Transfer NAS with Meta-learned Bayesian Surrogates

Gresa Shala[1], **Thomas Elsken**[2], **Frank Hutter**[1,2] **& Josif Grabocka**[1]
[1] Department of Computer Science, University of Freiburg
{shalag,fh,grabocka}@cs.uni-freiburg.de
[2]Bosch Center for Artificial Intelligence
thomas.elsken@de.bosch.com

ICLR 2023

Background

Hyperparameter Optimization (HPO)

Neural Architecture Search (NAS)

NAS的意义在于解决深度学习模型的调参问题，是结合了优化和机器学习的交叉研究。在深度学习之前，传统的机器学模型习也会遇到模型的调参问题，因为浅层模型结构相对简单，因此多数研究都将模型的结构统一为超参数来进行搜索，比如三层神经网络中隐层神经元的个数。优化这些超参数的方法主要是黑箱优化方法，比如分别为进化优化，贝叶斯优化和强化学习等。

Meta Learning

Meta Learning的核心就是让机器学会学习（learn to learn）举个例子就是，机器之前学习了100个task，之后机器学习第101个task的时候，会因为之前学习的100个task所具有的学习能力，而让第101个task表现得更好。比如说第一个任务是语音识别，第二个任务是图像识别，第一百个任务是文本分类，机器会因为之前所学到得任务，所以在后面得任务学习得更好。

Transfer Learning

迁移学习（Transfer Learning）通俗来讲就是学会举一反三的能力，通过运用已有的知识来学习新的知识，其核心是找到已有知识和新知识之间的相似性，通过这种相似性的迁移达到迁移学习的目的。世间万事万物皆有共性，如何合理地找寻它们之间的相似性，进而利用这个桥梁来帮助学习新知识，是迁移学习的核心问题

Bayesian optimization (BO)

假设我们有一个函数$f: x \to \mathbb{R}$,我们需要在$x \subseteq X$内找到 $x^* = \underset{x \in X}{\operatorname{argmin}} f(x)$

Algorithm 1 Sequential Model-Based Optimization

Input: $f, \mathcal{X}, S, \mathcal{M}$
$\mathcal{D} \leftarrow \text{INITSAMPLES}(f, \mathcal{X})$
for $i \leftarrow |\mathcal{D}|$ **to** T **do**
  $p(y \mid \mathbf{x}, \mathcal{D}) \leftarrow \text{FITMODEL}(\mathcal{M}, \mathcal{D})$
  $\mathbf{x}_i \leftarrow \arg\max_{\mathbf{x} \in \mathcal{X}} S(\mathbf{x}, p(y \mid \mathbf{x}, \mathcal{D}))$
  $y_i \leftarrow f(\mathbf{x}_i)$ ▷ Expensive step
  $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{x}_i, y_i)$
end for

- f是评估函数，即输入一组超参数，得到一个输出值
- X是超参数搜索空间
- D表示一个由若干对数据组成的数据集，每一对数组表示为(x,y)，x是一组超参数，y表示该组超参数对应的结果
- S是Acquisition Function(采集函数)，这个函数的作用是用来选择x
- M是对数据集D进行拟合的模型，可以用来假设的模型有很多种，例如随机森林、高斯过程等

Gaussian Processes (GP)

一元高斯分布

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}}\exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

多元高斯分布

$$p(\boldsymbol{x}) = (2\pi)^{-\frac{n}{2}}|K|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T K^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

$$x \sim \mathcal{N}(\boldsymbol{\mu}, K)$$

其中 $\boldsymbol{\mu} \in \mathbb{R}^n$ 是均值向量，$K \in \mathbb{R}^{n\times n}$ 为协方差矩阵

Gaussian Processes (GP)

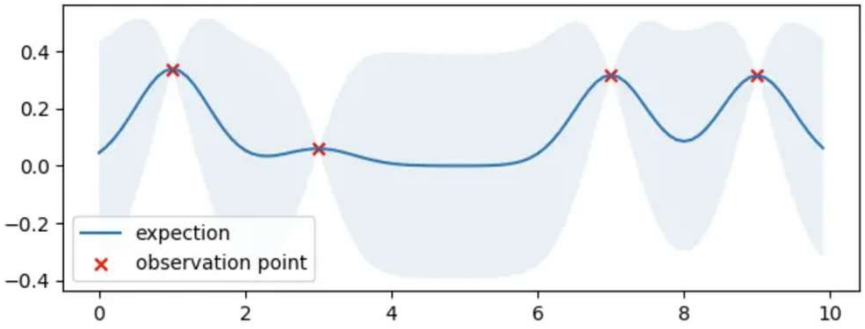
高斯过程 $f(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}), \kappa(\boldsymbol{x}, \boldsymbol{x}))$

$\boldsymbol{\mu}(\boldsymbol{x}) : \mathbb{R}^{\mathrm{n}} \rightarrow \mathbb{R}^{\mathrm{n}}$ 表示均值函数Mean function，返回各个维度的均值

$\kappa(\boldsymbol{x}, \boldsymbol{x}) : \mathbb{R}^{\mathrm{n}} \times \mathbb{R}^{\mathrm{n}} \rightarrow \mathbb{R}^{\mathrm{n} \times \mathrm{n}}$ 表示协方差函数 Covariance Function，也叫核函数 Kernel Function）返回两个向量各个维度之间的协方差矩阵

径向基函数 RBF $K(x_i, x_j) = \sigma^2 \exp\left(-\frac{\|x_i - x_j\|_2^2}{2l^2}\right)$

