Provably Safe Model Updates

Leo Elmecker-Plakolm*
Department of Computing
Imperial College London
London, United Kingdom
le24@ic.ac.uk

Pierre Fasterling*
School of Computer and Comm. Sciences (IC)
École Polytechnique Fédérale de Lausanne
Lausanne, Switzerland
pierre.fasterling@epfl.ch

Philip Sosnin*
Department of Computing
Imperial College London
London, United Kingdom
p.sosnin23@imperial.ac.uk

Calvin Tsay
Department of Computing
Imperial College London
London, United Kingdom
c.tsay@imperial.ac.uk

Matthew Wicker
Department of Computing
Imperial College London
London, United Kingdom
m.wicker@imperial.ac.uk

***Abstract*—Safety-critical environments are inherently dynamic. Distribution shifts, emerging vulnerabilities, and evolving requirements demand continuous updates to machine learning models. Yet even benign parameter updates can have unintended consequences, such as catastrophic forgetting in classical models or alignment drift in foundation models. Existing heuristic approaches (e.g., regularization, parameter isolation) can mitigate these effects but cannot certify that updated models continue to satisfy required performance specifications. We address this problem by introducing a framework for provably safe model updates. Our approach first formalizes the problem as computing the largest locally invariant domain (LID): a connected region in parameter space where all points are certified to satisfy a given specification. While exact maximal LID computation is intractable, we show that relaxing the problem to parameterized abstract domains (orthotopes, zonotopes) yields a tractable primal-dual formulation. This enables efficient certification of updates—independent of the data or algorithm used—by projecting them onto the safe domain. Our formulation further allows computation of multiple approximately optimal LIDs, incorporation of regularization-inspired biases, and use of look-ahead data buffers. Across continual learning and foundation model fine-tuning benchmarks, our method matches or exceeds heuristic baselines for avoiding forgetting while providing formal safety guarantees.**

***Index Terms*—verification, continual learning, fine-tuning, safety**

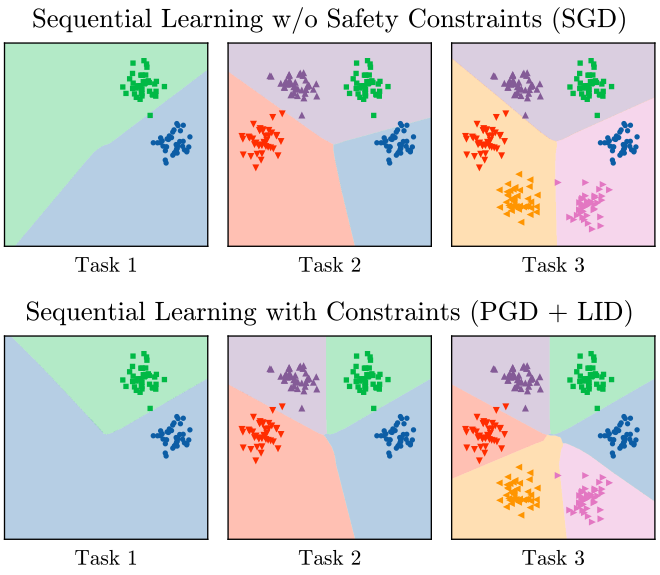

Fig. 1. Illustration of our method on a simple 'blobs' dataset, where the model sequentially learns pairs of classes. The true class label for each data point corresponds to the color of the point and the background region to the output of the model. Top: Standard training forgets previously learned classes. Bottom: Training within our locally invariant domain successfully preserves performance on earlier tasks.

I. Introduction

Machine learning capabilities are evolving at unprecedented speed, driving deployment in high-stakes domains where safety and security are critical requirements [1], [2]. Models used in such critical domains (e.g., autonomous driving, medical applications, and financial services) must continually contend with input distribution shifts, newly discovered vulnerabilities, and evolving regulatory requirements. As a result, regular updates are a prerequisite to safe deployment. However, even benign updates can trigger severe unintended effects on model behavior—most notably catastrophic forgetting and loss of alignment—which are extensively documented in both foundation models [3]–[6] and in continual learning [7], [8]. In absence of *a priori* guarantees, model updates cannot be deployed into safety critical contexts without complete re-evaluation to satisfy regulators [9] or internal auditors [10], [11]. Yet comprehensive re-validation is often impractical due to prohibitive computational cost, latency constraints, or inaccessible training data. This tension between the need for adaptation and the demand for assurance is a central obstacle to deploying learning systems safely in the real world.

Prior works usually seek to combat catastrophic forgetting or preserve model alignment during updates through approaches such as regularization [12], [13], replay buffers [14], [15], or parameter isolation [16], [17]. While these methods display promising results, they do not come with

formal guarantees of avoiding forgetting and therefore cannot avoid the need for revalidation. To address this, we investigate the problem of providing an update framework that has *a priori* guarantees that the updated model satisfies a given performance specification e.g., prior task performance or predictive stability. Beyond eliminating the revalidation bottleneck, we discuss how such a framework may offer certifiable safety evidence for regulators [9], reliable model version control, enforceable terms-of-use for downstream users [18], [19], and safe mechanisms for adapting to emerging vulnerabilities.

To provide safe update guarantees, we introduce the problem of computing a local invariant domain (LID): a connected region of parameter space within which all parameters are provably certified to satisfy a specified performance requirement measured on the prior task. Once a LID has been computed, the safety of an arbitrary update can be efficiently enforced by projecting any proposed parameter change onto the certified LID, and where multiple tasks are observed in sequence one can project onto the intersection of each task's LID to certify safety (discussed in detail in Section III).

The primary technical contribution of this paper is the development of effective and practical LID computation techniques as not all LIDs are practically useful. For example, the LID containing only the initial model parameter vector is trivially valid, but effectively prohibits model updates. Thus, we formulate the general LID problem as finding the *maximal* such domain (by volume) over all connected subsets of the parameter space. Because optimizing over arbitrary connected domains is intractable, we propose a tractable yet conservative relaxation: an optimization problem over parameterized abstract domains, which allows us to leverage advances in bounding neural networks [20], [21] and abstract interpretation [22], [23]. Of course, the optimization problem over parameterized abstract domains must be constrained to only domains for which we can certify that every parameter respects the safety property. Thus, we instantiate a primal-dual formulation of the constrained maximal LID problem enabling the computation of approximately maximal LIDs via alternating gradient-descent-ascent.

Practical implementation of this framework requires careful solutions to several important problems to avoid unsound or trivial LIDs. Firstly, where safety constraints are formulated as properties of an input distribution, one must use a finite-sample approximation of the constraint thus introducing a probability of failure. To address this, we provide concentration bounds that allow us to give worst-case probabilistic guarantees of constraint satisfaction. Secondly, our primal-dual formulation only finds locally optimal solutions to the LID problem. To address this, we propose to use prior work from continual learning to bias our optimization towards better solutions and propose to compute multiple LIDs (LID multiplicity) to avoid problematic local optima. Finally, many model updating scenarios either maintain access to previous task data or have prior information about what future updates might be relevant, e.g., planned support for additional disease distributions in medical applications or planned multi-lingual support for language models. Taking inspiration from the continual learning community, we support such scenarios by introducing a finite-buffer extension to the LID problem.

In Section V, we experimentally demonstrate the effectiveness of our approach across standard continual learning and foundation model fine-tuning tasks. We benchmark the performance of our algorithm against established regularization-based approaches and benchmark the guarantees of our algorithm against the continual learning approach *InterContiNet* (ICN) [24]. An embedded step of the ICN algorithm can be viewed as the computation of a LID, though the authors do not pose the problem as such. Thus, ICN does not consider finite sample error, use of replay buffers, LID multiplicity, or principled constraint encoding which leads to potentially trivial or unsound bounds and numerical instability. Across a wide variety of synthetic and real-world benchmarks including MNIST [25], CIFAR-10 [26], hate speech classification, and multi-lingual adaptation, we find that our approach is able to match or best benchmark performance while providing state-of-the-art guarantees. In summary, this paper makes the following contributions:

- We introduce the *maximum local invariant domain (LID)* problem, and develop a general framework for computing provable guarantees that model updates preserve behavioral specifications such as alignment or task performance.
- We demonstrate how LIDs can be used to certify consecutive tasks and can leverage replay buffers thus leading to a more general approach for certified continual learning.
- We propose a relaxation of the maximum LID via abstract interpretation techniques, enabling an efficient first-order primal-dual algorithm to approximate maximal LIDs.
- We extend our LID computation algorithm using biasing methods inspired by continual learning literature to identifying empirically favorable LID solutions.
- We evaluate our approach across synthetic and real-world benchmarks, including MNIST, CIFAR, and content moderation, demonstrating its ability to provide non-trivial and practically useful safety guarantees for model updates.

II. Related Works

a) Model Updates Eroding Alignment[1]*:* Ensuring alignment of foundation models is a growing area of research [27]–[29]. Despite extensive training-time safeguards, it has been shown that fine-tuning rapidly erodes many safeguards [4], even when users are not malicious [3]. Some heuristics have been developed to defend against this erosion [30]–[32], but have been found unsuccessful in the face of more sophisticated attackers [19], [33]. These studies indicate the start of an arms race between attackers and defenders often observed in adversarial settings, and they point to the importance of formal verification in safety-critical contexts.

[1]We provide an extended discussion of related works in Appendix A.

b) Catastrophic Forgetting in Continual Learning: Ensuring that training updates do not degrade prior model capabilities (as seen in Figure 1), an issue termed catastrophic forgetting [7], [12], [34], has been widely studied within the continual learning community [8]. Avoiding forgetting is often studied under the assumption that data from prior tasks are unavailable [34]. In this case, heuristic forgetting defenses can largely be split into approaches that use parameter isolation (i.e., only updating a subset of parameters [16], [17], [35]) or regularization [7], [12], [13] (i.e., soft constraints limiting changes to important parameters). With access to prior task data for revalidation, replay methods have been developed [14], [15]. Each of these methods has demonstrated some considerable success in reducing forgetting; however, they do not come with concrete, formal guarantees.

c) Formal Methods in Machine Learning: The field of formal methods for machine learning algorithms has produced proofs of a wide range of notions including adversarial robustness [20], [23], explainability [36], privacy [37], and uncertainty [38]. Most related to this paper is the growing literature on formal security guarantees for machine learning models [23], [39] which, however, focus on specific dataset threat models. The most closely related work is *InterContiNet* (ICN) [24], which introduces a continual learning algorithm that uses heuristic optimization to identify safe parameter intervals. The algorithm begins with a large, potentially unsafe parameter interval and iteratively contracts it via SGD to meet an accuracy threshold. In contrast to our approach, ICN seeks only to find *some* valid LID rather than a *maximal* LID, reducing the model's flexibility to learn new tasks while preserving safety. Moreover, its shrinkage-based optimization procedure is susceptible to interval collapse and numerical instability, which hinders scalability to larger models. Finally, the method does not incorporate multiplicity, biasing, replay buffers, or finite-sample safety guarantees – key contributions of this work.

