Journal of Machine Learning Research 1 (2022) 1-48 Submitted 4/00; Published 10/00

Adapting Machine Learning Classifiers using Sequential Copies

Nahuel Statuto NAHUEL.STATUTO@ESADE.EDU
Department of Operations, Innovation and Data Sciences
Universitat Ramon Llull, ESADE
Sant Cugat del Vallès, 08172, Catalonia, Spain

Irene Unceta IRENE.UNCETA@ESADE.EDU
Department of Operations, Innovation and Data Sciences
Universitat Ramon Llull, ESADE
Sant Cugat del Vallès, 08172, Catalonia, Spain

Jordi Nin JORDI.NIN@ESADE.EDU
Department of Operations, Innovation and Data Sciences
Universitat Ramon Llull, ESADE
Sant Cugat del Vallès, 08172, Catalonia, Spain

Oriol Pujol ORIOL_PUJOL@UB.EDU
Department of Mathematics and Computer Science
Universitat de Barcelona
Barcelona, 08007, Catalonia, Spain

Editor:

Abstract

Copying corresponds to the process of replicating the decision behaviour of a machine learning model using another that is endowed with new features and characteristics. This process plays a relevant role in circumstances where external constraints limit the performance of a predictive system. In such cases, copies can retain the original prediction capabilities, while showing additional properties that better adapt the existing solution to the demands of its environment. So far, copies have been studied under the single-pass framework. In this work, we propose an extension to introduce the sequential approach. The main contribution of this new approach is to limit the number of computer resources needed to train or maintain a copy. This computer resource reduction allows companies to reduce the maintenance costs of machine learning models in production. To empirically validate the sequential approach, this work contains experiments with synthetic and real-world datasets.

Keywords: Sustainable AI, machine learning copies, knowledge transfer

1. Introduction

Machine learning (ML) based solutions have become increasingly widespread in many industries and sectors. Examples range from credit card fraud detection (Gómez et al., 2018) to targeted advertising on social networks (Gharibshah and Zhu, 2021), aid in medical diagnosis (Shehab et al., 2022), or supply chain optimization (Feizabadi, 2022). ML can draw from large databases to automate processes previously done by humans, leading to higher

precision while incurring lower costs and enabling a more optimal allocation of tasks. Nevertheless, ML requires constant supervision. To ensure sustained delivery models need to deal with continuous changes in their environment. These changes can be either internal or external to the companies and may arise from new production requirements, updates in the technological infrastructure, novel market trends, or shifts in the existing regulation. Failure to adapt to such changes can result in decreased performance and hinder the efficiency of this technology. Monitoring is therefore a key activity of any industrial ML pipeline.

Once served into production, ML models are regularly inspected to check for signs of performance deviation. Depending on the considered sector, these signs can appear as early as a few months, or even a few weeks. Whenever they do, models are retrained from scratch and substituted (Wu et al., 2020). This is a process that can consume very substantial computational resources (Chen, 2019). Given the increasing complexity of ML architectures, training a single model can be a long, costly task. Managing and updating several ML models at a time will quickly become unaffordable for most businesses. Hence, long-term sustainability is one of the main challenges faced by industrial ML practitioners today (Paleyes et al., 2022).

In this context, differential replication through copying (Unceta et al., 2020a) has been proposed as a viable, agile solution to re-use a model's acquired knowledge to train a new one better adapted to the environment demands (Unceta et al., 2020b). This approach, which builds upon previous ideas on knowledge distillation (Hinton et al., 2015b; Buciluǎ et al., 2006b), effectively circumvents the issue of having to retrain models from scratch. It can therefore bring numerous benefits in terms of cost and use of companies' resources. In practice, copying refers to the projection of an existing decision function onto a new hypothesis space that satisfies the new requirements. As opposed to scenarios where label probabilities are used as soft targets for training the new model (Liu et al., 2018; Wang et al., 2020), the copying process is agnostic to the form of the original model. Information on its predictions can only be accessed through a hard-membership query interface, Finally, copying allows no access to the original training data. Instead, training is performed using a set of synthetic samples randomly drawn from a given probability distribution and labeled according to the predictions of the original model.

Solving the copying problem in theory requires solving an optimization problem where we simultaneously optimize the copy parameters and the synthetic samples used for training. Practical approaches to copying, however, have so far relied on a simplification of this problem. Here, a large set of synthetic data points is generated and labeled in a single pass. This dataset is then used to optimize copy parameters. While successfully demonstrating the value of copies in practice, this approach has several drawbacks. Notably, it makes inefficient memory usage and requires large computational times.

In this article, we propose a novel approach to copying based on an iterative scheme. Copies are built sequentially from small synthetic datasets. At each step, copies are updated to account for the new information. This avoids the need for keeping a massive synthetic dataset in memory and avoids redundancy. In particular, we propose a methodology that exploits the memorization capacity of the copy to sequentially refine its decision boundary until it replicates the original model behavior to a desired level of fidelity. We combine this approach with a data points dropout technique resulting in a high-fidelity and low-resources

copying methodology. This is an important result as it overcomes the main drawback of copies and enables the process of copying to be performed effectively and efficiently.

Section 2 describes the context in which copies are proposed and provides a brief overview of related techniques. We discuss differential replication through copying and introduce the single-pass approach as the most straightforward solution to this problem.

In Section 3, we propose the sequential approach to copying, an alternating optimization approach that is efficient in memory and time. We start by demonstrating the convergence of this approach to the optimal solution. Then, we study data uncertainty as a model compression measure to drive the copying optimization problem and how we can use it to drop redundant data points. Finally, this section ends by proposing a regularization term to avoid model forgetting.

Once the methodology has been described, Section 4 empirically demonstrates the validity of the sequential approach performing a large set of experiments with the UCI database problems. In the experiments, we evaluate the performance of the sequential copying framework in terms of accuracy and efficiency. We end this section with a discussion of the achieved results. Finally, Section 5 provides some conclusions and future work.

2. Background on copying

Differential replication through copying was first introduced by (Unceta et al., 2020b) as a solution to the problem of *environmental adaptation*. This problem refers to situations where a ML model designed under a set of constraints is required to fulfill a new set of requirements imposed by changes in its environment. To accommodate the new requirements, the model needs to adapt from the *source scenario* s, where it was originally trained and served into production, to a *target scenario* t, where it is currently being deployed.

2.1 The problem of environmental adaptation

Formally speaking, we can define this problem as follows. Let us consider a task $\mathcal{T}$ and a domain $\mathcal{D}$. Let us also consider a trained model $h \in \mathcal{H}_s$, for $\mathcal{H}_s$ a model hypothesis space that solves $\mathcal{T}$ in $\mathcal{D}$ under a given set of constraints $\mathcal{C}_s$. The problem of environmental adaptation refers to situations where the original set of constraints $\mathcal{C}_s$ evolves towards a new set $\mathcal{C}_t$. Under these circumstances, we need to define a potentially different hypothesis space $\mathcal{H}_t$ in the same domain $\mathcal{D}$ and for the same task $\mathcal{T}$, which is compatible with the new constraints set $\mathcal{C}_t$ [1]. Unless the considered model h is compatible with this new hypothesis space, i.e., with the new constraints, it will be rendered obsolete. Hence, environmental adaptation refers to the need to adapt h, such that $h \in \mathcal{H}_t$.

Source Scenario for $\mathcal{T}$ in $\mathcal{D}$			Target Scenario for $\mathcal{T}$ in $\mathcal{D}$	
		$\rightarrow$		
$\underset{\text{for } h \in \mathcal{H}_s}{\text{maximize}}$	$\mathsf{P}(y\|x;h)$		$\underset{for\ h \in \mathcal{H}_t}{\text{maximize}}$	$\mathsf{P}(y\|x;h)$
subject to	$\mathcal{C}_s$		subject to	$\mathcal{C}_t$

1. Note that in some cases the source and target hypothesis spaces $\mathcal{H}_s$ and $\mathcal{H}_t$ may be the same. However, in the most general case, where the new set of constraints defines a new set of feasible solutions, they are not.

Note that this problem is different from those of*domain adaptation* and *transfer learning*. The former refers to cases where there are two divergent domains. The goal is to adapt a model trained in a source domain $\mathcal{D}_s$ to a different but related target domain $\mathcal{D}_t$. Generally, this problem can be understood as arising from a change in the source and target data distributions. Contrary to this case, environmental adaptation preserves the domain. Instead, there is a change in the set of constraints. In the case of transfer learning, it refers to scenarios where the knowledge acquired when solving one task $\mathcal{T}_s$ is recycled to solve a different, yet related task $\mathcal{T}_t$ (Pan and Yang, 2010). In contrast, the environmental adaptation problem preserves the task.

There exist different approaches to solving the environmental adaptation problem. The most straightforward solution is simply retraining or fine-tuning the existing model under the new set of constraints (Barque et al., 2018). This approach requires having access to the original training data, as well as to the model internals. In many situations, however, one or both elements cannot be accessed or can only be accessed partially. Depending on the level of accessibility to the original model and its training data, other approaches are also possible, including retraining, the use of *wrappers* (Mena et al., 2019, 2020) or edited data subsets (Song et al., 2008), teacher-student networks and distillation mechanisms (Buciluǎ et al., 2006a; Hinton et al., 2015a; Szegedy et al., 2016; Yang et al., 2019), label regularization (Müller et al., 2019; Yuan et al., 2020), label refinery (Bagherinezhad et al., 2018), or the creation of generative synthetic data (Buciluǎ et al., 2006a; Zeng and Martinez, 2000). A complete overview about all the approaches to solving the problem of environmental adaptation can be found in (Unceta et al., 2020b). In the most extreme case, where we assume no access to the training dataset or the model internals, we can solve environmental adaptation using differential replication through copying Unceta et al. (2020a).

Differential replication enables the projection of an existing decision boundary over a new hypothesis space, which is compatible with the target scenario. Differential replication through copying refers to scenarios where this process is agnostic to the given model's internals and training data. In a classification setting and given an original classifier, we can frame copying as the problem of obtaining a new classifier, which does not necessarily belong to the same model family, yet displays the same performance and decision behavior.

In what follows, we introduce the problem of differential replication through copying and discuss possible approaches to solving it in practice.

2.2 Differential replication through copying

Let us assume a set of data points $D \subseteq \mathbb{R}^d$ for d the dimensionality of the input space labeled according to a set of classes C_i of cardinality n_c. Let us also consider a classifier $f_{\mathcal{O}} \in \mathcal{H}_s$ trained on these data, such that $f_{\mathcal{O}} : \mathbb{R}^d \longrightarrow \{0,1\}^{n_c}$. In the most restrictive case, we consider $f_{\mathcal{O}}$ to be a hard decision classifier that only outputs one-hot encoded label predictions. In other words, for any given data point, we assume that $f_{\mathcal{O}}$ returns an n_c-output vector where all the elements are zeros except for the target label position i, which takes value 1. The goal of copying is to obtain a new classifier $f_{\mathcal{C}} \in \mathcal{H}_t$, parameterized by $\theta \in \Theta$, whose decision function mimics $f_{\mathcal{O}}$ all over the sample space.

In the empirical risk minimization framework, we can consider an empirical risk function that measures the discrepancy between two classifiers. Hence, we can write the copying

problem as a dual optimization over the unknown copy parameters θ and a set of synthetic data points S over which the empirical risk is evaluated[2]. This problem can be written as follows:

$$\begin{aligned} \underset{\theta,S}{\text{minimize}} \quad & \Omega(\theta) \\ \text{subject to} \quad & \|R^{\mathcal{F}}_{emp}(f_{\mathcal{C}}(\theta), f_{\mathcal{O}}) - R^{\mathcal{F}}_{emp}(f_{\mathcal{C}}(\theta^{\dagger}), f_{\mathcal{O}})\| < \varepsilon, \end{aligned} \tag{1}$$

for ϵ a defined tolerance and $\Omega(\theta)$ a measure of the complexity of the considered classifier (for example, the ℓ_p-norm of the parameters). $R^{\mathcal{F}}_{emp}$ is a risk function that measures the discrepancy between the original model $f_{\mathcal{O}}$ and the copy model $f_{\mathcal{C}}$ we are optimizing. We refer to this function as the *empirical fidelity error*. Finally, $\theta^{\dagger}$ is the solution to the following unconstrained optimization problem:

$$\theta^{\dagger} = \underset{\theta,S}{\text{argmin}} \quad R^{\mathcal{F}}_{emp}(f_{\mathcal{C}}(\theta), f_{\mathcal{O}}). \tag{2}$$

The solution to Equation 1 is the combination of a set of synthetically generated data and copy model parameters that achieves minimum capacity, while minimizing the empirical risk. If the original decision boundary is compatible with the new model hypothesis space, $f_{\mathcal{O}} \in \mathcal{H}_t$, then the solution of the unconstrained problem always guarantees that $R^{\mathcal{F}}_{emp}(f_{\mathcal{C}}(\theta^{\dagger}), f_{\mathcal{O}}) = 0$. This is because the problem resulting from labeling any set of data points using the original hard-label classifier is always separable.

