Supplementary Material for HOTNAS: Hierarchical Optimal Transport for Neural Architecture Search

Jiechao Yang[1,2] Yong Liu[1,2,*] Hongteng Xu[1,2]
[1] Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China
[2] Beijing Key Laboratory of Big Data Management and Analysis Methods
{yangjiechao2021, liuyonggsai, hongtengxu}@ruc.edu.cn

S1. Related Work Continued

Background of Optimal Transport Due to the ability to jointly exploit the feature and geometric structure information of the implicit data distribution [16], Optimal Transport (OT) has been extensively employed across various domains, including object detection [10], image generation [28], domain adaptation [40], cellular dynamics [36], and cancer detection [42]. The optimal transport problem can be broadly classified into two categories, the Monge problem [3] and the Kantorovich problem [15], depending on whether the mass of each transport unit can be divided. Unlike the Monge problem, the Kantorovich problem permits the mass of a given transport unit in one distribution to be distributed and allocated to multiple target units in another distribution. Hence, the Kantorovich problem is more suitable for discrete measures with different transport units. In this study, our method is categorized as the Kantorovich problem due to the discrete nature of the network architecture, where each network may consist of a varying number of layers.

In our approach, we specifically address the Kantorovich problem, which involves two discrete measures, denoted by $\alpha = \sum_{i=1}^{n} u_i \delta_{x_i}$ and $\beta = \sum_{j=1}^{m} v_j \delta_{y_j}$. Here u_i and v_j denote the mass associated with points x_i and y_j, respectively. It is important to note that the total mass of each discrete measure is constrained to one, such that $\sum_{i=1}^{n} u_i = 1$, $\sum_{j=1}^{m} v_j = 1$, and $u_i \geq 0$, $v_j \geq 0$. Let $c_{ij} = c(x_i, y_j)$ denote the transportation cost of dispatching unit mass from point x_i to another point y_j, and $\pi_{ij=\pi(x_i,y_j)}$ indicate the quantity of mass transported from point x_i to point y_j. The goal of the Kantorovich problem is to determine the optimal transport plan between the discrete measures α and β while minimizing the total transport cost:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{m} \pi_{ij} c_{ij},\ s.t.\ \sum_{i=1}^{n} \pi_{ij} = \beta_j, \sum_{j=1}^{m} \pi_{ij} = \alpha_i. \quad \text{(S1)}$$

As the above formulation features a linear objective and linear constraints, the Kantorovich problem can be classified as a linear programming problem. Therefore, off-the-shelf linear programming algorithms can be employed to solve it. The computational complexity of the conventional linear programming algorithm [37] is $O\left(d^3 \ln d\right)$, where $d = \max(n, m)$. Recently, Cuturi [6] proposed the Sinkhorn algorithm. , which adds an entropic regularization penalty to the transport plan of the original problem, resulting in an approximation of the solution of the Kantorovich problem. This method can significantly accelerate the optimization process and reduce its computational complexity to $O(d^2/\varepsilon^2)$, where ε is a desired tolerance. Here we choose the Sinkhorn algorithm to optimize our proposed HOTNN metric. Suppose two networks each contain M cells and n nodes, the complexity of cell-level metric is $\widetilde{O}(n^2/\varepsilon^2)$, while that of the network-level metric is $\widetilde{O}(M^2/\varepsilon^2)$. By leveraging the hierarchy of the networks, the complexity of the HOTNN metric can be further reduced from $\widetilde{O}(M^2 n^2/\varepsilon^2)$ to $\widetilde{O}(M^2/\varepsilon^2 + Mn^2/\varepsilon^2)$. For a more in-depth understanding of optimal transport, interested readers can refer to Peyre *et al*. [26].

Hierarchical Optimal Transport. Optimal transport (OT) [1, 23, 29, 31] has recently received widespread attention as it provides a good measure of the differences between different distributions by seeking the optimal matching from one distribution to another with the minimum transportation cost. An attractive property of OT is that it can incorporate the structure information of the data distribution into the transportation cost function. However, there may exist additional structures in data like the hierarchical structures of some objects [21]. Hierarchical optimal transport [20, 30, 34, 43] is a generalization of OT that leverages the hierarchical structure of data to rule out unnecessary assignments, which can efficiently improve assignments and accelerate the computation. The hierarchical optimal transport can be viewed as a nested optimal trans-

*Corresponding Author

port problem, where the ground metric itself is an optimal transport problem. Lee *et al*. [20] explore the hierarchical optimal transport to improve alignment for multimodal distribution by leveraging the clustered hierarchical structure in data. Similarly, Mourad *et al*. [7] proposed a hierarchical optimal transport metric for domain adaptation by leveraging both the hierarchical structures of the source and target data, which can lead to a better adaptation. In the context of NAS, the cell-based search space can be seen as a special type of hierarchical search space. It typically consists of a two-level hierarchical structure. The inner is the cell-level micro-architecture, which can be represented as a directed acyclic attributed graph, where each graph node represents a layer with the specified type of operations, and each directed edge determines the information flow from one node to another. The outer is the network-level macro-architecture, which determines the layout of different cells. In this study, we propose a hierarchical optimal transport metric called HOTNN to measure the similarity between different networks by leveraging the hierarchical structure of the cell-based networks.

Bayesian Optimization for NAS. Bayesian optimization (BO) [4, 9, 32] is an efficient global optimization method, which is particularly suitable when evaluating the objective function is costly. In fact, the training of a deep neural network usually takes the order of hours or even days [24]. Hence, Bayesian optimization methods have a great potential prospect in NAS. Bayesian optimization for NAS requires comparing the similarity of different network architectures. However, the conventional Bayesian optimization methods mainly focus on the Euclidean and categorical search space [11, 13], which is unsuitable for measuring the similarity of different complex graph-like network architectures. To solve this problem, various Bayesian optimization for NAS methods [14, 25, 27, 38] have been proposed. Jin *et al*. [12] explore the use of edit distance to measure the similarity between networks by counting how many operations are needed to change from one network to another. White *et al*. [38] design a path encoding scheme to measure the similarity of networks by comparing differences in all paths from the input to the output. However, the edit distance and path encoding methods both ignore the topological structure information of the whole network [25]. To overcome this shortcoming, Ru *et al*. [27] views the network architecture as an acyclic directed graph and apply the Weisfeiler-Lehman graph kernel to compare different networks. It is natural for them to handle graph-like search spaces and capture complex structure information. Unfortunately, they also miss important information about cell-based networks, such as the number of channels at each layer, the difference in operation types between node pairs, and the depth of the entire network. Compared to current Bayesian optimization for NAS methods, our proposed HOTNAS method is more flexible as it can jointly optimize the cell-level micro-architectures and the network-level macro-architectures.

