EXPLICIT CONNECTION DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

One effective way to ease the deployment of deep neural networks on resource constrained devices is Knowledge Distillation (KD), which boosts the accuracy of a low-capacity student model by mimicking the learnt information of a high-capacity teacher (either a single model or a multi-model ensemble). Although great progress has been attained on KD research, existing efforts are primarily invested to design better distillation losses by using soft logits or intermediate feature representations of the teacher as the extra supervision. In this paper, we present Explicit Connection Distillation (ECD), a new KD framework, which addresses the knowledge distillation problem in a novel perspective of bridging dense intermediate feature connections between a student network and its corresponding teacher generated automatically in the training, achieving knowledge transfer goal via direct cross-network layer-to-layer gradients propagation, without need to define complex distillation losses and assume a pre-trained teacher model to be available. ECD has two interdependent modules. In the first module, given a student network, an auxiliary teacher architecture is temporally generated conditioned on strengthening feature representations of basic convolutions of the student network via replacing them with dynamic additive convolutions and keeping the other layers unchanged in structure. The teacher generated in this way guarantees its superior capacity and makes a perfect feature alignment (both in input and output dimensions) to the student at every convolutional layer. In the second module, dense feature connections between the aligned convolutional layers from the student to its auxiliary teacher are introduced, which allows explicit layer-to-layer gradients propagation from the teacher to the student via the merged model training from scratch. Intriguingly, as feature connection direction is one-way, all feature connections together with the auxiliary teacher merely exist during training phase. Compared to existing KD methods, ECD is more friendly to end users thanks to its simplicity. Experiments on popular image classification tasks validate the effectiveness of our method. Code will be made publicly available.

1 INTRODUCTION

Deep Neural Networks (DNNs) have achieved great success in tackling a variety of visual recognition tasks (Krizhevsky et al., 2012; Girshick et al., 2014; Long et al., 2015). Despite the appealing performance, the prevailing DNN models usually have large numbers of parameters, leading to heavy costs of memory and computation. Conventional techniques such as pruning weights from networks (Han et al., 2015; Li et al., 2017) and quantizing networks to use low-bit parameters (Courbariaux et al., 2015; Rastegari et al., 2016; Zhou et al., 2016) have proven to be effective for mitigating this computational burden. More recently, Knowledge Distillation (KD), another promising solution family to get compact yet accurate models, has attracted increasing attention.

The goal of KD is to transfer the learnt information (knowledge) of a high-capacity DNN model or an ensemble of multiple DNN models (teacher) to a low-capacity target DNN model (student), striking better accuracy-efficiency tradeoffs at runtime. There exist numerous KD methods to address the knowledge transfer from the teacher to the student. Many of them use a two-stage training process which begins with training a teacher model and keeping it fixed, and then learns a target student model by forcing it to match the outputted knowledge from the pre-trained teacher model. Various types of knowledge have been explored, such as outputted logits (Ba & Caruana, 2014; Hinton et al., 2015), intermediate feature representations (Romero et al., 2015; Zagoruyko & Komodakis, 2017),

and relational information of model outputs or representations (Park et al., 2019; Tian et al., 2020). Instead of defining new types of knowledge, the other methods adopt one-stage training process, jointly training the student and teacher/peer models using bidirectional knowledge distillation (Zhang et al., 2018; Yao & Sun, 2020) or on-the-fly ensemble distillation (Lan et al., 2018; Anil et al., 2018) or multi-exit distillation (Phuong & Lampert, 2019). Both two-stage and one-stage KD methods as above typically treat the knowledge transfer as an optimization problem of formulating robust distillation loss functions via introducing more informative knowledge supervisions and more effective knowledge matching strategies. However, this usually leads to complex optimization objectives which may need careful tuning of each hyper-parameter (such as the temperature to soften model outputs and the weighting factors to balance different loss terms) to get satisfied performance. Furthermore, in real applications, for two-stage KD methods well pre-trained teacher models are usually not available, and for one-stage KD methods finding good teacher/peer networks may also not be easy, especially for non-skilled users. Because of the aforementioned limitations, most existing KD methods are commercially unattractive to some extent.

To circumvent these limitations, instead of extracting informative knowledge and designing alternative distillation loss functions to fit a desired knowledge transfer optimization objective as existing KD methods, we investigate a new technical perspective in this paper: casting the knowledge distillation problem into designing auxiliary connection paths between the student and teacher networks to enable explicit layer-to-layer gradients distillation from the teacher to the student via training them from scratch simultaneously. We are partially inspired by recent advances on DNN architecture engineering, which show that designing sophisticated feature connection paths such as residual connections (He et al., 2016) and dense connections (Huang et al., 2017) across neighboring layers can allow better information and gradients flow throughout a single network, making the training easy to have significantly improved performance in model accuracy and convergence. We conjecture this simple principle would also be crucial to open the door to develop a totally new knowledge distillation framework if we can merge the student and teacher into a single network temporarily during training phase, and can also separate them easily after training. To explore this hypothesis, we present Explicit Connection Distillation (ECD), a very simple knowledge distillation framework. Specifically, we decompose the design of ECD into two interdependent modules, namely auxiliary teacher generation and dense feature connection distillation. Recent works (Liu et al., 2020; Yue et al., 2020) show that searching a good alignment of structural feature channels between the student and teacher networks can bring improved knowledge distillation performance under the premise that a pre-trained teacher model is available. Regarding the first module of ECD, we hope the generated teacher can make a perfect structure alignment (both in input and output feature dimensions of every convolutional layer) to the student network in an easier manner (no need of time-consuming searching procedure as used in (Liu et al., 2020; Yue et al., 2020)) while can attain superior model capacity. To this goal, we retain all structural units of the student network in constructing the teacher architecture, except for replacing original convolutions by dynamic additive convolutions (Yang et al., 2019) which have proven to be very effective for enhancing model capacity in network architecture engineering research. Regarding the second module of ECD, we hope knowledge distillation can be realized through the explicit layer-to-layer flows of gradients from the teacher to the student instead of the conventional mimicking procedure. To this goal, we add dense feature connections from the convolutional layers of the student to those aligned layers of the auxiliary teacher, and train the merged model from scratch. After training, all feature connections can be naturally removed as their connection direction is only from the student to its auxiliary teacher which also exists merely in training phase. Note that such a training procedure can be readily augmented by an online ensemble distillation at the heads of two networks, achieving further improved performance without introducing the additional cost.

