VRM: Knowledge Distillation via Virtual Relation Matching

Supplementary Material

8. Appendix

8.1. List of All Compared Methods

A list of all methods we have compared with in this paper is as follows:

Feature-based methods include FitNets [56], NST [31], AT [74], AB [26], OFD [25], VID [1], CRD [60], SRRL [68], SemCKD [7], PEFD [12], MGD [71], CAT-KD [23], TaT [42], ReviewKD [10], NORM [47], FCFD [45], and RSD [78].

Logit-based methods include KD [27], DML [80], TAKD [50], CTKD [41], NKD [72], DKD [81], LSKD [59], TTM [82], SDD [63], and CRLD [76].

Relation-based methods include FSP [73], RKD [52], PKT [53], CC [54], SP [61], ICKD [46], and DIST [29].

For MS-COCO object detection, we also compare VRM with FGFI [62]. For ConvNet-to-ViT experiments, we also present the results for LG [39] and AutoKD [40].

8.2. List of All Transformation Operations

For our main experiments, we borrow the RandAugment implementation from the `TorchSSL` codebase[1]. It comprises a total of 14 image transformation operations, namely:

1. `Autocontrast`: automatically adjust image contrast
2. `Brightness`: adjust image brightness
3. `Color`: adjust image colour balance
4. `Contrast`: adjust image contrast
5. `Equalize`: equalise image histogram
6. `Identity`: leave image unaltered
7. `Posterize`: reduce number of bits for each channel
8. `Rotate`: rotate image
9. `Sharpness`: adjust image sharpness
10. `Shear_x`: shear image horizontally
11. `Shear_y`: shear image vertically
12. `Solarize`: invert all pixels above a threshold
13. `Translate_x`: translate image horizontally
14. `Translate_y`: translate image vertically

Besides, we also apply `Cutout` with a probability of 1.0, which sets a square patch of random size within the image to gray. The above operations are preceded by `RandomCrop` and `RandomHorizontalFlip` in our virtual view image generation pipeline.

For ConvNet-to-ViT experiments, we follow Li et al. [39] and use the RandAugment function provided by the `timm` library [2]. This function contains 15 image transformation operations:

1. `AutoContrast`: automatically adjust image contrast
2. `Brightness`: adjust image brightness
3. `Color`: adjust image colour balance
4. `Contrast`: adjust image contrast
5. `Equalize`: equalise image histogram
6. `Invert`: invert image
7. `Posterize`: reduce number of bits for each channel
8. `Rotate`: rotate image
9. `Sharpness`: adjust image sharpness
10. `ShearX`: shear image horizontally
11. `ShearY`: shear image vertically
12. `Solarize`: invert all pixels above a threshold
13. `SolarizeAdd`: add a certain value to all pixels below a threshold
14. `TranslateXRel`: translate image horizontally by a fraction of its width
15. `TranslateYRel`: translate image vertically by a fraction of its height

Similar to the role of `Cutout`, the `timm` library additionally implements a `RandomErasing` operation, which

Algorithm 1 PyTorch-style pseudo-code for computing VRM losses

```
# x: a batch of raw samples
# T_r, T_v: transformation functions for real and virtual views
# f^T, f^S: teacher and student networks
# B: batch size, C: number of classes

# Generate real and virtual views of samples
x_r, x_v = T_r(x), T_v(x) # [B]

# Obtain teacher and student predictions
z_r^T, z_v^T = f^T(x_r), f^T(x_v) # [B,C]
z_r^S, z_v^S = f^S(x_r), f^S(x_v) # [B,C]

# Compute inter-sample virtual edge matrices
e_ISV^T = Norm(z_r^T.unsqueeze(0) - z_v^T.unsqueeze(1)) # [B,B,C]
e_ISV^S = Norm(z_r^S.unsqueeze(0) - z_v^S.unsqueeze(1)) # [B,B,C]

# Compute inter-class virtual edge matrices
e_ICV^T = Norm(z_r^T.unsqueeze(1) - z_v^T.unsqueeze(2)) # [C,C,B]
e_ICV^S = Norm(z_r^S.unsqueeze(1) - z_v^S.unsqueeze(2)) # [C,C,B]

# Obtain unreliable edge masks
M_ISV = JE(z_r^S.unsqueeze(0), z_v^S.unsqueeze(1)) < p_m # [B,B]
M_ICV = JE(z_r^S.unsqueeze(1), z_v^S.unsqueeze(2)) < p_m # [C,C]

# Compute VRM losses
loss_isv = (MSE(e_ISV^T, e_ISV^S) * M_ISV).sum()
loss_icv = (MSE(e_ICV^T, e_ICV^S) * M_ICV).sum()

return loss_isv, loss_icv
```

