# Hypertuning

## Search spaces & hypertune algorithms

## deep learning 4

Raoul Grouls, 19 Mei 2025

# Recap

# Neural networks

$$\sigma(wx + b)$$

# Neural networks

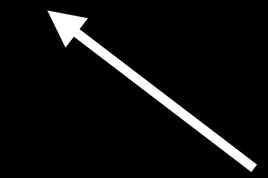
$$\sigma(wx + b)$$

Non-linear transformation

Linear transformation

# Neural networks

$$\hat{y} = \sigma(wx + b)$$



Prediction

# Neural networks

$$\hat{y} = \sigma(wx + b)$$

$$z = (y - \hat{y})^2$$

Change $w$ and $b$ such that $z$ is minimal

# Neural networks

$$w \leftarrow w + \eta \frac{\partial z}{\partial w}$$

$$b \leftarrow b + \eta \frac{\partial z}{\partial b}$$

Update the weights

# Neural networks

$$\left.\begin{aligned} w &\leftarrow w + \eta \frac{\partial z}{\partial w} \\ b &\leftarrow b + \eta \frac{\partial z}{\partial b} \end{aligned}\right\} \text{Optimizer}$$

# Convolutions

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 1 | -1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | -1 | 0 | 0 | 0 |
| 0 | 1 | -1 | 0 | 0 |

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

| | | |
|---|---|---|
| -1 | | |
| | | |
| | | |

# Convolutions

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 1 | -1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | -1 | 0 | 0 | 0 |
| 0 | 1 | -1 | 0 | 0 |

| | | |
|---|---|---|
| $w$ | $w$ | $w$ |
| $w$ | $w$ | $w$ |
| $w$ | $w$ | $w$ |

Learnable

| | | |
|---|---|---|
| $\hat{y}$ | $\hat{y}$ | $\hat{y}$ |
| $\hat{y}$ | $\hat{y}$ | $\hat{y}$ |
| $\hat{y}$ | $\hat{y}$ | $\hat{y}$ |

# Convolutions

$$\sigma(w * x)$$

Non-linear transformation

Linear transformation

# Concatenation

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 1 | -1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | -1 | 0 | 0 | 0 |
| 0 | 1 | -1 | 0 | 0 |

$h_{t-1}$

| | | | | |
|---|---|---|---|---|
| 1 | 1 | -1 | 0 | 1 |

$x_t$

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 1 | -1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | -1 | 0 | 0 | 0 |
| 0 | 1 | -1 | 0 | 0 |
| 1 | 1 | -1 | 0 | 1 |

$[x_t, h_{t-1}]$

# RNN

$$\sigma(w[x_t, h_{t-1}] + b)$$

# Gates

| | | | |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| 1 | -1 | 1 | 0 |
| 1 | 1 | 0 | -1 |
| 0 | -1 | 0 | 0 |

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |

| | | | |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| 1 | -1 | 1 | 0 |
| 1 | 1 | 0 | -1 |
| 0 | -1 | 0 | 0 |

$$X \quad \otimes \quad \Gamma \quad = \quad Y$$

# Gates

| | | | |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| 1 | -1 | 1 | 0 |
| 1 | 1 | 0 | -1 |
| 0 | -1 | 0 | 0 |

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

$$X \otimes \Gamma = Y$$

# Gates

| | | | |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| 1 | -1 | 1 | 0 |
| 1 | 1 | 0 | -1 |
| 0 | -1 | 0 | 0 |

$X$

$\otimes$

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |
| 0.5 | 0.5 | 0.5 | 0.5 |
| 0 | 0 | 0 | 0 |

$\Gamma$

$=$

| | | | |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| 1 | -1 | 1 | 0 |
| 0.5 | 0.5 | 0 | -0.5 |
| 0 | 0 | 0 | 0 |

