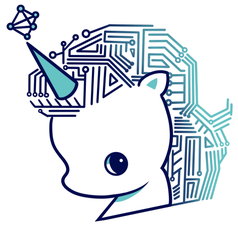
Advanced Topics II

Principles of Deep Learning (AI502/IE408/IE511)

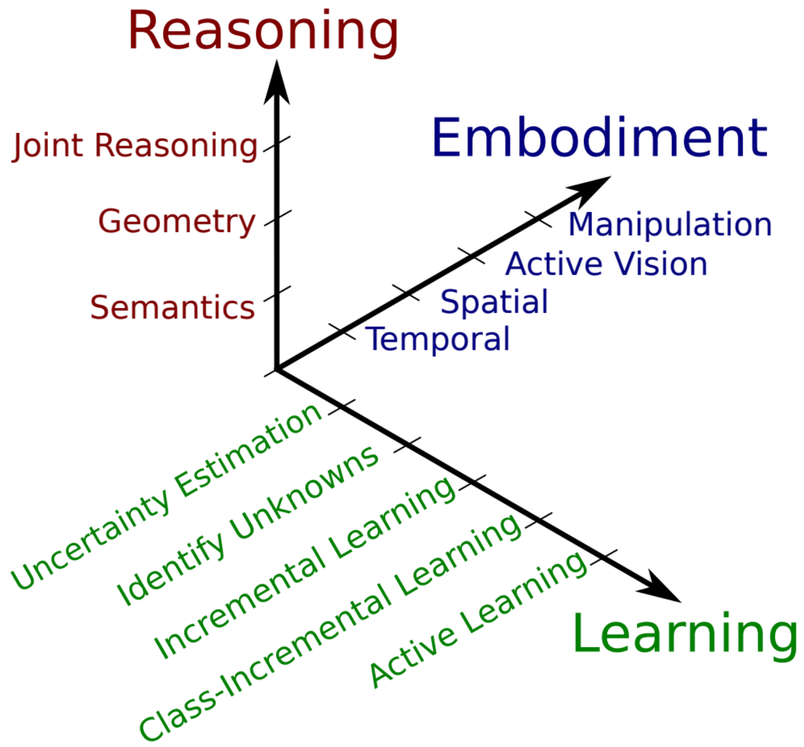
Sungbin Lim (UNIST AIGS & IE)



Contact: ai502deeplearning@gmail.com

The Direction of AI Research

- Challenges of AI Research
  - seeing like human (Detectron)
  - listening like human (LibriSpeech)
  - speaking like human (Tacotron)
  - drawing like human (GAN, Diffusion)
  - understanding like human (BERT)
  - **learning** like human (AutoML?)
  - **reasoning** like human (FM?)



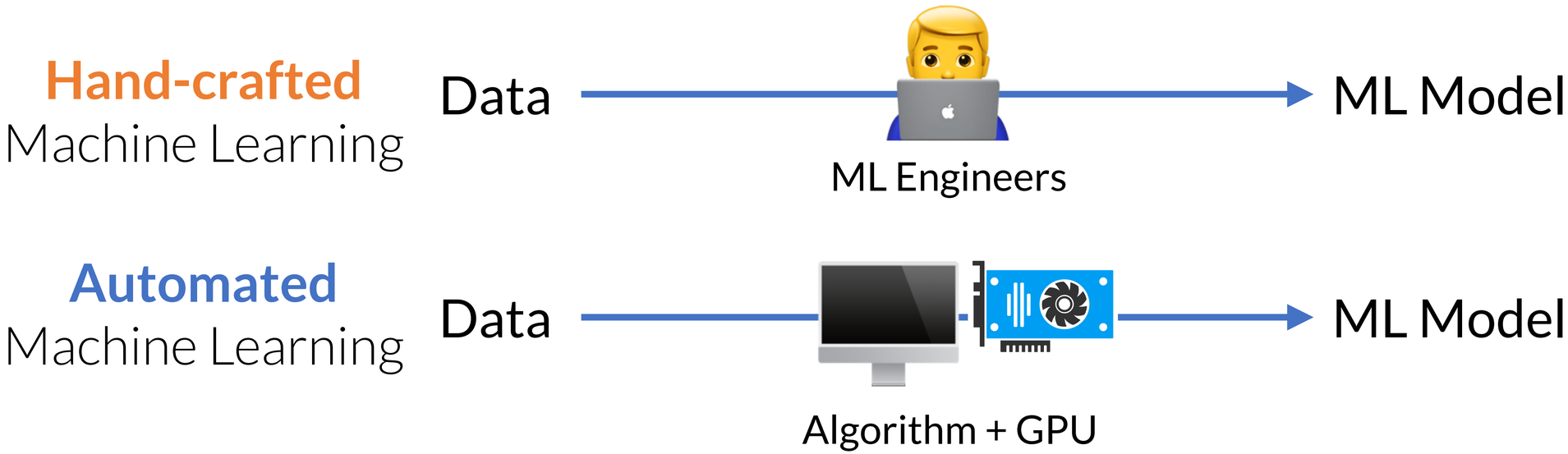
The Limits and Potentials of Deep Learning for Robotics, Niko et al., (2018)

Everyone knows ML/DL is good

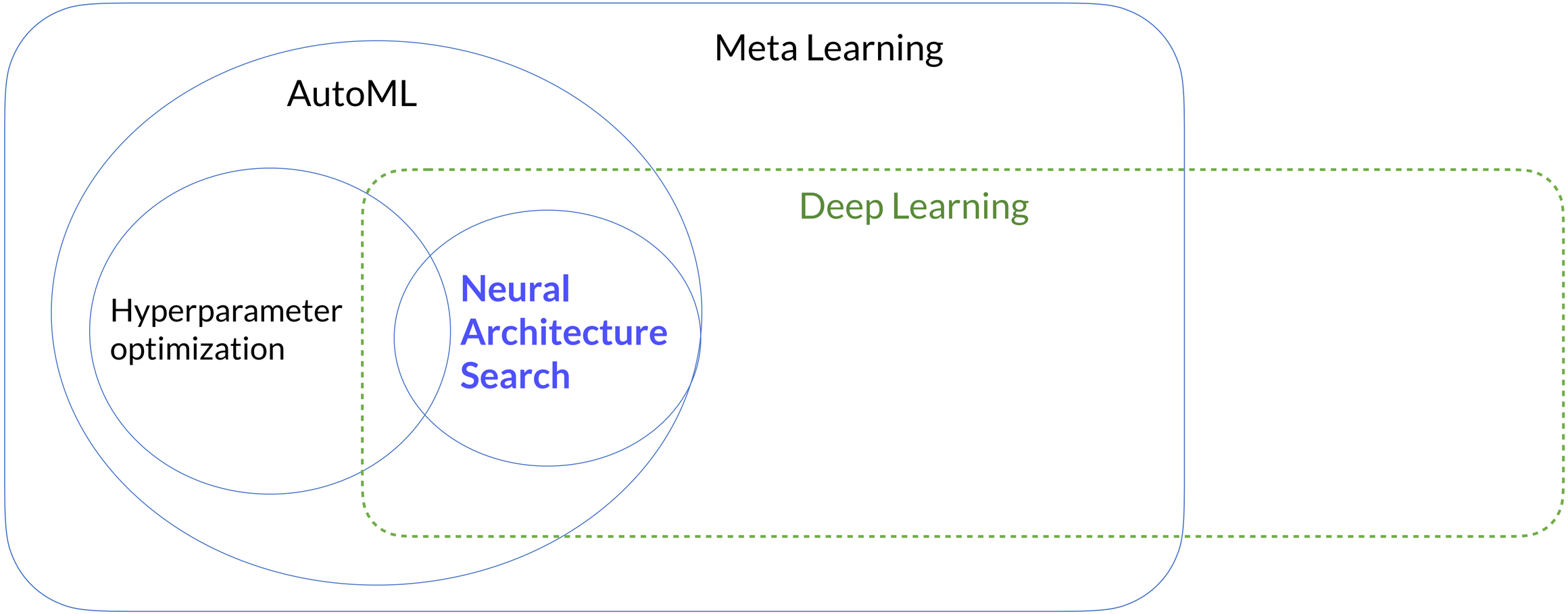
- But currently employed architectures have mostly been developed manually by **human experts**
  - time consuming
  - error prone process
- Growing interest in automating machine learning

What is AutoML?

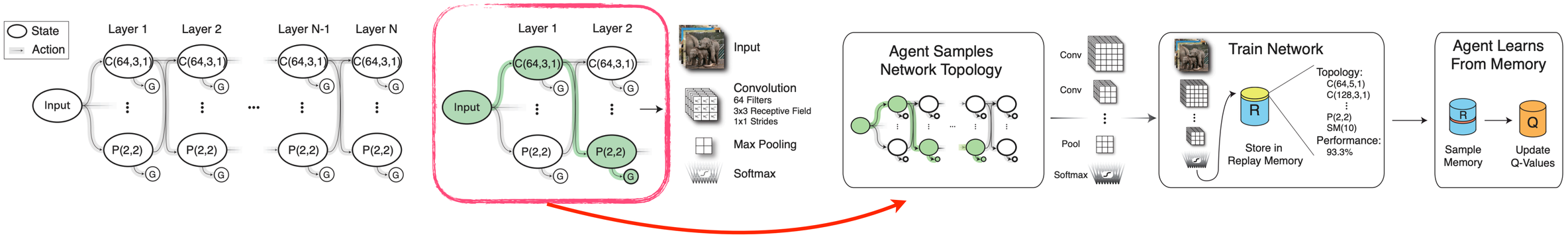
- Millions of organizations worldwide has ML problems
- AutoML allows **non-experts** to solve ML problems

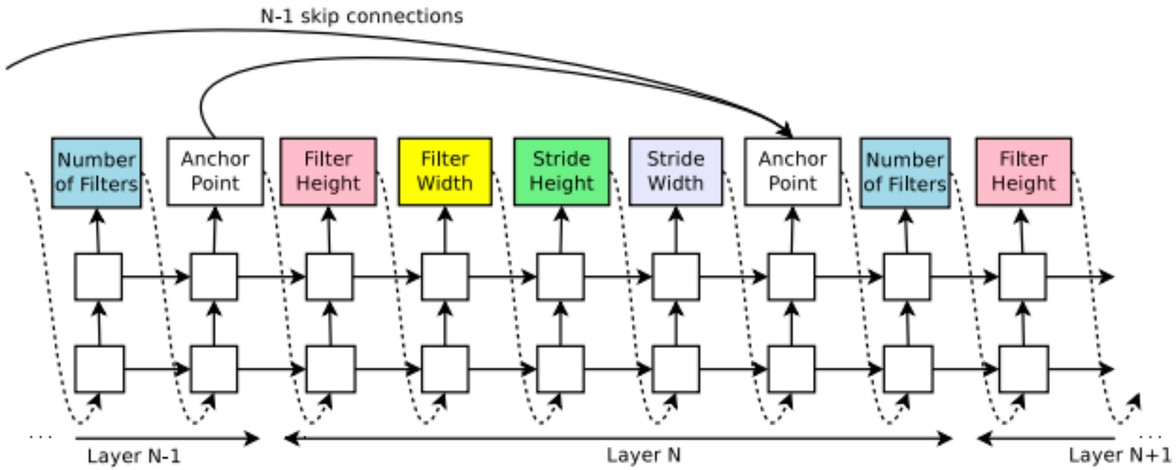


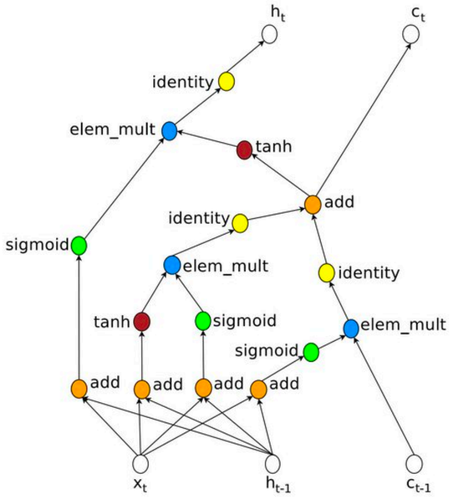
Vénn Diagram for AutoML

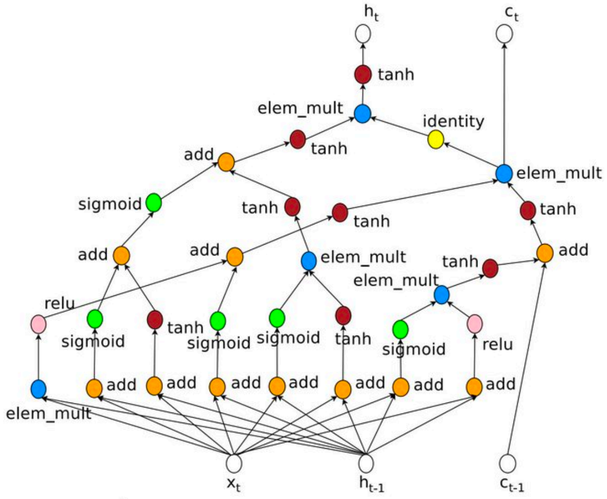


Neural Architecture Search

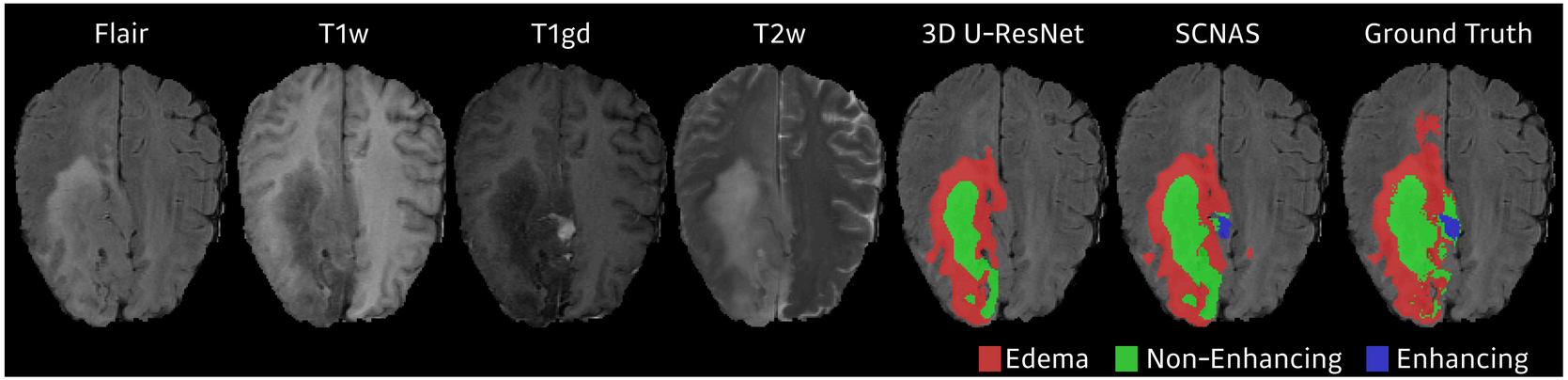


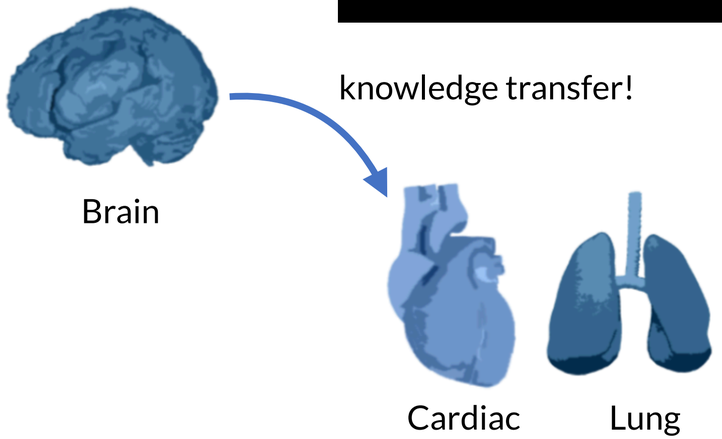






NAS for 3D Medical Images



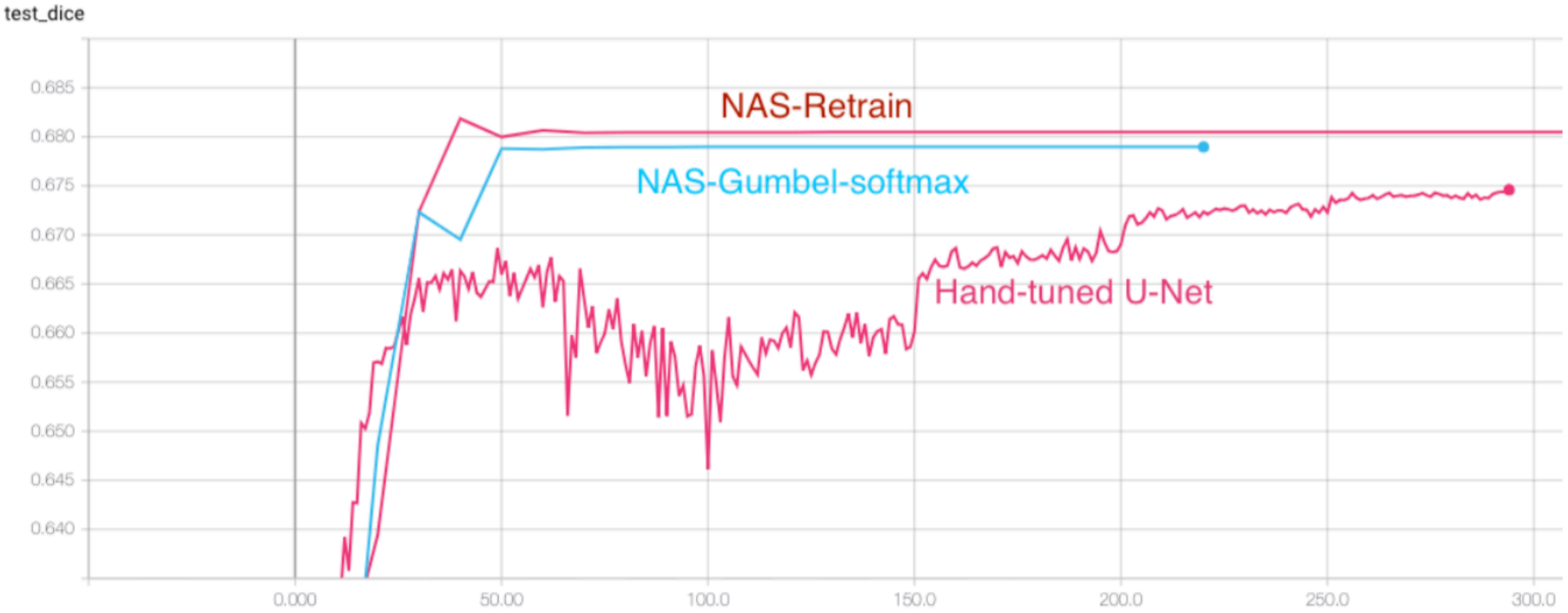


	Brain Tumor (MRI)				Heart (MRI)	Lung (CT)
Label	Edema	Non-Enhancing	Enhancing	Average	Left Atrium	Tumor
3D nnU-Net [2]	80.71	62.22	79.07	74.00	92.45	55.87
3D U-ResNet	70.74	56.69	73.23	66.89	91.48	63.28
SCNAS	80.41	59.85	78.50	72.92	91.29	64.82
SCNAS(transfer)	-	-	-	-	91.91	68.62

Scalable Neural Architecture Search for 3D Medical Image Segmentation, ***MICCAI*** (2019)

Joint work with S. Kim (Kakao Brain), I. Kim (Kakao Brain), W. Baek (Kakao Brain), C. Kim (Kakao Brain), H. Cho (SNU), B. Yoon (Kakao Brain), T. Kim (MILA)

NAS for 3D Medical Images



Brain

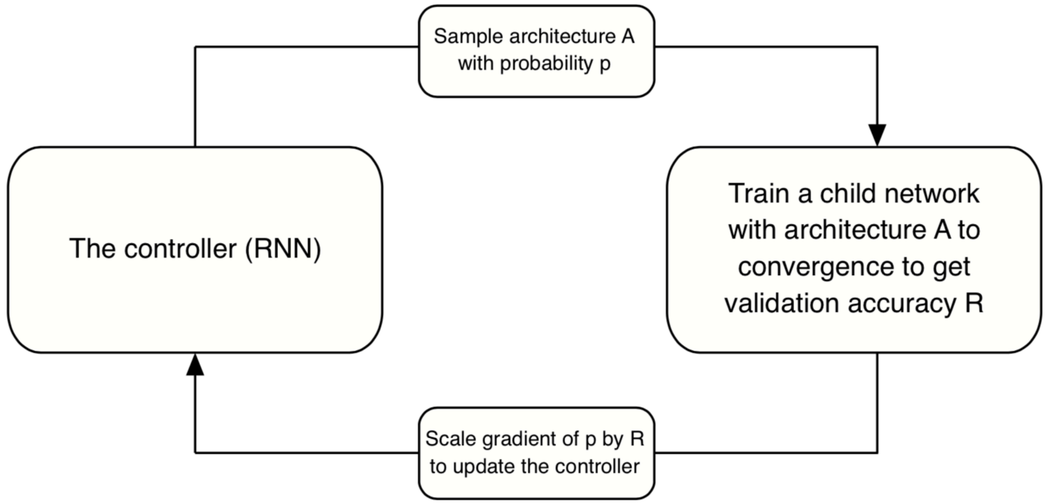
(CT)
ıor
37
28
32
62

Scalable Neural Architecture Search for 3D Medical Image Segmentation, ***MICCAI*** (2019)

