TKIL: Tangent Kernel Optimization for Class Balanced Incremental Learning

Jinlin Xiang
Department of Electrical & Computer Engineering
University of Washington, Seattle, USA
jinlinx@uw.edu

Eli Shlizerman
Department of Electrical & Computer Engineering,
Department of Applied Mathematics,
University of Washington, Seattle, USA
shlizee@uw.edu

Abstract

When learning multiple tasks in a sequence, 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. Memory-based Class Incremental Learning (CIL) methods address this problem by re-learning exemplars retained in the memory from previous tasks. However, due to data imbalances between the training data for the current task and the limited exemplars from previous tasks, existing methods struggle to balance the accuracy across all seen tasks. Here, we propose to address data imbalance and in addition to a generic model to learn a set of task-specific parameters. In particular, we propose a novel methodology of Tangent Kernel for Incremental Learning (TKIL) that seeks an equilibrium between current and previous representations. Specifically, TKIL achieves such equilibrium by tuning different task-specific parameters for different tasks with a new Gradient Tangent Kernel (GTK) loss. Therefore, when representing previous tasks, task-specific models are not impacted 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 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 five CIL benchmark datasets with ten incremental learning settings show that TKIL outperforms existing state-of-the-art methods, e.g.*, achieving 9.4% boost on CIFAR-100 with 25 incremental stages.*

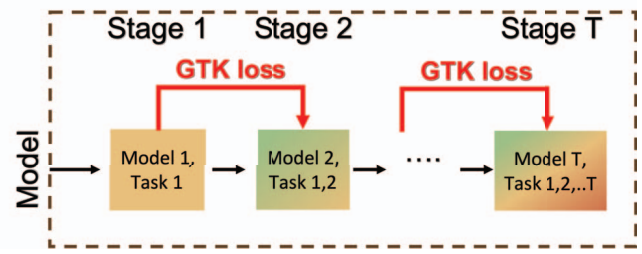

Figure 1. We propose a Tangent Kernel optimization for Incremental Learning (TKIL) with a novel Gradient Tangent Kernel (GTK) loss for optimal class-balanced learning.

1. Introduction

Visual content is evolving and its volume is rapidly increasing. Indeed, high-quality and large-scale visual media streaming data could become infeasible to store and process fully [8, 28, 31, 56]. Therefore, it is appealing to develop learning agents that do not require complete data at the onset of training and continuously learn new data [52]. Class Incremental Learning (CIL) addresses the development of such methods, where agents are expected to learn with incremental arrival of new tasks along with limited exemplars from previous tasks [54, 33].

This learning scenario is significantly different from conventional learning, especially for classification tasks, due to the data imbalance between current and previous tasks [17]. 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 [43]. Current methods propose additional finetuning steps to correct this imbalance, *e.g.*, balanced finetuning or task-level bias rectification. However, training a fixed model for all tasks limits the ability to deal

with imbalanced data [1, 52].

Here, we propose a different approach based on the principle of attaining task-independent generic representations. Instead of training a fixed model for all tasks, we propose to learn a set of generalized parameters and consolidate them into 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 [7].

To obtain generic representations, we propose a novel Tangent Kernel optimization approach for Incremental Learning (TKIL). This approach utilizes a novel Gradient Tangent Kernel (GTK) loss, which is inspired by tangent kernel theory. Tangent kernels describe the weights evolution of a neural network during training [26, 47]. To transfer learned generic representations, we use the previous tangent kernel to regulate the tangent kernel of the current model. Therefore, we define GTK loss as a cosine similarity loss that minimizes the discrepancies between the gradients (with respect to parameters) from previous and current tangent kernels. GTK is different from previously proposed tangent kernels, since NTK is designed for infinite-width neural networks only, while GTK is applicable to finite-width networks that particularly appear in CIL [15, 43]. During updates of the generic model, TKIL tunes different task-specific models representing current and previous tasks with GTK loss, and then takes the average of task-specific gradient updates as the update of the generic model. Since TKIL updates the task parameters separately, it overcomes the bias of the current task by design. In such updates, the GTK loss is expected to transfer from the previous model to task-specific models and prevent their divergence from each other. 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 classification.

In summary, our main contributions in this work 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) loss for finite-width neural networks. We show that minimizing GTK loss associated with TKIL achieves more balanced representations. Such representations allow TKIL to generate robust task predictions and to update 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.

2. Related Work

Class Incremental Learning (CIL). CIL addresses incremental learning in its full generality, in contrast to more specific continual learning, such as Task Continual Learning [19, 35, 34, 18]. In particular, CIL doesn't rely on the task category (domain) being specified. CIL typically includes three components: ***Exemplars Selection With Rehearsal*** [43, 5, 14, 30, 38, 37, 29], ***Forgetting Constraints*** [21, 24, 6, 2], and ***Bias Corrections*** [51, 4, 14, 55, 5]. Recent works additionally introduced **Memory-free Approaches** and **Online Continual Learning Settings** for class incremental learning [44, 22, 32, 9, 53, 39, 32]. These settings are more realistic when considering limited computational costs, *e.g.*, small memory with few updates. To follow online settings, we leverage a fixed-size memory and train all samples only once in an epoch. In summary, such works contributed to the enhancement and combination of these components, while in our work we mainly focus on memory-based methods combined with forgetting constraints. 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 [49] to select the most representative exemplars [43, 5, 14]. Additional methods estimated the distribution of previously learned tasks and generated extra pseudo-exemplars or images to avoid the imbalance between classes [30, 38, 37]. Recent works showed that dynamic expansions boost rehearsal methods, *e.g.*, tuning different task representations [20, 46, 7, 50, 42, 41]. While these methods can preserve unbiased representations by task parameters, the training cost grows linearly with the number of tasks. In contrast, our approach does not rely on a particular selection of exemplars or extra pseudo-exemplars to boost accuracy, and is designed to be computationally efficient with a constant training cost.