不同的核函数有不同的衡量方法，得到的高斯过程的性质也不一样。



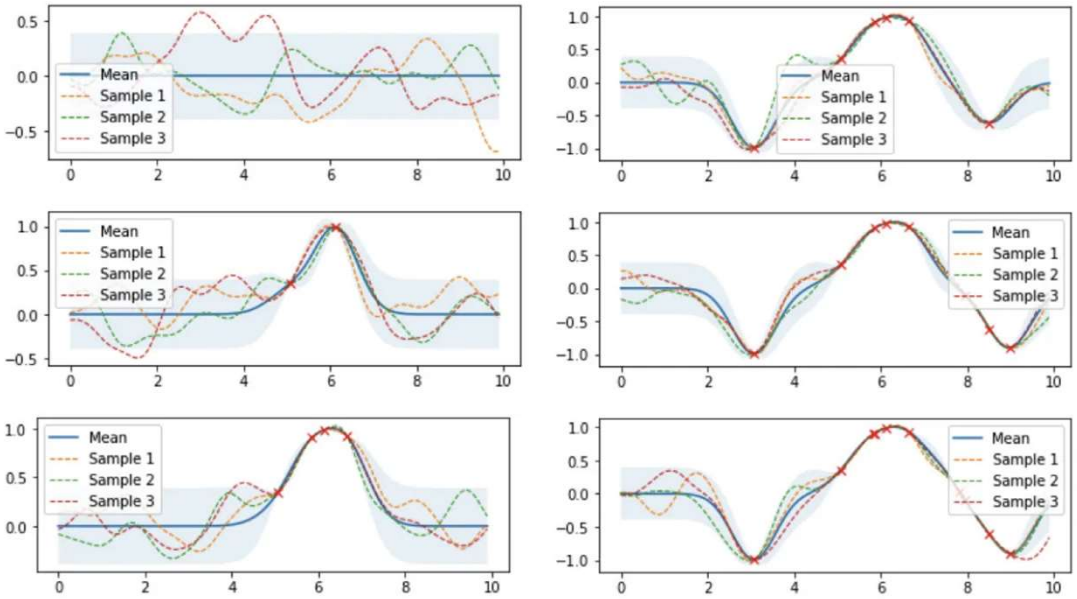
Gaussian Processes (GP)

将高斯过程的先验表示为 $f(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}_f, K_{ff})$

如果现在我们观测到一些数据 $(\boldsymbol{x}^*, \boldsymbol{y}^*)$

并且假设 $\boldsymbol{y}^*$ 与 $f(\boldsymbol{x})$ 服从联合高斯分布

$$\begin{bmatrix} f(\boldsymbol{x}) \\ \boldsymbol{y}^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_f \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} K_{ff} & K_{fy} \\ K_{fy}^T & K_{yy} \end{bmatrix} \right)$$



其中 $K_{ff} = \kappa(\boldsymbol{x}, \boldsymbol{x})$， $K_{fy} = \kappa(\boldsymbol{x}, \boldsymbol{x}^*)$， $K_{yy} = \kappa(\boldsymbol{x}^*, \boldsymbol{x}^*)$，则有

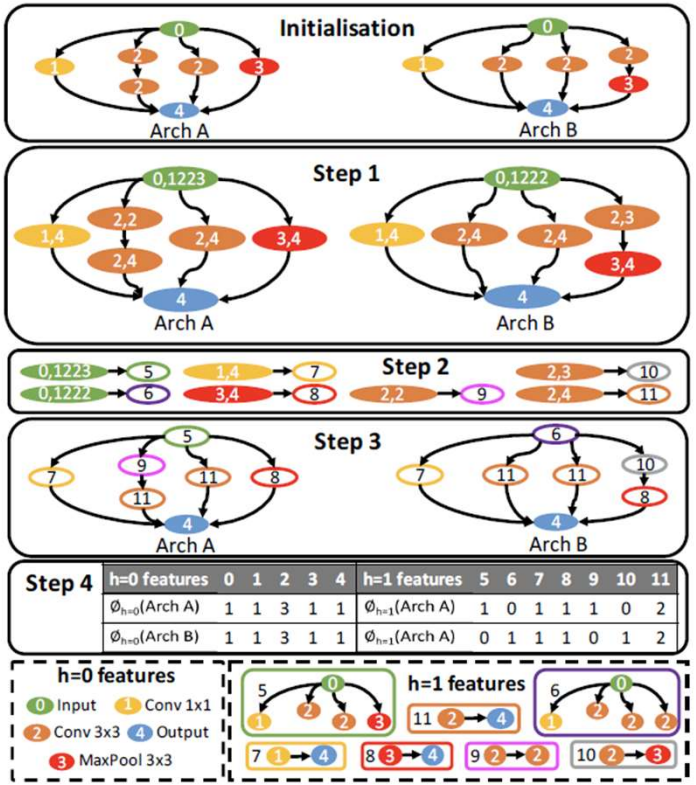
$$f \sim \mathcal{N}(K_{fy}^T K_{ff}^{-1} \boldsymbol{y} + \boldsymbol{\mu}_f, K_{yy} - K_{fy}^T K_{ff}^{-1} K_{fy})$$

Related work

Bayesian optimization for NAS

Algorithm 1 NAS-BOWL Algorithm.
Optional steps of the exemplary use of motif-based warm starting (Sec 3.2) are marked in *gray italics*.

