1. Proof

1.1. Proof of Equation 1 in Main Paper

Proof of Eq. 1 We consider the standard supervised learning setting with n training data points drawn from some underlying distribution. The deep neural network is $F(\boldsymbol{\theta}, \boldsymbol{x})$, where $\boldsymbol{\theta}$ is the collection of all the parameters in the network, $\boldsymbol{x}$ denotes the inputs, $\boldsymbol{y}$ denotes the labels and τ is the continuous time index. The parameters $\boldsymbol{\theta}$ evolve in the following equation:

$$\frac{d\boldsymbol{\theta}(\tau)}{d\tau} = -\nabla\mathcal{L}(\boldsymbol{\theta}(\tau)) = -(F(\boldsymbol{\theta}, \boldsymbol{x}) - \boldsymbol{y})\frac{\partial F(\boldsymbol{\theta}(\tau), \boldsymbol{x})}{\partial\boldsymbol{\theta}}, \quad (1)$$

From Equation 1, the evolution of the network outputs can be written as:

$$\frac{dF(\boldsymbol{\theta}(\tau), \boldsymbol{x})}{d\tau} = -(F(\boldsymbol{\theta}, \boldsymbol{x}) - \boldsymbol{y})\langle\frac{F(\boldsymbol{\theta}(\tau), \boldsymbol{x})}{\partial\boldsymbol{\theta}}\frac{F(\boldsymbol{\theta}(\tau), \boldsymbol{x})}{\partial\boldsymbol{\theta}}\rangle, \quad (2)$$

We denote $\boldsymbol{u}(\tau) = F(\boldsymbol{\theta}(\tau), \boldsymbol{x})$ and use differentiation and chain rule. Thus, we can obtain the dynamics of $\boldsymbol{u}(\tau)$ as follows:

$$\frac{d\boldsymbol{u}(\tau)}{d\tau} = -\mathcal{K}(\tau)(\boldsymbol{u}(\tau) - \boldsymbol{y}), \quad (3)$$

where $\mathcal{K}(\tau) \in \mathbb{R}^{n\times n}$ is a positive define matrix, whose i, j-th entry is $\left\langle \frac{\partial F(\boldsymbol{\theta}(\tau), x_i)}{\partial\theta}, \frac{\partial F(\boldsymbol{\theta}(\tau), x_j)}{\partial\theta} \right\rangle$

1.2. Proof of Equation 9 in Main Paper

Proof of Eq. 9 Consider the outputs of current model $F(\boldsymbol{\theta}_T)$ and previous generic model $F(\boldsymbol{\theta}_{T-1})$ with respect to the same input $\boldsymbol{x} \in \mathcal{M}_t$. The training prediction is:

$$F_T(\boldsymbol{\theta}_T, \boldsymbol{x}) = \mathcal{K}_T^T\mathcal{K}_T^{-1}\boldsymbol{y}, \quad (4)$$

$$where, \mathcal{K}_T = \langle\frac{\partial F_T(\boldsymbol{x}, \boldsymbol{\theta}_T)}{\partial\boldsymbol{\theta}}, \frac{\partial F_T(\boldsymbol{x}, \boldsymbol{\theta}_T)}{\partial\boldsymbol{\theta}}\rangle, \quad (5)$$

$$F_{T-1}(\boldsymbol{\theta}_{T-1}, \boldsymbol{x}) = \mathcal{K}_{T-1}^T\mathcal{K}_{T-1}^{-1}\boldsymbol{y}, \quad (6)$$

$$where, \mathcal{K}_{T-1} = \langle\frac{\partial F_{T-1}(\boldsymbol{x}, \boldsymbol{\theta}_{T-1})}{\partial\boldsymbol{\theta}}, \frac{\partial F_{T-1}(\boldsymbol{x}, \boldsymbol{\theta}_{T-1})}{\partial\boldsymbol{\theta}}\rangle, \quad (7)$$

Where $\mathcal{K}_{T-1}$ is constant as $F_{T-1}(\boldsymbol{\theta}_{T-1})$ is fully trained and requires no updating when training F_T.

From Equation 7, we note that the only variable is the mapping function $\mathcal{K}_T$ since $\boldsymbol{x}, \boldsymbol{y}$ are the same for the previous model and the current model. Such a mapping function determines the output of the model. Thus, minimizing the cosine distance of two Jacobian matrices $\frac{\partial F_T(\boldsymbol{x}, \boldsymbol{\theta}_T)}{\partial\boldsymbol{\theta}}$ and $\frac{\partial F_{T-1}(\boldsymbol{x}, \boldsymbol{\theta}_{T-1})}{\partial\boldsymbol{\theta}}$ can minimize the difference between the current model and the previous model.