2.3 The single-pass approach

In the *single-pass approach* (Unceta et al., 2020b), a sub-optimal solution to the copying problem is found by decoupling the optimization problem into two separate steps and sequentially solving each one. First, a suitable synthetic set S^* is found. Then, the copy parameters θ^* are optimized over this set. The process can be summarized as follows:

- **Step 1: Synthetic sample generation**. An optimal synthetic set S^* is created by drawing samples independently at random from a probability density function P_S that covers the operational space of the copy (region of the input space where the copy is defined to mimic the behavior of the original model), i.e.

$$S^* = \{z_j | z_j \sim P_{\mathcal{S}},\ j = 1, \ldots, N\}. \tag{3}$$

  In the simplest case, P_S can be assumed to follow a uniform distribution. Or even a uniform of Gaussian distribution, provided the original data points are normalized.[3].

2. Remember that we do not have access to any training data
3. For a more in depth discussion of different choice for the probability density function P_S we refer the reader to (Unceta et al., 2020c).

- **Step 2: Building the copy**. During the second step, the optimal parameter set for the copy is obtained by solving Equation 2 for the set S^*

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \quad R^{\mathcal{F}}_{emp}(f_{\mathcal{C}}(\theta), f_{\mathcal{O}})\Big|_{S=S^*} \tag{4}$$

The single-pass strategy is a simplified approach to finding a solution compatible with the problem modeled in Equation 1. Step 1 generates data agnostically. This simplifies the process, since no optimization step is required. But it also introduces the need for a large dataset. Step 2 focuses on solving the unconstrained version of the copying problem, as defined in Equation 2. This approach can be enforced in scenarios where the classifier complexity can be directly modeled. However, it does not cover the general case.

The single-pass method enables several theoretical and modeling simplifications. However, introducing these simplifications requires setting a substantial number of critical parameters beforehand. It also requires selecting a model that satisfies the indispensable constraints for the unconstrained problem to be a good approximation of the general setting described in Equation 1. Finally, avoiding the optimization of the synthetic sample set only guarantees a good performance if we generate a sufficiently large synthetic dataset.

The practical implementation of the single-pass approach requires either a machine learning model that scales well with the number of samples or applying online learning algorithms. As discussed below, both have their downsides. Firstly, our capacity to learn a large dataset with a single model is bounded by the available memory. Secondly, holding a lot of data samples in memory during training is costly and makes a sub-optimal use of computational resources. It is also no guarantee of performance. When implementing the single-pass approach in a single step, we do so "blindly" which results in a significantly inefficient process. Finally, when approaching single-pass using an online learning strategy, we can circumvent the issue of memory usage. Yet, we will get very slow convergence to the optimal solution. Hence, the process will be very time-consuming.

In the following section, we introduce a new approximate algorithm that solves Equations1 and 2 using an alternating optimization scheme. As a result, a fast and memory-efficient approach to the copying problem is obtained.

3. The sequential approach

In what follows, we introduce the sequential approach and provide theoretical guarantees for convergence. We begin by introducing two theorems that demonstrate that the sequential copy process converges in parameters and in behavior to the single-pass approach when the conditions are optimal. We then describe the details for a practical implementation of the sequential approach and introduce several optimizations to ensure low memory usage and fast convergence. In particular, we argue that a perfect copy should be able to compress the synthetic data points in its parameters and introduce epistemic uncertainty as a reliable measure of data compression in copies. Following this idea, we introduce a filtering process that controls the degree of compression for each sample and discards those that have already been learned. As a result of this process, very little data are required during each learning step. Finally, we propose a policy for automatic hyper-parameter tuning to ensure the optimal implementation of the sequential copying approach in practice.

3.1 An alternating optimization algorithm for copying

Let us start by introducing an alternating optimization approach to solving Equation 1. This algorithm iteratively alternates between two optimization steps. At each iteration i, partial optimizations are performed for the synthetic set S and the copy parameters θ as follows:

- **Step 1: Sample optimization**. The optimal synthetic set at iteration i is that for which the empirical fidelity error is maximal for the previous model solution θ^*_{i-1}. That is,

$$S^*_i = \arg\max_S R^{\mathcal{F},S}_{emp}\Big|_{\theta=\theta^*_{i-1}} \tag{5}$$

- **Step 2: Copy parameter optimization**. The optimal copy parameter set θ^*_i at iteration i over the samples S^*_i is obtained by solving

$$\begin{aligned} &\underset{\theta}{\text{minimize}} \quad \Omega(\theta) \\ &\text{subject to} \quad \|R^{\mathcal{F}}_{emp}(f_{\mathcal{C}}(\theta), f_{\mathcal{O}}) - R^{\mathcal{F}}_{emp}(f_{\mathcal{C}}(\theta^\dagger), f_{\mathcal{O}})\|\Big|_{S=S^*_i} < \varepsilon. \end{aligned} \tag{6}$$

The rationale for this algorithm is as follows. We begin each iteration i by finding a set of synthetic data points that maximizes the empirical risk. This set consists of points that are not correctly modeled by the copy resulting from the previous iteration $i-1$. Note that, as previously discussed, given a copy model with enough capacity, the empirical risk can be reduced exactly to zero. Hence, these are the points that contribute the most to the loss function. Then, in Step 2 we minimize the empirical loss over S_i while keeping the complexity of the copy model as small as possible.

The remainder of this section is focused on solving Equations 5 and 6.

3.2 Step 1: Sample optimization

We begin by formally introducing the sequential approach to sample generation and studying its convergence properties. This will serve as a foundational framework to build the final algorithm that solves Equation 5. To that end, we first move Equation 2 into a probabilistic setting and introduce two theorems that demonstrate that the sequential copying process converges in parameters and in behavior to the single-pass approach when the conditions are optimal. This is when the number of samples tends to infinity.

3.2.1 The sequential framework

Let us introduce a sequence of finite subsets S_i of the form

$$S_i \subseteq S_{i+1} \subseteq \cdots \subseteq S \tag{7}$$

for $i \in \mathbb{N}$. This sequence is defined such that, for increasing values of the iteration parameter i, the subset S_i grows closer to S ($|S| = \aleph^0$). In the limit where i approaches infinity, the set tends to S as $\lim_{i\to\infty} S_i = S$.

The main idea behind the sequential approach is as follows: if the condition that S_i converges to S is sufficient to ensure that the copy parameters θ_i^* obtained by optimizing the copy over S_i equally converge to θ^* (the copy parameters optimized over S), then we can iteratively approximate the optimal copy parameters on synthetic sets generated by drawing samples independently at random from a given probability density function. Hence, the first step is showing that this statement holds.

To that end, we rewrite Equation 2 in probabilistic terms so that it allows for a more general set of models [4] and redefine the unconstrained problem of copying as the solution to the following empirical distributional problem

$$\theta^* = \arg\max_{\theta} \sum_{z \in S} \mathcal{P}(\theta | f_{\mathcal{O}}(z), f_{\mathcal{C}}(z)) = \arg\max_{\theta} F(\theta), \tag{8}$$

where $S = \{z | z \sim P_{\mathcal{S}}\}$ is a synthetic dataset such that $|S| = \aleph^0$.

We now introduce Theorem 1, which shows that we can approximate the solution to Equation 8for S using a sequence of iterative values of S_i, as above defined.

Theorem 1 *Let $S_i \subseteq S_{i+1} \subseteq \cdots \subseteq S$ be a subsets' convergent sequence. Then, a sequence of functions $\left\{F_i\right\}_i$ defined as $F_i(\theta) = \sum_{z \in S_i} \mathcal{P}(\theta | f_{\mathcal{C}}(z, \theta), f_{\mathcal{O}}(z))$, uniformly converges to $F(\theta) = \sum_{z \in S} \mathcal{P}(\theta | f_{\mathcal{C}}(z, \theta), f_{\mathcal{O}}(z))$.*

From Theorem 1, we can prove Theorem 2 for the convergence of the parameters.

Theorem 2 *Under the conditions of Theorem 1, a sequence of parameters $\left\{\theta_i^*\right\}_i$ defined as $\theta_i^* = \arg\max_{\theta \in \Theta} F_i(\theta)$, converges to $\theta^* = \arg\max_{\theta \in \Theta} F(\theta)$, where Θ is the complete set of parameter.*

The complete mathematical proof of convergence of the copy parameter sequence in Theorem 2 can be found in Appendix A.

Definition: The *sequential approach* learning algorithm for copying is an iterative process that incrementally optimizes the copy parameters θ_i over iterative sets of synthetic data points S_i, $i = 1 \dots T$.

The sequential approach is described in Algorithm 1. We start by generating a first synthetic set S_0 of size n and optimize the copy parameters accordingly. Then, as described in line 4, at each iteration t the new synthetic set S_i includes S_{i-1}, and the number of samples is increased by n at each iteration. The cardinality of S_i is therefore given by $\left|S_i\right| = i \cdot n$ and grows linearly with i. This policy has been chosen as the most straightforward strategy for building the subsets in Equation 7. It is worth noting that line 5 describes the optimization of the algorithm considering the solution of the previous step.

For illustrative purposes, we assay the sequential approach in a toy binary classification dataset: the *spirals* problem. We use this dataset to train a Gaussian kernel SVM that

4. Observe that we can easily recover the empirical risk minimization framework considering probability density functions of the exponential family. Consider the empirical loss defined as $\frac{1}{N}\sum_{i=1}^{N} \ell(a, b; \theta)$, then

$$\arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \ell(a, b; \theta) = \arg\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} e^{-\gamma \cdot \ell(a, b; \theta)}.$$

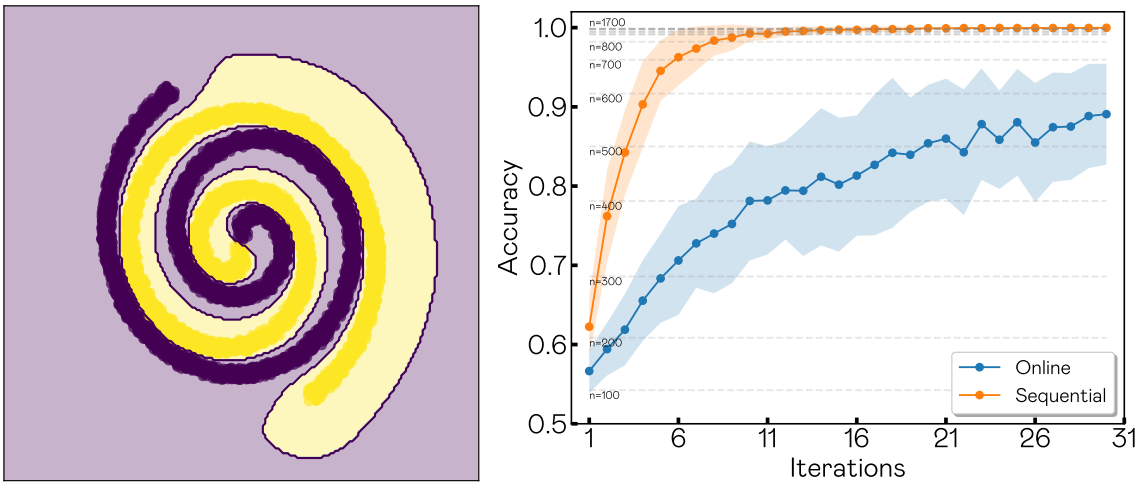
Algorithm 1 Sequential Copy(**int** T, **int** n, **Classifier** $f_{\mathcal{O}}$)

$S_0 \leftarrow \{(z_j, f_{\mathcal{O}}(z_j)) \mid z_j \sim P_{\mathcal{S}}, j = 1 \dots n\}$
$\theta_0^* \leftarrow \arg\max_\theta \sum_{z \in S_0} \mathcal{P}(\theta | f_{\mathcal{O}}(z), f_{\mathcal{C}}(z))$
for $i = 1$ to T **do**
    $S_i \leftarrow S_{i-1} \bigcup \{(z_j, f_{\mathcal{O}}(z_j)) \mid z_j \sim P_{\mathcal{S}}, j = 1 \dots n\}$
    $\theta_i^* \leftarrow \arg\max_\theta \sum_{z \in S_i} \mathcal{P}(\theta | f_{\mathcal{O}}(z), f_{\mathcal{C}}(z), \theta_{i-1}^*)$
end for
return θ_i^*

achieves perfect accuracy. The corresponding decision boundary is shown in Figure 1a), together with the original data points. We copy using a fully-connected neural network with three hidden *ReLu* layers, each consisting of with 64, 32, and 10 neurons, respectively, and a *SoftMax* output. We use this model to train copies using the single-pass approach, both in a single step and using an online strategy, and the sequential approach. In all cases, we use 1000 epochs and a batch size of 32 samples. Figure 1b) shows the average accuracy computed at different iterations for copies trained using the different methods. The online (in blue) and sequential (in orange) are trained by generating $n = 100$ new samples at each iteration t. The grey dashed lines correspond to the accuracy achieved by copies trained using the single-pass approach in a unique step, for different values of n. Note that the reported values correspond to the accuracy measure over the original test data points.

The first conclusion we can extract from this plot is the different convergence speeds. A copy trained using a unique step of the single-pass approach with $n = 500$ yields an accuracy of 0.85. When implementing this same approach in an online setting, the accuracy is several points below for $t = 5$, when the model has been fed the same number of points. Moreover, the maximum accuracy reached by the online learner after 30 iterations is approximately 0.85, after being fed 3000 samples in total. In the case of the sequential approach, at $t = 5$, when the copy has seen 500 samples, the average accuracy is 0.94. This seems to indicate that the sequential approach converges much faster than the single-pass approach, in any of its implementations. However, once the number of points grows up to 1000, both the unique step *single-pass* and the *sequential* settings show the same accuracy level (≈ 1), as we expected from our previous mathematical discussion.