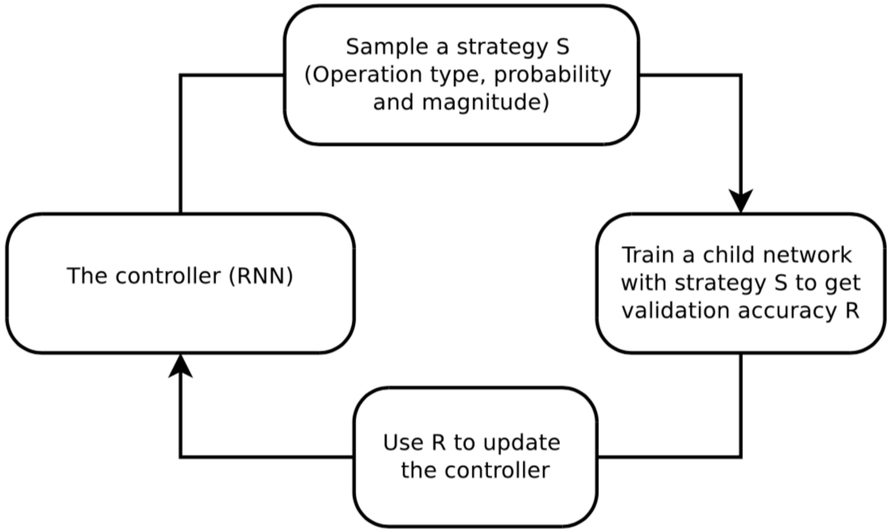
Joint work with S. Kim (Kakao Brain), I. Kim (Kakao Brain), W. Baek (Kakao Brain), C. Kim (Kakao Brain), H. Cho (SNU), B. Yoon (Kakao Brain), T. Kim (MILA)

AutoAugment for Insufficient Data

- NAS = automated **architecture** search
- AutoAugment = automated **augmentation** search



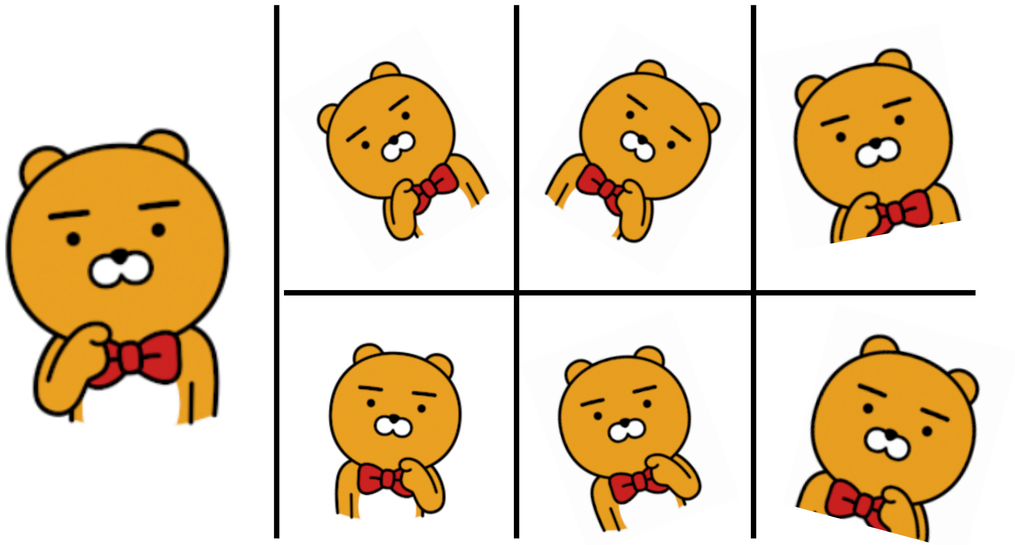
NAS

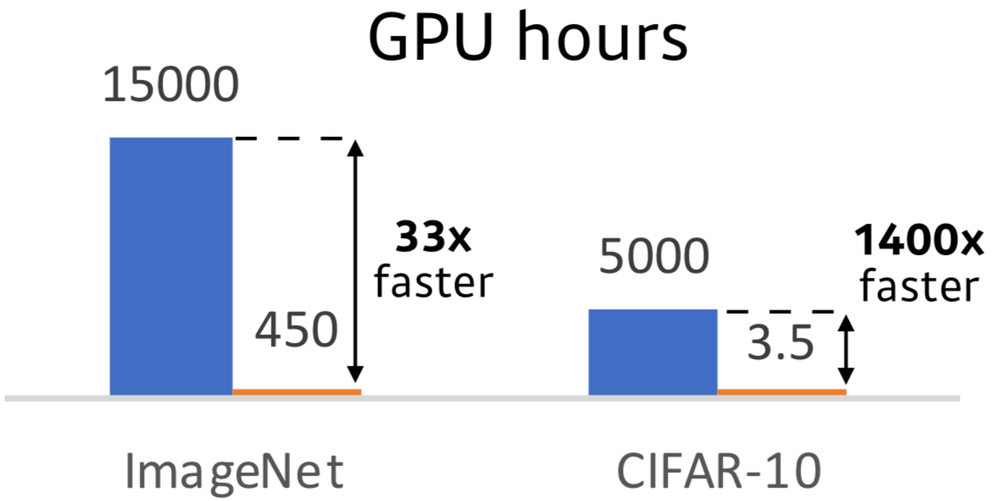


AutoAugment

Fast AutoAugment

- Pre-training is important
- Random augmentation shows poor performance



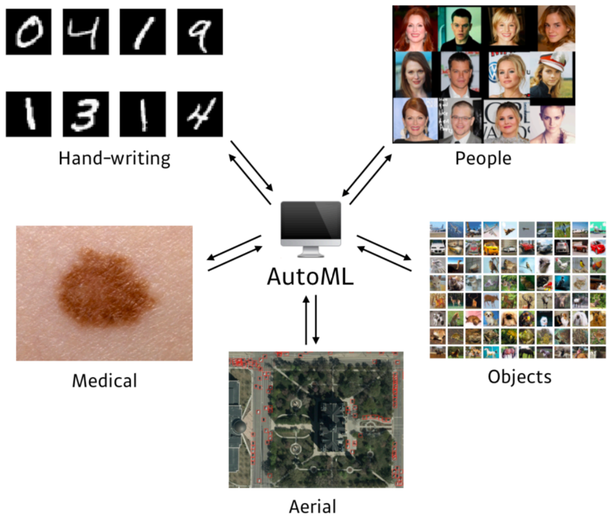


Fast AutoAugment, ***NeurIPS*** (2019)

Joint work with I. Kim (Kakao Brain), C. Kim (Kakao Brain), T. Kim (MILA), S. Kim (Kakao Brain)

NeurIPS AutoDL Challenge

- 20min with 1 GPU
  - **Fast AutoAugment**



Username	Rank	Major DL Framework	Strategy	Prize
kakaobrain	1st	PyTorch	Fast AutoAugment	$2000
tanglang	2nd	PyTorch	Inspired by Fast AutoAugment	$1500
kvr	3rd	PyTorch	Inspired by Fast AutoAugment	$500

NIPS 2019 Auto Computer Vision Challenge

Top **1st** Place on AutoCV & AutoCV2

NeurIPS AutoDL challenges

Enter AutoSpeech (deadline Oct 15)

Enter AutoWeakly (deadline October 29)

Congratulations to the **ECML PKDD conf**. AutoCV2 winners:

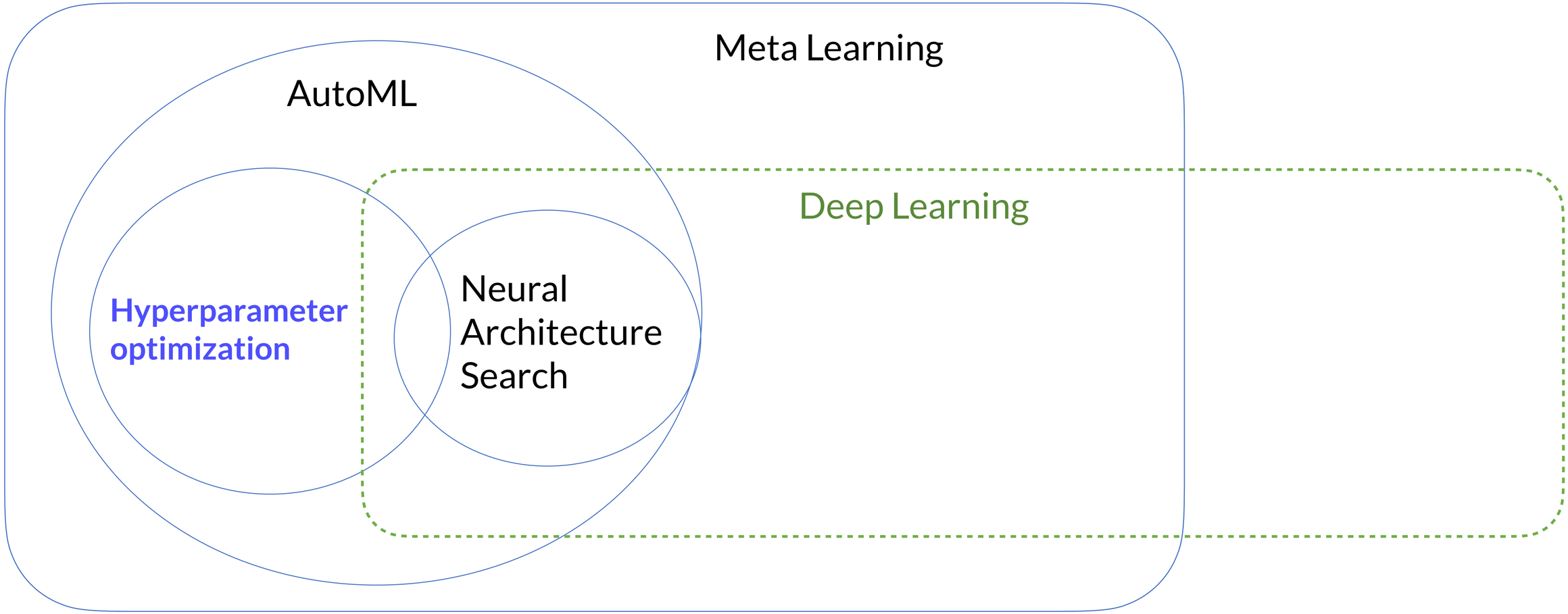
- **First place:** kakaobrain [**GitHub repo**]
- **Second place:** tanglang [**GitHub repo**]
- **Third place:** kvr [**GitHub repo**]

They will be invited to present at the **discovery challenge workshops of ECML PKDD**.

The **IJCNN conf.** AutoCV winners [**slides**] were:

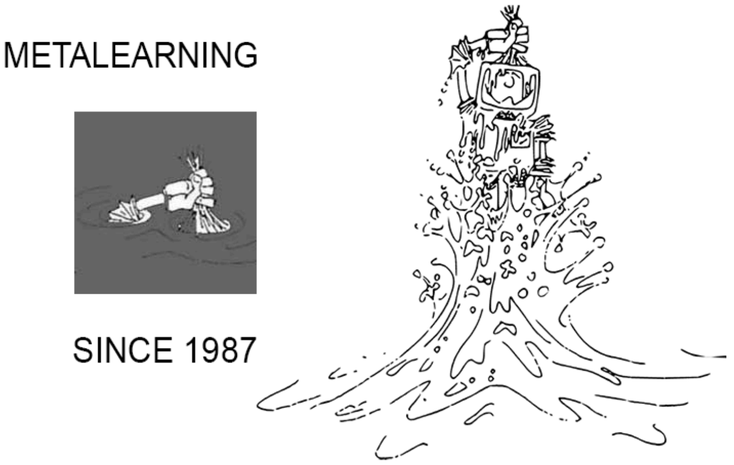
- **First place:** KakaoBrain [**GitHub repo**]
- **Second place:** DKKimHCLee [**GitHub repo**][**slides**]
- **Third place:** base_1 [**GitHub repo**][**slides**]

Vénn Diagram for AutoML



Learning vs Meta Learning

- Learning Algorithm A
  - input: $\mathcal{D}_{\texttt{train}} = \{(x_i, y_i)\}$
  - output: M
  - objective: $\mathcal{D}_{\texttt{test}} = \{(\tilde{x}_i, \tilde{y}_i)\}$
- Meta Learning Algorithm $\mathfrak{A}$
  - input: $\mathfrak{D}_{\texttt{meta-train}} = \{(\mathcal{D}^{(t)}_{\texttt{train}}, \mathcal{D}^{(t)}_{\texttt{test}})\}_{t=1}^{T}$
  - output: A
  - objective: $\mathfrak{D}_{\texttt{meta-test}} = \{(\tilde{\mathcal{D}}^{(t)}_{\texttt{train}}, \tilde{\mathcal{D}}^{(t)}_{\texttt{test}})\}_{t=1}^{\tilde{T}}$



Jürgen Schmidhuber (1987)

Bilevel optimization framework

- Bilevel formulation for HPO and Meta Learning

$$\min\{f(\lambda) : \lambda \in \Lambda\}, \tag{1}$$

where function $f : \Lambda \to \mathbb{R}$ is defined at $\lambda \in \Lambda$ as

$$f(\lambda) = \inf\{E(w_\lambda, \lambda) : w_\lambda \in \arg\min_{u \in \mathbb{R}^d} L_\lambda(u)\}. \tag{2}$$

model $g_w : \mathcal{X} \to \mathcal{Y}$

inner objective $L_\lambda(w) = \sum_{(x,y) \in D_{\text{tr}}} \ell(g_w(x), y) + \Omega_\lambda(w)$

outer objective $E(w, \lambda) = \sum_{(x,y) \in D_{\text{val}}} \ell(g_w(x), y).$

Bilevel programming	Hyperparameter optimization	Meta-learning
Inner variables	Parameters	Parameters of Ground models
Outer variables	Hyperparameters	Parameters of Meta-learner
Inner objective	Training error	Training errors on tasks (Eq. 3)
Outer objective	Validation error	Meta-training error (Eq. 4)

Bilevel Programming for Hyperparameter Optimization and Meta-Learning, Franceschi et al., ***ICML*** 2018

Why is HPO difficult?

- Traditional optimization techniques are usually unsuitable for HPO
  - usually employed to optimize expensive to evaluate functions
  - objective function is non-convex and has multiple local optima
  - limited information of the gradient i.e. black-box function
  - mixture of variable types
    - conditionality
- Limited time and resources
  - credit allocation

Why is HPO difficult?

- Optimization target lacking smoothness makes derivative-free methods perform poorly (Sparks et al. 2016)
- ML models or Solvers include various types of hyperparameters
  - e.g. learning rate, layer normalization, number of operations
- HPO techniques sometimes use data sampling to approximate values of the objective function
  - function evaluation time is often ignored in black-box optimization algorithms

How to categorize HPO algorithm?

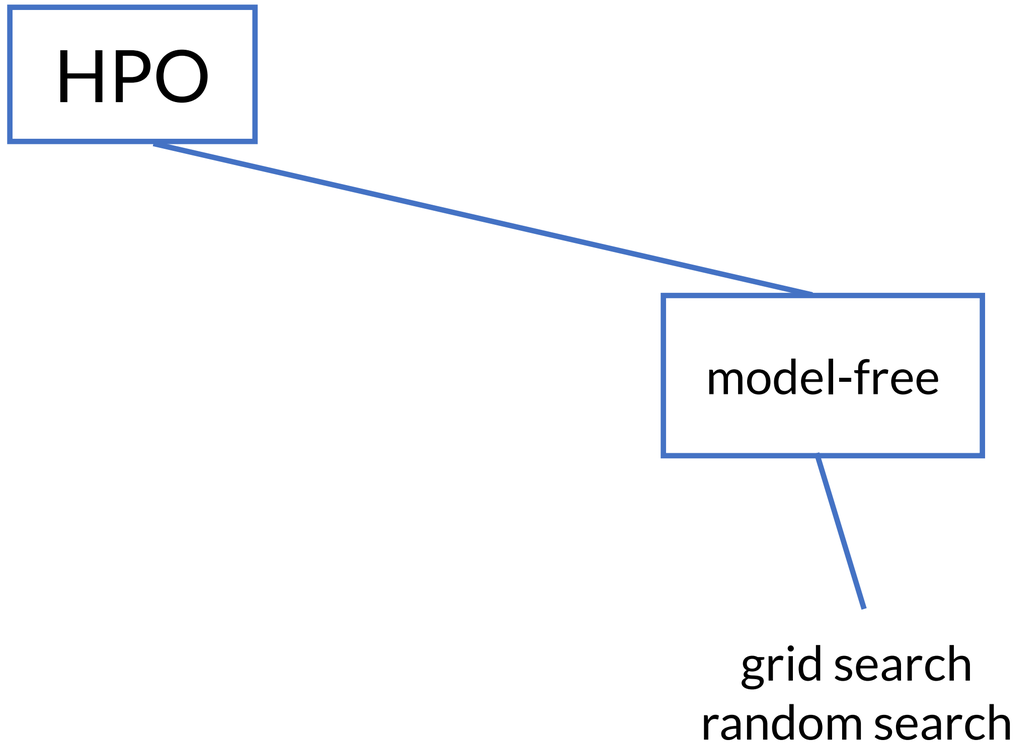
- Estimator with its Objective function
- Search space
- Search strategy
- Performance estimation strategy

How to categorize HPO algorithm?

- Estimator with its Objective function
- Search space
- Search strategy
- Performance estimation strategy

HPO Algorithms Taxonomy

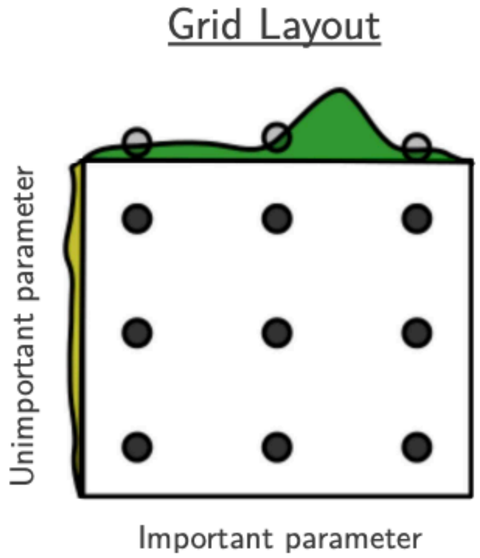
- Model-free methods
  - trial & error
  - grid & random search



On Hyperparameter Optimization of Machine Learning Algorithms: Theory & Practice, Yang & Shami, ***Neurocomputing*** 2020

Grid & Random Search

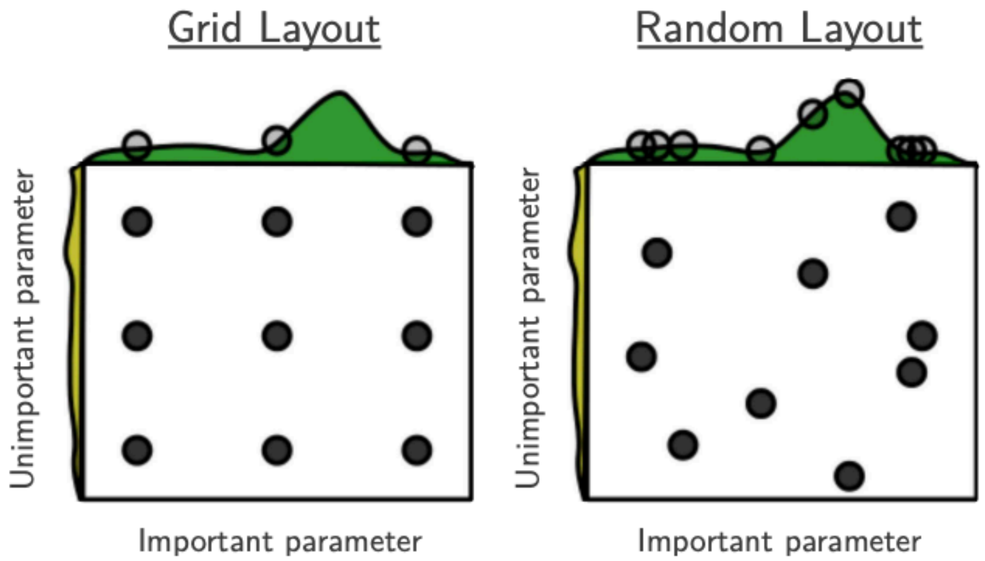
- ***Grid search*** is one of the most commonly used methods
  - it is easy to implement and parallelize
  - curse of dimension $O(n^k)$ → k parameters, n values



Random Search for Hyperparameter Optimization, Bergstra & Bengio, ***JMLR*** 2012

