Munich Center for Machine Learning

ellis

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

relAI

Semi-Structured Regression
Current Advances and Challenges

David Rügamer

Research Seminar @ WU Vienna
Mar 5, 2025

Outline

Semi-Structured Regression

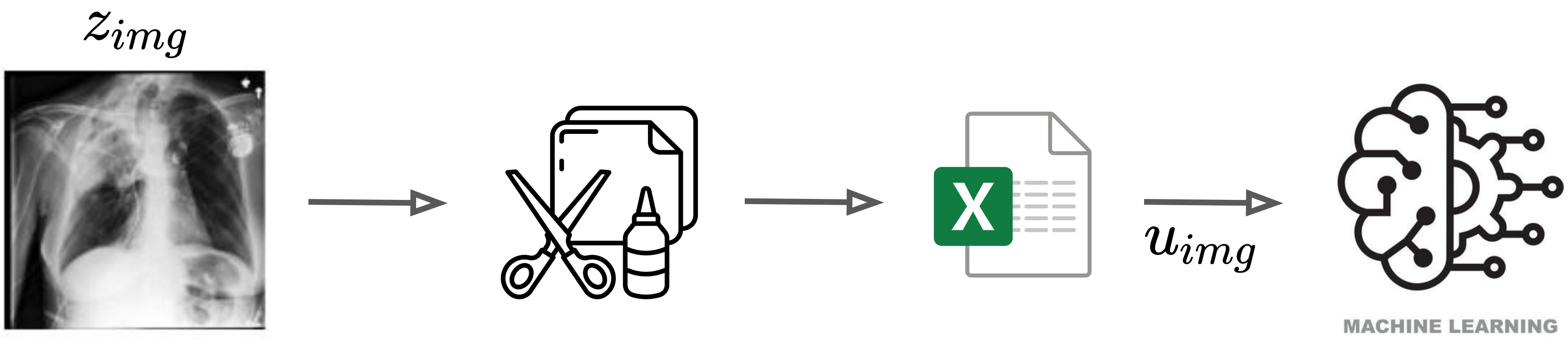
1. **Intro**: Motivation & Implementation
2. **Advantages**: Flexibility & Scalability
3. **Challenges**: Structured Sparsity
4. **Current & Future**: Statistical Inference

Motivation

Motivating Example: Radiology

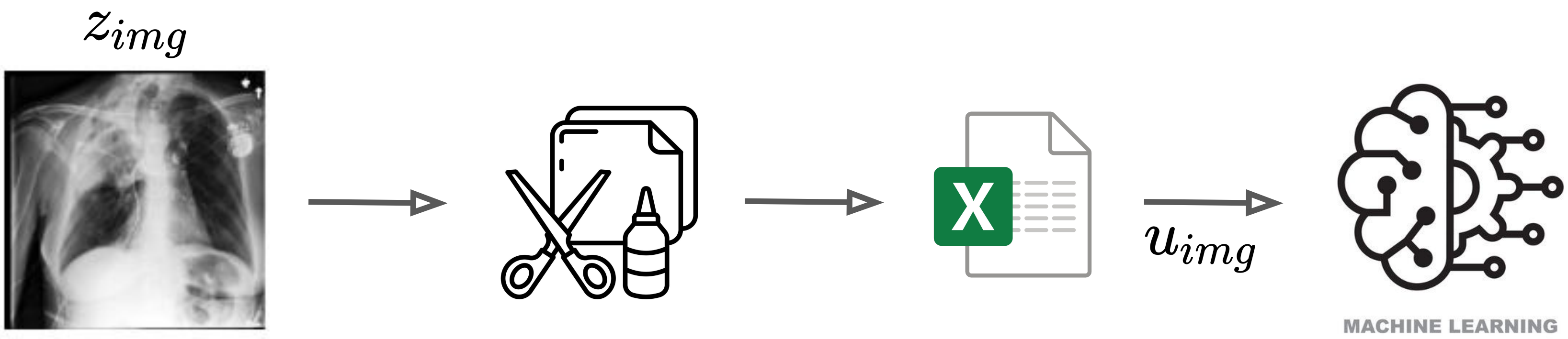
Prof. Ingrisch (LMU)
Clinical Data Science

Typical workflow



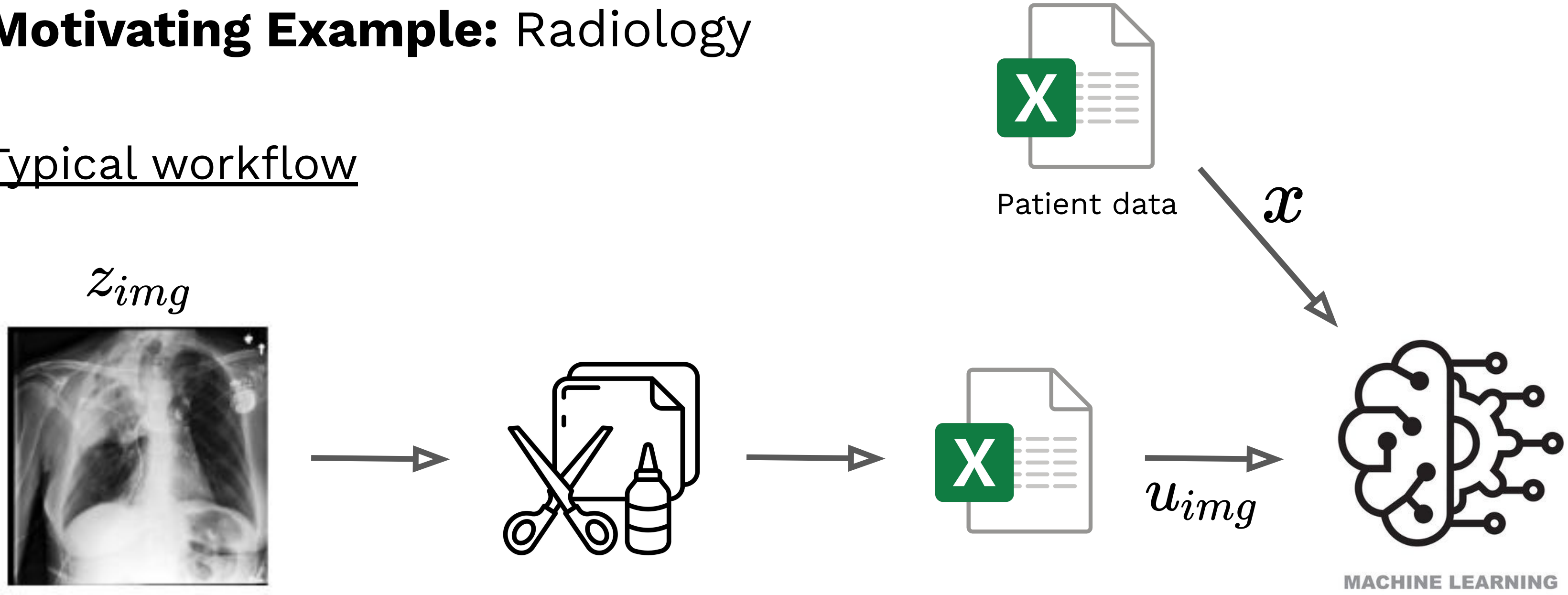
Motivating Example: Radiology

<u>Typical workflow</u>



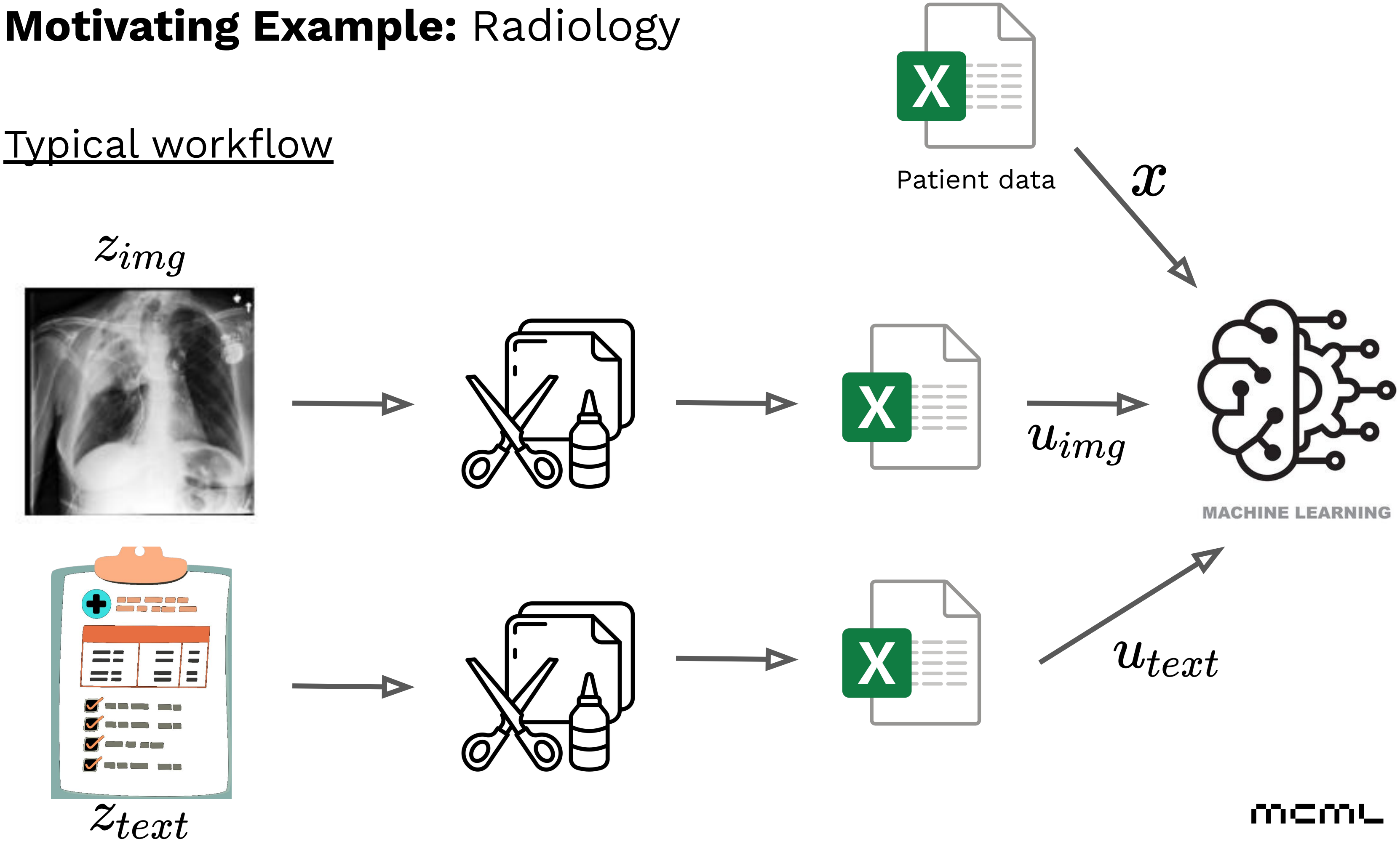
Motivating Example: Radiology

Typical workflow

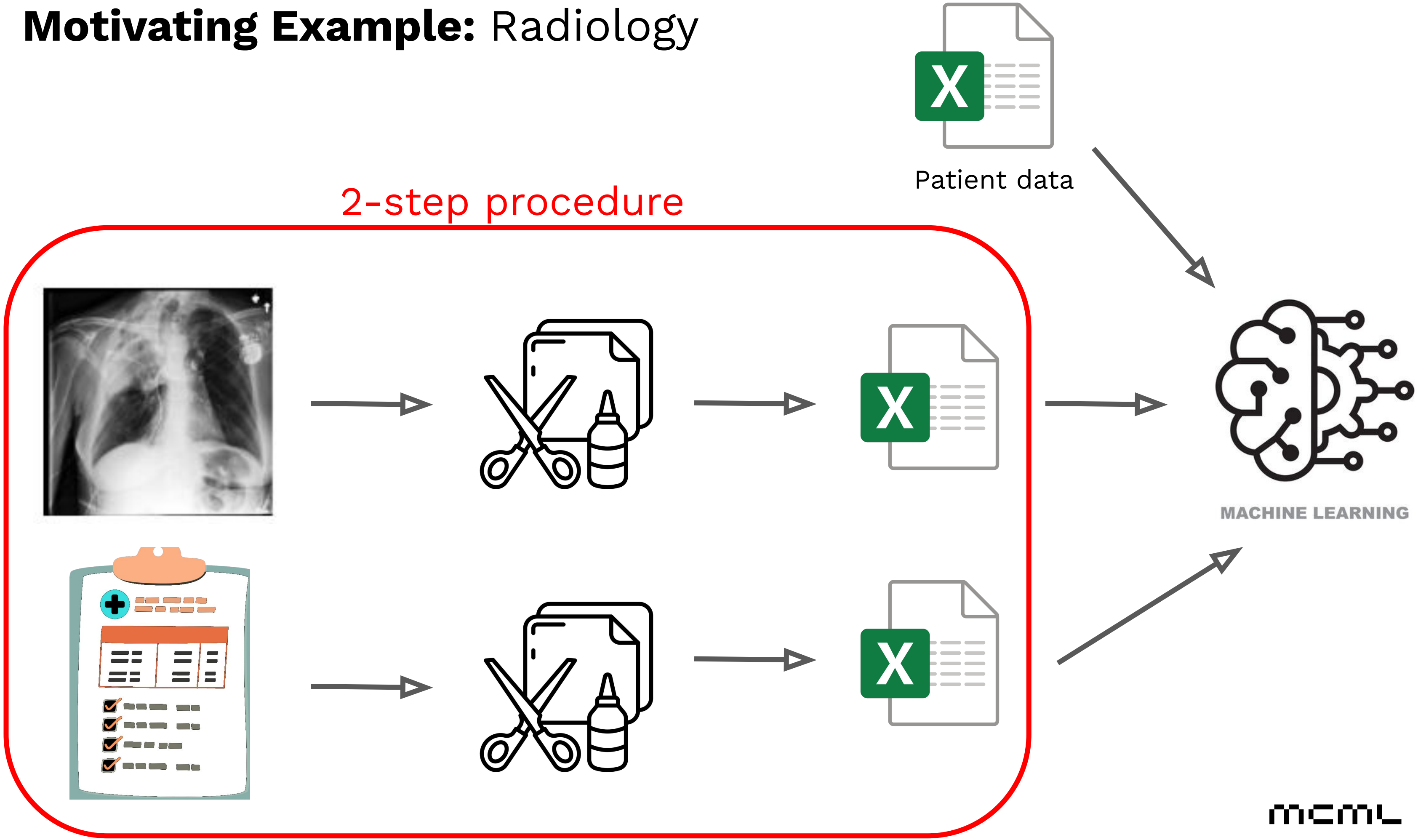


Motivating Example: Radiology

Typical workflow



Motivating Example: Radiology



Semi-Structured Regression

Idea:

- Jointly train statistical model
- and deep neural network(s)
- in one large unifying neural network **end-to-end**

Semi-Structured Regression

Idea:

- Jointly train statistical model
- and deep neural network(s)

$$\left.\right\} w := (\beta, \phi, \xi)$$

- in one large unifying neural network **end-to-end**

Semi-Structured Regression

Idea:

- Jointly train statistical model
- and deep neural network(s)
- in one large unifying neural network **end-to-end**

$w := (\beta, \phi, \xi)$ (model and network(s))

$$\eta = x^\top \beta + \underbrace{\mathrm{NN}_\phi(z_{img})}_{u_{img}} + \underbrace{\mathrm{NN}_\xi(z_{text})}_{u_{text}}$$

Semi-Structured Distributional Regression
Rügamer et al., TAS 2024
A New PHO-rmula for Improved Performance of Semi-Structured Networks
Rügamer, ICML 2023

Semi-Structured Regression

Idea:

- Jointly train statistical model
- and deep neural network(s)
- in one large unifying neural network **end-to-end**

$$\left.\begin{array}{l}\text{Jointly train statistical model}\\ \text{and deep neural network(s)}\end{array}\right\} w := (\beta, \phi, \xi)$$

$$\eta = x^\top \beta + \underbrace{\mathrm{NN}_\phi(z_{img})}_{u_{img}} + \underbrace{\mathrm{NN}_\xi(z_{text})}_{u_{text}}$$

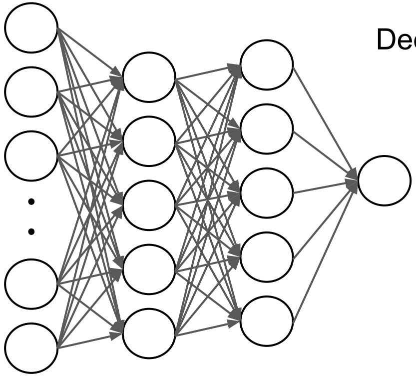
$$= \mathrm{NN}_w(x, z_{img}, z_{text})$$

Semi-Structured Distributional Regression
Rügamer et al., TAS 2024
A New PHO-rmula for Improved Performance of Semi-Structured Networks
Rügamer, ICML 2023

How?

