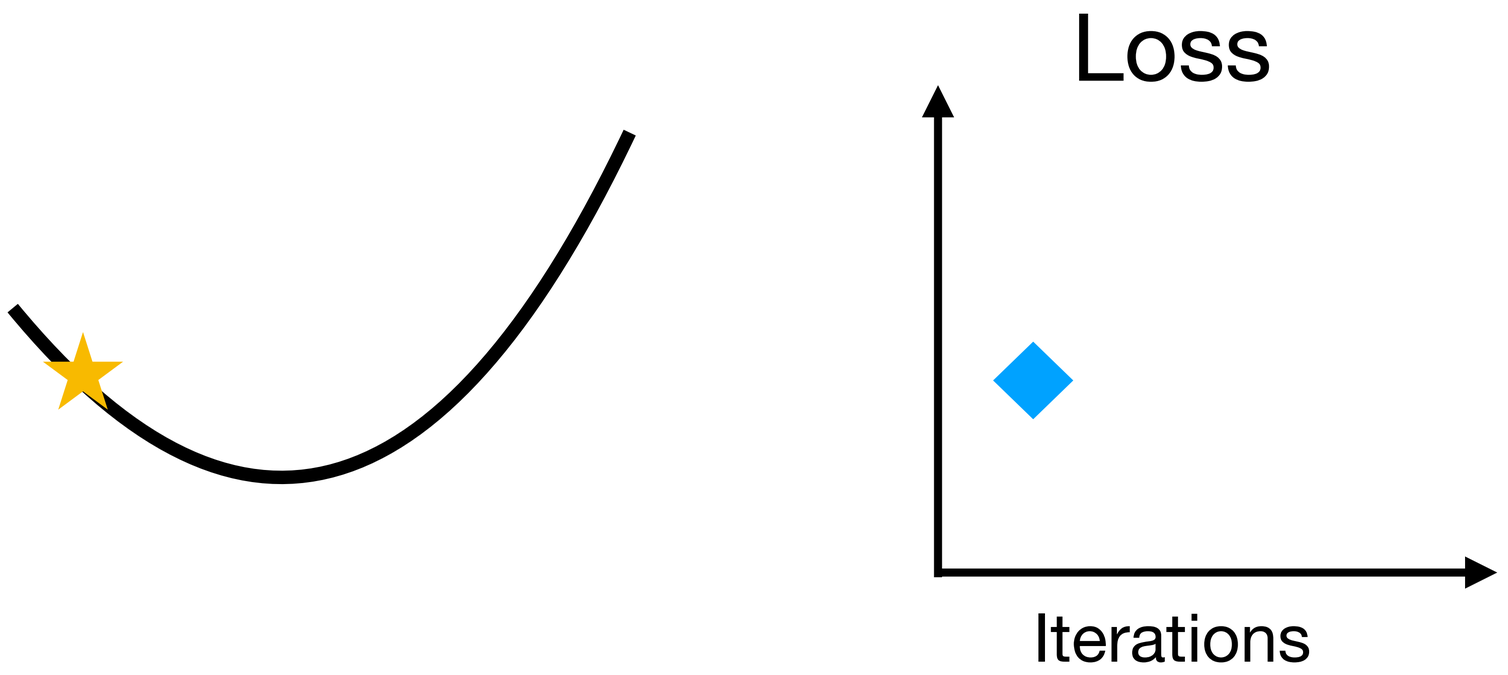


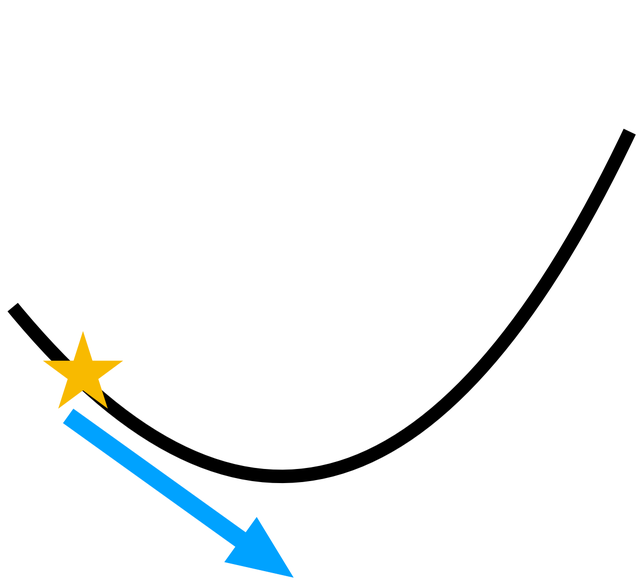
gradient x step size

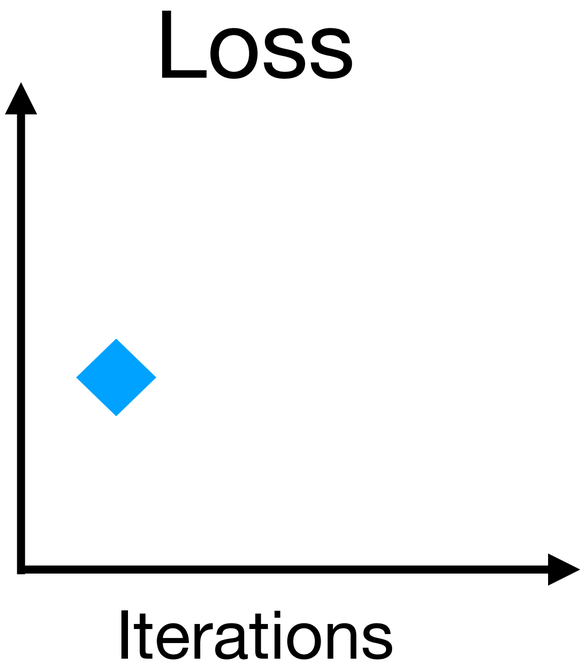


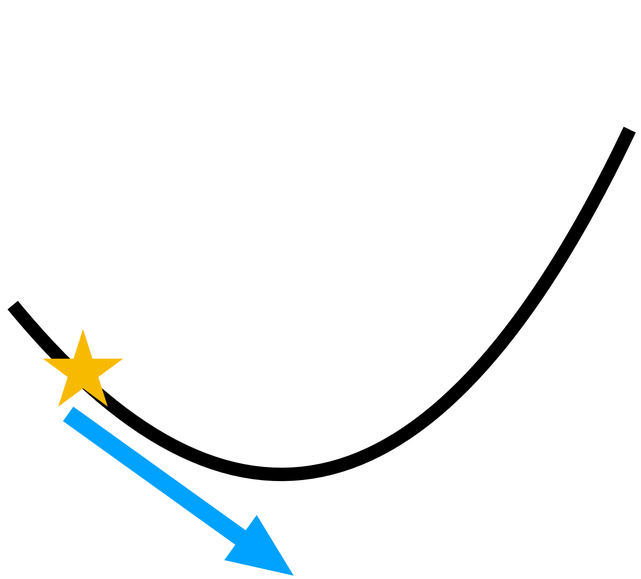
gradient x step size

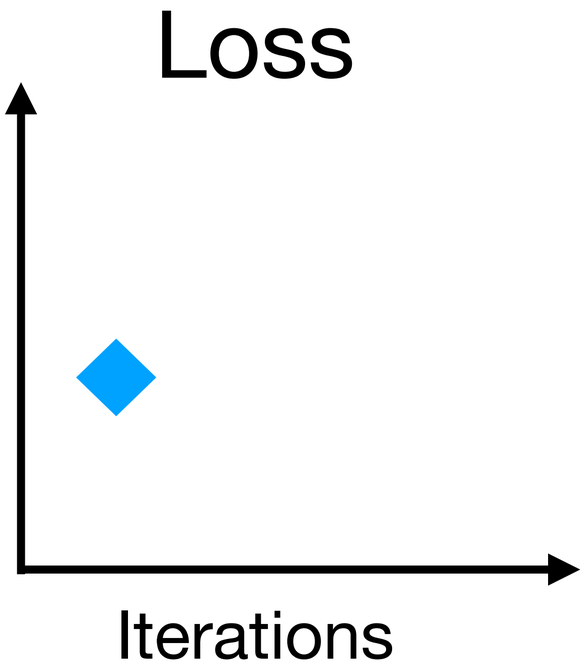



Loss
Iterations

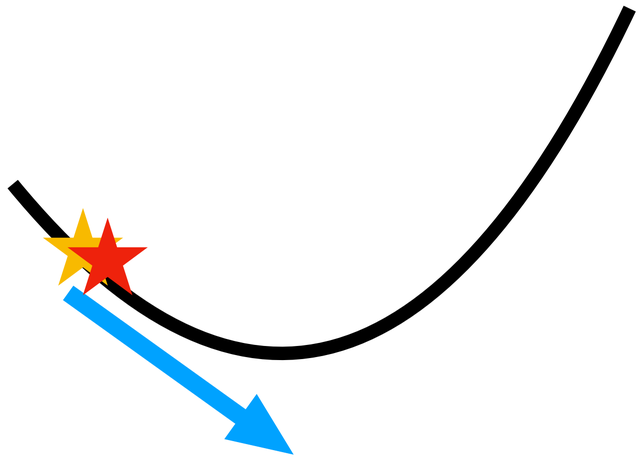


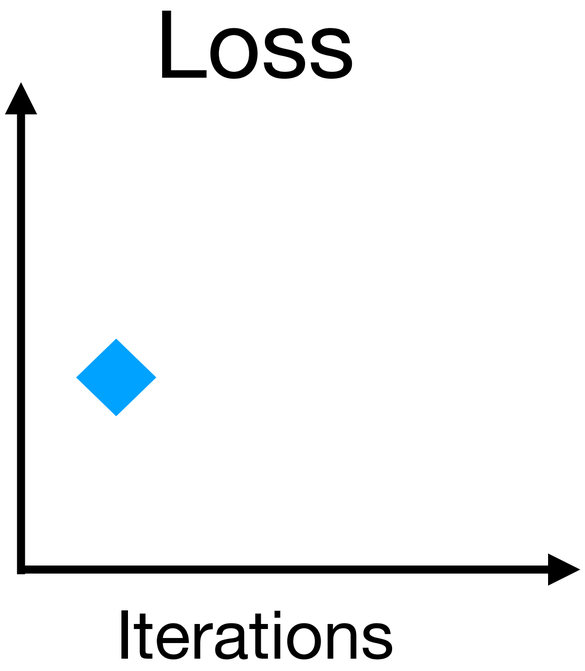

Loss
Iterations



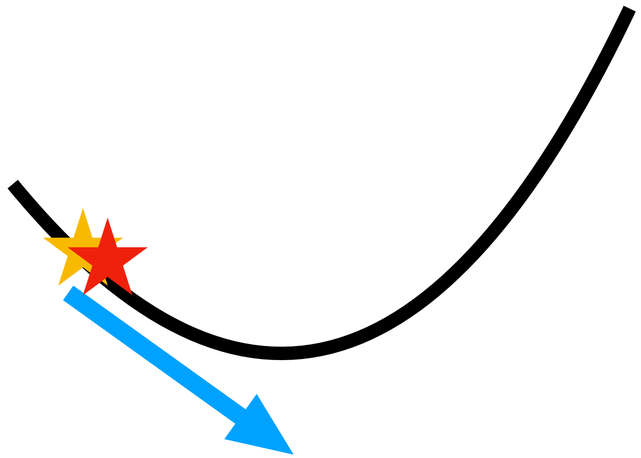


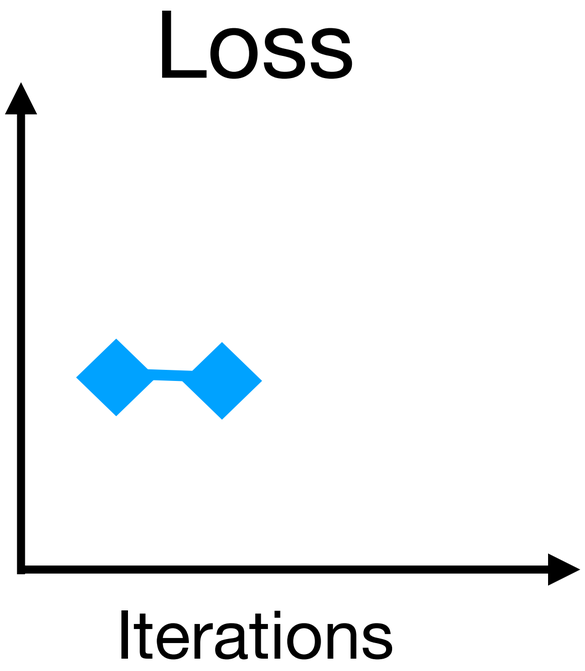
gradient x step size





gradient x step size





gradient x step size

Iris dataset

Input

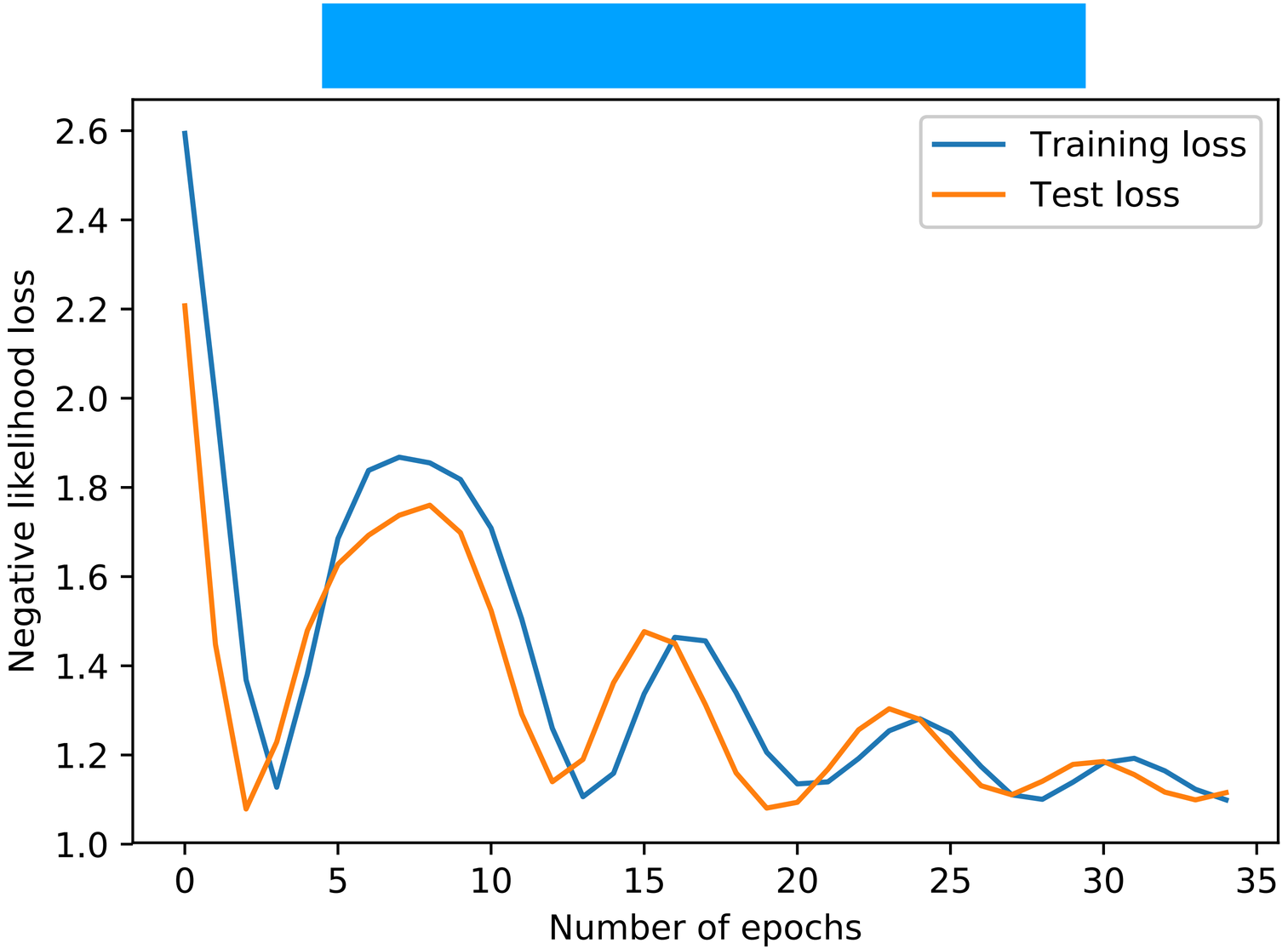
Dense 1

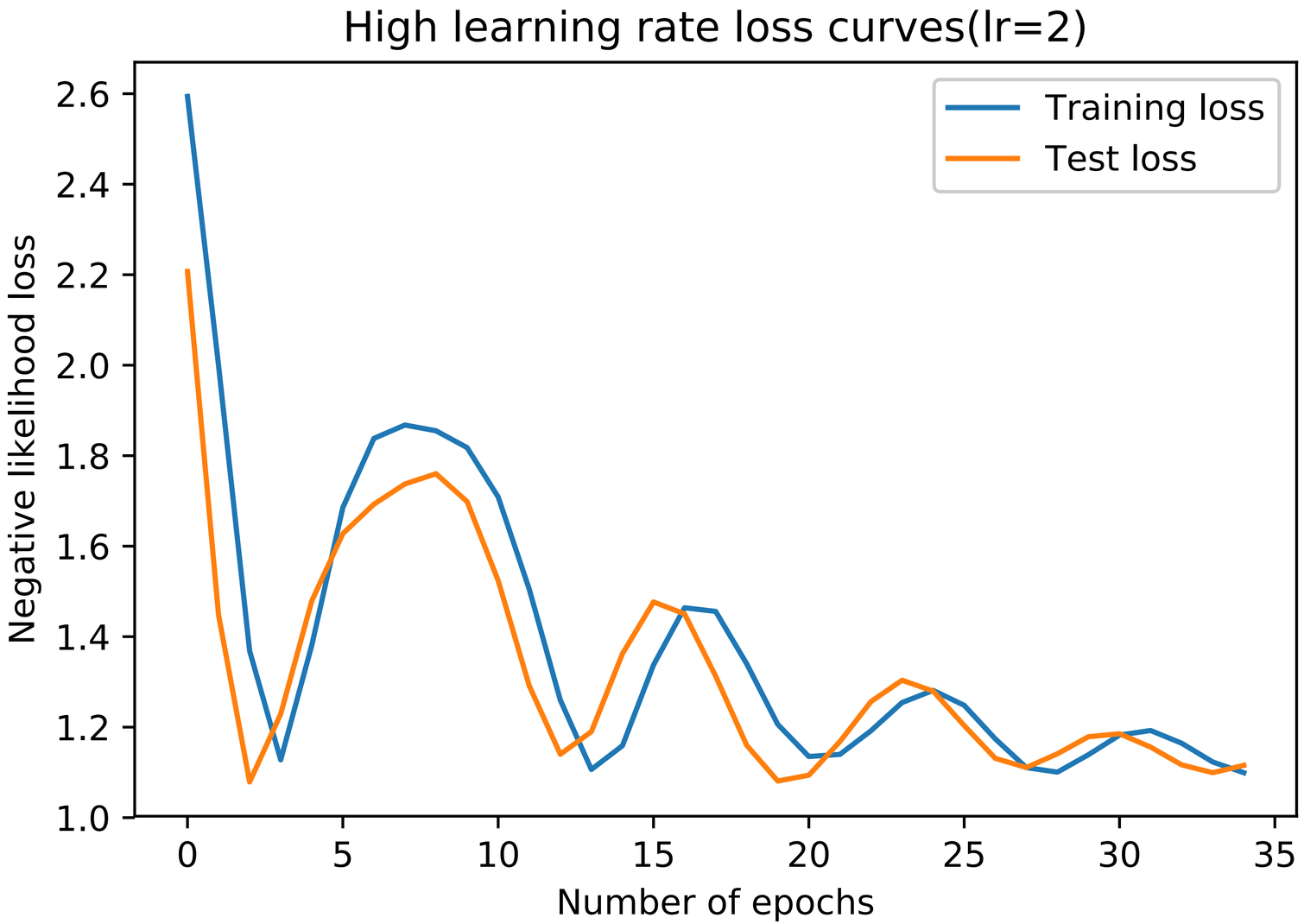
Dense 2

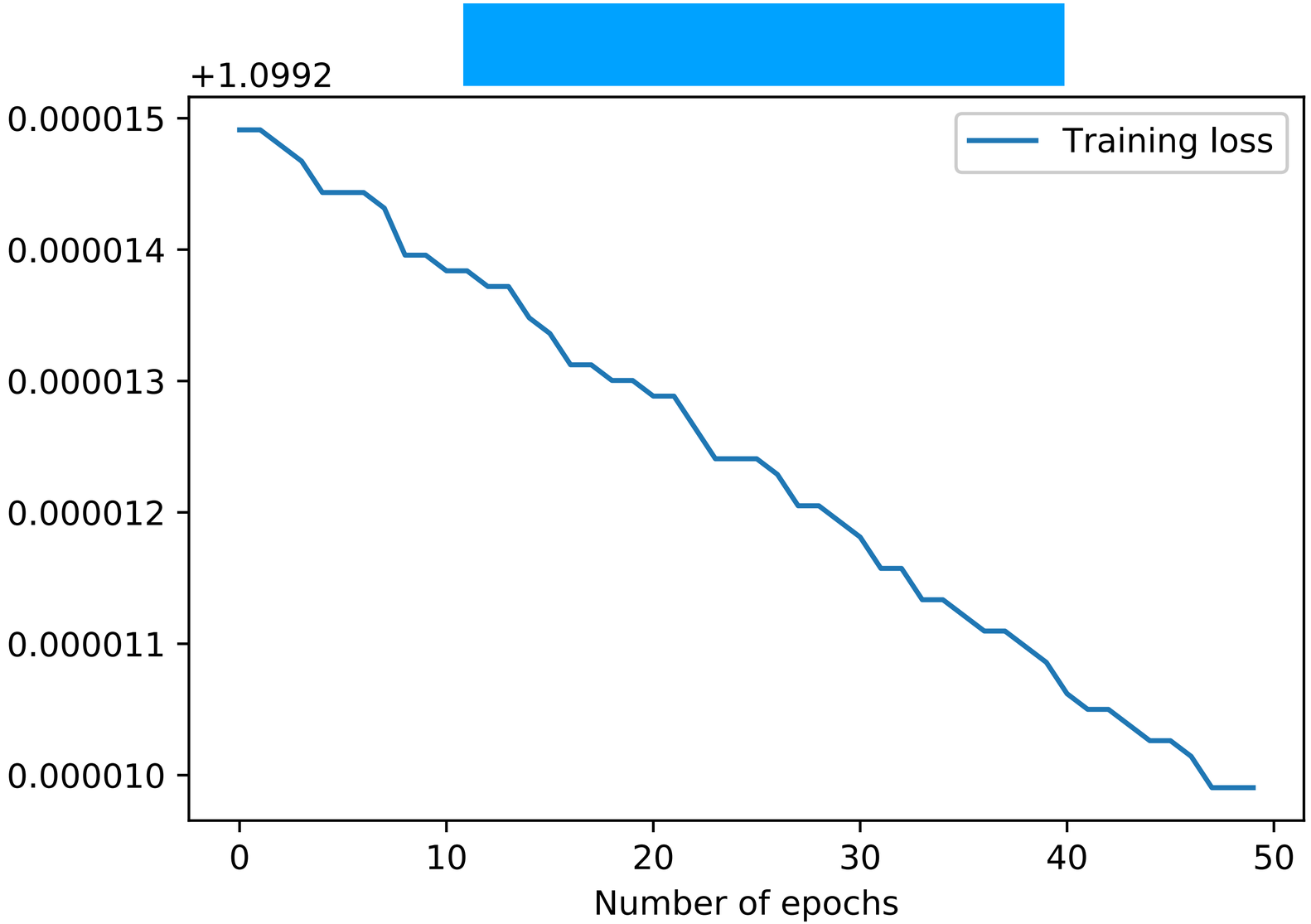
Linear

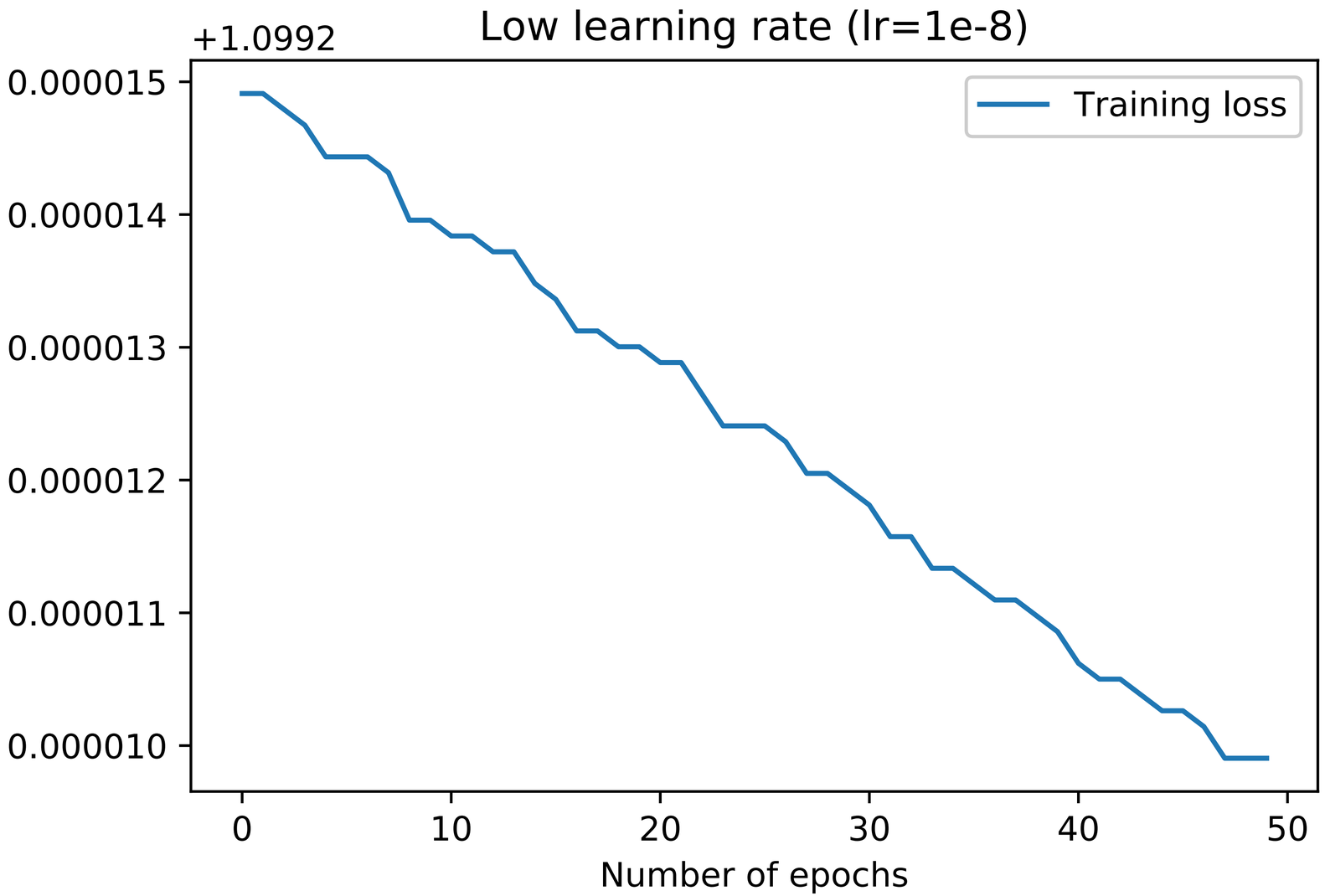
Softmax(3)

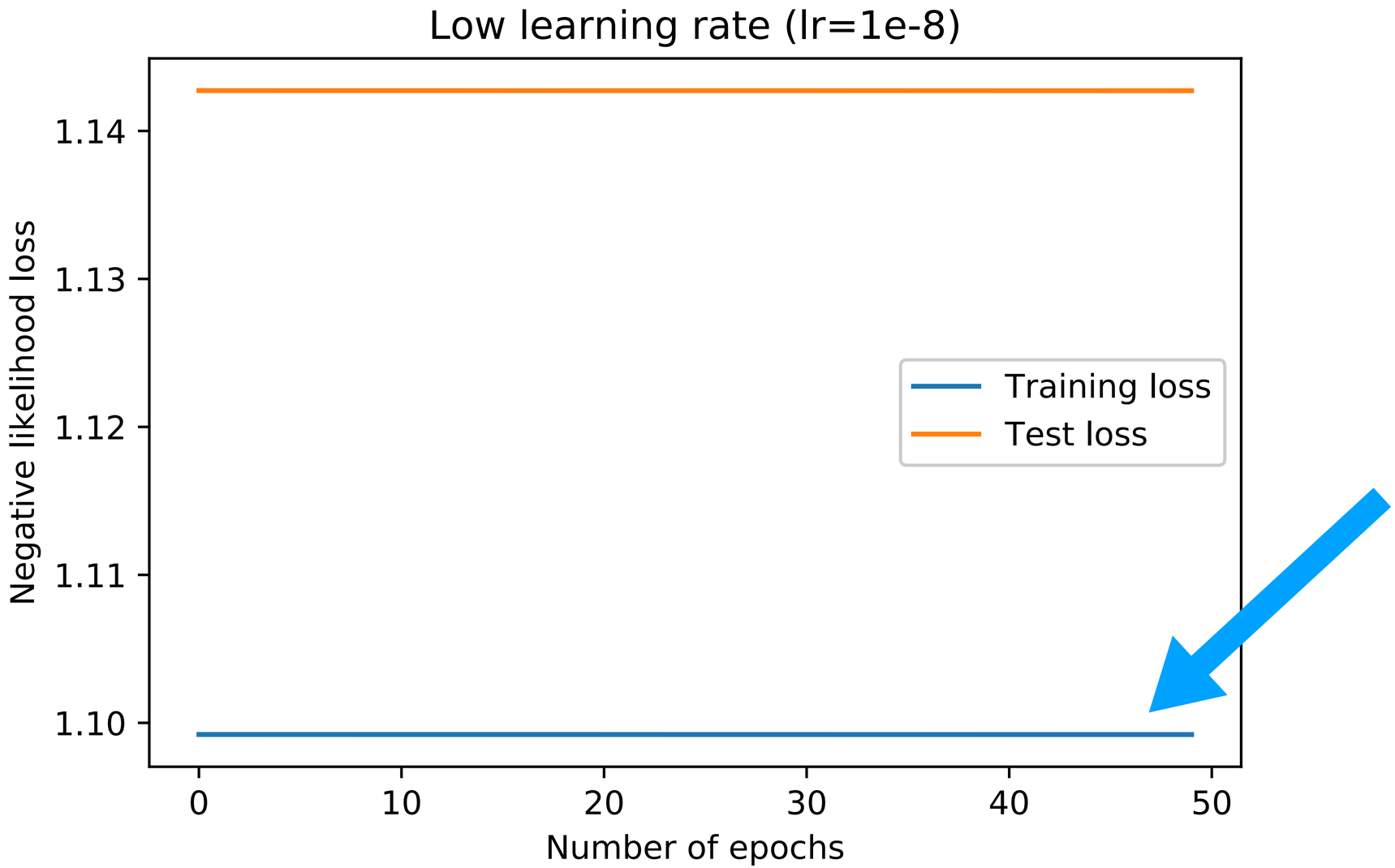
Output

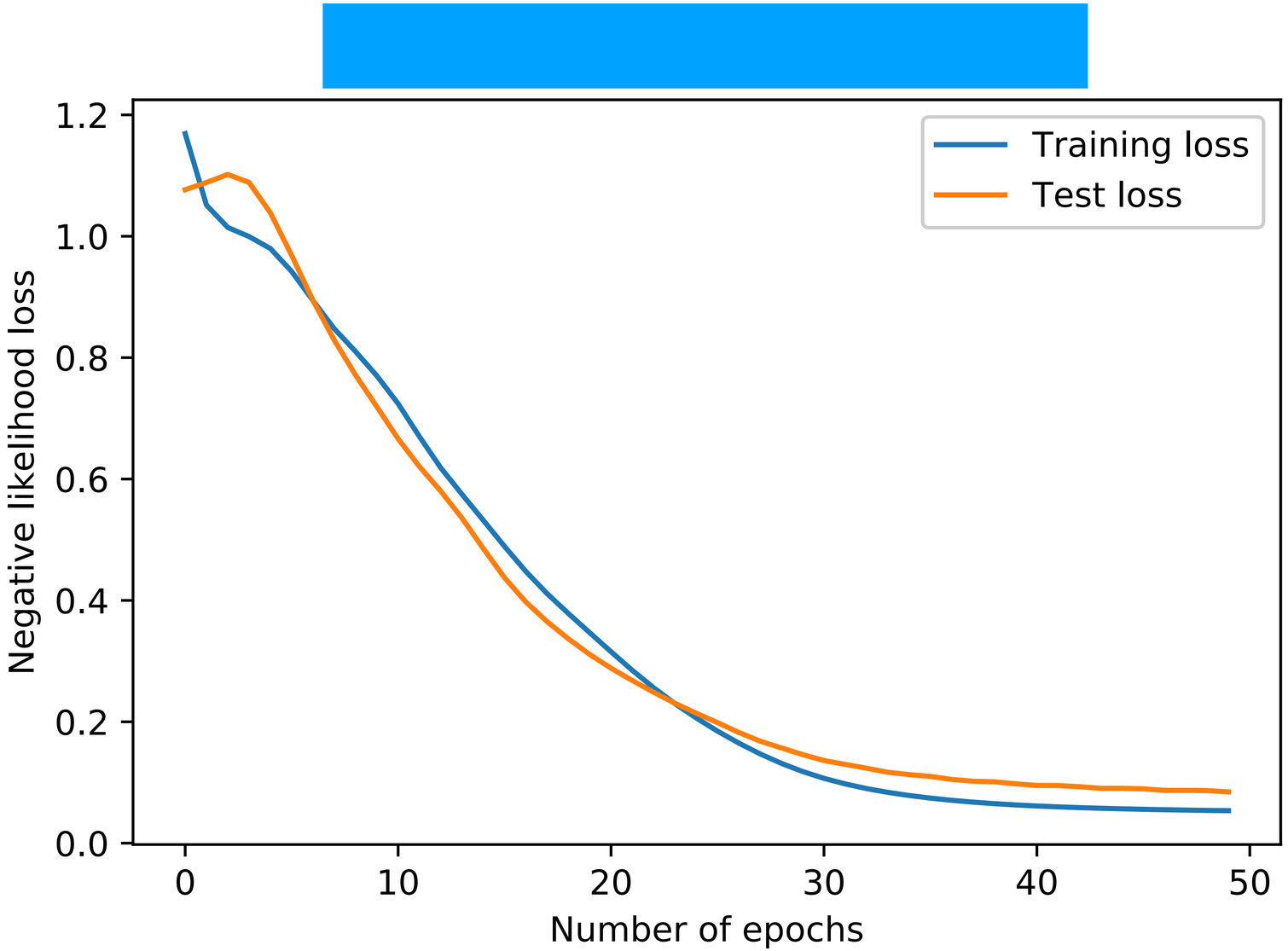

Training loss
Test loss
Negative likelihood loss
