Learning to Retain while Acquiring: Combating Distribution-Shift in Adversarial Data-Free Knowledge Distillation
Supplemental Material

Gaurav Patel† Konda Reddy Mopuri‡ Qiang Qiu†
†Purdue University ‡Indian Institute of Technology Hyderabad
{gpatel10, qqiu}@purdue.edu, krmopuri@ai.iith.ac.in

A. Gradient Matching Ablation

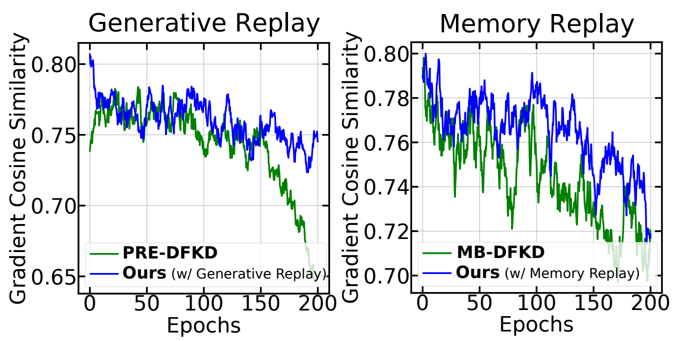

Figure 1. Gradient Cosine Similarity vs. Epochs.

In Figure 1, we monitor the alignment as the layer-wise average gradient *Cosine Similarity* (of $\mathcal{L}_{Acq}$ and $\mathcal{L}_{Ret}$) during the course of the learning process for our method against PRE-DFKD [1] and MB-DFKD [2] on SVHN [17]. We observe a better alignment retention of the gradients in our method.

B. Training Details:

B.1. Teacher Model Training Details

We train the ResNet-34 [8] teacher model for SVHN [17] and Tiny-ImageNet [13]. For SVHN we use the ResNet-34 model definition made available by Binci *et al.*[1] and for Tiny-ImageNet, we use the `torchvision` model definition from PyTorch[2]. To train the teacher models we use SGD optimizer with an initial learning rate of 0.1, momentum of 0.9 and a weight-decay of 5e-4, with a batch size of 128 for 400 epochs. Moreover, the learning rate is decayed at each iteration till 0, using cosine annealing.

B.2. Student Model Training Details

For fair comparisons, we use the same Generator ($\mathcal{G}$) network (shown in Table 1) for all the methods. Unless not explicitly specified, for MB-DFKD [2] and our method (w/ Memory Buffer), we maintain a memory buffer of size 10 and update the memory buffer at a frequency of $f = 5$, following previous work [2] (Algorithm 1). Also, for PRE-DFKD [1] and our method (w/ Generative Replay), we use the same VAE architecture (as in Table 1 (Decoder) and 2 (Encoder)), from [1], to transfer the pseudo samples as memory, and use the decoder part (same as the generator architecture in Table 1) to replay the learnt distribution, with the VAE update parameters of $f = 1$ and $s^{gp}_{max} = 4$ (Algorithm 2), following previous works [1]. For all the methods and datasets, we use SGD optimizer with a momentum of 0.9 and a variable learning rate ($\alpha_{\mathcal{S}}$) with cosine annealing starting from 1e-1 and annealing it at each epoch to 0 to optimize the student parameters ($\theta_{\mathcal{S}}$). For the one-step gradient

[1] https://github.com/kuluhan/PRE-DFKD
[2] https://pytorch.org/

Algorithm 1: Proposed DFKD method, with Memory-Buffer replay.

Input: $\mathcal{T}_{\theta_{\mathcal{T}}}$, $\mathcal{S}_{\theta_{\mathcal{S}}}$, $\mathcal{G}_{\theta_{\mathcal{G}}}$, $\mathcal{M}$, $\mathcal{E}_{max}$, $\mathcal{I}$, g, $\alpha_{\mathcal{G}}$, s, α, $\alpha_{\mathcal{S}}$, f
Output: $\mathcal{S}_{\theta_{\mathcal{S}}}$

```
E = 1
while E ≤ E_max do
    for I iterations do
        for g iterations do
            z ~ N(0, I)
            L_G ← −D(T(G_θG(z)), S(G_θG(z))) + L_P(G_θG(z))
            θ_G ← θ_G − α_G ∇_θG L_G
        end
        for s iterations do
            z ~ N(0, I)
            x̂ ← G_θG(z)
            Compute L_Acq(θ_S) using x̂
            L_S ← L_Acq(θ_S)
            if M is not empty then
                x̂_m ~ M
                θ'_S ← θ_S − α∇L_Acq(θ_S)
                Compute L_Ret(θ_S) and L_Ret(θ'_S) using x̂_m
                L_S ← L_S + L_Ret(θ_S) + L_Ret(θ'_S)
            end
            θ_S ← θ_S − α_S ∇_θS L_S
        end
    end
    if E mod f == 0 then
        Update M with x*_m, where, x*_m ⊆ x̂
    end
    E ← E + 1
end
```

