Self-Supervised Knowledge Transfer via Loosely Supervised Auxiliary Tasks

Anonymous CVPR 2021 submission

Paper ID 11846

Abstract

Knowledge transfer using convolutional neural networks (CNN) is the way to efficiently train CNN with fewer parameters or to maximize generalization performance on limited supervision. To enable the more efficient transfer of pretrained knowledge under relaxed conditions, we propose a simple but powerful knowledge transfer methodology without the restrictions of the network structure and dataset, namely Self-Supervised Knowledge Transfer (SSKT) via loosely supervised auxiliary tasks. For this, we devise a training methodology that transfers previously learned knowledge to the current training process as an auxiliary task for the target task through self-supervision using the soft label. The SSKT does not depend on the network structure and dataset to train differently from the existing knowledge transfer methods; hence, it has the advantage that the prior knowledge acquired from various tasks can be naturally transmitted during the training process to the target task. Further, it can improve the generalization performance in most datasets through the proposed knowledge transfer between different problem domains from multiple source networks. The SSKT improves generalization performance through experiments under various knowledge transfer settings. The source code will open to the public[1].

1. Introduction

Knowledge transfer is the most representative training methodology to improve generalization capability and training efficiency for convolutional neural networks (CNN). The most widely used knowledge transfer is transfer learning [36, 34] based on using pretrained weights trained on large scale data sets as initial values for new tasks. The pretrained weights is used as a feature encoder after finetuning for different vision tasks such as image classification, object detection, or semantic segmentation [24, 25, 29]. Taskonomy [37, 38] provides an extensive database and analysis approach for analyzing the effects

[1] https://github.com/generation21/generation6011

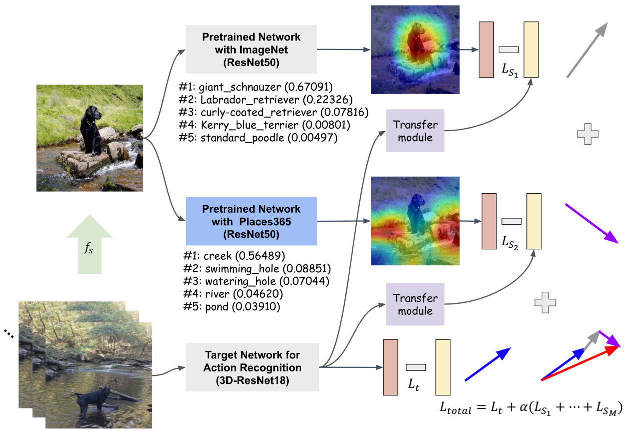
Pretrained Network with ImageNet (ResNet50)
#1: giant_schnauzer (0.67091)
#2: Labrador_retriever (0.22326)
#3: curly-coated_retriever (0.07816)
#4: Kerry_blue_terrier (0.00801)
#5: standard_poodle (0.00497)
Transfer module
L_{S_1}
f_s
Pretrained Network with Places365 (ResNet50)
#1: creek (0.56489)
#2: swimming_hole (0.08851)
#3: watering_hole (0.07044)
#4: river (0.04620)
#5: pond (0.03910)
Transfer module
L_{S_2}
Target Network for Action Recognition (3D-ResNet18)
L_t
$L_{total} = L_t + \alpha(L_{S_1} + \cdots + L_{S_M})$

Figure 1. **Motivation of the SSKT.** The CNN responses that can be obtained from the same image vary depending on the type of supervision. SSKT performs auxiliary training using soft labels to convey the previously trained prior knowledge from the source tasks. At this time, the supervision used in the auxiliary training generates a gradient (grey and purple arrows) that can help the generalization performance of the target task. The gradient to update the weight of the target network is obtained as a linear combination of all losses (red arrow). Refer to Equation (7) for the notation of each loss term and the final loss function.

of transfer learning using pretrained networks on the finetuning of target tasks. However, transfer learning using pretrained networks presupposes structural dependencies that must share essentially the same (whole or partial) network structure from the source network. At the same time, it has been questioned whether transfer learning using pretrained weights provides a good initial weights, which is helpful for improving convergence or training with less data but not very effective for knowledge transfer to different task [12].

Another representative example of knowledge transfer using CNN is the knowledge distillation (KD) [14], where the pretrained teacher networks distill and deliver dark (hidden) knowledge in the inference output or learned features during the student network training. KD methods include the loss-based KD, which delivers dark knowledge through loss with soft label [14] and KD by increasing the similarity between features extracted at specific stages of the

CNN [30]. Knowledge Transfer through KD has the advantage of training the student networks that have a generalized performance comparable to the teacher networks by using fewer parameters. In general KD setup, the training dataset of both teacher and student networks should be the same. However, for the more general knowledge transfer, these constraints must be relaxed.

Figure 1 shows an example that the information to be transferred can vary according to the type of supervision of each pretrained CNN through Grad-CAM visualization for the same input image. For this, we propose a simple but powerful method that aims for better generalization with minimal additional supervision. The proposed knowledge transfer conveys prior knowledge as a soft label from multiple pretrained networks obtained with different types of supervision in a self-supervised manner. The process of the proposed knowledge transfer utilizes a pretrained network but does not depend on the structure of the target network or the training dataset of the pretrained network to train target task. The target network updates through the linear combination of the target loss and multiple auxiliary losses with the soft label-based self-supervision from the pretrained networks together by trying to guess what type of knowledge inferred from source network (Figure 1). We expect that the target network reflects the dark knowledge as a gradient from multiple supervision of the existing training data through soft-label-based self-supervision. Since it does not refer to the previously trained data and performs auxiliary training in a self-supervised manner, proposed knowledge transfer can be performed within any task with only the final output dimension (number of classes) required for calculation of each auxiliary loss. We call this form of knowledge transfer the *Self-Supervised Knowledge Transfer* (SSTK). The technical contributions of the SSKT are:

- Unlike former knowledge transfer techniques, the SSKT does not depend on the structure of the source networks or training dataset. Even SSKT-based knowledge transfer is possible between structurally complete heterogeneous networks, such as 2D to 3D CNN tknowledge transfer.
- The SSKT has the potential for simultaneous knowledge transfer from multiple pretrained networks trained from completely different types of datasets or supervision.
- The SSKT has improved generalization performance in most knowledge transfer scenarios. Particularly in different problem domains, SSKT achieves superior performance compared to existing KD-based knowledge transfers.