[1] https://github.com/TorchSSL

[2] https://github.com/huggingface/pytorch-image-models

Teacher Student	Venue	ResNet56 ResNet20	ResNet110 ResNet32	ResNet32×4 ResNet8×4	WRN-40-2 WRN-16-2	WRN-40-2 WRN-40-1	VGG13 VGG8
Teacher		72.34	74.31	79.42	75.61	75.61	74.64
Student		69.06	71.14	72.50	73.26	71.98	70.36
Feature-based							
FitNets [56]	ICLR'15	69.21	71.06	73.50	73.58	72.24	71.02
NST [31]	arXiv'17	69.60	71.96	73.30	73.68	72.24	71.53
AT [74]	ICLR'17	70.55	72.31	73.44	74.08	72.77	71.43
AB [26]	AAAI'19	69.47	70.98	73.17	72.50	72.38	70.94
OFD [25]	ICCV'19	70.98	73.23	74.95	75.24	74.33	73.95
VID [1]	CVPR'19	70.38	72.61	73.09	74.11	73.30	71.23
CRD [60]	ICLR'20	71.16	73.48	75.51	75.48	74.14	73.94
SRRL [68]	ICLR'21	71.13	73.48	75.33	75.59	74.18	73.44
PEFD [12]	NeurIPS'22	70.07	73.26	76.08	76.02	74.92	74.35
CAT-KD [23]	CVPR'23	71.05	73.62	76.91	75.60	74.82	74.65
TaT [42]	CVPR'22	71.59	74.05	75.89	76.06	74.97	74.39
ReviewKD [10]	CVPR'21	71.89	73.89	75.63	76.12	75.09	74.84
NORM [47]	ICLR'23	71.35	73.67	76.49	75.65	74.82	73.95
FCFD [45]	ICLR'23	71.96	-	76.62	76.43	75.46	75.22
Logit-based							
KD [27]	arXiv'15	70.66	73.08	73.33	74.92	73.54	72.98
DML [80]	CVPR'18	69.52	72.03	72.12	73.58	72.68	71.79
TAKD [50]	AAAI'20	70.83	73.37	73.81	75.12	73.78	73.23
CTKD [41]	AAAI'23	71.19	73.52	73.79	75.45	73.93	73.52
NKD [72]	ICCV'23	70.40	72.77	76.35	75.24	74.07	74.86
DKD [81]	CVPR'22	71.97	74.11	76.32	76.24	74.81	74.68
LSKD [59]	CVPR'24	71.43	74.17	76.62	76.11	74.37	74.36
TTM [82]	ICLR'24	71.83	73.97	76.17	76.23	74.32	74.33
CRLD [76]	MM'24	72.10	74.42	77.60	76.45	75.58	75.27
Relation-based							
FSP [73]	CVPR'17	69.95	71.89	72.62	72.91	-	70.20
RKD [52]	CVPR'19	69.61	71.82	71.90	73.35	72.22	71.48
PKT [53]	ECCV'18	70.34	72.61	73.64	74.54	73.45	72.88
CCKD [54]	CVPR'19	69.63	71.48	72.97	73.56	72.21	70.71
SP [61]	ICCV'19	69.67	72.69	72.94	73.83	72.43	72.68
ICKD [46]	ICCV'21	71.76	73.89	75.25	75.64	74.33	73.42
DIST [29]	NeurIPS'22	71.75	-	76.31	-	74.73	-
VRM	-	**72.09**	**75.03**	**78.76**	**77.47**	**76.46**	**76.19**

Table 12. Top-1 accuracy (%) on CIFAR-100 for same-model teacher-student pairs.

sets a rectangular patch of random size and shape within the image to random pixels. The above operations are preceded by `RandomResizedCropAndInterpolation` and `RandomHorizontalFlip` in our strong view image generatino pipeline, which is the default configuration in `timm`.

8.3. Pseudo-Code

In Algorithm 1, we provide the PyTorch-style pseudo-code for the calculation of the proposed VRM losses. The construction of relation edges can be conveniently implemented through some matrix operations, with the redundancy edge pruning implicitly incorporated. Overall, the proposed losses can be neatly implemented in about 10 lines of codes in PyTorch.

8.4. Details on Experimental Configurations

Datasets. We conduct experiments on CIFAR-100 and ImageNet for image classification, and MS-COCO for object detection. CIFAR-100 [35] contains 60k 32×32 RGB images annotated in 100 classes. It is split into 50,000 training and 10,000 validation images. ImageNet [19] is a 1,000-category large-scale image recognition dataset. It provides 1.28 million RGB images for training and 5k for validation. MS-COCO [43] is an object detection dataset with images of common objects in 80 categories. We experiment with its `train2017` and `val2017` that include 118k training and 5k validation images, respectively.

Configurations for main experiments. For CIFAR-100 main experiments, we strictly follow the standard training configurations in previous works [45, 59, 81]. Specifically,

Teacher Student	Venue	ResNet32×4 ShuffleNetV2	VGG13 MobileNetV2	ResNet50 MobileNetV2	ResNet50 VGG8	ResNet32×4 ShuffleNetV1	WRN-40-2 ShuffleNetV1
Teacher		79.42	74.64	79.34	79.34	79.42	75.61
Student		71.82	64.60	64.60	70.36	70.50	70.50
Feature-based							
FitNets [56]	ICLR'15	73.54	64.16	63.16	70.69	73.59	73.73
NST [31]	arXiv'17	74.68	58.16	64.96	71.28	74.12	74.89
AB [26]	AAAI'19	74.31	66.06	67.20	70.65	73.55	73.34
AT [74]	ICLR'17	72.73	59.40	58.58	71.84	71.73	73.32
VID [1]	CVPR'19	73.40	65.56	67.57	70.30	73.38	73.61
OFD [25]	ICCV'19	76.82	69.48	69.04	-	75.98	75.85
CRD [60]	ICLR'20	75.65	69.63	69.11	74.30	75.11	76.05
MGD [71]	ECCV'22	76.65	69.44	68.54	73.89	76.22	75.89
SemCKD [7]	AAAI'21	77.02	69.98	68.69	74.18	76.31	76.06
ReviewKD [10]	CVPR'21	77.78	70.37	69.89	75.34	77.45	77.14
NORM [47]	ICLR'23	78.32	69.38	71.17	75.67	77.79	77.63
FCFD [45]	ICLR'23	78.18	70.65	71.00	-	78.12	77.99
CAT-KD [23]	CVPR'23	78.41	69.13	71.36	-	78.26	77.35
Logit-based							
KD [27]	arXiv'15	74.45	67.37	67.35	73.81	74.07	74.83
DML [80]	CVPR'18	73.45	65.63	65.71	-	72.89	72.76
TAKD [50]	AAAI'20	74.82	67.91	68.02	-	74.53	75.34
CTKD [41]	AAAI'23	75.31	68.46	68.47	-	74.48	75.78
NKD [72]	ICCV'23	76.26	70.22	70.76	74.01	75.31	75.96
DKD [81]	CVPR'22	77.07	69.71	70.35	-	76.45	76.70
LSKD [59]	CVPR'24	75.56	68.61	69.02	-	-	-
TTM [82]	ICLR'24	76.55	69.16	69.59	74.82	74.37	75.42
SDD [63]	CVPR'24	76.67	68.79	69.55	74.89	76.30	76.54
CRLD [76]	MM'24	78.27	70.39	71.36	-	-	-
Relation-based							
RKD [52]	CVPR'19	73.21	64.52	64.43	71.50	72.28	72.21
PKT [53]	ECCV'18	74.69	67.13	66.52	73.01	74.10	73.89
CCKD [54]	CVPR'19	71.29	64.86	65.43	70.25	71.14	71.38
SP [61]	ICCV'19	74.56	66.30	68.08	73.34	73.48	74.52
DIST [29]	NeurIPS'22	77.35	68.50	68.66	74.11	76.34	76.40
VRM	-	**79.34**	**71.66**	**72.30**	**76.96**	**78.28**	**78.62**

