KDExplainer: A Task-oriented Attention Model for Explaining Knowledge Distillation

Mengqi Xue[1] , **Jie Song**[1] , **Xinchao Wang**[2] , **Ying Chen**[1] , **Xingen Wang**[1*] , **Mingli Song**[1]
[1]Zhejiang University
[2]National University of Singapore
{mqxue, sjie, lynesychen, newroot, brooksong}@zju.edu.cn, xinchao@nus.edu.sg

Abstract

Knowledge distillation (KD) has recently emerged as an efficacious scheme for learning compact deep neural networks (DNNs). Despite the promising results achieved, the rationale that interprets the behavior of KD has yet remained largely understudied. In this paper, we introduce a novel task-oriented attention model, termed as KDExplainer, to shed light on the working mechanism underlying the vanilla KD. At the heart of KDExplainer is a Hierarchical Mixture of Experts (HME), in which a multi-class classification is reformulated as a multi-task binary one. Through distilling knowledge from a free-form pre-trained DNN to KDExplainer, we observe that KD implicitly modulates the knowledge conflicts between different subtasks, and in reality has much more to offer than label smoothing. Based on such findings, we further introduce a portable tool, dubbed as virtual attention module (VAM), that can be seamlessly integrated with various DNNs to enhance their performance under KD. Experimental results demonstrate that with a negligible additional cost, student models equipped with VAM consistently outperform their non-VAM counterparts across different benchmarks. Furthermore, when combined with other KD methods, VAM remains competent in promoting results, even though it is only motivated by vanilla KD. The code is available at *https://github.com/zju-vipa/KDExplainer*.

1 Introduction

Recent progress in deep neural networks (DNNs) has significantly benefited many if not all tasks in artificial intelligence, ranging from computer vision to natural language processing. The encouraging results, however, come with an enormous cost: state-of-the-art DNNs have been scaled up to hundreds and even thousands of layers with millions of parameters, making it demanding to train and deploy even on GPU clusters, let alone on edge devices like mobile phones.

Many research efforts have thus been made to craft lightweight DNNs applicable to in-the-wild scenarios. Representative schemes include weight pruning [Han *et al.*, 2015], model quantization [Jacob *et al.*, 2018], and knowledge distillation (KD) [Hinton *et al.*, 2015], among which KD has recently emerged as one of the most flourishing topics in the field. The goal of KD is to extract knowledge from a well-behaved but cumbersome teacher model, often known as *dark knowledge*, and to learn a compact student model capable to handle the task of the teacher but with fewer parameters. Since the pioneering work of [Hinton *et al.*, 2015], a variety of dark knowledge has been explored, including *hint* [Romero *et al.*, 2015], *attention* [Zagoruyko and Komodakis, 2017], and *instance relationship* [Liu *et al.*, 2019].

Despite the above forms of dark knowledge showcase promising results, the naive *soft target* [Hinton *et al.*, 2015] is found still among the most competitive ones [Tian *et al.*, 2020]. Nevertheless, few attempts have been dedicated to explaining the rationale of soft targets on the student learning. A common belief is that soft labels reveal richer information like category similarities than the widely-used one-hot vector, so that the student obtains more supervision signals for learning. Recently, the work [Yuan *et al.*, 2020] argues that the soft target is intrinsically a type of learned label smoothing [Szegedy *et al.*, 2016] regularization. [Furlanello *et al.*, 2018] conjectures that soft targets resemble importance-weighting, where weights correspond to confidences of the teachers in the correct prediction. [Tang *et al.*, 2020], on the other hand, dissects the effects of soft targets into three main factors: label smoothing, importance weighting, and category similarities. Albeit the inspiring insights provided by these works, the underlying mechanism of how category similarities regularize the student model learning has remained largely under-studied to date.

In this paper, we take a closer look at the role of the soft label through the lens of a novel attention model, which we term as KDExplainer. KDExplainer is an interpretation-friendly student model, which distinguishes itself from conventional student DNNs from two main aspects, as illustrated in Figure 1. First, KDExplainer takes the form of a Hierarchical Mixture of Experts (HME), where each expert is expected to specialize in specific subtasks and learn adaptive features. Second, KDExplainer casts the multi-class classification task as a multi-task problem, in which each task is formulated as a

*Corresponding author

binary classification problem. Such a design explicitly shapes KDExplainer as a neural tree, for which the inherent working mechanism is well understood, in the aim to interpret KD.

We then carry out KD from a free-form pre-trained DNN to the dedicated KDExplainer, through which process the rationale of KD is highlighted thanks to the interpretable essence of the HME architecture. Interestingly, we find that the KD objective promotes lower entropy of the attention distribution, indicating that soft labels, in reality, encourage specialization of different experts and hence play a role in modulating the knowledge conflicts for solving different subtasks. To further understand the connection and difference between KD and label smoothing, we train a KDExplainer using label smoothing, and discover that the derived attention distribution exhibits no significant differences with those by vanilla training without KD. This phenomenon marks that soft labels indeed have more to offer, including feature specialization, than label smoothing.

Inspired by these observations, we further introduce a portable component, termed as virtual attention module (VAM), to enhance the performance of conventional DNNs under KD. The key idea of VAM is to coordinate the knowledge conflicts for discriminating different categories, achieved via lowering the entropy of the attention distribution. VAM can be readily integrated with existing DNNs while bringing negligible additional computation cost. Moreover, since VAM is naturally orthogonal to KD, it can be seamlessly combined with various KD schemes, rendering it a handy module to facilitate KD.