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0
5
10
15
20
25
30
35
Number of epochs

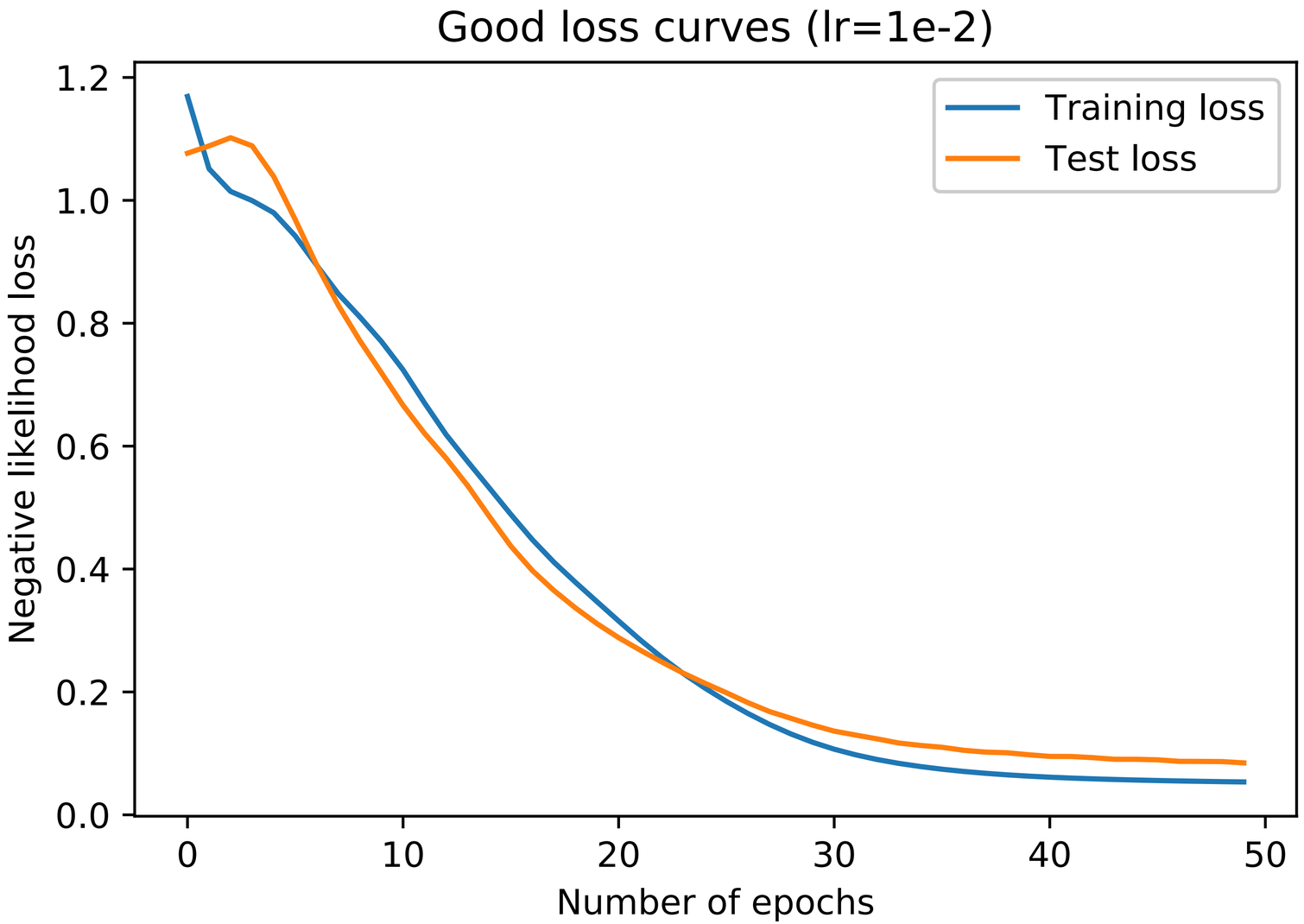

High learning rate loss curves(lr=2)
Training loss
Test loss
Negative likelihood loss
Number of epochs
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0
5
10
15
20
25
30
35

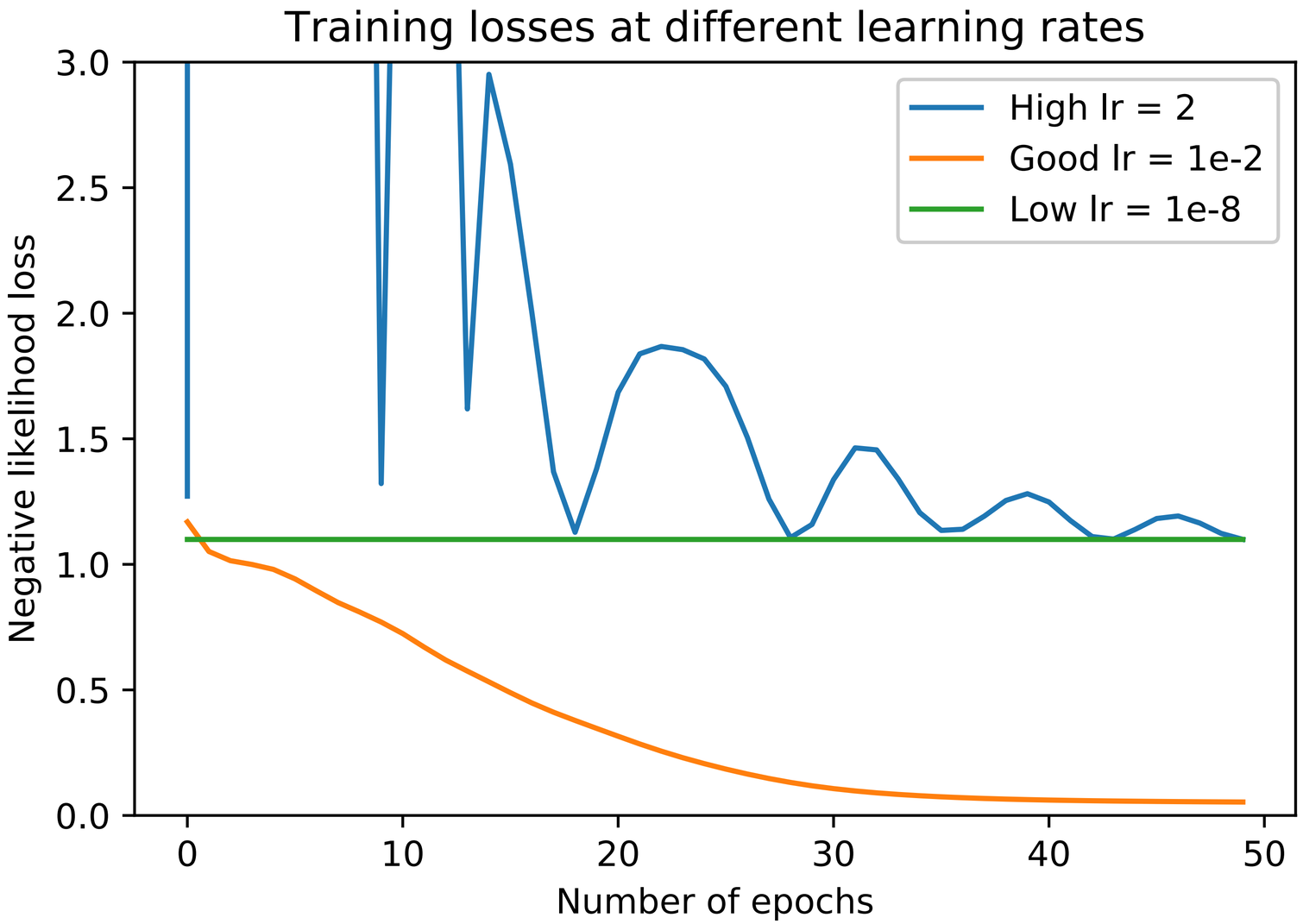

+1.0992
0.000015
0.000014
0.000013
0.000012
0.000011
0.000010
Training loss
0
10
20
30
40
50
Number of epochs


Low learning rate (lr=1e-8)
+1.0992
0.000015
0.000014
0.000013
0.000012
0.000011
0.000010
Training loss
0
10
20
30
40
50
Number of epochs


Low learning rate (lr=1e-8)
Negative likelihood loss
1.14
1.13
1.12
1.11
1.10
0
10
20
30
40
50
Number of epochs
Training loss
Test loss

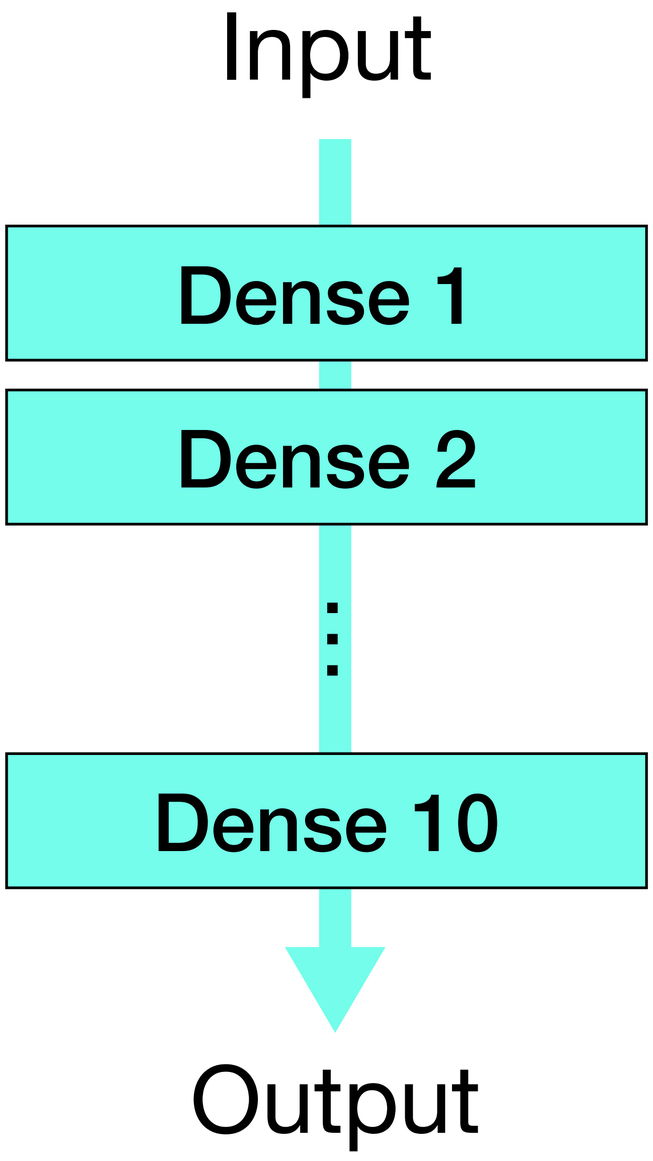

Training loss
Test loss
Negative likelihood loss
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
Number of epochs