Table 13. Top-1 accuracy (%) on CIFAR-100 for different-model teacher-student pairs.

we train our framework for 240 epochs using the SGD optimiser and a batch size of 64. The initial LR is 0.01 for MobileNets [28] and ShuffleNets [79] and 0.05 for other architectures, which decay by a factor of 10 at [150th, 180th, 210th] epochs. Momentum and weight decay are set to 0.9 and 5e-4, respectively. Softmax temperature is set to 4.

For ImageNet experiments, as per standard practice, we train our framework for 100 epochs with a batch size of 256 on two GPUs, with an initial LR of 0.1 that decays by a factor of 10 at [30th, 60th, 90th] epochs. Momentum and weight decay are set to 0.9 and 1e-4, respectively. Softmax temperature is set to 2.

For MS-COCO object detection, we adopt the configurations of Chen et al. [10], Liu et al. [47], Sun et al. [59], Wang et al. [62], Zhao et al. [81] whereby we experiment with Faster-RCNN-FPN [44] with different backbone models. All models are trained for 180,000 iterations on 2 GPUs with a batch size of 8. The LR is initially set as 0.01 and decays at the 120,000th and 160,000th iterations.

Configurations for ConvNet-to-ViT experiments. As few studies have considered this setting, we developed our experiments following [40, 59] on the codebase provided by [39]. Our experimental configurations follow [39] and [49]. Specifically, the ResNet56 teacher is trained for 300 epochs with an initial LR of 0.1 and a cosine LR schedule. The resulted pretrained teacher has a top-1 accuracy of 71.61%. All ViTs are trained for 300 epochs (including 20-epoch linear warm-up) using the AdamW optimiser. The initial LR is 5e-4 with a weight decay of 0.05, which eventually decays to 5e-6 via a cosine LR policy. The ResNet56 teacher is trained on 32×32 resolution images, while ViT students are fed with 224×224 images. The default RandAugment is applied for data augmentation, with number of randomly sampled operations n set to 2, transform magnitude m to 9, and probability of applying random erasing p

	ResNet32×4 ResNet8×4	VGG13 VGG8
Baseline	**78.76**	**76.19**
w/o L_{ce}^{S}	78.47	75.66

Table 14. Effect of L_{ce}^{V} supervision.

to 0.25. All models are trained on a single NVIDIA RTX 3090 GPU with a batch size of 128.

Implementations. Our method is implemented in the `mdistiller`[3] codebase in PyTorch for image classification experiments. For object detection, it also partially builds upon the `detectron2`[4] library. For ConvNet-to-ViT experiments, we utilise the `pycls`[5] and the `tiny-transformer`[6] codebases. All reported results are average over 3 trials.

Efficiency benchmarking. Tab. 10 benchmarks the training time per batch and the peak GPU memory usage of various methods on a workstation equipped with 20 Intel Core i9-10850K CPUs (10 cores) and an NVIDIA RTX 3090 GPU. All measurements are taken on CIFAR-100 with a batch size of 64.

8.5. More Experimental Results

We present the full results on CIFAR-100 in Tabs. 12 and 13 to include more same-model and different-model distillation pairs and additional methods for comparison.

8.6. More Ablation Studies

Effect of GT supervision policies. Tab. 14 shows the effect of removing the GT supervision on the student model's predictions of the virtual-view image. It demonstrates that supervising student predictions of the virtual view is important for ensuring the quality of the virtual view predictions. The quality of vertices has a direct impact on the quality of the edges (*i.e.*, relations) constructed within the affinity graphs. As such, we choose to also supervise the virtual vertices of our graphs with GT labels.

Effect of longer training. The construction and transfer of richer and more diverse relations mean that VRM may benefit more from longer training. To verify this, we devise a longer training policy (denoted as "LT") than the standard 240-epoch policy in existing KD methods. For our LT policy, the model is trained for 360 epochs and the LR decays by a factor of 10 at the 150th, 180th, 210th, and 270th epochs. All other configurations are kept the same. According to Tab. 15, VRM indeed benefits from longer training as a 0.21% Top-1 accuracy gain is obtained with LT. In comparison, the performance of other methods plateaued with more training epochs, which is likely due to overfitting to the training samples and a lack of richer guidance signals from the teacher.

