IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 6, 2022, accepted April 27, 2022, date of publication May 2, 2022, date of current version May 6, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3171741

Fisher Task Distance and Its Application in Neural Architecture Search

CAT P. LE, MOHAMMADREZA SOLTANI, JUNCHENG DONG, AND VAHID TAROKH, (Fellow, IEEE)
Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA
Corresponding author: Cat P. Le (cat.le@duke.edu)

This work was supported in part by the Army Research Office under Grant W911NF-15-1-0479.

ABSTRACT We formulate an asymmetric (or non-commutative) distance between tasks based on Fisher Information Matrices, called Fisher task distance. This distance represents the complexity of transferring the knowledge from one task to another. We provide a proof of consistency for our distance through theorems and experiments on various classification tasks from MNIST, CIFAR-10, CIFAR-100, ImageNet, and Taskonomy datasets. Next, we construct an online neural architecture search framework using the Fisher task distance, in which we have access to the past learned tasks. By using the Fisher task distance, we can identify the closest learned tasks to the target task, and utilize the knowledge learned from these related tasks for the target task. Here, we show how the proposed distance between a target task and a set of learned tasks can be used to reduce the neural architecture search space for the target task. The complexity reduction in search space for task-specific architectures is achieved by building on the optimized architectures for similar tasks instead of doing a full search and without using this side information. Experimental results for tasks in MNIST, CIFAR-10, CIFAR-100, ImageNet datasets demonstrate the efficacy of the proposed approach and its improvements, in terms of the performance and the number of parameters, over other gradient-based search methods, such as ENAS, DARTS, PC-DARTS.

INDEX TERMS Task affinity, fisher information matrix, neural architecture search.

I. INTRODUCTION

This paper is motivated by a common assumption made in transfer and lifelong learning [1]–[11]: similar tasks usually have similar neural architectures and shares common knowledge. In multi-task learning [12], it also found that utilizing the shared knowledge of related tasks can boost the performance of training. However, if the non-related tasks were trained together, the overall performance degrades significantly. Up until now, in order to identify the closeness of tasks, people often depend on the domain knowledge, or brute-force approach with transfer learning. Building on this observation, we propose a novel task metric that is easy and efficient to compute, and represents the complexity of transferring the knowledge of one task to another, called *Fisher Task* distance (FTD). This distance is a non-commutative measure by design, since transferring the knowledge of a comprehensive task to a simple one is much easier than the other way around. FTD is defined in terms of the Fisher Information matrix defined as the second-derivative of the loss function with respect to the parameters of the model under consideration. By definition, FTD is always greater or equal to zero, where the equality holds if and only if it is the distance from a task to itself. To show that our task distance is mathematically well-defined, we provide several theoretical analysis. Moreover, we empirically verify that the FTD is statistically a consistent distance through experiments on numerous tasks and datasets, such as MNIST, CIFAR-10, CIFAR-100, ImageNet, and Taskonomy [11]. In particular, Taskonomy dataset indicate our computational efficiency while achieving similar results in term of distance between tasks as the brute-force approach proposed by [11]. Next, we instantiate our proposed task distance on Neural Architecture Search (NAS). In the traditional NAS [13]–[17], the search for the architecture of the target task often starts from scratch, with some prior knowledge about the dataset to initialize the search space. The past learned tasks are not considered as the useful knowledge for the target task. Being motivated by the advantages of transfer learning, we would like to apply the knowledge from previous

The associate editor coordinating the review of this manuscript and approving it for publication was Shovan Barma.

learned tasks to the target task to remove the dependency on the domain knowledge. Here, we construct a continuous NAS framework using our proposed task distance, that is capable of learning an appropriate architecture for a target task based on its similarity to other learned tasks. For a target task, the closest task in a given set of baseline tasks is identified and its corresponding architecture is used to construct a neural search space for the target task without requiring prior domain knowledge. Subsequently, our gradient-based search algorithm called FUSE [18] is applied to discover an appropriate architecture for the target task. Briefly, the utilization of the related tasks' architectures helps reduce the dependency on prior domain knowledge, consequently reducing the search time for the final architecture and improving the robustness of the search algorithm. Extensive experimental results for the classification tasks on MNIST [19], CIFAR-10, CIFAR-100 [20], and ImageNet [21] datasets demonstrate the efficacy and superiority of our proposed approach compared to the state-of-the-art approaches.

In this paper, we provide the relevant works in transfer learning and neural architecture search literature in Section II. Next, we introduce the definition of tasks, the Fisher task distance, and theoretical analysis in Section III. The continuous NAS framework, called Task-aware Neural Architecture Search, is proposed in Section IV. Lastly, we provide the experimental studies of our distance and its application in NAS in Section V.

II. RELATED WORKS

The task similarity has been mainly considered in the transfer learning (TL) literature. Similar tasks are expected to have similar architectures as manifested by the success of applying transfer learning in many applications [1]–[11]. However, the main goal in TL is to transfer trained weights from a related task to a target task. Recently, a measure of closeness of tasks based on the Fisher Information matrix has been used as a regularization technique in transfer learning [22] and continual learning [23] to prevent catastrophic forgetting. Additionally, the task similarity has also been investigated between visual tasks in [11], [12], [24]–[27]. These works only focus on weight-transfer and do not utilize task similarities for discovering the high-performing architectures. Moreover, the introduced measures of similarity from these works are often assumed to be symmetric which is not typically a realistic assumption. For instance, consider learning a binary classification between cat and dog images in the CIFAR-10 dataset. It is easier to learn this binary task from a pre-trained model on the entire CIFAR-10 images with 10 classes than learning the 10-class classification using the knowledge of the binary classification in CIFAR-10.

In the context of the neural architecture search (NAS), the similarity between tasks has not been explicitly considered. Most NAS methods focus on reducing the complexity of search by using an explicit architecture search domain and specific properties of the given task at hand. NAS techniques have been shown to offer competitive or even better performance to those of hand-crafted architectures. In general, these techniques include approaches based on evolutionary algorithms [28], [29], reinforcement learning [16], [17], and optimization-based approaches [14]. However, most of these NAS methods are computationally intense and require thousands of GPU-days operations. To overcome the computational issues and to accelerate the search procedure, recently, differentiable search methods [13], [30]–[37] have been proposed. These methods, together with random search methods [38]–[40] and sampling sub-networks from one-shot super-networks [39], [41]–[44], can significantly speed up the search time in the neural architecture space. Additionally, some reinforcement learning methods with weight-sharing [45]–[47], similarity architecture search [18], [48], neural tangent kernel [49], network transformations [50]–[53], and few-shot approaches [54]–[57] have yielded time-efficient NAS methods. None of these approaches consider the task similarities in their search space. In contrast, our approach exploits asymmetric relation between tasks to reduce the search space and accelerate the search procedure.

III. FISHER TASK DISTANCE

Before discussing the task distance, we recall the definition of the Fisher Information matrix for a neural network.

Definition 1 (Fisher Information Matrix): Let N be a neural network with data X, weights θ, and the negative log-likelihood loss function $L(\theta) := L(\theta, X)$. The Fisher Information Matrix is defined as follows:

$$F(\theta) = \mathbb{E}\Big[\nabla_\theta L(\theta) \nabla_\theta L(\theta)^T\Big] = -\mathbb{E}\Big[\mathbf{H}\big(L(\theta)\big)\Big], \tag{1}$$

where $\mathbf{H}$ *is the Hessian matrix, i.e.,* $\mathbf{H}\big(L(\theta)\big) = \nabla^2_\theta L(\theta)$, *and expectation is taken w.r.t the distribution of data.*

We use the empirical Fisher Information Matrix computed as follows:

$$\hat{F}(\theta) = \frac{1}{|X|} \sum_{i \in X} \nabla_\theta L^i(\theta) \nabla_\theta L^i(\theta)^T, \tag{2}$$

where $L^i(\theta)$ is the loss value for the i^{th} data point in X.[1]

Consider a dataset $X = X^{(1)} \cup X^{(2)}$ with $X^{(1)}$ and $X^{(2)}$ denote the training and the test data, respectively. Let's denote a task T and its corresponding dataset X jointly by a pair (T, X). Also, let $\mathcal{P}_N((T, X^{(2)})) \in [0, 1]$ be a function that measures the performance of a given architecture N on a task T using the test dataset $X^{(2)}$.

Definition 2 (ε-approximation Network for Task T): An architecture N is called an ε-approximation network for task T and the corresponding data X if it is trained using training data $X^{(1)}$ such that $\mathcal{P}_N(T, X^{(2)}) \geq 1 - \varepsilon$, for a given $0 < \varepsilon < 1$.

In practice, architectures for ε-approximation networks for a given task T are selected from a pool of well-known hand-designed architectures.

Definition 3 (Fisher Task Distance): Let a and b be two tasks with N_a and N_b denote their corresponding

[1] We use F instead of $\hat{F}$ onward for the notation simplicity.

Algorithm 1 Fisher Task Distance

Data: $X_a = \{X_a^{(1)} \cup X_a^{(2)}\}, X_b = \{X_b^{(1)} \cup X_b^{(2)}\}$
Input: ε-approx. network N
Output: Distance from task a to task b
Function `TaskDistance` $(X_a^{(1)}, X_b, N)$ **:**
  Initialize N_a, N_b from N
  Train N_a using $X_a^{(1)}$, N_b using $X_b^{(1)}$
  Compute $F_{a,b}$ (equation 2) using $X_b^{(2)}$ on N_a
  Compute $F_{b,b}$ (equation 2) using $X_b^{(2)}$ on N_b
  return $d[a,b] = \frac{1}{\sqrt{2}} \left\| F_{a,b}^{1/2} - F_{b,b}^{1/2} \right\|_F$

ε-approximation networks, respectively. Let $F_{a,b}$ be the Fisher Information Matrix of N_a with the dataset $X_b^{(2)}$ from the task b, and $F_{b,b}$ be the Fisher Information Matrix of N_b with the dataset $X_b^{(2)}$ from the task b. We define the FTD from the task a to the task b based on Fréchet distance as follows:

$$d[a,b] = \frac{1}{\sqrt{2}} Tr\Big(F_{a,b} + F_{b,b} - 2(F_{a,b}F_{b,b})^{1/2}\Big)^{1/2}, \quad (3)$$

where Tr denotes the trace of a matrix.

