Class-Incremental SAR ATR in Noisy and Adversarial Environments

Edison Mucllari[a], Aswin Raghavan[b], and Zachary Alan Daniels[b]

[a]University of Kentucky, Lexington, Kentucky, USA
[b]SRI International, Princeton, New Jersey, USA

ABSTRACT

Automatic target recognition (ATR) models generally consist of machine learning (ML) models trained on a collection of clean, well-labeled data samples with a fixed known set of target classes. During deployment, it is assumed that these models operate over new data samples that align with those from the training data distribution. Practical systems require updating ATR models over time in response to changing sensor technology, sensor degradation, changing environmental conditions, appearance of new targets, and adversarial interference. This work focuses on class-incremental synthetic aperture radar (SAR)-based ATR where new classes are sequentially added to an existing ML model, and the model has limited access to data samples from past targets during training. The model must balance the ability to adapt to unseen targets with the ability to maintain high recognition accuracy on past targets. Continual learning introduces additional avenues for corrupting models during data collection. We study class-incremental SAR ATR under three types of noise that may arise during data collection that may interfere with learning: noisy labels, perturbed data measurements, and adversarial poisoning attacks. We propose a novel approach that constructs a small noise-tolerant experience replay buffer based on selecting coresets using the CRUST[1] algorithm. Experiments are conducted on the MSTAR[2] and SynthWakeSAR[3] datasets to validate the utility of the approach. To test effectiveness against adversarial poisoning attacks, we adapt the "Scattering Model Guided Adversarial Attack" algorithm (SMGAA)[4] for inserting backdoor triggers.

Keywords: SAR-Based Automatic Target Recognition, Continual Learning, Class-Incremental Learning, Robust Machine Learning, Coreset Selection

1. INTRODUCTION

Automatic target recognition (ATR) involves autonomously analyzing sensor data to identify and classify objects-of-interests (targets). For defense applications, targets generally consist of different classes of vehicles, man-made structures such as buildings, and natural background clutter (e.g., flocks of animals and vegetation). This work focuses on ATR from synthetic aperture radar (SAR) imagery.

Many modern ATR models consist of machine learning (ML) models that map input sensor data to output target labels. Traditionally, these models are trained on a collection of clean, well-labeled data samples with a fixed known set of target classes. During deployment, it is assumed that these models operate on new data samples that align with those from the training data distribution. In contrast, practical systems often require updating ATR models over time in response to i) changing sensor technology, ii) sensor degradation and failure, iii) changing environmental conditions, iv) adversarial interference, v) appearance of new targets, and vi) other factors. Recent work has started to focus on (class-)incremental ATR (see Section 2.3), where new target classes are added sequentially over time to an existing ML-based recognition model. Due to constraints on the system

Further author information: (Send correspondence to Z.A.D.)
Z.A.D.: E-mail: zachary.daniels@sri.com, Telephone: (609) 734-3161

Edison Mucllari, Aswin Raghavan, and Zachary Alan Daniels "Class-Incremental SAR ATR in Noisy and Adversarial Environments," Proc. SPIE, Automatic Target Recognition XXXV, (April 14, 2025).

(e.g., limited memory, controlled access to past data, limited resources for model training), the recognition model has limited access to past data samples when performing model updates. This means that the model must be able to adapt to shifts in the data distribution using at most i) a small, fixed-size buffer of past data samples and ii) access to a newly collected set of data samples from a previously unseen data distribution. In this continual learning (CL) problem,[5,6] the model must balance plasticity (ability to adapt to unseen targets) with stability (ability to maintain high recognition accuracy on past targets).

Unlike traditional ML, CL consists of a repeated loop of data collection followed by model updates. This problem structure introduces additional avenues for model corruption. This work investigates class-incremental SAR ATR under three types of noise that may arise during data collection that may interfere with learning:

1. **Label Noise:** Data annotation is expensive in a CL system, which requires continuous data collection and labeling. To reduce the burden and expense of human annotation, data labeling can be automated in several ways. For example, targets can be automatically (imperfectly) filtered from uninformative background measurements, which may introduce false detections. Furthermore, inexact pseudo-labeling can be employed: for each new data collection event, humans can annotate a small subset of the data and semi-supervised learning[7] can be applied to assign (initial) labels to the remaining target data. Both of these methods (and even purely human-driven annotation), introduce errors in the labeling of the data, which provide antagonistic feedback that may disrupt the continual learning process.

2. **Measurement Noise:** Noise can also arise in the measured data. For example, sensors can fail over time introducing artifacts into the captured imagery. Additionally, severe environmental conditions can produce perturbed sensor measurements. Speckle is an especially common[8] type of measurement noise that can appear in SAR imagery.

3. **Adversarial Noise:** SAR ATR is commonly used in intelligence, surveillance, and reconnaissance (ISR) use cases. In such applications, adversaries may try to exploit flaws in the ATR models to avoid detection or trigger false detections. Recently, there has been some work on physically-plausible adversarial attacks for SAR ATR[4,9] where scatter patterns of targets-of-interest are manipulated to avoid detection. Similar ideas can be employed for poisoning incremental ATR models, where adversaries may try to insert backdoor triggers[10] that encourage ATR models to misclassify targets when specific trigger patterns are encountered.

Most existing work for incremental SAR ATR fail to consider how noise in the data collection pipeline can corrupt the recognition models, reducing the utility of the model and potentially, making the model more susceptible to adversarial manipulation. In contrast, we propose a novel, theoretically-motivated algorithm for noise-tolerant (w.r.t. both label and measurement noise) class-incremental SAR ATR. Our approach, *Continual CRUST*, leverages a coreset-based replay mechanism that extends the CRUST algorithm[1] to the continual learning setting. CRUST has been previously shown to be robust to noisy labels in the static learning setting. We derive additional theoretical guarantees on CRUST's robustness to measurement noise.

To understand the problem and evaluate our approach, we conduct experiments on the benchmark MSTAR[2] and SynthWakeSAR[3] datasets examining:

- Challenges associated with SAR ATR in the continual learning setting
- Resilience of CL algorithms to label flipping noise under different formulations of the class-incremental learning problem
- Ability of the proposed Continual CRUST algorithm to filter foreground from background data samples
- Resilience of the proposed Continual CRUST algorithm to measurement noise (including robustness to realistic speckle as well as total sensor failure/ total image corruption)
- Resilience of the proposed Continual CRUST algorithm to adversarial noise in the form of physically-plausible poisoning attacks.

To explore robustness to physically-plausible backdoor attacks, we generate a poisoned variant of the MSTAR dataset by adapting the "Scattering Model Guided Adversarial Attack" algorithm (SMGAA)[4] , which was originally designed for avoidance attacks, to the problem of inserting backdoor triggers.

When confronted with significant label noise, our proposed algorithm shows promising improvement over existing baselines. When confronted with significant measurement noise, our proposed algorithm shows similar performance as existing baselines, but shows improved purity (filtering of noisy samples) in the learned coresets. Preliminary studies suggest that a possible weakness of our method is susceptibility to backdoor triggers under heavily poisoned data sets.

2. RELATED WORK

2.1 Overview of Continual Learning

Continual learning[11, 12] involves training an agent on a sequence of tasks, where as the agent encounters new tasks, it must balance plasticity (adaptation to the new task) and stability (maintaining performance on previous tasks). There are three major classes of problem in supervised continual learning:[12] *task-incremental learning (TIL)*, *domain-incremental learning (DIL)*, and *class-incremental learning (CIL)*. In TIL, an agent solves sequences of tasks with explicit knowledge of the identity of the current task-of-interest. In DIL, the current task-of-interest is not known by the agent, but the structure of the problem remains consistent between tasks as novel data distributions appear. In CIL, new class sets are added over time. This work primarily focuses on the CIL setting; although, we expect the proposed algorithm could be adapted to the other problem settings.

Most algorithms for CIL follow one of three strategies.[13] *Memory-based methods*[14–19]) save data samples (or derived representations) from previous tasks and periodically "replay" the samples to reduce catastrophic forgetting. *Regularization-based methods*[20–22] attempt to prevent neural networks from overfitting to the new task by constraining the learning behavior of neural networks, e.g. by penalizing large changes in the weights of the network. Architecture-based methods[23–27] adaptively modify the model architecture (e.g. by incrementally growing and pruning the architecture). This paper proposes a memory-based approach to CIL using noise-tolerant coresets as the memory mechanism.

2.1.1 Memory-Based Approaches for Continual Learning

In many practical settings, continual learning algorithms are restricted to run on hardware with constraints on computation, e.g. edge-based processors on small unmanned aerial vehicles. Often time and energy constraints make it infeasible to retrain models using all of the data obtained from past and current data collection events. These limitations necessitate that CL models must be able to perform model updates with at most a small, fixed-size memory (replay buffer) available. The primary concerns of memory-based CL is i) how to intelligently select or compress past examples to fit within the fixed memory buffer while ii) minimizing catastrophic forgetting on past tasks and iii) minimizing interference with learning new tasks.

The simplest form of memory-based continual learning is *random experience replay*, which randomly selects samples to add to the buffer at the end of every learning experience. *Prioritized replay*[15, 28, 29] selects samples for the replay buffer more intelligently by employing heuristics. *Coresets* are a mathematical tool that is effective at summarizing large datasets using only a few data points, which makes them useful tools for constructing replay buffers.[30–32] The quality of data samples w.r.t. downstream classification performance can often be approximated from gradients. Many approaches use *gradient information* to construct replay buffers.[16, 19] Our approach leverages gradient information to form coreset-based buffers. Other methods extract *exemplars* from the training data to serve as a replay mechanism. For example, iCARL[18] extracts exemplars using class means. Often, it is sufficient to save intermediate representations (e.g. *latent replay*[33]) or *compressed representations*[34] in place of raw data samples. *Generative replay*[35] involves modeling the distribution of samples from past experiences via learning a *generative model.* Data can then be sampled on-the-fly from the generative model to construct replay buffers that have higher diversity.

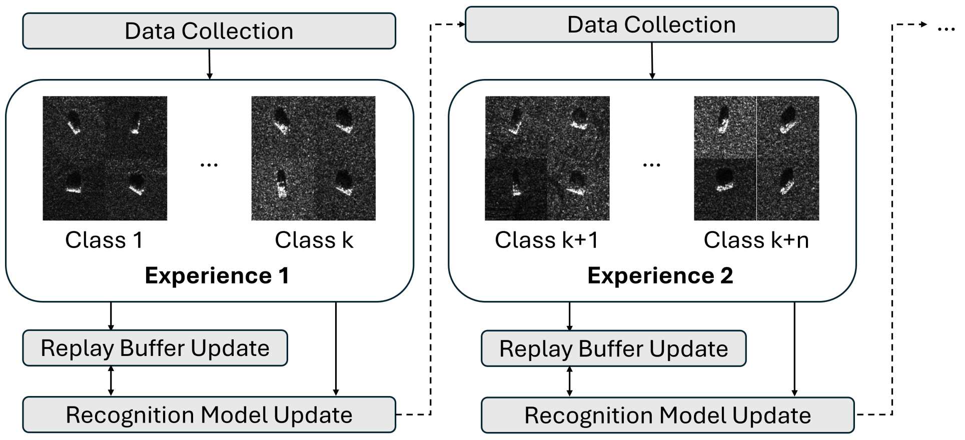


Figure 1. An illustration of the basic class-incremental SAR ATR problem with memory-based replay. Data is collected consisting of new classes. This data simultaneously updates the replay buffer (to preserve performance on classes that have been observed in the past) and recognition model (to adapt to new classes). As new data collection events occur, this process is repreats.

2.2 Noise-Tolerant Continual Learning

Continual learning introduces an iterative data collection-model update loop. This loop exposes a unique attack vector for corrupting model training if noisy samples are introduced during the data collection stages. Recent work has examined multiple directions of attacking continual learning agents. These approaches[36–42] are designed to i) induce forgetting, ii) prevent the learning of new tasks, and iii) inject backdoor triggers. Of particular relevance to this work, Li et al.[37] showed that continual learners are susceptible to label flipping attacks.

Several approaches[43–48] have sought to improve the noise tolerance of continual learning algorithms. Many of these works focus on adversarially-robust continual learning. Other works focus on label noise[49–52] and on non-adversarial measurement noise.[31] Existing approaches generally focus on one class of noise and generally rely on empirical validation as evidence of noise tolerance. We propose an approach for continual learning that is theoretically-motivated and empirically-validated to be robust to both label and measurement noise.

2.3 Machine Learning for SAR ATR and Incremental SAR ATR

ATR involves three components:[53] detection of the potential targets, discrimination to reduce false detections, and classification of the isolated target-of-interest. In this work, we are primarily concerned with the classification step. In the early days of machine learning-based SAR ATR, methods employed hand-crafted feature extraction[54] in combination with shallow machine learning models. This pipeline has since been replaced with deep learning-based approaches that simultaneously learn the feature extractor and classifier. We refer the reader to a recent survey paper by Li et al.[55] for a more complete treatment of the topic of deep learning applied to SAR ATR. Recently, there has been great interest[56–83] in developing ATR models that can learn incrementally. These methods vary widely, spanning many topics including few shot incremental learning, incremental open world/set classification, exemplar-based approaches, distillation-based approaches, and data augmentation/simulation-driven approaches, among many others. In contrast to the work presented in this paper, many of these approaches do not view the incremental SAR ATR problem through the lens of resilience to noisy data collection.

3. METHODS

3.1 Class-Incremental ATR Learning Problem

We first define the basic problem our algorithm is designed to address. There are multiple ways of formulating class-incremental learning problems. We first describe the most basic formulation, *disjoint incremental learning*, and then discuss a relaxation of the problem to *blurry incremental learning*. In most of our experiments, we evaluate using the blurry setting, but we have verified that Continual CRUST can also work under the disjoint setting in some cases.

3.1.1 Disjoint Incremental ATR

We provide a visual aid for the disjoint incremental ATR setting in Fig.1. In the disjoint incremental setting,[18] training data is partitioned into disjoint sets based on their class labels. The model training is broken up into a sequence of *learning experiences* with disjoint sets of classes per experience. Suppose there are six classes (A, B, C, D, E, and F), and each learning experience consists of two classes. Then, the system may collect data for classes A and B and then update the recognition model. Next, the system may collect data for classes C and D and then update the recognition model. Finally, the system may collect data for classes E and F and then update the recognition model. In each case, the model only has the capacity to store a few samples from the most recent experience before moving on to the next experience.

In practice, the partitioning of the training classes and the need for sequential learning would arise naturally. For example, over time, new vehicles are developed, so class sets can grow. Similarly, surveillance targets change over time, and different adversaries have access to different technologies, which result in different target sets.

We now provide the formal mathematical description of the problem. Continual learning addresses the challenge of learning from a dynamic stream of data that exhibit significant distribution shifts. Let $\{(x_t, y_t)\}_{t=1}^{T}$ be a sequence of input-output pairs (data and labels in the supervised case), where $x_t \in \mathcal{X}$ (the set of all training samples) and $y_t \in \mathcal{Y}$ (the label set of all training samples). The goal is to learn a function $f_\theta : \mathcal{X} \to \mathcal{Y}$ that minimizes:

$$\min_\theta \mathbb{E}_{(x,y)\sim p_t(x,y)}[\mathcal{L}(f_\theta(x), y)] \tag{1}$$

where $p_t(x, y)$ is the joint distribution at time t, and $\mathcal{L}$ is a loss function. In most CL settings, the learning process is divided into sequences of K tasks with associated probabilities $p_k(x, y)$. We can express the CL objective as:

$$\min_\theta \sum_{k=1}^{K} \mathbb{E}_{(x,y)\sim p_k(x,y)}[\mathcal{L}(f_\theta(x), y)] \tag{2}$$

This objective is subject to a constraint that performance on previous tasks does not significantly degrade:

$$\mathbb{E}_{(x,y)\sim p_j(x,y)}[\mathcal{L}(f_\theta(x), y)] \leq \epsilon_j, \quad \forall j < k \tag{3}$$

where ϵ_j represents the acceptable level of performance degradation on previous tasks.

3.1.2 Blurry Incremental ATR

Blurry incremental ATR[19] is a relaxation of disjoint incremental ATR. Differences between disjoint and blurry learning are highlighted in Fig. 2. Unlike disjoint learning where experiences are restricted to disjoint sets of classes between experiences, blurry learning consists of experiences that contain a mix of i) major classes, which make up the majority of the data within an experience, and ii) minor classes, which consist of a subset of the other potential classes and form a small proportion of data within an experience. In our experiments, the major classes are disjoint between experiences, and the minor classes always contain some samples from the remainder of all classes not within the major classes. However, our proposed algorithm is designed to trivially generalize to other other settings (e.g. with repeated major classes and with minor classes consisting of subsets of the remainder classes).

This is a more realistic reflection of what would likely occur in practice where data collection events may accidentally capture targets that are not currently of interest, but may become targets-of-interest in future data collects. Furthermore, we've empirically observed that training on blurry incremental learning scenarios benefits the coreset selection algorithms used to construct the replay buffers, especially when class differences are very subtle/fine-grained as is the case in the SynthWakeSAR dataset.

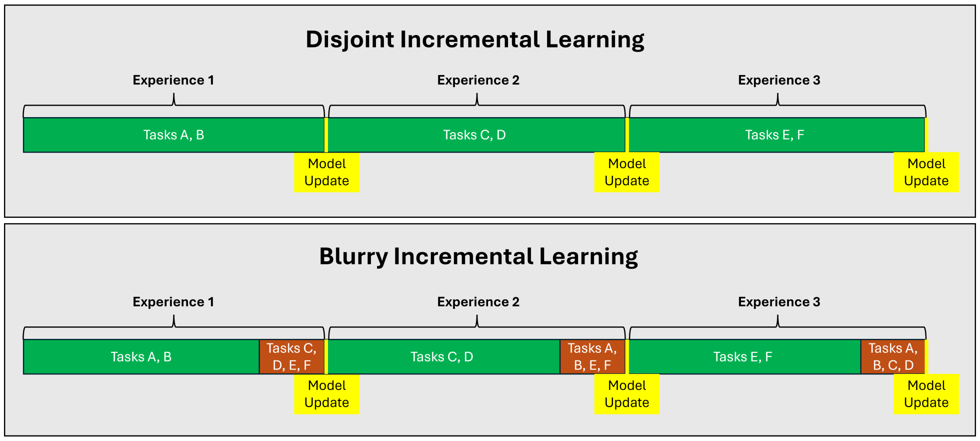


Figure 2. An illustration of the differences between disjoint and blurry incremental learning. Unlike disjoint learning where experiences are restricted to disjoint sets of classes, blurry learning consists of experiences that contain mix of i) major classes, which make up the majority of the data within an experience, and ii) minor classes, which consist of a subset of the other potential classes and form a small proportion of data within an experience.