Figure 2 shows the differences between SSKT and other knowledge transfer methods. For clarity, we denote that the *problem domain* is a superset of tasks (e.g. image classification, action recognition, object detection, and semantic segmentation) and the *task* is bounded in the same problem domain (e.g. ImageNet, Places365, and STL10 in image classification).

2. Relalted Works

Transfer Learning with Pretrained Initializer. As the most commonly used method of knowledge transfer, it has been applied to various fields of computer vision using high-performance models [20, 32, 13] with large training data. Due to the belief that low-level feature information is shared by most visual recognition problems, one often applies fine-tuning by updating only a specific upper-layer when learning new tasks [25, 29]. Generally, this can be used in the same network structure, and many image recognition applications use only the feature encoder part of the entire task architecture [24, 25, 29]. Although transfer learning using ImageNet has been very successful so far, questions on the transfer learning using ImagNet data have been raised [12]; the experimental results of object recognition and key point recognition problems were reported as examples where the pretrained initial value does not convey additional knowledge to the different problem domains.

Knowledge Distillation. Pretrained teacher networks with larger parameters can improve the generalization performance of the student networks by performing knowledge transfer during the training process to a relatively small number of parameters [14]. Often, student networks show better generalization performance over teacher networks [8]. Recent knowledge distillation techniques for effective knowledge transfer used the relation between the inferred posteriors from teacher and student networks [27] or the dataset as a subclass of the original classes to be learned [26]. In [8], the ensemble model achieved high recognition performance by distilling the knowledge of student networks with different initialization values by using teacher and student networks with the same structure. The SSKT uses a knowledge transfer method such as KD with a soft label. However, unlike the existing KD methods, since the SSKT transfers the knowledge through the auxiliary task-based self-supervision technique, there is no dataset dependency.

Deep Mutual Learning (DML). DML is one of the representative variations of KD-based knowledge transfer method to use multiple networks for efficient training through KD loss within networks [39]. DML was able to achieve performance improvement through mutual learning in the middle of training without a powerful pretrained teacher network. Recently, a learning methodology using feature fusion [16], generative adversarial networks [3], or collaborative learning [35] structures has been proposed

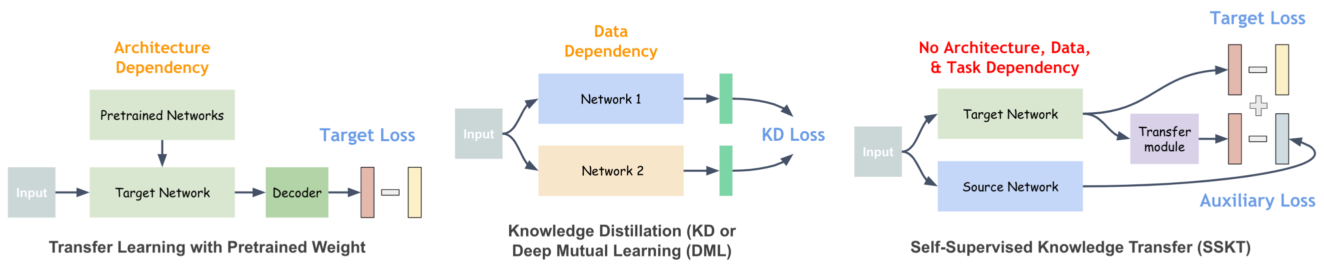

Figure 2. **Schematic of the difference between the proposed SSKT and other knowledge transfer methods.** Unlike existing knowledge transfer methods such as KD and DML, SSKT is not dependent on the network structure or data. In addition, generalization performance improvements can be achieved without additional supervision during the training process.

to improve the efficiency of mutual learning during training. It differs from SSKT in that DML performs multi-task training-based learning on the same dataset and updates all networks during training. Besides, it is fundamentally different from SSKT in that DML not utilizes prerequisite knowledge from former training.

Auxiliary Learning. Unlike for the general multi-task learning, in the case of auxiliary task learning the purpose of a deep neural network is not to increase the performance of auxiliary tasks, but to improve the performance of the target task. It was recently reported that the auxiliary task learning could help improve the test performance of the target task, and the cosine similarity of the gradient of loss of the target task and the auxiliary task was analyzed to visualize the cause of contribution by the auxiliary task [6]. The basic structure of the SSKT was inspired by meta-learning techniques using auxiliary tasks under self-supervision [23]. Unlike the multi-task learning using general auxiliary tasks with certain supervision, SSKT does not require any additional supervision and does not share part of the networks. Besides, updates to the source networks are not performed during training.

Self-Supervised Learning (SSL). Typical self-supervised learning techniques train pretrained weights that perform unsupervised learning on pretext tasks to obtain good initial weights [9, 1, 2, 11]. The pretrained network utilized in SSKT was not obtained by unsupervised learning, and the target task training differed from the general self-supervised approach, requiring meta information such as the number of classes used in the source networks training. However, SSKT can still be regarded as self-supervision from the viewpoint of training auxiliary tasks by using only soft labels from pretrained networks, without requiring additional supervision with hard labels when training the auxiliary task.

