IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 11 July 2023, accepted 30 July 2023, date of publication 9 August 2023, date of current version 16 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3303869

RESEARCH ARTICLE

A Continual Learning Algorithm Based on Orthogonal Gradient Descent Beyond Neural Tangent Kernel Regime

DA EUN LEE, KENSUKE NAKAMURA, JAE-HO TAK, AND BYUNG-WOO HONG
Department of Artificial Intelligence, Chung-Ang University, Seoul 06974, South Korea

Corresponding author: Byung-Woo Hong (hong@cau.ac.kr)

This work was supported in part by the Institute for Information and Communication Technology Planning and Evaluation (IITP) funded by the Korean Government [Ministry of Science and ICT (MSIT)] through the Artificial Intelligence Graduate School Program, Chung-Ang University, under Grant 2021-0-01341; in part by the National Research Foundation of Korea under Grant NRF-RS-2023-00251366; and in part by the Chung-Ang University Graduate Research Scholarship, in 2021.

ABSTRACT Continual learning aims to enable neural networks to learn new tasks without catastrophic forgetting of previously learned knowledge. Orthogonal Gradient Descent algorithms have been proposed as an effective solution to mitigate catastrophic forgetting. However, these algorithms often rely on the Neural Tangent Kernel regime, which imposes limitations on network architecture. In this study, we propose a novel method to construct an orthonormal basis set for orthogonal projection by leveraging Catastrophic Forgetting Loss. In contrast to the conventional gradient-based basis that reflects an update of model within an infinitesimal range, our loss-based basis can account for the variance within two distinct points in the model parameter space, thus overcoming the limitations of the Neural Tangent Kernel regime. We provide both quantitative and qualitative analysis of the proposed method, discussing its advantages over conventional gradient-based baselines. Our approach is extensively evaluated on various model architectures and datasets, demonstrating a significant performance advantage, especially for deep or narrow networks where the Neural Tangent Kernel regime is violated. Furthermore, we offer a mathematical analysis based on higher-order Taylor series to provide theoretical justification. This study introduces a novel theoretical framework and a practical algorithm, potentially inspiring further research in areas such as continual learning, network debugging, and one-pass learning.

INDEX TERMS Catastrophic forgetting, continual learning, neural tangent kernel, orthogonal gradient descent, orthogonal projection.

I. INTRODUCTION

Continual learning is an essential building block of real-time learning [1], [2], [3], [4] and artificial general intelligence [5]. Unlike the conventional batch learning (also referred to as offline learning in some contexts [6]), wherein neural networks are trained exclusively during the training phase and remain unaltered during the test time, continual learning involves the incremental training of a non-independent and identically distributed (Non-IID) sequence of tasks. However, fine-tuning neural networks for a new task with different distributions often results in a performance decline in previous tasks. This is a well-known catastrophic forgetting problem [7], [8]. Although retraining neural networks from scratch with all past data can effectively mitigate the forgetting, joint training every time a new data batch is added incurs significant overhead. For this reason, extensive researches have been undertaken to efficiently address the forgetting problem in various deep learning domains, such as robotics [9], natural language processing [10], image classification [11], [12], and others [6].

One notable algorithm in continual learning is Orthogonal Gradient Descent (OGD) [13]. OGD is a projection-based method founded on two key principles. First, during the training of a new task, the model is restricted from moving in the direction of large forgetting. This can be achieved by

The associate editor coordinating the review of this manuscript and approving it for publication was Amjad Ali.

removing the directional component from the gradient vector of the loss for the new task using orthogonal projection. Second, in order to identify the direction of severe forgetting, OGD computes the gradient vector of the model prediction on the previous data. However, using the gradient of the model function can only be justified under the Neural Tangent Kernel (NTK) regime, where the network width is assumed to be infinite [14], [15] in NTK Theory [16], [17]. In the case of narrow or deep networks, the abrupt change in model predictions leads to a violation of the NTK regime. Outside the NTK regime, the gradient-based OGD algorithms result in substantial performance degradation.

In this paper, we propose a novel method for discovering the directions of large forgetting by training an orthonormal basis set using Catastrophic Forgetting loss (CF loss). Introducing a loss function that accounts for the variance between a pair of two distinct points eliminates the necessity for the NTK regime condition, which we believe contributes to the performance enhancement of our algorithm for both narrow and deep networks. Our method facilitates overcoming architectural limitations of OGD, thereby paving the way for its utilization in practical applications.

II. RELATED WORKS

A. REPLAY-BASED METHODS

The most straightforward continual learning algorithms are replay methods. Experience replay [18], [19], [20] trains a new task with replay data, which consists of exemplar sets from past tasks stored in a fixed-sized memory. The main interest in replay methods is to develop an effective selection of core exemplars that most help defy catastrophic forgetting. Maximally Interfered Retrieval (MIR) [21] ranks exemplars based on the growth of loss during the virtual update phase. Several variants based on the same underlying concept have been proposed [22], [23], [24], [25], yet the direct use of replay data under limited memory restricts obtaining a comprehensive representation of the entire task. Generative Replay [26] adopts pseudo-rehearsal data generated by generative adversarial networks, removing reliance on sample selection. However, the efficacy of this technique depends on the quality of the generative model. Latent Replay [26] represents a different attempt to bypass direct storage of replay data, as it saves and replays latent vectors instead of the actual data. Even though the encoding of data knowledge offers an alternative solution to naive sample selection, individual latent vectors still originate from single data points, which fail to capture the task's complete features.

B. PARAMETER ISOLATION METHODS

Parameter isolation methods, which involve freezing certain portions of model parameters, are conceptually similar to dropout [27], [28], [29], [30], [31], [32]. Reference [27] examines how dropout functions as a flexible gating mechanism, demonstrating that the allocation of distinct pathways to separate tasks helps to alleviate interference between tasks during parameter update. PackNet [29] implements controlled version of network pruning. PathNet [28] introduces an genetic algorithm that utilizes selection, replication, mutation processes to search for optimal pathways. Progressive Networks [31] design dynamically growing neural networks with masking trained parameters, and Dynamically Expandable Networks (DEN) [32] establish criteria for determining when and how many nodes or layers should be added. All of these methods exhibit a binary state feature (on/off) for constraints with individual parameter tuning, which in turn limits the potential for more flexible constraints.

