Introduction to Relative Representations

Emanuele Rodolà

Sapienza University of Rome

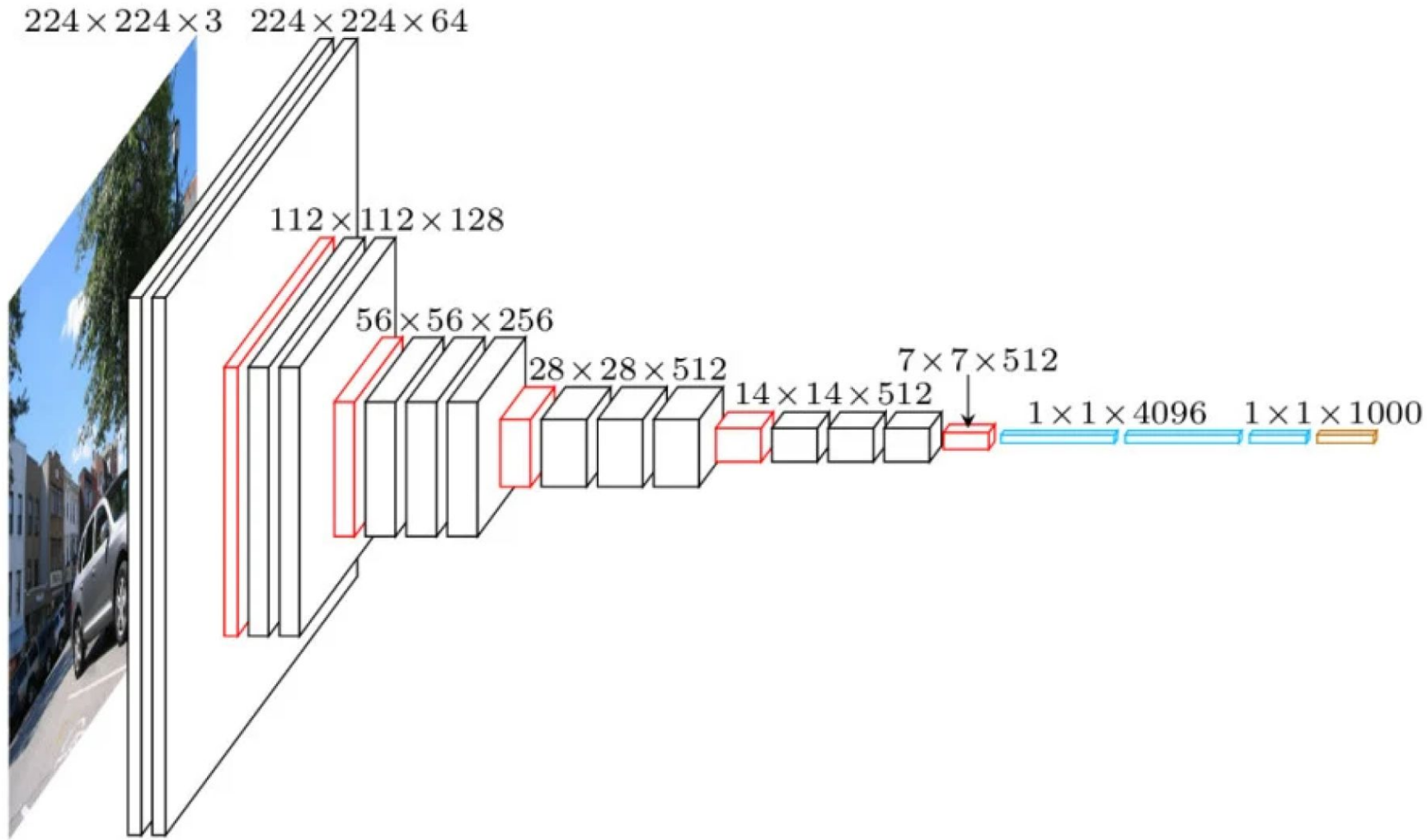
SAPIENZA
UNIVERSITÀ DI ROMA

Outline (tentative)

- 9:00 – 10:00 Background and motivation
- 10:00 – 11:00 Relative representations
- 11:00 – 12:00 Follow-ups
- 12:00 – 13:00 Lunch break
- 13:00 – 16:00 Open lab

Representations

In deep learning, we deal with highly parametrized models called deep neural networks:



We tend to concentrate on:

- Network architecture
- Choice of activations
- Data
- Loss

But these two have something to say:

- Hidden representations
- SGD

Hidden representations

$$\text{output} \leftarrow \underbrace{(\sigma \circ f)}_{\text{layer } n} \circ (\sigma \circ f) \circ \cdots \circ \underbrace{(\sigma \circ f)}_{\text{layer } 1}(\mathbf{x}) \leftarrow \text{input}$$

Each layer outputs an intermediate hidden representation:

$$\mathbf{x}_{\ell+1} = \sigma_\ell(\mathbf{W}_\ell \mathbf{x}_\ell + \mathbf{b}_\ell)$$

where we encode the weights at layer ℓ in the matrix $\mathbf{W}_\ell$ and bias $\mathbf{b}_\ell$.

Neurons

At each hidden layer we have:

$$\mathbf{x}_{\ell+1} = \sigma_\ell(\mathbf{W}_\ell \mathbf{x}_\ell)$$

Each row of the weight matrix is called a neuron or hidden unit:

$$\mathbf{W}\mathbf{x} = \begin{pmatrix} \text{— unit —} \\ \vdots \\ \text{— unit —} \end{pmatrix} \begin{pmatrix} | \\ \mathbf{x} \\ | \end{pmatrix}$$

Neurons

$$\mathbf{W}\mathbf{x} = \begin{pmatrix} \text{— unit —} \\ \vdots \\ \text{— unit —} \end{pmatrix} \begin{pmatrix} | \\ \mathbf{x} \\ | \end{pmatrix}$$

We have two interpretations:

1. Each layer is a vector-to-vector function $\mathbb{R}^p \to \mathbb{R}^q$.

Neurons

$$\mathbf{W}\mathbf{x} = \begin{pmatrix} \text{— unit —} \\ \vdots \\ \text{— unit —} \end{pmatrix} \begin{pmatrix} | \\ \mathbf{x} \\ | \end{pmatrix}$$

We have two interpretations:

1. Each layer is a vector-to-vector function $\mathbb{R}^p \to \mathbb{R}^q$.
2. Each layer has q units acting in parallel.
   Each unit acts as a scalar function $\mathbb{R}^p \to \mathbb{R}$.

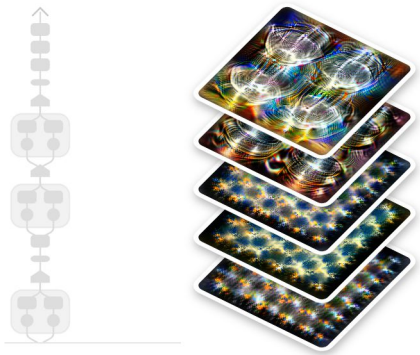
Visualization

General idea: **activation maximization** heuristic

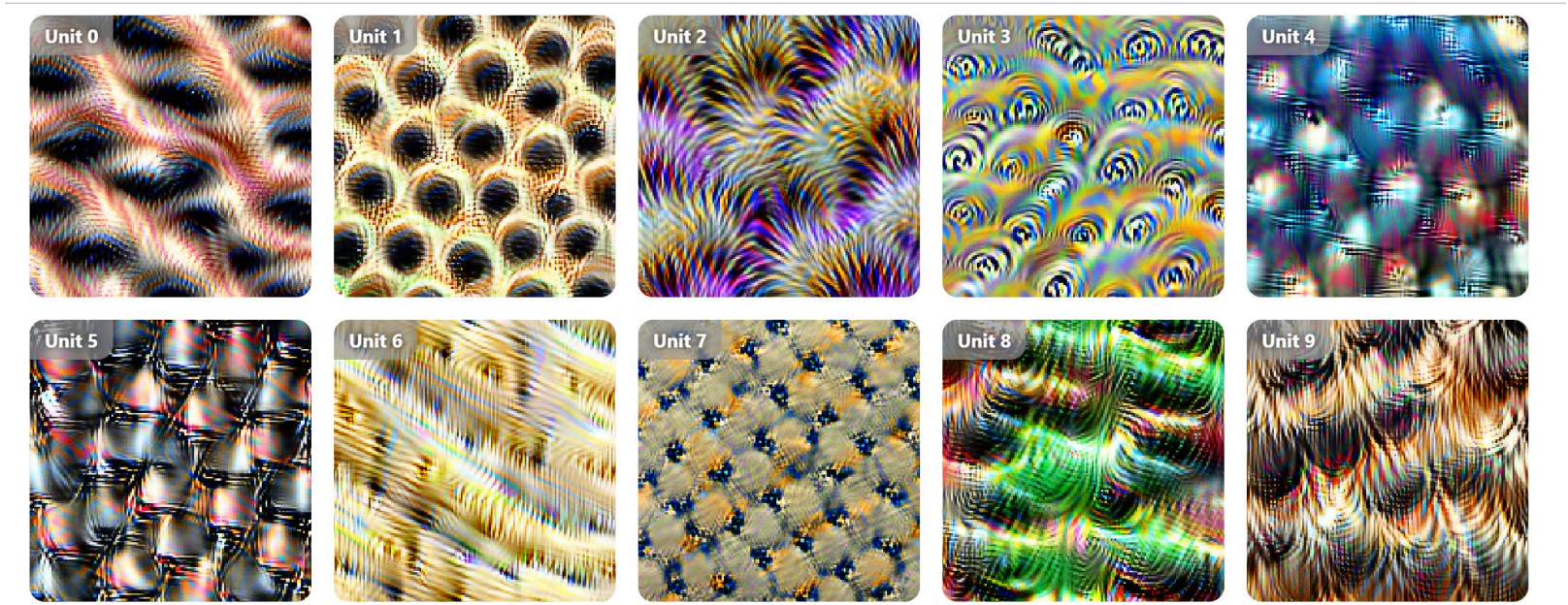
$$\mathbf{x}^* = \arg \max_{\mathbf{x}\ s.t.\ ||\mathbf{x}||=\rho} h_{ij}(\theta, \mathbf{x}).$$

AlexNet

A landmark in computer vision, this 2012 winner of ImageNet has over 50,000 citations.