Good loss curves (lr=1e-2)
Training loss
Test loss
Negative likelihood loss
Number of epochs
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50

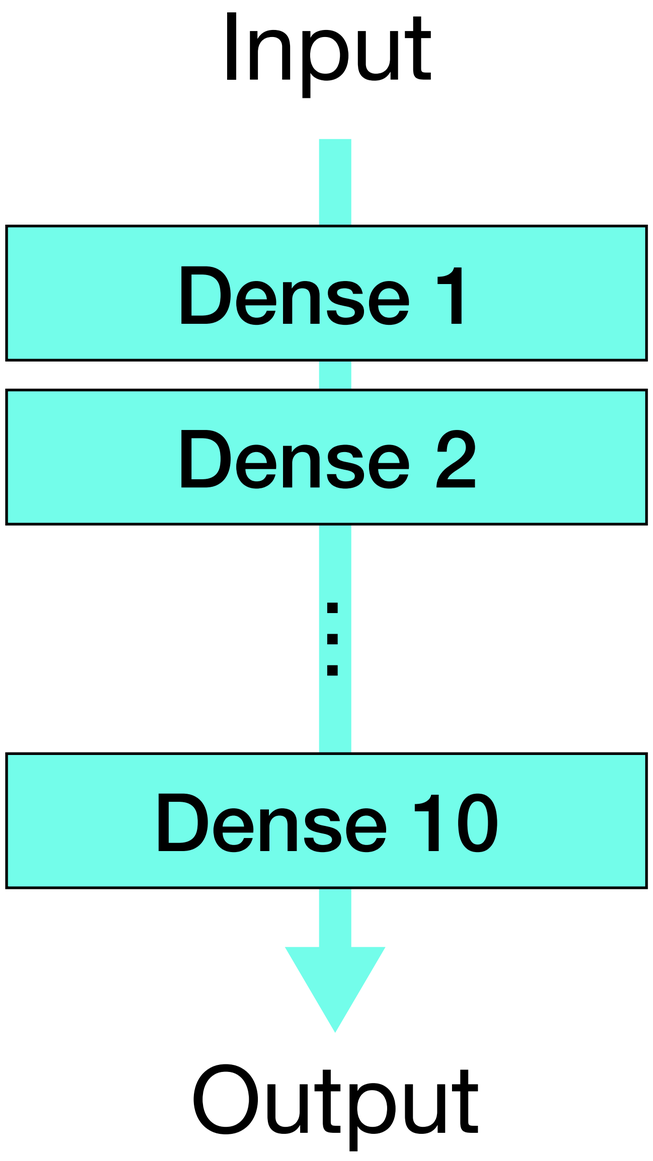

Training losses at different learning rates
Negative likelihood loss
Number of epochs
High lr = 2
Good lr = 1e-2
Low lr = 1e-8
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
10
20
30
40
50

How calculate gradient?

How calculate gradient?

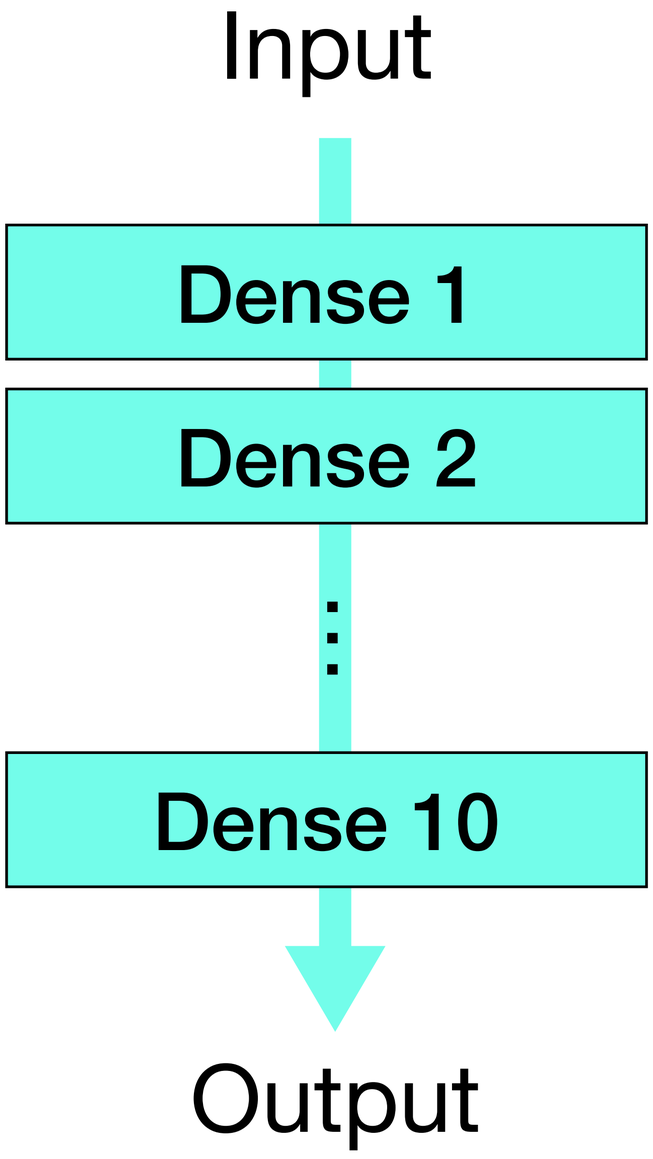


$$\frac{\partial \text{Loss}}{\partial W_1}$$



$$\frac{\partial \mathrm{Loss}}{\partial \mathrm{W}_1}$$

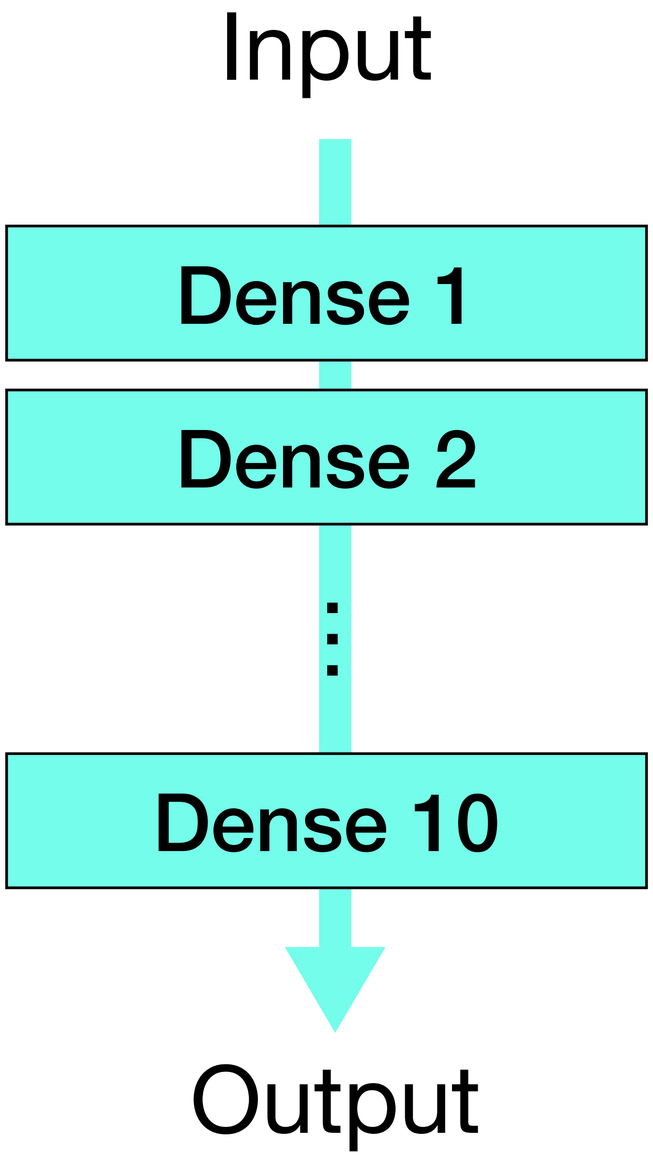
$$= \frac{\partial \mathrm{Loss}}{\partial \mathrm{O}_{10}} \frac{\partial}{\partial \mathrm{W}_1} \big(\sigma(W_{10}\sigma(W_9\sigma(\ldots)))\big)$$



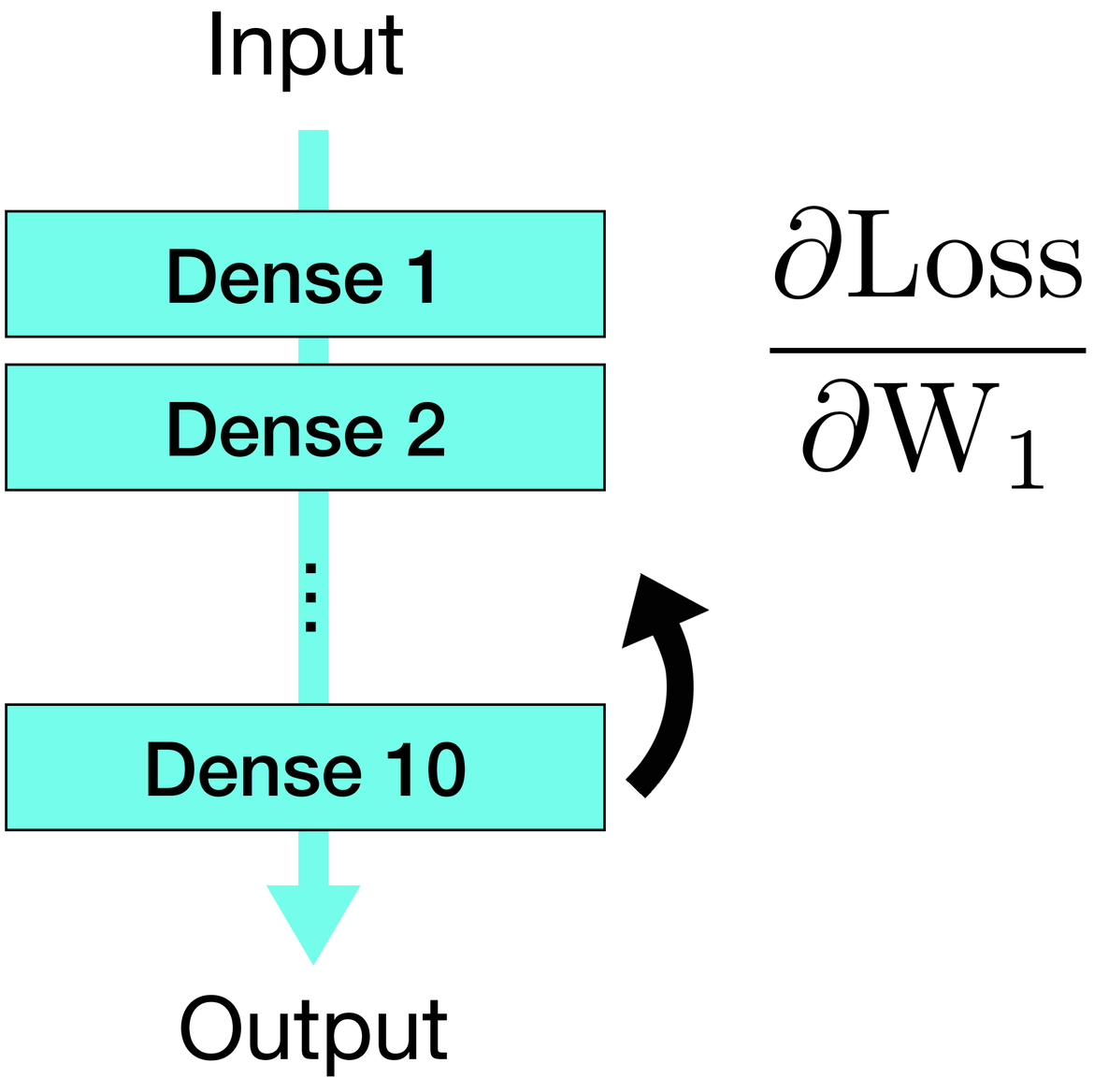
$$\frac{\partial \text{Loss}}{\partial W_1}$$

$$= \frac{\partial \text{Loss}}{\partial O_{10}} \frac{\partial}{\partial W_1} \left(\sigma(W_{10}\sigma(W_9\sigma(\ldots))\right)$$

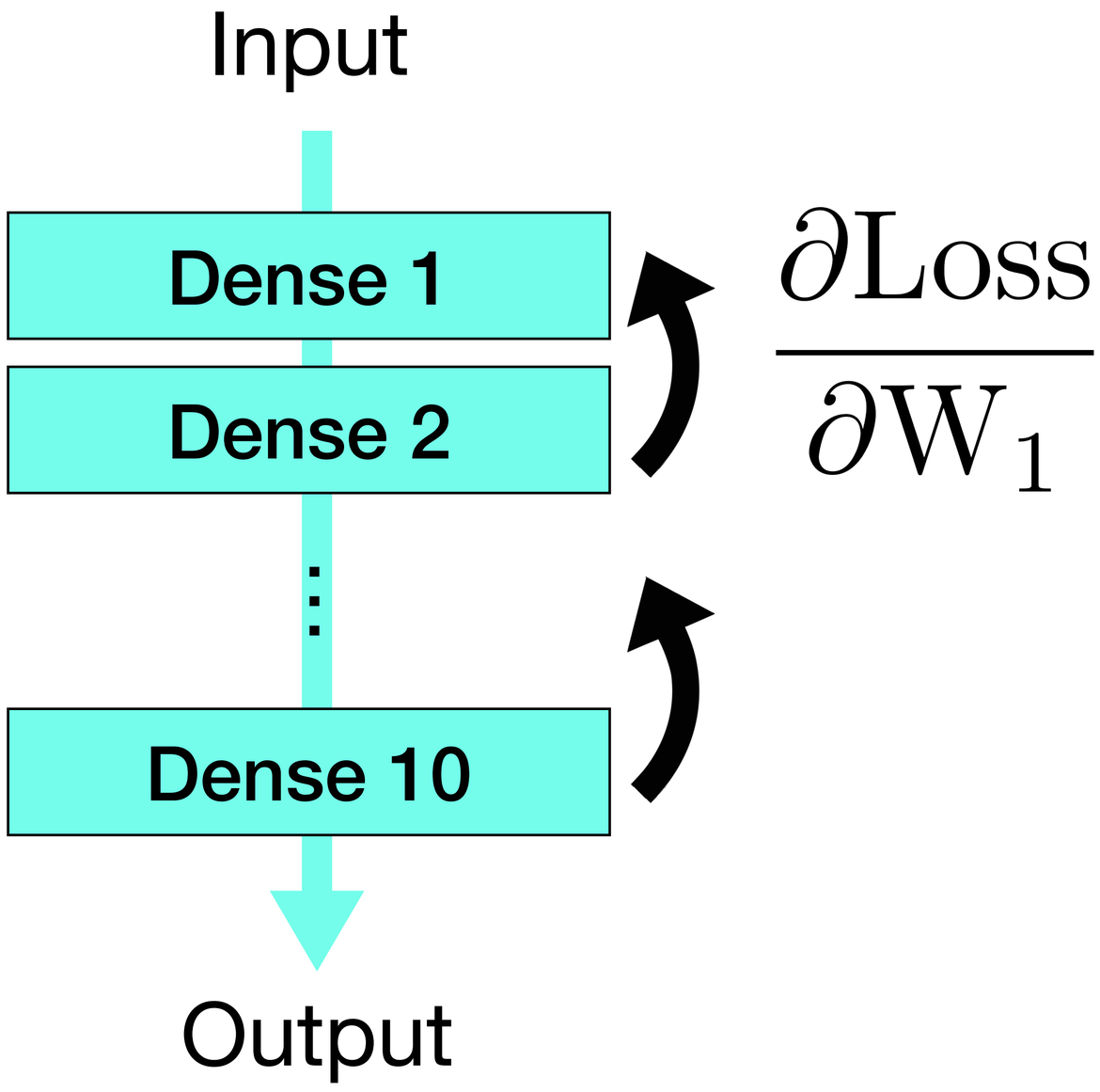
CENSORED

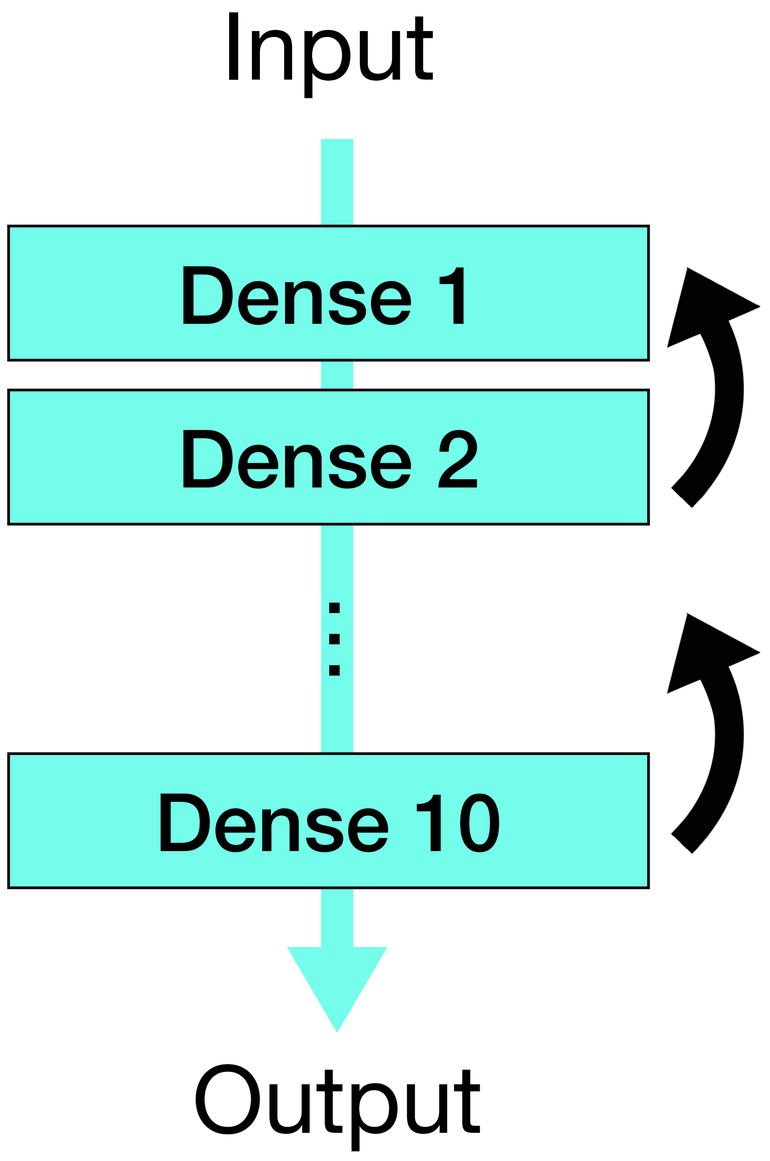


$$\frac{\partial \text{Loss}}{\partial W_1}$$



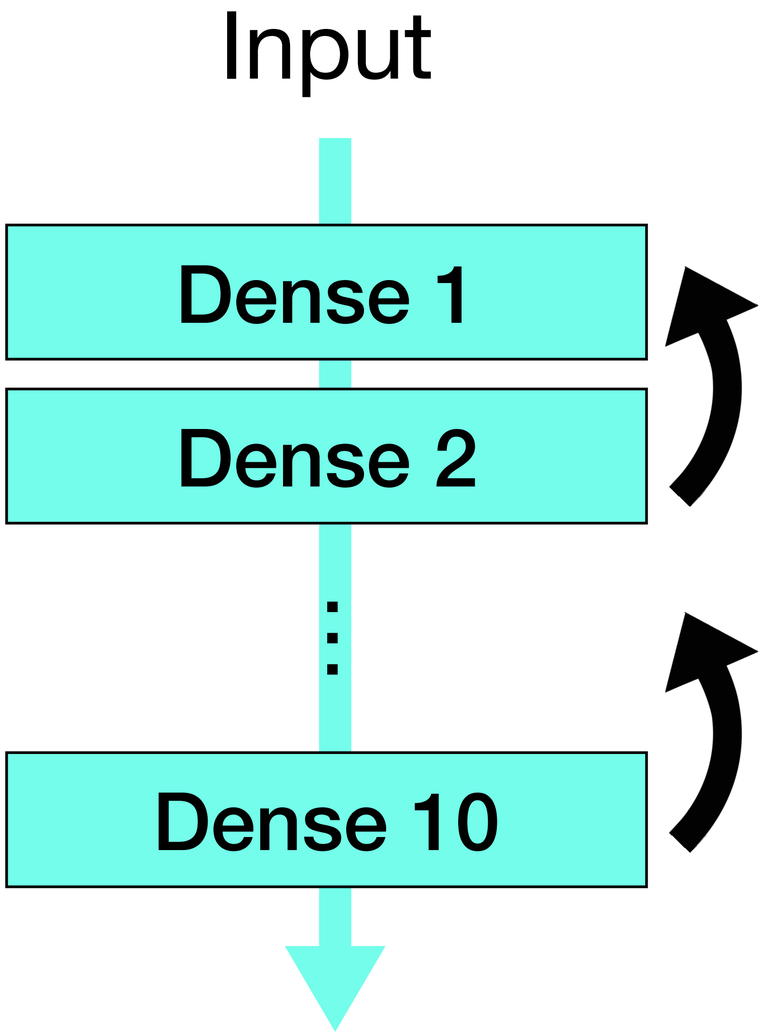
$$\frac{\partial \text{Loss}}{\partial W_1}$$