$Y$

# GRU

$$\Gamma = \sigma(W_{\Gamma}[X_t, h_{t-1}] + b_{\Gamma})$$

$$\tilde{h}_t = \sigma(W_H[X_t, h_{t-1}] + b_H)$$

$$h_t = \Gamma \otimes h_{t-1} + (1 - \Gamma) \otimes \tilde{h}_t$$

# GRU

Sigmoid

Tanh

$$\Gamma = \sigma(W_{\Gamma}[X_t, h_{t-1}] + b_{\Gamma})$$

$$\tilde{h}_t = \sigma(W_H[X_t, h_{t-1}] + b_H)$$

$$h_t = \Gamma \otimes h_{t-1} + (1 - \Gamma) \otimes \tilde{h}_t$$

$$\sigma(wx + b)$$

$$\sigma(w * x)$$

$$\sigma(w[x_t, h_{t-1}] + b)$$

# Hyperparameters

# Neural networks

$$\sigma(f(x))$$

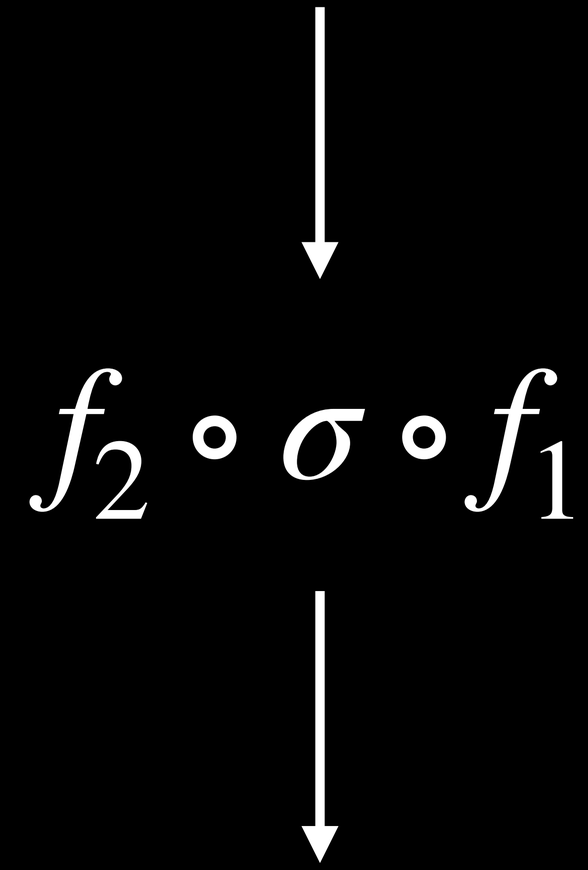