The observed differences among the three training approaches are summarized in Table 1, where we consider three distinct categories for comparison purposes: *memory, accuracy* and *computational time*. The number of data points to which the model has been exposed seems to play a key role. When implementing the single-pass approach in a unique step, the generated dataset is only used once. A lot of data are therefore required to achieve high accuracy. Moreover, the number of training data points must be estimated at the beginning. Hence, this approach can easily lead to under or over-training and therefore to poor performance, or to unnecessary use of computational resources. In contrast, one can learn a single-pass copy using an online learning setting, where the copy model is exposed to a smaller batch of data at each iteration. This can help alleviate the issue of memory allocation. Yet, the resulting accuracy is bounded and strongly dependent on the value of n. While it is guaranteed to converge to the optimal solution, this is an asymptotic guarantee.



(a) (b)

Figure 1: **Single-pass vs Sequential approach.** a) Decision boundary learned by a Gaussian kernel SVM trained on the *spirals dataset*. b) Copy accuracy for both the sequential (orange) and the single-pass approaches averaged over 30 independent runs for $n = 100$. The grey dashed lines show the accuracy achieved using a unique step of the *single-pass* approach for different values of n, while the results for the online implementation are shown in blue for different iterations. The shaded regions correspond to one standard deviation from the mean.

Finally, sequential training combines the best of the previous two approaches. First, the number of points increases over time. Thus, there is no need to estimate this parameter at the beginning. In addition, the copying process can stop once the desired accuracy is reached. This is in line with the discussion for Theorem 1: the accuracy for the sequential approach increases monotonically with the number of iterations. The main drawback of this method is its computational cost. The sequential approach grows quadratically with the number of data points, as opposed to the linear growth of both the single-pass and pure online approaches. This means that, in its current implementation, the sequential approach is highly time-consuming.

In what follows, we discuss several improvements to overcome this obstacle.

3.2.2 Epistemic uncertainty as a model compression measure in copies

The ultimate goal of a machine learning model is to discover the relevant patterns contained in the data for a given task. In a traditional learning setting, this discovery takes the form of pattern compression in the model. But, defining a model compression measure is far from trivial. In the copying framework, the original model can be understood as representing the ground truth: disregarding the actual performance of the original model on the task it has been trained on, our goal is to reproduce its decision function. Due to the fact that

	Single-pass	Sequential	Online
Memory	Fixed large value Needs to be estimated	Increase monotonically	Fixed small value
Accuracy	High but N-dependent	Increase with iteration	Increase with iteration Upper bound is N-dependent
Time	$T \propto O(t \cdot N)$	$T \propto O(t^2 \cdot N)$	$T \propto O(t \cdot N)$

Table 1: Comparison table that summarize the principal differences between both three learning approaches: *single-pass*, *sequential* and *online*.

the original classifier produces a hard classification boundary on any dataset, we expect the copy to reproduce that boundary with zero uncertainty.

The statement above only holds for the copying setting, where the original decision boundary defines a fully separable problem. Hence, as opposed to the standard learning case, in the copying setting, aleatoric uncertainty should be non-existent. This means that all the uncertainty measured should correspond to that coming from the model itself, not from the data. In other words, when copying we should be able to effectively compress all the data patterns with perfect certainty.

Assumption: Any uncertainty measured during the copying process is epistemic[5].

It follows from this assumption that when the epistemic uncertainty over a given data point is minimum, then the data point is perfectly compressed by the model. This is, we can assume that a data point is perfectly learned by the copy when uncertainty is null.

Estimating uncertainty in practice requires that we measure the difference between the predictive distribution and the target label distribution. This can be done using various ratios or predictive entropy. We refer the reader to (DeVries and Taylor, 2018; Senge et al., 2014; Nguyen et al., 2018) for full coverage of this topic. For the sake of simplicity, here we consider the simplest approach to measuring uncertainty. For this purpose, we relax the hard-output constraint for the copy. Instead, we impose that, for any given data point, the copy returns a n_c-output vector of class probability predictions, such that

$$f_{\mathcal{C}} : \mathbb{R}^d \times \Theta \longrightarrow \big[0, 1\big]^{n_c}.$$

We define the *uncertainty*, ρ, of the copy for a given data point z as the normalized euclidean norm of the distance between the n_c-vectors output by the original model and the copy, such that

$$\rho(z, \theta) = \frac{\| f_{\mathcal{C}}(z, \theta) - f_{\mathcal{O}}(z) \|_2}{\sqrt{n_c}} \in [0, 1]. \tag{9}$$

5. Epistemic uncertainty corresponds to the uncertainty coming from the distribution of the model parameters. It can be reduced as the number of training samples increases. This allows us to single out the optimal model parameters. Thus, any measurement of uncertainty comes from the mismatch between the model parameters and the desired parameters, i.e. a perfect copy will have zero epistemic uncertainty.

A small ρ value indicates that the copy has low uncertainty (or strong confidence) in the class prediction output for z and that z is properly classified. A large ρ value indicates that, despite the copy model having strong confidence in the class predicted for z, this prediction is incorrect. Or alternatively, despite the prediction being correct, there is a large dispersion among the output class probabilities. Consider, for example, a case where the copy outputs a uniform class probability distribution over the different classes.

This uncertainty measure can be used in the sequential framework to assess how well the individual data points in S_i are compressed by the copy model. This makes this quantity a good measurement for assessing how well the copy model is mimicking the original classifier. And, it motivates us to use it as a realization of the empirical risk. For a given set of data points S_i the loss function is

$$R^{\mathcal{F}}_{emp}(f_{\mathcal{C}}(\theta_j), f_{\mathcal{O}}) = \frac{1}{|S_i|} \sum_{z \in S_i} \rho^2(z, \theta). \tag{10}$$

In Figure 2 we show the results of applying this measure for the different approaches to copying introduced in Figure 1b. In particular, the plot shows the average value of ρ_i over all the data points considered during that iteration and for all the independent runs. The average uncertainty of the sequential learning is extremely low by iteration $t = 10$. In comparison, both the single-pass and online learning approaches exhibit larger uncertainty levels throughout the process. This difference can be attributed to the fact that during the sequential training the model is continuously exposed to data from previous iterations, which helps reduce the uncertainty iteration by iteration. In contrast, the other two methods expose the model only once to the same data points. Even so, some differences can be observed among them too. At iteration $t = 13$, when both the single-pass and the online models have been exposed to $n = 1300$ data points, the uncertainty of the online learning is statistically lower than that of the single-pass. This indicates that the model trained with the online approach achieves a higher compression of the information. Mostly because it deals with a smaller dataset each time.

3.2.3 Filtering data

Including an uncertainty measure as part of the training algorithm allows us to estimate the degree of compression for every data point. A higher compression indicates that the copy model succeeds in capturing the pattern encoded by each point for the considered task. This estimation can be used for selecting those points that contribute the most to the learning process. By filtering out the rest of the samples, we can reduce the number of resources consumed when copying. Hence, we enforce a policy that uses uncertainty as a point selection criterion.

At each iteration, points with an uncertainty below a given threshold δ are removed from the learning process. Algorithm 2 explicitly shows this procedure. In line 1, a new sequential set is built by randomly sampling a new set of n points and adding to the elements of S_{i-1}. Then, in line 2, the uncertainty measure is used to select those points such that $\rho_j \leq \delta$. The set S_i that results from such sample selection is then used to optimize the copy parameters.

Note that this algorithm effectively solves the problem modeled in Equation 5. The uncertainty measure ρ is related to the empirical fidelity error $\mathcal{R}^{FC}_{emp}$. In particular, the

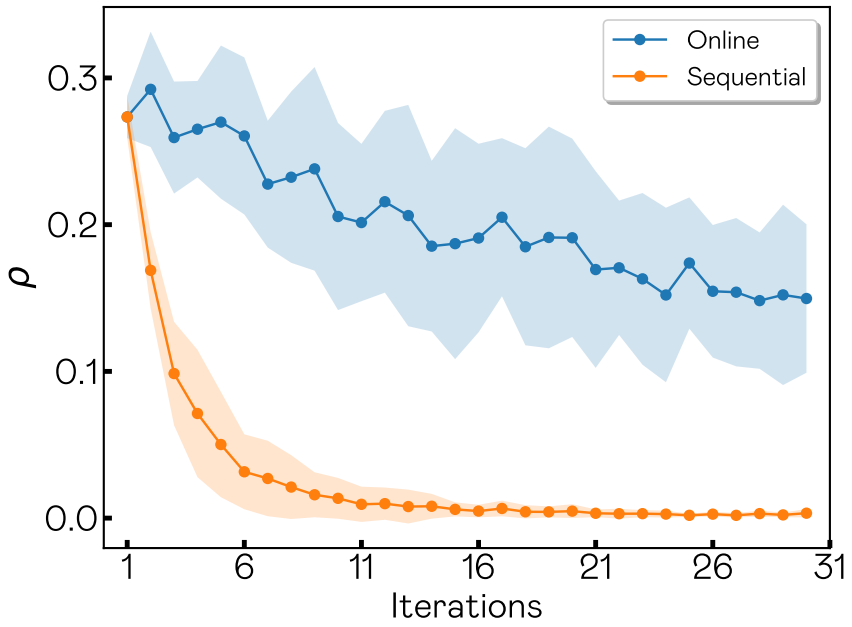


Figure 2: **Uncertainty parameter over the number of iterations.** Average uncertainty over the complete dataset for both sequential (orange) and online (blue) training approaches, averaged over 30 independent runs. Dashed lines show the average uncertainty achieved using the *single-pass* approach for different values of n. The shaded region represents one standard deviation from the mean.

Algorithm 2 Step1(**Sample Set** S_{i-1}, **int** n, **float** δ, **Distribution** $P_{\mathcal{Z}}$, **Measurement** $\rho()$, **Classifier** $f_{\mathcal{O}}$, **Classifier** $f_{\mathcal{C}}(\theta^*_{i-1})$)

1: $\Xi \leftarrow S_{i-1} \bigcup \{(z_j, f_{\mathcal{O}}(z_j)) \mid z_j \sim P_{\mathcal{Z}}, j = 1 \dots n\}$
2: $S_i \leftarrow \{z | z \in \Xi, \rho(z, \theta^*_{i-1}) \geq \delta\}$ ▷ Filter according to uncertainty
3: **return** S_i

points with a higher uncertainty are also those for which the empirical fidelity error is maximal.

However, removing points from the set S_i breaks the assumptions in Theorems 1 and 2. Figure 3 shows different results for the sequential training with $n = 100$ and different values of the sample selection parameter δ in the *spirals* dataset. For comparison purposes, results for both an online approach and a *vanilla* sequential approach are also shown for the same problem. Figure 3a) shows the average accuracy value at each iteration. As expected, curves corresponding to sequential training with sample selection exhibit much better performance than online training. Yet, they lie far from the vanilla sequential setting.

Figures 3b) and c) show how the ρ measure and the number of synthetic data points used for training change for increasing iterations, respectively. The overall uncertainty remains approximately constant for the online learning setting, while it rapidly approaches 0 for the

pure sequential setting. Contrary to what one might expect, the effect of introducing sample selection is an overall increase in model uncertainty. On the bright side, the sample removal policy leads to a decrease in the number of data points used for training. Notably, in all cases, the number of samples Nn seems to converge to a fixed value after a few iterations.

We can extract the following insights by considering all three plots in Figure 3 at once. During the first iterations, when copy models are still not sufficiently tuned to the data, there is no filtering effect. The number of points Nn grows almost equally for all the settings for a few iterations, as shown in Figure 3c). During this stage, accuracy increases gradually for all models, as shown in Figure 3a), and ρ decreases as the copies are fed more data, as displayed in Figure 3b). This is the expected behavior for a model that is correctly compressing information. The differences appear after a few iterations when the filtering effect starts to kick in and the number of data points decreases for those settings where the sample selection policy is enforced. At this stage, copy models have reached an overall confidence level that is close to δ and this enables the dropping of samples. Accordingly, accuracy stops growing and becomes almost flat. Because only the most uncertain points are retained, the average uncertainty level starts to grow. With regard to the number of points, the curves quickly stabilize and become almost flat by $t = 18$. The model with the lower threshold, $\delta = 10^{-10}$, accumulates the larger number of points (≈ 250). This is less than 10% of the number of points required by the pure sequential method. Even so, this model reaches a reasonable accuracy by the end of the process. In contrast, the accuracy achieved by models trained for uncertainty thresholds of $\delta = 10^{-8}$ and $\delta = 10^{-6}$ is below that of the online setting.

We can summarize these findings as follows. When the copy model reaches high confidence (low uncertainty) over the synthetic dataset, the effect of sample removal is a switch from a sequential (purely incremental addition of points) to an online behavior (the number of points required at each iteration is a constant value). Whenever the models used are parametric, this switch to online learning behavior leads to a well-known pathology in the learning process, *catastrophic forgetting*. This is observed in Figure 3 as sequential models start to resemble the online behavior. Here, the effect of sample removal is an increase in its uncertainty: copy models forget the previous information. To solve this problem, we introduce several improvements to the practical implementation for Step 2 of the alternating optimization algorithm.

