

Popular NN Architectures

...and a little bit on the No Free Lunch theorems

Machine Learning and Data Mining, 2022

Artem Maevskiy

National Research University Higher School of Economics



LAMDA • HSE

November 24, 2021

Side note: No Free Lunch



A question from an IQ test

- ▶ Given the following pattern of numbers:

1, 8, 27, ?, 125, 216

what is the missing number?

- a) 36
- b) 45
- c) 46
- d) 64
- e) 99

A question from an IQ test

- ▶ Given the following pattern of numbers:

1, 8, 27, ?, 125, 216

what is the missing number?

- a) 36
- b) 45
- c) 46
- d) 64
- e) 99

	1	2	3	5	6
	1	8	27	125	216

$$X_{\text{test}} = (4,)$$

- ▶ Solution:

$$f(x) = \frac{1}{12}(35x^5 - 595x^4 + 3757x^3 - 10745x^2 + 13860x - 6300)$$

A question from an IQ test

- ▶ Given the following pattern of numbers:

1, 8, 27, ?, 125, 216

what is the missing number?

a) 36

b) 45

c) 46

d) 64

e) 99

	1	2	3	5	6
	1	8	27	125	216

$$X_{\text{test}} = (4,)$$

- ▶ Solution:

$$f(x) = \frac{1}{12}(35x^5 - 595x^4 + 3757x^3 - 10745x^2 + 13860x - 6300)$$

- ▶ Answer: $f(4) = 99$

A question from an IQ test

- ▶ Given the following pattern of numbers:

1, 8, 27, ?, 125, 216

what is the missing number?

a) 36

b) 45

c) 46

d) 64

e) 99

$$f(x) = x^3$$

	1	2	3	5	6
	1	8	27	125	216

$X_{\text{test}} = (4,)$

- ▶ Solution:

$$f(x) = \frac{1}{12}(35x^5 - 595x^4 + 3757x^3 - 10745x^2 + 13860x - 6300)$$

- ▶ Answer: $f(4) = 99$

Discussion

- ▶ Why $f(x) = x^3$ is better than

$$f(x) = \frac{1}{12}(35x^5 - 595x^4 + 3757x^3 - 10745x^2 + 13860x - 6300)?$$

Discussion

- ▶ Why $f(x) = x^3$ is better than

$$f(x) = \frac{1}{12}(35x^5 - 595x^4 + 3757x^3 - 10745x^2 + 13860x - 6300)?$$

- ▶ First one is more suitable **in the context of an IQ test**

Discussion

- ▶ Why $f(x) = x^3$ is better than

$$f(x) = \frac{1}{12}(35x^5 - 595x^4 + 3757x^3 - 10745x^2 + 13860x - 6300)?$$

- ▶ First one is more suitable **in the context of an IQ test**



Prior knowledge

- ▶ No free lunch theorem (roughly speaking): without prior knowledge all solutions are equally good (or bad)

No Free Lunch Theorem

- ▶ True (deterministic) mapping:

$$f: y = f(x), \quad x \in \mathcal{X}, \quad y \in \mathcal{Y}$$

No Free Lunch Theorem

- ▶ True (deterministic) mapping:
- ▶ Dataset:

$$f: y = f(x), \quad x \in \mathcal{X}, \quad y \in \mathcal{Y}$$

$$D = (X, y) \subseteq \mathcal{X} \times \mathcal{Y}$$

No Free Lunch Theorem

- ▶ True (deterministic) mapping:
- ▶ Dataset:
- ▶ ML algorithm:

$$f: y = f(x), \quad x \in \mathcal{X}, \quad y \in \mathcal{Y}$$

$$D = (X, y) \subseteq \mathcal{X} \times \mathcal{Y}$$

$$\mathcal{A}(D_{\text{train}}) = \hat{f}, \quad \hat{f}: \mathcal{X} \rightarrow \mathcal{Y}$$

No Free Lunch Theorem

- ▶ True (deterministic) mapping:
- ▶ Dataset:
- ▶ ML algorithm:
- ▶ Binary features and targets:

$$f: y = f(x), \quad x \in \mathcal{X}, \quad y \in \mathcal{Y}$$

$$D = (X, y) \subseteq \mathcal{X} \times \mathcal{Y}$$

$$\mathcal{A}(D_{\text{train}}) = \hat{f}, \quad \hat{f}: \mathcal{X} \rightarrow \mathcal{Y}$$

$$\mathcal{X} = \{0, 1\}^m, \quad \mathcal{Y} = \{0, 1\}$$

No Free Lunch Theorem

- ▶ True (deterministic) mapping:
- ▶ Dataset:
- ▶ ML algorithm:
- ▶ Binary features and targets:
- ▶ Training dataset of fixed size:

$$f: y = f(x), \quad x \in \mathcal{X}, \quad y \in \mathcal{Y}$$

$$D = (X, y) \subseteq \mathcal{X} \times \mathcal{Y}$$

$$\mathcal{A}(D_{\text{train}}) = \hat{f}, \quad \hat{f}: \mathcal{X} \rightarrow \mathcal{Y}$$

$$\mathcal{X} = \{0, 1\}^m, \quad \mathcal{Y} = \{0, 1\}$$

$$|D_{\text{train}}| = n$$

No Free Lunch Theorem

- ▶ True (deterministic) mapping:
- ▶ Dataset:
- ▶ ML algorithm:
- ▶ Binary features and targets:
- ▶ Training dataset of fixed size:
- ▶ Test set:

$$f: y = f(x), \quad x \in \mathcal{X}, \quad y \in \mathcal{Y}$$

$$D = (X, y) \subseteq \mathcal{X} \times \mathcal{Y}$$

$$\mathcal{A}(D_{\text{train}}) = \hat{f}, \quad \hat{f}: \mathcal{X} \rightarrow \mathcal{Y}$$

$$\mathcal{X} = \{0, 1\}^m, \quad \mathcal{Y} = \{0, 1\}$$

$$|D_{\text{train}}| = n$$

$$D_{\text{test}}: X_{\text{test}} = \mathcal{X} \setminus X_{\text{train}}$$

No Free Lunch Theorem

- ▶ True (deterministic) mapping:
- ▶ Dataset:
- ▶ ML algorithm:
- ▶ Binary features and targets:
- ▶ Training dataset of fixed size:
- ▶ Test set:
- ▶ Generalization performance:

$$f: y = f(x), \quad x \in \mathcal{X}, \quad y \in \mathcal{Y}$$

$$D = (X, y) \subseteq \mathcal{X} \times \mathcal{Y}$$

$$\mathcal{A}(D_{\text{train}}) = \hat{f}, \quad \hat{f}: \mathcal{X} \rightarrow \mathcal{Y}$$

$$\mathcal{X} = \{0, 1\}^m, \quad \mathcal{Y} = \{0, 1\}$$

$$|D_{\text{train}}| = n$$

$$D_{\text{test}}: X_{\text{test}} = \mathcal{X} \setminus X_{\text{train}}$$

$$\text{gP}(\mathcal{A}, D_{\text{train}}) = \frac{1}{|D_{\text{test}}|} \sum_{x, y \in D_{\text{test}}} \mathbb{I}[A(D_{\text{train}})(x) = y] - \frac{1}{2}$$

