Deep Learning 2.0:
Towards AI that Builds and Improves AI

Frank Hutter

University of Freiburg
fh@cs.uni-freiburg.de

@FrankRHutter
@AutoML_org

European Research Council

Towards AI that Builds and Improves AI

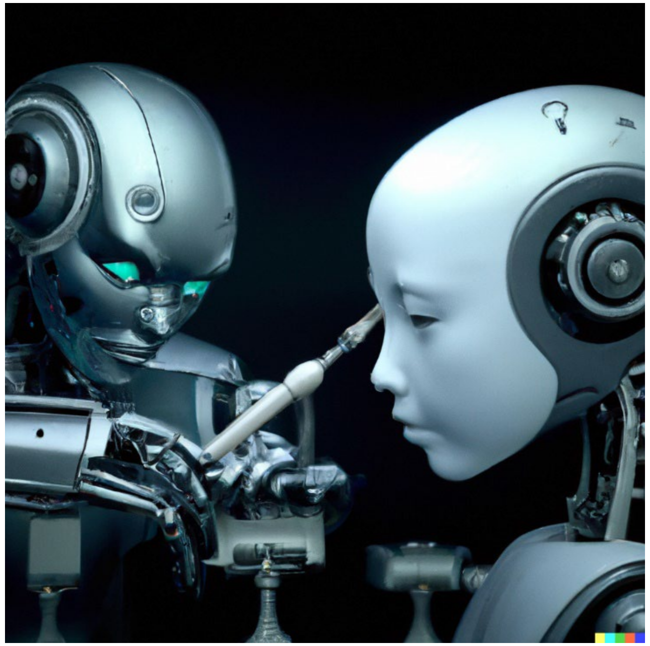


Image credit: DALLE-2

… to be more trustworthy

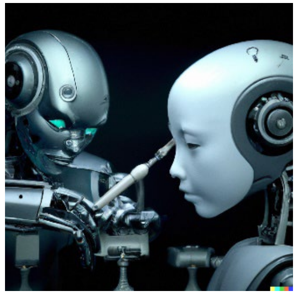
- performant
- fair
- calibrated
- energy-efficient
- robust
- …
- aligned with human values

Outline

Deep Learning 2.0:
Towards AI that Builds and Improves AI

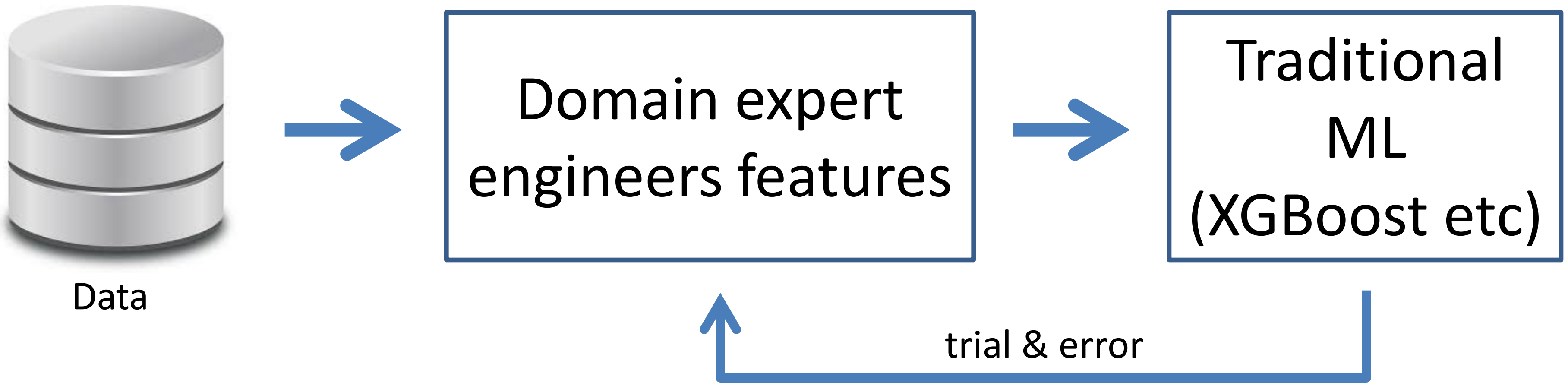
- ➡ Overview of Deep Learning 2.0
- Deep Dive: Meta-Learning a New ML Algorithm
- Outlook

European Research Council

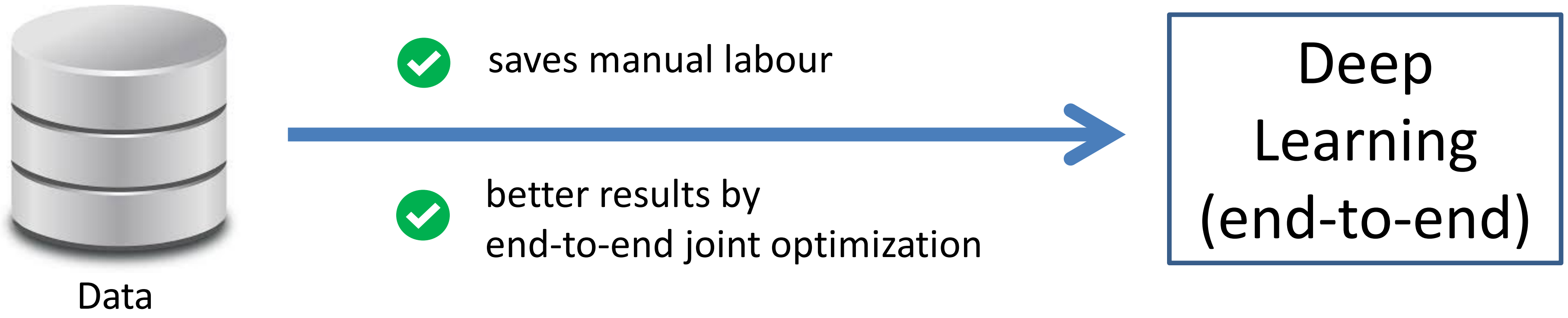


Why Deep Learning succeeded

Traditional ML practice before Deep Learning



Deep Learning



Traditional ML practice before Deep Learning

Data → Domain expert engineers features → Traditional ML (XGBoost etc)

trial & error

Deep Learning

Data → DL expert chooses architecture & hyperparameters → Deep Learning (end-to-end)

trial & error

From Deep Learning 1.0 to Deep Learning 2.0

Deep Learning 2.0

Data

saves human labour

better results by
end-to-end joint optimization

Deep
Learning 2.0
(end-to-end)

Deep Learning 1.0

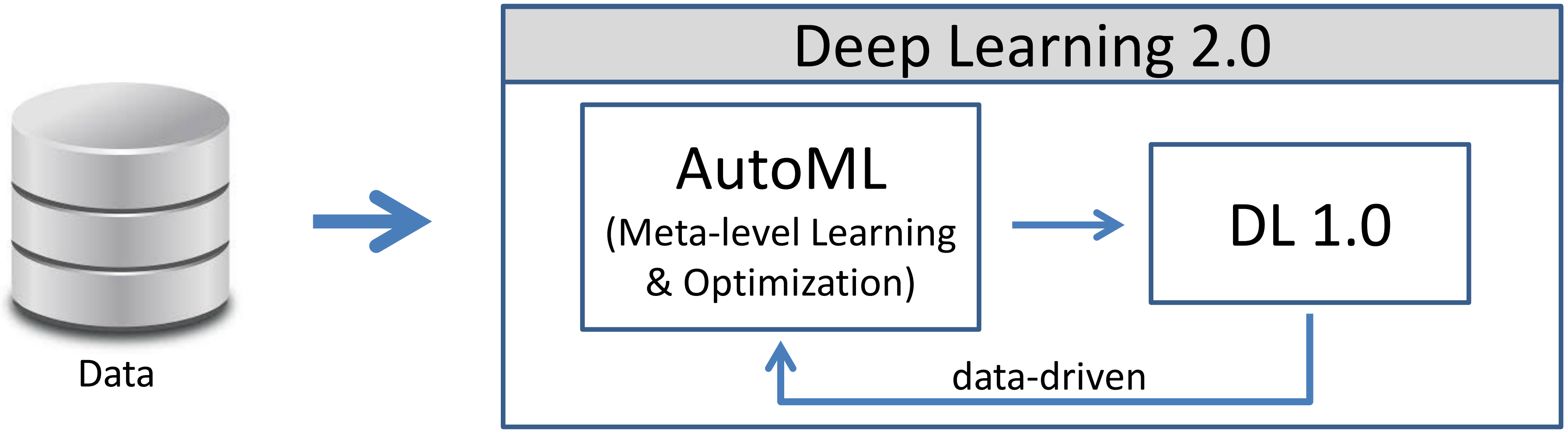
Data

DL expert chooses
architecture &
hyperparameters

Deep
Learning
(end-to-end)

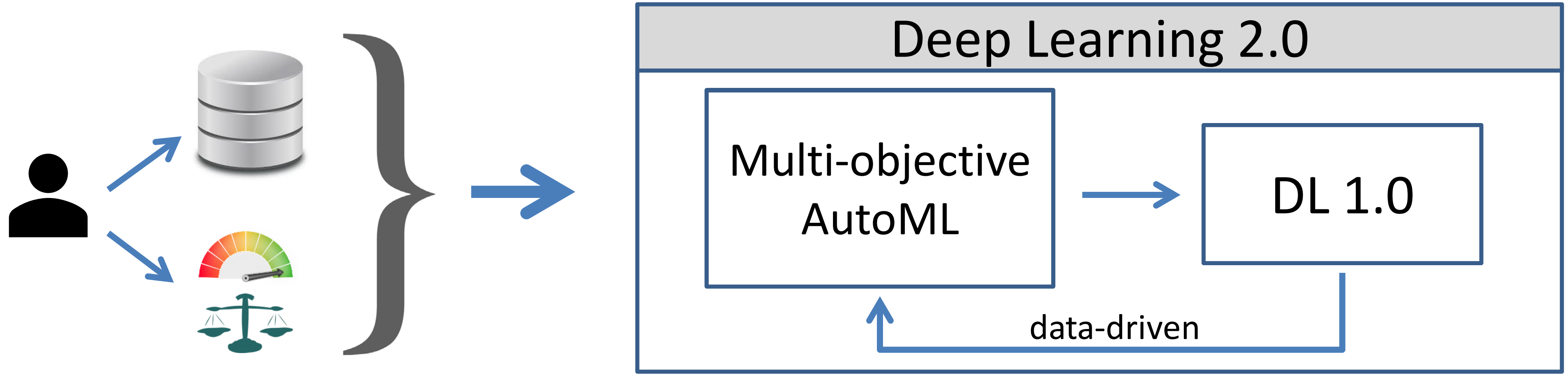
trial & error

From Deep Learning 1.0 to Deep Learning 2.0



- ✗ fairness
- ✗ robustness
- ✗ model calibration
- ✗ interpretability
- ✗ latency of predictions
- ✗ size(memory) of the model

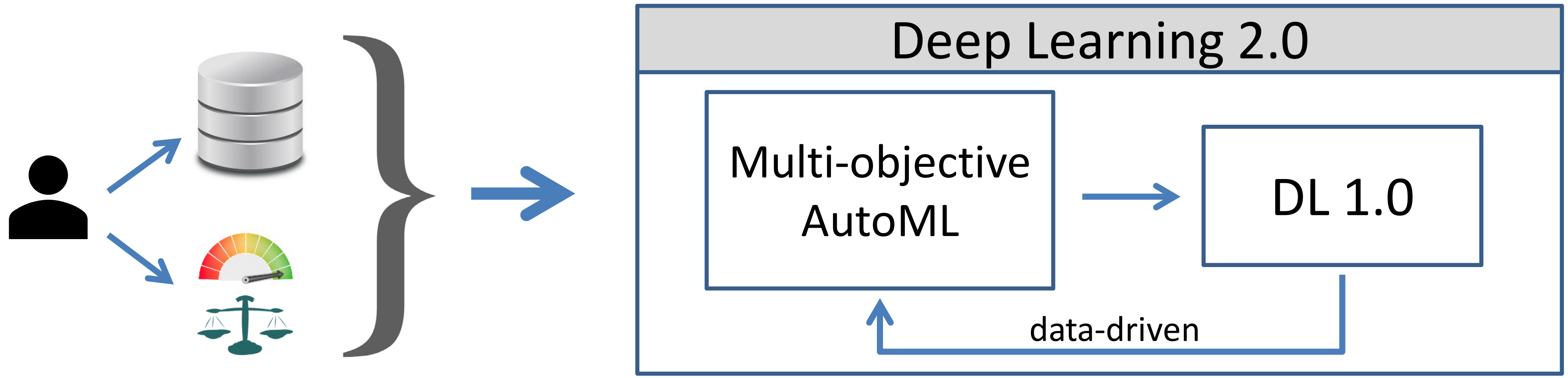
Deep Learning 2.0: Trustworthy AI by Design



- domain expert can specify objectives

- fairness
- robustness
- model calibration
- interpretability
- latency of predictions
- size(memory) of the model

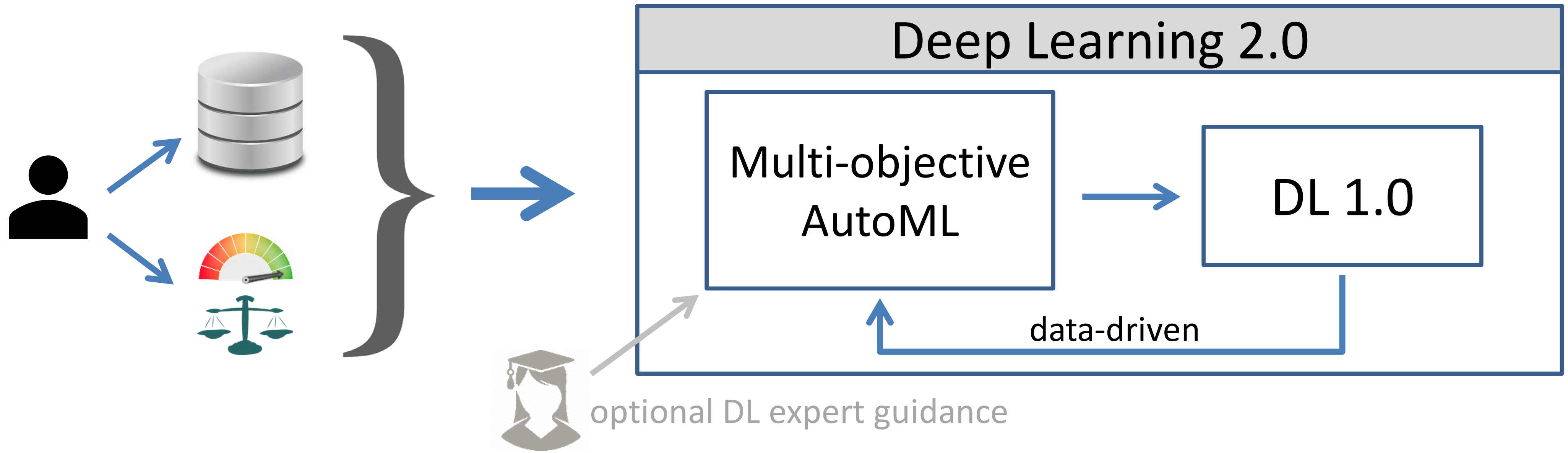
Expected Impact of Deep Learning 2.0



- **Paradigm-changing: democratizing Deep Learning**
  - DL 2.0 projects possible without a DL expert
  - DL 2.0 directly optimizes for user's objectives
    → Trustworthy AI by design