Input
Dense 1
Dense 2
Dense 10
Output
$\frac{\partial \text{Loss}}{\partial W_1}$



$$\frac{\partial \text{Loss}}{\partial W_1}$$

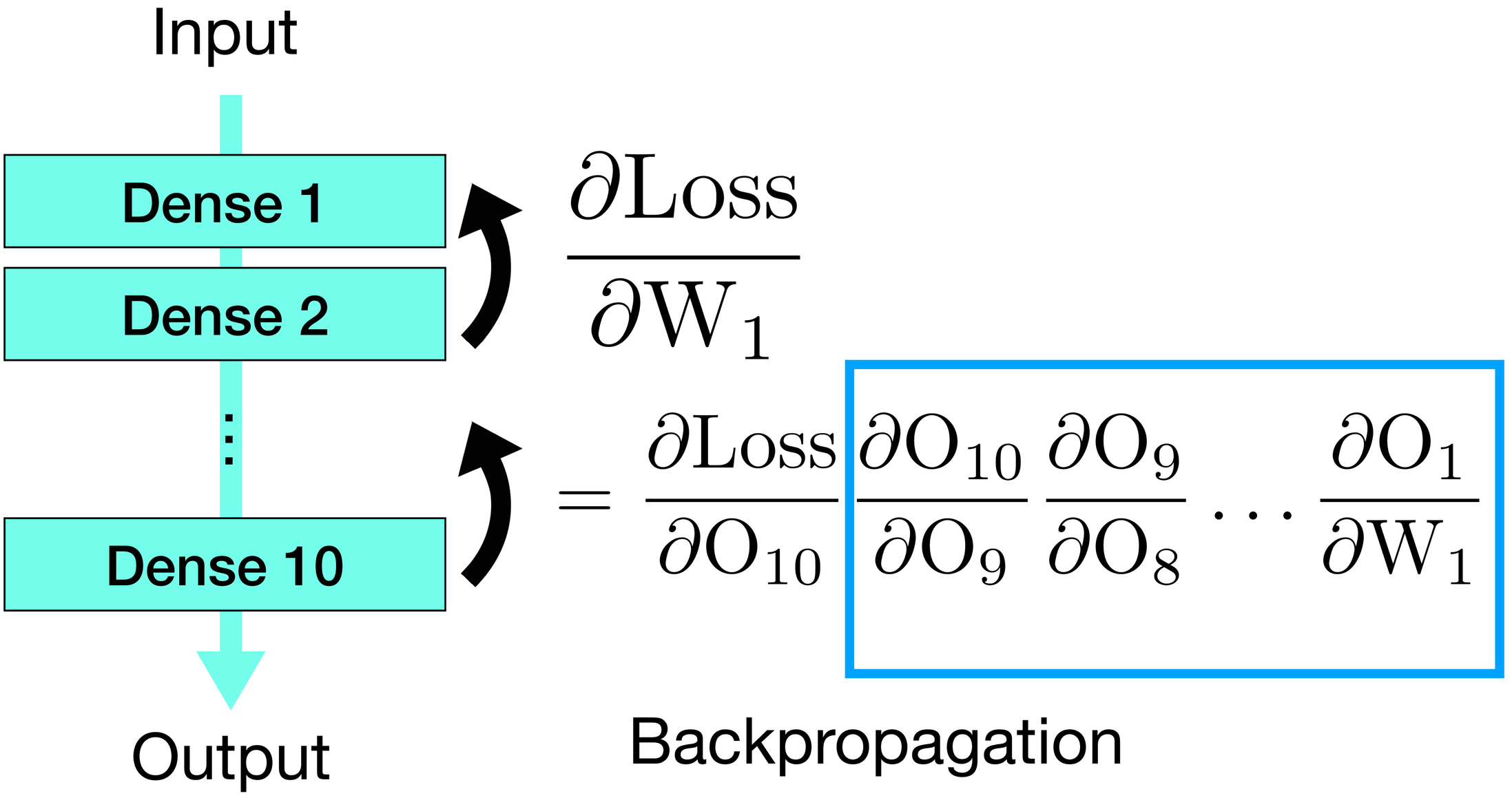
$$= \frac{\partial \text{Loss}}{\partial O_{10}} \frac{\partial O_{10}}{\partial O_9} \frac{\partial O_9}{\partial O_8} \cdots \frac{\partial O_1}{\partial W_1}$$

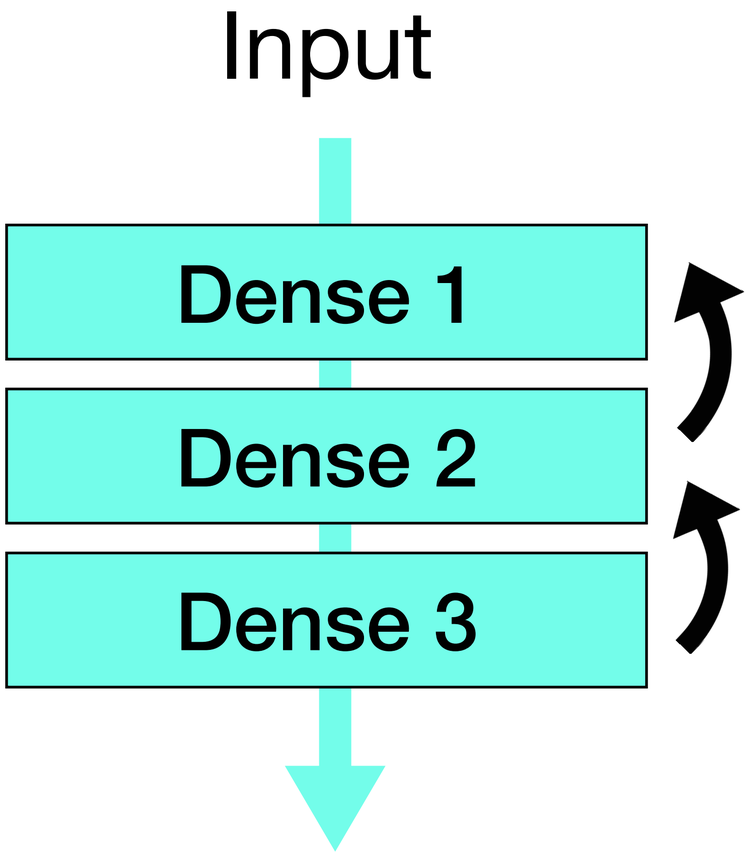


$$\frac{\partial \text{Loss}}{\partial \text{W}_1}$$

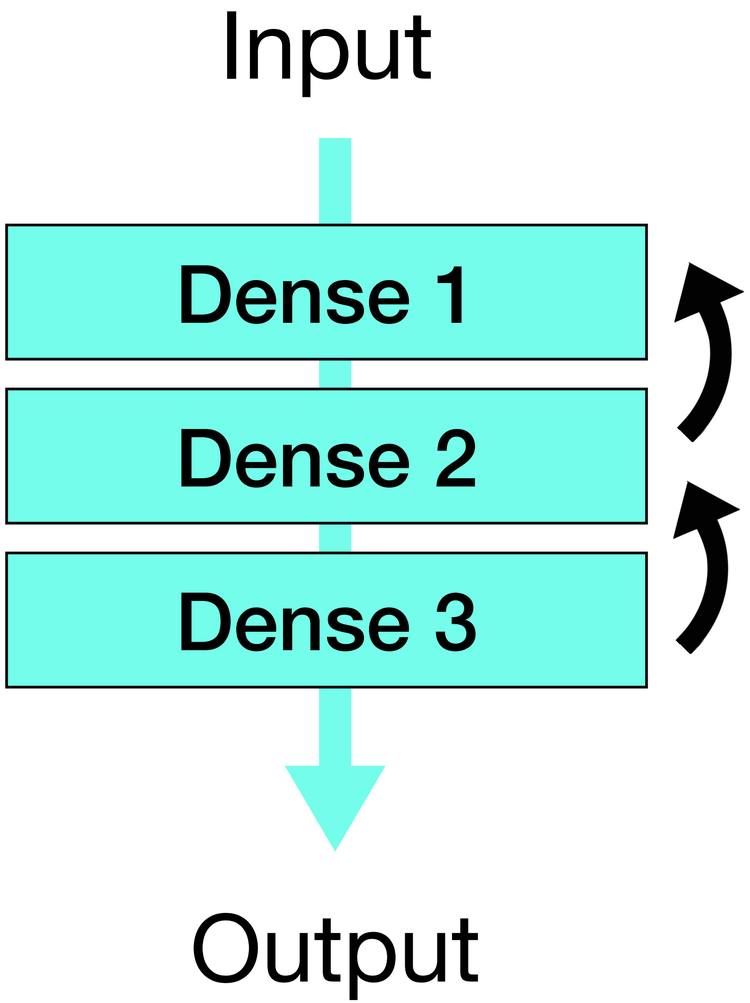
$$= \frac{\partial \text{Loss}}{\partial \text{O}_{10}} \frac{\partial \text{O}_{10}}{\partial \text{O}_9} \frac{\partial \text{O}_9}{\partial \text{O}_8} \cdots \frac{\partial \text{O}_1}{\partial \text{W}_1}$$

Backpropagation


Input
Dense 1
Dense 2
Dense 10
Output
$\frac{\partial \text{Loss}}{\partial W_1}$
$= \frac{\partial \text{Loss}}{\partial O_{10}} \frac{\partial O_{10}}{\partial O_9} \frac{\partial O_9}{\partial O_8} \cdots \frac{\partial O_1}{\partial W_1}$
Backpropagation



$$\frac{\partial O_3}{\partial O_2} \times \frac{\partial O_2}{\partial O_1} \times \frac{\partial O_1}{\partial W_1}$$



$$\boxed{\frac{\partial O_3}{\partial O_2}} \times \frac{\partial O_2}{\partial O_1} \times \frac{\partial O_1}{\partial W_1}$$