No Free Lunch Theorem

- ▶ True (deterministic) mapping:
- ▶ Dataset:
- ▶ ML algorithm:
- ▶ Binary features and targets:
- ▶ Training dataset of fixed size:
- ▶ Test set:
- ▶ Generalization performance:
- ▶ Theorem statement:

$$f: y = f(x), \quad x \in \mathcal{X}, \quad y \in \mathcal{Y}$$

$$D = (X, y) \subseteq \mathcal{X} \times \mathcal{Y}$$

$$\mathcal{A}(D_{\text{train}}) = \hat{f}, \quad \hat{f}: \mathcal{X} \rightarrow \mathcal{Y}$$

$$\mathcal{X} = \{0, 1\}^m, \quad \mathcal{Y} = \{0, 1\}$$

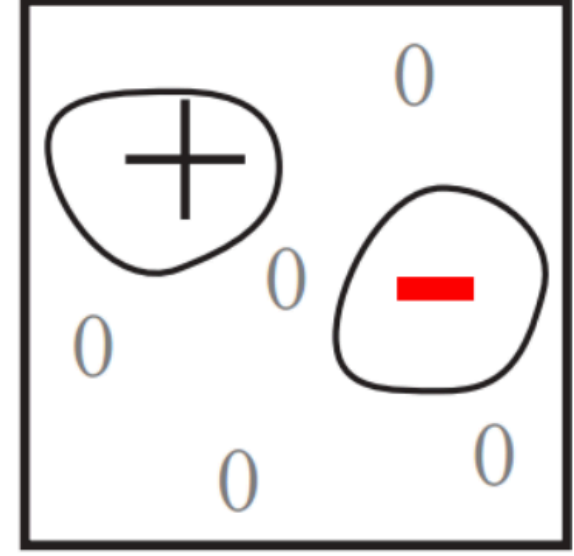
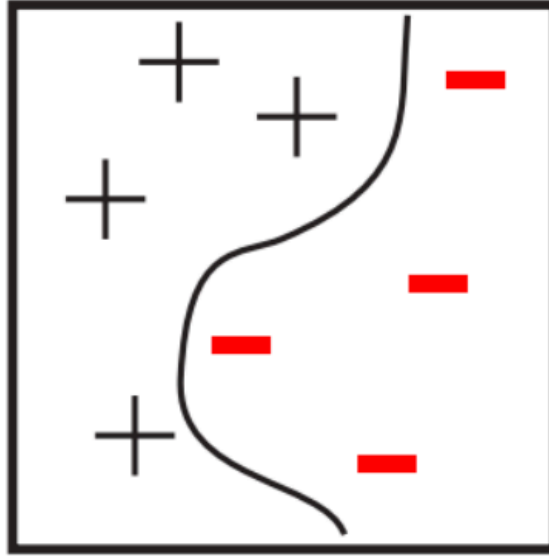
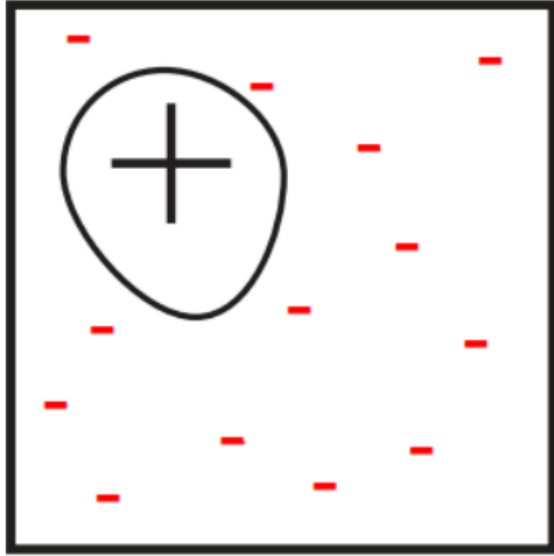
$$|D_{\text{train}}| = n$$

$$D_{\text{test}}: X_{\text{test}} = \mathcal{X} \setminus X_{\text{train}}$$

$$\text{gP}(\mathcal{A}, D_{\text{train}}) = \frac{1}{|D_{\text{test}}|} \sum_{x, y \in D_{\text{test}}} \mathbb{I}[A(D_{\text{train}})(x) = y] - \frac{1}{2}$$

$$\sum_{f, D_{\text{train}}} \text{gP}(\mathcal{A}, D_{\text{train}}) = 0$$

In the problem space



- Possible performance of learning algorithms:
 - Worse than average (—)
 - Better than average (+)

Back to the IQ test problem

- ▶ Without our prior knowledge about IQ tests any solution is equally likely
 - We know that the authors of the test don't expect people to fit 5th degree polynomials in their head

Back to the IQ test problem

- ▶ Without our prior knowledge about IQ tests any solution is equally likely
 - We know that the authors of the test don't expect people to fit 5th degree polynomials in their head
- ▶ To solve the test one must think like the authors of the test

Back to the IQ test problem

- ▶ Without our prior knowledge about IQ tests any solution is equally likely
 - We know that the authors of the test don't expect people to fit 5th degree polynomials in their head
- ▶ To solve the test one must think like the authors of the test
- ▶ To solve a real world ML problem one must think like... the real world.

NFL theorem: critique

- ▶ Not all problems are equally likely (in the real world):
 - continuity;
 - human bias: e.g. feature preselection
 - prior knowledge of the problem at hand
- ▶ E.g. memorization + interpolation becomes an effective strategy for continuous data
- ▶ The following still holds though:

To improve the performance on one class of problems
one must sacrifice the performance on others.