III. Certification for Provably Safe Updates

Throughout the paper we will consider tasks to be supervised learning problem with a fixed sample size N, i.e., $\mathcal{D} := \{(x^{(i)}, y^{(i)}\}_{i=1}^{N}$, where we describe $x \in \mathbb{R}^n$ as our feature vectors and $y \in \mathcal{Y}$ to be our labels; we will assume labels inhabit a c-dimensional space (i.e., c classes or c-dimensional regression). We denote machine learning models as parameterized functions $f^\theta : \mathbb{R}^n \to \mathcal{Y}$ with parameters $\theta \in \mathbb{R}^p$. A proposed update to the parameters comprises a vector $u \in \mathbb{R}^p$, with the result of applying the update being: $\theta' = \theta + u$. We denote a function that proposes model updates (SGD, SFT, DPO, etc.) as an update mechanism $\mathcal{M}$. Finally, we model behavioral properties of interest such as alignment or performance criteria using a function $\phi : \mathbb{R}^{p \times n \times c} \to \mathbb{R}$. Without loss of generality, we use the convention that lower scores of ϕ are better and indicate better performance/alignment, and in our experiments we take ϕ to be one minus the accuracy on a particular task, but this is left purposefully generic with further discussion of concrete choices in Appendix C. We now provide the general problem statement for a provably safe update mechanism:

Definition 1. *(δ-Safe Update Mechanism) Given a machine learning model f^θ trained on a task with data distribution $P(X, Y)$ and a property ϕ we say that a mechanism $\mathcal{M}$ taking in an arbitrary dataset $\mathcal{D}'$ and parameter θ is a provably δ-safe update mechanism if:*

$$\forall \mathcal{D}', \quad \mathbb{E}_{P(X,Y)}\Big[\phi\big(\theta + \mathcal{M}(\theta, \mathcal{D}'), x, y\big)\Big] \leq \delta. \tag{1}$$

Before proceeding to discuss how one might formulate a non-trivial mechanism satisfying this definition, we highlight several key implications of the definition. First, the definition is independent of the updated model's performance on the new task (presumably represented by $\mathcal{D}'$), as this would be unrealistic given that $\mathcal{D}'$ is unspecified and thus may be adversarially chosen. Second, the fact that it is possible to choose $\mathcal{D}' = \emptyset$ implies that the existence of any δ-safe update mechanism depends upon the initial parameter being satisfactory, i.e., $\mathbb{E}_{P(X,Y)}\big[\phi\big(\theta, x, y\big)\big] \leq \delta$. Finally, given that this inequality holds, we highlight that the trivially safe update mechanism is the do-nothing mechanism, i.e., $\forall \mathcal{D}', \mathcal{M}(\theta, \mathcal{D}') = \mathbf{0}$ which satisfies the definition.

A. Local Invariant Domains (LIDs)

As a foundation for constructing provably sound update mechanisms, we now introduce the notion of Local Invariant Domains (LIDs).

Definition 2. *(Locally Invariant Domain) Given a machine learning model f^θ trained on a task with data distribution $P(X, Y)$, and a property ϕ, we say that a connected domain $T \subseteq \mathbb{R}^p$ containing θ is a provably δ-invariant domain if:*

$$\forall \theta' \in T, \, \mathbb{E}_{P(X,Y)}\Big[\phi\big(\theta', x, y\big)\Big] \leq \delta.$$

Any update mechanism restricted to outputs in a domain satisfying Definition 2 is one that satisfies Definition 1; however, it may be that $T = \{\theta\}$ which yields the trivial mechanism. To learn non-trivial safe update mechanisms, where possible, we propose a stronger, modified definition:

Definition 3. *(Maximal Locally Invariant Domain) Given a machine learning model f^θ trained on a task with data distribution $P(X, Y)$ and a property ϕ, we say that a connected domain $T \subseteq \mathbb{R}^p$ containing θ is a maximally provably δ-invariant domain with respect to a specified size metric $|\cdot|_S$ (e.g., Hausdorff measure), if it is a solution to the following optimization problem:*

$$\max_{T} |T|_S \text{ s.t. } \forall \theta' \in T, \, \mathbb{E}_{P(X,Y)}\Big[\phi\big(\theta', x, y\big)\Big] \leq \delta. \tag{2}$$

B. Provably Safe Updates via LIDs

In this section, we connect the notions of safe update mechanisms and LIDs, highlighting specific cases of how LIDs can be used to augment popular model updating mechanisms.

1) One-Step Updates (Fine-Tuning): Given access to a known LID, one can make any arbitrary update mechanism $\mathcal{M}$ a provably safe mechanism by transforming it into $M_T := \Pi_T(M(\theta, \mathcal{D}'))$ where Π is the standard projection operator $\Pi_T(\theta) := \arg\min_{\theta^\star \in \bar{T}} ||\theta - \theta^\star||$ and $\bar{T}$ is the closure of T to ensure this is well-defined. We emphasize that while the projection itself can be a nontrivial operation, any approximate projector that results in a parameter that belongs to T is satisfactory. While projection satisfies the definition, we can go further by carefully choosing the best T. To guide selection, we observe that for any loss function $\mathcal{L}$ that one wishes to minimize for the new task $\mathcal{D}'$, observe that $\min_{\theta \in T^\star} \mathcal{L}(\theta, \mathcal{D}') \leq \min_{\theta \in T} \mathcal{L}(\theta, \mathcal{D}')$ if $T \subset T^\star$. Thus, intuitively, computing the largest domain T yields the best performance on subsequent downstream tasks. We highlight that purely finding the *largest* domain may not be effective on its own, as the set may only be large in directions that are not helpful for improving downstream performance. Research in sparse updates in parameter isolation [16] or efficient fine-tuning [31] may represent powerful heuristics for identifying important directions. Additionally with access to some small portion of data from the desired downstream task, one can estimate the important directions. We discuss both of these approaches for improving LID computation in Section III-B4.

2) Multi-Step Updates (Continual Learning): In continual learning, the model is updated multiple times [34], and each update must ensure that all previous tasks respect a given constraint. Definition 1 can be generalized to k-many, ordered updates by defining the k^{th} update safety as: $\forall \mathcal{D}'_k, \forall i \in [k-1]$,

$$\mathbb{E}_{P(X_i,Y_i)}\Big[\phi\big(\theta + \sum_{j=1}^{k-1} \mathcal{M}(\theta^{(j-1)}, \mathcal{D}'_j), x, y\big)\Big] \leq \delta, \quad (3)$$

where $\theta^{(j)}$ is the result of the first j updates and the initial $\theta' = \theta^{(0)}$. Though the definition of safety has changed, the fundamental solution with LIDs needs only a minor modification to generalize to this case. For the i^{th} (previous) task, we compute a LID T_i, and we can then compute a new LID that holds for all tasks as $T^\star = \cap_{j=1}^{i} T_j$, the intersection of all independently computed LIDs. We highlight that this intersection does not need to be explicitly/analytically computed as we only require projection onto the intersection which can be done approximately.

3) Use of Replay Buffer: An important assumption we make in the formulation of our optimization problem (Definition 3) is that the LID is optimized over only one joint distribution $P(X, Y)$. This encodes the assumption that after the model is trained on a task, the corresponding test data can no longer be accessed. Many popular approaches to combating forgetting relax this assumption by allowing a small amount of replay data. Allowing access to data from prior tasks, we can jointly optimize our LID as follows:

$$\max_T |T|_S \quad \text{s.t.},$$
$$\forall \theta' \in T, i \in [k-1], \mathbb{E}_{P(X_i,Y_i)}, \Big[\phi\big(\theta', x, y\big)\Big] \leq \delta_i$$

where δ_i is the performance requirement for the i^{th} task. Importantly, this avoids the intersection step described in the previous section. Typically, replay methods try to make use of as little data as possible, meaning that expectations must be estimated with only a few samples, which can be done using the tools outlined in Subsection III-B5.

4) LID Biasing: Despite formalizing the *maximal* locally invariant domain problem in Definition 3, the maximization is posed with respect to current task. Even if one computes the exact maximal LID, the domain may not span directions that are important for improving performance on subsequent tasks. Identifying such important directions has been the focus of many prior continual learning approaches. In particular, we draw from weight pruning literature, which uses weight importance measures to determine parameters that significantly contribute to the performance of a given model [40], [41]. Below, we discuss how these importance methods, broken down into weight pruning or regularization, can be used to bias LID computation.

Weight pruning approaches define a discrete set of weights that are determined to be important. Formally, let $\mathcal{I}(\cdot)$ be a function associated with a weight pruning method that maps parameter indices to either 0 (if non-important) or 1 (if important) as determined by the pruning method. We can then restrict our prior optimization formulation to only be over only the unimportant parameter subspace i.e.., leaving all important weights fixed. Where we denote members of this subspace with $\mathbb{R}^p_{\mathcal{I}} \subseteq \mathbb{R}^p$ we can reformulate the optimization problem as:

$$\max_{T \in \mathbb{R}^p_{\mathcal{I}}} |T|_S \quad s.t. \quad \max_{\theta' \in T} \left[\frac{1}{N} \sum_{i=1}^{N} \phi(\theta', x_i, y_i)\right] \leq \delta$$

Intuitively, this modification only allows for updates to those weights that are unimportant for the current task, making it less likely that one violates the stated constraint (after $s.t.$ above). A critical parameter in weight pruning methods is the proportion, $\alpha \in (0, 1)$ of weights to be pruned – we explore this hyper-parameter in detail in Section V.

Additionally, one might consider a softer approach to constraining the LID optimization where modification of important weights is not explicitly restricted but discouraged. For this one might use the regularization approaches discussed in the continual learning literature. Formally, we can define $\mathcal{R}(\cdot)$ to be a function that assigns a cost (a negative value) to domains that allow large updates to important parameters. For example, where per-parameter importance is captured in a vector $\beta \in \mathbb{R}^p_+$ and magnitude is measured the largest change allowed by T per parameter, $d_T \in \mathbb{R}^p_+$, one could define $\mathcal{R}(T) = -\beta^\top d$. Such regularization approaches can be directly used to regularize our optimization formulation:

$$\max_{T \in \mathbb{R}^p} |T|_S + \mathcal{R}(T) \quad s.t. \quad \max_{\theta' \in T} \left[\frac{1}{N} \sum_{i=1}^{N} \phi(\theta', x_i, y_i)\right] \leq \delta$$

Lookahead buffers. As stated, the above methods rely solely on current task data to bias the LID computation

towards good solutions. While this may be effective, it still encodes assumptions that may not hold, e.g., that the important weights for this task should not be important for the next task. We can more directly bias the LID computation for good solutions if given a small amount of data (approximately) drawn from future task distributions. We refer to this method as *lookahead* buffer.

The lookahead approach enables us to estimate the subspace of parameters that is important to the *next* task. Given the data in the buffer are representative of the future task, each of the biasing approaches above (weight pruning and isolation) can be leveraged, but rather than computing weight importance on the current task and discouraging modification, one can compute weight importance for the future task and reward domains that enable large magnitude modifications to these weights.

5) Finite-Sample Guarantees for δ-Safe Updates: Definition 1 involves a constraint with an intractable integral over the distribution $P(X, Y)$, which is typically taken to be unknown. In practice, we approximate the expectation using a finite evaluation set that is held out during training (or held out within a data buffer): $\{(x^{(i)}, y^{(i)})\}_{i=1}^{N} \sim P(X, Y)$, with the empirical estimate denoted as,

$$\hat{\phi}_N := \frac{1}{N} \sum_{i=1}^{N} \phi\big(\theta + \mathcal{M}(\theta, \mathcal{D}'), x^{(i)}, y^{(i)}\big).$$

Assuming finite variance of this estimator, Chebyshev's inequality can be applied, giving: $\Pr\left(\left|\hat{\phi}_N - \mathbb{E}[\phi]\right| \geq \epsilon\right) \leq \frac{\sigma^2}{N\epsilon^2}$. For any $\beta \in (0, 1)$, solving for ϵ such that the right-hand side equals β, gives that with probability at least $1 - \beta$:

$$\mathbb{E}_{P(X,Y)}\left[\phi\left(\theta + \mathcal{M}(\theta, \mathcal{D}'), x, y\right)\right] \leq \hat{\phi}_N + \sqrt{\frac{\sigma^2}{N\beta}}.$$

Therefore, to certify δ-safety with confidence $(1 - \beta)$, it suffices that: $\hat{\phi}_N \leq \delta - \sqrt{\frac{\sigma^2}{N\beta}}$. If the risk metric is bounded $\phi(\cdot) \in [a, b]$, we can tighten this bound using Hoeffding's inequality to: $\hat{\phi}_N \leq \delta - \sqrt{(b-a)^2 \log(1/\beta)/(2N)}$.

