HETEROGENEOUS MODEL TRANSFER BETWEEN DIFFERENT NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose an effective heterogeneous model transfer (HMT) method that can transfer the knowledge from one pretrained neural network to another neural network. Most of the existing deep learning methods depend much on a pretraining-finetuning strategy, i.e., pretraining a deep model on a large task-related (source) dataset and finetuning it on a small target dataset. Pretraining provides a universal feature representation for the target learning task and thus reduces the overfitting on a small target dataset. However, it is often assumed that the pretrained model and the target model share an identical backbone, which significantly limits the scalability of pretrained deep models. This paper relaxes this limitation and generalizes to heterogeneous model transfer between two different neural networks. Specifically, we select the longest chain from the source model and transfer it to the longest chain of the target model. Motivated by one-shot neural architecture search methods, the longest chain inherits merits from the source model and also serves as a weight-sharing path of the target model, thus provides a good initialization. With the longest chains, the layer-to-layer weight transfer is then transformed by bilinear interpolation and cyclic stack. HMT opens a new window for the pretraining-finetuning strategy and significantly improves the reuse efficiency of pretrained models without re-pretraining on the large source dataset. Experiments on several datasets show the effectiveness of HMT. Anonymous code is at `https://anonymous.4open.science/r/6ab184dc-3c64-4fdd-ba6d-1e5097623dfd/`.

Deep convolutional neural networks (LeCun et al., 1989; Krizhevsky et al., 2012; He et al., 2016) have achieved state-of-the-art performance in most of the machine learning tasks, e.g., recognition, detection, and segmentation. A deep model contains a large number of weights and thus requires a large target dataset to train. When it is unavailable to collect such a big target dataset, one can resort to the pretraining-finetuning strategy: pretraining on a large task-related (source) dataset (e.g., ImageNet (Deng et al., 2009)) as initialization and fine-tuning on a small target dataset.

In the past ten years, pretrained models on ImageNet are widely used for many downstream learning tasks to address the problem of lacking labeled data. For example, with the pretraining, a person re-identification model (Zhou et al., 2019) improves $\sim$3% rank-1 accuracy on the Market-1501 dataset (Zheng et al., 2015). The success of pretrained deep models is attributed to the hierarchical semantics and universal features. Low layers of neural networks share similar low-level features (e.g., color, texture), and high layers of neural networks contain high-level semantics (e.g., shape, class, and location) weighted by low-level features. Thanks to the hierarchical semantics, pretraining on a large source dataset provides a universal feature representation for the target learning task and thus improves the performance with less target data.

A notorious problem arises: pretraining a deep model on a large dataset (e.g., ImageNet) is expensive and time-consuming. To avoid the tedious pretraining procedure, researchers released ImageNet pretrained models of several classical neural networks (e.g., AlexNet (Krizhevsky et al., 2012), VGG (Simonyan & Zisserman, 2014), GoogLeNet (Szegedy et al., 2015), ResNet (He et al., 2016), DenseNet (Huang et al., 2017), and EfficientNet (Tan & Le, 2019)) as the backbones for downstream learning tasks. When a new learning task comes, we just select one of these classical neural networks as a backbone and slightly modify the last several layers to tailor for the new learning task.

However, this pretraining-finetuning pipeline is limited to homogeneous neural networks. The weight transfer requires that both the source model and the target model share an identical neural network backbone such that we can copy the weights from the source model to the target model (Fig-

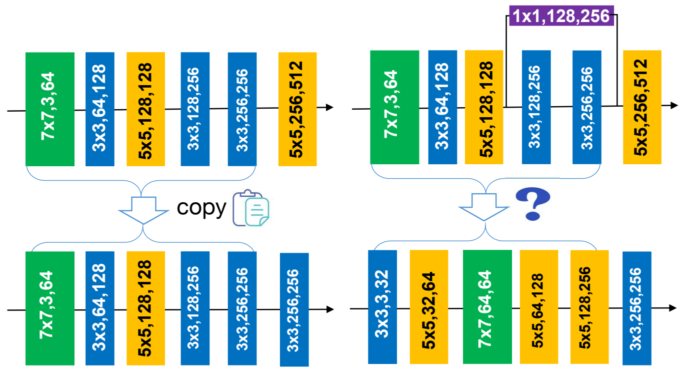

Figure 1: **(a)** Traditional homogeneous model transfer. It requires that source and target models share an identical backbone. For example, the first five layers' weights can be copied from the source model to the target model. **(b)** Heterogeneous model transfer. It focuses on transferring inconsistent weights from a source model to a target model (Best viewed in color).

ure 1 (a)). This greatly limits the scalability of the model transfer. Recent neural architecture search (NAS) methods (Guo et al., 2019; Li et al., 2020; Bender et al., 2018) show that different learning tasks have different optimal neural architectures. Researchers hope to search/design a custom neural architecture for a specific learning task instead of using common neural architectures. Because the custom neural architecture does not share an identical backbone with released pretrained models, we cannot simply copy weights from the released pretrained models (Figure 1 (b)).

In this paper, we propose a method of generalized model transfer that initializes a heterogeneous neural network by exploiting information from the released pretrained models. We refer to it as the *heterogeneous model transfer*. The heterogeneous model transfer is a challenging problem because of 1) inconsistent architectures (e.g., layer numbers and layer connections) between the source and target models; and 2) inconsistent filter sizes.

