# Speeding-up Hyperparameter Optimization with transfer and meta learning

## Metaheuristics Summer School.

David Salinas. July 2024.

# Goals

- Understand benefit of transfer learning to speed-up HPO
- Understand the key challenges to apply transfer learning to HPO
- Get an idea of the main techniques being used in state-of-the-art methods
- Know how to apply transfer learning to your problem

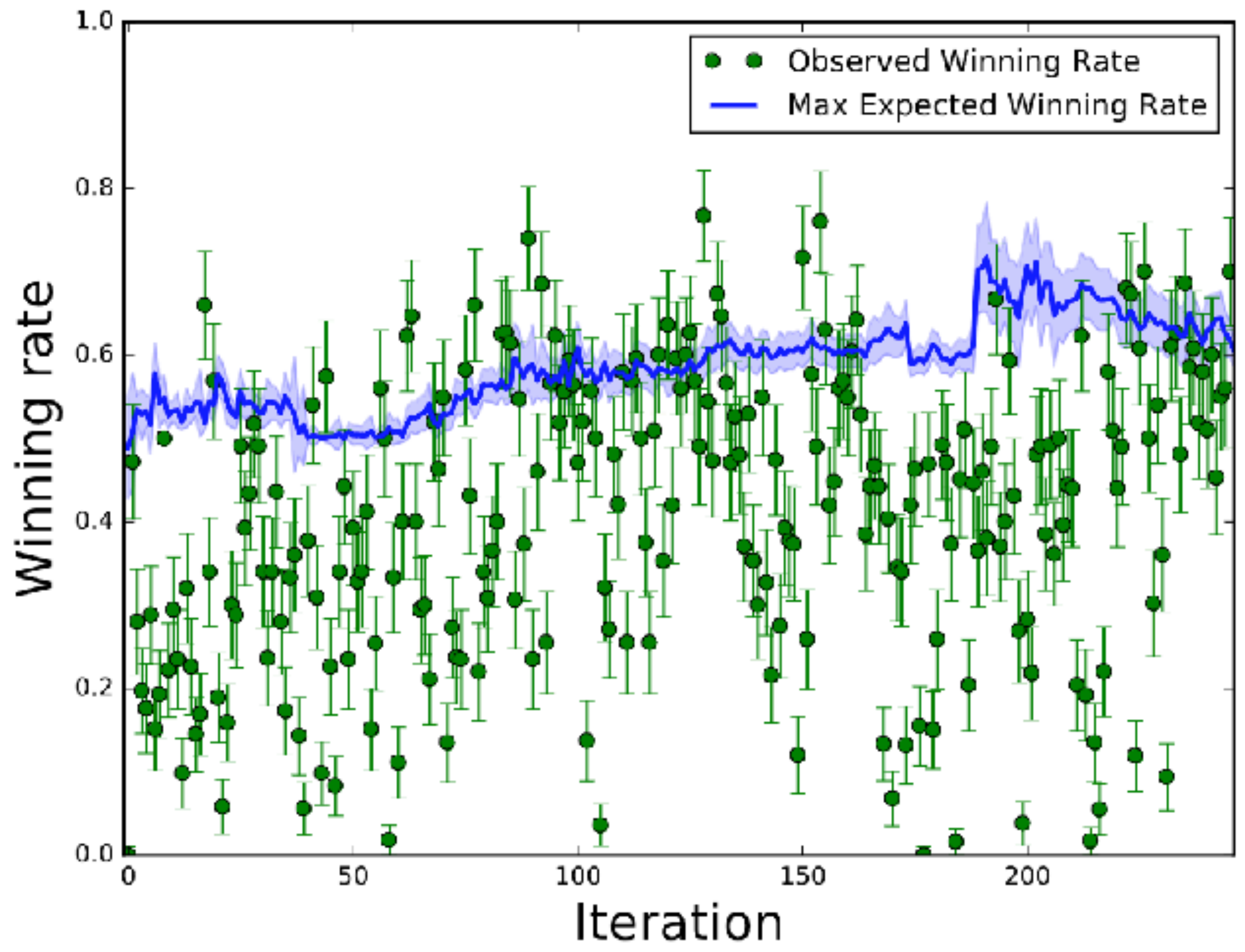
# Introduction

# Introduction

- Hyperoptimization often gives large improvements

# Introduction

- Hyperoptimization often gives large improvements

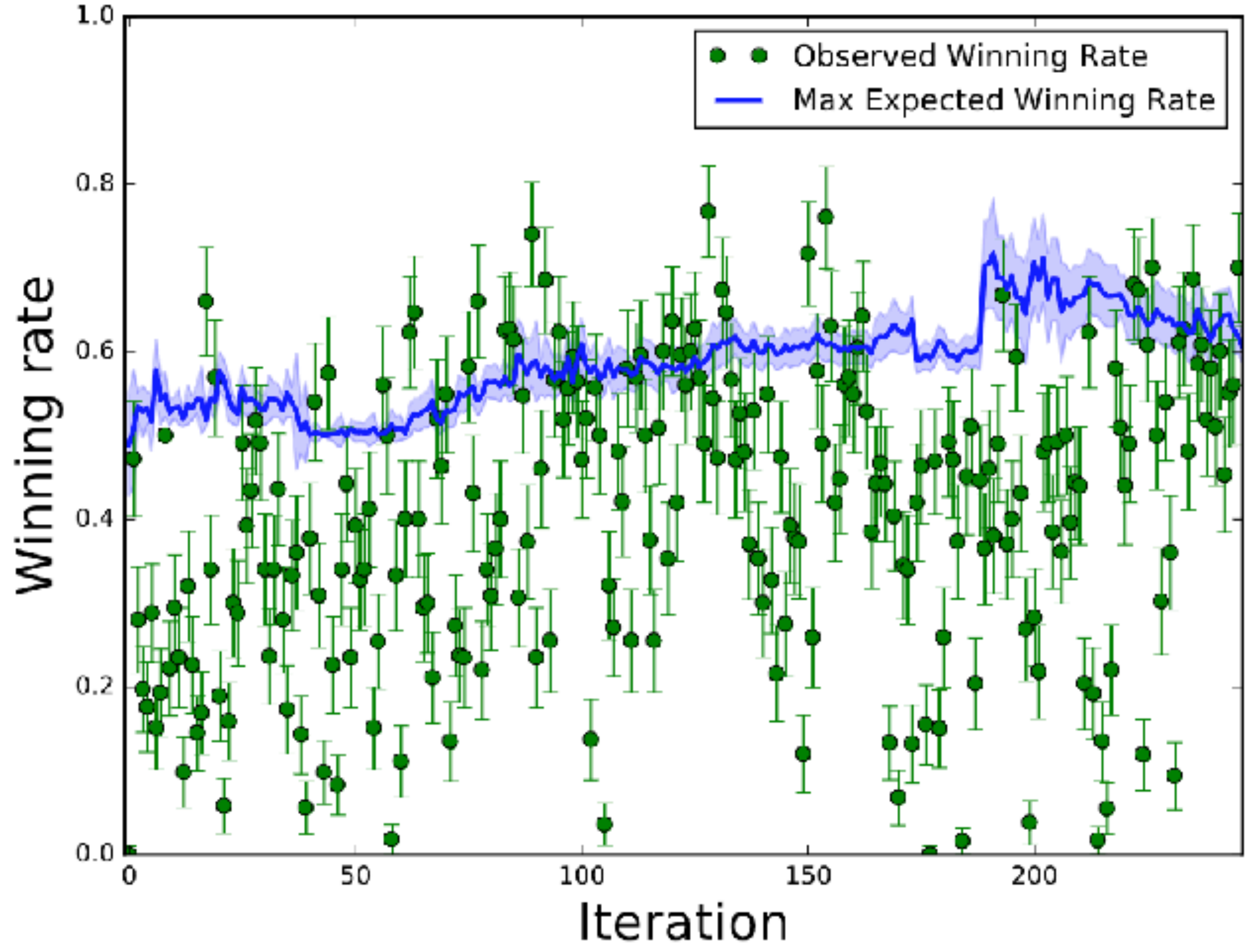



Bayesian Optimization in AlphaGo
[Chen 2018]

# Introduction

- Hyperoptimization often gives large improvements

Bayesian Optimization was key to improve AlphaGo!

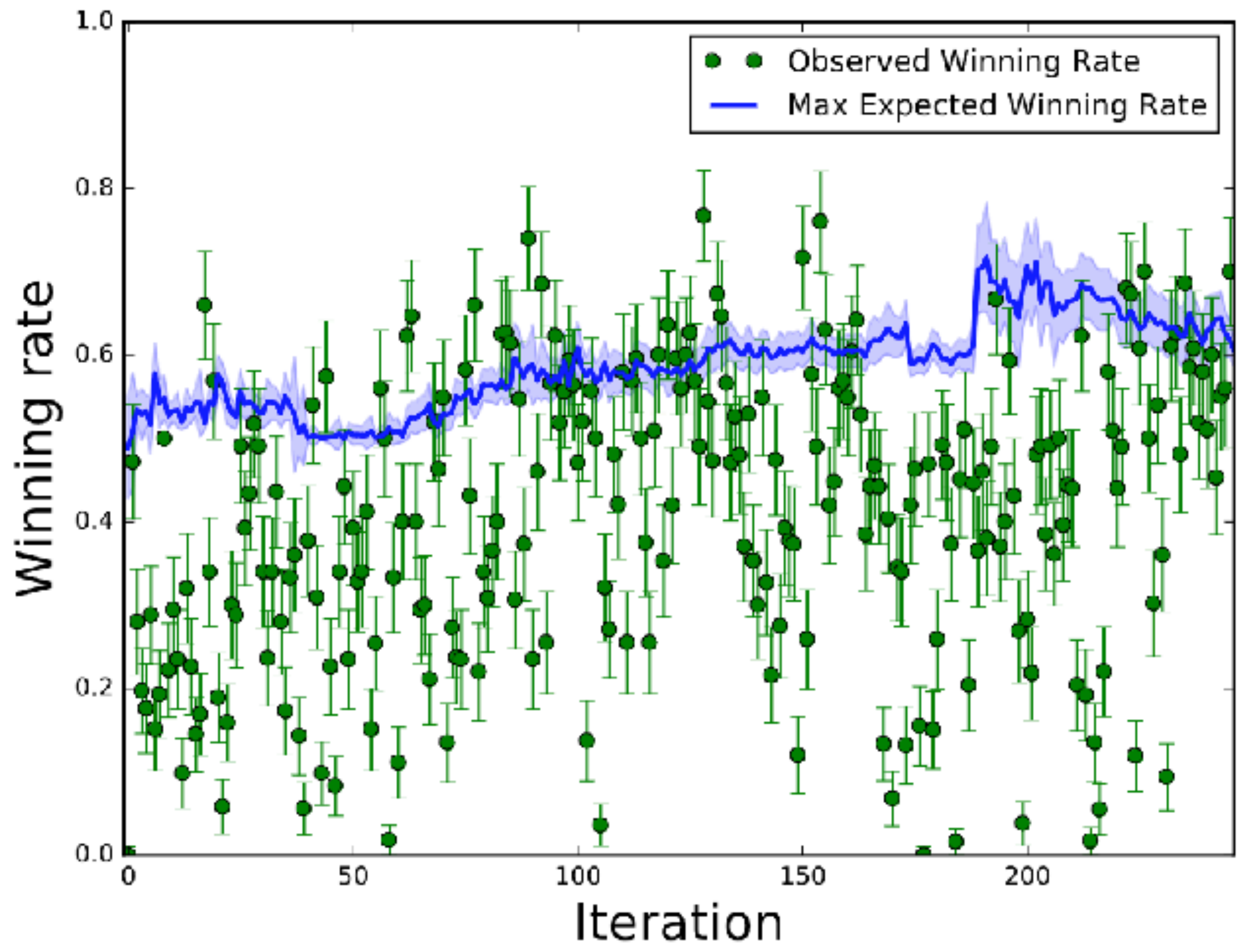



Bayesian Optimization in AlphaGo
[Chen 2018]

# Introduction

- Hyperoptimization often gives large improvements
- But it is not systematically used

Bayesian Optimization was key to improve AlphaGo!

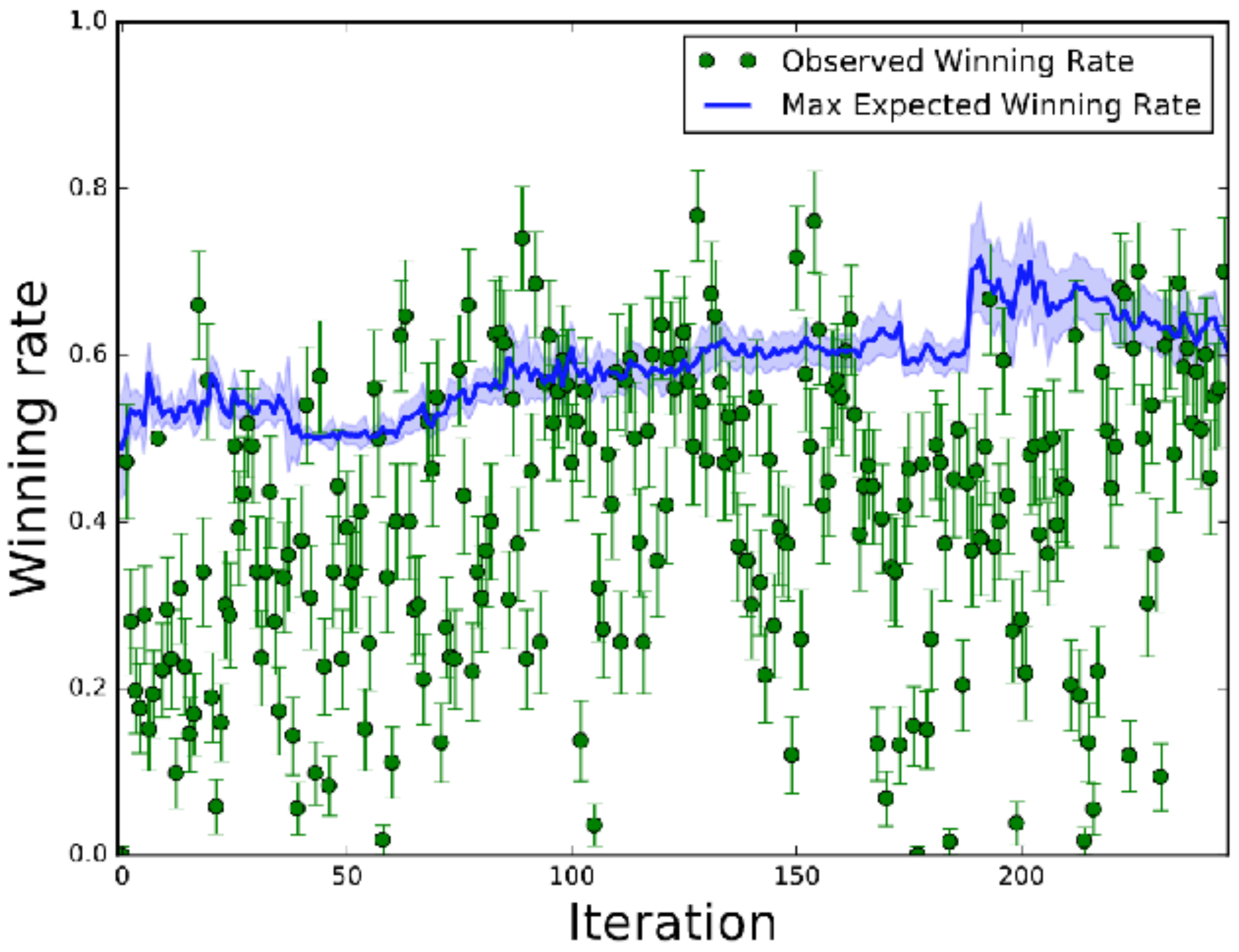



Bayesian Optimization in AlphaGo
[Chen 2018]

# Introduction

- Hyperoptimization often gives large improvements
- But it is not systematically used
- Why?

Bayesian Optimization was key to improve AlphaGo!

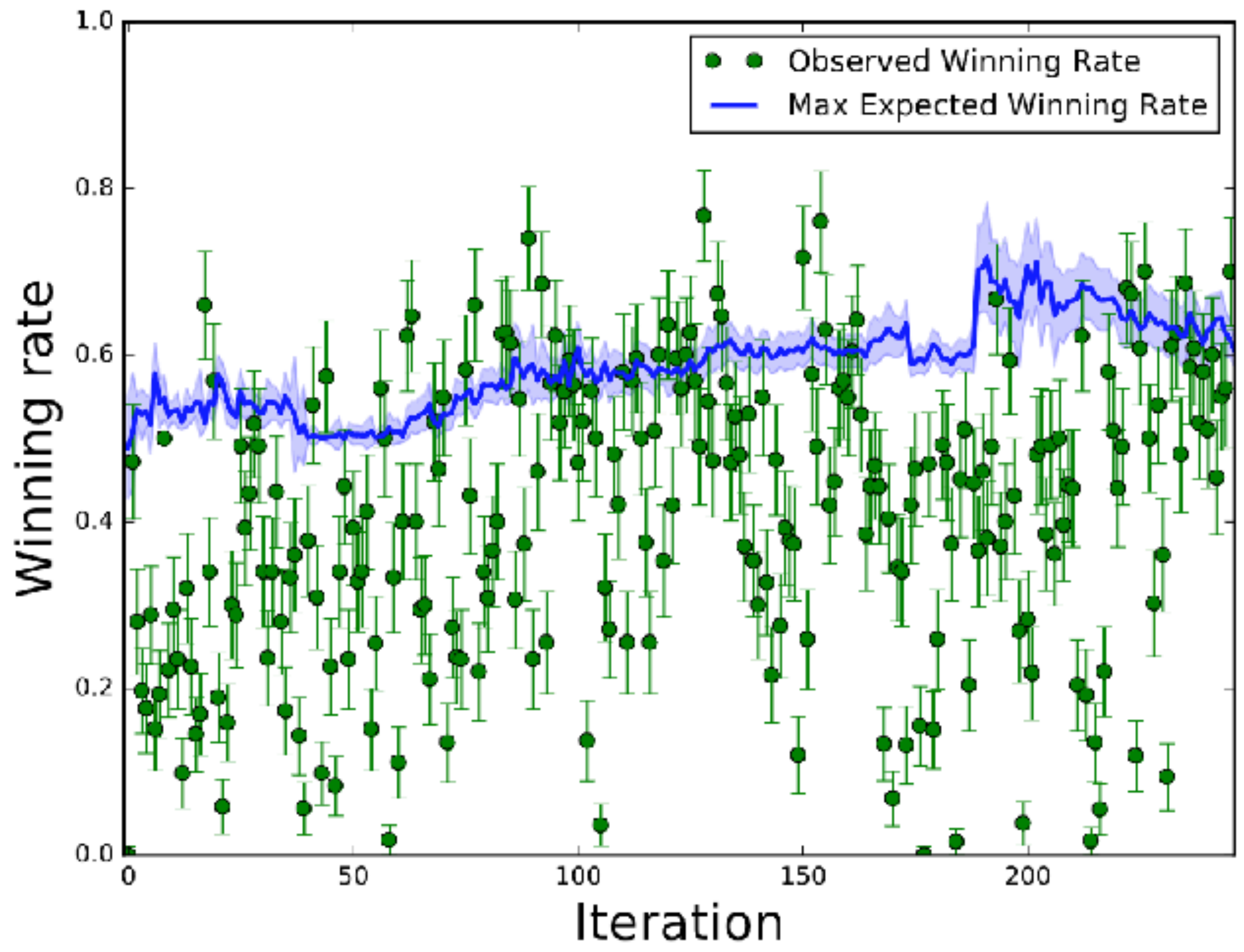



Bayesian Optimization in AlphaGo
[Chen 2018]

# Introduction

- Hyperoptimization often gives large improvements
- But it is not systematically used
- Why?
  - Lack of tool knowledge

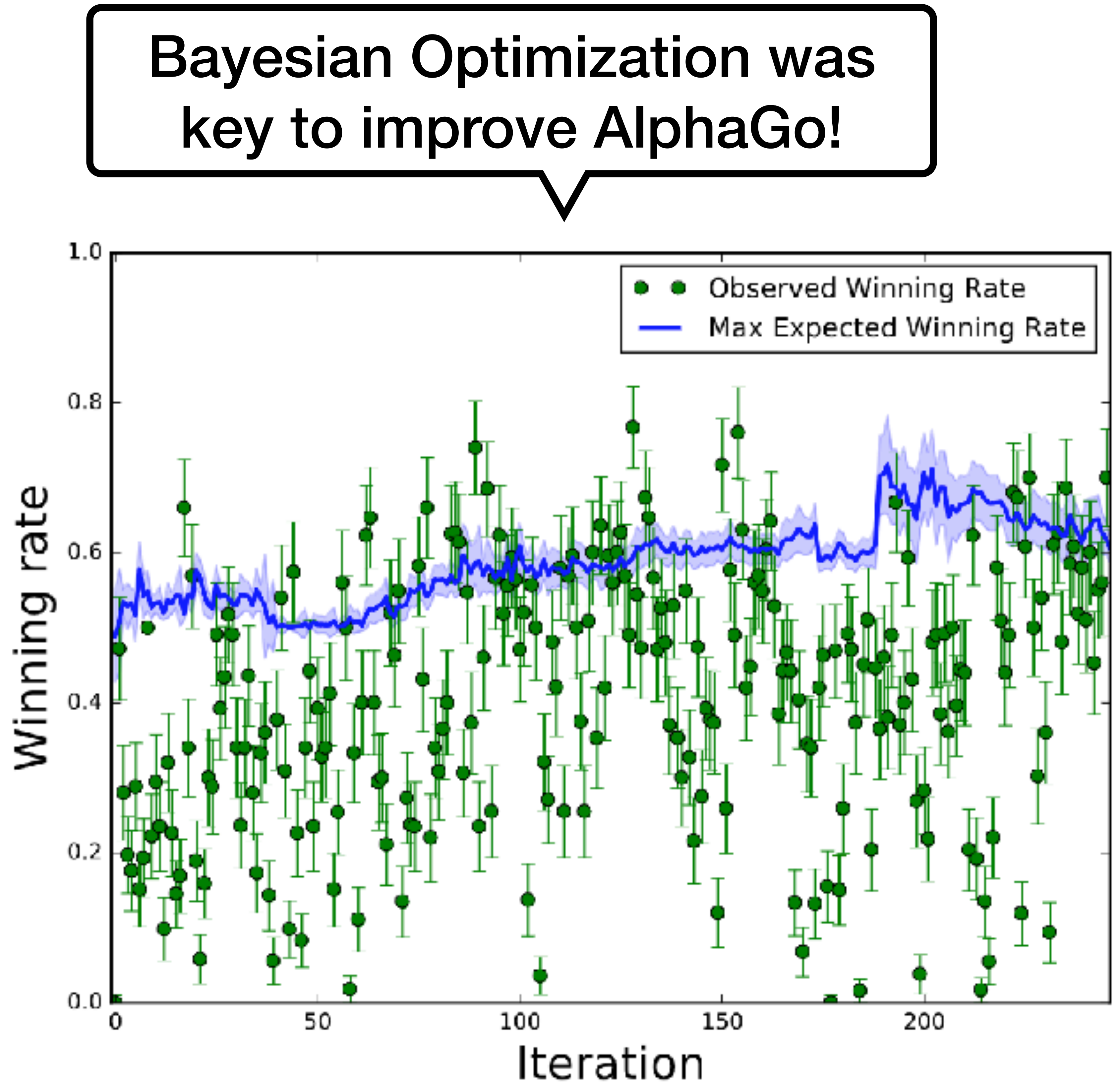
Bayesian Optimization was key to improve AlphaGo!




Bayesian Optimization in AlphaGo
[Chen 2018]

# Introduction

- Hyperoptimization often gives large improvements
- But it is not systematically used
- Why?
  - Lack of tool knowledge
  - Attachement to grid-search interpretability & ease

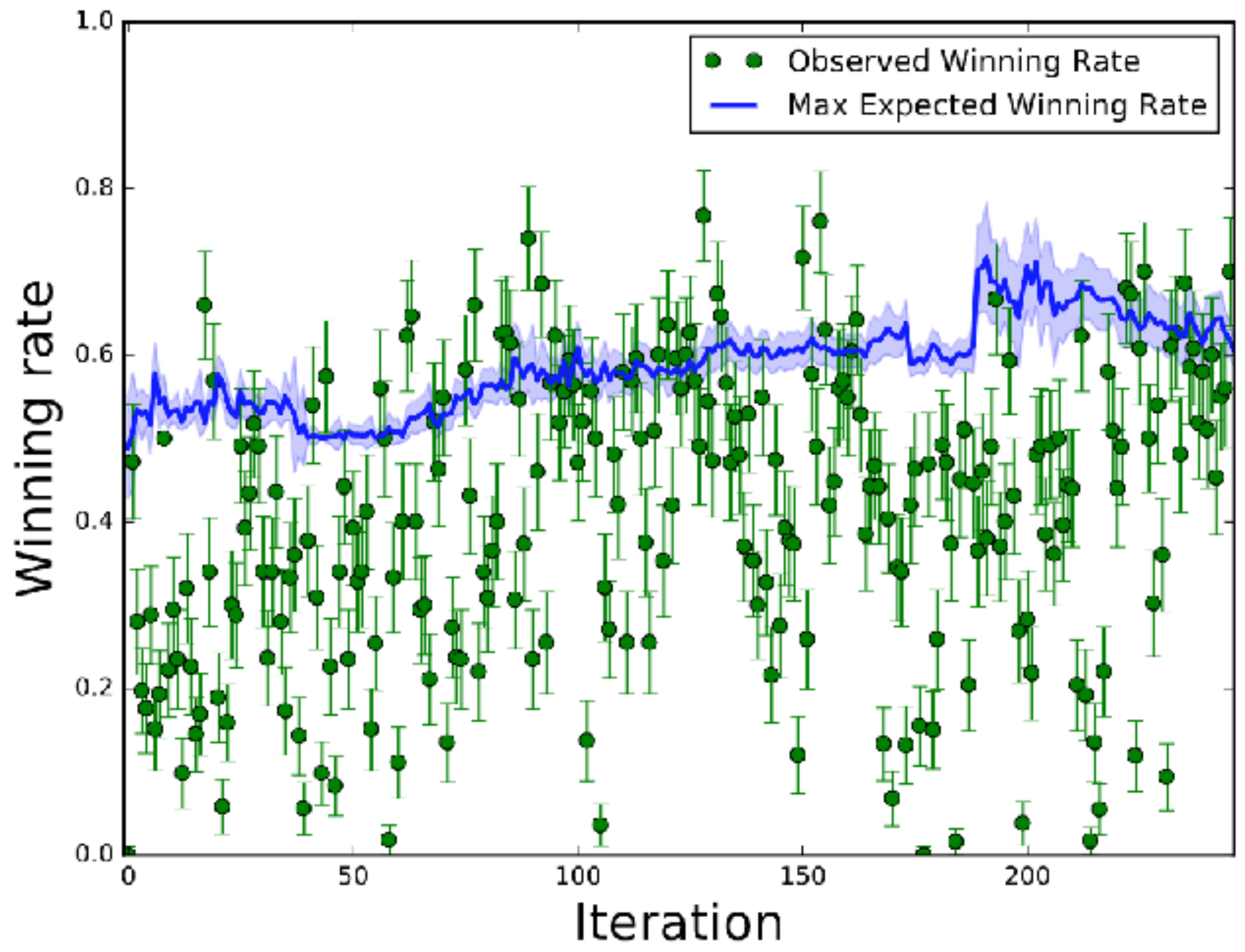



Bayesian Optimization in AlphaGo [Chen 2018]

# Introduction

- Hyperoptimization often gives large improvements
- But it is not systematically used
- Why?
  - Lack of tool knowledge
  - Attachement to grid-search interpretability & ease
  - Can be expensive

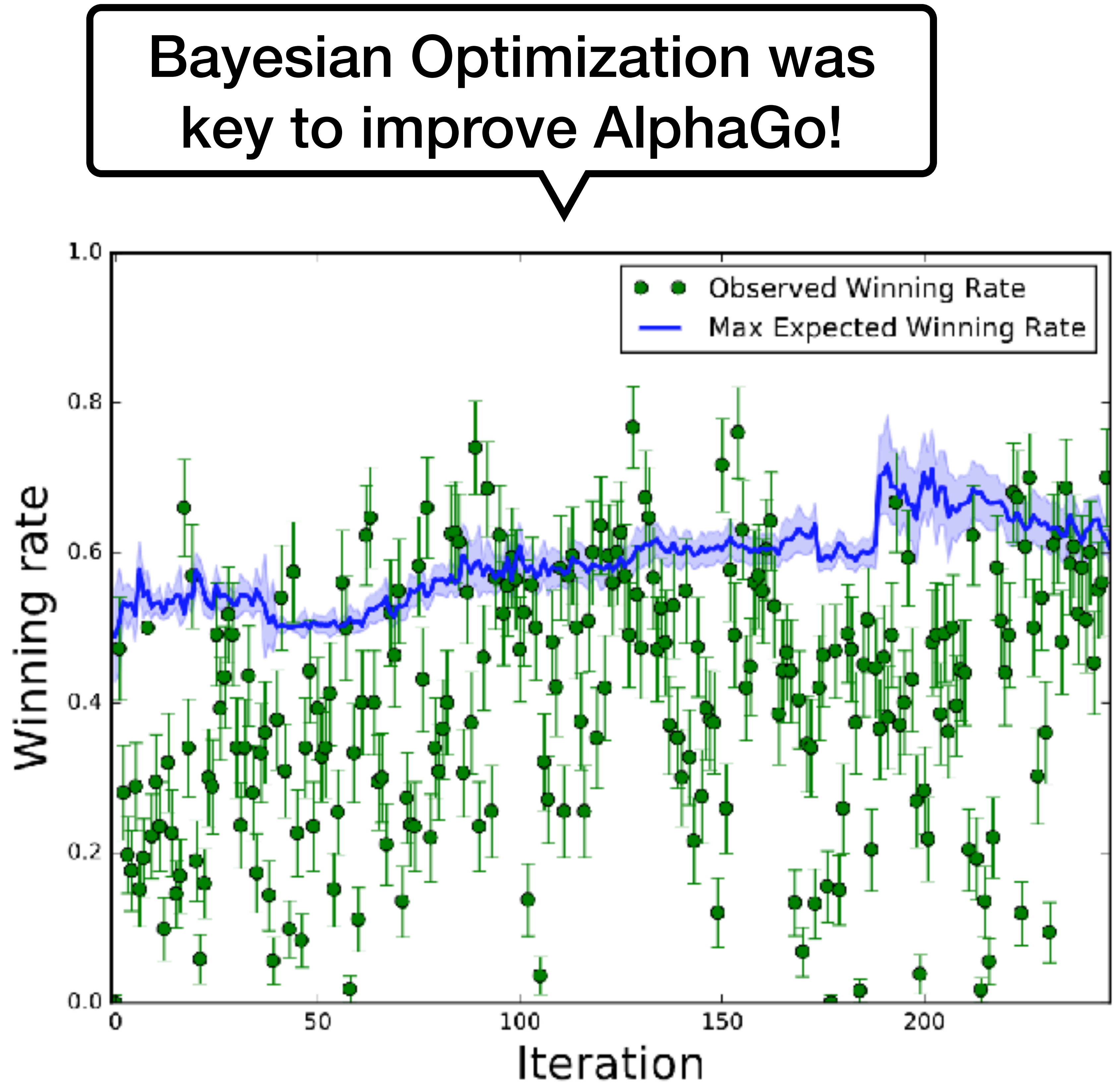
Bayesian Optimization was key to improve AlphaGo!




Bayesian Optimization in AlphaGo [Chen 2018]

# Introduction

- Hyperoptimization often gives large improvements
- But it is not systematically used
- Why?
  - Lack of tool knowledge
  - Attachement to grid-search interpretability & ease
  - Can be expensive
- **Transfer learning** is a subfield that **speeds up** HPO by looking at previous evaluations and tuning runs




Bayesian Optimization in AlphaGo [Chen 2018]

# Speeding-up HPO

# Speeding-up HPO

- *Bayesian Optimization*:

# Speeding-up HPO

- *Bayesian Optimization*:
  - allows to exploit previous evaluations by building a surrogate model

# Speeding-up HPO

- *Bayesian Optimization*:
  - allows to exploit previous evaluations by building a surrogate model
- *Multifidelity*:

# Speeding-up HPO

- *Bayesian Optimization*:
  - allows to exploit previous evaluations by building a surrogate model
- *Multifidelity*:
  - Evaluating configurations until the end is wasteful, we can often early stop

# Speeding-up HPO

- *Bayesian Optimization*:
  - allows to exploit previous evaluations by building a surrogate model
- *Multifidelity*:
  - Evaluating configurations until the end is wasteful, we can often early stop
- *Asynchronous distributed optimization*:

# Speeding-up HPO

- *Bayesian Optimization*:
  - allows to exploit previous evaluations by building a surrogate model
- *Multifidelity*:
  - Evaluating configurations until the end is wasteful, we can often early stop
- *Asynchronous distributed optimization*:
  - Use multiple asynchronous workers

# Speeding-up HPO

- *Bayesian Optimization*:
  - allows to exploit previous evaluations by building a surrogate model
- *Multifidelity*:
  - Evaluating configurations until the end is wasteful, we can often early stop
- *Asynchronous distributed optimization*:
  - Use multiple asynchronous workers
  - Don’t wait for intermediate results: schedule new tasks as soon as workers are available

# Speeding-up HPO

- *Bayesian Optimization*:
  - allows to exploit previous evaluations by building a surrogate model
- *Multifidelity*:
  - Evaluating configurations until the end is wasteful, we can often early stop
- *Asynchronous distributed optimization*:
  - Use multiple asynchronous workers
  - Don’t wait for intermediate results: schedule new tasks as soon as workers are available
- *Transfer-learning*:

# Speeding-up HPO

- *Bayesian Optimization*:
  - allows to exploit previous evaluations by building a surrogate model
- *Multifidelity*:
  - Evaluating configurations until the end is wasteful, we can often early stop
- *Asynchronous distributed optimization*:
  - Use multiple asynchronous workers
  - Don’t wait for intermediate results: schedule new tasks as soon as workers are available
- *Transfer-learning*:
  - exploit information from previous HPO runs

# Hyperparameter optimization

## Recap

# Hyperparameter optimization
## Recap

- Hyperparameter simple setting, find the best hyperparameter of $f(x) \in \mathbb{R}$

# Hyperparameter optimization

## Recap

- Hyperparameter simple setting, find the best hyperparameter of $f(x) \in \mathbb{R}$
  - $x^* = \text{argmin}_{x \in \mathcal{X}} f(x)$

# Hyperparameter optimization

## Recap

- Hyperparameter simple setting, find the best hyperparameter of $f(x) \in \mathbb{R}$
  - $x^* = \text{argmin}_{x \in \mathcal{X}} f(x)$
- $f(x)$ can be the accuracy obtained after training a neural network with hyperparameter $x$

# Hyperparameter optimization

## Recap

- Hyperparameter simple setting, find the best hyperparameter of $f(x) \in \mathbb{R}$
  - $x^* = \text{argmin}_{x \in \mathcal{X}} f(x)$
- $f(x)$ can be the accuracy obtained after training a neural network with hyperparameter $x$

An example of a search space $\mathcal{X}$

| **Hyperparameter** | **Range** | **Scale** |
|---|---|---|
| Architecture | {ConvNext, ViT, EfficientNet} | Discrete |
| Dropout | $[0.0, 1.0]$ | Uniform |
| Optimizer | {SGD, Adam, RMSProp} | Discrete |
| Learning Rate | $[10^{-5}, 10^{0}]$ | Log |

Wistuba and Grabocka. Meta-Learning for Hyperparameter Optimization 2023

# Hyperparameter optimization

## Recap

- Hyperparameter simple setting, find the best hyperparameter of $f(x) \in \mathbb{R}$
  - $x^* = \text{argmin}_{x \in \mathcal{X}} f(x)$
- $f(x)$ can be the accuracy obtained after training a neural network with hyperparameter $x$

An example of a search space $\mathcal{X}$

| Hyperparameter | Range | Scale |
|---|---|---|
| Architecture | {ConvNext, ViT, EfficientNet} | Discrete |
| Dropout | $[0.0, 1.0]$ | Uniform |
| Optimizer | {SGD, Adam, RMSProp} | Discrete |
| Learning Rate | $[10^{-5}, 10^{0}]$ | Log |

Wistuba and Grabocka. Meta-Learning for Hyperparameter Optimization 2023

For instance, $x^* = [\text{ViT}, 0.2, \text{Adam}, 10^{-4}]$

# Hyperparameter optimization

## Recap

- Hyperparameter simple setting, find the best hyperparameter of $f(x) \in \mathbb{R}$
  - $x^* = \text{argmin}_{x \in \mathcal{X}} f(x)$
- $f(x)$ can be the accuracy obtained after training a neural network with hyperparameter $x$
- What if we had extra evaluations?

An example of a search space $\mathcal{X}$

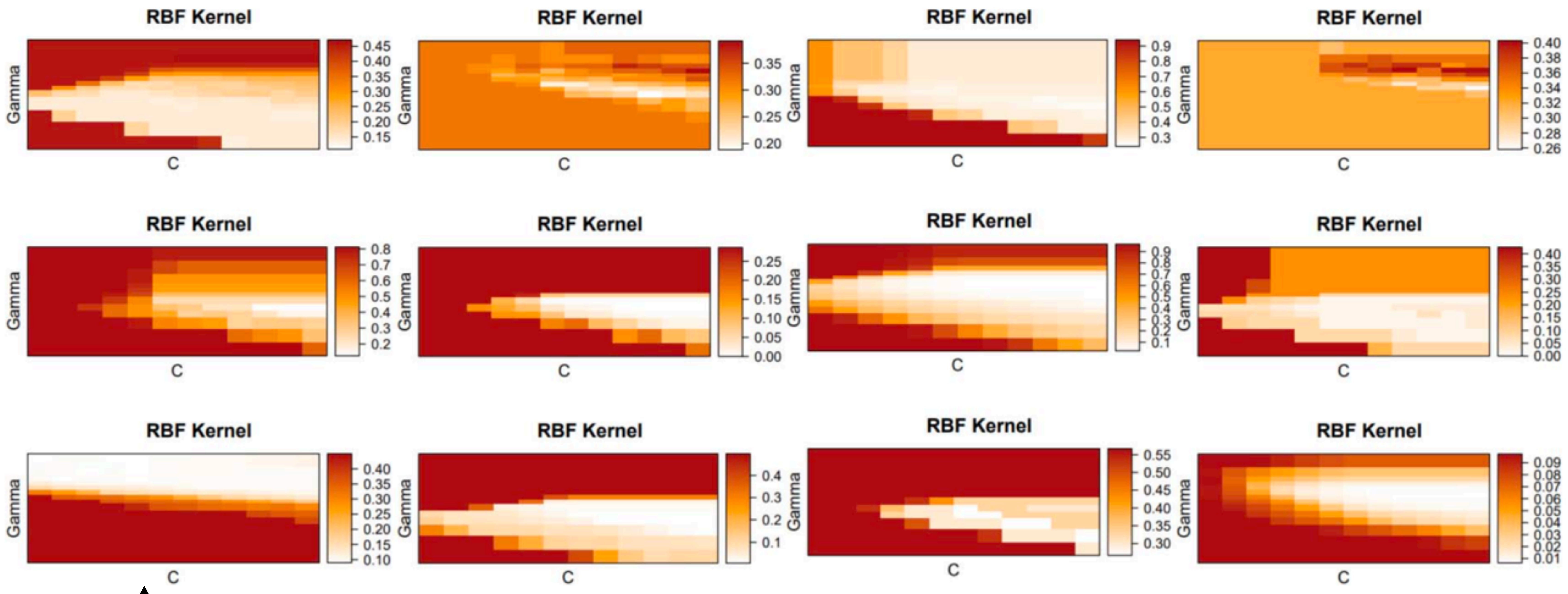
| **Hyperparameter** | **Range** | **Scale** |
|---|---|---|
| Architecture | {ConvNext, ViT, EfficientNet} | Discrete |
| Dropout | $[0.0, 1.0]$ | Uniform |
| Optimizer | {SGD, Adam, RMSProp} | Discrete |
| Learning Rate | $[10^{-5}, 10^{0}]$ | Log |

Wistuba and Grabocka. Meta-Learning for Hyperparameter Optimization 2023

For instance, $x^* = [\text{ViT}, 0.2, \text{Adam}, 10^{-4}]$

# Leveraging extra evaluations

## Why we expect it to work




Response function of different datasets can look very similar.

Can we exploit the similarity between datasets?

Wistuba and Grabocka. Meta-Learning for Hyperparameter Optimization 2023

# Hyperparameter optimization

## With off-line evaluations…

| task | model | learning-rate | #layers | error |
|---|---|---|---|---|
| electricity | LSTM | 0.001 | 10 | 0.1 |
| electricity | Transformer | 0.001 | 10 | 0.08 |
| electricity | Transformer | 1.0 | 2 | 0.9 |
| ... | | | | |
| traffic | LSTM | 0.1 | 2 | 0.9 |
| traffic | Transformer | 0.004 | 2.5 | 0.03 |

# Hyperparameter optimization

## With off-line evaluations…

Assume one has run many previous hyperparameter optimizations

| task | model | learning-rate | #layers | error |
|---|---|---|---|---|
| electricity | LSTM | 0.001 | 10 | 0.1 |
| electricity | Transformer | 0.001 | 10 | 0.08 |
| electricity | Transformer | 1.0 | 2 | 0.9 |
| ... | | | | |
| traffic | LSTM | 0.1 | 2 | 0.9 |
| traffic | Transformer | 0.004 | 2.5 | 0.03 |

# Hyperparameter optimization

## With off-line evaluations…

Assume one has run many previous hyperparameter optimizations

| task | model | learning-rate | #layers | error |
|---|---|---|---|---|
| electricity | LSTM | 0.001 | 10 | 0.1 |
| electricity | Transformer | 0.001 | 10 | 0.08 |
| electricity | Transformer | 1.0 | 2 | 0.9 |
| ... | | | | |
| traffic | LSTM | 0.1 | 2 | 0.9 |
| traffic | Transformer | 0.004 | 2.5 | 0.03 |

Error obtained when training a forecasting model on a given dataset

# Hyperparameter optimization

## With off-line evaluations…

Assume one has run many previous hyperparameter optimizations

| task | model | learning-rate | #layers | error |
|---|---|---|---|---|
| electricity | LSTM | 0.001 | 10 | 0.1 |
| electricity | Transformer | 0.001 | 10 | 0.08 |
| electricity | Transformer | 1.0 | 2 | 0.9 |
| ... | | | | |
| traffic | LSTM | 0.1 | 2 | 0.9 |
| traffic | Transformer | 0.004 | 2.5 | 0.03 |

Error obtained when training a forecasting model on a given dataset

| task | model | learning-rate | #layers | error |
|---|---|---|---|---|
| solar | DeepAR | 0.001 | 10 | ? |
| solar | Transformer | 0.001 | 10 | ? |
| solar | Transformer | 1.0 | 2 | ? |
| ... | | | | |
| solar | DeepAR | 0.1 | 2 | ? |
| solar | Transformer | 0.004 | 2.5 | ? |

# Hyperparameter optimization

## With off-line evaluations…

Assume one has run many previous hyperparameter optimizations

| task | model | learning-rate | #layers | error |
|---|---|---|---|---|
| electricity | LSTM | 0.001 | 10 | 0.1 |
| electricity | Transformer | 0.001 | 10 | 0.08 |
| electricity | Transformer | 1.0 | 2 | 0.9 |
| ... | | | | |
| traffic | LSTM | 0.1 | 2 | 0.9 |
| traffic | Transformer | 0.004 | 2.5 | 0.03 |

Error obtained when training a forecasting model on a given dataset

| task | model | learning-rate | #layers | error |
|---|---|---|---|---|
| solar | DeepAR | 0.001 | 10 | ? |
| solar | Transformer | 0.001 | 10 | ? |
| solar | Transformer | 1.0 | 2 | ? |
| ... | | | | |
| solar | DeepAR | 0.1 | 2 | ? |
| solar | Transformer | 0.004 | 2.5 | ? |

🤔 How to exploit past-observations to speed-up the search of a new task?

# Leveraging extra evaluations

## Notations

# Leveraging extra evaluations

## Notations

- Want to optimize $x^* = \text{argmin}_{x \in \mathcal{X}} f(x)$

# Leveraging extra evaluations

## Notations

- Want to optimize $x^* = \text{argmin}_{x \in \mathcal{X}} f(x)$
- Offline evaluations $\mathcal{D}^M = \bigcup_{i=1}^{M} \{x_i^j, y_i^j\}_{i=1}^{N_j}$ available for $M$ tasks where $y_i^j = f^j(x_i^j)$

# Leveraging extra evaluations

## Notations

$$\mathscr{D}^M = \bigcup_{i=1}^{M} \{x_i^j, y_i^j\}_{i=1}^{N_j}$$

| task | model | learning-rate | #layers | error |
|---|---|---|---|---|
| electricity | LSTM | 0.001 | 10 | 0.1 |
| electricity | Transformer | 0.001 | 10 | 0.08 |
| electricity | Transformer | 1.0 | 2 | 0.9 |
| ... | | | | |
| traffic | LSTM | 0.1 | 2 | 0.9 |
| traffic | Transformer | 0.004 | 2.5 | 0.03 |

- Want to optimize $x^* = \text{argmin}_{x \in \mathscr{X}} f(x)$
- Offline evaluations $\mathscr{D}^M = \bigcup_{i=1}^{M} \{x_i^j, y_i^j\}_{i=1}^{N_j}$ available for $M$ tasks where $y_i^j = f^j(x_i^j)$

# Leveraging extra evaluations

## Notations

- Want to optimize $x^* = \text{argmin}_{x \in \mathcal{X}} f(x)$
- Offline evaluations $\mathcal{D}^M = \bigcup_{i=1}^{M} \{x_i^j, y_i^j\}_{i=1}^{N_j}$ available for $M$ tasks where $y_i^j = f^j(x_i^j)$
- For instance $f^j$ are results of the same ML model as $f$ obtained when training on a different dataset $j$

$$\mathcal{D}^M = \bigcup_{i=1}^{M} \{x_i^j, y_i^j\}_{i=1}^{N_j}$$

| task | model | learning-rate | #layers | error |
|---|---|---|---|---|
| electricity | LSTM | 0.001 | 10 | 0.1 |
| electricity | Transformer | 0.001 | 10 | 0.08 |
| electricity | Transformer | 1.0 | 2 | 0.9 |
| ... | | | | |
| traffic | LSTM | 0.1 | 2 | 0.9 |
| traffic | Transformer | 0.004 | 2.5 | 0.03 |

# Leveraging extra evaluations

## Notations

$$\mathcal{D}^M = \bigcup_{i=1}^{M} \{x_i^j, y_i^j\}_{i=1}^{N_j}$$

| task | model | learning-rate | #layers | error |
|---|---|---|---|---|
| electricity | LSTM | 0.001 | 10 | 0.1 |
| electricity | Transformer | 0.001 | 10 | 0.08 |
| electricity | Transformer | 1.0 | 2 | 0.9 |
| ... | | | | |
| traffic | LSTM | 0.1 | 2 | 0.9 |
| traffic | Transformer | 0.004 | 2.5 | 0.03 |

- Want to optimize $x^* = \text{argmin}_{x \in \mathcal{X}} f(x)$
- Offline evaluations $\mathcal{D}^M = \bigcup_{i=1}^{M} \{x_i^j, y_i^j\}_{i=1}^{N_j}$ available for $M$ tasks where $y_i^j = f^j(x_i^j)$
- For instance $f^j$ are results of the same ML model as $f$ obtained when training on a different dataset $j$
- Can we use $\mathcal{D}^M$ to find good hyperparameter on our new task $f$ **much faster**?

# Leveraging extra evaluations

## Notations

$$\mathcal{D}^M = \bigcup_{i=1}^{M} \{x_i^j, y_i^j\}_{i=1}^{N_j}$$

- Want to optimize $x^* = \text{argmin}_{x \in \mathcal{X}} f(x)$
- Offline evaluations $\mathcal{D}^M = \bigcup_{i=1}^{M} \{x_i^j, y_i^j\}_{i=1}^{N_j}$ available for $M$ tasks where $y_i^j = f^j(x_i^j)$
- For instance $f^j$ are results of the same ML model as $f$ obtained when training on a different dataset $j$
- Can we use $\mathcal{D}^M$ to find good hyperparameter on our new task $f$ **much faster**?

| task | model | learning-rate | #layers | error |
|---|---|---|---|---|
| electricity | LSTM | 0.001 | 10 | 0.1 |
| electricity | Transformer | 0.001 | 10 | 0.08 |
| electricity | Transformer | 1.0 | 2 | 0.9 |
| ... | | | | |
| traffic | LSTM | 0.1 | 2 | 0.9 |
| traffic | Transformer | 0.004 | 2.5 | 0.03 |

| task | model | learning-rate | #layers | error |
|---|---|---|---|---|
| solar | DeepAR | 0.001 | 10 | ? |
| solar | Transformer | 0.001 | 10 | ? |
| solar | Transformer | 1.0 | 2 | ? |
| ... | | | | |
| solar | DeepAR | 0.1 | 2 | ? |
| solar | Transformer | 0.004 | 2.5 | ? |

# Leveraging extra evaluations

## Notations

$$\mathcal{D}^M = \bigcup_{i=1}^{M} \{x_i^j, y_i^j\}_{i=1}^{N_j}$$

- Want to optimize $x^* = \text{argmin}_{x \in \mathcal{X}} f(x)$
- Offline evaluations $\mathcal{D}^M = \bigcup_{i=1}^{M} \{x_i^j, y_i^j\}_{i=1}^{N_j}$ available for $M$ tasks where $y_i^j = f^j(x_i^j)$
- For instance $f^j$ are results of the same ML model as $f$ obtained when training on a different dataset $j$
- Can we use $\mathcal{D}^M$ to find good hyperparameter on our new task $f$ **much faster**?

| task | model | learning-rate | #layers | error |
|---|---|---|---|---|
| electricity | LSTM | 0.001 | 10 | 0.1 |
| electricity | Transformer | 0.001 | 10 | 0.08 |
| electricity | Transformer | 1.0 | 2 | 0.9 |
| ... | | | | |
| traffic | LSTM | 0.1 | 2 | 0.9 |
| traffic | Transformer | 0.004 | 2.5 | 0.03 |

| task | model | learning-rate | #layers | error |
|---|---|---|---|---|
| solar | DeepAR | 0.001 | 10 | ? |
| solar | Transformer | 0.001 | 10 | ? |
| solar | Transformer | 1.0 | 2 | ? |
| ... | | | | |
| solar | DeepAR | 0.1 | 2 | ? |
| solar | Transformer | 0.004 | 2.5 | ? |

🤔 Can you think about potential strategies?

# Methods

# Transfer learning methods

# Transfer learning methods

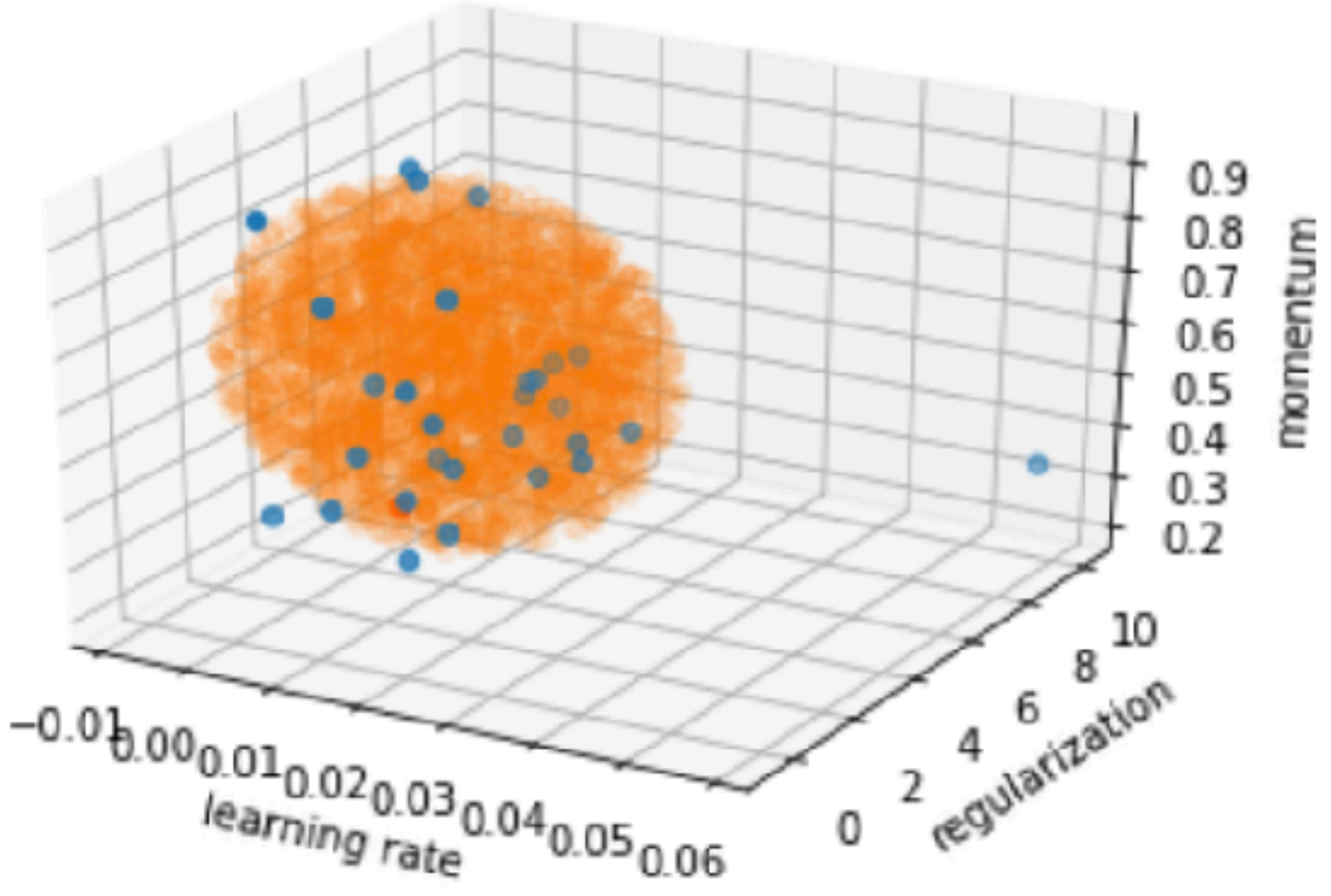
- Warm-starting: initialise Bayesian Optimization with best previous configuration [Feurer 2015]

# Transfer learning methods

- Warm-starting: initialise Bayesian Optimization with best previous configuration [Feurer 2015]
- Prune search space: restrict search to bounding-box of best solution found previously [Peronne 2019]

# Transfer learning methods

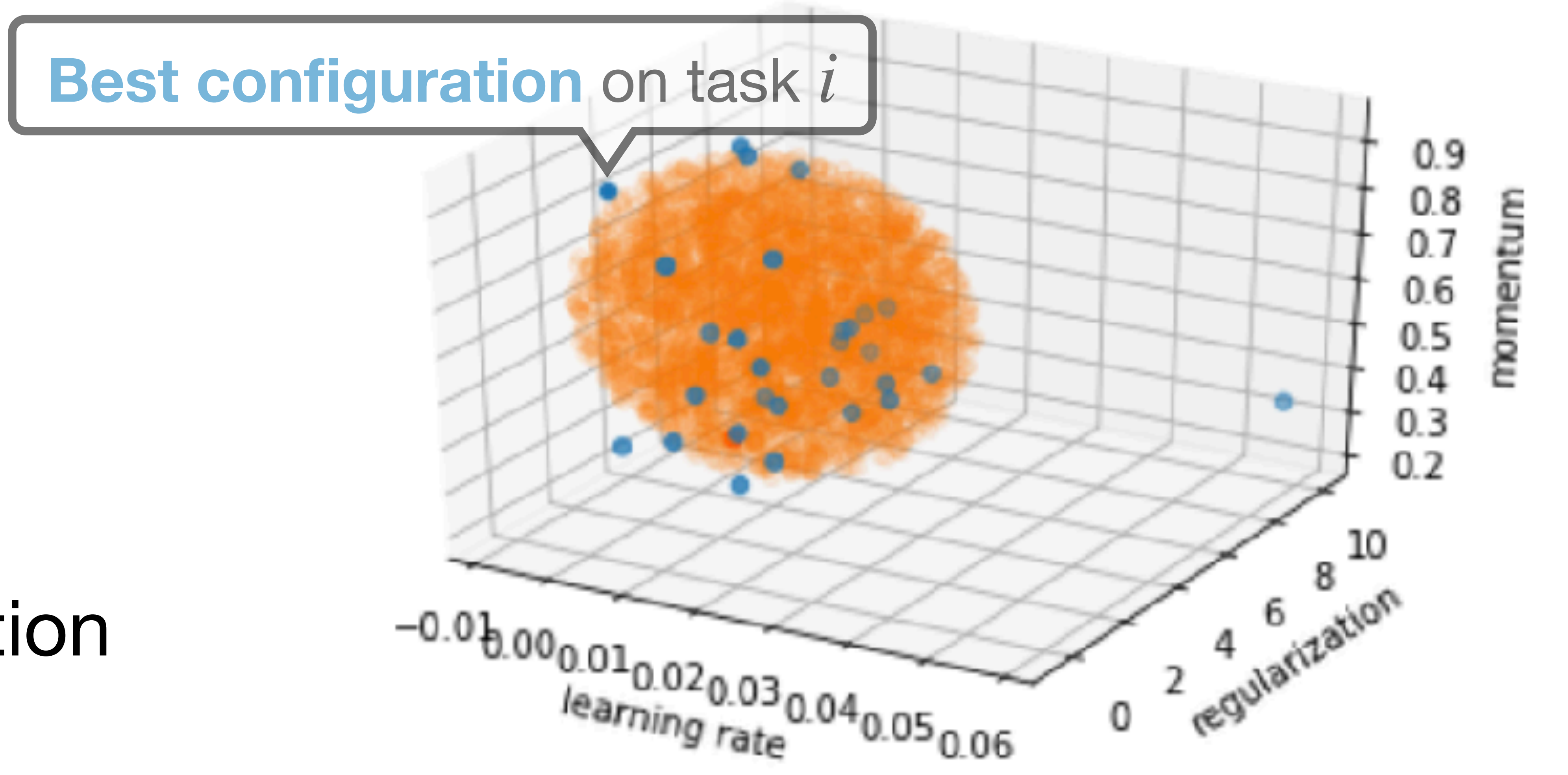
- Warm-starting: initialise Bayesian Optimization with best previous configuration [Feurer 2015]
- Prune search space: restrict search to bounding-box of best solution found previously [Peronne 2019]




Prune search space to an ellipse containing the best hyperparameters of previous tasks [Peronne 2019]

# Transfer learning methods

- Warm-starting: initialise Bayesian Optimization with best previous configuration [Feurer 2015]
- Prune search space: restrict search to bounding-box of best solution found previously [Peronne 2019]

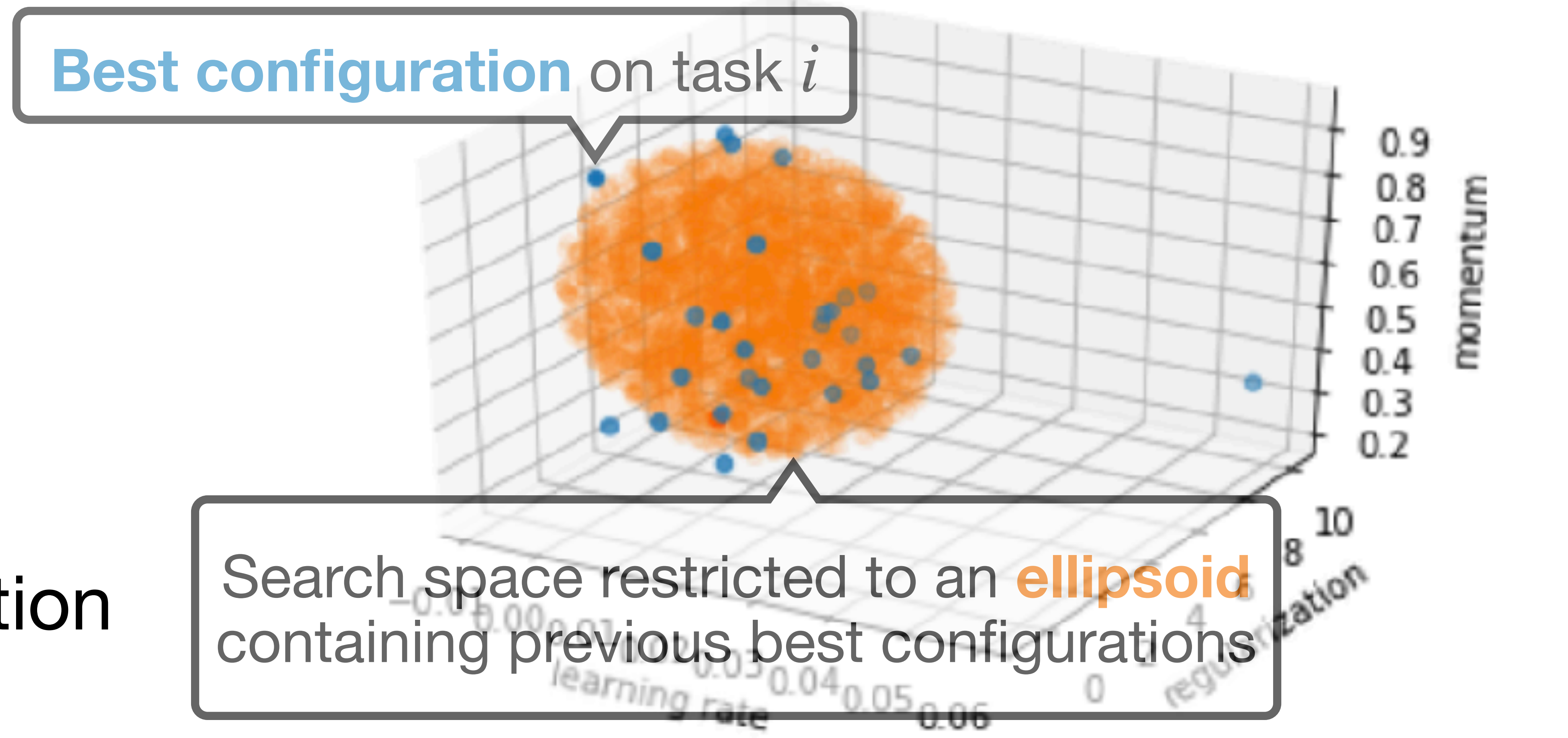



Prune search space to an ellipse containing the best hyperparameters of previous tasks [Peronne 2019]

# Transfer learning methods

- Warm-starting: initialise Bayesian Optimization with best previous configuration [Feurer 2015]

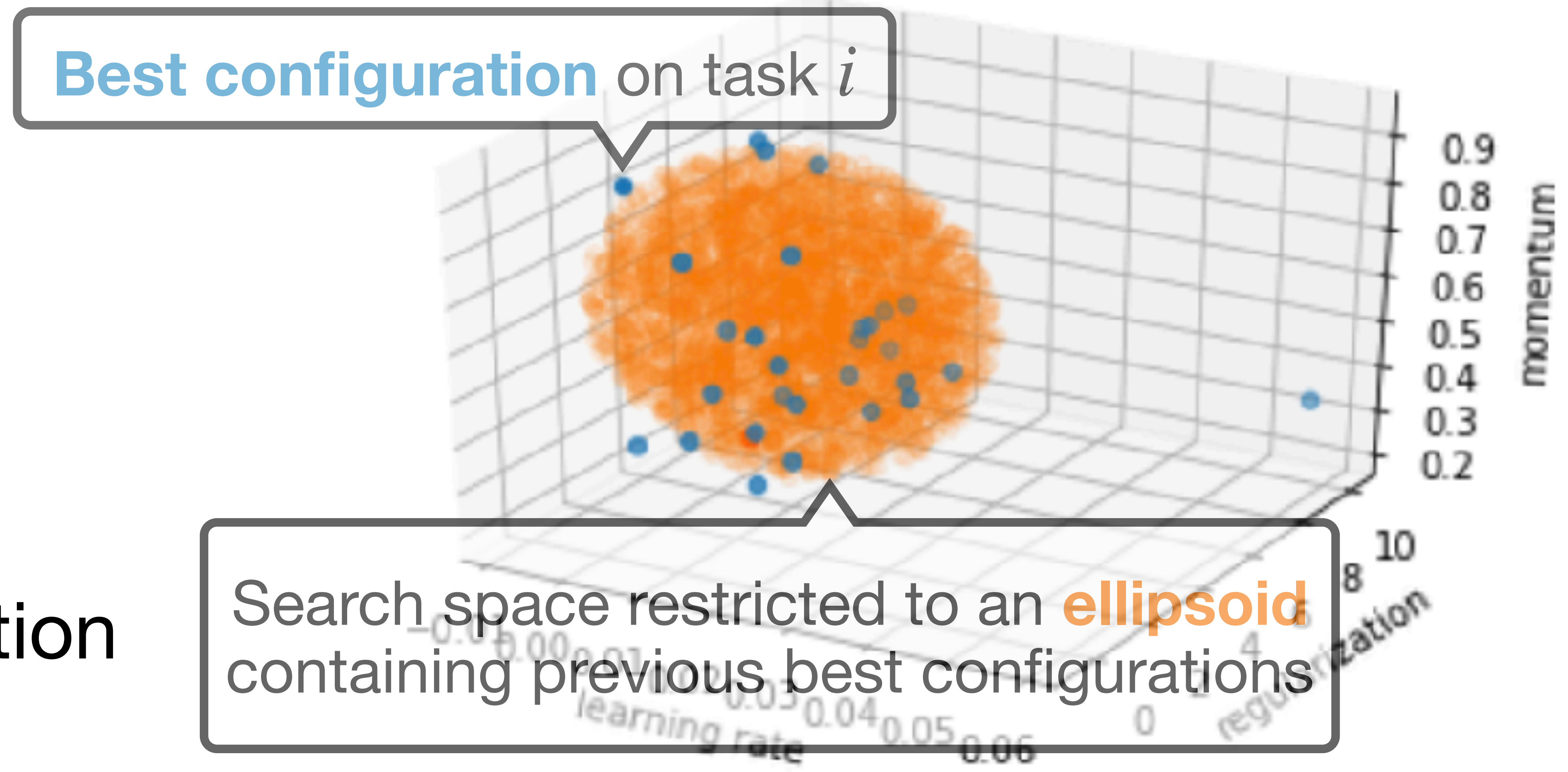
- Prune search space: restrict search to bounding-box of best solution found previously [Peronne 2019]




Prune search space to an ellipse containing the best hyperparameters of previous tasks [Peronne 2019]

# Transfer learning methods

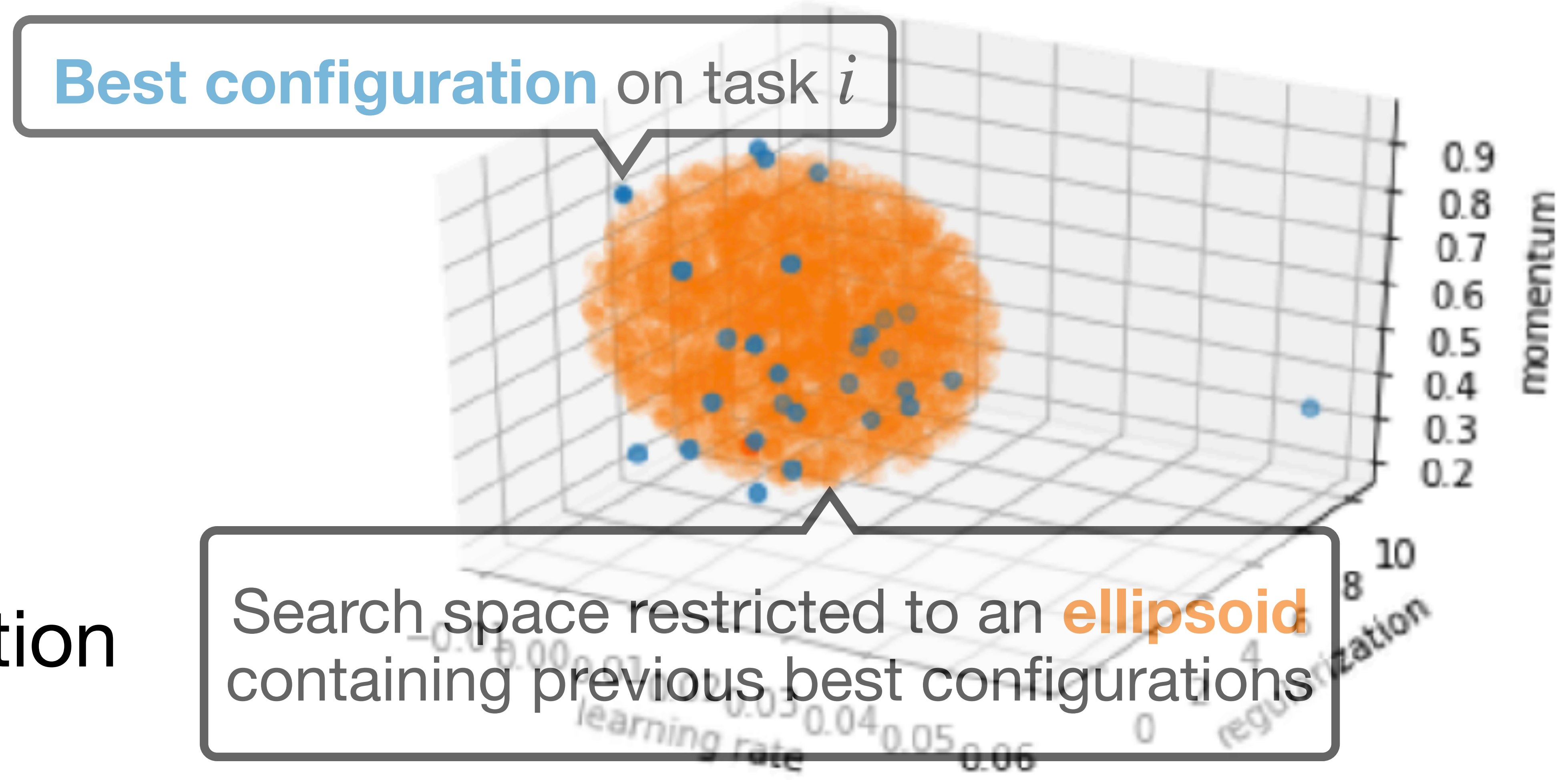
- Warm-starting: initialise Bayesian Optimization with best previous configuration [Feurer 2015]
- Prune search space: restrict search to bounding-box of best solution found previously [Peronne 2019]
- Leverage feature describing task to find most similar tasks [Wistuba 2015, Jomaa 2021]




Prune search space to an ellipse containing the best hyperparameters of previous tasks [Peronne 2019]

