Formalizing the Generalization-Forgetting Trade-off in Continual Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

We formulate the continual learning (CL) problem via dynamic programming and model the trade-off between catastrophic forgetting and generalization as a two-player sequential game. In this approach, player 1 maximizes the cost due to lack of generalization whereas player 2 minimizes the cost due to catastrophic forgetting. We show theoretically that a balance point between the two players exists for each task and that this point is stable (once the balance is achieved, the two players stay at the balance point). Next, we introduce balanced continual learning (BCL), which is designed to attain balance between generalization and forgetting and empirically demonstrate that BCL is comparable to or better than the state of the art.

1 Introduction

In continual learning (CL), we incrementally adapt a model to learn tasks (tasks are defined according to the problem at hand) observed sequentially. CL has two main objectives: maintain long-term memory (remember previous tasks) and navigate new experiences continually (quickly adapt to new tasks). An important characterization of these objectives is provided by the stability-plasticity dilemma [9], where the primary challenge is to balance network stability (preserve past knowledge; minimize catastrophic forgetting) and plasticity (rapidly learn from new experiences; quick generalization). This balance provides a natural objective for CL: *balance forgetting and generalization.*

Traditional CL methods either minimize catastrophic forgetting or improve quick generalization but do not model both. For example, common solutions to the catastrophic forgetting issue include (1) representation-driven approaches [49, 25], (2) regularization approaches [27, 2, 34, 16, 48, 48, 24, 37, 10, 43], and (3) memory/experience replay [31, 32, 11, 12, 17]. Similarly, solutions to the generalization problem include representation-learning approaches (matching nets [45], prototypical networks [42], and meta-learning approaches [18, 19, 8, 47]). More recently, several approaches [35, 16, 46, 23, 48, 15] have been introduced that combine methods designed for quick generalization with frameworks designed to minimize forgetting. Traditional CL approaches naively minimize a loss function (combination of forgetting and generalization loss) but do not explicitly account for the trade-off in their optimization setup. The first work to formalize this trade-off was presented in meta-experience replay (MER) [38], where the forgetting-generalization trade-off was posed as a gradient alignment problem. Although MER provides a promising methodology for CL, the balance between forgetting and generalization is enforced with several hyperparameters.

Therefore, two key challenges arise: there is a (1) lack of theoretical tools that study the existence (*under what conditions does a balance point between generalization and forgetting exists?*) and stability (*can this balance be realistically achieved?*) of a balance point; and (2) lack of a systematic approach to achieve the balance point. We address these challenges in this paper.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

We describe a framework where we first formulate CL as a sequential decision-making problem and seek to minimize a cost function summed over the complete lifetime of the model. At any time k, given that the future tasks are not available, the calculation of the cost function becomes intractable. To circumvent this issue, we use Bellman's principle of optimality [4] and recast the CL problem to model the catastrophic forgetting cost on the previous tasks and generalization cost on the new task. We show that equivalent performance on an infinite number of tasks is not practical (Lemma 1 and Corollary 1) and that tasks must be prioritized. To achieve a balance

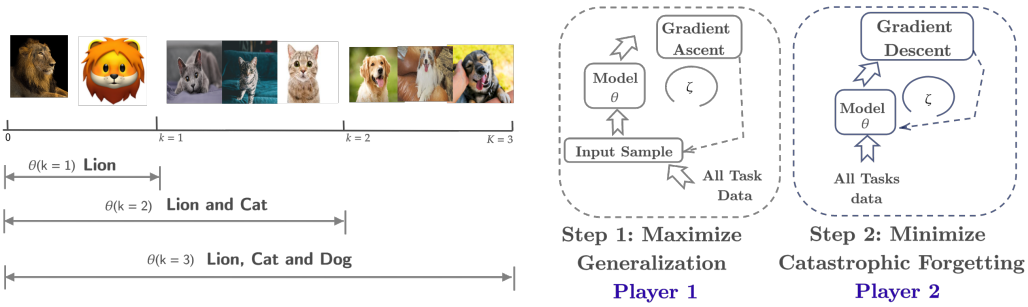

Figure 1: (left) Exemplary CL problem: the lifetime of the model can be split into three intervals. At $k = 1$ we seek to recognize lions; at $k = 2$ we seek to recognize both lions and cats; and at $k = 3$ we seek to recognize cats, lions, and dogs. (right) Illustration of the proposed method: our methodology comprises an interplay between two players. The first player maximizes generalization by simulating maximum discrepancy between two tasks. The second player minimizes forgetting by adapting to maximum discrepancy .

between forgetting and generalization, we pose the trade-off (maximization of generalization and minimization of forgetting) as a saddle point problem where we designate one player for maximizing the generalization cost (player 1) and another for minimizing the forgetting cost (player 2). We prove mathematically that there exists at least one saddle point between generalization and forgetting for each new task (Theorem 1). Furthermore, we show that this saddle point can be attained asymptotically (Theorem 2) when player strategies are chosen as gradient ascent-descent. We then introduce balanced continual learning (BCL), a new algorithm to achieve this saddle point. In our algorithm (see Fig. 1 for a description of BCL), the generalization cost is computed by training and evaluating the model on given new task data. The catastrophic forgetting cost is computed by evaluating the model on the task memory (previous tasks). We first maximize the generalization cost and then minimize catastrophic forgetting costs to achieve the balance. We compare our approach with other methods such as elastic weight consolidation (EWC) [27], online EWC [40], and MER [38] on continual learning benchmark data sets [21] to show that BCL is better than or comparable to the state-of-the-art methods. The contributions of this paper are (1) a theoretical framework to study the CL problem, (2) BCL-a method to attain balance between forgetting and generalization, and (3) advancement of the state of the art in CL.

2 Problem Formulation

We use $\mathbb{R}$ to denote the set of real numbers and $\mathbb{N}$ to denote the set of natural numbers. We use $\|.\|$ to denote the Euclidean norm for vectors and the Frobenius norm for matrices, while using bold symbols to illustrate matrices and vectors. We define an interval $[0, K], K \in \mathbb{N}$ and let $p(\boldsymbol{\mathcal{Q}})$ be the distribution over all the tasks observed in this interval. We define a parametric model $g(.)$ with $\boldsymbol{y}_k = g(\boldsymbol{x}_k; \boldsymbol{\theta}_k)$, where $\boldsymbol{\theta}_k$ is a vector comprised of all parameters of the model with $\boldsymbol{x}_k \in \boldsymbol{\mathcal{X}}_k$. Let n be the number of samples and m be the number of dimensions. Suppose a task at $k : k \in [0, K]$ is denoted as $\boldsymbol{\mathcal{Q}}_k : \boldsymbol{\mathcal{Q}}_k \sim p(\boldsymbol{\mathcal{Q}})$, where $\boldsymbol{\mathcal{Q}}_k = \{\boldsymbol{\mathcal{X}}_k, \ell_k\}$ is a tuple with $\boldsymbol{\mathcal{X}}_k \in \mathbb{R}^{n \times m}$ being the input data. Furthermore, ℓ_k quantifies the loss incurred by $\boldsymbol{\mathcal{X}}_k$ using the model g for each task k. We denote the sequence of parameters $\boldsymbol{\theta}_k$ as $\boldsymbol{u}_{k:K} = \{\boldsymbol{\theta}_\tau \in \Omega_\theta, k \leq \tau \leq K\}$, with Ω_θ being the compact (feasible) set. We denote the optimal value with a superscript $(*)$; for instance, we use $\boldsymbol{\theta}_k^{(*)}$ to denote the optimal value of $\boldsymbol{\theta}_k$ at task k. In this paper, we will use balance point, equillibrium point and saddle point to refer to the point of balance between generalization and forgetting. We will interchange between these terms whenever convenient for the discussion.

