V_kD : Improving Knowledge Distillation using Orthogonal Projections

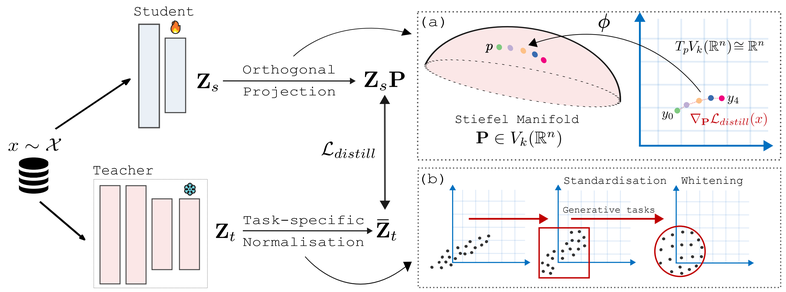
Roy Miles, Ismail Elezi, Jiankang Deng

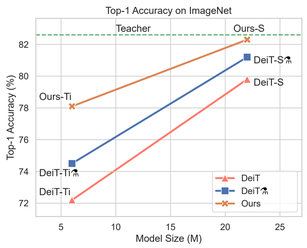
Huawei Noah's Ark Lab

CVPR SEATTLE, WA JUNE 17-21, 2024

Overview

We introduce a simple and cheap constrained feature distillation pipeline derived from a core set of principles. These principles lead us to two main components: (i) an orthogonal projection layer; and (ii) task-specific feature normalization.





Our transformer models surpass previous methods on ImageNet, achieving up to a 4.4% relative improvement.

Applied to object detection and image generation, our approach consistently outperforms state-of-the-art techniques, demonstrating its generality and effectiveness.

Background and limitations

- Although feature distillation is agnostic to the underlying task or modality, most proposed methods incur significant memory and computational overheads due to the construction of expensive relational objects.
- Generalising knowledge distillation to other tasks is often difficult. There is no simple framework for introducing domain/task-specific priors.

Contributions

Our contributions can be summarized as follows

- We propose a novel orthogonal projection layer to maximise the knowledge being distilled through to the student backbone.
- We complement our projection with a task-wise normalisation that enables knowledge distillation on generative tasks.
- We apply our method to a wide range of vision tasks, improving over the state-of-the-art by up to 4.4% on ImageNet.

Why use orthogonal projections?

- A learnable projection layer is needed to match the feature dimensions.
- However, we wish to mitigate the possibility of this layer learning any new representation of the data that is not shared by the feature extractor.
- This is crucial because the projection layer is discarded after training, and our end goal is to align the feature extractor with the teacher.
- To this end, we propose a projection that preserves the pair-wise similarity of features through the projection layer.

This pair-wise similarity can be expressed using a kernel function:

$$k(\mathbf{Z}_i^s, \mathbf{Z}_j^s) = \sum_{n=0}^{\infty} a_n \left\langle \mathbf{Z}_i^s, \mathbf{Z}_j^s \right\rangle^n \tag{1}$$

Which is preserved under any orthogonal transformation $\mathbf{P}$.

$$\mathbf{Z}_i^s(\mathbf{Z}_j^s)^T = \mathbf{Z}_i^s\mathbf{P}(\mathbf{Z}_j^s\mathbf{P})^T = \mathbf{Z}_i^s\mathbf{P}\mathbf{P}^T(\mathbf{Z}_j^s)^T \rightarrow \boxed{\mathbf{P}^T = \mathbf{P}^{-1}} \tag{2}$$

This constraint defines the set of matrices with orthonormal rows.

Enforcing this constraint is expensive. We need an efficient parameterization.

Efficient orthogonal parameterization

Knowledge distillation is already very memory and computationally intensive, thus we do want to introduce any additional overheads by using a parameterised projection layer.