26 nodes



https://openai.com/research/microscope

What affects the representations?

What affects the representations?

Recall:

$$\text{output} \leftarrow \underbrace{(\sigma \circ f)}_{\text{layer } n} \circ (\sigma \circ f) \circ \cdots \circ \underbrace{(\sigma \circ f)}_{\text{layer } 1}(\mathbf{x}) \leftarrow \text{input}$$

where each *f* is parametric and trained on **data** by minimizing a **loss**.

- Network architecture
- Choice of activations
- Data
- Loss
- Training hyperparameters

SGD is surprisingly robust

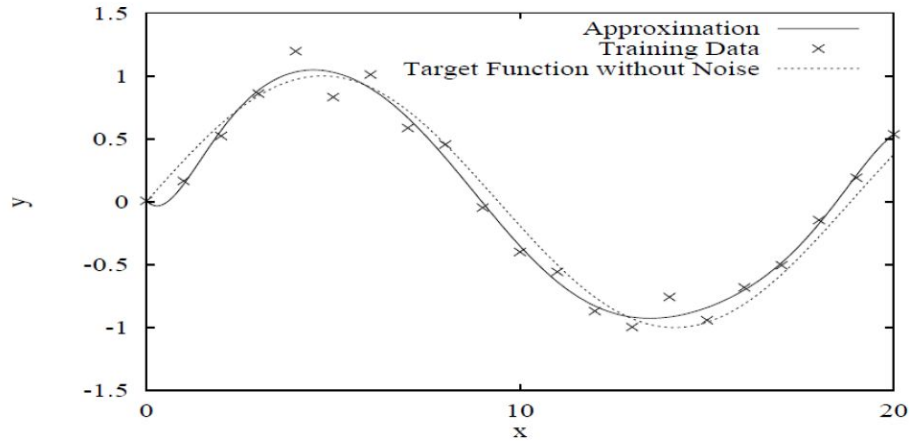
Old ML

Too many parameters lead to overfitting

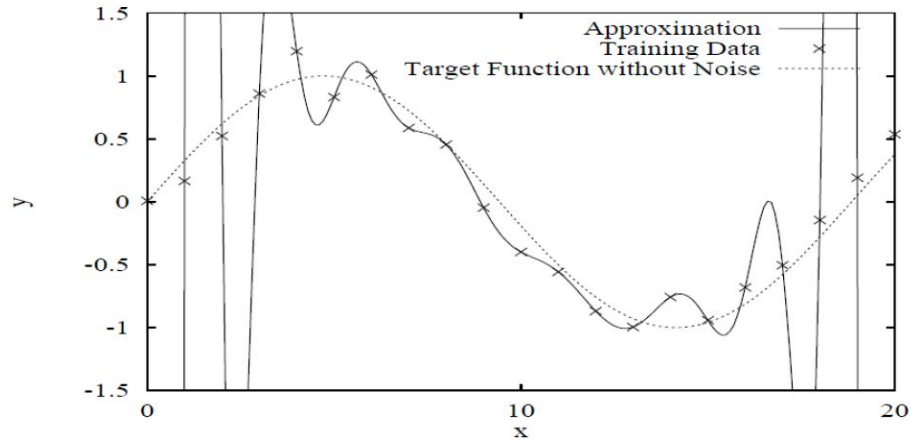
New ML

More parameters improve the solution space

Typical overfitting with polynomial regression:

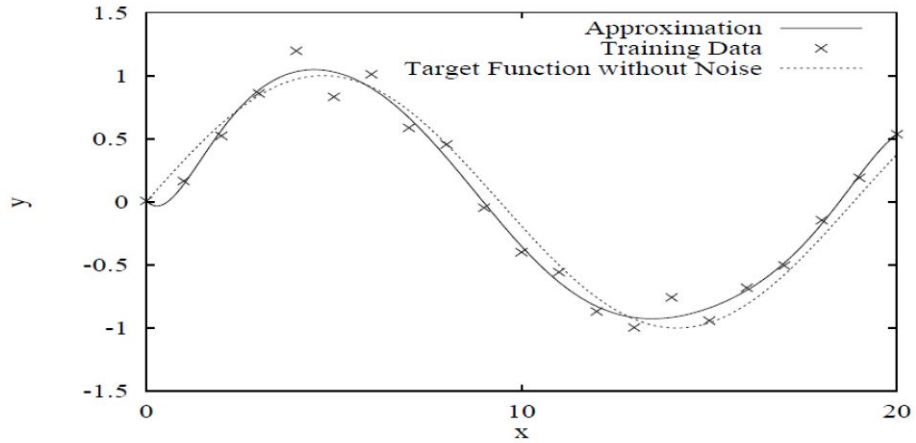


Order 10

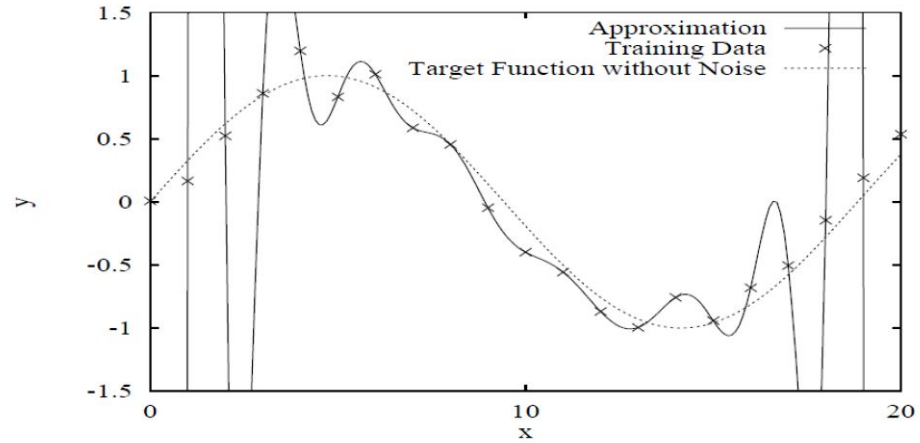


Order 20

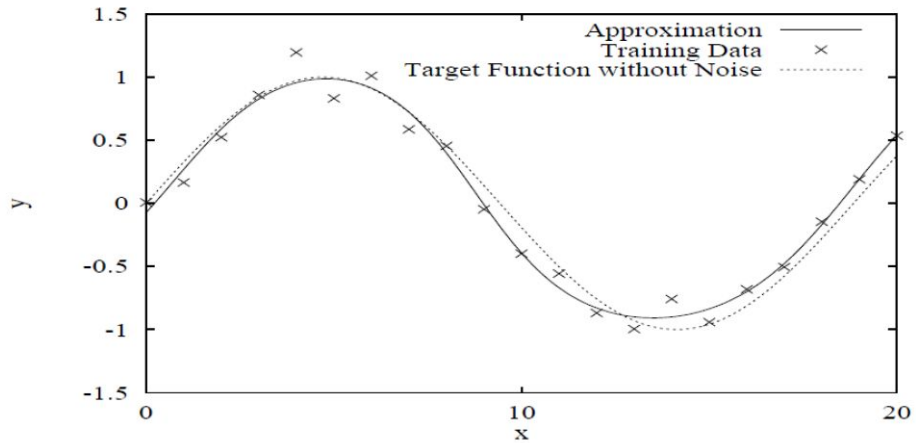
...but more MLP parameters not always lead to overfitting:



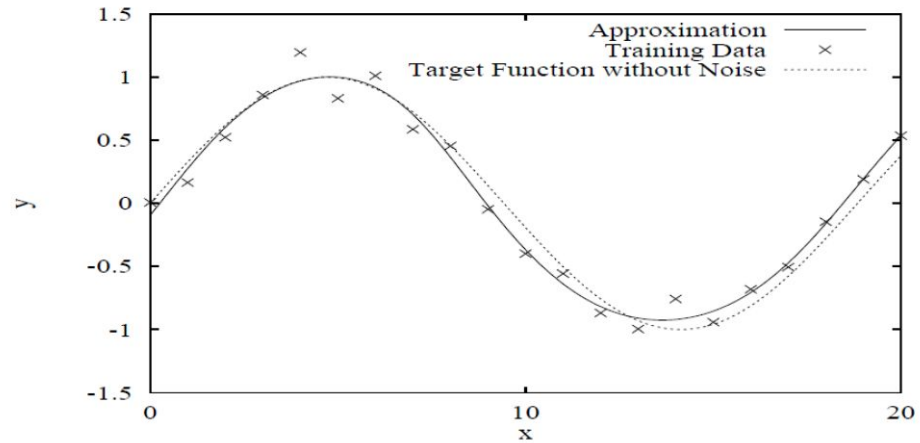
Order 10



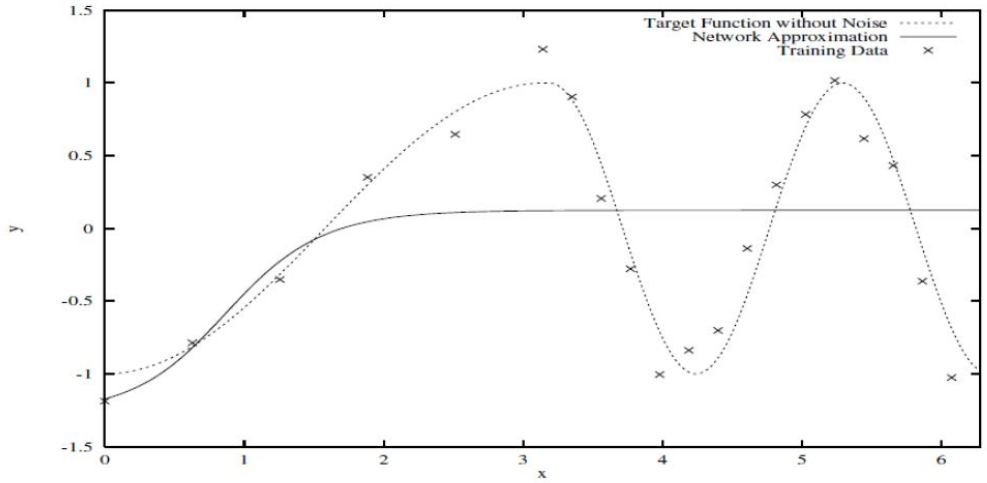
Order 20



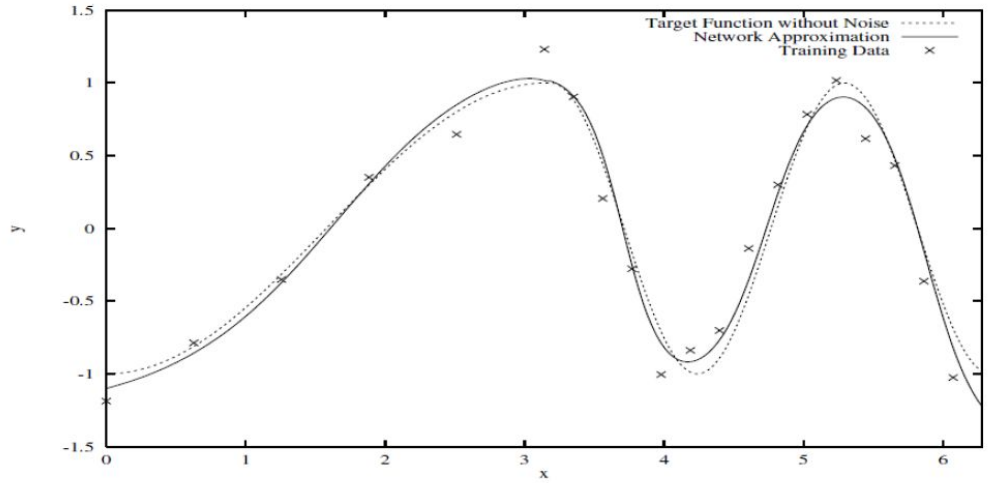
10 Hidden Nodes



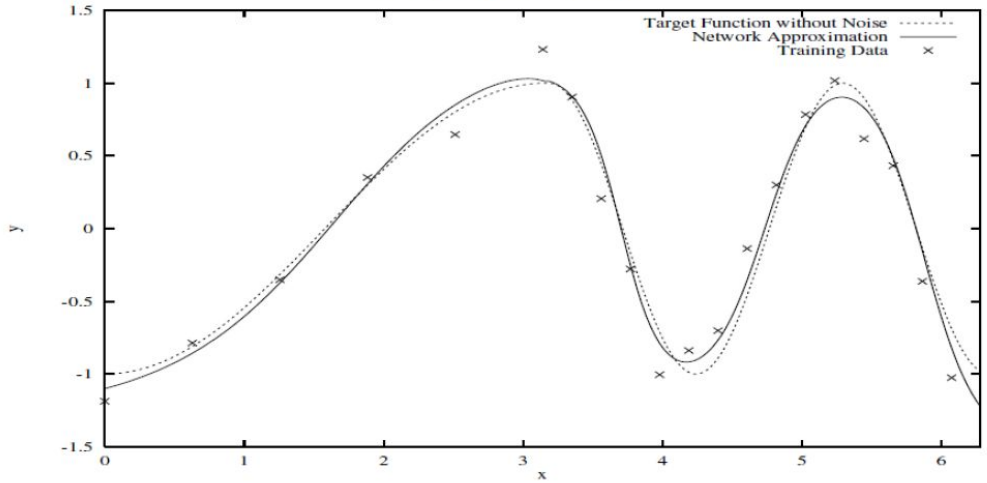
50 Hidden Nodes



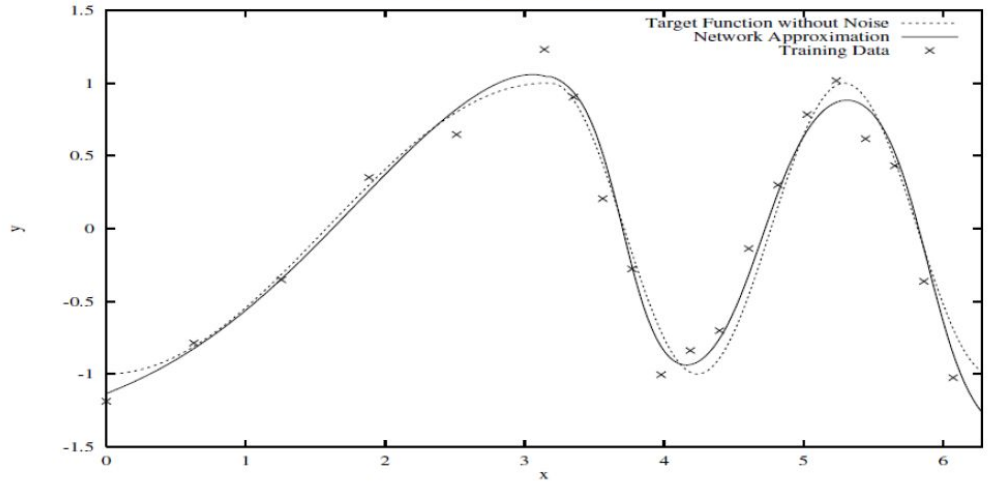
1 Hidden Unit



4 Hidden Units

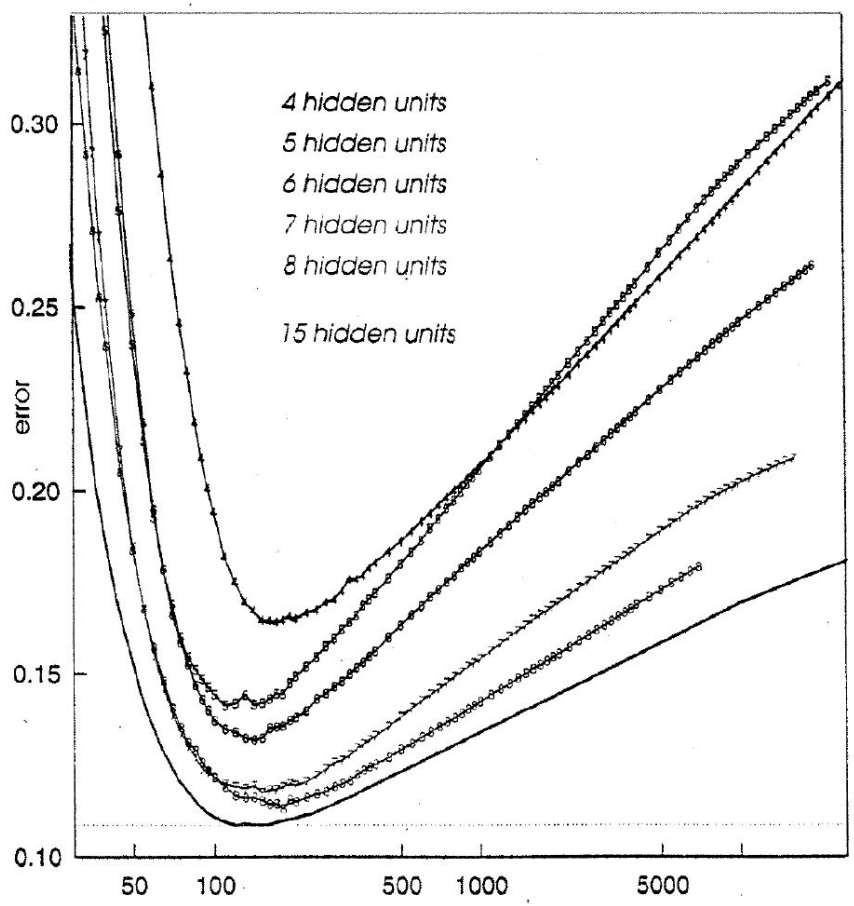


10 Hidden Units



100 Hidden Units

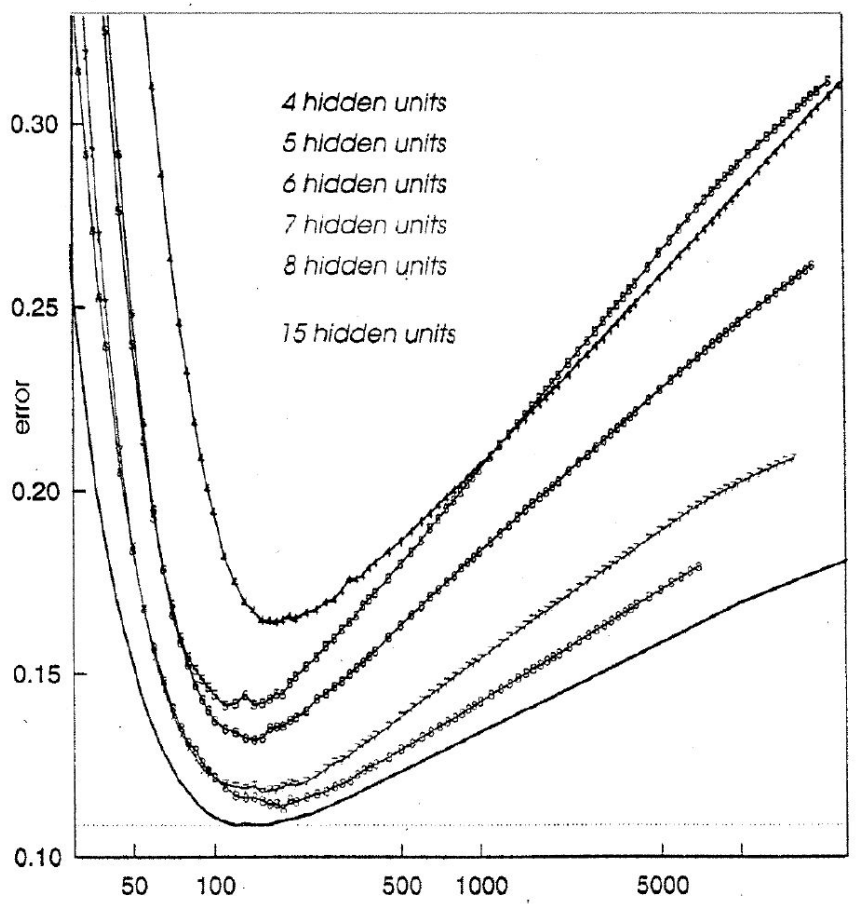
Overfitting can be recognized by looking at the validation error:



Weigend, "On overfitting and the effective number of hidden units", 1993

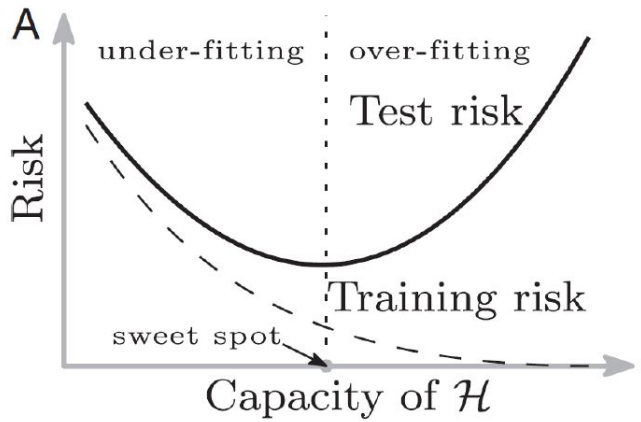
Overfitting can be recognized by looking at the validation error:

- Small networks can also overfit.
- Large networks have best performance if they stop early.
- Early stopping: Stop training as soon as performance on a validation set decreases.

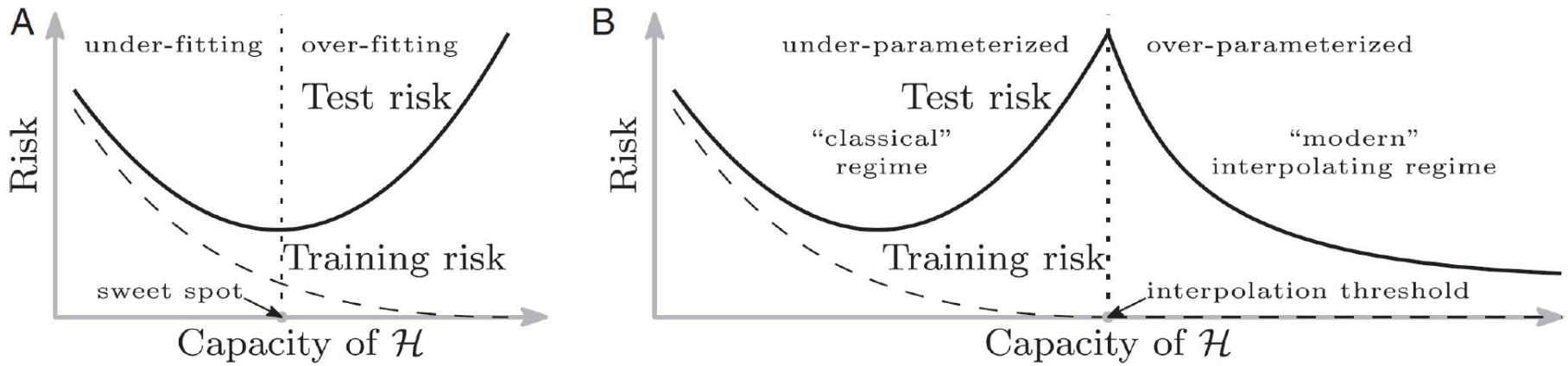


Weigend, "On overfitting and the effective number of hidden units", 1993

U-shaped curve as a function of # network parameters:

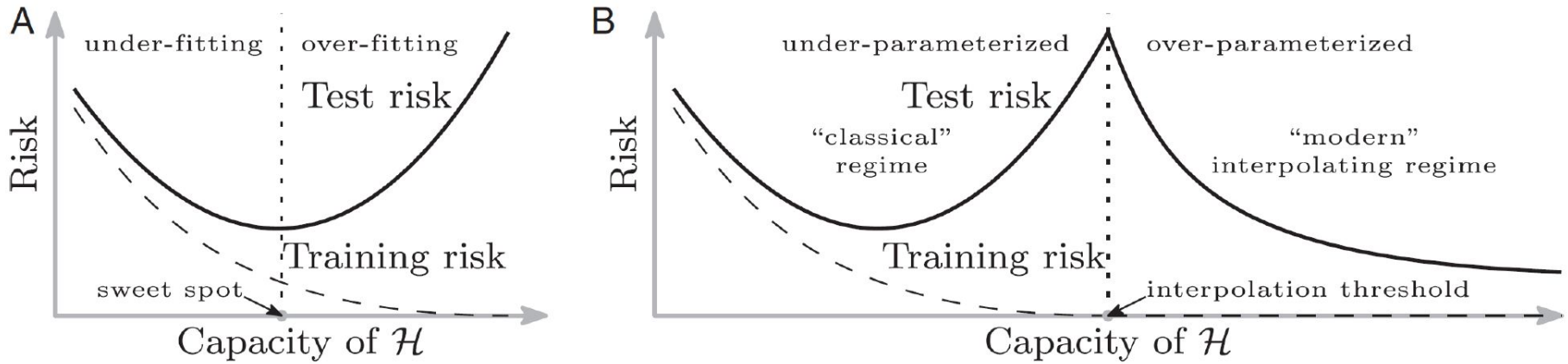


U-shaped curve as a function of # network parameters:



Interpolation: perfect fit on the training data.

U-shaped curve as a function of # network parameters:

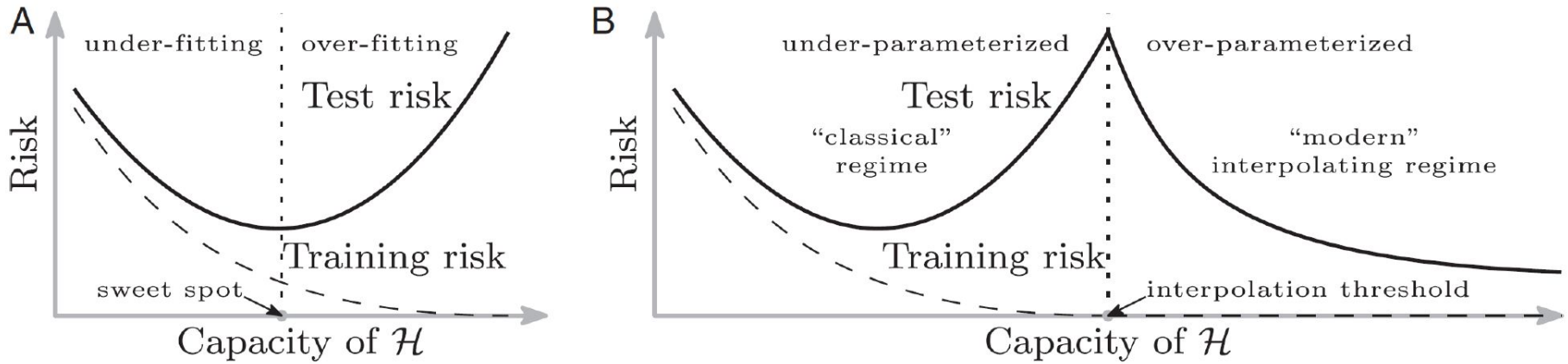


Interpolation: perfect fit on the training data.

"By considering larger function classes, which contain more candidate predictors compatible with the data, we are able to find interpolating functions that have smaller norm and are thus "simpler." Thus, increasing function class capacity improves performance of classifiers."

Belkin et al, "Reconciling modern machine-learning practice and the classical bias–variance trade-off", PNAS 2019

U-shaped curve as a function of # network parameters:



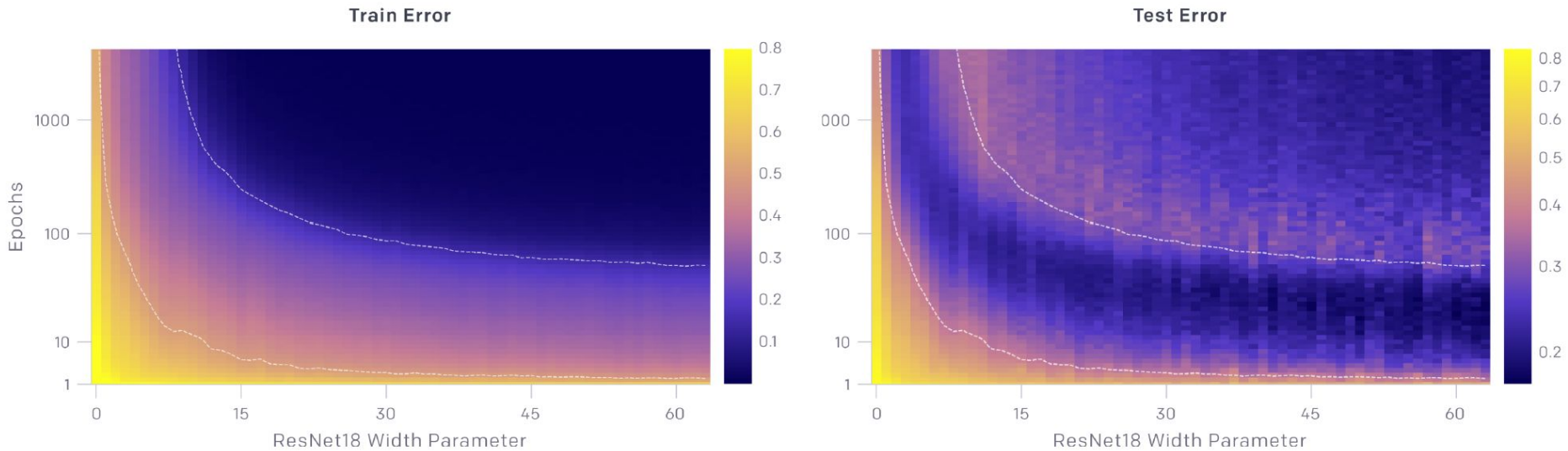
Interpolation: perfect fit on the training data.

