Learning What and Where to Transfer

Yunhun Jang*1 2  Hankook Lee*1 Sung Ju Hwang3 4 5  Jinwoo Shin1 4 5

Abstract
As the application of deep learning has expanded to real-world problems with insufficient volume of training data, transfer learning recently has gained much attention as means of improving the performance in such small-data regime. However, when existing methods are applied between heterogeneous architectures and tasks, it becomes more important to manage their detailed configurations and often requires exhaustive tuning on them for the desired performance. To address the issue, we propose a novel transfer learning approach based on meta-learning that can automatically learn what knowledge to transfer from the source network to where in the target network. Given source and target networks, we propose an efficient training scheme to learn meta-networks that decide (a) which pairs of layers between the source and target networks should be matched for knowledge transfer and (b) which features and how much knowledge from each feature should be transferred. We validate our meta-transfer approach against recent transfer learning methods on various datasets and network architectures, on which our automated scheme significantly outperforms the prior baselines that find “what and where to transfer” in a hand-crafted manner.

1. Introduction
Learning deep neural networks (DNNs) requires large datasets, but it is expensive to collect a sufficient amount of labeled samples for each target task. A popular approach for handling such lack of data is transfer learning (Pan & Yang, 2010), whose goal is to transfer knowledge from a known source task to a new target task. The most widely used method for transfer learning is pre-training with fine-tuning (Razavian et al., 2014): first train a source DNN (e.g. ResNet (He et al., 2016)) with a large dataset (e.g. ImageNet (Deng et al., 2009)) and then, use the learned weights as an initialization to train a target DNN. Yet, fine-tuning definitely is not a panacea. If the source and target tasks are semantically distant, it may provide no benefit. Cui et al. (2018) suggest to sample from the source dataset depending on a target task for pre-training, but it is only possible when the source dataset is available. There is also no straightforward way to use fine-tuning, if the network architectures for the source and target tasks largely differ.

Several existing works can be applied to this challenging scenario of knowledge transfer between heterogeneous DNNs and tasks. Learning without forgetting (LwF) (Li & Hoiem, 2018) proposes to use knowledge distillation, suggested in Hinton et al. (2015), for transfer learning by introducing an additional output layer on a target model, and thus it can be applied to situations where the source and target tasks are different. FitNet (Romero et al., 2015) proposes a teacher-student training scheme for transferring the knowledge from a wider teacher network to a thinner student network, by...
Learning What and Where to Transfer

using teacher’s feature maps to guide the learning of the student. To guide the student network, FitNet uses $\ell_2$ matching loss between the source and target features. Attention transfer (Zagoruyko & Komodakis, 2017) and Jacobian matching (Srinivas & Fleuret, 2018) suggest similar approaches to FitNet, but use attention maps generated from feature maps or Jacobians for transferring the source knowledge.

Our motivation is that these methods, while allowing to transfer knowledge between heterogeneous source and target tasks/architectures, have no mechanism to identify which source information to transfer, between which layers of the networks. Some source information is more important than others, while some are irrelevant or even harmful depending on the task difference. For example, since network layers generate representations at different level of abstractions (Zeiler & Fergus, 2014), the information of lower layers might be more useful when the input domains of the tasks are similar, but the actual tasks are different (e.g., fine-grained image classification tasks). Furthermore, under heterogeneous network architectures, it is not straightforward to associate a layer from the source network with one from the target network. Yet, since there was no mechanism to learn what to transfer to where, existing approaches require a careful manual configuration of layer associations between the source and target networks depending on tasks, which cannot be optimal.

Contribution. To tackle this problem, we propose a novel transfer learning method based on the concept of meta-learning (Naik & Mammon, 1992; Thrun & Pratt, 2012) that learns what information to transfer to where, from source networks to target networks with heterogeneous architectures and tasks. Our goal is learning to learn transfer rules for performing knowledge transfer in an automatic manner, considering the difference in the architectures and tasks between source and target, without hand-crafted tuning of transfer configurations. Specifically, we learn meta-networks that generate the weights for each feature and between each pair of source and target layers, jointly with the target network. Thus, it can automatically learn to identify which source network knowledge is useful, and where it should transfer to (see Figure 1). We validate our method, learning to transfer what and where (L2T-ww), to multiple source and target task combinations between heterogeneous DNN architectures, and obtain significant improvements over existing transfer learning methods. Our contributions are as follows:

- We introduce meta-networks for transfer learning that automatically decide which feature maps (channels) of a source model are useful and relevant for learning a target task and which source layers should be transferred to which target layers.
- To learn the parameters of meta-networks, we propose an efficient meta-learning scheme. Our main novelty is to evaluate the one-step adaptation performance (meta-objective) of a target model learned by minimizing the transfer objective only (as an inner-objective). This scheme significantly accelerates the inner-loop procedure, compared to the standard scheme.
- The proposed method achieves significant improvements over baseline transfer learning methods in our experiments. For example, in the ImageNet experiment, our meta-transfer learning method achieves 65.05% accuracy on CUB200, while the second best baseline obtains 58.90%. In particular, our method outperforms baselines with a large margin when the target task has an insufficient number of training samples and when transferring from multiple source models.

Organization. The rest of the paper is organized as follows. In Section 2, we describe our method for selective knowledge transfer, and training scheme for learning the proposed meta-networks. Section 3 shows our experimental results under various settings, and Section 4 states the conclusion.

2. Learning What and Where to Transfer

Our goal is to learn to transfer useful knowledge from the source network to the target network, without requiring manual layer association or feature selection. To this end, we propose a meta-learning method that learns what knowledge of the source network to transfer to which layer in the target network. In this paper, we primarily focus on transfer learning between convolutional neural networks, but our method is generic and is applicable to other types of deep neural networks as well.

In Section 2.1, we describe meta-networks that learn what to transfer (for selectively transfer only the useful channels/features to a target model), and where to transfer (for deciding a layer-matching configuration that encourages learning a target task). Section 2.2 presents how to train the proposed meta-networks jointly with the target network.

2.1. Weighted Feature Matching

If a convolutional neural network is well-trained on a task, then its intermediate feature spaces should have useful knowledge for the task. Thus, mimicking the well-trained features might be helpful for training another network. To formalize the loss forcing this effect, let $x$ be an input, and $y$ be the corresponding (ground-truth) output. For image classification tasks, $\{x\}$ and $\{y\}$ are images and their class labels. Let $S^m(x)$ be intermediate feature maps of the $m^{th}$ layer of the pre-trained source network $S$. Our goal is then to train another target network $T_\theta$ with parameter $\theta$ utilizing the knowledge of $S$. Let $T^m_\theta(x)$ be intermediate
Learning What and Where to Transfer

Figure 2. Our meta-transfer learning method for selective knowledge transfer. The meta-transfer networks are parameterized by \( \phi \) and are learned via meta-learning. The dashed lines indicate flows of tensors such as feature maps, and solid lines denote \( \ell_2 \) feature matching. (a) \( g^{m,n}_\phi \) outputs weights of matching pairs \( \lambda^{m,n} \) between the \( m \)th and \( n \)th layers of the source and target models, respectively, and (b) \( f^{m,n}_\phi \) outputs weights for each channel.

Where to transfer. When transferring knowledge from a source model to a target model, deciding pairs \((m, n)\) of layers in the source and target model is crucial to its effectiveness. Previous approaches (Romero et al., 2015; Zagoruyko & Komodakis, 2017) select the pairs manually based on prior knowledge of architectures or semantic similarities between tasks. For example, attention transfer (Zagoruyko & Komodakis, 2017) matches the last feature maps of each group of residual blocks in ResNet (He et al., 2016). However, finding the optimal layer association is not a trivial problem and requires exhaustive tuning based on trial-and-error, given models with different numbers of layers or heterogeneous architectures, e.g., between ResNet (He et al., 2016) and VGG (Simonyan & Zisserman, 2015). Hence, we introduce a learnable parameter \( \lambda^{m,n} \geq 0 \) for each pair \((m, n)\) which can decide the amount of transfer between the \( m \)th and \( n \)th layers of source and target models, respectively. We also set \( \lambda^{m,n} = g^{m,n}_\phi(S^m(x)) \) for each pair \((m, n)\) as an output of a meta-network \( g^{m,n}_\phi \) that automatically decides important pairs of layers for learning the target task.