C. REGULARIZATION-BASED METHODS

The goal of regularization methods is to effectively balance between learning new tasks (plasticity) and retaining prior knowledge (stability). These methods penalize significant changes in parameters in accordance with their importance of previous tasks. The most popular regularizer is the L2 penalty loss, but it constraints all parameters based solely on their magnitude and neglects their correlation with past tasks. Elastic Weight Consolidation (EWC) [33], [34] assesses per-parameter importance weights via the Fisher information matrix, whereas Memory Aware Synapses (MAS) [35] employs the model's sensitivity to its changes. Similarly, Synaptic Intelligence (SI) [36] measures the weight importance by tracking individual parameter contribution to a drop in loss over the entire learning trajectory. Learning without Forgetting (LwF) [37] applies distillation loss between the current model (student) and the previous model (teacher) to maintain consistency in their predictions on old tasks. Variational Continual Learning (VCL) [38] combines approximate Bayesian inference with the Monte Carlo method. The soft and parameter-adaptive constraints of regularization methods are noteworthy. However, integrated nature of regularization in penalty loss terms prevents the ability to deliberately exclude diverse forgetting contributions from the loss gradient.

D. PROJECTION-BASED METHODS

Elementary approach for minimizing interference among multiple tasks would be breaking down model parameters into several segments and exclusively training the specific segment assigned to each task. A more advanced approach is to map the knowledge of each task onto specific parameter subspace orthogonal to each other. References [39] and [40] proposed different projection operators to achieve the same objective. Gradient Episodic Memory (GEM) [41], [42] projects the loss gradient vector to the closest gradient that minimizes the increase in loss for previous tasks stored in memory. Contrastively, in Orthogonal Gradient Descent (OGD) [13], the loss gradient is orthogonally projected onto the gradient of the model's predictions for past data. OGD+ [16] builds the OGD basis by incorporating not only the current task but also samples stored in memory. PCA-OGD [17] proposes an alternative OGD basis based

on the eigenvector set encoded from the gradient of the model by principal component analysis (PCA) instead of using the gradient itself. Reference [43] also improves OGD exploiting incremental principal component analysis (IPCA) and orthogonal recursive fitting (ORFit). Motivated by the knowledge compression concept in PCA-OGD, our research presents a further enhanced technique for distilling information from large datasets into compact basis vectors.

III. PRELIMINARIES

A. PROBLEM DEFINITION OF CONTINUAL LEARNING

The objective of continual learning is to train a model $f_\omega : \mathcal{X} \mapsto \mathcal{Y}$ with a set of associated model parameters ω where $\mathcal{X}$ denotes a feature space and $\mathcal{Y}$ denotes a label space. It is assumed that a sequence of Non-IID tasks $\{T_1, T_2, \cdots, T_k\}$ is given to train the model where a temporal task T_k at time k consists of (x, y) a pair of input $x \in \mathcal{X}$ and output $y \in \mathcal{Y}$. In the course of training for task T_k, any data (x, y) in the previous tasks $\{T_s | s < k\}$ is assumed to be unavailable unless the use of reply buffer is introduced. The optimal set of model parameters ω^* for task T_k is obtained as an optimization problem of the following empirical loss:

$$L_k(\omega) = \frac{1}{|T_k|} \sum_{(x,y)\in T_k} \ell(x, y), \tag{1}$$

where $|T_k|$ is the cardinality of task set T_k and $\ell(x, y)$ is a discrepancy measure for a given pair (x, y) between label y and prediction $f(x; \omega) = f_\omega(x)$.

Before diving into the details of our proposed method, it is crucial to briefly discuss the continual learning scenarios. Among the several well-known scenarios [12], [44], this paper concentrates on task incremental and domain incremental settings to highlight the performance advantages of our method in comparison with other baselines. Task incremental scenarios consist of multiple tasks with distinct problems, whereas domain incremental scenarios assume all tasks originate from the same problem with an identical number of classes but varying data distributions. Therefore, task incremental scenarios necessitate task-specific multi-heads and task-ids, whereas domain incremental scenarios only require a single-head with no task-ids [45], as demonstrated in Table 1. Note that in both scenarios, each data pair (x, y) in a single task T_k is commonly assumed to be a data batch with certain size, unless we are considering real-time continual learning scenarios. When training each task, mini-batches are randomly sampled from the data batch and then trained based on usual Stochastic Gradient Descent (SGD).

B. ORTHOGONAL GRADIENT DESCENT

In this section, we provide brief introduction of OGD algorithm in [13]. The OGD algorithm considers the modification of empirical gradient of the loss function during the training using task T_k at time k as follows:

$$\tilde{g} = g - \sum_{u\in S_{k-1}} \text{proj}_u(g), \tag{2}$$

where S_{k-1} denotes the orthonormal basis set obtained from the previous task T_{k-1} at time $k-1$, $g = \nabla_\omega L_k(\omega)$, and the projection operator proj of gradient g in the direction of vector u is defined by:

$$\text{proj}_u(g) = \frac{g^T u}{u^T u} u, \tag{3}$$

where v^T means the transpose of the vector v and $v^T u$ is the inner product of the vector v and u. The gradient descent step is performed in the modified gradient direction $\tilde{g}$ as follows:

$$\omega \leftarrow \omega - \eta\, \tilde{g}, \tag{4}$$

where $\eta > 0$ denotes a learning rate.

In the computation of orthonormal basis set S_k for task T_k at time k, the original OGD method [13] computes the gradient of the model prediction for each data in task T_k and each class label $c \in \mathcal{Y}$ as follows:

$$S_k = S_{k-1} \cup \{\nabla_\omega f_c(x; \omega_k^*)\}_{x\in T_k,\, c\in\mathcal{Y}}, \tag{5}$$

where ω_k^* denotes the optimal point of the model parameter for the task T_k, $\nabla_\omega f_c(x; \omega_k^*)$ is the gradient of the c-th logit with respect to the model parameter ω at point $\omega = \omega_k^*$. The total number of basis for each task is given by multiplying the number of entities in task T_k and the number of classes in the label space $\mathcal{Y}$ (OGD-ALL). An alternative option is OGD-GTL [13], which selects a single output logit corresponding to the ground truth label and thereby reduces the number of basis in S_k. However, this method is essentially an approximation based on empirical experiments without theoretical guarantees. Even without using such empirical approximations, our method still possesses its own memory and computation-saving strategy.