descent, we use a learning rate (α) of 0.9. Furthermore, we use Adam optimizer with a learning rate ($\alpha_{\mathcal{G}}$) of 0.02 to optimize the Generator ($\mathcal{G}$). We test all our methods primarily on SVHN [17], CIFAR10 [12], CIFAR100 [12], and Tiny-ImageNet [13] for 200, 200, 400, and 500 epochs ($\mathcal{E}_{max}$), respectively.

B.3. GPU Memory Utilization

Moreover, our student update strategy brings in no practical memory overhead, compared to memory-based Adversarial DFKD methods. We observe only a minimal increase in the GPU memory usage of few MBs ($\approx$ 40 MB) due to the higher order gradients computed as a part of the update on $\theta_{\mathcal{S}}$ through $\theta'_{\mathcal{S}}$. Moreover, we use a *single* gradient descent step to obtain $\theta'_{\mathcal{S}}$, which does not incur a large memory overhead. Thus, we do not opt for a first order approximation [18] of our method, which is much prevalent in the meta-learning literature. Our experiments were run on a mixture of Nvidia RTX 2080Ti (11GB) and RTX 3090 (24GB) GPUs.

C. Attribution of Existing Assets:

C.1. Code-Base:

The code-base used to experiment with proposed method is adapted from the GitHub[1] repository of Binci *et al.* [1].

C.2. Pre Trained Teacher Model

The CIFAR10 pretrained [12] Teacher models of ResNet-34 and WRN-40-2 [22] are used used from the GitHub[3] repository made available by Fang *et al.* [6]. For the ResNet-34 Teacher model, pretrained on CIFAR100 [12], we used the model made

[3] https://github.com/zju-vipa/CMI

Algorithm 2: Proposed DFKD method, with Generative replay.

Input: $\mathcal{T}_{\theta_\mathcal{T}}$, $\mathcal{S}_{\theta_\mathcal{S}}$, $\mathcal{G}_{\theta_\mathcal{G}}$, $\mathcal{M}$, $\mathcal{E}_{max}$, $\mathcal{I}$, g, $\alpha_\mathcal{G}$, s, α, $\alpha_\mathcal{S}$, f, s^{gp}_{max}
Output: $\mathcal{S}_{\theta_\mathcal{S}}$

```
E = 1
while E ≤ E_max do
    for I iterations do
        for g iterations do
            z ~ N(0, I)
            L_G ← −D(T(G_θG(z)), S(G_θG(z))) + L_P(G_θG(z))
            θ_G ← θ_G − α_G ∇_θG L_G
        end
        for s iterations do
            z ~ N(0, I)
            x̂ ← G_θG(z)
            Compute L_Acq(θ_S) using x̂
            L_S ← L_Acq(θ_S)
            x̂_m ~ M
            θ'_S ← θ_S − α ∇L_Acq(θ_S)
            Compute L_Ret(θ_S) and L_Ret(θ'_S) using x̂_m
            L_S ← L_S + L_Ret(θ_S) + L_Ret(θ'_S)
            θ_S ← θ_S − α_S ∇_θS L_S
            s^gp = 0
            if E mod f == 0 and s^gp ≤ s^gp_max then
                Train M with x̂_m and x*_m, where, x*_m ⊆ x̂
                s^gp ← s^gp + 1
            end
        end
    end
    E ← E + 1
end
```

Table 1. Generator Network ($\mathcal{G}$) and Generative Replay (VAE [11]) Decoder Architecture.