Our contributions are therefore summarized as follows.

- We propose a novel attention model of interpretable nature, KDExplainer, to understand the role of soft labels in training the student.
- Through KDExplainer, we observe that soft labels implicitly modulate the knowledge conflicts between different subtasks by promoting feature specialization, and offer more regularization than only label smoothing.
- Understanding the KD rationale via KDExplainer further motivates us to design a portable and compact module, VAM, readily applicable to various DNNs and KDs.

Extensive experimental results across benchmarks demonstrate that VAM not only consistently improves the performance of vanilla KD under various experimental settings, but also can be readily integrated with other state-of-the-art KD schemes to further promote their results.

2 Related Work

Knowledge Distillation. Knowledge distillation has attracted increasing attention thanks to its important role in deploying deep networks to low-capacity edge devices [Yu *et al.*, 2017]. The main idea is leveraging the *dark knowledge* encoded in a bulky teacher to craft a lightweight student model with performance on par with the teacher. Over the last several years, most works devote themselves to the exploration of different forms of the dark knowledge, including soft targets [Hinton *et al.*, 2015; Chen *et al.*, 2020], features [Romero *et al.*, 2015; Ye *et al.*, 2019; Shen *et al.*, 2021; Shen *et al.*, 2019], attention [Zagoruyko and Komodakis, 2017; Yang *et al.*, 2020b; Liu *et al.*, 2021], factors [Kim *et al.*, 2018; Yang *et al.*, 2020a], activation boundary [Byeongho Heo, 2019], and instance relationship [Liu *et al.*, 2019; Park *et al.*, 2019; Tung and Mori, 2019]. By imitating the teacher to behave in a similar way, the student achieves comparable performance even with much fewer parameters.

Attention Mechanism. Inspired by human cognition, attention mechanisms focus on relevant regions of input data to solve the desired task rather than ingesting the entire input. Attention-based neural networks have been broadly adopted in natural language models for machine translation [Bahdanau *et al.*, 2015], image caption generation [Xu *et al.*, 2015], and unsupervised representation learning [Devlin *et al.*, 2019]. Attention mechanisms also achieve great success in vision models [Mnih *et al.*, 2014; Bello *et al.*, 2019; Yang *et al.*, 2020c]. Except the performance boost, attention mechanism also provides an important way to explain the workings of neural models [Li *et al.*, 2016; Wiegreffe and Pinter, 2019; Xu *et al.*, 2015]. Unlike most prior works, our focus here is to utilize an attention mechanism to interpret KD, which has been largely overlooked in previous literature.

3 KDExplainer

3.1 Knowledge Distillation with Soft Targets

Vanilla KD [Hinton *et al.*, 2015] distills the "dark knowledge" from the teacher via aligning the soft targets

$$\mathcal{O}_{\text{KD}} = \alpha \mathcal{L}_{\text{CE}}(p(\boldsymbol{z}^s), \boldsymbol{y}) + (1-\alpha)\mathcal{D}_{\text{KL}}\left(p(\boldsymbol{z}^t;\tau), p(\boldsymbol{z}^s;\tau)\right), \tag{1}$$

where $\boldsymbol{z}^s$ and $\boldsymbol{z}^t$ are respectively the logits from the student and the teacher models. $\boldsymbol{y}$ is its associated one-hot label vector, p denotes the softmax function that produces the category probabilities given the logits, and τ is a non-negative temperature hyperparameter used to smooth the distributions. As for $p_i(\boldsymbol{z};\tau)$, we have

$$p_i(\boldsymbol{z};\tau) = \frac{\exp(z_i/\tau)}{\sum_j \exp(z_j/\tau)}. \tag{2}$$

Also, $\mathcal{L}_{\text{CE}}$ is the conventional cross-entropy loss, and $\mathcal{D}_{\text{KL}}$ is the Kullback-Leibler divergence between the categorial distributions predicted from the teacher and the student models. α is a hyperparameter to trade off the two objective terms.

3.2 The Proposed KDExplainer

We propose the KDExplainer, a task-oriented attention model, as the student model to uncover the rationale underlying KD. KDExplainer makes two main modifications based on exiting popular DNNs: (1) dissecting the effects of class similarity in soft targets, KDExplainer reformulates the multi-class classification problem as an ensemble of multiple binary classification problems; (2) KDExplainer remodels the student model as a Hierarchical Mixture of Experts (HME) and introduces a task-oriented attention mechanism as the gating function to make the student model more friendly for human interpretation. The overview of the proposed KDExplainer is shown in Figure 1. KDExplainer is designed based