1.3. Gradients in training: TKIL vs. other methods

In this section, we need to prove the gradient updating direction of each method:

Training without task models:

$$g = \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}) = \nabla_{\theta,\phi}\mathcal{L}(\boldsymbol{\theta}, \phi, \boldsymbol{x}), \quad (8)$$

where $\boldsymbol{\theta}$ represents the a set of $\{\theta, \phi\}$.

KD with task models:

$$g_{KD} = \frac{1}{T}\sum_{i=1}^{T} g_{i,KD} \quad (9)$$

$$= \frac{1}{T}\sum_{i=1}^{T} \nabla\boldsymbol{\theta}_i\mathcal{L}_i^*(\boldsymbol{\theta}_i) \quad (10)$$

$$= \frac{1}{T}\sum_{i=1}^{T} \nabla_{\theta_i,\phi_i}\mathcal{L}_i^*(\theta_i, \phi_i) \quad (11)$$

where $\mathcal{L}^*$ denotes the Knowledge distillation loss.

while the GTK with task models:

$$g_{GTK} = \frac{1}{T}\sum_{i=1}^{T}(g_i + g_{GTK,i}) \quad (12)$$

$$= \frac{1}{T}\sum_{i=1}^{T} \nabla\boldsymbol{\theta}_i\mathcal{L}_i(\boldsymbol{\theta}_i) + \nabla_{\phi_i}\mathcal{L}_i(\phi_i) \quad (13)$$

$$= \frac{1}{T}\sum_{i=1}^{T} \nabla_{\theta_i,\phi_i}\mathcal{L}_i(\theta_i, \phi_i) + \nabla_{\phi_i}\mathcal{L}_i(\phi_i) \quad (14)$$

2. Inference Implementations

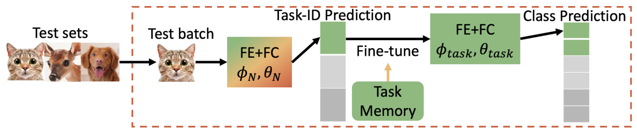

Figure 1. During Inference, the generic model generates task predictions when given the test sets for all samples. Then the predicted corresponding task model with the highest probability is finetuned from the generic model with exemplars from memory to predict the class.

During the inference, we adopt the task prediction procedure described in iTAML [5]. As we have mentioned, we finetune task model ($F(\phi_{task}, \theta_{task})$) from the generic model ($F(\phi_N, \theta_N)$) when given test samples x', since the generic model is not designed for any tasks and learns the unbiased representations. We capture the logistic outputs of the generic model and predict tasks based on the logistic outputs. Then the finetuned task model predicts the class classifications. The inference pipeline is shown in Figure 1.

3. Experiments Supplementary to Experiments Reported in the Main Paper

3.1. Experiment Settings

Memory Implementations. Table 2 provides additional information on the total number of samples used for var-

Table 1. Performance comparison between the TKIL and iTAML on CIFAR-100, 20 stages, 5 classes per stage, memory size $2,000$.

Task Accuracy	Task 1	Task 1-2	Task 1-3	Task 1-4	Task 1-5	Task 1-6	Task 1-7	Task 1-8	Task 1-9	Task 1-10
iTAML [5]	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	99.4%	99.5%	99.5%	100.0%
TKIL(ours)	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	99.6%
	Task 1-11	Task 1-12	Task 1-13	Task 1-14	Task 1-15	Task 1-16	Task 1-17	Task 1-18	Task 1-19	Task 1-20
iTAML [5]	98.9%	99.3%	96.0%	91.7%	91.6%	86.5%	87.7%	86.6%	85.5%	86.4%
TKIL(ours)	99.7%	100.0%	100.0%	99.9%	96.5%	97.0%	97.1%	96.2%	97.5%	**95.2%**
Classification Accuracy	Class 1-5	Class 1-10	Class 1-15	Class 1-20	Class 1-25	Class 1-30	Class 1-35	Class 1-40	Class 1-45	Class 1-50
iTAML [5]	89.8%	91.7%	92.1%	91.4%	91.2%	90.0%	89.2%	88.7%	87.3%	88.4%
TKIL(ours)	93.1%	92.2%	92.8%	92.4%	92.1%	92.4%	91.7%	90.75%	90.5%	89.6%
	Class 1-55	Class 1-60	Class 1-65	Class 1-70	Class 1-75	Class 1-80	Class 1-85	Class 1-90	Class 1-95	Class 1-100
iTAML [5]	86.8%	87.4%	83.7%	79.4%	78.3%	73.9%	75.3%	74.3%	73.2%	72.8%
TKIL(ours)	89.5%	90.1%	89.6%	88.8%	85.7%	83.5%	86.1%	85.4%	83.15%	**82.1%**

Table 2. Total Samples Budget for different settings.