DL 2.0 will be even more pervasive than DL 1.0, with huge impact on the billion-dollar DL market

Deep Learning 2.0: Human Centric



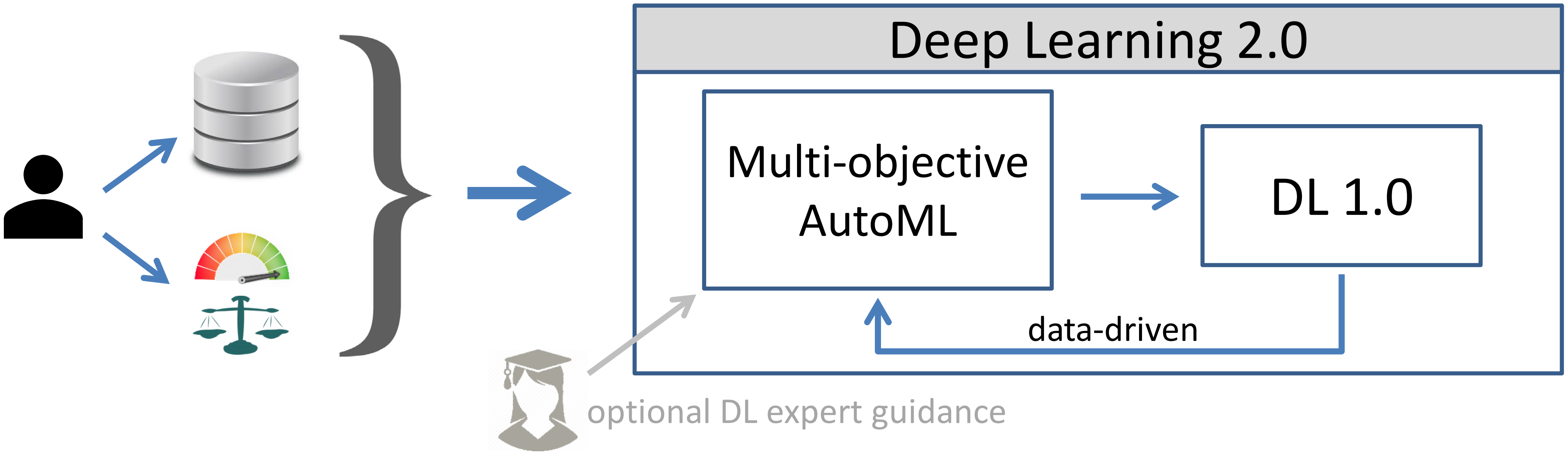
- **Paradigm-changing: democratizing Deep Learning**
  - DL 2.0 projects possible without a DL expert
  - DL 2.0 directly optimizes for user‘s objectives

    → Trustworthy AI by design

DL 2.0 will be even more pervasive than DL 1.0, with huge impact on the billion-dollar DL market

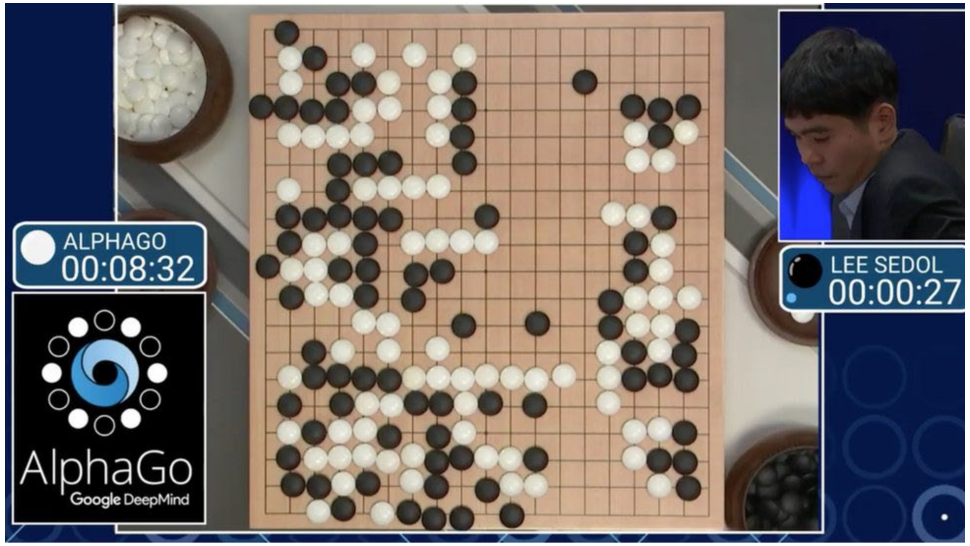
Important Parts of DL 2.0 / AutoML

- Hyperparameter optimization (HPO)
- Neural architecture search (NAS)
- Multi-objective AutoML
- AutoML systems
- Meta-learning entire algorithms



Important Parts of DL 2.0 / AutoML

- **Hyperparameter optimization (HPO)**



AlphaGo: tuning 10 hyperparameters improved win rate from 50% to 65% before playing Lee Sedol

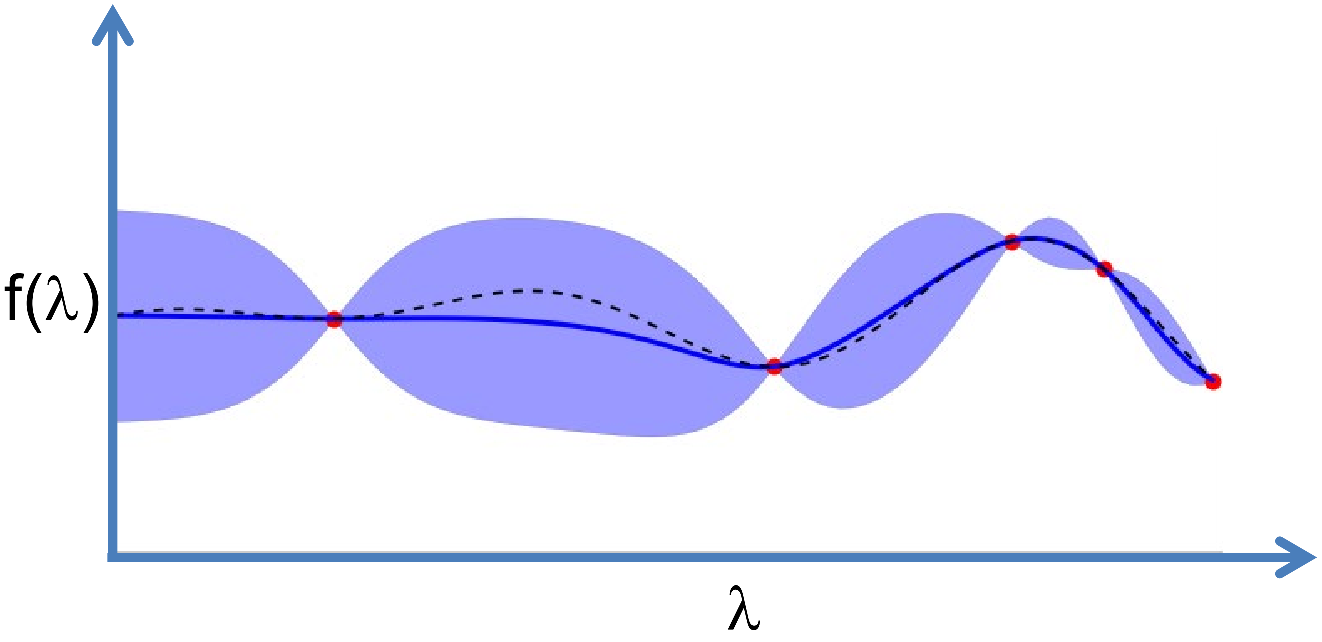
Hyperparameter Group	Hyperparameters
Finetuning Strategies	Percentage of the Model to Freeze, Layer Decay, Linear Probing, Stochastic Norm, SP-Regularization, DELTA Regularization, BSS Regularization, Co-Tuning
Regularization Techniques	MixUp, MixUp Probability*, CutMix, Drop-Out, Label Smoothing, Gradient Clipping
Data Augmentation	Data Augmentation Type (Trivial Augment, Random Augment, Auto-Augment), Auto-Augment Policy*, Number of operations*, Magnitude*
Optimization	Optimizer type (SGD, SGD+Momentum, Adam, AdamW, Adamp), Beta-s*, Momentum*, Learning Rate, Warm-up Learning Rate, Weight Decay, Batch Size
Learning Rate Scheduling	Scheduler Type (Cosine, Step, Multi-Step, Plateau), Patience*, Decay Rate*, Decay Epochs*

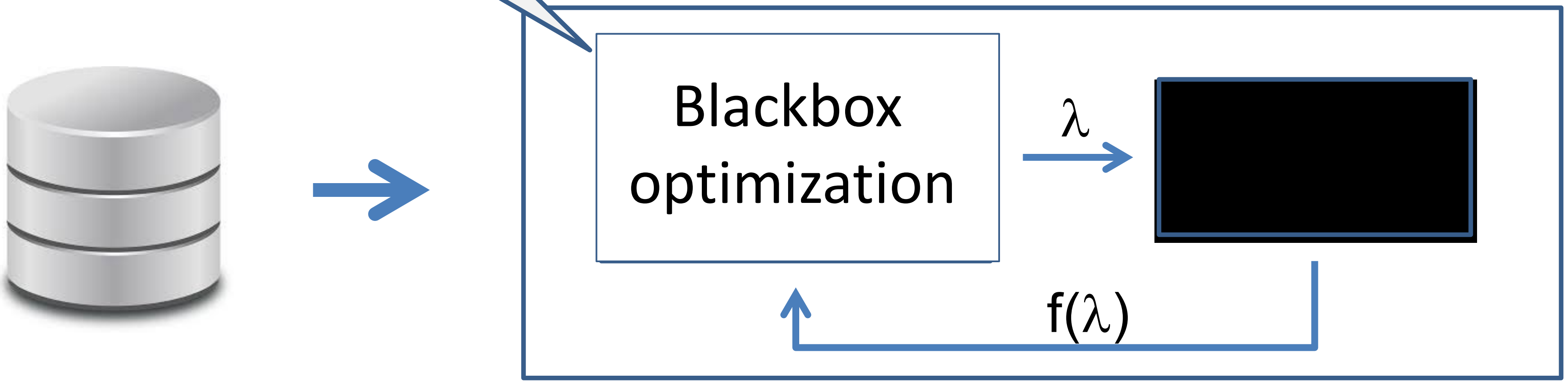
Hyperparameters for fine-tuning foundation models

Too many choices [Pineda et al, 2023]

HPO as Blackbox Optimization

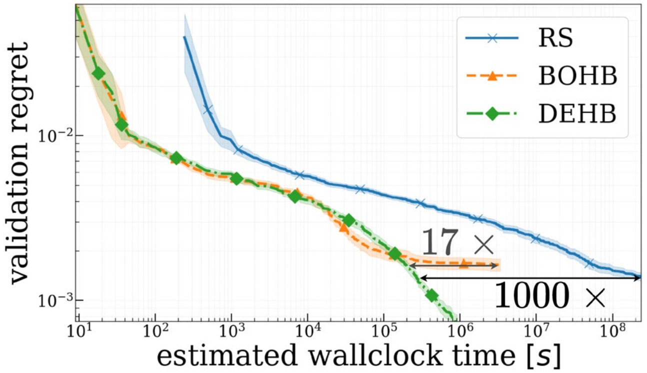
Random search,
evolutionary methods,
reinforcement learning,
…
Bayesian optimization



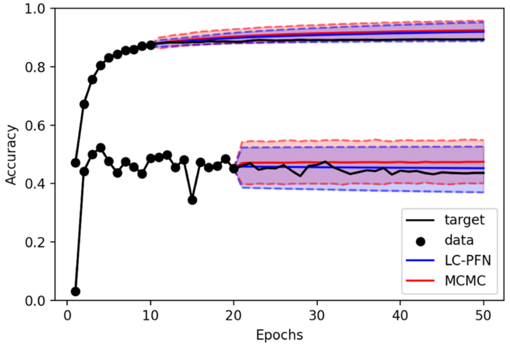


Speeding up HPO: Beyond Blackbox Optimization

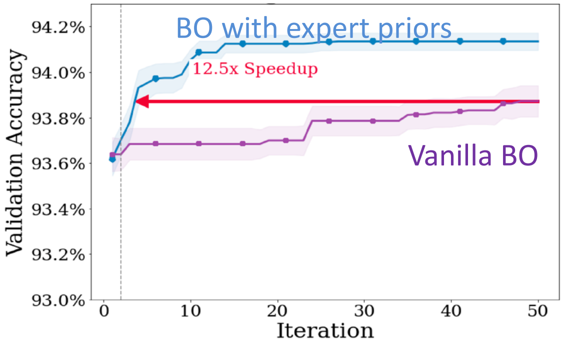
- **Multi-fidelity optimization**
  - Speedups by cheap proxies, reducing #epochs, resolution, #classes, #data points, width, depth

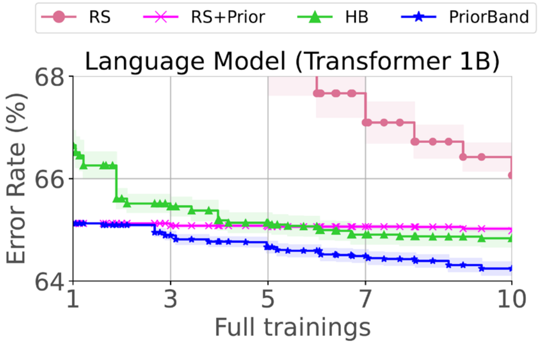


- **Meta-learning**
  - Learning to transfer across tasks
  - Learning to extrapolate learning curves [NeurIPS’23]



