TKIL: TANGENT KERNEL OPTIMIZATION FOR CLASS BALANCED INCREMENTAL LEARNING

Anonymous authors
Paper under double-blind review

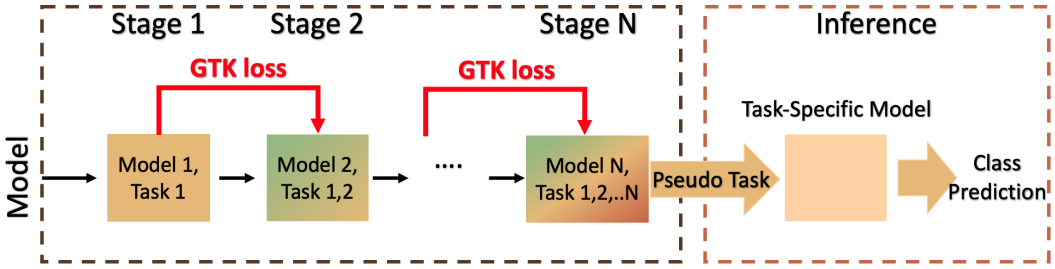

Figure 1: We propose a Tangent Kernel optimization for Incremental Learning (TKIL) with a novel Gradient Tangent Kernel (GTK) loss for optimal class balanced learning.

ABSTRACT

When sequentially learning multiple tasks, deep neural networks tend to loose accuracy on tasks learned in the past while gaining accuracy on the current task. This phenomenon is called catastrophic forgetting. Class Incremental Learning (CIL) methods address this problem by keeping a memory of exemplars from previous tasks aimed to assist with overall accuracy of all tasks. However, existing methods are not always able to balance the accuracy across all seen tasks and could overfit to the current task due to data imbalance between the complete training data points for the current task and the limited exemplars in the memory buffer for previous tasks. Here, we propose to address the data imbalance by learning a set of generalized non-task-specific parameters in addition to task-specific models. In particular, we propose a novel methodology of Tangent Kernel for Incremental Learning (TKIL) that seeks an equilibrium between current and previous representations. We achieve such equilibrium by computing and optimizing for a new Gradient Tangent Kernel (GTK). Specifically, TKIL tunes task-specific parameters for all tasks with GTK loss. Therefore, when representing previous tasks, task-specific models are not influenced by the samples of the current task and are able to retain learned representations. As a result, TKIL equally considers the contribution from all task models. The additional generalized parameters that TKIL obtains allow it to automatically identify which task is being considered and to adapt to it during inference. Extensive experiments on 5 CIL benchmark datasets with 10 incremental learning settings show that TKIL outperforms existing state-of-the-art methods, e.g., a 9.4% boost on CIFAR100 with 25 incremental stages. Furthermore, TKIL attains strong state-of-the-art accuracy on large-scale datasets, with a much smaller model size (36%) compared to other approaches.

1 INTRODUCTION

Visual content is evolving and its volume is rapidly increasing. Indeed, high-quality and large-scale visual media streaming data could easily become unfeasible to store and process in completeness (Cheng & Wang (2011)). Therefore, learning agents that do not require access to complete data and continuously learn new data are appealing. Class Incremental Learning (CIL) addresses the

development of such methods, where agents are expected to learn with incremental arrival of new training data and limited exemplars from previous tasks (Rebuffi et al. (2017)).

This learning scenario is significantly different from conventional learning, especially for classification tasks, due to the data imbalance between current and previous tasks. Since full past data is not retained, there is significantly less data available for them than for the current task. Therefore, CIL aims to achieve an equilibrium between current and previous representations. While current advanced methods propose additional finetuning steps to correct this imbalance, e.g., balanced finetuning or task-level bias rectification, training a fixed model for all tasks limits the ability to deal with imbalanced data (Ahn et al. (2021); Kang et al. (2022)).

Here, we take a different approach based on the principle of attaining generic representations that are task-independent. Instead of training a fixed model for all tasks, we propose to learn a set of generalized parameters as a generic model. The model predicts the task that is at hand and would adapt to the corresponding task automatically during inference. To achieve that, the gradients with respect to the parameters of the generic model are supposed to consider the contributions of all tasks in a balanced way. Motivated by Neural Tangent Kernel (NTK) theory that captures gradients (with respect to parameters) into a non-trained deterministic kernel to represent an unbiased trained network, we aim to find counterpart kernels in CIL where kernels represent previously trained unbiased model and are expected to assist training a new unbiased generic model (Jacot et al. (2018)).

Adaptation of NTK to incremental learning is not a direct implementation. The reason for this challenge stems from tangent kernels designed for ultra-wide neural networks only, while networks in CIL are finite-width (Li et al. (2019)). To overcome this challenge, we propose a novel Tangent Kernel optimization approach for Incremental Learning (TKIL). Specifically, we propose a novel Gradient Tangent Kernel (GTK) for CIL finite-width neural networks where GTK is a cosine similarity kernel between the gradients (with respect to parameters) from previous and current models. During updates of the generic model, TKIL tunes with GTK loss for different task-specific models representing current and previous tasks and takes the average of task-specific updates as the update of the generic model. In such updates, the GTK loss is expected to rectify the overfitting, if it occurs in the generic model. During inference, such a generic model cooperates with the inference pipeline to predict the task at hand when given testing points and adapts itself to this task to perform classifications.

In summary, our main contributions are: **1)** We propose a novel class incremental learning approach, TKIL, that addresses the imbalances in memory-based incremental learning. **2)** The core of TKIL is a novel tangent kernel (GTK) for finite-width neural networks. We mathematically prove that minimizing GTK loss associated with TKIL, expressed as the cosine distance of two Jacobian matrices, avoids local optimums and achieves balanced representations. Such balanced representations allow TKIL to generate robust task predictions and to update to corresponding task-specific models during inference. **3)** Extensive experiments on MNIST, SVHN, CIFAR-100, and ImageNet show that TKIL achieves robust accuracy on task predictions and this accuracy translates to outperforming existing incremental learning methods, especially in multi-class incremental learning scenarios.