• ***Forgetting Constraints.*** Various regularization terms have been added to the classification loss to constrain forgetting of previous representations [21, 24, 6, 2]. In addition, Knowledge Distillation (KD) has been proposed to restore previous knowledge representations [43, 5, 25, 1]. Furthermore, Adaptive Feature Consolidation (AFC) [17] modified KD further 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.

Tangent Kernels. Neural Tangent Kernels (NTK) were first introduced in [15] and developed in multiple subse-

quent works [3, 26, 36, 48]. 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. 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 [10]. Recently, Task Tangent Kernel (TTK) was introduced for finite-width neural networks [48], based on a kernelized distance across the gradients of multiple random initialized networks (at least 50 networks in their setting) to estimate the similarity over different tasks. In contrast, GTK loss calculates the tangent kernel of a trained network only a single time during new model training.

3. Methods

Given an imbalanced training distribution at each stage, we propose a Tangent Kernel Incremental Learning approach that aims to train a unified classifier sequentially. Figure 2 illustrates an overview of our approach.

3.1. Notations

We denote the sequential data as a batch of datasets $\mathcal{D} = \{D_1, D_2, \dots, D_T, \dots\}$. D_T is T-th dataset in $\mathcal{D}$, which contains training images $\boldsymbol{x} = \{x_i\}_{i=1}^n$ and labels $\boldsymbol{y} = \{y_i\}_{i=1}^n$. Each D_T dataset includes m classes, with total $m \times T$ classes. At T-th incremental stage, the training data in a fixed memory buffer is the full data for current classes D_T, and a small set of previous exemplars $\mathcal{M}_T = \{M_1 \subseteq D_1, M_2 \subseteq D_2, \dots, M_{T-1} \subseteq D_{T-1}\}$. We denote the T-th generic model in incremental learning as $F_T = F(\boldsymbol{\theta}_T, \boldsymbol{x})$, where **bold** $\boldsymbol{\theta}_T = \{\phi_T, \theta_T\}$ is all parameters in the network and $\{\phi_T, \theta_T\}$ denote the parameters in feature extractor layers and fully connected layers, respectively. We use $\{\phi_{T,\ task\ i}, \theta_{T,\ task\ i}\}$ as the parameters for the task-specific model i in T-th incremental stage. In Section 3.2, we denote $F(\boldsymbol{\theta})$ to represent a finite-width neural network when analyzing tangent kernels.

3.2. Tangent Kernels

In this section, we introduce tangent kernels and provide a comprehensive analysis. Consider a 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 $\frac{d\boldsymbol{\theta}(\tau)}{d\tau} = -\nabla\mathcal{L}(\boldsymbol{\theta}(\tau))$. Let $\boldsymbol{u}(\tau) = F(\boldsymbol{\theta}(\tau), \boldsymbol{x})$ be the network outputs at time τ and $\boldsymbol{y}$ is the ground truth. $\boldsymbol{u}(\tau)$ follows the following evolution [15, 3]:

$$\frac{d\boldsymbol{u}(\tau)}{d\tau} = -\mathcal{K}_\tau(\boldsymbol{x}, \boldsymbol{x})(\boldsymbol{u}(\tau) - \boldsymbol{y}), \tag{1}$$

where the Tangent Kernel is

$$\mathcal{K}_\tau(\boldsymbol{x}, \boldsymbol{x}) = \left\langle \frac{\partial F(\boldsymbol{\theta}(\tau), \boldsymbol{x})}{\partial \boldsymbol{\theta}}, \frac{\partial F(\boldsymbol{\theta}(\tau), \boldsymbol{x})}{\partial \boldsymbol{\theta}} \right\rangle. \tag{2}$$

The dynamics in Eq. 1 are identical to the dynamics of kernel regression under the gradient flow (Assuming $\boldsymbol{u}(0) = 0$). Thus, the output of a trained neural network at time τ for any testing input $\boldsymbol{x}'$ is

$$F(\boldsymbol{\theta}(\tau), \boldsymbol{x}') = \mathcal{K}_\tau(\boldsymbol{x}', \boldsymbol{x})^T \mathcal{K}_\tau(\boldsymbol{x}, \boldsymbol{x})^{-1} \boldsymbol{y}. \tag{3}$$

From Eq. 3, we obtain training predictions when we set the inputs to be the training data points, i.e., $\boldsymbol{x}' = \boldsymbol{x}$,

$$\boldsymbol{u}(\tau) = F(\boldsymbol{\theta}(\tau), \boldsymbol{x}) = \mathcal{K}_\tau(\boldsymbol{x}, \boldsymbol{x})^T \mathcal{K}_\tau(\boldsymbol{x}, \boldsymbol{x})^{-1} \boldsymbol{y}. \tag{4}$$

From Eqs. 1 and 4, we observe that the only variable in training predictions is the kernel function $\mathcal{K}_\tau(\boldsymbol{x}, \boldsymbol{x})$ determined by the Jacobian matrix $\frac{\partial F(\boldsymbol{\theta}(\tau), x)}{\partial \boldsymbol{\theta}}$. Therefore, the Jacobian matrix solely determines the training outputs $\boldsymbol{u}(\tau)$. This motivates us to employ the Jacobian matrix from the previous model to assist with training a new model. Since the previous model does not need to be retrained, its tangent kernel and the Jacobian matrix are both **deterministic**.

While some components of the Tangent Kernel definition are extended from Neural Tangent Kernel (NTK) definition and theory, these two types of kernels are different in their purpose and regime. Tangent Kernels are defined for finite-width neural networks, while NTK is defined and applies to the scenario of infinite-width neural networks [3]. The key difference is that Tangent Kernels evolve during back-propagation in finite-width neural networks, whereas NTK kernels are deterministic after initialization [47].