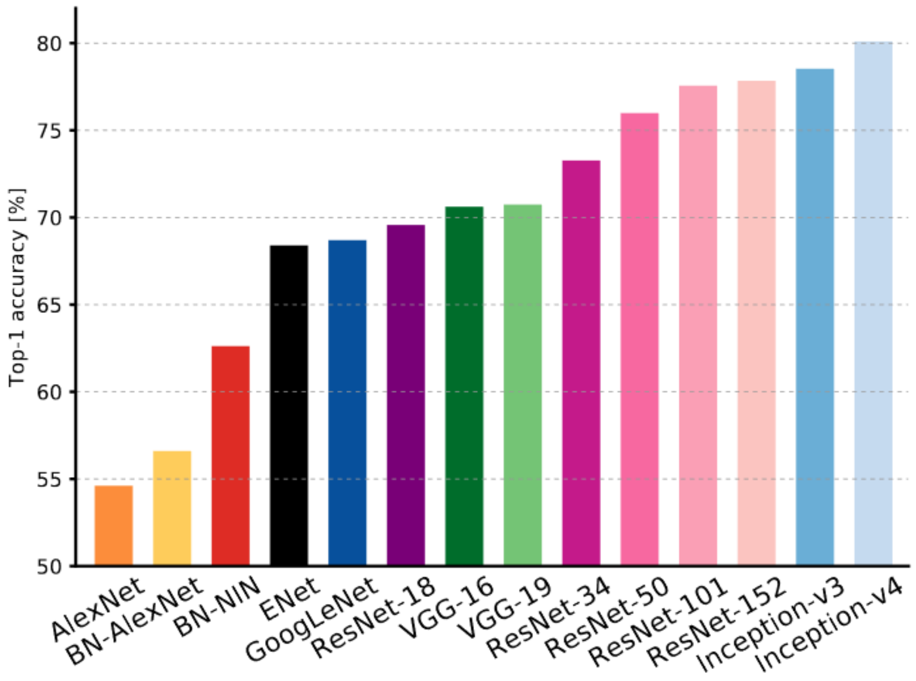
- **Integrating human expert priors**
  - Bridging the gap to manual search [ECML-PKDD’21, ICLR’22]
  - Combined with multi-fidelity optimization [NeurIPS’23]

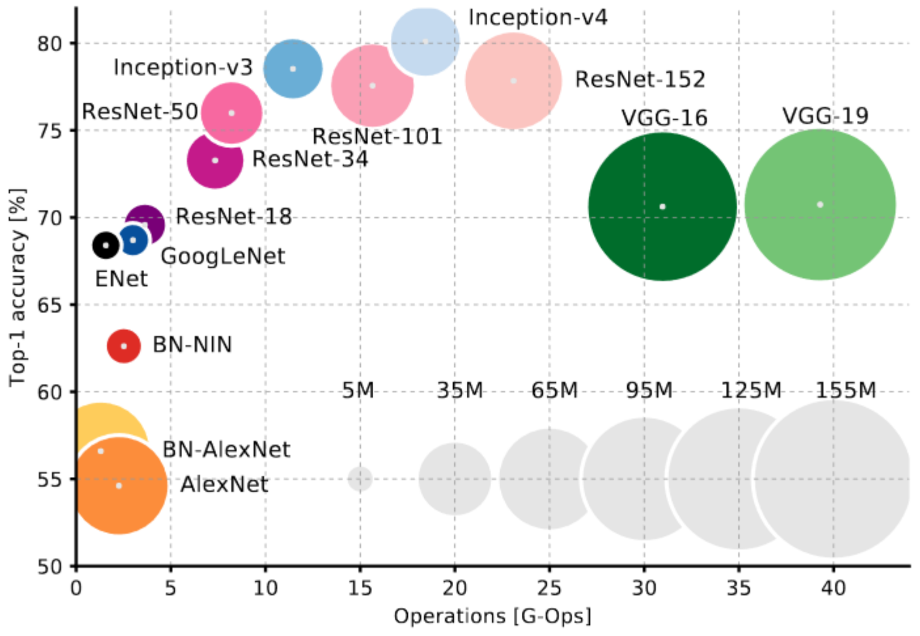




Important Parts of DL 2.0 / AutoML

- Hyperparameter optimization (HPO)
- **Neural architecture search (NAS)**





The choice of architecture matters – can we find automatically find better architectures?

[Image credit: Canziani et al., arXiv 2017]

Neural Architecture Search (NAS)

- Find neural architecture A such that deep learning works best for given data
  - Measured by validation error of architecture A with trained weights $w^*(A)$

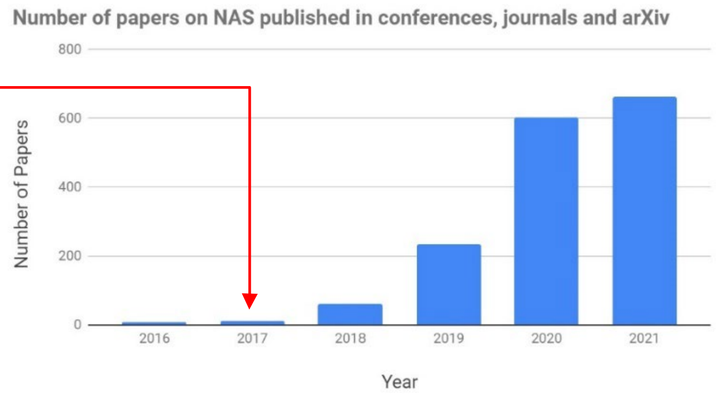
$$\min_{A \in \mathcal{A}} \mathcal{L}_{\text{val}}(w^*(A), A)$$

$$\text{s.t. } w^*(A) \in \text{argmin}_w \mathcal{L}_{\text{train}}(w, A)$$

Outer level: optimize the architecture

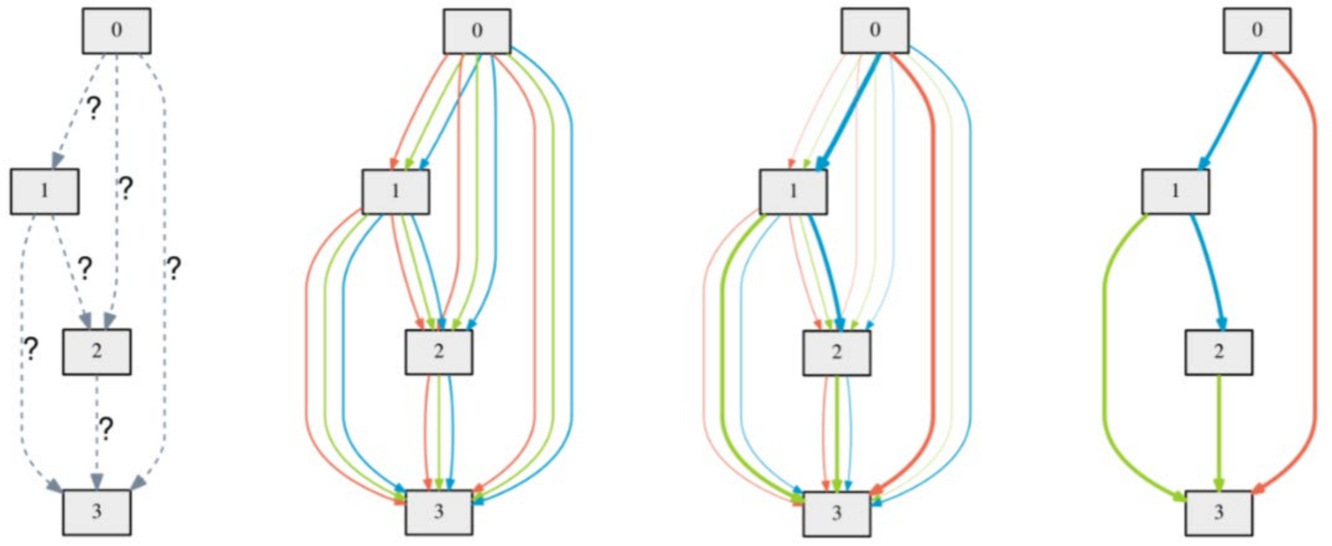
Inner level: optimize the weights

- Famously tackled by reinforcement learning [Zoph & Le, ICLR 2017]
  - 12.800 architectures trained fully
  - 800 GPUs for 2 weeks (about $60.000 USD)



Making NAS Efficient – DARTS: Differentiable Architecture Search

[Liu et al., ICLR 2019]



(a) (b) (c) (d)

- Relax the discrete NAS problem (a $\rightarrow$ b)
  - One-shot model with continuous architecture weight α for each operator
  - Mixed operator: $\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x)$
- Solve the bi-level optimization problem (c)

$$\min_\alpha \mathcal{L}_{\text{val}}(w^*(\alpha), \alpha)$$
$$\text{s.t. } w^*(\alpha) \in \text{argmin}_w \mathcal{L}_{\text{train}}(w, \alpha)$$

- In the end, discretize to obtain a single architecture (d)

[Visualization taken from Liu et al, 2019]

Main Research Trends in NAS

- Robustness of DARTS & successors

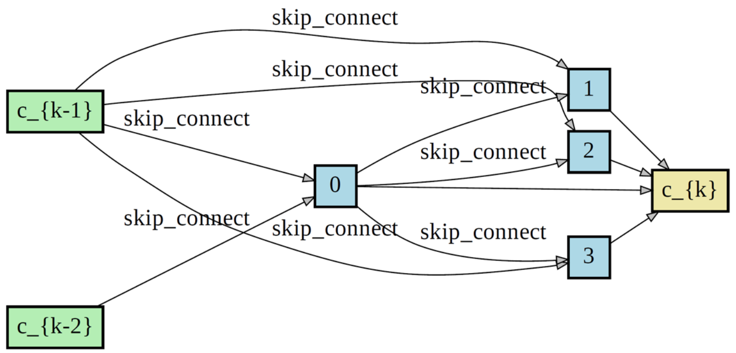
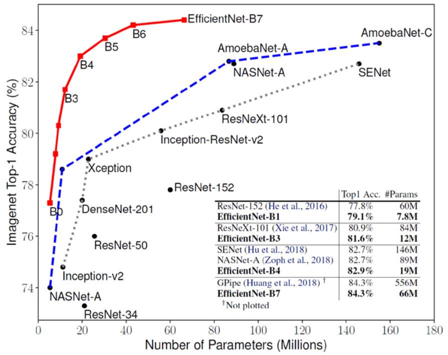
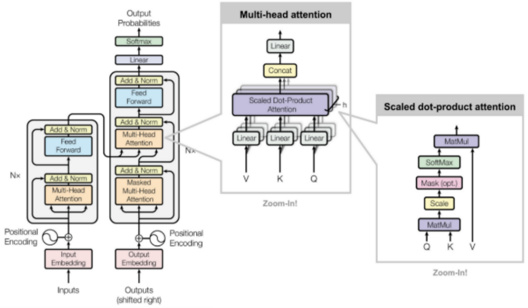


Figure: Failure mode of DARTS only picking skip connections

- Extensions of HPO techniques to NAS
  - Speedups: multi-fidelity, meta-learning, priors
- „Holy grail“: discovering entirely novel architectures
  - E.g., hierarchical search spaces based on grammars [NeurIPS’23]
- Heavily used in industry: **hardware-aware NAS**, trading off various objectives
  - Accuracy
  - Memory
  - Test-time latency
  - Energy usage

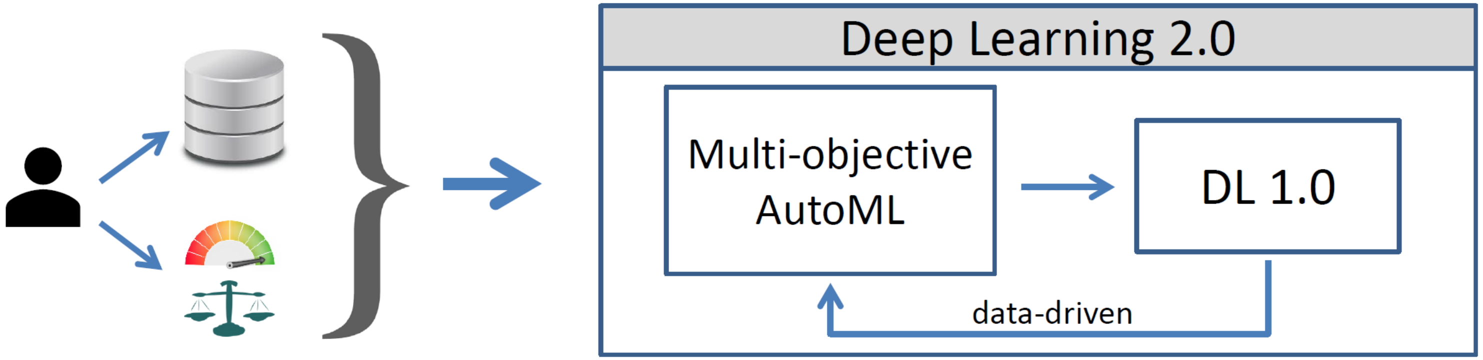




Optimizing latency of transformers

Important Parts of DL 2.0 / AutoML

- Hyperparameter optimization (HPO)
- Neural architecture search (NAS)
- **Multi-objective AutoML**

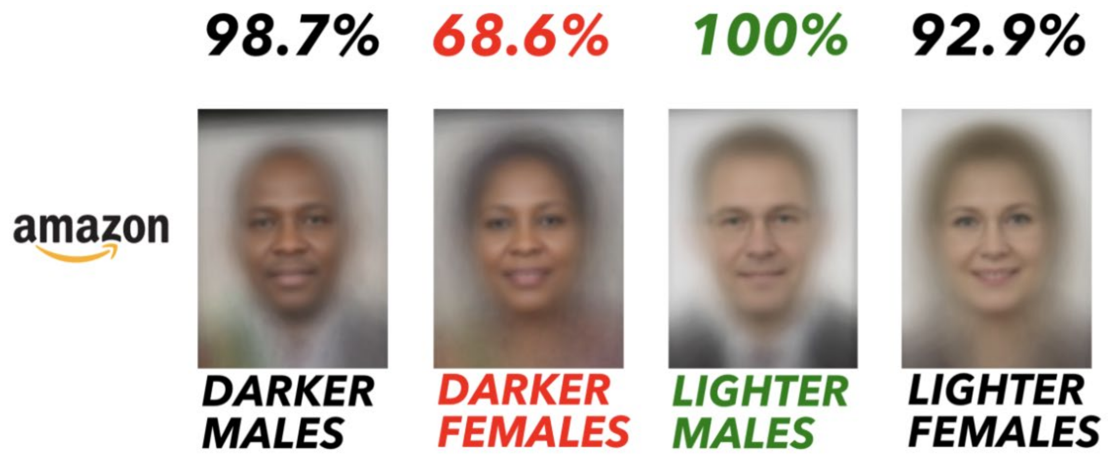


- ✅ domain expert can specify objectives

- ✅ fairness
- ✅ robustness
- ✅ model calibration
- ✅ interpretability
- ✅ latency of predictions
- ✅ size(memory) of the model

Can DL 2.0 Help with Fairness? A Case Study in Face Recognition

- Facial recognition (FR) systems are known to exhibit bias
  - sociodemographic dimensions, like gender and race



	Darker Male	Darker Female	Lighter Male	Lighter Female	Largest Gap
Amazon	98.7%	68.6%	100.0%	92.9%	31.4%
Kairos	98.7%	77.5%	100.0%	93.6%	22.5%
IBM	99.4%	83.0%	99.7%	97.6%	16.7%
Face++	98.7%	95.9%	99.5%	99.0%	3.6%
Microsoft	99.7%	98.5%	100.0%	99.7%	1.5%

- Face recognition is used by law enforcement agencies for sensitive applications
  - Identifying suspects; tracking down missing persons; biometric security

- How can we improve this?
  - Pre-processing, training, and post-processing methods have failed to close the gap
  - Can Deep Learning 2.0 help?