C. NEURAL TANGENT KERNEL

Later theoretical studies on OGD [16], [17] argued that using the gradient of the model for orthogonal projection can be justified within the Neural Tangent Kernel (NTK) framework. Stemming from the famous kernel trick, the kernel function is defined by the inner product of the feature map $\phi(x)$, described below:

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j), \tag{6}$$

where $\phi(x)$ is a mapping function from a simple data space to a complex feature space. In a nutshell, the kernel trick enables classification of nonlinearly distributed data using a linear classifier, without requiring direct computation of the feature maps. When the width of the model network is assumed to be infinite, the change of f with respect to ω occurs at a slow pace, allowing the update of the network to be approximated by a linear model as follows [14], [15], [16], [17]:

$$f(\omega) = f(\omega_0) + (\omega - \omega_0)^T \nabla f(\omega_0), \tag{7}$$

where ω_0 stands for the initial point of the model parameter and $\nabla f(\omega_0) = \nabla_\omega f(x; \omega_0)$. For simplification, we are assuming that f is a scalar function in the calculations. Letting the

feature map $\phi(x)$ be $\nabla_\omega f(x, \omega)$ allows us to define the Neural Tangent Kernel as

$$K(x_i, x_j) = \nabla_\omega f(x_i, \omega)^T \nabla_\omega f(x_j, \omega). \tag{8}$$

Due to the linearity of $f(\omega)$, the feature maps in the NTK are constant with respect to ω: $\phi(x) = \nabla_\omega f(x, \omega) = \nabla_\omega f(x, \omega_0)$. We now introduce the gradient flow of the model parameter, a useful theoretical concept for studying learning dynamics of the networks:

$$\frac{d\omega(t)}{dt} = -\nabla L(\omega(t)), \tag{9}$$

which is essentially a continuous form of the parameter update equation. As we are interested in the model's updates, taking a look at the gradient flow of the model function instead of the parameter would be more beneficial.

$$\frac{df(\omega)}{dt} = \nabla f(\omega)^T \frac{d\omega}{dt}, \tag{10}$$

where the right-hand side comes from the chain rule. Applying (9) and MSE loss $L(\omega) = \frac{1}{2}\|f(\omega) - y\|_2^2$, (10) becomes

$$\frac{df(\omega)}{dt} = -\nabla f(\omega)^T \nabla f(\omega)(f(\omega) - y) = -K(f(\omega) - y). \tag{11}$$

The conclusion of (11) is that the dynamics of neural networks can be tracked analytically by introducing the NTK. Under this framework, [16] presented that continual learning can be expressed as recursive kernel regression.

In this context, the NTK regime refers to the linearized regime of the model function. For instance, given a sufficiently large network width and a small learning rate, the model is close to the linear function with negligible error in comparison to infinitely wide networks. By setting $\omega = \omega_{k+1}^*$ and $\omega = \omega_k^*$ in (7), followed by substituting the two equations, we get

$$f(\omega_{k+1}^*) - f(\omega_k^*) = (\omega_{k+1}^* - \omega_k^*)^T \nabla f(\omega_0). \tag{12}$$

(12) implies that the update of the model is linearly dependent on its gradient. In conclusion, under the NTK regime, simply computing the gradient of the model is sufficient to predict the direction that will lead the model to the most significant change. However, when networks have a smaller dimensional width, the linear assumption is no longer valid. Additionally, we empirically found that an increase in the number of layers leads to faster model variation, which also breaks the NTK regime. One potential solution of this problem is to reduce the size of the learning rate [15]. Nonetheless, the deeper the networks, the smaller the learning rates required, potentially causing a considerable slowdown in network training. In Section IV-B, we will discuss how to address this problem.

IV. ORTHOGONAL GRADIENT DESCENT BASED ON CATASTROPHIC FORGETTING LOSS

In this section, we explain how to train an orthonormal basis set (OGD basis) using Catastrophic Forgetting Loss (CF Loss). We define necessary loss functions, summarize full algorithm, and discuss how and under which condition our algorithm can surpass gradient-based methods.

A. ORTHONORMAL BASIS BASED ON CATASTROPHIC FORGETTING LOSS

Following the general form of the CF suggested in [17], we define CF loss as follows:

$$\phi_k(U) = -\frac{1}{d|T_k|} \sum_{u \in U} \sum_{x \in T_k} \|f(x; \omega_k^* + u) - f(x; \omega_k^*)\|_2^2, \tag{13}$$

where u is a basis vector in an orthonormal basis set $U = (u_1\ u_2\ \cdots\ u_d)$ with d bases, and $\|A\|_2$ is the L2 norm of the vector A.

From this equation, our objective is to train U to minimize the CF Loss ϕ_k, thereby maximizing CF. To achieve this, we first initialize the basis vector by sampling from a normal distribution: $u \sim N(0, I)$. Subsequently, the magnitude of u is normalized to designated length l. Then, u is updated iteratively using gradient descent with ϕ_k. However, without any constraints on the length of u, the CF loss can decrease boundlessly (resulting CF divergence) as the point $w_k^* + u$ moves farther away from w_k^*. To address this problem during training a direction of u, we introduce an additional loss term to maintain u to the designated length l as follows:

$$\psi_k(U) = \sum_{u \in U} (\|u\|_2 - l)^2. \tag{14}$$

We now consider an extra penalty loss term for the orthogonality condition between the basis vectors. We start with the orthogonality condition between two vectors:

$$u_i \perp u_j \text{ if } i \neq j \tag{15}$$

or

$$u_i^T u_j = 0 \text{ if } i \neq j. \tag{16}$$

When training three basis vectors, for example, we need to consider all combination pairs of vectors in $U = (u_1\ u_2\ u_3)$, which can be represented by the matrix form $U^T U$ as shown in (17).

$$U^T U = \begin{pmatrix} u_1^T u_1 & u_1^T u_2 & u_1^T u_3 \\ u_2^T u_1 & u_2^T u_2 & u_2^T u_3 \\ u_3^T u_1 & u_3^T u_2 & u_3^T u_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{17}$$

Note that the diagonals are unit vector conditions. Since the matrix is symmetric, all we need is an off-diagonal lower triangular part to complete the orthogonality constraint loss ρ_k.

$$\begin{aligned} \rho_k(U) &= \sum_{i=2}^{3} \sum_{j=1}^{i-1} (u_i^T u_j)^2 \\ &= \frac{1}{2}[\|U^T U\|_F^2 - tr((U^T U) \circ (U^T U))], \end{aligned} \tag{18}$$

where $\|A\|_F$ is the Frobenius norm of the matrix A, $tr(A)$ is the trace of the matrix A, and $A \circ B$ is the Hadamard product of the matrix A and B.

Finally, our total objective function becomes the combination of all loss terms:

$$\mathcal{E}_k(U) = \phi_k(U) + \lambda_1 \psi_k(U) + \lambda_2 \rho_k(U), \tag{19}$$

where λ_1, λ_2 are the distance and the orthogonality penalty coefficient, respectively.

The full algorithm of the proposed method is summarized in Algorithm 1.

