Optimal Transport Kernels for Sequential and Parallel Neural Architecture Search

Vu Nguyen∗1  Tam Le∗2  Makoto Yamada2 3  Michael A. Osborne4

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; Luo et al., 2018; 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 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; https://www.automl.org/automl/literature-on-neural-architecture-search

---

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

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).
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 \( (\Omega, d) \) be a measurable metric space. For any \( x \in \Omega \), we use \( \delta_x \) 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 \( \omega, \nu \) be Borel probability distributions on \( \Omega \) and \( \mathcal{R}(\omega, \nu) \) be the set of probability distributions \( \pi \) 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 \( \omega \) and \( \nu \) is defined as:

\[
W_d(\omega, \nu) = \inf_{\pi \in \mathcal{R}(\omega, \nu)} \int_{\Omega \times \Omega} d(x, z) \pi(dx, dz),
\]

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 \( T \) with positive edge lengths such that \( \forall x, y \in \Omega \), then \( x \) is a node of \( T \); and \( \forall x, y \in \Omega \), \( d(x, y) \) is equal to the length of the (unique) path between \( x \) and \( y \) (Semple & Steel, 2003) (§7, p.145–182).

Let \( d_T \) be the tree metric on tree \( T \) rooted at \( r \). For \( x, y \in T \), we denote \( \mathcal{P}(x, y) \) as the (unique) path between \( x \) and \( y \). We write \( \Gamma(x) \) for a set of nodes in the subtree of \( T \) rooted at \( x \), let \( \Gamma(x) = \{ z \in T \mid z \in \mathcal{P}(r, x) \} \). For edge \( e \) in \( T \), 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 \( \omega, \nu \) supported on tree \( T \), and setting the tree metric \( d_T \) as the ground metric, then the TW distance \( W_{d_T} \) between \( \omega \) and \( \nu \) admits a closed-form solution as follows:

\[
W_{d_T}(\omega, \nu) = \sum_{e \in T} w_e |\omega(\Gamma(v_e)) - \nu(\Gamma(v_e))|,
\]

where \( \omega(\Gamma(v_e)) \) is the total mass of the probability measure \( \omega \) 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 \( W_{d_T} \) (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 \( x \) by \( (S^o, A) \) where \( S^o \) is a multi-set of operations in each layer of \( x \), and \( A \) is an adjacency matrix, representing the connection among these layers (i.e., network structure) in \( x \). 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 \( S^o \) 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 \( x^o \) for a neural network \( x \).

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

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

\[
\omega_x^o = \sum_{s \in S^n} x_s^o \delta_s, \quad \omega_z^o = \sum_{s \in S^n} x_s^o \delta_s, \quad \tag{3}
\]

where \( x_s^o \) and \( z_s^o \) are the frequency of \( n \)-gram operation \( s \in S^n \) in architecture \( x \) and \( z \), respectively.

We can leverage the TW distance to compare the \( n \)-gram representations \( \omega_x^o \) and \( \omega_z^o \) using Eq. (2), denoted as \( W_{d_{\mathcal{T}}} (\omega_x^o, \omega_z^o) \). 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 \texttt{conv1} and \texttt{conv3} in the same group and \texttt{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 \( L_x \) as the set of all layers which one can reach from the input layer for neural network \( x \). Let \( n_{x, \ell} \) and \( M_{x, \ell} \) be lengths of the longest paths from an input layer to the layer \( \ell \) 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 \( x \) as empirical measures \( \omega_x^{d^{-}} \) and \( \omega_x^{d^{+}} \), defined as

\[
\omega_x^{d^{-}} = \sum_{\ell \in L_x} x_{\ell}^{d^{-}} \delta_{n_{x, \ell + 1}} \quad \omega_x^{d^{+}} = \sum_{\ell \in L_x} x_{\ell}^{d^{+}} \delta_{M_{x, \ell + 1}} \quad \tag{4}
\]

where \( x_{\ell}^{d^{-}} \) and \( x_{\ell}^{d^{+}} \) are the normalized indegree and outdegree of the layer \( \ell \) of \( x \) respectively.

For indegree and outdegree information, the supports of empirical measures \( \omega_x^{d^{-}} \), and \( \omega_x^{d^{+}} \) are in one-dimensional space that a tree structure reduces to a chain of supports. Thus, we can use \( W_{d_{\mathcal{T}}} (\omega_x^{d^{-}}, \omega_z^{d^{-}}) \) to compare those empirical measures.

Similarly, we have \( W_{d_{\mathcal{T}}} (\omega_x^{d^{+}}, \omega_z^{d^{+}}) \) for empirical measures \( \omega_x^{d^{+}} \) and \( \omega_z^{d^{+}} \) built from outdegree information.

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

\[
d_{\mathcal{NN}} (x, z) = \alpha_1 W_{d_{\mathcal{T}}} (x^o, z^o) + \alpha_2 W_{d_{\mathcal{T}}} (\omega_x^{d^{-}}, \omega_z^{d^{-}}) + (1 - \alpha_1 - \alpha_2) W_{d_{\mathcal{T}}} (\omega_x^{d^{+}}, \omega_z^{d^{+}}). \quad \tag{5}
\]

The proposed discrepancy \( d_{\mathcal{NN}} \) 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 \( d_{\mathcal{NN}} \): it can compare two architectures with different layer sizes and/or operations sizes.