To solve these problems, we propose an effective heterogeneous model transfer (HMT) method. Overall, this paper makes three main contributions: **First**, we study the properties of the weight space of deep models, finding that 1) a chain is a basic coupled entity for model transfer; 2) filter interpolation nearly preserves the performance of neural networks; and 3) adding or removing a portion of input/output channels of filters nearly preserves the performance of a neural network because deep features are distributed. **Second**, based on these properties, we propose to perform heterogeneous model transfer between the two longest chains selected from the source and target model. We then perform filter interpolation to scale the kernel size of the source model's filters to achieve the consistent kernel size with the target model and propose a cyclic stack method to guarantee the consistent input/output channels. **Third**, with these transformations, the proposed method reduces the strong limitation of the existing pretraining-finetuning framework and significantly improves the reuse efficiency of pretrained models without re-pretraining on a large source dataset. Experiments on several datasets show that the heterogeneous model transfer gains significant improvement compared with the models without HMT.

1 Related Work

Traditional Homogeneous Model Transfer Pretraining-finetuning is a well-established paradigm that pretrains a model on a large dataset and finetunes the model on a small target dataset. Pretraining enables the model to obtain a universal feature representation and thus reduces the overfitting problem on a small dataset. Such a pipeline has been widely used in the deep learning community, including object detection (Girshick et al., 2014; Girshick, 2015; Ren et al., 2015), image segmentation (Long et al., 2015; He et al., 2017a), and person re-identification (Wang et al., 2020a; 2019; Liang et al., 2018). However, most of the existing methods only focus on homogeneous model transfer, where the source and target models share an identical backbone. When designing a new neural architecture, we need to re-pretrain it on a large-scale dataset, which is expensive and time-consuming. Unlike the homogeneous model transfer, this paper proposes a heterogeneous model transfer that can transfer the knowledge from the existing pretrained models to a new neural architecture.

Knowledge Distillation Knowledge distillation (Wang & Yoon, 2020; Hinton et al., 2015; Aguilar et al., 2020) is also a knowledge transfer framework that aims to transfer the knowledge from a trained teacher model to a student model. This is achieved by designing a loss to constrain the consistency between the output of the teacher model and the student model. Knowledge distillation requires re-training on a dataset to perform teacher-student learning. Different from this pipeline, the heterogeneous model transfer aims at the direct transformation from a trained model to a target model without re-pretraining on a source dataset.

Network to network and Network Morphism There are some works (Chen et al., 2015; Wei et al., 2016; Wang et al., 2017) focusing on growing a network from a trained seed network A_s to a new power network A_t. For example, Chen et al. (2015) proposed a $net2net$ method to accelerate the experimentation process by instantaneously transferring the knowledge from a previous network to each new deeper or wider network. Wei et al. (2016) extended the $net2net$ by expanding the changes of depth, width, kernel size, and subnet. However, these works significantly differ from our proposed method in several aspects. First, they limit the investigation for expanding networks, i.e., deepening networks, widening networks, expanding the kernel size, and copying sub-networks. However, in heterogeneous model transfer, the depth, width, kernel size of A_s could be larger than that of A_t. These methods focus on scaling up the networks and cannot extend to solve how to scale down the networks. Instead, our method has no limitation on this problem. Second, they are designed for the transfer between plain networks without skipping layers and hardly handle current complex state-of-the-art architectures with skipping layers, e.g., ResNet and NAS networks.

2 Heterogeneous Model Transfer

In this section, we introduce an effective method to transfer the weights from a source model to a heterogeneous target model without re-pretraining on the source dataset. In Section 2.1, we present the problem description for heterogeneous model transfer. Without any prior knowledge about the mathematical structure of the weights, we cannot solve this problem. Therefore, in Section 2.2, we investigate the mathematical structure of the weight space and study the properties of the weight space. Based on the properties, we propose a weight transformation to solve the heterogeneous model transfer problem in Section 2.3.

2.1 Problem description

Let $(\mathcal{A}_s, \mathbb{W}_s)$ and $(\mathcal{A}_t, \mathbb{W}_t)$ denote a source model and a target model, respectively. $\mathcal{A}_s$ and $\mathcal{A}_t$ are neural architectures, $\mathbb{W}_s$ and $\mathbb{W}_t$ are weights. Note that $\mathbb{W}_s$ has been pretrained on a large source dataset while $\mathbb{W}_t$ are initialized from scratch. Given a target neural architecture $\mathcal{A}_t$, the heterogeneous model transfer aims to directly transform $\mathbb{W}_s$ to $\mathbb{W}_{s*}$ such that the target model $(\mathcal{A}_t, \mathbb{W}_{s*})$ inherits the merits of the source model and has a good initialization for the target model. $\mathbb{W}_{s*}$ and $\mathbb{W}_t$ are of the same size. After transfer, the target model $(\mathcal{A}_t, \mathbb{W}_{s*})$ can be regarded as a pretrained model, which can be further finetuned on the target dataset. In this way, we do not need to re-pretrain the target model on the source dataset. Note that, when $\mathcal{A}_s$ and $\mathcal{A}_t$ share an identical backbone, the heterogeneous model transfer is degraded as the traditional homogeneous model transfer. We can directly copy $\mathbb{W}_s$ as $\mathbb{W}_{s*}$. Therefore, the heterogeneous model transfer can be regarded as a generalized version of the traditional homogeneous model transfer.