$$\frac{\partial O_3}{\partial O_2}$$

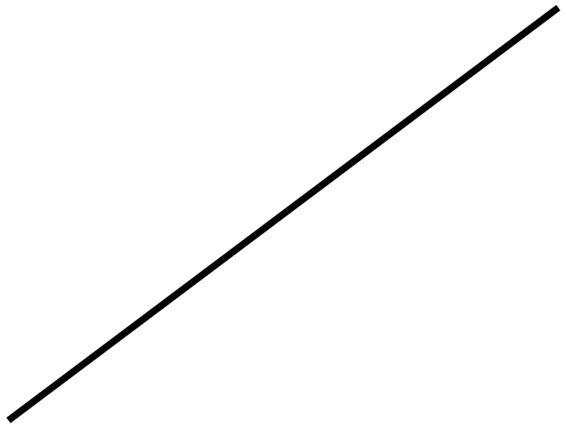
$$\frac{\partial O_3}{\partial O_2}$$

A single layer

$$\frac{\partial O_3}{\partial O_2}$$

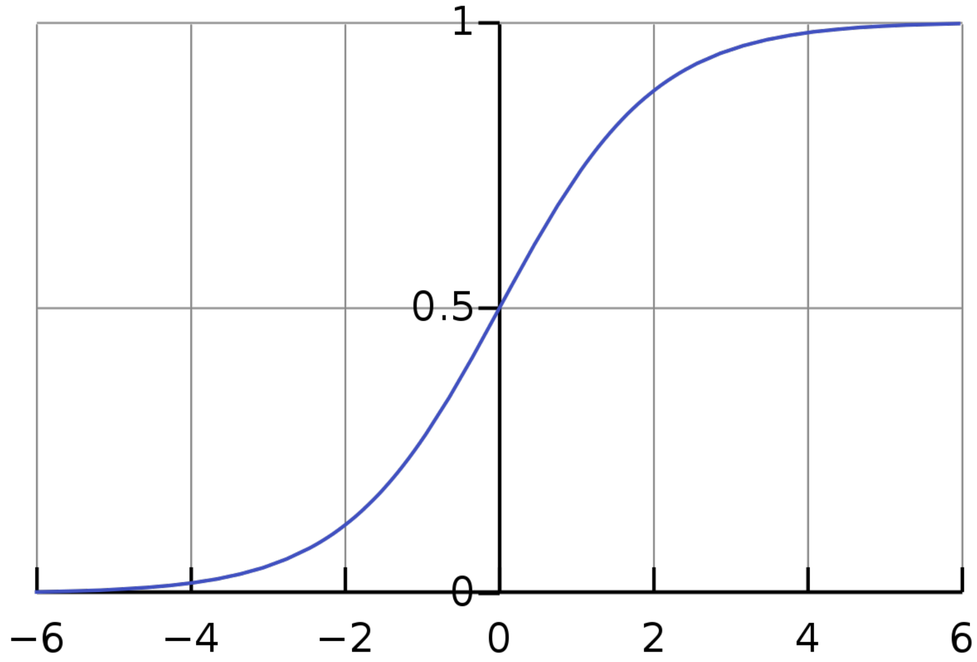
A single layer

Linear layer



$$\mathbf{W}\mathbf{x} + \mathbf{b}$$

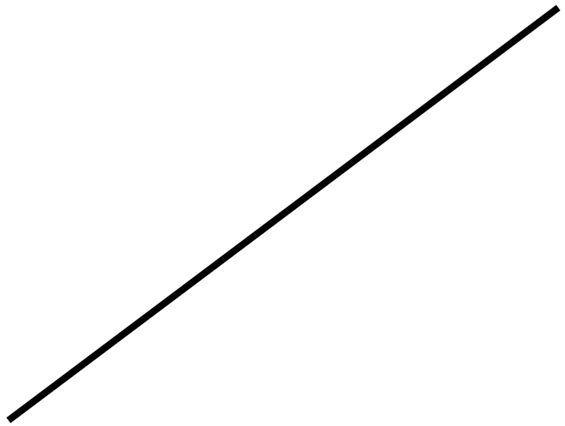
Non-linearity



$$\frac{\partial O_3}{\partial O_2}$$

A single layer

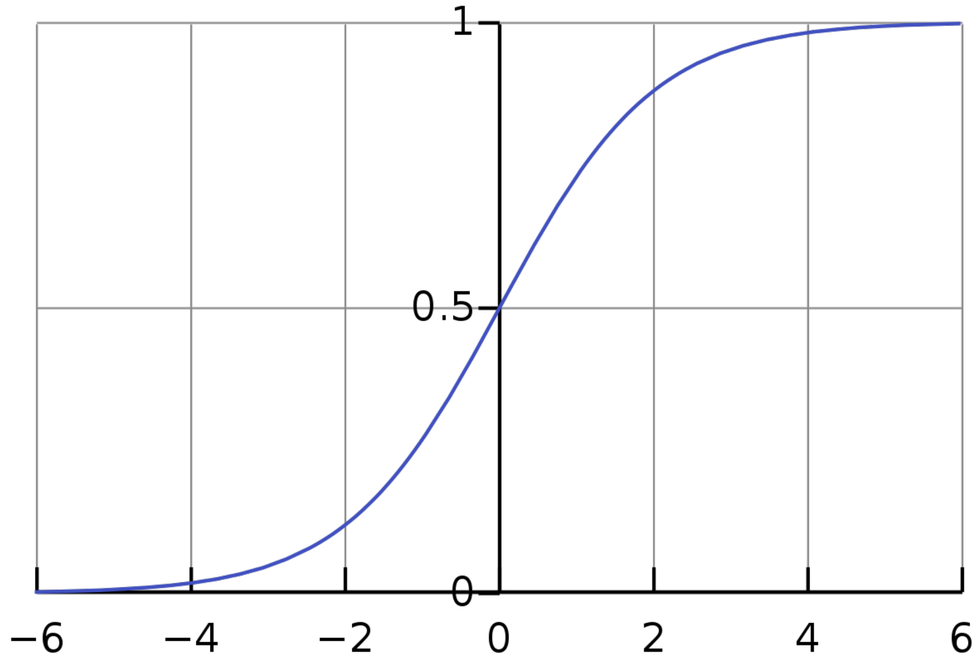
Linear layer



$$\mathbf{W}\mathbf{x} + \mathbf{b}$$

x

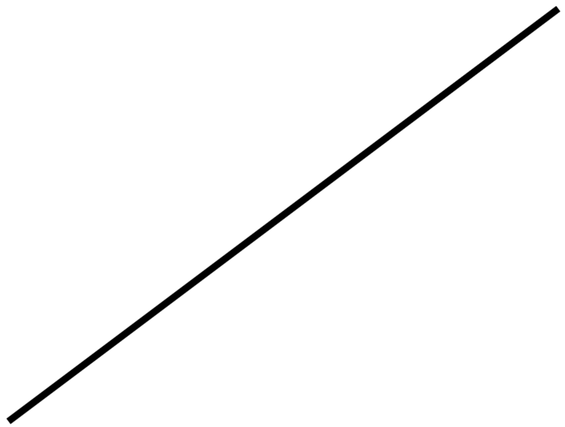
Non-linearity



$$\frac{\partial O_3}{\partial O_2}$$

A single layer

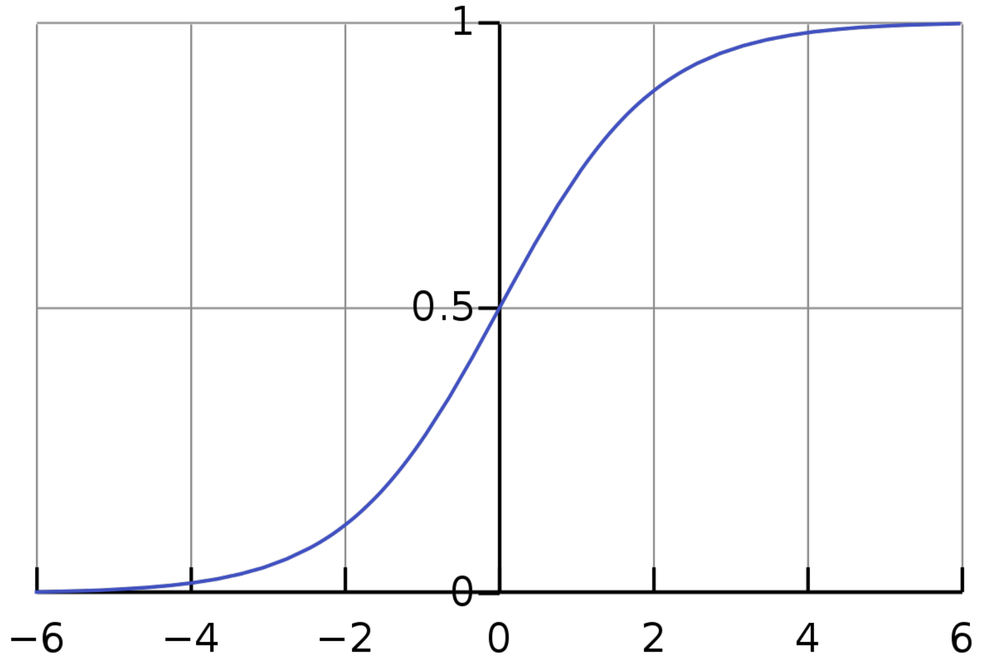
Linear layer

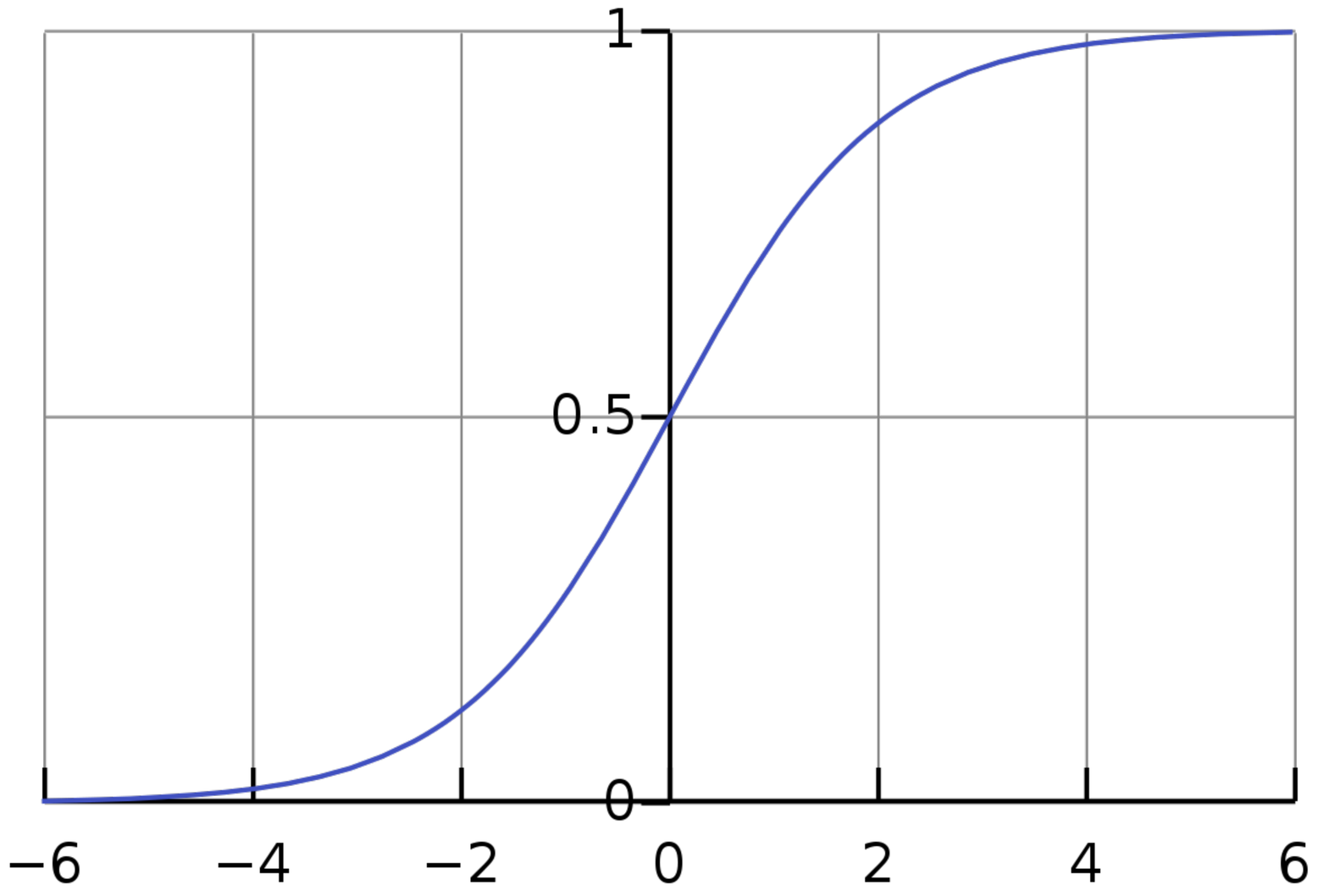


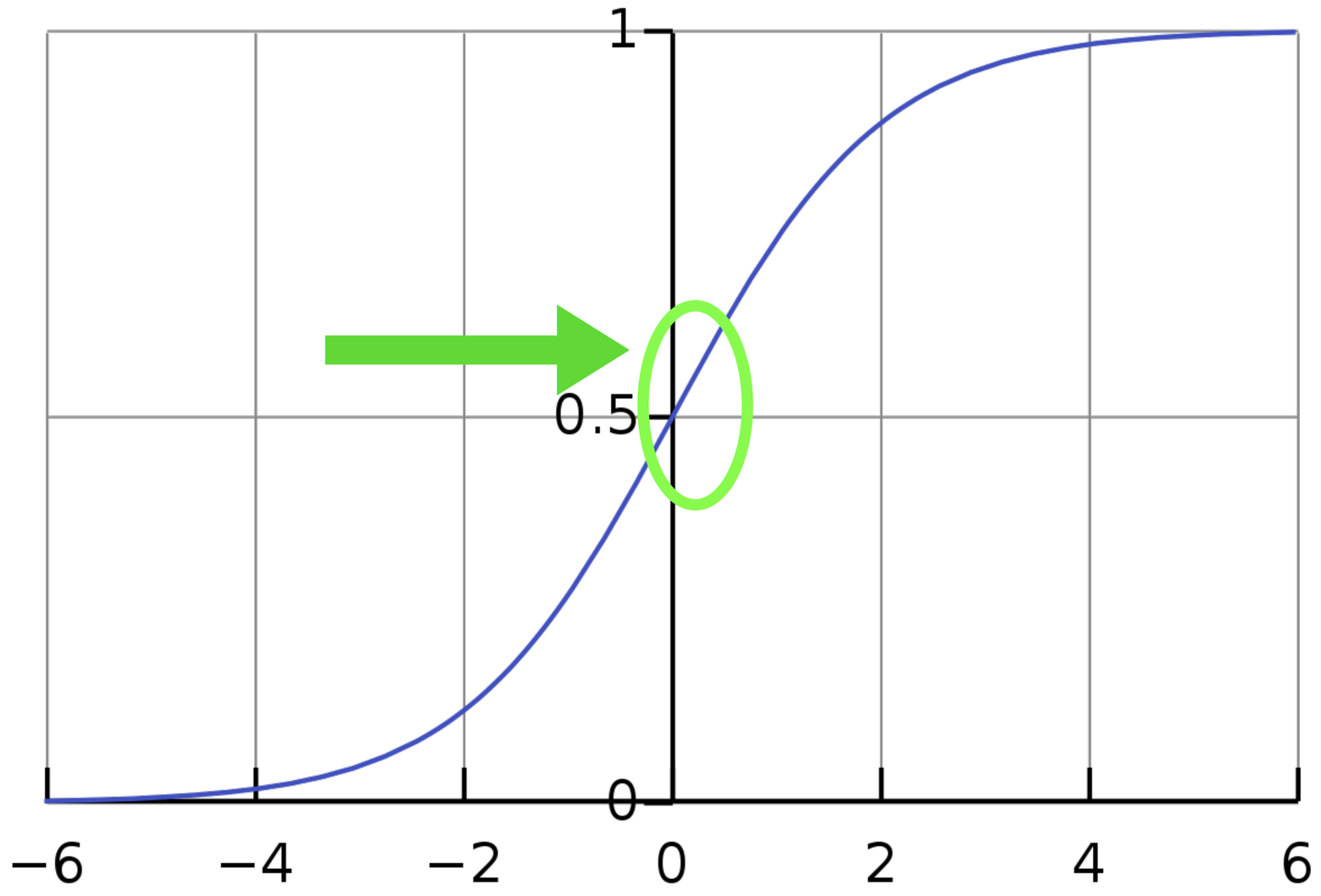
$$\mathbf{W}\mathbf{x} + \mathbf{b}$$

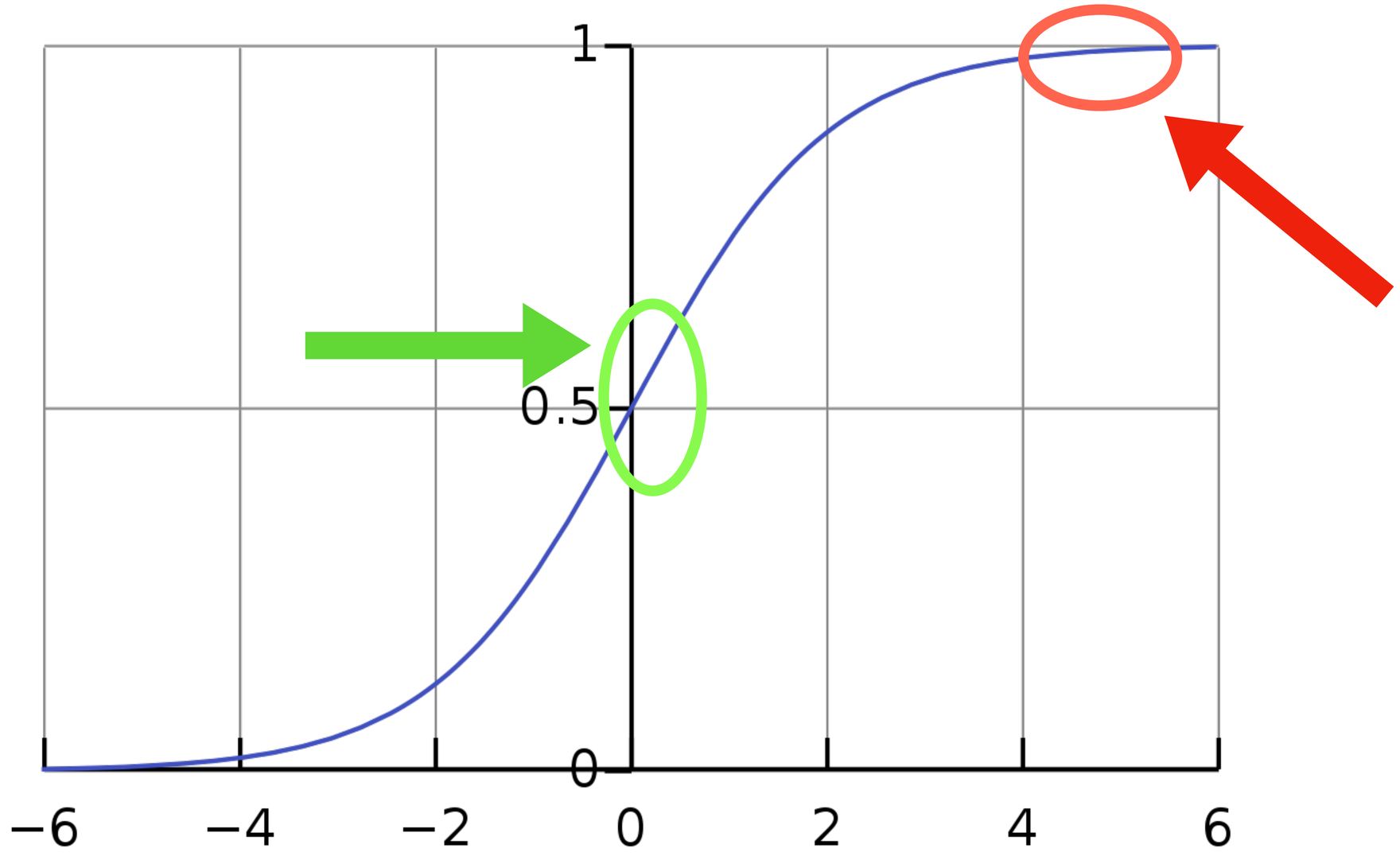
x

Non-linearity




1
0.5
0
−6
−4
−2
0
2
4
6


1
0.5
0
−6
−4
−2
0
2
4
6


1
0.5
0
−6
−4
−2
0
2
4
6

Initialise weights with SMALL variance!

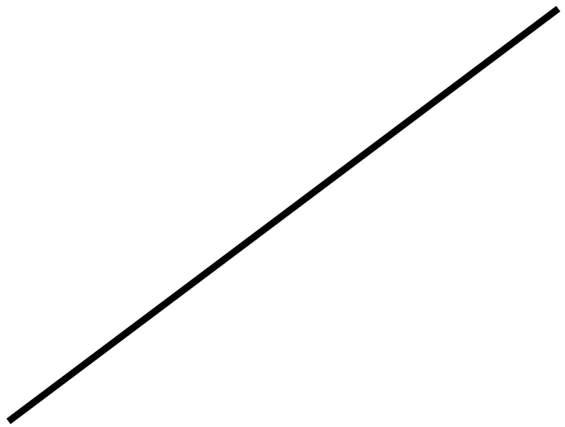
Initialise weights with SMALL variance!

e.g. Glorot initialisation

$$\frac{\partial O_3}{\partial O_2}$$

A single layer

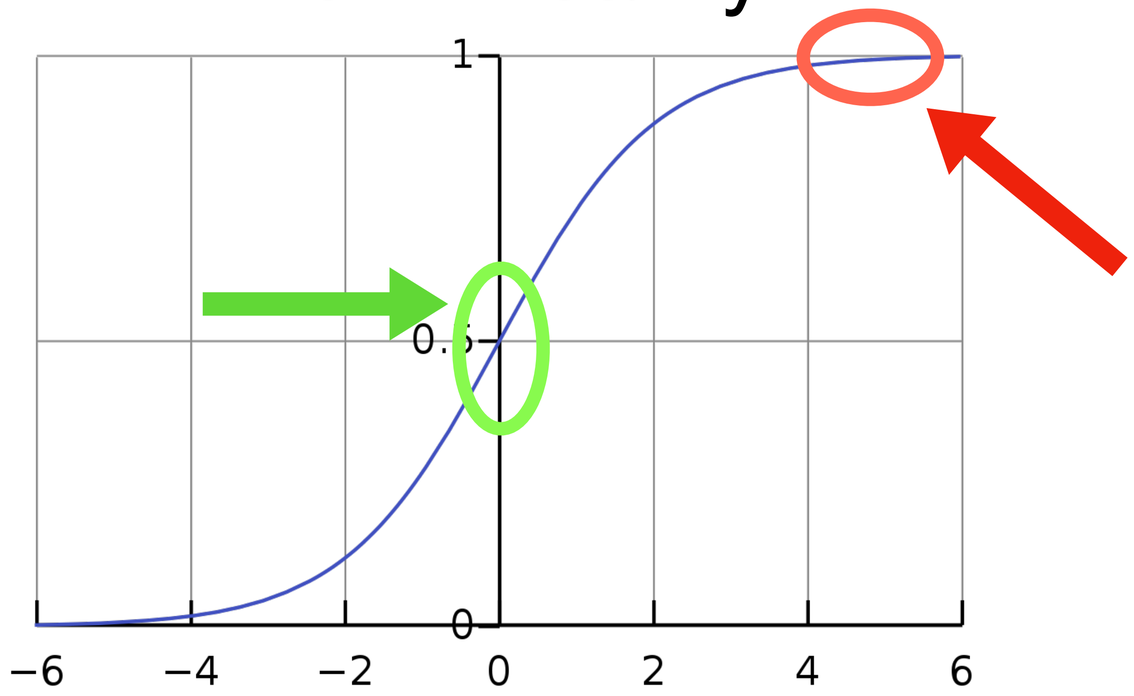
Linear layer



$$\mathbf{W}\mathbf{x} + \mathbf{b}$$

x

Non-linearity



Initialise weights with SMALL variance!

Initialise weights with SMALL variance!

e.g. Glorot initialisation

III. Deeper is better

ImageNet

ImageNet

leopard

ImageNet

sabre-toothed tiger

leopard

cat snow leopard

ImageNet

tree shrew sabre-toothed tiger

leopard

cat snow leopard

ImageNet

tree shrew

sabre-toothed tiger

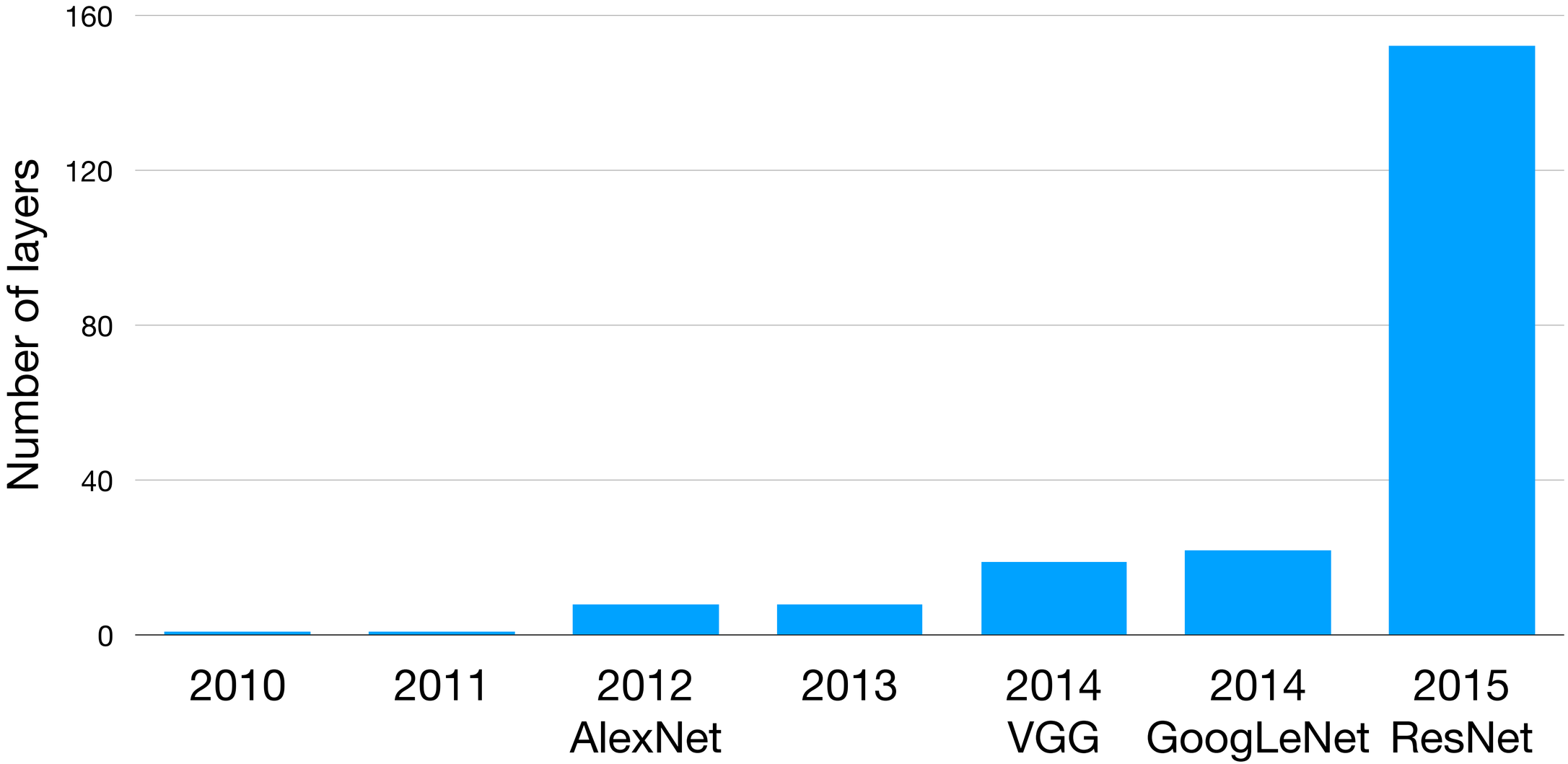
leopard

cat

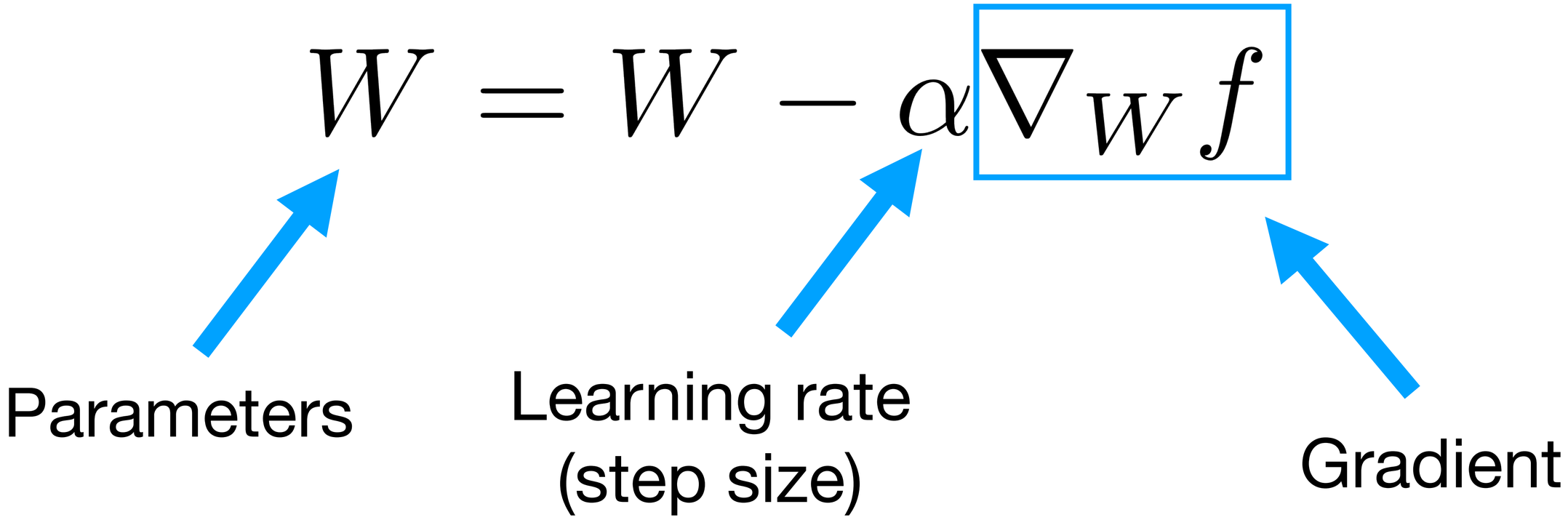
snow leopard

radiator

Revolution of Depth: Top models on ImageNet

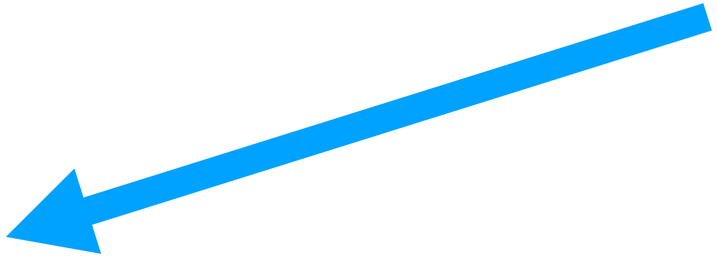


Data from Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. ‘Deep Residual Learning for Image Recognition’, CVPR 2016.


$W = W - \alpha \nabla_W f$
Parameters
Learning rate
(step size)
Gradient

$$W = W - \alpha \nabla_W f$$

$$W = W - \alpha \boxed{\nabla_W f}$$



$$\frac{\partial \text{Loss}}{\partial \text{W}_1} = \frac{\partial \text{Loss}}{\partial \text{O}_{10}} \frac{\partial \text{O}_{10}}{\partial \text{O}_9} \frac{\partial \text{O}_9}{\partial \text{O}_8} \cdots \frac{\partial \text{O}_1}{\partial \text{W}_1}$$

$$W = W - \alpha \nabla_W f$$

$$\frac{\partial \text{Loss}}{\partial \text{W}_1} = \frac{\partial \text{Loss}}{\partial \text{O}_{10}} \frac{\partial \text{O}_{10}}{\partial \text{O}_9} \frac{\partial \text{O}_9}{\partial \text{O}_8} \cdots \frac{\partial \text{O}_1}{\partial \text{W}_1}$$

$$\frac{\partial \mathrm{O}_{100}}{\partial \mathrm{O}_{99}} \times \frac{\partial \mathrm{O}_{99}}{\partial \mathrm{O}_{98}} \times \ldots \times \frac{\partial \mathrm{O}_{2}}{\partial \mathrm{O}_{1}} \times \frac{\partial \mathrm{O}_{1}}{\partial \mathrm{W}_{1}}$$

$$\frac{\partial O_{100}}{\partial O_{99}} = \boxed{0.1} \times \frac{\partial O_{99}}{\partial O_{98}} \times \ldots \times \frac{\partial O_2}{\partial O_1} \times \frac{\partial O_1}{\partial W_1}$$

$$\boxed{0.1} \times \boxed{0.1} \times \ldots \times \frac{\partial O_2}{\partial O_1} \times \frac{\partial O_1}{\partial W_1}$$

$$\boxed{0.1} \times \boxed{0.1} \times \cdots \times \boxed{0.1} \times \frac{\partial O_1}{\partial W_1}$$

$$\boxed{0.1} \times \boxed{0.1} \times \cdots \times \boxed{0.1} \times \boxed{0.1}$$

$$\boxed{0.1} \times \boxed{0.1} \times \cdots \times \boxed{0.1} \times \boxed{0.1}$$

$$= 10^{-100}$$

Vanishing gradients

$$\boxed{0.1} \times \boxed{0.1} \times \cdots \times \boxed{0.1} \times \boxed{0.1}$$