In this paper, we use the diagonal approximation of the Fisher Information matrix since computing the full Fisher matrix is prohibitive in the huge space of neural network parameters. We also normalize them to have a unit trace. As a result, the FTD in (3) can be simplified as follows:

$$\begin{aligned} d[a,b] &= \frac{1}{\sqrt{2}} \left\| F_{a,b}^{1/2} - F_{b,b}^{1/2} \right\|_F \\ &= \frac{1}{\sqrt{2}} \left[\sum_i \Big((F_{a,b}^{ii})^{1/2} - (F_{b,b}^{ii})^{1/2} \Big)^2 \right]^{1/2}, \end{aligned} \quad (4)$$

where F^{ii} is the i^{th} diagonal entry of the Fisher Information matrix. The procedure to compute the FTD is given by Algorithm 1. The FTD ranges from 0 to 1, with the distance $d = 0$ denotes a perfect similarity and the distance $d = 1$ indicates a perfect dissimilarity. As equation (4) shows, the FTD is *asymmetric*. This aligns with human's common sense that it is often easier to transfer knowledge of a complex task to a simple task than vice versa. Note that the FTD depends on the choice of the ε-approximation networks. That is, by using different network architectures to represent the tasks, the computed task distance can be different. This is similar to the human being's perception: Two people can provide different values of the distance between two tasks. However, it is not likely that their perceptions are different. For instance, people can give different values on the similarity between cat and tiger, but they agree that both cat and tiger are much different from car or plane. In the first part of our experiments, we empirically show this intuition and illustrate that although the computed distances may be different due to the choice of ε-approximation networks, the trend of these distances remains consistent across different architectures.

Definition 4 (Structurally-Similar ε-approximation Networks w.r.t. (T, X)): Two ε-approximation networks N_1 and N_2 are called structurally-similar w.r.t. (T, X) if they have exact architecture (the same number of units, the same number of layers, etc), and they are trained on task T using the training dataset $X^{(1)}$.

Next, we present some theoretical justification for our measure of task similarity. All the proofs are provided in the appendix. Firstly, if we train any pair of structurally-similar ε-approximation networks w.r.t some target (T, X) with the same conditions (e.g., initialization, batch order), the FTD between this pair of networks using the test dataset $X^{(2)}$ is zero. Formally, we have the following proposition:

Proposition 1: Let X be the dataset for the target task T. For any pair of structurally-similar ε-approximation networks w.r.t (T, X) using the full or stochastic gradient descent algorithm with the same initialization settings, learning rate, and the same order of data batches in each epoch for the SGD algorithm, the Fisher task distance between the above pair of ε-approximation networks is always zero.

In this proposition, all the training settings were assumed to be the same for two structurally-similar ε-approximation networks w.r.t (T, X). However, an important question is whether the FTD is still a *well-defined* measure regardless of the initial settings, learning rate, and the order of data batches. That is, if we train two structurally-similar ε-approximation networks w.r.t (T, X) using SGD with different settings, will the FTD between N_1 and N_2, as defined in Equation (4), be (close) zero? We answer this question affirmatively assuming a strongly convex loss function. To this end, we invoke Polyak theorem [58] on the convergence of the average sequence of estimation in different epochs from the SGD algorithm. While the loss function in a deep neural network is not a strongly convex function, establishing the fact that the FTD is mathematically well-defined even for this case is an important step towards the more general case in a deep neural network and a justification for the success of our empirical study. In addition, there are some recent works that try to establish Polyak theorem [58] for the convex or even some non-convex functions in an (non)-asymptotic way [59]. Here, we rely only on the asymptotic version of the theorem proposed originally by [58]. We first recall the definition of the strongly convex function.

Definition 5 (Strongly Convex Function): A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is strongly convex if for all $x, y \in \mathbb{R}^n$ and some $\mu > 0$, f satisfies the following inequality:

$$f(y) \geq f(x) + \nabla(f)^T (y - x) + \mu ||y - x||_2^2. \quad (5)$$

Through this paper, we denote ℓ_∞-norm of a matrix B as $||B||_\infty = \max_{i,j} |B_{ij}|$. Also, $|S|$ means the size of a set S.

Theorem 1: Let X be the dataset for the target task T. Consider N_1 and N_2 as two structurally-similar ε-approximation networks w.r.t. (T, X) respectively with the set of weights θ_1 and θ_2 trained using the SGD algorithm where a diminishing learning rate is used for updating weights. Assume that the loss function L for the task T is strongly convex, and its

3rd-order continuous derivative exists and bounded. Let the noisy gradient function in training N_1 and N_2 networks using SGD algorithm be given by:

$$g(\theta_{it}, \epsilon_{it}) = \nabla L(\theta_{it}) + \epsilon_{it}, \quad \text{for } i = 1, 2, \tag{6}$$

where θ_{it} is the estimation of the weights for network N_i at time t, and $\nabla L(\theta_{it})$ is the true gradient at θ_{it}. Assume that ϵ_{it} satisfies $\mathbb{E}[\epsilon_{it}|\epsilon_{i0}, \ldots, \epsilon_{it-1}] = 0$, and satisfies $s = \lim_{t\to\infty} \left|\left|[\epsilon_{it}\epsilon_{it}^T|\epsilon_{i0}, \ldots, \epsilon_{it-1}]\right|\right|_\infty < \infty$ almost surely (a.s.). Then the Fisher task distance between N_1 and N_2 computed on the average of estimated weights up to the current time t converges to zero as $t \to \infty$. That is,

$$d_t = \frac{1}{\sqrt{2}} \left|\left| \bar{F}_{1_t}^{1/2} - \bar{F}_{2_t}^{1/2} \right|\right|_F \xrightarrow{\mathcal{D}} 0, \tag{7}$$

where $\bar{F}_{i_t} = F(\bar{\theta}_{i_t})$ with $\bar{\theta}_{i_t} = \frac{1}{t}\sum_t \theta_{i_t}$, for $i = 1, 2$.

In our experiments, we found out that the weights averages $\bar{\theta}_{i_t}$, $i = 1, 2$ can be replaced with only their best estimates for θ_{i_t}, $i = 1, 2$, respectively. Next, we show that the FTD is also a *well-defined* measure between two task-data set pairs. In other words, the (asymmetric) FTD from the task (T_A, X_A) to the task (T_B, X_B) approaches a constant value regardless of the initialization, learning rate, and the order of data batches in the SGD algorithm provided that X_A and X_B have the same distribution.

Theorem 2: Let X_A be the dataset for the task T_A with the objective function L_A, and X_B be the dataset for the task T_B with the objective function L_B. Assume X_A and X_B have the same distribution. Consider an ε-approximation network N trained using both datasets $X_A^{(1)}$ and $X_B^{(1)}$ respectively with the objective functions L_A and L_B to result weights θ_{A_t} and θ_{B_t} at time t. Under the same assumptions on the moment of gradient noise in SGD algorithm and the loss function stated in Theorem 1, the FTD from the task A to the task B computed from the Fisher Information matrices of the average of estimated weights up to the current time t converges to a constant as $t \to \infty$. That is,

$$d_t = \frac{1}{\sqrt{2}} \left\| \bar{F}_{A_t}^{1/2} - \bar{F}_{B_t}^{1/2} \right\|_F \xrightarrow{\mathcal{D}} \frac{1}{\sqrt{2}} \left\| F_A^{*1/2} - F_B^{*1/2} \right\|_F, \tag{8}$$

where $\bar{F}_{A_t}$ is given by $\bar{F}_{A_t} = F(\bar{\theta}_{A_t})$ with $\bar{\theta}_{A_t} = \frac{1}{t}\sum_t \theta_{A_t}$, and $\bar{F}_{B_t}$ is defined in a similar way.

IV. NEURAL ARCHITECTURE SEARCH

In this section, we apply Fisher task distance to the task-aware neural architecture search (TA-NAS) framework [18], which finds the suitable architecture for a target task, based on a set of baseline tasks. Consider a set A consisting of K baseline tasks T_i and its corresponding data set X_i, denoted jointly by pairs (T_i, X_i) for $i = 1, 2, \ldots, K$. Below, TA-NAS framework with the FTD is presented for finding a well-performing architecture for a target task b, denoted by the pair (T_b, X_b), based on the knowledge of architectures of these K learned baseline tasks. We assume that $X_1, X_2, \ldots, X_K$ and X_b are known, and their data points are of the same dimension.

Algorithm 2 TA-NAS Framework

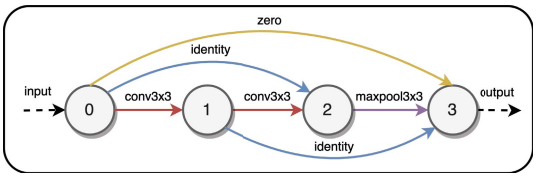
Data: $A = \{(T_1, X_1), \ldots, (T_K, X_K)\}$, $b = (T_b, X_b)$
Input: ε-approx. network N, # of candidates C, $\alpha = 1/|C|$, baseline spaces $\{S_1, \ldots, S_K\}$
Output: Best architecture for b
Function `FUSE` (*candidates C, data X*) **:**
  Relax the output of C (using Softmax function):
  $\bar{c}(X) = \sum_{c\in C} \frac{\exp(\alpha_c)}{\sum_{c'\in C}\exp(\alpha_{c'})} c(X)$
  while α *has not converged* **do**
    Update C by descending $\nabla_w \mathcal{L}_{train}(w; \alpha, \bar{c})$
    Update α by descending $\nabla_\alpha \mathcal{L}_{val}(\alpha; w, \bar{c})$
  return $c^* = \operatorname{argmin}_{c\in C} \alpha_c$
Function `Main`**:**
  for $a \in A$ **do**
    Compute Fisher Task Distance from a to b:
    $d[a, b] =$ `TaskDistance`$(X_a^{(1)}, X_b, N)$
  Select closest task: $a^* = \operatorname{argmin}_{a\in A} d[a, b]$
  Define search space $S = S_{a^*}$
  while *criteria not met* **do**
    Sample C candidates $\in$ S
    $c^* =$ `FUSE`$\big((C \cup c^*), X_b\big)$
  return best architecture c^*

The pipeline of the TA-NAS, whose pseudo-code is given by Algorithm 2, is summarized below:

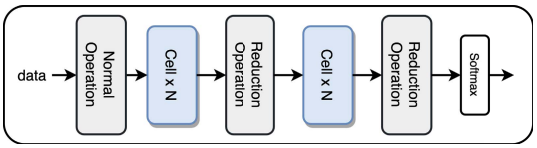
1) **Fisher Task Distance.** First, the FTD of each learned baseline task $a \in A$ to the target task b is computed. The closest baseline task a^*, based on the computed distances, is returned.
2) **Search Space.** Next, a suitable search space for the target task b is determined based on the closest task architecture.
3) **Search Algorithm.** Finally, the FUSE algorithm performs a search within this space to find a well-performing architecture for the target task b.

A. SEARCH SPACE

In an analogous manner to recent NAS techniques [13], [60], our architecture search space is defined by cells and skeletons. A cell, illustrated in Figure 1a, is a densely connected directed-acyclic graph (DAG) of nodes, where nodes are connected by operations. Each node has 2 inputs and 1 output. The operations (e.g., identity, zero, convolution, pooling) are normally set so that the dimension of the output is the same as that of the input. Additionally, a skeleton, illustrated in Figure 1b, is a structure consisting of multiple cells and other operations stacked together, forming a complete architecture. In our framework, the search space for the task is defined in terms of the cells and operations.



