EBERHARD KARLS UNIVERSITÄT TÜBINGEN

machine learning new perspectives for science

Seminar WS23/24

Automated Machine Learning and Hyperparameter Optimization

Katharina Eggensperger

/KEggensperger
katharina.eggensperger@uni-tuebingen.de
AutoML for Science

October 18th, 2023

AutoML.org

Story Line Today

[30min] **Big Picture**

[?] Your *Questions*

[10min] **Organization**

[?] Your *Questions*

[15min] **Topics/Papers**

[?] Your *Questions*

[20min; if time left] **How to give a good presentation**

The Big Picture

>> What is this about?

Machine learning is this …

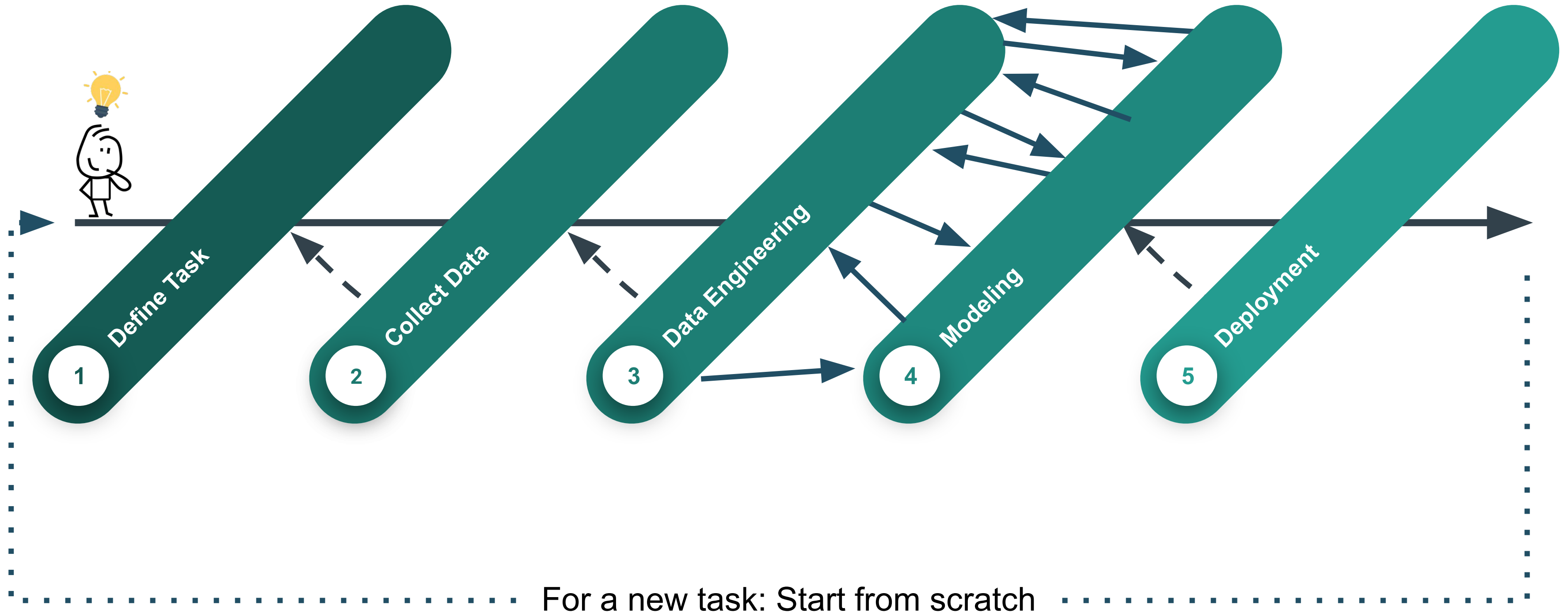
“Machine learning is the science of getting computers to act without being explicitly programmed.”

by Andrew Ng
(probably inspired by Arthur Samuels)

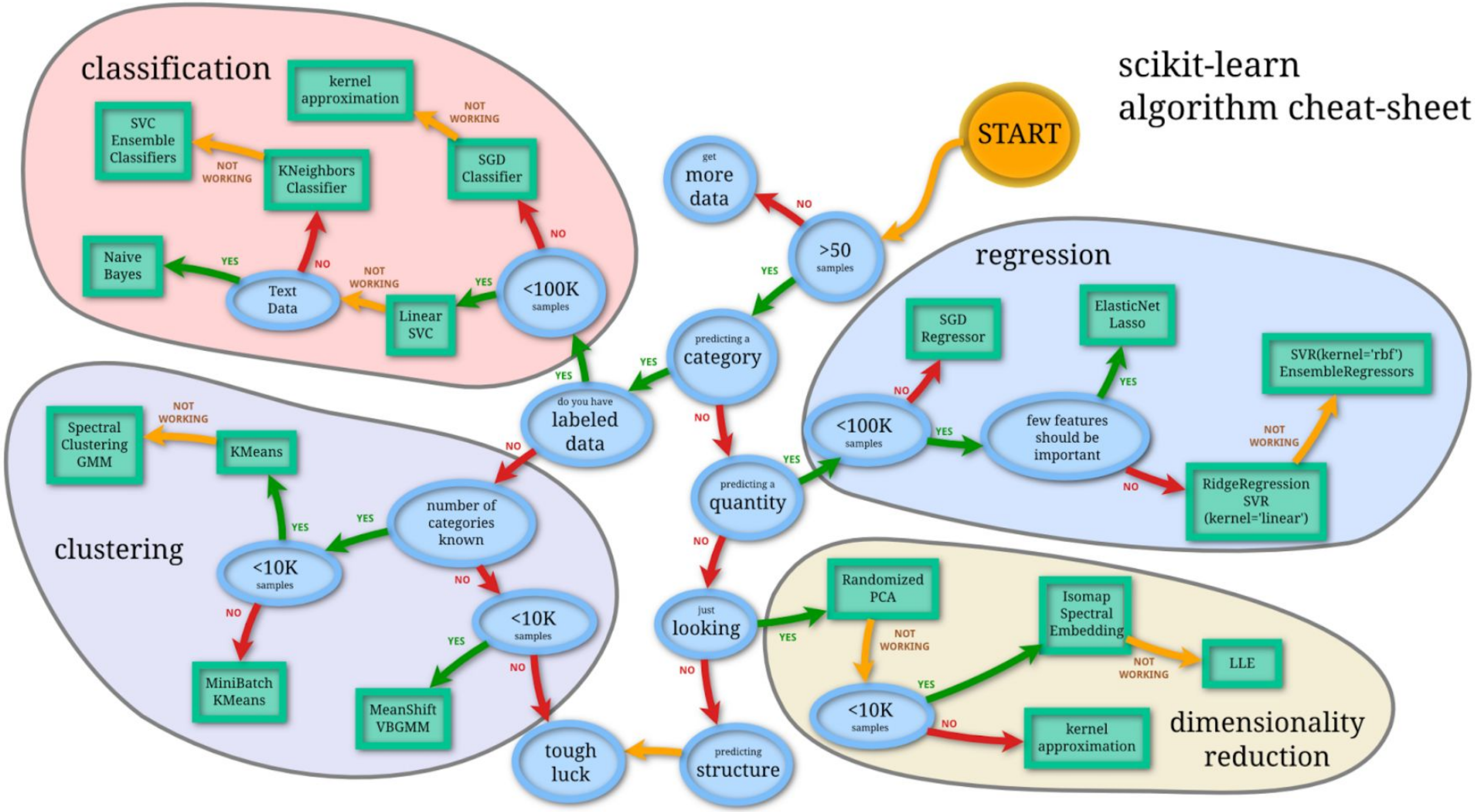
… and also this

source: XKDC

Why does ML development take a lot of time?