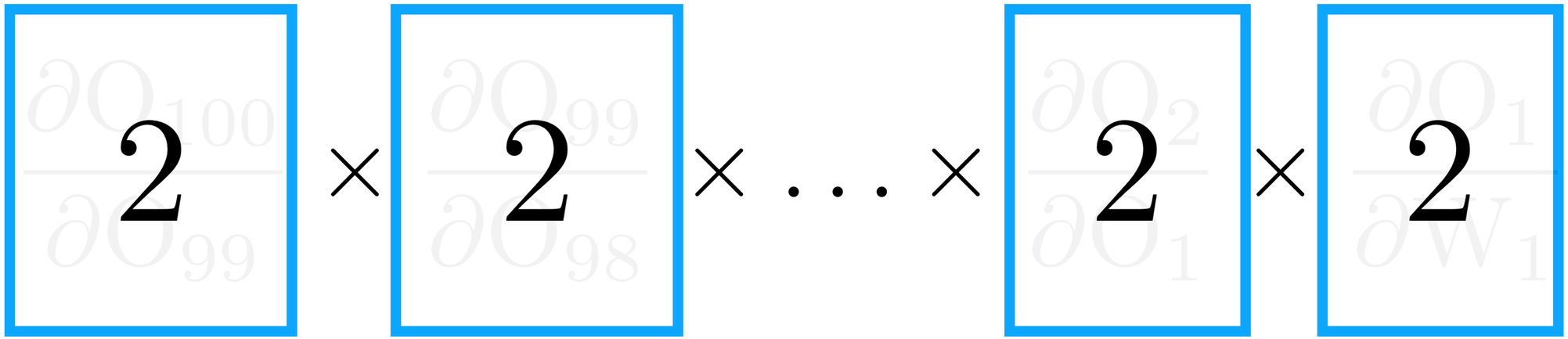
$$= 10^{-100}$$

$$\frac{\partial \mathrm{O}_{100}}{\partial \mathrm{O}_{99}} \times \frac{\partial \mathrm{O}_{99}}{\partial \mathrm{O}_{98}} \times \ldots \times \frac{\partial \mathrm{O}_{2}}{\partial \mathrm{O}_{1}} \times \frac{\partial \mathrm{O}_{1}}{\partial \mathrm{W}_{1}}$$

$$\boxed{\frac{\partial \mathrm{O}_{100}}{\partial \mathrm{O}_{99}} = 2} \times \frac{\partial \mathrm{O}_{99}}{\partial \mathrm{O}_{98}} \times \ldots \times \frac{\partial \mathrm{O}_{2}}{\partial \mathrm{O}_{1}} \times \frac{\partial \mathrm{O}_{1}}{\partial \mathrm{W}_{1}}$$

$$\boxed{\frac{\partial \mathrm{O}_{100}}{\partial \mathrm{O}_{99}}} \times \boxed{\frac{\partial \mathrm{O}_{99}}{\partial \mathrm{O}_{98}}} \times \ldots \times \frac{\partial \mathrm{O}_2}{\partial \mathrm{O}_1} \times \frac{\partial \mathrm{O}_1}{\partial \mathrm{W}_1}$$

$$\boxed{\frac{\partial O_{100}}{\partial O_{99}}} \times \boxed{\frac{\partial O_{99}}{\partial O_{98}}} \times \cdots \times \boxed{\frac{\partial O_{2}}{\partial O_{1}}} \times \frac{\partial O_1}{\partial W_1}$$


2
2
2
2

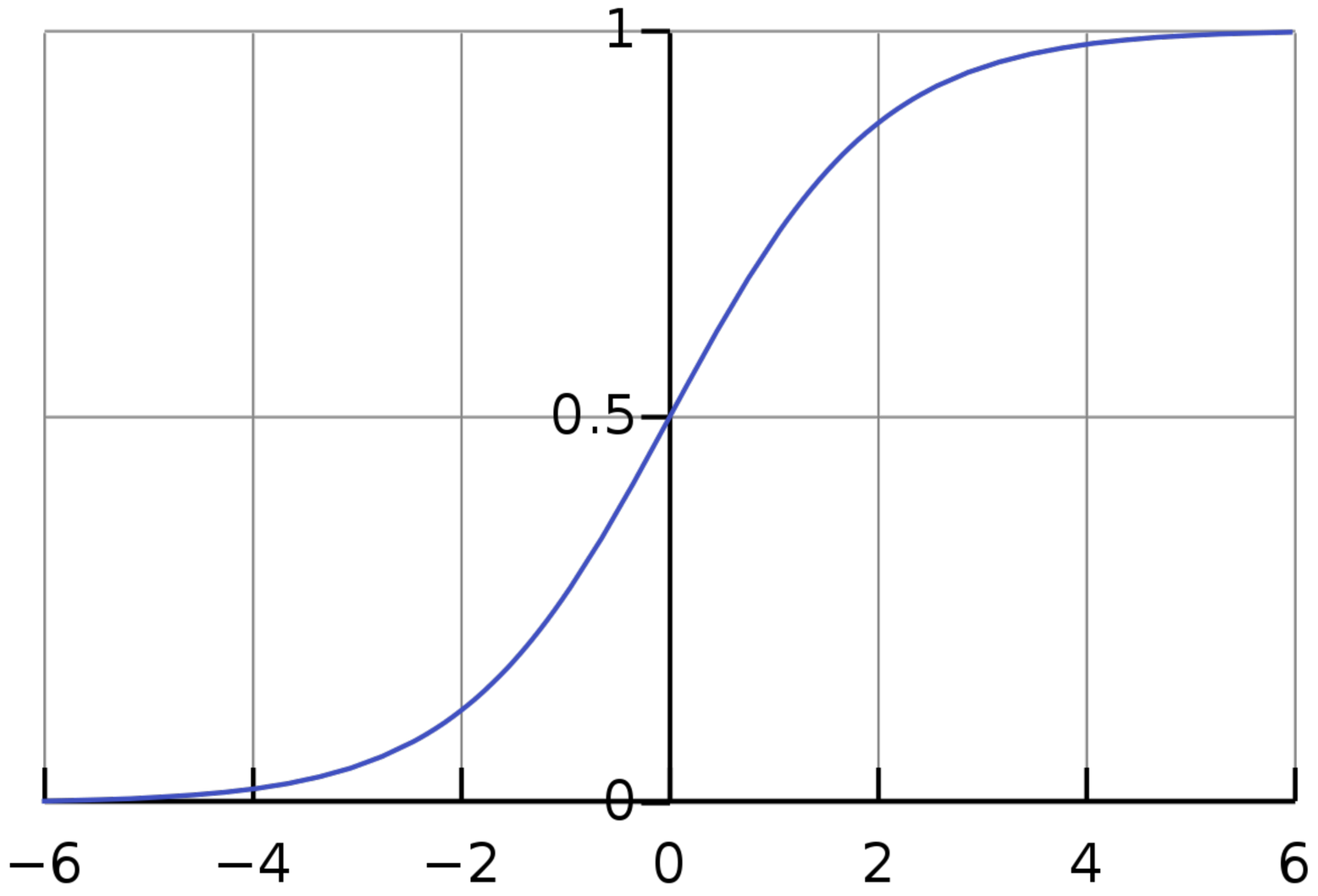
$$\boxed{2} \times \boxed{2} \times \cdots \times \boxed{2} \times \boxed{2}$$

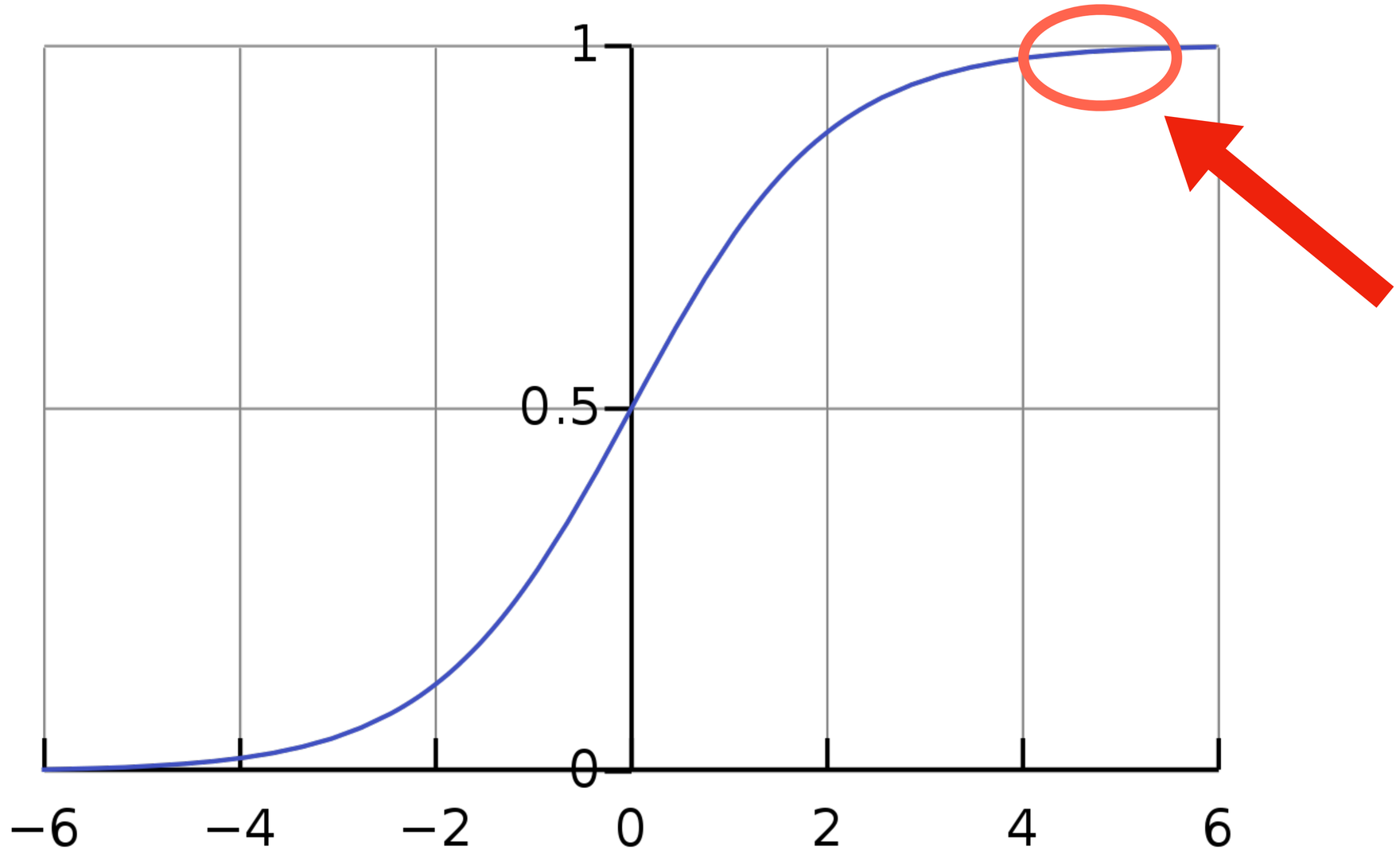
$$> 10^{30}$$

Exploding gradients

$$\boxed{2} \times \boxed{2} \times \ldots \times \boxed{2} \times \boxed{2}$$

$$> 10^{30}$$


1
0.5
0
−6
−4
−2
0
2
4
6


1
0.5
0
−6
−4
−2
0
2
4
6

Vanishing gradients

$$\boxed{0.1} \times \boxed{0.1} \times \cdots \times \boxed{0.1} \times \boxed{0.1}$$

$$= 10^{-100}$$

Vanishing gradients

$$\boxed{0.1} \times \boxed{\frac{\partial O_{100}}{\partial O_{99}}} \times \cdots \times \boxed{0.1} \times \boxed{0.1}$$

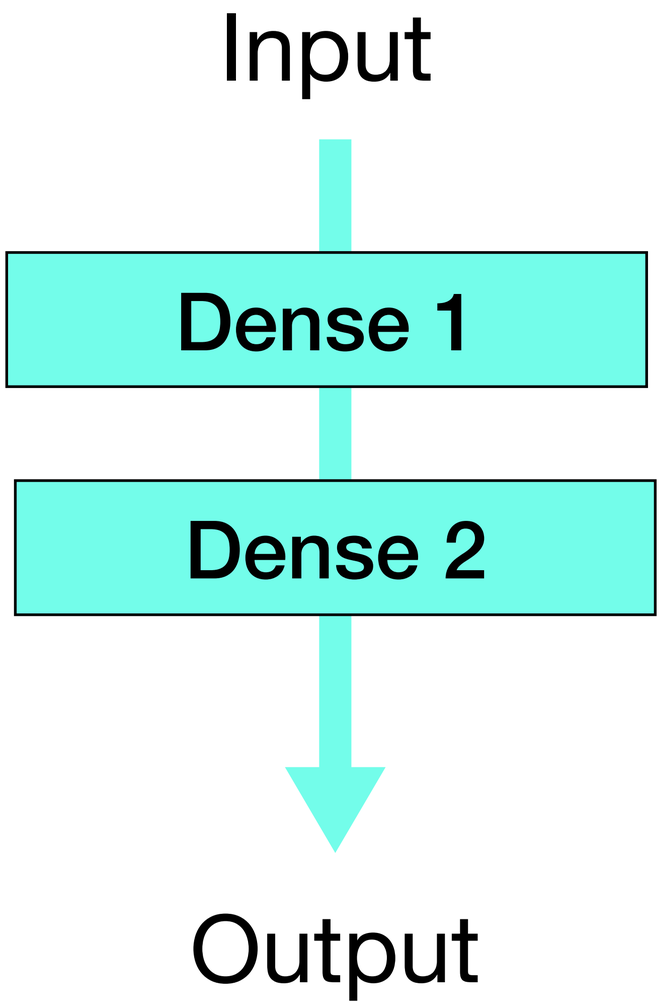
$$= 10^{-100}$$

Vanishing gradients

$$0.1 \times 1 \times \dots \times 0.1 \times 0.1$$

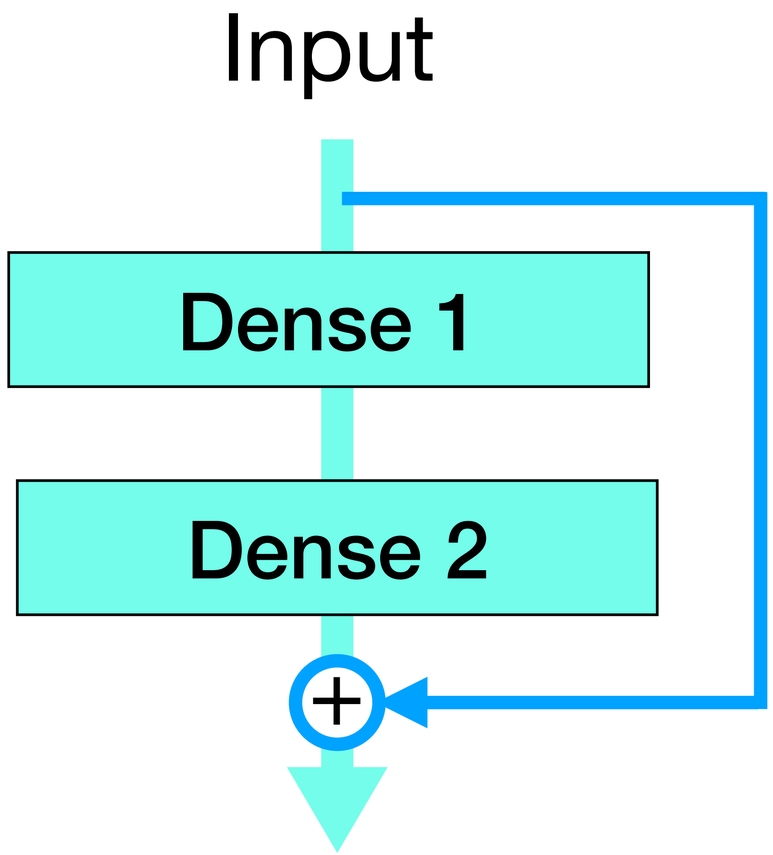
$$= 10^{-100}$$

Skip connections



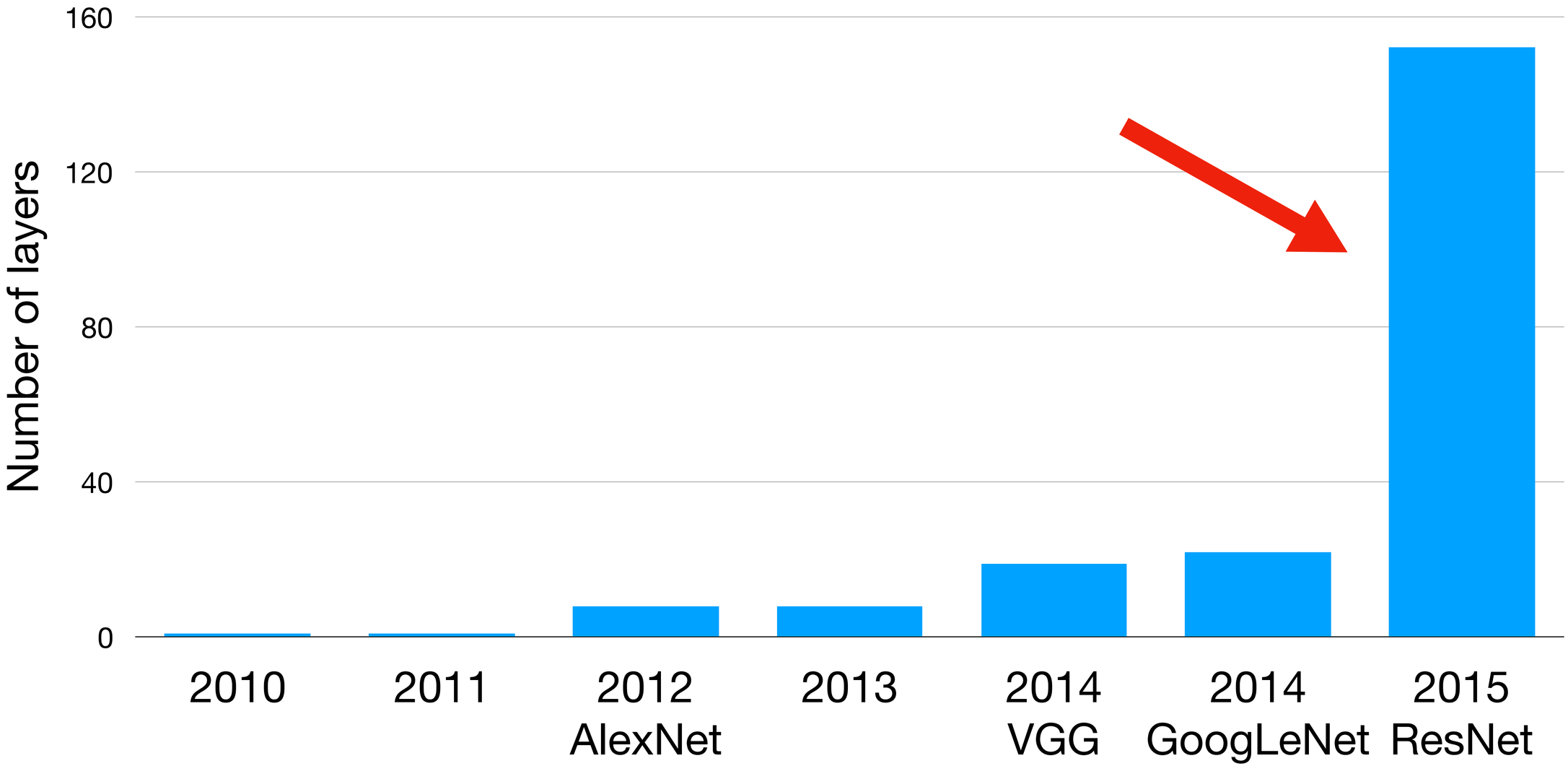
ResNets (He et. al., 2015)

Skip connections



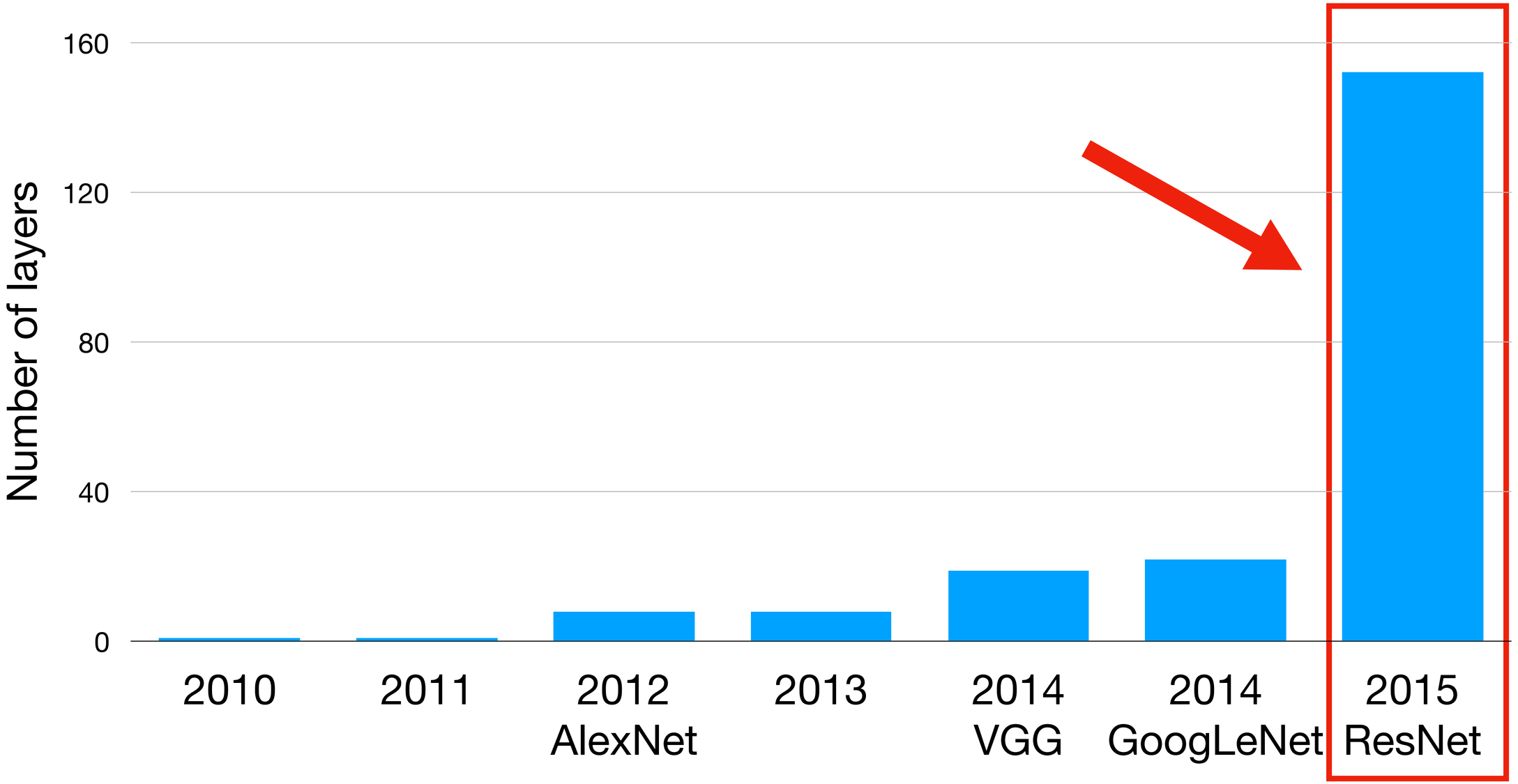
ResNets (He et. al., 2015)

Revolution of Depth: Top models on ImageNet



Data from Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. ‘Deep Residual Learning for Image Recognition’, CVPR 2016.