(a) A cell structure



(b) A skeleton structure

FIGURE 1. **An example of the cell and the skeleton.**

Given the computed task similarity obtained by FTD, we can identify the closest baseline task(s) to the target task. Build upon this knowledge, we construct the search space for the target task based on the search space of the closest baseline task(s). Since the architecture search space of TA-NAS for a target task is restricted only to the space of the most related task from the baseline tasks, the search algorithm performs efficiently and requires fewer computational resources as it is demonstrated in our experiments on classification tasks in MNIST, CIFAR-10, CIFAR-100, ImageNet datasets.

B. SEARCH ALGORITHM

The Fusion Search (FUSE) is a architecture search algorithm that considers the network candidates as a whole with weighted-sum output, and performs the optimization using gradient descent. Let C be the set of candidate networks which are drawn from the search space. Given $c \in C$ and training data X, denote by $c(X)$ the output of the network candidate c. The FUSE algorithm, is based on the continuous relaxation of the weighted outputs of the network candidates. It is capable of searching through all candidates in the relaxation space, and identifying the most promising network without fully training them. We defined the relaxed output $\bar{c}$ for all of the candidates in C by convex combination, where each weight in the combination given by exponential weights:

$$\bar{c}(X) = \sum_{c \in C} \frac{\exp(\alpha_c)}{\sum_{c' \in C} \exp(\alpha_{c'})} c(X), \tag{9}$$

where $\bar{c}$ is the weighted output of network candidates in C, and α_c is a continuous variable that assigned to candidate c's output. Next, we conduct our search for the most promising candidate in C by jointly training the network candidates, where the output is given by $\bar{c}$, and optimizing their α coefficients. Let X_{train}, X_{val} be the training and validation data set. The training procedure is based on alternative minimization and can be divided into: (i) freeze α coefficients, jointly train network candidates, (ii) freeze network candidates, update α coefficients. Initially, α coefficients are set to $1/|C|$ for all candidates. While freezing α, we update the weights in network candidates by jointly train the relaxed output $\bar{c}$ with cross-validation loss on training data:

$$\min_{w} \mathcal{L}_{train}(w; \alpha, \bar{c}, X_{train}), \tag{10}$$

where w are weights of network candidates in C. Next, the weights in those candidates are fixed while we update the α coefficients on validation data:

$$\min_{\alpha} \mathcal{L}_{val}(\alpha; w, \bar{c}, X_{val}). \tag{11}$$

These steps are repeated until α converges or certain criteria (e.g., a number of iterations, α is greater than a predetermined threshold) are met. The most promising candidate will be selected by: $c^* = \arg\max_{c \in C} \alpha_c$. This training procedure will select the best candidate network among candidates in C without fully training all of them. In order to go through the entire search space, this process is repeated, as more candidates can be drawn from the search space. The search stops when certain criteria, such as the number of iterations, the performance of the current most promising candidate, is met.

V. EXPERIMENTAL STUDY

In this section, we conduct experiments to show the consistency of our Fisher task distance (FTD), as well as its applications in neural architecture search (NAS) and transfer learning (TL).

A. DETAIL OF EXPERIMENTS

In our experiments, the first step is to represent each task and its corresponding dataset by an ε-approximation network. To this end, we train the ε-approximation network with the balanced data from each task. For classification tasks (e.g., tasks in MNIST [19], CIFAR-10 [20], CIFAR-100 [20], ImageNet [21]), three different architectures (e.g., VGG-16 [61], Resnet-18 [62], DenseNet-121 [63]) are chosen as ε-approximation network architectures. The training procedure is conducted in 100 epochs, with Adam optimizer [64], a batch size is set to 128, and cross-validation loss. For image processing tasks in Taskonomy dataset [11], we use an autoencoder as the ε-approximation network. The encoder of the autoencoder consists of one convolutional layer, and two linear layers. The convolutional layer has 3 input channels and 16 output channels with the kernel size equals to 5. We also use the zero padding of size 2, stride of size 4, and dilation equals to 1. The first linear layer has the size of (262144, 512), and the second linear layer has the size of (512, 128). The training procedure is conducted in 20 epochs with Adam optimizer [64], a batch size of 64, and mean-square error loss.

In order to construct the dictionary for baseline tasks, we need to perform the architecture search for these tasks using general search space. This space is defined by cell structures, consisting of 3 or 4 nodes, and 10 operations (i.e., zero, identity, maxpool3 × 3, avepool3 × 3, conv3 × 3, conv5 × 5, conv7 × 7, dil-conv3 × 3, dil-conv5 × 5,

conv7 × 1-1 × 7). After the best cell for each baseline task is founded, we save the structures and operations to the dictionary. Next, we apply the task-aware neural architecture search (TA-NAS) framework [18] to find the best architecture for a target task, given a knowledge of learned baseline tasks.

First, consider a set A consisting of K baseline tasks T_i and its corresponding data set X_i, denoted jointly by pairs (T_i, X_i) for $i = 1, 2, \ldots, K$ where $X_i = X_i^{(1)} \cup X_i^{(2)}$ with $X_i^{(1)}$ and $X_i^{(2)}$ denote the training and the test data, respectively. Now, suppose that for a given ε, the ε-approximation networks for the tasks, (T_i, X_i) for $i = 1, 2, \cdots, K$ denoted by $N_1, N_2, \ldots, N_k$, respectively, where ε is selected such that $\min_{i\in\{1,2,\ldots,K\}} \mathcal{P}_{N_i}(T_i, X_i^{(2)}) \geq 1 - \varepsilon$. Here, the Fisher task distance from each of the baseline tasks to the target is computed. The baseline task with the smallest distance (i.e., the most related task) will be selected and used to construct the restricted search space for the target task. Lastly, the FUSE algorithm is applied to search for the best architecture for the target task from the restricted search space.

The experiment is conducted using NVIDIA GeForce RTX 3090. The source code for the experiments is available at: https://github.com/lephuoccat/Fisher-Information-NAS. Next, we provide the detail of the TA-NAS framework.

B. FISHER TASK DISTANCE (FTD) CONSISTENCY

In this experiment, we show the stability of the FTD by applying our distance on various classification tasks in MNIST, CIFAR-10, CIFAR-100, ImageNet, and Taskonomy datasets with different ε-approximation networks. For each dataset, we define 4 classification tasks, all of which are variations of the full class classification task. For each task, we consider a balanced training dataset. That is, except for the classification tasks with all the labels, only a subset of the original training dataset is used such that the number of training samples across all the class labels to be equal. Additionally, we use 3 widely-used and high-performance architectures as the ε-approximation networks, including VGG-16 [61], ResNet-18 [62], DenseNet-121 [63]. To make sure that our results are statistically significant, we run our experiments 10 times with each of the ε-approximation networks being initialized with a different random seed each time and report the mean and the standard deviation of the computed distance.

1) MNIST

We define 4 tasks on the MNIST dataset. Task 0 and 1 are the binary classification tasks of detecting digits 0 and 6, respectively. Task 2 is a 5-class classification of detecting digits 0, 1, 2, 3, and anything else. Task 3 is the full 10 digits classification. Figure 2 illustrates the mean and standard deviation of the distances between each pair of tasks after 10 runs using 3 different architectures. The columns of the tables denote the distance to the target task and the rows represent the distance from the source tasks. Our results suggest that Task 0 and 1 are highly related, and Task 3 is the closest task to Task 2. Moreover, the relation between tasks remain the same regardless of choosing ε-approximation networks.

2) CIFAR-10

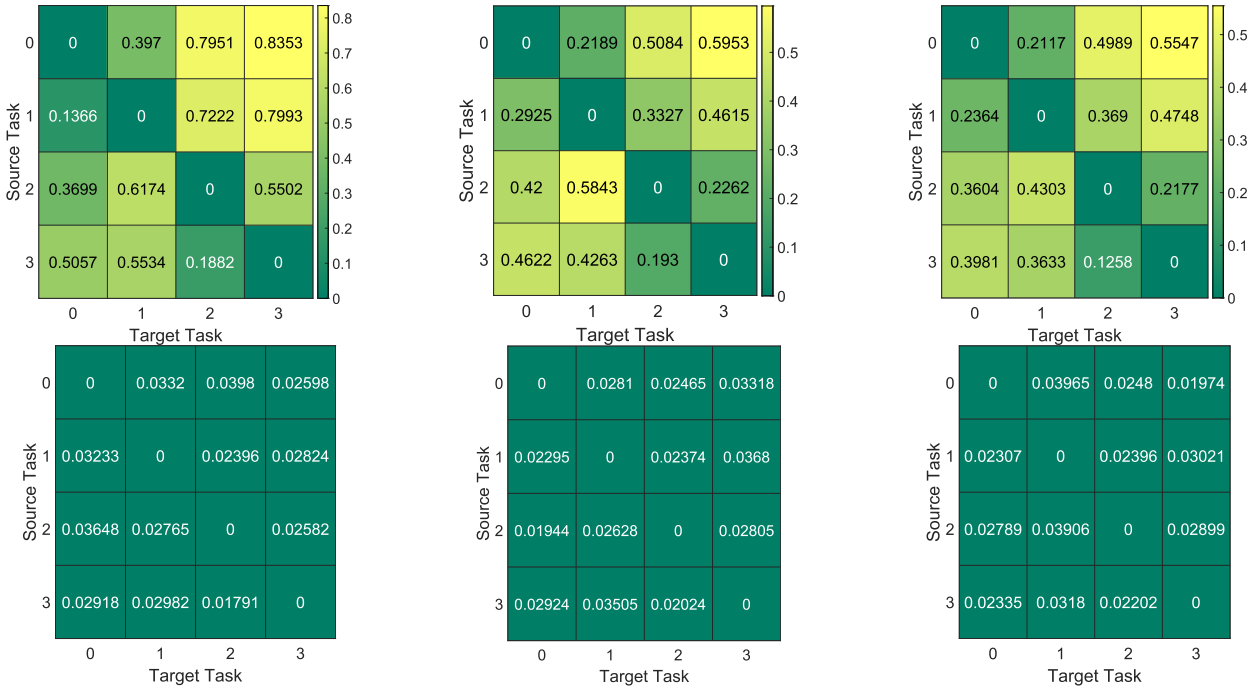
We define 4 tasks in the CIFAR-10 dataset. Task 0 is a binary classification of indicating 3 objects: automobile, cat, ship (i.e., the goal is to decide if the given input image consists of one of these three objects or not). Task 1 is analogous to Task 0 but with different objects: cat, ship, truck. Task 2 is a 4-class classification with labels bird, frog, horse, and anything else. Task 3 is the standard 10 objects classification. Figure 3 illustrates the mean and standard deviation of the distance between CIFAR-10 tasks over 10 trial runs, using 3 different architectures. As we can see in Figures 3, the closest tasks to target tasks in all the tables always result in a unique task no matter what ε-approximation network we choose. Additionally, in Figure 6 (in the appendix), we study the effect of different initial settings, such as training with/without data augmentation, or using unbalanced data set for the above 4 tasks on the CIFAR-10 data set and using VGG-16 as the ε-approximation network. Again the same conclusion about the consistency of the FTD holds.