3.3. Tangent Kernel Loss for Incremental Learning

In this section, we transfer previous representations by regulating the tangent kernel in the new model with a novel Gradient Tangent Kernel (GTK) loss.

In Incremental Learning, training the first task ($T = 1$) is similar to conventional learning. We feed the data from Task 1 to train a new model, without the need for tangent kernel loss.

In T-th incremental learning stage ($T \geq 2$), the current model F_T is supposed to address a new task while preserving previously learned knowledge representations. We thus proceed and customize Eq. 4. We show that training the T-th model with the assistance of the previous $(T-1)$-th model gives rise to the minimization of cosine distance in a Gradient Tangent Kernel (GTK) loss. We explain this more rigorously below.

Consider the training of T-th model F_T with a previously fully trained model F_{T-1}. We aim to minimize the **cosine distance** between training outputs of F_{T-1} and F_T with the same training data points $\boldsymbol{x}$ to transfer the learned representations of F_{T-1} to F_T in Eq. 5:

$$\min_{\boldsymbol{\theta}_T}\left(1 - \frac{\langle F_{T-1}(\boldsymbol{\theta}_{T-1}, \boldsymbol{x}), F_T(\boldsymbol{\theta}_T, \boldsymbol{x})\rangle}{\|F_{T-1}(\boldsymbol{\theta}_{T-1}, \boldsymbol{x})\| \|F_T(\boldsymbol{\theta}_T, \boldsymbol{x})\|)}\right), \tag{5}$$

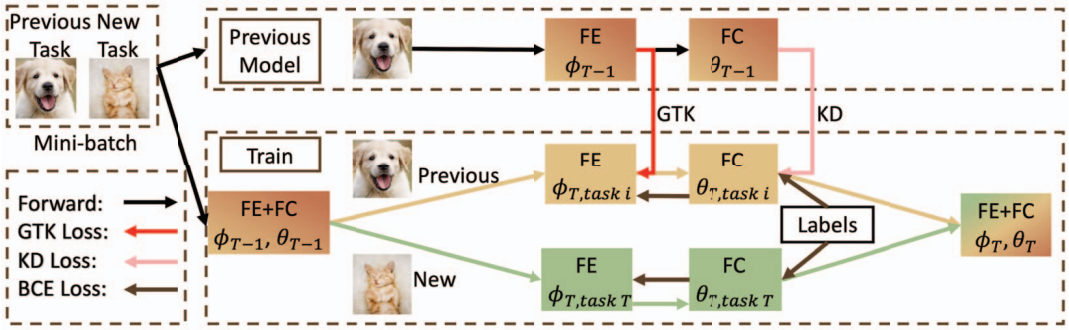

Figure 2. Illustration of Tangent Kernel Incremental Learning approach. $(\phi_{T-1}, \theta_{T-1})$ represent the parameters from the previous generic model and (ϕ_T, θ_T) represent the parameters of the current generic model. In the mini-batch, we separate training steps into different tasks. 1) For the previous tasks (task $i, i = 1, 2, 3, ...T-1$), we optimize T-th stage task-specific models $F_{T,\ task\ i}$ with Binary Cross-Entropy (BCE) loss (brown), Knowledge Distillation loss (pink), and the Gradient Tangent Kernel loss (red) with exemplars from a fixed memory; 2) For the current task (task T), only employs BCE loss to train $F_{T,\ task\ T}$ (brown). (FE: Feature extractor layers, FC: Fully Connected layers)

denoting the cosine distance minimization, Eq. 5, as

$$\min_{\boldsymbol{\theta}_T}(F_{T-1}(\boldsymbol{\theta}_{T-1}, \boldsymbol{x}), F_T(\boldsymbol{\theta}_T, \boldsymbol{x})). \tag{6}$$

Then we plug Eq. 4 into Eq. 5, and obtain Eq. 7:

$$\min_{\boldsymbol{\theta}_T}(\mathcal{K}_T(\boldsymbol{x}, \boldsymbol{x})^T \mathcal{K}_T(\boldsymbol{x}, \boldsymbol{x})^{-1} \boldsymbol{y}, \\ \mathcal{K}_{T-1}(\boldsymbol{x}, \boldsymbol{x})^T \mathcal{K}_{T-1}(\boldsymbol{x}, \boldsymbol{x})^{-1} \boldsymbol{y}), \tag{7}$$

where the tangent kernel $\mathcal{K}_{T-1}$ and the Jacobian matrix $\frac{\partial F(\boldsymbol{x}, \boldsymbol{\theta}_{T-1})}{\partial \boldsymbol{\theta}}$ from the previous model F_{T-1} are constant. The only variable is the tangent kernel function $\mathcal{K}_T(\boldsymbol{x}, \boldsymbol{x})$ from the current model. To further simply Eq. 7, We note that the Jacobian matrix $\frac{\partial F(\boldsymbol{x}, \boldsymbol{\theta}_T)}{\partial \boldsymbol{\theta}}$ solely determines the tangent kernel $\mathcal{K}_T(\boldsymbol{x}, \boldsymbol{x})$ and obtain Eq. 9, i.e., minimization of the cosine distance between two Jacobian matrices $\frac{\partial F(\boldsymbol{x}, \boldsymbol{\theta}_{T-1})}{\partial \boldsymbol{\theta}}$ and $\frac{\partial F(\boldsymbol{x}, \boldsymbol{\theta}_T)}{\partial \boldsymbol{\theta}}$.