Deep Learning 2.0 to find better & fairer models [NeurIPS’23 oral]

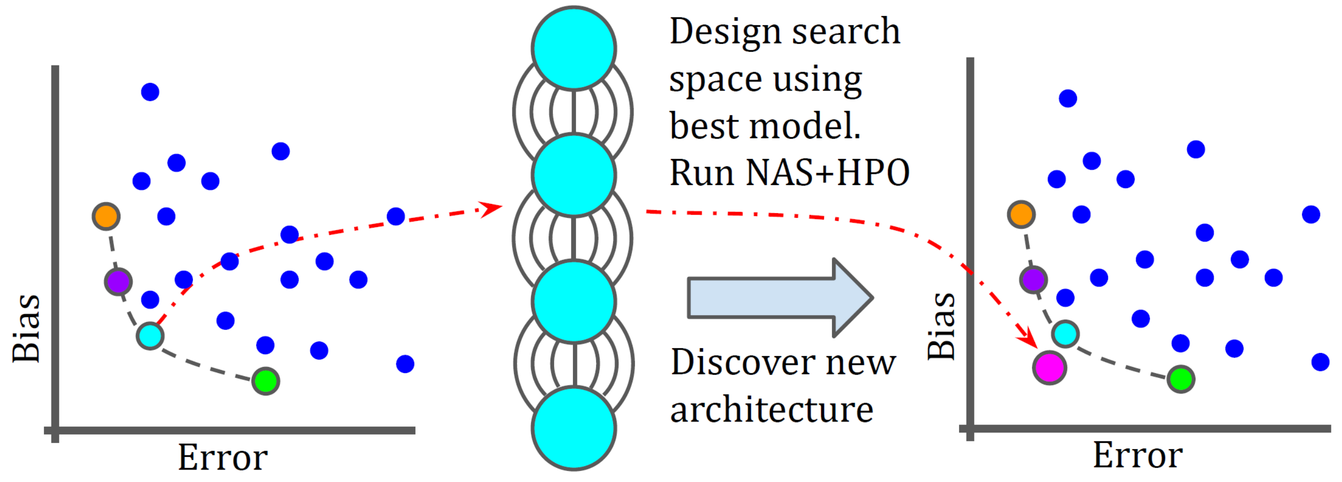
- Dataset: CelebA face recognition
- Protected attribute: Gender
- Fairness metric: difference in quality of classification („Rank disparity”)

Architectures

Examples:
Inception
ViTransformer
VovNet
MobileNetV3
Dual Path Net

Hyperparams

Examples:
MagFace (head)
ArcFace (head)
CosFace (head)
Optimizer
Learning Rate



355 models trained | **Pareto frontier** | **Novel Search Space** | **Pareto dominant**

- **Result: fairer and more accurate than traditional fairness mitigation algorithms**

Important Parts of DL 2.0 / AutoML

- Hyperparameter optimization (HPO)
- Neural architecture search (NAS)
- Multi-objective AutoML
- **AutoML systems**: optimize the entire data science pipeline

E.g., **Scikit-learn** [Pedregosa et al, 2011]

– 15 classifiers with a total of 59 hyperparameters

– 13 feature preprocessors

– 4 data preprocessors

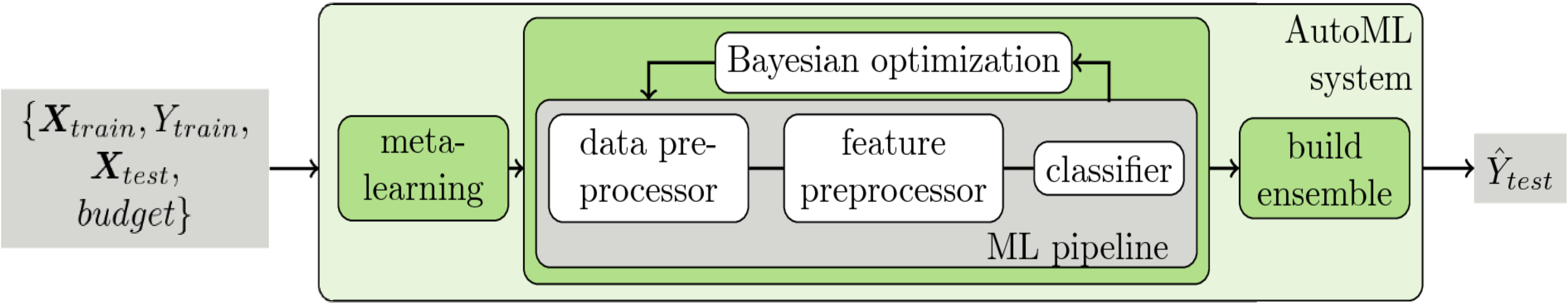
– In total: 110 hyperparameters

classifier	$\#\lambda$
AdaBoost (AB)	4
Bernoulli naïve Bayes	2
decision tree (DT)	4
extreml. rand. trees	5
Gaussian naïve Bayes	-
gradient boosting (GB)	6
kNN	3
LDA	4
linear SVM	4
kernel SVM	7
multinomial naïve Bayes	2
passive aggressive	3
QDA	2
random forest (RF)	5
Linear Class. (SGD)	10

preprocessor	$\#\lambda$
extreml. rand. trees prepr.	5
fast ICA	4
feature agglomeration	4
kernel PCA	5
rand. kitchen sinks	2
linear SVM prepr.	3
no preprocessing	-
nystroem sampler	5
PCA	2
polynomial	3
random trees embed.	4
select percentile	2
select rates	3
one-hot encoding	2
imputation	1
balancing	1
rescaling	1

AutoML Systems

- First full AutoML systems for tabular data:
  - **Auto-WEKA** [KDD 2013] KDD 2023 test of of time award
  - **Auto-sklearn** [NeurIPS 2015] and **Auto-sklearn 2.0** [JMLR 2022]



- Auto-sklearn won both the 1st and 2nd **AutoML challenges**
  - Much better than base-level systems & human experts
  - **20k monthly downloads**

- **Auto-Gluon**: fantastic engineering, and fully embracing ensembling [arXiv 2020]
- To be discussed later in the talk: **TabPFN** [ICLR 2023], **CAAFE** [NeurIPS 2023]

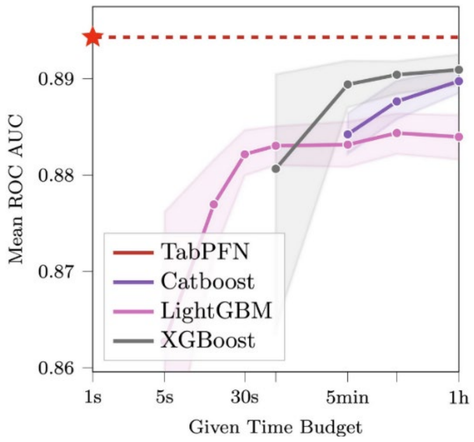
Important Parts of DL 2.0 / AutoML

- AutoML systems
- Hyperparameter optimization (HPO)
- Neural architecture search (NAS)
- Multi-objective AutoML
- **Meta-learning entire algorithms**

Learned matrix multiplication algorithm [Fawci et al, 2022]

```
def train(weight, gradient, momentum, lr):
  update = interp(gradient, momentum, β1)
  update = sign(update)
  momentum = interp(gradient, momentum, β2)
  weight_decay = weight * λ
  update = update + weight_decay
  update = update * lr
  return update, momentum
```

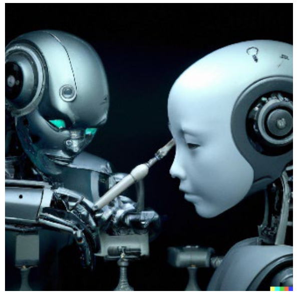
Learned Lion optimizer [Chen et al, 2023]



TabPFN: learned tabular classification algorithm

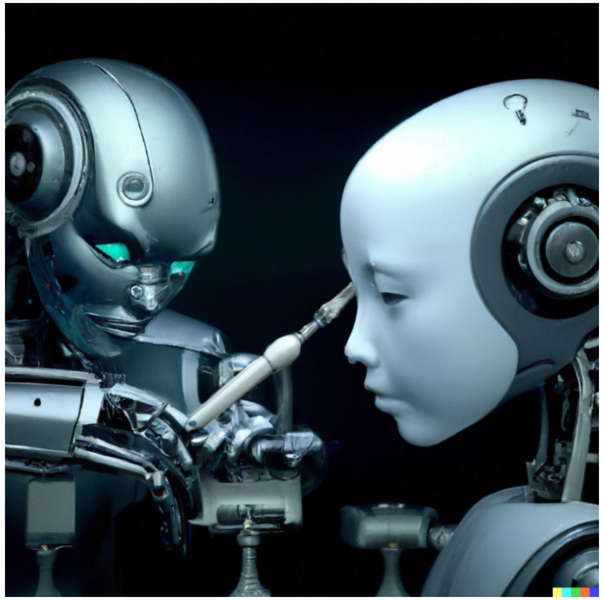
Deep Learning 2.0:
Towards AI that Builds and Improves AI

- Overview of Deep Learning 2.0
- ➡ Deep Dive: Meta-Learning a New ML Algorithm
- Outlook



Outline

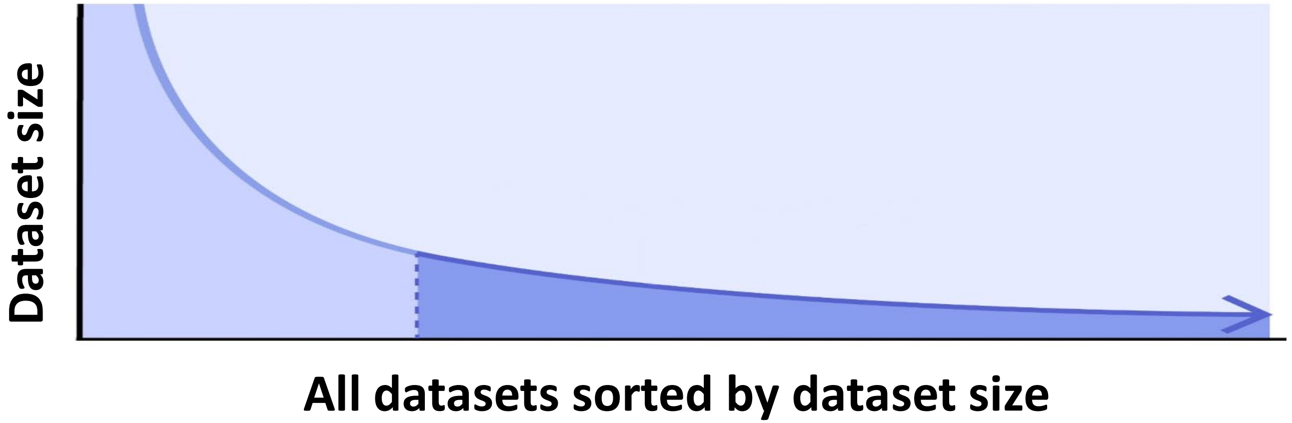
- Overview of Deep Learning 2.0
- Deep Dive: Meta-Learning a New ML Algorithm
  - ➡ Premises and Preview of Results
  - – PFNs: Transformers can approximate Bayesian Inference
  - – TabPFN: PFNs with a prior over tabular datasets
- Outlook



Premises

company	division	sector	tryint
00nil_Combined_Company	00nil_Combined_Division	00nil_Combined_Sector	14625
apple	00nil_Combined_Division	00nil_Combined_Sector	10125
apple	hardware	00nil_Combined_Sector	4500
apple	hardware	business	1350
apple	hardware	consumer	3150
apple	software	00nil_Combined_Sector	5625
apple	software	business	4950
apple	software	consumer	675
microsoft	00nil_Combined_Division	00nil_Combined_Sector	4500
microsoft	hardware	00nil_Combined_Sector	1890
microsoft	hardware	business	855
microsoft	hardware	consumer	1035
microsoft	software	00nil_Combined_Sector	2610
microsoft	software	business	1215
microsoft	software	consumer	1395

- **Tabular data** is the most common type of data
  - Yet, deep learning did not traditionally excel on it
  - SVMs, random forests, gradient-boosting, Auto-sklearn, …
- **Neural networks excel for large amounts of data**
  - But they are **slow** to train
  - But they are **prone to overfitting on small datasets**
- We care about the **long tail of small datasets**
  - Especially in scientific data
    - Biology
    - Medicine
    - Climate research
    - …



We introduce TabPFN, a GPT-like model for tabular data

- TabPFN is a **transformer pretrained to do tabular classification**
- Framed as next-word prediction: $x_1, y_1, \ldots, x_n, y_n, x_{n+1}, ?$

- To be more precise:

$$\{(x_1, y_1), \ldots, (x_n, y_n)\}, x_{n+1} \longrightarrow \boxed{\text{TabPFN}} \longrightarrow \hat{y}_{n+1}$$

- To be even more precise:

$$\{(x_1, y_1), \ldots, (x_n, y_n)\}, x_{n+1} \longrightarrow \boxed{\text{TabPFN}} \longrightarrow p(y_{n+1} \mid x_{n+1}, \{(x_1, y_1), \ldots, (x_n, y_n)\})$$

The very high-level intuition

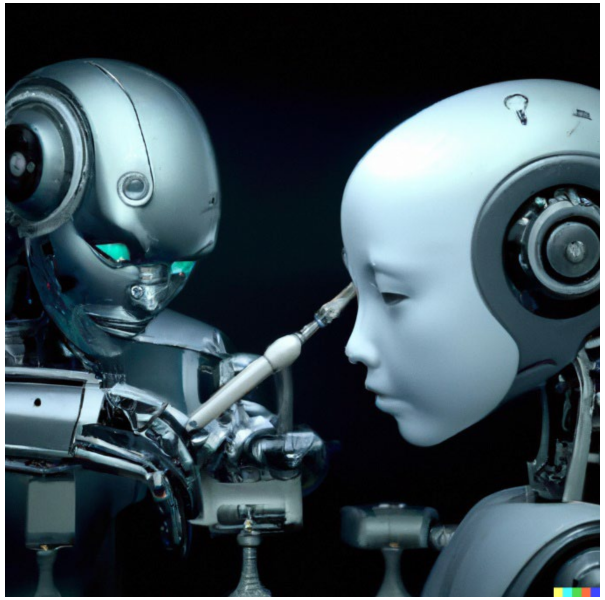
- TabPFN is a transformer with weights θ
  - A single forward pass directly approximates $p(y_{n+1} \mid x_{n+1}, \{(x_1, y_1), \ldots, (x_n, y_n)\})$
- We optimize θ to minimize average cross entropy loss across datasets
  - Across which datasets?
    - Millions of synthetically generated ones: $\{(x_1, y_1), \ldots, (x_{n+1}, y_{n+1})\}$
  - How do we train it?
    - Very standard transformer architecture (just drop the positional encoding)
    - Standard supervised learning with SGD

$$\{(x_1, y_1), \ldots, (x_n, y_n)\}, x_{n+1} \longrightarrow \boxed{\text{TabPFN}_\theta} \longrightarrow p(y_{n+1} \mid x_{n+1}, \{(x_1, y_1), \ldots, (x_n, y_n)\})$$