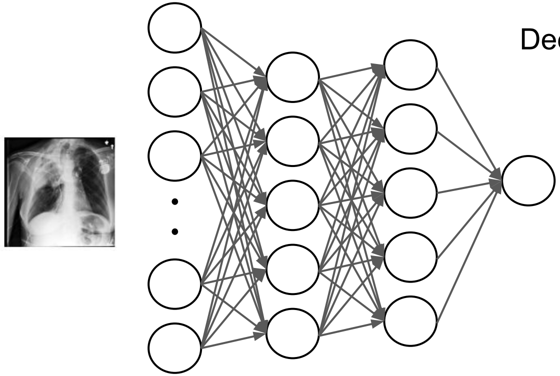
Semi-Structured Regression

Semi-Structured Regression



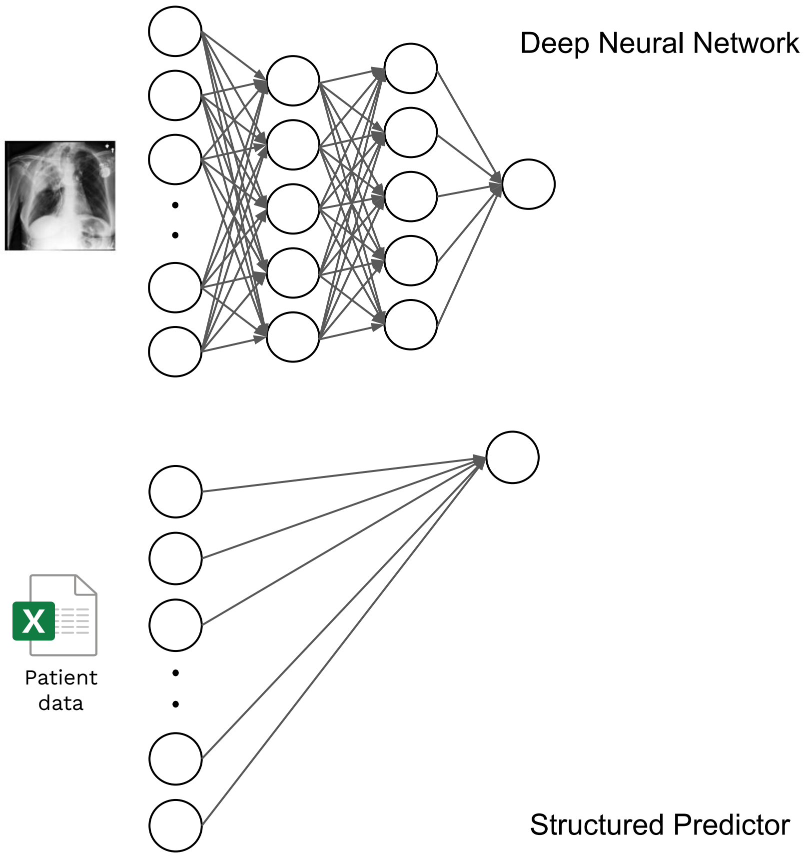
Deep Neural Network

Semi-Structured Regression

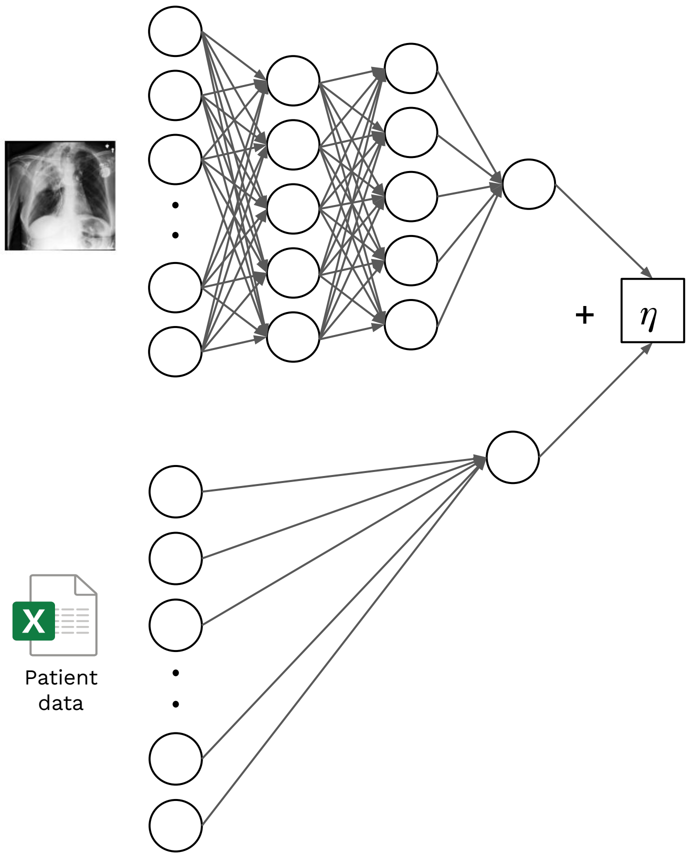


Deep Neural Network

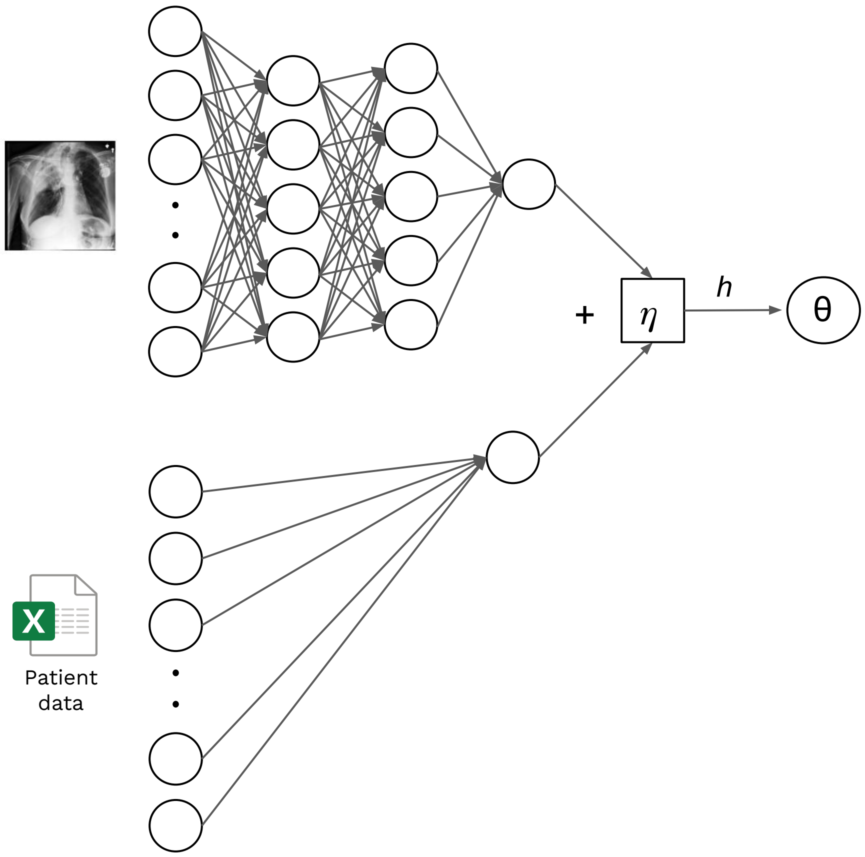
Semi-Structured Regression



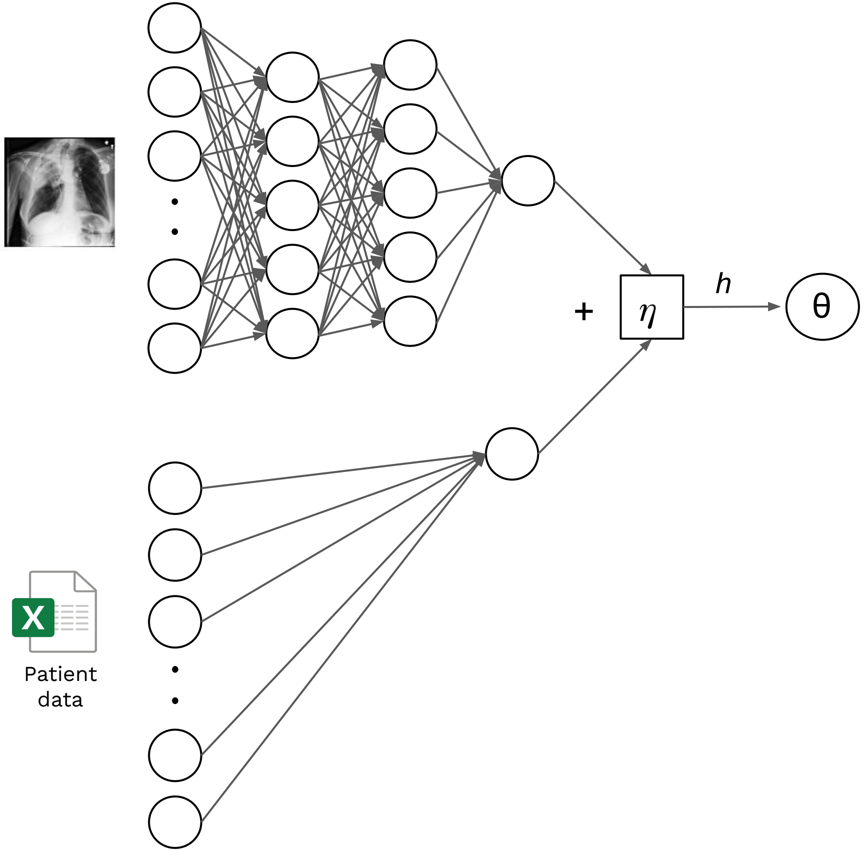
Semi-Structured Regression



Semi-Structured Regression



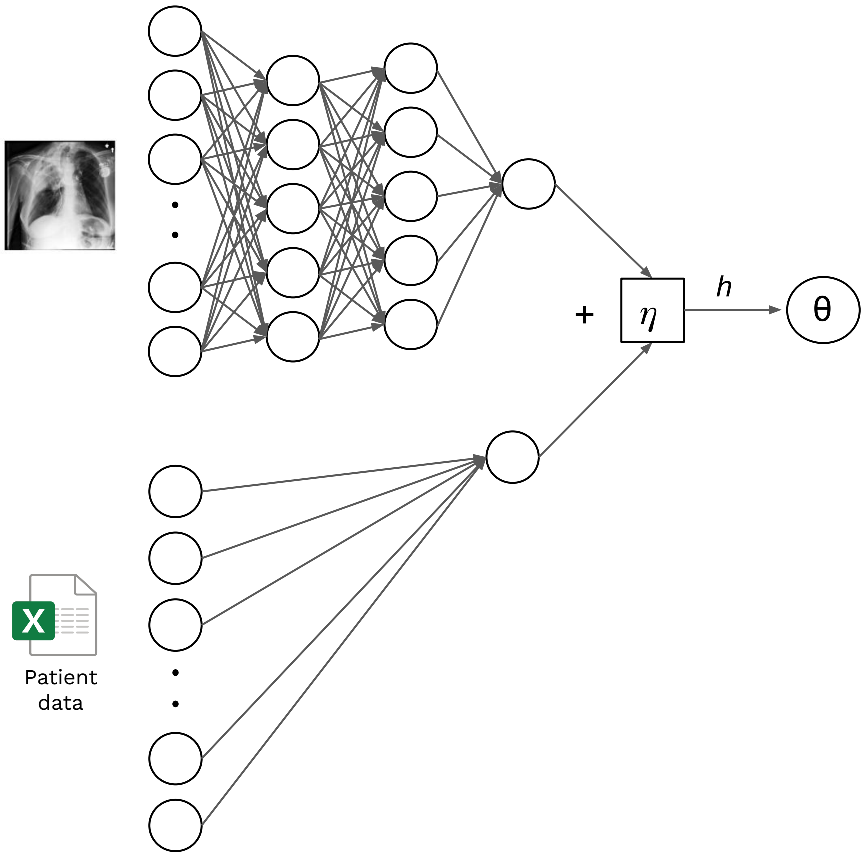
Semi-Structured Regression



Optimization via Maximum Likelihood by minimizing

$$-\sum_i \log f(y_i|\theta_i = h(\eta_i))$$

Semi-Structured Regression



Optimization via Maximum Likelihood by minimizing

$$-\sum_i \log f(y_i | \theta_i = h(\eta_i))$$

Implemented in `deepregression` (CRAN/Github)

deepregression: a Flexible Neural Network Framework for Semi-Structured Deep Distributional Regression
DR, Kolb, ..., Kook, et al., JSS 2023

Example: Predicting Airbnb Prices in Munich

- Tabular information:
  - bathrooms, bedrooms, room type,
  - latitude, longitude,
  - information on the host,
  - reviews,
  - …

Example: Predicting Airbnb Prices in Munich

- Tabular information:
  - bathrooms, bedrooms, room type,
  - latitude, longitude,
  - information on the host,
  - reviews,
  - ...
- **Text description**
- **Image**

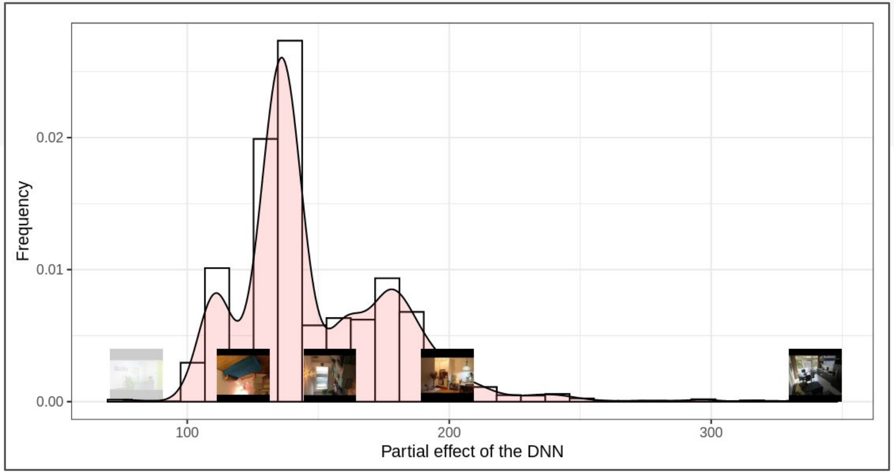
My appartment located in schwabing, Station for Metro, Bus,
Tram are ony 1 minute walking.
The room is 16 square meters, Free WIFI.
We use together Bathroom and Kichen.
I prepare your Bath Towel.
you get Breakfast, cafe oder Tee, Bread..

