SHIELD: Secure Hypernetworks for Incremental Expansion Learning Defense

Patryk Krukowski
Jagiellonian University
IDEAS NCBR
patryk.krukowski@doctoral.uj.edu.pl

Łukasz Gorczyca
Jagiellonian University

Piotr Helm
Jagiellonian University

Kamil Książek
Jagiellonian University

Przemysław Spurek
Jagiellonian University
IDEAS Research Institute

Abstract

Traditional deep neural networks suffer from several limitations, including catastrophic forgetting. When models are adapted to new datasets, they tend to quickly forget previously learned knowledge. Another significant issue is the lack of robustness to even small perturbations in the input data. In practice, we can often easily perform adversarial attacks and change the network's predictions, adding minimal noise to the input. Dedicated architectures and training procedures can solve each of the above problems separately. Unfortunately, currently, no model can simultaneously address both catastrophic forgetting and vulnerability to adversarial attacks. We introduce SHIELD (Secure Hypernetworks for Incremental Expansion and Learning Defense) [1], a novel approach that integrates a hypernetwork-based continual learning approach with interval arithmetic. SHIELD use the hypernetwork to transfer trainable task embedding vectors into the weights of a target model dedicated to specific data. This paradigm allows for the dynamic generation of separate networks for each subtask, while the hypernetwork aggregates and analyzes information across all tasks. The target model takes in the input a data sample with a defined interval range, and by creating a hypercube, produces a prediction for the given range. Therefore, such target models provide strict guarantees against all possible attacks for data samples within the interval range. Our approach enhances security without sacrificing network adaptability, addressing the overlooked challenge of safety in continual learning.

1 Introduction

Deep neural networks can provide outstanding predictions and often surpass human experts in multiple scoring tasks [15]. Nonetheless, they still face several limitations. Learning from a continuous stream of data presents substantial challenges for deep learning models. Artificial neural networks tend to experience catastrophic forgetting [33], where they quickly lose previously learned knowledge upon encountering new data. Continual learning (CL) strategies [17] are developed to enable efficient learning of sequential tasks while reducing the chance of forgetting older information. Another essential problem is the robustness of such models [44]. Although deep learning models typically handle random noise effectively, their performance can notably deteriorate when encountering adversarial perturbations. These perturbations involve slight modifications of the input data that

[1] The source code will be available soon.

Preprint. Under review.

remain unnoticed by humans but lead the model to make incorrect predictions [44, 11]. Such effects present challenges for applications such as autonomous vehicles or financial systems, where security is paramount [42].

Continual learning and adversarial robustness are among the most critical problems in deep learning. Although many different methods are dedicated to each of these tasks, there is a lack of a methodology that can solve both simultaneously. Only a few approaches are devoted to both issues in the existing literature. In AIR [52], for the first time, the authors introduced the concept of continual adversarial defense under a sequence of attacks. AIR uses unsupervised data augmentation to increase robustness in the continual learning process. Such a solution is agnostic to the adversarial attack and can be used for general robustness. The main limitation of the model is that it works with a limited number of subsequent continual adversarial defense tasks. In the AIR paper, the authors show results on only two or three such tasks. On the other hand, in Double Gradient Projection (DGP) [38], authors emphasized that their approach is more robust to adversarial attacks than classical CL methods like GEM [29] or GPM [40]. DGP, before weight updates, propagates gradients into the orthogonal direction to the subspace identified as crucial, preserving the smoothness of previous sample gradients, which holds the adversarial robustness. In [22], the authors verify the robustness of existing models to adversarial attacks.

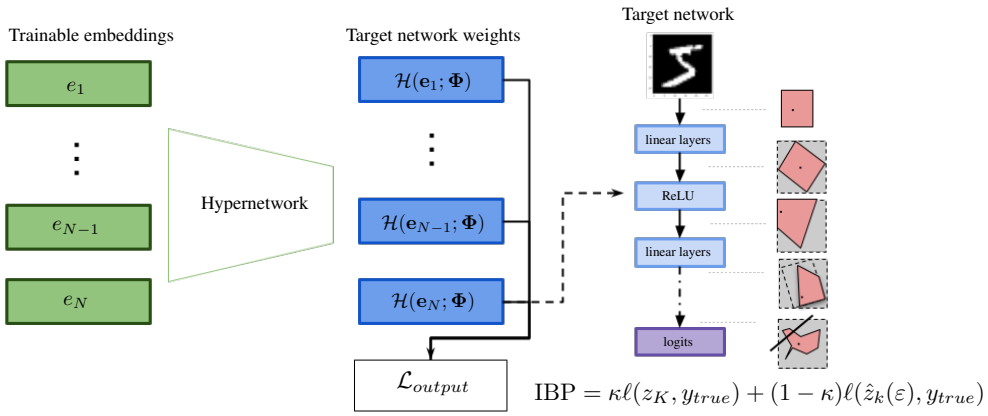

Figure 1: Commonly, the parameters of a neural network are directly adjusted from data to solve a task. In SHIELD, hypernetwork maps task-dedicated embedding vectors e_i, $i \in \{1, ..., N\}$, to the target model, which propagates intervals instead of points. Therefore, SHIELD is able to learn knowledge from multiple tasks and defend from adversarial attacks within the specified range.

One of the most promising strategies for continual learning is the hypernetwork approach [16, 45, 25, 24]. Essentially, a hypernetwork architecture [13] is a neural network whose purpose is to create weights for a different target network designed to handle a particular task. Within a continual learning framework, the hypernetwork formulates the weights for a target model by identifying the given task. After finishing training for all CL tasks, a single meta-model emerges, generating task-specific weights according to the considered task. This capability of producing distinctly different weights for each task ensures that hypernetwork-based models experience minimal forgetting. Although a hypernetwork-based approach addresses the issue of catastrophic forgetting, it lacks robustness against adversarial attacks.

On the other hand, Interval Bound Propagation (IBP) is emerging as a promising methodology to improve adversarial robustness [12, 34]. This approach utilizes interval arithmetic to propagate input data samples in terms of points and corresponding radii, creating intervals. When applied to classification tasks, it ensures that all points within the input interval produce the same prediction. Consequently, it enables explicit control over the neural network prediction behaviour within a specified neighborhood of the input data. In practice, IBP gives strict guarantees of robustness for points within learned intervals.

In our paper, we propose a novel approach that combines two paradigms for continual learning: hypernetworks [13] and interval arithmetic networks guaranteeing robustness for adversarial attacks [12, 34]. We demonstrate that our model also inherits resistance to catastrophic forgetting.

