Optimal Transport Kernels for Sequential and Parallel Neural Architecture Search

Vu Nguyen[*1] **Tam Le**[*2] **Makoto Yamada**[2 3] **Michael A. Osborne**[4]

Abstract

Neural architecture search (NAS) automates the design of deep neural networks. One of the main challenges in searching complex and non-continuous architectures is to compare the similarity of networks that the conventional Euclidean metric may fail to capture. Optimal transport (OT) is resilient to such complex structure by considering the minimal cost for transporting a network into another. However, the OT is generally not negative definite which may limit its ability to build the positive-definite kernels required in many kernel-dependent frameworks. Building upon tree-Wasserstein (TW), which is a negative definite variant of OT, we develop a novel discrepancy for neural architectures, and demonstrate it within a Gaussian process (GP) surrogate model for the sequential NAS settings. Furthermore, we derive a novel parallel NAS, using quality k-determinantal point process on the GP posterior, to select diverse and high-performing architectures from a discrete set of candidates. We empirically demonstrate that our TW-based approaches outperform other baselines in both sequential and parallel NAS.

1. Introduction

Neural architecture search (NAS) is the process of automating architecture engineering to find the best design of our neural network model. This output architecture will perform well for a given dataset. With the increasing interest in deep learning in recent years, NAS has attracted significant research attention (Dong & Yang, 2019; Elsken et al., 2019a; Liu et al., 2018; 2019; Luo et al., 2018; Real et al., 2019; 2017; Shah et al., 2018; Suganuma et al., 2017; Xie & Yuille, 2017; Yao et al., 2020). We refer the interested readers to the survey (Elsken et al., 2019b) for a detailed review of NAS and to the comprehensive list[1] for all of the related papers in NAS.

Bayesian optimization (BO) utilizes a probabilistic model, particularly Gaussian process (GP) (Rasmussen, 2006), for determining future evaluations and its evaluation efficiency makes it well suited for the expensive evaluations of NAS. However, the conventional BO approaches (Shahriari et al., 2016; Snoek et al., 2012) are not suitable to capture the complex and non-continuous designs of neural architectures. Recent work (Kandasamy et al., 2018) has considered optimal transport (OT) for measuring neural architectures. This views two networks as logistical *suppliers* and *receivers*, then optimizes to find the minimal transportation cost as the distance, i.e., similar architectures will need less cost for transporting and vice versa. However, the existing OT distance for architectures, such as OTMANN (Kandasamy et al., 2018), do not easily lend themselves to the creation of the positive semi-definite (p.s.d.) kernel (covariance function) due to the indefinite property of OT (Peyré & Cuturi, 2019) (§8.3). It is critical as the GP is not a valid random process when the covariance function (kernel) is not p.s.d. (see Lem. 2.1). In addition, there is still an open research direction for *parallel NAS* where the goal is to select multiple high-performing and diverse candidates from a *discrete* set of candidates for parallel evaluations. This discrete property makes the parallel NAS interesting and different from the existing batch BO approaches (Desautels et al., 2014; González et al., 2016), which are typically designed to handle continuous observations.

We propose a negative definite tree-Wasserstein (TW) distance for neural network architectures based on a novel design which captures both global and local information via n-gram and indegree/outdegree representations for networks. In addition, we propose the k-determinantal point process (k-DPP) quality for selecting diverse and high-performing architectures from a *discrete* set. This discrete property of NAS makes k-DPP ideal in sampling the choices overcoming the greedy selection used in the existing batch Bayesian optimization (Desautels et al., 2014; González et al., 2016;

[*]Equal contribution [1]Amazon Adelaide (work done prior to joining Amazon) [2]RIKEN AIP [3]Kyoto University [4]University of Oxford. Correspondence to: Vu Nguyen <vu@ieee.org>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

[1]https://www.automl.org/automl/literature-on-neural-architecture-search

Wang et al., 2018). At a high level, our contributions are three-fold as follows:

- A TW distance with a novel design for capturing both local and global information from architectures which results in a p.s.d. kernel while the existing OT distance does not.
- A demonstration of TW as the novel GP covariance function for sequential NAS.
- A parallel NAS approach using k-DPP for selecting diverse and high-quality architectures from a discrete set.

2. Tree-Wasserstein for Neural Network Architectures

We first argue that the covariance matrices associated with a kernel function of Gaussian process (GP) and k-DPP need to be positive semi-definite (p.s.d.) for a valid random process in Lemma 2.1. We then develop tree-Wasserstein (TW) (Do Ba et al., 2011; Le et al., 2019b), the negative definite variant of optimal transport (OT), for measuring the similarity of architectures. Consequently, we can build a p.s.d. kernel upon OT geometry for modeling with GPs and k-determinantal point processes (k-DPPs).

Lemma 2.1. *If a covariance function k of a Gaussian process is not positive semi-definite, the resulting GP is not a valid random process.*

Proof of Lemma 2.1 is placed in the Appendix §D.1.

2.1. Tree-Wasserstein

We give a brief review about OT, tree metric, tree-Wasserstein (TW) which are the main components for our NAS framework. We denote $[n] = \{1, 2, \ldots, n\}, \forall n \in \mathbb{N}_+$. Let (Ω, d) be a measurable metric space. For any $x \in \Omega$, we use δ_x for the Dirac unit mass on x.

Optimal transport. OT, a.k.a. Wasserstein, Monge-Kantorovich, or Earth Mover's distance, is the flexible tool to compare probability measures (Peyré & Cuturi, 2019; Villani, 2003). Let ω, ν be Borel probability distributions on Ω and $\mathcal{R}(\omega, \nu)$ be the set of probability distributions π on $\Omega \times \Omega$ such that $\pi(B \times \Omega) = \omega(B)$ and $\pi(\Omega \times B') = \nu(B')$ for all Borel sets B, B'. The 1-Wasserstein distance W_d (Villani, 2003) (p.2) between ω and ν is defined as:

$$W_d(\omega, \nu) = \inf_{\pi \in \mathcal{R}(\omega,\nu)} \int_{\Omega \times \Omega} d(x, z)\pi(\mathrm{d}x, \mathrm{d}z), \quad (1)$$

where d is a ground metric (i.e., cost metric) of OT.

Tree metrics and tree-Wasserstein. A metric $d : \Omega \times \Omega \to \mathbb{R}_+$ is a *tree metric* if there exists a tree $\mathcal{T}$ with positive edge lengths such that $\forall x \in \Omega$, then x is a node of $\mathcal{T}$; and $\forall x, z \in \Omega$, $d(x, z)$ is equal to the length of the (unique) path between x and z (Semple & Steel, 2003) (§7, p.145–182).

Let $d_{\mathcal{T}}$ be the tree metric on tree $\mathcal{T}$ rooted at r. For $x, z \in \mathcal{T}$, we denote $\mathcal{P}(x, z)$ as the (unique) path between x and z. We write $\Gamma(x)$ for a set of nodes in the subtree of $\mathcal{T}$ rooted at x, defined as $\Gamma(x) = \big\{z \in \mathcal{T} \mid x \in \mathcal{P}(r, z)\big\}$. For edge e in $\mathcal{T}$, let v_e be the deeper level node of edge e (the farther node to root r), and w_e be the positive length of that edge.