The role of data scientist

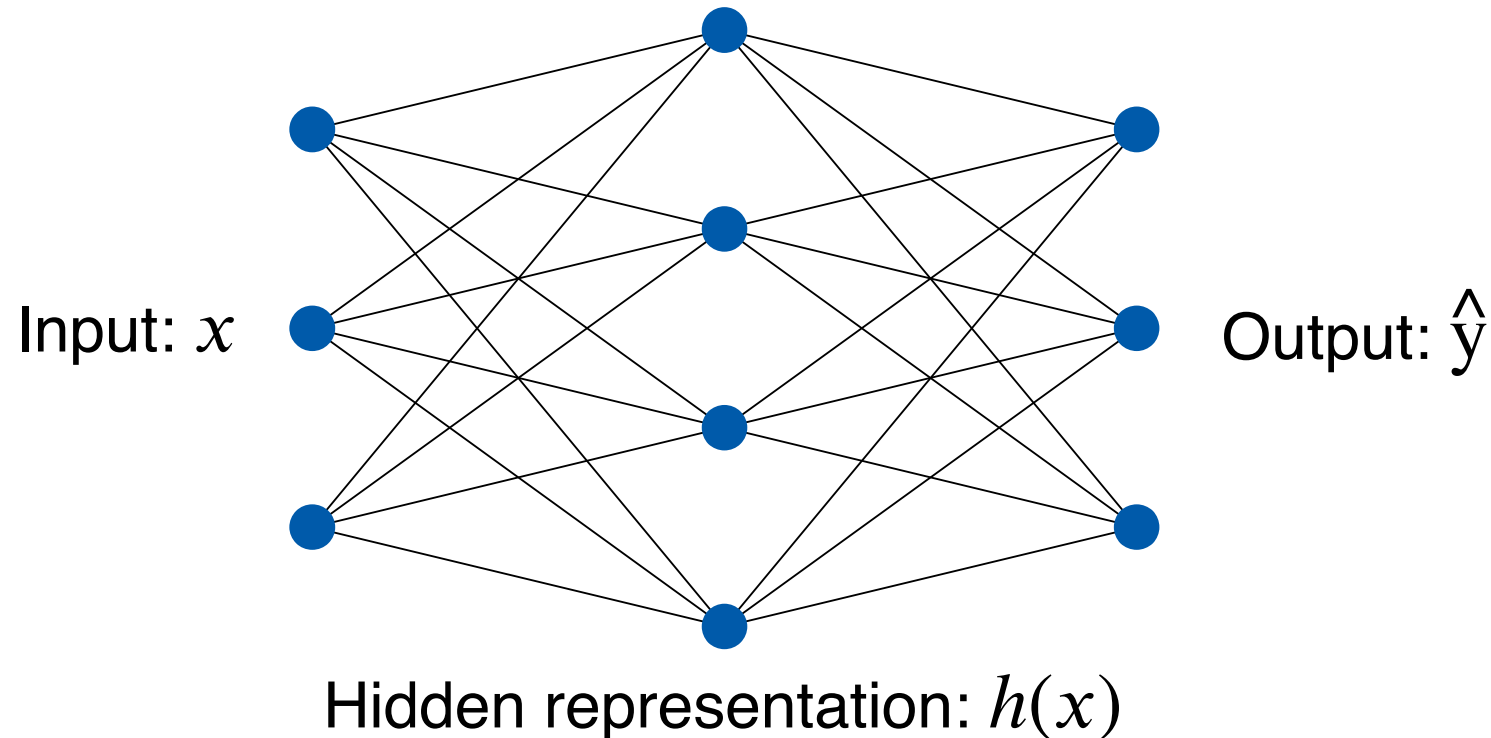
- ▶ Identify the suitable learning algorithm using the prior knowledge:
 - continuity;
 - structure of the data;
 - quality of the data;
 - domain knowledge;
 - size of the dataset;
 - common sense;
 - etc.
- ▶ In the context of deep learning: **find a suitable architecture**

NN architecture: simple examples



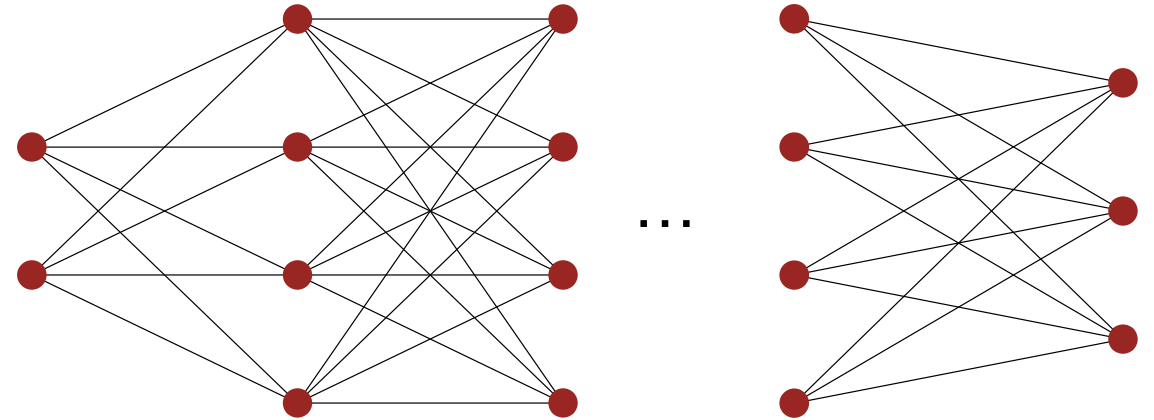
Single hidden layer fully-connected network

- ▶ Universal approximator
- ▶ May require infeasibly large hidden representation for more complex dependencies



