# Task-aware Neural Architecture Search

Cat Le

Electrical and Computer Engineering
Duke University

March 02, 2020

# Introduction

- The distance between tasks is a form of ***non-commutative*** information, which represents the difficulty to apply the knowledge of one task to the other.
- The notion of distance between tasks is extremely important, and can be applied to a wide range of applications (i.e., transfer learning, neural architecture search, continual learning, meta learning, etc.)
- In this presentation, we demonstrate our distance measure and how to apply it to the neural architecture search.

# Distance Between Tasks

- There are several methods that attempt to evaluate the difference between task/data set.
- Some of these approaches are based on transfer learning, or some complexity measure of transferring knowledge of one task to another.
- We propose 2 approaches based on Fisher information :
  - Log-determinant of Fisher Information Matrices
  - Frechet distance of Fisher Information Matrices
- Our task dissimilarity measures are inherently asymmetric (non-commutative).

# Key Assumptions

- All datasets $X_i$, $i = 1, 2, ..., t$ are of the same size.
- Any task T and its dataset X can be represented by a sufficiently trained $\varepsilon$-approximation network.
- Let $\mathcal{P}_N(T, X)$ be a function that measures the performance of the architecture N on task T with data set X, with a range in $[0, 1]$.
- For a given $0 < \varepsilon < 1$, an architecture $N$ is an $\varepsilon$-approximation for $(T, X)$ if $\mathcal{P}_N(T, X) \geq 1 - \varepsilon$.
- In practice, we can use well-known hand-designed architectures as the $\varepsilon$-approximation networks.

# Fisher Information Matrix

- Fisher Information Matrix is related to the $2^{nd}$ order partial derivatives of the loss function with respect to the parameters.
- Let $X$ be the input data and $\theta = [\theta_1, \theta_2, ..., \theta_N]$ be the parameters of the network model. We seek to optimize the likelihood $p(X|\theta)$ with respect to $\theta$ by maximizing the log-likelihood $\log(p(X|\theta))$. We define the gradient of the log-likelihood as $s(\theta)$ :

$$s(\theta) = \nabla_\theta \log(p(X|\theta)). \tag{1}$$

# Fisher Information Matrix

- The gradient of the log-likelihood $s(\theta)$ describes the changes in likelihood with respect to each parameter $\theta_i$ in $\theta$.
- The Fisher Information Matrix captures the relation between $s(\theta_i)$ and $s(\theta_j)$ for $i, j \in N$. The Fisher Information Matrix is the covariance of the gradient of the log-likelihood $s(\theta)$ :

$$F_\theta = E_{p(x|\theta)}[(s(\theta) - E_{p(x|\theta)}[s(\theta)])(s(\theta) - E_{p(x|\theta)}[s(\theta)])^T]. \quad (2)$$

- When $E_{p(x|\theta)}[s(\theta)] = 0$, $I_\theta$ is simplified as :

$$F_\theta = E_{p(x|\theta)}[\nabla_\theta \log(p(x|\theta)) \nabla_\theta \log(p(x|\theta))^T]. \quad (3)$$

# Fisher Information Matrix

- In order to compute the Fisher Information Matrix, we need the log-likelihood with respect to $\theta$, which is often intractable.
- Let $X = [x_1, x_2, ..., x_M]$ be samples drawn from $p(x|\theta)$. The empirical Fisher Information is defined as :

$$F_\theta^* = \frac{1}{M} \sum_{i=1}^{M} \nabla_\theta \log(p(x_i|\theta)) \nabla_\theta \log(p(x_i|\theta))^T. \tag{4}$$

# Tasks and Their Fisher Information Matrices

- Let $B = \{(T_1, X_1), (T_2, X_2), \ldots, (T_K, X_K)\}$ denote a set of $K$ pairs of baseline "learned" tasks and their corresponding data sets.
- Suppose that $\varepsilon$-approximation networks $N_1, N_2, ..., N_K$ respectively correspond to $(T_i, X_i)$, $i = 1, 2, \cdots, K$.
  - $\varepsilon$ is chosen such that $\varepsilon = \min(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_K)$, where $\mathcal{P}_{N_i}(T_i, X_i) \geq 1 - \varepsilon_i$.
- Let $N_t$ be an $\varepsilon_t$-approximation network for the target task-data pair $(T_t, X_t)$ for some $0 < \varepsilon_t \leq \varepsilon$.
- Our goal is to evaluate how well the baseline approximation networks $N_i$, $i = 1, 2, ..., K$ perform on the target data set $X_t$.
- The target data set $X_t$ is used to compute the empirical Fisher Information Matrices for all the approximation networks.

# The Similarity Measure (Log-Determinant Distance)

- For some $b \in \{1, , 2, \ldots, K\}$, let $N_b$ denote the $\varepsilon$-approximation network corresponding to $b^{th}$ baseline task-data set pair.
- Let $F_{b,t}$ be the Fisher Information Matrix of $N_b$ with data $X_t$ from the task $t$.
- Let $F_{t,t}$ be the Fisher Information Matrix of $N_t$ with data $X_t$ from the task $t$.
- We define the *dissimilarity* from the task $b$ to the task $t$ as follows :

$$d[b,t] = \left| \frac{\log(\det(F_{b,t} + \sigma^2 * I_{n\times n}))}{n} - \frac{\log(\det(F_{t,t} + \sigma^2 * I_{m\times m}))}{m} \right|, \quad (5)$$

- Where $I$ is the identity matrix, $\sigma$ is a pre-selected small constant, $n$ and $m$ are the number of parameters in $N_b$ and $N_t$, respectively.

# The Similarity Measure (Log-Determinant Distance)

- If $m = n$, then the distance can be expressed as :