Grid & Random Search

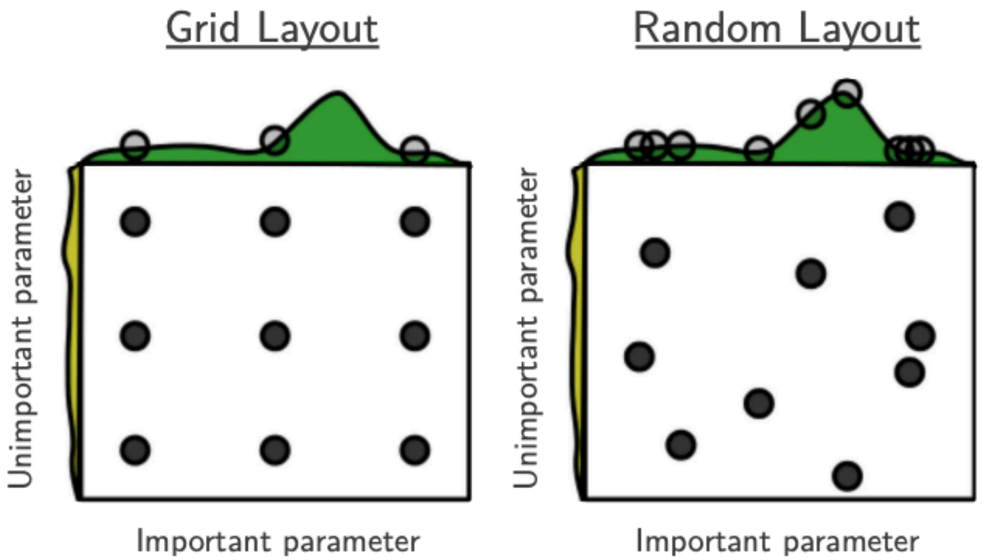
- ***Grid search*** is one of the most commonly used methods
  - it is easy to implement and parallelize
  - curse of dimension $O(n^k)$ → k parameters, n values
- ***Random search*** is similar to grid search, but it is able to explore a larger space

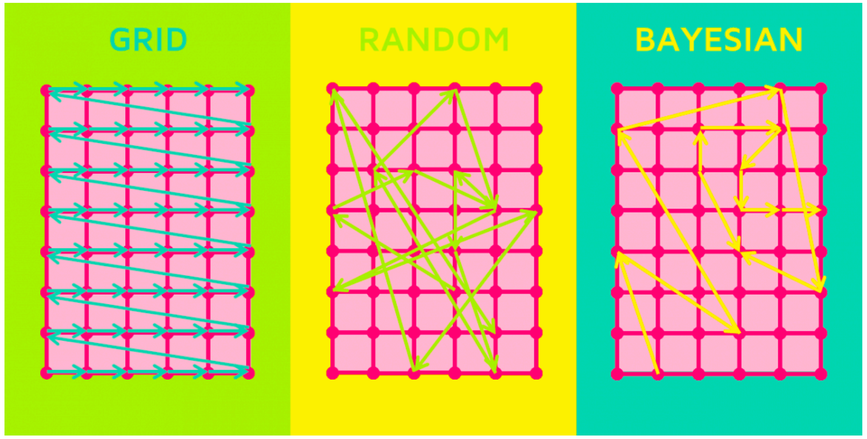


Random Search for Hyperparameter Optimization, Bergstra & Bengio, ***JMLR*** 2012

Grid & Random Search

- ***Grid search*** is one of the most commonly used methods
  - it is easy to implement and parallelize
  - curse of dimension $O(n^k)$ → k parameters, n values
- ***Random search*** is similar to grid search, but it is able to explore a larger space
- Both methods have limitation since every evaluation in their iterations is independent of previous trials

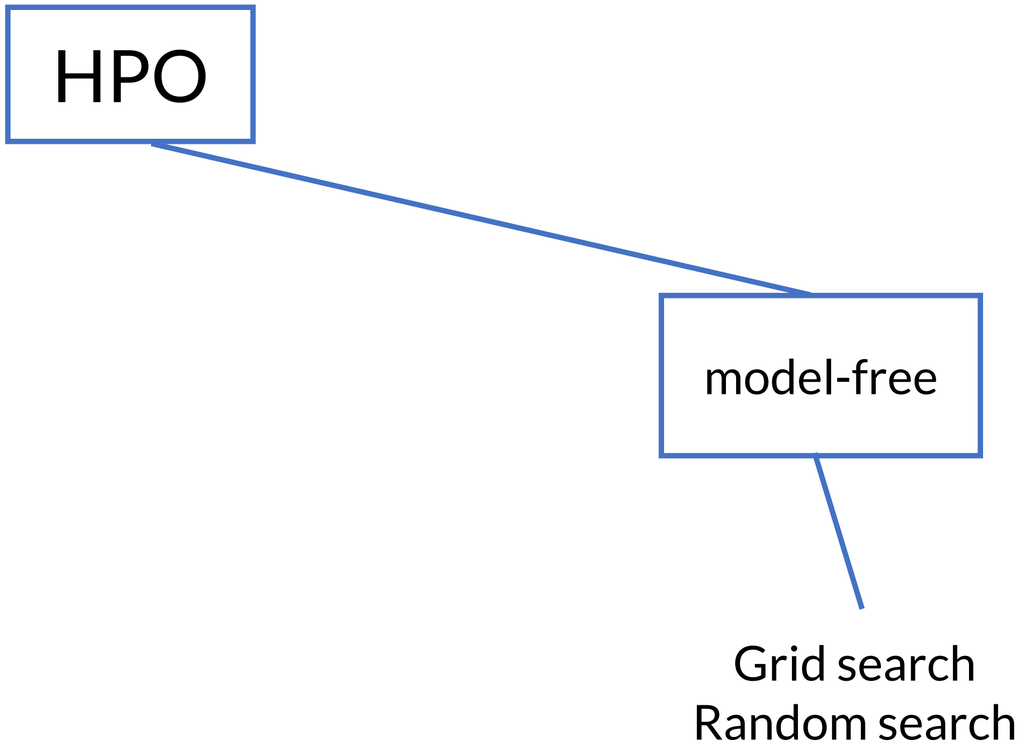




Random Search for Hyperparameter Optimization, Bergstra & Bengio, ***JMLR*** 2012

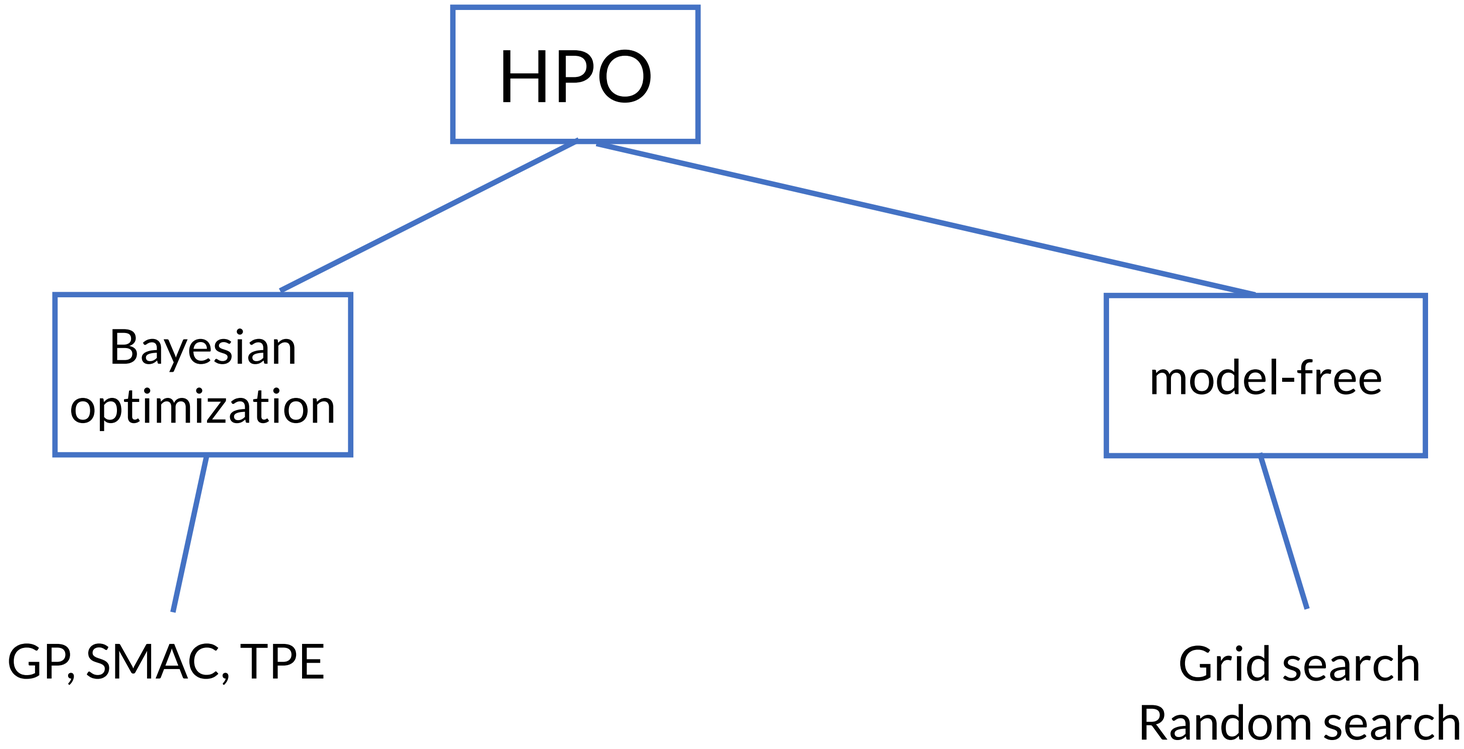
HPO Algorithms Taxonomy

- Model-free methods
  - trial & error
  - grid & random search
- Gradient-based optimization



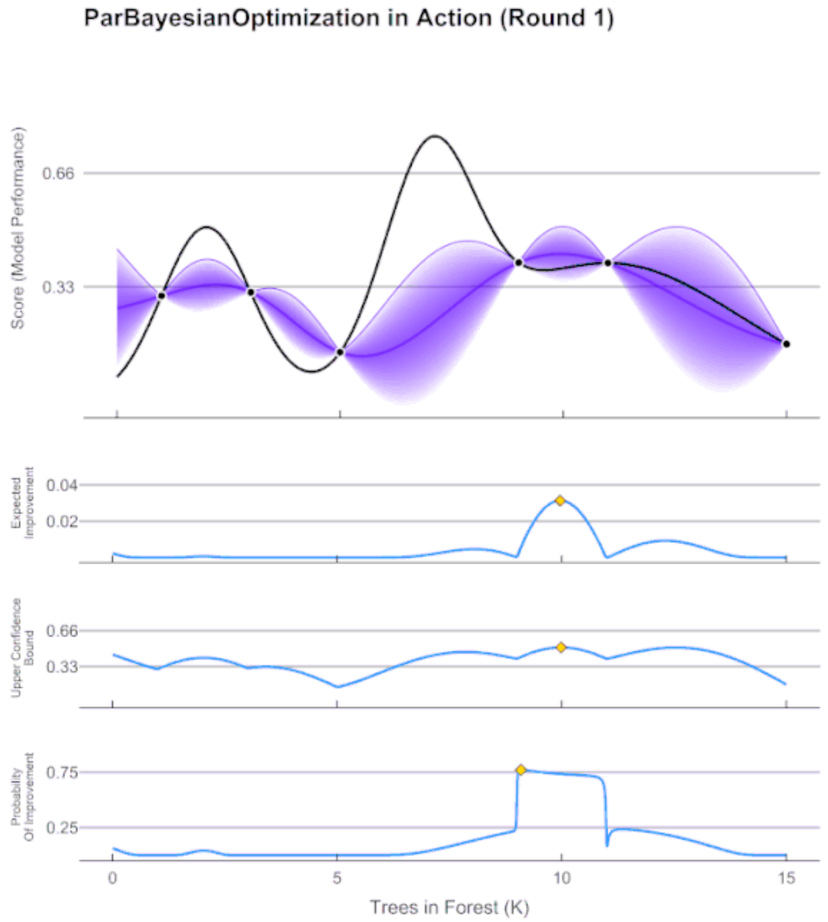
HPO Algorithms Taxonomy

- Model-free methods
  - trial & error
  - grid & random search
- Gradient-based optimization
- Bayesian optimization



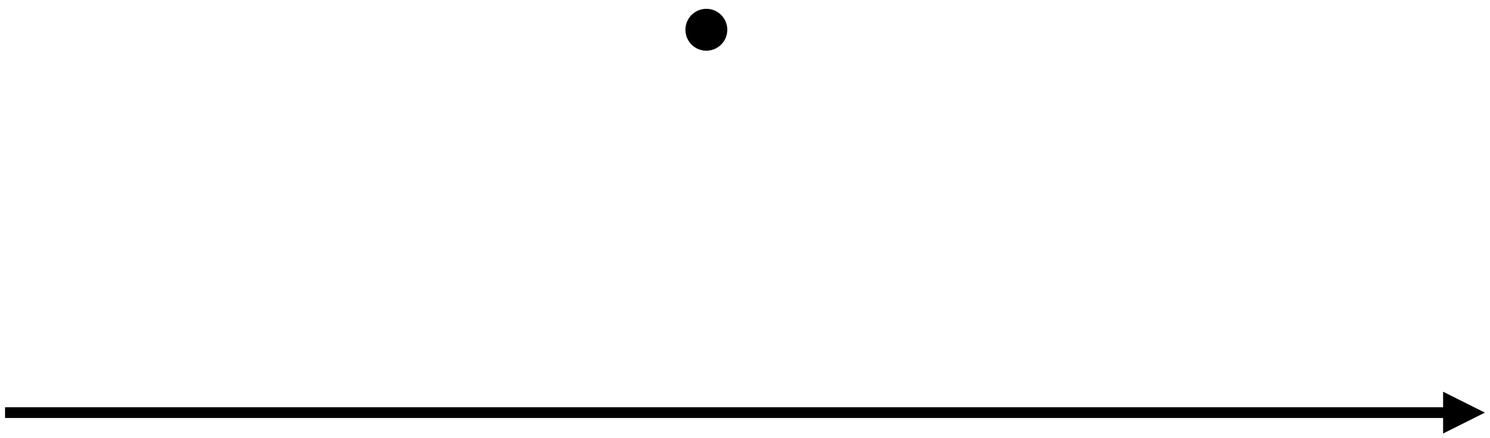
Bayesian Optimization

- BO is an iterative algorithm that is popularly used in HPO problems
  - BO determines the future evaluation points based on the previous results
- ***Surrogate model***
  - fit the observed points into the objective
- ***Acquisition function***
  - determines the usage of points by balancing between exploration and exploitation