Deep fully-connected network

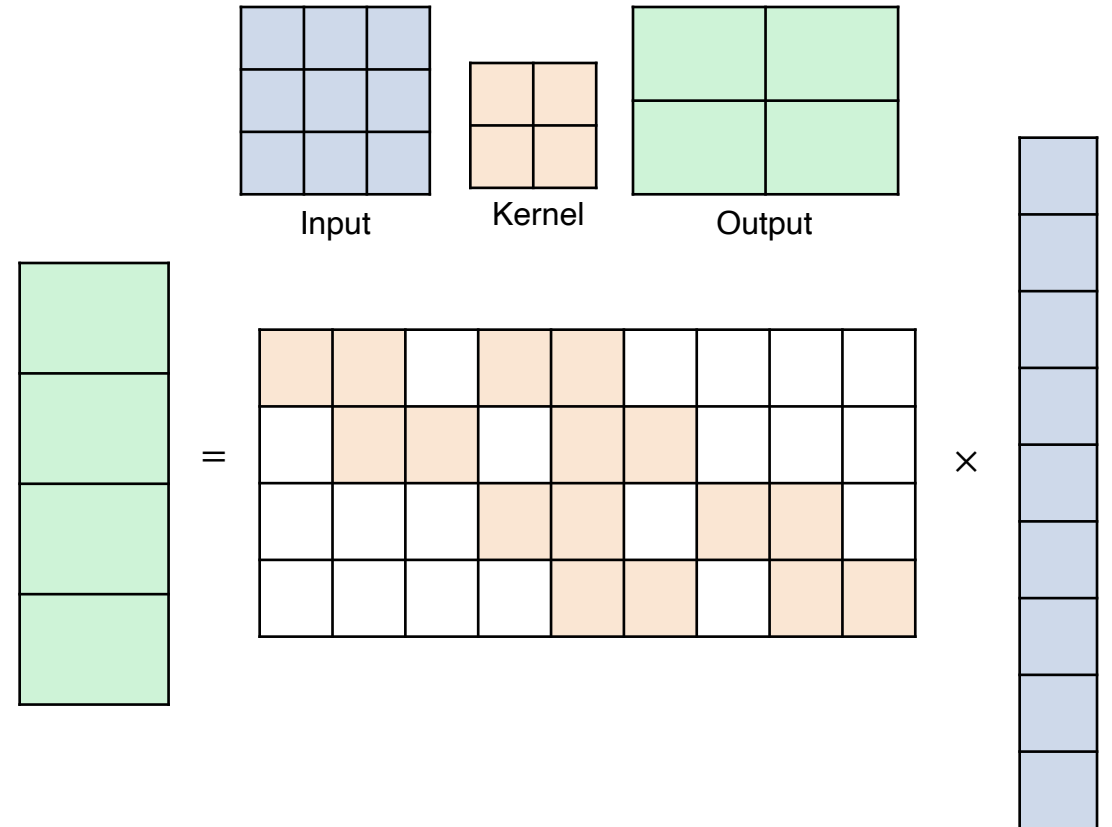
- ▶ May use smaller representations
- ▶ Suitable for many real-world problems
- ▶ Harder to optimize
 - Typically becomes quite challenging for 10+ layers
- ▶ **An initial point for other architectures.**



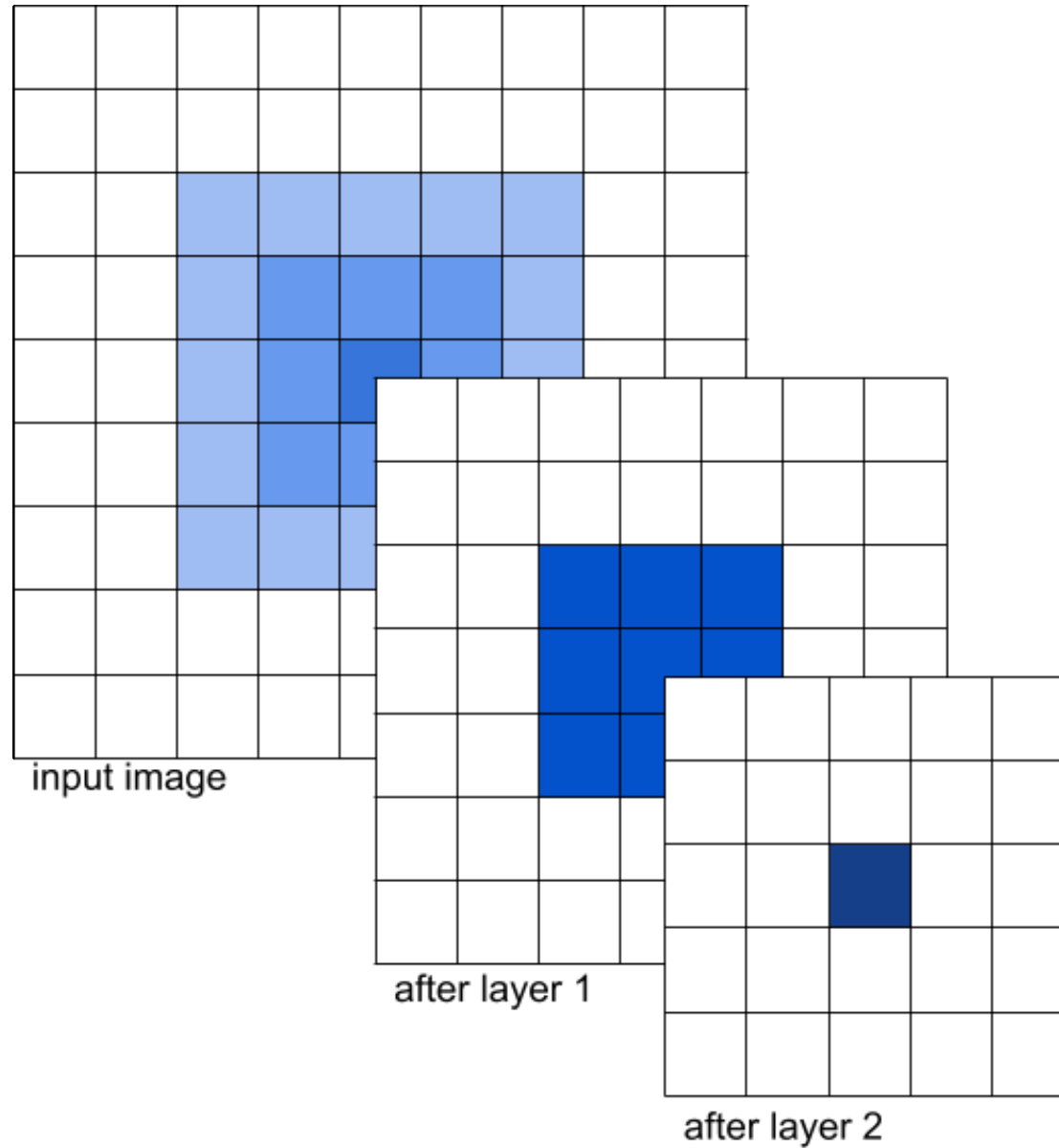
Convolution

- ▶ Spatial/temporal structure of the data
- ▶ Can be described in terms of a fully connected layer
- ▶ Uses much less parameters
 - Re-uses weights in a sliding window

