A. Gradient Matching Ablation

In Figure 1, we monitor the alignment as the layer-wise average gradient Cosine Similarity (of $L_{Acq}$ and $L_{Ret}$) 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 PyTorch². 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 ($G$) 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 ($\alpha_{S}$) with cosine annealing starting from 1e-1 and annealing it at each epoch to 0 to optimize the student parameters ($\theta_{S}$). For the one-step gradient

¹https://github.com/kuluhan/PRE-DFKD
²https://pytorch.org/
Algorithm 1: Proposed DFKD method, with Memory-Buffer replay.

Input: $T_{\theta_T}, S_{\theta_S}, \mathcal{G}_{\theta_G}, \mathcal{M}, \mathcal{E}_{\text{max}}, \overline{I}, g, \alpha_G, s, \alpha_S, f$
Output: $S_{\theta_S}$

$\mathcal{E} = 1$

while $\mathcal{E} \leq \mathcal{E}_{\text{max}}$ do

for $\overline{I}$ iterations do

for $g$ iterations do

$z \sim \mathcal{N}(0, I)$

$L_G \leftarrow -D(T(\mathcal{G}_{\theta_G}(z)), S(\mathcal{G}_{\theta_G}(z))) + L_P(\mathcal{G}_{\theta_G}(z))$

$\theta_G \leftarrow \theta_G - \alpha_G \nabla_{\theta_G} L_G$

end

for $s$ iterations do

$z \sim \mathcal{N}(0, I)$

$\hat{x} \leftarrow \mathcal{G}_{\theta_G}(z)$

Compute $L_{\text{Acq}}(\theta_S)$ using $\hat{x}$

$L_S \leftarrow L_{\text{Acq}}(\theta_S)$

if $\mathcal{M}$ is not empty then

$\hat{x}_m \sim \mathcal{M}$

$\theta'_S \leftarrow \theta_S - \alpha \nabla_{\theta_S} L_{\text{Acq}}(\theta_S)$

Compute $L_{\text{Ret}}(\theta_S)$ and $L_{\text{Ret}}(\theta'_S)$ using $\hat{x}_m$

$L_S \leftarrow L_S + L_{\text{Ret}}(\theta_S) + L_{\text{Ret}}(\theta'_S)$

end

$\theta_S \leftarrow \theta_S - \alpha_S \nabla_{\theta_S} L_S$

end

if $\mathcal{E} \mod f == 0$ then

Update $\mathcal{M}$ with $x^*_m$, where, $x^*_m \subseteq \hat{x}$

end

$\mathcal{E} \leftarrow \mathcal{E} + 1$

end

descent, we use a learning rate ($\alpha$) of 0.9. Furthermore, we use Adam optimizer with a learning rate ($\alpha_G$) 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 ($\mathcal{E}_{\text{max}}$), 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 $\theta_S$ through $\theta'_S$. Moreover, we use a single gradient descent step to obtain $\theta'_S$, 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 \textit{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 \textit{et al.} [6]. For the ResNet-34 Teacher model, pretrained on CIFAR100 [12], we used the model made

\(^{1}\)https://github.com/zju-vipa/CMI
Algorithm 2: Proposed DFKD method, with Generative replay.

Input: $T_{\theta_T}, S_{\theta_S}, G_{\theta_G}, M, E_{\text{max}}, I, g, \alpha_G, s, \alpha_S, f, s_{gp}^{\text{max}}$

Output: $S_{\theta_S}$

$E = 1$

while $E \leq E_{\text{max}}$

for $I$ iterations do

for $g$ iterations do

$z \sim \mathcal{N}(0, I)$

$L_G \leftarrow -D(T(G_{\theta_G}(z)), S(G_{\theta_G}(z))) + L_P(G_{\theta_G}(z))$

$\theta_G \leftarrow \theta_G - \alpha_G \nabla_{\theta_G} L_G$

end

for $s$ iterations do

$z \sim \mathcal{N}(0, I)$

$\hat{x} \leftarrow G_{\theta_G}(z)$

Compute $L_{\text{Acq}}(\theta_S)$ using $\hat{x}$

$L_S \leftarrow L_{\text{Acq}}(\theta_S)$

$x_m \sim M$

$\theta_S' \leftarrow \theta_S - \alpha_S \nabla_{\theta_S} L_{\text{Acq}}(\theta_S)$

Compute $L_{\text{Ret}}(\theta_S)$ and $L_{\text{Ret}}(\theta_S')$ using $x_m$

$L_S \leftarrow L_S + L_{\text{Ret}}(\theta_S) + L_{\text{Ret}}(\theta_S')$

$\theta_S \leftarrow \theta_S - \alpha_S \nabla_{\theta_S} L_S$

$s_{gp} = 0$

if $E \mod f == 0$ and $s_{gp} \leq s_{gp}^{\text{max}}$ then

Train $M$ with $x_m$ and $x_m^*$, where, $x_m^* \subseteq \hat{x}$

$s_{gp} \leftarrow s_{gp} + 1$

end

end

$E \leftarrow E + 1$

end

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

<table>
<thead>
<tr>
<th>Output Size</th>
<th>Layers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>Noise ($z \sim \mathcal{N}(0, I)$)</td>
</tr>
<tr>
<td>$128 \times h/4 \times w/4$</td>
<td>Linear, BatchNorm1D, Reshape</td>
</tr>
<tr>
<td>$128 \times h/4 \times w/4$</td>
<td>SpectralNorm(Conv (3 $\times$ 3)), BatchNorm2D, LeakyReLU</td>
</tr>
<tr>
<td>$128 \times h/2 \times w/2$</td>
<td>UpSample (2$\times$)</td>
</tr>
<tr>
<td>$64 \times h/2 \times w/2$</td>
<td>SpectralNorm(Conv (3 $\times$ 3)), BatchNorm2D, LeakyReLU</td>
</tr>
<tr>
<td>$64 \times h \times w$</td>
<td>UpSample (2$\times$)</td>
</tr>
<tr>
<td>$3 \times h \times w$</td>
<td>SpectralNorm(Conv (3 $\times$ 3)), TanH, BatchNorm2D</td>
</tr>
</tbody>
</table>

