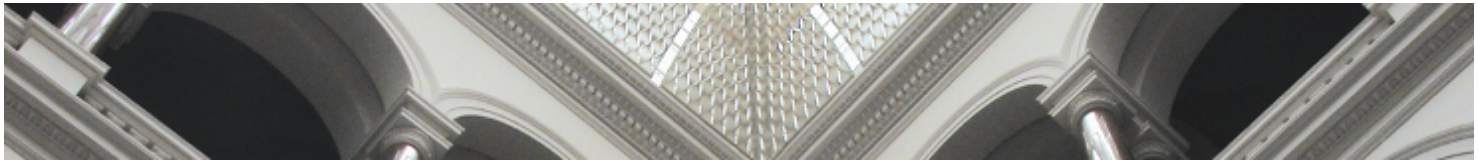
Technische Universität Berlin

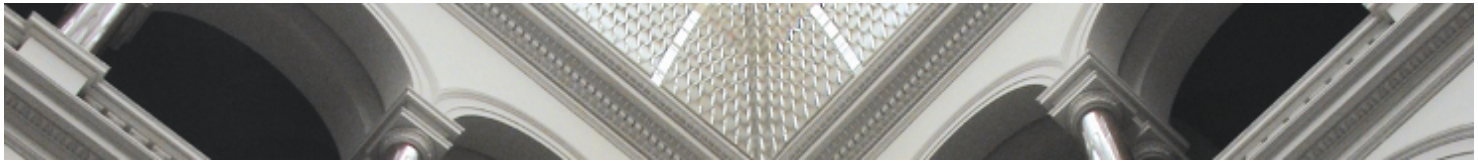
Model Order Reduction of Rarefied Gases Using Neural Networks

Zachary Schellin | Institut für Numerische Fluiddynamik



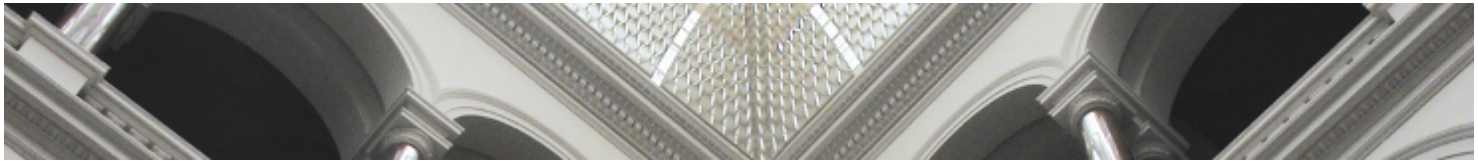
Technische Universität Berlin

Outline



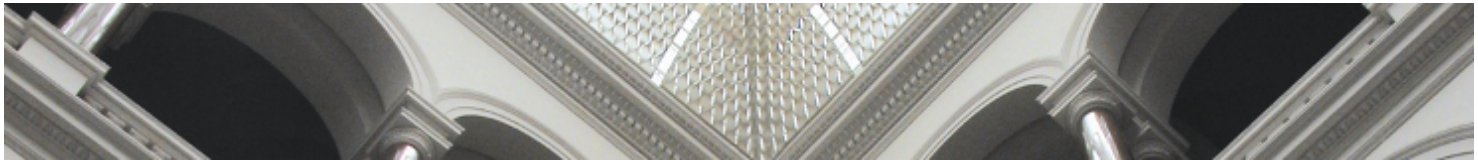
Technische Universität Berlin

Table of Contents



Technische Universität Berlin

Table of Contents



Technische Universität Berlin

Governing equations

– The Boltzmann equation approximated by $\boldsymbol{Q}$ the BGK operator as a source term with

$$\partial_t f + v\partial_x f = \overbrace{\frac{1}{\tau}(M_f - f)}^{Q} \tag{1}$$

– The equilibrium solution is a Maxwellian distribution $\boldsymbol{M_f}$ with

$$M_f = \frac{\rho(x,t)}{(2\pi RT(x,t))^{\frac{3}{2}}} \exp\left(-\frac{(v-u(x,t))^2}{2RT(x,t)}\right) \tag{2}$$

– The duration to evolve into equilibrium is given by the relaxation time $\boldsymbol{\tau}$ with

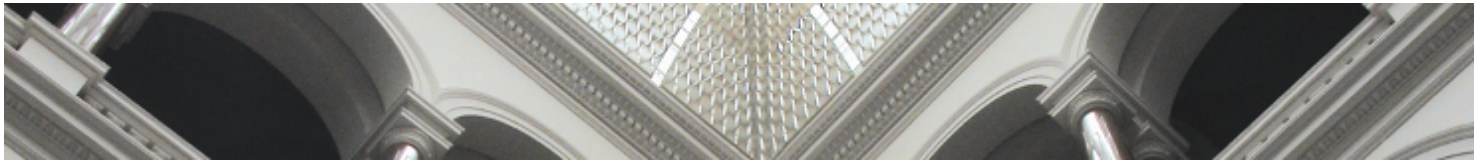
$$\tau^{-1} = \frac{\rho(x,t)T^{1-\nu}(x,t)}{Kn} \tag{3}$$

– The rarefaction level is defined over the Knudsen number $\boldsymbol{Kn}$ with

$$Kn = \frac{\lambda}{l} \tag{4}$$

1

[1] **PhysRev.94.511**.



Technische Universität Berlin

Knudsen number

– Solution is $f(x, v, t)$ in 1D and $f(x, y, v_x, v_y, t)$ in 2D and $f(x, y, z, v_x, v_y, v_z, t)$ in 3D

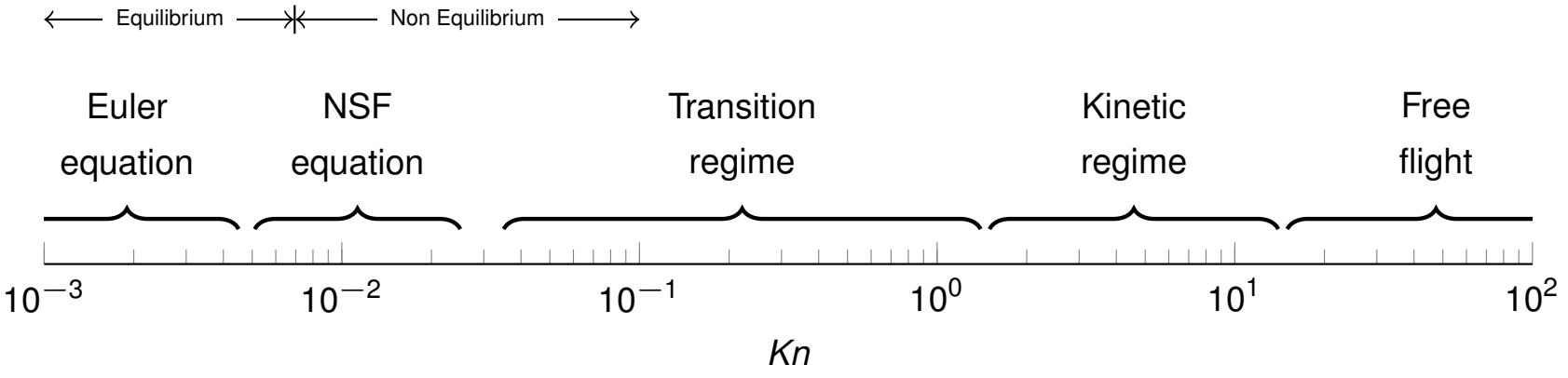
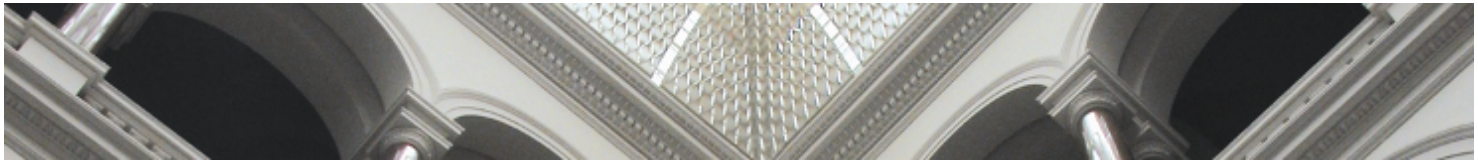


Figure: Partitioning of Kn, the Knudsen number, into levels of rarefaction.

2

[2] **NumaKUL**.



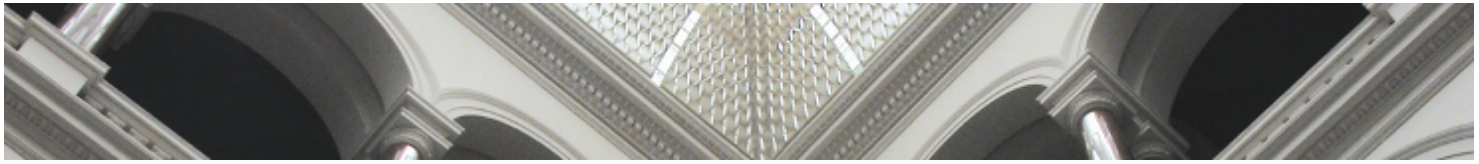
Technische Universität Berlin

Discretization in space and velocity space in 1D

- Space and time discretization considering a uniform grid i.e. $x_j = j\Delta x$ and $j \in \mathbb{Z}$, $v_k = k\Delta v$ and $k \in \mathbb{Z}$, $t^i = i\Delta t$ and $t \in \mathbb{N}$,
- is leading from the full PDE to a set of ODE's in time

$$\partial_t f_{j,k} = -(v_k)_1 D_x f\big|_{j,k}(t) + \frac{1}{\tau}(M_{f_{j,k}}(t) - f_{j,k}(t)). \tag{5}$$

- KJ first-order differential equations need to be evaluated when K and J are the number of gridpoints in space and velocity space.
- In 3D there are $K^3 J^3$ first-order differential equations.
- The discretization in velocity space requires the computation of the moments of f.



Technische Universität Berlin

Moments/ Expected values of f

- Collision invariants $\Phi(v) = [1, v, \frac{1}{2}v^2]$
- The first moment/ Density is

$$\rho(x,t) = \int f \, \mathrm{d}v \,, \tag{6}$$

- the second moment/ Momentum is

$$\rho(x,t)u(x,t) = \int vf \, \mathrm{d}v \,, \tag{7}$$

- the third moment/ kinetic Energy is

$$E(x,t) = \int \frac{1}{2} v^2 f \, \mathrm{d}v \,. \tag{8}$$

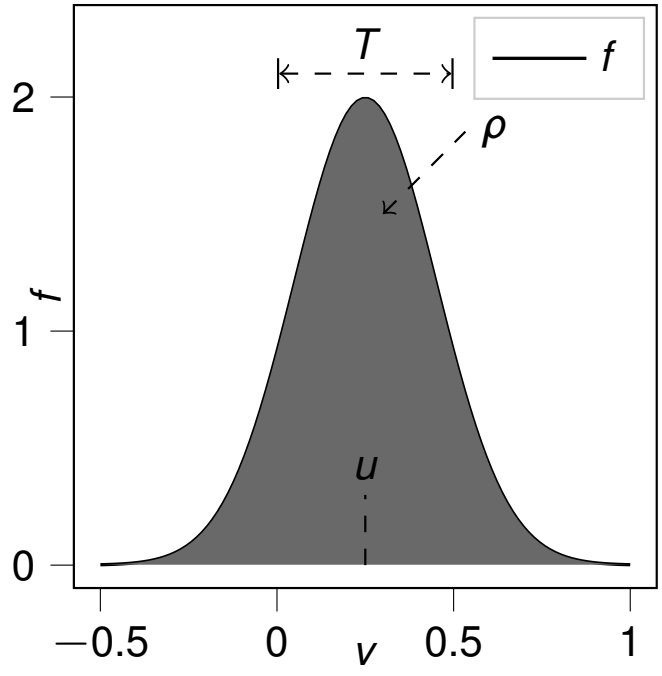
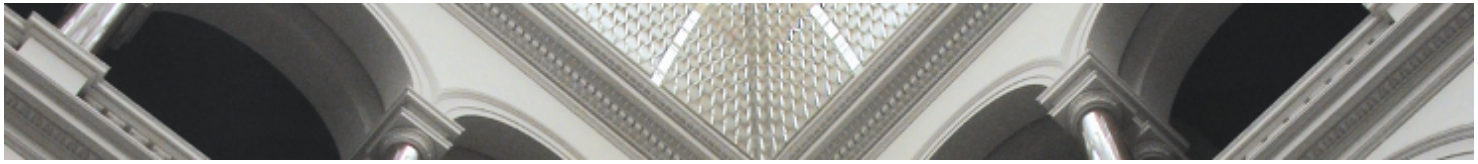
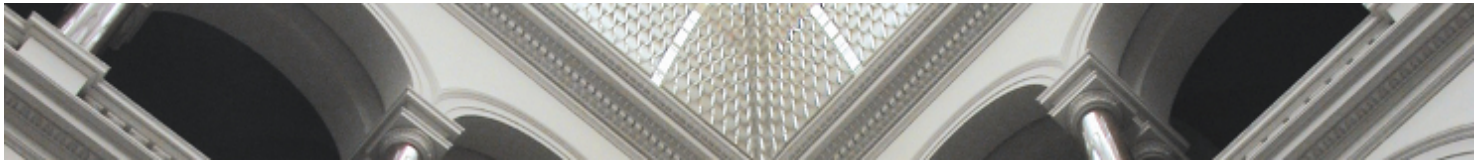


Figure: Illustration of the linkage between the macroscopic quantities of the gas flow and the distribution function f.



Technische Universität Berlin

Table of Contents



Sod's shock tube

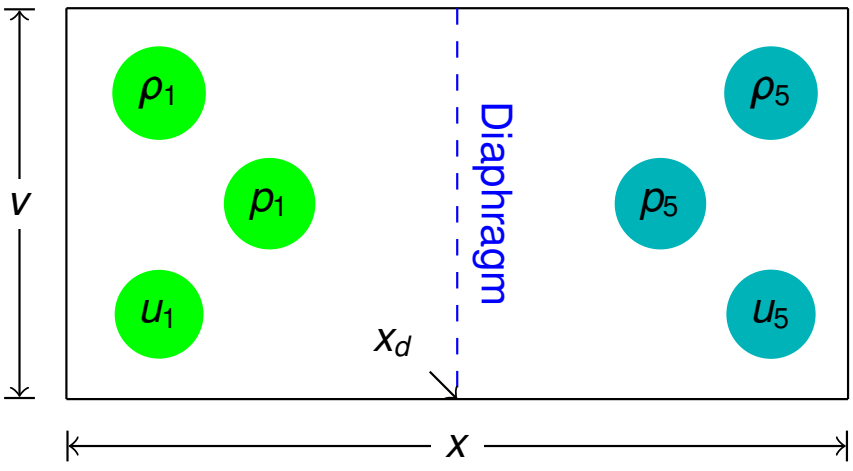
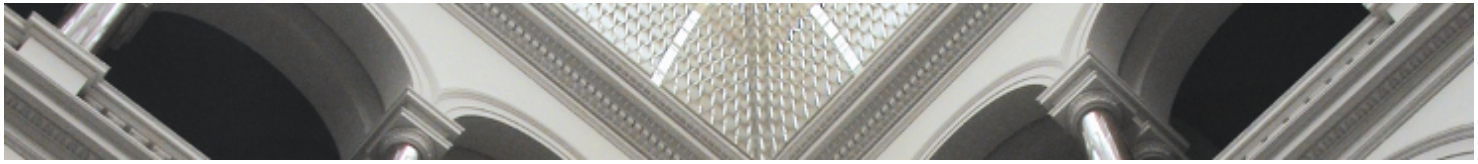


Figure: Problem setup of Sod's shock tube for the BGK model in 1D at $t = 0s$.

- Test case for numerical schemes solving
- **non-linear hyperbolic conservation laws** in gas dynamics (Gary A. Sod in 1978)
- **Idea**:
  - Solve problem analytically (Rankine-Hugoniot jump conditions)
  - Solve problem numerically
  - Compare results expecially **resolution of discontinuities**



Technische Universität Berlin

Sod's shock tube

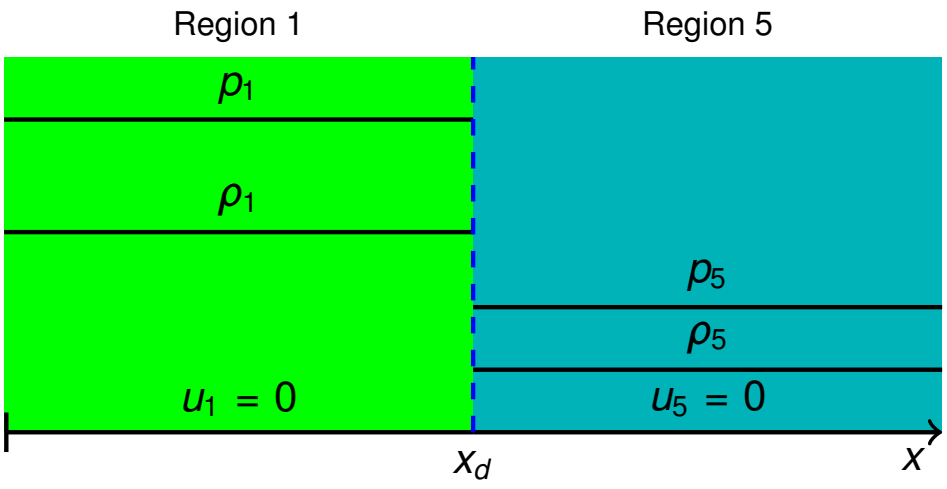


Figure: Problem setup of Sod's shock tube for the BGK model in 1D at $t = 0s$.

- Test case for numerical schemes solving
- **non-linear hyperbolic conservation laws** in gas dynamics (Gary A. Sod in 1978)
- **Idea**:
  - Solve problem analytically (Rankine-Hugoniot jump conditions)
  - Solve problem numerically
  - Compare results expecially **resolution of discontinuities**

Sod's shock tube

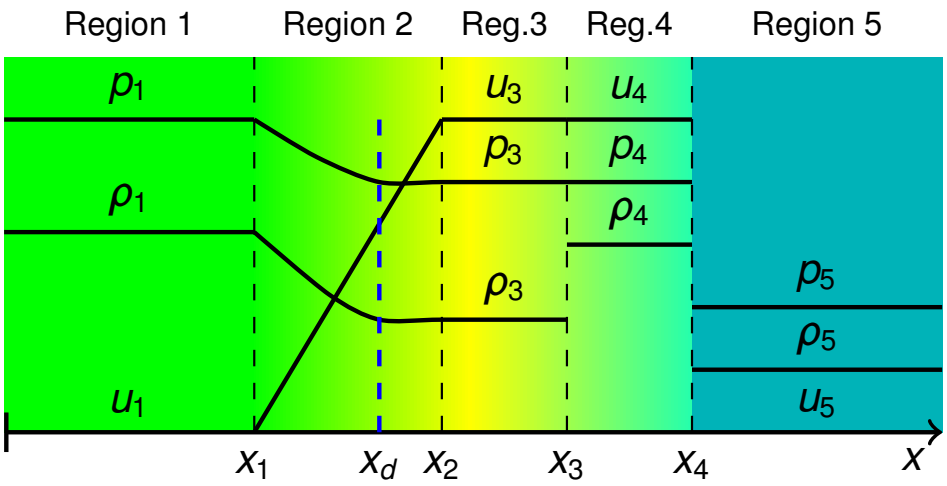


Figure: Problem setup of Sod's shock tube for the BGK model in 1D at $t > 0s$.