2D convolution as a matrix multiplication

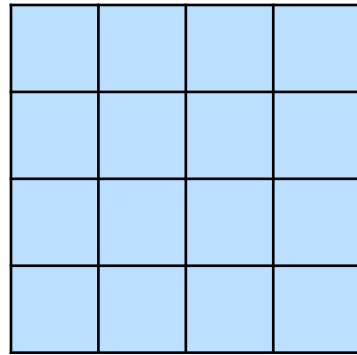


Receptive field

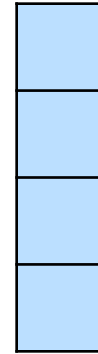


Combining simpler convolutions

- ▶ Replacing a $n \times n$ convolution with two subsequent convolutions with kernels $n \times 1$ and $1 \times n$:



$n \times n$ convolution



$n \times 1$ convolution

+



$1 \times n$ convolution

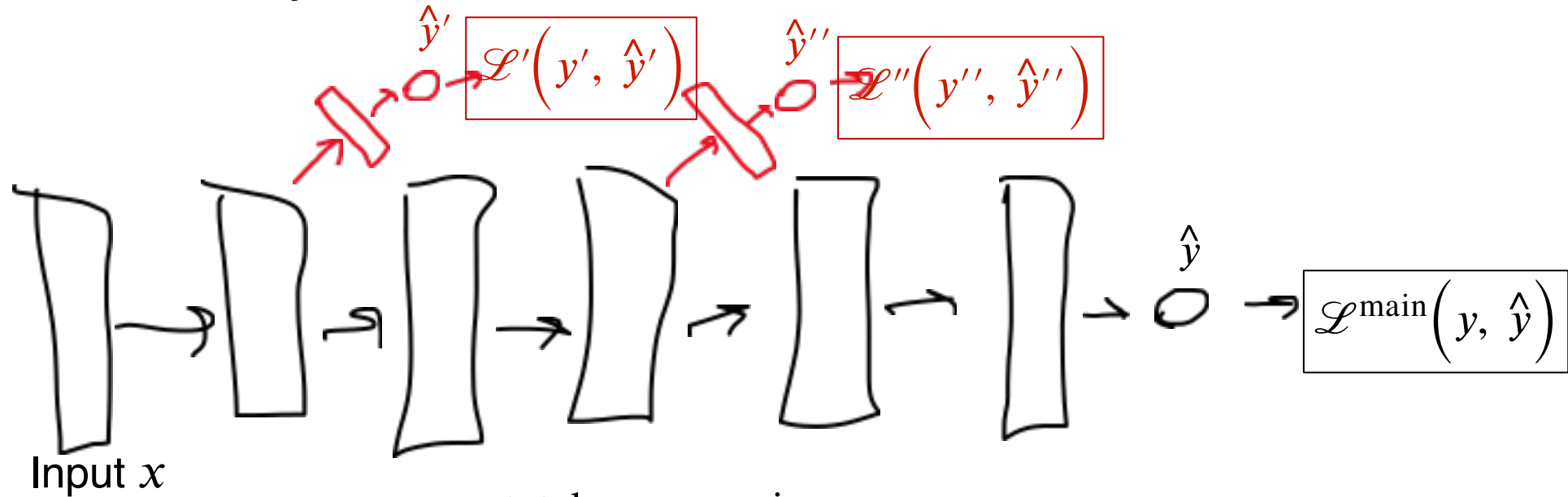
- ▶ Same receptive field, fewer parameters

Deeply supervised architectures



Main idea

- ▶ Hard to optimize deep network due to vanishing gradients
- ▶ Encourage good gradients by introducing additional “heads” from the intermediate layers



$$\mathcal{L}^{\text{total}} = \mathcal{L}^{\text{main}} + \alpha' \mathcal{L}' + \alpha'' \mathcal{L}''$$

- ▶ Typically, does not lead to good hidden representations
 - Makes sense to decay α' and α'' to 0 during training

Auxiliary tasks

- ▶ Introduce multiheaded models to solve similar tasks
 - E.g. when detecting images of people in glasses
 - add a head to predict the color of their hair
- ▶ The multiple heads may share common features and lead to a better result for the main task

Transfer learning and fine-tuning

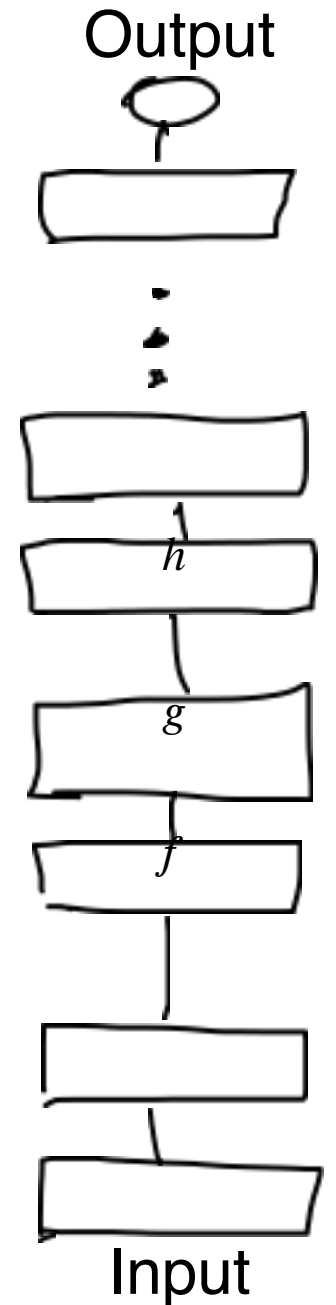
- ▶ Assume your problem provides a dataset too little to train a full-scale model from scratch
- ▶ Yet there exists a similar or a more general large dataset with a trained model that solves it well enough
- ▶ Transfer learning:
 - take first N layers of the trained model and freeze them
 - attach a new untrained head
 - train the whole thing (with only the head parameters being trainable)
- ▶ Fine-tuning:
 - After having trained the head, release the frozen weights and train the whole model further (with very small learning rate)