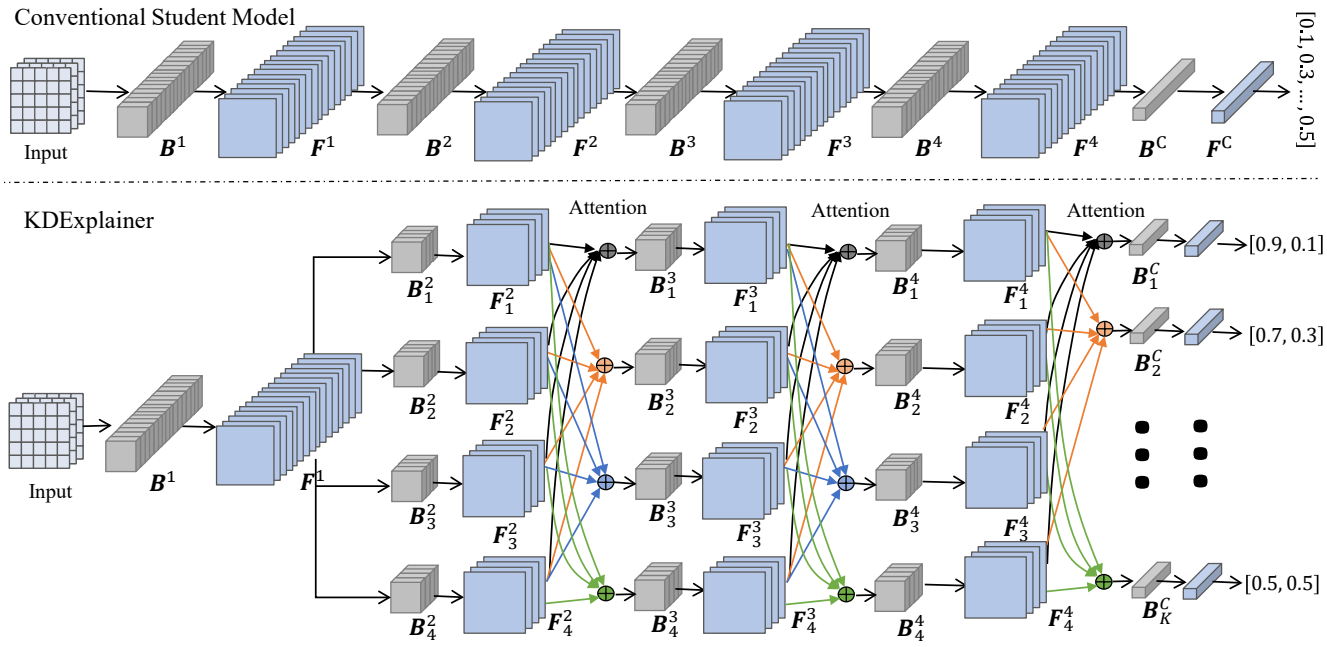


Figure 1: A conceptual illustration. **Top**: a conventional student network for knowledge distillation. **Bottom**: the proposed KDExplainer. To explain the effects of soft targets as the distillation objective, we use the KDExplainer as the student model for knowledge distillation.

on existing widely-used networks such as ResNet [He *et al.*, 2016], Wide Residual Network [Zagoruyko and Komodakis, 2016], and VGG [Simonyan and Zisserman, 2014]. These classic DNNs have some common characteristics: these models are usually composed by several blocks, each of which is a stack of convolutional layers, batch normalization layers [Ioffe and Szegedy, 2015], and nonlinear activation layers. The number of filters usually keeps fixed in the same block and changes across different blocks.

Formally, we use B^i to denote the i-th block, and then a standard DNN can be formulated as $F_{DNN} = B^C \circ B^L \circ \cdots \circ B^2 \circ B^1$, where symbol $\circ$ denotes the function composition operation. B^C is the classification block that consists of the fully connected layer and the softmax layer. KDExplainer roughly follows the design of the existing DNN, but divides each block B^i (except the first block B^1) into N_i equal-sized sub-blocks, *i.e.*, $\hat{B}^i = \{B^i_1, B^i_2, ..., B^i_{N_i}\}$ for any $i > 1$. We view each sub-block B^i_j as an expert, and introduce a task-oriented attention module before each expert as the gating function to select a combination of the outputs from previous experts. Therefore, the whole model can be viewed as an HME model.

Task-oriented Attention. The proposed attention is "task-oriented" as the parameters of the attention module are trained on the whole dataset, which is largely different from the attention mechanism in existing literature [Bahdanau *et al.*, 2015; Bello *et al.*, 2019] where the attention weights are determined by instances. For each sub-block B^i_j, we use a trainable parameter vector $\boldsymbol{v}^i_j = [v^i_{j,1}, v^i_{j,2}, ..., v^i_{j,N_{i-1}}]$ to learn the attention distribution over previous experts. Let output feature maps from previous blocks be $\left\{\boldsymbol{F}^{i-1}_1, ..., \boldsymbol{F}^{i-1}_{N_{i-1}}\right\}$. At training phase, the input of sub-block B^i_j is computed by

$$\widetilde{\boldsymbol{F}^i_j} = \sum\nolimits_{k=1}^{N_{i-1}} a^i_{j,k} \cdot \boldsymbol{F}^{i-1}_k, \tag{3}$$

where $a^i_{j,k}$ is the attention weight of the k-th expert, $a^i_{j,k} = \frac{\exp(v^i_{j,k}/T)}{\sum_m \exp(v^i_{j,m}/T)}$. T is a temperature hyper-parameter shared by all attention modules. Note that the classification block B^C is divided into K blocks, where K equals the number of classes in the classification problem. The multi-class classification problem thus turns into an ensemble of binary classification problems in KDExplainer, as shown in Figure 1.