- Test case for numerical schemes solving
- **non-linear hyperbolic conservation laws** in gas dynamics (Gary A. Sod in 1978)
- **Idea**:
  - Solve problem analytically (Rankine-Hugoniot jump conditions)
  - Solve problem numerically
  - Compare results expecially **resolution of discontinuities**
    - » $\mathbf{x_1}$ head of rarefaction wave
    - » $\mathbf{x_2}$ tail of rarefaction wave
    - » $\mathbf{x_3}$ contact discontinuity
    - » $\mathbf{x_4}$ position of shockwave

Two Case Studies

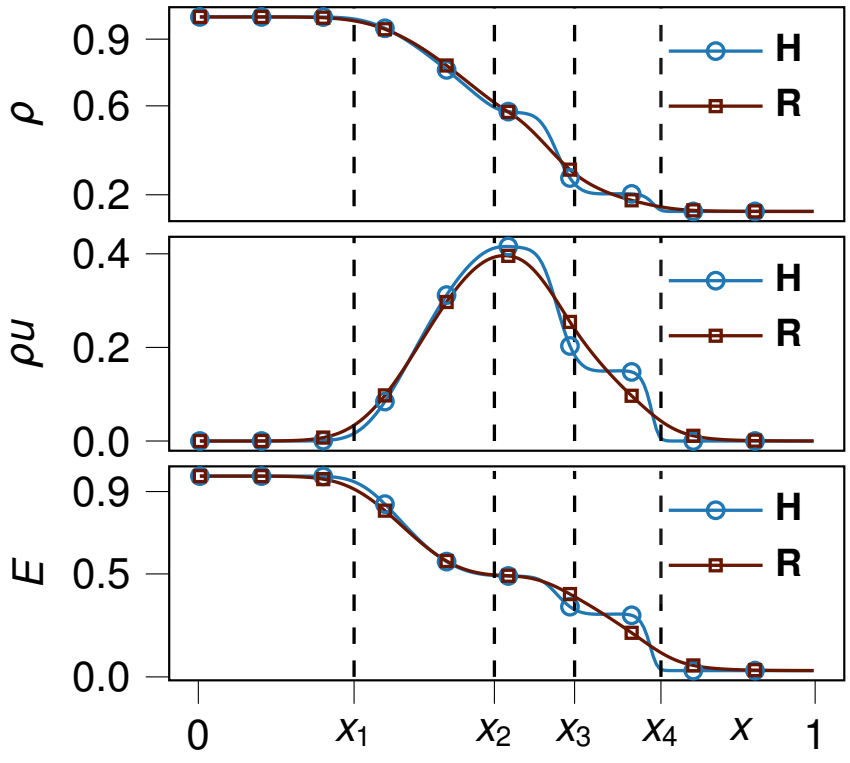


Figure: Moments of **H** and **R** at $t = 0.12s$ and $v = v_0$ in Sod's shock tube.

- Two solutions of the BGK model in Sod's shock tube
- Two levels of rarefaction
  - **H**, $Kn = 0.00001$, "Continuum Flow"
  - **R**, $Kn = 0.001$, "Slip flow"
- Present discontinuities

H

- x_1
- x_2
- x_3
- x_4

R

- x_1
- x_2
- x_3
- x_4

Two Case Studies

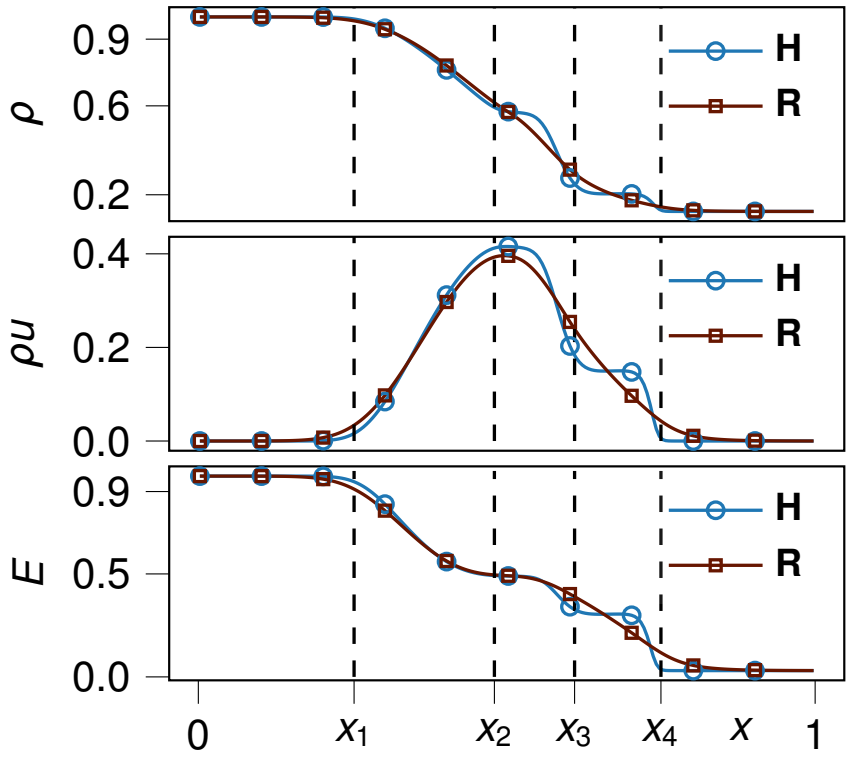


Figure: Moments of **H** and **R** at $t = 0.12s$ and $v = v_0$ in Sod's shock tube.

- Two solutions of the BGK model in Sod's shock tube
- Two levels of rarefaction
  - **H**, $Kn = 0.00001$, "Continuum Flow"
  - **R**, $Kn = 0.001$, "Slip flow"
- Present discontinuities

H

- x_1
- x_2
- x_3
- x_4

R

- x_1
- x_2
- x_3
- x_4

Two Case Studies

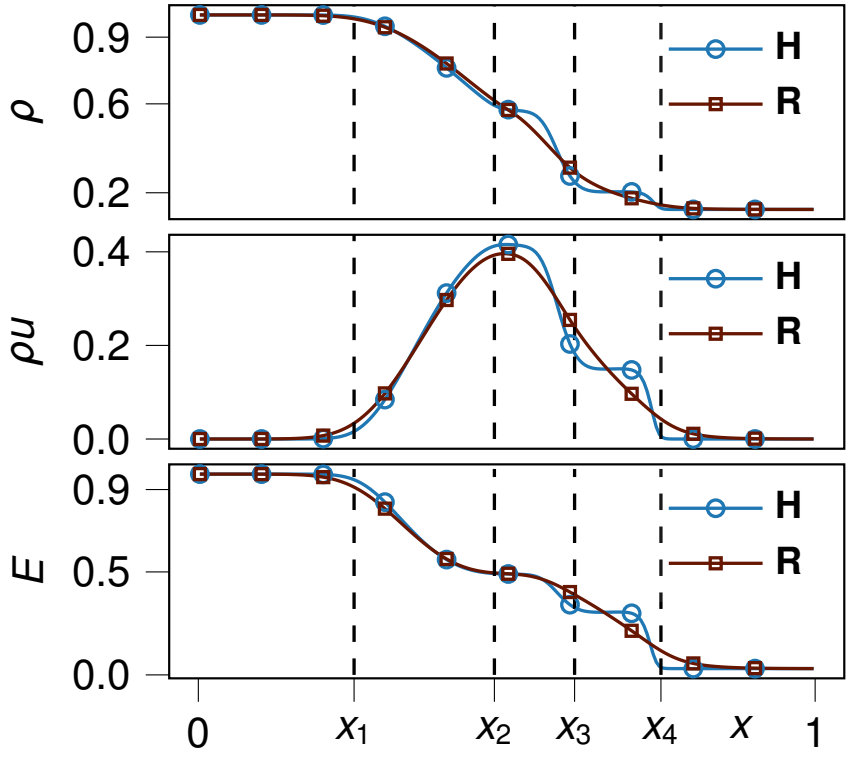


Figure: Moments of **H** and **R** at $t = 0.12s$ and $v = v_0$ in Sod's shock tube.

- Two solutions of the BGK model in Sod's shock tube
- Two levels of rarefaction
  - **H**, $Kn = 0.00001$, "Continuum Flow"
  - **R**, $Kn = 0.001$, "Slip flow"
- Present discontinuities

H

- x_1
- x_2
- x_3
- x_4

R

- x_1
- x_2
- x_3
- x_4

Two Case Studies

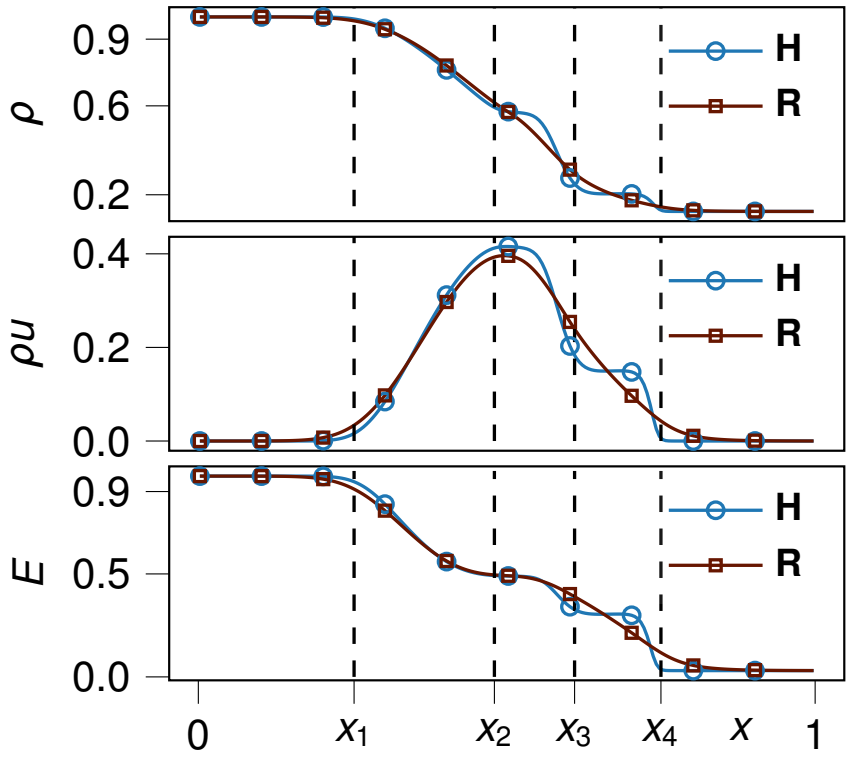


Figure: Moments of **H** and **R** at $t = 0.12s$ and $v = v_0$ in Sod's shock tube.

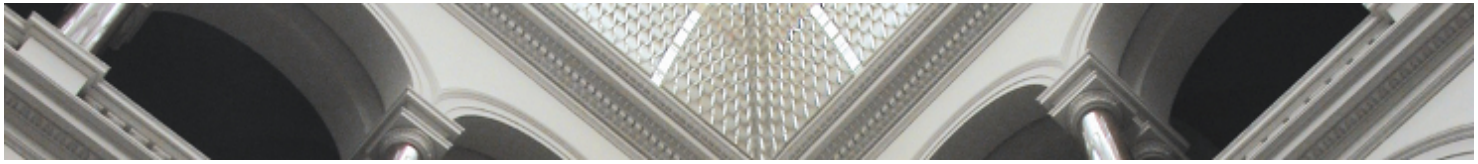
- Two solutions of the BGK model in Sod's shock tube
- Two levels of rarefaction
  - **H**, $Kn = 0.00001$, "Continuum Flow"
  - **R**, $Kn = 0.001$, "Slip flow"
- Present discontinuities

H

- x_1
- x_2
- x_3
- x_4

R

- x_1
- x_2
- x_3
- x_4



Technische Universität Berlin

Two Case Studies

- Spatial resolution $J = 200$
- Temporal resolution $I = 25$
- Velocious resolution $K = 40$

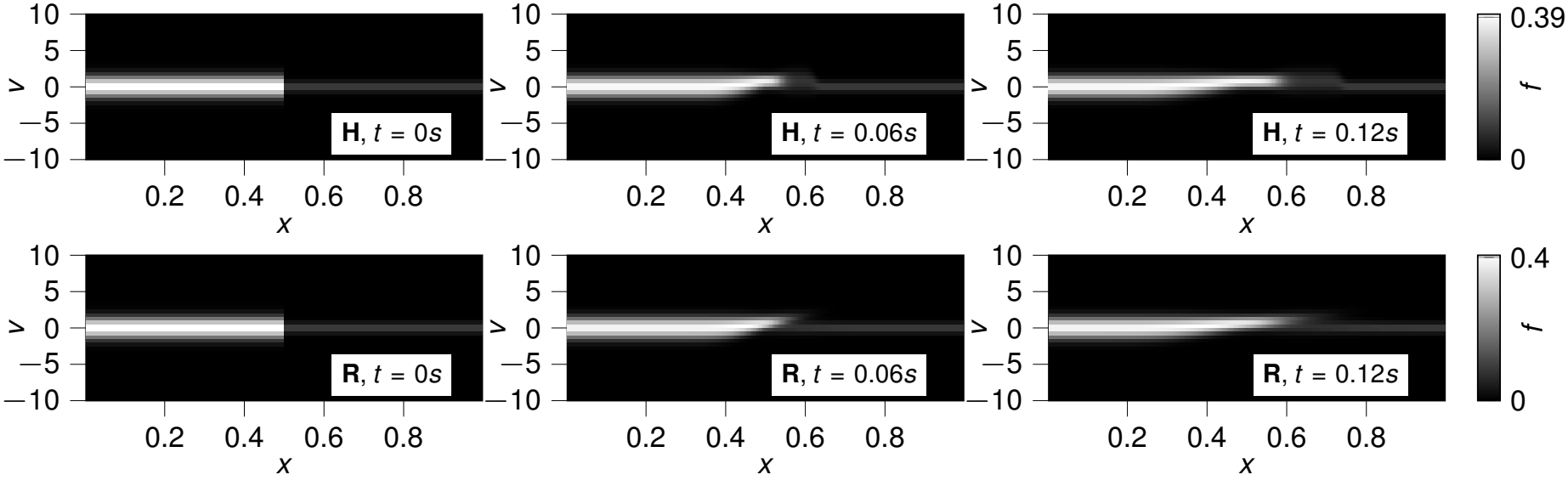
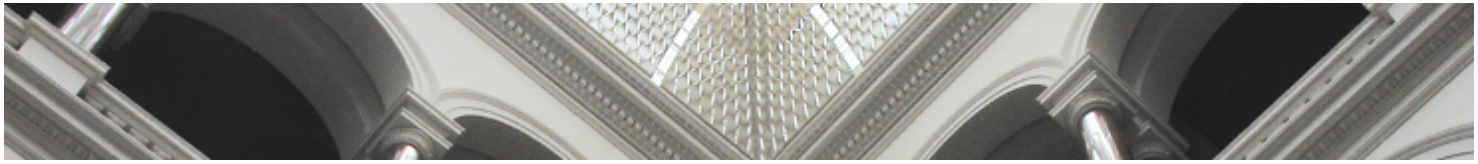
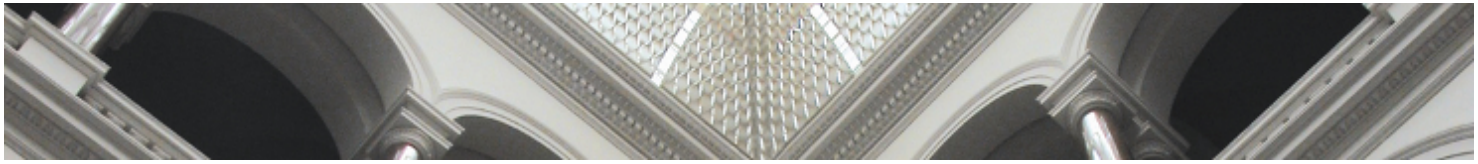


Figure: Solution f for **H** and **R** in x and v.



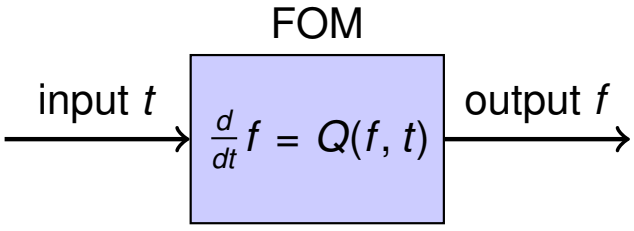
Technische Universität Berlin

Table of Contents

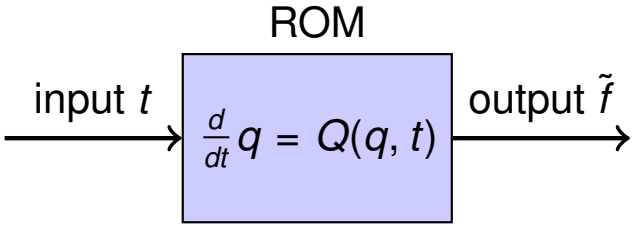


Model Order Reduction

– **Goal**: Reduce computational cost

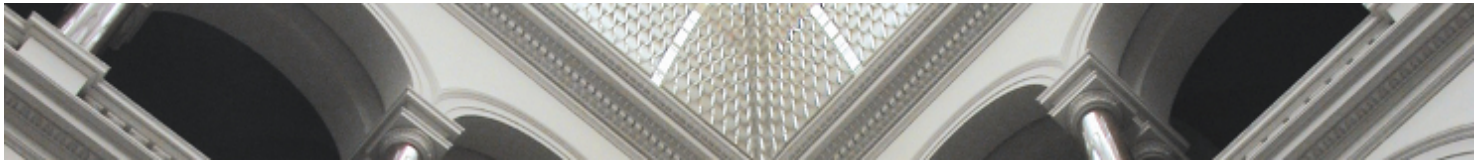


(a) Evolving the FOM in time.

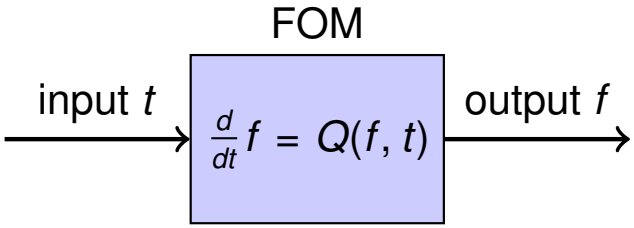


(b) Evolving the ROM in time.

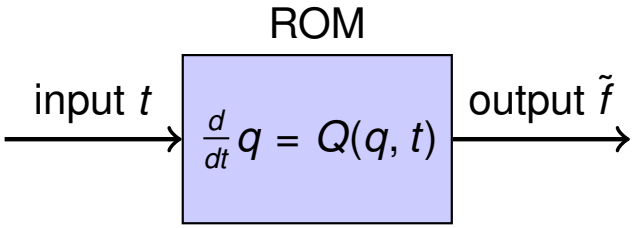
Figure: In the online phase the operator Q is different for the FOM and the ROM.



Model Order Reduction



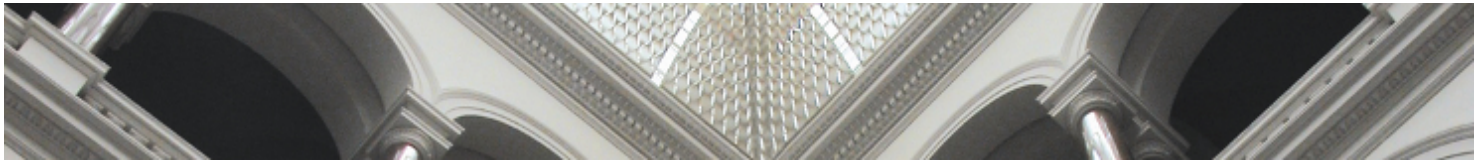
(a) Evolving the FOM in time.



(b) Evolving the ROM in time.

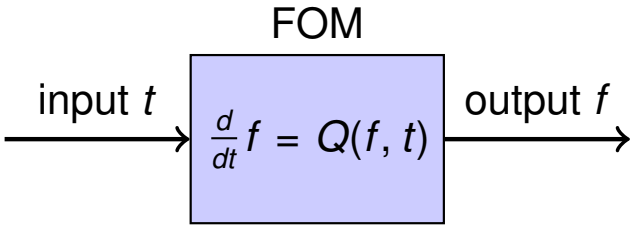
Figure: In the online phase the operator Q is different for the FOM and the ROM.