Datasets	CIFAR-100					ImageNet-100					ImageNet-1k	
Fixed Memory Buffer	$2,000$					$2,000$					$20,000$	
Stages	5	10	25	5	10	5	10	25	5	10	5	10
New Classes per Stage	10	5	2	20	10	10	5	2	20	10	200	100
Total Samples per Stage	$7,000$	$4,500$	$3,000$	$12,000$	$7,000$	$14,000$	$8,000$	$4,400$	$26,000$	$14,000$	$260,000$	$140,000$

ious benchmarks per stage. The memory budget is determined by the stage number N and fixed memory buffer size. For instance, in the case of CIFAR-100 with 5 stages (10 classes per stage), the total number of samples is set to 7,000, which includes 2,000 samples from the fixed memory buffer, and the remaining samples come from 10 classes per phase, each with 500 samples per class.

Memory Size Selection. In table 3, we investigate the impact of memory size on TKIL in 10 stages, where each stage learns 10 new classes. We find that TKIL achieves robust SOTA accuracy for different memory sizes. When the memory size is limited to $\mathcal{M} = 1k$, TKIL does not exhibit a significant collapse in the accuracy curve, and outperforms 10.4% than the most comparable existing method, AFC, even with a larger memory size ($\mathcal{M} = 2k$). We chose $2k$ memory size to balance memory cost and performance.

Hyperparameter Tuning. In Table 4, we demonstrate the importance of GTK loss and the related parameter, γ. Since the sensitivity of α and β has been studied by previous works. We select an optimal setting from these experiments [2, 5]. We compare the TKIL method with its variants with parameters ($\gamma = 0.1, 1, 10$) to estimate the ability to retain the feature representations with it. In Table 5, we experimentally show that $\alpha = 0.7, \beta = 0.3$ works well in our approach. γ controls GTK loss and when selecting γ to be large, the model remembers the first task and could overfit to it and could fail to make task predictions. When letting $\gamma = 0$, the model forgets the optimal representations and diverges to the latest task, which leads to an unstable task prediction. Based on these results we select γ as $\gamma = 0.1$.

Optimizer and Epochs. In Table 6, we evaluated TKIL using three optimizers SGD, Adam, and RAdam, and obtained comparable performance across methods, 82%±0.8% on CIFAR-100 and 65%±0.5% on ImageNet-1k. We selected RAdam for our experiments for class balanced IL baselines [5]. We also note that the training duration varies **from 70 to 160 epochs** for different methods [10, 32, 43], while TKIL appears to be more robust for training for different numbers of epochs.

3.2. Comparisons

TKIL VS. iTAML. Table 1 investigates the accuracy of TKIL and the compared approach, iTAML [5], when 20 tasks are added incrementally with 5 classes each time on the CIFAR-100 benchmark. In these experiments, TKIL consistently outperforms the compared methods by a large margin (e.g. 82% vs. 72% in 20 stages scenario). The top part in Table 1 reports the task accuracy over seen classes on CIFAR-100, and the bottom part Table 1 reports classification accuracy. TKIL consistently outperforms other existing methods across different stages.

TKIL VS. SS-IL. We consider the mixed training paradigm as our upper bound baseline. For mixed training, we train a single model on the whole data $\mathcal{D}$ and then use the best model for the inference. Our accuracy is closer to the upper bounds. Notably, while this training paradigm achieves better performance than existing methods, they require re-training the whole dataset (as shown in Table 7).

Table 3. Ablation study of the fixed memory $\mathcal{M}$ on CIFAR-100 (10 stages, 10 new classes per stage)

Stages	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**
$\mathcal{M} = 1k$	92.2%	89.5%	87.3%	85.4%	84.3%	80.4%	78.7%	77.1%	76.6%	75.3%
$\mathcal{M} = 2k$	92.2%	89.7%	88.2%	87.4%	86.7%	85.7%	84.5%	83.7%	83.1%	82.5%

Table 4. Ablation study of the hyperparameter γ selection on CIFAR (5 stages, 10 new classes per stage, memory size = $2k$)

Parameter	**S1**	**S2**	**S3**	**S4**	**S5**
	Task-prediction over all seen classes				
$\gamma = 10$	100%	82.0%	33.5%	21.9%	17.5%
$\gamma = 1$	100%	100%	97.5%	94.3%	92.5%
$\gamma = 0.1$	100%	100%	100%	100%	100%

Table 5. Ablation study of the γ selection on CIFAR-100, 10 stages, 10 classes per stage, memory size $2,000$.

