Supplementary Material for: Optimal Transport Kernels for Sequential and Parallel Neural Architecture Search

In this supplementary, we first review the related works in sequential and batch neural architecture search in §A and §B respectively. We next give further detailed information about the datasets in our experiments in §C. Then, we derive detailed proofs for the results in the main text in §D. After that, we give a brief discussion about the tree-Wasserstein geometry in §E, and follow with an example of TW computation for neural network architectures in §F. In §G, we provide additional details of the Bayesian optimization in estimating the hyperparameters. In §H, we show distance properties comparison. Finally, we illustrate further empirical comparisons and analysis for the model in §I.

A. Related works in Neural architecture search

We refer the interested readers to the survey paper (Elsken et al., 2019b) and the literature of neural architecture search[4] for the comprehensive survey on neural architecture search. Many different search strategies have been attempted to explore the space of neural architectures, including random search, evolutionary methods, reinforcement learning (RL), gradient-based methods and Bayesian optimization.

Evolutionary approaches. Real et al. (2017; 2019); Suganuma et al. (2017); Liu et al. (2018); Shah et al. (2018); Xie & Yuille (2017); Elsken et al. (2019a) have been extensively used for NAS. In the context of evolutionary, the mutation operations include adding a layer, removing a layer or changing the type of a layer (e.g., from convolution to pooling) from the neural network architecture. Then, the evolutionary approaches will update the population, e.g., tournament selection by removing the worst or oldest individual from a population.

Reinforcement learning. NASNet (Zoph & Le, 2017) is a reinforcement learning algorithm for NAS which achieves state-of-the-art results on CIFAR-10 and PTB; however, the algorithm requires 3000 GPU days to train. Efficient Neural Architecture Search (ENAS) (Pham et al., 2018) proposes to use a controller that discovers architectures by learning to search for an optimal subgraph within a large graph. The controller is trained with policy gradient to select a subgraph that maximizes the validation set's expected reward. The model corresponding to the subgraph is trained to minimize a canonical cross-entropy loss. Multiple child models share parameters, ENAS requires fewer GPU-hours than other approaches and 1000-fold less than "standard" NAS. Other reinforcement learning approaches for NAS have also proposed, such as MetaQNN (Baker et al., 2017) and BlockQNN (Zhong et al., 2018).

Gradient-based approaches. The works presented in Luo et al. (2018); Liu et al. (2019); Dong & Yang (2019); Yao et al. (2020) represent the search space as a directed acyclic graph (DAG) containing billions of sub-graphs, each of which indicates a kind of neural architecture. To avoid traversing all the possibilities of the sub-graphs, they develop a differentiable sampler over the DAG. The benefit of such an idea is that a differentiable space enables computation of gradient information, which could speed up the convergence of underneath optimization algorithm. Various techniques have been proposed, e.g., DARTS (Liu et al., 2019) , SNAS (Xie et al., 2019) , and NAO (Luo et al., 2018). While these approaches based on gradient-based learning can reduce the computational resources required for NAS, it is currently not well understood if an initial bias in exploring certain parts of the search space more than others might lead to the bias and thus result in premature convergence of NAS (Sciuto et al., 2019). In addition, the gradient-based approach may be less appropriate for exploring different space (e.g., with a completely different number of layers), as opposed to the approach presented in this paper.

Bayesian optimization. BO has been an emerging technique for black-box optimization when function evaluations are expensive (Rana et al., 2017; Frazier, 2018; Gopakumar et al., 2018; Wan et al., 2021), and it has seen great success in hyperparameter optimization for deep learning (Li & Jamieson, 2018; Nguyen et al., 2020; Ru et al., 2020; Parker-Holder et al., 2020). Recently, Bayesian optimization has been used for searching the best neural architecture (Kandasamy et al., 2018; Jin et al., 2018; White et al., 2021). BO relies on a covariance function to represent the similarity between two data points. For such similarity representation, we can (1) directly measure the similarity of the networks by optimal transport, then modeling with GP surrogate in Kandasamy et al. (2018); or (2) measure the graphs based on the path-based encodings, then modeling with neural network surrogate in White et al. (2021). OTMANN (Kandasamy et al., 2018) shares similarities with Wasserstein (earth mover's) distances which also have an OT formulation. However, it is not a Wasserstein

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

distance itself—in particular, the supports of the masses and the cost matrices change depending on the two networks being compared. One of the drawback of OTMANN is that it may not be negative definite for a p.s.d. kernel which is an important requirement for modeling with GP. This is the motivation for our proposed tree-Wasserstein.

Path-based encoding. Bananas (White et al., 2021) proposes the path-based encoding for neural network architectures. The drawback of path-based encoding is that we need to enumerate all possible paths from the input node to the output node, in terms of the operations. This can potentially raise although it can work well in NASBench dataset (Ying et al., 2019) which results in $\sum_{i=0}^{5} 3^i = 364$ possible paths.

Kernel graph. Previous work considers the neural network architectures as the graphs, then defining various distances and kernels on graphs (Messmer & Bunke, 1998; Wallis et al., 2001; Kondor & Lafferty, 2002; Smola & Kondor, 2003; Gao et al., 2010; Ru et al., 2021). However, they may not be ideal for our NAS setting because neural networks have additional complex properties in addition to graphical structure, such as the type of operations performed at each layer, the number of neurons, etc. Some methods do allow different vertex sets (Vishwanathan et al., 2010), they cannot handle layer masses and layer similarities.

B. Related works in batch neural architecture search

They are several approaches in the literature which can be used to select multiple architectures for evaluation, including monte carlo tree search (Wang et al., 2020), evolutionary search (Real et al., 2019) and most of the batch Bayesian optimization approaches, such as Krigging believer (Ginsbourger et al., 2010), GP-BUCB (Desautels et al., 2014), B3O (Nguyen et al., 2016), GP-Thompson Sampling (Hernández-Lobato et al., 2017), and BOHB (Falkner et al., 2018).

Krigging believer (KB) (Ginsbourger et al., 2010) exploits an interesting fact about GPs: the predictive variance of GPs depends only on the input $\mathbf{x}$, but not the outcome values y. KB will iteratively construct a batch of points. First, it finds the maximum of the acquisition function, like the sequential setting. Next, KB moves to the next maximum by suppressing this point. This is done by inserting the outcome at this point as a halucinated value. This process is repeated until the batch is filled.

GP-BUCB (Desautels et al., 2014) is related to the above Krigging believer in exploiting the GP predictive variance. Particularly, GP-BUCB is similar to KB when the halucinated value is set to the GP predictive mean.