- **Goal**: Reduce computational cost
  - $f(x, v, t)$ with KJ ODE's in time for 1D

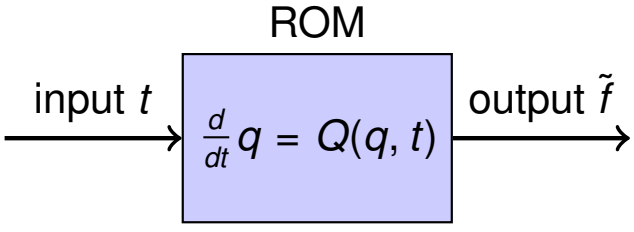


Technische Universität Berlin

Model Order Reduction



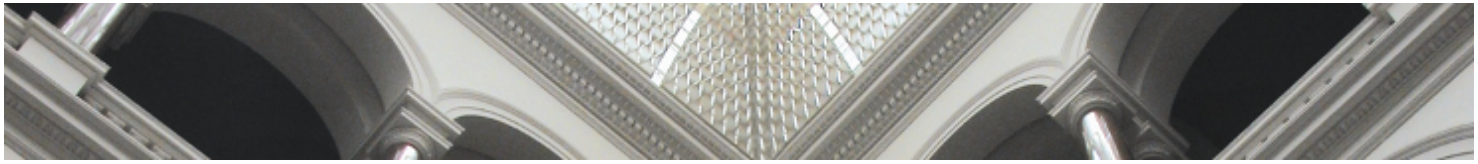
(a) Evolving the FOM in time.



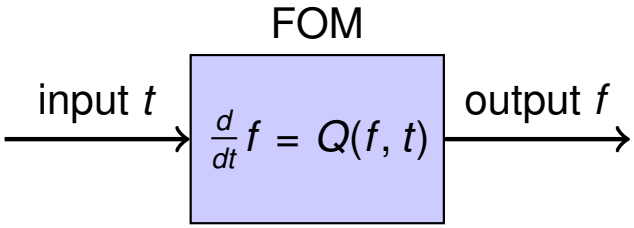
(b) Evolving the ROM in time.

Figure: In the online phase the operator Q is different for the FOM and the ROM.

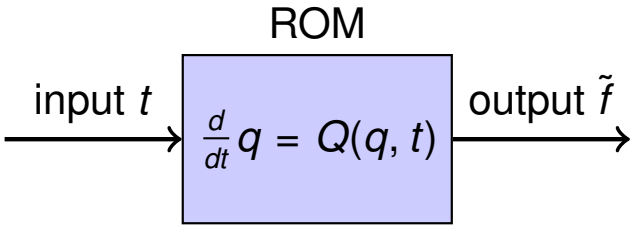
- **Goal**: Reduce computational cost
  - $f(x, v, t)$ with KJ ODE's in time for 1D
- **Require**: Reduction algorithm



Model Order Reduction



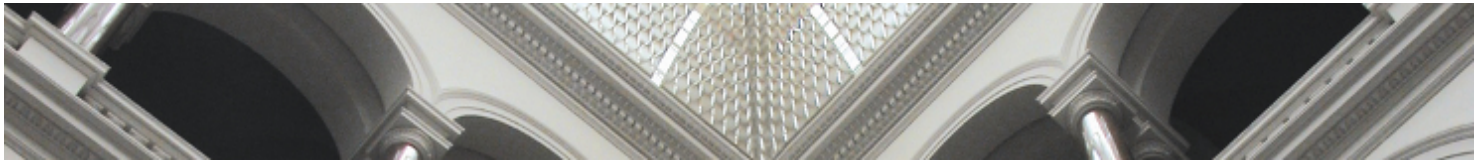
(a) Evolving the FOM in time.



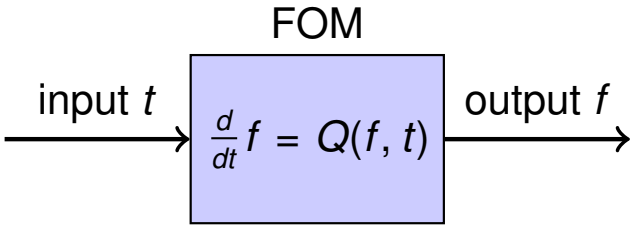
(b) Evolving the ROM in time.

Figure: In the online phase the operator Q is different for the FOM and the ROM.

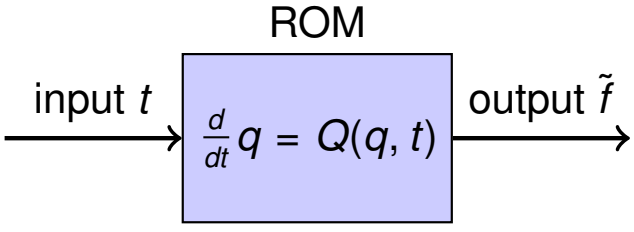
- **Goal**: Reduce computational cost
  - $f(x, v, t)$ with KJ ODE's in time for 1D
- **Require**: Reduction algorithm
  - Proper Orthogonal Decomposition **(POD)**
  - Neural Networks **(NN)**



Model Order Reduction



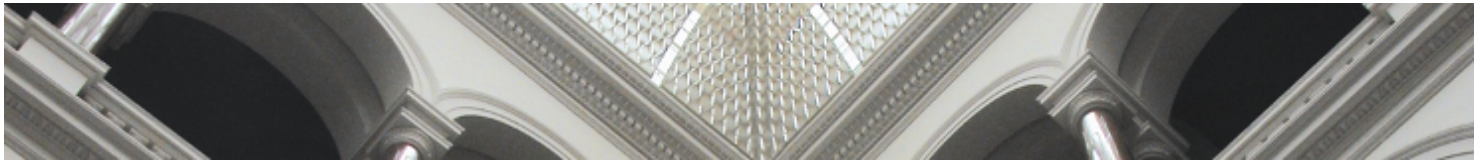
(a) Evolving the FOM in time.



(b) Evolving the ROM in time.

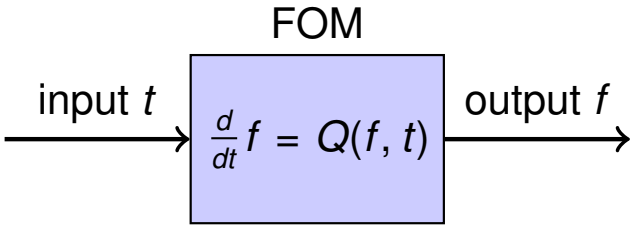
Figure: In the online phase the operator Q is different for the FOM and the ROM.

- **Goal**: Reduce computational cost
  - $f(x, v, t)$ with KJ ODE's in time for 1D
- **Require**: Reduction algorithm
  - Proper Orthogonal Decomposition **(POD)**
  - Neural Networks **(NN)**
- **Require**: Solution of f (only few timesteps)

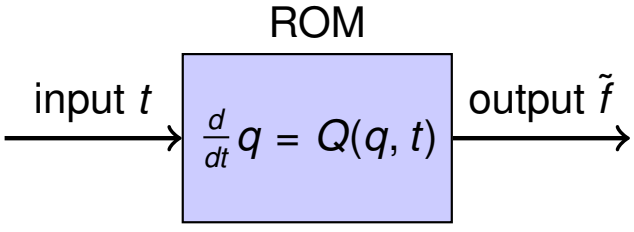


Technische Universität Berlin

Model Order Reduction



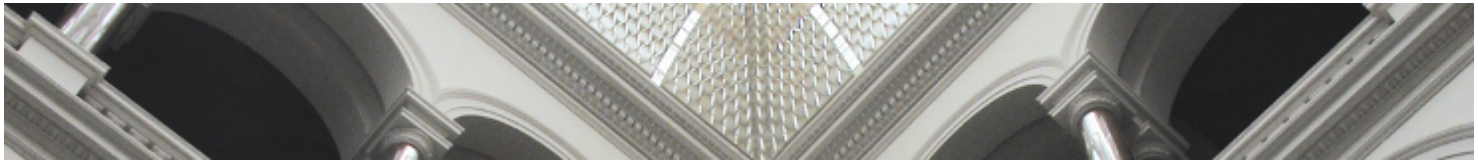
(a) Evolving the FOM in time.



(b) Evolving the ROM in time.

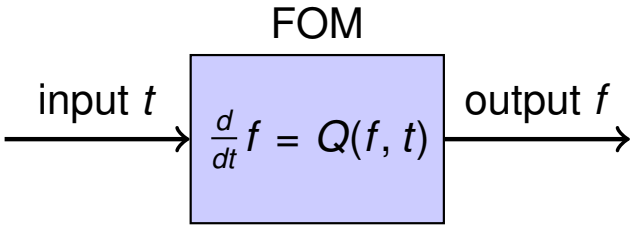
Figure: In the online phase the operator Q is different for the FOM and the ROM.

- **Goal**: Reduce computational cost
  - $f(x, v, t)$ with KJ ODE's in time for 1D
- **Require**: Reduction algorithm
  - Proper Orthogonal Decomposition **(POD)**
  - Neural Networks **(NN)**
- **Require**: Solution of f (only few timesteps)
- **Reduce**: $POD(f(x, v, t)) = q(x, n, t)$

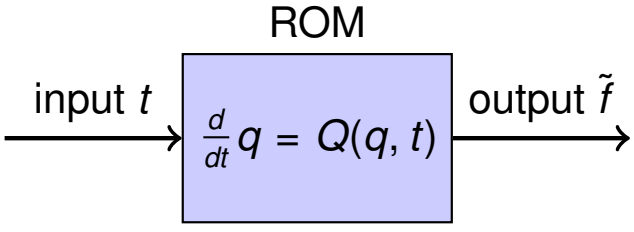


Technische Universität Berlin

Model Order Reduction



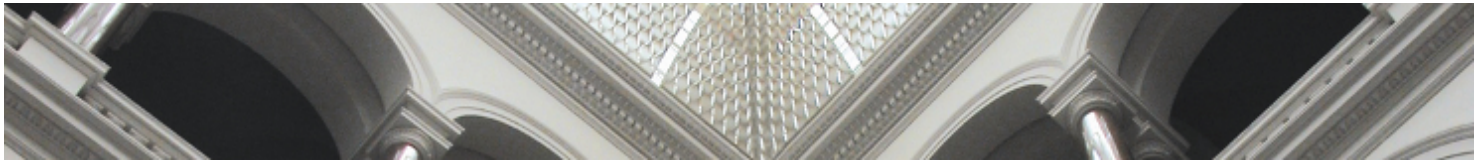
(a) Evolving the FOM in time.



(b) Evolving the ROM in time.

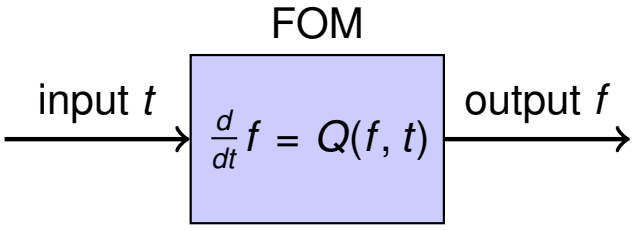
Figure: In the online phase the operator Q is different for the FOM and the ROM.

- **Goal**: Reduce computational cost
  - $f(x, v, t)$ with KJ ODE's in time for 1D
- **Require**: Reduction algorithm
  - Proper Orthogonal Decomposition **(POD)**
  - Neural Networks **(NN)**
- **Require**: Solution of f (only few timesteps)
- **Reduce**: $POD(f(x, v, t)) = q(x, n, t)$
  - P is # n with $P << K$

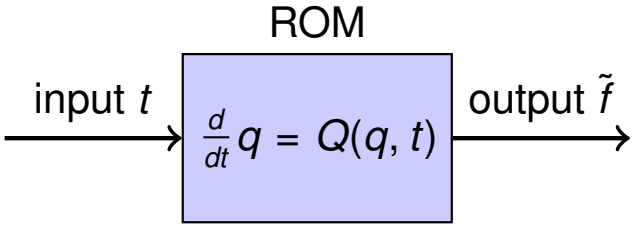


Technische Universität Berlin

Model Order Reduction



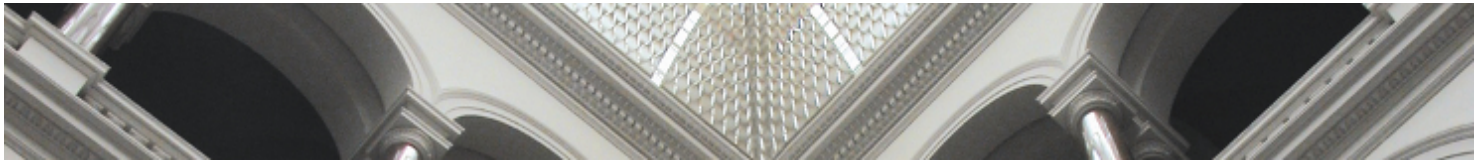
(a) Evolving the FOM in time.



(b) Evolving the ROM in time.

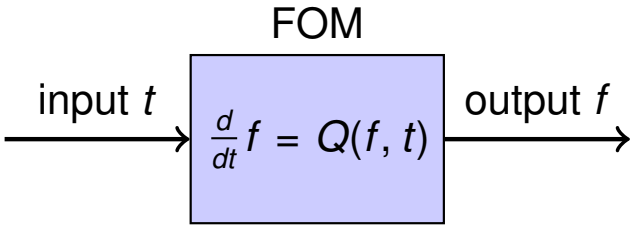
Figure: In the online phase the operator Q is different for the FOM and the ROM.

- **Goal**: Reduce computational cost
  - $f(x, v, t)$ with KJ ODE's in time for 1D
- **Require**: Reduction algorithm
  - Proper Orthogonal Decomposition **(POD)**
  - Neural Networks **(NN)**
- **Require**: Solution of f (only few timesteps)
- **Reduce**: $POD(f(x, v, t)) = q(x, n, t)$
  - P is # n with $P << K$
  - KJ ODE's vs. PJ ODE's

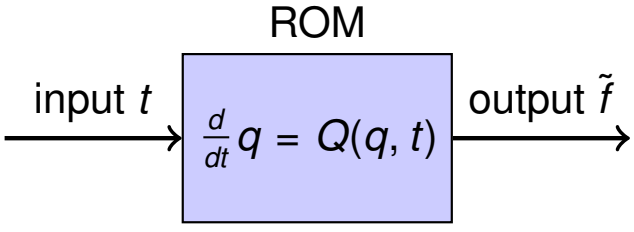


Technische Universität Berlin

Model Order Reduction



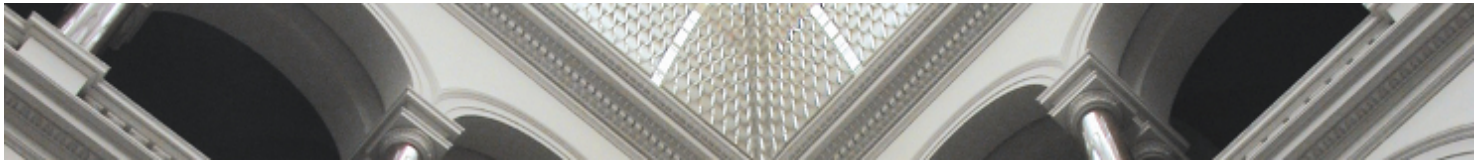
(a) Evolving the FOM in time.



(b) Evolving the ROM in time.

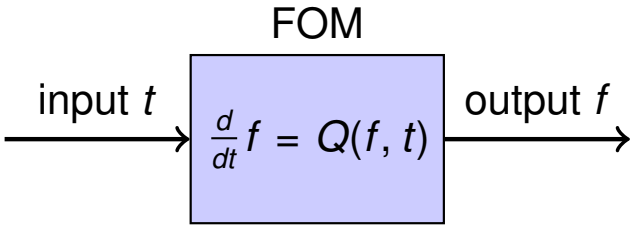
Figure: In the online phase the operator Q is different for the FOM and the ROM.

- **Goal**: Reduce computational cost
  - $f(x, v, t)$ with KJ ODE's in time for 1D
- **Require**: Reduction algorithm
  - Proper Orthogonal Decomposition **(POD)**
  - Neural Networks **(NN)**
- **Require**: Solution of f (only few timesteps)
- **Reduce**: $POD(f(x, v, t)) = q(x, n, t)$
  - P is # n with $P << K$
  - KJ ODE's vs. PJ ODE's
- **Evolve in time** $\longrightarrow$ $Q(q, t) = \tilde{f}$

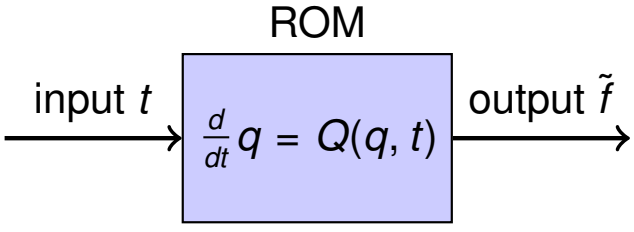


Technische Universität Berlin

Model Order Reduction



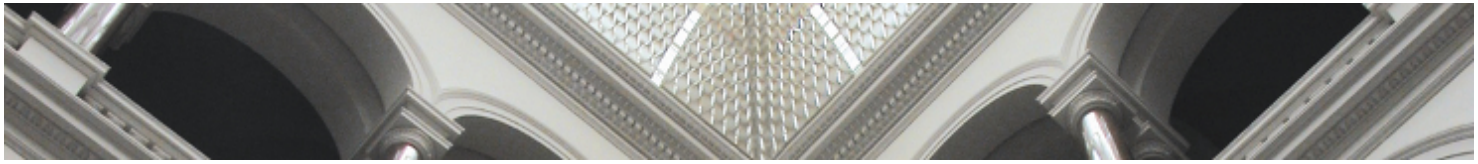
(a) Evolving the FOM in time.



(b) Evolving the ROM in time.

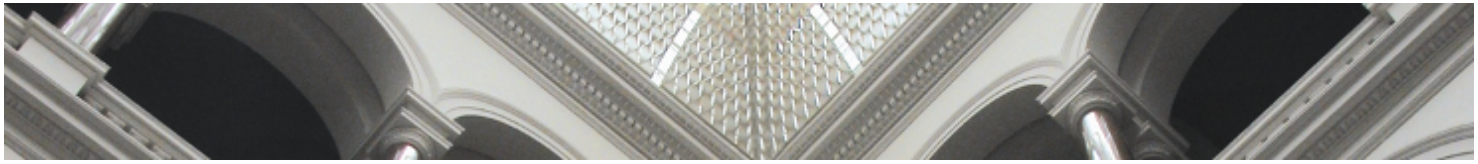
Figure: In the online phase the operator Q is different for the FOM and the ROM.

