SEQUENTIAL LEARNING OF NEURAL NETWORKS FOR PREQUENTIAL MDL

Anonymous authors
Paper under double-blind review

ABSTRACT

Minimum Description Length (MDL) provides a framework and an objective for principled model evaluation. It formalizes Occam's Razor and can be applied to data from non-stationary sources. In the prequential formulation of MDL, the objective is to minimize the cumulative next-step log-loss when sequentially going through the data and using previous observations for parameter estimation. It thus closely resembles a continual- or online-learning problem. In this study, we evaluate approaches for computing prequential description lengths for image classification datasets with neural networks. Considering the computational cost, we find that online-learning with rehearsal has favorable performance compared to the previously widely used block-wise estimation. We propose *forward-calibration* to better align the models predictions with the empirical observations and introduce *replay-streams*, a minibatch incremental training technique to efficiently implement approximate random replay while avoiding large in-memory replay buffers. As a result, we present description lengths for a suite of image classification datasets that improve upon previously reported results by large margins.

1 INTRODUCTION

Within the field of deep learning, the paradigm of *empirical risk minimization* (ERM, Vapnik (1991)) together with model and hyper-parameter selection based on held-out data is the prevailing training and evaluation protocol. This approach has served the field well, supporting seamless scaling to large model and data set sizes. The core assumptions of ERM are: a) the existence of fixed but unknown distributions $q(x)$ and $q(y|x)$ that represent the data-generating process for the problem under consideration; b) the goal is to obtain a function $\hat{y} = f(x, \theta^*)$ that minimizes some loss $\mathcal{L}(y, \hat{y})$ in expectation over the data drawn from q; and c) that we are given a set of (i.i.d.) samples from q to use as training and validation data. Its simplicity and well understood theoretical properties make ERM an attractive framework when developing learning machines.

However, sometimes we wish to employ deep learning techniques in situations where not all these basic assumptions hold. For example, if we do not assume a fixed data-generating distribution q we enter the realm of *continual learning*, *life long learning* or *online learning*. Multiple different terms are used for these scenarios because they operate under different constraints, and because there is some ambiguity about what problem exactly a learning machine is supposed to solve. A recent survey on continual- and life-long learning for example describes multiple, sometimes conflicting desiderata considered in the literature: forward-transfer, backward-transfer, avoiding catastrophic forgetting and maintaining plasticity (Hadsell et al., 2020).

Another situation in which the ERM framework is not necessarily the best approach is when minimizing the expected loss $\mathcal{L}$ is not the only, or maybe not even the primary objective of the learning machine. For example, recently deep-learning techniques have been used to aid structural and causal inference (Vowels et al., 2021). In these cases, we are more interested in model selection or some aspect of the learned parameters θ^* than the generalization loss $\mathcal{L}$. Here we have little to gain from the generalization bounds provided by the ERM framework and in fact some of its properties can be harmful.

Independent of ERM, compression based approaches to inference and learning such as Minimum Description Length (Rissanen, 1984; Grunwald, 2004), Minimum Message Length (Wallace, 2005)

and Solomonoffs theory of inductive inference (Solomonoff, 1964; Rathmanner & Hutter, 2011) have been extensively studied. In practice, however, they have been primarily applied to small scale problems and with very simple models compared to deep neural networks. These approaches strive to find and train models that can compress the observed data well and rely on the fact that such models have a good chance of generalizing on future data. It is considered a major benefit that these approaches come with a clear objective and have been studied in detail even for concrete sequence of observations; without assuming stationarity or a probabilistic generative process at all (also called the *individual sequence* scenario). The individual sequence setting is problematic for ERM because the fundamental step of creating training and validation splits is not clearly defined. In fact, creating such splits implies making an assumption about what is equally distributed across training and validation data. But creating validation splits can be problematic even with (assumed) stationary distributions, and this becomes more pressing as ML research moves from curated academic benchmark data sets to large user provided or web-scraped data. A discussion and examples can be found in A.3.

In contrast to ERM, compression approaches furthermore include a form of Occam's Razor, i.e. they prefer simpler models (according to some implicit measure of complexity) as long as the simplicity does not harm the model's predictive performance.

The literature considers multiple, subtly different approaches to defining and computing description lengths. Most of them are intractable and known approximations break down for overparameterized model families such as neural networks. One particular approach however, prequential MDL (Dawid & Vovk, 1999; Poland & Hutter, 2005), turns computing the description length $L(\mathcal{D}|M) = -\log p(\mathcal{D}|M)$ into a specific kind of continual or online learning problem: $\log p(\mathcal{D}|M) := \sum_{t=1}^{T} \log p_M(y_t|x_t, \hat{\theta}(\mathcal{D}_{<t}))$, where $\mathcal{D} = \{(x_t, y_t)\}_1^T$ is the sequence of inputs x_t and associated prediction targets y_t; $p_M(y|x, \theta)$ denotes a model family M and $\hat{\theta}(\mathcal{D}_{<t})$ an associated parameter estimator given training data $\mathcal{D}_{<t}$. In contrast to ERM which considers the performance of a learning algorithm on held-out data after training on a fixed dataset, prequential MDL evaluates a learner by its generalization performance on y_t after trained on initially short but progressively longer sequences of observations $\mathcal{D}_{<t}$. This resembles an online learning problem where a learner is sequentially exposed to new data (x_t, y_t), however a sequential learner is also allowed to revisit old data $x_{t'}, y_{t'}$ for training and when making a prediction at time $t > t'$.

Contributions. Previous work on computing prequential description lengths with neural networks relied on a block-wise (chunk-incremental) approximation: at some positions t a model is trained from random initialization to convergence on data $\mathcal{D}_{<t}$ and then their prediction losses on the next intervals are combined (Blier & Ollivier, 2018; Bornschein et al., 2020; Jin et al., 2021; Bornschein et al., 2021; Perez et al., 2021; Whitney et al., 2020). We investigate alternatives that are inspired by continual-learning (CL) based methods. In particular, chunk-incremental and mini-batch incremental fine-tuning with rehearsal. Throughout this empirical study, we consider the computational costs associated with these methods.

We propose two new techniques: *forward-calibration* and *replay streams*. *Forward-calibration* improves the results by making the model's predictive uncertainty better match the observed distribution. *Replay streams* makes replay for mini-batch incremental learning more scalable by providing approximate random rehearsal while avoiding large in-memory replay-buffers. We identify exponential moving parameter averages, label-smoothing and weight-standardization as generally useful techniques. As a result we present description lengths for a suite of popular image classification datasets that improve upon the previously reported results (Blier & Ollivier, 2018; Bornschein et al., 2020) by large margins.

The motivation for studying prequential MDL stems from the desire to apply deep-learning techniques in situations violating the assumptions of ERM. In this work however we concentrate on established iid. datasets: stationarity is most challenging for a sequential learner, which has to uniformly learn from all previously examples, and it allows us to evaluate well known architectures and compare their held-out performance against ERM-based training, which has been the focus of the community for decades. We include additional experimental results on non-stationary data in Appendix C.7.3.

An example: model selection using ERM v.s. MDL. To showcase the properties of prequential MDL and its bias towards simpler models, we generate images by randomly drawing 3 MNIST examples and superimposing them as red(R), green(G) and blue(B) channels. The label is always

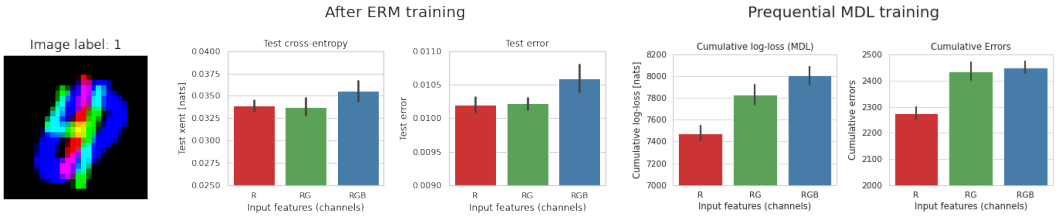


Figure 1: **Left:** Example images. **Center:** Best performing ERM-trained models on held-out data. There is no clear indication which conditioning should be preferred. **Right:** Log-loss and errors accumulated throughout the sequence. We observe that MDL based model selection provides strong evidence that the model conditioned on the R-channel alone should be preferred. In all cases we randomly sampled 50 hyperparameters and use bootstrap-sampling to obtain 95% confidence intervals for the performance of the best performing model. See Figure 2 for detailed regret plots.