Domain Adaptation or Expansion. Domain adaptation or extension [22, 17] aims to improve the generalization performance for new domains with existing trained networks. However, unlike in the case of the general domain adaptation problem, SSKT has a big difference in using the prior knowledge trained in the existing domain for new network training without sharing parameters. Besides, because it does not include any update process for the pretrained network, the update is performed regardless of the performance of the source domain. Moreover, there is no change in the inference performance of the source domain after training.

3. Self-Supervised Knowledge Transfer

SSKT supports a training structure that enables knowledge transfer in a variety of scenarios using CNN. Various scenarios refer to situations not influenced by the network architecture of the target task to be trained and in which the knowledge transfer does not depend on the type of task. We devised a structure that transfers knowledge naturally, without compromising the training information of the pre-trained network or requiring additional supervision in the target task training process. We achieved this goal using the soft label-based knowledge transfer techniques with auxiliary task learning through self-supervision, for the various domain of image recognition variants.

3.1. SSKT with Single Source

We define a multi-task network for auxiliary learning, $h_t(x; \theta_t, D_t, T_t)$, where x is the input, θ_t is a parameter of the target network, D_t is a target dataset, and T_t is the task to be trained. θ_t is updated simultaneously through target loss and auxiliary loss during training to solve the primary task. $h_s(x; \theta_s, D_s, T_s)$ describes a source network that receives the input x and delivers knowledge to the target network. θ_s denotes a parameter trained by the source task T_s for the source data set (ImageNet, Places, etc.) D_s. θ_s is not updated during the target task training.

The multi-task network h_t for the primary task training shares up to the top layer of the convolutional block below primary and auxiliary branch, except the last feature layer. The last feature layer of the branch for the target task loss outputs $h_t^{prim}(x; \theta_t, D_t, T_t)$, and the branch for the auxiliary task loss outputs $h_t^{aux}(x; \theta_t, D_t, T_s)$. At this time, the auxiliary task branch could have a transfer module composed of summation of bottleneck structure from each convolutional bocks for effective feature encoding for the auxiliary task. Figure 3 shows an example of a multi-task network with a transfer module. The ground truth label of the primary task for the total loss of the h_t is given by y_t^{prim}, for x belonging to D_t. The ground truth label for the auxiliary task is given by the softmax output y_s^{aux} from h_s.

The total loss function for a single task knowledge transfer based on a multitask network is as follows:

$$\underset{\theta_t}{\text{argmin}} \Big(L(h_t^{prim}(x_i;\theta_t.D_t,T_t), y_{t,i}^{prim}) \\ +\alpha L(h_t^{aux}(x_i;\theta_t.D_t,T_s), y_{s,i}^{aux}) \Big), \quad (1)$$

where i is the i^{th} batch of the training data, α is balanced parameter for total loss, and $y_{s,i}^{aux} = h_s(x_i;\theta_s,D_s,T_s)$ is the softmax output from the pretrained source network and conveys the dark knowledge of the pretrained dataset by soft labels. We applied two types of losses of cross-entropy (CE), and knowledge distillation (KD) [14], for auxiliary task learning:

$$L_{CE} = -ylog(c(\frac{g(x;\theta)}{T})), \quad (2)$$

$$L_{KD} = KL\Big(c(\frac{g(x;\theta_1)}{T}), c(\frac{g(x;\theta_2)}{T})\Big), \quad (3)$$

where y is one-hot vector from ground truth label, $g(x;\theta)$ is the final feature output obtained from the network during training, T controls the softening intensity for the softmax output as the temperature parameter, c denotes the softmax function, and KL denotes the KL divergence between output distributions. The gradient update after $k+1$ iterations is described by:

$$\theta_t^{k+1} = \theta_t^k - \eta\nabla_{\theta_t}\Big(L(h_t^{prim}(x_i;\theta_t.D_t,T_t), y_{t,i}^{prim}) \\ +\alpha L(h_t^{aux}(x_i;\theta_t.D_t,T_s), y_{s,i}^{aux}) \Big) \quad (4)$$

where η is the learning rate. Figure 2 (a) provides the structure schematics for the SSKT with a single-source scenario.

Transfer Module (TM). To encourage predicting $y_{s,i}^{aux}$ by h_t, we design bottleneck structure based transfer module supporting auxiliary task using feature output from each convolutional block. For an example of a ResNet based model, each output of res block encoded as fixed size output by bottleneck and average pooling, then summed each feature output from transfer bottleneck (See in Figure 3).

3.2. SSKT with Multiple Source

SSKT easily extends to multiple h_s using multiple auxiliary losses. We have improved the structure by including M source tasks for knowledge transfer to target tasks, as follows:

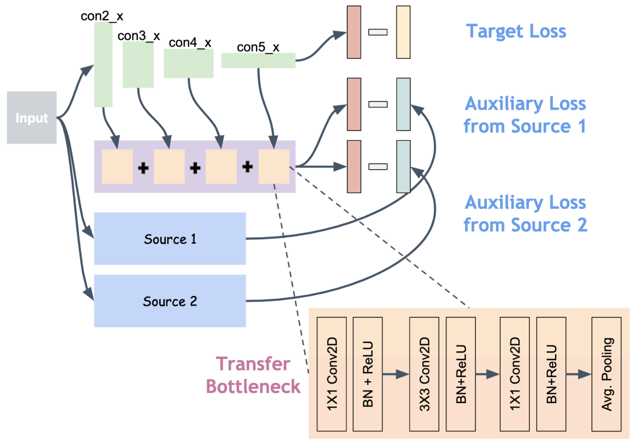

Figure 3. **Schematic of the transfer modules for efficient auxiliary learning.** The transfer module used in the SSKT consists of summation of feature output of bottleneck layers from each convolutional block. Schematic shows and example of the transfer module with ResNet variants for SSKT using multiple sources.