The combined transfer loss given the weights of channels \( w \) and weights of matching pairs \( \lambda \) is

\[
\mathcal{L}_{\text{ufn}}(\theta|x, w^{m,n}) = \sum_{(m,n) \in \mathcal{C}} \lambda^{m,n} \mathcal{L}_{\text{ufn}}^{m,n}(\theta|x, w^{m,n}),
\]

where \( \mathcal{C} \) be a set of candidate pairs. Our final loss \( \mathcal{L}_{\text{total}} \) to train a target model then is given as:

\[
\mathcal{L}_{\text{total}}(\theta, x, y, \phi) = \mathcal{L}_{\text{org}}(\theta|x, y) + \beta \mathcal{L}_{\text{ufn}}(\theta|x, \phi).
\]

where \( \mathcal{L}_{\text{org}} \) is the original loss (e.g., cross entropy) and \( \beta > 0 \) is a hyper-parameter. We note that \( w^{m,n} \) and \( \lambda^{m,n} \) decide what and where to transfer, respectively. We provide an illustration of our transfer learning scheme in Figure 2.
2.2. Training Meta-Networks and Target Model

Our goal is to achieve high performance on the target task when the target model is learned using the training objective $L_{\text{total}}(\theta|x, y, \phi)$. To maximize the performance, the feature matching term $L_{\text{wfn}}(|x, \phi|$ should encourage learning of useful features for the target task, e.g., predicting labels. To measure and increase usefulness of the feature matching decided by meta-networks parameterized by $\phi$, a standard approach is to use the following bilevel scheme (Colson et al., 2007) to train $\phi$, e.g., see (Finn et al., 2017; Franceschi et al., 2018):

1. Update $\theta$ to minimize $L_{\text{total}}(\theta|x, y, \phi)$ for $T$ times.
2. Measure $L_{\text{org}}(\theta|x, y)$ and update $\phi$ to minimize it.

In the above, the actual objective $L_{\text{total}}$ for learning the target model is used in the inner-loop, and the original loss $L_{\text{org}}$ is used as a meta-objective to measure the effectiveness of $L_{\text{total}}$ for learning the target model to perform well.

However, since our meta-networks affect the learning procedure of the target model weakly through the regularization term $L_{\text{wfn}}$, their influence on $L_{\text{org}}$ can be very marginal, unless one uses a very large number of inner-loop iterations $T$. Consequently, it causes difficulties on updating $\phi$ using gradient $\nabla_{\phi} L_{\text{org}}$. To tackle this challenge, we propose the following alternative scheme:

1. Update $\theta$ to minimize $L_{\text{wfn}}(\theta|x, \phi)$ for $T$ times.
2. Update $\theta$ to minimize $L_{\text{org}}(\theta|x, y)$ once.
3. Measure $L_{\text{org}}(\theta|x, y)$ and update $\phi$ to minimize it.

In the first stage, given the current parameter $\theta_0 = \theta$, we update the target model for $T$ times via gradient-based algorithms for minimizing $L_{\text{wfn}}$. Namely, the resulting parameter $\theta_T$ is learned only using the knowledge of the source model. Since transfer is done by the form of feature matching, it is feasible to train useful features for the target task by selectively mimic the source features. More importantly, it increases the influence of the regularization term $L_{\text{wfn}}$ on the learning procedure of the target model in the inner-loop, since the target features are solely trained by the source knowledge (without target labels). The second stage is an one-step adaptation $\theta_{T+1}$ from $\theta_T$ toward the target label. Then, in the third stage, the task-specific objective $L_{\text{org}}(\theta_{T+1})$ can measure how quickly the target model has adapted (via only one step from $\theta_T$) to the target task, under the sample used in the first and second stage. Finally, the meta-parameter $\phi$ can be trained by minimizing $L_{\text{org}}(\theta_{T+1})$. The above 3-stage scheme encourages significantly faster training of $\phi$, compared the standard 2-stage one. This is because the former measures the effect of the regularization term $L_{\text{wfn}}$ more directly to the original $L_{\text{org}}$, and allows to choose a small $T$ to update $\phi$ meaningfully (we choose $T = 2$ in our experiments).

In the case of using the vanilla gradient descent algorithm for updates, the 3-stage training scheme to learn meta-parameters $\phi$ can be formally written as the following optimization task:

$$\min_{\phi} \ L_{\text{org}}(\theta_{T+1}|x, y)$$
subject to
$$\theta_{T+1} = \theta_T - \alpha \nabla_{\theta} L_{\text{org}}(\theta_T|x, y), \quad \theta_{t+1} = \theta_t - \alpha \nabla_{\theta} L_{\text{wfn}}(\theta_t|x, \phi), \quad t = 0, \ldots, T - 1,$$

where $\alpha > 0$ is a learning rate. To solve the above optimization problem, we use Reverse-HG (Franceschi et al., 2017) that can compute $\nabla_{\phi} L_{\text{org}}(\theta_{T+1}|x, y)$ efficiently using Hessian-vector products.