An exemplary CL problem is described in Fig. 1 where we address a total of $K = 3$ tasks. To particularize the idea in Fig. 1, we define the cost (combination of catastrophic cost and generalization cost) at any instant k as $J_k(\boldsymbol{\theta}_k) = \gamma_k \ell_k + \sum_{\tau=0}^{k-1} \gamma_\tau \ell_\tau$, where ℓ_τ is computed on task $\boldsymbol{\mathcal{Q}}_\tau$ with γ_τ describing the contribution of $\boldsymbol{\mathcal{Q}}_\tau$ to this sum. To solve the problem at k, we seek $\boldsymbol{\theta}_k$ to minimize $J_k(\boldsymbol{\theta}_k)$. Similarly, to solve the problem in the complete interval $[0, K]$, we seek a $\boldsymbol{\theta}_k$ to minimize $J_k(\boldsymbol{\theta}_k)$ for each $k \in [0, K]$. In other words we seek to obtain $\boldsymbol{\theta}_k$ for each task such that the cost $J_k(\boldsymbol{\theta}_k)$ is minimized. Therefore, the optimization problem for the overall CL problem (overarching goal of CL) is provided as the minimization of the cumulative cost $V_k(\boldsymbol{u}_{k:K}) = \sum_{\tau=k}^{K} \beta_\tau J_\tau(\boldsymbol{\theta}_\tau)$ such that $V_k^{(*)}$, the minima and the optimal cost, is given as

$$V_k^{(*)} = min_{\boldsymbol{u}_{k:K}} V_k(\boldsymbol{u}_{k:K}), \tag{1}$$

with $0 \leq \beta_\tau \leq 1$ being the contribution of J_τ, and $\boldsymbol{u}_{k:K}$ being a weight sequence of length $K - k$.

Within this formulation, two parameters determine the contributions of tasks: γ_τ – contribution of each task in the past and β_τ – contribution of tasks in the future. To successfully solve the optimization problem, $V_k(\boldsymbol{u}_{k:K})$ must be bounded and differentiable, typically ensured by the choice of γ_τ, β_τ. Lemma 1 (full statement and proof in Appendix A) proves that *equivalent performance cannot be guaranteed for an infinite number of tasks*. Furthermore, Corollary 1 (full statement and proof in Appendix A) demonstrates that *if the task contributions are prioritized, the differentiability and boundedness of $J_\tau(\boldsymbol{\theta}_\tau)$ can be ensured*. A similar result was proved in [28], where a CL problem with infinite memory is shown to be NP-hard from a set theoretic perspective. These results (both ours and in [28]) demonstrate that a CL methodology cannot provide perfect performance on a large number of tasks and that tasks must be prioritized.

Despite these invaluable insights, the data corresponding to future tasks (interval $[k, K]$) is not available, and therefore $V_k(\boldsymbol{u}_{k:K})$ cannot be evaluated. The optimization problem in Eq. 1 naively minimizes the cost (due to both previous tasks and new tasks) and does not provide any explicit modeling of the trade-off between forgetting and generalization. Third, $\boldsymbol{u}_{k:K}$, the solution to Eq. 1 is a sequence of parameters, and it is not feasible to maintain $\boldsymbol{u}_{k:K}$ for a large number of tasks. Because of these three issues, the problem is intractable in its current form.

We will first recast the problem using tools from dynamic programming [29], specifically Bellman's principle of optimality, and derive a difference equation that summarizes the complete dynamics for the CL problem. Then, we will formulate a two-player differential game where we seek a saddle point solution to balance generalization and forgetting.

3 Dynamics of Continual Learning

Let $V_k^{(*)} = min_{\boldsymbol{u}_{k:K}} \sum_{\tau=k}^{K} \beta_\tau J_\tau(\boldsymbol{\theta}_\tau)$; the dynamics of CL (the behavior of optimal cost with respect to k) is provided as

$$\nabla_k(V_k^{(*)}) = -min_{\boldsymbol{\theta}_k \in \Omega_\theta} \big[\beta_k J_k(\boldsymbol{\theta}_k) + \big(\langle \nabla_{\boldsymbol{\theta}_k} V_k^{(*)}, \Delta\boldsymbol{\theta}_k\rangle + \langle \nabla_{\boldsymbol{x}_k} V_k^{(*)}, \Delta\boldsymbol{x}_k\rangle\big)\big]. \tag{2}$$

The derivation is presented in Appendix A (refer to Proposition 1). Note that $V_k^{(*)}$ is the minima for the overarching CL problem and $\nabla_k(V_k^{(*)})$ represents all perturbations in $V_k^{(*)}$. Moreover, zero perturbations ($\nabla_k(V_k^{(*)}) = 0$) at k imply that the introduction of a new task at k does not impact our current solution, i.e, solution optimal on all the previous tasks is optimal on the new task as well. Therefore, smaller the perturbations, the better the performance of a model on all tasks which provides our main objective: minimize the perturbations ($\nabla_k(V_k^{(*)})$). In Eq. 2, $\nabla_k(V_k^{(*)})$ is quantified by three terms: the cost contribution from all the previous tasks and the new task $J_k(\boldsymbol{\theta}_k)$; the change in the optimal cost due to the change in the parameters $\langle \nabla_{\boldsymbol{\theta}_k} V_k^{(*)}, \Delta\boldsymbol{\theta}_k\rangle$; and the change in the optimal cost due to change in the input (introduction of new task) $\langle \nabla_{\boldsymbol{x}_k} V_k^{(*)}, \Delta\boldsymbol{x}_k\rangle$.

The first issue with the cumulative CL problem (Eq. (1)) can be attributed to the need for information from the future. In Eq. (2), all information from the future is approximated by using the data from the new and the previous tasks. Therefore, the solution of the CL problem can directly be obtained by solving Eq. (2) using all the available data. Thus, $min_{\boldsymbol{\theta}_k \in \Omega} \big[H(\Delta\boldsymbol{x}_k, \boldsymbol{\theta}_k)\big]$ yields $\nabla_k(V_k^{(*)}) \approx 0$

for $\beta > 0$, with $H(\Delta \boldsymbol{x}_k, \boldsymbol{\theta}_k) = \beta_k J_k(\boldsymbol{\theta}_k) + \langle \nabla_{\boldsymbol{\theta}_k} V_k^{(*)}, \Delta \boldsymbol{\theta}_k \rangle + \langle \nabla_{\boldsymbol{x}_k} V_k^{(*)}, \Delta \boldsymbol{x}_k \rangle$. Essentially, minimizing $H(\Delta \boldsymbol{x}_k, \boldsymbol{\theta}_k)$ would minimize the perturbations introduced by any new task.

