A Appendix

A.1 A. Implementation Details

A.1.1 Hyperparameters

* **CIFAR-100** Krizhevsky et al. (2009) is widely used for image classification tasks and is publicly available. It contains 100 classes and the samples have an image size of 32×32. The dataset comprises of 50000 images in the training set and 10000 images in the test set. In our experiments, batch sizes of 64 and 480 epochs were used. The learning rate began at 0.05 and reduced by 0.1 at every 150, 180, 210, 240, 270, and 300 epochs. In addition, Tables 1 and 2 outline the hyperparameters utilized for CSKD.

* **ImageNet** Russakovsky et al. (2015) is a massive image classification dataset that contains 1000 classes. The samples are of size 224×224, and the dataset comprises of 1.28 million images in the training set and 5000 images in the test set. In our experiments, batch size of 512 and 200 epochs were used. The learning rate began at 0.1 and decreased by 0.1 at every 30, 60, 90, and 120, 150 epochs. Morevoer, table 3 provides details about the hyperparameters applied for CSKD.

A.1.2 Computing resource

The experimental server used in the study had the following hardware specifications: (CPU) AMD EPYC 7742; (GPU) NVIDIA RTX 3090; and (RAM) 1000GB. The experiments were conducted using CUDA version 11.3 and PyTorch version 1.11.0.

Student	**WRN-16-2**	**WRN-40-1**	**ResNet20**	**ResNet32**	**ResNet8x4**	**VGG8**
T_{max}	6.0	6.0	6.0	6.0	6.0	6.0
T_{min}	2.0	2.0	2.0	2.0	2.0	2.0
α	2.0	2.0	2.0	2.0	2.0	2.0
Teacher	**WRN-40-2**	**WRN-40-2**	**ResNet56**	**ResNet110**	**ResNet32x4**	**VGG13**

Table 1: Hyperparameters used in homogeneous model experiments using the CIFAR-100 dataset. Our framework, CSKD, utilizes the logit distillation with the same settings of Multi-KD Jin et al. (2023)

A.2 Training time and Temperature sensitivity

The left side of Fig. 1 illustrates the relationship between training time and accuracy for various knowledge distillation (KD) methods. In this experiment, ResNet32x4 served as the teacher model, while ResNet8x4 was the student model. CIFAR-100 was the dataset employed. Training time refers to the batch processing speed per batch, i.e., the processing of 64 image samples in each batch. As depicted in the figure, feature-based KD tends to be slower. Our approach, while it demands a higher processing speed compared to feature-based KD methods, outperforms other existing KD techniques, including logits- and features-based methods, by a significant margin.

We employ both a fixed temperature of 4 and our novel temperature scaling technique, which relies on cosine similarity values, consistently throughout the entire experiment. In this section, we demonstrate the stability of accuracy as we alter the fixed temperature of 4. As indicated on the right side of Fig. 1, we observe that accuracy remains relatively consistent. This signifies that our temperature scaling method can provide sufficient information to encompass a broad spectrum of temperatures.

A.3 Additional explanation of constraints

As illustrated by the example samples depicted in Fig. 2 with the blue and red boxes, increasing the batch size adds more equations to be satisfied (sum=1), and the upper and lower bounds on

Student	**ShuffleNet-V1**	**MobileNet-V2**	**ShuffleNet-V1**	**ShuffleNet-V2**	**MobileNet-V2**
T_{max}	6.0	6.0	6.0	6.0	6.0
T_{min}	2.0	2.0	2.0	2.0	2.0
α	2.0	2.0	2.0	2.0	2.0
Teacher	**WRN-40-2**	**ResNet50**	**ResNet32X4**	**ResNet32X4**	**VGG13**

Table 2: Hyperparameters used in heterogeneous model experiments using the CIFAR-100 dataset. Our framework, CSKD, utilizes the logit distillation with the same settings of Multi-KD Jin et al. (2023).

Student	**ResNet-18**	**MobileNetV1**
T_{max}	4.0	6.0
T_{min}	2.0	2.0
α	1.0	1.0
Teacher	**ResNet-34**	**ResNet-50**

Table 3: The hyperparameters used in the experiments for ImageNet. Our framework CSKD utilizes the advanced logit distillation with the same settings of Multi-KD Jin et al. (2023).

k are formed strictly. An experimental verification of this hypothesis using different batch sizes is presented in the Experimental section, providing experimental results for the claim that our approach using cosine similarity guarantees a variety of potential student predictions.

REFERENCES

Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Distilling knowledge via knowledge review. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5008–5017, 2021.

Ying Jin, Jiaqi Wang, and Dahua Lin. Multi-level logit distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24276–24285, 2023.

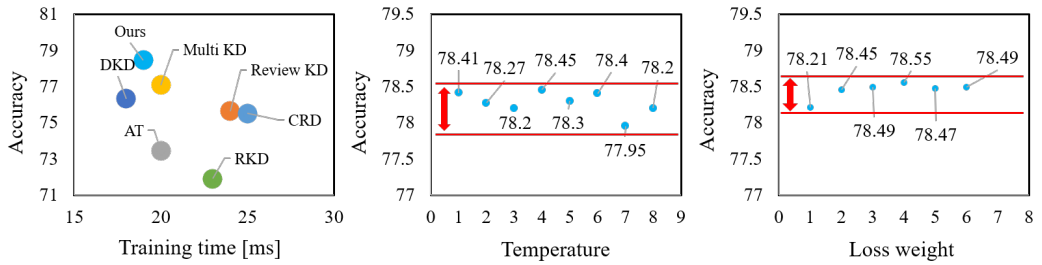
Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled knowledge distillation. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pp. 11953–11962, 2022.