$$\underset{\theta_t}{\text{argmin}} \Big(L(h_t^{prim}(x_i;\theta_t.D_t,T_t), y_{t,i}^{prim}) \\ +\alpha(L(h_t^{aux}(x_i;\theta_t.D_t,T_{s_1}), y_{s_1,i}^{aux}) \\ +L(h_t^{aux}(x_i;\theta_t.D_t,T_{s_2}), y_{s_2,i}^{aux}) + \dots \\ +L(h_t^{aux}(x_i;\theta_t.D_t,T_{s_M}), y_{s_M,i}^{aux})) \Big), \quad (5)$$

where s_M denotes the index of the source task for knowledge transfer. Theoretically, if the memory is sufficient, we can perform multiple task knowledge transfer from a large number of M source tasks.

3.3. SSKT within Different Problem Domains

Until now, transfers for single and multiple sources have been performed assuming tasks, in the same problem domain, that do not specify the form of D_t and D_s. We set up a scenario to enable knowledge transfer even when the modalities of D_t and D_s are different, or the problem domains of tasks are different. In this case, because the characteristics of the input data are different and there is a big difference in the structure of the network to train, the preparation process for knowledge transfer involves data conversion or preprocessing. For knowledge transfer with different problem domains, the total loss function can be defined as:

$$\underset{\theta_t}{\text{argmin}} \Big(L(h_t^{prim}(x_i;\theta_t.D_t,T_t), y_{t,i}^{prim}) \\ +\alpha L(h_t^{aux}(f_s(x_i);\theta_t.D_t,T_s), y_{s,i}^{aux}) \Big), \quad (6)$$

where the data transformation function f_s converts the data type to match the source task to infer the recognition information to the task of the source domain. For example, if

Table 1. Training details for each task, for knowledge transfer with SSKT. IC denotes Image Classification and AC for Action Classification, MCIC for Multi-Class Image Classification, β represents momentum, λ represents weight decay. CE denotes Cross Entropy, KD for Knowledge Distillation, and BCE for Binary Cross Entropy. In the training model, (s) was trained by learning from scratch and (f) was trained by fine-tuning.

Task (T)	Dataset (D)	# cls	Model	Optimizer	LR Scheduler	β,λ	Loss [Task + Auxiliary]
IC (T_t)	CIFAR10 (C10)	10	ResNet[20, 32]	SGD	step (0.1,[150:250:350])	0.9, 5e-4	[CE + CE or KD]
	CIFAR100 (C100)	100	ResNet[20, 32]	SGD	step (0.1,[60:120:160:200])	0.9, 5e-4	[CE + CE or KD]
	STL10 (S10)	10	ResNet18[20, 32, 44]	SGD	step (0.1,[60:120:160:200])	0.9, 5e-4	[CE + CE or KD]
			MoblieNetV2	SGD	step (0.1,[60:120:160:200])	0.9, 5e-4	[CE + CE or KD]
			DenseNet121	SGD	step (0.1,[60:120:160:200])	0.9, 5e-4	[CE + CE or KD]
	Places365 (P)	365	ResNet18	SGD	step (0.1,[30:60:90])	0.9, 1e-4	[CE + CE or KD]
	ImageNet (I)	1000	ResNet18	SGD	step (0.1,[30:60:90])	0.9, 1e-4	[CE + CE or KD]
MCIC (T_t)	Pascal VOC (VOC)	20	ResNet[18, 34, 50] (s)	SGD	step (0.1,[30:60:90])	0.9, 1e-4	[BCE + CE or KD]
			ResNet18 (f)	SGD	step (0.01,[30:60:90])	0.9, 1e-4	[BCE + CE or KD]
AC (T_t)	UCF101 (U101)	101	3D-ResNet18 (s)	SGD	reduce on plateau (0.1)	0.9, 1e-3	[CE + CE or KD]
			3D-ResNet18 (f)	SGD	reduce on plateau (0.01)	0.9, 1e-3	[CE + CE or KD]
	HMDB51 (H51)	51	3D-ResNet18 (s)	SGD	reduce on plateau (0.1)	0.9, 1e-3	[CE + CE or KD]
			3D-ResNet18 (f)	SGD	reduce on plateau (0.01)	0.9, 1e-3	[CE + CE or KD]
IC (T_s)	Places365 (P)	365	ResNet50	[40]	[40]	[40]	[40]
	ImageNet (I)	1000	ResNet50	[28]	[28]	[28]	[28]

T_t is an action recognition problem using 3D-CNN, the input $x^{w \times h \times d} \in D_t$ is defined as a three-dimensional tensor. In this case, if a pretrained network for knowledge transfer is obtained through the image recognition problem T_s using 2D-CNN, $f_s : x^{w \times h \times d} \rightarrow \hat{x}^{w \times h}$ should be defined as a function that maps a three-dimensional tensor to a two-dimensional matrix into which h_s can be input.