To train the target model jointly with meta-networks, we alternatively update the target model parameters $\theta$ and the meta-network parameters $\phi$. We first update the target model for a single step with objective $L_{\text{total}}(\theta|x, y, \phi)$. Then, given current target model parameters, we update the meta-networks parameters $\phi$ using the 3-stage bilevel training scheme described above. This eliminates an additional meta-training phase for learning $\phi$. The proposed training scheme is formally outlined in Algorithm 1.

3. Experiments

We validate our meta-transfer learning method that learns what and where to transfer, between heterogeneous network architectures and tasks.

3.1. Setups

Network architectures and tasks for source and target. To evaluate various transfer learning methods including ours, we perform experiments on two scales of image classification tasks, $32 \times 32$ and $224 \times 224$. For $32 \times 32$ scale,
Figure 3. (a)-(c) Matching configurations $C$ between ResNet32 (left) and VGG9 (right). (d) The amount $\lambda^{m,n}$ of transfer between layers after learning. Line widths indicates the transfer amount. We omit the lines when $\lambda^{m,n}$ is less than 0.1.

we use the TinyImageNet\(^1\) dataset as a source task, and CIFAR-10, CIFAR-100 (Krizhevsky & Hinton, 2009), and STL-10 (Coates et al., 2011) datasets as target tasks. We train 32-layer ResNet (He et al., 2016) and 9-layer VGG (Simonyan & Zisserman, 2015) on the source and target tasks, respectively. For $224 \times 224$ scale, the ImageNet (Deng et al., 2009) dataset is used as a source dataset, and Caltech-UCSD Bird 200 (Wah et al., 2011), MIT Indoor Scene Recognition (Quattoni & Torralba, 2009), Stanford 40 Actions (Yao et al., 2011) and Stanford Dogs (Khosla et al., 2011) datasets as target tasks. For these datasets, we use 34-layer and 18-layer ResNet as a source and target model, respectively, unless otherwise stated.

Meta-network architecture. For all experiments, we construct the meta-networks as 1-layer fully-connected networks for each pair $(m, n) \in C$ where $C$ is the set of candidates of pairs, or matching configuration (see Figure 3). It takes the globally average pooled features of the $m^{th}$ layer of the source network as an input, and outputs $u_{C}^{m,n}$ and $\lambda^{m,n}$. As for the channel assignments $u$, we use the softmax activation to generate them while satisfying $\sum_{c} u_{C}^{m,n} = 1$, and for transfer amount $\lambda$ between layers, we commonly use ReLU6 (Krizhevsky & Hinton, 2010), $\max(0, \min(6, x))$ to ensure non-negativity of $\lambda$ and to prevent $\lambda^{m,n}$ from becoming too large.

Compared schemes for transfer learning. We compare our methods with the following prior methods and their combinations: learning without forgetting (LwF) (Li & Hoiem, 2018), attention transfer (AT) (Zagoruyko & Komodakis, 2017) and unweighted feature matching (FM) (Romero et al., 2015).\(^2\) Here, AT and FM transfer knowledge on feature-level as like ours by matching attention maps or feature maps between source and target layers, respectively. The feature-level transfer methods generally choose layers just before down-scaling, e.g., the last layer of each residual group for ResNet, and match pairs of the layers of same spatial size. Following this convention, we evaluate two hand-crafted configurations (single, one-to-one) for prior methods and a new configurations (all-to-all) for our methods: (a) single: use a pair of the last feature in the source model and a layer with the same spatial size in the target model, (b) one-to-one: connect each layer just before down-scaling in the source model to a target layer of the same spatial size, (c) all-to-all: use all pairs of layers just before down-scaling, e.g., between ResNet and VGG architectures, we consider $3 \times 5 = 15$ pairs. For matching features of different spatial sizes, we simply use a bilinear interpolation. These configurations are illustrated in Figure 3. Among various combinations between prior methods and matching configurations, we only report the results of those achieving the meaningful performance gains.