**Proposition 1.** The \( d_{\mathcal{NN}} \) for neural networks is a pseudometric and negative definite.

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

Our discrepancy \( d_{\mathcal{NN}} \) 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 \( d_{\mathcal{NN}} \) for neural networks \( x, z \) as

\[
k(x, z) = \exp \left( -d_{\mathcal{NN}} (x, z) / \sigma^2 \right), \quad \tag{6}
\]

where the scalar \( \sigma^2 \) is the length-scale parameter. Our kernel has three hyperparameters including a length-scale \( \sigma^2 \) in Eq. (6); \( \alpha_1 \) and \( \alpha_2 \) 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 \( d_{\mathcal{NN}} \).

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

**Problem setting.** We consider a noisy black-box function \( f : \mathbb{R}^d \rightarrow \mathbb{R} \) over some domain \( \mathcal{X} \) containing neural network architectures. As a black-box function, we do not have since the tree is a chain, the TW distance is equivalent to the univariate OT.
Figure 1. We represent two architectures \( x \) and \( z \) 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 \( x^* \in X \) such that

\[
x^* = \arg \max_{x \in X} f(x).
\]

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

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

\[
\sigma^2(x') = k_{xx'} - k(x', X) \left[ K + \sigma_f^2 I \right]^{-1} k^T(x', X)
\]

where \( X = [x_1, \ldots, x_N] \) and \( y = [y_1, \ldots, y_N] \) are the collected architectures and performances respectively; \( K(U, V) \) is a covariance matrix whose element \( (i, j) \) is calculated as \( k(x_i, x_j) \) with \( x_i \in U \) and \( x_j \in V \); \( k_{xx'} = k(x', x') \); \( K := K(X, X) \); \( \sigma_f^2 \) is the measurement noise variance and \( I \) is the identity matrix.

**Generating a pool of candidates \( \mathcal{P}_t \).** We follow Kundasamy 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
We present the derivatives for estimating the hyperparameters via multi-started gradient descent. The best 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 $\alpha$ and $\sigma^2$ of the tree-Wasserstein $d_{WN}$ 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 $\alpha(x)$ (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 $\alpha(x) = \mu(x) + \kappa \sigma(x)$ where $\kappa$ is the parameter controlling the exploration, $\mu$ and $\sigma$ 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 $x_{t+1} = \text{arg} \max_{x \in \mathcal{P}_t} \alpha_t(x)$. 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 $K_{\mathcal{P}_t}$. Let $K_A$ be the submatrix of $K_{\mathcal{P}_t}$ consisting of the entries $K_{ij}$ with $i, j \in A \subseteq \mathcal{P}_t$. Then, the probability of observing $A \subseteq P$ is proportional to $\det(K_A)$,

$P(A \subseteq P) \propto \det(K_A), \quad (10) \quad K_{ij} = q_i \phi_i^T \phi_j q_j, \quad (11)$

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, $K = \Phi^T \Phi$, where the columns of $\Phi$ are vectors representing items in the set $S$. We now take this one step further, writing each column $\Phi_i$ as the product of a quality term $q_i \in \mathcal{R}^+$ and a vector of normalized diversity features $\phi_i$, $||\phi_i|| = 1$. The entries of the kernel can now be written in Eq. (11).

As discussed in (Kulesza et al., 2012), this decomposition of $K$ has two main advantages. First, it implicitly enforces the constraint that $K$ 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
5.4 27.0 53.0
5.6 27.5 53.5
5.9 5.6 53.5
6.1 6.2 54.5
6.2 6.3 28.5
6.3 55.0 6.4
6.4 29.0 55.5
6.5 29.0 6.6
6.6 Error

Figure 2. We study the relative contribution of the strategies generating a pool of candidate $P$ (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 $P$. Given the same strategy for generating $P$, BO is significantly better than Rand and Evolution for optimization. Evol+BO is the design in our approach that leads to the best performance.

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 $|P_t| = 100$. 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 $P$. 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 $P$. Given this generated candidate set $P$, 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, 2016).
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\(^1\) including the BO-OT (NASBOT) 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 \( 3 \times 3 \) stay together can be used to represent for a special effect of \( 5 \times 5 \) 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\(^1\) 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.

Estimating hyperparameters. We plot the estimated hyperparameters

\[
\lambda_1 = \frac{\alpha_1}{\sigma_1^2}, \quad \lambda_2 = \frac{\alpha_2}{\sigma_1^2}, \quad \lambda_3 = \frac{1 - \alpha_1 - \alpha_2}{\sigma_1^2},
\]

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

---

\(^1\)https://github.com/automl/nas_benchmarks
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 using Path distance (a red curve). This demonstrates the key benefits of comparing two complex architectures as logistical supplier and receiver.

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


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


Optimal Transport Kernels for Sequential and Parallel Neural Architecture Search


Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In Advances in Neural Information...
Optimal Transport Kernels for Sequential and Parallel Neural Architecture Search


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


Optimal Transport Kernels for Sequential and Parallel Neural Architecture Search


