Memory Bounds for Continual Learning

Xi Chen
Columbia University
xichen@cs.columbia.edu

Christos Papadimitriou
Columbia University
christos@columbia.edu

Binghui Peng
Columbia University
bp2601@columbia.edu

April 25, 2022

Abstract

Continual learning, or lifelong learning, is a formidable current challenge to machine learning. It requires the learner to solve a sequence of k different learning tasks, one after the other, while retaining its aptitude for earlier tasks; the continual learner should scale better than the obvious solution of developing and maintaining a separate learner for each of the k tasks. We embark on a complexity-theoretic study of continual learning in the PAC framework. We make novel uses of communication complexity to establish that any continual learner, even an improper one, needs memory that grows linearly with k, strongly suggesting that the problem is intractable. When logarithmically many passes over the learning tasks are allowed, we provide an algorithm based on multiplicative weights update whose memory requirement scales well; we also establish that improper learning is necessary for such performance. We conjecture that these results may lead to new promising approaches to continual learning.

1 Introduction

Machine learning has made dizzying empirical advances over the last decade through the powerful methodology of deep neural networks — learning devices consisting of compositions of parametrized piecewise linear functions trained by gradient descent — achieving results near or beyond human performance at a plethora of sophisticated learning tasks in a very broad number of domains. There are several remaining challenges for the field, however, and in order to address these theoretical work seems to be needed. These challenges include the methodology's reliance on massive labeled data, the brittleness of the resulting learning engines to adversarial attacks, and the absence of robust explainability and the ensuing risks for applying the technique to classification tasks involving humans. This paper is about another important ability that deep neural networks appear to lack when compared to animal learners: *They do not seem to be able to learn continually.*

Brains interact with the environment through sensory and motor organs, and survive by learning the environment's ever changing structure and dynamics, constraints and rules, risks and opportunities. This learning activity continues throughout the animal's life: it is *continual and life-long.* We learn new languages, new math and new skills and tricks, we visit new cities and we adapt to new situations, we keep correcting past mistakes and misapprehensions throughout our lives.

It has long been observed that neural networks cannot do this [MMO95]. In fact, when a neural network is tasked with learning a sequence of data distributions, the phenomenon of *catastrophic interference* or *catastrophic forgetting* is likely to occur: soon, the performance of the device on past tasks will degrade and ultimately collapse. This phenomenon seems to be quite robust and ubiquitous, see [GMX+13, PKP+19]. This inability is important, because many applications of machine learning involve autonomous agents and robots, domains in which one needs to continually learn and adapt [PKP+19].

There is a torrent of current empirical work in continual/life-long learning. There is a multitude of approaches, which can be roughly categorized this way: First, *regularization* of the loss function is often used to discourage changes in learned parameters in the face of new tasks. Second, the learning architecture (the parameter space of the device) can be designed so that it unfolds *dynamically* with each new task, making new parameters available to the task and freezing the values of some already learned parameters. Finally, there is the approach of *replay,* influenced by the *complementary learning systems* theory of biological learning [MMO95], in which previously used data are selected and revisited periodically. Even though none of these families of approaches seems to be particularly successful currently, our positive results seem to suggest that this latter replay approach may be the most promising.

Theoretical work on PAC learning over the past four decades [SSBD14] has shed light on the power and limitations of learning algorithms. The powerful tool of the VC-dimension [VC15] captures nicely the training data requirements of the learning task, while it also serves as a useful proxy for the number of parameters required of any neural network for solving a particular PAC learning task. The PAC theory's predictions of computational intractability, on the other hand, have been upended by the avalanche of deep nets, and the surprising aptness of gradient descent to approximate exquisitely a certain broad class of non-convex minimization problems. *This paper is a complexity analysis of continual learning in the context of PAC learning.*

How can one model continual learning in the PAC framework? Assume a sequence of k learning tasks, that is, a sequence of distributions over labeled data from corresponding hypothesis classes. Our goal is to learn these k tasks sequentially, starting from the first and proceeding, and end up with a learning device that performs well on all. That is, the device with high probability tests well in all tasks — this can be modeled by testing at the end on a randomly chosen task from those already learned.

Obviously, this problem can be solved by training k different devices. The interesting question is, can it be done with fewer resources, by reusing experience from past learning to achieve the next task? *Is there quantity discount in continual learning?* Our results strongly suggest, in a qualified way, a negative answer. Our main lower bound result is for *improper* learning — the more powerful kind, where the learner is allowed distributions outside the hypothesis class — this is crucial, since deep neural networks can be considered improper.

Our main result bounds from below the amount of *memory* needed for continual learning. Memory has been identified at a recent continual learning workshop [RTCM+21] as one of the main bottlenecks in the practice of continual learning. If we know that there is no "quantity discount" in memory, this suggests that a new parameter space is needed to accommodate new learning tasks — in effect, one essentially needs a new device for each task to avoid catastrophic forgetting.

On the positive side, we show that, if we are allowed to make multiple passes over the sequence of tasks, an algorithm based on the multiplicative weight update competition of several learners (inspired by the algorithm in [BBFM12]) requires significantly less memory to successfully learn the distributions. At a first glance, multiple pass algorithms may seem of limited value to life-long learning — after all, you only live once ... But in application domains such as robotics or autonomous driving, the revisiting of similar environments may be routine. Also, the positive result does suggest that the replay strategy in continual learning explained above may achieve some less exacting goal than learning all distributions — for example, passing the $k+1$ distributions test with some high probability. We develop this idea further in the Discussion section.

Finally, we show an *exponential* separation result establishing the power of improper learning algorithms in the multi-pass case: We show that any *proper* learning algorithm requires at least polynomial number of passes to match the performance of the best improper learner.

1.1 Our results

Let $\mathcal{H}$ be a hypothesis class with VC dimension d. Let $\mathcal{X}$ be its data universe and let $b = \log_2 |\mathcal{X}|$ be its description size (so every hypothesis $h \in \mathcal{H}$ maps $\mathcal{X}$ to $\{0,1\}$ and every data point takes b bits to store). We are interested in continual learners that make one pass (or multiple passes) on a sequence of k learning tasks $\mathcal{D}_1, \dots, \mathcal{D}_k$, each of which is a data distribution (i.e., supported on $\mathcal{X} \times \{0,1\}$). When making a pass on $\mathcal{D}_i$, the learner can draw as many samples as it wants from $\mathcal{D}_i$ and uses them to update its memory. We consider the realizable setting where there exists a hypothesis $h \in \mathcal{H}$ that is consistent with all $\mathcal{D}_i$'s. The goal of a (ϵ, δ)-continual learner is to recover a function $h : \mathcal{X} \to \{0,1\}$ that, with probability at least $1-\delta$, has no more than ϵ error with respect to every distribution $\mathcal{D}_i$. We say a learner is *proper* if the function h is always a hypothesis in $\mathcal{H}$, and we will consider both proper and improper learners. We are interested in minimizing the memory of continual learners. Formal definitions of the model can be found in Section 2.

Our main result is a lower bound, showing that $\Omega(kdb/\epsilon)$-bit memory is necessary for any general improper continual learning algorithm. In other words, there is no gain in learning resources from the sequential process:

Theorem 1.1. *There exists a sufficiently large constant C_0 such that the following holds. For any $\epsilon \in (0, 0.01]$ and any positive integers k, d and b with $b \geq C_0 \log(kd/\epsilon)$, there is a hypothesis class $\mathcal{H}$ of VC dimension $2d$ over a universe $\mathcal{X}$ of description size b such that any $(\epsilon, 0.01)$-continual learner for $\mathcal{H}$ over a sequence of $k+1$ tasks requires $\Omega(kdb/\epsilon)$ bits of memory.*

Next we show that allowing a continual learner to make multiple passes can significantly reduce the amount of memory needed. More formally, we give a *boosting* algorithm that uses $(k/\epsilon)^{O(1/c)}db$

bis of memory only by making c passes over the sequence of k tasks. Note that, after a logarithmic number of passes, the memory resource almost matches the requirement of a single task.

Theorem 1.2. *Let $\epsilon \in (0, 1/10]$ and k, d, b, c be four positive integers. Let $\mathcal{H}$ be any hypothesis class of VC dimension d over a data universe $\mathcal{X}$ of description size b. There is a c-pass $(\epsilon, 0.01)$-continual learner for $\mathcal{H}$ over sequences of k tasks that has memory requirement and sample complexity*

$$\left(\frac{k}{\epsilon}\right)^{2/c} \cdot db \cdot \mathrm{poly}\big(c, \log k/\epsilon\big) \quad \textit{and} \quad \left(\frac{k}{\epsilon}\right)^{4/c} \cdot \frac{k+d}{\epsilon} \cdot \mathrm{poly}\big(c, \log kd/\epsilon\big),$$

respectively. Whenever $c \geq 4\log(k/\epsilon)$, the memory requirement becomes $db \cdot \mathrm{polylog}(kd/\epsilon)$ and the sample complexity becomes $\widetilde{O}((k+d)/\epsilon)$, both are optimal up to a polylogarithmic factor.

Our continual learner in Theorem 1.2 is *improper*, and our last result shows that this is necessary: We prove that any proper c-pass continual learner needs (roughly) $kdb/(c^3\epsilon)$ memory. This gives an exponential separation between proper and improper continual learning in terms of c.

Theorem 1.3. *There exists a sufficiently large constant C_0 such that the following holds. For any $\epsilon \in (0, 1/4]$ and any positive integers c, k, d and b such that $d \geq C_0 c$ and $b \geq C_0 \log(kd/(c\epsilon))$, there is a hypothesis class $\mathcal{H}$ of VC dimension d defined over a universe $\mathcal{X}$ of description size b such that any c-pass $(\epsilon, 0.01)$-continual learner for $\mathcal{H}$ over $2k$ tasks requires $(1/c^2) \cdot \widetilde{\Omega}(kdb/(c\epsilon))$ memory.*

1.2 Technique overview

1.2.1 Lower bounds for improper learning

We start by explaining the main ideas in our lower bound proof of Theorem 1.1.

The third party. Communication complexity is a commonly used approach for establishing lower bounds in learning theory. Some of recent examples include [DKS19] and [KLMY19], both of which make reductions from two-party communication problems. A natural formulation of our continual learning setting as a two-party communication problem is to split the k tasks $\mathcal{D}_1, \ldots, \mathcal{D}_k$ as inputs of Alice and Bob, and the goal is to lowerbound the one-way communication complexity for Bob to output a function that has small loss for every $\mathcal{D}_i$. However, this two-party approach is doomed to fail since Alice can just send a hypothesis function $h \in \mathcal{H}$ that is consistent with her tasks (by Sauer-Shelah, each function in $\mathcal{H}$ can be described using roughly db bits). After receiving h, Bob can output a function h' such that for any $x \in \mathcal{X}$: (1) If x is in the support of any $\mathcal{D}_i$ held by Bob, $h'(x)$ is set according to $\mathcal{D}_i$; (2) Otherwise, $h'(x) = h(x)$. It is easy to verify that h' achieves 0 loss for all $\mathcal{D}_i$ (given that we are in the realizable setting).

The reason why the two-party approach fails is that it fails to capture the following key challenge behind improper learning with limited memory: The function returned by the learner at the end, while can be any function, must have a concise representation since it is determined by the learner's limited memory. *Our main conceptual contribution in circumventing this difficulty is the introduction of a third party: Charlie as a tester.* While $\mathcal{D}_1, \ldots, \mathcal{D}_k$ are split between Alice and Bob as before, Charlie receives a data point drawn from a certain mixture of distributions of Alice and Bob and needs to return its correct label after receiving the message from Bob. This three-party model captures the challenge described above since the number of bits needed to represent the function (in the message from Bob to Charlie) is counted in the communication complexity. We believe that the introduction of a third party in the communication problem may have further applications in understanding communication/lower bounds for improper learning problems in the future.

A warm-up: Low-dimensional data. We will consider ϵ as a small positive constant throughout the overview for convenience. We start with a warm-up for the case when the data universe is low-dimensional, i.e., when $b := \log_2 \mathcal{X} = O(\log(kd))$, and we aim for a memory lower bound of $\Omega(kd)$.

The basic building block is the following one-way three-party communication problem:

- Let $B = kd$. Alice holds a single element $a \in [B]$ with label 1;
- Bob holds $B/2$ elements from $[B] \setminus \{a\}$ with label 0;
- Charlie receives an element that is either a with probability 1/2 or drawn uniformly from Bob's elements with probability 1/2, and needs to output its label.

It is not difficult to show that either Alice sends $\Omega(1)$ bits of information to Bob, or Bob needs to send $\Omega(B)$ bits of information to Charlie, in order for Charlie to succeed with say probability 2/3.

Next we construct a hypothesis class $\mathcal{H}$ and use it to define a communication problem that (1) can be reduced to the continual learning of $\mathcal{H}$ and (2) can be viewed as an indexing version of the direct sum of kd copies of the building block problem described above. The data universe $\mathcal{X}$ of $\mathcal{H}$ is $[k] \times [d] \times [B]$ (so $b = O(\log(kd))$ given that $B = kd$). Each hypothesis $h_{i,A} : \mathcal{X} \to \{0, 1\}$ in $\mathcal{H}$ is specified by an $i \in [k]$ and $A = (a_1, \ldots, a_d) \in [B]^d$ as follows:

$$h_{i,A}(x) = \begin{cases} 1 & \text{if } x_1 \neq i \\ 1 & \text{if } x_1 = i \text{ and } x_3 = a_{x_2} \\ 0 & \text{otherwise (i.e., } x_1 = i \text{ and } x_3 \neq a_{x_2}) \end{cases}$$

It is easy to show that $\mathcal{H}$ has VC dimension $O(d)$. In the three-party communication problem:

- Alice holds $(a_{i,j} : i \in [k], j \in [d])$, where each $a_{i,j}$ is drawn from $[B]$ independently and uniformly at random, and she views her input as a sequence of k data distributions $\mathcal{D}_1, \ldots, \mathcal{D}_k$, where $\mathcal{D}_i$ is uniform over $(a_{i,j} : j \in [d])$ and all labels are 1.
- Bob holds i^* drawn uniformly random from $[k]$ and $(A_j : j \in [d])$, where each A_j is a size-$(B/2)$ subset drawn from $[B] \setminus \{a_{i^*,j}\}$ independently and uniformly at random. Bob views his input as a data distribution $\mathcal{D}_{k+1}$ that is uniform over $\cup_j A_j$ and all labels are 0. (Note that $\mathcal{D}_1, \ldots, \mathcal{D}_{k+1}$ are realizable in $\mathcal{H}$.)
- Charlie holds i^*, j^* and r^*, where j^* is drawn uniformly from $[d]$ and r^* is a_{i^*,j^*} with probability 1/2 and is drawn uniformly from A_{i^*,j^*} with probability 1/2, and Charlie needs to return the label of (i^*, j^*, r^*).

On the one hand, there is an intuitive reduction from this three-party communication problem to continual learning of $\mathcal{H}$: Alice and Bob simulates a continual learner for $\mathcal{H}$ over their $k+1$ data distributions $\mathcal{D}_1, \ldots, \mathcal{D}_{k+1}$ and Bob sends the memory of the learner at the end to Charlie. Charlie recovers a function $h : \mathcal{X} \to \{0, 1\}$ from Bob's message and uses it to return the label of (i^*, j^*, r^*). As a result, communication complexity of the three-party communication problem is a lower bound for memory requirement of the continual learning of $\mathcal{H}$ (of VC dimension $O(d)$) over $k + 1$ tasks.

On the other hand, to see (informally) why $\Omega(kd)$ is a natural lower bound for the communication complexity, we note that Alice has no idea about i^* and j^* so the amount of information she sends about a_{i^*,j^*} is expected to be $o(1)$ if her message to Bob is of length $o(kd)$ (this can be made formal using information-theoretic direct-sum-type arguments). When this happens, Bob must send $\Omega(B)$ $= \Omega(kd)$ bits (even if Bob knows j^*) in order for Charlie to have a good success probability.

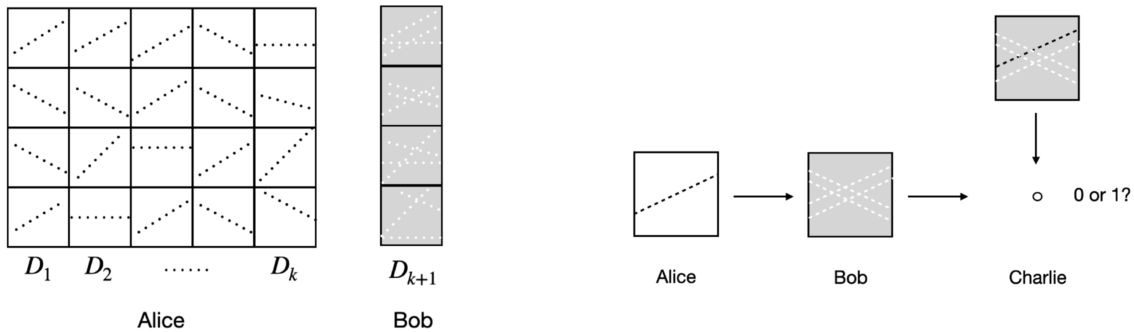

Figure 1: Hard instances of improper learning. A black dot indicate label-1 data, white background/dot means missing data, gray background means label 0 data. The left figure illustrates the hard instance of continual learning. The right figure presents the basic building block of the asymmetric communication problem.

Handling high-dimensional data. The key technical challenge behind the proof of Theorem 1.1 comes from the high-dimensional data case, i.e., when $b := \log_2 |\mathcal{X}| \gg \log(kd)$, in which case our target lower bound is $\Omega(kdb)$ instead of $\Omega(kd)$. A natural approach is to change the parameter B in the building block communication problem of the low-dimension case to (roughly) 2^b and show that it has a communication lower bound of $\Omega(b)$. However, this is *not true* as one can randomly hash Alice's input element $a \in [B]$. For example, the players can use public randomness to partition $[B]$ into 100 random sets $B_1, \ldots, B_{100}$ of equal size and Alice sends $O(1)$ bits to Bob to reveal the B_i that contains a. Bob can forward i to Charlie and Charlie returns 1 if his element lies in B_i and 0 otherwise. Given that Bob's set most likely has small overlap with B_i, this simple $O(1)$-bit protocol succeeds with high probability, and it holds no matter how large B is. The above issue does not come up in the low-dimensional case since we only need an $\Omega(1)$ lower bound for Alice's message there. There is indeed a deep connection with the so-called *representational dimension* of a hypothesis class (see [BNS19, FX15] for the exact definition). For example, for every hypothesis class $\mathcal{H}$, under uniform data distribution, there always exists an ϵ-net of $\mathcal{H}$ with size $O(\epsilon^{-d})$ and it is independent of the representation size b of its data universe [Hau95].

We overcome this technical challenge by imposing combinatorial structures on the hypothesis class through the *line functions.* A line function $f_{a,b}$ is defined over a finite field $\mathbb{F}_p^2$ and is specified by $a, b \in \mathbb{F}_p$; it labels $(x, y) \in \mathbb{F}_p^2$ 1 iff it lies on the line $ax + y \equiv b \pmod p$. The representational dimension of line functions is known to be $\Omega(\log p)$ [Aar04, FX15] which matches the representation size $2\log_2 p$. More precisely, one needs to reveal $\Omega(\log p)$ bits about (a, b) in order for a second party to distinguish whether a point is drawn uniformly from the line or uniformly from $\mathbb{F}_p^2$.

However, a lower bound on the representational dimension is far from enough for our purpose. We need a lower bound for the following three-party one-way communication problem as our building block (which plays a similar role as the building block problem in the low-dimensional case):

- Alice holds a line over $\mathbb{F}_p^2$. Bob holds a set of points of $\mathbb{F}_p^2$ in which points of exactly $0.2p$ lines are excluded, one of which is the line of Alice.

- Charlie receives a query point in $\mathbb{F}_p^2$ that is either (1) drawn uniformly from the line of Alice with probability 1/2, or (2) drawn from Bob's set uniformly with probability 1/2, and the goal is to tell which case it is.

We prove that in order for Charlie to succeed with high probability, either Alice needs to send a message of length $\Omega(\log p)$ to Bob, or Bob needs to send a message of length $\Omega(p^{0.5})$ to Charlie. We leave the technical detail of this lower bound to Section 3, and outline the key intuition here: (i) The advantage of the line function is that Charlie needs $\Omega(\log p)$ bits of information about Alice's line to succeed with high probability; (ii) If Alice's message has length $o(\log p)$, then many of Bob's $\Theta(p)$ candidate lines are equally likely to be Alice's line, in which case Bob must send a large amount of information to Charlie to satisfy (i).