Design Decisions

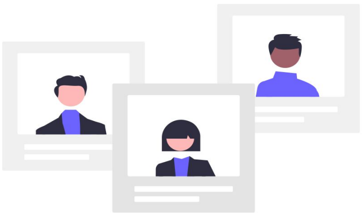


source: https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

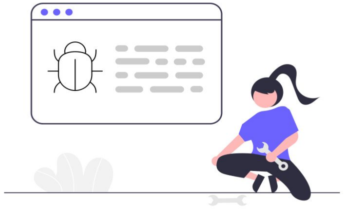
Challenges in Applying AI/ML these days

Required expertise in ML and AI

Long development time for new AI applications

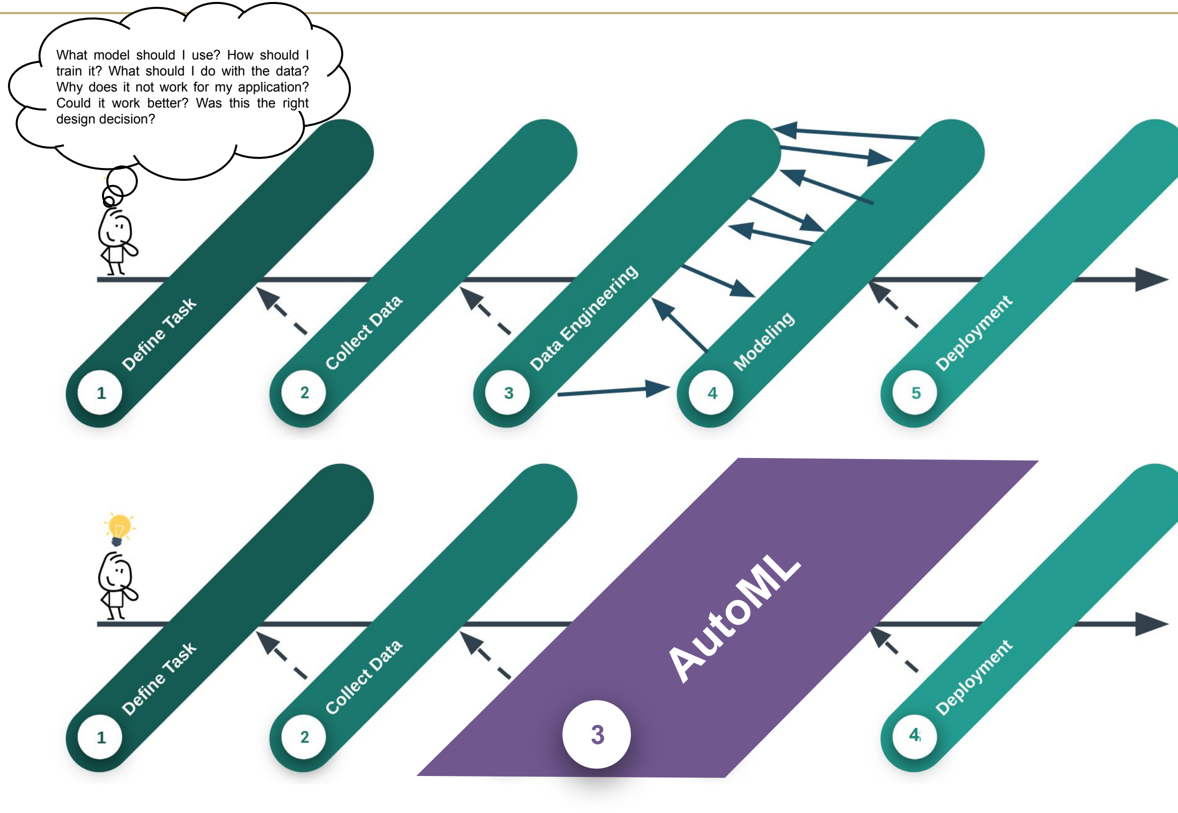


Few experts are available on the job market



Unstructured and error-prone development of AI application

ML vs AutoML


What model should I use? How should I train it? What should I do with the data? Why does it not work for my application? Could it work better? Was this the right design decision?
1
Define Task
2
Collect Data
3
Data Engineering
4
Modeling
5
Deployment
1
Define Task
2
Collect Data
3
AutoML
4
Deployment

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Advantages

machine learning new perspectives for science

AutoML enables

- 🚀 More **efficient** research (and development of ML applications)
  - → AutoML has been shown to outperform humans on subproblems
- 🧮 More **systematic** research (and development of ML applications)
  - → no (human) bias or unsystematic evaluation
- 📋 More **reproducible / robust** research
  - → since it is systematic!
- 🦾 **Broader use** of ML methods
  - → less required ML expert knowledge
  - → not only limited to computer scientists

Challenges For AutoML

But, it is not that easy, because

🔁 Each dataset potentially requires **different optimal ML-designs**
→ Design decisions have to be made for each dataset again

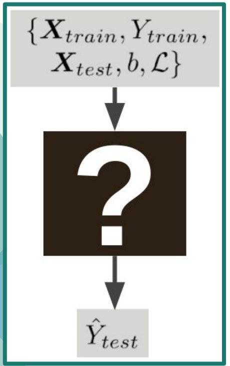
⌛ Training of a single ML model can be **quite expensive**
→We can not try many configurations

❓Mathematical **relation** between design and performance is (often) **unknown**
→ Gradient-based optimization not easily possible

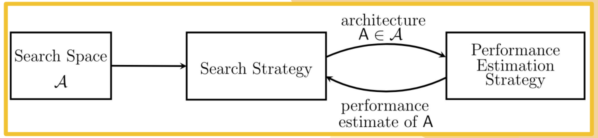
🎛️ 🎚️ Optimization in **highly complex spaces**
→ including categorical, continuous and conditional dependencies

AutoML Methods?

AutoML Systems



Neural Architecture Search



(Hyper-) Parameter Optimization

$$\lambda^* \in \arg\min_{\lambda \in \Lambda} f(\mathcal{A}_\lambda)$$

Hyperparameters

scikit learn

Home | Installation | Documentation | Examples

Google Custom Search

Previous sklearn.svm.O... | Next sklearn.svm.SVR | Up API Reference

scikit-learn v0.20.3
Other versions

Please **cite us** if you use the software.

`sklearn.svm.SVC`
Examples using `sklearn.svm.SVC`

sklearn.svm.SVC

class `sklearn.svm.` **SVC** (*C=1.0, kernel='rbf', degree=3, gamma='auto_deprecated', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None*) [source]

C-Support Vector Classification.

The implementation is based on libsvm. The fit time complexity is more than quadratic with the number of samples which makes it hard to scale to dataset with more than a couple of 10000 samples.

The multiclass support is handled according to a one-vs-one scheme.

For details on the precise mathematical formulation of the provided kernel functions and how gamma, coef0 and degree affect each other, see the corresponding section in the narrative documentation: Kernel functions.

Read more in the User Guide.

Parameters:

C : ***float, optional (default=1.0)***
Penalty parameter C of the error term.

kernel : ***string, optional (default='rbf')***
Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`.

degree : ***int, optional (default=3)***
Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

gamma : ***float, optional (default='auto')***
Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

Current default is 'auto' which uses 1 / n_features, if `gamma='scale'` is passed then it uses 1 / (n_features * X.var()) as value of gamma. The current default of gamma, 'auto', will change to 'scale' in version 0.22. 'auto_deprecated', a deprecated version of 'auto' is used as a default indicating that no explicit value of gamma was passed.

coef0 : ***float, optional (default=0.0)***
Independent term in kernel function. It is only significant in 'poly' and 'sigmoid'.

shrinking : ***boolean, optional (default=True)***
Whether to use the shrinking heuristic.

probability : ***boolean, optional (default=False)***
Whether to enable probability estimates. This must be enabled prior to calling fit, and will slow

PyTorch

SGD

CLASS torch.optim.SGD(*params, lr=<required parameter>, momentum=0, dampening=0, weight_decay=0, nesterov=False, *, maximize=False, foreach=None, differentiable=False*) [SOURCE]

Implements stochastic gradient descent (optionally with momentum).