3.2 Experience Replay for Class-Incremental Learning

The most straight-forward method for adapting machine learning models for new classes is to store the entire set of data encountered up to the current time and perform full model retraining. However, this is computationally-expensive in multiple ways. This requires saving all data samples from past classes, which can be a significant burden on memory. Furthermore, retraining on all data can be time-consuming and require many passes through the full dataset, which can require many compute cycles, blocking these compute resources from being used for other tasks and may result in increased energy consumption. An alternative approach is to finetune the model using just data from the new class. However, without special machinery (CL algorithms), this induces significant overfitting of the new class and catastrophic forgetting of past classes. To overcome these limitations, *experience replay* can be employed where a small, fixed-size memory (replay buffer) may be used to augment training during periodic model updates. Experience replay helps by storing samples from previous experiences/classes in a buffer after each data collection/model update phase. These samples are then replayed during the next model update, allowing the network to revisit and refine its knowledge about previous classes while integrating knowledge about new classes. While this idea is simple, there are a few components of the replay selection algorithm that define how effective this strategy is in practice:

1. How many samples need to be stored to successfully represent past observations?
2. How are the data samples selected (and more generally, how is the replay buffer periodically reconfigured) to maximize usefulness during subsequent replay periods?
3. How are samples drawn from the experience replay buffer and merged with samples from the new class during model updates?

Answering question (1) can be determined through experimentation, and In this work, we employ stratified sampling to address question (3). The solution to question (2) requires rigorous algorithmic formulation. We employ mathematically sound techniques to identify coresets that are representative of known classes that naturally filter out data samples that are uninformative or disrupt the learning process. Specifically, we adapt the CRUST technique,[1] originally developed for the static learning setting, to the continual learning setting.

3.3 Coresets for Robust Training of Neural Networks (CRUST)[1]

A coreset[84] S for a stationary dataset X w.r.t. to a loss function L is a weighted subset of X that approximates the loss of the full dataset for any set of parameters θ:

$$|\mathcal{L}(X,\theta) - \mathcal{L}(S,\theta)| \le \epsilon|\mathcal{L}(X,\theta)| \quad \forall\theta \tag{4}$$

where ϵ is a small error parameter.

CRUST[1] is a method developed by Mirzasoleiman, Cao, and Leskovec as a means of training of neural network (NN) classifiers that are theoretically and empirically robust to data containing noisy labels. The key idea of CRUST is to select subsets of clean and diverse data points that allow a NN to learn from the training data while avoiding overfitting to noisy labels. Examining the training dynamics of neural networks, one can observe that samples with clean labels tend to produce consistent and similar gradients w.r.t. the NN parameters. CRUST is built around two observed properties[85] of NN learning dynamics: i) learning along the singular vectors associated with the larger singular values of the Jacobian is fast and generalizes well, while learning along the singular vectors associated with smaller singular values tends to be slow and prone to overfitting; and ii) when label noise is present in a dataset, it tends to fall within the space of small singular values and impedes generalization. The original CRUST paper identifies coresets by finding the medoids of the data points that minimize average gradient dissimilarity to all other data points from a given training dataset, which as a result of the above properties, avoids overfitting noisy labels.

Let V denote the complete dataset. CRUST formulates the selection of a coreset S of size k as a k-medoids problem:

$$S^* = \arg\min_{S \subset V} \sum_{i \in V} \min_{j \in S} d_{ij} \quad \text{s.t.} \quad |S| \leq k \tag{5}$$

where $d_{ij} = \|\nabla \mathcal{L}(x_i, \theta) - \nabla \mathcal{L}(x_j, \theta)\|_2$ is the pairwise difference between gradients of sample i from V and j from S. Note that this optimization does not yield the optimal rank-k approximation of the Jacobian, but instead is a close approximation of the Jacobian *for* samples with clean labels. To further improve efficiency, CRUST uses a greedy iterative procedure exploiting the submodular structure of the problem.

Mirzasoleiman[1] proved that deep networks trained with gradient descent using the coresets output by CRUST, characterized by containing only a small fraction of data points with noisy labels, do not overfit the noisy labels:

THEOREM 3.1. ***(Robustness to Label Flipping)***[1] *Assume that gradient descent is applied to train a neural network with weights W using the mean-squared error (MSE) loss on a dataset of total size n containing some portion of samples with noisy labels. Suppose that the Jacobian of the network is L-smooth. Assume that the dataset has label margin of δ, and coresets found by CRUST contain a fraction of $\rho < \frac{\delta}{8}$ noisy labels. If the coreset approximates the Jacobian by an error of at most $\epsilon \leq \mathcal{O}(\frac{\delta\alpha^2}{k\beta\log(\sqrt{k}/\rho)})$, where $\alpha = \sqrt{r_{min}}\sigma_{min}(\mathcal{J}(W, X_S)))$, $\beta = \|\mathcal{J}(W, X)\| + \epsilon$, then for $L \leq \frac{\alpha\beta}{L\sqrt{2k}}$ and learning rate $\eta = \frac{1}{2\beta^2}$, after $\tau \geq \mathcal{O}(\frac{1}{\eta\alpha^2}\log(\frac{\sqrt{n}}{\rho}))$ iterations the NN classifies all samples in the Coreset correctly.*

where r_{min} is the size of the smallest cluster over the k-medoids, σ_{min} is the smallest singluar value, and the label margin δ is a dataset-specific constant.[86] Theorem 3.1 says that the number of iterations τ to fit the coreset is proportional to $\log\left(\frac{\sqrt{n}}{\rho}\right)$. As the noise fraction ρ increases, τ decreases for the same size of Coreset, which may initially seem counterintuitive. However, with more noise, the Coreset becomes less representative of the true data distribution, making it easier to fit (see robustness due to early stopping in[86]) but potentially less useful for generalization. Additionally, as ρ increases, the approximation error bound ϵ for the Jacobian matrix grows inversely on the order $\log\left(\frac{\sqrt{k}}{\rho}\right)$.

Practical incremental ATR systems are not limited to facing noise in the label space, but may also encounter perturbed measurements. In SAR ATR, measurement noise is extremely common in the form of speckle. In Appendix A, we prove that training on coresets discovered by CRUST is not only robust to label noise but is also theoretically robust to uncorrelated additive perturbations in the measurement (under similar assumptions of the previous theorem).

We provide the new bound for CRUST under general additive noise in the instances, i.e., $\tilde{X} = X + \mathrm{E}_X$. The following terms E_{J_1} and E_{J_2} appear in the bound and are associated with the Jacobian of the NN loss evaluated at the perturbation. For the Jacobian, we have

$$\begin{aligned} \mathcal{J}(W, X + E_X)^T &= \mathcal{J}(W, X)^T + \mathcal{J}(W, E_X)^T \quad (6) \\ &= \mathcal{J}(W, X)^T + E_{J_1} \quad (7) \end{aligned}$$

Definition 3.2. *(Average Jacobian)*[86] *We define the Average Jacobian along the path connecting two points* $u, v \in \mathbb{R}^p$ *as*

$$\bar{\mathcal{J}}(v,u) = \int_0^1 \mathcal{J}(u + \alpha(v-u))d\alpha$$

Let $\bar{\mathcal{J}}(v,u,x) = \bar{\mathcal{J}}(v,u)\Big|_x$ denote the average jacobian evaluated at x. Let $\hat{W} = W - \eta\nabla\mathcal{L}(W)$ denote a one-step gradient update to the NN weights. Then,

$$\bar{\mathcal{J}}(\hat{W}, W, X + E_X) = \bar{\mathcal{J}}(\hat{W}, W, X) + \bar{\mathcal{J}}(\hat{W}, W, E_X) \tag{8}$$

$$= \bar{\mathcal{J}}(\hat{W}, W, X) + E_{J_2} \tag{9}$$

where $E_{J_2} = \mathcal{J}(\hat{W}, W, E_X)$. Additionally, we define E_{min} and E_{max} as the numerical errors corresponding to the smallest and largest singular values, respectively. In contrast to Theorem 3.1, our bound depends on the residual error after T iterations of NN training on the Coreset.

Definition 3.3. *(Residual)* $r_T = f(X, W_T) - Y$.

Theorem 3.4. ***(Robustness to Measurement Perturbations)***
Assume that gradient descent is applied to train a neural network with mean-squared error (MSE) loss on a dataset with δ *fraction of perturbed data samples of the form* $\tilde{X} = X + E_X$. *Suppose the Jacobian is* L*-smooth. If the coreset approximates the Jacobian with an error of at most* $\epsilon \leq \mathcal{O}\left(\frac{\delta\alpha^2}{k\beta\log(\delta)}\right)$, *where* $\alpha = \sqrt{r_{min}}\sigma_{min}(\mathcal{J}(W, X_S)) - E_{min}$ *and* $\beta = \|\mathcal{J}(W,X)\|_2 + \epsilon + E_{max}$, *then for* $L \leq \frac{\alpha\beta}{L\sqrt{2k}}$ *and a learning rate* $\eta = \frac{1}{2\beta^2}$, *after*

$$T \geq \mathcal{O}\left(\frac{1}{\eta}\left(\frac{\alpha}{2} + E_{J_2}\alpha + E_{J_1}\alpha - \eta E_{J_2}\frac{\beta^3}{2} - \eta E_{J_1}\frac{\beta}{3}\right)^{-1}\log\frac{\|r_0\|_2}{\nu}\right)$$

iterations, $\nu > 0$ *is a constant that ensures that* $\|r_T\|_2 \leq \nu$, *the NN classifies samples in the Coreset correctly.*

We include the full proof of the theorem in Appendix A. In this case, convergence depends on both the size of the perturbations and the amount of data that exhibits perturbations. The theorem illustrates an important relationship between the fraction of perturbed data, δ, and the error in approximating the Jacobian, ϵ. As δ increases (data becomes noisier), the upper bound on ϵ also rises, but at a slower rate compared to the case of label noise. While noise does impact classification accuracy, its effect is more gradual. Similarly, as δ increases, the number of iterations required to fit the coreset decreases due to containing fewer clean examples available for learning. This decrease in iterations is not as rapid as seen with label noise.

3.4 Coresets for Robust *Continual* Training of Neural Networks (Continual CRUST)

In the previous section, we introduced the CRUST algorithm and discussed its theoretical robustness to both label noise and measurement noise. These noise-tolerance properties make CRUST an algorithm that is practical and well-suited for learning in the noisy settings often encountered by SAR ATR systems. However, CRUST is designed for stationary machine learning problems where the recognition model has knowledge of all potential target classes and access to the complete training set.

The major algorithmic contribution of this paper is the extension of the CRUST algorithm to the nonstationary, class-incremental SAR ATR setting. We leverage CRUST's ability to select clean, compact, and diverse coresets that are representative of the full dataset to serve as the mechanism for selecting samples for an experience replay buffer (introduced in Section 3.2). We denote our algorithm as *Continual CRUST*. The main idea behind Continual CRUST is to reconfigure the replay buffer after every data collection event (or more generally, within each learning experience) to find a coreset that can be used to update the model to preserve knowledge of past targets while incorporating information about new targets-of-interest.

Every time a data collection event occurs, it triggers a learning experience. Our training procedure consists of two phases:

1. **Phase 1 (Exposure to New Target Distribution):** The full data for the newest targets-of-interest is combined with the data from the CRUST-based replay buffer to finetune the neural network for feature extraction and target recognition with a standard cross-entropy loss function. When sampling batches of data, a weighted stratified sampler is used to ensure that the model sees diverse samples from the new class without completely biasing the model towards the new class (inducing forgetting). The purpose of phase 1 is to i) maximally expose the recognition model to the new data distribution to capture novel features of the new class and ii) seed the NN model with an "initial" state (w.r.t. the new class) that will lead to fast convergence for replay buffer reconfiguration by Continual CRUST.

2. **Phase 2 (Reconfiguring Replay Buffer):** The full data for the newest targets-of-interest is combined with past target data from the replay buffer up to the previous experience. CRUST is used to reconfigure the replay buffer, sampling target classes proportional to the levels they've been observed, so targets that have previously appeared as major classes are selected at a much higher rate than targets that have only appeared as minor classes up to the current learning experience. The neural network feature extraction and classification layers are then finetuned on only the replay data (which now includes the new targets) with a standard cross-entropy loss function. After every training epoch during phase 2, the replay buffer is incrementally reconfigured. Training accuracy is tracked, and the coreset that results in the highest accuracy becomes the replay buffer for the next learning experience.

We show pseudo-code for the basic version of our algorithm in Alg. 1.

3.5 Scattering Model-Guided Adversarial Attacks (SMGAA) for Poisoning Incremental SAR ATR Systems

We've described the main algorithmic contributions of our approach, a method for noise-tolerant class-incremental SAR ATR. We plan to evaluate our approach on three types of noise: label noise, measurement noise, and adversarial noise. How to create datasets for evaluating label and measurement noise is well-understood. However, since adversarial attacks of incremental SAR ATR systems is a new research area, we need to develop basic techniques for generating physically-plausible poisoning attacks.

Adversarial attacks can be broadly classified into two major categories:

- **Evasion Attacks:** This class of attack manipulates target appearance (either physically or in the measurement data) in order to fool a recognition model into misclassifying or misdetecting the samples. For example, an adversary might want to modify the appearance of a tank in order to fool the recognition model into thinking that specific instance of the tank is a truck. Importantly, evasion attack models have no interaction with the training procedure of the recognition model, and to formulate the attack often requires using a proxy recognition model to approximate the behavior of the true recognition model.

- **Poisoning Attacks:** In contrast, poisoning attacks try to manipulate target appearance during the data collection phase in order to corrupt the learning behavior of the recognition model (e.g. insert backdoor triggers that cause the model to misclassify targets when certain patterns appear in the measured data or induce forgetting of specific targets to avoid future detections). In this case, the adversary has knowledge of when a data collection might occur, but is still unlikely to have direct access to the recognition model and likely requires using a proxy recognition model in the formulation of the attack.

Our experiments w.r.t. adversarial attacks focus on poisoning attacks since the nature of continual learning introduces many opportunities for an adversary to corrupt recognition models.

There is a lot of work on evasion[87] and poisoning attacks[88] for electro-optic imagery, including for specifically attacking continual learning algorithms.[36–42] However, these attacks make assumptions that are ill-suited for SAR ATR. Specifically, these attacks often occur as a posthoc manipulation of an already-captured image. This is sometimes reasonable for the electro-optic domain where image datasets are often collected via mining the Internet. In defense applications, data collection and ML model sharing are much more restricted dues to national security concerns. As such, in the SAR ATR domain, manipulations must occur in the physical domain (e.g.

Algorithm 1 Continual CRUST

```
1: **Given**: Neural network $f_\theta$ with $L$ layers and trainable parameters $\theta_1, \ldots, \theta_L$.
2: **Input**: Dataset for the current experience $\mathcal{D}_i = (X_i, Y_i)$ consisting of measurements $X_i$ and corresponding
   target labels $Y_i$.
3: **Hyperparameters**: Learning rate $\eta$; Fixed coreset size $s$; Phase 1 epochs $e_1$; Phase 2 epochs $e_2$; Phase 1
   batch iterations $b_1$; Phase 2 batch iterations $b_2$;.
4: **Learns**: Replay buffer (coreset) $S$ (initially empty) and updates $f_\theta$ (initialized randomly or from pre-training
   on large dataset such as ImageNet).
5: for $i$ in CIL Experiences do
6:     repeat $e_1$ times
7:         repeat $b_1$ times
8:             $D_b \leftarrow$ stratefied_sampler($D_i \cup S_{i-1}$)                      ▷ Sample batch
9:             $f_\theta \leftarrow$ SGD($\mathcal{L}, \theta, \eta, D_b$)                     ▷ Update neural network
10:        end
11:    end
12:    $S_{i-1} \leftarrow S_i$                                         ▷ Save previous experience replay buffer
13:    repeat $e_2$ times
14:        $S_i \leftarrow \emptyset$
15:        for $c$ in Observed Target Classes do
16:            $S_c \subseteq S_{i-1}$ for data of class $c$ in previous experience replay buffer $S_{i-1}$
17:            $D_c \subseteq D_i$ for data of class $c$ in current experience dataset $D_i$
18:            $J_c \leftarrow S_c \cup D_c$
19:                    ▷ Merge instances of class $c$ from replay buffer and current experience into joined dataset
20:            $G_c \leftarrow \nabla(\mathcal{L}_{CE}(J_c), \theta_{L-1})$            ▷ Calculate gradient w.r.t. last layer params.
21:            $s_c \leq s \leftarrow$ compute_subcoreset_size($c$)
22:                        ▷ Compute number of samples to allocate to class $c$ in the experience replay buffer
23:            $S'_c \leftarrow$ CRUST($J_c$, $G_c$, $s_c$)
24:                                ▷ Perform CRUST on class $c$ dataset to select samples for replay buffer
25:            $S_i \leftarrow S_i \cup S'_c$
26:        end for
27:        repeat $b_2$ times
28:            $D_b \leftarrow$ sampler($S_i$)                                      ▷ Sample batch
29:            $f_\theta \leftarrow$ SGD($\mathcal{L}, \theta, \eta, D_b$)                     ▷ Update neural network
30:        end
31:    end
32: end for
```

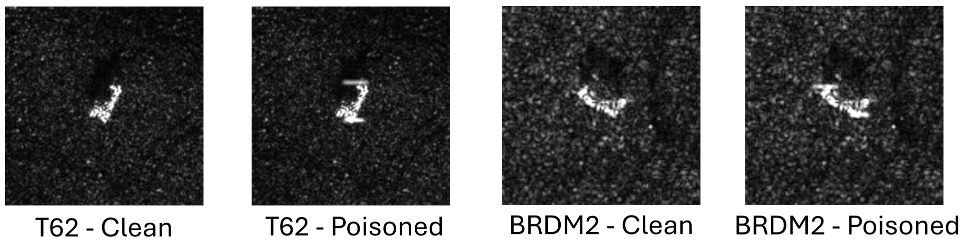


Figure 3. An example of inserting a backdoor trigger into the data samples in a physically-plausible manner. In this case, a trigger was identified that successfully targeted the T62 class. It was then inserted into the same class and a different class (BRDM2).

by manipulating the appearance of the target). There has been some work on developing physically-plausible *evasion* attacks for SAR ATR, generally by manipulating the electromagnetic scattering response of the target[4] and by hiding the attacks in the speckle.[9] However, since incremental SAR ATR is a topic that has only gained significant interest in the last few years, few techniques exist that focus on poisoning attacks in the incremental SAR ATR setting. As such, we repurpose an existing physically-plausible evasion attack for the purposes of inserting *backdoor triggers*[10] with limited but promising success.

We leverage the "Scattering Model Guided Adversarial Attack" (*SMGAA*)[4] algorithm of Peng et al for intelligently generating backdoor triggers as the basis for our poisoning attack. A backdoor trigger is a specific engineered input pattern that a trained model has been corrupted to strongly associate with a specific target. When the recognition model encounters a target with an active trigger, it will misclassify the target, but in when the trigger is absent, the recognition model performs as it normally would.