Tree-Wasserstein (TW) is a special case of OT whose ground metric is a tree metric (Do Ba et al., 2011; Le et al., 2019b). Given two measures ω, ν supported on tree $\mathcal{T}$, and setting the tree metric $d_{\mathcal{T}}$ as the ground metric, then the TW distance $W_{d_{\mathcal{T}}}$ between ω and ν admits a closed-form solution as follows:

$$W_{d_{\mathcal{T}}}(\omega, \nu) = \sum_{e \in \mathcal{T}} w_e \big|\omega\big(\Gamma(v_e)\big) - \nu\big(\Gamma(v_e)\big)\big|, \quad (2)$$

where $\omega(\Gamma(v_e))$ is the total mass of the probability measure ω in the subtree $\Gamma(v_e)$ rooted at v_e. It is important to note that we can derive p.s.d. kernels on tree-Wasserstein distance $W_{d_{\mathcal{T}}}$ (Le et al., 2019b), as opposed to the standard OT W_d for general ground metric d (Peyré & Cuturi, 2019).

2.2. Tree-Wasserstein for Neural Networks

We present a new approach leveraging the tree-Wasserstein for measuring the similarity of neural network architectures. We consider a neural network architecture $\mathbf{x}$ by $(\mathcal{S}^o, A)$ where $\mathcal{S}^o$ is a multi-set of operations in each layer of $\mathbf{x}$, and A is an adjacency matrix, representing the connection among these layers (i.e., network structure) in $\mathbf{x}$. We can also view a neural network as a directed labeled graph where each layer is a node in a graph, and an operation in each layer is a node label (i.e., A represents the graph structure, and $\mathcal{S}^o$ contains a set of node labels). We then propose to extract information from neural network architectures by distilling them into three separate quantities as follows:

• **n-gram representation for layer operations.** Each neural network consists of several operations from the input layer to the output layer. Inspired by the n-gram representation for a document in natural language processing, we view a neural network as a document and its operations as words. Therefore, we can use n-grams (i.e., n-length paths) to represent operations used in the neural network. We then normalize the n-gram, and denote it as $\mathbf{x}^o$ for a neural network $\mathbf{x}$.

Particularly, for $n = 1$, the n-gram representation is a frequency vector of operations, used in Nasbot (Kandasamy et al., 2018). When we use all $n \leq \ell$ where ℓ is the num-

ber of network layers, the n-gram representation shares the same spirit as the path encoding, used in Bananas (White et al., 2021).

Let $\mathbb{S}$ be the set of operations, and $\mathbb{S}^n = \mathbb{S} \times \mathbb{S} \times \cdots \times \mathbb{S}$ (n times of $\mathbb{S}$), the n-gram can be represented as empirical measures in the followings

$$\omega_{\mathbf{x}}^o = \sum_{s \in \mathbb{S}^n} \mathbf{x}_s^o \delta_s, \qquad \omega_{\mathbf{z}}^o = \sum_{s \in \mathbb{S}^n} \mathbf{z}_s^o \delta_s, \tag{3}$$

where $\mathbf{x}_s^o$ and $\mathbf{z}_s^o$ are the frequency of n-gram operation $s \in \mathbb{S}^n$ in architecture $\mathbf{x}$ and $\mathbf{z}$, respectively.

We can leverage the TW distance to compare the n-gram representations $\omega_{\mathbf{x}}^o$ and $\omega_{\mathbf{z}}^o$ using Eq. (2), denoted as $W_{d_{\mathcal{T}_o}}(\omega_{\mathbf{x}}^o, \omega_{\mathbf{z}}^o)$. To compute this distance, we utilize a predefined tree structure for network operations by hierarchically grouping similar network operations into a tree as illustrated in Fig. 1. We can utilize the domain knowledge to define the grouping and the edge weights, such as we can have `conv1` and `conv3` in the same group and `maxpool` is from another group. Inspired by the partition-based tree metric sampling (Le et al., 2019b), we define the edge weights decreasing when the edge is far from the root. Although such design can be subjective, the final distance (defined later in Eq. (5)) will be calibrated and normalized properly when modeling with a GP in §3. We refer to Fig. 8 and Appendix §F for the example of TW computation for neural network architectures.

• **Indegree and outdegree representations for network structure.** We extract the *indegree* and *outdegree* of each layer, which are the number of ingoing and outgoing layers respectively, as an alternative way to represent a network structure. We denote $L_{\mathbf{x}}$ as the set of all layers which one can reach from the input layer for neural network $\mathbf{x}$. Let $\eta_{x,\ell}$ and M_x be lengths of the longest paths from an input layer to the layer ℓ and to the output layer respectively. Such paths interpret the order of layers in a neural network which starts with the input layer, connect with some middle layers, and end with the output layer, we represent the indegree and outdegree of network layers in $\mathbf{x}$ as empirical measures $\omega_{\mathbf{x}}^{d^-}$ and $\omega_{\mathbf{x}}^{d^+}$, defined as

$$\omega_{\mathbf{x}}^{d^-} = \sum_{\ell \in L_{\mathbf{x}}} \mathbf{x}_\ell^{d^-} \delta_{\frac{\eta_{x,\ell}+1}{M_x+1}}, \qquad \omega_{\mathbf{x}}^{d^+} = \sum_{\ell \in L_{\mathbf{x}}} \mathbf{x}_\ell^{d^+} \delta_{\frac{\eta_{x,\ell}+1}{M_x+1}}, \tag{4}$$

where $\mathbf{x}_\ell^{d^-}$ and $\mathbf{x}_\ell^{d^+}$ are the normalized indegree and outdegree of the layer ℓ of $\mathbf{x}$ respectively.

For indegree and outdegree information, the supports of empirical measures $\omega_{\mathbf{x}}^{d^-}$, and $\omega_{\mathbf{z}}^{d^-}$ are in one-dimensional space that a tree structure reduces to a chain of supports. Thus, we can use $W_{d_{\mathcal{T}_-}}\left(\omega_{\mathbf{x}}^{d^-}, \omega_{\mathbf{z}}^{d^-}\right)$ to compare those empirical measures.[2] Similarly, we have $W_{d_{\mathcal{T}_+}}\left(\omega_{\mathbf{x}}^{d^+}, \omega_{\mathbf{z}}^{d^+}\right)$ for empirical measures $\omega_{\mathbf{x}}^{d^+}$ and $\omega_{\mathbf{z}}^{d^+}$ built from outdegree information.

Tree-Wasserstein distance for neural networks. Given neural networks $\mathbf{x}$ and $\mathbf{z}$, we consider three separate TW distances for the n-gram, indegree and outdegree representations of the networks respectively. Then, we define d_{NN} as a convex combination with nonnegative weights $\{\alpha_1, \alpha_2, \alpha_3 \mid \sum_i \alpha_i = 1, \alpha_i \geq 0\}$ for $W_{d_{\mathcal{T}_o}}$, $W_{d_{\mathcal{T}_-}}$, and $W_{d_{\mathcal{T}_+}}$ respectively, to compare neural networks $\mathbf{x}$ and $\mathbf{z}$ as:

$$\begin{aligned} d_{\text{NN}}(\mathbf{x}, \mathbf{z}) = \alpha_1 W_{d_{\mathcal{T}_o}}(\mathbf{x}^o, \mathbf{z}^o) + \alpha_2 W_{d_{\mathcal{T}_-}}\left(\omega_{\mathbf{x}}^{d^-}, \omega_{\mathbf{z}}^{d^-}\right) \\ + (1 - \alpha_1 - \alpha_2)\, W_{d_{\mathcal{T}_+}}\left(\omega_{\mathbf{x}}^{d^+}, \omega_{\mathbf{z}}^{d^+}\right). \end{aligned} \tag{5}$$

The proposed discrepancy d_{NN} can capture not only the frequency of layer operations, but also network structures, e.g., indegree and outdegree of network layers.