2 Related Work

Class Incremental Learning typically includes three components: ***Exemplars Selection With Rehearsal*** (Rebuffi et al. (2017); Castro et al. (2018); Hou et al. (2019)), ***Forgetting Constraints*** (Kirkpatrick et al. (2017); Li et al. (2020); Chaudhry et al. (2018); Aljundi et al. (2018)), and ***Bias Corrections*** (Wu et al. (2019); Belouadah & Popescu (2019); Hou et al. (2019); Zheng et al. (2020)). In addition, ***End-to-End IL*** (Castro et al. (2018)) proposed a finetuning phase to boost the overall accuracy. In summary, consequent works contributed to the enhancement and combination of these four components. As a result, the exemplar-based method combined with forgetting constraints has been shown to be the current leading approach (Ahn et al. (2021)). We describe these two components in detail below.

- ***Exemplars Selection With Rehearsal.*** Rehearsal methods store *a small set of previous task exemplars* in a memory buffer to represent previously learned tasks. Multiple methods proposed herding heuristics (Welling (2009)) to select the most representative exemplars (Rebuffi et al. (2017); Castro et al. (2018); Hou et al. (2019)). Additional methods estimated the distribution of

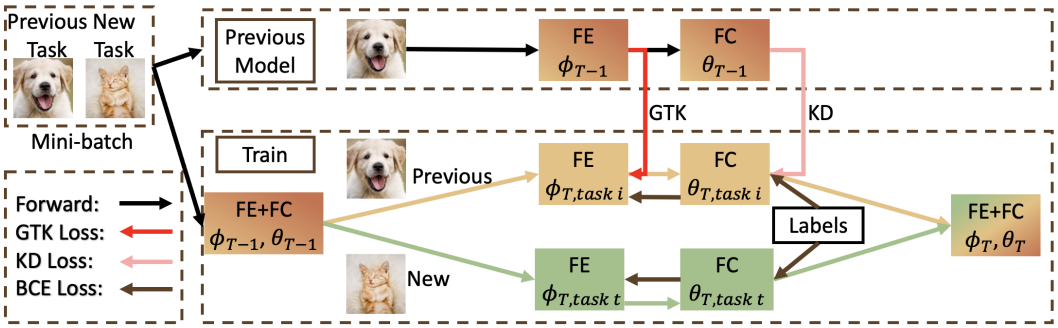

Figure 2: Illustration of Tangent kernel Incremental Learning approach. In the mini-batch, we separate training steps into different tasks. 1) For the previous tasks (task $i, i = 1, 2, 3, ...N-1$), we optimize task-specific models with BCE loss (brown color), knowledge Distillation loss (pink color), and the Gradient Tangent kernel loss (red loss) with exemplars from a fixed memory; 2) For the current task (task T), only employs BCE loss (brown loss backward) (FE: Feature extractor layers, FC: Fully Connected layers)

previously learned tasks and generated extra pseudo-exemplars or images to avoid the imbalance between classes (Liu et al. (2020); Ostapenko et al. (2019); Odena et al. (2017)). Our approach does not rely on a particular selection of exemplars to boost accuracy, and therefore, sampling is performed from each class randomly.

- ***Forgetting Constraints.*** Various regularization terms have been added to the classification loss to constrain forgetting of previous representations (Kirkpatrick et al. (2017); Li et al. (2020); Chaudhry et al. (2018); Aljundi et al. (2018)). In addition, Knowledge Distillation (KD) has been proposed to restore previous knowledge representations (Rebuffi et al. (2017); Castro et al. (2018); Li & Hoiem (2017); Ahn et al. (2021)). Furthermore, Adaptive Feature Consolidation (AFC) (Kang et al. (2022)) further modified KD to define the discrepancy loss, which estimates representation changes and retains important representation features. While KD loss turned out to be effective, it appears that KD alone cannot fully resolve imbalanced data distributions. Thus, our method tunes different task-specific models with GTK loss to learn previous and current tasks separately. As such, TKIL is able to obtain unbiased knowledge representations for all tasks.

Neural Tangent Kernels (NTK). NTK were first introduced in Jacot et al. (2018) and developed in multiple subsequent works (Arora et al. (2019); Li et al. (2019); Novak et al. (2019); Wallace et al. (2021)). NTK employs kernels gradient to describe the convergence behavior of over-parameterized DNNs in a limit of infinite width such that it can mimic the accuracy in this limit. A recent work introduced Task Tangent Kernel (TTK) (Wallace et al. (2021)), which uses a kernelized distance across the gradients of multiple random initialized networks to estimate the similarity over different tasks. While NTK and related methods motivated our work, infinite-width neural networks do not apply to incremental learning and finite-width networks require a novel tangent kernel framework for such a scenario.

3 METHODS

Given an imbalanced training distribution at each stage, incremental learning aims to train a unified classifier sequentially for all seen tasks. To this end, we propose a Tangent Kernel Incremental Learning approach for continually training an unbiased generic model. Figure 2 illustrates an overview of our approach. In the following subsections, we describe the approach by introducing notations in Section 3.1, and defining tangent kernels for CIL finite-width networks in Section 3.2. To resolve imbalances, TKIL tunes task-specific models with GTK loss and combines all task models into a final generic model in Section 3.3. We compare TKIL with other related methods in Section 3.4.

3.1 NOTATIONS

We denote a sequence of data as a batch of datasets $\mathcal{D}=\{D_1, D_2, \ldots, D_N\}$ for N tasks. $D_{T,\ (T\leq N)}$ is T-th dataset in $\mathcal{D}$, which contains the training images $\boldsymbol{x}=\{x_i\}_{i=1}^n$ and labels $\boldsymbol{y}=\{y_i\}_{i=1}^n$. Each $D_{T,\ (T\leq N)}$ dataset includes m classes, with total $m \times N$ classes in $\mathcal{D}$. At T-th incremental stage, only complete data for current classes D_T and a small set of previous exemplars $\mathcal{M}_T = \{M_1 \subseteq D_1, M_2 \subseteq D_2, \ldots, M_{T-1} \subseteq D_{T-1}\}$ in a fixed memory buffer are available for training. We denote the T-th generic model as $F_T = F(\boldsymbol{\theta}_T, \boldsymbol{x})$, where $\boldsymbol{\theta}_T = \{\theta_T, \phi_T\}$ is all parameters in the network and $\{\theta_T, \phi_T\}$ denote the parameters in feature extractor layers and fully connected layers, respectively. We use $\phi_{T,\ task\ i}$ and $\theta_{T,\ task\ i}$ as the parameters for the task-specific model i in T-th incremental stage. We denote $F(\boldsymbol{\theta})$ to represent a finite-width neural network and $\boldsymbol{\theta}$ as corresponding parameters when not involving incremental learning.