GP-Thompson Sampling (Hernández-Lobato et al., 2017) generates a batch of points by drawing from the posterior distribution of the GP to fill in a batch. In the continuous setting, we can draw a GP sample using random Fourier feature (Rahimi & Recht, 2007). In our discrete case of NAS, we can simply draw samples from the GP predictive mean.

C. Datasets

We summarize two benchmark datasets used in the paper. Neural architecture search (NAS) methods are notoriously difficult to reproduce and compare due to different search spaces, training procedures and computing cost. These make methods inaccessible to most researchers. Therefore, the below two benchmark datasets have been created.

NASBENCH101. The NAS-Bench-101 dataset[5] contains over $423,000$ neural architectures with precomputed training, validation, and test accuracy (Ying et al., 2019). In NASBench dataset, the neural network architectures have been exhaustively trained and evaluated on CIFAR-10 to create a queryable dataset. Each architecture training takes approximately 0.7 GPU hour.

NASBENCH201. NAS-Bench-201[6] includes all possible architectures generated by 4 nodes and 5 associated operation options, which results in $15,625$ neural cell candidates in total. The Nasbench201 dataset includes the tabular results for three subdatasets including CIFAR-10, CIFAR-100 and ImageNet-16-120. Each architecture training for Cifar10 takes approximately 0.7 GPU hours, Cifar100 takes 1.4 GPU hours and Imagenet takes 4 GPU hours.

[5]https://github.com/google-research/nasbench
[6]https://github.com/D-X-Y/NAS-Bench-201

D. Proofs

D.1. Proof for Lemma 2.1

Proof. We consider $X \sim GP(m(), k())$. If k is not a p.s.d. kernel, then there is some set of n points $(t_i)_{i=1}^n$ and corresponding weights $\alpha_i \in \mathcal{R}$ such that

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i k(t_i, t_j)\alpha_j < 0. \tag{13}$$

By the GP assumption, $\text{Cov}\big(X(t_i), X(t_j)\big) = k(t_i, t_j)$, we show that the variance is now negative

$$\text{Var}\left(\sum_{i=1}^{n} \alpha_i X(t_i)\right) = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \text{Cov}\big(X(t_i), X(t_j)\big)\alpha_j < 0. \tag{14}$$

The negative variance concludes our prove that the GP is no longer valid with non-p.s.d. kernel.

■

D.2. Proof for Proposition 1

Proof. We have that tree-Wasserstein (TW) is a metric and negative definite (Le et al., 2019b). Therefore, $W_{d_{\mathcal{T}_o}}, W_{d_{\mathcal{T}_-}}, W_{d_{\mathcal{T}_+}}$ are also a metric and negative definite.

Moreover, the discrepancy d_{NN} is a convex combination with positive weights for $W_{d_{\mathcal{T}_o}}, W_{d_{\mathcal{T}_-}}, W_{d_{\mathcal{T}_+}}$. Therefore, it is easy to verify that for given neural networks $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$, we have:

- $d_{\text{NN}}(\mathbf{x}_1, \mathbf{x}_1) = 0$,
- $d_{\text{NN}}(\mathbf{x}_1, \mathbf{x}_2) = d_{\text{NN}}(\mathbf{x}_2, \mathbf{x}_1)$,
- $d_{\text{NN}}(\mathbf{x}_1, \mathbf{x}_2) + d_{\text{NN}}(\mathbf{x}_1, \mathbf{x}_2) \geq d_{\text{NN}}(\mathbf{x}_2, \mathbf{x}_3)$.

Thus, d_{NN} is a pseudo-metric. Additionally, a convex combination with positive weights preserves the negative definiteness. Therefore, d_{NN} is negative definite. ■

D.3. Tree-Wasserstein kernel for neural networks

Proposition 3. *Given the scalar length-scale parameter σ_l^2, the tree-Wasserstein kernel for neural networks $k(x, z) = \exp(-\frac{d_{NN}(x,z)}{\sigma_l^2})$ is infinitely divisible.*

Proof. Given two neural networks $\mathbf{x}$ and $\mathbf{z}$, we introduce new kernels $k_\gamma(\mathbf{x}, \mathbf{z}) = \exp\big(-\frac{d_{\text{NN}}(\mathbf{x},\mathbf{z})}{\gamma\sigma_l^2}\big)$ for $\gamma \in \mathbb{N}^*$. Following Berg et al. (1984) (Theorem 3.2.2, p.74), $k_\gamma(x, z)$ is also positive definite. Moreover, we also have $k(\mathbf{x}, \mathbf{z}) = \big(k_\gamma(\mathbf{x}, \mathbf{z})\big)^\gamma$. Then, following Berg et al. (1984) (Definition 2.6, p.76), we complete the proof. ■

From Proposition 3, one does not need to recompute the Gram matrix of the TW kernel for each choice of σ_l^2, since it suffices to compute it once.

E. A brief discussion on tree-Wasserstein geometry

Tree-Wasserstein is a special case of the optimal transport (OT) where the ground cost is the tree metric (Do Ba et al., 2011; Le et al., 2019b). Tree-(sliced)-Wasserstein is a generalized version of the sliced-Wassersttein (SW) which projects supports into a line (i.e., one dimensional space) and relies on the closed-form solution of the univariate OT.[7] Moreover, the

[7]When a tree is a chain, the tree-(sliced)-Wasserstein is equivalent to the sliced-Wasserstein.

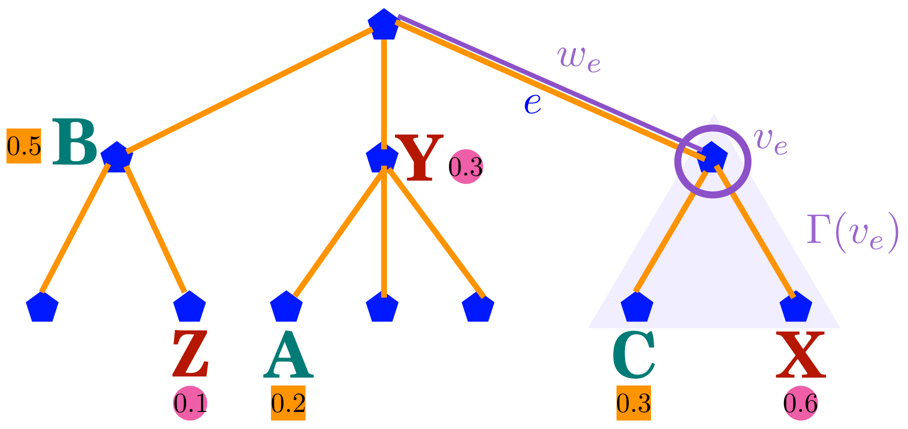

Figure 7. We illustrate a tree metric using a tree $\mathcal{T}$ which is used to explain the TW formula in Eq. (2). See text for more details.