3) CIFAR-100

We define 4 tasks in the CIFAR-100 dataset, consisting of 100 objects equally distributed in 20 sub-classes, each subclass has 5 objects. We define Task 0 as a binary classification of detecting an object that belongs to vehicles 1 and 2 subclasses or not (i.e., the goal is to decide if the given input image consists of one of these 10 vehicles or not). Task 1 is analogous to Task 0 but with different sub-classes: household furniture and devices. Task 2 is a multi-classification with 11 labels defined on vehicles 1, vehicles 2, and anything else. Finally, Task 3 is defined similarly to Task 2; however, with the 21-labels in vehicles 1, vehicles 2, household furniture, household device, and anything else. Figure 4 illustrates the mean and the standard deviation of the distance between CIFAR-100 tasks after 10 runs using 3 different ε-approximation networks. Here, the closest tasks to any target tasks are distinctive regardless of the choice of the ε-approximation network.

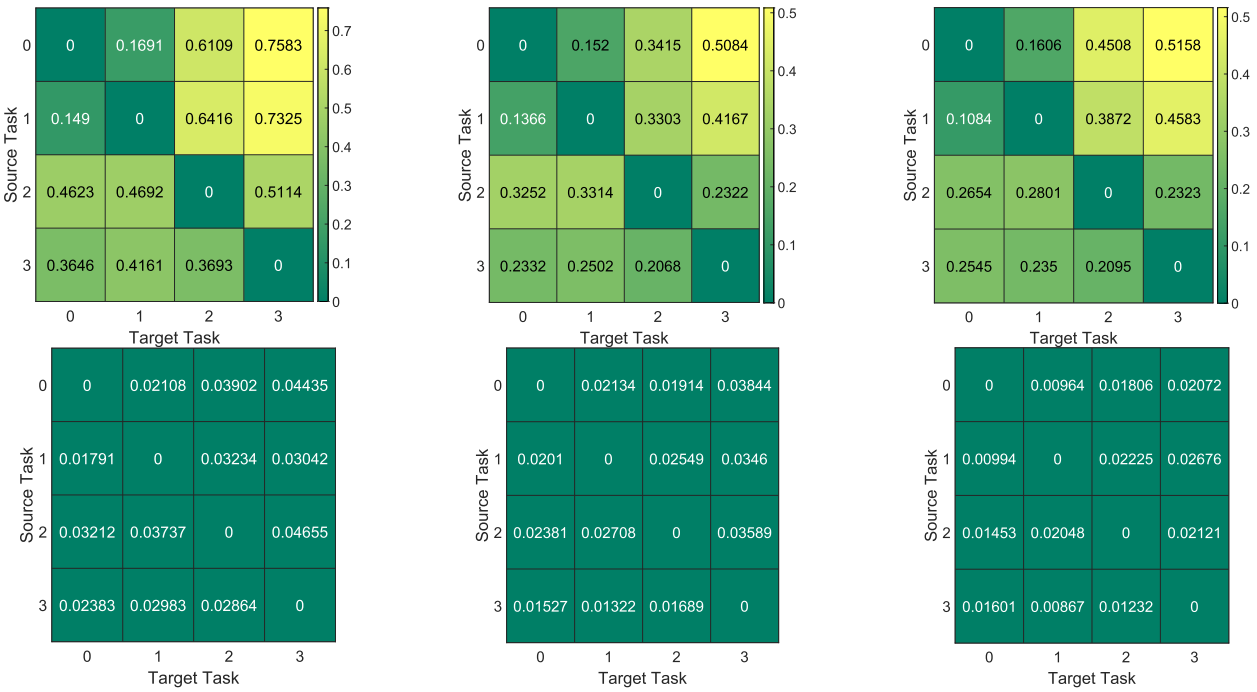
4) ImageNet

Finally, we define four 10-class classification tasks in ImageNet dataset. For each class, we consider 800 for training and 200 for the test samples. The list of 10 classes in Task 0 includes tench, English springer, cassette player, chain saw, church, French horn, garbage truck, gas pump, golf ball, parachute. Task 1 is similar to Task 0; however, instead of 3 labels of tench, golf ball, and parachute, it has samples from the grey whale, volleyball, umbrella classes. In Task 2, we also replace 5 labels of grey whale, cassette player, chain saw, volleyball, umbrella in Task 0 with another 5 labels given by platypus, laptop, lawnmower, baseball, cowboy hat. Lastly, Task 3 is defined as a 10-class classification task with



(a) VGG-16 (b) ResNet-18 (c) DenseNet-121

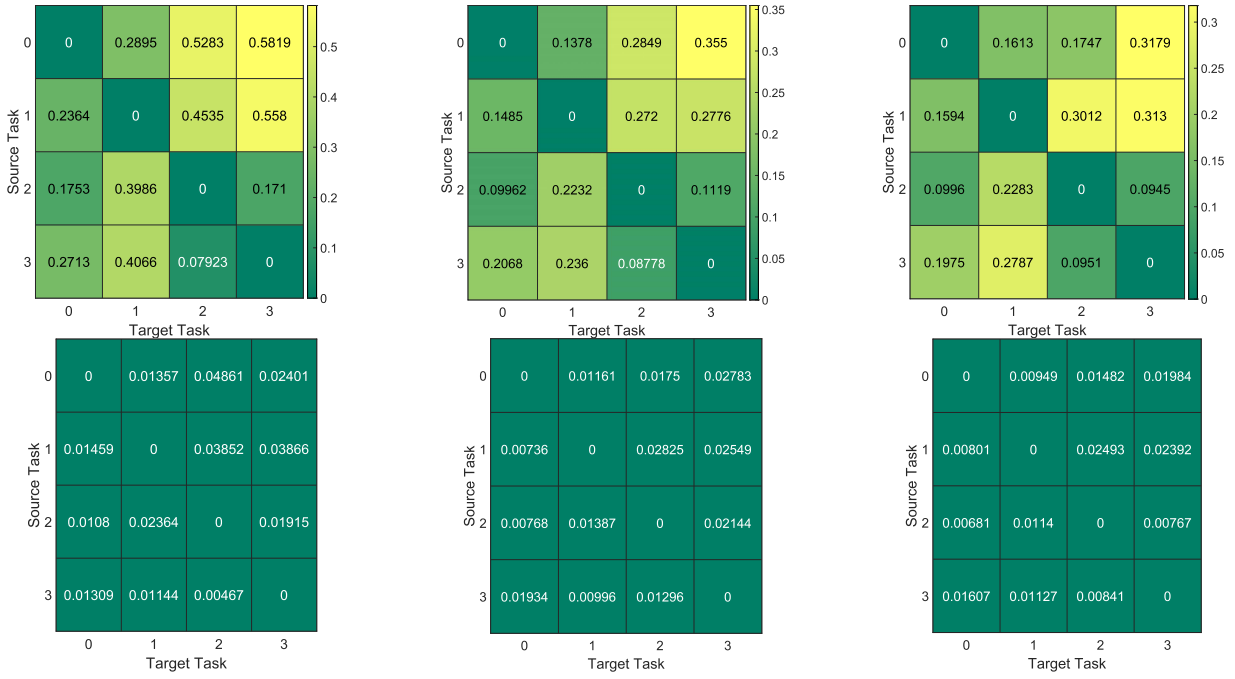
FIGURE 2. Distance from source tasks to the target tasks on MNIST. The top row shows the mean values and the bottom row denotes the standard deviation of distances between classification tasks over 10 different trials.



(a) VGG-16 (b) ResNet-18 (c) DenseNet-121

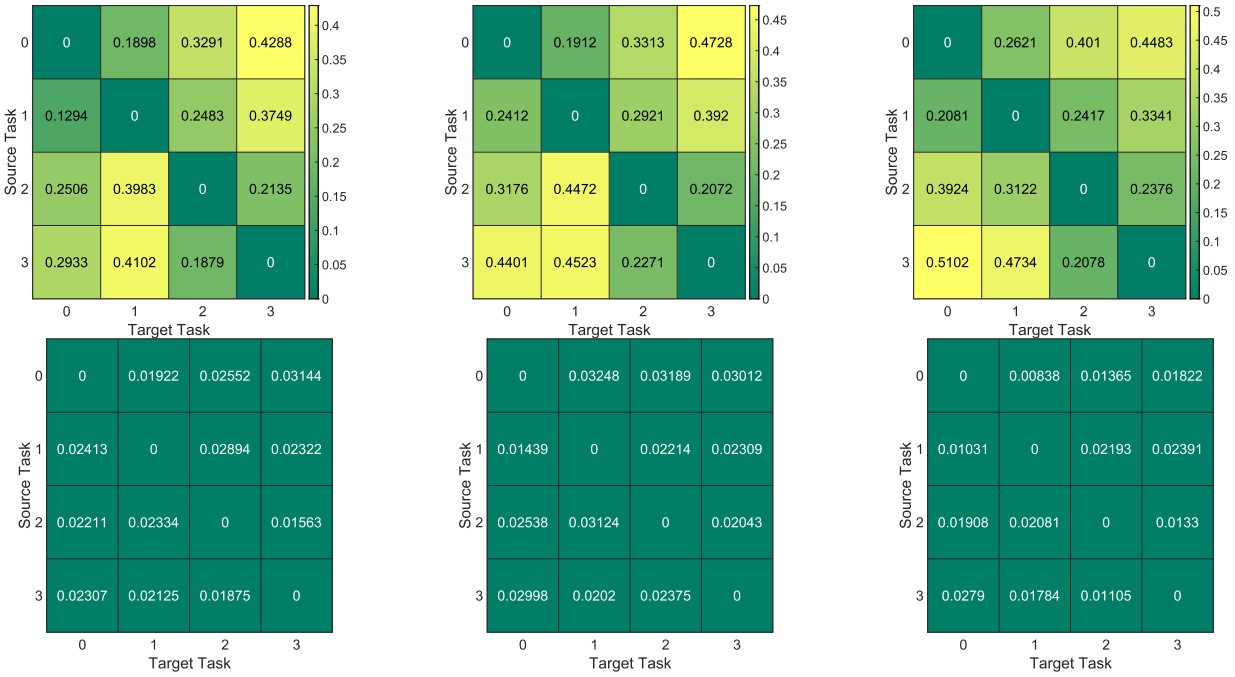
FIGURE 3. Distance from source tasks to the target tasks on CIFAR-10. The top row shows the mean values and the bottom row denotes the standard deviation of distances between classification tasks over 10 different trials.

samples from the following classes: analog clock, candle, sweatshirt, birdhouse, ping-pong ball, hotdog, pizza, school bus, iPod, beaver. The mean and standard deviation tables of the distances between ImageNet tasks for 10 trials with 3 ε-approximation networks are illustrated in Figure 5. Again, it is observed that the order of the distance between the source and target tasks remains the same independent of the ε-approximation networks.