3.2 GRADIENT TANGENT KERNEL

In this section, we describe a finite-width deep neural network as our generic model and show that training the T-th model in CIL with the assistance of the previous $(T-1)$-th model gives rise to the minimization of cosine distance of Gradient Tangent Kernel (GTK). GTK inherits its name from Neural Tangent Kernel (NTK), since NTK theory shows that randomly initialized ultrawide deep neural network trained with gradient descent with infinitesimal step size is equivalent to a kernel regression with a deterministic kernel (Arora et al. (2019)). However, in contrast to NTK, we consider a more general case of tangent kernels for finite-width neural networks. Therefore, we establish a similar property in Lemma 1 below (with the corresponding proof in the Appendix A.1).

Lemma 1. Consider minimization of the squared loss $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^n (F(\boldsymbol{\theta}, x_i) - y_i)^2$ by gradient descent with infinitesimally small learning rate $\frac{d\boldsymbol{\theta}(t)}{dt} = -\mathcal{L}(\boldsymbol{\theta}(t))$. Let $\boldsymbol{u}(t) = F(\boldsymbol{\theta}(t), \boldsymbol{x})$ be the network outputs at time t and $\boldsymbol{y}$ is the ground truth. $\boldsymbol{u}(t)$ follows the evolution with respect to a kernel $\mathcal{K}(t)$ with the dynamics of $\boldsymbol{u}(t)$ as follows

$$\frac{d\boldsymbol{u}(t)}{dt} = -\mathcal{K}(t)(\boldsymbol{u}(t) - \boldsymbol{y}),\ \textit{where}\ \mathcal{K}(t) = \left\langle \frac{\partial F(\boldsymbol{\theta}(t), \boldsymbol{x})}{\partial \boldsymbol{\theta}}, \frac{\partial F(\boldsymbol{\theta}(t), \boldsymbol{x})}{\partial \boldsymbol{\theta}} \right\rangle. \tag{1}$$

Note that the above dynamics are identical to dynamics of kernel regression under the gradient flow (Assuming $\boldsymbol{u}(0) = 0$). Thus, the output of a trained neural network at time t for any testing input x' is

$$F(\boldsymbol{\theta}(t), x') = \mathcal{K}_t(x', \boldsymbol{x})^T \mathcal{K}_t(\boldsymbol{x}, \boldsymbol{x})^{-1} \boldsymbol{y}, \tag{2}$$

where we have extended the notation from $\mathcal{K}(t)$ to $\mathcal{K}_t(\boldsymbol{x}, \boldsymbol{x})$ to include additional terms. From Eq. 2, we can obtain training predictions when we set the inputs to be the training data points, i.e., $x' = \boldsymbol{x}$,

$$F(\boldsymbol{\theta}(t), \boldsymbol{x}) = \mathcal{K}_t(\boldsymbol{x}, \boldsymbol{x})^T \mathcal{K}_t(\boldsymbol{x}, \boldsymbol{x})^{-1} \boldsymbol{y}. \tag{3}$$

Alternatively to NTK theory which shows that $\mathcal{K}_t(\boldsymbol{x}, \boldsymbol{x})$ is deterministic during training (Arora et al. (2019)), here we focus on finite-width networks and aim to customize the tangent kernel for training an unbiased generic model. From Eqs. 1 and 3, we note that the only variable in training predictions is the mapping function $\mathcal{K}_t$ determined by the Jacobian matrix $\frac{\partial F(\boldsymbol{\theta}(t), x)}{\partial \boldsymbol{\theta}}$ such that the Jacobian matrix solely determines the training outputs $\boldsymbol{u}(t)$.

In incremental learning, the current model F_T is supposed to address a new task while preserving previously learned knowledge representations. We thus proceed and customize Eq. 3 for incremental learning and show in Theorem 1 that minimization of GTK transfers previous knowledge to F_T. The proof of Theorem 1 is included in Appendix A.2.

Theorem 1. Consider the training of T-th model F_T with a previously fully trained model F_{T-1}. While the T-th task is being learned, the kernels $\mathcal{K}_{T-1}$ and $\mathcal{K}_T$ are the tangent kernels with training points $\boldsymbol{x}$, respectively. Minimization of the cosine distance between training outputs of F_{T-1} and F_T with the same training data points $\boldsymbol{x}$ transfers the learned representations of F_{T-1} to F_T. In a kernel formulation, such optimization corresponds to GTK loss, i.e. minimization of the cosine

Algorithm 1 TKIL Algorithm in one mini-batch

Require: Dataset: D_T, Memory: $\mathcal{M}_T$, Note mini-batch $B_T \subseteq \{D_T \cup \mathcal{M}_T\}$ Hyperparameters: α, β, γ

Initialize the Model (For the first mini-batch):
  $F_T(\phi_T, \theta_T)$: *Previous Model:* $F_{T-1}(\phi_{T-1}, \theta_{T-1}) \longrightarrow F_T(\phi_T, \theta_T)$
for All training Samples in B_T **do**
  for $i = 1, 2, 3, ..T$ **do**
    $F_T \longrightarrow F_{T,task\,i}$
    if $i < T$ **then**
      $\mathcal{L}_{\text{Class}} = \sum_j^{task\,i} \mathcal{L}_{\text{BCE}}(F_{T,task\,i}(\phi_{T,task\,i}, \theta_{T,task\,i}, B_{T,j}), y_j)$
      $\mathcal{L}_{\text{KD}} = \sum_j^{task\,i} \mathcal{L}_{\text{BCE}}(F_{T,task\,i}(\phi_{T,task\,i}, \theta_{T,task\,i}, B_{T,j}), F_{T-1}(\phi_{T-1}, \theta_{T-1}, B_{T,j}))$
      $\mathcal{L}_{\text{GTK}} = \sum_j^{task\,i} \mathcal{L}_{\text{BCE}}(G_{T,task\,i}(\phi_{T,task\,i}, B_{T,j}), G_{T-1}(\phi_{T-1}, B_{T,j}))$
      $\mathcal{L}_{\text{all}} = \alpha\mathcal{L}_{\text{Class}} + \beta\mathcal{L}_{\text{KD}} + \gamma\mathcal{L}_{\text{GTKL}}$
      $F_{T,task\,i} = (\phi_{T,task\,i}, \theta_{T,task\,i})$ by minimizing the $\mathcal{L}_{\text{all}}$
    else if $i = T$ **then**
      $\mathcal{L}_{\text{Class}} = \sum_j^{task\,i} \mathcal{L}_{\text{BCE}}(F_{T,task\,i}(\phi_{T,task\,i}, \theta_{T,task\,i}, B_{T,j}), y_j)$
      $F_{T,task\,T} = (\phi_T, \theta_T)$ by minimizing the $\mathcal{L}_{\text{class}}$
    end if
  end for
  $F_T \longleftarrow \frac{1}{T}\sum_i^T F_{T,task\,i}$
end for