Teacher Student	ResNet32x4 ResNet8x4	VGG13 VGG8	VGG13 MobileNetV2	ResNet32x4 ShuffleNetV1	ResNet32x4 ShuffleNetV2	ResNet50 MobileNetV2
T_{max}	6.0	6.0	6.0	6.0	6.0	6.0
T_{min}	2.0	2.0	2.0	2.0	2.0	2.0
α	2.0	2.0	2.0	2.0	2.0	2.0

Table 4: Orthogonality of ours methods with ReviewKD on CIFAR-100 datasets. ReviewKD settings followed Chen et al. (2021); Zhao et al. (2022).



(a) Accuracy-Time (b) Temperature sensitivity (c) Weight (α) sensitivity

Figure 1: The figure shows the training time (a) and the robustness of temperature sensitivity (b) and loss weight (c). The analysis was performed utilizing the ResNet32x4-ResNet8x4 architecture for the model.

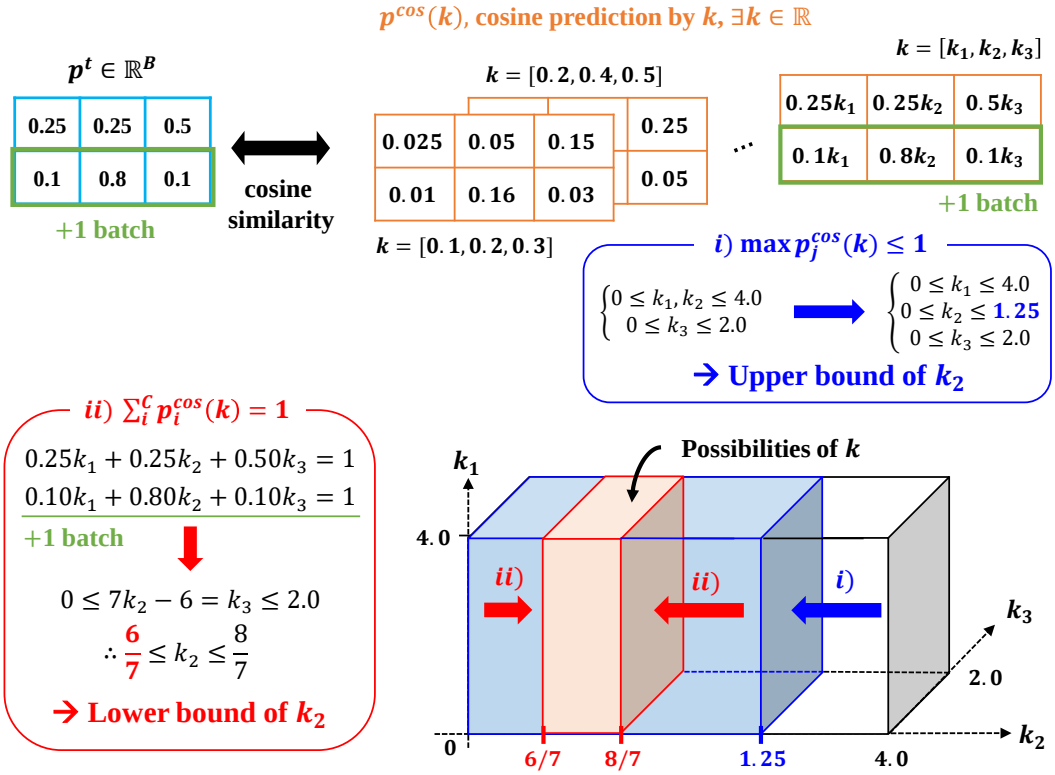


Figure 2: The figure shows a matrix table to explain the constraints.

Algorithm 1 Pseudo code for CSKD.

```
# x: input images
# model_t: teacher model
# model_s: student model
# K: temperature for KD (i.e. K=4)
# Ti: CSWT
# loss_cosine : cosine distance

o_t = model_t(x)
o_s = model_s(x)



# Probabilities
p_s = F.softmax(o_s/K, dim=1)
p_t = F.softmax(o_t/K , dim=1)
p_s_tr = p_s.transpose(1,0)
p_t_tr = p_t.transpose(1,0)

p_s_Ti = F.softmax(o_s/Ti, dim=1)
p_t_Ti = F.softmax(o_t/Ti , dim=1)
p_s_Ti_tr = p_s_Ti.transpose(1,0)
p_t_Ti_tr = p_t_Ti.transpose(1,0)




# Cosine similarity
def cosine_similarity(p_s_tr, p_t_tr):

    return (p_s_tr*p_t_tr).sum(1)/(p_s_tr.norm(p=2, dim=1)*p_t_tr.norm(p=2, dim=1))


#Consine similarity weighted temperature(CSWT)

T_i=(T_max-T_min)(cosine_max - cosine_i)/(cosine_max - cosine_min)+T_min



cs_max = max {cs_1, cs_2, ..., cs_B}

cs_min = min {cs_1, cs_2, ..., cs_B}



# Cosine distance
def cosine_loss(p_s_tr, p_t_tr):
    cosine_distance = 1.0 - cosine_similarity(p_s_tr, p_t_tr)
    return cosine_distance

loss_cosine = cosine_loss(p_s_tr, p_t_tr) * K**2

loss_Ti_cosine = cosine_loss(p_s_Ti_tr, p_t_Ti_tr) * Ti**2


if Multi_KD == True:
    loss_Multi_KD_KL_MSE = False;~Jin et al. (2023)
    loss_Multi_KD_Cosine_similarity = True;

total loss = Cross_entropy(labels, p_s) + loss_cosine + loss_Ti_cosine
                                                + loss_Multi_KD_Cosine_similarity
```