The rest of the proof is similar to the low-dimensional case. We introduce a hypothesis class $\mathcal{H}$ and a three-party communication problem that (1) can be reduced to the continual learning of $\mathcal{H}$ and (2) can be viewed as an indexing version of the direct sum of kd copies of the building block problem. See Figure 1 for an illustration for hard instances of the communication problem. Our lower bound is then obtained by reducing to that of the building block problem via standard compression and direct-sum arguments.

1.2.2 The multi-pass algorithm

When a continual learner can make c passes over the sequence of tasks, we show that its memory requirement can be significantly reduced to (roughly) $(k/\epsilon)^{O(1/c)}db$. In this overview, we focus on the case when $c = \Theta(\log(k/\epsilon))$.

Our algorithm builds on the idea of multiplicative weights update (MWU) [AHK12] or boosting [Sch90], which has been used before in similar but different contexts.

- [BHPQ17, CZZ18, NZ18] focus on the sample complexity of multi-task learning. By viewing each distribution $\mathcal{D}_i$ ($i \in [k]$) as an expert, one can run MWU over $O(\log k)$ rounds and return the majority vote. Implementing their algorithm as a $O(\log k)$-pass continual learner reduces the memory to $\widetilde{O}(db/\epsilon + k)$, with the optimal sample complexity of $\widetilde{O}((d + k)/\epsilon)$. However, the linear dependence on ϵ in the memory upper bound is inevitable, as they treat each distribution as a black box.

- [BBFM12] focuses on the communication aspect of PAC learning over a uniform mixture of tasks and gives an approach that achieves a logarithmic dependency on ϵ. More precisely, their protocol first draws $O(d/\epsilon)$ samples from the uniform mixture distribution $\mathcal{D}$ and maintains a weighted distribution over the empirical samples set (initially set to be uniform). Each round, it only transmits $O(d)$ samples and obtains a weak learner with $O(1)$ error. By updating the weighted distribution via MWU and taking a majority vote after $O(\log(1/\epsilon))$ rounds, the communication complexity is $\widetilde{O}(db)$.

MWU with rejection sampling. Our approach resembles that of [BBFM12]. The algorithm optimizes the loss on the uniform mixture $\mathcal{D} = (1/k)\sum_{i=1}^{k} \mathcal{D}_i$ of $\mathcal{D}_1, \ldots, \mathcal{D}_k$ and obtains an (ϵ/k)-accurate classifier of $\mathcal{D}$, which is a sufficient condition for the continual learner to succeed. However, to execute the MWU algorithm, the continual learner cannot draw an empirical training set as it requires $\Omega(kdb/\epsilon)$ memory already. Instead, we run the boosting algorithm over the entire distribution $\mathcal{D}$ and *implicitly* maintain the importance weight of every data point. In particular, (1) we explicitly compute the importance weight of each distribution $\mathcal{D}_i$; (2) we implicitly maintain the updated weight of every data point in $\mathcal{D}_i$. For (1), we can simply estimate the empirical weights via finite samples and a multiplicative approximation suffices. For (2), we cannot explicitly write down the updated weights of every single data point, but it suffices to sample from the updated distribution. In particular, we can run rejection sampling on the original distribution $\mathcal{D}_i$, and due to the MWU rule, the updated weights are within a multiplicative factor of $\widetilde{O}(k^2/\epsilon^2)$. Combining with

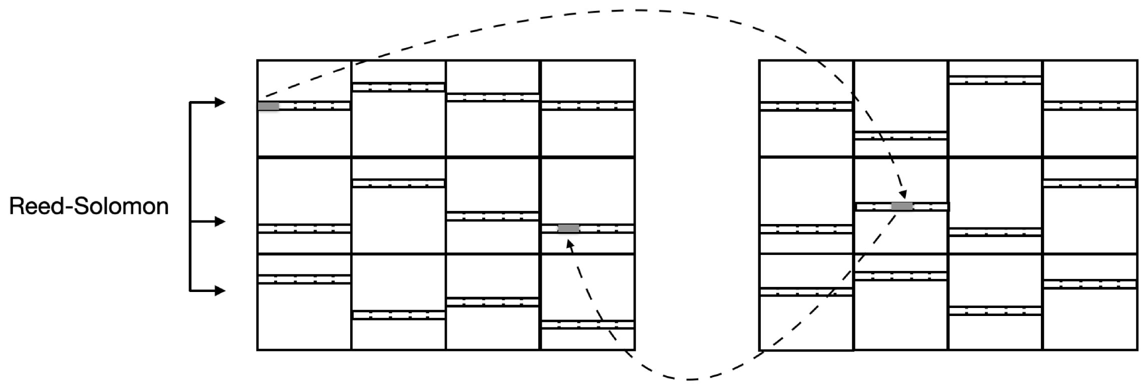

Figure 2: Hard instance of proper learner.

a standard analysis of MWU, this yields an algorithm of memory $\widetilde{O}(db+k)$ and sample complexity $\widetilde{O}(k^2d/\epsilon^2)$.

Optimal sample complexity via truncated rejection sampling. The major disadvantage of the previous approach comes from the increase of sample complexity. At a first glance, it seems impossible to use $o(kd/\epsilon)$ samples as it is the minimum requirement to achieve (ϵ/k)-error over the uniform mixture distribution $\mathcal{D}$. Technically, the overhead comes from both estimation of the weights and the rejection sampling step, as the weight could blow up by a factor of $O(k^2/\epsilon^2)$. The key observation is that one can iteratively truncate the top $O(\epsilon/c)$-quantile of the update distribution at each round. This brings a total loss of $O(\epsilon)$ for each distribution $\mathcal{D}_i$ after c-rounds (hence breaking the aforementioned lower bound) and this is affordable. Meanwhile, the truncated data distribution is within a multiplicative factor of $O(1/\epsilon)$ from the original data distribution $\mathcal{D}_i$ due to a Markov bound. Hence, both sampling and estimation get easier and introduce $O(1/\epsilon)$ overheads only. Combining with a robust analysis of MWU and a streaming implementation, overall, our algorithm requires $O(\log k/\epsilon)$ passes, $\widetilde{O}(db)$ memory and sample complexity $\widetilde{O}((d+k)/\epsilon)$.

1.2.3 Lower bounds for multi-pass proper learning.

Finally, we outline the proof of Theorem 1.3 which gives a memory lower bound for proper learners that make c passes on k tasks. We reduce from the classical *pointer chasing* problem. In the two-party pointer chasing problem, Alice and Bob hold two maps $f_A, f_B : [n] \to [n]$, respectively. Let $g : [n] \to [n]$ be $g(i) := f_B(f_A(i))$, and $g^{(1)}(i) = g(i)$, $g^{(\tau)}(i) := g(g^{(\tau-1)}(i))$. The goal is for Bob to compute $g^{(c+1)}(1) \pmod 2$. When Alice speaks first, if $2c+1$ rounds of communication are allowed, $O(c\log n)$ total communication is sufficient; on the other hand, when only $2c$ rounds are allowed, there is a lower bound of $\Omega(n/c - c\log n)$ [NW91, Kla00, Yeh20].

The major technical challenge lies in the construction of our hypothesis class which allows us to embed the pointer chasing problem in its multi-pass continual learning.

First attempt. Again we ignore ϵ and assume it to be a small positive constant. We also assume $b \gg \log(kd)$ for simplicity. Given c, k, d and b, our goal is to construct a hypothesis class $\mathcal{H}$ with VC dimension $O(cd)$ and representation size $\widetilde{O}(b)$ such that its continual learning over $2k$ tasks requires $\Omega(kdb/c^2)$ memory. Let $n = kdb$. We start by describing the data universe $\mathcal{X}$ of $\mathcal{H}$ and show that input pairs (f_A, f_B) with $f_A, f_B : [n] \to [n]$ of the pointer chasing problem can be viewed equivalently as a special class of Boolean functions over $\mathcal{X}$.

To this end, we partition $\mathcal{X}$ into $Y \cup Z$, Y (or Z) is partitioned into k blocks $Y_1, \ldots, Y_k$ (or $Z_1, \ldots, Z_k$), each block Y_i is further partitioned into d cells $Y_{i,1}, \ldots, Y_{i,d}$ (or $Z_{i,1}, \ldots, Z_{i,d}$), and each cell $Y_{i,j}$ (or $Z_{i,j}$) is a disjoint copy of $[(kdb)^b]$ (which we will view as a tuple from $[n]^b$ given that $n = kdb$). Note that $\log_2 |\mathcal{X}| = O(b \log(kdb)) = \widetilde{O}(b)$ as promised. Next we view f_A as kd data points in $\mathcal{X}$, one from each cell $Y_{i,j}$: $(f_A(1), \ldots, f_A(b))$ identifies an data point in $Y_{1,1}$ (viewing $Y_{1,1}$ as a copy of $[n]^b$) and in general $(f_A((i-1)db + (j-1)b + 1), \ldots, f_A((i-1)db + (j-1)b + b))$ identifies an data point in $Y_{i,j}$. Similarly we view f_B as kd data points, one from each cell $Z_{i,j}$. Then (f_A, f_B) corresponds to the function h that is 1 over these $2kd$ data points and 0 elsewhere.

Let $\mathcal{H}$ denote the class of all such Boolean functions h over $\mathcal{X}$ (for now). Then ideally we would like to reduce the pointer chasing problem about (f_A, f_B) to the c-pass continual learning of $\mathcal{H}$. To this end, we have Alice and Bob follow the following steps:

- Given f_A, Alice prepares the following k data distributions $\mathcal{D}_1, \ldots, \mathcal{D}_k$: each $\mathcal{D}_i$ is uniform over the d data points she identifies in Y_i (one from each $Y_{i,j}$) using f_A and all are labelled 1;

- Given f_B, Bob prepares the following k data distributions $\mathcal{D}_{k+1}, \ldots, \mathcal{D}_{2k}$: each D_{k+i} is uniform over the d data points he identifies in Z_i using f_B and all are labelled 1;

- Alice and Bob run a c-pass proper continual learner for $\mathcal{H}$ over these $2k$ data distributions using $2c - 1$ rounds of communication; When the continual learner terminates, Bob uses the function $h \in \mathcal{H}$ it returns (since the learner is proper) to decodes f_A and then solve the pointer chasing problem.

There are two issues with this approach. First, the idea of having Bob recover f_A at the end feels too good to be true. One would see the caveat when analyzing the VC dimension of $\mathcal{H}$ defined above, which turns out to be $O(kd)$ instead of the $O(cd)$ we aim for. To resolve this issue, we refine the definition of the hypothesis class $\mathcal{H}$ by requiring each $h \in \mathcal{H}$ to have $c + 1$ special blocks Y_i, each of which contains d data points labelled by 1 and everything else labeled by 0 as before; all data points in other non-special blocks must have label 1, and the same condition holds for Z. As expected, these $2c + 2$ special blocks are exactly those involved in the pointer chasing process. On the one hand, this refinement makes sure that the VC dimension of the new $\mathcal{H}$ stays at $O(cd)$ instead of $O(kd)$; on the other hand, if Bob recovers h, he can obtain label-1 data points from those $c + 1$ relevant blocks of Y which are sufficient for him to follow the pointer chasing process.

Incorporating error correction codes. The second issue is that the function $h \in \mathcal{H}$ that Bob receives from the continual learner only has the weak guarantee that its loss with respect to each $\mathcal{D}_i$ is small. In particular, for any relevant block Y_i of Y, this means that Bob is only guaranteed to recover a large portion of the d label-1 data points in it (or equivalently, a large portion of db many entries of f_A in which one of them is the next pointer for Bob to follow). However, this is not good enough since the wrong data points may contain the entry of f_A that Bob should follow.

Our key idea to circumvent this issue is to incorporate *error correction codes* (ECC) into the construction. To this end, the new data universe $\mathcal{X}$ will be similarly partitioned into blocks Y_i and Z_i but now each block Y_i has $2d$ cells $Y_{i,1}, \ldots, Y_{i,2d}$ instead, each cell being a disjoint copy of $[(kdb)^b]$. Taking the first block Y_1 as an example, Alice will still identify d elements in $[(kdb)^b]$:

$$\big(f_A(1), \ldots, f_A(b)\big), \ldots, \big(f_A((d-1)b + 1), \ldots, f_A((d-1)b)\big)$$

using f_A but now she will apply the Reed-Solomon code to obtain a $2d$-tuple $(z_1, \ldots, z_{2d})$ with $z_j \in [(kdb)^b]$ (for now imagine that $[(kdb)^b]$ is a finite field). The function will have $2d$ data points labelled 1 in Y_1: z_i in each $Y_{1,i}$. Modifying the hypothesis class $\mathcal{H}$ this way allows Bob to fully decode the $c+1$ relevant blocks in Y even if he only receives a hypothesis $h \in \mathcal{H}$ from the continual learner that has small loss on every $\mathcal{D}_i$. We believe that the use of ECC may find broader applications in helping understand the complexity of learning problems in the future.

1.3 Additional related work

Continual/Lifelong learning. The study of continual/lifelong learning dates back to [TM95] with the central goal of alleviating *catastrophic forgetting* [MC89, MMO95]. There is a surge of interest from both the machine learning and neuroscience communities in recent years. Most empirical approaches fall into three categories: regularization, replay, and dynamic architecture. The *regularization* approach penalizes the movement of hypothesis function across tasks. The elastic weight consolidation (EWC) [KPR+17] adds weighted ℓ_2 regularization to penalize the movement of neural network parameters. The orthogonal gradient descent (OGD) [FAML20, CKDT20] performs gradient descent orthogonal to previous directions. The *replay based* approach stores training samples from old tasks and rehearses during later task. The experience replay method [RAS+19] stores old experience for future use. Instead of explicitly storing data, [SLKK17] train a simulator of past data distributions using generative adversarial network (GAN). [vdVST20] propose a brain-inspired variant of replay method. The *dynamic architecture* approach gradually adds new components to the neural network. The progressive network approach [RRD+16] allocates new subnetworks for each new task while freezing parts of the existing component. We note that almost all current continual learning approaches suffer from a memory problem due to explicitly or implicitly retaining information from past experience. In [LPR17, CRRE18] the authors observe that allowing multiple passes over the sequence of tasks helps improve the accuracy. We refer the interested reader to the comprehensive survey of [PKP+19].

In contrast to the vast experimental literature, the theoretical investigation continual/lifelong learning is limited. Almost all existing theoretical analysis [RE13, PU16, BBV15, CLV21, PR22] fall into the *dynamic architecture* approach: They require known task identity at inference time and maintain task specific linear classifiers. [BBV15, CLV21] provide sample complexity guarantee for linear representation function. The work of [KHD20] is closer to ours, as they aim to quantify the memory requirement of continual learning. However, they only show a lower bound of $\Omega(db)$ for proper learning.

Communication complexity of learning. The study of the communication aspect of learning was initiated by the work of [BBFM12]; they apply the idea of boosting and prove that $O(db\log(1/\epsilon))$ bits of communication is sufficient for learning a uniform mixture of multiple tasks, in sharp contrast with the sample complexity lower bound of $\Omega(d/\epsilon)$. [KLMY19] study the two party communication problem of learning and prove a sample complexity lower bound. Their work is significantly different from ours as they consider an infinite data domain and data point transmission takes only one unit of communication. This makes their communication model *non-uniform* and they focus on sample complexity instead. In contrast, our work uses a finite data domain and takes into account the data description size. Communication complexity has also been examined for convex set disjointness [BKMS21] and optimization problems [VWW20].

In addition to these works, [DKS19] study space lower bound for linear predictor under streaming setting and prove a lower bound of $\Omega(d^2)$ for a proper learner. Other related work includes the time-space trade off [Raz18, GRT18, SSV19] and memorization [Fel20, BBF+21].

2 Preliminaries

PAC learning. We formulate the problem of continual learning through the PAC learning framework. Let $\mathcal{X}$ be the data universe. A hypothesis (or a function) is a mapping $h : \mathcal{X} \to \{0,1\}$, and a hypothesis class $\mathcal{H}$ contains a collection of functions. A sample is a pair from $\mathcal{X} \times \{0,1\}$ and a data distribution $\mathcal{D}$ is a distribution of samples, i.e., a distribution over $\mathcal{X} \times \{0,1\}$. Given a set S of n samples $\{(x_i, y_i)\}_{i\in[n]}$, the *empirical loss* of a function h with respect to S is defined as

$$\ell_S(h) = \frac{1}{|S|} \sum_{(x,y)\in S} \mathbf{1}[h(x) \neq y].$$

Given a data distribution $\mathcal{D}$, the *loss* of h with respect to $\mathcal{D}$ is is defined as

$$\ell_{\mathcal{D}}(h) = \Pr_{(x,y)\sim\mathcal{D}}[h(x) \neq y].$$

Definition 2.1 (PAC learning [Val84])**.** *Given a hypothesis class $\mathcal{H}$ over $\mathcal{X}$, an (ϵ, δ)-PAC learner $\mathcal{A}$ with sample complexity n is a learning algorithm that, given any data distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$, draws n samples $S \sim \mathcal{D}^n$ and outputs a function $h : \mathcal{X} \to \{0,1\}$ such that*

$$\Pr_{S\sim\mathcal{D}^n,\, h=\mathcal{A}(S)} \left[\ell_{\mathcal{D}}(h) \geq \min_{h^*\in\mathcal{H}} \ell_{\mathcal{D}}(h^*) + \epsilon\right] \leq \delta.$$

In the realizable *setting, we assume that there exists a hypothesis $h^* \in \mathcal{H}$ that satisfies $\ell_{\mathcal{D}}(h^*) = 0$. Hence an (ϵ, δ)-PAC learner satisfies that*

$$\Pr_{S\sim\mathcal{D}^n,\, h=A(S)} \left[\ell_{\mathcal{D}}(h) \geq \epsilon\right] \leq \delta.$$

We say a learner is proper *if it always outputs a hypothesis from the class $\mathcal{H}$; an* improper *learner is allowed to output an arbitrary mapping from $\mathcal{X}$ to $\{0,1\}$.*

The VC dimension measures the complexity of a function class and captures its learnability.

Definition 2.2 (VC dimension [VC15])**.** *Given a set of data $S = \{x_1, \ldots, x_d\} \subseteq \mathcal{X}$, we say the set S is shattered by function class $\mathcal{H}$ if $|\mathcal{H}(S)| = 2^d$, where $\mathcal{H}(S) := \{(h(x_1), \ldots, h(x_d)) : h \in \mathcal{H}\} \subseteq \{0,1\}^d$. The* VC dimension *of $\mathcal{H}$ is defined as the largest cardinality of sets shattered by $\mathcal{H}$.*

Lemma 2.3 (Sauer–Shelah Lemma [Sau72, She72])**.** *Let $\mathcal{H}$ be a hypothesis class with VC dimension d, then for any $S \subseteq \mathcal{X}$ with $|S| = n$, $|\mathcal{H}(S)| \leq \sum_{i=0}^{d} \binom{n}{i}$. In particular, $|\mathcal{H}(S)| \leq (en/d)^d$ if $n \geq d$.*

Lemma 2.4 (Learning with ERM [VC15])**.** *Let $\mathcal{H}$ be a hypothesis class with VC dimension d. Given any data distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$ that is realizable in $\mathcal{H}$, the ERM learner $\mathcal{A}(S) := \{h \in \mathcal{H} : \ell_S(h) = 0\}$ is an (ϵ, δ)-PAC learner with sample complexity*

$$n = O\left(\frac{d + \log(1/\delta)}{\epsilon}\right).$$

Continual learning. We next define the continual learning problem.

Definition 2.5 (Continual learning)**.** *Let $\mathcal{H}$ be a hypothesis class with VC dimension d over a data universe $\mathcal{X}$ of description size $b := \lceil \log_2 |\mathcal{X}| \rceil$. In the continual learning problem over $\mathcal{H}$, a learner has sequential access to k tasks, specified by k data distribution $\mathcal{D}_1, \ldots, \mathcal{D}_k$ over $\mathcal{X} \times \{0,1\}$. We will focus on the* realizable *setting: there exists a hypothesis $h \in \mathcal{H}$ such that $\ell_{\mathcal{D}_i}(h) = 0$ for all $i \in [k]$.*

A continual learner $\mathcal{A}$ with m bits of memory is initiated with a memory state $M_0 \in \{0,1\}^m$ and the learning process is divided into k stages. During the i-th stage, $\mathcal{A}$ can draw as many samples as it wants from $\mathcal{D}_i$ and use them to update the memory state from M_{i-1} to $M_i \in \{0,1\}^m$:

$$S_i \sim \mathcal{D}_i^* \quad \text{and} \quad M_i = \mathcal{A}(M_{i-1}, S_i).$$

At the end, $\mathcal{A}$ recovers a function $h : \mathcal{X} \to \{0,1\}$ from M_k. We say $\mathcal{A}$ is an (ϵ, δ)-continual learner for $\mathcal{H}$ over k tasks if, with probability at least $1-\delta$, the function h satisfies $\ell_{\mathcal{D}_i}(h) \geq \epsilon$ for all $i \in [k]$. Similar to the standard PAC model, a learner is proper if it always outputs a hypothesis $h \in \mathcal{H}$. We are interested in understanding the memory requirement m as a function of k, d, b and ϵ for both proper and improper learners.