Intuitively, there are two challenges for heterogeneous model transfer due to inconsistent architectures and inconsistent filter sizes. First, the source and target architectures greatly differ in layer numbers and layer connections. They often have different branches, and it is hard to determine the layer-to-layer transfer relationship between the source and target models. Second, even though we know the layer-to-layer transfer relationship, they often have different filter sizes. It is unknown how to find an effective transformation between inconsistent filter sizes. Considering these two challenges, we ask:

- First, how do we find an available layer-to-layer transfer relationship between heterogeneous models?
- Second, how do we transfer a source layer to a target layer with different filter sizes given a layer-to-layer transfer relationship?

Driven by these two questions, we first study architecture-based and layer-based properties of neural networks' weight space and uncover the underlying mathematical structure in the weight space. Based on these properties, we give answers to these two questions in the next section.

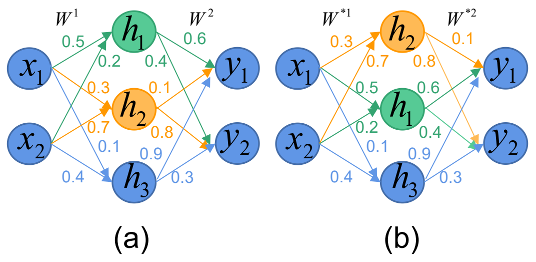

Figure 2: Illustration of the coupled relationship in a chain. **(a)** Before weight permutation. $\boldsymbol{W}^1 = [0.5, 0.3, 0.1; 0.2, 0.7, 0.4]$, $\boldsymbol{W}^2 = [0.6, 0.4; 0.1, 0.8; 0.9, 0.3]$. **(b)** After permutation. $\boldsymbol{W}^{*1} = [0.3, 0.5, 0.1; 0.7, 0.2, 0.4]$, $\boldsymbol{W}^{*2} = [0.1, 0.8; 0.6, 0.4; 0.9, 0.3]$. We observe that if we permute the columns of $\boldsymbol{W}^1$ and want to preserve the input-output mapping, we need to permute the corresponding rows of $\boldsymbol{W}_2$. (Best viewed in color).

2.2 Property of Weight Space of Neural Networks

2.2.1 Architecture-based property of the weight space of neural networks

Property 1 *A chain is a coupled entity in the weight space.*

We study the architecture-based property by considering the weight permutation relationship between different layers. A neural network is a directed acyclic graph. A chain is a sequence of connected layers, each of which has one input and one output (except the start layer and the end layer). A previous work (Wang et al., 2020b) has demonstrated that the weights of neural networks are constrained within a chain. Following this work, we consider a plain Multi-Layer Perceptron (MLP) neural network, whose weights are denoted as $(\boldsymbol{W}^l, \boldsymbol{W}^2, ...\boldsymbol{W}^l, ..., \boldsymbol{W}^L)$, where $\boldsymbol{W}^l$ is the weight of the l-layer. Because activation functions are element-based operations and have no impact on the permutation of neural networks, we ignore activation functions. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be the input and output, respectively. Let $\boldsymbol{Q}$ be a row permutation matrix. Therefore, $\boldsymbol{Q}^T\boldsymbol{Q}$ is an identity matrix. We have:

$$\begin{aligned} \boldsymbol{y} &= (\boldsymbol{W}^L)^T...(\boldsymbol{W}^{l+1})^T(\boldsymbol{W}^l)^T...(\boldsymbol{W}^1)^T\boldsymbol{x} \\ &= (\boldsymbol{W}^L)^T...(\boldsymbol{W}^{l+1})^T\boldsymbol{Q}^T\boldsymbol{Q}(\boldsymbol{W}^l)^T...(\boldsymbol{W}^1)^T\boldsymbol{x} \\ &= (\boldsymbol{W}^L)^T...(\boldsymbol{Q}\boldsymbol{W}^{l+1})^T(\boldsymbol{W}^l\boldsymbol{Q}^T)^T...(\boldsymbol{W}^1)^T\boldsymbol{x}. \end{aligned} \tag{1}$$

In Eq. (1), the chains $(\boldsymbol{W}^1, ..., \boldsymbol{W}^l, \boldsymbol{W}^{l+1}, ..., \boldsymbol{W}^L)$ and $(\boldsymbol{W}^1, ..., \boldsymbol{W}^l\boldsymbol{Q}^T, \boldsymbol{Q}\boldsymbol{W}^{l+1}, ..., \boldsymbol{W}^L)$ are equivalent because the input-output mapping is unchanged. It is observed that if we permute the columns of $\boldsymbol{W}_l$ and want to preserve the input-output mapping, we need to permute the corresponding rows of $\boldsymbol{W}_{l+1}$; if we permute the rows of $\boldsymbol{W}_l$ and want to preserve the input-output mapping, we need to permute the corresponding columns of $\boldsymbol{W}_{l-1}$. Therefore, the weights of a neural network are constrained in a chain form. A chain is a coupled entity that can describe the permutation relationship between layers. Figure 2 shows the illustration of the coupled relationship in a chain.