S2. An Example of Calculating the Similarity of Different Operations

An example of the operation tree can be seen in Fig. S1. To meet the requirements of measuring similarity between different operations, we group similar operations on the same branch and divide different types of operations into other branches. For example, we allocate the similar "conv1" and "conv3" operations into the "convolution" branch and put the distinctly different "pool" operation into the "pooling" branch. We define the similarity cost between two operations as the sum of all edge weights from one operation node to another operation node in the predefined operation tree. As the tree depth increases, we progressively reduce the edge weights as in Nguyen *et al*. [25] to enlarge the transport cost difference between dissimilar and similar operations. For example, the transport cost between the similar "conv1" and "conv3" operations is $0.1 + 0.1 = 0.2$, while the transport cost between the distinctly different "conv1" and "pool" is $0.1+0.9+0.9+0.1 = 2$. Afterward, we transform this operation tree into an operational similarity cost matrix $\boldsymbol{D}^{pq} \in \mathbb{R}^{n\times m}$ of different operations in two cell networks, where each element $\boldsymbol{D}(o_i^p, o_j^q)$ is computed by summing over all edge weights from the operation o_i^p to another operation o_j^q in the predefined operation tree.

S3. Proofs

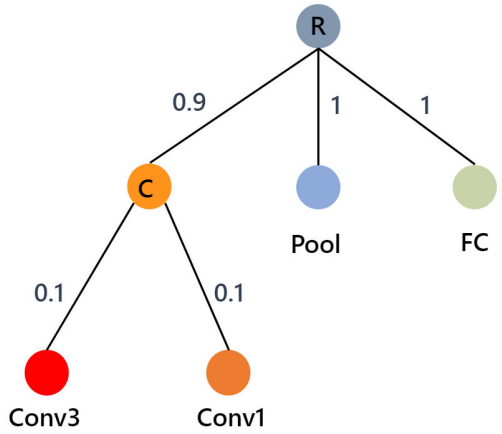
A valid metric $d(\cdot,\cdot)$ for neural networks should satisfy the following requirements:

- **Non-negativity.** $\forall \boldsymbol{a}_1, \boldsymbol{a}_2 \in \mathcal{A}, d(\boldsymbol{a}_1, \boldsymbol{a}_2) \geq 0$
- **Symmetry.** $\forall \boldsymbol{a}_1, \boldsymbol{a}_2 \in \mathcal{A}, d(\boldsymbol{a}_1, \boldsymbol{a}_2) = d(\boldsymbol{a}_2, \boldsymbol{a}_1)$
- **Definiteness.** $\boldsymbol{a}_1 = \boldsymbol{a}_2 \Longleftrightarrow d(\boldsymbol{a}_1, \boldsymbol{a}_2) = 0$
- **Triangle Inequality.** $\forall \boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3 \in \mathcal{A}, d(\boldsymbol{a}_1, \boldsymbol{a}_3) \leq d(\boldsymbol{a}_1, \boldsymbol{a}_2) + d(\boldsymbol{a}_2, \boldsymbol{a}_3)$

Proof of Theorem 3.1. Let $L^{pq}_{i,j,k,l} = |\boldsymbol{C}^p_{ik} - \boldsymbol{C}^q_{jl}|$, $D_{ij} = D(o_i^p, o_j^q)$, $D_{ik} = D(o_i^p, o_k^p)$, $D_{jl} = D(o_j^q, o_l^q)$, $H_{ij} = H(\boldsymbol{s}_i^p, \boldsymbol{s}_j^q)$, $H_{ik} = H(\boldsymbol{s}_i^p, \boldsymbol{s}_k^p)$, $H_{jl} = H(\boldsymbol{s}_j^q, \boldsymbol{s}_l^q)$. Then the iFGW$(\mathcal{G}_{B_1}, \mathcal{G}_{B_2})$ is:

$$\text{iFGW}(\mathcal{G}_{B_1}, \mathcal{G}_{B_2}) = \min_{\mathbf{T}\in U(\mathbf{p},\mathbf{q})} \sum_{i=1}^{n}\sum_{k=i+1}^{n}\sum_{j=1}^{m}\sum_{l=j+1}^{m} \lambda \mathbf{T}_{ij}\boldsymbol{C}^{pq}_{ij} + (1-\lambda)\mathbf{T}_{ij}\mathbf{T}_{kl}L^{pq}_{i,j,k,l}, \tag{S2}$$

where $\boldsymbol{C}^{pq}_{ij} = \varepsilon D_{ij} + (1-\varepsilon)H_{ij}$, $\boldsymbol{C}^{p}_{ik} = \varepsilon D_{ik} + (1-\varepsilon)H_{ik}$, $\boldsymbol{C}^{q}_{jl} = \varepsilon D_{jl} + (1-\varepsilon)H_{jl}$. Hence, iFGW$(\mathcal{G}_{B_1}, \mathcal{G}_{B_2})$ is



	Conv3	Conv1	Pool	FC
Conv3	0	0.1	2	2
Conv1	0.1	0	2	2
Pool	2	2	0	2
FC	2	2	2	0

Figure S1. An example of the operation tree, similar operations are grouped in the same branch, and dissimilar operations are assigned to the other branch. For example, we group the "conv3" and "conv1" operations into the "convolution" branch and put the distinctly different "pool" operation into the "pooling" branch. Then we can transform this operation tree to an operational similarity matrix, like the right part of this figure. Here we use "C" to represent the "convolution" abstract label, and "R" represents the root node of the operation tree. The similarity between different operations is computed by summing all edge weights from one operation node to another operation node, *e.g.*, the transport cost between the "conv1" and "FC" operations is $0.1+0.9+1=2$, the transport cost between the "conv1" and "conv3" operations is $0.1+0.1=0.2$.

equivalent to the original FGW metric if the cost function $\boldsymbol{C}_{ij}^{pq}$ and $L_{i,j,k,l}^{pq}$ is the distance between the nodes and node pairs, respectively. Next, we will demonstrate $\boldsymbol{C}_{ij}^{pq}$ and $L_{i,j,k,l}^{pq}$ are distances.

First, the $\boldsymbol{C}_{ij}^{pq}$ is a linear combination of the operational similarity D_{ij} and structural location difference H_{ij}. Le *et al.* [19] have demonstrated that TW is a metric. Since the operational similarity D_{ij} is built on tree-Wasserstein (TW), it is also a metric. Since kandasamy *et al.* have demonstrated that the shortest/longest/random-walk path length difference is a pseudo-distance, the average of all path length differences H_{ij} is still a pseudo-distance. Therefore, the $\boldsymbol{C}_{ij}^{pq}$ is a pseudo-distance.