Explaining KD by KDExplainer. Recall that KDExplainer is a substitute of the conventional student model, using KDExplainer to understand the effects of KD is straightforward: analyzing differences between the experimental results of KDExplainer trained with and without KD. As KDExplainer reformulates the multi-class classification as multiple binary classification tasks, each binary classification task encounters the imbalanced data problem. Let $\boldsymbol{p}^s_k = [p^s_{k,0}, p^s_{k,1}]$ be the probability predictions of KDExplainer for the k-th classification task, and $\boldsymbol{y}_k = [y_{k,0}, y_{k,1}]$ be the ground truth in the form of a one-hot vector. When the proposed KDExplainer is trained with KD, the objective function is

$$\mathcal{O}_{\text{KD}} = \sum\nolimits_{k=1}^{K} \left\{ \alpha \mathcal{L}_{\text{WCE}}\left(\boldsymbol{p}^s_k, \boldsymbol{y}_k\right) + (1-\alpha)\mathcal{D}_{\text{KL}}\left(\hat{\boldsymbol{q}}^t_k, \hat{\boldsymbol{p}}^s_k\right) \right\}, \tag{4}$$

where $\mathcal{L}_{\text{WCE}}$ is the weighted cross-entropy loss

$$\mathcal{L}_{\text{WCE}} = -w_0 y_{k,0} \log p^s_{k,0} - w_1 y_{k,1} \log p^s_{k,1}. \tag{5}$$

w_0 and w_1 are the weights balancing the cost incurred by the positive and the negative samples, which alleviates the negative effects caused by imbalanced data. $\hat{\boldsymbol{q}}^t_k$ and $\hat{\boldsymbol{p}}^s_k$ are the softened category probabilities from the teacher and the student

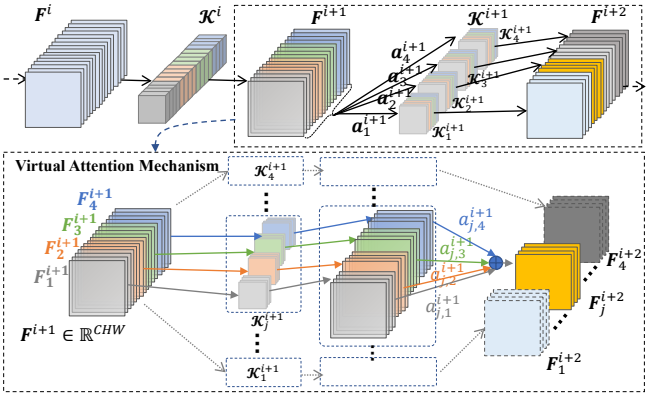


Figure 2: The proposed virtual attention mechanism. For simplicity, here we only depict the details of how $\boldsymbol{F}^{i+1}$ is convolved with the virtual filter block $\mathcal{K}_j^{i+1}$ to produce $\boldsymbol{F}_j^{i+2}$.

models, respectively. Note that all teacher models involved in this paper are still conventional DNNs for multi-class classification, so we convert the K-class probability prediction $\boldsymbol{p}^t = [p_0^t, p_1^t, ..., p_K^t]$ from the teacher to K two-dimensional probability vectors as follows

$$\hat{\boldsymbol{q}}_k^t = [1 - p_k^t, p_k^t], \text{ for any } k \in \{1, 2, ..., K\}. \tag{6}$$

If KDEplainer is trained without KD, the second term in Eqn. 4 is removed.

KDExplainer enjoys an appealing property thanks to the elaborate design: after training, it becomes a tree-like multi-branch network (i.e., neural tree) if only the maximum in the attention weights is kept:

$$a_{j,k}^i = \begin{cases} 1, \text{ if } k = \arg\max_m a_{j,m}^i; \\ 0, \text{ otherwise.} \end{cases} \tag{7}$$

The derived neural trees provide us with stronger support for interpretation and analysis of knowledge distillation.

4 Virtual Attention Mechanism

KDExplainer is tailored to understand the effects of soft targets in KD. However, it may suffer from slight accuracy sacrifice compared to conventional student DNNs due to the *ad hoc* design that multi-class classification turns into multiple binary classification tasks. Here we propose a virtual attention mechanism that is easily integrated into existing student models with few architecture modifications, to retain the capacity of these student models for higher KD performance. As shown in Figure 2, VAM views all convolution filters in a layer as several "virtual" filter blocks, each filter block akin to the expert in the KDExplainer. Note that "virtual" means that VAM does not really split the convolution filters into any blocks. The only modification is that VAM slightly changes the conventional convolution operation by incorporating an attention mechanism.

Formally, we use $\mathcal{K}^i \in \mathbb{R}^{C_{out}^i C_{in}^i SS}$ to denote the convolution filters in the i-th layer, where C_{in}^i and C_{out}^i denote the input and the output channels. Here for simplicity, we assume all filters are square, where both the width and the length are S. Assume the input tensor of the i-th convolution layer is $\boldsymbol{F}^i \in \mathbb{R}^{C_{in}^i HW}$, where H and W are the height and the width of the feature map. After the i-th layer and the $(i+1)$-th layer, the features turn into $\boldsymbol{F}^{i+1} \in \mathbb{R}^{C_{out}^i HW}$ and $\boldsymbol{F}^{i+2} \in \mathbb{R}^{C_{out}^{i+1} HW}$, respectively.