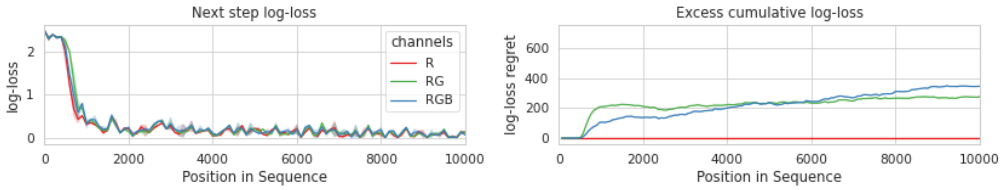


Figure 2: **Left:** Next-step neg. log-loss for the best performing models as a function of the position. The models are trained on all the previous examples. We observe that the models conditioned on the R, R and G, or R, G and B channels all rapidly improve their predictive performance when trained on more data. **Right:** The cumulative neg. log-loss (= description length) relative to the best performing model which is conditioned on the R channel only. During the first $\approx$ 2k examples the models conditioned on irrelevant channels accumulate hundreds of excess nats description length.

associated with the red channel; green and blue are distractors. We train a LeNet (LeCun et al., 1989) convolutional neural network on 100k examples from this dataset; either within the ERM framework by drawing i.i.d. examples, training to convergence and validating on held-out data, or within the MDL framework by treating the data as a sequence, training online with replay buffer (see Section 2 for details) and recording the (cumulative) next-step prediction losses and errors. We compare 3 different models: a model conditioned only on the R channel, on the R and G, or on R, G and B channels. Figure 1 (middle) shows the validation-set performance of the best performing models after ERM training. Figure 1 (right) shows description lengths on the same data. It is evident that ERM does not give a clear indication which conditioning should be preferred. However, MDL strongly prefers the model conditioned only on the R-channel. Figure 2 shows a more detailed analysis of the three differently conditioned models in the MDL scenario. We observe that prequential MDL identifies the minimal conditioning because it takes the small-data performance of the models into account. Given sufficient training data the three alternatives perform almost indistinguishable (as seen from their almost parallel regret curves). When we treat the description lengths as evidence for Bayesian model selection to choose between $M_{\mathrm{R}}, M_{\mathrm{RG}}$ and M_{RGB} we obtain $p(M_{\mathrm{R}}|\mathcal{D}) = {}^{p(\mathcal{D}|M_{\mathrm{R}})}/_{p(\mathcal{D}|M_{\mathrm{R}}) + p(\mathcal{D}|M_{\mathrm{RG}}) + p(\mathcal{D}|M_{\mathrm{RGB}})}$. With the description length $-\log p(\mathcal{D}|M_{\mathrm{R}})$ about 300 nats smaller than the ones for M_{RG} and M_{RGB}, we have $\approx 1{:}e^{-300}$ odds in favour of M_{R}. The Appendix C.1 contains additional experiments with more architectures and the experimental details; demonstrating that this is a robust result regardless of architectures and optimization hyperparameters.

2 Online Learning Approaches to Prequential MDL Estimation

In this section, we describe practical strategies for computing prequential description lengths with neural networks. Conceptually, computing the cumulative log-loss resembles a continual- or online-learning problem without constraints such as limiting access to previously observed data. In practice, however, some naive approaches such as retraining from scratch to convergence whenever a new example arrives are infeasible due to their high resource usage. In our study, we consider the compute requirements of each approach by counting the number of floating-point operations (FLOPs) until the learner reaches the end of the data sequence. We do not limit the learner's ability to access or sample

Algorithm 1 Mini-batch Incremental Training with Replay Streams

Require: data $(x_t, y_t)_{t=1}^{T}$; augmentation $Aug(.)$; number of replay streams K; EMA decay α

Initialize: parameters θ; EMA parameters $\bar{\theta} = \theta$; softmax temperature calibration $\beta = 1$; Replay positions $\{\rho_k = 1\}_{k=1}^{K}$; $\mathcal{L}_{preq} = 0$
for $t = 1$ **to** T **do**
  Compute next-step loss: $\mathcal{L}_t := -\log p(y_t|x_t, \bar{\theta}, \beta)$
  Update cumulative loss: $\mathcal{L}_{preq} \leftarrow \mathcal{L}_{preq} + \mathcal{L}_t$
  Update temperature calibration: $\beta \leftarrow \beta - \nabla_\beta \log p(y_t|x_t, \bar{\theta}, \beta)$
  Update parameters: $\theta \leftarrow \theta - \nabla_\theta \log p(y_t|\tilde{x}_t, \theta, 1.)$, with $\tilde{x}_t = Aug(x_t)$
  Update EMA parameters: $\bar{\theta} \leftarrow (1-\alpha)\bar{\theta} + \alpha\,\theta$
  for $k = 1$ **to** K **do**
    Get data from stream k: $x \leftarrow x_{\rho_k}$; $y \leftarrow y_{\rho_k}$, advance position $\rho_k \leftarrow \rho_k + 1$
    Update parameters: $\theta \leftarrow \theta - \nabla_\theta \log p(y|\tilde{x}, \theta, 1.)$ with $\tilde{x} = Aug(x)$
    Update EMA parameters: $\bar{\theta} \leftarrow (1-\alpha)\bar{\theta} + \alpha\,\theta$
    Maintain replay distribution: with probability p_{reset} reset $\rho_k \leftarrow 1$ (see Equation (1))
  end for
end for
return $\mathcal{L}_{preq}$

In practice we perform the algorithm with mini-batches instead of individual examples and use gradient based optimizers such as AdamW(Loshchilov & Hutter, 2019) instead of plain SGD.

previously observed data because storing and reading from large datasets is generally not considered a major technical bottleneck. However, in-memory (RAM) based implementations of replay-buffers for continual learning (rehearsal) suffer from limited capacity because data sets for training deep neural networks often exceed RAM sizes. Below we describe *replay-streams*, a simple approach to approximate random sampling for replay while only utilizing cheap sequential access to data on disk. With this approach, unlimited approximate random rehearsal from data on permanent storage becomes as easy as sampling from large datasets for ERM based optimization.

We compare the following approaches to compute prequential description lengths:

Chunk Incremental / From-Scratch (CI/FS): Variants of this approach have been used to compute the recently reported description lengths for deep-learning models (Blier & Ollivier, 2018; Bornschein et al., 2020; Jin et al., 2021; Bornschein et al., 2021; Perez et al., 2021). The data-sequence is partitioned into K non-overlapping intervals; typically of increasing size (i.e. choosing exponentially spaced split points $\{s_k\}_{k=1}^{K}$, with $s_k \in [2, \ldots, n]$, $s_k < s_{k+1}$ and $s_K = N$). For each k, a neural network is randomly initialized and trained to convergence on all the examples before the split point s_k and evaluated on the data between s_k and s_{k+1}. This corresponds to the block-wise approximation of the area under the curve: $\sum_{i=1}^{N} \log p(y_i|\hat{\theta}(\mathcal{D}_{<i})) \approx \sum_{k=1}^{K-1} \sum_{j=s_k}^{s_{k+1}} \log p(y_j|\hat{\theta}(\mathcal{D}_{<s_k}))$, where $\hat{\theta}(\mathcal{D})$ denotes parameters after training on data $\mathcal{D}$. To ensure that the model produces calibrated predictions even when $\mathcal{D}_{<s_k}$ is small, we use softmax temperature calibration (Guo et al., 2017): at each stage we split the data $\mathcal{D}_{<s_k}$ into a 90% training and a 10% calibration data. Conceptually, we could perform post-calibration by first training the network to convergence and then, with all parameters frozen, replacing the output layer $\mathrm{softmax}(h)$ with the calibrated output layer $\mathrm{softmax}(\mathrm{softplus}(\beta)h)$, where β is a scalar parameter chosen to minimize the loss on calibration data. In practice, we optimize β by gradient descent in parallel with the other model parameters. We alternate computing gradient steps for θ, calculated from the training set and using the uncalibrated network (with final layer $\mathrm{softmax}(h)$), with a gradient step on β, calculated from the validation set using the calibrated network (with final layer $\mathrm{softmax}(\mathrm{softplus}(\beta)h)$). This simple calibration procedure has proven to be surprisingly effective at avoiding overfitting symptoms when training large neural networks on small datasets (Guo et al., 2017; Bornschein et al., 2020).