3.3 Step 2: Optimizing copy model parameters

Let us briefly turn our attention to Equation 6. This equation models the requirement of controlling the capacity of the copy model while minimizing the loss function. To address this problem, we introduce a capacity-increasing scheme. For this purpose, let us recall that, if we assume the copy model has enough capacity, then for a given iteration i Equation 6 can be simplified as follows

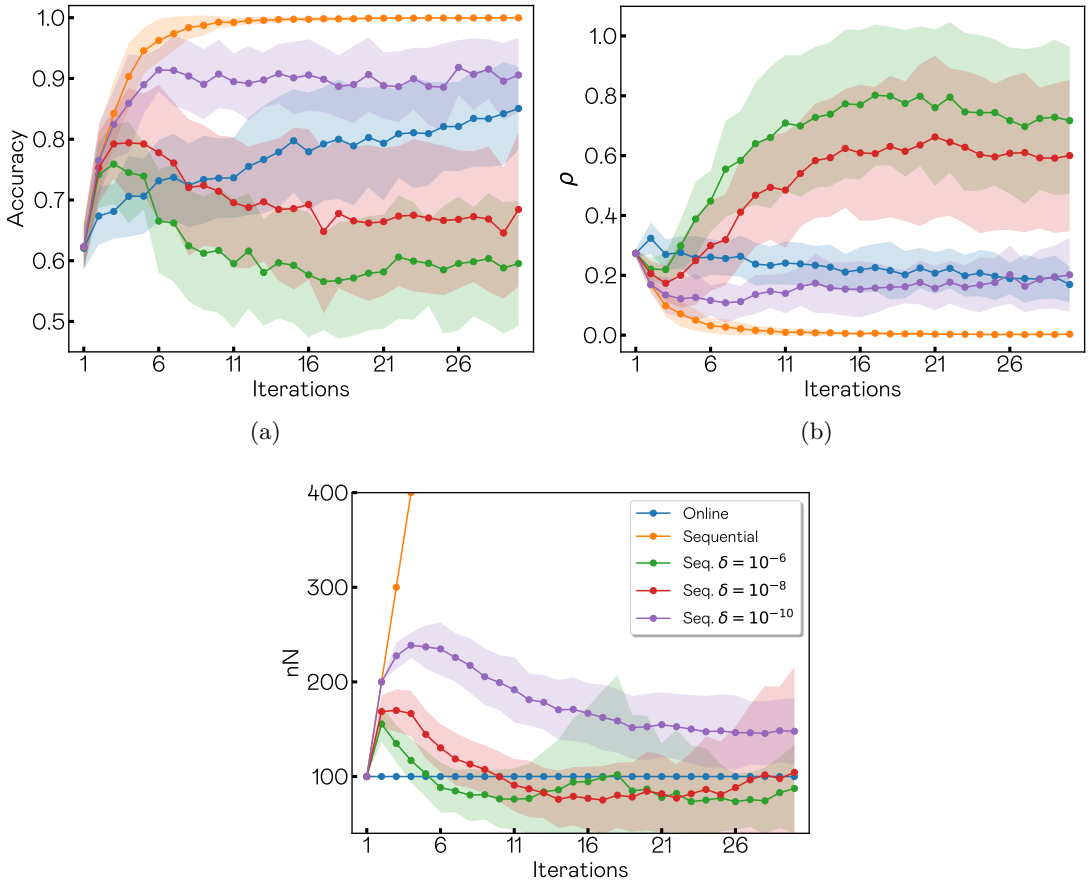


Figure 3: **Sequential dropping model.** a) Copy accuracy averaged over 30 independent runs for sequential, with and without dropping, and online training at each iteration. b) Average uncertainty at each iteration for sequential, with and without dropping, and online training. c) Number of data points used at each iteration. Y-Range of the image was restricted to 400 in order to make curves observable. The number of points of the sequential learning (orange line) grows linearly until $30 \cdot 100 = 3000$

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \Omega(\theta) \\ \text{subject to} \quad & \|R^{\mathcal{F}}_{emp}(f_{\mathcal{C}}(\theta), f_{\mathcal{O}})\| \Big|_{S=S_i^*} < \varepsilon. \end{aligned} \tag{11}$$

Having the above simplification in mind, we summarize our proposed scheme in Algorithm 3. In a nutshell, this algorithm forces the copy to iteratively solve the parameter

Algorithm 3 Step2_prev(**Sample Set** S, **float** ϵ, **float** ε,**Classifier** $f_{\mathcal{O}}$)

1: $k \leftarrow 0$, $\epsilon_k \leftarrow C$, $C >> \varepsilon$
2: **while** $R^{\mathcal{F}}_{emp}(f_{\mathcal{C}}(\theta_k), f_{\mathcal{O}}; \Omega) \geq \varepsilon$ **do**
3: $\quad \theta^*_k \leftarrow \arg\min_{\theta} R^{\mathcal{F}}_{emp}(f_{\mathcal{C}}(\theta), f_{\mathcal{O}}; \Omega)$ subject to $R^{\mathcal{F}}_{emp}(f_{\mathcal{C}}(\theta), f_{\mathcal{O}}; \Omega) \geq \epsilon_k \Big|_{\theta^0_k = \theta^*_{k-1}}$
4: $\quad \epsilon_j \leftarrow \min(\epsilon_{j-1}/2, \varepsilon)$ ▷ Reduce the value of epsilon
5: $\quad$ Increase Ω
6: $\quad$ k = k + 1
7: **end while**
8: **return** θ^*_k

optimization problem in stages, that the empirical risk decrease forcing at each stage. For a given iteration step i, we start with a copy model with a very small capacity Ω. At each sub-iteration k, we solve the optimization problem for an upper bound ϵ_k of the target value ε. For increasing values of k, we increase the capacity and reduce the upper bound. The larger the capacity, the more the empirical risk will be reduced for every sub-iteration. Because we also reduce the upper bound, this ensures that we approach the target value using an approximation with the smallest complexity.

In practice, we train copies using the loss function defined in Equation 10. Because we are using stochastic subgradient optimization-based models[6], we can control model complexity by means of an early stopping criterion. Thus, delaying the early stopping increases the complexity of the boundary learned by the model. To ensure a better convergence behavior in this framework, we update the hyperparameters at each iteration.

3.3.1 Forcing the model to remember

With the former scheme in mind, we recover the line of argument in Section 3.2.3. As stated, the sample removal policy violates some of the assumptions of the theorems introduced in Section 3.2. As a result, a forgetting effect appears in Step 2. In what follows, we make the required adjustments to recover the convergence properties. We begin by highlighting that in the sequential approach, Theorem 2 shows that parameters converge to their optimal value. A direct consequence of this convergence is that the difference between two consecutive sequence terms must also converge to zero, such that

$$\left|\left|\theta^*_{i+1} - \theta^*_i\right|\right| \longrightarrow 0. \tag{12}$$

However, the asymptotic invariant in Equation 12 can be forced to preserve the compressed model obtained from previous iterations, even when filtering the data points. To such end, we complement the loss function at iteration i by adding a regularization term that ensures the minimization of the left term in Equation 12, as follows

$$\mathcal{L} = R^{\mathcal{F}}_{emp}(f_{\mathcal{C}}(\theta), f_{\mathcal{O}}; \Omega) + \lambda \cdot \|\theta - \theta^*_{i-1}\| \tag{13}$$

6. In particular, neural networks with "Stochastic Gradient Descent".

Algorithm 4 Step2(**Sample Set** S, **float** ϵ, **float** ε,**Classifier** $f_{\mathcal{O}}$, **Parameters** θ^*_{i-1})

$j \leftarrow 0$, $\epsilon_j \leftarrow \varepsilon$
while $R^{\mathcal{F}}_{emp}(f_{\mathcal{C}}(\theta_j), f_{\mathcal{O}}; \Omega) \geq \varepsilon$ **do**

$$\theta^*_j \leftarrow \arg\min_{\theta} R^{\mathcal{F}}_{emp}(f_{\mathcal{C}}(\theta), f_{\mathcal{O}}; \Omega) + \lambda \cdot \|\theta - \theta^*_{i-1}\|$$

$$\text{subject to } R^{\mathcal{F}}_{emp}(f_{\mathcal{C}}(\theta), f_{\mathcal{O}}; \Omega) \geq \epsilon_j \Big|_{\theta^0_j = \theta^*_{j-1}}$$

    $\epsilon_i \leftarrow \min(\epsilon_{i-1}/2, \varepsilon)$ ▷ Reduce the value of epsilon
    Increase Ω
end while
return θ^*_j

Observe that this regularization term comes naturally from the derived theorems. A similar strategy can be found in the literature under the name of Elastic Weight Consolidation (EWC), albeit heuristically derived from a completely different set of assumptions (Kirkpatrick et al., 2017). Algorithm 4 describes how to implement this strategy.

Continuing with the spirals example, Figure 4 shows the experimental results obtained when conducting the sequential copying process for four different λ values and a threshold $\delta = 10^{-8}$ (see also Figure 3 for comparison). As before, Figure 4a) reports the change in accuracy for increasing iterations, while Figures 4b) and c) present the results for the uncertainty ρ and the number of data points used nN. Again, we can combine the information from all three panels to better understand the behavior displayed by the copy model when *forced to remember*. Initially, the model has not been trained, so the number of points increases while the uncertainty decreases. Once the removal threshold δ is reached, the regularization term becomes relevant. For low λ values, the ρ-term of Equation 12 dominates the cost function, and the optimization process learns the new data points. As a result, the copy parameters adapt to new data points and the model exhibits forgetting. Conversely, for*large* λ values, the regularization term will dominate the optimization. The model is therefore forced to retain more data points to ensure the ρ term can compete with the regularization term. In this setting, the number of data points still grows, but in a sub-linear way. The most desirable behavior is a constant number of points during the training process. For this particular example, we observe this behavior for $\lambda = 0.05$ and $\lambda = 0.1$.

3.3.2 Automatic Lambda

In the previous section, we observed an increment in the number of data points when using large lambda values, to compensate for the memorization effect introduced by the regularization term. This effect is desirable to consolidate the previous knowledge acquired by the model. On the contrary, small lambda values promote short-lived data compression. This is desirable at the beginning of the learning process or when the data distribution

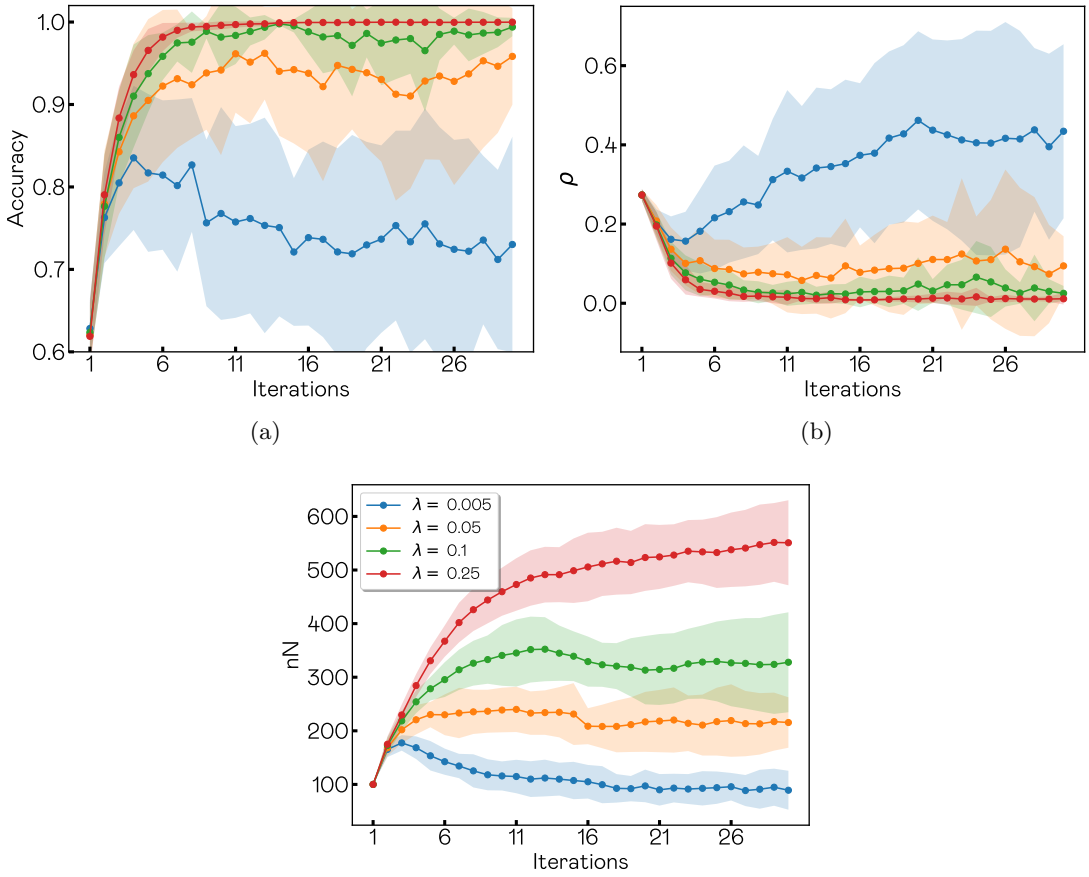


Figure 4: **Fixed lambda results.** a) Copy accuracy averaged over 30 independent runs for four different lambda values: $\lambda = 0.005, 0.05, 0.1, 0.25$ with the same drooping threshold $\delta = 10^{-8}$. b) Average uncertainty at each iteration. c) Number of data points used at each iteration. Shaded region shows $\pm\sigma$

suffers a shift. To retain the best of both regimes, we propose a heuristic to dynamically adapt the λ value to the needs of the learning process. Our underlying intuition is that, whenever the amount of data required increases, the memory term prevents the copy from adapting to new data. This signals that the value of λ must decrease. Equivalently, when we observe a decrement in the number of data points, this means that the model is able to classify most of them correctly. This indicates that we must stabilize it to avoid unnecessary model drift due to future disturbances in the future data. Hence, we must increase the λ parameter.