# Transfer learning methods

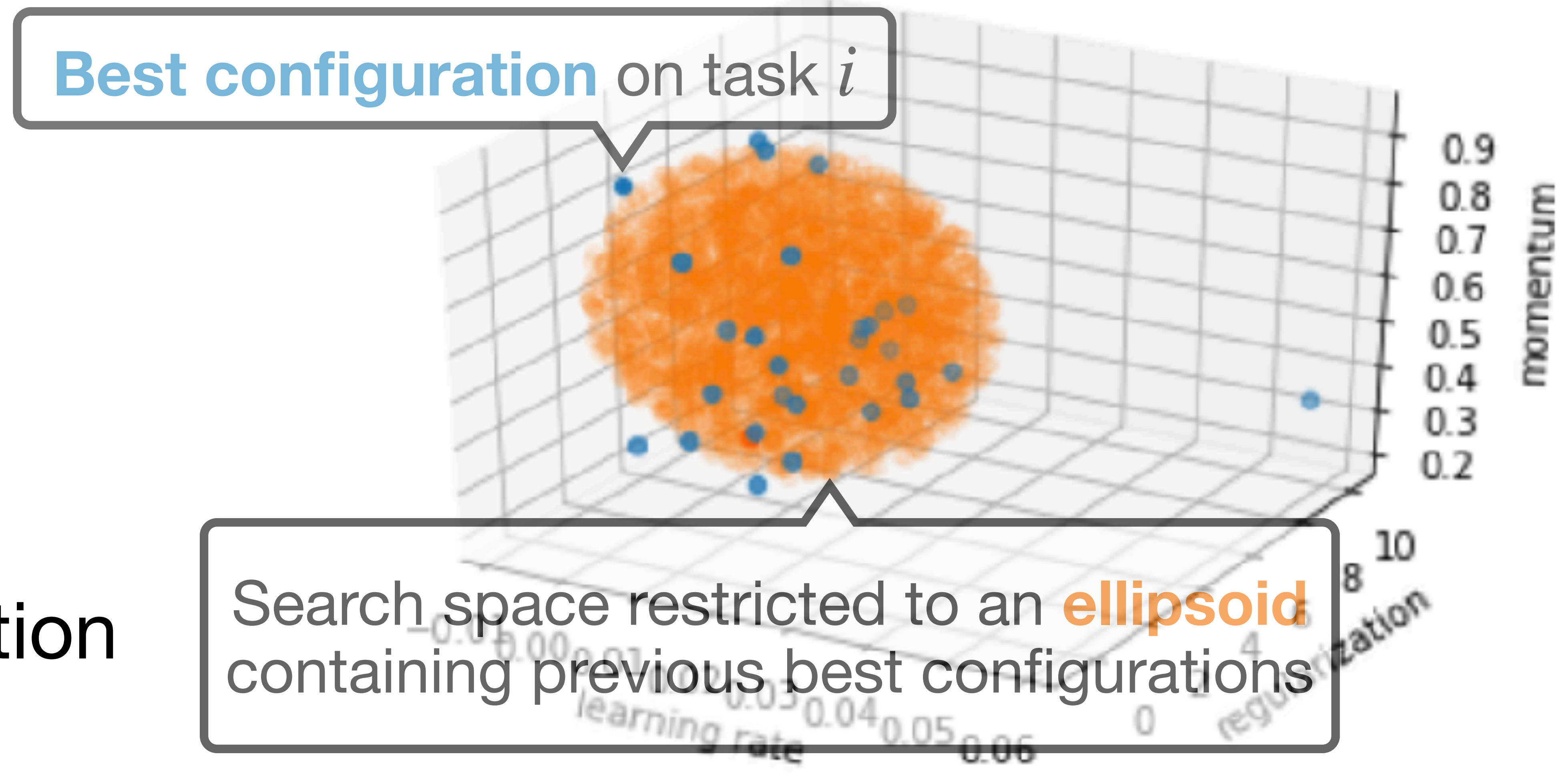
- Warm-starting: initialise Bayesian Optimization with best previous configuration [Feurer 2015]
- Prune search space: restrict search to bounding-box of best solution found previously [Peronne 2019]
- Leverage feature describing task to find most similar tasks [Wistuba 2015, Jomaa 2021]
- Surrogates: ABLR [Peronne 2018] / Deep kernel [Wistuba 2021]




Prune search space to an ellipse containing the best hyperparameters of previous tasks [Peronne 2019]

# Transfer learning methods

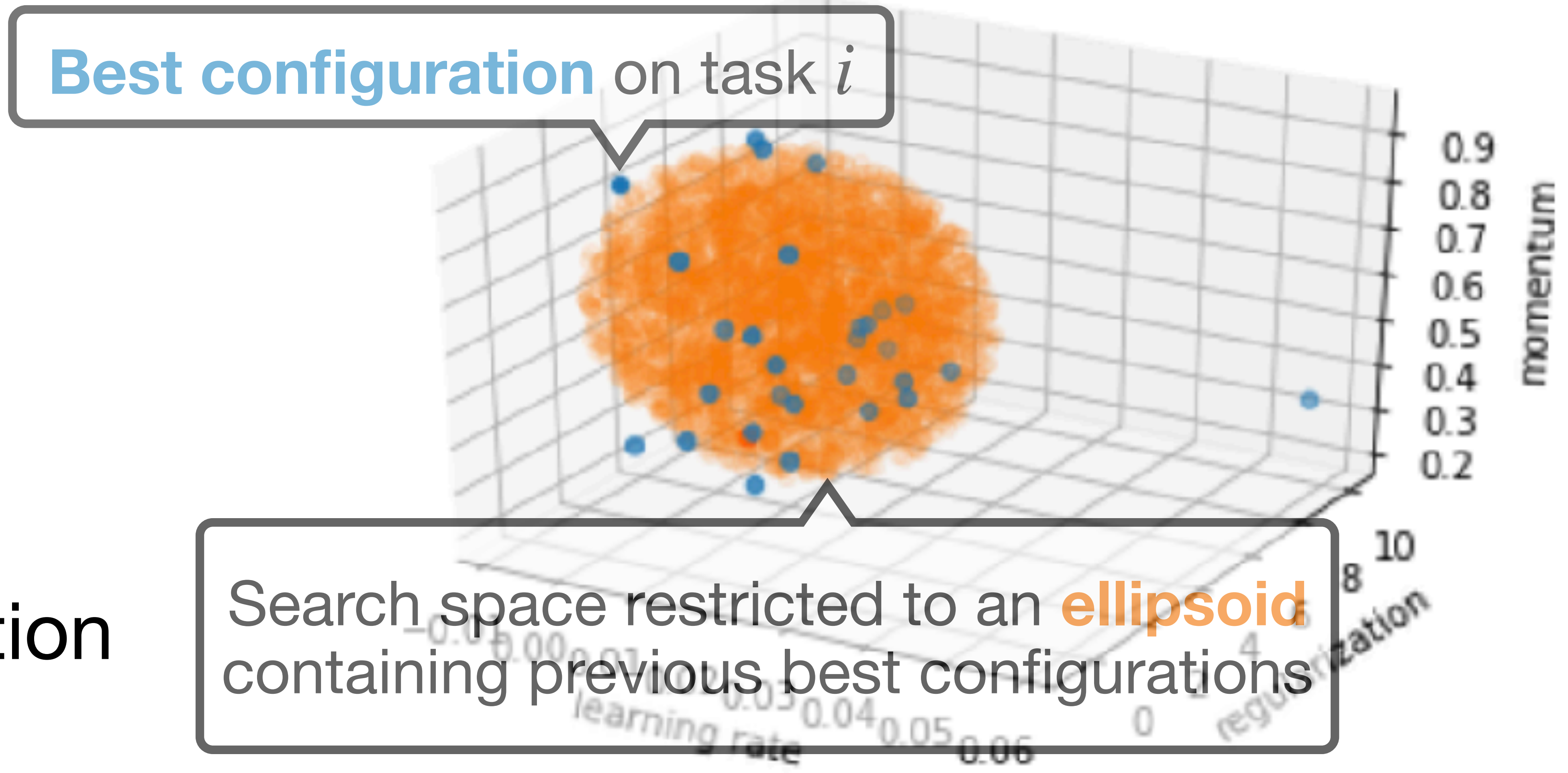
- Warm-starting: initialise Bayesian Optimization with best previous configuration [Feurer 2015]
- Prune search space: restrict search to bounding-box of best solution found previously [Peronne 2019]
- Leverage feature describing task to find most similar tasks [Wistuba 2015, Jomaa 2021]
- Surrogates: ABLR [Peronne 2018] / Deep kernel [Wistuba 2021]
- Learning curve learning [Wistuba 2020]



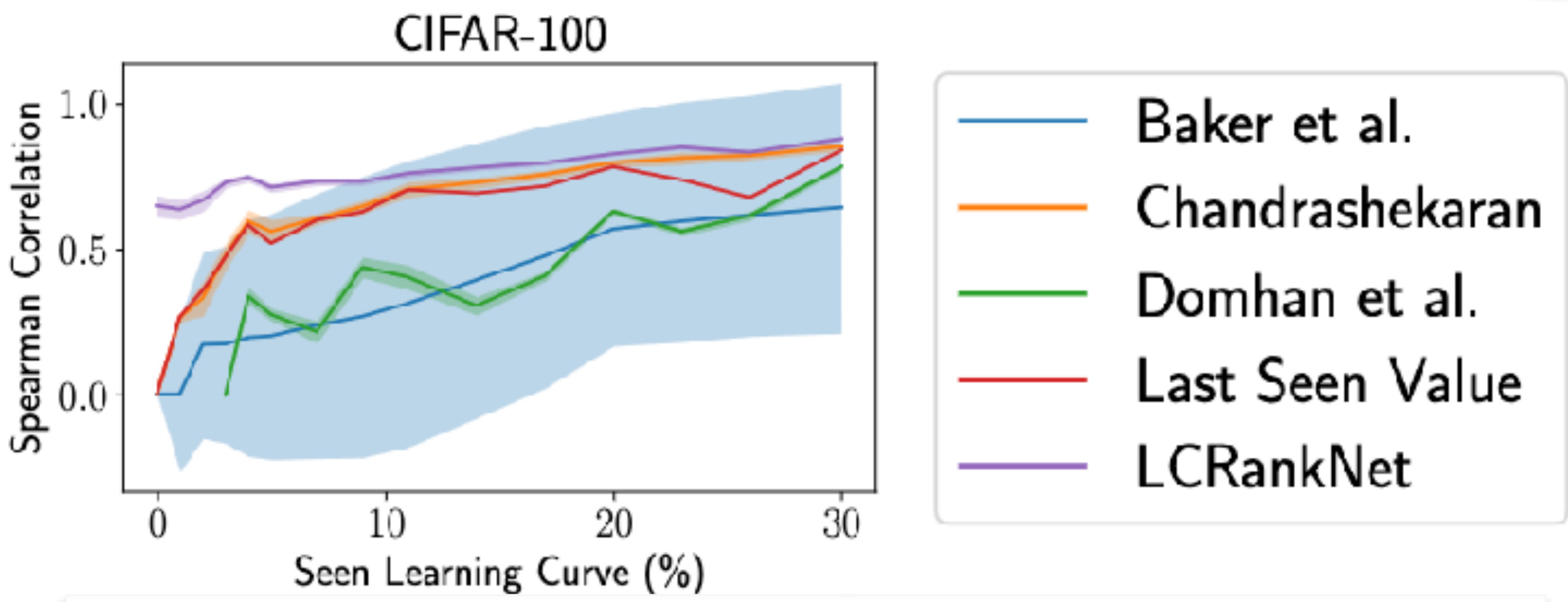

Prune search space to an ellipse containing the best hyperparameters of previous tasks [Peronne 2019]

# Transfer learning methods

- Warm-starting: initialise Bayesian Optimization with best previous configuration [Feurer 2015]
- Prune search space: restrict search to bounding-box of best solution found previously [Peronne 2019]
- Leverage feature describing task to find most similar tasks [Wistuba 2015, Jomaa 2021]
- Surrogates: ABLR [Peronne 2018] / Deep kernel [Wistuba 2021]
- Learning curve learning [Wistuba 2020]



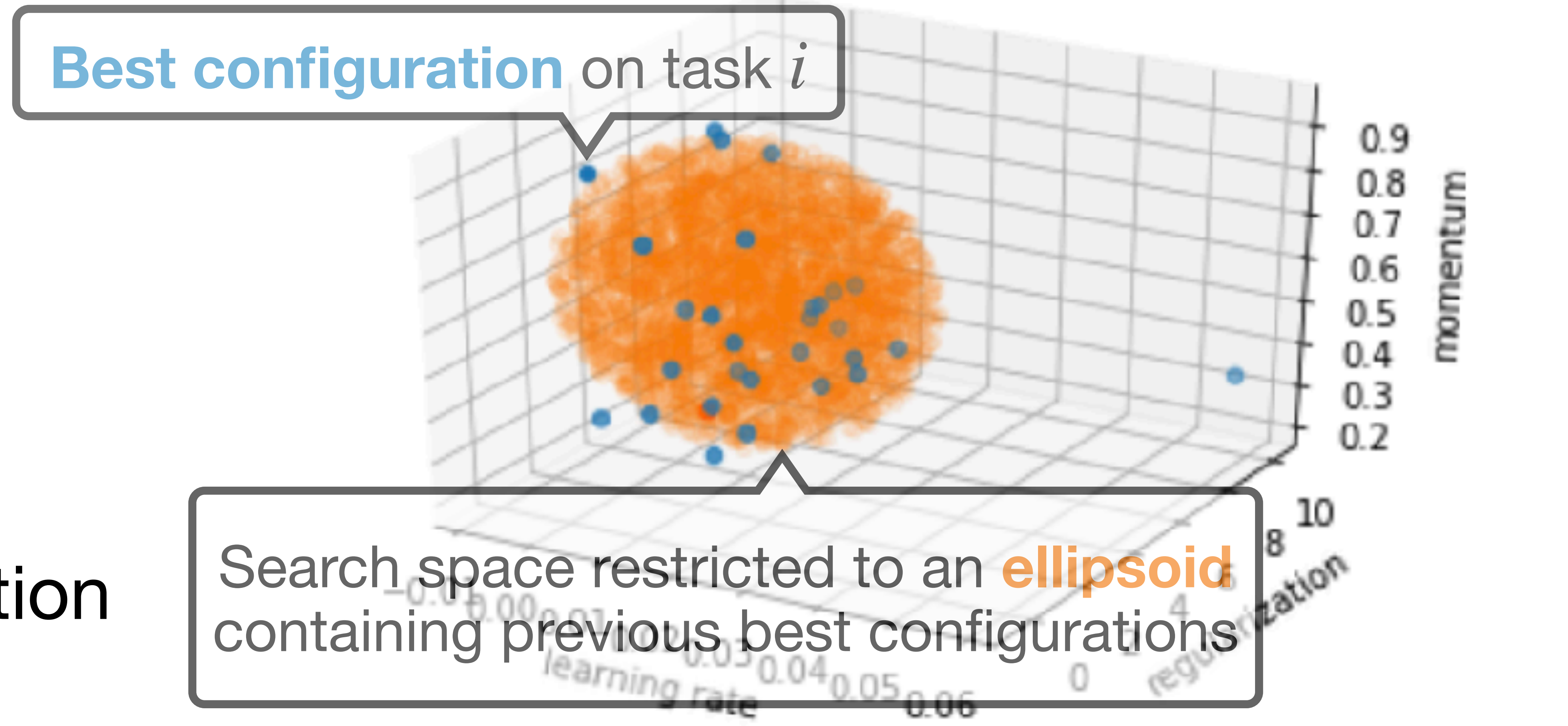

Prune search space to an ellipse containing the best hyperparameters of previous tasks [Peronne 2019]



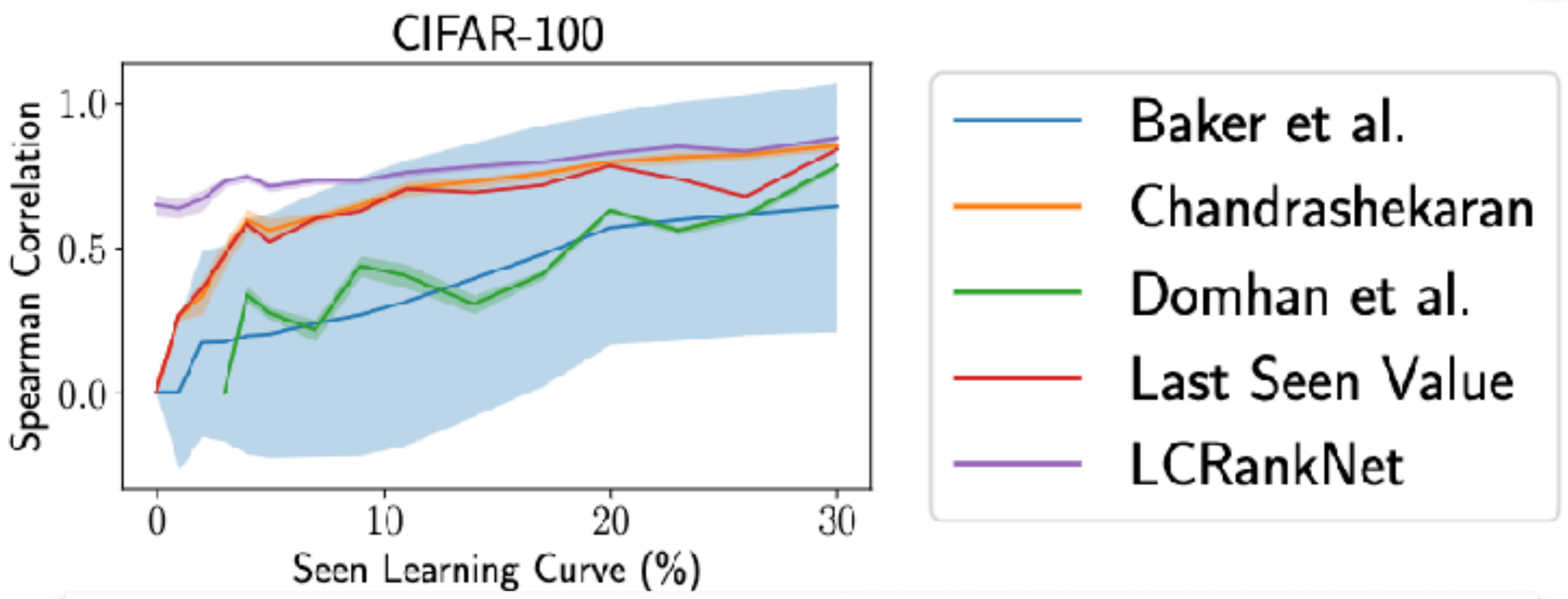

Learning curve model learned on offline evaluations allows to predict final values with high correlations [Wistuba 2020]

# Transfer learning methods

- Warm-starting: initialise Bayesian Optimization with best previous configuration [Feurer 2015]
- Prune search space: restrict search to bounding-box of best solution found previously [Peronne 2019]
- Leverage feature describing task to find most similar tasks [Wistuba 2015, Jomaa 2021]
- Surrogates: ABLR [Peronne 2018] / Deep kernel [Wistuba 2021]
- Learning curve learning [Wistuba 2020]
- Prior-based:



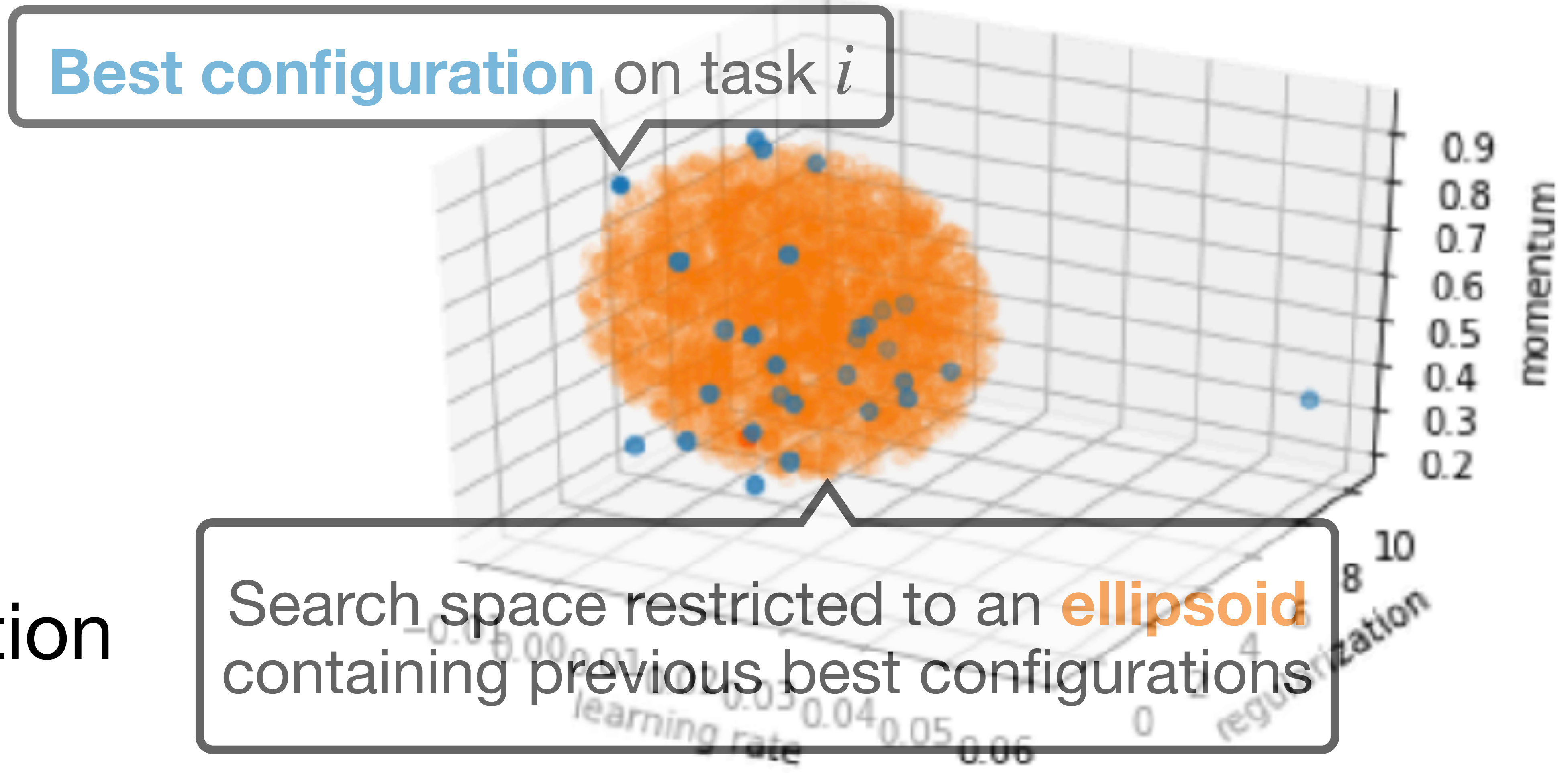

Prune search space to an ellipse containing the best hyperparameters of previous tasks [Peronne 2019]



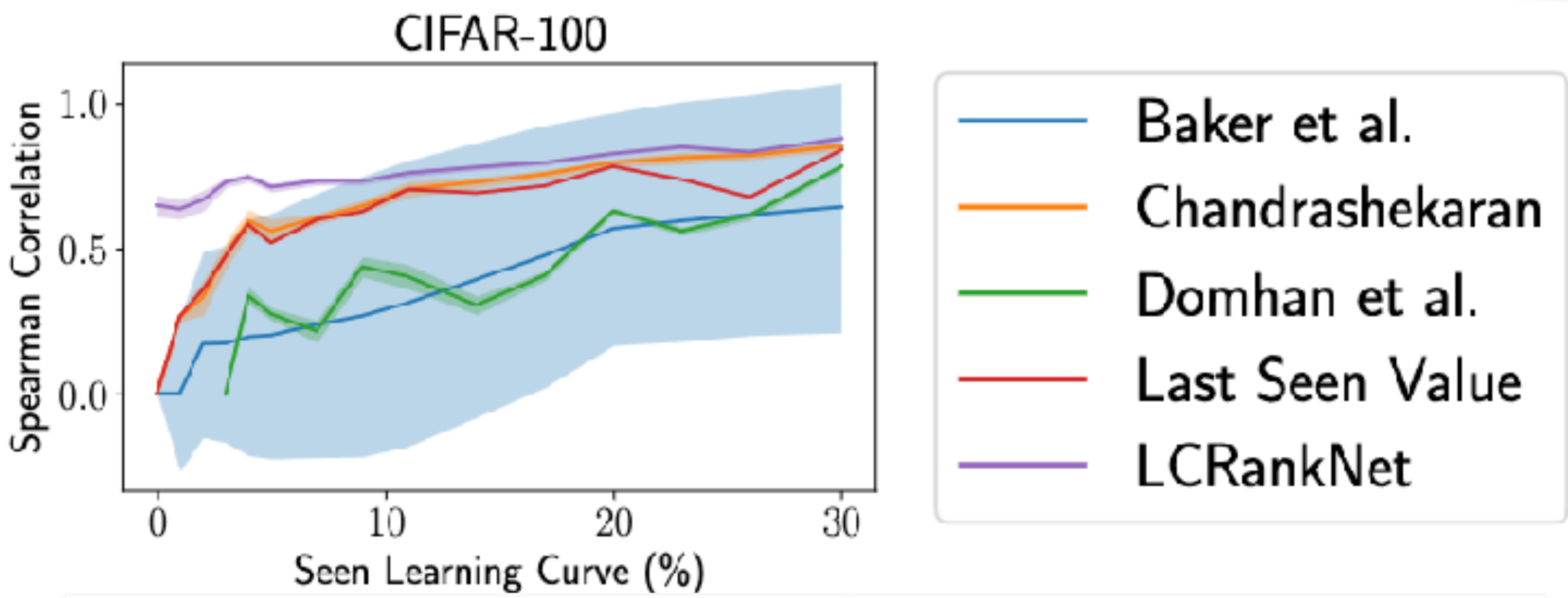

Learning curve model learned on offline evaluations allows to predict final values with high correlations [Wistuba 2020]

# Transfer learning methods

- Warm-starting: initialise Bayesian Optimization with best previous configuration [Feurer 2015]
- Prune search space: restrict search to bounding-box of best solution found previously [Peronne 2019]
- Leverage feature describing task to find most similar tasks [Wistuba 2015, Jomaa 2021]
- Surrogates: ABLR [Peronne 2018] / Deep kernel [Wistuba 2021]
- Learning curve learning [Wistuba 2020]
- Prior-based:
    - Learn prior from offline evaluations [Salinas 2019]



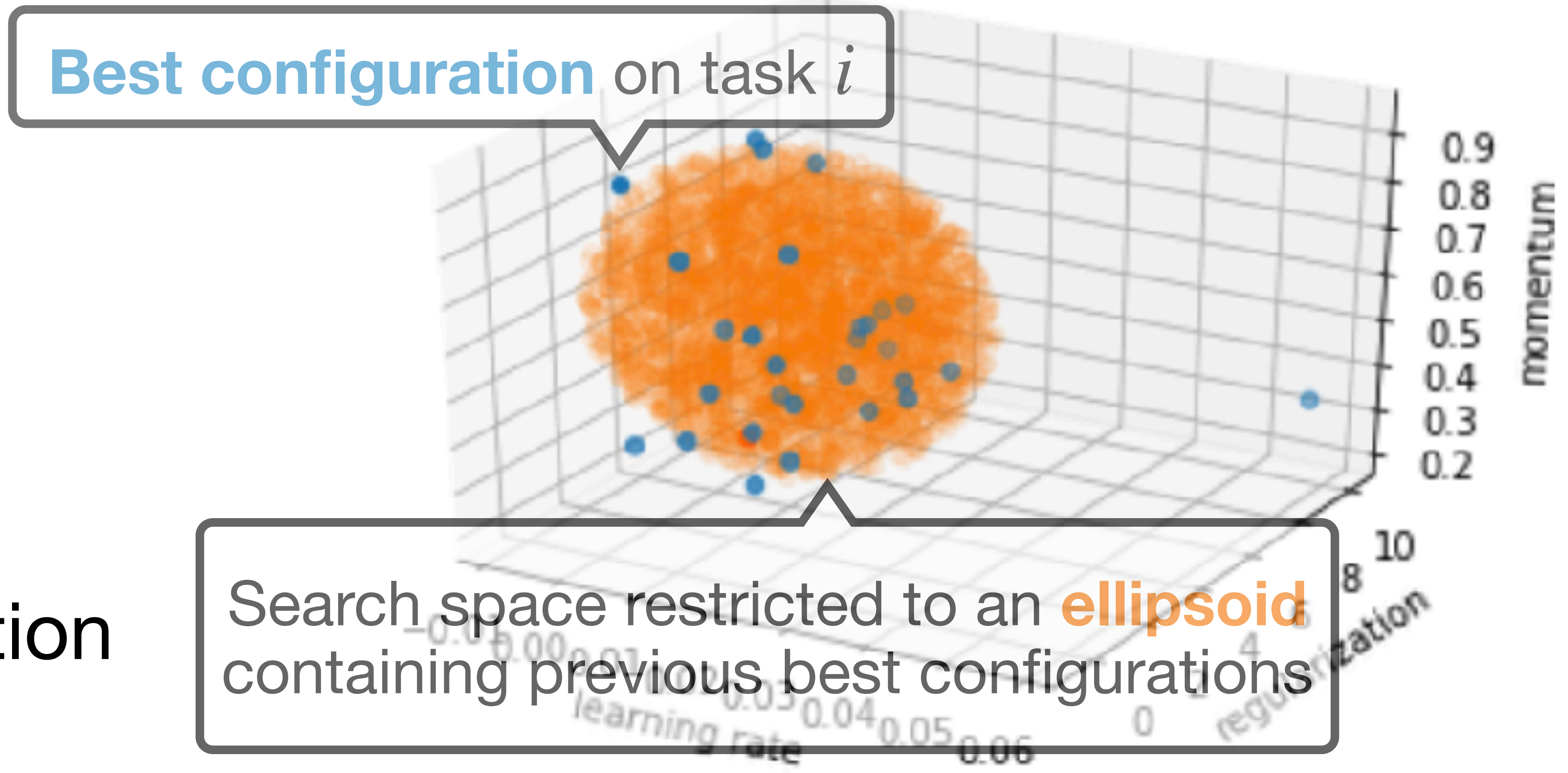

Prune search space to an ellipse containing the best hyperparameters of previous tasks [Peronne 2019]



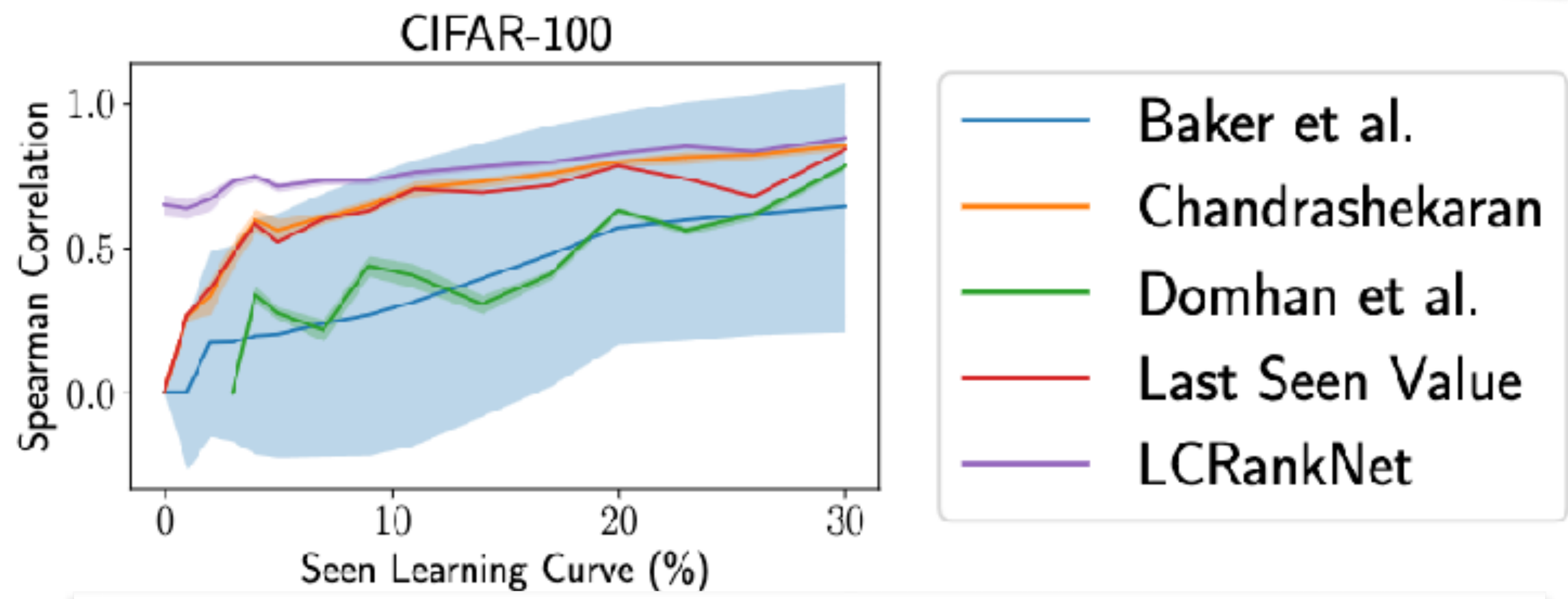

Learning curve model learned on offline evaluations allows to predict final values with high correlations [Wistuba 2020]

# Transfer learning methods

- Warm-starting: initialise Bayesian Optimization with best previous configuration [Feurer 2015]
- Prune search space: restrict search to bounding-box of best solution found previously [Peronne 2019]
- Leverage feature describing task to find most similar tasks [Wistuba 2015, Jomaa 2021]
- Surrogates: ABLR [Peronne 2018] / Deep kernel [Wistuba 2021]
- Learning curve learning [Wistuba 2020]
- Prior-based:
    - Learn prior from offline evaluations [Salinas 2019]
    - Leverage user priors: Priorband [Mallik 2023]



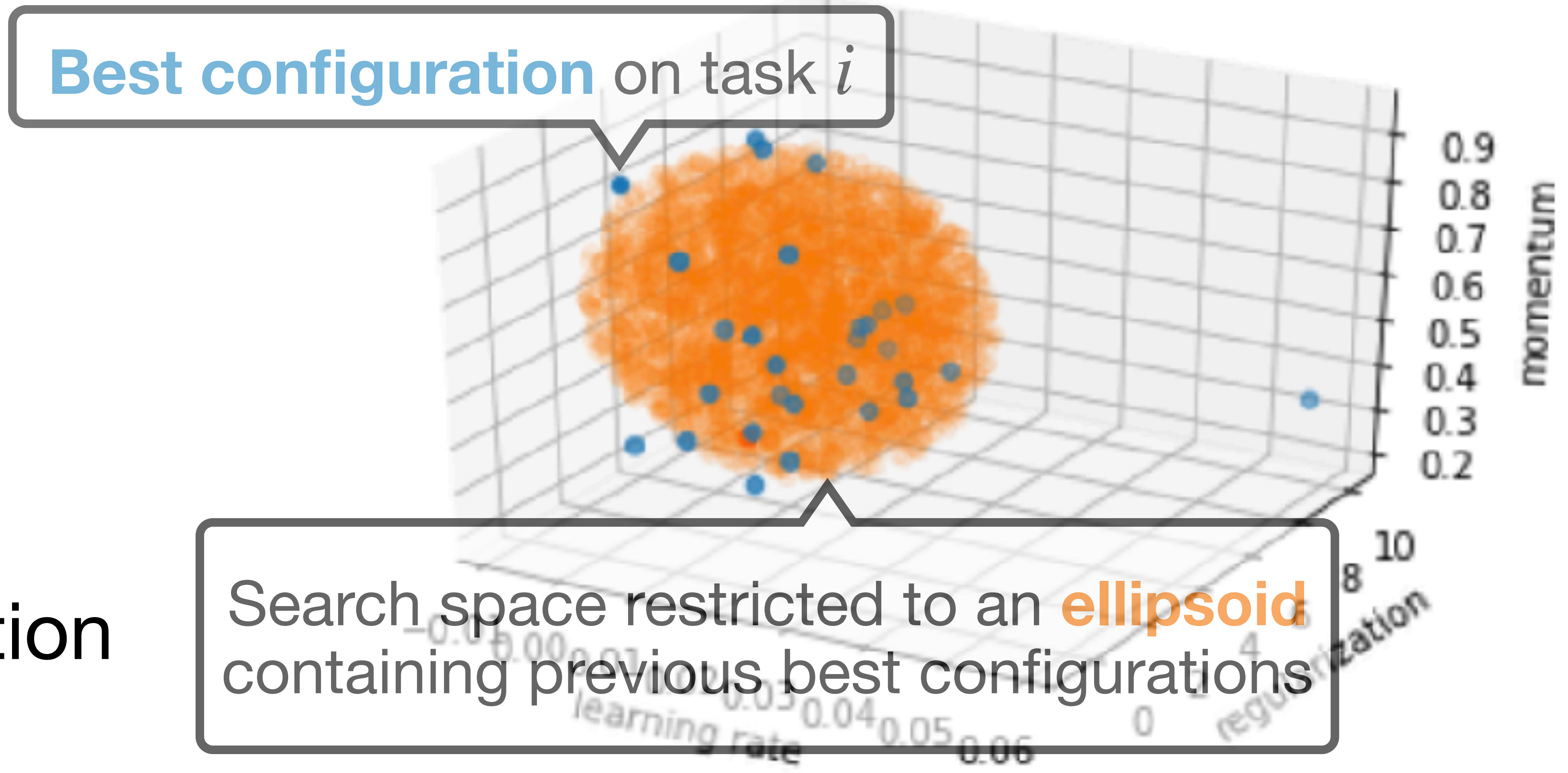

Prune search space to an ellipse containing the best hyperparameters of previous tasks [Peronne 2019]



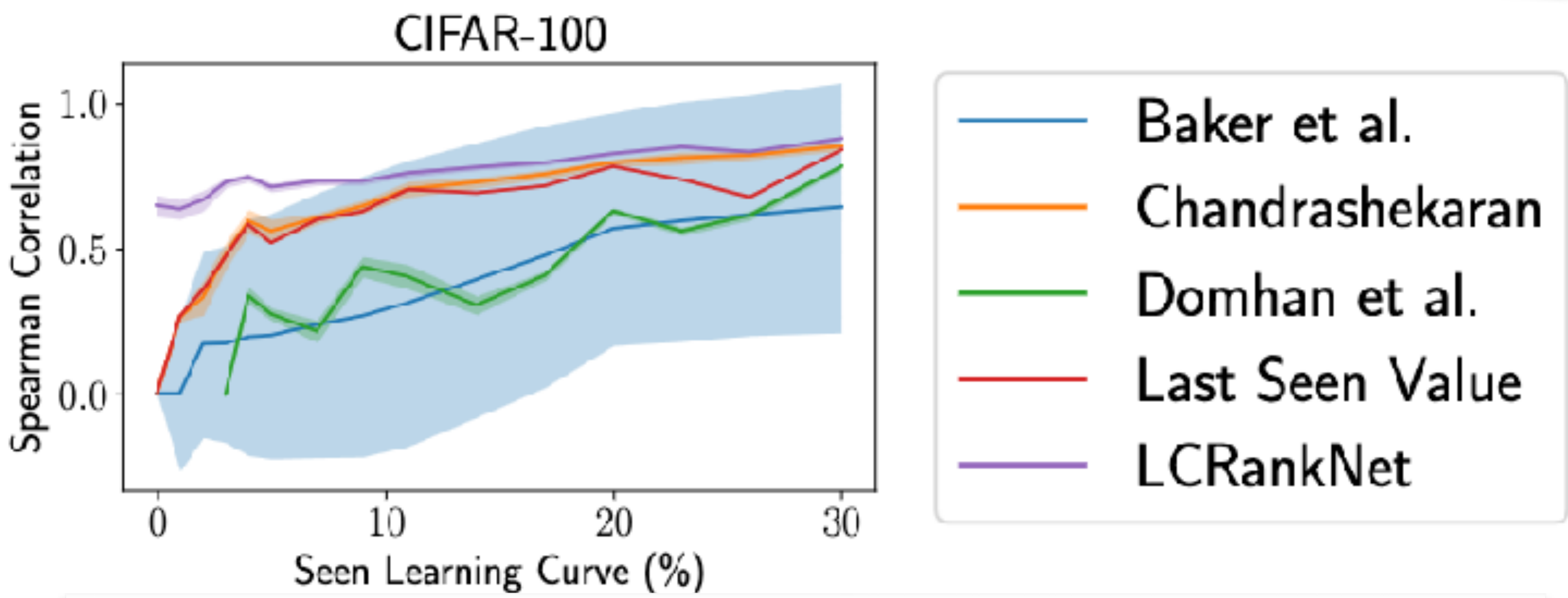

Learning curve model learned on offline evaluations allows to predict final values with high correlations [Wistuba 2020]

# Transfer learning methods

- Warm-starting: initialise Bayesian Optimization with best previous configuration [Feurer 2015]
- Prune search space: restrict search to bounding-box of best solution found previously [Peronne 2019]
- Leverage feature describing task to find most similar tasks [Wistuba 2015, Jomaa 2021]
- Surrogates: ABLR [Peronne 2018] / Deep kernel [Wistuba 2021]
- Learning curve learning [Wistuba 2020]
- Prior-based:
    - Learn prior from offline evaluations [Salinas 2019]
    - Leverage user priors: Priorband [Mallik 2023]
- Portfolio learning [Wistuba 2015]



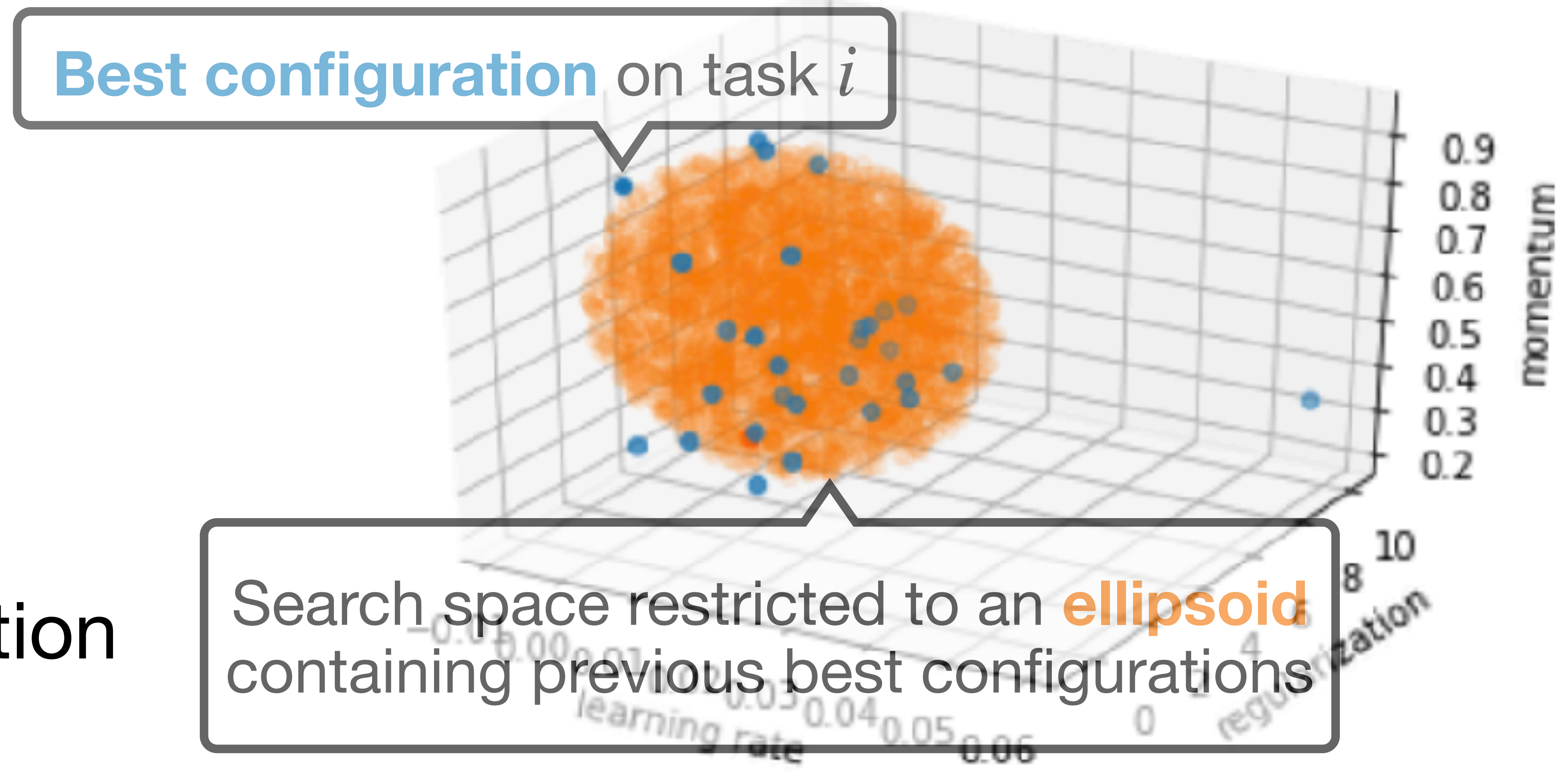

Prune search space to an ellipse containing the best hyperparameters of previous tasks [Peronne 2019]



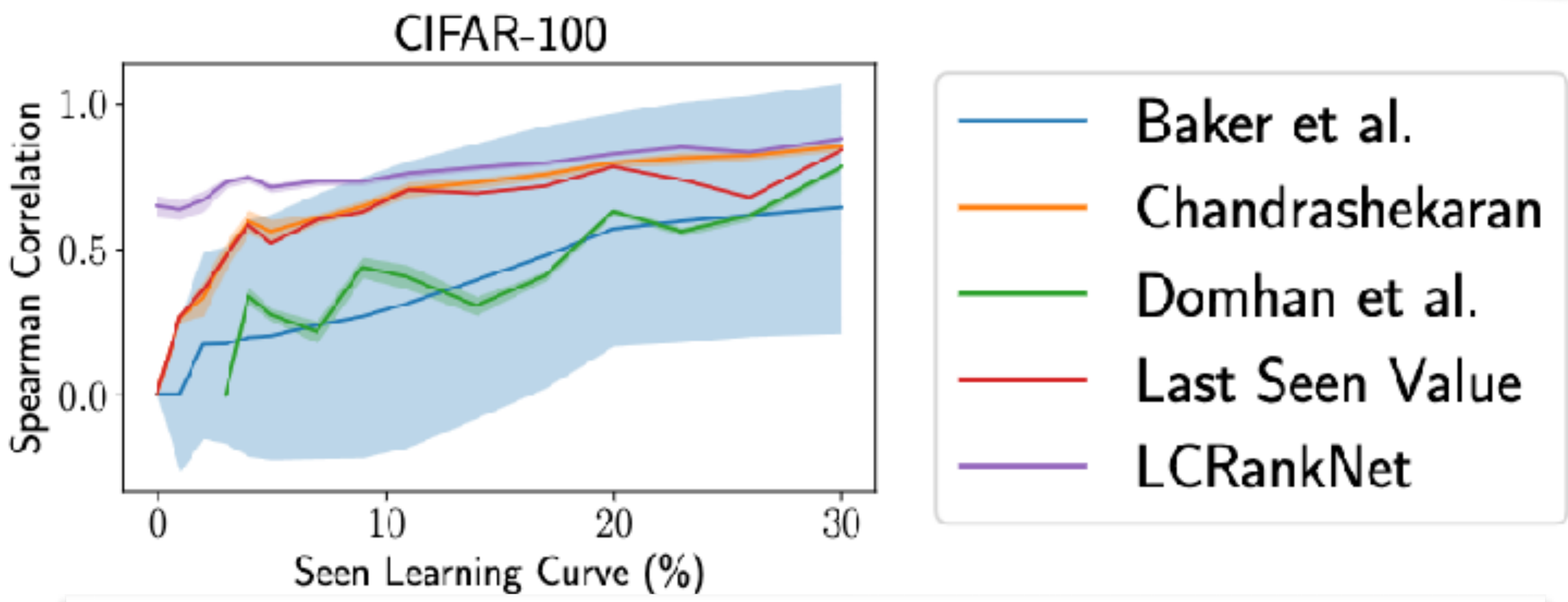

Learning curve model learned on offline evaluations allows to predict final values with high correlations [Wistuba 2020]

# Transfer learning methods

- Warm-starting: initialise Bayesian Optimization with best previous configuration [Feurer 2015]
- Prune search space: restrict search to bounding-box of best solution found previously [Peronne 2019]
- Leverage feature describing task to find most similar tasks [Wistuba 2015, Jomaa 2021]
- Surrogates: ABLR [Peronne 2018] / Deep kernel [Wistuba 2021]
- Learning curve learning [Wistuba 2020]
- Prior-based:
  - Learn prior from offline evaluations [Salinas 2019]
  - Leverage user priors: Priorband [Mallik 2023]
- Portfolio learning [Wistuba 2015]
- Foundational model: Optformer



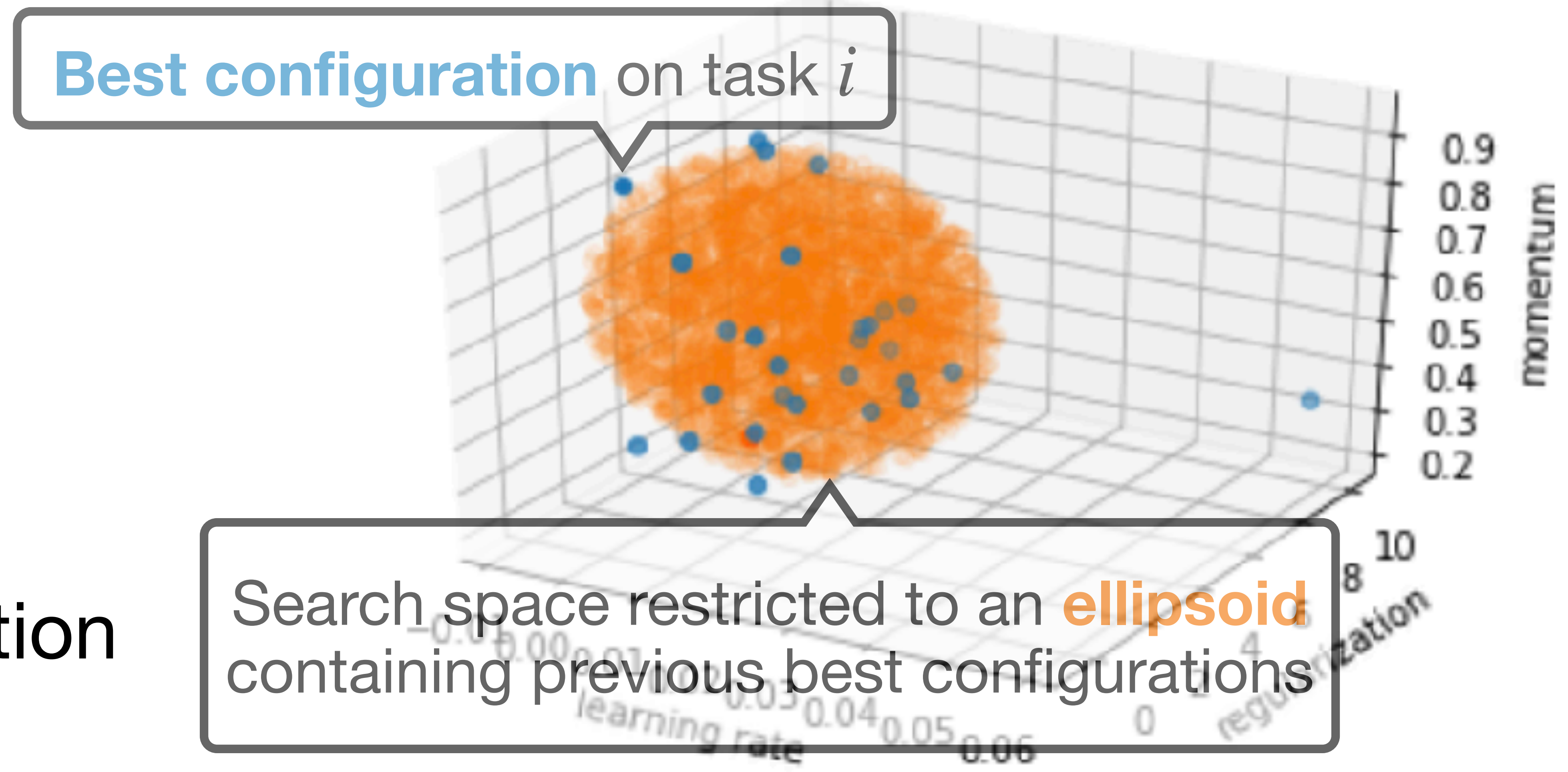

Prune search space to an ellipse containing the best hyperparameters of previous tasks [Peronne 2019]



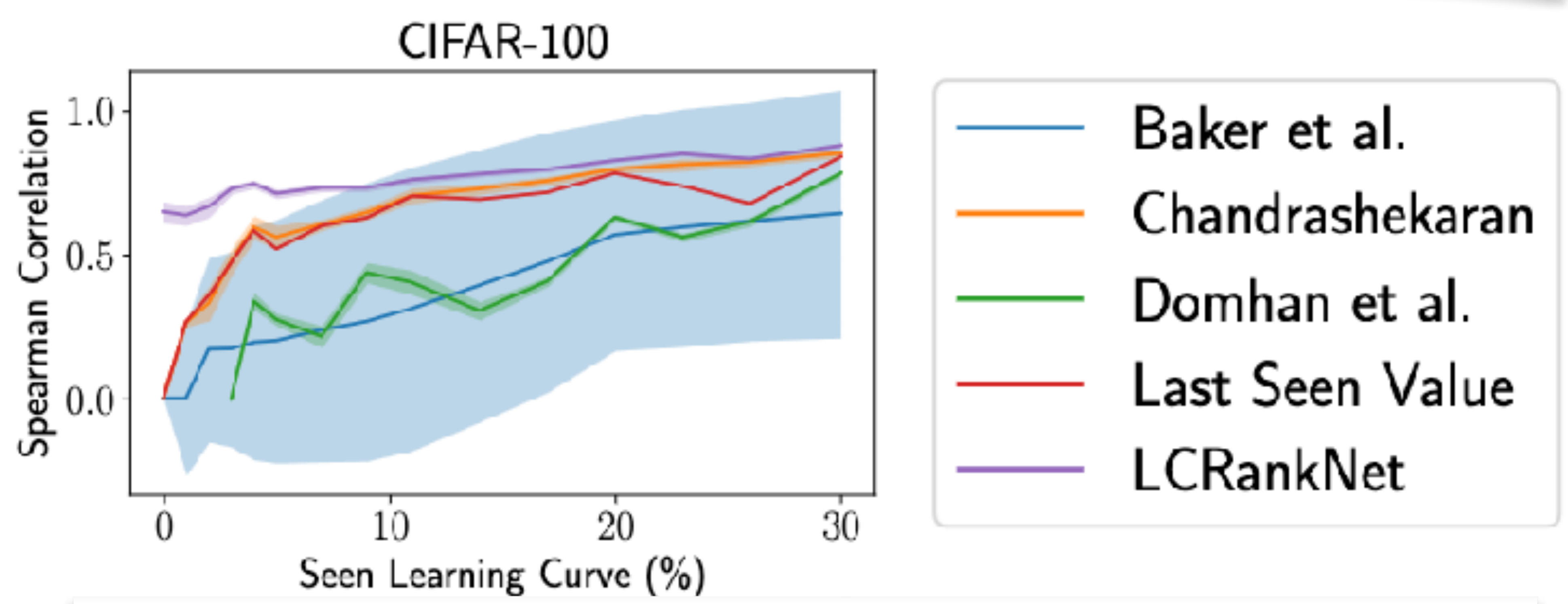

Learning curve model learned on offline evaluations allows to predict final values with high correlations [Wistuba 2020]

# Transfer learning methods

- Warm-starting: initialise Bayesian Optimization with best previous configuration [Feurer 2015]
- Prune search space: restrict search to bounding-box of best solution found previously [Peronne 2019]
- Leverage feature describing task to find most similar tasks [Wistuba 2015, Jomaa 2021]
- Surrogates: ABLR [Peronne 2018] / Deep kernel [Wistuba 2021]
- Learning curve learning [Wistuba 2020]
- Prior-based:
  - Learn prior from offline evaluations [Salinas 2019]
  - Leverage user priors: Priorband [Mallik 2023]
- Portfolio learning [Wistuba 2015]
- Foundational model: Optformer




Prune search space to an ellipse containing the best hyperparameters of previous tasks [Peronne 2019]




Learning curve model learned on offline evaluations allows to predict final values with high correlations [Wistuba 2020]

We will focus on those methods

# Methods

## Prior based methods

# Transfer learning

## Reoccurring difficulties

# Transfer learning

## Reoccurring difficulties

- Offline evaluations $\mathcal{D}^M = \bigcup_{i=1}^{M} \{x_i^j, y_i^j\}_{i=1}^{N_j}$ available for $M$ tasks

# Transfer learning

## Reoccurring difficulties

- Offline evaluations $\mathcal{D}^M = \bigcup_{i=1}^{M} \{x_i^j, y_i^j\}_{i=1}^{N_j}$ available for $M$ tasks
- Scale of objective $y_i^j$ often vary significantly across tasks

# Transfer learning

## Reoccurring difficulties

- Offline evaluations $\mathcal{D}^M = \bigcup_{i=1}^{M} \{x_i^j, y_i^j\}_{i=1}^{N_j}$ available for $M$ tasks
- Scale of objective $y_i^j$ often vary significantly across tasks
- Noise may be (very) far from Gaussian, annoying to apply Bayesian Optimization

# Transfer learning

## Reoccurring difficulties

- Offline evaluations $\mathscr{D}^M = \bigcup_{i=1}^{M} \{x_i^j, y_i^j\}_{i=1}^{N_j}$ available for $M$ tasks
- Scale of objective $y_i^j$ often vary significantly across tasks
- Noise may be (very) far from Gaussian, annoying to apply Bayesian Optimization
- Need to handle many observations: hard to apply (approximate) Gaussian Process (the cost of applying GP on $n$ evaluations is $\mathcal{O}(n^3)$)

# Transfer learning

## Reoccurring difficulties

- Offline evaluations $\mathscr{D}^M = \bigcup_{i=1}^{M} \{x_i^j, y_i^j\}_{i=1}^{N_j}$ available for $M$ tasks
- Scale of objective $y_i^j$ often vary significantly across tasks
- Noise may be (very) far from Gaussian, annoying to apply Bayesian Optimization
- Need to handle many observations: hard to apply (approximate) Gaussian Process (the cost of applying GP on $n$ evaluations is $\mathcal{O}(n^3)$)
- Avoid negative transfer (aka catastrophic *remembering*)

# Transfer learning

## Reoccurring difficulties

- Offline evaluations $\mathscr{D}^M = \bigcup_{i=1}^{M} \{x_i^j, y_i^j\}_{i=1}^{N_j}$ available for $M$ tasks
- Scale of objective $y_i^j$ often vary significantly across tasks
- Noise may be (very) far from Gaussian, annoying to apply Bayesian Optimization
- Need to handle many observations: hard to apply (approximate) Gaussian Process (the cost of applying GP on $n$ evaluations is $\mathcal{O}(n^3)$)
- Avoid negative transfer (aka catastrophic *remembering*)

# Transfer learning

## Reoccurring difficulties

- Offline evaluations $\mathcal{D}^M = \bigcup_{i=1}^{M} \{x_i^j, y_i^j\}_{i=1}^{N_j}$ available for $M$ tasks

We will discuss how to adressing scale issues…

- Scale of objective $y_i^j$ often vary significantly across tasks
- Noise may be (very) far from Gaussian, annoying to apply Bayesian Optimization
- Need to handle many observations: hard to apply (approximate) Gaussian Process (the cost of applying GP on $n$ evaluations is $\mathcal{O}(n^3)$)
- Avoid negative transfer (aka catastrophic *remembering*)

# Transfer learning
## Reoccurring difficulties

- Offline evaluations $\mathcal{D}^M = \bigcup_{i=1}^{M} \{x_i^j, y_i^j\}_{i=1}^{N_j}$ available for $M$ tasks

We will discuss how to adressing scale issues…

- Scale of objective $y_i^j$ often vary significantly across tasks
- Noise may be (very) far from Gaussian, annoying to apply Bayesian Optimization
- Need to handle many observations: hard to apply (approximate) Gaussian Process (the cost of applying GP on $n$ evaluations is $\mathcal{O}(n^3)$)

… then how to address computational issues

- Avoid negative transfer (aka catastrophic *remembering*)

# Gaussian Copula Transform

# Gaussian Copula Transform

- It would be easier if distribution $y^j$ for each task was Gaussian…

# Gaussian Copula Transform

- It would be easier if distribution $y^j$ for each task was Gaussian…
    - Apply change of variable $\Psi = \Phi^{-1} \circ F$

# Gaussian Copula Transform

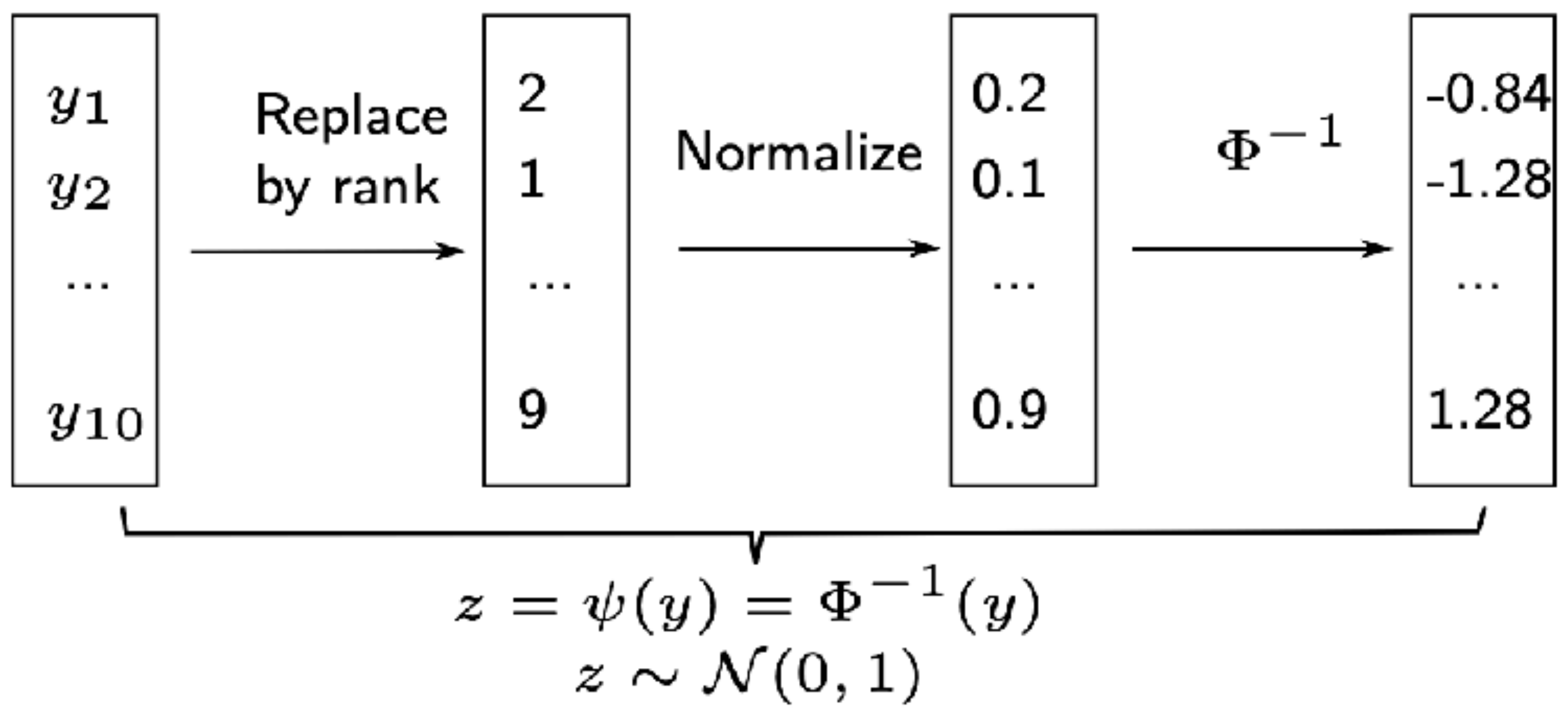
- It would be easier if distribution $y^j$ for each task was Gaussian…
  - Apply change of variable $\Psi = \Phi^{-1} \circ F$
  - $\Phi$: Gaussian CDF

# Gaussian Copula Transform

- It would be easier if distribution $y^j$ for each task was Gaussian…
  - Apply change of variable $\Psi = \Phi^{-1} \circ$ F
  - $\Phi$: Gaussian CDF
  - F: CDF of evaluations of a given task

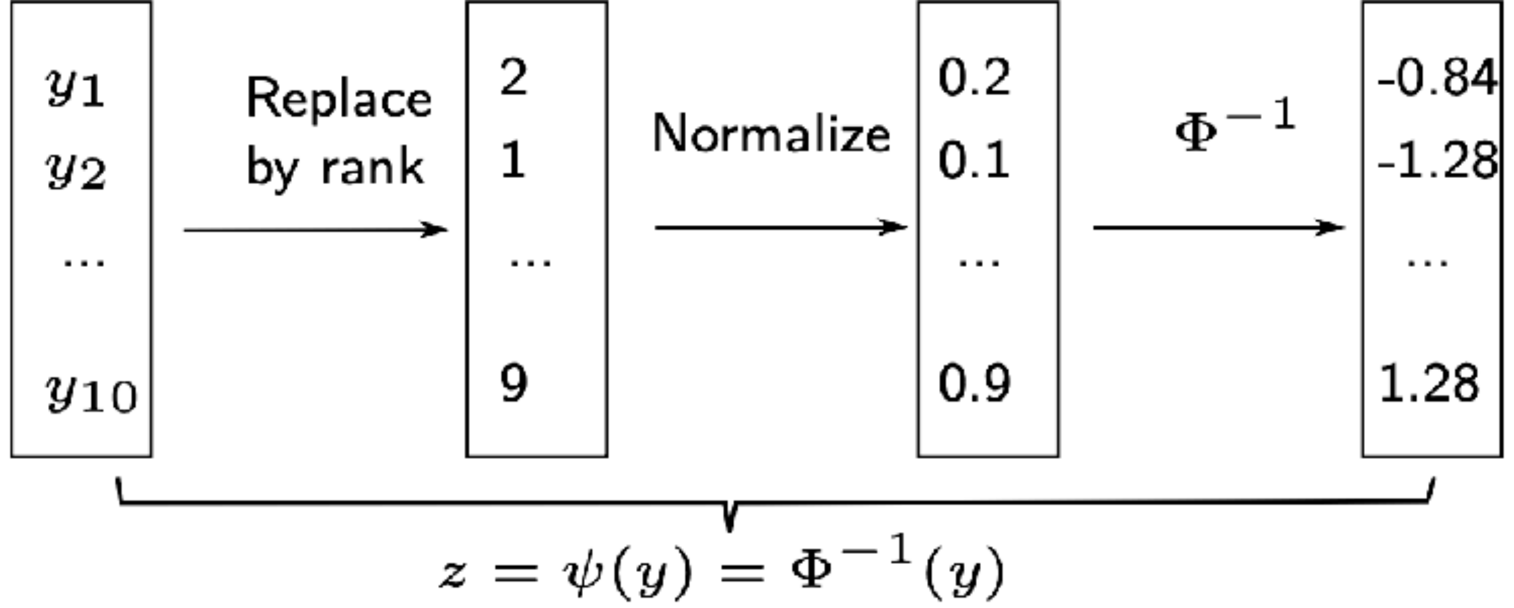
# Gaussian Copula Transform

- It would be easier if distribution $y^j$ for each task was Gaussian…
  - Apply change of variable $\Psi = \Phi^{-1} \circ$ F
  - $\Phi$: Gaussian CDF
  - F: CDF of evaluations of a given task

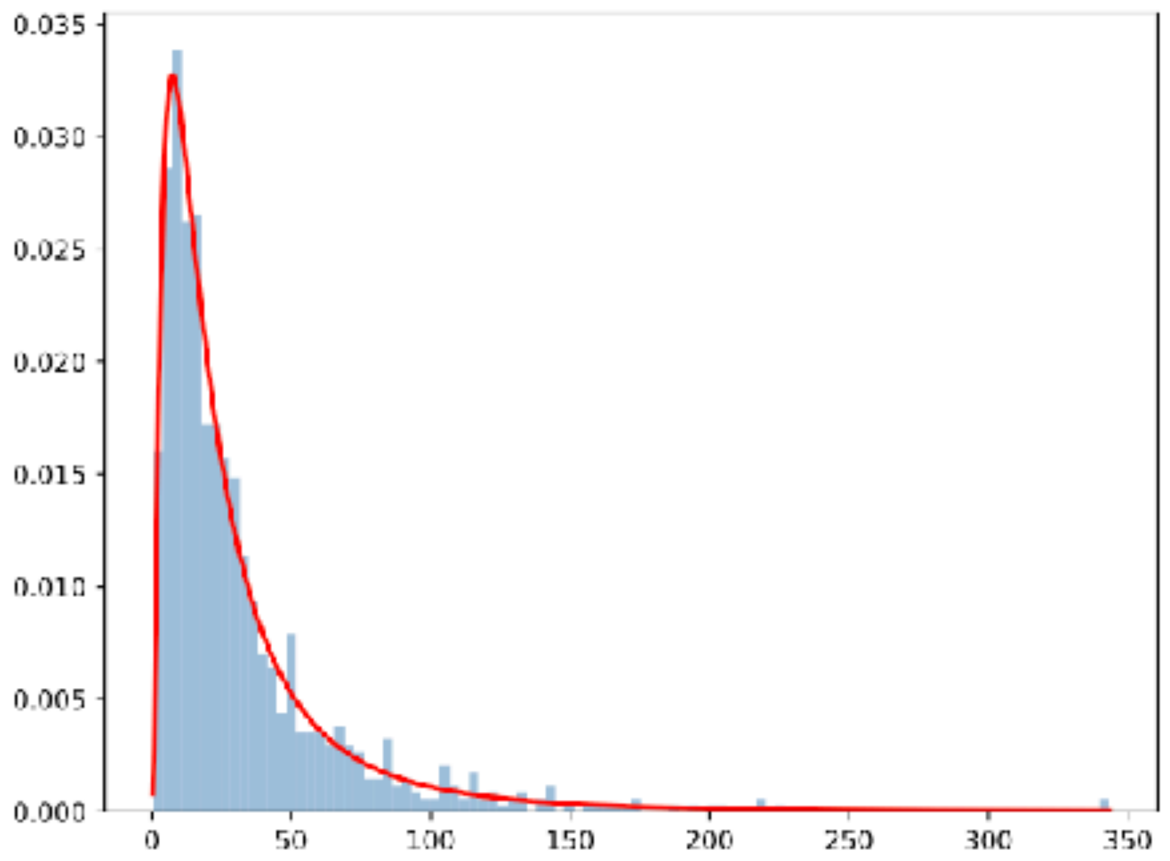


# Gaussian Copula Transform

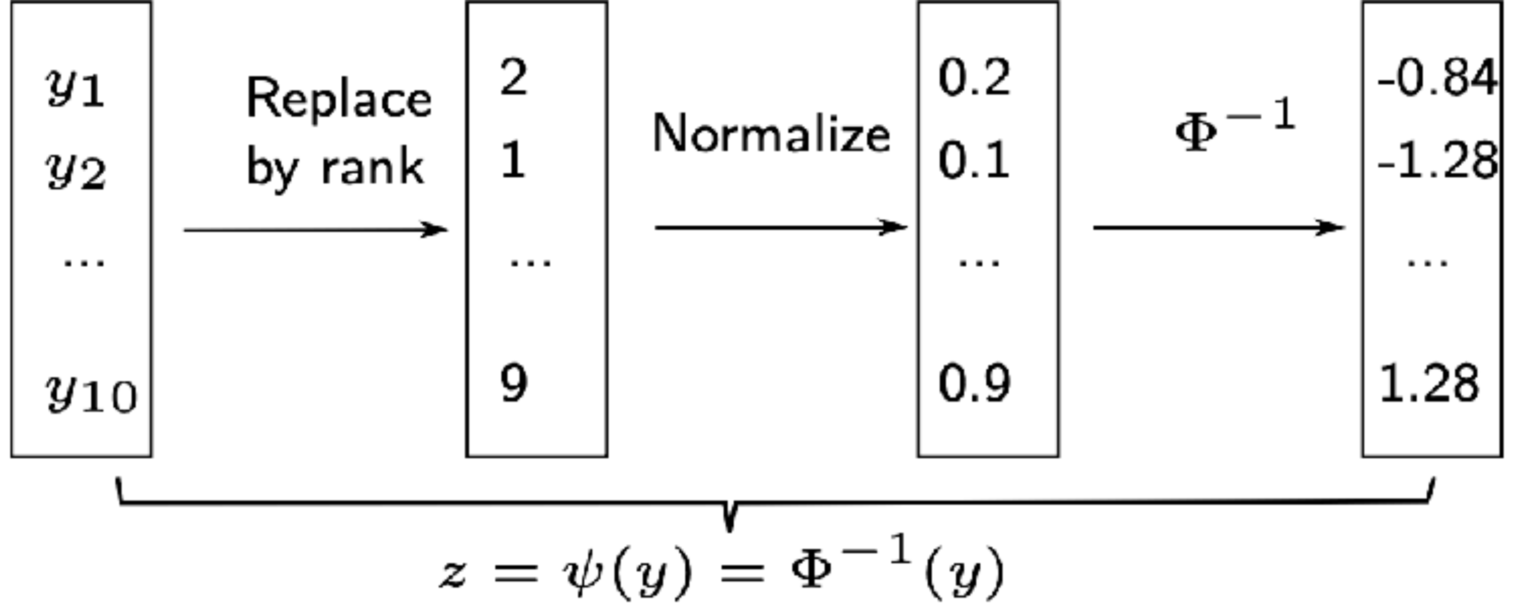
## Nice properties




$$y \xrightarrow{F} u = F(y) \xrightarrow{\Phi^{-1}} z = \Phi^{-1} \circ F(y)$$



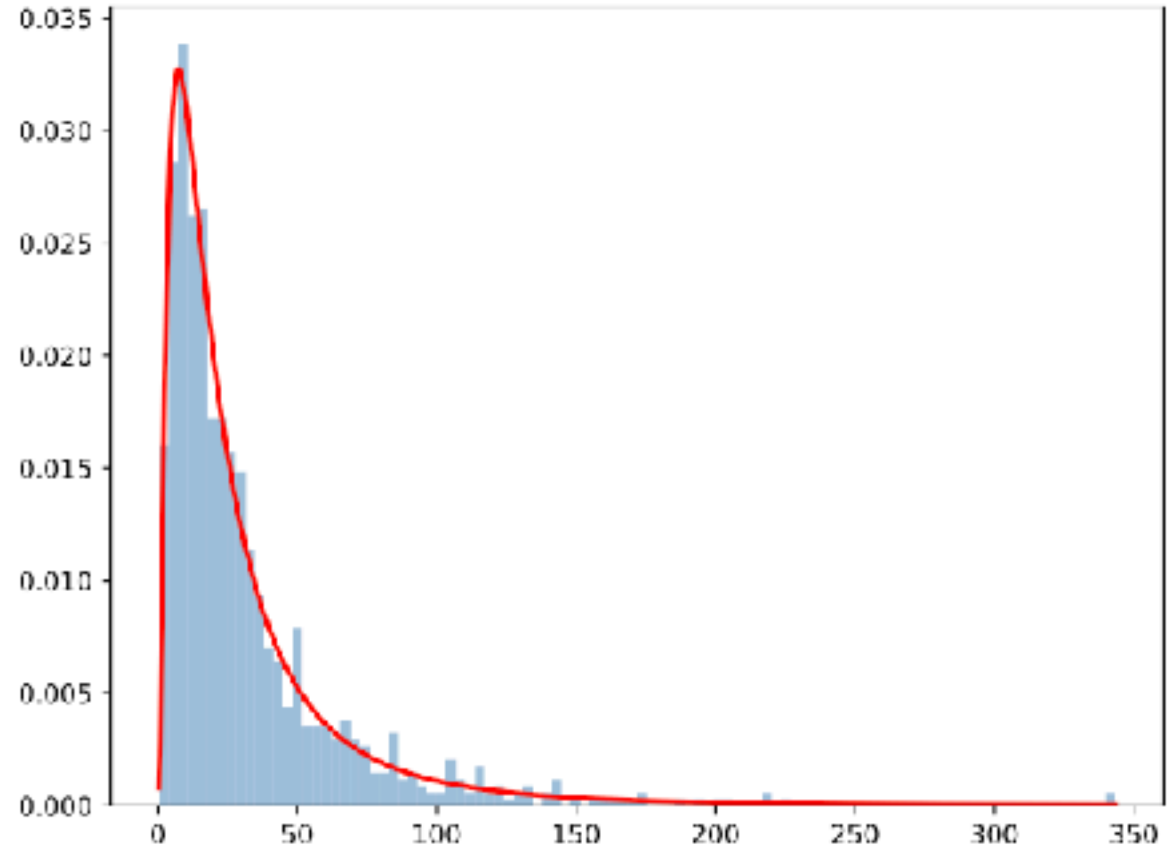

Original distribution




# Gaussian Copula Transform

## Nice properties

$$y \xrightarrow{F} u = F(y) \xrightarrow{\Phi^{-1}} z = \Phi^{-1} \circ F(y)$$

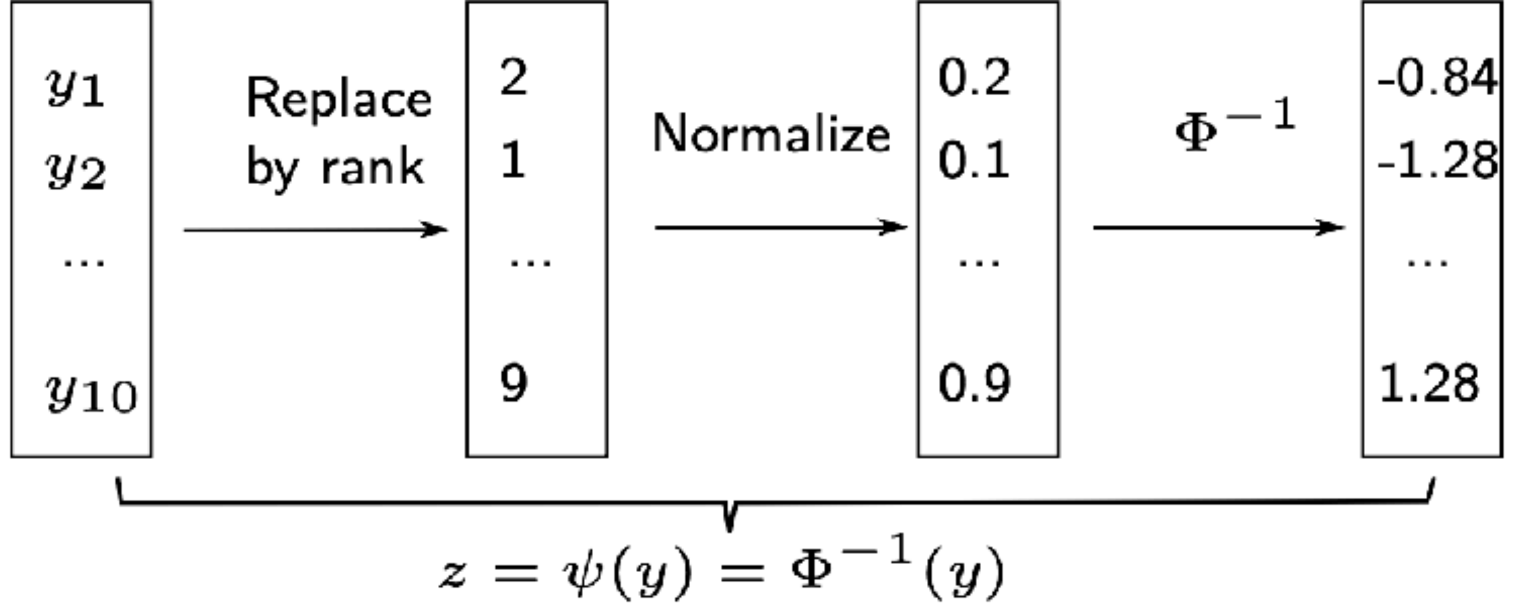



Original distribution

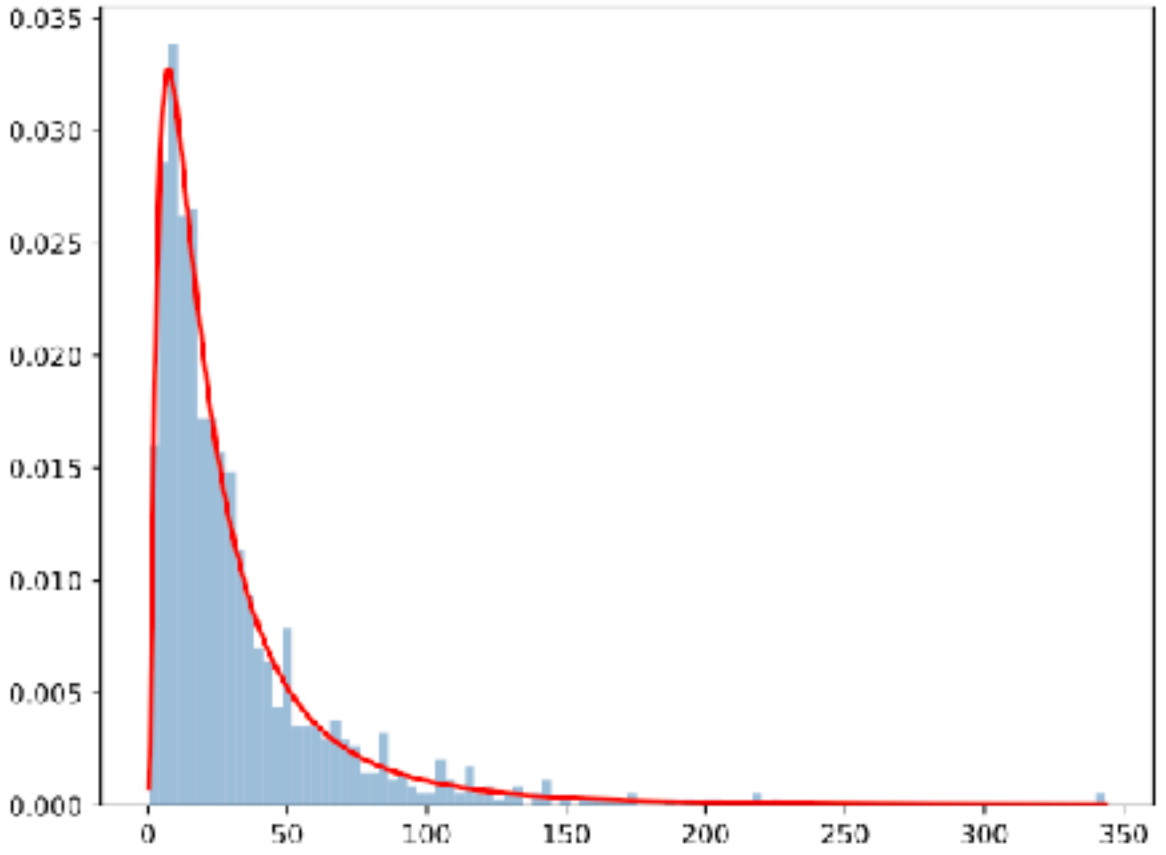
🤔 What is the distribution of $F(y)$?