Chunk Incremental / Continual Fine-tuning (CI/CF): Similar to CI/FS, the sequence is split into increasingly larger chunks. But instead of training the model from scratch at each stage, the network is continuously fine-tuned on the now larger dataset $\mathcal{D}_{<s_k}$. We use the same calibration strategy as for CI/FS. We expect to save compute by avoiding training from scratch. However, recent research suggests that, when first trained on data $\mathcal{D}_A$ and then trained on data $\mathcal{D}_A \cup \mathcal{D}_B$, deep-learning models

struggle to converge to a solution that generalizes as well as training on $\mathcal{D}_A \cup \mathcal{D}_B$ from scratch; even when both $\mathcal{D}_A$ and $\mathcal{D}_B$ are sampled from the same underlying distribution. In Ash & Adams (2020) the authors observe that shrinking the model-parameters and adding some noise after training on $\mathcal{D}_A$ can improve the final generalization performance. We therefore run an ablation study and perform *shrink & perturb* operation whenever advancing to the next chunk k.

Mini-batch Incremental / Replay-Buffer (MI/RB) This approach uses continual online mini-batch gradient descent with an additional replay buffer to store previously seen examples. At each time t, the learner performs a number of learning steps on data in the replay buffer. This could be all or a subset of the data $\mathcal{D}_{<t}$ depending on the capacity of the replay. We propose *forward-calibration* to optimize the calibration parameter β: each new batch of examples is first used for evaluation, then used to perform a single gradient step to optimize β, and finally used for training the parameters θ and placed into the replay-buffer. Calibration is computationally almost free: the forward-pass for the calibration gradient step can be shared with the evaluation forward-pass, and backward-propagation for β ends right after the softmax layer. Appendix B.1 contains a more detailed description. If the replay buffer is limited by the memory capacity, we either use a FIFO replacement-strategy or reservoir-sampling (Vitter, 1985) to maintain a fixed maximum buffer size.

Mini-batch Incremental / Replay-Streams (MI/RS) Implementing large replay buffers can be technically challenging: in-memory (RAM) replay buffers have limited capacity and random access and sampling from permanent storage is often slow. We propose a simple yet effective alternative to storing replay data in-memory. We instead assume that the data is stored in its original order on permanent storage and we maintain K replay streams (pointers into the sequence) , each reading the data in-order from the sequence $(x_1, y_1) \ldots (x_T, y_T)$. We can think of a replay-stream as a file-handle, currently at position $\rho \in 1 \ldots T$. A read operation yields (x_ρ, y_ρ) and increments the position to $\rho \leftarrow \rho + 1$. We denote the position of the kth replay-stream with ρ_k. Every time the learner steps forward from time t to $\tilde{t}{=}t{+}1$ we also read a replay example from each of the K replay-streams and perform gradient steps on them. By stochastically resetting individual streams with probability p_{reset} to position $\rho \leftarrow 1$, we can influence the distribution of positions ρ. If we wish to use uniform replay of previous examples we reset with probability $p_{\text{reset}} = 1/\tilde{t}$. If instead we wish to prioritize recent examples and replay them according to $p_{\text{replay}}(a)$, where $a{=}\tilde{t}{-}i$ is the age of the example (x_i, y_i) relative to the current position of the learner $\tilde{t}$, we use

$$p_{\text{reset}}(\tilde{t}) = \frac{\sum_{a=t}^{\tilde{t}} p_{\text{replay}}(a)}{\sum_{a'=0}^{\tilde{t}} p_{\text{replay}}(a')} \tag{1}$$

Note that p_{replay} can be an unnormalized distribution and that by setting $p_{\text{replay}} \propto 1$ we recover $p_{\text{reset}} = 1/\tilde{t}$ for uniform replay. See Algorithm 1 for details.

3 EXPERIMENTS

We empirically evaluate the approaches from Section 2 on a suite of models and image classification data sets. We use MNIST (LeCun et al., 2010), EMNIST, CIFAR-10, CIFAR-100 (Krizhevsky, 2009) and ImageNet (Russakovsky et al., 2015) and randomly shuffle each into a fixed sequence of examples. With MDL, we do not require validation or test set and could merge it with the training data. But because the data is known to be i.i.d. and we compare results with established ERM-based training, we instead maintain the standard test sets as held-out data. It must be emphasized that the primary objective throughout this study is to minimize the description length. All hyperparameters are tuned towards that goal. When we report test set performance it should be considered a sanity check. For more detailed analyses we will plot the "*regret*" a method M_i accumulates relative to some comparator method M_c: $R_i(t) := -\log p(\mathcal{D}_{\leq t}|M_i) + \log p(\mathcal{D}_{\leq t}|M_c)$. We evaluate the following architectures: MLPs, VGG (Simonyan & Zisserman, 2015), ResNets (He et al., 2016), WideResNets (Zagoruyko & Komodakis, 2016) and the transformer-based DeiT (Touvron et al., 2021) architecture. We refer to a variant of the VGG architecture with weight-standardized convolutions (Qiao et al., 2019) and with batch normalization (Ioffe & Szegedy, 2015) as *VGG++*.

For the bulk of our experiments, we use the AdamW (Loshchilov & Hutter, 2019) as optimizer; a series of control experiments however suggest that the quantitative results generalize to optimizers such as

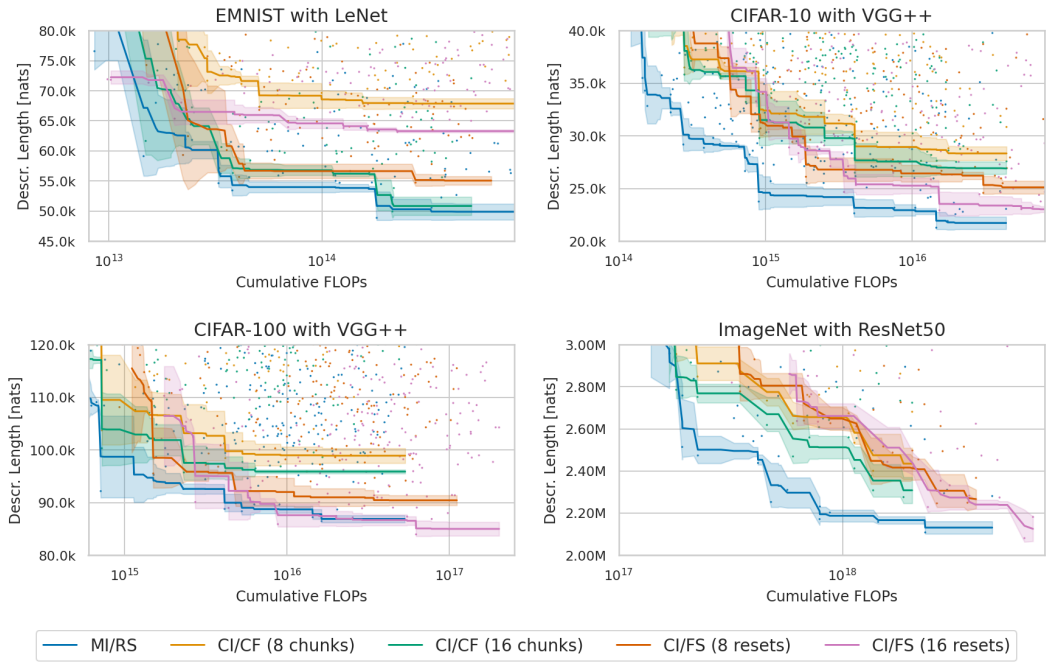


Figure 3: Shortest description lengths as function of compute resources consumed for different methods. For each method (in different colors) we run 250 (100 for ImageNet) independent experiments with randomly sampled hyperparameters. Each dot represents such an experiment and the solid lines show the Pareto-front of the best archived performance for different compute budgets. With the exception of CIFAR-100 in the large-FLOPs regime, the mini-batch incremental appoach MI/RS results in the shortest description lengths.

