Machine Learning in Weather and Climate

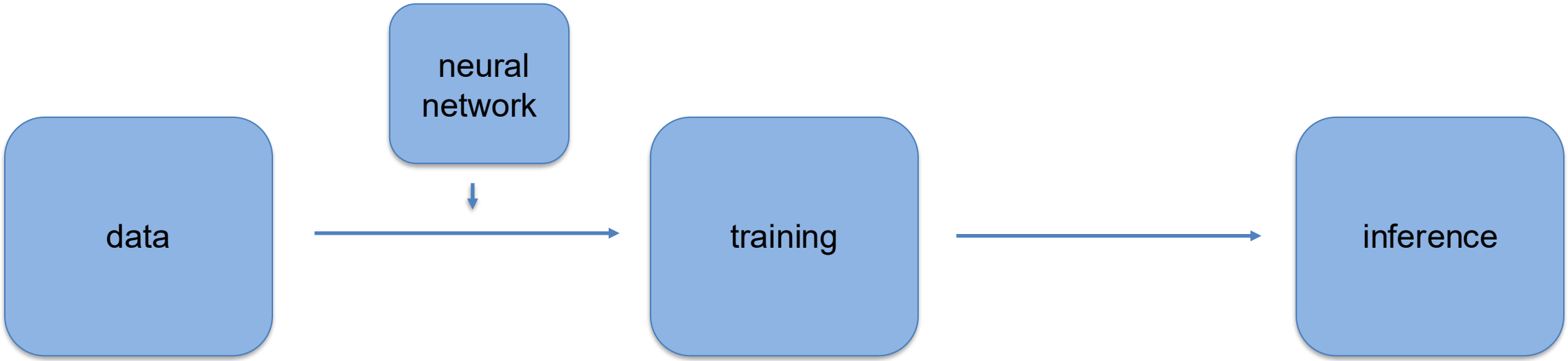
Machine Learning 1-0-1



Christian Lessig
christian.lessig@ecmwf.int

Overview

Classical machine learning pipeline:



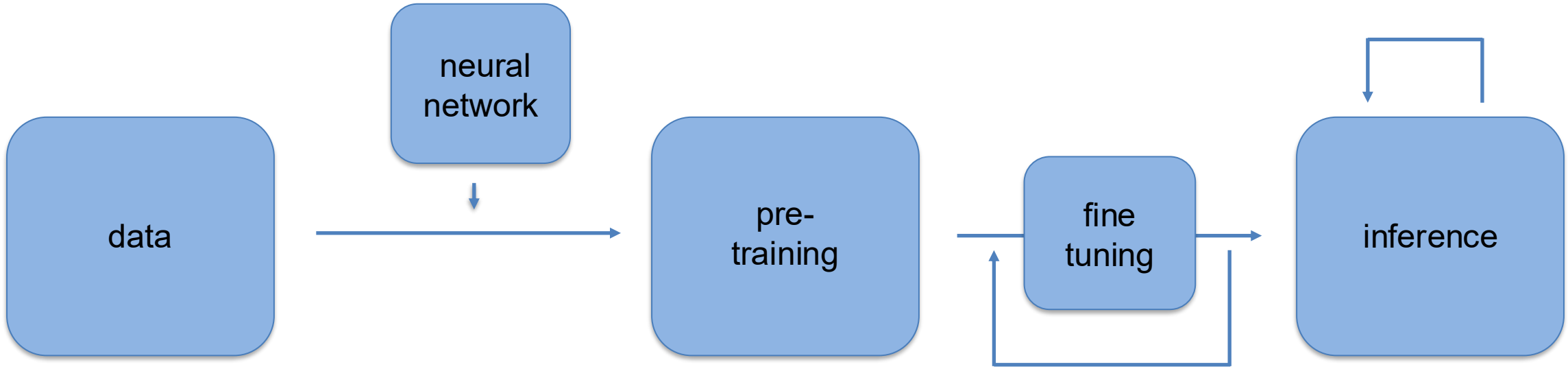
the stuff to model (usually with ambiguity and large variability)

Fit neural network to data

Apply neural network to new data

Overview

Modern machine learning pipeline:



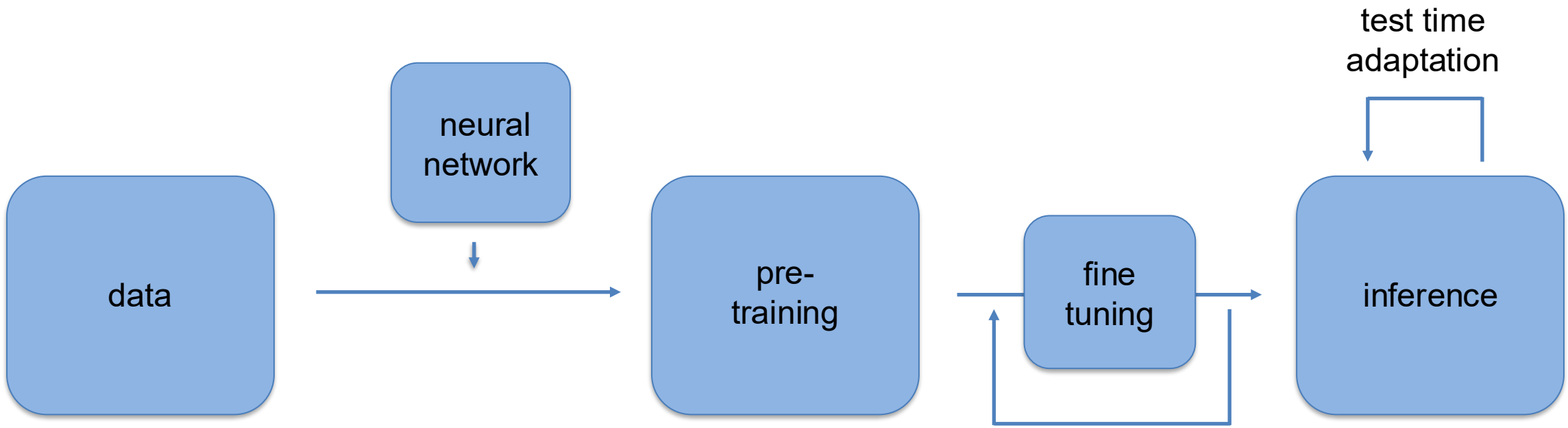
the stuff to model (usually with ambiguity and large variability)

Fit neural network to data

Apply neural network to new data

Overview

Modern machine learning pipeline:



the stuff to model
(usually with ambiguity
and large variability)