Algorithm 1 Training of an OGD Basis via CF Loss

Input: Model parameters ω_k^*, a data batch of the current task $(x, y) \in T_k$, a learning rate ξ, distance l, epoch N, a number of bases d, penalty coefficients λ_1, λ_2, orthonormal basis set for the previous task S_{k-1}

Output: orthonormal basis set for the current task S_k

$S_k = \{\}, U = \{\}$ (initialize empty sets)

$S_k \leftarrow S_k \cup S_{k-1}$

for $i = 1$ **to** d **do**

  $u \sim N(0, I)$

  $u \leftarrow \frac{l}{\|u\|} u$

  $U \leftarrow U \cup \{u\}$

for $n = 1$ **to** N **do**

  CF Loss:

  $\phi_k(U) = -\frac{1}{d|T_k|} \sum_{x \in T_k} \sum_{u \in U} \|f(x; \omega_k^* + u) - f(x; \omega_k^*)\|_2^2$

  Distance Loss:

  $\psi_k(U) = \sum_{u \in U} (\|u\|_2 - l)^2$

  Orthogonality Loss:

  $\rho_k(U) = \frac{1}{2}[\|U^T U\|_F^2 - tr((U^T U) \circ (U^T U))]$

  Total Loss:

  $\mathcal{E}_k(U) = \phi_k(U) + \lambda_1 \psi_k(U) + \lambda_2 \rho_k(U)$

  for $u \in U$ **do**

    $u \leftarrow u - \xi \nabla_u \mathcal{E}_k(U)$

for $u \in U$ **do**

  $u \leftarrow$ Gram-Schmidt(S_k, u) (Orthonormalize u)

  $S_k \leftarrow S_k \cup \{u\}$

B. COMPARATIVE THEORETICAL ANALYSIS OF PROPOSED ALGORITHM

In the preceding section, we introduced a novel OGD algorithm that trains an orthonormal basis set utilizing CF loss. In this section, we explore the circumstances under which this method can outperform conventional OGD algorithms that rely on the model's gradient. We begin with the Taylor expansion of $f(\omega)$ at the point ω_k^*:

$$f(\omega) = f(\omega_k^*) + (\omega - \omega_k^*)^T \nabla f(\omega_k^*) + \frac{1}{2}(\omega - \omega_k^*)^T H(\omega_k^*)(\omega - \omega_k^*) + \cdots, \tag{20}$$

where $H(\omega_k)$ represents the Hessian matrix of f at point ω_k^*. When $\omega = \omega_k^* + u$, (20) transforms into:

$$f(\omega_k^* + u) = f(\omega_k^*) + u^T \nabla f(\omega_k^*) + \frac{1}{2} u^T H(\omega_k^*) u + \cdots \tag{21}$$

or

$$f(\omega_k^* + u) - f(\omega_k^*) = u^T \nabla f(\omega_k^*) + \frac{1}{2} u^T H(\omega_k^*) u + \cdots. \tag{22}$$

The left-hand side is the drift term between the points $\omega_k^* + u$ and ω_k^*, which is included in the CF loss formula in (13). Comparing (22) with (12), the CF loss accounts for higher-order Taylor series terms, whereas gradient-based methods only consider the first order. The higher-order error becomes significant when the variation of f across ω is rapid, which is typical in narrow networks (with small width) or deep networks (with a large number of layers). In such cases, taking higher-order terms into account leads to a more precise prediction of the desired basis set. Consequently, the CF loss-based approach is generally more practical in a broader range of model architectures.

V. EXPERIMENTS

In this section, we provide quantitative evaluation of the proposed CF-loss based OGD algorithm in comparison to the conventional gradient-based baselines. We conducted three types of experiments: 1. Characteristic investigation, 2. Architectural investigation, 3. Comparative experiments. For characteristic investigation, we examined the effect of distance (length of u) and batch size to determine optimal settings. In our architectural investigation, we studied networks with diverse depths and widths to exhibit the influence of violating the NTK regime. Finally, we compared our method with other baselines, such as SGD, OGD [13], PCA-OGD [17]. Here, SGD stands for naive fine-tuning without any continual learning algorithms. Following the experimental setup in [17], the comparative experiments were performed on three popular image benchmark datasets: Split MNIST, Permuted MNIST, Split CIFAR-100. In our continual learning scenarios, we adopted task incremental for Split MNIST and Split CIFAR-100, and domain incremental for Permuted MNIST. Unlike the previous work [17], we selected sufficiently long training epochs to ensure complete model convergence, as early stopping based on future tasks is an unrealistic assumption. In addition, we applied the full dataset in PCA (e.g., 10,000 data for PermutedMNIST), as opposed to the 3,000 data used by [17]. Both characteristic and architectural investigations were conducted based on the Split MNIST setup. For the purpose of a fair comparison, we fixed the basis set size to be consistent across all algorithms. The specification of our experimental setup is outlined in Table 1.

In all experiments, we used a GeForce RTX 3090 with 24GB GPU memory. Every single test point was computed via single GPU. Across all experimental setups (Split MNIST, Permuted MNIST, Split CIFAR-100), the model training computation time was less than 10 minutes for the initial

task and under 30 minutes for the last task. Again, in all experiments, the time elapsed for computing the OGD basis for both OGD and PCA-OGD was negligible, as it took less than a minute per task. On the other hand, our algorithm spent around 20 to 30 minutes per task in all experiments. Comparing memory usage accurately is challenging due to PyTorch's built-in matrix multiplication function, which dynamically allocates and frees memory for parallel computation. All three–OGD, PCA-OGD, and our algorithm–use matrix multiplication operations in the orthogonal projection, singular value decomposition, and training of an orthonormal basis set as a matrix form. Under this environment, we found that OGD, PCA-OGD, and our algorithm all had comparable memory consumption, with all algorithms effectively running within a 24GB GPU memory in all test conditions.

For quantitative evaluation, we measured the test accuracy for all tasks after training each task, and then computed the average accuracy through the widely used formula [12]:

$$A_i = \frac{1}{i}\sum_{j=1}^{i} a_{ij}, \tag{23}$$

where a_{ij} is the accuracy of the j-th test set after completing the learning of the i-th training set.

A. CHARACTERISTIC INVESTIGATION

Initially, we examined two hyperparameters newly introduced in our method (distance, batch size) in order to discover the optimal values. As depicted in Fig. 1(a), the heuristically determined best distance was found to be 0.2 for shallow networks (2-3 layers) and 0.5 for deeper networks (4-6 layers). The fact that looking at farther distance yielded better results for deeper networks fits well in our intuition. Distances smaller than 0.1 or larger than 1 led to a significant drop in accuracy or even made training challenging due to excessively small or large CF loss. Very small CF loss values can lead the training too slow to converge, whereas too large CF loss values can make the training diverge. To determine whether training OGD basis has converged well, we followed the process as described below: 1. Save a loss value for a certain point. 2. When the updated loss changes less than a specified threshold ratio (e.g. 0.05) compared to the reference loss, we increment the stopping counter by one. 3. Should the difference between the updated loss and the reference loss exceed the threshold ratio, the stopping counter is reset. 4. When the stopping counter hits a predetermined number, such as 500 iterations, we consider the training to have converged. Under these criteria, we obeserved all the loss curves of the experiments performed in this section are well converged within the iterations presented in Table 1.