distance between two Jacobian matrices $\frac{\partial F(\boldsymbol{x}, \boldsymbol{\theta}_{T-1})}{\partial \boldsymbol{\theta}}$ and $\frac{\partial F(\boldsymbol{x}, \boldsymbol{\theta}_T)}{\partial \boldsymbol{\theta}}$.

$$\min_{\boldsymbol{\theta}_T} \langle F_{T-1}(\boldsymbol{\theta}_{T-1}, \boldsymbol{x}), F_T(\boldsymbol{\theta}_T, \boldsymbol{x})\rangle = \min_{\boldsymbol{\theta}_T} \langle \mathcal{K}_T \mathcal{K}_T^{-1} \boldsymbol{y}, \mathcal{K}_{T-1} \mathcal{K}_{T-1}^{-1} \boldsymbol{y}\rangle, \tag{4}$$

$$\Rightarrow \min_{\boldsymbol{\theta}_T} \langle \frac{\partial F(\boldsymbol{x}, \boldsymbol{\theta}_{T-1})}{\partial \boldsymbol{\theta}}, \frac{\partial F(\boldsymbol{x}, \boldsymbol{\theta}_T)}{\partial \boldsymbol{\theta}}\rangle, \tag{5}$$

Recent results from NTK for finite-width networks indicate that constraining the gradients in the last feature layer is sufficient to extract representations across different tasks (Wallace et al. (2021)). Thus, we propose Gradient Neural Tangent Kernel loss (GTK loss) on feature layers $g(\phi, \boldsymbol{x})$, where ϕ are parameters in feature layers. Based on the Eq. 5, we define the kernel objective function as follows

$$\min_{\phi_t} \mathbb{E}_{(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{M}_T}[\mathcal{L}_{GTK}(\frac{\langle G_T, G_{T-1}\rangle}{\|G_T\|\|G_{T-1}\|})], \textit{where}, G_{T-1} = \frac{\partial g(\boldsymbol{x}, \phi_{T-1})}{\partial \phi}, G_T = \frac{\partial g(\boldsymbol{x}, \phi_T)}{\partial \phi}, \tag{6}$$

where G_{T-1} and G_T are the gradients of the last feature layer from the previous and current model, respectively.

3.3 Tangent Kernel Incremental Learning Approach

Incremental learning progressively learns N tasks with m classes per task. From notations defined in Section 3.1, we consider $F_T(\boldsymbol{\theta}_T)$ as T-th stage generic model with feature extractor layers $g_T(\phi_T)$ and fully connected layers $f_T(\theta_T)$. Training the first task ($T = 1$) is straightforward. We initialize the parameters of the generic model ($\boldsymbol{\theta}_1$) with Gaussian distribution and optimize the model with D_1 with BCE loss.

The tangent kernel optimization approach (see Algorithm 1) is used to train subsequent T-th tasks ($T \in [2, N]$). In particular, before considering a new task T, the generic model obtained at $T - 1$ is replicated into T duplicates where each duplicate will correspond to a task-specific model up to task T. The different task-specific models are being trained on their corresponding tasks whether from memory buffer (previous tasks) or from current task data. At the end of the training, the task-specific models are collapsed into a single generic model that can accommodate T tasks. We describe the training procedures of the tasks-specific models and the generic model in detail below.

At the beginning of training T-th generic model, we initialize F_T with parameters of the previous generic model ($\boldsymbol{\theta}_{T-1}$). The training data points are from $\mathcal{M}_T \cup D_T$, where $\mathcal{M}_T$ are training samples

of previous tasks and D_T are training samples of the current task. We denote **Task-Specific Model Training** and **Generic Model Update** as one step, and then, repeat such a step to find an optimal T-th generic model.

Task-Specific Model Training: When a new task is considered for the T-th incremental learning stage, TKIL first trains T task-specific models. Each task-specific model $F_{T,task\,i}, i \leq T$ is initialized with parameters of current generic model ($\boldsymbol{\theta}_T$). The training data points are sampled from $B_T = \mathcal{M}_T \cup D_T$, such that the mini-batch B_T is the union of previous exemplars and current task data. The sampled data points from B_T are being separated into different tasks and for each task i, an i-th task-specific model $F_{T,task\,i}$ is created and is trained only with data samples from B_T that correspond to this task. To transfer learned representations from the previous generic model to task-specific models, we employ different losses as follows.

(1) For task-specific models representing previous tasks ($F_{T,task\,i}, i \in [1, T-1]$), we use three losses: Classification, Knowledge Distillation (KD) and GTK. Classification loss minimizes the difference between predicted logits $F(\boldsymbol{x})$ and labels $\boldsymbol{y}$. KD Loss penalizes the change with respect to the output from the previous expert model as a BCE loss. GTK loss rectifies the gradients and avoids divergence from the learned feature representations. Thus, the overall objective function $\mathcal{L}_{all}$ contains three loss functions, expressed as (also described in Algorithm 1, lines 6-11):

$$\min_{F} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}[\alpha\mathcal{L}(F(\boldsymbol{x}), \boldsymbol{y}) + \beta\mathcal{L}(F_{T-1}(\boldsymbol{x}), F(\boldsymbol{x})) + \gamma\mathcal{L}(\frac{\langle G_T, G_{T-1}\rangle}{\|G_{T-1}\|\|G_T\|}], \tag{7}$$

Where α, β and γ are hyperparameters.