The following is a concise summary of our contribution:

- We propose SHIELD, which is a model dedicated to continual learning tasks, maintaining robustness against adversarial attacks.
- SHIELD use both the hypernetwork paradigm and Interval Bound Propagation in one simple and elegant architecture, retaining benefits of both concepts.
- SHIELD is attack-agnostic and gives strict guarantees of robustness against attacks inside specific intervals around initial samples.
- We achieve state-of-the-art results for several CL datasets and different variants of adversarial attacks.

2 Related Works

Our method combines two deep learning paradigms: continual learning and adversarial examples. Therefore, our related work section is divided into two parts.

Adversarial Attack Adversarial attacks on deep neural networks (DNNs) are typically categorized as white-box or black-box, depending on the attacker's access to the target model. In white-box settings, the adversary fully knows the model, including its architecture, parameters, and gradients. Prominent white-box attacks include gradient-based methods such as FGSM [11], BIM [18], MIM [8], PGD [30], and AutoAttack (AA) [7], as well as optimization-based approaches like the OnePixel [43].

In black-box settings, attackers have limited access to the model and must rely on indirect signals. Some approaches use confidence scores to estimate gradients, such as zoo [6], while others operate with only the predicted label, as seen in decision-based methods like the Boundary Attack [4]. Another strategy involves crafting adversarial examples on a surrogate model and transferring them to the target model, exploiting the cross-model transferability of adversarial perturbations [8, 9].

Simultaneously with adversarial attack techniques, there are many defence algorithms. Early adversarial defenses relied on heuristics such as input transformations [48], model ensembles [3], and denoisers [21], but many were later found to rely on obfuscated gradients [2]. More robust approaches like adversarial training (AT) [11, 20] and defensive distillation [10, 53] have become dominant. Recent work has enhanced AT with learnable strategies [20], feature perturbation [36], and optimization techniques [37]. However, the above modls are typically static and struggle against evolving attack sequences. Test-Time Adaptation Defense (TTAD) [41, 49] addresses adaptation to new attacks but neglects previous ones.

Most of the existing model gives robustness without strict guarantees. Interval Bound Propagation (IBP) [12, 34] interval arithmetic produced robust neural networks against adversarial attacks with strict guarantees. The authors used worst-case cross-entropy to train the classification model in an approach. In our work, we use an IBP-based solution in continual learning mode.

Continual Learning Continual learning (CL) methods are often grouped into architectural, rehearsal, and regularization-based strategies [17]. Architectural approaches like Progressive Neural Networks (PNN) [39] and CopyWeights with Re-init (CWR) [28] prevent forgetting by expanding or modifying network structures. Further developments reuse a fixed architecture with iterative pruning, such as PackNet [31] or Supermasks [47]. Rehearsal-based methods, such as the buffer strategy in [14], retain selected past examples, while pseudo-rehearsal approaches train generative models to replay prior data [32]. Regularization-based techniques like EWC [23], SI [50], LwF [26], and MAS [1] constrain parameter updates based on past task

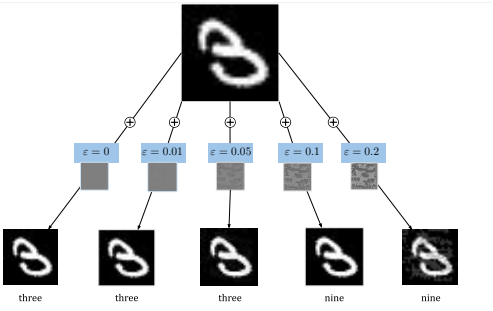

Figure 2: Impact of FGSM adversarial attack on MNIST three-digit image. Model trained with $\varepsilon = 0.05$ successfully classifies adversarial examples generated with strength $\leq \varepsilon$. For greater attack strength, the image is misclassified as nine, even when the visual perturbations are minimal and hard to notice.

importance, often using the Fisher Information Matrix or similar metrics.

Hypernetworks, introduced in [13], are defined as neural models that generate weights for a separate target network that solves a specific task. The authors aim to reduce the number of trainable parameters by designing a hypernetwork that often has fewer parameters than the target network. In CL, hypernetworks can directly generate individual weights for subsequent CL tasks, like in HNET [45]. Then, HNET was extended with a probabilistic approach called posterior meta-replay [16]. In HyperMask [25], authors used the lottery ticket hypothesis to produce task-specific masks for the trainable or fixed target model. In [24], the authors employ interval arithmetic on weights to regularize the target model. These methods are positioned between those based on architecture and those focused on regularization.

The above concepts have many different modifications and upgrades [46]. But the adversarial robustness of Continual Learning problems is unexplored in the literature. AIR [52] introduced the first framework for continual adversarial defense, addressing robustness against evolving attack sequences. The method leverages unsupervised data augmentation to enhance general robustness in a task-agnostic manner. While innovative, AIR is limited in scalability, having been evaluated on only two to three defense tasks, raising concerns about its effectiveness in longer, continual settings.

Alternatively, the study on Double Gradient Projection (DGP) [38] highlights that this technique is notably more resilient to adversarial attacks compared to traditional continual learning approaches such as GEM [29] and GPM [40]. DGP enhances adversarial robustness by projecting gradients orthogonally concerning a critical subspace before updating weights, thus maintaining the prior gradient smoothness from earlier samples. Furthermore, [22] examines the resistance of current models against adversarial assaults. These methodologies enhance the robustness of continual learning, yet they have several significant limitations. Firstly, such approaches are restricted to several subsequent tasks and do not give strict guarantees. In this paper, we present SHIELD – a natural combination of Hypernetworks and IBP models, which inherit adversarial robustness and prevent catastrophic forgetting.

3 SHIELD: Secure Hypernetworks for Incremental Expansion Learning Defense

In this section, we introduce SHIELD model dedicated to continual adversarial defense against a sequence of attacks. SHIELD incorporates two primary components: a hypernetwork and a target network utilizing interval arithmetic.

3.1 SHIELD architecture

Hypernetwork Our models use trainable embeddings $\boldsymbol{e}_t \in \mathbb{R}^N$, where $t \in \{1, ..., T\}$, which represent consecutive tasks. These vectors are the input to the hypernetwork, producing the target network weights for specific datasets. More specifically, the hypernetwork $\mathcal{H}$ with weights $\boldsymbol{\Phi}$ generates weights $\boldsymbol{\theta}_t$ of the target network f, dedicated to the t–th task, i.e. $\mathcal{H}(\boldsymbol{e}_t; \boldsymbol{\Phi}) = \boldsymbol{\theta}_t$. Therefore, the target network is not trained directly, and the meta-architecture generates distinct weights for each continual learning task. The function $f_{\boldsymbol{\theta}_t} : X \to Y$ predicts labels Y for data samples X based on weights $\boldsymbol{\theta}_t$. Finally, each continual learning task is represented by a classifier function

$$f_{\boldsymbol{\theta}_t} = f(\cdot; \boldsymbol{\theta}_t) = f\big(\cdot; H(\boldsymbol{e}_t; \boldsymbol{\Phi})\big). \tag{1}$$

At the end of training, we have a single meta-model, which produces task-dedicated weights. Due to the ability to generate completely different weights for each task, hypernetwork-based models feature minimal forgetting.