$$\min_{\boldsymbol{\theta}_T}(\mathcal{K}_{T-1}(\boldsymbol{x}, \boldsymbol{x}), \mathcal{K}_T(\boldsymbol{x}, \boldsymbol{x})) \tag{8}$$

$$\Rightarrow \min_{\boldsymbol{\theta}_T}(\frac{\partial F_{T-1}(\boldsymbol{x}, \boldsymbol{\theta}_{T-1})}{\partial \boldsymbol{\theta}}, \frac{\partial F_T(\boldsymbol{x}, \boldsymbol{\theta}_T)}{\partial \boldsymbol{\theta}}), \tag{9}$$

Since generic representations are already learned in F_{T-1}, there is a need to avoid drastic changes in feature extractor layers $F_T(\phi_T, \boldsymbol{x})$, where ϕ_T is the parameters in feature layers only. To achieve that, we leverage Eq. 9 and propose a novel Gradient Tangent Kernel loss (**GTK loss**) on feature layers to transfer representations. We select the cosine similarity loss since recent NTK work shows that the cosine loss is an effective way to extract feature representations [48]. Then, we define the kernel objective function as follows

$$\min_{\phi} \mathbb{E}_{(\boldsymbol{x}, \boldsymbol{y})}[\mathcal{L}_{GTK}(1 - \frac{\langle G_T, G_{T-1} \rangle}{\|G_T\| \|G_{T-1}\|})], \tag{10}$$

where $G_{T-1} = \frac{\partial F_{T-1}(\boldsymbol{x}, \phi_{T-1})}{\partial \phi}$ and $G_T = \frac{\partial F_T(\boldsymbol{x}, \phi_T)}{\partial \phi}$, which are Gradients (G) of the last feature layer from the previous and current model, respectively.

3.4. 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 and fully connected layers. Training for 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 classification loss.

The tangent kernel optimization approach (see Algorithm 1) is used to train subsequent T-th tasks $(T \in [2, N])$. In particular, we collect current data D_T and memory buffer M_T as the training data and divide them into multiple large batches. Our approach combines **Task-Specific Model Training** and **Generic Model Update** as a single step, and then, repeats the step for all batches. *Task-Specific Model Training* aims to preserve learned representations from the previous generic model. We separate different task images from one large batch into T mini-batches and feed them to T task-specific models only once. *Generic Model Update* aims to find an unbiased updating direction for all T tasks. To accomplish this, T task-specific models are collapsed into a single generic model that can accommodate T tasks. We describe the training procedures in detail below.

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 the current generic model $(\boldsymbol{\theta}_T)$. One large batch B_T from $\mathcal{M}_T \cup D_T$ is being separated into different tasks as T *mini-batches*. For each task i, an i-th task-specific model $F_{T,task\ i}$ is created and is updated only with i-th mini-batch that corresponds 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 generic 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 8-12):

$$\mathcal{L}_{Class} = \mathcal{L}(F_{T,\,task\,i}(\boldsymbol{x}), \boldsymbol{y}) \tag{11}$$

$$\mathcal{L}_{KD} = \mathcal{L}(F_{T-1}(\boldsymbol{x}), F_{T,\,task\,i}(\boldsymbol{x})) \tag{12}$$

$$\mathcal{L}_{GTK} = \mathcal{L}(G_{T,\,task\,i}, G_{T-1}) \tag{13}$$

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}[\alpha\mathcal{L}_{Class} + \beta\mathcal{L}_{KD} + \gamma\mathcal{L}_{GTK}] \tag{14}$$

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 at the first time that this task is being learned. The loss is expressed as (also shown in Algorithm 1, lines 14-15)

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}[\mathcal{L}(F_{T,\,task\,T}(\boldsymbol{x}), \boldsymbol{y})]. \tag{15}$$

When sequentially introducing more tasks, imbalances in training data points result in a larger mini-batch size for the current task model $F_{T,task\,T}$ (*e.g.*, 512 samples) compared to other tasks (*e.g.*, 128 samples). Since TKIL updates the task parameters separately, it inherently overcomes the bias of task imbalances.

Generic Model Update: At the end of batch training, the trained task-specific models are collapsed to a single generic model F_T, which represents the average direction of tasks [39, 46]. Since dynamic expansion methods show that each task-specific model represents one gradient updating direction, we obtain the generic model by computing the average of these updates [42] (also shown in Algorithm 1, line 19)

$$F_T = \frac{1}{T}\sum_{i}^{T} F_{T,task\,i}. \tag{16}$$

Due to imbalanced distribution of data between the current data and the memory buffer, training a fixed model becomes more biased towards the current task, *e.g.*, BIC and SS-IL train mixed samples from the memory buffer and the current data together [1, 51]. Dynamic expansion approaches train separate task parameters to mitigate overfits to the current task (*e.g.*, iTAML, KD), but they fail to maintain learned representations in the previous generic model [42, 17]. TKIL trains task models and leverages GTK loss to avoid dramatic changes in feature representations. Therefore, as the training progresses, the generic model is enhanced by learning new tasks while simultaneously preserving previous representations. The complementary calculation of the gradients for different methods appears in Supplementary.