Residual connections



Residual connections

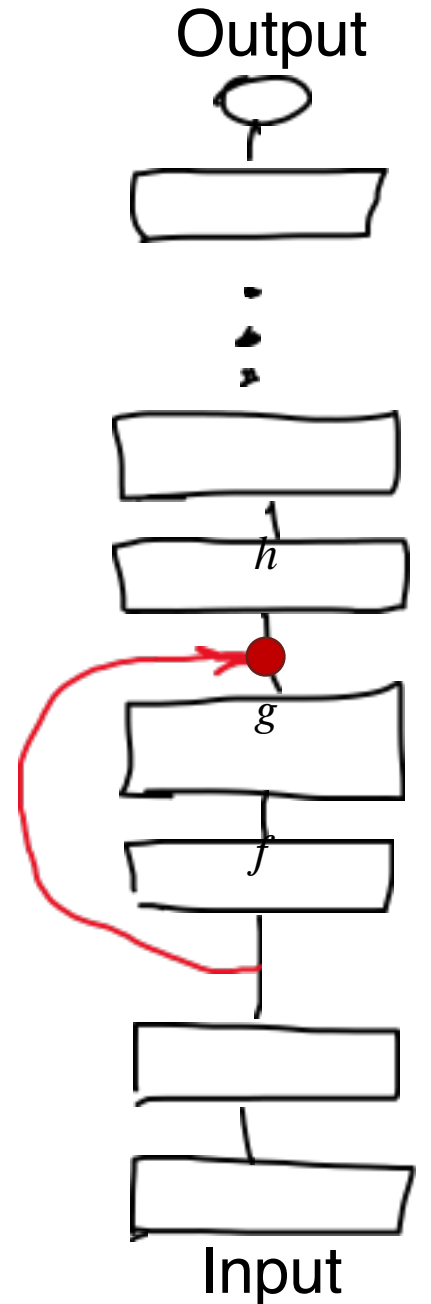
- ▶ Vanishing gradients: the derivative wrt early layer's parameters accumulates the product of derivatives for further layers (chain rule)



Residual connections

- ▶ Vanishing gradients: the derivative wrt early layer's parameters accumulates the product of derivatives for further layers (chain rule)
- ▶ Solution: add residual connections (skip-connections)

$$h\left(g\left(f(\tilde{x})\right)\right) \rightarrow h\left(\tilde{x} + g\left(f(\tilde{x})\right)\right)$$

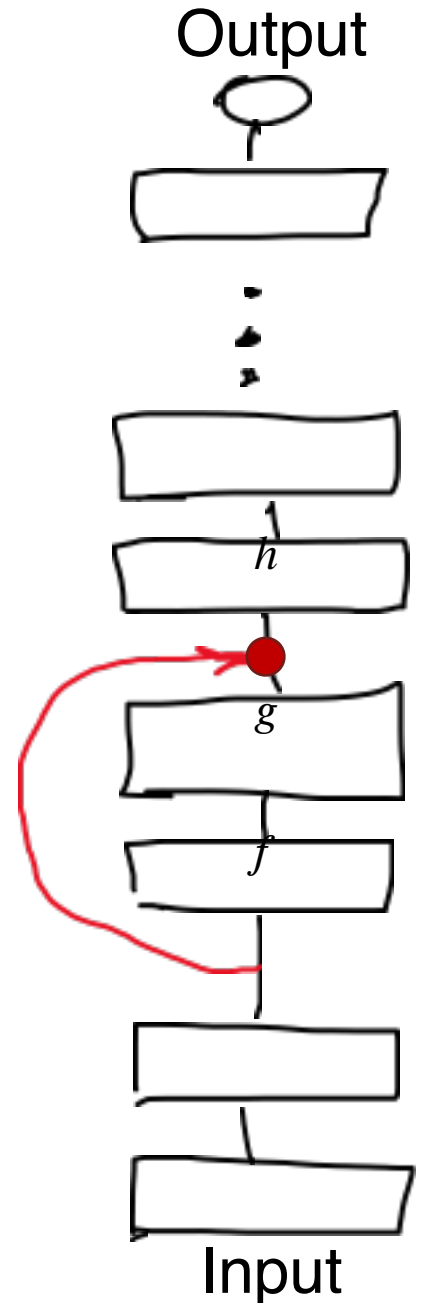


Residual connections

- ▶ Vanishing gradients: the derivative wrt early layer's parameters accumulates the product of derivatives for further layers (chain rule)
- ▶ Solution: add residual connections (skip-connections)

$$h\left(g\left(f(\tilde{x})\right)\right) \rightarrow h\left(\tilde{x} + g\left(f(\tilde{x})\right)\right)$$

- ▶ Allows extremely deep networks to be trained (like, 1000 layers!)

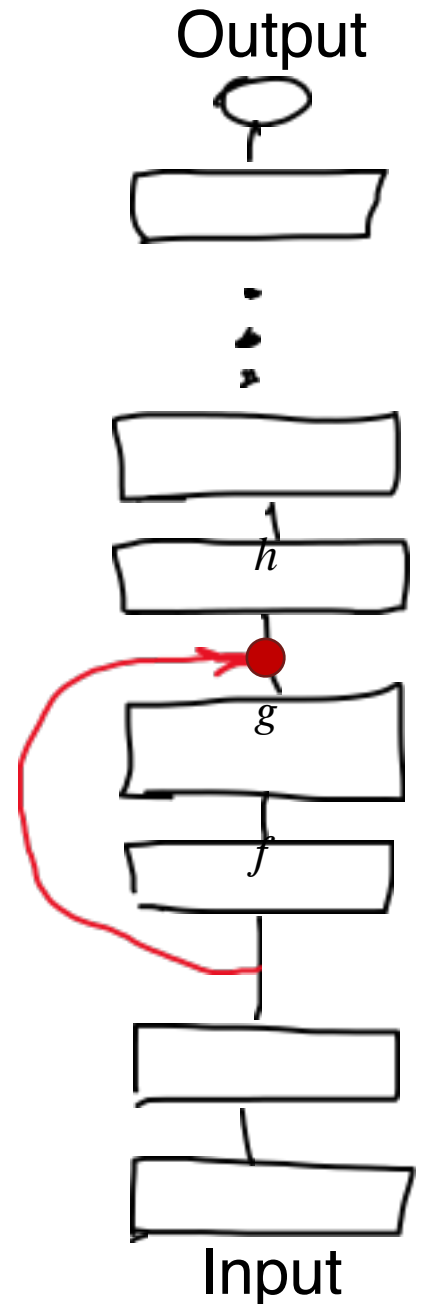


Residual connections

- ▶ Vanishing gradients: the derivative wrt early layer's parameters accumulates the product of derivatives for further layers (chain rule)
- ▶ Solution: add residual connections (skip-connections)

$$h\left(g\left(f(\tilde{x})\right)\right) \rightarrow h\left(\tilde{x} + g\left(f(\tilde{x})\right)\right)$$

- ▶ Allows extremely deep networks to be trained (like, 1000 layers!)
- ▶ Note: is this always possible to do?