[3] https://github.com/megvii-research/mdistiller
[4] https://github.com/facebookresearch/detectron2
[5] https://github.com/facebookresearch/pycls
[6] https://github.com/lkhl/tiny-transformers

	KD	RKD	DIST	VRM
Baseline	73.83	72.63	76.16	78.76
LT	73.82	72.49	75.90	**78.97**

Table 15. Effect of longer training. "LT": long-training policy.

τ	1	2	3	4	5	6
Acc. (%)	78.16	78.69	78.57	**78.76**	78.41	78.63

Table 16. Effect of different temperatures.

Effect of different temperatures. The temperature of Softmax, denoted by τ, controls the smoothness of the predicted probabilistic distribution. We simply opt for the common choice of $\tau = 4$ [27, 81], which is empirically shown to produce the best results. Moreover, the proposed method is sufficiently robust to varying values of τ, as shown in Tab. 16.

Effect of pruning redundant edges. We conduct ablation experiments to see the effect of pruning redundant edges, described in Sec. 5.4 of the main text. As presented in Tab. 17, matching the raw and bulky inter-sample affinity graph with redundancy and duplication is not only less efficient but also inferior in terms of performance. We postulate that this is partially ascribed the fact that each vertex in the raw graph is connected to a larger and more complex set of other vertices that involve both real and virtual vertices. This complicates the learning while making each vertex more vulnerable to an increased likelihood of adverse gradient propagation. Another possible reason is that matching real-virtual relations is more regularised, as opposed to matching real-real or virtual-virtual intra-view relations that are easier and more readily overfitted. Note that we also conjectured that the degraded performance of matching the raw graph may be ascribed to different distribution patterns of the prediction vectors at the vertices since they now have a dimension of $2 \times B$ compared to B. We experimented with different temperature τ in an attempt to re-adjust the distributions to be more relation-matching-friendly, but the results remain inferior.

Pruning Configuration	ResNet32×4 ResNet8×4	VGG13 VGG8
Redundant $\mathcal{E}$ ($\tau = 1$)	77.39	75.26
Redundant $\mathcal{E}$ ($\tau = 2$)	77.80	74.94
Redundant $\mathcal{E}$ ($\tau = 4$)	77.19	75.30
Pruned $\mathcal{E}$ ($\tau = 4$)	**78.76**	**76.19**

Table 17. Effect of pruning redundant edges.

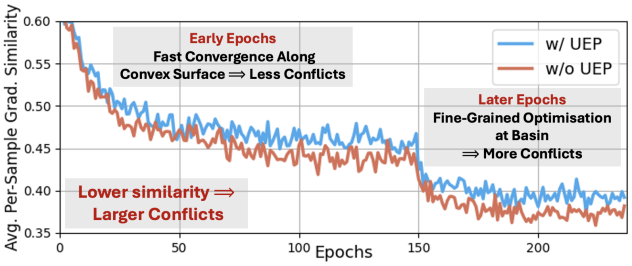


Figure 8. Effect of unreliable edge pruning on gradient conflicts in training.

8.7. More Analyses

Analysis of logit mean & standard deviation. In Fig. 9, we plot the histogram of the mean and standard deviation of instance-wise logit predictions given by various method. IM methods are found to produce logits closer to the teachers' in terms of both logit mean and standard deviation distributions. Intriguingly, the proposed VRM, being a purely relation-based method that is free of any explicit instance-wise logit matching, is on par with IM methods in this regard and markedly outdoes DKD. This suggests that the proposed real-virtual relational matching provides strong regularisation that better enables the student to learn the underlying logit distribution of the teacher. This is particularly evident given that RKD, a relation-based method which also has an IM objective, is way further from teacher's logit distribution.

Effect of UEP on optimisation conflicts in training. To gain further insights into the effect of unreliable edge pruning on the training dynamics, we trace the optimisation gradient similarity in matching the edges throughout training. Specifically, we compute the averaged pairwise cosine similarity between all sample-wise gradient vectors within a batch, and visualise the results in Fig. 8. We observe that UEP leads to higher gradient similarity on average. In other words, there are fewer gradient conflicts, which also explains the faster convergence evidenced by Figs 7d.

VRM on features. In this work, we have chosen to construct our virtual relation graphs $\mathcal{G}^{IS}$ and $\mathcal{G}^{IC}$ from network prediction logits $\{\mathbf{z_i}\}_{i=1}^{B}$. In this section, we conduct additional experiments to investigate to what extent VRM can work with features. To this end, we simply reconstruct our graphs from the feature maps $\{\mathbf{f}_i\}_{i=1}^{B}$ right before the final linear layer (denoted as "pooled feats" in Tab. 18) and in the `mdistiller` codebase. Our virtual relation graphs now become $\mathcal{G}^{IS} \in \mathbb{R}^{B\times B\times D}$ and $\mathcal{G}^{IC} \in \mathbb{R}^{D\times D\times B}$ where D is the dimension of the feature vector. Note that since we no longer work with probability distributions, we remove the Softmax operations that convert predictions to probabilities. Other operations remain unchanged.

Method	Location	ResNet32×4 ResNet8×4	VGG13 VGG8
RKD	pooled feats	72.63	70.87
PKT		74.41	72.78
CRD		75.51	73.94
ReviewKD		75.63	74.84
VRM		**76.39**	**74.92**
VRM	logits	**78.76**	**76.19**

Table 18. Applying VRM to feature embeddings.