Beyond the common wisdom of knowledge distillation that requires the knowledge mimicking process between the student and teacher networks, our ECD sheds new insight: by considering knowledge distillation from a novel student-to-teacher merging, co-training and splitting viewpoint, direct feature connections from the student to its well-aligned auxiliary teacher (generated from the student in an automatic manner) can also achieve competitive performance to improve low-capacity student models, as validated by extensive experiments on CIFAR-100 and ImageNet datasets.

2 Related Work

In this section, we make a brief summary of existing knowledge distillation works.

Two-stage KD methods. The idea of training a compact model to mimic the functions learnt by a larger ensemble of models is firstly proposed in (Bucilă et al., 2006). Ba & Caruana (2014) extends this idea, showing that shallower yet wider neural networks can also approximate the functions previously learnt by deep ones. Hinton et al. (2015) presents the famous Knowledge Distillation (KD) method, which adopts a teacher-student framework for transferring learnt soft knowledge from a high-capacity teacher to a low-capacity target student network. In this framework, the teacher is pre-trained and fixed, and then its soft logits on the training data are used as the extra supervision to guide the training of the student besides the ground truth labels. FitNets (Romero et al., 2015) shows the intermediate feature representations learnt by the teacher can be used as the complementary knowledge to soft logits, enhancing knowledge distillation performance to some extent. Zagoruyko & Komodakis (2017) proposes to use spatial attention activations instead of intermediate feature maps. Recent works further show the relations (Park et al., 2019) and higher order dependencies (Tian et al., 2020) of learnt logits or intermediate feature representations by the teacher capture important structural information, which are more effective knowledge to augment the knowledge distillation process. Following the teacher-student framework as above, many other works such as (Yim et al., 2017; Lee et al., 2018; Kim et al., 2018) attempt to further improve knowledge representations and matching losses.

One-stage KD methods. Unlike two-stage KD methods that rely on a pre-trained teacher model and one-way knowledge transfer from the teacher to the student, one-stage KD methods simplifying the knowledge distillation process by training all models simultaneously. Deep mutual learning (Zhang et al., 2018) considers the teacher as a peer of the target student, showing that the outputted logits of the student can be used to assist the training of the teacher by a peer-teaching strategy. In (Yao & Sun, 2020), the authors further show that deep mutual learning can also be boosted by intermediate soft outputs, just like that in two-stage KD methods. Lan et al. (2018) presents an on-the-fly ensemble distillation method in which a stronger ensemble teacher learnt over a multi-branch network is used to enhance the training of every branch. Anil et al. (2018) extends this idea to accelerate large-scale distributed neural network training applications. Other improved variants include but are not limited to (Phuong & Lampert, 2019; Malinin et al., 2020).

Other KD variants. Besides model compression, there also exist a lot of works using knowledge distillation methodology to handle other applications. Self-distillation methods (Furlanello et al., 2018; Bagherinezhad et al., 2015) explore the benefits of KD techniques to improve the model training, assuming only a single network is available. Lopes et al. (2015) and Chen et al. (2019) address data-free knowledge distillation where the original training data are no longer accessible due to safety or privacy concerns. Goldblum et al. (2020) and Chung et al. (2020) extend the idea of KD to study how adversarial robustness of the teacher model can be transferred to the student. Li & Hoiem (2016) and Hou et al. (2018) combine KD with fine-tuning and retrospection to handle life-long learning scenarios. Adapting KD to other tasks such as multi-modal visual recognition and natural language processing is explored in (Garcia et al., 2018; Kim & Rush, 2016).

3 Explicit Connection Distillation

In this section, we describe the formulation of the proposed Explicit Connection Distillation (ECD) framework, and detail how to design and implement its two key components: auxiliary teacher generation and dense feature connection distillation.

3.1 Conventional KD Formulation

For a better understanding of our method, we start with the formulation of conventional KD methods. Let S, T and θ_S, θ_T denote a target student model, its teacher (either a single model or an ensemble of multiple models) and their parameters correspondingly, and let x denote training data, and let $\mathcal{Q}$ be a set of layer location pairs where knowledge mimicking process between two networks is introduced. Regarding one-stage KD methods, where the student and teacher models are trained jointly, the overall objective function to be optimized can be defined as

$$\mathcal{L}_{\text{KD}} = \mathcal{L}_{\text{CE}}(\theta_S, x) + \mathcal{L}_{\text{CE}}(\theta_T, x) + \lambda \sum_{\boldsymbol{q} \in \mathcal{Q}} d(f_S^q(\boldsymbol{x}), f_T^q(\boldsymbol{x})), \tag{1}$$