Concerning the batch size, as illustrated in Fig. 1(b), we observed that the peaks (indicated by circles in the plots) emerged for batch sizes within the range of 50 to 200. Nevertheless, no distinct trends in performance variations were evident, leading us to conclude that employing mini-batch gradient descent do not damage basis training. Even with the use of mini-batch training, the complete dataset is trained eventually, guaranteeing that the task's entire attribute is embedded within the basis. Consequently, training the basis set using a small batch size proves to be an effective tactic for minimizing memory consumption and computational expenses without sacrificing performance.

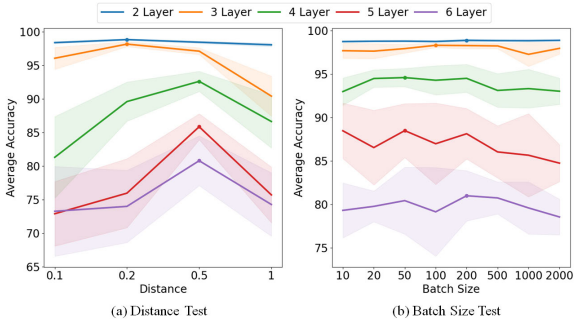


FIGURE 1. Average accuracy for Split MNIST over (a) the length of u and (b) the batch size. Each max point is indicated by a circle, respectively. (100 hidden dim, averaged over 8 seeds).

Taking these observations into account, we applied the optimized hyperparameters for all subsequent experiments.

B. ARCHITECTURAL INVESTIGATION

In contrast to [13] and [17], which employed only a 2-layer MLP, we tested MLPs ranging from 2 to 6 layers. The results presented in Fig. 2(a) reveals that the average accuracy of conventional methods descends rapidly as the number of layer increases. We interpret this outcome as a consequence of deep networks violating the NTK regime, as discussed in Section IV-B. Our method exhibited a remarkable improvement in accuracy, particularly for 4 and 5-layer cases.

Subsequently, we studied the impact of the network's width (Fig. 2(b)). The accuracy of baseline methods significantly deteriorated for very narrow widths, whereas our method displayed minimal change. Once again, this result is attributed to the fact that a sufficiently large width is a necessary condition for the NTK regime.

The comprehensive study on the impact of layer and width are depicted in Fig. 3. In each heatmap, the prevalence of bright squares implies a lesser performance drop, given an increase in layer size or a decrease in the hidden dimension. As observed in the results, our algorithm sustains a better average accuracy when compared to traditional gradient-based OGD algorithms.

To Confirm the performance enhancement shown in Fig. 2 is a result of the higher-order effect discussed in section IV.B, we conducted the following proof experiment. The test was conducted under the single-task Permuted MNIST setup. Once training the initial task had been concluded, we measured CF loss $\phi_{0.1}$ at $\|u\|_2$=0.1 and $\phi_{1,real}$ at $\|u\|_2$=1, averaging over 1,000,000 random basis vectors u. As a next step, we computed $\phi_{1,predict}$ and ϕ_{error} through the following equations:

$$\phi_{1,predict} = \frac{1}{0.1}\phi_{0.1} \tag{24}$$

TABLE 1. Experimental setup.

Model Training			
Epoch	500		
Optimizer	SGD		
Learning rate	0.001		
Batch size	32		
# basis	100		
Basis Training			
Optimizer	Adam		
Learning rate	0.0001		
Batch size	128		
λ_1	10000		
λ_2	100	100	10
Distance	0.5	0.5	0.2
Iteration	5000	2000	2000
Dataset	Split MNIST	Permuted MNIST	Split CIFAR-100
Data size per task	2000	10000	2500
Architecture	MLP	MLP	LeNet
# Conv layer	0	0	2
# Conv1 channel	0	0	20
# Conv2 channel	0	0	50
# linear hidden layer	4	4	1
# linear hidden dimension	100	100	200
# heads	5	1	20
output dimension	2	10	5
Scenario	Task Incremental	Domain Incremental	Task Incremental
# tasks	5	15	20

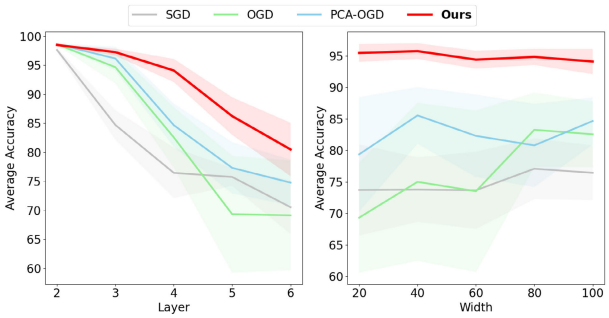


FIGURE 2. Average accuracy (in percent) for Split MNIST over (a) the size of the layer (with a fixed 100 hidden dimension) and (b) the width (with a fixed 4 layers). (averaged over 8 seeds) The curves with saturated color represent the mean values and the translucent areas signify the range within the standard deviation.

$$\phi_{error} = \phi_{1,real} - \phi_{1,predict} \tag{25}$$

In the equations, $\phi_{1,predict}$ indicates the estimated change in the model function $\phi(u)$ when $\|u\|_2$=1, which is predicted from $\phi_{0.1}$ under a linear change assumption. $\phi_{1,real}$ signifies the actual measured value at the same point and ϕ_{error} is the discrepancy between the actual and prediction values. When $\phi(u)$ exhibits notably large non-linearity, it will be reflected in the non-linear error ϕ_{error}. This approach allows us to assess the impact of higer-order factors on the non-linear change in the model's predictions. The schematics and the experimental result are shown in Fig. 4(a) and (b), respectively. We observed a substantial rise in the higher-order error ϕ^2_{error} in deeper networks, which could explain the performance falloff of OGD and PCA-OGD algorithms. For 2-layer MLP network, the value of ϕ^2_{error} is negligible, on the order of 10^{-5}, whereas for the 6-layer network, it becomes 7260 times larger.