Example: Predicting Airbnb Prices in Munich

```r
mod_airbnb <- deepregression(
    y = price,
    family = "log_normal",
    list_of_formulas = list(
      location = ~1 + te(latitude, longitude) + room_type + bedrooms +
                   cnn(image) + lstm(desc),
      scale = ~1
    ),
    data = d_train
)
```

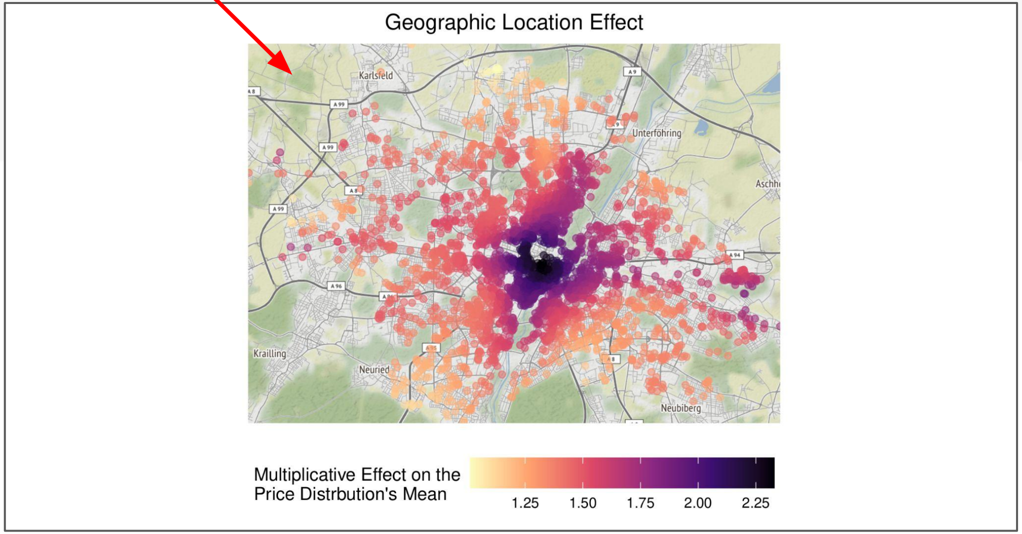
Example: Predicting Airbnb Prices in Munich

```
mod_airbnb <- deepregression(
    y = price,
    family = "log_normal",
    list_of_formulas = list(
      location = ~1 + te(latitude, longitude) + room_type + bedrooms +
                      cnn(image) + lstm(desc),
      scale = ~1
    ),
    data = d_train
)
```



Example: Predicting Airbnb Prices in Munich

```
mod_airbnb <- deepregression(
    y = price,
    family = "log_normal",
    list_of_formulas = list(
      location = ~1 + te(latitude, longitude) + room_type + bedrooms +
                   cnn(image) + lstm(desc),
      scale = ~1
    ),
    data = d_train
)
```



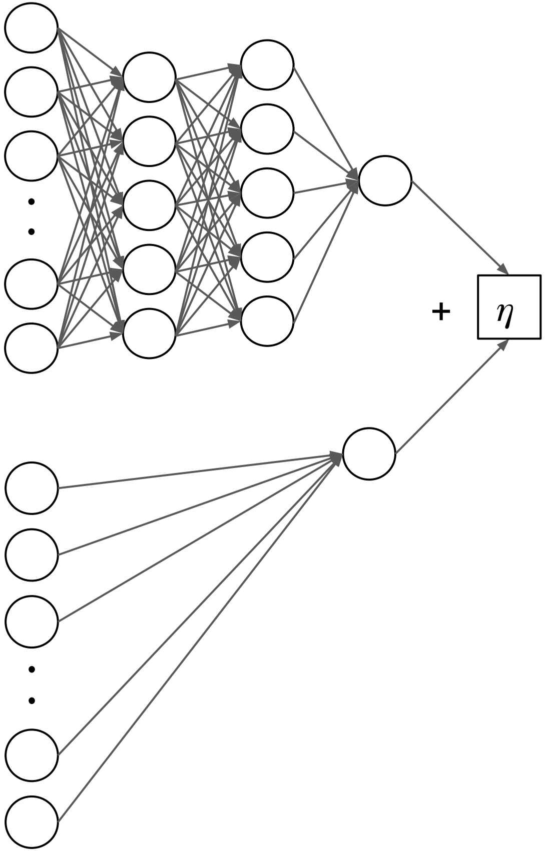
tl;dl: Motivation

Semi-structured models allow you

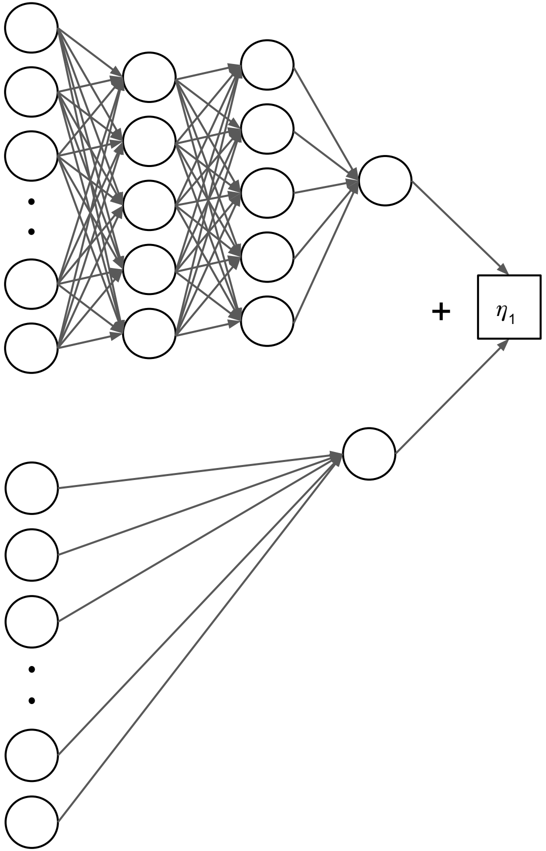
- to work with non-tabular data
- while estimating a structured model predictor

Flexibility

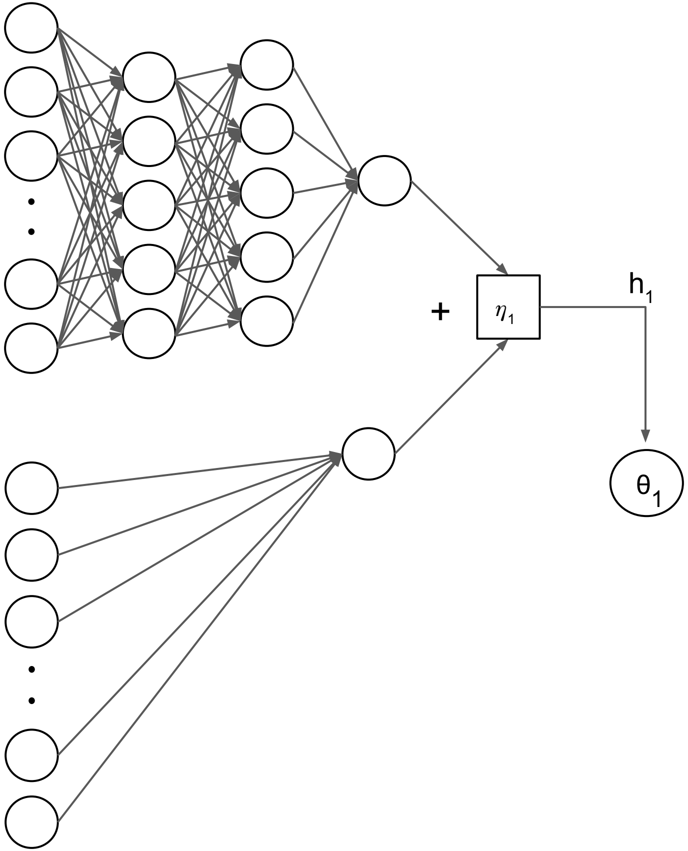
Semi-Structured Regression



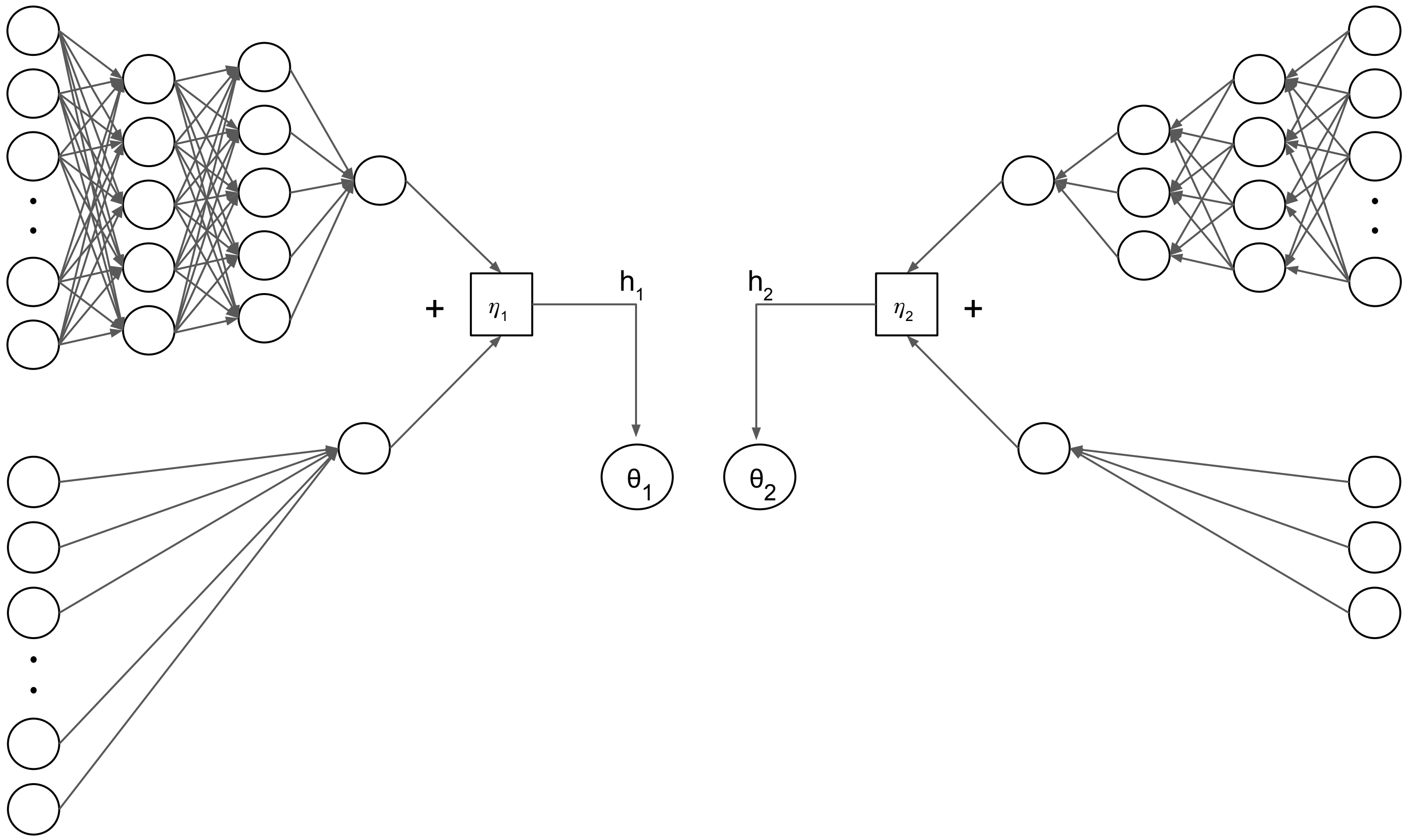
Semi-Structured **Distributional** Regression



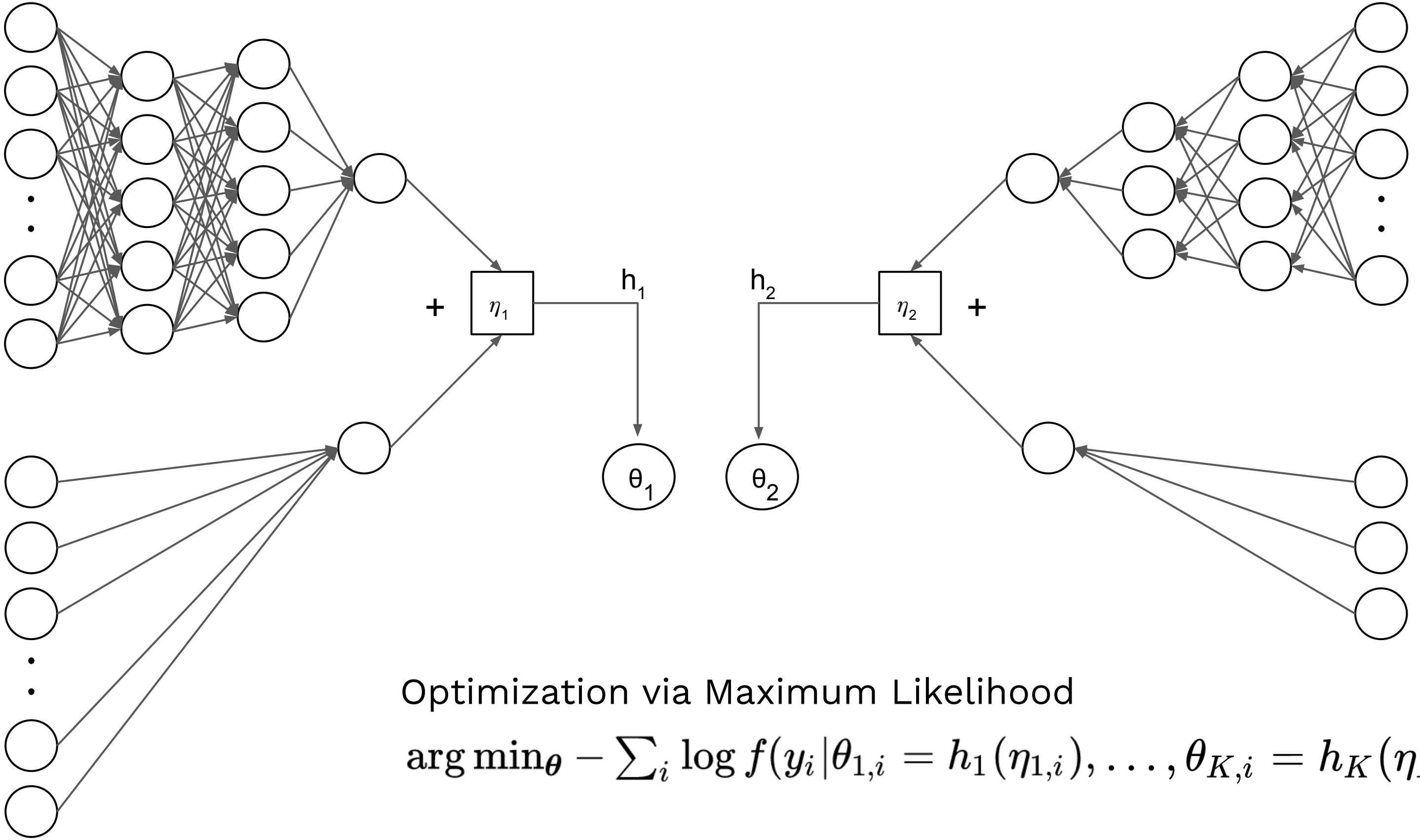
Semi-Structured Distributional Regression



Semi-Structured Distributional Regression



Semi-Structured Distributional Regression



Optimization via Maximum Likelihood

$$\arg\min_{\boldsymbol{\theta}} - \sum_i \log f(y_i | \theta_{1,i} = h_1(\eta_{1,i}), \ldots, \theta_{K,i} = h_K(\eta_{K,i}))$$

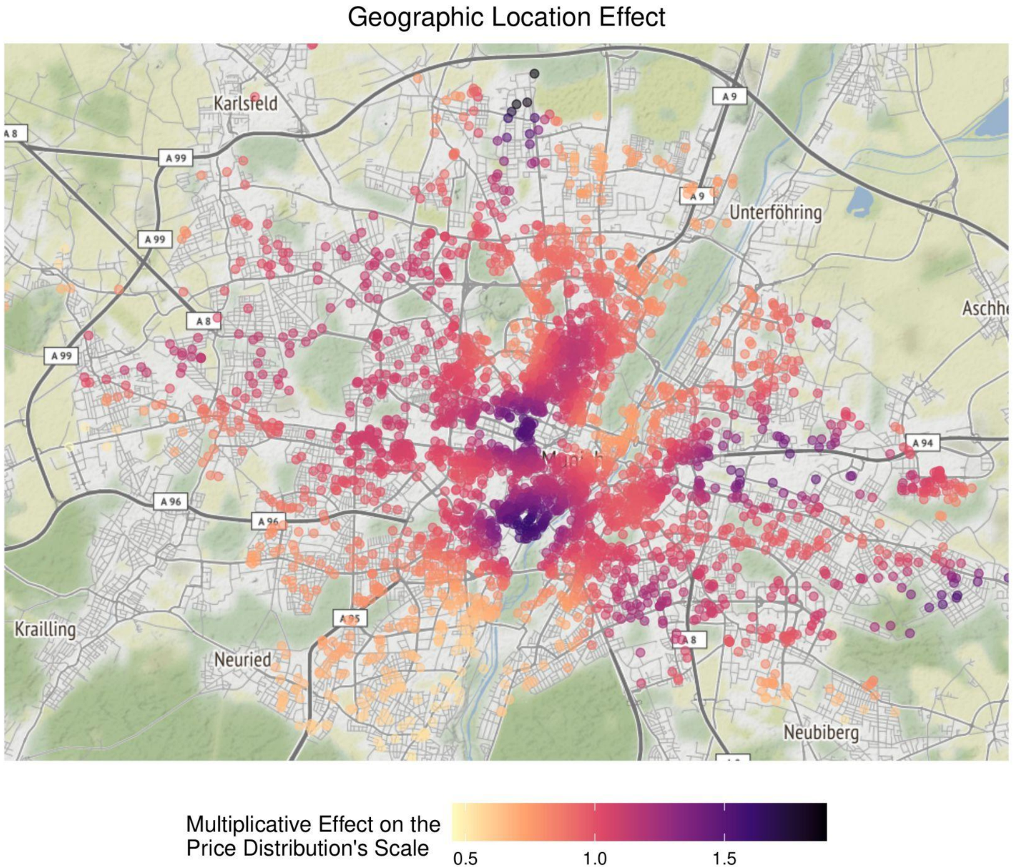
Example: Predicting Airbnb Prices in Munich

```r
mod_airbnb <- deepregression(
    y = price,
    family = "log_normal",
    list_of_formulas = list(
      location = ~1 + te(latitude, longitude) + room_type + bedrooms +
                   cnn(image) + lstm(desc),
      scale = ~1
    ),
    data = d_train
)
```

Example: Predicting Airbnb Prices in Munich

```
mod_airbnb <- deepregression(
    y = price,
    family = "log_normal",
    list_of_formulas = list(
      location = ~1 + te(latitude, longitude) + room_type + bedrooms +
                   cnn(image) + lstm(desc),
      scale = ~1 + te(latitude, longitude)
    ),
    data = d_train
)
```

Example: Predicting Airbnb Prices in Munich



Not flexible enough?

