Improving Knowledge Distillation with Teacher's Explanation

Sayantan Chowdhury,* **Ben Liang,*** **Ali Tizghadam,†** **Ilijc Albanese†**
*University of Toronto, †TELUS
sayantan.chowdhury@mail.utoronto.ca

Abstract

Knowledge distillation (KD) improves the performance of a low-complexity student model with the help of a more powerful teacher. The teacher in KD is a black-box model, imparting knowledge to the student only through its predictions. This limits the amount of transferred knowledge. In this work, we introduce a novel Knowledge Explaining Distillation (KED) framework, which allows the student to learn not only from the teacher's predictions but also from the teacher's explanations. We propose a class of superfeature-explaining teachers that provide explanation over groups of features, along with the corresponding student model. We also present a method for constructing the superfeatures. We then extend KED to reduce complexity in convolutional neural networks, to allow augmentation with hidden-representation distillation methods, and to work with a limited amount of training data using chimeric sets. Our experiments over a variety of datasets show that KED students can substantially outperform KD students of similar complexity.

1 Introduction

The computational complexity of machine learning often hinders its deployment in devices with limited hardware capability. Knowledge distillation (KD) addresses this problem by designing low-complexity student models that are trained utilizing the information from more powerful teacher models. In the conventional setting of KD Hinton et al. [2015], the student is trained with true labels as well as the teacher's predictions smoothed by a temperature parameter. This helps the student generalize better and thus achieve superior performance on the test set. At high temperatures, KD has been shown to be equivalent to model compression proposed by Bucilă et al. [2006]. The success of Hinton's distillation has led to the development of various other distillation frameworks in the subsequent years, including logit-based approaches similar to Hinton's original KD Lopez-Paz et al. [2016], Vapnik and Izmailov [2015], Mirzadeh et al. [2020], Yuan et al. [2020], Fang et al. [2021], Zhao et al. [2022], Huang et al. [2022], Zheng and Yang [2024], Zhang et al. [2018], Lan et al. [2018], Dennis et al. [2023] and more complex techniques that also utilize the teacher's intermediate-layer outputs Romero et al. [2015], Zagoruyko and Komodakis [2017], Tung and Mori [2019], Kim et al. [2018], Tian et al. [2020], Yang et al. [2021], Park et al. [2019], Ahn et al. [2019], Heo et al. [2019], Liu et al. [2019], Shu et al. [2021], Chen et al. [2021a,b], Guo et al. [2023], Liu et al. [2023], Wang et al. [2024].

However, learning from the teacher's final or intermediate-layer predictions only transfers a fraction of the teacher's knowledge to the student. A good teacher should not just dictate the prediction outcome, but also explain to the student how to predict. This motivates us to propose a new *Knowledge Explaining Distillation (KED)* framework that utilizes teachers who transfer knowledge with both predictions and explanations.

One may add explanation to the black-box teacher used in conventional KD, by applying existing methods of Interpretable AI Ribeiro et al. [2016], Lundberg and Lee [2017], Štrumbelj and Kononenko [2014]. However, the explanation thus generated is sample-specific, and hence ultimately is not

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

useful for training a student. A brief survey of existing literature on distillation and explanation is provided in Appendix A in the supplementary material. Instead of explaining a black-box teacher, we consider new interpretable teachers that output the contribution of features toward the final prediction. Furthermore, an interpretable teacher that provides per-feature-based explanation will impose strict requirements on the dependency among the features and thus will achieve poor accuracy. Therefore, we propose to first group the features into superfeatures Ribeiro et al. [2016], Chowdhury et al. [2019], Jullum et al. [2021] and generate explanation only on the superfeatures. We name the resultant teachers *superfeature-explaining teachers*. We show that such teachers transfer more knowledge to the students than the black-box ones in regular KD and hence improve the student performance.

Our main contributions are summarized as follows:

- We propose a novel KED framework using superfeature-explaining teachers. A teacher in KED provides soft predictions on the training samples as well as their associated explanations. The student has similar but scaled-down architecture as the teacher and learns from the teacher's predictions and explanations. Instead of using sample-specific explanation in the conventional interpretable AI, we construct teachers that are architecturally hardwired to provide consistent explanation over different samples. We also develop an algorithm that groups feature into the required superfeatures.
- We further enhance the core design of KED in multiple directions. First, we extend KED to efficiently perform distillation with unstructured datasets using convolutional neural networks (CNNs). The model complexity of CNNs can be greatly reduced by our proposed method compared with existing KD techniques. Furthermore, we show that KED is easily composable with the prevalent hidden-representation distillation methods and thus can further improve the performance of the student. In addition, when the available training data is limited, we show that the explanations of different samples can be combined to construct labels for out-of-distribution samples. Thus, we introduce the concept of KED student training on a chimeric set and show that it is an effective method for data augmentation.
- We conduct extensive experiments over a variety of datasets including MNIST, Fashion-MNIST, Unicauca, CIFAR10, CIFAR100, and Tiny Imagenet. In all cases, we observe significant improvement over KD. As an example, for CIFAR100, a black-box student with the VGG8 architecture achieves 68.72% and 70.39% accuracy without and with KD, respectively. With KED, a student of similar complexity achieves 73.50% accuracy. We also study the effect of combining KED with hidden-representation distillation methods and the chimeric set, showing that the KED approach is flexible and effective.

2 Preliminaries on Knowledge Distillation

We first outline the general concept of knowledge distillation. We will set up the basic mathematical notations that will be used throughout this paper. Consider a classification problem given a labeled training dataset $\mathcal{D}$. Let $\mathcal{X}$ be the set of d-dimensional training samples and $\mathcal{Y}$ be the set of classes with $|\mathcal{X}| = K$ and $|\mathcal{Y}| = C$. Let $\mathbf{x} \in \mathcal{X}$ denote the feature vector of a sample and $y \in \mathcal{Y}$ denote its label. Then we have $\mathcal{D} = \{(\mathbf{x}, y)\}_{\mathbf{x}\in\mathcal{X}}$. In the classical, i.e., no distillation, setting, a classifier g minimizes the following categorical crossentropy loss function:

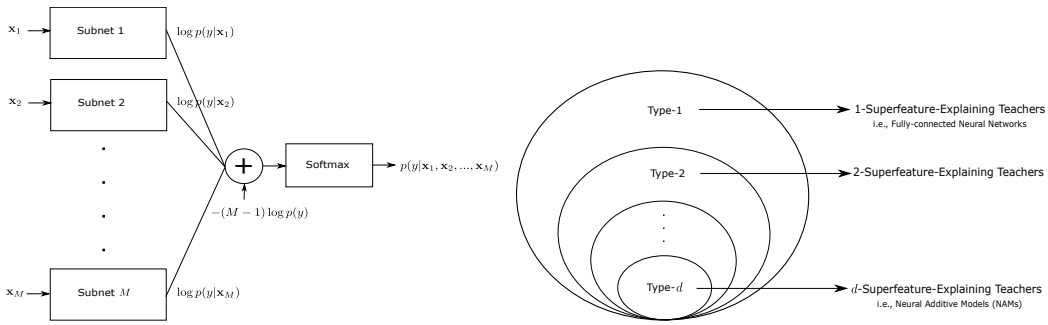
$$\mathcal{L}(g) = \mathbb{E}_{\mathcal{D}}[\mathcal{L}_{\text{ce}}(y, g(\mathbf{x}))]. \tag{1}$$

The goal of KD is to design a low-complexity *student model* with the help of a more powerful *teacher model*. Let f be the teacher that provides predictions for the student g. The student g is trained with the hard labels from the dataset $\mathcal{D}$ and some soft labels supplied by the teacher f Hinton et al. [2015]. The soft labels are the teacher's logits passed through a softmax function with a temperature parameter, T, or equivalently the teacher's predictions $f(\mathbf{x})$ passed through a function $\sigma_T(\cdot)$ defined as follows. Let $\{p_i : i \in \{1, ..., C\}\}$ denotes the teacher's predicted probability distribution over C classes. Then, $\sigma_T(\cdot)$ is given by

$$\sigma_T(p_i) = \frac{\exp(\log p_i/T)}{\sum_{j=1}^{C} \exp(\log p_j/T)}, \ \forall i \in \{1, ..., C\}. \tag{2}$$

The student minimizes the following loss function during training:

$$\mathcal{L}(g) = (1-\lambda)\mathbb{E}_{\mathcal{D}}[\mathcal{L}_{\text{ce}}(y, g(\mathbf{x}))] + T^2\lambda\mathbb{E}_{\mathcal{D}}[\mathcal{L}_{\text{kl}}(\sigma_T(f(\mathbf{x})), \sigma_T(g(\mathbf{x})))], \tag{3}$$



(a) (b)

Figure 1: Architecture and hierarchy of superfeature-explaining teacher.

where y is the hard label, $\mathcal{L}_{\text{ce}}$ is the categorical crossentropy loss, and $\mathcal{L}_{\text{kl}}$ is the KL divergence loss. The second term is scaled by T^2 as prescribed in Hinton et al. [2015], and the loss weight λ is a hyperparameter. The original KD approach has many successes in generating low-complexity models with superior performance Hinton et al. [2015], Kim et al. [2018], Mirzadeh et al. [2020]. As seen in (3), the teacher in KD remains a black-box model that provides only limited information to the student. In this work, our objective is to design a richer distillation framework that is more conducive to student learning, where the teacher provides explanations for its predictions in addition to the soft labels.

3 Knowledge Explaining Distillation

In this section, we elaborate upon the proposed KED framework. We first discuss the limitation of existing explanation methods and the motivation for constructing superfeature-explaining teachers. Then, we describe the general architecture of such teachers and the corresponding student model, followed by the mathematical description of KED. Finally, we present an algorithm to group features into superfeatures.