Knowledge transfer to other problem domains can also perform knowledge transfer with multiple (M) source tasks:

$$\begin{aligned} \underset{\theta_t}{\operatorname{argmin}} \Big(& L(h_t^{prim}(x_i; \theta_t.D_t, T_t), y_{t,i}^{prim}) \\ & + \alpha(L(h_t^{aux}(f_{s_1}(x_i); \theta_t.D_t, T_{s_1}), y_{s_1,i}^{aux}) \\ & + L(h_t^{aux}(f_{s_2}(x_i); \theta_t.D_t, T_{s_2}), y_{s_2,i}^{aux}) + \dots \\ & + L(h_t^{aux}(f_{s_M}(x_i); \theta_t.D_t, T_{s_M}), y_{s_M,i}^{aux})) \Big), \end{aligned} \tag{7}$$

In this case, up to M transformation functions could be defined. Gradient updates for knowledge transfer to other problem domains are defined in the same way as single or multiple task knowledge transfers between the same domains.

4. Experiments Results

To verify the SSKT, we designed a series of experimental scenarios. First, we set a knowledge transfer with a single source task in the same problem domain. Pretrained CNN with image classification tasks (ImageNet, Places365, etc.) was used for knowledge transfer for the same problem domain. Second, we performed a knowledge transfer for the image classification problem using multiple source networks. In this case, the target network for target task learning has two or more auxiliary tasks. Third, knowledge transfer scenarios using source networks with different problem domains were tested in two manners. The first manner consists of a target task learning with a different purpose and a knowledge transfer to a domain within the same image recognition category. For example, to solve the multi-class classification problem for 2D images, the target network (2D-CNN) has the same type of architecture as the source network (2D-CNN). In this case, knowledge transfer between domains is possible without any special definition of the transformation function f_s for the source network. The second manner is a scenario of knowledge transfer using different problem domains with heterogeneous networks. For example, in training the action recognition network using 3D-CNN, we verified whether the network learned by the image classification problem could transfer the knowledge to the action recognition problem.

In addition to verifying the knowledge transfer performance of SSKT for each knowledge transfer scenario, the optimization result for the hyper-parameters of loss functions was included. At the same time, our experiments include the relationship between the change of knowledge transfer performance according to the model architecture and finetuning, and performance comparison with other knowledge transfer methods such as KD and DML. All the experimental results included in the table are selected based on the high performance among combinations of auxiliary loss, transfer module, and source network[2]. Table 1 summarizes the experimental settings for the verification of the SSKT.

4.1. Image Classification → Image Classification

SSKT Setting. To perform SSKT between image classification problems, we used networks pretrained by ImageNet (I) [5] or Places365 (P) [40] dataset as the source network.

[2]All combinations of test results and architecture details can be found in the supplement.

The target task's datasets were CIFAR10, CIFAR100 [19], and STL10 [4], which contain relatively small amounts of data, as well as large datasets such as ImageNet and Places365. As a pretrained source network, we used the ResNet50 network provided by the torchvision package of PyTorch [28]. The network for the target task and the training setup used for training is shown in Table 1. The transfer module is composed of bottleneck and average pooling shown in Figure 3. Multiple pretrained networks can be utilized in SSKT to apply multi-source task-based knowledge transfer. We used both a ResNet50 model trained on ImageNet and Places365 (P+I) datasets as our source tasks. The training details are shown in Table 1.

Results. Table 2 shows the performances of the learning from scratch and SSKT in the image classification problem in CIFAR10 (C10), 100 (C100), and STL10 (S10). For the knowledge transfer between image classification problems, the use of SSKT improves the generalization performance for all test datasets. The performance change when the transfer module is applied to efficiently transform features for auxiliary task learning is also shown. Notably, the SLT10 dataset with a small proportion of the training data showed the greatest generalization performance improvement. This means that knowledge transfer using SSKT is more effective when the amount of supervision is relatively small. Table 3 shows the performance changes of SSKT for large scale image datasets. When SSKT was applied to both Place365 and ImageNet, there was a steady performance improvement. In particular, the greatest performance improvement was obtained when source networks were used at the same time (P+I). However, in the case of large datasets, it was difficult to confirm the effect of improving the performance of the transfer module.

4.2. Image Classification → Multi-class Image Classification

SSKT Setting. To verify whether knowledge transfer using SSKT is effective for knowledge transfer between different problem domains, knowledge transfer between image classification and multi-class image classification was performed. We tested the multi-class image classification using the PASCAL VOC dataset [7] as the target task. The target loss function for solving the multi-class image classification was used as Binary Cross-Entropy (BCE). The training details for the target task and the source task are shown in Table 1.

Results. Table 4 shows the performance of learning from scratch for multi-class image classification and the performance change when SSKT is included in the training process. For PASCAL VOC multi-class classification task, we use the standard average precision (AP) as accuracy to evaluate the predictions. The use of SSKT significantly improved the performance of the knowledge transfer between different problem domains. The improvement was greater than for the knowledge transfer between the same problem domains. Besides, we verified an additional performance improvement when applying SSKT from multi-source networks.

Table 2. Performance change of SSKT for a single source compared to the training from scratch. The best performing model for each dataset is highlighted in bold. All experiments evaluated test performance 3 times from the same random seed for the model. TM denotes Transfer Module and R[depth] denotes ResNet structure.