However, if ϕ is an unbounded function (e.g., cross-entropy) and the variance is unknown, strict certification becomes challenging. In such cases, we can rely on the Central Limit Theorem (CLT) to provide a probabilistic argument for safety, approximating the error of an empirical estimate as $z_{1-\beta}\frac{\hat{\sigma}}{\sqrt{N}}$ where $z_{1-\beta}$ is the z-score at $(1 - \beta)$ and $\hat{\sigma}$ is the sample standard deviation. We note, however, while this is effective in practice for large sample sizes N, it remains an asymptotic approximation.

To ensure strict, provable finite-sample safety, we restrict our formal guarantees to bounded specification functions where $\phi(\cdot) \in [a, b]$. For nominally unbounded objectives, we employ a clipped version $\phi_{\text{clipped}} = \max(L, \min(\phi, U))$ where U forms the upper bound and L the lower bound. This ensures the variance is strictly bounded, validating the use of Chebyshev's inequality and further tightening arguments using Hoeffding's inequality. In our experiments, we utilize accuracy (bounded in $[0, 1]$), ensuring these finite-sample guarantees hold. While a theoretically valuable discussion, in practice most specification functions are naturally bounded or can be reasonably artificially bounded.

We emphasize that without such guarantees being formally stated, LID computation may be unsound. For example, without a finite-sample correction, a LID computed with safety constraints from a single held out test point could encompass a much larger parameter domain, but will ultimately merely yield a trivial safety guarantee. Moreover, the requirement for our concentration argument that sample points are i.i.d. from the joint distribution $P(X, Y)$ rules out any constraint sampling heuristics.

IV. Practical Computations for Safe Updates

We now present our computational framework for estimating maximal LIDs. Directly solving the optimization problem (2) over arbitrary connected domains T is computationally intractable. To obtain tractable approximations, we introduce two relaxations. First, we restrict T to a parametric family of abstract domains, denoted $T(\alpha)$, where α encodes the domain parameters (e.g., widths of orthotopes or linear constraints of zonotopes). Second, we approximate the inner expectation by an empirical average over a finite sample. These relaxations yield the following bi-level optimization problem:

$$\max_{\alpha} |T(\alpha)|_S \;\; s.t. \;\; \max_{\theta' \in T(\alpha)} \left[\frac{1}{N} \sum_{i=1}^{N} \phi(\theta', x_i, y_i)\right] \leq \delta. \quad (4)$$

In the subsequent sections, we outline the propagation of our chosen abstract domains and describe our primal-dual algorithm for solving this formulation.

A. Propagation of Abstract Domains

The inner maximization in (4) corresponds to a *parameter-space verification problem*, which seeks to upper bound the empirical loss over a set of model parameter values. This formulation is analogous to recent work on verifying robustness to parameter-space perturbations [23], [36], but differs from standard adversarial robustness settings that consider perturbations over the input space. Specifically, we are concerned with the worst-case model outputs over all parameter vectors rather than input vectors. For ease of exposition, we discuss the case where T is the *interval domain*, denoted $T(\alpha)$ with $\alpha := [\alpha^L, \alpha^U]$, which encodes all parameter vectors $\theta \in \mathbb{R}^p$ satisfying

$$\theta \in T(\alpha) \quad \iff \quad \forall i \in \{1, \ldots, p\}, \quad \alpha_i^L \leq \theta_i \leq \alpha_i^U.$$

This representation defines a box in parameter space and serves as an abstract domain, formally known as the orthotope domain, over which we can compute bounds on the output of a neural network using Interval Bound Propagation (IBP) [20].

Given a neural network f^θ and an interval domain $T(\alpha)$, IBP computes lower and upper bounds on the network output at an input point x for any $\theta \in T(\alpha)$. This propagation relies on all used activation functions being monotonic (e.g., ReLU,

tanh) [20]. This assumption should not be substantially restrictive in practice, as most commonly used activation functions are monotonic. Moreover, there are some ways to relax this requirement. Subsequently, these output bounds can then be propagated through the specification function ϕ to obtain a sound upper bound on the empirical specification estimate. In particular, we use the following IBP-based relaxation:

$$\max_{\theta \in T(\alpha)} \frac{1}{n} \sum_{i=1}^{n} \phi\left(\theta, x_i, y_i\right) \;\; \leq \;\; \frac{1}{n} \sum_{i=1}^{n} \phi_{\text{IBP}}\left(T\left(\alpha\right), x_i, y_i\right),$$

where $\phi_{\text{IBP}}(\cdot)$ denotes the IBP-derived upper bound on $\phi\left(\theta, x_i, y_i\right)$ valid for all $\theta \in T\left(\alpha\right)$. The IBP approximation is both *sound* (it provides a guaranteed upper bound) and *computationally efficient*, requiring at most $4\times$ the cost of a standard forward pass [20]. Moreover, the IBP procedure is differentiable with respect to the parameters $[\alpha^L, \alpha^U]$, enabling gradient-based optimization of the outer problem.

B. Primal-Dual Optimization for Computing Maximal Locally Invariant Domains

Interval domains offer not only a natural parameterization for the constrained problem in (4), but also a differentiable and sound over-approximation of the constraint. This yields a single-level constrained optimization problem that constitutes a *sound under-approximation* of the original formulation. That is, any solution identified via this relaxation defines a LID that satisfies the safety specification, although it may be strictly smaller than the solution of the original bi-level problem (4).

Although the relaxed problem still does not admit an analytic solution, it is amenable to Lagrangian-based optimization. Specifically, to maximize the size of the abstract domain subject to the IBP-based constraint, we formulate the following Lagrangian:

$$L(\alpha, \lambda) = |T(\alpha)|_S + \lambda \left(\delta - \frac{1}{n} \sum_{i=1}^{n} \phi_{\text{IBP}}\left(T\left(\alpha\right), x_i, y_i\right) \right),$$

where $\lambda \geq 0$ is the Lagrange multiplier for the (bounded) safety specification constraint.

This gives rise to the saddle-point optimization $\max_\alpha \min_{\lambda \geq 0} L(\alpha, \lambda)$ which can be locally solved using *alternating gradient descent-ascent* [42], which performs the updates:

$$\lambda_{t+1} = \left[\lambda_t + \eta_d \nabla_\lambda L(\alpha_t, \lambda_t)\right]_+, \tag{5}$$

$$\alpha_{t+1} = \alpha_t - \eta_p \nabla_\alpha L(\alpha_t, \lambda_{t+1}), \tag{6}$$

where η_d and η_p are step sizes for the dual and primal updates, respectively, and $[\cdot]_+$ denotes projection onto the non-negative orthant to ensure $\lambda \geq 0$[2]. We additionally apply an extra projection step to α that enforces that the nominal parameters θ remain inside $T(\alpha)$. The above primal-dual optimization procedure is designed to maintain feasibility with respect to the safety specification constraint throughout training. However, due to the inherent stochasticity of gradient-based optimization, the final iterate may not strictly satisfy the original constraint. To formally verify that the returned LID, denoted $T(\alpha)$, satisfies the desired specification, we compute a post-optimization certificate by computing $\max_{\theta' \in T(\alpha)} \frac{1}{n} \sum_{i=1}^{n} \phi(\theta', x_i, y_i)$.

[2]We employ the `cooper` library for Lagrangian-based optimization [43].

C. Choice of Specification Function

Effectively applying *alternating gradient descent-ascent* algorithms and solving Equation 4 imposes specific requirements on the specification function ϕ. Most notably, any specification function must be differentiable. While a convenient choice for ϕ is accuracy, due to its interpretability, we cannot directly use accuracy since it is non-differentiable. Thus, our experiments rely on using soft accuracy ϕ_{soft} as a differentiable surrogate:

$$\phi_{\text{soft}}(\theta) = \frac{1}{N} \sum_{k=0}^{N} \sum_{c=1}^{C} y_{k,c} \text{Softmax}(f^\theta(x_k))_c$$

where y_k is taken to be the one-hot encoding vector of the correct class label.

Although soft accuracy, or any differentiable surrogate, serves as the specification function during maximal LID computation, the final certificate of the LID must be based on the true specification function (e.g., minimum accuracy over the LID). This distinction preserves the correctness of our approach. For further discussion we refer to Appendix C.

D. Early stopping and Checkpointing

One specific improvement we propose—alongside less notable, further adaptations found in Appendix B—is early stopping and check-pointing based on findings in Figure 2. Figure 2 shows an example LID computation using our primal-dual optimization. We observe that the primal objective (the LID size metric) does not converge within 1000 iterations. On the other hand, by inspecting the LIDs at each step of optimization, we observe that the LIDs with the best downstream performance on Task 2 occur early on in the LID optimization. This suggests that the LID computation tends to prioritize weights that do not contribute significantly to the model's output while keeping decisive weights for the next task frozen (i.e., assigning them a very small interval).

To mitigate this effect our implementation saves the current LID periodically during the computation and terminates the LID optimization before convergence. This produces a set $\mathcal{O}$ of domain parameters each corresponding to a LID. We then issue a certificate of the minimum accuracy for each LID in $\mathcal{O}$. When training the next task, we must project the model parameters (the result of proposed updates) θ onto the set of LIDs $\mathcal{O}$. Several strategies could be used to choose the LID o^* on which we project θ. An intuitive extension of the general case above is projecting onto the closest LID using:

$$o^* = \arg\min_{o \in \mathcal{O}} \|\Pi(\theta, o) - \theta\|_2^2$$

where Π is the projection operator defined as:

$$\Pi(\theta, o)_i = \text{Clamp}(\theta_i, \theta_i^L, \theta_i^U), \qquad o = [\theta^L, \theta^U]$$

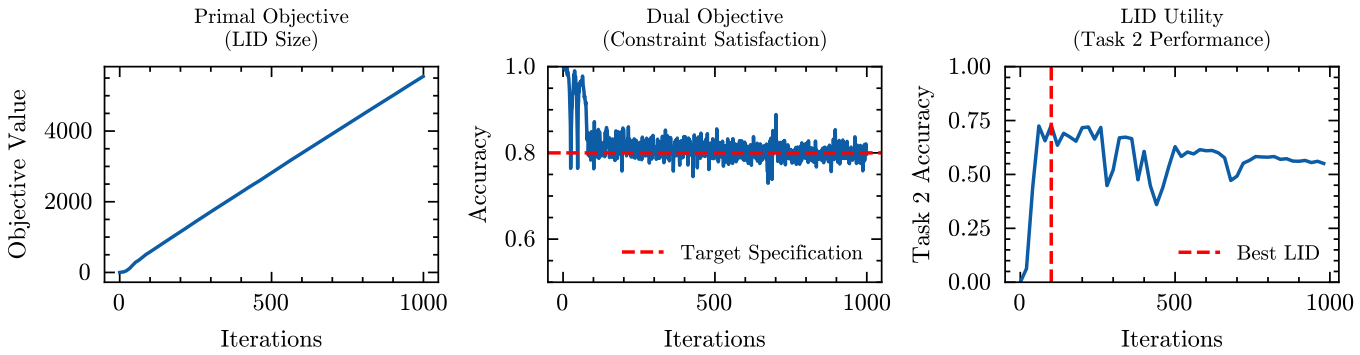

Fig. 2. Analysis of primal-dual optimization for LID computation. We observe that although the primal-dual optimization continues to find larger LIDs according to the primal objective, the LIDs with the best utility occur early on in optimization.