Next, we must account for the balance between generalization and forgetting at each task k. A model exhibits generalization when it successfully adapts to a new task. The degree of generalization depends on the discrepancy between the previous tasks and the new task. The larger the discrepancy between two subsequent tasks, the more the model must generalize. Therefore, worst-case discrepancy prompts maximum generalization. However, large discrepancy increases forgetting, and the worst-case discrepancy yields maximum forgetting. Maximum generalization can be obtained by maximizing the third term in Eq. (2) with respect to $\Delta \boldsymbol{x}_k$ since $\Delta \boldsymbol{x}_k$ quantifies the change in the subsequent tasks. However, $\Delta \boldsymbol{x}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$, and $\boldsymbol{x}_{k+1}$ is unknown at k. Therefore, we simulate the worst-case discrepancy by iteratively updating $\Delta \boldsymbol{x}_k$ through gradient ascent, thus maximizing generalization. Next, we minimize forgetting under maximum generalization by iteratively updating $\boldsymbol{\theta}_k$ through gradient descent.

To formalize our idea, let us indicate the iteration index at k by i and write $\Delta \boldsymbol{x}_k$ as $\Delta \boldsymbol{x}_k^{(i)}$ and $\boldsymbol{\theta}_k$ as $\boldsymbol{\theta}_k^{(i)}$ with $H(\Delta \boldsymbol{x}_k, \boldsymbol{\theta}_k)$ as $H(\Delta \boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}_k^{(i)})$ (for simplicity of notations, we will denote $H(\Delta \boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}_k^{(i)})$ as H whenever convenient). Next, we write

$$
\begin{aligned}
&\min_{\boldsymbol{\theta}_k^{(i)} \in \Omega} \left[H(\Delta \boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}_k^{(i)}) \right] = \min_{\boldsymbol{\theta}_k^{(i)} \in \Omega} \left[\beta_k J_k(\boldsymbol{\theta}_k^{(i)}) + \langle \nabla_{\boldsymbol{\theta}_k^{(i)}} V_k^{(*)}, \Delta \boldsymbol{\theta}_k^{(i)} \rangle + \langle \nabla_{\boldsymbol{x}_k^{(i)}} V_k^{(*)}, \Delta \boldsymbol{x}_k^{(i)} \rangle \right] \\
&\leq \min_{\boldsymbol{\theta}_k^{(i)} \in \Omega} \left[\beta_k J_k(\boldsymbol{\theta}_k^{(i)}) + \langle \nabla_{\boldsymbol{\theta}_k^{(i)}} V_k^{(*)}, \Delta \boldsymbol{\theta}_k^{(i)} \rangle + \max_{\Delta \boldsymbol{x}_k^{(i)} \in p(\boldsymbol{\mathcal{Q}})} \langle \nabla_{\boldsymbol{x}_k^{(i)}} V_k^{(*)}, \Delta \boldsymbol{x}_k^{(i)} \rangle \right] \\
&\leq \min_{\boldsymbol{\theta}_k^{(i)} \in \Omega} \max_{\Delta \boldsymbol{x}_k^{(i)} \in p(\boldsymbol{\mathcal{Q}})} \left[H(\Delta \boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}_k^{(i)}) \right].
\end{aligned}
\tag{3}
$$

In Eq. (3), we seek the solution pair $(\Delta \boldsymbol{x}_k^{(*)}, \boldsymbol{\theta}_k^{(*)})$, where $\Delta \boldsymbol{x}_k^{(*)}$ maximizes H (maximizing player, player 1) while $\boldsymbol{\theta}_k^{(*)}$ minimizes H (minimizing player, player 2). The solution is attained when $(\Delta \boldsymbol{x}_k^{(*)}, \boldsymbol{\theta}_k^{(*)})$ is found to satisfy

$$
H(\Delta \boldsymbol{x}_k^{(*)}, \boldsymbol{\theta}_k^{(i)}) \geq H(\Delta \boldsymbol{x}_k^{(*)}, \boldsymbol{\theta}_k^{(*)}) \geq H(\Delta \boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}_k^{(*)}) \tag{4}
$$

and $(\Delta \boldsymbol{x}_k^{(*)}, \boldsymbol{\theta}_k^{(*)})$, is said to be the equilibrium point.

3.1 Theoretical Analysis

With our formulation, two key questions arise: Does our problem setup have an equilibrium point satisfying Eq. (4)? and How can one attain this equilibrium point? We answer these questions with Theorems 1 and 2, respectively. Full statements and proofs are provided in Appendix A.

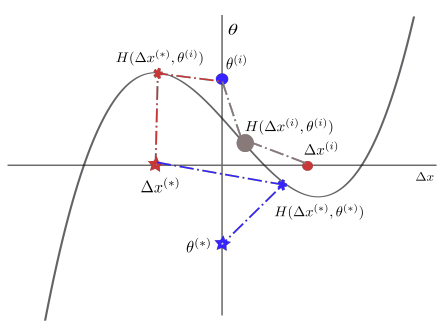

Figure 2: Illustration of proofs. $\Delta \boldsymbol{x}$ (player 1) is the horizontal axis and the vertical axis indicates $\boldsymbol{\theta}$ (player 2) where the curve indicates H. If we start from red circle for player 1 (player 2 is fixed at the blue circle) H is increasing (goes from a grey circle to a red asterisk) with player 1 reaching the red star. Next, start from the blue circle ($\boldsymbol{\theta}$ is at the red star), the cost decreases.

To illustrate the theory, we refer to Fig. 2, where the initial values for the two players are characterized by the pair $\{\boldsymbol{\theta}_k^{(i)}$(blue circle), $\Delta \boldsymbol{x}_k^{(i)}$(red circle)$\}$, and the cost value at $\{\boldsymbol{\theta}_k^{(i)}, \Delta \boldsymbol{x}_k^{(i)}\}$ is indicated by $H(\Delta \boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}_k^{(i)})$ (the grey circle on the cost curve (the dark blue curve)). Our proofing strategy is as follows. First, we fix $\boldsymbol{\theta}_k^{(\cdot)} \in \Omega_\theta$ and construct a neighborhood $\mathcal{M}_k = \{\boldsymbol{\theta}_k^{(\cdot)}, \Omega_x\}$. Within this neighborhood we prove in Lemma 2 and 4 that if we search for $\Delta \boldsymbol{x}_k^{(i)}$ through gradient ascent, we can converge to a local maximizer and H is maximizing with respect to $\Delta \boldsymbol{x}_k^{(i)}$. Second, we let $\Delta \boldsymbol{x}_k^{(\cdot)} \in \Omega_x$ be fixed, and search for $\boldsymbol{\theta}_k^{(i)}$ through gradient descent.