# Gaussian Copula Transform

## Nice properties




$$y \xrightarrow{F} u = F(y) \xrightarrow{\Phi^{-1}} z = \Phi^{-1} \circ F(y)$$

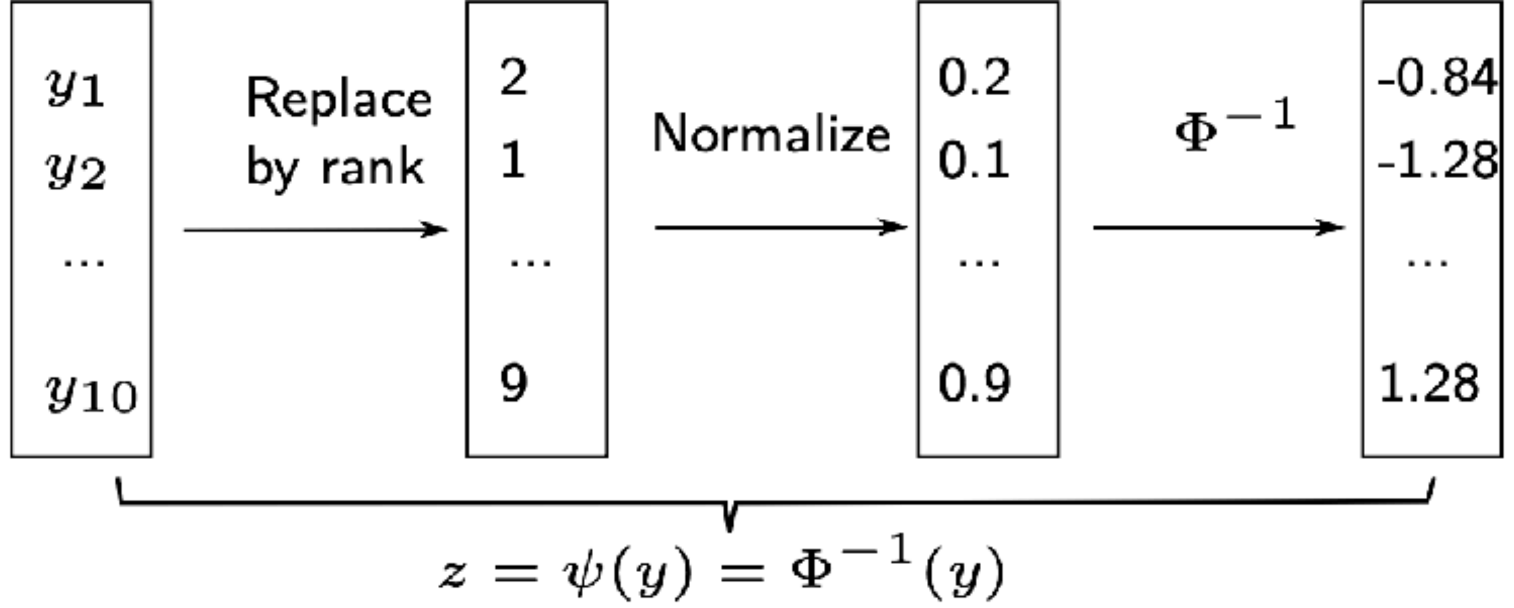



Original distribution

🤔 What is the distribution of $F(y)$?

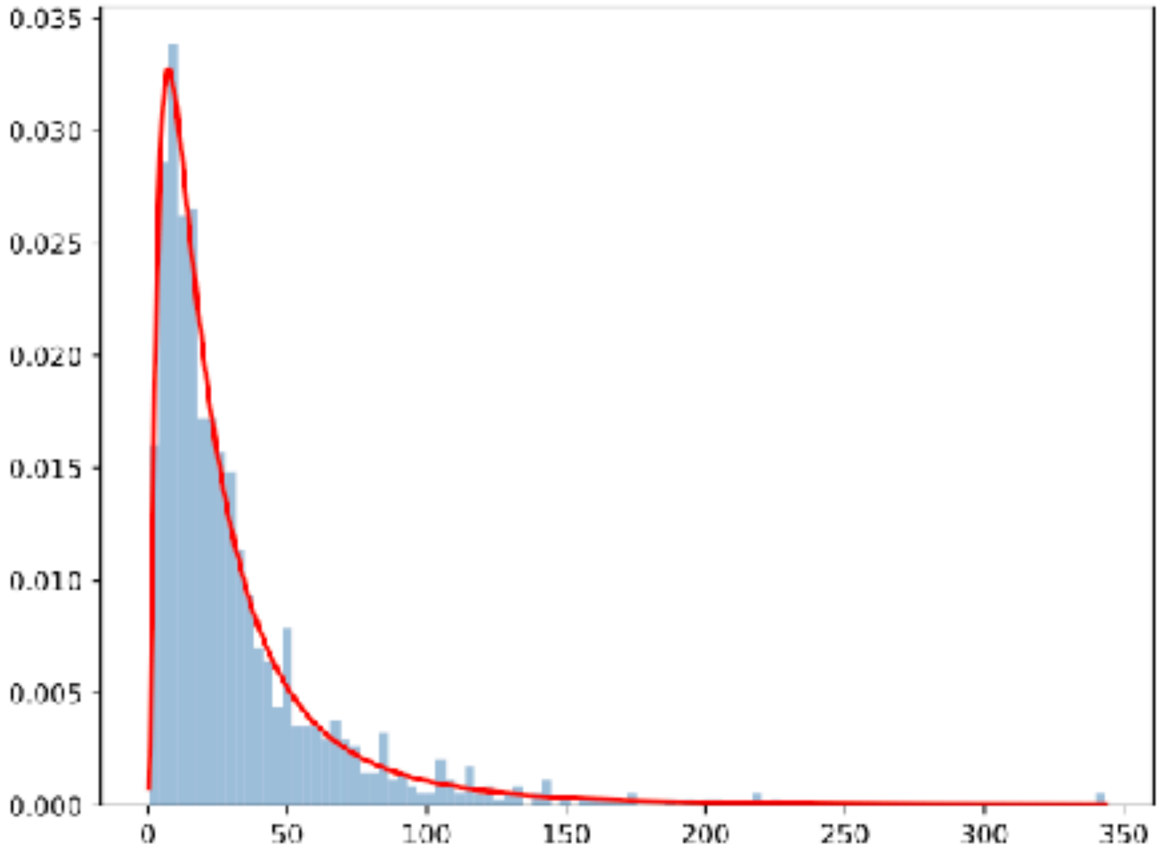
💡 Uniform $\mathcal{U}([0,1])$!

# Gaussian Copula Transform

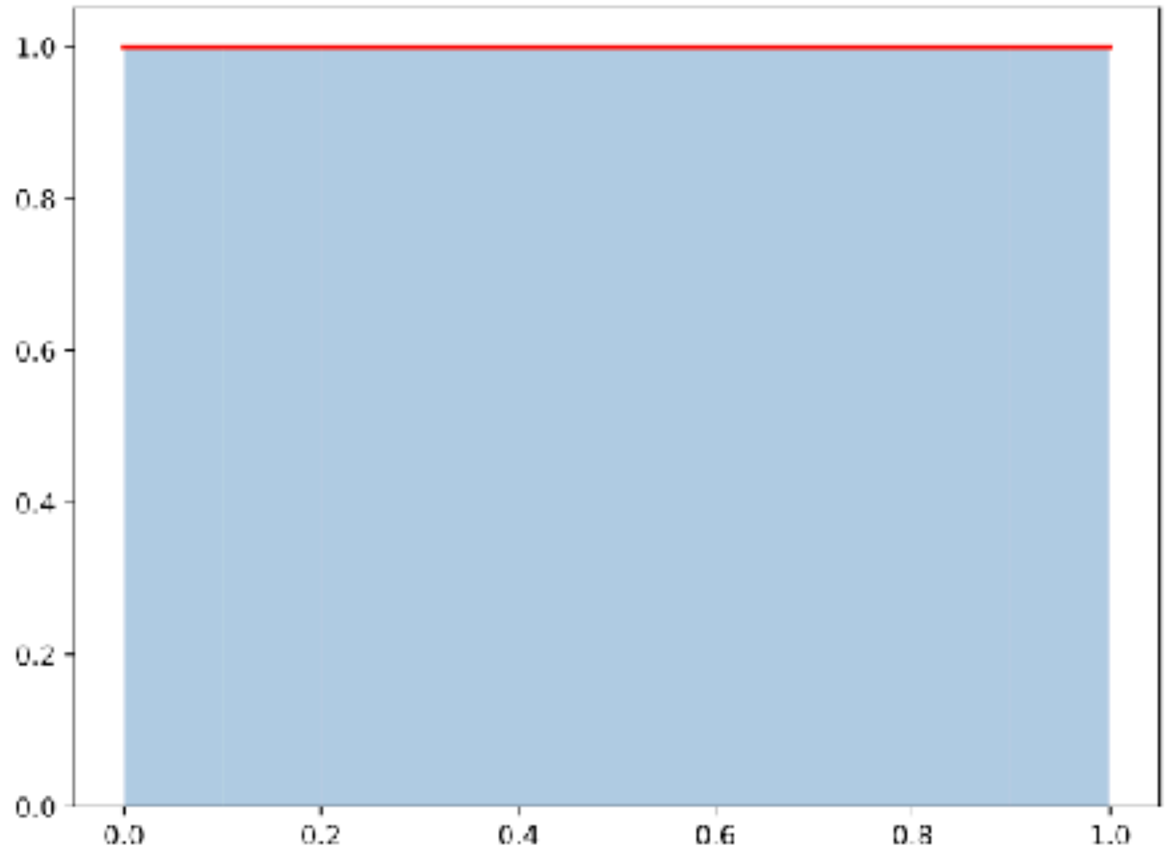



## Nice properties

$y \xrightarrow{F} u = F(y) \xrightarrow{\Phi^{-1}} z = \Phi^{-1} \circ F(y)$



Original distribution



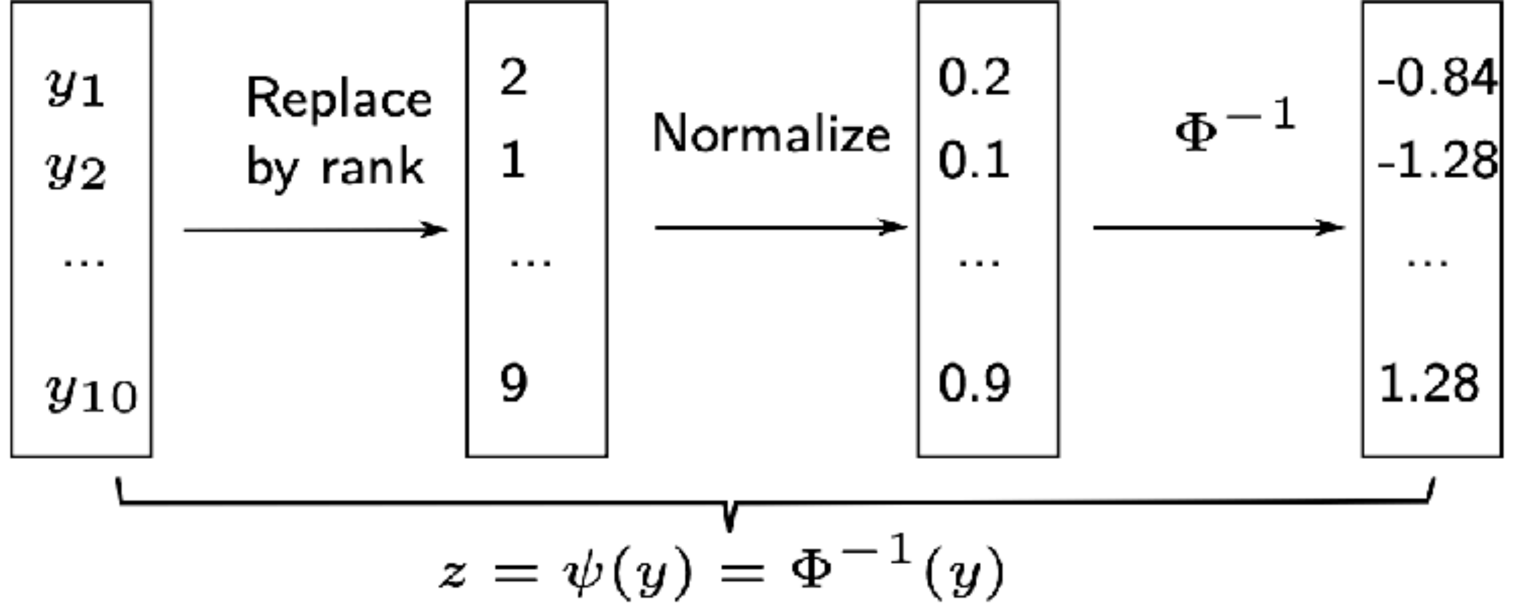
$u \sim \mathcal{U}([0,1])$

🤔 What is the distribution of $F(y)$?

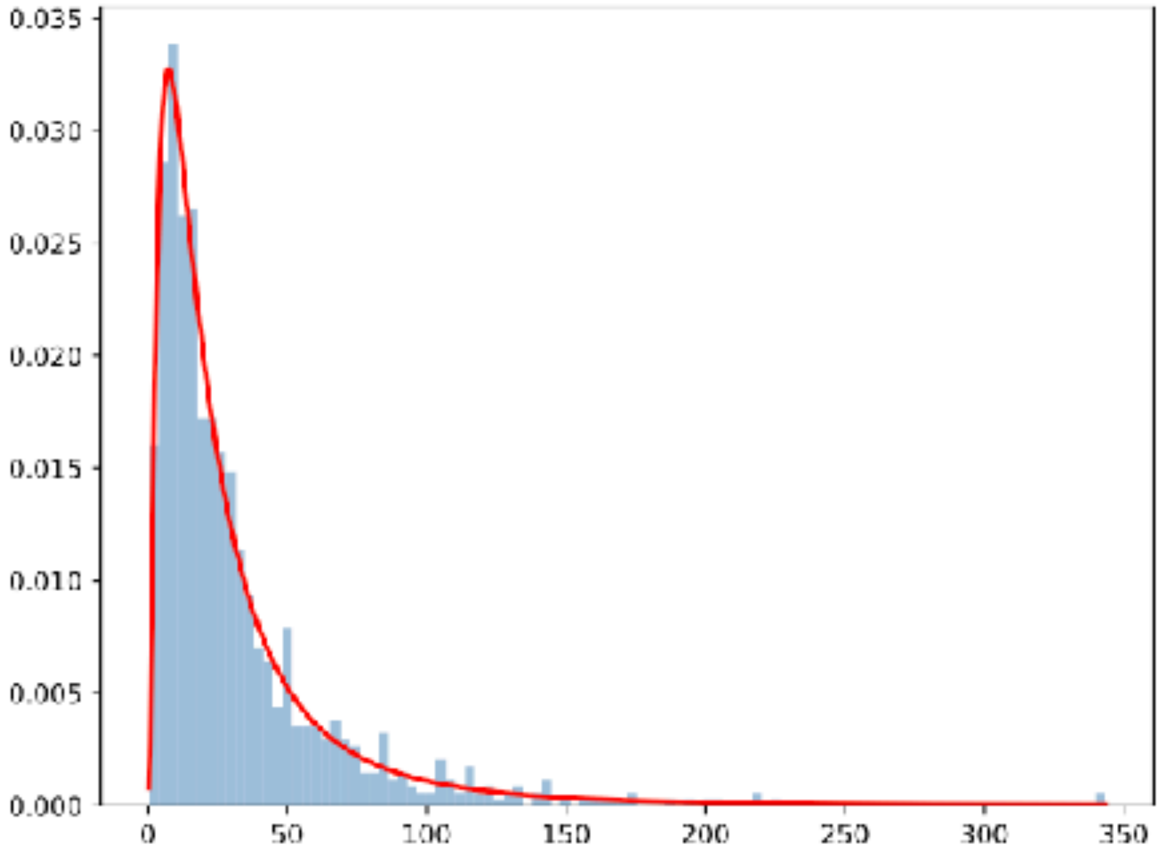
💡 Uniform $\mathcal{U}([0,1])$!

# Gaussian Copula Transform

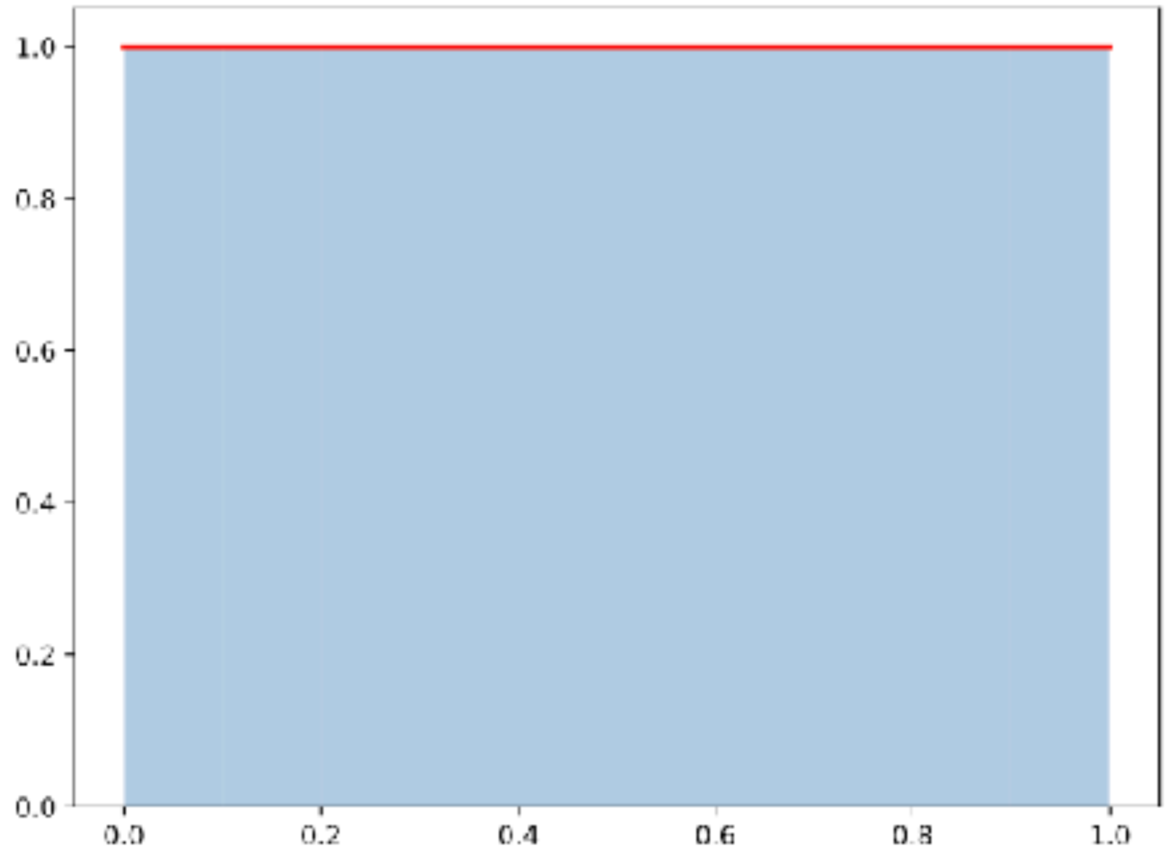
## Nice properties




$y$ $\xrightarrow{F}$ $u = F(y)$ $\xrightarrow{\Phi^{-1}}$ $z = \Phi^{-1} \circ F(y)$




Original distribution

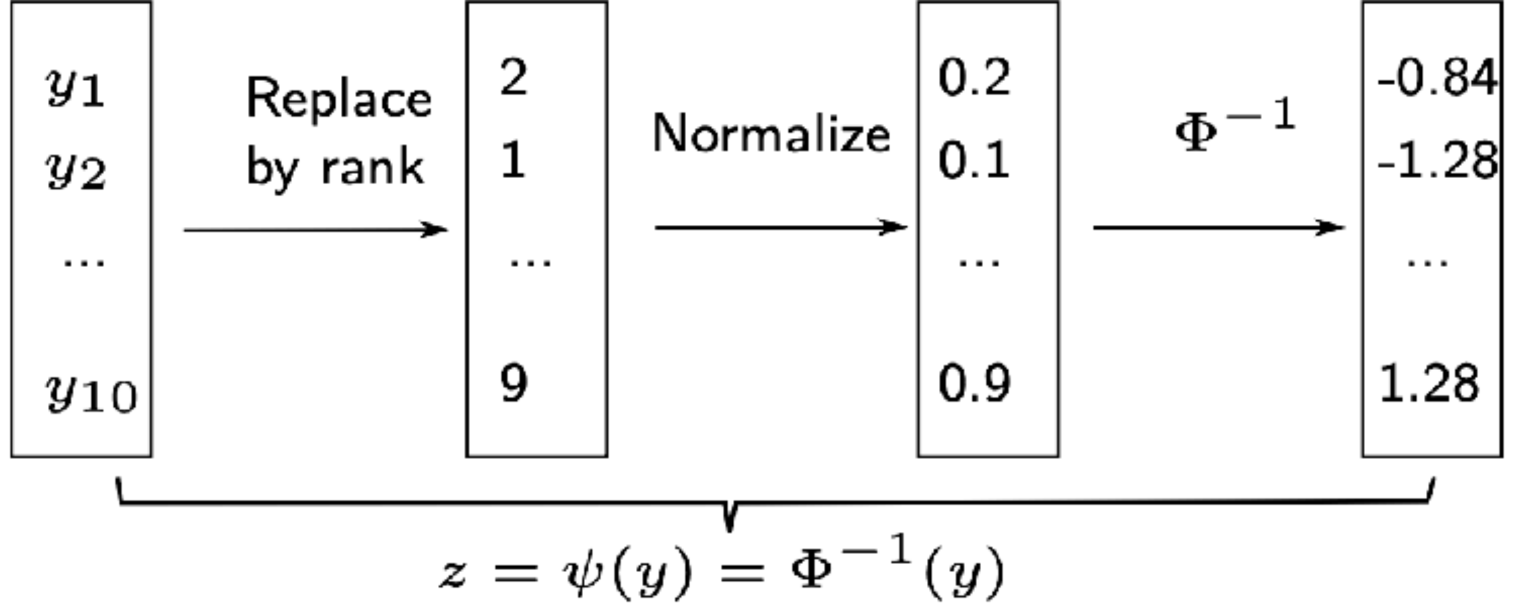



$u \sim \mathcal{U}([0,1])$

🤔 What is the distribution of $F(y)$?

💡 Uniform $\mathcal{U}([0,1])$!

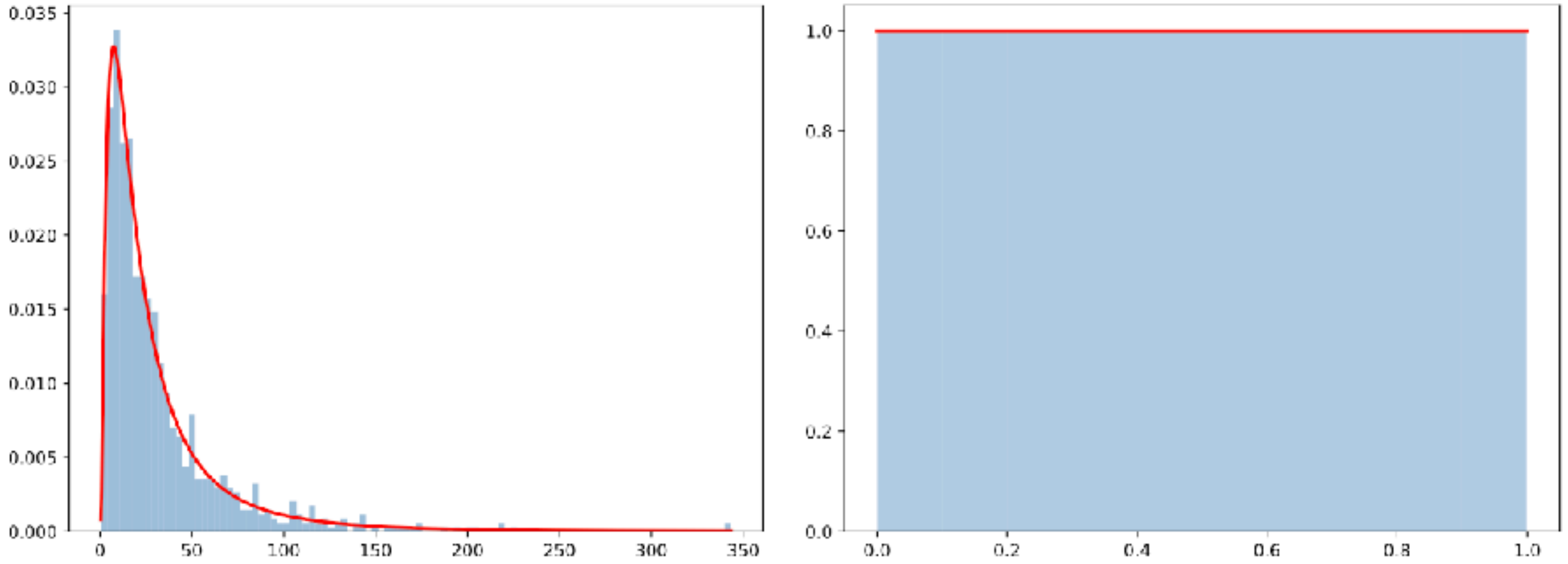
🤔 What is the distribution of $\Phi^{-1} \circ F(y)$?




# Gaussian Copula Transform

## Nice properties

$y \xrightarrow{F} u = F(y) \xrightarrow{\Phi^{-1}} z = \Phi^{-1} \circ F(y)$



Original distribution

$u \sim \mathcal{U}([0,1])$

🤔 What is the distribution of $F(y)$?

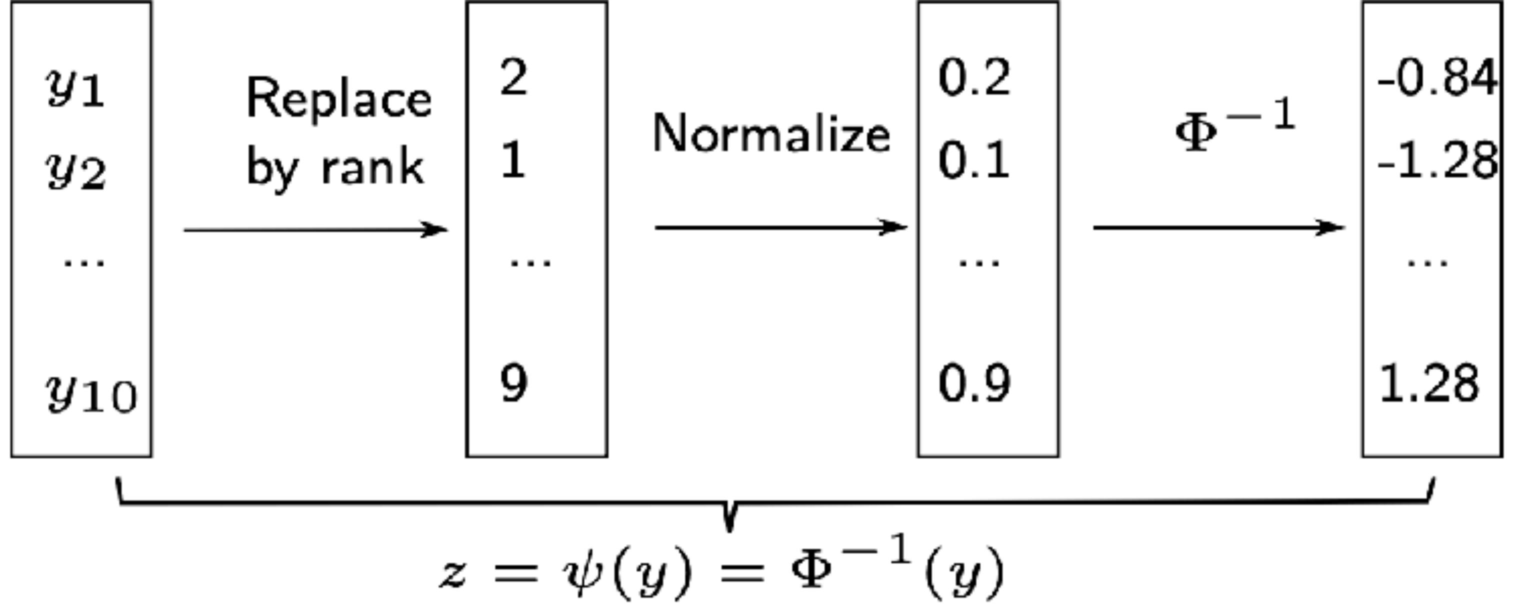
💡 Uniform $\mathcal{U}([0,1])$!

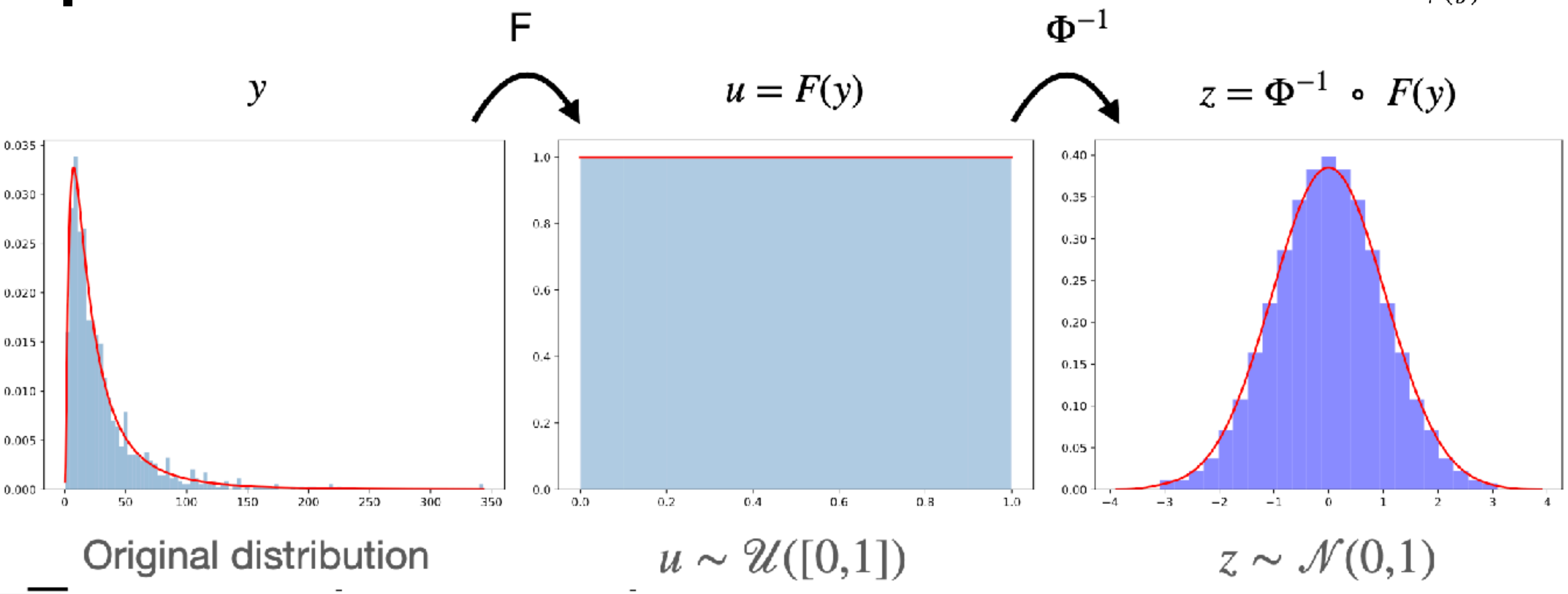
🤔 What is the distribution of $\Phi^{-1} \circ F(y)$?

💡 Normal $\mathcal{N}(0,1)$!

# Gaussian Copula Transform

## Nice properties




🤔 What is the distribution of $F(y)$?

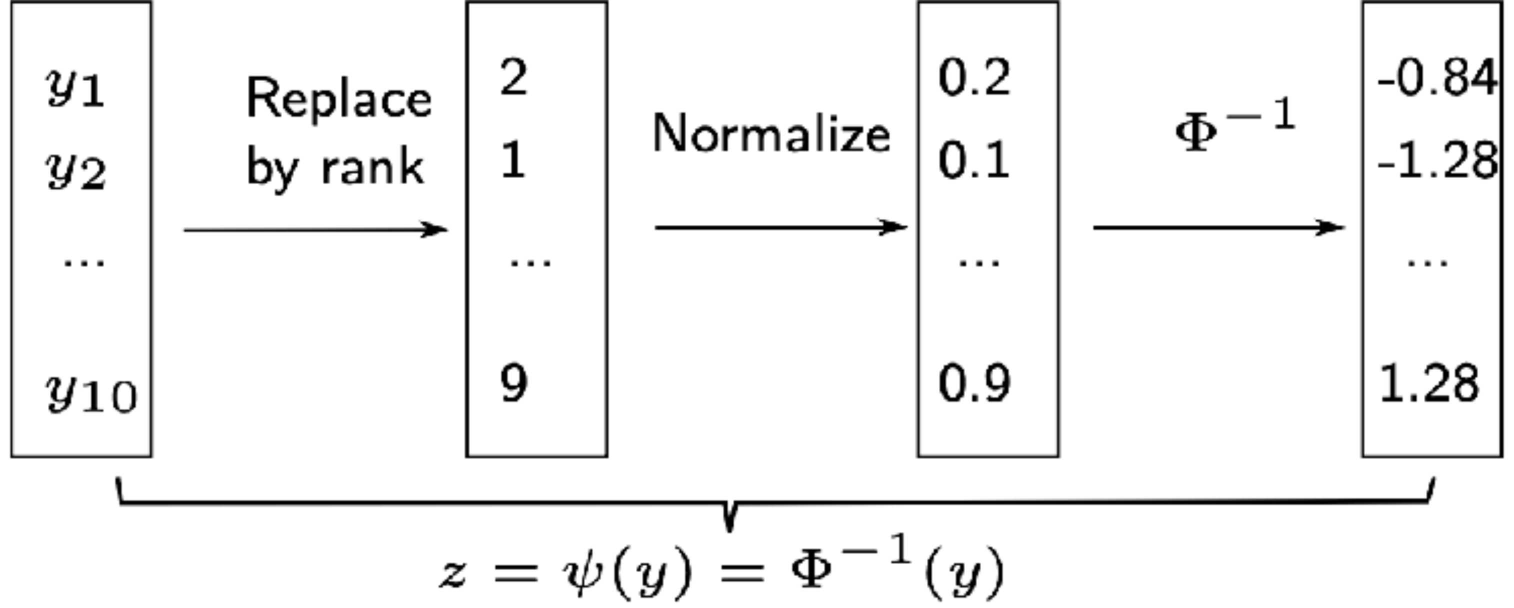
💡 Uniform $\mathcal{U}([0,1])$!

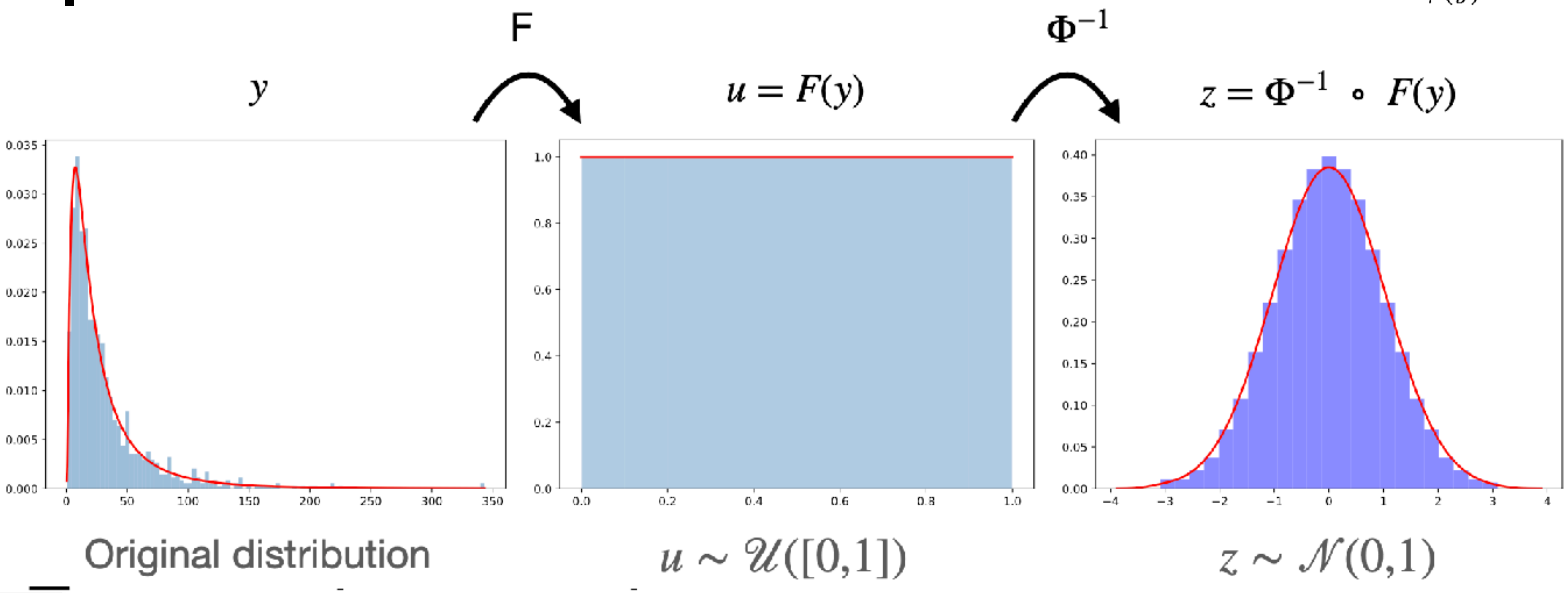
🤔 What is the distribution of $\Phi^{-1} \circ F(y)$?

💡 Normal $\mathcal{N}(0,1)$!

# Gaussian Copula Transform

## Nice properties




🤔 What is the distribution of $F(y)$?

💡 Uniform $\mathcal{U}([0,1])$!

🤔 What is the distribution of $\Phi^{-1} \circ F(y)$?

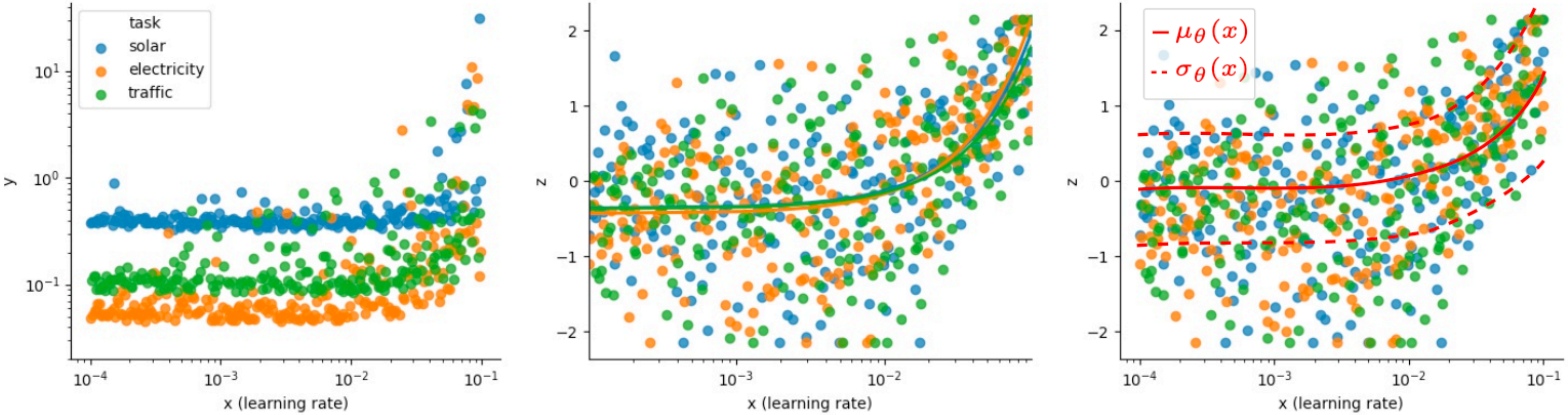
💡 Normal $\mathcal{N}(0,1)$!

Two very nice properties of $\Psi = \Phi^{-1} \circ F$:

- 😀 $z^j = \Psi(y^j) \sim \mathcal{N}(0,1)$ (great for GP)
- 😀 Built-in invariance of $z^j$ to monotonic transformation on $y^j$

# Gaussian Copula Transform

## Learning joint representations across tasks




**Left:** Plot blackbox error $y$ in log-space against a single hyperparameter $x$ for different tasks.
**Middle:** Running mean after transforming each task objectives with $z = \psi(y) = \Phi^{-1} \circ F(y)$.
**Right:** Illustrative plot of possible mean/variance fit of a model $\mu_\theta(x), \sigma_\theta(x)$ trained jointly on all tasks with shared parameters $\theta$.

# Learning parametric priors for HPO

## Gaussian Copula Process with Parametric Prior (GC3P)

# Learning parametric priors for HPO

## Gaussian Copula Process with Parametric Prior (GC3P)

- Outline:

# Learning parametric priors for HPO

## Gaussian Copula Process with Parametric Prior (GC3P)

- Outline:
    - Transform z = ψ(y) where $\Psi = \Phi^{-1} \circ F$ on every task

# Learning parametric priors for HPO

## Gaussian Copula Process with Parametric Prior (GC3P)

- Outline:
    - Transform z = ψ(y) where $\Psi = \Phi^{-1} \circ F$ on every task
    - Learn a parametric prior with a MLP that predicts $z \mid x \approx \mathcal{N}(\mu_\theta(x), \sigma^2_\theta(x))$

# Learning parametric priors for HPO

## Gaussian Copula Process with Parametric Prior (GC3P)

- Outline:
  - Transform z = ψ(y) where $\Psi = \Phi^{-1} \circ F$ on every task
  - Learn a parametric prior with a MLP that predicts $z \mid x \approx \mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$
  - Strategy 1: Thompson sampling with predictive distribution $z \mid x \approx \mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$

# Learning parametric priors for HPO

## Gaussian Copula Process with Parametric Prior (GC3P)

- Outline:
  - Transform z = ψ(y) where $\Psi = \Phi^{-1} \circ$ F on every task
  - Learn a parametric prior with a MLP that predicts $z \mid x \approx \mathcal{N}(\mu_\theta(x), \sigma^2_\theta(x))$
  - Strategy 1: Thompson sampling with predictive distribution $z \mid x \approx \mathcal{N}(\mu_\theta(x), \sigma^2_\theta(x))$
  - Strategy 2: Optimise with Gaussian Copula Process using $\mathcal{N}(\mu_\theta(x), \sigma^2_\theta(x))$ as a prior

# Learning parametric priors for HPO

## Gaussian Copula Process with Parametric Prior (GC3P)

- Outline:
  - Transform z = ψ(y) where $\Psi = \Phi^{-1} \circ$ F on every task
  - Learn a parametric prior with a MLP that predicts $z \mid x \approx \mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$
  - Strategy 1: Thompson sampling with predictive distribution $z \mid x \approx \mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$
  - Strategy 2: Optimise with Gaussian Copula Process using $\mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$ as a prior
- Benefits:

# Learning parametric priors for HPO

## Gaussian Copula Process with Parametric Prior (GC3P)

- Outline:
  - Transform z = ψ(y) where $\Psi = \Phi^{-1} \circ F$ on every task
  - Learn a parametric prior with a MLP that predicts $z \mid x \approx \mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$
  - Strategy 1: Thompson sampling with predictive distribution $z \mid x \approx \mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$
  - Strategy 2: Optimise with Gaussian Copula Process using $\mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$ as a prior
- Benefits:
  - Amplitude, noise issues: addressed by using ψ

# Learning parametric priors for HPO

## Gaussian Copula Process with Parametric Prior (GC3P)

- Outline:
  - Transform z = ψ(y) where $\Psi = \Phi^{-1} \circ$ F on every task
  - Learn a parametric prior with a MLP that predicts $z \mid x \approx \mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$
  - Strategy 1: Thompson sampling with predictive distribution $z \mid x \approx \mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$
  - Strategy 2: Optimise with Gaussian Copula Process using $\mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$ as a prior
- Benefits:
  - Amplitude, noise issues: addressed by using ψ
  - Scale with many offline evaluations: inference done with a single pass over an MLP

# Learning parametric priors for HPO

## Gaussian Copula Process with Parametric Prior (GC3P)

- Outline:
  - Transform z = ψ(y) where $\Psi = \Phi^{-1} \circ F$ on every task
  - Learn a parametric prior with a MLP that predicts $z \mid x \approx \mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$
  - Strategy 1: Thompson sampling with predictive distribution $z \mid x \approx \mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$
  - Strategy 2: Optimise with Gaussian Copula Process using $\mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$ as a prior
- Benefits:
  - Amplitude, noise issues: addressed by using ψ
  - Scale with many offline evaluations: inference done with a single pass over an MLP
  - Negative transfer: alleviated with a Gaussian Copula Process

# Gaussian Copula Process with Parametric Prior (GC3P)

**High-level idea: A Gaussian Copula Process whose prior is** $\mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$

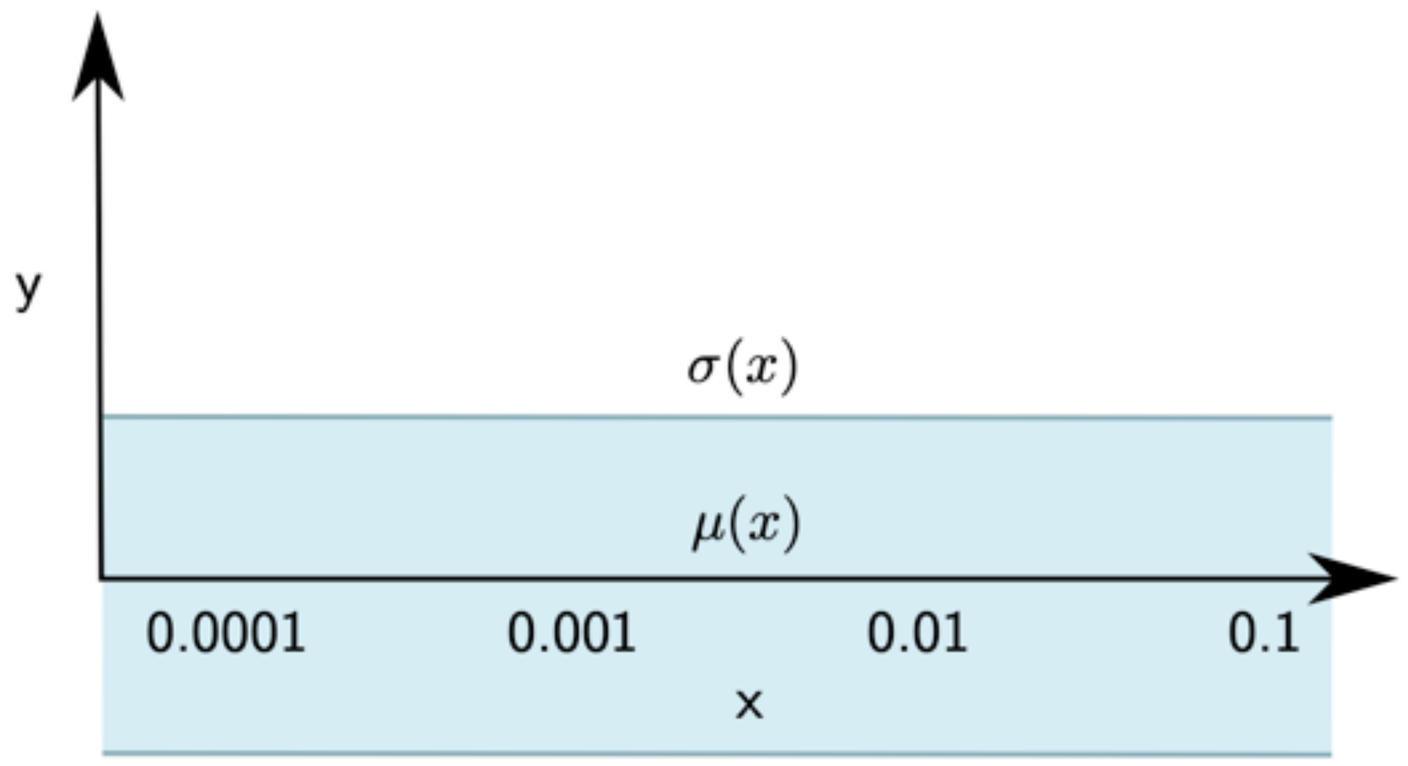
# Gaussian Copula Process with Parametric Prior (GC3P)

**High-level idea: A Gaussian Copula Process whose prior is** $\mathcal{N}(\mu_\theta(x), \sigma^2_\theta(x))$

Standard
Gaussian Process

# Gaussian Copula Process with Parametric Prior (GC3P)

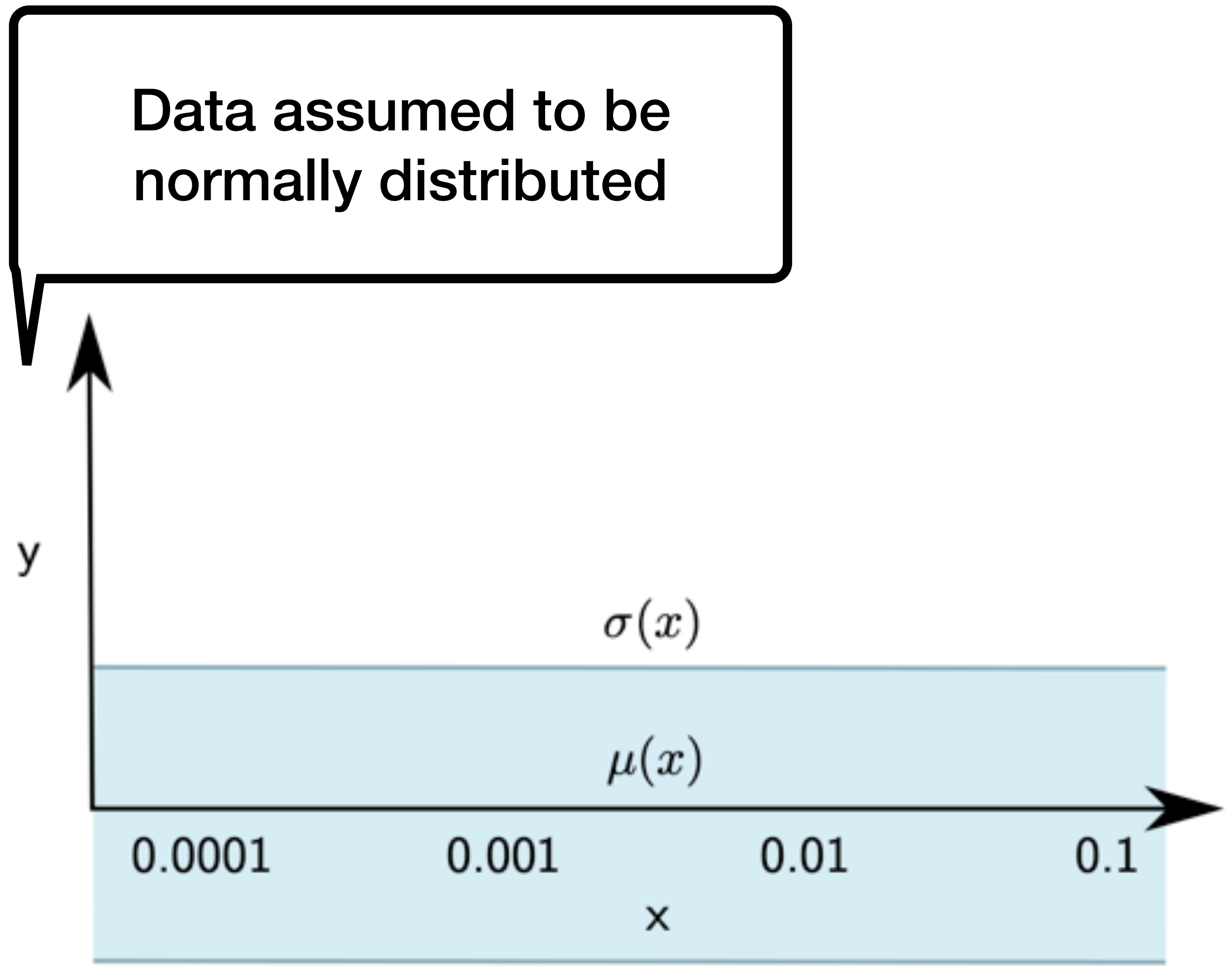
**High-level idea: A Gaussian Copula Process whose prior is $\mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$**




Standard
Gaussian Process

# Gaussian Copula Process with Parametric Prior (GC3P)

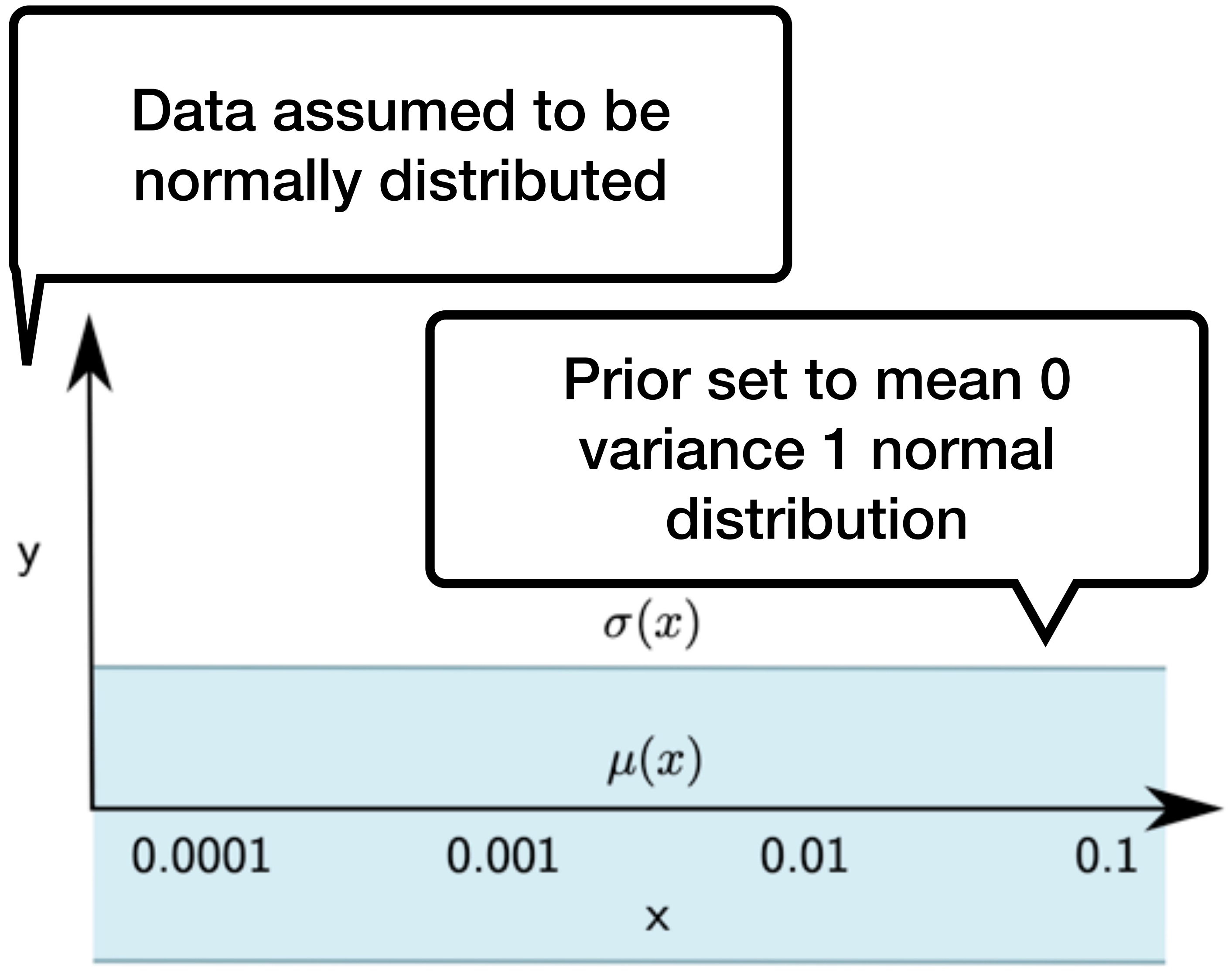
**High-level idea: A Gaussian Copula Process whose prior is** $\mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$




Standard Gaussian Process

# Gaussian Copula Process with Parametric Prior (GC3P)

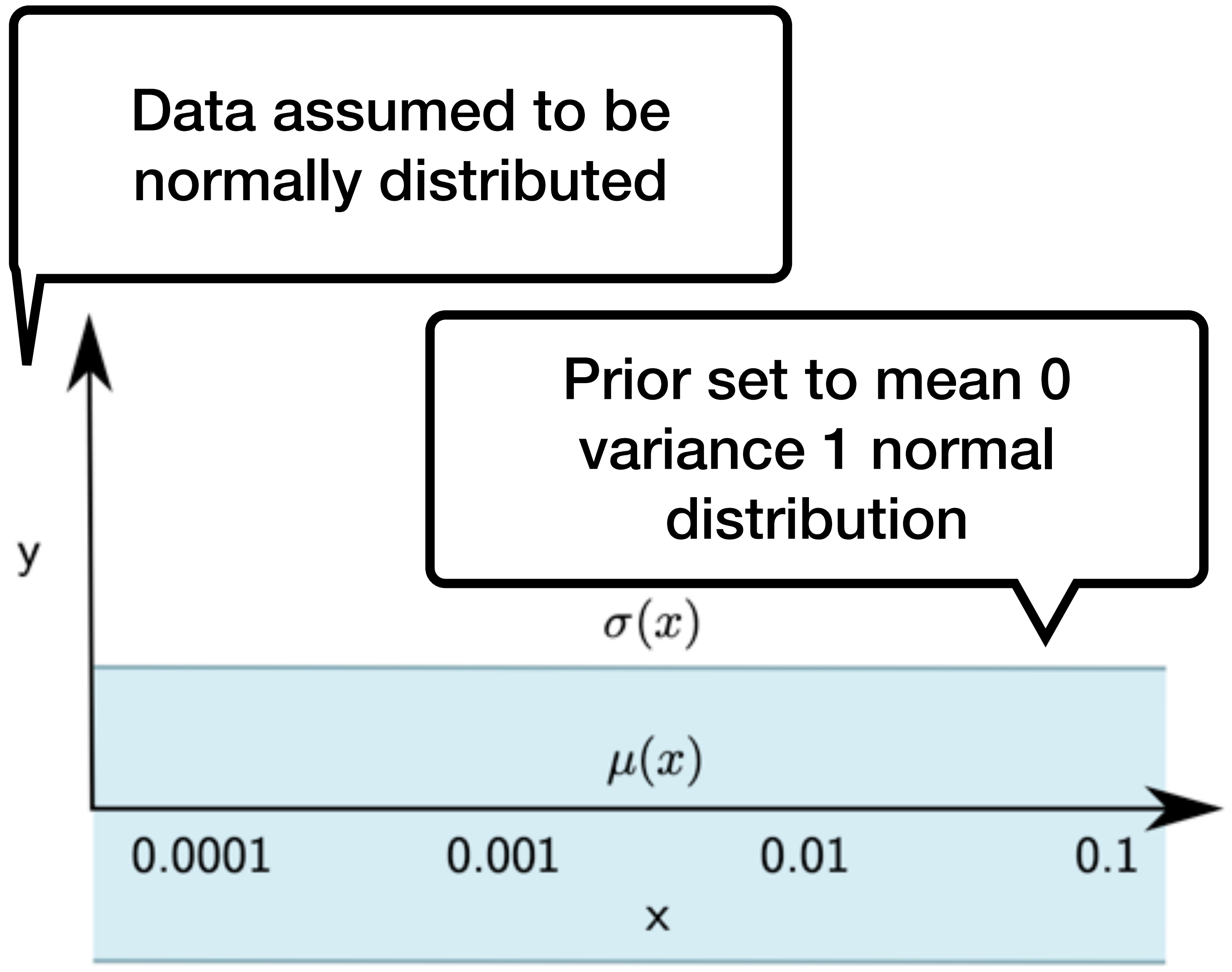
**High-level idea: A Gaussian Copula Process whose prior is $\mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$**