tree-Wasserstein alleviates the curse of dimensionality for the sliced-Wasserstein.[8] Moreover, the tree-Wasserstein has a closed-form for computation and is negative definite which supports to build positive definite kernels.[9]

The tree structure has been leveraged for various optimal transport problems, especially for large-scale settings such as the unbalanced OT problem where distributions have different total mass (Le & Nguyen, 2021), the Gromov-Wasserstein problem where supports of distributions are in different spaces (Le et al., 2021; Mémoli et al., 2021), Gromov-Hausdoff problem for metric measure spaces (Mémoli et al., 2019), or the Wasserstein barycenter problem which finds the closest distribution to a given set of distributions (Le et al., 2019a).

In our work, we propose tree-Wasserstein for neural network architectures by leveraging a novel design of tree structures on network operations, indegree and outdegree representation for network structure. Therefore, our tree-Wasserstein for neural network can capture not only the frequency of layer operations (i.e., n-gram representation for layer operations) but also the entire network structure (i.e., indegree and outdegree representation).

F. An example of TW computation for neural network architectures

We would like to recall the TW distance between probability measures in Equation (2):

$$W_{d_{\mathcal{T}}}(\omega, \nu) = \sum_{e \in \mathcal{T}} w_e \left| \omega\big(\Gamma(v_e)\big) - \nu\big(\Gamma(v_e)\big) \right|. \tag{15}$$

We emphasize that tree Wasserstein (TW) is computed as *a sum over edges* in a tree, but *not the sum over supports* (e.g., in 1-gram representation). We next explain all components in TW (Equation (2)) for computation in details.

Let consider the tree metric illustrated in Fig. 7, and two probability measures $\omega = 0.2\delta_A + 0.5\delta_B + 0.3\delta_C$, and $\nu = 0.6\delta_X + 0.3\delta_Y + 0.1\delta_Z$. In this example, A, B, C are supports of probability measure ω with corresponding weights $0.2, 0.5, 0.3$. In other words, the mass of the probability measure ω at each support A, B, C is $0.2, 0.5, 0.3$ respectively. Similarly, X, Y, Z are supports of probability measure ν with corresponding weights $0.6, 0.3, 0.1$. Next, let consider edge e in Fig. 7 (the blue e), we have the following:

- w_e is the weight (or length) of the edge e (the purple line to emphasize the edge blue e);
- v_e is one of the two nodes of the edge e which is farther away from the tree root (the node with the purple circle);
- $\Gamma(v_e)$ is the subtree rooted at v_e (illustrated by the purple triangle blurred region);
- For the probability measure ω: $\omega\big(\Gamma(v_e)\big)$ is the total mass of the probability measure ω in the subtree $\Gamma(v_e)$; and only the support C (with weight 0.3) of ω is on the subtree $\Gamma(v_e)$. Hence, $\omega(\Gamma(v_e)) = 0.3$.

[8]SW projects supports into one dimensional space which limits its capacity to capture structures of distributions. The tree-Wasserstein alleviates this effect since a tree is more flexible and has a higher degree of freedom than a chain.

[9]The standard OT is in general indefinite.

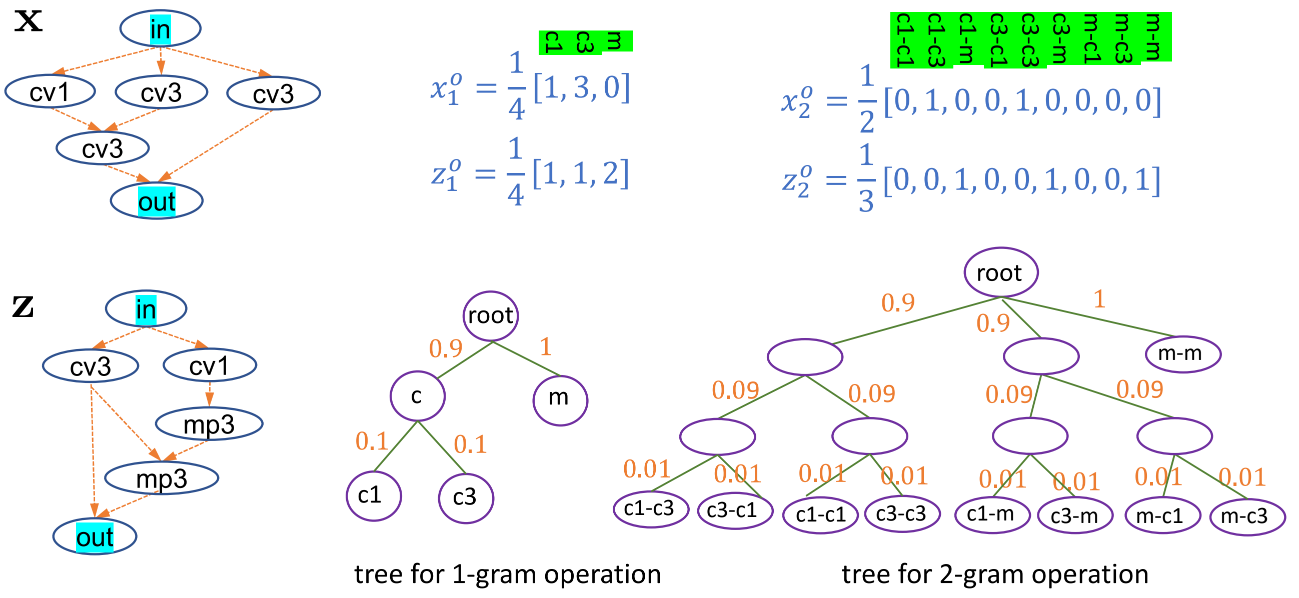
x
in
cv1
cv3
cv3
cv3
out
c1 c3 m
$x_1^o = \frac{1}{4}[1,3,0]$
$z_1^o = \frac{1}{4}[1,1,2]$
c1-c1 c1-c3 c1-m c3-c1 c3-c3 c3-m m-c1 m-c3 m-m
$x_2^o = \frac{1}{2}[0,1,0,0,1,0,0,0,0]$
$z_2^o = \frac{1}{3}[0,0,1,0,0,1,0,0,1]$
z
in
cv3
cv1
mp3
mp3
out
root
0.9
1
c
m
0.1
0.1
c1
c3
0.09
0.01
c1-c3
c3-c1
c1-c1
c3-c3
c1-m
c3-m
m-c1
m-c3
m-m
tree for 1-gram operation
tree for 2-gram operation