Parameters:

- **params** (*iterable*) – iterable of parameters to optimize or dicts defining parameter groups
- **lr** (*float*) – learning rate
- **momentum** (*float, optional*) – momentum factor (default: 0)
- **weight_decay** (*float, optional*) – weight decay (L2 penalty) (default: 0)
- **dampening** (*float, optional*) – dampening for momentum (default: 0)
- **nesterov** (*bool, optional*) – enables Nesterov momentum (default: False)
- **maximize** (*bool, optional*) – maximize the params based on the objective, instead of minimizing (default: False)
- **foreach** (*bool, optional*) – whether foreach implementation of optimizer is used. If unspecified by the user (so foreach is None), we will try to use foreach over the for-loop implementation on CUDA, since it is usually significantly more performant. (default: None)
- **differentiable** (*bool, optional*) – whether autograd should occur through the optimizer step in training. Otherwise, the step() function runs in a torch.no_grad() context. Setting to True can impair performance, so leave it False if you don't intend to run autograd through this instance (default: False)

Definition

Let

- $\boldsymbol{\lambda}$ be the hyperparameters of an ML algorithm $\mathcal{A}$ with domain $\boldsymbol{\Lambda}$,
- $\mathcal{D}_{opt}$ be a dataset which is split into $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$
- $c(\mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ denote the cost of $\mathcal{A}_{\boldsymbol{\lambda}}$ trained on $\mathcal{D}_{\text{train}}$ and evaluated on $\mathcal{D}_{\text{val}}$.

The *hyper-parameter optimization (HPO)* problem is to find a hyper-parameter configuration that minimizes this cost:

$$\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} c(\mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$$

Remarks:

- $\arg\min$ returns a set of optimal points of a given function. It suffices to find one element of this set and thus we use $\in$ instead of $=$.
- Sometimes, we want to optimize for different metrics, instead of one
  $\leadsto$ multi-objective optimization and Pareto fronts

What?

Optimization Methods?

- Bayesian Optimization (see intro next week and session #1, #2)
- Multi-fidelity Optimization (see session #3)
- Evolutionary Approaches
- Reinforcement Learning

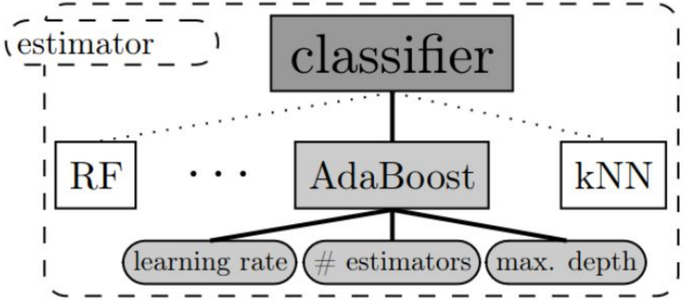
Open Challenges?

- Scale to high dimensions (see #2)
- Efficiently optimize expensive objectives (see #3)
- Optimize Hyper-hyper parameters
- Handle mixed search spaces
- Re-use experience on similar tasks
- Incorporate Expert knowledge

Choosing an Algorithm

- Many ML-algorithms exist
- Most of these (still) have a reason for existence
- Examples for classification:
  - logistic regression
  - k-nearest neighbor
  - naïve Bayes
  - random forest
  - gradient boosting
  - multi-layer perceptron
  - SVM
  - decision tree
  - residual networks
- [Fernández-Delgado et al. 2014] studied 179 classifiers on 121 datasets

→ In practice: We want to jointly choose the best ML-algorithm **and** its hyperparameters



CASH: Combined Algorithm Selection and Hyperparameter Optimization

[Thornton et al. 2013]

Definition

Let

- $\mathbf{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k\}$ be a set of algorithms (a.k.a. portfolio)
- $\boldsymbol{\Lambda}$ be a set of hyperparameters of each machine learning algorithm $\mathcal{A}_i$
- $\mathcal{D}_{opt}$ be a dataset which is split into $\mathcal{D}_{train}$ and $\mathcal{D}_{valid}$
- $c(\mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ denote the cost of $\mathcal{A}_{\boldsymbol{\lambda}}$ trained on $\mathcal{D}_{train}$ and evaluated on $\mathcal{D}_{valid}$.

we want to find the best combination of algorithm $\mathcal{A} \in \mathbf{A}$ and its hyperparameter configuration $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ minimizing:

$$(\mathcal{A}^*, \boldsymbol{\lambda}^*) \in \arg\min_{\mathcal{A} \in \mathbf{A}, \boldsymbol{\lambda} \in \boldsymbol{\Lambda}} c(\mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$$

Please don't trust LLMs telling you that AutoML was invented by Google in 2017. Obviously wrong!

What?

Methods used in/for AutoML systems?

- Bayesian Optimization (and everything from the slide before; see #4)
- Ensembling (see #4)

Open Challenges (for tabular data)?

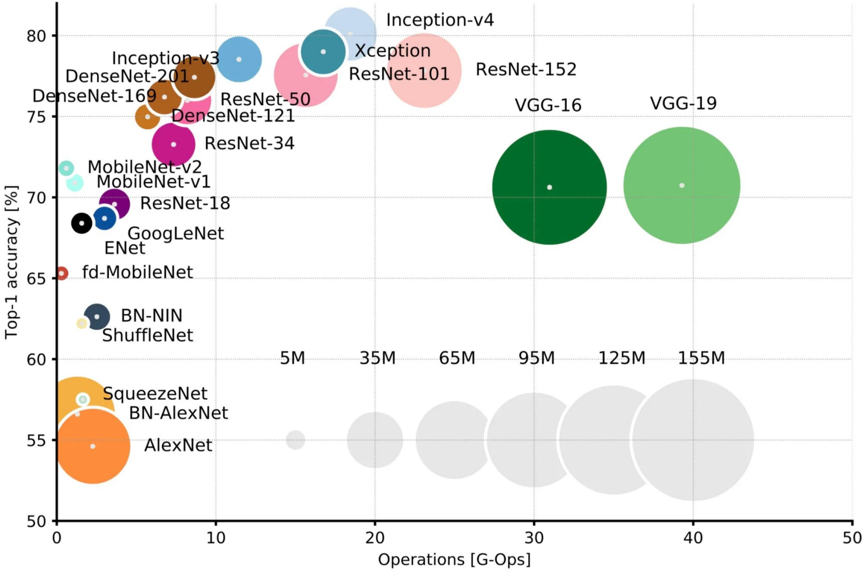
- Trees vs Nets (see #5)
- Add custom pipelines
- Optimize Hyper-hyperparameters
- Tune pre-processing (Auto Data Science)
- Handle non-tabular features (multi-modality)

Architectures of Neural Networks

Many architectures exist and differ in

- Depth
- Resolution
- Width
- Operators
- Connections
- ...

Already on a single dataset (e.g. ImageNet), it is **not obvious** which architecture to choose



- On **different** datasets → different architectures
- On **similar** datasets → scaled versions of known architectures (e.g. ImageNet and Cifar10)

Source: [Culurciello et al. 2018]

NAS: Neural Architecture Search

Definition

Let

- $\boldsymbol{\lambda}$ be an architecture for a deep neural network N with domain $\boldsymbol{\Lambda}$,
- $\mathcal{D}_{opt}$ be a dataset which is split into $\mathcal{D}_{train}$ and $\mathcal{D}_{valid}$
- $c(N_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ denote the cost of $N_{\boldsymbol{\lambda}}$ trained on $\mathcal{D}_{train}$ and evaluated on $\mathcal{D}_{valid}$.

The *neural architecture search (NAS)* problem is to find an architecture that minimizes this cost:

$$\boldsymbol{\lambda}^* \in \underset{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}}{\arg\min}\, c(N_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$$

Remarks:

- very similar to the HPO definition
- In practice, you want jointly optimize HPO and NAS [Zela et al. 2018]

How?

Methods used in/for NAS?

- Bayesian Optimization (and everything from the slides before)
- Multi-Objective Optimization (see #9)
- Zero-Cost Proxies (see #10)
- Evolutionary Algorithms (see #9)

Open Challenges?