"By considering larger function classes, which contain more candidate predictors compatible with the data, we are able to find interpolating functions that have smaller norm and are thus "simpler." Thus, increasing function class capacity improves performance of classifiers."

The surprising fact is that SGD is able to find such good models.

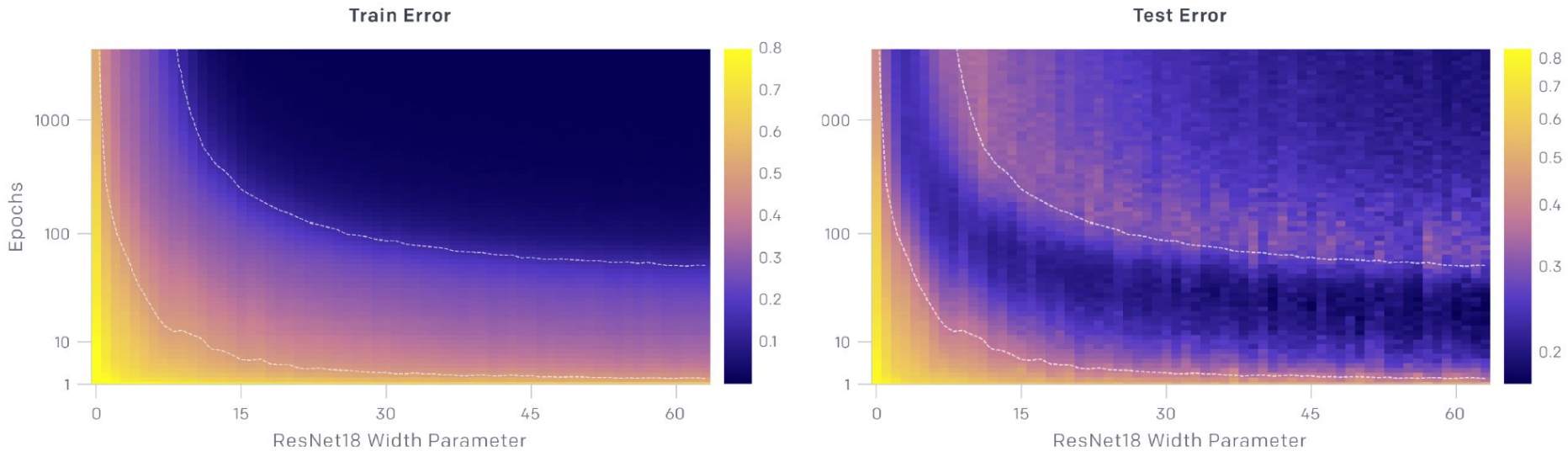
Belkin et al, "Reconciling modern machine-learning practice and the classical bias–variance trade-off", PNAS 2019

There is a regime where training longer reverses overfitting.



For a fixed number of epochs, the "usual" double descent.

There is a regime where training longer reverses overfitting.



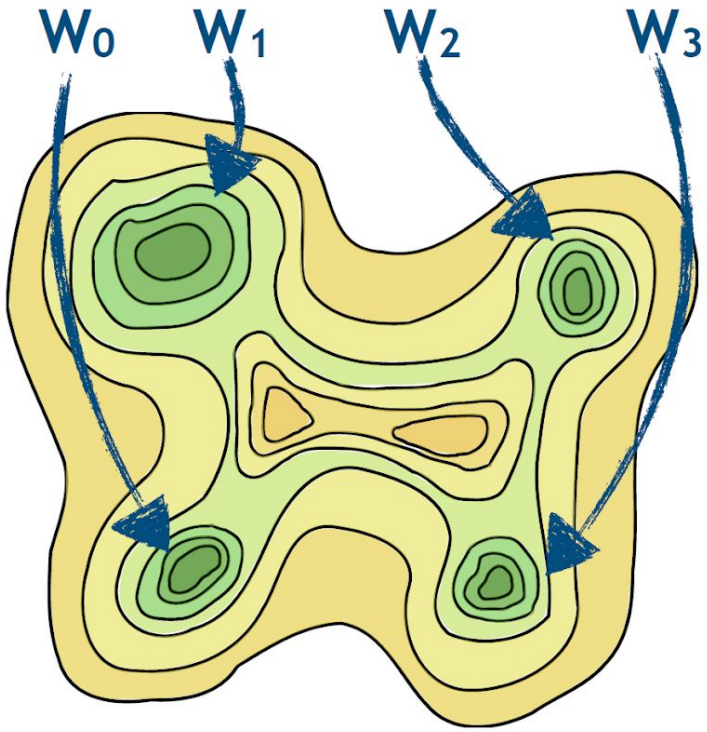
For a fixed number of epochs, the "usual" double descent.

For a fixed number of parameters, we observe double descent as a function of training time.

https://openai.com/blog/deep-double-descent/

Seeking equivalent models

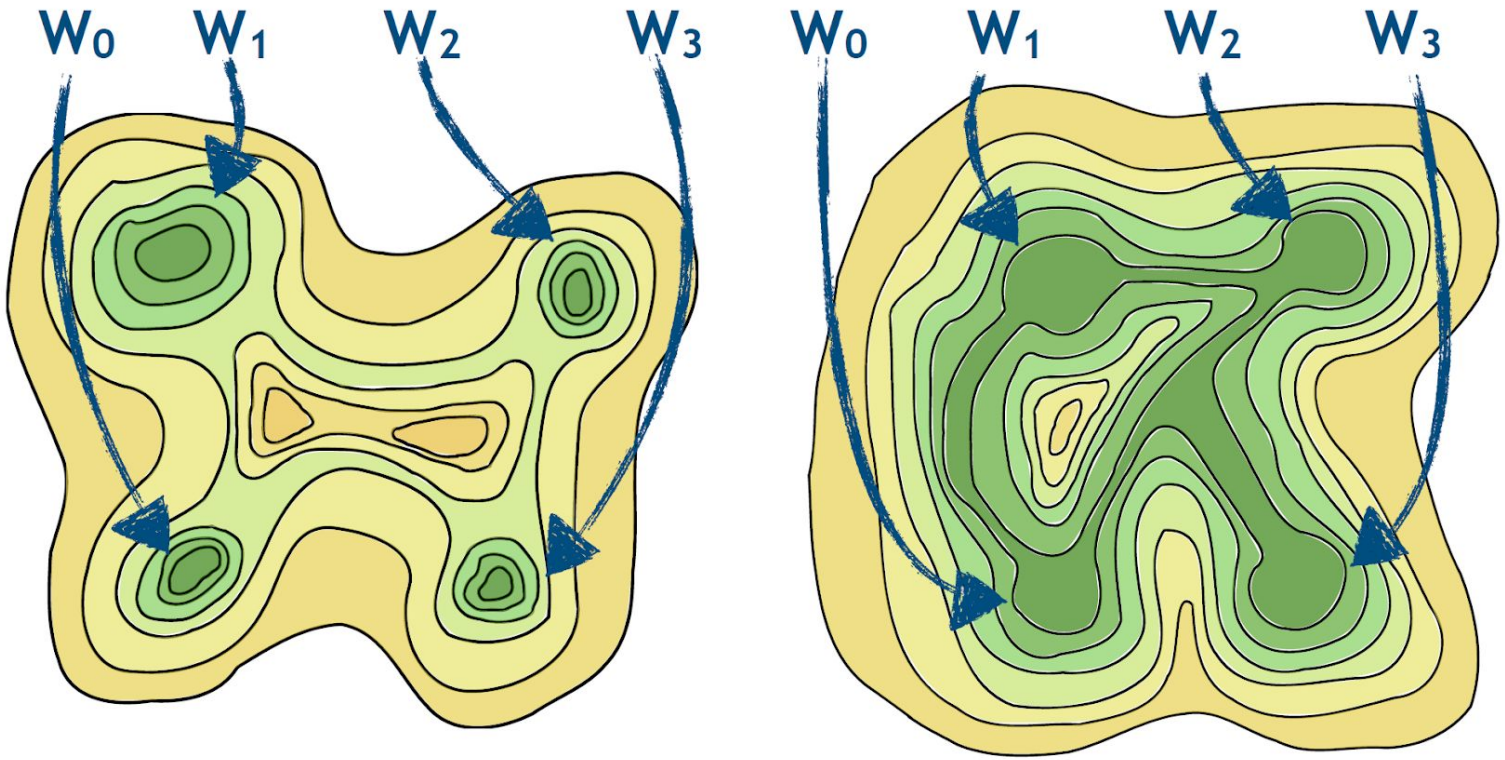
Progressive understanding of loss landscape



Islands

"Loss Surface Simplexes for Mode Connecting Volumes and Fast Ensembling". Benton et al, 2021