available by Binci et al. [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 ($T_{\text{Acc}}$) of the methods proposed and reported by Li et al. [14], we compare our methods with the accuracies reported by them.
For instance, when applying the fundamental theorem of calculus to each component of $\nabla L \theta$.

**Proof.** Applying the fundamental theorem of calculus to each component of $L_{\text{Ret}}$, we have:

$$\nabla L_{\text{Ret}}(\theta + \phi) = \nabla L_{\text{Ret}}(\theta) + \nabla^2 L_{\text{Ret}}(\theta) \phi + \int_{k=0}^{1} (\nabla^2 L_{\text{Ret}}(\theta + k \phi) - \nabla^2 L_{\text{Ret}}(\theta)) \phidk.$$

(1)
we have:

\[ \nabla L(\theta + \phi_y) - (\nabla L(\theta) + \nabla^2 L(\theta)\phi_y) \] 

\[ = \left\| \sum_{k=0}^{1} (\nabla^2 L(\theta + k\phi_y) - \nabla^2 L(\theta))\phi_y dk \right\| \] 

\[ \Rightarrow \| \nabla L(\theta + \phi_y) - (\nabla L(\theta) + \nabla^2 L(\theta)\phi_y) \| \leq \sum_{k=0}^{1} \| (\nabla^2 L(\theta + k\phi_y) - \nabla^2 L(\theta))\phi_y \| dk \] 

\[ \Rightarrow \| \nabla L(\theta + \phi_y) - (\nabla L(\theta) + \nabla^2 L(\theta)\phi_y) \| \leq \sum_{k=0}^{1} \rho k||\phi_y||.\|\phi_y\| dk \] 

from \( \rho \)-Lipschitzness

\[ \Rightarrow \| \nabla L(\theta + \phi_y) - (\nabla L(\theta) + \nabla^2 L(\theta)\phi_y) \| \leq \frac{\rho}{2} ||\phi_y||^2. \] 

\[ \square \]

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

\[ \frac{\partial L_{Ret}(\theta')}{\partial \theta} = \nabla L_{Ret}(\theta) - \alpha \nabla^2 L_{Ret}(\theta) \nabla L_{Acq}(\theta) - \alpha \nabla^2 L_{Acq}(\theta) \nabla L_{Ret}(\theta) + O(\alpha^2). \]

Proof. We have

\[ \frac{\partial L_{Ret}(\theta')}{\partial \theta} = \nabla L_{Ret}(\theta'), \frac{\partial \theta'}{\partial \theta} \] 

\[ \Rightarrow \frac{\partial L_{Ret}(\theta')}{\partial \theta} = \nabla L_{Ret}(\theta'), \frac{\partial (\theta - \alpha \nabla L_{Acq}(\theta))}{\partial \theta} \] 

\[ \Rightarrow \frac{\partial L_{Ret}(\theta')}{\partial \theta} = \nabla L_{Ret}(\theta). (I - \alpha \nabla^2 L_{Acq}(\theta)) \] 

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

\[ \frac{\partial L_{Ret}(\theta')}{\partial \theta} = \nabla L_{Ret}(\theta) + \nabla^2 L_{Ret}(\theta). (\theta' - \theta) - \alpha \nabla^2 L_{Acq}(\theta) \nabla L_{Ret}(\theta) + O(\alpha^2) \] 

\[ \Rightarrow \frac{\partial L_{Ret}(\theta')}{\partial \theta} = \nabla L_{Ret}(\theta) + \nabla^2 L_{Ret}(\theta). (\theta' - \theta) - \alpha \nabla^2 L_{Acq}(\theta) \nabla L_{Ret}(\theta) + O(\alpha^2) \] 

\[ \Rightarrow \frac{\partial L_{Ret}(\theta')}{\partial \theta} = \nabla L_{Ret}(\theta) - \alpha \nabla^2 L_{Acq}(\theta) \nabla L_{Ret}(\theta) + O(\alpha^2) \] 

\[ \Rightarrow \frac{\partial L_{Ret}(\theta')}{\partial \theta} = \nabla L_{Ret}(\theta) - \alpha \nabla^2 L_{Acq}(\theta) \nabla L_{Ret}(\theta) + O(\alpha^2) \] 

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

\[ \frac{\partial L_{Ret}(\theta')}{\partial \theta} = \nabla L_{Ret}(\theta) - \alpha \nabla^2 L_{Ret}(\theta) \nabla L_{Acq}(\theta) - \alpha \nabla^2 L_{Acq} \nabla L_{Ret}(\theta) + O(\alpha^2). \] 

\[ \square \]

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 straightforward to consider the methods such as 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

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