Algorithm 1 VkD, Pytorch-like

```
# W: dt x dt
# Zs: B x N x ds
# Zt: B x dt

def distill_loss(Zs, Zt):
    # average pool over token-dim
    Zs = Zs.mean(1)

    # orthogonal projection
    A = torch.linalg.matrix_exp(W)
    P = A[:, 0:ds]
    Zs = F.linear(Zs, P)

    # task-specific normalisation
    Zt = F.layer_norm(Zt)

    loss = F.mse_loss(Zs, Zt)
    return loss
```

Although the Cayley transformation can construct orthogonal matrices from skew-symmetric matrices, it does require computing the inverse of a potentially very large matrix, which will incur a significant memory overhead.

We propose to perform a cheap parameterisation map onto $SO(dt)$ using the matrix exponential and then truncate the excess columns.

Introducing domain-specific priors

Standardisation improves model convergence. We empirically find that standardising the teacher features across the depth, i.e. Layer Norm, improves model convergence by smoothing the loss landscape.

Whitening improves feature diversity. Applying KD to image generation often requires additional diversity losses. We find that simply whitening the teacher features is sufficient for unifying the diversity and distillation objectives.

Data-efficient training of transformer models

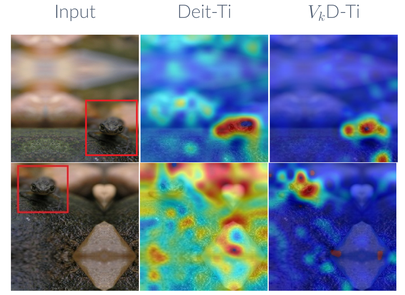
Network	acc@1	#params
DeiT-Ti	72.2	5M
CivT-Ti	74.9	6M
DeiT-Ti ⚗	74.5	6M
↳ 1000 epochs	76.6	6M
DearKD	74.8	6M
↳ 1000 epochs	77.0	6M
V_kD-Ti	**79.2**	6M
DeiT-S	79.8	22M
CivT-S	82.0	22M
DeiT-S ⚗	81.2	22M
↳ 1000 epochs	82.6	22M
DearKD	81.5	22M
↳ 1000 epochs	82.8	22M
V_kD-S	**82.9**	22M

We observe a significant improvement over both DeiT and CivT when the capacity gap is large. We also achieve competitive performance to other distillation methods that are trained for 1000 epochs.

CivT requires multiple teacher models, while DearKD uses intermediate feature losses. In contrast, our method is both simple and easy to implement, while showing strong model convergence.

To further demonstrate the generality of our proposed pipeline, we also conduct experiments with ViDT for object detection. See our paper for more details on this!

To shed some insight into why our method is more effective, we look at the intermediate feature maps. We find that forcing all the patch tokens to more closely align with the teacher features will provide a more stronger equivariance objective than using just a single distillation token.



Data limited image generation

We have shown how and why cross-architecture distillation improves data-efficiency, but how can we improve data-efficiency in the same-architecture setting? For image generation, we find that whitening the teacher features is very important here.

Although only using layer norm and an orthogonal projection does generalise well to this task, whitening introduces more appropriate task-specific priors.

This simple modification then outperforms KD-DLGAN, a single task-specific KD method, across both the CIFAR10 and CIFAR100 datasets and also at various levels of data scarcity.

Method	CIFAR-10 10% Data	CIFAR-100 10% Data
DA (Baseline)	23.34 ± 0.09	35.39 ± 0.08
FitNets	22.03 ± 0.07	33.93 ± 0.09
PKD	21.34 ± 0.08	32.15 ± 0.13
SPKD	19.11 ± 0.07	31.97 ± 0.10
KD-DLGAN	14.20 ± 0.06	18.03 ± 0.11
V_kD	16.47 ± 0.07	24.92 ± 0.15
↳ w/ whitening	$\mathbf{13.16 \pm 0.06}$	$\mathbf{16.87 \pm 0.09}$