# Neural networks

$$f_2(\sigma(f_1(x)))$$

# Neural networks

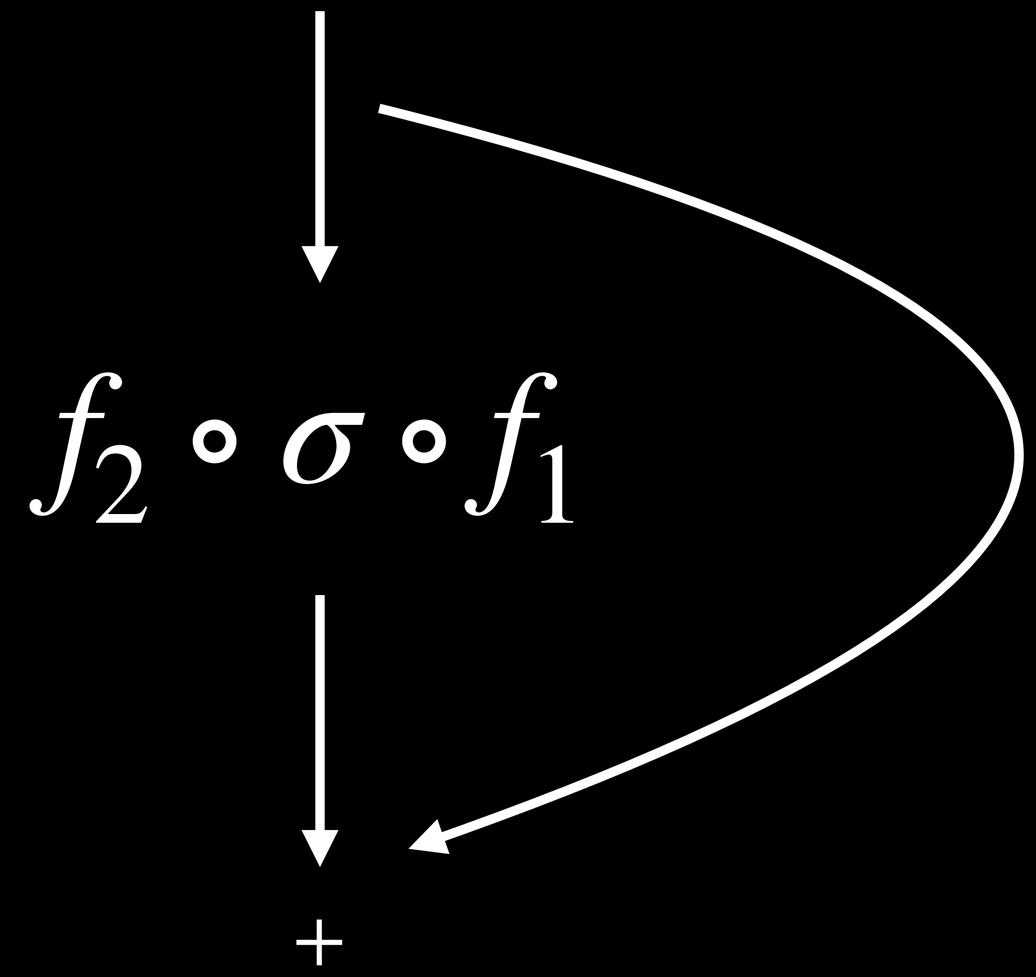
$$f_2 \circ \sigma \circ f_1$$

# Deep neural networks

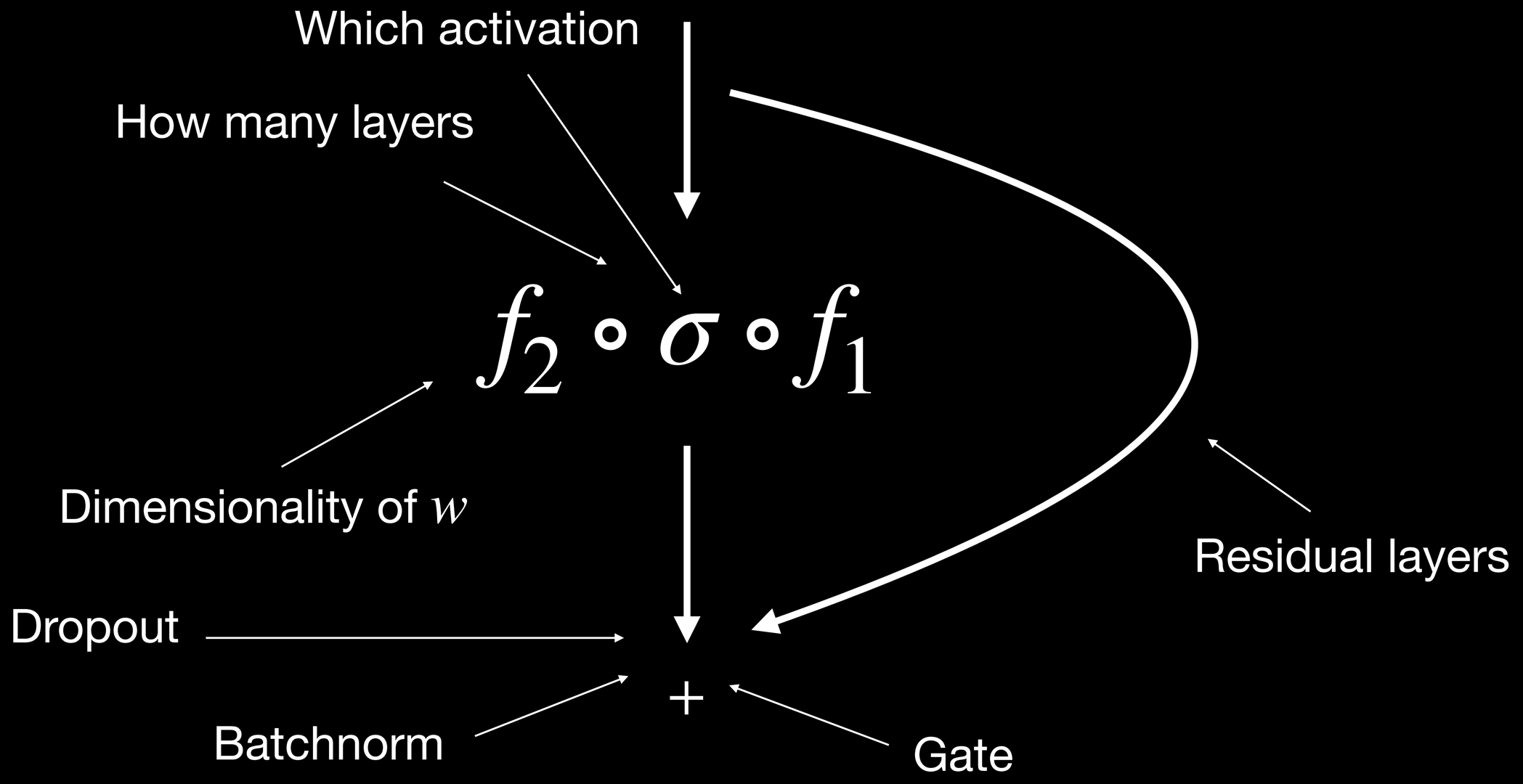


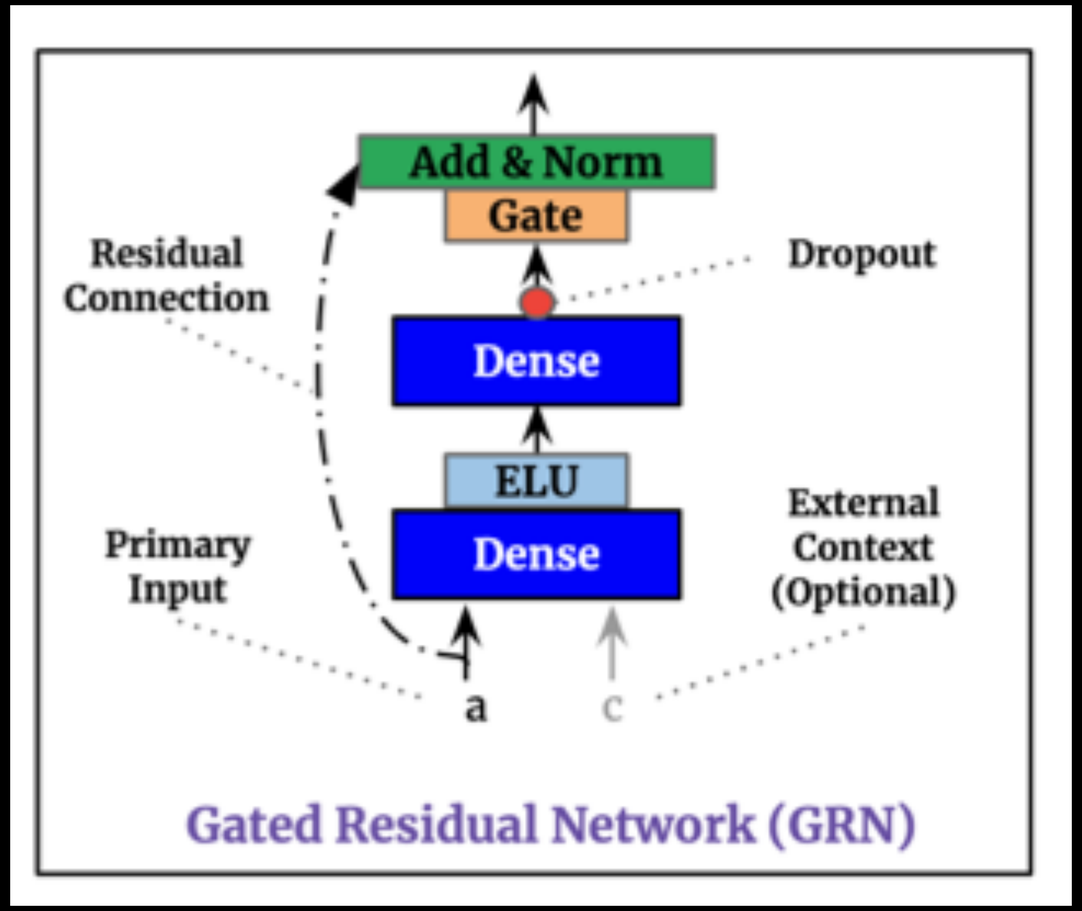
$$f_2 \circ \sigma \circ f_1$$

# Deep neural networks



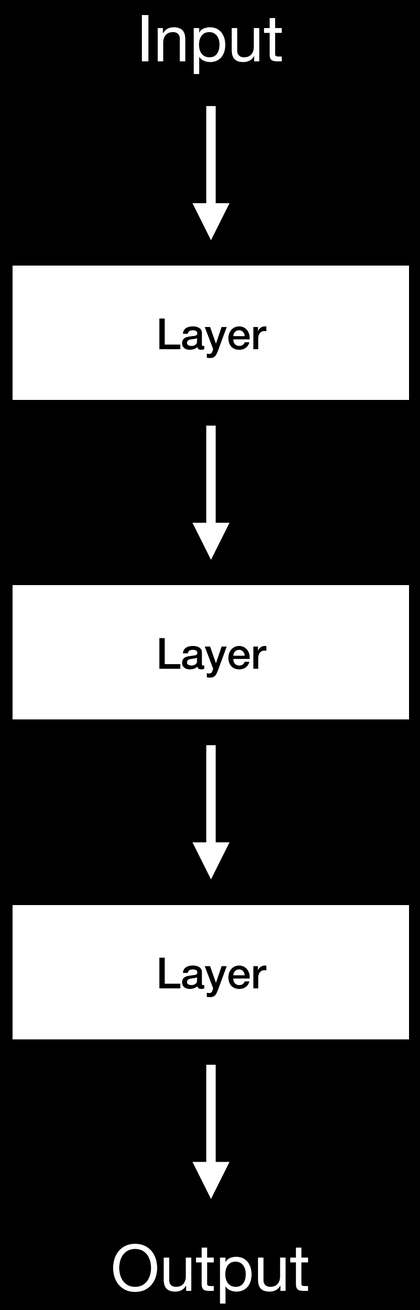
# Hyperparameters



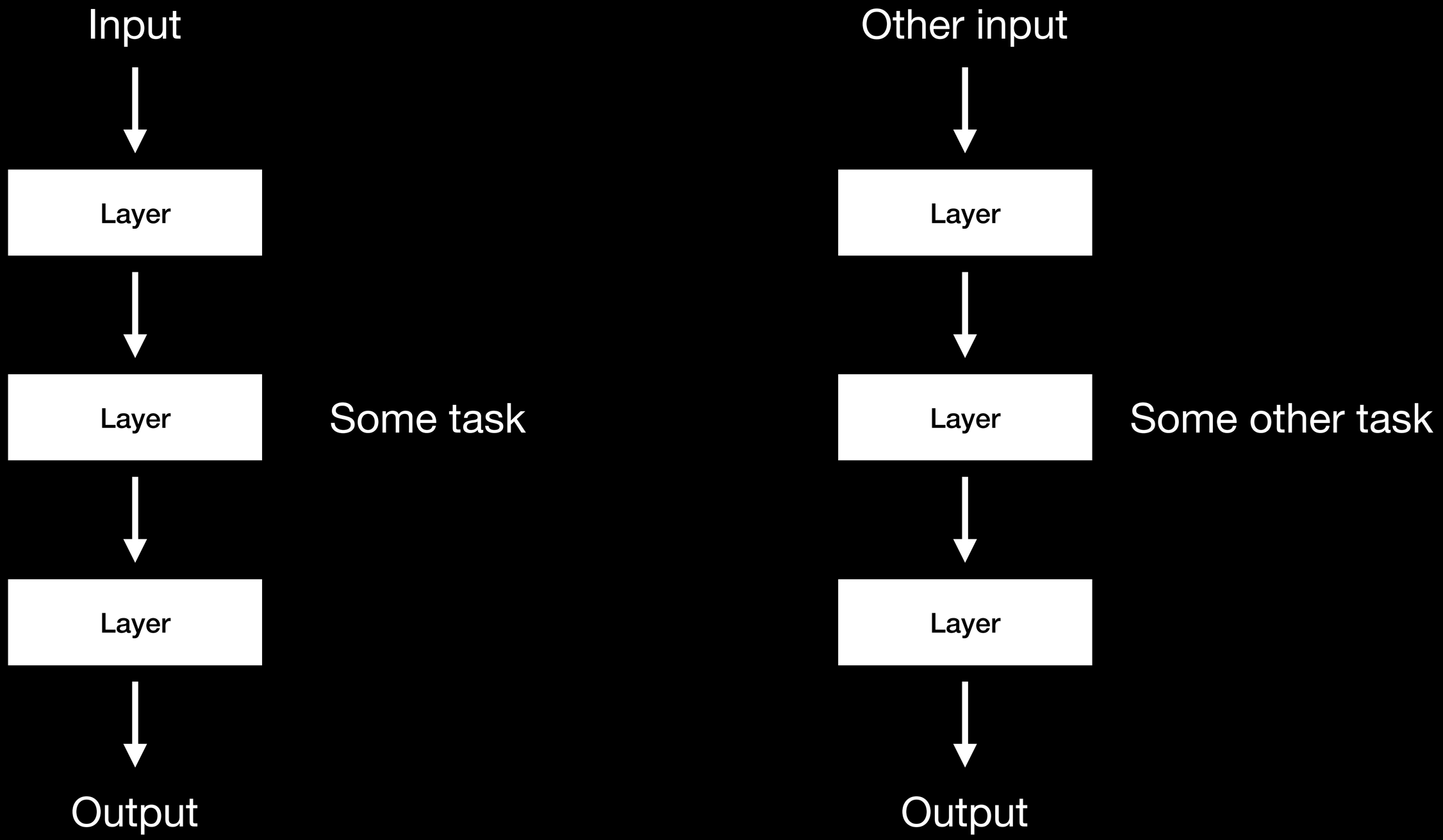


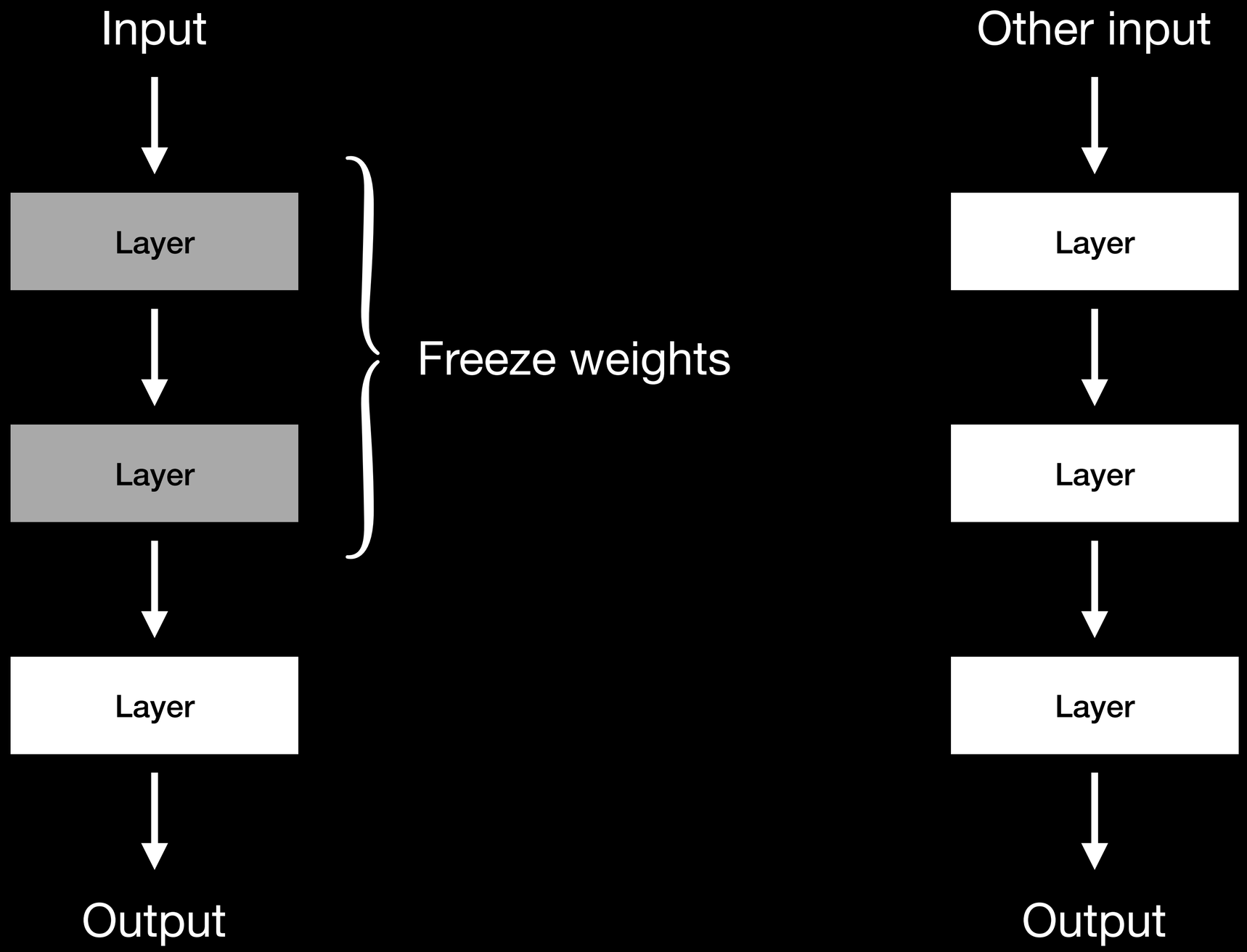

Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting, Lim B., 2019, https://arxiv.org/abs/1912.09363