Input: Maximum BO iterations T, BO batch size b, acquisition function $\alpha(\cdot)$, initial observed data on the target task $\mathcal{D}_0$, Optional: past-task query data $\mathcal{D}_{\text{past}}$ and surrogate $\mathcal{S}_{\text{past}}$
Output: The best architecture G_T^*
Initialise the GPWL surrogate $\mathcal{S}$ with $\mathcal{D}_0$
for $t = 1, \ldots, T$ **do**
  if *Pruning based on the past-task motifs* **then**
    Compute the motif importance scores (equation 3.4) with $\mathcal{S}_{\text{past}}/\mathcal{S}$ on $\mathcal{D}_{\text{past}}/\mathcal{D}_t$
    while $|\mathcal{G}_t| < B$ **do**
      Generate a batch of candidate architectures and reject those which contain none of the top 25% good motifs (similar procedure as Fig. 2(a))
    end while
  else
    Generate B candidate architectures $\mathcal{G}_t$
  end if
  $\{G_{t,i}\}_{i=1}^{b} = \arg\max_{G \in \mathcal{G}_t} \alpha_t(G|\mathcal{D}_{t-1})$
  Evaluate their validation accuracy $\{y_{t,i}\}_{i=1}^{b}$
  $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup (\{G_{t,i}\}_{i=1}^{B}, \{y_{t,i}\}_{i=1}^{b})$
  Update the surrogate $\mathcal{S}$ with $\mathcal{D}_t$
end for
Return the best architecture seen so far G_T^*



Bayesian optimization with deep kernel gaussian processes

$$\begin{bmatrix} y \\ y^* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{pmatrix} K(x,x) & K(x,x^*) \\ K(x,x^*)^T & K(x^*,x^*) \end{pmatrix}\right)$$

$$\text{kernel function } k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$$
$$K(x,x^*)_{i,j} := k(x_i, x_j^*)$$

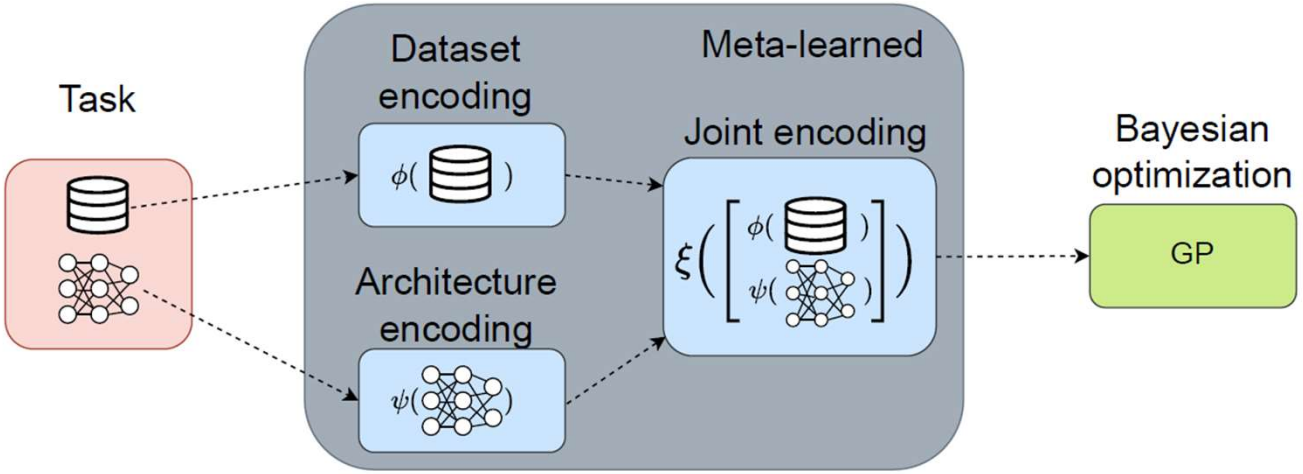
$$\mathbb{E}[y^* \mid x^*, x, y] = K(x^*,x)K(x,x)^{-1}y$$
$$\text{cov}[y^* \mid x^*, x] = K(x^*,x^*) - K(x,x^*)^T K(x,x)^{-1} K(x,x^*)$$

GPs are lazy models that rely on the similarity of the test instances to the training instances via kernel functions k, such as the Matern kernel.

Unfortunately, typical kernels used with GPs are designed manually and rely on sub-optimal assumptions.

A promising direction for designing powerful and efficient kernel functions that adapt to a learning task is Deep Kernel Learning

Method



The embedding ξ for the deep kernel

fully connected neural network $\xi : R_{K+L} \rightarrow R_M$

The architecture encoding ψ

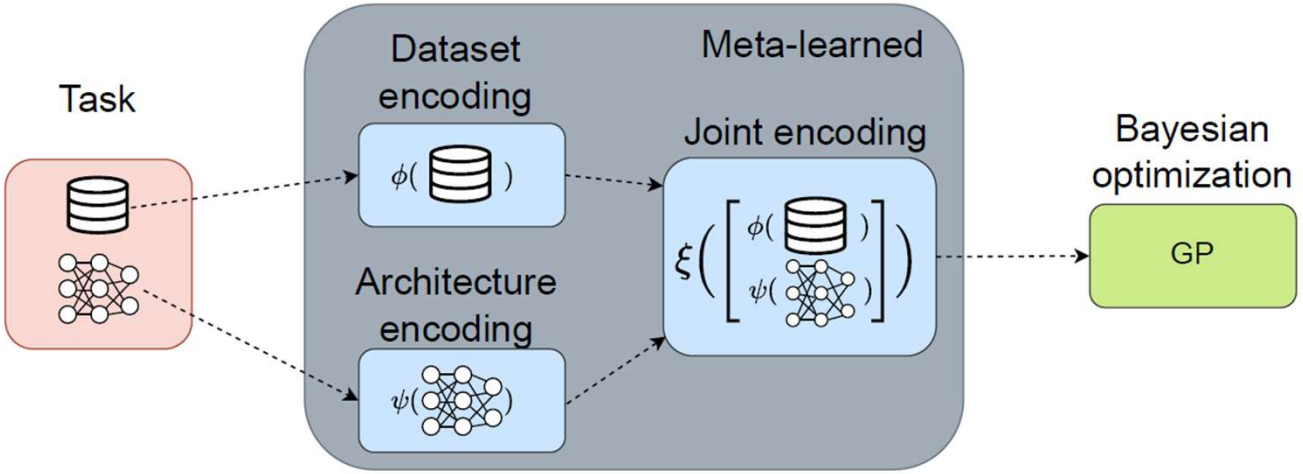
consists of a directed acyclic graph encoder (Zhang et al., 2019)