SMGAA defines a specific parametric scattering model, the Attributed Scattering Center Model,[89] and a general imaging method to describe the scattering behavior of typical geometric structures in the SAR imaging process. In the original SMGAA paper, a gradient-based optimization scheme is employed to tune the parameters of the scattering model for individual instances of targets of interest. The goal is to generate a response that causes a proxy recognition model to misclassify the specific instance of the target-of-interest in order to evade recognition.

For our use case, we want to find a set of *common parameters* of a *generic* scattering response (and its associated geometric structure) that maximally reduces the confidence of predicting a specific target averaged over many instances of the target (i.e., minimizes the expected confidence for a specific target class). By finding a trigger that lowers the confidence in predicting the target class *in general*, the hope is that the trigger will overpower the discriminatory features that the model would otherwise utilize, strengthening the association between the trigger and target class. This is a common idea utilized by existing algorithms for generating backdoor triggers. Then, during the data collection process, a large set of instances will be poisoned with the trigger (e.g. by attaching the physical scatterer capable of producing the desired response to the target-of-interest), encouraging the recognition model to learn a spurious association between the scatter pattern and specific target class.

Note that in our experiments, we simulate the aforementioned poisoning process entirely via simulation using the code base provided by SMGAA.[4] Also note, that we are <u>not</u> trying to insert *hidden* backdoor triggers, which are designed to be extremely difficult to detect. This may be an interesting future direction to consider.

While SMGAA uses gradient descent to optimize the scatter pattern, we employ a different technique. We leverage the Optuna library[90] to perform Bayesian optimization via Tree-Structured Parzen Estimators (TPEs).[91] The optimization procedure to generate the poisoned dataset is simple:

1. A proxy recognition model is trained on the full static training dataset. In our experiments, this involves training an EfficientNetV2 model[92] on the training split of the MSTAR dataset. Note that the proxy model is intentionally chosen to not match the true recognition model (a small MobileNet model[93]) since it cannot be assumed the adversary has access to the real recognition model.

2. The optimization system starts with an initial set of random parameters of the scattering model and a specific target class to attack. In our experiments, we are optimizing the parameters of two distinct scatterers placed along the boundaries of the object (obtained via the SARBake overlays[94] for the MSTAR dataset).
3. All of the training data is poisoned according to the current scattering model parameters.
4. Each poisoned sample is passed through the proxy recognition model to obtain confidence scores associated with predicting each target class. The average prediction confidence for the true class is recorded and used as the metric (objective function) for evaluating the success of the attack.
5. The TPEs model the multivariate distribution (via kernel density estimation) that associate the parameters of the scattering model with the objective function. The TPEs then define how to sequentially sample from this distribution to intelligently search the parameter space to find a good solution to the optimization problem, balancing exploration (selecting samples to improve the density estimation) and exploitation (selecting samples likely to produce stronger attacks).
6. Every time the TPEs select a new set of parameters, the optimization loop repeats.

Once the optimization process terminates after a fixed number of iterations (200 iterations in our experiments), the most effective set of scattering parameters is saved. This information can then be used to create poisoned datasets with varying amounts of manipulated samples in a controlled manner.

Ultimately, we find that this method is effective at finding useful triggers for some target classes but fails for other classes. See Section 4.11 for more details. In our experiments, we target the T62 class. We show examples of the learned trigger applied to the T62 and BRDM2 classes in Fig. 3.

4. EXPERIMENTS AND RESULTS

To understand the problem of noisy incremental SAR ATR and to evaluate the proposed approach, we conduct a diverse set of experiments on the benchmark MSTAR[2] and SynthWakeSAR[3] datasets. These experiments aim to understand the following:

- **Section 4.5:** How challenging is the incremental SAR ATR problem compared to the static SAR ATR problem?
- **Section 4.6:** Does the addition of label noise make incremental SAR ATR a meaningfully more challenging problem?
- **Section 4.7:** Are existing continual learning algorithms (including Continual CRUST) empirically robust to label noise?
- **Section 4.8:** Does the addition of measurement noise make incremental SAR ATR a meaningfully more challenging problem?
- **Section 4.9:** Is Continual CRUST empirically robust to measurement noise?
- **Section 4.10:** Are Scattering Model Guided Adversarial Attacks effective for constructing backdoor triggers for the incremental SAR ATR problem?
- **Section 4.11:** Is Continual CRUST empirically sensitive to adversarial noise in the form of poisoning attacks (specifically, backdoor trigger attacks)?

Before we describe each experiment and the associated findings, we first describe the general experimental setting.

4.1 Datasets

We run experiments on two benchmark datasets for SAR ATR: MSTAR[2] and SynthWakeSAR,[3] which we briefly describe:

- **MSTAR:** The "Moving and Stationary Target Acquisition and Recognition" (MSTAR) dataset is a widely used benchmark dataset for SAR ATR and contains X-band SAR imagery of defense-relevant vehicles. The dataset consists of ten classes (2S1, BMP2, BRDM2, BTR60, BTR70, D7, T62, T72, ZIL131, ZSU23) composed of tanks, personnel carriers, trucks, bulldozers, self-propelled howitzers, and anti-aircraft artillery. The version of the dataset that we use for our experiments consists of 2,747 training images and 2,426 test images, split based on depression angle and with roughly evenly proportioned numbers of samples for the ten classes. We resize images to be 224x224 for easy processing by a standard convolutional neural network architecture. For generating poisoned data, we make use of the SARBake overlays[94] to ensure scatterers are placed on top of the targets-of-interest.
- **SynthWakeSAR:** The SynthWakeSAR dataset consists of synthetic SAR images and is intended for building models capable of classifying ship type based on ship wake signatures. SynthWakeSAR has matched noise-free, realistic speckle, and despeckled data samples. In our experiments, we utilize the noise-free and realistic speckle variants and in some cases, augment the dataset by artificially adding additional measurement noise. Like MSTAR, SynthWakeSAR is composed of ten classes, consisting of variants of cargo ships, fishing vessels, high-speed craft, passenger vessels, and tankers. For our experiments, we use a subset of the dataset consisting of 32,260 images for training and 13,820 test images. As with MSTAR, we resize all images to 224x224 for compatibility with the standard convolutional neural network architecture used in our experiments.

4.2 Baseline Algorithms

While there are many recent papers focused on incremental SAR ATR published in the last few years (refer to Section 2.3), for our experiments, we choose to compare against standard baselines and state-of-the-art algorithms from the broader continual learning literature as these algorithms are more readily available and generalize easily to new evaluation settings. In future work, it would be beneficial to compare against algorithms specifically designed for incremental SAR ATR. The algorithms we compare against include:

- **Joint Learner:** The joint learner concatenates the data from all of the learning experiences and trains a single model, converting the continual learning problem into a static machine learning problem. This baseline serves as an approximation of the upperbound of what a continual leaner may be able to achieve w.r.t. accuracy.
- **Joint Learner + CRUST:** This baseline is a modification of the joint learner by applying CRUST on top of the full dataset. This baseline is useful for approximating how much performance is lost by compressing the full dataset into a coreset.
- **Naïve Sequential Learner:** The naïve sequential learner trains sequentially on one experience at a time without any special machinery for continual learning. This often results in significant overfitting to the most recently seen classes since the model is unable to revisit past experiences. This baseline approximates a lower bound on the continual learning problem and demonstrates the need for specific algorithms for continual learning.
- **Random Replay:** Random replay is the simplest form of experience replay where data samples are selected for the replay buffer using uniform random sampling after each learning experience.
- **Dark Experience Replay (DER):**[17] DER is a popular algorithm in the continual learning literature that mixes rehearsal (replay) with knowledge distillation and regularization. It works by trying to force the current network's logits to match those sampled throughout the optimization trajectory, which has the effect of encouraging stability of the learning process and maintenance of performance on past tasks. We employ the implementation within the Avalanche framework[95, 96] for continual learning.

- **DER++:[17]** DER++ is an extension of Dark Experience Replay, which adds an additional term to the objective function to stabilize learning when distribution shifts are very sudden. We employ the implementation within the Avalanche framework.[95,96]
- **Reservoir Sampling (RSV)[97] + DivideMix:[98]** Reservoir sampling is a simple technique for selecting data samples for the experience replay buffer and solves the problem of keeping some limited number m of n total items seen before with equal probability $\frac{m}{n}$ when n isn't known in advance. Following the experimental procedure of Bang et al.,[50] we combine RSV with DivideMix,[98] a technique for learning with noisy labels for the static learning scenario, to improve the robustness of RSV under label noise.
- **Greedy Sampling (GS)[99] + DivideMix:** Given a fixed memory budget, this sampler greedily stores samples from the data-stream in such a way that asymptotically balances the class distribution. Following the experimental procedure of Bang et al., we combine GS with DivideMix.
- **Rainbow Memory (RM)[100] + DivideMix:** Rainbow memory is designed for the blurry-CIL setting. Samples are selected for the memory by measuring per-sample robustness against perturbations. Diversity in the memory is improved by sampling both perturbation-robust and more fragile data that helps the models to preserve discriminative boundaries for each class. Following the experimental procedure of Bang et al., we combine RM with DivideMix.
- **PuriDivER:[50]** PuriDivER is a recent method designed for blurry-CIL. PuriDivER specifically attempts to balance diversity of samples with purity (w.r.t. label noise) in the episodic memory, and proposes a label noise-aware sampling method.

Note that we modify the code implementations from the PuriDivER code base[50] to extend RSV, GS, RM, and PuriDivER from streaming-based CL to batch-based CL.

4.3 Metrics

We consider several metrics when evaluating the baseline and proposed approaches on the incremental SAR ATR problem. Let R_i, j be the test accuracy on class j after training on class i.

- **Final Experience Average Accuracy:** [16,101] Let T be the final experience. After training on all experiences, $acc\_final = \frac{1}{T}\sum_{j=1..T} R_{T,j}$.
- **Forgetting:** [16] Forgetting measures the loss of performance on previous experiences after learning new experiences. We define $forg = \frac{1}{|C|}\sum_{i=j+1..T}\sum_{j=1..T} Rj, j - R_{i,j}$ where $|C|$ is the number of pairwise comparisons in the double summation. If $forg > 0$, the model has lost performance.
- **Memory Purity**: Purity represents the proportion of clean samples selected within the replay memory, where $purity = 1$ implies noiseless memory.
- **Poisoned Class Precision:** For the adversarial noise case, we want to understand how effective the poisoning attack was w.r.t. corrupting our continual learner. To understand this, we look at two metrics. The first of these metrics looks at the precision ($\frac{true\_positives}{true\_positives+false\_positives}$) of predicting the poisoned class. If precision is low, it means the recognition model is confusing other classes with the poisoned class, so the poisoning attack was successful.
- **Poisoned Class Recall:** The second metric we consider for evaluating adversarial robustness is the poisoned class recall ($\frac{true\_positives}{true\_positives+false\_negatives}$). If this number is low, it means we've corrupted the continual learner's ability to recognize the poisoned class, which means the target can successfully evade recognition.

Table 1. Per learning experience training loop epochs for various CL algorithms.

Method(s)	**Static Training on Full Dataset**	**Phase 1: New Data: Full Past Data: Coresets**	**Online Streaming for Memory Construction**	**Phase 2: New Data: Coresets Past Data: Coresets**
Joint Learner, Joint Learner + CRUST	80 Epochs (Note: One Learning Experience)			
Naive Sequential Learner, Random Replay, Continual CRUST		40 Epochs (Measurement Noise Ablations: 20 Epochs)		20 Epochs (Measurement Noise Ablations: 10 Epochs)
DER, DER++		60 Epochs		
Reservoir Sampling, Greedy Sampling, Rainbow Memory, PuriDivER		40 Epochs	1 Epoch	40 Epochs (Note: Batch Size: 50)

4.4 Experimental Setup

Neural Network Training Hyperparameters: We use a MobileNet[93] convolutional neural network as our recognition model, initialized with weights trained on ImageNet[102] (general electro-optic imagery). This model was selected because it provides a good balance between efficiency and strong discriminative ability. Unlike some work in CIL, our approach involves learning both the feature extractor and classifier layers of the network. Imagery is resized to 224x224 pixels and expanded to the RGB color space for compatibility with existing NN architectures. Every time a batch of data is sampled, data augmentations are applied using the auto-augmentation code of Bang et al.,[50] and the data samples are normalized using precomputed mean and standard deviation values. For most experiments, we use a standard ADAM optimizer[103] with a fixed learning rate of $1.0e{-4}$ and weight decay of $1.0e{-5}$. For the "disjoint" setting on MSTAR, we use a higher learning rate of $1.0e{-3}$. A step scheduler reduces the learning rate by a factor of 0.9 every epoch. Optimizers are reset between every learning experience and between phase 1 and phase 2 as described in Section 3.4.

Continual Learning Hyperparameters: Slightly different training procedures are followed based on the CL algorithm. We use ten instances per minibatch unless otherwise noted. We report the number of epochs for different stages of the per learning experience training loop for different continual learning algorithms in Table 1. It is important to note that ***for experiments involving creating a replay buffer, we use a fixed 200 sample buffer for MSTAR and a fixed 2000 sample buffer for SynthWakeSAR***. These sizes were determined via experimentation as described in Appendix B. We use default hyperparameters for the CL algorithms as defined in their code implementations from Avalanche and the PuriDivER code base unless otherwise noted.

Curriculum Design: Most experiments focus on the blurry class-incremental learning setting; although, we do run an additional experiment on MSTAR for disjoint class-incremental learning. ***The standard curriculum (ordering of learning experiences) we employ involves five experiences with two major classes per experience with no repeated major classes under the blurry setting with 90% of data within each experience sampled from the major classes and 10% sampled from the remaining classes.*** Experiments use this setting unless otherwise noted. To study how the model performs for longer curriculum in the more challenging one-class incremental learning setting,[104] we run an additional experiment on MSTAR where we seed the first experience with two classes and add an only a single class at a time for eight additional experiences.

Experimental Trial Settings: For each MSTAR-based experiment, we run five trials with different random seeds (defining different task orderings, batch orderings, model parameter instantiations, etc.) per experimental setting. For each SynthWakeSAR-based experiment, we run three trials with different random seeds per experimental setting (due to the much larger dataset size). In most experiments, we control the level of noise, typically considering the noise-free, 20% corrupted, and 40% corrupted settings.

Hardware and Software: Our code is developed in PyTorch with GPU acceleration. We use the Avalanche framework[95, 96] for continual learning to manage our experiments and for implementations of DER and DER++. The PuriDivER codebase[50] is used for data auto-augmentation and for implementations of reservoir sampling, greedy sampling, rainbow memory, and PuriDivER. Many experiments are run in parallel on a cluster of compute nodes with a range of intel-based server-grade CPUs (48-96 cores per node) and NVidia GPUs (2080Tis and A5000s) with large amounts (32GB+) of RAM. Note that in practice, our approach can easily run on much more limited hardware resources (e.g. consumer-grade laptops and embedded systems).

4.5 Research Question: How Challenging is the Incremental SAR ATR Problem Compared to the Static SAR ATR Problem?

The first question to be addressed is whether existing SAR ATR approaches can be directly leveraged for the incremental SAT ATR problem, or is there something inherently more challenging about the incremental setting that requires special machinery? The machine learning literature[13] suggests that continuous finetuning of a machine learning model across distribution shifts results in significant catastrophic forgetting in most domains. To verify that this hypothesis holds in the incremental SAR ATR domain, we train two models, a “joint learner”, and a “naïve sequential learner” on a curriculum without any added noise. As a reminder, the joint learner trains on all available data during a single learning experience, which is equivalent to the “static” machine learning problem, and the naïve sequential learner trains on a sequence of learning experiences with different major classes per experience without revisiting any past data. We run experiments on the standard curriculum defined above for both MSTAR and SynthWakeSAR. Results appear in the “Noise=0.0” columns of Tables 2 and 3. For MSTAR, by switching from the static setting (joint learner) to the continual setting (naïve sequential learner), we see a drop in accuracy of 10.3%. For SynthWakeSAR, we see a larger drop of 16.4%. While these are not fatal drops in performance, they still represent a sizable loss, and we’d expect to see bigger drops on more challenging datasets and curricula. This suggests that incremental SAR ATR is a non-trivial problem. We will see in future sections, that incremental SAR ATR becomes even more challenging when combined with certain types of noise.

Table 2. **Standard Blurry Curriculum - MSTAR.** Evaluating how different algorithms are affected as the proportion of label noise increases on the MSTAR dataset. This setting involves a blurry CIL curriculum with five experiences with two major classes per experience and varying amounts of label noise. The replay memory size is fixed at 200 samples.

	Noise=0.0		**Noise=0.2**		**Noise=0.4**	
Algorithm	Accuracy	Forgetting	Accuracy	Forgetting	Accuracy	Forgetting
Joint Learner	0.976 ±0.007	N/A	0.941 ±0.006	N/A	0.892 ±0.013	N/A
Joint Learner + CRUST	0.956 ±0.014	N/A	0.921 ±0.008	N/A	0.865 ±0.021	N/A
Naive Sequential Learner	0.873 ±0.033	0.119 ±0.056	0.655 ±0.056	0.264 ±0.05	0.449 ±0.045	0.396 ±0.056
Random Replay	0.95 ±0.014	-0.046 ±0.018	0.819 ±0.036	-0.016 ±0.031	0.592 ±0.024	0.046 ±0.032
DER	0.966 ±0.012	0.014 ±0.012	0.791 ±0.019	0.209 ±0.044	0.525 ±0.016	0.494 ±0.037
DER++	0.958 ±0.008	0.028 ±0.018	0.626 ±0.041	0.393 ±0.063	0.364 ±0.027	0.693 ±0.024
Reservoir Sampling + DivideMix	0.902 ±0.075	-0.121 ±0.071	0.826 ±0.013	-0.081 ±0.091	0.616 ±0.055	-0.084 ±0.073
Greedy Sampler + DivideMix	0.919 ±0.01	0.002 ±0.035	0.777 ±0.052	0.032 ±0.065	0.601 ±0.029	0.083 ±0.063
Rainbow Memory + DivideMix	0.909 ±0.025	-0.019 ±0.065	0.735 ±0.083	0.11 ±0.026	0.6 ±0.043	0.132 ±0.064
PuriDivER	0.899 ±0.031	0.055 ±0.03	0.72 ±0.033	0.203 ±0.085	0.536 ±0.041	0.277 ±0.056
Continual CRUST	0.944 ±0.016	-0.049 ±0.018	0.895 ±0.02	-0.053 ±0.035	0.743 ±0.046	-0.048 ±0.061

Table 3. **Standard Blurry Curriculum - SynthWakeSAR.** Evaluating how different algorithms are affected as the proportion of label noise increases on the SynthWakeSAR dataset. This setting involves a blurry CIL curriculum with five experiences with two major classes per experience and varying amounts of label noise. The replay memory size is fixed at 2000 samples.