Second, the $L_{i,j,k,l}^{pq}$ is a linear combination of the operational movement cost difference $|D_{ik}-D_{jl}|$ and the structural movement cost difference $|H_{ik}-H_{jl}|$. Assuming there exist another cell network $\mathcal{G}_{B_3}$ and node pairs $(y,z)\in\mathcal{G}_{B_3}$. It is obvious that $|D_{ik}-D_{jl}|$ satisfies the non-negativity and symmetry properties, *i.e.*, $|D_{ik}-D_{jl}|\geq 0, |D_{ki}-D_{lj}|=|D_{ik}-D_{jl}|$. If the operation information of the node pair (i,k) and the node pair (j,l) is same, $|D_{ik}-D_{jl}|=0$, and vice versa. Besides, it satisfies the triangle inequality, *i.e.*, $|D_{ik}-D_{jl}|+|D_{jl}-D_{yz}|\geq|D_{ik}-D_{yz}|$. Hence $|D_{ik}-D_{jl}|$ is a distance. Similarity, $|H_{ik}-H_{jl}|\geq 0$, $|H_{ki}-H_{lj}|=|H_{ik}-H_{jl}|$, $|H_{ik}-H_{jl}|+|H_{jl}-H_{yz}|\geq|H_{ik}-H_{yz}|$, thus $|H_{ik}-H_{jl}|$ is also a distance. As a result, $L_{i,j,k,l}^{pq}$ is a distance.

Finally, the iFGW metric is a combination of the pointwise matching cost and pairwise matching cost. Hence, the iFGW$(\mathcal{G}_{B_1},\mathcal{G}_{B_2})$ is a pseudo-metric. iFGW$(\mathcal{G}_{B_1},\mathcal{G}_{B_2})$ is 0 only if the cost function $\boldsymbol{C}_{ij}^{pq}$ and $L_{i,j,k,l}^{pq}$ are 0. It means the networks $\mathcal{G}_{B_1}$ and $\mathcal{G}_{B_2}$ have the same number of nodes with the same type of operations and are connected by the same edges. □

Proof of Theorem 3.2. The transport cost $\boldsymbol{S}_{st}^{12}$ between cell B_s^1 in the network $\boldsymbol{a}^1$ and cell B_t^2 in the network $\boldsymbol{a}^2$ is a convex combination of the cell-level similarity iFGW$(\mathcal{G}_{B_1},\mathcal{G}_{B_2})$ and the global position difference $P(B_s^1,B_t^2)$. Assuming there exists another network $\boldsymbol{a}^3=B_1^3\circ B_2^3\ldots\circ B_L^3$, where $\mathbf{r}=(r_1,r_2,\ldots,r_L)\in\Delta_L$ is its probability distribution. Let $\boldsymbol{S}_{tl}^{23}=(1-\eta)I(B_t^2,B_l^3)+\eta P(B_t^2,B_l^3)$, $\boldsymbol{S}_{sl}^{13}=(1-\eta)I(B_s^1,B_l^3)+\eta P(B_s^1,B_l^3)$. It is obvious that $P(B_s^1,B_t^2)$ satisfies the non-negativity and symmetry properties, *i.e.*, $P(B_s^1,B_t^2)\geq 0, P(B_s^1,B_t^2)=P(B_t^2,B_s^1)$. When $|\ \delta^1(B_s^1)/\delta^1\ |=|\ \delta^2(B_t^2)/\delta^2\ |$, then $P(B_s^1,B_t^2)=0$, and vice versa. Let $P(B_s^1,B_t^2)=|\ \delta^1(B_s^1)/\delta^1-\delta^2(B_t^2)/\delta^2\ |$, $P(B_t^2,B_l^3)=|\ \delta^2(B_t^2)/\delta^2-\delta^3(B_l^3)/\delta^3\ |$, $P(B_s^1,B_l^3)=|\ \delta^1(B_s^1)/\delta^1-\delta^3(B_l^3)/\delta^3\ |$, then $P(B_s^1,B_t^2)+P(B_t^2,B_l^3)\geq P(B_s^1,B_l^3)$. Therefore, $\boldsymbol{S}_{st}^{12}+\boldsymbol{S}_{tl}^{23}\geq\boldsymbol{S}_{sl}^{13}$.

Since iFGW$(\mathcal{G}_{B_1},\mathcal{G}_{B_2})\geq 0$ and $P(B_s^1,B_t^2)\geq 0$, HOTNN$(\boldsymbol{a}^1,\boldsymbol{a}^2)\geq 0$. Since iFGW$(\mathcal{G}_{B_1},\mathcal{G}_{B_2})=$

iFGW$(\mathcal{G}_{B_2}, \mathcal{G}_{B_1})$ and $P(B_s^1, B_t^2) = P(B_t^2, B_s^1)$, HOTNN$(\boldsymbol{a}^1, \boldsymbol{a}^2)$ = HOTNN$(\boldsymbol{a}^2, \boldsymbol{a}^1)$. Besides, HOTNN$(\boldsymbol{a}^1, \boldsymbol{a}^2)$ is zero only if iFGW$(\mathcal{G}_{B_1}, \mathcal{G}_{B_2}) = 0$ and $P(B_s^1, B_t^2) = 0$. It means the two networks have the same number of cells with the same architectures. Let $\Gamma_{\mathrm{st}} \in \mathbb{R}_+^{\mathrm{N}\times\mathrm{M}}$ and $\Gamma_{\mathrm{tl}} \in \mathbb{R}_+^{\mathrm{M}\times\mathrm{L}}$ be the solution to HOTNN$(\boldsymbol{a}^1, \boldsymbol{a}^2)$ and HOTNN$(\boldsymbol{a}^2, \boldsymbol{a}^3)$, respectively. Let $\Gamma_{\mathrm{sl}} = \Gamma_{\mathrm{st}}\mathrm{diag}(1/\mathbf{g})\Gamma_{\mathrm{tl}} \in \mathbb{R}_+^{\mathrm{N}\times\mathrm{L}}$. The Γ_{sl} is a feasible solution for HOTNN$(\boldsymbol{a}^1, \boldsymbol{a}^3)$,*i.e.*,$\Gamma_{\mathrm{sl}}\mathbf{1}_{\mathrm{L}} = \Gamma_{\mathrm{st}}\mathrm{diag}(1/\mathbf{g})\Gamma_{\mathrm{tl}}\mathbf{1}_{\mathrm{L}} = \Gamma_{\mathrm{st}}\mathrm{diag}(1/\mathbf{g})\mathbf{g} = \Gamma_{\mathrm{st}}\mathbf{1}_{\mathrm{M}} = \mathbf{f}$, $\Gamma_{\mathrm{sl}}^{\mathrm{T}}\mathbf{1}_{\mathrm{N}} = \Gamma_{\mathrm{tl}}^{\mathrm{T}}\mathrm{diag}(1/\mathbf{g})\Gamma_{\mathrm{st}}^{\mathrm{T}}\mathbf{1}_{\mathrm{N}} = \Gamma_{\mathrm{tl}}^{\mathrm{T}}\mathrm{diag}(1/\mathbf{g})\mathbf{g} = \Gamma_{\mathrm{tl}}^{\mathrm{T}}\mathbf{1}_{\mathrm{M}} = \mathbf{r}$.