Algorithm 1 TKIL Algorithm in one batch

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

1: Initialize the current Model (For the first mini-batch):
2: $\quad F_{T-1}(\phi_{T-1}, \theta_{T-1}) \longrightarrow F_T(\phi_T, \theta_T)$
3: ***Task-Specific Model Training:***
4: Separate B_T into T mini-batch
5: **for** $i = 1, 2, 3, ..T$ **do**
6: $\quad F_T \longrightarrow F_{T,task\,i}$
7: $\quad$ **if** $i < T$ **then**
8: $\quad\quad \mathcal{L}_{\text{Class}} = \sum_{j}^{task\,i} \mathcal{L}_{\text{BCE}}(F_{T,task\,i}(\boldsymbol{x}_j), \boldsymbol{y}_j)$
9: $\quad\quad \mathcal{L}_{\text{KD}} = \sum_{j}^{task\,i} \mathcal{L}_{\text{BCE}}(F_{T,task\,i}(\boldsymbol{x}_j), F_{T-1}(\boldsymbol{x}_j))$
10: $\quad\quad \mathcal{L}_{\text{GTK}} = \sum_{j}^{task\,i} \mathcal{L}_{\text{BCE}}(G_{T,task\,i}(\boldsymbol{x}_j), G_{T-1}(\boldsymbol{x}_j))$
11: $\quad\quad \mathcal{L}_{\text{all}} = \alpha\mathcal{L}_{\text{Class}} + \beta\mathcal{L}_{\text{KD}} + \gamma\mathcal{L}_{\text{GTK}}$
12: $\quad\quad$ Backpropagation for $F_{T,task\,i}$
13: $\quad$ **else if** $i = T$ **then**
14: $\quad\quad \mathcal{L}_{\text{Class}} = \sum_{j}^{task\,T} \mathcal{L}_{\text{BCE}}(F_{T,task\,T}(\boldsymbol{x}_j), \boldsymbol{y}_j)$
15: $\quad\quad$ Backpropagation for $F_{T,task\,T}$
16: $\quad$ **end if**
17: **end for**
18: ***Generic Model Update:***
19: $F_T \longleftarrow \frac{1}{T}\sum_{i}^{T} F_{T,task\,i}$

4. Experiments and Results

4.1. Datasets, Implementation Details, and Evaluations

Datasets. We conduct experiments on five benchmarks, *e.g.*, CIFAR-100 [23], ImageNet-100 and ImageNet-1K [45] with different incremental learning scenarios. CIFAR-100 dataset contains $50,000$ training images and $10,000$ testing images. ImageNet-1k dataset consists of $1,281,167$ training images and $50,000$ images for validation across $1,000$ classes. ImageNet-100 dataset includes 100 randomly sampled classes from ImageNet-1k.

Incremental Learning Settings. We select benchmarks used by existing Class IL works [43, 42, 17, 11]. We employ a ResNet-34 as the generic model for ImageNet-1k and a ResNet-18 for other benchmarks. For all datasets, we apply several optimizers (SGD, Adam, and RAdam [27], experiments included in Supplementary) and experimentally select RAdam with an initial learning rate of 0.01 for 70 epochs. The learning rate is divided by 10 after every 20 epochs. The memory buffer is fixed to 2k for MNIST, SVHN, CIFAR-100, and ImageNet-100, and 20K for ImageNet-1K. To follow the online continual learning

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

Methods	**CIFAR-100**, Memory size $\mathcal{M} = 2k$					**ImageNet-100**, Memory size $\mathcal{M} = 2k$				
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
Joint Training (Upper Bound)		86.3%		84.6%			81.3%		76.8%	
iCaRL [43]	50.6%	53.8%	58.1%	57.2%	52.6%	54.6%	60.8%	65.6%	60.1%	59.6%
iTAML [42]	55.9%	74.9%	75.4%	74.5%	74.6%	64.7%	69.5%	71.9%	69.3%	70.4%
RMM [29]	59.5%	60.9%	69.5%	62.7%	60.6%	68.8%	71.4%	73.8%	70.5%	69.4%
SS-IL [1]	58.0%	71.5%	75.1%	74.8%	71.1%	69.5%	71.7%	73.5%	68.8%	67.6%
Mnemonics [30]	61.0%	62.3%	64.1%	63.3%	62.2%	69.7%	71.4%	72.6%	70.6%	70.4%
PODNet [11]	62.7%	64.1%	64.5%	58.9%	59.7%	68.3%	74.3%	75.6%	72.5%	71.5%
AFC [17]	64.1%	64.3%	65.9%	64.9%	64.4%	73.4%	75.8%	75.9%	72.9%	71.7%
TKIL (Ours)	**73.5%**	**80.5%**	**83.6%**	**80.6%**	**82.5%**	**77.3%**	**78.5%**	**79.7%**	**75.7%**	**75.3%**

settings, we feed data from the memory buffer and the current task only once to train the model in each epoch [32, 40]. The mini-batch size is set to 128 for CIFAR-100 and 64 for ImageNet. Our inference pipeline records task predictions first and then finetunes the generic model to the corresponding task model to perform the classification, see Supplementary and [42].

TKIL Implementation Details. To avoid a linear increase of the training cost at T-th stage, we train the first task model, add its weights to a temporary generic model, and then delete this model to free up the memory. After repeatedly training and adding weights of all T task models to the new generic model, the total number of weights is divided by T to obtain the final generic model (based on Eq. 16). In Inference, TKIL fine-tunes each of the task models once, then TKIL infers the task using the generic model and the class using the corresponding task model. Therefore, the total computational expense includes fine-tuning, inferring tasks, and inferring classes and is comparable to other methods which include fine-tuning only [35, 42]. The efficiency of fine-tuning depends on the memory size (# samples in memory) and the number of updates.

Evaluation. Since TKIL is designed for a memory-based incremental learning method that leverages GTK loss as a forgetting constraint, we used recent memory-based approaches as **baseline benchmarks** in our main results [42, 30, 1, 11, 17]. Additionally, we conducted ablation studies to evaluate TKIL's ability to address class imbalances, including comparisons with methods specifically designed to tackle class imbalances [42, 46, 7]. The **upper bound** accuracy in incremental learning is to train without increments a generic model with a standard loss (MSE) on the full dataset, denoted as **joint training**. We implemented existing methods based on the publicly available official code and trained all models on 2080Ti GPUs with a parallel computation mode [42, 16]. We report the accuracy at the last incremental stage on all learned classes.