	Noise=0.0		**Noise=0.2**		**Noise=0.4**	
Algorithm	Accuracy	Forgetting	Accuracy	Forgetting	Accuracy	Forgetting
Joint Learner	0.935 ±0.002	N/A	0.902 ±0.003	N/A	0.85 ±0.001	N/A
Joint Learner + CRUST	0.908 ±0.001	N/A	0.847 ±0.001	N/A	0.778 ±0.003	N/A
Naive Sequential Learner	0.771 ±0.024	0.221 ±0.023	0.503 ±0.014	0.438 ±0.028	0.326 ±0.015	0.544 ±0.025
Random Replay	0.874 ±0.008	0.048 ±0.006	0.674 ±0.003	0.125 ±0.017	0.487 ±0.009	0.211 ±0.011
DER	0.895 ±0.006	0.093 ±0.012	0.682 ±0.018	0.285 ±0.029	0.481 ±0.008	0.474 ±0.02
DER++	0.856 ±0.012	0.144 ±0.019	0.562 ±0.03	0.447 ±0.032	0.349 ±0.017	0.643 ±0.027
Reservoir Sampling + DivideMix	0.802 ±0.01	-0.1 ±0.041	0.559 ±0.008	0.049 ±0.042	0.389 ±0.011	0.063 ±0.025
Greedy Sampler + DivideMix	0.81 ±0.01	-0.144 ±0.013	0.573 ±0.009	-0.043 ±0.076	0.418 ±0.021	0.084 ±0.063
Rainbow Memory + DivideMix	0.726 ±0.012	0.115 ±0.035	0.571 ±0.034	0.184 ±0.029	0.455 ±0.006	0.189 ±0.013
PuriDivER	0.812 ±0.012	0.155 ±0.018	0.599 ±0.021	0.249 ±0.039	0.442 ±0.006	0.321 ±0.039
Continual CRUST	0.851 ±0.009	0.053 ±0.014	0.72 ±0.012	0.097 ±0.029	0.532 ±0.013	0.174 ±0.016

4.6 Research Question: Does Label Noise Impact the Incremental SAR ATR Problem?

Next, we aim to understand if combining continual learning with label noise has a compounding effect on the difficulty of the learning problem. Following the standard experimental curriculum, we look at how the joint learner and naïve sequential learner perform as we increase the proportion of label noise for the MSTAR and SynthWakeSAR datasets. Initially, data is sampled with perfect label information. Then, it is increased so 20% of samples are mislabelled (noisy samples are selected uniformly randomly; i.e., noise is symmetric across classes). Finally, we mislabel 40% of the data. Results appear in Tables 2 and 3.

For the joint learner, going from zero noise to 20% noise, we see a 3.5% drop on MSTAR and 3.3% drop on SynthWakeSAR. Going from zero noise to 40% noise, we see an 8.4% drop in accuracy on MSTAR and 8.5% drop on SynthWakeSAR. This suggests that the ATR problem does become more challenging as label noise increases. However, even under extreme label noise, for the static learning setting, the ATR models are still able to largely learn around the noise, not exhibiting a fatal loss of accuracy.

Results are much more stark when looking at the naïve sequential learner. Increasing from zero noise to 20% noise, we see a 21.8% drop on MSTAR and 26.8% drop on SynthWakeSAR. Going from zero noise to 40% noise, we see a 42.4% drop in accuracy on MSTAR and 44.5% drop on SynthWakeSAR. Label noise in combination with incremental learning causes a catastrophic loss in recognition performance, providing evidence that noisy incremental SAR ATR is a major problem worthy of additional study.

4.7 Research Question: Are Continual Learning Algorithms Robust to Label Noise?

We've seen that label noise in incremental SAR ATR is a significant problem. This prompts us to ask, are there any standard and state-of-the-art methods that are robust to label noise? Specifically, Continual CRUST is theoretically motivated w.r.t. robustness to label noise, do these motivations translate to practical (i.e. empirically-measurable) robustness? We investigate this question from several different perspectives including the i) shorter vs longer (one-class) curricula, ii) the blurry vs disjoint setting, and iii) label noise resulting from improperly filtered background measurements.

4.7.1 Blurry Incremental SAR ATR w/ Label Noise: Standard Curriculum

We examine how a number of common and state-of-the-art baselines perform on the standard curriculum described earlier. If we look at the results in Tables 2 for MSTAR and 3 for SynthWakeSAR, we see several interesting trends. Unsurprisingly, in general, employing *any* machinery for continual learning improves performance over the naïve sequential learner. Some methods do outperform others. In particular, our proposed Continual CRUST method outperforms the second best CL method by at least 4% in terms of accuracy in all cases when at least 20% of samples are mislabelled and is competitive with other approaches in the noise-free setting. It also tends to exhibit less forgetting than other high-performing approaches.

In general, we are a bit surprised to see methods designed specifically for label noise (i.e., Reservoir Sampling + DivideMix, Greedy Sampler + DivideMix, Rainbow Memory + DivideMix, PuriDiVER) do not always outperform generic CL algorithms like DER and random replay. This is surprising as in other domains, these methods (especially PuriDivER) are state-of-the-art for noisy label continual learning. One explanation for lackluster performance of these methods is that the implementations used in this work are based on versions originally designed for *streaming* settings and our extension to batch-based learning might not be optimal. However, if we look at the purity of replay buffers learned by these approaches (Fig. 4), we see that for the methods where the replay is specifically designed to filter out noisy samples (rainbow memory, PuriDivER, Continual CRUST), in most cases, there is a marked improvement in purity compared to those that do not prioritize filtering out noisy samples. This suggests that the models are in fact learning to separate clean from noisy samples, but i) the replay buffers may be focusing on samples that are too "easy" (i.e. not diverse w.r.t. the challenging clean samples) or ii) the models may be overfitting to the samples in the small replay buffer. Interestingly, Continual CRUST finds purer coresets for the replay buffers while achieving higher performance than standard CL algorithms.

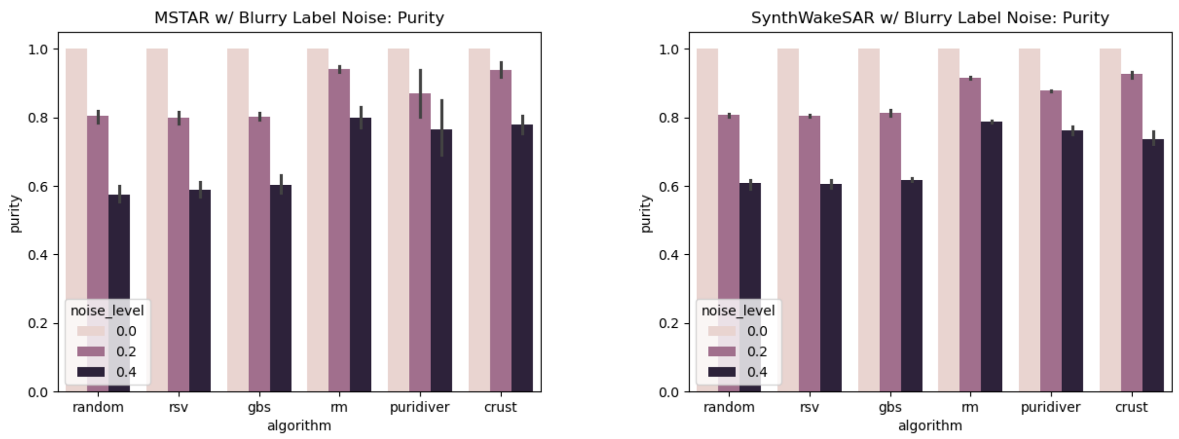


Figure 4. Evaluating how different algorithms are effected in terms of purity as the proportion of label noise increases on the MSTAR (left) and SynthWakeSAR (right) datasets. This setting involves a blurry CIL curricula with five experiences with two major classes per experience and varying amounts of label noise. The replay memory size is fixed at 200 samples for MSTAR and 2000 samples for SynthWakeSAR.

4.7.2 One-Class Blurry Incremental SAR ATR w/ Label Noise

The five tasks, two major classes per task setting is a relatively easy setting for incremental SAR ATR. We want to see if the same trends occur when evaluating on a more challenging curriculum. In this section, we consider the one-class blurry incremental SAR ATR problem, where during the first experience, the system is initialized with two major classes, but for every subsequent experience, only one additional major class appears. This makes the problem more challenging for two reasons: i) the curriculum length is approximately doubled, which means the recognition model needs to maintain knowledge about classes for longer stretches, i.e. there are more opportunities for the model to succumb to catastrophic forgetting and ii) there is less information available for the recognition model to utilize within each experience because most data only comes from a single major class, which makes learning effective decision boundaries a bigger challenge. This is also realistic of potential practical situations, i.e. where data collects are focused on a specific new target-of-interest, and information about the new target class needs to be integrated with an existing recognition model. Results are reported for MSTAR in Table 4 and Fig. 5. Trends are very similar to those of Section 4.7.1. On average, for most methods, we see a small (few points) decrease in final experience accuracy. Furthermore, we see higher purity for methods that specifically aim to filter out noisy data. This suggests that most of these methods are at least somewhat robust to more challenging operating conditions; albeit, it should be noted that MSTAR is a relatively easy dataset from a machine learning perspective.

Table 4. **One-Class Blurry Curriculum - MSTAR.** Evaluating how different algorithms are affected as the proportion of label noise increases on the MSTAR dataset. This setting involves a blurry CIL curriculum with nine experiences with the first experience containing two major classes with only a single major class for each subsequent experience and varying amounts of label noise. The replay memory size is fixed at 200 samples.

	Noise=0.0		**Noise=0.2**		**Noise=0.4**	
Algorithm	Accuracy	Forgetting	Accuracy	Forgetting	Accuracy	Forgetting
Joint Learner	0.973 ±0.009	N/A	0.936 ±0.029	N/A	0.878 ±0.036	N/A
Joint Learner + CRUST	0.932 ±0.028	N/A	0.924 ±0.024	N/A	0.855 ±0.037	N/A
Naive Sequential Learner	0.833 ±0.048	0.157 ±0.051	0.606 ±0.079	0.337 ±0.064	0.437 ±0.078	0.455 ±0.04
Random Replay	0.947 ±0.018	-0.077 ±0.032	0.822 ±0.02	-0.028 ±0.032	0.591 ±0.016	0.089 ±0.04
DER	0.97 ±0.024	0.029 ±0.036	0.853 ±0.051	0.155 ±0.03	0.62 ±0.03	0.385 ±0.057
DER++	0.948 ±0.03	0.049 ±0.042	0.717 ±0.056	0.297 ±0.033	0.454 ±0.051	0.571 ±0.065
Reservoir Sampling + DivideMix	0.918 ±0.024	-0.291 ±0.035	0.766 ±0.054	-0.257 ±0.093	0.582 ±0.034	-0.202 ±0.067
Greedy Sampler + DivideMix	0.891 ±0.035	-0.007 ±0.027	0.679 ±0.153	0.026 ±0.059	0.571 ±0.038	0.111 ±0.028
Rainbow Memory + DivideMix	0.87 ±0.046	0.004 ±0.03	0.784 ±0.017	0.037 ±0.069	0.606 ±0.043	0.175 ±0.068
PuriDivER	0.893 ±0.033	0.069 ±0.037	0.714 ±0.058	0.198 ±0.032	0.519 ±0.037	0.352 ±0.053
Continual CRUST	0.937 ±0.019	-0.062 ±0.033	0.889 ±0.022	-0.053 ±0.032	0.728 ±0.031	-0.005 ±0.025

4.7.3 Noisy Disjoint Continual Learning

Up to this point, we've only considered the blurry setting where every experience has a small amount of data from the other targets-of-interest in addition to the major classes. In practice, this assumption may be unrealistic due

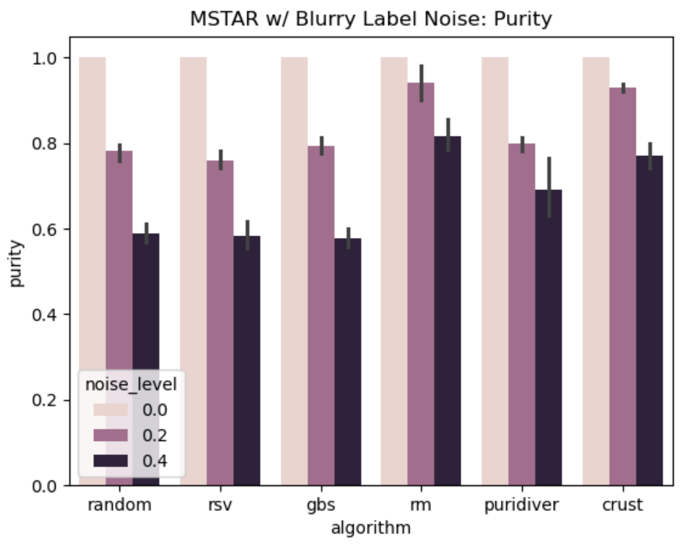


Figure 5. Evaluating how different algorithms are effected in terms of purity as the proportion of label noise increases on the MSTAR dataset. This setting involves a blurry one-class CIL curricula with varying amounts of label noise. The replay memory size is fixed at 200 samples.

to how data collection events are organized or as a result of the approach to labeling. Furthermore, the disjoint CL setting is important to consider because it also helps us understand how the recognition model may perform when data from unseen future classes appear prematurely and are confused with known targets-of-interest. We've observed that this can make the incremental SAR ATR problem more challenging.

In this set of experiments, we want to understand if Continual CRUST, specifically, can handle the disjoint CIL setting. Note that in our companion technical report on noise-tolerant learning with electro-optic imagery,[105] we've seen Continual CRUST is capable of handling this setting, even under the challenging one-class incremental learning scenario.

As in most other experiments, we consider the incremental SAR ATR problem under label noise with five experiences and two classes per experience. However, unlike earlier experiments, class labels are entirely disjoint between learning experiences, (but other classes, including unknown future classes, can still appear mislabeled in earlier experiences). For this problem, we observe mixed results.

On the MSTAR dataset (Table 5, Fig. 6), we see that the disjoint incremental SAR ATR problem is a more challenging problem than the blurry incremental SAR ATR problem. When 40% of data is mislabelled, final experience accuracy on Continual CRUST drops from 74.3% to 50.9%, and the model is over twice as prone to forgetting. However, Continual CRUST still shows a marked improvement over random replay (11.3% higher final experience accuracy) and about a 10-20% improvement in the purity of the replay buffers.

In contrast, on the SynthWakeSAR dataset with similar experimental parameters, we see no major differences in any metrics between random replay and Continual CRUST. Results are not reported for brevity's sake. This suggests that when the dataset involves performing recognition between classes exhibiting very fine-grained differences, the disjoint setting may present a large challenge for Continual CRUST.

4.7.4 Background Patches as Mislabeled Data Samples

In practice, label noise is not purely the result of confusing different salient targets. In many cases, it can also arise from false detections where the background or nontargets-of-interest pollute the data collection. Note that

Table 5. **Disjoint Curriculum - MSTAR.** Evaluating how different algorithms are affected as the proportion of label noise increases on the MSTAR dataset. This setting involves a disjoint CIL curriculum with five experiences with two classes per experience and varying amounts of label noise. The replay memory size is fixed at 200 samples.

	Noise=0.0		**Noise=0.2**		**Noise=0.4**	
Algorithm	Accuracy	Forgetting	Accuracy	Forgetting	Accuracy	Forgetting
Joint Learner	0.992 ±0.005	N/A	0.885 ±0.011	N/A	0.757 ±0.022	N/A
Joint Learner + CRUST	0.962 ±0.02	N/A	0.928 ±0.015	N/A	0.804 ±0.032	N/A
Naive Sequential Learner	0.198 ±0.003	0.999 ±0.001	0.194 ±0.007	0.973 ±0.016	0.188 ±0.005	0.95 ±0.005
Random Replay	0.926 ±0.034	0.04 ±0.01	0.696 ±0.072	0.24 ±0.04	0.396 ±0.055	0.493 ±0.049
Continual CRUST	0.868 ±0.045	0.071 ±0.025	0.818 ±0.04	0.119 ±0.03	0.509 ±0.076	0.362 ±0.063

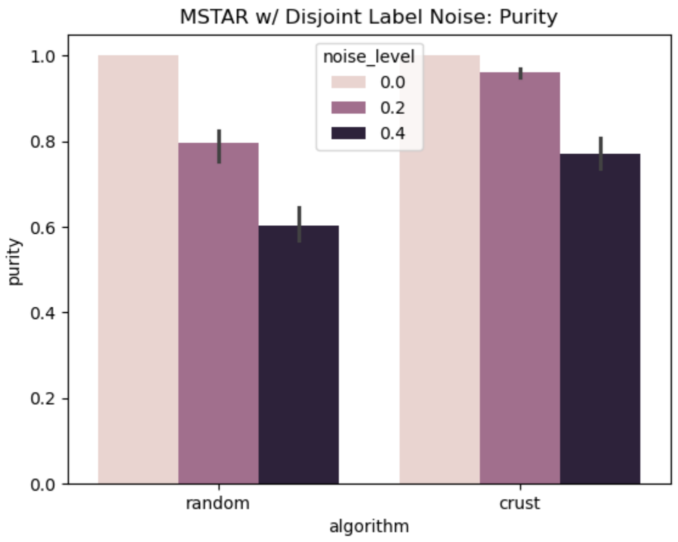


Figure 6. Evaluating how Continual CRUST compares to random replay in terms of purity as the proportion of label noise increases on the MSTAR datasets. This setting involves a disjoint CIL curricula with varying amounts of label noise. The replay memory size is fixed at 200 samples.

this type of noise affects the training of the recognition models in a different way than the label noise that has been previously discussed. Label flipping in earlier experiments provides feedback in direct opposition to learning. Confusing background patches and nontarget objects (assuming appearance is sufficiently different from the targets-of-interest) with targets does not directly oppose learning, instead, it acts as an uninformative feedback signal. Experiments show that if these uninformative samples are mixed with informative samples, the recognition model is likely to be able to learn around the noise with minor loss in performance.

Since MSTAR does not include background patches in the dataset, we generate a dataset of synthetic background patches. Given an image with a target in it, we mask the object using information from the SARBake overlays,[94] replace the masked regions with patches from elsewhere in the same sample, and apply light smoothing to reduce boundary artifacts. An example of an inpainted synthetic background image appears in Fig. 7. We then run experiments using the standard blurry curriculum on MSTAR but we do not employ traditional label noise; instead, we replace 40% of the data with images with the targets removed. Results appear in Table 6. What we see is even with 40% noisy data, the continual learners do not exhibit catastrophic loss w.r.t.