Then, we can write:

$$\begin{aligned}
\sum_{sl}\Gamma_{\mathrm{sl}}\mathrm{S}_{\mathrm{sl}}^{13} &= \sum_{sl}\boldsymbol{S}_{sl}^{13}\sum_{t}\frac{\Gamma_{\mathrm{st}}\Gamma_{\mathrm{tl}}}{\mathbf{g}} \\
&\leq \sum_{stl}(\boldsymbol{S}_{st}^{12}+\boldsymbol{S}_{tl}^{23})\frac{\Gamma_{\mathrm{st}}\Gamma_{\mathrm{tl}}}{\mathbf{g}} \\
&= \sum_{stl}\boldsymbol{S}_{st}^{12}\frac{\Gamma_{\mathrm{st}}\Gamma_{\mathrm{tl}}}{\mathbf{g}} + \sum_{stl}\boldsymbol{S}_{tl}^{23}\frac{\Gamma_{\mathrm{st}}\Gamma_{\mathrm{tl}}}{\mathbf{g}} \\
&= \sum_{st}\boldsymbol{S}_{st}^{12}\Gamma_{\mathrm{st}}\sum_{\mathrm{l}}\frac{\Gamma_{\mathrm{tl}}}{\mathbf{g}} + \sum_{tl}\boldsymbol{S}_{tl}^{23}\Gamma_{\mathrm{tl}}\sum_{\mathrm{s}}\frac{\Gamma_{\mathrm{st}}}{\mathbf{g}} \\
&= \sum_{st}\Gamma_{\mathrm{st}}\mathrm{S}_{\mathrm{st}}^{12} + \sum_{tl}\Gamma_{\mathrm{tl}}\mathrm{S}_{\mathrm{tl}}^{23}.
\end{aligned} \tag{S3}$$

Thus,

$$\mathrm{HOTNN}(\boldsymbol{a}^1, \boldsymbol{a}^2) + \mathrm{HOTNN}(\boldsymbol{a}^2, \boldsymbol{a}^3) \geq \mathrm{HOTNN}(\boldsymbol{a}^1, \boldsymbol{a}^3). \tag{S4}$$

Therefore, HOTNN$(\boldsymbol{a}^1, \boldsymbol{a}^2)$ is a pseudo-metric. Since each network has a chain structure, the support space of the HOTNN metric is one-dimensional. According to the Theorem 5 in Kolouri *et al*. [17], the HOTNN metric is negative semidefinite. □

S4. Algorithm Details of HOTNAS

Here, we describe the framework of our proposed HOTNAS method in detail. Bayesian optimization is a sequential model-based optimization method that iteratively evaluates a limited number of points to find the optimum of the objective function effectively. It uses a surrogate model to learn the complex relationship between the input variables and objective function and an acquisition function to carefully select the next promising sample points by fully balancing the exploration and exploitation of the whole search space. Next, we will describe the surrogate model and acquisition function in our proposed HOTNAS method.

Surrogate model. We choose the Gaussian process (GP) as our surrogate model because it provides a well-defined predictive distribution and good uncertainty estimation [2]. An important property of the Gaussian process is that if a gaussian prior is assumed over the objective function, the posterior distribution remains a smooth gaussian distribution [39], where the predictive mean $\mu(\boldsymbol{a})$ and predictive variance $\sigma(\boldsymbol{a})$ are represented as follows:

$$\mu(\boldsymbol{a}) = \mathbf{k}(\boldsymbol{a}, \mathbf{A})\boldsymbol{K}^{-1}\mathbf{y}, \tag{S5}$$

$$\sigma^2(\boldsymbol{a}) = k(\boldsymbol{a}, \boldsymbol{a}) - \mathbf{k}(\boldsymbol{a}, \mathbf{A})\boldsymbol{K}^{-1}\mathbf{k}^T(\boldsymbol{a}, \mathbf{A}), \tag{S6}$$

where $\mathbf{A} = [\boldsymbol{a}^1, \ldots, \boldsymbol{a}^{n_0}]$ and $\mathbf{y} = [y^1, \ldots, y^{n_0}]$ are the current network architectures and the corresponding validition loss on the initial observation set $\mathcal{D}_0 = \{\boldsymbol{a}^i, y^i = f(\boldsymbol{a}^i)\}_{i=1}^{n_0}$, $\boldsymbol{K}$ is the kernel matrix whose element is the kernel function $k(\boldsymbol{a}^i, \boldsymbol{a}^j)$ between different input networks on the observation set $\mathcal{D}_0$.

Acquisition Function The acquisition function is a utility function that evaluates the possibility that a neural network architecture is optimal. To efficiently guide the search for the optima, the acquisition function balances the exploration of unknown search regions and the exploitation of existing architectures. Common acquisition functions include probability of improvement (PI) [18], expected improvement (EI) [13], and upper confidence bound (UCB) [33]. Here, we choose UCB as our acquisition function because it is computationally simple, easy to optimize, and has an excellent theoretical guarantee for convergence [33]. The UCB acquisition function is defined as:

$$u_t(\boldsymbol{a}) = \mu_t(\boldsymbol{a}) - \alpha \cdot \sigma_t(\boldsymbol{a}), \quad \alpha \geq 0, \tag{S7}$$

where $\mu(\boldsymbol{a})$ and $\sigma(\boldsymbol{a})$ are the predicted mean and standard deviation of the GP surrogate model at architecture $\boldsymbol{a}$ in the tth iteration, α is the hyperparameter that balances the exploration and exploitation. We set α at initialization to $\alpha_0 = 2$ and decay it $\alpha_t = \alpha_0\sqrt{\frac{1}{2}\log(2(n+1))}$ as suggested by Srinivas *et al*. [33], where t is the number of iterations. To optimize this function, we first generate a set of candidate architectures $\mathcal{P}_t \in \mathcal{A}$ by using the mutation method same as White *et al*. [38]. After that, we compute the UCB value of each candidate architecture and select the most promising architecture with the maximum acquisition function, *i.e.*, $\boldsymbol{a}_{\mathrm{new}} = \arg\max_{\boldsymbol{a}\in\mathcal{P}_t} u_t(\boldsymbol{a})$.