Figure 8. Example of TW used for calculating two architectures. The label of each histogram bin is highlighted in green. The distance between two nodes in a tree is sum of total cost if we travel between the two nodes, see Eq. (2). For example, the cost for moving `maxpool` (m) to `conv1` ($c1$) is $1 + 0.9 + 0.1 = 2$. We use similar analogy for computing in 2-gram (2g) representation. For the tree construction, let consider the the 1-gram tree, we group *c1* (`conv1`) and *c3* (`conv3`) due to their similarity, and "c" (`conv`) is an abstract label for the group of "convolution". We define the edge weights decreasing when the edge is far from the root, inspired by the partition-based tree metric sampling method (see the "n-gram representation for layer operations" part in §2.2).

- For the probability measure ν: $\nu\big(\Gamma(v_e)\big) = 0.6$. There is only the support X with weight 0.6 of ν is in the subtree $\Gamma(v_e)$.

We do the same procedure for all other edges in the tree $\mathcal{T}$ for the TW computation.

Intuitively, in case we consider the map $g : \omega \mapsto w_e\omega\big(\Gamma(e)\big) \mid_{e\in\mathcal{T}}$, we have $g(\omega) \in \mathbb{R}^m$ where m is the number of edges in the tree $\mathcal{T}$. The TW distance between two probability measures ω, ν is equivalent to the ℓ_1 distance between two m-dimensional (non-negative) vectors $g(\omega), g(\nu)$.

Now, we are ready to present an example of using TW for transporting two neural network architectures in Fig. 8. We consider a set $\mathbb{S}$ of interest operations as follow $\mathbb{S} = \{$`cv1, cv3, mp3`$\}$.

Neural network information $(\mathcal{S}^o, A)$**.** We use the order top-bottom and left-right for layers in $\mathcal{S}^o$.

- For neural network $\mathbf{x}$, we have $\mathcal{S}_{\mathbf{x}}^o = \{$`in, cv1, cv3, cv3, cv3, out`$\}$,

$$A_{\mathbf{x}} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Table 1. The tree-metric for operations in $W_{d_{\mathcal{T}_o}}$ from our predefined tree in Fig. 8 satisfying the following properties: (i) identical operation (`cv1`,`cv1`) has zero cost, (ii) similar operations (`cv1`,`cv3`) have small cost, and (iii) very different operations (`cv1`,`mp3`) have high cost. Then, the similarity score is normalized in Eq. (5) and Eq. (6) in which the weighting parameters α are learnt directly from the data. The utility score is further standardized $\mathcal{N}(0,1)$ for robustness. Therefore, our model is robust to the choice of the predefined tree for the cost in this table. (Recall that the cost in the table is computed by tree metric (i.e., the length of the shortest path) between the pair-wise operations in the predefined tree in Fig. 8.)

	`cv1`	`cv3`	`mp3`
`cv1`	0	0.2	2
`cv3`	0.2	0	2
`mp3`	2	2	0

- For neural network $\mathbf{z}$, we have $\mathcal{S}^o_{\mathbf{z}} = \{$`in, cv3, cv1, mp3, mp3, out`$\}$,

$$A_{\mathbf{z}} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

We show how to calculate these three representations (layer operation, indegree and outdegree) using tree-Wasserstein.

F.1. n-gram representation for layer operations

• **1-gram representation.** The 1-gram representations $\mathbf{x}^o_1$ and $\mathbf{z}^o_1$ for neural network $\mathbf{x}$, and $\mathbf{z}$ respectively are:

$$\mathbf{x}^o_1 = \frac{1}{4}(1,3,0) \qquad \mathbf{z}^o_1 = \frac{1}{4}(1,1,2)$$

where we use the order (`1:cv1, 2:cv3, 3:mp3`) for the frequency of interest operations in the set $\mathbb{S}$ for the 1-gram representation of neural network.

• **2-gram representation.** For the 2-gram representations $\mathbf{x}^o_2$ and $\mathbf{z}^o_2$ for neural networks $\mathbf{x}$ and $\mathbf{z}$ respectively, we use the following order for $\mathbb{S} \times \mathbb{S}$: (`1:cv1-cv1, 2:cv1-cv3, 3:cv1-mp3, 4:cv3-cv1, 5:cv3-cv3, 6:cv3:mp3, 7:mp3-cv1, 8:mp3:cv3, 9:mp3-mp3`). Thus, we have

$$\mathbf{x}^o_2 = \frac{1}{2}(0,1,0,0,1,0,0,0,0), \qquad \mathbf{z}^o_2 = \frac{1}{3}(0,0,1,0,0,1,0,0,1).$$

Or, we can represent them as empirical measures

$$\omega_{\mathbf{x}^o_2} = \tfrac{1}{2}\delta_{\texttt{cv1-cv3}} + \tfrac{1}{2}\delta_{\texttt{cv3-cv3}}, \qquad \omega_{\mathbf{z}^o_2} = \frac{1}{3}\delta_{\texttt{cv1-mp3}} + \frac{1}{3}\delta_{\texttt{cv3-mp3}} + \frac{1}{3}\delta_{\texttt{mp3-mp3}}.$$

• **Tree metrics for n-gram representations for layer operations.** We can use the tree metric in Fig. 8 for 1-gram and 2-gram representations. The tree metric for operations are summarized in Table 1.

Using the closed-form computation of tree-Wasserstein presented in Eq. (2) in the main text, we can compute $W_{d_{\mathcal{T}_o}}(\mathbf{x}^o_1, \mathbf{z}^o_1)$ for 1-gram representation and $W_{d_{\mathcal{T}_o}}(\mathbf{x}^o_2, \mathbf{z}^o_2)$ for 2-gram representation.