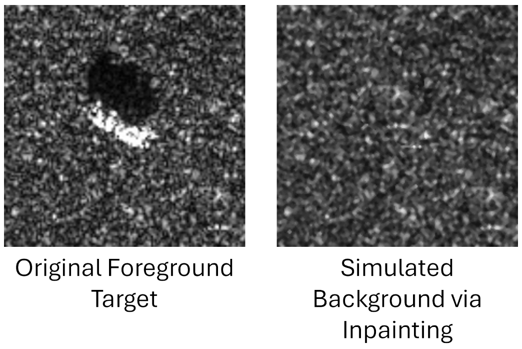
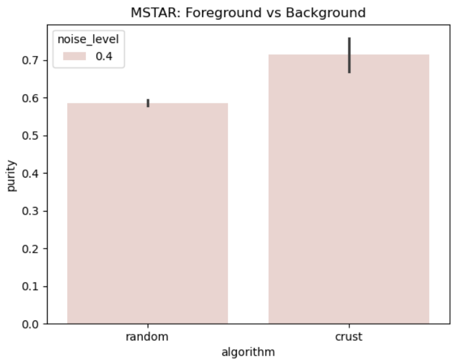

Original Foreground Target
Simulated Background via Inpainting

Figure 7. We generate synthetic background image patches by inpainting over foreground targets in the MSTAR dataset.

final experience accuracy: the naïve sequential learner is within 1%, the random replay model loses 7.88%, and Continual CRUST loses 6.5%. This can be partially explained by having fewer clean samples to train on in addition to fitting to noisy distractor data. The random replay model and Continual CRUST model are also within 1% accuracy of each other, so it would initially appear Continual CRUST is not necessarily robust to this type of noise. However, if we instead look at the difference between the random replay model and Continual CRUST in terms of replay memory purity, we see Continual CRUST produces a replay memory that is about 10% more pure than selecting samples randomly. This suggests that Continual CRUST is still exhibiting some level of robustness even when the distractor data is uninformative instead of antagonistic as is the case in earlier experiments.

Table 6. **Filtering Foreground from Background - MSTAR.** Evaluating how different algorithms are affected as foreground objects are increasingly replaced with background patches in the MSTAR dataset. This setting involves a blurry CIL curriculum with five experiences with two classes per experience. The replay memory size is fixed at 200 samples.

	Noise=0.4	
Algorithm	Accuracy	Forgetting
Joint Learner	0.909 ±0.028	N/A
Joint Learner + CRUST	0.877 ±0.019	N/A
Naive Sequential Learner	0.761 ±0.02	0.229 ±0.078
Random Replay	0.883 ±0.034	-0.07 ±0.054
Continual CRUST	0.879 ±0.033	-0.054 ±0.053



4.8 Research Question: Does Measurement Noise Impact the Incremental SAR ATR Problem?

We've thoroughly examined the incremental SAR ATR problem under the noisy label regime. Next, we consider the incremental SAR ATR problem when measurement noise is present. Whereas earlier, we saw that Continual CRUST is *theoretically-motivated* to be robust to noise in the label and measurement space. We now focus on examining whether Continual CRUST is *empirically* robust to measurement noise.

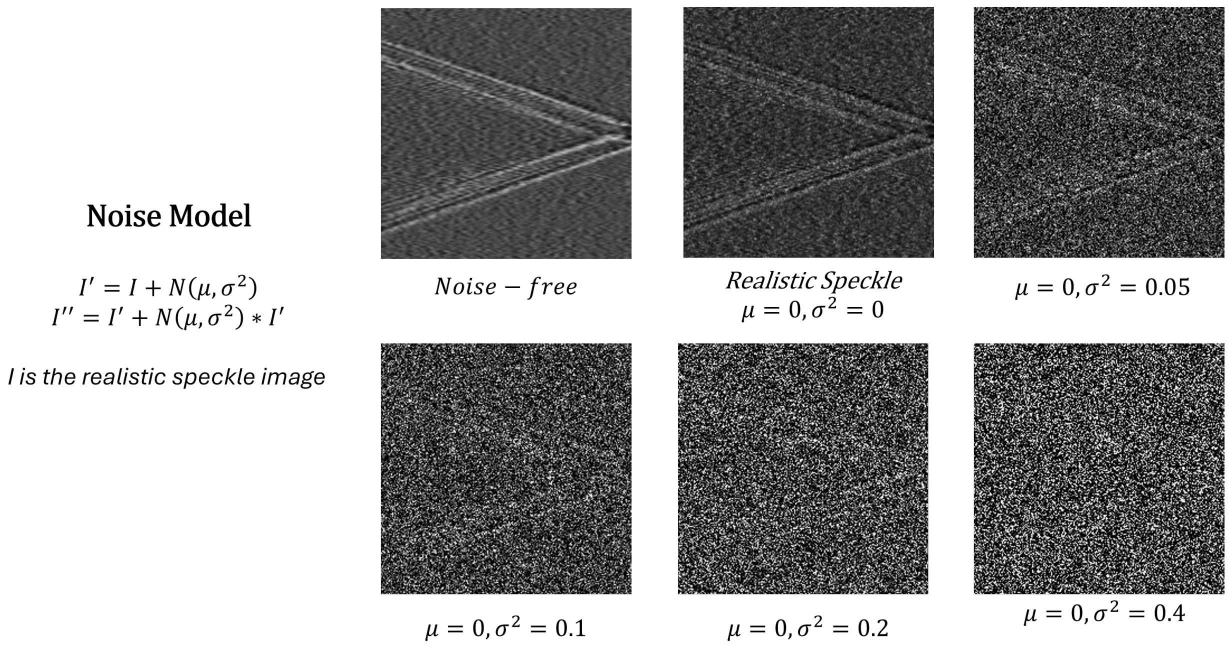

Noise Model
$I' = I + N(\mu, \sigma^2)$
$I'' = I' + N(\mu, \sigma^2) * I'$
I is the realistic speckle image
$Noise - free$
Realistic Speckle
$\mu = 0, \sigma^2 = 0$
$\mu = 0, \sigma^2 = 0.05$
$\mu = 0, \sigma^2 = 0.1$
$\mu = 0, \sigma^2 = 0.2$
$\mu = 0, \sigma^2 = 0.4$

Figure 8. Examples of controlling the amount of noise in SAR imagery. The noise-free and realistic speckle imagery comes directly from the SynthWakeSAR[3] dataset. To understand how measurement noise affects the system as it approaches total corruption, we employ a noise model that consists of perturbing an image with additive Gaussian noise followed by multiplicative noise. Note that these perturbations are performed on top of the realistic speckle images, not the noise-free images.

4.8.1 Noise-Free vs Realistic Speckle Images

For the first set of experiments, we want to see how well incremental SAR ATR systems are able to learn around images perturbed with realistic speckle. For these experiments, we use the SynthWakeSAR dataset and evaluate on the standard curriculum, but we mix noise-free images with images that contain realistic speckle in increasing proportions (0% of data has speckle, 20% of data has speckle, and 40% of data has speckle). Results appear in Table 7. Unlike in earlier experiments with label noise, we see that it is much easier for SAR ATR models to learn around measurement noise in the form of realistic speckle. The joint learner and random replay models exhibit no loss of performance going from a dataset of completely noise-free images to an image dataset where 40% of images have realistic speckle. Continual CRUST loses about 1% accuracy, and even the naïve sequential learner only loses 2.5% performance. This finding isn't particularly surprising as convolutional neural networks are known for learning well under small amounts of uncorrelated random noise. Furthermore, if we look at the purity of the replay memory learned by Continual CRUST versus random selection, we see little difference (Fig. 9). This suggests that the Continual CRUST model doesn't view images with realistic speckle as misleading or uninformative. This is actually desirable behavior as this experimental setup is a toy formulation; in practice, one would expect all imagery to contain some amount of speckle.

4.9 Research Question: Is Continual CRUST (Empirically) Robust to Measurement Noise?

Following on the results of the previous experiment, we ask the question, are there any cases where adding additional measurement noise to the SAR imagery either negatively impacts the learning of the incremental ATR model or where Continual CRUST is capable of learning nontrivial coresets for the replay memory?

Table 7. **Noise-Free vs Realistic Speckle - SynthWakeSAR.** Evaluating how different algorithms are affected as the proportion of samples with realistic speckle increases on the SynthWakeSAR dataset. This setting involves a blurry CIL curriculum with five experiences with two major classes per experience. The replay memory size is fixed at 2000 samples.

	Noise=0.0		**Noise=0.2**		**Noise=0.4**	
Algorithm	Accuracy	Forgetting	Accuracy	Forgetting	Accuracy	Forgetting
Joint Learner	0.97 $\pm$0.002	N/A	0.969 $\pm$0.001	N/A	0.967 $\pm$0.002	N/A
Joint Learner + CRUST	0.956 $\pm$0.001	N/A	0.954 $\pm$0.001	N/A	0.951 $\pm$0.006	N/A
Naive Sequential Learner	0.881 $\pm$0.012	0.127 $\pm$0.017	0.868 $\pm$0.008	0.145 $\pm$0.017	0.856 $\pm$0.008	0.149 $\pm$0.025
Random Replay	0.935 $\pm$0.006	0.02 $\pm$0.005	0.931 $\pm$0.005	0.015 $\pm$0.009	0.93 $\pm$0.0	0.015 $\pm$0.005
Continual CRUST	0.926 $\pm$0.004	0.019 $\pm$0.007	0.921 $\pm$0.005	0.013 $\pm$0.01	0.915 $\pm$0.004	0.018 $\pm$0.004

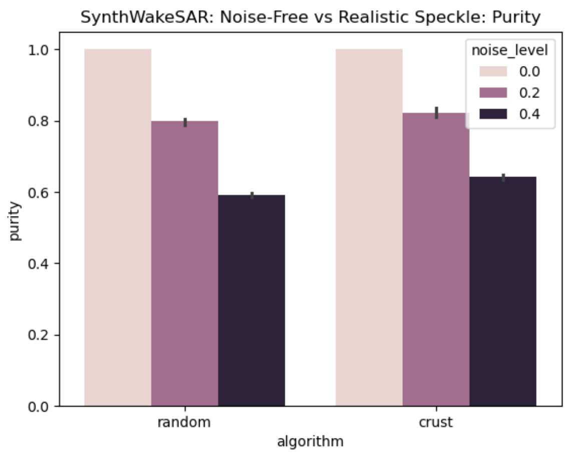


Figure 9. Evaluating how Continual CRUST compares to random replay in terms of purity as the proportion of samples exhibiting realistic speckle increases on the SynthWakeSAR datasets. This setting involves a disjoint CIL curricula with varying amounts of label noise. The replay memory size is fixed at 2000 samples

4.9.1 Realistic Speckle vs Highly-Corrupt Images

To answer the aforementioned question, we generate a series of artificial datasets where 60% of images contain realistic speckle and 40% of images have been perturbed with stronger measurement noise, varying from no additional noise to total corruption of the image. We highlight the progression of noise levels in Fig. 8. This experiment is designed to test the case where a SAR ATR system may be effected by periodic bursts of increased noise in the measurements beyond normal speckle, e.g. in very extreme environmental conditions and when dramatic sensor hardware failures occur.

To generate a corrupted image I'' from a realistic speckle image I, we use the following noise model, where $N(\mu, \sigma^2)$ defines a normal distribution sampled at the same dimensions as the input image:

$$I' = I + N(\mu, \sigma^2), I'' = I' + N(\mu, \sigma^2) * I' \tag{10}$$

This process perturbs an image with additive Gaussian noise followed by multiplicative noise. We always use a mean of zero and we vary the variance from zero to 0.4. Results appear in Fig. 10.

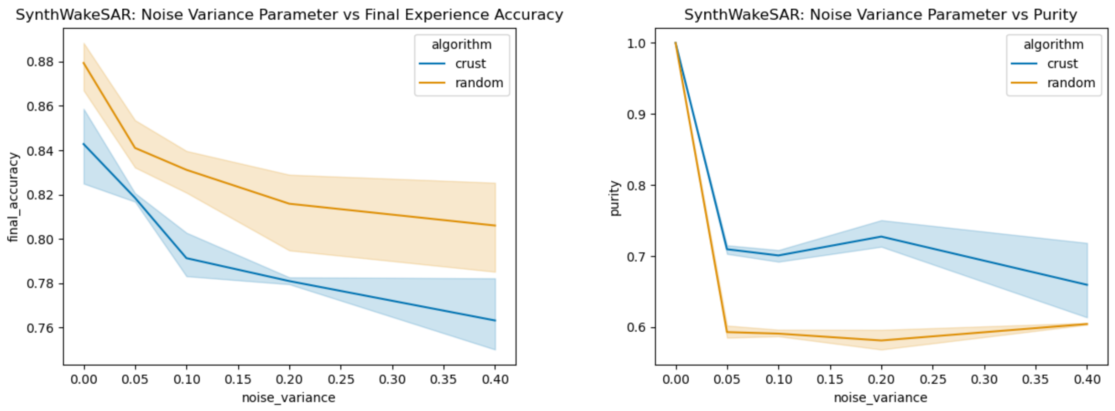


Figure 10. **Measurement Noise Ablation - SynthWakeSAR.** Exploring how affected random replay and Continual CRUST are to increasingly stronger perturbations in the measured images. Experiments are run on SynthWakeSAR on the blurry curriculum with five experiences of two major classes per experience. In all cases 40% of images are perturbed.

We observe that performance decreases as the strength of the perturbations increases for both random replay and Continual CRUST. It is interesting to note that Continual CRUST does construct cleaner replay buffers than random replay, but consistently underperforms random replay by several points in terms of accuracy at every noise level. This suggests that Continual CRUST in this setting, may focus too much on selecting "easy" clean samples and may be filtering out more diverse that benefit or regularize the learning. As such, we conclude that Continual CRUST may be more tolerant to measurement noise when constructing the replay buffer, but this may actually have an adverse effect on learning.

4.10 Research Question: Are Scattering Model Guided Adversarial Attacks Effective for Constructing Backdoor Triggers?

The final type of noise we consider in our explorations is adversarial noise, specifically, backdoor trigger attacks. These attacks and the generating process for these attacks are described in Section 3.5. Before we can use the backdoor triggers to study adversarial robustness, we first must determine if the extension of SMGAA to poisoning attacks is effective. We generate generic backdoor triggers for all classes of MSTAR, poison all samples with their corresponding trigger, and measure the change in average prediction confidence for each class using a proxy recognition model. Results appear in Table 8.

We note that the attack has mixed results. In many cases confidence is only lowered by less than ten percent. However, for D7, T62, and ZIL131, the attack substantially lowers confidence. This attack is designed with the intent of being used to get an initial sense for how robust our proposed Continual CRUST is to poisoning attacks. As such, for preliminary studies, we select one specific target to attack that may be useful of stress-testing our algorithm. Based on our observations, we use T62 as the target of our poisoning attacks in the next set of experiments.

4.11 Research Question: Is Continual CRUST Sensitive to Poisoning Attacks, Specifically Backdoor Trigger Attacks?

We perform blurry continual learning on MSTAR over five experiences with two major tasks per experience and a replay buffer size of 200 samples. We poison the T62 class training data in varying amount (0% of training samples are poisoned up to 100% of training data samples are poisoned). We evaluate replay buffer purity, final experience accuracy, recall w.r.t. the poisoned class, and precision w.r.t. the poisoned class on both a 100% clean test set and a test set where 100% of samples from *all* classes have the trigger inserted. Results are reported in Fig. 11.

Table 8. We generate generic backdoor triggers for all classes of MSTAR, poison all samples with their corresponding trigger, and measure the change in average prediction confidence for each class using a proxy recognition model. Strong attacks are highlighted in bold italics.

Class	**Average Confidence Pre-Poisoning**	**Average Confidence Post-Poisoning**
2S1	0.996	0.955
BMP2	0.985	0.961
BRDM2	0.994	0.891
BTR60	0.989	0.896
BTR70	0.968	0.895
D7	0.997	***0.680***
T62	0.997	***0.501***
T72	0.987	0.934
ZIL131	0.999	***0.383***
ZSU23	0.998	0.928

We hypothesize that the incremental SAR ATR will be affected by the backdoor trigger attacks. Unlike in the label and measurement noise scenarios, we hypothesize that Continual CRUST will be especially sensitive to these types of attacks. Continual CRUST constructs coresets by performing k-medians clustering in the information space of the gradients. Backdoor triggers form strong correlations between specific input patterns and class labels. This is exactly the case where one might expect the k-medians clustering to focus on the trigger, thus, corrupting the coreset used to form the replay buffer.

Looking at the final accuracy metric, we do observe that there is a loss in performance as the poisoning amount increases, and we see the expected behavior of a backdoor attack: there is relatively little loss in performance when tested on clean data samples, but significant drops when tested on a test set with the trigger inserted. This is further highlighted in the precision and recall metrics where regardless of poisoning amount, the clean dataset has precision near one and the poisoned dataset has recall near one. We also observe that the attack isn't very effective until over fifty percent of the data has been poisoned, which is unlikely to occur in practice, suggesting that the SAR ATR models exhibit some natural robustness w.r.t. these poisoning attacks, but this is just an initial impression and requires significantly more exploration and validation. As the poisoning amount approaches one-hundred percent, we start seeing the signs of an effective attack, enabling evasion in two ways:

- The poisoned class precision on the poisoned test set nears 0.4. This means the ATR model is misclassifying data samples from other targets as the poisoned target when the trigger is present. If the poisoned class is considered "unimportant" than this means that important targets (non-poisoned) can successfully masquerade as unimportant targets (the poisoned class), evading detection.
- The poisoned class recall of the clean samples approaches 0.2. This means if the trigger is removed from the poisoned target, the ATR system likely won't know how to handle the target. If the poisoned class is considered "important" in this case, then this enables the important target (poisoned class) to evade recognition by removing the trigger.

The final question that remains is whether Continual CRUST is more susceptible than randomly selecting samples for the replay buffer. Looking at the purity metrics, the answer appears to be that for low amounts ($< 50\%$) of poisoning, Continual CRUST behaves very similarly to the random replay algorithm. However, at 50% poisoning, Continual CRUST is slightly more susceptible (by about 4%) to selecting poisoned samples. This is actually less significant than we expected to see, and thus, merits further investigation across different datasets and different methods for generating the poisoning attack before we can definitively determine if Continual CRUST is truly susceptible to backdoor trigger attacks.