A c-pass continual learner is defined similarly, except that it can now make c sequential passes on $\mathcal{D}_1, \ldots, \mathcal{D}_k$. We are interested in understanding m as a function of c, k, d, b and ϵ.

Remark 2.6 (Sample complexity)**.** *Our lower bounds for continual learning presented in Theorem 1.1 and 1.3 are for memory requirements (which make them stronger); our algorithm in Theorem 1.2, on the other hand, has sample complexity comparable with its memory requirement.*

Remark 2.7 (Representation issue and bit complexity)**.** *Our memory bounds are measured in terms of bit complexity. We assume a finite data universe $\mathcal{X}$ in which each data point can be written as $b = \lceil \log_2 |\mathcal{X}| \rceil$ bits. Due to the Sauer–Shelah Lemma, the total number of hypotheses in $\mathcal{H}$ is $2^{O(db)}$ and thus, writing down a hypothesis in $\mathcal{H}$ requires at most $O(db)$ bits.*

Notation We write $[k] = \{1, 2, \ldots, k\}$ and $[a : b] = \{a, a+1, \ldots, b\}$. We write $(i, j) \prec (i', j')$ if (1) $i < i'$ or (2) $i = i'$ and $j < j'$. For any $x, y, z \in \mathbb{R}$, $x = y \pm z$ means $x \in [y - z, y + z]$. The natural logarithm of x is generally written as $\log x$, and the base 2 logarithm is written as $\log_2 x$. For any two random variables A, B, we use $\|A - B\|_{\mathsf{TV}}$ to denote the total variation distance, $\mathsf{KL}(A \parallel B)$ to denote the KL divergence between A, B and $I(A; B)$ to denote the mutual information. The entropy function is written as $H(A)$. For any set S, we write $a \sim S$ if the element a is drawn uniformly at random from S.

Prerequisites from information theory. We provide some basic facts in information theory. The proof of Fact 2.9 is provided in the Appendix A for completeness.

Fact 2.8. *If $A_1, \ldots, A_n$ are totally independent, then*

$$\sum_{i \in [n]} I(A_i; B) \leq I(A_1, \ldots, A_n; B).$$

Fact 2.9. *For any random variables $A_1, \ldots, A_n, B$, we have*

$$\sum_{i \mathbf{i} [n]} I(A_i; B) \leq I(A_1, \ldots, A_n; B) + \sum_{i \in [n]} H(A_i) - H(A_1, \ldots, A_n).$$

Fact 2.10. *If $A_1, \ldots, A_n$ are totally independent when conditioning on C, then*

$$\sum_{i \in [n]} I(A_i; B|C) \leq I(A_1, \ldots, A_n; B|C).$$

Fact 2.11. *For random variables A, B, C, D, if B, D are independent when conditioned on C, then*

$$I(A; B|C) \leq I(A; B|C, D).$$

We also need the following compression lemma from [JPY12].

Lemma 2.12 (Lemma 3.4 in [JPY12], taking $t = 1$)**.** *Let X, Y, M, R be random variables such that*

$$I(X; M|Y, R) \le c \quad \textit{and} \quad I(Y; M|X, R) = 0.$$

There exists a one-way public-coin protocol between Alice and Bob, with inputs X and Y respectively, such that Alice sends a single message of size at most

$$\frac{c+5}{\epsilon'} + O\left(\log \frac{1}{\epsilon'}\right)$$

to Bob and at the end of the protocol, Alice and Bob both hold a random variable M' such that

$$\left\|(X, Y, M, R) - (X, Y, M', R)\right\|_{\mathsf{TV}} \le 6\epsilon'.$$

3 Lower bounds for continual learning

We restate our memory lower bounds for continual learning.

Theorem 1.1. *There exists a sufficiently large constant C_0 such that the following holds. For any $\epsilon \in (0, 0.01]$ and any positive integers k, d and b with $b \ge C_0 \log(kd/\epsilon)$, there is a hypothesis class $\mathcal{H}$ of VC dimension $2d$ over a universe $\mathcal{X}$ of description size b such that any $(\epsilon, 0.01)$-continual learner for $\mathcal{H}$ over a sequence of $k+1$ tasks requires $\Omega(kdb/\epsilon)$ bits of memory.*

Our plan for the proof of Theorem 1.1 is as follows. In Section 3.1 we first introduce a three-party one-way (distributional) communication complexity problem parameterized by two positive integers n, d and a prime number p, where we use $f_{n,d,p}$ to denote the Boolean-valued function that the three parties would like to evaluate and $\mu_{n,d,p}$ to denote the (joint) distribution of their inputs. Given k, d, b and ϵ that satisfy conditions of Theorem 1.1, we construct a hypothesis class $\mathcal{H} = \mathcal{H}_{k,d,b,\epsilon}$ with VC dimension $2d + 1$ and description size b in Section 3.2, and show via a simple reduction that the memory requirement of any $(\epsilon, 0.01)$-continual learner for $\mathcal{H}$ over $k + 1$ tasks is at least the communication complexity of $f_{n,d,p}$ over $\mu_{n,d,p}$ with (roughly) $n \approx k/\epsilon$ and $p \approx 2^{b/2}$. Finally we prove that the communication complexity of latter is at least $\Omega(\min(\sqrt{p}, nd \log_2 p))$ in the rest of the section, from which Theorem 1.1 follows directly.

3.1 Communication problem

Let n, d be two positive integers and p be a prime. We will consider p to be sufficiently large (given that we will set p to be roughly $2^{b/2}$ in the reduction and that C_0 can be chosen to be a sufficiently large constant in Theorem 1.1). Let $t \coloneqq \lfloor 0.2p \rfloor$. The communication problem we are interested in is one-way, distributional, and has three players Alice, Bob and Charlie. We start by describing the probability distribution $\mu_{n,d,p}$ over inputs X, Y and Z of Alice, Bob and Charlie, respectively.

We follow the following procedure to draw $(X, Y, Z) \sim \mu_{n,d,p}$:

1. For each $i \in [n]$ and $j \in [d]$, we draw $a_{i,j} = (a_{i,j,1}, a_{i,j,2})$ from $[p] \times [p]$ independently and uniformly at random. Each $a_{i,j}$ defines a *line* in $[p] \times [p]$, denoted by $L(a_{i,j}) \subset [p] \times [p]$, which contains all points $(r_1, r_2) \in [p] \times [p]$ such that $a_{i,j,1} r_1 + r_2 \equiv a_{i,j,2} \pmod p$.

2. Next for each $i \in [n]$ and $j \in [d]$, we draw a size-t subset $A_{i,j}$ of $[p] \times [p]$ independently and uniformly at random but conditioning on $a_{i,j} \in A_{i,j}$. We write $L(A_{i,j}) \subset [p] \times [p]$ to denote the union of these t lines: $L(A_{i,j}) \coloneqq \cup_{a \in A_{i,j}} L(a)$.

3. Finally we draw $i^* \in [n]$, $j^* \in [d]$ and a secret bit $s^* \in \{0,1\}$ independently and uniformly at random. If $s^* = 0$, we draw a point $r^* = (r_1^*, r_2^*)$ from $[p] \times [p] \setminus L(A_{i^*,j^*})$ uniformly at random; if $s^* = 1$, we draw r^* from $L(a_{i^*,j^*})$ uniformly at random. Note that possible choices of r^* are disjoint in these two cases given that $a_{i^*,j^*} \in A_{i^*,j^*}$.

4. The input X of Alice is $(a_{i,j} : i \in [n], j \in [d])$.
   The input Y of Bob consists of $(A_{i,j} : i \in [n], j \in [d])$ and i^*.
   The input Z of Charlie consists of i^*, j^* and r^*.
   The goal is for Charlie to output a bit that matches s^*.

Equivalently we can define a Boolean-valued function $f_{n,d,p}(X, Y, Z)$ that is set to be 0 by default if (X, Y, Z) is not in the support of $\mu_{n,d,p}$ and is set to be s^* if (X, Y, Z) is in the support of $\mu_{n,d,p}$ (note that s^* can be uniquely determined by (X, Y, Z) given that $a_{i^*,j^*} \in A_{i^*,j^*}$). Thus the problem we are interested in is the distributional communication complexity about $f_{n,d,p}$ over $\mu_{n,d,p}$.

An (m_1, m_2)-bit one-way deterministic communication protocol is a triple (Π_1, Π_2, Π_3) of functions, where (1) $\Pi_1(X) \in \{0,1\}^{m_1}$ is the message from Alice to Bob, (2) $\Pi_2(Y, \Pi_1) \in \{0,1\}^{m_2}$ is the message from Bob to Charlie, and (3) $\Pi_3(Z, \Pi_2) \in \{0,1\}$ is Charlie's output.

Our main technical result in this section is the following lower bound.

Theorem 3.1. *Let n, d be two positive integers and p be a sufficiently large prime. Any (m_1, m_2)-bit protocol that fails in computing $f_{n,d,p}$ over $\mu_{n,d,p}$ with probability at most $1/40$ must have*

$$m_1 = \Omega(nd \log_2 p) \quad \text{or} \quad m_2 \geq \sqrt{p}.$$

We delay the proof of the theorem to Section 3.3.

3.2 Hypothesis class

Let k, d, b and ϵ be parameters that satisfy conditions of Theorem 1.1. Let

$$n := k \cdot \left\lfloor \frac{1}{100\epsilon} \right\rfloor = \Theta\left(\frac{k}{\epsilon}\right)$$

using $\epsilon \leq 0.01$, and let p be the largest prime with

$$p \leq \frac{2^{b/2}}{\sqrt{nd}} \quad \text{so that} \quad p = \Theta\left(\frac{2^{b/2}}{\sqrt{nd}}\right)$$

satisfies $\log_2 p = \Theta(b)$ given that $b \geq C_0 \log(kd/\epsilon)$ for some sufficiently large C_0 in Theorem 1.1.

We describe the hypothesis class $\mathcal{H} = \mathcal{H}_{k,d,b,\epsilon}$ below. We start with the data universe $\mathcal{X}$ of $\mathcal{H}$.

Data universe. Let $\mathcal{X} = [n] \times [d] \times ([p] \times [p])$. The description size of $\mathcal{X}$ is no more than b by our choice of p. For any data point $x \in \mathcal{X}$, we write $x = (x_1, x_2, x_3)$ where $x_1 \in [n]$, $x_2 \in [d]$ and $x_3 = (x_{3,1}, x_{3,2}) \in [p] \times [p]$. For any $i \in [n], j \in [d]$, we refer to $\mathcal{X}_i = \{x \in \mathcal{X} : x_1 = i\}$ as the i-th *block* and $\mathcal{X}_{i,j} = \{x \in \mathcal{X} : x_1 = i, x_2 = j\}$ as the j-th *cell* of the i-th block.

Hypothesis class. Each function h in the hypothesis class $\mathcal{H}$ is specified by an index $i \in [n]$ and a tuple $(a_1, \ldots, a_d) \in ([p] \times [p])^d$ such that $h : \mathcal{X} \to \{0,1\}$ is defined as

$$h(x) = \begin{cases} 1 & \text{if } x_1 \neq i \\ 1 & \text{if } x_1 = i \text{ and } x_3 \in L(a_{x_2}) \\ 0 & \text{if } x_1 = i \text{ and } x_3 \notin L(a_{x_2}) \end{cases}$$

In short, $h \in \mathcal{H}$ has label 1 everywhere except in the i-th block $\mathcal{X}_i$, where each cell $\mathcal{X}_{i,j}$ ($j \in [d]$) is all 0 except for points on the line $L(a_j)$.

A quick observation is that $\mathcal{H}$ has VC dimension at most $2d$.

Lemma 3.2. *The VC dimension of $\mathcal{H}$ is at most $2d$.*

Proof. For any subset $S = \{x^{(1)}, \ldots, x^{(2d+1)}\}$ of size $2d+1$, we prove $\mathcal{H}$ can not shatter S. First, if there exists two indices $i_1, i_2 \in [2d+1]$ such that $x_1^{(i_1)} \neq x_1^{(i_2)}$, then $\mathcal{H}$ can not shatter $(0,0)$ on $(x^{(i_1)}, x^{(i_2)})$. On the other hand, suppose $x_1^{(1)} = \ldots = x_1^{(2d+1)} = 1$ for simplicity, then there must exists $i_1, i_2, i_3 \in [2d+1]$ such that $x^{(i_1)}, x^{(i_2)}, x^{(i_3)}$ satisfy $x_2^{(i_1)} = x_2^{(i_2)} = x_2^{(i_3)}$, that is, there are three data points in the same cell. (1) If $(x_{3,1}^{(i_1)}, x_{3,2}^{(i_1)}), (x_{3,1}^{(i_2)}, x_{3,2}^{(i_2)}), (x_{3,1}^{(i_3)}, x_{3,2}^{(i_3)})$ are on a same line of $([p] \times [p])$, then $\mathcal{H}$ can not shatter $(1,1,0)$ on $(x^{(i_1)}, x^{(i_2)}, x^{(i_3)})$. (2) If they are not on the same line, then pick any index $i_4 \in [2d+1]$, we can assume $x^{(i_4)}$ is not on the same cell and same line with any two of $x^{(i_1)}, x^{(i_2)}, x^{(i_3)}$ WLOG. Then $\mathcal{H}$ can not shatter $(1,1,1,0)$ on $(x^{(i_1)}, x^{(i_2)}, x^{(i_3)}, x^{(i_4)})$. $\square$

Via a simple reduction, we show below that communication complexity lower bounds for $f_{n,d,p}$ over $\mu_{n,d,p}$ can be used to obtain memory lower bounds for the continual learning problem.

Lemma 3.3 (Reduction)**.** *If there is an (ϵ, δ)-continual learner with an m-bit memory for $\mathcal{H}_{k,d,b,\epsilon}$ over $k+1$ tasks, then there is an (m,m)-bit protocol that fails in computing $f_{n,d,p}$ over $\mu_{n,d,p}$ with probability at most $0.01 + \delta$.*

Proof. Fix any pair of inputs X and Y of Alice and Bob, respectively, where X contains $(a_{i,j} : i \in [n], j \in [d])$ and Y consists of i^* and $(A_{i,j} : i \in [n], j \in [d])$. Conditioning on X and Y, Charlie's input Z contains i^*, j^* and r^*, where j^* is uniform over $[d]$ and r^* with probability $1/2$ is uniform over $L(a_{i^*,j^*})$ and with probability $1/2$ is uniform over $[p] \times [p] \setminus L(A_{i^*,j^*})$. We describe below a communication protocol such that Charlie fails with probability no more than $0.01 + \delta$.

Let $\mathcal{X}$ be the data universe of $\mathcal{H}_{k,d,b,\epsilon}$. Given their inputs X and Y, respectively, Alice uses X to a construct a sequence of k data distributions $\mathcal{D}_1, \ldots, \mathcal{D}_k$ and Bob uses Y to construct one data distribution $\mathcal{D}_{k+1}$ over $\mathcal{X} \times \{0,1\}$. We first describe these distributions and then show that they are consistent with a function $h \in \mathcal{H}_{k,d,b,\epsilon}$:

1. Let $\alpha := \lceil 1/(100\epsilon) \rceil$. For each $i \in [k]$, $\mathcal{D}_i$ is a uniform distribution over data points (x_1, x_2, x_3), where $x_1 = (i-1)\alpha + 1, \ldots, i\alpha$, $x_2 \in [d]$ and $x_3 \in L(a_{x_1,x_2})$ (so there are αdp points in total in the support of each $\mathcal{D}_i$, each with probability $1/(\alpha dp)$). All points in $\mathcal{D}_i$ are labelled 1. Alice can construct all these k distributions using her input X only.

2. Bob constructs a data distribution $\mathcal{D}_{k+1}$ that is uniform over (x_1, x_2, x_3), where $x_1 = i^*$, $x_2 \in [d]$ and $x_3 \in ([p] \times [p]) \setminus L(A_{i^*,x_2})$, and all points are labelled 0. Given that Bob has i^* and all of $A_{i^*,j}$ in Y, Bob can construct $\mathcal{D}_{k+1}$ using his input Y only.

3. It is clear that these data distributions are consistent with a function $h \in \mathcal{H}_{k,d,b,\epsilon}$ that is specified by i^* and $(a_{i^*,1}, \ldots, a_{i^*,d})$.

The communication protocol has Alice and Bob simulate the (ϵ, δ)-continual learner for $\mathcal{H}_{k,d,b,\epsilon}$ where Alice passes the memory of the learner after going through $\mathcal{D}_1, \ldots, \mathcal{D}_k$ to Bob and Bob finally passes the memory to Charlie after going through $\mathcal{D}_{k+1}$. So both messages of Alice and Bob have m-bits and with probability at least $1 - \delta$, Charlie has in hand a function $h^* : \mathcal{X} \to \{0,1\}$ such that $\ell_{\mathcal{D}}(h^*) \leq \epsilon$ for all $\mathcal{D}$ in $\mathcal{D}_1, \ldots, \mathcal{D}_{k+1}$. We finish the proof by showing that when this event happens, Charlie fails with probability no more than 0.01 (over randomness of Z conditioning on X and Y).

First, let i' be the integer such that $(i'-1)\alpha+1 \le i^* \le i'\alpha$. Then the number of data points $x=(x_1,x_2,x_3)$ with $x_1=i^*$, $x_2\in[d]$, $x_3\in L(a_{x_1,x_2})$ and $h^*(x)\ne h(x)$ is no more than $\epsilon\cdot\alpha dp$ using $\ell_{\mathcal{D}_{i'}}(h^*)\le\epsilon$. Thus, Charlie fails because of one of these points with probability at most

$$\frac{1}{2}\cdot\frac{\epsilon\alpha dp}{dp}\le\frac{0.01}{2}$$

using $\alpha\le 1/(100\epsilon)$. The other error $0.01/2$ comes from that $\ell_{\mathcal{D}_{k+1}}(h^*)\le\epsilon\le 0.01$. $\square$

We are now ready to prove Theorem 1.1 (assuming Theorem 3.1 for now):

Proof of Theorem 1.1. Given an $(\epsilon,0.01)$-continual learner with an m-bit memory for $\mathcal{H}_{k,d,b,\epsilon}$ over $k+1$ tasks, by Lemma 3.3, there is an (m,m)-bit protocol that fails in computing $f_{n,d,p}$ over $\mu_{n,d,p}$ with probability at most $0.02<\frac{1}{40}$. Hence by Theorem 3.1, we conclude $m\ge\Omega(\min\{nd\log_2 p,\sqrt{p}\})=\Omega(kdb/\epsilon)$. $\square$

3.3 Communication lower bound

We prove the communication lower bound in Theorem 3.1. The proof proceeds as follows. First we prove a lower bound for the communication problem over one single cell in Section 3.3.1. Then we follow a standard direct sum argument to extend it to a lower bound for $f_{n,d,p}$ over $\mu_{n,d,p}$ in Section 3.3.2.

3.3.1 Communication problem over one cell

Let p be a sufficiently large prime. We first focus on the following three-party one-way distributional communication problem over one single cell $[p]\times[p]$. The inputs (a^*,A^*,r^*) of the three players Alice, Bob and Charlie are drawn from a distribution μ_p as follows: (1) Alice's input a^* is drawn from $[p]\times[p]$ uniformly at random, which defines a line $L(a^*)$ of p points; (2) Bob's input A^* is a size-t subset of $[p]\times[p]$ drawn uniformly at random conditioning on $a^*\in A^*$, and we write $L(A^*)$ to denote the union of $L(a)$ over $a\in A^*$; (3) With probability 1/2, Charlie's input $r^*\in[p]\times[p]$ is drawn uniformly at random from $L(a^*)$; with probability 1/2, r^* is drawn uniformly at random from $([p]\times[p])\setminus L(A^*)$. The function $f_p(a^*,A^*,r^*)$ that Charlie needs to compute is set to be 1 if $r^*\in L(a)$ and 0 if $r^*\in([p]\times[p])\setminus L(A^*)$.