To incorporate the attention module into the DNN, we view convolution filters in each layer as a set of filter blocks. Assume the filters in the i-th layer are divided into M virtual groups, such that $\mathcal{K}^i = \{\mathcal{K}_1^i, \mathcal{K}_2^i, ..., \mathcal{K}_M^i\}, \mathcal{K}_j^i \in \mathbb{R}^{(C_{out}^i/M) C_{in}^i SS}$ for $1 \le j \le M$. After the i-th layer, the feature maps produced by each block can be calculated as follows

$$\boldsymbol{F}_j^{i+1} = \boldsymbol{F}^i \odot \mathcal{K}_j^i, \text{ for any } j \in \{1, 2, ..., M\}, \tag{8}$$

where $\odot$ denotes the convolution operation. The whole feature tensor can be denoted by

$$\boldsymbol{F}^{i+1} = \coprod_{m=1}^{M} \boldsymbol{F}_m^{i+1}, \tag{9}$$

where $\coprod$ denotes feature concatenation. Assume filters in the $(i+1)$-th layer are divided into N virtual blocks, such that $\mathcal{K}^{i+1} = \{\mathcal{K}_1^{i+1}, \mathcal{K}_2^{i+1}, ..., \mathcal{K}_N^{i+1}\}, \mathcal{K}_j^{i+1} \in \mathbb{R}^{(C_{out}^{i+1}/N) C_{in}^{i+1} SS}$ for $1 \le j \le N$. To align each filter block in the $(i+1)$-th layer with the output from each block of the i-th layer, we further divide each block $\mathcal{K}_j^{i+1}$ into M groups along the input channel, such that $\mathcal{K}_j^{i+1} = \{\mathcal{K}_{j,1}^{i+1}, \mathcal{K}_{j,2}^{i+1}, ..., \mathcal{K}_{j,M}^{i+1}\}, \mathcal{K}_{j,l}^{i+1} \in \mathbb{R}^{(C_{out}^{i+1}/N)(C_{in}^{i+1}/M) SS}$ for any $1 \le l \le M$. Similar to KDExplainer, we introduce a task-oriented attention module $\boldsymbol{a}_j^{i+1} = \left[a_{j,1}^{i+1}, a_{j,2}^{i+1}, ..., a_{j,M}^{i+1}\right]$ before each filter block $\mathcal{K}_j^{i+1}$, thus the output of the block can be computed by

$$\boldsymbol{F}_j^{i+2} = \sum_{m=1}^{M} a_{j,m}^{i+1} \boldsymbol{F}_m^{i+1} \odot \mathcal{K}_{j,m}^{i+1}. \tag{10}$$

The attention weights are computed by a softmax function over trainable parameters $\boldsymbol{v} = \left[v_{j,1}^{i+1}, ..., v_{j,M}^{i+1}\right]$, i.e.,

$$a_{j,m}^{i+1} = \frac{\exp\left(v_{j,m}^{i+1}\right)}{\sum_{k=1}^{M} \exp\left(v_{j,k}^{i+1}\right)}. \tag{11}$$

As soft targets are found to encourage lower entropy of the attention distribution (seen in Section 5.1), we introduce another regularization term based on the conventional KD objective in Eqn. 1

$$\begin{aligned} \mathcal{O}_{\text{KD}} =& (1-\alpha)\mathcal{D}_{\text{KL}}\left(p(\boldsymbol{z}^t;\tau), p(\boldsymbol{z}^s;\tau)\right) + \\ & \alpha\mathcal{L}_{\text{CE}}(p(\boldsymbol{z}^s), \boldsymbol{y}) + \gamma\mathcal{H}(A), \end{aligned} \tag{12}$$

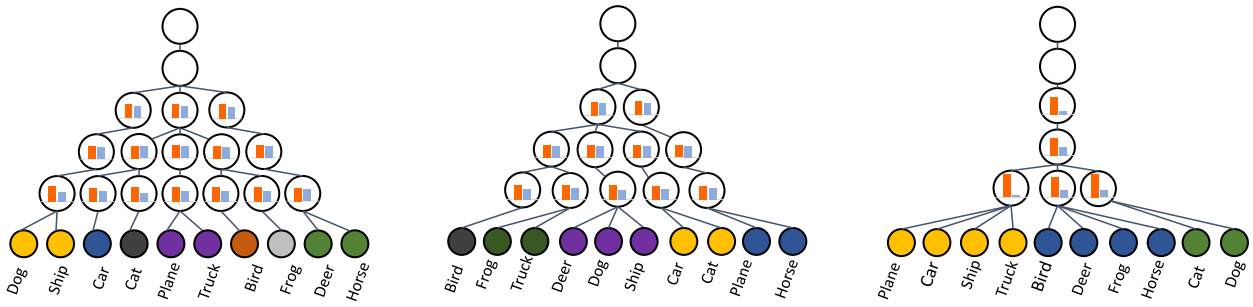
where A denotes all the involved attention distributions, and $\mathcal{H}$ is the sum of their entropy

$$\mathcal{H}(A) = \sum_i \sum_j \sum_k -a_{j,k}^i \log a_j^i, k. \tag{13}$$

5 Experiments

5.1 Explaining KD with KDExplainer

Experimental settings. Experiments are conducted on CIFAR-10 and CIFAR-100 [Krizhevsky *et al.*, 2009]. We