Thus, considering the definition of S_i in lines 4 and 5 of Algorithm 5, we force the described behavior by automatically updating the value of λ parameter as follows

Algorithm 5 Alternating Optimization Sequential Copy (**int** T, **int** n, **float** ε, **Classifier** $f_{\mathcal{O}}$)

$S_0 \leftarrow \{(z_j, f_{\mathcal{O}}(z_j)) \mid z_j \sim P_{\mathcal{Z}}, j = 1 \dots n\}$
$\theta_0^* \leftarrow \arg\min_{\theta} \frac{1}{|S_0|} \sum_{z \in S_0} \rho^2(z, \theta)$
for $i = 1$ to T **do**
    $S_i^* \leftarrow \text{Step1}(S_{i-1}^*, n, \delta, P_{\mathcal{Z}}, \rho(), f_{\mathcal{O}}, f_{\mathcal{C}}(\theta_{i-1}^*))$ ▷ Algorithm 2
    $\lambda \leftarrow \begin{cases} \lambda/2 & \text{if} \quad |S_i| \geq |S_{i-1}|, \\ 1.5 \cdot \lambda & \text{otherwise.} \end{cases}$ ▷ Adaptive lambda
    $\theta_i^* \leftarrow \text{Step2}(S_i, \varepsilon, f_{\mathcal{O}}, \theta_{i-1}^*)$ ▷ Algorithm 4
end for
return θ_i^*

$$\lambda = \begin{cases} \lambda/2 & \text{if} \quad |S_i| \geq |S_{i-1}|, \\ 1.5 \cdot \lambda & \text{otherwise.} \end{cases}$$

The complete optimization process, including the improvement above, is described in Algorithm 5. This is the final algorithmic proposal of this article. In our implementation, data trends are modeled using the difference between the number of points at the previous iteration with the number of points after the filtering process. Data filtering is applied at the start of every iteration, after generating a new set of n samples.

As before, we show how this improvement works for the *spirals* example. We repeat the experiments using Algorithm 5 with $n = 100$ and three different dropping thresholds: $\delta = 10^{-6}, 10^{-8}$ and 10^{-10} as we did in Subsection 3.2.3 where the dropping procedure was first introduced. Recall that, as discussed in Figure 3 only the setting corresponding to the smallest δ value managed to perform better than the online approach, yet still performed worse than the pure sequential implementation. The remaining thresholds lost too many data points, and their performance was worse than the online approach when the number of data points used was similar. The results obtained when implementing the full Algorithm 5 are depicted in Figure 5. The accuracy level obtained using the automatic regularization term is comparable with the *pure sequential* case, where the model keeps all data points. Even for the most conservative approach, where we use a very small dropping threshold, the number of points used after 30 iterations is 1/3 smaller than those required for the pure sequential setting. Moreover, even when deliberately forcing a very volatile setup by using a large value of delta, $\delta = 10^{-6}$, the obtained results exhibit an accuracy larger than 0.95 for an almost constant number of data points equal to 200.

4. Experiments

In what follows, we present our experimental findings when implementing the sequential approach to copying over a heterogeneous set of problems. We discuss our results in terms of different performance metrics and compare them to those obtained using the single-pass approach. We begin by describing the data and the experimental setup.

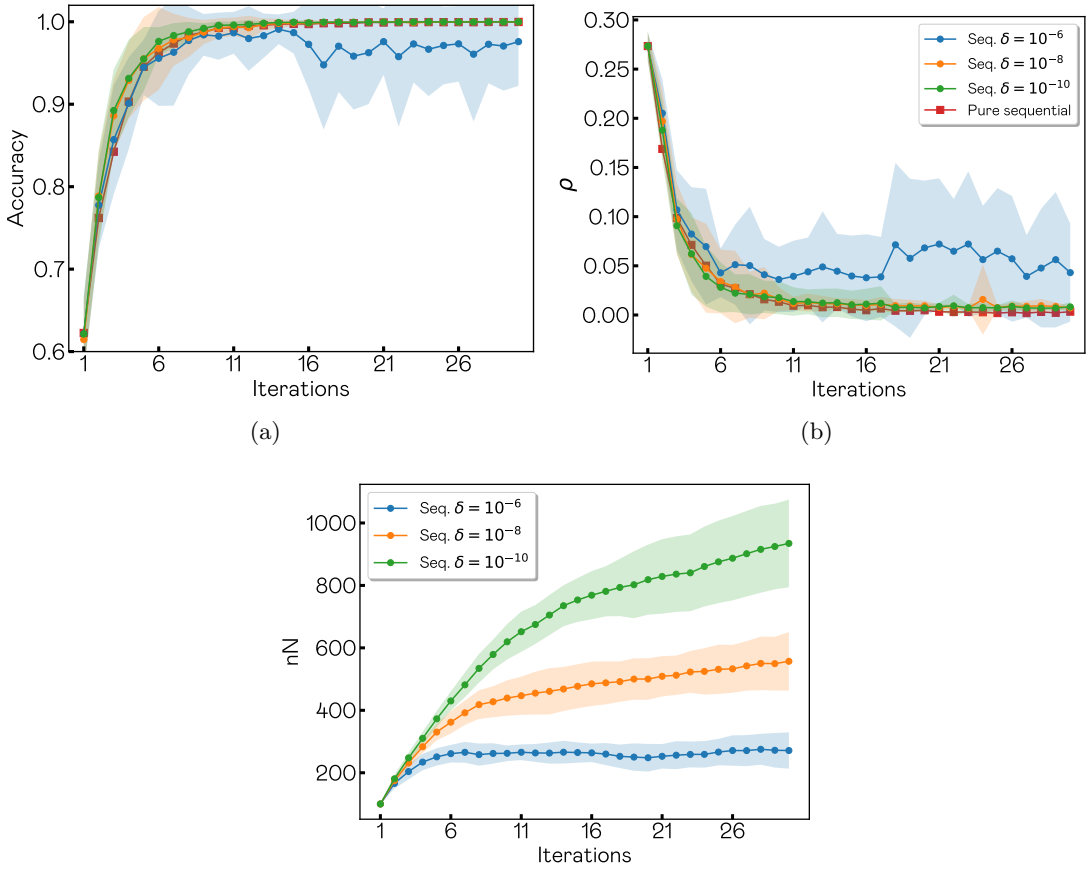


Figure 5: **Automatic lambda results.** a) Copy accuracy averaged over 30 independent runs for three different drooping thresholds: 10^{-6}, 10^{-8}, and 10^{-10}. b) Average uncertainty at each iteration. c) Number of data points used at each iteration. Shaded region shows $\pm\sigma$

4.1 Experimental settings

To assess the validity of the proposed method, we use 60 datasets from the UCI Machine Learning Repository database (Dheeru and Karra Taniskidou, 2017). We follow the experimental methodology in (Unceta et al., 2020a) and refer the reader to this article for a more in-depth description of how the problems have been selected and prepared for analysis.

In all cases, we convert nominal attributes to numerical and re-scale variables to zero mean and unit variance. We split the pre-processed data into stratified 80/20 training and test sets. We sort the datasets in alphabetical order and group them in sets of 10. For all datasets in each group, we randomly assign one of the following classifiers: AdaBoost (*adaboost*), artificial neural networks (*ann*), random forest (*rfc*), linear SVM (*linear_svm*), radial basis kernel SVM (*rbf_svm*) and gradient-boosted trees (*xgboost*). We train all models

using a 3-fold cross-validation grid search over a fixed parameter grid. A full description of the 60 datasets, including general data attributes, the assigned classifier, and the achieved accuracy, can be found in Table 2.

We copy each of the resulting classifiers using fully-connected neural networks with three hidden layers, each consisting of 64, 32, and 10 neurons, followed by a SoftMax output layer. We use *ReLu* activations and no dropout. We use no pretraining and random weights initialization. We train these models sequentially through 30 iterations. At each iteration t, we generate $n = 100$ new data points by randomly sampling from a standard normal distribution $\mathcal{N}(0, 1)$. To ensure a faster convergence, we drop all the data points for which the instantaneous copy model yields an uncertainty ρ below a defined threshold ϵ. We use the remaining data to adjust the weights at each iteration with *Adam* optimizer and a learning rate equal to $5e-4$. For each value of t, we use 1000 epochs and balanced batches consisting of 32 data points.

We use the previously defined normalized uncertainty average as a loss function and assay different threshold values for the ρ parameter to evaluate its impact on the learning process. In particular, we run independent trials for $\epsilon \in \{5e-4, 1e-4, 5e-5, 1e-5, 5e-6, 1e-6, 5e-7, 1e-7, 5e-8, 1e-8, 1e-9, 1e-10\}$. In addition, we also allow an automatic update of the λ parameter, starting from a value of 0.5.

We run all experiments on a server with 28 dual-core AMD EPYC 7453 at 2.75 GHz, with 768 Gb RDIMM/3200 of RAM, running on Linux 5.4.0. We use Python version 3.10.4 and train copies on TensorFlow 2.8. For validation purposes, we run each experiment 30 times and report averages over all repetitions.

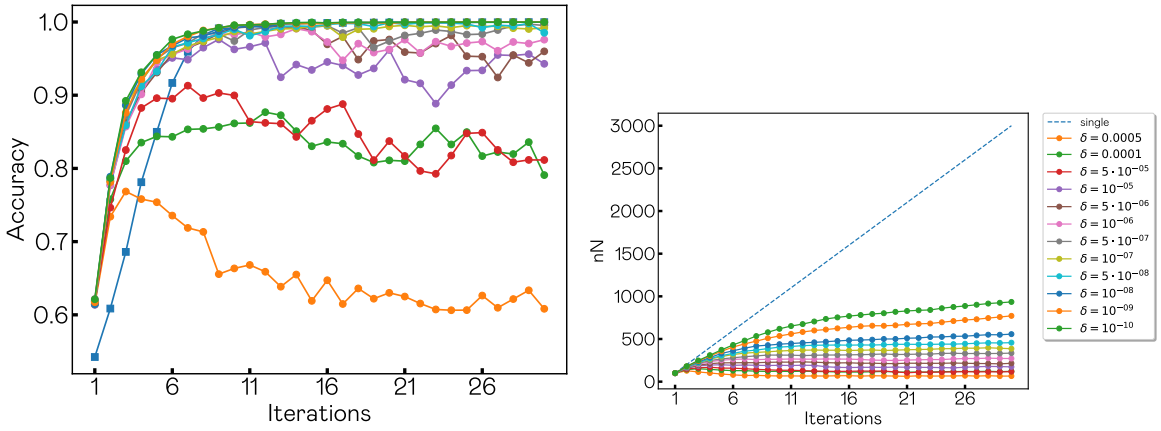
4.2 Metrics and performance plots

We evaluate copies trained using the sequential approach based on three performance metrics. We report these metrics for all datasets simultaneously using different performance plots. Thus, we demonstrate the differences between the results obtained using the single-pass approach and the sequential approach for several parameter settings.

Following (Unceta et al., 2020a) we report the copy accuracy and empirical fidelity error. The former corresponds to the accuracy achieved by the copy on the original data space, while the latter refers to the accuracy over a test synthetic set. Note that computing the copy accuracy requires that we assume the original test data to be known and accessible. We consider this assumption to be reasonable in the current development stage and therefore report this metric as a performance check. We use both metrics to compare the results obtained when using the sequential approach with those obtained when training copies based on the single-pass approach.

Figure 6 shows the average copy accuracy and number of data points for copies trained using the single-pass (blue) and the sequential approach over the *spirals* problem discussed above. Note that for the sequential approach, we report our results for different experimental settings, corresponding to different values of the removal threshold δ. Accuracy results are better for small thresholds. Analyzing the different curves, we observe three disjoint groups, thresholds smaller than $1e-5$, with high accuracy, thresholds between $1e-4$ and $5e-5$, with some accuracy degradation, and finally, the threshold equal to $5e-4$, which achieves a low performance. This difference also exists with the number of points. However, it is not

as steep as in the accuracy plot. Regarding the number of data points, we want to highlight the big difference between the single-pas and the tested sequential approaches.



(a) (b)

Figure 6: **Spirals dataset.** a) Copy accuracy averaged over 30 independent runs for 12 different values of the removal threshold δ b) Number of data points used at each iteration.