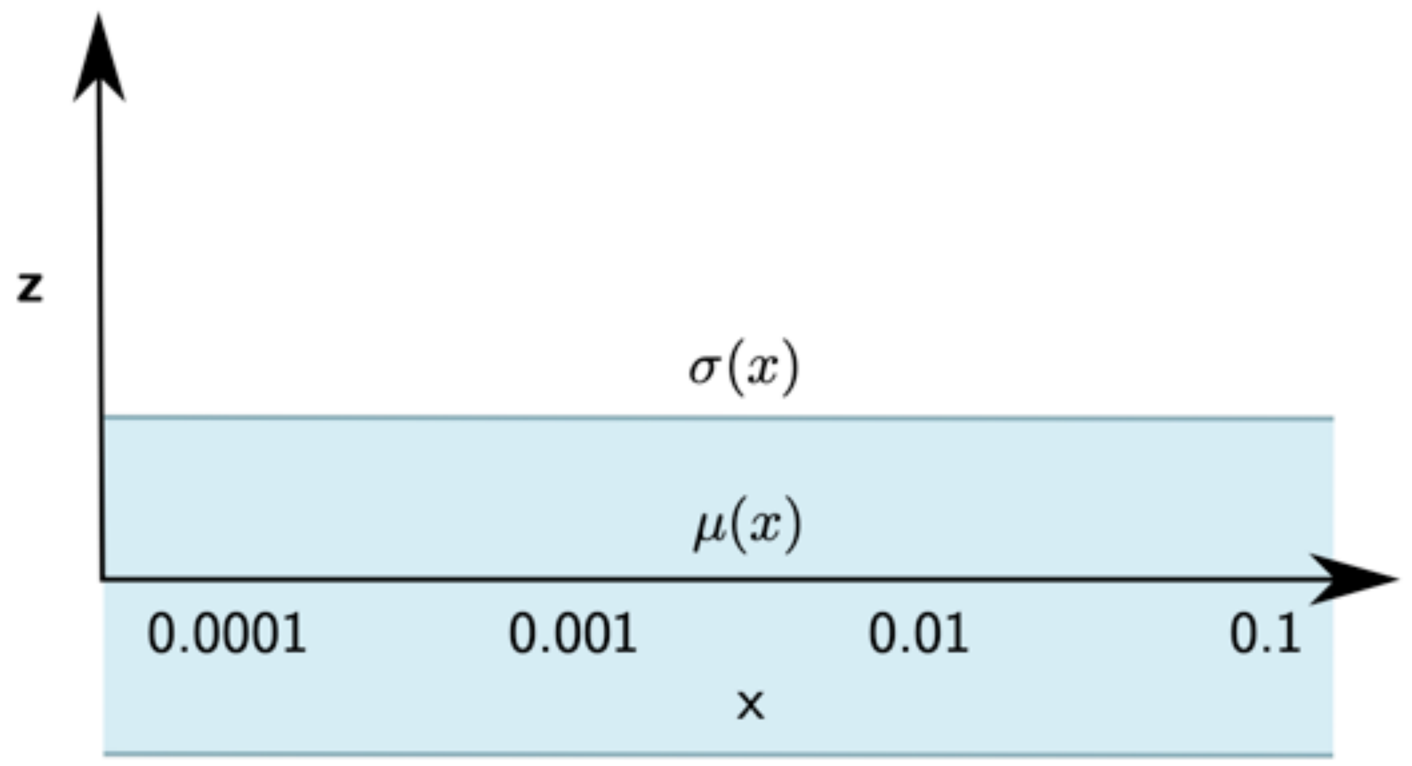

Standard Gaussian Process

# Gaussian Copula Process with Parametric Prior (GC3P)

**High-level idea: A Gaussian Copula Process whose prior is** $\mathcal{N}(\mu_\theta(x), \sigma^2_\theta(x))$



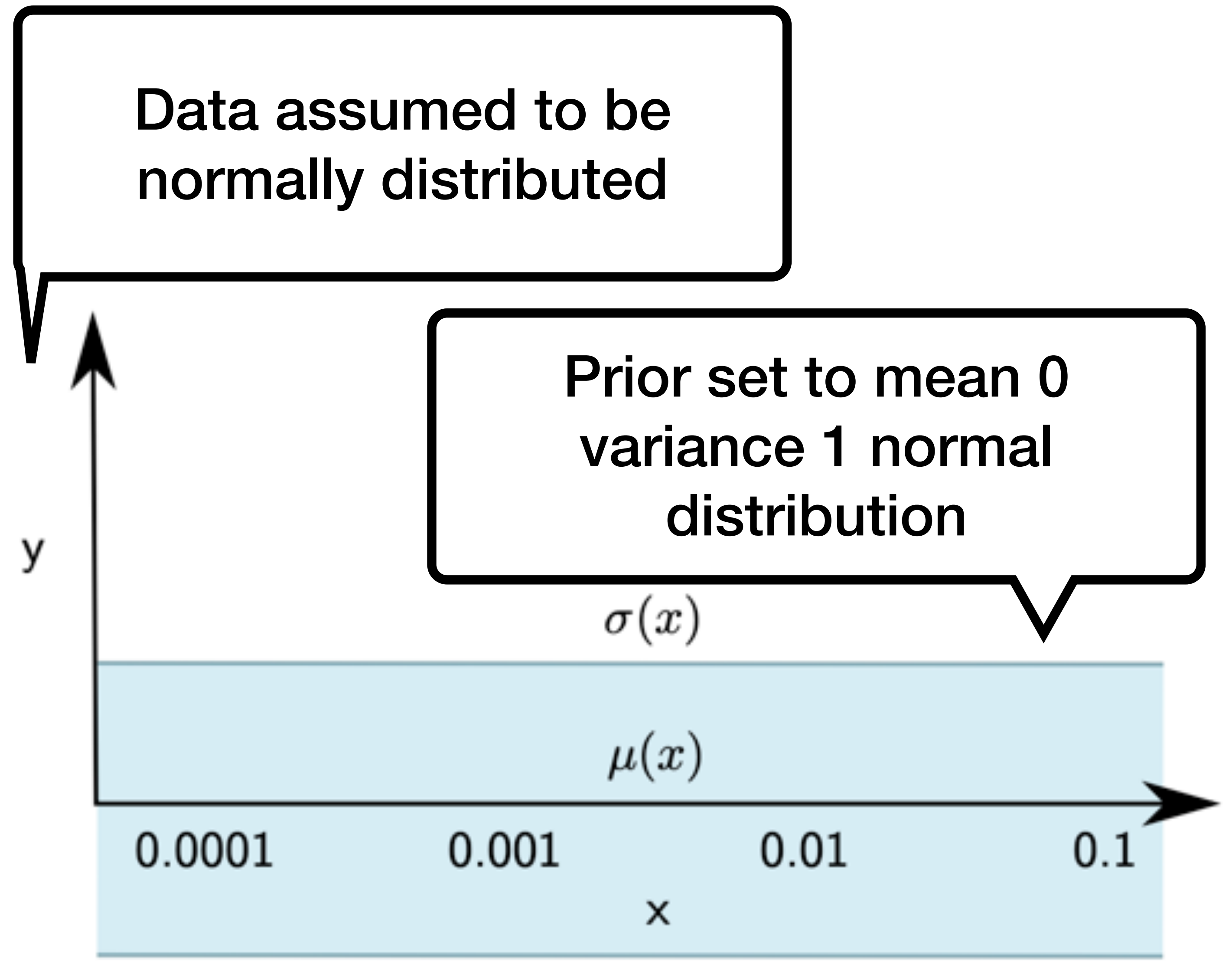

Standard Gaussian Process



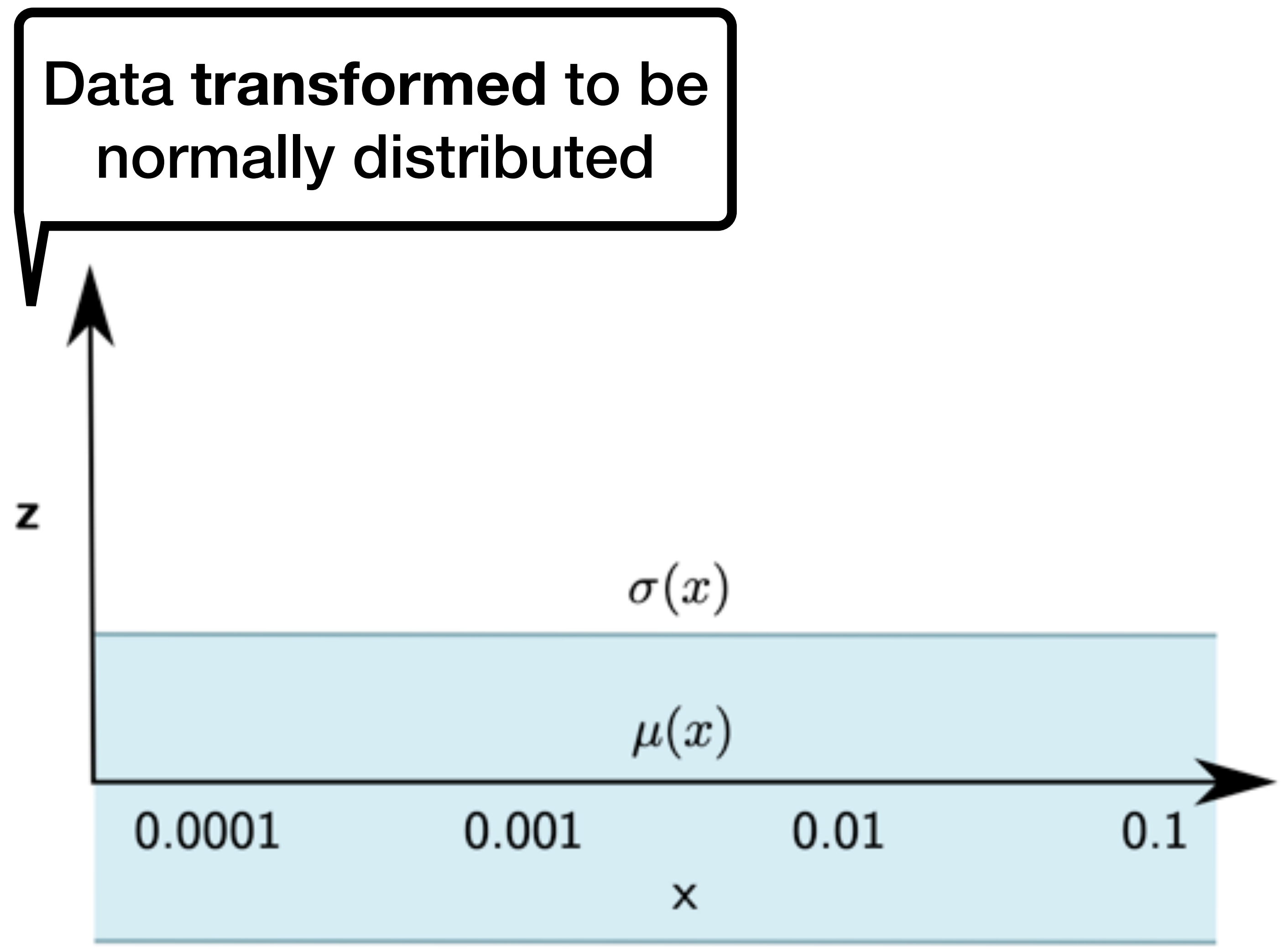

Gaussian Copula Process

# Gaussian Copula Process with Parametric Prior (GC3P)

**High-level idea: A Gaussian Copula Process whose prior is** $\mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$



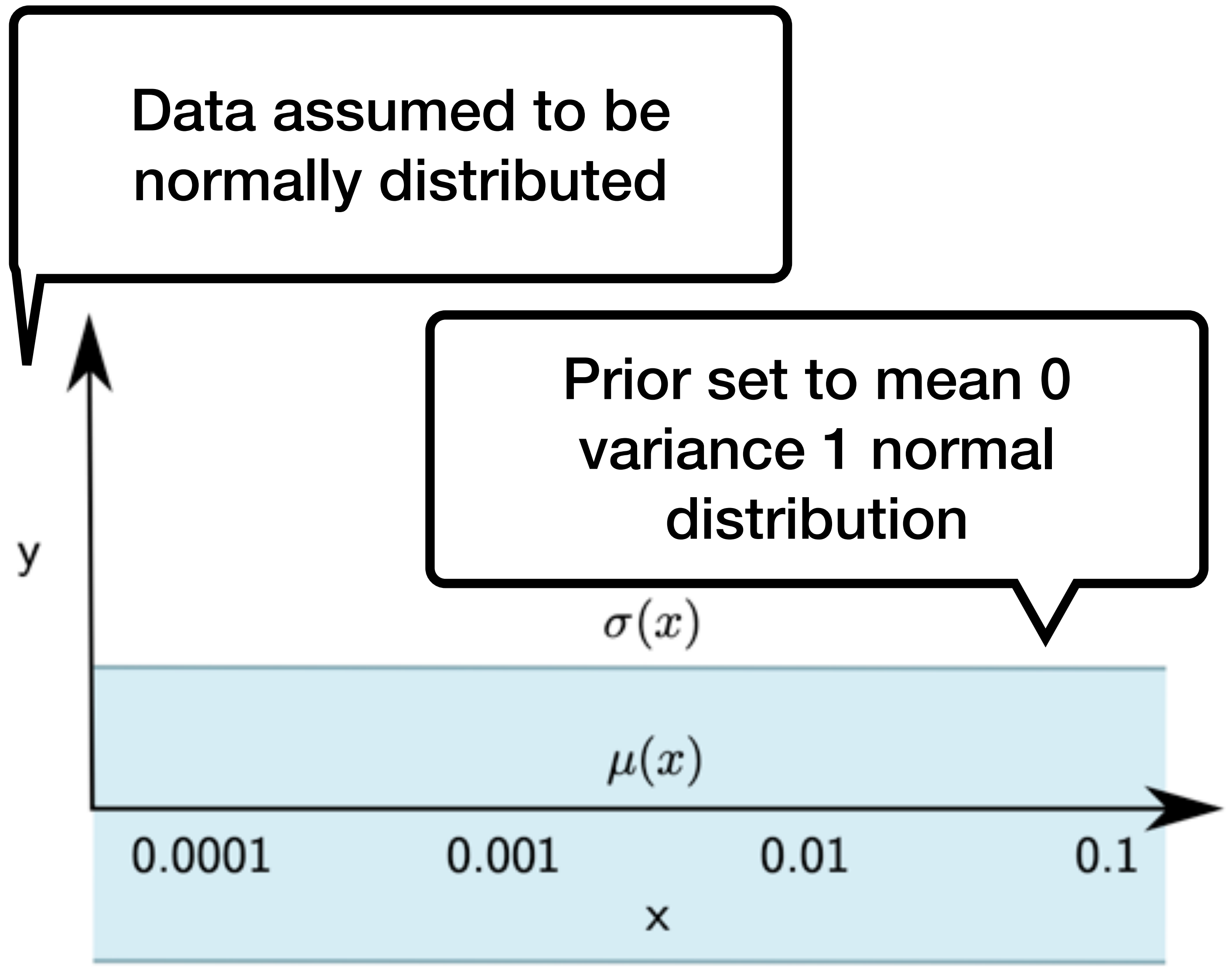

Standard Gaussian Process



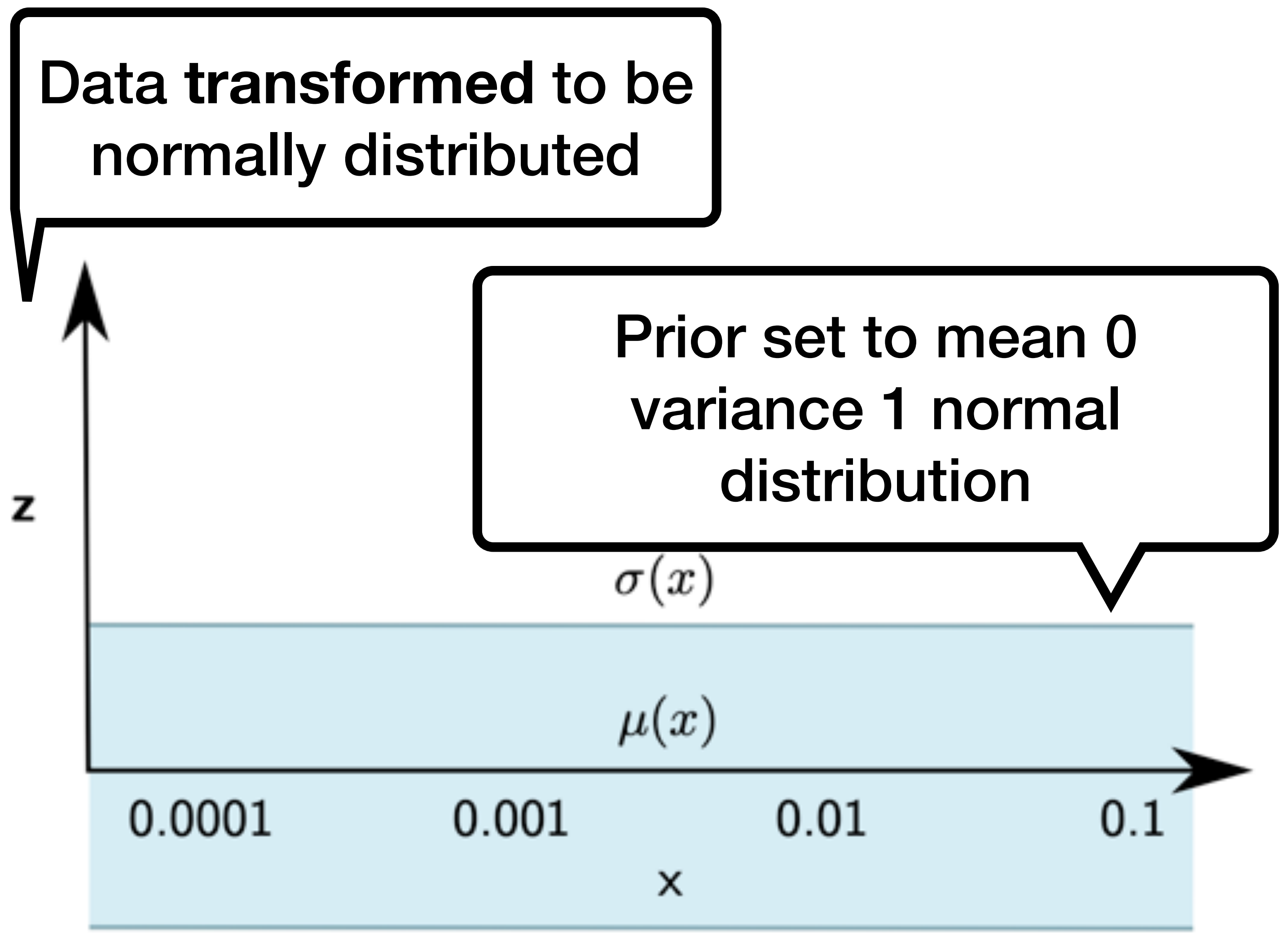

Gaussian Copula Process

# Gaussian Copula Process with Parametric Prior (GC3P)

**High-level idea: A Gaussian Copula Process whose prior is $\mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$**



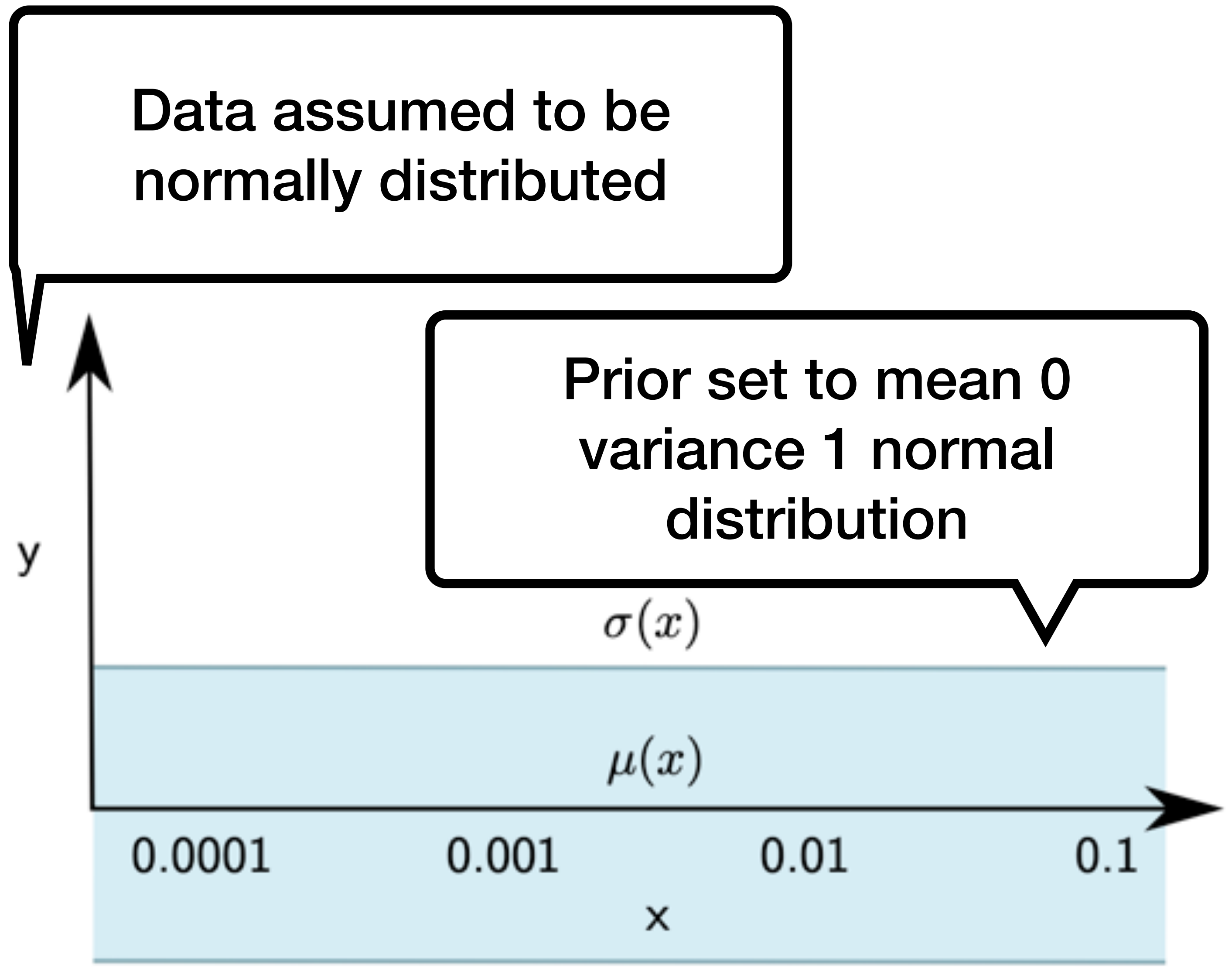

Standard Gaussian Process



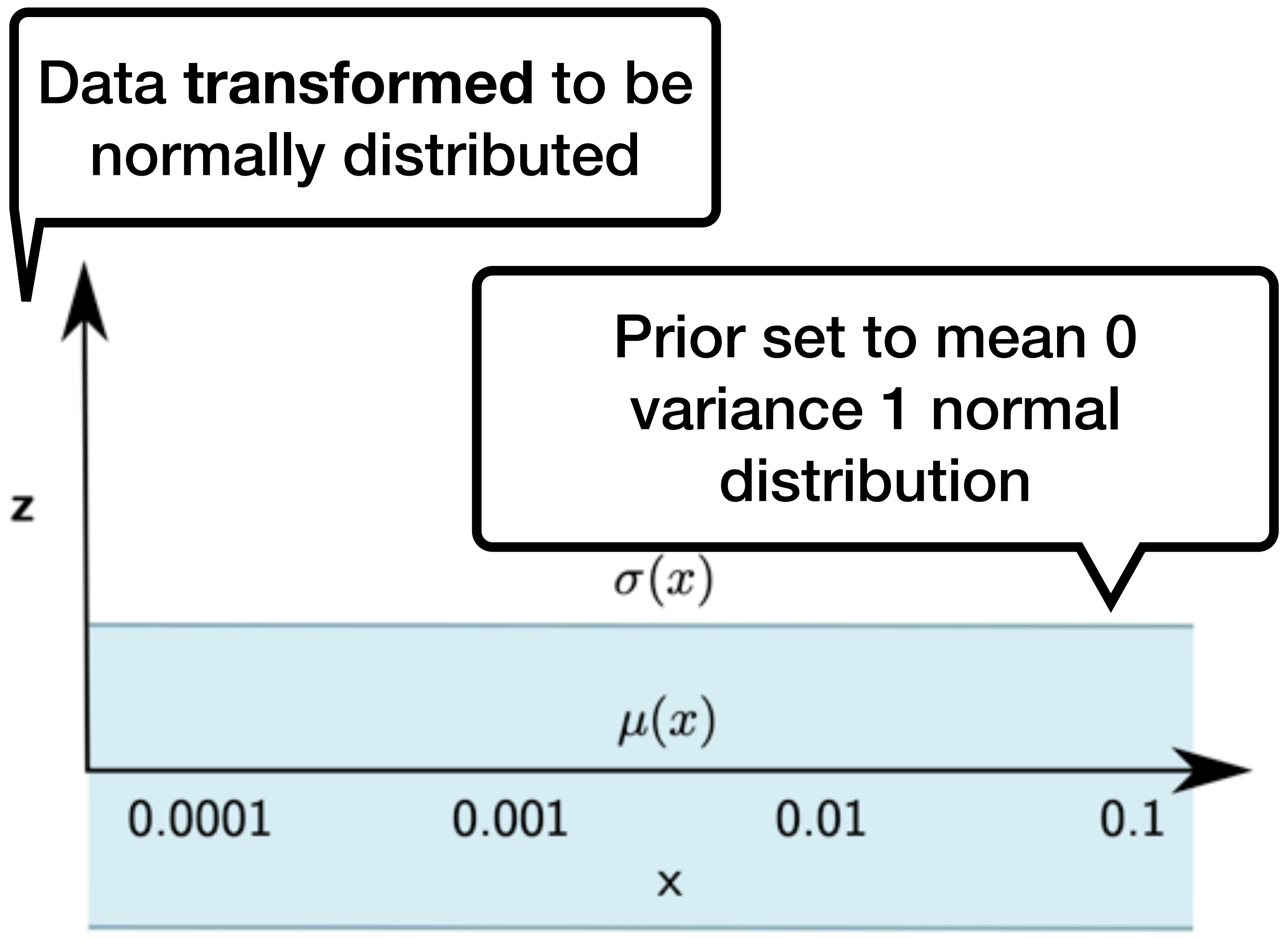

Gaussian Copula Process

# Gaussian Copula Process with Parametric Prior (GC3P)

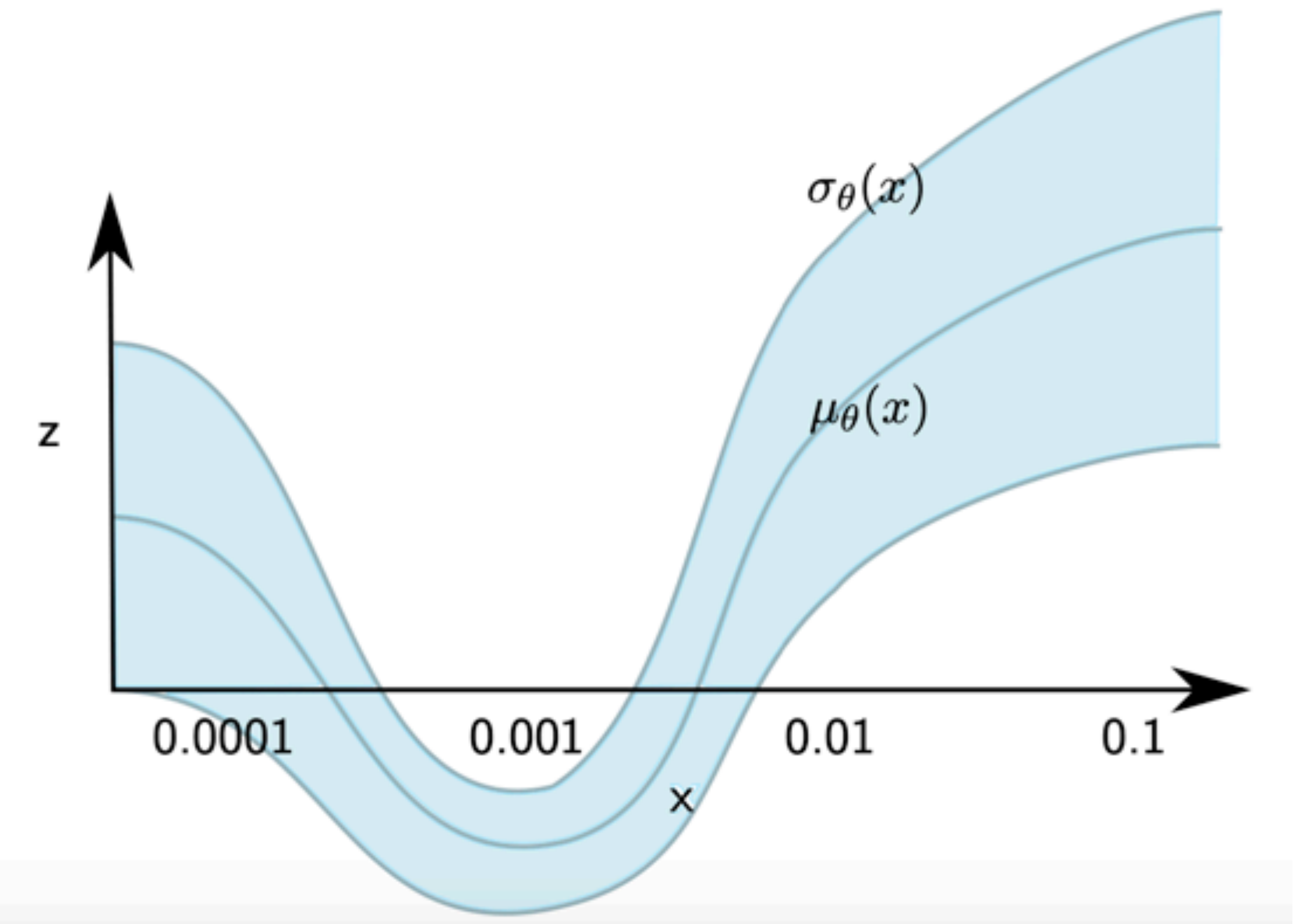
**High-level idea: A Gaussian Copula Process whose prior is $\mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$**




Standard Gaussian Process



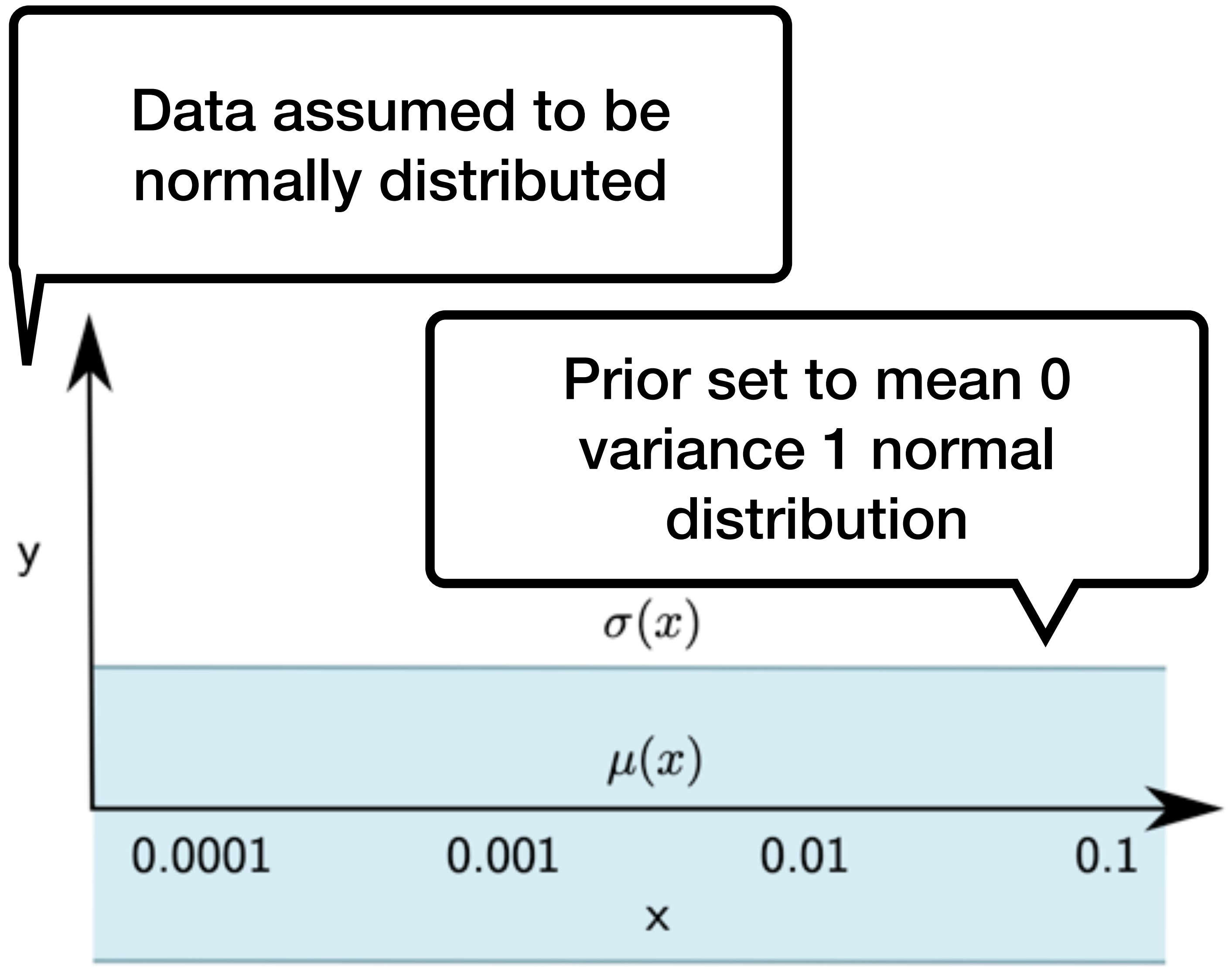

Gaussian Copula Process



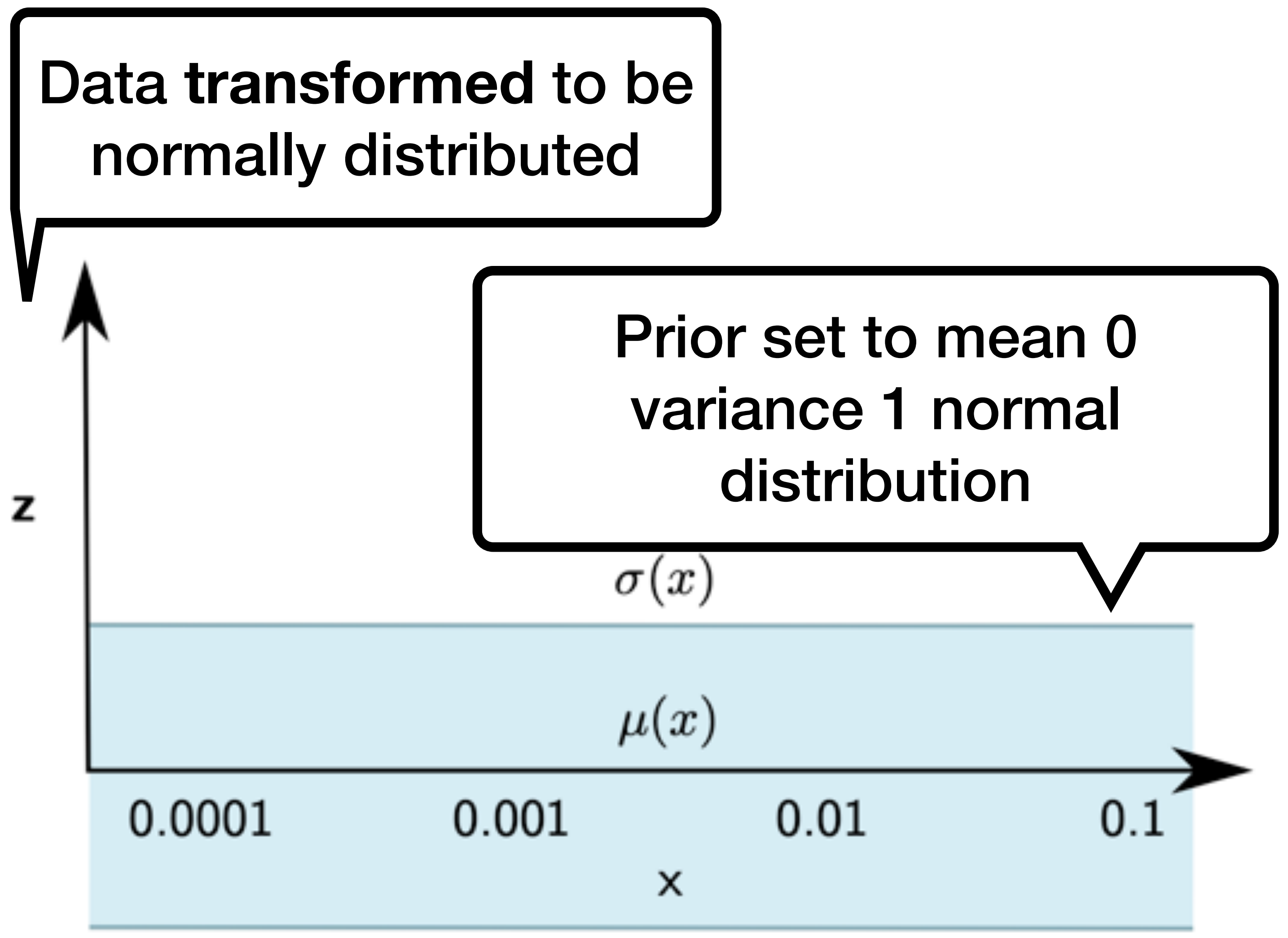

Gaussian Copula Process whose prior is $\mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$

# Gaussian Copula Process with Parametric Prior (GC3P)

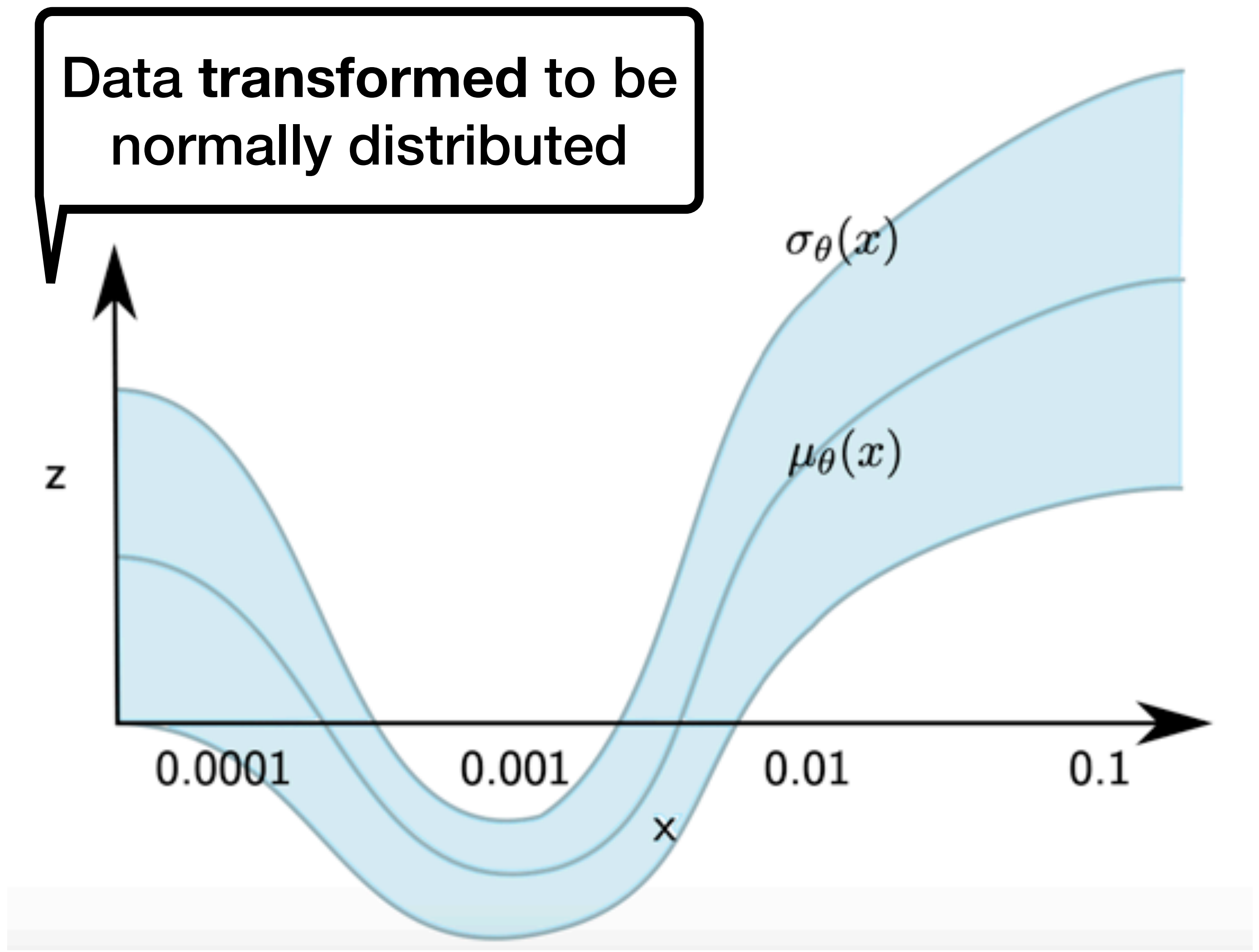
**High-level idea: A Gaussian Copula Process whose prior is $\mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$**




Standard Gaussian Process



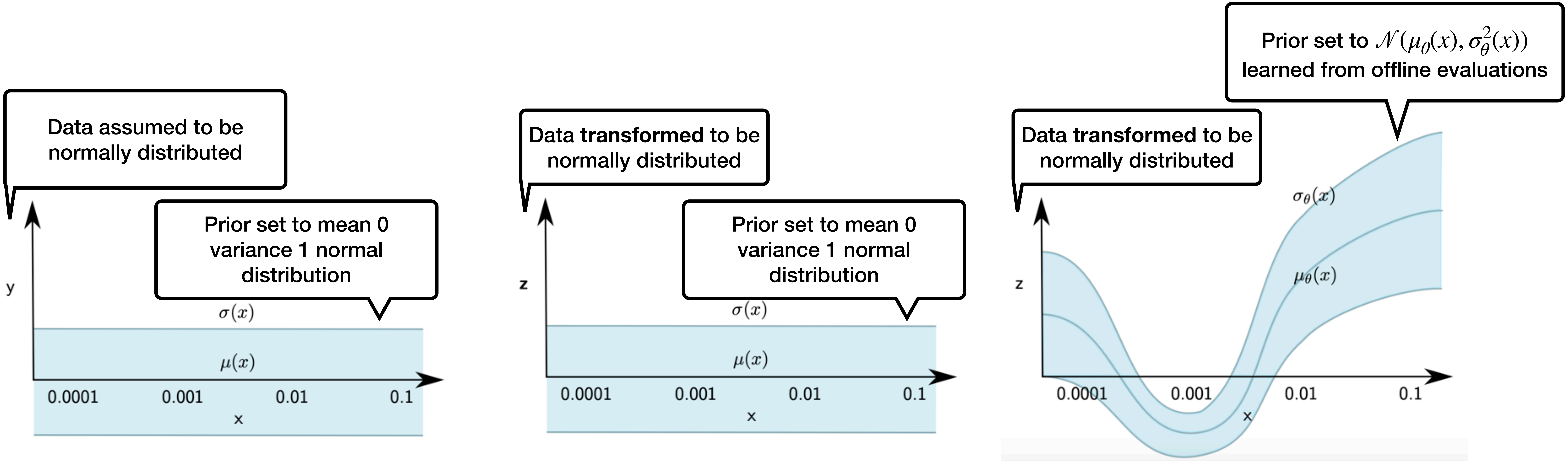

Gaussian Copula Process




Gaussian Copula Process whose prior is $\mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$

# Gaussian Copula Process with Parametric Prior (GC3P)

**High-level idea: A Gaussian Copula Process whose prior is $\mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$**




Standard Gaussian Process

Gaussian Copula Process

Gaussian Copula Process whose prior is $\mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$

# Evaluations

# Evaluations

- Evaluations on 4 blackboxes with precomputed evaluations

# Evaluations

- Evaluations on 4 blackboxes with precomputed evaluations
- Leave-one-out setting:

# Evaluations

- Evaluations on 4 blackboxes with precomputed evaluations
- Leave-one-out setting:
    - use all tasks but one for transfer learning methods

# Evaluations

- Evaluations on 4 blackboxes with precomputed evaluations
- Leave-one-out setting:
  - use all tasks but one for transfer learning methods
  - evaluate performance on hold-out task

# Evaluations

- Evaluations on 4 blackboxes with precomputed evaluations
- Leave-one-out setting:
    - use all tasks but one for transfer learning methods
    - evaluate performance on hold-out task
- Report average distance to the minimum (ADTM) averaged over 30 seeds

# Evaluations

- Evaluations on 4 blackboxes with precomputed evaluations
- Leave-one-out setting:
  - use all tasks but one for transfer learning methods
  - evaluate performance on hold-out task
- Report average distance to the minimum (ADTM) averaged over 30 seeds

| blackbox | # tasks | # hyperparameters | # evaluations/task | objective |
|---|---|---|---|---|
| DeepAR | 11 | 6 | $\sim$ 220 | quantile loss |
| FCNET | 4 | 9 | 62208 | MSE |
| XGBoost | 9 | 9 | 5000 | 1-AUC |
| NAS | 3 | 6 | 46875 | accuracy |

# Evaluations

Table: Average method rank, best two methods are in bold.

| | DeepAR | FCNET | XGBoost | NAS |
|---|---|---|---|---|
| RS | 7.1 | 10.8 | 8.2 | 11.7 |
| GP | 7.9 | 8.0 | 8.4 | 9.3 |
| GCP | | | | |
| AutoGP | | | | |
| WS GP | | | | |
| ABLR | | | | |
| SGPT | | | | |
| BOHB | | | | |
| R-EA | | | | |
| REINFORCE | | | | |
| CTS (ours) | | | | |
| GCP + prior (ours) | | | | |
| TS (w/o Copula) | | | | |
| GP + prior (w/o Copula) | | | | |
| ABLR + Copula | | | | |
| SGPT + Copula | | | | |

# Evaluations

Table: Average method rank, best two methods are in bold.

| | DeepAR | FCNET | XGBoost | NAS |
|---|---|---|---|---|
| RS | 7.1 | 10.8 | 8.2 | 11.7 |
| GP | 7.9 | 8.0 | 8.4 | 9.3 |
| GCP | | | | |
| AutoGP | | | | |
| WS GP | | | | |
| ABLR | | | | |
| SGPT | | | | |
| BOHB | | | | |
| R-EA | | | | |
| REINFORCE | | | | |
| CTS (ours) | | | | |
| GCP + prior (ours) | | | | |
| TS (w/o Copula) | | | | |
| GP + prior (w/o Copula) | | | | |
| ABLR + Copula | | | | |
| SGPT + Copula | | | | |

GP > RS as the method can exploit

# Evaluations

Table: Average method rank, best two methods are in bold.

| | DeepAR | FCNET | XGBoost | NAS |
|---|---|---|---|---|
| RS | 7.1 | 10.8 | 8.2 | 11.7 |
| GP | 7.9 | 8.0 | 8.4 | 9.3 |
| GCP | 4.3 | 3.8 | **3.1** | 7.7 |
| AutoGP | | | | |
| WS GP | | | | |
| ABLR | | | | |
| SGPT | | | | |
| BOHB | | | | |
| R-EA | | | | |
| REINFORCE | | | | |
| CTS (ours) | | | | |
| GCP + prior (ours) | | | | |
| TS (w/o Copula) | | | | |
| GP + prior (w/o Copula) | | | | |
| ABLR + Copula | | | | |
| SGPT + Copula | | | | |

GP > RS as the method can exploit

# Evaluations

Table: Average method rank, best two methods are in bold.

| | DeepAR | FCNET | XGBoost | NAS |
|---|---|---|---|---|
| RS | 7.1 | 10.8 | 8.2 | 11.7 |
| GP | 7.9 | 8.0 | 8.4 | 9.3 |
| GCP | 4.3 | 3.8 | **3.1** | 7.7 |
| AutoGP | | | | |
| WS GP | | | | |
| ABLR | | | | |
| SGPT | | | | |
| BOHB | | | | |
| R-EA | | | | |
| REINFORCE | | | | |
| CTS (ours) | | | | |
| GCP + prior (ours) | | | | |
| TS (w/o Copula) | | | | |
| GP + prior (w/o Copula) | | | | |
| ABLR + Copula | | | | |
| SGPT + Copula | | | | |

GP > RS as the method can exploit

GCP > GP as we made less restriction on the noise

# Evaluations

Table: Average method rank, best two methods are in bold.

| | DeepAR | FCNET | XGBoost | NAS |
|---|---|---|---|---|
| RS | 7.1 | 10.8 | 8.2 | 11.7 |
| GP | 7.9 | 8.0 | 8.4 | 9.3 |
| GCP | 4.3 | 3.8 | **3.1** | 7.7 |
| AutoGP | 7.3 | 5.5 | 4.2 | 2.7 |
| WS GP | 7.6 | 5.2 | 5.9 | 6.0 |
| ABLR | 10.2 | 10.2 | 9.1 | 10.3 |
| SGPT | 8.8 | 8.2 | 8.6 | 7.3 |
| BOHB | - | - | - | 14.3 |
| R-EA | - | - | - | 10.0 |
| REINFORCE | - | - | - | 13.0 |
| CTS (ours) | | | | |
| GCP + prior (ours) | | | | |
| TS (w/o Copula) | | | | |
| GP + prior (w/o Copula) | | | | |
| ABLR + Copula | | | | |
| SGPT + Copula | | | | |

GP > RS as the method can exploit

GCP > GP as we made less restriction on the noise

# Evaluations

Table: Average method rank, best two methods are in bold.

| | DeepAR | FCNET | XGBoost | NAS |
|---|---|---|---|---|
| RS | 7.1 | 10.8 | 8.2 | 11.7 |
| GP | 7.9 | 8.0 | 8.4 | 9.3 |
| GCP | 4.3 | 3.8 | **3.1** | 7.7 |
| AutoGP | 7.3 | 5.5 | 4.2 | 2.7 |
| WS GP | 7.6 | 5.2 | 5.9 | 6.0 |
| ABLR | 10.2 | 10.2 | 9.1 | 10.3 |
| SGPT | 8.8 | 8.2 | 8.6 | 7.3 |
| BOHB | - | - | - | 14.3 |
| R-EA | - | - | - | 10.0 |
| REINFORCE | - | - | - | 13.0 |
| CTS (ours) | | | | |
| GCP + prior (ours) | | | | |
| TS (w/o Copula) | | | | |
| GP + prior (w/o Copula) | | | | |
| ABLR + Copula | | | | |
| SGPT + Copula | | | | |

GP > RS as the method can exploit

GCP > GP as we made less restriction on the noise

Transfer learning generally improve performance

# Evaluations

Table: Average method rank, best two methods are in bold.

| | DeepAR | FCNET | XGBoost | NAS |
|---|---|---|---|---|
| RS | 7.1 | 10.8 | 8.2 | 11.7 |
| GP | 7.9 | 8.0 | 8.4 | 9.3 |
| GCP | 4.3 | 3.8 | **3.1** | 7.7 |
| AutoGP | 7.3 | 5.5 | 4.2 | 2.7 |
| WS GP | 7.6 | 5.2 | 5.9 | 6.0 |
| ABLR | 10.2 | 10.2 | 9.1 | 10.3 |
| SGPT | 8.8 | 8.2 | 8.6 | 7.3 |
| BOHB | - | - | - | 14.3 |
| R-EA | - | - | - | 10.0 |
| REINFORCE | - | - | - | 13.0 |
| CTS (ours) | 4.5 | **2.5** | 7.6 | **2.7** |
| GCP + prior (ours) | **1.7** | **1.0** | **1.9** | **1.3** |
| TS (w/o Copula) | 13.0 | 13.0 | 12.7 | 14.7 |
| GP + prior (w/o Copula) | 11.8 | 12.0 | 11.0 | 15.3 |
| ABLR + Copula | **3.1** | 5.5 | 7.0 | 5.7 |
| SGPT + Copula | 3.7 | 5.2 | 3.3 | 4.0 |

GP > RS as the method can exploit

GCP > GP as we made less restriction on the noise

Transfer learning generally improve performance

# Evaluations

Table: Average method rank, best two methods are in bold.

| | DeepAR | FCNET | XGBoost | NAS |
|---|---|---|---|---|
| RS | 7.1 | 10.8 | 8.2 | 11.7 |
| GP | 7.9 | 8.0 | 8.4 | 9.3 |
| GCP | 4.3 | 3.8 | **3.1** | 7.7 |
| AutoGP | 7.3 | 5.5 | 4.2 | 2.7 |
| WS GP | 7.6 | 5.2 | 5.9 | 6.0 |
| ABLR | 10.2 | 10.2 | 9.1 | 10.3 |
| SGPT | 8.8 | 8.2 | 8.6 | 7.3 |
| BOHB | - | - | - | 14.3 |
| R-EA | - | - | - | 10.0 |
| REINFORCE | - | - | - | 13.0 |
| CTS (ours) | 4.5 | **2.5** | 7.6 | **2.7** |
| GCP + prior (ours) | **1.7** | **1.0** | **1.9** | **1.3** |
| TS (w/o Copula) | 13.0 | 13.0 | 12.7 | 14.7 |
| GP + prior (w/o Copula) | 11.8 | 12.0 | 11.0 | 15.3 |
| ABLR + Copula | **3.1** | 5.5 | 7.0 | 5.7 |
| SGPT + Copula | 3.7 | 5.2 | 3.3 | 4.0 |

GP > RS as the method can exploit

GCP > GP as we made less restriction on the noise

Transfer learning generally improve performance

Parametric prior improves over baseline significantly

# Evaluations

Table: Average method rank, best two methods are in bold.

| | DeepAR | FCNET | XGBoost | NAS |
|---|---|---|---|---|
| RS | 7.1 | 10.8 | 8.2 | 11.7 |
| GP | 7.9 | 8.0 | 8.4 | 9.3 |
| GCP | 4.3 | 3.8 | **3.1** | 7.7 |
| AutoGP | 7.3 | 5.5 | 4.2 | 2.7 |
| WS GP | 7.6 | 5.2 | 5.9 | 6.0 |
| ABLR | 10.2 | 10.2 | 9.1 | 10.3 |
| SGPT | 8.8 | 8.2 | 8.6 | 7.3 |
| BOHB | - | - | - | 14.3 |
| R-EA | - | - | - | 10.0 |
| REINFORCE | - | - | - | 13.0 |
| CTS (ours) | 4.5 | **2.5** | 7.6 | **2.7** |
| GCP + prior (ours) | **1.7** | **1.0** | **1.9** | **1.3** |
| TS (w/o Copula) | 13.0 | 13.0 | 12.7 | 14.7 |
| GP + prior (w/o Copula) | 11.8 | 12.0 | 11.0 | 15.3 |
| ABLR + Copula | **3.1** | 5.5 | 7.0 | 5.7 |
| SGPT + Copula | 3.7 | 5.2 | 3.3 | 4.0 |

GP > RS as the method can exploit

GCP > GP as we made less restriction on the noise

Transfer learning generally improve performance

Parametric prior  improves over baseline significantly

Only if we use the right transformation…

# Evaluations

Table: Average method rank, best two methods are in bold.

| | DeepAR | FCNET | XGBoost | NAS |
|---|---|---|---|---|
| RS | 7.1 | 10.8 | 8.2 | 11.7 |
| GP | 7.9 | 8.0 | 8.4 | 9.3 |
| GCP | 4.3 | 3.8 | **3.1** | 7.7 |
| AutoGP | 7.3 | 5.5 | 4.2 | 2.7 |
| WS GP | 7.6 | 5.2 | 5.9 | 6.0 |
| ABLR | 10.2 | 10.2 | 9.1 | 10.3 |
| SGPT | 8.8 | 8.2 | 8.6 | 7.3 |
| BOHB | - | - | - | 14.3 |
| R-EA | - | - | - | 10.0 |
| REINFORCE | - | - | - | 13.0 |
| CTS (ours) | 4.5 | **2.5** | 7.6 | **2.7** |
| GCP + prior (ours) | **1.7** | **1.0** | **1.9** | **1.3** |
| TS (w/o Copula) | 13.0 | 13.0 | 12.7 | 14.7 |
| GP + prior (w/o Copula) | 11.8 | 12.0 | 11.0 | 15.3 |
| ABLR + Copula | **3.1** | 5.5 | 7.0 | 5.7 |
| SGPT + Copula | 3.7 | 5.2 | 3.3 | 4.0 |

GP > RS as the method can exploit

GCP > GP as we made less restriction on the noise

Transfer learning generally improve performance

Parametric prior  improves over baseline significantly

Only if we use the right transformation…

Using this transformation also improves baselines a lot

# Robustness to negative transfer

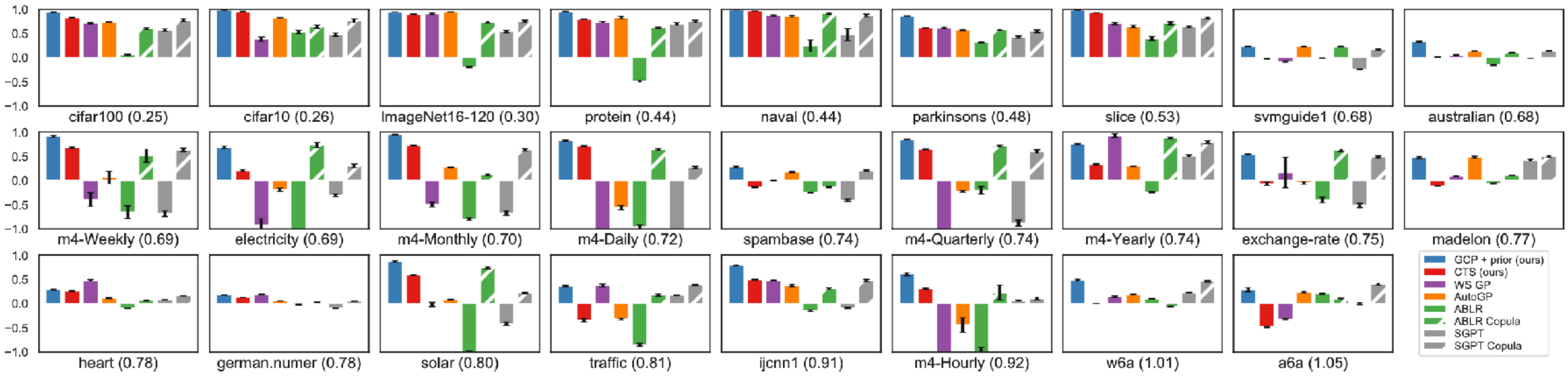
# Robustness to negative transfer

- If a new task differ from previous evaluations, we would still want to get reasonable performance!

# Robustness to negative transfer

- If a new task differ from previous evaluations, we would still want to get reasonable performance!
- Improvement over random search (↑), tasks sorted by transfer learning difficulty (RMSE of prior predictor on the new task)

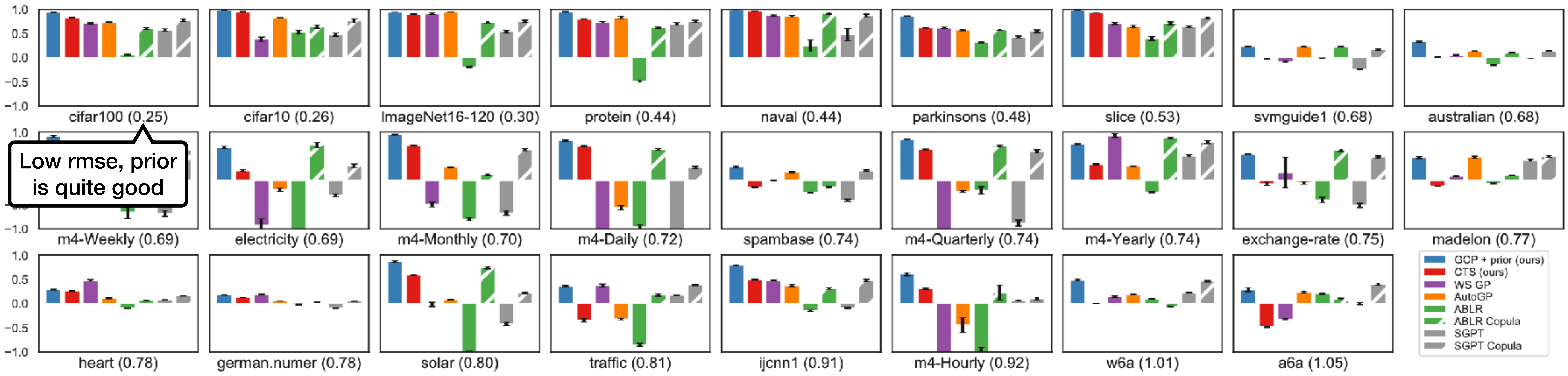
# Robustness to negative transfer




- If a new task differ from previous evaluations, we would still want to get reasonable performance!

- Improvement over random search (↑), tasks sorted by transfer learning difficulty (RMSE of prior predictor on the new task)

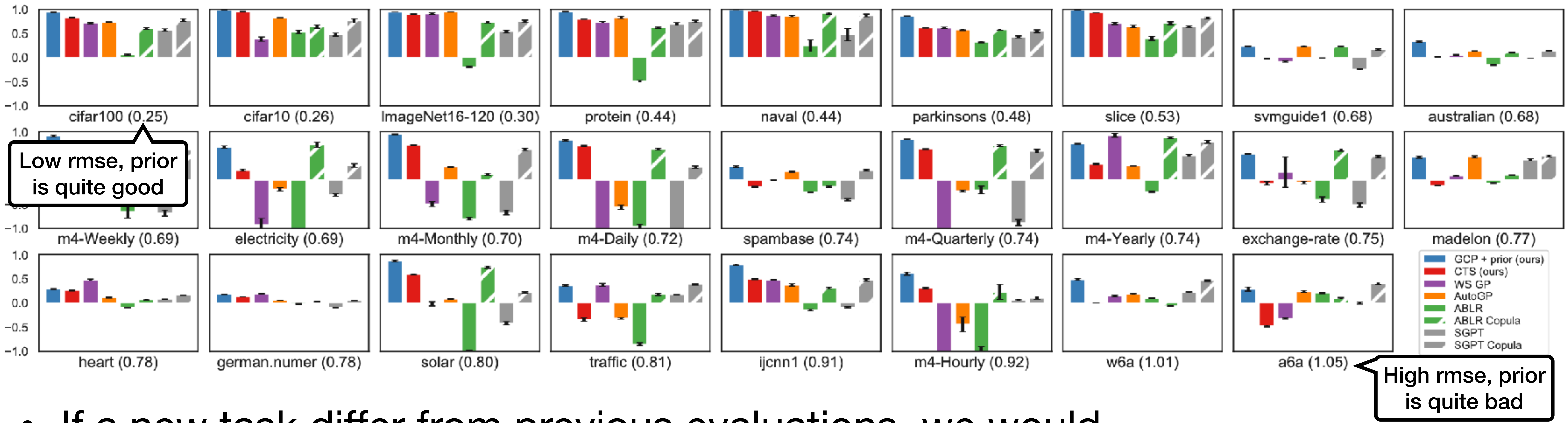
# Robustness to negative transfer




- If a new task differ from previous evaluations, we would still want to get reasonable performance!

- Improvement over random search (↑), tasks sorted by transfer learning difficulty (RMSE of prior predictor on the new task)

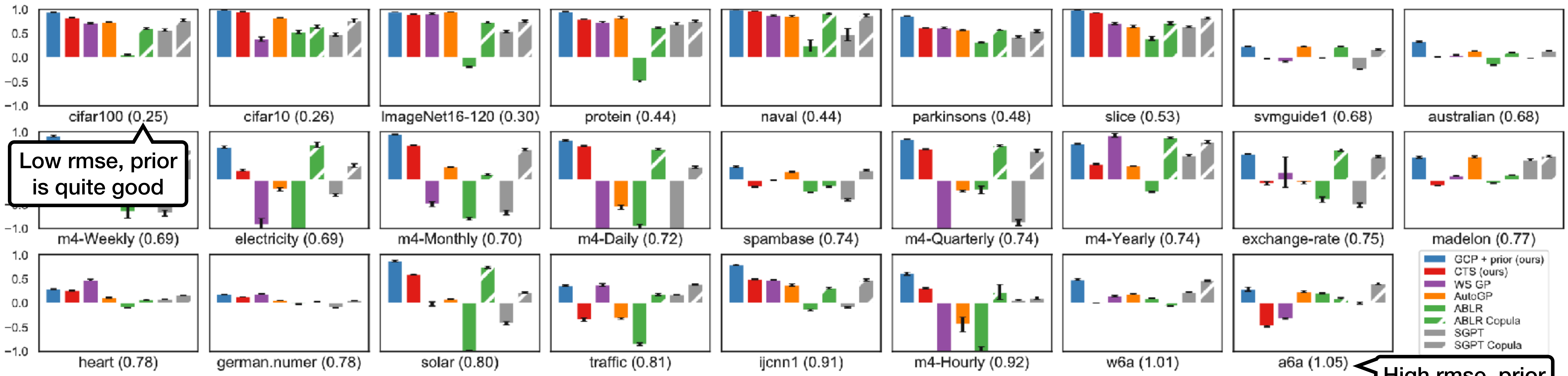
# Robustness to negative transfer




- If a new task differ from previous evaluations, we would still want to get reasonable performance!

- Improvement over random search (↑), tasks sorted by transfer learning difficulty (RMSE of prior predictor on the new task)

# Robustness to negative transfer



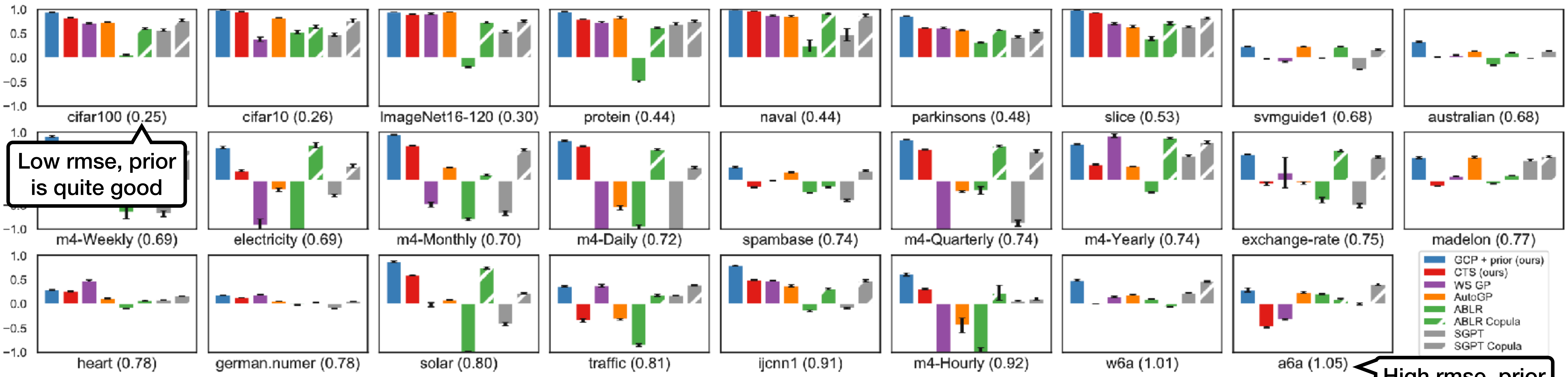

High rmse, prior is quite bad

- If a new task differ from previous evaluations, we would still want to get reasonable performance!

- Improvement over random search (↑), tasks sorted by transfer learning difficulty (RMSE of prior predictor on the new task)

GCP is robust to negative transfer even in challenging scenarios ...

# Robustness to negative transfer




High rmse, prior is quite bad

- If a new task differ from previous evaluations, we would still want to get reasonable performance!
- Improvement over random search (↑), tasks sorted by transfer learning difficulty (RMSE of prior predictor on the new task)

GCP is robust to negative transfer even in challenging scenarios …

… as opposed to CTS that just exploits prior from transfer learning

# GC3P pros and cons

# GC3P pros and cons

- GCP is robust to negative transfer even in challenging scenarios

# GC3P pros and cons

- GCP is robust to negative transfer even in challenging scenarios
- Robust to outliers and scale changes between tasks

# GC3P pros and cons

- GCP is robust to negative transfer even in challenging scenarios
- Robust to outliers and scale changes between tasks
- Can handle many offline evaluations without cubic bottleneck

# GC3P pros and cons

- GCP is robust to negative transfer even in challenging scenarios
- Robust to outliers and scale changes between tasks
- Can handle many offline evaluations without cubic bottleneck
- … But requires offline evaluations

# GC3P pros and cons

- GCP is robust to negative transfer even in challenging scenarios
- Robust to outliers and scale changes between tasks
- Can handle many offline evaluations without cubic bottleneck
- … But requires offline evaluations

🤔 **Could we avoid the need of offline evaluations, perhaps the user knows some good prior distribution?**

# Theoretical analysis

# Theoretical analysis

- Few work on analysing performance of transfer learning

# Theoretical analysis

- Few work on analysing performance of transfer learning
- Shoutout to [Ram 2023] for being one of the first!

# Theoretical analysis

- Few work on analysing performance of transfer learning
- Shoutout to [Ram 2023] for being one of the first!

**On the Optimality Gap of Warm-Started Hyperparameter Optimization**

**Parikshit Ram**[1]

[1]IBM Research AI

**Abstract** We study the general framework of warm-started hyperparameter optimization (HPO) where we have some source datasets (tasks) on which we have already performed HPO, and we wish to leverage the results of these HPO to warm-start the HPO on an unseen target dataset and perform few-shot HPO. Various meta-learning schemes have been proposed over the last decade (and more) for this problem. In this paper, we theoretically analyse the optimality gap of the hyperparameter obtained via such warm-started few-shot HPO, and provide novel results for multiple existing meta-learning schemes. We show how these results allow us identify situations where certain schemes have advantage over others.

# Theoretical analysis

- Few work on analysing performance of transfer learning
- Shoutout to [Ram 2023] for being one of the first!
- Study bound on few shot hyperparameter optimization for pruning and surrogate based approaches

## On the Optimality Gap of Warm-Started Hyperparameter Optimization

**Parikshit Ram**[1]

[1]IBM Research AI

**Abstract** We study the general framework of warm-started hyperparameter optimization (HPO) where we have some source datasets (tasks) on which we have already performed HPO, and we wish to leverage the results of these HPO to warm-start the HPO on an unseen target dataset and perform few-shot HPO. Various meta-learning schemes have been proposed over the last decade (and more) for this problem. In this paper, we theoretically analyse the optimality gap of the hyperparameter obtained via such warm-started few-shot HPO, and provide novel results for multiple existing meta-learning schemes. We show how these results allow us identify situations where certain schemes have advantage over others.

# Theoretical analysis

- Few work on analysing performance of transfer learning
- Shoutout to [Ram 2023] for being one of the first!
- Study bound on few shot hyperparameter optimization for pruning and surrogate based approaches
- Formalize assumption that “tasks are similar”

## On the Optimality Gap of Warm-Started Hyperparameter Optimization

**Parikshit Ram**[1]

[1]IBM Research AI

**Abstract** We study the general framework of warm-started hyperparameter optimization (HPO) where we have some source datasets (tasks) on which we have already performed HPO, and we wish to leverage the results of these HPO to warm-start the HPO on an unseen target dataset and perform few-shot HPO. Various meta-learning schemes have been proposed over the last decade (and more) for this problem. In this paper, we theoretically analyse the optimality gap of the hyperparameter obtained via such warm-started few-shot HPO, and provide novel results for multiple existing meta-learning schemes. We show how these results allow us identify situations where certain schemes have advantage over others.

# Theoretical analysis

- Few work on analysing performance of transfer learning
- Shoutout to [Ram 2023] for being one of the first!
- Study bound on few shot hyperparameter optimization for pruning and surrogate based approaches
- Formalize assumption that “tasks are similar”
- In the case of surrogate based approaches (such as GCP)

## On the Optimality Gap of Warm-Started Hyperparameter Optimization

Parikshit Ram[1]

[1]IBM Research AI

**Abstract** We study the general framework of warm-started hyperparameter optimization (HPO) where we have some source datasets (tasks) on which we have already performed HPO, and we wish to leverage the results of these HPO to warm-start the HPO on an unseen target dataset and perform few-shot HPO. Various meta-learning schemes have been proposed over the last decade (and more) for this problem. In this paper, we theoretically analyse the optimality gap of the hyperparameter obtained via such warm-started few-shot HPO, and provide novel results for multiple existing meta-learning schemes. We show how these results allow us identify situations where certain schemes have advantage over others.