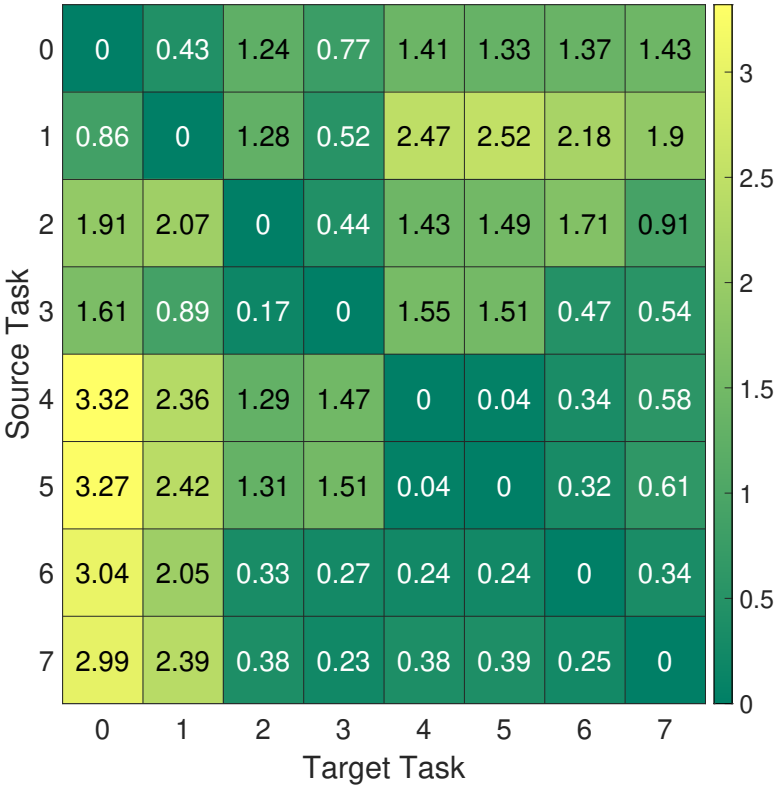
$$d[b,t] = \frac{1}{n}\left|\sum_i \log\left(\frac{\lambda^i_{b,t} + \sigma^2}{\lambda^i_{t,t} + \sigma^2}\right)\right|, \tag{6}$$

- Where $\lambda^i$ is the $\text{i}^{\text{th}}$ eigenvalue of the Fisher information matrix.
- This proposed dissimilarity is greater than or equal to 0, with the distance $d = 0$ indicating perfect similarity.

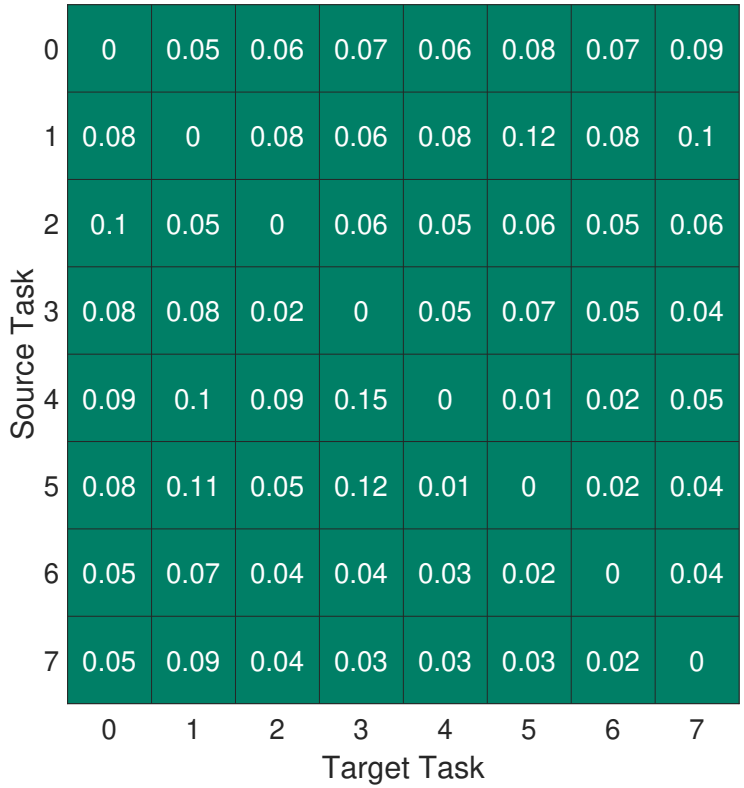
# Experiments with The Log-Determinant Distance

- Consider 8 tasks of classifications in MNIST and CIFAR-10.
  - Task 0 : binary classification of detecting digit 0
  - Task 1 : binary classification of detecting digit 6
  - Task 2 : binary classification of odd vs. even digits
  - Task 3 : 10-class classification in MNIST
  - Task 4 : binary classification of detecting (car, cat, ship)
  - Task 5 : binary classification of detecting (cat, ship, truck)
  - Task 6 : 4-class classification of bird, frog, horse, and anything else
  - Task 7 : 10-class classification in CIFAR-10
- Let VGG-16 be the $\varepsilon$-approximation for these tasks.
- To apply VGG-16 to MNIST tasks without architecture modification, we convert MNIST data to greyscale (i.e., 3 channels) and reshape images into $32 \times 32$.

# Experiments with The Log-Determinant Distance




(a) Mean




(b) Standard Deviation

FIGURE – Distances between classification tasks from MNIST and CIFAR10.

# Experiments with The Log-Determinant Distance

- The $i^{th}$ column of the mean table represents the average distance from other tasks to the target Task $i$ ($t = i$).
- Our results suggest that two tasks from the same data set (e.g., MNIST or CIFAR-10) are often more similar than tasks involving different data sets.
- It is perhaps surprising that the closest task to Task 3 is Task 7.
- Since other MNIST tasks are binary classification tasks, they do not intuitively appear as similar to the 10-class classification, even though Task 3 and Task 7 are using different data sets.

# Frechet Distance of Fisher Information Matrices

- Another approach to measure the distance between tasks is using the Frechet distance on Fisher Information Matrices.
- Let $F_{b,t}$ be the Fisher Information Matrix of $N_b$ with data $X_t$ from task $t$.
- Let $F_{t,t}$ be the Fisher Information Matrix of $N_t$ with data $X_t$ from task $t$.
- We define the distance from baseline task b to target task t as :

$$d_f[b,t] = \frac{1}{\sqrt{2}} tr(F_{b,t} + F_{t,t} - 2(F_{b,t}F_{t,t})^{0.5})^{0.5}. \quad (7)$$

# Frechet Distance of Fisher Information Matrices

- Assume that $F_{b,t}$, $F_{t,t}$ are approximated by only their diagonal entries.
- We normalize $F_{b,t}, F_{t,t}$ to have unit trace.
- The distance can be expressed by :

$$d_f[b,t] = \frac{1}{\sqrt{2}} \left\| F_{b,t}^{1/2} - F_{t,t}^{1/2} \right\|_F = \frac{1}{\sqrt{2}} \sqrt{\sum_i ((\lambda_{b,t}^i)^{0.5} - (\lambda_{t,t}^i)^{0.5})^2}. \quad (8)$$

- This task dissimilarity measure ranges in $[0, 1]$, with $d = 0$ denotes the perfect similarity and $d = 1$ means totally dissimilar tasks.