Target network In our solution, the target network is designed for interval representation of input points instead of single real values. In practice, we use the IBP approach [12, 35] in the target model. This means that we do not use a single-valued point $x \in \mathbb{R}^d$ as input to the target model $f_{\boldsymbol{\theta}_t}$, but a point x with fixed radius ε

$$I_\varepsilon(x) = [x - \varepsilon, x + \varepsilon] = [x_1 - \varepsilon, x_1 + \varepsilon] \times ... \times [x_d - \varepsilon, x_d + \varepsilon]. \tag{2}$$

Such intervals are an d-dimensional hypercube centered at x with side length 2ε. When we consider a linear target model $f_{\boldsymbol{\theta}_t}$, it is composed of K layers given by K transformations

$$z_k = h_k(z_{k-1}) = W_k z_{k-1} + b_k, \text{ for } k = 1, \ldots, K. \tag{3}$$

In practice, h_k represents either an affine transformation or a non-linear, non-decreasing function like ReLU or sigmoid, while we utilize the former. In the IBP approach [12], in each layer, we find the smallest bounding box that encloses the transformed bounding box returned by the previous layer. In other words, we bound the activation z_k of the k–th layer by an axis-aligned bounding box

$$I_\varepsilon(z_k) = [\underline{z}_k, \bar{z}_k] = [\underline{z}_{k,1}, \bar{z}_{k,1}] \times \ldots \times [\underline{z}_{k,d_k}, \bar{z}_{k,d_k}], \tag{4}$$

where d_k is the output dimension of the k–th layer. In the case of neural networks, computing such a bounding box for consecutive layers can be done efficiently using interval arithmetic. By applying the affine transformation $h_k(z_{k-1}) = W_k z_{k-1} + b_k$ to $[\underline{z}_{k-1}, \bar{z}_{k-1}]$, the smallest bounding box $[\underline{z}_k, \bar{z}_k]$ is represented by

$$\mu_{k-1} = \frac{\bar{z}_{k-1} + \underline{z}_{k-1}}{2}, \quad r_{k-1} = \frac{\bar{z}_{k-1} - \underline{z}_{k-1}}{2}, \tag{5}$$

$$\mu_k = W_k \mu_{k-1} + b_{k-1}, \quad r_k = |W_k| r_{k-1}, \tag{6}$$

$$\underline{z}_k^\star = \mu_k - r_k, \quad \bar{z}_k^\star = \mu_k + r_k, \tag{7}$$

where $|\cdot|$ is an element-wise absolute value operator. For a non-decreasing activation function $g(\cdot)$, we get the interval bound defined by

$$\underline{z}_k = g(\underline{z}_k^\star), \quad \bar{z}_k = g(\bar{z}_k^\star). \tag{8}$$

To obtain a robust classification, during training, we consider the worst-case loss function (see Eq. 10) for the entire interval bound $[\underline{z}_K, \bar{z}_K]$ of the final logits. During inference, the class proposed by the model, $\hat{y}$, is represented by the largest interval center after using the softmax function:

$$\hat{y} = \underset{i \in \{1,\ldots,C\}}{\arg\max} \left(\operatorname{softmax} \left(\frac{\underline{z}_K + \bar{z}_K}{2} \right) \right). \tag{9}$$

We want to perform a correct classification not only for the input sample x, but for all points $\hat{x}$ within the interval, i.e. for $\hat{x} \in [x - \varepsilon, x + \varepsilon]$.

Ultimately, the model comprises a learnable embedding and a hypernetwork designed to create the IBP-based target model for a given task. Therefore, SHIELD has improved robustness against adversarial attacks.

3.2 SHIELD cost function

One of the most essential parts of our model is the cost function, which prevents catastrophic forgetting and provides resilience against adversarial attacks. We selected the worst-case loss function over the interval $[\underline{z}_K, \bar{z}_K]$ for the final logits. Specifically, it is necessary to verify that the entire bounding box is classified accurately, which means that certain input perturbations do not alter the correct class prediction. Therefore, the logit for the true class y_{true} corresponds to its lowest bound, while the logits for the remaining classes match their highest bounds:

$$\hat{z}_{K,\hat{y}}(\varepsilon) = \begin{cases} \bar{z}_{K,\hat{y}}(\varepsilon), & \text{for } \hat{y} \neq y_{true}, \\ \underline{z}_{K,y_{true}}(\varepsilon), & \text{otherwise.} \end{cases} \tag{10}$$

This means that the individual coordinates of $\hat{z}_{K,\hat{y}}(\varepsilon)$ are filled with the most pessimistic interval-based prediction values. Finally, the softmax function and the cross-entropy loss are applied to $\hat{z}_{K,\hat{y}}(\varepsilon)$. As demonstrated in [12], determining interval bounds requires just two forward passes through the neural network, which makes this method practically advantageous. The authors of [12] proposed a loss function combining the typical cross-entropy classification loss with a component forcing the interval boundaries for wrongly predicted classes, utilizing Eq. 10:

$$\text{IBP} = \kappa \cdot \ell(\hat{y}, y_{true}) + (1 - \kappa) \cdot \ell(\hat{z}_{K,\hat{y}}(\varepsilon), y_{true}), \tag{11}$$

where ℓ represents the cross-entropy loss and κ is a trade-off parameter. When $\varepsilon = 0$, $\hat{z}_{K,\hat{y}}(\varepsilon) = \hat{y}$, and Eq. 11 reduces to the typical cross-entropy loss for single real values.