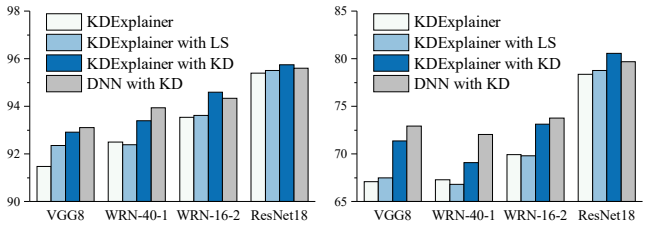
(a) Normal (b) Label Smoothing (c) KD

Figure 3: Visualization of the derived neural trees from the proposed KDExplainer, which is designed based on ResNet18 and trained on CIFAR-10. (a) KDExplainer trained with normal cross-entropy loss; (b) KDExplainer trained with label smoothing; (c) KDExplainer trained with KD. Each node denotes the retained block by removing the unused blocks according to Eqn 7. The histogram in each node represents the attention distribution over its input tensors. The red and the blue bars denote the maximum and the minimum, respectively.

implement the proposed KDExplainers based on four DNN architectures: ResNet18 [He *et al.*, 2016], VGG8 [Simonyan and Zisserman, 2014], WRN-16-2, and WRN-40-1 [Zagoruyko and Komodakis, 2016]. For all models, we use ResNet50 as their teacher model for KD. The initial learning rate is 0.1 and decayed every 30 epochs. The training ceases at 300 epochs. For more details, please refer to supplementary materials.

Experimental results. To understand the effects of vanilla KD, KDExplainer is trained with and without soft targets as the optimization objective. Furthermore, as label smoothing [Szegedy *et al.*, 2016] is recently also viewed as a type of KD [Yuan *et al.*, 2020], we also train the KDExplainer with label smoothing to understand its effects. In Figure 3, we visualize the derived neural trees by keeping only the maximum attention weight in every attention module as Eqn. 7. It can be seen that KD significantly encourages sparse connectivity between blocks, as its retained blocks are much fewer than normal training and label smoothing. It can be also verified by the attention distribution shown in the histograms in internal nodes, where KD produces sharper (i.e., lower entropy) attention distribution than normal training and label smoothing. Furthermore, KD produces more human-interpretable branching architecture than normal training and label smoothing. For example, in the derived trees, man-made vehicle categories are attached to the same branch, while animal categories are attached to other branches. This is a reasonable organization as similar categories or tasks share similar decision patterns, and thus they can share the same network branch. Less related categories or tasks, on the other hand, depend on different patterns to make their decision, such that they should be solved separately. The results of normal training and label smoothing somewhat violate this principle, thus their derived trees are much larger than that of KD.

Experimental analysis. In Figure 4, we provide the accuracies of all the KDExplainers. Under all experimental settings, KD significantly outperforms label smoothing and vanilla training. Label smoothing marginally improves the performance in most cases compared to vanilla training, but it sometimes leads to performance degradation. For example, it causes performance drops by 0.51% and 0.14% of WRN-40-1 and WRN-16-2 on CIFAR-100. These results prove



(a) CIFAR-10 (b) CIFAR-100

Figure 4: Accuracy (%) of KDExplainers trained with different objectives in different architectures. "DN N with KD" and "LS" denote the conventional DNN trained with KD and label smoothing.

that by properly organizing categories into different branches, KD helps modulate the knowledge conflicts between different tasks and thus achieves higher accuracy. Label smoothing, on the other hand, provides some regularization on the model learning, but it seems to have play no role in addressing conflicts between decision patterns of different categories. In the supplementary material, we provide more results and analyses to explain the differences between KD and label smoothing.

5.2 Improving KD with VAM

Experimental settings. Experiments are conducted on CIFAR-10, CIFAR-100, and Tiny-ImageNet. We incorporate VAM into six widely-used DNNs as student models, including ResNet18, resnet20[1], VGG8, WRN-16-2, WRN-40-1, and ShuffleNetV1 [Zhang *et al.*, 2018]. For all involved student models, we adopt ResNet50 as their teacher model to provide the soft logits. During the training phase of the student model, the learning rate is initially 0.05 (attention module 0.01), gets decayed at epoch 150, 180, 210 by a factor of 0.1, and ceases at 240. The temperature hyper-parameter is set to 1 for all attention modules and 4 for KD loss. The trading-off factor α is set to 0.9. The number of channels in each virtual block is 8 for VGG8, WRN-16-2, and WRN-40-1, 4 for resnet20, 16 for ResNet18, and 10 for ShuffleNetV1. For more details, please refer to the supplementary material.

[1]Fowlloing [Tian *et al.*, 2020], ResNet and resnet represent cifar- and ImageNet-style networks, respectively.