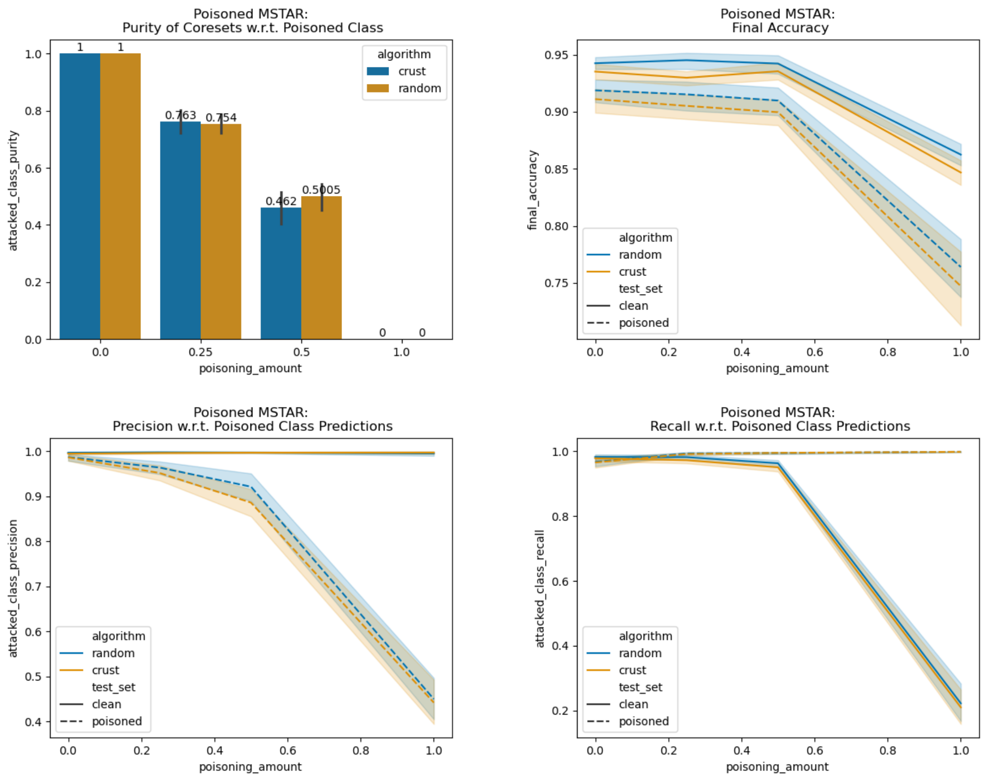


Figure 11. **Poisoning Attacks - MSTAR.** We perform blurry continual learning on MSTAR over five experiences with two major tasks per experience and a replay buffer size of 200 samples. We poison the T62 class in varying amount (0% of training samples are poisoned up to 100% of data samples are poisoned). We evaluate replay buffer purity (top left), final experience accuracy (top right), recall w.r.t. the poisoned class (bottom left), and precision w.r.t. the poisoned class (bottom right) on both a 100% clean test set and a test set where 100% of samples from all classes have the trigger inserted.

4.12 Research Question: Is Continual CRUST Effective for Class-Incremental Learning Problems Beyond SAR-Based Automatic Target Recognition?

This paper focused on understanding Continual CRUST in the incremental SAR ATR setting. However, it should be noted that this method is not limited to the SAR domain. In our companion technical report,[105] we investigated Continual CRUST on challenging class-incremental learning curricula evaluated with a diverse set of electro-optic imagery, ranging from handwritten digits to natural images to pathology images. Those findings largely match those observed in the incremental SAR ATR setting: Continual CRUST is clearly empirically robust to label noise and showed some robustness to measurement noise. For additional details on these studies, we refer the reader to our companion technical report.

5. CONCLUSIONS AND FUTURE WORK

This work focused on class-incremental synthetic aperture radar-based automatic target recognition. In this setting, new target classes are sequentially added to an existing ML-based recognition model, and the model has limited access to data samples from past targets during training. Continual learning introduces additional avenues for corrupting models during data collection. We extensively studied class-incremental SAR ATR under three types of noise that may arise during data collection that may interfere with learning: noisy labels, perturbed data, and adversarial poisoning attacks. Our experimental results suggest that under the right circumstances, each of these types of noise can present problems to a continual learning system, especially noisy labels.

We introduced a novel extension of the CRUST[1] algorithm to the incremental learning setting. The proposed algorith, Continual CRUST, is theoretically-motivated to be robust to label and measurement noise, and empirical studies provided some initial evidence validating this robustness. When confronted with significant label noise, our proposed algorithm showed promising improvement over existing baselines. When confronted with significant measurement noise, our proposed algorithm performed similar to continual learning baselines, but also demonstrated improved filtering of noisy samples within the replay memory.

To test effectiveness against adversarial poisoning attacks, we repurposed the "Scattering Model Guided Adversarial Attack" algorithm (SMGAA),[4] which was originally designed for evasion attacks, for inserting backdoor triggers. Experimental results were mixed: the attacks were successful for some classes in MSTAR, but failed for others, and the continual learning algorithms were relatively unaffected by the poisoning attack until the learned backdoor trigger appeared in at least 50% of the poisoned target class' data. Once, the data was extremely poisoned, we observed behavior consistent with successful poisoning attacks.

Noisy incremental SAR ATR is a new research topic, and there is much left to explore in this problem setting. Our methods were tested on relatively "easy" SAR ATR datasets (at least for static machine learning). Further exploration needs to be conducted on a wider range of datasets with more challenging settings. Furthermore, there are adjacent problems to incremental SAR ATR (e.g. the target *detection* stage, open world recognition) that this paper did not touch on, which requires significant investigation under the noisy incremental learning setting. Finally, we proposed a novel algorithm for noise-tolerant class incremental learning, but this algorithm is not without its flaws. We saw some preliminary evidence that it may be slightly more susceptible to poisoning attacks in the form of backdoor trigger attacks. In future work, we'd like to explore how the model performs on more effective poisoning attacks. We also expect the approach would benefit from being combined with techniques such as adversarial training to improve its adversarial robustness. Furthermore, we saw cases (e.g. with measurement noise) where the Continual CRUST outperformed baseline algorithms w.r.t. replay memory purity but then underperformed w.r.t. final experience accuracy. This suggests that the algorithm may need to be further modified to the improve diversity of the selected samples in the replay buffer while maintaining or improving the purity of the learned coresets. Overall, this paper highlights significant new challenges in noisy incremental SAR ATR and provides promising future directions to advance the field.

REFERENCES

[1] Mirzasoleiman, B., Cao, K., and Leskovec, J., "Coresets for robust training of deep neural networks against noisy labels," *NeurIPS* (2020).

[2] Keydel, E. R., Lee, S. W., and Moore, J. T., "Mstar extended operating conditions: A tutorial," *Algorithms for Synthetic Aperture Radar Imagery III* (1996).

[3] Rizaev, I. G. and Achim, A., "Synthwakesar: A synthetic sar dataset for deep learning classification of ships at sea," *Remote Sensing* (2022).

[4] Peng, B., Peng, B., Zhou, J., Xie, J., and Liu, L., "Scattering model guided adversarial examples for sar target recognition: Attack and defense," *IEEE Transactions on Geoscience and Remote Sensing* (2022).

[5] Thrun, S., "Lifelong learning algorithms," in [*Learning to learn*], Springer (1998).

[6] Van de Ven, G. M. and Tolias, A. S., "Three scenarios for continual learning," *Machine Intelligence* (2022).

[7] Van Engelen, J. E. and Hoos, H. H., "A survey on semi-supervised learning," *Machine learning* (2020).

[8] Franceschetti, G. and Lanari, R., [*Synthetic aperture radar processing*], CRC press (2018).

[9] Peng, B., Peng, B., Zhou, J., Xia, J., and Liu, L., "Speckle-variant attack: Toward transferable adversarial attack to sar target recognition," *IEEE Geoscience and Remote Sensing Letters* (2022).

[10] Saha, A., Subramanya, A., and Pirsiavash, H., "Hidden trigger backdoor attacks," in [*AAAI*], (2020).

[11] De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T., "A continual learning survey: Defying forgetting in classification tasks," *IEEE TPAMI* (2021).

[12] van de Ven, G. M., Tuytelaars, T., and Tolias, A. S., "Three types of incremental learning," *Nature Machine Intelligence* (2022).

[13] Chen, Z. and Liu, B., "Lifelong machine learning," *Synthesis Lectures on AI and ML* (2018).

[14] Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., and Wayne, G., "Experience replay for continual learning," *Advances in neural information processing systems* **32** (2019).

[15] Isele, D. and Cosgun, A., "Selective experience replay for lifelong learning," in [*Proceedings of the AAAI Conference on Artificial Intelligence*], **32**(1) (2018).

[16] Lopez-Paz, D. and Ranzato, M., "Gradient episodic memory for continual learning," *Advances in neural information processing systems* **30** (2017).

[17] Buzzega, P., Boschini, M., Porrello, A., Abati, D., and Calderara, S., "Dark experience for general continual learning: a strong, simple baseline," *Advances in neural information processing systems* **33**, 15920–15930 (2020).

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

[19] Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y., "Gradient based sample selection for online continual learning," *Advances in neural information processing systems* **32** (2019).

[20] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al., "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences* **114**(13), 3521–3526 (2017).

[21] Zenke, F., Poole, B., and Ganguli, S., "Continual learning through synaptic intelligence," in [*International conference on machine learning*], 3987–3995, PMLR (2017).

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

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

[24] Fayek, H. M., Cavedon, L., and Wu, H. R., "Progressive learning: A deep learning framework for continual learning," *Neural Networks* **128** (2020).

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

[26] Hung, C.-Y., Tu, C.-H., Wu, C.-E., Chen, C.-H., Chan, Y.-M., and Chen, C.-S., "Compacting, picking and growing for unforgetting continual learning," *NeurIPS* (2019).

[27] Li, X., Zhou, Y., Wu, T., Socher, R., and Xiong, C., "Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting," in [*ICML*], PMLR (2019).

[28] Schaul, T., "Prioritized experience replay," *arXiv preprint arXiv:1511.05952* (2015).

[29] Wang, R., Frans, K., Abbeel, P., Levine, S., and Efros, A. A., "Prioritized generative replay," *arXiv preprint arXiv:2410.18082* (2024).

[30] Borsos, Z., Mutny, M., and Krause, A., "Coresets via bilevel optimization for continual learning and streaming," *Advances in neural information processing systems* **33**, 14879–14890 (2020).

[31] Yoon, J., Madaan, D., Yang, E., and Hwang, S. J., "Online coreset selection for rehearsal-based continual learning," in [*International Conference on Learning Representations*], (2021).

[32] Tiwari, R., Killamsetty, K., Iyer, R., and Shenoy, P., "Gcr: Gradient coreset based replay buffer selection for continual learning," in [*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*], 99–108 (2022).

[33] Pellegrini, L., Graffieti, G., Lomonaco, V., and Maltoni, D., "Latent replay for real-time continual learning," in [*2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*], 10203–10209, IEEE (2020).

[34] Hayes, T. L., Kafle, K., Shrestha, R., Acharya, M., and Kanan, C., "Remind your neural network to prevent catastrophic forgetting," in [*European conference on computer vision*], 466–483, Springer (2020).

[35] Shin, H., Lee, J. K., Kim, J., and Kim, J., "Continual learning with deep generative replay," *Advances in neural information processing systems* **30** (2017).

[36] Li, H. and Ditzler, G., "Targeted data poisoning attacks against continual learning neural networks," in [*2022 International Joint Conference on Neural Networks (IJCNN)*], 1–8, IEEE (2022).

[37] Li, H. and Ditzler, G., "Pacol: Poisoning attacks against continual learners," *arXiv preprint arXiv:2311.10919* (2023).

[38] Abbasi, A., Nooralinejad, P., Pirsiavash, H., and Kolouri, S., "Brainwash: A poisoning attack to forget in continual learning," in [*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*], 24057–24067 (2024).

[39] Guo, Z., Kumar, A., and Tourani, R., "Persistent backdoor attacks in continual learning," *arXiv preprint arXiv:2409.13864* (2024).

[40] Kang, S., Shi, Z., and Zhang, X., "Poisoning generative replay in continual learning to promote forgetting," in [*International Conference on Machine Learning*], 15769–15785, PMLR (2023).

[41] Umer, M. and Polikar, R., "Adversarial targeted forgetting in regularization and generative based continual learning models," in [*2021 International Joint Conference on Neural Networks (IJCNN)*], 1–8, IEEE (2021).

[42] Kang, S., Shi, Z., and Zhang, X., "Continual poisoning of generative models to promote catastrophic forgetting," in [*NeurIPS ML Safety Workshop*], (2022).

[43] Khan, H., Bouaynaya, N. C., and Rasool, G., "Adversarially robust continual learning," in [*2022 International Joint Conference on Neural Networks (IJCNN)*], 1–8, IEEE (2022).

[44] Bai, T., Chen, C., Lyu, L., Zhao, J., and Wen, B., "Towards adversarially robust continual learning," in [*ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*], 1–5, IEEE (2023).

[45] Wang, Z., Shen, L., Zhan, D., Suo, Q., Zhu, Y., Duan, T., and Gao, M., "Metamix: Towards corruption-robust continual learning with temporally self-adaptive data transformation," in [*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*], 24521–24531 (2023).

[46] Umer, M. and Polikar, R., "Adversary aware continual learning," *IEEE Access* (2024).

[47] Jia, J., Zhang, Y., Song, D., Liu, S., and Hero, A., "Robustness-preserving lifelong learning via dataset condensation," in [*ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*], 1–5, IEEE (2023).

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

[49] Millunzi, M., Bonicelli, L., Porrello, A., Credi, J., Kolm, P. N., Calderara, S., et al., "May the forgetting be with you: Alternate replay for learning with noisy labels," in [*BMVC*], (2024).

[50] Bang, J., Koh, H., Park, S., Song, H., Ha, J.-W., and Choi, J., "Online continual learning on a contaminated data stream with blurry task boundaries," in [*CVPR*], (2022).

[51] Kim, C. D., Jeong, J., Moon, S., and Kim, G., "Continual learning on noisy data streams via self-purified replay," in [*ICCV*], (2021).

[52] Karim, N., Khalid, U., Esmaeili, A., and Rahnavard, N., "Cnll: A semi-supervised approach for continual noisy label learning," in [*CVPR*], (2022).

[53] Ross, T. D., Bradley, J. J., Hudson, L. J., and O'connor, M. P., "Sar atr: so what's the problem? an mstar perspective," in [*Algorithms for Synthetic Aperture Radar Imagery VI*], SPIE (1999).

[54] Yi-Bo, L., Chang, Z., and Ning, W., "A survey on feature extraction of sar images," in [*ICCASM*], IEEE (2010).

[55] Li, J., Yu, Z., Yu, L., Cheng, P., Chen, J., and Chi, C., "A comprehensive survey on sar atr in deep-learning era," *Remote Sensing* (2023).

[56] Tang, Z., Sun, Y., Liu, C., Xu, Y., and Lei, L., "Simulated data-guided incremental sar atr through feature aggregation," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2024).

[57] Wang, L., Yang, X., Tan, H., Bai, X., and Zhou, F., "Few-shot class-incremental sar target recognition based on hierarchical embedding and incremental evolutionary network," *IEEE Transactions on Geoscience and Remote Sensing* (2023).

[58] Zhao, Y., Zhao, L., Zhang, S., Ji, K., Kuang, G., and Liu, L., "Azimuth-aware subspace classifier for few-shot class-incremental sar atr," *IEEE Transactions on Geoscience and Remote Sensing* (2024).

[59] Xu, Y., Sun, H., Zhao, Y., He, Q., Zhang, S., Kuang, G., and Chen, H., "Simulated data feature guided evolution and distillation for incremental sar atr," *IEEE Transactions on Geoscience and Remote Sensing* (2024).

[60] Li, B., Cui, Z., Cao, Z., and Yang, J., "Incremental learning based on anchored class centers for sar automatic target recognition," *IEEE Transactions on Geoscience and Remote Sensing* (2022).

[61] Tang, J., Xiang, D., Zhang, F., Ma, F., Zhou, Y., and Li, H., "Incremental sar automatic target recognition with error correction and high plasticity," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2022).

[62] Zhao, Y., Zhao, L., Ding, D., Hu, D., Kuang, G., and Liu, L., "Few-shot class-incremental sar target recognition via cosine prototype learning," *IEEE Transactions on Geoscience and Remote Sensing* (2023).

[63] Huang, H., Gao, F., Wang, J., Hussain, A., and Zhou, H., "An incremental sar target recognition framework via memory-augmented weight alignment and enhancement discrimination," *IEEE Geoscience and Remote Sensing Letters* (2023).

[64] Li, B., Cui, Z., Wang, H., Deng, Y., Ma, J., Yang, J., and Cao, Z., "Sar incremental automatic target recognition based on mutual information maximization," *IEEE Geoscience and Remote Sensing Letters* (2024).

[65] Ren, H., Dong, F., Zhou, R., Yu, X., Zou, L., and Zhou, Y., "Dynamic embedding relation distillation network for incremental sar automatic target recognition," *IEEE Geoscience and Remote Sensing Letters* (2024).

[66] Ma, X., Ji, K., Feng, S., Zhang, L., Xiong, B., and Kuang, G., "Open set recognition with incremental learning for sar target classification," *IEEE Transactions on Geoscience and Remote Sensing* (2023).

[67] Oveis, A. H., Giusti, E., Ghio, S., Meucci, G., and Martorella, M., "Lime-assisted automatic target recognition with sar images: Toward incremental learning and explainability," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2023).

[68] Ren, H., Zhou, R., Zou, L., and Tang, H., "Hierarchical distribution-based exemplar replay for incremental sar automatic target recognition," *IEEE Transactions on Aerospace and Electronic Systems* (2025).

[69] Zhou, R., Ren, H., Li, Y., Dong, F., Zhou, Y., and Zou, L., "Layer-wise representative exemplar selection-based incremental learning for sar target recognition," in [*IGARSS*], IEEE (2024).

[70] Karantaidis, G., Pantsios, A., Kompatsiaris, I., and Papadopoulos, S., "Incsar: A dual fusion incremental learning framework for sar target recognition," *IEEE Access* (2025).

[71] Li, B., Cui, Z., Sun, Y., Yang, J., and Cao, Z., "Density coverage-based exemplar selection for incremental sar automatic target recognition," *IEEE Transactions on Geoscience and Remote Sensing* (2023).

[72] Yu, X., Dong, F., Ren, H., Zhang, C., Zou, L., and Zhou, Y., "Multilevel adaptive knowledge distillation network for incremental sar target recognition," *IEEE Geoscience and Remote Sensing Letters* (2023).

[73] Xu, Y., Sun, H., Zhang, S., Ji, K., and Kuang, G., "Class-incremental sar automatic target recognition using simulated data," in [*IET International Radar Conference (IRC 2023)*], IET (2023).

[74] Zhao, Y., Zhao, L., Zhang, S., Ji, K., and Kuang, G., "Few-shot class-incremental sar target recognition via decoupled scattering augmentation classifier," in [*IGARSS 2024-2024 IEEE International Geoscience and Remote Sensing Symposium*], 7584–7587, IEEE (2024).

[75] Zhao, Y., Zhao, L., Zhang, S., Liu, L., Ji, K., and Kuang, G., "Decoupled self-supervised subspace classifier for few-shot class-incremental sar target recognition," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2024).

[76] Zhu, J., Du, C., and Guo, D., "Target-aspect domain continual sar-atr based on task hard attention mechanism," *IEEE Geoscience and Remote Sensing Letters* (2024).

[77] Sun, H., Xu, Y., Fu, K., Lei, L., Ji, K., and Kuang, G., "An evaluation of representative samples replay and knowledge distillation regularization for sar atr continual learning," in [*PIERS*], IEEE (2024).

[78] Li, B., Cui, Z., Deng, Y., Zhou, Z., and Cao, Z., "Random interpolation data augmentation for incremental automatic target recognition," in [*IGARSS*], IEEE (2024).

[79] Oveis, A. H., Giusti, E., Ghio, S., Meucci, G., and Martorella, M., "Incremental learning in synthetic aperture radar images using openmax algorithm," in [*RadarConf*], IEEE (2023).