The general algorithm of our proposed HOTNAS method is summarized in the Algorithm 1. We first initialize n_0 neural network architectures by randomly sampling from the cell-based network search space and then evaluate these architectures to obtain the initial observation set $\mathcal{D}_0 = \{\boldsymbol{a}^i, y^i = f(\boldsymbol{a}^i)\}_{i=1}^{n_0}$. After that, we compute the HOTNN distance between different network pairs in the current observation set $\mathcal{D}_t = \mathcal{D}_0$ and embed it in the GP surrogate model. Afterward, we fit the GP model on the current observation set $\mathcal{D}_t$ and construct the UCB acquisition

function based on the predictive distribution of the GP surrogate model. We then generate a candidate pool of the network architectures by mutating the current best-performing network architectures and selecting the next promising architecture $\boldsymbol{a}_{\text{new}}$ with the largest UCB value. The surrogate mode is updated by fitting on the new observation set $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{\boldsymbol{a}_{\text{new}}, y_{\text{new}}\}$. This process continues until it finds the best network architecture with the lowest validation loss or the maximum number of iterations T is reached.

Algorithm 1: Hierarchical Optimal Transport for Neural Architecture Search

Input: Total number of iterations N, initial datapoints $\mathcal{D}_0$, search space $\mathcal{A}$, The maxinum iterations T

Output: The best architecture $\boldsymbol{a}^*$

for $t = 0$ ***to*** $T - 1$ **do**
- Compute HOTNN metric between different architectures on the current observation set $\mathcal{D}_t = \mathcal{D}_0$;
- Embed the HOTNN metric to the kernel function of GP;
- Fit the GP on the current observation set $\mathcal{D}_t$;
- Construct the UCB acquisition function based on the predictive mean and variance (see Eq. (S7));
- Generate a pool of candidate architectures $\mathcal{P}_t$ by mutating the current best-performing architectures;
- Select the next promising architectures $\boldsymbol{a}_{\text{new}} = \arg\max_{\boldsymbol{a} \in \mathcal{P}_t} u_t(\boldsymbol{a})$;
- Evaluate $\boldsymbol{a}_{\text{new}}$ to obtain its validation loss y_{new};
- Update the observation set $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{\boldsymbol{a}_{\text{new}}, y_{\text{new}}\}$;

end

return the best-performing architecture $\boldsymbol{a}^* = \arg\min_{\boldsymbol{a} \in \mathcal{D}_T} f(\boldsymbol{a})$

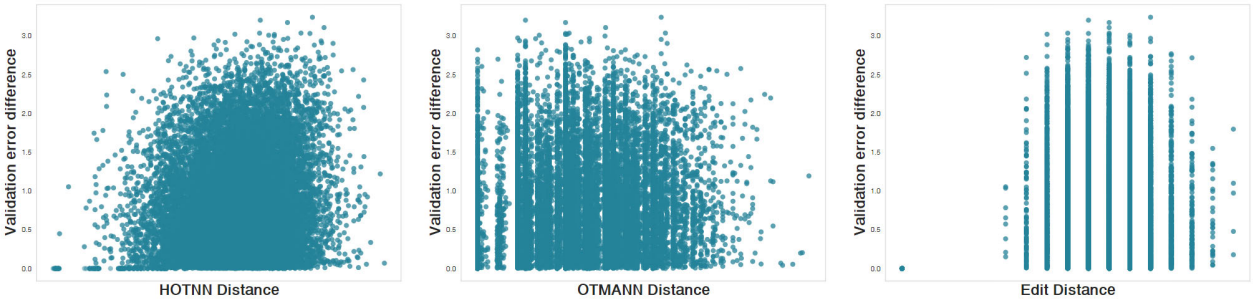
S5. Illustrations of the HOTNN distance

In order to demonstrate that the HOTNN metric provides a reliable measure for cell-based networks, we have visualized the relationship between the HOTNN distance and the difference in validation errors on the DARTS benchmark. To achieve this, we first generated 200 networks by randomly sampling from the DARTS search space and computing the HOTNN distance between them. We also included the edit distance and the OTMANN distance for comparison. As depicted in Fig. S2, each point on the graph represents a pair of networks. It is noteworthy that the edit distance is a discrete measure and fails to reflect the smooth variation of the validation error. Our proposed HOTNN metric, on the other hand, is smooth in relation to validation performance and provides a reasonable measure of the similarity between different networks. When the HOTNN distance is low, the pair of networks exhibit a small difference in the validation performance. However, as the HOTNN distance increases, the validation performance differences become more scattered. This aligns with the notion that similar networks possess a small difference in the validation performance, while dissimilar networks exhibit a large variation in the validation performance. In contrast, OTMANN fails to capture this phenomenon in the DARTS search space, likely due to its disregard of the similarity between cells in the modular cell-based search space. Furthermore, the results indicate that our proposed HOTNN metric correlates with validation error performance and can thus be seamlessly embedded into the Bayesian optimization for NAS framework.

S6. Experimental settings

To ensure the comparability of all experiments, we run each experiment five times initialized by different random seeds, and report the mean and standard performance of different methods across various tasks. We use the POT [8] library to implement our proposed HOTNN metric. We conduct all experiments on a machine with an Intel Xeon Gold 5218R CPU and four NVIDIA GeForce RTX 3090 GPUs. Next, we will describe the experimental settings of all methods in different search spaces.

Experimental Settings on TransNAS-Bench-101 Benchmark. The TransNAS-Bench-101 benchmark is a tabular benchmark that covers seven common vision tasks. Each task has a different evaluation metric. For object classification, scene classification, and jigsaw tasks, we use the best top-1 accuracy acc as the evaluation metric and use $100 - acc$ as the evaluation loss. For the room layout task, we choose the best negative log loss $negloss$ as the evaluation metric and use $-100negloss$ as the evaluation loss. For the semantic segmentation task, we choose the best $mloU$ as the evaluation metric and use $100 - mloU$ as the evaluation loss. For autoencoding and surface normal tasks, we choose the best $SSIM$ evaluation metric and use $100(1 - SSIM)$ as the evaluation loss. Here we choose its macro skeleton search space. The whole network is stacked with four to six modules, with each module comprising two residual blocks. The residual blocks can choose whether to downsample features with a stride of one or two. We set the similarity between the residual blocks with down-sampling and without down-sampling to 1. We generate a pool of 100 candidate architectures by mutating the current best architectures at each iteration. We then set the number of initial architectures to 20 and the maximum number of iterations to 100. Except for the above changes, the implementation and



(a) (b) (c)

Figure S2. An illustration of the relationship between different distances and validation error differences on the DARTS benchmark: (a) HOTNN (b) OTMANN (c) edit distance.

Table S1. Comparisons of the impact of macro-architectures and micro-architectures in the final performance, where HOTNASWOA represents HOTNAS without macro-architectures and HOTNASWOI represents HOTNAS without micro-architectures.