Under this condition, we demonstrate two ideas in Lemma 3 and 5: (1) we show that H is minimizing in the neighborhood $\mathcal{N}_k : \mathcal{N}_k = \{\Omega_\theta, \Delta\boldsymbol{x}_k^{(\cdot)}\}$. (2) We converge to the local minimizer in the neighborhood $\mathcal{N}_k$. Third, in the union of the two neighborhoods $\mathcal{M}_k \cup \mathcal{N}_k$, (proven to be nonempty according to Lemma 6), we show that there exists at least one local equilibrium point (Theorem 1); that is, there is at least one balance point.

Theorem 1 (Existence of an Equilibrium Point)**.** *Consider $(\Delta\boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}_k^{(i)})$ as the initial parameter pair at k. Let us define $\mathcal{M}_k = \{\Omega_x, \boldsymbol{\theta}_k^{(\cdot)}\}$ for $\boldsymbol{\theta}_k^{(\cdot)} \in \Omega_\theta$ and $\mathcal{N}_k = \{\Delta\boldsymbol{x}_k^{(\cdot)}, \Omega_\theta\}$ for $\Delta\boldsymbol{x}_k^{(\cdot)} \in \Omega_x$. Let $(\boldsymbol{\theta}_k^{(\cdot)}, \Delta\boldsymbol{x}_k^{(*)})$ be the local maximizer for H in $\mathcal{M}_k$ and $(\boldsymbol{\theta}_k^{(*)}, \Delta\boldsymbol{x}_k^{(\cdot)})$ be the local minimizer for H in $\mathcal{N}_k$ with H defined through Proposition 2. Given that Lemmas 4 and 5 are true with a nonempty $\mathcal{M}_k \cup \mathcal{N}_k$ (by Lemma 6), it follows that there exists at least one local equilibrium point for H.*

We next show that this equilibrium point is stable (Theorem 2) under a sequential play. Specifically, we show that when player 1 plays first and player 2 plays second, we asymptotically reach a saddle point pair $(\Delta\boldsymbol{x}_k^{(*)}, \boldsymbol{\theta}_k^{(*)})$ for H. At this saddle point, both players have no incentive to move, and the game converges.

Theorem 2 (Stability of the Equilibrium Point)**.** *Let $(\Delta\boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}_k^{(i)})$ be defined at k with $\mathcal{M}_k = \{\Omega_x, \boldsymbol{\theta}_k^{(\cdot)}\}$ for $\boldsymbol{\theta}_k^{(\cdot)} \in \Omega_\theta$ and $\mathcal{N}_k = \{\Delta\boldsymbol{x}_k^{(\cdot)}, \Omega_\theta\}$ for $\Delta\boldsymbol{x}_k^{(\cdot)} \in \Omega_x$. Let $(\Delta\boldsymbol{x}_k^{(*)}, \boldsymbol{\theta}_k^{(\cdot)})$ be the local maximizer for H in $\mathcal{M}_k$ and $(\Delta\boldsymbol{x}_k^{(\cdot)}, \boldsymbol{\theta}_k^{(*)})$ be the local minimizer for H in $\mathcal{N}_k$ with H defined through Proposition 2. Let $\Delta\boldsymbol{x}_k^{(i)} = \alpha_k^{(i)} \times (\nabla_{\Delta\boldsymbol{x}_k^{(i)}} H(\Delta\boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}_k^{(\cdot)}) / \|\nabla_{\Delta\boldsymbol{x}_k^{(i)}} H(\Delta\boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}_k^{(\cdot)})\|^2)$ with the strategy for $\Delta\boldsymbol{\theta}_k^{(i)} = -\alpha_k^{(i)} \times \nabla_{\boldsymbol{\theta}_k^{(i)}} H(\Delta\boldsymbol{x}_k^{(\cdot)}, \boldsymbol{\theta}_k^{(i)})$. Given that Lemmas 2 and 3 are true with nonempty $\mathcal{M}_k \cup \mathcal{N}_k$ (by Lemma 6), then $(\Delta\boldsymbol{x}_k^{(*)}, \boldsymbol{\theta}_k^{(*)})$ is a stable equilibrium point for $H(\Delta\boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}_k^{(i)})$ when $\Delta\boldsymbol{x}_k^{(i)}$ plays first and $\boldsymbol{\theta}_k^{(*)}$ plays second.*

In this game, the interplay between these two opposing players (representative of generalization and forgetting, respectively) introduces dynamics required to play the game. Although we utilize gradient descent-ascent strategies to provide this interplay, one can design alternative strategies for this game.

The results presented in this section are local to the task. In other words, we prove that we can achieve a balance between generalization and forgetting for each task k (neighborhoods are task dependent, and we achieve a local solution given a task k). Furthermore, our game is sequential; that is, there is a leader (player 1) and a follower (player 2). The leader $(\Delta\boldsymbol{x}_k^{(i)})$ plays first, and the follower $(\boldsymbol{\theta}_k^{(i)})$ plays second with complete knowledge of the leader's play. The game is directed by $\Delta\boldsymbol{x}_k^{(i)}$, and any changes in the task (reflected in $\Delta\boldsymbol{x}_k^{(i)}$) will shift the input and thus the equilibrium point. Consequently, the equilibrium point varies with respect to a task, and one will need to attain a new equilibrium point for each shift in a task. Without complete knowledge of the tasks (not available in a CL scenario), only a local result is possible. This highlights one of the key limitations of this work. Ideally, we would like a balance between forgetting and generalization that is independent of tasks. However, this would require learning a trajectory of the equilibrium point (How does the equilibrium point change with change in the tasks?) and is beyond the scope of this paper. One work that attempts to do this is [39] where the authors learn a parameter per task. However, for a large number of tasks such an approach is computationally prohibitive.

These results are valid only under certain assumptions: (1) the Frobenius norm of the gradient is bounded, always positive; (2) the cost function is bounded and differentiable and (3) the learning rate goes to zero as i tends to infinity. The first assumption is reasonable in practice, and gradient clipping or perturbation strategies can be used to ensure it. The boundedness of the cost (second assumption) is ensured by the prioritizing the contributions of the task (Lemma 1 and Corollary 1). The third assumption assumes decaying learning rate. Learning rate decay is a common strategy and is employed widely. Therefore, all the assumptions are practical and reasonable.

3.2 Balanced Continual Learning (BCL)

Equipped with the theory, we develop a new CL method to achieve a balance between forgetting and generalization. By Proposition 2, the cost function is written as $H(\Delta\boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}_k^{(i)}) \approx \beta_k J_k(\boldsymbol{\theta}_k^{(i)}) +$

$(J_k(\boldsymbol{\theta}_k^{(i+\zeta)}) - J_k(\boldsymbol{\theta}_k^{(i)})) + (J_{k+\zeta}(\boldsymbol{\theta}_k^{(i)}) - J_k(\boldsymbol{\theta}_k^{(i)}))$, where $J_{k+\zeta}$ indicates ζ updates on player 1 and $\boldsymbol{\theta}_k^{(i+\zeta)}$ indicates ζ updates on player 2.