Fit neural
network to data

calibrate net
for application

Apply neural
network to new data

Neural networks

Overall structure:

$$\mathcal{N} : \mathbb{R}^N \to \mathbb{R}^M$$

Neural networks

Overall structure:

nonlinear mapping $\mathcal{N} : \mathbb{R}^N \rightarrow \mathbb{R}^M$

Neural networks

Overall structure:

nonlinear mapping $\mathcal{N} : \mathbb{R}^N \rightarrow \mathbb{R}^M$

data item

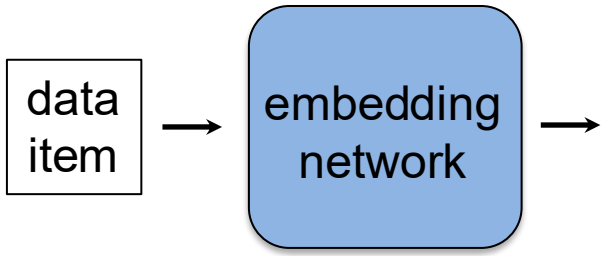
$\mathbb{R}^N$

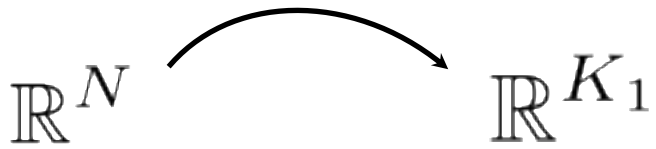
vectorized input data

Neural networks

Overall structure:

nonlinear mapping $\mathcal{N} : \mathbb{R}^N \rightarrow \mathbb{R}^M$





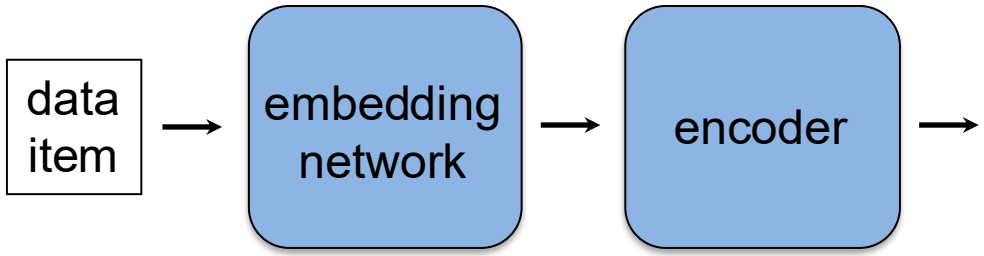
vectorized input data

map input to neural net

Neural networks

Overall structure:

nonlinear mapping $\mathcal{N} : \mathbb{R}^N \rightarrow \mathbb{R}^M$





vectorized input data

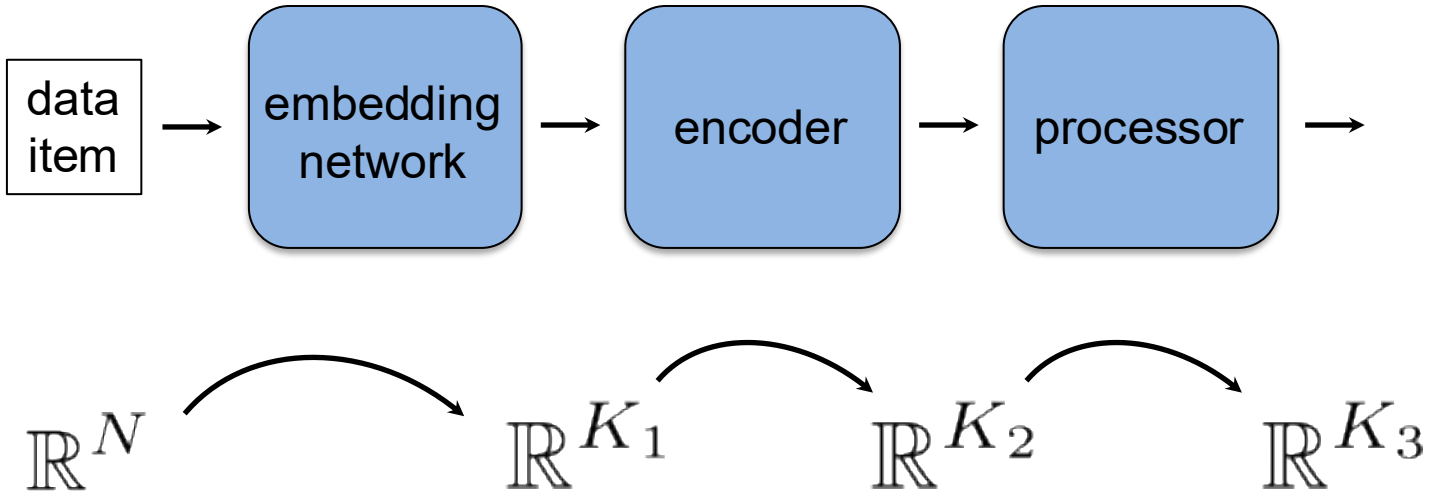
map input to neural net

generate salient data representation

Neural networks

Overall structure:

nonlinear mapping $\mathcal{N} : \mathbb{R}^N \rightarrow \mathbb{R}^M$



vectorized input data

map input to neural net

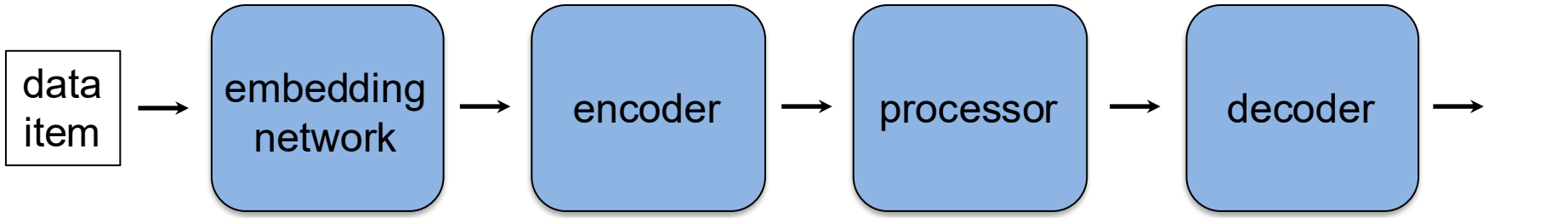
generate salient data representation

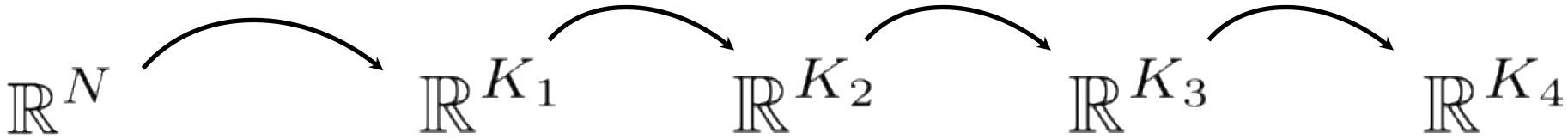
process in salient data representation

Neural networks

Overall structure:

nonlinear mapping $\mathcal{N} : \mathbb{R}^N \rightarrow \mathbb{R}^M$





vectorized input data

map input to neural net

generate salient data representation

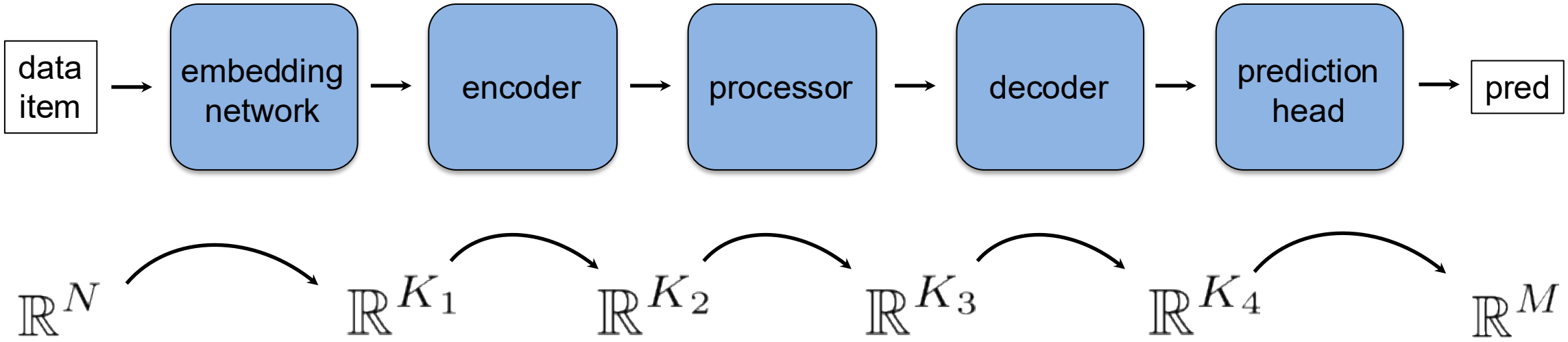
process in salient data representation

decode from neural net

Neural networks

Overall structure:

nonlinear mapping $\mathcal{N} : \mathbb{R}^N \rightarrow \mathbb{R}^M$



vectorized input data

map input to neural net

generate salient data representation

process in salient data representation

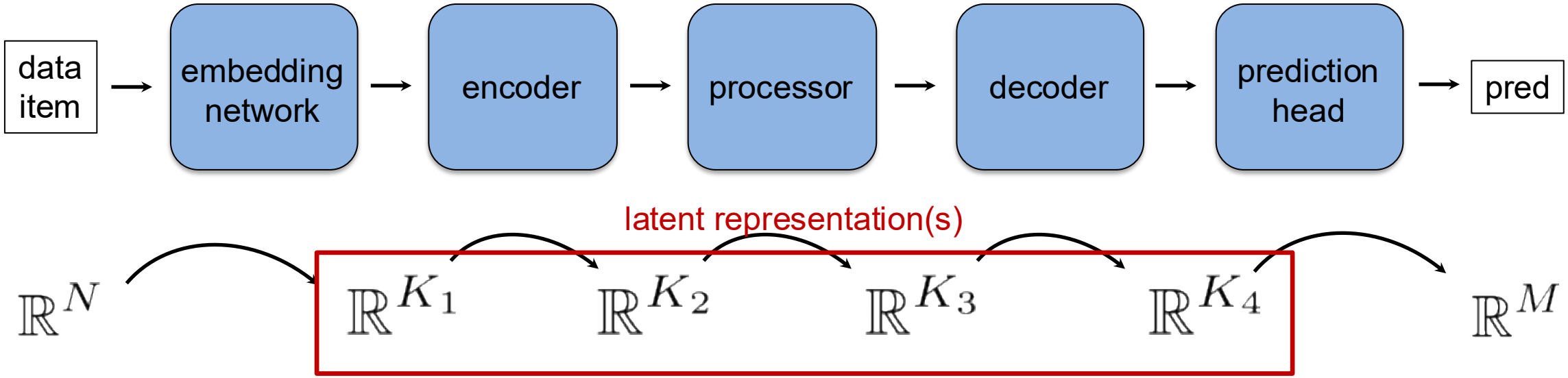
decode from neural net

map to output space

Neural networks

Overall structure:

nonlinear mapping $\mathcal{N} : \mathbb{R}^N \rightarrow \mathbb{R}^M$



vectorized input data

map input to neural net

generate salient data representation

process in salient data representation

decode from neural net

map to output space

Neural networks

Neural network blocks:

- Consists of simple building blocks, known as layers
- Simplest layer: linear, i.e. matrix

$$L : \mathbb{R}^n \to \mathbb{R}^m$$

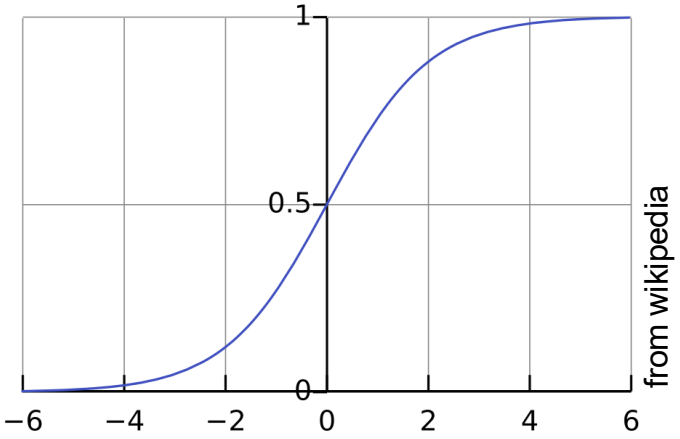
Neural networks

Neural network blocks:

- Consists of simple building blocks, known as layers
- Simplest layer: linear, i.e. matrix

$$L : \mathbb{R}^n \to \mathbb{R}^m$$

- Linear layers are typically combined with elementwise nonlinearities, e.g. sigmoid σ



Neural networks

Neural network blocks:

- Multi-layer perceptron:

$$\mathrm{MLP} = \sigma \circ L \circ \cdots \sigma \circ L$$

Neural networks

Neural network blocks:

- Multi-layer perceptron:

$$\mathrm{MLP} = \sigma \circ L \circ \cdots \sigma \circ L$$

Composition of simple layers is crucial for hardware optimization

Neural networks

Neural network blocks:

- Multi-layer perceptron:

$$\mathrm{MLP} = \sigma \circ L \circ \cdots \sigma \circ L$$

Matrix entries are trainable parameters that are fit to the data

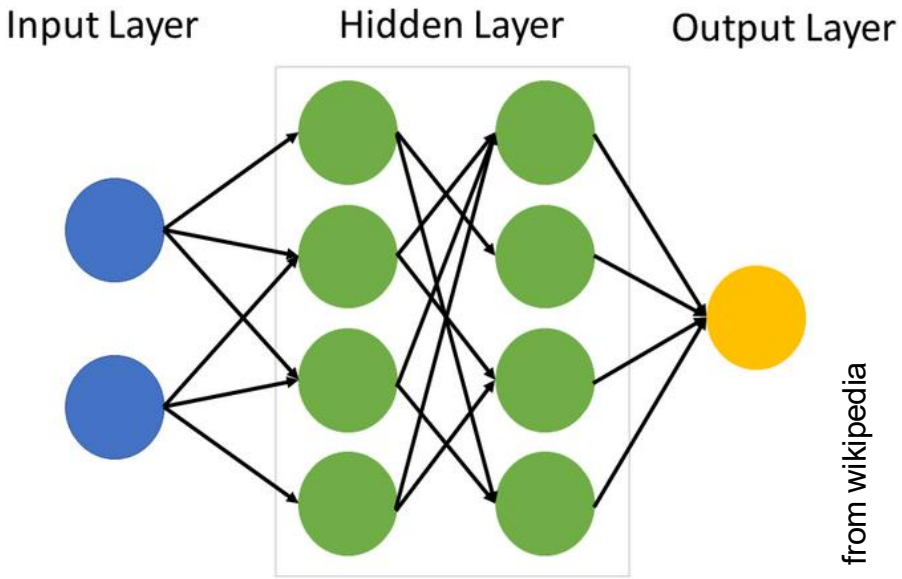
Neural networks

Neural network blocks:

- Multi-layer perceptron:

$$\mathrm{MLP} = \sigma \circ L \circ \cdots \sigma \circ L$$

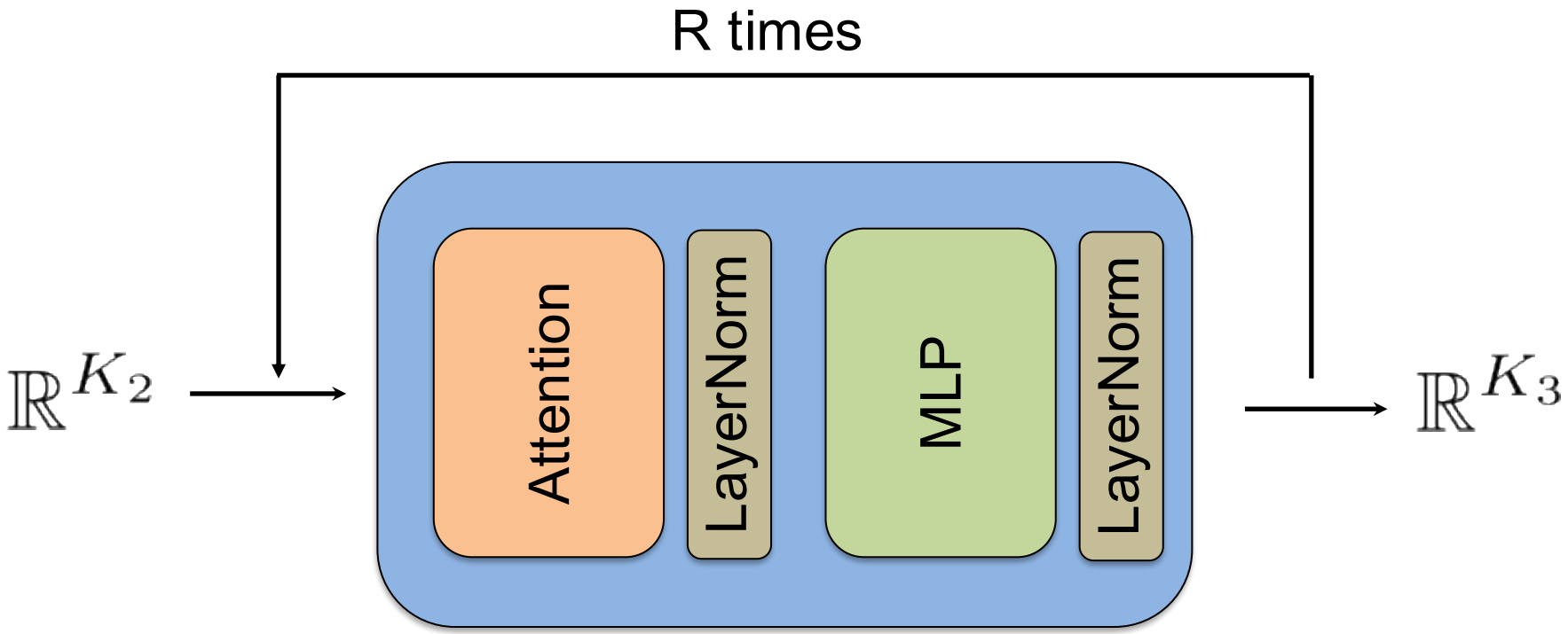
Matrix entries are trainable parameters that are fit to the data