Algorithm	Valid loss	Test loss	Search time (GPU days)	Params (M)
BANANAS	5.62±0.04	2.72±0.06	10.6±0.2	4.2±0.1
BO-TW	5.58±0.02	2.61±0.05	9.4±0.6	4.1±0.4
BO-TW-2G	5.43±0.01	2.54±0.02	10.9±0.1	3.7±0.2
NAS-BOWL	5.76±0.06	2.78±0.09	11.4±0.3	4.6±0.1
HOTNASWOA	5.44±0.01	2.64±0.05	4.6±0.1	8.7±0.3
HOTNASWOI	5.64±0.01	2.56±0.12	**8.2±0.3**	3.9±0.1
HOTNAS	**5.37±0.01**	**2.43±0.04**	9.8±0.1	**3.6±0.2**

experimental settings of other methods remain unchanged as the original study.

Experimental Settings on NAS-Bench-101 Benchmark. The NAS-Bench-101 is also a tabular benchmark that focuses on searching a single cell architecture. Note that the NAS-Bench-101 benchmark comprises a larger number of network architectures than the TransNAS-Bench-101. Therefore, we increase the pool size to 200 at each iteration. We then set the number of initial architectures to 20 and the maximum number of iterations to 200. For better adaptation to the cell architectures, we use a modified version of the NASBOT proposed by Nguyen *et al*. [25]. With the exception of the aforementioned modifications, the implementation and experimental settings of other methods are consistent with those of the original study.

Experimental Settings on DARTS Benchmark. DARTS is a large non-tabular benchmark that does not provide the training and test performance of each network. To accommodate our settings, we first transform the edge-attributed directed acyclic graph representation into the node-attributed directed acyclic graph representation as proposed by Ru *et al*. [27]. Since each cell in the DARTS search space has two input nodes from the previous two cells, we compute the sum of structural position differences relative to each input as the position difference between nodes in different networks when computing the HOTNN metric. We follow the most common DARTS setup [5, 22, 41]. During the search stage, we train each network using half of the CIFAR-10 training data for 100 epochs with a batch size of 64, an initial channel size of 16, and validate using the other half of the CIFAR-10 training data. We optimize network weights using the SGD optimizer with a momentum of 0.9, a weight decay of 3×10^{-4}, and an initial learning rate of 0.1, which is annealed to zero over 100 epochs following a cosine schedule. We set the number of initial architectures to 20 and the maximum number of iterations to 100. Since the search space is vast, we generate a pool of 200 candidate networks at each iteration by mutating the current top-5 best-performing architectures and 100 networks by randomly sampling from the search space. After the searching stage, we select the best-performing architecture with the lowest validation loss and evaluate it on the CIFAR-10 test set. During the evaluation stage, we train the network from scratch using the entire CIFAR-10 training set for 600 epochs with a batch size of 128, an initial channel size of 36, and optimize the network weights using the SGD optimizer with a momentum of 0.9, a weight decay of 3×10^{-4}, a norm gradient clipping at 5, and an initial learning rate of 0.025, which is annealed to zero over 600 epochs following a cosine schedule. Furthermore, we directly transfer the network to the CIFAR-100 dataset. The training settings are the same as in the CIFAR-10 dataset except for the weight decay is 5×10^{-4}.

S7. Ablation Studies

Ablation Study on the importance of operational similarity and location difference. During the construction of the iFGW metric, we take into account both operational similarity and positional differences. A natural question is which one is more important. To address this issue, we

Table S2. Comparisons of experiment results on the NAS-Bench-101 benchmark.

Algorithm	Valid loss	Test loss	Search time (h)	Params (M)
Evolutionary search	5.44±0.01	5.96±0.01	1.92±0.13	11.68±0.26
Random search	5.64±0.01	6.22±0.02	1.54±0.27	12.32±0.26
BO-edit	5.39±0.02	5.96±0.02	1.20±0.12	10.81±0.11
NASBOT	5.58±0.03	6.16±0.03	1.45±0.26	10.22±0.20
BANANAS	5.41±0.02	6.05±0.03	0.92±0.08	11.65±0.48
BO-TW	5.38±0.05	6.02±0.03	1.02±0.06	**9.12±0.94**
BO-TW-2G	5.41±0.01	5.94±0.02	0.91±0.12	9.84±0.62
NAS-BOWL	5.49±0.02	6.10±0.02	1.61±0.18	10.21±0.11
HOTNAS	**5.36±0.01**	**5.94±0.01**	**0.83±0.01**	9.43±0.08

conducted an experiment where we compared the impact of HOTNAS without operational similarity (by setting $\varepsilon = 0$ in the iFGW metric) to HOTNAS without positional difference (by setting $\varepsilon = 1$ in the iFGW metric) on the final performance. Here we choose to use NAS-Bench-101 benchmark for this comparison since its macro-architectures are fixed and the distance between the whole network pairs is equivalent to the distance between the cell network pairs. Our results indicate that operation similarity (valid loss: 5.62 ± 0.02) is more crucial than location difference (valid loss: 5.47 ± 0.06) in determining the final performance.

Ablation Study on the impact of macro-architectures and micro-architectures. To investigate the impact of macro-architectures and micro-architectures on our HOTNAS method, we choose the widely used DARTS benchmark as our example. Firstly, we fixed the macro-architectures of the entire search space by setting the network depth to 20 and then observed the effect of macro-architectures on the final performance of our HOTNAS method. Furthermore, we compared our method with other cell network search methods such as BANANAS [38], BO-TW [25], BO-TW-2G [25], and NAS-BOWL [27]. To determine the extent to which performance can degrade when ignoring micro-architectural changes, we fixed the micro-architecture with the best performance discovered by our proposed HOTNAS method. Tab. S1 shows the performance comparison of HOTNAS when either macro-architectures or micro-architectures are ignored. The results indicate that the performance of HOTNAS significantly decreases when either the macro-architecture or micro-architecture is ignored. Our results demonstrate that the performance of HOTNAS significantly decreases when either the macro-architecture or micro-architecture is ignored. This highlights the critical role that both macro-architecture and micro-architecture play in the final performance of our HOTNAS method.

S8. Additional Figures and Tables

References

[1] David Alvarez-Melis, Tommi Jaakkola, and Stefanie Jegelka. Structured optimal transport. In *AISTATS*, pages 1771–1780, 2018. 1

[2] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization. In *NeurIPS*, pages 21524–21538, 2020. 4

[3] Jean-David Benamou, Brittany D Froese, and Adam M Oberman. Numerical solution of the optimal transportation problem using the monge–ampère equation. *Journal of Computational Physics*, 260:107–126, 2014. 1

[4] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010. 2

[5] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *ICCV*, pages 1294–1303, 2019. 6

[6] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NeurIPS*, 2013. 1

[7] Mourad El Hamri, Younès Bennani, and Issam Falih. Hierarchical optimal transport for unsupervised domain adaptation. *Machine Learning*, pages 1–24, 2022. 2

[8] Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. 5

[9] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018. 2

[10] Zheng Ge, Songtao Liu, Zeming Li, Osamu Yoshie, and Jian Sun. Ota: Optimal transport assignment for object detection. In *CVPR*, pages 303–312, 2021. 1