We illustrate our proposed TW for neural networks in Fig. 10 describing each component in Eq. (5). We also describe the detailed calculations in the Appendix §F. We highlight a useful property of our proposed d_{NN}: it can compare two architectures with *different* layer sizes and/or operations sizes.

Proposition 1. *The d_{NN} for neural networks is a pseudo-metric and negative definite.*

Proof of Proposition 1 is placed in the Appendix §D.2.

Our discrepancy d_{NN} is negative definite as opposed to the OT for neural networks considered in Kandasamy et al. (2018) which is indefinite. Therefore, from Proposition 1 and following Theorem 3.2.2 in Berg et al. (1984), we can derive a positive definite TW kernel upon d_{NN} for neural networks $\mathbf{x}, \mathbf{z}$ as

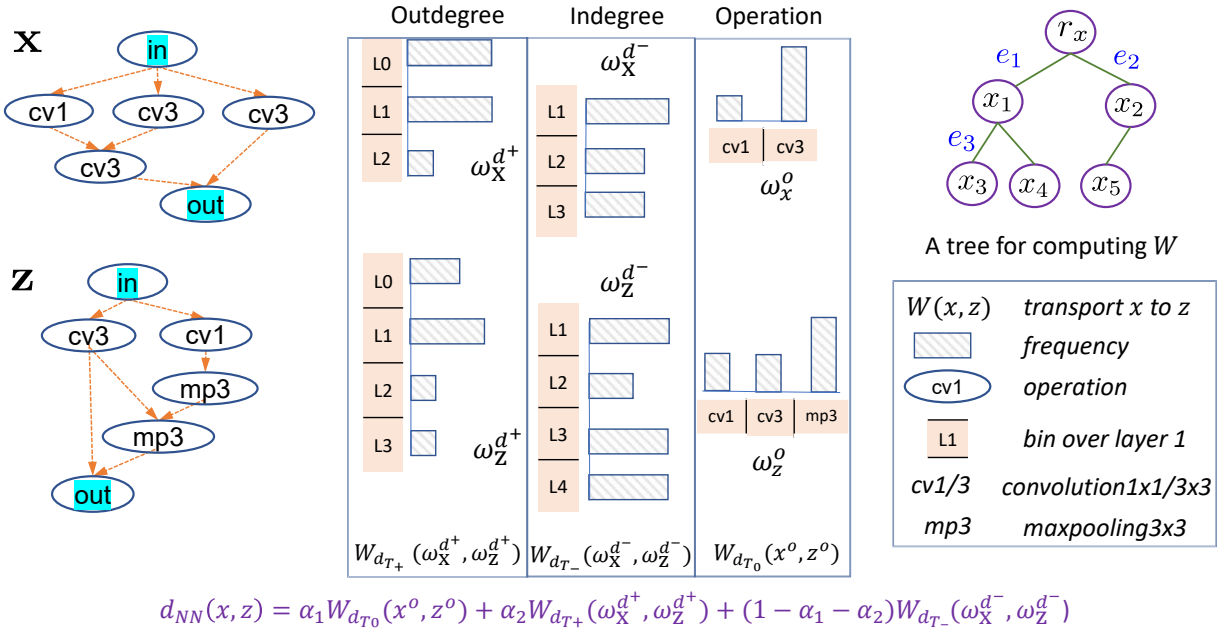
$$k(\mathbf{x}, \mathbf{z}) = \exp\left(-d_{\text{NN}}(\mathbf{x}, \mathbf{z})/\sigma_l^2\right), \tag{6}$$

where the scalar σ_l^2 is the length-scale parameter. Our kernel has three hyperparameters including a length-scale σ_l^2 in Eq. (6); α_1 and α_2 in Eq. (5). These hyperparameters will be estimated by maximizing the log marginal likelihood (see Appendix §G). We refer to the Appendix §H for a further discussion about the properties of the pseudo-distance d_{NN}.

3. Neural Architecture Search with Gaussian Process and k-DPP

Problem setting. We consider a noisy black-box function $f : \mathbb{R}^d \to \mathbb{R}$ over some domain $\mathcal{X}$ containing neural network architectures. As a black-box function, we do not have

[2] Since the tree is a chain, the TW distance is equivalent to the univariate OT.



$$d_{NN}(x,z) = \alpha_1 W_{d_{T_0}}(x^o, z^o) + \alpha_2 W_{d_{T+}}(\omega_{\mathrm{X}}^{d^+}, \omega_{\mathrm{Z}}^{d^+}) + (1-\alpha_1-\alpha_2) W_{d_{T-}}(\omega_{\mathrm{X}}^{d^-}, \omega_{\mathrm{Z}}^{d^-})$$

Figure 1. We represent two architectures $\mathbf{x}$ and $\mathbf{z}$ by network structure (via outdegree and indegree) and network operation (using 1-gram in this example). The similarity between each respective representation is estimated by tree-Wasserstein to compute the minimal cost of transporting one object to another. As a nice property of optimal transport, our tree-Wasserstein can handle *different* layer sizes and *different* operation types. The weights in each histogram are calculated from the architectures. The histogram bins in outdegree and indegree are aligned with the network structure in the left. See the Appendix §F for detailed calculations.

an explicit formulation for f and it is expensive to evaluate. Our goal is to find the best architecture $\mathbf{x}^* \in \mathcal{X}$ such that

$$\mathbf{x}^* = \underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmax}}\, f(\mathbf{x}). \tag{7}$$

We view the black-box function f as a machine learning experiment which takes an input as a neural network architecture $\mathbf{x}$ and produces an accuracy y. We can write $y = f(\mathbf{x}) + \epsilon$ where we have considered Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma_f^2)$ given the noise variance σ_f^2 estimated from the data.

Bayesian optimization (BO) optimizes the black-box function by sequentially evaluating it (Garnett et al., 2010; Shahriari et al., 2016; Nguyen & Osborne, 2020). Particularly, BO can speed up the optimization process by using a probabilistic model to guide the search (Snoek et al., 2012). BO has demonstrated impressive success for optimizing the expensive black-box functions across domains.

Surrogate models. Bayesian optimization reasons about f by building a surrogate model, such as a Gaussian process (GP) (Rasmussen, 2006), Bayesian deep learning (Springenberg et al., 2016) or deep neural network (Snoek et al., 2015; White et al., 2021). Among these choices, GP is the most popular model, offering three key benefits: (i) closed-form uncertainty estimation, (ii) evaluation efficiency, and (iii) learning hyperparameters. GP imposes a normally distributed random variable at every point in the input space. The predictive distribution for a new observation also follows a Gaussian distribution (Rasmussen, 2006) where we can estimate the expected function value $\mu(\mathbf{x})$ and the predictive uncertainty $\sigma(\mathbf{x})$ as

$$\mu(\mathbf{x}') = \mathbf{k}(\mathbf{x}', \mathbf{X}) \left[\boldsymbol{K} + \sigma_f^2 \mathbf{I}\right]^{-1} \mathbf{y} \tag{8}$$

$$\sigma^2(\mathbf{x}') = k_{**} - \mathbf{k}(\mathbf{x}', \mathbf{X}) \left[\boldsymbol{K} + \sigma_f^2 \mathbf{I}\right]^{-1} \mathbf{k}^T(\mathbf{x}', \mathbf{X}) \tag{9}$$

where $\mathbf{X} = [\mathbf{x}_1, ...\mathbf{x}_N]$ and $\mathbf{y} = [y_1, ..y_N]$ are the collected architectures and performances respectively; $K(U, V)$ is a covariance matrix whose element (i, j) is calculated as $k(\mathbf{x}_i, \mathbf{x}_j)$ with $\mathbf{x}_i \in U$ and $\mathbf{x}_j \in V$; $k_{**} = k(\mathbf{x}', \mathbf{x}')$; $\boldsymbol{K} := \mathbf{K}(\mathbf{X}, \mathbf{X})$; σ_f^2 is the measurement noise variance and $\mathbf{I}$ is the identity matrix.