from wikipedia

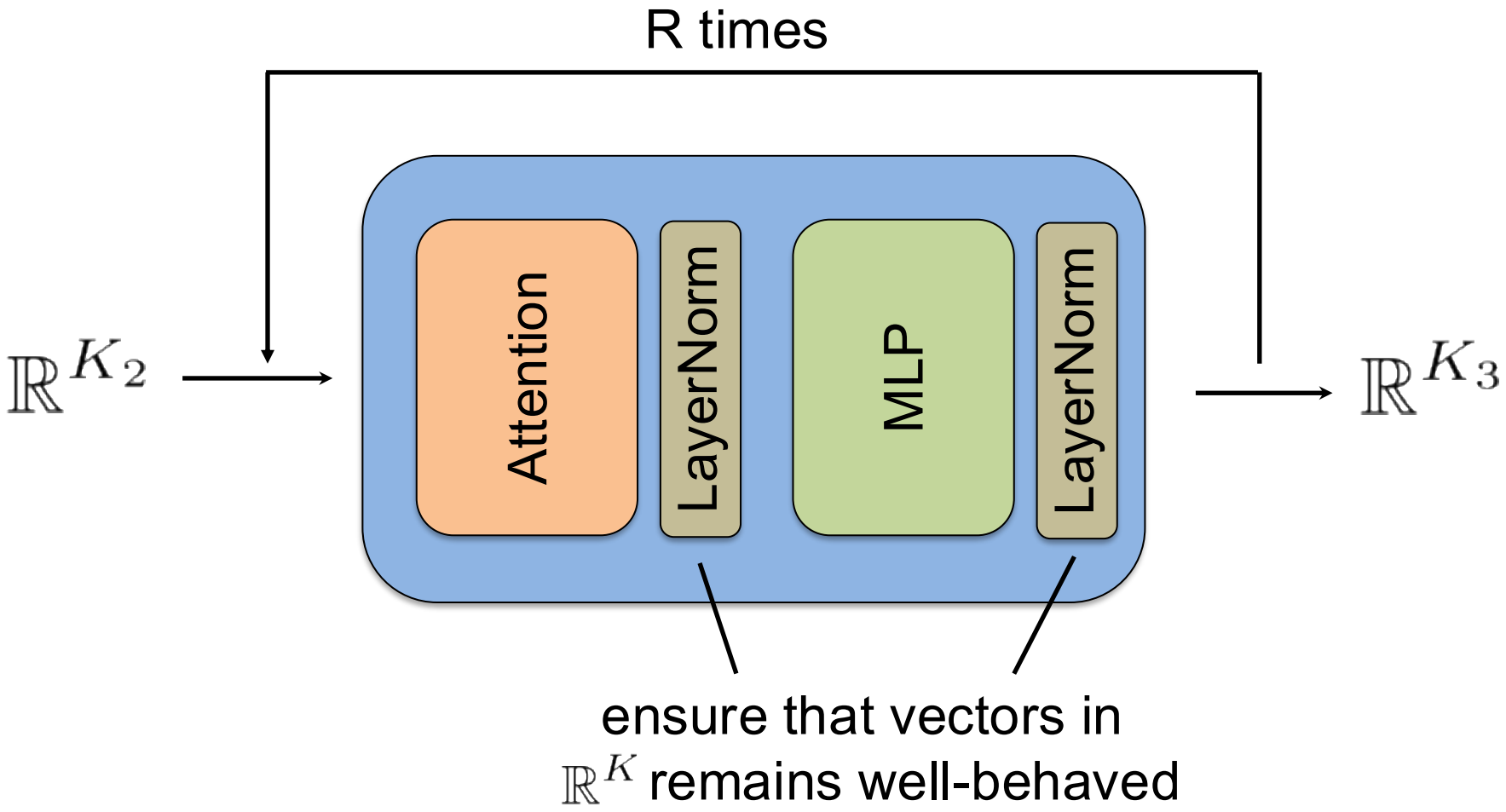
Neural networks

Example for processor:



Neural networks

Example for processor:



Neural network training

Goal: find network weights to optimal model the data as given by loss function:

Neural network training

Goal: find network weights to optimal model the data as given by loss function:

- Loss:

$$L(y, \bar{y}) : \mathbb{R}^M \times \mathbb{R}^M \rightarrow \mathbb{R}$$

Neural network training

Goal: find network weights to optimal model the data as given by loss function:

- Loss:

$$L(y, \bar{y}) : \mathbb{R}^M \times \mathbb{R}^M \rightarrow \mathbb{R}$$

- For example:

$$\mathrm{MSE}(y, \bar{y}) = \frac{1}{M} \sum_{i=1}^{M} (y_i - \bar{y}_i)^2$$

Neural network training

Goal: find network weights to optimal model the data as given by loss function:

- Loss:

$$L(y, \bar{y}) : \mathbb{R}^M \times \mathbb{R}^M \rightarrow \mathbb{R}$$

Training: gradient-based optimization of network weights

Loss is a function of network weights:

Neural network training

Goal: find network weights to optimal model the data as given by loss function:

- Loss:

$$L(y, \bar{y}) : \mathbb{R}^M \times \mathbb{R}^M \rightarrow \mathbb{R}$$

Training: gradient-based optimization of network weights

Loss is a function of network weights:

$$L(y, \bar{y}) = L(y, \mathcal{N}_\theta(x))$$

Neural network training

Goal: find network weights to optimal model the data as given by loss function:

- Loss:

$$L(y, \bar{y}) : \mathbb{R}^M \times \mathbb{R}^M \rightarrow \mathbb{R}$$

Training: gradient-based optimization of network weights

Loss is a function of network weights:

$$L(y, \bar{y}) = L(y, \mathcal{N}_\theta(x))$$

Optimization problem:

$$\min_\theta L(y, \mathcal{N}_\theta(x)) + \Omega(\theta) \quad \forall (x, y) \in \mathcal{T}$$

Neural network training

Goal: find network weights to optimal model the data as given by loss function:

- Loss:

$$L(y, \bar{y}) : \mathbb{R}^M \times \mathbb{R}^M \rightarrow \mathbb{R}$$

Training: gradient-based optimization of network weights

Loss is a function of network weights:

$$L(y, \bar{y}) = L(y, \mathcal{N}_\theta(x))$$

Optimization problem:

$$\min_\theta L(y, \mathcal{N}_\theta(x)) + \Omega(\theta) \qquad \forall (x, y) \in \boxed{\mathcal{T}}$$

training data (or set)

Neural network training

Goal: find network weights to optimal model the data as given by loss function:

- Loss:

$$L(y, \bar{y}) : \mathbb{R}^M \times \mathbb{R}^M \rightarrow \mathbb{R}$$

Training: gradient-based optimization of network weights

Loss is a function of network weights:

$$L(y, \bar{y}) = L(y, \mathcal{N}_\theta(x))$$

Optimization problem:

$$\min_\theta L(y, \mathcal{N}_\theta(x)) + \Omega(\theta) \qquad \forall (x, y) \in \boxed{\mathcal{T}}$$

training data (or set)

Neural network training

Goal: find network weights to optimal model the data as given by loss function:

- Loss:

$$L(y, \bar{y}) : \mathbb{R}^M \times \mathbb{R}^M \rightarrow \mathbb{R}$$

Training: gradient-based optimization of network weights

Loss is a function of network weights:

$$L(y, \bar{y}) = L(y, \mathcal{N}_\theta(x))$$

Optimization problem:

$$\min_\theta L(y, \mathcal{N}_\theta(x)) + \boxed{\Omega(\theta)} \quad \forall (x, y) \in \boxed{\mathcal{T}}$$

weight regularization

training data (or set)

e.g.: $\lambda ||\theta||$

Neural network training

Training: gradient-based optimization of network weights

Optimization problem: $\min_\theta L(y, \mathcal{N}_\theta(x)) + \Omega(\theta) \quad \forall (x, y) \in \mathcal{T}$

Neural network training

Training: **gradient**-based optimization of network weights

Optimization problem: $\min_\theta L(y, \mathcal{N}_\theta(x)) + \Omega(\theta) \quad \forall (x, y) \in \mathcal{T}$

Neural network training

Training: **gradient**-based optimization of network weights

Optimization problem: $\min_\theta L(y, \mathcal{N}_\theta(x)) + \Omega(\theta) \quad \forall (x, y) \in \mathcal{T}$

Optimization: $\theta^{j+1} = \theta^j + \lambda \tilde{\nabla}_\theta \mathcal{L}_\mathcal{T}(y, \mathcal{N}_{\theta_j}) \quad j = 1, \cdots, J$