# Theoretical analysis

- Few work on analysing performance of transfer learning
- Shoutout to [Ram 2023] for being one of the first!
- Study bound on few shot hyperparameter optimization for pruning and surrogate based approaches
- Formalize assumption that “tasks are similar”
- In the case of surrogate based approaches (such as GCP)

**On the Optimality Gap of**
**Warm-Started Hyperparameter Optimization**

**Parikshit Ram**[1]

[1]IBM Research AI

**Abstract** We study the general framework of warm-started hyperparameter optimization (HPO) where we have some source datasets (tasks) on which we have already performed HPO, and we wish to leverage the results of these HPO to warm-start the HPO on an unseen target dataset and perform few-shot HPO. Various meta-learning schemes have been proposed over the last decade (and more) for this problem. In this paper, we theoretically analyse the optimality gap of the hyperparameter obtained via such warm-started few-shot HPO, and provide novel results for multiple existing meta-learning schemes. We show how these results allow us identify situations where certain schemes have advantage over others.

*Assumption* 3.2. For each surrogate loss function $s_t, t \in [T]$, we assume that, for some small $\epsilon > 0$

$$|L(\theta; D_t) - s_t(\theta)| \leq \epsilon \forall \theta \in \Theta. \quad (5)$$

# Theoretical analysis

- Few work on analysing performance of transfer learning
- Shoutout to [Ram 2023] for being one of the first!
- Study bound on few shot hyperparameter optimization for pruning and surrogate based approaches
- Formalize assumption that “tasks are similar”
- In the case of surrogate based approaches (such as GCP)

**On the Optimality Gap of Warm-Started Hyperparameter Optimization**

Parikshit Ram[1]

[1]IBM Research AI

**Abstract** We study the general framework of warm-started hyperparameter optimization (HPO) where we have some source datasets (tasks) on which we have already performed HPO, and we wish to leverage the results of these HPO to warm-start the HPO on an unseen target dataset and perform few-shot HPO. Various meta-learning schemes have been proposed over the last decade (and more) for this problem. In this paper, we theoretically analyse the optimality gap of the hyperparameter obtained via such warm-started few-shot HPO, and provide novel results for multiple existing meta-learning schemes. We show how these results allow us identify situations where certain schemes have advantage over others.

*Assumption* 3.2. For each surrogate loss function $s_t, t \in [T]$, we assume that, for some small $\epsilon > 0$

$$|L(\theta; D_t) - s_t(\theta)| \leq \epsilon \forall \theta \in \Theta. \quad (5)$$

Assume that surrogate error is small

# Theoretical analysis

- Few work on analysing performance of transfer learning
- Shoutout to [Ram 2023] for being one of the first!
- Study bound on few shot hyperparameter optimization for pruning and surrogate based approaches
- Formalize assumption that “tasks are similar”
- In the case of surrogate based approaches (such as GCP)

## On the Optimality Gap of Warm-Started Hyperparameter Optimization

Parikshit Ram[1]

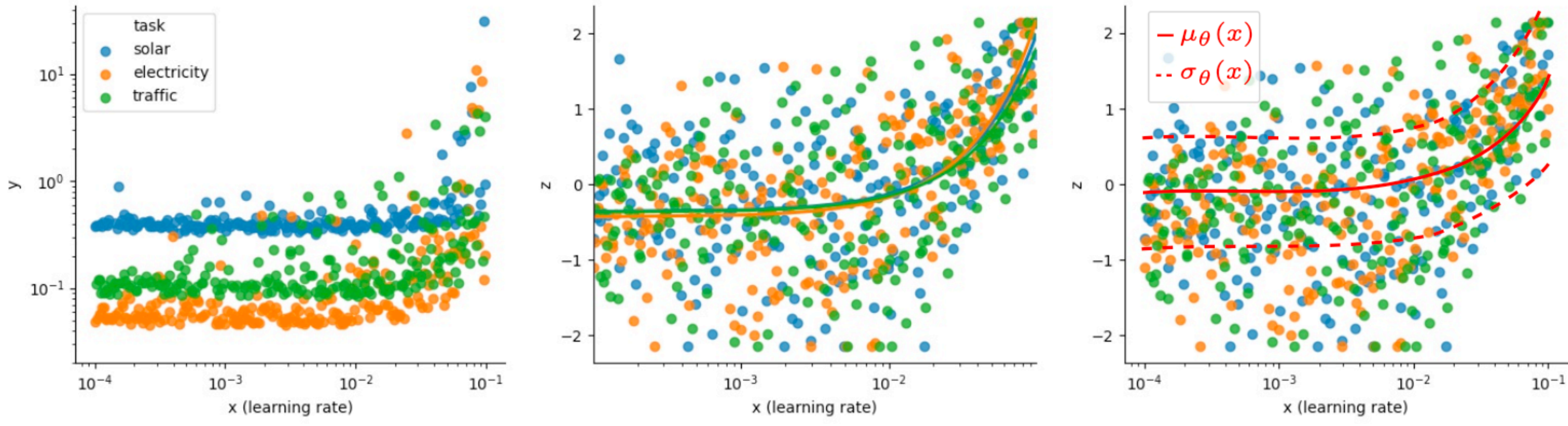
[1]IBM Research AI

**Abstract** We study the general framework of warm-started hyperparameter optimization (HPO) where we have some source datasets (tasks) on which we have already performed HPO, and we wish to leverage the results of these HPO to warm-start the HPO on an unseen target dataset and perform few-shot HPO. Various meta-learning schemes have been proposed over the last decade (and more) for this problem. In this paper, we theoretically analyse the optimality gap of the hyperparameter obtained via such warm-started few-shot HPO, and provide novel results for multiple existing meta-learning schemes. We show how these results allow us identify situations where certain schemes have advantage over others.

*Assumption* 3.2. For each surrogate loss function $s_t, t \in [T]$, we assume that, for some small $\epsilon > 0$

Assume that surrogate error is small

$$|L(\theta; D_t) - s_t(\theta)| \leq \epsilon \forall \theta \in \Theta. \quad (5)$$




**Left:** Plot blackbox error $y$ in log-space against a single hyperparameter $x$ for different tasks.
**Middle:** Running mean after transforming each task objectives with $z = \psi(y) = \Phi^{-1} \circ F(y)$.
**Right:** Illustrative plot of possible mean/variance fit of a model $\mu_\theta(x), \sigma_\theta(x)$ trained jointly on al
with shared parameters $\theta$.

# Theoretical analysis

- Few work on analysing performance of transfer learning
- Shoutout to [Ram 2023] for being one of the first!
- Study bound on few shot hyperparameter optimization for pruning and surrogate based approaches
- Formalize assumption that "tasks are similar"
- In the case of surrogate based approaches (such as GCP)

## On the Optimality Gap of Warm-Started Hyperparameter Optimization

**Parikshit Ram**[1]

[1]IBM Research AI

**Abstract** We study the general framework of warm-started hyperparameter optimization (HPO) where we have some source datasets (tasks) on which we have already performed HPO, and we wish to leverage the results of these HPO to warm-start the HPO on an unseen target dataset and perform few-shot HPO. Various meta-learning schemes have been proposed over the last decade (and more) for this problem. In this paper, we theoretically analyse the optimality gap of the hyperparameter obtained via such warm-started few-shot HPO, and provide novel results for multiple existing meta-learning schemes. We show how these results allow us identify situations where certain schemes have advantage over others.

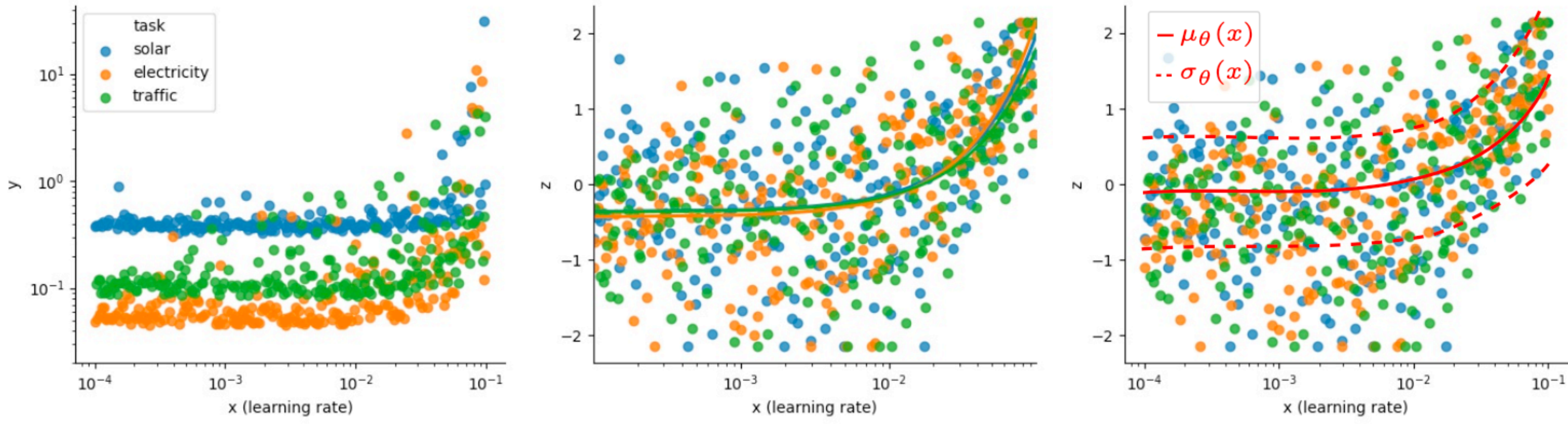
*Assumption* 3.2. For each surrogate loss function $s_t, t \in [T]$, we assume that, for some small $\epsilon > 0$

Assume that surrogate error is small

$$|L(\theta; D_t) - s_t(\theta)| \leq \epsilon \forall \theta \in \Theta. \quad (5)$$

**Corollary 5.1.** *Under conditions of Theorem 5.1 and Assumption 3.2, we bound the optimality ga*

$$L(\hat{\theta}; D) - L(\theta^\star; D) \leq 2\epsilon + 2\beta \cdot \max_{\theta \in \Theta} \sum_{t \in [T]} \alpha_t(\theta) W_1 \left(P_\theta(D), P_\theta(D_t)\right).$$




**Left:** Plot blackbox error $y$ in log-space against a single hyperparameter $x$ for different tasks.
**Middle:** Running mean after transforming each task objectives with $z = \psi(y) = \Phi^{-1} \circ F(y)$.
**Right:** Illustrative plot of possible mean/variance fit of a model $\mu_\theta(x), \sigma_\theta(x)$ trained jointly on al with shared parameters $\theta$.

# Theoretical analysis

- Few work on analysing performance of transfer learning
- Shoutout to [Ram 2023] for being one of the first!
- Study bound on few shot hyperparameter optimization for pruning and surrogate based approaches
- Formalize assumption that “tasks are similar”
- In the case of surrogate based approaches (such as GCP)

**On the Optimality Gap of Warm-Started Hyperparameter Optimization**


Parikshit Ram[1]

[1]IBM Research AI



**Abstract** We study the general framework of warm-started hyperparameter optimization (HPO) where we have some source datasets (tasks) on which we have already performed HPO, and we wish to leverage the results of these HPO to warm-start the HPO on an unseen target dataset and perform few-shot HPO. Various meta-learning schemes have been proposed over the last decade (and more) for this problem. In this paper, we theoretically analyse the optimality gap of the hyperparameter obtained via such warm-started few-shot HPO, and provide novel results for multiple existing meta-learning schemes. We show how these results allow us identify situations where certain schemes have advantage over others.


*Assumption* 3.2. For each surrogate loss function $s_t, t \in [T]$, we assume that, for some small $\epsilon > 0$

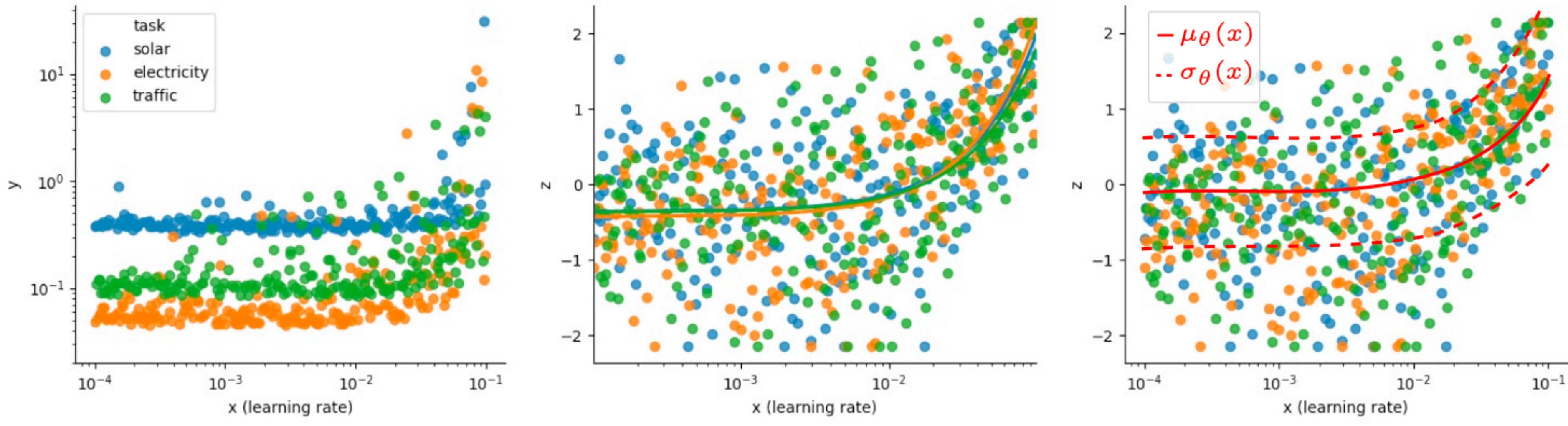
Assume that surrogate error is small

$$|L(\theta; D_t) - s_t(\theta)| \leq \epsilon \forall \theta \in \Theta. \quad (5)$$

Gap with optimal performance

*of Theorem 5.1 and Assumption 3.2, we bound the optimality ga*

$$L(\hat{\theta}; D) - L(\theta^\star; D) \leq 2\epsilon + 2\beta \cdot \max_{\theta \in \Theta} \sum_{t \in [T]} \alpha_t(\theta) W_1\left(P_\theta(D), P_\theta(D_t)\right).$$




**Left:** Plot blackbox error $y$ in log-space against a single hyperparameter $x$ for different tasks.
**Middle:** Running mean after transforming each task objectives with $z = \psi(y) = \Phi^{-1} \circ F(y)$.
**Right:** Illustrative plot of possible mean/variance fit of a model $\mu_\theta(x), \sigma_\theta(x)$ trained jointly on al
with shared parameters $\theta$.

# Theoretical analysis

- Few work on analysing performance of transfer learning
- Shoutout to [Ram 2023] for being one of the first!
- Study bound on few shot hyperparameter optimization for pruning and surrogate based approaches
- Formalize assumption that “tasks are similar”
- In the case of surrogate based approaches (such as GCP)

**On the Optimality Gap of Warm-Started Hyperparameter Optimization**

**Parikshit Ram**[1]

[1]IBM Research AI

**Abstract** We study the general framework of warm-started hyperparameter optimization (HPO) where we have some source datasets (tasks) on which we have already performed HPO, and we wish to leverage the results of these HPO to warm-start the HPO on an unseen target dataset and perform few-shot HPO. Various meta-learning schemes have been proposed over the last decade (and more) for this problem. In this paper, we theoretically analyse the optimality gap of the hyperparameter obtained via such warm-started few-shot HPO, and provide novel results for multiple existing meta-learning schemes. We show how these results allow us identify situations where certain schemes have advantage over others.

*Assumption* 3.2. For each surrogate loss function $s_t, t \in [T]$, we assume that, for some small $\epsilon > 0$

Assume that surrogate error is small

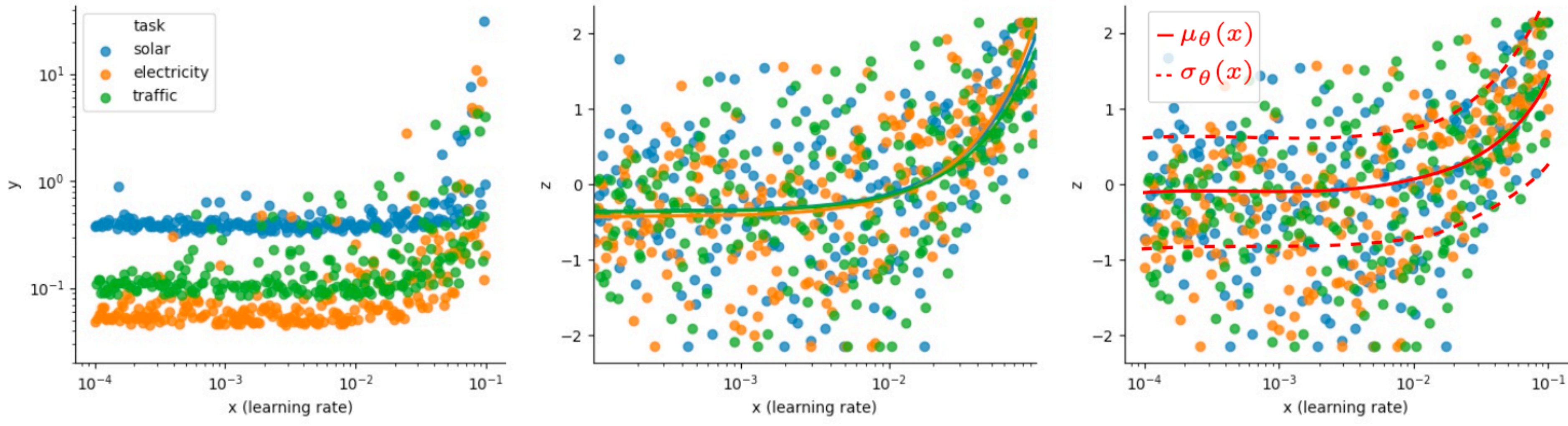
$$|L(\theta; D_t) - s_t(\theta)| \leq \epsilon \forall \theta \in \Theta. \quad (5)$$

Gap with optimal performance

*of Theorem 5.1 and Assumption 3.2, we bound the optimality ga*

$$L(\hat{\theta}; D) - L(\theta^\star; D) \leq 2\epsilon + 2\beta \cdot \max_{\theta \in \Theta} \sum_{t \in [T]} \alpha_t(\theta) W_1\left(P_\theta(D), P_\theta(D_t)\right).$$

Wasserstein distance between the new task and task $t$




**Left:** Plot blackbox error $y$ in log-space against a single hyperparameter $x$ for different tasks.
**Middle:** Running mean after transforming each task objectives with $z = \psi(y) = \Phi^{-1} \circ F(y)$.
**Right:** Illustrative plot of possible mean/variance fit of a model $\mu_\theta(x), \sigma_\theta(x)$ trained jointly on al with shared parameters $\theta$.

# Theoretical analysis

- Few work on analysing performance of transfer learning
- Shoutout to [Ram 2023] for being one of the first!
- Study bound on few shot hyperparameter optimization for pruning and surrogate based approaches
- Formalize assumption that “tasks are similar”
- In the case of surrogate based approaches (such as GCP)

## On the Optimality Gap of Warm-Started Hyperparameter Optimization

**Parikshit Ram**[1]

[1]IBM Research AI

**Abstract** We study the general framework of warm-started hyperparameter optimization (HPO) where we have some source datasets (tasks) on which we have already performed HPO, and we wish to leverage the results of these HPO to warm-start the HPO on an unseen target dataset and perform few-shot HPO. Various meta-learning schemes have been proposed over the last decade (and more) for this problem. In this paper, we theoretically analyse the optimality gap of the hyperparameter obtained via such warm-started few-shot HPO, and provide novel results for multiple existing meta-learning schemes. We show how these results allow us identify situations where certain schemes have advantage over others.

*Assumption* 3.2. For each surrogate loss function $s_t, t \in [T]$, we assume that, for some small $\epsilon > 0$

Assume that surrogate error is small

$$|L(\theta; D_t) - s_t(\theta)| \leq \epsilon \forall \theta \in \Theta. \qquad (5)$$

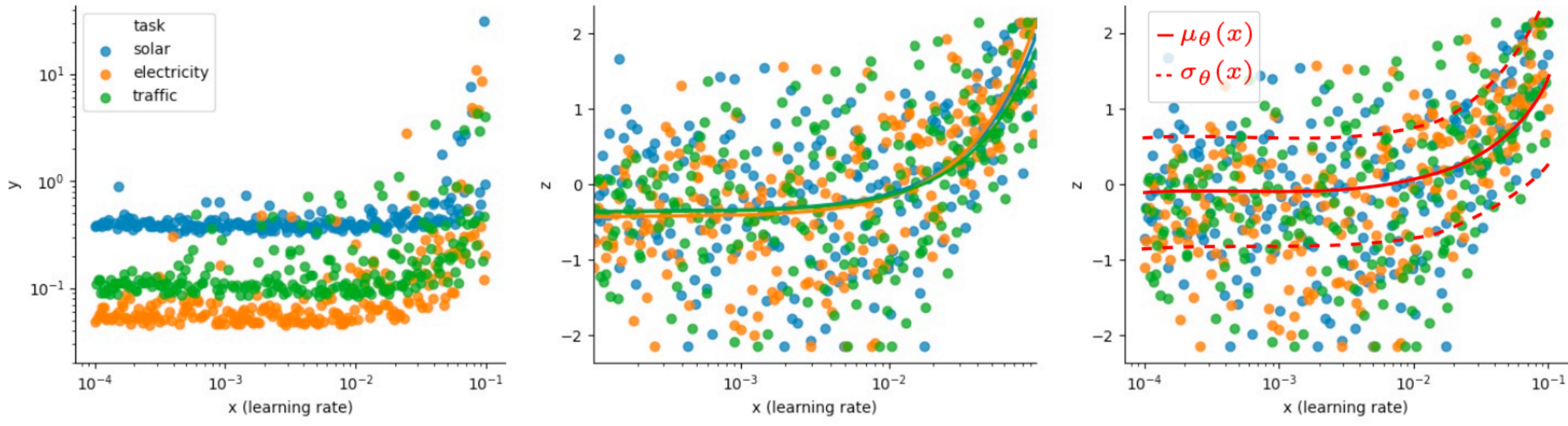
Gap with optimal performance

*of Theorem 5.1 and Assumption 3.2, we bound the optimality ga*

$$L(\hat{\theta}; D) - L(\theta^\star; D) \leq 2\epsilon + 2\beta \cdot \max_{\theta \in \Theta} \sum_{t \in [T]} \alpha_t(\theta) W_1\left(P_\theta(D), P_\theta(D_t)\right).$$

Task weights which can be based on dataset features

Wasserstein distance between the new task and task $t$




**Left:** Plot blackbox error $y$ in log-space against a single hyperparameter $x$ for different tasks.
**Middle:** Running mean after transforming each task objectives with $z = \psi(y) = \Phi^{-1} \circ F(y)$.
**Right:** Illustrative plot of possible mean/variance fit of a model $\mu_\theta(x), \sigma_\theta(x)$ trained jointly on al with shared parameters $\theta$.

# Theoretical analysis

- Few work on analysing performance of transfer learning
- Shoutout to [Ram 2023] for being one of the first!
- Study bound on few shot hyperparameter optimization for pruning and surrogate based approaches
- Formalize assumption that "tasks are similar"
- In the case of surrogate based approaches (such as GCP)

On the Optimality Gap of
Warm-Started Hyperparameter Optimization

Parikshit Ram[1]
[1]IBM Research AI

**Abstract** We study the general framework of warm-started hyperparameter optimization (HPO) where we have some source datasets (tasks) on which we have already performed HPO, and we wish to leverage the results of these HPO to warm-start the HPO on an unseen target dataset and perform few-shot HPO. Various meta-learning schemes have been proposed over the last decade (and more) for this problem. In this paper, we theoretically analyse the optimality gap of the hyperparameter obtained via such warm-started few-shot HPO, and provide novel results for multiple existing meta-learning schemes. We show how these results allow us identify situations where certain schemes have advantage over others.

*Assumption* 3.2. For each surrogate loss function $s_t, t \in [T]$, we assume that, for some small $\epsilon > 0$

Assume that surrogate error is small

$$|L(\theta; D_t) - s_t(\theta)| \leq \epsilon \forall \theta \in \Theta. \tag{5}$$

Error due to surrogate

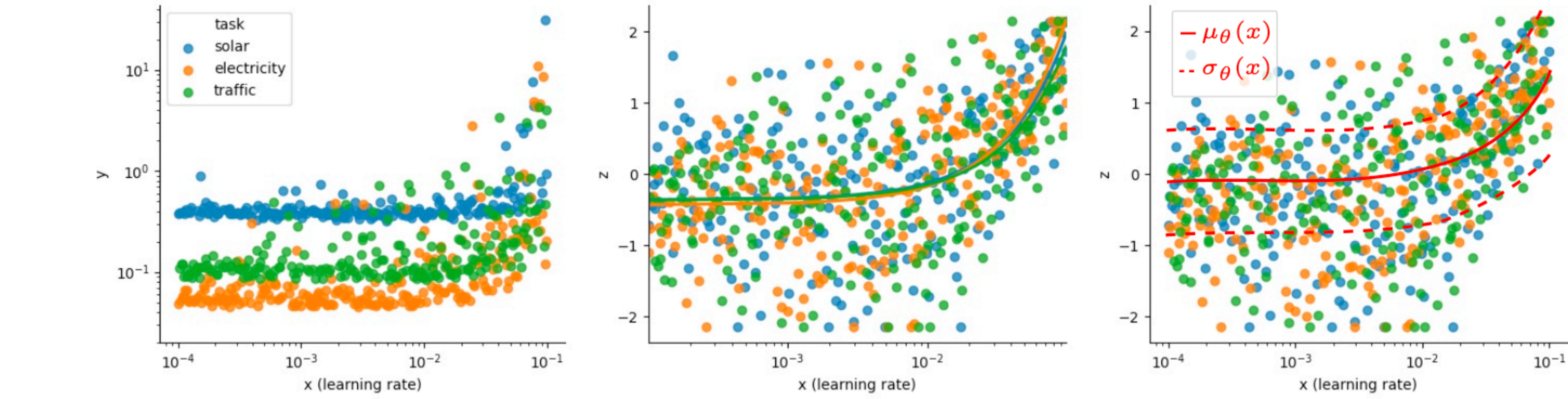
Gap with optimal performance

*of Th… orem 5.1 and Assumption 3.2, we bound the optimality ga…*

$$L(\hat{\theta}; D) - L(\theta^\star; D) \leq 2\epsilon + 2\beta \cdot \max_{\theta \in \Theta} \sum_{t \in [T]} \alpha_t(\theta) W_1 \left(P_\theta(D), P_\theta(D_t)\right).$$

Task weights which can be based on dataset features

Wasserstein distance between the new task and task $t$




**Left:** Plot blackbox error $y$ in log-space against a single hyperparameter $x$ for different tasks.
**Middle:** Running mean after transforming each task objectives with $z = \psi(y) = \Phi^{-1} \circ F(y)$.
**Right:** Illustrative plot of possible mean/variance fit of a model $\mu_\theta(x), \sigma_\theta(x)$ trained jointly on al… with shared parameters $\theta$.

# Leverage user priors

# Leverage user priors

- GC3P works great but require offline data to estimate the prior distribution

# Leverage user priors

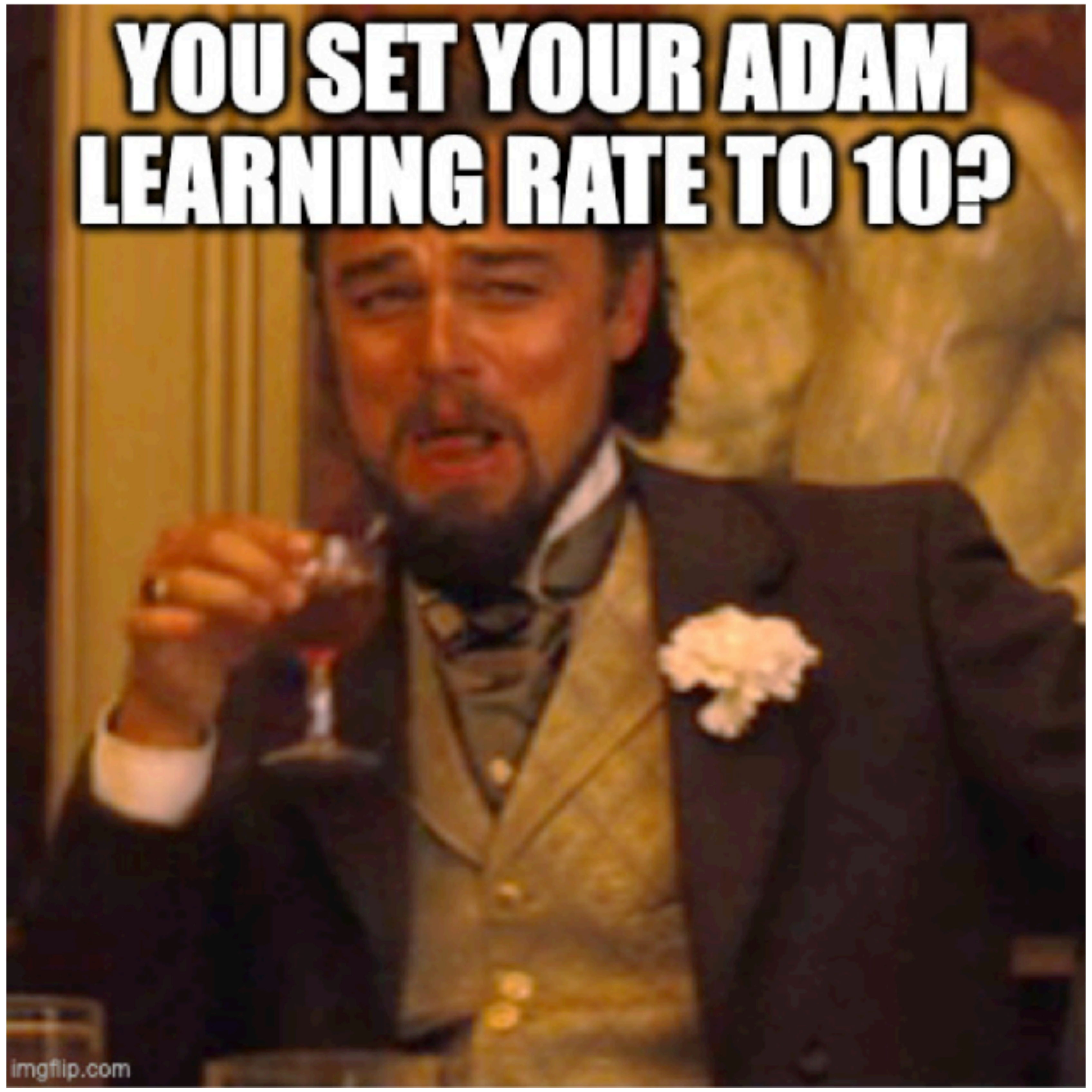
- GC3P works great but require offline data to estimate the prior distribution
- What if you don’t have data but know region that aim to work well or not?

# Leverage user priors

- GC3P works great but require offline data to estimate the prior distribution
- What if you don't have data but know region that aim to work well or not?
- Practitioners often know good range of hyperparameters

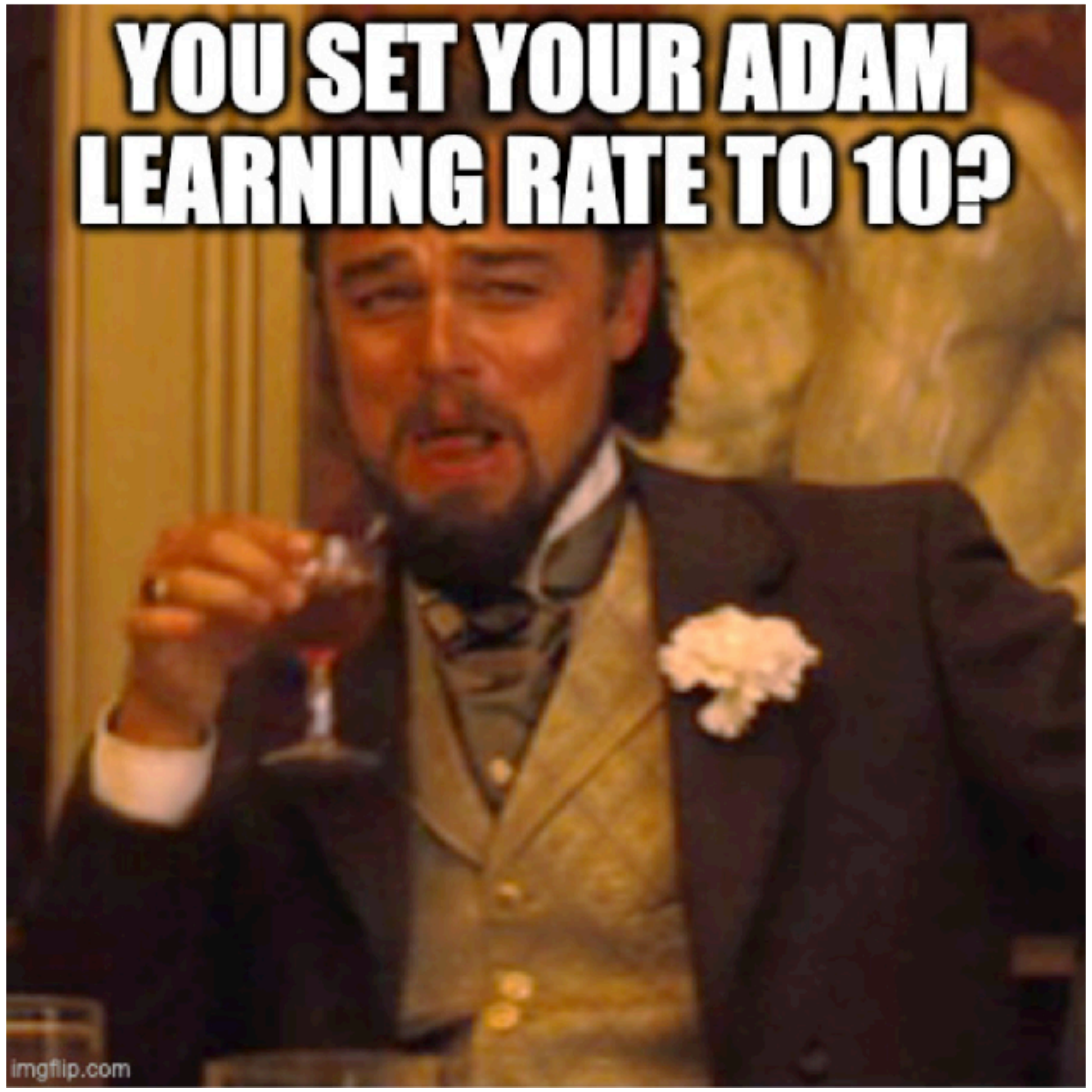
# Leverage user priors

- GC3P works great but require offline data to estimate the prior distribution
- What if you don't have data but know region that aim to work well or not?
- Practitioners often know good range of hyperparameters



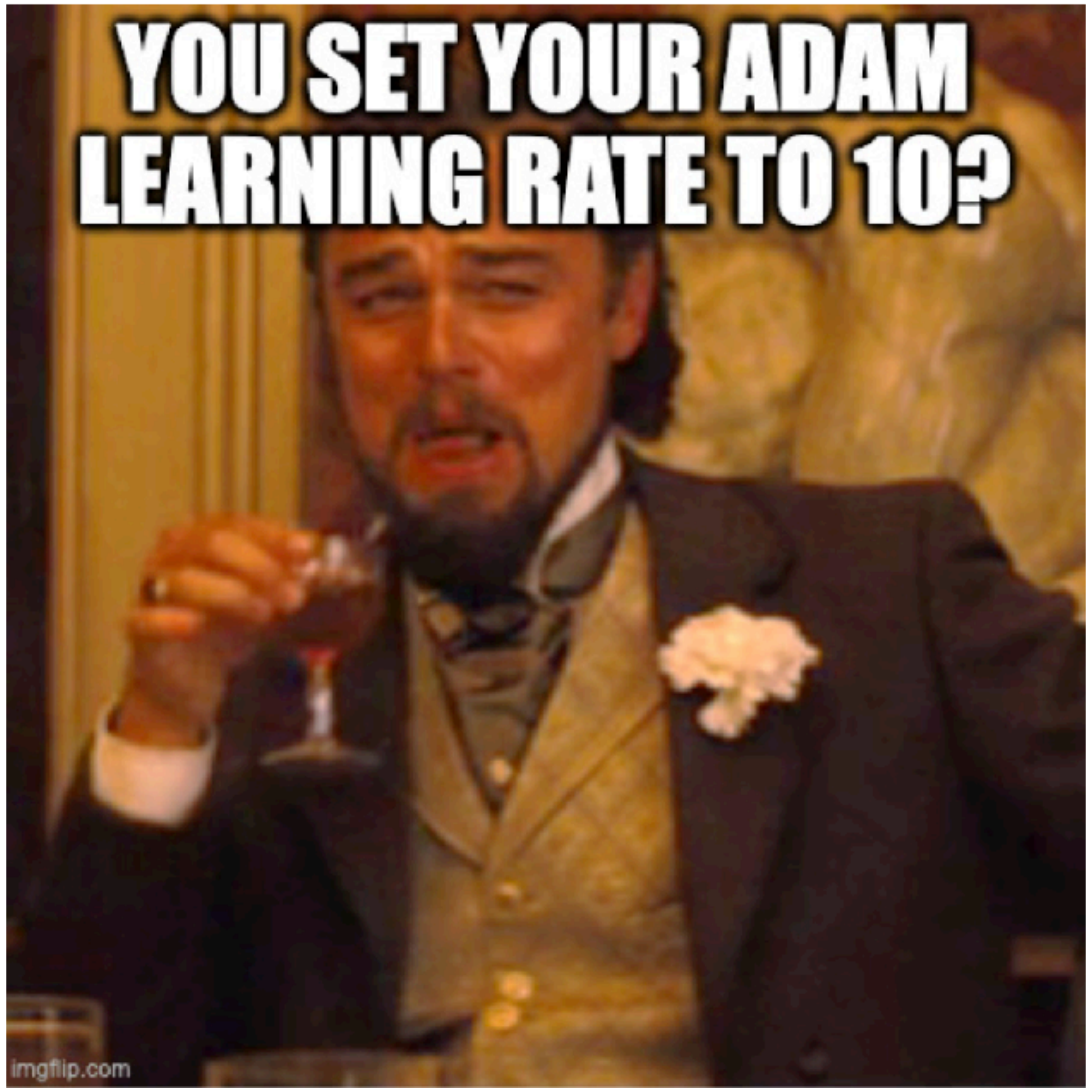
# Leverage user priors

- GC3P works great but require offline data to estimate the prior distribution
- What if you don't have data but know region that aim to work well or not?
- Practitioners often know good range of hyperparameters
- What if we ask them their prior instead of learning it?



# Leverage user priors

- GC3P works great but require offline data to estimate the prior distribution
- What if you don't have data but know region that aim to work well or not?
- Practitioners often know good range of hyperparameters
- What if we ask them their prior instead of learning it?
- Needs robustness to potential user prior misspecification



# Leverage user priors

- Practitioners often know good range of hyperparameters
- What if we ask them their prior instead of learning it?
- Needs robustness to potential user prior misspecification

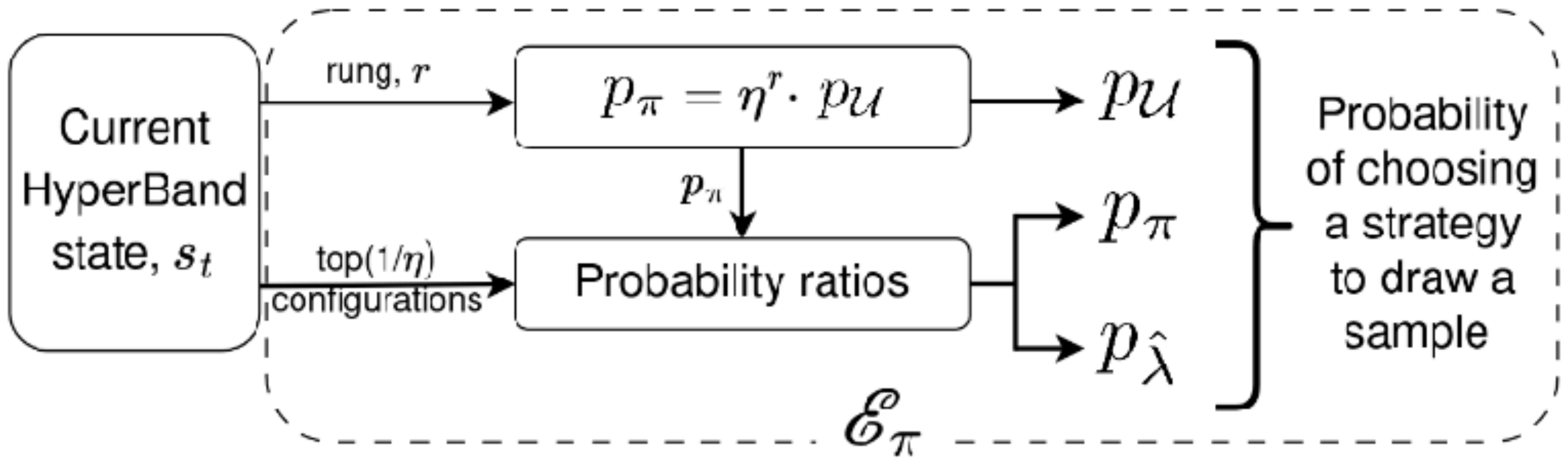
# Leverage user priors

- Practitioners often know good range of hyperparameters
- What if we ask them their prior instead of learning it?
- Needs robustness to potential user prior misspecification

PriorBand: Practical Hyperparameter Optimization in the Age of Deep Learning [Mallik 2023]

# Leverage user priors

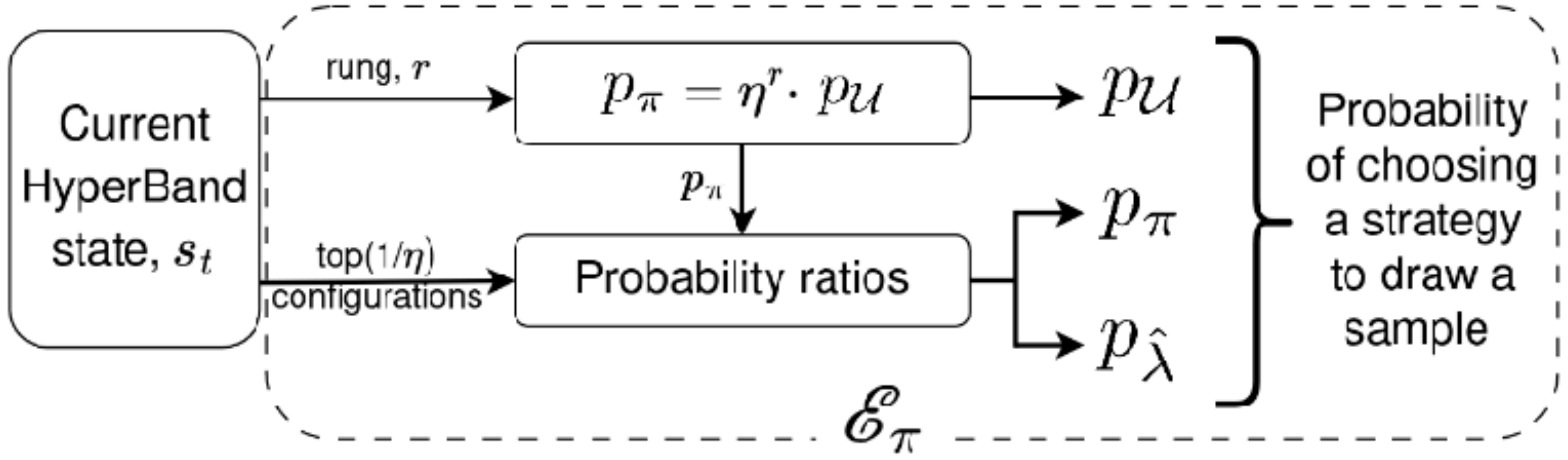
- Practitioners often know good range of hyperparameters
- What if we ask them their prior instead of learning it?
- Needs robustness to potential user prior misspecification




PriorBand: Practical Hyperparameter Optimization in the Age of Deep Learning [Mallik 2023]

# Leverage user priors

- Practitioners often know good range of hyperparameters
- What if we ask them their prior instead of learning it?
- Needs robustness to potential user prior misspecification

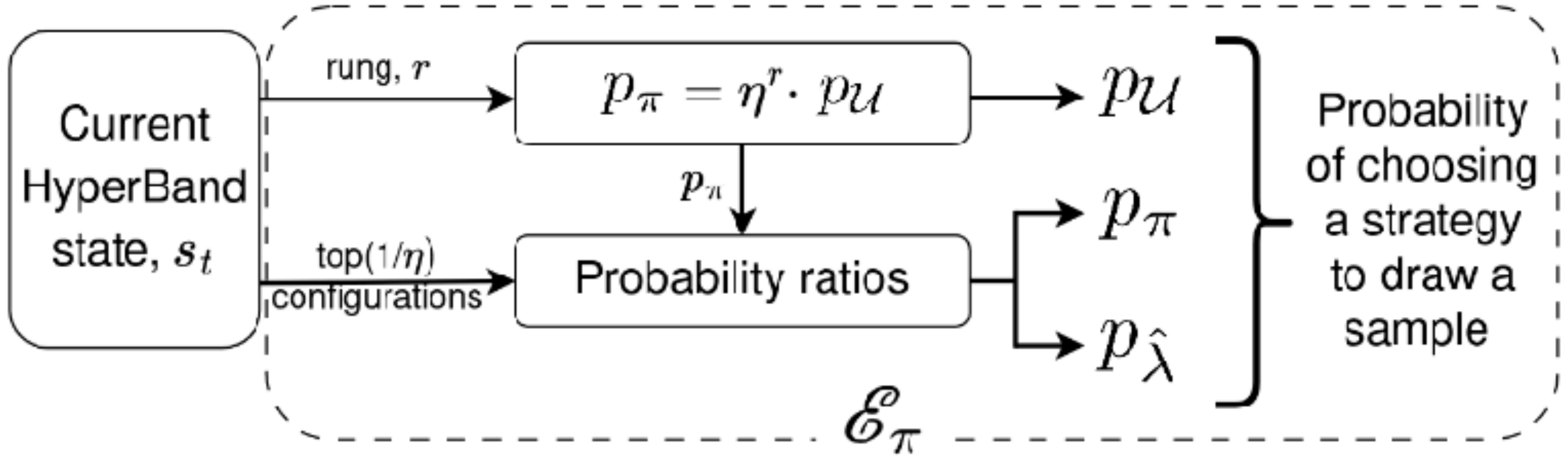



Learn which strategy is better between user strategy and random search

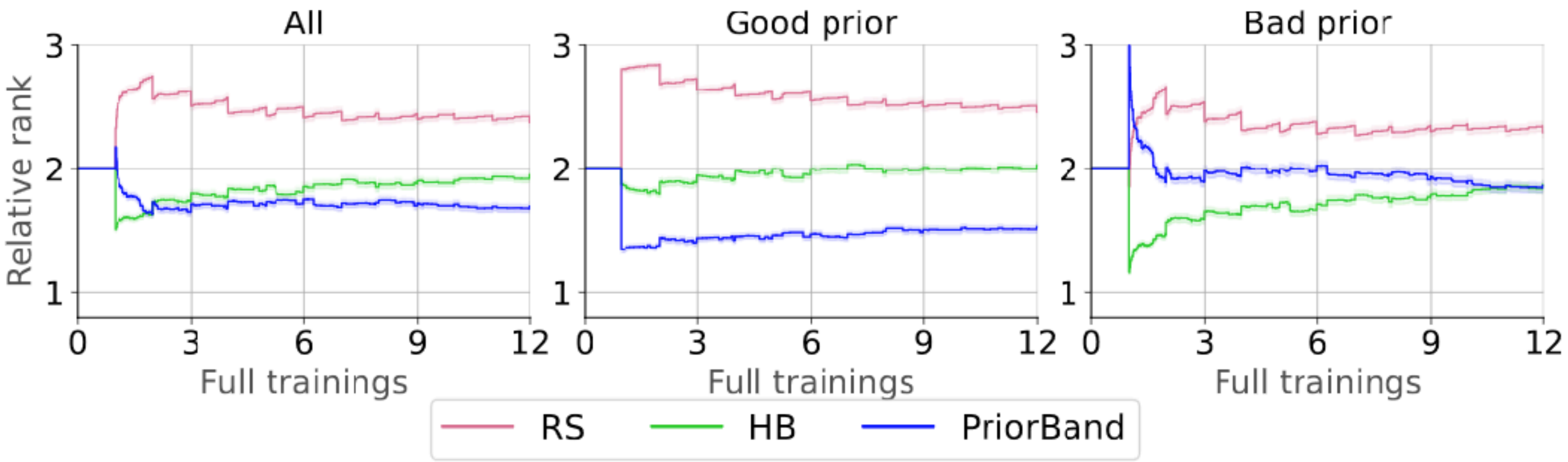
PriorBand: Practical Hyperparameter Optimization in the Age of Deep Learning [Mallik 2023]

# Leverage user priors

- Practitioners often know good range of hyperparameters
- What if we ask them their prior instead of learning it?
- Needs robustness to potential user prior misspecification



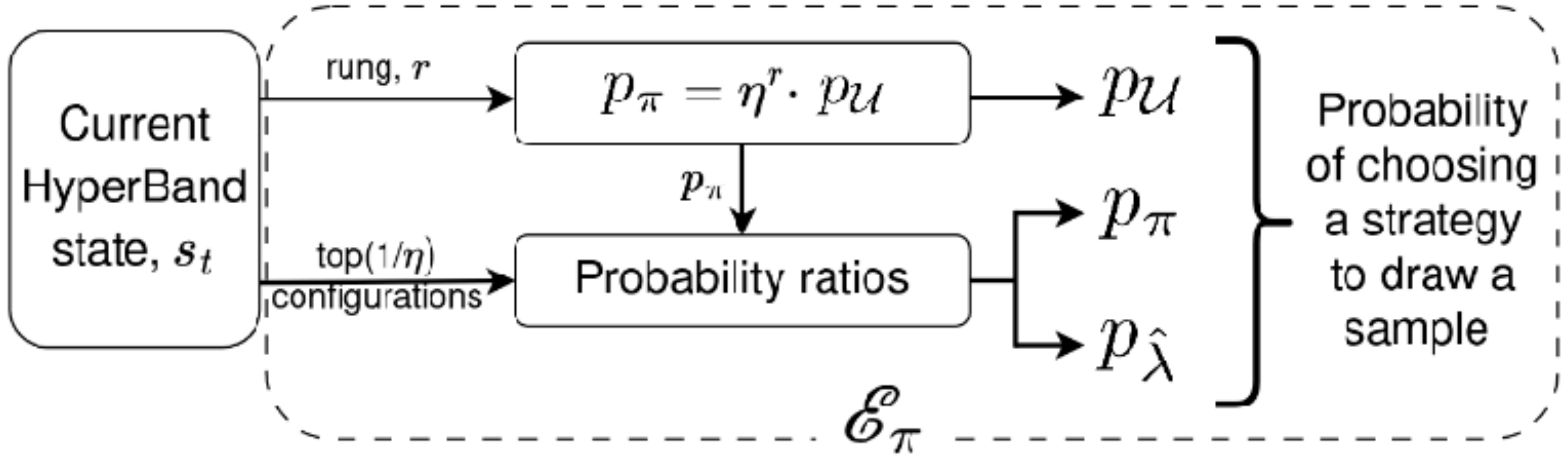

Learn which strategy is better between user strategy and random search



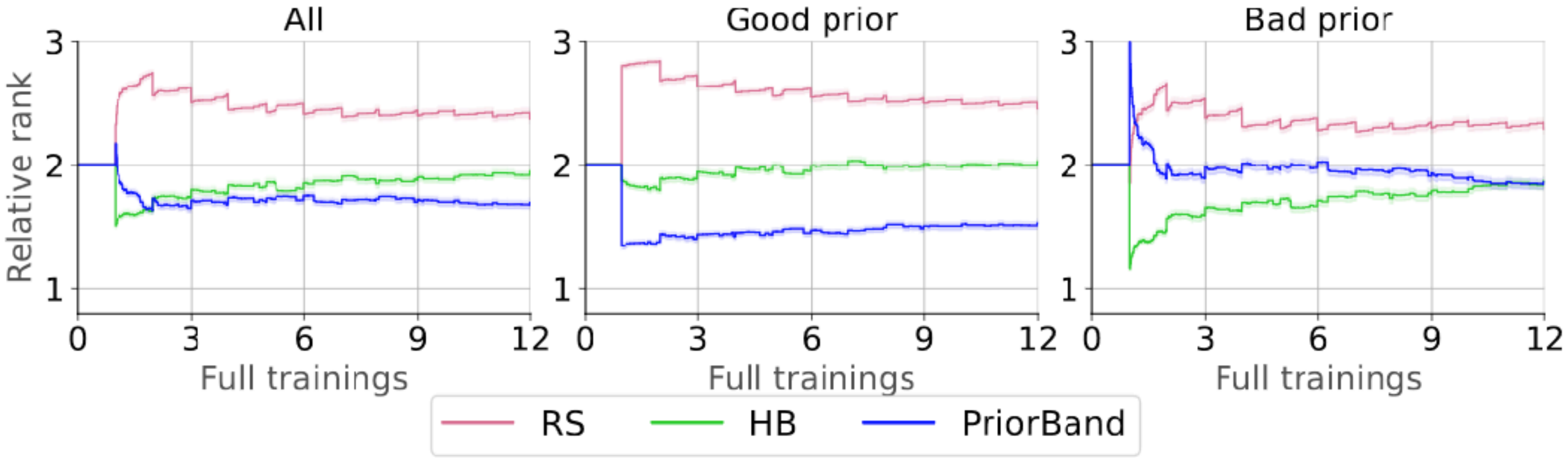

PriorBand: Practical Hyperparameter Optimization in the Age of Deep Learning [Mallik 2023]

# Leverage user priors

- Practitioners often know good range of hyperparameters
- What if we ask them their prior instead of learning it?
- Needs robustness to potential user prior misspecification




Learn which strategy is better between user strategy and random search




🥳 Learning which strategy is better allows failure in cases when user priors are bad

PriorBand: Practical Hyperparameter Optimization in the Age of Deep Learning [Mallik 2023]

# Methods

## Optformer

# Optformer

# Optformer

- Assume you have *a lot* of offline evaluation of tuning runs

# Optformer

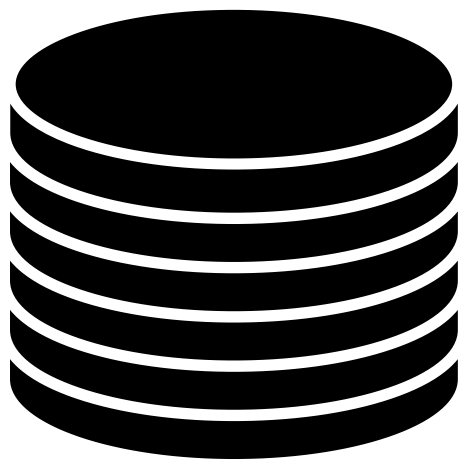
- Assume you have *a lot* of offline evaluation of tuning runs

Table 1: Example of a study $(m, \boldsymbol{h})$ with two parameters and two trials. Metadata $m$ appears in blue and history $\boldsymbol{h}$ in purple.


```
"name": "convnet on cifar10",
"metric": "accuracy",
"goal": "MAXIMIZE",
"algorithm": "random_search",
"parameter": {
"name":  "opt_kw.lr",
"type":  "DOUBLE",
"min_value":  1e-6,
"max_value":  1e-2,
"scale_type":  "LOG"
}
"parameter": {
"name":  "opt_type",
"type":  "CATEGORICAL",
"categories":  ["SGD", "Adam"],
}
"trial" {
"parameter":  {
"opt_kw.lr":  0.0021237573,
"opt_type":  "SGD"
}
"metric":  {
"accuracy":  0.69482429,
}}
"trial" {
"parameter":  {
"opt_kw.lr":  0.00038292234,
"opt_type":  "Adam"
}
"metric":  {
"accuracy":  0.71642583
}}
```

# Optformer

- Assume you have *a lot* of offline evaluation of tuning runs



**Database of HPO runs containing**

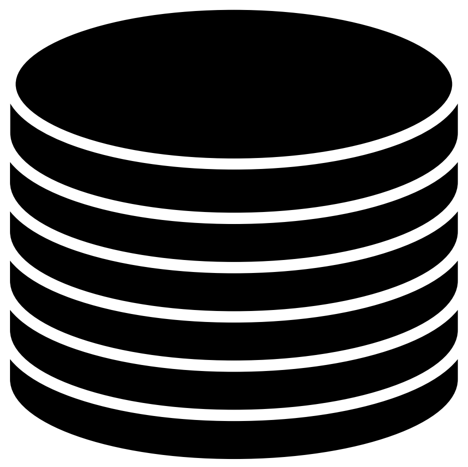
- Metadata of problem studied (Convolution? Tabular?)
- Search space considered
- HPO method used
- List of hyperparameters evaluated and blackbox accuracy

Table 1: Example of a study $(m, \boldsymbol{h})$ with two parameters and two trials. Metadata $m$ appears in blue and history $\boldsymbol{h}$ in purple.


```
"name": "convnet on cifar10",
"metric": "accuracy",
"goal": "MAXIMIZE",
"algorithm": "random_search",
"parameter": {
"name":  "opt_kw.lr",
"type":  "DOUBLE",
"min_value":  1e-6,
"max_value":  1e-2,
"scale_type":  "LOG"
}
"parameter": {
"name":  "opt_type",
"type":  "CATEGORICAL",
"categories":  ["SGD", "Adam"],
}
"trial" {
"parameter":  {
"opt_kw.lr":  0.0021237573,
"opt_type":  "SGD"
}
"metric":  {
"accuracy":  0.69482429,
}}
"trial" {
"parameter":  {
"opt_kw.lr":  0.00038292234,
"opt_type":  "Adam"
}
"metric":  {
"accuracy":  0.71642583
}}
```

# Optformer

- Assume you have *a lot* of offline evaluation of tuning runs
- One could train a foundational model for HPO!



**Database of HPO runs containing**

- Metadata of problem studied (Convolution? Tabular?)
- Search space considered
- HPO method used
- List of hyperparameters evaluated and blackbox accuracy

Table 1: Example of a study $(m, \boldsymbol{h})$ with two parameters and two trials. Metadata $m$ appears in blue and history $\boldsymbol{h}$ in purple.


"name": "convnet on cifar10",
"metric": "accuracy",
"goal": "MAXIMIZE",
"algorithm": "random_search",
"parameter": {
"name": "opt_kw.lr",
"type": "DOUBLE",
"min_value": 1e-6,
"max_value": 1e-2,
"scale_type": "LOG"
}
"parameter": {
"name": "opt_type",
"type": "CATEGORICAL",
"categories": ["SGD", "Adam"],
}
"trial" {
"parameter": {
"opt_kw.lr": 0.0021237573,
"opt_type": "SGD"
}
"metric": {
"accuracy": 0.69482429,
}}
"trial" {
"parameter": {
"opt_kw.lr": 0.00038292234,
"opt_type": "Adam"
}
"metric": {
"accuracy": 0.71642583
}}

# Optformer

- Assume you have *a lot* of offline evaluation of tuning runs
- One could train a foundational model for HPO!

**Database of HPO runs containing**

- Metadata of problem studied (Convolution? Tabular?)
- Search space considered
- HPO method used
- List of hyperparameters evaluated and blackbox accuracy

🤔 Can you think about one strategy?

Table 1: Example of a study $(m, \boldsymbol{h})$ with two parameters and two trials. Metadata $m$ appears in blue and history $\boldsymbol{h}$ in purple.


```
"name": "convnet on cifar10",
"metric": "accuracy",
"goal": "MAXIMIZE",
"algorithm": "random_search",
"parameter": {
"name": "opt_kw.lr",
"type": "DOUBLE",
"min_value": 1e-6,
"max_value": 1e-2,
"scale_type": "LOG"
}
"parameter": {
"name": "opt_type",
"type": "CATEGORICAL",
"categories": ["SGD", "Adam"],
}
"trial" {
"parameter": {
"opt_kw.lr": 0.0021237573,
"opt_type": "SGD"
}
"metric": {
"accuracy": 0.69482429,
}}
"trial" {
"parameter": {
"opt_kw.lr": 0.00038292234,
"opt_type": "Adam"
}
"metric": {
"accuracy": 0.71642583
}}
```

# Optformer

- Assume you have *a lot* of offline evaluation of tuning runs
- One could train a foundational model for HPO!
- Optformer proposes a transformer trained to predict the next token of HPO runs

**Database of HPO runs containing**

- Metadata of problem studied (Convolution? Tabular?)
- Search space considered
- HPO method used
- List of hyperparameters evaluated and blackbox accuracy

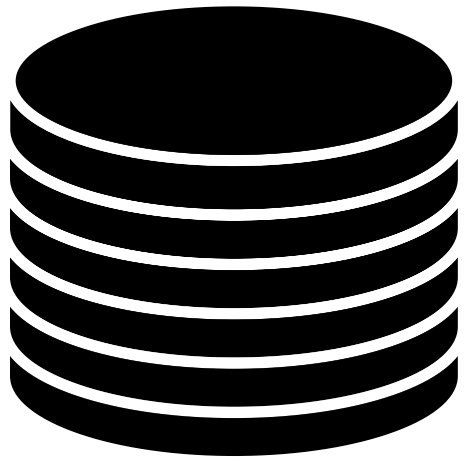
🤔 Can you think about one strategy?

Table 1: Example of a study $(m, \boldsymbol{h})$ with two parameters and two trials. Metadata $m$ appears in blue and history $\boldsymbol{h}$ in purple.


```
"name": "convnet on cifar10",
"metric": "accuracy",
"goal": "MAXIMIZE",
"algorithm": "random_search",
"parameter": {
"name": "opt_kw.lr",
"type": "DOUBLE",
"min_value": 1e-6,
"max_value": 1e-2,
"scale_type": "LOG"
}
"parameter": {
"name": "opt_type",
"type": "CATEGORICAL",
"categories": ["SGD", "Adam"],
}
"trial" {
"parameter": {
"opt_kw.lr": 0.0021237573,
"opt_type": "SGD"
}
"metric": {
"accuracy": 0.69482429,
}}
"trial" {
"parameter": {
"opt_kw.lr": 0.00038292234,
"opt_type": "Adam"
}
"metric": {
"accuracy": 0.71642583
}}
```

# Optformer

- Assume you have *a lot* of offline evaluation of tuning runs
- One could train a foundational model for HPO!
- Optformer proposes a transformer trained to predict the next token of HPO runs
- The model also takes as input metadata for the given task and possibly the HPO method being used



**Database of HPO runs containing**

- Metadata of problem studied (Convolution? Tabular?)
- Search space considered
- HPO method used
- List of hyperparameters evaluated and blackbox accuracy

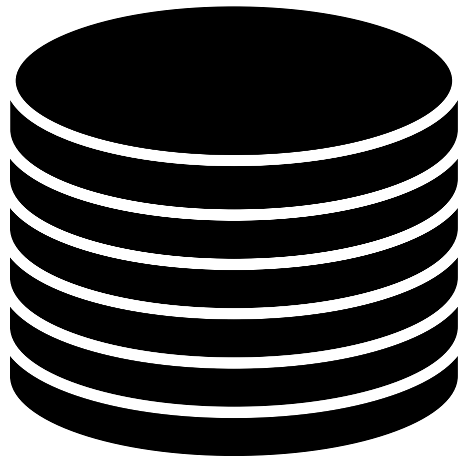
🤔 Can you think about one strategy?

Table 1: Example of a study $(m, \boldsymbol{h})$ with two parameters and two trials. Metadata $m$ appears in blue and history $\boldsymbol{h}$ in purple.


```
"name": "convnet on cifar10",
"metric": "accuracy",
"goal": "MAXIMIZE",
"algorithm": "random_search",
"parameter": {
"name": "opt_kw.lr",
"type": "DOUBLE",
"min_value": 1e-6,
"max_value": 1e-2,
"scale_type": "LOG"
}
"parameter": {
"name": "opt_type",
"type": "CATEGORICAL",
"categories": ["SGD", "Adam"],
}
"trial" {
"parameter": {
"opt_kw.lr": 0.0021237573,
"opt_type": "SGD"
}
"metric": {
"accuracy": 0.69482429,
}}
"trial" {
"parameter": {
"opt_kw.lr": 0.00038292234,
"opt_type": "Adam"
}
"metric": {
"accuracy": 0.71642583
}}
```

# Optformer

- Assume you have *a lot* of offline evaluation of tuning runs
- One could train a foundational model for HPO!
- Optformer proposes a transformer trained to predict the next token of HPO runs
- The model also takes as input metadata for the given task and possibly the HPO method being used
- Trained on 750K tuning runs, each with on average 300 trials



**Database of HPO runs containing**

- Metadata of problem studied (Convolution? Tabular?)
- Search space considered
- HPO method used
- List of hyperparameters evaluated and blackbox accuracy

🤔 Can you think about one strategy?

Table 1: Example of a study $(m, \boldsymbol{h})$ with two parameters and two trials. Metadata $m$ appears in blue and history $\boldsymbol{h}$ in purple.


```
"name": "convnet on cifar10",
"metric": "accuracy",
"goal": "MAXIMIZE",
"algorithm": "random_search",
"parameter": {
"name": "opt_kw.lr",
"type": "DOUBLE",
"min_value": 1e-6,
"max_value": 1e-2,
"scale_type": "LOG"
}
"parameter": {
"name": "opt_type",
"type": "CATEGORICAL",
"categories": ["SGD", "Adam"],
}
"trial" {
"parameter": {
"opt_kw.lr": 0.0021237573,
"opt_type": "SGD"
}
"metric": {
"accuracy": 0.69482429,
}}
"trial" {
"parameter": {
"opt_kw.lr": 0.00038292234,
"opt_type": "Adam"
}
"metric": {
"accuracy": 0.71642583
}}
```

# Optformer

- Optformer proposes a transformer trained to predict the next token of HPO runs

- The model also takes as input metadata for the given task and possibly the HPO method being used

- Trained on 750K tuning runs, each with on average 300 trials

# Optformer

- Optformer proposes a transformer trained to predict the next token of HPO runs
- The model also takes as input metadata for the given task and possibly the HPO method being used
- Trained on 750K tuning runs, each with on average 300 trials

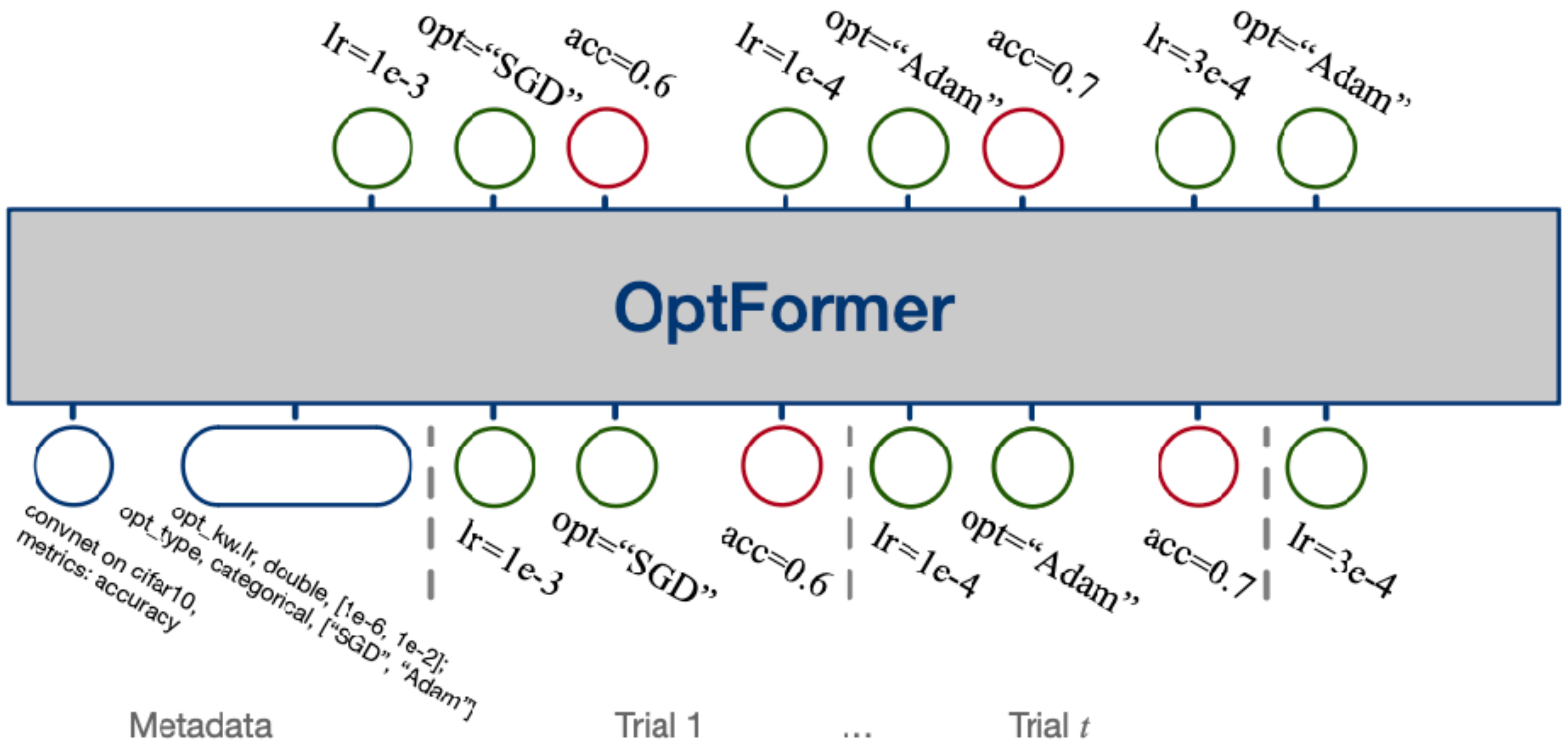
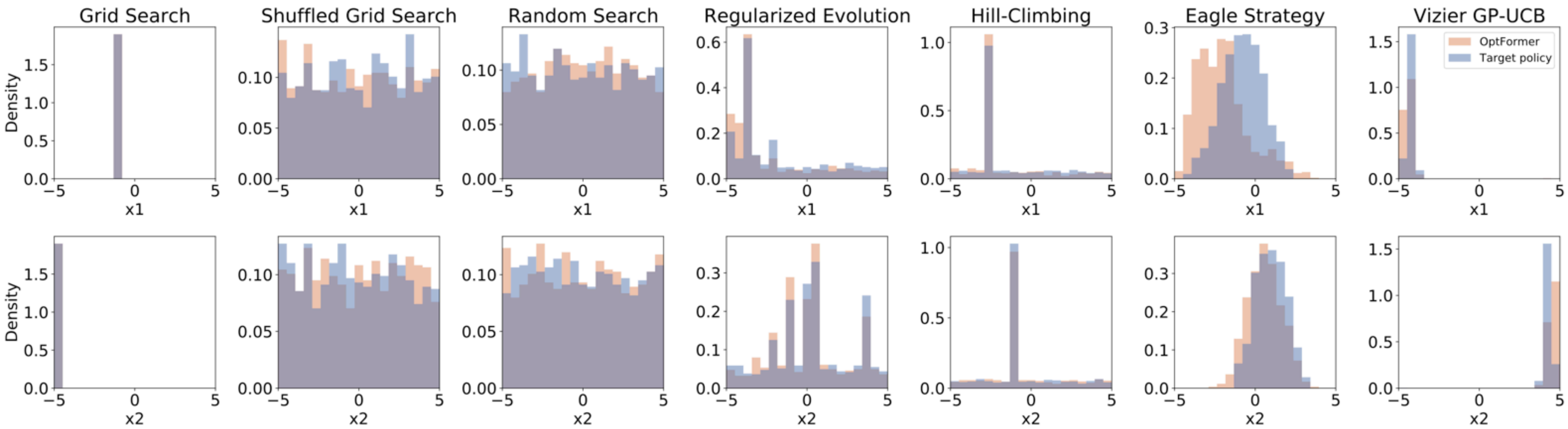



Figure 1: Illustration of the OptFormer model over a hyperparameter optimization trajectory. It is trained to predict both hyperparameter suggestions (in green) and response function values (in red).