3.1 Limitation of Existing Feature Attribution-based Explanation Methods

It is nontrivial to add explanation to the teacher in KD. For a sample $\mathbf{x} \in \mathcal{X}$ and a pretrained teacher f, the existing feature attribution-based explanation methods Štrumbelj and Kononenko [2014], Lundberg and Lee [2017] quantify the contribution of each feature to the final prediction $f(\mathbf{x})$. However, this explanation is local and specific to the sample $\mathbf{x}$, in the sense that the same feature value can be mapped to different explanations for different samples and thus the underlying relation between input features and explanation becomes one-to-many. Thus the target function between the features and the teacher's explanation obtained using these methods is simply undefined. Therefore, a student cannot estimate the target function from such an explanation in a distillation framework. This argument holds also for the superfeature-based explanation methods Ribeiro et al. [2016], Chowdhury et al. [2019], Jullum et al. [2021].

The limitation of existing explanation methods motivates us to develop more interpretable teachers instead of directly applying current methods to explain black-box KD teachers. To facilitate effective learning by students, interpretable teachers should be learning models hardwired to provide consistent explanations over different samples. In the next section, we will see that the function between the features and their explanation is built into the architecture of the proposed superfeature-explaining teachers. Therefore, this class of teachers is perfectly suitable for guiding the students in a distillation setting.

3.2 Superfeature-Explaining Teachers

In one extreme, the black-box teacher in the original KD does not provide any explanation to the student. In the other extreme, however, a teacher that provides an explanation for each feature may be too restrictive. Therefore, a more general approach is to obtain an explanation for groups of features. In this work, we consider superfeature-based explanation for constructing an interpretable teacher in a distillation framework. We define superfeatures as disjoint groups of features. Let $\mathbf{x} \in \mathcal{X}$ be the feature vector of a sample. Then, we can group the features in $\mathbf{x}$ as a set of superfeatures

$\{\mathbf{x}_1, ..., \mathbf{x}_M\}$ where $\mathbf{x}_m,\ 1 \leq m \leq M$, is the m-th superfeature. We will defer the discussion on how to group features to Section 3.4.

Given an input $\mathbf{x}$, any classifier's job is to estimate the distribution $p(y|\mathbf{x})$ for each particular class label $y \in \mathcal{Y}$. To motivate our design, let's first suppose we have M independent superfeatures. Then, by Bayes' rule, we have

$$p(y|\mathbf{x}) = \frac{\frac{1}{p(y)^{M-1}} \prod_{m=1}^{M} p(y|\mathbf{x}_m)}{\sum_y \frac{1}{p(y)^{M-1}} \prod_{m=1}^{M} p(y|\mathbf{x}_m)}, \tag{4}$$

where $p(y)$ is the prior of class y. Thus $p(y|\mathbf{x}_m)$ may be viewed as the contributions of the superfeature $\mathbf{x}_m$ to the prediction $p(y|\mathbf{x})$. We will use $p(y|\mathbf{x}_m), \forall m$, as the teacher's explanation for its prediction $p(y|\mathbf{x})$.

Figure 1a illustrates the architecture of a superfeature-explaining teacher with M superfeatures. This teacher estimates the contribution of each superfeature using a subnet, which is a neural network with a softmax layer at the output. We compute the logarithm of the softmax outputs from all the subnets and sum them up to produce the total logit adjusted by the prior. An ultimate softmax function is applied to this total logit in order to obtain the final prediction $p(y|\mathbf{x})$. Both the teacher's final prediction $p(y|\mathbf{x})$ and its superfeature explanation $p(y|\mathbf{x}_m), \forall m$, will be used as inputs to train the student model.

We remark that there is a hierarchy of superfeature-explaining teachers according to the number of superfeatures. A black-box teacher may be viewed as having only one superfeature, i.e., the set of all features. On the other hand, the generalized additive model, e.g., NAM Agarwal et al. [2021] may be viewed as having d superfeatures, i.e., each feature is a superfeature. The hierarchy of superfeature-explaining teachers is shown in Figure 1b. At the top of this hierarchy sits type-1 teachers or the black-box teachers that are the most unrestricted. Type-1 teachers do not assume independence between their input features and therefore theoretically can estimate any distribution that can be estimated by the types below it. At the bottom of this hierarchy is type-d teachers that are the most restricted as they assume independence of all the features. As we go down the hierarchy, we see more interpretability of the teacher model at the expense of a drop in its prediction accuracy. The proposed KED in this work can be applied to any type. In particular, for type-1 teachers, our method reduces to Hinton's KD.

Interestedly, in Appendix B in the supplementary material, we show that the outputs of the superfeature-explaining teacher can be interpreted as Shapley value-based explanations.

3.3 Student in KED

In the KED framework, a type-M student g is trained with the hard labels from $\mathcal{D}$, and the soft labels and explanations provided by a type-M teacher f. As in the original KD, the soft labels are the teacher's predictions $f(x)$ passed through function $\sigma_T(\cdot)$ as defined in (2). Let $f_m(\mathbf{x}_m),\ 1 \leq m \leq M$, be the teacher's explanation for the m-th superfeature. The student outputs $g_m(\mathbf{x}_m)$, which is matched against $f_m(\mathbf{x}_m)$ to compute the loss. We denote the teacher's as well as the student's prior by f_0. We further apply another temperature parameter τ to the teacher's explanation passing through function $\sigma_\tau(\cdot)$, which is as defined in (2) with T replaced by τ.

The student's total loss is measured as a weighted combination of individual loss against the hard labels, the soft labels, and the explanations. Therefore, the student's loss function consists of three terms. The first term is the categorical crossentropy loss between the student's predictions $g(\mathbf{x})$ and the hard labels y. The second term is the KL divergence loss between the student's soft label predictions $\sigma_T(g(\mathbf{x}))$ and the teacher's soft labels $\sigma_T(f(\mathbf{x}))$ at temperature T. The third term is the average KL divergence loss over M superfeatures between the student's outputs $\sigma_\tau(g_m(\mathbf{x}_m))$ and the teacher's explanation $\sigma_\tau(f_m(\mathbf{x}_m)),\ 1 \leq m \leq M$, at temperature τ. Thus we write the student's total loss as

$$\begin{aligned}\mathcal{L}(g) = (1-\lambda)\mathbb{E}_{\mathcal{D}}[\mathcal{L}_{\text{ce}}(y, g(\mathbf{x}))] + T^2\lambda(1-\mu)\mathbb{E}_{\mathcal{D}}[\mathcal{L}_{\text{kl}}(\sigma_T(f(\mathbf{x})), \sigma_T(g(\mathbf{x})))] \\ + \frac{\tau^2\lambda\mu}{M}\sum_{m=1}^{M}\mathbb{E}_{\mathcal{D}}[\mathcal{L}_{\text{kl}}(\sigma_\tau(f_m(\mathbf{x}_m)), \sigma_\tau(g_m(\mathbf{x}_m)))].\end{aligned} \tag{5}$$

The loss weights λ and μ are two hyperparameters. Figure 2 given in Appendix C in the supplementary material illustrates the student in the KED framework.

Algorithm 1 Algorithm for constructing superfeatures

Input: Resolution
Output: Set of superfeatures
1: Compute average Hessian matrix over a random subset of training samples, $\bar{\mathbf{H}} \approx \sum_{y \in \mathcal{Y}} \mathbb{E}[\nabla \otimes \nabla \log p(y|\mathbf{x})]$.
2: Construct a matrix $\bar{\mathbf{H}}_{\text{abs}}$ taking absolute values of entries in $\bar{\mathbf{H}}$.
3: $\mathbf{W} \leftarrow \bar{\mathbf{H}}_{\text{abs}} + \bar{\mathbf{H}}_{\text{abs}}^T$
4: $\text{diag}(\mathbf{W}) \leftarrow 0$
5: Set up a graph with weight matrix $\mathbf{W}$.
6: Perform community detection using the Louvain algorithm with the given resolution.
7: Record the communities, i.e., the set of superfeatures.
8: **return** Set of superfeatures

3.4 Constructing Superfeatures

One key aspect of superfeature-explaining teachers is the superfeatures themselves. If there is sufficient domain knowledge, the data can be collected in a way such that the groups of features naturally form independent superfeatures. However, this is not always possible in general machine learning tasks. An alternative strategy is to group strongly dependent features together into a superfeature. Here we use a Hessian-based measure of feature dependency. Referring to Figure 1a, we define the total logit z for a type-M superfeature-explaining teacher as follows:

$$z = \sum_{m=1}^{M} \log p(y|\mathbf{x}_m) - (M-1) \log p(y). \tag{6}$$

We see that z is additively separable over the superfeatures $\mathbf{x}_1, ..., \mathbf{x}_M$. Therefore, if two features x_i and x_j belong to two different superfeatures, the cross-partial derivatives of z with respect to those two features should be zero, i.e.,

$$\frac{\partial^2 z}{\partial x_i \partial x_j} = \frac{\partial^2 z}{\partial x_j \partial x_i} = 0, \quad \forall x_i \in \mathbf{x}_m, x_j \in \mathbf{x}_n, m \neq n. \tag{7}$$

Thus, we measure the dependency between each pair of features by the entries of $d \times d$ Hessian matrix of z. Note that only the magnitude of Hessian entries is significant for determining dependency.