Output Size	**Layers**
1000	Noise ($z \sim \mathcal{N}(0, I)$)
$128 \times h/4 \times w/4$	*Linear*, *BatchNorm1D*, *Reshape*
$128 \times h/4 \times w/4$	*SpectralNorm (Conv* (3×3)*)*, *BatchNorm2D*, *LeakyReLU*
$128 \times h/2 \times w/2$	*UpSample* $(2\times)$
$64 \times h/2 \times w/2$	*SpectralNorm (Conv* (3×3)*)*, *BatchNorm2D*, *LeakyReLU*
$64 \times h \times w$	*UpSample* $(2\times)$
$3 \times h \times w$	*SpectralNorm (Conv* (3×3)*)*, *TanH*, *BatchNorm2D*

available by Binci *et al.*[1] [1].

D. Extended Results

In Figure 2, we visualize the Cumulative Mean Accuracies (%) across the training epochs with Buffer-based and Generative Replay. The plots in Figure 2 complement the ones shown in Figure 5 of the main manuscript.

Based on the similarity of the Tiny-ImageNet teacher accuracy ($\mathcal{T}_{Acc}$) of the methods proposed and reported by Li *et al.* [14], we compare our methods with the accuracies reported by them.

Table 2. Generative Replay (VAE [11]) Encoder Architecture.

Output Size	**Layers**
$3 \times h \times w$	Input Example
$64 \times h \times w$	*SpectralNorm(Conv* (3×3)*)*, *BatchNorm2D*, *LeakyReLU*
$128 \times h \times w$	*SpectralNorm(Conv* (3×3)*)*, *BatchNorm2D*, *LeakyReLU*
$128 \times h/2 \times w/2$	*DownSample* $(0.5\times)$
$128 \times h/2 \times w/2$	*SpectralNorm(Conv* (3×3)*)*, *BatchNorm2D*
$128 \times h/4 \times w/4$	*DownSample* $(0.5\times)$
$\{1000, 1000\}$	*Reshape*, *Linear*

Method	Teacher Accuracy (%) ($\mathcal{T}_{Acc}$)	**Student Accuracy (%) ($\mathcal{S}_{Acc}$)**
ADI[a] [21]	61.47	6.00
CMI[a] [6]	61.47	1.85
DAFL[b] [3]	60.83	35.46
DFAD[b] [5]	60.83	19.60
DFQ[a] [4]	61.47	41.30
CuDFKD[a] [14]	61.47	43.42
Ours-1 (w/ Memory Bank)	60.83	47.96
Ours-2 (w/ Generative Replay)	60.83	49.88

Table 3. Classification accuracy (in %) of the student trained using various DFKD methods on Tiny-ImageNet [13] with ResNet-34 [8] as the teacher and ResNet-18 [8] as the student. [a] and [b] denote results obtained from [14] and our implementation, respectively.

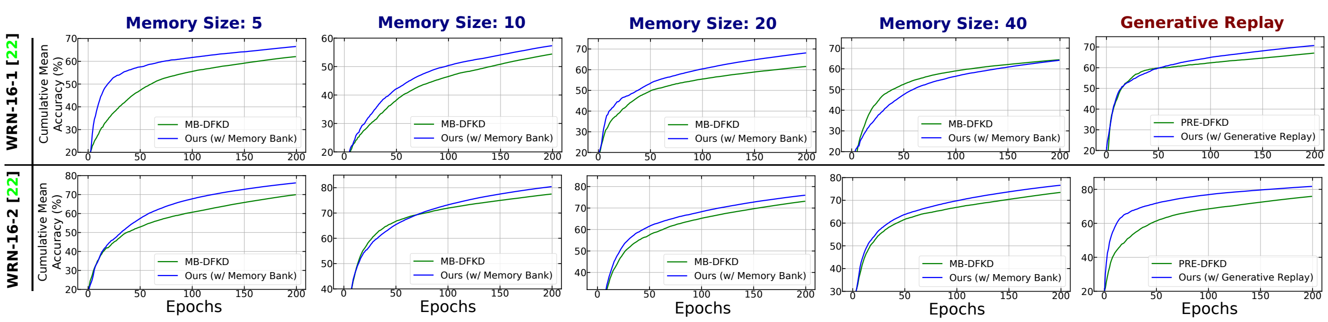

Figure 2. Cumulative Mean Accuracy (%) evolution of Wide-ResNet (WRN) [22]. The WRN-16-1 (*top-row*), and WRN-16-2 (*bottom-row*) networks are distilled by a WRN-40-2 teacher network pre-trained on CIFAR10 ($\mathcal{T}_{Acc} = 94.87\%$). Each column represent the learning curves with the Buffer-based (with different memory buffer sizes) and Generative replay schemes. The proposed method is in **Blue**.