2.2.2 Layer-based property of the weight space of neural networks

Property 2 *Filter interpolation nearly preserves the performance of neural networks.*

We consider filter interpolation that scales the kernel size of a convolutional filter. Let K denote a filter and I denote a feature map; and $\mathsf{K}^{'}$ and $\mathsf{I}^{'}$ denote their scaled version. Let $\bigotimes$ denote a convolution operation. As is empirically investigated in the multi-scale training/inference or image pyramids (Najibi et al., 2019; Singh et al., 2018; Witkin, 1984; Lin et al., 2017), we know that feature map interpolation nearly preserves the performance of neural networks, which indicates that $\mathsf{K}^{'} \bigotimes \mathsf{I} \approx \mathsf{K}^{'} \bigotimes \mathsf{I}^{'}$. Next, we discuss $\mathsf{K} \bigotimes \mathsf{I} \approx \mathsf{K}^{'} \bigotimes \mathsf{I}^{'}$. For notation simplification, we use 1-D convolution as illustration. Ideally and mathematically, a 1-D convolution is defined by:

$$K(x) \bigotimes I(x) = \int_{-\infty}^{\infty} K(\tau)I(x-\tau)d\tau, \tag{2}$$

where $I(x)$ denote a 1-D continuous function and $K(x)$ denote a 1-D continuous filter function. In actual engineering implementation, a convolution (i.e., Eq. (2)) is always implemented using discretization, i.e., sampling is required. $\mathsf{K} \bigotimes \mathsf{I}$ and $\mathsf{K}^{'} \bigotimes \mathsf{I}^{'}$ can be regarded as two ways of sampling using different sampling rates. Both of them are approximations of $\int_{-\infty}^{\infty} K(\tau)I(x-\tau)d\tau$. Therefore, we have $\mathbf{K}\bigotimes\mathbf{I} \approx \mathbf{K}^{'}\bigotimes\mathbf{I}^{'}$. Finally, by combining $\mathbf{K}\bigotimes\mathbf{I} \approx \mathbf{K}^{'}\bigotimes\mathbf{I}^{'}$ and $\mathbf{K}^{'}\bigotimes\mathbf{I} \approx \mathbf{K}^{'}\bigotimes\mathbf{I}^{'}$, we have $\mathbf{K}\bigotimes\mathbf{I} \approx \mathbf{K}^{'}\bigotimes\mathbf{I}$. This indicates that the filter interpolation nearly preserves the performance of neural networks. Figure 3 presents the details.

Property 3 *Adding or removing a portion of the rows/columns of the weight of neural layers nearly preserves the performance of a neural network.*

Let $\mathbf{W}^l \in \mathbb{R}^{n^l\times c^l\times h^l\times w^l}$ be a 4-D tensor. n^l, c^l, h^l, and w^l denote output channels, input channels, kernel height, and kernel width at the l-th layer, respectively. We consider the output channels and input channels, i.e., n^l and c^l. In a convolutional layer, one feature space is mapped into another feature space by a filter. A deep model produces distributed channel features, each of which is related to global features. Therefore, a deep model still works even though a portion of neurons are deactivated. This implies that removing/repeating a portion of rows or columns of weights in neural layers does not significantly harm the performance of a neural network. Because distributed features have been fully investigated in previous works (Hinton, 1984; Bengio et al., 2013), we do not focus on describing the distributed features of a neural network in this paper. Note that, removing a portion of the rows/columns of the weight of a layer is also investigated in neural network pruning methods (He et al., 2017b).

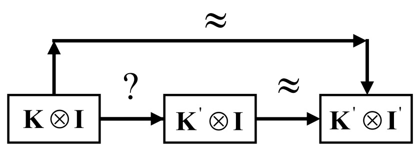

Figure 3: Illustration of filter interpolation. Given a filter $\mathbf{K}$ and a feature map $\mathbf{I}$, we scale both of them using bilinear interpolation, and obtains $\mathbf{K}^{'}$ and $\mathbf{I}^{'}$. Because $\mathbf{K}\bigotimes\mathbf{I} \approx \mathbf{K}^{'}\bigotimes\mathbf{I}^{'}$ and $\mathbf{K}^{'}\bigotimes\mathbf{I} \approx \mathbf{K}^{'}\bigotimes\mathbf{I}^{'}$, we have $\mathbf{K}\bigotimes\mathbf{I} \approx \mathbf{K}^{'}\bigotimes\mathbf{I}$.

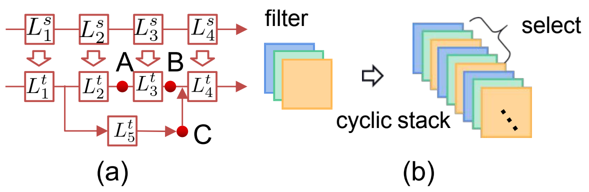

Figure 4: (a) Longest chain. In this example, $L_1^t \rightarrow L_2^t \rightarrow L_3^t \rightarrow L_4^t$ is the longest chain. (b) Cyclic stack. A 3-channel source filter is recursively stacked, which is used to select the first five channels for a 5-channel target filter (Best viewed in color).