For 1-gram representation, we have

$$W_{d_{\mathcal{T}_o}}(\mathbf{x}^o_1, \mathbf{z}^o_1) = 0.1\left|\frac{1}{4} - \frac{1}{4}\right| + 0.1\left|\frac{3}{4} - \frac{1}{4}\right| + 0.9\left|1 - \frac{2}{4}\right| + 1\left|0 - \frac{2}{4}\right| = 1. \tag{16}$$

For 2-gram representation, we have

$$\begin{aligned} W_{d_{\mathcal{T}_o}}(\mathbf{x}_2^o, \mathbf{z}_2^o) =& 0.1\left|\frac{1}{2}-0\right| + 0.1\left|\frac{1}{2}-0\right| + 0.9\left|1-0\right| \\ &+ 0.01\left|0-\frac{1}{3}\right| + 0.01\left|0-\frac{1}{3}\right| + 0.99\left|0-\frac{2}{3}\right| + 1\left|0-\frac{1}{3}\right| = 2. \end{aligned} \tag{17}$$

F.2. Indegree and outdegree representations for network structure

The indegree and outdegree empirical measures $(\omega_{\mathbf{x}}^{d^-}, \omega_{\mathbf{x}}^{d^+})$ and $(\omega_{\mathbf{z}}^{d^-}, \omega_{\mathbf{z}}^{d^+})$ for neural networks $\mathbf{x}$ and $\mathbf{z}$ respectively are:

$$\omega_{\mathbf{x}}^{d^-} = \sum_{i=1}^{6} \mathbf{x}_i^{d^-} \delta_{\frac{\eta_{\mathbf{x},i}+1}{M_{\mathbf{x}}+1}}, \qquad \omega_{\mathbf{x}}^{d^+} = \sum_{i=1}^{6} \mathbf{x}_i^{d^+} \delta_{\frac{\eta_{\mathbf{x},i}+1}{M_{\mathbf{x}}+1}} \tag{18}$$

$$\omega_{\mathbf{z}}^{d^-} = \sum_{i=1}^{6} \mathbf{z}_i^{d^-} \delta_{\frac{\eta_{\mathbf{z},i}+1}{M_{\mathbf{z}}+1}}, \qquad \omega_{\mathbf{z}}^{d^+} = \sum_{i=1}^{6} \mathbf{z}_i^{d^+} \delta_{\frac{\eta_{\mathbf{z},i}+1}{M_{\mathbf{z}}+1}}, \tag{19}$$

where

$$\mathbf{x}^{d^-} = \left(0, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{2}{7}, \frac{2}{7}\right), \qquad \mathbf{x}^{d^+} = \left(\frac{3}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, 0\right), \tag{20}$$

$$\mathbf{z}^{d^-} = \left(0, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{2}{7}, \frac{2}{7}\right), \qquad \mathbf{z}^{d^+} = \left(\frac{2}{7}, \frac{2}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, 0\right), \tag{21}$$

$$\eta_{\mathbf{x}} = (0, 1, 1, 1, 2, 3), \qquad \eta_{\mathbf{z}} = (0, 1, 1, 2, 3, 4), \tag{22}$$

$M_{\mathbf{x}} = 3$, $M_{\mathbf{z}} = 4$, and $\mathbf{x}_i^{d^-}, \mathbf{x}_i^{d^+}, \eta_{\mathbf{x},i}$ are the i^{th} elements of $\mathbf{x}^{d^-}, \mathbf{x}^{d^+}, \eta_{\mathbf{x}}$ respectively. Consequently, one can leverage the indegree and outdegree for network structures to distinguish between $\mathbf{x}$ and $\mathbf{z}$.

We demonstrate in Fig. 9 how to calculate the tree-Wasserstein for indegree and outdegree. The supports of empirical measures $\omega_{\mathbf{x}}^{d^-}$ and $\omega_{\mathbf{z}}^{d^-}$ are in a line. For simplicity, we choose a tree as a chain of real values for the tree-Wasserstein distance. The tree becomes a chain of increasing real values, i.e., $\frac{1}{4} \to \frac{2}{4} \to \frac{3}{4} \to 1$, the weight in each edge is the ℓ_1 distance between two nodes of that edge. Particularly, the tree-Wasserstein is equivalent to the univariate optimal transport. It is similar for empirical measures $\omega_{\mathbf{x}}^{d^+}$ and $\omega_{\mathbf{z}}^{d^+}$.

• $W_{d_{\mathcal{T}_-}}(\omega_{\mathbf{x}}^{d^-}, \omega_{\mathbf{z}}^{d^-})$ **for indegree representation.** Using Eq. (2) we have

$$\begin{aligned} W_{d_{\mathcal{T}_-}}(\omega_{\mathbf{x}}^{d^-}, \omega_{\mathbf{z}}^{d^-}) &= \left(\tfrac{1}{4}-\tfrac{1}{5}\right)\underbrace{\left|\frac{7}{7}-\frac{7}{7}\right|}_{\Gamma\left(\delta_{\frac{1}{4}}\right)} + \left(\tfrac{2}{5}-\tfrac{1}{4}\right)\underbrace{\left|\frac{7}{7}-\frac{7}{7}\right|}_{\Gamma\left(\delta_{\frac{2}{5}}\right)} + \left(\tfrac{2}{4}-\tfrac{2}{5}\right)\underbrace{\left|\frac{7}{7}-\frac{5}{7}\right|}_{\Gamma\left(\delta_{\frac{2}{4}}\right)} + \left(\tfrac{3}{5}-\tfrac{2}{4}\right)\underbrace{\left|\frac{4}{7}-\frac{5}{7}\right|}_{\Gamma\left(\delta_{\frac{3}{5}}\right)} \\ &+ \left(\tfrac{3}{5}-\tfrac{2}{4}\right)\underbrace{\left|\frac{4}{7}-\frac{4}{7}\right|}_{\Gamma\left(\delta_{\frac{3}{4}}\right)} + \left(\tfrac{3}{5}-\tfrac{2}{4}\right)\underbrace{\left|\frac{2}{7}-\frac{4}{7}\right|}_{\Gamma\left(\delta_{\frac{4}{5}}\right)} + \left(\tfrac{3}{5}-\tfrac{2}{4}\right)\underbrace{\left|\frac{2}{7}-\frac{2}{7}\right|}_{\Gamma(\delta_1)} = \tfrac{4}{70} = 0.0571 \end{aligned} \tag{23}$$

where for each value $(t_1 - t_2)\,|a_1 - a_2|$, the value in the parenthesis, e.g., $(t_1 - t_2)$, is defined as the edge weights from δ_{t_1} to δ_{t_2} in Fig. 9 and the values in the absolute difference, e.g., $|a_1 - a_2|$ is the total mass of empirical measures in the subtrees rooted as the deeper node (i.e., δ_{t_2}) of corresponding edge (from δ_{t_1} to δ_{t_2}) as defined in Eq. (2).