Lemma 1. *If $\mathcal{L}_{Ret}$ has Lipschitz Hessian, i.e., $\|\nabla^2 \mathcal{L}_{Ret}(\theta_1) - \nabla^2 \mathcal{L}_{Ret}(\theta_2)\| \leq \rho \|\theta_1 - \theta_2\|$ for some $\rho > 0$, then:*

$$\nabla \mathcal{L}_{Ret}(\theta + \phi_\theta) = \nabla \mathcal{L}_{Ret}(\theta) + \nabla^2 \mathcal{L}_{Ret}(\theta)\phi_\theta + \mathcal{O}(\|\phi_\theta\|^2).$$

For instance, when $\phi_\theta = -\alpha \nabla \mathcal{L}_{Acq}(\theta)$, we have,

$$\nabla \mathcal{L}_{Ret}(\theta - \alpha \nabla \mathcal{L}_{Acq}(\theta)) = \nabla \mathcal{L}_{Ret}(\theta) - \alpha \nabla^2 \mathcal{L}_{Ret}(\theta) \nabla \mathcal{L}_{Acq}(\theta) + \mathcal{O}(\alpha^2).$$

Proof. Applying the fundamental theorem of calculus to each component of $\mathcal{L}_{Ret}$, we have:

$$\nabla \mathcal{L}_{Ret}(\theta + \phi_\theta) = \nabla \mathcal{L}_{Ret}(\theta) + \nabla^2 \mathcal{L}_{Ret}(\theta)\phi_\theta + \int_{k=0}^{1} (\nabla^2 \mathcal{L}_{Ret}(\theta + k\phi_\theta) - \nabla^2 \mathcal{L}_{Ret}(\theta))\phi_\theta dk. \tag{1}$$

Omitting the subscript Ret for brevity,

$$\implies \|\nabla\mathcal{L}(\theta+\phi_\theta)-(\nabla\mathcal{L}(\theta)+\nabla^2\mathcal{L}(\theta)\phi_\theta)\| = \|\int_{k=0}^{1}(\nabla^2\mathcal{L}(\theta+k\phi_\theta)-\nabla^2\mathcal{L}(\theta))\phi_\theta dk\| \tag{2}$$

$$\implies \|\nabla\mathcal{L}(\theta+\phi_\theta)-(\nabla\mathcal{L}(\theta)+\nabla^2\mathcal{L}(\theta)\phi_\theta)\| \le \int_{k=0}^{1}\|(\nabla^2\mathcal{L}(\theta+k\phi_\theta)-\nabla^2\mathcal{L}(\theta))\phi_\theta\| dk \tag{3}$$

$$\implies \|\nabla\mathcal{L}(\theta+\phi_\theta)-(\nabla\mathcal{L}(\theta)+\nabla^2\mathcal{L}(\theta)\phi_\theta)\| \le \int_{k=0}^{1}\rho\|k\phi_\theta\|.\|\phi_\theta\| dk \qquad \text{from } \rho\text{-Lipschitzness} \tag{4}$$

$$\implies \|\nabla\mathcal{L}(\theta+\phi_\theta)-(\nabla\mathcal{L}(\theta)+\nabla^2\mathcal{L}(\theta)\phi_\theta)\| \le \frac{\rho}{2}\|\phi_\theta\|^2. \tag{5}$$

□

Theorem 1. *If $\theta' = \theta - \alpha\nabla\mathcal{L}_{Acq}(\theta)$, denotes the one step gradient descent on θ with the objective $\mathcal{L}_{Acq}(\theta)$, where α is a scalar, and $\nabla\mathcal{L}_{Acq}(\theta)$ denotes the gradients of $\mathcal{L}_{Acq}$ at θ, then:*

$$\frac{\partial\mathcal{L}_{Ret}(\theta')}{\partial\theta} = \nabla\mathcal{L}_{Ret}(\theta) - \alpha\nabla^2\mathcal{L}_{Ret}(\theta).\nabla\mathcal{L}_{Acq}(\theta) - \alpha\nabla^2\mathcal{L}_{Acq}(\theta).\nabla\mathcal{L}_{Ret}(\theta) + \mathcal{O}(\alpha^2).$$