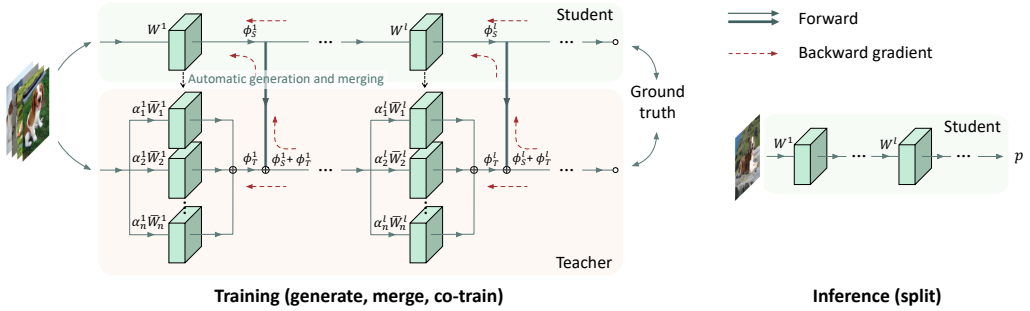

Figure 1: A schematic overview of the training stage and inference stage of our ECD. In the training phase, the teacher network is automatically generated by taking the student as the reference, and then both networks are merged by adding dense feature connections and co-trained from scratch. In the inference phase, in order to obtain the prediction (p), only the student network is needed which is simply split from the teacher.

where $\mathcal{L}_{\text{CE}}(\theta_S, x)$ and $\mathcal{L}_{\text{CE}}(\theta_T, x)$ are the standard cross-entropy loss functions of the student model and the teacher model, respectively. d is the loss for knowledge mimicking, which measures the distance of learnt knowledge $f_S^q(\boldsymbol{x})$, $f_T^q(\boldsymbol{x})$ at all specific layer location pairs between the student and teacher models. As we discussed in introduction section, intermediate feature representations, soft logits, and their relations from learnt models are popularly used as the knowledge. λ is a tunable weighting factor to balance loss terms, which is usually initialized to a relatively large value and decays during training. Simply removing the loss term $\mathcal{L}_{\text{CE}}(\theta_T, x)$ from Equation (1) gets the objective function of two-stage KD methods, as in them the teacher model is pre-trained and fixed.

3.2 Formulation of ECD

The goal of our ECD is to realize knowledge transfer from the teacher to the student without need of the common knowledge mimicking process described above, making KD design as simple as possible. That is, for ECD, the optimization objective is

$$\mathcal{L}_{\text{ECD}} = \mathcal{L}_{\text{CE}}(\theta_S, x) + \mathcal{L}_{\text{CE}}(\theta_T, x). \tag{2}$$

As illustrated in Figure 1, merely given a student network and training data, our ECD achieves such a goal by a smart teacher-generating, student-to-teacher merging, co-training and splitting framework, consisting of two interdependent modules:

- **Auxiliary teacher generation module (in § 3.3)**: A teacher model, which structurally aligns with the given student at any network depth, is generated automatically from the student. Thus there is no longer any need to prepare large teacher architectures.
- **Dense feature connection distillation module (in § 3.4)**: The student model is temporally merged with the teacher by adding explicit layer-to-layer connections to the teacher. The knowledge is transferred to the student during co-training of the merged model. Thus there is no longer any need to design distillation losses and tune λ to balance different losses.

3.3 Auxiliary Teacher Generation

Unlike existing KD methods which usually need to prepare different teacher architectures for their corresponding student networks, in our method, a teacher model is generated automatically from the student by the auxiliary teacher generation module. To guarantee superior capacity of the generated teacher network and make a perfect structure alignment to the student network simultaneously, this module retains all structural units of any given student network in generating auxiliary teacher architecture, but replaces original convolutions by dynamic additive convolutions. This new type of convolution is recently proposed in (Yang et al., 2019), which has proven to be very effective for enhancing model capacity in network architecture engineering research. We extend it to construct the desired teacher which can naturally align with the student structure, and thus is compatible to our overall design goal. Denote convolutional kernels in the student network as $\mathcal{K}_S = \left\{\boldsymbol{W}^l\right\}, l \in$

$1, \cdots, L$, where L is the number of convolutional layers. The generated kernels in the teacher network can be written as $\mathcal{K}_T = \{\sum_{i=1}^{n} \alpha_i^l \bar{\boldsymbol{W}}_i^l\}, l \in 1, \cdots, L$, where for the lth layer and the ith kernel $\bar{\boldsymbol{W}}_i^l$ it has the same shape with $\boldsymbol{W}^l$, and α_i^l is a learnable weight conditioned on the input feature maps, and n is the kernel number (we set it to 8/16 as suggested in (Yang et al., 2019). Thanks to additive property, by replacing normal convolutions with dynamic additive convolutions, both input and output feature dimensions of every convolutional layer between the teacher and student networks are the same completely. You are referred to the block tagged as "teacher" in Figure 1 for a clear understanding of this generation process.

3.4 Dense Feature Connection Distillation

Once the teacher and student networks are perfectly aligned at every convolutional layer, then effective knowledge distillation process of our method is realized by dense feature connections from the student to the teacher which merges two networks temporally and enables co-training from scratch. As illustrated in Figure 1, we add skip connections to bridge both models, formulated as $\widehat{\boldsymbol{\phi}}_T^l = \boldsymbol{\phi}_T^l + \boldsymbol{\phi}_S^l, l \in 1, \cdots, L$, where $\boldsymbol{\phi}_S^l$ and $\boldsymbol{\phi}_T^l$ are the feature maps after the lth convolutional layer of the student and teacher models, respectively, and $\widehat{\boldsymbol{\phi}}_T^l$ is the new feature maps of the teacher after addition. We call this module dense feature connection distillation which is simple and neat, yet can well enhances the learning performance of the student model due to the backward gradient flows from the teacher. After training, all skip connections can be naturally removed as their connection direction is only from the student to its auxiliary teacher which also exists only in training phase.