Generating a pool of candidates $\mathcal{P}_t$. We follow Kandasamy et al. (2018); White et al. (2021) to generate a list of candidate networks using an evolutionary algorithm (Back, 1996). First, we stochastically select top-performing candidates with higher acquisition function values. Then, we apply a mutation operator to each candidate to produce modified architectures. Finally, we evaluate the acquisition given these mutations, add them to the initial pool, and repeat for several steps to get a pool of candidates $\mathcal{P}_t$. We design the ablation study in Fig. 2 demonstrating that the evolution

Algorithm 1 Sequential and Parallel NAS using Gaussian process with tree-Wasserstein kernel

Input: Initial data $\mathcal{D}_0$, black-box function $f(\mathbf{x})$. **Output:** The best architecture $\mathbf{x}^*$
for $t = 1, \ldots, T$ **do**
  Generate architecture candidates $\mathcal{P}_t$ by random permutation from the top architectures
  Learn a GP (including hyperparameters) using TW from $\mathcal{D}_{t-1}$ to perform estimation over $\mathcal{P}_t$ including (i) covariance matrix $K_{\mathcal{P}_t}$, (ii) predictive mean $\mu_{\mathcal{P}_t}$ and (iii) predictive variance $\sigma_{\mathcal{P}_t}$
  If **Sequential:** (i) select a next architecture $\mathbf{x}_t = \underset{\forall \mathbf{x} \in \mathcal{P}_t}{\text{argmax}}\ \alpha(\mathbf{x} \mid \mu_{\mathcal{P}_t}, \sigma_{\mathcal{P}_t})$; then (ii) evaluate the new architecture $y_t = f(\mathbf{x}_t)$; after that, (iii) augment $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup (\mathbf{x}_t, y_t)$.
  If **Parallel:** (i) select B architectures $\mathbf{X}_t = [\mathbf{x}_{t,1}, , ...\mathbf{x}_{t,B}] =$ k-DPP($\boldsymbol{K}_{\mathcal{P}_t}$) in Eq. (12); then (ii) evaluate in parallel $Y_t = f(\mathbf{X}_t)$; after that, (iii) augment $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup (\mathbf{X}_t, Y_t)$.
end for

strategy outperforms the random strategy for this task.

Optimizing hyperparameters. We optimize the model hyperparameters by maximizing the log marginal likelihood. We present the derivatives for estimating the hyperparameters α_1 and α_2 of the tree-Wasserstein d_{NN} for neural networks in the Appendix §G. We shall optimize these variables via multi-started gradient descent.

3.1. Sequential NAS using Bayesian optimization

We sequentially suggest a *single* architecture for evaluation using a decision function $\alpha(\mathbf{x})$ (i.e., acquisition function) from the surrogate model. This acquisition function is carefully designed to trade off between exploration of the search space and exploitation of current promising regions. We utilize the GP-UCB (Srinivas et al., 2010) as the main decision function $\alpha(\mathbf{x}) = \mu(\mathbf{x}) + \kappa\sigma(\mathbf{x})$ where κ is the parameter controlling the exploration, μ and σ are the GP predictive mean and variance in Eqs. (8, 9). Empirically, we find that this GP-UCB generally performs better than expected improvement (EI) (see the Appendix §I.1) and other acquisition functions (see (White et al., 2021)). We note that the GP-UCB also comes with a theoretical guarantee for convergence (Srinivas et al., 2010).

We maximize the acquisition function to select the next architecture $\mathbf{x}_{t+1} = \arg\max_{\mathbf{x} \in \mathcal{P}_t} \alpha_t(\mathbf{x})$. This maximization is done on the discrete set of candidate $\mathcal{P}_t$ obtained previously. The selected candidate is the one we expect to be the best if we are optimistic in the presence of uncertainty.

3.2. Parallel NAS using Quality k-DPP and GP

The parallel setting speeds up the optimization process by selecting a *batch* of architectures for parallel evaluations. We present the k-determinantal point process (k-DPP) with *quality* to select from a discrete pool of candidate $\mathcal{P}_t$ for (i) high-performing and (ii) diverse architectures that cover the most information while avoiding redundancy. In addition, diversity is an important property for not being stuck at a local optimal architecture.

The DPP (Kulesza et al., 2012) is an elegant probabilistic measure used to model negative correlations within a subset and hence promote its diversity. A k-determinantal point process (k-DPP) (Kulesza & Taskar, 2011) is a distribution over all subsets of a ground set $\mathcal{P}_t$ of cardinality k. It is determined by a positive semidefinite kernel $\boldsymbol{K}_{\mathcal{P}_t}$. Let $\boldsymbol{K}_A$ be the submatrix of $\boldsymbol{K}_{\mathcal{P}_t}$ consisting of the entries $\boldsymbol{K}_{ij}$ with $i, j \in A \subseteq \mathcal{P}_t$. Then, the probability of observing $A \subseteq P$ is proportional to $\det(\boldsymbol{K}_A)$,

$$P(A \subseteq \mathcal{P}_t) \propto \det(\boldsymbol{K}_A), \quad (10) \qquad \boldsymbol{K}_{ij} = q_i \phi_i^T \phi_j q_j. \quad (11)$$

k-DPP with quality. While the original idea of the k-DPP is to find a diverse subset, we can extend it to find a subset that is both diverse and high-quality. For this, we write a DPP kernel k as a Gram matrix, $\boldsymbol{K} = \Phi^T\Phi$, where the columns of Φ are vectors representing items in the set S. We now take this one step further, writing each column Φ as the product of a quality term $q_i \in \mathcal{R}^+$ and a vector of normalized diversity features ϕ_i, $||\phi_i|| = 1$. The entries of the kernel can now be written in Eq. (11).

As discussed in (Kulesza et al., 2012), this decomposition of $\boldsymbol{K}$ has two main advantages. First, it implicitly enforces the constraint that $\boldsymbol{K}$ must be positive semidefinite, which can potentially simplify learning. Second, it allows us to independently model quality and diversity, and then combine them into a unified model. Particularly, we have

$$P_K(A) \propto \left(\prod_{i \in A} q_i^2\right) \det(\phi_i^T \phi_i),$$

where the first term increases with the quality of the selected items, and the second term increases with the diversity of the selected items. Without the quality component, we would get a very diverse set of architectures. However, we might fail to include the most high-performance architectures in $\mathcal{P}_t$, focusing instead on low-quality outliers. By integrating the two models, we can achieve a more balanced result.