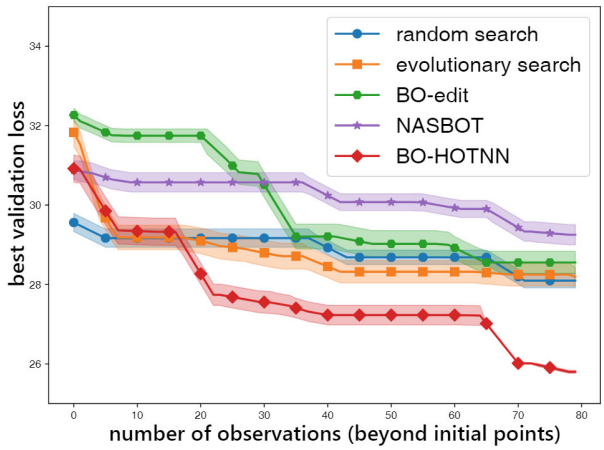
[11] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated Machine Learning: Methods, Systems, Challenges*. Springer Nature, 2019. 2

[12] Haifeng Jin, Qingquan Song, and Xia Hu. Auto-keras: An efficient neural architecture search system. In *SIGKDD*, pages 1946–1956, 2019. 2

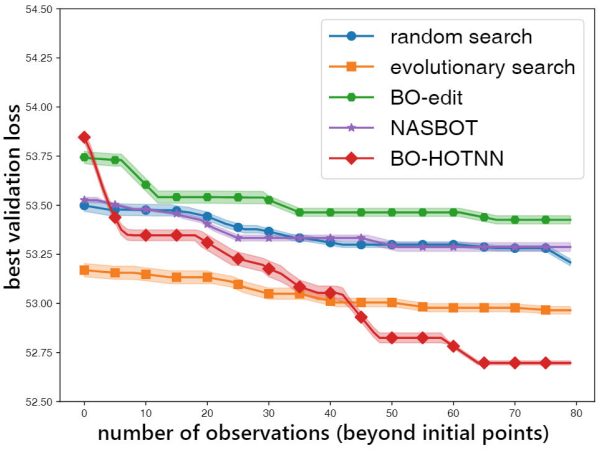
[13] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998. 2, 4

[14] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabás Póczos, and Eric P. Xing. Neural architecture search with bayesian optimisation and optimal transport. In *NeurIPS*, pages 2020–2029, 2018. 2

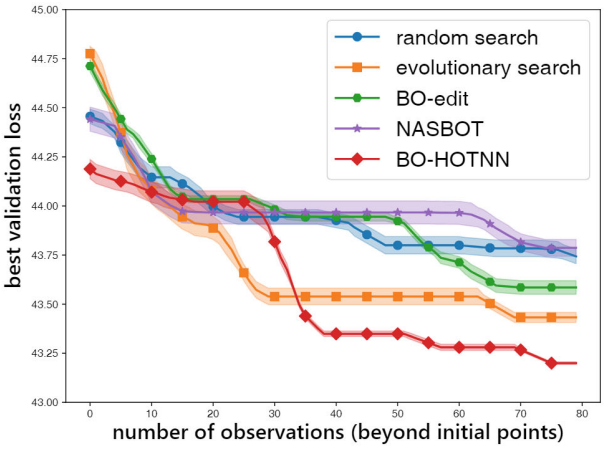
[15] Leonid V Kantorovich. On the translocation of masses. *Journal of Mathematical Sciences*, 133(4):1381–1382, 2006. 1