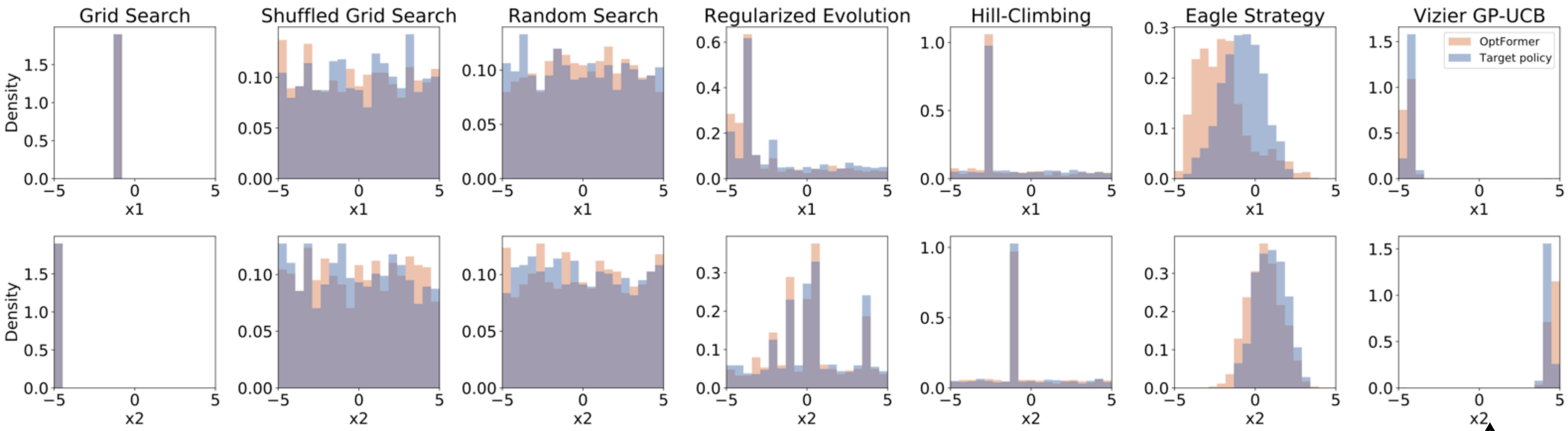
# Optformer

## Imitating other HPO methods



# Optformer

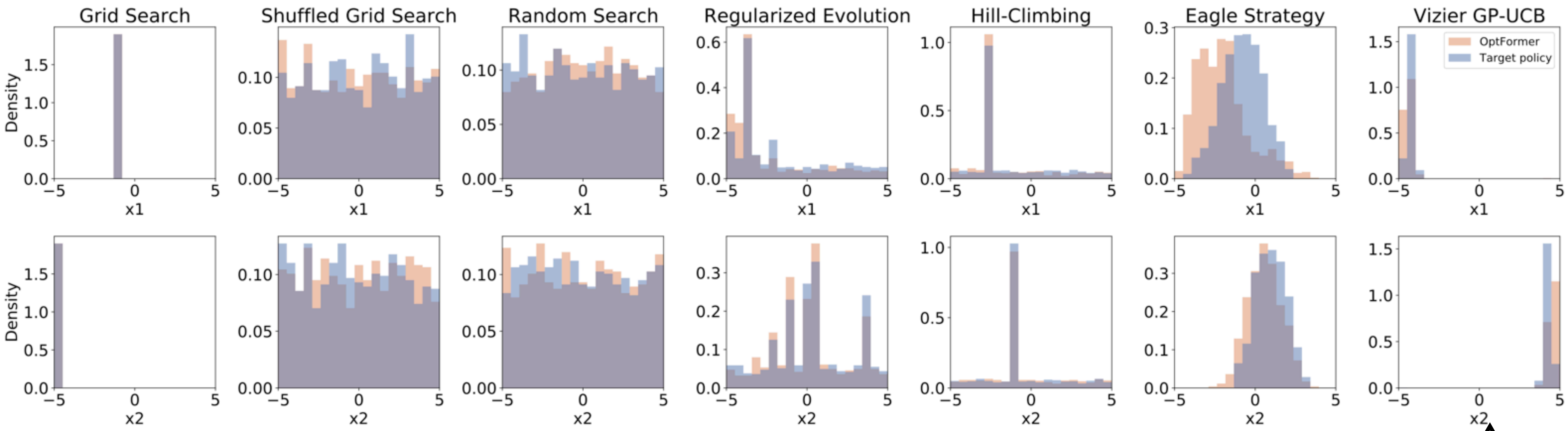
## Imitating other HPO methods




The method can imitate different HPO strategy!

# Optformer

## Imitating other HPO methods




🤔 This is an impressive result! some methods are complex, for instance GP requires $\mathcal{O}(n^3)$ operations to select the next candidate given $n$ previous observations

The method can imitate different HPO strategy!

# Optformer

## Outperforming other HPO methods

# Optformer

## Outperforming other HPO methods

- The method learn a predictive model able to simulate multiple HPO methods

# Optformer

## Outperforming other HPO methods

- The method learn a predictive model able to simulate multiple HPO methods
- The method can also select what is the best hyperparameter to evaluate…

# Optformer

## Outperforming other HPO methods

- The method learn a predictive model able to simulate multiple HPO methods
- The method can also select what is the best hyperparameter to evaluate…
- … by sampling many hyperparameter and use the model prediction to select the best one (EI)

# Optformer

## Outperforming other HPO methods

- The method learn a predictive model able to simulate multiple HPO methods
- The method can also select what is the best hyperparameter to evaluate…
- … by sampling many hyperparameter and use the model prediction to select the best one (EI)

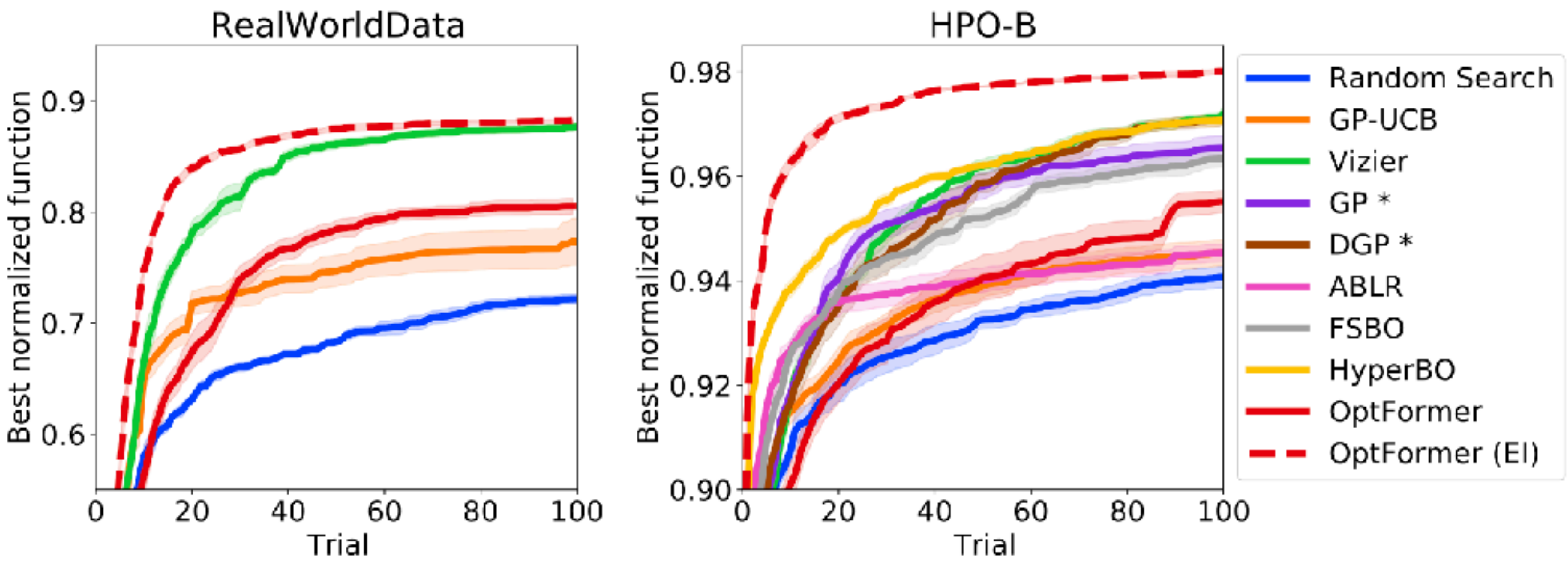



Figure 4: Higher is better. Best normalized function value averaged over 16 RealWorldData test functions (left) and over 86 HPO-B test functions (right) with 1-std confidence interval from 5 runs. GP* and DGP* results are provided by [5]. The transfer learning methods ABLR, FSBO and HyperBO cannot be applied to RealWorldData.

# Optformer

## Outperforming other HPO methods

- The method learn a predictive model able to simulate multiple HPO methods
- The method can also select what is the best hyperparameter to evaluate…
- … by sampling many hyperparameter and use the model prediction to select the best one (EI)

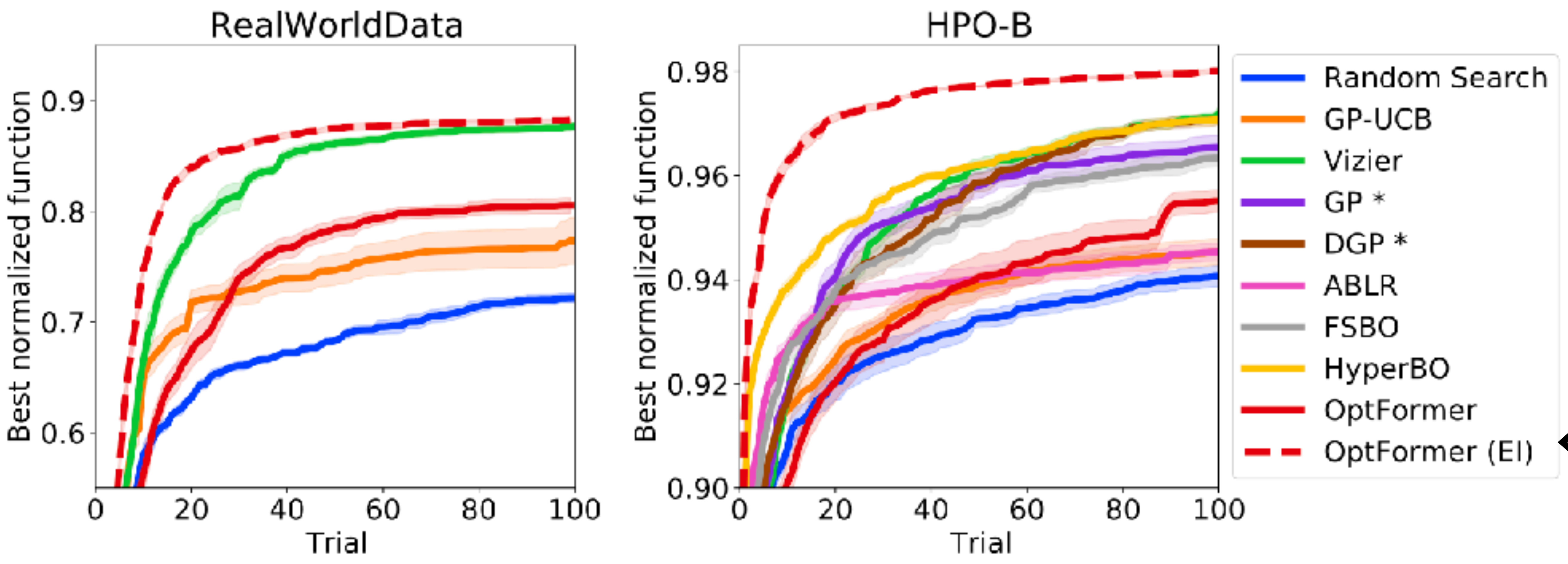



Figure 4: Higher is better. Best normalized function value averaged over 16 RealWorldData test functions (left) and over 86 HPO-B test functions (right) with 1-std confidence interval from 5 runs. GP* and DGP* results are provided by [5]. The transfer learning methods ABLR, FSBO and HyperBO cannot be applied to RealWorldData.

Outperforms method considered in both Google data and public dataset considered ✳

# Optformer

## Outperforming other HPO methods

- The method learn a predictive model able to simulate multiple HPO methods
- The method can also select what is the best hyperparameter to evaluate…
- … by sampling many hyperparameter and use the model prediction to select the best one (EI)

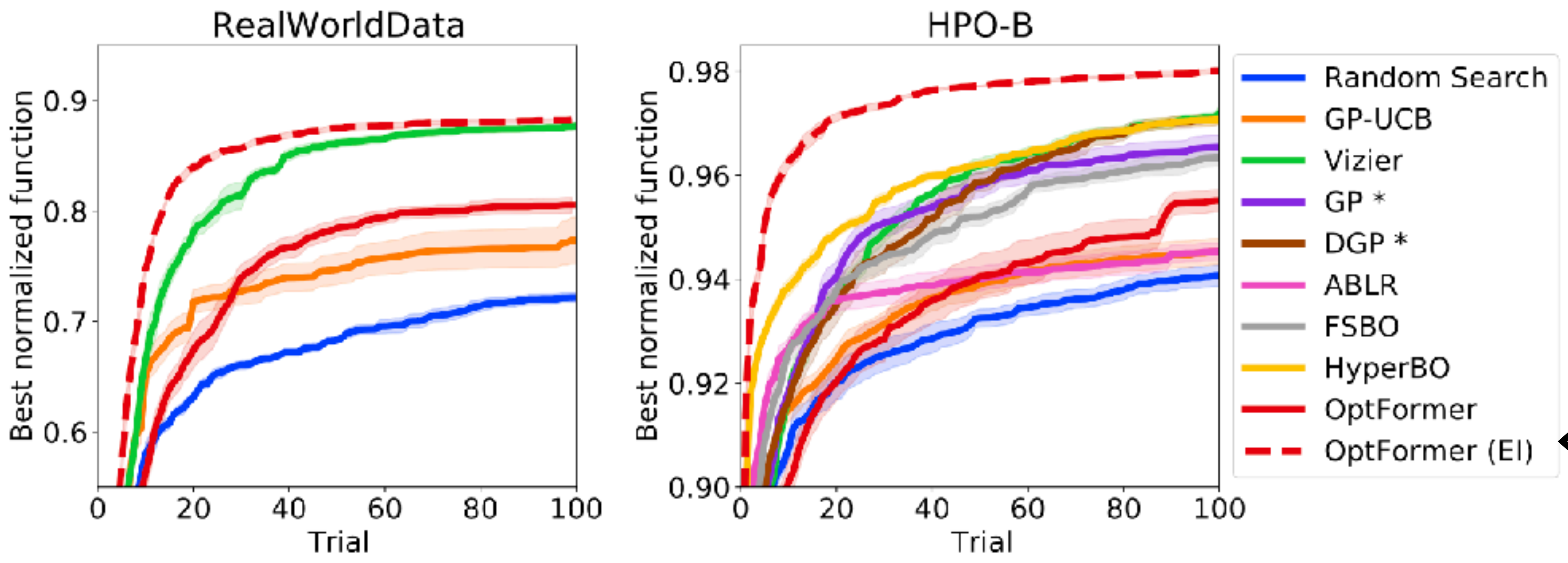



Figure 4: Higher is better. Best normalized function value averaged over 16 RealWorldData test functions (left) and over 86 HPO-B test functions (right) with 1-std confidence interval from 5 runs. GP* and DGP* results are provided by [5]. The transfer learning methods ABLR, FSBO and HyperBO cannot be applied to RealWorldData.

Outperforms method considered in both Google data and public dataset considered ✳

✳ private model and evaluation code

# Application: Improving Tabular prediction with transfer learning

# Tabular prediction

# Tabular prediction

- Tabular prediction: problem definition

# Tabular prediction

- Tabular prediction: problem definition
- Current state of tabular prediction evaluation

# Tabular prediction

- Tabular prediction: problem definition
- Current state of tabular prediction evaluation
- A quick glance at the current SOTA tabular system: AutoGluon

# Tabular prediction

- Tabular prediction: problem definition
- Current state of tabular prediction evaluation
- A quick glance at the current SOTA tabular system: AutoGluon
- Improving AutoGluon with offline evaluations and portfolio learning

# Tabular prediction

```python
import pandas as pd
from autogluon.tabular import TabularPredictor

df_train = pd.read_csv('train.csv')
df_test = pd.read_csv('train.csv')

predictor = TabularPredictor(label='class').fit(df_train)
predictions = predictor.predict(df_test)
```

Tabular prediction API example

# Tabular prediction

- Input: a training data frame, a target column and a training time budget

```python
import pandas as pd
from autogluon.tabular import TabularPredictor

df_train = pd.read_csv('train.csv')
df_test = pd.read_csv('train.csv')

predictor = TabularPredictor(label='class').fit(df_train)
predictions = predictor.predict(df_test)
```

Tabular prediction API example

# Tabular prediction

- Input: a training data frame, a target column and a training time budget
- Output: a predictor able to give predictions given a test dataframe

```
import pandas as pd
from autogluon.tabular import TabularPredictor

df_train = pd.read_csv('train.csv')
df_test = pd.read_csv('train.csv')

predictor = TabularPredictor(label='class').fit(df_train)
predictions = predictor.predict(df_test)
```

Tabular prediction API example

# Tabular prediction

- Input: a training data frame, a target column and a training time budget
- Output: a predictor able to give predictions given a test dataframe
- Metrics:
  - RMSE (regression), log-prob (classification)
  - Prediction latency, memory, …

```
import pandas as pd
from autogluon.tabular import TabularPredictor

df_train = pd.read_csv('train.csv')
df_test = pd.read_csv('train.csv')

predictor = TabularPredictor(label='class').fit(df_train)
predictions = predictor.predict(df_test)
```

Tabular prediction API example

# Tabular prediction

- Input: a training data frame, a target column and a training time budget
- Output: a predictor able to give predictions given a test dataframe
- Metrics:
    - RMSE (regression), log-prob (classification)
    - Prediction latency, memory, …
- Potential candidate: any tabular method and system that returns predictions given the time constrain
    - Can consider multiple model family, ensemble, …

```python
import pandas as pd
from autogluon.tabular import TabularPredictor

df_train = pd.read_csv('train.csv')
df_test = pd.read_csv('train.csv')

predictor = TabularPredictor(label='class').fit(df_train)
predictions = predictor.predict(df_test)
```

Tabular prediction API example

# AutoGluon at a glance

# AutoGluon at a glance

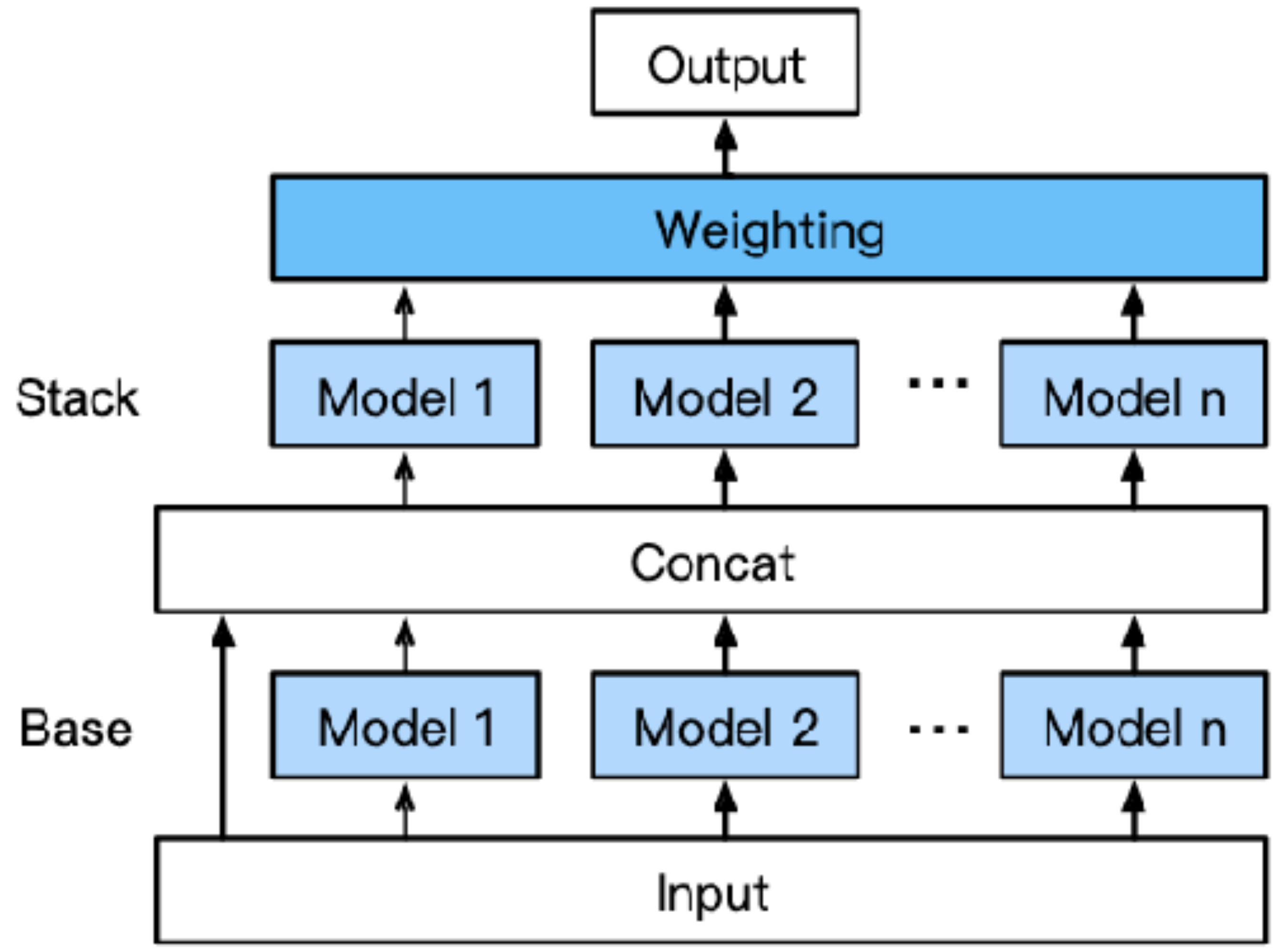



*Figure 2.* AutoGluon's multi-layer stacking strategy, shown here using two stacking layers and $n$ types of base learners.

# AutoGluon at a glance

- AutoGluon recipe:

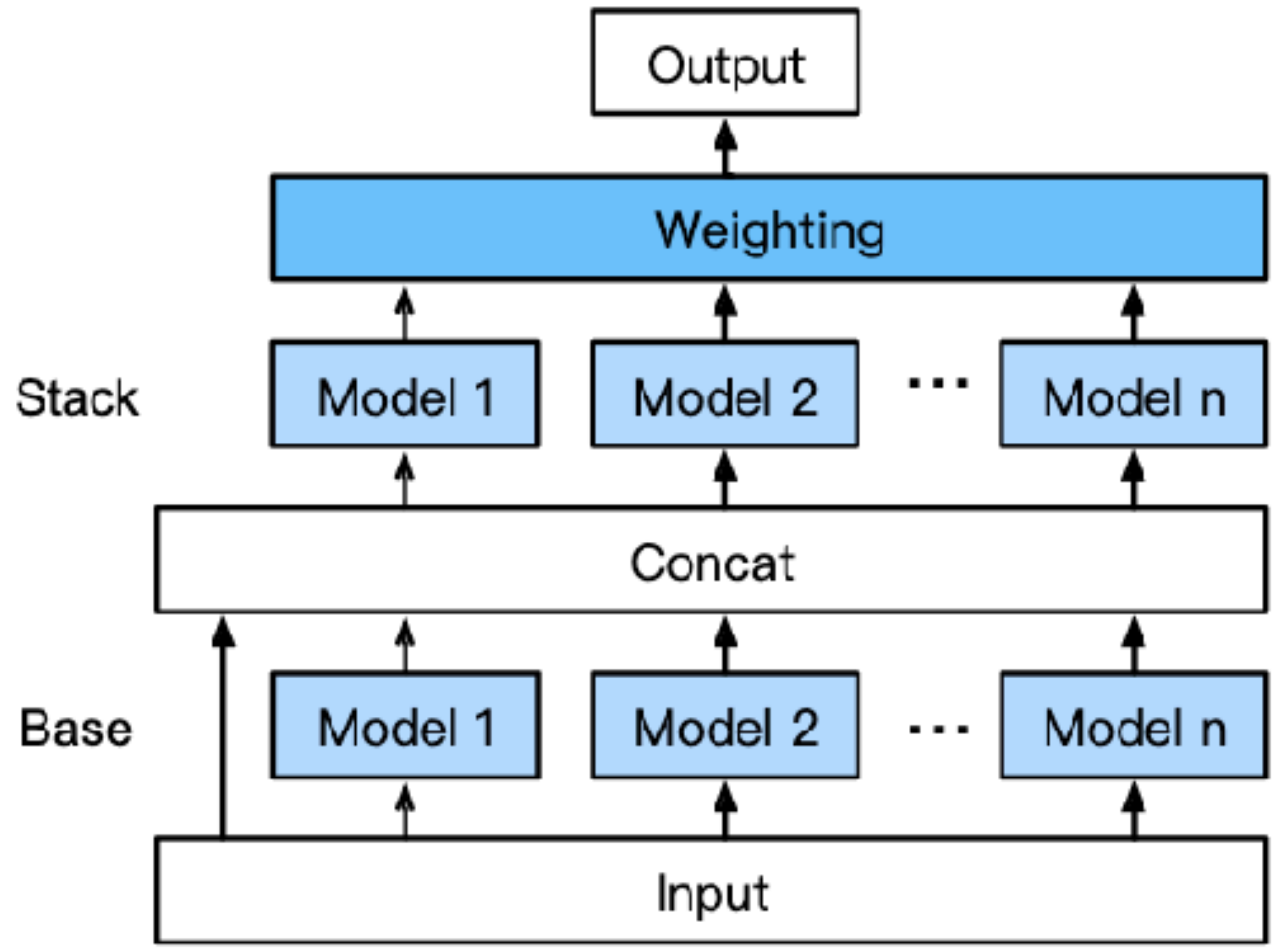



*Figure 2.* AutoGluon's multi-layer stacking strategy, shown here using two stacking layers and $n$ types of base learners.

# AutoGluon at a glance

- AutoGluon recipe:
    - Runs 13 models (KNN, linear, Catboost, LightGBM, MLPs, RandomForest, …) in a first *layer*

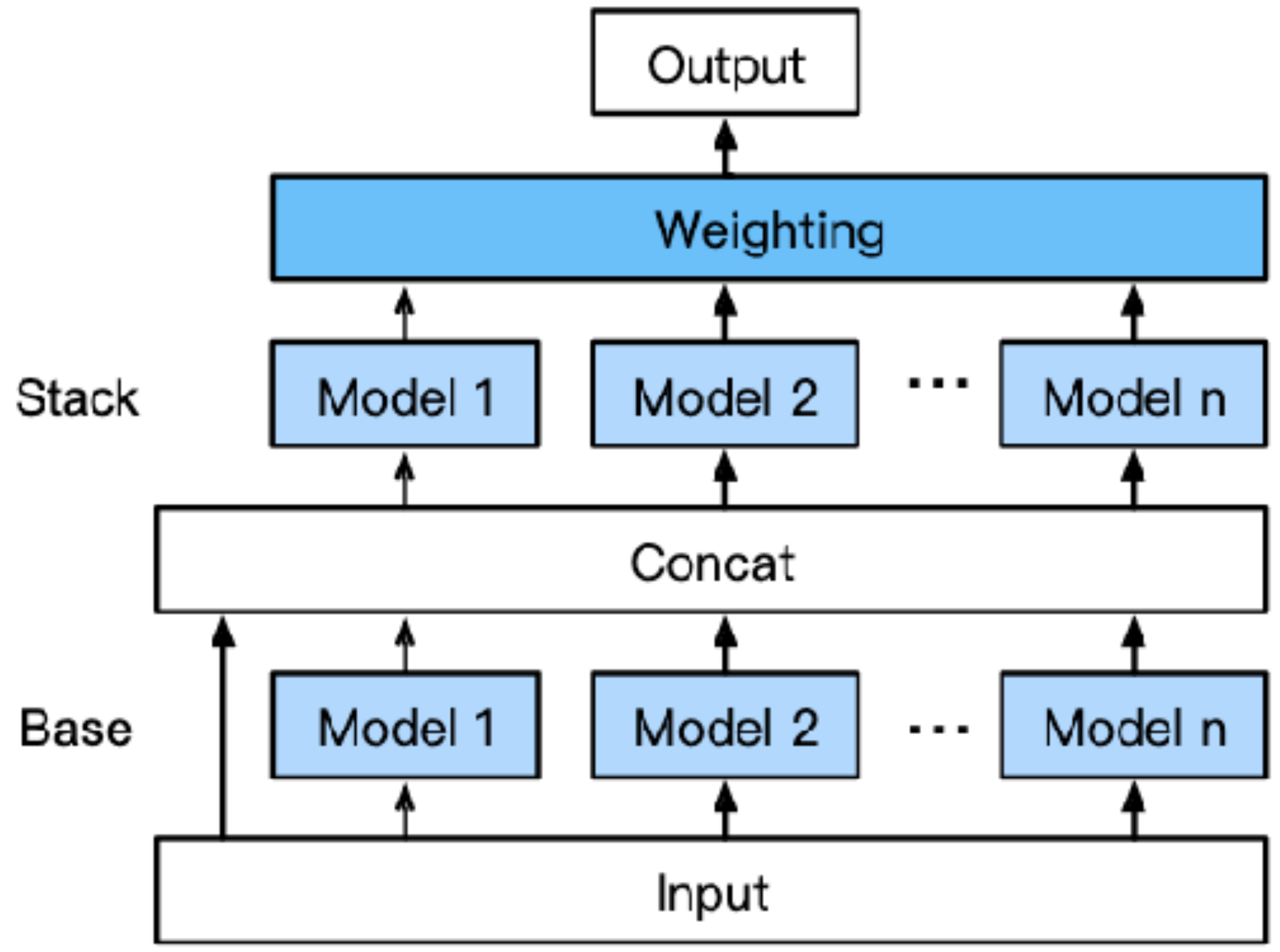



*Figure 2.* AutoGluon's multi-layer stacking strategy, shown here using two stacking layers and $n$ types of base learners.

# AutoGluon at a glance

- AutoGluon recipe:
  - Runs 13 models (KNN, linear, Catboost, LightGBM, MLPs, RandomForest, …) in a first *layer*
  - For each model, Autogluon performs **bagging with out of fold cross-validation**

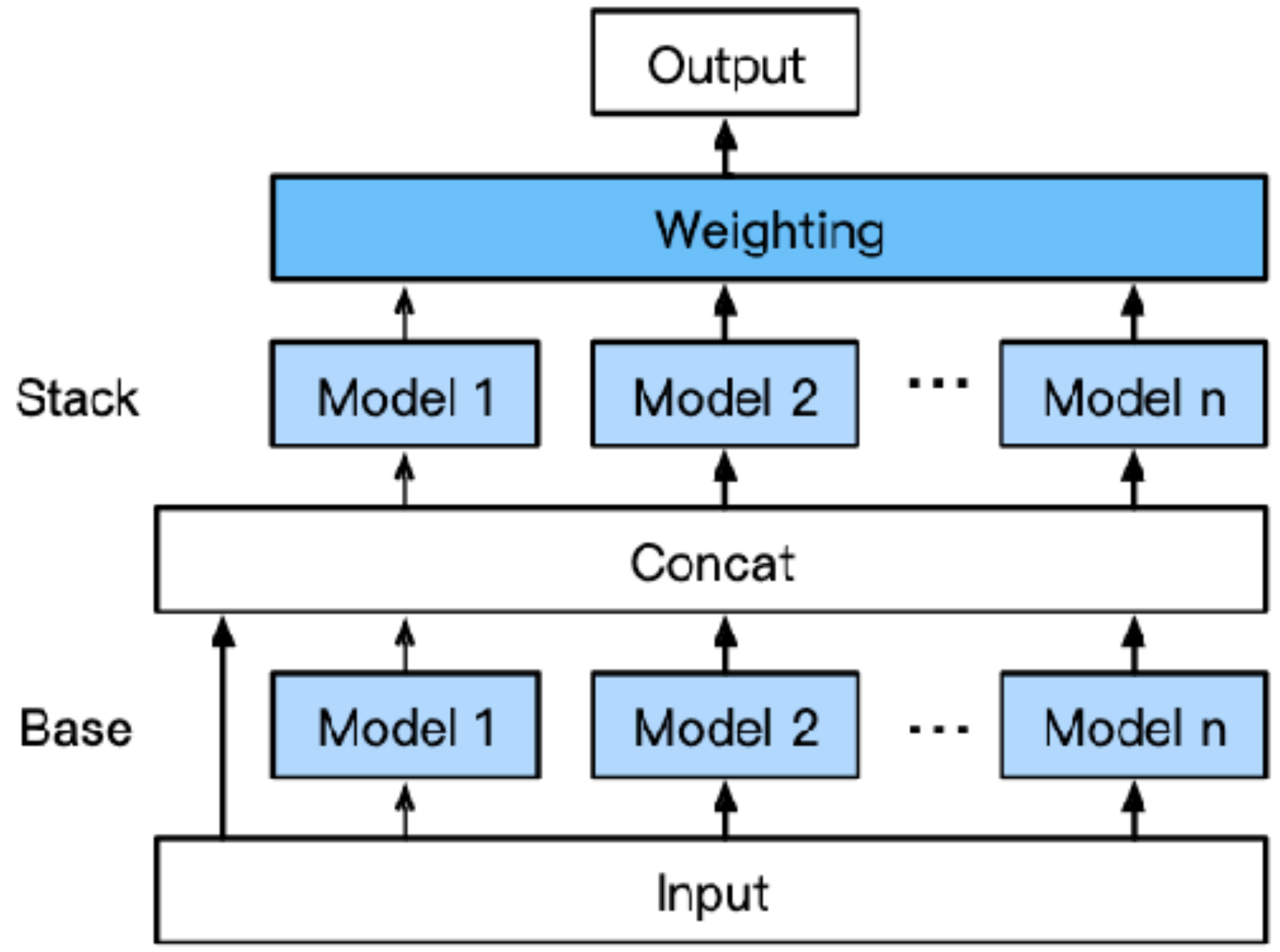



*Figure 2.* AutoGluon's multi-layer stacking strategy, shown here using two stacking layers and $n$ types of base learners.

# AutoGluon at a glance

- AutoGluon recipe:
    - Runs 13 models (KNN, linear, Catboost, LightGBM, MLPs, RandomForest, …) in a first *layer*
    - For each model, Autogluon performs **bagging with out of fold cross-validation**

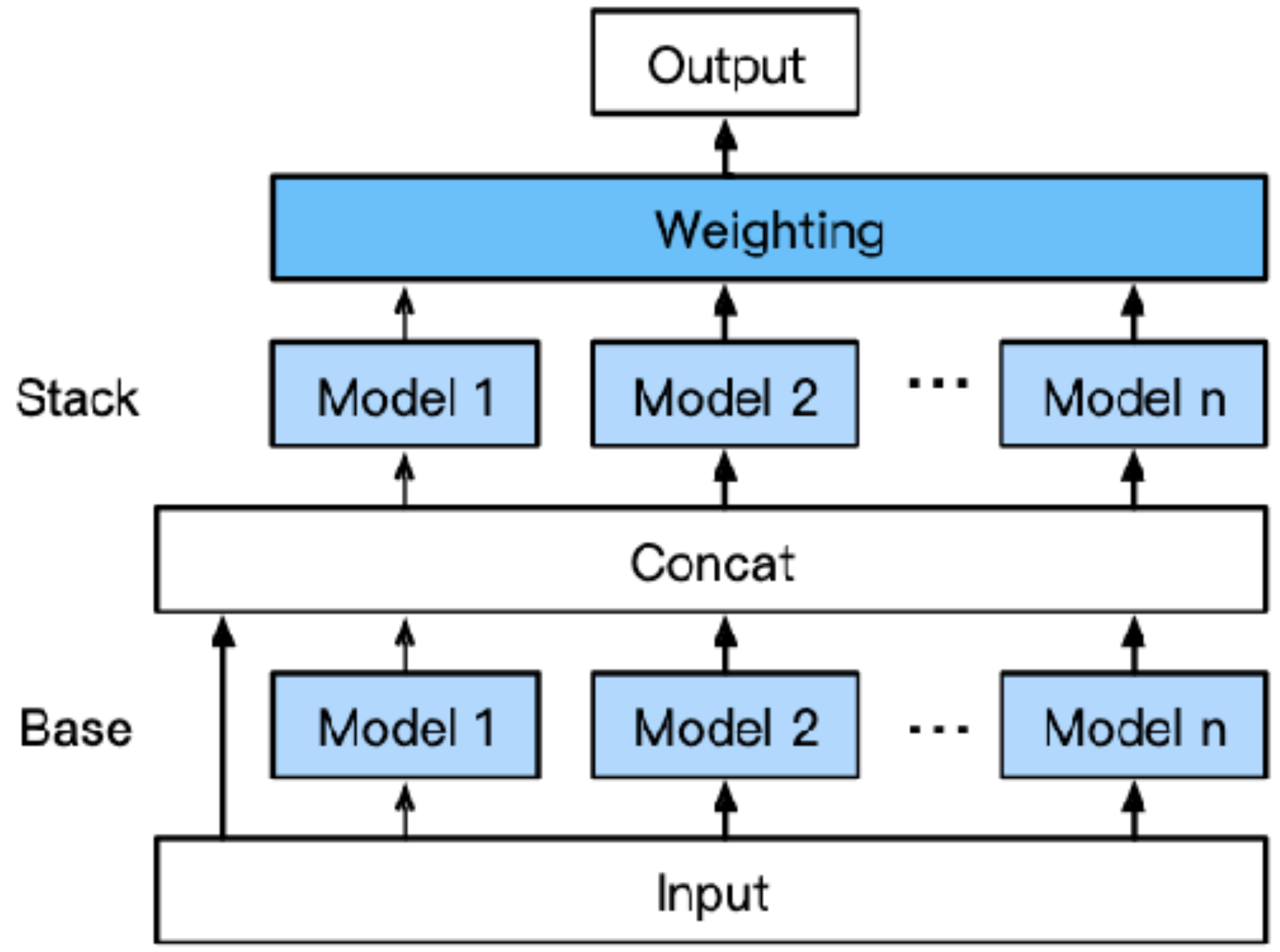



*Figure 2.* AutoGluon's multi-layer stacking strategy, shown here using two stacking layers and $n$ types of base learners.


Out of fold evaluation, image credit: data camp

# AutoGluon at a glance

- AutoGluon recipe:
  - Runs 13 models (KNN, linear, Catboost, LightGBM, MLPs, RandomForest, …) in a first *layer*
  - For each model, Autogluon performs **bagging with out of fold cross-validation**
  - Each model is learned on 8 non-overlapping fold of the data and the predictions are averaged

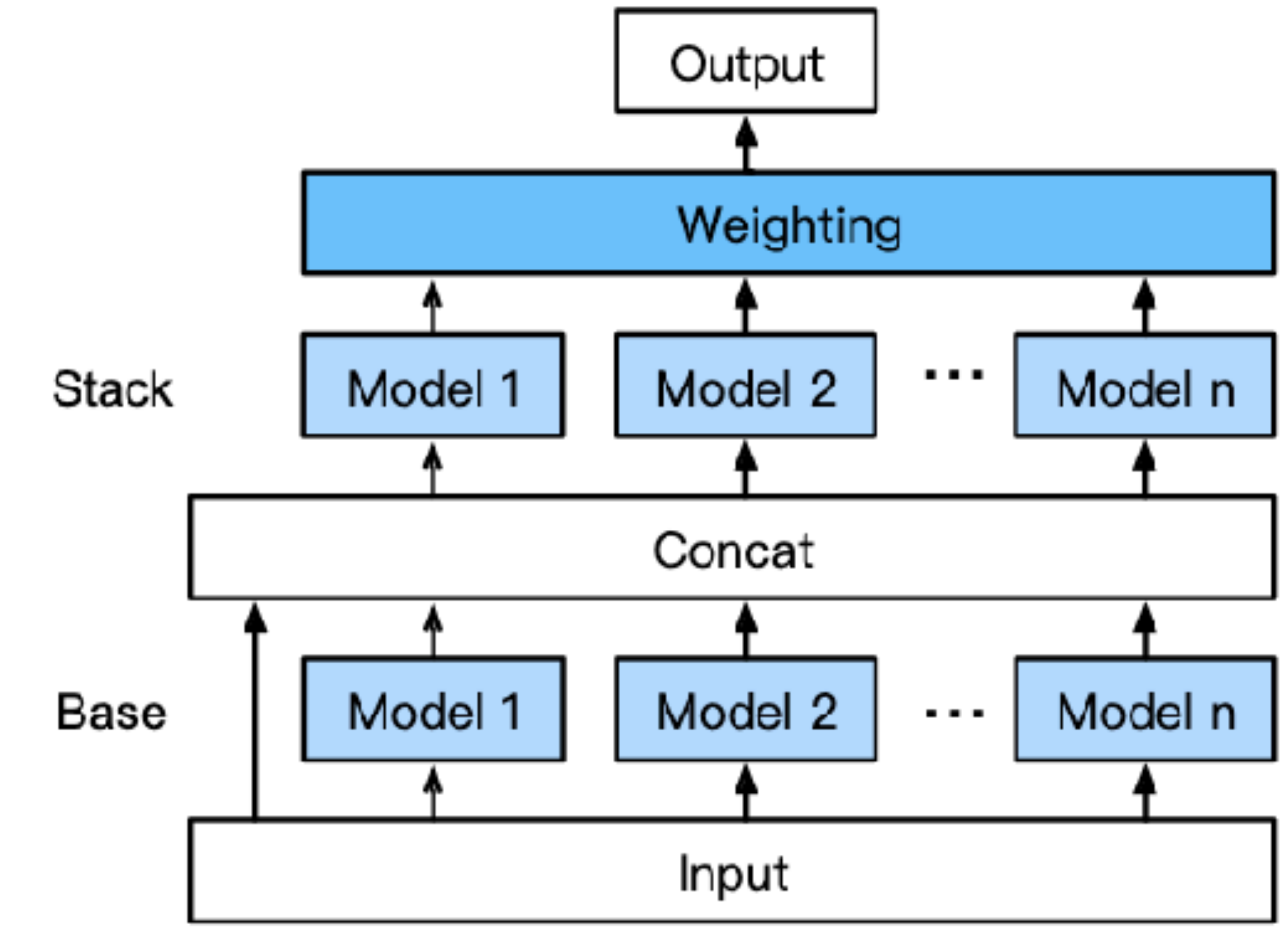
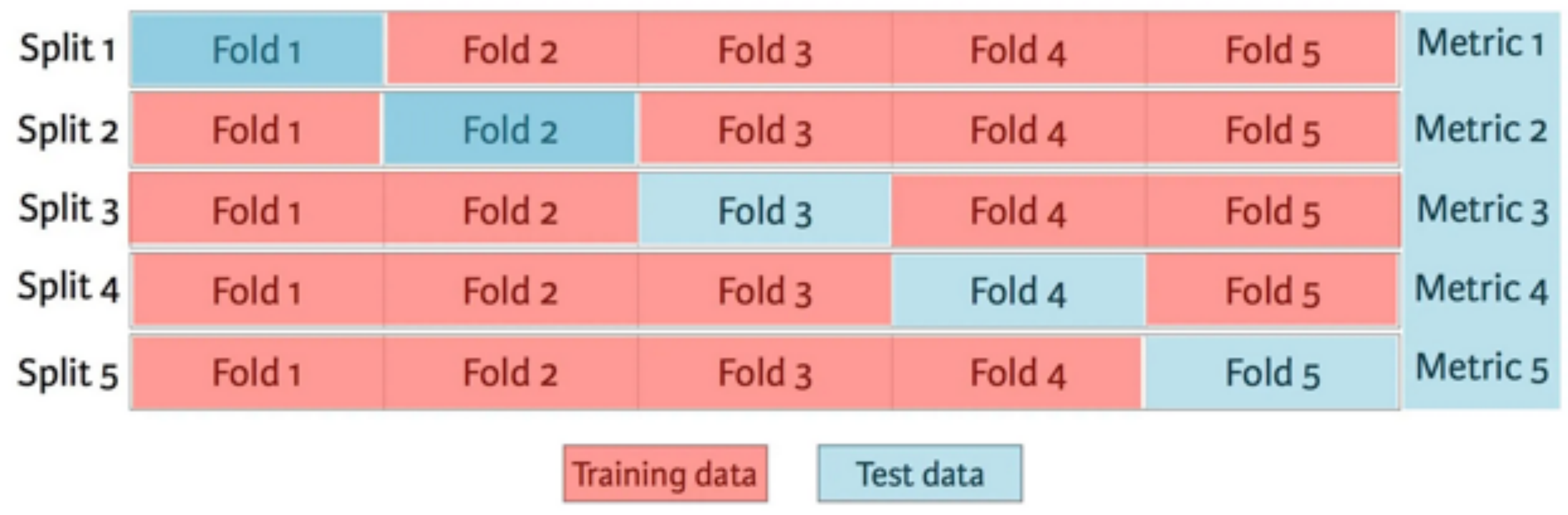



*Figure 2.* AutoGluon's multi-layer stacking strategy, shown here using two stacking layers and $n$ types of base learners.




Out of fold evaluation, image credit: data camp

# AutoGluon at a glance

- AutoGluon recipe:
  - Runs 13 models (KNN, linear, Catboost, LightGBM, MLPs, RandomForest, …) in a first *layer*
  - For each model, Autogluon performs **bagging with out of fold cross-validation**
  - Each model is learned on 8 non-overlapping fold of the data and the predictions are averaged
  - Then perform ***stacking***: e.g. learn the models again while concatenating the predictions of the first *layer* with the original features

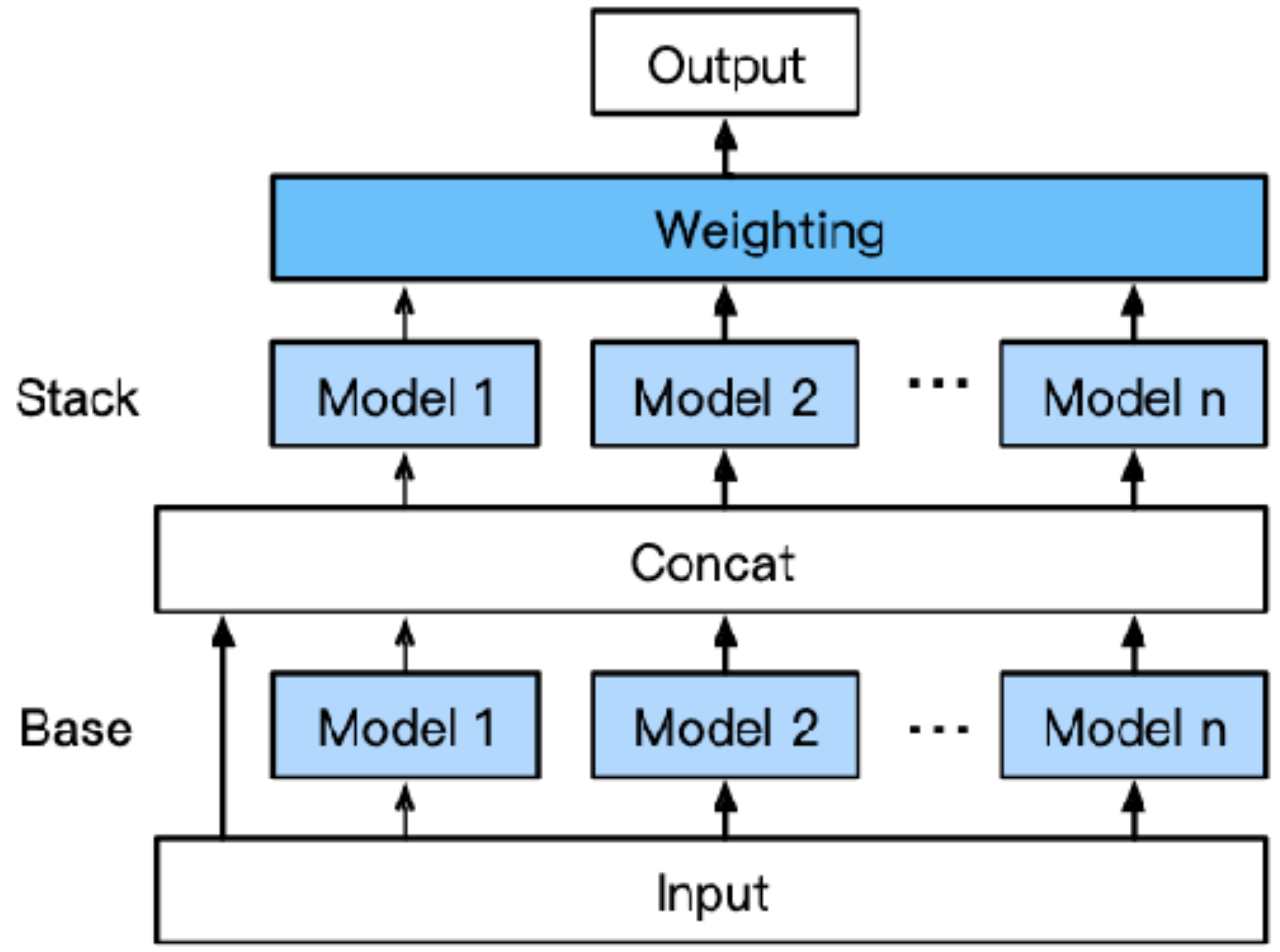
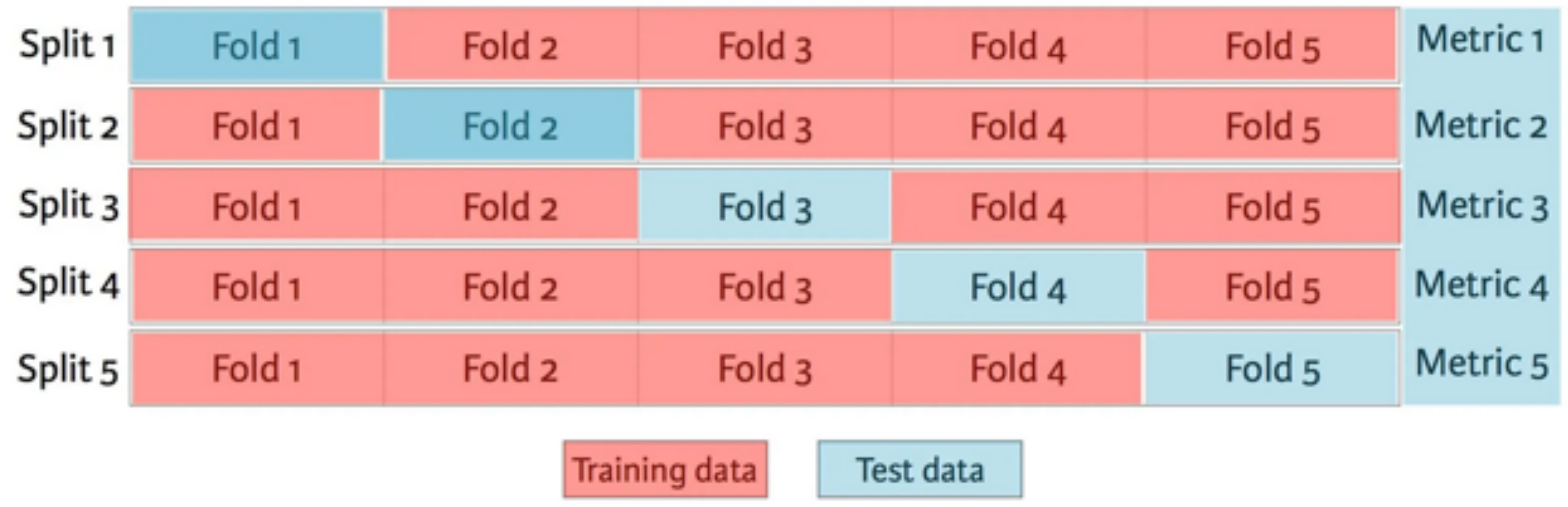



*Figure 2.* AutoGluon's multi-layer stacking strategy, shown here using two stacking layers and $n$ types of base learners.




Out of fold evaluation, image credit: data camp

# AutoGluon at a glance

- AutoGluon recipe:
  - Runs 13 models (KNN, linear, Catboost, LightGBM, MLPs, RandomForest, …) in a first *layer*
  - For each model, Autogluon performs **bagging with out of fold cross-validation**
  - Each model is learned on 8 non-overlapping fold of the data and the predictions are averaged
  - Then perform ***stacking***: e.g. learn the models again while concatenating the predictions of the first *layer* with the original features
  - Then perform ***ensembling***: by estimating the weights on hold-out data

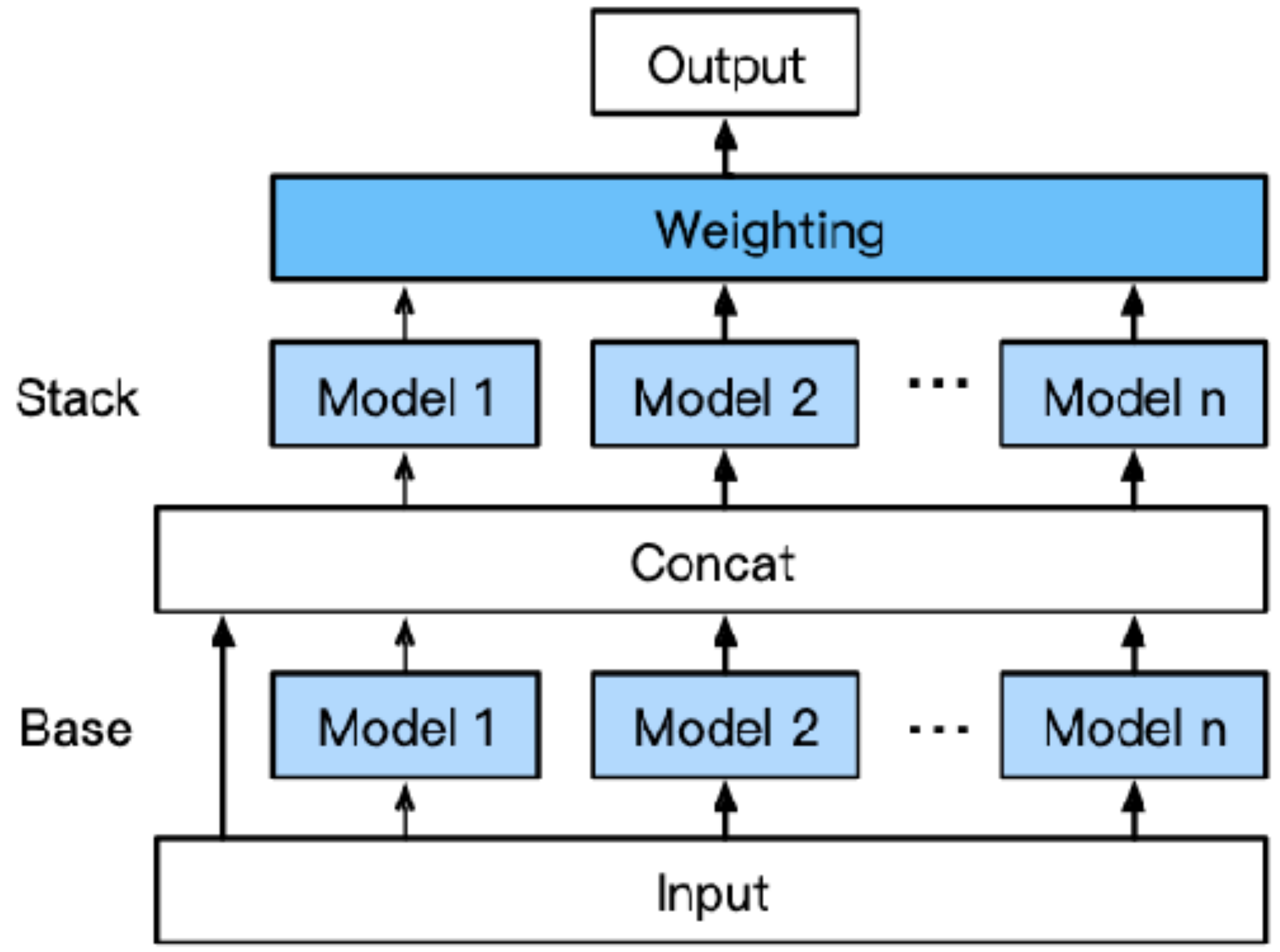
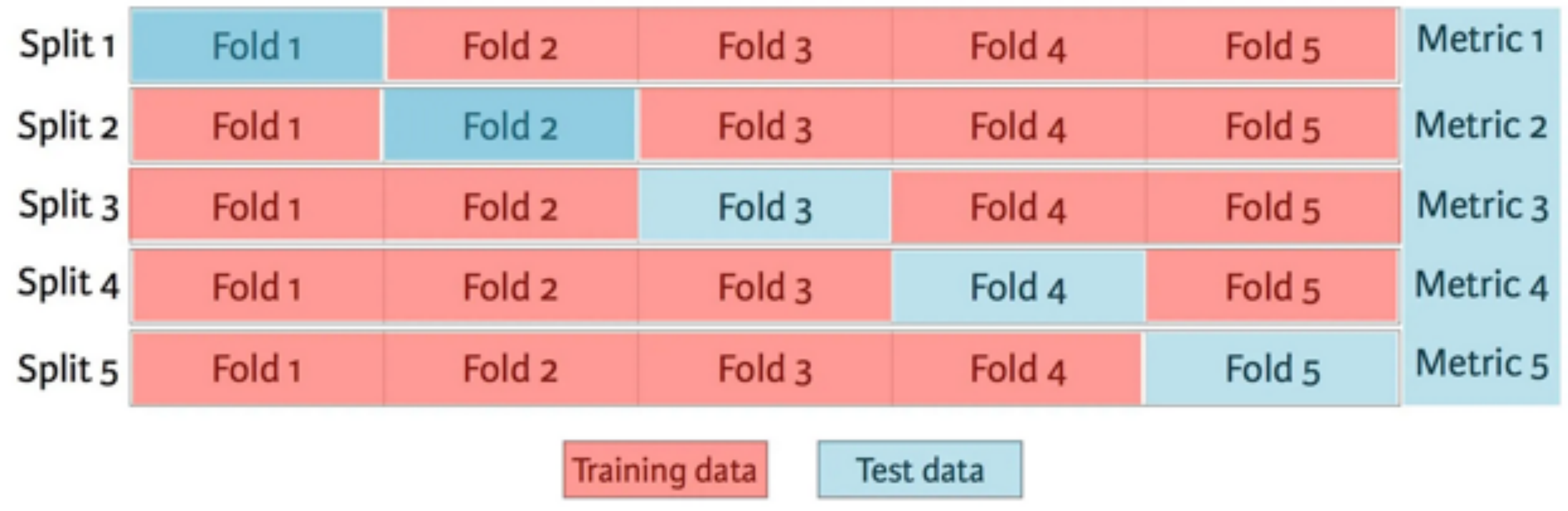



*Figure 2.* AutoGluon's multi-layer stacking strategy, shown here using two stacking layers and $n$ types of base learners.




Out of fold evaluation, image credit: data camp

# AutoGluon at a glance

- AutoGluon recipe:
  - Runs 13 models (KNN, linear, Catboost, LightGBM, MLPs, RandomForest, …) in a first *layer*
  - For each model, Autogluon performs **bagging with out of fold cross-validation**
  - Each model is learned on 8 non-overlapping fold of the data and the predictions are averaged
  - Then perform ***stacking***: e.g. learn the models again while concatenating the predictions of the first *layer* with the original features
  - Then perform ***ensembling***: by estimating the weights on hold-out data
- Lets have a look at Autogluon now!

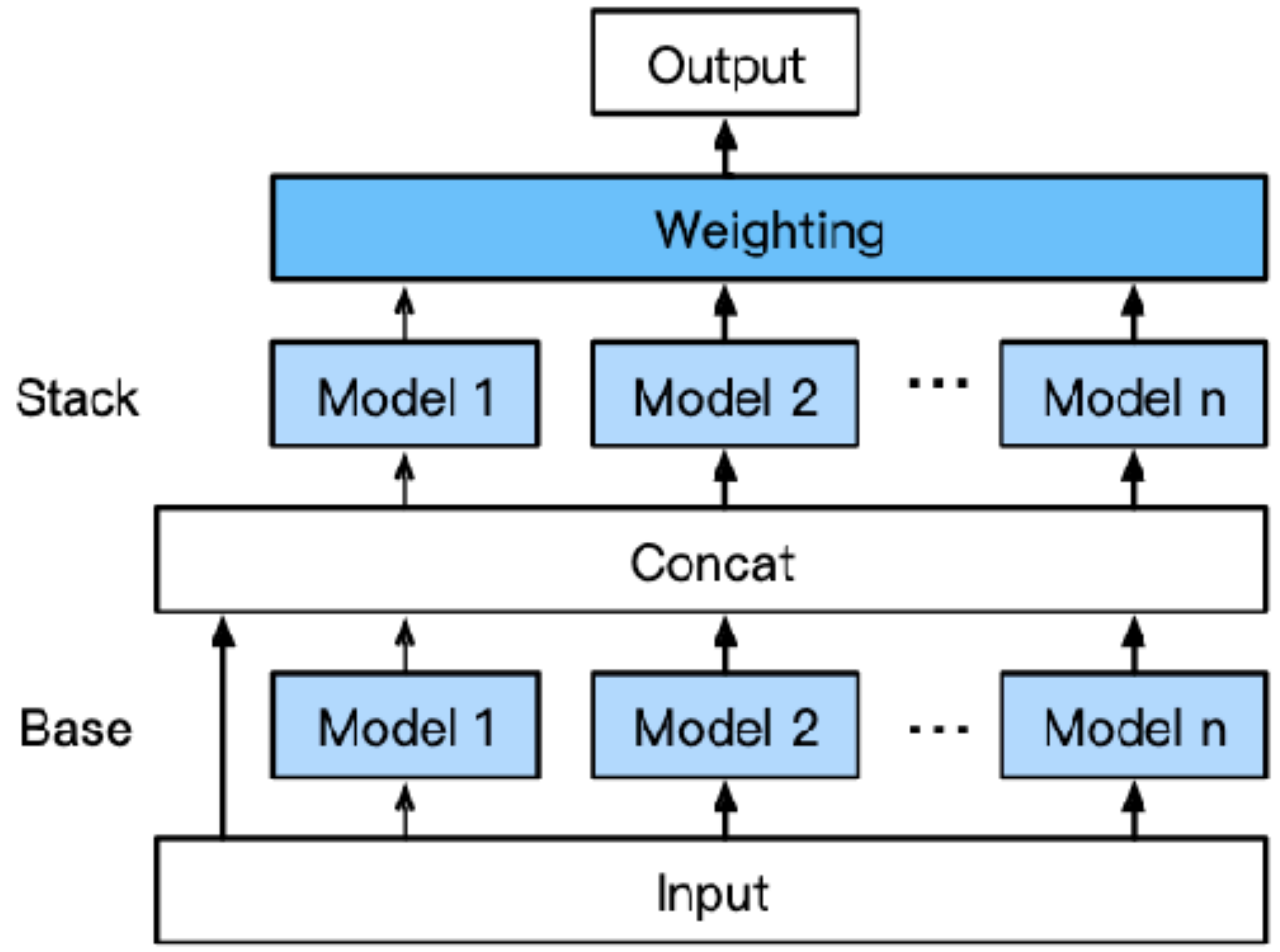
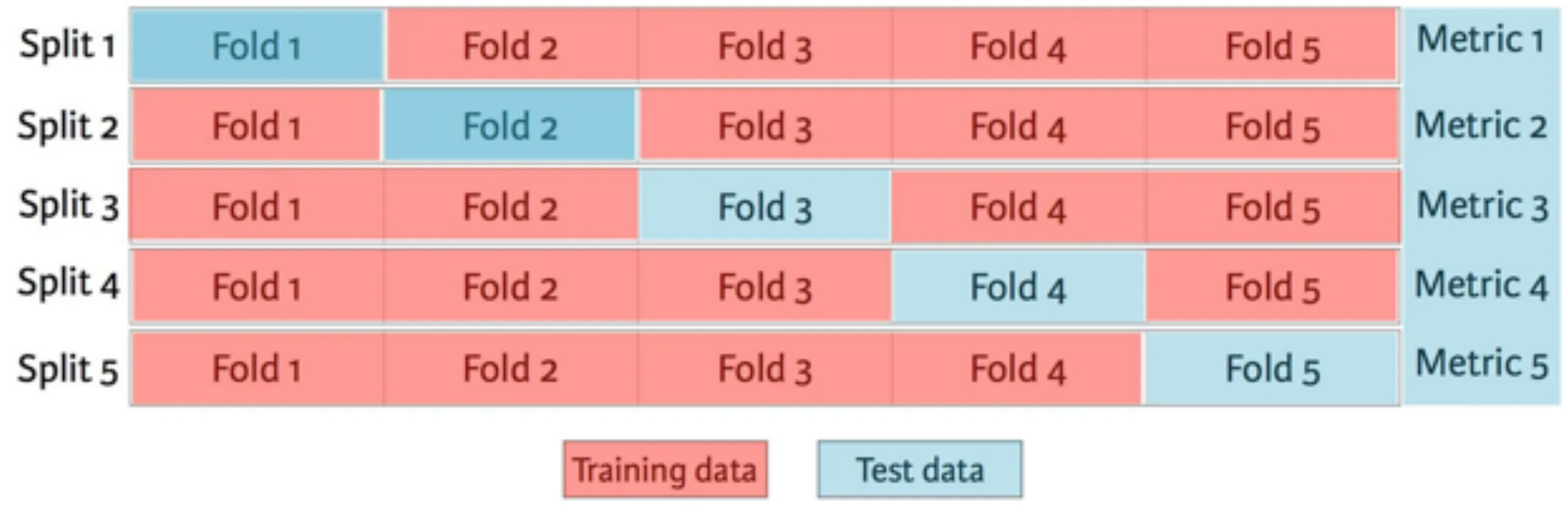
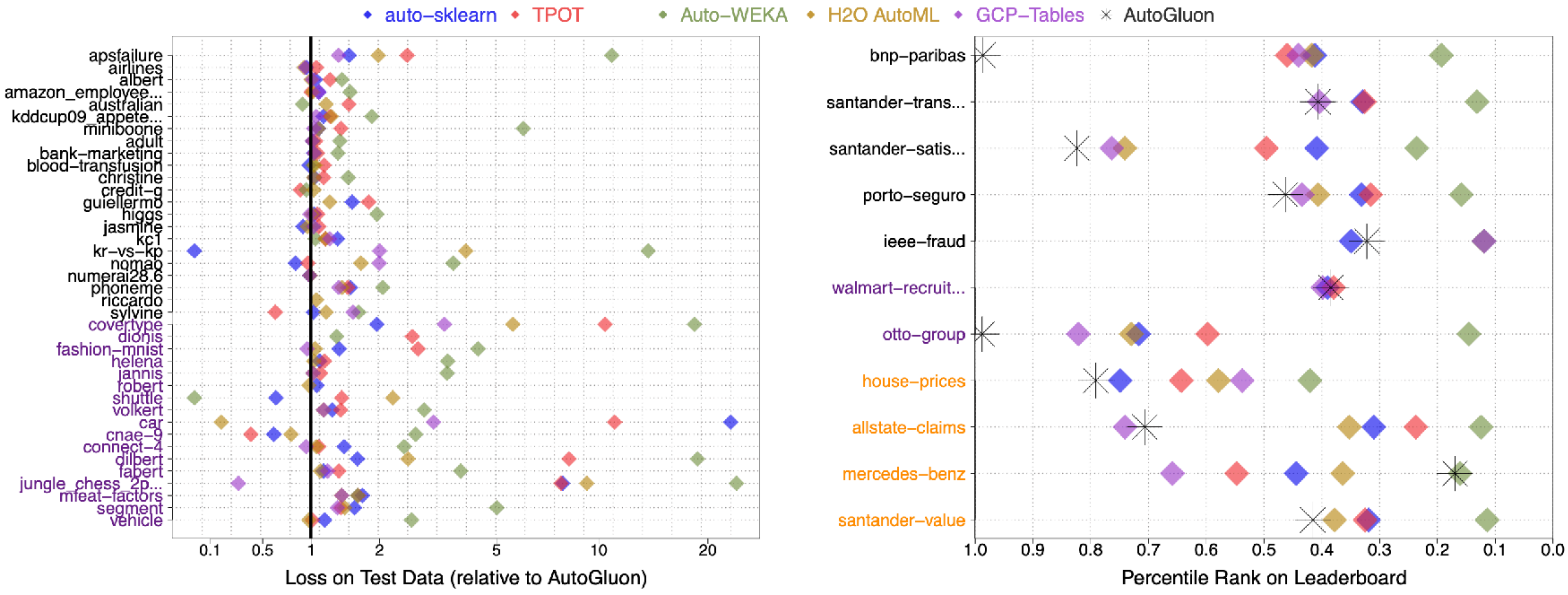



*Figure 2.* AutoGluon's multi-layer stacking strategy, shown here using two stacking layers and $n$ types of base learners.