Proof. We have

$$\frac{\partial\mathcal{L}_{Ret}(\theta')}{\partial\theta} = \nabla\mathcal{L}_{Ret}(\theta').\frac{\partial\theta'}{\partial\theta} \tag{6}$$

$$\implies \frac{\partial\mathcal{L}_{Ret}(\theta')}{\partial\theta} = \nabla\mathcal{L}_{Ret}(\theta').\frac{\partial(\theta-\alpha\nabla\mathcal{L}_{Acq}(\theta))}{\partial\theta} \tag{7}$$

$$\implies \frac{\partial\mathcal{L}_{Ret}(\theta')}{\partial\theta} = \nabla\mathcal{L}_{Ret}(\theta').(I-\alpha\nabla^2\mathcal{L}_{Acq}(\theta)) \tag{8}$$

Using Lemma 1, we substitute the value of $\nabla\mathcal{L}_{Ret}(\theta')$, where $\theta' = \theta - \alpha\nabla\mathcal{L}_{Acq}(\theta)$ in (8), and obtain:

$$\frac{\partial\mathcal{L}_{Ret}(\theta')}{\partial\theta} = \overbrace{(\nabla\mathcal{L}_{Ret}(\theta)+\nabla^2\mathcal{L}_{Ret}(\theta).\underbrace{(\theta'-\theta)}_{=-\alpha\nabla\mathcal{L}_{Acq}(\theta)} + \underbrace{\mathcal{O}(\|\theta'-\theta\|^2)}_{=\mathcal{O}(\alpha^2)})}^{=\nabla\mathcal{L}_{Ret}(\theta')}.(I-\alpha\nabla^2\mathcal{L}_{Acq}(\theta)) \tag{9}$$

$$\implies \frac{\partial\mathcal{L}_{Ret}(\theta')}{\partial\theta} = \nabla\mathcal{L}_{Ret}(\theta)+\nabla^2\mathcal{L}_{Ret}(\theta).\underbrace{(\theta'-\theta)}_{=-\alpha\nabla\mathcal{L}_{Acq}(\theta)} - \alpha\nabla^2\mathcal{L}_{Acq}(\theta)\nabla\mathcal{L}_{Ret}(\theta)+\mathcal{O}(\alpha^2) \tag{10}$$

$$\implies \frac{\partial\mathcal{L}_{Ret}(\theta')}{\partial\theta} = \nabla\mathcal{L}_{Ret}(\theta)-\alpha\nabla^2\mathcal{L}_{Ret}(\theta)\nabla\mathcal{L}_{Acq}(\theta)-\alpha\nabla^2\mathcal{L}_{Acq}(\theta)\nabla\mathcal{L}_{Ret}(\theta)+\mathcal{O}(\alpha^2) \tag{11}$$

$$\implies \frac{\partial\mathcal{L}_{Ret}(\theta')}{\partial\theta} = \nabla\mathcal{L}_{Ret}(\theta)-\alpha\underbrace{(\overbrace{\nabla^2\mathcal{L}_{Ret}(\theta)\nabla\mathcal{L}_{Acq}}^{Hessian\ Product-1}-\overbrace{\nabla^2\mathcal{L}_{Acq}\nabla\mathcal{L}_{Ret}(\theta)}^{Hessian\ Product-2})}_{Gradient\ Matching}+\mathcal{O}.(\alpha^2) \tag{12}$$

Note that, Lemma 1 provides an efficient way to obtain $Hessian\ Product-1$ (highlighted in (12)) by computing the gradient of $\mathcal{L}_{Ret}$ at θ', thus, eradicating the time and memory overhead of explicitly computing $Hessian\ Product-1$. Hence, we have:

$$\frac{\partial\mathcal{L}_{Ret}(\theta')}{\partial\theta} = \nabla\mathcal{L}_{Ret}(\theta)-\alpha\nabla^2\mathcal{L}_{Ret}(\theta)\nabla\mathcal{L}_{Acq}(\theta)-\alpha\nabla^2\mathcal{L}_{Acq}\nabla\mathcal{L}_{Ret}(\theta)+\mathcal{O}(\alpha^2). \tag{13}$$