We note that the total logit $z \approx \log p(y|\mathbf{x})$. We first estimate $p(y|\mathbf{x})$ using a type-1 black-box teacher. Then, we approximate the expectation of the Hessian matrix, $\bar{\mathbf{H}} \approx \sum_{y \in \mathcal{Y}} \mathbb{E}[\nabla \otimes \nabla \log p(y|\mathbf{x})]$ over a random subset of training samples. Let $\bar{\mathbf{H}}_{\text{abs}}$ denote the matrix of absolute values of the entries in $\bar{\mathbf{H}}$. We define the $d \times d$ pairwise dependency matrix as $\mathbf{W} = \bar{\mathbf{H}}_{\text{abs}} + \bar{\mathbf{H}}_{\text{abs}}^T$. Thus the (i, j) entry of this matrix is

$$\mathbf{W}_{ij} \approx \left| \sum_{y \in \mathcal{Y}} \mathbb{E}\left[\frac{\partial^2 z}{\partial x_i \partial x_j}\right] \right| + \left| \sum_{y \in \mathcal{Y}} \mathbb{E}\left[\frac{\partial^2 z}{\partial x_j \partial x_i}\right] \right|. \tag{8}$$

Now we are ready to build a weighted undirected graph $G = (V, E)$ where V is the set of vertices, i.e., d features, E is the edges between each pair of those features, and each edge is assigned a weight given by the matrix $\mathbf{W}$. Then, the grouping of features into approximately independent superfeatures corresponds to finding approximately independent communities in graph G. We may use any community detection method, e.g., Louvain Blondel et al. [2008], to find such superfeatures. Even though many practical problems require a large number of features and hence a large graph G, Louvain is a simple and fast algorithm that can detect communities efficiently in $O(|V| \log |V|)$ run time.

Algorithm 1 summarizes the above procedure for constructing the superfeatures. The detected communities constitute our set of superfeatures. Typically, we find a higher or lower number of superfeatures by tuning the resolution parameter of the Louvain method.

Due to the lack of space, we discuss the extensions of KED for CNNs, composition with hidden-representation methods and the chimeric set in Appendix D in the supplementary material.

4 Experimental Results

In this section, we study the performance of KED and compare it against that of conventional KD with black-box teachers. We also compare KED against KD when augmented with various hidden-representation distillation methods. The details of our experimental setup and the numerical results are provided in Appendix E in the supplementary material.

Table 1: Test accuracy for CIFAR10 with varying model architectures.

Teacher	Resnet44x2	Resnet56x2	Resnet56x2	WRN-16-8	WRN-16-8	WRN-28-4	WRN-28-4	VGG13
Black-box	94.47±0.03%	94.22±0.03%	94.22±0.03%	95.02±0.03%	95.02±0.03%	94.81±0.03%	94.81±0.03%	93.35±0.04%
Type-M	94.37±0.03%	94.37±0.03%	94.37±0.03%	94.75±0.03%	94.75±0.03%	94.95±0.03%	94.95±0.03%	92.90±0.04%
Student	WRN-10-1	Resnet8	WRN-10-1	Resnet8	WRN-16-1	Resnet8	VGG8	VGG8
No Distl.	88.44±0.05%	87.98±0.05%	88.44±0.05%	87.98±0.05%	91.42±0.04%	87.98±0.05%	90.93±0.04%	90.93±0.04%
KD	88.69±0.04%	88.53±0.04%	88.39±0.05%	88.96±0.04%	91.57±0.04%	88.68±0.04%	90.26±0.05%	90.86±0.04%
e^2-KD	89.13±0.04%	88.58±0.05%	89.25±0.04%	89.16±0.05%	91.74±0.04%	88.93±0.04%	91.06±0.04%	91.31±0.04%
KED	89.88±0.04%	**89.43±0.04%**	89.82±0.04%	**89.70±0.05%**	91.89±0.04%	89.38±0.04%	92.13±0.04%	91.57±0.04%
KD+FitNet	89.17±0.04%	88.67±0.05%	89.42±0.04%	88.94±0.04%	91.83±0.04%	88.98±0.04%	91.31±0.04%	91.07±0.04%
KED+FitNet	**90.08±0.04%**	89.06±0.05%	89.62±0.04%	89.36±0.04%	92.04±0.04%	89.19±0.05%	91.77±0.04%	91.87±0.04%
KD+AT	88.98±0.04%	88.32±0.05%	89.04±0.05%	88.26±0.05%	92.07±0.04%	88.15±0.05%	90.96±0.04%	91.29±0.04%
KED+AT	90.03±0.04%	89.21±0.05%	89.31±0.04%	89.30±0.04%	92.18±0.04%	89.05±0.04%	91.81±0.04%	91.69±0.04%
KD+SP	88.92±0.04%	88.43±0.04%	89.10±0.04%	88.87±0.05%	92.18±0.04%	88.64±0.04%	91.79±0.04%	91.15±0.04%
KED+SP	89.43±0.05%	88.95±0.04%	89.67±0.04%	89.54±0.05%	92.39±0.03%	89.09±0.04%	**92.36±0.04%**	**91.94±0.04%**
KD+DKD	89.05±0.04%	88.47±0.04%	89.16±0.04%	88.91±0.05%	92.22±0.04%	88.78±0.04%	91.44±0.04%	91.26±0.04%
KED+DKD	90.01±0.04%	89.17±0.04%	**89.93±0.04%**	89.57±0.05%	92.56±0.03%	**89.47±0.04%**	92.28±0.04%	91.73±0.04%
KD+DIST	88.98±0.04%	88.35±0.04%	88.97±0.05%	88.64±0.05%	92.33±0.04%	88.71±0.04%	91.36±0.04%	91.18±0.04%
KED+DIST	89.74±0.05%	89.02±0.04%	89.75±0.04%	89.45±0.05%	**92.73±0.03%**	89.25±0.04%	92.19±0.04%	91.56±0.04%

4.1 Distillation Performance with CNNs

In Tables 1, 2 and 3, we present the test accuracy of KED with CNNs over full datasets of CIFAR10, CIFAR100 and Tiny Imagenet, respectively. Tables 2 and 3 are included in Appendix E in the supplementary material. Here e^2-KD is the explanation-enhanced KD method of Parchami-Araghi et al. [2024], the labels FitNet, AT, and SP indicate the hidden-representation methods of Romero et al. [2015], Zagoruyko and Komodakis [2017], Tung and Mori [2019], respectively and the labels DKD and DIST indicate the logit-based methods of Zhao et al. [2022] and Huang et al. [2022], respectively.

Let us consider Resnet56x2 over CIFAR10 as shown in Table 1. When distilled to Resnet8 using KD, we achieve 88.53% accuracy. However, the KED student outperforms the KD student, reaching 89.43% accuracy. We notice a similar trend for all other architectures.

Furthermore, take Resnet56x2 over CIFAR100 for another example, which is shown in Table 2. When distilled to Resnet20 using KD, we achieve 69.19% accuracy. However, the KED student outperforms KD reaching 70.94% accuracy. Augmented with FitNet, the KED student further improves to 71.79%, but the same augmentation of KD reaches only 69.42%. We observe a similar trend for all other combinations of teacher and student architectures.

As shown in Table 3, for Tiny Imagenet, when the teacher is a Resnet56x2 model, a black-box Resnet20 student achieves an accuracy of 52.72% with KD. In KED, the student attains 54.96% test accuracy. Augmenting KED with attention transfer, the student's performance further improves to 55.35%, while the same augmentation of KD achieves only 53.71%. More importantly, the gap between the KED teacher and the KED student is always significantly narrower than in KD.

4.2 Further Experiments

Further experiments are conducted to evaluate the impact of the hyperparameters on KED and the performance of Algorithm 1 (see Appendix E in the supplementary material).

5 Conclusion

In this work, we propose a new KED framework for training a low-complexity student model with knowledge transfer from a more powerful teacher. Unlike the conventional KD, under KED the teacher does not simply give predictions to the student but also explains those predictions. The proposed solution can be adapted to reduce the complexity of CNNs and to allow more effective distillation along with hidden-representation distillation methods as well as small training datasets. Our experimental results show that a KED teacher transfers more knowledge and can substantially improve student learning, leading to superior distillation performance in a wide variety of datasets and network architectures.

References

G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *Proc. NeurIPS Deep Learning and Representation Learning Workshop*, 2015.

C. Bucilă, R. Caruana, and A. Niculescu-Mizil. Model compression. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2006.

D. Lopez-Paz, B. Schölkopf, L. Bottou, and V. Vapnik. Unifying distillation and privileged information. In *Proc. International Conference on Learning Representations (ICLR)*, 2016.

V. Vapnik and R. Izmailov. Learning using privileged information: Similarity control and knowledge transfer. *Journal of Machine Learning Research*, 16(61):2023–2049, 2015.

S. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *Proc. AAAI Conference on Artificial Intelligence*, 2020.

L. Yuan, F. Tay, G. Li, T. Wang, and J. Feng. Revisiting knowledge distillation via label smoothing regularization. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

G. Fang, J. Song, X. Wang, C. Shen, X. Wang, and M. Song. Contrastive model invertion for data-free knolwedge distillation. In *Proc. International Joint Conference on Artificial Intelligence*, 2021.

B. Zhao, Q. Cui, R. Song, Y. Qiu, and J. Liang. Decoupled knowledge distillation. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

T. Huang, S. You, F. Wang, C. Qian, and C. Xu. Knowledge distillation from A stronger teacher. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

K. Zheng and E. Yang. Knowledge distillation based on transformed teacher matching. In *Proc. International Conference on Learning Representations (ICLR)*, 2024.