The strategies for the two players $\Delta\boldsymbol{x}_k, \boldsymbol{\theta}_k$ are chosen as

$$\underbrace{\frac{\alpha_k^{(i)} \nabla_{\Delta\boldsymbol{x}_k} E[\alpha_k^{(i)} \times H(\Delta\boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}_k^{(i)})]}{\|\nabla_{\Delta\boldsymbol{x}_k} H(\Delta\boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}_k^{(i)})\|^2}}_{\text{Player 1}}, \quad \underbrace{-\alpha_k^{(i)} \times \nabla_{\boldsymbol{\theta}_k} E[H(\Delta\boldsymbol{x}_k^{(*)}, \boldsymbol{\theta}_k^{(i)})]}_{\text{Player 2}}, \tag{5}$$

where E is the expected value operator. We can approximate the required terms in our update rule (player strategies) using data samples (batches). Note that the approximation is performed largely through one-sided finite difference, which may introduce an error and is another potential drawback.

Algorithm 1: BCL

Initialize $\boldsymbol{\theta}, D_P, D_N$
while $k = 1, 2, 3, ...K$ **do**
  j = 0
  while $i < \rho$ **do**
    Get $\boldsymbol{b}_N \in D_k^N$
    Get $\boldsymbol{b}_P \in D_k^P$
    Get $\boldsymbol{b}_{PN} = \boldsymbol{b}_P \cup \boldsymbol{b}_N$
    Copy $\boldsymbol{b}_{PN}$ into $\boldsymbol{x}_k^{PN}$
    i = 0 **while** $j + 1 <= \zeta$ **do**
      Update $\boldsymbol{x}_k^{PN}$ with $J_k(\boldsymbol{\theta}_k)$ using gradient ascent
      j = j+1
    Calculate $J_{k+\zeta}(\boldsymbol{\theta}_k^{(i)}) - J_k(\boldsymbol{\theta}_k^{(i)})$
    Copy $\boldsymbol{\theta}_k^{(i)}$ into $\boldsymbol{\theta}_k^B$
    j = 0 **while** $j + 1 <= \zeta$ **do**
      Update $\boldsymbol{\theta}_k^B$ with $J_k(\boldsymbol{\theta}_k^B)$
      j = j+1
    Calculate $(J_k(\boldsymbol{\theta}_k^B) - J_k(\boldsymbol{\theta}_k^{(i)}))$
    Calculate $H(\Delta\boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}_k^{(i)})$
    Update $\boldsymbol{\theta}_k^{(i)}$ using gradient descent
  i = i+1
Update D_P with D_N

The pseudo code of the BCL is shown in Algorithm. 1. We define a new task array $\boldsymbol{\mathcal{D}}_N(k)$ and a task memory array $\boldsymbol{\mathcal{D}}_P(k) \subset \cup_{\tau=0}^{k-1} \boldsymbol{\mathcal{Q}}_\tau$ (samples from all the previous tasks). For each batch $b_N \in \boldsymbol{\mathcal{D}}_N(k)$, we sample b_P from $\boldsymbol{\mathcal{D}}_P(k)$, combine to create $b_{PN}(k) = b_P(k) \cup b_N(k)$, and perform a sequential play. Specifically, for each task the first player initializes $x_k^{PN} = b_{PN}(k)$ and performs ζ updates on x_k^{PN} through gradient ascent. The second player, with complete knowledge of the first player's strategy, chooses the best play to reduce $H(\Delta\boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}_k^{(i)})$. To estimate player 2's play, we must estimate different terms in $H(\Delta\boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}_k^{(i)})$. This procedure: involves three steps. (1) We use the first player's play and approximate $(J_{k+\zeta}(\boldsymbol{\theta}_k^{(i)}) - J_k(\boldsymbol{\theta}_k^{(i)}))$; (2) To approximate $(J_k(\boldsymbol{\theta}_k^{(i+\zeta)}) - J_k(\boldsymbol{\theta}_k^{(i)}))$: (a), we copy $\hat{\boldsymbol{\theta}}$ into $\hat{\boldsymbol{\theta}}_B$ (a temporary network) and perform ζ updates on $\hat{\boldsymbol{\theta}}_B$; (b) we compute $J_k(\boldsymbol{\theta}_k^{(i+\zeta)})$ using $\hat{\boldsymbol{\theta}}_B(k+\zeta)$ and evaluate $(J_k(\boldsymbol{\theta}_k^{(i+\zeta)}) - J_k(\boldsymbol{\theta}_k^{(i)}))$; (3) Equipped with these approximations, we compute $H(\Delta\boldsymbol{x}_k^{(i)}, \boldsymbol{\theta}_k^{(i)})$ and obtain the play for the second player. Both these players perform the steps repetitively for each piece of information (batch of data). Once all the data from the new task is exhausted, we move to the next task.

3.3 Related work

Traditional solutions to the CL focus on either the forgetting issue [39, 49, 47, 7, 27, 50, 2, 31, 32, 11] or the generalization issue [45, 42, 18, 19, 8]. Common solution to the forgetting problem involve dynamic architectures and flexible knowledge representation such as [39, 49, 47, 7], regularization approaches including [27, 50, 2] and memory/experience replay involving [31, 32, 11]. Similarly, quick generalization to new task has been addressed through few-shot and one-shot learning approaches such as matching nets [45] and prototypical network [42]. More recently, the field of meta learning has approached the generalization problem by designing a meta learner that can perform quick generalization from very little data [18, 19, 8].

In the past few years, metalearners for quick generalization have been combined with methodologies specifically designed for reduced forgetting [22, 3]. For instance, the approaches in [22, 3] adapt the model-agnostic meta learning (MAML) framework in [18] with robust representation to minimize forgetting and provide impressive results on CL. However, both these approaches require a pretraining phase for learning representation. Simultaneously, Gupta et. al [20] introduced LA-MAML-a metalearning approach where the impact of learning rates on the CL problem is reduced through the use of per-parameter learning rates. LA-MAML [20] also introduced episodic memory to address the forgetting issue. Other approaches also have attempted to model both generalization and forgetting. In [16], the gradients from new tasks are projected onto a subspace that is orthogonal to the older

tasks, and forgetting is minimized. Similarly, Joseph and Balasubramanian [23] utilized a Bayesian framework to consolidate learning across previous and current tasks while [48] provided a framework to approximate loss function to summarize the forgetting in the CL setting. Furthermore, [1] was focused towards sampling episodes in the reinforcement learning setting and [15] introduced a generative adversarial network (GAN) type structure to progressively learn shared features assisting reduced forgetting and improved generalization. Despite significant progress, these methods [22, 3, 20, 16, 23, 48, 1, 15] are still inherently tilted toward maximizing generalization or minimizing forgetting because they naively minimize the loss function. Therefore, the contribution of different terms in the loss function become important, for instance, if generalization cost is given more weight, a method would generalize better. Similarly, if forgetting cost is given more weight a method would forget less. Therefore, the resolution of the trade-off inherently depends on an hyperparameter.