Preview of results

- TabPFN works astonishingly well
  - A single forward pass $\rightarrow$ **10.000x faster** than Auto-sklearn
  - But nevertheless **SOTA performance**

- A very different approach to algorithm design
  - We do not code an algorithm but merely **define the type of data the algorithm should work well for**
  - **The algorithm is fully learned**
    - It lives in the weights of a transformer



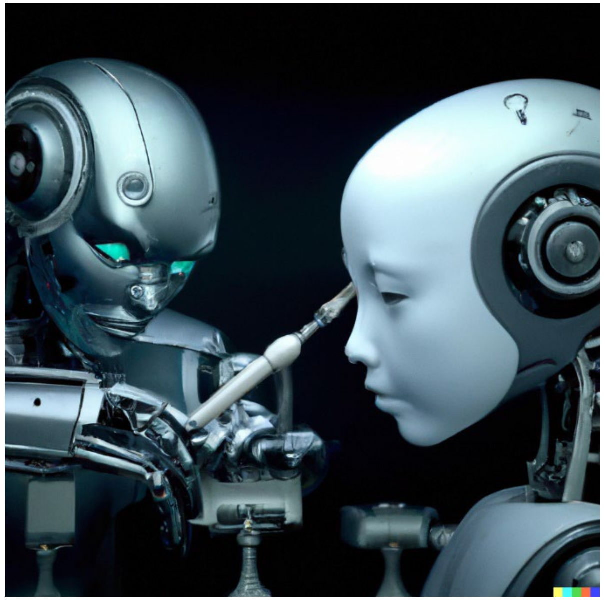
- **Received a lot of attention**

  $\rightarrow$ **Best paper award** at NeurIPS 2022 workshop on tabular representation learning

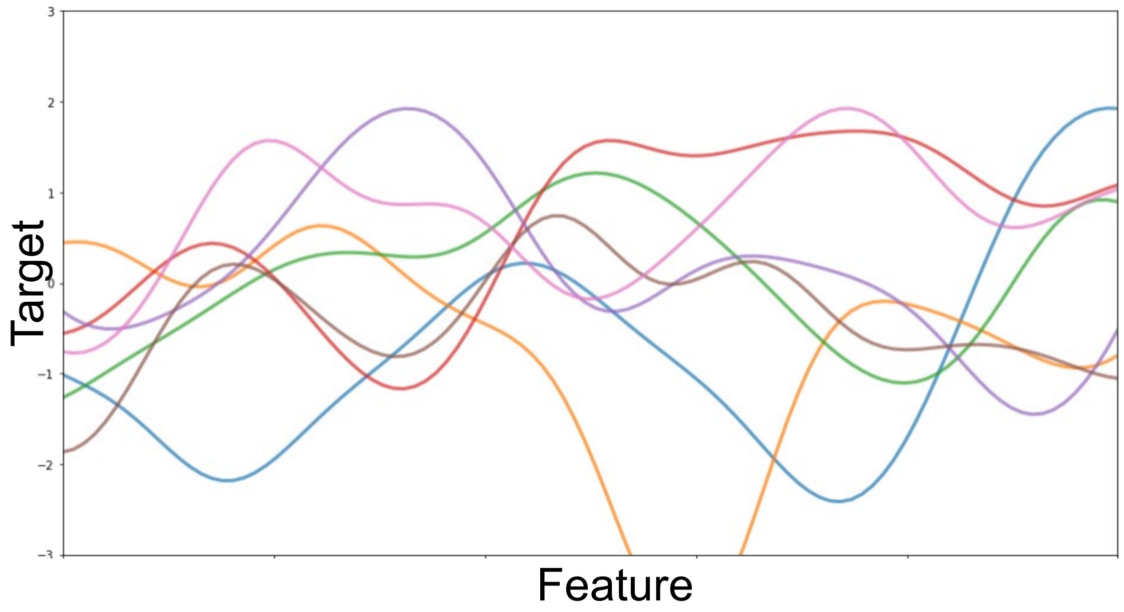
  $\rightarrow$ **1M views** of the **tweet** introducing the method

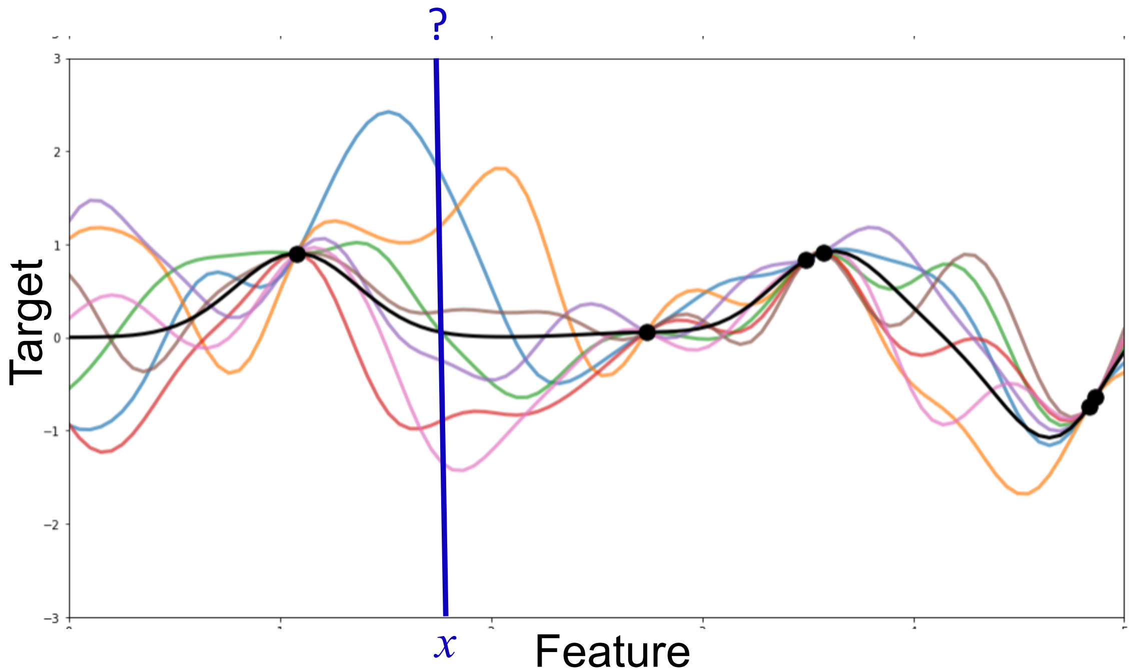
Outline

- Overview of Deep Learning 2.0
- Deep Dive: Meta-Learning a New ML Algorithm
  - Premises and Preview of Results
  - PFNs: Transformers can approximate Bayesian Inference
  - TabPFN: PFNs with a prior over tabular datasets
- Outlook



PFN approximates Bayesian predictions





Prior over functions
parameterized by latents t

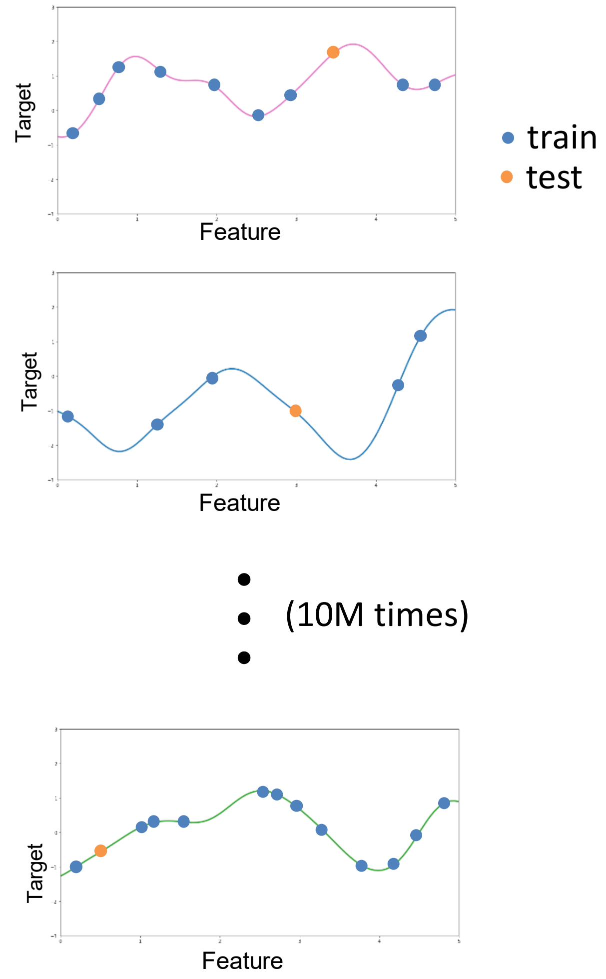
Posterior $p(t|D) = \frac{p(D|t)p(t)}{p(D)}$

Intractable to compute exactly!

Posterior predictive distribution $p(y|x, D) = \int p(y|x, t)p(t|D)dt$

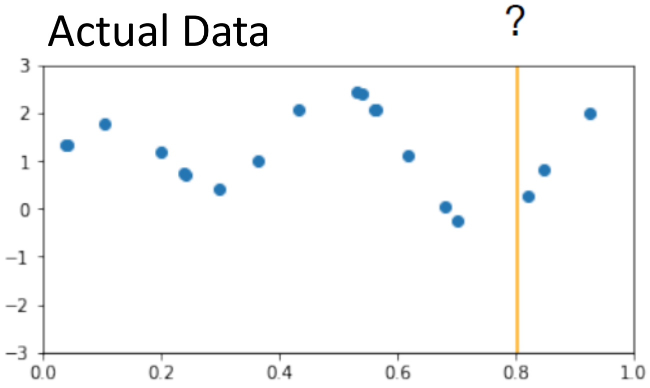
Illustration of Prior-Fitted Networks (PFNs)

[ICLR 2022]

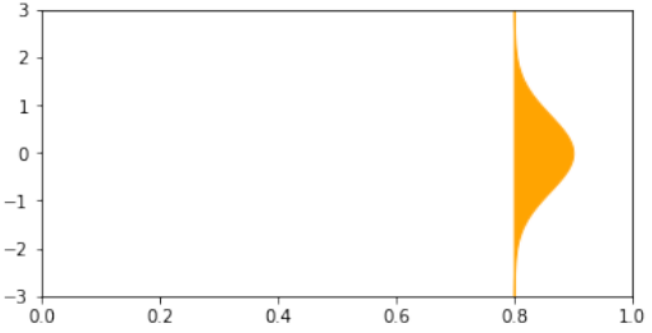


Samples from the prior:
a data-generating mechanism

Learn PFN_θ to
predict test from train



PFN_θ forward pass

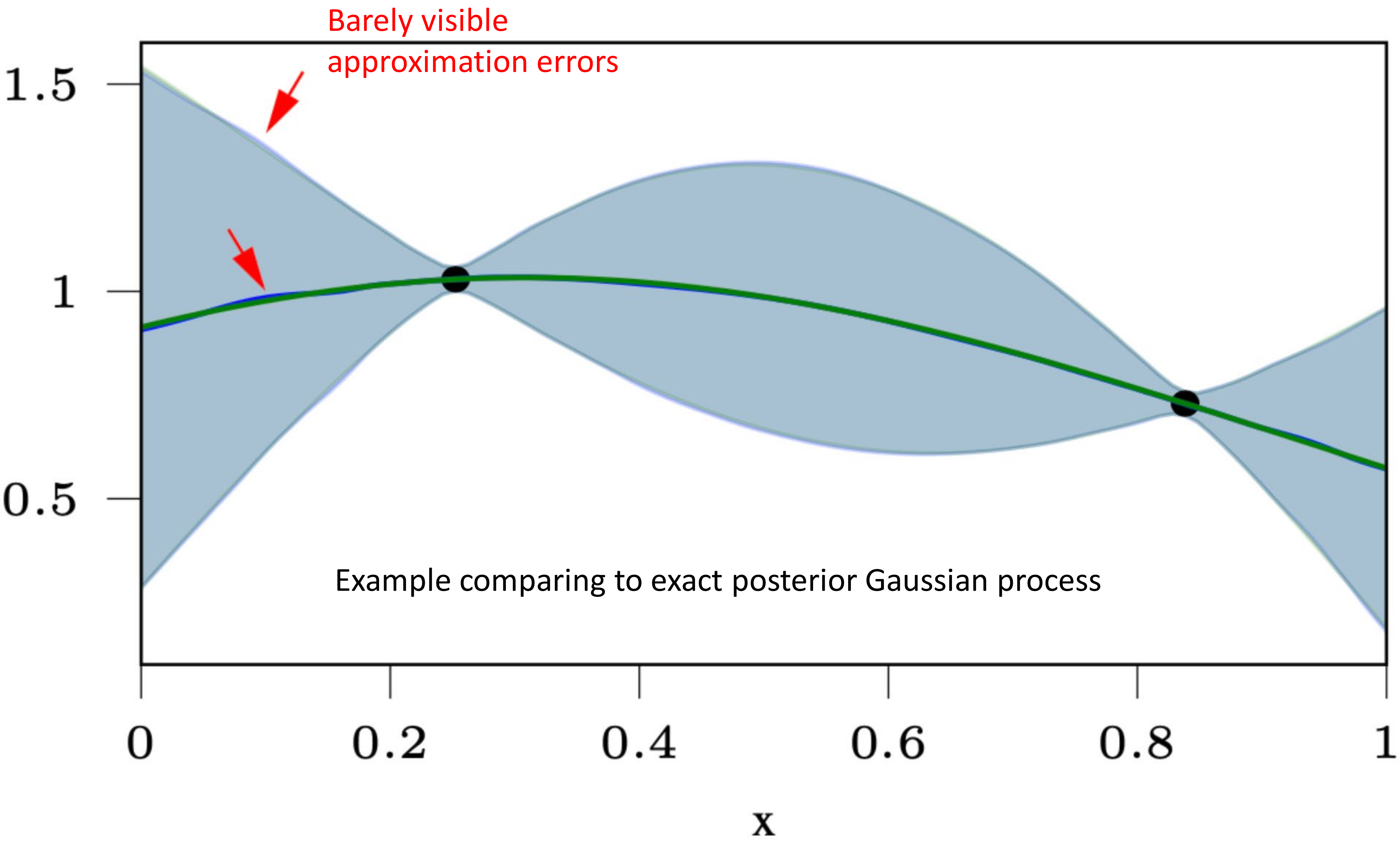


Posterior predictive distribution

We have thus meta-learned to
approximate Bayesian inference,
purely by supervised learning
of data generated with the prior