Y. Zhang, T. Xiang, T. Hospedales, and H. Lu. Deep mutual learning. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

X. Lan, X. Zhu, and S. Gong. Knowledge distillation by on-the-fly native ensemble. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

D. Dennis, A. Shetty, A. Sevekari, K. Koishida, and V. Smith. Progressive ensemble distillation: Building ensembles for efficient inference. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

A. Romero, N. Ballas, S. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. In *Proc. International Conference on Learning Representations (ICLR)*, 2015.

S. Zagoruyko and N. Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *Proc. International Conference on Learning Representations (ICLR)*, 2017.

F. Tung and G. Mori. Similarity-preserving knowledge distillation. In *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

J. Kim, S. Park, and N. Kwak. Paraphrasing complex network: Network compression via factor transfer. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Y. Tian, D. Krishnan, and P. Isola. Contrastive representation distillation. In *Proc. International Conference on Learning Representations (ICLR)*, 2020.

J. Yang, B. Martínez, A. Bulat, and G. Tzimiropoulos. Knowledge distillation via softmax regression representation learning. In *Proc. International Conference on Learning Representations (ICLR)*, 2021.

W. Park, D. Kim, Y. Lu, and M. Cho. Relational knowledge distillation. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

S. Ahn, S. Hu, A. Damianou, N. Lawrence, and Z. Dai. Variational information distillation for knowledge transfer. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

B. Heo, M. Lee, S. Yun, and J. Choi. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *Proc. AAAI Conference on Artificial Intelligence*, 2019.

Y. Liu, K. Chen, C. Liu, Z. Qin, Z. Luo, and J. Wang. Structured knowledge distillation for semantic segmentation. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

C. Shu, Y. Liu, J. Gao, Z. Yan, and C. Shen. Channel-wise knowledge distillation for dense prediction. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

P. Chen, S. Liu, H. Zhao, and J. Jia. Distilling knowledge via knowledge review. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021a.

D. Chen, J. Mei, Y. Zhang, C. Wang, Z. Wang, Y. Feng, and C. Chen. Cross-layer distillation with semantic calibration. In *Proc. AAAI Conference on Artificial Intelligence*, 2021b.

Z. Guo, H. Yan, H. Li, and X. Lin. Class attention transfer based knowledge distillation. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

X. Liu, L. Li, C. Li, and A. Yao. Norm: Knowledge distillation via n-to-one representation matching. In *Proc. International Conference on Learning Representations (ICLR)*, 2023.

J. Wang, Y. Chen, Z. Zheng, X. Li, M. Cheng, and Q. Hou. Crosskd: Cross-head knowledge distillation for object detection. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

M. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

S. Lundberg and S. Lee. A unified approach to interpreting model predictions. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

E. Štrumbelj and I. Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, 41(3):647–665, 2014.

S. Chowdhury, B. Liang, and A. Tizghadam. Explaining class-of-service oriented network traffic classification with superfeatures. In *Proc. ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks (Big-DAMA)*, 2019.

M. Jullum, A. Redelmeier, and K. Aas. Efficient and simple prediction explanations with groupshapley: A practical perspective. In *Proc. CEUR Workshop*, 2021.

R. Agarwal, L. Melnick, N. Frosst, X. Zhang, B. Lengerich, R. Caruana, and G. Hinton. Neural additive models: Interpretable machine learning with neural nets. In *Proc. Advances in Neural Information Processing Systems (Neurips)*, 2021.

V. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10), 2008.

A. Parchami-Araghi, M. Böhle, S. Rao, and B. Schiele. Good teachers explain: Explanation-enhanced knowledge distillation. In *Proc. European Conference on Computer Vision*, 2024.

L. Shapley. A value for n-person games. *Contributions to the Theory of Games (AM-28)*, 2:307–318, 1953.

B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016.

R. Alharbi, M. N. Vu, and M. T. Thai. Learning interpretation with explainable knowledge distillation. In *Proc. IEEE International Conference on Big Data (Big Data)*, 2021.

R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017.

M. McCloskey and N. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165, 1989.

S. Yun, D. Han, S. Oh, S. Chun, J. Choe, and Y. Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

J. Rojas. Universidad del Cauca traffic dataset, 2017. URL `https://www.kaggle.com/datasets/jsrojas/ip-network-traffic-flows-labeled-with-87-apps`.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *ArXiv preprint*, arXiv:1708.07747, 2017.

A. Krizhevsky. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

Y. Le and X. Yang. Tiny imagenet visual recognition challenge. *Stanford course CS 231N*, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

S. Zagoruyko and N. Komodakis. Wide residual networks. In *Proc. British Machine Vision Conference*, 2016.

T. Bonald, N. Lara, Q. Lutz, and B. Charpentier. Scikit-network: Graph analysis in python. *Journal of Machine Learning Research*, 21(185):1–6, 2020.

Supplementary Material

A Related Works

A.1 Knowledge Distillation

The idea of KD first appeared in Bucilă et al. [2006], where the authors showed that it is possible to distill the knowledge of an ensemble of machine learning models to a single model by matching logits. Subsequently, Hinton et al. [2015] popularized the technique to design low-complexity models. These models are often easy to deploy and provide faster inference.

There have been several extensions to the original KD framework. The authors of Lopez-Paz et al. [2016] unified Hinton's KD with the privileged teacher framework of Vapnik and Izmailov [2015] from a generalization error perspective. The authors of Mirzadeh et al. [2020] further proposed gradual KD through a teaching assistant. In Yuan et al. [2020], the authors argued that KD is equivalent to label smoothing regularization and proposed a teacher-free distillation framework where the student is self-taught or it learns from a designed regularization distribution. A data-free distillation method was introduced in Fang et al. [2021] based on contrastive model inversion, where a generative model was used to obtain synthetic data for distillation. The authors of Zhao et al. [2022] presented decoupled KD, enabling efficient distillation for target class and non-target classes. This work also demonstrated the efficacy of logit-based distillation. In Huang et al. [2022], the authors studied knowledge transfer from a stronger teacher using correlation-based distillation encouraging both intra-class and inter-class agreement between the teacher and the student. The authors in Zheng and Yang [2024] proposed a transformed teacher matching framework by removing the temperature scaling on the student side. In Zhang et al. [2018] and Lan et al. [2018], the authors investigated online KD, where the teacher and the student learn together mutually from each other. The authors in Dennis et al. [2023] presented a reversed version of original KD, where a student ensemble is built progressively to approximate a large teacher for fast inference.

None of these works consider teachers that explain the predictions that are taught to the students. In contrast, our work shows that substantial learning improvement can be achieved by a teacher that explains.

A.2 Hidden-Representation Distillation Methods

The hidden-representation distillation framework has been developed in parallel to KD. In this framework, the student attempts to match the teacher's intermediate layer outputs, instead of the teacher's logits or final inference outcomes as in KD. In Romero et al. [2015], the student is a thin model and its latent representation is aligned with the teacher's intermediate layer hints using a projector. In Zagoruyko and Komodakis [2017] and Tung and Mori [2019], the dependency in Romero et al. [2015] upon the projector is removed by attention transfer and similarity preservation over training samples. The authors of Kim et al. [2018] proposed paraphrasing a complex teacher by extracting so-called factors from the hidden layers. The authors of Tian et al. [2020] and Yang et al. [2021] proposed distillation via contrastive representation and softmax regression, respectively. In Park et al. [2019], the authors explored the distillation of relational knowledge between different samples as characterized by the teacher. An information-theoretic knowledge transfer scheme was introduced in Ahn et al. [2019] that maximizes mutual information between the teacher and the student. The authors of Heo et al. [2019] investigated the distillation of activation boundaries produced by the hidden neurons where a student matches the teacher's boundaries instead of the output of the layers. A structured knowledge distillation framework was proposed in Liu et al. [2019] for semantic segmentation. In Shu et al. [2021], the channel-wise probability maps are used for distillation so that the student learns the most salient parts of each channel for dense prediction. Cross-layer matching of hidden representations was studied in Chen et al. [2021a] and Chen et al. [2021b], which transfers knowledge via connection paths between different stages of the teacher and the student network. In Guo et al. [2023], the authors introduced a class-attention-based distillation framework by modifying the typical architecture of CNNs. The authors in Liu et al. [2023] designed a *feature transform* module that expands student's hidden representation into multiple non-overlapping segments for matching with the teacher. This module can be collapsed after training due to its linear property. In Wang et al. [2024], a cross-head distillation for object detection tasks was proposed by propagating student's hidden representation through teacher's head and thus improving model performance.

However, learning directly from the teacher's hidden layers may not be always effective. Since the student model usually has far lower complexity than the teacher, trying to match the teacher's hidden layers may distract from the student's final classification task. In contrast, KED constructs a teacher that offers explicit explanations on groups of features, and it allows a hierarchy of different numbers of superfeatures. Furthermore, a distinguishing feature of KED is that the explanations can be considered as Shapley values Shapley [1953] of a cooperative game. More importantly, as shown later, KED can be augmented with existing hidden-representation distillation methods in scenarios when they are effective.

A.3 Distillation and Interpretable AI

To design an explaining teacher, we must circumvent the general lack of interpretability in modern machine learning techniques. Interpretable AI deals with this problem by providing explanations for the model's predictions. This is often achieved through feature attribution. The authors of Ribeiro et al. [2016] proposed local interpretable model-agnostic explanation (LIME), which provides insights about the classifier's perception for a given sample. The authors of Lundberg and Lee [2017] unified existing feature attribution methods in the Shapley value-based framework and introduced low complexity algorithms for computing approximate Shapley values. The cooperative game defined in Štrumbelj and Kononenko [2014] was generalized in Chowdhury et al. [2019], Jullum et al. [2021] using the set of superfeatures as players and thus obtaining superfeature-based explanation. However, as detailed in Section 3.1, the explanations generated by the above methods are local to specific samples and hence not conducive to knowledge transfer in distillation.