3.2. Evaluation on Various Target Tasks

We first evaluate the effect of learning to transfer what (L2T-w) without learning to transfer where. To this end, we use conventional hand-crafted matching configurations, single and one-to-one, illustrated in Figure 3(a) and 3(b), respectively. For most cases reported in Table 1, L2T-w improves the performance on target tasks compared to the unweighted counterpart (FM): for fine-grained target tasks transferred from ImageNet, the gain of L2T-w over FM is more significant. The results support that our method, learning what to transfer, is more effective when target tasks have specific types of input distributions, e.g., fine-grained classification, while the source model is trained on a general task.

1https://tiny-imagenet.herokuapp.com/

2 In our experimental setup, we reproduce similar relative improvements from the scratch for these baselines as reported in the original papers. We do not report the results of Jacobian matching (JM) (Srinivas & Fleuret, 2018) as the improvement of LwF+AT+JM over LwF+AT is marginal in our setups.
Next, instead of using hand-crafted matching pairs of layers, we also learn where to transfer starting from all matching pairs illustrated in Figure 3(c). The proposed final scheme in our paper, learning to transfer what and where (L2T-ww), often improves the performance significantly compared to fixed matching pairs (L2T-w). As a result, L2T-ww achieves the best accuracy for all cases (with large margin) reported in Table 1, e.g., on the CUB200 dataset, we attain 10.4% relative improvement compared to the second best baseline.

Figure 3(d) shows the amounts $\lambda^{m,n}$ of transfer between pairs of layers after learning transfer from TinyImageNet to STL-10. As shown in the figure, our method transfers knowledge to higher layers in the target model: $\lambda^{2,5} = 1.40$, $\lambda^{1,5} = 2.62$, $\lambda^{3,4} = 2.88$, $\lambda^{2,4} = 0.74$. The amounts $\lambda^{m,n}$ of other pairs are smaller than 0.1, except $\lambda^{1,2} = 0.21$. Clearly, those matching pairs are not trivial to find by hand-crafted tuning, which justifies that our method for learning where to transfer is useful. Furthermore, since our method outputs sample-wise $\lambda^{m,n}$, amounts of transfer are adjusted more effectively compared to fixed matching pairs over all the samples. For example, amounts of transfer from source features $S^1(x)$ have relatively smaller variance over the samples (Figure 4(a)) compared to the those of $S^3(x)$ (Figure 4(c)). This is because higher-level features are more task-specific while lower-level features are more task-agnostic. It evidences that meta-networks $g_\phi$ adjust the amounts of transfer for each sample considering the relationship between tasks and the levels of abstractions of features.

### 3.3. Experiments on Limited-Data Regimes

When a target task has a small number of labeled samples for training, transfer learning can be even more effective. To evaluate our method (L2T-ww) on such limited-data scenario, we use CIFAR-10 as a target task dataset by reducing the number of samples. We use $N \in \{50, 100, 250, 500, 1000\}$ training samples for each class.
Learning What and Where to Transfer

Table 2. Classification accuracy (%) of VGG9 on STL-10 transferred from multiple source models. The first source model is ResNet32 trained on TinyImageNet. The additional source model is one of three: ResNet20 trained on TinyImageNet, another ResNet32 trained on TinyImageNet, and ResNet32 trained on CIFAR-10. We report the performance of the target model transferred from a single source model and two source models.

<table>
<thead>
<tr>
<th>First source model</th>
<th>TinyImageNet (ResNet32)</th>
<th>CIFAR-10 (ResNet32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TinyImageNet (ResNet20)</td>
<td>65.18 ± 0.91</td>
<td>65.18 ± 0.91</td>
</tr>
<tr>
<td>TinyImageNet (ResNet32)</td>
<td>68.64 ± 0.58</td>
<td>68.56 ± 0.56</td>
</tr>
<tr>
<td>Scratch</td>
<td>74.19 ± 0.22</td>
<td>73.78 ± 1.16</td>
</tr>
<tr>
<td>LwF+AT</td>
<td>75.06 ± 0.57</td>
<td>74.77 ± 0.30</td>
</tr>
<tr>
<td>AT</td>
<td>75.00 ± 0.34</td>
<td>75.99 ± 0.11</td>
</tr>
<tr>
<td>FM (single)</td>
<td>76.38 ± 1.18</td>
<td>77.69 ± 0.79</td>
</tr>
<tr>
<td>FM (one-to-one)</td>
<td>78.31 ± 0.21</td>
<td>79.35 ± 0.41</td>
</tr>
<tr>
<td>L2T-ww (all-to-all)</td>
<td>79.80 ± 0.52</td>
<td>80.52 ± 0.29</td>
</tr>
</tbody>
</table>