[80] Chen, H., Du, C., Zhu, J., and Guo, D., "Target-aspect domain continual learning for sar target recognition," *IEEE Transactions on Geoscience and Remote Sensing* (2025).

[81] Zhou, Y., Zhang, S., Sun, X., Ma, F., and Zhang, F., "Sar target incremental recognition based on hybrid loss function and class-bias correction," *Applied Sciences* (2022).

[82] Kong, L., He, X., Wang, J., Sun, J., Zhou, H., Hussain, A., et al., "Few-shot class-incremental sar target recognition via orthogonal distributed features," *IEEE Transactions on Aerospace and Electronic Systems* (2024).

[83] Li, P., Hu, X., Feng, C., and Feng, W., "Few-shot class-incremental sar target classification based on class-aware bilateral distillation," *IET Conference Proceedings* (2024).

[84] Munteanu, A. and Schwiegelshohn, C., "Coresets-methods and history: A theoreticians design pattern for approximation and streaming algorithms," *KI-Künstliche Intelligenz* **32**, 37–53 (2018).

[85] Oymak, S. and Soltanolkotabi, M., "Toward moderate overparameterization: Global convergence guarantees for training shallow neural networks," *IEEE Journal on Selected Areas in Information Theory* **1**(1), 84–105 (2020).

[86] Li, M., Soltanolkotabi, M., and Oymak, S., "Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks," in [*International conference on artificial intelligence and statistics*], 4313–4324, PMLR (2020).

[87] Akhtar, N. and Mian, A., "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access* (2018).

[88] Tian, Z., Cui, L., Liang, J., and Yu, S., "A comprehensive survey on poisoning attacks and countermeasures in machine learning," *ACM Computing Surveys* (2022).

[89] Gerry, M., Potter, L., Gupta, I., and Van Der Merwe, A., "A parametric model for synthetic aperture radar measurements," *IEEE Transactions on Antennas and Propagation* (1999).

[90] Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M., "Optuna: A next-generation hyperparameter optimization framework," in [*KDD*], ACM (2019).

[91] Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B., "Algorithms for hyper-parameter optimization," in [*NeurIPS*], Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K., eds., Curran Associates, Inc. (2011).

[92] Tan, M. and Le, Q., "Efficientnetv2: Smaller models and faster training," in [*ICML*], PMLR (2021).

[93] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR* (2017).

[94] Malmgren-Hansen, D. and Nobel-Jørgensen, M., "Convolutional neural networks for sar image segmentation," *IEEE International Symposium on Signal Processing and Information Technology* (2015).

[95] Lomonaco, V., Pellegrini, L., Cossu, A., Carta, A., Graffieti, G., Hayes, T. L., De Lange, M., Masana, M., Pomponi, J., Van de Ven, G. M., et al., "Avalanche: an end-to-end library for continual learning," in [*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*], 3600–3610 (2021).

[96] Carta, A., Pellegrini, L., Cossu, A., Hemati, H., and Lomonaco, V., "Avalanche: A pytorch library for deep continual learning," *Journal of Machine Learning Research* **24**(363), 1–6 (2023).

[97] Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., and Tesauro, G., "Learning to learn without forgetting by maximizing transfer and minimizing interference," in [*ICLR*], (2018).

[98] Li, J., Socher, R., and Hoi, S. C., "Dividemix: Learning with noisy labels as semi-supervised learning," in [*ICLR*], (2020).

[99] Prabhu, A., Torr, P. H., and Dokania, P. K., "Gdumb: A simple approach that questions our progress in continual learning," in [*ECCV*], Springer (2020).

[100] Bang, J., Kim, H., Yoo, Y., Ha, J.-W., and Choi, J., "Rainbow memory: Continual learning with a memory of diverse samples," in [*CVPR*], (2021).

[101] Díaz-Rodríguez, N., Lomonaco, V., Filliat, D., and Maltoni, D., "Don't forget, there is more than forgetting: new metrics for continual learning," *arXiv preprint arXiv:1810.13166* (2018).

[102] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L., "Imagenet: A large-scale hierarchical image database," in [*CVPR*], IEEE (2009).

[103] Kingma, D. P. and Ba, J., "Adam: A method for stochastic optimization," *arXiv:1412.6980* (2014).

[104] Asadi, N., Mudur, S., and Belilovsky, E., "Tackling online one-class incremental learning by removing negative contrasts," *arXiv preprint arXiv:2203.13307* (2022).

[105] Mucllari, E., Raghavan, A., and Daniels, Z., "Noise-tolerant coreset-based class incremental continual learning," *Technical Report* (2025).

APPENDIX A. PROOF OF THEOREM 3.4

As it is mentioned in,[1] the Jacobian spectrum can be split into information space $\mathcal{I}$ and nuisance space $\mathcal{N}$, associated with the large and small singular values. We first need to define subspaces S_+ and S_-.

First we need:

Definition A.1. *In our following analysis, the Jacobian matrix is going to be L-smooth, which means that for any W_1 and W_2 learnable parameter matrices and $L > 0$ we have that*

$$\|\mathcal{J}(W_1) - \mathcal{J}(W_2)\|_2 \leq L\|W_1 - W_2\|_2$$

Definition A.2. *Let $f : \mathbb{R}^p \longrightarrow \mathbb{R}^n$ be a nonlinear mapping (i.e. neural network). Then if we have $S_+, S_- \subseteq \mathbb{R}^p$ and for all unit vectors $w_+ \in S_+$ and $w_- \in S_-$ and scalars $0 \leq \mu \ll \alpha \leq \beta$ we have:*

$$\alpha \leq \|\mathcal{J}^T(W, X)w_+\|_2 \leq \beta \quad and \quad \|\mathcal{J}^T(W, X)w_-\|_2 \leq \mu$$

Where, W are the learnable parameters and X is the data.

To have a better idea from Definition A.2, S_+ is the subset consisting of the large singular values and S_- is the subset consisting of the smaller singular values. It will be better for us for the future proofs to have bounds for the Jacobian $\mathcal{J}$ and so we get the following Lemma:

Lemma A.3. *Let W be the learnable parameters that satisfy $y = f(W, x)$ and let $\mathcal{J}$ be the Jacobian matrix. Define the matrices, $\mathcal{J}_+ = \Pi_{S_+}(\mathcal{J})$ and $\mathcal{J}_- = \Pi_{S_-}(\mathcal{J})$, where $\pi_{S_+}(\mathcal{J})$ and $\pi_{S_-}(\mathcal{J})$ are the projections of $\mathcal{J}$ onto S_+ and S_-, respectively. Then from Definition A.2 we have:*

$$\alpha \leq \|\mathcal{J}_+\|_2 \leq \beta \quad , \|\mathcal{J}_-\|_2 \leq \mu$$

Proof. By just using the definition of the norm of a matrix we obtain,

$$\alpha \leq \|\mathcal{J}_+\|_2 = \sup_{w_+ \in S_+, \|w_+\|_2 = 1} \|\mathcal{J}^T w_+\|_2 \leq \beta \text{ and } \|\mathcal{J}_-\|_2 = \sup_{w_- \in S_-, \|w_-\|_2 = 1} \|\mathcal{J}^T w_-\|_2 \leq \mu$$

□

Prior to proceeding with the proofs, we must define our learning algorithm (in this case, a simple Gradient Descent) and the applied loss function.

$$\hat{W} = W - \eta \nabla \mathcal{L}(W) \tag{11}$$

$$\mathcal{L}(W) = \frac{1}{2} \sum_{i \in V} (y_i - f(W, x_i))^2 \ , \ \text{V} = \{1, 2, .., n\} \tag{12}$$

Combining the two previous equations, we obtain the following expression for iteration $t+1$:

$$W_{t+1} = W_t - \eta \nabla \mathcal{L}(W_t) = W_t - \eta \mathcal{J}(W_t)^T (f(W_t, x) - y) \tag{13}$$

where η is the learning rate, $\nabla \mathcal{L}(W)$ is the gradient at W, $\mathcal{L}$ is the MSE and $\mathcal{J}$ is the Jacobian matrix.

Furthermore, we define the average Jacobian definition as it will be used throughout our analysis.

Definition A.4. *(Average Jacobian)*[86] *We define the Average Jacobian along the path connecting two points $u, v \in \mathbb{R}^p$ as*

$$\bar{\mathcal{J}}(v, u) = \int_0^1 \mathcal{J}(u + \alpha(v - u)) d\alpha$$

Lemma A.5. [86] *Given the Gradient Descent in Equation 11 we define*

$$C(W) = \bar{\mathcal{J}}(\hat{W}, W) \mathcal{J}(W)^T$$

By using the residual formulas $\hat{r} = f(\hat{W}) - y$ and $r = f(W) - y$ we obtain the following equation

$$\hat{r} = (I - \eta C(W)) r$$

Proof. By using the residual formulas we get

$$\hat{r} = f(\hat{W}) - y \tag{14}$$
$$\overset{\text{(a)}}{=} f(\hat{W}) + r - f(W) \tag{15}$$
$$\overset{\text{(b)}}{=} r + \bar{\mathcal{J}}(\hat{W}, W)(\hat{W} - W) \tag{16}$$
$$\overset{\text{(c)}}{=} r - \eta \bar{\mathcal{J}}(\hat{W}, W) \mathcal{J}(W)^T r \tag{17}$$
$$\overset{\text{(d)}}{=} (I - \eta C(W)) r \tag{18}$$

Now, let us prove each of the previously stated equalities. For (a), we can establish its validity using the residual formulas, leading to the expression $-y = r - f(W)$. (b) This part requires more work. First we need to prove that $f(\hat{W}) - f(W) = \bar{\mathcal{J}}(\hat{W}, W)(\hat{W} - W)$. Consider Definition A.4. Let us introduce the following variable substitutions: $\hat{W} = v$ and $W = u$. We perform a change of variables $t = u + \alpha(v-u)$, which yields $d\alpha = \frac{dt}{v-u}$ with bounds $t \in [u, v]$ and thus we get:

$$\bar{\mathcal{J}}(v,u) = \int_u^v \mathcal{J}(t)\frac{dt}{v-u} = \frac{1}{v-u}\int_u^v \mathcal{J}(t)dt = \frac{1}{v-u}f(t)|_u^v = \frac{f(v)-f(u)}{v-u} \tag{19}$$

The last two equations at the previous equalities are true due to the fact that $\mathcal{J}$ is the Jacobian. This proves (b) since by changing v and u back to $\hat{W}$ and W we get $f(\hat{W}) - f(W) = \bar{\mathcal{J}}(\hat{W}, W)(\hat{W} - W)$. (c) is true from Equation 11 and we get $\hat{W} - W = -\eta\nabla\mathcal{L}(W)$. (d) holds true because we can factor r, and we have defined $C(W) = \bar{\mathcal{J}}(\hat{W}, W)\mathcal{J}(W)^T$.

□

Lemma A.6. *(Perturbed dataset)*

If we follow the previous Lemma A.5 but applying perturbed data $X + E_X$ instead of X then

$$\hat{r} \approx (I - \eta(C(W) + E_{J_2}\mathcal{J}(W)^T + E_{J_1}\bar{\mathcal{J}}(\hat{W}, W))r$$

where $E_{J_2} := \bar{\mathcal{J}}(\hat{W}, W, E_X)$ and $E_{J_1} := \mathcal{J}(W, E_X)^T$.

Proof. We continue in a similar way as in Lemma A.5 but the Jacobian matrices change as follows $\bar{\mathcal{J}}(\hat{W}, W, X + E_X) = \bar{\mathcal{J}}(\hat{W}, W, X) + \bar{\mathcal{J}}(\hat{W}, W, E_X)$. And we write, $\bar{\mathcal{J}}(\hat{W}, W, E_X) = E_{J_2}$. In a similar way, $\mathcal{J}(W, X + E_X)^T = \mathcal{J}(W, X)^T + \mathcal{J}(W, E_X)^T$ and we write, $\mathcal{J}(W, E_X)^T = E_{J_1}$. Following the equalities in Lemma A.5 with the corresponding changes to the perturbed images $X + E_X$ we obtain the equality:

$$\hat{r} \approx (I - \eta(C(W) + E_{J_2}\mathcal{J}(W)^T + \bar{\mathcal{J}}(\hat{W}, W)E_{J_1}))r$$

(Numerical analysis suggests we can ignore the second order error term $E_{J_2}E_{J_1}$). □

We require the following Lemma:

Lemma A.7.

If we have that the Jacobian is L-smooth as in Definition A.1, then we get:

$$\|\bar{\mathcal{J}}(W_2, W_1) - \mathcal{J}(W_1)\|_2 \lessapprox \frac{\alpha(\beta + E_{J_1})}{2\beta} + \|E_{J_2} - E_{J_1}\|_2$$

where α and β are given as in Definition A.2, Jacobian and Average Jacobian are applied to $X + E_X$ and E_{J_1} and E_{J_2} are the errors from perturbed data in Jacobian and Average Jacobian.

Proof. We will first give all the equalities and inequalities and then explain in details.

$$\begin{aligned}
\|\bar{\mathcal{J}}(W_2, W_1) - \mathcal{J}(W_1)\|_2 &\overset{(a)}{\approx} \|\int_0^1 \mathcal{J}(W_1 + t(W_2 - W_1))dt + E_{J_2} - \mathcal{J}(W_1) - E_{J_1}\|_2 && (20)\\
&\overset{(b)}{=} \|\int_0^1 (\mathcal{J}(W_1 + t(W_2 - W_1)) + E_{J_2} - \mathcal{J}(W_1) - E_{J_1})dt\|_2 && (21)\\
&\overset{(b)}{\leq} \int_0^1 \|(\mathcal{J}(W_1 + t(W_2 - W_1)) - \mathcal{J}(W_1))\|_2 dt + \|E_{J_2} - E_{J_1}\|_2 && (22)\\
&\overset{(d)}{\leq} \int_0^1 L\|W_1 + t(W_2 - W_1) - W_1\|_2 dt + \|E_{J_2} - E_{J_1}\|_2 && (23)\\
&= \int_0^1 Lt\|W_2 - W_1\|_2 dt + \|E_{J_2} - E_{J_1}\|_2 && (24)\\
&= L\|W_2 - W_1\|_2 \frac{t^2}{2}|_0^1 + \|E_{J_2} - E_{J_1}\|_2 && (25)\\
&= \frac{L}{2}\|W_2 - W_1\|_2 + \|E_{J_2} - E_{J_1}\|_2 && (26)
\end{aligned}$$

In part (a), we use the Average Jacobian formula from Definition A.4. At part (b) we use the fact that $\int_0^1 (E_{J_2} - \mathcal{J}(W_1) - E_{J_1})\,dt = (E_{J_2} - \mathcal{J}(W_1) - E_{J_1}) \int_0^1 dt = (E_{J_2} - \mathcal{J}(W_1) - E_{J_1})$, where E_{J_2} is the error from the perturbed data using the Average Jacobian and E_{J_1} is the corresponding error for the Jacobian. At (c), we use the properties of norms in the integral and note that E_{J_2} and E_{J_1} do not depend on t, which gives us $\int_0^1 \|E_{J_2} - E_{J_1}\|_2 dt = \|E_{J_2} - E_{J_1}\|_2 t\ |_0^1 = \|E_{J_2} - E_{J_1}\|_2$.since they are constants. At (d) we use the fact that the Jacobian is L-smooth. Then, we simplify the formulas and find the integral.

For $\eta \leq \frac{\alpha}{L\beta\|r_0\|_2}$, where r_0 is the initial residual and α, β are defined as in Definition A.2, and L is the Lipschitz constant, we also have

$$W_{\mathcal{T}+1} = W_{\mathcal{T}} - \eta\nabla\mathcal{L}(W_{\mathcal{T}}) = W_{\mathcal{T}} - \eta\mathcal{J}(W_{\mathcal{T}})^T f(W_{\mathcal{T}}, x).$$

From our previous work in this lemma, we obtain

$$\|\bar{\mathcal{J}}(W_{\mathcal{T}+1}, W_{\mathcal{T}}) - \mathcal{J}(W_{\mathcal{T}})\|_2 \leq \frac{L}{2}\|W_{\mathcal{T}+1} - W_{\mathcal{T}}\|_2 + \|E_{J_2} - E_{J_1}\|_2.$$

Furthermore we get the equations,

$$\begin{aligned}
\|\bar{\mathcal{J}}(W_{\mathcal{T}+1}, W_{\mathcal{T}}) - \mathcal{J}(W_{\mathcal{T}})\|_2 &\overset{(a)}{\lessapprox} \frac{\eta L}{2}\|(\mathcal{J}(W_{\mathcal{T}})^T + E_{J_1})(f(W_{\mathcal{T}}) - y)\|_2 + \|E_{J_2} - E_{J_1}\|_2 && (27)\\
&\overset{(b)}{\leq} \frac{\eta(\beta + E_{J_1})L}{2}\|r_{\mathcal{T}}\|_2 + \|E_{J_2} - E_{J_1}\|_2 && (28)\\
&\leq \frac{\eta(\beta + E_{J_1})L}{2}\|r_0\|_2 \leq \frac{\alpha(\beta + E_{J_1})}{2\beta} + \|E_{J_2} - E_{J_1}\|_2 && (29)
\end{aligned}$$

At (a), we utilized the equation $W_{\mathcal{T}+1} = W_{\mathcal{T}} - \eta\mathcal{J}(W_{\mathcal{T}})^T f(W_{\mathcal{T}}, x)$, which leads to the expression $-\eta\mathcal{J}(W_{\mathcal{T}})^T f(W_{\mathcal{T}}, x) = W_{\mathcal{T}+1} - W_{\mathcal{T}}$. Then, in (b), we applied Lemma A.3, which gives us $\|\mathcal{J}(W_{\mathcal{T}})^T\|_2 \leq \beta$. Finally, we used the fact that $\eta \leq \frac{\alpha}{L\beta\|r_0\|_2}$ in the last inequality.