Recently, a few explanation-based distillation schemes have appeared in the literature using class activation mapping (CAM) Zhou et al. [2016]. In Alharbi et al. [2021], the authors implemented convolutional autoencoder to approximate teacher's explanation. However, since the explanations are produced by different model, the efficacy of explanation is reduced. In Guo et al. [2023], the authors proposed class activation transfer between the teacher and the student. Another popular explanation method namely GradCAM Selvaraju et al. [2017] was used in Parchami-Araghi et al. [2024] for maintaining agreement between the teacher and the student. However, one key drawback of CAM and GradCAM-based explanations is that they are limited to images and particularly designed to extract insights about final predictions from CNN architecture. Furthermore, the computational complexity for obtaining these explanations adds considerable overhead to the distillation process. A new approach is needed to integrate interpretable AI and knowledge distillation in our KED framework.

B Relationship to Shapley values

An alternative interpretation of superfeature-explaining teachers can be obtained using cooperative games. A cooperative game $\mathcal{G} = \langle \mathcal{N}, v \rangle$ is defined by a set of players, $\mathcal{N}$, and a value function, $v : 2^{\mathcal{N}} \to \mathbb{R}$. The Shapley value Shapley [1953] of each player measures the average marginal contribution of the player to the game's outcome. Let $\mathcal{N}$ be the set of superfeatures in KED, and we define the value function for any coalition $\mathcal{S}$ as

$$v(\mathcal{S}) = \sum_{m \in \mathcal{S}} \log p(y|\mathbf{x}_m) - (M-1) \log p(y). \tag{9}$$

Note that the value of the grand coalition in this game is the total logit, i.e.,

$$v(\mathcal{N}) = \sum_{m=1}^{M} \log p(y|\mathbf{x}_m) - (M-1) \log p(y). \tag{10}$$

For any sample $\mathbf{x}$, the Shapley value of the m-th superfeature can be obtained as

$$q_m(\mathbf{x}_m, y) = \sum_{\mathcal{S} \subseteq \mathbf{x} \setminus \{\mathbf{x}_m\}} \frac{|\mathcal{S}|!(M - |\mathcal{S}| - 1)!}{M!} [v(\mathcal{S} \cup \{\mathbf{x}_m\}) - v(\mathcal{S})], \tag{11}$$

where $v(\mathcal{S} \cup \{\mathbf{x}_m\})$ indicates the value of the game when the m-th superfeature is included in the coalition $\mathcal{S}$. Substituting (9) in (11), the Shapley value of the m-th superfeature in this game is given

by

$$q_m(\mathbf{x}_m, y) = \sum_{\mathcal{S} \subseteq \mathbf{x} \setminus \{\mathbf{x}_m\}} \frac{|\mathcal{S}|!(M - |\mathcal{S}| - 1)!}{M!} \log p(y|\mathbf{x}_m) = \log p(y|\mathbf{x}_m). \tag{12}$$

Hence, we establish that the outputs of the superfeature-explaining teacher are Shapley values.

C Diagram of KED Framework

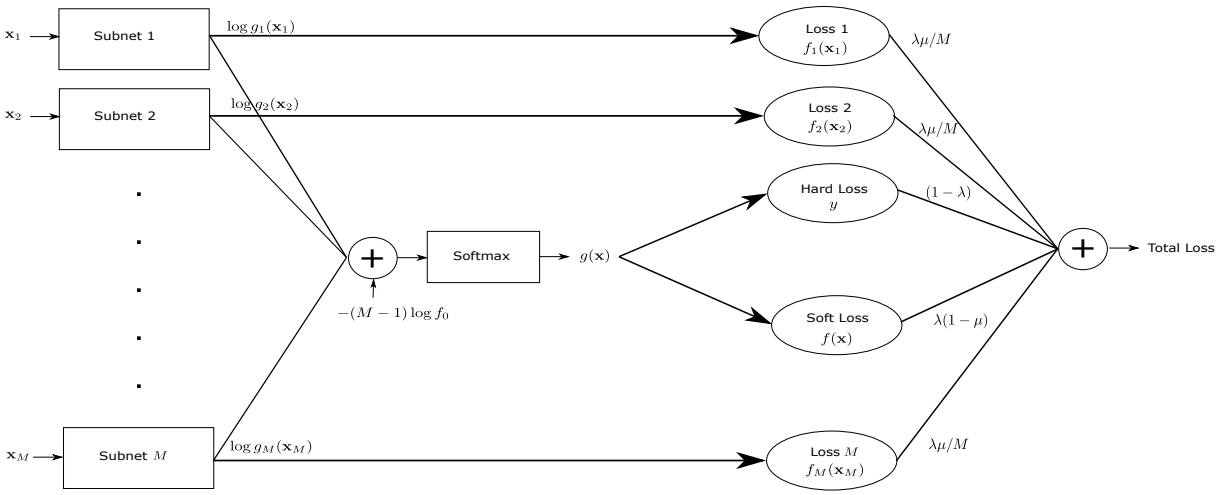


Figure 2: Student in KED.

D Extensions: CNN, Hidden-Representation and Chimeric Set

We explore three interesting extensions of the KED framework. For CNNs, our proposed technique can significantly improve the distillation performance. Furthermore, we show how KED can be augmented with hidden-representation distillation methods. Also, in scenarios with limited training data, the performance of KED can be enhanced using a chimeric set.

D.1 KED for CNNs

Since the pixels of the input images are usually unstructured, a CNN has to extract its own features. As mentioned in Section 3.4, if we can design the features such that they form independent groups, then we do not need any posterior process of constructing superfeatures. Thus, for building a type-M CNN-based superfeature-explaining teacher, it is enough to split the architecture into M subnets at some intermediate layer so that during training, the backpropagation algorithm can construct the superfeatures automatically. As the explanations from M subnets are combined at the output end using (4), reducing the training loss is equivalent to promoting independence among the superfeatures.

An efficient approach for designing low-complexity CNN student models is to split the architecture along the filter dimension (see Figure 3). Let F be the number of output filters of the shared convolutional layers. If we split it into M superfeatures, the input to each subnet will have only F/M filters. We remark that the modern CNN architectures, particularly Resnets, are narrow and thus splitting into M subnets often creates a bottleneck at each subnet for signal propagation. In such cases, we use a projector to increase the filter dimension before splitting into M superfeatures. A projector $r : \mathbb{R}^{H \times W \times F} \to \mathbb{R}^{H \times W \times F'}$ is a function with $F' > F$, where $H \times W$ is the spatial dimension of the shared convolutional layer output. We use a 1×1 convolutional layer as projector, which contains very few parameters. Such a linear projection also has whitening effect which decorrelates the filters and helps in promoting independence among the superfeatures. In Section 4, we will see that with this approach, the KED type-M teacher can achieve nearly identical classification accuracy as the original black-box CNN, leading to superior student performance.

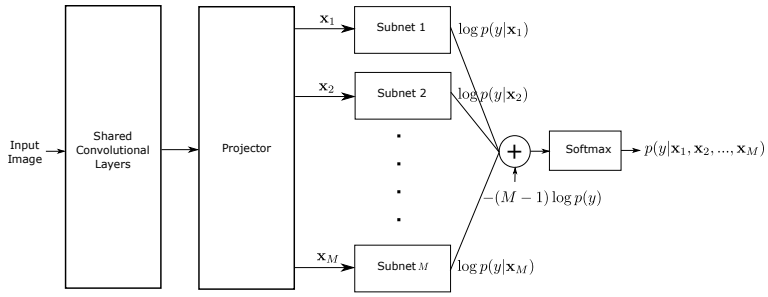


Figure 3: Superfeature-explaining teacher for CNNs.

D.2 KED and Hidden-Representation Distillation Methods

KED can be straightforwardly augmented with existing hidden-representation distillation methods. Let $h^g_m(\mathbf{x}_m)$, $1 \leq m \leq M$, be the student's representations that we want to align with the teacher's intermediate layer outputs $h^f_m(\mathbf{x}_m)$, $1 \leq m \leq M$. Adding the loss between the teacher's and the student's hidden representations, we rewrite (5) as

$$\mathcal{L}(g) = (1-\lambda)\mathbb{E}_{\mathcal{D}}[\mathcal{L}_{\text{ce}}(y, g(\mathbf{x}))] + T^2\lambda(1-\mu)\mathbb{E}_{\mathcal{D}}[\mathcal{L}_{\text{kl}}(\sigma_T(f(\mathbf{x})), \sigma_T(g(\mathbf{x})))]$$
$$+ \frac{\tau^2\lambda\mu(1-\rho)}{M}\sum_{m=1}^{M}\mathbb{E}_{\mathcal{D}}[\mathcal{L}_{\text{kl}}(\sigma_\tau(f_m(\mathbf{x}_m)), \sigma_\tau(g_m(\mathbf{x}_m)))] + \frac{\lambda\mu\rho}{M}\sum_{m=1}^{M}\mathbb{E}_{\mathcal{D}}[\mathcal{L}_{\text{h}}(h^f_m(\mathbf{x}_m), h^g_m(\mathbf{x}_m))], \quad (13)$$

where ρ is an additional hyperparameter. The loss term $\mathcal{L}_{\text{h}}$ is generic and can be set according to most existing hidden-representation distillation methods, e.g., FitNet Romero et al. [2015], attention transfer Zagoruyko and Komodakis [2017], and similarity-preserving distillation Tung and Mori [2019].