- **Goal**: Reduce computational cost
  - $f(x, v, t)$ with KJ ODE's in time for 1D
- **Require**: Reduction algorithm
  - Proper Orthogonal Decomposition **(POD)**
  - Neural Networks **(NN)**
- **Require**: Solution of f (only few timesteps)
- **Reduce**: $POD(f(x, v, t)) = q(x, n, t)$
  - P is # n with $P << K$
  - KJ ODE's vs. PJ ODE's
- **Evolve in time** $\longrightarrow Q(q, t) = \tilde{f}$
- $f - \tilde{f} < \epsilon$



Technische Universität Berlin

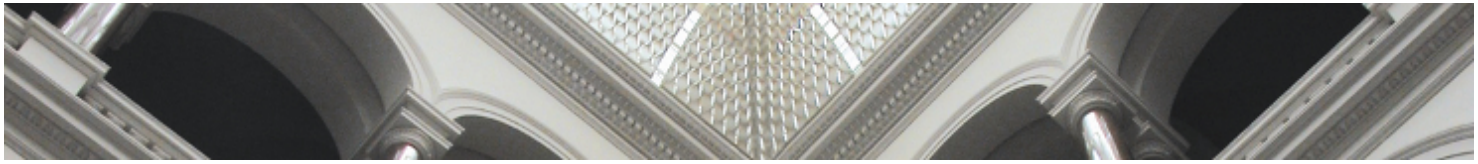
Table of Contents



Technische Universität Berlin

Proper Orthogonal Decomposition

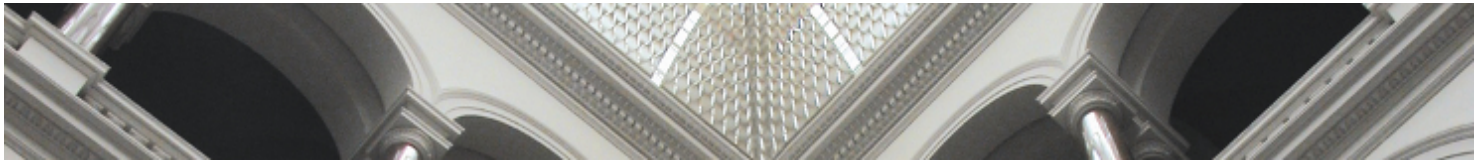
– Solution of a PDE is $f(x, v, t)$ can be obtained



Technische Universität Berlin

Proper Orthogonal Decomposition

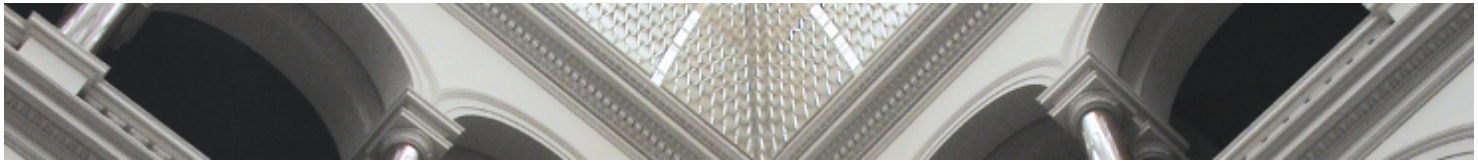
- Solution of a PDE is $f(x, v, t)$ can be obtained
  - Discretization into a system of ODE's



Technische Universität Berlin

Proper Orthogonal Decomposition

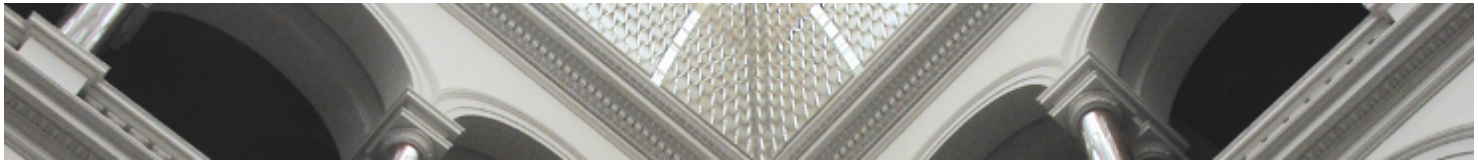
- Solution of a PDE is $f(x, v, t)$ can be obtained
  - Discretization into a system of ODE's
  - Separation of variables ansatz



Technische Universität Berlin

Proper Orthogonal Decomposition

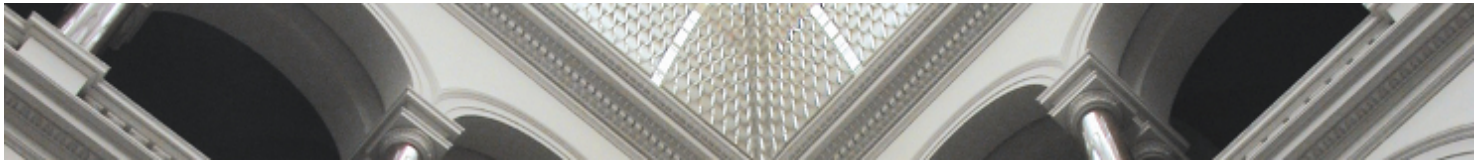
- Solution of a PDE is $f(x, v, t)$ can be obtained
  - Discretization into a system of ODE's
  - Separation of variables ansatz
    - » $f(t, v, x) = \sum_{i=1}^{n} a_i(t)\, \Phi_i(x, v)$ (9)



Technische Universität Berlin

Proper Orthogonal Decomposition

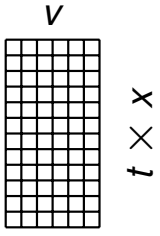
– How to get Φ_i?

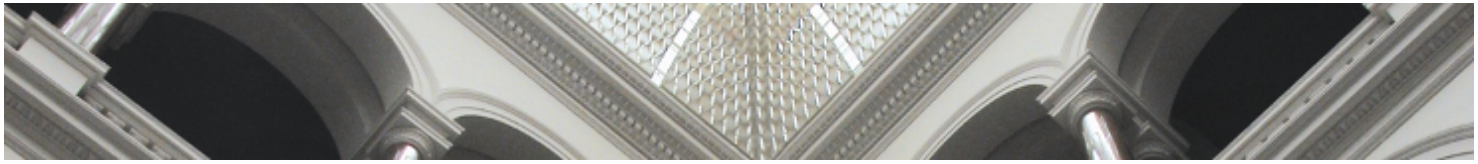


Proper Orthogonal Decomposition

– How to get Φ_i?

- **Preprocessing**: Seperating the spatial and temporal axis of the solution $f(x, v, t)$

» $P =$ 

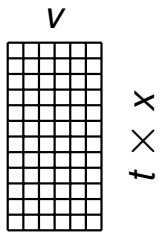


Technische Universität Berlin

Proper Orthogonal Decomposition

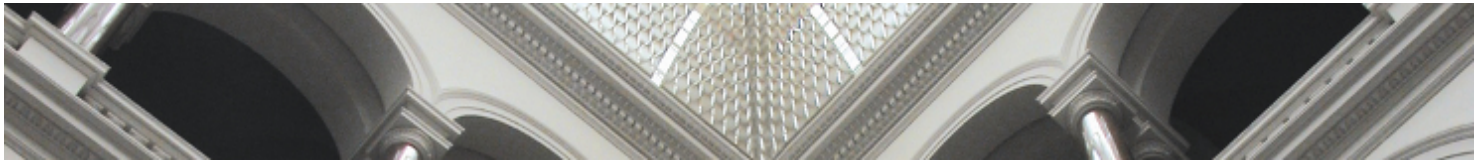
– How to get Φ_i?

- **Preprocessing**: Seperating the spatial and temporal axis of the solution $f(x, v, t)$

» $P =$ 

- **Sigular Value Decomposition**: $X = \underbrace{U}_{\text{LSV}} \underbrace{\Sigma}_{\text{SV}} \underbrace{V^*}_{\text{RSV}}$ (9)

- **Truncation**: $\tilde{X} = \underbrace{\tilde{U}}_{\Phi} \tilde{\Sigma} \tilde{V}^*$ (10), $\tilde{U} = \Phi = [\Phi_1, \Phi_2, \dots, \Phi_p]$ (11)

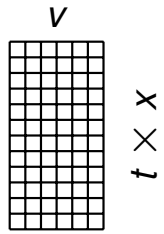


Technische Universität Berlin

Proper Orthogonal Decomposition

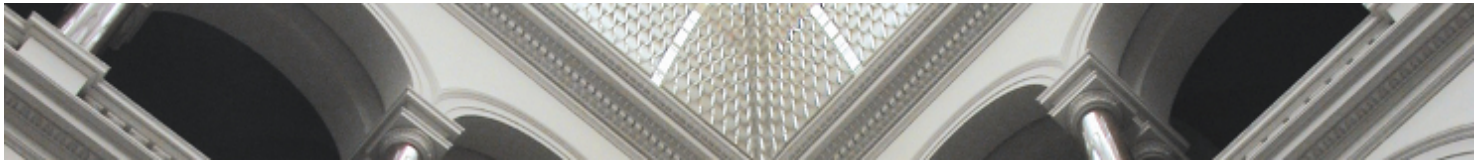
– How to get Φ_i?

- **Preprocessing**: Seperating the spatial and temporal axis of the solution $f(x, v, t)$

» $P =$ 

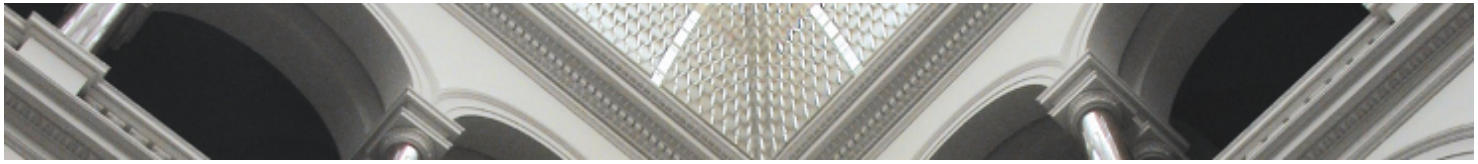
- **Sigular Value Decomposition**: $X = \underbrace{U}_{\text{LSV}} \underbrace{\Sigma}_{\text{SV}} \underbrace{V^*}_{\text{RSV}}$ (9)
- **Truncation**: $\tilde{X} = \underbrace{\tilde{U}}_{\Phi} \tilde{\Sigma} \tilde{V}^*$ (10), $\tilde{U} = \Phi = [\Phi_1, \Phi_2, \ldots, \Phi_p]$ (11)
- **Eckard-Young Theorem**:

$$\underset{\tilde{X}, s.t. rank(\tilde{X})=r}{\operatorname{argmin}} \left\| X - \tilde{X} \right\|_F = \tilde{U}\tilde{\Sigma}\tilde{V}^* \tag{12}$$



Technische Universität Berlin

Table of Contents



Technische Universität Berlin

Terminology

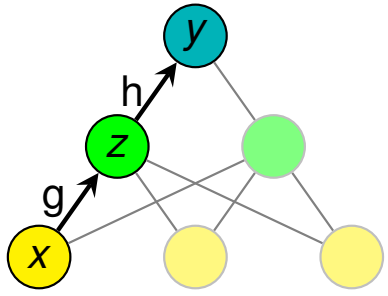
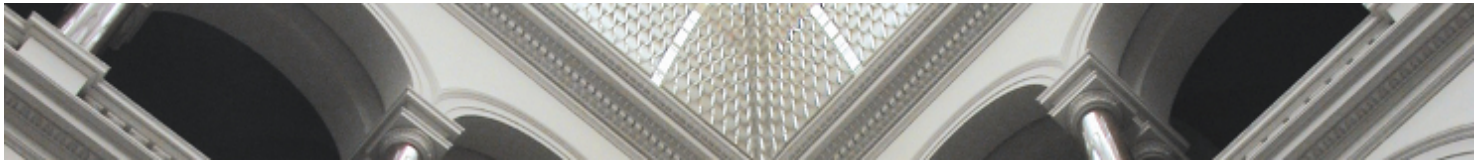


Figure: Example of a simple network.

– Network with three layers



Technische Universität Berlin

Terminology

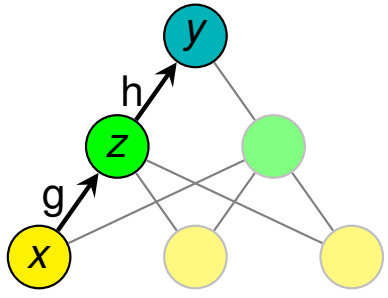
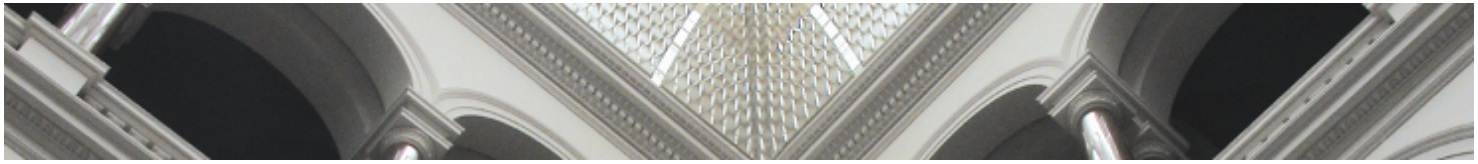


Figure: Example of a simple network.

– Network with three layers
  - Input layer
  - Hidden layer
  - Output layer



Terminology

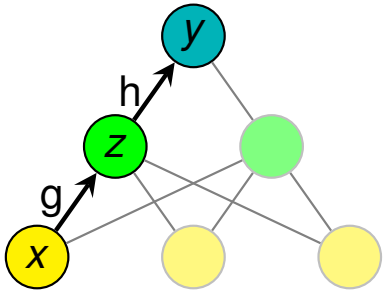
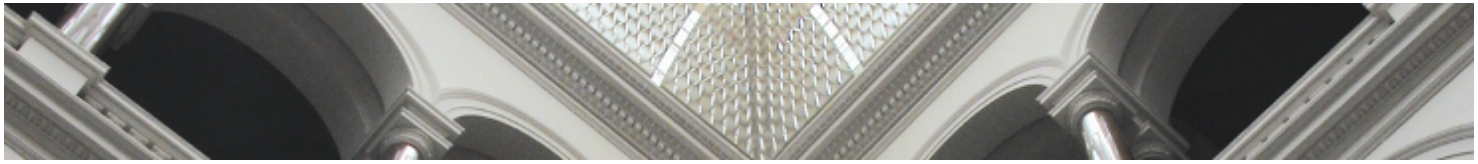


Figure: Example of a simple network.

- **Layer**: Stage of computation

- Network with three layers
  - Input layer
  - Hidden layer
  - Output layer



Technische Universität Berlin

Terminology

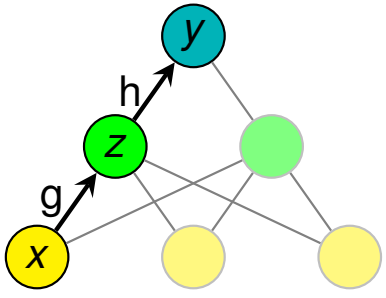
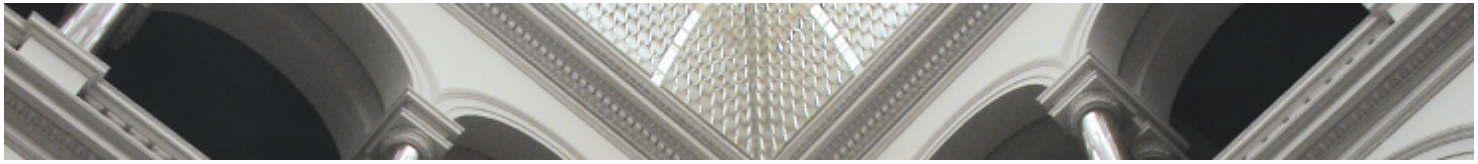


Figure: Example of a simple network.

- Network with three layers
  - Input layer
  - Hidden layer
  - Output layer

- **Layer**: Stage of computation
- **Computations**/ Forward pass
  - $g(x) = g(xW + b) = z$
  - $h(z) = h(zW + b) = y$
    - » $h(g(x)) = y$



Technische Universität Berlin

Terminology

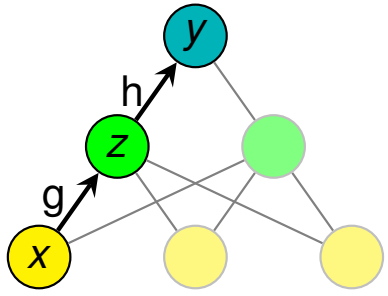
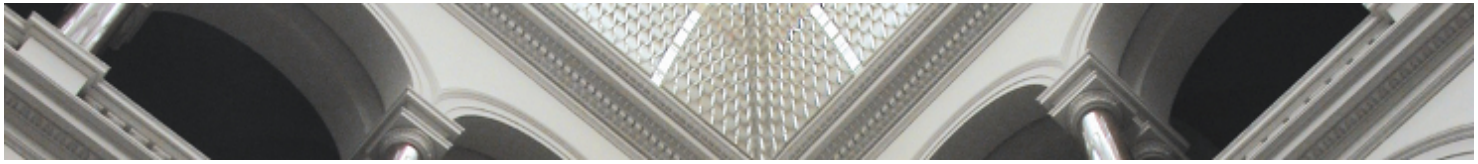


Figure: Example of a simple network.

- Network with three layers
  - Input layer
  - Hidden layer
  - Output layer

- **Layer**: Stage of computation
- **Computations**/ Forward pass
  - $g(x) = g(xW + b) = z$
  - $h(z) = h(zW + b) = y$
    - » $h(g(x)) = y$
- **Neuron**: Entry in 'Tensor'



Technische Universität Berlin

Terminology

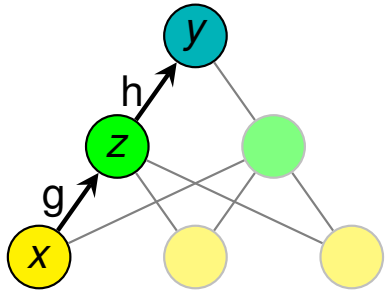
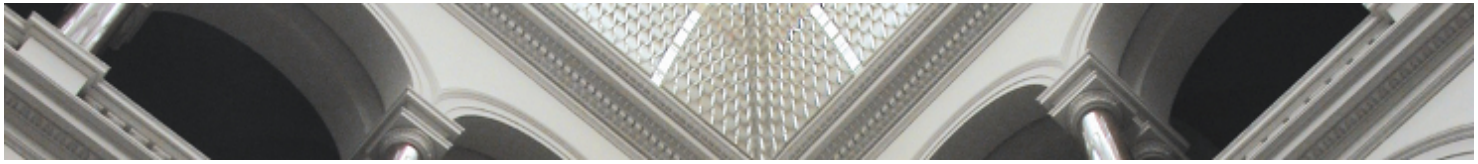


Figure: Example of a simple network.