4.2. Results: CIFAR-100, ImageNet-100, MNIST, and SVHN

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 more optimal accuracy than other compared methods in all scenarios. The margin in the accuracy of TKIL vs. existing approaches is particularly evident in large tasks (stages) scenarios, and 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 77.6% due to unstable prediction of tasks, where AFC becomes leading method among existing methods with 64.1%. In Table 2, we conduct similar experiments on MNIST and SVHN. TKIL achieves 97% or higher accuracy on these benchmarks when learning 2 classes each time. We also observe that the accuracy of TKIL is approaching the upper bound in both two datasets (76% for ImageNet-100 and 85% for CIFAR-100). As a result, TKIL translates to more optimal accuracy for multi-class problems.

Table 2. Performance comparison between the TKIL and other state-of-the-art methods on MNIST and SVHN (5 stages, 2 new classes per stage, Memory size $= 2k$)

Methods	**MNIST**	**SVHN**
EWC [21]	19.80%	18.21%
RPS-net [41]	96.16%	88.91%
iTAML [42]	97.15%	92.93%
TKIL (Ours)	**97.91% (+0.76%)**	**97.51% (+5.58%)**

Table 3. Performance comparison between TKIL and other SOTA methods on ImageNet-1k. Memory size: $\mathcal{M}$, Upper Bound: Joint Training

Methods	ImageNet-1k $\mathcal{M} = 10k$	ImageNet-1k $\mathcal{M} = 20k$	ImageNet-1k $\mathcal{M} = 20k$
Stages	10	5	10
New classes per Stages	100	200	100
Joint Training (ResNet-18)		67.9%	
Joint Training (ResNet-34)		71.2%	
iCaRL (ResNet-34) [43]	44.8%	51.5%	46.8%
BIC (ResNet-34) [51]	48.5%	62.6%	58.7%
Mnemonics (ResNet-34) [30]	48.6%	64.5%	63.5%
PODNet (ResNet-34) [11]	48.8%	64.1%	62.0%
SS-IL (ResNet-18) [1]	57.3%	59.6%	59.4%
SS-IL (ResNet-34) [1]	64.5%	65.5%	65.2%
TKIL (ResNet-18)	**64.9%**	**66.9%**	**65.7%**
TKIL (ResNet-34)	**65.6%**	**68.9%**	**67.9%**

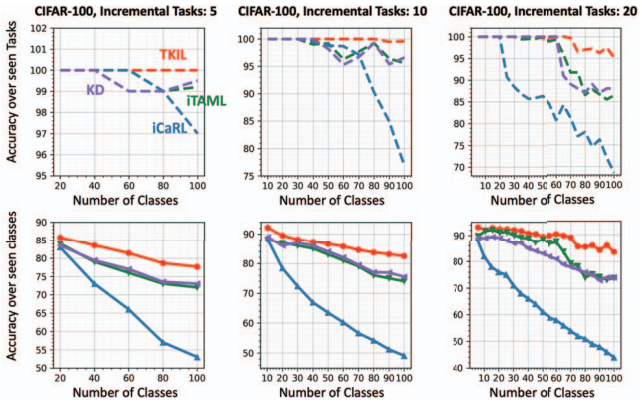

Figure 3. Top: Task accuracy over seen classes on CIFAR-100, with 5, 10, and 20 stages from left to right; bottom: classification accuracy over seen classes on CIFAR-100, with 5, 10, and 20 stages from left to right. TKIL consistently outperforms other existing methods across different settings.

4.3. Results: ImageNet-1k

Table 3 shows the results of CIL algorithms and TKIL on a large-scale dataset with different memory settings. Our results indicate that TKIL is the most accurate method in all settings. The second accurate baseline, SS-IL, is lower by about 5% when applied to a compact model, ResNet-18. This gap indicates that the forgetting constraints strategies fail to maintain the generalization in feature layers. Since SS-IL trains a fixed model and only tunes classification layers for all tasks, the generic parameters in feature extractor layers are not learned. We reaffirm it with more comparisons in the ablation study, 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

Training Cost Analysis. **TKIL training complexity** is similar to other memory-based IL methods. In each epoch, images from the memory buffer and from the current data are used once. Specifically, we collect a large batch and feed different task images as mini-batches (*e.g.*, 128 samples in one mini-batch) to different task models. Due to computational limitations, when working with extremely large mini-batches, we split them. Non-task training methods (*e.g.*, SS-IL, BIC) mix the current data and memory buffer and feed them to a single model with multiple small batches (*e.g.*, 128 samples). Therefore, the total training time in TKIL and other baselines is **total number of images/ batch size $*$ processing time of one batch**. **TKIL Memory usage** is constant in each T-th ($T > 1$) stage and TKIL does not accumulate memory storage. In training, we store 3 models, the previous generic model, the current task model, and the new generic model. For example, when ResNet-18 is used as the generic model, TKIL maximum memory usage is $12.8M \times 3 \approx 38M$ compared to $12.8M \times 2 \approx 26M$ for other baselines.

TKIL VS. Dynamic Expansion Approaches. In Figure 3, we compare both tasks and classes accuracy of TKIL with baseline (iCaRL) and dynamic expansion approaches (KD, iTAML) when tasks are added incrementally, i.e., 5, 10, or 20 classes are added each time on the CIFAR-100 benchmark. iTAML tunes task-specific models with a classification loss only, while KD adds an extra knowledge distillation loss to transfer representations on feature extractor layers. In these experiments, TKIL consistently outperforms the compared methods by a large margin (*e.g.*, 82% vs. 74% in 20 stages scenario). While KD slightly improves the accuracy, the divergence among task models still exists, making it difficult to consolidate into an optimum generic model. TKIL, however, maintains learned representations and does not show significant events of collapse in the accuracy curve.