Our goal is to prove the following lower bound for computing f_p over μ_p:

Lemma 3.4. *Let $\Pi=(\Pi_1,\Pi_2,\Pi_3)$ be an (m_1,m_2)-bit one-way deterministic communication protocol. If $m_1\le 0.05\log_2 p$ and $m_2\le\sqrt{p}$, then Charlie has error probability at least $9/200$.*

In the proof we use Π_1 (or Π_2) to denote the random variable as the m_1-bit (or m_2-bit) message from Alice to Bob (or from Bob to Charlie, respectively). Lemma 3.4 is a direct corollary of Lemma 3.5 and Lemma 3.6 below. In Lemma 3.5 we show that our construction makes sure that in order for Charlie to succeed with high probability, he must receive at least $(1-o(1))\log_2 p$ bits of information regarding the hidden line a from Π_2. Formally, we have

Lemma 3.5. *For any communication protocol $\Pi=(\Pi_1,\Pi_2,\Pi_3)$, we have*

$$\Pr_{(a^*,A^*,r^*)\sim\mu_p}\Big[\Pi \textit{ succeeds on } (a^*,A^*,r^*)\Big]\le\frac{19}{20}+\frac{1}{20}\cdot\frac{I(a^*;\Pi_2)+1}{\log_2 p}. \tag{1}$$

We next prove the mutual information $I(a^*;\Pi_2)$ is small for any deterministic communication protocol Π with small amount of communication. Formally

Lemma 3.6. *Let $\Pi = (\Pi_1, \Pi_2, \Pi_3)$ be an (m_1, m_2)-bit one-way deterministic communication protocol. If $m_1 \le 0.05 \log_2 p$ and $m_2 \le \sqrt{p}$, then we have $I(a^*; \Pi_2) < 0.08 \log_2 p$.*

Combining Lemma 3.5 and Lemma 3.6, we conclude the proof of Lemma 3.4.

Proof of Lemma 3.5. Fixing a message $\Pi_2 = \widehat{\Pi}_2$ from Bob, we upper bound the probability that Π succeeds on $(a^*, A^*, r^*) \sim \mu_p$ conditioning on $\Pi_2 = \widehat{\Pi}_2$. For convenience, we write $\mu_{p,\widehat{\Pi}_2}$ to denote this conditional distribution.

With $\widehat{\Pi}_2$ fixed, we can assume that Charlie uses a subset of *points* $H \subseteq [p] \times [p]$ to finish the protocol, i.e., after receiving $\widehat{\Pi}_2$ from Bob, Charlie returns 1 on input r if $r \in H$ and returns 0 if $r \notin H$. We further define a set of *lines* S from H: $a \in [p] \times [p]$ is in S iff at least 90% of points in $L(a)$ lie in H. We divide into two cases:

Case 1: $|S| \ge p$. Let $a_1, \ldots, a_p$ be any p lines in S. We have

$$\left(\bigcup_{i\in[p]} L(a_i)\right) \bigcap H \ge \sum_{i\in[p]} \left| \left(L(a_i) \cap H\right) \setminus \bigcup_{j\in[i-1]} L(a_j) \right| \ge \sum_{i\in[p]} \max\left(0.9p - (i-1), 0\right) > 0.4p^2.$$

On the other hand, fixing any inputs $\widehat{a}$ and $\widehat{A}$ of Alice and Bob such that Bob sends $\widehat{\Pi}_2$, we have

$$\left| ([p] \times [p]) \setminus L(\widehat{A}) \right| \ge p^2 - pt = 0.8p^2$$

and the correct answer for all these points are 0. As a result, we have that when $(a^*, A^*, r^*) \sim \mu_p$ conditioning on $a^* = \widehat{a}$ and $A^* = \widehat{A}$, the probability that Π errors is at least

$$\frac{1}{2} \cdot \frac{0.8p^2 + 0.4p^2 - p^2}{p^2} = 0.1.$$

This implies that Π succeeds on $(a^*, A^*, r^*) \sim \mu_{p,\widehat{\Pi}_2}$ with probability at most 0.9.

Case 2: $|S| < p$. We show that the probability that Π succeeds on $(a^*, A^*, r^*) \sim \mu_{p,\widehat{\Pi}_2}$ is at most

$$\frac{19}{20} + \frac{1}{20} \cdot \frac{2\log_2 p - H(a^* \mid \Pi_2 = \widehat{\Pi}_2) + 1}{\log_2 p}. \tag{2}$$

To see this, we examine the distribution of a^* conditioning on $\Pi_2 = \widehat{\Pi}_2$ and write q to denote the probability that $a^* \in S$. Using $|S| \le p$, we have

$$\begin{aligned} H(a^* \mid \Pi_2 = \widehat{\Pi}_2) &\le q \cdot \log_2\left(\frac{p}{q}\right) + (1-q) \cdot \log_2\left(\frac{p^2 - p}{1-q}\right) \\ &\le q \cdot \log_2\left(\frac{1}{q}\right) + (1-q) \cdot \log_2\left(\frac{1}{1-q}\right) + q\log_2 p + (1-q)\log_2 p^2 \\ &\le 1 + (2-q) \cdot \log_2 p. \end{aligned}$$

Hence, we have

$$q \le \frac{2\log_2 p - H(a^* \mid \Pi_2 = \widehat{\Pi}_2) + 1}{\log_2 p}. \tag{3}$$

Given that every line not in S has at least 10% of points not in H, Π errors on $(a^*, A^*, r^*) \sim \mu_{p,\widehat{\Pi}_2}$ with probability at least

$$\frac{1}{2} \cdot (1-q) \cdot \frac{1}{10} \geq \frac{1}{20} - \frac{1}{20} \cdot \frac{2\log_2 p - H(a^* \mid \Pi_2 = \widehat{\Pi}_2) + 1}{\log_2 p}$$

from which (2) follows.

Combining the two cases (and noting that $0.9 < 19/20$), we have

$$\Pr_{(a^*,A^*,r^*)\sim\mu_{p,\widehat{\Pi}_2}}\left[\Pi \text{ succeeds on } (a^*, A^*, r^*)\right] \leq \frac{19}{20} + \frac{1}{20} \cdot \frac{2\log_2 p - H(a^* \mid \Pi_2 = \widehat{\Pi}_2) + 1}{\log_2 p}.$$

As a result, using $H(a^*) = 2\log_2 p$ we have

$$\begin{aligned}
&\Pr_{(a^*,A^*,r^*)\sim\mu_p}\left[\Pi \text{ succeeds on } (a^*, A^*, r^*)\right] \\
&= \sum_{\widehat{\Pi}_2} \Pr\left[\Pi_2 = \widehat{\Pi}_2\right] \cdot \Pr_{(a^*,A^*,r^*)\sim\mu_{p,\widehat{\Pi}_2}}\left[\Pi \text{ succeeds on } (a^*, A^*, r^*)\right] \\
&\leq \sum_{\widehat{\Pi}_2} \Pr\left[\Pi_2 = \widehat{\Pi}_2\right] \cdot \left(\frac{19}{20} + \frac{1}{20} \cdot \frac{2\log_2 p - H(a^* \mid \Pi_2 = \widehat{\Pi}_2) + 1}{\log_2 p}\right) \\
&= \frac{19}{20} + \frac{1}{20} \cdot \frac{I(a^*; \Pi_2) + 1}{\log_2 p}.
\end{aligned}$$

We conclude the proof. $\square$

Proof of Lemma 3.6. For convenience, we denote $m_1 = \gamma \log_2 p$ with $\gamma \leq 0.05$. For any realization of Alice's message $\Pi_1 = \widehat{\Pi}_1$, define

$$\mathsf{Img}(\widehat{\Pi}_1) = \left\{a \in [p] \times [p] : \Pi_1(a) = \widehat{\Pi}_A\right\}.$$

Next we write Π_{large} to denote the set of $\widehat{\Pi}_1$ such that $\mathsf{Img}(\widehat{\Pi}_1)$ is large:

$$\Pi_{\mathsf{large}} = \left\{\widehat{\Pi}_1 : \left|\mathsf{Img}(\widehat{\Pi}_1)\right| \geq \frac{p^{2-\gamma}}{100}\right\}.$$

First we show that to bound $I(a^*; \Pi_2)$, it suffices to bound $I(a^*; \Pi_2 \mid \Pi_1 = \widehat{\Pi}_1)$ for $\widehat{\Pi}_1 \in \Pi_{\mathsf{large}}$:

$$\begin{aligned}
I(a^*; \Pi_2) &\leq I(a^*; \Pi_1\Pi_2) \\
&\leq I(a^*; \Pi_2 \mid \Pi_1) + \gamma \log_2 p \\
&= \sum_{\widehat{\Pi}_1 \in \Pi_{\mathsf{large}}} \Pr\left[\Pi_1 = \widehat{\Pi}_1\right] \cdot I(a^*; \Pi_2 \mid \Pi_1 = \widehat{\Pi}_1) \\
&\qquad + \sum_{\widehat{\Pi}_1 \notin \Pi_{\mathsf{large}}} \Pr\left[\Pi_1 = \widehat{\Pi}_1\right] \cdot I(a^*; \Pi_2 \mid \Pi_1 = \widehat{\Pi}_1) + \gamma \log_2 p \\
&\leq \sum_{\widehat{\Pi}_1 \in \Pi_{\mathsf{large}}} \Pr\left[\Pi_1 = \widehat{\Pi}_1\right] \cdot I(a^*; \Pi_2 \mid \Pi_1 = \widehat{\Pi}_1) + \frac{1}{p^2} \cdot \frac{p^{2-\gamma}}{100} \cdot p^{\gamma} \cdot 2\log_2 p + \gamma \log_2 p \\
&= \sum_{\widehat{\Pi}_1 \in \Pi_{\mathsf{large}}} \Pr\left[\Pi_1 = \widehat{\Pi}_1\right] \cdot I(a^*; \Pi_2 \mid \Pi_1 = \widehat{\Pi}_1) + (\gamma + 0.02) \log_2 p. \qquad (4)
\end{aligned}$$

It suffices to show that for any message $\widehat{\Pi}_1 \in \Pi_{\mathsf{large}}$, we have

$$I(a^*; \Pi_2 \mid \Pi_1 = \widehat{\Pi}_1) = o_p(1). \tag{5}$$

Fixing a message $\widehat{\Pi}_1 \in \Pi_{\mathsf{large}}$, let's take a pause and review the distribution of $(a^*, A^*, r^*) \sim \mu_p$ conditioning on $\Pi_1 = \widehat{\Pi}_1$. In particular, let's denote the distribution of (a^*, A^*) as $\mu_{p,\widehat{\Pi}_1}$ because we don't care about r^* in this lemma; $(a^*, A^*) \sim \mu_{p,\widehat{\Pi}_1}$ is drawn as follows:

1. Sample a line a^* from $\mathsf{Img}(\widehat{\Pi}_1)$ uniformly at random;
2. Sample a set of $t-1$ lines from $[p] \times [p] \setminus \{a^*\}$ uniformly at random and add a^* to form A^*.

Let $S^* = A^* \cap \mathsf{Img}(\widehat{\Pi}_1)$ and $T^* = A^* \setminus S^*$. We prove that $|S^*|$ is large with high probability, when $(a^*, A^*) \sim \mu_{p,\widehat{\Pi}_1}$. Formally, after drawing a^*, we draw a sequence of $t-1$ lines from $[p] \times [p] \setminus \{a^*\}$ without replacements. The probability that the ith line is in $\mathsf{Img}(\widehat{\Pi}_1)$ is at least

$$\frac{0.01 \cdot p^{2-\gamma} - i}{p^2} \geq \frac{p^{-\gamma}}{101}$$

using $t = 0.2p$. Using Chernoff bound, we have

$$\Pr_{(a^*,A^*)\sim\mu_{p,\widehat{\Pi}_1}}\left[|S^*| < \frac{p^{1-\gamma}}{200}\right] \leq \exp(-\Omega(p^{1-\gamma})). \tag{6}$$

Denote $C = p^{1-\gamma}/200$. We show that to prove (5), it suffices to prove that

$$I\big(a^*; \Pi_2 \mid \Pi_1 = \widehat{\Pi}_1, |S^*| = c\big) = o_p(1) \tag{7}$$

for every $c \geq C$. This is because

$$\begin{aligned}
I(a^*; \Pi_2 \mid \Pi_1 = \widehat{\Pi}_1) &\leq I(a^*; \Pi_2 \mid \Pi_1 = \widehat{\Pi}_1, |S^*|) \\
&= \sum_{c \geq C} \Pr\Big[|S^*| = c \mid \Pi_1 = \widehat{\Pi}_1\Big] \cdot I\big(a^*; \Pi_2 \mid \Pi_1 = \widehat{\Pi}_1, |S^*| = c\big) \\
&\qquad + \sum_{c < C} \Pr\Big[|S^*| = c \mid \Pi_1 = \widehat{\Pi}_1\Big] \cdot I\big(a^*; \Pi_1 \mid \Pi_1 = \widehat{\Pi}_1, |S^*| = c\big) \\
&\leq \sum_{c \geq C} \Pr\Big[|S^*| = c \mid \Pi_1 = \widehat{\Pi}_1\Big] \cdot I\big(a^*; \Pi_2 \mid \Pi_1 = \widehat{\Pi}_1, |S^*| = c\big) + o(1).
\end{aligned} \tag{8}$$

The first step holds as $|S^*|$ is independent of a^* when conditioning on $\Pi_1 = \widehat{\Pi}_1$ (see Fact 2.11). The last step follows from (6) and that a^* takes values in $[p] \times [p]$.

For a fixed $\widehat{\Pi}_1 \in \Pi_{\mathsf{large}}$ and a fixed $c \geq C$, we prove (7) in the rest of the proof. Let's again take a pause and think about the distribution of a^*, S^*, T^* and Π_2 when conditioning on $\Pi_1 = \widehat{\Pi}_1$ and $|S^*| = c$. Equivalently, a^*, S^*, T^* and Π_2 can be drawn as follows:

1. Draw a sequence of c lines $a_1, \ldots, a_c$ uniformly from $\mathsf{Img}(\widehat{\Pi}_1)$ without replacements;
2. Draw a subset T^* of size $t - c$ from $([p] \times [p]) \setminus \mathsf{Img}(\widehat{\Pi}_1)$ uniformly at random;
3. Draw $i^* \in [c]$ uniformly at random; and
4. Set $a^* = a_{i^*}$, $S^* = \{a_1, \ldots, a_c\}$ and $\Pi_2 = \Pi_2(S^* \cup T^*, \widehat{\Pi}_1)$.

From this description we have that

$$
\begin{aligned}
I\big(a^*;\Pi_2 \mid \Pi_1=\widehat{\Pi}_1,|S^*|=c\big) &\le I\big(a^*;\Pi_2 \mid \Pi_1=\widehat{\Pi}_1,|S^*|=c,i^*\big)\\
&= \frac{1}{c}\sum_{i\in[c]} I\big(a_i;\Pi_2 \mid \Pi_1=\widehat{\Pi}_1,|S^*|=c,i^*=i\big)\\
&= \frac{1}{c}\sum_{i\in[c]} I\big(a_i;\Pi_2 \mid \Pi_1=\widehat{\Pi}_1,|S^*|=c\big) \qquad (9)
\end{aligned}
$$

The first step holds since conditioning on $\Pi_1=\widehat{\Pi}_1$ and $|S^*|=c$, the index i^* and the line a^* are independent (no matter what i^* is, a^* is always uniform over $\mathsf{Img}(\widehat{\Pi}_1)$); the last step holds since i^* is independent of $(a_1,\ldots,a_c,\Pi_2)$ so whether knowing $i^*=i$ or not does not affect the distribution of (a_i,Π_2).

Let $N=|\mathsf{Img}(\widehat{\Pi}_1)|\ge p^{2-\gamma}/100$. To bound the RHS of (9), we have

$$
\begin{aligned}
&\sum_{i\in[c]} I\big(a_i;\Pi_2 \mid \Pi_1=\widehat{\Pi}_1,|S^*|=c\big)\\
&\quad\le I\big(a_1\ldots a_c;\Pi_2 \mid \Pi_1=\widehat{\Pi}_1,|S^*|=c\big)\\
&\qquad\quad+\sum_{i\in[c]} H\big(a_i \mid \Pi_1=\widehat{\Pi}_1,|S^*|=c\big)-H\big(a_1\ldots a_c \mid \Pi_1=\widehat{\Pi}_1,|S^*|=c\big)\\
&\quad\le m_2+c\cdot\log_2 N-\sum_{i\in[c]}\log_2(N-i+1)\\
&\quad= m_2+\sum_{i\in[c]}\log_2\left(\frac{N}{N-i+1}\right)
\end{aligned}
$$

where the first step holds due to Fact 2.9. Using $c\le t=0.2p$ and $N\ge p^{2-\gamma}/100$, we have

$$
\sum_{i\in[c]}\log_2\left(\frac{N}{N-i+1}\right)\le 2\sum_{i\in[t]}\frac{i}{N-i}\le 2\sum_{i\in[t]}\frac{t}{N-t}\le\frac{2t^2}{N-t}\le 100p^{\gamma},
$$

where we used $\log_2(1+x)\le 2x$ when $x\ge 0$. Combining with (9) and $c\ge C$, we have

$$
I\big(a^*;\Pi_2 \mid \Pi_1=\widehat{\Pi}_1,|S^*|=c\big)\le\frac{m_2+100p^{\gamma}}{c}=o_p(1), \qquad (10)
$$

where we used the assumption that $\gamma\le 0.05$ and $m_2\le\sqrt{p}$. This finishes the proof of (7). □

3.3.2 Proof of Theorem 3.1 via a direct sum argument

Finally we apply the compression protocol from [JPY12] (see Lemma 2.12) and a direct sum argument similar to that of [BBCR13, BR11, Bra15] to reduce the lower bound of Theorem 3.1 to that of Lemma 3.4 for one cell.

Proof of Theorem 3.1. Recall in the one-cell problem, Alice receives $a\in[p]\times[p]$, Bob receives a size-t subset A of $[p]\times[p]$ with $a\in A$, and Charlie receives $r\in[p]\times[p]$, distributed according to μ_p.

Assume for a contradiction that there is an (m_1,m_2)-bit deterministic communication protocol (Π_1,Π_2,Π_3) for $f_{n,d,p}$ over $\mu_{n,d,p}$ with $m_1=o(nd\log p)$, $m_2<\sqrt{p}$, and failure probability at most $1/40$. We first use it to obtain an (m_1,m_2)-bit public-coin communication protocol Π' for the single cell communication problem f_p over μ_p that has the same failure probability as Π. While the first message Π'_1 of Π' has length m_1 (which is too large to reach a contradiction with Lemma 3.4), we show the amount of (internal) information it contains is small:

$$I(\Pi'_1; a \mid A, R) = o(\log p),$$

where R denotes the public random string of Π'. Then we apply a compression on Π'_1 to obtain a new public-coin communication protocol Π'' for the single-cell problem such that the first message has $o(\log p)$ bits only (while the second message remains the same as Π'). This leads to a contradiction with our lower bound for the single-cell problem in Lemma 3.4.

Consider the single-cell communication problem f_p over μ_p, with $(a, A, r) \sim \mu_p$ as inputs of the three players, respectively. Given $\Pi = (\Pi_1, \Pi_2, \Pi_3)$ for $f_{n,d,p}$ over $\mu_{n,d,p}$, we describe a public-coin protocol for the single-cell communication problem as follows:

1. Public-coin randomness R: The protocol starts by sampling a public-coin random string R. First R contains $i^* \in [n]$ and $j^* \in [d]$ sampled independently and uniformly at random. For each $(i, j) \in [n] \times [d]$ with $(i, j) \prec (i^*, j^*)$, R contains $a_{i,j}$ drawn from $[p] \times [p]$ independently and uniformly at random. For each $(i, j) \in [n] \times [d]$ with $(i, j) \succ (i^*, j^*)$, R contains a size-t subset $A_{i,j}$ sampled from $[p] \times [p]$ independently and uniformly at random.

2. Alice, Bob and Charlie run Π on (X, Y, Z) constructed as follows and Charlie returns the same bit that Π returns. The input X of Alice contains $(a_{i,j} : i \in [n], j \in [d])$, where $a_{i,j}$ for each $(i, j) \prec (i^*, j^*)$ is from R, a_{i^*,j^*} is set to be a (which is her original input in the single-cell problem), and $a_{i,j}$ for each $(i, j) \succ (i^*, j^*)$ is sampled from $A_{i,j}$ (from R) independently and uniformly at random using her private randomness. The input Y of Bob contains i^* and $(A_{i,j} : i \in [n], j \in [d])$, where $A_{i,j}$ for each $(i, j) \prec (i^*, j^*)$ is sampled as a size-t subset of $[p] \times [p]$ that contains $a_{i,j}$ (from R) independently and uniformly at random using his private randomness, A_{i^*,j^*} is set to be A (which is his original input), and $A_{i,j}$ for each $(i, j) \succ (i^*, j^*)$ is from R. Finally the input Z of Charlie contains i^*, j^* and r (which is his original input).

It is easy to verify that when $(a, A, r) \sim \mu_p$, the triple (X, Y, Z) above is distributed exactly the same as $\mu_{n,d,p}$ and thus, the error probability of the above single-cell protocol Π' is exactly the same as the error probability of Π (at most 1/40).