Revolution of Depth: Top models on ImageNet



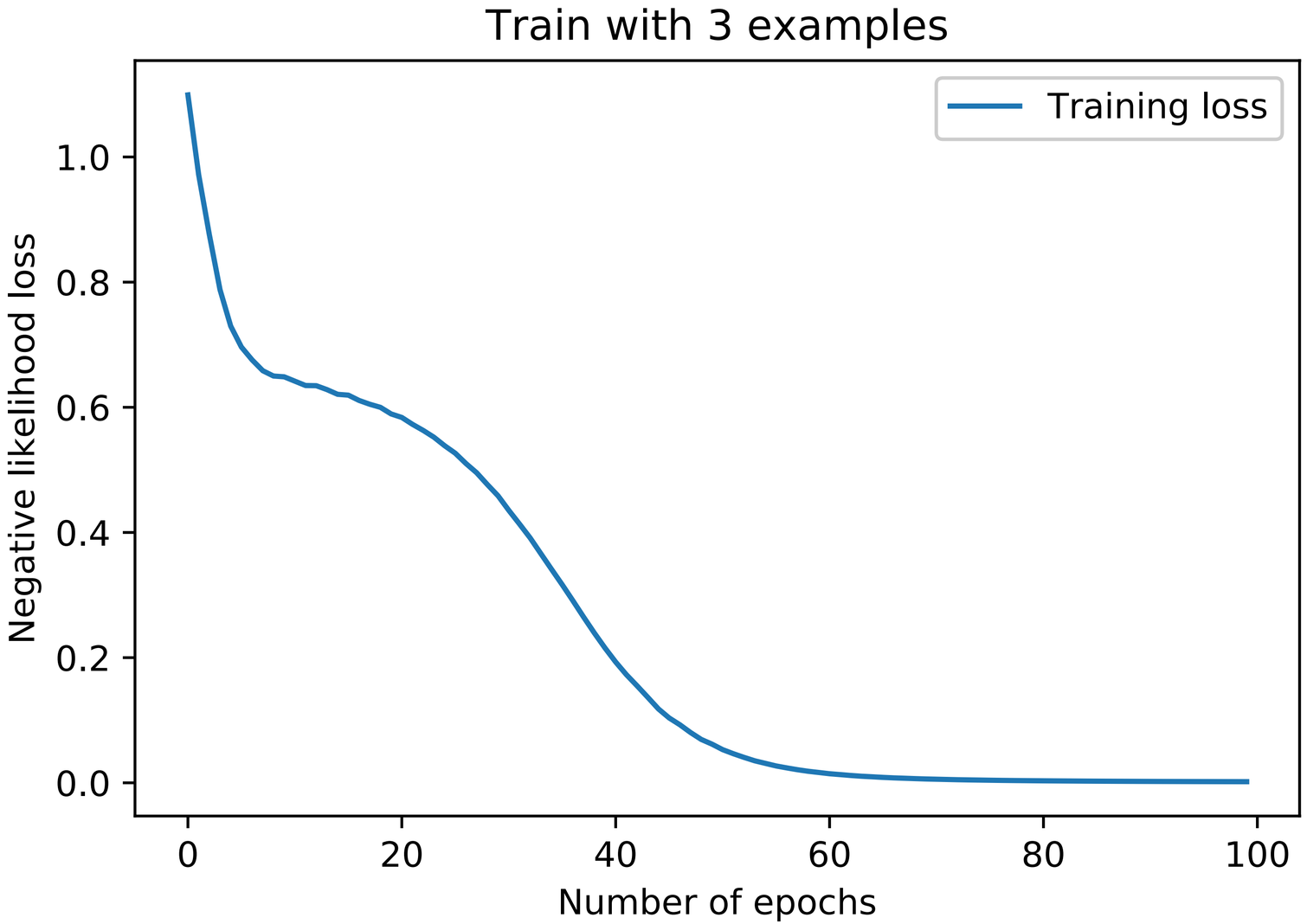
Data from Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. ‘Deep Residual Learning for Image Recognition’, CVPR 2016.

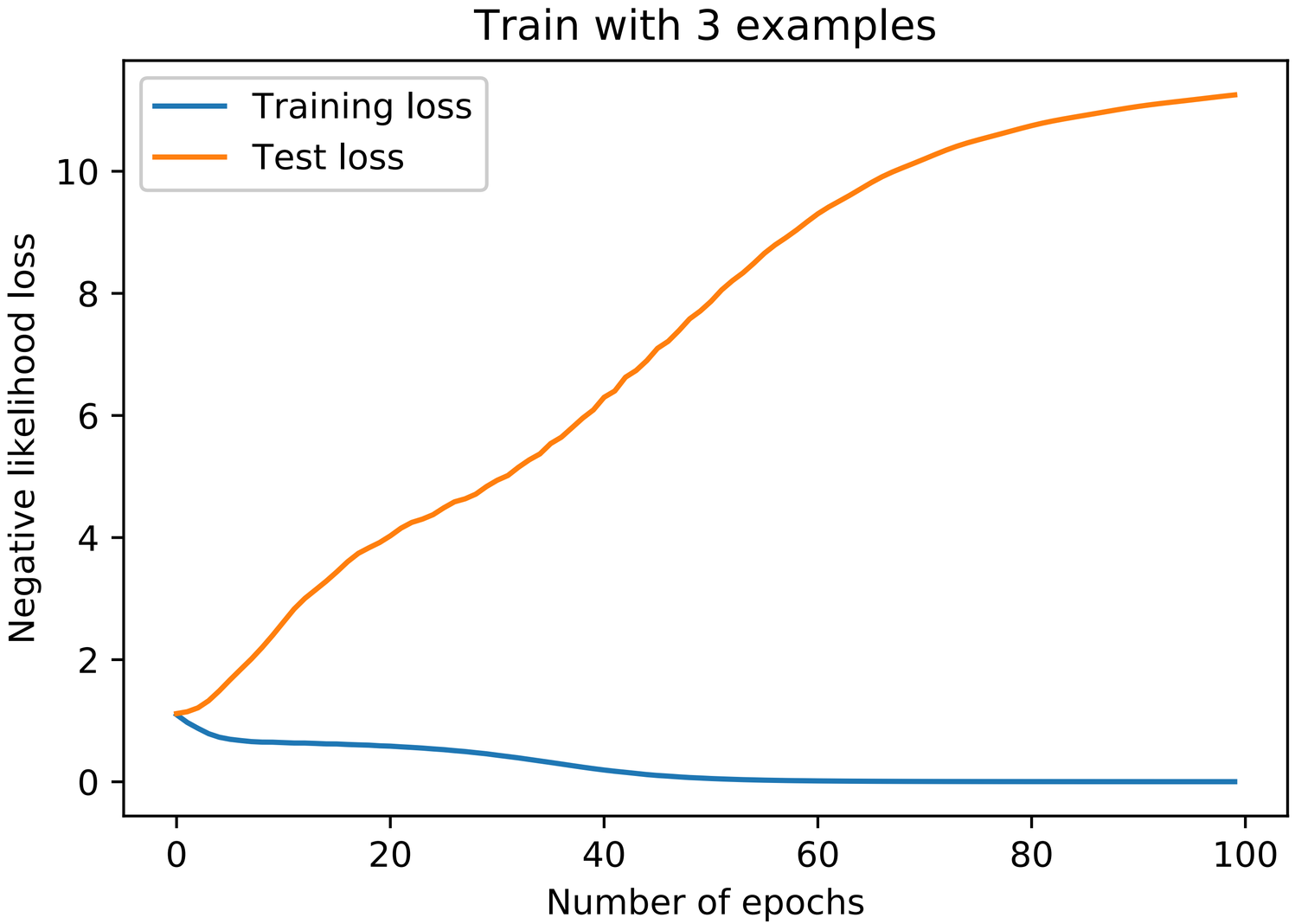
IV. Practical Tips

Overfit First

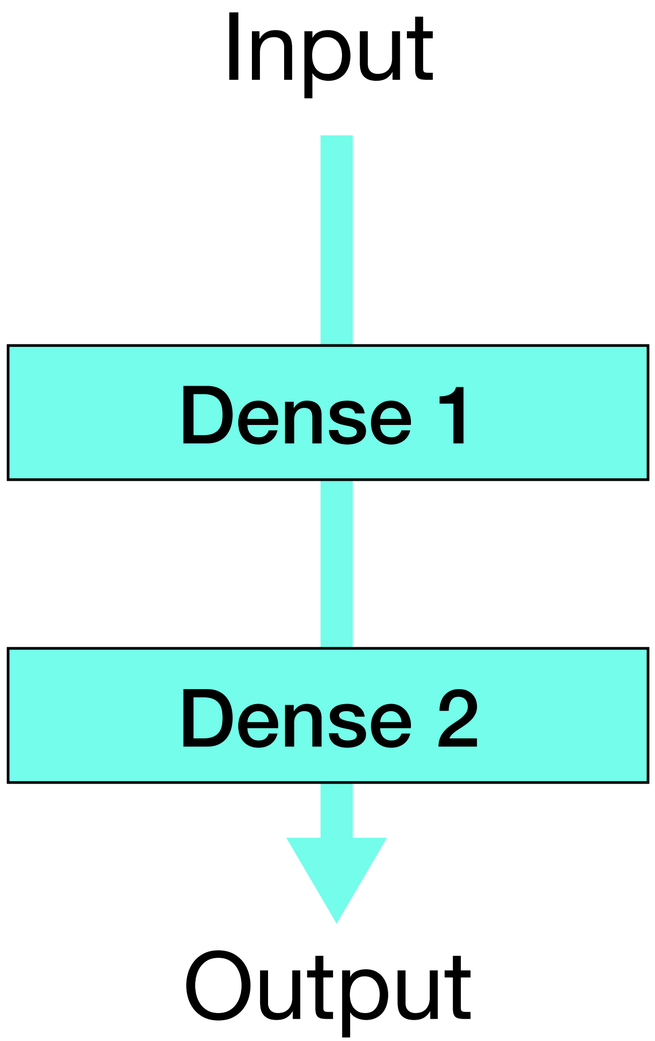
a few examples

Model

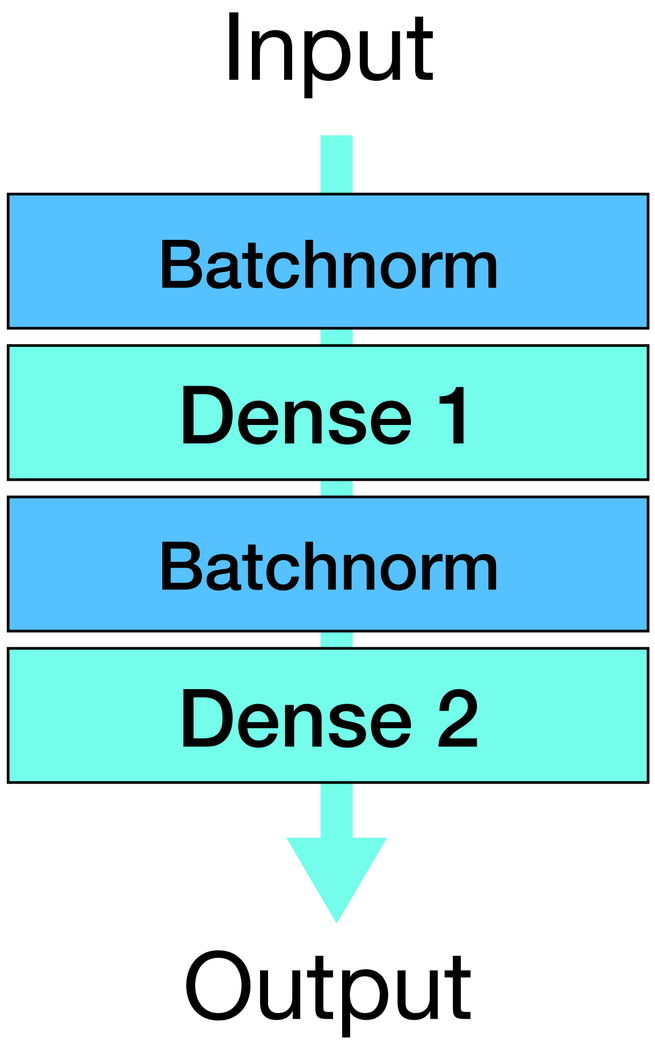

Train with 3 examples
Training loss
Negative likelihood loss
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
Number of epochs


Train with 3 examples
Training loss
Test loss
Negative likelihood loss
0
2
4
6
8
10
0
20
40
60
80
100
Number of epochs