Conditioning. In the parallel setting, given the training data, we would like to select high quality and diverse architectures from a pool of candidate $\mathcal{P}_t$ described above. We shall condition on the training data in constructing the covariance matrix over the testing candidates from $\mathcal{P}_t$. We

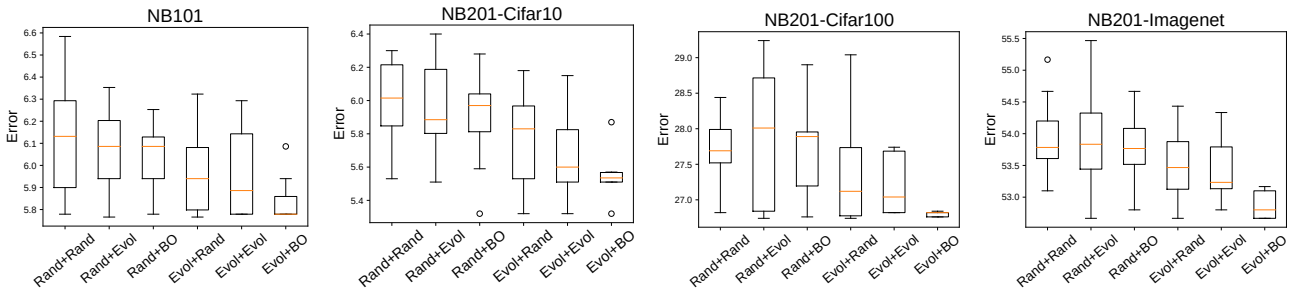


Figure 2. We study the relative contribution of the strategies generating a pool of candidate $\mathcal{P}$ (Rand and Evolution) versus the main optimization algorithm (Rand, Evolution and BO). The result shows that Evolution can help to improve the performance than Rand in generating a pool of candidate $\mathcal{P}$. Given the same strategy for generating $\mathcal{P}$, BO is significantly better than Rand and Evolution for optimization. Evol+BO is the design in our approach that leads to the best performance.

make the following proposition in connecting the k-DPP conditioning and GP uncertainty estimation. This view allows us to learn the covariance matrix using GP, such as we can maximize the GP marginal likelihood for learning the TW distance and kernel hyperparameters for k-DPP.

Proposition 2. *Conditioned on the training set, the probability of selecting new candidates from a pool $\mathcal{P}_t$ is equivalent to the determinant of the Gaussian process predictive covariance matrix.*

Proof of Proposition 2 is placed in the Appendix §G.1.

We can utilize the GP predictive mean $\mu(\cdot)$ in Eq. (8) to estimate the quality for any unknown architecture q_i defined in Eq. (11). Then, we construct the covariance (kernel) matrix over the test candidates for selection by rewriting Eq. (11) as

$$\boldsymbol{K}_{\mathcal{P}_t}(\mathbf{x}_i, \mathbf{x}_j) = \exp\big(-\mu(\mathbf{x}_i)\big)\sigma(\mathbf{x}_i, \mathbf{x}_j)\exp\big(-\mu(\mathbf{x}_j)\big), \quad (12)$$

for all $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{P}_t$ where $\mu(\mathbf{x}_i)$ and $\sigma(\mathbf{x}_i, \mathbf{x}_j)$ are the GP predictive mean and variance defined in Eqs. (8, 9). Finally, we sample B architectures from the covariance matrix $\boldsymbol{K}_{\mathcal{P}_t}$ which encodes both the diversity (exploration) and high-utility (exploitation). The sampling algorithm requires precomputing the eigenvalues (Kulesza & Taskar, 2011). Sampling from a k-DPP requires $\mathcal{O}(NB^2)$ time overall where B is the batch size.

Advantages. The connection between GP and k-DPP allows us to directly sample diverse and high-quality samples from the GP posterior. This leads to the key advantage that we can *optimally* sample a batch of candidates without the need of greedy selection. On the other hand, the existing batch BO approaches rely either on greedy strategy (Contal et al., 2013; Desautels et al., 2014; González et al., 2016) to sequentially select the points in a batch or independent sampling (Falkner et al., 2018; Hernández-Lobato et al., 2017). The greedy algorithm is non-optimal and the independent sampling approaches can not fully utilize the information across points in a batch.

We note that our k-DPP above is related to (Kathuria et al., 2016), but different from two perspectives that Kathuria et al. (2016) considers k-DPP for batch BO (i) in the continuous setting and (ii) using pure exploration (without quality). We will consider this as the baseline in our experiments.

4. Experiments

We evaluate our proposed approach on both sequential and parallel neural architecture search (NAS).

Experimental settings. All experimental results are averaged over 30 independent runs with different random seeds. We set the number of candidate architecture $|\mathcal{P}_t| = 100$. We utilize the popular NAS tabular datasets of Nasbench101 (NB101) (Ying et al., 2019) and Nasbench201 (NB201) (Dong & Yang, 2020) for evaluations. TW and TW-2G stand for our TW using 1-gram and 2-gram representation, respectively. We release the Python code for our experiments at `https://github.com/ntienvu/TW_NAS`.

4.1. Sequential NAS

Ablation study: different mechanisms for generating a pool of candidates $\mathcal{P}$. We analyze the relative contribution of the process of generating architecture candidates versus the main optimization algorithm in Fig. 2. The result suggests that the evolutionary algorithm is better than a random strategy to generate a pool of candidates $\mathcal{P}$. Given this generated candidate set $\mathcal{P}$, BO is significantly better than Rand and Evolution approaches. Briefly, the combination of Evol+BO performs the best across datasets.

Ablation study: different distances for BO. We design an ablation study using different distances within a BO framework. Notably, we consider the vanilla optimal transport (Wasserstein distance) in which we follow Kandasamy et al. (2018) to define the cost metric for OT. This baseline can be seen as the modified version of the Nasbot (Kandasamy et al., 2018). In addition, we compare our approach with the BO using the Gromov-Wasserstein distance (Mémoli,

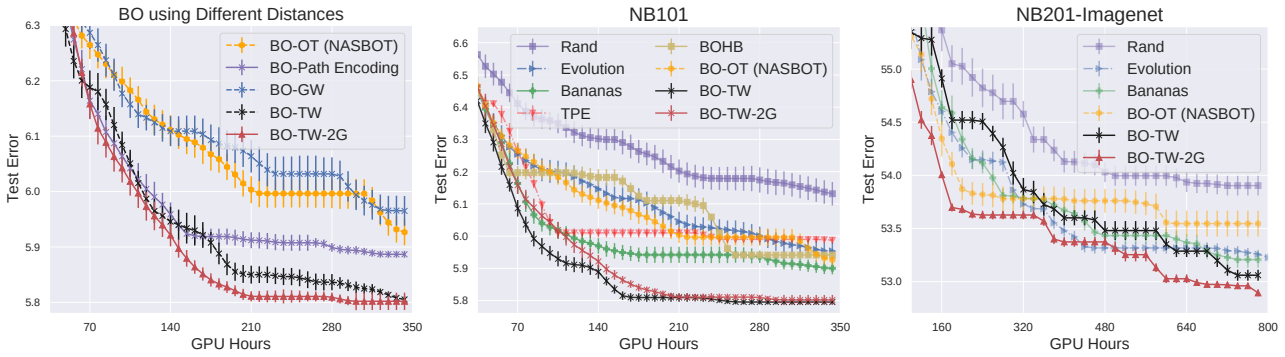


Figure 3. Sequential NAS on different distances for BO (*left*) and different baselines (*middle* and *right*). Our approaches of BO-TW (**black** curve) and BO-TW 2G (red curve) for 1-gram and 2-gram representation consistently outperform the other baselines. We use 500 iterations on NB101 and 200 iterations on NB201.