Progressive understanding of loss landscape

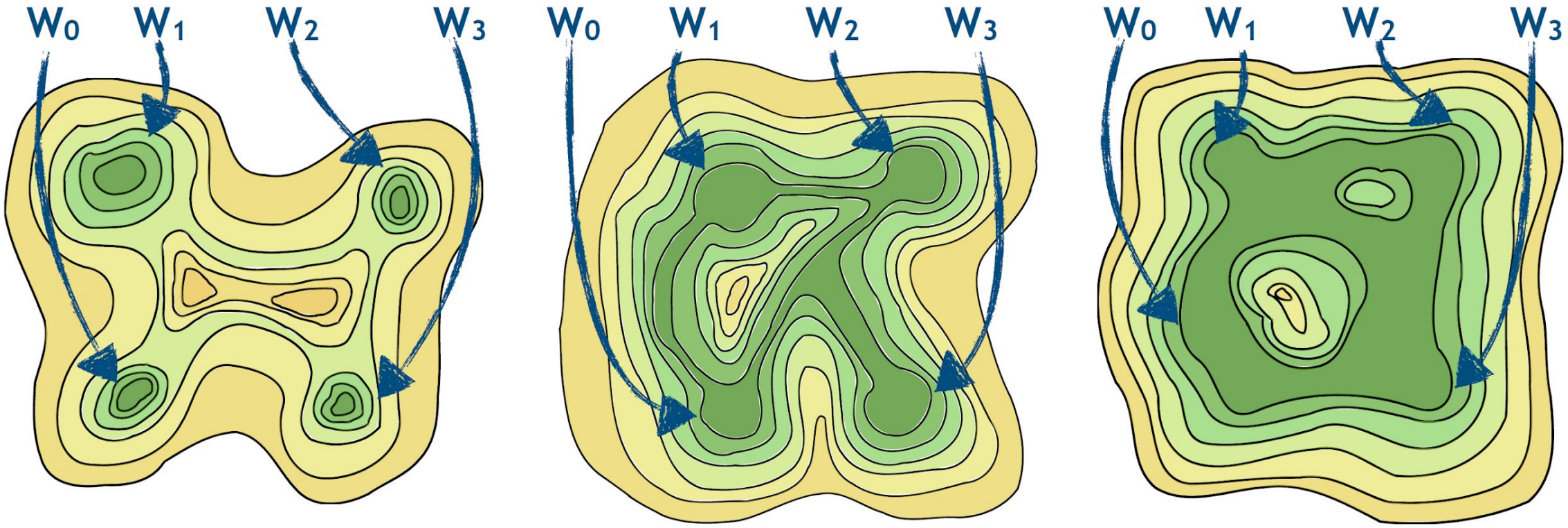


Islands

Tunnels

"Loss Surface Simplexes for Mode Connecting Volumes and Fast Ensembling". Benton et al, 2021

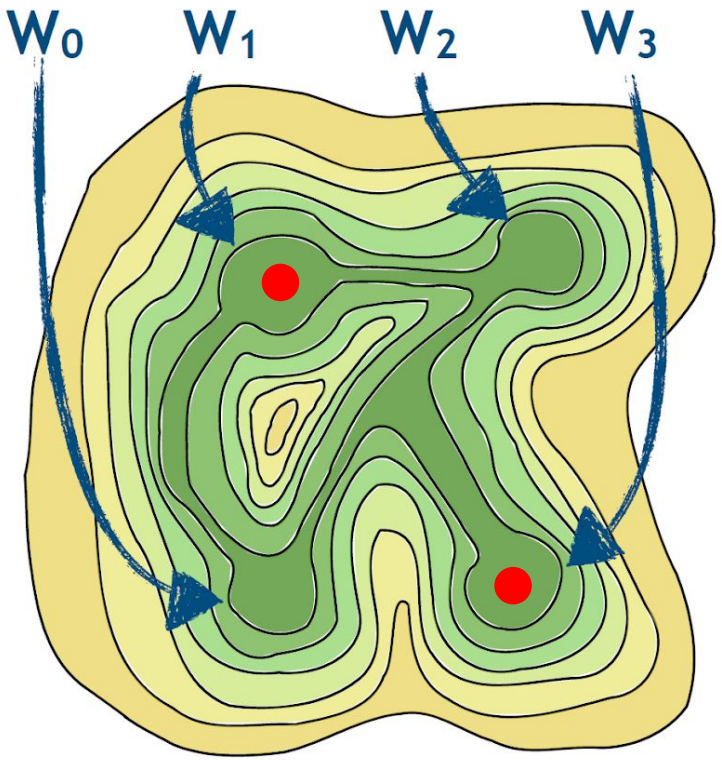
Progressive understanding of loss landscape



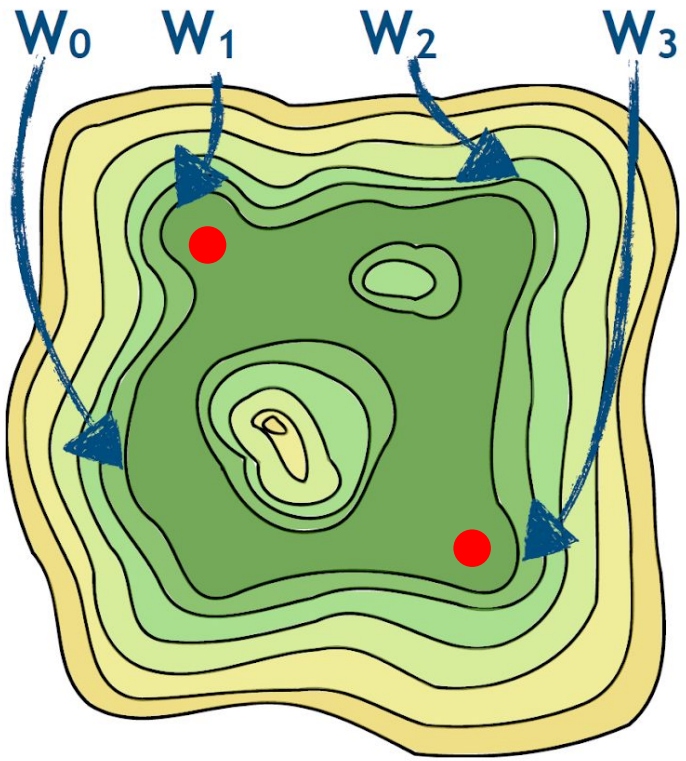
Islands Tunnels Volume

"Loss Surface Simplexes for Mode Connecting Volumes and Fast Ensembling". Benton et al, 2021

Mode connectivity

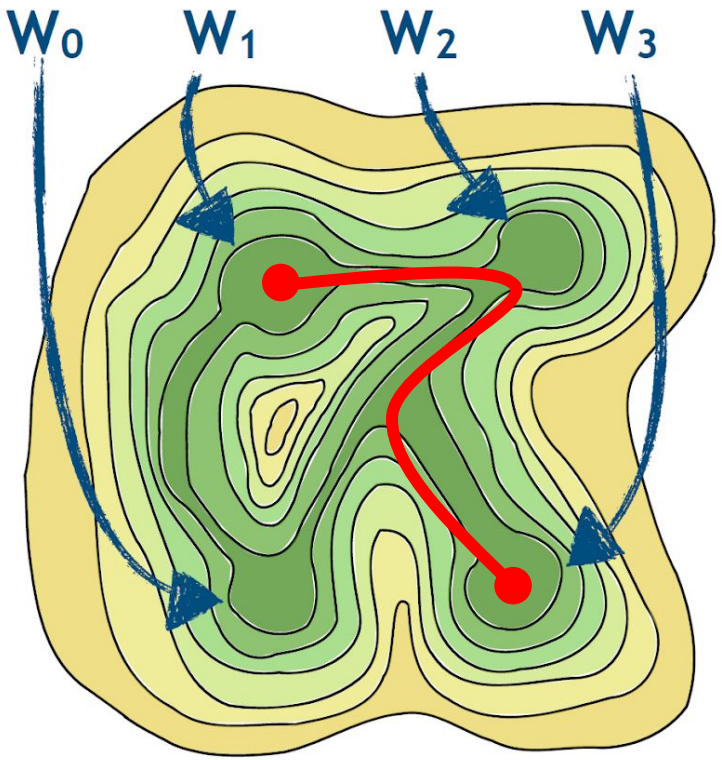


Tunnels

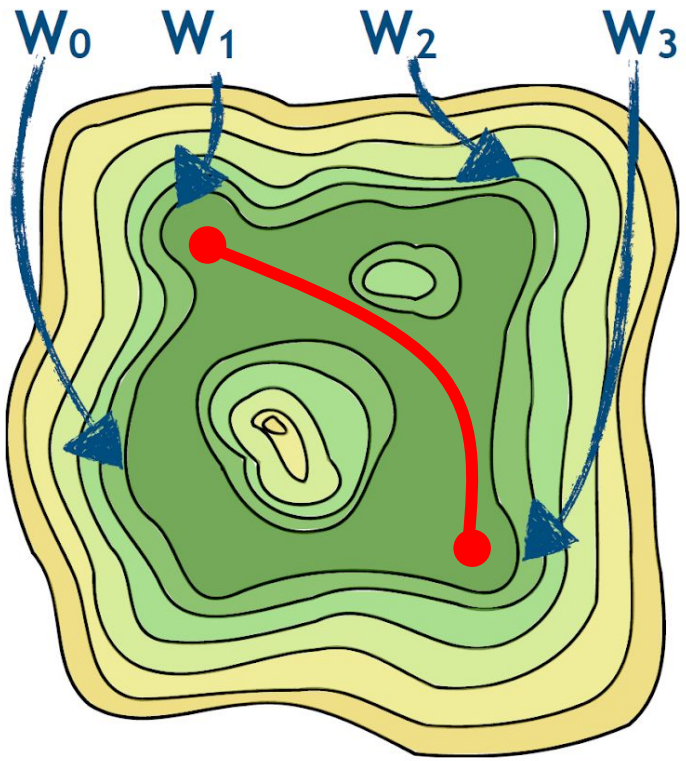


Volume

Mode connectivity



Tunnels

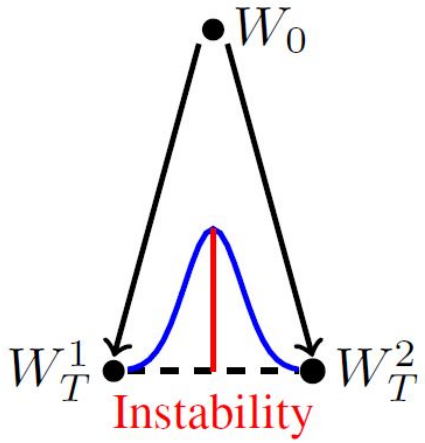


Volume

Linear mode connectivity

Algorithm:

- Train a network W^1
- Reshuffle the data and retrain to get W^2
- Linearly interpolate weights from W^1 to W^2