• $W_{d_{\mathcal{T}_+}}(\omega_{\mathbf{x}}^{d^+}, \omega_{\mathbf{z}}^{d^+})$ **for outdegree representation.** Similarly, for outdegree representation, we have

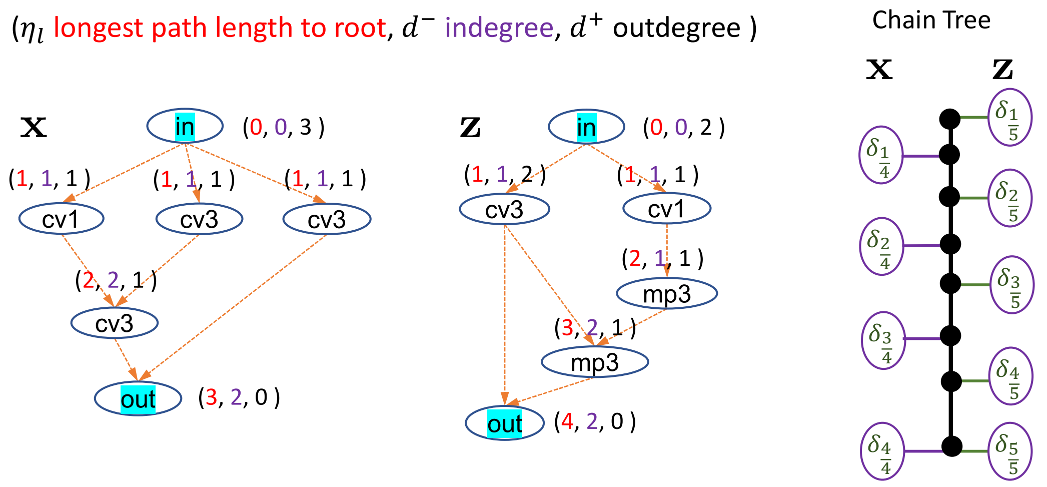
(η_l longest path length to root, d^- indegree, d^+ outdegree)
X
in (0, 0, 3)
(1, 1, 1) (1, 1, 1) (1, 1, 1)
cv1 cv3 cv3
(2, 2, 1)
cv3
out (3, 2, 0)
Z
in (0, 0, 2)
(1, 1, 2) (1, 1, 1)
cv3 cv1
(2, 1, 1)
mp3
(3, 2, 1)
mp3
out (4, 2, 0)
Chain Tree
X Z
$\delta_{\frac{1}{5}}$ $\delta_{\frac{1}{4}}$ $\delta_{\frac{2}{5}}$ $\delta_{\frac{2}{4}}$ $\delta_{\frac{3}{5}}$ $\delta_{\frac{3}{4}}$ $\delta_{\frac{4}{5}}$ $\delta_{\frac{4}{4}}$ $\delta_{\frac{5}{5}}$

Figure 9. Illustration of indegree and outdegree used in TW. Let $\eta_{x,\ell}$ and M_x be lengths of the longest paths from an input layer to a layer ℓ and to an output layer respectively. We can represent the empirical measure as $\omega_{\mathbf{x}}^{d^-} = \sum_{\ell\in L_{\mathbf{x}}} \mathbf{x}_\ell^{d^-}\delta_{\frac{\eta_{x,\ell}+1}{M_x+1}} = \frac{3}{7}\delta_{\frac{2}{4}} + \frac{2}{7}\delta_{\frac{3}{4}} + \frac{2}{7}\delta_{\frac{4}{4}}$ and $\omega_{\mathbf{x}}^{d^+} = \sum_{\ell\in L_{\mathbf{x}}} \mathbf{x}_\ell^{d^+}\delta_{\frac{\eta_{x,\ell}+1}{M_x+1}} = \frac{3}{7}\delta_{\frac{1}{4}} + \frac{3}{7}\delta_{\frac{2}{4}} + \frac{1}{7}\delta_{\frac{3}{4}}$ and for $\mathbf{z}$ as $\omega_{\mathbf{z}}^{d^-} = \sum_{\ell\in L_{\mathbf{z}}} \mathbf{z}_\ell^{d^-}\delta_{\frac{\eta_{z,\ell}+1}{M_z+1}} = \frac{2}{7}\delta_{\frac{2}{5}} + \frac{1}{7}\delta_{\frac{3}{5}} + \frac{2}{7}\delta_{\frac{4}{5}} + \frac{2}{7}\delta_{\frac{5}{5}}$ and $\omega_{\mathbf{x}}^{d^+} = \sum_{\ell\in L_{\mathbf{z}}} \mathbf{z}_\ell^{d^+}\delta_{\frac{\eta_{z,\ell}+1}{M_z+1}} = \frac{2}{7}\delta_{\frac{1}{5}} + \frac{3}{7}\delta_{\frac{2}{5}} + \frac{1}{7}\delta_{\frac{3}{5}} + \frac{1}{7}\delta_{\frac{4}{5}}$. The tree, which is a chain (1/5 → 1/4 → 2/5 → 2/4 → 3/5 → 3/4 → 4/5 → 1), is used to compute the distance.

$$\begin{aligned} W_{d_{\mathcal{T}_+}}(\omega_{\mathbf{x}}^{d^+},\omega_{\mathbf{z}}^{d^+}) &= \left(\tfrac{1}{4}-\tfrac{1}{5}\right)\underbrace{\left|\frac{7}{7}-\frac{5}{7}\right|}_{\Gamma\left(\delta_{\frac{1}{4}}\right)} + \left(\tfrac{2}{5}-\tfrac{1}{4}\right)\underbrace{\left|\frac{4}{7}-\frac{5}{7}\right|}_{\Gamma\left(\delta_{\frac{2}{5}}\right)} + \left(\tfrac{2}{4}-\tfrac{2}{5}\right)\underbrace{\left|\frac{4}{7}-\frac{2}{7}\right|}_{\Gamma\left(\delta_{\frac{2}{4}}\right)} + \left(\tfrac{3}{5}-\tfrac{2}{4}\right)\underbrace{\left|\frac{1}{7}-\frac{2}{7}\right|}_{\Gamma\left(\delta_{\frac{3}{5}}\right)} \\ &\quad + \left(\tfrac{3}{5}-\tfrac{2}{4}\right)\underbrace{\left|\frac{1}{7}-\frac{1}{7}\right|}_{\Gamma\left(\delta_{\frac{3}{4}}\right)} + \left(\tfrac{3}{5}-\tfrac{2}{4}\right)\underbrace{\left|\frac{0}{7}-\frac{1}{7}\right|}_{\Gamma\left(\delta_{\frac{4}{5}}\right)} + \left(\tfrac{3}{5}-\tfrac{2}{4}\right)\underbrace{\left|\frac{0}{7}-\frac{0}{7}\right|}_{\Gamma(\delta_1)} = \frac{6}{70} = 0.0857. \end{aligned} \tag{24}$$

From $W_{d_{\mathcal{T}_o}}$, $W_{d_{\mathcal{T}_-}}$ and $W_{d_{\mathcal{T}_+}}$, we can obtain the discrepancy d_{NN} between neural networks $\mathbf{x}$ and $\mathbf{z}$ as in Eq. (5) with predefined values $\alpha_1, \alpha_2, \alpha_3$.