Excessively wide intervals can adversely affect the stability of the training process, particularly in its initial stages. To address this, we conduct the initial training phase with $\kappa = 1$. Subsequently, the importance of the interval loss expressed by the second component of IBP progressively increases, until it reaches $\kappa = 1/2$ in half of the number of training epochs. At the same time, training begins with a minimal perturbation radius ε, and its value is incrementally increased in subsequent epochs. In the second half of training, they remain constant. Let us denote by ϵ_i perturbation during the i–th training epoch, where $i \in \{1, \ldots, E\}$, and E is the selected number of epochs for consecutive tasks. Also, assume that κ_i is the value of κ during the i-th epoch. Then, we calculate the importance of IBP components and perturbation radius according to equations:

$$\kappa_i = \begin{cases} 1 - \frac{i}{2E} & \text{if } i < \lfloor \frac{E}{2} \rfloor - 1, \\ 0.5 & \text{otherwise,} \end{cases} \quad \epsilon_i = \begin{cases} \frac{2i \cdot \epsilon}{E} & \text{if } i < \lfloor \frac{E}{2} \rfloor - 1, \\ \epsilon & \text{otherwise.} \end{cases} \tag{12}$$

Although ensuring adversarial robustness is possible under worst-case accuracy conditions, it is also essential to avoid catastrophic forgetting in the hypernetwork. Consequently, we incorporate a regularization term responsible for maintaining knowledge from previous CL tasks. Therefore, we can compare the fixed hypernetwork output (i.e. the target network weights) that was generated before learning the current task $t_c \in \{1, ..., T\}$, denoted by $\mathbf{\Phi}^*$, with the hypernetwork output after recent suggestions for its weight changes, $\mathbf{\Phi} + \Delta\mathbf{\Phi}$. Ultimately, in SHIELD, the formula for the regularization loss is presented as:

$$\mathcal{L}_{output}(\mathbf{\Phi}^*, \mathbf{\Phi}, \Delta\mathbf{\Phi}, \{\mathbf{e}_t\}, t_c) = \frac{1}{\max\{1, t_c - 1\}} \sum_{t=1}^{t_c - 1} \|\mathcal{H}(\mathbf{e}_t; \mathbf{\Phi}^*) - \mathcal{H}(\mathbf{e}_t; \mathbf{\Phi} + \Delta\mathbf{\Phi})\|^2. \tag{13}$$

The final cost function consists of the IBP loss and hypernetwork regularization term $\mathcal{L}_{output}$:

$$\mathcal{L} = \text{IBP} + \beta \cdot \mathcal{L}_{output} = \kappa \cdot \ell(\hat{y}, y_{true}) + (1 - \kappa) \cdot \ell(\hat{z}_{K,\hat{y}}(\varepsilon), y_{true}) + \beta \cdot \mathcal{L}_{output}, \tag{14}$$

where β and κ are hyperparameters that influence the hypernetwork stability and the worst-case accuracy.

3.3 Robustness guarantees of SHIELD

In this section, we provide theoretical insights into the certified robustness achieved via IBP in SHIELD. We first formalize certified robustness for standard feedforward networks, then extend the concept to the CL setting, and finally apply it to our proposed method.

Definition 3.1 (Certified Robustness via IBP)**.** Let $y \in \{1, \ldots, K\}$ be the true class of an input $x \in \mathbb{R}^n$. A network f_θ is said to be *certifiably robust* at x under an ℓ_∞ perturbation of radius ϵ if:

$$\underline{z}_y^{(L)} > \overline{z}_j^{(L)}, \quad \forall j \neq y, \tag{15}$$

where $[\underline{z}^{(L)}, \overline{z}^{(L)}]$ are lower and upper bounds on the output logits at the final layer L, computed using IBP over the perturbation set $\{x + \delta : \|\delta\|_\infty \leq \epsilon\}$.

Definition 3.2 (Certified Robustness in Continual Learning via IBP)**.** Consider a continual learner exposed to a sequence of T tasks, $\{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$, where each dataset $\mathcal{D}_t = \{(x_i^{(t)}, y_i^{(t)})\}_{i=1}^{N_t}$ is drawn from a task-specific distribution $P_t(x, y)$. Let θ_t denote the model parameters after learning task t. The learner is said to be *certifiably robust via IBP* up to time t if, for all previously seen samples $(x, y) \in \bigcup_{s=1}^{t} \mathcal{D}_s$, the following condition holds:

$$\underline{z}_y^{(L)}(x; \theta_t) > \max_{j \neq y} \overline{z}_j^{(L)}(x; \theta_t), \quad \text{for all } \|\delta\|_\infty \leq \epsilon. \tag{16}$$

This condition guarantees that the model's predictions remain robust to bounded adversarial perturbations across all tasks encountered so far, without requiring access to the full datasets $\mathcal{D}_1, \ldots, \mathcal{D}_{t-1}$.

SHIELD is specifically designed to satisfy this robustness criterion by combining IBP-based training with a hypernetwork-based CL framework. This leads to the following result:

Theorem 3.1 (Certified Robustness of SHIELD)**.** Let $f_{\boldsymbol{\theta}_t}$ be a target network whose weights are generated by a hypernetwork $H(\boldsymbol{e}_t; \mathbf{\Phi})$, where $\boldsymbol{e}_t$ is a task embedding and $\mathbf{\Phi}$ are the learnable hypernetwork parameters. Then, under successful optimization, SHIELD satisfies Definition 3.2, i.e., it is a certifiably robust continual learner.

Proof. According to Definition 3.2, we must show that at time t, for any previously encountered task $s \leq t$ and any sample $(x, y) \in \mathcal{D}_s$, the following certified robustness condition holds:

$$\underline{z}_y^{(L)}(x; \theta_t) > \max_{j \neq y} \overline{z}_j^{(L)}(x; \theta_t), \quad \text{for all } \|\delta\|_\infty \leq \epsilon. \tag{17}$$

In SHIELD, the task-specific parameters $\boldsymbol{\theta}_t$ of the target network $f_{\boldsymbol{\theta}_t}$ are generated via a shared hypernetwork: $\boldsymbol{\theta}_t = H(\boldsymbol{e}_t; \boldsymbol{\Phi})$. The hypernetwork is trained to generate networks that are certifiably robust using IBP. Specifically, it minimizes a certified loss described by Equation 11. Thus, successful optimization of the loss provides certified robustness for each task t.

To retain certified robustness on prior tasks, SHIELD includes a regularization term in the hypernetwork loss (see Equation 13). This encourages the hypernetwork to preserve the outputs of previously learned tasks while adapting to the new one. While the extent of robustness retention depends on optimization effectiveness, this design enables SHIELD to uphold certified guarantees under ideal learning conditions.

Thus, assuming that both the certified objective and regularization are effectively optimized, the resulting model $f_{\boldsymbol{\theta}_t}$ satisfies:

$$\underline{z}_y^{(L)}(x; \theta_t) > \max_{j \neq y} \overline{z}_j^{(L)}(x; \theta_t),$$

for all $(x, y) \in \bigcup_{s=1}^{t} \mathcal{D}_s$ and all $\|\delta\|_\infty \leq \epsilon$. This concludes the proof that SHIELD satisfies Definition 3.2. □

In this section, we have extended the notion of certified robustness to continual learning and demonstrated that SHIELD—through its use of IBP-based training and a hypernetwork architecture—can theoretically preserve certified robustness across tasks. While the guarantees rely on successful optimization, the formulation offers a principled framework for designing certifiably robust continual learners.