D.3 Out-of-Distribution KED using Chimeric Set

In scenarios where only a small training dataset is available, it is important to regularize the student. We propose to use the chimeric set as a natural technique for performance enhancement. For a type-M teacher-student pair, let us rewrite the training dataset $\mathcal{X}$ as $\mathcal{X} = \{\mathcal{X}_1, ..., \mathcal{X}_M\}$ where $\mathcal{X}_m$ denotes the set of m-th superfeatures among all training samples. Note that $|\mathcal{X}_m| = |\mathcal{X}| = K$. Then, we define the chimeric set as the M-fold Cartesian product $\mathcal{X}^M = \mathcal{X}_1 \times ... \times \mathcal{X}_M$. Thus, $|\mathcal{X}^M| = K^M$. This set contains only K in-distribution samples and the rest are out-of-distribution (OOD) samples.

Given a type-M teacher's explanations for all the samples in the training dataset $\mathcal{X}$, we know the contribution of superfeatures in $\mathcal{X}_m, \forall m$. Therefore, we can calculate the teacher's explanations and predictions for all the samples in the chimeric set. Note that the chimeric samples do not have any true labels. However, to apply the framework of KED, we still treat the teacher's predictions converted to hard labels as if they were true labels. Then we can train a type-M student over the chimeric set as described in Section 3.3.

We remark that, for KD with a black-box teacher, training on the chimeric set may result in "catastrophic forgetting" McCloskey and Cohen [1989] for black-box students, i.e., seeing OOD samples, the student may forget the information learned from the in-distribution samples. In contrast, for KED students, the chimeric set serves as a special data augmentation technique. This is different from other data augmentation method such as CutMix Yun et al. [2019], particularly since here we take advantage of the architecture of type-M students. When being trained on the chimeric set, a type-M student always sees in-distribution superfeatures even in OOD samples. In other words, it learns from the same explanations but different predictions.

E Further Experiments

E.1 Datasets and Experimental Setup

We run experiments on classification over various datasets, including the Unicauca network traffic dataset Rojas [2017], MNIST LeCun et al. [1998], FashionMNIST Xiao et al. [2017], CIFAR10

Table 2: Test accuracy for CIFAR100 with varying model architectures.

Teacher	Resnet44x2	Resnet56x2	Resnet56x2	WRN-16-8	WRN-16-8	WRN-28-4	WRN-28-4	VGG13
Black-box	75.11±0.06%	75.48±0.06%	75.48±0.06%	77.89±0.05%	77.89±0.05%	76.73±0.06%	76.73±0.06%	72.98±0.06%
Type-M	74.96±0.06%	75.17±0.06%	75.17±0.06%	78.17±0.05%	78.17±0.05%	76.98±0.06%	76.98±0.06%	71.86±0.06%
Student	WRN-10-2	Resnet20	WRN-10-2	Resnet20	WRN-16-2	Resnet20	VGG8	VGG8
No Distl.	69.01±0.06%	68.30±0.07%	69.01±0.06%	68.30±0.07%	72.25±0.06%	68.30±0.07%	68.72±0.06%	68.72±0.06%
KD	69.47±0.06%	69.19±0.06%	69.59±0.06%	69.29±0.07%	74.34±0.06%	69.66±0.06%	70.39±0.06%	69.95±0.06%
e^2-KD	69.55±0.06%	69.32±0.06%	69.76±0.07%	69.55±0.06%	74.51±0.06%	69.78±0.06%	70.43±0.06%	70.16±0.06%
KED	71.33±0.06%	70.94±0.06%	71.12±0.07%	70.89±0.06%	74.94±0.06%	**71.36±0.06%**	**73.50±0.06%**	71.30±0.06%
KD+FitNet	69.91±0.06%	69.42±0.07%	69.98±0.07%	69.36±0.06%	74.37±0.06%	69.60±0.06%	70.89±0.07%	69.98±0.06%
KED+FitNet	71.36±0.06%	**71.79±0.07%**	71.28±0.06%	**71.39±0.06%**	75.09±0.06%	71.18±0.06%	72.94±0.06%	70.92±0.06%
KD+AT	69.53±0.06%	69.24±0.06%	69.50±0.06%	69.04±0.06%	73.82±0.06%	69.02±0.06%	70.29±0.07%	69.27±0.07%
KED+AT	70.84±0.07%	70.38±0.06%	70.57±0.06%	70.98±0.06%	74.57±0.06%	70.75±0.06%	72.61±0.06%	70.84±0.06%
KD+SP	69.83±0.06%	69.57±0.06%	69.35±0.06%	68.69±0.07%	73.68±0.06%	68.79±0.06%	69.87±0.06%	70.40±0.06%
KED+SP	**71.42±0.06%**	70.81±0.06%	71.23±0.06%	70.52±0.06%	74.67±0.06%	71.19±0.06%	72.07±0.06%	**71.67±0.06%**
KD+DKD	69.87±0.06%	69.58±0.06%	69.85±0.07%	69.07±0.07%	74.40±0.06%	69.26±0.06%	70.67±0.06%	70.02±0.06%
KED+DKD	71.40±0.06%	70.93±0.06%	**71.43±0.06%**	70.94±0.06%	**75.26±0.06%**	71.22±0.06%	72.85±0.06%	71.42±0.06%
KD+DIST	69.46±0.06%	69.31±0.06%	69.52±0.06%	68.91±0.07%	74.31±0.06%	69.17±0.07%	70.56±0.06%	69.93±0.06%
KED+DIST	71.32±0.06%	70.56±0.06%	71.05±0.06%	70.67±0.06%	75.02±0.06%	71.31±0.06%	72.91±0.06%	71.27±0.06%

Table 3: Test accuracy for Tiny Imagenet with varying model architectures.

Teacher	Resnet44x2	Resnet56x2	Resnet56x2	WRN-16-8	WRN-16-8	WRN-28-4	WRN-28-4	VGG13
Black-box	61.04±0.07%	61.09±0.07%	61.09±0.07%	63.46±0.07%	63.46±0.07%	62.60±0.07%	62.60±0.07%	59.32±0.07%
Type-M	60.79±0.07%	60.80±0.07%	60.80±0.07%	63.35±0.07%	63.35±0.07%	61.99±0.07%	61.99±0.07%	58.88±0.07%
Student	WRN-10-2	Resnet20	WRN-10-2	Resnet20	WRN-16-2	Resnet20	VGG8	VGG8
No Distl.	49.39±0.08%	51.84±0.07%	49.39±0.08%	51.84±0.07%	56.13±0.07%	51.84±0.07%	53.50±0.07%	53.50±0.07%
KD	50.52±0.07%	52.72±0.07%	51.26±0.07%	52.88±0.07%	57.38±0.07%	52.47±0.07%	57.55±0.07%	56.34±0.06%
e^2-KD	51.73±0.07%	53.36±0.07%	52.47±0.07%	52.73±0.07%	57.84±0.07%	53.06±0.07%	58.25±0.07%	56.71±0.06%
KED	52.98±0.07%	54.96±0.07%	53.18±0.07%	53.87±0.07%	58.72±0.07%	54.94±0.07%	60.42±0.07%	59.06±0.07%
KD+FitNet	53.01±0.07%	53.71±0.07%	53.35±0.07%	52.97±0.07%	59.41±0.07%	53.27±0.07%	58.55±0.07%	56.28±0.07%
KED+FitNet	54.79±0.07%	54.90±0.07%	**55.27±0.07%**	54.43±0.07%	**60.23±0.07%**	55.40±0.07%	60.74±0.07%	58.69±0.07%
KD+AT	51.96±0.07%	53.32±0.07%	52.33±0.07%	52.22±0.07%	59.14±0.07%	53.04±0.07%	57.79±0.07%	54.63±0.07%
KED+AT	53.82±0.07%	**55.35±0.08%**	54.02±0.07%	54.59±0.07%	59.93±0.07%	55.10±0.07%	**60.99±0.07%**	58.59±0.07%
KD+SP	52.08±0.07%	52.59±0.07%	52.25±0.07%	52.49±0.07%	58.44±0.07%	52.75±0.07%	57.35±0.07%	58.52±0.07%
KED+SP	54.46±0.07%	54.75±0.08%	54.36±0.08%	53.59±0.07%	59.70±0.07%	54.38±0.07%	60.14±0.07%	**59.47±0.07%**
KD+DKD	52.31±0.07%	53.42±0.07%	52.29±0.07%	52.35±0.07%	58.65±0.07%	52.83±0.07%	57.65±0.07%	56.81±0.07%
KED+DKD	**54.93±0.07%**	54.91±0.07%	54.48±0.07%	54.47±0.07%	59.87±0.07%	55.16±0.07%	60.53±0.07%	59.12±0.07%
KD+DIST	52.03±0.07%	53.24±0.07%	52.37±0.07%	52.31±0.07%	58.53±0.07%	52.92±0.07%	57.81±0.07%	57.37±0.07%
KED+DIST	54.62±0.07%	54.83±0.08%	54.62±0.07%	**54.73±0.07%**	59.66±0.07%	**55.71±0.07%**	60.47±0.07%	58.26±0.07%

Krizhevsky [2009], CIFAR100 Krizhevsky [2009], and Tiny Imagenet Le and Yang [2015]. We implement multi-layer perceptrons (MLPs), Resnets He et al. [2016], wide Resnets Zagoruyko and Komodakis [2016], and VGGs for the teacher and student models.