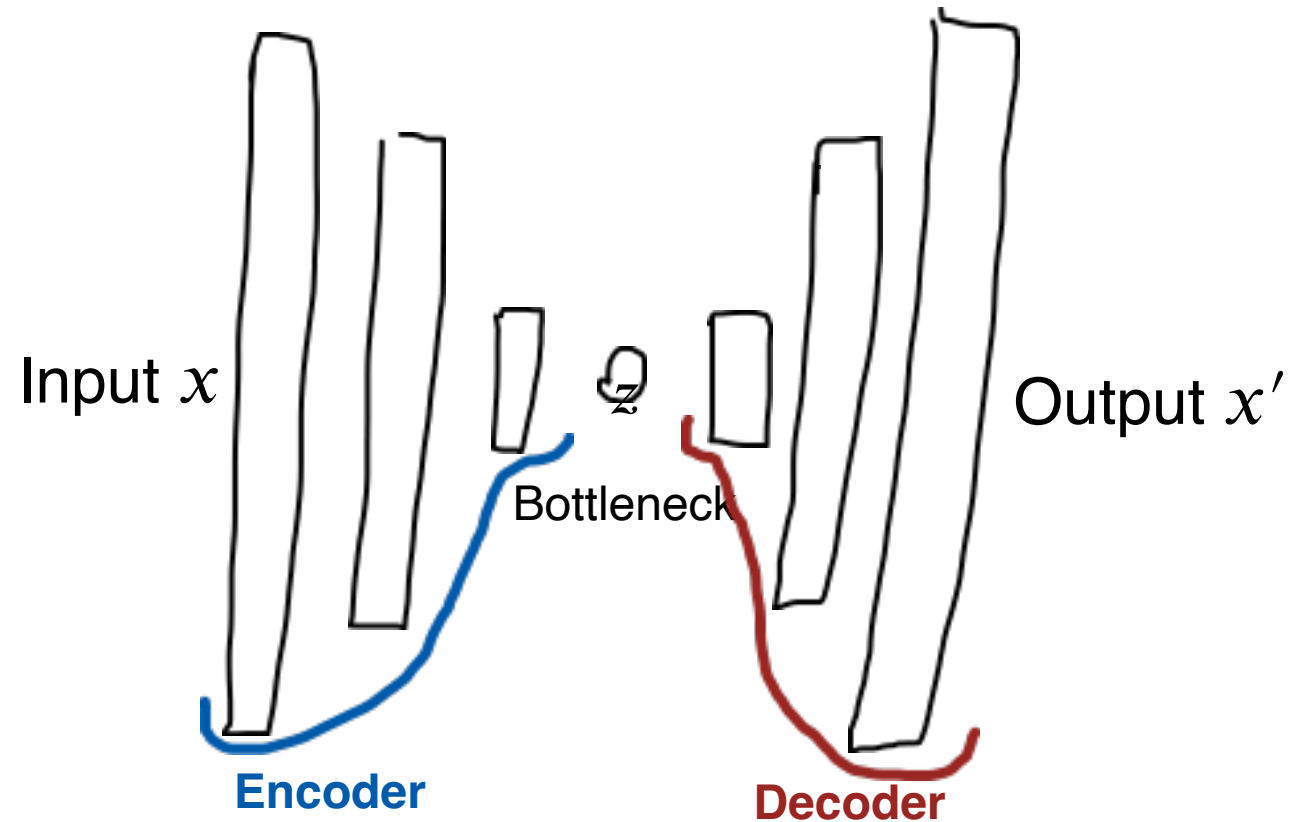


Autoencoders



Autoencoders

- ▶ A network that reconstructs its own input
- ▶ Has a 'bottleneck' representation
- ▶ Common use cases:
 - Dimensionality reduction
 - Anomaly detection
 - Denoising
 - Pretraining
 - Auxiliary loss, regularization





Semi-supervised learning with AE

- ▶ Note that training an AE doesn't require any targets!
- ▶ Imagine a situation having a small labelled dataset and large unlabelled
- ▶ Semi-supervised approach:
 - train an AE on the unlabelled dataset
 - then train a classification head on top of a hidden representation from the AE
 - may also train both models simultaneously