4 Experiments

This section outlines the experimental setup and results for SHIELD. We describe the datasets used, along with the architectures of both the target network and the hypernetwork. All experiments are conducted in a Task-Incremental Learning setting, where the task identity is known to the model during both training and evaluation.

Benchmarks To assess the effectiveness of our approach, we perform experiments on three standard benchmarks: Permuted MNIST, Rotated MNIST, and Split CIFAR-100. The structure of tasks and class divisions follows the protocol established in [38]. Permuted MNIST introduces a different fixed pixel permutation for each task, applied to the original MNIST images. Rotated MNIST, on the other hand, creates tasks by rotating the digits by varying angles. These two datasets are widely used in the continual learning literature as synthetic benchmarks derived from MNIST [11, 27]. Split CIFAR-100 [50] is constructed by randomly partitioning the 100 classes of the CIFAR-100 dataset into 10 mutually exclusive groups, with each group forming a separate classification task of 10 classes. In Split CIFAR-100, each class appears in only one group, ensuring no class overlap between tasks. This design allows each subset to be treated as an independent classification task, resulting in a sequence of tasks for continual learning.

Architectures For Permuted MNIST and Rotated MNIST, we used the same interval-based target network architectures as in [38], with the addition of biases. For Split CIFAR-100, we used a modified interval version of AlexNet. Specifically, we removed dropout, included biases, and used interval-aware batch normalization layers [19]. The number of hidden units in the linear layers is reduced to 100, and max pooling layers are replaced with average pooling, which provides greater stability when propagating interval bounds. Additionally, we apply a 4×4 average pooling layer after the final convolutional layer. This reduction of the original AlexNet architecture is motivated by the use of hypernetworks, which generate target network weights. Smaller target architectures help keep the overall system memory- and compute-efficient. Further discussion of this design choice can be found in Appendix B. A shared MLP hypernetwork is used across all datasets.

Training details For the Split CIFAR-100 experiments, we used an AlexNet-based target network along with a hypernetwork comprising two hidden layers of 100 and 50 neurons respectively. The embedding size was set to 512, with a batch size of 32 and learning rate of 0.001. The hypernetwork used a regularization coefficient $\beta = 0.01$, and training was performed with an ℓ_∞ perturbation of $\epsilon = 0.005$. The optimizer was Adam, and no data augmentation was used. Training was conducted for 200 epochs with ReLU activation in both the target and hypernetwork. We applied a learning rate scheduler: `ReduceLROnPlateau` with monitoring set to the maximum validation accuracy, reduction factor $\sqrt{0.1}$, patience of 5 epochs, minimum learning rate of 5×10^{-7}, and no cooldown. Batch normalization was enabled throughout.

For the Permuted MNIST task, we employed a 2-layer MLP with 256 neurons per layer in the target network and 100 neurons per layer in the hypernetwork. The embedding size was 24, the batch size was 128, and the learning rate was 0.001. We used $\beta = 0.001$ and performed training with a perturbation budget of $\epsilon = \frac{2}{255}$. For the Rotated MNIST experiments, the setup mirrored that of Permuted MNIST but used an embedding size of 96 and $\beta = 0.001$. Training on Permuted MNIST and Rotated MNIST was conducted for 5000 iterations.

Model selection was based on the best validation loss. The optimizer across all experiments was Adam, and ReLU was used as the activation function throughout. In all cases, the same adversarial attack settings were used as in [38], except that the number of attack iterations for PGD and AutoAttack was not specified in that work; we defaulted to 10 iterations for consistency across evaluations. Experiments were conducted on NVIDIA RTX 4090 and A100 GPUs. Full details of the hyperparameter grid search are provided in the Appendix C.

Metrics We evaluate performance using two standard metrics in continual learning: average accuracy (ACC) and backward transfer (BWT). Average accuracy is computed as $\text{ACC} = \frac{1}{T}\sum_{t=1}^{T} R_{T,t}$, where $R_{T,t}$ denotes the test accuracy on task t after training on the final task T. Backward transfer is defined as $\text{BWT} = \frac{1}{T-1}\sum_{t=1}^{T-1}(R_{T,t} - R_{t,t})$, measuring the influence of learning new tasks on performance over previously learned ones. To assess continual learning performance, we evaluate accuracy on the test sets of all prior tasks.

Results Tables 1, 2, and 3 present the average classification accuracy (ACC) and backward transfer (BWT) achieved by SHIELD and a range of baseline methods after sequential learning across all tasks for the Permuted MNIST, Rotated MNIST, and Split CIFAR-100 benchmarks.

Table 1: Comparisons of ACC and BWT after learning all the tasks on the Permuted MNIST dataset.

Method	Permuted MNIST				
	AutoAttack	PGD	FGSM	Original samples	
	ACC(%)	ACC(%)	ACC(%)	ACC(%)	BWT
SGD	14.1	15.4	21.8	36.8	-0.66
SI	14.3	16.5	22.3	36.9	-0.67
A-GEM	14.1	19.7	22.9	48.4	-0.54
EWC	39.4	43.1	50.0	84.9	-0.12
GEM	12.1	75.5	72.8	96.4	-0.01
OGD	19.7	24.1	26.0	46.8	-0.57
GPM	70.4	72.9	65.7	97.2	-0.01
DGP	81.6	81.2	75.8	**97.6**	-0.01
SHIELD	$\mathbf{85.64 \pm 1.32}$	$\mathbf{90.02 \pm 0.8}$	$\mathbf{78.87 \pm 2.06}$	93.58 ± 0.52	$\mathbf{0.02 \pm 0.01}$

On the Permuted MNIST dataset (Table 1), SHIELD outperforms all baselines across all adversarial attack settings. Specifically, it achieves $85.64 \pm 1.32\%$ ACC under AutoAttack, $90.02 \pm 0.8\%$ under PGD, and $78.87 \pm 2.06\%$ under FGSM, along with $93.58 \pm 0.52\%$ on clean/original data. Notably, SHIELD is the only method showing positive backward transfer, with a BWT of 0.02 ± 0.01, indicating it not only avoids catastrophic forgetting but also improves over past tasks. Other methods like DGP and GPM maintain near-zero BWT but do not surpass SHIELD in adversarial robustness.

On the Rotated MNIST benchmark (Table 2), a similar trend is observed. SHIELD achieves the highest accuracy under AutoAttack ($88.85 \pm 0.27\%$), PGD ($92.85 \pm 0.16\%$), and FGSM ($83.82 \pm$

Table 2: Comparisons of ACC and BWT after learning all the tasks on the Rotated MNIST dataset.