# Properties of Frechet Distance

## Remark (Remark 1)

*Let $X$ be the data of task $T$. For any Fisher Information matrix found by the $\varepsilon$-approximation network $N$ on data $X$ using gradient descent, with the same initialization settings and learning rate, then the Frechet distance between each pair of these Fisher information matrices are 0.*

## Proof

*Let $N_t, N_{t+1}$ be the network trained with data $X$ at time $t$ and $(t+1)$ using full gradient descent.*
*The weights of $N_t$ and $N_{t+1}$ are the same, because of the initial settings (e.g., seed, learning rate) are fixed.*
*Since Fisher Information matrix is the function of weights, networks with same weights will give the same Fisher matrices.*
*Therefore, the Frechet distance between these matrices is 0.*

# Properties of Frechet Distance

## Remark (Remark 2)

*Let $X$ be the input data from task $T$. For any Fisher Information matrix found by the $\varepsilon$-approximation network $N$ on data $X$ using stochastic gradient descent, with the same initialization settings, and the same order of data points, the Frechet distance between each pair of these Fisher matrices are 0.*

## Proof

*Let $N_t, N_{t+1}$ be the network trained with data $X$ at time $t$ and $(t+1)$ using stochastic gradient descent.*
*The weights of $N_t$ and $N_{t+1}$ are the same, because of the initial settings (e.g., seed, learning rate) and the order of data are fixed.*
*Since Fisher Information matrix is the function of weights, networks with same weights will give the same Fisher matrices.*
*Therefore, the Frechet distance between these matrices is 0.*

# Properties of Frechet Distance

**Theorem (Theorem 1)**

*Let $X$ be the input data from task $T$. Assume the objective loss function $L$ is strongly convex and its 3rd-order continuous derivative exists and bounded. Let the stochastic gradient function denotes as :*

$$g(\theta_t, \epsilon_t) = \nabla L(\theta_t) + \epsilon_t, \tag{9}$$

*where $\nabla L(\theta_t)$ is the true gradient, $\epsilon_t$ is some added noise, and $\mathbb{E}[\epsilon_t | \epsilon_0, \epsilon_1, ..., \epsilon_{t-1}] = 0$, and $S = \lim_{t \to \infty} \mathbb{E}[\epsilon_t {\epsilon_t}^T | \epsilon_0, \epsilon_1, ..., \epsilon_{t-1}]$ is finite.*
*For any Fisher information matrix found by the architecture network $N$ trained on dataset $X$ using stochastic gradient descent, the Frechet distance between each pair of these Fisher Information Matrices are close to 0.*

# Properties of Frechet Distance

**Proof (Proof of Theorem 1)**

*Let $N_1$ be the network trained on data $X$ with random seed $i$, characterized by $\theta_1$. Let $N_2$ be the network trained on data $X$ with random seed $j$, characterized by $\theta_2$. Since the objective function is strongly convex, both of these network will obtain the optimum solution $\theta^*$ after training a certain number of epochs with stochastic gradient descend.*
*[Polyak and Juditsky, 1992] states that for $t \to \infty$, then :*

$$\sqrt{t}(\bar{\theta}_t - \theta^*) \xrightarrow{\mathcal{D}} \mathcal{N}(0, \mathbf{H}(L(\theta^*))^{-1} S(\mathbf{H}(L(\theta^*))^{-1})^T), \tag{10}$$

*where $\mathbf{H}$ is Hessian matrix, $\bar{\theta}_t = \frac{1}{t}\Sigma_t \theta_t$.*

# Properties of Frechet Distance

## Proof (Proof of Theorem 1 (cont'))

*Applying the equation 10,* $\sqrt{t}(\bar{\theta}_{1t} - \theta^*)$ *and* $\sqrt{t}(\bar{\theta}_{2t} - \theta^*)$ *are asymptotically normal random variables :*

$$\sqrt{t}(\bar{\theta}_{1t} - \theta^*) \xrightarrow{\mathcal{D}} \mathcal{N}(0, \mathbf{H}(L(\theta^*))^{-1} S_1 (\mathbf{H}(L(\theta^*))^{-1})^T). \tag{11}$$

*and :*

$$\sqrt{t}(\bar{\theta}_{2t} - \theta^*) \xrightarrow{\mathcal{D}} \mathcal{N}(0, \mathbf{H}(L(\theta^*))^{-1} S_2 (\mathbf{H}(L(\theta^*))^{-1})^T). \tag{12}$$

*Given that the empirical Fisher Information matrix is defined as :*

$$F(\theta) = -\mathbb{E}[\mathbf{H}(L(\theta))] = -\lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} \mathbf{H}(L(\theta_i)),$$

*where N is the number of data points in the dataset. The Fisher Information* $F(\theta)$ *is a function of* $\theta$ *and it is continuous and differentiable for all* $\theta$*.*

# Properties of Frechet Distance

## Proof (Proof of Theorem 1 (cont'))

*Given that the Fisher Information matrix is positive definite,* $F(\theta)^{1/2}$ *is continuous and differentible. Applying the Delta method, we have :*

$$\sqrt{t}(\bar{F_1}_t^{1/2} - F^{*1/2}) \xrightarrow{\mathcal{D}} \mathcal{N}(0, \Sigma_1), \tag{13}$$

*where* $\Sigma_1 = \mathbf{J}_\theta(\mathbf{vec}(F(\theta^*)^{1/2}))\mathbf{H}(L(\theta^*))^{-1} S_1 (\mathbf{H}(L(\theta^*))^{-1})^T \mathbf{J}_\theta(\mathbf{vec}(F(\theta^*)^{1/2}))^T$
$\mathbf{vec}()$ *is the vectorization operator,* $\theta^*$ *is* $n \times 1$ *vector of the optimum parameters,* $F(\theta^*)$ *is* $n \times n$ *optimum Fisher matrix,* $\mathbf{J}_\theta(F(\theta^*))$ *is* $n^2 \times n$ *Jacobian matrix of the Fisher matrix.*
*As the result,* $\sqrt{t}(\bar{F_1}_t^{1/2} - \bar{F_2}_t^{1/2})$ *is asymptotically normal random variable :*

$$(\bar{F_1}_t^{1/2} - \bar{F_2}_t^{1/2}) \xrightarrow{\mathcal{D}} \mathcal{N}(0, \frac{1}{t}\Sigma_d). \tag{14}$$

$\Sigma_d = \mathbf{J}_\theta(\mathbf{vec}(F(\theta^*)^{1/2}))\mathbf{H}(L(\theta^*))^{-1}(S_1 + S_2)(\mathbf{H}(L(\theta^*))^{-1})^T \mathbf{J}_\theta(\mathbf{vec}(F(\theta^*)^{1/2}))^T.$