Batchnorm



Ioffe and Szegedy, 2015



Ioffe and Szegedy, 2015

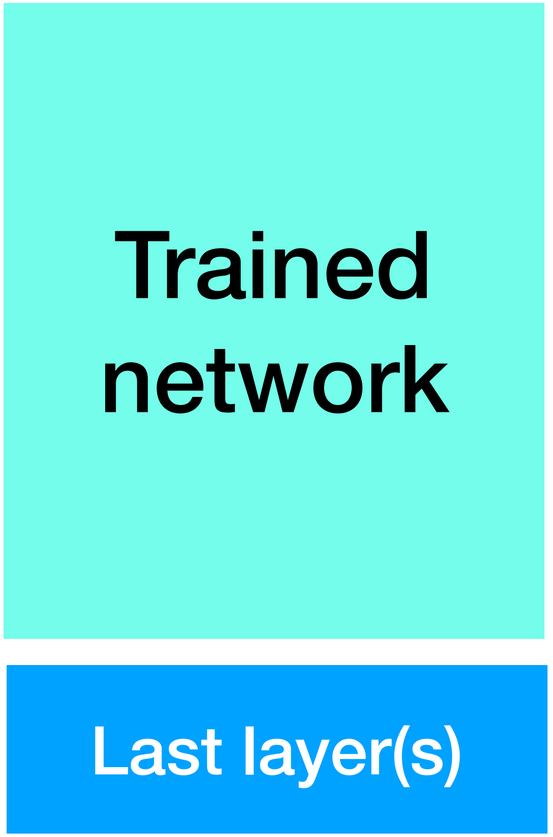
Transfer Learning

Low

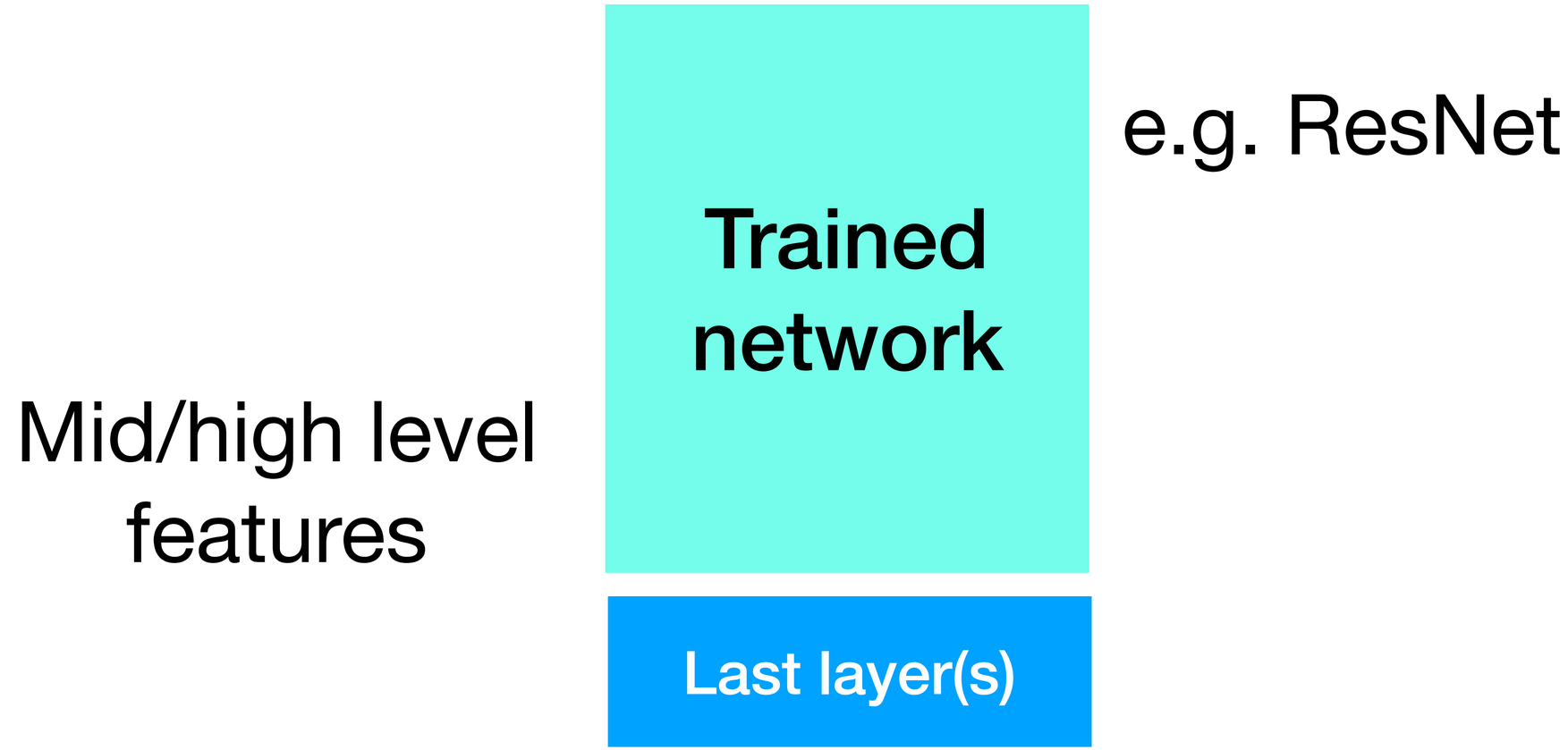
Mid

High

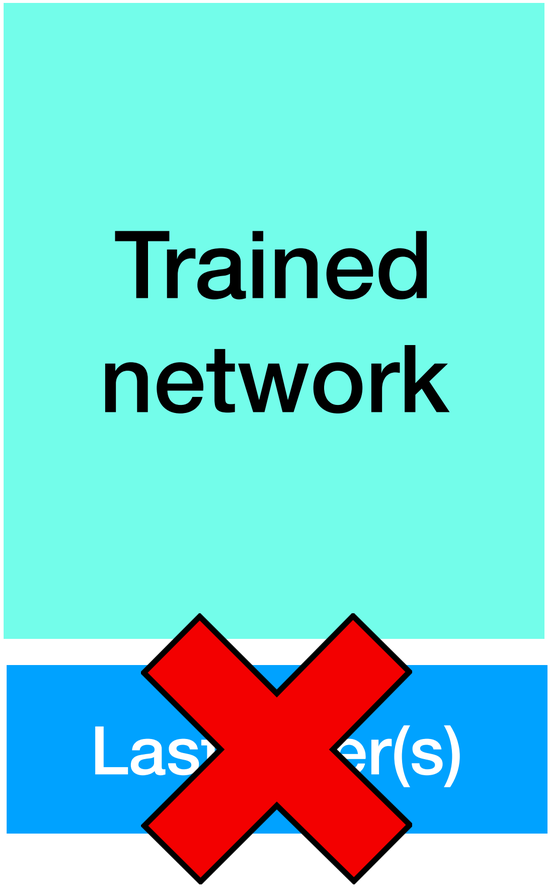
Based on Zeiler and Fergus, 2013



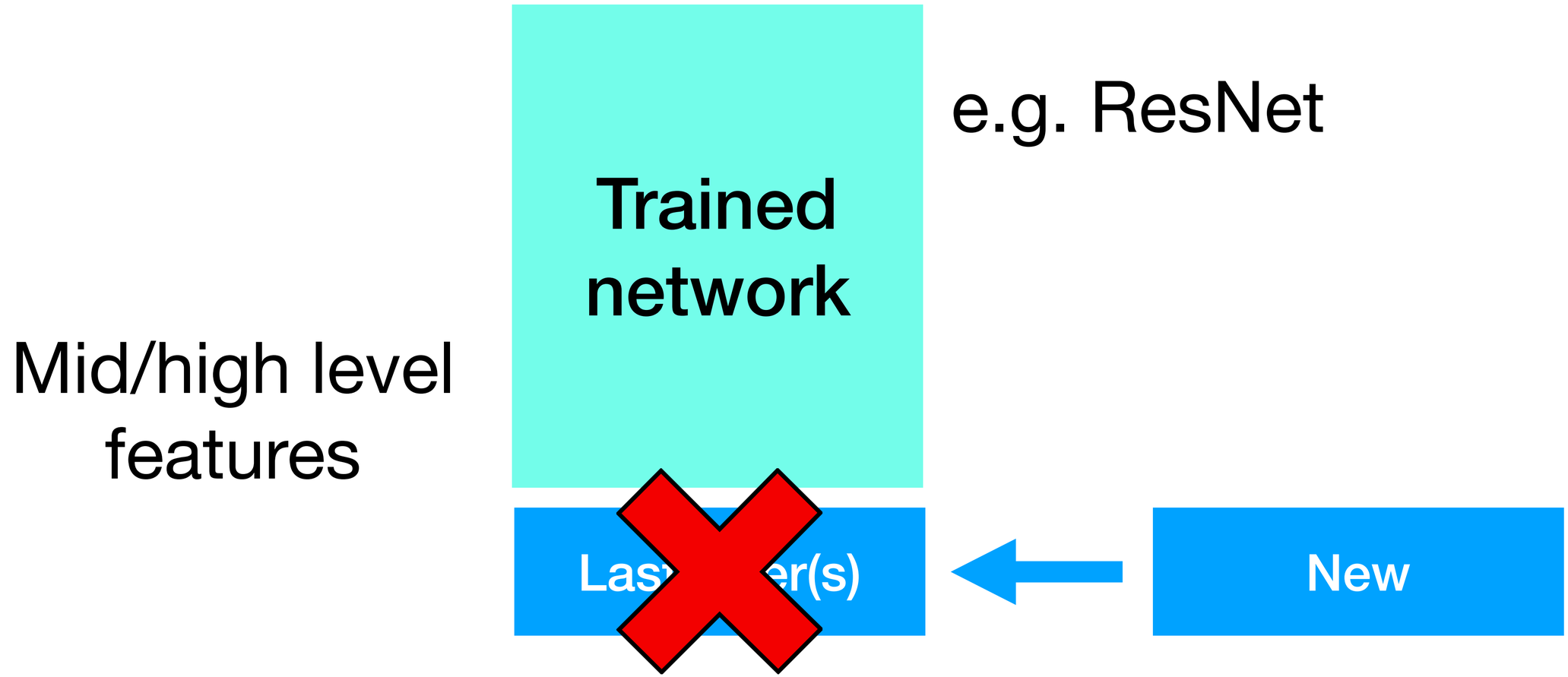
e.g. ResNet

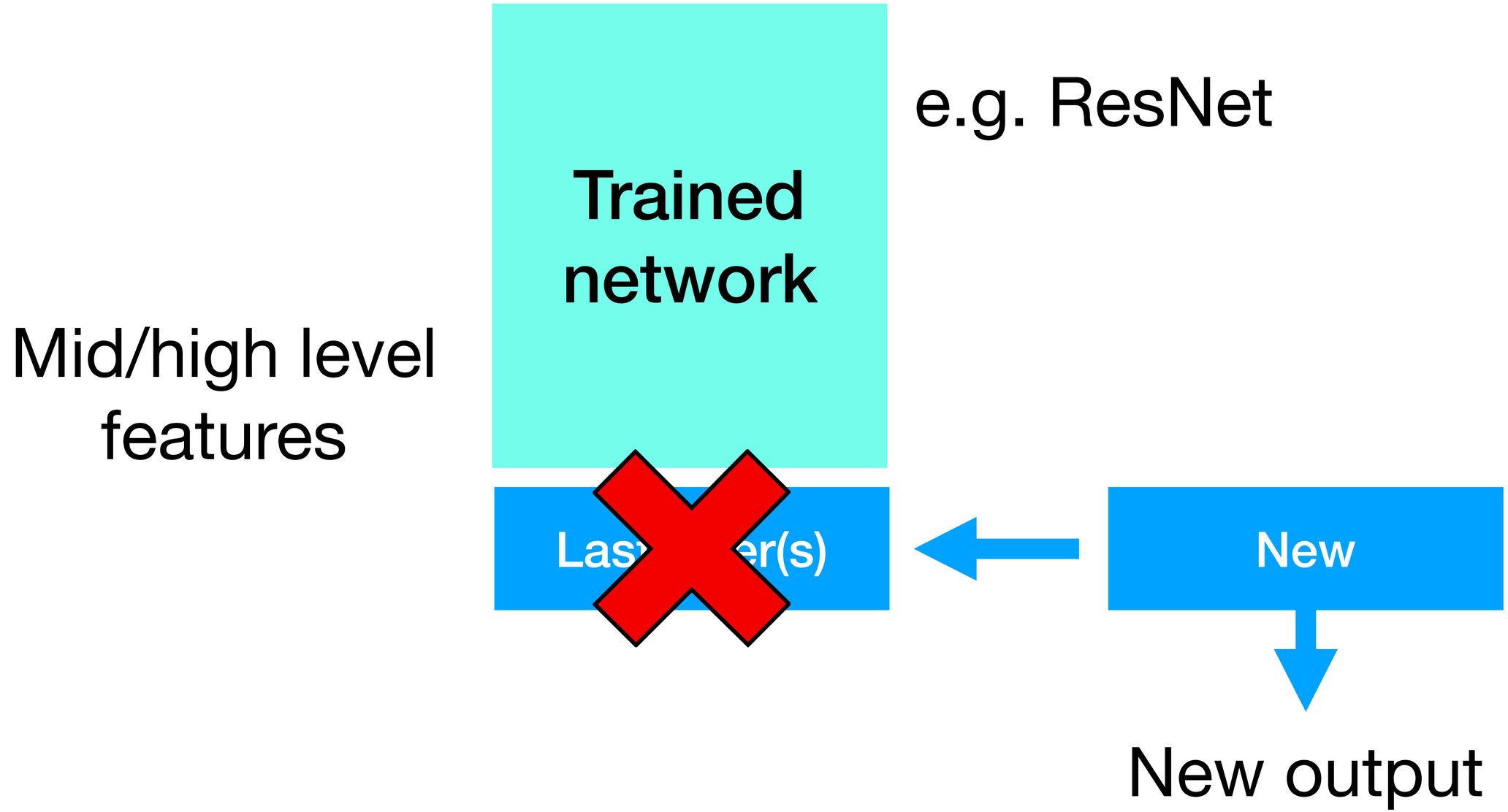

Trained network
e.g. ResNet
Mid/high level features
Last layer(s)

Mid/high level features



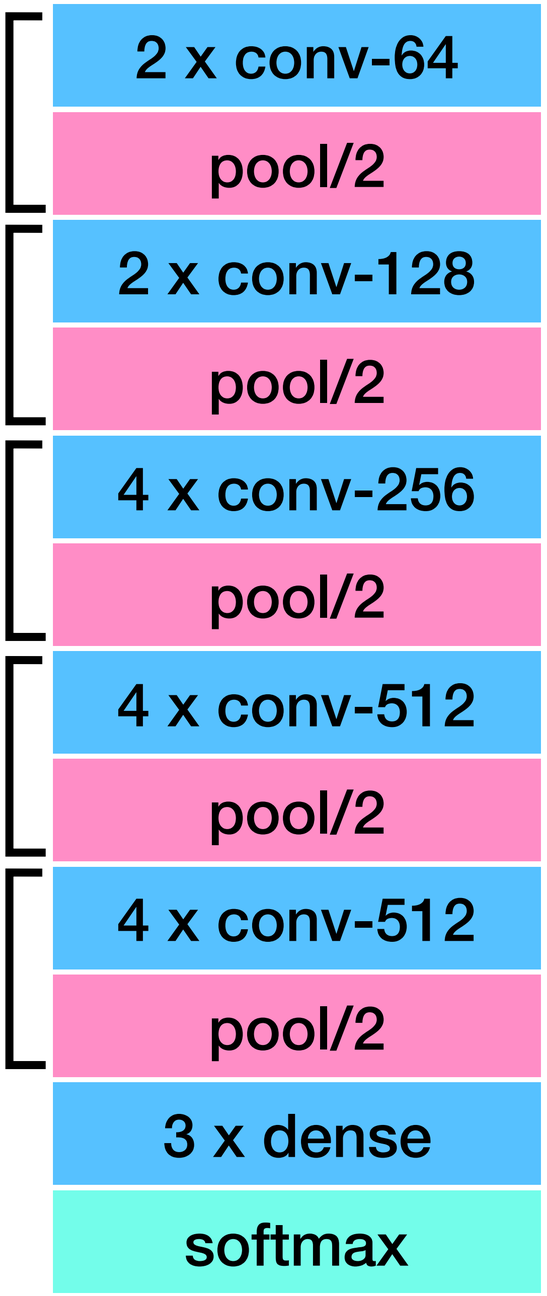
e.g. ResNet


Trained network
e.g. ResNet
Mid/high level features
Last er(s)
New

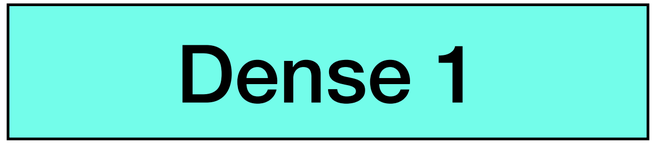

Trained network
e.g. ResNet
Mid/high level features
Last layer(s)
New
New output

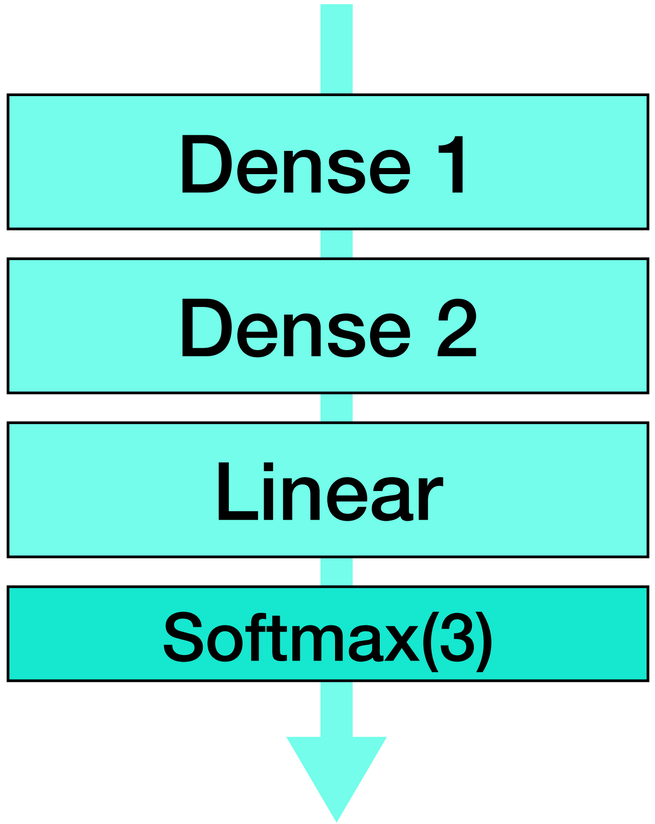
Simplify

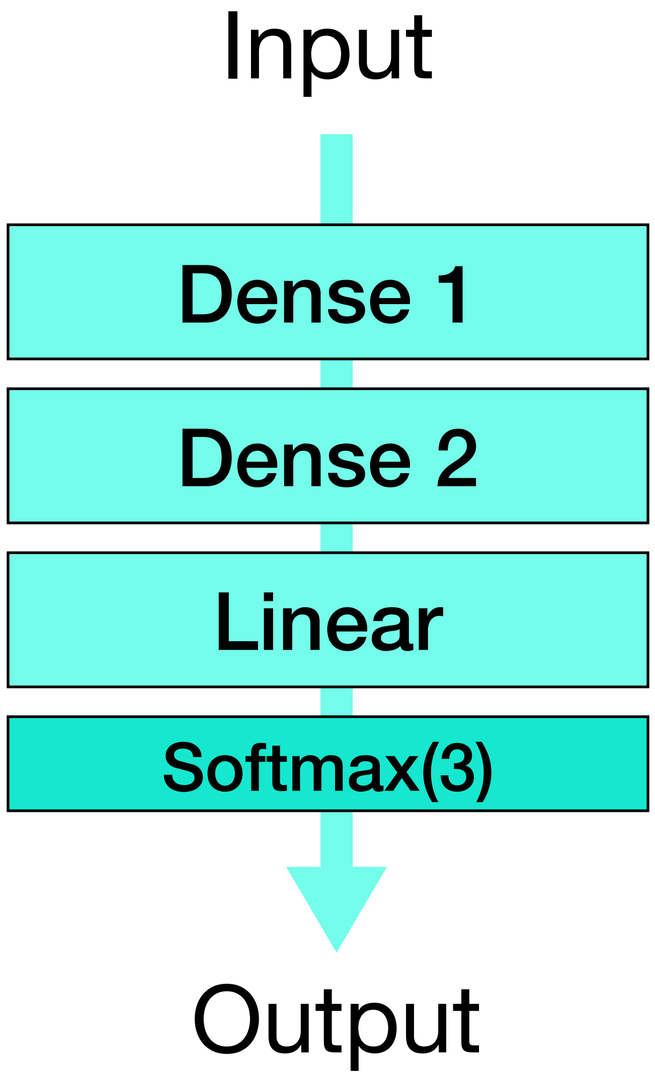
VGG-19

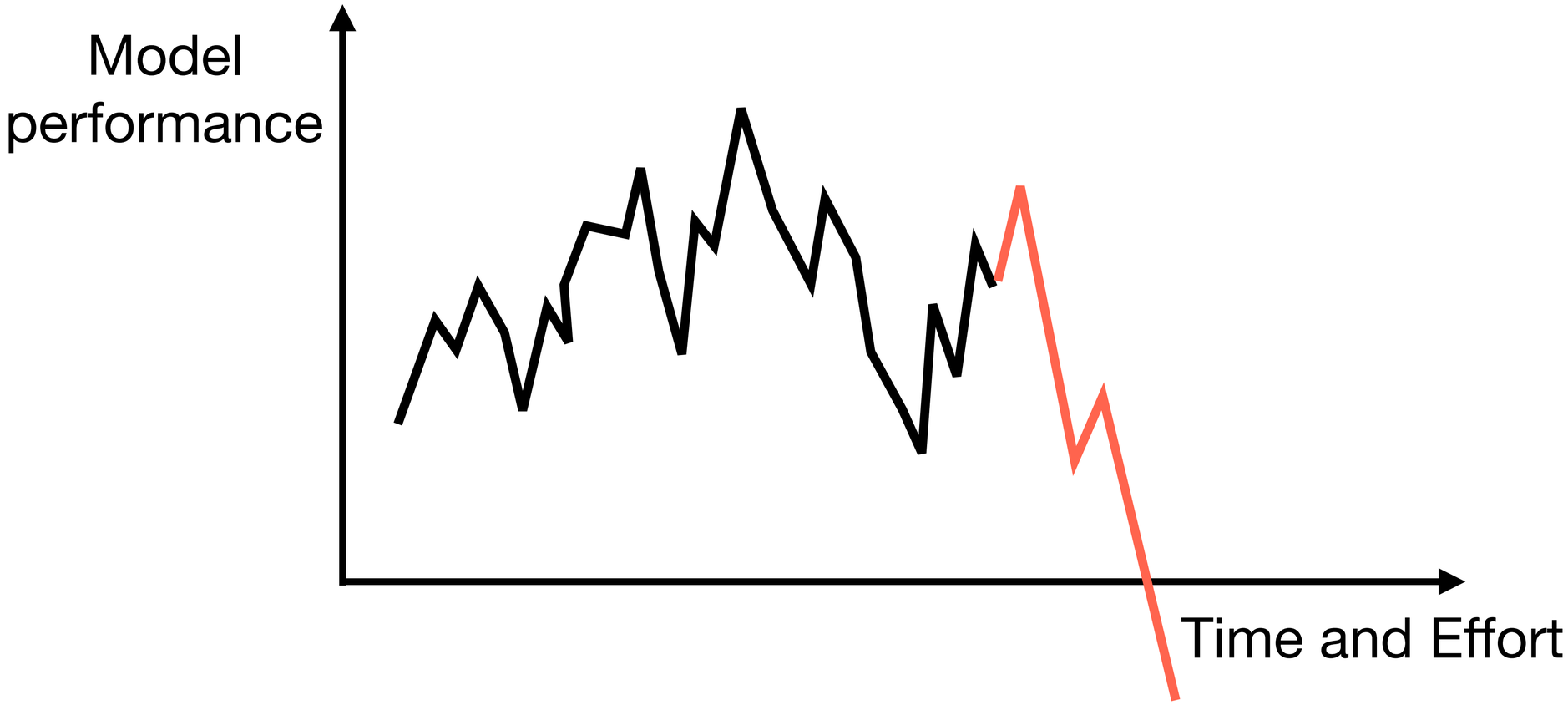


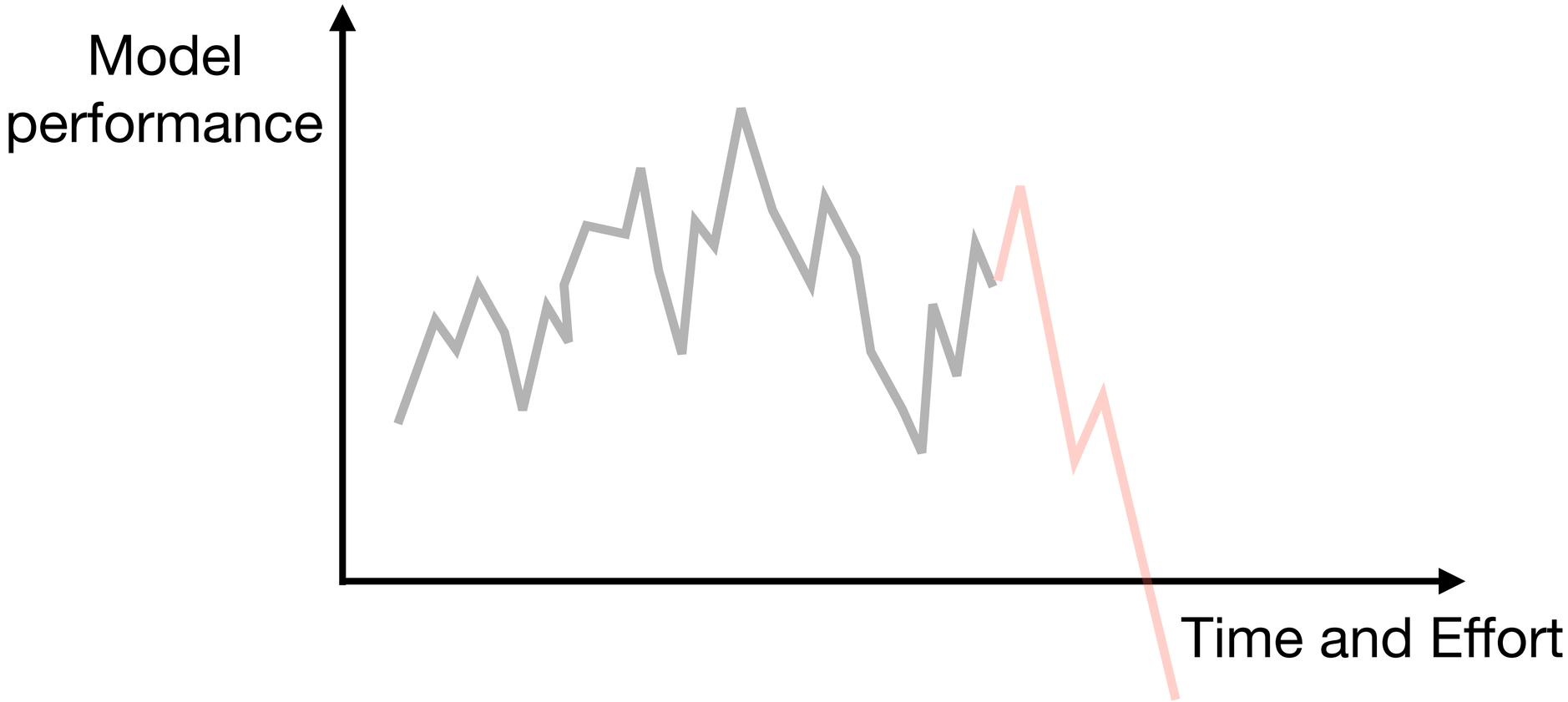
Simonyan and Zisserman, 2014

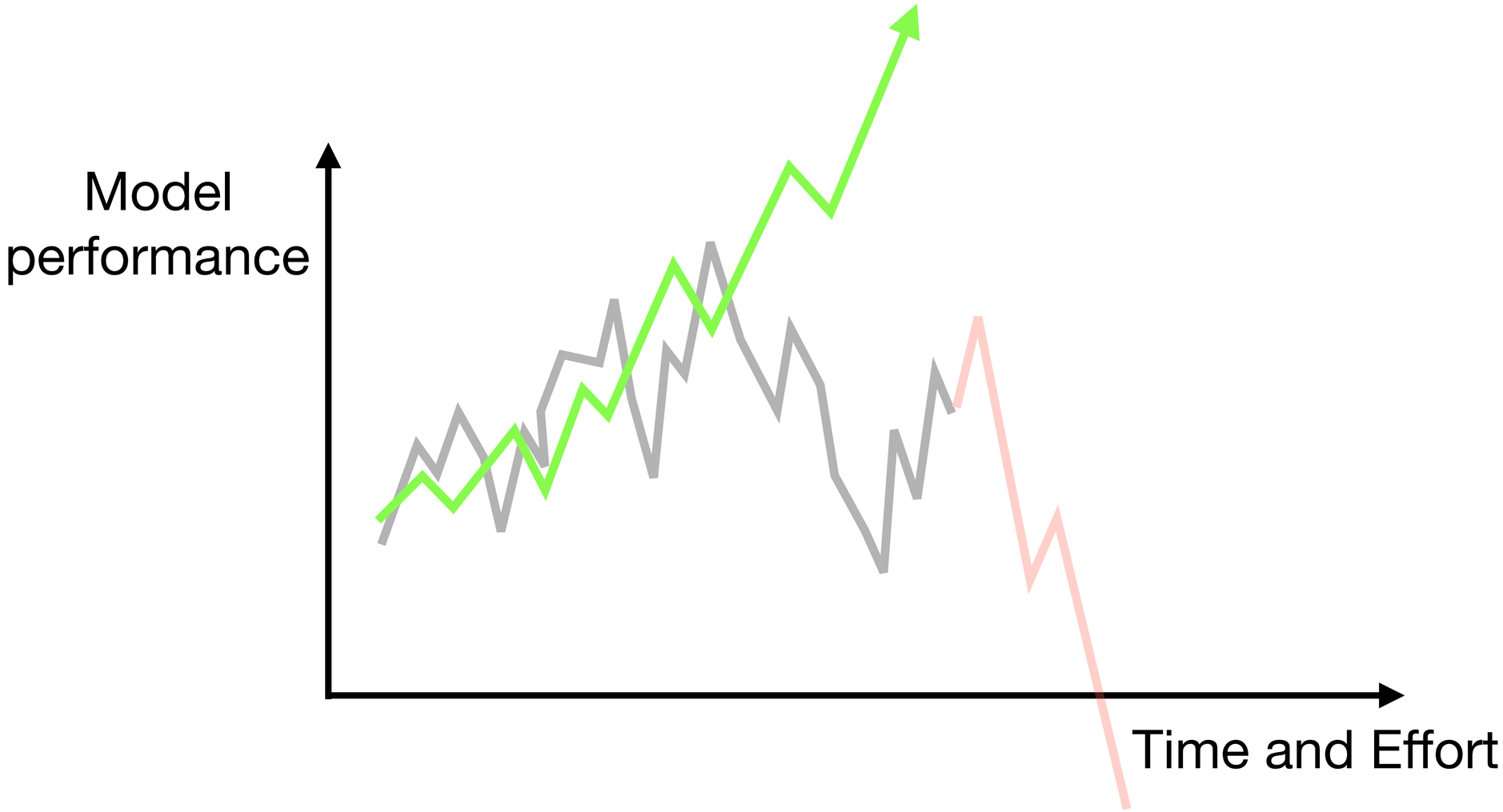

Dense 1


Dense 1
Dense 2
Linear
Softmax(3)


Input
Dense 1
Dense 2
Linear
Softmax(3)
Output


Model
performance
Time and Effort


Model
performance
Time and Effort


Model
performance
Time and Effort

Thank You!

Appendix

Resources on Coding Neural Networks

- Jason Brownlee's Machine Learning Mastery (lots of code)
- My blog posts on training models in TensorFlow (training a model, MLP, CNN)

Resources on Deep Learning

- THE Deep Learning book (Ian Goodfellow, Yoshua Bengio, Aaron Courville)
- Stanford's course on Convolutional Neural Networks
- fast.ai courses on neural networks (haven't tried much myself but I've heard they are good)