(a) VGG-16 (b) ResNet-18 (c) DenseNet-121

FIGURE 4. Distance from source tasks to the target tasks on CIFAR-100. The top row shows the mean values and the bottom row denotes the standard deviation of distances between classification tasks over 10 different trials.



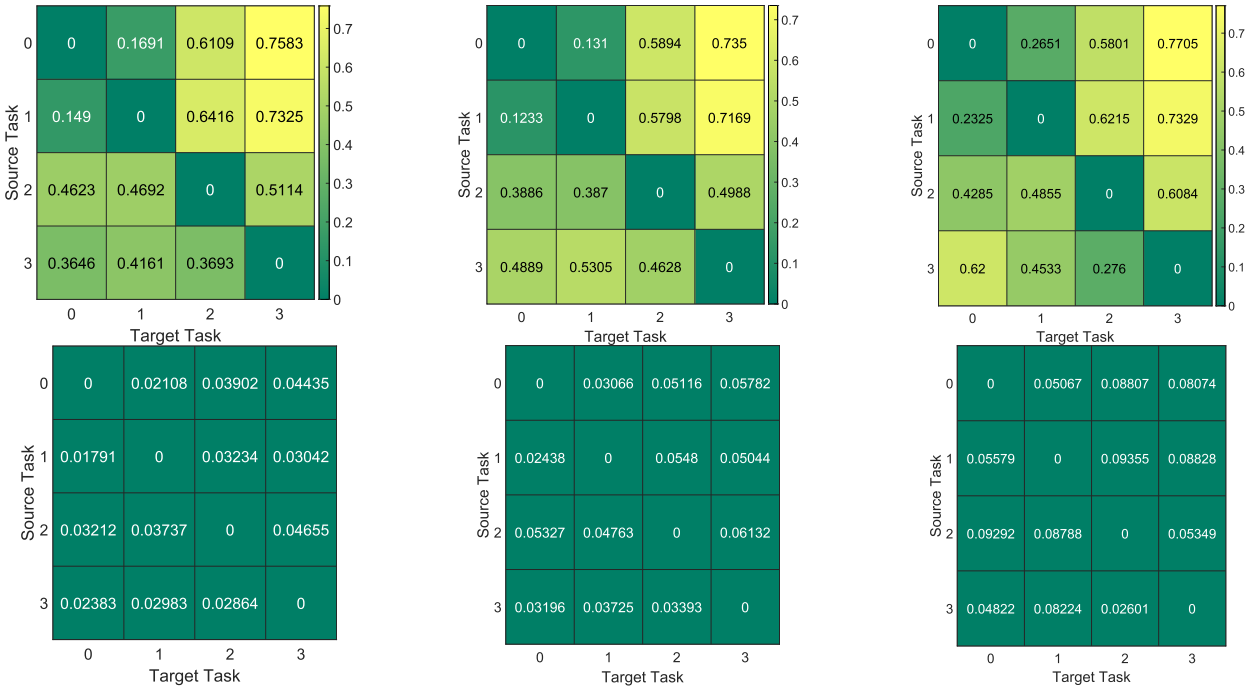
(a) VGG-16 (b) ResNet-18 (c) DenseNet-121

FIGURE 5. Distance from source tasks to the target tasks on ImageNet. The top row shows the mean values and the bottom row denotes the standard deviation of distances between classification tasks over 10 different trials.

Overall, although the value of task distance depends on the ε-approximation network, the trend remains the same across all 3 architectures. In addition, the standard deviation values in the bottom row of the figures suggest that the computed task distance is stable as the fluctuations over the mean values do not show any overlap with each other.

5) TASKONOMY

Here, we apply the FTD to the Taskonomy dataset and compare the task affinity found using our task distance with the brute-force method proposed by [11], and the cosine similarity on the Taskonomy dataset. The Taskonomy dataset is a collection of 512 $\times$ 512 colorful images of varied



(a) data augmentation (b) no data augmentation (c) unbalanced data (full dataset)

FIGURE 6. The effect of different initial settings on computing distance between tasks defined on CIFAR-10 using VGG-16 as the ϵ-approximation network. The top row shows the mean values and the bottom row denotes the standard deviation of distances over 10 different trials.

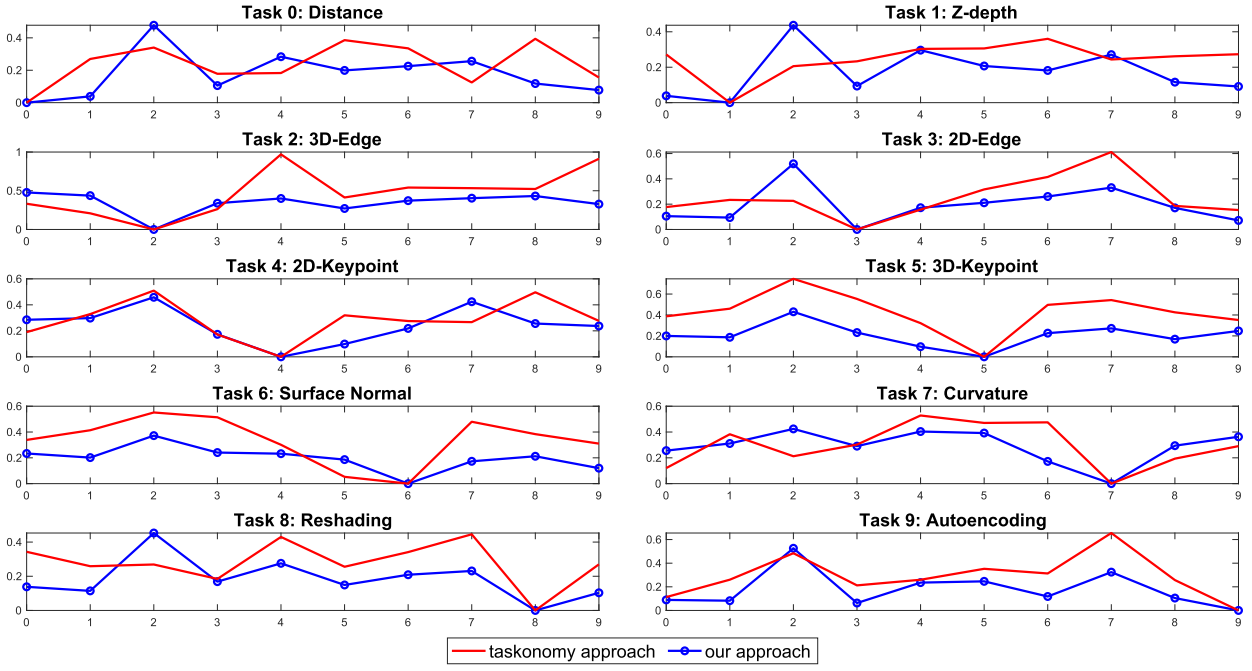


FIGURE 7. The comparison of task affinity between our approach and Taskonomy [11] approach for each task.

indoor scenes. It provides the pre-processed ground truth for 25 vision tasks including semantic and low-level tasks. In this experiment, we consider a set of 10 visual tasks, including: (0) Euclidean distance, (1) z-depth, (2) 3D-edge, (3) 2D-edge (4) 2D-keypoint, (5) 3D-keypoint, (6) surface normal, (7) curvature, (8) reshading, (9) autoencoding. Please see [11] for detailed task descriptions. Each task has 40, 000 training samples and 10, 000 test samples. A deep autoencoder

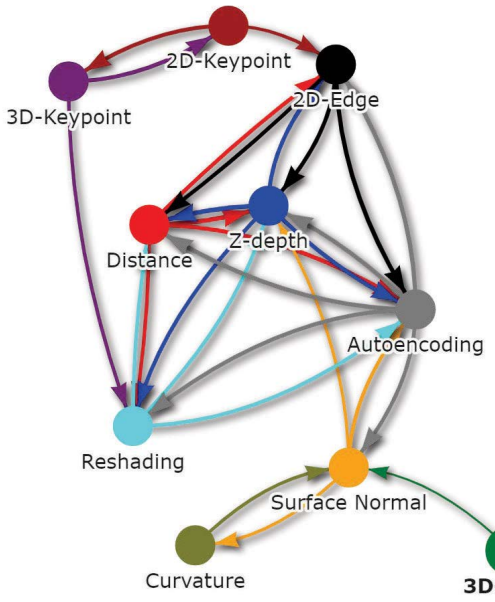


FIGURE 8. The atlas plot of tasks found from our approach indicates the computed relationship between tasks according to locations in space.

architecture, including convolutional and linear layers, with a total of 50.51 Mil parameters, is chosen to be the ε-approximation network for all visual tasks.