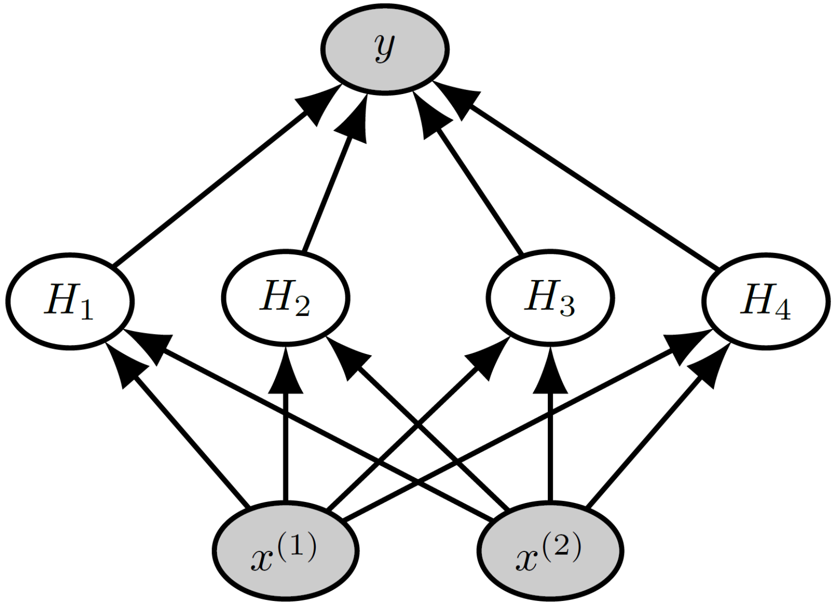
PFNs can predict the true posterior arbitrarily closely

[ICLR 2022]

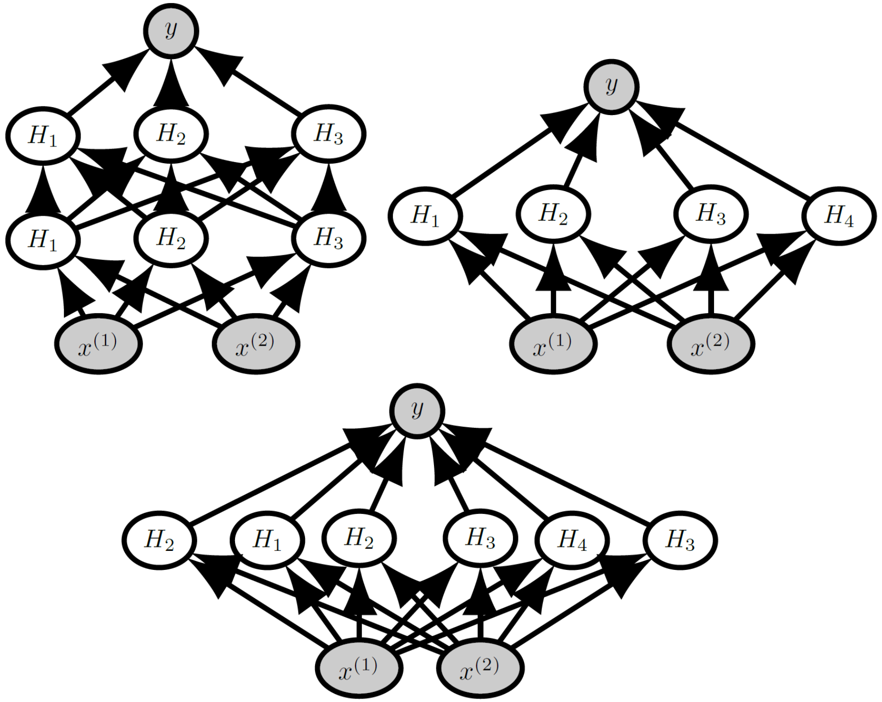


PFNs enable Bayesian deep learning in a forward pass [ICLR 2022]

- Prior: weights of a given neural net
- Posterior predictive: Bayesian neural net
  - 10000x speedups over MCMC etc

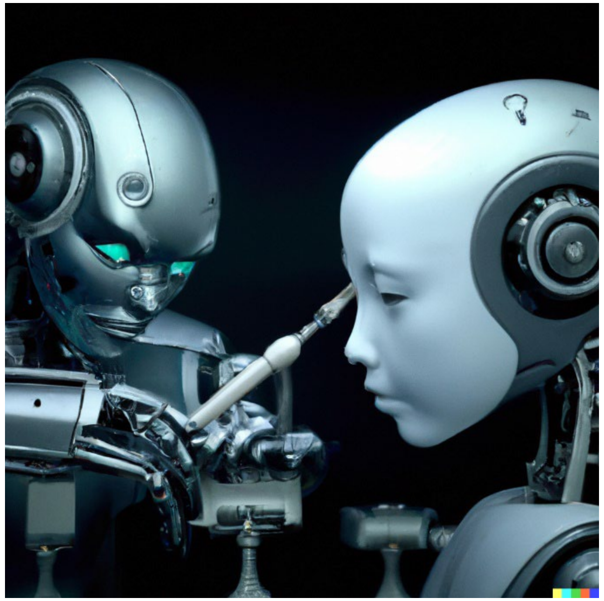


- Prior: different neural architectures & their weights
- Posterior predictive: "**Bayesian NAS**"
  - Not even possible with MCMC etc



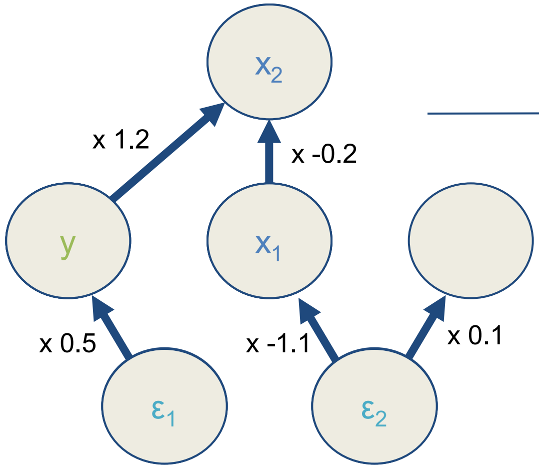
Outline

- Overview of Deep Learning 2.0
- Deep Dive: Meta-Learning a New ML Algorithm
  - Premises and Preview of Results
  - PFNs: Transformers can approximate Bayesian Inference
  - ➡ TabPFN: PFNs with a prior over tabular datasets
- Outlook



TabPFN prior: integrating principles from causality

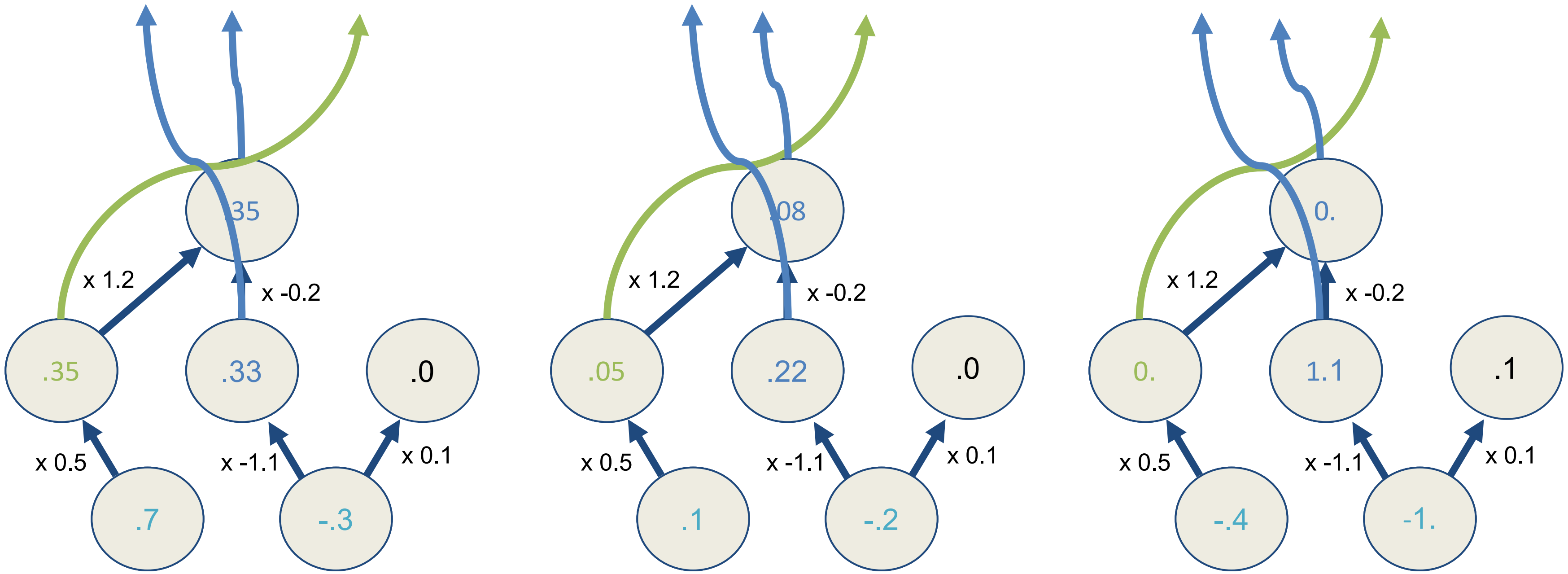
Sample & initialize
a causal graph



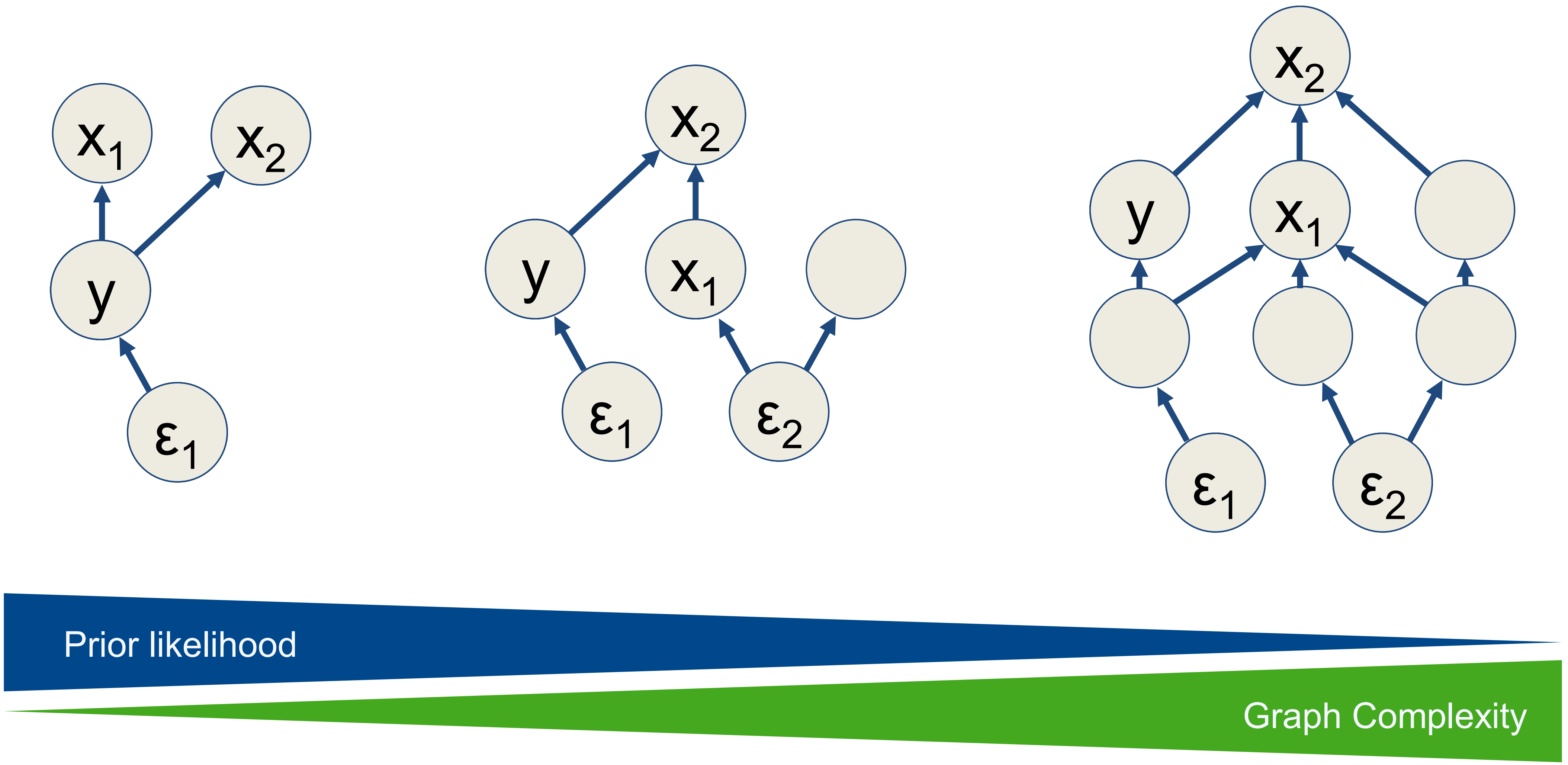
Build dataset: output >0.2?