RMSProp and simple mini-batch SGD with momentum. We identified the following techniques that consistently improve the results and use them throughout this paper if not mentioned otherwise:

Exponentially moving average (EMA) Similar to Polyak averaging (Polyak & Juditsky, 1992), EMA maintains a moving average of the model parameters θ and is used for next-step evaluations as well as when tuning the softmax temperature β.

Label smoothing (Szegedy et al., 2015) A reliable regularizer, label smoothing often helps for the continual fine-tuning methods such as MI/RB, MI/RS and CI/CF.

Weight standardization For the VGG architecture, weight standardized convolutions lead to significant improvements - bigger than for conventional ERM-based training. For other architectures such as ResNets, the results are not conclusive.

3.1 Comparison of Online Learning Approaches.

To compare the effectiveness of the different approaches listed in Section 2, we run a broad random hyperparameter sweep for EMNIST with LeNet, CIFAR-10 and CIFAR-100 with VGG++ and ImageNet with ResNet50. The hyperparameter intervals depend on the data and are detailed in Appendix B. We sample learning rate, EMA step size, batch size, weight decay; but crucially also number of epochs (or, correspondingly, number of replay streams for MI/RS) and an overall scaling of the model width (number of channels). We thus cover a large range of computational budgets, spanning about 2 orders of magnitude. When the capacity of the replay-buffer matches the total length of the sequence, MI/RB and MI/RS are equivalent. In these cases we omit MI/RB and instead use MI/RS to represent online-learning with unlimited rehearsal. A study of replay buffer v.s. replay streams can be found in Section 3.1. We show the overall Pareto front for these experiments in Figure 3. Mini-batch incremental learning obtains the lowest description lengths for a wide range of computational budgets. Only for very large compute budgets on CIFAR-100 we observe that the block-wise approximation with training from scratch (CI/FS) has a better results.

Data	Model	MI/RS					ERM		
		Cumulative		Test			Test		
		Loss	Errors	Loss	Error(%)	FLOPs	Loss	Error(%)	FLOPs
MNIST	LeNet	4.4k	1362	0.03	1.0	1.4e13	0.03	0.8	1.5e13
MNIST	MnistNet	2.1k	643	0.02	0.5	4.5e14	0.02	0.5	3.2e14
CIFAR-10	VGG++	22.1k	6.6k	0.24	5.8	5.9e15	0.21	6.9	8.0e15
CIFAR-10	WRN-28-10	22.9k	7.5k	0.23	7.4	3.7e16	0.20	6.7	5.4e16
CIFAR-100	VGG++	93.7k	23.1k	1.14	32.0	4.2e15	1.14	28.1	2.6e15
CIFAR-100	WRN-28-10	87.0k	22.3k	1.10	30.3	2.5e16	1.08	28.3	4.3e16
ImageNet	VGG++	2.55M	549k	1.63	36.8	7.5e18			
ImageNet	ResNet-34	2.35M	507k	1.30	30.9	1.6e18	1.25	29.9	7.0e17
ImageNet	ResNet-50	2.21M	496k	1.29	29.1	8.5e17	1.15	28.7	1.9e18
ImageNet	ResNet-50†	1.93M	431k	1.06	26.1	4.0e18			
ImageNet	ResNet-101	1.97M	447k	1.20	28.2	3.1e18	1.10	26.6	4.6e18
ImageNet	DeiT-S	2.73M	588k	1.40	31.7	1.2e18	1.22	28.1	4.8e18
Prior Works:									
MNIST(Blier & Ollivier, 2018)	MnistNet	2.8k			0.5				
CIFAR-10(Blier & Ollivier, 2018)	≈ VGG++	31.4k			6.7				
ImageNet(Bornschein et al., 2020)	ResNet-50	3.32M			-				

Table 1: **Left half of the table**: Suite of results when applying MI/RS for description length estimation. Cummulative loss in nats; cumulative error counts the number of prediction errors after reaching the end of the sequence. The test set metrics report the performance of the model selected by lowest description length. **Right half**: Results when using the same architecture and hyperparameter-sweep for conventional ERM training. † Manually tuned hyper parameters.

MDL for Deep Neural Networks. We apply MI/RS to a wider range of architectures. For most experiments we again use random hyperparameter sweeps, this time however without scaling the size of the models and with an increased lower bound for the number of replay streams (see Appendix B). We present our results for the best run in each sweep and contrast them with previously reported description lengths from the literature in Table 1. We also show the test set performance of these models after they reached the end of training data. The rightmost columns show the test set performance when performing the same hyperparameter sweep for conventional ERM training on the full data set. Note that these runs do not use any kind of learning rate annealing, which is otherwise often employed to improve results.

Based on these experiments we plot a regret comparison for different architectures trained on ImageNet in Figure 4 (left). We observe that the additional depth and capacity of a ResNet101 is beneficial from the very beginning of the data-sequence. Figure 4 (right) shows the effect of scaling the number of channels of a VGG++ architecture on CIFAR-100 and it is evident that decreasing the model width has negative effects throughout the sequence and not just when the data gets larger.

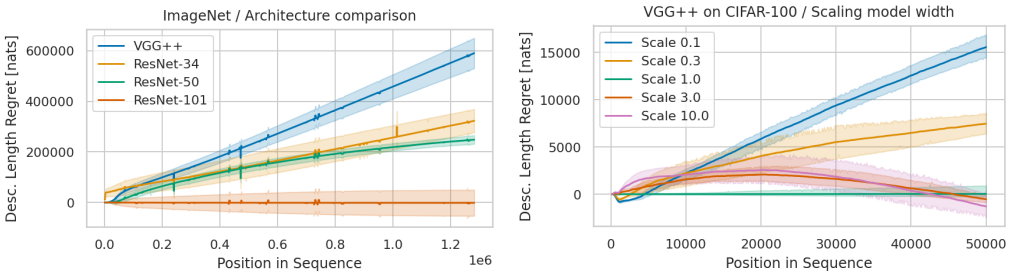


Figure 4: **Left**: Comparing architectures on ImageNet. **Right**: Scaling the size of VGG++.

Replay Buffer and Replay Streams are equivalent for short data sequences, they differ when the length of a sequence exceeds the capacity of the replay buffer. In this section, we study the effects of limiting the replay buffer size. Figure 5 shows that reducing the capacity has a significant negative effect on the results. In addition, the strategy of which samples to keep in the buffer plays an important role. Using reservoir sampling instead of a FIFO replacement policy leads to further severely degraded results. This is maybe not surprising because with reservoir-sampling insertion to the buffer is stochastic; a fraction of the examples are thus never replayed at all. Appendix C.5 contains ablations for the number of replay-streams on CIFAR-100 and ImageNet.

Ablations. We provide ablations for using forward-calibration, label smoothing and weight standardization in Table 2. It is evident that each of these techniques helps obtaining better description

lengths for the datasets and architectures under consideration. Regret plots for these ablations can be found in Appendix C.

Data	Model	None	+FC	+LS	+WS	+ALL
CIFAR-10	VGG++	32.4k	31.8k (-0.6k)	32.1k (-0.3k)	26.4k (-5.9k)	23.3K (-9.0K)
CIFAR-100	VGG++	103.4k	100.7k (-2.7k)	102.8k (-0.6k)	93.1k (-10.2k)	89.1k (-14.2)
ImageNet	ResNet50	2.33M	2.27M (-60.0k)	2.28M (-50k)		2.21M (-120k)

Table 2: Description lengths (in nats) with various techniques added to the baseline model: forward-calibration (FC), label smoothing (LS) and weight standardization (WS).

Plasticity and the Warm-starting problem We take a closer look at the large-FLOPs regime for CIFAR-100 where CI/FS obtained better results than MI/RS. Figure 6 (left) shows the regret comparison between the best runs for each training method. The split positions s_k are clearly visible as kinks in the graph for CI/FS and CI/CF. For CI/CF the plot shows that right after fine-tuning, the predictive performance is approximately equal to MI/RS. For CI/FS we see that the regret relative to MI/RS decreases: it has better predictive performance. This effect persists even after running extensive additional hyperparameter sweeps. We conjecture that this is a signature of the warm-starting problem described in (Ash & Adams, 2020). Figure 6 (right) shows the Pareto front for the final test set accuracy of the best models (selected by their description length) over FLOPs. We observe a large gap between fine-tuning based methods and CI/FS; which is consistent with this hypothesis. Note that the last split-point $s_K{=}T$ implies that these models are effectively trained like in a conventional ERM paradigm. Appendix C shows test set pareto fronts for a range of datasets. Close inspection of various regret plots sometimes reveals the same signatures, suggesting that the effect is ubiquitous, but often much less pronounced. It is plausible that this also contributes to the gap in final test set performance in Table 1.