T_s	T_t	Model	Method	TM	Loss	acc.
-	C10	R20	scratch	-	CE	92.19±0.09
P		R20	SSKT	o	CE+KD	92.25±0.04
I		R20	SSKT	o	CE+CE	92.44±0.05
P+I		R20	SSKT	x	CE+KD	**92.46±0.15**
-		R32	scratch	-	CE	93.21±0.09
P		R32	SSKT	x	CE+KD	92.87±0.31
I		R32	SSKT	x	CE+CE	93.26±0.08
P+I		R32	SSKT	o	CE+CE	**93.38±0.02**
-	C100	R20	scratch	-	CE	68.26±0.36
P		R20	SSKT	x	CE+KD	68.01±0.42
I		R20	SSKT	o	CE+CE	**68.63±0.12**
P+I		R20	SSKT	o	CE+CE	68.56±0.23
-		R32	scratch	-	CE	70.33±0.19
P		R32	SSKT	x	CE+CE	69.97±0.16
I		R32	SSKT	o	CE+CE	70.75±0.06
P+I		R32	SSKT	o	CE+CE	**70.94±0.36**
-	S10	R20	scratch	-	CE	81.15±0.34
P		R20	SSKT	o	CE+CE	82.76±0.05
I		R20	SSKT	o	CE+CE	83.45±0.07
P+I		R20	SSKT	o	CE+CE	**84.56±0.35**
-		R32	scratch	-	CE	81.19±0.17
P		R32	SSKT	o	CE+CE	83.06±0.27
I		R32	SSKT	o	CE+CE	**83.68±0.28**
P+I		R32	SSKT	o	CE+CE	83.4±0.2
-		R44	scratch	-	CE	80.18±0.54
P		R44	SSKT	o	CE+CE	82.68±0.39
I		R44	SSKT	o	CE+CE	**83.59±0.13**
P+I		R44	SSKT	o	CE+CE	83.44±0.15

Table 3. Performance change by applying the SSKT and from scratch training on a large scale image dataset.

T_s	T_t	Model	Method	TM	Loss	acc.
-	P	R18	scratch	-	CE	50.92
P		R18	SSKT	o	CE+CE	54.5
I		R18	SSKT	o	CE+CE	53.67
P+I		R18	SSKT	x	CE+CE	**54.78**
-	I	R18	scratch	-	CE	64.14
P		R18	SSKT	o	CE+CE	64.99
I		R18	SSKT	x	CE+CE	67.79
P+I		R18	SSKT	x	CE+CE	**70.57**

Table 4. SSKT results from the image classification problem to the multi-class image classification problem.

T_s	T_t	Model	Method	TM	Loss	acc.
-	VOC	R18	scratch	-	BCE	67.28±0.25
P		R18	SSKT	o	BCE+CE	74.76±0.17
I		R18	SSKT	x	BCE+CE	74.78±0.09
P+I		R18	SSKT	o	BCE+CE	**76.42±0.06**
-		R34	scratch	-	BCE	66.0±0.49
P		R34	SSKT	o	BCE+CE	75.65±0.12
I		R34	SSKT	o	BCE+CE	75.14±0.14
P+I		R34	SSKT	o	BCE+CE	**77.02±0.02**
-		R50	scratch	-	BCE	61.16±0.34
P		R50	SSKT	o	BCE+CE	74.44±0.06
I		R50	SSKT	o	BCE+CE	74.24±0.05
P+I		R50	SSKT	o	BCE+CE	**77.1±0.14**

4.3. Image Classification → Action Classification

SSKT Setting. Even though there is a difference between problem domains, the performance improvement of knowledge transfer may be relatively predictable because the modalities of the input data of the source task and the target task are the same. We used a pretrained image classification network as the source task and set the target task as an action recognition problem. We employed 3D-CNN to design an experiment for more extended knowledge transfer. To utilize the dark knowledge of the source network as described in Equation (6), we need to define the conversion function $f_s : x^{w \times h \times d} \rightarrow \hat{x}^{w \times h}$. Because SSKT's is not designed to define good f_s, we simply limited the role of f_s to extracting the center frame from a three-dimensional video clip. As a transfer module for inferring auxiliary tasks in 3D-CNN, the output features of the target model were further processed using a 3D bottleneck structure. We used a 3D ResNet-based baseline model [10] to train action classification networks. The training details for the 3D ResNet and source networks are shown in Table 1.

Results. Table 5 shows the performance of learning from scratch with the 3D-ResNet for UCF101 [33] and HMDB51 [21] datasets, as well as the performance change when SSKT is used in the training process. All results were obtained from *split 1* of each dataset. Note that the auxiliary task in SSKT has the effect of preventing overfitting during the training of the 3D-CNN model. Similar to other SSKT settings, there was the greatest performance improvement when using a multi-source network. We further note that the use of SSKT between heterogeneous problem domains can improve the performance further, depending on how f_s is defined.

4.4. Further Analysis and Discussion

Parameter Optimization. We performed a series of experiments for the temperature parameter T included in the auxiliary loss, the balance parameter α included in the total loss, and the presence of the transfer module. Figure 4 shows graphs of parameter optimization results for the STL10 dataset and PASCAL VOC data. For the STL10, the use of a transfer module in the SSKT process showed high performance in most cases, but in the case of PASCAL VOC, it was found that the transfer module did not play a big role.

Table 5. SSKT results from the image classification to the action classification with different modalities of input data.

T_s	T_t	Model	Method	TM	Loss	acc.
-	U101	3DR18	scratch	-	CE	43.28
P		3DR18	SSKT	o	CE+CE	45.35
I		3DR18	SSKT	x	CE+CE	46.62
P+I		3DR18	SSKT	x	CE+CE	**52.19**
-	H51	3DR18	scratch	-	CE	17.14
P		3DR18	SSKT	o	CE+CE	18.77
I		3DR18	SSKT	o	CE+KD	18.77
P+I		3DR18	SSKT	o	CE+CE	**20.54**

Table 6. SSKT results using MobileNet V2 (MV2) and DenseNet121 (D121).