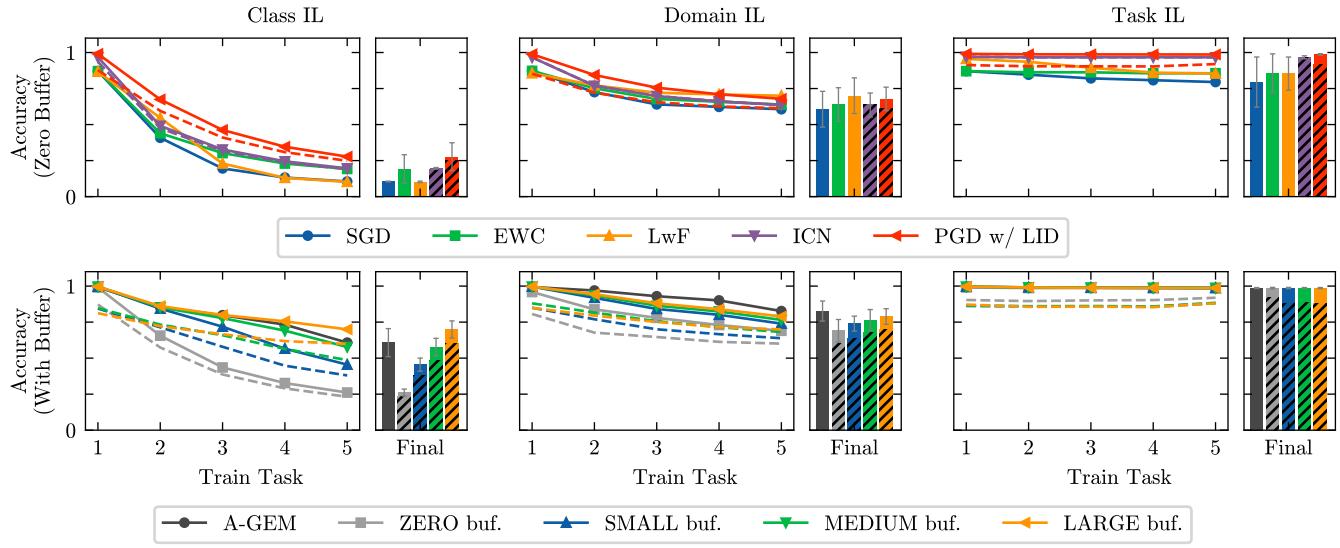

Fig. 3. Performance on the Split-MNIST dataset under three continual learning scenarios. The top panel shows the performance of our method with zero buffer, while the bottom panel shows the performance with various buffer sizes. The dashed line and hashed bars display the certificates on performance. Error bars illustrate the 80% confidence interval over 100 runs for the zero buffer and 15 runs for the buffer case.

Another strategy is to choose the LID o^* that results in the best immediate performance on our current batch. Letting $\mathcal{B} := \{(x^{(i)}, y^{(i)}\}_{i=1}^{B}$ be a batch of data the loss :

$$o^* = \arg\min_{o \in \mathcal{O}} \mathcal{L}(\Pi(\theta, o), \mathcal{B})$$

where $\mathcal{L}(\theta, \mathcal{B})$ denotes the loss over the batch at θ.

a) Downstream Advantages of Checkpointing: The introduction of checkpointing forced an adaptation in our overall algorithm to allow for multiple LIDs for a single model. This can be combined with aforementioned biasing methods to compute distinctly biased LIDs that can be adapted to different types of downstream tasks. Having LIDs that are expanded along separate dimensions of the parameter space, lends a significant amount of flexibility to our algorithm that predecessors like InterContiNet did not exploit [24].

While we do not further explore the practical impact of obtaining differently biased LIDs for a single model, we consider this a considerable advantage of isolating the LID problem. LID multiplicity may be a fruitful area of study for future works.

E. Algorithm Discussion

a) Strengths: A major strength of our proposed framework is its compatibility with a wide range of prior research advances. In particular, we can leverage a large number of existing approaches in forgetting-safe continual learning and in alignment-preserving fine-tuning. Moreover, given that our method is independent of how one arrives at θ for their initial task, our method can be used, in principle with any machine learning model. This means we can accommodate arbitrarily large foundation models. Additionally, because maintaining our guarantees only requires the introduction of a projection operator, we can apply our approach to any proposed updating scheme. This is particularly advantageous given the rapid rate at which model updating algorithms advance, e.g., RLHF algorithms such as DPO, PPO, and GRPO. Finally, our approach relying on abstract interpretation is computationally efficient

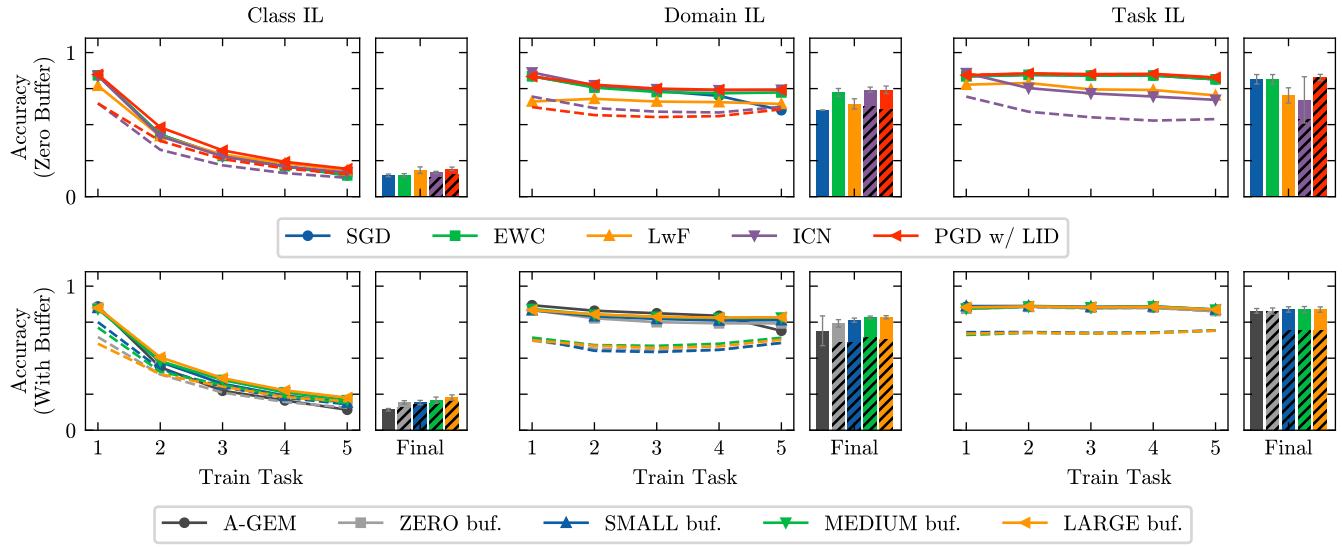

Fig. 4. Performance on the Split-CIFAR10 [26] dataset under three continual learning scenarios. The top panel shows the average accuracy over all seen tasks without replay data, while the bottom panel the average performance over seen tasks, while having varying amounts of buffer data available. The dashed line and hashed bars display the certificates returned by our method. Error bars illustrate the 80% confidence interval over 15 runs.

and is able to certify a LID, expressed as an interval domain, in $4\times$ the cost of a simple forward pass.

b) Limitations: Despite its computational efficiency and differentiability, the IBP approximation we leverage is known to exhibit increasing looseness with network depth and width [23], [44]. Thus, while arbitrarily large networks can be considered, parameter intervals over the early layers of the network contribute disproportionately to the overall interval bounds on the outputs [45]. As a result, our LIDs likely exhibit larger intervals for the later layers of the network compared to the earlier layers. In sufficiently deep networks, this effect can lead to the early layers becoming effectively 'frozen,' with little (or no) flexibility for future adaptation, while the final layers, subject to less over-approximation, remain relatively less constrained. This difficulty means that ensuring useful guarantees for large models requires careful design and hyper-parameter choices; we provide further systematic discussion of this phenomenon in Appendix C.

c) Potential Future Applications: In this section we discuss some of the potential future applications enabled by a framework that enables *a priori* safe update guarantees. Most directly, the requirement for an *a priori* safe update mechanism has been put forth as a regulatory requirement for adjusting models in some critical domains. For instance, the U.S. Food and Drug Administration (FDA) outlined guidelines that enable pre-planned model updates, but only if they can be demonstrated to be safe *a priori* [9]. Furthermore, regulatory attention has also been attracted by the wide-spread use of foundation models (e.g. large language models). Regulations surrounding foundation models may require persistent safety mechanisms, especially given that models must respond dynamically to emerging vulnerabilities. While a valid response would be model de-provisioning (taking the model offline), using known-safe model updates is preferable for availability-critical systems. In the context of updating provided or open-source large language models, a potential use case of our method could be guaranteeing the preservation of learned behavior (e.g. alignment) for down-stream fine-tuners. As we have discussed, if not done carefully, fine-tuning can rapidly erode alignment properties [18], [19]. Thus, allowing third-party users to safely tune a model while the publishing entity can ensure, *a priori*, that this individualized updating procedure does not yield a model that violates originally aligned behavior. Such a guarantee may lead to robust terms-of-use.

V. Experiments

We empirically validate our approach on standard continual learning benchmarks and provide further experimental details and ablations in Appendix D. All experiments were performed on compute infrastructure equipped with 2x NVIDIA L40 GPUs, using a custom Pytorch implementation. Our codebase will be publicly released upon publication.

A. Continual Learning Experiments

Baselines We compare our method against the following established continual learning methods: (1) EWC [7]: a regularization approach that selectively weights important parameters based on Fisher information, (2) LwF [12]: a distillation-based method that preserves functional behavior by using current task samples, (3) SGD: sequential training across tasks without explicit mitigation of catastrophic forgetting and (4) InterContiNet (ICN): a weight-space-constraint based approach to avoiding catastrophic forgetting [24]. To benchmark our buffer extension, we observe our method compared to its standard, zero-buffer implementation and Averaged GEM

(A-GEM) [46]. For each baseline we consider the main incremental learning protocols outlined in [34]:

- **Class-IL:** The model learns to discriminate between an expanding set of classes. Each context introduces a new set of classes (e.g., first context contains '0' and '1', second context contains '2' and '3', etc.), and the model must classify across all previously encountered classes without task identity information at test time.
- **Domain-IL:** The underlying task remains constant while the input distribution shifts across contexts. The model performs binary classification (odd vs. even digits in Split-MNIST and animal vs. non-animal in Split-CIFAR), but each context presents previously unseen input domains. Task identity remains unavailable during inference.
- **Task-IL:** Learning to classify an expanding set of classes, but only within the boundaries of each task context. Critically, task identity is provided during inference, allowing the model to constrain its prediction space appropriately.

a) Evaluations: In Figures 3 and 4 we compare the performance of our approach and baselines. In the top row of each figure we plot the average accuracy/certificate on all previously seen tasks (y-axis), as we observe the current task (x-axis). In the bottom row, we show the performance of the buffer-extension to our approach using a small (1000 samples), medium (5000 samples), and large (15000 samples) buffer. Additionally, we compare our method with A-GEM with a large buffer and present results in a manner similar to the top row plots.

b) Split-MNIST: We divide the standard MNIST dataset [25] into five sequential contexts, each containing an odd and an even number. Digit allocation across contexts was randomized for each experimental run, with reported performance metrics averaged across 100 independent runs. Figure 3 presents our method's performance across the three Split-MNIST scenarios. We train a feedforward neural network with 2 convolutional layers (each with 8 channels and a 5×5 kernel), followed by a dense layer. In Task-IL, all continual learning approaches effectively acquire new tasks while preserving performance on previous ones. However, for the more challenging Class-IL and Domain-IL settings, average accuracy deteriorates as task count increases. Impressively, our approach demonstrates competitive performance in the Class-IL setting, while simultaneously providing theoretical guarantees (dashed line) that bound the minimum accuracy under any future learning. In Domain-IL, our method maintains competitive average performance compared to baseline approaches while exhibiting improved retention of first-task performance. Moreover, our method observes more flexibility in its constraints, whereas ICN is highly prone to constraint collapse, as evident by the depicted tightness of the bounds. Additionally, using buffer data can drastically increase the performance of our method with a large buffer outperforming even A-GEM in the most difficult continual learning setting, CIL, and strictly improving performance in Domain-IL. Task-IL does not show the same trend as the setting is too simple and allows for enough model flexibility for even the zero buffer case to extract consistently high performance.

c) Split-CIFAR: We now turn to the Split-CIFAR10 dataset and consider the training and subsequent fine-tuning of a pre-trained ResNet18 model. The entire model is first trained on the initial task, after which subsequent tasks are learned by fine-tuning only the final linear layer of the network. The 10 classes are randomly allocated to 5 tasks, following the three incremental learning settings described above. In Figure 4, we observe similar overall trends to Split-MNIST; specifically, our method is consistently competitive with baseline approaches, while also offering formal guarantees on forgetting performance. In Domain-IL, we observe consistently better accuracy than baselines though our worst-case guarantees are below the performance of baseline methods (which is generally to be expected). Class incremental learning remains extremely challenging for all methods–where we consider only preserving task 1 performance, our method significantly out-performs baselines.