We next bound the amount of internal information the first message $\Pi'_1 = \Pi_1(X)$ contains:

$$I(\Pi'_1; a \mid A, R) = I(\Pi_1; a \mid A, R).$$

To this end, note that these four random variables we care about can also be drawn equivalently as follows: First we draw $(a_{i,j} : i \in [n], j \in [d])$ uniformly and then draw $(A_{i,j} : i \in [n], j \in [d])$ as size-t subsets that contain $a_{i,j}$ uniformly at random. Next we draw $i^* \in [n]$ and $j^* \in [d]$ uniformly. Finally we set Π_1 to be $\Pi_1(a_{i,j} : i \in [n], j \in [d])$, a to be a_{i^*,j^*}, A to be A_{i^*,j^*}, R to be $(i^*, j^*, a^{\prec}_{i^*,j^*}, A^{\succ}_{i^*,j^*})$, where we write $a^{\prec}_{i^*,j^*}$ to denote the ordered tuple of $a_{i,j}$ with $(i, j) \prec (i^*, j^*)$ and $A^{\succ}_{i^*,j^*}$ to denote the ordered tuple of $A_{i,j}$ with $(i, j) \succ (i^*, j^*)$.

In this view we have the following sequence of inequalities:

$$\begin{aligned} I(\Pi_1; a \mid A, R) &= \frac{1}{nd} \sum_{i\in[n], j\in[d]} I(\Pi_1; a_{i,j} \mid a_{i,j}^{\prec}, A_{i,j}, A_{i,j}^{\succ}) \\ &\leq \frac{1}{nd} \sum_{i\in[n], j\in[d]} I(\Pi_1; a_{i,j} \mid a_{i,j}^{\prec}, A_{1,1}, \ldots, A_{n,d}) \\ &= \frac{1}{nd} \cdot I(\Pi_1; a_{1,1}, \ldots, a_{n,d} \mid A_{1,1}, \ldots, A_{n,d}) \leq \frac{m_1}{nd} = o(\log p), \end{aligned}$$

where the second step follows from Fact 2.11 as $A_{i,j}^{\prec}$ is independent of $a_{i,j}$ conditioning on $a_{i,j}^{\prec}, A_{i.j}, A_{i,j}^{\succ}$, the third step follows from the chain rule of mutual information.

Now we apply Lemma 2.12 on the first message Π_1' of Π. Formally in Lemma 2.12 we take X to be the input a of Alice, Y to be the input A of Bob, R to be the public randomness, and M to be the first message Π_1' from Alice to Bob. Setting $\epsilon' = 1/600$ in Lemma 2.12 and noting that Π_1' is independent of Y conditioning on X and R, we can replace the first message Π_1' by a public-coin one-way protocol from Alice to Bob, where Bob uses the random variable M' in Lemma 2.12 as the message from Alice to continue to run Π'. Let Π'' be the new public-coin protocol. Then the length of its first message by Lemma 2.12 is at most

$$\frac{o(\log p) + 5}{\epsilon'} + O\left(\log \frac{1}{\epsilon'}\right) = o(\log p)$$

and the length of its second message is still at most $\sqrt{p}$. The error probability of Π'' is at most

$$\frac{1}{40} + 6\epsilon' = \frac{1}{40} + \frac{1}{100} < \frac{9}{200}$$

by Lemma 2.12. This contradicts with Lemma 3.4. Hence we conclude the proof. □

4 Memory efficient multi-pass algorithm

When the continual learner is allowed to take multiple passes over the sequence of tasks, the memory requirement can be significantly reduced. Formally, we recall the following theorem:

Theorem 1.2. *Let $\epsilon \in (0, 1/10]$ and k, d, b, c be four positive integers. Let $\mathcal{H}$ be any hypothesis class of VC dimension d over a data universe $\mathcal{X}$ of description size b. There is a c-pass $(\epsilon, 0.01)$-continual learner for $\mathcal{H}$ over sequences of k tasks that has memory requirement and sample complexity*

$$\left(\frac{k}{\epsilon}\right)^{2/c} \cdot db \cdot \mathrm{poly}\big(c, \log k/\epsilon\big) \quad \textit{and} \quad \left(\frac{k}{\epsilon}\right)^{4/c} \cdot \frac{k+d}{\epsilon} \cdot \mathrm{poly}\big(c, \log kd/\epsilon\big),$$

respectively. Whenever $c \geq 4\log(k/\epsilon)$, the memory requirement becomes $db \cdot \mathrm{polylog}(kd/\epsilon)$ and the sample complexity becomes $\widetilde{O}((k+d)/\epsilon)$, both are optimal up to a polylogarithmic factor.

Notation Let $\delta = 0.01$ be the confidence parameter and $\gamma = \frac{1}{10c^2}$. We assume a lexical order for elements in $\mathcal{X}$ and slightly abuse of notation, for any $x_1, x_2 \in \mathcal{X}$ and $q_1, q_2 \in \mathbb{R}$, we write $(x_1, q_1) \preceq (x_2, q_2)$ if $(q_1 < q_2) \vee (q_1 = q_2 \wedge x_1 < x_2)$. Let $\mu_{\mathcal{D}_i}(x)$ be the probability density of x on the i-th distribution $\mathcal{D}_i$ ($i \in [k]$). It is WLOG to assume $\mu_{\mathcal{D}_i}(x) = o(\frac{\epsilon}{c^3})$ for every $i \in [k]$ and $x \in \mathcal{X}$, as one can always append $\log_2(c/\epsilon)$ random bits to the data point x.

Algorithm 1 Multi-pass continual learner

Initialize $\alpha = \frac{1}{4}(\frac{2k}{\epsilon})^{-2/c}$, $\eta = \log \frac{1-\alpha}{\alpha}$, $\epsilon_t = (1+\frac{1}{c})^t \cdot \frac{\epsilon}{20c}$, $\forall t \in [0:c]$
Initialize $N = \Theta(\frac{d+\log(c/\delta)}{\alpha})$, $M_1 = \Theta(\frac{c^4 \log(kc/\delta)}{\epsilon_0})$, $M_2 = \Theta(\frac{\log(kc/\delta)}{\epsilon_0 \alpha^2})$
for $t = 1, 2, \dots, c$ **do**
    $\widehat{w}_t \leftarrow 0, S_t \leftarrow \mathcal{D}_1^N$
    for $i = 1, 2, \dots, k$ **do**
        $\{(x_{i,t,\tau}, \widehat{q}_{i,t,\tau})\}_{\tau \in [t-1]} \leftarrow \textsc{EstimateQuantile}(\mathcal{D}_i, h_1, \dots, h_{t-1})$
        $\widehat{w}_{i,t} \leftarrow \textsc{EstimateWeight}(\mathcal{D}_i, h_1, \dots, h_{t-1})$
        $\widehat{w}_t \leftarrow \widehat{w}_t + \widehat{w}_{i,t}$
        $\widehat{\mathcal{D}}_{i,t} \leftarrow \textsc{TruncatedRejectionSampling}(\mathcal{D}_i, h_1, \dots, h_{t-1})$
        for each training data in S_t **do**
            With probability $\widehat{w}_{i,t}/\widehat{w}_t$, replace it with a sample from $\widehat{\mathcal{D}}_{i,t}$
        end for
    end for
    $h_t = \arg\min_{h \in \mathcal{H}} \ell_{S_t}(h)$
end for
Return $h = \mathsf{maj}(\sum_{t \in [c]} h_t)$

$\textsc{EstimateQuantile}(\mathcal{D}_i, h_1, \dots, h_{t-1})$

for $\tau = 1, 2, \dots, t-1$ **do**
    $S_{i,t,\tau}^q \sim \mathcal{D}_i^{M_1}$
    $\overline{S}_{i,t,\tau}^q = \{(x,y) \in S_{i,t,\tau}^q : (x, \exp(\eta \sum_{\xi=1}^{\nu} \mathbf{1}[h_\xi(x) \neq y])) \preceq (x_{i,t,\nu}, \widehat{q}_{i,t,\nu}), \forall \nu \in [\tau-1]\}$
    Sort $\{(x, \exp(\eta \sum_{\nu=1}^{\tau} \mathbf{1}[h_\nu(x) \neq y]))\}_{x \in \overline{S}_{i,t,\tau}^q}$ and select the the top ϵ_t-quantile $(x_{i,t,\tau}, \widehat{q}_{i,t,\tau})$
end for
Return $\{(x_{i,t,\tau}, \widehat{q}_{i,t,\tau})\}_{\tau \in [t-1]}$

$\textsc{EstimateWeight}(\mathcal{D}_i, h_1, \dots, h_{t-1})$

$S_{i,t}^w \sim \mathcal{D}_i^{M_2}$
$\overline{S}_{i,t}^w = \{(x,y) \in S_{i,t}^w : (x, \exp(\eta \sum_{\nu=1}^{\tau} \mathbf{1}[h_\nu(x) \neq y])) \preceq (x_{i,\tau}, \widehat{q}_{i,t,\tau}), \forall \tau \in [t-2]\}$
Return $\frac{1}{|M_2|} \sum_{(x,y) \in \overline{S}_{i,t}^w} \min\{\exp(\eta \sum_{\tau=1}^{t-1} \mathbf{1}[h_\tau(x) \neq y]), \widehat{q}_{i,t,t-1}\}$

$\textsc{TruncatedRejectionSampling}(\mathcal{D}_i, h_1, \dots, h_{t-1})$

1: Sample $(x,y) \sim \mathcal{D}_i$
2: Reject and go back to Step 1 if $\exists \tau \in [t-2]$, s.t. $(x, \exp(\eta \sum_{\nu=1}^{\tau} \mathbf{1}[h_\nu(x) \neq y])) \preceq (x_{i,t,\tau}, \widehat{q}_{i,t,\tau})$
3: Sample λ from $[0, \widehat{q}_{i,t,t-1}]$ uniformly
4: Reject and return to Step 1 if $\lambda \geq \min\{\exp(\eta \sum_{\nu=1}^{t-1} \mathbf{1}[h_\nu(x) \neq y]), \widehat{q}_{i,t,t-1}\}$
5: Return (x,y)

Overview of the algorithm We provide a high level overview of our approach, the pseudocode can be found at Algorithm 1. Our algorithm is based on the idea of boosting (note that this means it will be improper). During each pass $t \in [c]$, the algorithm maintains a *hierarchical truncated* set of the i-th data distribution $\mathcal{D}_i$ for each $i \in [k]$. Let

$$\mathcal{X}_{i,t,\tau} = \left\{ x \in \mathcal{X} : (x, \exp(\sum_{\xi=1}^{\nu} \mathbf{1}[h_\xi(x) \neq y])) \preceq (x_{i,t,\nu}, \widehat{q}_{i,t,\nu}), \forall \nu \in [\tau - 1] \right\}, \quad \forall \tau \in [t] \tag{11}$$

and we take $\mathcal{X}_{i,t,1} = \cdots = \mathcal{X}_{k,t,1} = \mathcal{X}$ in the above definition. These are hierarchical sets and with high probability satisfy (See Lemma 4.2 for the proof)

(1) $\mathcal{X}_{i,t,t} \subseteq \mathcal{X}_{i,t,t-1} \subseteq \cdots \subseteq \mathcal{X}_{i,t,1}, \quad \forall i \in [k], t \in [c]$;

(2) $\mathcal{X}_{i,c,\tau} \subseteq \cdots \subseteq \mathcal{X}_{i,\tau+1,\tau} \subseteq \mathcal{X}_{i,\tau,\tau}, \quad \forall i \in [k], \tau \in [c]$.

These sets are obtained via the EstimateQuantile procedure, which iteratively computes the top ϵ_t-quantile of $\mathcal{D}_i$.

Roughly speaking, $\mathcal{X}_{i,t,t-1}$ is the "support" of $\mathcal{D}_i$ in the t-th pass. Our algorithm performs multiplicative weights update over every data point in $\mathcal{X}$ and the distribution $\mathcal{D}_i$ gets reweighted at every round. EstimateWeight estimates the updated weight of $\mathcal{X}_{i,t,t-1}$ with the top ϵ_t-quantile truncated. TruncatedRejectionSampling has access to the original distribution $\mathcal{D}_i$ and performs rejection sampling to get samples from the truncated distribution.

We sample the training set S_t of the t-th round in a streaming manner. Initially we add N dummy samples. In the i-th task, we discard old data and replace them with a new one with probability $\widehat{w}_{i,t}/\widehat{w}_t$. We run ERM over the training set S_t at the end, and derive a hypothesis h_t. The final output is determined by the majority vote $h = \mathsf{maj}(\sum_{t\in[c]} h_t)$, defined as

$$h(x) = \mathsf{maj}\left(\sum_{t\in[c]} h_t(x)\right) = \begin{cases} 1 & \sum_{t\in[c]} h_t(x) \geq c/2 \\ 0 & \text{otherwise} \end{cases}$$

We sketch the proof of Theorem 1.2 and detailed proof of this section can be found in the Appendix B. We first prove that the EstimateQuantile procedure provides good estimations on the quantile each step.

Lemma 4.1 (Quantile estimation)**.** *With probability at least $1 - \frac{\delta}{20}$, the following event holds*

1. *For any $i \in [k], t \in [c], \tau \in [t-1]$,*

$$\Pr_{(x,y)\sim\mathcal{D}_i|x\in\mathcal{X}_{i,t,\tau}}\left[\left(x, \exp(\eta \sum_{\nu=1}^{\tau} \mathbf{1}\{h_\tau(x) \neq y\})\right) \succeq (x_{i,t,\tau}, \widehat{q}_{i,t,\tau})\right] \in [(1-\gamma)\epsilon_t, (1+\gamma)\epsilon_t]\,.$$

Here $\mathcal{D}_i|x \in \mathcal{X}_{i,t,\tau}$ denotes the conditional distribution of $\mathcal{D}_i$ on the event of $x \in \mathcal{X}_{i,t,\tau}$.

2. *Denote $P_{i,t,\tau} := \Pr_{(x,y)\sim\mathcal{D}_i}[x \in \mathcal{X}_{i,t,\tau}]$. For any $i \in [k], t \in [c], \tau \in [t-1]$,*

$$P_{i,t,\tau+1} \in [(1-(1+\gamma)\epsilon_t)P_{i,t,\tau}, (1-(1-\gamma)\epsilon_t)P_{i,t,\tau}]\,.$$

We next prove the sets $\{X_{i,t,\tau}\}_{i\in[k],t\in[c],\tau\in[t]}$ have a hierarchical structure.

Lemma 4.2 (Hierarchical set)**.** *Condition on the event of Lemma 4.1, we have*

1. $\mathcal{X}_{i,t,t} \subseteq \mathcal{X}_{i,t,t-1} \subseteq \cdots \subseteq \mathcal{X}_{i,t,1}, \quad \forall i \in [k], t \in [c]$;

2. $\mathcal{X}_{i,c,\tau} \subseteq \cdots \subseteq \mathcal{X}_{i,\tau+1,\tau} \subseteq \mathcal{X}_{i,\tau,\tau}, \quad \forall i \in [k], \tau \in [c]$.

We formally define the updated distribution at each pass. Ideally, we sample from the *truncated* distribution.

Definition 4.3 (Truncated distribution). *For any $i \in [k], t \in [c]$, define the truncated distribution $\mathcal{D}_{i,t,\mathsf{trun}}$ as*

$$\mu_{\mathcal{D}_{i,t,\mathsf{trun}}}(x) = \frac{\mathbf{1}[x \in \mathcal{X}_{i,t,t-1}] \cdot \mu_{\mathcal{D}_i}(x) \cdot \min\{\exp(\eta \sum_{\tau=1}^{t-1} \mathbf{1}[h_\tau(x) \neq y]), \widehat{q}_{i,t,t-1}\}}{\sum_{x' \in \mathcal{X}} \mathbf{1}[x' \in \mathcal{X}_{i,t,t-1}] \cdot \mu_{\mathcal{D}_i}(x') \cdot \min\{\exp(\eta \sum_{\tau=1}^{t-1} \mathbf{1}[h_\tau(x') \neq y]), \widehat{q}_{i,t,t-1}\}}, \quad \forall x \in \mathcal{X}.$$

The mixed truncated distribution $\mathcal{D}_{t,\mathsf{trun}}$ is defined as

$$\mu_{\mathcal{D}_{t,\mathsf{trun}}}(x) = \frac{\sum_{i=1}^{k} \mathbf{1}[x \in \mathcal{X}_{i,t,t-1}] \cdot \mu_{\mathcal{D}_i}(x) \cdot \min\{\exp(\eta \sum_{\tau=1}^{t-1} \mathbf{1}[h_\tau(x) \neq y]), \widehat{q}_{i,t,t-1}\}}{\sum_{i=1}^{k} \sum_{x' \in \mathcal{X}} \mathbf{1}[x' \in \mathcal{X}_{i,t,t-1}] \cdot \mu_{\mathcal{D}_i}(x') \cdot \min\{\exp(\eta \sum_{\tau=1}^{t-1} \mathbf{1}[h_\tau(x') \neq y]), \widehat{q}_{i,t,t-1}\}}, \quad \forall x \in \mathcal{X}.$$

To connect $\mathcal{D}_{t,\mathsf{trun}}$ and $\mathcal{D}_{i,t,\mathsf{trun}}$, we define the weights $\{w_{i,t}\}_{i \in [k]}$

$$w_{i,t} = \sum_{x \in \mathcal{X}} \mathbf{1}[x \in \mathcal{X}_{i,t,t-1}] \cdot \mu_{\mathcal{D}_i}(x) \cdot \min\left\{\exp(\eta \sum_{\tau=1}^{t-1} \mathbf{1}[h_\tau(x) \neq y]), \widehat{q}_{i,t,t-1}\right\}$$

and $p_{i,t} = \frac{w_{i,t}}{\sum_{i' \in [k]} w_{i',t}}$. The mixed distribution $\mathcal{D}_{i,\mathsf{trun}}$ can be derived by first drawing an index i from $[k]$ according to $\{p_{i,t}\}_{i \in [k]}$, and then drawing from $\mathcal{D}_{i,t,\mathsf{trun}}$. That is to say, $\mathcal{D}_{t,\mathsf{trun}} = \sum_{i=1}^{k} p_{i,t} \mathcal{D}_{i,t,\mathsf{trun}}$.

We next prove that the EstimateWeight procedure approximately estimates the weight of each distribution $\mathcal{D}_{i,t}$.

Lemma 4.4 (Weight estimation). *Condition on the event of Lemma 4.1, with probability at least $1 - \frac{\delta}{20}$, the* EstimateWeight *procedure outputs $\widehat{w}_{i,t} = (1 \pm \alpha/8) w_{i,t}$. Furthermore, define $\widehat{p}_{i,t} = \frac{\widehat{w}_{i,t}}{\sum_{i' \in [k]} \widehat{w}_{i',t}}$, then $\widehat{p}_{i,t} = (1 \pm \alpha/3) p_{i,t}$ for any $i \in [k], t \in [c]$.*

We then analyse the TruncatedRejectionSampling process and prove that it returns a sample from $\mathcal{D}_{i,t,\mathsf{trun}}$ without much overhead on sample complexity.

Lemma 4.5 (Truncated rejection sampling). *For any $i \in [k], t \in [c]$, condition on the event of Lemma 4.1, the* TruncatedRejectionSampling*$(\mathcal{D}_i, h_1, \ldots, h_{t-1})$ returns a sample from the distribution of $\mathcal{D}_{i,t,\mathsf{trun}}$. With probability at least $1 - \frac{\delta}{20}$, the total number of samples that have been drawn for the training set $\{S_t\}_{t \in [c]}$ is at most $O\left(\frac{dc^2 \log^3(kdc/\epsilon\delta)}{\epsilon\alpha}\right)$.*

For any $t \in [c]$, define $\widehat{\mathcal{D}}_{t,\mathsf{trun}} = \sum_{i=1}^{t} \widehat{p}_{i,t} \mathcal{D}_{i,t,\mathsf{trun}}$. Recall that our algorithm samples and updates the training data S_t in a streaming manner. It is easy to verify that each data of S_t is drawn from $\widehat{\mathcal{D}}_{t,\mathsf{trun}}$.