In order to use the autoencoder for all the visual tasks without architecture modification, we convert all of the ground truth outputs to three channels. The top panel of Figure 9 shows the mean of our task distance between each pair of tasks over 10 different initial settings in the training of the ε-approximation network. The middle panel indicates the distance founded using the cosine similarity between a pair of Fisher Information matrices. The bottom panel shows the task affinity achieved by the brute-force method from the Taskonomy paper for only a single run. Note that, the task affinities are asymmetric (or non-commutative) in these approaches. As shown in Figure 9, our approach and the brute-force approach have a similar trend in identifying the closest tasks for a given incoming task. The task affinity found by the brute-force approach, however, requires a lot more computations, and does not give a clear boundary between tasks (e.g., difficult to identify the closest tasks to some target tasks). Our approach, on the other hand, is statistical and determines a clear boundary for the identification of related tasks based on the distance. Lastly, the cosine similarity distance between Fisher Information matrices differs from both our approach and the brute-force approach in several tasks, while not showing any clear boundary between tasks. Consequently, the cosine similarity with Fisher Information matrices is not suitable for this dataset. Additionally, Figure 7 illustrates the comparison of task affinity by our approach and by the brute-force approach in the Taskonomy paper. We note that both approaches follow a similar trend for most of the tasks. Additionally, Figure 8 shows the atlas plot of tasks

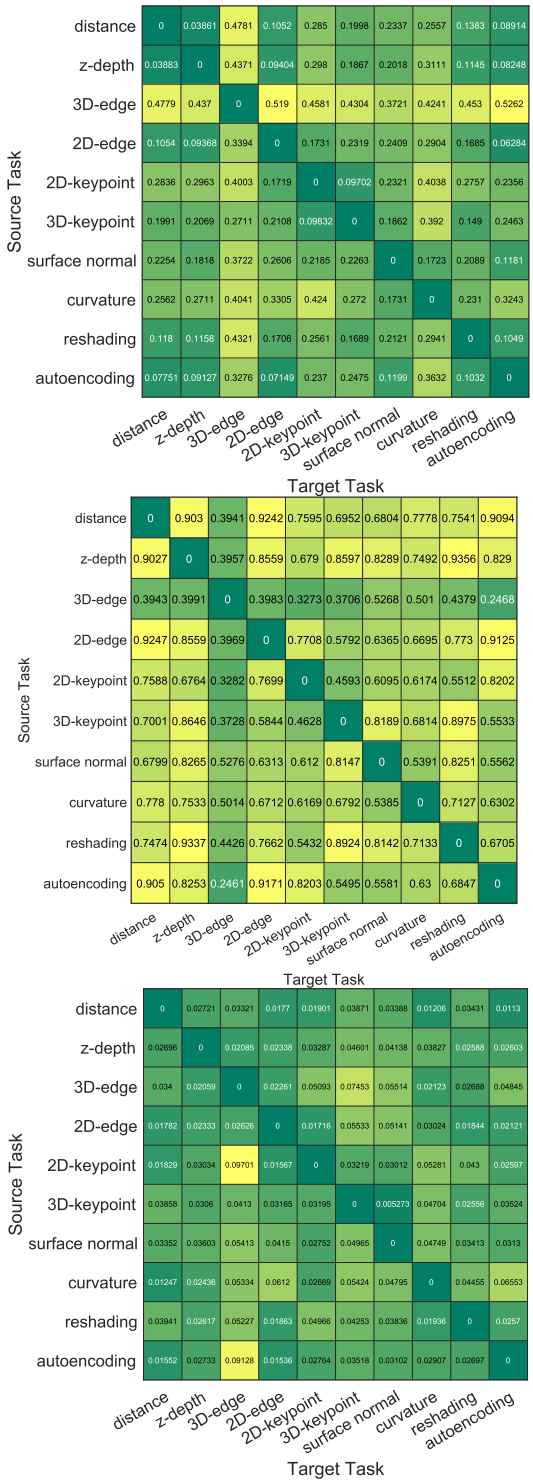


FIGURE 9. Comparison between FTD, cosine similarity, and the Taskonomy approach on tasks from Taskonomy dataset. The top panel shows the averaged distance found by our FTD approach over 10 different trials. The middle panel is the distances obtained by cosine similarity of the Fisher Information matrices. The bottom panel shows the task affinity found by brute-force approach [11] after a single run.

found by our approach, which represents the relationship of tasks according to the location in space. Overall, our FTD is capable of identifying the related tasks with clear statistical boundaries between tasks (i.e., low standard deviation)

TABLE 1. Comparison of our TA-NAS framework with the hand-designed image classifiers, and state-of-the-art NAS methods on Task 2 (binary classification) of MNIST.

Architecture	Accuracy	No. Params. (Mil)	GPU days
VGG-16	99.41	14.72	-
ResNet-18	99.47	11.44	-
DenseNet-121	99.61	6.95	-
Random Search	99.52	2.12	5
ENAS (1st)	94.29	4.60	2
ENAS (2nd)	94.71	4.60	4
DARTS (1st)	98.82	2.17	2
DARTS (2nd)	99.44	2.23	4
PC-DARTS (1st)	98.76	1.78	2
PC-DARTS (2nd)	99.88	2.22	4
TE-NAS	99.71	2.79	2
TA-NAS (ours)	**99.86**	**2.14**	**2**

while requiring significantly less computational resources compared to the brute-force approach (with less statistical significance) in the Taskonomy paper.

C. APPLICATION IN NEURAL ARCHITECTURE SEARCH

In this experiment, we show the application of the FTD in Neural Architecture Search (NAS) by utilizing the computed distances between tasks from the various classification tasks in MNIST, CIFAR-10, CIFAR-100, and ImageNet datasets, and perform the architecture search.

1) MNIST

We consider the problem of learning architecture for the target Task 2 of MNIST dataset, using the other aforementioned tasks as our baseline tasks. It is observed in Figure 2 that Task 3 is the closest one to Task 2. As the result, we apply cell structure and the operations of Task 3 to generate a suitable search space for the target task. The results in Table 1 show the best test accuracy of the optimal architecture found by our method compared to well-known handcrafted networks (i.e., VGG-16, ResNet-18, DenseNet-121), the state-of-the-art NAS methods (i.e., random search algorithm [39], ENAS [45], DARTS [13], PC-DARTS [65], TE-NAS [49]). The architecture discovered by our method is competitive with these networks while it results in a significantly smaller amount of parameters and GPU days.

2) CIFAR-10 & CIFAR-100

We consider the problem of searching for a high-performing and efficient architecture for Task 2 in CIFAR-10, and Task 2 in CIFAR-100 datasets. As observed in Figure 3 and Figure 4, we consider Task 3 as the closest task for both cases. Results in Table 3 and 2 suggest that our constructed architectures for these tasks have higher test accuracy with a fewer number of parameters and GPU days compared to other approaches. The poor performance of ENAS in CIFAR-100 highlights the lack of robustness of this method since its search space is defined to only for full class classification in CIFAR-10.

3) ImageNet

We consider Task 1 in ImageNet dataset as the target task. Based on the computed distances in Figure 5, we use Task 0 as the closest source task to our target task. Table 4 presents results indicating that our model has higher test accuracy with a fewer number of parameters compared to other approaches. In complex datasets (e.g., CIFAR-100, ImageNet), the method with fixed search space (i.e., ENAS) is only capable of finding architectures for tasks in standard datasets, performs poorly compared with other methods in our benchmark. Our experiments suggest that the proposed framework can utilize the knowledge of the most similar task in order to find a high-performing architecture for the target task with a fewer number of parameters.

TABLE 2. Comparison of our TA-NAS framework with the hand-designed image classifiers, and state-of-the-art NAS methods on Task 2 (11-class classification) of CIFAR-100.

Architecture	Accuracy	No. Params. (Mil)	GPU days
VGG-16	83.93	14.72	-
ResNet-18	84.56	11.44	-
DenseNet-121	88.47	6.95	-
Random Search	88.55	3.54	5
ENAS	10.49	4.60	4
DARTS	87.57	3.32	4
PC-DARTS	85.36	2.43	4
TE-NAS	88.92	3.66	4
TA-NAS (ours)	**90.96**	**3.17**	**4**

TABLE 3. Comparison of our TA-NAS framework with the hand-designed image classifiers, and state-of-the-art NAS methods on Task 2 (4-class classification) of CIFAR-10.

Architecture	Accuracy	No. Params. (Mil)	GPU days
VGG-16	86.75	14.72	-
ResNet-18	86.93	11.44	-
DenseNet-121	88.12	6.95	-
Random Search	88.55	3.65	5
ENAS (1st)	73.23	4.60	2
ENAS (2nd)	75.22	4.60	4
DARTS (1st)	90.11	3.12	2
DARTS (2nd)	91.19	3.28	4
PC-DARTS (1st)	92.07	3.67	2
PC-DARTS (2nd)	92.49	3.66	4
TE-NAS	91.02	3.78	2
TA-NAS (ours)	**92.58**	**3.13**	**2**

TABLE 4. Comparison of our TA-NAS framework with the hand-designed image classifiers, and state-of-the-art NAS methods on Task 1 (10-class classification) of ImageNet.

Architecture	Accuracy	No. Params. (Mil)	GPU days
VGG-16	89.88	14.72	-
ResNet-18	91.14	11.44	-
DenseNet-121	94.76	6.95	-
Random Search	95.02	3.78	5
ENAS	33.65	4.60	4
DARTS	95.22	3.41	4
PC-DARTS	88.00	1.91	4
TE-NAS	95.37	4.23	4
TA-NAS (ours)	**95.92**	**3.43**	**4**

VI. CONCLUSION

A task similarity measure based on the Fisher Information matrix has been introduced in this paper. This non-commutative measure called Fisher Task distance (FTD),

represents the complexity of applying the knowledge of one task to another. The theory and experimental experiments demonstrate that the distance is consistent and well-defined. In addition, two applications of FTD, including transfer learning and NAS has been investigated. In particular, the task affinity results found using this measure is well-aligned with the result found using the traditional transfer learning approach. Moreover, the FTD is applied in NAS to define a reduced search space of architectures for a target task. This reduces the complexity of the architecture search, increases its efficiency, and leads to superior performance with a smaller number of parameters.

APPENDIX

Here, we provide the proof of the proposition 1, the proof of the theorem 1, and the proof of the theorem 2.

Proposition 1: Let X be the dataset for the target task T. For any pair of structurally-similar ε-approximation network w.r.t (T, X) using the full or stochastic gradient descent algorithm with the same initialization settings, learning rate, and the same order of data batches in each epoch for the SGD algorithm, the Fisher task distance between the above pair of ε-approximation networks is always zero.

Proof of Proposition 1: Let N_1 and N_2 be two structurally-similar ε-approximation network w.r.t (T, X) trained using the full or stochastic gradient descent algorithm. According to the Definition 4 and assumptions in the proposition, the Fisher Information Matrices of N_1 and N_2 are the same; hence, the Fisher task distance is zero. □

Theorem 1: Let X be the dataset for the target task T. Consider N_1 and N_2 as two structurally-similar ε-approximation networks w.r.t. (T, X) respectively with the set of weights θ_1 and θ_2 trained using the SGD algorithm where a diminishing learning rate is used for updating weights. Assume that the loss function L for the task T is strongly convex, and its 3rd-order continuous derivative exists and bounded. Let the noisy gradient function in training N_1 and N_2 networks using SGD algorithm be given by:

$$g(\theta_{it}, \epsilon_{it}) = \nabla L(\theta_{it}) + \epsilon_{it}, \quad for \;\; i = 1, 2, \tag{12}$$

where θ_{it} is the estimation of the weights for network N_i at time t, and $\nabla L(\theta_{it})$ is the true gradient at θ_{it}. Assume that ϵ_{it} satisfies $\mathbb{E}[\epsilon_{it}|\epsilon_{i0}, \ldots, \epsilon_{it-1}] = 0$, and satisfies $s = \lim_{t\to\infty} \left|\left|[\epsilon_{it}{\epsilon_{it}}^T|\epsilon_{i0}, \ldots, \epsilon_{it-1}]\right|\right|_\infty < \infty$ almost surely (a.s.). Then the Fisher task distance between N_1 and N_2 computed on the average of estimated weights up to the current time t converges to zero as $t \to \infty$. That is,

$$d_t = \frac{1}{\sqrt{2}}\left|\left|\bar{F_1}_t^{1/2} - \bar{F_2}_t^{1/2}\right|\right|_F \xrightarrow{\mathcal{D}} 0, \tag{13}$$

where $\bar{F}_{i_t} = F(\bar{\theta}_{i_t})$ with $\bar{\theta}_{i_t} = \frac{1}{t}\sum_t \theta_{i_t}$, for $i = 1, 2$.