Neural network training

Training: **gradient**-based optimization of network weights

Optimization problem: $\min_\theta L(y, \mathcal{N}_\theta(x)) + \Omega(\theta) \quad \forall (x, y) \in \mathcal{T}$

Optimization: $\theta^{j+1} = \theta^j + \lambda \tilde{\nabla}_\theta \mathcal{L}_\mathcal{T}(y, \mathcal{N}_{\theta_j}) \quad j = 1, \cdots, J$

Neural network training

Training: **gradient**-based optimization of network weights

Optimization problem: $\min_\theta L(y, \mathcal{N}_\theta(x)) + \Omega(\theta) \quad \forall (x, y) \in \mathcal{T}$

Optimization: $\theta^{j+1} = \theta^j + \lambda \tilde{\nabla}_\theta \mathcal{L}_\mathcal{T}(y, \mathcal{N}_{\theta_j})$ $\boxed{j = 1, \cdots, J}$

optimizer steps

Neural network training

Training: **gradient**-based optimization of network weights

Optimization problem: $\min_\theta L(y, \mathcal{N}_\theta(x)) + \Omega(\theta) \quad \forall (x, y) \in \mathcal{T}$

Optimization: $\theta^{j+1} = \theta^j + \lambda \tilde{\nabla}_\theta \mathcal{L}_\mathcal{T}(y, \mathcal{N}_{\theta_j})$ $\boxed{j = 1, \cdots, J}$

optimizer steps
typically subdivided into epochs
(epoch = one sweep through
training dataset)

Neural network training

Training: **gradient**-based optimization of network weights

Optimization problem: $\min_\theta L(y, \mathcal{N}_\theta(x)) + \Omega(\theta) \quad \forall (x, y) \in \mathcal{T}$

Optimization: $\theta^{j+1} = \theta^j + \lambda \tilde{\nabla}_\theta \mathcal{L}_\mathcal{T}(y, \mathcal{N}_{\theta_j}) \quad j = 1, \cdots, J$

Euler-type update to weights

Neural network training

Training: **gradient**-based optimization of network weights

Optimization problem: $\min_\theta L(y, \mathcal{N}_\theta(x)) + \Omega(\theta) \quad \forall (x, y) \in \mathcal{T}$

Optimization: $\theta^{j+1} = \theta^j + \boxed{\lambda} \tilde{\nabla}_\theta \mathcal{L}_\mathcal{T}(y, \mathcal{N}_{\theta_j}) \quad j = 1, \cdots, J$

learning rate

Neural network training

Training: **gradient**-based optimization of network weights

Optimization problem: $\min_\theta L(y, \mathcal{N}_\theta(x)) + \Omega(\theta) \quad \forall (x, y) \in \mathcal{T}$

Optimization: $\theta^{j+1} = \theta^j + \lambda \boxed{\tilde{\nabla}_\theta \mathcal{L}_\mathcal{T}(y, \mathcal{N}_{\theta_j})} \quad j = 1, \cdots, J$

approximation
to gradient

Neural network training

Training: **gradient**-based optimization of network weights

Optimization problem: $\min_\theta L(y, \mathcal{N}_\theta(x)) + \Omega(\theta) \quad \forall (x, y) \in \mathcal{T}$

Optimization: $\theta^{j+1} = \theta^j + \lambda \boxed{\tilde{\nabla}_\theta \mathcal{L}_\mathcal{T}(y, \mathcal{N}_{\theta_j})} \quad j = 1, \cdots, J$

approximation
to gradient

- makes training computationally tractable

Neural network training

Training: **gradient**-based optimization of network weights

Optimization problem: $\min_\theta L(y, \mathcal{N}_\theta(x)) + \Omega(\theta) \quad \forall (x, y) \in \mathcal{T}$

Optimization: $\theta^{j+1} = \theta^j + \lambda \boxed{\tilde{\nabla}_\theta \mathcal{L}_\mathcal{T}(y, \mathcal{N}_{\theta_j})} \quad j = 1, \cdots, J$

approximation
to gradient

- makes training computationally tractable
- helps to avoid getting stuck in local minima

Neural network training

Training: **gradient**-based optimization of network weights

Optimization problem: $\min_\theta L(y, \mathcal{N}_\theta(x)) + \Omega(\theta) \quad \forall (x, y) \in \mathcal{T}$

Optimization: $\theta^{j+1} = \theta^j + \lambda \tilde{\nabla}_\theta \mathcal{L}_\mathcal{T}(y, \mathcal{N}_{\theta_j}) \quad j = 1, \cdots, J$

$$\theta^{j+1} = \theta^j + \lambda \nabla_\theta \Big(\sum_{(x,y) \in B} L(y, \mathcal{N}_{\theta_j}(x)) \Big)$$

Neural network training

Training: **gradient**-based optimization of network weights

Optimization problem: $\min_\theta L(y, \mathcal{N}_\theta(x)) + \Omega(\theta) \quad \forall (x, y) \in \mathcal{T}$

Optimization: $\theta^{j+1} = \theta^j + \lambda \tilde{\nabla}_\theta \mathcal{L}_\mathcal{T}(y, \mathcal{N}_{\theta_j}) \quad j = 1, \cdots, J$

$$\theta^{j+1} = \theta^j + \lambda \nabla_\theta \Big(\sum_{(x,y) \in B} L(y, \mathcal{N}_{\theta_j}(x)) \Big)$$

mini batch

$$|B| \ll |\mathcal{T}|$$

Neural network training

Training: **gradient**-based optimization of network weights

Optimization problem: $\min_\theta L(y, \mathcal{N}_\theta(x)) + \Omega(\theta) \quad \forall (x, y) \in \mathcal{T}$

Optimization: $\theta^{j+1} = \theta^j + \lambda \tilde{\nabla}_\theta \mathcal{L}_\mathcal{T}(y, \mathcal{N}_{\theta_j}) \quad j = 1, \cdots, J$

$$\theta^{j+1} = \theta^j + \lambda \nabla_\theta \Big(\sum_{(x,y) \in B} L(y, \mathcal{N}_{\theta_j}(x)) \Big)$$

$$\theta^{j+1} = \theta^j + \lambda \Big(\sum_{(x,y) \in B} \nabla_\theta L(y, \mathcal{N}_{\theta_j}(x)) \Big)$$

Neural network training

Training: **gradient**-based optimization of network weights

Optimization problem: $\min_\theta L(y, \mathcal{N}_\theta(x)) + \Omega(\theta) \quad \forall (x, y) \in \mathcal{T}$

Optimization: $\theta^{j+1} = \theta^j + \lambda \tilde{\nabla}_\theta \mathcal{L}_\mathcal{T}(y, \mathcal{N}_{\theta_j}) \quad j = 1, \cdots, J$

$$\theta^{j+1} = \theta^j + \lambda \nabla_\theta \Big(\sum_{(x,y) \in B} L(y, \mathcal{N}_{\theta_j}(x)) \Big)$$

$$\theta^{j+1} = \theta^j + \lambda \Big(\sum_{(x,y) \in B} \boxed{\nabla_\theta L(y, \mathcal{N}_{\theta_j}(x))} \Big)$$

per sample gradient
is computed with back propagation
(algorithm to compute chain rule)