# Properties of Frechet Distance

## Proof (Proof of Theorem 1 (cont'))

*Given that the Frobenius norm of the $n \times n$ matrix is bounded by :*

$$\left\|\bar{F}_{1_t}^{1/2} - \bar{F}_{2_t}^{1/2}\right\|_\infty \leq \left\|\bar{F}_{1_t}^{1/2} - \bar{F}_{2_t}^{1/2}\right\|_F \leq n \left\|\bar{F}_{1_t}^{1/2} - \bar{F}_{2_t}^{1/2}\right\|_\infty . \tag{15}$$

*Since $(\bar{F}_{1_t}^{1/2} - \bar{F}_{2_t}^{1/2})$ is asymptotically normal random variable with zero mean and the covariance $\frac{1}{t}\Sigma_d \to 0$ as $t \to \infty$, all of its entries will be zero. As a result, $\left\|\bar{F}_{1_t}^{1/2} - \bar{F}_{2_t}^{1/2}\right\|_\infty \to 0$ as $t \to \infty$. As $t \to \infty$, the Frobenius norm of $(\bar{F}_{1_t}^{1/2} - \bar{F}_{2_t}^{1/2})$ is bounded as below :*

$$0 \leq \left\|\bar{F}_{1_t}^{1/2} - \bar{F}_{2_t}^{1/2}\right\|_F \leq n \times 0 = 0.$$

*Therefore, the Frechet distance $d = \frac{1}{\sqrt{2}} \left\|\bar{F}_{1_t}^{1/2} - \bar{F}_{2_t}^{1/2}\right\|_F \to 0$ as $t \to \infty$.*

# Properties of Frechet Distance

## Theorem (Theorem 2)

*Let $X_A$ be the input data for task $T_A$ with the objective functions $L_A$.*
*Let $X_B$ be the input data for task $T_B$ with the objective functions $L_B$.*
*Assume $X_A, X_B$ come from the same data distribution*
*Under the same assumptions Theorem 1, for any pair of Fisher information matrices $F_A, F_B$ found by the network $N$ trained on datasets $X_A, X_B$ with objective functions $L_A, L_B$, respectively, using stochastic gradient descent, the Frechet distance between $F_A, F_B$ is a Gaussian random variable whose the variance $\rightarrow 0$ as $t \rightarrow \infty$.*

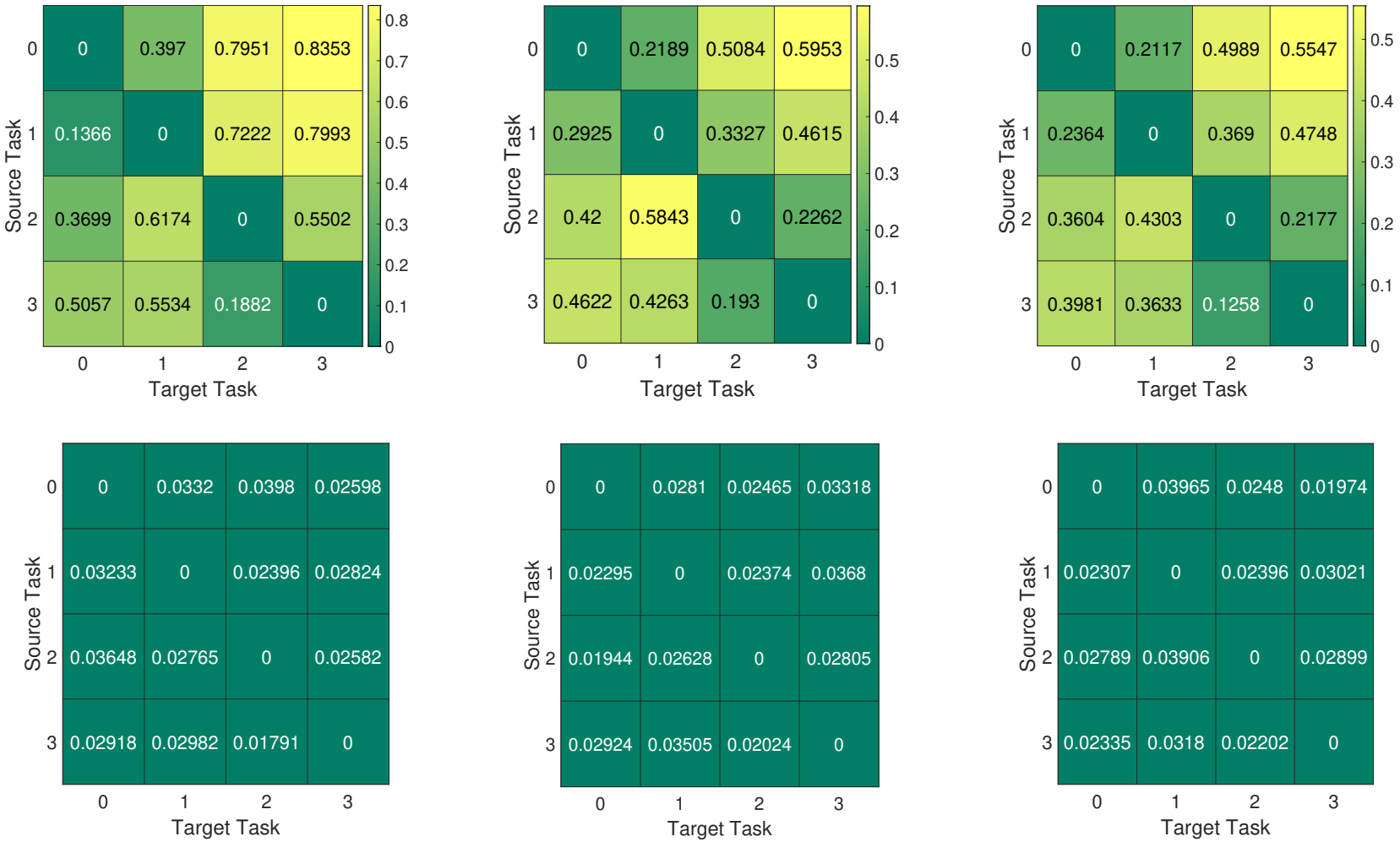
## Proof (Proof of Theorem 2)

*Similar to the proof of Theorem 1*

# Frechet Distance on MNIST

- Consider 4 tasks in MNIST dataset :
  - Task 0 : binary classification of detecting digit 0
  - Task 1 : binary classification of detecting digit 6
  - Task 2 : 5-class classification of digit 0, 1, 2, 3, and anything else
  - Task 3 : 10-class classification
- We compute the distance between these tasks using 3 different $\varepsilon$-approximation networks : VGG-16, ResNet-18, DenseNet-121.
- For each network, we repeat the training procedure 10 times, with different initial settings.