TKIL VS. Visual Domain Decathlon (VDD). In addition to demonstrating the efficiency of TKIL on multiple-domain dataset learning, we conducted experiments to learn three domains (CIFAR, ImageNet, and MNIST) sequentially from scratch in Table 4. VDD challenge focuses on multiple-domain dataset learning but does not address sequential incoming tasks without task (domain) categories as TKIL does [12, 13]. For example, for a scenario with two tasks (CIFAR and ImageNet), during inference, if the task category is not provided, VDD methods do not feed correct samples to the CIFAR or ImageNet task models. In contrast, TKIL automatically predicts task categories and feeds samples to the corresponding task models.

Sample Imbalance Experiments. In Table 5, we compare TKIL and SS-IL with several methods that address sample imbalance in continual learning [46, 9, 7]. Table 5

Table 4. TKIL on Visual Domain Decathlon benchmarks

	MNIST	CIFAR	ImageNet
VDD [12, 13]	96.4%±1.3%	86.3%±1.2%	84.3%±1.5%
TKIL	98.4%±0.4%	92.4%±0.5%	87.4%±0.5%

Table 5. Ablation study of Sample Imbalanced Experiments (MNIST: 5 stages, CIFAR-100:10 stages, ImageNet-1k: 10 stages)

Methods	MNIST		CIFAR-100		ImageNet-1k	
	Mean ↑	σ ↓	Mean ↑	σ ↓	Mean ↑	σ ↓
Layerwise [46]	84.3%	0.92	71.3%	2.45	39.5%	1.96
OCL [9]	85.4%	0.73	60.4%	1.51	41.4%	1.82
Subspaces [7]	86.6%	0.97	64.3%	3.14	53.6%	3.37
SS-IL [1]	94.6%	1.23	75.1%	3.45	59.4 %	4.36
TKIL (Ours)	**97.9%**	**0.74**	**82.5%**	**1.34**	**65.7%**	**1.89**

Table 6. Ablation study on large task setting. CIFAR dataset, 50 stages, 1 new class per stage, memory size $= 2k$.

iTAML [42]	SS-IL [1]	PODNet [11]	AFC [17]	**TKIL**
43.5%	53.4 %	57.8%	60.8%	**67.9%**

shows that most methods (*e.g.*, OCL) maintain a low level of variation for imbalanced data. However, the accuracy of these methods is limited when applied to ImageNet. SS-IL achieves similar accuracy as TKIL, but the deviation across different tasks is larger – almost double than TKIL. This indicates that SS-IL struggles to address imbalanced data. In summary, TKIL outperforms other methods by showing improvements in accuracy and a smaller standard deviation (σ) across all tasks.

Scalability to Large T**.** In Table 6, we experiment with a more challenging scenario of a large task incremental learning setting. When $T = 50$, the closest comparable dynamic expansion method, SS-IL, experiences a significant decrease in accuracy due to attaining non-optimal representations. In this case as in other cases, TKIL outperforms other comparable baselines and approaches the upper optimal bound.

Qualitative Results. In Figure 4, we illustrate t-SNE visualization of features representation with MNIST dataset. The feature representations are taken from the final layer of the feature extractor and are projected into a 2D space. We find that the features without GTK (top) are not well clustered when adding more stages, while representations obtained with GTK loss (bottom) are more efficiently clustered, as indicated by the lower Davies-Bouldin Index. The possible reason for the successful clustering in TKIL could be that GTK Loss reduces divergence between task models. Despite the utilization of dynamic expansions in the non-GTK approach, a high level of variation among task-specific models results in an overfitted generic model, especially when scaling up to more stages. In this scenario, the non-GTK approach separates only the newly added classes (i.e., the brown class), but fails to classify the previously learned tasks (i.e., the navy blue class). Therefore, we observe that GTK loss has a more robust effect on preserving

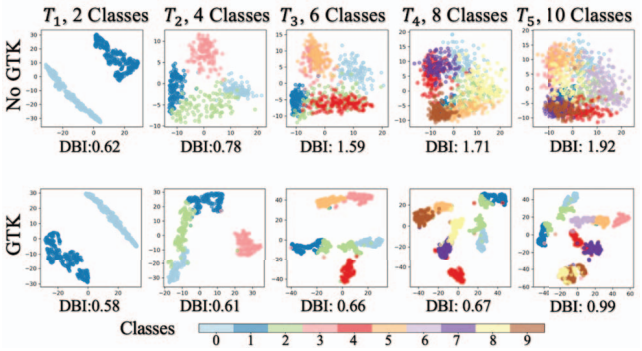

Figure 4. T-SNE visualization (colors indicate classes) of feature representations from the last feature extractor on MNIST with 5 Tasks and 2 new classes introduced in each stage. **Representations by GTK (bottom) are clustered more efficiently (supported by Davies-Bouldin Index (DBI - lower better))** than representations that do not use GTK (top).

learned representations compared to conventional methods such as KD or MSE loss.

Ablation Study for Inference. Table 7 demonstrates the accuracy as memory size increases and shows that improves with memory size expansion and also that more updates with at least 20 memory samples are needed. This leads to estimate that TKIL is able to fine-tune each of the task models once with 20 samples from memory during inference.

Table 7. Ablation Study for Inference (CIFAR-100, 10 tasks)

# Memory Samples	5	**20**	**50**	**100**
1 Update	66.4%±2.4%	82.5%±0.3%	83.3%±0.3%	84.1%±0.2%
2 Updates	67.1%±2.1%	83.4%±0.3%	83.9%±0.3%	84.5%±0.2%
5 Updates	68.2%±1.8%	84.1%±0.3%	84.5%±0.3%	85.3%±0.2%

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 loss over feature layers to learn balanced representations for the generic model. The generic model is able to execute accurate task predictions and automatically adapt 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.