Method	Rotated MNIST				
	AutoAttack	PGD	FGSM	Original samples	
	ACC(%)	ACC(%)	ACC(%)	ACC(%)	BWT
SGD	14.1	9.9	20.4	32.3	-0.71
SI	13.9	15.3	20.1	33.0	-0.72
A-GEM	14.1	21.6	24.8	45.4	-0.57
EWC	45.1	49.5	46.5	80.7	-0.18
GEM	11.9	76.5	74.4	96.7	-0.01
OGD	19.7	23.8	23.8	48.0	-0.55
GPM	68.8	71.5	65.9	97.1	-0.01
DGP	81.6	82.6	78.6	**98.1**	**-0.00**
SHIELD	$\mathbf{88.85 \pm 0.27}$	$\mathbf{92.85 \pm 0.16}$	$\mathbf{83.82 \pm 0.2}$	95.62 ± 0.06	-0.03 ± 0.05

0.2%). It also performs competitively on clean samples ($95.62 \pm 0.06\%$), with a small BWT of -0.03 ± 0.05, which is comparable to the best-performing methods in this setting.

Table 3: Comparisons of ACC and BWT after learning all the tasks on the Split-CIFAR100 dataset.

Method	Split CIFAR-100				
	AutoAttack	PGD	FGSM	Original samples	
	ACC(%)	ACC(%)	ACC(%)	ACC(%)	
SGD	10.3	12.8	19.4	46.5	-0.49
SI	13.0	15.2	19.8	45.4	-0.48
A-GEM	12.6	12.9	20.7	40.6	-0.48
EWC	12.6	23.2	30.5	56.8	-0.35
GEM	21.2	19.4	47.7	60.6	-0.13
OGD	11.8	14.1	18.9	44.2	-0.50
GPM	34.4	36.6	**53.7**	58.2	**-0.10**
DGP	36.6	39.2	48.0	**67.2**	-0.13
SHIELD	$\mathbf{49.78 \pm 1.06}$	$\mathbf{59.76 \pm 0.77}$	45.37 ± 0.41	64.24 ± 0.73	-0.34 ± 0.23

On the more challenging Split CIFAR-100 benchmark (Table 3), SHIELD again achieves state-of-the-art performance under AutoAttack and PGD, with $49.78 \pm 1.06\%$ and $59.76 \pm 0.77\%$ ACC respectively. Its clean accuracy ($64.24 \pm 0.73\%$) is close to DGP's (67.2%), while its BWT (-0.34 ± 0.23) remains better than most baselines except GPM and GEM, which however do not match SHIELD's adversarial robustness.

The consistently strong performance of SHIELD under adversarial conditions highlights its robustness across varied continual learning tasks. Importantly, achieving high accuracy on clean/original samples is non-trivial when training involves large ℓ_∞ perturbations. This robustness-clean trade-off is a known challenge: stronger adversarial training (with larger perturbation norms) often enhances defense capability but can moderately degrade clean accuracy due to reduced overfitting capacity and increased regularization pressure.

Additionally, the lower FGSM accuracy observed in all benchmarks is primarily due to mismatch in attack strength: FGSM in these evaluations was typically run with a larger ϵ than used during training. Since FGSM is a single-step attack and less adaptive than PGD or AutoAttack, it becomes particularly sensitive to such mismatch, leading to lower accuracy in this column.

Taken together, the results confirm that SHIELD not only preserves knowledge over time (as evidenced by competitive or positive BWT) but also builds adversarially robust representations that generalize well across tasks.

References

[1] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154, 2018.

[2] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018.

[3] A. Bagnall, R. Bunescu, and G. Stewart. Training ensembles to detect adversarial examples. *arXiv preprint arXiv:1712.04006*, 2017.

[4] W. Brendel, J. Rauber, and M. Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

[5] A. Chaudhry, N. Khan, P. K. Dokania, and P. H. S. Torr. Continual learning in low-rank orthogonal subspaces, 2020. URL `https://arxiv.org/abs/2010.11635`.

[6] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26, 2017.

[7] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

[8] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[9] Y. Dong, T. Pang, H. Su, and J. Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4312–4321, 2019.

[10] M. Goldblum, L. Fowl, S. Feizi, and T. Goldstein. Adversarially robust distillation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3996–4003, 2020.

[11] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[12] 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.

[13] D. Ha, A. Dai, and Q. V. Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[14] T. L. Hayes, N. D. Cahill, and C. Kanan. Memory efficient experience replay for streaming learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9769–9776. IEEE, 2019.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] C. Henning, M. Cervera, F. D'Angelo, J. Von Oswald, R. Traber, B. Ehret, S. Kobayashi, B. F. Grewe, and J. Sacramento. Posterior meta-replay for continual learning. *Advances in Neural Information Processing Systems*, 34:14135–14149, 2021.

[17] Y.-C. Hsu, Y.-C. Liu, A. Ramasamy, and Z. Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*, 2018.

[18] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58, 2011.

[19] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. URL `https://arxiv.org/abs/1502.03167`.

[20] X. Jia, Y. Zhang, B. Wu, K. Ma, J. Wang, and X. Cao. Las-at: adversarial training with learnable attack strategy. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13398–13408, 2022.

[21] G. Jin, S. Shen, D. Zhang, F. Dai, and Y. Zhang. Ape-gan: Adversarial perturbation elimination with gan. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3842–3846. IEEE, 2019.

[22] H. Khan, P. M. Shah, S. F. A. Zaidi, Q. Zia, et al. Susceptibility of continual learning against adversarial attacks. *arXiv preprint arXiv:2207.05225*, 2022.

[23] 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*, 114(13):3521–3526, 2017.

[24] P. Krukowski, A. Bielawska, K. Książek, P. Wawrzyński, P. Batorski, and P. Spurek. Hyperinterval: Hypernetwork approach to training weight interval regions in continual learning. *arXiv preprint arXiv:2405.15444*, 2024.

[25] K. Książek and P. Spurek. Hypermask: Adaptive hypernetwork-based masks for continual learning. *arXiv preprint arXiv:2310.00113*, 2023.

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

[27] H. Liu and H. Liu. Continual learning with recursive gradient optimization, 2022. URL `https://arxiv.org/abs/2201.12522`.

[28] V. Lomonaco and D. Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on robot learning*, pages 17–26. PMLR, 2017.

[29] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6470–6479, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[30] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[31] 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*, pages 7765–7773, 2018.

[32] M. Mazur, Ł. Pustelnik, S. Knop, P. Pagacz, and P. Spurek. Target layer regularization for continual learning using cramer-wold distance. *Information Sciences*, 609:1369–1380, 2022.