# Frechet Distance on MNIST



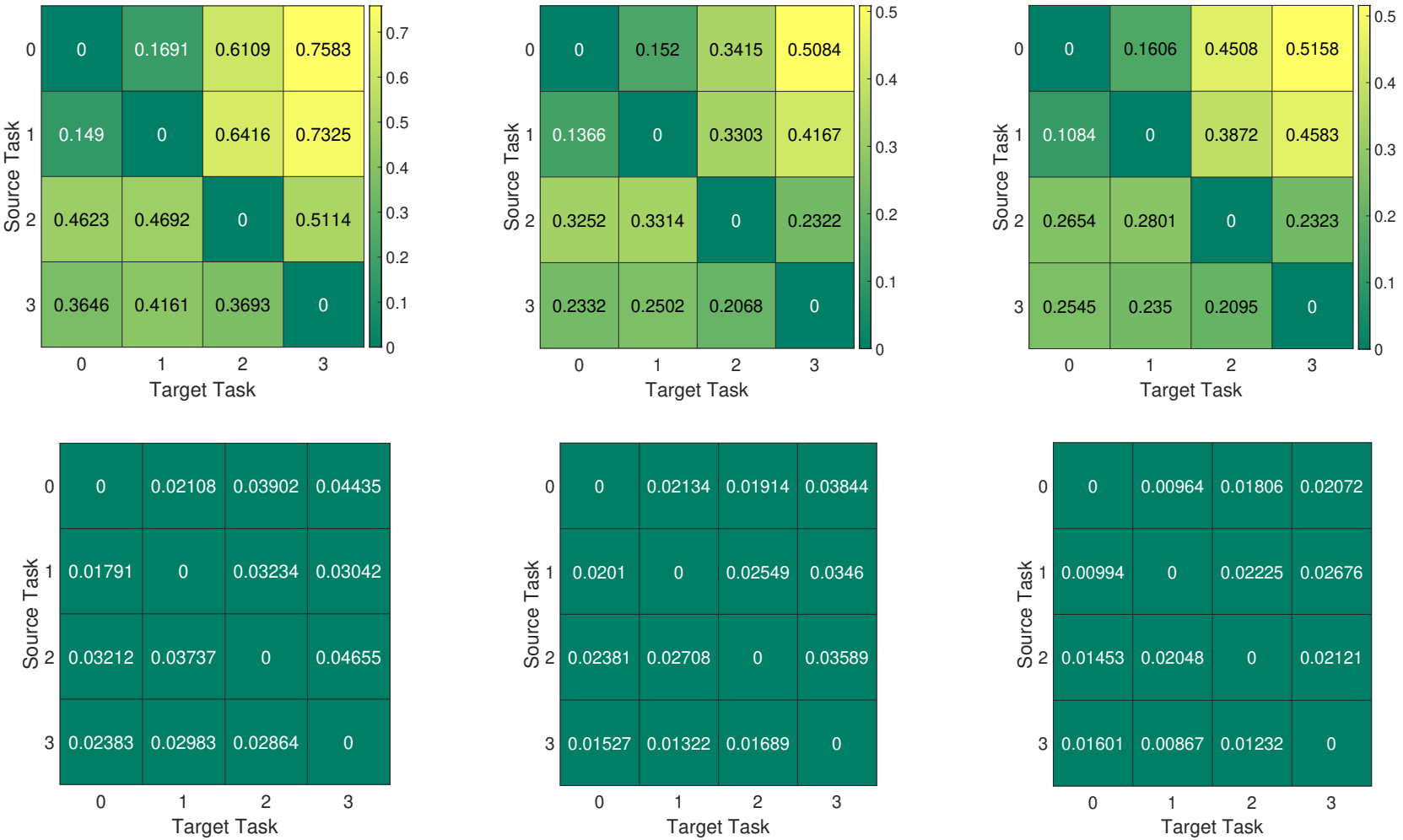

(a) VGG-16 (b) Resnet-18 (c) Densenet-21

FIGURE – Distances between classification tasks from MNIST.

# Frechet Distance on CIFAR-10

- Consider 4 tasks in CIFAR-10 dataset :
  - Task 0 : binary classification of detecting (car, cat, ship)
  - Task 1 : binary classification of detecting (cat, ship, truck)
  - Task 2 : 4-class classification of bird, frog, horse, and anything else
  - Task 3 : 10-class classification
- We compute the distance between these tasks using 3 different $\varepsilon$-approximation networks : VGG-16, ResNet-18, DenseNet-121.
- For each network, we repeat the training procedure 10 times, with different initial settings.