6. Acknowledgment

We acknowledge the partial support of National Science Foundation grant EFMA-2223495 (EFRI BRAID), National Science Foundation grant OAC-2117997 (A3D3), and the departments of Applied Mathematics and Electrical and Computer Engineering at the University of Washington.

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] 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)*, pages 139–154, 2018.

[3] 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.

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

[5] 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)*, pages 233–248, 2018.

[6] 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)*, pages 532–547, 2018.

[7] Arslan Chaudhry, Naeemullah Khan, Puneet Dokania, and Philip Torr. Continual learning in low-rank orthogonal subspaces. *Advances in Neural Information Processing Systems*, 33:9900–9911, 2020.

[8] 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*, pages 1–4. IEEE, 2011.

[9] Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. In *International Conference on Machine Learning*, pages 1952–1961. PMLR, 2020.

[10] Thang Doan, Mehdi Abbana Bennani, Bogdan Mazoure, Guillaume Rabusseau, and Pierre Alquier. A theoretical analysis of catastrophic forgetting through the ntk overlap matrix. In *International Conference on Artificial Intelligence and Statistics*, pages 1072–1080. PMLR, 2021.

[11] 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*, pages 86–102. Springer, 2020.

[12] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[13] Yunhui Guo, Yandong Li, Liqiang Wang, and Tajana Rosing. Depthwise convolution is all you need for learning multiple visual domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8368–8375, 2019.

[14] 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*, pages 831–839, 2019.

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

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

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

[18] Ryo Karakida and Shotaro Akaho. Learning curves for continual learning in neural networks: Self-knowledge transfer and forgetting. *arXiv preprint arXiv:2112.01653*, 2021.

[19] Zixuan Ke, Bing Liu, Nianzu Ma, Hu Xu, and Lei Shu. Achieving forgetting prevention and knowledge transfer in continual learning. *Advances in Neural Information Processing Systems*, 34:22443–22456, 2021.

[20] Chris Dongjoo Kim, Jinseo Jeong, and Gunhee Kim. Imbalanced continual learning with partitioning reservoir sampling. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*, pages 411–428. Springer, 2020.

[21] 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.

[22] Hyunseo Koh, Dahyun Kim, Jung-Woo Ha, and Jonghyun Choi. Online continual learning on class incremental blurry task configuration with anytime inference. *arXiv preprint arXiv:2110.10031*, 2021.

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

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

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

[26] 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.

[27] 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.

[28] Wei Liu, Karoll Quijano, and Melba M Crawford. Yolov5-tassel: detecting tassels in rgb uav imagery with improved yolov5 based on transfer learning. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 15:8085–8094, 2022.

[29] Yaoyao Liu, Bernt Schiele, and Qianru Sun. Rmm: Reinforced memory management for class-incremental learning. *Advances in Neural Information Processing Systems*, 34:3478–3490, 2021.

[30] 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*, pages 12245–12254, 2020.

[31] Xiaoling Luo, Xiaobo Ma, Matthew Munden, Yao-Jan Wu, and Yangsheng Jiang. A multisource data approach for estimating vehicle queue length at metered on-ramps. *Journal of Transportation Engineering, Part A: Systems*, 148(2):04021117, 2022.

[32] 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.

[33] Yongsheng Mei, Tian Lan, and Guru Venkataramani. Exploiting partial common information microstructure for multi-modal brain tumor segmentation. *arXiv preprint arXiv:2302.02521*, 2023.

[34] Pedro Morgado and Nuno Vasconcelos. Nettailor: Tuning the architecture, not just the weights. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3044–3054, 2019.

[35] Kengo Nakata, Youyang Ng, Daisuke Miyashita, Asuka Maki, Yu-Chieh Lin, and Jun Deguchi. Revisiting a knn-based image classification system with high-capacity storage. In *European Conference on Computer Vision*, pages 457–474. Springer, 2022.

[36] 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.

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

[38] 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*, pages 11321–11329, 2019.

[39] 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.

[40] Quang Pham, Chenghao Liu, and Steven Hoi. Continual normalization: Rethinking batch normalization for online continual learning. *arXiv preprint arXiv:2203.16102*, 2022.

[41] 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.

[42] 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.

[43] 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.

[44] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[45] 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.

[46] Shixiang Tang, Dapeng Chen, Jinguo Zhu, Shijie Yu, and Wanli Ouyang. Layerwise optimization by gradient decomposition for continual learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 9634–9643, 2021.

[47] Nikolaos Tsilivis and Julia Kempe. What can the neural tangent kernel tell us about adversarial robustness? *arXiv preprint arXiv:2210.05577*, 2022.

[48] 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*, pages 1245–1254, 2021.

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

[50] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. *arXiv preprint arXiv:2002.06715*, 2020.

[51] 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*, pages 374–382, 2019.

[52] Jinlin Xiang, Shane Colburn, Arka Majumdar, and Eli Shlizerman. Knowledge distillation circumvents nonlinearity for optical convolutional neural networks. *Applied Optics*, 61(9):2173–2183, 2022.

[53] 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.

[54] Dan Zhang, Fangfang Zhou, Yuwen Jiang, and Zhengming Fu. Mm-bsn: Self-supervised image denoising for real-world with multi-mask based on blind-spot network. In *Proceed-*

ings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4188–4197, 2023.

[55] Yang Zheng, Jinlin Xiang, Kun Su, and Eli Shlizerman. Bimaml: Balanced incremental approach for meta learning. *arXiv preprint arXiv:2006.07412*, 2020.

[56] Fangfang Zhou, Zhengming Fu, and Dan Zhang. High dynamic range imaging with context-aware transformer. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2023.