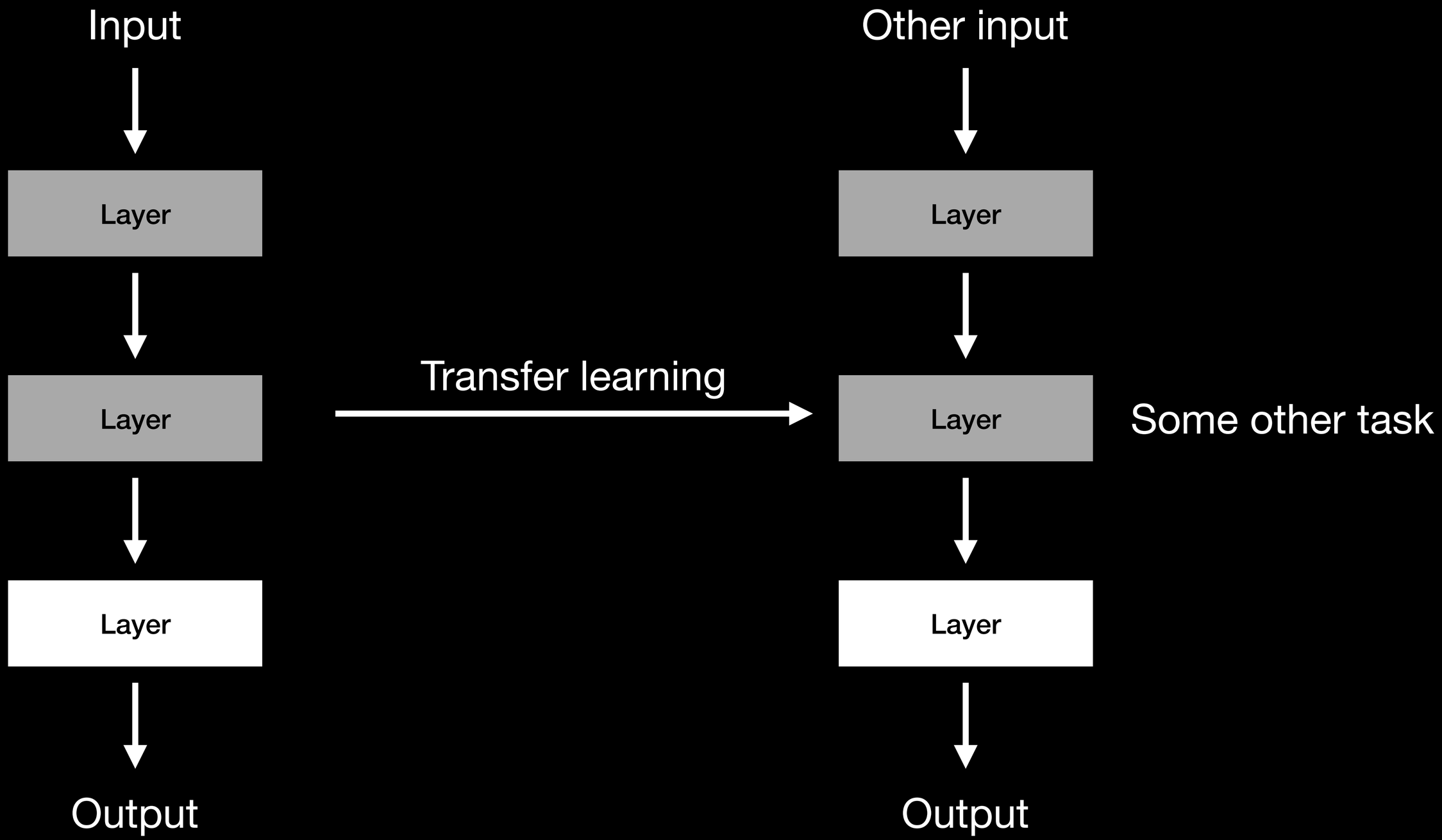
# Transfer learning

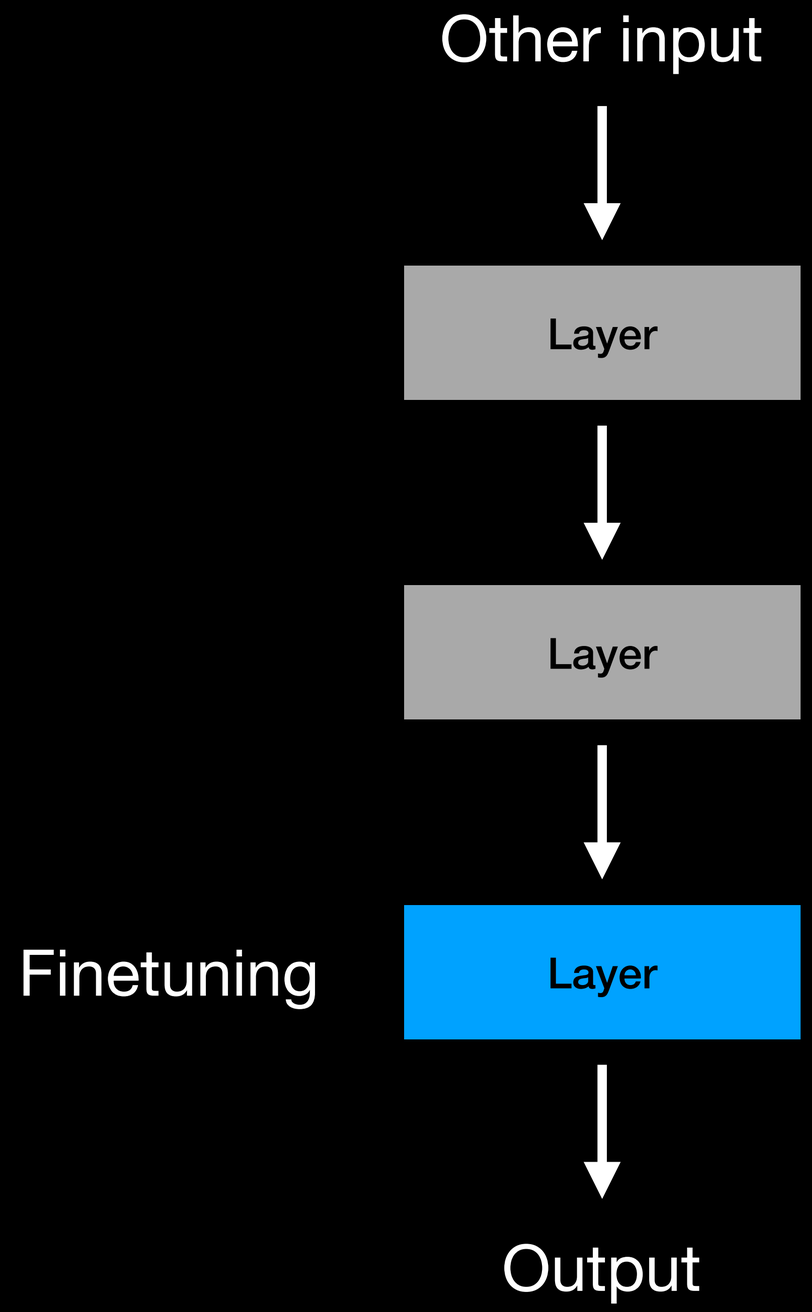



Some task

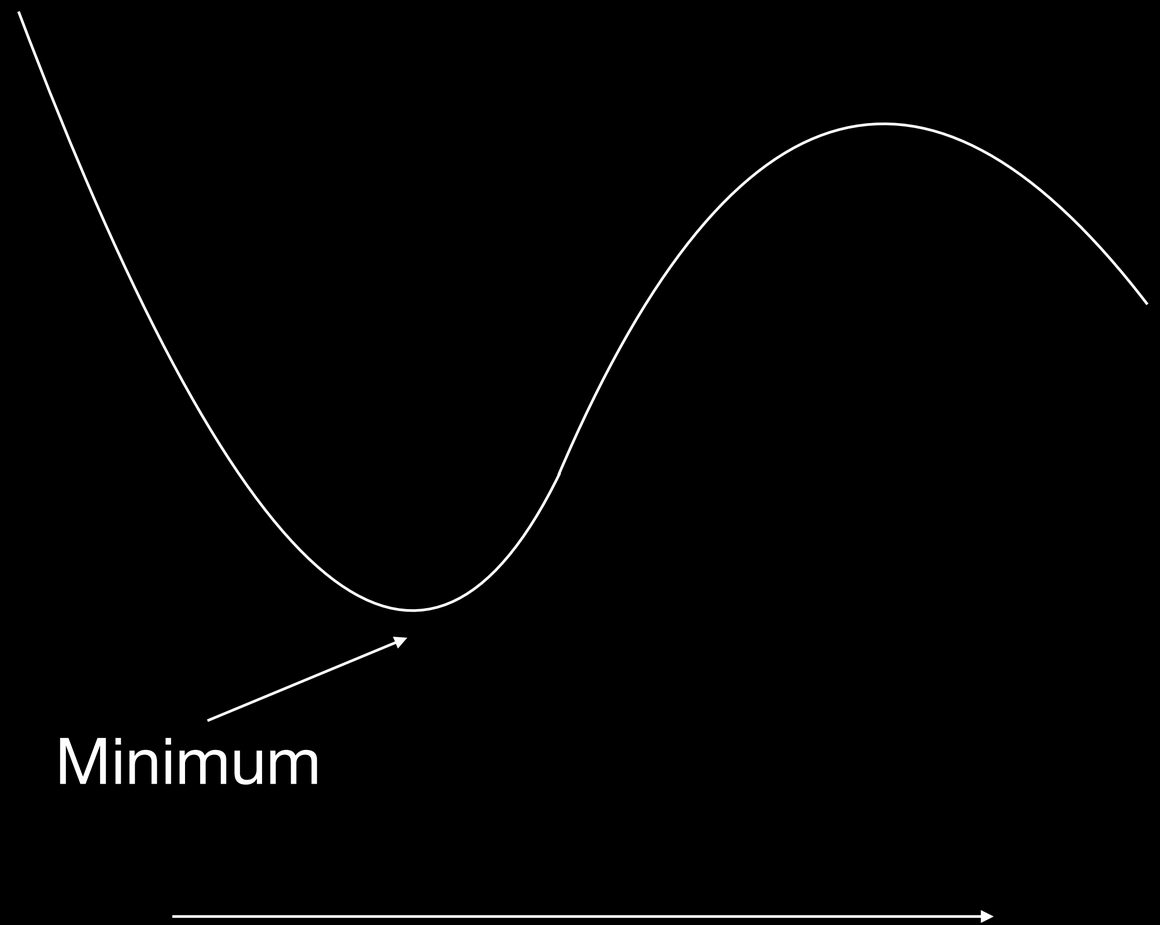

Input
Layer
Layer
Layer
Output
Some task
Other input
Layer
Layer
Layer
Output
Some other task

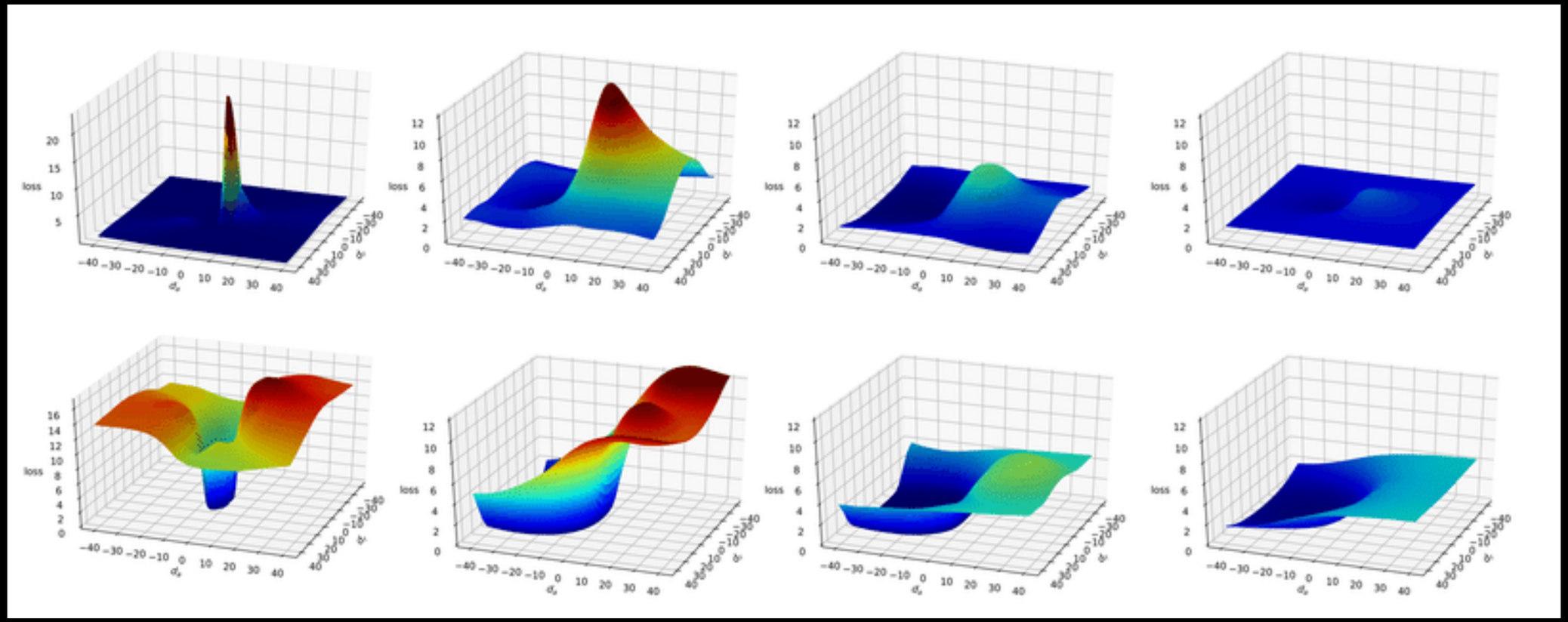

Input
Layer
Layer
Layer
Output
Freeze weights
Other input
Layer
Layer
Layer
Output


Input
Layer
Layer
Layer
Output
Transfer learning
Other input
Layer
Layer
Layer
Output
Some other task