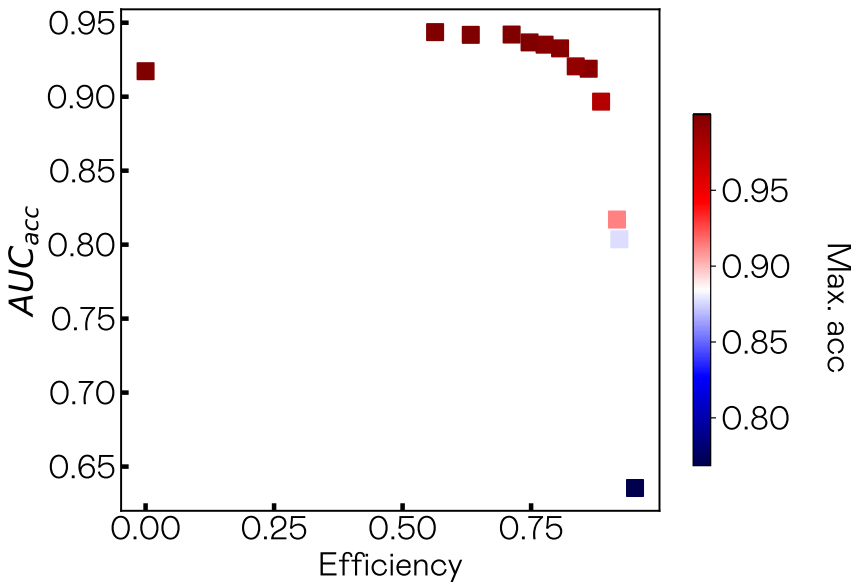
In addition, we also compute the Area Under the Curve (AUC) for both the empirical fidelity error and the memory efficiency. We refer to the former as AUC_{acc}. We compute this value in terms of the accuracy over a distinct set of synthetic data used for testing purposes. This is a measure that combines information on the actual performance of the copy and the convergence speed of the sequential approach. The higher the value of this metric, the quicker the copies converge to a higher accuracy result. We compute memory efficiency concerning the results obtained using the single pass approach for the different values of n, i.e., at each iteration t. We assume single-pass has no memory efficiency and introduce AUC_{nN} as the Area Under the Curve for the number of training data points per iteration for the sequential approach. AUC_{nN} can be interpreted as the total amount of samples used during the convergence process. The difference between this quantity and the single-pass value is a measurement related to memory usage. Specifically, we define the memory efficiency for the sequential approach as

$$\text{Efficiency} = 1 - \frac{AUC_{nN}}{AUC_{single}}$$

.

To compactly evaluate the proposed algorithm's overall performance, the former two metrics are combined in a single plot by proceeding as follows. For each dataset, we plot all the models resulting from the different parameter settings for which the obtained accuracy A_C is such that $A_C > A_C^{single} * 0.95$. That is all the models for which copy accuracy is within a 5% of the equivalent single-pass model. Figure 7 shows such a plot for the *spirals* toy dataset. Each sample in this plot corresponds to a different parameter setting, while the colors indicate the corresponding accuracy. This plot shows the trade-off between accuracy

convergence and memory efficiency for different settings. In the particular case of the plot in Figure 7 we can observe that sequential models can achieve large accuracy values with a high convergence rate, as well as efficiencies between 50% to 80% without a significative drop in the accuracy score.



(a) Summary plot

Figure 7: **Summary plot for spirals** Area under the accuracy curve as a function of the memory efficiency. Colormap represents the accuracy achieved by each element point.

As a final summarization plot, once having identified these two models, we compare them by plotting them together and reporting their differences. Ideally, we would expect the most accurate model to also be the most efficient. While this is not always the case, in general, the difference in accuracy between the most accurate and the most efficient instances is negligible.

In what follows, we report the results obtained for 60 UCI datasets in terms of the metrics and performance plots discussed above.

4.3 Results

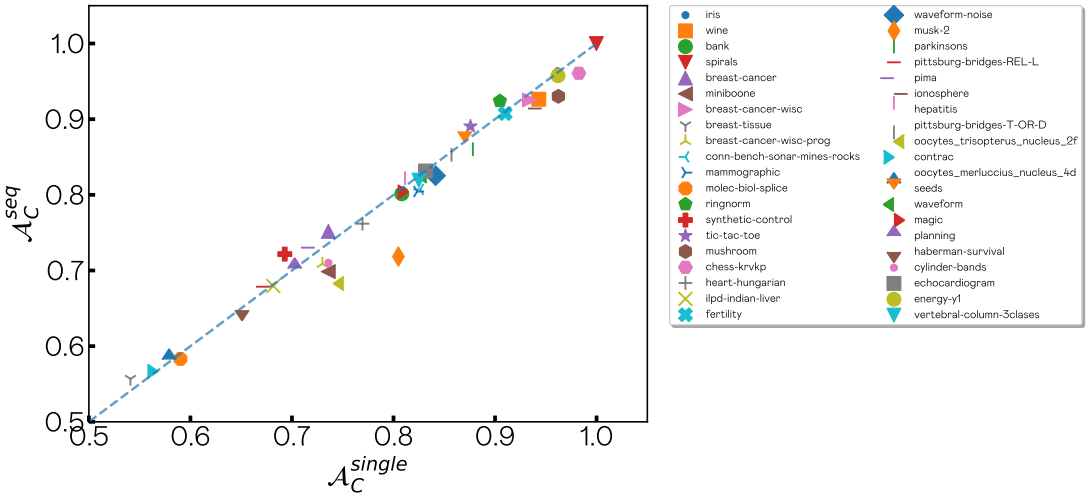
Note that the diagonal line represents those cases for which both approaches yield an equal accuracy.

Figure 8 shows the overall results obtained for all the different datasets. Figure 9 a shows the comparison between average accuracies for the best performing copies based on the sequential approach and the corresponding single-pass results for copies trained using the same number of data points. Values plotted in the diagonal correspond to cases where both approaches yield comparable results. We can observe that, in most cases, copies based

on sequential approach recover most of the single pass accuracy despite having being trained on much smaller synthetic datasets.

There are some cases where results for sequential copies fall below the diagonal. These correspond to situations where the iterative models fail to reach the optimal accuracy. It happens when the amount of memory (N) is not enough to describe the decision boundary entirely.

Finally, for some datasets, we observe that the copy accuracy obtained using the sequential approach improves over the single-pass results. These cases are those laying above the diagonal. It happens when the single-pass uses so many points that it cannot converge to an optimal solution when training.



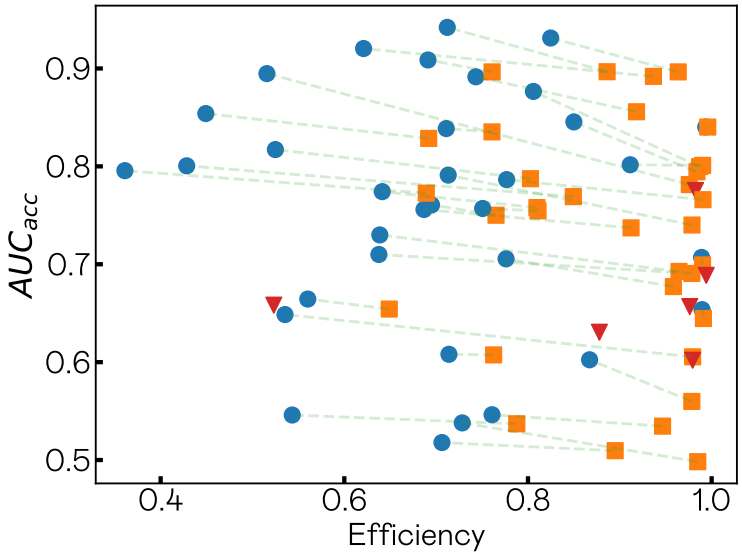
(a)

Figure 8: **UCI datasets' results** Accuracy comparison between sequential approach and singlepass. Points over the dashed line exhibit the same accuracy using both methods whereas points above (below) the line are points where sequential (single-pass) method gives a better performance.

Figure 9 depicts the Accuracy and efficiency relationship of all the UCI databases, together with the linear fitting of all the intermediate solutions. Observing the results, we see five databases with no difference between the most efficient sequential copy and the highest accuracy. It means the task is relatively simple and can be solved using limited memory. For most databases, we observe a high increment in efficiency without suffering degradation in terms of accuracy. These databases correspond to those with large green dashed lines with almost no slope.

5. Conclusions

In this paper, we have introduced the sequential approach. To this aim, we have translated Equation 2 into a probabilistic setting. Then, we have introduced two theorems in Subsection 3.2 to prove the sequential copy process converges in parameters and behavior to



(a) Max. accuracy Vs. most efficient: UCI datasets

Figure 9: **UCI datasets' results** Highest accuracy (blue points) and most efficient (orange squares) results for each dataset. Green-dashed segments connect both points for the same dataset showing a linear fitting of all the intermediate solutions. Red triangles represent datasets where the most efficient point is also the one with highest accuracy.

the single-pass when the number of samples used for the copying process tends to infinity. From that point, Subsection 3.2.2 studied the duality compression-memorization of the copy model. In that section, we sketched the idea that a perfect copy can compress all data in the model parameters. To measure this effect, we argued that epistemic uncertainty is a reliable data compression measure for copying. This statement is only valid for copies and can not be extrapolated to standard learning procedures because contrary to the standard learning case, there is no aleatoric uncertainty when copying. Therefore, all uncertainty measured corresponds to that coming from the model itself. As such, we can devise copy models that effectively compress all the data with guarantees. With this in mind, in Subsection 3.3.1, we have spotted and defined the phenomenon of catastrophic forgetting in copies, a well-known effect that appears in online learning processes. To mitigate this effect, we introduced a regularization term derived from an invariant in Theorem 2. As a result, the sequential approach became more stable.

Then, to reduce the computational time and memory used, a filtering process was introduced in Subsection 3.2.3. This filtering process controls the compression level that each data sample undertakes. As such, if new data points are already well compressed and represented by the model when training, it is unnecessary to feed them into the learning process. As a result of this process, very little data is required during the learning step. We saw that the required data during the learning process stabilizes to a certain amount.

Then, an automatic adjustment policy was introduced in Subsection 3.3.2 to control the hyper-parameter that governs the regularization term. This policy resorts to a simple but stable meta-learning algorithm that allows us to weight the dynamic adjustment of the regularization term. As a result, there is no need for hyperparameter tuning.

Once we developed all the necessary theories, we experimented with 60 databases from several domains and six machine learning models. Our results showed that it is possible to create a copy using the sequential approach with the same accuracy as the single-pass approach. Besides, we introduced some measures to study the performance improvement of the sequential approach. With these measures, we described how it is possible to choose the most suitable copying configuration for the available computing resources in terms of memory and execution time.

Acknowledgments

This work was funded by Huawei Technologies Duesseldorf GmbH under project TC20210924032.

References

Hessam Bagherinezhad, Maxwell Horton, Mohammad Rastegari, and Ali Farhadi. Label refinery: Improving imagenet classification through label progression. arXiv:1805.02641, 2018. [Online] Available from: https://arxiv.org/abs/1805.02641.

Mariam Barque, Simon Martin, Jérémie Etienne Norbert Vianin, Dominique Genoud, and David Wannier. Improving wind power prediction with retraining machine learning algorithms. In *Proceedings of the International Workshop on Big Data and Information Security, Jakarta, Indonesia*, pages 43–48, Jakarta, Indonesia, 2018. IEEE Computer Society, Washington, DC, USA.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 535–541, Philadelphia, PA, USA, 2006a. Association for Computing Machinery, New York, NY, USA. ISBN 1595933395. URL https://doi.org/10.1145/1150402.1150464. doi: 10.1145/1150402.1150464.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 535–541, Philadelphia, PA, USA, 2006b. Association for Computing Machinery, New York, NY, USA. ISBN 1595933395. URL https://doi.org/10.1145/1150402.1150464. doi: 10.1145/1150402.1150464.

Tao Chen. All versus one: An empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software. In *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 157–168, 2019. doi: 10.1109/SEAMS.2019.00029.

Terrance DeVries and Graham W Taylor. Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865*, 2018.

Dua Dheeru and Efi Karra Taniskidou. UCI Machine Learning Repository, 2017.

Javad Feizabadi. Machine learning demand forecasting and supply chain performance. *International Journal of Logistics Research and Applications*, 25(2):119–142, 2022. doi: 10.1080/13675567.2020.1803246. URL https://doi.org/10.1080/13675567.2020.1803246.

Zhabiz Gharibshah and Xingquan Zhu. User response prediction in online advertising. *ACM Comput. Surv.*, 54(3), may 2021. doi: 10.1145/3446662. URL https://doi.org/10.1145/3446662.

Jon Ander Gómez, Juan Arévalo, Roberto Paredes, and Jordi Nin. End-to-end neural network architecture for fraud scoring in card payments. *Pattern Recognition Letters*, 105: 175–181, 2018. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2017.08.024. URL https://www.sciencedirect.com/science/article/pii/S016786551730291X. Machine Learning and Applications in Artificial Intelligence.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015a.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015b.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017. doi: 10.1073/pnas.1611835114. URL `https://doi.org/10.1073/pnas.1611835114`.