Out of fold evaluation, image credit: data camp

# What is the best Tabular method?

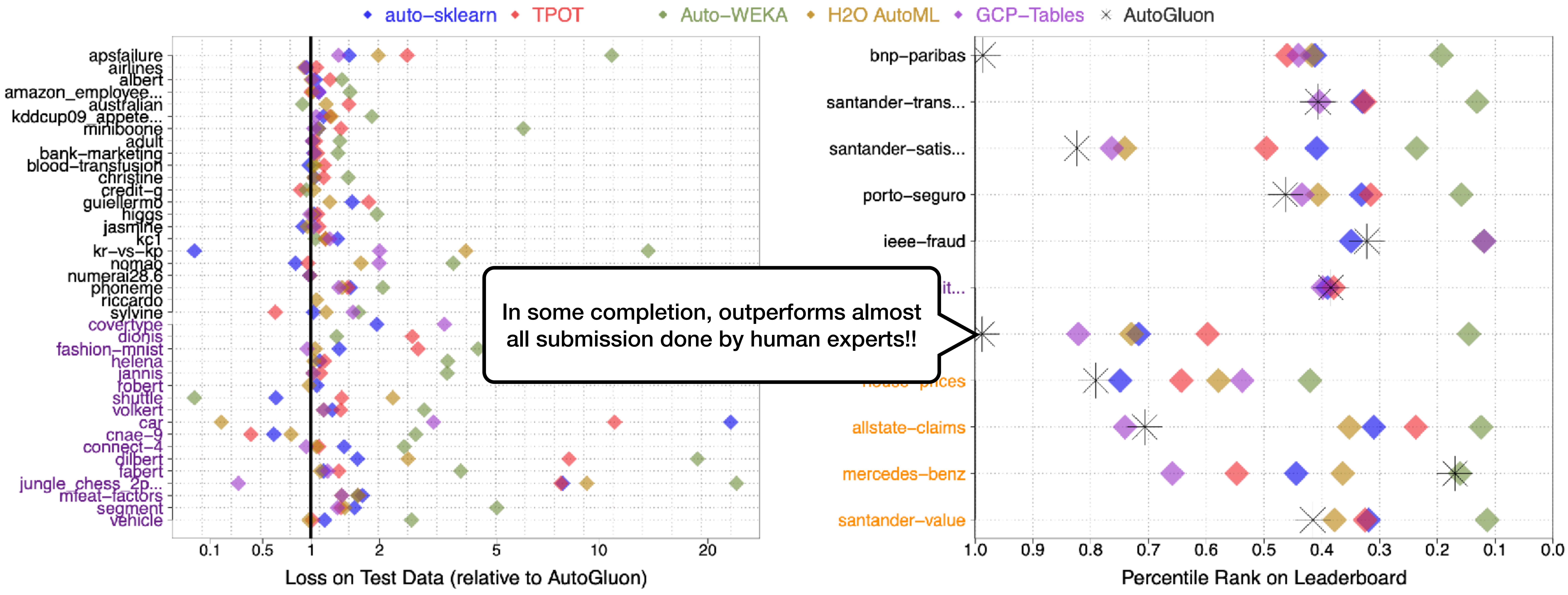



(A) AutoML Benchmark (1h)

(B) Kaggle Benchmark (4h)

Erickson & Mueller et al 2020

# What is the best Tabular method?

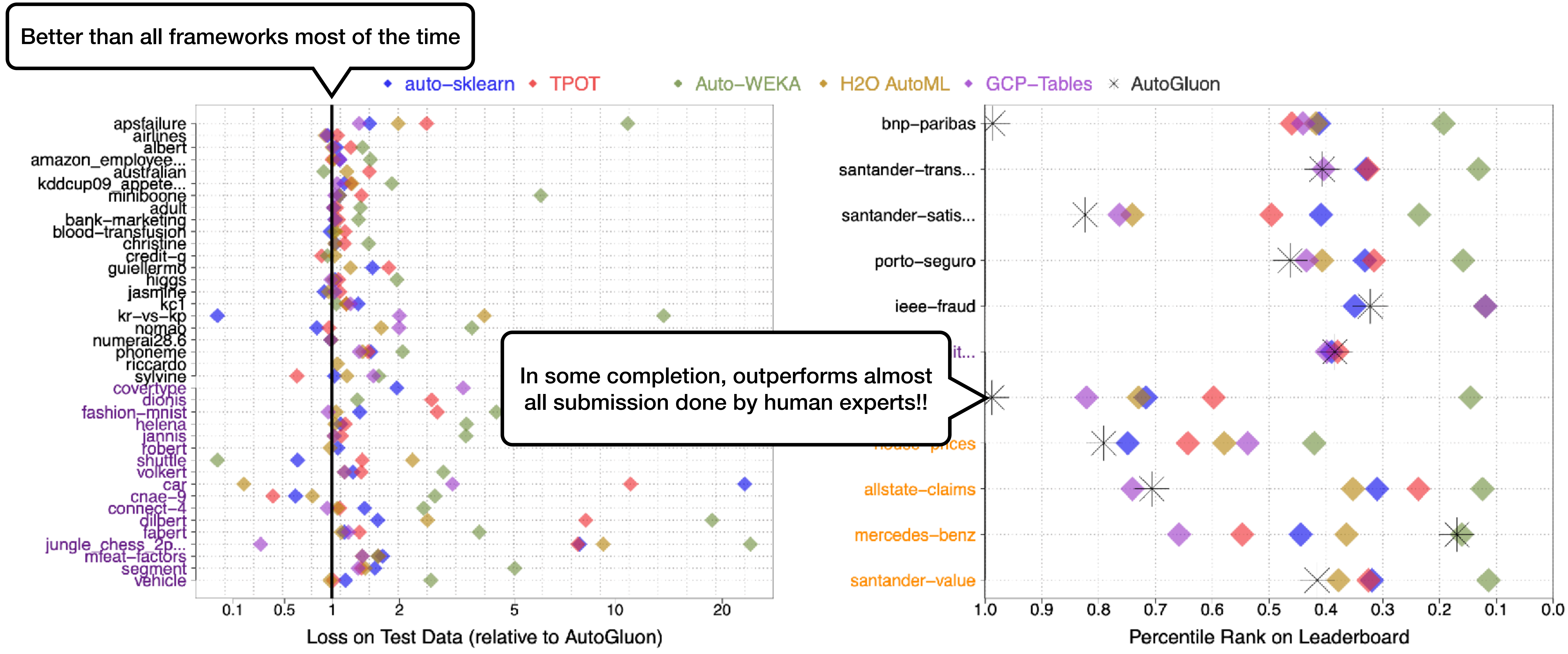



(A) AutoML Benchmark (1h)

(B) Kaggle Benchmark (4h)

Erickson & Mueller et al 2020

# What is the best Tabular method?




**(A)** AutoML Benchmark (1h)

**(B)** Kaggle Benchmark (4h)

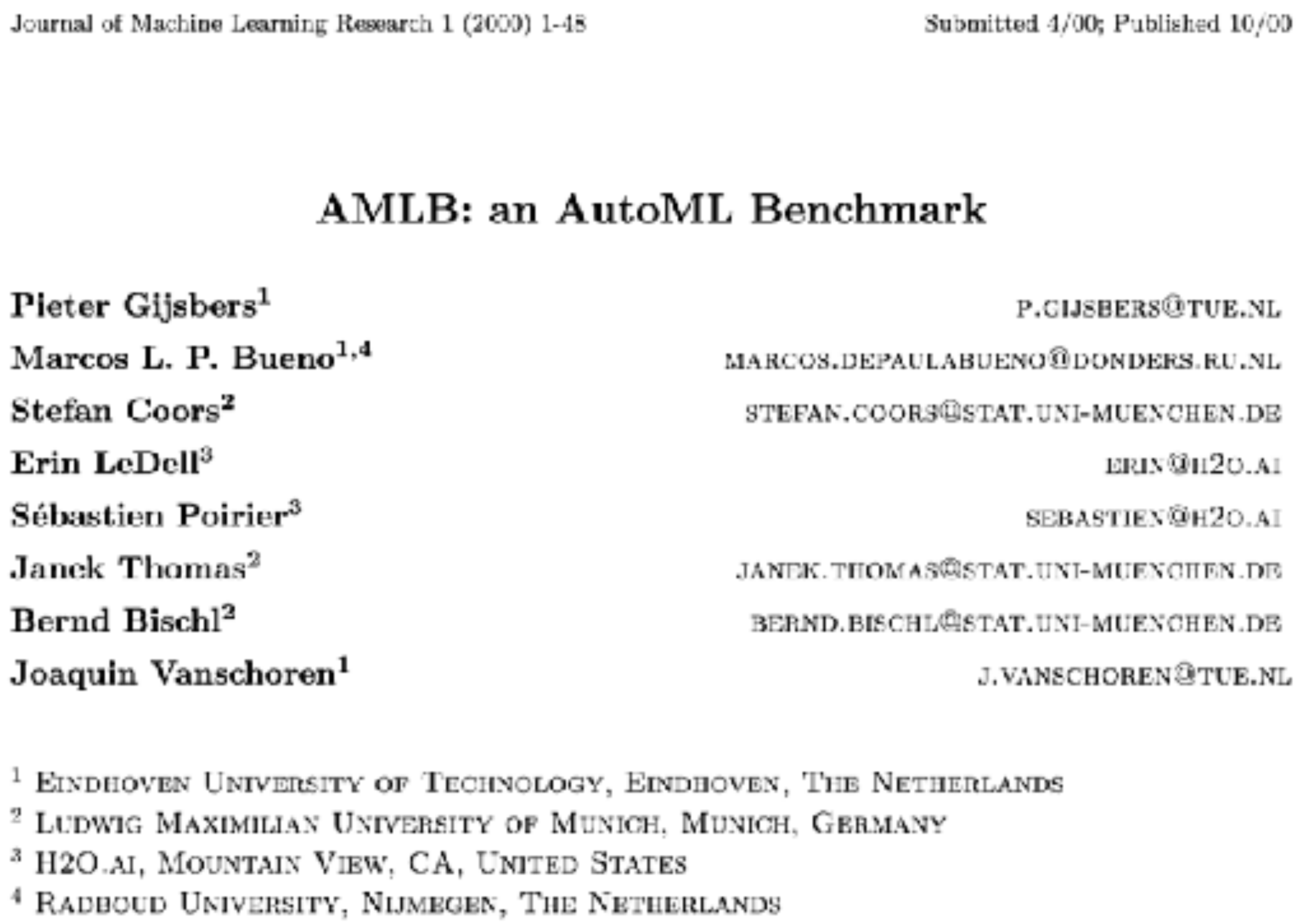
Erickson & Mueller et al 2020

# What is the best Tabular method?

- AutoML Benchmark [Ginsberg et al 2023] considered 71 classification and 33 regression datasets

# What is the best Tabular method?

- AutoML Benchmark [Ginsberg et al 2023] considered 71 classification and 33 regression datasets


Journal of Machine Learning Research 1 (2000) 1-48 Submitted 4/00; Published 10/00

AMLB: an AutoML Benchmark

Pieter Gijsbers[1] P.GIJSBERS@TUE.NL
Marcos L. P. Bueno[1,4] MARCOS.DEPAULABUENO@DONDERS.RU.NL
Stefan Coors[2] STEFAN.COORS@STAT.UNI-MUENCHEN.DE
Erin LeDell[3] ERIN@H2O.AI
Sébastien Poirier[3] SEBASTIEN@H2O.AI
Janek Thomas[2] JANEK.THOMAS@STAT.UNI-MUENCHEN.DE
Bernd Bischl[2] BERND.BISCHL@STAT.UNI-MUENCHEN.DE
Joaquin Vanschoren[1] J.VANSCHOREN@TUE.NL

[1] Eindhoven University of Technology, Eindhoven, The Netherlands
[2] Ludwig Maximilian University of Munich, Munich, Germany
[3] H2O.ai, Mountain View, CA, United States
[4] Radboud University, Nijmegen, The Netherlands

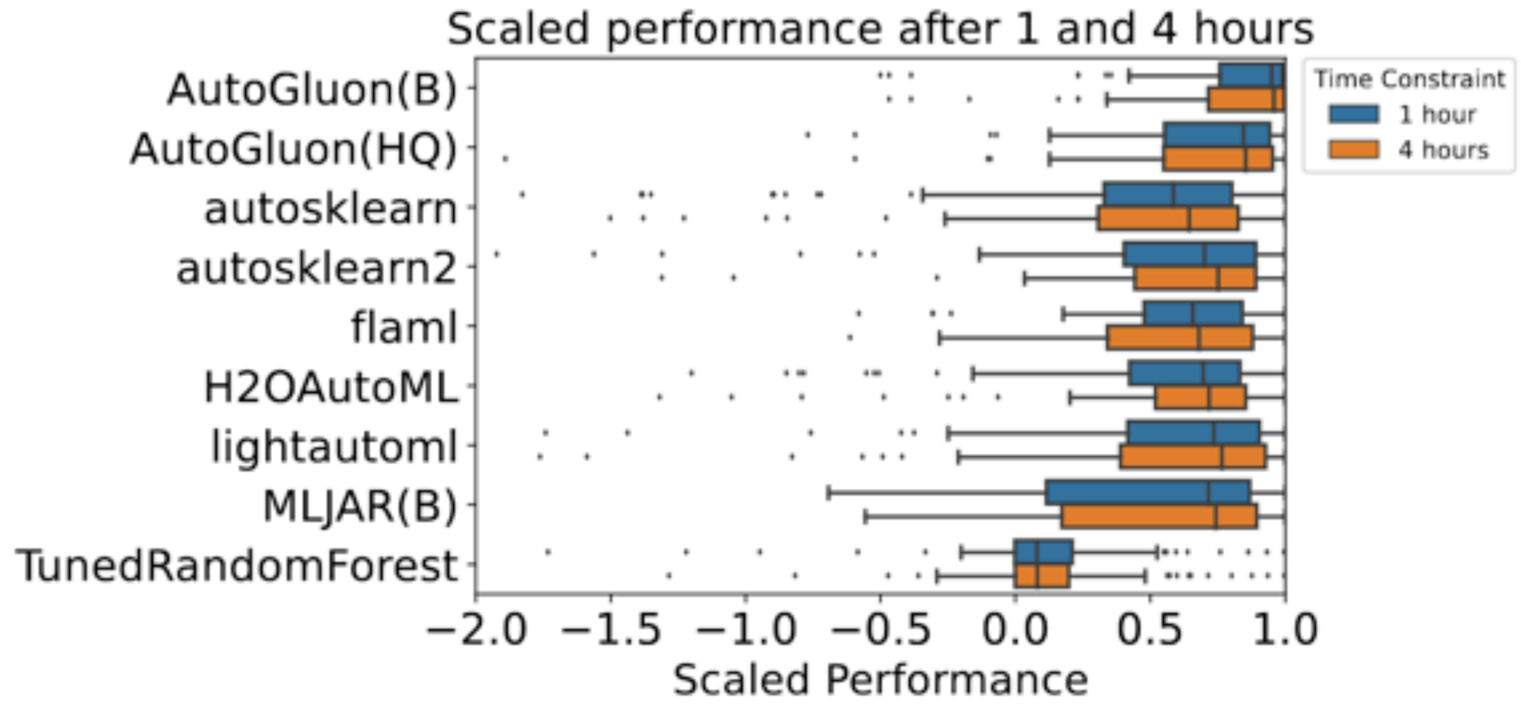
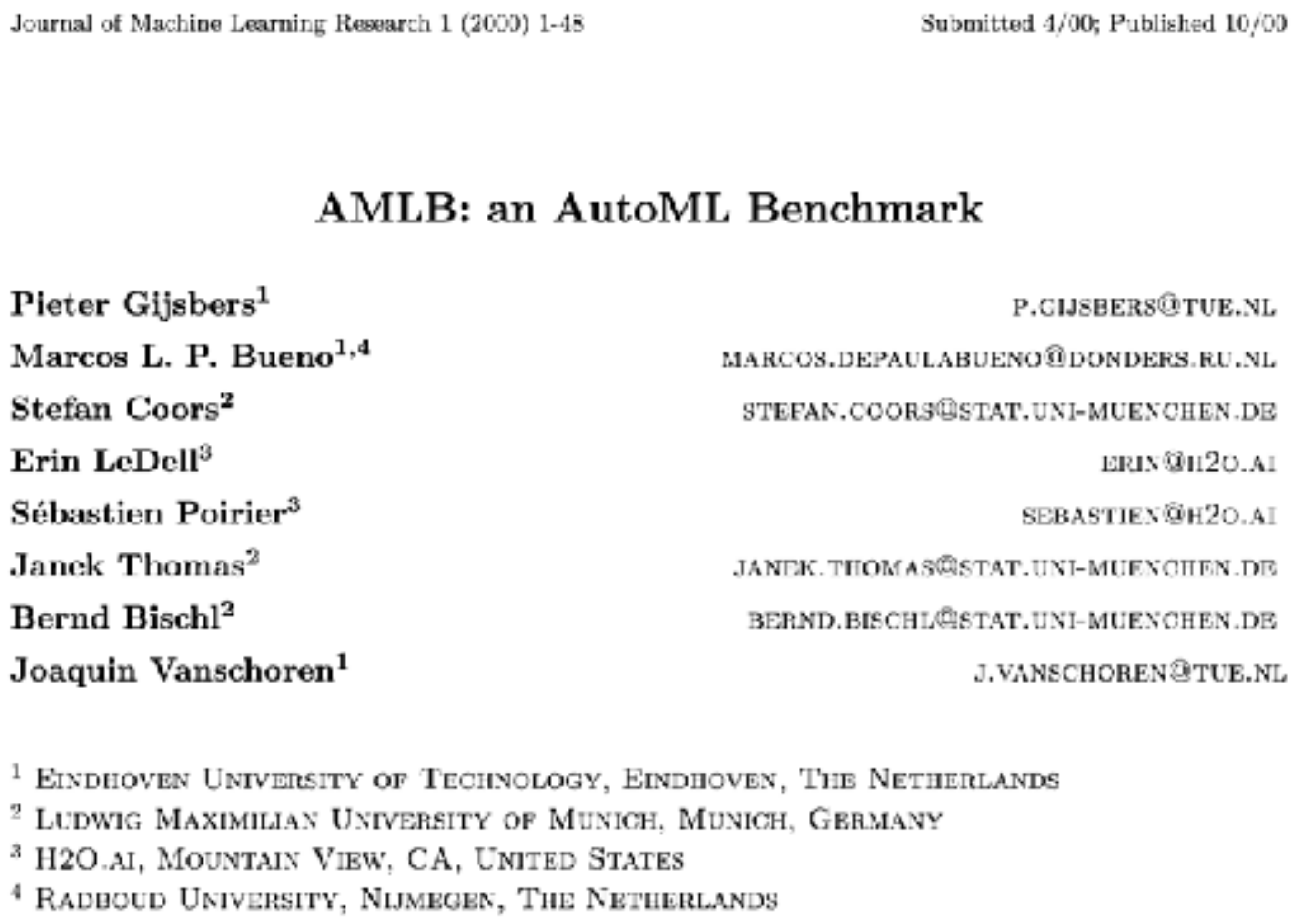



Figure 4: Scaled performance for each framework under different time constraints. Only frameworks which have evaluations on all tasks for both time constraints are shown. Performance generally does not improve much with more time.

# What is the best Tabular method?

- AutoML Benchmark [Ginsberg et al 2023] considered 71 classification and 33 regression datasets


Journal of Machine Learning Research 1 (2000) 1-48 Submitted 4/00; Published 10/00

AMLB: an AutoML Benchmark

Pieter Gijsbers[1] P.GIJSBERS@TUE.NL
Marcos L. P. Bueno[1,4] MARCOS.DEPAULABUENO@DONDERS.RU.NL
Stefan Coors[2] STEFAN.COORS@STAT.UNI-MUENCHEN.DE
Erin LeDell[3] ERIN@H2O.AI
Sébastien Poirier[3] SEBASTIEN@H2O.AI
Janek Thomas[2] JANEK.THOMAS@STAT.UNI-MUENCHEN.DE
Bernd Bischl[2] BERND.BISCHL@STAT.UNI-MUENCHEN.DE
Joaquin Vanschoren[1] J.VANSCHOREN@TUE.NL

[1] Eindhoven University of Technology, Eindhoven, The Netherlands
[2] Ludwig Maximilian University of Munich, Munich, Germany
[3] H2O.ai, Mountain View, CA, United States
[4] Radboud University, Nijmegen, The Netherlands

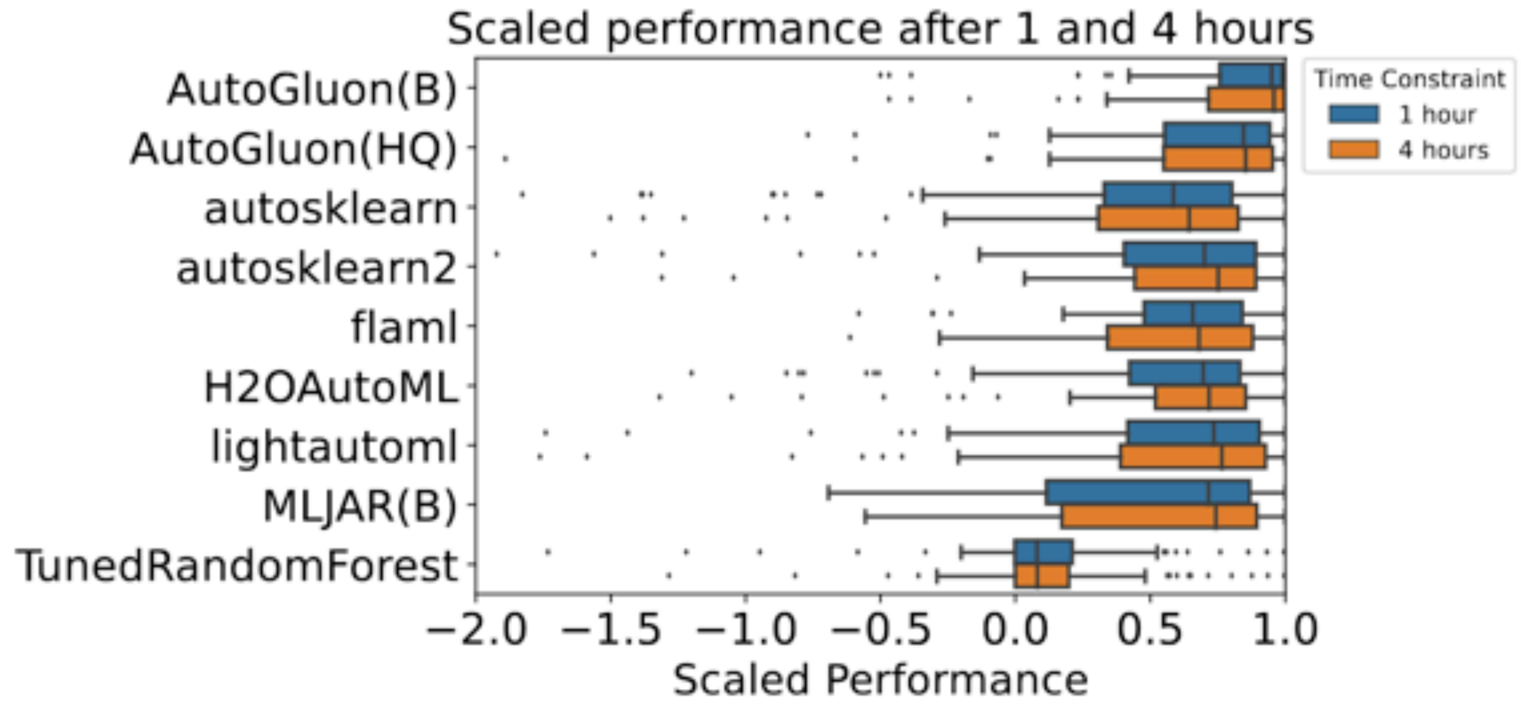



Figure 4: Scaled performance for each framework under different time constraints. Only frameworks which have evaluations on all tasks for both time constraints are shown. Performance generally does not improve much with more time.

- Considered 9 AutoML frameworks, evaluated on 1h and 4h fitting budget

# What is the best Tabular method?

- AutoML Benchmark [Ginsberg et al 2023] considered 71 classification and 33 regression datasets

Journal of Machine Learning Research 1 (2000) 1-48 Submitted 4/00; Published 10/00

AMLB: an AutoML Benchmark

Pieter Gijsbers[1] P.GIJSBERS@TUE.NL
Marcos L. P. Bueno[1,4] MARCOS.DEPAULABUENO@DONDERS.RU.NL
Stefan Coors[2] STEFAN.COORS@STAT.UNI-MUENCHEN.DE
Erin LeDell[3] ERIN@H2O.AI
Sébastien Poirier[3] SEBASTIEN@H2O.AI
Janek Thomas[2] JANEK.THOMAS@STAT.UNI-MUENCHEN.DE
Bernd Bischl[2] BERND.BISCHL@STAT.UNI-MUENCHEN.DE
Joaquin Vanschoren[1] J.VANSCHOREN@TUE.NL

[1] Eindhoven University of Technology, Eindhoven, The Netherlands
[2] Ludwig Maximilian University of Munich, Munich, Germany
[3] H2O.ai, Mountain View, CA, United States
[4] Radboud University, Nijmegen, The Netherlands

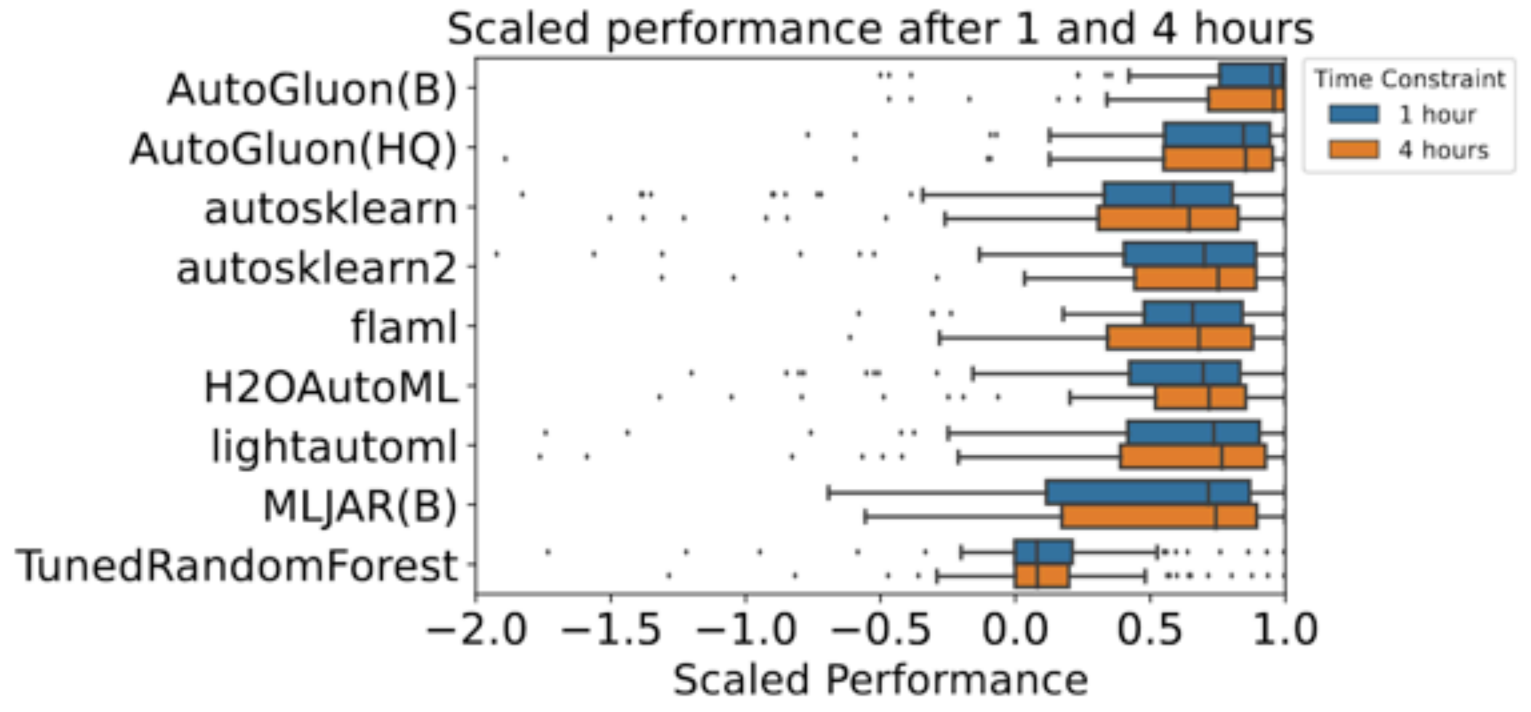


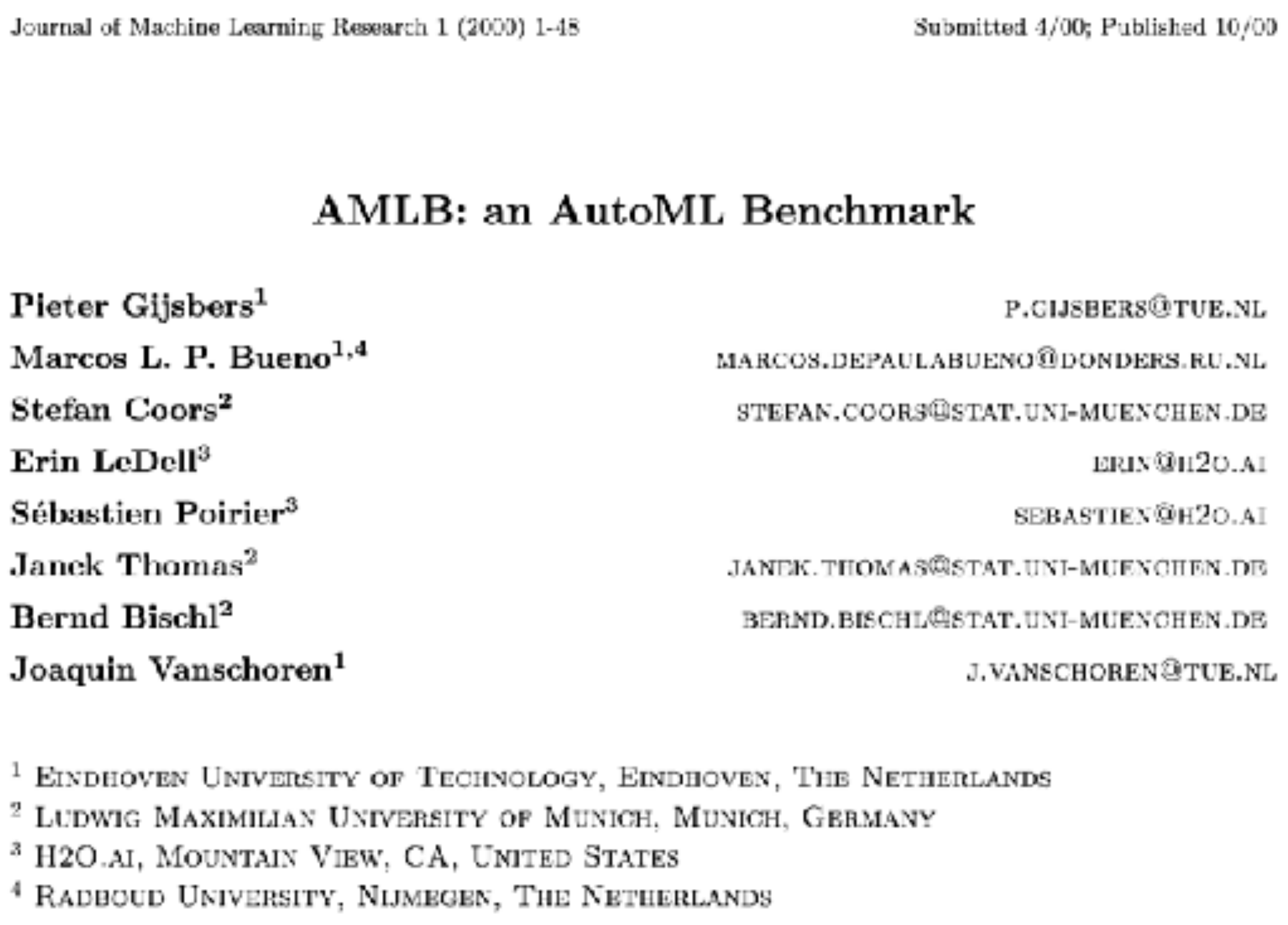

Figure 4: Scaled performance for each framework under different time constraints. Only frameworks which have evaluations on all tasks for both time constraints are shown. Performance generally does not improve much with more time.

- Considered 9 AutoML frameworks, evaluated on 1h and 4h fitting budget

- AutoGluon was then the best model by a large margin

# What is the best Tabular method?

- AutoML Benchmark [Ginsberg et al 2023] considered 71 classification and 33 regression datasets



Journal of Machine Learning Research 1 (2000) 1-48 Submitted 4/00; Published 10/00

AMLB: an AutoML Benchmark


Pieter Gijsbers[1] P.GIJSBERS@TUE.NL
Marcos L. P. Bueno[1,4] MARCOS.DEPAULABUENO@DONDERS.RU.NL
Stefan Coors[2] STEFAN.COORS@STAT.UNI-MUENCHEN.DE
Erin LeDell[3] ERIN@H2O.AI
Sébastien Poirier[3] SEBASTIEN@H2O.AI
Janek Thomas[2] JANEK.THOMAS@STAT.UNI-MUENCHEN.DE
Bernd Bischl[2] BERND.BISCHL@STAT.UNI-MUENCHEN.DE
Joaquin Vanschoren[1] J.VANSCHOREN@TUE.NL

[1] Eindhoven University of Technology, Eindhoven, The Netherlands
[2] Ludwig Maximilian University of Munich, Munich, Germany
[3] H2O.ai, Mountain View, CA, United States
[4] Radboud University, Nijmegen, The Netherlands

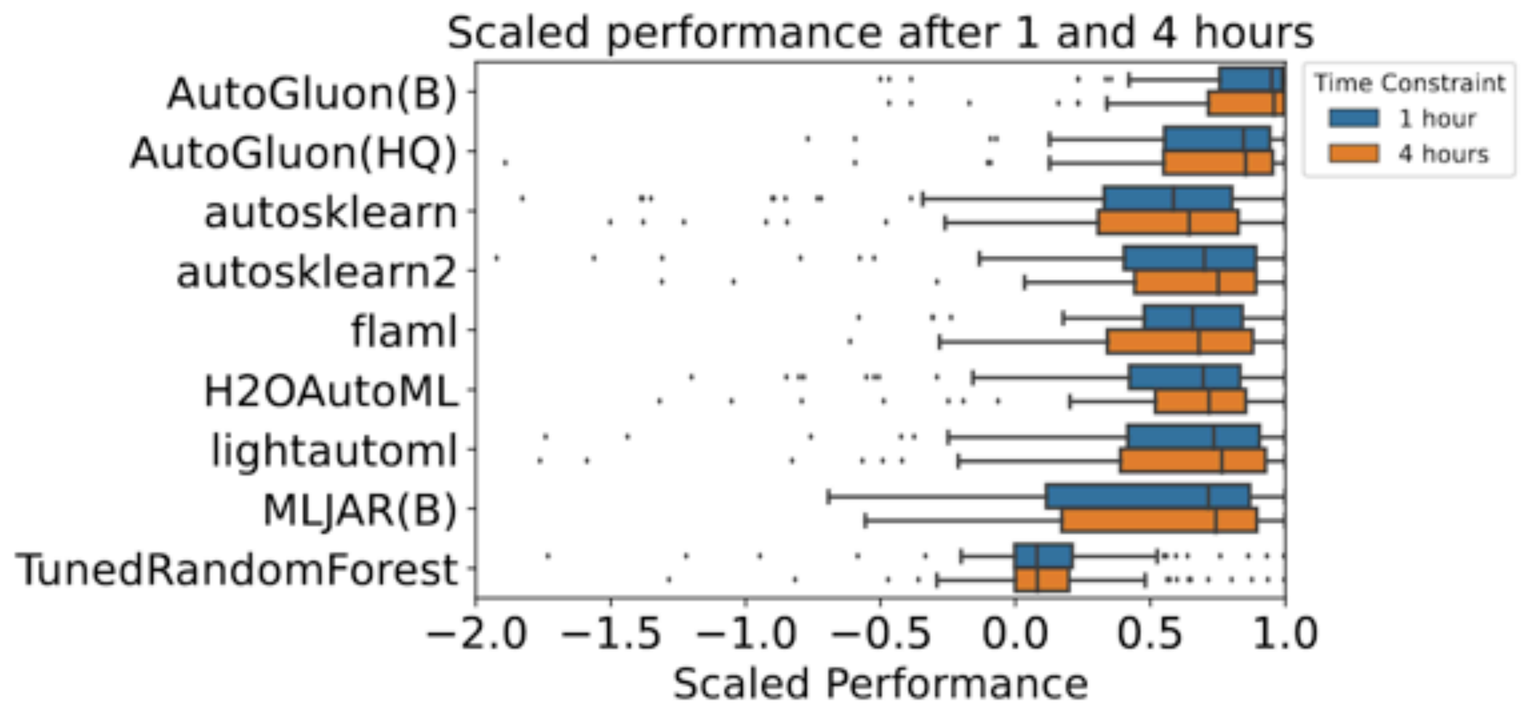




Figure 4: Scaled performance for each framework under different time constraints. Only frameworks which have evaluations on all tasks for both time constraints are shown. Performance generally does not improve much with more time.

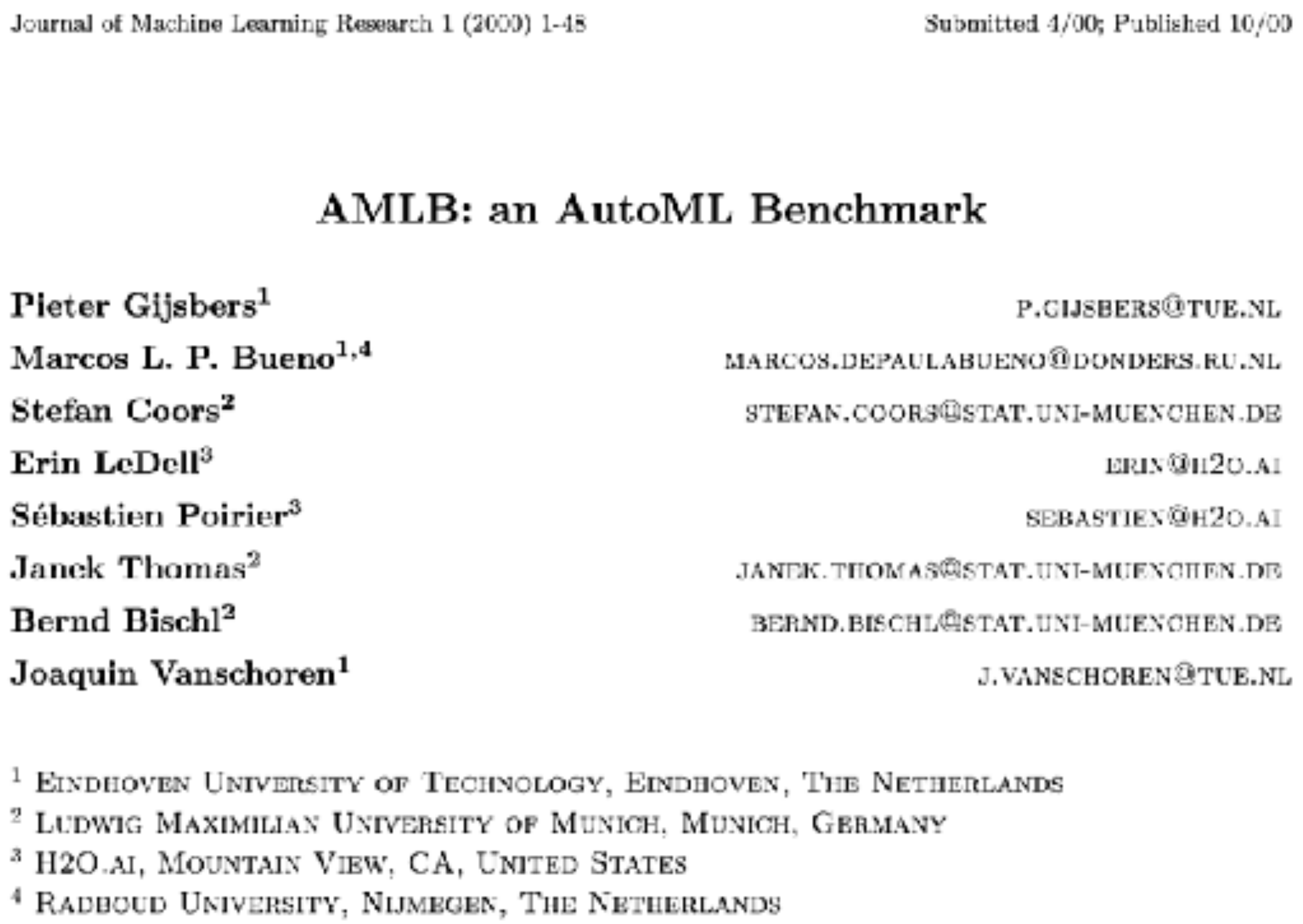
Evaluating a single method costs 40K CPU hours of compute!

- Considered 9 AutoML frameworks, evaluated on 1h and 4h fitting budget

- AutoGluon was then the best model by a large margin

# What is the best Tabular method?

- AutoML Benchmark [Ginsberg et al 2023] considered 71 classification and 33 regression datasets



Journal of Machine Learning Research 1 (2000) 1-48 Submitted 4/00; Published 10/00

AMLB: an AutoML Benchmark

Pieter Gijsbers[1] P.GIJSBERS@TUE.NL
Marcos L. P. Bueno[1,4] MARCOS.DEPAULABUENO@DONDERS.RU.NL
Stefan Coors[2] STEFAN.COORS@STAT.UNI-MUENCHEN.DE
Erin LeDell[3] ERIN@H2O.AI
Sébastien Poirier[3] SEBASTIEN@H2O.AI
Janek Thomas[2] JANEK.THOMAS@STAT.UNI-MUENCHEN.DE
Bernd Bischl[2] BERND.BISCHL@STAT.UNI-MUENCHEN.DE
Joaquin Vanschoren[1] J.VANSCHOREN@TUE.NL

[1] Eindhoven University of Technology, Eindhoven, The Netherlands
[2] Ludwig Maximilian University of Munich, Munich, Germany
[3] H2O.ai, Mountain View, CA, United States
[4] Radboud University, Nijmegen, The Netherlands

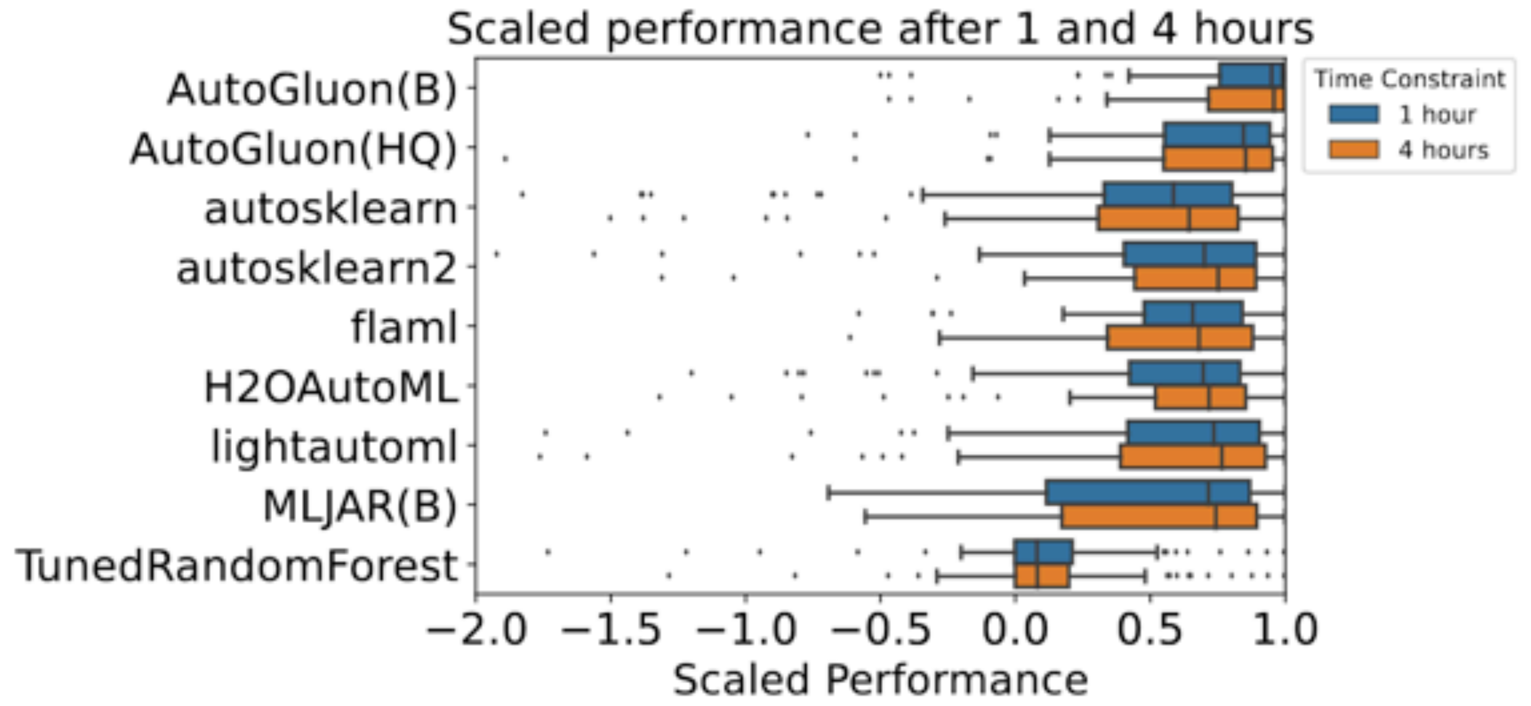



Figure 4: Scaled performance for each framework under different time constraints. Only frameworks which have evaluations on all tasks for both time constraints are shown. Performance generally does not improve much with more time.

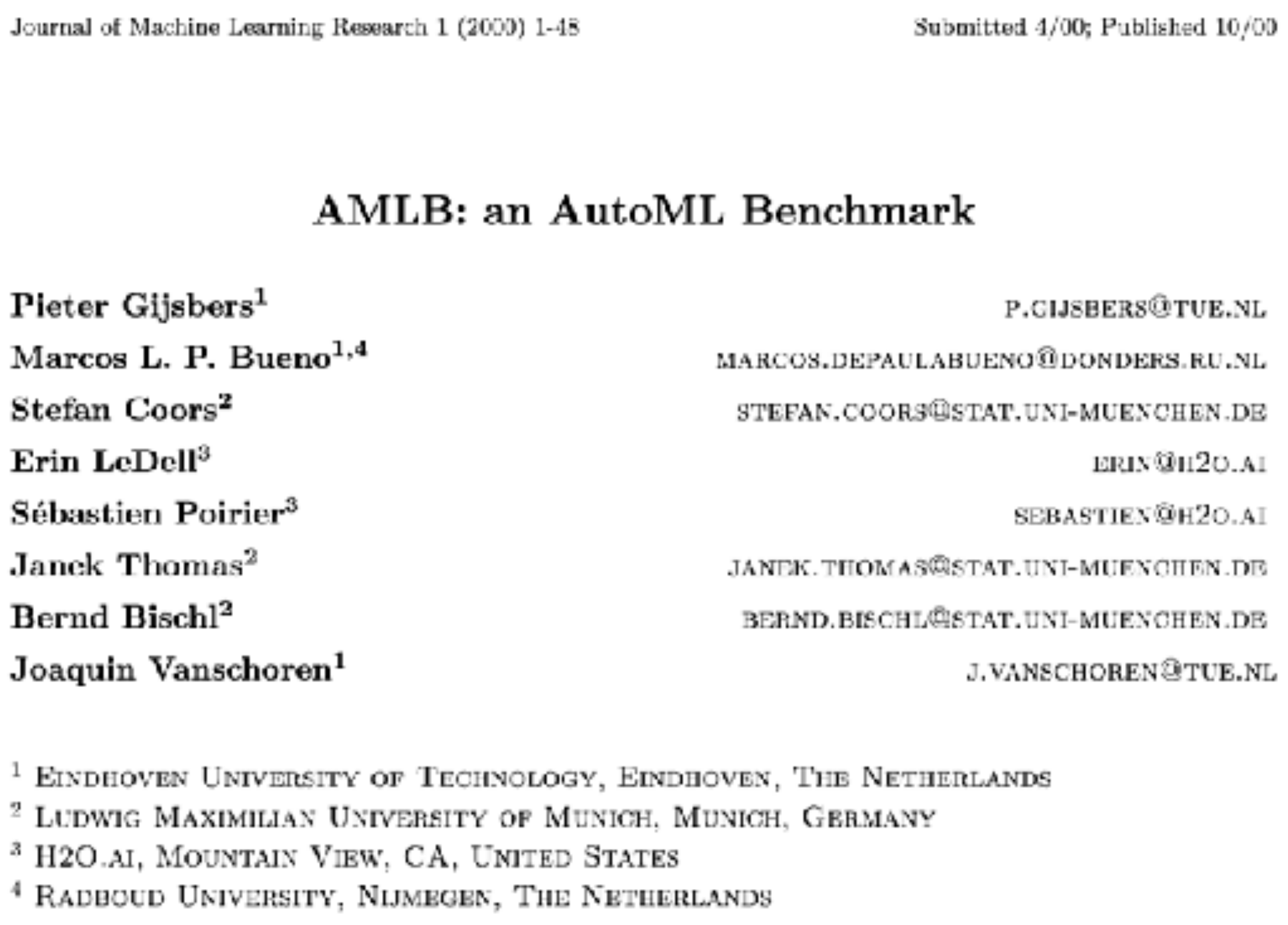
Evaluating a single method costs 40K CPU hours of compute!

Can we limit this cost? 🤔

- Considered 9 AutoML frameworks, evaluated on 1h and 4h fitting budget
- AutoGluon was then the best model by a large margin

# What is the best Tabular method?

- AutoML Benchmark [Ginsberg et al 2023] considered 71 classification and 33 regression datasets



Journal of Machine Learning Research 1 (2000) 1-48 Submitted 4/00; Published 10/00

AMLB: an AutoML Benchmark


Pieter Gijsbers[1] P.GIJSBERS@TUE.NL
Marcos L. P. Bueno[1,4] MARCOS.DEPAULABUENO@DONDERS.RU.NL
Stefan Coors[2] STEFAN.COORS@STAT.UNI-MUENCHEN.DE
Erin LeDell[3] ERIN@H2O.AI
Sébastien Poirier[3] SEBASTIEN@H2O.AI
Janek Thomas[2] JANEK.THOMAS@STAT.UNI-MUENCHEN.DE
Bernd Bischl[2] BERND.BISCHL@STAT.UNI-MUENCHEN.DE
Joaquin Vanschoren[1] J.VANSCHOREN@TUE.NL

[1] Eindhoven University of Technology, Eindhoven, The Netherlands
[2] Ludwig Maximilian University of Munich, Munich, Germany
[3] H2O.ai, Mountain View, CA, United States
[4] Radboud University, Nijmegen, The Netherlands

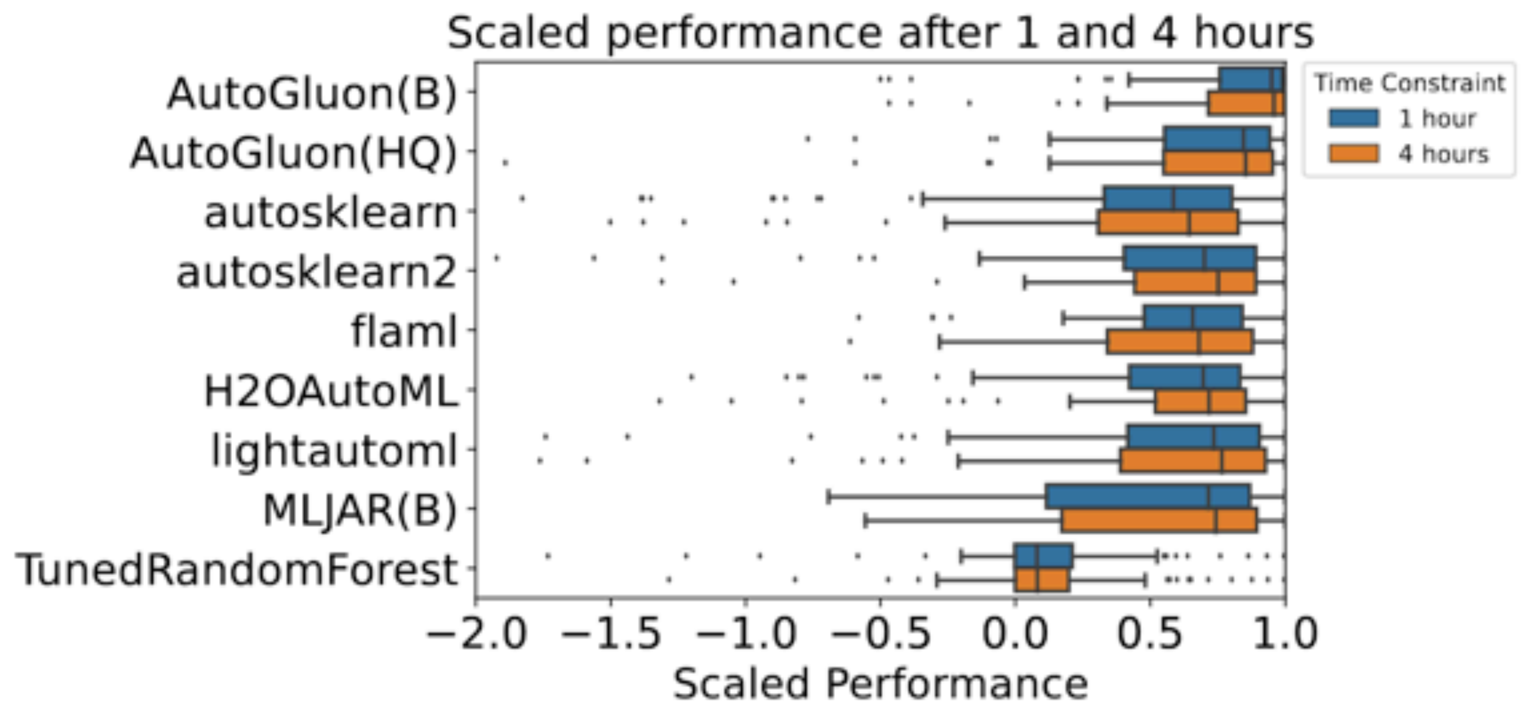




Figure 4: Scaled performance for each framework under different time constraints. Only frameworks which have evaluations on all tasks for both time constraints are shown. Performance generally does not improve much with more time.

Evaluating a single method costs 40K CPU hours of compute!

Can we limit this cost? 🤔

- Considered 9 AutoML frameworks, evaluated on 1h and 4h fitting budget

- AutoGluon How does this work? best model by a large margin

# AutoGluon

## Hyperparameter optimization

# AutoGluon

## Hyperparameter optimization

- Strikingly, AutoGluon achieved state-of-the-art results **without** HPO with its mix of bagging, stacking, ensembling and good heuristic featurizers

# AutoGluon

## Hyperparameter optimization

- Strikingly, AutoGluon achieved state-of-the-art results **without** HPO with its mix of bagging, stacking, ensembling and good heuristic featurizers
- It is not that HPO does not help, it does but compute is better spent evaluating a good set of default models (with more folds, more rounds, etc)

# AutoGluon

## Hyperparameter optimization

- Strikingly, AutoGluon achieved state-of-the-art results **without** HPO with its mix of bagging, stacking, ensembling and good heuristic featurizers
- It is not that HPO does not help, it does but compute is better spent evaluating a good set of default models (with more folds, more rounds, etc)
- AutoGluon default models: 13 default hyperparameters chosen manually by experts

# AutoGluon

## Hyperparameter optimization

- Strikingly, AutoGluon achieved state-of-the-art results **without** HPO with its mix of bagging, stacking, ensembling and good heuristic featurizers
- It is not that HPO does not help, it does but compute is better spent evaluating a good set of default models (with more folds, more rounds, etc)
- AutoGluon default models: 13 default hyperparameters chosen manually by experts
- Can transfer learning help?

# TabRepo

## TabRepo: A Large Scale Repository of Tabular Model Evaluations and its AutoML Applications

David Salinas[1,*] Nick Erickson[1,*]

# TabRepo

- Goals:

TabRepo: A Large Scale Repository of Tabular Model Evaluations and its AutoML Applications

David Salinas[1,*] Nick Erickson[1,*]

# TabRepo

**TabRepo: A Large Scale Repository of Tabular Model Evaluations and its AutoML Applications**

David Salinas[1,*] Nick Erickson[1,*]

- Goals:
    - 1) reduce cost of evaluation (40K CPU hours to evaluate a single method on AutoML Benchmark)

# TabRepo

**TabRepo: A Large Scale Repository of Tabular Model Evaluations and its AutoML Applications**

David Salinas[1,*] Nick Erickson[1,*]

- Goals:
    - 1) reduce cost of evaluation (40K CPU hours to evaluate a single method on AutoML Benchmark)
    - 2) improve over the manual selection of AutoGluon default models

# TabRepo

**TabRepo: A Large Scale Repository of Tabular Model Evaluations and its AutoML Applications**

David Salinas[1,*] Nick Erickson[1,*]

- Goals:
  - 1) reduce cost of evaluation (40K CPU hours to evaluate a single method on AutoML Benchmark)
  - 2) improve over the manual selection of AutoGluon default models
- Precomputed evaluations and results on:

# TabRepo

**TabRepo: A Large Scale Repository of Tabular Model Evaluations and its AutoML Applications**

David Salinas[1,*] Nick Erickson[1,*]

- Goals:
  - 1) reduce cost of evaluation (40K CPU hours to evaluate a single method on AutoML Benchmark)
  - 2) improve over the manual selection of AutoGluon default models
- Precomputed evaluations and results on:
  - 200 datasets from regression, classification, multi-class (thanks OpenML 🥰)

# TabRepo

TabRepo: A Large Scale Repository of Tabular Model Evaluations and its AutoML Applications

David Salinas[1,*] Nick Erickson[1,*]

- Goals:
  - 1) reduce cost of evaluation (40K CPU hours to evaluate a single method on AutoML Benchmark)
  - 2) improve over the manual selection of AutoGluon default models
- Precomputed evaluations and results on:
  - 200 datasets from regression, classification, multi-class (thanks OpenML 🥰)
  - 200 random configurations of models used in AutoGluon (CatBoost, MLP, LightGBM, RandomForest, …) on all datasets with 3 seeds

# TabRepo

TabRepo: A Large Scale Repository of Tabular Model Evaluations and its AutoML Applications

David Salinas[1,*] Nick Erickson[1,*]

- Goals:
  - 1) reduce cost of evaluation (40K CPU hours to evaluate a single method on AutoML Benchmark)
  - 2) improve over the manual selection of AutoGluon default models
- Precomputed evaluations and results on:
  - 200 datasets from regression, classification, multi-class (thanks OpenML 🥰)
  - 200 random configurations of models used in AutoGluon (CatBoost, MLP, LightGBM, RandomForest, …) on all datasets with 3 seeds
- Performance metrics (latency, accuracy, …) **and predictions** available for every dataset, model, seed

# TabRepo

TabRepo: A Large Scale Repository of Tabular Model Evaluations and its AutoML Applications

David Salinas[1,*] Nick Erickson[1,*]

- Goals:
  - 1) reduce cost of evaluation (40K CPU hours to evaluate a single method on AutoML Benchmark)
  - 2) improve over the manual selection of AutoGluon default models
- Precomputed evaluations and results on:
  - 200 datasets from regression, classification, multi-class (thanks OpenML 🥰)
  - 200 random configurations of models used in AutoGluon (CatBoost, MLP, LightGBM, RandomForest, …) on all datasets with 3 seeds
- Performance metrics (latency, accuracy, …) **and predictions** available for every dataset, model, seed
- ~100GB of data, ~200K CPU hours of compute

# TabRepo

**TabRepo: A Large Scale Repository of Tabular Model Evaluations and its AutoML Applications**

David Salinas[1,*] Nick Erickson[1,*]

- Goals:
  - 1) reduce cost of evaluation (40K CPU hours to evaluate a single method on AutoML Benchmark)
  - 2) improve over the manual selection of AutoGluon default models
- Precomputed evaluations and results on:
  - 200 datasets from regression, classification, multi-class (thanks OpenML 🥰)
  - 200 random configurations of models used in AutoGluon (CatBoost, MLP, LightGBM, RandomForest, …) on all datasets with 3 seeds
- Performance metrics (latency, accuracy, …) **and predictions** available for every dataset, model, seed
- ~100GB of data, ~200K CPU hours of compute

> 💡 **Storing predictions** and target labels allows to obtain the performance of **any ensemble** on the fly!

# TabRepo

**TabRepo: A Large Scale Repository of Tabular Model Evaluations and its AutoML Applications**

David Salinas[1,*] Nick Erickson[1,*]

- Goals:
  - 1) reduce cost of evaluation (40K CPU hours to evaluate a single method on AutoML Benchmark)
  - 2) improve over the manual selection of AutoGluon default models
- Precomputed evaluations and results on:
  - 200 datasets from regression, classification, multi-class (thanks OpenML 🥰)
  - 200 random configurations of models used in AutoGluon (CatBoost, MLP, LightGBM, RandomForest, …) on all datasets with 3 seeds
- Performance metrics (latency, accuracy, …) **and predictions** available for every dataset, model, seed
- ~100GB of data, ~200K CPU hours of compute

💡 **Storing predictions** and target labels allows to obtain the performance of **any ensemble** on the fly!

🥳 The dataset combined with **portfolio learning** allows to outperform Autogluon!

# TabRepo

## Studying the effect of HPO and ensembling

# TabRepo

## Studying the effect of HPO and ensembling

💡 **Storing predictions** and target labels allows to obtain the performance of **any ensemble** on the fly!

# TabRepo

## Studying the effect of HPO and ensembling

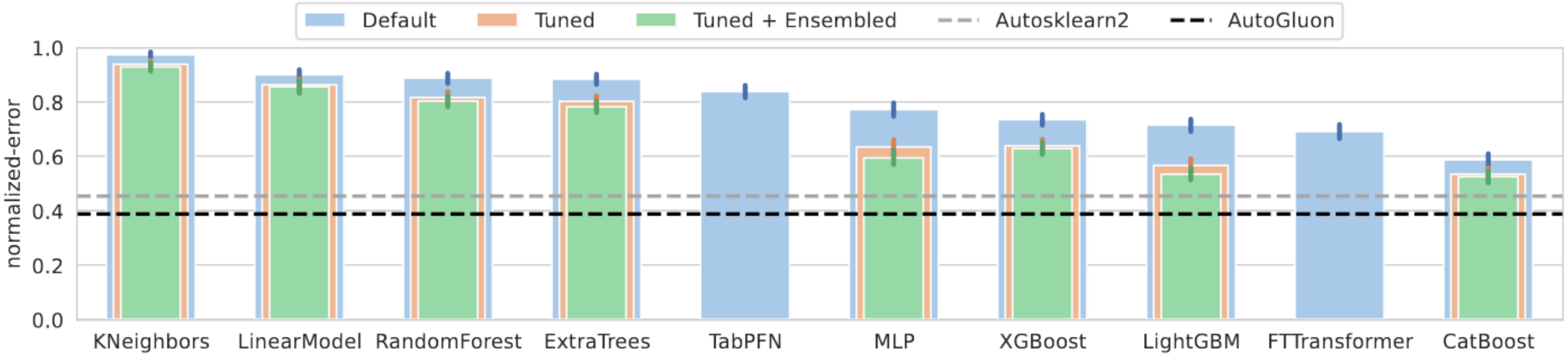



Figure 2: Normalized error for all model families when using default hyperparameters, tuned hyperparameters, and ensembling after tuning. All methods are run with a 4h budget.

💡 **Storing predictions** and target labels allows to obtain the performance of **any ensemble** on the fly!

# TabRepo

## Studying the effect of HPO and ensembling

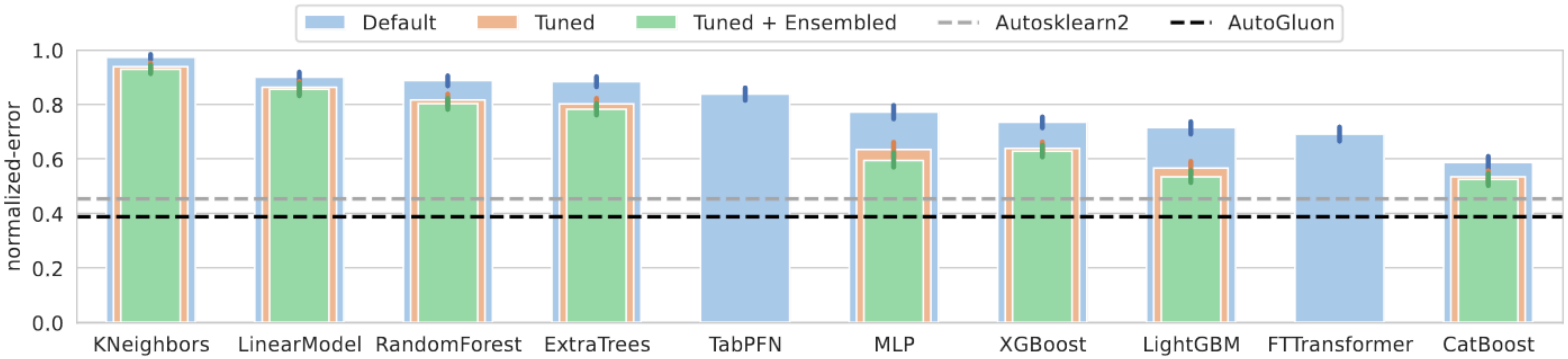


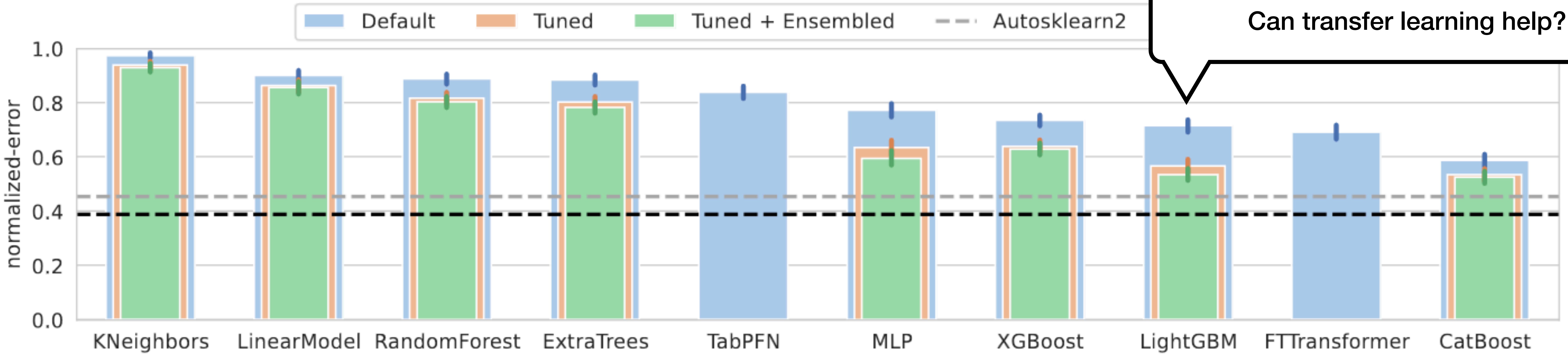

Figure 2: Normalized error for all model families when using default hyperparameters, tuned hyperparameters, and ensembling after tuning. All methods are run with a 4h budget.

💡 **Storing predictions** and target labels allows to obtain the performance of **any ensemble** on the fly!

Doing this analysis just costs a few minutes on a laptop (as opposed to days on a cluster!)

# TabRepo

## Studying the effect of HPO and ensembling




Tuning hyperparameters helps a lot but it is not done in AutoGluon because of cost. Can transfer learning help?

Figure 2: Normalized error for all model families when using default hyperparameters, tuned hyperparameters, and ensembling after tuning. All methods are run with a 4h budget.

💡 **Storing predictions** and target labels allows to obtain the performance of **any ensemble** on the fly!

Doing this analysis just costs a few minutes on a laptop (as opposed to days on a cluster!)

# Portfolio learning

# Portfolio learning

- Assume we have access to error metrics of $n$ datasets on $m$ models, denoted as $\varepsilon \in \mathbb{R}^{n \times m}$

# Portfolio learning

- Assume we have access to error metrics of $n$ datasets on $m$ models, denoted as $\varepsilon \in \mathbb{R}^{n \times m}$
- How can we select the best set of $k$ default models for an average dataset?

# Portfolio learning

- Assume we have access to error metrics of $n$ datasets on $m$ models, denoted as $\varepsilon \in \mathbb{R}^{n \times m}$
- How can we select the best set of $k$ default models for an average dataset?
- Solve the optimization problem:

# Portfolio learning

- Assume we have access to error metrics of $n$ datasets on $m$ models, denoted as $\varepsilon \in \mathbb{R}^{n \times m}$
- How can we select the best set of $k$ default models for an average dataset?
- Solve the optimization problem:

$$(j_1, \ldots, j_k) = \text{argmin}_{(j_1, \ldots, j_k) \in [m]^k} \frac{1}{n} \sum_{i=1}^{n} \min(\varepsilon_{ij_1}, \ldots, \varepsilon_{ij_k})$$

# Portfolio learning

- Assume we have access to error metrics of $n$ datasets on $m$ models, denoted as $\varepsilon \in \mathbb{R}^{n \times m}$
- How can we select the best set of $k$ default models for an average dataset?
- Solve the optimization problem:

Select among all possible sets of k models

$$(j_1, \ldots, j_k) = \text{argmin}_{(j_1, \ldots, j_k) \in [m]^k} \frac{1}{n} \sum_{i=1}^{n} \min(\varepsilon_{ij_1}, \ldots, \varepsilon_{ij_k})$$

# Portfolio learning

- Assume we have access to error metrics of $n$ datasets on $m$ models, denoted as $\varepsilon \in \mathbb{R}^{n \times m}$
- How can we select the best set of $k$ default models for an average dataset?
- Solve the optimization problem:

With best avg. performance across datasets ...

Select among all possible sets of k models

$$(j_1, \ldots, j_k) = \text{argmin}_{(j_1, \ldots, j_k) \in [m]^k} \frac{1}{n} \sum_{i=1}^{n} \min(\varepsilon_{ij_1}, \ldots, \varepsilon_{ij_k})$$

# Portfolio learning

- Assume we have access to error metrics of $n$ datasets on $m$ models, denoted as $\varepsilon \in \mathbb{R}^{n \times m}$
- How can we select the best set of $k$ default models for an average dataset?
- Solve the optimization problem:

With best avg. performance across datasets ...

... when using the best performing model on a given dataset

Select among all possible sets of k models

$$(j_1, \ldots, j_k) = \text{argmin}_{(j_1, \ldots, j_k) \in [m]^k} \frac{1}{n} \sum_{i=1}^{n} \min(\varepsilon_{ij_1}, \ldots, \varepsilon_{ij_k})$$

# Portfolio learning

- Assume we have access to error metrics of $n$ datasets on $m$ models, denoted as $\varepsilon \in \mathbb{R}^{n \times m}$
- How can we select the best set of $k$ default models for an average dataset?
- Solve the optimization problem:

With best avg. performance across datasets ...

... when using the best performing model on a given dataset

Select among all possible sets of k models

$$(j_1, \ldots, j_k) = \text{argmin}_{(j_1, \ldots, j_k) \in [m]^k} \frac{1}{n} \sum_{i=1}^{n} \min(\varepsilon_{ij_1}, \ldots, \varepsilon_{ij_k})$$

- NP-hard [Feurer 2022], but admits an approximation

# Portfolio learning

- Assume we have access to error metrics of $n$ datasets on $m$ models, denoted as $\varepsilon \in \mathbb{R}^{n \times m}$
- How can we select the best set of $k$ default models for an average dataset?
- Solve the optimization problem:

With best avg. performance across datasets ...

... when using the best performing model on a given dataset

Select among all possible sets of k models

$$(j_1, \ldots, j_k) = \text{argmin}_{(j_1,\ldots,j_k) \in [m]^k} \frac{1}{n} \sum_{i=1}^{n} \min(\varepsilon_{ij_1}, \ldots, \varepsilon_{ij_k})$$

- NP-hard [Feurer 2022], but admits an approximation
- Greedy algorithm:

# Portfolio learning

- Assume we have access to error metrics of $n$ datasets on $m$ models, denoted as $\varepsilon \in \mathbb{R}^{n \times m}$
- How can we select the best set of $k$ default models for an average dataset?
- Solve the optimization problem:

With best avg. performance across datasets ...

... when using the best performing model on a given dataset

Select among all possible sets of k models

$$(j_1, \ldots, j_k) = \text{argmin}_{(j_1,\ldots,j_k) \in [m]^k} \frac{1}{n} \sum_{i=1}^{n} \min(\varepsilon_{ij_1}, \ldots, \varepsilon_{ij_k})$$

- NP-hard [Feurer 2022], but admits an approximation
- Greedy algorithm:

$$j_1 = \text{argmin}_{j_1 \in [m]} \frac{1}{n} \sum_{i=1}^{n} \varepsilon_{ij_1}, \qquad j_n = \text{argmin}_{j_n \in [m]} \frac{1}{n} \sum_{i=1}^{n} \min(\varepsilon_{ij_1}, \ldots, \varepsilon_{ij_n})$$

# Portfolio learning

- Assume we have access to error metrics of $n$ datasets on $m$ models, denoted as $\varepsilon \in \mathbb{R}^{n \times m}$
- How can we select the best set of $k$ default models for an average dataset?
- Solve the optimization problem:

With best avg. performance across datasets ...

... when using the best performing model on a given dataset

Select among all possible sets of k models

$$(j_1, \ldots, j_k) = \text{argmin}_{(j_1, \ldots, j_k) \in [m]^k} \frac{1}{n} \sum_{i=1}^{n} \min(\varepsilon_{ij_1}, \ldots, \varepsilon_{ij_k})$$

- NP-hard [Feurer 2022], but admits an approximation
- Greedy algorithm:

$$j_1 = \text{argmin}_{j_1 \in [m]} \frac{1}{n} \sum_{i=1}^{n} \varepsilon_{ij_1}, \qquad j_n = \text{argmin}_{j_n \in [m]} \frac{1}{n} \sum_{i=1}^{n} \min(\varepsilon_{ij_1}, \ldots, \varepsilon_{ij_n})$$

Pick the model performing best on average

# Portfolio learning

- Assume we have access to error metrics of $n$ datasets on $m$ models, denoted as $\varepsilon \in \mathbb{R}^{n \times m}$
- How can we select the best set of $k$ default models for an average dataset?
- Solve the optimization problem:

With best avg. performance across datasets ...