The first work to formalize the trade-off in CL was MER where the trade-off was formalized as a gradient alignment problem. Similar to MER, Doan et. al also ([13]) attempted to study forgetting as an alignment problem. In MER, the angle between the gradients was approximated by using Reptile [36], which promotes gradient alignment by reducing weight changes. On the other hand, [13] formalizes the alignment as an eigenvalue problem and introduce a PCA-driven method to ensure alignment. Our approach models this balance as a saddle point problem such that the saddle point (balance point or equilibrium point) resolves the trade-off.

Our approach is the first in the CL literature to prove the existence of the saddle point (the balance point) between generalization and forgetting given a task in a CL problem. Furthermore, we are also the first to theoretically demonstrate that the saddle point can be achieved reasonably under a gradient ascent-descent game. The work closest to ours is [15], where an adversarial framework is described to minimize forgetting in CL by generating task-invariant representation. However, [15] is not model agnostic (the architecture of the network is important) and requires a considerable amount of data at the start of the learning procedure. Because of these issues, [15] is not suitable for learning in the sequential scenario.

Several attempts have been made in the literature to theoretically analyze different aspects of CL. For instance, [6] attempts to unify regularization methods such as [27, 50, 2, 31]. On the other hand, [5] provides generalization (in the context of training to test data generalization) guarantees with orthogonal gradient descent while [48] provides generalization (in the context of training to test data generalization) analysis while regularizing with the second order approximation of the loss functions. Although these works [6, 13, 5, 48] provide important results, they are focused towards quantifying catastrophic forgetting or generalization but do attempt to model the trade-off in any manner.

4 Experiments

We use the CL benchmark [21] for our experiments and retain the experimental settings (hyperparameters) from [21, 44]. For comparison, we use the split-MNIST, permuted-MNIST and split-CiFAR100 data set while considering three scenarios: incremental domain learning (IDL), incremental task learning (ITL), and incremental class learning (ICL). The splitting and permutation strategies when applied on the MNIST or CiFAR100 data set can generate task sequences for all three scenarios (illustrated in Figure 1 and Appendix: Figure 2 of [21]). For comparing our approach, we use three baseline strategies: standard neural network with Adam [26], Adagrad [14], and SGD, and L_2-regularization and naive rehearsal (which is similar to experience replay). For CL approaches, we use EWC [27], online EWC [40], SI [50], LwF [30], DGR [41], RtF [44], MAS [2], MER [38], and GEM [33]. We utilize data pre-processing as provided by [21]. Additional details on experiments can be found in Appendix B and [21, 44]. Furthermore, all experiments are conducted in Python 3.4 using the pytorch 1.7.1 library with the NVIDIA-A100 GPU for our simulations.

Comparison with the state of the art: The results for our method are summarized in Table 1 and 2. The efficiency for any method is calculated by observing the average accuracy (retained accuracy (RA) [38]) at the end of each repetition and then evaluating the mean and standard deviation of RA across different repetitions. For each method, we report the mean and standard deviation of RA over five repetitions of each experiment. In each column, we indicate the best-performing method in bold. For the split-MNIST data set, we obtain 99.52 ± 0.07 for ITL, 98.71 ± 0.06 for IDL, and 97.32 ± 0.17 for ICL. Similarly, with the permuted-MNIST data set, we obtain 97.41 ± 0.01 for ITL, 97.51 ± 0.05 for IDL, and 97.61 ± 0.01 for ICL. Furthermore, with the split-CiFAR100 data

Table 1: Performance of our approach in comparison with other methods in the literature. We record the mean and standard deviation of the retained accuracy for different methods. The best scores are in bold.

Method	split-MNIST			permuted-MNIST		
	Incremental task learning [ITL]	Incremental domain learning [IDL]	Incremental class learning [ICL]	Incremental task learning [ITL]	Incremental domain learning [IDL]	Incremental class learning [ICL]
Adam	95.52 ± 2.14	54.75 ± 2.06	19.72 ± 0.03	93.42 ± 0.56	77.87 ± 1.27	14.02 ± 1.25
SGD	97.65 ± 0.28	62.80 ± 0.34	19.36 ± 0.02	90.95 ± 0.20	78.17 ± 1.16	12.82 ± 0.95
Adagrad	98.37 ± 0.29	57.59 ± 2.54	19.59 ± 0.17	92.45 ± 0.16	91.59 ± 0.46	29.09 ± 1.48
L_2	97.62 ± 0.69	$66.84 \pm 3.91.$	22.92 ± 1.90	94.87 ± 0.38	92.81 ± 0.32	13.92 ± 1.79
Naive rehearsal	99.32 ± 0.10	94.85 ± 0.80	90.88 ± 0.70	96.23 ± 0.04	95.84 ± 0.06	96.25 ± 0.10
Naive rehearsal-C	99.41 ± 0.04	97.13 ± 0.37	94.92 ± 0.63	97.13 ± 0.03	96.75 ± 0.03	97.24 ± 0.05
EWC	96.59 ± 0.99	57.31 ± 1.07	19.70 ± 0.14	95.38 ± 0.33	89.54 ± 0.52	26.32 ± 4.32
Online EWC	99.01 ± 0.12	58.25 ± 1.23	19.68 ± 0.05	95.15 ± 0.49	93.47 ± 0.01	42.58 ± 6.50
SI	99.10 ± 0.16	64.63 ± 1.67	19.67 ± 0.25	94.35 ± 0.51	91.12 ± 0.93	58.52 ± 4.20
MAS	98.88 ± 0.14	61.98 ± 7.17	19.70 ± 0.34	94.74 ± 0.52	93.22 ± 0.80	50.81 ± 2.92
GEM	98.32 ± 0.08	97.37 ± 0.22	93.04 ± 0.05	95.44 ± 0.96	96.86 ± 0.02	96.72 ± 0.03
DGR	99.47 ± 0.03	95.74 ± 0.23	91.24 ± 0.33	92.52 ± 0.08	95.09 ± 0.04	92.19 ± 0.09
RtF	$\mathbf{99.66 \pm 0.03}$	97.31 ± 0.11	92.56 ± 0.21	97.31 ± 0.01	97.06 ± 0.02	96.23 ± 0.04
MER	97.12 ± 0.10	92.16 ± 0.35	93.20 ± 0.12	97.15 ± 0.08	96.11 ± 0.31	91.71 ± 0.03
BCL (With Game)	99.52 ± 0.07	$\mathbf{98.71 \pm 0.06}$	$\mathbf{97.32 \pm 0.17}$	$\mathbf{97.41 \pm 0.01}$	$\mathbf{97.51 \pm 0.05}$	$\mathbf{97.61 \pm 0.01}$
BCL (Without Game)	97.73 ± 0.03	96.43 ± 0.29	91.88 ± 0.55	96.16 ± 0.03	96.08 ± 0.06	95.96 ± 0.06

set, we obtain 81.82 ± 0.17 for ITL, 62.11 ± 0.00 for IDL, and 69.27 ± 0.03 for ICL. BCL is the best-performing methodology for all cases (across both data sets) except RtF for ITL (0.14% drop) with the split-MNIST data set.