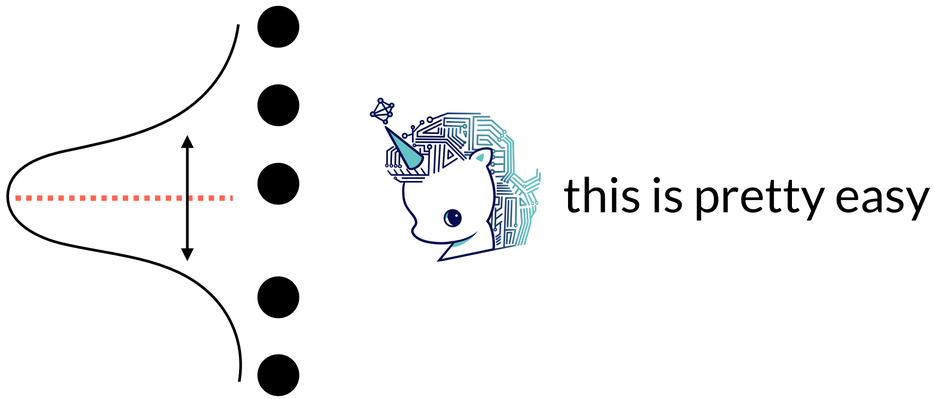
Functional Uncertainty

- Distribution over the space of trajectories or functions



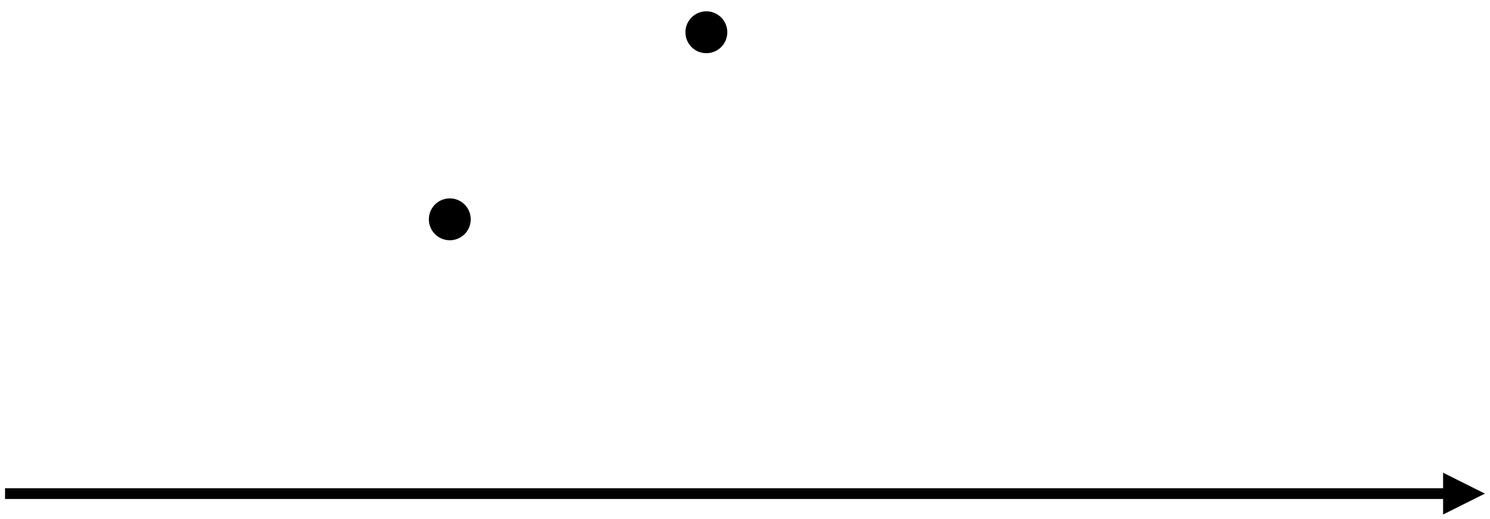
Functional Uncertainty

- Distribution over the space of trajectories or functions



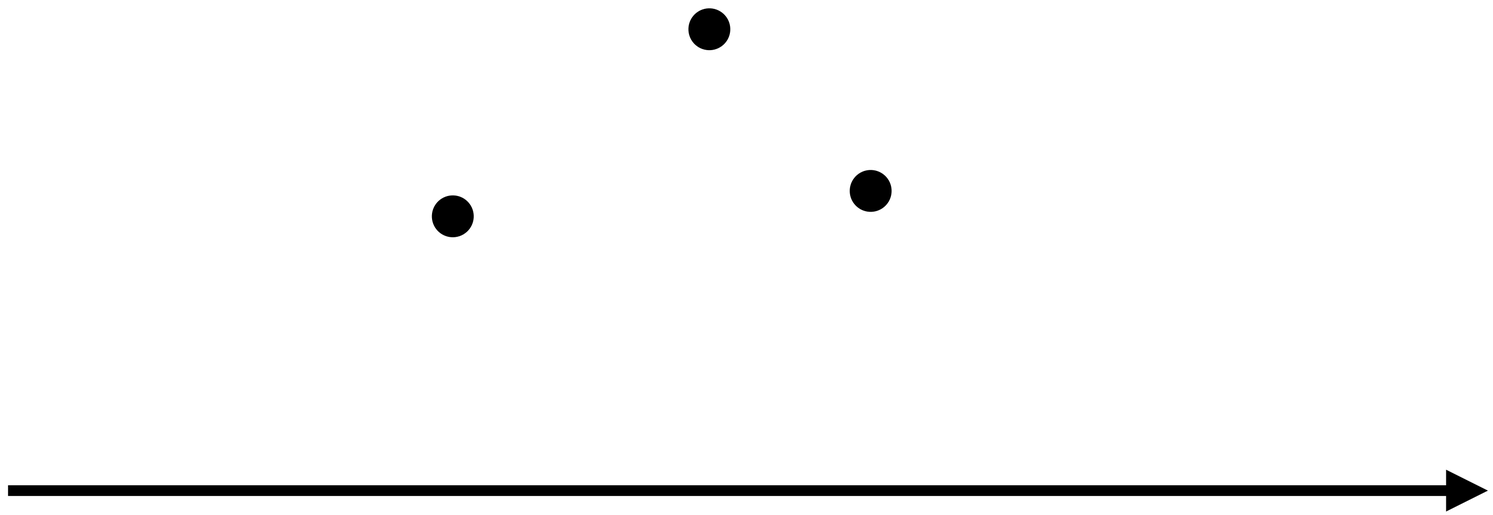
Functional Uncertainty

- Distribution over the space of trajectories or functions



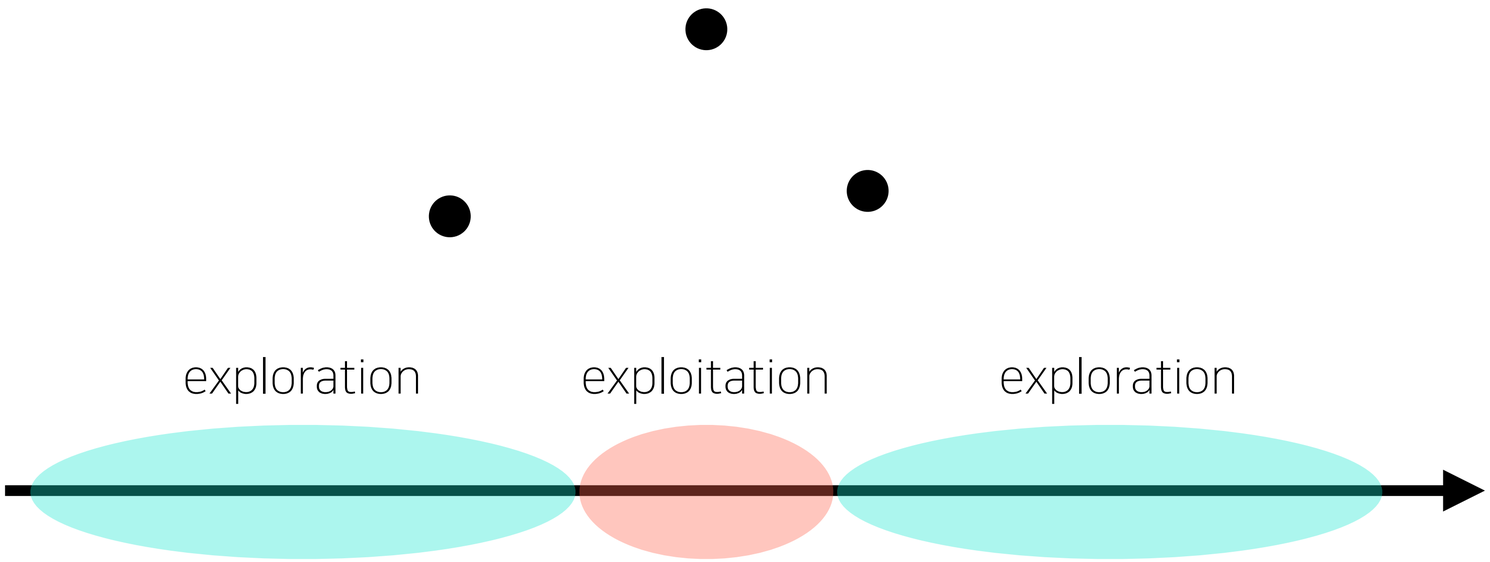
Functional Uncertainty

- Distribution over the space of trajectories or functions



Functional Uncertainty

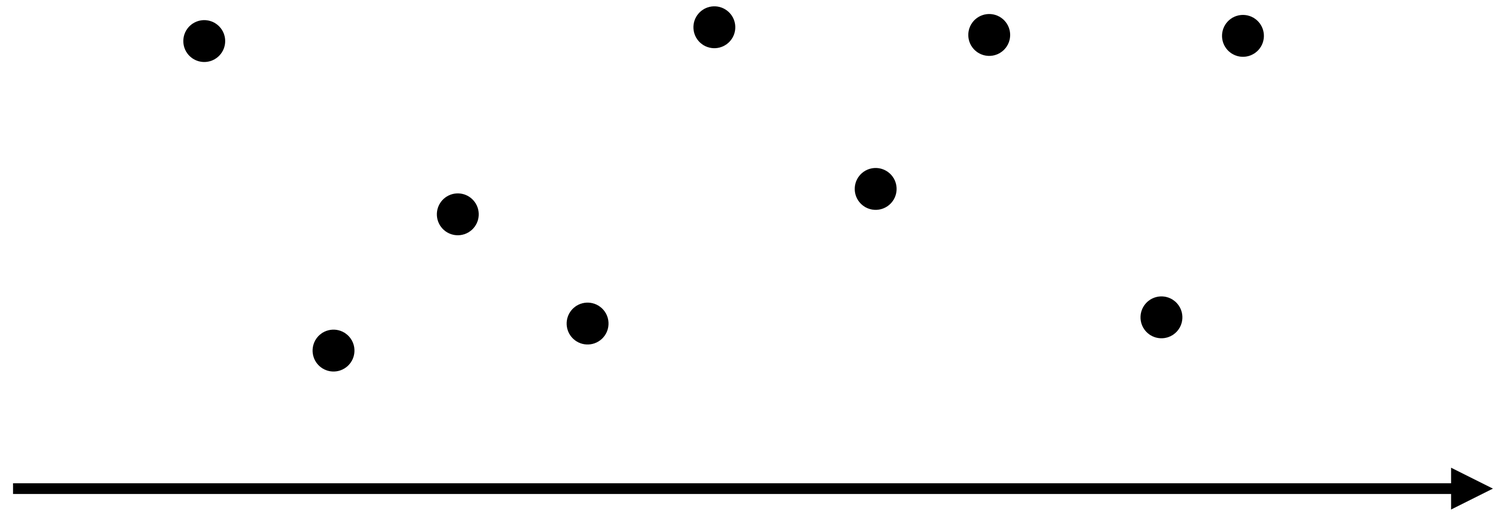
- Distribution over the space of trajectories or functions



Functional Uncertainty

- Distribution over the space of trajectories or functions

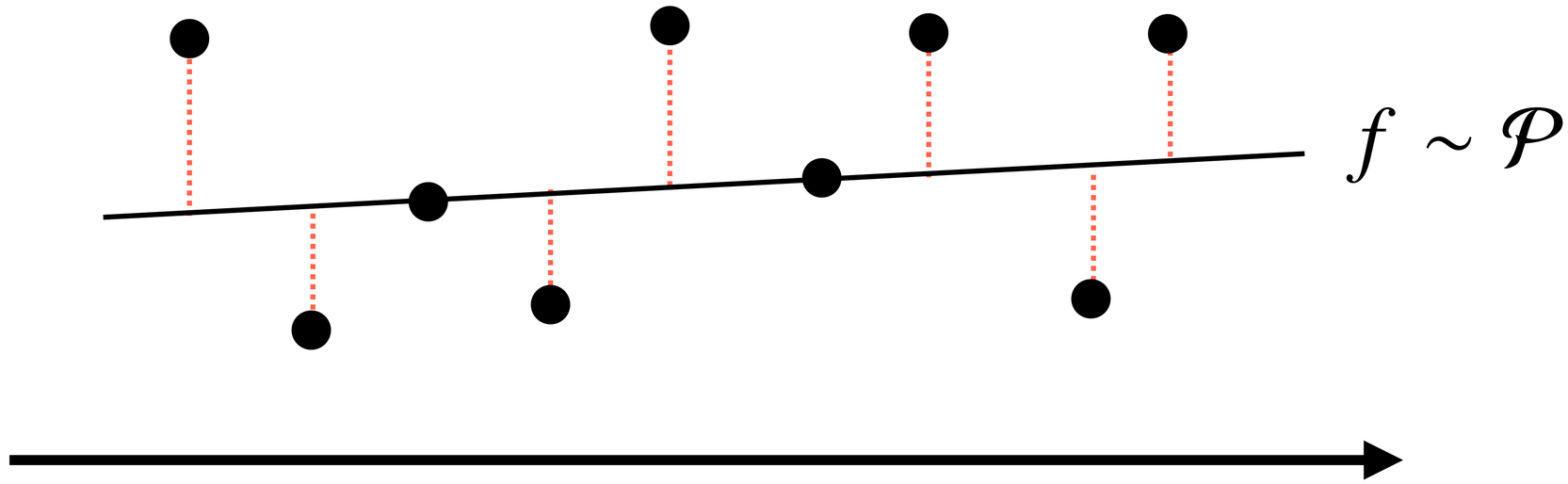
can you infer the shape of the function through this data?



Functional Uncertainty

- Distribution over the space of trajectories or functions

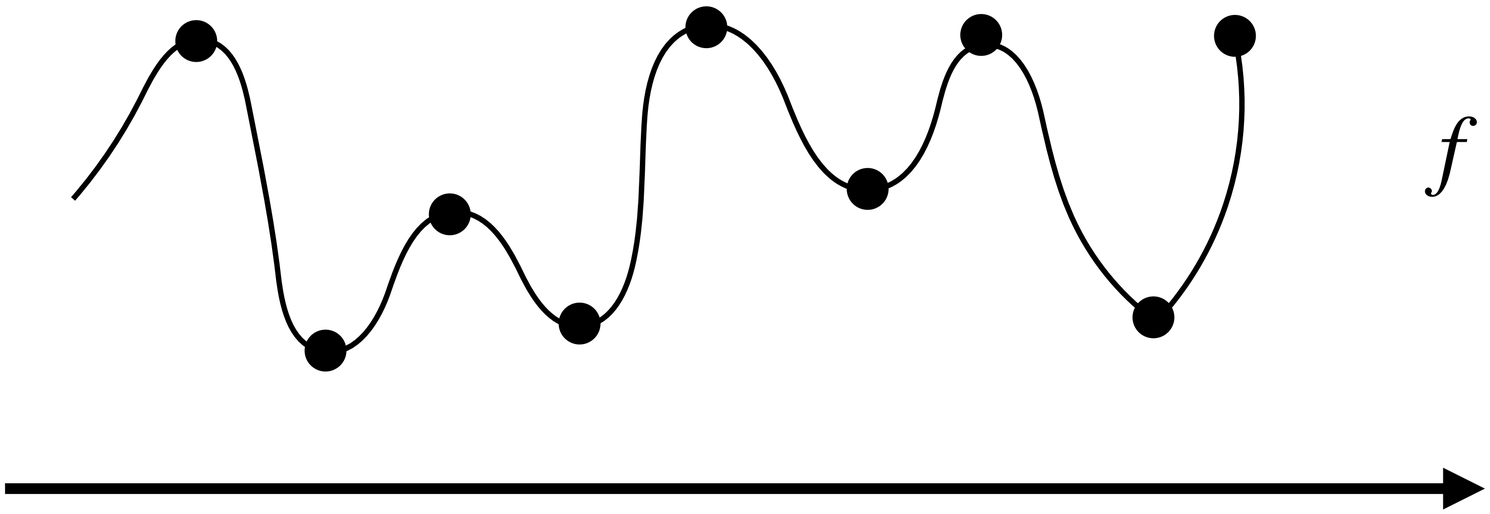
can you infer the shape of the function through this data?



Functional Uncertainty

- Distribution over the space of trajectories or functions

can you infer the shape of the function through this data?



$$f \sim \mathcal{P}'$$

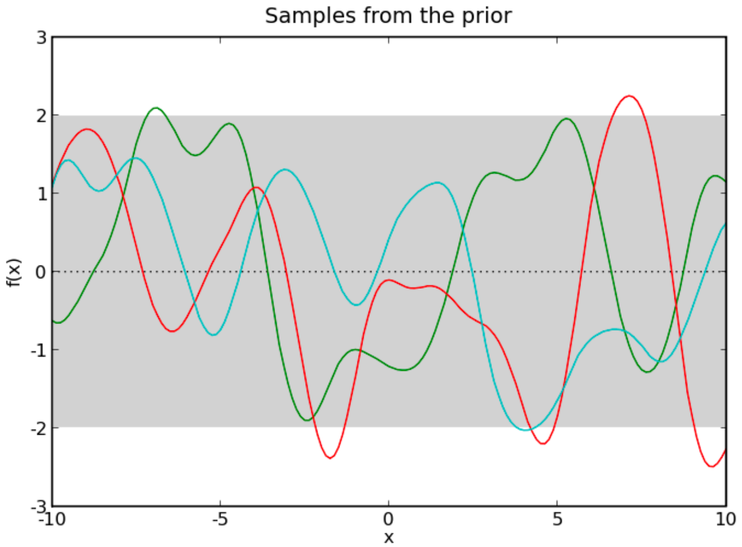
Gaussian Process

- We can assume that the function is drawn from a Gaussian Distribution
  - formally, $P_{x_{1:n}} = \mathcal{N}(\mu, \Sigma)$

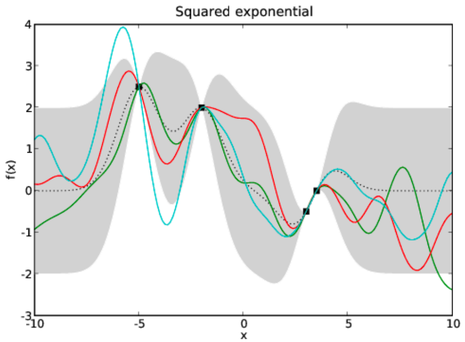
mean vector

covariance matrix

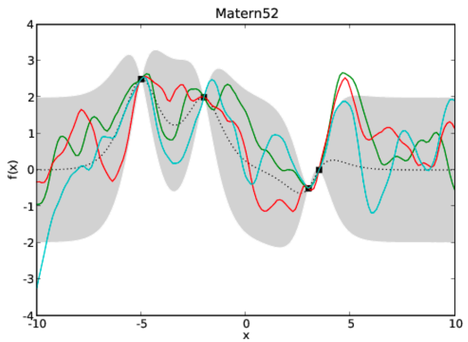
this is not data-driven!



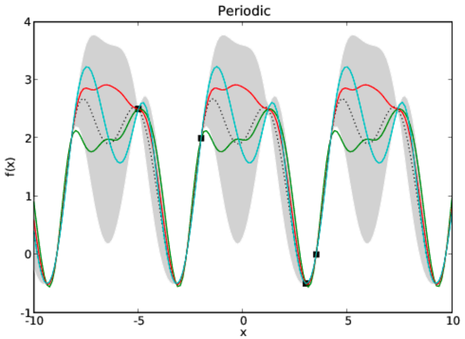
pythonhostd.org



squared exponential



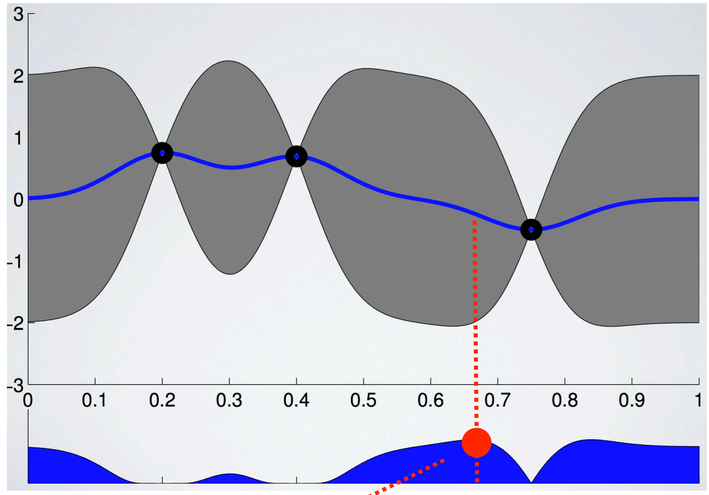
Matern



Periodic

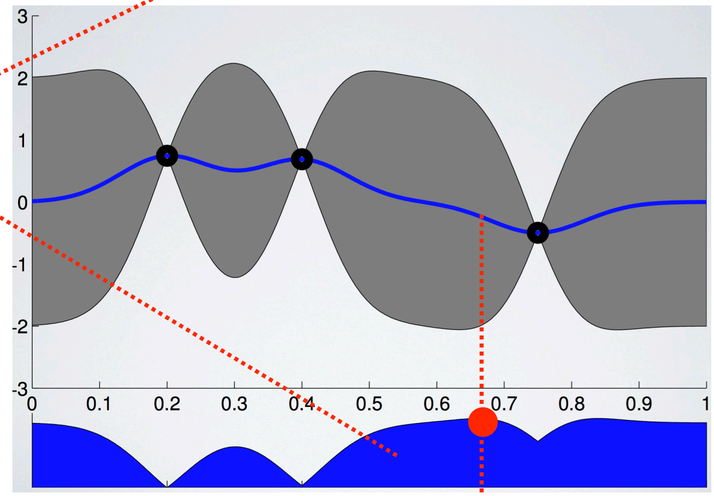
Acquisition Function

- Balance between exploration & exploitation
  - Probability Improvement
  - Expected Improvement
  - Upper Confidence Bound (UCB)
  - Entropy Search
  - Thompson Sampling



EI (expected improvement)

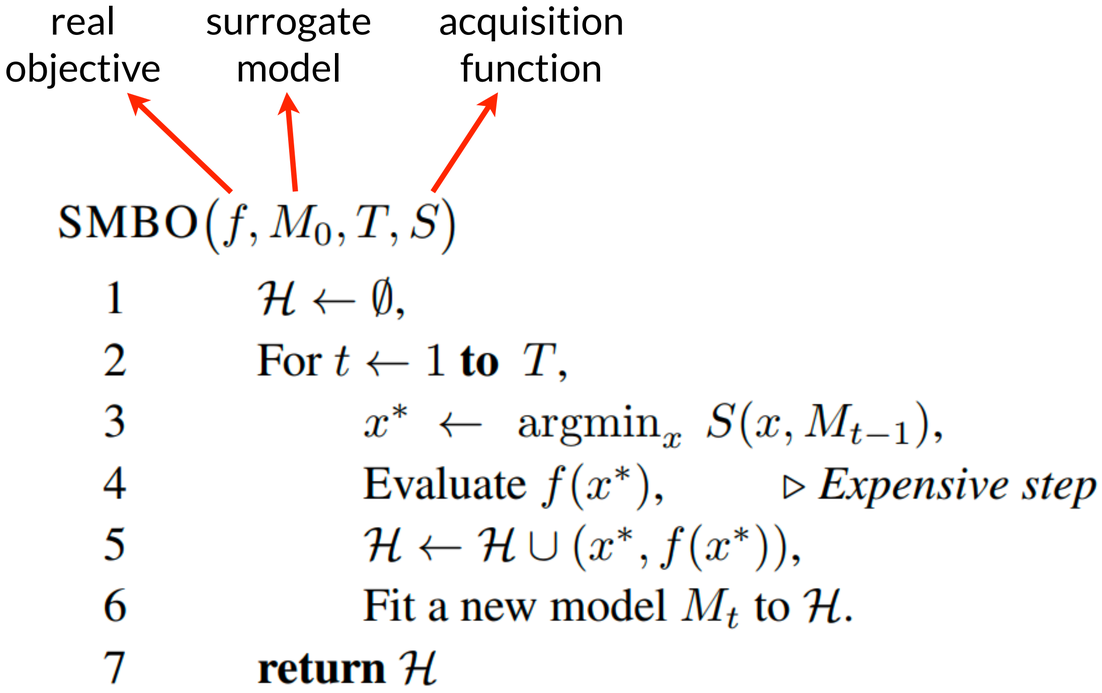
acquisition function



UCB (upper confidence bound)

Basic procedure of BO

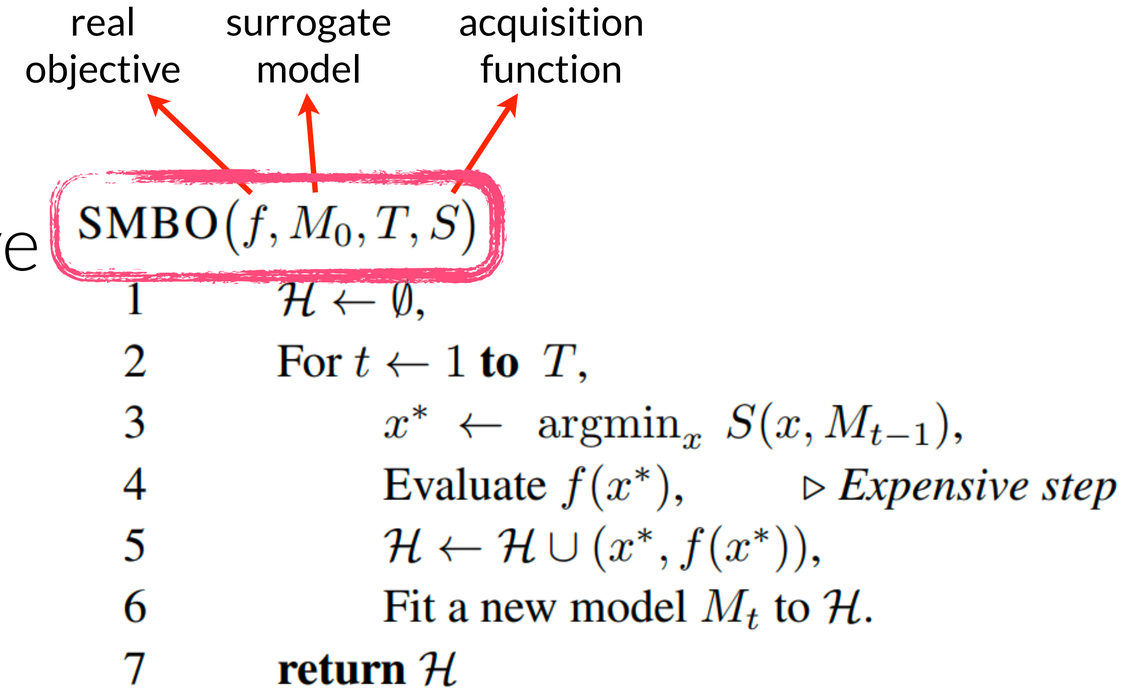
- **S**equential **M**odel **B**ased **O**ptimization



Sequential Model-Based Optimization for General Algorithm Configuration, Hutter et al., 2011