(2) For the T-th task-specific model, $F_{T,task\,T}$, we employ the classification loss only as it is the first time that this task is being learned. The loss is expressed as (also shown in Algorithm 1, lines 13-14)

$$\min_{F} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}[\mathcal{L}(F(\boldsymbol{x}), \boldsymbol{y})]. \tag{8}$$

Generic Model Update: At the end of each mini-batch training, the trained task-specific models are being collapsed to a single updated generic model F_T, which represents the average direction of all task-specific models ($F_{T,task\,i}, i \in [1, T]$). Specifically, we update the parameters by capturing the average of all parameters from the task-specific models as parameters of the generic model. The average of task-specific models is computed as follows (Algorithm 1, line 17)

$$F_T = \frac{1}{T}\sum_{i}^{T} F_{T,task\,i}. \tag{9}$$

As the training progresses, the generic model is enhanced by learning new tasks while simultaneously preserving previous representations.

3.4 TANGENT KERNEL APPROACH VS. OTHER METHODS

To further interpret TKIL and its effectiveness, we demonstrate in Figure 3 how GTK loss updates the generic model parameters such that these will fall in the optimal area. The complementary calculation of the gradients for different methods appears in Appendix A.3.

Training a generic model with all samples in B_t together will result in a biased model due to the imbalanced data distribution in M_T and D_T. The gradient for such training is $\nabla_{\theta,\phi}\mathcal{L}(\theta, \phi, \boldsymbol{x})$. To address this imbalance distribution in each incremental learning stage, existing methods train task-specific models with a standard loss (like MSE or KD). The gradient of such models is $\frac{1}{T}\sum_{i=1}^{T} \nabla_{\theta_i,\phi_i}\mathcal{L}_i^*(\theta_i, \phi_i)$, where $\mathcal{L}^*$ denotes MSE or KD loss (shown as blue arrow per task-specific model in Figure 3 (a)). It is typically the case that the current task (navy blue) will skew the gradients of the generic model (green arrow) when the task-specific models are being collapsed to it, leading to a non-optimal generic model (located outside the green area).

TKIL approach introduces an additional loss, GTK loss (red transformation in Figure 3 (b)), which adapts the gradients of previous tasks. Therefore, the new generic model, collapsed from the rectified task-specific models, will be balanced and more optimal as shown by the green arrow being positioned in the expected green area in Figure 3 (b). In effect, the rectification of task-specific models corresponding to previous tasks with GTK loss allows TKIL to avoid local optimums and stay in the balanced optimal position for the generic model as it learns novel incremental tasks.

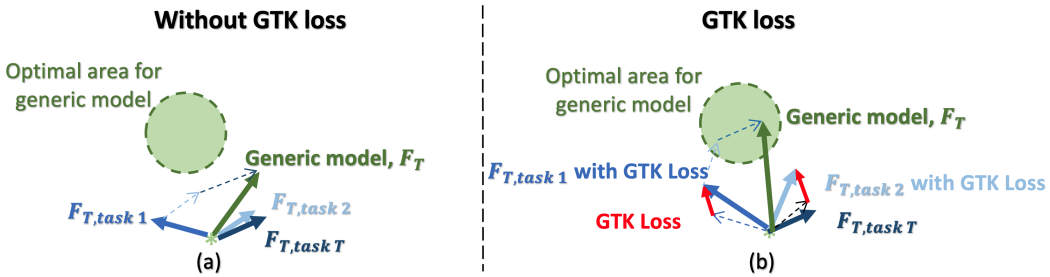

Figure 3: An illustration of how TKIL moves model parameters when learning a new task T, e.g., T=3. Assume the green dash circle is the optimal parameters area for all tasks. The objective is to constrain the parameters of the generic model (green arrow) in that circle. (a) Training task-specific models without GTK loss and taking the average direction of the task is non-optimal (located outside the green area). (b) TKIL rectifies updating directions of the previously learned tasks (Task i, $i \in [1, T-1]$) with GTK loss (red arrows) to keep the generic model in the optimal circle.

4 Experiments and Results

4.1 Datasets and Implementation

We conduct experiments on three popular benchmarks, i.e., CIFAR (Krizhevsky et al. (2009)), ImageNet-100 and ImageNet-1K (Russakovsky et al. (2015) with different incremental learning scenarios. CIFAR100 dataset contains $50,000$ training images and $10,000$ testing images. ImageNet-1k dataset consists of $1,281,167$ images for training and $50,000$ images for validation across $1,000$ classes. ImageNet-100 dataset includes 100 randomly sampled classes from ImageNet-1k. Experiments on additional two benchmarks of MNIST (Deng (2012)) and SVHN (Netzer et al. (2011)) are included in the Appendix.

We employ a ResNet-18 as the generic model. For all datasets, we use RAdam (Liu et al. (2019)) with an initial learning rate of 0.01 for 70 epochs, which is notably less than previous methods, e.g., PODNet Douillard et al. (2020) requires 160 epochs. The batch size is set to 128 for CIFAR and 64 for ImageNet. The learning rate is divided by 10 after every 20 epochs. The fixed exemplar memory is 2k for MNIST, SVHN, CIFAR100, and ImageNet-100, and 20K for ImageNet-1K. Our inference pipeline records the task predictions first and then finetunes the generic model to the corresponding task model to perform the classification. Implementation details are included in Appendix A.4. All models are trained on 2080Ti GPUs with a parallel computation mode. The previous method is implemented based on the publicly available official code (Rajasegaran et al. (2020); Javed & Shafait (2018)). For fair comparisons with previous methods, we report the average incremental accuracy over seen classes across all incremental stages with the same setting.

4.2 Results: CIFAR-100 and ImageNet-100

We test TKIL on various incremental learning scenarios and compare it with existing methods on CIFAR-100 and ImageNet-100 in Table 1. TKIL achieves accuracy that is more optimal than other compared methods on all compared scenarios. The margin in the accuracy of TKIL vs. existing approaches is particularly evident in the large tasks (stages) scenario which is consistent with the intent of TKIL to balance performance across tasks and classes. For example, on CIFAR-100 with 25 incremental stages, TKIL achieves an improvement of 9.4% in accuracy vs. the second best method AFC. In such a scenario, the accuracy of the existing state-of-the-art method, iTAML, drops to 55.9% from 76.6% due to unstable prediction of tasks, where AFC becomes a leading existing method among existing methods with 64.1%. In addition, we conduct similar experiments on MNIST and SVHN (in Appendix, Table 5). TKIL achieves 97% or higher accuracy on these benchmarks when learning 2 classes each time. These experiments show that TKIL predicts consistently stable and accurate task predictions. As a result, TKIL translates to more optimal accuracy for multi-class problems.