- How to encode configurations (see intro NAS)
- One-Shot vs. Finetuning (see #9)
- Efficiently evaluate configurations
- Optimize Hyper-hyperparameters
- Handle different data types

Summary

HPO	Search for the best hyperparameter configuration of a ML algorithm
AutoML Systems / CASH	Search for the best combination of algorithm and hyperparameter configuration
NAS	Search for the architecture of neural network

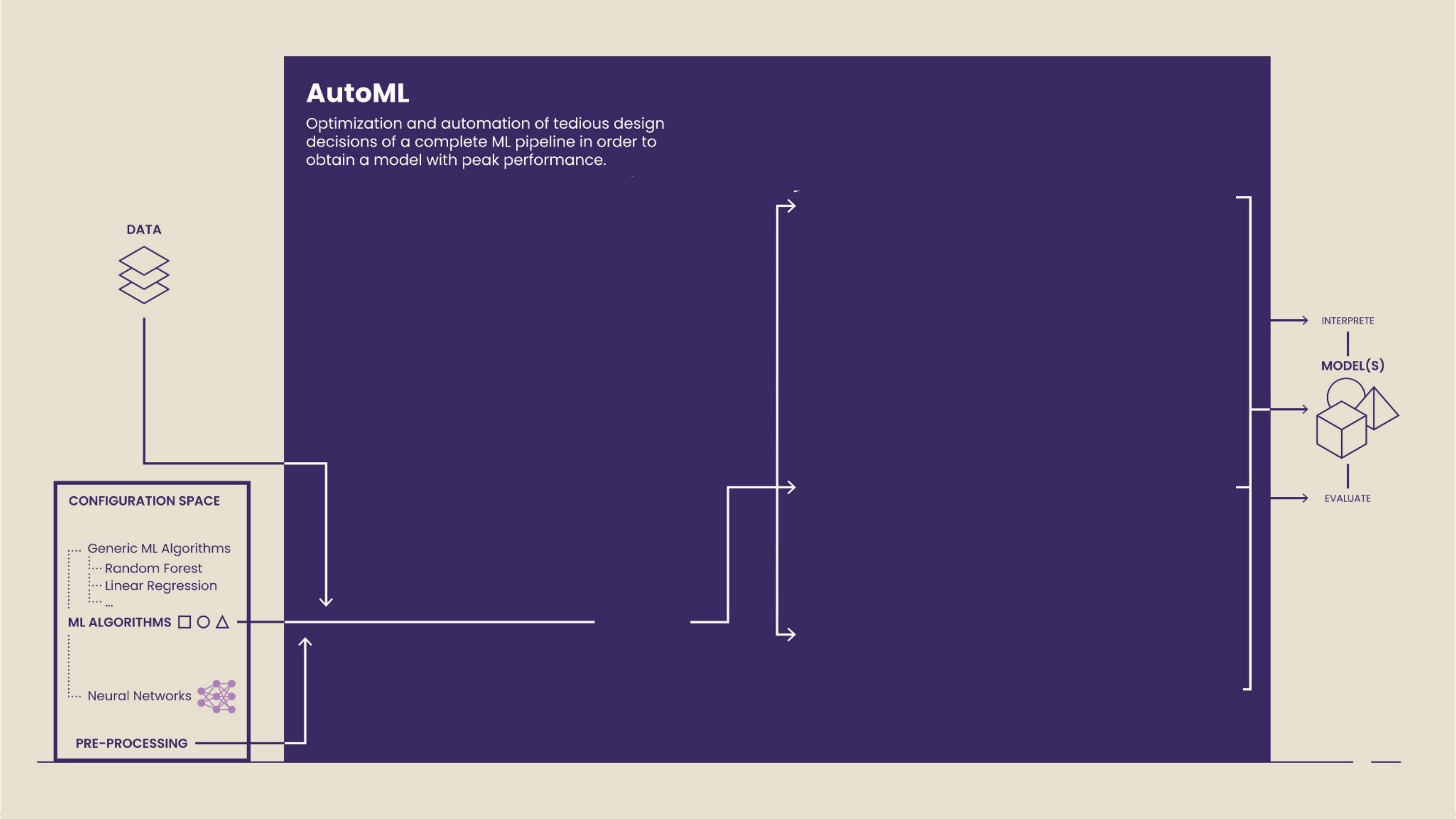
Have you ever had a HPO / CASH / NAS Problem?

→ Say hi to your neighbour! (~7min)