Other input
Layer
Layer
Finetuning
Layer
Output

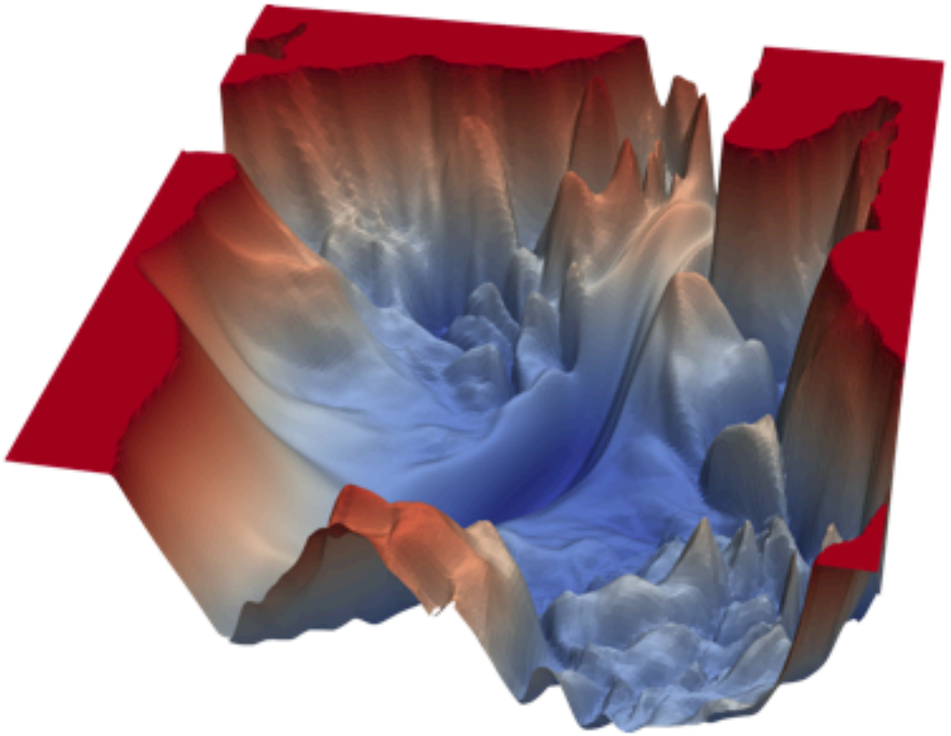
# Search spaces


Minimum
One hyperparameter

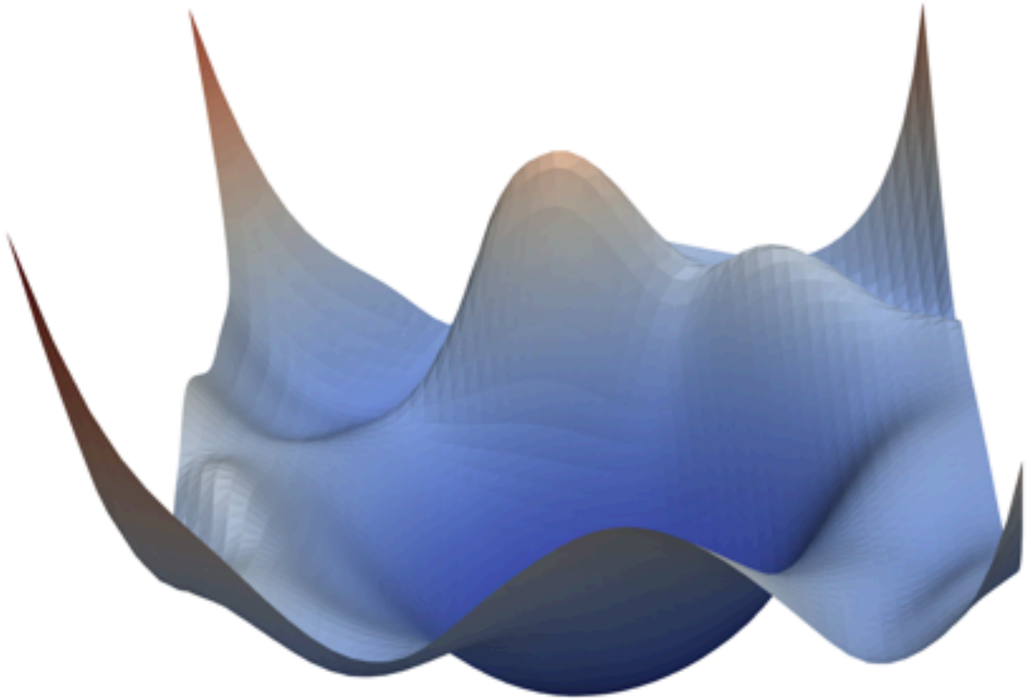



Two hyperparameters

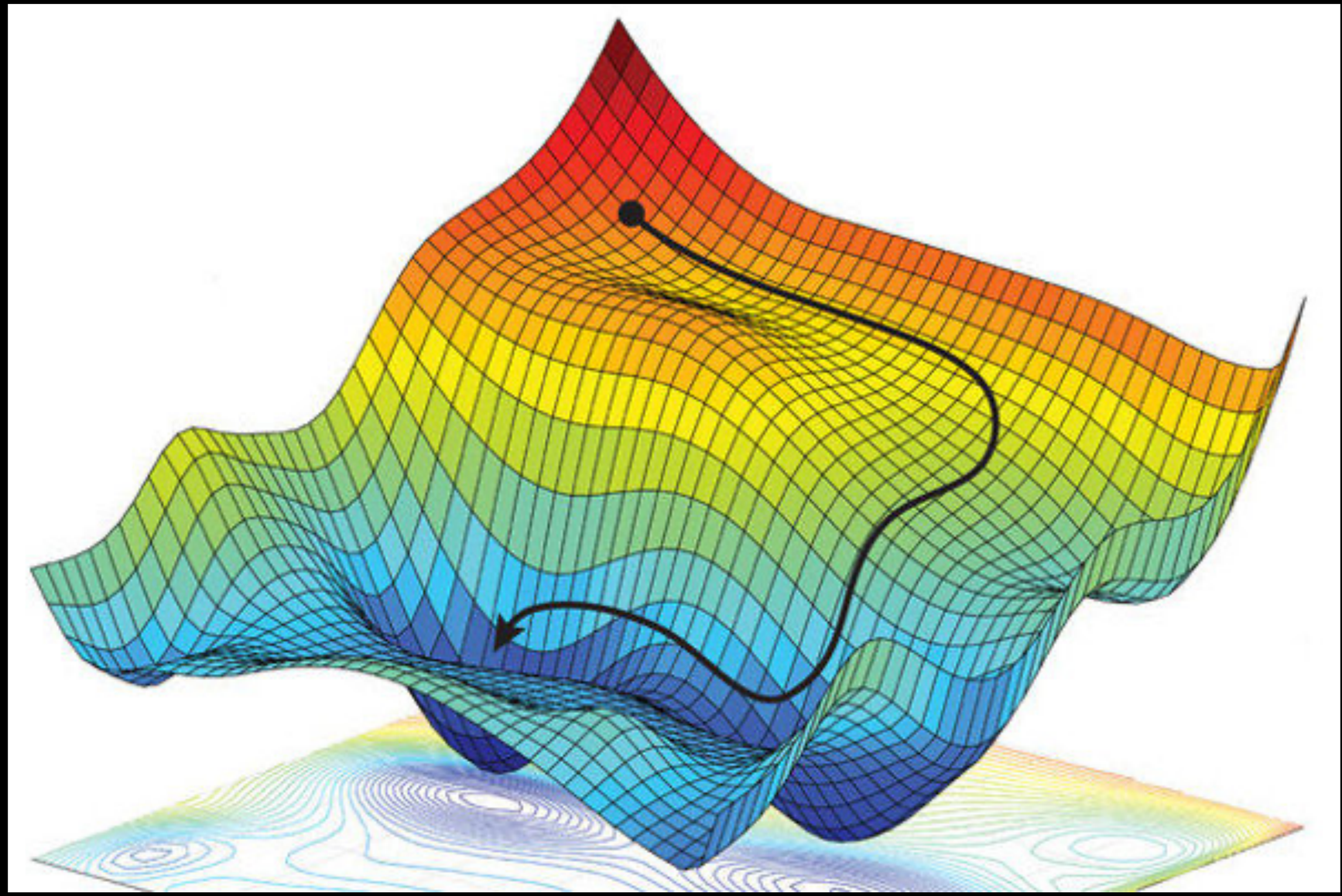
No residual connections



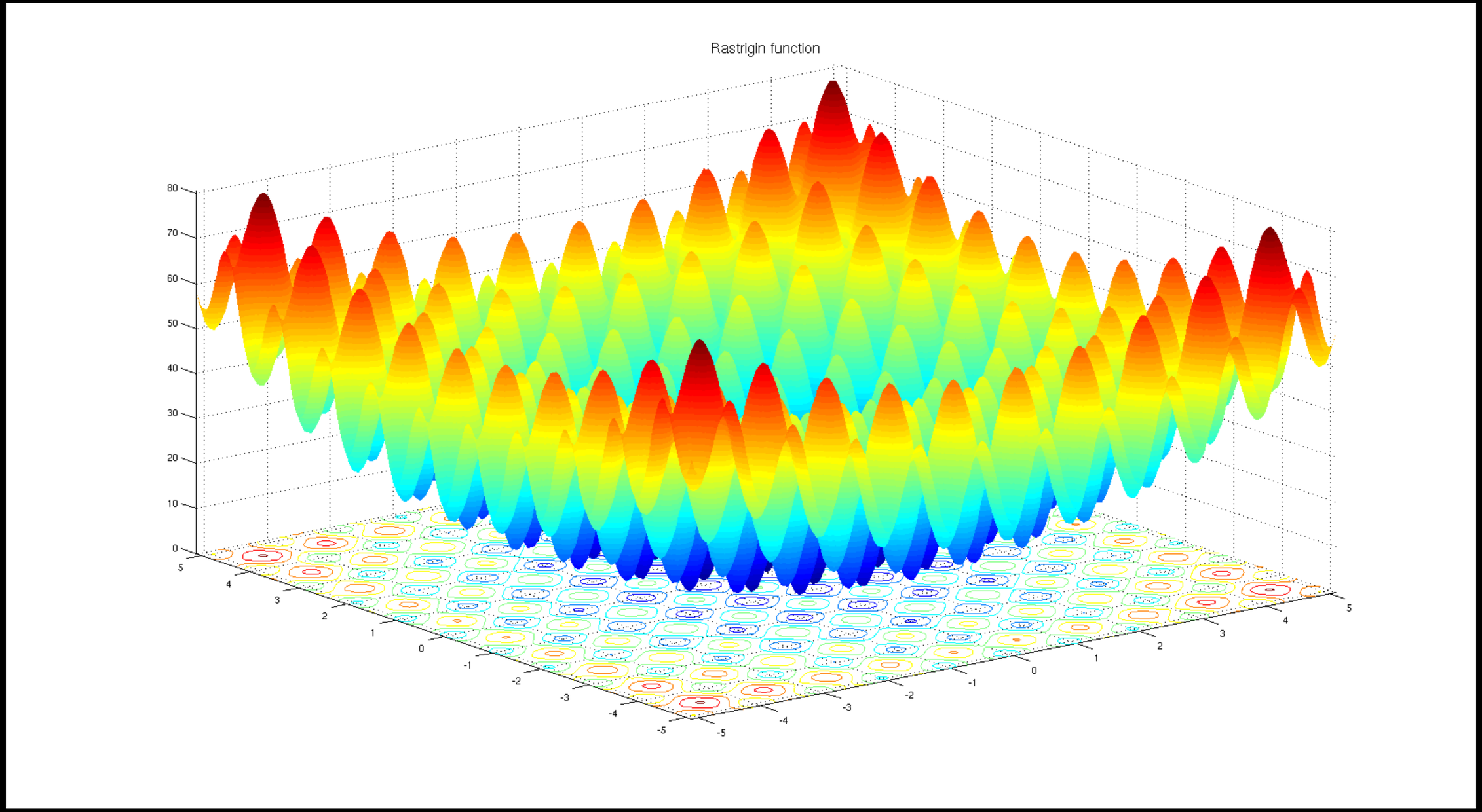
With residual connections



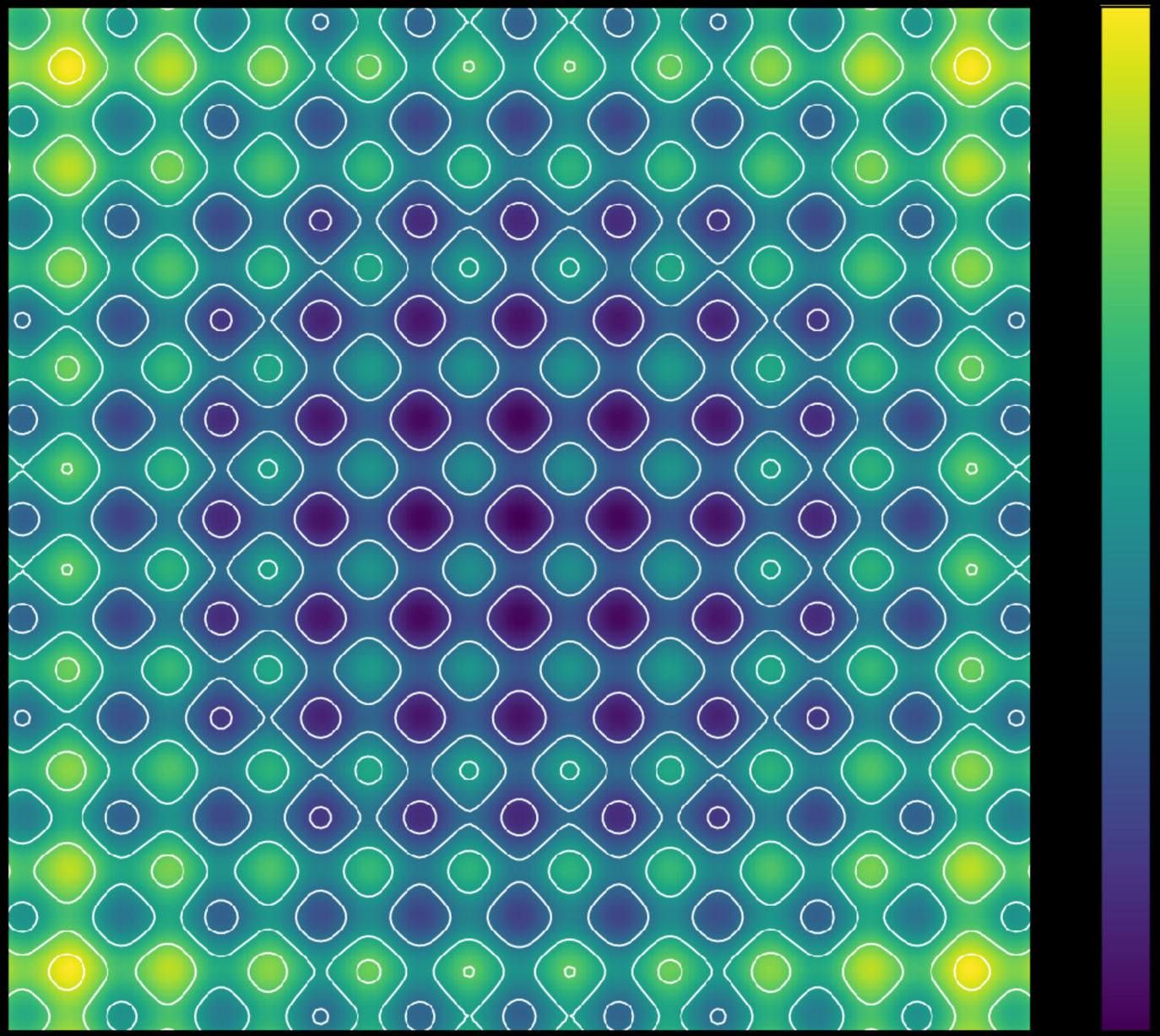
Same general network architecture



Find the minimum without getting stuck




Rastrigin function



Rastrigin function

# Search space size

Let’s say you want to hypertune:

- Linear units (between 16 and 256)
- Depth of linear layers (between 1 and 10)
- Convolution filters (between 16 and 64)
- kernel size (between 3 and 5)
- dropout (4 options)
- depth of convolutions (between 1 and 5)
- learning rate (4 options)
- optimizers (3 options)
- activation functions (3 options)
- batchsize (between 16 and 128)

Can you calculate how big the search space is?