□

E. Relation to Continual-Learning and Adoption as Baselines

The fundamental objective of Continual-Learning (CL) is towards complete remembrance of previously acquired task knowledge. Our setting avoids losing previous knowledge (retention) by constraining the deviation of the current learning from the previous (Figure 1), by aligning them in the gradient space. It is important to note that the proposed method brings a unique perspective and contribution in the context of DFKD, where *discrete task boundaries* do not exist like in CL. Therefore, it is not straight forward to consider the methods such GEM [15], OML [10], and La-MAML [7] as baselines. GEM require task descriptors for CL and are not learned in an online fashion. OML introduces a meta-objective for pre-training the network to learn an optimal representation offline, which is subsequently frozen and used for CL. La-MAML uses a meta update strategy to learn a sparse set of learning-rates (LRs), for each individual parameter with multiple inner loops, for the final outer-loop update using those LRs. Nonetheless, the aforementioned methods do not specifically target DFKD.

F. Societal Impact

Similar to other DFKD methods, our method may be framed as an attack strategy to create clones of proprietary pre-trained models that are accessible online [19]. However, this work makes no such efforts and does not support such practices. Moreover, in the Undistillable or Nasty teacher setting [16, 20] the teacher network predictions are transformed to carry out knowledge-distillation. Hence, the method suggested by Jandial *et al.* [9] can be used as an add-on in our framework. Nonetheless, the presented method introduces no such objective for the teacher and will fail in the Undistillable teacher setting.

References

[1] Kuluhan Binici, Shivam Aggarwal, Nam Trung Pham, Karianto Leman, and Tulika Mitra. Robust and resource-efficient data-free knowledge distillation by generative pseudo replay. In *AAAI*, 2022. 1, 2, 3

[2] Kuluhan Binici, Nam Trung Pham, Tulika Mitra, and Karianto Leman. Preventing catastrophic forgetting and distribution mismatch in knowledge distillation via synthetic data. In *WACV*, 2022. 1

[3] Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Dafl: Data-free learning of student networks. In *ICCV*, 2019. 4

[4] Yoojin Choi, Jihwan Choi, Mostafa El-Khamy, and Jungwon Lee. Data-free network quantization with adversarial knowledge distillation. In *CVPR Workshops*, 2020. 4

[5] Gongfan Fang, Jie Song, Chengchao Shen, Xinchao Wang, Da Chen, and Mingli Song. Data-free adversarial distillation. In *CVPR*, 2020. 4

[6] Gongfan Fang, Jie Song, Xinchao Wang, Chen Shen, Xingen Wang, and Mingli Song. Contrastive model inversion for data-free knowledge distillation. In *IJCAI*, 2021. 2, 4

[7] Gunshi Gupta, Karmesh Yadav, and Liam Paull. Look-ahead meta learning for continual learning. *NeurIPS*, 2020. 6

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 4

[9] Surgan Jandial, Yash Khasbage, Arghya Pal, Vineeth N Balasubramanian, and Balaji Krishnamurthy. Distilling the undistillable: Learning from a nasty teacher. In *ECCV*, 2022. 6

[10] Khurram Javed and Martha White. Meta-learning representations for continual learning. *NeurIPS*, 2019. 6

[11] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019. 3, 4

[12] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009. 2

[13] Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015. 1, 2, 4

[14] Jingru Li, Sheng Zhou, Liangcheng Li, Xifeng Yan, Zhi Yu, and Jiajun Bu. How to teach: Learning data-free knowledge distillation from curriculum. *arXiv preprint arXiv:2208.13648*, 2022. 3, 4

[15] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *NeurIPS*, 2017. 6

[16] Haoyu Ma, Tianlong Chen, Ting-Kuei Hu, Chenyu You, Xiaohui Xie, and Zhangyang Wang. Undistillable: Making a nasty teacher that CANNOT teach students. In *ICLR*, 2021. 6

[17] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. 1, 2

[18] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018. 2

[19] Jean-Baptiste Truong, Pratyush Maini, Robert J Walls, and Nicolas Papernot. Data-free model extraction. In *CVPR*, 2021. 6

[20] Jingwen Ye, Yining Mao, Jie Song, Xinchao Wang, Cheng Jin, and Mingli Song. Safe distillation box. In *AAAI*, 2022. 6

[21] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *CVPR*, 2020. 4

[22] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 2, 4