⇒ Mixture regression models (`mixdistreg`)

⇒ Transformation models (`deeptrafo`)

Mixture of Experts Distributional Regression
DR, Pfisterer, Bischl and Grün, AStA 2023
Deep Conditional Transformation Models
Baumann, Hothorn Rügamer, ECML 2021
Estimating Conditional Distributions with Neural Networks Using R Package deeptrafo
Kook, Baumann, Sick, Dürr and DR, JSS 2024
How Inverse Conditional Flows Can Serve as a Substitute for Distributional Regression
Kook, et al. and DR, UAI 2024

Other Model Classes

- Time Series (Schiele et al., ‘22)
- Survival (Kopper et al., DR, AAAI ‘20; PAKDD ‘22)
- Functional Data (Rügamer et al., NeurIPS ‘24)
- Density Data (Jung et al., ‘25+)
- ...

tl;dl: Flexibility

Embedding structured models into neural networks

- provides a flexible toolbox
- allows previously unimagined modeling combinations
- using Stochastic Gradient Descent (SGD) → model-agnostic

Scalability

Other Model Classes

- Time Series (Schiele et al., ‘22)
- Survival (Kopper et al., DR, AAAI ‘20; PAKDD ‘22)
- Functional Data (Rügamer et al., NeurIPS ‘24)
- Density Data (Jung et al., ‘25+)
- ...

Other Model Classes

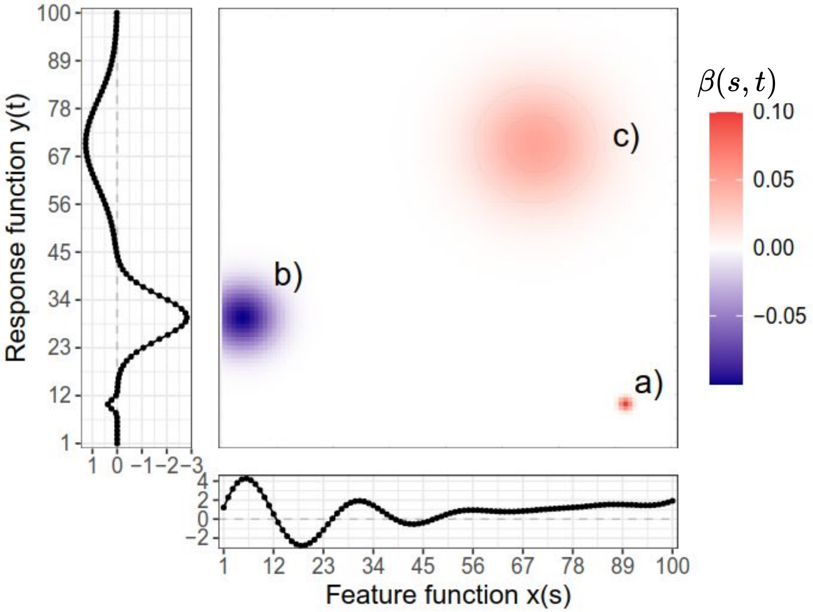
- Time Series (Schiele et al., ‘22)
- Survival (Kopper et al., DR, AAAI ‘20; PAKDD ‘22)
- **Functional Data** (Rügamer et al., NeurIPS ‘24)
- Density Data (Jung et al., ‘25+)
- ...

Functional Regression Models

$$Y_i(t) = \sum_{j=1}^{J} \int_{s \in \mathcal{S}} x_{ji}(s)\beta(s,t)ds + \varepsilon_i(t) \quad t \in \mathcal{T}$$

Functional Regression Models

$$Y_i(t) = \sum_{j=1}^{J} \int_{s \in \mathcal{S}} x_{ji}(s)\beta(s,t)ds + \varepsilon_i(t) \quad t \in \mathcal{T}$$



Functional Regression Models

$$Y_i(t) = \sum_{j=1}^{J} \int_{s \in \mathcal{S}} x_{ji}(s)\beta(s,t)ds + \varepsilon_i(t) \quad t \in \mathcal{T}$$

$$\approx \sum_{j=1}^{J} \sum_{r=1}^{R} \Delta_r x_{ji}(s_r)\beta(s_r,t) + \varepsilon_i(t)$$

Functional Regression Models

$$
\begin{aligned}
Y_i(t) &= \sum_{j=1}^{J} \int_{s \in \mathcal{S}} x_{ji}(s)\beta(s,t)ds + \varepsilon_i(t) \quad t \in \mathcal{T} \\
&\approx \sum_{j=1}^{J} \sum_{r=1}^{R} \Delta_r x_{ji}(s_r)\beta(s_r,t) + \varepsilon_i(t) \\
&\approx \sum_{j=1}^{J} \sum_{r=1}^{R} \Delta_r x_{ji}(s_r)[\mathbf{B}^s(s_r) \otimes \mathbf{B}^t(t)]^\top \boldsymbol{\gamma} + \varepsilon_i(t)
\end{aligned}
$$

Functional Regression Models

$$
\begin{aligned}
Y_i(t) &= \textstyle\sum_{j=1}^{J} \int_{s\in\mathcal{S}} x_{ji}(s)\beta(s,t)ds + \varepsilon_i(t) \quad t \in \mathcal{T} \\
&\approx \textstyle\sum_{j=1}^{J} \sum_{r=1}^{R} \Delta_r x_{ji}(s_r)\beta(s_r, t) + \varepsilon_i(t) \\
&\approx \textstyle\sum_{j=1}^{J} \sum_{r=1}^{R} \Delta_r x_{ji}(s_r)[\mathbf{B}^s(s_r) \otimes \mathbf{B}^t(t)]^\top \boldsymbol{\gamma} + \varepsilon_i(t)
\end{aligned}
$$

$$
Y_i(t_q) \approx \textstyle\sum_{j=1}^{J} \sum_{r=1}^{R} \Delta_r x_{ji}(s_r)[\mathbf{B}^s(s_r) \otimes \mathbf{B}^t(t_q)]^\top \boldsymbol{\gamma} + \varepsilon_i(t_q)
$$

Functional Regression Models

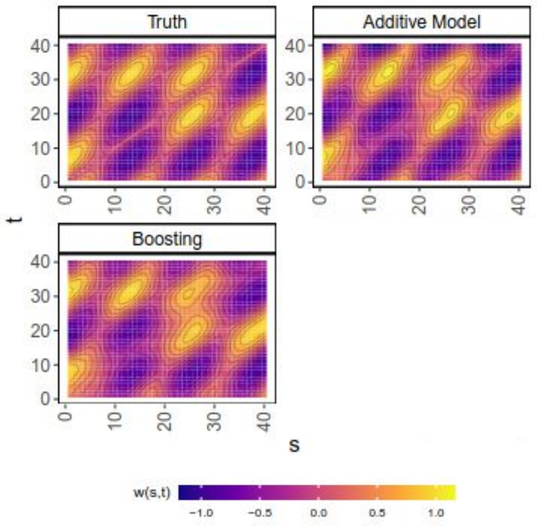
$$
\begin{aligned}
Y_i(t) &= \sum_{j=1}^{J} \int_{s \in \mathcal{S}} x_{ji}(s)\beta(s,t)ds + \varepsilon_i(t) \quad t \in \mathcal{T} \\
&\approx \sum_{j=1}^{J} \sum_{r=1}^{R} \Delta_r x_{ji}(s_r)\beta(s_r, t) + \varepsilon_i(t) \\
&\approx \sum_{j=1}^{J} \sum_{r=1}^{R} \Delta_r x_{ji}(s_r)[\mathbf{B}^s(s_r) \otimes \mathbf{B}^t(t)]^\top \boldsymbol{\gamma} + \varepsilon_i(t)
\end{aligned}
$$

$$
Y_i(t_q) \approx \sum_{j=1}^{J} \sum_{r=1}^{R} \Delta_r x_{ji}(s_r)[\mathbf{B}^s(s_r) \otimes \mathbf{B}^t(t_q)]^\top \boldsymbol{\gamma} + \varepsilon_i(t_q)
$$

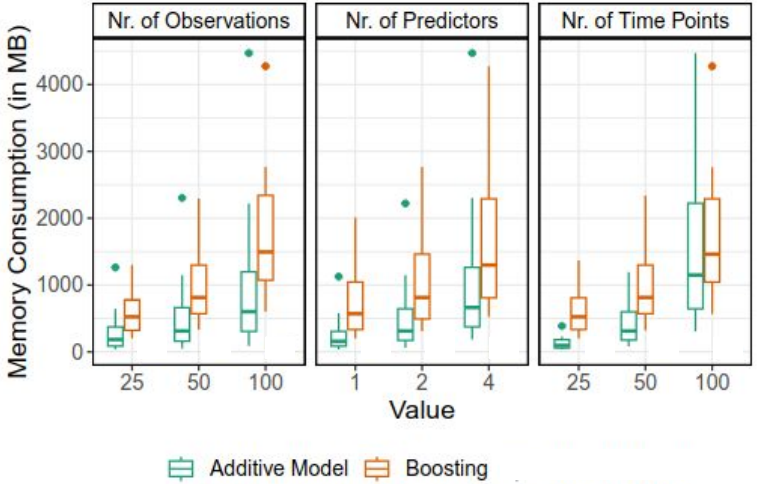
for

$$
\begin{aligned}
i &= 1, \ldots, n \\
r &= 1, \ldots, R \\
q &= 1, \ldots, Q
\end{aligned}
$$

Functional Regression Models

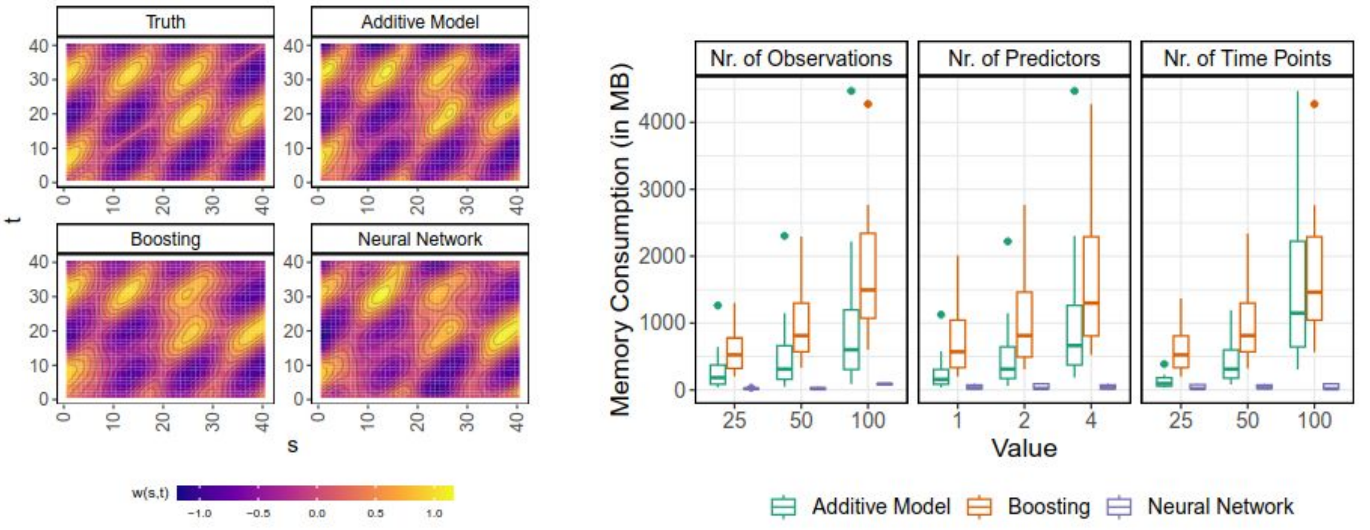


(a) True weight surface $w(s, t)$ used in the simulation study for large SNR along with estimation results of different methods.



(b) Memory consumption of different methods (colors) for different amounts of functional observations n (left), functional predictors J (center), and time points R (right).

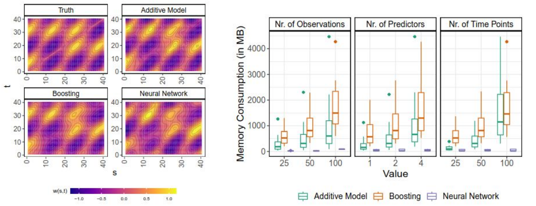
Functional Regression Models



(a) True weight surface $w(s,t)$ used in the simulation study for large SNR along with estimation results of different methods.

(b) Memory consumption of different methods (colors) for different amounts of functional observations n (left), functional predictors J (center), and time points R (right).

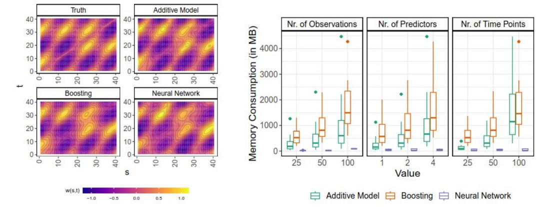
Functional Regression Models



(a) True weight surface $w(s,t)$ used in the simulation study for large SNR along with estimation results of different methods.

(b) Memory consumption of different methods (colors) for different amounts of functional observations n (left), functional predictors J (center), and time points R (right).

Functional Regression Models

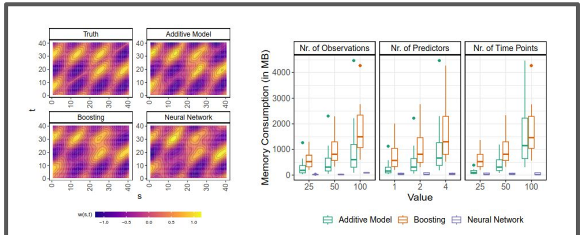


(a) True weight surface $w(s,t)$ used in the simulation study for large SNR along with estimation results of different methods.

(b) Memory consumption of different methods (colors) for different amounts of functional observations n (left), functional predictors J (center), and time points R (right).

$$Y_i(t_q) \approx \sum_{j=1}^{J} \sum_{r=1}^{R} \Delta_r x_{ji}(s_r)[\mathbf{B}^s(s_r) \otimes \mathbf{B}^t(t_q)]^\top \boldsymbol{\gamma} + \varepsilon_i(t_q)$$

Functional Regression Models



(a) True weight surface $w(s,t)$ used in the simulation study for large SNR along with estimation results of different methods.

(b) Memory consumption of different methods (colors) for different amounts of functional observations n (left), functional predictors J (center), and time points R (right).