Data	Model	P-DNN	VAM	$\mathcal{L}_{\text{CE}}$	$\mathcal{D}_{\text{KL}}$	$\mathcal{H}$	VGG8	WRN-16-2	WRN-40-1	resnet20	ResNet18	ShufNetV1
CIFAR-10	M1	✓		✓			91.41	93.71	93.34	92.83	95.26	92.56
	M2		✓	✓			91.69	93.46	93.47	92.56	95.51	92.82
	M3		✓	✓		✓	91.82	93.90	93.95	92.86	94.99	92.83
	M4	✓		✓	✓		93.14	94.55	93.86	93.08	95.43	93.50
	M5		✓	✓	✓		93.19	94.67	93.96	93.39	95.51	93.70
	M6		✓	✓	✓	✓	**93.36**	**94.85**	**94.32**	**93.50**	**95.59**	**93.79**
CIFAR-100	M1	✓		✓			70.37	73.15	71.36	69.84	77.18	71.45
	M2		✓	✓			70.54	73.56	71.15	69.58	76.62	71.70
	M3		✓	✓		✓	70.98	73.92	71.61	69.71	78.30	71.81
	M4	✓		✓	✓		73.53	75.01	73.44	70.05	79.54	75.41
	M5		✓	✓	✓		73.78	75.43	73.87	70.32	79.63	75.62
	M6		✓	✓	✓	✓	**74.17**	**75.63**	**73.92**	**70.43**	**79.77**	**76.10**
Tiny-ImageNet	M1	✓		✓			56.47	57.52	56.26	50.54	65.59	60.52
	M2		✓	✓			57.18	58.16	56.02	50.53	65.79	61.24
	M3		✓	✓		✓	57.63	58.08	56.23	50.70	66.33	62.47
	M4	✓		✓	✓		60.41	57.91	56.39	52.78	69.37	65.54
	M5		✓	✓	✓		60.45	**58.24**	56.80	52.83	69.63	65.32
	M6		✓	✓	✓	✓	**60.57**	58.01	**56.99**	**53.54**	**69.90**	**65.65**

Table 1: Top-1 classification accuracy in % of six model variants. "P-DNN" denotes the plain DNN. Bold font indicates the best performance, and blue font denotes the second best. Experiments are repeated three times and the average results are reported.

Method	Model	CIFAR-10		CIFAR-100	
		resnet20	WRN-16-2	resnet20	WRN-16-2
FitNet	P-DNN	92.49	93.99	69.09	72.98
	VAM	**92.95**	**94.08**	**70.11**	**73.79**
FT	P-DNN	92.39	93.74	69.47	72.78
	VAM	**93.01**	**93.78**	**69.67**	**73.43**
SP	P-DNN	93.01	94.55	69.17	74.40
	VAM	**93.51**	**94.61**	**70.35**	**74.91**
CRD	P-DNN	91.83	93.36	70.64	74.97
	VAM	**92.16**	**93.53**	**70.85**	**75.15**

Table 2: Results of combing VAM with other KD methods. "P-DNN" denotes the plain DNN.

Improving vanilla KD. As VAM is motivated by vanilla KD, we first validate its effectiveness upon vanilla KD. To give a more comprehensive view of VAM, we make comparisons between six model variants. Note that we simply use VAM to denote the DNN incorporated with the proposed virtual attention module. (M1) Plain+$\mathcal{L}_{\text{CE}}$: plain DNN trained with only $\mathcal{L}_{\text{CE}}$; (M2) VAM+$\mathcal{L}_{\text{CE}}$: VAM trained with $\mathcal{L}_{\text{CE}}$; (M3) VAM+$\mathcal{L}_{\text{CE}}$+$\mathcal{H}$: VAM trained with $\mathcal{L}_{\text{CE}}$ and $\mathcal{H}$ (Eqn. 13); (M4) Plain+$\mathcal{L}_{\text{CE}}$+$\mathcal{D}_{\text{KL}}$: plain DNN trained with $\mathcal{L}_{\text{CE}}$ and $\mathcal{D}_{\text{KL}}$; (M5) VAM+$\mathcal{L}_{\text{CE}}$+$\mathcal{D}_{\text{KL}}$: VAM trained with $\mathcal{L}_{\text{CE}}$ and $\mathcal{D}_{\text{KL}}$; (M6) VAM+$\mathcal{L}_{\text{CE}}$+$\mathcal{D}_{\text{KL}}$+$\mathcal{H}$: VAM trained with $\mathcal{L}_{\text{CE}}$, $\mathcal{D}_{\text{KL}}$ and $\mathcal{H}$. Experimental results are shown in Table 1. In general, DNN incorporated with VAM yields consistently superior performance to the plain DNN without VAM. For example, M5 consistently outperforms M4 in almost all our experiments, which validates its effectiveness under KD. Furthermore, when optimized with low entropy constraint $\mathcal{H}$, VAM produces better performance (M3>M2, M6>M5) under almost all settings. As $\mathcal{H}$ is motivated by the results from KDExplainer, it indicates that the proposed KDExplainer indeed provides us with general and valuable insights into KD.

Improving state-of-the-art KD. Although VAM is motivated by vanilla KD, it is also straightforward to be combined with other KD methods. Here we evaluate VAM combined with other KD methods, including FitNets [Romero *et al.*, 2014], FT [Kim *et al.*, 2018], SP [Tung and Mori, 2019], and CRD [Tian *et al.*, 2020]. Results are listed in Table 2. It can be seen that combined with other KD methods, VAM still yields performances consistently superior to the plain DNN though it is motivated by only vanilla KD. For resnet20, VAM achieves $0.20\% \sim 1.18\%$ performance boosts on CIFAR-100 in four KD methods. For WRN-16-2, VAM improves $0.18\% \sim 0.81\%$ on CIFAR-100. Since VAM brings negligible additional overhead, it is an economical way to further improve the performance of existing KD methods. Please refer to the supplementary material for more experimental results, including how the block size and the hyper-parameter γ affect the final performance.