1	0	0
{((.33,.35), .35),	((.22,.08), .05),	((0.,1.1), 0.)}

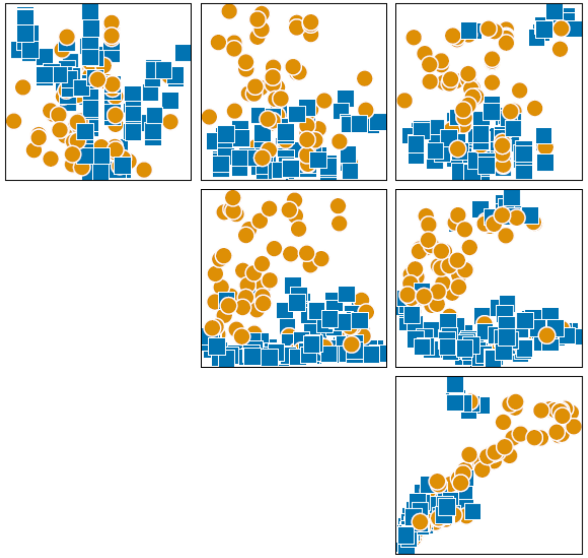
Sample noise
per example
& forward pass



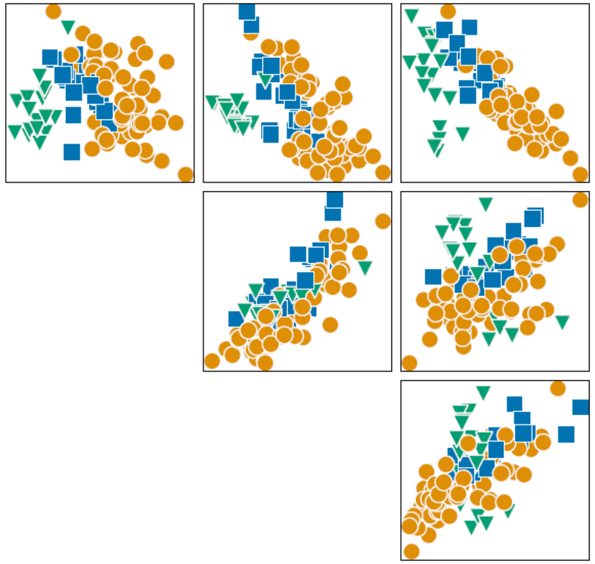
TabPFN prior: simplicity principle

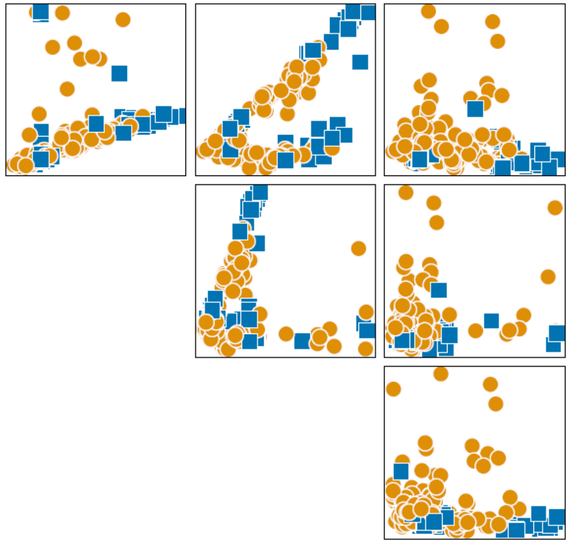


The generated datasets look similar to real datasets

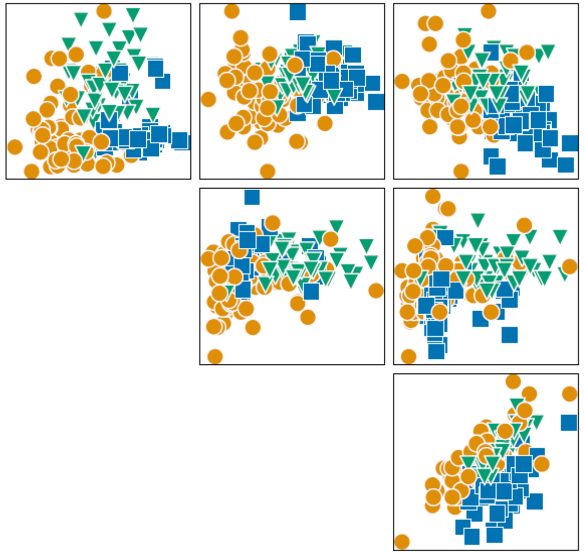


Synthetic datasets



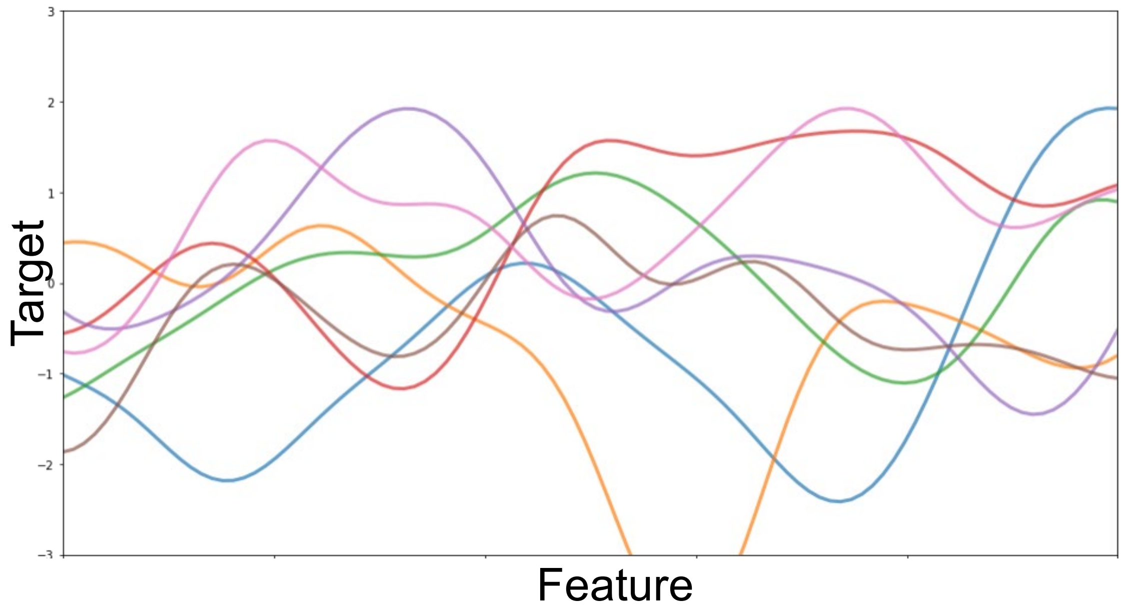


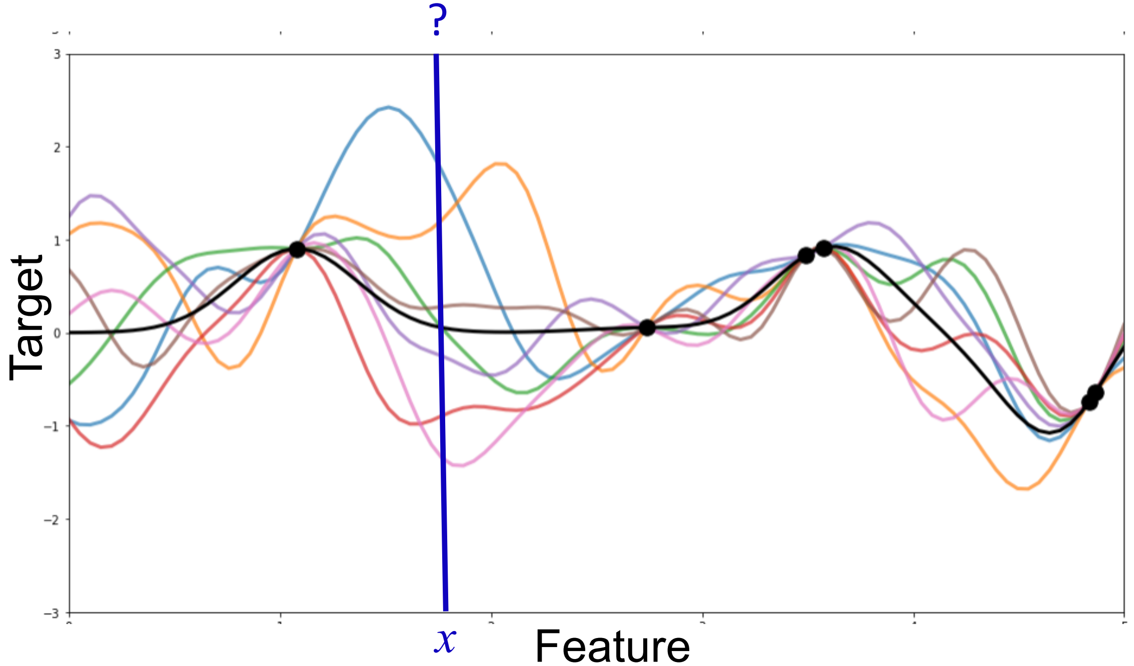
Parkinsons dataset



Wine dataset

Relation to Bayesian supervised learning





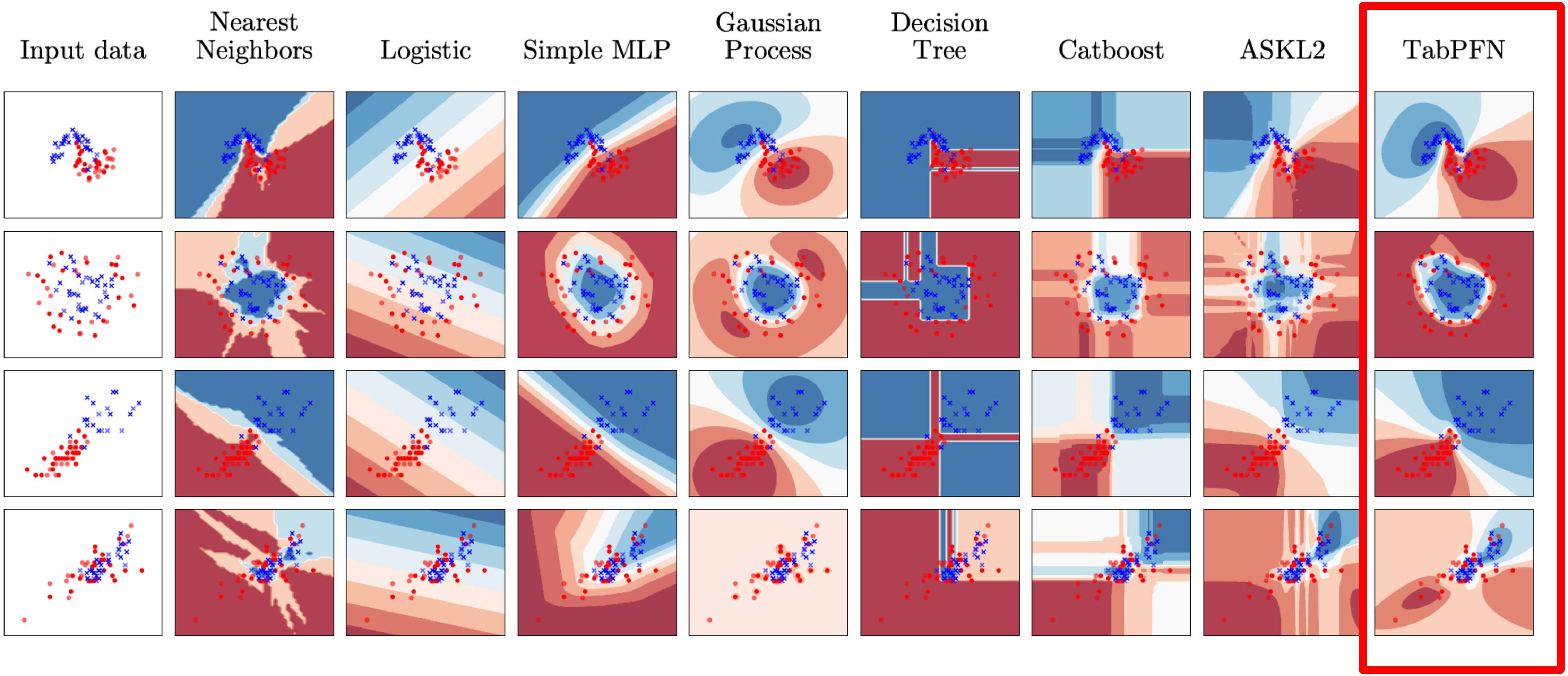
Prior over functions parameterized by latents t

- Structural causal model: graph structure, weights, activation functions, etc

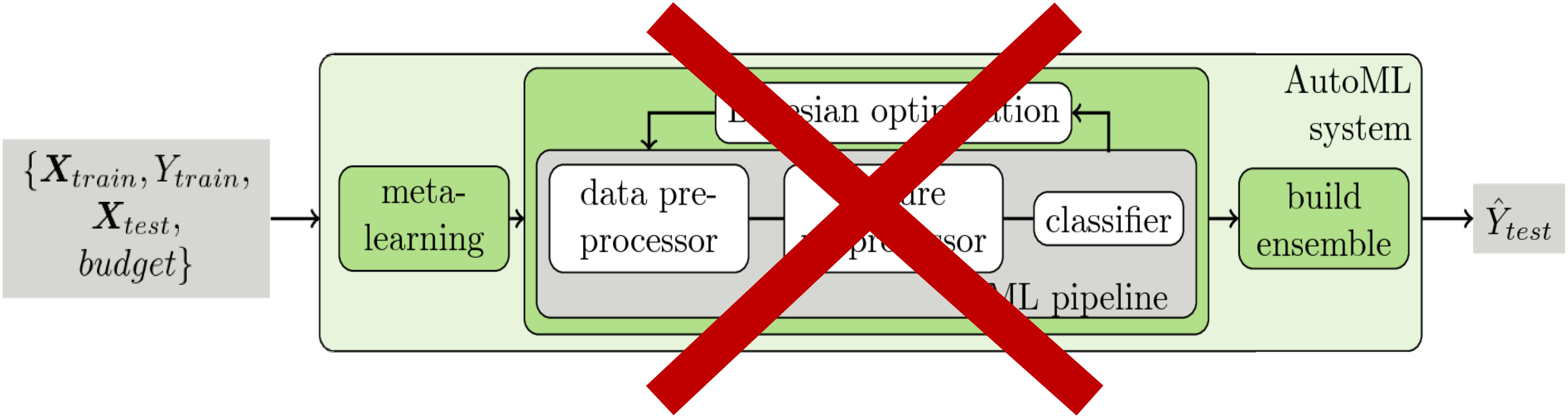
Posterior $p(t|D) = \frac{p(D|t)p(t)}{p(D)}$

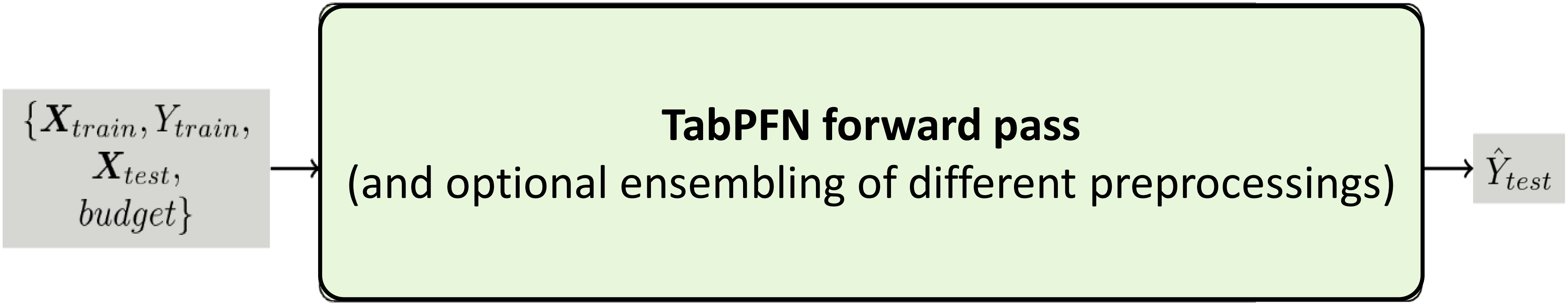
Posterior predictive distribution $p(y|x, D) = \int p(y|x,t)p(t|D)dt$