□

To apply it in other proofs, we also need the following lemma:

LEMMA A.8. [86]

Let matrices $A, C \subset \mathbb{R}^{n\times p}$ where $\|A-C\|_2 \leq \frac{\alpha}{2}$. Furthermore, if we assume A and C lie in S_+ then we get:

$$AC^T \geq \frac{CC^T}{2}$$

Proof. For every $u \in S_+$ with unit Euclidean norm we get

$$u^TAC^Tu = u^T(C+(A-C))C^Tu = u^TCC^Tu + u^T(A-C)C^Tu \tag{30}$$

$$= \|C^Tu\|_2^2 + u^T(A-C)C^Tu \overset{(a)}{\geq} \|C^Tu\|_2^2 - \|u^T(A-C)\|_2\|C^Tu\|_2 \tag{31}$$

$$\overset{(b)}{=} (\|C^Tu\|_2 - \|u^T(A-C)\|_2)\|C^Tu\|_2 \overset{(c)}{\geq} (\|C^Tu\|_2 - \frac{\alpha}{2}\|C^Tu\|_2 \tag{32}$$

$$\overset{(d)}{\geq} \frac{\|C^Tu\|_2^2}{2} = \frac{u^TCC^Tu}{2} \tag{33}$$

First, we need to clarify the equalities and inequalities from the previous equations. In the first row, we rewrite $A = C + (A-C)$ and then expand. In the second row, we start with the fact that $\|C^Tu\|_2^2 = u^TCC^Tu$, and at (a), we apply the product norm and change of signs. In (b), we factor out $\|C^Tu\|_2$, and in (c), we use the lemma stating that $\|A-C\|_2 \leq \frac{\alpha}{2}$, which implies $-\|A-C\|_2 \geq -\frac{\alpha}{2}$. In (d), we note that $\alpha > 0$ because we have $0 \leq \mu \ll \alpha \leq \beta$. Finally, using the fact that $\|u\|_2 = 1$ since they are unit vectors, we obtain

$$AC^T \geq \frac{CC^T}{2}.$$

□

We can use Lemma A.8 even when the constant is $\frac{\alpha(\beta+E)}{2\beta} + \frac{|E-E_{av}|}{2}$ instead of $\frac{\alpha}{2}$ and the proof is in a similar way since $\frac{\alpha(\beta+E)}{2\beta} + \frac{|E-E_{av}|}{2}$ is also positive.

LEMMA A.9. [86] *By using the results from Lemma A.5, prove that*

$$\|r_{\mathcal{T}+1}\|_2 \leq (1 - \frac{\eta\alpha^2}{2})\|r_{\mathcal{T}}\|_2$$

Proof. We will first present all the equations and inequalities, and then provide an explanation for each part.

$$\|r_{\mathcal{T}+1}\|_2^2 = \|(I-\eta C(W_{\mathcal{T}}))r_{\mathcal{T}}\|_2^2 \overset{(a)}{=} \|r_{\mathcal{T}}\|_2 - 2\eta r_{\mathcal{T}}C(W_{\mathcal{T}})r_{\mathcal{T}} \tag{34}$$

$$+ \eta^2 r_{\mathcal{T}}^T C(W_{\mathcal{W}})^T C(W_{\mathcal{T}})r_{\mathcal{T}} \tag{35}$$

$$\overset{(b)}{\leq} \|r_{\mathcal{T}}\|_2^2 - \eta r_{\mathcal{T}}\mathcal{J}(W_{\mathcal{T}})\mathcal{J}(W_{\mathcal{T}})^T r_{\mathcal{T}} \tag{36}$$

$$+ \eta^2\beta^2 r_{\mathcal{T}}^T\mathcal{J}(W_{\mathcal{T}})\mathcal{J}(W_{\mathcal{T}})^T r_{\mathcal{T}} \tag{37}$$

$$\overset{(c)}{\leq} \|r_{\mathcal{T}}\|_2^2 - (\eta - \eta^2\beta^2)\|\mathcal{J}(W_{\mathcal{T}})^T r_{\mathcal{T}}\|_2^2 \tag{38}$$

$$\overset{(d)}{\leq} \|r_{\mathcal{T}}\|_2^2 - \frac{\eta}{2}\|\mathcal{J}(W_{\mathcal{T}})^T r_{\mathcal{T}}\|_2^2 \tag{39}$$

$$\overset{(e)}{\leq} (1 - \frac{\eta\alpha^2}{2})\|r_{\mathcal{T}}\|_2^2 \tag{40}$$

At (a) we use the properties of the two norm, i.e. $\|x\|_2^2 = x^T x$ and then simplify. At (b) we use results from Lemma A.8 and the fact that $\|\bar{\mathcal{J}}(W_{\mathcal{T}+1}, W_{\mathcal{T}})\|_2 \le \beta$ and Definition A.4. At (c) we combine what we have at (b) and use the fact that $r_{\mathcal{T}}^T \mathcal{J}(W_{\mathcal{T}})\mathcal{J}(W_{\mathcal{T}})^T r_{\mathcal{T}} = \|\mathcal{J}(W_{\mathcal{T}})^T r_{\mathcal{T}}\|_2^2$. At (d) we use the fact that $\eta \le \frac{1}{2\beta^2}$ and plug in η at (c). At (e) we use the fact that $\|\mathcal{J}(W_{\mathcal{T}})\|_2 \ge \alpha$ and so $-\|\mathcal{J}(W_{\mathcal{T}})\|_2 \le -\alpha$. ◻

In the next Lemma, we are going to introduce the perturbed dataset:

Lemma A.10. *By following the results from Lemma A.6 prove that,*

$$\|r_{t+1}\|_2^2 \lessapprox (1 - \eta(\frac{\alpha}{2} + 2E_{J_2}\alpha + 2E_{J_1}\alpha - 2\eta E_{J_2}\frac{\beta^3}{2} - 2\eta E_{J_1}\frac{\beta}{3}))\|r_t\|_2^2$$

Proof. A portion of the proof follows a similar approach to Lemma A.9, and we will first present each equality and inequality before explaining them in detail. From Lemma A.6, when the perturbation occurs in the data rather than the labels, we have $r_{t+1} = (I - \eta(C(W_t + E_{J_2}\mathcal{J}(W_t)^T + \bar{\mathcal{J}}(W_{t+1}, W_t)E_{J_1})))r_t$. We use similar reasoning to Lemma A.9 to derive the result.

$$\begin{aligned}
r_{\tau+1}^T r_{\tau+1} &= (r_\tau^T - \eta r_\tau^T C(W_\tau) - \eta E_{J_2} r_\tau^T \mathcal{J}(W_\tau) - \eta E_{J_1} r_\tau^T \bar{\mathcal{J}}(W_{\tau+1}, W_\tau)^T) && (41)\\
&\quad (r_\tau - \eta C(W_\tau) r_\tau - \eta E_{J_2}\mathcal{J}(W_\tau)^T r_\tau - \eta \bar{\mathcal{J}}(W_{\tau+1}, W_\tau) E_{J_1} r_\tau) && (42)\\
&\overset{(a)}{\lessapprox} \|r_\tau\|_2^2 - \frac{\eta\alpha^2}{2}\|r_\tau\|_2^2 - 2\eta E_{J_2} r_\tau^T \mathcal{J}(W_\tau) r_\tau - \eta E_{J_1} r_\tau^T \bar{\mathcal{J}}(W_{\tau+1}, W_\tau) r_\tau && (43)\\
&\quad + 2\eta^2 E_{J_2} r_\tau^T \mathcal{J}(W_\tau) C(W_\tau) r_\tau + 2\eta^2 E_{J_1} r_\tau^T \bar{\mathcal{J}}(W_{\tau+1}, W_\tau) C(W_\tau) r_\tau && (44)\\
&\overset{(b)}{\le} (1 - \eta(\frac{\alpha}{2} + 2E_{J_2}\alpha + 2E_{J_1}\alpha - 2\eta E_{J_2}\frac{\beta^3}{2} - 2\eta E_{J_1}\frac{\beta}{3}))\|r_t\|_2^2 && (45)
\end{aligned}$$

At step (a), we applied the bounds from Lemma A.9, and the remaining part simplifies the multiplication from the first equality. In step (b), we use the facts that $\|\mathcal{J}(W_\tau)\|_2 \le \beta$, $\|\bar{\mathcal{J}}(W_{\tau+1}, W_\tau)\|_2 \le \beta$, and also that $-\|\mathcal{J}(W_\tau)\|_2 \le -\alpha$ and $-\|\bar{\mathcal{J}}(W_{\tau+1}, W_\tau)\|_2 \le -\alpha$.

◻

By continuing the calculations in Lemma A.10 back to the first iteration and assuming $\mathcal{T}$ iterations, we obtain:

$$\|r_{\mathcal{T}}\|_2^2 \lessapprox \left(1 - \eta\left(\frac{\alpha}{2} + 2E_{J_2}\alpha + 2E_{J_1}\alpha - 2\eta E_{J_2}\frac{\beta^3}{2} - 2\eta E_{J_1}\frac{\beta}{3}\right)\right)^{\mathcal{T}} \|r_0\|_2^2 \quad (46)$$

Now we give the Lemma about the bounds for T_{total} where T_{total} is the number of iterations

Lemma A.11. *(Bounds for iterations)*
Continuing with the results from Equation 46 and if $\nu > 0$ to bound the error after T_{total} iterations $\|r_{T_{total}}\|_2 = \|f(W_{T_{total}}, X) - Y\|_2 \le \nu$ then after T_{total} iterations the Neural Network f correctly classifies where T_{total} is

$$T_{total} \ge \mathcal{O}(\frac{1}{\eta(\frac{\alpha}{2} + E_{J_2}\alpha + E_{J_1}\alpha - \eta E_{J_2}\frac{\beta^3}{2} - \eta E_{J_1}\frac{\beta}{3})}\log\frac{\|r_0\|_2}{\nu})$$

Proof. We first need to establish some general formulas that will be useful for the subsequent steps. Specifically, we need to show that for any $0 < x < 1$, the inequality $\log\left(\frac{1}{1-x}\right) \ge \log(1+x)$ holds.

First, we subtract $\log(1+x)$ from both sides of the inequality:

$$\log\left(\frac{1}{1-x}\right) - \log(1+x) \ge 0.$$

Using the logarithmic identity $\log(a) - \log(b) = \log\left(\frac{a}{b}\right)$, this simplifies to:

$$\log\left(\frac{1}{(1-x)(1+x)}\right) \geq 0.$$

Now, we simplify the expression inside the logarithm:

$$(1-x)(1+x) = 1 - x^2.$$

Thus, the inequality becomes:

$$\log\left(\frac{1}{1-x^2}\right) \geq 0,$$

which is equivalent to:

$$\frac{1}{1-x^2} \geq 1.$$

Next, we multiply both sides by $1 - x^2$ (which is positive for $0 < x < 1$):

$$1 \geq 1 - x^2.$$

This is true since $0 < x < 1$ and thus it proves our inequality.

Furthermore, we need to prove that $g(x) = \log(1+x) - \frac{x}{2}$ is also increasing. We still need the derivatives and we get

$$g'(x) = \frac{1}{x+1} - \frac{1}{2} = \frac{1-x}{2(x+1)}$$

Which means that $g'(x) > 0$ for $0 < x < 1$.

We can now proceed with the proof of the lemma using the results from Equation 46. For some $\nu > 0$, we require that $\|r_{T_{total}}\|_2 \leq \nu$, where T_{total} is the number of iterations needed for accurate classification. We want ν to be very small, which implies that $r_{T_{total}} = f(W_{T_{total}}, X) - y$ should also be very small. Expanding from Equation 46, we obtain

$$(1 - \eta\left(\frac{\alpha}{2} + 2E_{J_2}\alpha + 2E_{J_1}\alpha - 2\eta E_{J_2}\frac{\beta^3}{2} - 2\eta E_{J_1}\frac{\beta}{3}\right))^{T_{total}}\|r_0\|_2 \leq \nu.$$

Furthermore, we can express this as

$$1 - \eta\left(\frac{\alpha}{2} + 2E_{J_2}\alpha + 2E_{J_1}\alpha - 2\eta E_{J_2}\frac{\beta^3}{2} - 2\eta E_{J_1}\frac{\beta}{3}\right) = 1 - A.$$

$$(1-A)^{T_{total}} \overset{\text{(a)}}{\leq} \frac{\nu}{\|r_0\|_2} \tag{47}$$

$$T_{total}\log(1-A) \overset{\text{(b)}}{\leq} \log\frac{\nu}{\|r_0\|_2} \tag{48}$$

$$T_{total}\log(\frac{1}{1-A}) \overset{\text{(c)}}{\geq} \log(\frac{\|r_0\|_2}{\nu}) \tag{49}$$

$$\tag{50}$$

At (a), this is valid because we move $\|r_0\|_2$ to the other side without changing the sign, as it is positive. In (b), we apply the logarithm and use the property that allows us to bring powers in front of the log. In (c), we multiply both sides by -1 and make the necessary adjustments to the logarithmic functions. Thus, using what we established at the beginning of the proof of this lemma, we obtain

$$T_{total} \log(1 + A) \geq \log \frac{\|r_0\|_2}{\nu}.$$

Moreover, from another result earlier in this lemma, we have

$$T_{total} \frac{A}{2} \geq \log \frac{\|r_0\|_2}{\nu}.$$

This leads us to our final results.

$$T_{total} \geq \frac{2}{A} \log \frac{\|r_0\|_2}{\nu} = \frac{2}{\eta(\frac{\alpha}{2} + 2E_{J_2}\alpha + 2E_{J_1}\alpha - 2\eta E_{J_2}\frac{\beta^3}{2} - 2\eta E_{J_1}\frac{\beta}{3})} \log \frac{\|r_0\|_2}{\nu}$$

□

We now present the Main Theorem, with certain parts modified from the results of Theorem 4.1 in.[1]

Theorem A.12. *(Main Theorem)*
Assume we apply gradient descent using mean-squared error (MSE) loss to train a neural network on a dataset with perturbed data. Suppose the Jacobian is L-smooth. The Coresets found by CRUST,[1] considering perturbed datasets where δ represents the fraction of noisy data and k is the Coreset size, approximate the Jacobian matrix with an error of at most $\epsilon \leq \mathcal{O}\left(\frac{\delta\alpha^2}{k\beta\log(\delta)}\right)$, where $\alpha = \sqrt{r_{min}}\sigma_{min}(\mathcal{J}(W, X_S)) - E_{min}$ and $\beta = \|\mathcal{J}(W, X)\|_2 + \epsilon + E_{max}$, with further details regarding E_{min} and E_{max} provided in the proof. For $L \leq \frac{\alpha\beta}{L\sqrt{2k}}$ and a step size $\eta = \frac{1}{2\beta^2}$, after $T_{total} \geq \mathcal{O}\left(\frac{1}{\eta\left(\frac{\alpha}{2} + E_{J_2}\alpha + E_{J_1}\alpha - \eta E_{J_2}\frac{\beta^3}{2} - \eta E_{J_1}\frac{\beta}{3}\right)} \log \frac{\|r_0\|_2}{\nu}\right)$ iterations, the network correctly classifies all selected elements.

Proof. The proof regarding T_{total} and the network's correct classification is based on Lemma A.11, which helps us determine the values for α and β. Consider a vector v, where we define $\hat{p} = \mathcal{J}_r(W, X_S + E_{X_S})v$ and $p = \mathcal{J}(W, X_S + E_{X_S})v$ (with $\mathcal{J}_r$ as described in[1]). As explained in that reference, to obtain $\hat{p}$, we need to multiply the entries of p by a factor between r_{min} and r_{max}. Thus, when working with vectors over S_+, we have

$$\sqrt{r_{min}}\sigma_{min}(\mathcal{J}(W, X_S + E_{X_S}), S_+) \leq \sigma_{i\in[k]}(\mathcal{J}_r(W, X_S + E_{X_S}), S_+) \leq \sqrt{r_{max}}\|\mathcal{J}(W, X_S)\|_2.$$

After analyzing the Jacobian in the middle part and subtracting that value from all three sides, we can simplify the left and right sides to obtain E_{min} and E_{max}.

The proof concerning ϵ is straightforward; we can follow the steps outlined in.[1] In their proof, they focus on the residual $r = f(W, X) - Y$, which involves noisy labels. In our case, however, we are dealing with perturbed images. The key difference is that δ in our context pertains to the noise introduced by perturbations to the images, rather than being a parameter related to flipped labels. □

APPENDIX B. ABLATION: SIZE OF REPLAY MEMORY

To select an appropriate size for the experience replay buffer (i.e. coreset in Continual CRUST) in our memory-based CIL algorithms, we run experiments on the blurry CIL setting of five experiences with two major tasks per experience and a label noise level of 40% (40% of samples are mislabelled, uniform randomly) for both MSTAR and SynthWakeSAR. We compare the performance of the benchmark random replay method with Continual CRUST. Results appear in Fig. 12. We see MSTAR performance tends to approximately plateau with a coreset size of 200 samples. The SynthWakeSAR is close to plateauing with a buffer size of 2000 samples. Based on these observations, we select these as the replay buffer sizes for the experiments in the main text.

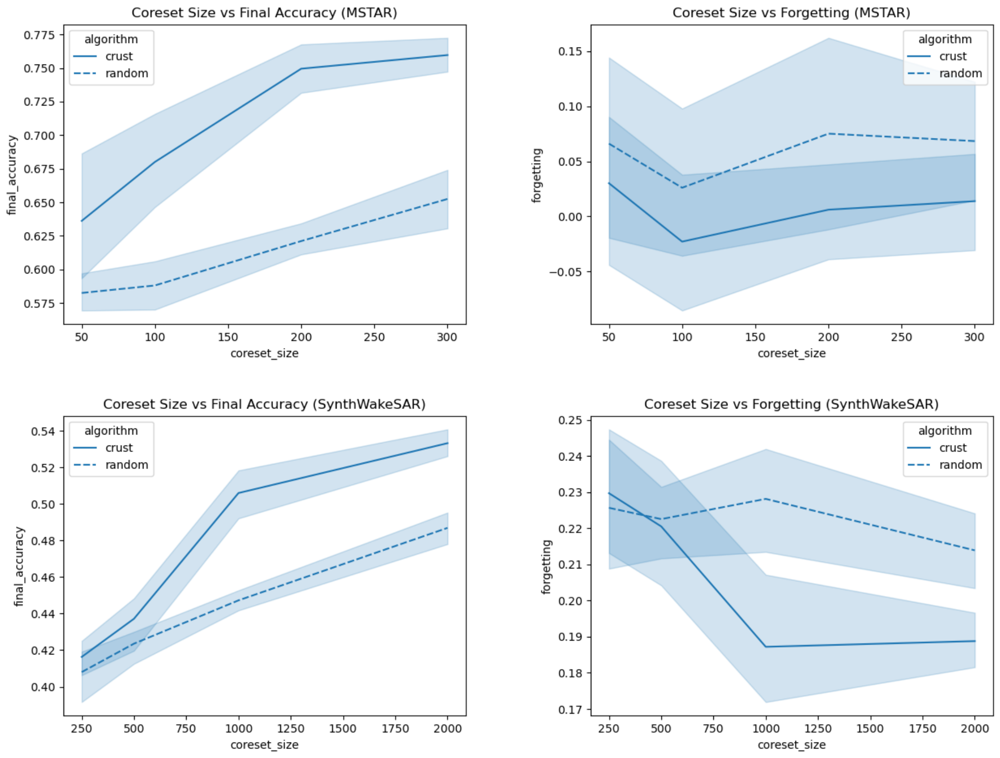


Figure 12. **Top-Left:** Final experience accuracy as coreset size is increased on the blurry 2x5 MSTAR setting with 40% label noise. **Top-Right:** Forgetting metric as coreset size is increased on the blurry 2x5 MSTAR setting with 40% label noise. **Bottom-Left:** Final experience accuracy as coreset size is increased on the blurry 2x5 SynthWakeSAR setting with 40% label noise. **Bottom-Right:** Forgetting metric as coreset size is increased on the blurry 2x5 SynthWakeSAR setting with 40% label noise.