Basis Recycling

$$Y_i(t_q) \approx \sum_{j=1}^{J} \sum_{r=1}^{R} \Delta_r x_{ji}(s_r)[\mathbf{B}^s(s_r) \otimes \mathbf{B}^t(t_q)]^\top \boldsymbol{\gamma} + \varepsilon_i(t_q)$$

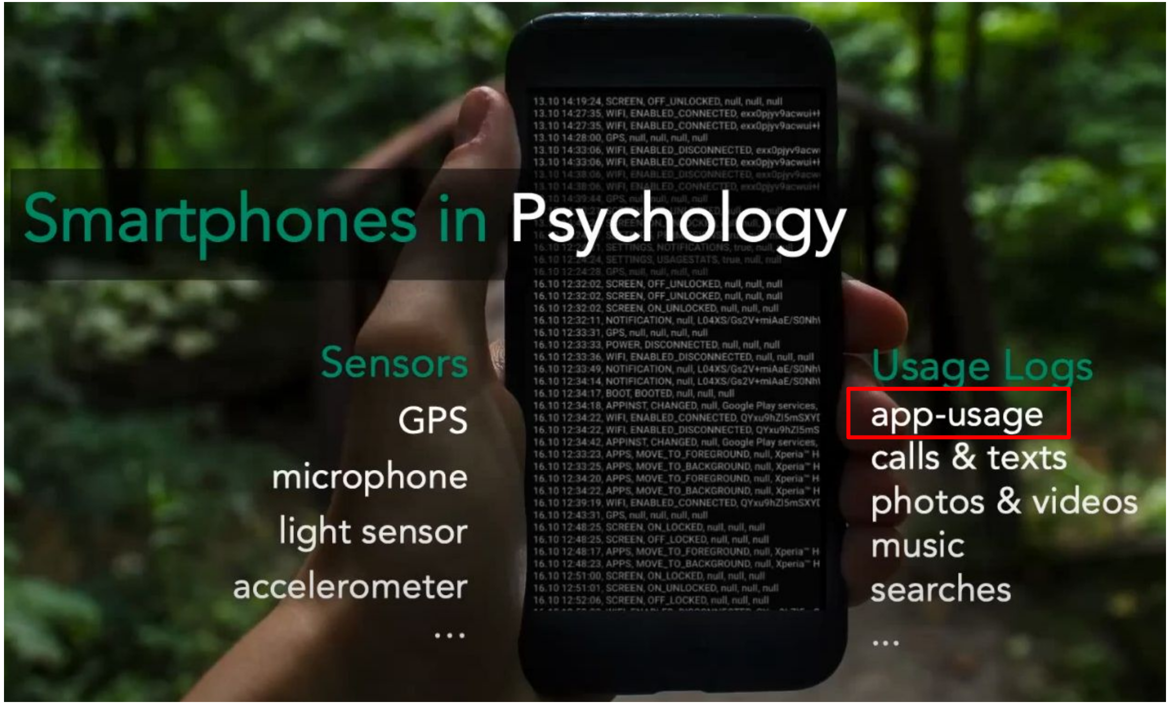
Stoch. Optimization

Array Computations

Large-Scale Applications

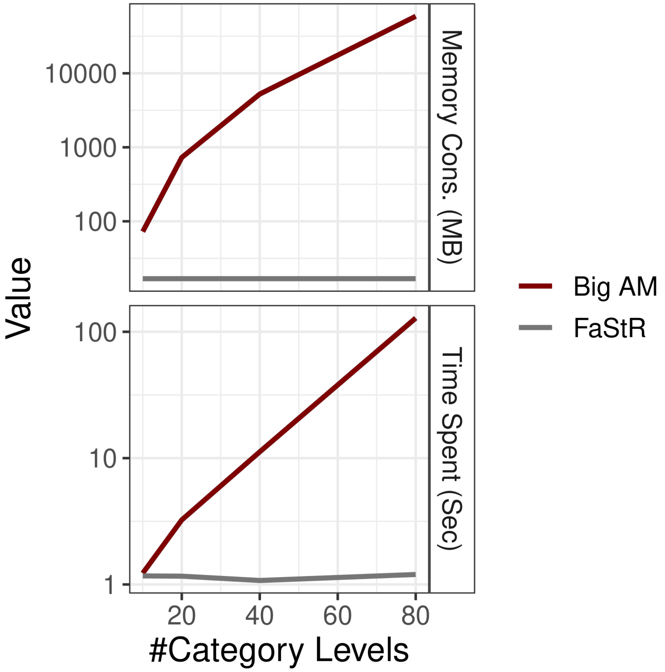
Prof. Stachl (St. Gallen)
Behavioral Psychology

Large-Scale Applications



Large-Scale Applications

Rügamer et al., ECML-PKDD 2022



tl;dl: Scalability

Embedding structured models into neural networks

- provides an easy way to scale for large datasets
- offers many ways to also scale to high dimensions and more complex model predictors

Challenge: Structured Sparsity

Sparsity in Neural Networks

Sparsity in Neural Networks – Problem Setup

Lasso:

$$\frac{1}{n}||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_1$$

Sparsity in Neural Networks – Problem Setup

Lasso:

$$\underbrace{\frac{1}{n}||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2}_{\text{cont. + convex}} + \underbrace{\lambda||\boldsymbol{\beta}||_1}_{\substack{\text{cont. + convex} \\ \textbf{but non-smooth}}}$$

Sparsity in Neural Networks – Problem Setup

Lasso:

$$\underbrace{\frac{1}{n}||\boldsymbol{y} - \boldsymbol{X}\beta||_2^2}_{\text{cont. + convex}} + \lambda \underbrace{||\boldsymbol{\beta}||_1}_{\text{cont. + convex}}$$

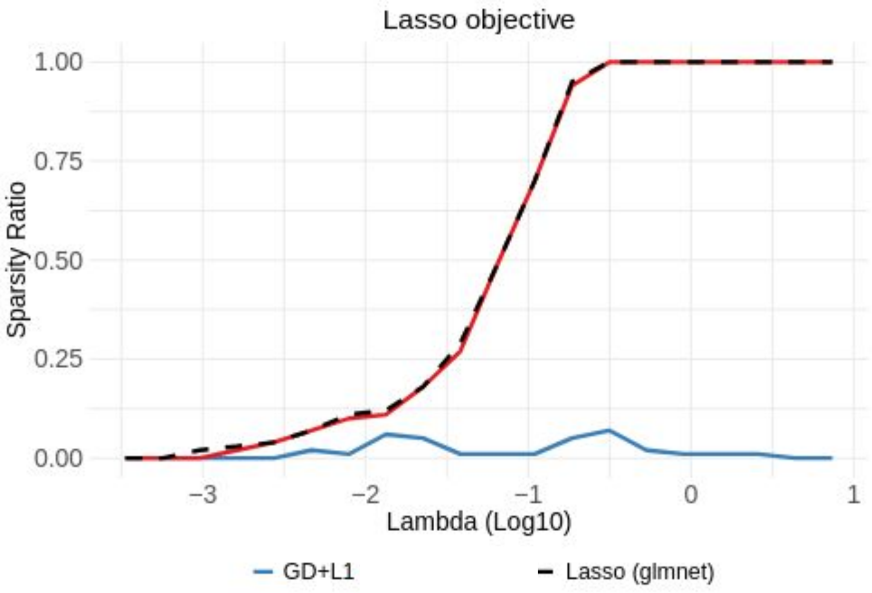
cont. + convex

cont. + convex
but non-smooth

↓

SGD will not work

Comparison with Proximal-type Routines



Hadamard Product Parameterization
for Lasso

- Parametrize $\boldsymbol{\beta} = \boldsymbol{u} \odot \boldsymbol{v}$
- Replace non-smooth $||\boldsymbol{\beta}||_1$
  by smooth $||\boldsymbol{u}||_2^2 + ||\boldsymbol{v}||_2^2$

$$\frac{1}{n}||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_1 \longrightarrow \frac{1}{n}||\boldsymbol{y} - \boldsymbol{X}(\boldsymbol{u} \odot \boldsymbol{v})||_2^2 + \lambda(||\boldsymbol{u}||_2^2 + ||\boldsymbol{v}||_2^2)$$

Hadamard Product Parameterization

for Lasso

- Parametrize $\boldsymbol{\beta} = \boldsymbol{u} \odot \boldsymbol{v}$
- Replace non-smooth $||\boldsymbol{\beta}||_1$

  by smooth $||\boldsymbol{u}||_2^2 + ||\boldsymbol{v}||_2^2$

$$\frac{1}{n}||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_1 \longrightarrow \frac{1}{n}||\boldsymbol{y} - \boldsymbol{X}(\boldsymbol{u} \odot \boldsymbol{v})||_2^2 + \lambda(||\boldsymbol{u}||_2^2 + ||\boldsymbol{v}||_2^2)$$

$\Rightarrow$ Optimal solution is the same,
& introduces no additional local minima

Hadamard Product Parameterization
for Lasso

- Parametrize $\boldsymbol{\beta} = \boldsymbol{u} \odot \boldsymbol{v}$
- Replace non-smooth $||\boldsymbol{\beta}||_1$

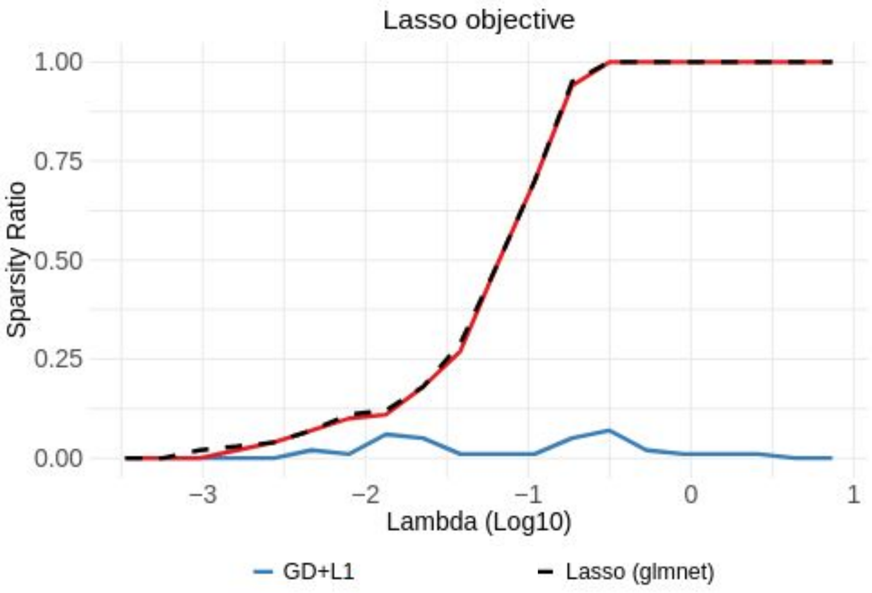
  by smooth $||\boldsymbol{u}||_2^2 + ||\boldsymbol{v}||_2^2$

$$\frac{1}{n}||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_1 \longrightarrow \frac{1}{n}||\boldsymbol{y} - \boldsymbol{X}(\boldsymbol{u} \odot \boldsymbol{v})||_2^2 + \lambda(||\boldsymbol{u}||_2^2 + ||\boldsymbol{v}||_2^2)$$

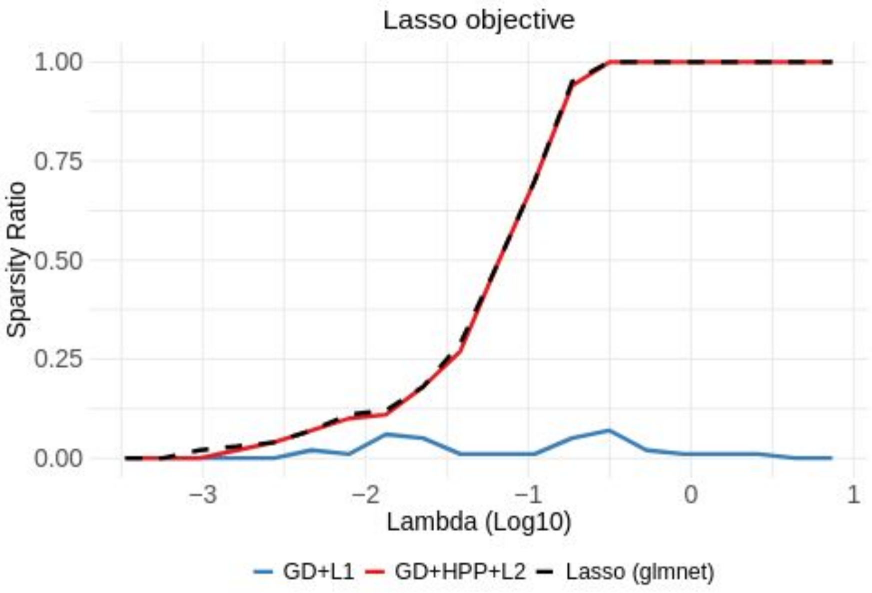
⇒ Optimal solution is the same,
& introduces no additional local minima

General guarantees: Theorem 2.10 in Kolb et al., 2023

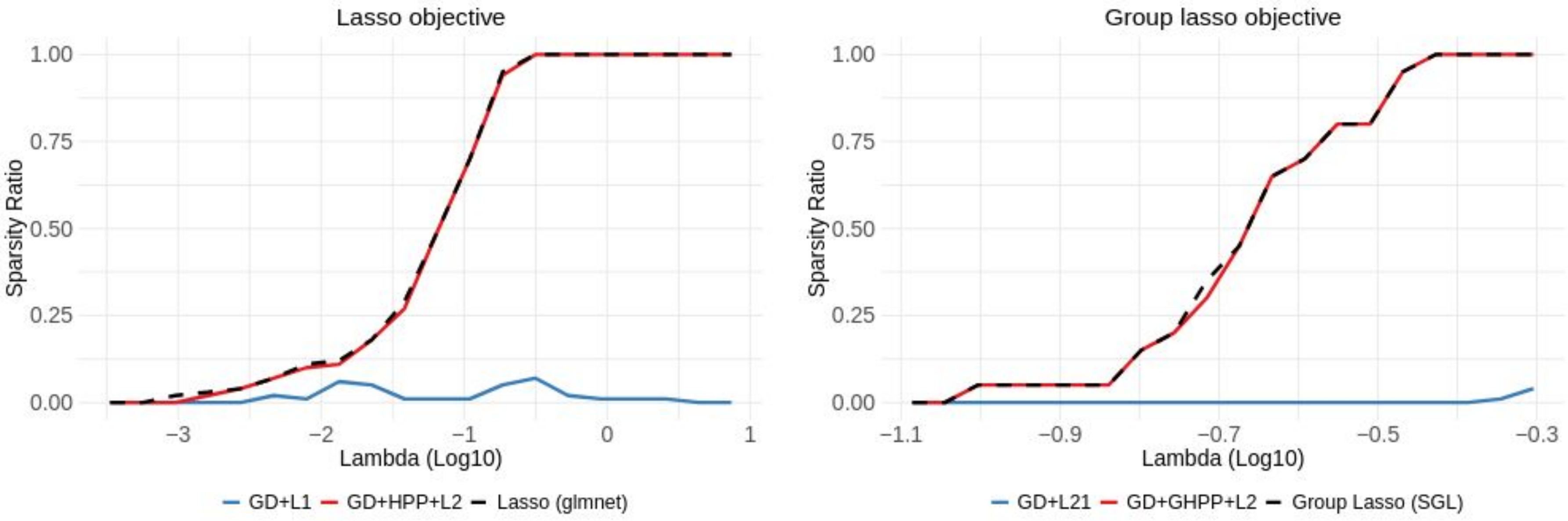
Comparison with Proximal-type Routines



Comparison with Proximal-type Routines



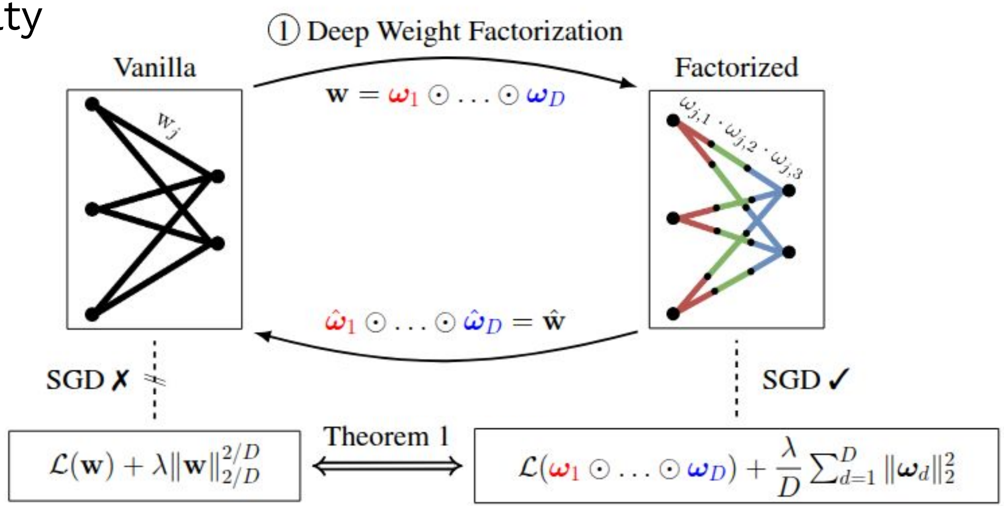
Comparison with Proximal-type Routines



tl;dl: Sparsity

When using (S)GD-type optimization

- sparsity penalties doesn't work
- use a smooth surrogate penalty



Deep Weight Factorization: Sparse Learning Through the Lens of Artificial Symmetries
Kolb, Weber, Bischl, DR, ICLR 2025

Current & Future: Statistical Inference

Statistical Inference

Being Bayesian helps ...