In Tab. 18, we compare the results of VRM trained using graphs constructed from feature maps with existing methods that also build relations from the same features (*i.e.*, "pooled feats"), namely RKD [52], PKT [53], CRD [60], and ReviewKD [10]. It can be observed that the performance of VRM deteriorates when applied to features. The reason may be that predicted logits are more compact condensation of categorical knowledge, which is therefore more beneficial for our downstream task. This is particularly so given that VRM does not contain an IM objective that directly matches the logits. As such, VRM works best when applied to logits. Nonetheless, when applied to features, VRM still substantially outperforms all other methods that also work on the very same feature maps. This shows that VRM still encodes better and richer knowledge for distillation compared to the weaker relations transferred by RKD and PKT.

Discussions on the use of joint entropy. In the formulation of our unreliable edge pruning scheme, we use the joint entropy (JE) between two predictions (of two nodes) as a measure of edge uncertainty. While other measures may be used, JE suits our purpose with several appealing properties:

- Higher discrepancy between two vertices leads to higher JE, which is a relative measure of uncertainty.
- As two predictions get aligned, JE approaches their individual uncertainty, which is an absolute measure of uncertainty.

As such, our criterion takes account of both relative and absolute edge uncertainties throughout the learning process.

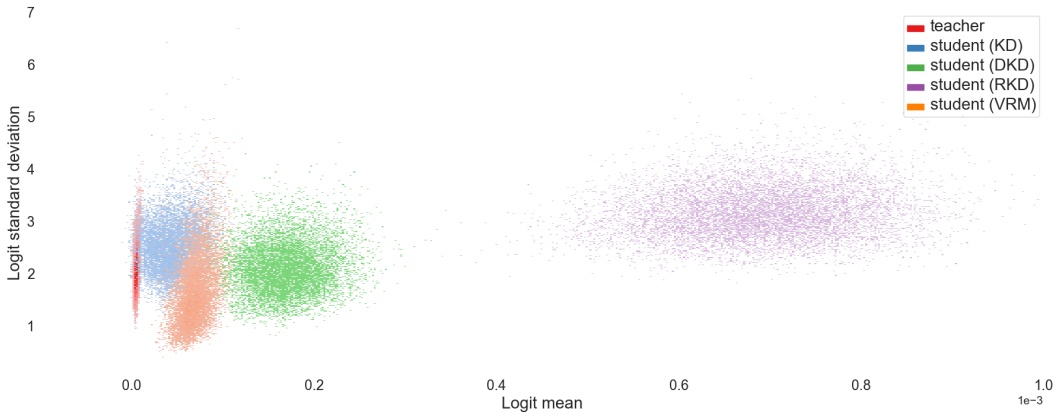


Figure 9. Bivariate histogram of the mean and standard deviation of logits predicted by different models on CIFAR-100.

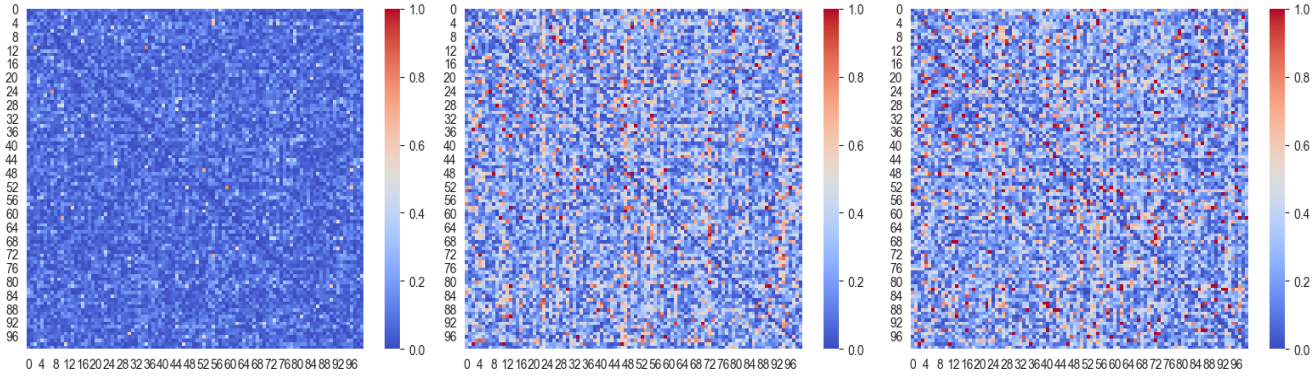
Comparison to KD methods using SSL. We highlight the difference between our method and two KD methods that utilise self-supervised learning (SSL), namely SSKD [65] and HSAKD [66].

SSKD utilises self-supervision signals via image transformations and pretext tasks for knowledge distillation. The proposed VRM fundamentally differs from SSKD in at least the following aspects:

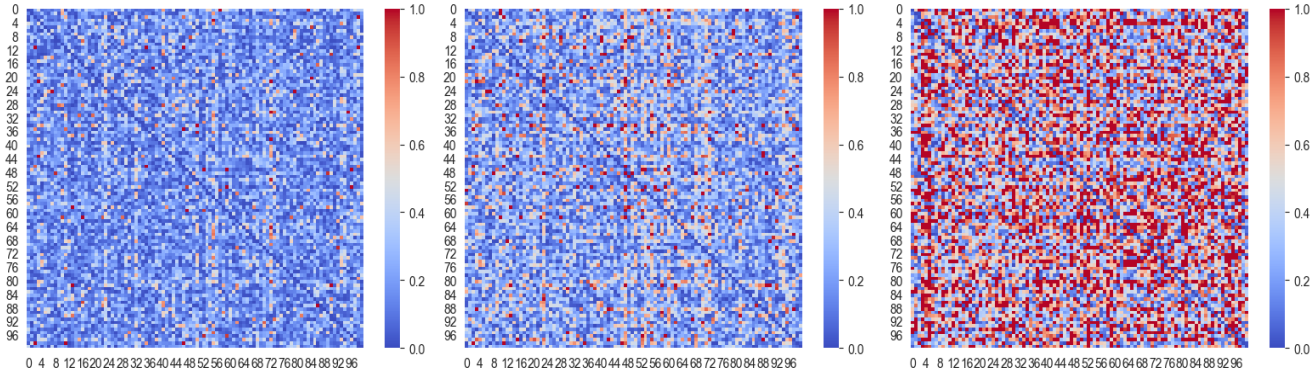
- *Motivation*: While both methods involve the utilisation of transformations to produce augmented views of input images, SSKD is directly inspired by and leverage the *pretext task* in self-supervised learning [11]. In contrast, our method is pretext-task-free. Instead, VRM is motivated by *transformation invariance regularisation* which was originally popularised in semi-supervised learning [3, 37] and domain adaptation [4].
- *Training formulation*: A direct consequence of the point above is that SSKD's teacher first needs to be re-trained with additional augmentations (which also causes SSKD to use teachers of higher accuracy than ours), followed by a separate fine-tuning stage for the pretext task. These lead to significantly more procedures and computations, whereas VRM is entirely free of such palaver.
- *Nature of matching objectives*: SSKD is essentially a *hybrid* method that employs both relation matching and instance-to-instance matching objectives, whereas VRM is *purely relation-based* method. In other words, SSKD relies on IM to achieve competitive performance, while VRM involves purely relation-based objectives.
- *Design choices*: The designs of both methods are vastly different, including but not limited to the formulation of relations and the choices of augmentation policies, relation distance metrics, and model outputs used for computing relations.

HSAKD is another method that makes use of self-supervised learning and transformed views of input images. This method is also fundamentally different from VRM from the following aspects:

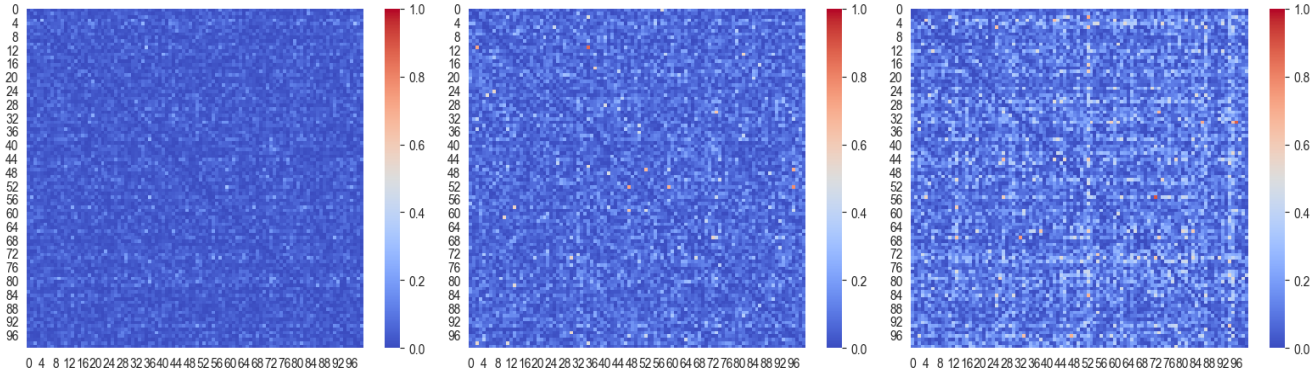
- *Motivation*: Like SSKD, HSAKD is also directly motivated by the use of pretext tasks in self-supervised learning. HSAKD employs rotation prediction as its pretext task. The proposed VRM is free of pretext task learning.
- *Training formulation*: To enable pretext task learning, HSAKD appends auxiliary classifiers to the intermediate features at each stage to perform transformation classification. This means that, akin to SSKD, HSAKD also needs to re-train the teacher model with modified architecture over the pretext task. The auxiliary classifiers also introduce extra parameters. In contrast, the proposed VRM does not involve these additional procedural, parameter, and computational costs.
- *Nature of matching objectives*: By matching the predictions made by a set of auxiliary classifiers between teacher and student for each sample (as well as matching the final predicted probability distributions between teacher and student), HSAKD is fundamentally a instance matching approach, whereas VRM transfers purely relational knowledge. Moreover, HSAKD employs symmetric matching, which means the matching between teacher and student auxiliary predictions are for the same view of the input samples. By contrast, VRM exploits the relations across asymmetric real and virtual views with different difficulties.
- *Design choices*: HSAKD also differs from the proposed VRM in terms of specific designs made. For example, HSAKD adopts rotation to construct its pretext task, whereas VRM utilises RandAugment policies. HSAKD also relies on the use of the instance-wise logit matching loss from vanilla KD [27] to reach competitive performance, whereas VRM does not use any instance-matching KD objective and still achieves much more superior performance.