2.3 An efficient heterogeneous model transfer method

Let $\mathbb{W}_s = \{\mathbf{W}_s^l\}_{l=1}^{L_s}$ be a sequence of the weights of the neural layers, where L_s is the total number of layers in $\mathbb{W}_s$. $\mathbf{W}_s^l$ is the weight of the l-th layer of $\mathbb{W}_s$. In a similar way, we have $\mathbb{W}_t = \{\mathbf{W}_t^l\}_{l=1}^{L_t}$. Let $\mathbf{W}_s^l \in \mathbb{R}^{n_s^l\times c_s^l\times h_s^l\times w_s^l}$ and $\mathbf{W}_t^l \in \mathbb{R}^{n_t^l\times c_t^l\times h_t^l\times w_t^l}$ be two 4-D tensors, where n^l, c^l, h^l, and w^l denote output channels, input channels, kernel height, and kernel width at the l-th layer of a model, respectively. Their subscripts s and t denote the source or target models, respectively. The goal of the heterogeneous model transfer is to transfer the weights $\mathbb{W}_s = \{\mathbf{W}_s^l\}_{l=1}^{L_s}$ to $\mathbb{W}_t = \{\mathbf{W}_t^l\}_{l=1}^{L_t}$.

2.3.1 A layer-to-layer transfer relationship between heterogeneous models

Longest chain principle. As demonstrated in the first property, we attempt to perform the heterogeneous model transfer based on chains to preserve the coupled permutation relationship. As discussed in Section 2.1, a neural network often contains lots of branches, which makes it challenging to find an available layer-to-layer transfer relationship for heterogeneous model transfer.

Recently, one-shot based neural architecture search (NAS) methods (Bender et al., 2018; Guo et al., 2019; You et al., 2020; Li et al., 2020) achieve promising performance. They train a weight-sharing super-network by sampling a sub-network each time. During the evaluation, they search the optimal sub-network as the final network. Due to the weight-sharing training, sub-networks inherit the merits of super-networks. Motivated by the one-shot NAS methods, we develop an effective heterogeneous model transfer method. Given a pretrained source networks $\mathcal{A}_s$, we select a chain $\mathcal{C}_s$ (sub-network) from $\mathcal{A}_s$. Because $\mathcal{C}_s$ is one path of $\mathcal{A}_s$, one-shot NAS methods demonstrate that $\mathcal{C}_s$ inherits the merits of $\mathcal{A}_s$ and thus provides a good initialization for $\mathcal{A}_t$. We then transfer $\mathcal{C}_s$ to $\mathcal{A}_t$.

Specifically, we propose a *longest chain* principle where a chain of $\mathcal{A}_s$ that contains the most layers should be selected as the candidate transfer chain. The more layers we transfer, the fewer layers we need to train from scratch. For the other layers, we directly discard them due to the collision permutation of chains.

As illustrated in Figure 4 (a), there are two chains in the target model, i.e., $L_1^t \to L_2^t \to L_3^t \to L_4^t$ and $L_1^t \to L_5^t \to L_4^t$. As suggested in the *longest chain* principle, $L_1^t \to L_2^t \to L_3^t \to L_4^t$ is selected as a candidate transferred chain. We transfer $L_1^s \to L_2^s \to L_3^s \to L_4^s$ to $L_1^t \to L_2^t \to L_3^t \to L_4^t$ layer by layer. As for L_5, we do not perform any transfer due to a collision permutation problem. Suppose we copy the weight of L_2^s to both L_2^t and L_5^t, the output A and C have the same permutation. However, the columns of L_3^t can be randomly permuted. Therefore, combining the features maps B and C will mix up the permutation of the feature maps, leading to the collision permutation problem.

Remark. Note that, in some cases, the longest chain is not unique, and we can just select one of them. In other cases, there is no collision permutation between different chains, and it is appropriate to simultaneously transfer a source layer to several target layers. However, we do not focus on these special cases in this paper.

2.3.2 Layer-to-layer transfer

After we find a layer-to-layer transfer relationship between heterogeneous models, the heterogeneous model transfer is degraded as a layer-to-layer transfer problem between two chains. Let $\mathcal{C}_s = (\tilde{\mathbf{W}}_s^1, \tilde{\mathbf{W}}_s^2, ..., \tilde{\mathbf{W}}_s^m)$ and $\mathcal{C}_t = (\tilde{\mathbf{W}}_t^1, \tilde{\mathbf{W}}_t^2, ..., \tilde{\mathbf{W}}_t^n)$ be the longest chains of $\mathcal{A}_s$ and $\mathcal{A}_t$, respectively. m and n denote the length of $\mathcal{C}_s$ and $\mathcal{C}_t$, respectively. We perform layer-to-layer transfer, i.e., from $\tilde{\mathbf{W}}_s^l \in \mathbb{R}^{n_s^l \times c_s^l \times h_s^l \times w_s^l}$ to $\tilde{\mathbf{W}}_t^l \in \mathbb{R}^{n_t^l \times c_t^l \times h_t^l \times w_t^l}$ $(l \leq min(m,n))$. The key problem is to find a transformation from $\tilde{\mathbf{W}}_s^l$ to $\tilde{\mathbf{W}}^{*l}_s$ such that $\tilde{\mathbf{W}}^{*l}_s$ and $\tilde{\mathbf{W}}_t^l$ are of the same tensor size and $\tilde{\mathbf{W}}^{*l}_s$ inherits universal features from $\tilde{\mathbf{W}}_s^l$ as much as possible.

As demonstrated in the second property, filter interpolation preserves the performance of neural networks. Feature map interpolation is widely investigated in feature map pyramid-based neural networks and often has little impact on neural networks' performance. Therefore, to obtain the consistent kernel height and kernel width, we can use the filter interpolation to transform $\tilde{\mathbf{W}}_s^l$ to $\tilde{\mathbf{W}}'^l_s$, where $\tilde{\mathbf{W}}'^l_s \in \mathbb{R}^{n_s^l \times c_s^l \times h_t^l \times w_t^l}$.