Neural network training

Training, validation, and test set:
- Goal: perform well on unseen (during training) data
- Problem: Neural network can memorize training data, i.e. **overfit**

Neural network training

Training, validation, and test set:
- Goal: perform well on unseen (during training) data
- Problem: Neural network can memorize training data, i.e. **overfit**

- Training set: used to optimize neural network
- Validation set: independent dataset to monitor overfitting during training and chose hyper-parameter (learning rate, network shape and form, …)
- Test set: independent dataset to evaluate skill independent of hyper-parameter

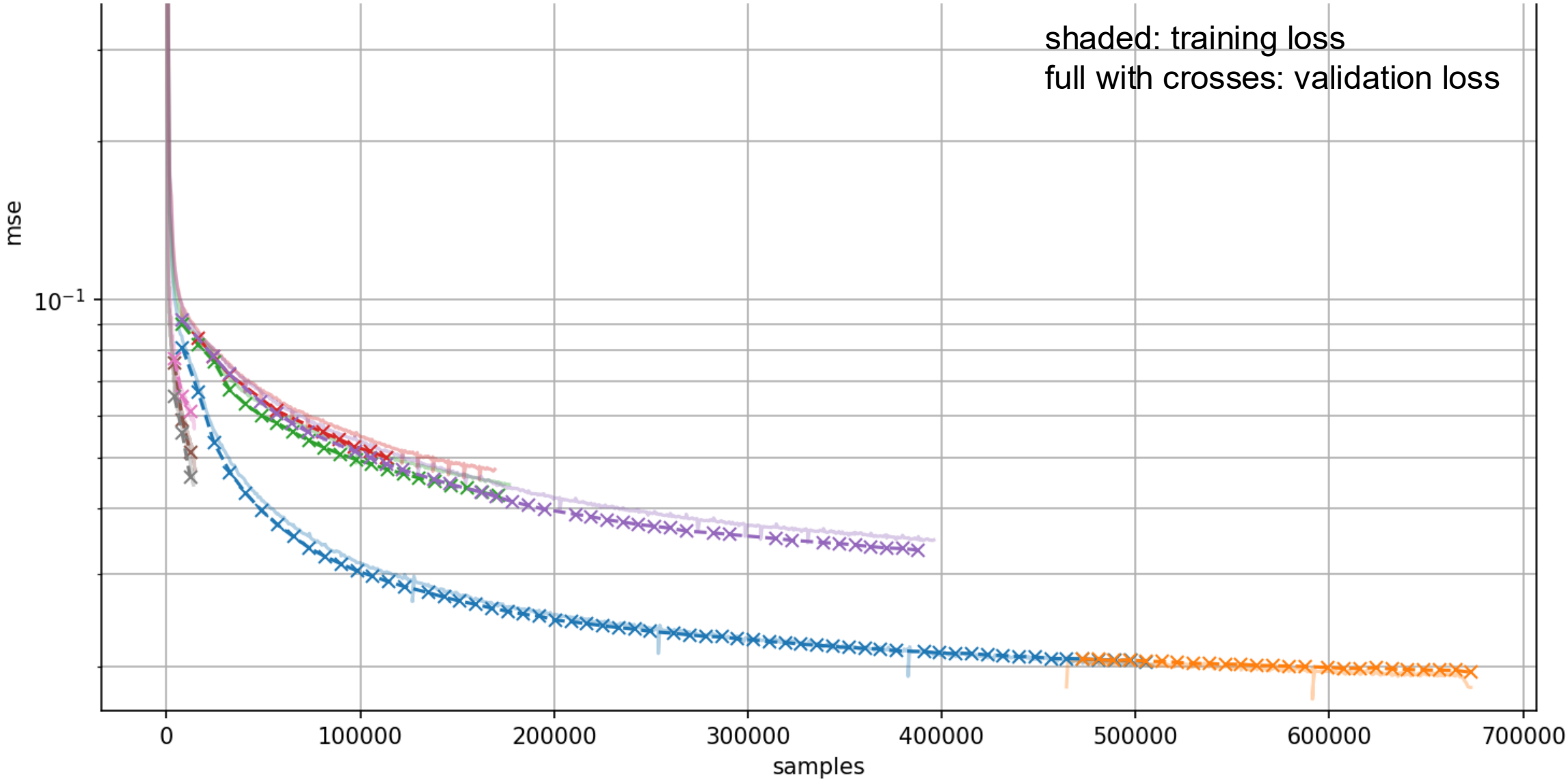
Neural network training

Training, validation, and test set:
- Goal: perform well on unseen (during training) data
- Problem: Neural network can memorize training data, i.e. **overfit**