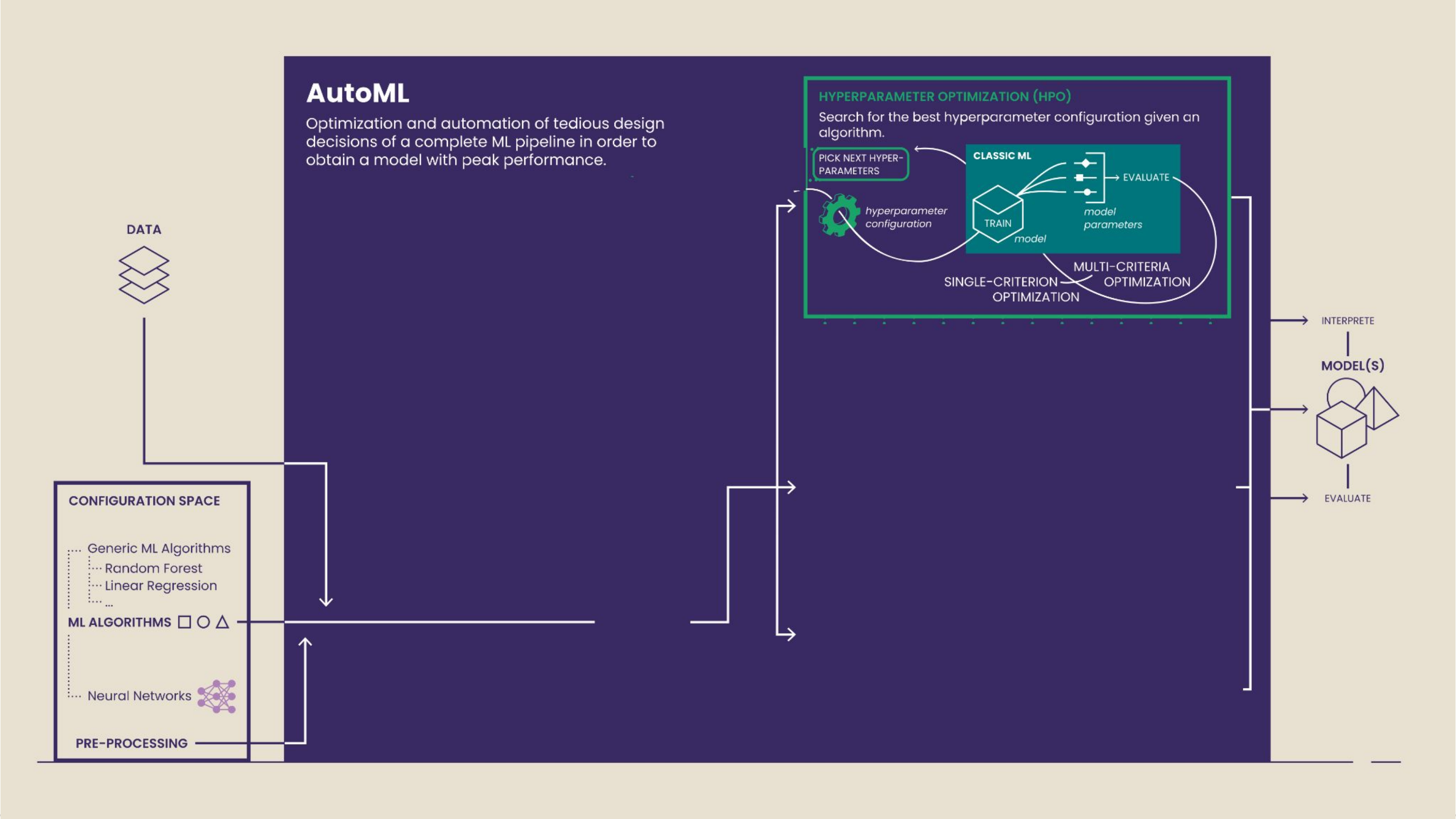
The Big Picture II

`>> Show me a nice picture!`

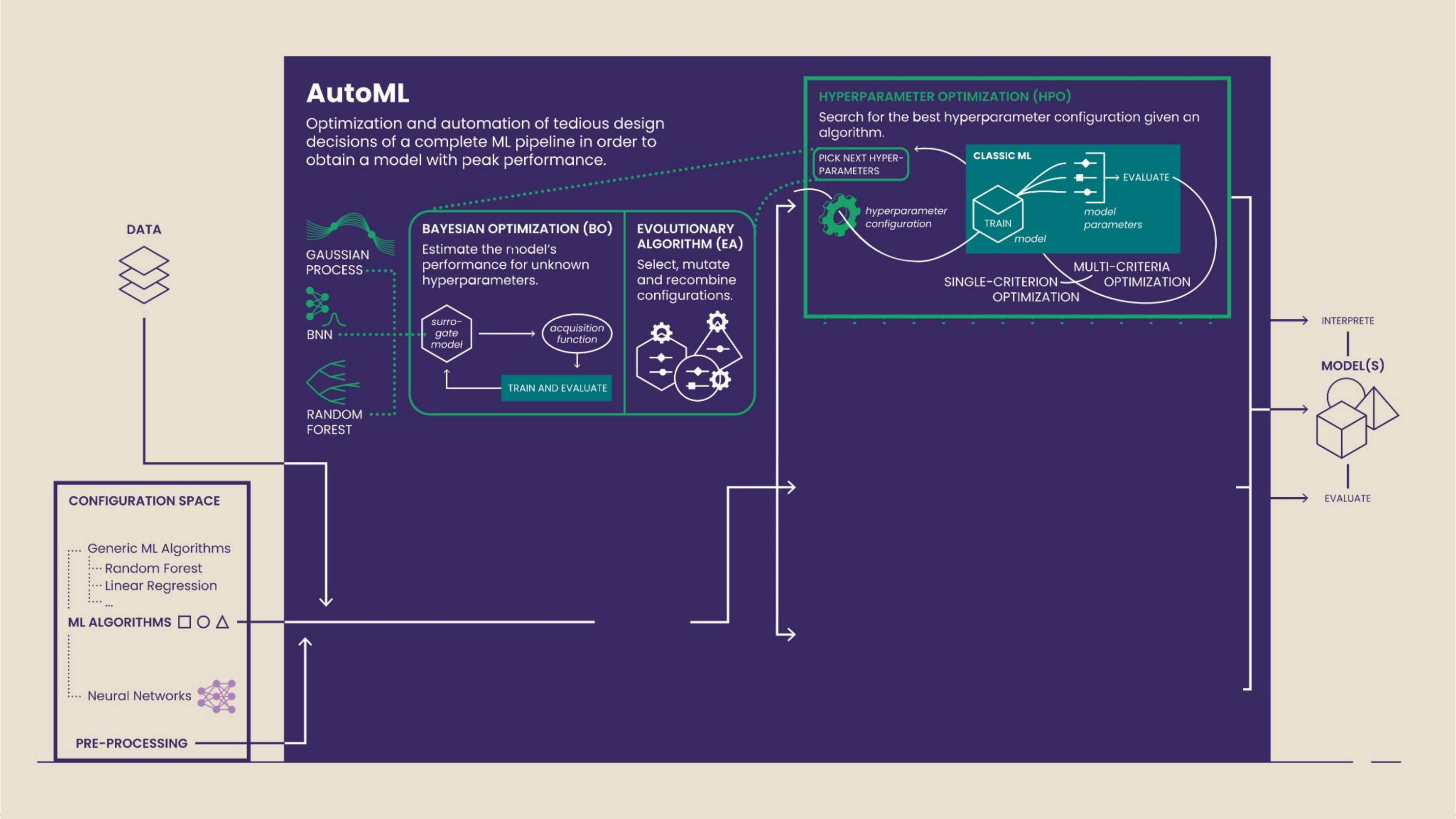
How everything is connected



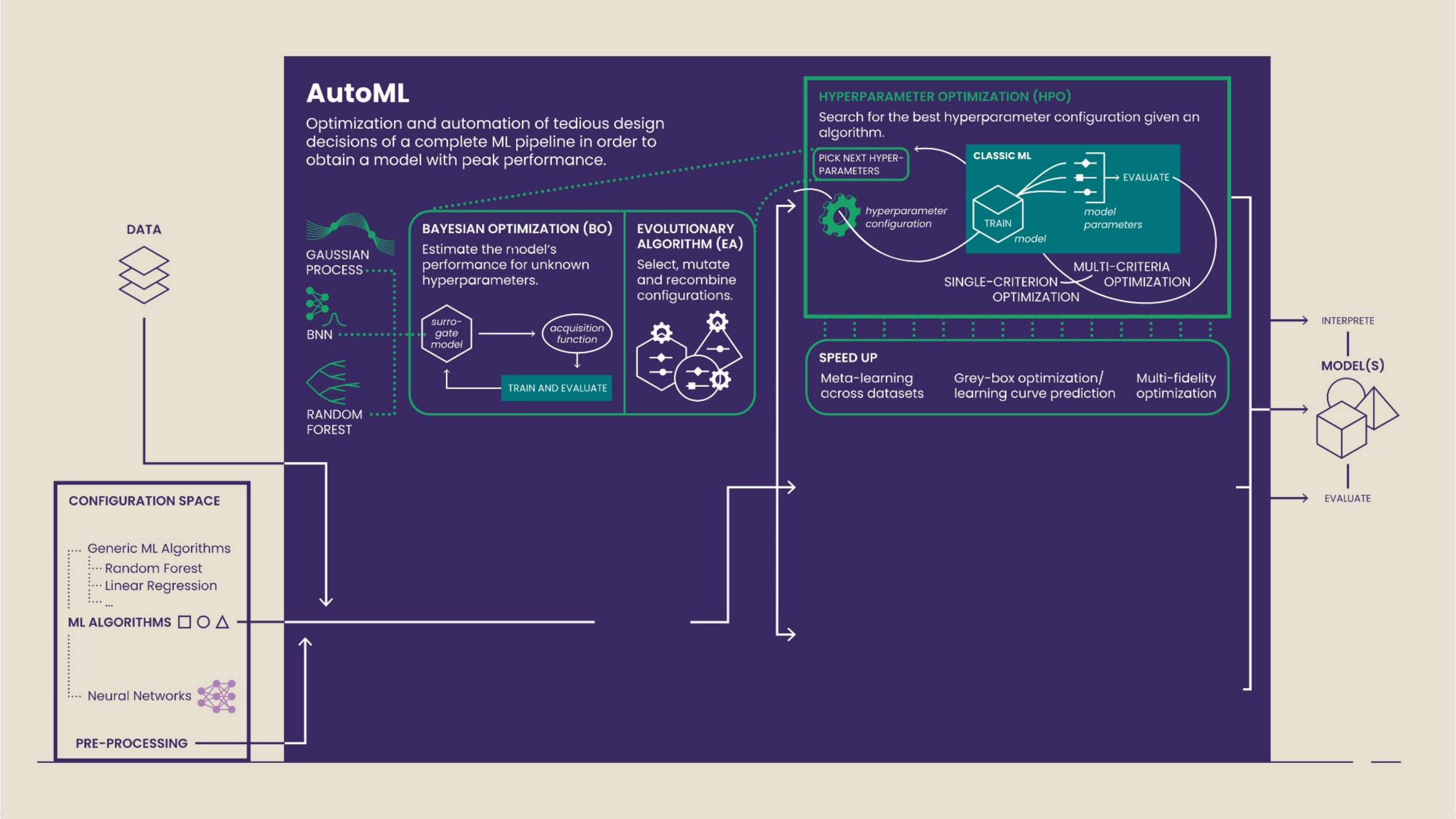
How everything is connected



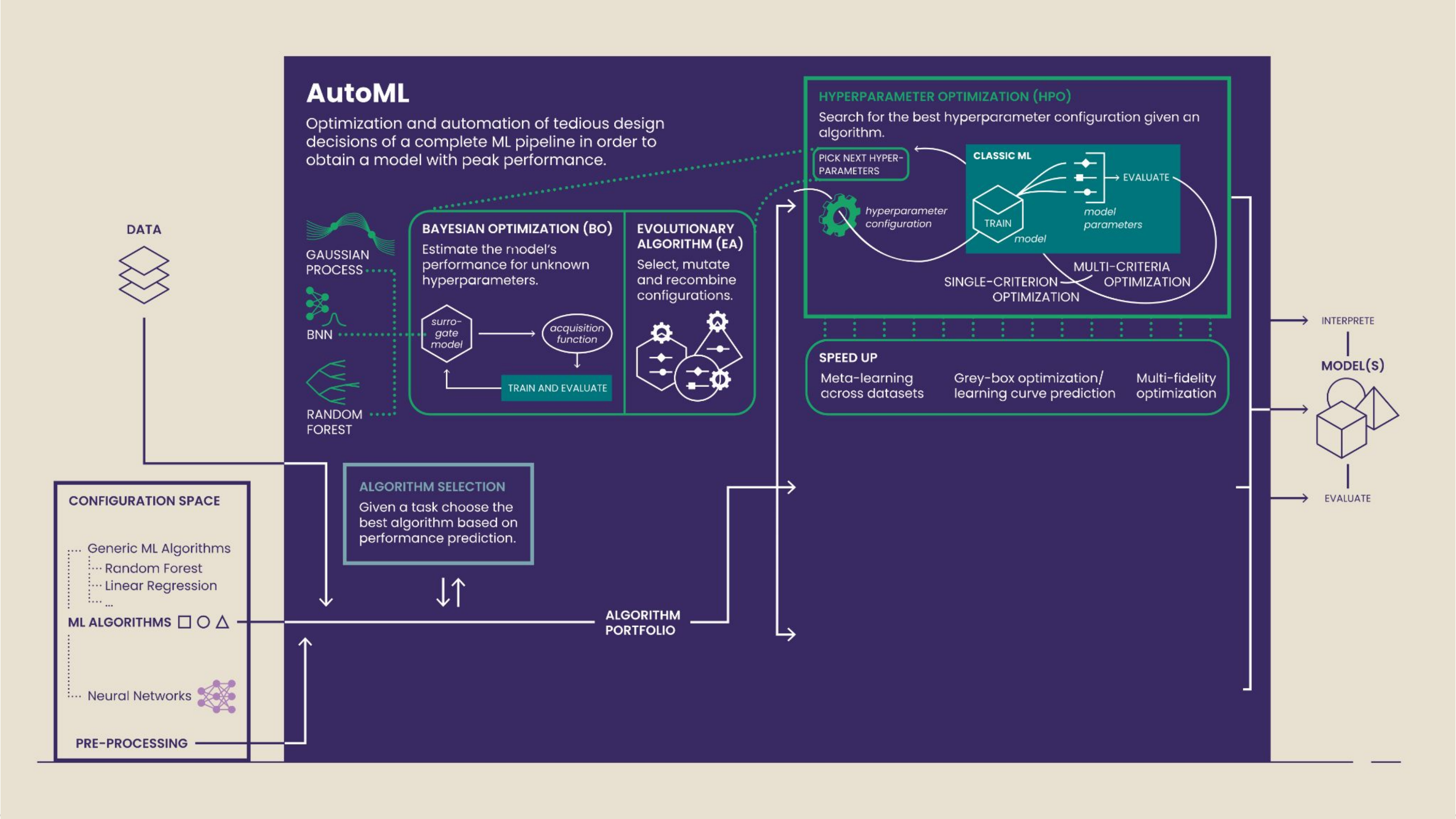
How everything is connected



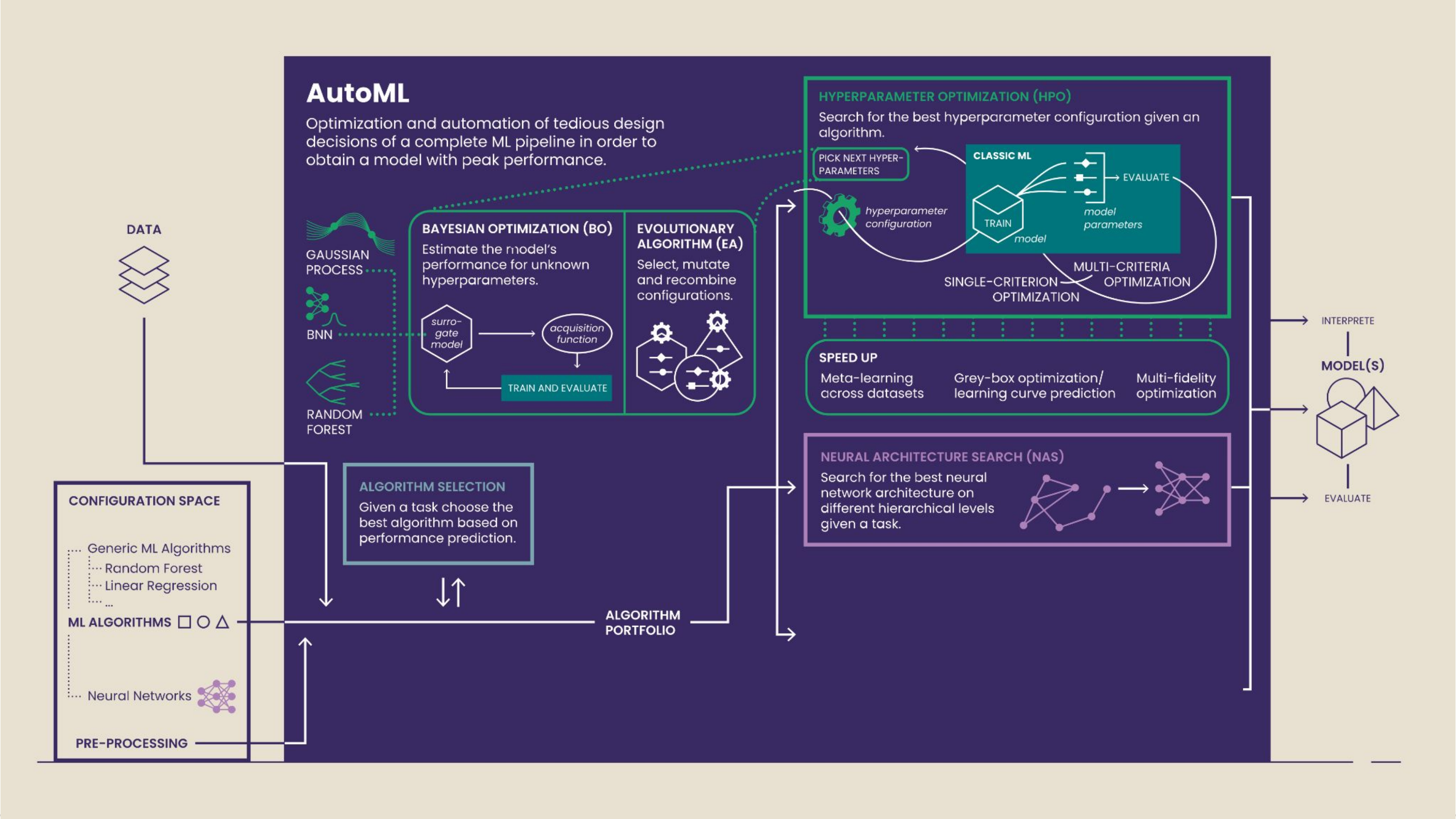
How everything is connected



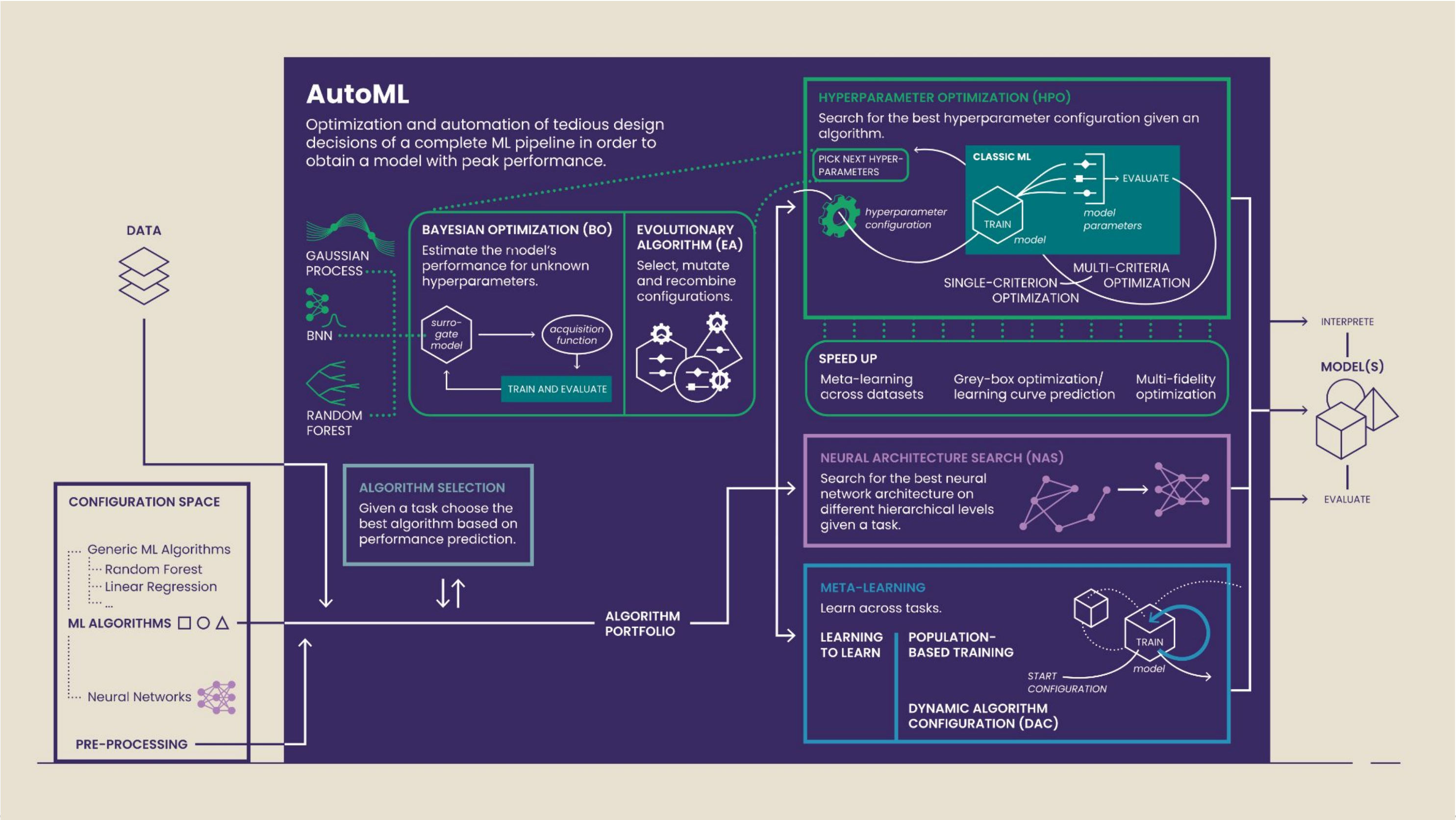
How everything is connected



How everything is connected



How everything is connected



Questions?

Break?

Organization

`>> When, where and how?`

Organisational Information

When? Wednesdays 12:00 (c.t.) - 14:00 (actually 12:15-13:45)

Where? MvL6, lecture hall (with some exceptions, see later)

Alternative: 16-18.
We'll discuss this later.

Expected Background Knowledge

- Machine Learning
- Deep Learning
- (optional) Practical experience with ML/DL

Grades

- Presentation & Slides: **100%** (20-25 min + 20 min Q&A)
- *+ Bonus for active participation in discussions*

All info will be on my website: https://keggensperger.github.io/teaching/2023-winter-seminar

🎁 Features of this Seminar

Here you will:

- Get a **broad overview** of current methods in AutoML
- Understand **challenges for research** in AutoML