For the Unicauca dataset, we group the network traffic application types into three delay classes and categorize flows according to their delay sensitivity. In KD, the teacher has size [200, 200, 200, 200], i.e., four hidden layers with 200 neurons each. The architecture of the student is [50, 50, 50, 50]. For MNIST and FashionMNIST, the KD teacher has size [500, 500]. The black-box student in KD has size [20, 20] for MNIST and [60, 60] for FashionMNIST. For KED, we construct the superfeature-explaining teacher and student by choosing the number of neurons per hidden layer such that they have a similar number of total model parameters as the benchmark KD

Table 4: Test accuracy for Unicauca with varying student's training dataset size.

$\lvert\mathcal{X}\rvert$	Teacher		Student		
	Black-box	Type-M	No Distl.	KD	KED
10000	76.51±0.04%	77.33±0.03%	66.78±0.04%	67.51±0.04%	**68.99±0.04%**
30000			70.50±0.04%	69.64±0.04%	**72.74±0.04%**
50000			71.70±0.04%	70.43±0.04%	**74.15±0.04%**
70000			72.88±0.04%	71.47±0.04%	**74.76±0.04%**
90000			73.03±0.04%	72.31±0.04%	**75.96±0.03%**
108158 (Full)			74.12±0.04%	72.29±0.04%	**75.84±0.04%**

Table 5: Test accuracy for MNIST with varying student's training dataset size.

$\lvert\mathcal{X}\rvert$	Teacher		Student		
	Black-box	Type-M	No Distl.	KD	KED
10000	98.43±0.02%	98.40±0.02%	93.32±0.03%	95.15±0.03%	**96.87±0.02%**
20000			94.31±0.04%	95.64±0.03%	**97.28±0.02%**
30000			94.81±0.03%	96.33±0.03%	**97.32±0.02%**
40000			95.32±0.03%	96.37±0.03%	**97.44±0.02%**
50000			95.49±0.03%	96.78±0.03%	**97.63±0.02%**
60000 (Full)			95.41±0.03%	96.69±0.02%	**97.52±0.02%**

with a black-box teacher. In particular, we note that a type-M model with L hidden layers has $M(L-1)n_h^2 + (ML + MC + d)n_h + MC$ parameters where n_h is the number of neurons per hidden layer. Thus, we set this expression equal to the number of model parameters in black-box KD and solve for n_h.

For CIFAR10, CIFAR100 and Tiny Imagenet, we use standard Resnets, wide Resnets and VGGs. We use the labels "Resnet$n$xk," "WRN-n-k," and "VGGn" where the variables n and k denote the number of layers and the widening factor, respectively. For KED, type-M models of similar complexity are constructed by splitting the last stack of convolutional and fully connected layers in the aforementioned models.

For all experiments, the default setting for the hyperparameters is $T = 10$, $\tau = 10$, $\lambda = 0.7$, $\mu = 0.7$, and $\rho = 0.7$. We keep the number of superfeatures fixed at $M = 4$. For the Unicauca, MNIST, and FashionMNIST datasets, the teachers and the students, respectively, are trained for 100 epochs with a batch size of 500 and 100. We use RELU activation and the Adam optimizer with a learning rate of 0.001. The prior for type-M models $p(y)$ is estimated by taking a sample average of the black-box teacher's predictions over the training set. For picking superfeatures, we apply Algorithm 1 computing the average Hessian of the black-box teacher over a random subset of 1000 training samples. We use the Louvain community detection method in scikit-network version 0.27.1 Bonald et al. [2020]. We tune the resolution with a stepsize of 0.01. For CIFARs and Tiny Imagenet, we train the teachers and the students for 150 and 75 epochs, respectively, with a batch size of 100. For data augmentation, we pad 4 pixels on each side of a training image, and then apply random crop and random horizontal flip to it. We use RELU activation and the SGD optimizer with Nesterov momentum 0.9. The initial learning rate is set to 0.01 for the first 10 epochs as warm-up period. Then, for CIFARs, we use a learning rate of 0.05 and divide it by 10 after 120 and 140 epochs. The L2 regularization coefficient is set to 5×10^{-4}. For Tiny Imagenet, we use a learning rate of 0.1 and divide it by 10 after 60 and 70 epochs. The L2 regularization coefficient is set to 1×10^{-4}. As described in Section D.1, for CNNs, the superfeatures are created from the output of a projector, which is a 1x1 convolutional layer. We assume uniform prior over the classes and thus eliminate $p(y)$ from (4).

For the experiments on the chimeric set, we randomly generate one million samples and continue training the distilled students for 5 epochs with a batch size of 100. To avoid numerical instability, we have added a small bias of 10^{-15} to the argument of the $\log$ function. In all experiments, we obtain a 95% confidence intervals for inference by bootstrapping the test set.

Table 6: Test accuracy for FashionMNIST with varying student's training dataset size.

$\|\mathcal{X}\|$	Teacher		Student		
	Black-box	Type-M	No Distl.	KD	KED
10000	89.98±0.04%	90.16±0.04%	84.86±0.05%	85.31±0.05%	**87.50±0.05%**
20000			85.98±0.04%	86.78±0.04%	**88.58±0.04%**
30000			86.69±0.04%	87.38±0.04%	**88.41±0.05%**
40000			87.18±0.05%	87.49±0.05%	**88.79±0.04%**
50000			87.88±0.05%	87.47±0.05%	**88.88±0.05%**
60000 (Full)			87.96±0.05%	88.11±0.05%	**89.38±0.04%**

Table 7: Test accuracy for Unicauca using chimeric set while varying the student's training dataset size.

Student's training dataset size	KED	KED with chimeric set	KD	KD with chimeric set	No distillation
2000	63.61% ±0.04%	65.51% ±0.04%	60.56% ±0.04%	59.03% ±0.04%	59.95% ±0.04%
4000	66.63% ±0.04%	68.74% ±0.04%	62.92% ±0.04%	59.90% ±0.04%	62.83% ±0.04%
6000	68.07% ±0.04%	69.79% ±0.04%	65.49% ±0.04%	61.31% ±0.04%	64.55% ±0.04%
8000	69.37% ±0.04%	71.10% ±0.04%	66.22% ±0.04%	61.19% ±0.04%	66.42% ±0.04%

E.2 Distillation Performance with MLPs

Table 4 presents the test accuracy of KED for the Unicauca dataset varying the student's training dataset size. We observe that KED substantially outperforms KD and no distillation. For example, the student with 10000 samples in the training dataset achieves 66.78% and 67.51% accuracy without and with KD, respectively. The KED student achieves an accuracy of 68.99%. On the full dataset, the black-box student achieves 74.12% and 72.29% accuracy without and with KD, respectively, whereas the KED student reaches an accuracy of 75.84%. In this case, KD fails to improve the performance of the student. In contrast, KED still benefits the student significantly.

We show further experimental results for the MNIST and FashionMNIST datasets in Tables 5 and 6, respectively. For MNIST, with 10000 samples in the training dataset, the black-box student achieves an accuracy of 93.32%, which increases to 95.15% under KD. In comparison, the KED student achieves an improved accuracy of 96.87%. When trained on the full dataset, the black-box student provides 95.41% and 96.69% accuracy without and with KD, respectively. The KED student reaches an accuracy of 97.52%.

We see a similar trend for the FashionMNIST dataset as well. Training the black-box student using 10000 samples results in a test accuracy of 84.86% and 85.31% under no distillation and KD, respectively. In comparison, the KED student attains 87.50% accuracy. Furthermore, when trained on the full dataset, the black-box student provides an accuracy of 87.96% and 88.11% without and with KD, respectively. KED enables the student to achieve 89.38% accuracy.

In all of these experiments, we observe that the accuracy gap between the KED teacher and the KED student is much narrower compared with their black-box KD counterparts. This clearly demonstrates that a superfeature-explaining teacher can transfer more knowledge than a black-box teacher of similar complexity.

E.3 Benefits of Chimeric Set

We show the performance of knowledge distillation using the chimeric set in Tables 7, 8, and 9. We run these experiments with fewer than 10000 samples in the student's training dataset to illustrate the impact of the chimeric set in a limited-data scenario. For each of these experiments, the student is first trained via the regular KD or KED and then we continue training it on the chimeric set. We observe that for many cases in MNIST and FashionMNIST, the chimeric set improves the performance of both KD and KED. However, for Unicauca, only KED benefits significantly from the chimeric set.

Table 8: Test accuracy for MNIST using chimeric set while varying the student's training dataset size.

Student's training dataset size	KED	KED with chimeric set	KD	KD with chimeric set	No distillation
2000	95.23% ±0.03%	96.26% ±0.02%	91.38% ±0.04%	92.69% ±0.04%	90.27% ±0.04%
4000	96.01% ±0.03%	96.92% ±0.02%	93.89% ±0.04%	93.07% ±0.03%	91.72% ±0.04%
6000	96.57% ±0.02%	96.92% ±0.02%	94.41% ±0.03%	93.24% ±0.04%	92.63% ±0.04%
8000	96.91% ±0.02%	96.98% ±0.02%	94.62% ±0.04%	90.44% ±0.04%	93.15% ±0.04%

Table 9: Test accuracy for FashionMNIST using chimeric set while varying the student's training dataset size.

Student's training dataset size	KED	KED with chimeric set	KD	KD with chimeric set	No distillation
2000	84.89% ±0.05%	85.44% ±0.05%	82.52% ±0.06%	84.60% ±0.05%	81.14% ±0.05%
4000	86.26% ±0.05%	87.27% ±0.05%	83.32% ±0.05%	84.77% ±0.05%	83.58% ±0.06%
6000	87.46% ±0.05%	87.91% ±0.05%	84.45% ±0.05%	84.53% ±0.05%	84.17% ±0.06%
8000	87.51% ±0.05%	88.16% ±0.04%	84.86% ±0.05%	84.76% ±0.05%	84.53% ±0.06%

E.4 Impact of Hyperparameter Values and Ablation Studies

We conduct further experiments to study the impact of the hyperparameters M, T, τ, λ and μ on KED and to evaluate the performance of Algorithm 1. We perform these experiments on Unicauca and MNIST with 10000 samples in the student's training dataset. We also provide ablation studies on CIFAR10, CIFAR100, and Tiny Imagenet.