- Network with three layers
  - Input layer
  - Hidden layer
  - Output layer

- **Layer**: Stage of computation
- **Computations**/ Forward pass
  - $g(x) = g(xW + b) = z$
  - $h(z) = h(zW + b) = y$
    - » $h(g(x)) = y$
- **Neuron**: Entry in 'Tensor'
- **Trainable parameters**: W, b



Technische Universität Berlin

Autoencoders

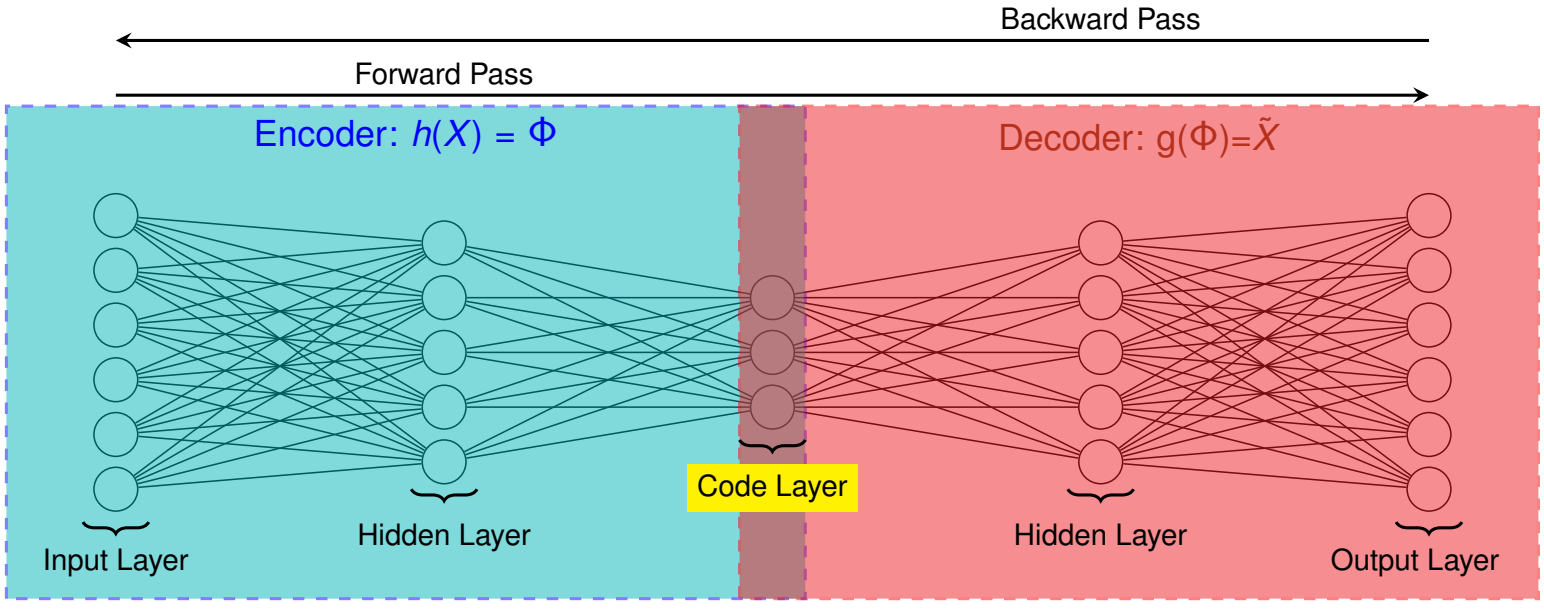
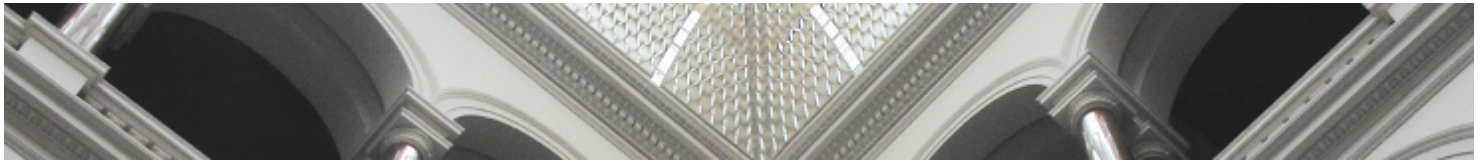


Figure: A fully connected autoencoder.



Technische Universität Berlin

Autoencoders

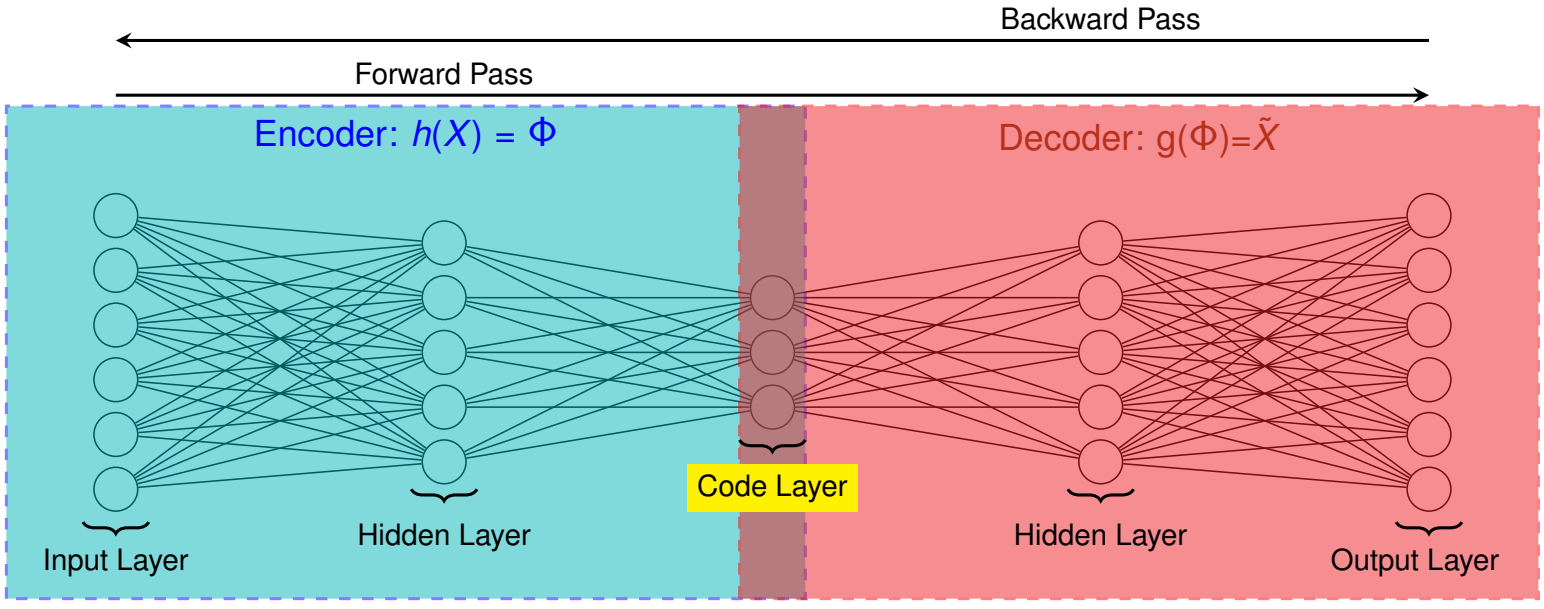
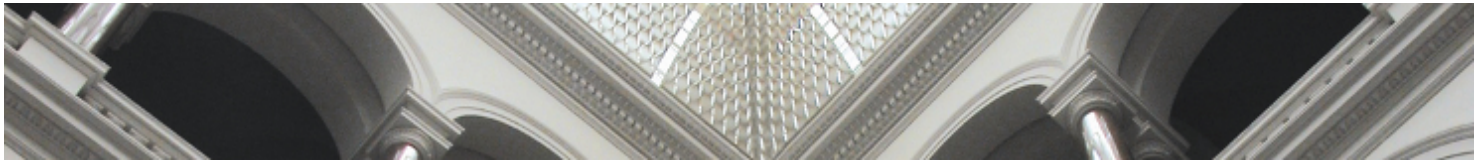
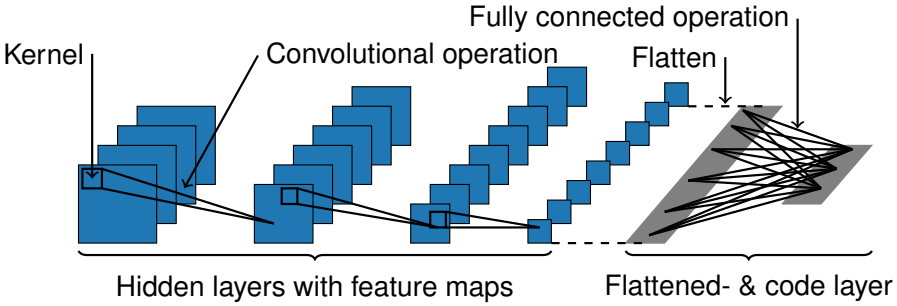


Figure: A fully connected autoencoder.

- **Structure**: Encoder & Decoder
- **Layers**: Input-, Output- **and** Code layer
- **Category**: Self-supervised learning
- **Main hyperparameters**:
  Number & size of hidden layers
  esp. size of code layer



Convolutional Autoencoder

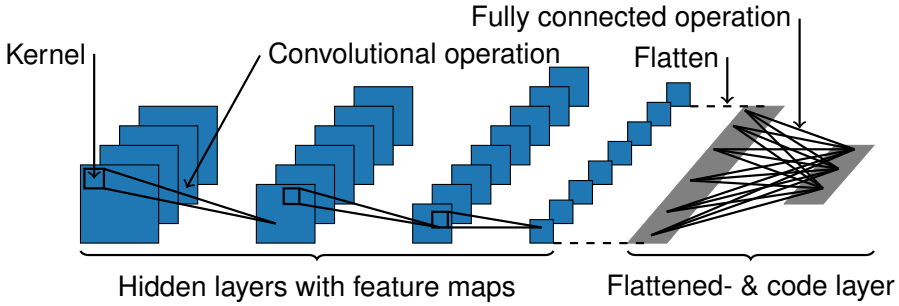


(a) Encoder of a convolutional autoencoder without input layer.

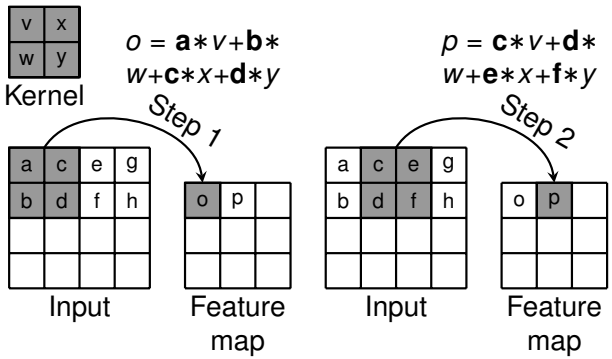
Figure: Fundamental features of conv. networks.

– Designed for 2D/3D input

Convolutional Autoencoder



(a) Encoder of a convolutional autoencoder without input layer.

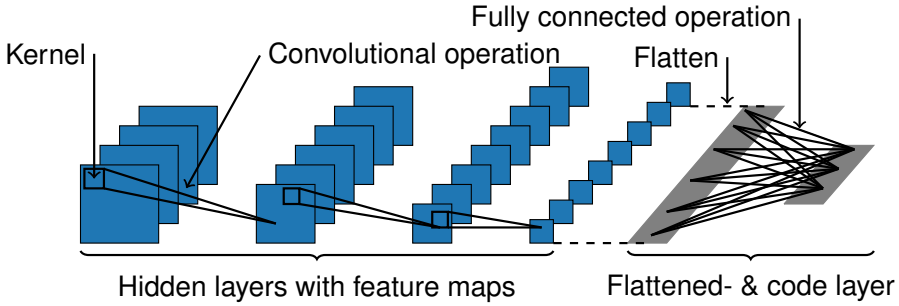


(b) Convolutional operation, 1 strided.

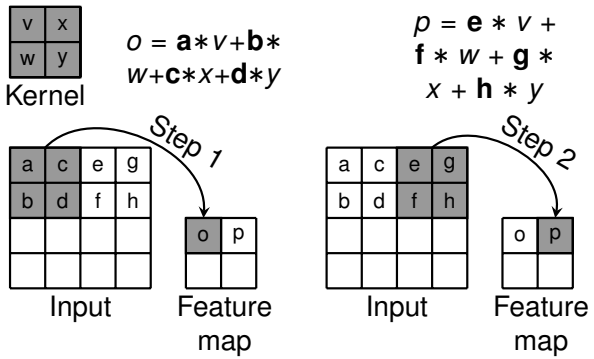
Figure: Fundamental features of conv. networks.

– Designed for 2D/3D input

Convolutional Autoencoder



(a) Encoder of a convolutional autoencoder without input layer.

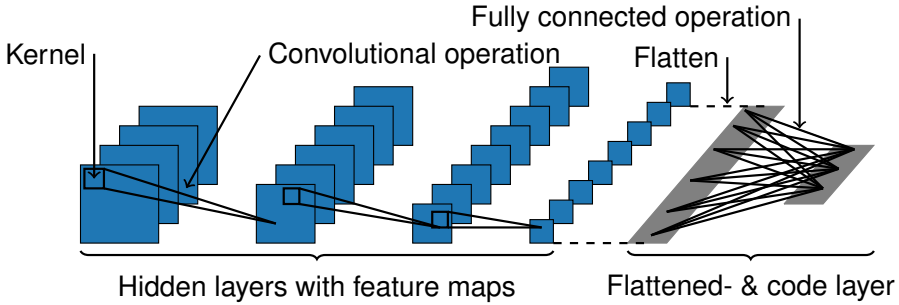


(b) Convolutional operation, 2 strided.

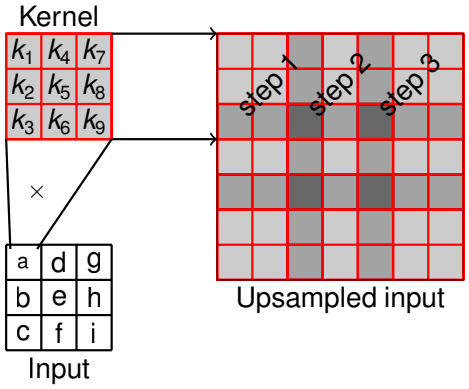
Figure: Fundamental features of conv. networks.

– Designed for 2D/3D input

Convolutional Autoencoder



(a) Encoder of a convolutional autoencoder without input layer.

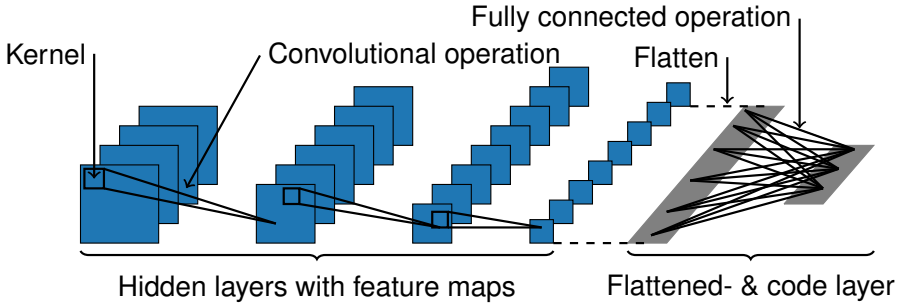


(b) Even deconvolution

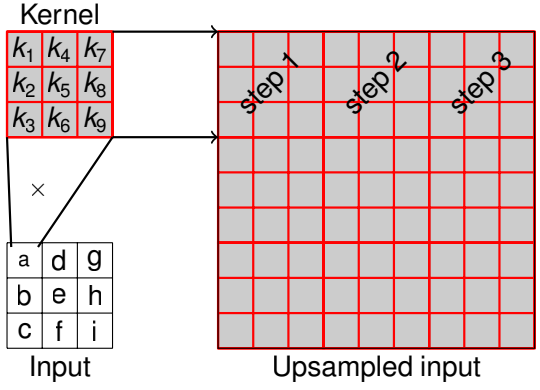
Figure: Fundamental features of conv. networks.

– Designed for 2D/3D input

Convolutional Autoencoder



(a) Encoder of a convolutional autoencoder without input layer.

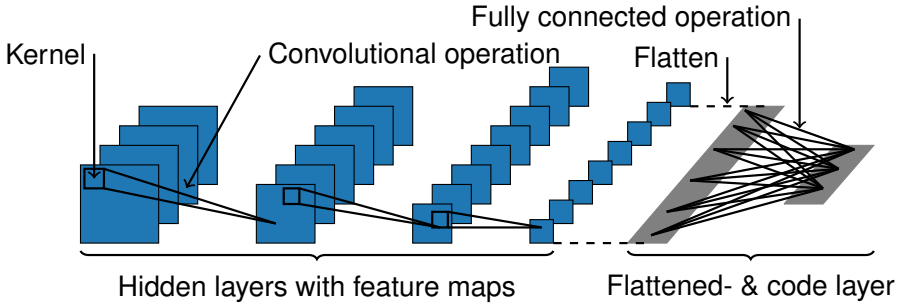


(b) Uneven deconvolution

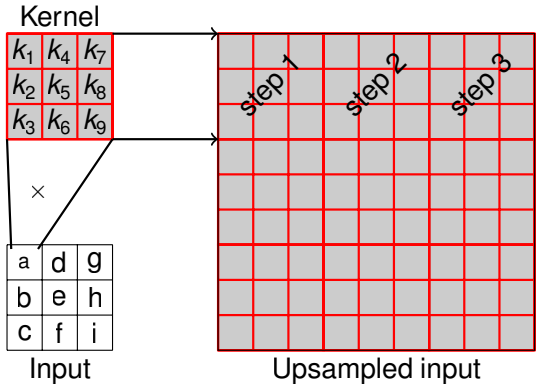
Figure: Fundamental features of conv. networks.

– Designed for 2D/3D input

Convolutional Autoencoder



(a) Encoder of a convolutional autoencoder without input layer.

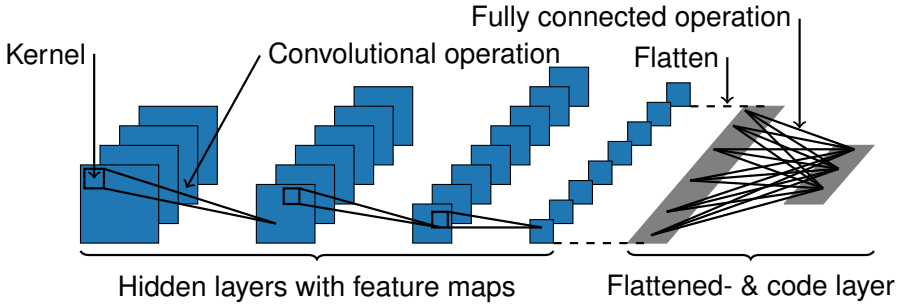


(b) Uneven deconvolution

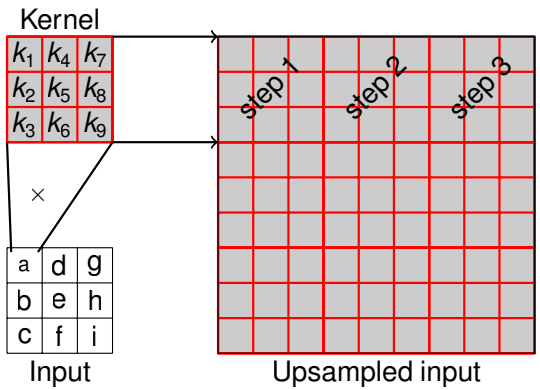
Figure: Fundamental features of conv. networks.

- Designed for 2D/3D input
- **Peculiarities**: Sparse connections, parameter sharing
  - Promotes generalization

Convolutional Autoencoder



(a) Encoder of a convolutional autoencoder without input layer.

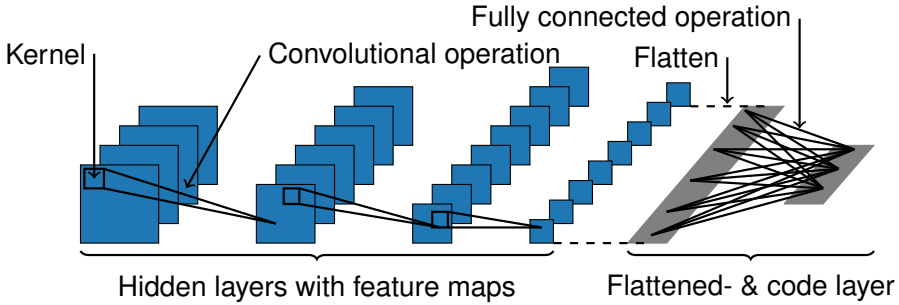


(b) Uneven deconvolution

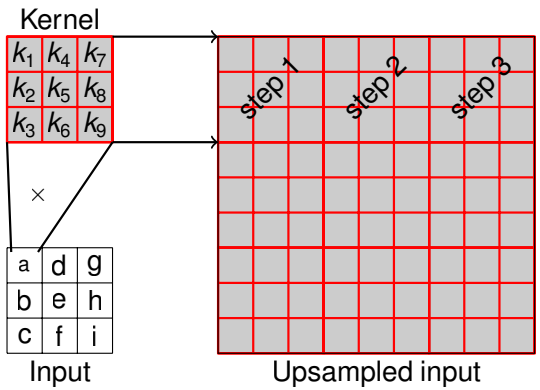
Figure: Fundamental features of conv. networks.