Proof of Theorem 1: Here, we show the proof for the full Fisher Information Matrix; however, the same results holds for the diagonal approximation of the Fisher Information Matrix. Let N_1 with weights θ_1 and N_2 with weights θ_2 be the two structurally-similar ε-approximation networks w.r.t. (T, X). Let n be the number of trainable parameters in N_1 and N_2. Since the objective function is strongly convex and the fact that N_1 and N_2 are structurally-similar ε-approximation networks w.r.t. (T, X), both of these network will obtain the optimum solution θ^* after training a certain number of epochs with stochastic gradient descend. By the assumption on the conditional mean of the noisy gradient function and the assumption on S, the conditional covariance matrix is finite as well, i.e., $C = \lim_{t\to\infty} \mathbb{E}[\epsilon_{it}{\epsilon_{it}}^T|\epsilon_{i0}, \ldots, \epsilon_{it-1}] < \infty$; hence, we can invoke the following result due to Polyak *et al.* [58]:

$$\sqrt{t}(\bar{\theta}_t - \theta^*) \xrightarrow{\mathcal{D}} \mathcal{N}\Big(0, \mathbf{H}\big(L(\theta^*)\big)^{-1} C \mathbf{H}^T \big(L(\theta^*)\big)^{-1}\Big), \tag{14}$$

as $t \to \infty$. Here, $\mathbf{H}$ is Hessian matrix, θ^* is the global minimum of the loss function, and $\bar{\theta}_t = \frac{1}{t}\sum_t \theta_t$. Hence, for networks N_1 and N_2 and from Equation (14), $\sqrt{t}(\bar{\theta}_{1t} - \theta^*)$ and $\sqrt{t}(\bar{\theta}_{2t} - \theta^*)$ are asymptotically normal random vectors:

$$\sqrt{t}(\bar{\theta}_{1t} - \theta^*) \xrightarrow{\mathcal{D}} \mathcal{N}(0, \Sigma_1), \tag{15}$$

$$\sqrt{t}(\bar{\theta}_{2t} - \theta^*) \xrightarrow{\mathcal{D}} \mathcal{N}(0, \Sigma_2), \tag{16}$$

where $\Sigma_1 = \mathbf{H}\big(L(\theta^*)\big)^{-1} C_1 \mathbf{H}^T \big(L(\theta^*)\big)^{-1}$, and $\Sigma_2 = \mathbf{H}\big(L(\theta^*)\big)^{-1} C_2 \mathbf{H}^T \big(L(\theta^*)\big)^{-1}$. The Fisher Information $F(\theta)$ is a continuous and differentiable function of θ. Since it is also a positive definite matrix, $F(\theta)^{1/2}$ is well-defined. Hence, by applying the Delta method to Equation (15), we have:

$$\sqrt{t}(\bar{F_1}_t^{1/2} - {F^*}^{1/2}) \xrightarrow{\mathcal{D}} \mathcal{N}(0, \Sigma_1^*), \tag{17}$$

where $\bar{F}_{1_t} = F(\bar{\theta}_{1_t})$, and the covariance matrix Σ_1^* is given by $\Sigma_1^* = \mathbf{J}_\theta\Big(\mathbf{vec}\big(F(\theta^*)^{1/2}\big)\Big)\Sigma_1 \mathbf{J}_\theta\Big(\mathbf{vec}\big(F(\theta^*)^{1/2}\big)\Big)^T$. Here, **vec**() is the vectorization operator, θ^* is a $n \times 1$ vector of the optimum parameters, $F(\theta^*)$ is a $n \times n$ Matrix evaluated at the minimum, and $\mathbf{J}_\theta(F(\theta^*))$ is a $n^2 \times n$ Jacobian matrix of the Fisher Information Matrix. Similarly, from Equation (16), we have:

$$\sqrt{t}(\bar{F_2}_t^{1/2} - {F^*}^{1/2}) \xrightarrow{\mathcal{D}} \mathcal{N}(0, \Sigma_2^*), \tag{18}$$

where $\Sigma_2^* = \mathbf{J}_\theta\Big(\mathbf{vec}\big(F(\theta^*)^{1/2}\big)\Big)\Sigma_2 \mathbf{J}_\theta\Big(\mathbf{vec}\big(F(\theta^*)^{1/2}\big)\Big)^T$. As a result, $(\bar{F_1}_t^{1/2} - \bar{F_2}_t^{1/2})$ is asymptotically a normal random vector:

$$(\bar{F_1}_t^{1/2} - \bar{F_2}_t^{1/2}) \xrightarrow{\mathcal{D}} \mathcal{N}\Big(0, V_1\Big). \tag{19}$$

where $V_1 = \frac{1}{t}(\Sigma_1^* + \Sigma_2^*)$. As t approaches infinity, $\frac{1}{t}(\Sigma_1^* + \Sigma_2^*) \to 0$. As a result, $d_t = \frac{1}{\sqrt{2}}\left\|\bar{F_1}_t^{1/2} - \bar{F_2}_t^{1/2}\right\|_F \xrightarrow{\mathcal{D}} 0$. □

Theorem 2: Let X_A be the dataset for the task T_A with the objective function L_A, and X_B be the dataset for the task T_B with the objective function L_B. Assume X_A and X_B have the same distribution. Consider an ε-approximation network N trained using both datasets $X_A^{(1)}$ and $X_B^{(1)}$ respectively with the objective functions L_A and L_B to result weights θ_{A_t} and θ_{B_t} at time t. Under the same assumptions on the moment of gradient noise in SGD algorithm and the loss function stated

in Theorem 1, the FTD from the task A to the task B computed from the Fisher Information matrices of the average of estimated weights up to the current time t converges to a constant as $t \to \infty$. That is,

$$d_t = \frac{1}{\sqrt{2}} \left\| \bar{F}_{A_t}^{1/2} - \bar{F}_{B_t}^{1/2} \right\|_F \xrightarrow{\mathcal{D}} \frac{1}{\sqrt{2}} \left\| F_A^{*1/2} - F_B^{*1/2} \right\|_F, \tag{20}$$

where $\bar{F}_{A_t}$ is given by $\bar{F}_{A_t} = F(\bar{\theta}_{A_t})$ with $\bar{\theta}_{A_t} = \frac{1}{t}\sum_t \theta_{A_t}$, and $\bar{F}_{B_t}$ is defined in a similar way.

Proof of Theorem 2: Let θ_{A_t} and θ_{B_t} be the sets of weights at time t from the ε-approximation network N trained using both data sets $X_A^{(1)}$ and $X_B^{(1)}$, respectively with the objective functions L_A and L_B. Since the objective functions are strongly convex, both of these sets of weights will obtain the optimum solutions θ_A^* and θ_B^* after training a certain number of epochs with stochastic gradient descend. Similar to the proof of Theorem 1, by invoking the Polyak *et al.* [58], random vectors of $\sqrt{t}(\bar{\theta}_{At} - \theta_A^*)$ and $\sqrt{t}(\bar{\theta}_{Bt} - \theta_B^*)$ are asymptotically normal:

$$\sqrt{t}(\bar{\theta}_{At} - \theta_A^*) \xrightarrow{\mathcal{D}} \mathcal{N}(0, \Sigma_A), \tag{21}$$

$$\sqrt{t}(\bar{\theta}_{Bt} - \theta_B^*) \xrightarrow{\mathcal{D}} \mathcal{N}(0, \Sigma_B), \tag{22}$$

where $\Sigma_A = \mathbf{H}\big(L(\theta_A^*)\big)^{-1} C_A \mathbf{H}^T \big(L(\theta_A^*)\big)^{-1}$, and $\Sigma_B = \mathbf{H}\big(L(\theta_B^*)\big)^{-1} C_B \mathbf{H}^T \big(L(\theta_B^*)\big)^{-1}$ (here, C_A and C_B denote the conditional covariance matrices, corresponding to the gradient noise in θ_A^* and θ_A^*, respectively). The Fisher Information $F(\theta)$ is a continuous and differentiable function of θ, and it is also a positive definite matrix; thus, $F(\theta)^{1/2}$ is well-defined. Now, by applying the Delta method to Equation (21), we have:

$$(\bar{F}_{A_t}^{1/2} - F_A^{*1/2}) \xrightarrow{\mathcal{D}} \mathcal{N}\big(0, \frac{1}{t}\Sigma_A^*\big), \tag{23}$$

where $\bar{F}_{At} = F(\bar{\theta}_{At})$, and the covariance matrix is given by $\Sigma_A^* = \mathbf{J}_\theta\Big(\mathbf{vec}\big(F(\theta_A^*)^{1/2}\big)\Big)\Sigma_A \mathbf{J}_\theta\Big(\mathbf{vec}\big(F(\theta_A^*)^{1/2}\big)\Big)^T$. Likewise, from Equation (22), we have:

$$(\bar{F}_{B_t}^{1/2} - F_B^{*1/2}) \xrightarrow{\mathcal{D}} \mathcal{N}\big(0, \frac{1}{t}\Sigma_B^*\big), \tag{24}$$

where $\bar{F}_{Bt} = F(\bar{\theta}_{Bt})$, and the covariance matrix is given by $\Sigma_B^* = \mathbf{J}_\theta\Big(\mathbf{vec}\big(F(\theta_B^*)^{1/2}\big)\Big)\mathbf{J}_\theta\Big(\mathbf{vec}\big(F(\theta_B^*)^{1/2}\big)\Big)^T$. From Equation (23) and (24), we obtain:

$$(\bar{F}_{A_t}^{1/2} - \bar{F}_{B_t}^{1/2}) \xrightarrow{\mathcal{D}} \mathcal{N}\Big(\mu_2, V_2\Big), \tag{25}$$

where $\mu_2 = (F_A^{*1/2} - F_B^{*1/2})$ and $V_2 = \frac{1}{t}(\Sigma_A^* + \Sigma_B^*)$. Since $(\bar{F}_{A_t}^{1/2} - \bar{F}_{B_t}^{1/2}) - (F_A^{*1/2} - F_B^{*1/2})$ is asymptotically normal with the covariance goes to zero as t approaches infinity, all of the entries go to zero, we conclude that

$$d_t = \frac{1}{\sqrt{2}} \left\| \bar{F}_{A_t}^{1/2} - \bar{F}_{B_t}^{1/2} \right\|_F \xrightarrow{\mathcal{D}} \frac{1}{\sqrt{2}} \left\| F_A^{*1/2} - F_B^{*1/2} \right\|_F 5. \tag{26}$$

□

REFERENCES

[1] D. L. Silver and K. P. Bennett, "Guest editor's introduction: Special issue on inductive transfer learning," *Mach. Learn.*, vol. 73, no. 3, pp. 215–220, Dec. 2008.