As demonstrated in the third property, adding or removing a portion of the rows/columns of the weight of neural layers nearly preserves the performance of a neural network. To obtain the consistent input/output channels, we propose a cyclic stack method to obtain the consistent filter size, i.e., transform $\tilde{\mathbf{W}}'^l_s$ to $\tilde{\mathbf{W}}^{*l}_s$ by the cyclic stack, where $\tilde{\mathbf{W}}'^l_s \in \mathbb{R}^{n_s^l \times c_s^l \times h_t^l \times w_t^l}$ and $\tilde{\mathbf{W}}^{*l}_s \in \mathbb{R}^{n_t^l \times c_t^l \times h_t^l \times w_t^l}$. Specifically, for the input channel size, we recursively stack the source filters and copy the channels of filters from a source filter to a target filter one after one. As illustrated in Figure 4 (b), a 3-channel source filter is recursively stacked. The first five channels are selected for a 5-channel target filter. The cyclic stack of the output channel is similar.

3 Experiments

In this section, we analyze the effectiveness of our heterogeneous model transfer (HMT) method. The HMT model mainly includes chain-based transfer, filter interpolation, and cyclic stack. Therefore, we present ablation studies to reveal the effect of each key component of HMT. The detailed training implementation is shown in Appendix A.

Datasets The Market-1501 dataset (Zheng et al., 2015) has six cameras. It contains 32,668 annotated bounding boxes of 1,501 identities. Among them, 12,936 images from 751 identities are used for training, and 19,732 images from 750 identities plus distractors are used for the gallery. As for query, 3,368 hand-drawn bounding boxes from 750 identities are adopted.

The DukeMTMC-reID dataset (Zheng et al., 2017) has 8 cameras. There are 1,404 identities appearing in more than two cameras and 408 identities in only one camera. Specifically, 702 IDs are selected as the training set, and the remaining 702 IDs are used as the testing set. In the testing set, one query image is picked for each ID in each camera, and the remaining images are put in the gallery. In this way, there are 16,522 training images of 702 identities, 2,228 query images of the other 702 identities, and 17,661 gallery images.

The CIFAR-100 dataset (Krizhevsky & Hinton, 2009), 32 × 32 in size, has 100 classes. Each class contains 600 images, including 500 training images and 100 testing images. In this paper, we only use 1,000 samples for training to form a small-scale target dataset.

Evaluation Metrics For person re-identification, we adopt the standard Cumulative Match Characteristic (CMC) and mean Average Precision (mAP) as evaluation metrics. For classification, we use top-1, top-5, and top-10 accuracy as evaluation metrics.

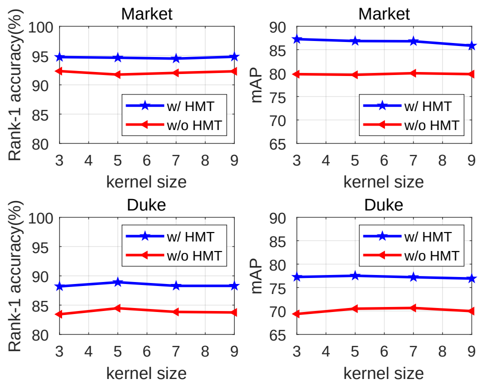

Figure 5: Effectiveness of the filter interpolation.

3.1 Effectiveness of the filter interpolation

To show the effectiveness of the filter interpolation, we modify the kernel size of five convolutional layers of ResNet-50, i.e., the 1-st, 2-nd, 11-th, 29-th, and 41-st layers. The 3 × 3 kernels of ResNet-50 are replaced by 5 × 5, 7 × 7, and 9 × 9, respectively. We apply the modified network to the person re-identification task. We use a state-of-the-art multi-pooling method [1] as my framework. Because the modified networks are different from the vanilla ResNet-50, the existing methods cannot handle this kind of model transfer. We directly train a model from scratch as a baseline. As shown in Figure 5, with the filter interpolation, the proposed method (w/ HMT) significantly improves the performance against the baseline (w/o HMT).

3.2 Effectiveness of the cyclic stack

To show the effectiveness of the cyclic stack, we modify the input/output channel size of eight convolutional layers of ResNet-50, i.e., the 2-nd, 3-rd, 11-th, 12-th, 29-th, 30-th, 40-th, and 41-st layers. For the channel size of these layers, we use ×0.5, ×1.0, and ×1.5 channels, respectively. Similar to Section 3.1, we directly train a model on the target dataset from scratch as a baseline. As shown in Figure 6, the proposed method with the cyclic stack (w/ HMT) achieves significant improvement compared with the baseline (w/o HMT).

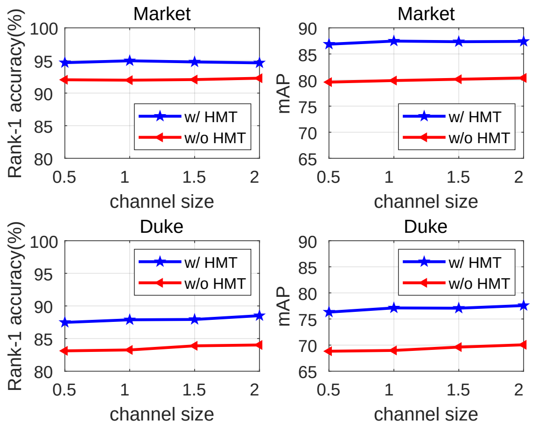

Figure 6: Effectiveness of the cyclic stack.