[33] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[34] M. Mirman, T. Gehr, and M. Vechev. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*, pages 3578–3586. PMLR, 2018.

[35] P. Morawiecki, P. Spurek, M. Śmieja, and J. Tabor. Fast and stable interval bounds propagation for training verifiably robust models. *arXiv preprint arXiv:1906.00628*, 2019.

[36] A. Mustafa, S. H. Khan, M. Hayat, R. Goecke, J. Shen, and L. Shao. Deeply supervised discriminative learning for adversarial defense. *IEEE transactions on pattern analysis and machine intelligence*, 43(9):3154–3166, 2020.

[37] T. Pang, X. Yang, Y. Dong, H. Su, and J. Zhu. Bag of tricks for adversarial training. *arXiv preprint arXiv:2010.00467*, 2020.

[38] X. Ru, X. Cao, Z. Liu, J. M. Moore, X.-Y. Zhang, X. Zhu, W. Wei, and G. Yan. Maintaining adversarial robustness in continuous learning. *arXiv preprint arXiv:2402.11196*, 2024.

[39] 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.

[40] G. Saha, I. Garg, and K. Roy. Gradient projection memory for continual learning. *arXiv preprint arXiv:2103.09762*, 2021.

[41] C. Shi, C. Holtz, and G. Mishne. Online adversarial purification based on self-supervision. *arXiv preprint arXiv:2101.09387*, 2021.

[42] C. Sitawarin, A. N. Bhagoji, A. Mosenia, M. Chiang, and P. Mittal. Darts: Deceiving autonomous cars with toxic signs. *arXiv preprint arXiv:1802.06430*, 2018.

[43] J. Su, D. V. Vargas, and K. Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[44] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[45] J. von Oswald, C. Henning, B. F. Grewe, and J. Sacramento. Continual learning with hypernetworks. In *International Conference on Learning Representations*, 2019.

[46] 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.

[47] M. Wortsman, V. Ramanujan, R. Liu, A. Kembhavi, M. Rastegari, J. Yosinski, and A. Farhadi. Supermasks in superposition. *Advances in Neural Information Processing Systems*, 33:15173–15184, 2020.

[48] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*, 2017.

[49] Z. Yang, Z. Xu, J. Zhang, R. Hartley, and P. Tu. Adaptive test-time defense with the manifold hypothesis. *arXiv preprint arXiv:2210.14404*, 3, 2022.

[50] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR, 2017.

[51] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization, 2018. URL `https://arxiv.org/abs/1710.09412`.

[52] Y. Zhou and Z. Hua. Defense without forgetting: Continual adversarial defense with anisotropic & isotropic pseudo replay. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24263–24272, 2024.

[53] J. Zhu, J. Yao, B. Han, J. Zhang, T. Liu, G. Niu, J. Zhou, J. Xu, and H. Yang. Reliable adversarial distillation with unreliable teachers. *arXiv preprint arXiv:2106.04928*, 2021.

A Interval neural network layers

The implementation of linear and convolutional layers in the interval domain is straightforward and follows the approach described in Section 3.1 of the main paper. However, implementing other interval layers requires more care and is not as trivial.

Interval batch normalization Batch normalization in the interval domain is more involved. Instead of simply normalizing the lower and upper bounds separately, we first concatenate the lower and upper bounds into a single tensor. We then compute the batch statistics—mean and variance—over this combined tensor, capturing the distribution of the entire interval. Using these statistics, we apply the standard batch normalization transformation.

More concretely, given pre-activation interval bounds $[\underline{x}, \overline{x}]$, we form the concatenated tensor:

$$X_{\text{concat}} = [\underline{x}, \overline{x}], \tag{18}$$

and compute the expected mean μ and variance σ^2 over X_{concat}. We then normalize and scale the interval bounds as follows:

$$\left[\frac{\underline{x} - \mu}{\sqrt{\sigma^2 + \epsilon}}, \frac{\overline{x} - \mu}{\sqrt{\sigma^2 + \epsilon}} \right], \tag{19}$$

followed by the affine transformation with learned parameters γ (scale) and β (shift):

$$\gamma \cdot \left[\frac{\underline{x} - \mu}{\sqrt{\sigma^2 + \epsilon}}, \frac{\overline{x} - \mu}{\sqrt{\sigma^2 + \epsilon}} \right] + \beta. \tag{20}$$

Special attention is required to handle the sign of γ correctly to maintain valid lower and upper bounds. If the sign of γ is negative, the lower and upper bounds must be swapped after scaling to preserve correct interval ordering.

Interval pooling layers Pooling operations require special treatment in the interval domain. For max pooling, given an input interval over a pooling window with elements $\{[\underline{x}_i, \overline{x}_i]\}_{i=1}^n$, the output interval bounds are computed as:

$$\text{MaxPool}_{\text{inf}} = \max_{1 \le i \le n} \underline{x}_i, \quad \text{MaxPool}_{\text{sup}} = \max_{1 \le i \le n} \overline{x}_i, \tag{21}$$

where $\text{MaxPool}_{\text{inf}}$ and $\text{MaxPool}_{\text{sup}}$ denote the lower and upper bounds of the max pooled interval, respectively. However, max pooling is non-monotonic with respect to interval bounds, which can lead to overly loose and unstable interval approximations.

In contrast, average pooling computes the output interval by averaging the lower and upper bounds across the pooling window:

$$\text{AvgPool}([\underline{x}, \overline{x}]) = \left[\frac{1}{n} \sum_{i=1}^{n} \underline{x}_i, \quad \frac{1}{n} \sum_{i=1}^{n} \overline{x}_i \right]. \tag{22}$$

This method produces a tighter and more stable over-approximation for interval bounds, which is why average pooling is preferred for interval neural networks.

B Discussion on learnable number of parameters used in SHIELD

Since we use hypernetworks to generate the weights of the target network, the total number of learnable parameters typically exceeds that of baseline architectures. However, we show that it is possible to significantly reduce the number of parameters—bringing it in line with standard models—while still achieving state-of-the-art performance. Importantly, the parameter count of the target network remains comparable to, or even smaller than, that of baseline models; the overhead comes solely from the hypernetwork. For the Permuted MNIST dataset, we reduce the number of learnable parameters to match the number of learnable parameters in [38] by using a hypernetwork consisting of a 2-layer MLP with hidden dimensions 100 and 50, and an embedding size of 96. The target network is an MLP with two hidden layers of 50 neurons each. For Rotated MNIST, we use the same architecture as for Permuted MNIST, but with a smaller embedding size of 24 to reduce the parameter count accordingly. For Split CIFAR-100, we match the parameter budget from [38] by setting the embedding size to 512 and using a hypernetwork MLP with two hidden layers of 200 and 50 neurons. Additionally, we reduce the complexity of the target AlexNet by removing two hidden linear layers, replacing max pooling with a 4×4 average pooling layer after the last convolution, and reducing the final linear layer's dimension to 100. The total number of learnable parameters used can be found in Table 4.