- Designed for 2D/3D input
- **Peculiarities**: Sparse connections, parameter sharing
  - Promotes generalization
- **Hyperparameters**: Number & size of layers, kernel dimensions, stride increments
  - Non-trivial influence of output dimensions & quality

Convolutional Autoencoder



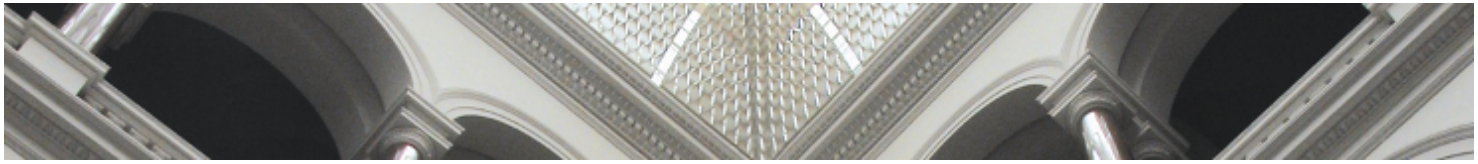
(a) Encoder of a convolutional autoencoder without input layer.



(b) Uneven deconvolution

Figure: Fundamental features of conv. networks.

- Designed for 2D/3D input
- **Peculiarities**: Sparse connections, parameter sharing
  - Promotes generalization
- **Hyperparameters**: Number & size of layers, kernel dimensions, stride increments
  - Non-trivial influence of output dimensions & quality



Technische Universität Berlin

Training

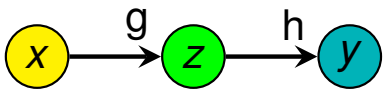
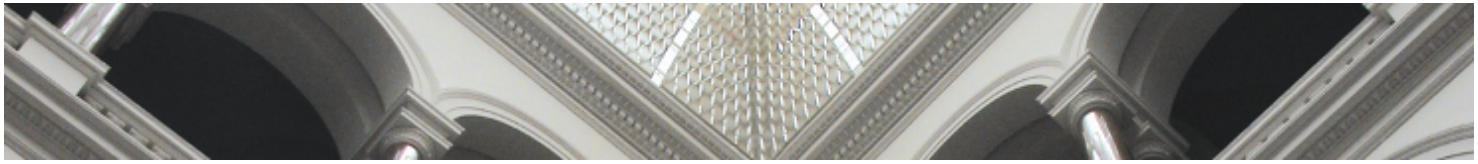


Figure: A very simple network

– **Forward propagation**:

$$y = h(z, b) = h((g(x, a)), b) \qquad (13)$$



Technische Universität Berlin

Training

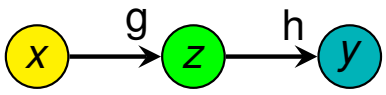


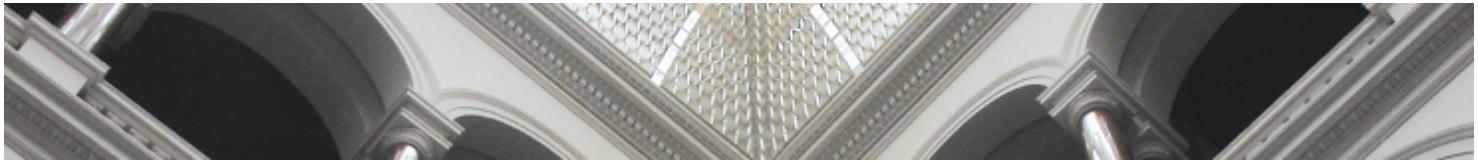
Figure: A very simple network

- **Forward propagation**:

$$y = h(z, b) = h((g(x, a)), b) \tag{13}$$

- **Loss function**:

$$L(y, \tilde{y}_0) = \frac{1}{2}(y - \tilde{y}_0)^2 = E \tag{14}$$



Technische Universität Berlin

Training

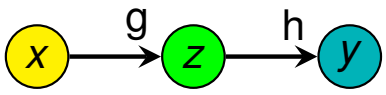


Figure: A very simple network

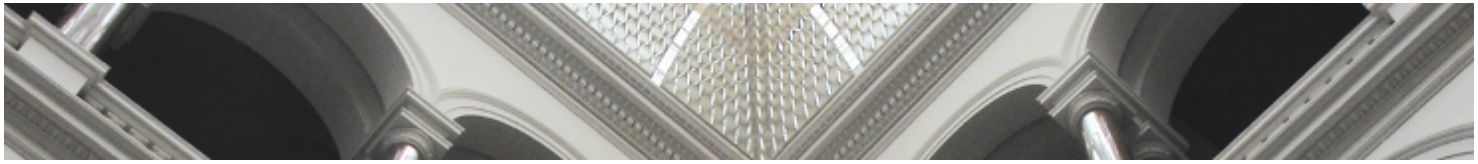
- **Forward propagation**:

$$y = h(z, b) = h((g(x, a)), b) \qquad (13)$$

- **Loss function**:

$$L(y, \tilde{y}_0) = \frac{1}{2}(y - \tilde{y}_0)^2 = E \qquad (14)$$

- **Backpropagation**:



Technische Universität Berlin

Training

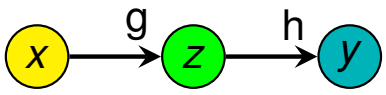


Figure: A very simple network

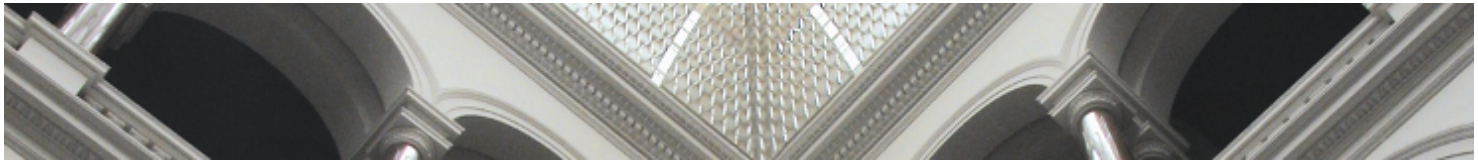
- **Forward propagation**:

$$y = h(z, b) = h((g(x, a)), b) \tag{13}$$

- **Loss function**:

$$L(y, \tilde{y_0}) = \frac{1}{2}(y - \tilde{y_0})^2 = E \tag{14}$$

- **Backpropagation**:
  - $\frac{\partial E}{\partial a} = -(y_0 - y)\frac{\partial y}{\partial z}\frac{\partial z}{\partial a}$ (15)
  - $\frac{\partial E}{\partial b} = -(y_0 - y)\frac{\partial y}{\partial b}$ (16)



Technische Universität Berlin

Training

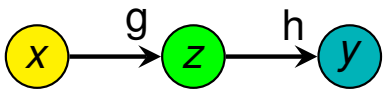


Figure: A very simple network

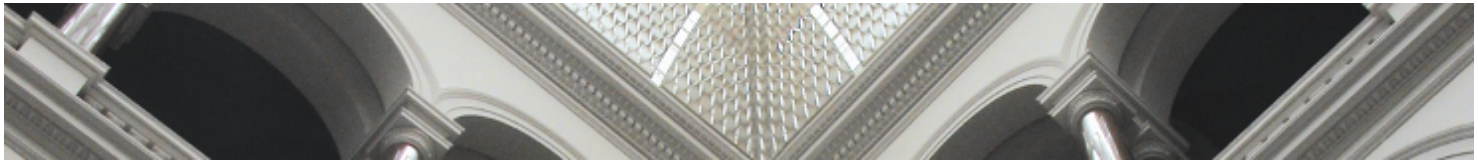
- **Forward propagation**:

$$y = h(z, b) = h((g(x, a)), b) \tag{13}$$

- **Loss function**:

$$L(y, \tilde{y_0}) = \frac{1}{2}(y - \tilde{y_0})^2 = E \tag{14}$$

- **Backpropagation**:
  - $\frac{\partial E}{\partial a} = -(y_0 - y)\frac{\partial y}{\partial z}\frac{\partial z}{\partial a}$ (15)
  - $\frac{\partial E}{\partial b} = -(y_0 - y)\frac{\partial y}{\partial b}$ (16)
- **Optimize**:
  - $a_{i+1} = a_i + \epsilon \frac{\partial E}{\partial a_i}$ (17)
  - $b_{i+1} = b_i + \epsilon \frac{\partial E}{\partial b_i}$ (18)



Technische Universität Berlin

Training

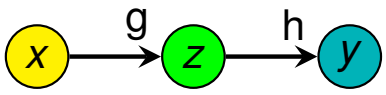


Figure: A very simple network

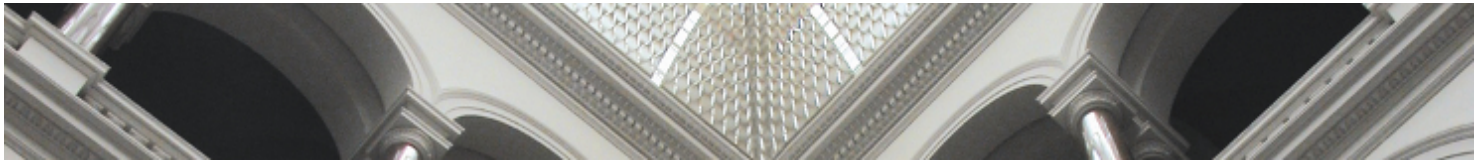
- **Forward propagation**:

$$y = h(z, b) = h((g(x, a)), b) \tag{13}$$

- **Loss function**:

$$L(y, \tilde{y_0}) = \frac{1}{2}(y - \tilde{y_0})^2 = E \tag{14}$$

- **Backpropagation**:
  - $\frac{\partial E}{\partial a} = -(y_0 - y)\frac{\partial y}{\partial z}\frac{\partial z}{\partial a}$ (15)
  - $\frac{\partial E}{\partial b} = -(y_0 - y)\frac{\partial y}{\partial b}$ (16)
- **Optimize**:
  - $a_{i+1} = a_i + \epsilon \frac{\partial E}{\partial a_i}$ (17)
  - $b_{i+1} = b_i + \epsilon \frac{\partial E}{\partial b_i}$ (18)
- **Hyperparameter**: learning rate ϵ



Concepts

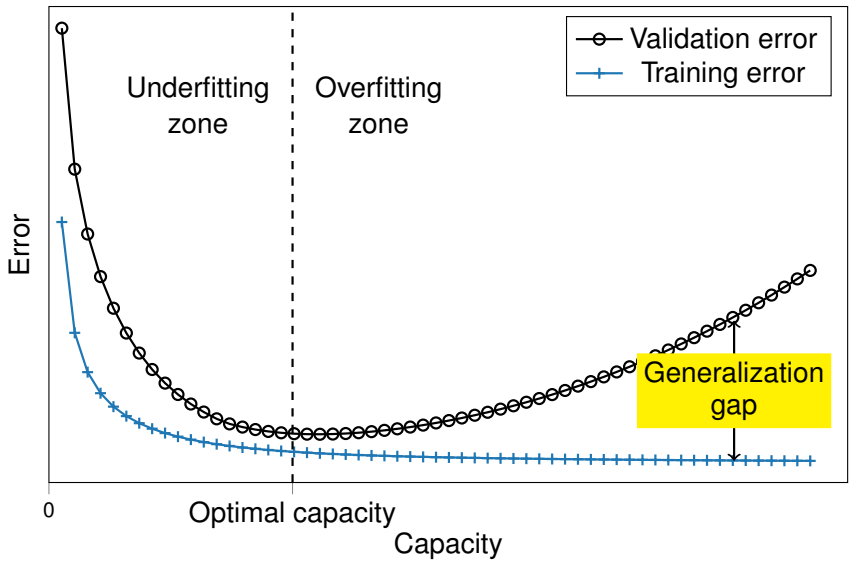
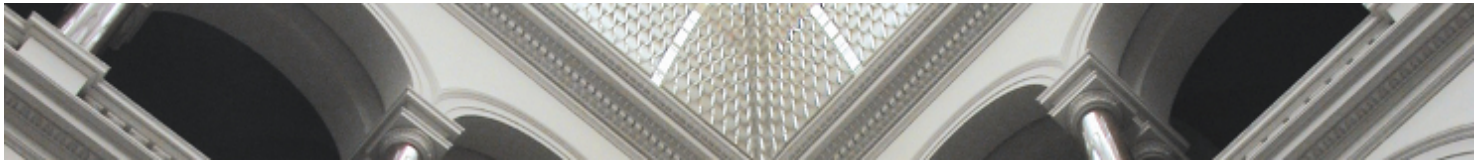


Figure: Influence of capacity

– **Over- and Underfitting**:



Concepts

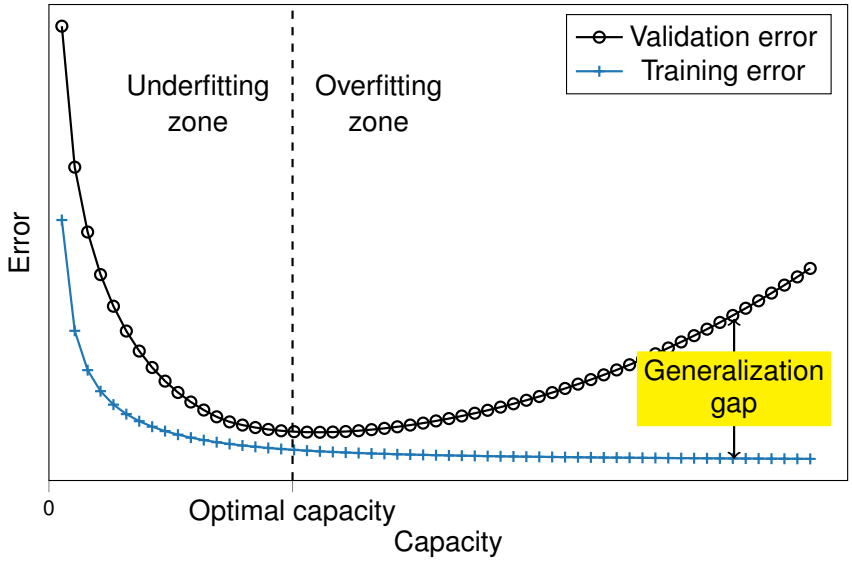
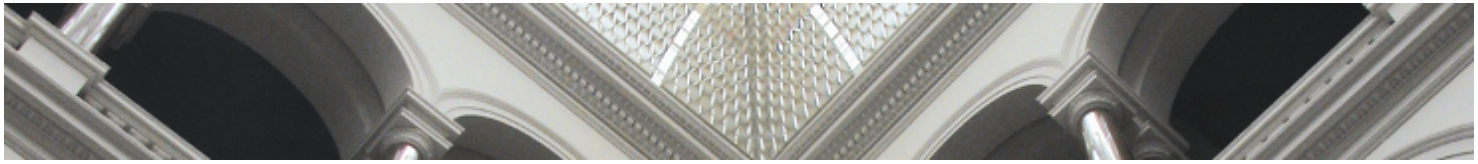


Figure: Influence of capacity

- **Over- and Underfitting**:
  - **Goal**: Reach optimal capacity



Concepts

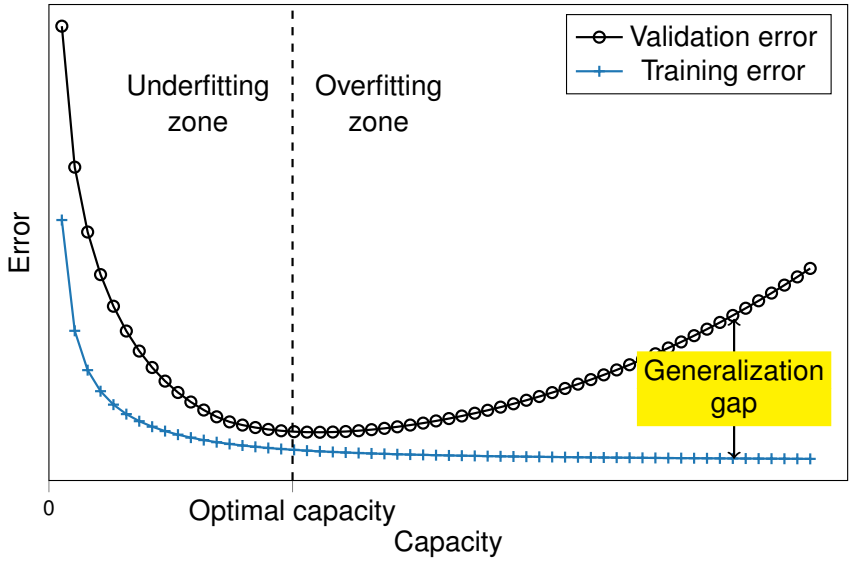
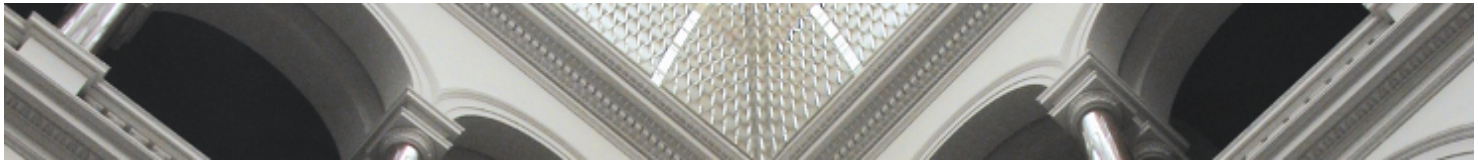


Figure: Influence of capacity

- **Over- and Underfitting**:
  - **Goal**: Reach optimal capacity
- **How to direct capacity ?**:



Technische Universität Berlin

Concepts

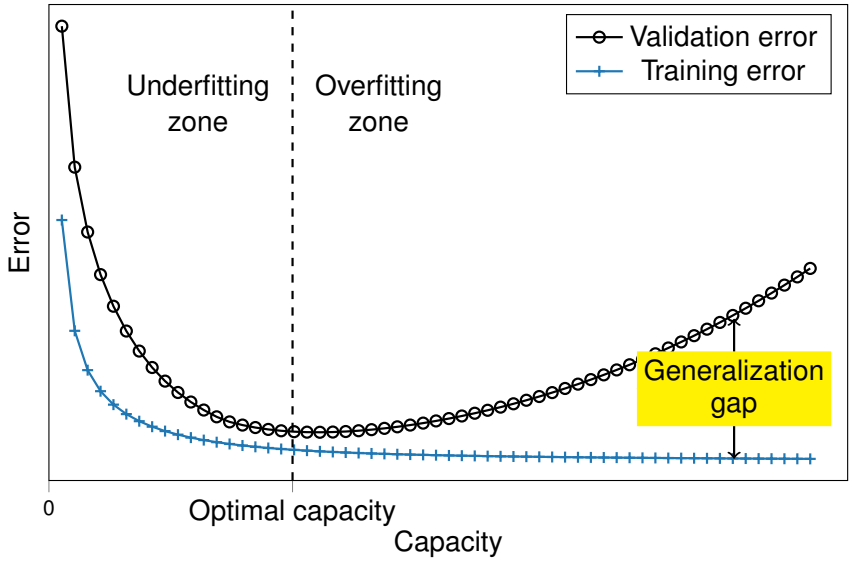
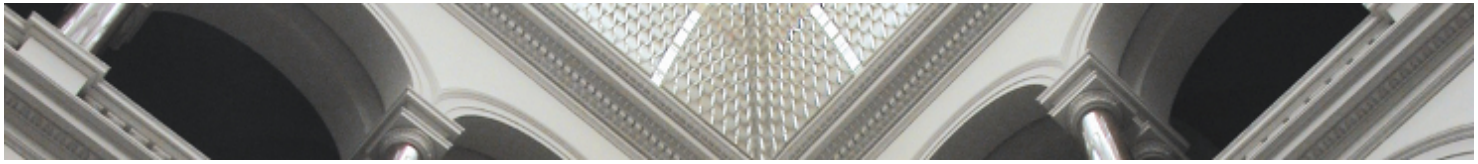


Figure: Influence of capacity

- **Over- and Underfitting**:
  - **Goal**: Reach optimal capacity
- **How to direct capacity ?**:
  - Size of the network