4 Discussion & Related Work

Compression based approaches to inference and learning, such as Solomonoff induction (Solomonoff, 1964), Minimum Message Length (Wallace, 2005) and Minimum Description Length (Rissanen, 1984; Grunwald, 2004), have been extensively studied. The Bayesian and variational Bayesian approaches to estimating code lengths have been popular and, at least outside the field of deep learning, also been very successful. This line of work includes methods such as AIC and BIC, as well as more sophisticated approximations to the posterior distribution over model parameters for many kinds of models, including neural networks (Hinton & Van Camp, 1993; MacKay, 2003). The description length depends crucially on the quality of these posterior approximations. Unfortunately, estimating the parameter posterior for modern neural networks is notoriously hard and an area of active research (Izmailov et al., 2021). Blier & Ollivier (2018) demonstrated that much shorter code lengths can be achieved by using the prequential approach with standard neural network architectures and optimizers. The prequential perspective with the block-wise approximation technique have been successively used in the context of image-classification (Bornschein et al., 2020), natural language processing (Voita & Titov, 2020; Jin et al., 2021; Perez et al., 2021) and causal structure discovery (Bornschein et al., 2021).

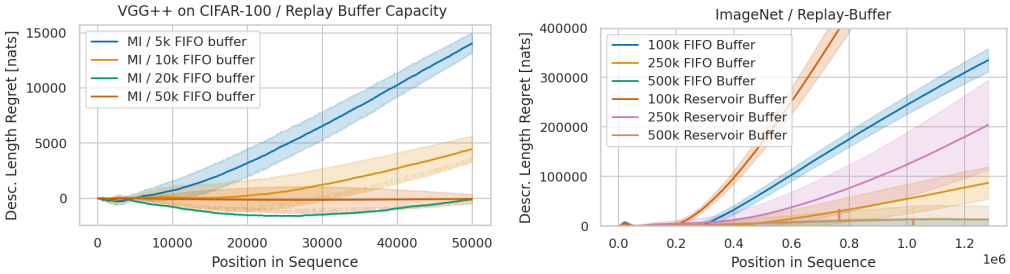


Figure 5: The effect of reducing the replay-buffer capacity. **Left**: Regret for different replay buffer sizes for CIFAR-100. **Right**: Comparing buffer capacity and FIFO vs. reservoir sampling based replacement strategies for ImageNet with ResNet50.

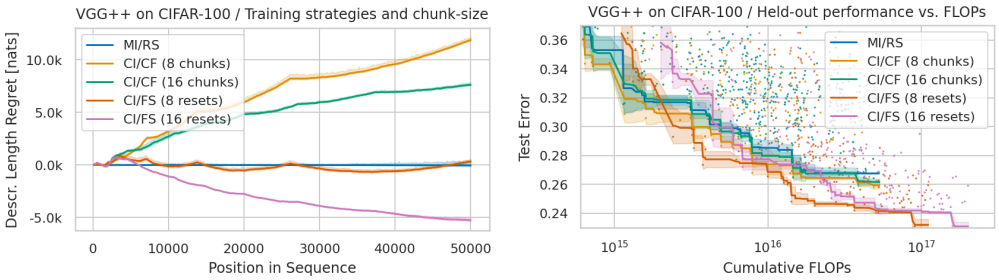


Figure 6: **Left**: Comparing the regret for CI/FS and CI/CF relative to MI/RS for CIFAR-100. **Right**: Pareto-front for the test set performance after reaching the end of the sequence

With our work, we aim to simplify and improve prequential code length estimation for neural networks with insights from the field of continuous learning (CL). For CL, fine-tuning with rehearsal is considered a strong baseline and large number of methods have been proposed to maintain its benefits while minimizing the size of the replay buffer (Mnih et al., 2013; Delange et al., 2021). We instead propose a method that makes it practical and relatively cheap to work with full rehearsal. Caccia et al. (2022) investigate very similar scenario to ours; largely phrased around how long a learner should wait and accumulate data when it is compute constraint; most experiments however consider the case without full rehearsal. He and Lin (He & Lin, 2020) discuss the close relationship between prequential coding and CL, however focus on compressive replay instead of accessing previous data. In general, the literature on CL methods and evaluation metrics is too broad to be adequately discussed here (Hadsell et al., 2020). However, we would not be surprised if some existing CL methods have the potential to vastly improve prequential MDL computation. In this sense, we see our work as a step of bringing CL and prequential MDL related works closer together.

Besides of taking inspiration from CL to improve description lengths, we believe that the MDL perspective can help guide future CL research: Much of the CL literature is rooted in an ERM-based perspective and thus considers distinct training, validation and test sets at different stages during learning. This has potentially played a role in the proliferation of many different evaluation metrics and training protocols used in CL research (Hadsell et al., 2020). This heterogeneity of the evaluation metrics makes it challenging to compare results. Forward-transfer is a commonly used evaluation metric and it is closely related to a prequential evaluation. However, it is often computed on separate held-out sets after training and does not take sample-efficiency into account as prequential MDL does. Recent work has embraced forward-validation in terms of average prediction-error on future training data as a evaluation metric, thus bringing it closer to a prequential MDL based evaluation but without discussing the connection (Cai et al., 2021; Lin et al., 2021). MDL has been conceived and analyzed as a theoretically well motivated objective when dealing with non-stationary data; without reference to the practical difficulties of actually performing parameter estimates. It is a challenging objective even for learners without constraints on compute and (replay-) memory.

5 Conclusions & Limitations

In this paper, we compare different approaches to computing prequential description lengths while considering their computational cost. We find that continual learning inspired mini-batch incremental training with full rehearsal is the preferred method over a wide range of computational budgets. We introduce two techniques that either improve results, or make it practical to apply full rehearsal even for very large data. As a result we report description lengths for a range of standard data sets that improve significantly over previously reported figures. Some of our results show that the warmstarting-effect can have a negative effect on the performance of continual fine-tuning. We also generally observe a gap in the final held-out performance after training models within the prequental framework vs. training models with the ERM-based approach. We are not aware of any technique to reliably mitigate the impact. It would be an important topic to tackle for future works. Furthermore, learning rate schedules such as cosine decay are popular and effective techniques for achieving better performance in ERM-based training. It is not obvious how to best translate them to the prequential learning scenario as the learner constantly receives new data.

REFERENCES

Jordan Ash and Ryan P Adams. On warm-starting neural network training. *Advances in Neural Information Processing Systems*, 33:3884–3894, 2020.

Léonard Blier and Yann Ollivier. The description length of deep learning models. February 2018.

Jörg Bornschein, Francesco Visin, and Simon Osindero. Small data, big decisions: Model selection in the small-data regime. In *International Conference on Machine Learning*, pp. 1035–1044. PMLR, 2020.

Jörg Bornschein, Silvia Chiappa, Alan Malek, and Rosemary Nan Ke. Prequential MDL for causal structure learning with neural networks. In *NeurIPS Workshop: Causal Inference & Machine Learning*, 2021.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

L Caccia, J Xu, M Ott, MA Ranzato, and L Denoyer. On anytime learning at macroscale. In *Conference on Lifelong Learning Agents (CoLLAs)*, 2022. URL https://arxiv.org/abs/2106.09563.

Zhipeng Cai, Ozan Sener, and Vladlen Koltun. Online continual learning with natural distribution shifts: An empirical study with visual data. August 2021.

Bilge Celik, Prabhant Singh, and Joaquin Vanschoren. Online automl: An adaptive automl framework for online learning. *arXiv preprint arXiv:2201.09750*, 2022.

A. Philip Dawid and Vladimir G. Vovk. Prequential probability: principles and properties. *Bernoulli*, 5(1):125–162, 1999.

Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Marco F Duarte and Yu Hen Hu. Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing*, 64(7):826–838, 2004.