(a) KD (b) RKD (c) PKT



(d) SP (e) ICKD (f) DIST



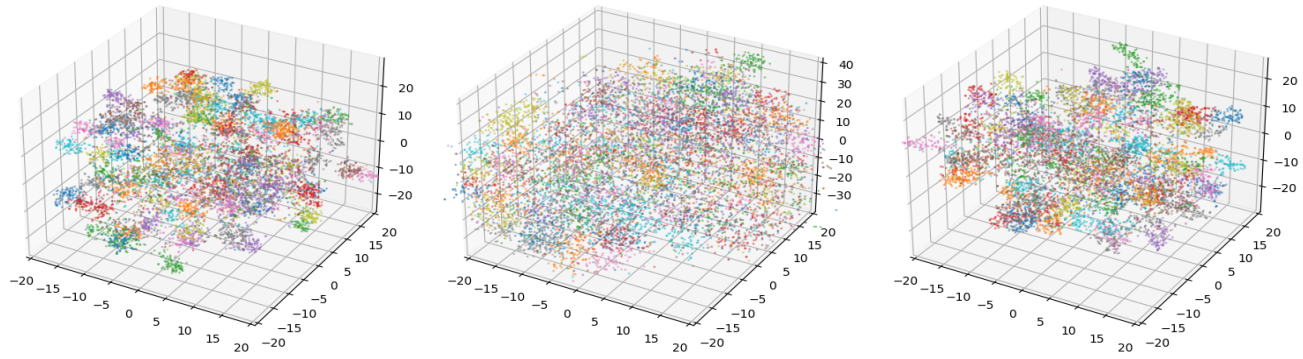
(g) DKD (h) MLLD (i) VRM

Figure 10. More visualisations of teacher-student prediction discrepancy maps for different KD methods.

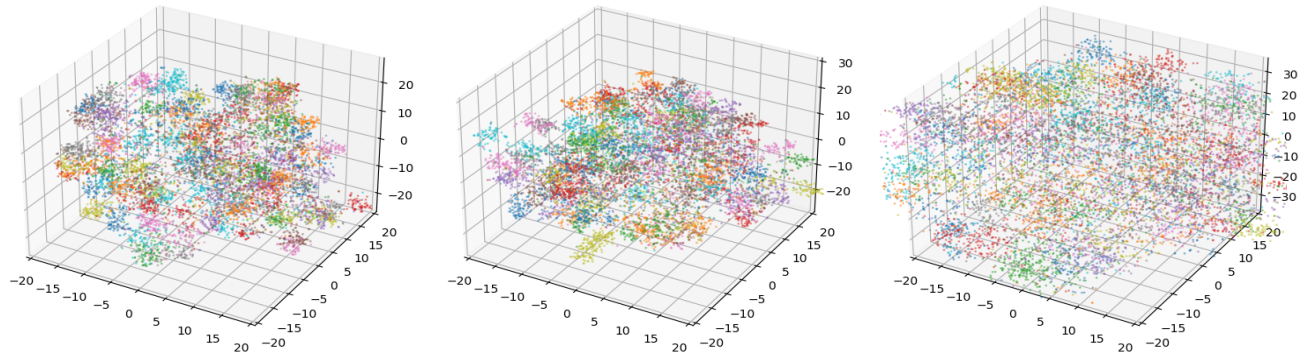
More teacher-student discrepancy maps visualisations. Fig. 10, we provide more visualisations of the class-wise prediction discrepancy between teacher and student models for different KD methods (ResNet32$\times$4 $\rightarrow$ ResNet8$\times$4 on CIFAR-100).

More t-SNE visualisations. In Fig. 11, we further showcase the t-SNE visualisations of embeddings learnt by more KD methods as well as those by the teacher model (ResNet32$\times$4$\rightarrow$ResNet8$\times$4 on CIFAR-100).

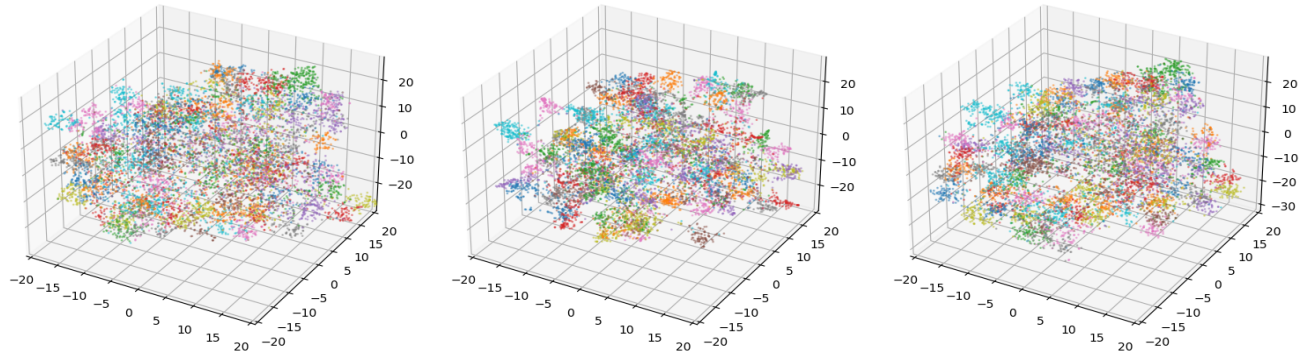
More loss landscape visualisations. Fig. 12 provides more visualisations of loss landscape for different KD methods (ResNet32$\times$4$\rightarrow$ResNet8$\times$4 on CIFAR-100).