Technische Universität Berlin

Concepts

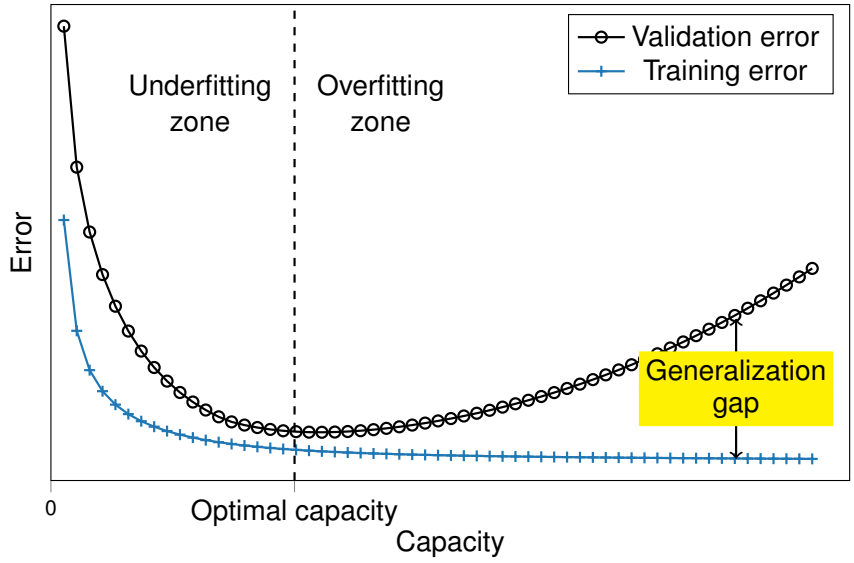
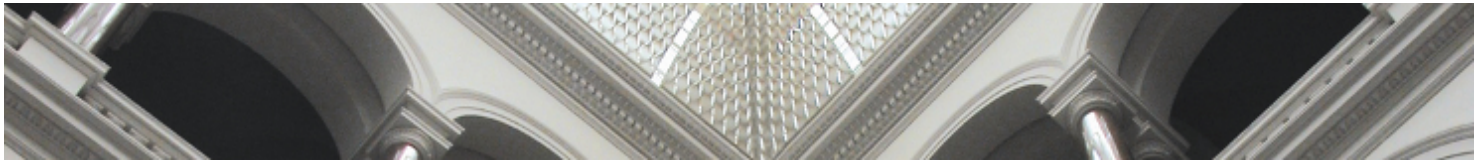


Figure: Influence of capacity

- **Over- and Underfitting**:
  - **Goal**: Reach optimal capacity
- **How to direct capacity ?**:
  - Size of the network
  - Activation functions

- **Initialization**



Concepts

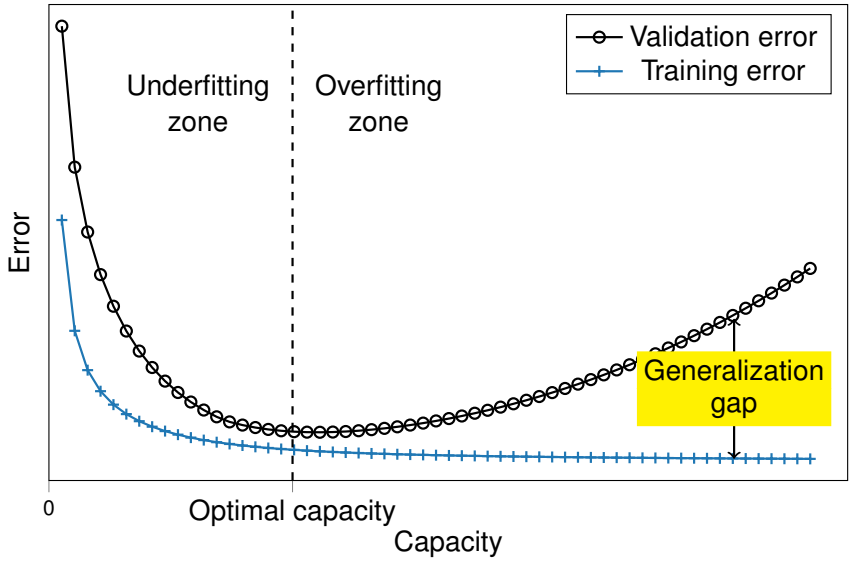
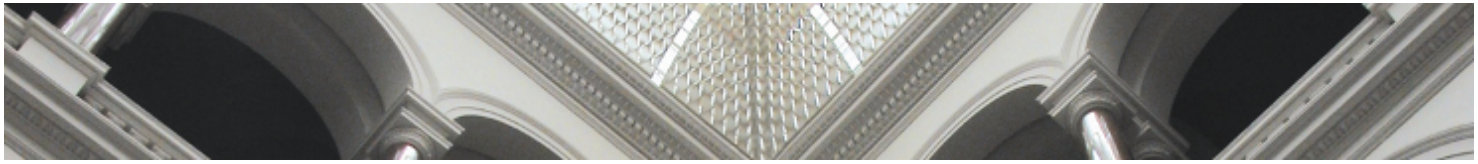


Figure: Influence of capacity

- **Over- and Underfitting**:
  - **Goal**: Reach optimal capacity
- **How to direct capacity ?**:
  - Size of the network
  - Activation functions
  - Loss function

- **Initialization**



Technische Universität Berlin

Concepts

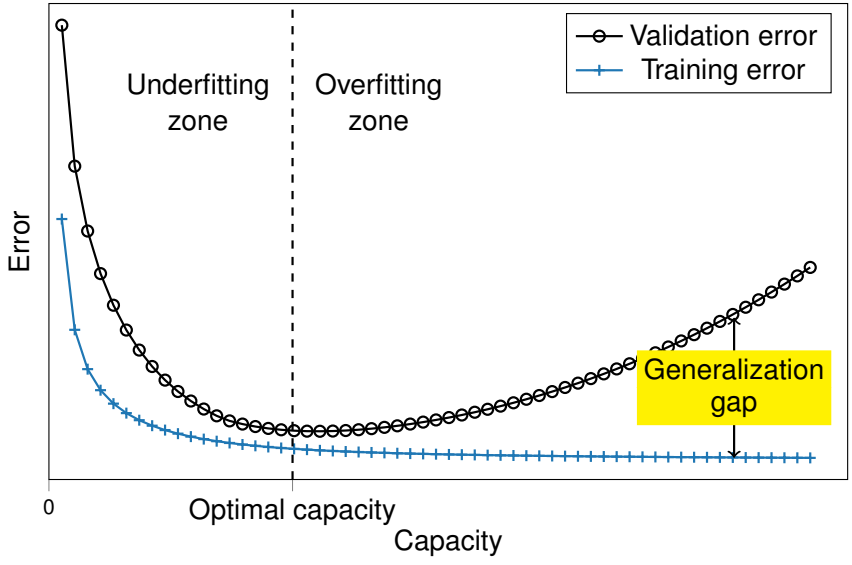
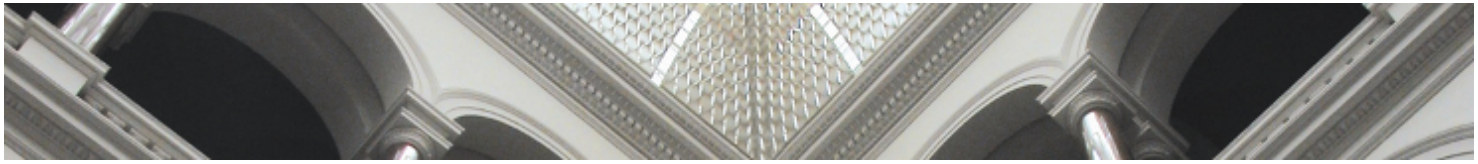


Figure: Influence of capacity

- **Over- and Underfitting**:
  - **Goal**: Reach optimal capacity
- **How to direct capacity ?**:
  - Size of the network
  - Activation functions
  - Loss function
  - Data distortion/ add variation to existing data
- **Initialization**



Concepts

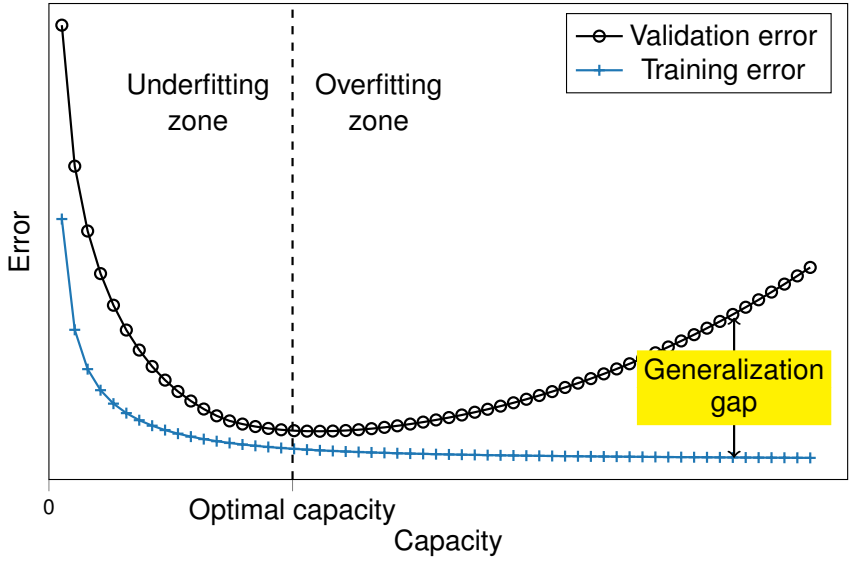
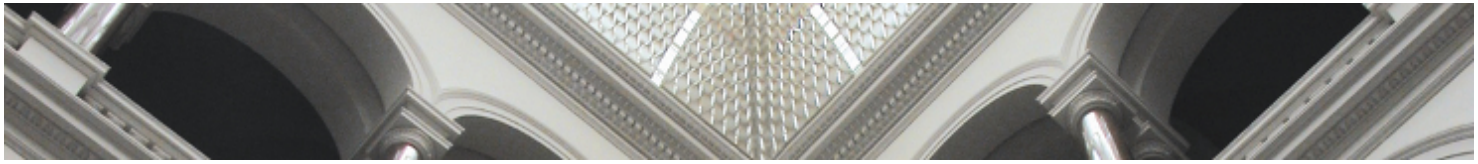


Figure: Influence of capacity

- **Over- and Underfitting**:
  - **Goal**: Reach optimal capacity
- **How to direct capacity ?**:
  - Size of the network
  - Activation functions
  - Loss function
  - Data distortion/ add variation to existing data
  - ...
  - Any other means of regularization
- **Initialization**



Technische Universität Berlin

Table of Contents

Number of intrinsic variables

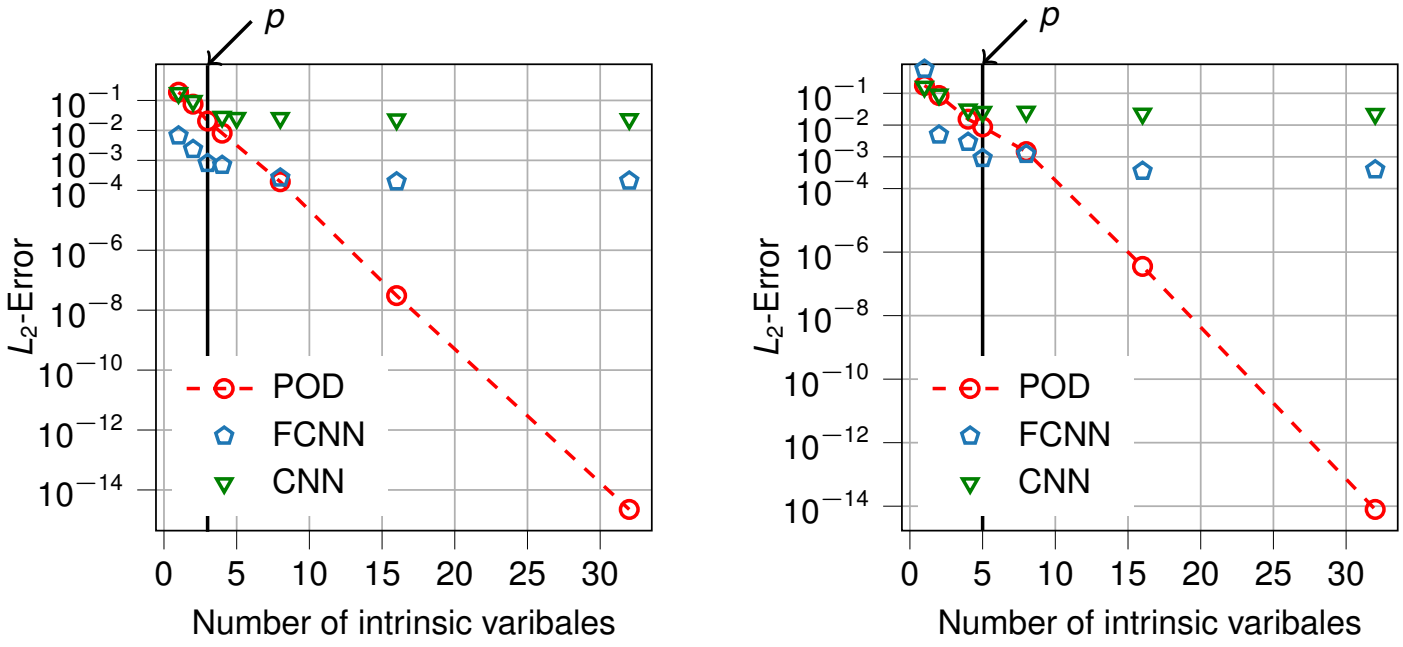
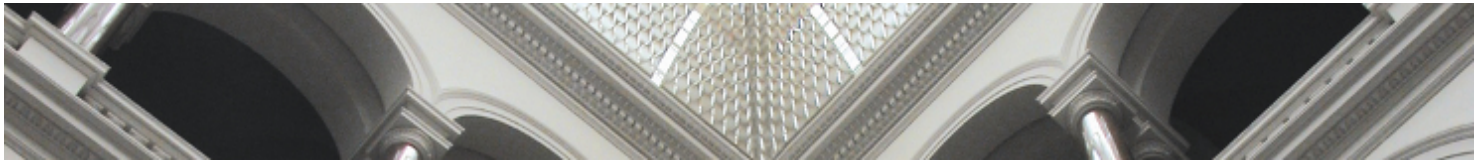


Figure: Variation of p for **H** left and **R** right.

– **Evaluation metric**:

$$L_2\text{-Error} = \frac{||f - \tilde{f}||_2}{||f||_2} \tag{19}$$

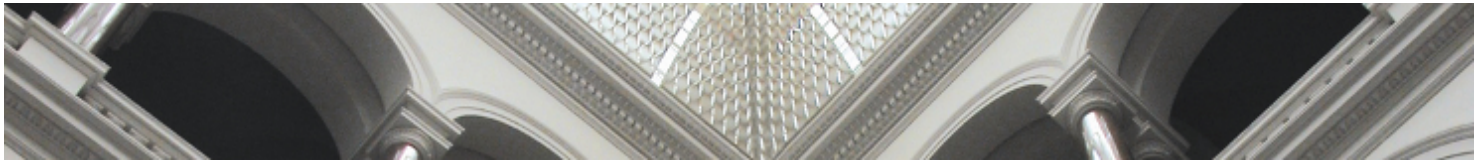


Technische Universität Berlin

Amount of parameters

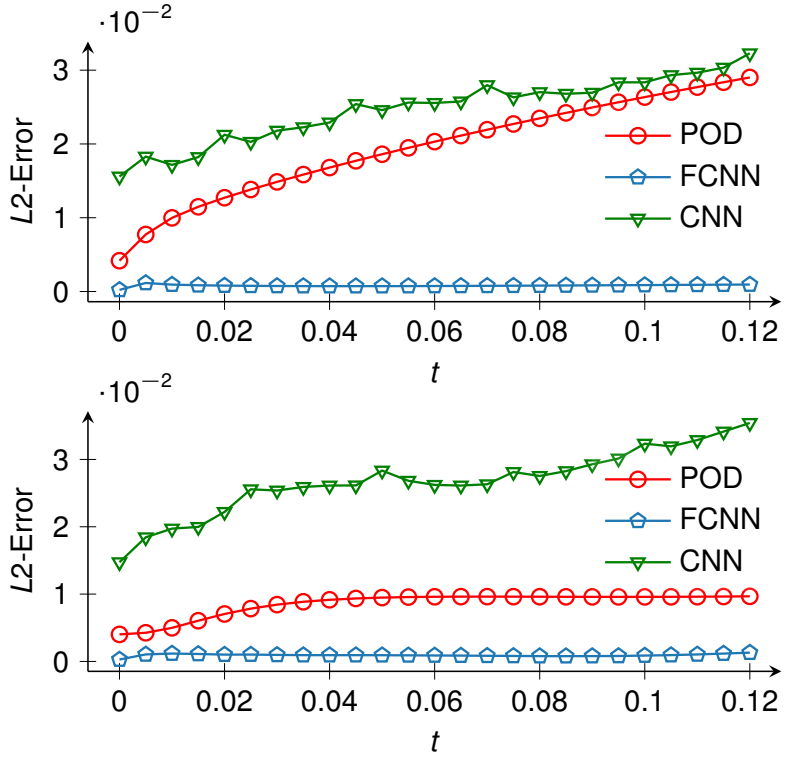
Table: Amount of parameters used to reconstruct f, the number of intrinsic variables p and the corresponding L_2-Error for POD, the FCNNs, and the CNN.

Algorithm	Parameters		Int. variables p		Ł2-error	
	H	**R**	**H**	**R**	**H**	**R**
POD	15129	25225	3	5	0.0205	0.0087
FCNN	2683	3725	3	5	0.0008	0.0009
CNN	8246	8246	5	5	0.025	0.027



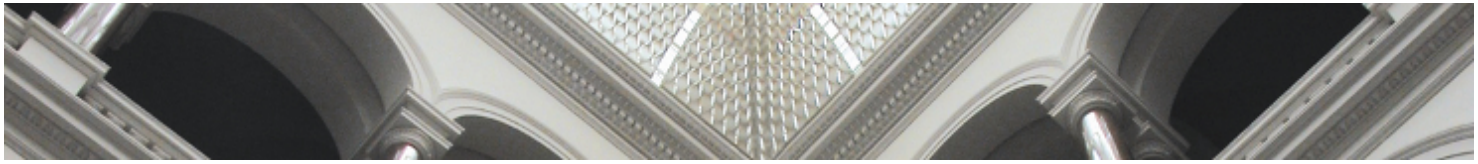
Technische Universität Berlin

Time dependece of L_2-Error



- **POD**:
  - **H** - lin. increase of L_2
  - **R** - increase & stagnation of L_2
- **FCNN**:
  - **H** & **R** - no distinct time dependence L_2
  - biggest value at onset
- **CNN**:
  - **H** & **R** - similar evolution

Figure: Comparison of the L_2-Error over time, **H** top and **R** bottom.