# Search space size

- $256 \times 10 \times 64 \times 3 \times 4 \times 5 \times 4 \times 3 \times 3 \times 128$
- $45.298.483.200$
- $4.5 \times 10^{10}$ with 3 minutes per test
- about $1.3 \times 10^{11}$ minutes
- about $258.000$ years

# The curse of dimensionality

The search space has about $4.5 \times 10^{10}$ options. This is roughly the amount of sand when you fill your $40m^2$ living room with 50 cm of sand.

If you test at random 50 options, you are just scanning a very small fraction of all options.

You shouldn’t be surprised if the result of random hypertuning would be worse than the result of your manual hypertuning!


searchspace too big

# Manual search

Manual tuning is necessary to develop some intuition about complexity of models.

When encountering a problem, you will be able to make a reasonable guess:

- Have you already solved problems that are sort-of-similar?
- What are ranges of parameters you have seen working in similar complexity?
- How much time does it take / do you have to hypertune a problem?
- How do you spot learning (or a lack of it) in tensorboard?

# Manual search

When encountering a problem, it helps to have reference points:

- This problem seems more complex than MNIST, but not as complex as the flower photos;

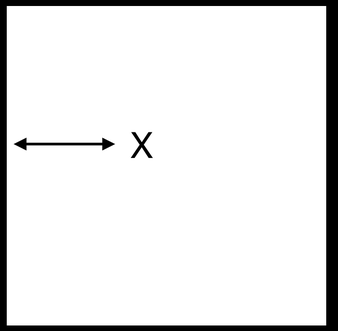
So let’s start with settings somewhere between what worked for those two.

However, *after* you have an intuition, we can get the help of algorithms.

# Things behave different in high-dimensional space

Pick a random point in a $d$-dimensional unit square/hypercube. Let the distance from the border be $\delta$.

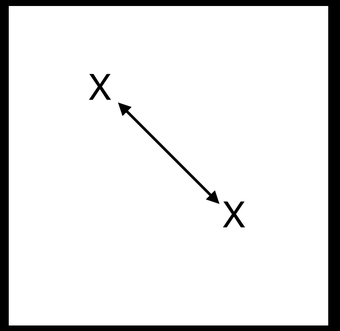
1. For $d = 2$ (square), $P(\delta < 0.001) = 0.004$
2. For $d = 10{,}000$, $P(\delta < 0.001) = 0.99999999$



# Things behave different in high-dimensional space

Pick two points randomly in a $d$-dimensional unit-square/cube/hypercube.

1. For $d = 2$ (square), expected distance is 0.52
2. For $d = 3$ (cube), expected distance is 0.66
3. For $d = 1{,}000{,}000$, expected distance is 408.25



# Hypertuning

- Manual tuning helps you to restrict your hyperparameter search
- Combined with your theoretical knowledge,
- you will be able to use hypertuning at scale with an algorithm

# Tuning frameworks

## Algorithms for tuning algorithms

- we will use Ray tune
- HyperOpt is another alternative
- You want a framework that is agnostic to ML frameworks; this way you dont have to switch if your framework changes.
- Ray offers parallel computing, also for training, data preprocessing, etc.



| | HyperOpt | Optuna | Keras-Tuner | HpBandSter | Tune |
|---|---|---|---|---|---|
| Distributed Execution | ✅ | ✅ | ☐ | ✅ | ✅ |
| Fault Tolerance | ☐ | ☐ | ☐ | ☐ | ✅ |
| Search algorithms | 2 | 4 | 3 | 3 | 6 |
| Framework Support (TF/Keras, Sklearn, PyTorch) | ✅ | ✅ | ☐ | ☐ | ✅ |

# Keeping track

There are a lot of ways to keep track of your experiments.

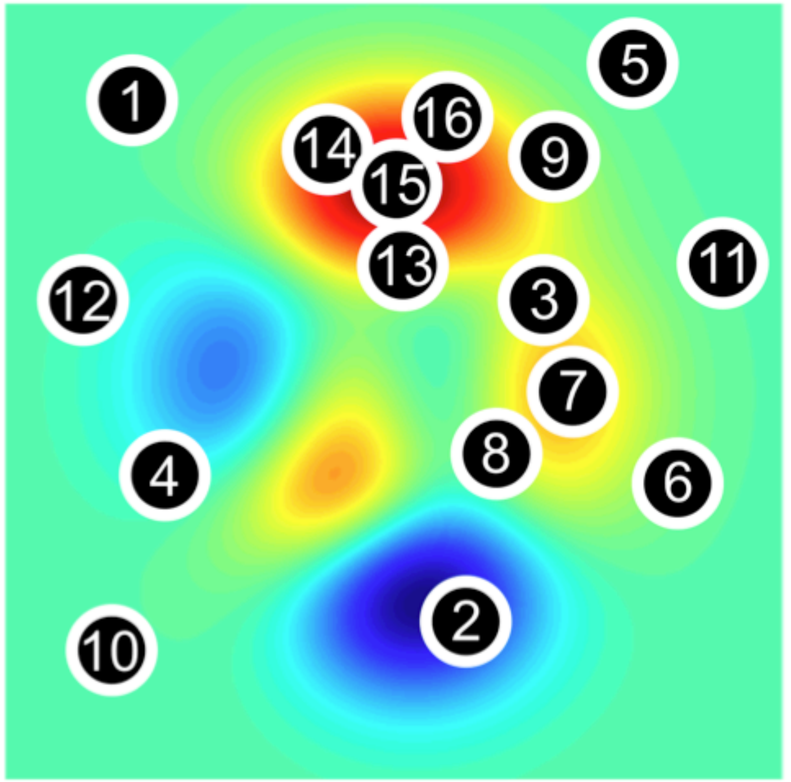
- **tomlserializer:** uses small toml files to log your hyperparameters
- **TensorBoard**: integrated in vscode, nice to have fast feedback on your manual hypertuning
- **MLflow**: database and dashboard, ready for production
- **Ray**: An open source framework to build and scale (distributed) ML: batch inference, model training, hyperparameter tuning, model serving.

# Hypertune algorithms

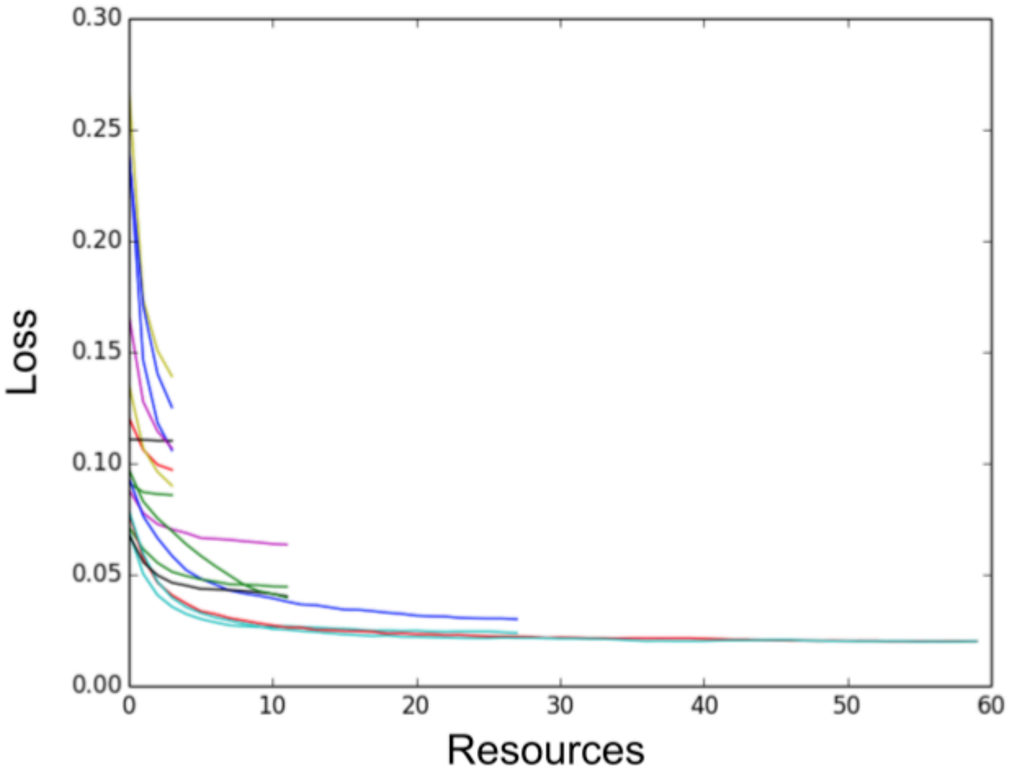
- bayesian search
- hyperband


Baysian: how high is the probability that you find value in this area?

Hyperband: only the best performing configurations stay in the game




(a) Configuration Selection




(b) Configuration Evaluation

Baysian

Hyperband

# Hypertune algorithms

- bayesian search
- hyperband
- Bayesian Optimization and Hyperband (BOHB) combines bayesian & hyperband techniques.