Figure 5. Transfer from TinyImageNet to CIFAR-10 with varying numbers of training samples per class in CIFAR-10. x-axis is plotted in logarithmic scale.

and compare the performance of learning from scratch, LwF, AT, LwF+AT and L2T-ww. The results are reported in Figure 5. They show that ours achieves significant more improvements compared to other baselines, when the volume of the target dataset is smaller. For example, in the case of N = 50, our method achieves 64.91% classification accuracy, while the baselines, LwF+AT, AT, LwF and scratch show 53.76%, 51.76%, 43.32% and 39.99%, respectively. Observe that ours needs only 50 samples per class to achieve similar accuracy of LwF with 250 samples per class.

3.4. Experiments on Multi-Source Transfer

In practice, one can have multiple pre-trained source models with various source datasets. Transfer from multiple sources may potentially provide more knowledge for learning a target task, however, using them simultaneously could require more hand-crafted configurations of transfer, such as balancing the transfer from many sources or choosing different pairs of layers depending on the source models. To evaluate the effects of using multiple source models, we consider the scenarios transferred from two source models simultaneously, where the models are different architectures (ResNet20, ResNet32) or trained on different datasets (TinyImageNet, CIFAR-10). In Table 2, we report the results of ours (L2T-ww) and other transfer methods on a target task STL-10 with 9-layer VGG as a target model architecture.

Our method consistently improves the target model performance over more informative transitions (from left to right in Table 2) on sources, i.e., when using a larger source model (ResNet20 → ResNet32) or using a different second source dataset (TinyImageNet → CIFAR-10). This is not the case for all other methods. In particular, compare the best performance of each method transferred from two TinyImageNet models and TinyImageNet+CIFAR-10 models as sources. Then, one can conclude that ours is the only one that effectively aggregates the heterogeneous source knowledge, i.e., TinyImageNet+CIFAR-10. It shows the importance of choosing the right configurations of transfer when using multiple source models, and confirms that ours can automatically decide the useful configuration from many possible candidate pairs for transfer.

3.5. Visualization

With learning what to transfer, our weighted feature matching will allocate larger attention to task-related channels of feature maps. To visualize the attention used in knowledge transfer, we compare saliency maps (Simonyan et al., 2014) for unweighted (FM) and weighted (L2T-w) matching between the last layers of source and target models. Saliency maps can be computed as follows:

\[ M_{i,j} = \max_c \left| \frac{\partial L_{\text{trans}}(\theta|x, w^{m,n})}{\partial x_{c,i,j}} \right| \]
Learning What and Where to Transfer

where $x$ is an image, $c$ is a channel of the image, e.g., RGB, and $(i, j) \in \{1, 2, \ldots, H\} \times \{1, 2, \ldots, W\}$ is a pixel position. For the unweighted case, we use uniform weights. On the other hand, for the weighted case, we use the outputs $w_{m,n} = f_{\phi}^{m,n}(S^m(x))$ of meta-networks learned by our meta-training scheme. Figure 6 shows which pixels are more or less activated in the saliency map of L2T-w compared to FM. As shown in the figure, pixels containing task-specific objects (birds or dogs) are more activated when using L2T-w, while background pixels are less activated. It means that the weights $w_{m,n}$ make knowledge of the source model be more task-specific, consequently it can improve transfer learning.

4. Conclusion

We propose a transfer method based on meta-learning which can transfer knowledge selectively depending on tasks and architectures. Our method transfers more important knowledge for learning a target task, with identifying what and where to transfer using meta-networks. To learn the meta-networks, we design an efficient meta-learning scheme which requires a few steps in the inner-loop procedure. By doing so, we jointly train the target model and the meta-networks. We believe that our work would shed a new angle for complex transfer learning tasks between heterogeneous or/and multiple network architectures and tasks.
Acknowledgements

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government MSIT (No.2016-0-00563, Research on Adaptive Machine Learning Technology Development for Intelligent Autonomous Digital Companion) and supported by the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921).

References