Following the findings on Split-MNIST, our method further distinguishes itself from InterContiNet on Split-CIFAR10, where ICN lacks significant adaptability to different task settings due to its sensitivity to hyperparameters and numerical instability.

B. Fine-Tuning Experiments

Previous experiments have primarily focused on multi-step updates, we now further explore our method on single-step updates via foundation model fine-tuning.

1) Foundation Model Performance on Hate-Speech Classification: We evaluate our approach on the binary task of Hate-Speech Classification (non-hate, hate labels) for the following datasets: Jigsaw [47], Twitter Hate Speech [48], Civil Comments [49], Hatemoji [50], Sbic [51] and Hatecheck [52]. For that purpose we use the frozen embeddings of the following foundation models that at the time of writing have top performance on existing benchmarks : 3-large (OpenAI) [53], MXBAI (Mistral) [54], M2-BERT (Together) [55], GTE-Large (Alibaba) [56], Voyage-3 (Voyage) [57].

For each foundation model, we proceed by first training a linear layer in an unconstrained manner on the frozen embeddings of that model on the Jigsaw dataset. For each model, we then certify approximately maximal LIDs for the linear layer assuming a macro-accuracy (also known as balanced accuracy) constraint on Jigsaw.

We then evaluate our approach by updating the Jigsaw-trained model on each of the five downstream datasets. The results are summarized in Figure 5, where each subplot corresponds to a different target dataset and visualizes the performance trade-offs induced by LID-constrained updates. Along the x-axis, we show performance on the original task (Jigsaw), while the y-axis reflects performance on the new task. In particular, the x-axes show the initial performance on Jigsaw (denoted with a star), the post-finetuning performance with the LID (denoted with an open dot), and the formal lower-bound from the LID on Jigsaw macro accuracy (denoted with

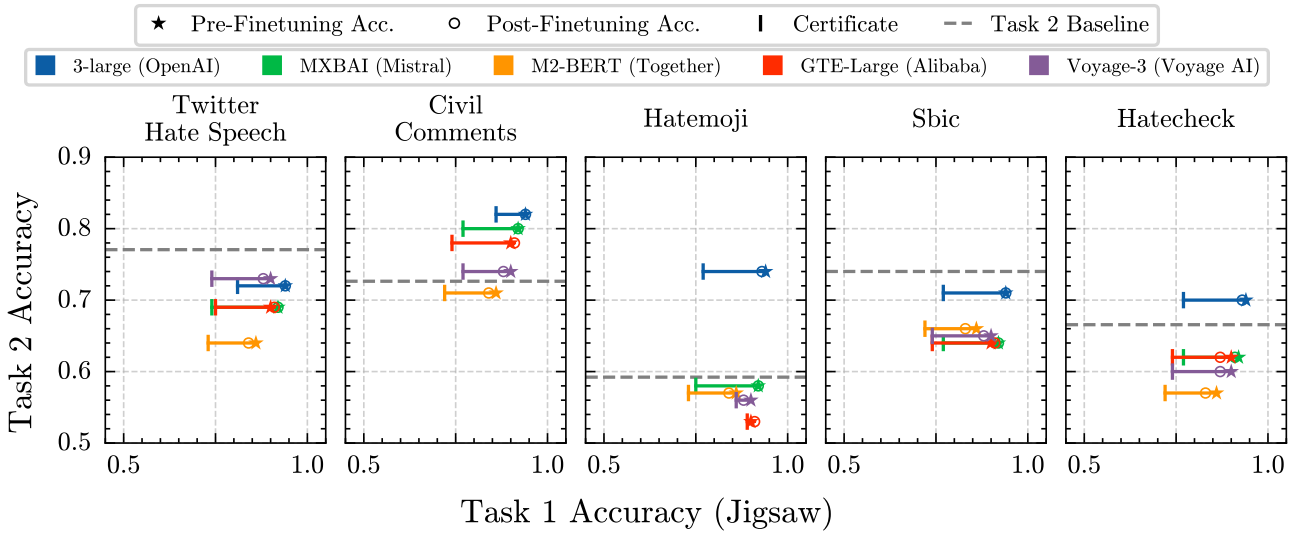

Fig. 5. Performance of fine-tuning with LIDs in the Hate-Speech Classification setting, where certificates are given with respect to the first task. Each color corresponds to a different LLM embedding model with the company of origin in parentheses. The gray dashed line denotes the average baseline performance on the fine-tuned task when directly using the M2-BERT model..

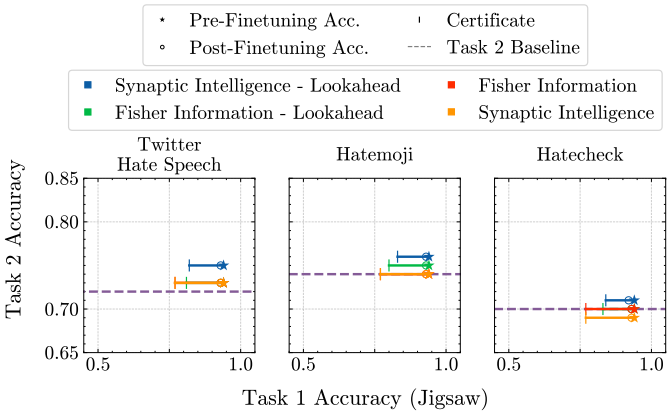

Fig. 6. Performance of various biasing metrics applied to LLM head fine-tuning for hate speech classification. All data points are based on 3-large (OpenAI) as the underlying foundation model. The baseline corresponds to certified fine-tuning without any biasing.

a bar). Performance on the new task is shown along the y-axis, with the gray dashed line indicating the baseline performance of M2-BERT with EWC. Across all model-dataset pairs, we observe that LID-constrained updates may slightly degrade downstream accuracy, though in some cases performance is on par with or exceeding the unconstrained M2-BERT baseline is observed. Simultaneously, our method certifies non-trivial lower bounds on the original task's macro accuracy, often guaranteeing that substantial accuracy is retained even under worst-case perturbations within the safe domain.

2) Foundation Model Performance on Multilingual Sentiment Analysis: Following the same procedure as above, we additionally evaluate our approach on the multilingual classification task of sentiment analysis (negative, neutral, positive) for the following multilingual dataset : XLM-T [58] which contains 8 splits corresponding to 8 different languages

We provide a subset of the results in Figure 7. The dotted gray line represents the baseline performance of Voyage-3 using EWC [7]. When looking at the performances on Gemini, Voyage-AI and 3-large, we observe that our method achieves performances comparable to the EWC baseline and even occasionally outperforms it. Additionally, our method certifies non-trivial bounds on the original task macro-accuracy. These results provide the first empirical evidence that formal forgetting/alignment-preserving guarantees can be achieved in real-world LLM-based NLP pipelines, without sacrificing utility on the new task. The certified safety margins, though conservative, are large enough to ensure that updates remain practically useful.

C. LID Biasing Experiments

In this section we explore ablations on our proposed LID biasing approaches by considering the Split-MNIST CIL setting. We keep the convolutional architecture described in prior experiments and consider a non-biased LID computation as a baseline. We explore biasing the optimization using the Fisher information matrix [59] and synaptic intelligence [60] to obtain weight importance metrics by training our model on a batch with either lookahead data or the original task data. All experiments in this section consider pruning rather than regularization.

In Table I, we show the performance on a downstream task across different pruning percentages. We re-run our results using 15 different random seeds and plot the standard deviation as our error bar. For both weight selection using Fisher information and synaptic intelligence, we observe significant gains in the average case, but with large variance. On the other hand, with a lookahead buffer, we observe that biasing can aid in downstream performance across a relatively broad range of

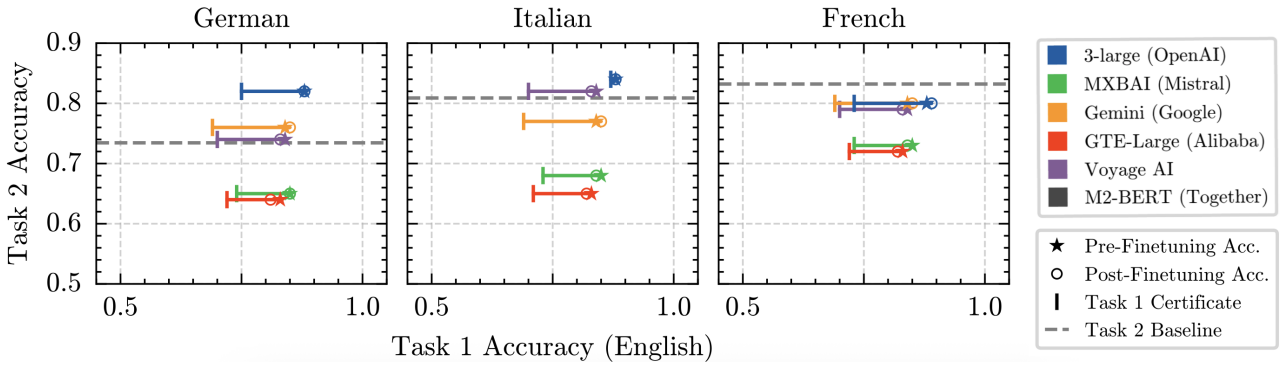

Fig. 7. Performance of fine-tuning with LIDs on multilingual sentiment analysis task. We update a sentiment analysis classifier originally trained on english language reviews on multilingual reviews. We observe non-trivial guarantees on all benchmarks (vertical line) while maintaining performance on task 1 (hollow dot, x-axis) and besting baseline performance (dashed line, y-axis).

TABLE I

TASK 2 ACCURACY COMPARISON ACROSS DIFFERENT PRUNING PROPORTIONS WITH AND WITHOUT LOOKAHEAD BUFFERS. VALUES ARE REPORTED AS MEAN ± STD DEV. THE BASELINE PERFORMANCE FOR AN UNBIASED LID IS 0.20.