Statistical Inference

Being Bayesian helps ...

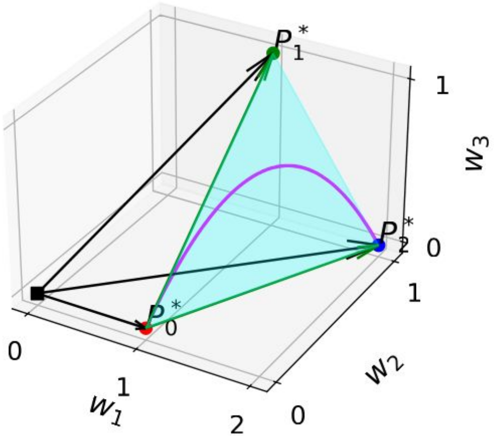
- Subspace Inference
  - ➢ Approximate Deep NN part using a subspace

Bayesian Semi-structured Subspace Inference
Dold, Dr, et al., AIStats 2024
Paths and Ambient Spaces in Neural Loss Landscapes
Dold, ... DR, Dürr, AIStats 2025

Statistical Inference

Being Bayesian helps ...

- Subspace Inference
  - ➢ Approximate Deep NN part using a subspace

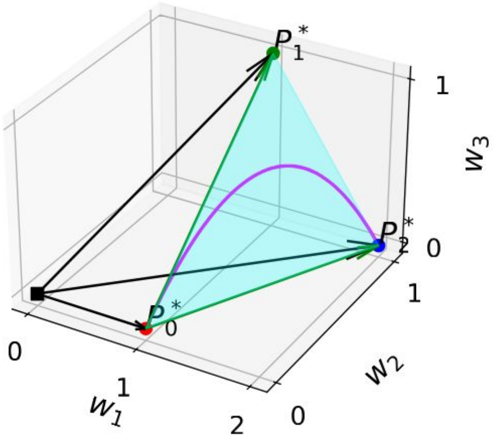


Bayesian Semi-structured Subspace Inference
Dold, Dr, et al., AIStats 2024
Paths and Ambient Spaces in Neural Loss Landscapes
Dold, ... DR, Dürr, AIStats 2025

Statistical Inference

Being Bayesian helps ...

- Subspace Inference
  - ➢ Approximate Deep NN part using a subspace



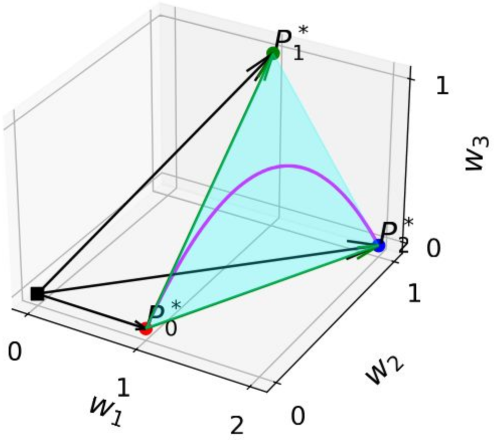
$$\eta = x^\top \beta + \overbrace{\underbrace{\mathrm{NN}_\phi(z_{img})}_{u_{img}} + \underbrace{\mathrm{NN}_\xi(z_{text})}_{u_{text}}}$$

Bayesian Semi-structured Subspace Inference
Dold, Dr, et al., AIStats 2024
Paths and Ambient Spaces in Neural Loss Landscapes
Dold, ... DR, Dürr, AIStats 2025

Statistical Inference

Being Bayesian helps ...

- Subspace Inference
  - ➢ Approximate Deep NN part using a subspace
  - ➢ For small enough subspace,
    common sampling approaches possible



$$\eta = x^\top \beta + \underbrace{\mathrm{NN}_\phi(z_{img})}_{u_{img}} + \underbrace{\mathrm{NN}_\xi(z_{text})}_{u_{text}}$$

Bayesian Semi-structured Subspace Inference
Dold, Dr, et al., AIStats 2024
Paths and Ambient Spaces in Neural Loss Landscapes
Dold, ... DR, Dürr, AIStats 2025

Statistical Inference

Being Bayesian helps ...

- Subspace Inference
  - ➢ Approximate Deep NN part using a subspace
  - ➢ For small enough subspace,
    common sampling approaches possible
  - ➢ Unbiasedly estimates structured parameters

$$\eta = x^\top \beta + \underbrace{\mathrm{NN}_\phi(z_{img})}_{u_{img}} + \underbrace{\mathrm{NN}_\xi(z_{text})}_{u_{text}}$$

Bayesian Semi-structured Subspace Inference
Dold, Dr, et al., AIStats 2024
Paths and Ambient Spaces in Neural Loss Landscapes
Dold, ... DR, Dürr, AIStats 2025

Statistical Inference

Being Bayesian helps ...

- Subspace Inference
  - ➢ Approximate Deep NN part using a subspace
  - ➢ For small enough subspace, common sampling approaches possible
  - ➢ Unbiasedly estimates structured parameters
- Full-Space Inference

Statistical Inference

Being Bayesian helps ...

- Subspace Inference
  - ➢ Approximate Deep NN part using a subspace
  - ➢ For small enough subspace,
    common sampling approaches possible
  - ➢ Unbiasedly estimates structured parameters
- Full-Space Inference
  - ➢ Deemed to be too expensive

Statistical Inference

Being Bayesian helps ...

- Subspace Inference
  - ➢ Approximate Deep NN part using a subspace
  - ➢ For small enough subspace,
    common sampling approaches possible
  - ➢ Unbiasedly estimates structured parameters
- Full-Space Inference
  - ➢ Deemed to be too expensive

Towards Efficient MCMC Sampling in Bayesian Neural Networks by Exploiting Symmetry
Wiese, et al., DR, ECML 2023
Connecting the Dots: Is Mode-Connectedness the Key to Feasible Sample-Based Inference in Bayesian Neural Networks?
Sommer, et al., DR, ICML 2024
Microcanonical Langevin Ensembles: Advancing the Sampling of Bayesian Neural Networks
Sommer et al., DR, ICLR 2025

Statistical Inference

Being Bayesian helps ...

- Subspace Inference
  - ➢ Approximate Deep NN part using a subspace
  - ➢ For small enough subspace, common sampling approaches possible
  - ➢ Unbiasedly estimates structured parameters
- Full-Space Inference
  - ➢ Deemed to be too expensive
  - ➢ $\mathcal{O}$(Sampling) = $\mathcal{O}$(Optimization)

Towards Efficient MCMC Sampling in Bayesian Neural Networks by Exploiting Symmetry
Wiese, et al., DR, ECML 2023
Connecting the Dots: Is Mode-Connectedness the Key to Feasible Sample-Based Inference in Bayesian Neural Networks?
Sommer, et al., DR, ICML 2024
Microcanonical Langevin Ensembles: Advancing the Sampling of Bayesian Neural Networks
Sommer et al., DR, ICLR 2025

Statistical Inference

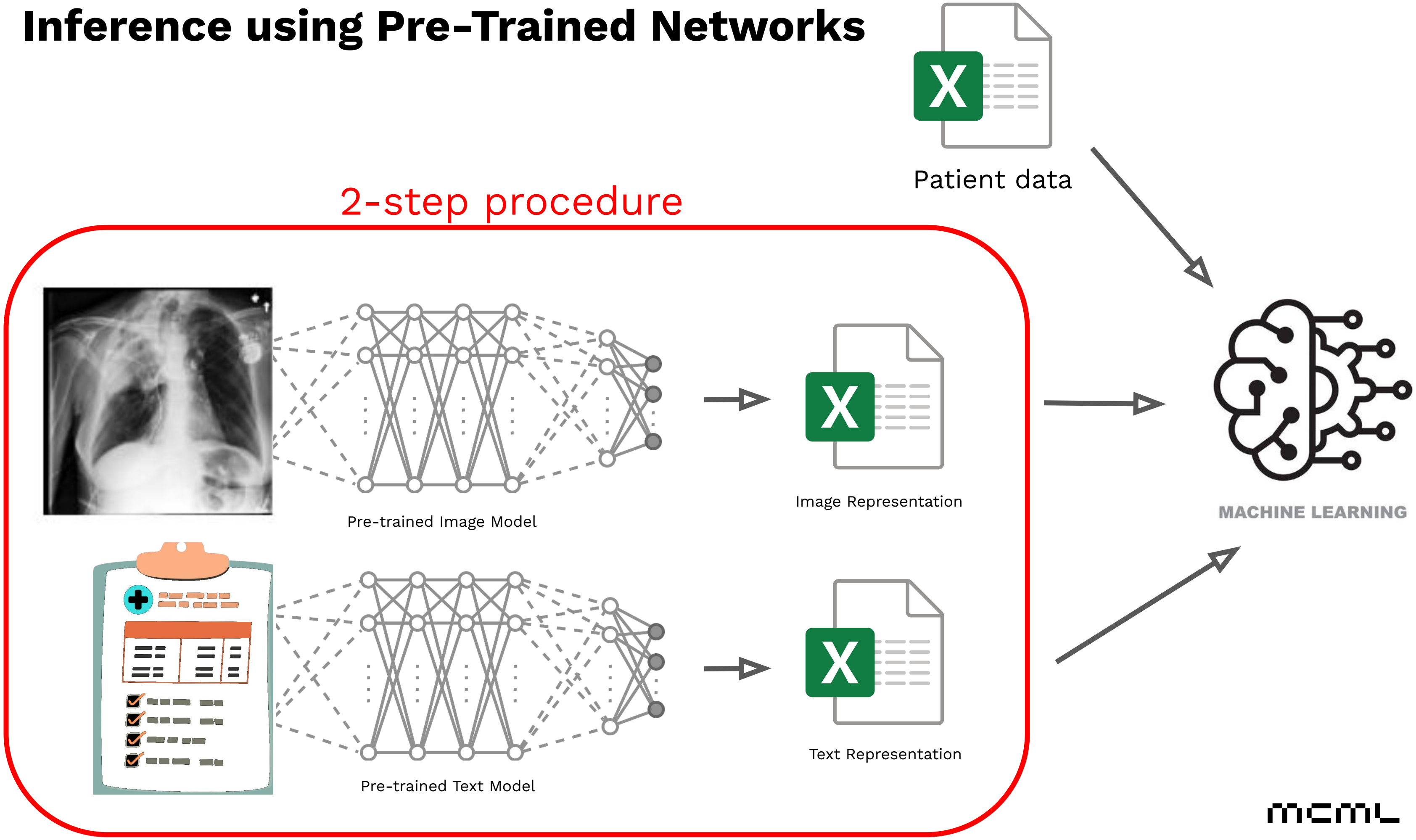
What about frequentist inference?

Statistical Inference

What about frequentist inference?

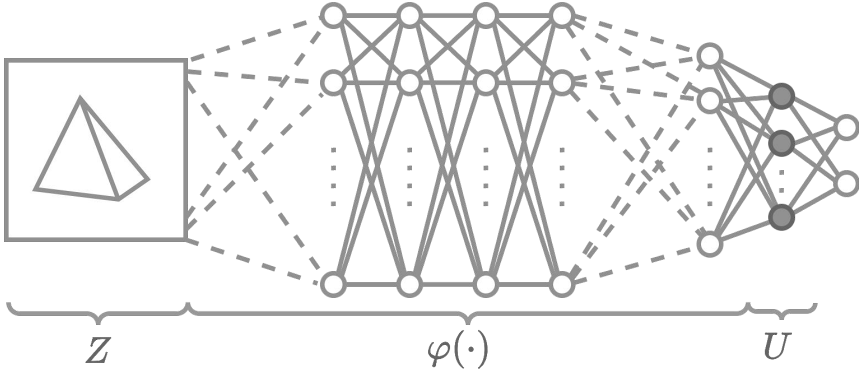
- Challenging
- Pre-trained?

Inference using Pre-Trained Networks



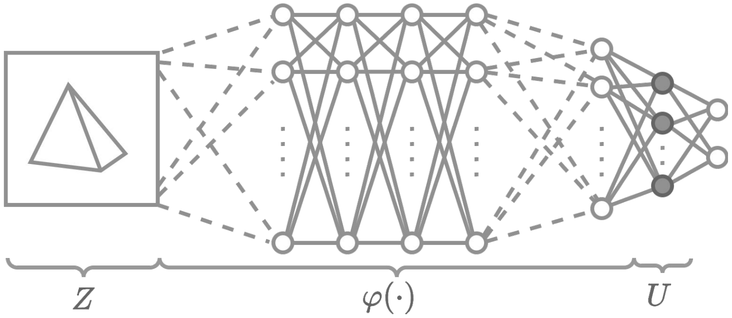
Statistical Inference with Non-tabular data

... using representations from **pre-trained** networks with $U = \varphi(Z)$



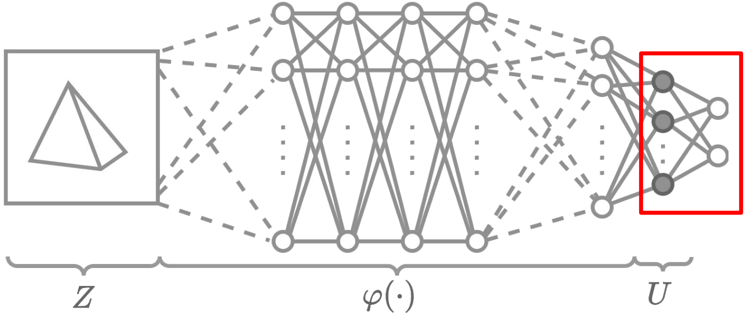
Non-Identifiability of Pre-trained Representations

- Even if relevant information is contained in *U*
- Representations typically not identifiable

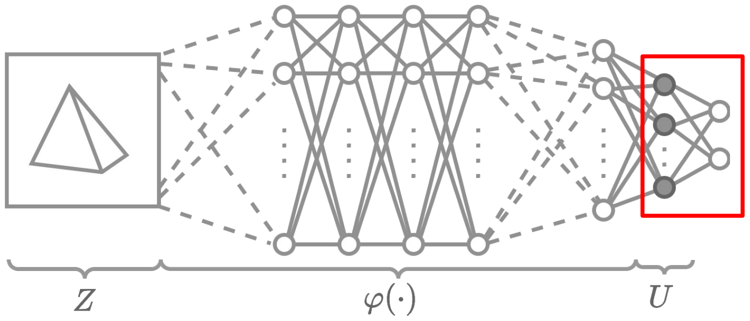


Non-Identifiability of Pre-trained Representations

- Even if relevant information is contained in *U*
- Representations typically not identifiable
- In the model head information does not change under bijective transformations



Non-Identifiability of Pre-trained Representations



- Even if relevant information is contained in U
- Representations typically not identifiable
- In the model head information does not change under bijective transformations

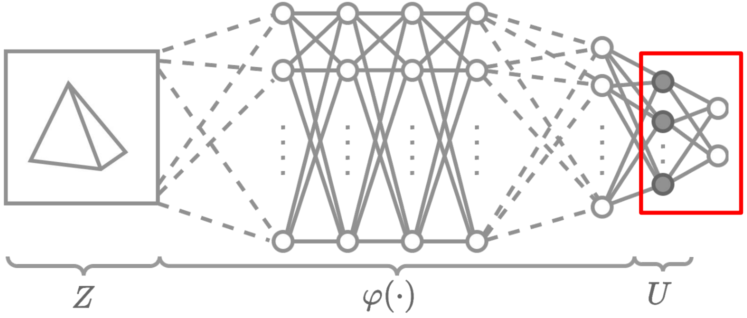
$$\text{head}(U) = \phi(AU + b)$$

Non-Identifiability of Pre-trained Representations

- Even if relevant information is contained in U
- Representations typically not identifiable
- In the model head information does not change under bijective transformations

$$U \mapsto QU$$

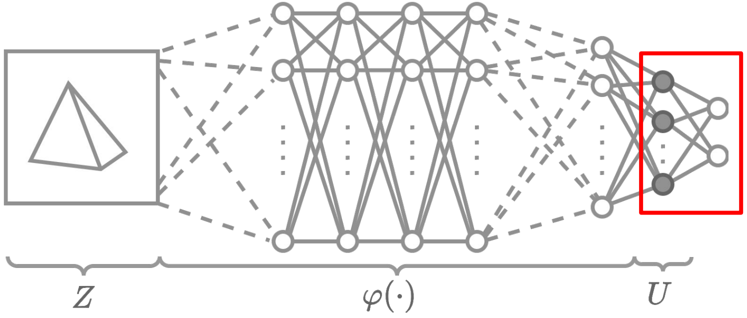
$$\text{head}(U) = \phi(AU + b)$$



Non-Identifiability of Pre-trained Representations

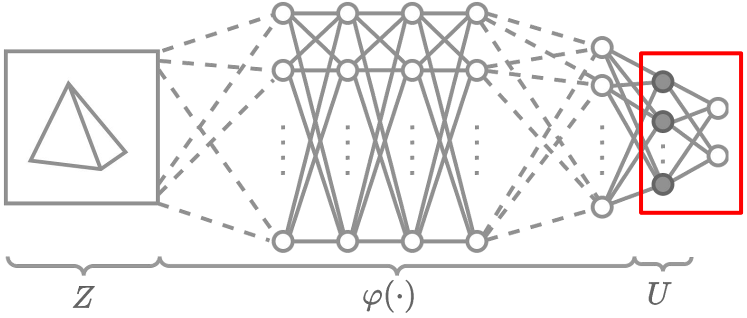
- Even if relevant information is contained in U
- Representations typically not identifiable
- In the model head information does not change under bijective transformations

$$U \mapsto QU$$



$$\text{head}(U) = \phi(AU + b) = \phi((AQ^{-1})QU + b)$$

Non-Identifiability of Pre-trained Representations



- Even if relevant information is contained in U
- Representations typically not identifiable
- In the model head information does not change under bijective transformations