Generally, ITL is the easiest learning scenario [21], and all methods therefore perform well on ITL (the performance is close). For the ITL scenario with the split-MNIST data set, BCL is better than most methods; but several methods, such as naive rehearsal, naive rehearsal-C, RtF, and DGR, attain close RA values (less than 1% from BCL). Note that both DGR and RtF involve a generative model pretrained with data from all the tasks. In a sequential learning scenario, one cannot efficiently train generative models because data corresponding to all the tasks is not available beforehand. Although RtF provides improved performance for split-MNIST (ITL), the improvement is less than 1% (not significant). In fact, RtF performance is poorer for BCL in ICL by 4.76% (a significant drop in performance) and in IDL by 1.4%.

Table 2: Performance of BCL for the split-split-CiFAR100 data set. We record the retained accuracy for different methods. We obtained RA scores for all methods except BCL from [21].

Method	split-CiFAR100		
	Incremental task learning	Incremental domain learning	Incremental class learning
Adam	30.53 ± 0.58	19.65 ± 0.14	17.20 ± 0.06
SGD	43.77 ± 1.15	19.17 ± 0.12	17.18 ± 0.12
Adagrad	36.27 ± 0.43	19.06 ± 0.14	15.83 ± 0.20
L_2	51.73 ± 1.30	19.96 ± 0.15	17.12 ± 0.04
Naive rehearsal	70.20 ± 0.17	35.94 ± 0.39	34.33 ± 0.19
Naive rehearsal-C	78.41 ± 0.37	51.81 ± 0.18	51.28 ± 0.17
EWC	61.11 ± 1.43	57.31 ± 1.07	19.70 ± 0.14
Online EWC	63.22 ± 0.97	20.03 ± 0.10	17.16 ± 0.09
SI	64.81 ± 1.00	20.26 ± 0.09	17.26 ± 0.11
MAS	64.77 ± 0.78	19.99 ± 0.16	17.07 ± 0.12
BCL(With Game)	$\mathbf{81.82 \pm 0.17}$	$\mathbf{62.11 \pm 0.00}$	$\mathbf{69.27 \pm 0.03}$
BCL(Without Game)	69.17 ± 0.12	51.82 ± 0.19	52.82 ± 0.01

Two additional observations can be made on the split-MNIST data set. First, Adagrad, SGD, and L2 achieve better performance than does Adam. Therefore, in our analysis Adagrad appears more appropriate although Adam is popularly used for this task. Second, naive rehearsal (both naive rehearsal and naive rehearsal-C approaches) achieve performance equivalent to the state-of-the-art methods with similar memory overhead. Furthermore, naive rehearsal performs much better than online EWC and SI, especially in the ICL scenario. These limitations indicate that regularization-driven approaches are not much better than baseline models and in fact perform poorer to methods involving memory (naive rehearsals, MER, BCL, etc.). In [28], it was shown theoretically that memory-based approaches typically do better than regularization-driven approaches which is empirically observed in this paper too. Another interesting observation is that EWC and online EWC require significant hyperparameter tuning, which would be difficult to do in real-world scenarios. Other regularization-based methods, such as SI and MAS, also suffer from the same issue.

The observations from the split-MNIST carry forward to the permuted-MNIST data set. Moreover, RA values for the permutation MNIST data set are better than the split-MNIST data set across the board, indicating that the permutation MNIST data set presents an easier learning problem. Similar to the observations made with the split-MNIST data set, BCL is better than all methods for the permuted-MNIST dataset with naive rehearsal and RtF providing RA values that are close (less than

1%). The only methodology in the literature that attempts to model the balance between forgetting and generalization is MER, an extension of GEM. From our results, we observe that BCL is better than MER in all cases (Split-MNIST–2.4% improvement for ITL, 6.55% improvement for IDL, 4.12% improvement for ICL and Permuted-MNIST–0.26% improvement for ITL, 1.4% improvement for IDL and 5.9% improvement for ICL).

The substantial improvements obtained by BCL are more evident in Table 2 where the results on the split-CiFAR100 data set are summarized. BCL is clearly the best-performing method. The next best-performing method is naive rehearsal-C where BCL improves performance by 3.41% for ITL, 10.3% for IDL, and 17.99% for ICL. Other observations about the regularization methods and the rest of the baseline methods carry forward from Table1. However, one key difference is that while AdaGrad was observed to be better than Adam with the MNIST data set, Adam was comparable to SGD and Adagrad for split-CiFAR100. In summary, BCL is comparable to or better than the state of the art in the literature for both the MNIST and CiFAR100 data sets.

Do we need a game to achieve this performance? To provide additional insights into the benefits of the game, we compare RA values with and without the game. In our setup, $\Delta\boldsymbol{x}_k^{(i)}$ aims at increasing the cost, and $\boldsymbol{\theta}_k^{(i)}$ aims at reducing the cost. If we hold the play for $\Delta\boldsymbol{x}_k^{(i)}$, then the dynamics required to play the game do not exist, thus providing a method that can perform CL without the game. Therefore, we induce the absence of a game by fixing $\Delta\boldsymbol{x}_k^{(i)}$. We then perform each of the nine experiments for five repetitions (ICL, IDL and ITL for split-MNIST, permuted-MNIST and split-CiFAR100) and summarize results in the last two rows of Table. 1 and 2. We make two observations. First, even without the game, we achieve RA values comparable to the state of the art. This is true for both the MNIST and CiFAR100 data sets. Second, with the introduction of the game, we observe improved RA values across the board (at least by 1% with MNIST). The difference is clearer with the CiFAR100 data set where we observe substantial improvement in RA values (at least 10%). We experimentally observe that by engineering the balance point, performance in the CL scenario is improved.

5 Conclusion

We developed a dynamic programming-based framework to enable the methodical study of key challenges in CL. We show that an inherent trade-off between generalization and forgetting exists and must be modeled for optimal performance. To this end, we introduce a two-player sequential game that models the trade-off. We show in theory that there is an equilibrium point that resolves this trade-off (Theorem 1) and that this saddle point can be attained (Theorem 2). However, we observe that any change in task modifies the equilibrium point. Therefore, a global equilibrium point between generalization and forgetting is not possible, and our results are valid only in a neighborhood (defined given a task). To attain this equilibrium point, we develop BCL and demonstrate state-of-the-art performance on a CL benchmark [21]. In future, we will extend our framework for non-Euclidean tasks such as graphs, point clouds, and manifolds.