Pruning Prop.	**0.5**	**0.7**	**0.8**	**0.825**	**0.85**	**0.9**	**0.95**
With Lookahead							
Synaptic Intelligence	0.37 ± 0.22	0.53 ± 0.29	0.67 ± 0.19	0.73 ± 0.18	0.65 ± 0.19	0.39 ± 0.22	0.07 ± 0.14
Fisher Information	0.35 ± 0.18	0.45 ± 0.31	0.64 ± 0.26	0.73 ± 0.15	0.67 ± 0.12	0.38 ± 0.25	0.05 ± 0.12
Pruning Prop.	**0.5**	**0.3**	**0.2**	**0.175**	**0.15**	**0.1**	**0.05**
Standard (No Buffer)							
Synaptic Intelligence	0.14 ± 0.11	0.37 ± 0.20	0.28 ± 0.18	0.28 ± 0.17	0.38 ± 0.18	0.38 ± 0.26	0.39 ± 0.24
Fisher Information	0.14 ± 0.10	0.35 ± 0.21	0.31 ± 0.21	0.31 ± 0.22	0.36 ± 0.23	0.39 ± 0.21	0.39 ± 0.25

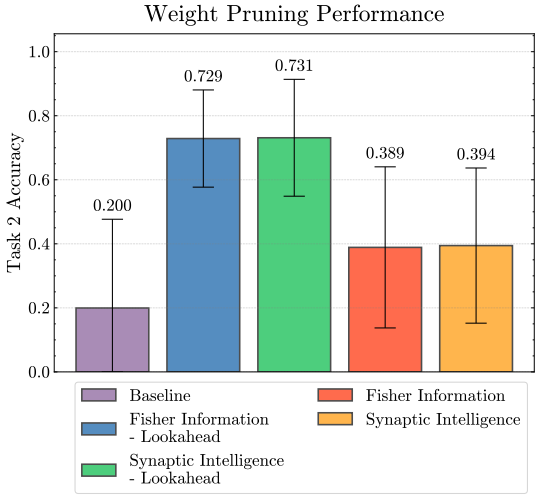

Fig. 8. Performance of all of our biasing approaches for the best performing pruning percentages based on Table I. Pruning percentages are 82.5% and 5% for lookahead and no lookahead pruning, respectively. All methods are run across 15 seeds, displaying the mean performance with error bars showing the standard deviation.

pruning proportions. Performance seems to peak at 82.5% and 5% of frozen parameters for the lookahead and naive method, respectively.

Figure 8 further highlights the significance of the performance improvements for single-step continual learning scenarios (i.e. limited to two tasks) when using LID biasing. In particular, lookahead biasing unlocks a significant amount of performance by drastically simplifying the LID optimization problem. However, even the smaller amount of pruning that occurs in the non-lookahead optimization yields a simplification that is often sufficient to extract additional performance.

We additionally explore these ablations for fine-tuning LLMs. Figure 6 shows that lookahead biasing performs better than or equal to an unbiased baseline in all of the explored settings, whether it be hate speech detection or multilingual sentiment analysis. Our observations from the MNIST setting remain valid in LLM fine-tuning with LID biasing playing a relatively minor role when lookahead buffer data is not used. Interestingly, we find that synaptic intelligence with a lookahead buffer allows noticeable improvements in both formal guarantees and downstream performance. Further experimental details can be found in Table V in the appendix.

VI. CONCLUSION

In this work, we introduce a novel framework for computing provably safe model updates via maximal locally invariant domains (LIDs), offering formal guarantees that behavioral properties such as alignment and task performance are preserved. By leveraging abstract interpretation and a primal-dual optimization strategy, our approach enables efficient

certification of model updates in both continual learning and foundation model fine-tuning scenarios. Across diverse empirical settings, we demonstrated that our method not only provides formal safety guarantees for these updates, but also matches or outperforms existing baselines in practice.

We then extend this approach by directing the optimization of locally invariant domains using biasing inspired by classical continual learning approaches. In particular, we find that biasing drastically reduces the complexity of the LID computation, yielding more effective parameter subspaces. We further show that this not only holds theoretically but also experimentally.

REFERENCES

[1] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[2] L. Szpruch, A. Orfanoudaki, C. Maple, M. Wicker, Y. Bengio, K.-Y. Lam, and M. Detyniecki, "Insuring ai: Incentivising safe and secure deployment of ai workflows," *Available at SSRN 5505759*, 2025.

[3] X. Qi, Y. Zeng, T. Xie, P.-Y. Chen, R. Jia, P. Mittal, and P. Henderson, "Fine-tuning aligned language models compromises safety, even when users do not intend to!" *arXiv preprint arXiv:2310.03693*, 2023.

[4] X. Yang, X. Wang, Q. Zhang, L. Petzold, W. Y. Wang, X. Zhao, and D. Lin, "Shadow alignment: The ease of subverting safely-aligned language models," *arXiv preprint arXiv:2310.02949*, 2023.

[5] Q. Zhan, R. Fang, R. Bindu, A. Gupta, T. Hashimoto, and D. Kang, "Removing rlhf protections in gpt-4 via fine-tuning," *arXiv preprint arXiv:2311.05553*, 2023.

[6] S. Lermen, C. Rogers-Smith, and J. Ladish, "Lora fine-tuning efficiently undoes safety training in llama 2-chat 70b," *arXiv preprint arXiv:2310.20624*, 2023.

[7] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[8] L. Wang, X. Zhang, H. Su, and J. Zhu, "A comprehensive survey of continual learning: Theory, method and application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[9] U.S. Food and Drug Administration and Health Canada and Medicines and Healthcare products Regulatory Agency, "Predetermined change control plans for machine learning-enabled medical devices: guiding principles," 2023.

[10] J. Mökander, J. Schuett, H. R. Kirk, and L. Floridi, "Auditing large language models: a three-layered approach," *AI and Ethics*, vol. 4, no. 4, pp. 1085–1115, 2024.

[11] M. Wicker, L. Szpruch, and S. Mørk, "Move fast without breaking the bank model risk management of genai workflows," 2025.

[12] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[13] Y. Shen, Y. Xiong, W. Xia, and S. Soatto, "Towards backward-compatible representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6368–6377.

[14] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

[15] T. Huang, S. Hu, F. Ilhan, S. Tekin, and L. Liu, "Lisa: Lazy safety alignment for large language models against harmful fine-tuning attack," *Advances in Neural Information Processing Systems*, vol. 37, pp. 104 521–104 555, 2024.

[16] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[17] H. Kang, R. J. L. Mina, S. R. H. Madjid, J. Yoon, M. Hasegawa-Johnson, S. J. Hwang, and C. D. Yoo, "Forget-free continual learning with winning subnetworks," in *International Conference on Machine Learning*. PMLR, 2022, pp. 10 734–10 750.

[18] M. Anderljung, J. Barnhart, A. Korinek, J. Leung, C. O'Keefe, J. Whittlestone, S. Avin, M. Brundage, J. Bullock, D. Cass-Beggs *et al.*, "Frontier ai regulation: Managing emerging risks to public safety," *arXiv preprint arXiv:2307.03718*, 2023.

[19] X. Qi, B. Wei, N. Carlini, Y. Huang, T. Xie, L. He, M. Jagielski, M. Nasr, P. Mittal, and P. Henderson, "On evaluating the durability of safeguards for open-weight llms," *arXiv preprint arXiv:2412.07097*, 2024.

[20] S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelovic, T. Mann, and P. Kohli, "On the effectiveness of interval bound propagation for training verifiably robust models," *arXiv preprint arXiv:1810.12715*, 2018.

[21] C. Tsay, J. Kronqvist, A. Thebelt, and R. Misener, "Partition-based formulations for mixed-integer optimization of trained relu neural networks," *Advances in neural information processing systems*, vol. 34, pp. 3068–3080, 2021.

[22] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev, "Ai2: Safety and robustness certification of neural networks with abstract interpretation," in *2018 IEEE symposium on security and privacy (SP)*. IEEE, 2018, pp. 3–18.

[23] P. Sosnin, M. N. Müller, M. Baader, C. Tsay, and M. Wicker, "Certified robustness to data poisoning in gradient-based training," *arXiv preprint arXiv:2406.05670*, 2024.

[24] M. Wołczyk, K. Piczak, B. Wójcik, L. Pustelnik, P. Morawiecki, J. Tabor, T. Trzcinski, and P. Spurek, "Continual learning with guarantees via weight interval constraints," in *International Conference on Machine Learning*. PMLR, 2022, pp. 23 897–23 911.

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[26] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[27] T. Huang, S. Hu, F. Ilhan, S. F. Tekin, and L. Liu, "Harmful fine-tuning attacks and defenses for large language models: A survey," *arXiv preprint arXiv:2409.18169*, 2024.

[28] S. Zhao, M. Jia, Z. Guo, L. Gan, X. XU, X. Wu, J. Fu, F. Yichao, F. Pan, and A. T. Luu, "A survey of recent backdoor attacks and defenses in large language models," *Transactions on Machine Learning Research*.

[29] X. Huang, W. Ruan, W. Huang, G. Jin, Y. Dong, C. Wu, S. Bensalem, R. Mu, Y. Qi, X. Zhao *et al.*, "A survey of safety and trustworthiness of large language models through the lens of verification and validation," *Artificial Intelligence Review*, vol. 57, no. 7, p. 175, 2024.

[30] J. Mukhoti, Y. Gal, P. H. Torr, and P. K. Dokania, "Fine-tuning can cripple your foundation model; preserving features may be the solution," *arXiv preprint arXiv:2308.13320*, 2023.

[31] B. Wei, K. Huang, Y. Huang, T. Xie, X. Qi, M. Xia, P. Mittal, M. Wang, and P. Henderson, "Assessing the brittleness of safety alignment via pruning and low-rank modifications," *arXiv preprint arXiv:2402.05162*, 2024.

[32] D. Rosati, J. Wehner, K. Williams, L. Bartoszcze, R. Gonzales, S. Majumdar, H. Sajjad, F. Rudzicz *et al.*, "Representation noising: A defence mechanism against harmful finetuning," *Advances in Neural Information Processing Systems*, vol. 37, pp. 12 636–12 676, 2024.

[33] J. Rando, J. Zhang, N. Carlini, and F. Tramèr, "Adversarial ml problems are getting harder to solve and to evaluate," *arXiv preprint arXiv:2502.02260*, 2025.

[34] G. M. Van de Ven, T. Tuytelaars, and A. S. Tolias, "Three types of incremental learning," *Nature Machine Intelligence*, vol. 4, no. 12, pp. 1185–1197, 2022.

[35] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," *arXiv preprint arXiv:1708.01547*, 2017.

[36] M. Wicker, J. Heo, L. Costabello, and A. Weller, "Robust explanation constraints for neural networks," *arXiv preprint arXiv:2212.08507*, 2022.

[37] M. Wicker, P. Sosnin, I. Shilov, A. Janik, M. N. Müller, Y.-A. de Montjoye, A. Weller, and C. Tsay, "Certification for differentially private prediction in gradient-based training," *arXiv preprint arXiv:2406.13433*, 2024.

[38] J. Bitterwolf, A. Meinke, and M. Hein, "Certifiably adversarially robust detection of out-of-distribution data," *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 085–16 095, 2020.

[39] W. Wang, A. J. Levine, and S. Feizi, "Improved certified defenses against data poisoning with (deterministic) finite aggregation," in *International Conference on Machine Learning*. PMLR, 2022, pp. 22 769–22 783.

[40] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[41] S. Golkar, M. Kagan, and K. Cho, "Continual learning via neural pruning," 2019. [Online]. Available: https://arxiv.org/abs/1903.04476

[42] G. Zhang, Y. Wang, L. Lessard, and R. B. Grosse, "Near-optimal local convergence of alternating gradient descent-ascent for minimax optimization," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 7659–7679.

[43] J. Gallego-Posada, J. Ramirez, M. Hashemizadeh, and S. Lacoste-Julien, "Cooper: A library for constrained optimization in deep learning," *arXiv preprint arXiv:2504.01212*, 2025.

[44] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*. Springer, 2017, pp. 97–117.

[45] M. Wicker, L. Laurenti, A. Patane, and M. Kwiatkowska, "Probabilistic safety for bayesian neural networks," in *Conference on uncertainty in artificial intelligence*. PMLR, 2020, pp. 1198–1207.

[46] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," 2019. [Online]. Available: https://arxiv.org/abs/1812.00420

[47] cjadams, J. Sorensen, J. Elliott, L. Dixon, M. McDonald, nithum, and W. Cukierski, "Toxic comment classification challenge," https://kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge, 2017, kaggle.

[48] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in *Proceedings of the international AAAI conference on web and social media*, vol. 11, no. 1, 2017, pp. 512–515.