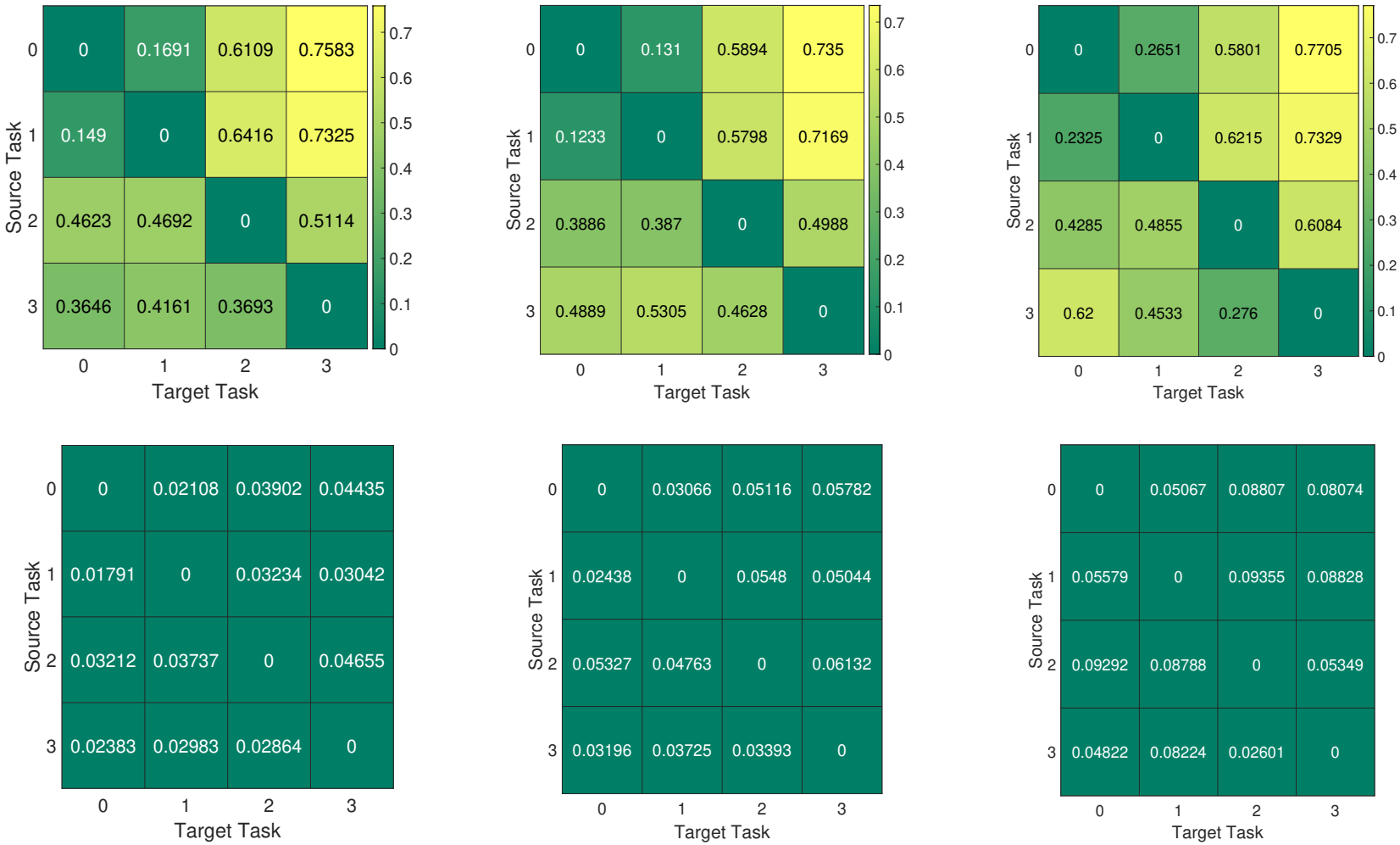
# Frechet Distance on CIFAR-10




(a) VGG-16 (b) Resnet-18 (c) Densenet-21

FIGURE – Distances between classification tasks from CIFAR-10.

# Frechet Distance on CIFAR-10



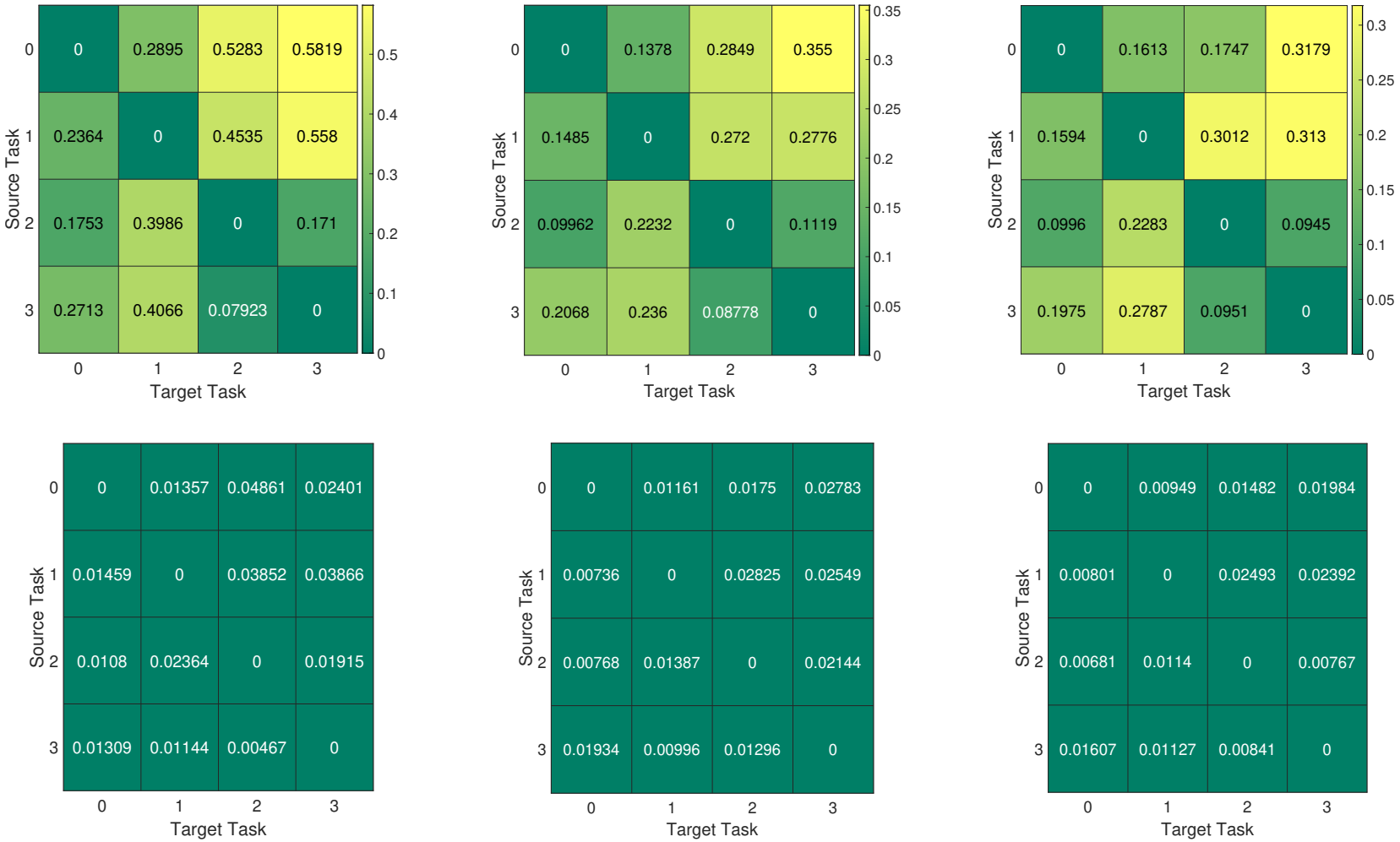

(a) data augmentation (b) no data augmentation (c) unbalanced data

FIGURE – Distances between classification tasks from CIFAR-10 using VGG-16.

# Frechet Distance on CIFAR-100

- In CIFAR-100, data is divided into 20 classes, where each class consists of 5 objects of similar type.
- Consider 4 tasks in CIFAR-100 dataset :
  - Task 0 : binary classification of detecting vehicles 1 & 2
  - Task 1 : binary classification of detecting household devices & furniture
  - Task 2 : 11-class classification of all vehicle and anything else
  - Task 3 : 21-class classification of all vehicle, household devices & furniture, and anything else.
- We compute the distance between these tasks using 3 different $\varepsilon$-approximation networks : VGG-16, ResNet-18, DenseNet-121.
- For each network, we repeat the training procedure 10 times, with different initial settings.