6 Broader Impact

Positive Impacts: CL has wide range of applicability. It helps avoids retraining and improve learning efficiency of learning methods. Therefore, in science applications where the data is generated sequentially but the data distribution varies with time, our method provides the potential for improved performance. A theoretically grounded CL approach can provide the performance guarantees that are necessary in hi-fidelity applications where any mistakes on part of the learning algorithm leads at least to monetary loss and at worse to loss of life. *Negative Impacts:* Our theoretical framework does not have the potential for direct adverse impacts. However, the potential advantages of our approach can improve the efficiency of hackers who can potentially adapt to cyber-security mechanism. Furtermore CL can potentially improve the effectiveness of synthetic media such as the deepfake system directly impacting the privacy of common individuals.

References

[1] Elmira Amirloo Abolfathi, Jun Luo, Peyman Yadmellat, and Kasra Rezaee. CoachNet: An adversarial sampling approach for reinforcement learning, 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] Shawn Beaulieu, Lapo Frati, Thomas Miconi, Joel Lehman, Kenneth O Stanley, Jeff Clune, and Nick Cheney. Learning to continually learn. *arXiv preprint arXiv:2002.09571*, 2020.

[4] Richard E Bellman. *Adaptive control processes: a guided tour*. Princeton university press, 2015.

[5] Mehdi Abbana Bennani, Thang Doan, and Masashi Sugiyama. Generalisation guarantees for continual learning with orthogonal gradient descent. *arXiv preprint arXiv:2006.11942*, 2020.

[6] Frederik Benzing. Unifying regularisation methods for continual learning, 2021.

[7] Massimo Caccia, Pau Rodriguez, Oleksiy Ostapenko, Fabrice Normandin, Min Lin, Lucas Caccia, Issam Laradji, Irina Rish, Alexandre Lacoste, David Vazquez, and Laurent Charlin. Online fast adaptation and knowledge accumulation: a new approach to continual learning, 2021.

[8] Massimo Caccia, Pau Rodríguez, Oleksiy Ostapenko, Fabrice Normandin, Min Lin, Lucas Caccia, Issam H. Laradji, Irina Rish, Alexande Lacoste, David Vázquez, and Laurent Charlin. Online fast adaptation and knowledge accumulation: A new approach to continual learning. *ArXiv*, abs/2003.05856, 2020.

[9] Gail A Carpenter and Stephen Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer vision, graphics, and image processing*, 37(1):54–115, 1987.

[10] Arslan Chaudhry, Naeemullah Khan, Puneet K Dokania, and Philip HS Torr. Continual learning in low-rank orthogonal subspaces. *arXiv preprint arXiv:2010.11635*, 2020.

[11] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. Continual learning with tiny episodic memories. *arXiv preprint arXiv:1902.10486*, 2019.

[12] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

[13] 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.

[14] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[15] Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. Adversarial continual learning, 2020.

[16] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning, 2019.

[17] Enrico Fini, Stéphane Lathuilière, Enver Sangineto, Moin Nabi, and Elisa Ricci. Online continual learning under extreme memory constraints. In *European Conference on Computer Vision*, pages 720–735. Springer, 2020.

[18] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[19] Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. *arXiv preprint arXiv:1902.08438*, 2019.

[20] Gunshi Gupta, Karmesh Yadav, and Liam Paull. La-MAML: Look-ahead meta learning for continual learning, 2020.

[21] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. In *NeurIPS Continual learning Workshop*, 2018.

[22] Khurram Javed and Martha White. Meta-learning representations for continual learning. In *Advances in Neural Information Processing Systems*, pages 1818–1828, 2019.

[23] K J Joseph and Vineeth N Balasubramanian. Meta-consolidation for continual learning, 2020.

[24] Sangwon Jung, Hongjoon Ahn, Sungmin Cha, and Taesup Moon. Continual learning with node-importance based adaptive group sparse regularization. *arXiv e-prints*, pages arXiv–2003, 2020.

[25] Zixuan Ke, Bing Liu, and Xingchang Huang. Continual learning of a mixed sequence of similar and dissimilar tasks. *Advances in Neural Information Processing Systems*, 33, 2020.

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[27] 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.

[28] Jeremias Knoblauch, Hisham Husain, and Tom Diethe. Optimal continual learning has perfect memory and is NP-hard. In *International Conference on Machine Learning*, pages 5327–5337. PMLR, 2020.

[29] Frank L Lewis, Draguna Vrabie, and Vassilis L Syrmos. *Optimal control*. John Wiley & Sons, 2012.

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

[31] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3-4):293–321, 1992.

[32] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in neural information processing systems*, pages 6467–6476, 2017.

[33] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *arXiv preprint arXiv:1706.08840*, 2017.

[34] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Razvan Pascanu, and Hassan Ghasemzadeh. Understanding the role of training regimes in continual learning. *arXiv preprint arXiv:2006.06958*, 2020.

[35] Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep online learning via meta-learning: Continual adaptation for model-based RL, 2019.

[36] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms, 2018.

[37] Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard E Turner, and Mohammad Emtiyaz Khan. Continual deep learning by functional regularisation of memorable past. *arXiv preprint arXiv:2004.14070*, 2020.

[38] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

[39] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[40] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4528–4537. PMLR, 2018.

[41] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *arXiv preprint arXiv:1705.08690*, 2017.

[42] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017.

[43] Michalis K Titsias, Jonathan Schwarz, Alexander G de G Matthews, Razvan Pascanu, and Yee Whye Teh. Functional regularisation for continual learning with gaussian processes. *arXiv preprint arXiv:1901.11356*, 2019.

[44] Gido M. van de Ven and Andreas S. Tolias. Generative replay with feedback connections as a general strategy for continual learning, 2019.

[45] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638, 2016.

[46] Risto Vuorio, Dong-Yeon Cho, Daejoong Kim, and Jiwon Kim. Meta continual learning. *CoRR*, abs/1806.06928, 2018.

[47] Huaxiu Yao, Longkai Huang, Ying Wei, Li Tian, Junzhou Huang, and Zhenhui Li. Don't overlook the support set: Towards improving generalization in meta-learning. *arXiv preprint arXiv:2007.13040*, 2020.

[48] Dong Yin, Mehrdad Farajtabar, Ang Li, Nir Levine, and Alex Mott. Optimization and generalization of regularization-based continual learning: a loss approximation viewpoint, 2020.

[49] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

[50] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. *Proceedings of machine learning research*, 70:3987, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Fig 1 (right)
   (b) Did you describe the limitations of your work? [Yes] Refer Section 3.1 and Section 3.2
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Refer Section Broader Impact
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] Yes
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Refer to the theorem statements in section. 3.1
   (b) Did you include complete proofs of all theoretical results? [Yes] Refer to proofs in Appendix A
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Refer to supplementary materials
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Refer to Appendix C in supplementary materials and [21]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Five repetitions and mean and standard deviations are reported. Refer to Table. 1
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Refer to the Results section
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We used the continual learning benchmark by [21]; we cite their paper.
   (b) Did you mention the license of the assets? [Yes] No licence information was provided by [21].

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you are using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]