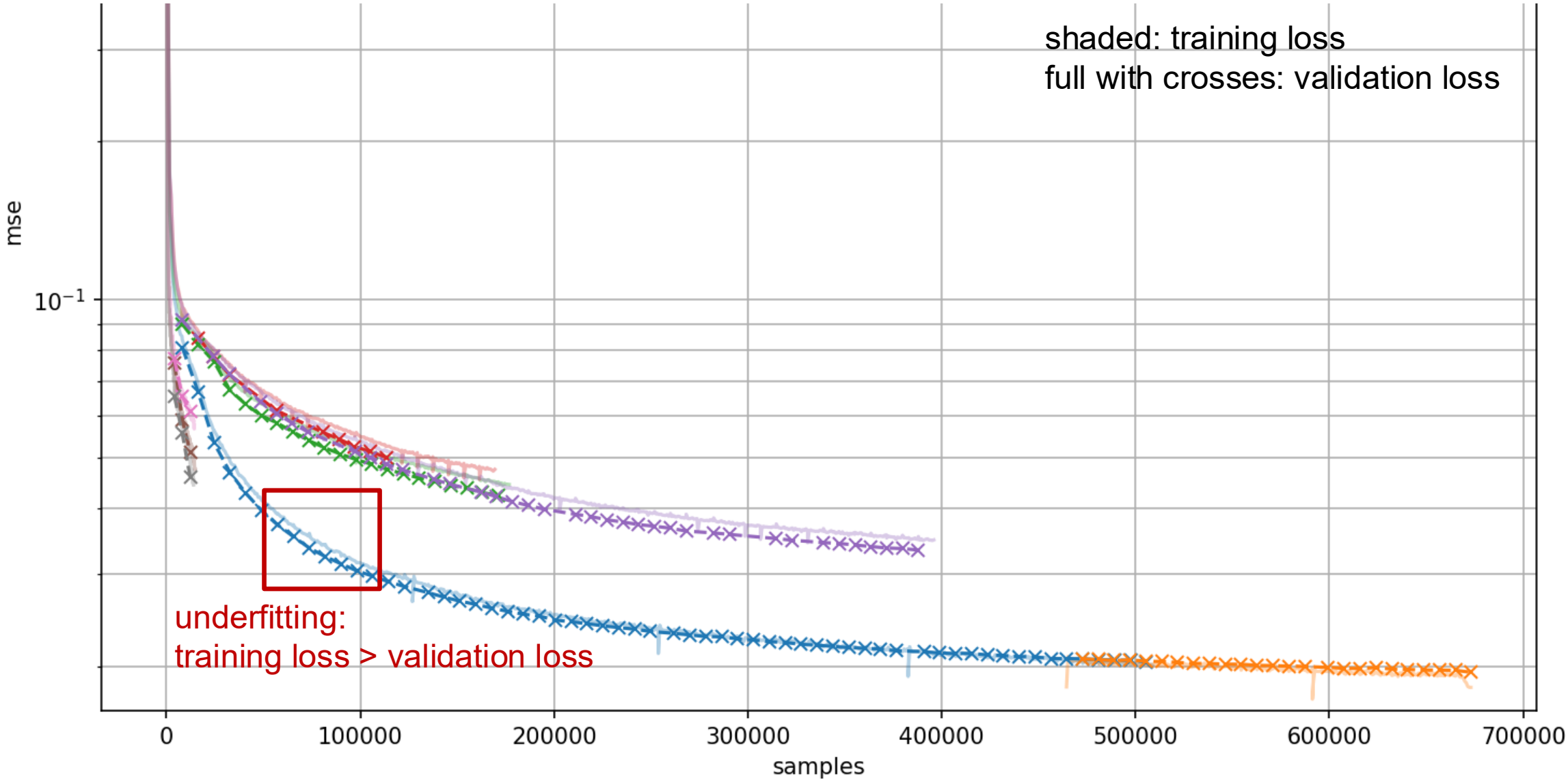
- Training set: used to optimize neural network
- Validation set: independent dataset to monitor overfitting during training and chose hyper-parameter (learning rate, network shape and form, …)
- Test set: independent dataset to evaluate skill independent of hyper-parameter

- Important: training, validation, and test set are iid samples from the same distribution

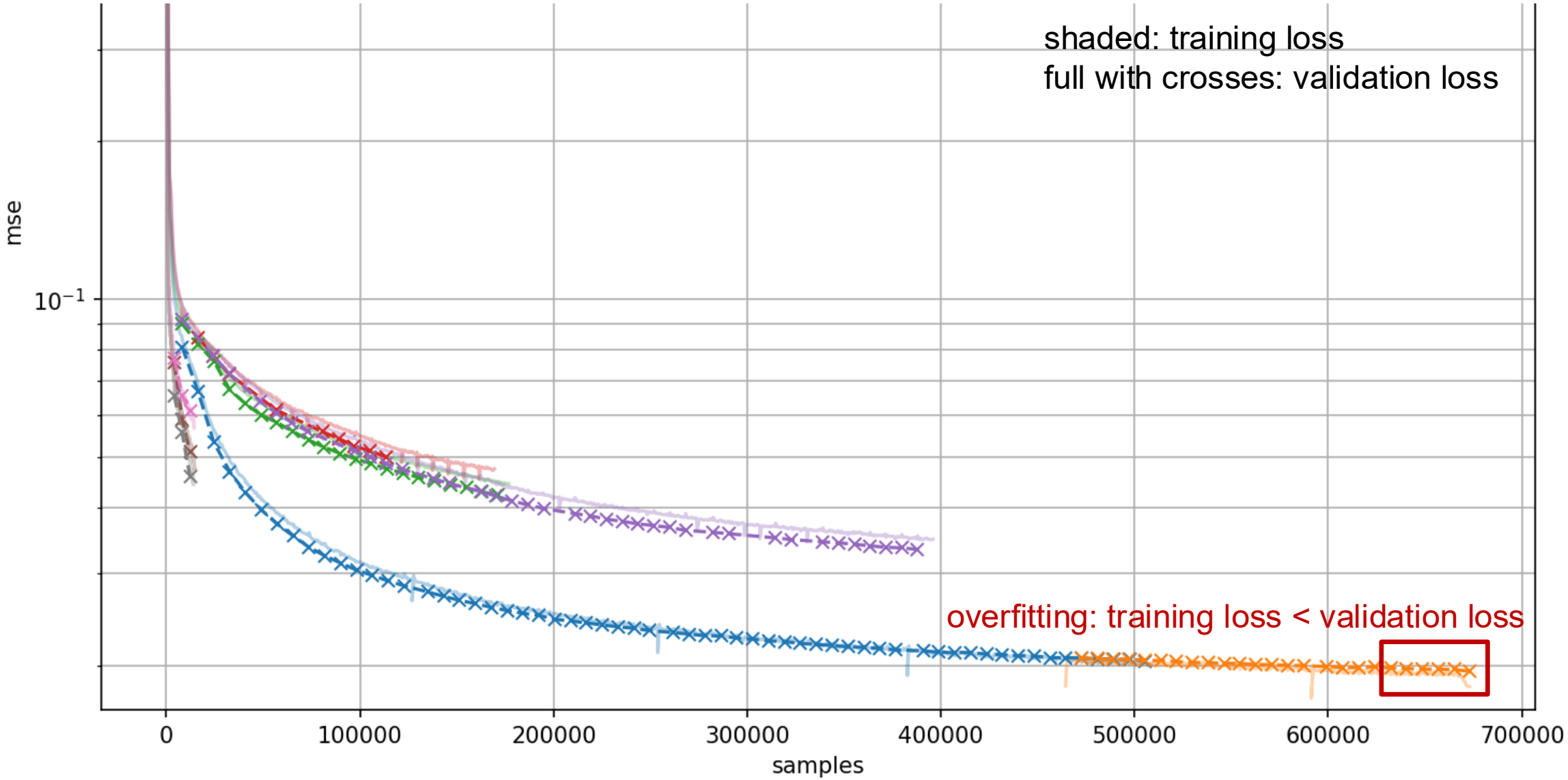
Neural network training



Neural network training



Neural network training



Conclusion

- Neural networks
  - Weakly nonlinear mappings $\mathcal{N} : \mathbb{R}^N \rightarrow \mathbb{R}^M$
  - Consist of simple building blocks that are repeated
    - Crucial for fast execution
    - Building
- Training
  - Solution of nonlinear optimization problem to find optimal neural network weights given the training data
  - Stochastic gradient descent (and variants such as Adam) to solve optimization
- Inference
  - Application of neural network in application to unseen data


ECMWF
EUROPEAN CENTRE FOR MEDIUM-RANGE WEATHER FORECASTS