Ruishan Liu, Nicolo Fusi, and Lester Mackey. Teacher-student compression with generative adversarial networks. arXiv:1812.02271, 2018. [Online] Available from: `https://arxiv.org/pdf/1812.02271.pdf`.

Jose Mena, Axel Brando, Oriol Pujol, and Jordi Vitrià. Uncertainty estimation for black-box classification models: a use case for sentiment analysis. In *Pattern Recognition and Image Analysis. IbPRIA 2019. Lecture Notes in Computer Science*, volume 11867, pages 29–40, Madrid, Spain, 2019. ISBN 978-3-030-31331-9. doi: `10.1007/978-3-030-31332-6_3`.

Jose Mena, Oriol Pujol, and Jordi Vitrià. Uncertainty-based rejection wrappers for black-box classifiers. *IEEE Access*, 8:101721–101746, 2020. doi: `10.1109/ACCESS.2020.2996495`.

Rafael Müller, Simon Kornblith, and Geoffrey Hinton. When does label smoothing help? In *Proceedings of the 33rd Conference on Neural Information Processing Systems*, Vancouver, Canada, 2019. MIT Press, Cambridge, MA, USA.

Vu-Linh Nguyen, Sébastien Destercke, Marie-Hélène Masson, and Eyke Hüllermeier. Reliable multi-class classification based on pairwise epistemic and aleatoric uncertainty. In *27th International Joint Conference on Artificial Intelligence (IJCAI 2018)*, pages 5089–5095, 2018.

Andrei Paleyes, Raoul-Gabriel Urma, and Neil D. Lawrence. Challenges in deploying machine learning: A survey of case studies. *ACM Computing Surveys*, apr 2022. ISSN 0360-0300. doi: 10.1145/3533378. URL `https://doi.org/10.1145/3533378`.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions in Knowledge Data Engineering*, 22:1345–1359, 2010.

Robin Senge, Stefan Bösner, Krzysztof Dembczyński, Jörg Haasenritter, Oliver Hirsch, Norbert Donner-Banzhoff, and Eyke Hüllermeier. Reliable classification: Learning classifiers that distinguish aleatoric and epistemic uncertainty. *Information Sciences*, 255: 16–29, 2014. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2013.07.030. URL `https://www.sciencedirect.com/science/article/pii/S0020025513005410`.

Mohammad Shehab, Laith Abualigah, Qusai Shambour, Muhannad A. Abu-Hashem, Mohd Khaled Yousef Shambour, Ahmed Izzat Alsalibi, and Amir H. Gandomi. Machine learning in medical applications: A review of state-of-the-art methods. *Computers in Biology and Medicine*, 145:105458, 2022. ISSN 0010-4825. doi: https://doi.org/10.1016/j.compbiomed.2022.105458. URL https://www.sciencedirect.com/science/article/pii/S0010482522002505.

Xiaomu Song, Guoliang Fan, and Mahesh Rao. SVM-Based data editing for enhanced one-class classification of remotely sensed imagery. *IEEE Geoscience Remote Sensors Letters*, 5:189–193, 2008.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, Las Vegas, NV, USA, 2016. IEEE Computer Society, Washington, DC, USA. doi: 10.1109/CVPR.2016.308.

Irene Unceta, Jordi Nin, and Oriol Pujol. Copying machine learning classifiers. *IEEE Access*, 8(11):160268–160284, 2020a. doi = 10.1109/ACCESS.2020.3020638.

Irene Unceta, Jordi Nin, and Oriol Pujol. Environmental adaptation and differential replication in machine learning. *Entropy*, 22(10), 2020b. doi = 10.3390/e22101122.

Irene Unceta, Diego Palacios, Jordi Nin, and Oriol Pujol. Sampling unknown decision functions to build classifier copies. In *Proceedings of 17th International Conference on Modeling Decisions for Artificial Intelligence*, pages 192–204, Sant Cugat, Spain, 2020c. doi: 10.1007/978-3-030-57524-3_16.

Dongdong Wang, Yandong Li, Liqiang Wang, and Boqing Gong. Neural networks are more productive teachers than human raters:active mixup for data-efficient knowledge distillation from a blackbox model. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1498–1507, Seattle, WV, USA, 2020. IEEE Computer Society, Washington, DC, USA.

Yinjun Wu, Edgar Dobriban, and Susan Davidson. DeltaGrad: Rapid retraining of machine learning models. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10355–10366. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/wu20b.html.

Chenglin Yang, Lingxi Xie, Siyuan Qiao, and Alan L. Yuille. Training deep neural networks in generations: A more tolerant teacher educates better students. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):5628–5635, Jul. 2019. doi: 10.1609/aaai.v33i01.33015628. URL https://ojs.aaai.org/index.php/AAAI/article/view/4506.

Li Yuan, Francis E.H. Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisiting knowledge distillation via label smoothing regularization. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. IEEE Computer Society, Washington, DC, USA, 2020.

Xinchuan Zeng and Tony R. Martinez. Using a Neural Network to Approximate an Ensemble of Classifiers. *Neural Procesing Letters*, 12:225–237, 2000.

A. Proof of convergence

Hence, in what follows we show that solving the optimization problem in Eq. **??** equals solving the same problem in Eq. **??** in the limit where i approaches infinity. To do this, we first prove that the sequence of functions F_i converges to $F(\theta)$ for increasing values of i, *i.e.* that $\lim_{i\to\infty} F_i(\theta) = F(\theta)$. Then, we also prove that from this result it derives that the sequence of θ_i^* converges to θ^* as i approaches infinity. We do so under several assumptions.

Uniform convergence for F_i

Let us introduce the following proposition on the uniform convergence of functions.

Proposition 3 *A sequence of functions $f_i : D \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}$ is uniformly convergent to a limit function $f : D \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}$, if and only if*

$$||f - f_i||_\infty \xrightarrow{i\to\infty} 0$$

where $||\cdot||_\infty$ denotes the supremum norm of the functions $f_i - f$ on D [7].

On this basis, let's prove Theorem 1 from main text:
Theorem 1 Let $S_i \subseteq S_{i+1} \subseteq \cdots \subseteq S$ be a subsets' convergent sequence. Then, a sequence of functions $\{F_i\}_i$ defined as $F_i(\theta) = \sum_{z\in S_i} \mathcal{P}(\theta|f_{\mathcal{C}}(z,\theta), f_{\mathcal{O}}(z))$, uniformly converges to $F(\theta) = \sum_{z\in S} \mathcal{P}(\theta|f_{\mathcal{C}}(z,\theta), f_{\mathcal{O}}(z))$.
Proof Let us start by taking the supremum norm

$$\begin{aligned}
||F(\theta) - F_i(\theta)||_\infty &= \sup_{\theta\in\Theta}\Big\{\Big|F(\theta) - F_i(\theta)\Big|\Big\} \\
&= \sup_{\theta\in\Theta}\Big\{\Big|\sum_{z\in S}\mathcal{P}(\theta|fc(z,\theta), f_{\mathcal{O}}(z)) - \sum_{z\in S_i}\mathcal{P}(\theta|fc(z,\theta), f_{\mathcal{O}}(z))\Big|\Big\} \\
&= \sup_{\theta\in\Theta}\Big\{\Big|\sum_{z\in S\setminus S_i}\mathcal{P}(\theta|fc(z,\theta), f_{\mathcal{O}}(z))\Big|\Big\}
\end{aligned}$$

Since $\mathcal{P}(\theta|fc(z,\theta), f_{\mathcal{O}}(z))$ is a probability function, it holds that $0 \leq \mathcal{P}(\theta|fc(z,\theta), f_{\mathcal{O}}(z)) \leq 1$. Hence, the norm is bounded by the equation below

$$\begin{aligned}
||F(\theta) - F_i(\theta)||_\infty &= \sup_{\theta\in\Theta}\Big\{\Big|\sum_{z\in S\setminus S_i}\mathcal{P}(\theta|fc(z,\theta), f_{\mathcal{O}}(z))\Big|\Big\} \\
&\leq \sup_{\theta\in\Theta}\Big\{\sum_{z\in S\setminus S_i}\Big|\mathcal{P}(\theta|fc(z,\theta), f_{\mathcal{O}}(z))\Big|\Big\} \leq \Big|S\setminus S_i\Big|
\end{aligned}$$

for $\Big|S\setminus S_i\Big|$ the cardinality of the set $S\setminus S_i$. As discussed before, by definition S_i converges to S for large values of i. According to the proposition above, therefore, we can prove that

$$||F(\theta) - F_i(\theta)||_\infty \xrightarrow{i\to\infty} 0$$

7. https://www.bookofproofs.org/branches/supremum-norm-and-uniform-convergence/proof/

From this proof, it follows that when i approaches infinity the function $F_i(\theta)$ uniformly converges to $F(\theta)$

$$||F(\theta) - F_i(\theta)||_\infty \xrightarrow{i\to\infty} 0 \implies F_i(\theta) \rightrightarrows F(\theta) \tag{14}$$

■

As a consequence of this uniform convergence, two properties of the function F naturally arise. Firstly, $F_i(\theta)$ converges point-wise to $F(\theta)$. Secondly, $F(\theta)$ is a continuous function on Θ.

Parameter convergence

Theorem 2 Under the conditions of Theorem 1, a sequence of parameters $\left\{\theta_i^*\right\}_i$ defined as $\theta_i^* = \arg\max_{\theta\in\Theta} F_i(\theta)$, converges to $\theta^* = \arg\max_{\theta\in\Theta} F(\theta)$, where Θ is the complete set of parameter .

Proof As previously introduced, we can define the optimal copy parameters for a given value of i, and its corresponding function F_i, according to the following equation

$$\theta_i^* = \arg\max_{\theta\in\Theta} F_i(\theta) \tag{15}$$

for Θ the complete parameter set. We assume that this set is only *well defined* if F_i and F have a unique global maximum. This is a commonly made assumption in the literature. In addition, we also assume that Θ is compact.

From the definition of θ_i^* above it follows that

$$F_i(\theta_i^*) \geq F_i(\theta), \ \ \forall\theta \in \Theta, \ \forall i \in \mathbb{N}$$

and using the point-wise convergence of F_i we obtain

$$F(\hat{\theta}^*) \geq F(\theta), \ \ \forall\theta \in \Theta \tag{16}$$

where $\hat{\theta}^*$ is the limit of the sequence $\left(\theta_i^*\right)_i$. As a consequence of the compactness of Θ and the continuity of F, we can conclude that $\hat{\theta}^*$ must exist. Moreover, given our assumption that Θ is *well defined*, we can conclude that

$$\hat{\theta}^* = \theta^* = \arg\max_{\theta} F(\theta). \tag{17}$$

■

The proof above shows that we can approximate the true optimal parameters by sequentially estimating their value using a sequence of subsets S_i that uniformly converge to S. In other words, it demonstrates the feasibility of the sequential approach. This is a theoretical feasibility.