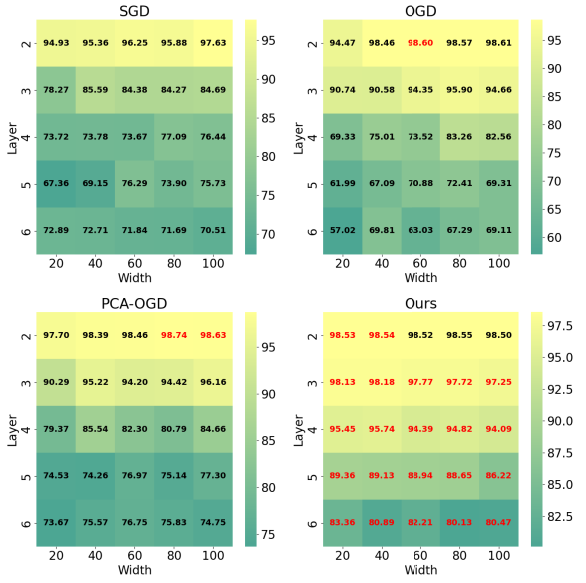
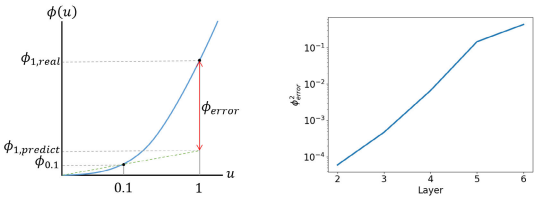


FIGURE 3. Average accuracy (in percent) for Split MNIST ranging over 2 to 6 layers and 20 to 100 in hidden dimension (width). (averaged over 8 seeds) The highest score among the algorithms for each condition is displayed in red text.

C. COMPARATIVE EXPERIMENTS

Lastly, we performed comparison tests on three datasets using the continual learning setup shown in Table 1. Following the previous work [17], in the Lenet architecture, only linear layers were updated except for the first task. The final accuracy for individual tasks as well as their averages, are presented in Fig. 5 and Table 2 for Split MNIST, Fig. 6 and Table 3 for Permuted MNIST, Table 4 for Split CIFAR-100, respectively. In Table 2-4, PCA stands for PCA-OGD. Note that Permuted MNIST experiments are implemented as an 8-task test for Fig. 6 and a 15-task test for Table 3. The setup used to



(a) Schematics of ϕ_{error} calculation (b) Measurement of ϕ^2_{error} by a number of layers

FIGURE 4. The proof experiment demonstrating the impact of the higher-order effect on the model function with respect to the number of layers (b) , along with the associated schematics (a). The higher-order terms induce a non-linear change, which is denoted as ϕ_{error} in (a).

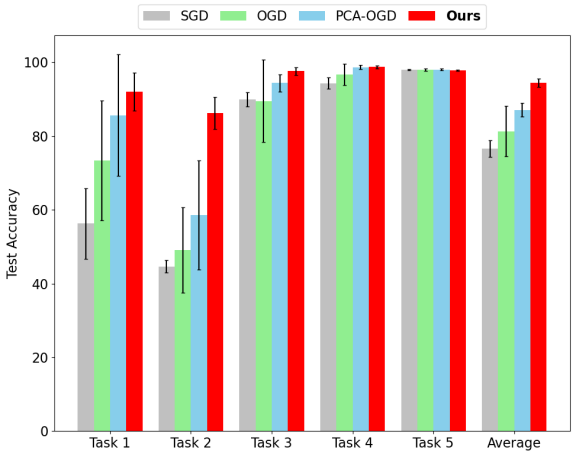


FIGURE 5. Final Accuracy (in percent) for Split MNIST. (5 tasks, averaged over 8 seeds) The bar heights represent mean values, while the error bars (shown as black I-beams) indicate the standard deviations.

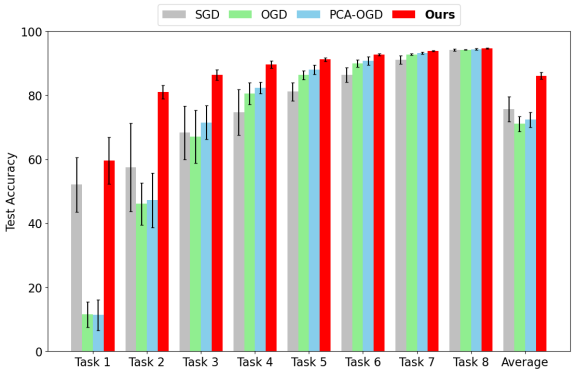


FIGURE 6. Final Accuracy (in percent) for Permuted MNIST. (8 tasks, averaged over 8 seeds) The bar heights represent mean values, while the error bars (shown as black I-beams) indicate the standard deviations.

perform the 8-task Permuted MNIST experiment is exactly the same as the one used for the 15-task test. Throughout all experimental scenarios, our method achieved the highest average accuracy, with an especially large margin for Permuted MNIST. Interestingly, the average accuracy depicted in Fig. 6 indicates that conventional OGD and PCA-OGD would perform worse than SGD when applied to deep networks. This finding suggests that gradient-based formation of an OGD basis may be inappropriate for deep network environments, due to the violation of the NTK regime.

Table 5 presents a comparison of various continual learning algorithms applied to different datasets. The experimental

TABLE 2. Final Accuracy (in percent) for Split MNIST. (5 tasks, averaged over 8 seeds) Each value is comprised of the mean (on the left) and the standard deviation (on the right).

	SGD	OGD	PCA	Ours
Task1	56.26±9.55	73.40±16.20	85.67±16.54	92.02±5.14
Task2	44.63±1.71	49.08±11.63	58.61±14.75	86.23±4.32
Task3	89.86±1.93	89.51±11.20	94.36±2.37	97.59±1.09
Task4	94.35±1.48	96.67±2.84	98.69±0.56	98.71±0.38
Task5	97.94±0.17	97.99±0.27	98.01±0.22	97.84±0.19
Average	76.61±2.21	81.33±6.82	87.07±1.85	94.48±1.10

TABLE 3. Final Accuracy (in percent) for Permuted MNIST. (15 tasks, averaged over 8 seeds) Each value is comprised of the mean (on the left) and the standard deviation (on the right).

	SGD	OGD	PCA	Ours
Task1	35.41±3.34	6.38±1.92	6.28±2.63	37.18±6.41
Task2	44.62±7.36	23.22±6.91	22.32±7.39	63.87±4.24
Task3	47.63±8.01	36.44±9.38	38.66±5.79	69.38±7.08
Task4	48.31±5.92	47.07±6.50	47.08±3.55	73.42±4.92
Task5	55.57±4.72	57.24±6.72	60.27±6.32	76.44±4.67
Task6	55.36±7.71	62.56±3.56	63.46±6.11	77.40±1.87
Task7	57.88±3.91	67.59±2.41	74.00±4.52	80.52±2.91
Task8	65.23±3.72	75.27±3.94	76.36±2.90	84.30±1.42
Task9	70.28±6.78	79.83±2.38	80.88±2.58	86.34±1.43
Task10	72.07±8.05	82.11±3.11	84.40±1.41	87.95±0.90
Task11	78.90±3.10	86.37±1.24	86.62±0.93	89.04±1.13
Task12	84.71±2.83	88.96±0.92	90.23±0.58	91.29±0.79
Task13	88.35±1.15	90.56±0.58	91.28±0.41	92.10±0.21
Task14	90.87±0.83	92.45±0.28	92.74±0.20	93.35±0.29
Task15	93.65±0.15	93.78±0.17	93.82±0.29	94.34±0.18
Average	65.92±1.76	65.99±2.01	67.23±1.59	79.79±1.31

TABLE 4. Final Accuracy (in percent) for Split CIFAR-100. (20 tasks, averaged over 8 seeds) Each value is comprised of the mean (on the left) and the standard deviation (on the right).