3.3 Effectiveness of both the filter interpolation and cyclic stack

To show the effectiveness of both the filter interpolation and cyclic stack, we simultaneously modify the channel size and the kernel size of the filters of the vanilla ResNet-50. For the kernel size of each layer, we replace the 3 × 3 kernels of ResNet-50 by randomly sampling 3 × 3, 5 × 5, or 7 × 7 kernels. For the channel size of each layer, we randomly select ×0.5, ×1.0, or ×1.5 channels. The results are presented in Figure. 7. We find that even though we modify most of convolutional filters of

[1]https://github.com/douzi0248/Re-ID

Table 1: Effectiveness of the chain-based transfer.

	Baseline (w/o HMT)	w/ shuffled HMT	w/ interval HMT	w/ chain HMT
top-1	17.08	15.70	17.69	**19.67**
top-5	38.80	36.64	39.33	**43.24**
top-10	51.81	49.41	52.40	**56.20**

Table 2: Transfer between different neural architectures.

	VGG-11$\rightarrow$PlainNet-5		VGG-11$\rightarrow$ResNet-8		ResNet-18$\rightarrow$PlainNet-5		ResNet-18$\rightarrow$ResNet-8	
	w/o HMT	w/ HMT	w/o HMT	w/ HMT	w/o HMT	w/ HMT	w/o HMT	w/ HMT
top-1	17.08	**19.67**	14.55	**18.63**	17.08	**17.35**	14.55	**15.89**
top-5	38.80	**43.24**	34.03	**40.60**	38.80	**39.67**	34.03	**36.73**
top-10	51.87	**56.20**	46.61	**53.66**	51.87	**52.39**	46.61	**49.65**

ResNet-50, the proposed method with the filter interpolation and cyclic stack still obtains promising improvement compared with the baseline model without HMT.

3.4 Effectiveness of the chain-based transfer

To show the effectiveness of the chain-based transfer, we implement three methods for comparison. First, we implement a baseline that trains a model from scratch, i.e., Baseline w/o HMT. Second, we implement a method that shuffles the layers within a chain and then transfer them to the target model (shuffled HMT, e.g., $L_4^s \rightarrow L_2^s \rightarrow L_1^s \rightarrow L_3^s$ to $L_1^t \rightarrow L_2^t \rightarrow L_3^t \rightarrow L_4^t$). Third, we implement another method that performs the weight transfer every two layers (interval HMT, e.g., $L_1^s \rightarrow L_3^s \rightarrow L_5^s \rightarrow L_7^s$ to $L_1^t \rightarrow L_2^t \rightarrow L_3^t \rightarrow L_4^t$). As shown in Table 1, we can see that the chain-based transfer (chain HMT) achieves better performance than the other three methods. This implies that non-chain based transfer methods destroy the mathematical structure of the weight of neural networks and thus cannot help the training procedure on the target dataset.

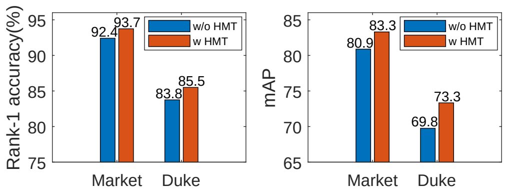

Figure 7: Effectiveness of both the filter interpolation and cyclic stack.

3.5 Transfer between different neural architectures

We further widen the architecture gap between the source and target neural networks, which differ in layer connections, architecture depth, input/output channels, and kernel size. To this, we design two networks (i.e., ResNet-8 and PlainNet-5) as described in Appendix B and Appendix C. We transfer the existing pretrained networks to PlainNet-5 and ResNet-8. The pretrained networks include VGG-11 and ResNet-18. We compare the proposed HMT method with baseline models without HMT, and the results show the consistent improvement of our proposed method, as shown in Table 2.

4 Conclusion

In this paper, we propose an effective heterogeneous model transfer method that can transfer the knowledge from a pretrained neural network to a new neural architecture. We first study the architecture-based properties and layer-based properties of neural networks. Based on these properties, we propose to select the longest chains from the source and target models and perform filter interpolation and cyclic stack to obtain the consistent filter size. Experiments on several datasets show that the heterogeneous model transfer gains significant improvement compared with baselines.

Heterogeneous model transfer relaxes the limitation of the identical backbone assumption in the traditional homogeneous model transfer, which opens a new window for the pretraining-finetuning strategy. The heterogeneous model transfer does not need to re-pretrain a newly designed neural architecture on a large source dataset and thus significantly improves the reuse efficiency of pretrained models. We believe that this is an important direction in the deep learning community.

References

Gustavo Aguilar, Yuan Ling, Yu Zhang, Benjamin Yao, Xing Fan, and Chenlei Guo. Knowledge distillation from internal representations. In *AAAI*, pp. 7350–7357, 2020.

Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pp. 550–559, 2018.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. *arXiv preprint arXiv:1904.00420*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017a.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389–1397, 2017b.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Geoffrey E Hinton. Distributed representations. 1984.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Xiang Li, Chen Lin, Chuming Li, Ming Sun, Wei Wu, Junjie Yan, and Wanli Ouyang. Improving one-shot nas by suppressing the posterior fading. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13836–13845, 2020.

Wenqi Liang, Guangcong Wang, Jianhuang Lai, and Junyong Zhu. M2m-gan: Many-to-many generative adversarial transfer learning for person re-identification. *arXiv preprint arXiv:1811.03768*, 2018.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

Mahyar Najibi, Bharat Singh, and Larry S Davis. Autofocus: Efficient multi-scale inference. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9745–9755, 2019.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pp. 91–99, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Bharat Singh, Mahyar Najibi, and Larry S Davis. Sniper: Efficient multi-scale training. In *Advances in neural information processing systems*, pp. 9310–9320, 2018.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

Guangcong Wang, Xiaohua Xie, Jianhuang Lai, and Jiaxuan Zhuo. Deep growing learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2812–2820, 2017.

Guangcong Wang, Jian-Huang Lai, Wenqi Liang, and Guangrun Wang. Smoothing adversarial domain attack and p-memory reconsolidation for cross-domain person re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10568–10577, 2020a.

Guangcong Wang, Jianhuang Lai, Guangrun Wang, and Wenqi Liang. Function feature learning of neural networks, 2020b. URL `https://openreview.net/forum?id=rJgCOySYwH`.

Guangrun Wang, Guangcong Wang, Xujie Zhang, Jianhuang Lai, and Liang Lin. Weakly supervised person re-id: Differentiable graphical learning and a new benchmark. *arXiv preprint arXiv:1904.03845*, 2019.

Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *arXiv preprint arXiv:2004.05937*, 2020.

Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen. Network morphism. In *International Conference on Machine Learning*, pp. 564–572, 2016.

Andrew Witkin. Scale-space filtering: A new approach to multi-scale description. In *ICASSP'84. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 9, pp. 150–153. IEEE, 1984.

Shan You, Tao Huang, Mingmin Yang, Fei Wang, Chen Qian, and Changshui Zhang. Greedynas: Towards fast one-shot nas with greedy supernet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1999–2008, 2020.

L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian. Scalable person re-identification: A benchmark. In *ICCV*, pp. 1116–1124, 2015.

Z. Zheng, L. Zheng, and Y. Yang. Unlabeled samples generated by gan improve the person re-identification baseline in vitro. In *ICCV*, 2017.

Kaiyang Zhou, Yongxin Yang, Andrea Cavallaro, and Tao Xiang. Omni-scale feature learning for person re-identification. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3702–3712, 2019.

A Appendix: Implementation

On the Market-1501 and DukeMTMC-reID datasets, we train the multi-pooling model for 220 epochs. We use the SGD optimizer. The learning rate is set to 0.03 and is divided by 10 after 60 and 130 epochs. We use both the triplet loss and the cross-entropy loss. The detailed implementation is provided in the reference link.

On the CIFAR-100 dataset, we train the model for 350 epochs using the SGD optimizer as the standard implementation. The learning rate is set to 0.03 and is divided by 10 after 150 and 250 epochs. We use the cross-entropy loss for classification.

B Appendix: The architecture of ResNet-8

ResNet-8 contains one convolution layer, three building blocks, one global pooling, and one linear layer. Each building block contains two convolutional layers with a skipping layer. Here, each convolution layer is followed by a ReLU layer and a BatchNorm layer.

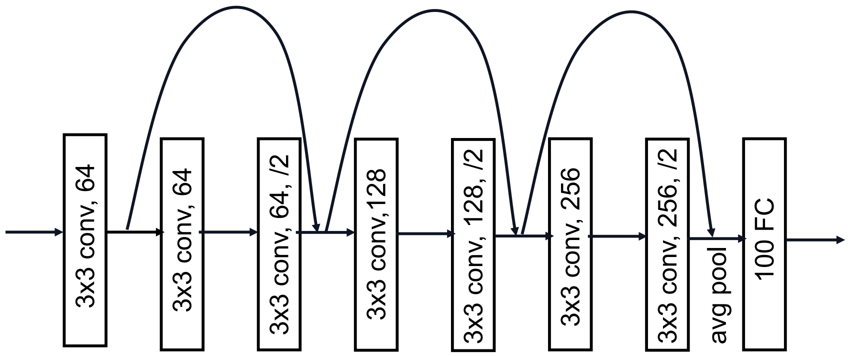

Figure 8: The architecture of ResNet-8.

C Appendix: The architecture of PlainNet-5

PlainNet-5 contains four convolution layers and one linear layer. Here, each convolution layer is followed by a ReLU layer and a BatchNorm layer. The first three convolution layer is also followed by a max pooling layer.

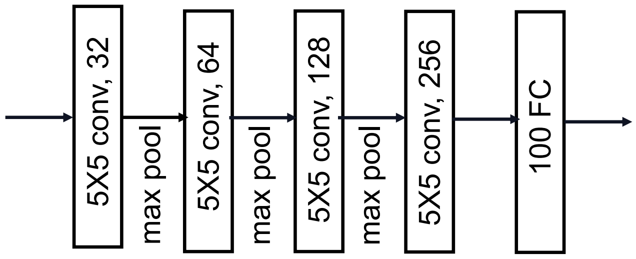

Figure 9: The architecture of PlainNet-5.