G. Optimizing hyperparameters in TW and GP

As equivalence, we consider $\lambda_1 = \frac{\alpha_1}{\sigma_l^2}$, $\lambda_2 = \frac{\alpha_2}{\sigma_l^2}$ and $\lambda_3 = \frac{1-\alpha_1-\alpha_2}{\sigma_l^2}$ in Eq. (6) and present the derivative for estimating the variable λ in our kernel.

$$k(\mathbf{u},\mathbf{v}) = \exp\left(-\lambda_1 W_{d_{\mathcal{T}_o}}(\mathbf{u},\mathbf{v}) - \lambda_2 W_{d_{\mathcal{T}_-}}(\mathbf{u},\mathbf{v}) - \lambda_3 W_{d_{\mathcal{T}_+}}(\mathbf{u},\mathbf{v})\right). \tag{25}$$

The hyperparameters of the kernel are optimised by maximising the log marginal likelihood (LML) of the GP surrogate

$$\theta^* = \arg\max_{\theta} \mathcal{L}(\theta,\mathcal{D}), \tag{26}$$

where we collected the hyperparameters into $\theta = \{\lambda_1, \lambda_2, \lambda_3, \sigma_n^2\}$. The LML (Rasmussen, 2006) and its derivative are defined as

$$\mathcal{L}(\theta) = -\frac{1}{2}\mathbf{y}^\intercal \mathbf{K}^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}| + \text{constant} \tag{27}$$

Table 2. Properties comparison across different distances for using with GP-BO and k-DPP. GW is Gromov-Wasserstein. TW is tree-Wasserstein. OT is optimal transport.

Representation	Matrix/Graph	Path-Encoding	OT (or W)	GW	TW
Closed-form estimation	✓	✓	✗	✗	✓
Positive semi definite	✓	✓	✗	✗	✓
Different architecture sizes	✓, ✗	✗	✓	✓	✓
Scaling with architecture size	✓	✗	✓	✓	✓

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{1}{2}\left(\mathbf{y}^{\intercal}\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \theta}\mathbf{K}^{-1}\mathbf{y} - \operatorname{tr}\left(\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \theta}\right)\right), \tag{28}$$

where $\mathbf{y}$ are the function values at sample locations and $\mathbf{K}$ is the covariance matrix of $k(\mathbf{x}, \mathbf{x}')$ evaluated on the training data. Optimization of the LML was performed via multi-started gradient descent. The gradient in Eq. (28) relies on the gradient of the kernel k w.r.t. each of its parameters:

$$\frac{\partial k(\mathbf{u}, \mathbf{v})}{\partial \lambda_1} = -W_{d_{\mathcal{T}_o}}(\mathbf{u}, \mathbf{v}) \times k(\mathbf{u}, \mathbf{v}) \tag{29}$$

$$\frac{\partial k(\mathbf{u}, \mathbf{v})}{\partial \lambda_2} = -W_{d_{\mathcal{T}_-}}(\mathbf{u}, \mathbf{v}) \times k(\mathbf{u}, \mathbf{v}) \tag{30}$$

$$\frac{\partial k(\mathbf{u}, \mathbf{v})}{\partial \lambda_3} = -W_{d_{\mathcal{T}_+}}(\mathbf{u}, \mathbf{v}) \times k(\mathbf{u}, \mathbf{v}). \tag{31}$$

G.1. Proof for Proposition 2

Proof. Let A and B be the training and test set respectively, we utilize the Schur complement to have $K_{A \cup B} = K_A \times \left[K_B - K_{BA}K_A^{-1}K_{AB}\right]$ and the probability of selecting B is

$$P\left(B \subset \mathcal{P} \mid A\right) = \frac{\det\left(K_{A \cup B}\right)}{\det(K_A)} = \det\left(K_B - K_{BA}K_A^{-1}K_{AB}\right) = \det\left(\sigma(B \mid A)\right). \tag{32}$$

This shows that the conditioning of k-DPP is equivalent to the GP predictive variance $\sigma(B \mid A)$ in Eq. (9).

■

H. Distance Properties Comparison

We summarize the key benefits of using tree-Wasserstein (n-gram) as the main distance with GP for sequential NAS and k-DPP for batch NAS in Table 2. Tree-Wasserstein offers close-form computation and positive semi-definite covariance matrix which is critical for GP and k-DPP modeling.

Comparison with graph kernel. Besides the adjacency matrix representation, each architecture includes layer masses and operation type. We note that two different architectures may share the same adjacency matrix while they are different in operation type and layer mass.

Comparison with path-based encoding. TW can scale well to more nodes, layers while the path-based encoding is limited to.

Comparison with OT approaches in computational complexity. In general, OT is formulated as a linear programming problem and its computational complexity is super cubic in the size of probability measures (Burkard & Cela, 1999) (e.g., using network simplex). On the other hand, TW has a closed-form computation in Eq. (2), and its computational complexity is linear to the number of edges in the tree. Therefore, TW is much faster than OT in applications (Le et al., 2019b), and especially useful for large-scale settings where the computation of OT becomes prohibited.

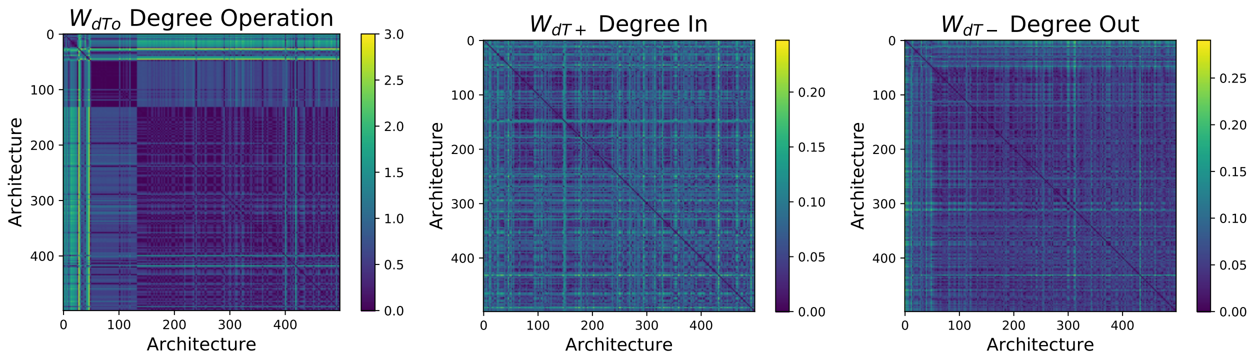

Figure 10. Tree-Wasserstein distances over 500 architectures on NASBENCH101.