	SGD	OGD	PCA	Ours
Task1	66.17±4.78	73.50±1.26	71.60±1.97	73.20±1.59
Task2	65.40±7.02	71.58±3.78	71.82±2.21	74.88±1.98
Task3	68.70±3.44	72.67±1.67	72.20±1.78	72.62±1.41
Task4	60.73±5.06	66.95±1.82	67.18±1.30	67.38±1.34
Task5	73.60±6.15	85.10±1.85	85.72±1.24	85.53±0.81
Task6	57.53±4.90	69.08±2.39	68.65±2.17	71.08±1.33
Task7	68.15±8.66	76.15±1.28	75.70±1.79	77.12±1.14
Task8	56.12±2.96	70.38±2.46	68.85±0.57	71.23±1.73
Task9	55.52±10.15	70.15±1.78	69.28±1.74	71.07±1.64
Task10	72.18±5.95	81.47±0.96	81.97±1.00	82.50±0.61
Task11	76.18±4.69	82.03±1.52	83.53±1.39	82.52±0.92
Task12	66.53±4.04	75.32±1.19	75.08±1.24	75.90±2.56
Task13	75.12±4.78	83.57±0.92	83.43±1.71	83.67±1.44
Task14	68.10±5.23	76.03±0.97	76.53±1.80	78.32±1.25
Task15	70.70±4.29	78.75±0.75	78.38±1.25	79.42±1.08
Task16	60.53±6.50	72.20±1.52	72.00±1.74	72.35±1.28
Task17	64.58±6.66	75.03±1.66	75.30±1.78	76.23±1.64
Task18	68.95±3.36	74.35±1.27	75.53±1.33	74.93±1.15
Task19	78.22±1.98	82.20±2.01	82.10±1.02	82.48±1.34
Task20	80.33±1.52	80.90±0.92	81.10±1.10	80.30±1.70
Average	67.67±1.83	75.87±0.45	75.80±0.55	76.64±0.46

setup for each dataset corresponds to the one described in Table 1. We selected Elastic Weight Consolidation (EWC) [33] as a representative for the regularization-based method and Averaged Gradient Episodic Memory (A-GEM) [42] for the replay-based method. Note that A-GEM can be viewed as a hybrid of the replay and the projection method, as it incorporates saving of the raw exemplar set for replay and orthogonal projection for the parameter updates. In A-GEM, 100 examplars for each task were randomly sampled and saved into memory buffer. In EWC, the regularization constant of the penalty loss term is tuned as 50 for Split

TABLE 5. Average Accuracy (in percent) for various algorithms and datasets. (averaged over 8 seeds) Each value is comprised of the mean (on the left) and the standard deviation (on the right). The algorithms marked with an asterisk (*) require storing the real data in memory.

	Split MNIST	Permuted MNIST	Split CIFAR-100
SGD	76.61±2.21	65.92±1.76	67.67±1.83
OGD	81.33±6.82	65.99±2.01	75.87±0.45
PCA-OGD	87.07±1.85	67.23±1.59	75.80±0.55
EWC	84.50±3.61	76.76±1.86	71.30±1.55
A-GEM*	93.42±3.27	85.14±0.55	75.67±0.32
Ours	94.48±1.10	79.79±1.31	76.64±0.46

MNIST, 10 for Permuted MNIST and Split CIFAR-100. Other training hyperparameters are the same as shown in Table 1. Our algorithm beat all other baselines for Split MNIST and Split CIFAR-100 datasets, whereas A-GEM showed better performance for Permuted MNIST. Although the results imply that A-GEM may be a better option in certain scenarios, it does require the storage of raw data, which could be problematic for private personal data.

D. DISCUSSION

Before concluding this part, we would like to discuss a few limitations associated with the proposed method. First, the basis training takes considerably more time than the gradient-based method. Although this longer (yet realistic) training time may not be a concern for many deep learning applications, it could be problematic for real-time learning situations. Second, our method demands the calibration of two hyperparameters (λ_1, λ_2) without any fundamental principle for achieving optimal values. Further study is needed to develop an advanced optimization trick for the basis training. We suggest two possible solutions: firstly, inspired by OGD, we can avoid using penalty loss terms by implementing orthogonal projection for distance and orthogonality constraint; secondly, applying meta-learning could potentially enhance the speed of basis training.

We finish the discussion by presenting several potential applications of our advanced OGD algorithm. Besides continual learning, the no-forgetting ability of OGD can work for a wide range of deep learning areas. For example, [46] proposed unlearning and relearning algorithm based on OGD, which are necessary tricks for network debugging (also known as model editing). They intentionally induced forgetting in the model to erase mislearned information without losing important knowledge, then retrained the model using accurate data labels. Reference [43] applied OGD to one-pass learning, in which the model accesses each data point only once, resulting in much faster adaptation of the model training. We argue that this no-forgetting one-pass learning is not only more efficient than SGD, but also has the capacity to serve as a more effective generalization paradigm then repetition-based training.

VI. CONCLUSION

We presented a novel approach to generate an OGD basis, a crucial component for orthogonal projection in OGD algorithms. In order to tackle the architectural restrictions posed by the NTK regime, we devised an OGD basis training scheme that makes use of a loss function containing the model's discrepancy at two remote points. To the best of our knowledge, this is the first approach that suggests an alternative solution rather than using the model's gradient to obtain the OGD basis. We conducted characteristic, architectural, and comparative experiments on Split MNIST, Permuted MNIST, and Split CIFAR-100 datasets. In all experimental scenarios, our method significantly outperformed the state-of-the-art gradient-based OGD algorithms, especially for deep or narrow networks where the NTK regime is violated. Furthermore, we provided a mathematical analysis based on higher-order Taylor series to explain why our new basis is more accurate than gradient-based OGD basis. We also discussed the limitations of our methods, including additional training time and hyperparameter tuning, as well as potential applications like continual learning, network debugging, and one-pass learning. We anticipate that the new mathematical concepts and engineering techniques for neural networks introduced in this study will influence a wide range of deep learning research areas, including the OGD applications discussed above.

REFERENCES

[1] D. Li, S. Tasci, S. Ghosh, J. Zhu, J. Zhang, and L. Heck, "RILOD: Near real-time incremental learning for object detection at the edge," in *Proc. 4th ACM/IEEE Symp. Edge Comput.*, Nov. 2019, pp. 113–126.