Kyle Gorman and Steven Bedrick. We need to talk about standard splits. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 2786–2791, 2019.

Peter Grunwald. A tutorial introduction to the minimum description length principle, 2004. URL https://arxiv.org/abs/math/0406077.

Peter Grunwald and Paul Vitanyi. Shannon information and kolmogorov complexity. October 2004.

Peter D Grünwald. *The minimum description length principle*. MIT press, 2007.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pp. 1321–1330, 2017.

Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends Cogn. Sci.*, 24(12):1028–1040, December 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Xu He and Min Lin. Continual learning from the perspective of compression. In *4th Lifelong Machine Learning Workshop at ICML 2020*. arXiv, 2020. doi: 10.48550/ARXIV.2006.15078. URL https://arxiv.org/abs/2006.15078.

Geoffrey E. Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Conference on Computational Learning Theory*, pp. 5–13, 1993.

Marcus Hutter. Optimality of universal Bayesian prediction for general loss and alphabet. *Journal of Machine Learning Research*, 4:971–1000, 2003. ISSN 1532-4435. URL http://arxiv.org/abs/cs.LG/0311014.

Marcus Hutter. Human knowledge compression prize, 2006/2020.

Marcus Hutter. Algorithmic complexity. *Scholarpedia*, 3(1):2573, 2008. ISSN 1941-6016. doi: 10.4249/scholarpedia.2573.

Marcus Hutter. Testing independence of exchangeable random variables. Technical report, 2022.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Gordon Wilson. What are bayesian neural network posteriors really like? In *International Conference on Machine Learning*, pp. 4629–4640. PMLR, 2021.

Zhijing Jin, Julius von Kuegelgen, Jingwei Ni, Tejas Vaidhya, Ayush Kaushal, Mrinmaya Sachan, and Bernhard Schoelkopf. Causal direction of data collection matters: Implications of causal and anticausal learning for NLP. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021*. Association for Computational Linguistics, 2021. URL https://arxiv.org/abs/2110.03618.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4): 541–551, 12 1989. ISSN 0899-7667. doi: 10.1162/neco.1989.1.4.541. URL https://doi.org/10.1162/neco.1989.1.4.541.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer, Berlin, 3rd edition, 2008. ISBN 987-0-387-49820-1.

Zhiqiu Lin, Jia Shi, Deepak Pathak, and Deva Ramanan. The clear benchmark: Continual learning on real-world imagery. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

David J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. Rissanen data analysis: Examining dataset characteristics via description length. In *International Conference on Machine Learning*, pp. 8500–8513. PMLR, 2021.

Jan Poland and Marcus Hutter. Asymptotics of discrete MDL for online prediction. *IEEE Transactions on Information Theory*, 51(11):3780–3795, 2005.

Boris Polyak and Anatoli Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30:838–855, 07 1992. doi: 10.1137/0330046.

Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. Weight standardization. *arXiv preprint arXiv:1903.10520*, 2019.

Samuel Rathmanner and Marcus Hutter. A philosophical treatise of universal induction. *Entropy*, 13 (6):1076–1136, 2011. ISSN 1099-4300. doi: 10.3390/e13061076.

J Rissanen. Universal coding, information, prediction, and estimation. *IEEE Trans. Inf. Theory*, 30 (4):629–636, July 1984.

Jorma Rissanen. Stochastic complexity. *Journal of the Royal Statistical Society: Series B (Methodological)*, 49(3):223–239, 1987.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115 (3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27: 379–423, 623–656, 1948.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Anders Søgaard, Sebastian Ebert, Jasmijn Bastings, and Katja Filippova. We need to talk about random splits. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 1823–1832, 2021.

Ray J Solomonoff. A formal theory of inductive inference. part i. *Information and control*, 7(1):1–22, 1964.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. URL `http://arxiv.org/abs/1512.00567`.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, volume 139, pp. 10347–10357, July 2021.

V Vapnik. Principles of risk minimization for learning theory. In *Proceedings of the 4th International Conference on Neural Information Processing Systems*, NIPS'91, pp. 831–838, San Francisco, CA, USA, December 1991. Morgan Kaufmann Publishers Inc.

Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, mar 1985. ISSN 0098-3500. doi: 10.1145/3147.3165. URL `https://doi.org/10.1145/3147.3165`.

Elena Voita and Ivan Titov. Information-theoretic probing with minimum description length. *CoRR*, abs/2003.12298, 2020. URL `https://arxiv.org/abs/2003.12298`.

Matthew J Vowels, Necati Cihan Camgoz, and Richard Bowden. D'ya like dags? a survey on structure learning and causal discovery. *arXiv preprint arXiv:2103.02582*, 2021.

C S Wallace. *Statistical and Inductive Inference by Minimum Message Length*. Springer Science & Business Media, December 2005.

William F Whitney, Min Jae Song, David Brandfonbrener, Jaan Altosaar, and Kyunghyun Cho. Evaluating representations by the complexity of learning low-loss predictors. *arXiv preprint arXiv:2009.07368*, 2020.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith (eds.), *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 87.1–87.12. BMVA Press, September 2016. ISBN 1-901725-59-6. doi: 10.5244/C.30.87. URL `https://dx.doi.org/10.5244/C.30.87`.

SEQUENTIAL LEARNING OF NEURAL NETWORKS FOR PREQUENTIAL MDL SUPPLEMENTARY MATERIAL

A THE MINIMUM DESCRIPTION LENGTH PRINCIPLE

In this section we try to give a short overview of the mathematical formulation of the Minimum Description Length Principle (MDL). For a proper introduction we refer to (Rissanen, 1984) and (Grunwald & Vitanyi, 2004).

Like other closely related approaches such as Minimum Message Length (Wallace, 2005), MDL is based on the fundamental idea that learning and comprehension correspond to compression (Rathmanner & Hutter, 2011): Given data $\mathcal{D}=\{y_t\}_i^T$ and a hypothesis space $\mathcal{M} = \{M_1, M_2, \dots\}$, each referring to a probabilistic model family $p(\mathcal{D}|\theta, M_i)$ parameterized with the model parameters θ, the most plausible model $M \in \mathcal{M}$ is the one that yields the shortest description of the data, i.e. compresses it the most. To simplify the notation in this section we will use $\mathcal{D}=\{y_t\}_i^T$ to denote the predicted data only and omit the explicit conditioning on x_t. We use code and model interchangeably, since according to Kraft–McMillan's inequality, the negative logarithm of probability mass functions corresponds to code-length functions: $L(\mathcal{D} \,|\, \theta, M_i) = -\log p(\mathcal{D} \,|\, \theta, M_i)$.

A crude way to obtain a description length is to consider $L(\mathcal{D} \,|\, M)=L_M(\theta)+L_M(\mathcal{D} \,|\, \theta)$, where $L_M(\theta)$ is the cost of encoding the parameters θ and $L_M(\mathcal{D} \,|\, \theta) = -\log p(\mathcal{D} \,|\, \theta, M)$ is the cost of compressing the data with the parameterized model. While this approach might be intuitive, it is suboptimal and does not specify how to encode the parameters. To avoid the ambiguity that comes with choosing a code $L_M(\theta)$ for the parameters, the MDL literature instead suggests to use a *universal code* $\overline{p}(\mathcal{D} \,|\, M)$, which is defined to be the distribution that as closely as possible tracks the code-length of the maximum log-likelihood solution $\hat{\theta}^{\text{MLE}}(\mathcal{D}) = \arg\max_\theta p(\mathcal{D} \,|\, \theta, M)$ within M on all possible data $z_{1:T} \in \mathcal{Y}^T$:

$$\overline{p}(\cdot \,|\, M) := \arg\min_q \max_{z_{1:T}\in\mathcal{Y}^T} \Big(-\log q(z_{1:T}) - \big(-\log p(z_{1:T} \,|\, \hat{\theta}^{\text{MLE}}(z_{1:T}), M)\big)\Big) . \tag{2}$$

If it exists, the solution of q in Eq. (2) is the Normalized Maximum Likelihood (NML) $p_{\text{NML}}(y_{1:n}|M) = {}^{p(y_{1:n}|\hat{\theta}^{\text{MLE}}(y_{1:n}),\, M)}\big/{}_{\sum_{z_{1:T}\in\mathcal{Y}^n} p(z_{1:T}|\hat{\theta}^{\text{MLE}}(z_{1:T}),\, M)}$. Hence, the NML code-length is

$$L_{\text{NML}}(y_{1:T} \,|\, M) = \underbrace{-\log p(\mathcal{D} \,|\, \hat{\theta}^{\text{MLE}}(\mathcal{D}), M)}_{\text{Log-likelihood of data}} + \underbrace{\log \sum_{z_{1:T}\in\mathcal{Y}^n} p(z_{1:T}|\hat{\theta}^{\text{MLE}}(z_{1:T}), M)}_{\text{Model complexity}} \tag{3}$$

Eq. (3) has two terms: the log-likelihood term, assessing how well the model fits the data, and the model complexity term, formalising Occam's Razor. Intuitively, a model's complexity is large if it can fit well many possible data $z_{1:T}$. As a result, using the universal code for model selection leads to favoring models that have expressiveness to model the observed data $\mathcal{D}$, but do not waste expressiveness on dissimilar data.