T_s	T_t	Model	Method	TM	Loss	acc.
-	S10	R20	scratch	-	CE	81.15±0.34
-		MV2	scratch	-	CE	72.26±0.83
-		D121	scratch	-	CE	72.02±0.48
P+I		R20	SSKT	o	CE+ CE	**84.56±0.35**
P+I		MV2	SSKT	o	CE+CE	**76.96±0.39**
P+I		D121	SSKT	x	CE+CE	**77.03±0.17**

Model Comparison. We applied SSKT to MobileNet V2 (MV2) [31] and DenseNet121 (D121) [15] to evaluate model performance in CNN architectures other than ResNet variants. The source network used the ResNet50 model, which was previously trained with Places365 and ImageNet, like other SSKT settings. In evaluating the STL10 dataset, Table 6 shows that MobileNet V2 and DenseNet121 has performance improvements with the SSKT. The results in the table are the highest performance in configuration among single and multi-source SSKT settings.

Relation to Finetuning. In SSKT scenarios, finetuning as well as learning from scratch can be applied for target network training. We adopted finetuning-based SSKT with a ResNet18 model trained in ImageNet in a multi-class image classification problem using the PASCAL VOC dataset. Besides, using the pretrained 3D-ResNet18 model with the Kinetics-400 dataset [18], SSKT was performed on the action classification problem for the UCF101 and HMDB51 datasets. Table 7 shows that even if finetuning based on pretrined weights was performed, performance was improved in all experimental settings. However, unlike the evaluation results of from scratch, the performance improvement from multi-source was not significant, and the best performance was obtained when KD loss was used.

Comparison with Other Knowledge Transfer Methods. For further analysis of SSKT, we compared the performance of SSKT with typical knowledge transfer, KD, and DML. For KD, the details for learning were set the same as for

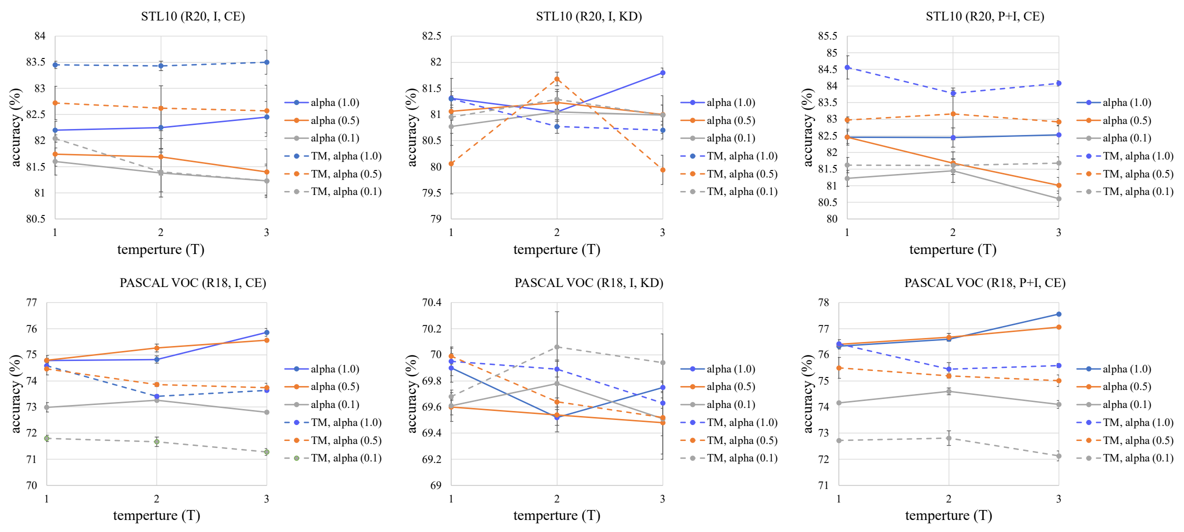

Figure 4. **Graph visualization for parameter optimization of SSKT.** The title of each graph is composed of D_t (target model, T_s, auxiliary loss). T is the temparture parameter of each auxiliary loss, and α is the balance parameter of the total loss.

Table 7. SSKT results using pretrained weights. *ft* denotes finetuning and K denotes Kinetics-400 dataset.

T_s	T_t	Model	Method	TM	Loss	acc.
-	VOC	R18	*ft* (I)	-	CE	90.52±0.11
P		R18	SSKT	x	CE+KD	**92.28±0.06**
I		R18	SSKT	x	CE+KD	92.26±0.07
P+I		R18	SSKT	o	CE+KD	92.25±0.07
-	U101	3DR18	*ft* (K)	-	CE	83.95
P		3DR18	SSKT	x	CE+KD	**84.58**
I		3DR18	SSKT	o	CE+KD	84.37
P+I		3DR18	SSKT	o	CE+KD	84.19
-	H51	3DR18	*ft* (K)	-	CE	56.64
P		3DR18	SSKT	o	CE+KD	**57.82**
I		3DR18	SSKT	o	CE+KD	57.75
P+I		3DR18	SSKT	o	CE+CE	57.29

Table 8. Comparison with other knowledge transfer methods. (s) denote learning from scratch and (f) denote finetuning.

T_t	Model	KD	DML	SSKT (T_s)
C10	R20	91.75±0.24	92.37±0.15	**92.46±0.15** (P+I)
	R32	92.61±0.31	93.26±0.21	**93.38±0.02** (P+I)
C100	R20	68.66±0.24	**69.48±0.05**	68.63±0.12 (I)
	R32	70.5±0.05	**71.9±0.03**	70.94±0.36 (P+I)
S10	R20	77.67±1.41	78.23±1.23	**84.56±0.35** (P+I)
	R32	76.07±0.67	77.14±1.64	**83.68±0.28** (I)
VOC	R18	64.11±0.18	39.89±0.07	**76.42±0.06** (P+I)
	R34	64.57±0.12	39.97±0.16	**77.02±0.02** (P+I)
	R50	62.39±0.6	39.65±0.03	**77.1±0.14** (P+I)
U101	3DR18 (s)	-	13.8	**52.19** (P+I)
	3DR18 (f)	-	83.95	**84.58** (P)
H51	3DR18 (s)	-	3.01	**17.91** (P+I)
	3DR18 (f)	-	56.44	**57.82** (P)