The dataset encoding ϕ

consists of two stacked Set-Transformer (Lee et al., 2019) architectures.

The first Set-Transformer layer captures the interaction between randomly sampled data points of the same class, whereas the second one captures the interactions between the different classes of the dataset.

Method



Putting it all together the kernel/similarity between two architectures, specifically x evaluated on dataset D, and x' evaluated on dataset D', is:

$$k\left(\mathbf{x}, \mathcal{D}, \mathbf{x}', \mathcal{D}'; w\right) = k\Bigg(\xi\Big(\Big[\psi(\mathbf{x}; w^{(\psi)}), \phi\left(\mathcal{D}; w^{(\phi)}\right)\Big]; w^{(\xi)}\Big),$$
$$\xi\Big(\Big[\psi(\mathbf{x}'; w^{(\psi)}), \phi\left(\mathcal{D}'; w^{(\phi)}\right)\Big]; w^{(\xi)}\Big); w^{(k)}\Bigg)$$

Method

The parameters w of the deep kernel are optimized jointly by maximizing the log marginal likelihood of the GP surrogate on the meta training dataset:

$$\arg\max_{w} \quad \log p\left(\mathbf{y} \mid \mathbf{x}, \mathcal{D}\,;\, w\right)$$

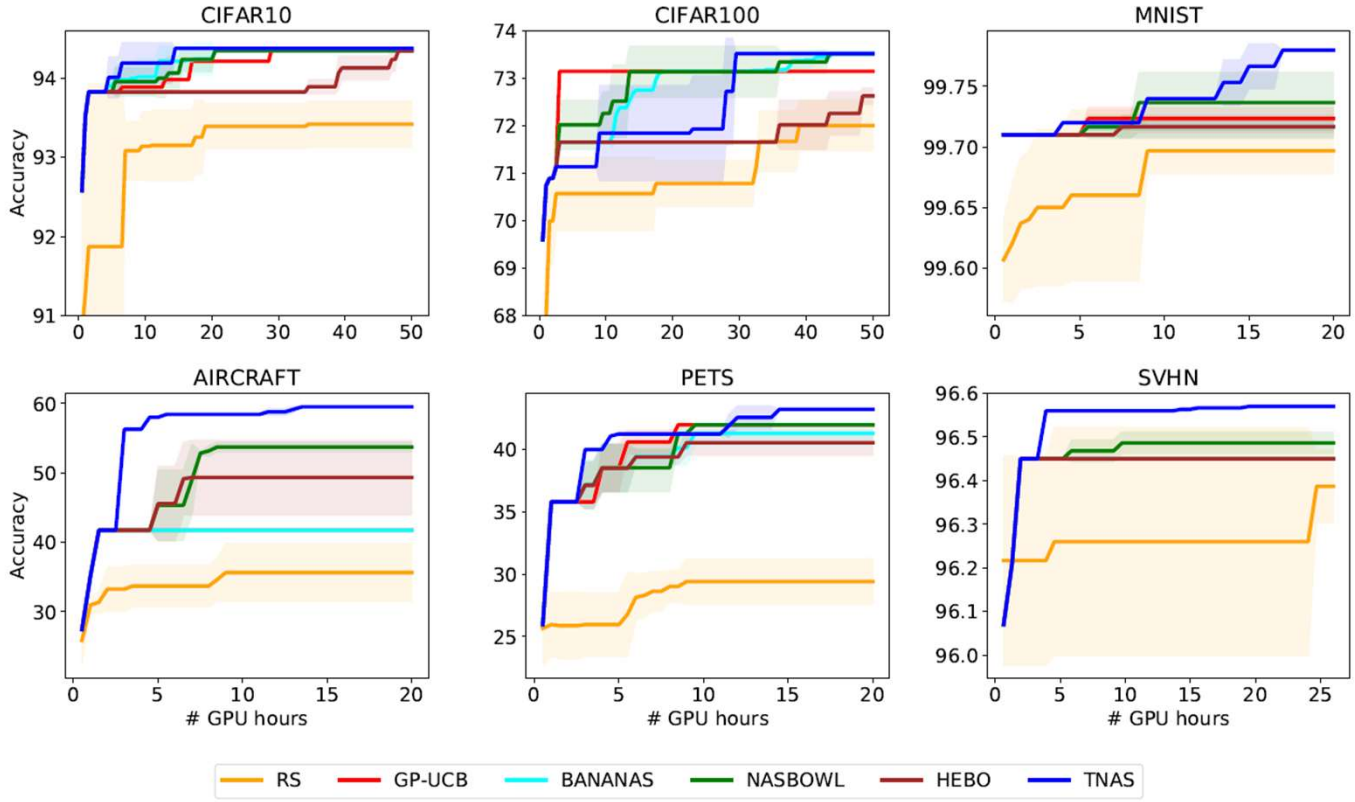
$$\propto \arg\min_{w} \quad \mathbf{y}^T K^{-1}(\mathbf{x}, \mathcal{D}; w)\ \mathbf{y} + \log |K(\mathbf{x}, \mathcal{D}; w)|.$$