Lemma 4.6. *For any $t \in [c]$, the training set S_t is drawn from $\widehat{\mathcal{D}}_{t,\mathsf{trun}}^N$.*

Additionally, combining Lemma 4.4 and Lemma 4.5, one can prove $\widehat{\mathcal{D}}_{t,\mathsf{trun}}$ is close to $\mathcal{D}_{t,\mathsf{trun}}$

Lemma 4.7 (Bounded TV distance). *For any $t \in [c]$, conditioned on the events of Lemma 4.1 and Lemma 4.4, the empirical truncated distribution $\widehat{\mathcal{D}}_{t,\mathsf{trun}}$ and truncated distribution $\mathcal{D}_{t,\mathsf{trun}}$ satisfies $\|\widehat{\mathcal{D}}_{t,\mathsf{trun}} - \mathcal{D}_{t,\mathsf{trun}}\|_{\mathsf{TV}} \leq \alpha/3$.*

By a standard application of VC theory, we can show the hypothesis h_t returned by Algorithm 1 in the t-th pass achieves good accuracy on $\mathcal{D}_{t,\mathsf{trun}}$.

Lemma 4.8. *For any $t \in [c]$, condition on the events of Lemma 4.7, with probability at least $1 - \frac{\delta}{20c}$, we have $\ell_{\mathcal{D}_{t,\mathsf{trun}}}(h_t) \le \frac{\alpha}{2}$.*

Now we are able to derive the accuracy guarantee.

Lemma 4.9. *Let $h = \mathsf{maj}(\sum_{t\in[c]} h_t)$ be the function returned by Algorithm 1, with probability at least $1 - \frac{\delta}{4}$, $\ell_{\mathcal{D}_i}(h) \le \epsilon$ holds for any $i \in [k]$.*

Proof. We proceed the proof by conditioning on the event of Lemma 4.1, Lemma 4.4 and Lemma 4.8. The key is to track the following potential

$$\Phi_t = \frac{1}{k}\sum_{i=1}^{k}\sum_{x\in\mathcal{X}_{i,t,t-1}} \mu_{\mathcal{D}_i}(x)\exp(\eta\sum_{\tau=1}^{t-1}\mathbf{1}[h_\tau(x)\neq y]), \quad t \in [c+1].$$

Note that, when $t = 1$, we take $\mathcal{X}_{i,t,0} = \mathcal{X}$ and therefore $\Phi_1 = \frac{1}{k}\sum_{i=1}^{k}\sum_{x\in\mathcal{X}}\mu_{\mathcal{D}_i}(x) = 1$; When $t = c+1$, for the purpose of purpose, we extend the definition to $\mathcal{X}_{i,c+1,c}$ by imagining an extra pass and assuming that the hierarchical property holds (i.e., Lemma 4.2) as well.

For any $t \in [2:c]$, we have that

$$\begin{aligned}
&\log\Phi_{t+1} - \log\Phi_t\\
=\ &\log\sum_{i=1}^{k}\sum_{x\in\mathcal{X}_{i,t+1,t}} \frac{\mu_{\mathcal{D}_i}(x)\exp(\eta\sum_{\tau=1}^{t}\mathbf{1}[h_\tau(x)\neq y])}{\sum_{i=1}^{k}\sum_{x'\in\mathcal{X}_{i,t,t-1}}\mu_{\mathcal{D}_i}(x')\exp(\eta\sum_{\tau=1}^{t-1}\mathbf{1}[h_\tau(x')\neq y])}\\
\le\ &\log\sum_{i=1}^{k}\sum_{x\in\mathcal{X}_{i,t,t}} \frac{\mu_{\mathcal{D}_i}(x)\exp(\eta\sum_{\tau=1}^{t}\mathbf{1}[h_\tau(x)\neq y])}{\sum_{i=1}^{k}\sum_{x'\in\mathcal{X}_{i,t,t-1}}\mu_{\mathcal{D}_i}(x')\exp(\eta\sum_{\tau=1}^{t-1}\mathbf{1}[h_\tau(x')\neq y])}\\
=\ &\log\sum_{i=1}^{k}\sum_{x\in\mathcal{X}_{i,t,t}} \frac{\mu_{\mathcal{D}_i}(x)\exp(\eta\sum_{\tau=1}^{t-1}\mathbf{1}[h_\tau(x)\neq y])}{\sum_{i=1}^{k}\sum_{x'\in\mathcal{X}_{i,t,t-1}}\mu_{\mathcal{D}_i}(x')\exp(\eta\sum_{\tau=1}^{t-1}\mathbf{1}[h_\tau(x')\neq y])}\cdot\exp(\eta\mathbf{1}[h_t(x)\neq y]))\\
\le\ &\log\sum_{i=1}^{k}\sum_{x\in\mathcal{X}_{i,t,t}} \frac{\mu_{\mathcal{D}_i}(x)\cdot\min\{\exp(\eta\sum_{\tau=1}^{t-1}\mathbf{1}[h_\tau(x)\neq y]),\widehat{q}_{i,t,t-1}\}}{\sum_{i=1}^{k}\sum_{x'\in\mathcal{X}_{i,t,t-1}}\mu_{\mathcal{D}_i}(x')\cdot\min\{\exp(\eta\sum_{\tau=1}^{t-1}\mathbf{1}[h_\tau(x')\neq y]),\widehat{q}_{i,t,t-1}\}}\cdot\exp(\eta\mathbf{1}[h_t(x)\neq y]))\\
\le\ &\log\sum_{i=1}^{k}\sum_{x\in\mathcal{X}_{i,t,t-1}} \frac{\mu_{\mathcal{D}_i}(x)\cdot\min\{\exp(\eta\sum_{\tau=1}^{t-1}\mathbf{1}[h_\tau(x)\neq y]),\widehat{q}_{i,t,t-1}\}}{\sum_{i=1}^{k}\sum_{x'\in\mathcal{X}_{i,t,t-1}}\mu_{\mathcal{D}_i}(x')\cdot\min\{\exp(\eta\sum_{\tau=1}^{t-1}\mathbf{1}[h_\tau(x')\neq y]),\widehat{q}_{i,t,t-1}\}}\cdot\exp(\eta\mathbf{1}[h_t(x)\neq y]))\\
=\ &\log(1-\ell_{\mathcal{D}_{t,\mathsf{trun}}}(h_t)+\ell_{\mathcal{D}_{t,\mathsf{trun}}}(h_t)\cdot\exp(\eta))\\
\le\ &\log(1-\alpha+\alpha\cdot\exp(\eta))\\
\le\ &\log 2(1-\alpha)
\end{aligned}\tag{12}$$

The second step follows from $\mathcal{X}_{i,t+1,t} \subseteq \mathcal{X}_{i,t,t}$ (see Lemma 4.2), the fourth step follows from

$$\exp(\eta\sum_{\tau=1}^{t-1}\mathbf{1}[h_\tau(x)\neq y]) \le \widehat{q}_{i,t,t-1}, \quad \forall x\in\mathcal{X}_{i,t,t}.$$

The fifth step follows from $\mathcal{X}_{i,t,t} \subseteq \mathcal{X}_{i,t,t-1}$, the sixth step follows from the definition of $\ell_{\mathcal{D}_{t,\mathsf{trun}}}(h_t)$ and the seventh step follows from $\ell_{\mathcal{D}_{t,\mathsf{trun}}}(h_t) \le \alpha$ (Lemma 4.8) and we plug in $\eta = \log(\frac{1-\alpha}{\alpha})$ in the last step.

For any $i \in [k]$, define

$$\mathcal{X}_{i,\mathsf{bad}} = \left\{ x \in \mathcal{X}_{i,c+1,c} : \sum_{t=1}^{c} \mathbf{1}[h_t(x) \neq y] \geq c/2 \right\}.$$

That is to say, $\mathcal{X}_{i,\mathsf{bad}}$ contains all data points that are wrongly labeled by function h in $\mathcal{X}_{i,c+1,c}$. Note that

$$\begin{aligned}
\Phi_{c+1} &= \frac{1}{k} \sum_{i'=1}^{k} \sum_{x \in \mathcal{X}_{i',c+1,c}} \mu_{\mathcal{D}_{i'}}(x) \exp(\eta \sum_{t=1}^{c} \mathbf{1}[h_t(x) \neq y]) \\
&\geq \frac{1}{k} \sum_{x \in \mathcal{X}_{i,c+1,c}} \mu_{\mathcal{D}_i}(x) \exp(\eta \sum_{t=1}^{c} \mathbf{1}[h_t(x) \neq y]) \\
&\geq \frac{1}{k} \sum_{x \in \mathcal{X}_{i,\mathsf{bad}}} \mu_{D_i}(x) \exp(\eta c/2). \qquad (13)
\end{aligned}$$

Now taking a telescopic summation on Eq. (12), we have

$$\begin{aligned}
c \log 2(1-\alpha) \geq \ & \log \Phi_{c+1} - \log \Phi_1 = \log \Phi_{c+1} \geq \eta c/2 + \log \sum_{x \in \mathcal{X}_{i,\mathsf{bad}}} \mu_{D_i}(x) - \log k \\
&= \frac{c}{2} \log \frac{1-\alpha}{\alpha} + \log \sum_{x \in \mathcal{X}_{i,\mathsf{bad}}} \mu_{D_i}(x) - \log k,
\end{aligned}$$

where the second step holds as $\Phi_1 = 1$, the third step holds due to Eq. (13), we plug in $\eta = \log \frac{1-\alpha}{\alpha}$ in the last step. Rearranging these terms and plug in $\alpha = \frac{1}{4}(\frac{2k}{\epsilon})^{-2/c}$,

$$\log \sum_{x \in \mathcal{X}_{i,\mathsf{bad}}} \mu_{\mathcal{D}_i}(x) - \log k \leq c \log 2\sqrt{\alpha(1-\alpha)} \leq \log(\epsilon/2k),$$

Therefore, we have

$$\sum_{x \in \mathcal{X}_{i,\mathsf{bad}}} \mu_{\mathcal{D}_i}(x) \leq \frac{\epsilon}{2}. \qquad (14)$$

For those outside of $\mathcal{X}_{i,\mathsf{bad}}$, we have

$$\Pr_{x \sim \mathcal{D}_i} [x \notin \mathcal{X}_{i,c+1,c}] \leq 1 - (1 - (1-\gamma)\epsilon_{c+1})^c \leq 3c\epsilon_0 \leq \frac{\epsilon}{2}. \qquad (15)$$

The first step follows from Lemma 4.1, the third step holds as $\epsilon_0 = \frac{\epsilon}{10c}$. Combining Eq. (14) and Eq. (15), we conclude $\ell_{\mathcal{D}_i}(h) \leq \epsilon$ and finish the proof. □

We now wrap up the proof of Theorem 1.2.

Proof of Theorem 1.2. The accuracy guarantee of Algorithm 1 has already been established by Lemma 4.9. We next calculate the memory requirement and sample complexity.

Memory requirement For the memory requirement, we need (1) storing the training data S_t, which takes $O(N) \le O((\frac{k}{\epsilon})^{2/c} db \log(kc/\delta))$ bits; (2) storing samples $S_{i,t}^q$ for EstimateQuantile, which takes $O(bc + M_1 b\epsilon_0) = O(bc^4 \log(kc/\delta))$ bits using streaming implementation; (3) storing samples $S_{i,t}^w$ for EstimateWeight, which takes $O(\log(k/\epsilon))$ bits using streaming implementation; (4) storing the hypothesis $h_1, \ldots, h_c$, which takes $O(dbc)$ bits; Hence the total space requirement is

$$O\left(\left(\frac{k}{\epsilon}\right)^{2/c} db\log(kc/\delta) + bc^4\log(kc/\delta) + dbc + \log(k/\epsilon)\right).$$

Sample complexity Algorithm 1 draws $(M_1c+M_2)kc = O(kc^7\log(kc/\delta)\epsilon^{-1} + (\frac{k}{\epsilon})^{4/c}kc^2\log(kc/\delta)\epsilon^{-1})$ samples for estimating weights and quantile. By Lemma 4.5, with probability at least $1-\frac{\delta}{20}$, the training set $\{S_t\}_{t\in[c]}$ takes $O\left(\frac{dc^2\log^3(kdc/\epsilon\delta)}{\epsilon\alpha}\right)$ samples. Hence the sample complexity is

$$O\left(\left(\frac{k}{\epsilon}\right)^{2/c} dc^2\log^3(kdc/\epsilon\delta)\epsilon^{-1} + kc^7\log(kc/\delta)\epsilon^{-1} + \left(\frac{k}{\epsilon}\right)^{4/c} kc^2\log(kc/\delta)\epsilon^{-1}\right).$$

This concludes the proof. □

5 An exponential separation result for multi-pass learning

We prove a lower bound for the memory requirement of c-pass *proper* continual learners, showing that making c passes can only help save memory by a factor of no more than c^3. This is in sharp contrast to the $1/c$ saving in the exponent by the improper learner presented in the last section.

Theorem 1.3. *There exists a sufficiently large constant C_0 such that the following holds. For any $\epsilon \in (0, 1/4]$ and any positive integers c, k, d and b such that $d \ge C_0 c$ and $b \ge C_0 \log(kd/(c\epsilon))$, there is a hypothesis class $\mathcal{H}$ of VC dimension d defined over a universe $\mathcal{X}$ of description size b such that any c-pass $(\epsilon, 0.01)$-continual learner for $\mathcal{H}$ over $2k$ tasks requires $(1/c^2)\cdot\widetilde{\Omega}(kdb/(c\epsilon))$ memory.*

5.1 Pointer chasing

We reduce from the classic communication problem of pointer chasing.

Definition 5.1 (Pointer chasing [PS82])**.** *Let n and c be two positive integers. In a pointer chasing problem, Alice gets an input map $f_A : [n] \to [n]$ and Bob receives another input map $f_B : [n] \to [n]$. The pointers $w^{(1)}, w^{(2)}, \ldots$ are recursive defined as*

$$w^{(1)} = 1, \quad w^{(2)} = f_A(w^{(1)}), \quad w^{(3)} = f_B(w^{(2)}), \quad w^{(4)} = f_A(w^{(3)}), \quad w^{(5)} = f_B(w^{(4)}), \quad \ldots$$

In the communication problem, Alice speaks first and the communication proceeds in at most $2c-1$ rounds. The goal is for Bob to output the binary value of $w^{(2c+2)}$ (mod 2).

The problem requires $\Omega(n/c - c\log n)$ bits of (total) communication:

Theorem 5.2 ([NW91, Kla00, Yeh20])**.** *The randomized communication complexity of the pointer chasing problem (with error probability at most $1/3$) is at least $n/(2000c) - 2c\log n$.*

5.2 Hypothesis class

Let c, k, d and b be four positive integers. We describe the hypothesis class $\mathcal{H} = \mathcal{H}_{c,k,d,b}$ that will be used to prove our lower bound in Theorem 1.3. We start with the data universe $\mathcal{X}$ of $\mathcal{H}$:

Data universe. Let p be the smallest prime with $p \geq (kdb)^b$. The data universe $\mathcal{X}$ consists of $\mathcal{X} = Y \cup Z$, where Y and Z can be further partitioned into

$$Y = Y_1 \cup \cdots \cup Y_k \quad \text{and} \quad Z = Z_1 \cup \cdots , Z_k,$$

and each of Y_i and Z_i is a disjoint copy of $[2d] \times \mathbb{F}_p$.

For notation convenience, we use a 4-tuple $x = (x_1, x_2, x_3, x_4)$ to represent a data point $x \in \mathcal{X}$, where $x_1 \in \{1, 2\}$ determines $x \in Y$ (when $x_1 = 1$) or $x \in Z$ (when $x_1 = 2$), $x_2 \in [k]$ indicates that x is from Y_{x_2} or Z_{x_2} (depending on the value of x_1), $x_3 \in [2d]$ and $x_4 \in \mathbb{F}_p$.

We will refer to a triple $q \in [k] \times [d] \times [b]$ as a *pointer* (looking ahead we will set $n = kdb$ in our reduction from the pointer chasing problem over $[n]$), and write $Q = [k] \times [d] \times [b]$ to denote the set of pointers. Let $B = \{1, \ldots, (kdb)^b\}$ be $(kdb)^b$ elements in $\mathbb{F}_p$; the choices do not matter. Given $|B| = (kdb)^b$, we view each element $a \in B$ as a tuple $(a_1, \ldots, a_b)$ of b pointers with $a_i \in Q$; this can be done by picking an arbitrary bijection between B and Q^b.

Hypothesis class. Our main challenge lies in the construction of the hypothesis class $\mathcal{H}$. First we review the Reed-Solomon code, which is a key building block of our construction.

Definition 5.3 (Reed-Solomon code)**.** *Let p be a prime number and let m, n be two positive integers that satisfy $n \leq m \leq p$. The* Reed-Solomon code $\mathsf{RS}_{n,m} : \mathbb{F}_p^n \to \mathbb{F}_p^m$ *maps $\mathbb{F}_p^n$ to $\mathbb{F}_p^m$ by setting*

$$(z_1, \ldots, z_m) = \mathsf{RS}_{n,m}(y_1, \ldots, y_n), \quad \textit{where } z_j = \sum_{i \in [n]} y_i f_j^{i-1}, \quad \forall j \in [m],$$

where f_j is the j-th element of $\mathbb{F}_p$ under an arbitrary ordering.

We need the following error correction guarantee of Reed-Solomon code.

Lemma 5.4. *For any distinct $y, y' \in \mathbb{F}_p^n$, $z = \mathsf{RS}_{n,m}(y) \in \mathbb{F}_p^m$ and $z' = \mathsf{RS}_{n,m}(y') \in \mathbb{F}_p^m$ satisfy*

$$\left| j \in [m] : z_j \neq z'_j \right| \geq \left(1 - \frac{n}{m}\right) m.$$

We can describe the hypothesis class $\mathcal{H}$ now. Each function $h_A : \mathcal{X} \to \{0, 1\}$ is specified by a tuple $A = (a_{i,j} \in B : i \in [2c+1], j \in [d])$. As discussed earlier, we view each $a_{i,j}$ in A as a tuple $(a_{i,j,\ell} : \ell \in [b])$ from Q^b. We use the following procedure to determine h_A over $\mathcal{X} = Y \cup Z$:

1. Initialization: Let $I : [k] \to [2c+1] \cup \{\mathsf{nil}\}$ and $J : [k] \to [2c+2] \cup \{\mathsf{nil}\}$ be two arrays such that all entries are set to nil. Let $q^{(1)} \in Q$ be a point set to be $(1, 1, 1)$.

2. For each $\tau = 2, \ldots, 2c+2$ (letting q denote the previous pointer $q^{(\tau-1)}$):

   (a) If τ is even and $I(q_1) = \mathsf{nil}$, we set $h_A(x)$ for all data points x in Y_{q_1} as follows (the fact that $I(q_1) = \mathsf{nil}$ makes sure that we have not set $h_A(x)$ for points in Y_{1,q_1} before). Let

   $$(z_1, \ldots, z_{2d}) = \mathsf{RS}_{d,2d}(a_{\tau-1,1}, \ldots, a_{\tau-1,d}) \in \mathbb{F}_p^{2d}. \tag{16}$$

   For each data point $x \in \mathcal{X}$ with $x_1 = 1$ and $x_2 = q_1$,

   $$h_A(x) = \begin{cases} 1 & \text{if } x_4 = z_{x_3} \\ 0 & \text{if } x_4 \neq z_{x_3} \end{cases}$$

   Recall that $x_3 \in [2d]$ and $x_4 \in \mathbb{F}_p$. We set $I(q_1) = \tau - 1$ (since h_A has been set for points in Y_{q_1} using $(a_{\tau-1,j} : j \in [d])$ from A). Before moving to the next loop, we set the new pointer $q^{(\tau)}$ to be $a_{\tau-1,q_2,q_3} \in Q$ (recall that $q_2 \in [d]$, $q_3 \in [b]$, and we view each $a_{\tau-1,q_2} \in B$ as a tuple from Q^b).

(b) If τ is even and $t = I(q_1) \neq \mathsf{nil}$, we don't need to set h_A for points in Y_{q_1} (since they have already been set using $(a_{t,j} : j \in [d])$). We just set the new pointer $q^{(\tau)} = a_{t,q_2,q_3}$.

(c) The cases with τ being odd is symmetric, with Y being replaced by Z and I being replaced by J. If τ is odd and $J(q_1) = \mathsf{nil}$, we set $h_A(x)$ for points in Z_{q_1} as follows: Let

$$(z_1, \ldots, z_{2d}) = \mathsf{RS}_{d,2d}(a_{\tau-1,1}, \ldots, a_{\tau-1,d}) \in \mathbb{F}_p^{2d}. \tag{17}$$

For each data point $x \in \mathcal{X}$ with $x_1 = 2$ and $x_2 = q_1$, we set

$$h_A(x) = \begin{cases} 1 & \text{if } x_4 = z_{x_3} \\ 0 & \text{if } x_4 \neq z_{x_3} \end{cases}$$

We set $J(q_1) = \tau - 1$ and set the new pointer $q^{(\tau)}$ to be $a_{\tau-1,q_2,q_3}$.

(d) If τ is odd and $t = J(q_1) \neq \mathsf{nil}$, we just set the new pointer $q^{(\tau)} = a_{t,q_2,q_3}$.