4 Experiments

In this section, we evaluate our method on CIFAR-100 and ImageNet classification datasets, and compare the performance against existing knowledge distillation methods. For fair comparisons, we use the public codes of different KD methods, and adopt the same training and data preprocessing settings throughout the experiments. All experiments are implemented with PyTorch. To our ECD, we add Conv1×1 on each connection path as we find it slightly improves the distillation performance (see ablative study part). Based on ECD, we also apply another ensemble distillation on the outputted logits from the two heads of the merged model to further boost the performance of each head, which is named as ECD*. Full implementation details are referred to Appendix.

4.1 Experiments on CIFAR-100

Dataset. CIFAR-100 (Krizhevsky & Hinton, 2009), containing 50,000 training images and 10,000 test images with 100 classes, is the most popular classification dataset for evaluating the performance of knowledge distillation methods.

Implementation. We performed experiments on prevalent ResNets (He et al., 2016) of different depths and WRNs (Zagoruyko & Komodakis, 2016) of different widths with the typical training settings (see Appendix for details). For auxiliary teacher models, we replace the standard convolutions in BasicBlock or Bottleneck of the student with dynamic additive convolutions (with $n = 16$ following (Yang et al., 2019)). For the experiments of each setting, we run each method 3 times and report top-1 "mean(std)" accuracies.

Main results. In Table 1, we provide average results for baselines, teachers and our method ECD. Note that results of baselines are slightly higher than those reported in the original papers. For our ECD experiments, we uniformly apply feature connections in the first two stages of ResNets or WRNs (we also provide ablative experiments to analyze where to add connection paths). Regarding to ResNet backbones, we observe $1.2\% \sim 2.0\%$ absolute accuracy gains for ECD, and $1.5\% \sim 2.6\%$ for ECD*. This shows that our design is effective for BasicBlock and Bottleneck structures with different depths. Besides, on WRN backbones, ECD and ECD* outperform baselines with $0.6\% \sim 1.1\%$ and $0.7\% \sim 1.4\%$ margins, respectively. In summary, the proposed ECD noticeably improves the performance of each student network, and ECD* further provides additional improvements. In most cases, the students trained by our method show even better performance than the corresponding teachers. These results verify the effectiveness of our proposed method. The performance is not that strong for WRN-40-2, as this architecture has already widened the network channels.

Table 1: Main experimental results on the CIFAR-100 dataset. † means using learning rate warm-up to smooth the training process. We report top-1 "mean(std)" accuracies (%) over 3 runs.

Model	Baseline	Teacher	ECD	Gain	ECD*	Gain*
ResNet20	68.78 (0.22)	71.05 (0.35)	70.75 (0.29)	1.97	71.07 (0.13)	2.29
ResNet32	70.80 (0.15)	72.78 (0.44)	72.40 (0.14)	1.60	73.09 (0.14)	2.29
ResNet44	71.88 (0.13)	73.48 (0.48)	73.53 (0.16)	1.65	74.08 (0.51)	2.20
ResNet56	72.29 (0.17)	73.79 (0.51)	73.94 (0.08)	1.65	74.88 (0.05)	2.59
ResNet110	73.15 (0.45)	75.32 (0.64)	75.11 (0.14)	1.96	75.47 (0.56)	2.32
ResNet164	76.64 (0.61)	78.78 (0.73)	77.94 (0.39)	1.30	78.34 (0.79)	1.70
ResNet110†	74.41 (0.09)	75.68 (0.21)	75.59 (0.10)	1.18	76.46 (0.20)	2.05
ResNet164†	77.15 (0.10)	78.47 (0.11)	78.33 (0.12)	1.18	78.64 (0.19)	1.49
WRN-40-1	71.44 (0.14)	72.48 (0.23)	72.55 (0.17)	1.11	72.83 (0.05)	1.39
WRN-40-2	75.96 (0.12)	77.23 (0.19)	76.55 (0.15)	0.59	76.64 (0.07)	0.68

Table 2: Results comparison with state-of-the-art two-stage and one-stage KD methods. See Appendix for specific settings of these methods. We report top-1 "mean(std)" accuracies (%) over 3 runs.

Two-stage			
Student	68.78 (0.22) (ResNet20)	70.80 (0.15) (ResNet32)	71.44 (0.14) (WRN-40-1)
Teacher	73.15 (ResNet110)	73.15 (ResNet110)	73.23 (WRN-40-2)
KD	70.42 (0.16)	72.08 (0.18)	73.23 (0.23)
FitNet	70.36 (0.16)	72.09 (0.36)	71.98 (0.19)
AT	70.22 (0.03)	71.64 (0.32)	71.74 (0.10)
FSP	69.95 (0.11)	71.69 (0.06)	71.61 (0.03)
SP	70.35 (0.07)	72.11 (0.32)	72.47 (0.30)
VID	70.14 (0.11)	71.64 (0.30)	71.68 (0.33)
PKT	70.06 (0.08)	71.66 (0.05)	72.51 (0.18)
FT	71.14 (0.37)	72.82 (0.10)	71.65 (0.09)
NST	70.12 (0.06)	71.76 (0.36)	71.31 (0.40)
RKD	70.24 (0.09)	71.93 (0.19)	71.63 (0.21)
CC	70.37 (0.19)	71.67 (0.18)	71.44 (0.09)
CRD	70.82 (0.12)	72.84 (0.65)	72.73 (0.18)