Table 1: Performance comparison between TKIL and other SOTA methods on CIFAR100 (left-half) and ImageNet-100 (right-half)

Method	**CIFAR-100**, $\mathcal{M} = 2k$					**ImageNet-100**, $\mathcal{M} = 2k$				
Stages	25	10	5	5	10	25	10	5	5	10
New classes per stage	2	5	10	20	10	2	5	10	20	10
iCaRL Rebuffi et al. (2017)	50.6%	53.8%	58.1%	57.2%	52.6%	54.6%	60.8%	65.6%	60.1%	59.6%
BIC Wu et al. (2019)	49.0%	53.2%	56.9%	59.3%	54.2%	59.7%	65.1%	69.0%	70.1%	64.9%
iTAML Rajasegaran et al. (2020)	55.9%	-	-	74.5%	76.6%	64.7%	-	-	69.3%	70.4%
Mnemonics Liu et al. (2020)	61.0%	62.3%	64.1%	63.3%	62.2%	69.7%	71.4%	72.6%	70.6%	70.4%
PODNet Douillard et al. (2020)	62.7%	64.1%	64.5%	58.9%	59.7%	68.3%	74.3%	75.6%	72.5%	71.5%
PODNet+DDE Hu et al. (2021)	-	60.1%	63.4%	59.1%	55.3%	-	70.2%	72.3%	69.2%	65.5%
AFC Kang et al. (2022)	64.1%	64.3%	65.9%	64.9%	64.4%	73.4%	75.8%	76.9%	72.9%	71.7%
TKIL(Ours)	**73.5%**	**80.5%**	**83.6%**	**79.5%**	**82.5%**	**75.5%**	**77.5%**	**79.7%**	**74.7%**	**73.3%**

4.3 RESULTS: IMAGENET-1K

Table 2: Performance comparison between TKIL and other SOTA methods on ImageNet-1k

Method	**ImageNet-1k**, $\mathcal{M} = 10k$	**ImageNet-1k**, $\mathcal{M} = 20k$		**Model Size**
Stages	10	5	10	
iCaRL Rebuffi et al. (2017)	44.8%	51.5%	46.8%	35M
BIC Wu et al. (2019)	48.5%	62.6%	58.7%	35M
Mnemonics Liu et al. (2020)	48.6%	64.5%	63.5%	34.7M
PODNet Douillard et al. (2020)	48.8%	64.1%	62.0%	35.4M
SS-IL Ahn et al. (2021)	64.5%	65.5%	65.2%	35.4M
TKIL(Ours)	**64.9%**	**66.9%**	**65.4%**	12.8M

Table 2 shows the results of all existing CIL algorithms and TKIL on the large-scale dataset with different memory settings. The results again indicate that TKIL is the most accurate method in all settings. The most comparable existing method, SS-IL, has similar accuracy. However, notably, TKIL's model size is much more compact with only 36% of SS-IL's model size. This gap is due to TKIL's generic model being unbiased on all tasks while SS-IL trains a fixed model to fit all tasks. We reaffirm it with more comparisons in Appendix Table 6, which shows that TKIL is more robust than SS-IL and outperforms SS-IL in all incremental learning settings in CIFAR-100 and ImageNet-100.

4.4 ANALYSIS AND ABLATION STUDIES

TKIL VS. Other Approaches. In Figure 4, we investigate the accuracy of the compared approaches (KD, iTAML) when tasks are added incrementally, i.e., 5, 10, or 20 classes are added 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). These results also show that KD on Features only slightly improves the accuracy compared to iTAML (also shown in Appendix, Table 7). The reason for only a slight improvement is that those losses fail to constrain the task model and direct it toward a common optimum for all seen tasks. However, TKIL succeeds to maintain the smallest slope of the drop and no significant collapse events in the accuracy curve. In addition, we compute the accuracy of task prediction for all compared methods and add the prediction component to methods if they did not originally include one, such as iCaRL.

Table 3: Ablation study of the fixed memory $\mathcal{M}$

Stage	**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%	89.5%	88.2%	87.4%	86.7%	85.7%	84.5%	83.7%	83.1%	82.5%
$\mathcal{M} = 4k$	92.2%	90.5%	89.4%	88.8%	87.7%	86.5%	85.4%	84.3%	84.2%	83.7%

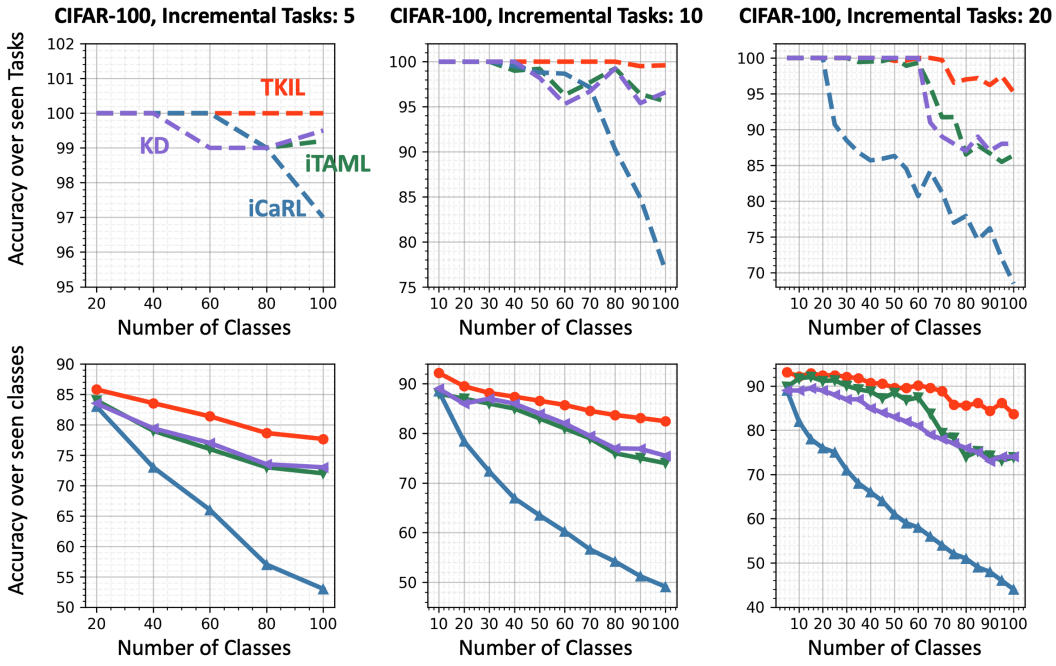

Figure 4: Top: Task accuracy over seen classes on CIFAR-100, with 5, 10, and 20 tasks from left to right; bottom: classification accuracy over seen classes on CIFAR100, with 5, 10, and 20 tasks from left to right. TKIL consistently outperforms other existing methods across different settings.