Qualitative result: smooth & well-calibrated predictions



Simplicity: it's just a forward pass

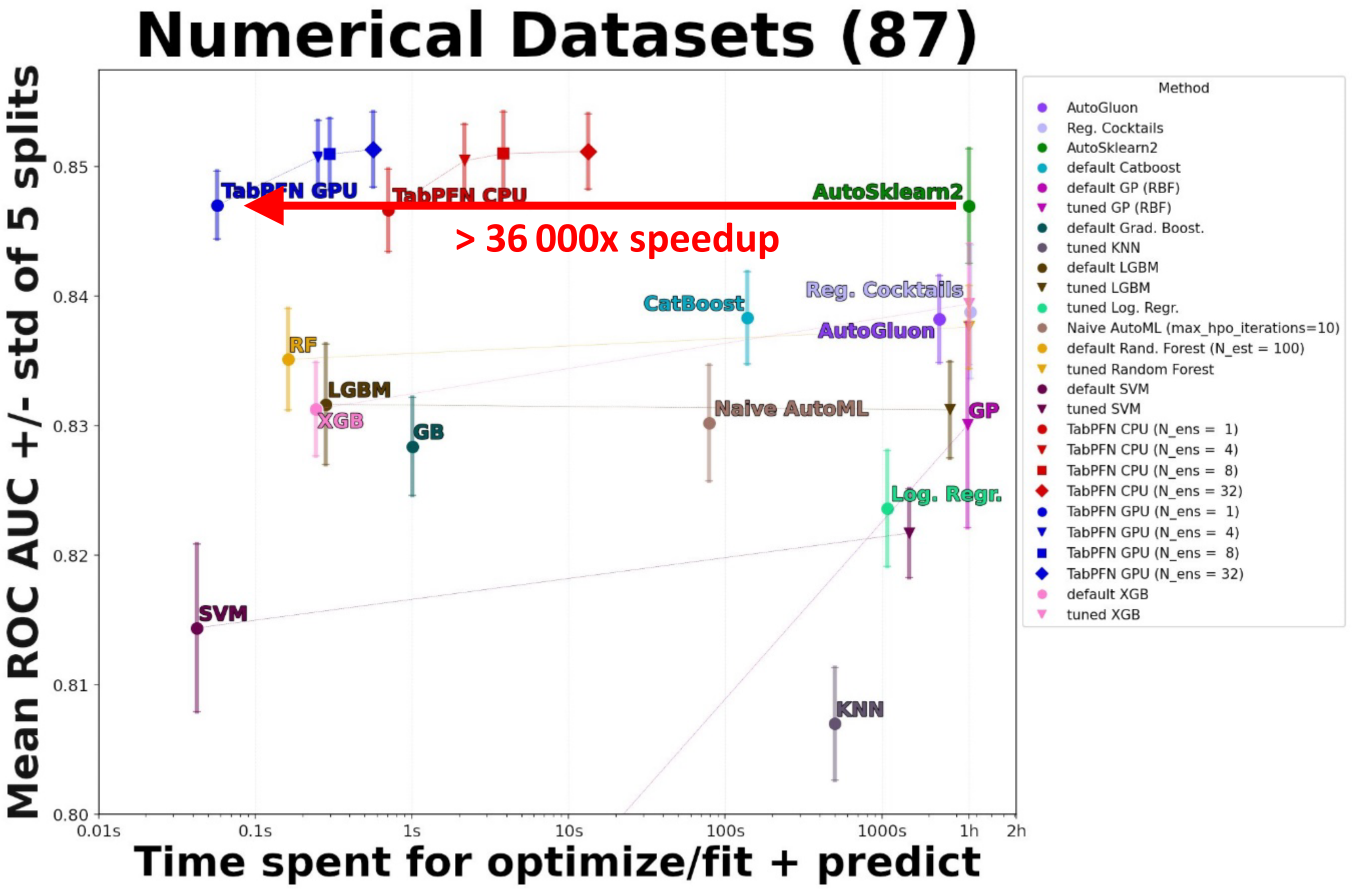




Quantitative result (87 numerical datasets without missing values)

- **Better performance in 1s than than any other ML / AutoML method in 1h**
  - Disclaimer: these are average results; TabPFN is not the best on every single dataset

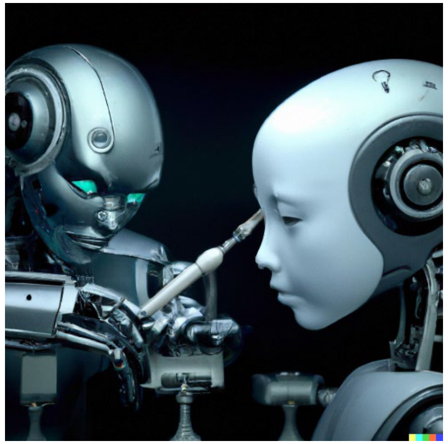
- **Current limitations**
  - Size: up to 1000 data points, 100 features, 10 classes
  - Not (yet) designed for: categorical features, missing values, uninformative features
  - High inference time



TabPFN summary

- TabPFN computes **posterior Bayesian inference** for the given prior
  - In our prior: elements of causality & simplicity
- **SOTA performance** on **small tabular datasets**

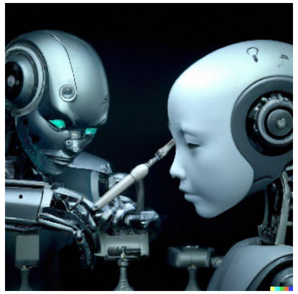
- **Potentially disruptive to ML algorithm development**
  - **TabPFN is fully learned**
  - Instead of manual algorithm development:
    Just state the type of data to fit well on and hit “train”



Outline

Deep Learning 2.0: Towards AI that Builds and Improves AI

European Research Council



- Overview of Deep Learning 2.0
- Deep Dive: Meta-Learning a New ML Algorithm
- ➡ Outlook

TabPFN Follow-ups

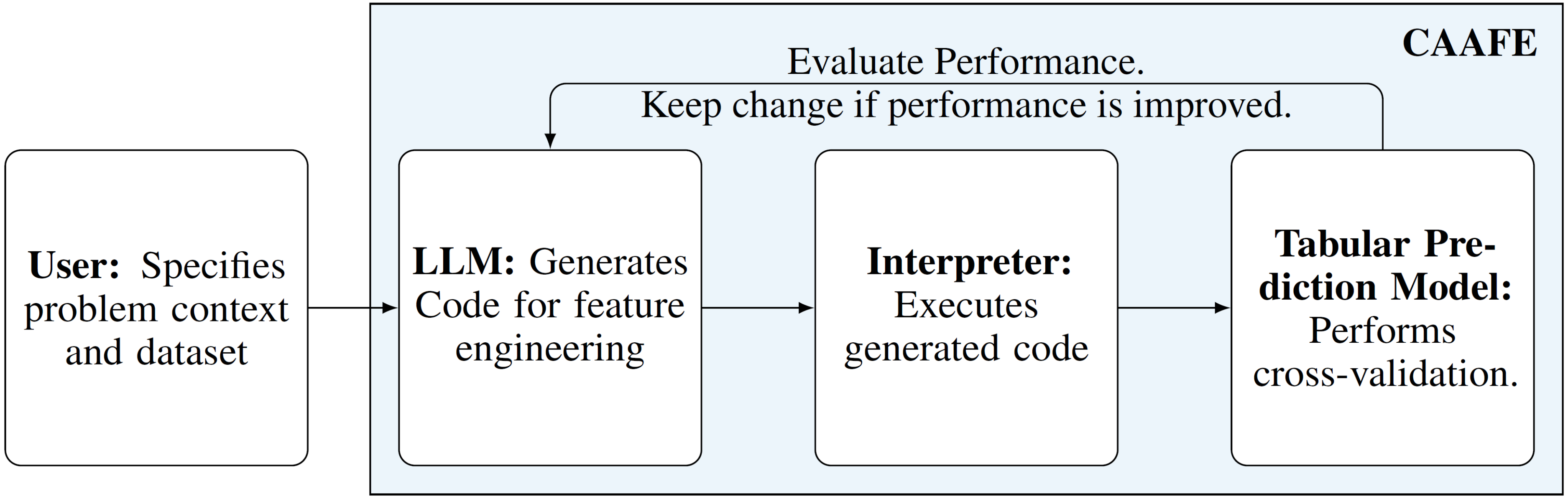
- **Fix remaining limitations**
  - Goal: create new default algorithm for tabular data, better & faster than XGBoost
  - Approach for fixing the prediction latency issue: learning to distill
    - Use a hypernetwork (a “hen”) that generates the weights of a simple MLP (an “egg”)
    - Training the hypernetwork (hen) is like usual TabPFN training
      - Loss function for a given meta-dataset D: how good are the y_{test} predictions by the egg laid for D and x_{test}
    - Given a new actual dataset D’:
      - Fit: Use the hen to lay an egg for D’
      - Predict: Use the egg to make very fast predictions for new test samples

- Many **open research questions**
  - Understanding how the learned TabPFN algorithm works
    - This might also lead to general insights about in-context learning
  - Also approximating posterior inference over the model parameters
    - E.g., answer questions like “What’s the probability that X causes Y?”

Foundation Models for AutoML

- Foundation models for HPO and NAS
  - **Optformer** [Chen et al, 2022]

- Large language models for semi-automated data science
  - **CAAFE**: use GPT-4 to engineer new features based on problem context [NeurIPS'23]



AutoML for Foundation Models

- **Optimize choices for the training process (budget: < 2 full trainings)**
  - Transformer tuning by gradient-based NAS with weight entanglement
  - Multi-objective gradient-based NAS
  - Exploiting scaling laws
  - Exploiting subsidiary objective functions (training stability, etc)

- **AutoML for fine-tuning foundation models**
  - **Fine-tuning is cheap** (especially parameter-efficient fine-tuning, like LoRa)
    - Select which model to fine-tune & how [ICML'22; under review]
  - We can thus easily afford multi-objective AutoML
    - Objectives: robustness, truthfulness, lawfulness, alignment, etc

Call to arms: Trustworthy AI is key

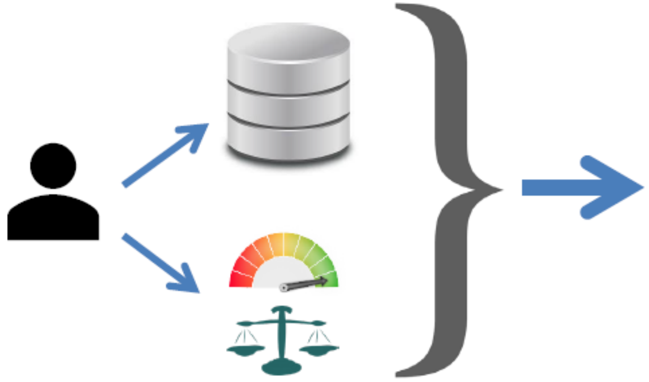
- AI is currently progressing at an incredible pace
  - AI systems will govern an ever-increasing part of our lifes

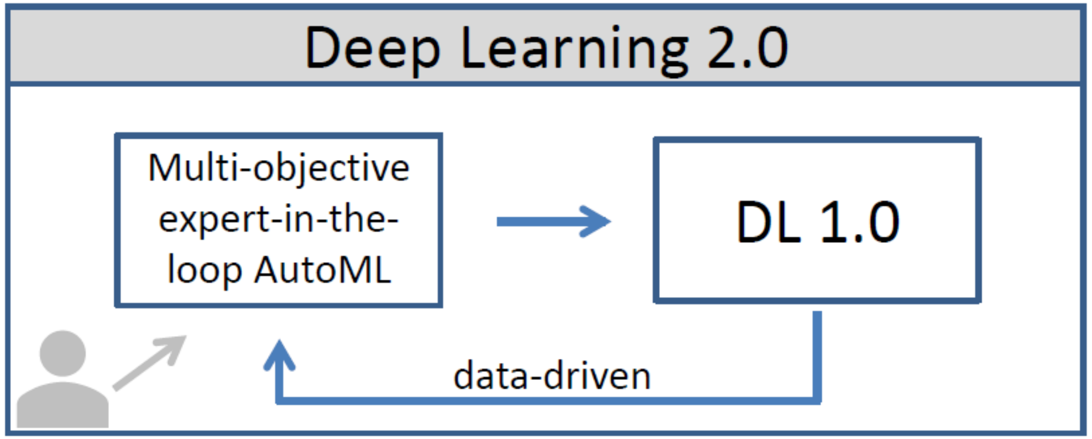
- We need to ensure that we can trust these systems!

- **If you can, please work on trustworthy AI!**
  - Robustness
  - Interpretability
  - Lawfulness
  - Fairness
  - Privacy
  - Alignment with human values

Take-aways

Deep Learning 2.0: expert-guided Auto-DL for the objectives at hand



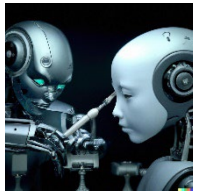


1. Trustworthy AI by design

– DL 2.0 directly optimizes for user‘s objectives

2. Breakthrough results

– DL 2.0 is now SOTA on tabular data



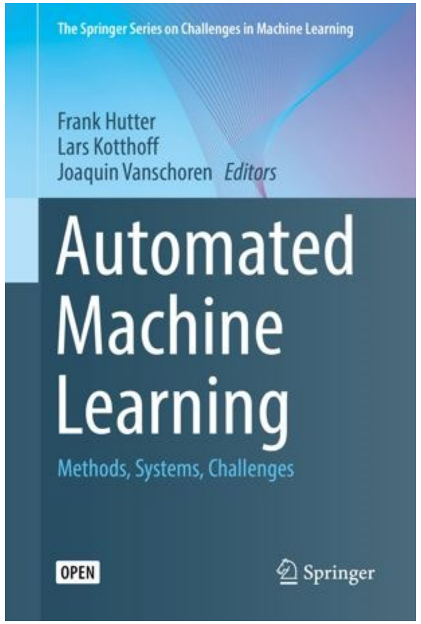
3. It doesn‘t have to be expensive

– DL 2.0 includes many speedup methods

all our code
is open-source:
github.com/automl

AutoML.org

get involved:
AutoML conference series
automl.cc



Thank you for your attention!

Funding sources

BOSCH

European Research Council (erc)

Emmy Noether-Programm
DFG Deutsche Forschungsgemeinschaft

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

DFG
Priority Programme 1527

TAILOR
https://tailor-network.eu/

DFG Deutsche Forschungsgemeinschaft

European Commission

BrainLinks BrainTools
acting.thoughts

My fantastic team

@FrankRHutter
@AutoML_org

AutoML.org

We're hiring!