Comparing to ERM, MDL doesn't need to have separate validation data, which means the entirety of the dataset can be used for training purpose. In addition, MDL applies for i.i.d data as well as non-stationarity data.

A.1 PRACTICAL INTERPRETATIONS OF MDL

Although NML is intuitive to understand, it is intractable and even undefined in many cases. There are two other versions of MDL which are also provably universal codes and, in addition, easier to compute in practice:

Bayesian and variational two-part codes. In the Bayesian approach, we choose a prior $p(\theta \,|\, M)$ over the parameters. The Bayesian interpretation of the MDL results in a code-length $L_{Bayes}(\mathcal{D}) = -\log \int_\theta p(\mathcal{D} \,|\, \theta)p(\theta)d\theta$. One can show that the Bayesian code is a universal code (Hutter, 2003) and it is equivalent to Bayesian model selection with marginal-likelihood. The variational approximation

to the Bayesian posterior can be considered a two part-code and makes it feasible to obtain an upper bound on the Bayesian description length. By sampling the parameters from an approximation $q(\theta)$ to the posterior distribution $p(\theta \mid \mathcal{D}, M)$, the code-length is equivalent to the negated ELBO objective in variational inference:

$$L_{var}(\mathcal{D} \mid M) = KL[q(\theta) \,||\, p(\theta \mid M)] - \mathop{\mathbb{E}}_{\theta \sim q} \log p(\mathcal{D} \mid \theta, M)$$

Prequential code. We can think of prequential coding as forward validation, where the loss incurs when predicting the datapoint y_{t+1} based on previously datapoints $y_{1:t}$. The code-length $L_{preq}(\mathcal{D} \mid M) = -\sum_{t=1}^{T} \log p(y_t | \hat{\theta}(y_{<t}), M)$, where $\hat{\theta}(y_{<t})$ is an estimator of the parameters after seeing $t-1$ datapoints. Typically, $\hat{\theta}(\cdot)$ is a regularized maximum likelihood estimator, but for many reasonable choices of $\hat{\theta}(\cdot)$, the resulting prequential (plug-in) code is a universal model despite not having an explicit model complexity term.

In practice, the prequential code can be shorter than that of the approximated Baysian code, especially for neural networks (Blier & Ollivier, 2018). Unlike Bayesian MDL which requires a Bayesian neural network, prequential MDL can work with any model architecture and optimizer. A central property that makes the prequential approach attractive for Neural Networks is that any prediction system that predicts the next-step log-loss given the previous data leads to a valid description length. There are no constraints on the type or implementation of the system as long as it obeys the sequential nature of the data and is deterministic (given the PRNG seeds). Even a naive system that predicts a uniform distribution over all outcomes constitutes a valid system. It leads to a description length of $T \log(\text{number-of-classes})$ and can be considered an upper bound for all sensible systems.

However, the choice and properties of the prequential estimator are important when considering the optimality of the code-length. Regularized maximum likelihood estimators for the next-step plug-in distribution generally lead to code-lengths that can be considered optimal for that model family because they form a universal-code (Rissanen, 1987). If it exists, it coincides with the normalized maximum-likelihood code-length up to neglectable offsets (Rissanen, 1987; Grünwald, 2007; Dawid & Vovk, 1999).

SGD as an estimator for Deep Learning. Deep Neural Networks (DNN) are typically trained with SGD which does not guarantee convergence to a maximum likelihood solution. Nonetheless, empirically, for ERM, well-tuned SGD finds solutions with good in-distribution generalization. Equivalently, although we cannot guarantee that DNN+SGD achieves a universal code when used as a next-step predictor in prequential MDL, it still gives a valid code length that can be used to tune hyperparameters. Just as different techniques developed for DNN with ERM evaluation, techniques can be developed and compared fairly with prequential MDL. For example, we can compare code length of the different CL techniques or replay strategies to determine their effectiveness.

A.2 Individual Sequences: Algorithmic Complexity and Shannon Information Theory

MDL is also closely related and sometimes phrased in terms of Algorithmic Complexity Theory (ACT) (Hutter, 2008; Li & Vitányi, 2008) instead of Shannon's Information Theory (Shannon, 1948). With ACT the fundamental concept of *information* is defined in terms of symbols and computation, instead of random variables and distributions. Quantities such as the entropy of a distribution, the mutual information between random variables and the cross entropy have analogous quantities in ACT: Kolmogorov complexity, algorithmic mutual information and description length respectively. In ACT however they are defined for individual sequences without the notion of an underlying probabilistic process. Grünwald and Vitanyi (Grunwald & Vitanyi, 2004) discuss this formally: The Kolmogorov complexity for example is a quantity associated with a concrete, individual sequence of symbols. If however we consider a stochastic process defining a distribution over such sequences, then the expected Kolmogorov complexity is equal to the Entropy of that process (up to an additive constant). Similarly, the expected description length of sequences with respect to a model corresponds to the cross-entropy between an empirical distribution and a model distribution. A similar relationship exists for mutual information and algorithmic mutual information.

Minimizing the description length for a given sequence of observations thus corresponds to the ACT analogon of minimizing the cross-entropy given only a single sample from the sequence-generating

process. Much like the cross-entropy is lower-bounded by the entropy of the empirical distribution, the description length of a sequence is lower-bounded by its Kolmogorov complexity.

A.3 PITFALLS OF USING TEST SPLITS: TRAIN- / TEST SET CONTAMINATION

Test set contamination is an increasingly grave unsolved problem that becomes more pressing as machine learning research moves from curated academic benchmark data to large user provided or web-scraped data. In the recent GPT-3 paper on large scale language modeling the authors dedicate multiple pages in the Appendix to discuss the issue (Brown et al., 2020). The paper details the method used to detect training-/test set duplicates and presents experiments and the conclusion that their particular model and evaluation approach is relatively insensitive to contamination. Indeed, some of the contamination was only discovered after the model had been trained, and the model could not be retrained with decontaminated data due to the prohibitive (computational) cost. This prompted some follow-up discussion on how validation and test sets should be created and decontaminated.

In (Søgaard et al., 2021) the authors show that depending on the exact splitting procedure significantly different results can be obtained. The discussion lays out arguments in favour of either using deliberately biased splits, or forward-validating on future data. This work refers to prior work (Gorman & Bedrick, 2019), which also described inconsistent results between different test sets, and between the random split, but at large came to a different conclusion.

To the best of our knowledge these issues have been primarily discussed in the NLP community. We mention them to point out that validation on held-out data can not blindly be regarded as the gold-standard for model evaluation when, at times, the procedure to create those splits is subject to intense discussion. Instead, careful consideration will be necessary in order to avoid misleading conclusions based on inappropriate validation and test sets.

Test set contamination is not only an increasingly important problem but also in a certain sense an unsolvable one, because it is ill-defined. At what threshold for example should a document that verbatimly cites a training-set document be removed from the test-set? When are two images scraped from the internet essentially the same?

On the other side: Test set contamination is no problem when using any of the MDL variants for model evaluation. For NML and Bayes this is true because they use model complexity and marginal predictions for regularization instead of held-out sets to avoid overfitting. But our limited ability to work with good parameter posterior estimates and the benign overfitting of overparametrized NNs make it largely non-applicable to deep learning. For prequential MDL it is true, because the model is evaluated and hence has to perform well already the first time a (potentially duplicate) data item appears.