2011) (BO-GW) and path encoding (BO-Path Encoding) as used in (White et al., 2021). The results in the left plot of Fig. 3 suggest that the proposed TW using 2-gram performs the best among the BO distance for NAS. The standard OT and GW will result in (non-p.s.d.) indefinite kernels. For using OT and GW in our GP, we keep adding ("jitter") noise to the diagonal of the kernel matrices until they become p.s.d. kernels. We make use of the POT library (Flamary & Courty, 2017) for the implementation of OT and GW.

While our framework is able to handle n-gram representation, we learn that 2-gram is empirically the best choice. This choice is well supported by the fact that two convolution layers of 3×3 stay together can be used to represent for a special effect of 5×5 convolution kernel. In addition, the use of full n-gram may result in very sparse representation and some features are not so meaningful anymore. Therefore, in the experiment we only consider 1-gram and 2-gram.

Sequential NAS. We validate our GP-BO model using tree-Wasserstein on the sequential setting. Since NB101 is somewhat harder than NB201, we allocate 500 queries for NB101 and 200 queries for NB201 including 10% of random selection at the beginning of BO.

We compare our approach against common baselines including Random search, evolutionary search, TPE (Bergstra et al., 2011), BOHB (Falkner et al., 2018), Nasbot (Kandasamy et al., 2018) and Bananas (White et al., 2021). We use the AutoML library for TPE and BOHB[3] including the results for NB101, but not NB201. We do not compare with Reinforcement Learning approaches (Pham et al., 2018) and AlphaX (Wang et al., 2020) which have been shown to perform poorly in (White et al., 2021).

We show in Fig. 3 that our tree-Wasserstein including 1-gram and 2-gram will result in the best performance with a wide margin to the second best – the Bananas (White et al., 2021), which needs to specify a meta neural network with extra hyperparameters (layers, nodes, learning rate). Random search performs poorly in NAS due to the high-dimensional and complex space. Our GP-based optimizer offers a closed-form uncertainty estimation without iterative approximation in neural network (via back-propagation). As a property of GP, our BO-TW can generalize well using fewer observations. This can be seen in the right plot of Fig. 3 that our approaches can outperform Bananas when the number of BO iteration (or the number of network architectures for training) is small. On the other hand, both Bananas and ours are converging to the same performance when the training data becomes abundant – but this is not the case in practice for NAS.

[3] https://github.com/automl/nas_benchmarks

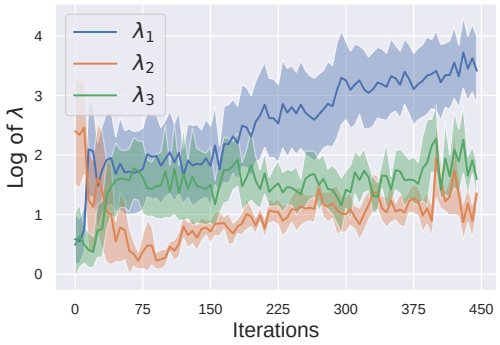


Figure 4. Estimated hyperparameters on NB101.

Estimating hyperparameters. We plot the estimated hyperparameters

$$\lambda_1 = \frac{\alpha_1}{\sigma_l^2}, \quad \lambda_2 = \frac{\alpha_2}{\sigma_l^2}, \quad \lambda_3 = \frac{1-\alpha_1-\alpha_2}{\sigma_l^2},$$

over iterations in Fig. 4. This indicates the relative contribution of the operation, indegree and outdegree toward the TW d_{NN} for neural networks in Eq. (5). Particularly, the operation contributes receives higher weight and is useful information than either the individual indegree or outdegree.

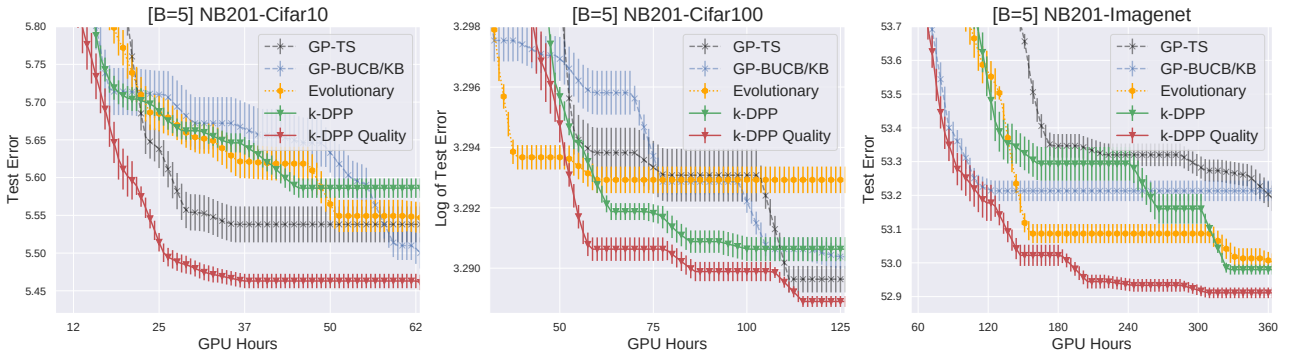


Figure 5. Batch NAS comparison using TW-2Gram and a batch size $B = 5$. Our proposed k-DPP Quality (red) outperforms other baselines in all cases, especially when the number of training architecture (iterations) is low. This is the desirable property of NAS when the training cost is extremely expensive. The experiments are run over 100 iterations.

4.2. Parallel NAS

We next demonstrate our model on selecting multiple architectures for parallel evaluation (i.e., parallel NAS) setting. There are fewer approaches for parallel NAS compared to the sequential setting. We select to compare our k-DPP quality against Thompson sampling (Hernández-Lobato et al., 2017), GP-BUCB (Desautels et al., 2014) and k-DPP for batch BO (Kathuria et al., 2016). The GP-BUCB is equivalent to Kriging believer (Ginsbourger et al., 2010) when the hallucinated observation value is set to the GP predictive mean. Therefore, we label them as GP-BUCB/KB. We also compare with the vanilla k-DPP (without using quality) (Kathuria et al., 2016).

We allocate a maximum budget of 500 queries including 50 random initial architectures. The result in Fig. 5 shows that our proposed k-DPP quality is the best among the considered approaches. We refer to the Appendix for additional experiments including varying batch sizes and more results on NB201.

Our sampling from k-DPP quality is advantageous against the existing batch BO approaches (Ginsbourger et al., 2010; Desautels et al., 2014; Kathuria et al., 2016; Hernández-Lobato et al., 2017) in which we can optimally select a batch of architectures without relying on the greedy selection strategy. In addition, our k-DPP quality can leverage the benefit of the GP in estimating the hyperparameters for the covariance matrix.

Ablation study of k-DPP quality with path distance. In addition to the proposed tree-Wasserstein, we demonstrate the proposed k-DPP quality using path distance (White et al., 2021). We show that our k-DPP quality is not restricted to TW-2G, but it can be generally used with different choices of kernel distances.

Particularly, we present in Fig. 6 the comparison using two datasets: Imagenet and Cifar100 in NB201. The results validate two following messages. First, our k-DPP quality is the best among the other baselines in selecting high-performing and diverse architectures. Second, our k-DPP quality with TW2G (a black curve) performs better than k-DPP quality using Path distance (a red curve). This demonstrates the key benefits of comparing two complex architectures as logistical supplier and receiver.