One-stage			
Student	68.78 (0.22) (ResNet20)	70.80 (0.15) (ResNet32)	71.44 (0.14) (WRN-40-1)
Teacher	74.43 (0.21) (ResNet110)	75.27 (0.11) (ResNet110)	73.99 (0.12) (WRN-40-2)
DML	70.47 (0.25)	72.57 (0.23)	72.44 (0.21)
Teacher	72.55 (0.18) (Ensemble)	74.82 (0.08) (Ensemble)	N/A (Ensemble)
ONE	70.55 (0.07)	72.29 (0.12)	N/A
Teacher	71.05 (0.35) (Generated)	72.78 (0.44) (Generated)	72.48 (0.23) (Generated)
ECD	70.75 (0.18)	72.40 (0.14)	72.55 (0.17)
ECD*	71.07 (0.12)	73.09 (0.14)	72.83 (0.05)

Comparison with SOTAs. In Table 2, we compare our method with state-of-the-art knowledge distillation methods under the same training settings with ours. For ResNet20 and WRN-40-1, the automatically generated teachers of our method reach 71.05% and 72.48% accuracies (as previously shown in Table 1), which are much lower than the teachers used for other methods (73.15% and 76.23%), yet our method is able to achieve very competitive performance. For ResNet32, under the comparable capacity of the teach model (ours 72.78% vs. others 73.15%), our ECD* outperforms all the other methods with obvious margins.

4.2 Experiments on ImageNet

Dataset. We additionally performed experiments on the ImageNet dataset (ILSVRC12) (Russakovsky et al.), which is known as the most challenging image classification dataset containing about 1.2 million training images and 50 thousand validation images, and each image belongs to one of 1000 categories.

Implementation. Experiments are also conducted on prevalent ResNets. For auxiliary teacher models, we replace the standard convolutions in BasicBlock or Bottleneck of the student with dynamic additive convolutions (with $n = 8$ following (Yang et al., 2019)). We chose ResNet18 and Resnet50-half models, because they are often evaluated in knowledge distillation and the auxiliary teacher model and the student model have a suitable performance gap. Implementation details are available in Appendix.

Table 3: Results on the Imagnet dataset. We report top-1 accuracies (%).

Model	Baseline	Teacher	ECD	Gain	ECD*	Gain*
ResNet18	69.53	72.86	70.33	0.80	70.78	1.25
ResNet50 (0.5×)	71.41	75.27	72.19	0.78	72.48	1.07

Results. Table 3 reports the performance of our method on ImageNet. ECD and ECD* improve baseline models by 0.8% and over 1.0% absolute gains in top-1 accuracy respectively, which support the benefit of our method on the large-scale dataset.

4.3 Ablation Study

In this section, we isolate the influence of each element of our method, as well as compare with possible variants. All experiments are conducted on CIFAR-100 dataset. We use ECD but not ECD*, and do not use learning rate warm-up throughout all experiments for a better ablation study. For the experiments of each setting, we run our method 3 times and report top-1 "mean(std)" accuracies.

Location of connections. We explore the influence of the locations to add dense feature connections. This is very important because the semantics and robustness of different locations are different. We consider different settings by adding our dense feature connections to at most three blocks (including the block Conv2_x, Conv3_x, and Conv4_x, denoted as C2, C3 and C4 respectively) on CIFAR100. Detailed results are shown in Table 4. We observe that adding connections in both Conv2_x and Conv3_x brings the best performance improvement, followed by adding connections in Conv2_x alone. These results indicate that adding connections in shallow layers such as Conv2_x and Conv3_x can migrate information well, while Conv4_x extracts higher-level semantics and thus may be less robust, which limits the effect of connection supervision. To some extent, this is consistent with multi-task learning and multi-branch network designs (Lan et al., 2018; Anil et al., 2018) where the shallow network is shared and the higher level network is divided into separate branches.

Table 4: Comparison of ECD with dense feature connections added to different blocks. In the table, C2, C3 and C4 refer to Conv2_x, Conv3_x and Conv4_x, respectively. We report top-1 "mean(std)" accuracies (%) over 3 runs.

Connection	ResNet20	ResNet110	WRN-40-1
Baseline	68.78 (0.22)	73.15 (0.33)	71.44 (0.14)
C2	70.36 (0.03)	75.00 (0.35)	72.00 (0.39)
C3	70.12 (0.12)	74.41 (0.12)	72.14 (0.38)
C4	69.75 (0.12)	73.85 (0.60)	71.32 (0.37)
C2+C3	70.75 (0.29)	75.11 (0.14)	72.55 (0.17)
C2+C3+C4	69.90 (0.10)	73.75 (0.21)	70.93 (0.18)

Comparison with using feature loss. Recall that the distance of intermediate features between the student and teacher models is popularly used in existing KD methods to formulate feature loss (Romero et al., 2015; Zagoruyko & Komodakis, 2017). Instead of using feature loss, we simply use dense feature connections. In Table 5, we compare the performance of feature loss and dense feature connections under the same generated teacher. The results show that in our proposed framework, using dense connections is superior to using feature loss.