Basic procedure of BO

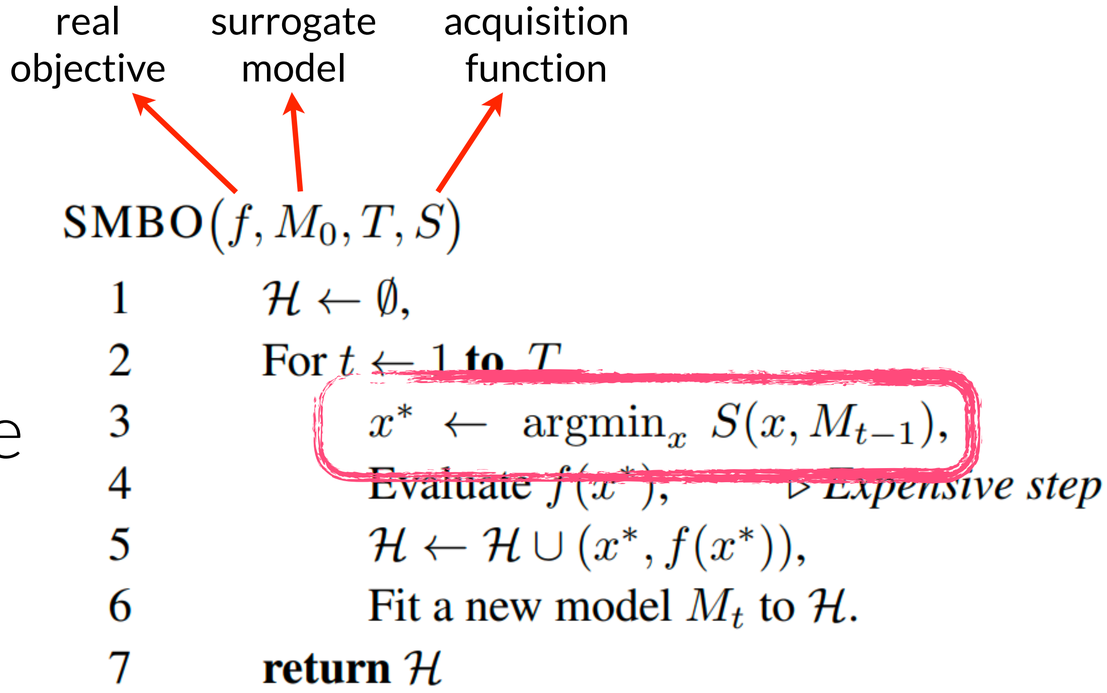
- Sequential Model Based Optimization
- Build a **surrogate model** of the objective functions → GP, Random Forest, TPE



Sequential Model-Based Optimization for General Algorithm Configuration, Hutter et al., 2011

Basic procedure of BO

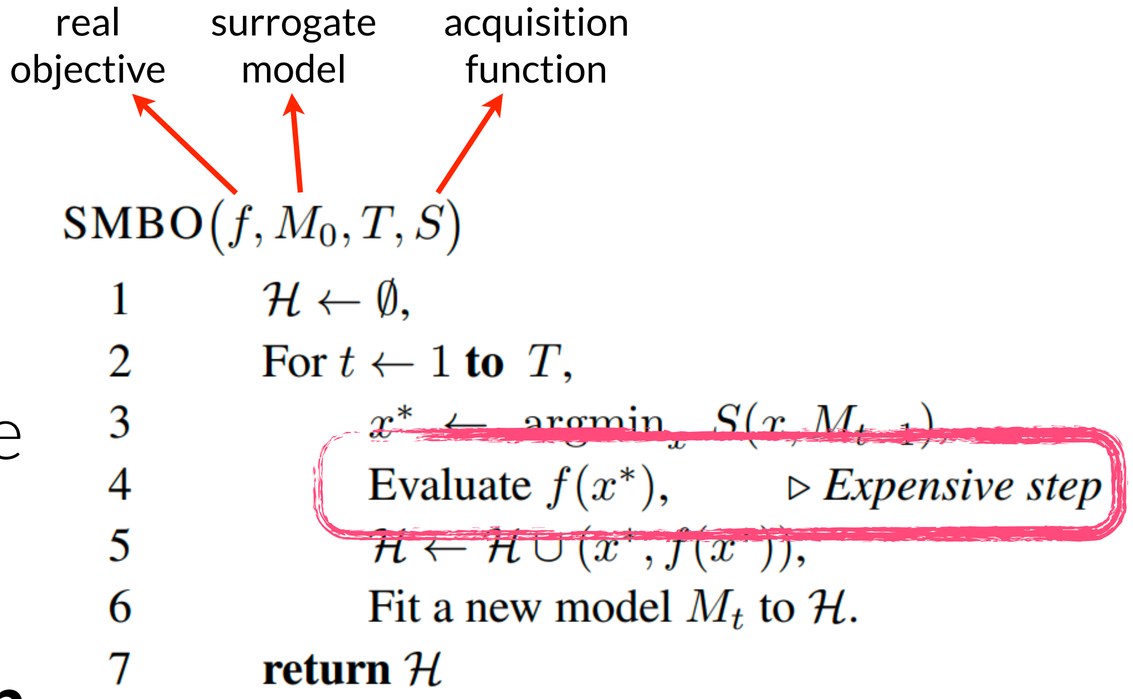
- Sequential Model Based Optimization
- Build a surrogate model of the objective functions → GP, Random Forest, TPE
- Detect the optimal configurations on the surrogate model → **optimization**



Sequential Model-Based Optimization for General Algorithm Configuration, Hutter et al., 2011

Basic procedure of BO

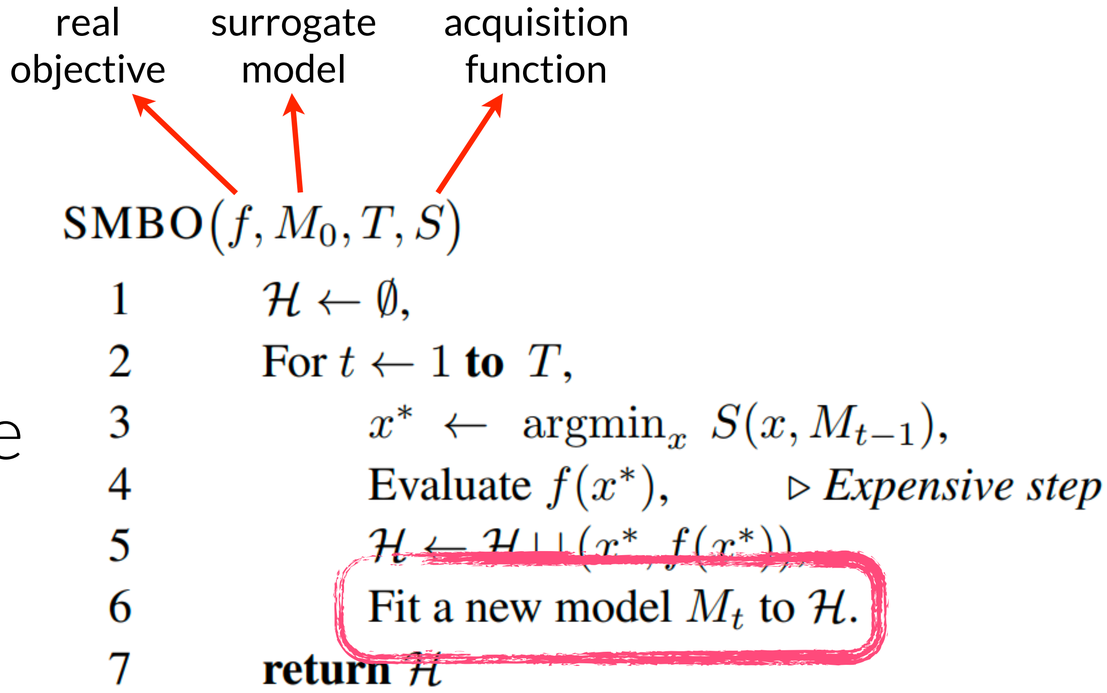
- Sequential Model Based Optimization
- Build a surrogate model of the objective functions → GP, Random Forest, TPE
- Detect the optimal configurations on the surrogate model → optimization
- Apply these hyperparameter values to the real objective function → **evaluation**



Sequential Model-Based Optimization for General Algorithm Configuration, Hutter et al., 2011

Basic procedure of BO

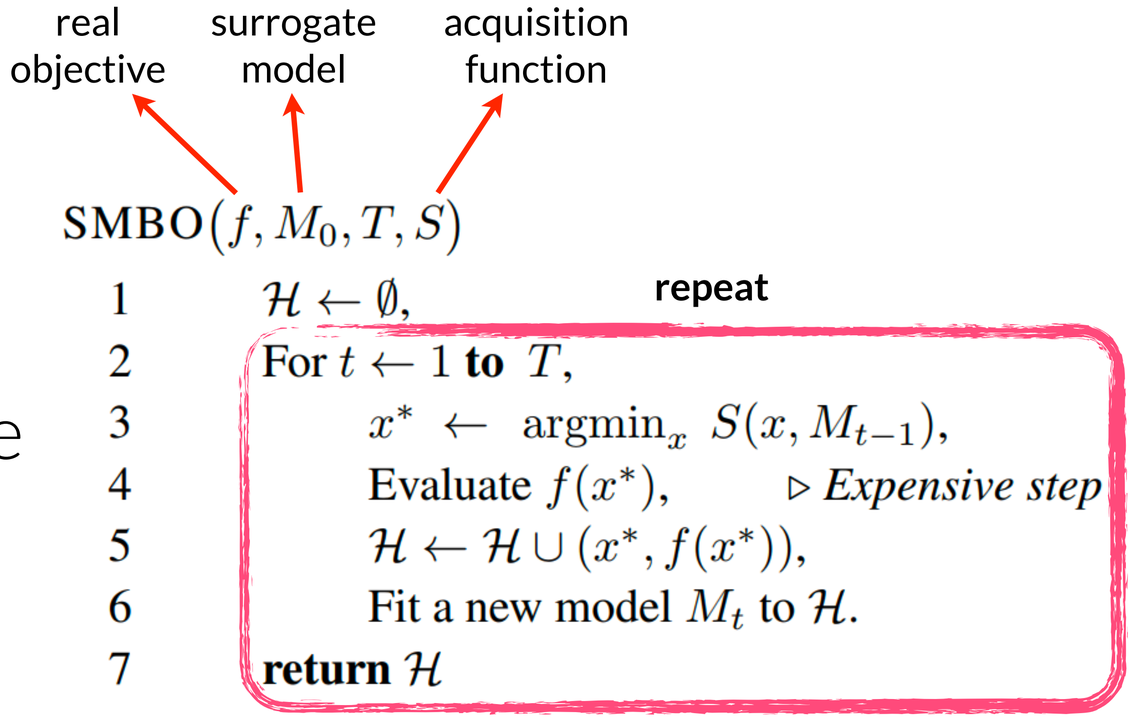
- Sequential Model Based Optimization
- Build a surrogate model of the objective functions → GP, Random Forest, TPE
- Detect the optimal configurations on the surrogate model → optimization
- Apply these hyperparameter values to the real objective function → evaluation
- Update the surrogate model with new results → **learning**



Sequential Model-Based Optimization for General Algorithm Configuration, Hutter et al., 2011

Basic procedure of BO

- Sequential Model Based Optimization
- Build a surrogate model of the objective functions → GP, Random Forest, TPE
- Detect the optimal configurations on the surrogate model → optimization
- Apply these hyperparameter values to the real objective function → evaluation
- Update the surrogate model with new results → learning



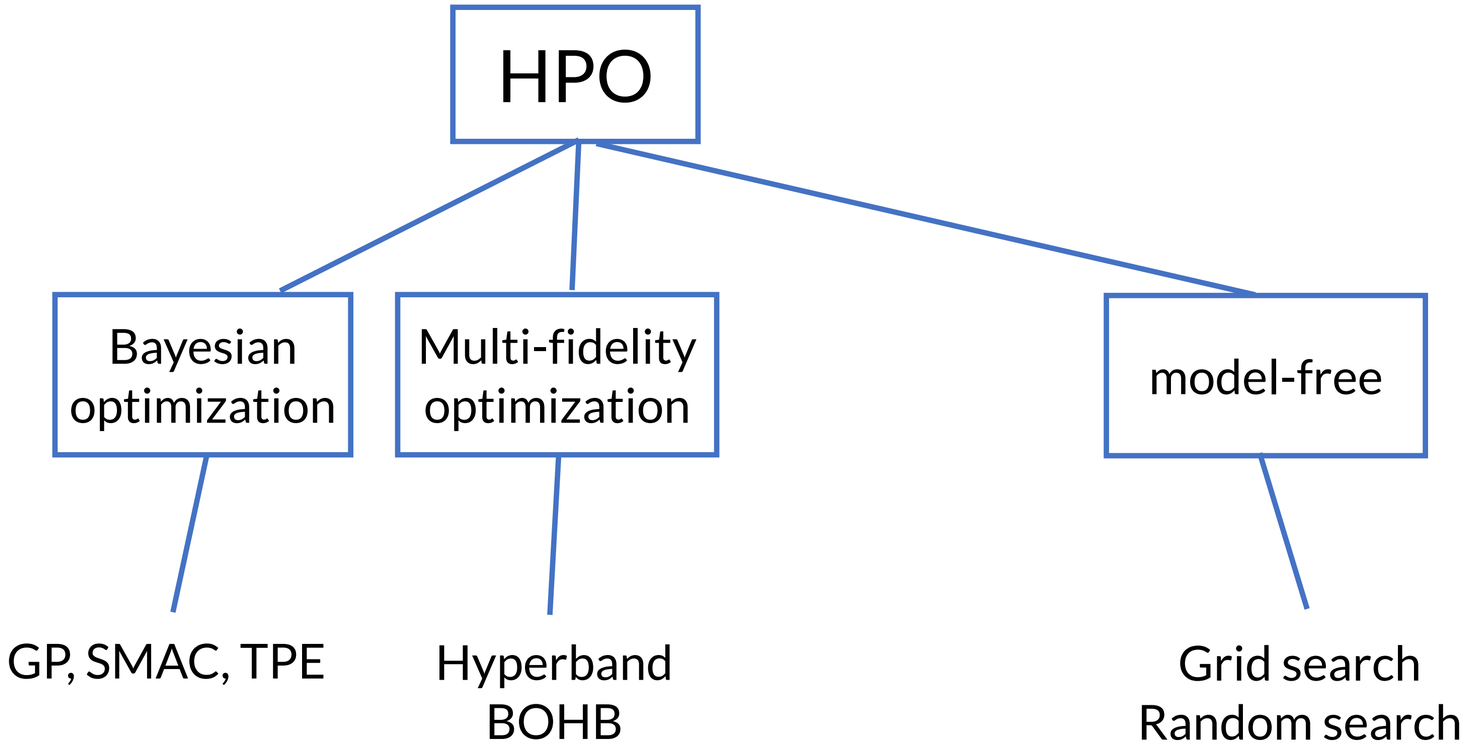
Sequential Model-Based Optimization for General Algorithm Configuration, Hutter et al., 2011

Pros and Cons of BO Algorithms

- **Pros**
  - more efficient than GS and RS
  - can detect near-optimal configuration combinations within a few iterations
  - require less computational resources
- **Cons**
  - since BO models are executed based on the preciously-tested values, they are difficult to parallelize
  - another configurations for BO procedures

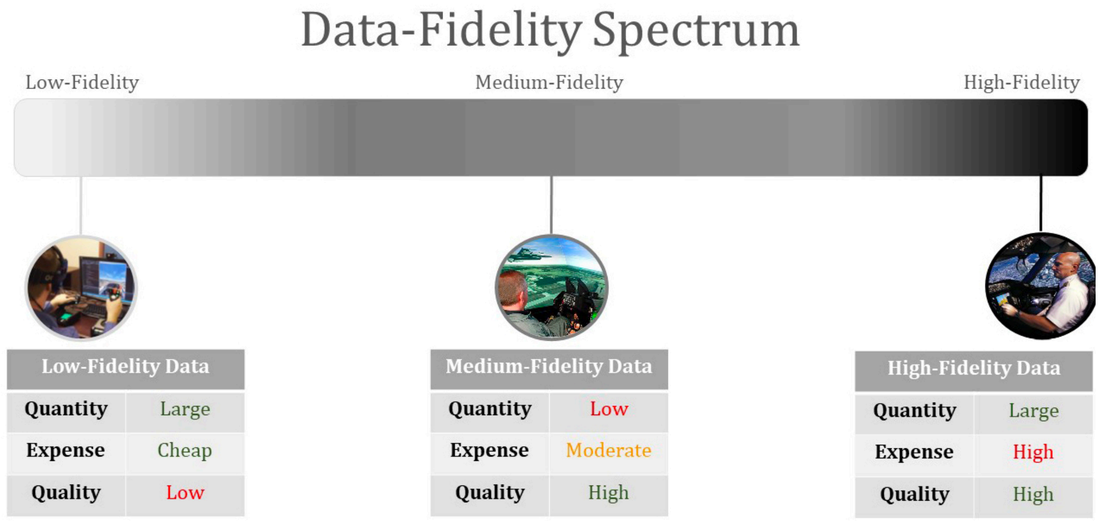
HPO Algorithms Taxonomy

- Model-free methods
  - trial & error
  - grid & random search
- Gradient-based optimization
- Bayesian optimization
- Multi-fidelity optimization



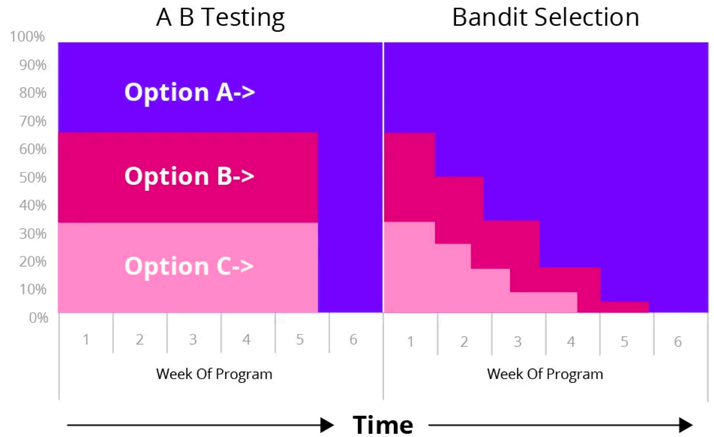
Multi-Fidelity Optimization

- ***Low-fidelity***
  - low cost, poor generalization
- ***High-fidelity***
  - higher cost, better generalization
- One can use a subset of the original dataset or a subset of the features → multi-fidelity
  - Poorly performing configurations are discarded after each round of hyperparameter evaluation on generated subsets
  - Well-performing configurations are evaluated on the entire train data



Schlicht (2017)

Successive Halving Algorithm



- Idea of SHA
  - uniformly allocate the budget to a set of arms for iterations
  - evaluate and throw out the half with the double budgets
- The main concern of SHA is how to allocate the budget $b = B/n$ and how to balance the trade-off between the evaluation and budget

Successive Halving Algorithm

input: Budget B, n arms where $\ell_{i,k}$ denotes the kth loss from the ith arm

Initialize: $S_0 = [n]$.

For $k = 0, 1, \ldots, \lceil \log_2(n) \rceil - 1$

Pull each arm in S_k for $r_k = \lfloor \frac{B}{|S_k| \lceil \log_2(n) \rceil} \rfloor$ additional times and set $R_k = \sum_{j=0}^{k} r_j$.