- Identify **opportunities for AutoML** (and be able to pick a good approach)

Also relevant:

- **Active participation** in the discussion is key! Reading the papers beforehand is necessary.
  → You will gain a much better understanding

- **Feedback survey after each presentation**: We will take 5 minutes to collect anonymous feedback from everyone.
  →Learn what others like about your presentation and how to improve

- **Get feedback from a fellow student**: For each presenter, I will also assign a *study buddy*. Do a trial presentations, discuss open questions, prepare Q&A, etc. To get feedback from me, you (ideally) have incorporated their feedback
  → Higher quality of presentations; practice giving feedback

The Ideal Timeline (for the Presenter)

> -14d — Read Paper

Read and understand your topic (also take into account further info, e.g., videos, blogposts, posters, slides)

-7d — Prepare Presentation

Get feedback from your study buddy

-2d — Get Feedback

(optional) Get Feedback from me, e.g., Mondays 16-18

0 — Presentation

Enjoy!

+0.1h — Get Feedback

Get anonymous feedback from your peers and me

+2d — Send Slides

Presentation part is done.

Questions?

Dates Overview

18.10.2023	Today
25.10.2023, **3. OG (!)**	Intro Session
01.11.2023	No Meeting (holiday)
HPO I	
08.11.2023	#1 Spearmint / Input Warping ←Classics
15.11.2023; **16-18, 4. OG (!)**	#2 HEBO / TurBO ←State of the Art
22.11.2023	#3 BOHB / PriorBand ← Speedup techniques
AutoML for Tabular Data	
29.11.2023	#4 AutoGluon / ASKL ←AutoML Systems
06.12.2023	#5 FT-Transformer / Trees vs Nets ← Why do we need AutoML
13.12.2023	#6 TabPFN / XTAB ← Modern DL for tabular data
HPO II	
20.12.2023	#7 OptFormer / PFNs4BO ← Meta-Learn HPO
27.12.2023	No Meeting
03.01.2024	No Meeting
NAS	
10.01.2024	NAS Intro
17.01.2024	#8 DARTS / Understanding One-Shot NAS ← One-Shot
24.01.2024	#9 Once for all / HAT ←Hardware-aware NAS
31.01.2024	No Meeting
07.02.2024	#10 Zero-Cost Proxies I / Zero-Cost Proxies II ←Speedup techniques

Session schedule:

x 2

1. 20min Presentation
2. 20min Q&A
3. 5min Feedback

HPO I

[Spearmint] **Practical Bayesian Optimization of Machine Learning Algorithms**
Jasper Snoek, Hugo Larochelle, Ryan P. Adams; NeurIPS 2012
#Intro #BayesianOptimization #GaussianProcesses #Parallelization

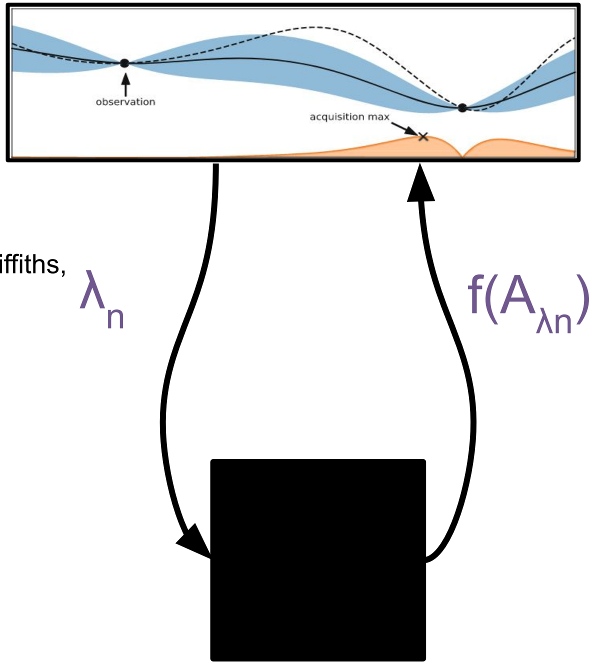
[InputWarping] **Input Warping for Bayesian Optimization of Non-Stationary Functions**
Jasper Snoek, Kevin Swersky, Rich Zemel, Ryan Adams; ICML 2014
#Intro #BayesianOptimization #GaussianProcesses #MultiTask