E.4.1 Impact of the Number of Superfeatures, M

We demonstrate the impact of the number of superfeatures on KED. In Table 10, we show type-M teacher's and type-M student's test accuracy on Unicauca and MNIST varying the number of superfeatures M. Increasing M has two opposing effects on the student's performance. On one hand, a larger M implies more explanation from the teacher. On the other hand, both the teacher's and the student's architecture becomes more restricted with respect to modeling the dependency among the superfeatures. In particular, we observe some degradation in the teacher's performance as M increases. Thus, the student's performance improves with increasing M only when M is not too large.

Table 10: Test accuracy of KED on Unicauca and MNIST, varying the number of superfeatures M.

# Superfeatures	Unicauca		MNIST	
	Teacher	Student	Teacher	Student
$M = 2$	76.80±0.04%	68.15±0.04%	98.31±0.02%	96.49±0.02%
$M = 4$	77.33±0.03%	68.99±0.04%	98.40±0.02%	96.87±0.02%
$M = 8$	76.71±0.03%	70.13±0.04%	98.18±0.02%	96.67±0.03%
$M = 16$	76.23±0.04%	70.09±0.04%	97.94±0.02%	96.65±0.02%

E.4.2 Impact of Temperature Parameters, T and τ

In Tables 11 and 12, we present the impact of temperature parameters T and τ on Unicauca and MNIST, respectively. For Unicauca, the accuracy varies between 68.34% at $T = 20$ and $\tau = 10$, and 70.45% at $T = 1$ and $\tau = 1$. For MNIST, the minimum accuracy of 96.21% is observed at $T = 20$ and $\tau = 1$, and the maximum accuracy of 97.21% is achieved at $T = 10$ and $\tau = 20$. We note that the choice of distillation temperature significantly affects the performance in both cases.

Table 11: Test accuracy of KED student on Unicauca, varying the temperature parameters, T and τ.

Temperature	$T = 1$	$T = 5$	$T = 10$	$T = 20$
$\tau = 1$	70.45±0.04%	69.65±0.04%	68.56±0.04%	68.86±0.04%
$\tau = 5$	70.14±0.04%	69.24±0.04%	69.34±0.04%	68.52±0.04%
$\tau = 10$	70.09±0.04%	69.39±0.04%	68.99±0.04%	68.34±0.04%
$\tau = 20$	69.77±0.04%	69.49±0.04%	68.82±0.04%	68.77±0.04%

Table 12: Test accuracy of KED student on MNIST, varying the temperature parameters, T and τ.

Temperature	$T = 1$	$T = 5$	$T = 10$	$T = 20$
$\tau = 1$	96.59±0.03%	96.37±0.03%	96.49±0.02%	96.21±0.02%
$\tau = 5$	96.61±0.02%	96.71±0.02%	96.95±0.02%	96.39±0.02%
$\tau = 10$	96.85±0.02%	96.92±0.02%	96.87±0.02%	96.68±0.02%
$\tau = 20$	96.72±0.03%	97.02±0.02%	97.21±0.02%	96.75±0.02%

E.4.3 Impact of Loss Weights, λ and μ

We observe the impact of loss weights λ and μ in Tables 13 and 14, on Unicauca and MNIST, respectively. For Unicauca, the minimum accuracy of 67.90% is observed at $\lambda = 0.8$ and $\mu = 0.2$, and the maximum accuracy of 69.66% is achieved at $\lambda = 0.2$ and $\mu = 0.6$. For MNIST, the accuracy varies between 96.39% at $\lambda = 0.6$ and $\mu = 0.2$, and 97.03% at $\lambda = 0.6$ and $\mu = 0.8$. We note that the choice of loss weights significantly affects the performance in Unicauca but not as much in MNIST.

Table 13: Test accuracy of KED student on Unicauca, varying the loss weights, λ and μ.

Loss weights	$\lambda = 0.2$	$\lambda = 0.4$	$\lambda = 0.6$	$\lambda = 0.8$
$\mu = 0.2$	69.29±0.04%	68.83±0.04%	68.41±0.04%	67.90±0.04%
$\mu = 0.4$	69.09±0.04%	68.81±0.04%	68.75±0.04%	68.53±0.04%
$\mu = 0.6$	69.66±0.04%	68.96±0.04%	69.35±0.04%	68.55±0.04%
$\mu = 0.8$	69.41±0.04%	69.33±0.04%	69.05±0.04%	68.96±0.04%

Table 14: Test accuracy of KED student on MNIST, varying the loss weights, λ and μ.

Loss weights	$\lambda = 0.2$	$\lambda = 0.4$	$\lambda = 0.6$	$\lambda = 0.8$
$\mu = 0.2$	96.40±0.02%	96.56±0.03%	96.39±0.03%	96.43±0.03%
$\mu = 0.4$	96.49±0.03%	96.79±0.02%	96.72±0.03%	96.83±0.03%
$\mu = 0.6$	96.65±0.03%	96.83±0.02%	96.67±0.03%	96.83±0.02%
$\mu = 0.8$	96.89±0.02%	96.74±0.02%	97.03±0.02%	96.92±0.02%

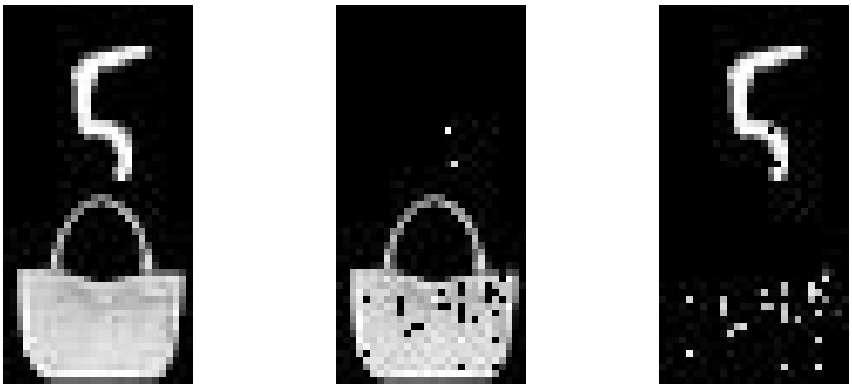
E.4.4 Ablation Studies

In Table 15, we show the impact of the KED teacher's prediction and explanation on the performance of the KED student. Note that even in the absence of the teacher's explanation, the KED student can learn the contribution of superfeatures just from the teacher's prediction. This is because of the superfeature-explaining architecture of both the teacher and the student. However, only when teacher's prediction and explanation are provided simultaneously, the student achieves the best performance.

Table 15: Ablation studies with KED student (Teacher: WRN-28-4, Student: Resnet8 / Resnet20)

Loss weights	Teacher's Prediction	Teacher's Explanation	CIFAR10	CIFAR100	Tiny Imagenet
$\lambda = 0.7, \mu = 0.0$	✓	✗	89.15±0.04%	70.57±0.06%	54.82±0.06%
$\lambda = 0.7, \mu = 1.0$	✗	✓	88.97±0.04%	69.35±0.06%	53.43±0.07%
$\lambda = 0.7, \mu = 0.7$	✓	✓	89.38±0.04%	71.36±0.06%	54.94±0.07%

E.5 Evaluation of Algorithm 1 on MNIST+FashionMNIST Combined Dataset



(a) Image (b) Superfeature 1 (c) Superfeature 2

Figure 4: Detecting two independent superfeatures in a combined image created from MNIST and FashionMNIST datasets.

In addition to the above experiments to test the performance of the overall KED framework, here we present a special experiment to directly evaluate the superfeature construction method in Algorithm 1. We combine images and their labels from MNIST and FashionMNIST datasets to create a new dataset MNIST+FashionMNIST with 1568 features and 100 classes. For simplicity of illustration, let's say MNIST has 10 classes, namely '0',..., '9', and FashionMNIST has 10 classes, namely 'A',..., 'J'. Then the generated MNIST+FashionMNIST dataset has 100 classes, namely '0A',..., '9J'. In this combined dataset, we know the natural superfeature composition: there are two superfeatures, one containing MNIST features, and the other containing FashionMNIST features. In Figure 4, we show an example of the combined image and the superfeatures constructed by Algorithm 1. Note that this is different from image segmentation since our algorithm does not utilize pixel arrangement information and thus will discover the superfeatures even if the pixels of the combined image are randomly shuffled. Our experimental setup for this dataset is the same as that for MNIST in Section E.1.

Table 16 shows the distillation performance using the actual superfeatures, the superfeatures constructed by Algorithm 1, and two sets of randomly generated superfeatures. We observe that the superfeatures constructed by Algorithm 1 achieve similar performance as the actual superfeatures. Furthermore, the superfeatures constructed by Algorithm 1 provides significant performance gain over random groupings.

Table 16: Performance of Algorithm 1 on MNIST+FashionMNIST combined dataset.

Superfeature Design	Teacher		Student		
	Black-box	Type-M	No Distl.	KD	KED
Actual	82.55±0.05%	85.83±0.05%	75.09±0.07%	76.31±0.06%	82.82±0.06%
Algorithm 1		85.31±0.05%			81.89±0.06%
Random 1		82.14±0.06%			77.28±0.06%
Random 2		82.06±0.05%			76.57±0.07%