# Frechet Distance on CIFAR-100




(a) VGG-16 (b) Resnet-18 (c) Densenet-21

FIGURE – Distances between classification tasks from CIFAR-100.

# Neural Architecture Search (NAS)

- NAS is a framework to find a best performing architecture for a given task.
- It usually requires a lot of computational resources.
- It normally takes hundreds of GPU hours.
- Recent one-shot approach with a gradient-based search algorithm has reduced the search time to a few GPU days.
- However, all of these techniques heavily depend on the prior knowledge about the search space.
- They don't consider the relation between tasks in the search space.

# Task-aware Neural Architecture Search (TA-NAS)

- TA-NAS is a NAS framework that incorporate the measure of similarity between tasks into the search.
- Based on the assumption in transfer learning, similar tasks should have similar architecture.
- It uses the knowledge of learned tasks to help define the search space for the target task.
- The search for best performing architecture becomes more efficient in this restricted search space.

# Task-aware Neural Architecture Search (TA-NAS)

- TA-NAS works as follows :
  1. **Task Similarity** : Given $(T_t, X_t)$, TA-NAS finds the most related task-data set pairs in $B$.
  2. **Search Space** : It defines a search space as a combination of the related tasks.
  3. **Search Algorithm** : It searches for the best performing architecture.
- Currently, we only consider one related task to the target task.
- Multiple related tasks can be examined (**exploration - exploitation** problem)

# Definition of Search Space

- Defining a meaningful search space is the key to efficiently finding the best architecture for a specific task.
- In recent one-shot NAS techniques [Liu et al., 2018, Dong and Yang, 2020], the search space is defined by cells and skeletons.

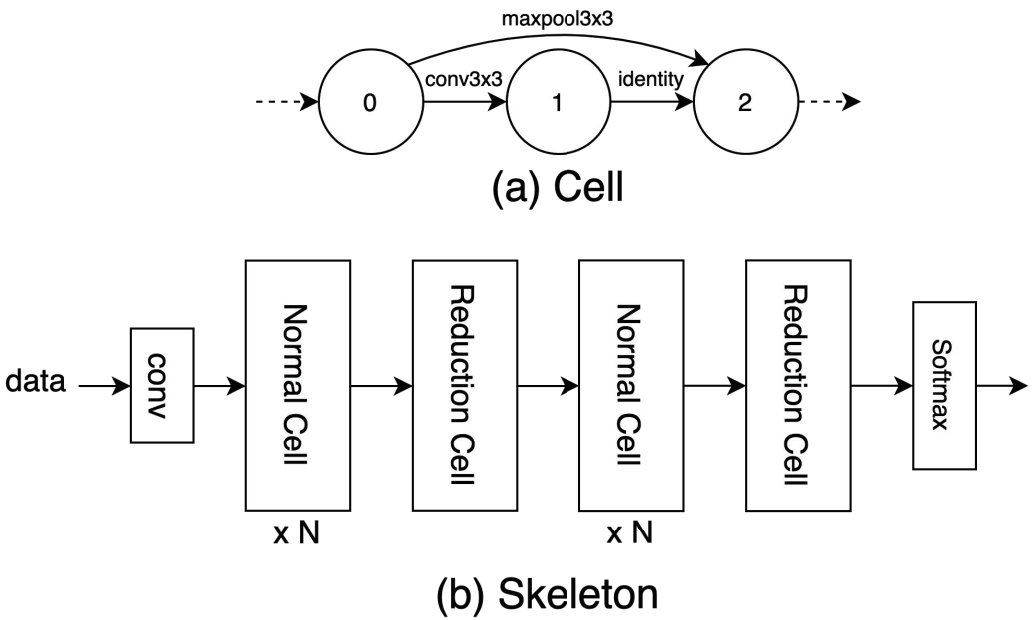



FIGURE – Example of cell and skeleton architectures.

# Cell

- A cell is a densely connected directed-acyclic graph (DAG) of nodes, where all nodes are connected by operations.
- Each node has 2 inputs and 1 output.
- The operations (e.g., identity, zero, convolution, pooling) are set so that the dimension of the output is the same as that of the input.
- If **n** is the number of nodes in a cell and **m** is the number of operations, the total number of possible cells is given by : $\mathbf{m} \times \exp\left(\frac{\mathbf{n}!}{2(\mathbf{n}-2)!}\right)$.

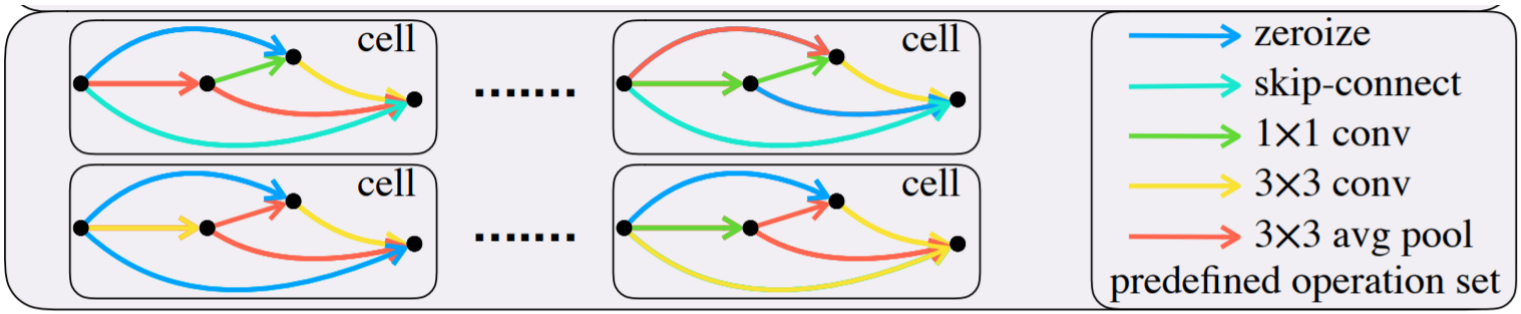



FIGURE – Example of cell architectures.

# Skeleton

- A skeleton is a combination of cells with other operations, forming the complete network architecture.
- A skeleton is normally predefined.
- The goal of NAS algorithm is to find the optimal cells.