3. For any Y_i with $I(i) = \mathsf{nil}$ (so these data points have not been set yet), we set $h_A(x) = 1$ for all points in Y_i; similarly for any Z_i with $J(i) = \mathsf{nil}$, we set $h_A(x) = 1$ for all points in Z_i.

This finishes the description h_A and the hypothesis class $\mathcal{H}$.

We first observe that the VC dimension of $\mathcal{H}$ is bounded above by $O(cd)$.

Lemma 5.5. *The VC dimension of $\mathcal{H}$ is at most $4cd + 2c + 2d + 1$.*

Proof. For any subset $S \subseteq \mathcal{X}$ with size $|S| = 4cd + 2c + 2d + 2$, we prove that the hypothesis class $\mathcal{H}$ can not shatter S.

Case 1 $|S \cap Y| \geq (c+1)(2d+1) + 1$. Let $S_Y = \{x^{(1)}, \ldots, x^{((c+1)(2d+1)+1)}\} \subseteq S \cap Y$. First, if S_Y contains data points from at least $c+2$ blocks, that is, there exists $c+2$ indices $i_1, \ldots, i_{c+2}$ such that $Y_{i_1} \cap S_Y \neq \emptyset, \ldots, Y_{i_{c+2}} \cap S_Y \neq \emptyset$, then $\mathcal{H}$ can not shatter $(0, \ldots, 0)$. The reason is that for any function $h \in \mathcal{H}$, there are at most $(c+1)$ blocks contain label 0 data points, i.e., $|\{i \in [k] : I(i) \neq \mathsf{nil}\}| \leq c+1$. On the other hand, if S_Y contains data points from at most $(c+1)$ blocks, then by pigeonhole principle, there exists an index $i \in [k]$ such that $|S_Y \cap Y_i| \geq 2d+2$. Note that any function h in $\mathcal{H}$ either labels all data points in Y_i with 1, or labels at most $2d$ data points with 1, hence $\mathcal{H}$ can not shatter $(1, \ldots, 1, 0)$.

Case 2 $|S \cap Z| \geq c(2d+1) + 1$. This follows from a similar argument of the first case.

In summary, whenever $|S| \geq (2c+1)(2d+1) + 1 = 4cd + 2c + 2d + 2$, $\mathcal{H}$ can not shatter the set S. Hence the VC dimension of $\mathcal{H}$ is at most $4cd + 2c + 2d + 1$. □

5.3 Reduction from pointer chasing to proper learning

We prove Theorem 1.3 by reducing from the communication problem of pointer chasing.

Proof of Theorem 1.3. For notation convenience, we use ϵ, c, k', d' and b' to denote parameters in the statement of Theorem 1.3 (because we will use k, d and b to denote integers used to specify the hypothesis class $\mathcal{H}$). So $\epsilon \in (0, 1/4]$ and c, k', d', b' are positive integers such that $d' \geq C_0 c$ and $b' \geq C_0 \log(k'd'/(c\epsilon))$ for some sufficiently large constant C_0.

Let $\alpha = \lfloor 1/(4\epsilon) \rfloor$ and $k = k'\alpha = \Theta(k'/\epsilon)$. Let d be the largest integer such that

$$4cd + 2c + 2d + 2 \leq d'$$

so we have $d = \Theta(d'/c)$ given that $d' \geq C_0 c$. Finally, we choose b to be the largest integer such that the description size $\log_2 |\mathcal{X}|$ of hypothesis class $\mathcal{H}_{c,k,d,b}$ is at most b'. Given that

$$\log_2 |\mathcal{X}| = O\big(b \log(kdb)\big)$$

and $b' \geq C_0 \log(k'd'/(c\epsilon))$, we have that b is a positive integer that satisfies

$$b = \Omega\left(\frac{b'}{\log(k'd'b'/(c\epsilon))}\right).$$

With these choices of k, d and b, we use $\mathcal{H} = \mathcal{H}_{c,k,d,b}$ as the hypothesis class in the rest of the proof, which has VC dimension at most d' and description size at most b'. We prove below that any c-pass $(\epsilon, 0.01)$-continual learner for $\mathcal{H}$ over $2k'$ tasks must require $\widetilde{\Omega}(kdb/c^2)$ memory; the theorem follows since by our choices of k, d and b, we have

$$\frac{kdb}{c^2} = \Omega\left(\frac{k'}{\epsilon} \cdot \frac{d'}{c} \cdot \frac{b'}{\log(k'd'b'/(c\epsilon))} \cdot \frac{1}{c^2}\right) = \widetilde{\Omega}\left(\frac{k'd'b'}{c^3\epsilon}\right) = \frac{1}{c^2} \cdot \widetilde{\Omega}\left(\frac{k'd'b'}{c\epsilon}\right).$$

For this purpose, we show that any c-pass $(\epsilon, 0.01)$-continual learner for $\mathcal{H}$ over $2k'$ tasks with an m-bit memory can be used to obtain a protocol for the pointer chasing communication problem with parameters $n \coloneqq kdb$ and c (as in Definition 5.1) that has total communication $2cm$ and success probability 0.99. Our goal then follows from Theorem 5.2.

Reduction. With $n \coloneqq kdb$, we let ρ be the following bijection between $Q = [k] \times [d] \times [b]$ and $[n]$:

$$\rho(i, j, \ell) = (i-1)db + jb + \ell, \quad \text{for all } (i, j, \ell) \in Q.$$

with $\rho(1, 1, 1) = 1$.

On input $f_A : [n] \to [n]$, Alice creates k' data distributions $\mathcal{D}_1, \ldots, \mathcal{D}_{k'}$ over $\mathcal{X}$ as follows. First she views f_A as a tuple of pointers $(y_{i,j,\ell} \in Q : i \in [k], j \in [d], \ell \in [b])$, with

$$y_{i,j,\ell} = \rho^{-1}\Big(f_A\big(\rho(i, j, \ell)\big)\Big) \in Q.$$

She also views $y_{i,j} = (y_{i,j,1}, \ldots, y_{i,j,b}) \in Q^b$ as an element in B (and recall that B is a subset of $\mathbb{F}_p$). For each $i \in [k']$, the data distribution $\mathcal{D}_i$ is constructed as follows:

- The distribution $\mathcal{D}_i$ is supported on $Y_{(i-1)\alpha+1}, \ldots, Y_{i\alpha}$. For each $j : (i-1)\alpha + 1 \leq j \leq i\alpha$, $\mathcal{D}_i$ has probability $1/(2\alpha d)$ on each of the following $2d$ points in Y_j:

$$(1, j, 1, z_{j,1}), \ldots, (1, j, 2d, z_{j,2d}),$$

  where $(z_{j,1}, \ldots, z_{j,2d})$ come from

$$(z_{j,1}, \ldots, z_{j,2d}) = \mathsf{RS}_{d,2d}(y_{j,1}, \ldots, y_{j,d}) \in \mathbb{F}_p^{2d}.$$

  Note that there are $2\alpha d$ points so the probabilities sum to 1. All these points are labelled 1.

Similarly Bob uses $f_B : [n] \to [n]$ to create k' data distributions $\mathcal{D}_{k'+1}, \ldots, \mathcal{D}_{2k'}$; the only difference is that these distributions are supported on Z. We prove in the following claim that these $2k'$ data distributions are consistent with a function $h : \mathcal{X} \to \{0, 1\}$ in the hypothesis class $\mathcal{H}$.

Claim 5.6. *For any $f_A : [n] \to [n]$ and $f_B : [n] \to [n]$, there is a function h in $\mathcal{H}$ that is consistent with all $2k'$ data distributions $\mathcal{D}_1, \ldots, \mathcal{D}_{2k'}$.*

Proof. Let $w^{(1)}, \ldots, w^{(2c+2)}$ be defined in Definition 5.1 using f_A and f_B. Consider the function $h \in \mathcal{H}$ specified by the tuple $(a_{i,j} \in [B] : i \in [2c+1], j \in [d])$, where $a_{i,j} = y_{\rho^{-1}(w^{(i)})_1, j}$ for $i \in [2c+1], j \in [d]$. It is easy to verify that h is consistent with $\mathcal{D}_1, \ldots, \mathcal{D}_{2k'}$. $\square$

After creating these data distributions (with no communication so far), Alice and Bob simulate a c-pass $(\epsilon, 0.01)$-continual learner for $\mathcal{H}$ over $\mathcal{D}_1, \ldots, \mathcal{D}_{2k'}$. If the c-pass learner uses m-bit memory, the simulation can be done by a $(2c-1)$-round protocol with total communication $O(cm)$. At the end of the simulation, Bob recovers from the learner a function $h_A \in \mathcal{H}$ for some tuple $A = (a_{i,j} \in B : i \in [2c+1], j \in [d])$. With probability at least 0.99, we have

$$\ell_{\mathcal{D}_i}(h_A) \leq \epsilon, \quad \text{for all } i \in [2k']. \tag{18}$$

Let $q^{(1)}, \ldots, q^{(2c+2)} \in Q$ be the sequence of $2c+2$ pointers obtained by following the procedure on A described in the definition of the hypothesis class $\mathcal{H}$. The following claim shows that whenever h_A satisfies (18), we must have $\rho(q^{(\tau)}) = w^{(\tau)}$ for all $\tau \in [2c+2]$, where $w^{(1)}, \ldots, w^{(2c+2)}$ are defined in Definition 5.1 using f_A and f_B. As a result, to end the protocol, Bob just follows the procedure to compute $q^{(1)}, \ldots, q^{(2c+2)}$ and output $\rho(q^{(2c+2)}) \pmod 2$. This gives a $(2c-1)$-round communication protocol for the pointer chasing problem with success probability 0.99. This finishes the proof. $\square$

Claim 5.7. *When h_A satisfies (18), we have $\rho(q^{(\tau)}) = w^{(\tau)}$ and*

$$y_{q_1^{(\tau-1)}, j} = \begin{cases} a_{I(q_1^{(\tau-1)}), j} & \text{for even } \tau \\ a_{J(q_1^{(\tau-1)}), j} & \text{for odd } \tau \end{cases}$$

for all $\tau = 2, \ldots, 2c+2$ and $j \in [d]$.

Proof. We prove by induction on $\tau = 2, \ldots, 2c+2$. We start with the base case when $\tau = 2$. Note that $\rho(q^{(1)}) = 1 = w^{(1)}$ since $\rho(1,1,1) = 1$. We also have $I(q_1^{(1)}) = 1$. Assume for a contradiction

$$(a_{1,1}, \ldots, a_{1,d}) \neq (y_{1,1}, \ldots, y_{1,d}).$$

Then it follows from the error correction guarantee of Reed-Solomon code that at least d entries of

$$\mathsf{RS}_{d,2d}(a_{1,1}, \ldots, a_{1,d}) \quad \text{and} \quad \mathsf{RS}_{d,2d}(y_{1,1}, \ldots, y_{1,d})$$

are different. This implies that $\mathcal{D}_1$ is inconsistent with h_A on at least d points and thus,

$$\ell_{\mathcal{D}_1}(h_A) \geq \frac{d}{3\alpha d} \geq \frac{1}{2\alpha} > \epsilon,$$

a contradiction with (18). So $(a_{1,1}, \ldots, a_{1,d}) = (y_{1,1}, \ldots, y_{1,d})$. To prove $\rho(q^{(2)}) = w^{(2)}$, note that

$$q^{(2)} = a_{1,1,1} = y_{1,1,1} = \rho^{-1}(f_A(\rho(1,1,1))) = \rho^{-1}(f_A(1)) = \rho^{-1}\big(w^{(2)}\big).$$

This finishes the proof of the base case.

The induction step is similar. Assume that the statement holds for all $2, \ldots, \tau-1$ and we now prove it holds for $\tau \leq 2c+2$. Assume τ is even; the proof for odd τ is symmetric. Consider two cases. If in the τ-th loop of the procedure, $t = I(q_1^{(\tau-1)}) \neq \mathsf{nil}$, then $t < \tau$ and $q_1^{(\tau-1)} = q_1^{(t)}$. Hence

$$y_{q_1^{(\tau-1)}, j} = a_{I(q_1^{(\tau-1)}), j} \tag{19}$$

for all $j \in [d]$ follows from the inductive hypothesis. If in the τ-th loop, $I(q_1^{(\tau-1)}) = \mathsf{nil}$, then we set $I(q_1^{(\tau-1)}) = \tau - 1$ at the end of this loop and (19) follows from an argument similar to the base case using the error correction guarantee of Reed-Solomon code. So (18) holds in both cases.

In both cases, let $t \le \tau - 1$ be the final value of $I(q_1^{(\tau-1)})$. Then

$$q^{(\tau)} = a_{t, q_2^{(\tau-1)}, q_3^{(\tau-1)}} = y_{q_1^{(\tau-1)}, q_2^{(\tau-1)}, q_3^{(\tau-1)}} = \rho^{-1}\Big(f_A\big(\rho(q^{(\tau-1)})\big)\Big) = \rho^{-1}\big(w^{(\tau)}\big),$$

where the last step used the inductive hypothesis $\rho(q^{(\tau-1)}) = w^{(\tau-1)}$. This finishes the proof. $\square$

6 Discussion

The problem of continual, or lifelong, learning is a major and crucial open challenge for Machine Learning, a key roadblock in the field's quest to transcend the stereotype of individual specialized tasks and make small steps towards brain-like learning: robust, unsupervised, self-motivated, embodied in a sensory-motor apparatus and embedded in the world, where data collection is deliberate and the sum total of the animal's experience is applied to each new task. There seems to be a consensus in the field that the chief problem in extending the practical successes of Machine Learning in this direction is memory.

We formulated the problem of continual learning as a sequence of k PAC learning tasks followed by a test, and showed a devastating lower bound for memory: the memory requirements for solving the problem are essentially increased from that of individual tasks by a factor of k. That is, unless a new approach is discovered — or the PAC formalism is for some fundamental reason inadequate to model this variant of learning — continual learning is impossible, with respect to memory requirements, in the worst case.

It has been argued that worst-case lower bounds do not always predict the difficulty of making practical progress in computational problems, and that this is especially true in the context of Machine Learning. However, such lower bounds have a way of identifying the aspects of the problem that must be attended to in order to make progress, and pointing to alternative formulations that are more promising. We feel that a comprehensive understanding of the theoretical difficulties of continual learning is a prerequisite for making progress in this important front.

Our multi-pass approach and the ensuing upper and lower bounds were admittedly inspired by streaming — an algorithmic domain of a very different nature than continual learning. However, it does provide a nice demonstration of the power of improper learning. We believe that the multi-pass MWU-based algorithm developed here may point the way to new empirical approaches to continual learning, perhaps involving the periodic replay of past data, as well as competition between several evolving variants of the learning device mediated by boosting. Naturally, experiments will be needed to explore this direction.

References

[Aar04] Scott Aaronson. Limitations of quantum advice and one-way communication. In *Proceedings. 19th IEEE Annual Conference on Computational Complexity, 2004.*, pages 320–332. IEEE, 2004.

[AHK12] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of computing*, 8(1):121–164, 2012.

[BBCR13] Boaz Barak, Mark Braverman, Xi Chen, and Anup Rao. How to compress interactive communication. *SIAM Journal on Computing*, 42(3):1327–1363, 2013.

[BBF+21] Gavin Brown, Mark Bun, Vitaly Feldman, Adam Smith, and Kunal Talwar. When is memorization of irrelevant training data necessary for high-accuracy learning? In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 123–132, 2021.

[BBFM12] Maria Florina Balcan, Avrim Blum, Shai Fine, and Yishay Mansour. Distributed learning, communication complexity and privacy. In *Conference on Learning Theory*, pages 26–1, 2012.

[BBV15] Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. Efficient representations for lifelong learning and autoencoding. In *Conference on Learning Theory*, pages 191–210. PMLR, 2015.

[BHPQ17] Avrim Blum, Nika Haghtalab, Ariel D Procaccia, and Mingda Qiao. Collaborative pac learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2389–2398, 2017.

[BKMS21] Mark Braverman, Gillat Kol, Shay Moran, and Raghuvansh R Saxena. Near optimal distributed learning of halfspaces with two parties. In *Conference on Learning Theory*, pages 724–758. PMLR, 2021.

[BNS19] Amos Beimel, Kobbi Nissim, and Uri Stemmer. Characterizing the sample complexity of pure private learners. *J. Mach. Learn. Res.*, 20:146–1, 2019.

[BR11] Mark Braverman and Anup Rao. Information equals amortized communication. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 748–757, 2011.

[Bra15] Mark Braverman. Interactive information complexity. *SIAM Journal on Computing*, 44(6):1698–1739, 2015.

[CKDT20] Arslan Chaudhry, Naeemullah Khan, Puneet Dokania, and Philip Torr. Continual learning in low-rank orthogonal subspaces. *Advances in Neural Information Processing Systems*, 33, 2020.

[CLV21] Xinyuan Cao, Weiyang Liu, and Santosh S Vempala. Provable lifelong learning of representations. *arXiv preprint arXiv:2110.14098*, 2021.

[CRRE18] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations*, 2018.

[CZZ18] Jiecao Chen, Qin Zhang, and Yuan Zhou. Tight bounds for collaborative pac learning via multiplicative weights. *Advances in neural information processing systems*, 2018.

[DKS19] Yuval Dagan, Gil Kur, and Ohad Shamir. Space lower bounds for linear prediction in the streaming model. In *Conference on Learning Theory*, pages 929–954. PMLR, 2019.

[FAML20] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR, 2020.

[Fel20] Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.

[FX15] Vitaly Feldman and David Xiao. Sample complexity bounds on differentially private learning via communication complexity. *SIAM Journal on Computing*, 44(6):1740–1764, 2015.

[GMX+13] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[GRT18] Sumegha Garg, Ran Raz, and Avishay Tal. Extractor-based time-space lower bounds for learning. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 990–1002, 2018.

[Hau95] David Haussler. Sphere packing numbers for subsets of the boolean n-cube with bounded vapnik-chervonenkis dimension. *Journal of Combinatorial Theory, Series A*, 69(2):217–232, 1995.

[JPY12] Rahul Jain, Attila Pereszlényi, and Penghui Yao. A direct product theorem for the two-party bounded-round public-coin communication complexity. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 167–176. IEEE, 2012.

[KHD20] Jeremias Knoblauch, Hisham Husain, and Tom Diethe. Optimal continual learning has perfect memory and is np-hard. In *International Conference on Machine Learning*, pages 5327–5337. PMLR, 2020.

[Kla00] Hartmut Klauck. On quantum and probabilistic communication: Las vegas and one-way protocols. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 644–651, 2000.

[KLMY19] Daniel Kane, Roi Livni, Shay Moran, and Amir Yehudayoff. On communication complexity of classification problems. In *Conference on Learning Theory*, pages 1903–1943. PMLR, 2019.

[KPR+17] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[LPR17] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

[MC89] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[MMO95] James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

[NW91] Noam Nisan and Avi Widgerson. Rounds in communication complexity revisited. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 419–429, 1991.

[NZ18] Huy Nguyen and Lydia Zakynthinou. Improved algorithms for collaborative pac learning. *Advances in Neural Information Processing Systems*, 31:7631–7639, 2018.

[PKP+19] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[PR22] Binghui Peng and Andrej Risteski. Continual learning: a feature extraction formalization, an efficient algorithm, and fundamental obstructions. *https://arxiv.org/abs/2203.14383*, 2022.

[PS82] Christos H Papadimitriou and Michael Sipser. Communication complexity. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 196–200, 1982.

[PU16] Anastasia Pentina and Ruth Urner. Lifelong learning with weighted majority votes. *Advances in Neural Information Processing Systems*, 29, 2016.

[RAS+19] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[Raz18] Ran Raz. Fast learning requires good memory: A time-space lower bound for parity learning. *Journal of the ACM (JACM)*, 66(1):1–18, 2018.

[RE13] Paul Ruvolo and Eric Eaton. Ella: An efficient lifelong learning algorithm. In *International conference on machine learning*, pages 507–515. PMLR, 2013.

[RRD+16] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[RTCM+21] Amal Rannen-Triki, Arslan Chaudhry, Bogdan Mazoure, Xu He, Thang Doan, Rahaf Aljundi, and Vincenzo Lomonaco. Icml workshop on theory and foundation of continual learning. `https://sites.google.com/view/cl-theory-icml2021?pli=1&authuser=1/`, 2021.

[Sau72] Norbert Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1):145–147, 1972.

[Sch90] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[She72] Saharon Shelah. A combinatorial problem; stability and order for models and theories in infinitary languages. *Pacific Journal of Mathematics*, 41(1):247–261, 1972.

[SLKK17] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[SSV19] Vatsal Sharan, Aaron Sidford, and Gregory Valiant. Memory-sample tradeoffs for linear regression with small error. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 890–901, 2019.