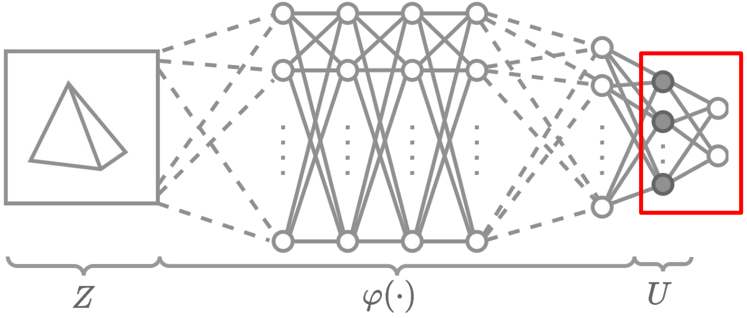
$$U \mapsto QU$$

$$\text{head}(U) = \phi(AU + b) = \phi(\underbrace{(AQ^{-1})}_{\tilde{A}}QU + b)$$

Non-Identifiability of Pre-trained Representations

- Even if relevant information is contained in U
- Representations typically not identifiable
- In the model head information does not change under bijective transformations

$$U \mapsto QU$$



$$\text{head}(U) = \phi(AU + b) = \phi(\underbrace{(AQ^{-1})}_{\tilde{A}}QU + b)$$

➡ Assumptions should hold for Inverse Linear Transformations (ILTs)

Convergence Rates and Invariances

Smoothness	+ Additivity	+ Sparsity & Linearity	Intrinsic Dimension
Stone (1982)	Stone (1985)	Raskutti et al. (2009)	Bickel & Li (2007)
$O(n^{-\frac{s}{2s+d}})$	$O(n^{-\frac{s}{2s+1}})$	$O(\sqrt{p\log(d)/n}),\, p \ll d$	$O(n^{-\frac{s}{2s+d_{\mathcal{M}}}}),\, d_{\mathcal{M}} \ll d$

Table 1. Assumptions and related minimax convergence rates of the estimation error

Convergence Rates and Invariances

Smoothness	+ Additivity	+ Sparsity & Linearity	Intrinsic Dimension
Stone (1982)	Stone (1985)	Raskutti et al. (2009)	Bickel & Li (2007)
$O(n^{-\frac{s}{2s+d}})$	$O(n^{-\frac{s}{2s+1}})$	$O(\sqrt{p\log(d)/n}), p \ll d$	$O(n^{-\frac{s}{2s+d_{\mathcal{M}}}}), d_{\mathcal{M}} \ll d$

Table 1. Assumptions and related minimax convergence rates of the estimation error

Lemma 4.2 (Non-Invariance of Additivity and Sparsity under ILTs). *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a function of $x \in \mathbb{R}^d$. We distinguish between two cases:*

(i) ***Additive****: $f(x) = \sum_{j=1}^{d} f_j(x_j)$, with univariate functions $f_j : \mathbb{R} \to \mathbb{R}$, and at least one f_j being non-linear.*

(ii) ***Sparse Linear****: $f(x) = \sum_{j=1}^{d} \beta_j x_j$, where $\beta_j \in \mathbb{R}$ and at least one (but not all) $\beta_j = 0$.*

Then, for almost every Q drawn from the Haar measure on the set of ILTs, it holds:

(i) If f is additive, then $h = f \circ Q^{-1}$ is not additive.

(ii) If f is sparse linear, then $h = f \circ Q^{-1}$ is not sparse.

❌

Convergence Rates and Invariances

Smoothness	+ Additivity	+ Sparsity & Linearity	Intrinsic Dimension
Stone (1982)	Stone (1985)	Raskutti et al. (2009)	Bickel & Li (2007)
$O(n^{-\frac{s}{2s+d}})$	$O(n^{-\frac{s}{2s+1}})$	$O(\sqrt{p \log(d)/n}), p \ll d$	$O(n^{-\frac{s}{2s+d_{\mathcal{M}}}}), d_{\mathcal{M}} \ll d$

Table 1. Assumptions and related minimax convergence rates of the estimation error

Lemma 4.2 (Non-Invariance of Additivity and Sparsity under ILTs). *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a function of $x \in \mathbb{R}^d$. We distinguish between two cases:*

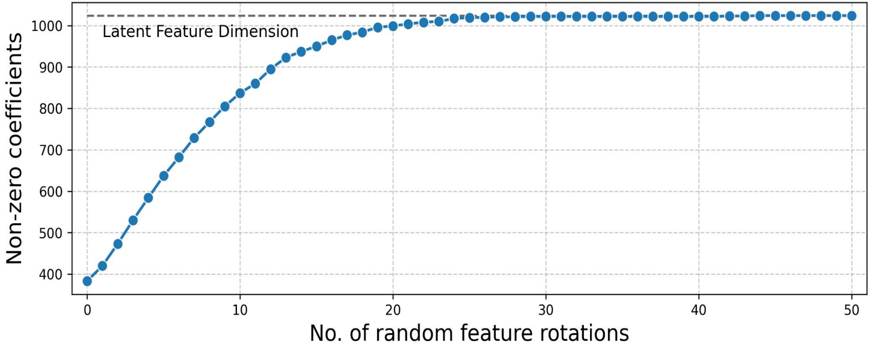
(i) ***Additive****: $f(x) = \sum_{j=1}^d f_j(x_j)$, with univariate functions $f_j : \mathbb{R} \to \mathbb{R}$, and at least one f_j being non-linear.*

(ii) ***Sparse Linear****: $f(x) = \sum_{j=1}^d \beta_j x_j$, where $\beta_j \in \mathbb{R}$ and at least one (but not all) $\beta_j = 0$.*

Then, for almost every Q drawn from the Haar measure on the set of ILTs, it holds:

(i) If f is additive, then $h = f \circ Q^{-1}$ is not additive.

(ii) If f is sparse linear, then $h = f \circ Q^{-1}$ is not sparse.



Convergence Rates and Invariances

Smoothness	+ Additivity	+ Sparsity & Linearity	Intrinsic Dimension
Stone (1982)	Stone (1985)	Raskutti et al. (2009)	Bickel & Li (2007)
$O(n^{-\frac{s}{2s+d}})$	$O(n^{-\frac{s}{2s+1}})$	$O(\sqrt{p\log(d)/n}), p \ll d$	$O(n^{-\frac{s}{2s+d_{\mathcal{M}}}}), d_{\mathcal{M}} \ll d$

Table 1. Assumptions and related minimax convergence rates of the estimation error

Smoothness

Lemma 4.1 (Smoothness Invariance under ILTs). *Let $D \subseteq \mathbb{R}^d$ be an open set, $f : D \to \mathbb{R}$ be an s-smooth-function on D, and Q by any ILT. Then $h = f \circ Q^{-1} : Q(D) \to \mathbb{R}$ is also s-smooth on the transformed domain $Q(D)$.*

Idea: Since Q^{-1} is C^∞, and f is C^s, their composition $h = f \circ Q^{-1}$ is C^s. ✅

Convergence Rates and Invariances

Smoothness	+ Additivity	+ Sparsity & Linearity	Intrinsic Dimension
Stone (1982)	Stone (1985)	Raskutti et al. (2009)	Bickel & Li (2007)
$O(n^{-\frac{s}{2s+d}})$	$O(n^{-\frac{s}{2s+1}})$	$O(\sqrt{p\log(d)/n}), p \ll d$	$O(n^{-\frac{s}{2s+d_{\mathcal{M}}}}), d_{\mathcal{M}} \ll d$

Table 1. Assumptions and related minimax convergence rates of the estimation error

Smoothness

Lemma 4.1 (Smoothness Invariance under ILTs). *Let $D \subseteq \mathbb{R}^d$ be an open set, $f : D \to \mathbb{R}$ be an s-smooth-function on D, and Q by any ILT. Then $h = f \circ Q^{-1} : Q(D) \to \mathbb{R}$ is also s-smooth on the transformed domain $Q(D)$.*

Idea: Since Q^{-1} is C^∞, and f is C^s, their composition $h = f \circ Q^{-1}$ is C^s. ✅

Intrinsic Dimensionality (ID)

Lemma 4.3 (Intrinsic Dimension Invariance under ILTs). *Let $\mathcal{M} \subset \mathbb{R}^d$ be a smooth manifold of dimension $d_{\mathcal{M}} \leq d$. For any ILT Q, the transformed set*

$$Q(\mathcal{M}) = \{Qx \mid x \in \mathcal{M}\}.$$

is also a smooth manifold of dimension $d_{\mathcal{M}}$. ✅

Convergence Rates and Invariances

Smoothness	+ Additivity	+ Sparsity & Linearity	Intrinsic Dimension
Stone (1982)	Stone (1985)	Raskutti et al. (2009)	Bickel & Li (2007)
$O(n^{-\frac{s}{2s+d}})$	$O(n^{-\frac{s}{2s+1}})$	$O(\sqrt{p\log(d)/n}), p \ll d$	$O(n^{-\frac{s}{2s+d_{\mathcal{M}}}}), d_{\mathcal{M}} \ll d$

Table 1. Assumptions and related minimax convergence rates of the estimation error

Smoothness

Lemma 4.1 (Smoothness Invariance under ILTs). *Let $D \subseteq \mathbb{R}^d$ be an open set, $f : D \to \mathbb{R}$ be an s-smooth-function on D, and Q by any ILT. Then $h = f \circ Q^{-1} : Q(D) \to \mathbb{R}$ is also s-smooth on the transformed domain $Q(D)$.*

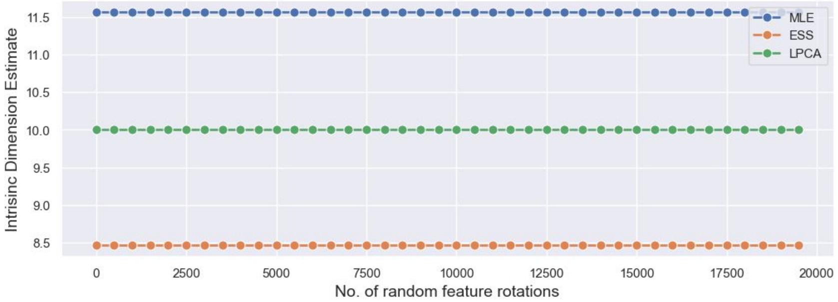
Idea: Since Q^{-1} is C^∞, and f is C^s, their composition $h = f \circ Q^{-1}$ is C^s. ✅

Intrinsic Dimensionality

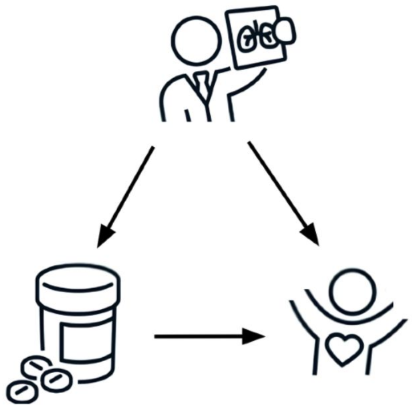
Lemma 4.3 (Intrinsic Dimension Invariance under ILTs). *Let $\mathcal{M} \subset \mathbb{R}^d$ be a smooth manifold of dimension $d_{\mathcal{M}} \leq d$. For any ILT Q, the transformed set*

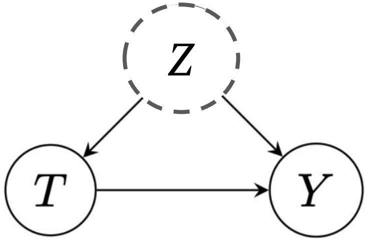
$$Q(\mathcal{M}) = \{Qx \mid x \in \mathcal{M}\}.$$

is also a smooth manifold of dimension $d_{\mathcal{M}}$. ✅

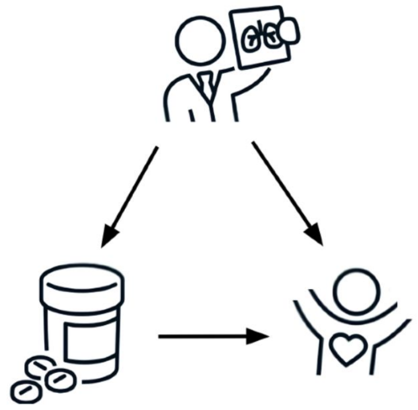


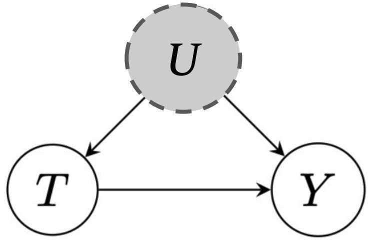
Convergence Rates for Nuisance Function Estimation



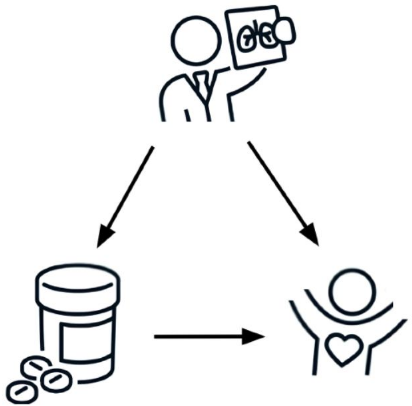


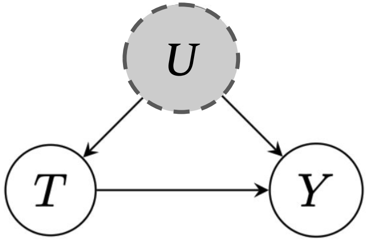
Convergence Rates for Nuisance Function Estimation





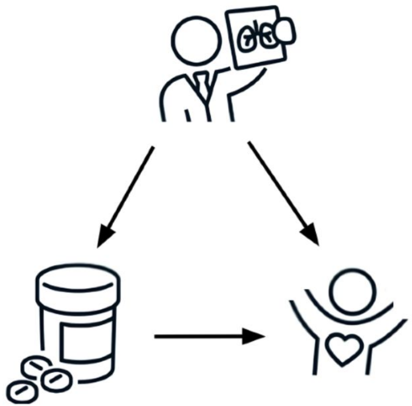
Convergence Rates for Nuisance Function Estimation

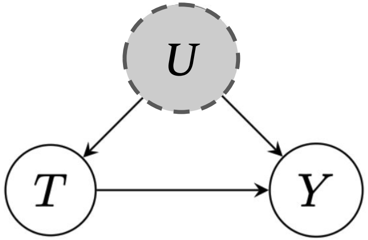




➢ Image data and its latent representations often of low ID

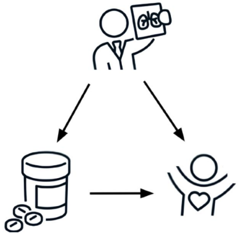
Convergence Rates for Nuisance Function Estimation

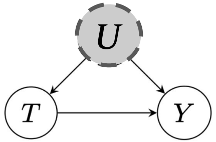




- ➢ Image data and its latent representations often of low ID
- ➢ NN can adapt to low ID and achieve fast conv. rates

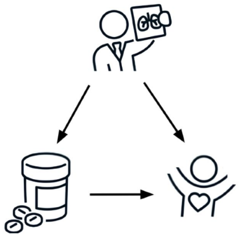
Convergence Rates for Nuisance Function Estimation