[14], and for DML, training was performed in the same way as [39]. Table 8 shows the evaluation performance according to each knowledge transfer. In the case of 3D-CNN-based action classification, KD was not performed, and the network used for DML used the model with the same conditions as the source and target networks of SSKT. In the case of DML for 3D-CNN, training was performed by replacing ResNet50 with 3D-ResNet50 under the same conditions. In the case of action classification, both learning from scratch and finetuning results were included. In most cases, under similar training conditions, SSKT showed higher generalization performance than other knowledge transfer techniques. At the same time, training in problem domains other than the image classification problem for DML was difficult. In particular, in the case of multi-class classification, both networks were not sufficiently trained. SSKT was able to achieve improved performance in all problem domains with multiple experimental setups.

5. Conclusions

We proposed a simple but powerful knowledge transfer, SSKT, that enables knowledge transfer between heterogeneous networks and datasets. We validated the SSKT by several knowledge transfer scenarios based on CNN. For further investigations of the SSKT, depending on the target task where the knowledge transfer takes place, additional considerations for the transfer module design must be taken into account. At the same time, the following experiments are needed to verify how the performance of the target model varies with the architecture of the source model trained on the same dataset. Variations of the data transformation function f_s for the source networks may be another research avenue worth pursuing. Further, there is a need to perform analyses such as the one approached in [6], to analyze the factors involved when the backpropagation of the auxiliary task loss affects the performance of the target task. We expect that further research will follow as SSKT may provide a different perspective on knowledge transfer.

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *In Proc. of ICML*, 2020. 3

[2] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *In Proc. of NeurIPS*, 2020. 3

[3] Inseop Chung, SeongUk Park, Jangho Kim, and Nojun Kwak. Feature-map-level online adversarial knowledge distillation. *In Proc. of CVPR*, 2020. 2

[4] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. *In Proc. of AISTAT*, 2011. 6

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *In Proc. of CVPR*, 2009. 5

[6] Yunshu Du, Wojciech M. Czarnecki, Siddhant M. Jayakumar, Razvan Pascanu, and Balaji Lakshminarayanan. Adapting auxiliary losses using gradient similarity. *In Proc. of NeurIPSW*, 2019. 3, 8

[7] Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge - a retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015. 6

[8] Tommaso Furlanello, Zachary C. Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. *In Proc. of ICML*, 2018. 2

[9] Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. *In Proc. of ICCV*, 2019. 3

[10] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? *In Proc. of CVPR*, 2018. 7

[11] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *In Proc. of CVPR*, 2020. 3

[12] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. *In Proc. of ICCV*, 2019. 1, 2

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *In Proc. of CVPR*, 2016. 2

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *In Proc. of NIPSW*, 2014. 1, 2, 4, 8

[15] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. *In Proc. of CVPR*, 2017. 7

[16] Inseop Chung Nojun Kwak Jangho Kim, Minsung Hyun. Feature fusion for online mutual knowledge distillation. *In Proc. of ICPR*, 2020. 2

[17] Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetful learning for domain expansion in deep neural networks. *In Proc. of AAAI*, 2018. 3

[18] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *CoRR abs/1705.06950*, 2017. 7

[19] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Technical Report*, 2009. 6

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *In Proc. of NIPS*, 2012. 2

[21] Hilde Kuehne, Hueihan Jhuang, Estibaliz Garrote, Tomaso A. Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. *In Proc. of ICCV*, 2011. 7

[22] Zhizhong Li and Derek Hoiem. Learning without forgetting. *In Proc. of ECCV*, 2016. 3

[23] Shikun Liu, Andrew J. Davison, and Edward Johns. Self-supervised generalisation with meta auxiliary learning. *In Proc. of NeurIPS*, 2019. 3

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *In Proc. of ECCV*, 2016. 1, 2

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *In Proc. of CVPR*, 2015. 1, 2

[26] Rafael Müller, Simon Kornblith, and Geoffrey Hinton. Subclass distillation. *CoRR abs/2002.03936*, 2020. 2

[27] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Imagenet classification with deep convolutional neural networks. *In Proc. of CVPR*, 2019. 2

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *In Proc. of NeurIPS*, 2019. 5, 6

[29] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *In Proc. of NIPS*, 2015. 1, 2

[30] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *In Proc. of ICMR*, 2015. 2

[31] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *In Proc. of CVPR*, 2018. 7

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *In Proc. of ICLR*, 2015. 2

[33] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *CoRR abs/1212.0402*, 2012. 7

[34] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. *In Proc. of ICANN*, 2018. 1

[35] Guile Wu and Shaogang Gong. Peer collaborative learning for online knowledge distillation. *In Proc. of CVPR*, 2020. 2

[36] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *In Proc. of NIPS*, 2014. 1

[37] Amir Zamir, Alexander Sax, William Shen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. *In Proc. of CVPR*, 2018. 1

[38] Amir Zamir, Alexander Sax, Teresa Yeo, Oğuzhan Kar, Nikhil Cheerla, Rohan Suri, Zhangjie Cao, Jitendra Malik, and Leonidas Guibas. Robust learning through cross-task consistency. *In Proc. of CVPR*, 2020. 1

[39] Ying Zhang, Tao Xiang, Timothy M. Hospedales, and Huchuan Lu. Deep mutual learning. *In Proc. of CVPR*, 2018. 2, 8

[40] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, , and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1452–1464, 2018. 5