6 Conclusion and Future Work

In this paper, we propose KDExplainer to shed light on the working mechanism underlying soft targets during KD. We find that KD implicitly modulates the knowledge conflicts between different subtasks, and effectively brings about more benefits as compared to label smoothing. Based on these observations, we propose a portable module, VAM, to further improve the results of vanilla KD alongside other state-of-the-art ones. Extensive experimental results exhibit that the proposed VAM significantly enhances the KD performance at a negligible additional cost. In our future work, we will extend the proposed VAM to other tasks and systematically evaluate its effectiveness beyond the scope of KD.

Acknowledgments

This work is funded by the National Key R&D Program of China (Grant No: 2018AAA0101503) and the Science and technology project of SGCC (State Grid Corporation of China): fundamental theory of human-in-the-loop hybrid-augmented intelligence for power grid dispatch and control.

References

[Bahdanau *et al.*, 2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arxiv*, 2015.

[Bello *et al.*, 2019] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V. Le. Attention augmented convolutional networks. In *ICCV*, 2019.

[Byeongho Heo, 2019] Sangdoo Yun Jin Young Choi Byeongho Heo, Minsik Lee. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *AAAI*, 2019.

[Chen *et al.*, 2020] Hanting Chen, Yunhe Wang, Chunjing Xu, Boxin Shi, Chao Xu, Qi Tian, and Chang Xu. Addernet: Do we really need multiplications in deep learning? *CVPR*, 2020.

[Devlin *et al.*, 2019] J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[Furlanello *et al.*, 2018] T. Furlanello, Zachary Chase Lipton, Michael Tschannen, L. Itti, and Anima Anandkumar. Born again neural networks. In *ICML*, 2018.

[Han *et al.*, 2015] Song Han, J. Pool, John Tran, and W. Dally. Learning both weights and connections for efficient neural network. *arXiv*, 2015.

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

[Hinton *et al.*, 2015] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *arXiv*, 2015.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv*, 2015.

[Jacob *et al.*, 2018] B. Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *CVPR*, 2018.

[Kim *et al.*, 2018] Jangho Kim, SeongUk Park, and Nojun Kwak. Paraphrasing complex network: Network compression via factor transfer. In *NeurIPS*, 2018.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

[Li *et al.*, 2016] J. Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *arXiv*, 2016.

[Liu *et al.*, 2019] Yufan Liu, Jiajiong Cao, Bing Li, Chunfeng Yuan, Weiming Hu, Yangxi Li, and Yunqiang Duan. Knowledge distillation via instance relationship graph. *CVPR*, 2019.

[Liu *et al.*, 2021] Huihui Liu, Yiding Yang, and Xinchao Wang. Overcoming catastrophic forgetting in graph neural networks. *AAAI*, 2021.

[Mnih *et al.*, 2014] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu. Recurrent models of visual attention. *arXiv*, 2014.

[Park *et al.*, 2019] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. *CVPR*, 2019.

[Romero *et al.*, 2014] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv*, 2014.

[Romero *et al.*, 2015] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arxiv*, 2015.

[Shen *et al.*, 2019] Chengchao Shen, Xinchao Wang, Jie Song, Li Sun, and Mingli Song. Amalgamating knowledge towards comprehensive classification. *AAAI*, 2019.

[Shen *et al.*, 2021] Chengchao Shen, Xinchao Wang, Youtan Yin, Jie Song, Sihui Luo, and Mingli Song. Progressive network grafting for few-shot knowledge distillation. *AAAI*, 2021.

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

[Szegedy *et al.*, 2016] Christian Szegedy, V. Vanhoucke, S. Ioffe, Jon Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *CVPR*, 2016.

[Tang *et al.*, 2020] Jiaxi Tang, Rakesh Shivanna, Zhe Zhao, D. Lin, Anima Singh, Ed Huai hsin Chi, and S. Jain. Understanding and improving knowledge distillation. *arXiv*, 2020.

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

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

[Wiegreffe and Pinter, 2019] Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *EMNLP/IJCNLP*, 2019.

[Xu *et al.*, 2015] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, R. Salakhutdinov, R. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[Yang *et al.*, 2020a] Yiding Yang, Zunlei Feng, Mingli Song, and Xinchao Wang. Factorizable graph convolutional networks. *NeurIPS*, 2020.

[Yang *et al.*, 2020b] Yiding Yang, Jiayan Qiu, Mingli Song, Dacheng Tao, and Xinchao Wang. Distilling knowledge from graph convolutional networks. *CVPR*, 2020.

[Yang *et al.*, 2020c] Yiding Yang, Jiayan Qiu, Mingli Song, Dacheng Tao, and Xinchao Wang. Learning propagation rules for attribution map generation. *ECCV*, 2020.

[Ye *et al.*, 2019] Jingwen Ye, Yixin Ji, Xinchao Wang, Kairi Ou, Dapeng Tao, and Mingli Song. Student becoming the master: Knowledge amalgamation for joint scene parsing, depth estimation, and more. *CVPR*, 2019.

[Yu *et al.*, 2017] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. *CVPR*, 2017.

[Yuan *et al.*, 2020] Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisiting knowledge distillation via label smoothing regularization. In *CVPR*, 2020.

[Zagoruyko and Komodakis, 2016] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv*, 2016.

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

[Zhang *et al.*, 2018] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018.