[49] D. Borkan, L. Dixon, J. Sorensen, N. Thain, and L. Vasserman, "Nuanced metrics for measuring unintended bias with real data for text classification," *CoRR*, vol. abs/1903.04561, 2019. [Online]. Available: http://arxiv.org/abs/1903.04561

[50] H. R. Kirk, B. Vidgen, P. Röttger, T. Thrush, and S. Hale, "Hatemoji: A test suite and adversarially-generated dataset for benchmarking and detecting emoji-based hate," *CoRR*, vol. abs/2108.05921, 2021. [Online]. Available: https://arxiv.org/abs/2108.05921

[51] M. Sap, S. Gabriel, L. Qin, D. Jurafsky, N. A. Smith, and Y. Choi, "Social bias frames: Reasoning about social and power implications of language," in *ACL*, 2020.

[52] P. Röttger, B. Vidgen, D. Nguyen, Z. Waseem, H. Z. Margetts, and J. B. Pierrehumbert, "Hatecheck: Functional tests for hate speech detection models," *CoRR*, vol. abs/2012.15606, 2020. [Online]. Available: https://arxiv.org/abs/2012.15606

[53] OpenAI, "text-embedding-3-large," https://platform.openai.com/docs/guides/embeddings, 2024, accessed May 2025. An embedding model from the GPT family.

[54] C. Choi, J. Kim, S. Lee, J. Kwon, S. Gu, Y. Kim, M. Cho, and J.-y. Sohn, "Linq-embed-mistral technical report," *arXiv preprint arXiv:2412.03223*, 2024.

[55] T. Computer, "m2-bert-80m-2k-retrieval," https://huggingface.co/togethercomputer/m2-bert-80M-2k-retrieval, 2024, accessed May 2025. Open-weight embedding model optimized for retrieval tasks.

[56] X. Ye, S. Iyer, A. Celikyilmaz, V. Stoyanov, G. Durrett, and R. Pasunuru, "Complementary explanations for effective in-context learning," *arXiv preprint arXiv:2211.13892*, 2022.

[57] V. AI, "Voyage-3 embedding model," https://docs.voyageai.com/docs/embeddings, 2024, accessed May 2025. Proprietary embedding model optimized for search and classification.

[58] F. Barbieri, L. Espinosa Anke, and J. Camacho-Collados, "XLM-T: Multilingual language models in Twitter for sentiment analysis and beyond," in *Proceedings of the Thirteenth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, Jun. 2022, pp. 258–266. [Online]. Available: https://aclanthology.org/2022.lrec-1.27

[59] A. Aich, "Elastic weight consolidation (ewc): Nuts and bolts," 2021. [Online]. Available: https://arxiv.org/abs/2105.04093

[60] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 3987–3995. [Online]. Available: https://proceedings.mlr.press/v70/zenke17a.html

[61] S. M. Rump, "Fast and parallel interval arithmetic," *BIT Numerical Mathematics*, vol. 39, pp. 534–554, 1999.

[62] M. Wicker, A. Patane, L. Laurenti, and M. Kwiatkowska, "Adversarial robustness certification for bayesian neural networks," *arXiv preprint arXiv:2306.13614*, 2023.

[63] M. Wicker, L. Laurenti, A. Patane, N. Paoletti, A. Abate, and M. Kwiatkowska, "Certification of iterative predictions in bayesian neural networks," in *Uncertainty in Artificial Intelligence*. PMLR, 2021, pp. 1713–1723.

[64] H. Dang, M. R. Wicker, G. Botterweck, and A. Patane, "Certifiably quantisation-robust training and inference of neural networks," in *The 28th International Conference on Artificial Intelligence and Statistics*, 2025.

[65] P. Krukowski, A. Bielawska, K. Książek, P. Wawrzyński, P. Batorski, and P. Spurek, "Hint: Hypernetwork approach to training weight interval regions in continual learning," *Information Sciences*, p. 122261, 2025.

APPENDIX

A. Additional Related Works

In this section, we expand upon the related works provided in the main text to discuss provable guarantees that are related to this work. Firstly, we highlight that guarantees we leverage, in the interval case, build on the algorithm proposed in [61]. This algorithm found use in neural network architectures with Bayesian neural networks [45], [62] and has subsequently been used in explainability [36], control [63], quantization robustness [64], and in the initial work on continual learning [24]. In the initial work, [24], the authors focus particularly on continual learning application and spend considerable time discussing the problem of intersecting safe regions and its complexity. In this paper, on the other hand, we focus primarily on a principled approach to computing a locally invariant domain. Thus, our problem formulation is slightly different and our perspective on the problem leads us to explore some practically important aspects not covered in [24] including the use of buffer data and providing analysis of finite-sample guarantees for our method. Within our main text we reference the numerical instability of the approach given by [24], a symptom of which can be observed in their appendix where they use a learning rate hyperparameter that ranges in magnitude from $1e3$ to $1e-20$. This is due to the fact that interval bound propagation can be very loose for even moderate networks, thus as the problem scales to medium networks (CNNs even), the approach of [24] will need to use vanishingly small learning rates. Similar to ICN, the approach provided in [65] attempts to improve the computation of LIDs using hypernetworks. We do not compare with this method as the proposal is not strictly sound due to the lack of hard constraints when updating hypernetwork weights.

B. Interval Bound Propagation

This section details the interval bound propagation (IBP) procedure we use to compute a sound over-approximation of the specification function required for our LID computation. For clarity, we focus on feed-forward networks with monotonic activations, though the approach generalizes straightforwardly to other architectures such as convolutional networks.

Feedforward Neural Networks. We consider a neural network model $f^\theta : \mathbb{R}^{n_0} \to \mathbb{R}^{n_K}$ with parameters $\theta = \{(W_i, b_i)\}_{i=1}^K$ composed of K layers. Each layer performs the following operations:

$$\hat{z}_k = W_k z_{k-1} + b_k, \qquad z_k = \sigma(\hat{z}_k) \tag{7}$$

where $z_0 := x$, $f^\theta(x) := \hat{z}_K$, and σ is the activation function, which we assume to be monotonic (e.g., ReLU).

Interval Arithmetic. We now define interval representations for matrices, and their corresponding arithmetic operations. We represent each matrix A by its element-wise interval bounds $[A^L, A^U]$, where A^L and A^U denote the lower and upper bounds respectively. The key operations required for neural network forward passes are interval matrix addition and multiplication.

Given interval matrices $[A^L, A^U]$ and $[B^L, B^U]$, we define: leftmargin=*

- *Interval Matrix Addition:* $[A^L, A^U] \oplus [B^L, B^U] = [A^L + B^L, A^U + B^U]$
- *Interval Matrix Multiplication:* Using center matrices $A_\mu = (A^U + A^L)/2$, $B_\mu = (B^U + B^L)/2$ and radius matrices $A_r = (A^U - A^L)/2$, $B_r = (B^U - B^L)/2$, Rump's algorithm gives $[A^L, A^U] \otimes [B^L, B^U] = [C^L, C^U]$ where

$$C^L = A_\mu B_\mu - |A_\mu| B_r - A_r |B_\mu| - A_r B_r$$
$$C^U = A_\mu B_\mu + |A_\mu| B_r + A_r |B_\mu| + A_r B_r$$

These operations ensure $A' \oplus B' \in [C^L, C^U]$ and $A' \otimes B' \in [C^L, C^U]$ for all $A' \in [A^L, A^U]$, $B' \in [B^L, B^U]$. Both operations incur modest computational costs compared with their non-interval counterparts ($2\times$ for addition and $4\times$ for multiplication).

Bounding the Forward Pass. Using the interval operations defined above, we can propagate bounds through the neural network. For any input x and parameters $W_k \in [W_k^L, W_k^U]$, $b_k \in [b_k^L, b_k^U]$ for $k = 1, \ldots, K$, we compute interval bounds:

$$[\hat{z}_k^L, \hat{z}_k^U] = [W_k^L, W_k^U] \otimes [z_{k-1}^L, z_{k-1}^U] \oplus [b_k^L, b_k^U]$$
$$[z_k^L, z_k^U] = [\sigma(\hat{z}_k^L), \sigma(\hat{z}_k^U)]$$

This procedure guarantees that $f^\theta(x) \in [\hat{z}_K^L, \hat{z}_K^U]$ for all valid parameter combinations within their respective intervals.

For the monotonic activation function σ, we can apply the function element-wise to both the lower and upper bounds of the input interval to obtain a valid output interval, preserving the soundness of the over-approximation throughout the forward pass.

Bounding the Specification. The procedure for propagating intervals through the specification function depends on the exact choice of specification function. We assume we have the interval bounds on the output logits of the network, $[\hat{z}_K^L, \hat{z}_K^U]$, computed using the procedure above. Let y denote the true class index.

Standard performance metrics, such as model accuracy, are monotonically increasing in the logit of the correct class, and monotonically decreasing in the logits of all other classes. Thus, the 'worst-case' output logits within the interval $[\hat{z}_K^L, \hat{z}_K^U]$ are given by

$$\hat{z}_K^{\text{worst}} = \hat{z}_K^L \cdot \mathbf{e}_y + \hat{z}_K^U \cdot (1 - \mathbf{e}_y), \tag{8}$$

where $\mathbf{e}_y \in \mathbb{R}^C$ is the one-hot vector corresponding to the true class y.

Thus, propagating interval bounds through common performance metrics corresponds to evaluating this worst-case output:

- *Cross-Entropy Loss:*

$$\phi_{\text{ce}}(\theta, x, y) = -\log\left(\text{softmax}(f^\theta(x))_y\right)$$
$$\leq \phi_{\text{ce}}^{\text{IBP}} = -\log\left(\text{softmax}(\hat{z}_K^{\text{worst}})_y\right),$$

where the inequality reflects that $\phi_{\text{acc}}^{\text{IBP}}$ is a certified upper bound on the true loss for any θ in our interval domain.

- *(Negative) Accuracy:*

$$\begin{aligned}\phi_{\text{acc}}(\theta, x, y) &= -\mathbb{I}\left(\arg\max_i\{f^\theta(x)_i\} = y\right) \\ &\leq \phi_{\text{acc}}^{\text{IBP}} = -\mathbb{I}\left(\arg\max_i\{\hat{z}_{K,i}^{\text{worst}}\} = y\right),\end{aligned}$$

  where the inequality reflects that $\phi_{\text{acc}}^{\text{IBP}}$ represents a lower bound on the accuracy.
- *(Negative) Soft Accuracy:*

$$\begin{aligned}\phi_{\text{soft acc}}(\theta, x, y) &= -\operatorname{softmax}(f^\theta(x))_y \\ &\leq \phi_{\text{soft acc}}^{\text{IBP}} = -\operatorname{softmax}(\hat{z}_K^{\text{worst}})_y,\end{aligned}$$

  which represents a certified lower bound on the soft accuracy of the model. We include the soft accuracy as a differentiable proxy of a hard accuracy specification in the primal-dual computation of LIDs.

Since our definition of LID requires ϕ to be bounded from above, we define the specification for accuracy and soft accuracy to be the negation of the nominal value.