Let σ_k be a bijection on S_k such that $\ell_{\sigma_k(1),R_k} \leq \ell_{\sigma_k(2),R_k} \leq \cdots \leq \ell_{\sigma_k(|S_k|),R_k}$

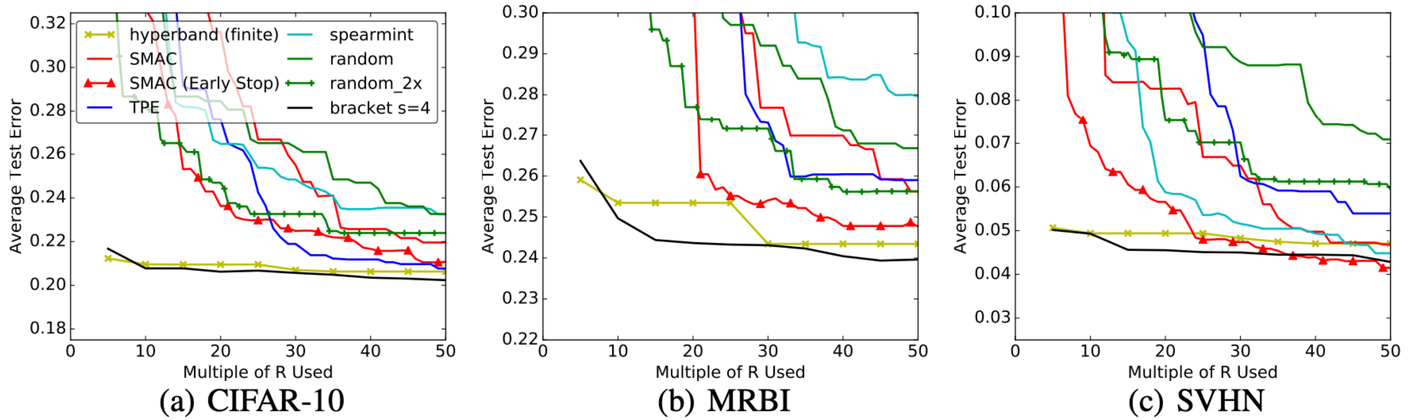
$S_{k+1} = \left\{ i \in S_k : \ell_{\sigma_k(i),R_k} \leq \ell_{\sigma_k(\lfloor |S_k|/2 \rfloor),R_k} \right\}$.

output : Singleton element of $S_{\lceil \log_2(n) \rceil}$

Non-stochastic Best Arm Identification and Hyperparameter Optimization, Jamieson & Talwalkar, ***AISTATS*** 2015

Hyperband

- Idea of Hyperband
  - adaptive resource allocation
  - early stopping
- Solve the dilemma of SHA by dynamically choosing a reasonable number of configurations
  - use random search to get n number of configurations



(a) CIFAR-10 (b) MRBI (c) SVHN

Successive Halving Algorithm

input: Budget B, n arms where $\ell_{i,k}$ denotes the kth loss from

Algorithm 1: HYPERBAND algorithm for hyperparameter optimization.

input : R, η (default $\eta = 3$)
initialization : $s_{\max} = \lfloor \log_\eta(R) \rfloor$, $B = (s_{\max} + 1)R$
for $s \in \{s_{\max}, s_{\max} - 1, \ldots, 0\}$ **do**
  $n = \lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \rceil$, $r = R\eta^{-s}$
  // begin SUCCESSIVEHALVING with (n, r) inner loop
  T =get_hyperparameter_configuration(n)
  for $i \in \{0, \ldots, s\}$ **do**
    $n_i = \lfloor n\eta^{-i} \rfloor$
    $r_i = r\eta^i$
    $L = \{\text{run\_then\_return\_val\_loss}(t, r_i) : t \in T\}$
    T =top_k($T, L, \lfloor n_i/\eta \rfloor$)
  end
end
return *Configuration with the smallest intermediate loss seen so far.*

Hyperband: A novel bandit-based approach to hyperparameter optimization, Li et al, ***JMLR*** 2018

BOHB: BO + Hyperband

- State-of-the art HPO technique that combines BO and Hyperband
- Hyperband uses a random search to get the configuration space
  - BOHB replace RS with BO!
- Easy to parallelizable
- Require the evaluations on subsets with small budgets

Algorithm 2: Pseudocode for sampling in BOHB

input : observations D, fraction of random runs ρ, percentile q, number of samples N_s, minimum number of points N_{min} to build a model, and bandwidth factor b_w

output : next configuration to evaluate

if *rand()* $< \rho$ **then return** random configuration

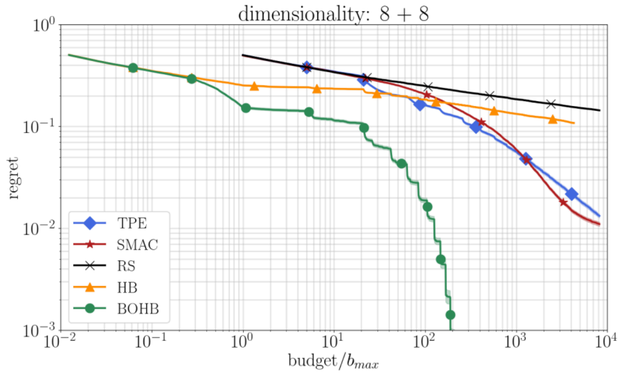
$b = \arg\max \{D_b : |D_b| \geq N_{min} + 2\}$

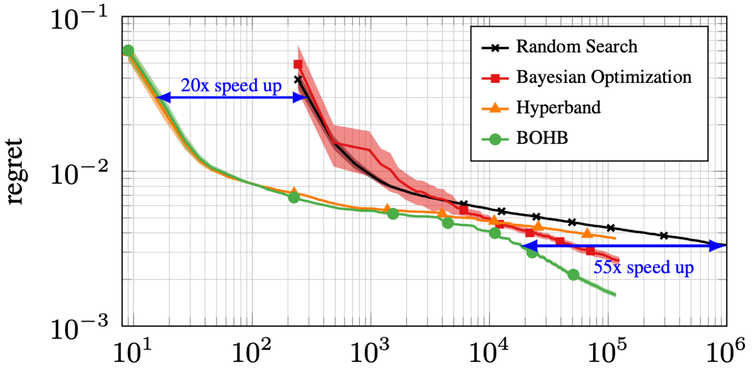
if $b = \emptyset$ **then return** random configuration

fit KDEs according to Eqs. (2) and (3)

draw N_s samples according to $l'(\boldsymbol{x})$ (see text)

return sample with highest ratio $l(\boldsymbol{x})/g(\boldsymbol{x})$





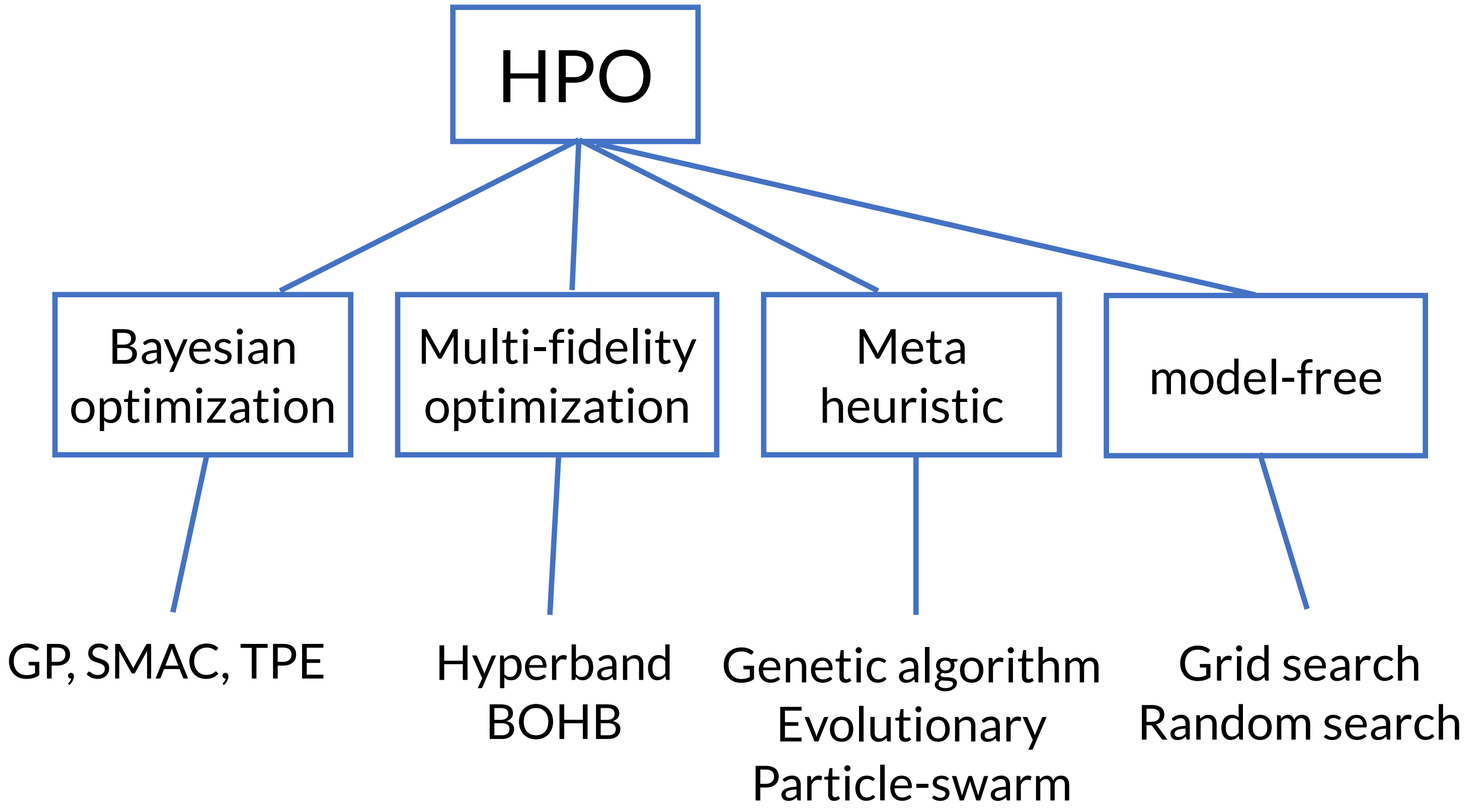
BOHB: Robust and Efficient Hyperparameter Optimization at Scale, Falkner et al, ***ICML*** 2018

Pros and Cons of Multi-fidelity optimization

- **Pros**
  - **parallelizable!**
  - fast convergence
- **Cons**
  - requires budget and resource allocation
  - can be poor at final performance

HPO Algorithms Taxonomy

- Model-free methods
  - trial & error
  - grid & random search
- Gradient-based optimization
- Bayesian optimization
- Multi-fidelity optimization
- Meta-heuristic algorithms



Population-Based Approach

- Population based approaches are a major type of meta-heuristic method
  - start by creating and updating a population as each generation
  - each individual in every generation is evaluated until the global optimum is identified
- The main difference between population-based approaches are the methods used to generate and select populations
  - genetic algorithms (GA)
  - particle swarm optimization (PSO)

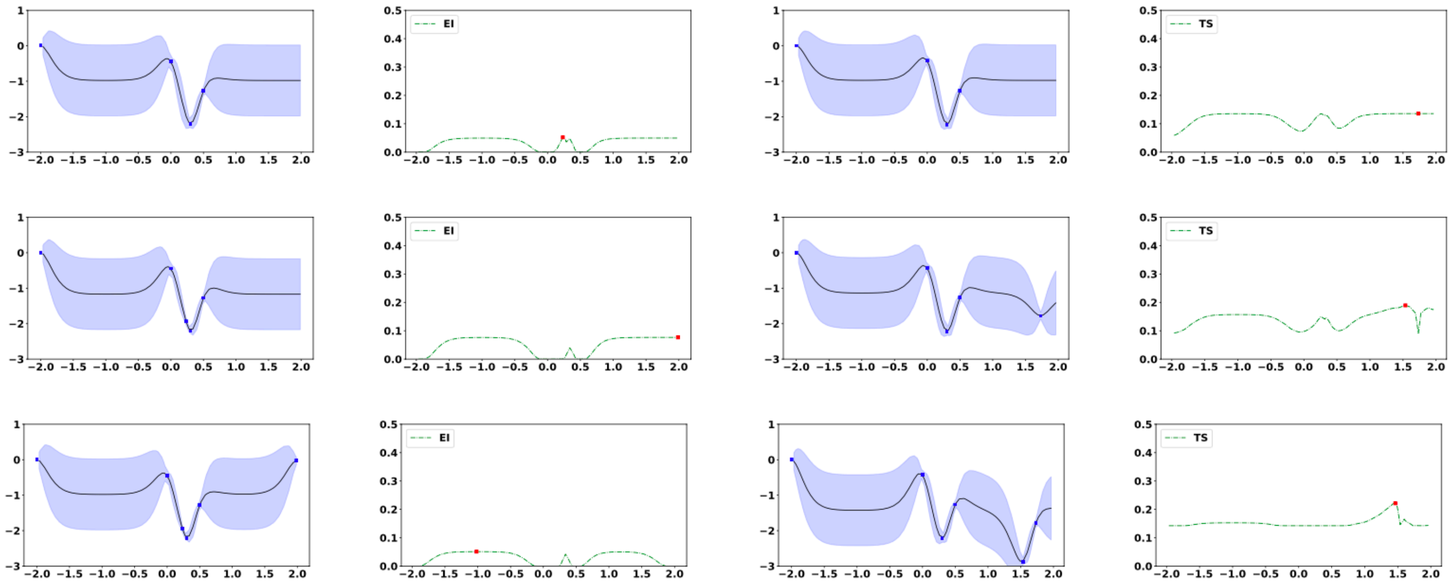
Comparison of HPO algorithms

		Pros	Cons	Time Complexity
model-free	Grid Search	Simple	Inefficient	$O(n^k)$
	Random Search	Easy to implement Parallelizable	Ignore the previous observations	$O(n)$
Bayesian optimization	Gaussian Process	Data efficient	Poor parallelization kernel selection	$O(n^3)$
	SMAC	Efficient with variable types	Poor parallelization Poor performance	$O(n \log n)$
	TPE	Efficient with variable types Efficient with conditional	Poor parallelization	$O(n \log n)$
Multi-fidelity optimization	Hyperband	Parallelizable	Inefficient with conditional Poor final performance	$O(n \log n)$
	BOHB	Efficient with variable types Parallelizable	Require the evaluations on subsets with small budgets	$O(n \log n)$
Meta-heuristic	PSO	Efficient with variable types Parallelizable	Require initialization	$O(n \log n)$

On Hyperparameter Optimization of Machine Learning Algorithms: Theory & Practice, Yang & Shami, ***Neurocomputing*** 2020

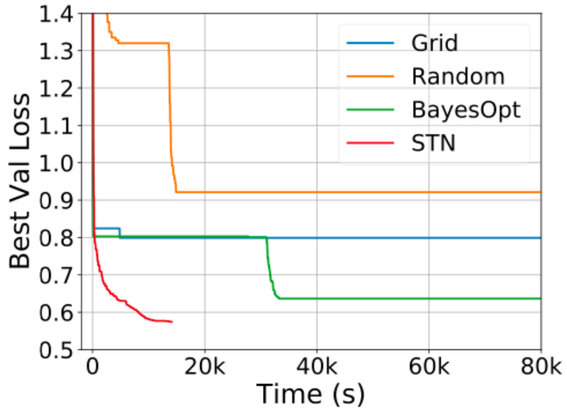
Advances: Two-step look-ahead BO

- Myopic exploration of expected improvement
  - estimate acquisition function via Monte Carlo method

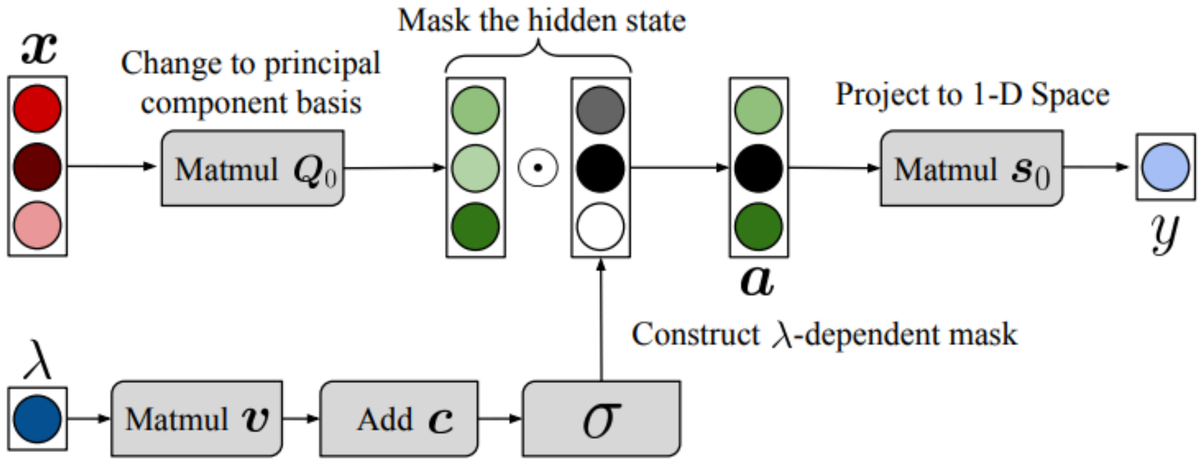


Practical Two-step Look-Ahead Bayesian Optimization, Wu & Frazier., ***NeurIPS*** 2019

Advances: Self-tuning networks



- Hyperparameter scheduling method
  - self-tuning network tunes hyperparameters in the training
  - transforms bilevel optimization into single-level optimization



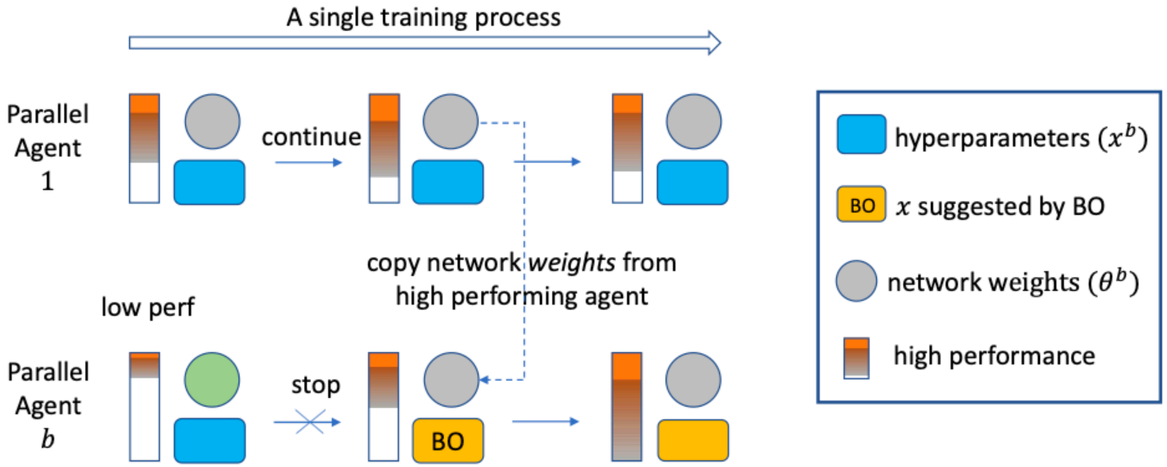
Algorithm 1 STN Training Algorithm

Initialize: Best-response approximation parameters $\boldsymbol{\phi}$, hyperparameters $\boldsymbol{\lambda}$, learning rates $\{\alpha_i\}_{i=1}^3$
while not converged **do**
  for $t = 1, \ldots, T_{train}$ **do**
    $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}|\boldsymbol{\sigma})$
    $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \alpha_1 \frac{\partial}{\partial \boldsymbol{\phi}} f(\boldsymbol{\lambda} + \boldsymbol{\epsilon}, \hat{\mathbf{w}}_{\boldsymbol{\phi}}(\boldsymbol{\lambda} + \boldsymbol{\epsilon}))$
  for $t = 1, \ldots, T_{valid}$ **do**
    $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}|\boldsymbol{\sigma})$
    $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} - \alpha_2 \frac{\partial}{\partial \boldsymbol{\lambda}} \left(F(\boldsymbol{\lambda} + \boldsymbol{\epsilon}, \hat{\mathbf{w}}_{\boldsymbol{\phi}}(\boldsymbol{\lambda} + \boldsymbol{\epsilon})) - \tau \mathbb{H}[p(\boldsymbol{\epsilon}|\boldsymbol{\sigma})]\right)$
    $\boldsymbol{\sigma} \leftarrow \boldsymbol{\sigma} - \alpha_3 \frac{\partial}{\partial \boldsymbol{\sigma}} \left(F(\boldsymbol{\lambda} + \boldsymbol{\epsilon}, \hat{\mathbf{w}}_{\boldsymbol{\phi}}(\boldsymbol{\lambda} + \boldsymbol{\epsilon})) - \tau \mathbb{H}[p(\boldsymbol{\epsilon}|\boldsymbol{\sigma})]\right)$

Bilevel Optimization of Hyperparameters using Structured Best-Response Functions, MacKay et al., ***ICLR*** 2019

Advances: Population based Bandit

- PBT explores configuration space randomly
  - PB2 explores the space based on the probabilistic model (GP-UCB)
  - proves the theoretical guarantees



Algorithm 1: Population-Based Bandit Optimization (PB2)

Initialize: Network weights $\{\theta_0^b\}_{b=1}^B$, hyperparameters $\{x_0^b\}_{b=1}^B$, dataset $D_0 = \emptyset$
(in parallel) for $t = 1, \ldots, T-1$ **do**

1. **Update Models:** $\theta_t^b \leftarrow \text{step}(\theta_{t-1}^b | x_{t-1}^b)$
2. **Evaluate Models:** $y_t^b = F_t(x_t^b) - F_{t-1}(x_{t-1}^b) + \epsilon_t$ for all b
3. **Record Data:** $D_t = D_{t-1} \cup \{(y_t^b, t, x_t^b)\}_{b=1}^B$
4. If $t \mod t_{\text{ready}} = 0$:
   - **Copy weights:** Rank agents, if θ^b is in the bottom $\lambda\%$ then copy weights θ^j from the top $\lambda\%$.
   - **Select hyperparameters:** Fit a GP model to D_t and select hyper-parameters $x_t^b, \forall b \leq B$ by maximizing Eq. (5).