[2] C. Finn, X. Yu Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 512–519.

[3] L. Mihalkova, T. Huynh, and R. J. Mooney, "Mapping and revising Markov logic networks for transfer learning," in *Proc. 22nd Nat. Conf. Artif. Intell.*, vol. 7, 2007, pp. 608–614.

[4] A. Niculescu-Mizil and R. Caruana, "Inductive transfer for Bayesian network structure learning," in *Artificial Intelligence and Statistics*. 2007, pp. 339–346.

[5] Z. Luo, Y. Zou, J. Hoffman, and L. F. Fei-Fei, "Label efficient learning of transferable representations acrosss domains and tasks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 164–176.

[6] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features Off-the-shelf: An astounding baseline for recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Washington, DC, USA, Jun. 2014, pp. 512–519.

[7] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[8] A. Mallya and S. Lazebnik, "Piggyback: Adding multiple tasks to a single, fixed network by learning to mask," 2018, *arXiv:1801.06519*.

[9] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. and Wierstra, "PathNet: Evolution channels gradient descent in super neural networks," 2017, *arXiv:1701.08734*.

[10] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," 2016, *arXiv:1606.04671*.

[11] A. R. Zamir, A. Sax, W. B. Shen, L. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 3712–3722.

[12] T. Standley, A. Zamir, D. Chen, L. Guibas, J. Malik, and S. Savarese, "Which tasks should be learned together in multi-task learning?" in *Proc. Int. Conf. Mach. Learn.*, Jul. 2020, pp. 9120–9132.

[13] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1–13.

[14] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 19–34.

[15] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," 2017, *arXiv:1711.00436*.

[16] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–16.

[17] N. Zhu, Z. Yu, and C. Kou, "A new deep neural architecture search pipeline for face recognition," *IEEE Access*, vol. 8, pp. 91303–91310, 2020.

[18] C. P. Le, M. Soltani, R. Ravier, and V. Tarokh, "Task-aware neural architecture search," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 4090–4094.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[20] A. Krizhevsky and G. Hilton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep. TR-2009, 2009.

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[22] S. Chen, C. Zhang, and M. Dong, "Coupled end-to-end transfer learning with generalized Fisher information," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4329–4338.

[23] K. James, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, and D. Hassabis, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.

[24] A. Pal and V. N. Balasubramanian, "Zero-shot task transfer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019.

[25] K. Dwivedi and G. Roig, "Representation similarity analysis for efficient task taxonomy transfer learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12387–12396.
[26] A. Achille, M. Lam, R. Tewari, A. Ravichandran, S. Maji, C. Fowlkes, S. Soatto, and P. Perona, "Task2 Vec: Task embedding for meta-learning," 2019, *arXiv:1902.03545*.
[27] A. Wang, M. Tarr, and L. Wehbe, "Neural taskonomy: Inferring the similarity of task-derived representations from brain activity," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.
[28] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. Assoc. Adv. Art. Intell. (AAAI)*, 2019, pp. 4780–4789.
[29] Y. Sun, X. Sun, Y. Fang, G. G. Yen, and Y. Liu, "A novel training protocol for performance predictors of evolutionary neural architecture search algorithms," *IEEE Trans. Evol. Comput.*, vol. 25, no. 3, pp. 524–536, Jun. 2021.
[30] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–13.
[31] A. Noy, N. Nayman, T. Ridnik, N. Zamir, S. Doveh, I. Friedman, R. Giryes, and L. Zelnik-Manor, "ASAP: Architecture search, anneal and prune," 2019, *arXiv:1904.04123*.
[32] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, "Neural architecture optimization," in *Proc. Adv. Neural Inf. Proc. Sys. (NIPS)*, 2018, pp. 1–12.
[33] S. Xie, H. Zheng, C. Liu, and L. Lin, "SNAS: Stochastic neural architecture search," 2018, *arXiv:1812.09926*.
[34] A. Wan, X. Dai, P. Zhang, Z. He, Y. Tian, S. Xie, B. Wu, M. Yu, T. Xu, K. Chen, P. Vajda, and J. E. Gonzalez, "FBNetV2: Differentiable neural architecture search for spatial and channel dimensions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 12965–12974.
[35] N. Awad, N. Mallik, and F. Hutter, "Differential evolution for neural architecture search," 2020, *arXiv:2012.06400*.
[36] C. He, H. Ye, L. Shen, and T. Zhang, "MiLeNAS: Efficient neural architecture search via mixed-level reformulation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11993–12002.
[37] Y. Weng, T. Zhou, Y. Li, and X. Qiu, "NAS-Unet: Neural architecture search for medical image segmentation," *IEEE Access*, vol. 7, pp. 44247–44257, 2019.
[38] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, M. Hardt, B. Recht, and A. Talwalkar, "A system for massively parallel hyperparameter tuning," 2018, *arXiv:1810.05934*.
[39] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," in *Uncertainty in Artificial Intelligence*. PMLR, 2020.
[40] K. Yu, C. Sciuto, M. Jaggi, C. Musat, and M. Salzmann, "Evaluating the search phase of neural architecture search," 2019, *arXiv:1902.08142*.
[41] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8697–8710.
[42] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, "Understanding and simplifying one-shot architecture search," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 550–559.
[43] M. Cho, M. Soltani, and C. Hegde, "One-shot neural architecture search via compressive sensing," 2019, *arXiv:1906.02869*.
[44] Z. Yang, Y. Wang, X. Chen, B. Shi, C. Xu, C. Xu, Q. Tian, and C. Xu, "CARS: Continuous evolution for efficient neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1829–1838.
[45] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4095–4104.
[46] X. Song, K. Choromanski, J. Parker-Holder, Y. Tang, Q. Zhang, D. Peng, D. Jain, W. Gao, A. Pacchiano, T. Sarlos, and Y. Yang, "ES-ENAS: Blackbox optimization over hybrid spaces via combinatorial and continuous evolution," 2021, *arXiv:2101.07415*.
[47] G. Bender, H. Liu, B. Chen, G. Chu, S. Cheng, P.-J. Kindermans, and Q. V. Le, "Can weight sharing outperform random architecture search? An investigation with TuNAS," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 14323–14332.
[48] V. Nguyen, T. Le, M. Yamada, and M. A. Osborne, "Optimal transport kernels for sequential and parallel neural architecture search," 2020, *arXiv:2006.07593*.
[49] W. Chen, X. Gong, and Z. Wang, "Neural architecture search on ImageNet in four GPU hours: A theoretically inspired perspective," 2021, *arXiv:2102.11535*.
[50] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *Proc. Assoc. Adv. Art. Intell. (AAAI)*, 2018, pp. 1–8.
[51] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via Lamarckian evolution," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–23.
[52] H. Jin, Q. Song, and X. Hu, "Auto-Keras: An efficient neural architecture search system," 2018, *arXiv:1806.10282*.
[53] H. Hu, J. Langford, R. Caruana, S. Mukherjee, E. Horvitz, and D. Dey, "Efficient forward architecture search," in *Proc. Adv. Neural Inf. Proc. Sys. (NIPS)*, 2019, pp. 1–10.
[54] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," 2019, *arXiv:1908.09791*.
[55] D. Zhou, X. Zhou, W. Zhang, C. C. Loy, S. Yi, X. Zhang, and W. Ouyang, "EcoNAS: Finding proxies for economical neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11396–11404.
[56] M. Zhang, H. Li, S. Pan, X. Chang, and S. Su, "Overcoming multi-model forgetting in one-shot NAS with diversity maximization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 7809–7818.
[57] Y. Zhao, L. Wang, Y. Tian, R. Fonseca, and T. Guo, "Few-shot neural architecture search," 2020, *arXiv:2006.06863*.
[58] B. T. Polyak and A. B. Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM J. Control Optim.*, vol. 30, no. 4, pp. 838–855, Jul. 1992.
[59] S. Gadat and F. Panloup, "Optimal non-asymptotic bound of the Ruppert-Polyak averaging without strong convexity," 2017, *arXiv:1709.03342*.
[60] X. Dong and Y. Yang, "NAS-Bench-201: Extending the scope of reproducible neural architecture search," 2020, *arXiv:2001.00326*.
[61] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[62] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[63] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.
[64] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[65] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong, "PC-DARTS: Partial channel connections for memory-efficient architecture search," 2019, *arXiv:1907.05737*.

CAT P. LE received the B.S. degree *(summa cum laude)* in electrical and computer engineering from Rutgers University, in 2016, and the M.S. degree in electrical engineering from the California Institute of Technology (Caltech), in 2017. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Duke University, under the supervision of Dr. Vahid Tarokh. His research interests include image processing, computer vision, and machine learning, with a focus on transfer learning, continual learning, and neural architecture search. His awards and honors include the Matthew Leydt Award and John B. Smith Award.

MOHAMMADREZA SOLTANI received the master's degrees in electrical engineering and telecommunication engineering with a minor in mathematics and the Ph.D. degree in electrical engineering from Iowa State University, in 2019. He is currently a Postdoctoral Associate with the Department of Electrical and Computer Engineering, Duke University. His recent projects include neural architecture search, radar signal processing using machine learning techniques, and meta-material design using deep learning. His research interests include the intersection of signal processing, machine learning, and numerical optimization.

JUNCHENG DONG is currently pursuing the master's degree in computer science with Duke University, under supervision of Prof. Vahid Tarokh. Before joining Duke University, he studied computer science and mathematics with the University of California San Diego (UCSD). Before UCSD, he graduated from the Nanyang Model High School, Shanghai, China. His research interests include machine learning, representation learning, and reinforcement learning. He received the Dean's Research Award from Duke University.

VAHID TAROKH (Fellow, IEEE) worked at AT&T Labs-Research, until 2000. From 2000 to 2002, he was an Associate Professor with the Massachusetts Institute of Technology (MIT). In 2002, he joined Harvard University as a Hammond Vinton Hayes Senior Fellow of electrical engineering and a Perkins Professor of applied mathematics. In January 2018, he joined as a Rhodes Family Professor of electrical and computer engineering, computer science, and mathematics, and a Bass Connections Endowed Professor with Duke University. He was also a Gordon Moore Distinguished Research Fellow at Caltech, in 2018. Since Jan 2019, he has been named as a Microsoft Data Science Investigator at Duke University.