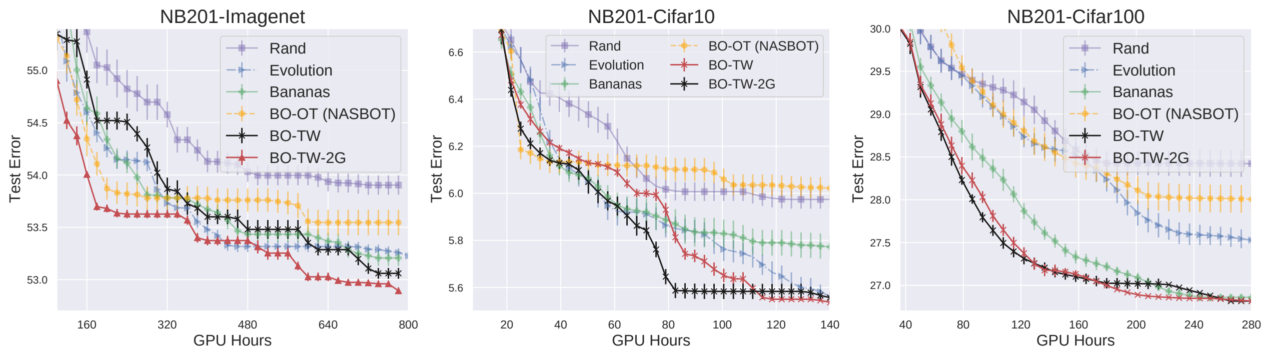

Figure 11. Additional sequential NAS comparison on NASBENCH201.

I. Additional Experiments and Illustrations

I.1. Model Analysis

We illustrate three internal distances of our tree-Wasserstein including $W_{d_{\mathcal{T}_o}}, W_{d_{\mathcal{T}_+}}, W_{d_{\mathcal{T}_-}}$ in Fig. 10. Each internal distance captures different aspects of the networks. The zero diagonal matrix indicates the correct estimation of the same neural architectures.

I.2. Further sequential and batch NAS experiments

To complement the results presented in the main paper, we present additional experiments on both sequential and batch NAS setting using NB101 and NB201 dataset in Fig. 11. In addition, we present experiments on batch NAS settings in Fig. 12 that the proposed k-DPP quality achieves the best performance consistently.

I.3. Ablation study using different acquisition functions

We evaluate our proposed model using two comon acquisition functions including UCB and EI. The result suggests that UCB tends to perform much better than EI for our NAS setting. This result is consistent with the comparison presented in Bananas (White et al., 2021).

I.4. Ablation study with different batch size B

Finally, we study the performance with different choices of batch size B in Fig. 14 which naturally confirms that the performance increases with larger batch size B.

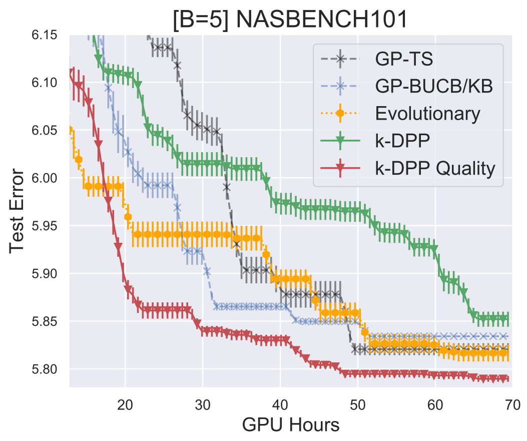

Figure 12. Additional result of batch NAS on NB101. We use TW-2G and a batch size $B = 5$

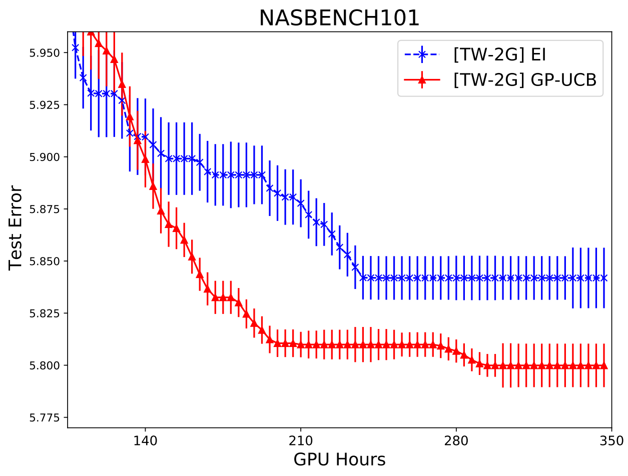

Figure 13. Optimizing the acquisition function using GP-UCB and EI on NB101. The results suggest that using GP-UCB will lead to better performance overall.

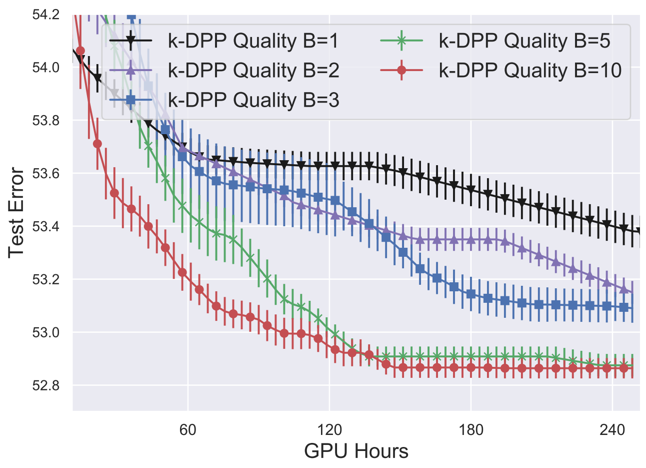

Figure 14. Performance with different batch sizes B on Imagenet. The result shows that the performance increases with larger batch size B, given the same wall-clock time budget (or batch iteration).