Algorithm 1 Meta-learning our deep-kernel GPs

Require: meta-dataset $\mathcal{M}$; learning rates η_{SGD}, η_{REP}; inner update steps v.
while not converged **do**
  Sample mini-batch from $\mathcal{M}$:
  $\mathbf{x} = [x_1, \ldots, x_k], \mathbf{y} = [y_1, \ldots, y_k], \mathcal{D} = [\mathcal{D}_1, \ldots, \mathcal{D}_k]$
  $\mathcal{L}(w) = \mathbf{y}^T K^{-1}(\mathbf{x}, \mathcal{D}; w)\mathbf{y} + \log |K(\mathbf{x}, \mathcal{D}; w)|$
  $w' \leftarrow w$
  for $j = 1$ to v **do**
    $w' \leftarrow w' - \eta_{SGD} \nabla_{w'} \mathcal{L}(w')$
  Update $w \leftarrow w - \eta_{REP}\,(w - w')$

Experiment

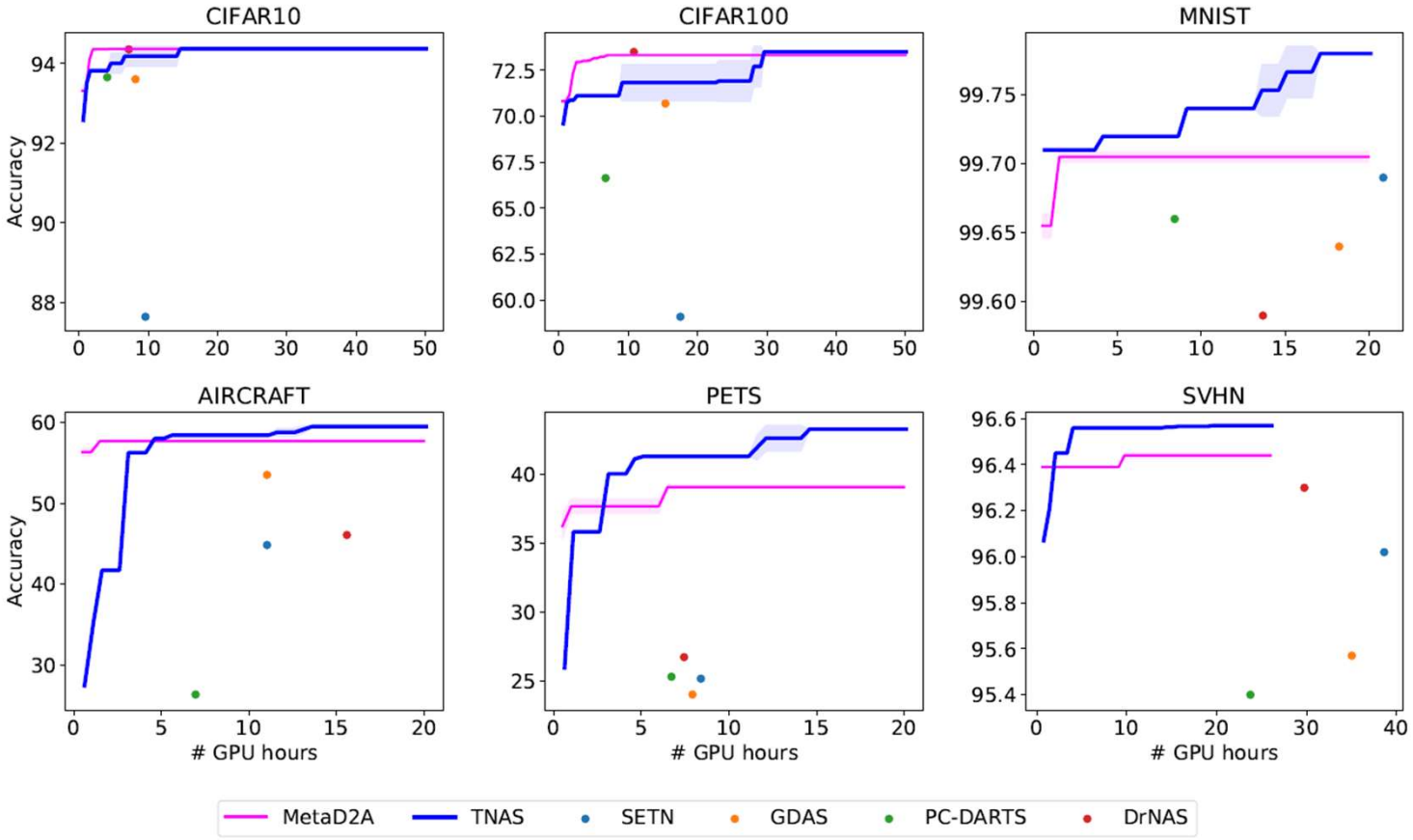
Comparing TNAS to random search (RS) and the four different Bayesian optimization methods on six image datasets



TNAS is more efficient than classical HPO methods applied to NAS as well as HPO methods specifically adapted to NAS and outperforms them in terms of anytime-performance, while achieving strong final performance.