Impact of connection transform. In Table 6, we compare the performance when using direct connection and adding a Conv1×1 layer to each connection for improved alignment in feature semantics. We observe that adding Conv1×1 achieves slightly higher results than direct connection. That means direct connections without any transform like Conv1×1 still get promising knowledge distillation performance. This is mainly due to our first module for auxiliary teacher generation. By replacing normal convolutions with dynamic additive convolutions, the teacher is well aligned to the student in network depth, and the input and output feature dimensions of every convolutional layer between the teacher and student networks are the same completely.

Table 5: Results comparison of our ECD and using feature losses. "stage-wise" means feature supervision after each stage, which is consistent with many typical KD methods, and the distillation loss factor here is set to 10^{-2}; "block-wise" refers to supervision on the same blocks where we add connections in our ECD, and the distillation loss factor here is set to 10^{-4}. We report top-1 "mean(std)" accuracies (%) over 3 runs.

Model	ResNet20	ResNet110	WRN-40-1
Baseline	68.78 (0.22)	73.15 (0.45)	71.44 (0.14)
l2-loss (stage wise)	70.42 (0.16)	74.64 (0.34)	72.10 (0.22)
l2-loss (block wise)	70.12 (0.10)	74.32 (0.26)	71.64 (0.15)
ECD	70.75 (0.18)	75.11 (0.12)	72.55 (0.17)

Table 6: Comparison with adding Conv1×1 on each connection path. We report top-1 "mean(std)" accuracies (%) over 3 runs.

Transfor	ResNet20	ResNet110	WRN-40-1
Baseline	68.78 (0.22)	73.15 (0.45)	71.44 (0.14)
None	70.42 (0.17)	75.07 (0.21)	72.13 (0.06)
Conv1×1	70.75 (0.18)	75.11 (0.12)	72.55 (0.17)

Auxiliary teacher structure design. As described in § 3.3, the teacher is generated automatically through replacing the convolutional kernels in the student by a linear combination of several kernels. As the knowledge is transferred from the generated teacher to the student, the capability of the teacher could affect the improvements. In Table 7, we compare our default setting (DAConv) with two other attention based alternatives SE (Hu et al., 2018) and CBAM (Woo et al., 2018). We find that under our proposed framework, generating teacher using attention modules such as SE and CBAM can also bring noticeable improvements over the student networks. Comparatively, our default setting achieves higher performance than these two alternatives.

Table 7: Comparison of auxiliary teacher generation with different methods. DAConv refers to our default setting. We report top-1 "mean(std)" accuracies (%) over 3 runs.

Hyper-Conv	Method	ResNet20	ResNet110	WRN-40-1
	Baseline	68.78 (0.22)	73.15 (0.45)	71.44 (0.14)
SE	Teacher	70.23 (0.33)	74.98 (0.31)	72.06 (0.23)
	ECD	70.05 (0.17)	74.96 (0.79)	72.05 (0.17)
CBAM	Teacher	70.45 (0.13)	75.50 (0.11)	72.26 (0.16)
	ECD	70.08 (0.13)	74.94 (0.59)	72.26 (0.07)
DAConv	Teacher	71.05 (0.35)	75.32 (0.64)	72.48 (0.23)
	ECD	70.75 (0.29)	75.11 (0.14)	72.55 (0.17)

In Appendix, we provide additional experiments, *e.g.* combining our method with labeling smoothing. More comparisons and discussions can also be found in Appendix.

5 Conclusion

In this paper, we present Explicit Connection Distillation (ECD), a new knowledge distillation framework, which addresses the knowledge distillation problem in a novel perspective of bridging dense intermediate feature connections between a student network and its corresponding teacher which is generated automatically in the training. ECD achieves knowledge transfer goal via direct cross-network layer-to-layer gradients propagation, without need to define complex distillation losses and assume a pre-trained teacher model to be available. We hope our work can inspire the future research on knowledge distillation designs.

REFERENCES

Sungsoo Ahn, Shell Xu Hu, Andreas Damianou, Neil D Lawrence, and Zhenwen Dai. Variational information distillation for knowledge transfer. In *CVPR*, 2019.

Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. Large scale distributed neural network training through online distillation. In *ICLR*, 2018.

Lei Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *NIPS*, 2014.

Hessam Bagherinezhad, Maxwell Horton, Mohammad Rastegari, and Ali Farhadi. Label refinery: Improving imagenet classification through label progression. *arXiv preprint arXiv:1805.02641*, 2015.

Cristian Bucilă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *KDD*, 2006.

Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Data-free learning of student networks. In *ICCV*, 2019.

Inseop Chung, SeongUk Park, Jangho Kim, and Nojun Kwak. Feature-map-level online adversarial knowledge distillation. In *ICML*, 2020.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NIPS*, 2015.

Tommaso Furlanello, Zachary C. Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born-again neural networks. In *ICML*, 2018.

Nuno C Garcia, Pietro Morerio, and Vittorio Murino. Modality distillation with multiple stream networks for action recognition. In *ECCV*, 2018.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

Micah Goldblum, Liam Fowl, Soheil Feizi, and Tom Goldstein. Adversarially robust distillation. In *AAAI*, 2020.

Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. In *NIPS*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

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

Saihui Hou, Xinyu Pan, Chen Chang Loy, Zilei Wang, and Dahua Lin. Lifelong learning via progressive distillation and retrospection. In *ECCV*, 2018.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

Zehao Huang and Naiyan Wang. Like what you like: Knowledge distill via neuron selectivity transfer. *arXiv preprint arXiv:1707.01219*, 2017.

Jangho Kim, SeoungUk Park, and Nojun Kwak. Paraphrasing complex network: Network compression via factor transfer. In *NeurIPS*, 2018.

Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. In *EMNLP*, 2016.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Tech Report*, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Xu Lan, Xiatian Zhu, and Shaogang Gong. Knowledge distillation by on-the-fly native ensemble. In *NeurIPS*, 2018.

Seung Hyun Lee, Ha Dae Kim, and Byung Cheol Song. Self-supervised knowledge distillation using singular value decomposition. In *NeurIPS*, 2018.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *ICLR*, 2017.

Zhizhong Li and Derek Hoiem. Learning without forgetting. In *ECCV*, 2016.

Yu Liu, Xuhui Jia, Mingxing Tan, Raviteja Vemulapalli, Yukun Zhu, Bradley Green, and Xiaogang Wang. Search to distill: Pearls are everywhere but not the eyes. In *CVPR*, 2020.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

Raphael Gontijo Lopes, Stefano Fenu, and Thad Starner. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535*, 2015.

Andrey Malinin, Bruno Mlodozeniec, and Mark Gales. Ensemble distribution distillation. In *ICLR*, 2020.

Wonpyo Park, Yan Lu, Minsu Cho, and Dongju Kim. Relational knowledge distillation. In *CVPR*, 2019.

Nikolaos Passalis and Anastasios Tefas. Learning deep representations with probabilistic knowledge transfer. In *ECCV*, 2018.

Baoyun Peng, Xiao Jin, Jiaheng Liu, Shunfeng Zhou, Yichao Wu, Yu Liu, Dong-sheng Li, and Zhaoning Zhang. Correlation congruence for knowledge distillation. In *ICCV*, 2019.

Mary Phuong and Christoph H Lampert. Distillation-based training for multi-exit architectures. In *ICCV*, 2019.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Joseph. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *ICLR*, 2020.

Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *ICCV*, 2019.

Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *ECCV*, 2018.

Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. In *NeurIPS*, 2019.

Anbang Yao and Dawei Sun. Knowledge transfer via dense cross-layer mutual-distillation. In *ECCV*, 2020.

Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *CVPR*, 2017.

Kaiyu Yue, Jiangfan Deng, and Feng Zhou. Matching guided distillation. In *ECCV*, 2020.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017.

Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *CVPR*, 2018.

Shuchang Zhou, Wu Yuxin, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

A APPENDIX

A.1 MAIN EXPERIMENTAL SETTINGS

In this section, we provide detailed settings of the experiments conducted on the CIFAR-100 and ImageNet datasets.

A.1.1 DETAILED EXPERIMENTAL SETTINGS ON CIFAR-100

The experiment was carried out on the CIFAR-100 dataset without additional data augmentation. The network is trained for 200 epochs, the batch size is 128, the weight decay is 5×10^{-4}, and the optimizer is SGD. For ResNets, we set the initial learning rate to 0.1, which decays by 0.1 at epochs 100 and 150. For WRNs, we do not adopt dropout, and the initial learning rate is set to 0.1, which decays by 0.1 at epochs 60, 120, and 160. The auxiliary teacher is expanded from 3×3 convolutional kernels (in the block of the original network) to dynamic additive convolutional kernels, and the number of multiplexing weight is 16. The Conv2_x and Conv3_x blocks of models have dense feature connections as shown in Table 10. Features after 3×3 convolution and after the block will be connected to the same position on the teacher network as shown in Figure 2. The features of the connected parts can be aligned better with the channel semantically with a little gain brought by 1×1 convolutions. For ECD*, a learnable ensemble is applied on both outputted logits of the student and the teacher.

A.1.2 DETAILED EXPERIMENTAL SETTINGS ON IMAGENET

In the ImageNet experiments, the student model is trained with 100 training epochs, which is common setting[1]. The batch size is set to 256 and the multi-step learning rate is initialized to 0.1, which decays by 0.1 at 30, 60, and 90 epochs. The auxiliary teacher has the same ResNet structure except that the ordinary convolution is replaced with the dynamic additive convolution with the multiplexing weight 8. In the training process, the features of Conv2_x, Conv3_x and Conv4_x blocks of the student network are connected to the alignment positions of the auxiliary teacher enhanced by 1×1 convolution layers and BN layers. The output logits of the two output to supervise the training of the two branch heads. In the experiment, we chose ResNet18 and Resnet50-half as shown in Table 11 to verify the effectiveness of our method.

A.2 COMPARISON EXPERIMENTS ON CIFAR-100

In order to make a fair comparison with existing knowledge distillation methods, we refer to the original settings and CRD code zoo[2] to implement various knowledge methods. The setting details, including settings of the factor (λ) in Equation (1), are shown in Table 8. We adopt the same standard training settings as our CIAFR-100 experiments in Table 1.

[1] https://github.com/pytorch/examples/tree/master/imagenet
[2] https://github.com/HobbitLong/RepDistiller

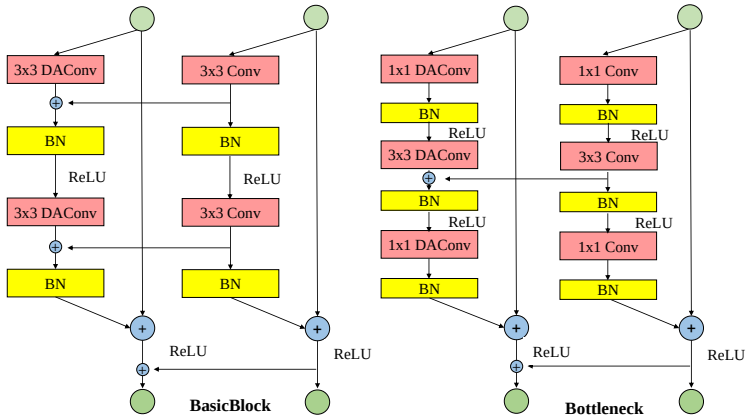

Figure 2: The detailed structure diagram of dense feature connections in BasicBlock and Bottleneck of ResNets, where DAConv refers to the dynamic additive convolution.