Table 4: Ablation study of the hyperparameter γ selection

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%

Memory Size. In Table 3, we investigate the impact of memory size on TKIL. We find that TKIL achieves robust SOTA accuracy for different memory sizes. Increasing memory size from $1k$ to $4k$ boosts the final accuracy from 75.3% to 83.7%. 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 ($\mathcal{M} = 2k$).

Hyperparameter Tuning. In Table 4, we demonstrate the importance of GTK loss and the related parameter, γ. 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. Based on these results, we select γ as $\gamma = 0.1$. More parameter searching experiments are included in Appendix Table 8.

5 Conclusion

We propose a tangent kernel optimization approach for class balanced incremental learning, TKIL, that addresses imbalances in memory-based incremental learning. Specifically, we formulated gradient tangent kernels over feature layers to learn balanced representations for the generic model. The generic model is able to execute accurate task predictions and automatically to adapt itself to the corresponding task during inference. Our experiments on multiple benchmarks show that TKIL outperforms existing state-of-the-art methods in various incremental learning settings.

REFERENCES

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*, pp. 844–853, 2021.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 139–154, 2018.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in Neural Information Processing Systems*, 32, 2019.

Eden Belouadah and Adrian Popescu. Il2m: Class incremental learning with dual memory. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 583–592, 2019.

Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 233–248, 2018.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 532–547, 2018.

Kwang-Ting Cheng and Yi-Chu Wang. Using mobile gpu for general-purpose computing–a case study of face recognition on smartphones. In *Proceedings of 2011 International Symposium on VLSI Design, Automation and Test*, pp. 1–4. IEEE, 2011.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *European Conference on Computer Vision*, pp. 86–102. Springer, 2020.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 831–839, 2019.

Xinting Hu, Kaihua Tang, Chunyan Miao, Xian-Sheng Hua, and Hanwang Zhang. Distilling causal effect of data in class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3957–3966, 2021.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Khurram Javed and Faisal Shafait. Revisiting distillation and incremental classifier learning. In *Asian Conference on Computer Vision*, pp. 3–17. Springer, 2018.

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

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yijun Li, Richard Zhang, Jingwan Lu, and Eli Shechtman. Few-shot image generation with elastic weight consolidation. *arXiv preprint arXiv:2012.02780*, 2020.

Zhiyuan Li, Ruosong Wang, Dingli Yu, Simon S Du, Wei Hu, Ruslan Salakhutdinov, and Sanjeev Arora. Enhanced convolutional neural tangent kernels. *arXiv preprint arXiv:1911.00809*, 2019.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 12245–12254, 2020.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A Alemi, Jascha Sohl-Dickstein, and Samuel S Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. *arXiv preprint arXiv:1912.02803*, 2019.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, pp. 2642–2651. PMLR, 2017.

Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11321–11329, 2019.

Jathushan Rajasegaran, Munawar Hayat, Salman H Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

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*, pp. 13588–13597, 2020.

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*, pp. 2001–2010, 2017.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Bram Wallace, Ziyang Wu, and Bharath Hariharan. Can we characterize tasks without labels or features? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1245–1254, 2021.

Max Welling. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1121–1128, 2009.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 374–382, 2019.

Yang Zheng, Jinlin Xiang, Kun Su, and Eli Shlizerman. Bi-maml: Balanced incremental approach for meta learning. *arXiv preprint arXiv:2006.07412*, 2020.

A APPENDIX

A.1 PROOF OF LEMMA 1

Proof of lemma 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 t is the continuous time index. The parameters $\boldsymbol{\theta}$ evolve in the following equation:

$$\frac{d\boldsymbol{\theta}(t)}{dt} = -\nabla \mathcal{L}(\boldsymbol{\theta}(t)) = -(F(\boldsymbol{\theta}, \boldsymbol{x}) - \boldsymbol{y}) \frac{\partial F(\boldsymbol{\theta}(t), \boldsymbol{x})}{\partial \boldsymbol{\theta}}, \tag{10}$$

From Equation 10, the evolution of the network outputs can be written as:

$$\frac{dF(\boldsymbol{\theta}(t), \boldsymbol{x})}{dt} = -(F(\boldsymbol{\theta}, \boldsymbol{x}) - \boldsymbol{y}) \langle \frac{F(\boldsymbol{\theta}(t), \boldsymbol{x})}{\partial \boldsymbol{\theta}} \frac{F(\boldsymbol{\theta}(t), \boldsymbol{x})}{\partial \boldsymbol{\theta}} \rangle, \tag{11}$$

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

$$\frac{d\boldsymbol{u}(t)}{dt} = -\mathcal{K}(t)(\boldsymbol{u}(t) - \boldsymbol{y}), \tag{12}$$

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

A.2 PROOF OF THEOREM 1

Proof of Theorem 1. 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:

$$F(\boldsymbol{\theta}_T, \boldsymbol{x}) = \mathcal{K}_T^T \mathcal{K}_T^{-1} \boldsymbol{y}, \textit{where}, \mathcal{K}_T = \langle \frac{\partial F(\boldsymbol{x}, \boldsymbol{\theta}_T)}{\partial \boldsymbol{\theta}}, \frac{\partial F(\boldsymbol{x}, \boldsymbol{\theta}_T)}{\partial \boldsymbol{\theta}} \rangle, \tag{13}$$

$$F(\boldsymbol{\theta}_{T-1}, \boldsymbol{x}) = \mathcal{K}_{T-1}^T \mathcal{K}_{T-1}^{-1} \boldsymbol{y}, \textit{where}, \mathcal{K}_{T-1} = \langle \frac{\partial F(\boldsymbol{x}, \boldsymbol{\theta}_{T-1})}{\partial \boldsymbol{\theta}}, \frac{\partial F(\boldsymbol{x}, \boldsymbol{\theta}_{T-1})}{\partial \boldsymbol{\theta}} \rangle, \tag{14}$$

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 14, we note that the only variable is the mapping function $\mathcal{K}$ and such mapping function determines the output of the model. Thus, minimizing the cosine distance of two Jacobian matrices $\frac{\partial F(\boldsymbol{x}, \boldsymbol{\theta}_T)}{\partial \boldsymbol{\theta}}$ and $\frac{\partial F(\boldsymbol{x}, \boldsymbol{\theta}_{T-1})}{\partial \boldsymbol{\theta}}$ can minimize the difference between current model and previous model.