B. Toy problems instantaneous decision boundaries

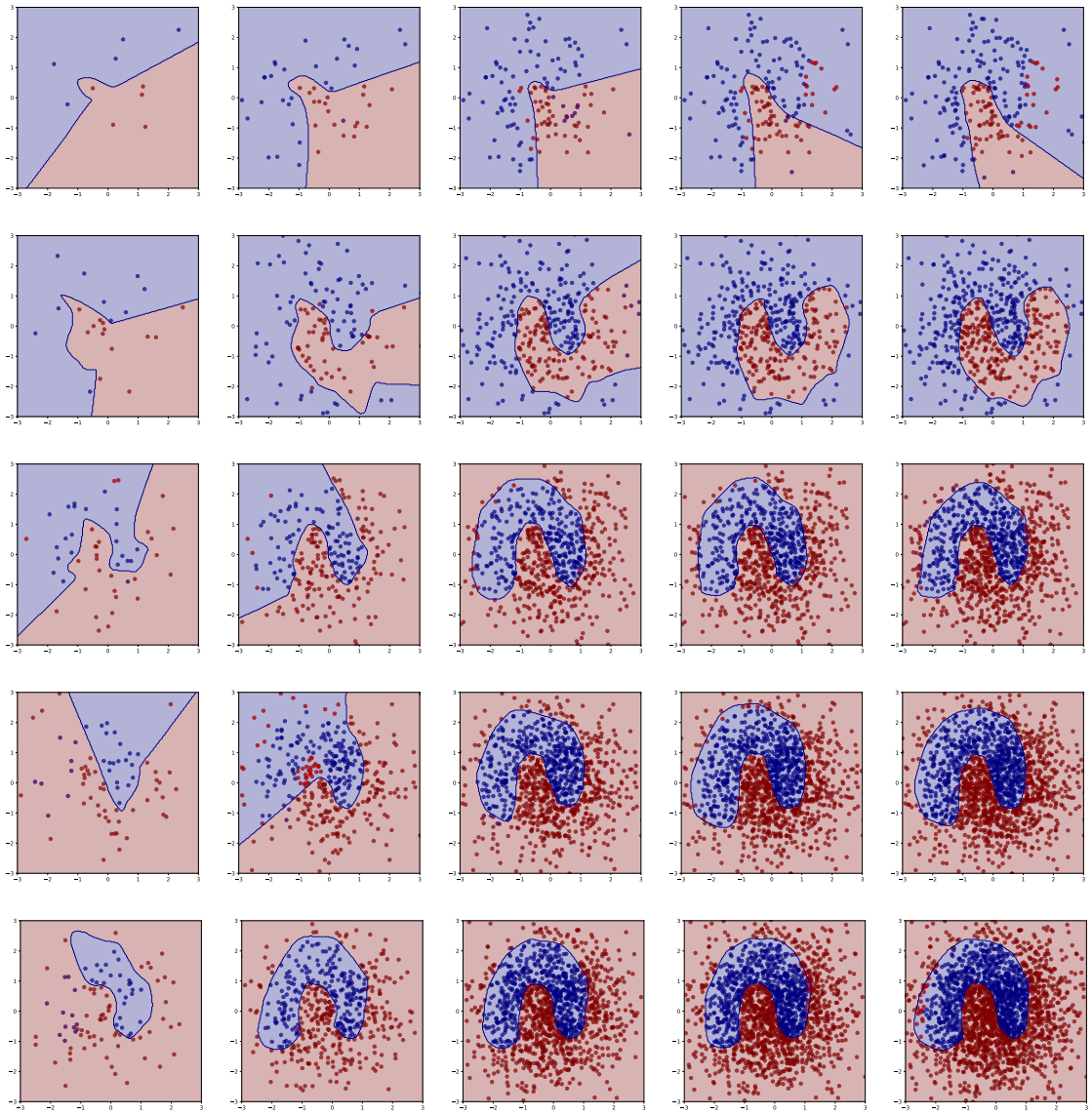
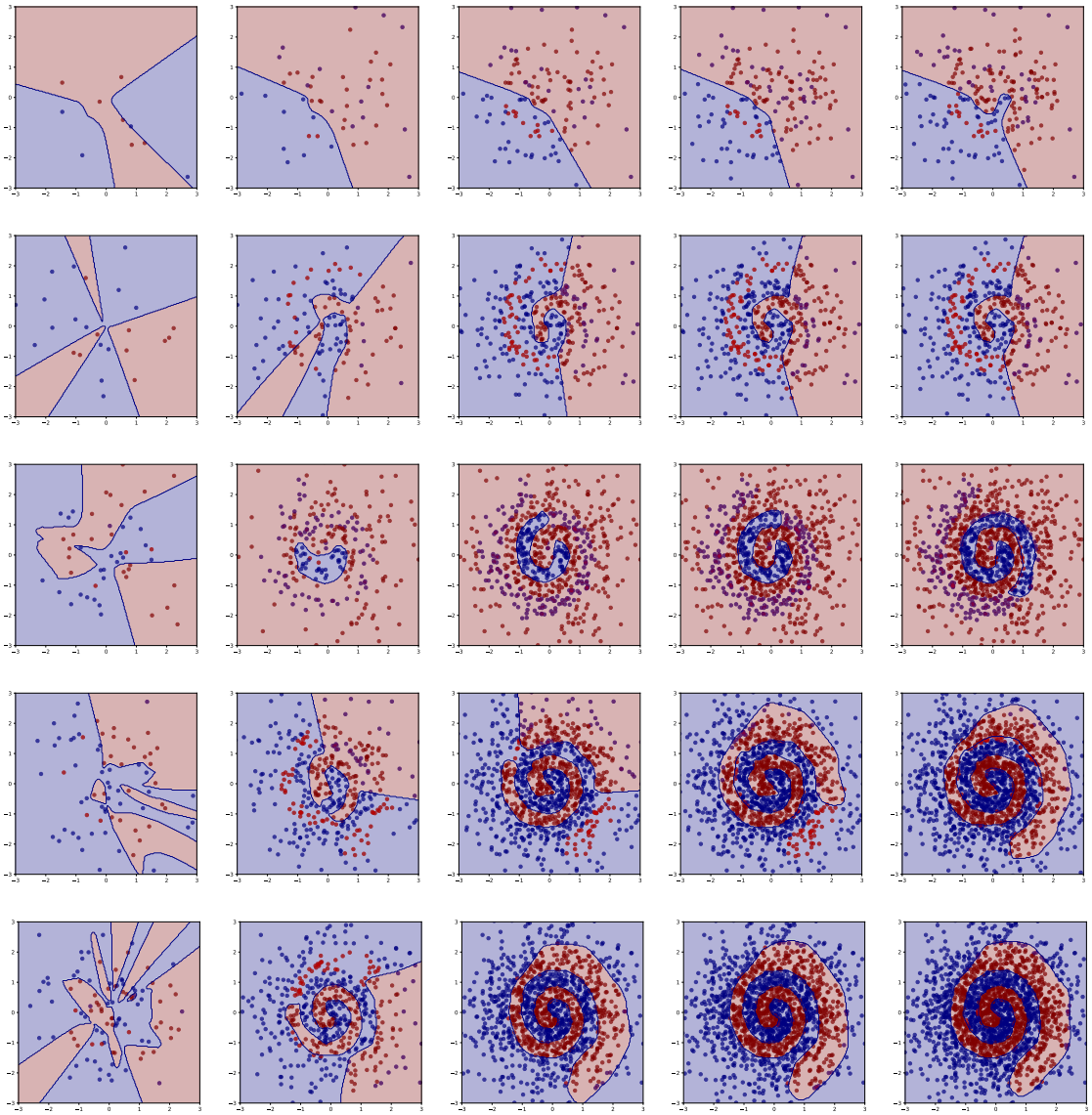


Figure 10: From top to bottom, instantaneous decision boundaries for $N = 10$, $N = 25$, $N = 50$, $N = 75$ and $N = 100$ at steps (a) 1, (b) 5, (c) 10, (d) 15 and (e) 20 for the *moons* dataset.



(u) (v) (w) (x) (y)

Figure 11: From top to bottom, instantaneous decision boundaries for $N = 10$, $N = 25$, $N = 50$, $N = 75$ and $N = 100$ at steps (a) 1, (b) 5, (c) 10, (d) 15 and (e) 20 for the *spirals* dataset.

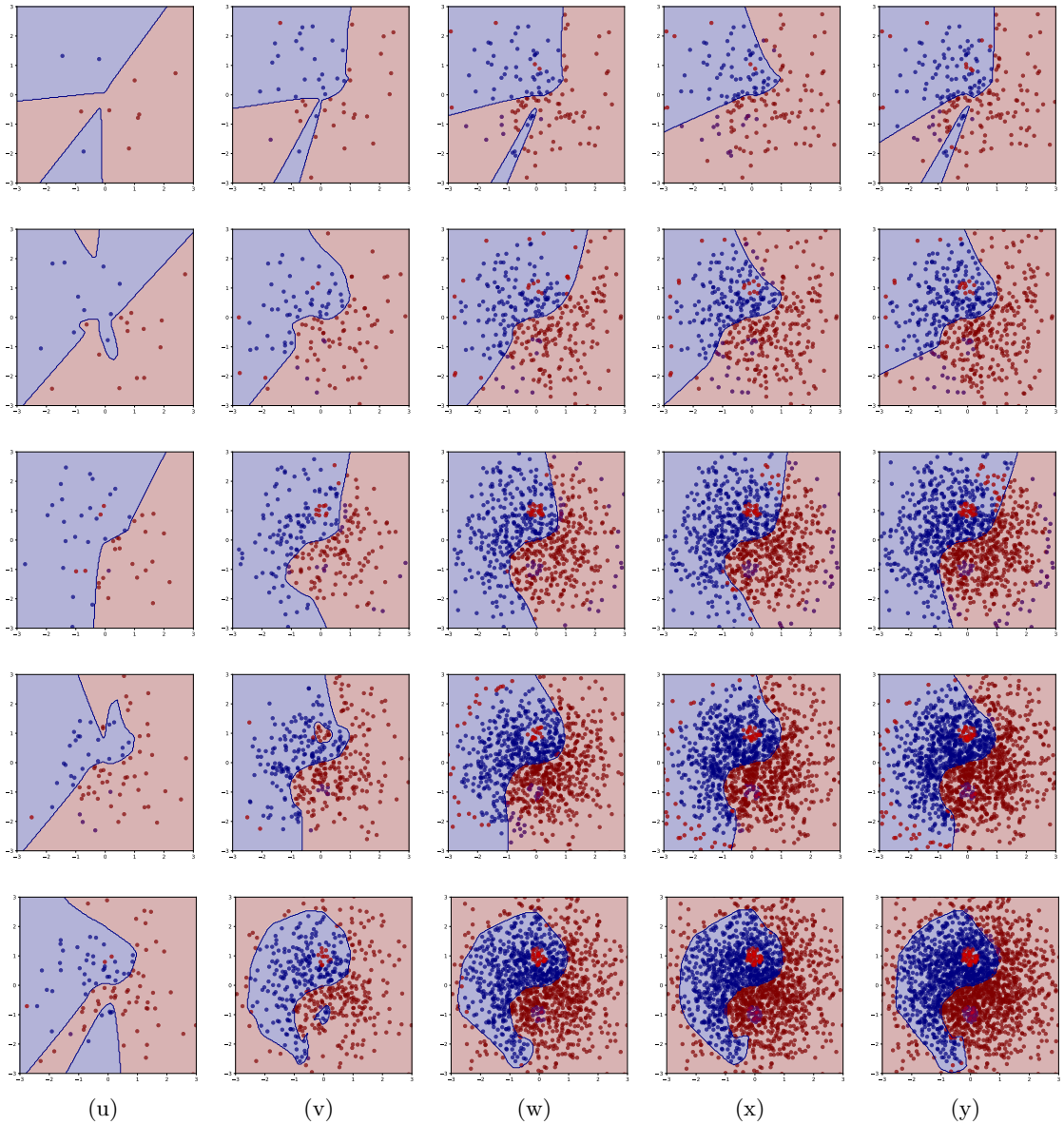


Figure 12: From top to bottom, instantaneous decision boundaries for $N = 10$, $N = 25$, $N = 50$, $N = 75$ and $N = 100$ at steps (a) 1, (b) 5, (c) 10, (d) 15 and (e) 20 for the *yin-yang* dataset.

Dataset	$\mathcal{A}_{\mathcal{C},sp}$	$\mathcal{A}_{\mathcal{C},sq}$	*Best accuracy*		*Best efficiency*	
			eff.	AUC_{acc}	*eff.*	AUC_{acc}
iris	0.962	0.963	0.825	0.931	0.964	0.897
wine	0.943	0.926	0.743	0.891	0.918	0.856
bank	0.808	0.801	0.776	0.705	0.958	0.677
spirals	1.0	1.0	0.712	0.942	0.886	0.897
breast-cancer	0.736	0.752	0.638	0.71	0.978	0.691
miniboone	0.736	0.699	0.523	0.658	0.523	0.658
breast-cancer-wisc	0.934	0.925	0.516	0.895	0.976	0.781
breast-tissue	0.541	0.557	0.706	0.518	0.895	0.51
breast-cancer-wisc-prog	0.73	0.709	0.976	0.657	0.976	0.657
conn-bench-sonar-mines-rocks	0.826	0.807	0.695	0.761	0.765	0.75
mammographic	0.823	0.804	0.641	0.774	0.81	0.758
molec-biol-splice	0.59	0.583	0.543	0.546	0.788	0.537
ringnorm	0.905	0.924	0.687	0.756	0.912	0.737
synthetic-control	0.693	0.722	0.878	0.631	0.878	0.631
tic-tac-toe	0.876	0.891	0.525	0.817	0.803	0.787
mushroom	0.962	0.93	0.711	0.839	0.761	0.835
chess-krvkp	0.983	0.961	0.691	0.909	0.761	0.896
heart-hungarian	0.769	0.762	0.639	0.73	0.964	0.693
ilpd-indian-liver	0.682	0.679	0.535	0.649	0.979	0.605
fertility	0.91	0.908	0.806	0.877	0.987	0.8
waveform-noise	0.841	0.825	0.361	0.796	0.689	0.773
musk-2	0.805	0.718	0.979	0.602	0.979	0.602
parkinsons	0.878	0.86	0.751	0.757	0.811	0.755
pittsburg-bridges-REL-L	0.671	0.679	0.99	0.654	0.991	0.645
pima	0.722	0.73	0.989	0.707	0.99	0.699
ionosphere	0.932	0.914	0.449	0.854	0.692	0.829
hepatitis	0.811	0.813	0.982	0.775	0.982	0.775
pittsburg-bridges-T-OR-D	0.857	0.862	0.994	0.84	0.996	0.84
oocytes_trisopterus_nucleus_2f	0.741	0.683	0.714	0.608	0.763	0.607
contrac	0.568	0.567	0.761	0.546	0.947	0.535
oocytes_merluccius_nucleus_4d	0.579	0.596	0.728	0.538	0.985	0.498
seeds	0.87	0.87	0.85	0.845	0.984	0.794
waveform	0.833	0.825	0.428	0.801	0.849	0.769
magic	0.804	0.804	0.777	0.786	0.991	0.766
planning	0.703	0.703	0.994	0.689	0.994	0.689
haberman-survival	0.651	0.648	0.867	0.602	0.978	0.56
cylinder-bands	0.736	0.71	0.56	0.664	0.649	0.654
echocardiogram	0.831	0.831	0.911	0.802	0.99	0.801
energy-y1	0.962	0.957	0.621	0.92	0.937	0.892
vertebral-column-3clases	0.825	0.819	0.713	0.791	0.979	0.74

Table 2: Summary of UCI datasets and their main characteristics