Return the best trained model θ

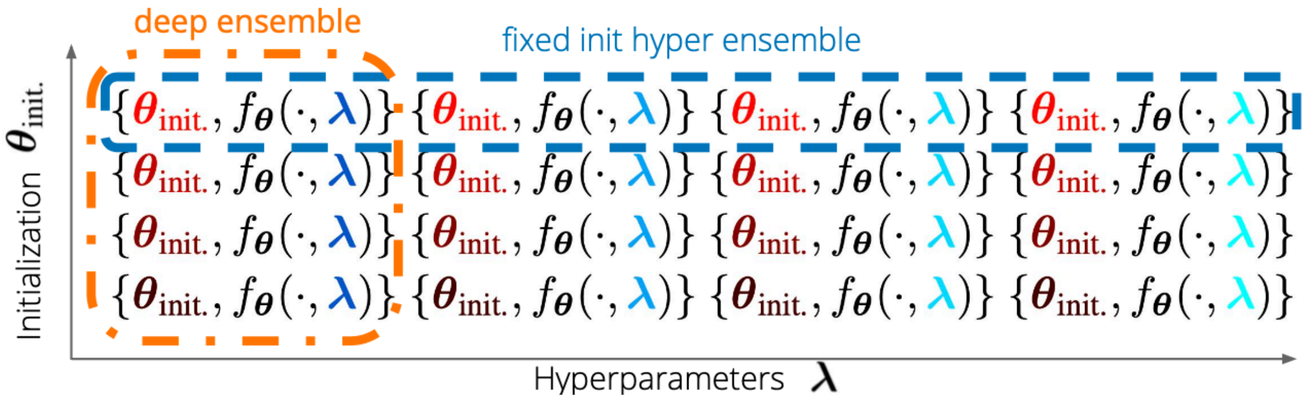
Provably efficient hyperparameter optimization with population-based bandits, Parker-Holder et al., ***NeurIPS*** 2020

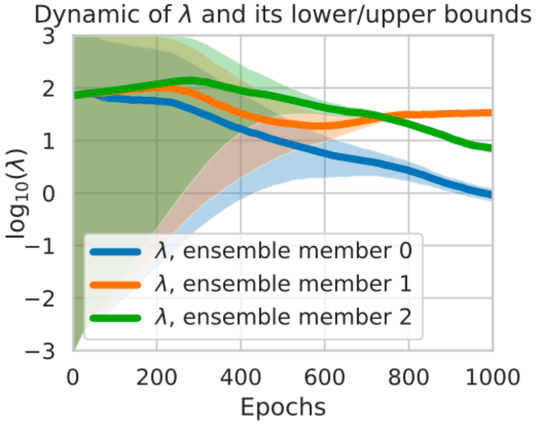
Advances: Hyper Deep Ensemble

- Hyper Deep Ensemble
  - Deep Ensemble
  - Hyper Ensemble
- BatchEnsemble → reduces memory cost

Algorithm 1: `hyper_deep_ens`(K, κ)

$\mathcal{M}_0 = \{f_{\boldsymbol{\theta}_j}(\cdot, \boldsymbol{\lambda}_j)\}_{j=1}^{\kappa} \leftarrow$ `rand_search`(κ);
$\mathcal{E}_0 \leftarrow$ `hyper_ens`$(\mathcal{M}_0,\ K)$ and $\mathcal{E}_{\text{strat.}} = \{\ \}$;
foreach $f_{\boldsymbol{\theta}}(\cdot, \boldsymbol{\lambda}) \in \mathcal{E}_0$.unique() **do**
  foreach $k \in \{1, \ldots, K\}$ **do**
    $\boldsymbol{\theta}' \leftarrow$ random initialization;
    $f_{\boldsymbol{\theta}_k}(\cdot, \boldsymbol{\lambda}) \leftarrow$ train $f_{\boldsymbol{\theta}'}(\cdot, \boldsymbol{\lambda})$;
    $\mathcal{E}_{\text{strat.}} = \mathcal{E}_{\text{strat.}} \cup \{\ f_{\boldsymbol{\theta}_k}(\cdot, \boldsymbol{\lambda})\}$;
  end
end
return `hyper_ens`$(\mathcal{E}_{\text{strat.}},\ K)$;

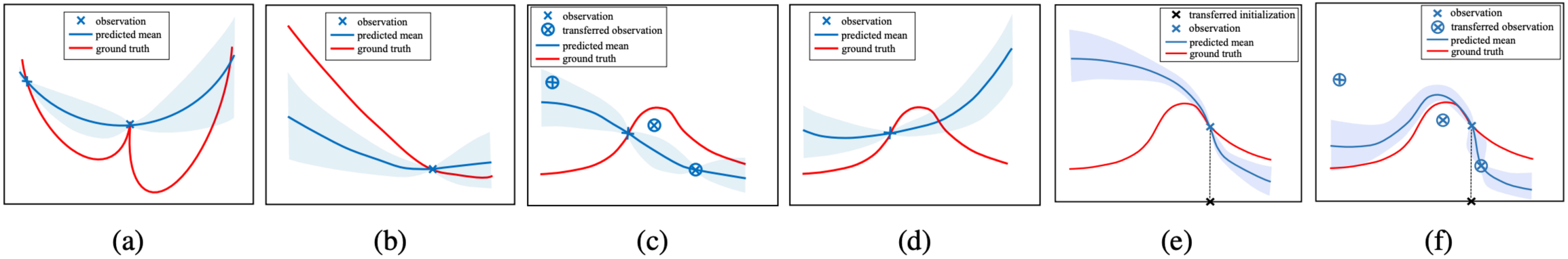




Hyperparameter Ensembles for Robustness and Uncertainty Quantification, Wenzel et al., ***NeurIPS*** 2020

Advances: Transferable HPO

- Transferable Neural Process (TNP)
  - end-to-end surrogate model that learns a comprehensive set of meta-knowledge
- Follows the strategy of MAML



Meta-learning Hyperparameter Performance Prediction with Neural Processes, Wei et al., ***ICML*** 2021

Neural Bootstrapper

Minsuk Shin[1]*, Hyungjoo Cho[2,3]*, Hyun-seok Min[2], Sungbin Lim[4]†
Department of Statistics, University of South Carolina[1]
Tomocube Inc.[2]
Department of Transdisciplinary Studies, Seoul National University[3]
Artificial Intelligence Graduate School, UNIST[4]
sungbin@unist.ac.kr


Shin, Minsuk


Cho, Hyungjoo


Min, Hyun-seok


LIM

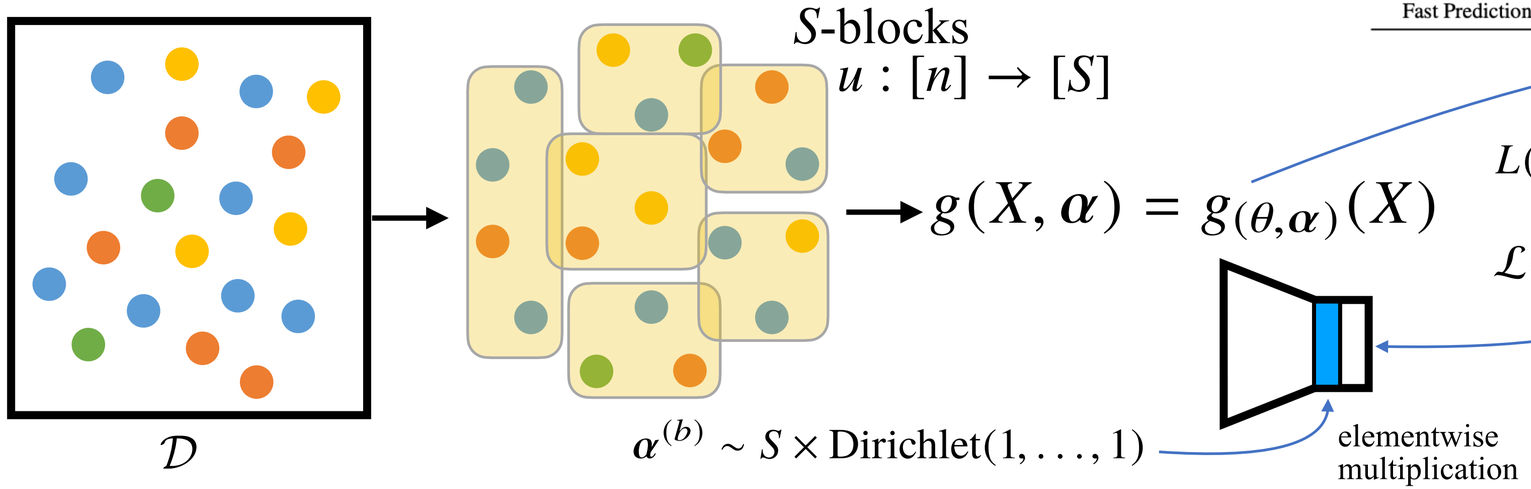
Neural Bootstrapper, ***NeurIPS*** (2021)
Joint work with M. Shin (Univ. of South Carolina), H. Cho (SNU), H. Min (Tomocube), **LIM**

NeuBoots: Neural Bootstrapper

NeuBoots = RWB + Generative Bootstrap + Adaptive Block Bootstrap

- Random Weight Bootstrapping (RWB)
- Generative Bootstrap Sampler (GBS, Shin et al., 2020)
- **Adaptive** Block Bootstrapping

	Standard Bootstrap [5]	MCDrop [9]	Deep Ensemble [18]	NeuBoots
Memory Efficiency	✗	✓	✗	✓
Fast Training	✗	✓	✗	✓
Fast Prediction	✗	✗	✗	✓



$$L(g, \alpha^{(b)}, \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} \alpha_{u(i)} \ell(g(X_i, \alpha), y_i)$$

$$\mathcal{L}(g, \mathcal{D}) = \mathbb{E}_{\alpha \sim S \times \text{Dirichlet}(1,\ldots,1)}[L(g, \alpha, \mathcal{D})]$$

Neural Bootstrapper, ***NeurIPS*** (2021)

Joint work with M. Shin (Univ. of South Carolina), H. Cho (SNU), H. Min (Tomocube), **LIM**

Neural Bootstrapping Attentive Neural Processes for Sequential Decision-Making Problems

Minsub Lee[1]*, Junhyun Park[1]*, Chanhui Lee[1], Hyungjoo Cho[2], Minsuk Shin[3], Sungbin Lim[1]†
Artificial Intelligence Graduate School, UNIST[1]
Department of Transdisciplinary Studies, Seoul National University[2]
Gauss Labs[3]

Lee, Minsub

Park, Junhyun

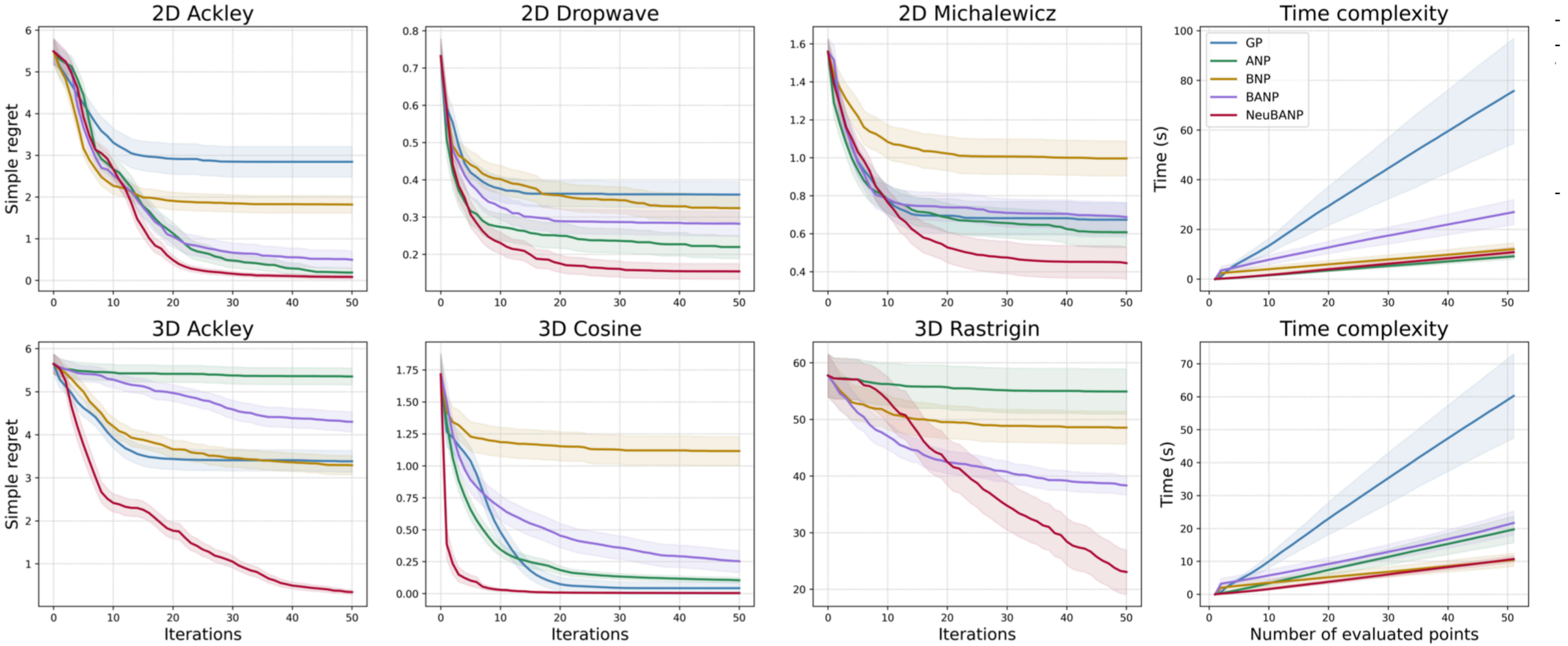
Lee, Chanhui

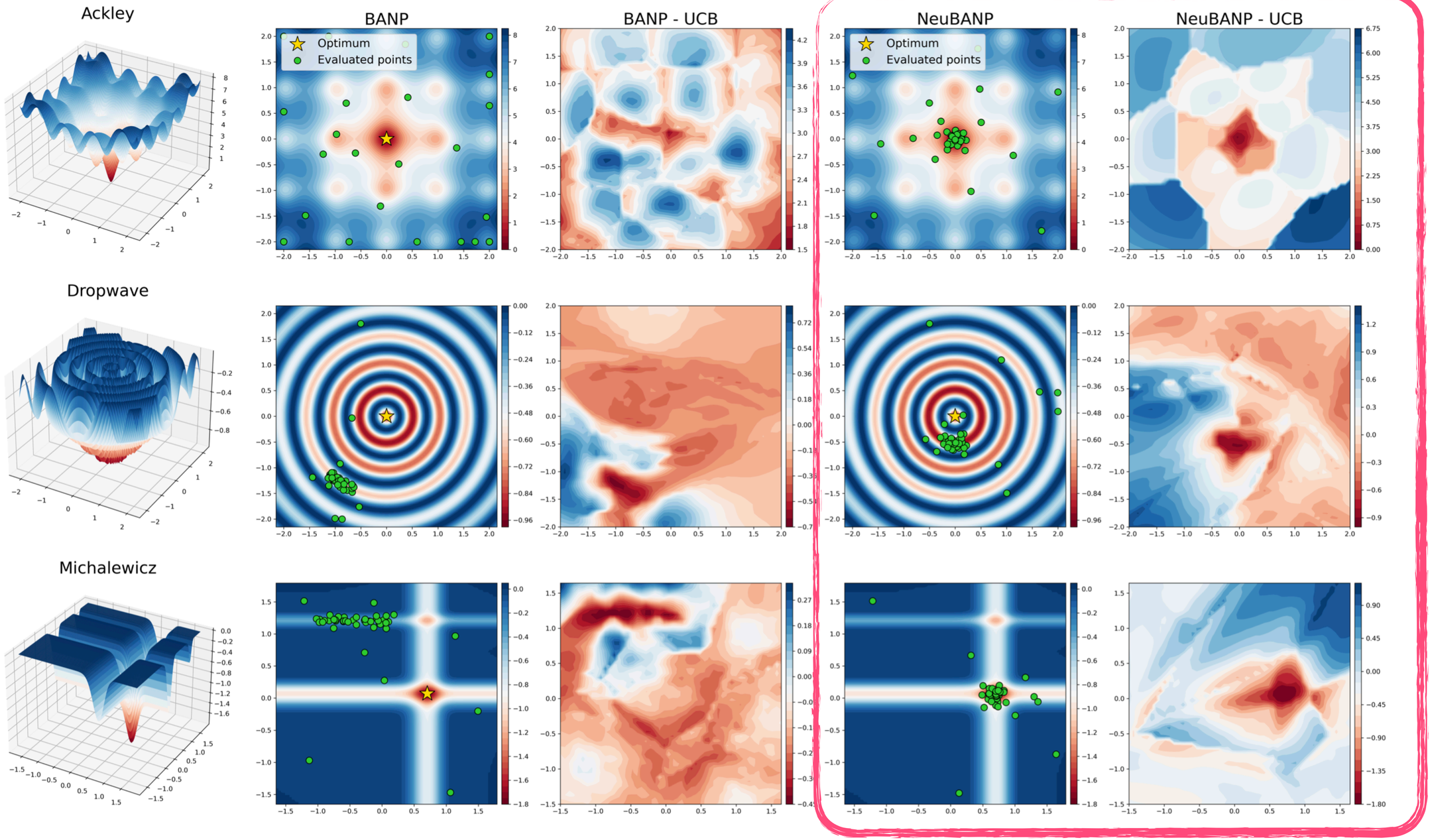
Cho, Hyungjoo

Shin, Minsuk

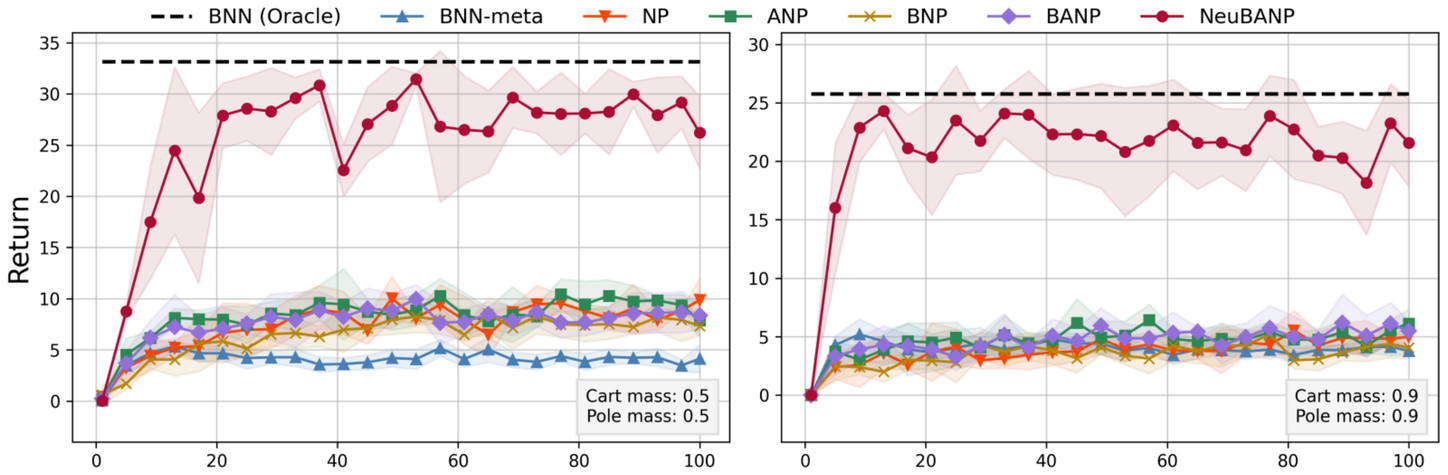
LIM

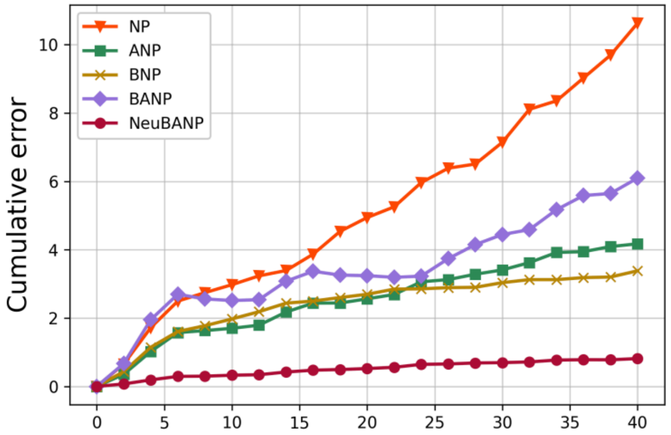
NeuBoots for Bayesian Optimization

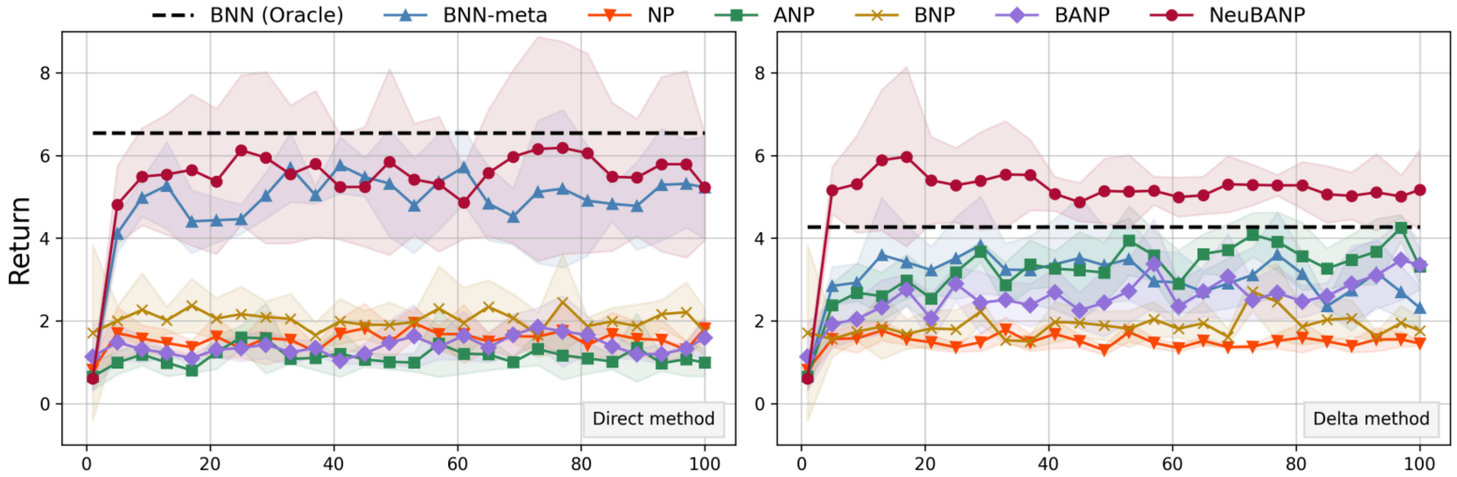



Ackley
BANP
BANP - UCB
NeuBANP
NeuBANP - UCB
Optimum
Evaluated points
Dropwave
Michalewicz