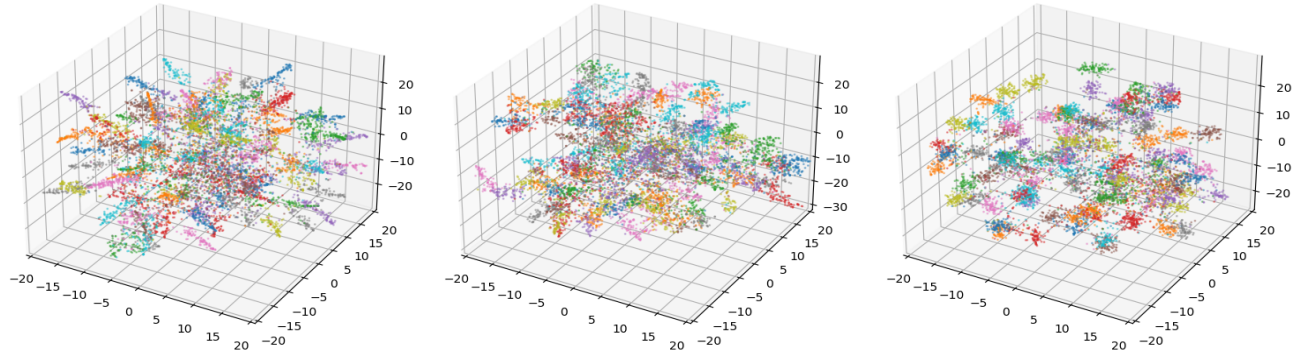
(a) KD (b) RKD (c) PKT



(d) SP (e) ICKD (f) DIST

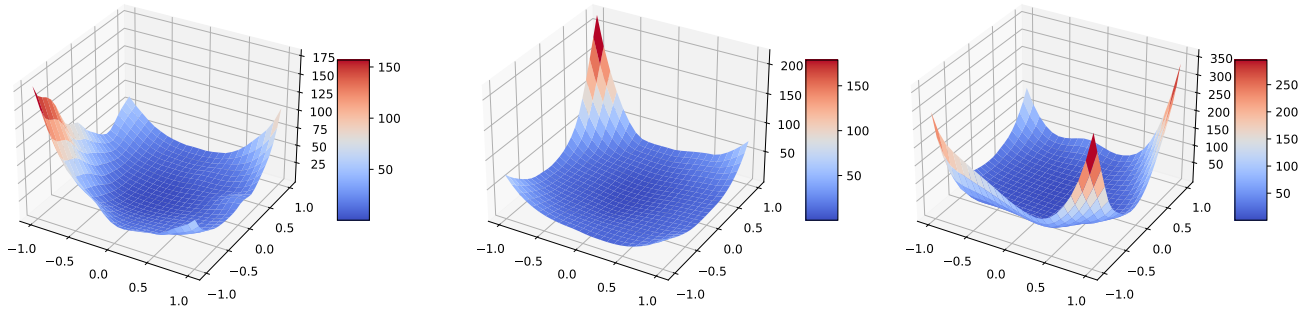


(g) FitNets (h) DKD (i) MLLD

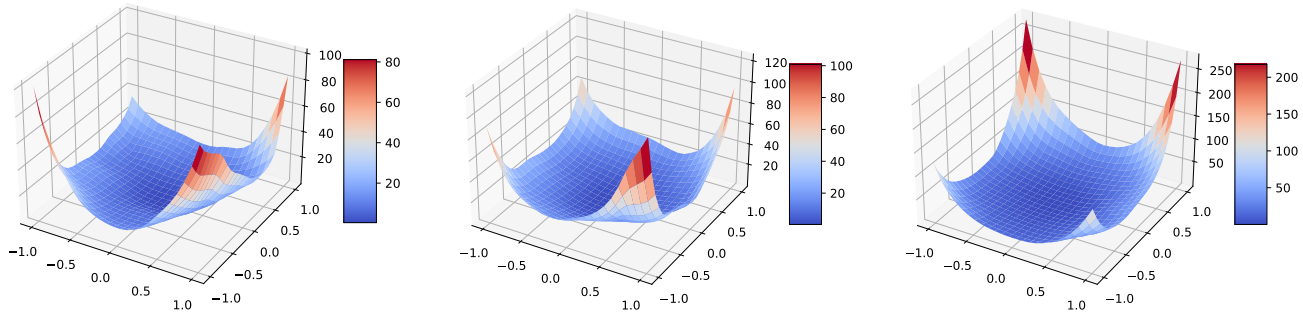


(j) LSKD (k) VRM (l) Teacher

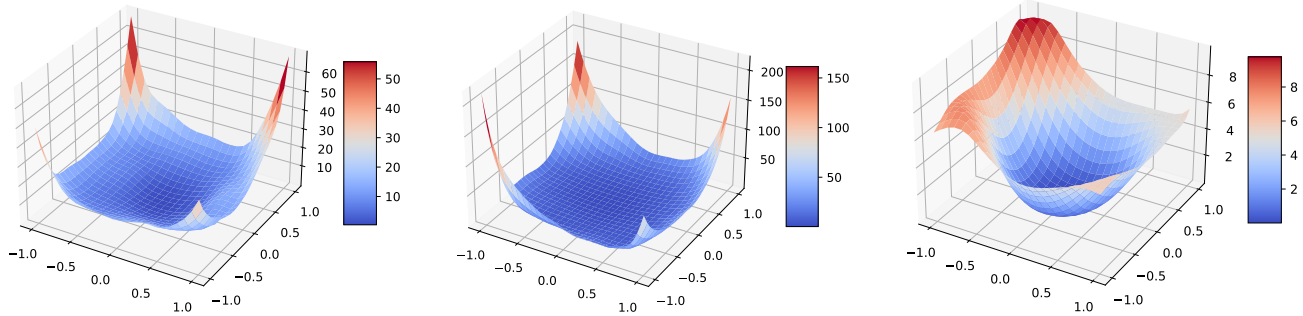
Figure 11. More t-SNE visualisations of features learnt by different KD methods.



(a) KD (b) RKD (c) PKT



(d) SP (e) ICKD (f) DIST



(g) DKD (h) VRM (i) Teacher

Figure 12. More loss landscape visualisations of different KD methods.