A.3 GRADIENTS IN TRAINING: TKIL VS. OTHER METHODS

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

Given a set of feature space parameter θ and task classification parameters $\phi = \{\phi_1, \phi_2 ..., \phi_T\}$, we can obtain the gradients for different methods:

Training without task models:

$$g = \nabla_{\theta, \phi} \mathcal{L}(\theta, \phi, \boldsymbol{x}), \tag{15}$$

KD with task models:

$$\begin{aligned} g_{KD} &= \frac{1}{T} \sum_{i=1}^{T} g_{i,KD} \\ &= \frac{1}{T} \sum_{i=1}^{T} \nabla \Phi_i \mathcal{L}_i^*(\Phi_i) \\ &= \frac{1}{T} \sum_{i=1}^{T} \nabla_{\theta_i, \phi_i} \mathcal{L}_i^*(\theta_i, \phi_i) \end{aligned}$$

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

while the GTK with task models:

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

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

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

A.4 Inference Implementations

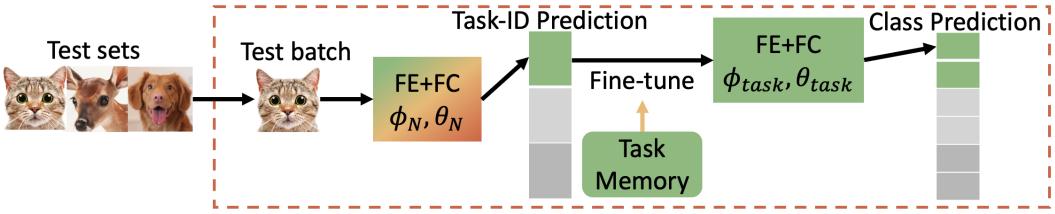

Figure 5: During Inference, the generic model generates task predictions when given the test sets for all samples. Then the predicted corresponding task model with the highest probability is finetuned from the generic model with exemplars from memory to predict the class.

During the inference, we adopt the task prediction procedure described in iTAML Rajasegaran et al. (2020). 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 5.

A.5 Experiments Supplementary to Experiments Reported in the Main Paper

Table 5: Performance comparison between the TKIL and other state-of-the-art methods on MNIST and SVHN

Method	MNIST(5)	SVHN(5)
EWC Kirkpatrick et al. (2017)	19.80%	18.21%
MAS Aljundi et al. (2018)	19.52%	17.32%
RPS-net Rajasegaran et al. (2019)	96.16%	88.91%
iTAML Rajasegaran et al. (2020)	97.15%	92.93%
TKIL	97.81%	97.51%

Upper Bound Baseline. 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 6). Specifically, TKIL is closer to the upper bound and outperforms SS-IL in 5 and 10 tasks with CIFAR-100 and ImageNet-100, respectively.

Components Contribution. To demonstrate how different components contribute to TKIL performance, we investigated ablations of the TKIL architecture in Table 7. We create baselines with

Table 6: Performance comparison between the TKIL and SS-IL on CIFAR100 and ImageNet dataset

Method	**CIFAR-100**		**ImageNet-1k**	
Number of Tasks	5	10	5	10
iCaRL Rebuffi et al. (2017) (*baseline*)	57.2%	52.6%	51.5%	46.8%
SS-IL(Ahn et al. (2021))	74.3%	76.7%	65.5%	65.2%
TKIL *(Ours)*	**79.5%**	**82.5%**	**66.9%**	**65.4%**
upper bound: TKIL	88.9%		69.7%	
upper bound: SS-IL	93.5%		73.3%	

Table 7: Ablation study on TKIL components: KD, GTK Loss on CIFAR-100, 10 incremental stages

Component		Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6	Stage 7	Stage 8	Stage 9	Stage 10
KD	Task ↑	100%	100%	100%	99.0%	97.2%	95.6%	94.0%	81.5%	74.2%	47.6%
	Class ↑	90.7%	88.0%	86.6%	82.8%	79.7%	76.8%	74.6%	62.8%	57.8%	46.6%
KD + Task model	Task ↑	100%	100%	100%	99.5%	100%	95.6%	100%	97.5%	94.2%	96.1%
	Class ↑	90.3%	86.2%	86.5%	84.4%	81.3%	81.2%	78.7%	79.1%	77.2%	76.4%
TKIL	Task ↑	100%	100%	100%	100%	100%	100%	100%	100%	100%	**99.9%**
	Class ↑	92.2%	89.6%	88.2%	87.4%	86.6%	85.7%	84.6%	83.7%	83.2%	**82.5%**

KD-only and KD with task models. We find that task and class predictions decrease as more incremental stages are introduced for the KD-only method. When considering KD with task models, we observe improvement in task prediction from the KD-only variant however there is still a decrease in accuracy for class prediction, eventually decreasing to 76.4%. When GTK Loss is added we indeed observe a boost in accuracy on both task and class predictions with task prediction being 99.9% and class prediction 82.46%.

Table 8: Ablation study of the γ selection

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%
	Task 1, S1	**Task 1, S2**	**Task 1, S3**	**Task 1, S4**	**Task 1,S5**
	Task 1 Class-prediction at different Stages				
$\gamma = 10$	89.6%	89.4%	88.3%	89.4%	87.4%
$\gamma = 1$	75.6%	89.4%	85.3%	78.4%	70.4%
$\gamma = 0.1$	92.2%	89.5%	88.5 %	88.3 %	87.9%