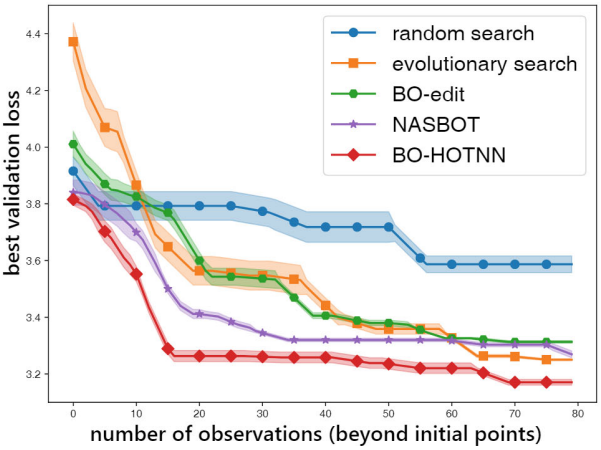
(a) Autoencoder



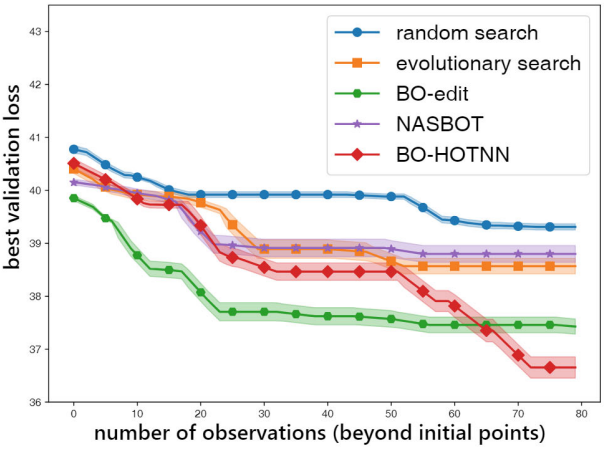
(b) Object classification



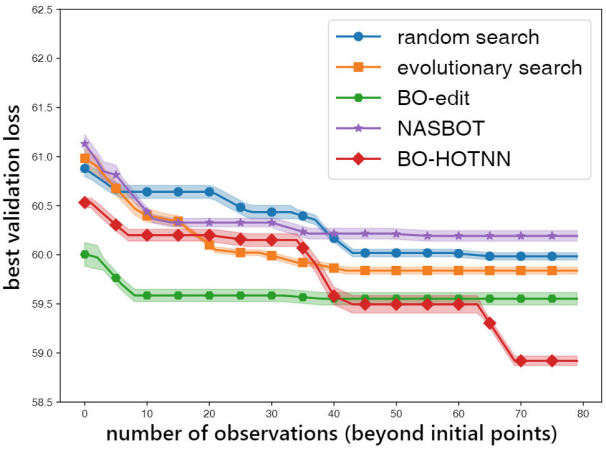
(c) Scene classification



(d) Jigsaw

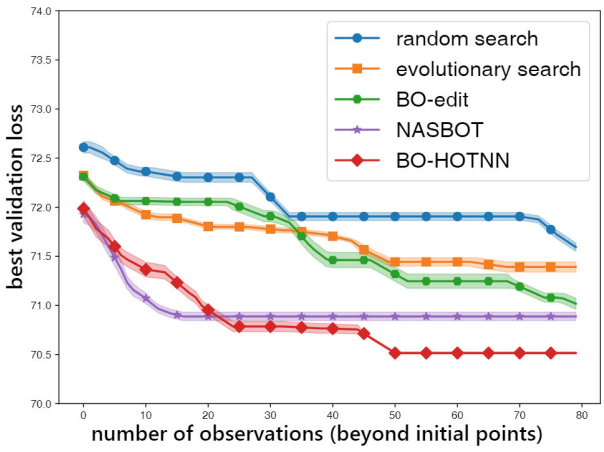


(e) Surface normal

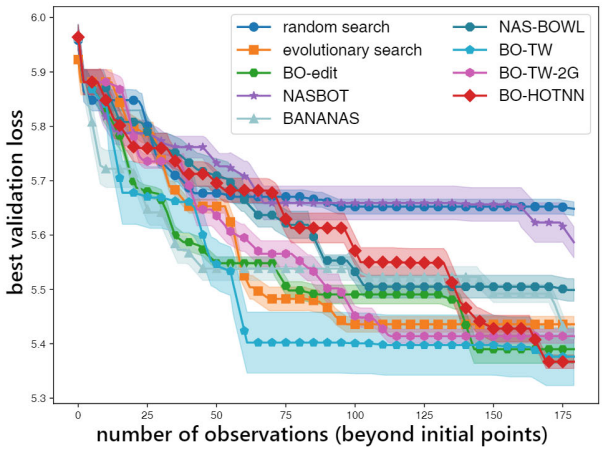


(f) Room layout

Figure S3. The best found validation loss over the number of iterations (beyond initial points) of various NAS methods on the (a-g) TransNAS-Bench-101 and (h) NAS-Bench-101 benchmark.



(g) Semantic segmentation



(h) NAS-Bench-101

Figure S3. The best found validation loss over the number of iterations (beyond initial points) of various NAS methods on the (a-g) TransNAS-Bench-101 and (h) NAS-Bench-101 benchmark.

[16] Soheil Kolouri, Se Rim Park, Matthew Thorpe, Dejan Slepcev, and Gustavo K Rohde. Optimal mass transport: Signal processing and machine-learning applications. *IEEE Signal Processing Magazine*, 34(4):43–59, 2017. 1

[17] Soheil Kolouri, Yang Zou, and Gustavo K. Rohde. Sliced wasserstein kernels for probability distributions. In *CVPR*, 2016. 4

[18] H. J. Kushner. A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise. *Journal of Basic Engineering*, 86(1):97–106, 03 1964. 4

[19] Tam Le, Makoto Yamada, Kenji Fukumizu, and Marco Cuturi. Tree-sliced variants of wasserstein distances. In *NeurIPS*, 2019. 3

[20] John Lee, Max Dabagia, Eva L Dyer, and Christopher J Rozell. Hierarchical optimal transport for multimodal distribution alignment. In *NeurIPS*, pages 13475–13485, 2019. 1, 2

[21] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. In *ICLR*, 2018. 1

[22] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *ICLR*, 2019. 6

[23] Antoine Liutkus, Umut Simsekli, Szymon Majewski, Alain Durmus, and Fabian-Robert Stöter. Sliced-wasserstein flows: Nonparametric generative modeling via optimal transport and diffusions. In *ICML*, pages 4104–4113, 2019. 1

[24] Jisoo Mok, Byunggook Na, Ji-Hoon Kim, Dongyoon Han, and Sungroh Yoon. Demystifying the neural tangent kernel from a practical perspective: Can it be trusted for neural architecture search without training? In *CVPR*, pages 11861–11870, 2022. 2

[25] Vu Nguyen, Tam Le, Makoto Yamada, and Michael A Osborne. Optimal transport kernels for sequential and parallel neural architecture search. In *ICML*, pages 8084–8095, 2021. 2, 6, 7

[26] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019. 1

[27] Binxin Ru, Xingchen Wan, Xiaowen Dong, and Michael Osborne. Interpretable neural architecture search via bayesian optimisation with weisfeiler-lehman kernels. In *ICLR*, 2020. 2, 6, 7

[28] Tim Salimans, Han Zhang, Alec Radford, and Dimitris Metaxas. Improving gans using optimal transport. In *ICLR*, 2018. 1

[29] Filippo Santambrogio. Optimal transport for applied mathematicians. *Progress in Nonlinear Differential Equations and Their Applications*, 87:XXVII, 353, 2017. 1

[30] Bernhard Schmitzer and Christoph Schnörr. A hierarchical approach to optimal transport. In *SSVM*, pages 452–464, 2013. 1

[31] Vivien Seguy, Bharath Bhushan Damodaran, Remi Flamary, Nicolas Courty, Antoine Rolet, and Mathieu Blondel. Large scale optimal transport and mapping estimation. In *ICLR*, 2018. 1

[32] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: a review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015. 2

[33] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *ICML*, 2010. 4

[34] Fariborz Taherkhani, Ali Dabouei, Sobhan Soleymani, Jeremy Dawson, and Nasser M Nasrabadi. Transporting labels via hierarchical optimal transport for semi-supervised learning. In *ECCV*, pages 509–526, 2020. 1

[35] Vayer Titouan, Nicolas Courty, Romain Tavenard, and Rémi Flamary. Optimal transport for structured data with application on graphs. In *ICML*, pages 6275–6284, 2019.

[36] Alexander Tong, Jessie Huang, Guy Wolf, David Van Dijk, and Smita Krishnaswamy. TrajectoryNet: A dynamic optimal transport network for modeling cellular dynamics. In *ICML*, pages 9526–9536, Jul 2020. 1

[37] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer, 2009. 1

[38] Colin White, Willie Neiswanger, and Yash Savani. Bananas: Bayesian optimization with neural architectures for neural architecture search. In *AAAI*, number 12, pages 10293–10301, 2021. 2, 4, 7

[39] Christopher K Williams and Carl Edward Rasmussen. *Gaussian Processes for Machine Learning*, volume 2. MIT press Cambridge, MA, 2006. 4

[40] Renjun Xu, Pelen Liu, Liyan Wang, Chao Chen, and Jindong Wang. Reliable weighted optimal transport for unsupervised domain adaptation. In *CVPR*, June 2020. 1

[41] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. In *ICLR*, 2020. 6

[42] Anna Yeaton, Rahul G Krishnan, Rebecca Mieloszyk, David Alvarez-Melis, and Grace Huynh. Hierarchical optimal transport for comparing histopathology datasets. *arXiv preprint arXiv:2204.08324*, 2022. 1

[43] Mikhail Yurochkin, Sebastian Claici, Edward Chien, Farzaneh Mirzazadeh, and Justin Solomon. Hierarchical optimal transport for document representation. In *NeurIPS*, pages 1601–1611, 2019. 1