Table 8: Brief implementation details of KD methods and settings of the distillation loss factor (λ).

Method	factor (λ)	Brief notes of Implementation details.
KD (Hinton et al., 2015)	0.1	The temperature factor T is 4.
FitNet (Romero et al., 2015)	0.1	One stage without hint layer.
AT (Zagoruyko & Komodakis, 2017)	1000	The sum of absolute values with power p=2 is used as the attention.
FSP (Yim et al., 2017)	1	
SP (Tung & Mori, 2019)	3000	
VID (Ahn et al., 2019)	1	Hidden channel number is the same as output channel, and remove BN in μ.
PKT (Passalis & Tefas, 2018)	1000	
FT (Kim et al., 2018)	200	
NST (Huang & Wang, 2017)	10	The polynomial kernel d is 2 and c is 0.
RKD (Park et al., 2019)	(25,50)	The distance is 50 and the angle is 25.
CC (Peng et al., 2019)	100	
CRD (Tian et al., 2020)	0.8	
DML (Zhang et al., 2018)	1	
ONE (Lan et al., 2018)	1	Ensemble distillation of three branch.

A.3 More discussions about ECD

ECD & label smoothing. Label smoothing is an effective regularization method. Noise is added through soften one-hot labels, which reduces the weight of the real sample label category when calculating the loss function, and suppresses over-fitting. ECD can be combined with label smoothing to provide additional improvements as well.

Table 9: Experiment results of applying Label Smoothing (LS) to ECD.

Method	ResNet20	ResNet110	WRN-40-1
Baseline	68.78(0.22)	73.15(0.45)	71.44(0.14)
Label smoothing	69.89(0.16)	74.59(0.31)	71.56(0.08)
ECD	70.75(0.29)	75.11(0.14)	72.55(0.17)
ECD + LS	71.03(0.08)	75.13(0.43)	72.66(0.07)

Table 10: Details of the convolutional blocks of the ResNet110, ResNet164 and WRN-40-2 backbones evaluated on the CIFAR-100 dataset.

layer name	output size	ResNet110	ResNet164	WRN-40-2
conv1	32×32	3×3, 16	3×3, 16	3×3, 16
conv2_x	32×32	$\begin{bmatrix} 3\times3,\ 16 \\ 3\times3,\ 16 \end{bmatrix}\times 18$	$\begin{bmatrix} 1\times1,\ 16 \\ 3\times3,\ 16 \\ 1\times1,\ 64 \end{bmatrix}\times 18$	$\begin{bmatrix} 3\times3,\ 32 \\ 3\times3,\ 32 \end{bmatrix}\times 6$
conv3_x	16×16	$\begin{bmatrix} 3\times3,\ 32 \\ 3\times3,\ 32 \end{bmatrix}\times 18$	$\begin{bmatrix} 1\times1,\ 32 \\ 3\times3,\ 32 \\ 1\times1,\ 128 \end{bmatrix}\times 18$	$\begin{bmatrix} 3\times3,\ 64 \\ 3\times3,\ 64 \end{bmatrix}\times 6$
conv4_x	8×8	$\begin{bmatrix} 3\times3,\ 64 \\ 3\times3,\ 64 \end{bmatrix}\times 18$	$\begin{bmatrix} 1\times1,\ 64 \\ 3\times3,\ 64 \\ 1\times1,\ 256 \end{bmatrix}\times 18$	$\begin{bmatrix} 3\times3,\ 128 \\ 3\times3,\ 128 \end{bmatrix}\times 6$
classifier	1×1	average pool, 100-d fc, softmax		

Table 11: Details of the convolutional blocks of the ResNet backbones evaluated on the ImageNet dataset.

layer name	output size	ResNet18	ResNet50 (0.5x)
conv1	112×112	7×7, 64, stride	7×7, 32, stride
		3×3 max pool, stride	
conv2_x	56×56	$\begin{bmatrix} 3\times3,\ 64 \\ 3\times3,\ 64 \end{bmatrix}\times 2$	$\begin{bmatrix} 1\times1,\ 32 \\ 3\times3,\ 32 \\ 1\times1,\ 128 \end{bmatrix}\times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3,\ 128 \\ 3\times3,\ 128 \end{bmatrix}\times 2$	$\begin{bmatrix} 1\times1,\ 64 \\ 3\times3,\ 64 \\ 1\times1,\ 256 \end{bmatrix}\times 4$
conv4_x	16×16	$\begin{bmatrix} 3\times3,\ 256 \\ 3\times3,\ 256 \end{bmatrix}\times 2$	$\begin{bmatrix} 1\times1,\ 128 \\ 3\times3,\ 128 \\ 1\times1,\ 512 \end{bmatrix}\times 6$
conv5_x	8×8	$\begin{bmatrix} 3\times3,\ 512 \\ 3\times3,\ 512 \end{bmatrix}\times 2$	$\begin{bmatrix} 1\times1,\ 256 \\ 3\times3,\ 256 \\ 1\times1,\ 2048 \end{bmatrix}\times 3$
classifier	1×1	average pool, 1000-d fc, softmax	