U-net

- Problems involving image to image transformation or image segmentation

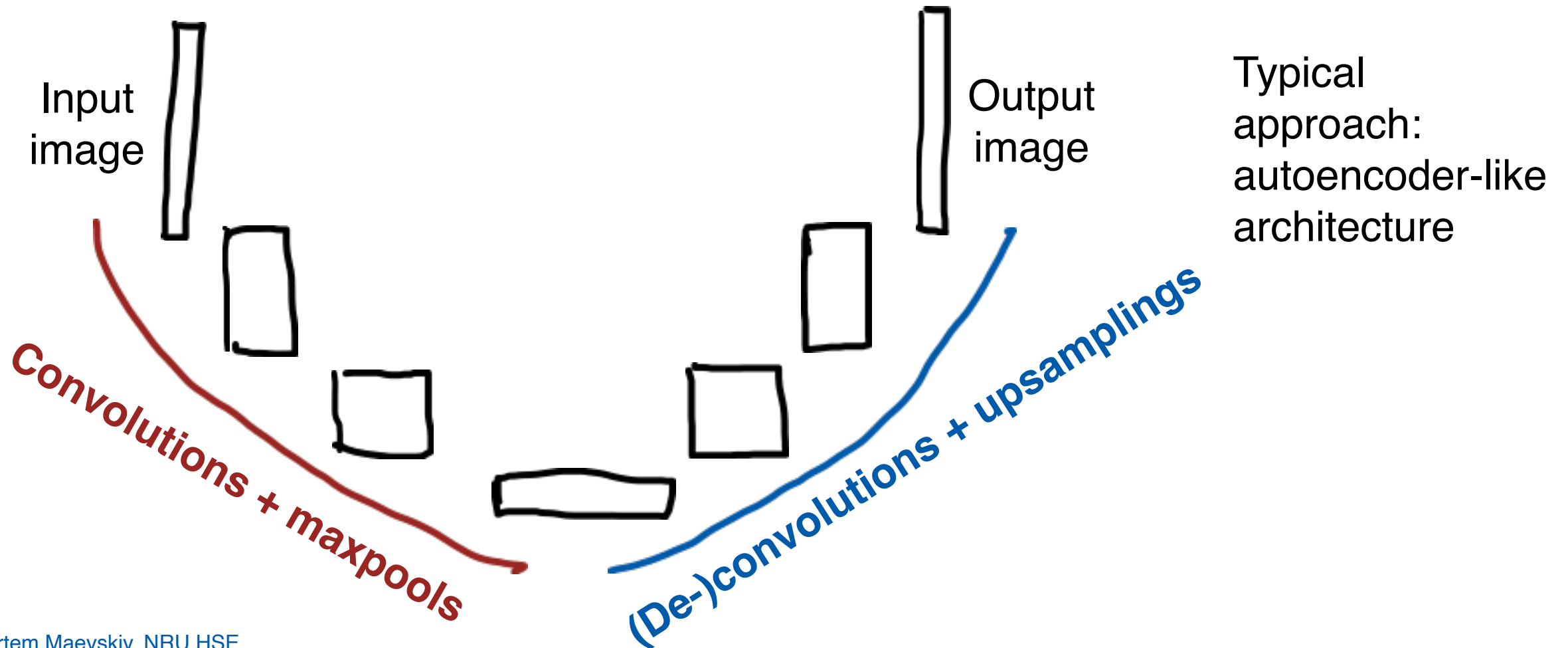
Input
image



Output
image

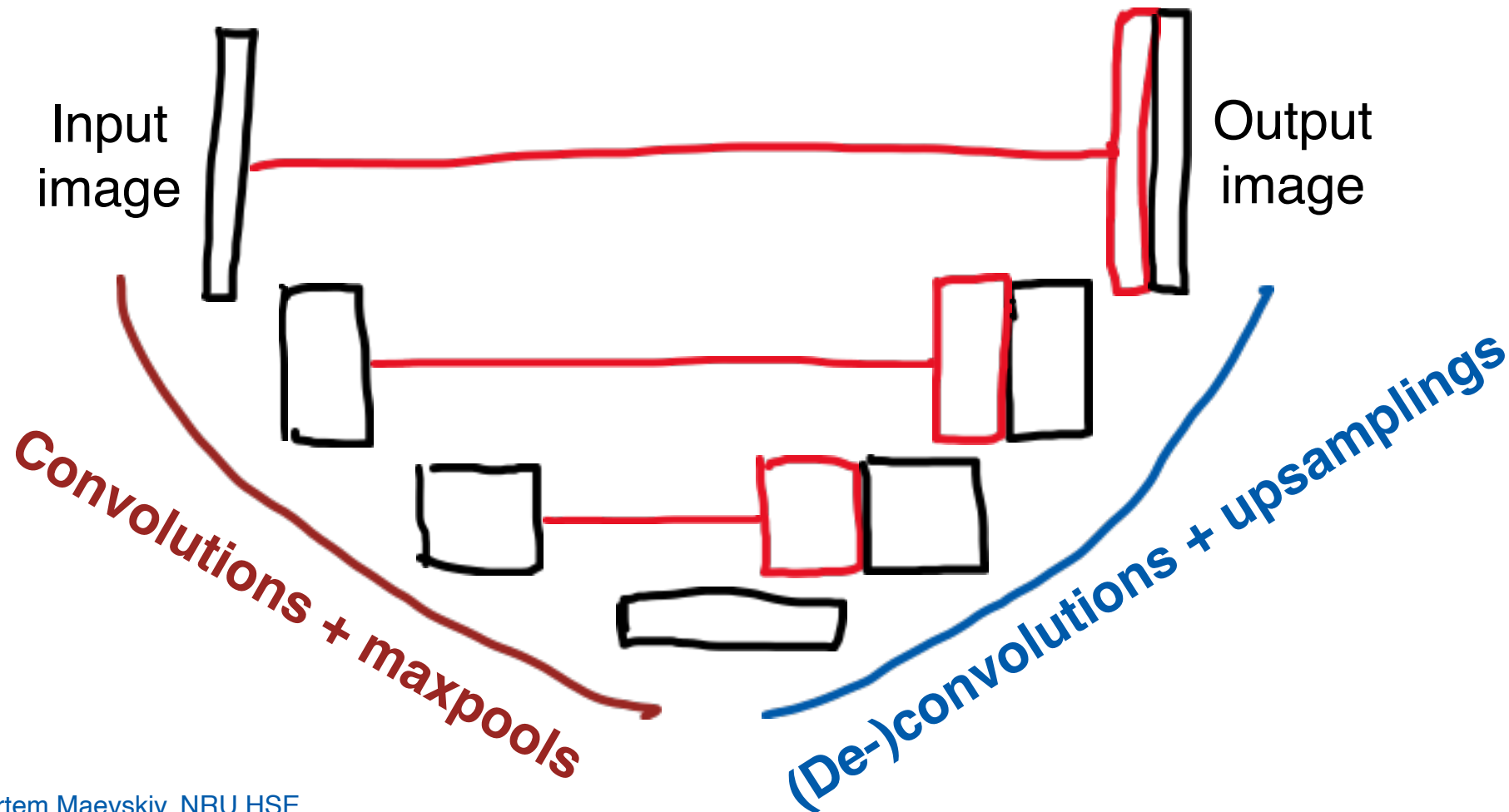
U-net

- Problems involving image to image transformation or image segmentation



U-net

- Problems involving image to image transformation or image segmentation



Typical approach: autoencoder-like architecture

Additional detail: skip-connections (typically concatenated)

- this combines low- and high-level information in the “decoder” branch

Summary

- ▶ According to the no-free-lunch theorem, all learning algorithms are equally useless
- ▶ It's the goal of the data scientist to make them useful leveraging the prior knowledge about the problem
- ▶ In the context of deep learning this typically involves finding (inventing) a suitable architecture
- ▶ As you can see, neural networks are extremely flexible
 - Finding a good architecture may require some creativity
- ▶ The list of shown architectures is by no means comprehensive
 - Countless other architectures and tricks

Thank you!



amaevskij@hse.ru



SiLiKhon

Artem Maevskiy