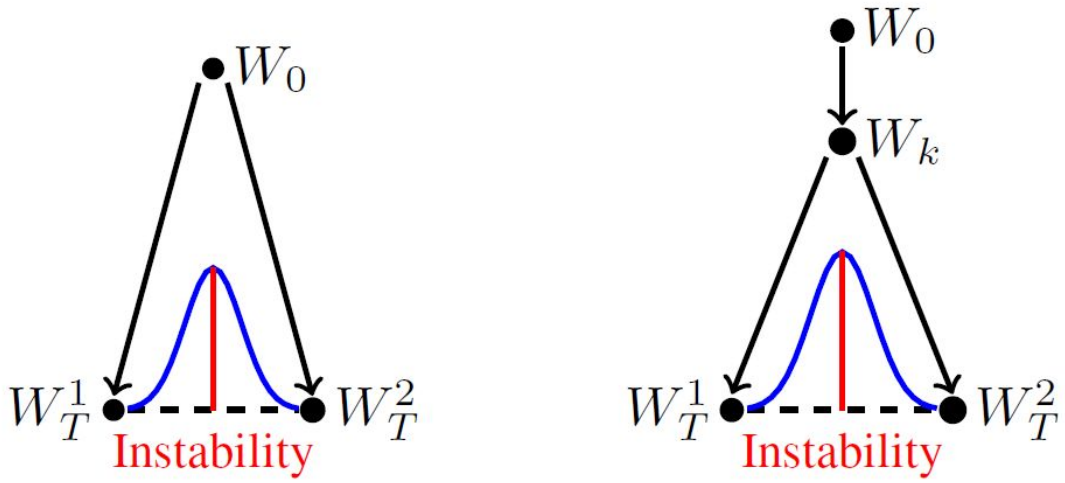


"Linear Mode Connectivity and the Lottery Ticket Hypothesis". Frankle et al, 2020

Linear mode connectivity

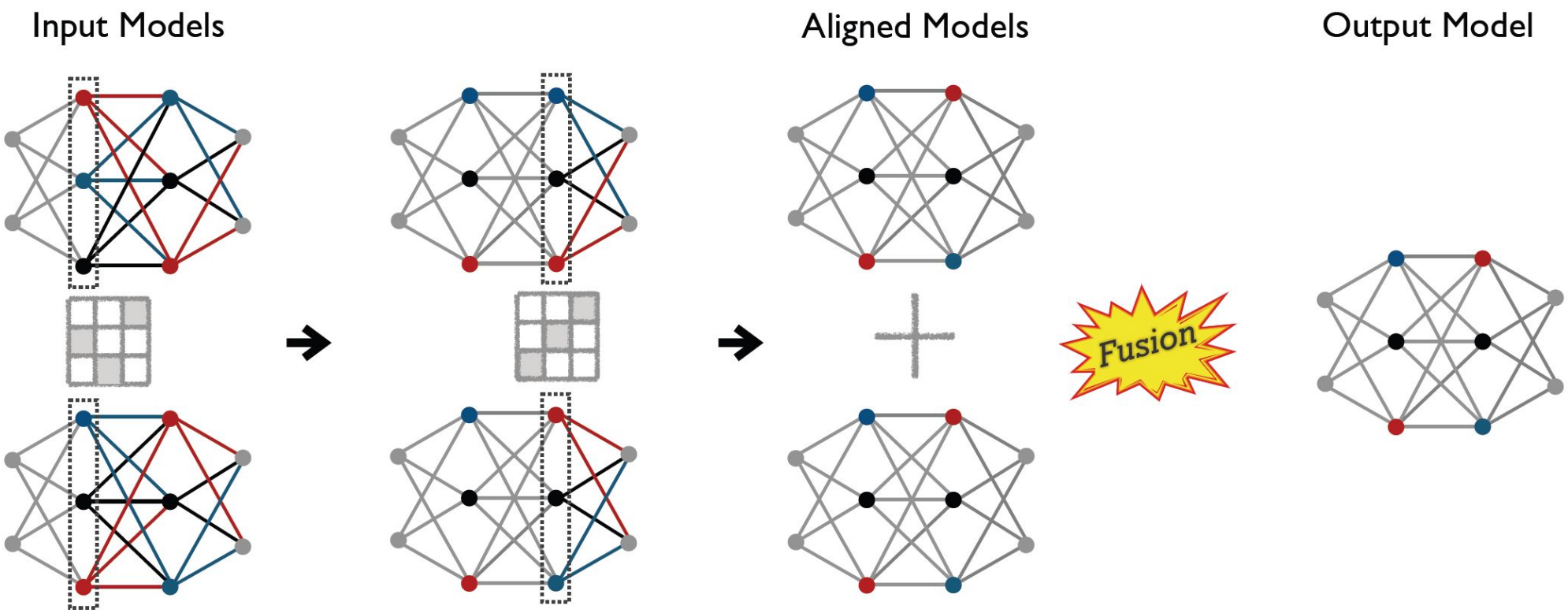
Algorithm:

- Train a network W^1
- Reshuffle the data and retrain to get W^2
- Linearly interpolate weights from W^1 to W^2



"Linear Mode Connectivity and the Lottery Ticket Hypothesis". Frankle et al, 2020

Equivalence up to neuron alignment



"Model Fusion via Optimal Transport". Singh and Jaggi, 2020

Equivalence up to permutation

Let ***P*** be a permutation matrix. Then:

$$\boldsymbol{z}_{\ell+1} = \boldsymbol{P}^\top \boldsymbol{P} \boldsymbol{z}_{\ell+1}$$

"Model Fusion via Optimal Transport". Singh and Jaggi, 2020

Equivalence up to permutation

Let ***P*** be a permutation matrix. Then:

$$\boldsymbol{z}_{\ell+1} = \boldsymbol{P}^\top \boldsymbol{P} \boldsymbol{z}_{\ell+1} = \boldsymbol{P}^\top \boldsymbol{P} \sigma(\boldsymbol{W}_\ell \boldsymbol{z}_\ell + \boldsymbol{b}_\ell)$$

"Model Fusion via Optimal Transport". Singh and Jaggi, 2020

Equivalence up to permutation

Let ***P*** be a permutation matrix. Then:

$$\boldsymbol{z}_{\ell+1} = \boldsymbol{P}^\top \boldsymbol{P} \boldsymbol{z}_{\ell+1} = \boldsymbol{P}^\top \boldsymbol{P} \sigma(\boldsymbol{W}_\ell \boldsymbol{z}_\ell + \boldsymbol{b}_\ell) = \boldsymbol{P}^\top \sigma(\boldsymbol{P}\boldsymbol{W}_\ell \boldsymbol{z}_\ell + \boldsymbol{P}\boldsymbol{b}_\ell)$$

"Model Fusion via Optimal Transport". Singh and Jaggi, 2020

Equivalence up to permutation

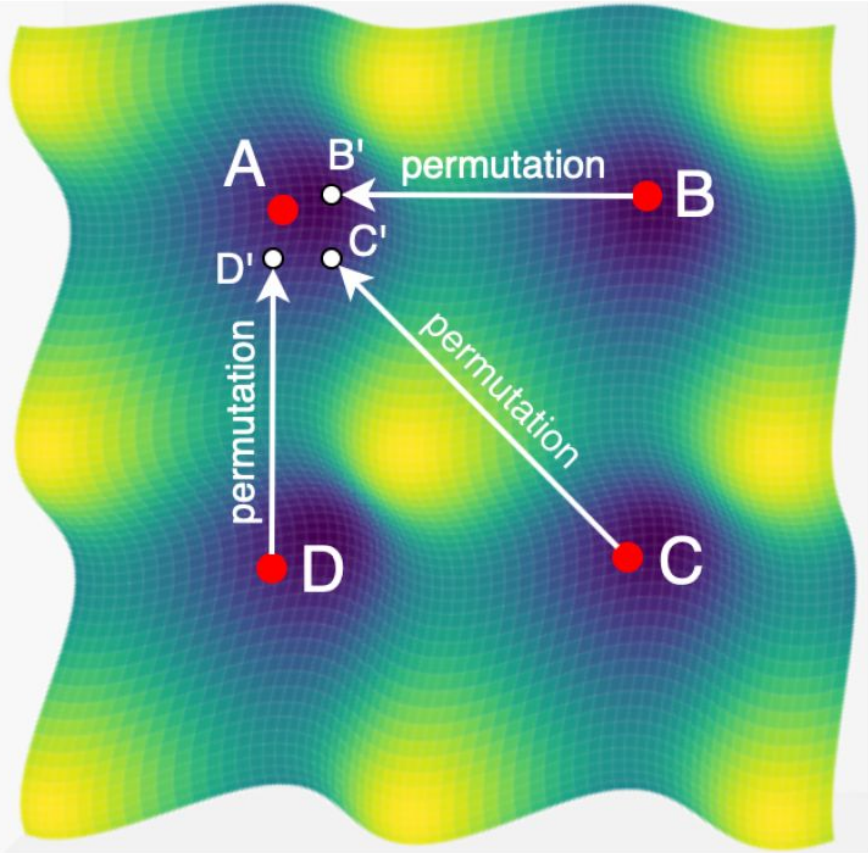
Let ***P*** be a permutation matrix. Then:

$$\boldsymbol{z}_{\ell+1} = \boldsymbol{P}^\top \boldsymbol{P} \boldsymbol{z}_{\ell+1} = \boldsymbol{P}^\top \boldsymbol{P} \sigma(\boldsymbol{W}_\ell \boldsymbol{z}_\ell + \boldsymbol{b}_\ell) = \boldsymbol{P}^\top \sigma(\boldsymbol{P}\boldsymbol{W}_\ell \boldsymbol{z}_\ell + \boldsymbol{P}\boldsymbol{b}_\ell)$$

Conjecture: SGD tends to converge to functionally equivalent models, up to permutation.