... when using the best performing model on a given dataset

Select among all possible sets of k models

$$(j_1, \ldots, j_k) = \text{argmin}_{(j_1,\ldots,j_k)\in[m]^k} \frac{1}{n} \sum_{i=1}^{n} \min(\varepsilon_{ij_1}, \ldots, \varepsilon_{ij_k})$$

- NP-hard [Feurer 2022], but admits an approximation
- Greedy algorithm:

$$j_1 = \text{argmin}_{j_1\in[m]} \frac{1}{n} \sum_{i=1}^{n} \varepsilon_{ij_1}, \qquad j_n = \text{argmin}_{j_n\in[m]} \frac{1}{n} \sum_{i=1}^{n} \min(\varepsilon_{ij_1}, \ldots, \varepsilon_{ij_n})$$

Pick the model performing best on average

Pick the model performing best on average when combined with the ones previously selected

# Portfolio learning

- Assume we have access to error metrics of $n$ datasets on $m$ models, denoted as $\varepsilon \in \mathbb{R}^{n \times m}$
- How can we select the best set of $k$ default models for an average dataset?
- Solve the optimization problem:

With best avg. performance across datasets ...

... when using the best performing model on a given dataset

Select among all possible sets of k models

$$(j_1, \ldots, j_k) = \text{argmin}_{(j_1, \ldots, j_k) \in [m]^k} \frac{1}{n} \sum_{i=1}^{n} \min(\varepsilon_{ij_1}, \ldots, \varepsilon_{ij_k})$$

**Benefits** 👍:
- Approximation guarantees from the original (sub-modular) problem
- Tractable
- Works extremely well in practice

- NP-hard [Feurer 2022], but admits an approximation
- Greedy algorithm:

$$j_1 = \text{argmin}_{j_1 \in [m]} \frac{1}{n} \sum_{i=1}^{n} \varepsilon_{ij_1}, \qquad j_n = \text{argmin}_{j_n \in [m]} \frac{1}{n} \sum_{i=1}^{n} \min(\varepsilon_{ij_1}, \ldots, \varepsilon_{ij_n})$$

Pick the model performing best on average

Pick the model performing best on average when combined with the ones previously selected

# Portfolio learning

- Assume we have access to error metrics of $n$ datasets on $m$ models, denoted as $\varepsilon \in \mathbb{R}^{n \times m}$
- How can we select the best set of $k$ default models for an average dataset?
- Solve the optimization problem:

With best avg. performance across datasets ...

... when using the best performing model on a given dataset

Select among all possible sets of k models

$$(j_1, \ldots, j_k) = \text{argmin}_{(j_1, \ldots, j_k) \in [m]^k} \frac{1}{n} \sum_{i=1}^{n} \min(\varepsilon_{ij_1}, \ldots, \varepsilon_{ij_k})$$

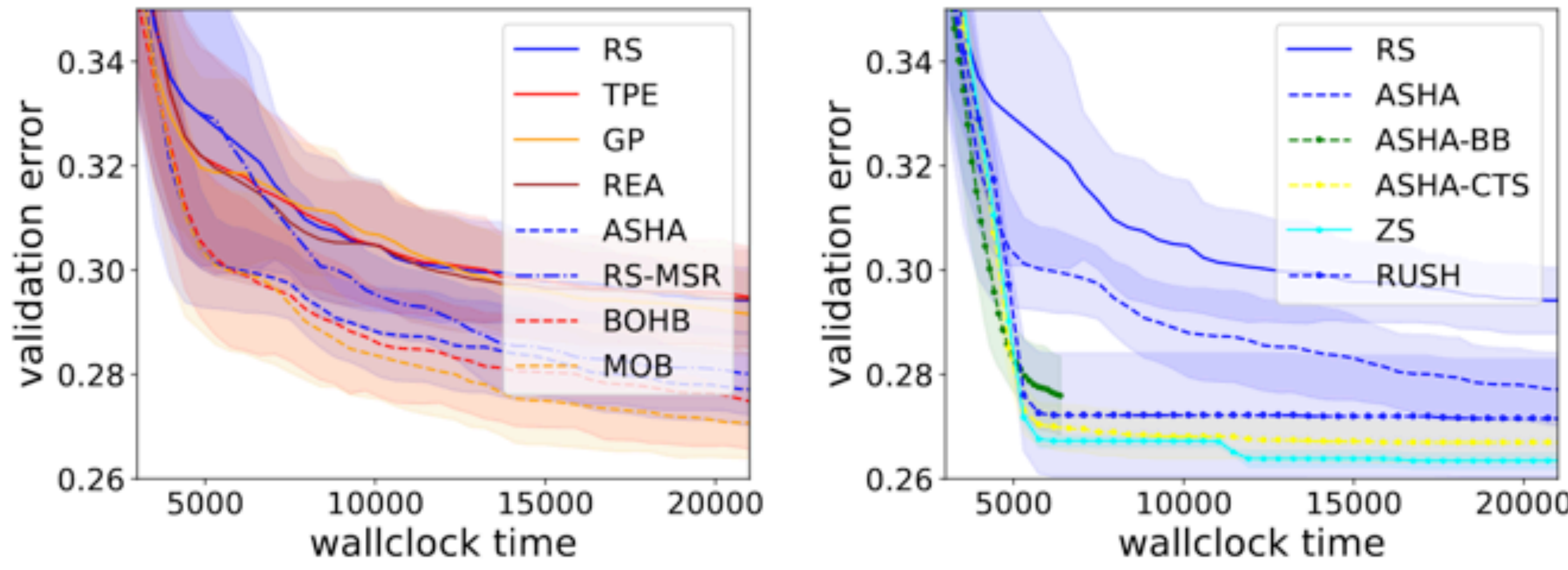
- NP-hard [Feurer 2022], but admits an approximation
- Greedy algorithm:

$$j_1 = \text{argmin}_{j_1 \in [m]} \frac{1}{n} \sum_{i=1}^{n} \varepsilon_{ij_1}, \qquad j_n = \text{argmin}_{j_n \in [m]} \frac{1}{n} \sum_{i=1}^{n} \min$$

Pick the model performing best on average

Pick the model performing best on average when combined with the ones previously selected

**Benefits** 👍:
- Approximation guarantees from the original (sub-modular) problem
- Tractable
- Works extremely well in practice




(b) NAS201: CIFAR-100.

Syne Tune: A Library for Large-Scale Hyperparameter Tuning and Reproducible Research [Salinas 2022]

# Portfolio learning

- Assume we have access to error metrics of $n$ datasets on $m$ models, denoted as $\varepsilon \in \mathbb{R}^{n \times m}$
- How can we select the best set of $k$ default models for an average dataset?
- Solve the optimization problem:

With best avg. performance across datasets …

… when using the best performing model on a given dataset

Select among all possible sets of k models

$$(j_1, \ldots, j_k) = \text{argmin}_{(j_1,\ldots,j_k)\in[m]^k} \frac{1}{n}\sum_{i=1}^{n} \min(\varepsilon_{ij_1}, \ldots, \varepsilon_{ij_k})$$

**Benefits** 👍:
- Approximation guarantees from the original (sub-modular) problem
- Tractable
- Works extremely well in practice

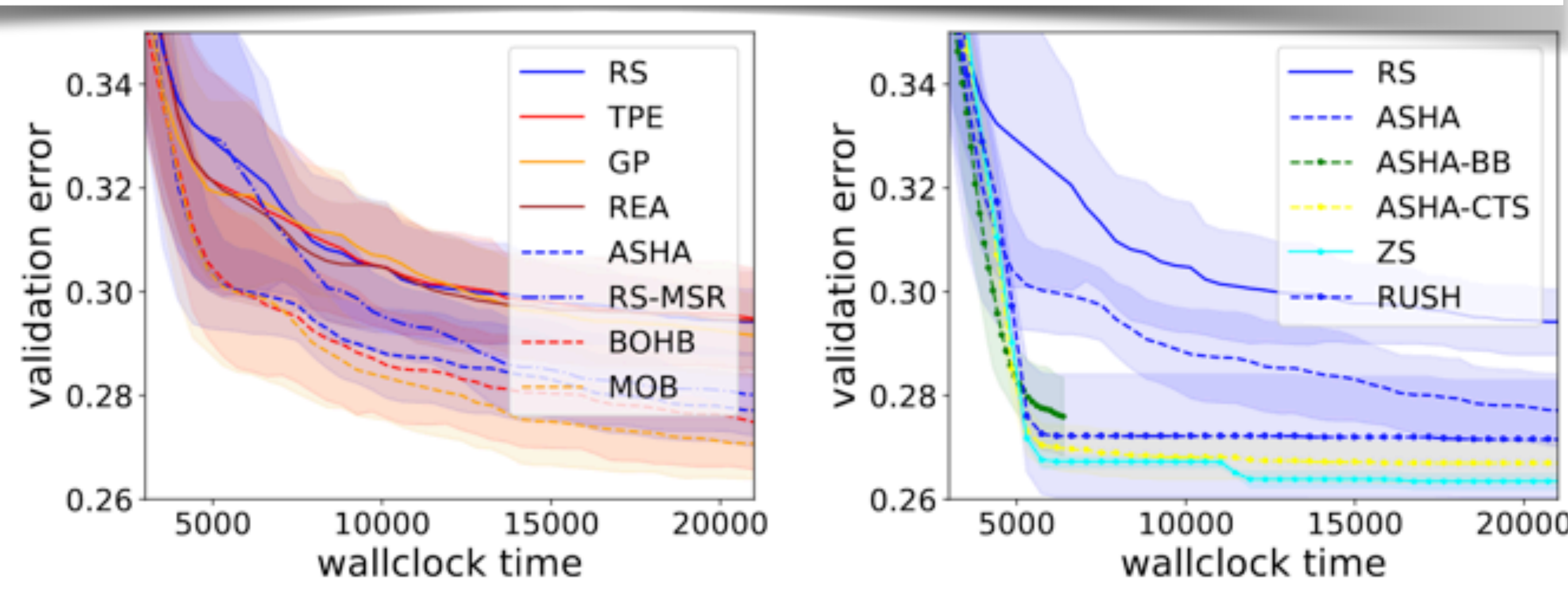
**Disadvantage** 👎: needs a grid or to build a surrogate

- NP-hard [Feurer 2022], but admits an approximation
- Greedy algorithm:

$$j_1 = \text{argmin}_{j_1\in[m]} \frac{1}{n}\sum_{i=1}^{n} \varepsilon_{ij_1}, \qquad j_n = \text{argmin}_{j_n\in[m]} \frac{1}{n}\sum_{i=1}^{n} \min$$

Pick the model performing best on average

Pick the model performing best on average when combined with the ones previously selected




(b) NAS201: CIFAR-100.

Syne Tune: A Library for Large-Scale Hyperparameter Tuning and Reproducible Research [Salinas 2022]

# Portfolio learning

- Assume we have access to error metrics of $n$ datasets on $m$ models, denoted as $\varepsilon \in \mathbb{R}^{n \times m}$
- How can we select the best set of $k$ default models for an average dataset?
- Solve the optimization problem:

With best avg. performance across datasets …

… when using the best performing model on a given dataset

Select among all possible sets of k models

$$(j_1, \ldots, j_k) = \text{argmin}_{(j_1,\ldots,j_k) \in [m]^k} \frac{1}{n} \sum_{i=1}^{n} \min(\varepsilon_{ij_1}, \ldots, \varepsilon_{ij_k})$$

- NP-hard [Feurer 2022], but admits an approximation
- Greedy algorithm:

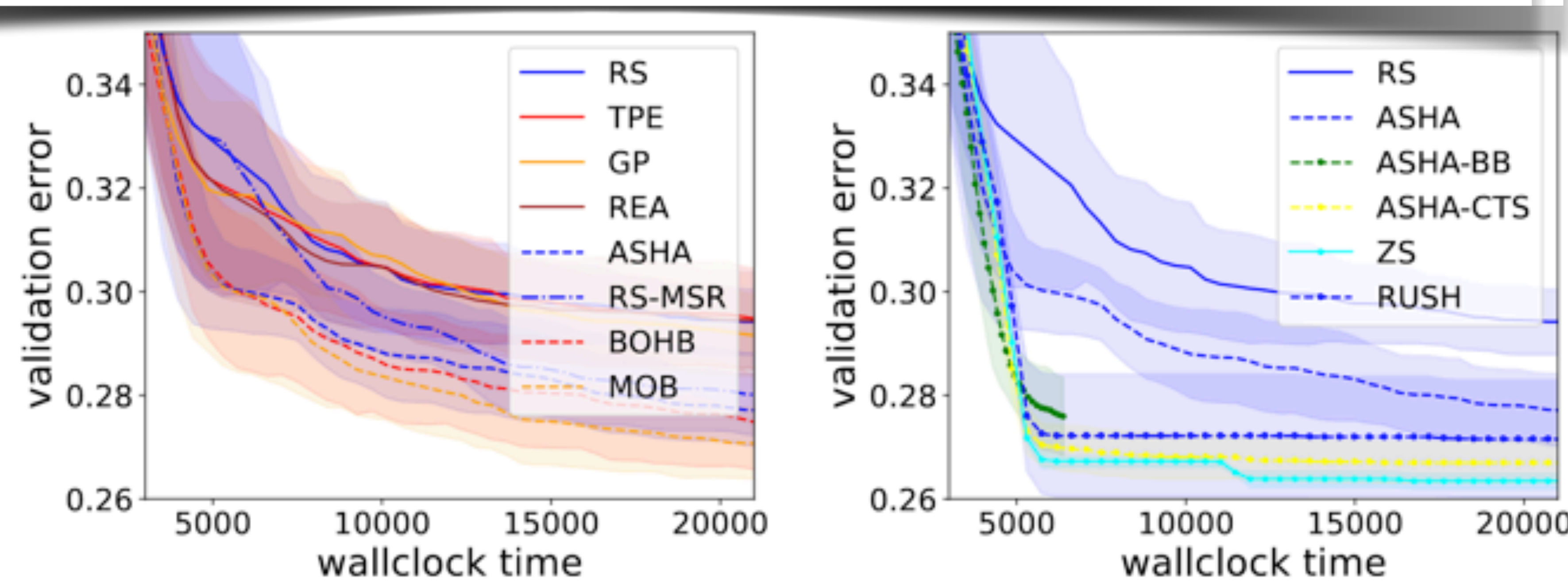
$$j_1 = \text{argmin}_{j_1 \in [m]} \frac{1}{n} \sum_{i=1}^{n} \varepsilon_{ij_1}, \qquad j_n = \text{argmin}_{j_n \in [m]} \frac{1}{n} \sum_{i=1}^{n} \min$$

Pick the model performing best on average

Pick the model performing best on average when combined with the ones previously selected

**Benefits** 👍:
- Approximation guarantees from the original (sub-modular) problem
- Tractable
- Works extremely well in practice

**My favourite transfer learning algorithm** ❤️ ❤️ ❤️

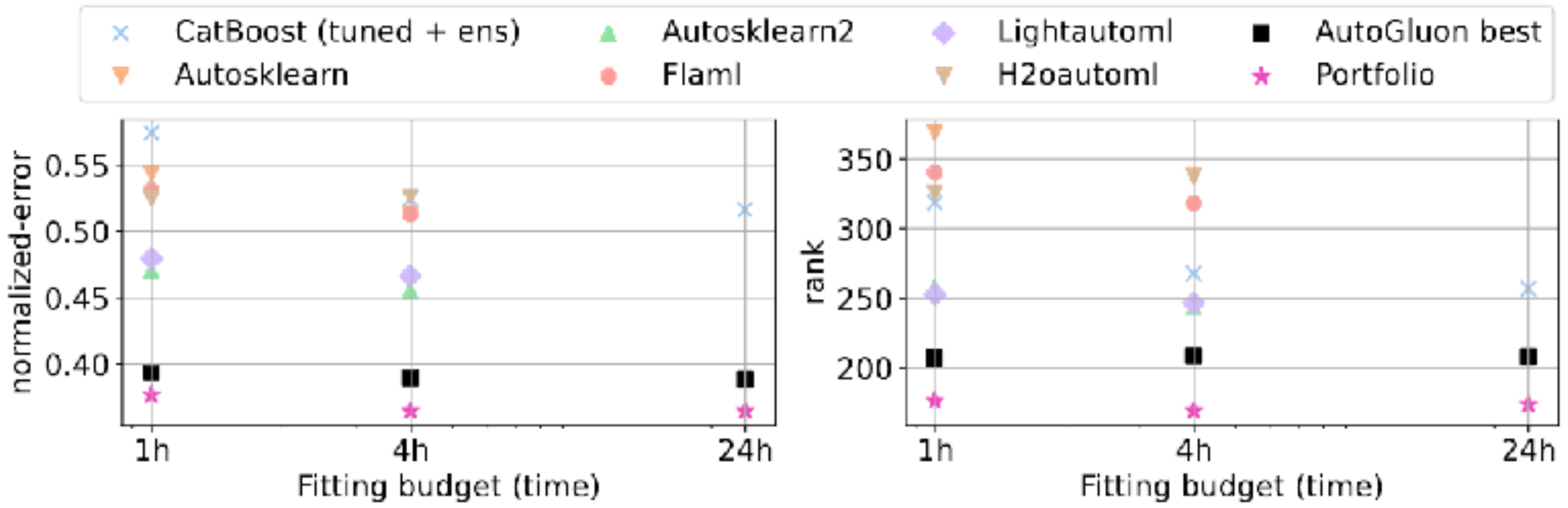



(b) NAS201: CIFAR-100.

Syne Tune: A Library for Large-Scale Hyperparameter Tuning and Reproducible Research [Salinas 2022]

# Results

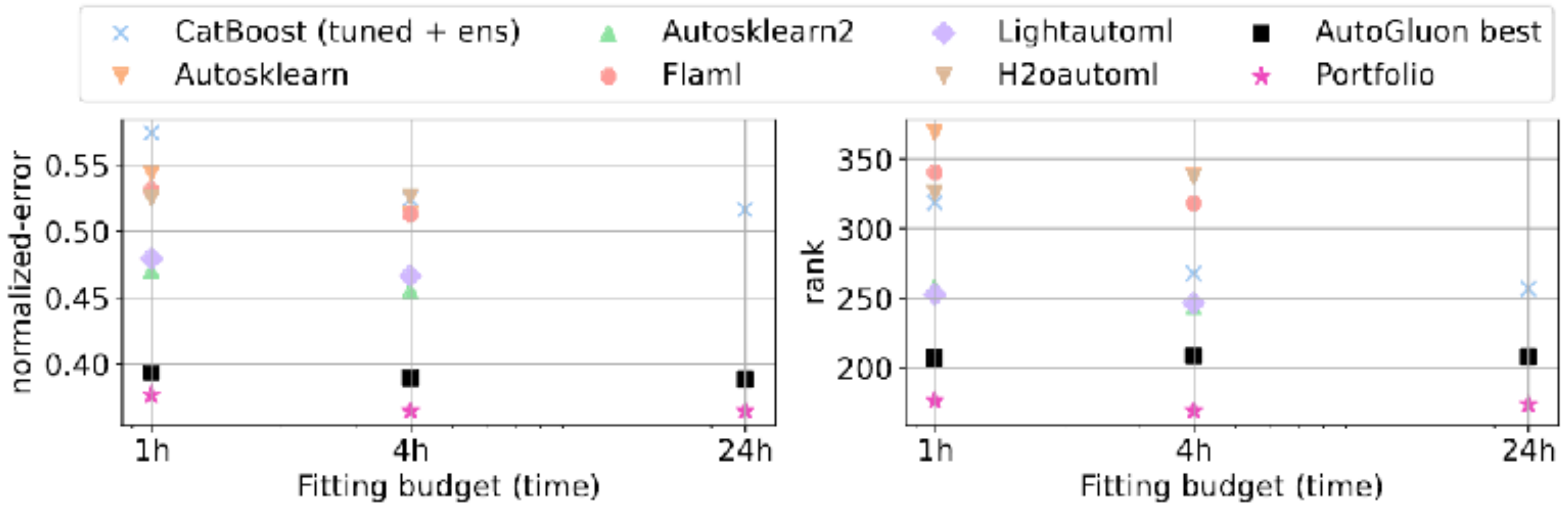
# Results



CatBoost (tuned + ens)
Autosklearn
Autosklearn2
Flaml
Lightautoml
H2oautoml
AutoGluon best
Portfolio
normalized-error
0.55
0.50
0.45
0.40
1h
4h
24h
Fitting budget (time)
rank
350
300
250
200
1h
4h
24h
Fitting budget (time)

# Results

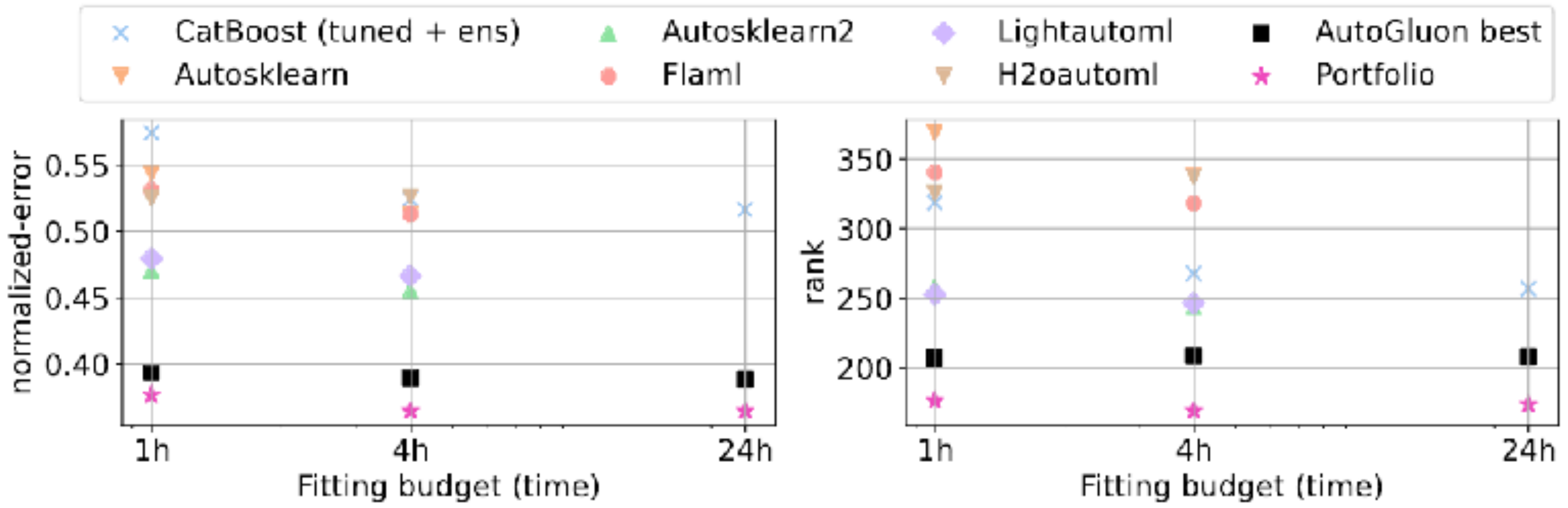
- Just fitting portfolio configuration on evaluations of TabRepo outperforms all SOTA tabular methods studied



# Results

- Just fitting portfolio configuration on evaluations of TabRepo outperforms all SOTA tabular methods studied

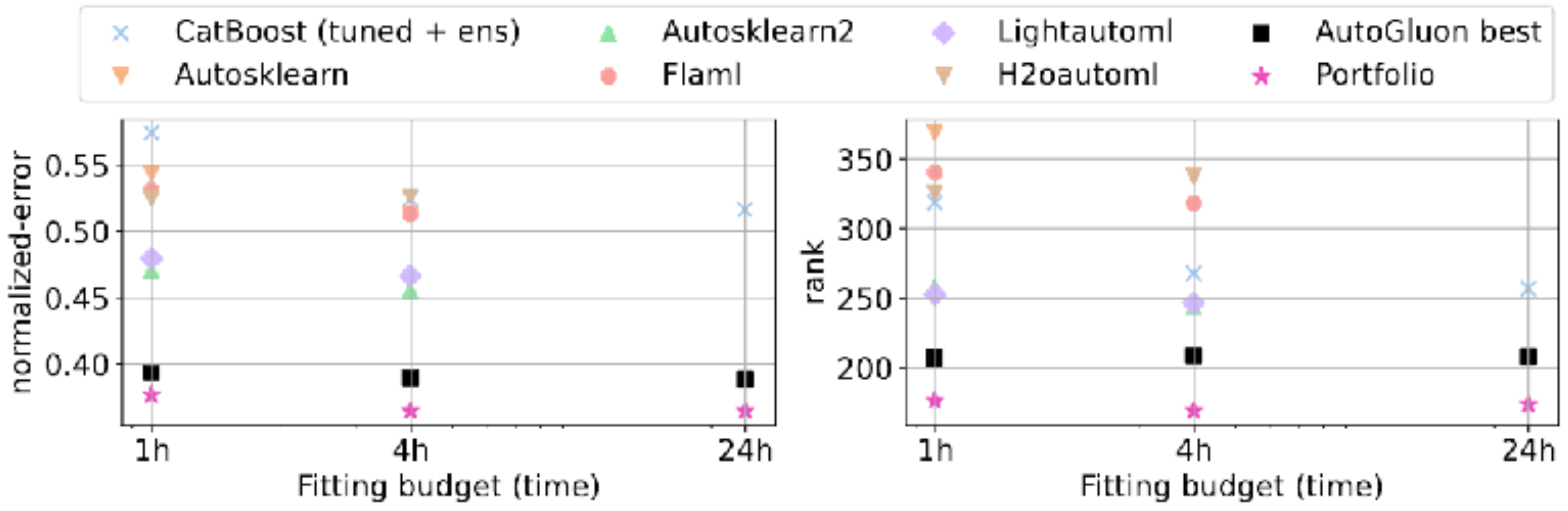
- We can analyse the performance of various components: ensembling, #configurations, #datasets, #configurations



# Results

- Just fitting portfolio configuration on evaluations of TabRepo outperforms all SOTA tabular methods studied

- We can analyse the performance of various components: ensembling, #configurations, #datasets, #configurations



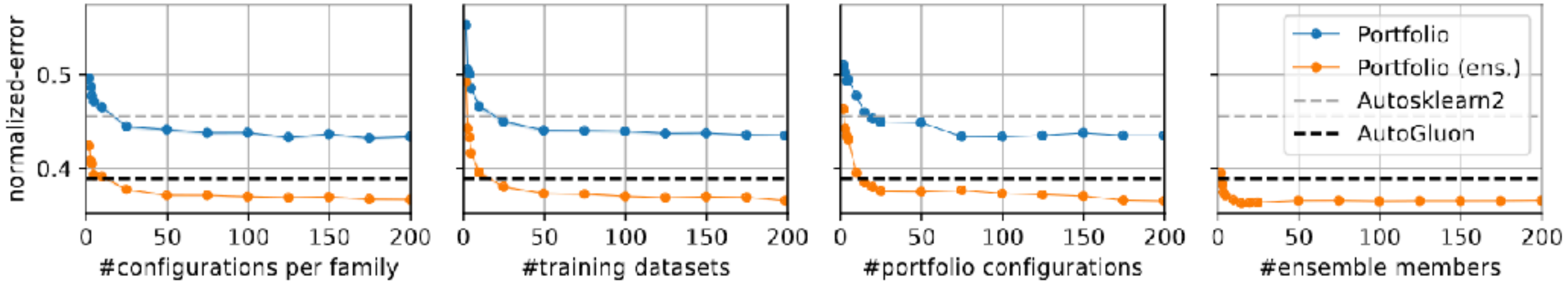



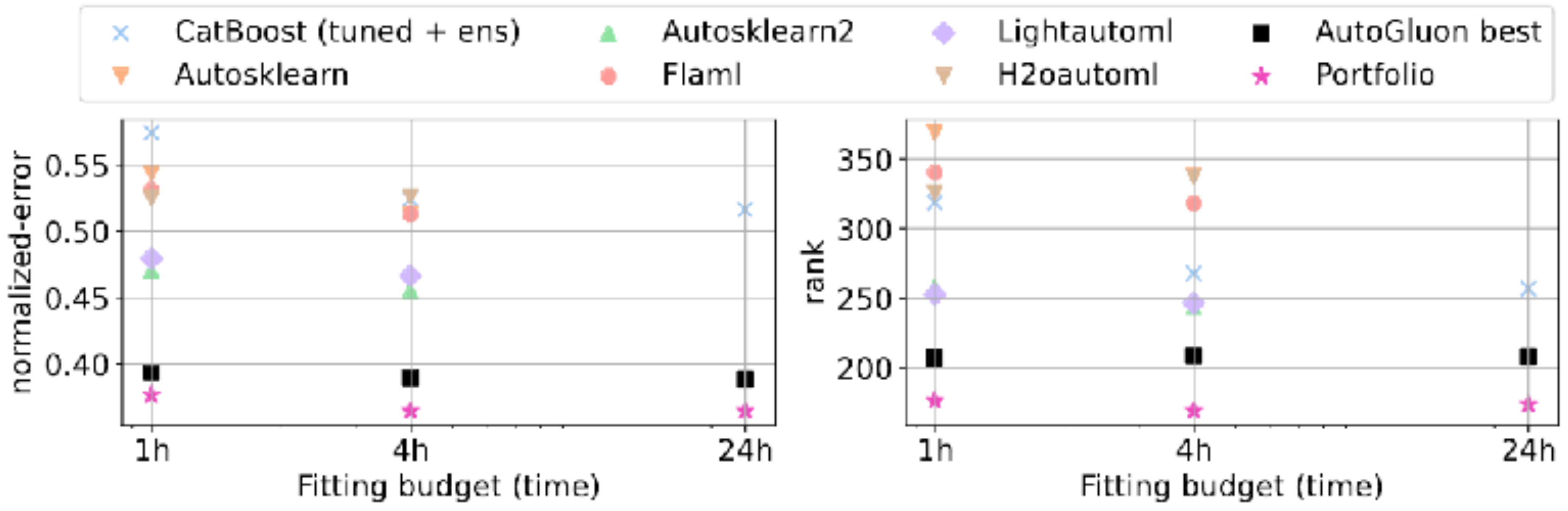

Figure 4: Impact on normalized error when varying the (a) number of configurations per family, (b) number of training datasets, (c) portfolio size and (d) number of ensemble members.

# Results

- Just fitting portfolio configuration on evaluations of TabRepo outperforms all SOTA tabular methods studied

- We can analyse the performance of various components: ensembling, #configurations, #datasets, #configurations

- Portfolio configurations has replaced the manually configured defaults and improved significantly AutoGluon



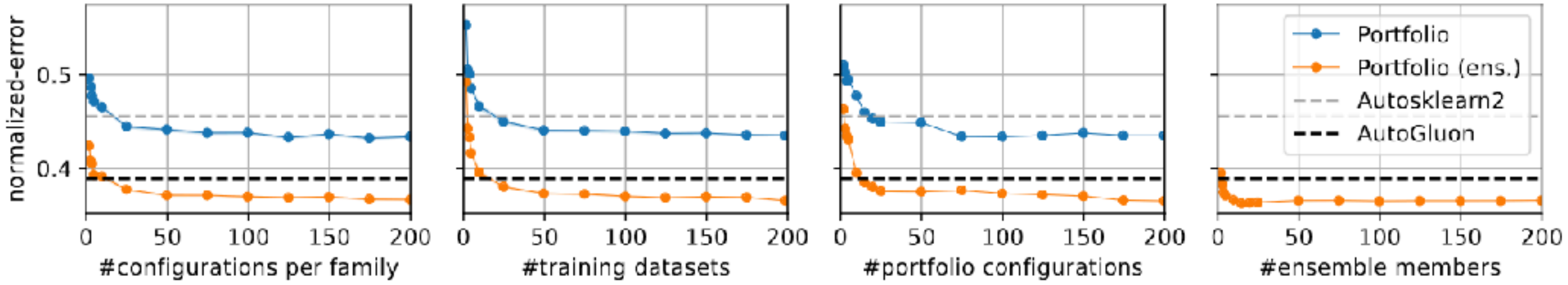




Figure 4: Impact on normalized error when varying the (a) number of configurations per family, (b) number of training datasets, (c) portfolio size and (d) number of ensemble members.

# Results

- Just fitting portfolio configuration on evaluations of TabRepo outperforms all SOTA tabular methods studied
- We can analyse the performance of various components: ensembling, #configurations, #datasets, #configurations
- Portfolio configurations has replaced the manually configured defaults and improved significantly AutoGluon

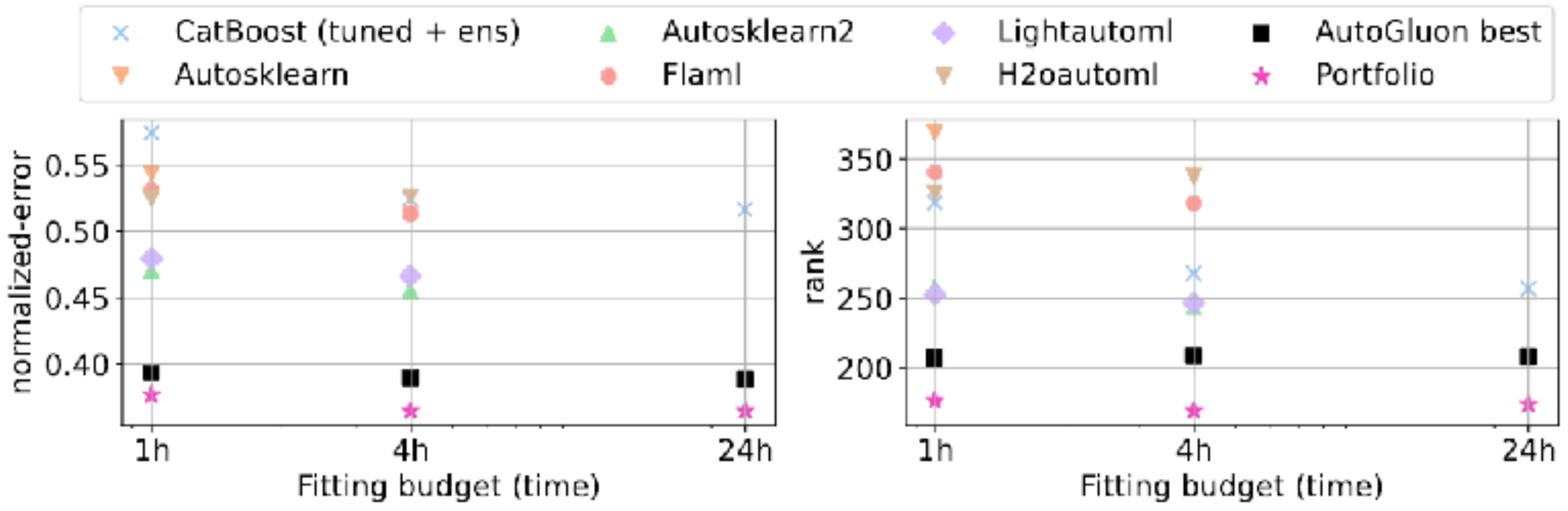



Table 2: Performance of AutoGluon combined with portfolios on AMLB.

| method | win-rate | loss reduc. |
|---|---|---|
| **AG + Portfolio (ours)** | - | 0% |
| AG | 67% | 2.8% |
| MLJAR | 81% | 22.5% |
| lightautoml | 83% | 11.7% |
| GAMA | 86% | 15.5% |
| FLAML | 87% | 16.3% |
| autosklearn | 89% | 11.8% |
| H2OAutoML | 92% | 10.3% |
| CatBoost | 94% | 18.1% |
| TunedRandomForest | 94% | 22.9% |
| RandomForest | 97% | 25.0% |
| XGBoost | 98% | 20.9% |
| LightGBM | 98% | 23.6% |

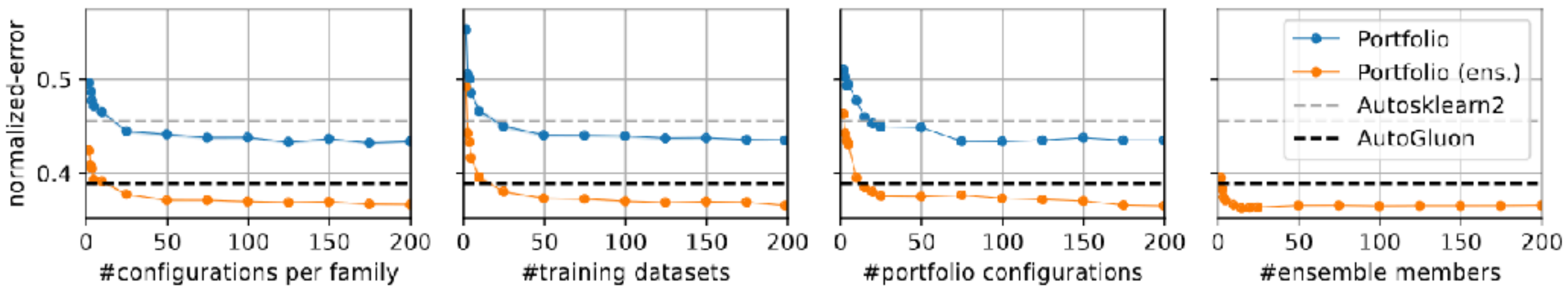



Figure 4: Impact on normalized error when varying the (a) number of configurations per family, (b) number of training datasets, (c) portfolio size and (d) number of ensemble members.

# Results

# Results

- 🥳 All those experiments (fitting portfolio and evaluating) can be done using TabRepo for a very small cost (e.g. many table lookups)

# Results

- 🥳 All those experiments (fitting portfolio and evaluating) can be done using TabRepo for a very small cost (e.g. many table lookups)
- High value use-case to apply meta-heuristics or research idea at very low computational cost!

# Results

- 🥳 All those experiments (fitting portfolio and evaluating) can be done using TabRepo for a very small cost (e.g. many table lookups)
- High value use-case to apply meta-heuristics or research idea at very low computational cost!
- Possible research ideas:

# Results

- 🥳 All those experiments (fitting portfolio and evaluating) can be done using TabRepo for a very small cost (e.g. many table lookups)
- High value use-case to apply meta-heuristics or research idea at very low computational cost!
- Possible research ideas:
    - Find best tabular configurations given time budget

# Results

- 🥳 All those experiments (fitting portfolio and evaluating) can be done using TabRepo for a very small cost (e.g. many table lookups)
- High value use-case to apply meta-heuristics or research idea at very low computational cost!
- Possible research ideas:
  - Find best tabular configurations given time budget
  - Apply different meta-heuristics to optimise the learned default portfolio list of configurations on a new dataset

# Results

- 🥳 All those experiments (fitting portfolio and evaluating) can be done using TabRepo for a very small cost (e.g. many table lookups)
- High value use-case to apply meta-heuristics or research idea at very low computational cost!
- Possible research ideas:
  - Find best tabular configurations given time budget
  - Apply different meta-heuristics to optimise the learned default portfolio list of configurations on a new dataset
  - Multiobjective optimization taking latency into account…

# Results

- 🥳 All those experiments (fitting portfolio and evaluating) can be done using TabRepo for a very small cost (e.g. many table lookups)
- High value use-case to apply meta-heuristics or research idea at very low computational cost!
- Possible research ideas:
  - Find best tabular configurations given time budget
  - Apply different meta-heuristics to optimise the learned default portfolio list of configurations on a new dataset
  - Multiobjective optimization taking latency into account…
  - All those experiments can be done… with your laptop!!

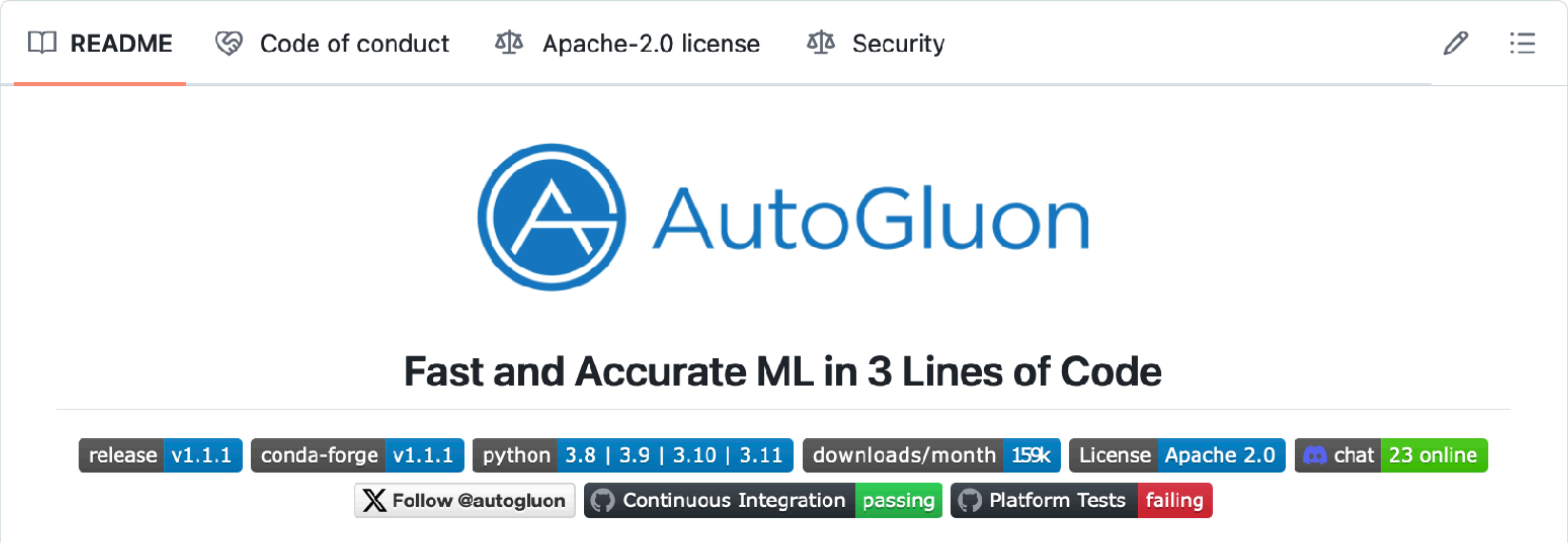
# Results

- 🥳 All those experiments (fitting portfolio and evaluating) can be done using TabRepo for a very small cost (e.g. many table lookups)
- High value use-case to apply meta-heuristics or research idea at very low computational cost!
- Possible research ideas:
  - Find best tabular configurations given time budget
  - Apply different meta-heuristics to optimise the learned default portfolio list of configurations on a new dataset
  - Multiobjective optimization taking latency into account…
  - All those experiments can be done… with your laptop!!
- 🧑‍💻 https://github.com/autogluon/tabrepo

# Results

- 🥳 All those experiments (fitting portfolio and evaluating) can be done using TabRepo for a very small cost (e.g. many table lookups)
- High value use-case to apply meta-heuristics or research idea at very low computational cost!
- Possible research ideas:
  - Find best tabular configurations given time budget
  - Apply different meta-heuristics to optimise the learned default portfolio list of configurations on a new dataset
  - Multiobjective optimization taking latency into account…
  - All those experiments can be done… with your laptop!!
- 🧑‍💻 https://github.com/autogluon/tabrepo
- Quick demo

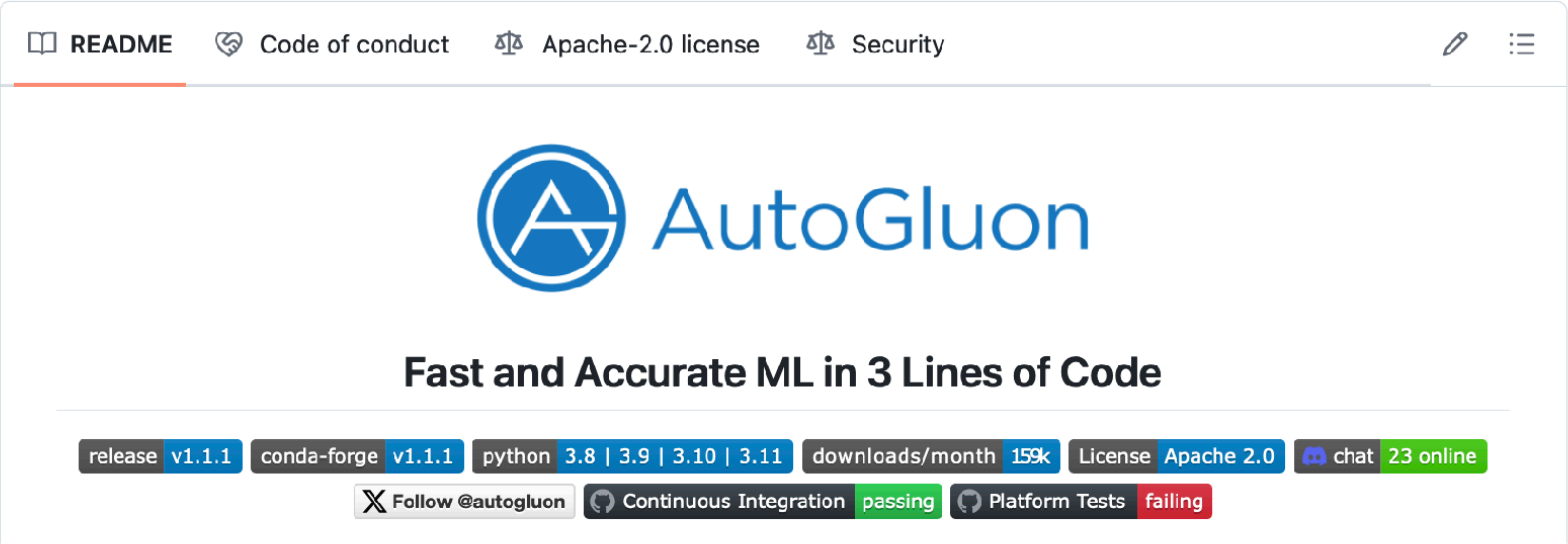
# Try it out for your self!




https://github.com/autogluon/autogluon

# Try it out for your self!

- State of the art for tabular prediction and time series forecasting




https://github.com/autogluon/autogluon

# Code and libraries

# Code

## HPO Libraries that support transfer learning

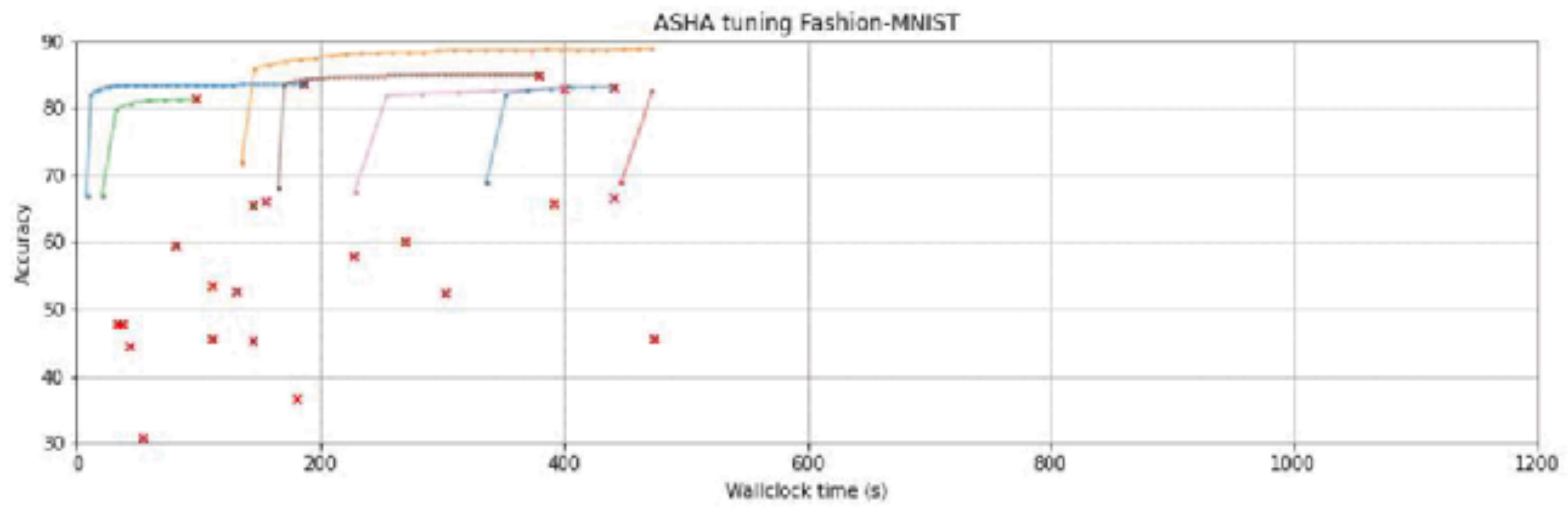
# Code

## HPO Libraries that support transfer learning

README | Code of conduct | Apache-2.0 license | License | Security

### Syne Tune: Large-Scale and Reproducible Hyperparameter Optimization

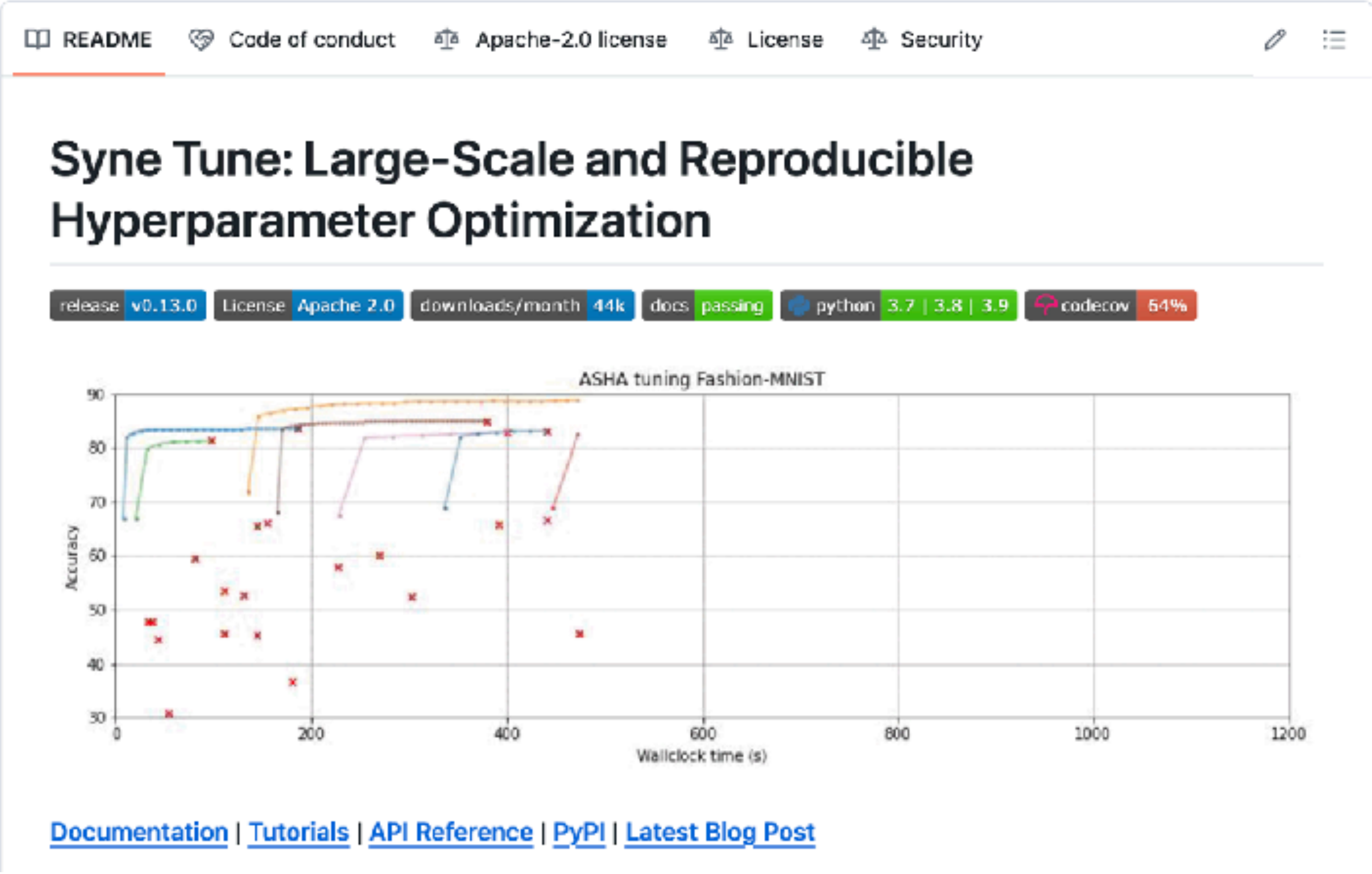
release v0.13.0 | license Apache 2.0 | downloads/month 44k | docs passing | python 3.7 | 3.8 | 3.9 | codecov 64%




Documentation | Tutorials | API Reference | PyPI | Latest Blog Post

# Code

## HPO Libraries that support transfer learning



README | Code of conduct | Apache-2.0 license | License | Security

### Syne Tune: Large-Scale and Reproducible Hyperparameter Optimization

release v0.13.0 | license Apache 2.0 | downloads/month 44k | docs passing | python 3.7 | 3.8 | 3.9 | codecov 64%



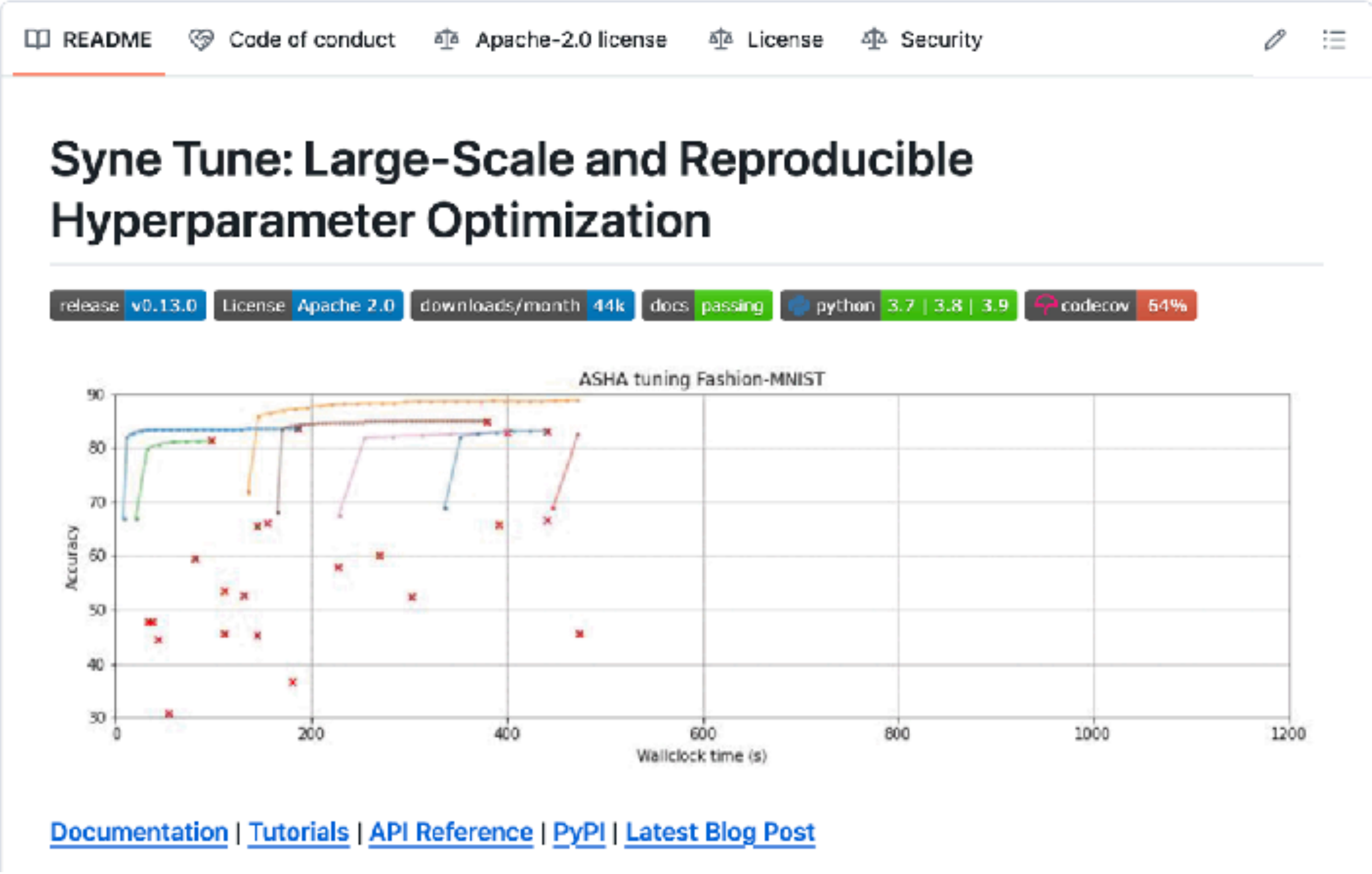
Documentation | Tutorials | API Reference | PyPI | Latest Blog Post

**Syne Tune**

- ZeroShot/Portfolio
- CTS
- RUSH
- Bounding-box
- BO+WarmStart

# Code

## HPO Libraries that support transfer learning



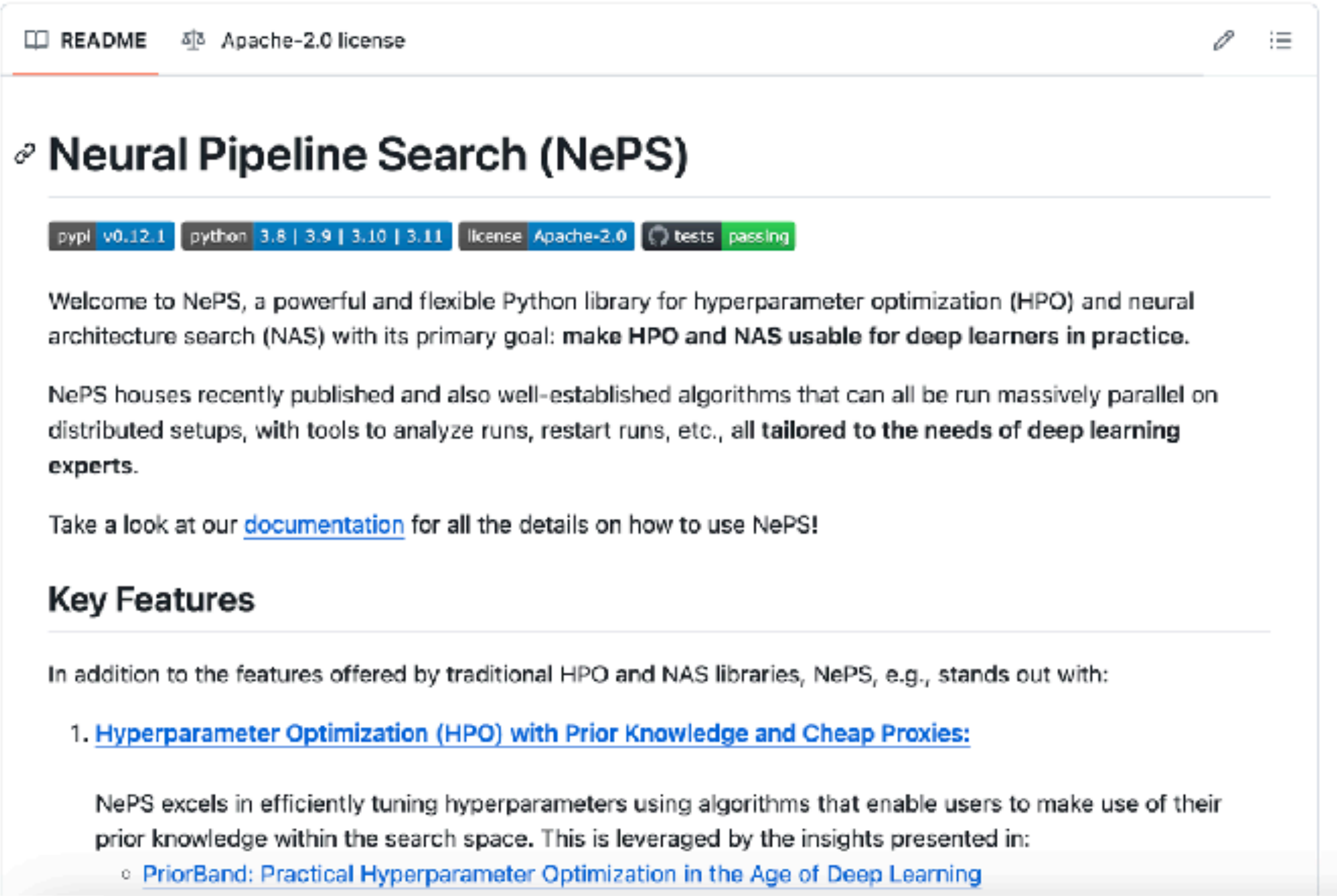
README | Code of conduct | Apache-2.0 license | License | Security

### Syne Tune: Large-Scale and Reproducible Hyperparameter Optimization

release v0.13.0 | License Apache 2.0 | downloads/month 44k | docs passing | python 3.7 | 3.8 | 3.9 | codecov 64%



Documentation | Tutorials | API Reference | PyPI | Latest Blog Post



README | Apache-2.0 license

### Neural Pipeline Search (NePS)

pypi v0.12.1 | python 3.8 | 3.9 | 3.10 | 3.11 | license Apache-2.0 | tests passing

Welcome to NePS, a powerful and flexible Python library for hyperparameter optimization (HPO) and neural architecture search (NAS) with its primary goal: **make HPO and NAS usable for deep learners in practice.**

NePS houses recently published and also well-established algorithms that can all be run massively parallel on distributed setups, with tools to analyze runs, restart runs, etc., all **tailored to the needs of deep learning experts.**

Take a look at our documentation for all the details on how to use NePS!

#### Key Features

In addition to the features offered by traditional HPO and NAS libraries, NePS, e.g., stands out with:

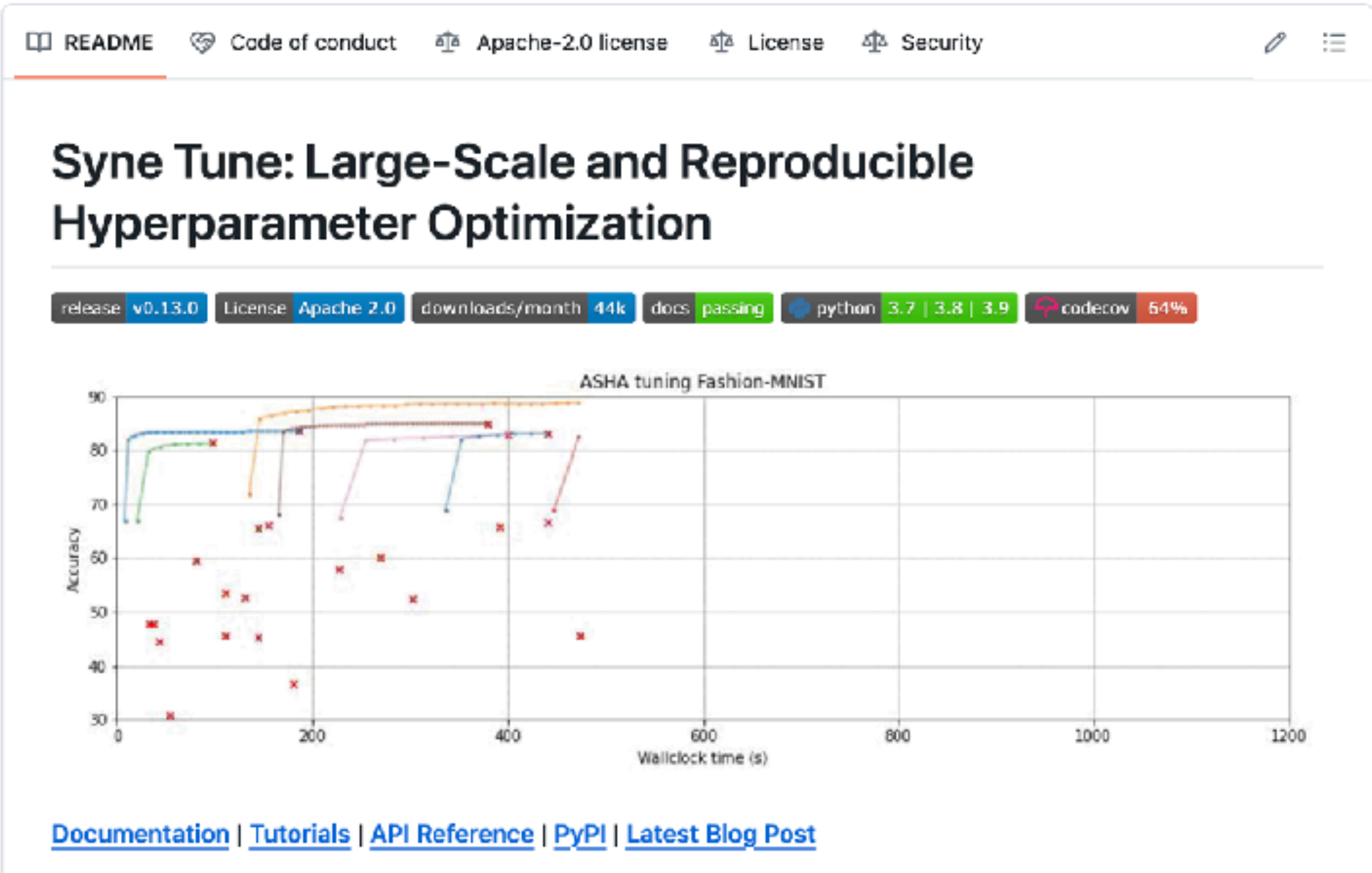
1. **Hyperparameter Optimization (HPO) with Prior Knowledge and Cheap Proxies:**

   NePS excels in efficiently tuning hyperparameters using algorithms that enable users to make use of their prior knowledge within the search space. This is leveraged by the insights presented in:
   - PriorBand: Practical Hyperparameter Optimization in the Age of Deep Learning

**Syne Tune**
- ZeroShot/Portfolio
- CTS
- RUSH
- Bounding-box
- BO+WarmStart

# Code

## HPO Libraries that support transfer learning

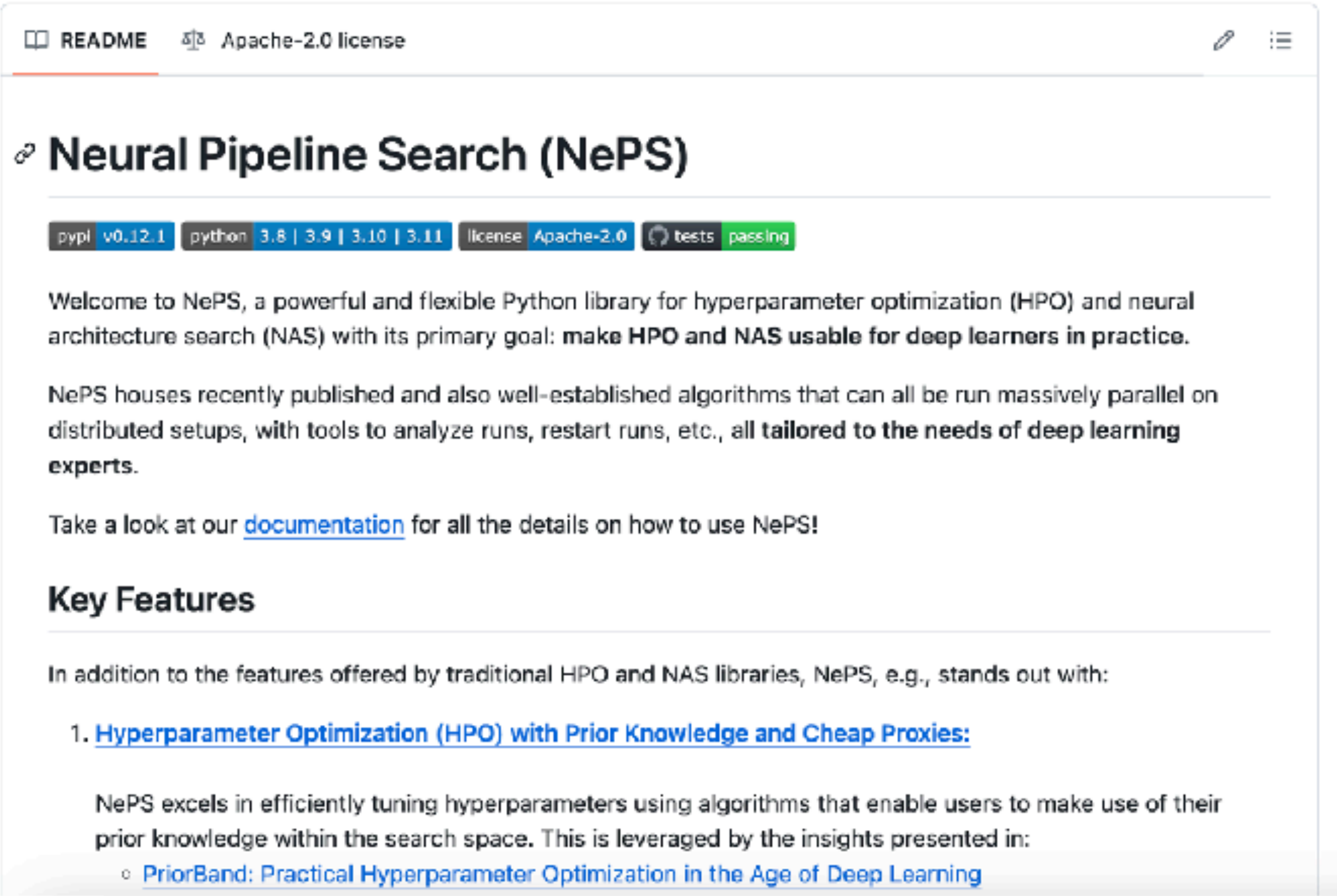

README | Code of conduct | Apache-2.0 license | License | Security

Syne Tune: Large-Scale and Reproducible Hyperparameter Optimization

release v0.13.0 | License Apache 2.0 | downloads/month 44k | docs passing | python 3.7 | 3.8 | 3.9 | codecov 64%



Documentation | Tutorials | API Reference | PyPI | Latest Blog Post


README | Apache-2.0 license

Neural Pipeline Search (NePS)

pypi v0.12.1 | python 3.8 | 3.9 | 3.10 | 3.11 | license Apache-2.0 | tests passing

Welcome to NePS, a powerful and flexible Python library for hyperparameter optimization (HPO) and neural architecture search (NAS) with its primary goal: **make HPO and NAS usable for deep learners in practice.**

NePS houses recently published and also well-established algorithms that can all be run massively parallel on distributed setups, with tools to analyze runs, restart runs, etc., all **tailored to the needs of deep learning experts.**

Take a look at our documentation for all the details on how to use NePS!

Key Features

In addition to the features offered by traditional HPO and NAS libraries, NePS, e.g., stands out with:

1. **Hyperparameter Optimization (HPO) with Prior Knowledge and Cheap Proxies:**

   NePS excels in efficiently tuning hyperparameters using algorithms that enable users to make use of their prior knowledge within the search space. This is leveraged by the insights presented in:
   - PriorBand: Practical Hyperparameter Optimization in the Age of Deep Learning

**Syne Tune**
- ZeroShot/Portfolio
- CTS
- RUSH
- Bounding-box
- BO+WarmStart

**NEPS**
- Prior band

# Conclusion

# Conclusion

# Conclusion

- Transfer learning in HPO works by reusing previous evaluations

# Conclusion

- Transfer learning in HPO works by reusing previous evaluations
- Many methods

# Conclusion

- Transfer learning in HPO works by reusing previous evaluations
- Many methods
  - Adapt BO: prune search space, warm-start, learn priors…

# Conclusion

- Transfer learning in HPO works by reusing previous evaluations
- Many methods
  - Adapt BO: prune search space, warm-start, learn priors…
  - Learning curve prediction

# Conclusion

- Transfer learning in HPO works by reusing previous evaluations
- Many methods
  - Adapt BO: prune search space, warm-start, learn priors…
  - Learning curve prediction
  - Foundational approach

# Conclusion

- Transfer learning in HPO works by reusing previous evaluations
- Many methods
  - Adapt BO: prune search space, warm-start, learn priors…
  - Learning curve prediction
  - Foundational approach
- Transfer learning can speed HPO significantly!