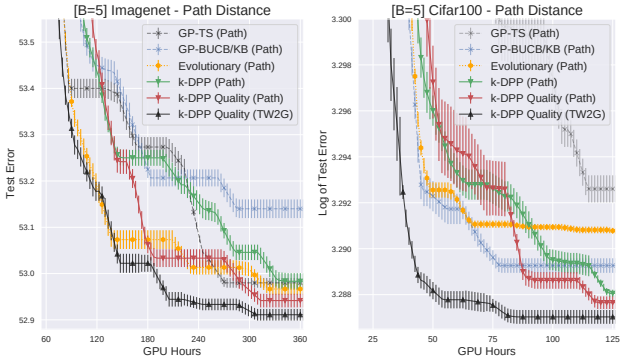


Figure 6. We compare different batch approaches using a path distance (White et al., 2021) and a batch size $B = 5$. We show that (i) the k-DPP quality outperforms the other batch approaches and (ii) the k-DPP using TW2G (a black curve) performs better than using path distance (a red curve).

5. Conclusion

We have presented a new framework for sequential and parallel NAS. Our proposed framework constructs the similarity between architectures using tree-Wasserstein geometry. Then, it utilizes the Gaussian process surrogate for modeling and optimization. We draw the connection between GP predictive distribution to k-DPP quality for selecting diverse and high-performing architectures from discrete set. We demonstrate our model using Nasbench101 and Nasbench201 that our methods outperform the existing baselines in sequential and parallel settings.

Acknowledgements

We thank anonymous reviewers and area chairs for their comments to improve the paper. VN would like to thank NVIDIA for sponsoring GPU hardware and Google Cloud Platform for sponsoring computing resources in this project. TL acknowledges the support of JSPS KAKENHI Grant number 20K19873.

References

Back, T. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996. (Cited on page 4)

Baker, B., Gupta, O., Naik, N., and Raskar, R. Designing neural network architectures using reinforcement learning. *International Conference on Learning Representations*, 2017. (Cited on page 13)

Berg, C., Christensen, J. P. R., and Ressel, P. *Harmonic analysis on semigroups*. Springer-Verlag, 1984. (Cited on pages 3 and 15)

Bergstra, J. S., Bardenet, R., Bengio, Y., and Kégl, B. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pp. 2546–2554, 2011. (Cited on page 7)

Burkard, R. E. and Cela, E. Linear assignment problems and extensions. In *Handbook of combinatorial optimization*, pp. 75–149. Springer, 1999. (Cited on page 21)

Contal, E., Buffoni, D., Robicquet, A., and Vayatis, N. Parallel gaussian process optimization with upper confidence bound and pure exploration. In *Machine Learning and Knowledge Discovery in Databases*, pp. 225–240. Springer, 2013. (Cited on page 6)

Desautels, T., Krause, A., and Burdick, J. W. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *The Journal of Machine Learning Research*, 15(1):3873–3923, 2014. (Cited on pages 1, 6, 8, and 14)

Do Ba, K., Nguyen, H. L., Nguyen, H. N., and Rubinfeld, R. Sublinear time algorithms for Earth Mover's distance. *Theory of Computing Systems*, 48(2):428–442, 2011. (Cited on pages 2 and 15)

Dong, X. and Yang, Y. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1761–1770, 2019. (Cited on pages 1 and 13)

Dong, X. and Yang, Y. Nas-bench-201: Extending the scope of reproducible neural architecture search. *International Conference on Learning Representation*, 2020. (Cited on page 6)

Elsken, T., Metzen, J. H., and Hutter, F. Efficient multi-objective neural architecture search via lamarckian evolution. *International Conference on Learning Representation*, 2019a. (Cited on pages 1 and 13)

Elsken, T., Metzen, J. H., and Hutter, F. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019b. (Cited on pages 1 and 13)

Falkner, S., Klein, A., and Hutter, F. Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pp. 1436–1445, 2018. (Cited on pages 6, 7, and 14)

Flamary, R. and Courty, N. Pot python optimal transport library. *GitHub: https://github. com/rflamary/POT*, 2017. (Cited on page 7)

Frazier, P. I. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018. (Cited on page 13)

Gao, X., Xiao, B., Tao, D., and Li, X. A survey of graph edit distance. *Pattern Analysis and applications*, 13(1): 113–129, 2010. (Cited on page 14)

Garnett, R., Osborne, M. A., and Roberts, S. J. Bayesian optimization for sensor set selection. In *Proceedings of the 9th ACM/IEEE international conference on information processing in sensor networks*, pp. 209–219, 2010. (Cited on page 4)

Ginsbourger, D., Le Riche, R., and Carraro, L. Kriging is well-suited to parallelize optimization. In *Computational Intelligence in Expensive Optimization Problems*, pp. 131–162. Springer, 2010. (Cited on pages 8 and 14)

González, J., Dai, Z., Hennig, P., and Lawrence, N. D. Batch Bayesian optimization via local penalization. In *International Conference on Artificial Intelligence and Statistics*, pp. 648–657, 2016. (Cited on pages 1 and 6)

Gopakumar, S., Gupta, S., Rana, S., Nguyen, V., and Venkatesh, S. Algorithmic assurance: An active approach to algorithmic testing using Bayesian optimisation. In *Advances in Neural Information Processing Systems*, pp. 5465–5473, 2018. (Cited on page 13)

Hernández-Lobato, J. M., Requeima, J., Pyzer-Knapp, E. O., and Aspuru-Guzik, A. Parallel and distributed Thompson sampling for large-scale accelerated exploration of chemical space. *In International Conference on Machine Learning*, pp. 1470–1479, 2017. (Cited on pages 6, 8, and 14)

Jin, H., Song, Q., and Hu, X. Auto-keras: Efficient neural architecture search with network morphism. 2018. (Cited on page 13)

Kandasamy, K., Neiswanger, W., Schneider, J., Poczos, B., and Xing, E. P. Neural architecture search with bayesian optimisation and optimal transport. In *Advances in Neural Information Processing Systems*, pp. 2016–2025, 2018. (Cited on pages 1, 2, 3, 4, 6, 7, and 13)

Kathuria, T., Deshpande, A., and Kohli, P. Batched Gaussian process bandit optimization via determinantal point processes. In *Advances in Neural Information Processing Systems*, pp. 4206–4214, 2016. (Cited on pages 6 and 8)

Kondor, R. and Lafferty, J. Diffusion kernels on graphs and other discrete input spaces. In *International Conference on Machine Learning*, pp. 315–322, 2002. (Cited on page 14)

Kulesza, A. and Taskar, B. k-dpps: Fixed-size determinantal point processes. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 1193–1200, 2011. (Cited on pages 5 and 6)

Kulesza, A., Taskar, B., et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012. (Cited on page 5)

Le, T. and Nguyen, T. Entropy partial transport with tree metrics: Theory and practice. In *International Conference on Artificial Intelligence and Statistics*, pp. 3835–3843, 2021. (Cited on page 16)

Le, T., Huynh, V., Ho, N., Phung, D., and Yamada, M. On scalable variant of wasserstein barycenter. *arXiv preprint arXiv:1910.04483*, 2019a. (Cited on page 16)

Le, T., Yamada, M., Fukumizu, K., and Cuturi, M. Tree-sliced variants of Wasserstein distances. In *Advances in Neural Information Processing Systems*, pp. 12283–12294, 2019b. (Cited on pages 2, 3, 15, and 21)

Le, T., Ho, N., and Yamada, M. Flow-based alignment approaches for probability measures in different spaces. In *International Conference on Artificial Intelligence and Statistics*, pp. 3934–3942, 2021. (Cited on page 16)

Li, L. and Jamieson, K. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18:1–52, 2018. (Cited on page 13)