Table 4: Comparison of learnable parameters (in millions) between our method and the baseline methods from [38], which we refer to simply as "Baseline". Our reduced version achieves comparable or better performance with a similar or smaller number of parameters.

Method	Permuted MNIST	Rotated MNIST	Split CIFAR-100
Baseline	2M	2M	5.5M
SHIELD	2.8M	2.8M	5.6M

Tables 5, 6, and 7 present the average classification accuracy (ACC) of SHIELD and several baselines after learning all tasks sequentially. The results include performance under three adversarial attacks (AutoAttack, PGD, FGSM) and on original clean data.

SHIELD consistently achieves state-of-the-art robustness across all benchmarks. On Permuted MNIST (Table 5), SHIELD achieves $86.53 \pm 1.57\%$ (AutoAttack), $90.47 \pm 1.36\%$ (PGD), and $79.6 \pm 1.34\%$ (FGSM), outperforming all baselines under adversarial conditions. Similar trends are observed in Table 6 for Rotated MNIST, with SHIELD achieving $82.07 \pm 0.58\%$ (AutoAttack) and $86.32 \pm 0.28\%$ (PGD), and in Table 7 for Split CIFAR-100, where SHIELD reaches $47.03 \pm 0.9\%$ (AutoAttack) and $56.66 \pm 1.04\%$ (PGD).

Table 5: Comparison of ACC after learning all the tasks on the Permuted MNIST dataset. We averaged ACC results of SHIELD over 5 seeds.

Method	Permuted MNIST			
	AutoAttack	PGD	FGSM	Original samples
	ACC(%)	ACC(%)	ACC(%)	ACC(%)
SGD	14.1	15.4	21.8	36.8
SI	14.3	16.5	22.3	36.9
A-GEM	14.1	19.7	22.9	48.4
EWC	39.4	43.1	50.0	84.9
GEM	12.1	75.5	72.8	96.4
OGD	19.7	24.1	26.0	46.8
GPM	70.4	72.9	65.7	97.2
DGP	81.6	81.2	75.8	**97.6**
SHIELD	$\mathbf{86.53 \pm 1.57}$	$\mathbf{90.47 \pm 1.36}$	$\mathbf{79.6 \pm 1.34}$	90.71 ± 1.8

Table 6: Comparison of ACC after learning all the tasks on the Rotated MNIST dataset. We averaged ACC results of SHIELD over 5 seeds.

Method	Rotated MNIST			
	AutoAttack	PGD	FGSM	Original samples
	ACC(%)	ACC(%)	ACC(%)	ACC(%)
SGD	14.1	9.9	20.4	32.3
SI	13.9	15.3	20.1	33.0
A-GEM	14.1	21.6	24.8	45.4
EWC	45.1	49.5	46.5	80.7
GEM	11.9	76.5	74.4	96.7
OGD	19.7	23.8	23.8	48.0
GPM	68.8	71.5	65.9	97.1
DGP	81.6	82.6	**78.6**	**98.1**
SHIELD	$\mathbf{82.07 \pm 0.58}$	$\mathbf{86.32 \pm 0.28}$	73.39 ± 0.5	93.12 ± 0.22

While SHIELD does not achieve the highest accuracy on clean/original samples, its performance remains competitive with top-performing methods such as DGP. This modest reduction in clean accuracy is an expected consequence of two design choices: (1) SHIELD is trained using a relatively small target network architecture, which limits overfitting to clean data; and (2) stronger adversarial perturbations (with higher ℓ_∞ norms) used during training often result in lower clean accuracy

Table 7: Comparison of ACC after learning all the tasks on the Split CIFAR-100 dataset. We averaged ACC results of SHIELD over 5 seeds.

Method	Split CIFAR-100			
	AutoAttack	PGD	FGSM	Original samples
	ACC(%)	ACC(%)	ACC(%)	ACC(%)
SGD	10.3	12.8	19.4	46.5
SI	13.0	15.2	19.8	45.4
A-GEM	12.6	12.9	20.7	40.6
EWC	12.6	23.2	30.5	56.8
GEM	21.2	19.4	47.7	60.6
OGD	11.8	14.1	18.9	44.2
GPM	34.4	36.6	**53.7**	58.2
DGP	36.6	39.2	48.0	**67.2**
SHIELD	$\mathbf{47.03 \pm 0.9}$	$\mathbf{56.66 \pm 1.04}$	47.71 ± 0.86	59.36 ± 1.11

but significantly enhance robustness. This trade-off enables SHIELD to maintain a strong balance between robustness and overall performance, making it a reliable choice for adversarially robust continual learning.

C Hyperparameter grid search details

To identify the most effective configuration for each dataset, we conducted an extensive grid search over key hyperparameters, evaluating combinations to maximize validation performance. Below we summarize the search spaces used for each dataset.

Permuted MNIST. For the Permuted MNIST dataset, the grid search was performed using a 2-layer MLP with 256 neurons per layer in the target network. As a hypernetwork, we used an MLP, with architecture defined by the hypernetwork hidden layers parameter. The following hyperparameters were explored:

- Embedding sizes: 24, 48, 96
- Learning rate: 0.001
- Batch sizes: 64, 128
- Regularization coefficients β: 0.0005, 0.001, 0.005, 0.01, 0.05
- Perturbation sizes ϵ: $\frac{2}{255}$, $\frac{20}{255}$, $\frac{25}{250}$
- Hypernetwork hidden layers (MLP): [100, 50], [200, 50], [100, 100]
- Number of training iterations: 5000

Rotated MNIST. We use the same grid search as for the Permuted MNIST dataset.

Split CIFAR-100. For the Split CIFAR-100 dataset, the grid search was conducted using AlexNet as the target network and an MLP-based hypernetwork. Batch normalization was enabled, and data augmentation was disabled. The full hyperparameter search space included:

- Embedding sizes: 128, 256, 512
- Learning rate: 0.001
- Batch sizes: 32, 64
- Regularization coefficients β: 0.01, 0.05, 0.1
- Perturbation sizes ϵ: $\frac{4}{255}$, 0.005, 0.01
- Hypernetwork hidden layers (MLP): [200, 50], [100, 50], [100, 100], [100], [200]
- Number of training epochs: 200

The Adam optimizer was used across all tasks. These grid searches enabled a systematic selection of optimal configurations for each dataset.