A detailed look at reconstructions

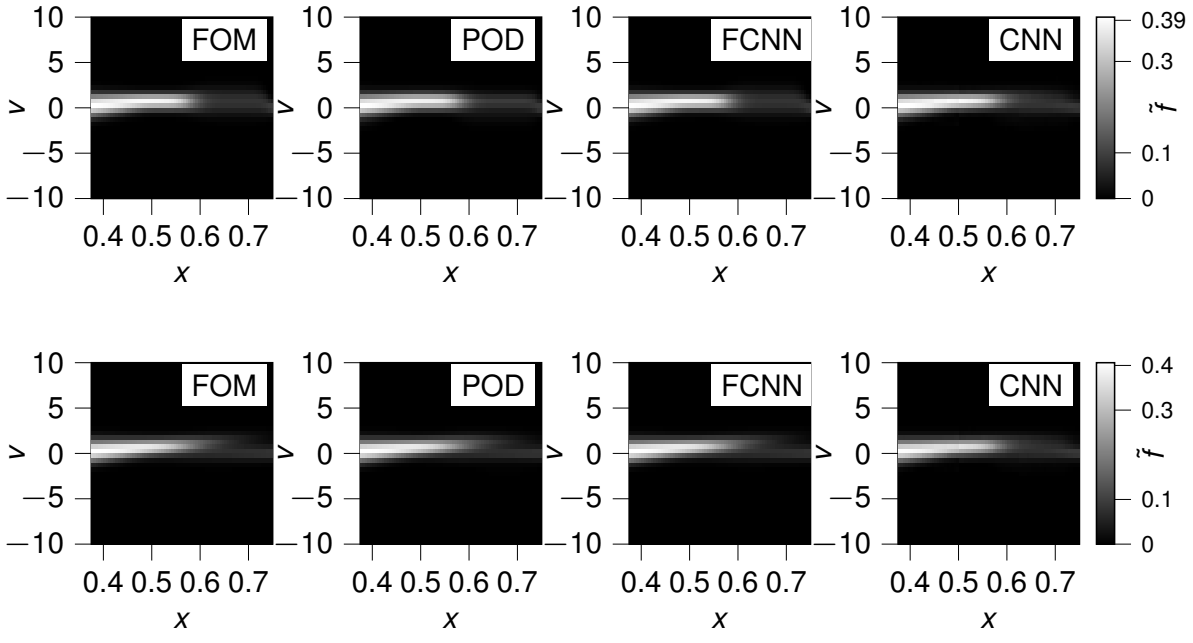
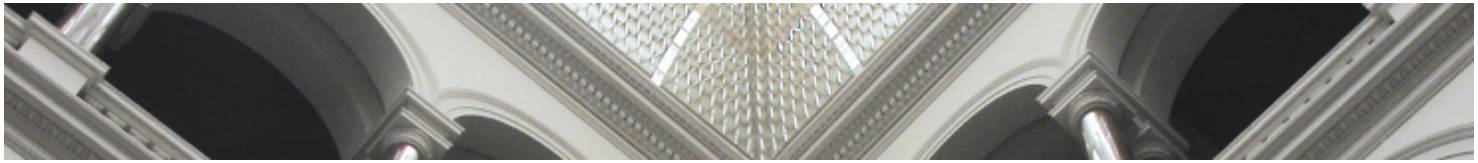


Figure: Comparision of f and $\tilde{f}$ at $t = t_{end}$ and $x \in [0.375, 0.75]$, **H** top and **R** bottom.

- **POD**:
  - **H** - defective after $x = 0.6$: Errors in temperature
  - **R** - almost exact
- **FCNN**:
  - **H** & **R** - almost exact
- **CNN**:
  - **H** & **R** - average of **H** & **R**



Technische Universität Berlin

Moments of f and $\tilde{f}$

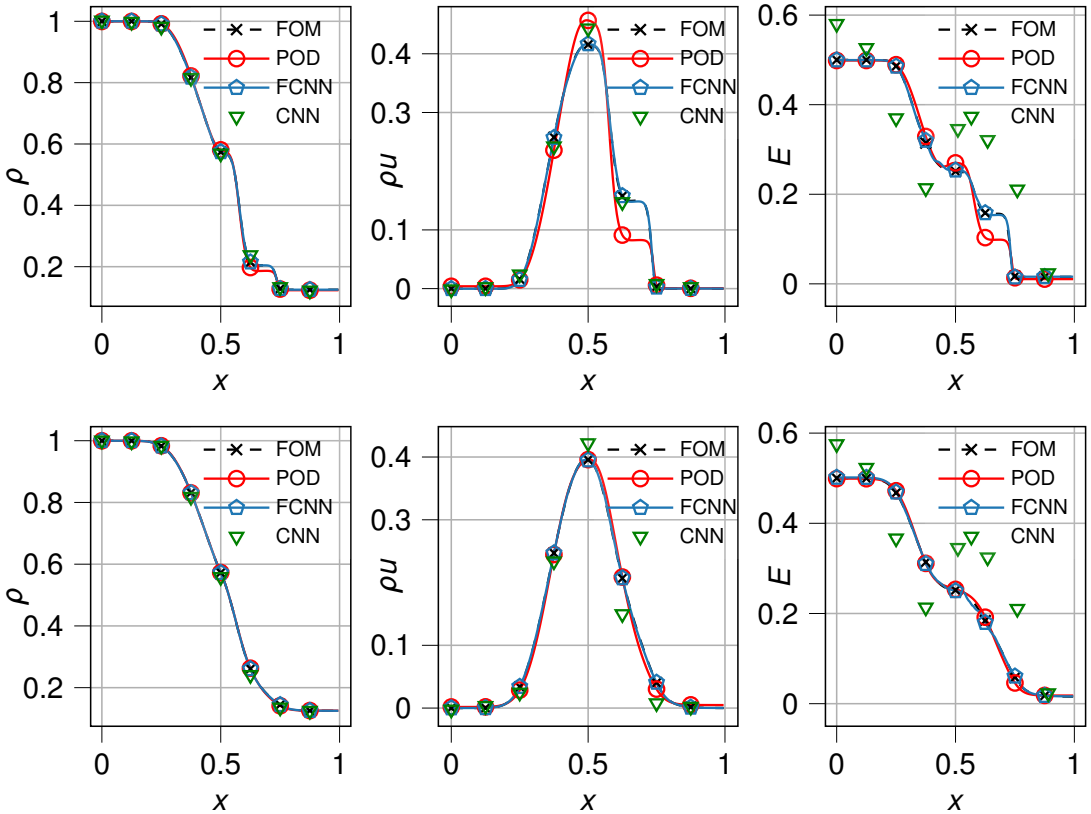
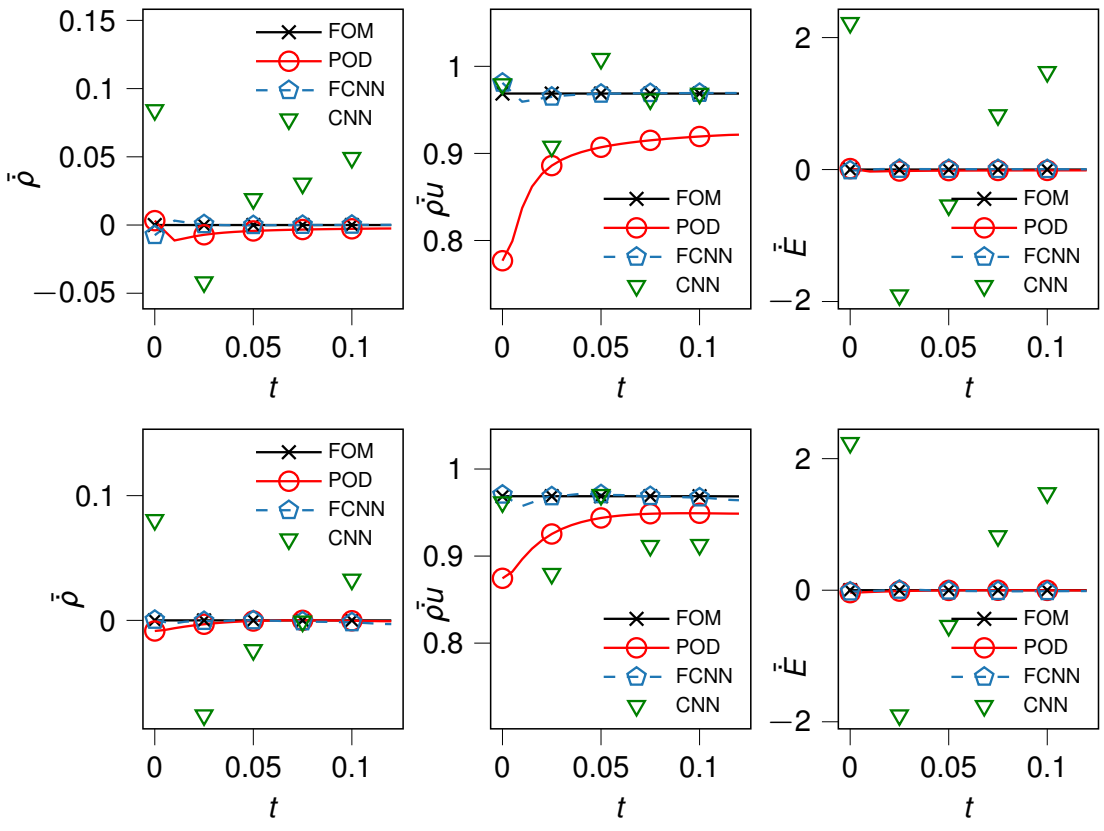


Figure: Comparison of moments of f and $\tilde{f}$, **H** top and **R** bottom.

- **POD**:
  - **H** - undercuts shockwave in ρ, ρu and E
    ++ - ρu exceeds tail of rarefaction wave
  - **R** - only small deviations at shockwave for ρu and E
- **FCNN**:
  - **R** - severe deviation at transition: tail of rarefaction wave $\longrightarrow$ shockfront
- **CNN**:
  - **R** is copy of **H**

Technische Universität Berlin

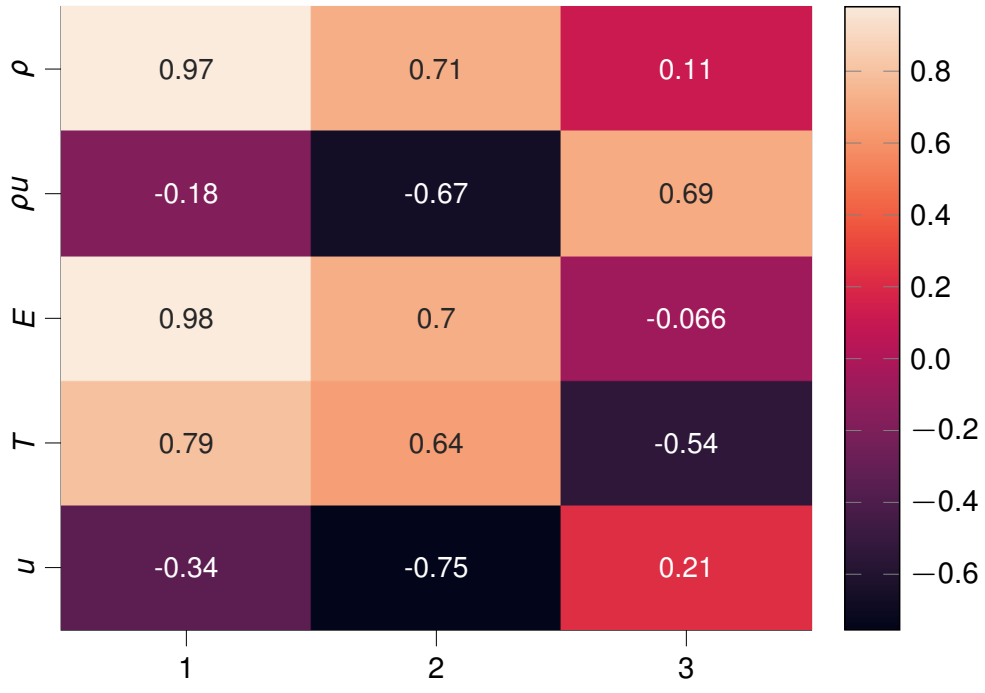
Physical consistency



- **POD**:
  - **H** & **R** - mass & + mass
  - **H** & **R** +++ momentum
  - **H** & **R** energy
- **FCNN**:
  - **H** & **R** mass
  - **H** & **R** - momentum & + momentum
  - **H** & **R** energy
- **CNN**:
  - No conservation

Figure: Conservation properties of f and $\tilde{f}$, **H** top and **R** bottom.

Interpretabiliy



- **1**:
  - E: 0.98 & ρ: 0.97
- **2**:
  - u: -0.75 & ρ: 0.71
- **3**:
  - ρu: 0.69 & T: -0.54

Figure: Pearson correlation between of macroscopic quantities and intrinsic variables for **H**

Interpretabiliy

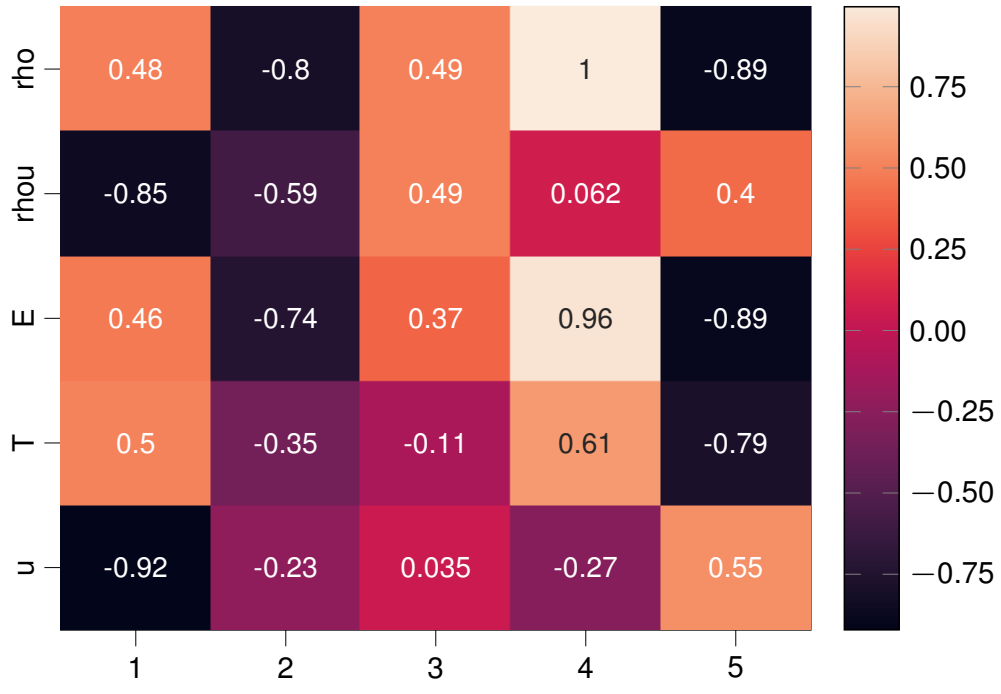
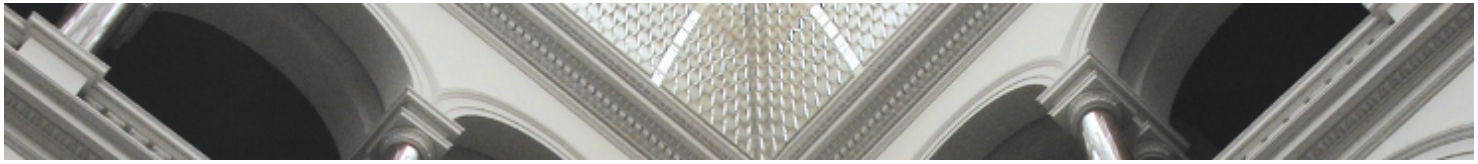


Figure: Pearson correlation between of macroscopic quantities and intrinsic variables for **H**

- **1**:
  - u: -0.92 & ρu: -0.85
- **2**:
  - ρ: -0.8 & E: -0.74
- **3**:
  - ρ: 0.49 & ρu: 0.49
- **4**:
  - ρ: 1 & E: 0.96
- **5**:
  - ρ: -0.89 & T: -0.89



Technische Universität Berlin

Interpolation

Table: Validation and metric results for the interpolation task with 13, 9, 7 and 5 time steps.

n	Δt^*	Validation error		L_2-error		L_2-error	
		$\mathbf{H}^*$	$\mathbf{R}^*$	$\mathbf{H}^*$	$\mathbf{R}^*$	$\tilde{\mathbf{H}}^*$	$\tilde{\mathbf{R}}^*$
13	0.01s	2.5×10^{-8}	2.9×10^{-7}	0.0018	0.0054	0.0036	0.0058
9	0.015s	2.9×10^{-8}	9.5×10^{-8}	0.0017	0.0038	0.0067	0.0056
7	0.02s	2.5×10^{-8}	1.6×10^{-7}	0.0019	0.0042	0.0101	0.0073
5	0.025s	1.7×10^{-7}	1.6×10^{-7}	0.0039	0.0051	0.0367	0.0138

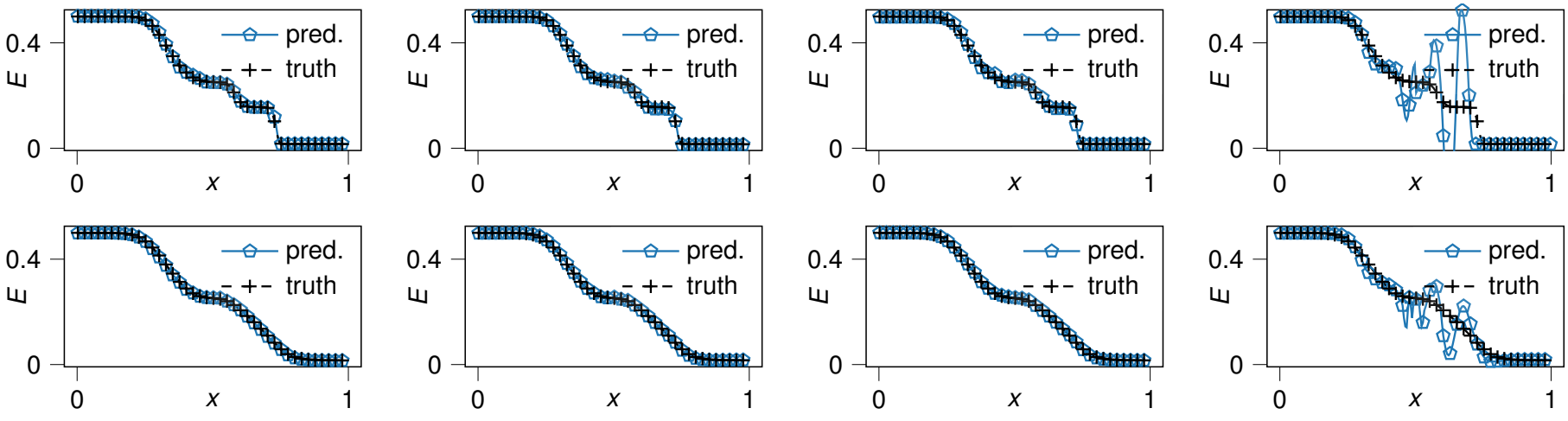
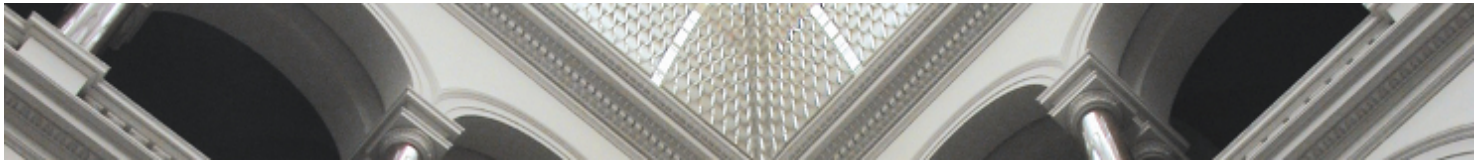
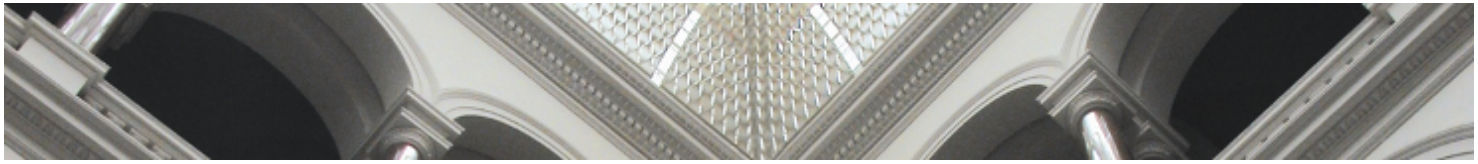


Figure: Energy after interpolation using the FCNN trained on 13,9,7 and 5 time steps, **H** top and **R** bottom.



Technische Universität Berlin

Table of Contents



Technische Universität Berlin

ToDo

- **ToDo** schreiben
- **ToDo** abarbeiten