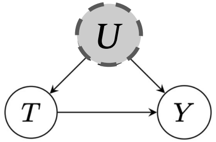




Convergence Rates for Nuisance Function Estimation

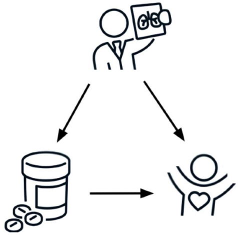
Estimating Average Treatment Effect T → Y using Double Machine Learning (DML)

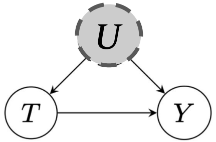


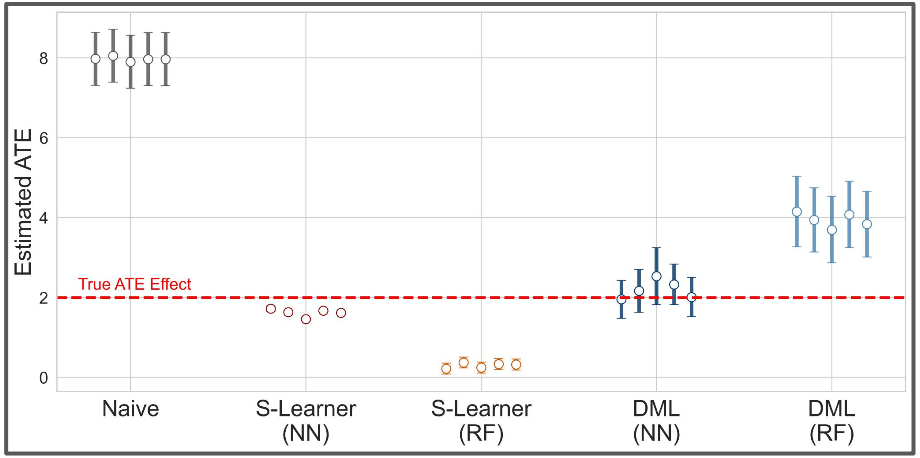


Convergence Rates for Nuisance Function Estimation

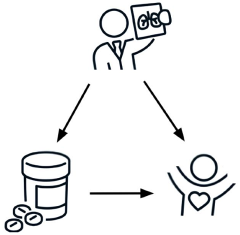
Estimating Average Treatment Effect T → Y using Double Machine Learning (DML)

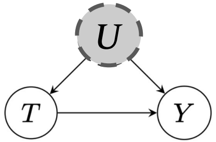






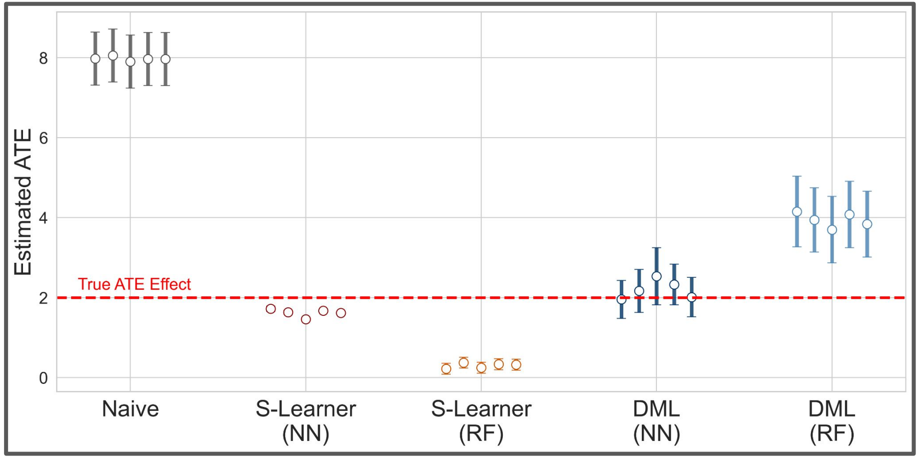
Convergence Rates for Nuisance Function Estimation





Estimating Average Treatment Effect T ➝ Y using Double Machine Learning (DML)

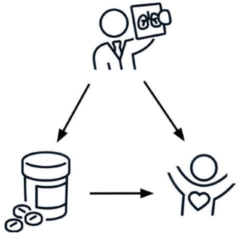
➢ Random Forest fails, Doubly robust estimation superior
(RF: axis-aligned splits not compatible with ILTs)

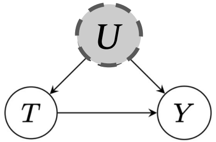


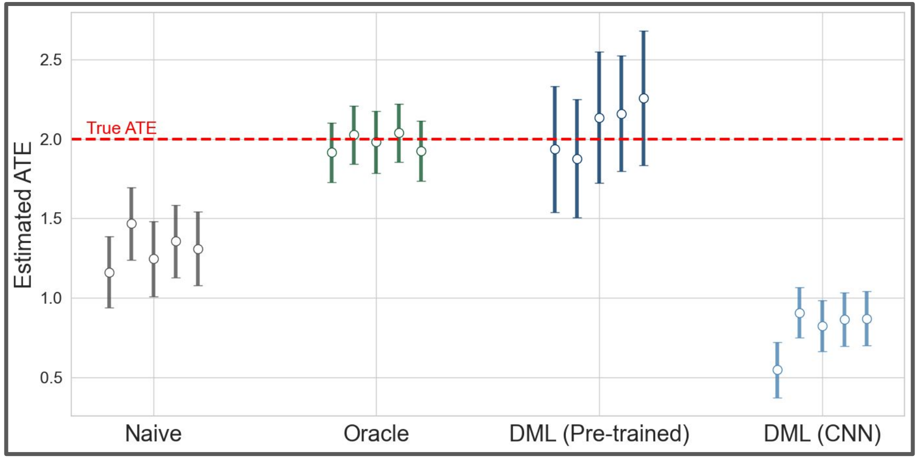
Adjustment for Confounding using Pre-Trained Representations
Schulte, DR, Nagler, 2025

Convergence Rates for Nuisance Function Estimation

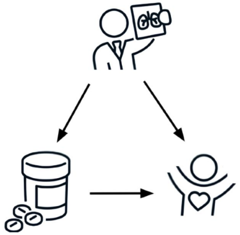
Estimating Average Treatment Effect T → Y using Double Machine Learning (DML)

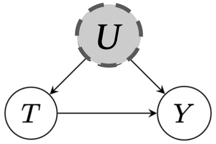






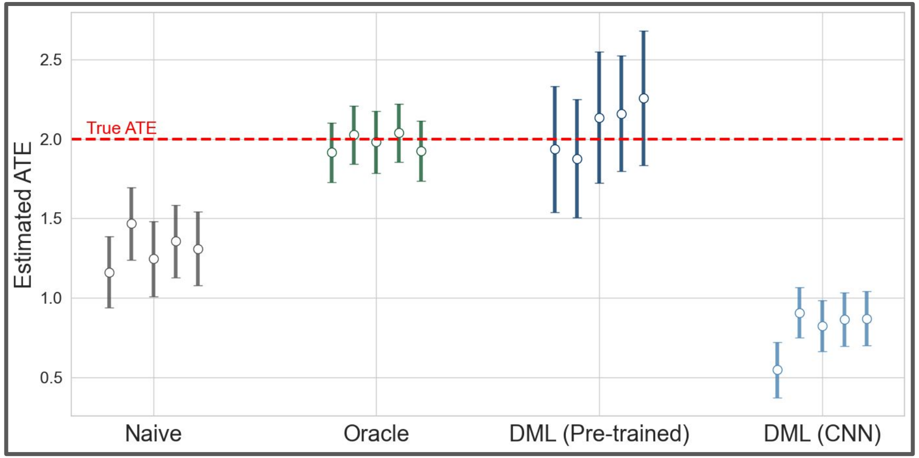
Convergence Rates for Nuisance Function Estimation



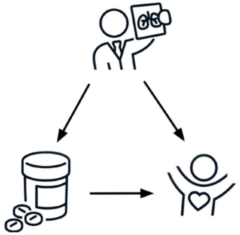


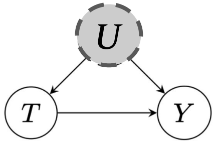
Estimating Average Treatment Effect T → Y using Double Machine Learning (DML)

➢ Pre-trained can work better than from-scratch training (CNN)



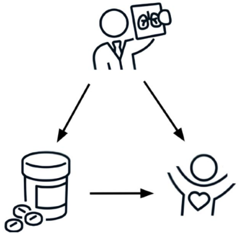
Convergence Rates for Nuisance Function Estimation

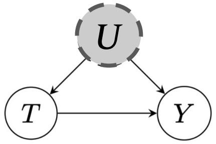




Estimating Average Treatment Effect T → Y using Double Machine Learning (DML)

Convergence Rates for Nuisance Function Estimation

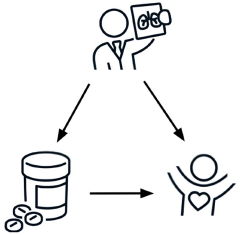


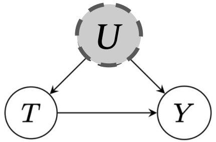


Estimating Average Treatment Effect T → Y using Double Machine Learning (DML)

Assuming
1. Validity of representation
2. Low ID dim. of representation
3. Hierarchical composition of target function

Convergence Rates for Nuisance Function Estimation

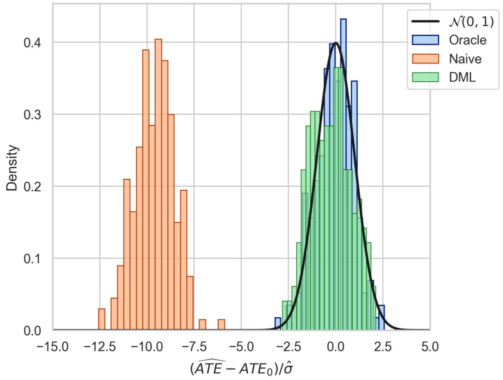




Estimating Average Treatment Effect T → Y using Double Machine Learning (DML)

Assuming
1. Validity of representation
2. Low ID dim. of representation
3. Hierarchical composition of target function

We derive
- ➢ Convergence rates for NN-based (nuisance) estimation
- ➢ Asym. normality for doubly-robust $\widehat{\text{ATE}}$



tl;dl: Statistical Inference

Statistical Inference for semi-structured models

- Bayes POV: possible in subspace and maybe also full space
- Frequentist POV: with some assumptions and pre-trained models

Summary

- **Semi-Structured Regression:**
  Combine statistical models & neural networks

- **Flexibility & Scalability**
  Many model classes & easier to make scalable

- **Sparsity in Neural Networks**
  Can be achieved by an optimization transfer

- **Inference in Semi-Structured Regression**
  Pre-trained models or Bayesian approaches

Joint work with many others ...

Appendix

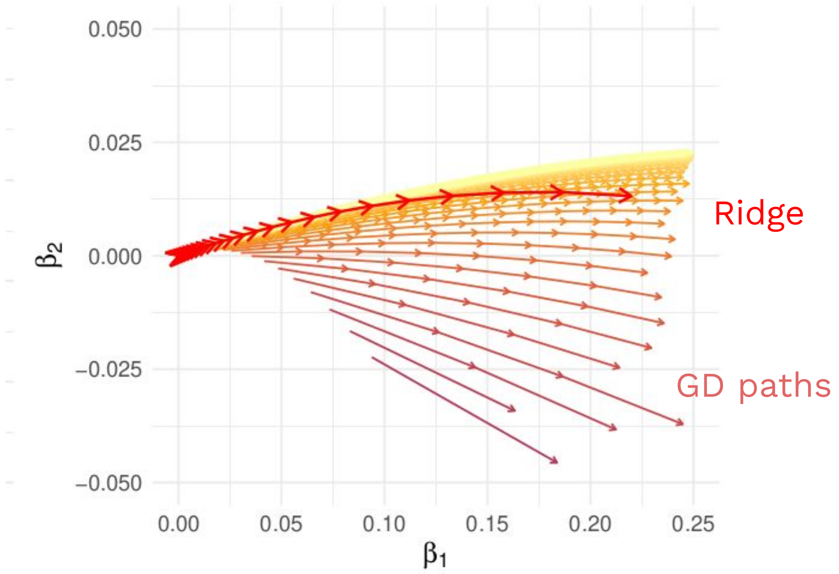
Challenge:
(Implicit) Regularization

(Implicit) Regularization

Working with stochastic gradient descent (SGD)-type optimization can be challenging

(Implicit) Regularization

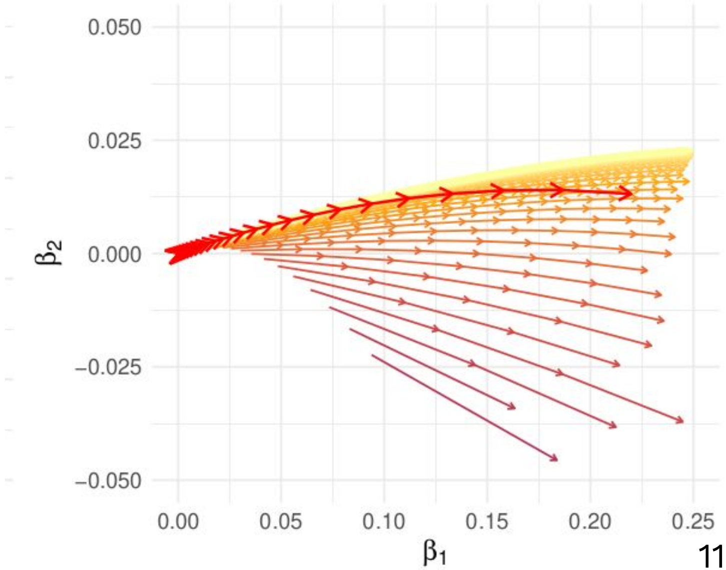
Working with stochastic gradient descent (SGD)-type optimization can be challenging



(Implicit) Regularization

Working with stochastic gradient descent (SGD)-type optimization can be challenging

➢ Implicit regularization of (NN) optimizers behaves differently than classical Ridge-type regularization



Additive Model Boosting: New Insights and Path(ologie)s
Schulte and DR, AIStats 2025

(Implicit) Regularization

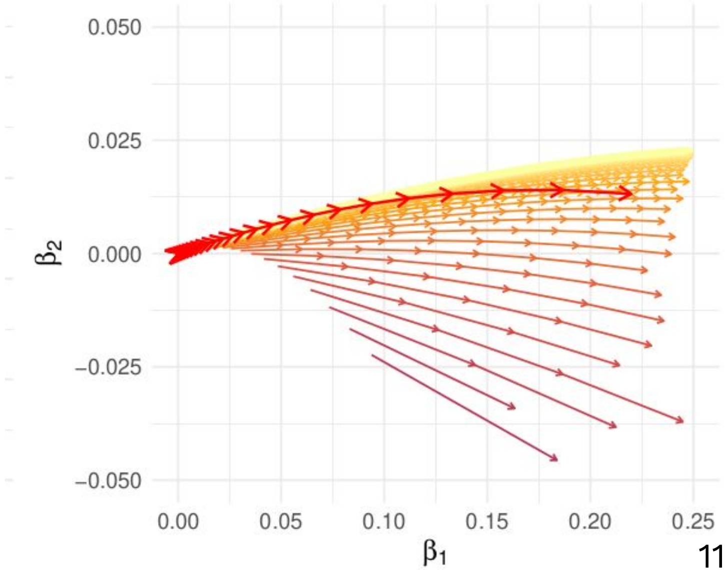
Working with stochastic gradient descent (SGD)-type optimization can be challenging

➢ Implicit regularization of (NN) optimizers behaves differently than classical Ridge-type regularization

Theorem 1. *Given full column rank matrix X, L_2-Boosting with quadratic penalty and joint updates (7) uniquely solves at each iteration $k \in \mathbb{N}$ the explicitly regularized problem*

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{2}\|y - X\beta\|^2 + \frac{1}{2}\beta^\top \Gamma_k \beta, \qquad (8)$$

with $\Gamma_k := (X^\top X)\ S_\lambda^{-1}\ [(I - \nu S_\lambda)^{-k} - I]^{-1}\ S_\lambda$ *as penalty matrix and* $S_\lambda := (X^\top X + \lambda P)^{-1} X^\top X$.



Additive Model Boosting: New Insights and Path(ologie)s
Schulte and DR, AIStats 2025

TL;DL: Regularization

When using (S)GD-type optimization

- there is implicit regularization
- it’s not clear, what we are actually optimizing — even in a linear model