Liu, H., Simonyan, K., Vinyals, O., Fernando, C., and Kavukcuoglu, K. Hierarchical representations for efficient architecture search. *International Conference on Learning Representation*, 2018. (Cited on pages 1 and 13)

Liu, H., Simonyan, K., and Yang, Y. Darts: Differentiable architecture search. *International Conference on Learning Representation*, 2019. (Cited on pages 1 and 13)

Luo, R., Tian, F., Qin, T., Chen, E., and Liu, T.-Y. Neural architecture optimization. In *Advances in Neural Information Processing Systems*, pp. 7816–7827, 2018. (Cited on pages 1 and 13)

Mémoli, F. Gromov–wasserstein distances and the metric approach to object matching. *Foundations of computational mathematics*, 11(4):417–487, 2011. (Cited on page 6)

Mémoli, F., Smith, Z., and Wan, Z. Gromov-hausdorff distances on p-metric spaces and ultrametric spaces. *arXiv preprint arXiv:1912.00564*, 2019. (Cited on page 16)

Mémoli, F., Munk, A., Wan, Z., and Weitkamp, C. The ultrametric gromov-wasserstein distance. *arXiv preprint arXiv:2101.05756*, 2021. (Cited on page 16)

Messmer, B. T. and Bunke, H. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE transactions on pattern analysis and machine intelligence*, 20(5): 493–504, 1998. (Cited on page 14)

Nguyen, V. and Osborne, M. A. Knowing the what but not the where in Bayesian optimization. In *International Conference on Machine Learning*, pp. 7317–7326. PMLR, 2020. (Cited on page 4)

Nguyen, V., Rana, S., Gupta, S. K., Li, C., and Venkatesh, S. Budgeted batch Bayesian optimization. In *16th International Conference on Data Mining (ICDM)*, pp. 1107–1112, 2016. (Cited on page 14)

Nguyen, V., Schulze, S., and Osborne, M. A. Bayesian optimization for iterative learning. In *Advances in Neural Information Processing Systems*, 2020. (Cited on page 13)

Parker-Holder, J., Nguyen, V., and Roberts, S. J. Provably efficient online hyperparameter optimization with population-based bandits. *Advances in Neural Information Processing Systems*, 33, 2020. (Cited on page 13)

Peyré, G. and Cuturi, M. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5-6): 355–607, 2019. (Cited on pages 1 and 2)

Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, pp. 4095–4104, 2018. (Cited on pages 7 and 13)

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In *Advances in Neural Information*

Processing Systems, pp. 1177–1184, 2007. (Cited on page 14)

Rana, S., Li, C., Gupta, S., Nguyen, V., and Venkatesh, S. High dimensional Bayesian optimization with elastic Gaussian process. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 2883–2891, 2017. (Cited on page 13)

Rasmussen, C. E. Gaussian processes for machine learning. 2006. (Cited on pages 1, 4, and 20)

Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., Le, Q. V., and Kurakin, A. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 2902–2911, 2017. (Cited on pages 1 and 13)

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 33, pp. 4780–4789, 2019. (Cited on pages 1, 13, and 14)

Ru, B., Alvi, A. S., Nguyen, V., Osborne, M. A., and Roberts, S. J. Bayesian optimisation over multiple continuous and categorical inputs. In *International Conference on Machine Learning*, 2020. (Cited on page 13)

Ru, B., Wan, X., Dong, X., and Osborne, M. A. Interpretable neural architecture search via bayesian optimisation with weisfeiler-lehman kernels. *International Conference on Learning Representation*, 2021. (Cited on page 14)

Sciuto, C., Yu, K., Jaggi, M., Musat, C., and Salzmann, M. Evaluating the search phase of neural architecture search. *arXiv preprint arXiv:1902.08142*, 2019. (Cited on page 13)

Semple, C. and Steel, M. Phylogenetics. *Oxford Lecture Series in Mathematics and its Applications*, 2003. (Cited on page 2)

Shah, S. A. R., Wu, W., Lu, Q., Zhang, L., Sasidharan, S., DeMar, P., Guok, C., Macauley, J., Pouyoul, E., Kim, J., et al. Amoebanet: An sdn-enabled network service for big data science. *Journal of Network and Computer Applications*, 119:70–82, 2018. (Cited on pages 1 and 13)

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and de Freitas, N. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104 (1):148–175, 2016. (Cited on pages 1 and 4)

Smola, A. J. and Kondor, R. Kernels and regularization on graphs. In *Learning theory and kernel machines*, pp. 144–158. Springer, 2003. (Cited on page 14)

Snoek, J., Larochelle, H., and Adams, R. P. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pp. 2951–2959, 2012. (Cited on pages 1 and 4)

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M., Prabhat, M., and Adams, R. Scalable Bayesian optimization using deep neural networks. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 2171–2180, 2015. (Cited on page 4)

Springenberg, J. T., Klein, A., Falkner, S., and Hutter, F. Bayesian optimization with robust bayesian neural networks. In *Advances in Neural Information Processing Systems*, pp. 4134–4142, 2016. (Cited on page 4)

Srinivas, N., Krause, A., Kakade, S., and Seeger, M. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 1015–1022, 2010. (Cited on page 5)

Suganuma, M., Shirakawa, S., and Nagao, T. A genetic programming approach to designing convolutional neural network architectures. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 497–504, 2017. (Cited on pages 1 and 13)

Villani, C. *Topics in optimal transportation*. American Mathematical Soc., 2003. (Cited on page 2)

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., and Borgwardt, K. M. Graph kernels. *Journal of Machine Learning Research*, 11(Apr):1201–1242, 2010. (Cited on page 14)

Wallis, W. D., Shoubridge, P., Kraetz, M., and Ray, D. Graph distances using graph union. *Pattern Recognition Letters*, 22(6-7):701–704, 2001. (Cited on page 14)

Wan, X., Nguyen, V., Ha, H., Ru, B., Lu, C., and Osborne, M. A. Think global and act local: Bayesian optimisation over high-dimensional categorical and mixed search spaces. In *International Conference on Machine Learning*, 2021. (Cited on page 13)

Wang, L., Zhao, Y., Jinnai, Y., Tian, Y., and Fonseca, R. Alphax: exploring neural architectures with deep neural networks and monte carlo tree search. *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020. (Cited on pages 7 and 14)

Wang, Z., Gehring, C., Kohli, P., and Jegelka, S. Batched large-scale Bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics*, pp. 745–754, 2018. (Cited on page 2)

White, C., Neiswanger, W., and Savani, Y. Bananas: Bayesian optimization with neural architectures for neural architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021. (Cited on pages 3, 4, 5, 7, 8, 13, 14, and 22)

Xie, L. and Yuille, A. Genetic cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1379–1388, 2017. (Cited on pages 1 and 13)

Xie, S., Zheng, H., Liu, C., and Lin, L. Snas: stochastic neural architecture search. *International Conference on Learning Representation*, 2019. (Cited on page 13)

Yao, Q., Xu, J., Tu, W.-W., and Zhu, Z. Efficient neural architecture search via proximal iterations. In *AAAI Conference on Artificial Intelligence*, 2020. (Cited on pages 1 and 13)

Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., and Hutter, F. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, pp. 7105–7114, 2019. (Cited on pages 6 and 14)

Zhong, Z., Yan, J., Wu, W., Shao, J., and Liu, C.-L. Practical block-wise neural network architecture generation. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2423–2432, 2018. (Cited on page 13)

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. *International Conference on Representation Learning*, 2017. (Cited on page 13)