Parameter	**S1**	**S2**	**S3**	**S4**	**S5**
	Class-prediction at different Stages				
$\alpha = 0.5, \beta = 0.5, \gamma = 0$	92.2%	88.3%	83.2%	75.2%	70.4%
$\alpha = 0.5, \beta = 0.5, \gamma = 0.1$	92.2%	89.4%	85.3%	78.4%	77.4%
$\alpha = 0.6, \beta = 0.4, \gamma = 0.1$	92.2%	90.4%	85.3%	82.3%	79.5%
$\alpha = 0.7, \beta = 0.3, \gamma = 0.1$	92.2%	89.7%	88.2%	87.4%	86.7%

Table 6. Training Epochs: CIFAR-100 and ImageNet-1k, 10 tasks

Epochs	70	100	160
CIFAR-100	82.5%±0.3%	84.3%±0.3%	84.9%±0.2%
ImageNet-1k	65.7%±0.2%	66.8%±0.2%	67.5%±0.1%

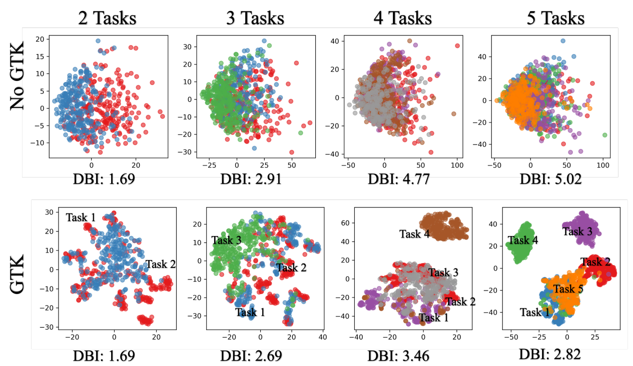

Figure 2. T-SNE visualization (colors indicate classes) of feature representations from the last feature extractor on CIFAR-100 with 5 Tasks and 10 new classes introduced in each stage. **Representations by GTK (bottom) are clustered more efficiently (supported by Davies-Bouldin Index (DBI - lower better))** than representations that do not use GTK (top).

Specifically, TKIL is closer to the upper bound and outperforms SS-IL in 5 and 10 tasks with CIFAR-100 and ImageNet-100, respectively.

Table 7. Performance comparison between the TKIL and SS-IL on CIFAR-100 and ImageNet dataset, memory size $2,000$.

Method	**CIFAR-100**		**ImageNet-1k**	
Stages	5	10	5	10
New classes per stage	20	10	20	10
iCaRL [6] (*baseline*)	57.2%	52.6%	51.5%	46.8%
SS-IL([1])	74.8%	71.1%	59.6%	59.4%
TKIL *(Ours)*	**79.5%**	**82.5%**	**66.9%**	**65.4%**
Upper Bound	84.6%		76.8%	

Memory-based VS. Non-Memory-based Class IL: In Table 8, we show side-by-side comparisons of non-memory based approaches and TKIL. While TKIL relies on memory (2k samples memory consisting **1-3%** of the total samples), it achieves more favorable accuracy in less epochs (**+20%∼30%**) than non-memory methods, thus requiring less epochs and less training load, defined as #samples×#epochs.

4. Qualitative Results

In Figure 2, we add t-SNE visualization with CIFAR-100 dataset. The feature representations are taken from the final layer of the feature extractor and are projected into a 2D space.

Table 8. Memory-based vs. non Memory-based Class IL

	Training Samples	Training Epochs	Accuracy
Non Memory-based [7, 4, 3]	$5k$ (CIFAR) $12k$ (ImageNet)	90-150 epochs	49.7% ∼ 54.7% 34.5% ∼ 39.4%
Memory-based (TKIL)	$5k$+$2k$ (CIFAR) $12k + 2k$ (ImageNet)	70 epochs	**82.5%(+27.8%)** **75.3%(+35.9%)**

Table 9. Comparison of different ordering strategies. We report mean and standard deviation across 5 random trials on the CIFAR-100, 10 incremental learning stages, 10 classes per stage, memory size $2,000$.

Random seed	2357	4305	5367	7275	8524	mean	deviation
Accuracy	82.49%	82.19%	82.45%	82.76%	81.94%	82.36%	0.085

References

[1] Hongjoon Ahn, Jihwan Kwak, Subin Lim, Hyeonsu Bang, Hyojun Kim, and Taesup Moon. Ss-il: Separated softmax for incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 844–853, 2021.

[2] Minsoo Kang, Jaeyoo Park, and Bohyung Han. Class-incremental learning by knowledge distillation with adaptive feature consolidation. *arXiv preprint arXiv:2204.00895*, 2022.

[3] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022.

[4] Grégoire Petit, Adrian Popescu, Eden Belouadah, David Picard, and Bertrand Delezoide. Plastil: Plastic and stable memory-free class-incremental learning. *arXiv preprint arXiv:2209.06606*, 2022.

[5] Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. itaml: An incremental task-agnostic meta-learning approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13588–13597, 2020.

[6] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[7] Xiaomeng Xin, Yiran Zhong, Yunzhong Hou, Jinjun Wang, and Liang Zheng. Memory-free generative replay for class-incremental learning. *arXiv preprint arXiv:2109.00328*, 2021.