"Model Fusion via Optimal Transport". Singh and Jaggi, 2020

Equivalence up to permutation

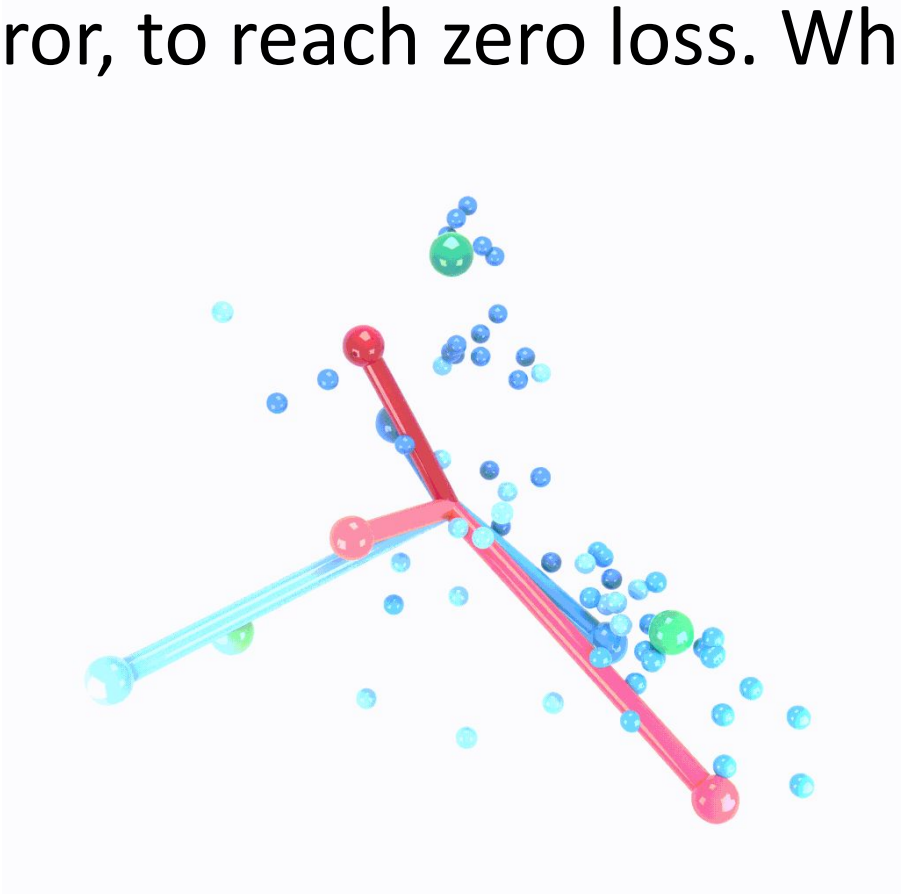


After applying the correct permutation, it is easier to observe linear mode connectivity.

"The Role of Permutation Invariance in Linear Mode Connectivity of Neural Networks". Entezari et al, 2022

Neural collapse

Experiment: Keep training a classifier after zero classification error, to reach zero loss. What happens?



The last-layer representations converge to **tight clusters**, one per class, **equally distributed on a hypersphere**.

The positions on the hypersphere matter **up to rotation.**

"Prevalence of neural collapse during the terminal phase of deep learning training". Papyan et al, 2020

Wrap-up

Key observation:

SGD tends to converge to functionally equivalent neural models

Mode connectivity shows entire spaces of equivalent models

Running theme: Isometries

Orthogonal matrices preserve lengths:

$$\|\mathbf{Q}\mathbf{x}\|_2^2 = \mathbf{x}^\top \mathbf{Q}^\top \mathbf{Q}\mathbf{x} = \mathbf{x}^\top \mathbf{I}\mathbf{x} = \mathbf{x}^\top \mathbf{x} = \|\mathbf{x}\|_2^2$$

Running theme: Isometries

Orthogonal matrices preserve lengths:

$$\|\mathbf{Q}\mathbf{x}\|_2^2 = \mathbf{x}^\top \mathbf{Q}^\top \mathbf{Q}\mathbf{x} = \mathbf{x}^\top \mathbf{I}\mathbf{x} = \mathbf{x}^\top \mathbf{x} = \|\mathbf{x}\|_2^2$$

Orthogonal matrices also preserve angles (i.e. inner products):

$$\langle \mathbf{Q}\mathbf{x}, \mathbf{Q}\mathbf{y} \rangle = \mathbf{x}^\top \mathbf{Q}^\top \mathbf{Q}\mathbf{y} = \mathbf{x}^\top \mathbf{I}\mathbf{y} = \mathbf{x}^\top \mathbf{y}$$

Running theme: Isometries

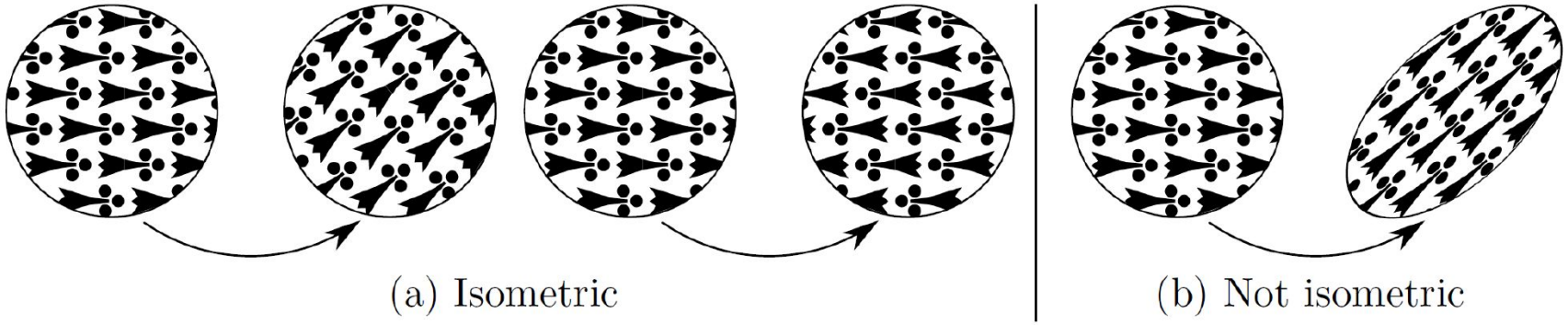
Orthogonal matrices preserve lengths:

$$\|\mathbf{Q}\mathbf{x}\|_2^2 = \mathbf{x}^\top \mathbf{Q}^\top \mathbf{Q}\mathbf{x} = \mathbf{x}^\top \mathbf{I}\mathbf{x} = \mathbf{x}^\top \mathbf{x} = \|\mathbf{x}\|_2^2$$

Orthogonal matrices also preserve angles (i.e. inner products):

$$\langle \mathbf{Q}\mathbf{x}, \mathbf{Q}\mathbf{y} \rangle = \mathbf{x}^\top \mathbf{Q}^\top \mathbf{Q}\mathbf{y} = \mathbf{x}^\top \mathbf{I}\mathbf{y} = \mathbf{x}^\top \mathbf{y}$$

By these properties, the map $\mathbf{x} \mapsto \mathbf{Q}\mathbf{x}$ is an isometry of $\mathbb{R}^n$.



(a) Isometric

(b) Not isometric

Running theme: Isometries

Models can be aligned via permutations

Euclidean isometries of the neurons

Neural collapse suggests equivalence up to rotation

Euclidean isometries of the representations

Running theme: Isometries

Models can be aligned via permutations

Euclidean isometries of the neurons

Neural collapse suggests equivalence up to rotation

Euclidean isometries of the representations

✗ only on training data

✗ only at zero error

In the next hour:

Functional equivalence via **non-Euclidean** isometries of the **representations**, on both **training and test** data

Relative representations

Small demo

- Visit the URL: https://tinyurl.com/dtuvae

- Familiarize with the notebook (a pre-trained model is also included)

- Reach the point where the latent codes are visualized as a colored point set in 2D

Relative representations

enable zero-shot latent space communication

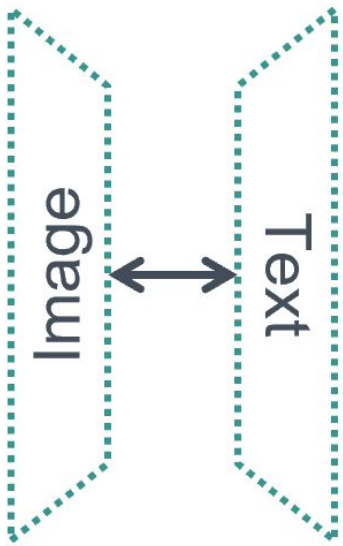
Luca Moschella Valentino Maiorca

Marco Fumero Antonio Norelli Francesco Locatello Emanuele Rodolà

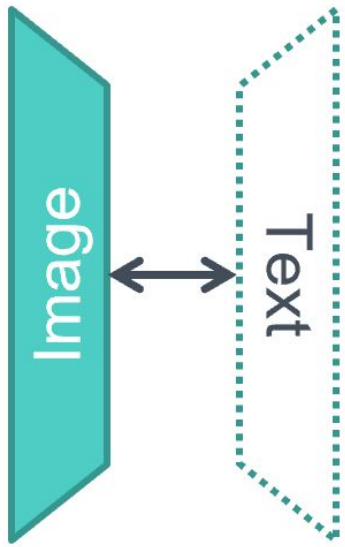
CLIP paradigm evolution

= Frozen pretrained model

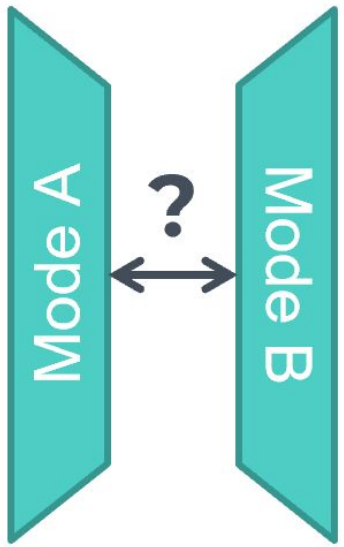
= trained from scratch



CLIP



LIT



ASIF

Open Lab

Open-ended demo

- Visit the URL: https://tinyurl.com/dturae
- Play with the notebook (a pre-trained model is also included)
- Try with your own data, or look at the next slide

Open-ended demo

- Change the similarity function
- See if relative representations also work at the other layers, not just the bottleneck
- See if relative representations work whenever linear mode connectivity works