Experiment

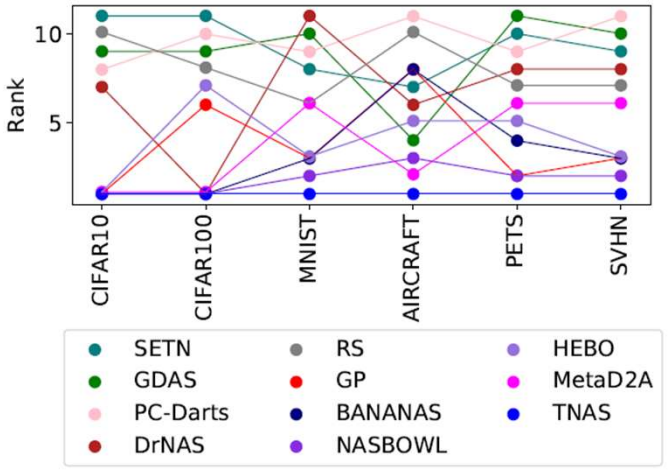
Comparison of TNAS to state-of-the-art NAS methods



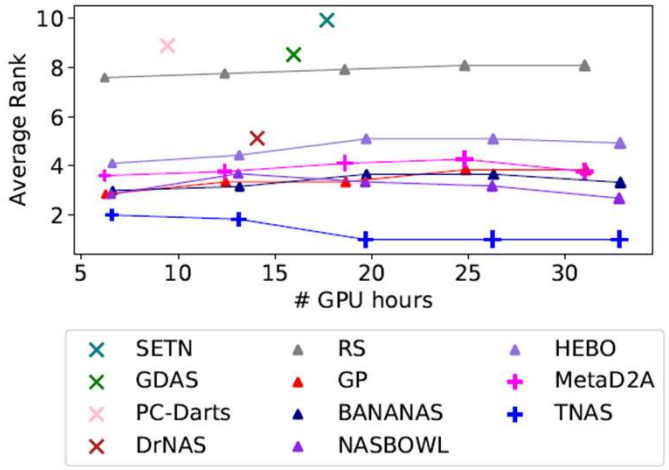
TNAS is competitive with one-shot approaches in terms of runtime

Experiment

Consistency of TNAS compared to baselines



(a) Ranking of TNAS across benchmarks.

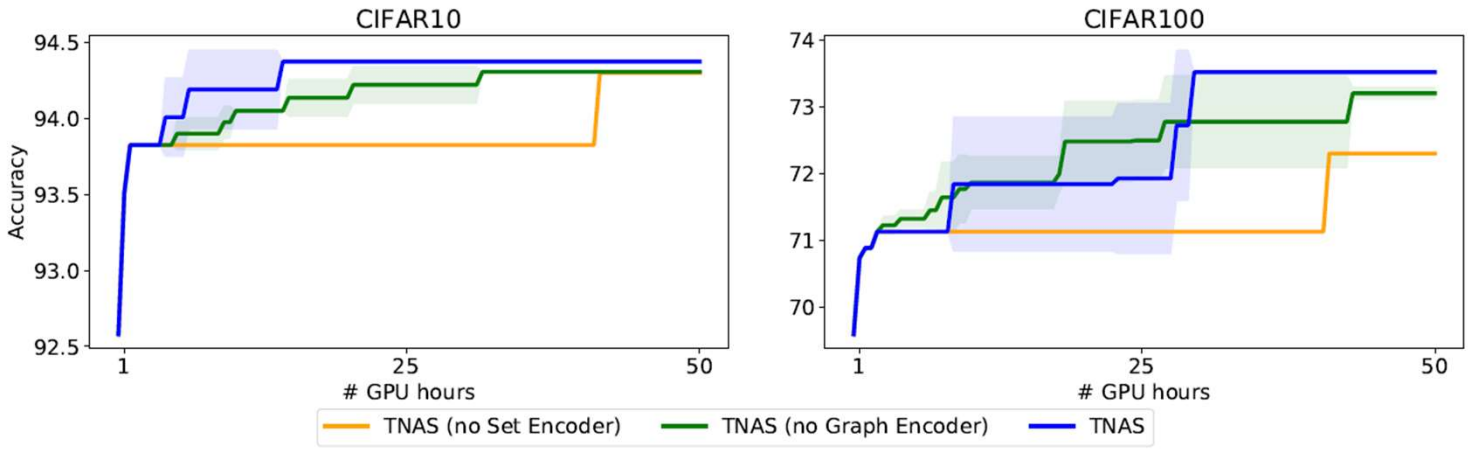


(b) Ranking (averaged across benchmarks) over the course of runtime.

TNAS consistently achieves strong results, while the existing state-of-the-art NAS baselines have much higher variance - their ranking changes across benchmarks