# Hypertune algorithms

- bayesian search
- hyperband
- Bayesian Optimization and Hyperband (BOHB)
- Tree Parzen Estimators (bayesian optimization over awkward search spaces with real, discrete and conditional parameters)
- PSO: Particle Swarm Optimization, swarm intelligence for difficult loss surfaces (eg rastrigin). Can be used via ray with `NevergradSearch`

Table III
RESULT OF BENCHMARK-I

| Dataset | Method | Score | Time |
|---|---|---|---|
| dna | HyperOpt Random | 0.9538178 | 366 |
| | HyperOpt TPE | 0.9579369 | 322 |
| | Optuna Random | 0.9580583 | 94 |
| | Optuna TPE | 0.9602619 | 521 |
| | Optunity PSO | 0.925369 | 200 |
| | SMAC | 0.9595218 | 807 |
| electricity | HyperOpt Random | 0.6969229 | 546 |
| | HyperOpt TPE | 0.7011213 | 341 |
| | Optuna Random | 0.6942903 | 494 |
| | Optuna TPE | 0.7011213 | 167 |
| | Optunity PSO | 0.9074971 | 190 |
| | SMAC | 0.7011213 | 393 |
| gas drift | HyperOpt Random | 0.850121 | 23869 |
| | HyperOpt TPE | 0.88168 | 14105 |
| | Optuna Random | 0.868639 | 14549 |
| | Optuna TPE | 0.882114 | 4403 |
| | Optunity PSO | 0.982105 | 11923 |
| | SMAC | 0.825001 | 12966 |
| nomao | HyperOpt Random | 0.926516 | 1711 |
| | HyperOpt TPE | 0.931569 | 1441 |
| | Optuna Random | 0.93677 | 1064 |
| | Optuna TPE | 0.937144 | 1124 |
| | Optunity PSO | 0.938943 | 1554 |
| | SMAC | 0.937003 | 2622 |
| pendigits | HyperOpt Random | 0.979188 | 1783 |
| | HyperOpt TPE | 0.987569 | 1012 |
| | Optuna Random | 0.982508 | 336 |
| | Optuna TPE | 0.983886 | 897 |
| | Optunity PSO | 0.958028 | 2980 |
| | SMAC | 0.984226 | 860 |
| semeion | HyperOpt Random | 0.901575 | 762 |
| | HyperOpt TPE | 0.915246 | 458 |
| | Optuna Random | 0.92832 | 895 |
| | Optuna TPE | 0.934616 | 1989 |
| | Optunity PSO | 0.700295 | 538 |
| | SMAC | 0.929794 | 1647 |

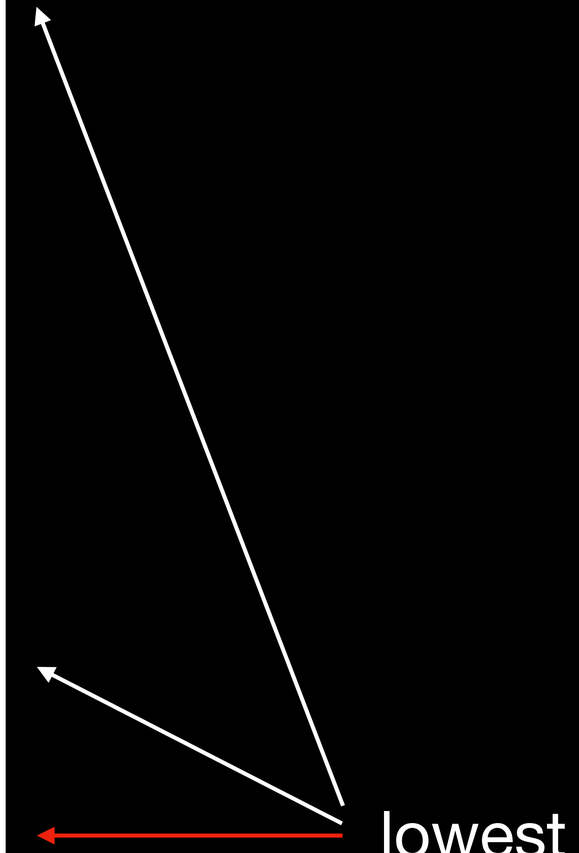
Table IV
CONFIGURATION SPACE FOR MLP TUNING. THE TYPE, NATURE OF SEARCH SPACE AND THE RANGE OF VALUES FOR SEARCHING ARE GIVEN AS COLUMNS.

| Hyper-parameter | Type | Space | Range |
|---|---|---|---|
| hidden layer size | integer | linear | (50, 200) |
| alpha | real | log | $(10^{-5}, 10)$ |
| batch size | integer | linear | (10, 250) |
| learning rate | real | log | $(10^{-5}, 10^{-1})$ |
| tolerance | real | log | $(10^{-5}, 10^{-1})$ |
| validation fraction | real | real | (0.1, 0.9) |
| beta 1 | real | logit | (0.5, 0.99) |
| beta 2 | real | logit | $(0.9, 1.0\text{-}10^{-6})$ |
| epsilon | real | log | $(10^{-9}, 10^{-6})$ |

A Comparative study of Hyper-Parameter Optimization Tools. Shekar S. et al., 2021, https://arxiv.org/abs/2201.06433

Table III
RESULT OF BENCHMARK-I

| Dataset | Method | Score | Time |
|---|---|---|---|
| dna | HyperOpt Random | 0.9538178 | 366 |
| | HyperOpt TPE | 0.9579369 | 322 |
| | Optuna Random | 0.9580583 | 94 |
| | Optuna TPE | 0.9602619 | 521 |
| | Optunity PSO | 0.925369 | 200 |
| | SMAC | 0.9595218 | 807 |
| electricity | HyperOpt Random | 0.6969229 | 546 |
| | HyperOpt TPE | 0.7011213 | 341 |
| | Optuna Random | 0.6942903 | 494 |
| | Optuna TPE | 0.7011213 | 167 |
| | Optunity PSO | 0.9074971 | 190 |
| | SMAC | 0.7011213 | 393 |
| gas drift | HyperOpt Random | 0.850121 | 23869 |
| | HyperOpt TPE | 0.88168 | 14105 |
| | Optuna Random | 0.868639 | 14549 |
| | Optuna TPE | 0.882114 | 4403 |
| | Optunity PSO | 0.982105 | 11923 |
| | SMAC | 0.825001 | 12966 |
| nomao | HyperOpt Random | 0.926516 | 1711 |
| | HyperOpt TPE | 0.931569 | 1441 |
| | Optuna Random | 0.93677 | 1064 |
| | Optuna TPE | 0.937144 | 1124 |
| | Optunity PSO | 0.938943 | 1554 |
| | SMAC | 0.937003 | 2622 |
| pendigits | HyperOpt Random | 0.979188 | 1783 |
| | HyperOpt TPE | 0.987569 | 1012 |
| | Optuna Random | 0.982508 | 336 |
| | Optuna TPE | 0.983886 | 897 |
| | Optunity PSO | 0.958028 | 2980 |
| | SMAC | 0.984226 | 860 |
| semeion | HyperOpt Random | 0.901575 | 762 |
| | HyperOpt TPE | 0.915246 | 458 |
| | Optuna Random | 0.92832 | 895 |
| | Optuna TPE | 0.934616 | 1989 |
| | Optunity PSO | 0.700295 | 538 |
| | SMAC | 0.929794 | 1647 |




A Comparative study of Hyper-Parameter Optimization Tools. Shekar S. et al., 2021, https://arxiv.org/abs/2201.06433

Table III
RESULT OF BENCHMARK-I

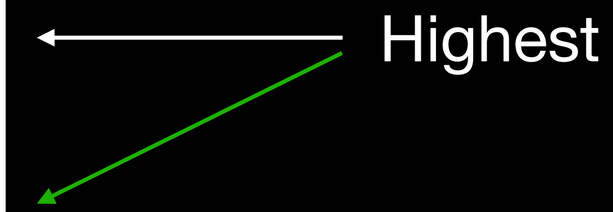
| Dataset | Method | Score | Time |
|---|---|---|---|
| dna | HyperOpt Random | 0.9538178 | 366 |
| | HyperOpt TPE | 0.9579369 | 322 |
| | Optuna Random | 0.9580583 | 94 |
| | Optuna TPE | 0.9602619 | 521 |
| | Optunity PSO | 0.925369 | 200 |
| | SMAC | 0.9595218 | 807 |
| electricity | HyperOpt Random | 0.6969229 | 546 |
| | HyperOpt TPE | 0.7011213 | 341 |
| | Optuna Random | 0.6942903 | 494 |
| | Optuna TPE | 0.7011213 | 167 |
| | Optunity PSO | 0.9074971 | 190 |
| | SMAC | 0.7011213 | 393 |
| gas drift | HyperOpt Random | 0.850121 | 23869 |
| | HyperOpt TPE | 0.88168 | 14105 |
| | Optuna Random | 0.868639 | 14549 |
| | Optuna TPE | 0.882114 | 4403 |
| | Optunity PSO | 0.982105 | 11923 |
| | SMAC | 0.825001 | 12966 |
| nomao | HyperOpt Random | 0.926516 | 1711 |
| | HyperOpt TPE | 0.931569 | 1441 |
| | Optuna Random | 0.93677 | 1064 |
| | Optuna TPE | 0.937144 | 1124 |
| | Optunity PSO | 0.938943 | 1554 |
| | SMAC | 0.937003 | 2622 |
| pendigits | HyperOpt Random | 0.979188 | 1783 |
| | HyperOpt TPE | 0.987569 | 1012 |
| | Optuna Random | 0.982508 | 336 |
| | Optuna TPE | 0.983886 | 897 |
| | Optunity PSO | 0.958028 | 2980 |
| | SMAC | 0.984226 | 860 |
| semeion | HyperOpt Random | 0.901575 | 762 |
| | HyperOpt TPE | 0.915246 | 458 |
| | Optuna Random | 0.92832 | 895 |
| | Optuna TPE | 0.934616 | 1989 |
| | Optunity PSO | 0.700295 | 538 |
| | SMAC | 0.929794 | 1647 |

Slowest

A Comparative study of Hyper-Parameter Optimization Tools. Shekar S. et al., 2021, https://arxiv.org/abs/2201.06433

Table III
RESULT OF BENCHMARK-I

| Dataset | Method | Score | Time |
|---|---|---|---|
| dna | HyperOpt Random | 0.9538178 | 366 |
| | HyperOpt TPE | 0.9579369 | 322 |
| | Optuna Random | 0.9580583 | 94 |
| | Optuna TPE | 0.9602619 | 521 |
| | Optunity PSO | 0.925369 | 200 |
| | SMAC | 0.9595218 | 807 |
| electricity | HyperOpt Random | 0.6969229 | 546 |
| | HyperOpt TPE | 0.7011213 | 341 |
| | Optuna Random | 0.6942903 | 494 |
| | Optuna TPE | 0.7011213 | 167 |
| | Optunity PSO | 0.9074971 | 190 |
| | SMAC | 0.7011213 | 393 |
| gas drift | HyperOpt Random | 0.850121 | 23869 |
| | HyperOpt TPE | 0.88168 | 14105 |
| | Optuna Random | 0.868639 | 14549 |
| | Optuna TPE | 0.882114 | 4403 |
| | Optunity PSO | 0.982105 | 11923 |
| | SMAC | 0.825001 | 12966 |
| nomao | HyperOpt Random | 0.926516 | 1711 |
| | HyperOpt TPE | 0.931569 | 1441 |
| | Optuna Random | 0.93677 | 1064 |
| | Optuna TPE | 0.937144 | 1124 |
| | Optunity PSO | 0.938943 | 1554 |
| | SMAC | 0.937003 | 2622 |
| pendigits | HyperOpt Random | 0.979188 | 1783 |
| | HyperOpt TPE | 0.987569 | 1012 |
| | Optuna Random | 0.982508 | 336 |
| | Optuna TPE | 0.983886 | 897 |
| | Optunity PSO | 0.958028 | 2980 |
| | SMAC | 0.984226 | 860 |
| semeion | HyperOpt Random | 0.901575 | 762 |
| | HyperOpt TPE | 0.915246 | 458 |
| | Optuna Random | 0.92832 | 895 |
| | Optuna TPE | 0.934616 | 1989 |
| | Optunity PSO | 0.700295 | 538 |
| | SMAC | 0.929794 | 1647 |