Meta Reinforcement Learning

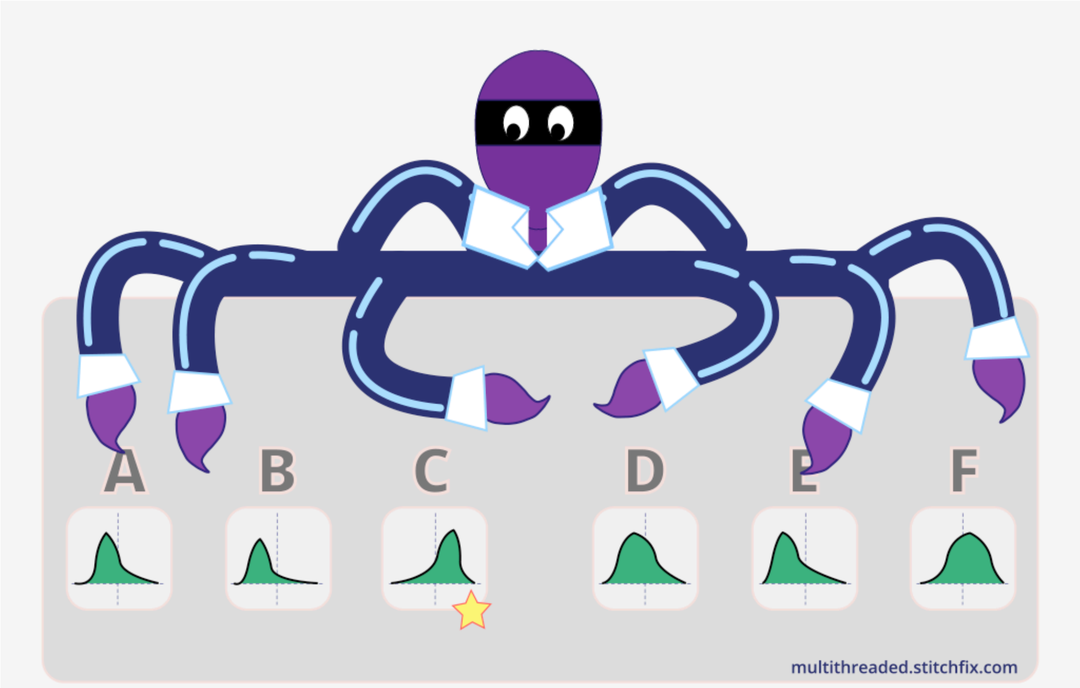


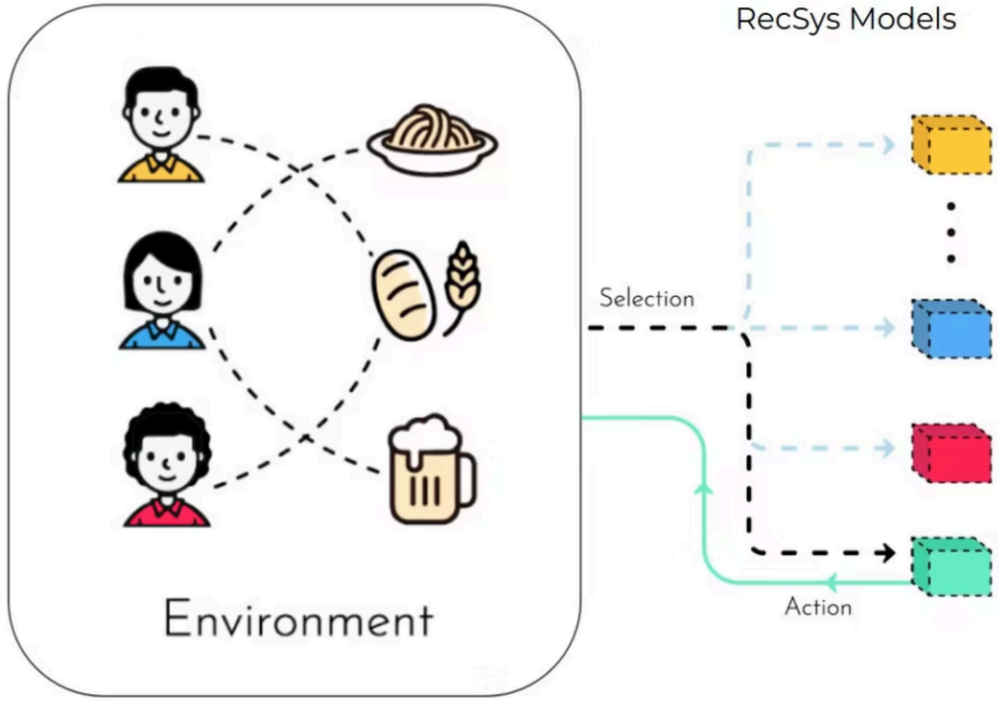




Contextual Multi-Armed Bandits

- How to find the best choice in a data-efficient way?





Contextual Multi-Armed Bandits

Difficult Problem

Regret	Method	$\delta = 0.5$	$\delta = 0.7$	$\delta = 0.9$	$\delta = 0.95$	$\delta = 0.99$
Cumulative	Uniform	100.00 ± 0.08	100.00 ± 0.09	100.00 ± 0.25	100.00 ± 0.37	100.00 ± 0.78
	Neural Linear	0.95 ± 0.02	1.60 ± 0.03	4.65 ± 0.18	9.56 ± 0.36	49.63 ± 2.41
	MAML	2.95 ± 0.12	3.11 ± 0.16	4.84 ± 0.22	7.01 ± 0.33	22.93 ± 1.57
	NP	1.60 ± 0.06	1.75 ± 0.05	3.31 ± 0.10	5.71 ± 0.24	22.13 ± 1.23
	NeuBANP	**0.85** ± 0.22	**1.02** ± 0.27	**1.85** ± 0.56	**3.04** ± 0.88	**9.76** ± 1.93
Simple	Uniform	100.00 ± 0.45	100.00 ± 0.78	100.00 ± 1.18	100.00 ± 2.21	100.00 ± 4.21
	Neural Linear	**0.33** ± 0.04	**0.79** ± 0.07	2.17 ± 0.14	4.08 ± 0.20	35.89 ± 2.98
	MAML	2.49 ± 0.12	3.00 ± 0.35	4.75 ± 0.48	7.10 ± 0.77	22.89 ± 1.41
	NP	1.04 ± 0.06	1.26 ± 0.21	2.90 ± 0.35	5.45 ± 0.47	21.45 ± 1.3
	NeuBANP	0.86 ± 0.06	1.04 ± 0.08	**1.88** ± 0.14	**3.09** ± 0.23	**9.96** ± 0.70

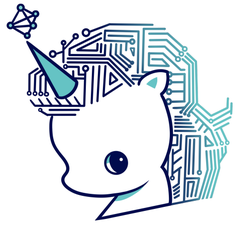
Neural Bootstrapping Attention Neural Processes, ***submitted*** (2021)

Joint work with M. Lee (UNIST), J. Park (UNIST), S. Jang (UNIST), C. Lee (UNIST), H. Cho (SNU), M. Shin (Univ. of South Carolina)

NeurIPS Paper Reading

Principles of Deep Learning (AI502/IE408/IE511)

Sungbin Lim (UNIST AIGS & IE)



Contact: ai502deeplearning@gmail.com

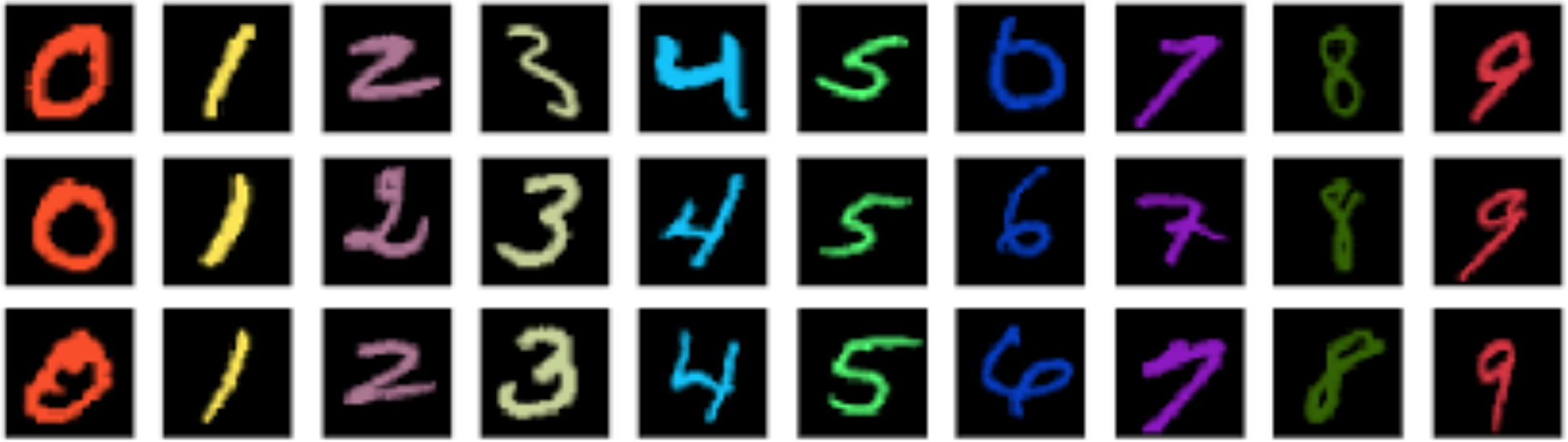
On Feature Learning in the Presence of Spurious Correlations

Pavel Izmailov* **Polina Kirichenko*** **Nate Gruver*** **Andrew Gordon Wilson**
New York University

Abstract

Deep classifiers are known to rely on spurious features — patterns which are correlated with the target on the training data but not inherently relevant to the learning problem, such as the image backgrounds when classifying the foregrounds. In this paper we evaluate the amount of information about the core (non-spurious) features that can be decoded from the representations learned by standard empirical risk minimization (ERM) and specialized group robustness training. Following recent work on Deep Feature Reweighting (DFR), we evaluate the feature representations by re-training the last layer of the model on a held-out set where the spurious correlation is broken. On multiple vision and NLP problems, we show that the features learned by simple ERM are highly competitive with the features learned by specialized group robustness methods targeted at reducing the effect of spurious correlations. Moreover, we show that the quality of learned feature representations is greatly affected by the design decisions beyond the training method, such as the model architecture and pre-training strategy. On the other hand, we find that strong regularization is not necessary for learning high-quality feature representations. Finally, using insights from our analysis, we significantly improve upon the best results reported in the literature on the popular Waterbirds, CelebA hair color prediction and WILDS-FMOW problems, achieving 97%, 92% and 50% worst-group accuracies, respectively.

Spurious Correlation



Reconstructing Training Data from Trained Neural Networks

Niv Haim*
Weizmann Institute of Science
niv.haim@weizmann.ac.il

Gal Vardi*†
TTI Chicago and Hebrew University
galvardi@ttic.edu

Gilad Yehudai*
Weizmann Institute of Science
gilad.yehudai@weizmann.ac.il

Ohad Shamir
Weizmann Institute of Science
ohad.shamir@weizmann.ac.il

Michal Irani
Weizmann Institute of Science
michal.irani@weizmann.ac.il

Project page: https://giladude1.github.io/reconstruction

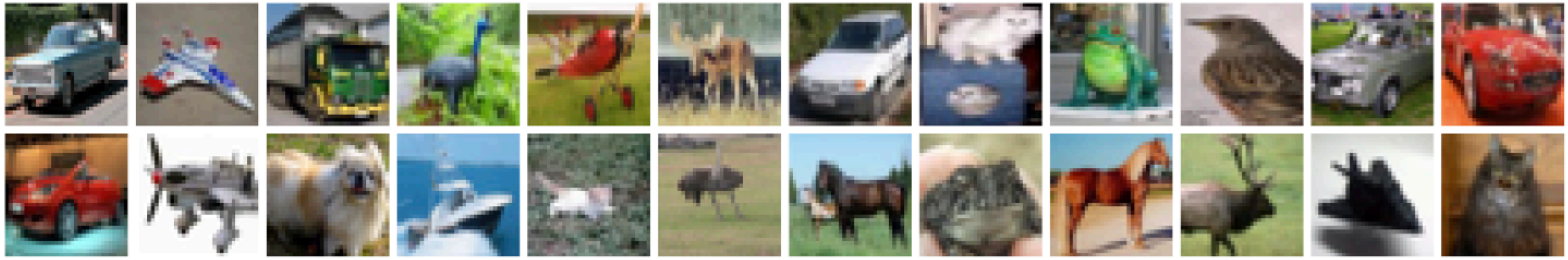
Abstract

Understanding to what extent neural networks memorize training data is an intriguing question with practical and theoretical implications. In this paper we show that in some cases a significant fraction of the training data can in fact be reconstructed from the parameters of a trained neural network classifier. We propose a novel reconstruction scheme that stems from recent theoretical results about the implicit bias in training neural networks with gradient-based methods. To the best of our knowledge, our results are the first to show that reconstructing a large portion of the actual training samples from a trained neural network classifier is generally possible. This has negative implications on privacy, as it can be used as an attack for revealing sensitive training data. We demonstrate our method for binary MLP classifiers on a few standard computer vision datasets.

Can we recover training data from neural net?

(a) Top 24 images reconstructed from a binary classifier trained on 50 CIFAR10 images

(b) Their corresponding nearest neighbours from the training-set of the model



On the Strong Correlation Between Model Invariance and Generalization

Weijian Deng Stephen Gould Liang Zheng
Australian National University
{firstname.lastname}@anu.edu.au

Abstract

Generalization and invariance are two essential properties of machine learning models. Generalization captures a model's ability to classify unseen data while invariance measures consistency of model predictions on transformations of the data. Existing research suggests a positive relationship: a model generalizing well should be invariant to certain visual factors. Building on this qualitative implication we make two contributions. First, we introduce effective invariance (EI), a simple and reasonable measure of model invariance which does not rely on image labels. Given predictions on a test image and its transformed version, EI measures how well the predictions agree and with what level of confidence. Second, using invariance scores computed by EI, we perform large-scale quantitative correlation studies between generalization and invariance, focusing on rotation and grayscale transformations. From a model-centric view, we observe generalization and invariance of different models exhibit *a strong linear relationship*, on both in-distribution and out-of-distribution datasets. From a dataset-centric view, we find a certain model's accuracy and invariance *linearly correlated* on different test sets. Apart from these major findings, other minor but interesting insights are also discussed.

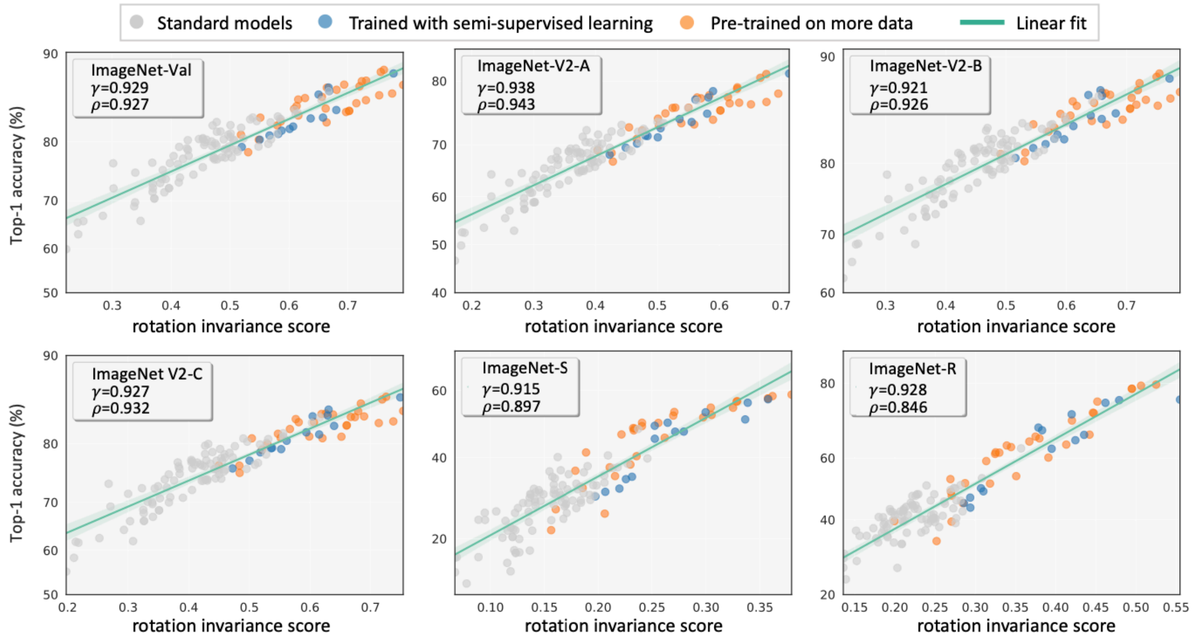


Figure 2: **Correlation between accuracy (%) and rotation invariance (EI) for** 150 **models.** Each figure is obtained from testing on a different ImageNet test set. In each figure, each dot denotes a model, and straight lines are fit by robust linear fit [90]. The shaded region in each figure is a 95% confidence region for the linear fit from $1,000$ bootstrap samples. We clearly observe a strong linear relationship (Pearson's Correlation $r > 0.915$ and Spearman's Correlation $\rho > 0.875$).

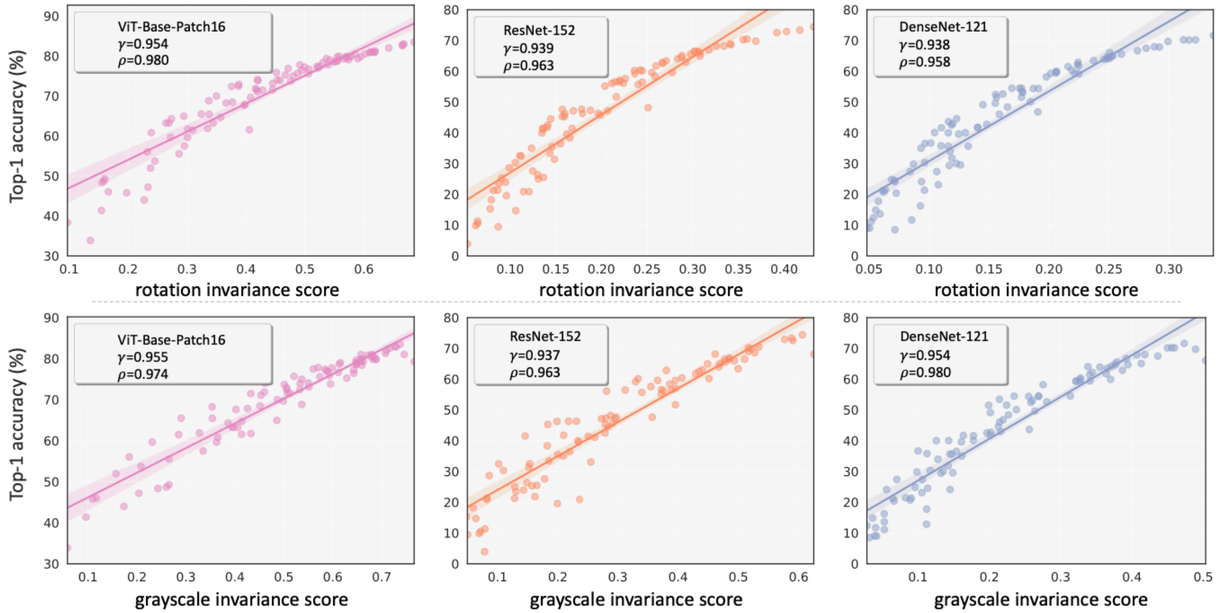


Figure 8: **Correlation between a model's invariance and accuracy on various OOD test sets.** In each figure, a data point corresponds to a test set from ImageNet-C [87]. The straight lines are fit with robust linear regression [90]. We test rotation invariance (top) and grayscale invariance (bottom). In all sugfigures, we observe a strong negative correlation (Pearson's Correlation r and Spearman's Rank Correlation ρ are greater than 0.930) between invariance and accuracy.

Q&A