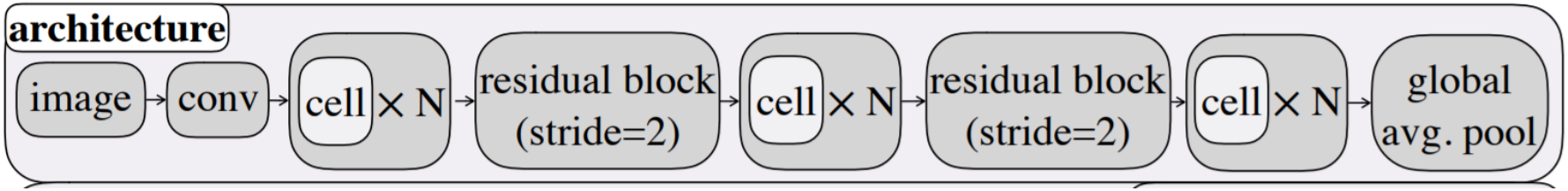



FIGURE – Example of skeleton architecture.

# Related Search Space

- The dissimilarity measures give us knowledge about how related two tasks are.
- Hence, we can define the search space of the target task by **combining** the cells, operations, and skeleton architectures from only the most similar task(s) in the baseline.
- Since the search space is restricted only to the space of the most related tasks, the search algorithm is efficient and requires a reduced number of GPU-hours.

# Fusion Search (FUSE)

- FUSE is our gradient-based search algorithm that evaluates all of the network candidates as a whole.
- It is based on the continuous relaxation of the outputs from all of the network candidates.
- Let $C$ be the set of candidate networks from the search space.
- Given $c \in C$ and data $X$, denote by $c(X)$ the output of the network $c$.
- The relaxed output $\bar{c}$ is the convex combination of the outputs from all candidates in $C$, where each weight in the combination given by exponential weights :

$$\bar{c}(X) = \sum_{c \in C} \frac{\exp(\alpha_c)}{\sum_{c' \in C} \exp(\alpha_{c'})} c(X), \tag{16}$$

- where $\alpha_c$ is a continuous variable assigned to network $c$'s output.

# FUSE (cont')

- Next, we conduct the evaluation by jointly training the network candidates and optimizing their $\alpha$ coefficients :
  - freeze $\alpha$ coefficients, jointly train network candidates and their weights.
  - freeze network candidates' weights, update $\alpha$ coefficients.
- While freezing $\alpha$, we update the weights $w$ in network candidates by jointly train the relaxed output $\bar{c}$ with cross-validation loss on training data :

$$\min_{w} \mathcal{L}_{train}(w; \alpha, \bar{c}, X_{train}), \tag{17}$$

- The weights $w$ in those candidates are fixed while we update the $\alpha$ coefficients with cross-validation loss validation data :

$$\min_{\alpha} \mathcal{L}_{val}(\alpha; w, \bar{c}, X_{val}). \tag{18}$$

- These steps are repeated for a number of iterations or until $\alpha$ coefficients converge. The best candidate in $C$ will be selected by :

$$c^* = \arg\max_{c \in C} \alpha_c. \tag{19}$$

# FUSE (cont')

**Algorithm 1:** FUSE Algorithm

**Initilization** : $c^*$, $\alpha$;
**Input** : search space S, $X_{train}$, $X_{val}$, I;
**Output** : Best architecture;
**while** *criteria not met* **do**
  Sample $C$ candidates $\in$ S;
  Relax the output of C using : $\bar{c}(X) = \sum_{c \in C} \frac{\exp(\alpha_c)}{\sum_{c' \in C} \exp(\alpha_{c'})} c(X)$;
  **while** $\alpha$ *not converge* **do**
    Update $C$ by descending $\nabla_w \mathcal{L}_{train}(w; \alpha, \bar{c})$;
    Update $\alpha$ by descending $\nabla_\alpha \mathcal{L}_{val}(\alpha; w, \bar{c})$;
  **end**
  **return** $c^* = \underset{c \in C}{\operatorname{argmin}} \alpha_c$;
**end**

# Search Result

- The target task : Task 3 (10-class classification in MNIST)
- The closest task : Task 7 (10-class classification in CIFAR-10)

TABLE – Comparison with state-of-art classifiers on Task 3 in MNIST.

| **Architecture** | **Accuracy (Task 3)** | **Parameter (M)** | **GPU days** |
|---|---|---|---|
| VGG-16 | 99.55 | 14.72 | - |
| ResNet-18 | 99.56 | 11.44 | - |
| DenseNet-121 | 99.61 | 6.95 | - |
| Random Search | 99.59 | 2.23 | 4 |
| FUSE w. related search space | 99.67 | 2.28 | 2 |

# Search Result (cont')

- The target task : Task 6 (4-class classification in CIFAR-10)
- The closest task : Task 7 (10-class classification in CIFAR-10)

TABLE – Comparison with state-of-art classifiers on Task 6 in CIFAR-10.

| **Architecture** | **Accuracy (Task 6)** | **Parameter (M)** | **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 | 4 |
| FUSE w. related search space | 90.87 | 3.02 | 2 |

# Search Result (cont')

- The target task : binary classification of indicating moon in 10-object sub-dataset from Quick, Draw !
- The closest task : (i) digit 0, (ii) trouser, (iii) digit 3 indicators from MNIST and fashion-MNIST.

TABLE – Comparison with state-of-art image classifiers on Quick, Draw ! dataset.

| **Architecture** | **Error (%)** | **Parameter (M)** | **GPU days** |
|---|---|---|---|
| ResNet-18 | 1.42 | 11.44 | - |
| ResNet-34 | 1.2 | 21.54 | - |
| DenseNet-161 | 1.17 | 27.6 | - |
| Random Search | 1.33 | 2.55 | 4 |
| FUSE w. standard search space | 1.21 | 2.89 | 2 |
| FUSE w. related search space | 1.18 | 2.72 | 2 |

# Conclusion

1. We define 2 methods to measure the similarity between tasks.
2. By definition, these measures are asymmetric and non-commutative.
3. Using these measures, a reduced search space of architectures for a target task can be constructed using the closest baseline tasks.
4. This reduces the complexity of architecture search, increases its efficiency.
5. Then, our FUSE algorithm quickly evaluates the candidates without fully train them.
6. The optimum architectures founded by our approach have superior performance with a smaller number of parameters.


Dong, X. and Yang, Y. (2020).
Nas-bench-102 : Extending the scope of reproducible neural architecture search.
*arXiv preprint arXiv :2001.00326*.


Liu, H., Simonyan, K., and Yang, Y. (2018).
Darts : Differentiable architecture search.
*Proc. Int. Conf. Machine Learning*.


Polyak, B. and Juditsky, A. (1992).
Acceleration of stochastic approximation by averaging.
*Siam Journal on Control and Optimization*, 30 :838–855.