A Comparative study of Hyper-Parameter Optimization Tools. Shekar S. et al., 2021, https://arxiv.org/abs/2201.06433

Table III
RESULT OF BENCHMARK-I

| Dataset | Method | Score | Time |
|---|---|---|---|
| dna | HyperOpt Random | 0.9538178 | 366 |
| | HyperOpt TPE | 0.9579369 | 322 |
| | Optuna Random | 0.9580583 | 94 |
| | Optuna TPE | 0.9602619 | 521 |
| | Optunity PSO | 0.925369 | 200 |
| | SMAC | 0.9595218 | 807 |
| electricity | HyperOpt Random | 0.6969229 | 546 |
| | HyperOpt TPE | 0.7011213 | 341 |
| | Optuna Random | 0.6942903 | 494 |
| | Optuna TPE | 0.7011213 | 167 |
| | Optunity PSO | 0.9074971 | 190 |
| | SMAC | 0.7011213 | 393 |
| gas drift | HyperOpt Random | 0.850121 | 23869 |
| | HyperOpt TPE | 0.88168 | 14105 |
| | Optuna Random | 0.868639 | 14549 |
| | Optuna TPE | 0.882114 | 4403 |
| | Optunity PSO | 0.982105 | 11923 |
| | SMAC | 0.825001 | 12966 |
| nomao | HyperOpt Random | 0.926516 | 1711 |
| | HyperOpt TPE | 0.931569 | 1441 |
| | Optuna Random | 0.93677 | 1064 |
| | Optuna TPE | 0.937144 | 1124 |
| | Optunity PSO | 0.938943 | 1554 |
| | SMAC | 0.937003 | 2622 |
| pendigits | HyperOpt Random | 0.979188 | 1783 |
| | HyperOpt TPE | 0.987569 | 1012 |
| | Optuna Random | 0.982508 | 336 |
| | Optuna TPE | 0.983886 | 897 |
| | Optunity PSO | 0.958028 | 2980 |
| | SMAC | 0.984226 | 860 |
| semeion | HyperOpt Random | 0.901575 | 762 |
| | HyperOpt TPE | 0.915246 | 458 |
| | Optuna Random | 0.92832 | 895 |
| | Optuna TPE | 0.934616 | 1989 |
| | Optunity PSO | 0.700295 | 538 |
| | SMAC | 0.929794 | 1647 |

Fastest

A Comparative study of Hyper-Parameter Optimization Tools. Shekar S. et al., 2021, https://arxiv.org/abs/2201.06433

# Learning rate schedulers

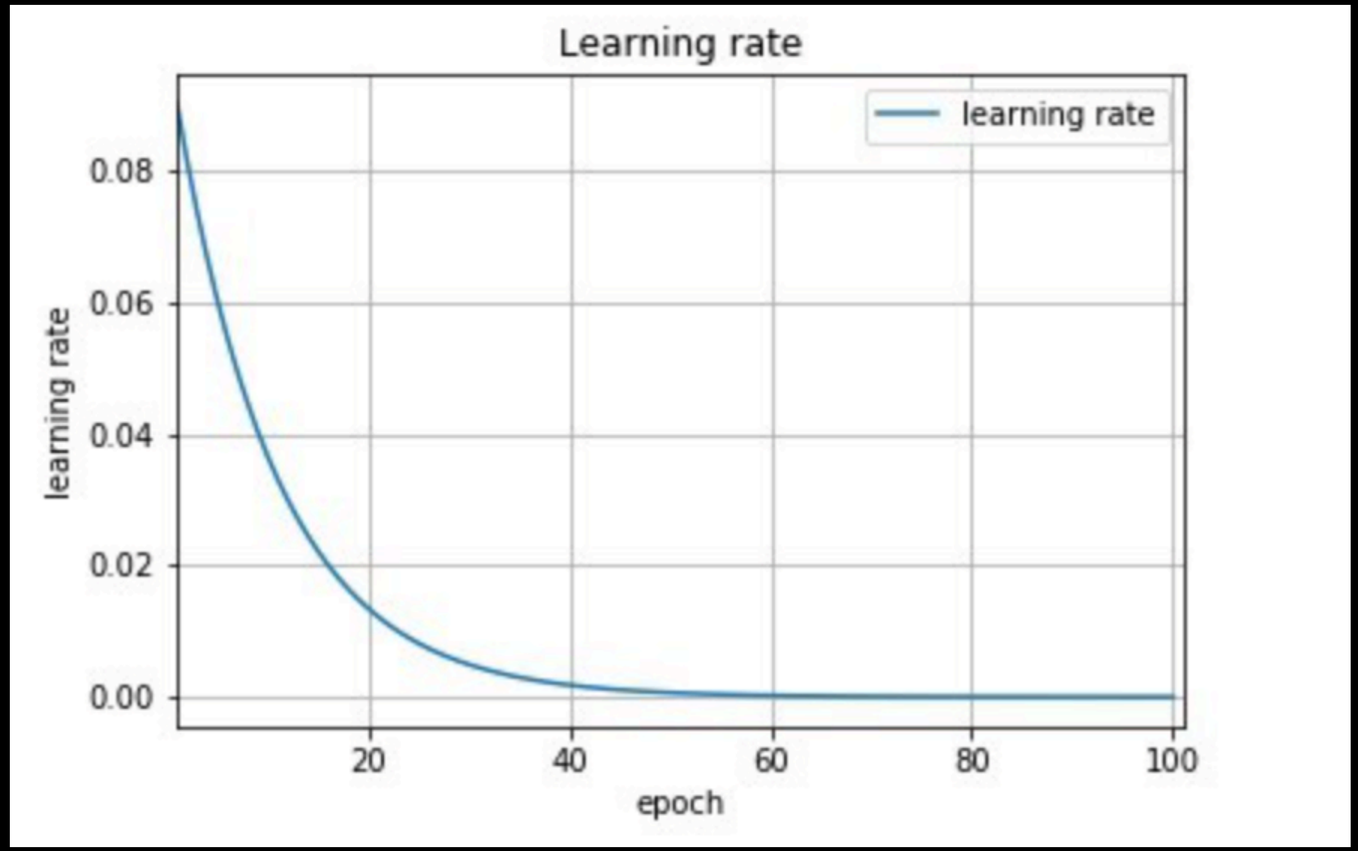
# Learning rate schedulers

The learning rate $\eta$ tells how fast weights are adjusted

$$W_{t+1} \leftarrow W_t - \eta \times gradient$$

The learning rate can be compared to different sizes of tools to carve wood. Big tools go fast, but miss the details.

So, often a decay of the learning rate is a good idea.

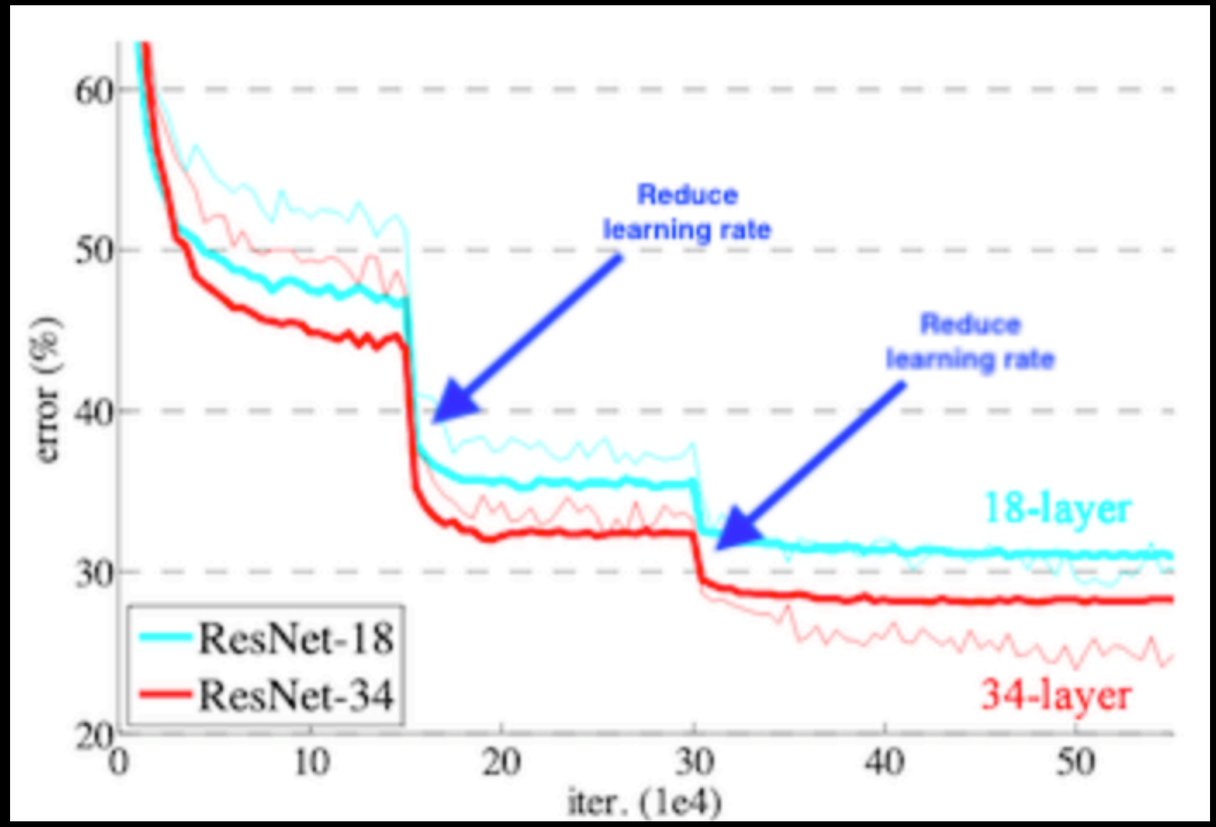



Too low
J(θ)
θ
Just right
J(θ)
θ
Too high
J(θ)
θ

# Learning rate schedulers

ResNet can be trained for very long. From the paper:

> *"The learning rate starts from 0.1 and is divided by 10 when the error plateaus, and the models are trained for up to* $60 \times 10^4$ *iterations"*



# Learning rate schedulers

The first  iterations can have very high gradients, and could “lock” the model in the wrong direction.

A solution is to gradually increase the learning rate from 0 on to the originally specified learning rate in the first few iterations.

This is typically used with transformer architectures.