[TM95] Sebastian Thrun and Tom M Mitchell. Lifelong robot learning. *Robotics and autonomous systems*, 15(1-2):25–46, 1995.

[Val84] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[VC15] Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*, pages 11–30. Springer, 2015.

[vdVST20] Gido M van de Ven, Hava T Siegelmann, and Andreas S Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1):1–14, 2020.

[VWW20] Santosh S Vempala, Ruosong Wang, and David P Woodruff. The communication complexity of optimization. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1733–1752. SIAM, 2020.

[Yeh20] Amir Yehudayoff. Pointer chasing via triangular discrimination. *Combinatorics, Probability and Computing*, 29(4):485–494, 2020.

A Missing proof from Section 2

Proof of Fact 2.9. One has

$$
\begin{aligned}
\sum_{i=1}^{n} I(A_i;B) &= \sum_{i=1}^{n} H(A_i) - H(A_i|B) \\
&\leq \sum_{i=1}^{n} H(A_i) - H(A_i|A_1,\dots,A_{i-1},B) \\
&= \sum_{i=1}^{n} H(A_i) - H(A_i|A_1,\dots,A_{i-1}) \\
&\quad + \sum_{i=1}^{n} H(A_i|A_1,\dots,A_{i-1}) - H(A_i|A_1,\dots,A_{i-1},B) \\
&= \sum_{i=1}^{n} H(A_i) - H(A_1,\dots,A_n) + \sum_{i=1}^{n} I(A_i;B|A_1,\dots,A_{i-1}) \\
&= \sum_{i=1}^{n} H(A_i) - H(A_1,\dots,A_n) + I(A_1,\dots,A_n;B).
\end{aligned}
$$

This concludes the proof. □

B Missing proof from Section 4

First, we provide the proof of Lemma 4.1.

Proof of Lemma 4.1. Fix an index $i \in [k]$ and a round $t \in [c]$, we prove the first and second claim by induction. The base case $\tau = 0$ holds trivially and suppose it continues to hold up to time $\tau - 1$, then

$$\mathbb{E}[|\overline{S}^q_{i,t,\tau}|] = \mathbb{E}[|S^q_{i,t,\tau} \cap \mathcal{X}_{i,t,\tau}|] = M_1 \cdot \Pr_{(x,y)\sim\mathcal{D}_i}[x \in \mathcal{X}_{i,t,\tau}] \geq (1-(1+\gamma)\epsilon_t)^{t-1} M_1 \geq \frac{3}{4} M_1,$$

where the first step follows from the definition of $\overline{S}^q_{i,t,\tau}$, the second step follows from the linearity of expectation, the third step holds due to the inductive hypothesis, and the last step holds as $(1-(1+\gamma)\epsilon_t)^{t-1} \geq 1-(1+\gamma)\epsilon_t \cdot (t-1) \geq \frac{3}{4}$.

Therefore, by Chernoff bound, one has

$$\Pr_{S^q_{i,t,\tau}\sim\mathcal{D}_i^{M_1}}[|S^q_{i,t,\tau} \cap \mathcal{X}_{i,t,\tau}| < M_1/2] \leq \exp(-M_1/36) \leq \frac{\delta}{60kc^2} \tag{20}$$

Now we condition on $|\overline{S}^q_{i,t,\tau}| = |S^q_{i,t,\tau} \cap \mathcal{X}_{i,t,\tau}| \geq M_1/2$. For any $\lambda \in [0,1]$, define the exact λ-quantile of $\{(x, \exp(\eta \sum_{\nu=1}^{\tau} \mathbf{1}[h_\nu(x) \neq y]))\}_{(x,y)\sim\mathcal{D}_i|x\in\mathcal{X}_{i,t,\tau}}$ to be $(x_{i,t,\tau,\lambda}, q_{i,t,\tau,\lambda})$. The EstimateQuantile returns the ϵ_t-quantile of the empirical sample, then we have that

$$\Pr\left[(x_{i,t,\tau}, \widehat{q}_{i,t,\tau}) \succeq (x_{i,t,\tau,(1+\gamma)\epsilon_t}, q_{i,t,\tau,(1+\gamma)\epsilon_t})\right] \leq \exp(-\epsilon_t\gamma^2 M_1/24) \leq \frac{\delta}{60kc^2}. \tag{21}$$

where the second step holds by Chernoff bound the fact that tie breaking only affects $o(\epsilon_0/c^3)$ quantile. Similarly,

$$\Pr\left[(x_{i,t,\tau}, \widehat{q}_{i,t,\tau}) \preceq (x_{i,t,\tau,(1-\gamma)\epsilon_t}, q_{i,t,\tau,(1-\gamma)\epsilon_t})\right] \leq \exp(-\epsilon_0\gamma^2 M_1/6) \leq \frac{\delta}{60kc^2}. \tag{22}$$

Combining Eq. (21), Eq. (22) and an union bound, we can conclude the proof the first claim. Note that

$$\mathcal{X}_{i,t,\tau}\backslash\mathcal{X}_{i,t,\tau+1} = \left\{x \in \mathcal{X}_{i,t,\tau} : (x, \exp(\eta\sum_{\tau=1}^{\tau}\mathbf{1}\{h_\tau(x) \neq y\})) \succeq (x_{i,t,\tau}, \widehat{q}_{i,t,\tau})\right\},$$

and therefore,

$$P_{i,t,\tau+1} \in \left[(1-(1+\gamma)\epsilon_t)\cdot P_{i,t,\tau}, (1-(1-\gamma)\epsilon_t)\cdot P_{i,t,\tau}\right].$$

Hence completing the proof of second claim. □

We then prove

Proof of Lemma 4.2. The first claim holds due to definition and therefore we focus on the second claim. We prove by induction on τ. Fix an index $i \in [k]$, the case of $\tau = 1$ holds trivially, as $\mathcal{X}_{i,1,1} = \cdots = \mathcal{X}_{i,t,1} = \mathcal{X}$ by definition. Suppose the induction holds up to time $\tau - 1$, then we prove $\mathcal{X}_{i,t+1,\tau} \subseteq \mathcal{X}_{i,t,\tau}$ holds for any $t \in [\tau : c-1]$. By definition, we have (i) $\mathcal{X}_{i,t+1,\tau} \subseteq \mathcal{X}_{i,t+1,\tau-1}$ and by induction, (ii) $\mathcal{X}_{i,t+1,\tau-1} \subseteq \mathcal{X}_{i,t,\tau-1}$. Condition on the event of Lemma 4.1, it suffices to prove for any data point $x \in \mathcal{X}_{i,t+1,\tau}$, it does not belong to the top $(1+\gamma)\epsilon_t$-quantile of $\mathcal{X}_{i,t,\tau-1}$.

First of all, we know data points in $\mathcal{X}_{i,t+1,\tau-1}\backslash\mathcal{X}_{i,t+1,\tau}$ has decent amount of probability, i.e.

$$\Pr_{(x,y)\sim\mathcal{D}_i}[x \in \mathcal{X}_{i,t+1,\tau-1}\backslash\mathcal{X}_{i,t+1,\tau}] \geq (1-\gamma)\epsilon_{t+1}\cdot P_{i,t+1,\tau-1}.$$

Combining (i) and (ii), these data points belong to $\mathcal{X}_{i,t,\tau-1}$, i.e.,

$$\mathcal{X}_{i,t+1,\tau-1}\backslash\mathcal{X}_{i,t+1,\tau} \subseteq \mathcal{X}_{i,t,\tau-1}.$$

Hence it suffices to prove data points in $\mathcal{X}_{i,t+1,\tau-1}\backslash\mathcal{X}_{i,t+1,\tau}$ take up at least $(1+\gamma)\epsilon_t$ portion of $\mathcal{X}_{i,t,\tau-1}$ (again, assuming the event of Lemma 4.1 holds), i.e.,

$$(1-\gamma)\epsilon_{t+1}\cdot P_{i,t+1,\tau-1} \geq (1+\gamma)\epsilon_t P_{i,t,\tau-1}. \tag{23}$$

This is true as

$$\frac{(1-\gamma)\epsilon_{t+1}}{(1+\gamma)\epsilon_t} \geq \left(\frac{1-(1-\gamma)\epsilon_t}{1-(1+\gamma)\epsilon_{t+1}}\right)^c \geq \left(\frac{1-(1-\gamma)\epsilon_t}{1-(1+\gamma)\epsilon_{t+1}}\right)^{\tau-1} \geq \frac{P_{i,t,\tau-1}}{P_{i,t+1,\tau-1}},$$

where the first step holds due to the choice of parameter (see Lemma B.1), the third step holds due to Lemma 4.1. Hence, we have proved Eq. (23) and concludes the proof. □

Next, we prove

Proof of Lemma 4.4. For any $i \in [k], t \in [c]$, according to the Definition 4.3:

$$\begin{aligned} w_{i,t} &= \sum_{x\in\mathcal{X}}\mathbf{1}[x \in \mathcal{X}_{i,t,t-1}]\cdot\mu_{\mathcal{D}_i}(x)\cdot\min\left\{\exp(\eta\sum_{\tau=1}^{t-1}\mathbf{1}[h_\tau(x) \neq y]), \widehat{q}_{i,t,t-1}\right\} \\ &= \mathop{\mathbb{E}}_{x\sim\mathcal{D}_i}\left[\mathbf{1}[x \in \mathcal{X}_{i,t,t-1}]\cdot\min\left\{\exp(\eta\sum_{\tau=1}^{t-1}\mathbf{1}[h_\tau(x) \neq y]), \widehat{q}_{i,t,t-1}\right\}\right]. \end{aligned}$$

Note each term

$$\mathbf{1}[x \in \mathcal{X}_{i,t,t-1}] \cdot \min\left\{\exp(\eta \sum_{\tau=1}^{t-1} \mathbf{1}[h_\tau(x) \neq y]), \widehat{q}_{i,t,t-1}\right\} \in [0, \widehat{q}_{i,t,t-1}], \tag{24}$$

condition on the event of Lemma 4.1, the expectation satisfies

$$\begin{aligned}
& \mathop{\mathbb{E}}_{x\sim\mathcal{D}_i}\left[\mathbf{1}[x \in \mathcal{X}_{i,t,t-1}] \cdot \min\left\{\exp(\eta \sum_{\tau=1}^{t-1} \mathbf{1}[h_\tau(x) \neq y]), \widehat{q}_{i,t,t-1}\right\}\right] \\
\geq & \mathop{\mathbb{E}}_{x\sim\mathcal{D}_i}\left[\mathbf{1}[x \in \mathcal{X}_{i,t,t-1}\backslash\mathcal{X}_{i,t,t}] \cdot \min\left\{\exp(\eta \sum_{\tau=1}^{t-1} \mathbf{1}[h_\tau(x) \neq y]), \widehat{q}_{i,t,t-1}\right\}\right] \\
= & \mathop{\mathbb{E}}_{x\sim\mathcal{D}_i} \mathbf{1}[x \in \mathcal{X}_{i,t,t-1}\backslash\mathcal{X}_{i,t,t}] \cdot \widehat{q}_{i,t,t-1} \\
= & (P_{i,t,t-1} - P_{i,t,t}) \cdot \widehat{q}_{i,t,t-1} \\
\geq & (1-\gamma)\epsilon_t P_{i,t,t-1}\widehat{q}_{i,t,t-1} \geq \frac{1}{2}\epsilon_t \widehat{q}_{i,t,t-1}.
\end{aligned} \tag{25}$$

The second step holds as every $x \in \mathcal{X}_{i,t,t-1}\backslash\mathcal{X}_{i,t,t}$ satisfies $\exp(\eta\sum_{\tau=1}^{t-1} \mathbf{1}[h_\tau(x) \neq y]) \geq \widehat{q}_{i,t,t-1}$ (see the definition at Eq. (11)), the third step holds due to the definition of $P_{i,t,t-1}$ and $P_{i,t,t}$, the fourth step holds due to Lemma 4.1 and the last holds as $(1-\gamma)P_{i,t,t-1} \geq (1-\gamma)(1-(1+\gamma)\epsilon_t)^{t-1} \geq \frac{1}{2}$.

Recall in the procedure of EstimateWeight, we draw $M_2 = \Omega(\log(kc/\delta)/\epsilon_0\alpha^2)$ samples from $\mathcal{D}_i$ to estimate $w_{i,t}$, hence we have

$$\begin{aligned}
& \Pr\left[|\widehat{w}_{i,t} - w_{i,t}| \geq \frac{\alpha}{8} w_{i,t}\right] \\
= & \Pr\left[\left|\frac{1}{|M_2|}\sum_{(x,y)\in S^w_{i,t}\cap\mathcal{X}_{i,t,t-1}} \min\{\exp(\eta\sum_{\tau=1}^{t-1}\mathbf{1}[h_\tau(x)\neq y]), \widehat{q}_{i,t,t-1}\} - w_{i,t}\right| \geq \frac{\alpha}{8}w_{i,t}\right] \\
\leq & 2\exp(-M_2\alpha^2\epsilon_t/384) \leq \frac{\delta}{20kc}.
\end{aligned}$$

where the first step holds due to the definition of $\overline{S}^w_{i,t}$, the second step holds due to Chernoff bound, Eq. (24) and Eq. (25), the last step follows from the choice M_2.

Using an union bound over all $i \in [k]$, we further have

$$\widehat{p}_{i,t} = \frac{\widehat{w}_{i,t}}{\sum_{i'=1}^k \widehat{w}_{i',t}} \in (1 \pm \frac{\alpha}{3})\frac{w_{i,t}}{\sum_{i'=1}^k w_{i',t}} = (1 \pm \frac{\alpha}{3})p_{i,t}$$

holds with probability at least $1 - \frac{\delta}{20c}$. We finish the proof by applying an union bound over $t \in [c]$. □

We next analyse the TruncatedRejectionSampling procedure.

Proof of Lemma 4.5. The first claim follows directly from the TruncatedRejectionSampling procedure. In particular, it rejects element from $\mathcal{X}\backslash\mathcal{X}_{i,t,t-1}$, and for element $x \in \mathcal{X}_{i,t,t-1}$, it accepts with probability $\min\{\exp(\eta\sum_{\nu=1}^{t-1}\mathbf{1}[h_\nu(x) \neq y]), \widehat{q}_{i,t,t-1}\}/\widehat{q}_{i,t,t-1}$.

For the second claim, condition on the event of Lemma 4.1, we first bound the expected overhead of TruncatedRejectionSampling. Note TruncatedRejectionSampling would not reject element in $X_{i,t,t-1}\backslash X_{i,t,t}$ and

$$\Pr_{x\sim\mathcal{D}_i}[x \in \mathcal{X}_{i,t,t}\backslash\mathcal{X}_{i,t,t-1}] \geq (1-\gamma)\epsilon_t P_{i,t,t-1} \geq (1-\gamma)(1-(1+\gamma)\epsilon_t)^{t-1}\epsilon_t \geq \frac{1}{2}\epsilon_0,$$

where the first two steps hold due to Lemma 4.1 and the last step holds due to the choice of ϵ_t. Hence, the overhead from rejection sampling is at most $2/\epsilon_0$ (in expectation).

Fix a round $t \in [c]$, for any $i \in [k]$, the expected number of samples drawn from $\widehat{D}_{i,t}$ equals $n_{i,t} := \frac{\widehat{w}_{i,t}}{\sum_{i' \leq i} \widehat{w}_{i',t}} \cdot N$ and it satisfies

$$\sum_{i=1}^{k} n_{i,t} = N \sum_{i=1}^{k} \frac{\widehat{w}_{i,t}}{\sum_{i' \leq i} \widehat{w}_{i',t}} \leq N + N \sum_{i=2}^{k} \log \frac{\sum_{i' \leq i} \widehat{w}_{i',t}}{\sum_{i' \leq i-1} \widehat{w}_{i',t}}$$
$$\leq \log \frac{\sum_{i \in [k]} \widehat{w}_{i,t}}{\widehat{w}_{1,t}} \leq O(N \log(k/\epsilon))$$

where the first step follows from the definition, the second step follows from $\frac{a-b}{a} \leq \log \frac{a}{b}$, the last step holds as $\widehat{w}_{1,t} \geq P_{1,t,t-1} \geq \frac{1}{2}$ (Lemma 4.1) and

$$\widehat{w}_{i,t} \leq \exp(\eta \sum_{\tau=1}^{t-1} \mathbf{1}[h_\tau(x) \neq y]) \leq \exp((c-1) \cdot \eta) = O\left(\frac{k^2}{\epsilon^2}\right)$$

holds for any $i \in [k]$.

Via a simple application of Chernoff bound, one can prove with probability at least $1 - \frac{\delta}{20c}$, (1) the total number of sample drawn from the truncated distribution $\{\mathcal{D}_{i,t,\mathsf{trun}}\}_{i \in [k]}$ is at most $O(N \log(kdc/\epsilon\delta))$, (2) the overhead of TruncatedRejectionSampling procedure never exceeds $O(c \log(kdc/\epsilon\delta)/\epsilon)$. Using an union bound, the total number of sample being drawn for the training set $\{S_t\}_{t \in [c]}$ are bounded by

$$O(N \log(kdc/\epsilon\delta)) \cdot O(c \log(kdc/\epsilon\delta)/\epsilon) \cdot c \leq O\left(\frac{dc^2 \log^3(kdc/\epsilon\delta)}{\epsilon\alpha}\right).$$

We conclude the proof here. □

Next, we prove Lemma 4.6 and Lemma 4.8.

Proof of Lemma 4.6. For any $t \in [c]$, and for any training sample preserved at the end, it comes from the i-th task with probability $\frac{\widehat{w}_{i,t}}{\sum_{i' \leq i} \widehat{w}_{i',t}} \cdot \prod_{i' \geq i+1} \frac{\sum_{i'' < i'} \widehat{w}_{i'',t}}{\sum_{i'' \leq i} \widehat{w}_{i'',t}} = \frac{\widehat{w}_{i,t}}{\sum_{i' \in [k]} \widehat{w}_{i',t}} = \widehat{p}_{i,t}$. For any training sample comes from the i-th task, by Lemma 4.5, it follows from the distribution $\mathcal{D}_{i,t,\mathsf{trun}}$. □

Proof of 4.8. The algorithm draws $N = O(\frac{d+\log(kc/\delta)}{\alpha})$ samples from $\widehat{\mathcal{D}}_{t,\mathsf{trun}}$ and runs ERM on it. Due to the realizable assumption, with probability at least $1 - \frac{\delta}{20c}$, the output hypothesis satisfies

$$\ell_{\mathcal{D}_{t,\mathsf{trun}}}(h_t) \leq \ell_{\widehat{\mathcal{D}}_{t,\mathsf{trun}}}(h_t) + \frac{\alpha}{3} \leq \frac{\alpha}{2},$$

where the first step follows from Lemma 4.7, the seconds step holds due to the VC theory (see Lemma 2.4) □

Lemma B.1 (Technical lemma). *Let $\epsilon \in (0, 1/10), c \geq 1$, $\gamma = \frac{1}{10c^2}$, $\epsilon_t = (1 + \frac{1}{c})^t \cdot \frac{\epsilon}{20c}$, then*

$$\frac{(1-\gamma)\epsilon_{t+1}}{(1+\gamma)\epsilon_t} \geq \left(\frac{1-(1-\gamma)\epsilon_t}{1-(1+\gamma)\epsilon_{t+1}}\right)^c.$$

Proof. The RHS satisfies

$$\begin{aligned}
\left(\frac{1-(1-\gamma)\epsilon_t}{1-(1+\gamma)\epsilon_{t+1}}\right)^c &= \left(1+\frac{\gamma\epsilon_{t+1}+\gamma\epsilon_t+\epsilon_{t+1}-\epsilon_t}{1-(1+\gamma)\epsilon_t}\right)^c \\
&\leq \left(1+\frac{\epsilon}{c^2}\right)^c \\
&\leq \exp(\frac{\epsilon}{c}) \leq 1+\frac{2\epsilon}{c}
\end{aligned}$$

The second step holds as $\gamma\epsilon_{t+1} \leq \frac{\epsilon}{30c^3}, \gamma\epsilon_t \leq \frac{\epsilon}{30c^3}$, $\epsilon_{t+1}-\epsilon_t \leq \frac{\epsilon}{3c^2}$ and $1-(1+\gamma)\epsilon_{t+1} \geq 1/2$, the second step follows from $1+x \leq \exp(x)$, the third step follows from $\exp(x) \leq 1+2x$ when $x \in (0, 1/10)$.

Meanwhile, the LHS satisfies

$$\frac{(1-\gamma)\epsilon_{t+1}}{(1+\gamma)\epsilon_t} = (1+\frac{1}{c}) \cdot \frac{10c^2-1}{10c^2+1} \geq 1+\frac{2\epsilon}{c}.$$

We complete the proof. $\square$