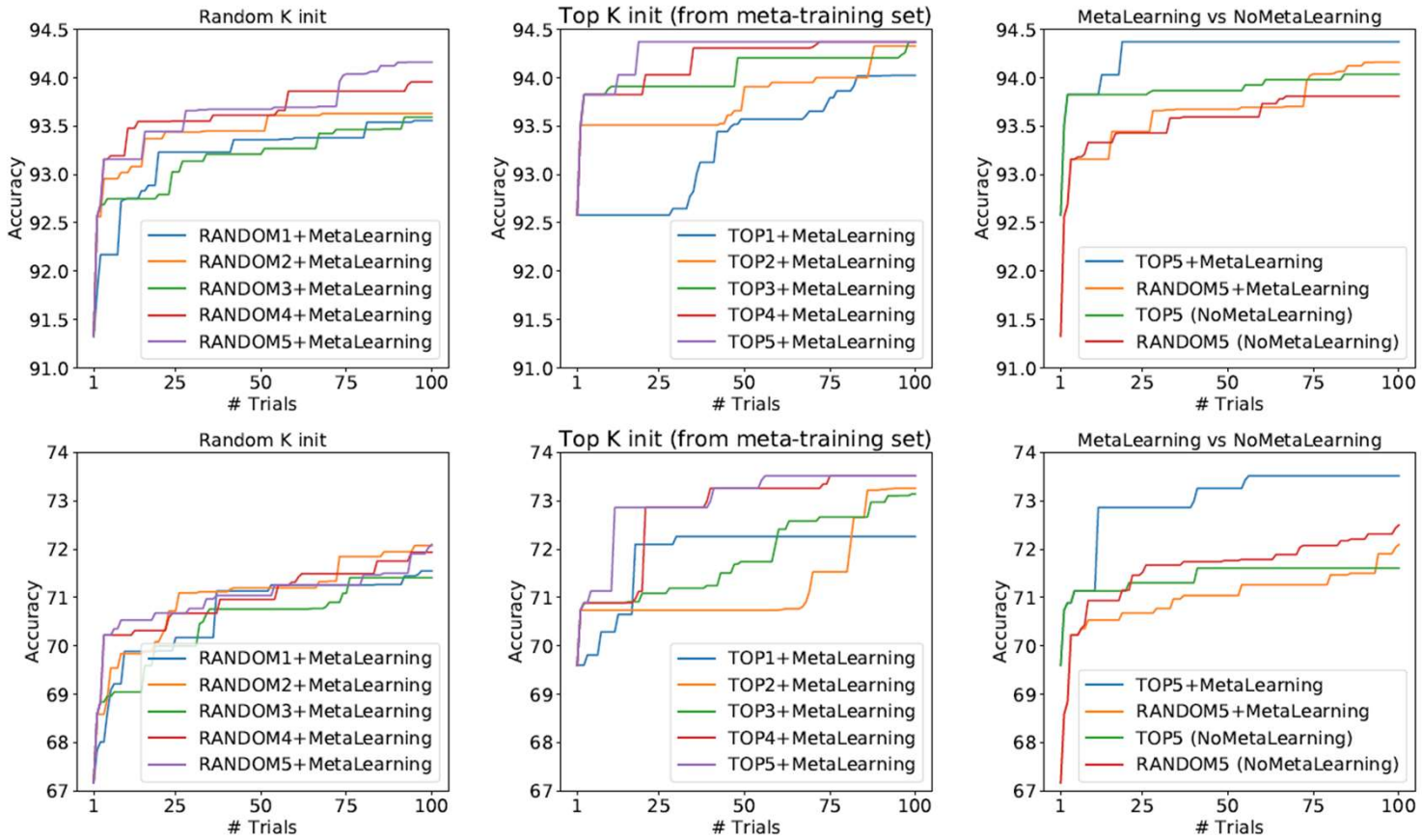
Ablation

Ablation of the components of TNAS



Ablation

Ablation of the initial design and the initialization of TNAS



Comparison

Comparison of TNAS to MetaD2A

	predictor	candidate architecture generation	test-time adaptation
Lee et al. (2021)	(dataset, architecture) encoding, MLP (deterministic prediction)	GNN-based architecture generator, select top-K based on predictor	no
Ours	(dataset, architecture) encoding, MLP, GP (probabilistic prediction)	BO, maximize acquisition function	yes, adapt GP predictor w.r.t. new candidate evaluations

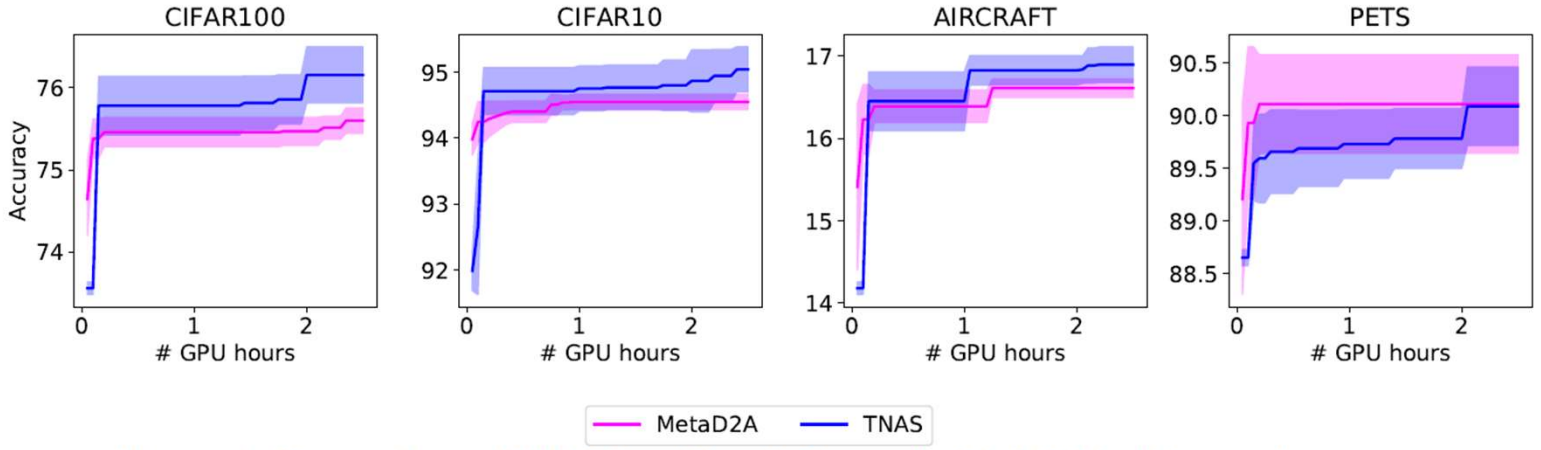


Figure 7: Comparison of TNAS to MetaD2A on the MobileNetV3 search space.