[HEBO] **HEBO: Pushing The Limits of Sample-Efficient Hyper-parameter Optimisation**
Alexander I. Cowen-Rivers, Wenlong Lyu, Rasul Tutunov, Zhi Wang, Antoine Grosnit, Ryan Rhys Griffiths, Alexandre Max Maraval, Hao Jianye, Jun Wang, Jan Peters, Haitham Bou-Ammar; JAIR 2022
#BayesianOptimization #GaussianProcesses #SOTA

[TurBO] **Scalable Global Optimization via Local Bayesian Optimization**
David Eriksson, Michael Pearce, Jacob Gardner, Ryan D. Turner, Matthias Poloczek; NeurIPS 2019
#BayesianOptimization #highdimensional #TrustRegionOptimization #SOTA

[BOHB] **BOHB: Robust and Efficient Hyperparameter Optimization at Scale**
Stefan Falkner, Aaron Klein, Frank Hutter; ICML 2018
#HPO #Successive Halving #Multi-fidelity #Parallelization #Speedup

[PriorBand] **PriorBand: Practical Hyperparameter Optimization in the Age of Deep Learning**
Neeratyoy Mallik, Edward Bergman, Carl Hvarfner, Danny Stoll, Maciej Janowski, Marius Lindauer, Luigi Nardi, Frank Hutter; NeurIPS 2023
#HPO #Multi-Fidelity #Parallelization #Speedup



AutoML for Tabular Data

[Auto-Sklearn] **Efficient and Robust Automated Machine Learning**
Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, Frank Hutter; NeurIPS 2015
#CASH #AutoMLSystem #BayesianOptimization

[AutoGluon] **AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data**
Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, Alexander Smola; arXiv 2020
#Ensembling #AutoMLSystem #SOTA

[FT-Transformer] **Revisiting Deep Learning Models for Tabular Data**
Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, Artem Babenko; NeurIPS 2021
#TabularDL #Transformer #BenchmarkPaper #Intro

[TreeVsNets] **Why do tree-based models still outperform deep learning on tabular data?**
Léo Grinsztajn, Edouard Oyallon, Gaël Varoquaux: NeurIPS 2022
#TabularDL #Meta-Study

[XTab] **XTab: Cross-table Pretraining for Tabular Transformers**
Bingzhao Zhu, Xingjian Shi, Nick Erickson, Mu Li, George Karypis, Mahsa Shoaran; ICML 2023
#TabularDL #Transformer

[TabPFN] **TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second**
Noah Hollmann, Samuel Müller, Katharina Eggensperger, Frank Hutter; ICLR 2023
#TabularDL #Transformer

HPO II

[after we've discussed transformer approaches]

[OptFormer] **Towards Learning Universal Hyperparameter Optimizers with Transformers**
Yutian Chen, Xingyou Song, Chansoo Lee, Zi Wang, Richard Zhang, David Dohan, Kazuya Kawakami, Greg Kochanski, Arnaud Doucet, Marc'Aurelio Ranzato, Sagi Perel, Nando de Freitas, NeurIPS 2022
#BayesianOptimization #LLM

[PFNs4BO] **PFNs4BO: In-Context Learning for Bayesian Optimization**
Samuel Müller, Matthias Feurer, Noah Hollmann, Frank Hutter, ICML 2023
#BayesianOptimization

Neural Architecture Search

[DARTS] **DARTS: Differentiable Architecture Search**
Hanxiao Liu, Karen Simonyan, Yiming Yang; ICLR 2019
#OneShot #Supernetwork

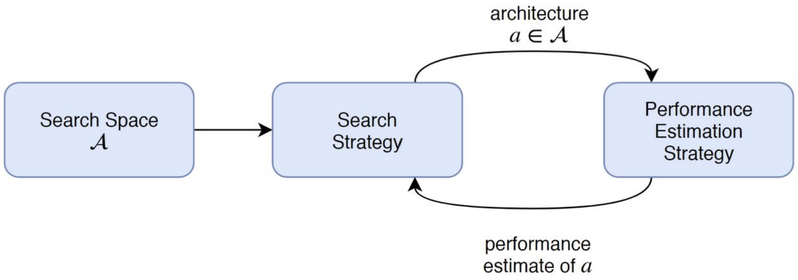
[Understanding One-Shot NAS] **Understanding and Simplifying One-Shot Architecture Search**
Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, Quoc Le; ICML 2018
#OneShot #Supernetwork #Analysis

[Once for all] **Once-for-All: Train One Network and Specialize it for Efficient Deployment**
Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, Song Han; ICLR 2020
#OneShot #Hardware-awareNAS

[HAT] **HAT: Hardware-Aware Transformers for Efficient Natural Language Processing**
Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, Song Han; ACL 2020
#OneShot #Hardware-awareNAS #NLP

[Zero-CostProxies I] **Neural Architecture Search without Training**
Joe Mellor, Jack Turner, Amos Storkey, Elliot J Crowley; ICML 2021
#Speedup #PerformancePrediction

[Zero-Cost Proxies II] **Zero-Cost Proxies for Lightweight NAS**
Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, Nicholas Donald Lane; ICLR 2021
#Speedup #PerformancePrediction



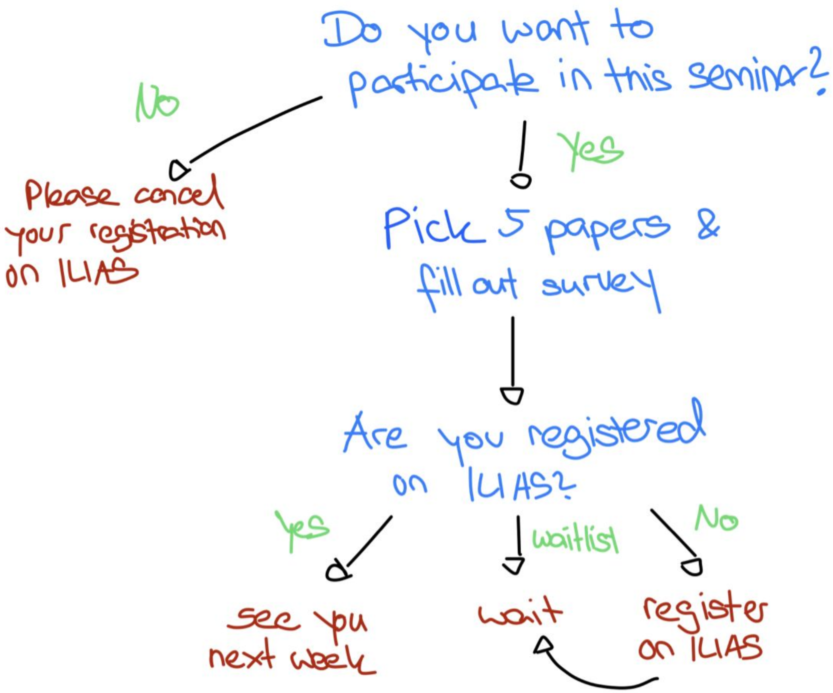
source: [Elsken at al. 2019]

Questions?

What’s next?

```
>> What shall I do?
```

Your Next Task?



Link to survey and slides will be on my website

Deadline: Next Monday, 24.10.2023, 1pm

→ You will hear back from me <u>before the next session</u>

NOTE: The next session will be in the seminar room 3.OG(!)