[2] L. Pellegrini, G. Graffieti, V. Lomonaco, and D. Maltoni, "Latent replay for real-time continual learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 10203–10209.

[3] L. Pellegrini, V. Lomonaco, G. Graffieti, and D. Maltoni, "Continual learning at the edge: Real-time training on smartphone devices," in *Proc. ESANN*, 2021, pp. 23–28.

[4] R. Aljundi, K. Kelchtermans, and T. Tuytelaars, "Task-free continual learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11246–11255.

[5] D. L. Silver, "Machine lifelong learning: Challenges and benefits for artificial general intelligence," in *Artificial General Intelligence*. Cham, Switzerland: Springer, 2011, pp. 370–375.

[6] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Netw.*, vol. 113, pp. 54–71, May 2019.

[7] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychol. Learn. Motivat.*, vol. 24, pp. 109–165, Jan. 1989.

[8] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," 2013, *arXiv:1312.6211*.

[9] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez, "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges," *Inf. Fusion*, vol. 58, pp. 52–68, Jun. 2020.

[10] M. Biesialska, K. Biesialska, and M. R. Costa-Jussa, "Continual lifelong learning in natural language processing: A survey," in *Proc. 28th Int. Conf. Comput. Linguistics*, 2020, pp. 6523–6541.

[11] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3366–3385, Jul. 2022.

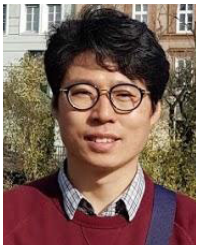
[12] Z. Mai, R. Li, J. Jeong, D. Quispe, H. Kim, and S. Sanner, "Online continual learning in image classification: An empirical survey," *Neurocomputing*, vol. 469, pp. 28–51, Jan. 2022.

[13] M. Farajtabar, N. Azizan, A. Mott, and A. Li, "Orthogonal gradient descent for continual learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 3762–3773.
[14] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–15.
[15] J. Lee, L. Xiao, S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington, "Wide neural networks of any depth evolve as linear models under gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–12.
[16] M. Abbana Bennani, T. Doan, and M. Sugiyama, "Generalisation guarantees for continual learning with orthogonal gradient descent," 2020, *arXiv:2006.11942*.
[17] T. Doan, M. A. Bennani, B. Mazoure, G. Rabusseau, and P. Alquier, "A theoretical analysis of catastrophic forgetting through the NTK overlap matrix," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2021, pp. 1072–1080.
[18] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "ICaRL: Incremental classifier and representation learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5533–5542.
[19] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, "Experience replay for continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.
[20] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, pp. 3302–3309.
[21] A. Rahaf and C. Lucas, "Online continual learning with maximally interfered retrieval," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–12.
[22] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, "Gradient based sample selection for online continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–10.
[23] A. Prabhu, P. H. Torr, and P. K. Dokania, "GDumb: A simple approach that questions our progress in continual learning," in *Proc. 16th Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 524–540.
[24] D. Shim, Z. Mai, J. Jeong, S. Sanner, H. Kim, and J. Jang, "Online class-incremental continual learning with adversarial Shapley value," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 11, pp. 9630–9638.
[25] Y. Gu, X. Yang, K. Wei, and C. Deng, "Not just selection, but exploration: Online class-incremental continual learning via dual view consistency," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 7432–7441.
[26] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–10.
[27] S. I. Mirzadeh, M. Farajtabar, and H. Ghasemzadeh, "Dropout as an implicit gating mechanism for continual learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 945–951.
[28] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra, "PathNet: Evolution channels gradient descent in super neural networks," 2017, *arXiv:1701.08734*.
[29] A. Mallya and S. Lazebnik, "PackNet: Adding multiple tasks to a single network by iterative pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7765–7773.
[30] A. Mallya, D. Davis, and S. Lazebnik, "Piggyback: Adapting a single network to multiple tasks by learning to mask weights," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 67–82.
[31] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," 2016, *arXiv:1606.04671*.
[32] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–11.
[33] K. James, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, and K. Milan, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.
[34] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 532–547.
[35] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 139–154.
[36] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3987–3995.
[37] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, Dec. 2018.
[38] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, "Variational continual learning," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–18.
[39] A. Chaudhry, N. Khan, P. Dokania, and P. Torr, "Continual learning in low-rank orthogonal subspaces," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 9900–9911.
[40] G. Zeng, Y. Chen, B. Cui, and S. Yu, "Continual learning of context-dependent processing in neural networks," *Nature Mach. Intell.*, vol. 1, no. 8, pp. 364–372, Aug. 2019.
[41] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–10.
[42] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-GEM," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–20.
[43] Y. Min, K. Ahn, and N. Azizan, "One-pass learning via bridging orthogonal gradient descent and recursive least-squares," in *Proc. IEEE 61st Conf. Decis. Control (CDC)*, Dec. 2022, pp. 4720–4725.
[44] G. M. van de Ven and A. S. Tolias, "Three scenarios for continual learning," 2019, *arXiv:1904.07734*.
[45] H. Hihn and D. A. Braun, "Hierarchically structured task-agnostic continual learning," *Mach. Learn.*, vol. 112, no. 2, pp. 655–686, Feb. 2023.
[46] N. Chilkuri and C. Eliasmith, "Debugging using orthogonal gradient descent," 2022, *arXiv:2206.08489*.

DA EUN LEE received the B.S. and M.S. degrees in physics from Chung-Ang University (CAU), Seoul, South Korea, and the Ph.D. degree from the Image Laboratory, AI Department, CAU. After completing his studies, he spent two years working as a web developer. His research interests include deep learning, online learning, and continual learning.

KENSUKE NAKAMURA received the M.S. and Ph.D. degrees in engineering from the Institute of Technology, Kyoto Institute of Technology, Japan, in 2004 and 2012, respectively. He is currently a Research Associate with Chung-Ang University. He has participated in research projects on models and applications of the two-/three-dimensional human body shapes. His current research interests include computer vision and a support system for fashion design.

JAE-HO TAK received the bachelor's degree in mechanical engineering from Chung-Ang University. After studying the prediction of fuel cell lifespan and the control of electric motors using machine learning, he began his master's degree program with the AI Department. His research interest includes improving the efficiency of optimization-based meta-learning.



BYUNG-WOO HONG received the M.S. degree in computer vision from the Weizmann Institute of Science, in 2001, under the supervision of Prof. Shimon Ullman, and the Ph.D. degree in computer vision from the University of Oxford, in 2005, under the supervision of Sir Prof. Michael Brady. He joined the Computer Science Department, Chung-Ang University, as a Faculty Member, in 2008, after his postdoctoral research with the Computer Science Department, University of California at Los Angeles, with Prof. Stefano Soatto. His research interests include image processing, computer vision, machine learning, and medical image analysis.