Comparison

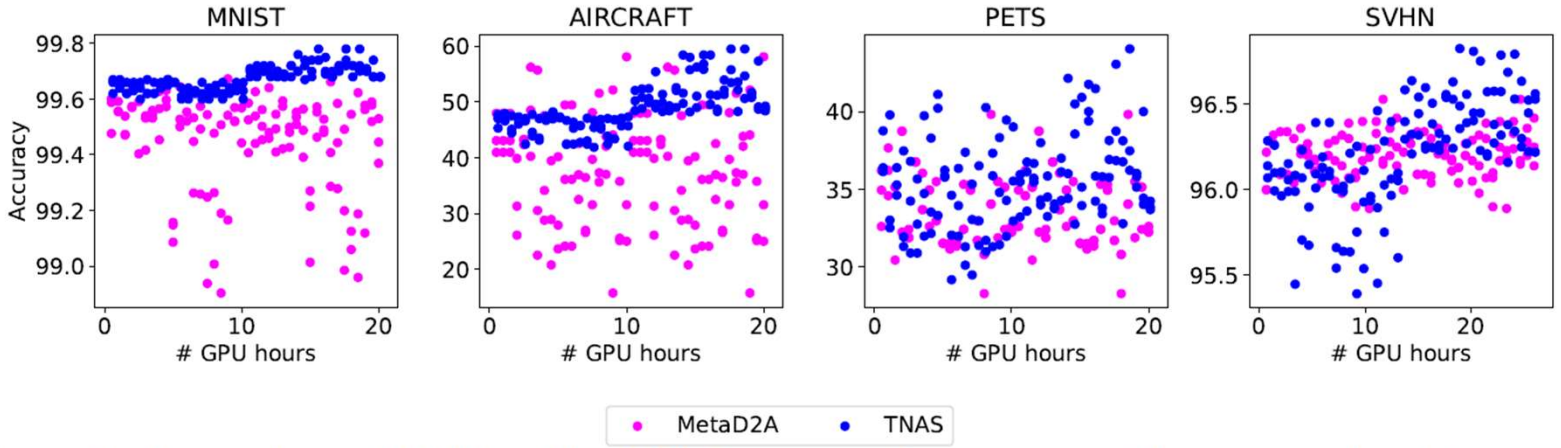


Figure 8: Comparison of TNAS to MetaD2A in terms of the accuracy of the suggested architectures to evaluate on the NASBench201 search space. For MetaD2A the architectures generated by its architecture generator are evaluated sequentially based on MetaD2A's predictor ranking. TNAS adapts during the BO loop iterations, and thus suggests architectures conditioned on the previous evaluations.

Comparison

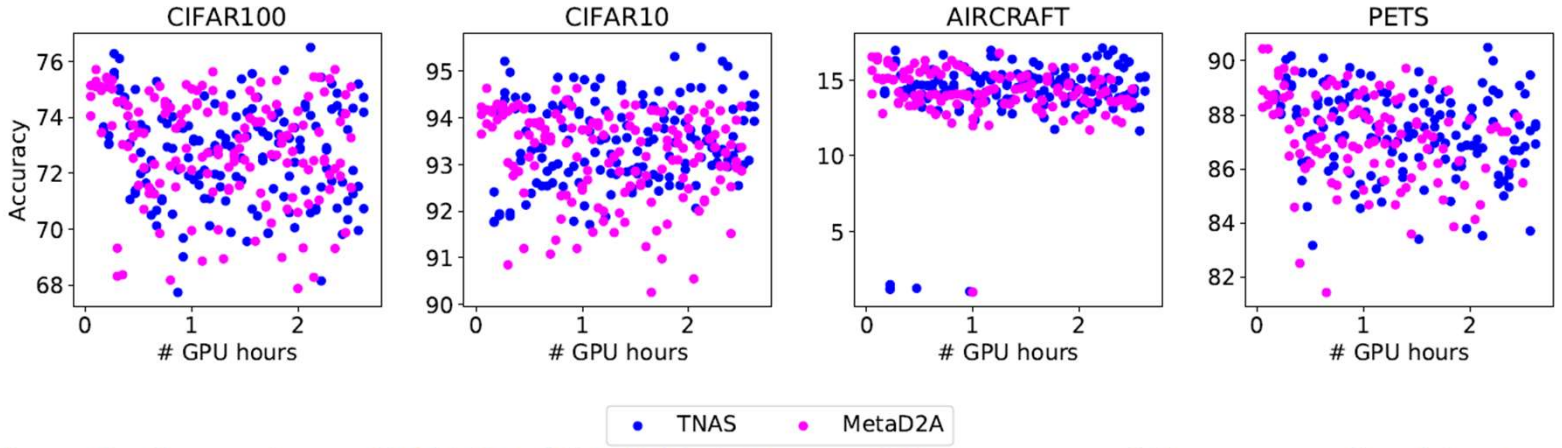


Figure 9: Comparison of TNAS to MetaD2A in terms of the accuracy of the suggested architectures to evaluate on the MobileNetV3 search space.

Thanks