C. Heuristics for Effective LID Computation

De-bias regularisation One observation from our first experiments in the CIL setting was that the model trained on the first task is heavily biased towards the first classes (i.e. the network will result in a very confident prediction of a class from task 1 when confronted with data points from task 2). This means that there might be an important distance in parameter space that needs to be travelled before the model can start predicting other output classes, making future learning challenging within our current local invariant domain. Thus, we added a regularisation term that passes noise through the model and penalises over-confidence on any particular output. Here is a formal description of our method: Let F be a noisy distribution over the input space $\mathcal{X} \in \mathbb{R}^n$. F can be taken to be gaussian, uniform or anything else. We show below which distributions worked well in practice. We denote our machine learning model as parameterized functions $f^\theta : \mathbb{R}^n \to \mathcal{Y}$ with parameters $\theta \in \mathbb{R}^p$. In practice this function can be further expressed as follow:

$$f^\theta(x) = \arg\max_i h^\theta(x)$$

where $h^\theta : \mathcal{X} \to \mathbb{R}^{\mathcal{C}}$ maps the input to their corresponding logits in $\mathbb{R}^C$ with C being the number of classes we consider. The de-bias regularisation term $U(\theta)$ is expressed as follow :

$$U(\theta) = \frac{1}{m}\sum_{k=0}^{m} \|\mathrm{s}(h^\theta(X_k)) - \bar{\mathrm{s}}(h^\theta(X_k))\|_2$$
$$X_1, ..., X_m \overset{\text{i.i.d.}}{\sim} F$$

where s is the softmax function and $\bar{\mathrm{s}}$ is the coordinate-wise mean softmax function. This encourages the predicted probabilities of each classes to stay close to each other for noisy samples. During gradient descent the de-bias term $U(\theta)$ is added to the overall loss multiplied by a regularisation factor λ.

Choosing the right distribution F is decisive in ensuring that the neural network stays unbiased when confronted with new samples. Note that F should be sufficiently distinct from the task distribution $\mathcal{D}_t$, otherwise learning task t may effectively be compromised. In addition, F should stay general enough to represent all potential inputs we might be confronted with in the future. In our experiments we choose F to be uniform over the minimum and maximum value coordinate wise we encounter in our dataset $\mathcal{D} := \{(x^{(i)}, y^{(i)}\}_{i=1}^N$.

D. Experimental Details and Additional Ablation Studies

In this chapter we explore the experimental setups further. All of the three main experiments follow similar testing and evaluation pipelines:

Split MNIST. As mentioned in section V-A we use a two layer CNN follow by a single fully connected classification head. We first train this neural network on an initial task – task details are depending on the specific continual learning setup – then compute a LID for the given, trained model. Subsequently, we use projected gradient descent to train the model on each of the following tasks within the constraints provided by the LID. In-between tasks, we continue to compute LIDs for the most recently learned task, taking intersections between the existing LID and the newly computed LID to produce guarantees not only for the most recent task, but for all previous tasks. This procedure results in algorithm 1:

Algorithm 1 Certified Continual Learning with Locally Invariant Domains

Input: f - NN model, θ' - initial parameter, E - # of epochs, ϕ - specification function, α - learning rate, δ - minimum accuracy.
Output: θ - Param. with acceptable error, t - number of tasks completed

$\mathcal{D}_1 = \{(x^{(i)}, y^{(i)})\}_{i=1}^k \sim P_1(X, Y)$
$\theta_1 \leftarrow \texttt{SGD}(\theta', \mathcal{D}_1, E, \alpha)$
$[\theta_1^L, \theta_1^U] \leftarrow \texttt{LID}(\theta, \mathcal{D}_1, \phi, \delta)$
for $j \in 2, ..., t$ **do**
  $\mathcal{D}_j = \{(x^{(i)}, y^{(i)})\}_{i=1}^k \sim P_j(X, Y)$
  $\theta_j \leftarrow \texttt{PGD}(\theta_{j-1}, \mathcal{D}_j, E, \alpha, [\theta_{j-1}^L, \theta_{j-1}^U])$
  $[\theta_j^L, \theta_j^U] \leftarrow \texttt{LID}(\theta, \mathcal{D}_j, \phi, \delta)$
  $[\theta_j^L, \theta_j^U] \leftarrow [\theta_j^L, \theta_j^U] \cap [\theta_{j-1}^L, \theta_{j-1}^U]$
  if $[\theta_1^L, \theta_1^U] == \emptyset$ **then**
    return θ_j
  end if
end for
return θ_t

To split MNIST into separate tasks we randomly separate the ten digits from MNIST into five tasks of two digits each, while ensuring that each of the pairs is a combination of an odd and an even number to make sure that the tasks are well defined for domain IL.

Split CIFAR. For split CIFAR we also follow algorithm 1 with the only difference being the task creation. Here, we hand pick well defined tasks to create the following splits: *cat*

TABLE II
EXPERIMENTAL HYPERPARAMETERS FOR SPLIT-MNIST AND SPLIT-CIFAR.

Category	**Hyperparam**	Split-MNIST *(TIL/DIL/CIL)*	Split-CIFAR *(TIL/DIL/CIL)*
Learning	Learning Rate	0.001	0.02
	Batch Size	64	128
	Number of Epochs	5	5
	ℓ_2 Regularisation	0.01	0.01
	De-bias Regularisation	0.01	0.01
	Projection strategy	sample_largest_closest	best loss
LID Optimization	Primal Learning Rate (η_p)	0.33	0.33
	Dual Learning Rate (η_d)	0.01	0.01
	Batch Size	400	400
	Number of Iterations	200	200
	LID Saving Period	20	2
Baselines	LwF λ	0.001/0.05/0.05	0.1
	EWC λ	1e6	1e6
InterContiNet	Learning Rate	0.01	0.01
	Batch Size	1e6	128
	Epochs	15/5/5	30
	Radii lr	100/1000/1	1/1/0.1

TABLE III
EXPERIMENTAL HYPERPARAMETERS FOR LLM EXPERIMENTS

Category	**Hyperparam**	Default	M2-BERT	MXBAI
Learning	Learning Rate	0.0005	0.0005	0.0005
	Batch Size	64	64	64
	Number of Epochs	1	1	1
	ℓ_2 Regularisation	0	0.01	0.01
	De-bias Regularisation	0	0	0
	Projection strategy	best loss	best loss	best loss
LID Optimization	Primal Learning Rate (η_p)	0.1	0.03	0.03
	Dual Learning Rate (η_d)	0.1	0.001	0.001
	Batch Size	1024	1024	1024
	Number of Iterations	700	700	700
	LID Saving Period	20	20	20

vs *truck*, *frog* vs *ship*, *horse* vs *automobile*, and *dog* vs *airplane*. To make the tasks compatible with domain incremental learning, note that each of the tasks uses an animal and a mode of transportation - two separate domains. Another slight difference for CIFAR is that we used a ResNet18 model and then train a MLP classifier on the embeddings obtained from the ResNet. We restrict our LID computations and continual learning to the MLP only as opposed to the entire model pipeline as is the case for the MNIST setup.

E. Foundation Model Hyperparameters

We present the hyperparameters used in Table III.

F. Buffer-based Learning Scheme

The algorithm for the buffer-based case is reasonably similar to the procedure described in algorithm 1. The main difference is that instead of exiting after achieving the maximum accuracy within a given LID, we check if the achieved performance exceeds a certain target performance. If it does, we proceed to the next task, if it doesn't, we recompute the LID using samples stored in a buffer. When updating, we draw $k = 200$ samples of each task from the buffer and use the combined data to compute an updated LID. This updated LID, now centered at the most recent set of model parameters, θ, ought to allow for further increases in performance on the current task, while still maintaining a safe-area with guarantees. This results in algorithm 2.

The hyperparameters for the buffer algorithm are largely the same as those shown in table II with the only addition being the aforementioned k, the target accuracy, and the maximum buffer calls per task shown in table IV

G. Empirical Computational Analysis

To discuss the introduced computational overhead of providing safe updates we train each method found in Table VI on two separate tasks of Split MNIST with an interleaved safe space computation for our method and ICN. Across 50 runs we report the mean time taken and one standard deviation of variance in seconds. It becomes apparent that our method does introduce some overhead compared to methods that do not compute safe spaces. While there is some additional delta due to the projection step during projected gradient descent, most of the overhead comes from the one-off step of the LID computation.

TABLE IV
EXPERIMENTAL HYPERPARAMETERS FOR SPLIT-MNIST AND SPLIT-CIFAR FOR OUR BUFFER-BASED ALGORITHM.

Category	**Hyperparam**	Split-MNIST *(TIL/DIL/CIL)*	Split-CIFAR *(TIL/DIL/CIL)*
LID Optimization	Target Accuracy	0.65/0.65/0.65	0.65/0.75/0.55
	Max Buffer Calls	1/3/7	1/3/7
Baselines	A-GEM memory size	3750	3750

TABLE V
EXPERIMENTAL HYPERPARAMETERS FOR WEIGHT PRUNING EXPERIMENTS. THIS ACTS AS AN EXTENSION TO LLM AND SPLIT-MNIST EXPERIMENTS, HENCE THE BASELINE USES HYPERPARAMETERS SPECIFIED IN TABLES II AND III UNLESS MENTIONED OTHERWISE.

Category	**Hyperparam**	Split-MNIST	LLM
LID Optimization	Number of Iterations	300	300
	LID Saving Period	100	100
	Freezing Proportion - Lookahead	0.825	0.85
	Freezing Proportion	0.05	0.05

TABLE VI
EMPIRICAL ANALYSIS OF COMPUTATIONAL COST OF DIFFERENT CONTINUAL LEARNING METHODS IN SECONDS AND ONE STANDARD DEVIATION.

Method	**Value [s]**
Ours	43.18 ± 2.19
of which LID	12.92 ± 0.79
ICN	70.09 ± 2.88
of which LID	51.81 ± 2.13
SGD	16.08 ± 0.82
EWC	15.17 ± 1.49
A-GEM	20.29 ± 0.89
LwF	14.96 ± 0.80

Algorithm 2 Certified Continual Learning with Locally Invariant Domains and Buffers

Input: f - NN model, θ' - initial parameter, E - # of epochs, ϕ - specification function, α - learning rate, δ - minimum accuracy, $target\_acc$ - target accuracy, M - buffer, b - buffer size, acc - accuracy metric, max_calls - maximum number of buffer calls per task.

Output: θ - Param. with acceptable error, t - number of tasks completed

$\mathcal{D}_1 = \{(x^{(i)}, y^{(i)})\}_{i=1}^{k} \sim P_1(X, Y)$
$\theta_1 \leftarrow \texttt{SGD}(\theta', \mathcal{D}_1, E, \alpha)$
$[\theta_1^L, \theta_1^U] \leftarrow \texttt{LID}(\theta, \mathcal{D}_1, \phi, \delta)$
$\mathcal{D}_1^* = \{(x^{(i)}, y^{(i)})\}_{i=1}^{b_i} \sim P_1(X, Y)$
$M_1 \leftarrow D_1$
for $j \in 2, ..., t$ **do**
  $\mathcal{D}_j = \{(x^{(i)}, y^{(i)})\}_{i=1}^{k} \sim P_j(X, Y)$
  $\theta_j \leftarrow \texttt{PGD}(\theta_{j-1}, \mathcal{D}_j, E, \alpha, [\theta_{j-1}^L, \theta_{j-1}^U])$
  $[\theta_j^L, \theta_j^U] \leftarrow \texttt{LID}(\theta, \mathcal{D}_j, \phi, \delta)$
  $[\theta_j^L, \theta_j^U] \leftarrow [\theta_j^L, \theta_j^U] \cap [\theta_{j-1}^L, \theta_{j-1}^U]$
  $curr\_acc = acc(\theta, D_j)$
  while $n < max\_calls$ & $not\ empty(M_{i:j-1})$ & $target\_acc > curr\_acc$ **do**
    $\mathcal{D}_{1:j-1} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{k} \sim M_{1:j-1}$
    $[\theta_{j-1}^L, \theta_{j-1}^U] \leftarrow \texttt{LID}(\mathcal{D}_{1:j-1}, \phi, \delta)$
    $\theta_j \leftarrow \texttt{PGD}(\theta_j, \mathcal{D}_j, E, \alpha, [\theta_{j-1}^L, \theta_{j-1}^U])$
    $curr\_acc = acc(\theta, D_j)$
  end while
  $D_j^* = \{(x^{(i)}, y^{(i)})\}_{i=1}^{b_i} \sim P_j(X, Y)$
  $M_j \leftarrow D_j^*$
  if $[\theta_1^L, \theta_1^U] == \emptyset$ **then**
    return θ_j
  end if
end for
return θ_t