For instance, assume the whole photo data base contains three or more (approximate) copies of each photo. If we randomly split off 10% as the test set, then the train set contains nearly all test set items. With heldout-validation a pure memorizer without any generalization capacity will perform nearly perfectly on the test set, but will fail miserably in practice on any newly taken photo. On the other hand, NML and marginal Bayes punish a memorizer due to its huge complexity, and prequential MDL due to the inability to predict (well) the first occurrence of each item. Test set evaluation *is* empirically sound for *i.i.d.* data, therefore the failure of this paradigm must be attributed to the non-iid nature of data with duplicates Hutter (2022).

For a more detailed discussion, see the FAQ in Hutter (2006/2020).

A.4 PROPERTIES OF PMDL BASED MODEL SELECTION

In the following we aim to build some intuition for why pMDL can intuitively be regarded an adequate metric in the i.i.d. scenario. We will consider two situations separately:

The large data regime, where the amount of training data significantly exceeds the amount necessary to train a "good" model: the pMDL score will be dominated by the large number of datapoints where the model's performance barely improves while learning from more data. For comparison purposes, the description length can therefore be approximated by # datapoints * generalization-cross-entopy (the area under the long tail). In this scenario pMDL based model selection will basically

choose the model with the best generalization performance, just like ERM+validation would. Only if two models have approximately the same performance in the large data regime, pMDL will use sample-efficiency as a tie-breaker.

In the small-data regime, where each training example can noticeably improve the model, the considerations are different: With ERM one has to choose the fraction of examples used for validation purposes to tune hyperparameters and architectures. Choosing a small validation set increases variance and thus the risk of choosing hyperparameters that work only well on the validation set (overfitting to the validation set). With a large validation set however, fewer examples are available for training. How should one choose the size of the validation set? Choosing hyperparameters with a large validation set and then re-training on the full data bears the risk that optimal hyperparameters change with increasing training set size. With pMDL on the other hand each example is used for validation first and successively available for training: no decision about the split size must be made and all data points are available for parameter estimation.

For specific proofs on the identifiability and convergence of MDL based model selection we refer to Grünwald (2007). Even assuming optimal continual learners we can not guarantee that MDL will always lead to better decisions than ERM (or vice-versa), especially when considering i.i.d. scenarios. However, the arguments laid out weaken the position that ERM+validation should be considered the gold standard without considering its limitations.

B Experimental Details

B.1 Forward Calibration

We train the neural network with standard cross-entropy loss and the categorical prediction-head $p = \mathrm{softmax}(h)$, where h is a vector of logits generated by the neural network. When using the model for evaluation on next-step data or on test set examples, we replace it with a calibrated prediction head $p = \mathrm{softmax}(\mathrm{softplus}(\beta)\, h)$, where β is a scalar temperature parameter (Guo et al., 2017). We use *forward-calibration* to optimize the calibration parameter β: each new batch of examples from the training sequence is used first for evaluation, then used to perform a single gradient step to optimize β, and finally used for a training to optimizer θ and placed into the replay-buffer. At the beginning of training we initialize β to $\beta_0 = \log(\exp(1) - 1) \approx 0.5416$, such that $\mathrm{softplus}(\beta_0) = 1$. We use the same optimizer as for the model parameters θ, however scale the learning rate by a factor of $\sqrt{K}$, where K is the number of training steps per evaluation step forward (i.e., number replay-streams plus one). Calibration is computationally almost free: the forward-pass for the calibration gradient step can be shared with the evaluation forward-pass, and backward-propagation for β ends shortly after the softmax layer.

B.2 MNIST, EMNIST and RGB-MNIST Hyperparameters

We run experiments with various model architectures: a) MLP with 1, 2 or 3 hidden layers, 512 units each, with dropout, and ReLU non-linearities, b) LeNet (LeCun et al., 1989) and c) the VGG inspired architecture used by (Blier & Ollivier, 2018), which we call MnistNet here.

We use the same hyperparameter and sampling intervals for all three model architectures. For the pareto front experiment we extend the number of replay-streams range to $10 \ldots 100$; otherwise we use the same intervals.

Parameter	Distribution	Values / Interval
Number of replay-streams	log-uniform	25 ... 100
Learning rate	log-uniform	1e-4 ... 3e-3
AdamW ϵ	log-uniform	1e-4 ... 1
EMA step size (Polyak averaging)	log-uniform	1e-3 ... 1e-1
Weight decay	log-uniform	1e-4 ... 1.
Batch size	fixed	32
Label smoothing	fixed	0.001

B.3 CIFAR-10 AND CIFAR-100 WITH VGG++, AND WIDERESNET HYPERPARAMETERS

We use the same hyperparmater sampling intervals for all model architectures. Following Zagoruyko & Komodakis (2016) we use only minimal augmentations during training: Images will be horizontally flipped with a probability of 0.5. For the pareto-front plots we extend the *number of epochs* (= number of replay-streams) range to $10 \ldots 100$ and

Parameter	Distribution	Values / Interval
Number of replay-streams	log-uniform	25 . . . 100
Learning rate	log-uniform	1e-4 . . . 3e-3
AdamW ϵ	log-uniform	1e-4 . . . 1
EMA step size (Polyak averaging)	log-uniform	1e-3 . . . 1e-1
Weight decay	log-uniform	1e-4 . . . 1e-1
Batch size	uniform	{32, 64, 128}
Label smoothing	fixed	0.01

B.4 IMAGENET WITH VGG++ AND RESNETS HYPERPARAMETERS

We use randaugment for data augmentation and the same hyperparameter intervals for all experiments. For Pareto front experiments we additionally scale the architecture size (number of channels throughout) from $1/4\times$ to $4\times$ and extend the number of replay-streams interval to $10 \ldots 100$.

Parameter	Distribution	Values / Interval
Number of replay-streams	log-uniform	25 . . . 100
Learning rate	log-uniform	1e-4 . . . 3e-3
AdamW ϵ	log-uniform	1e-4 . . . 1
EMA step size (Polyak averaging)	log-uniform	1e-3 . . . 1e-2
Weight decay	log-uniform	1e-4 . . . 1.
Batch size	fixed	512
Label smoothing	fixed	0.01

We report additional baseline metrics for the standard ResNet-50 model on ImageNet, obtained by running the same hyperparameter-sweep:

- Replay Streams: pMDL score 2.21M
- Replay Buffer (500k capacity): pMDL score 2.26M nats
- Chunkwise-from-scratch (16 split points, <1e19 FLOPs): pMDL score 2.52M nats,

Note that for chunkwise-from-scratch we use the techniques from Bornschein et al. (2020), who report 3.32M nats. Using better (wider) hyperparameter search and using more split-points we thus found a significantly better result for their method.

Using a 4x wider architecture:

- Replay Streams: pMDL score 2.18M nats
- Replay Buffer (500k capacity): pMDL score 2.22M nats
- Chunkwise-from-scratch (16 split points, 1e19 FLOPs): pMDL score 2.18M nats,

B.5 CLOC WITH RESNETS HYPERPARAMETERS

We use randaugment for data augmentation and the same hyperparameter intervals for all experiments.

Parameter	Distribution	Values / Interval
Number of replay-streams	log-uniform	10 . . . 25
Learning rate	log-uniform	1e-4 . . . 3e-3
EMA step size (Polyak averaging)	log-uniform	1e-3 . . . 1e-2
Batch size	fixed	128
Label smoothing	fixed	0.1

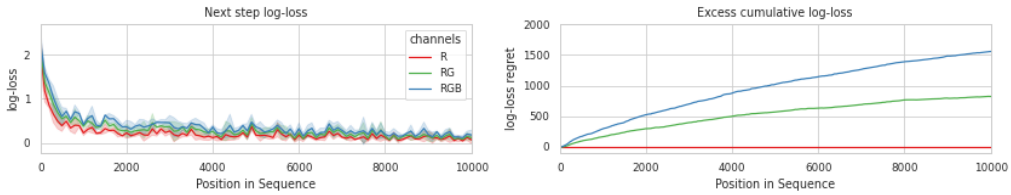
C ADDITIONAL EMPIRICAL RESULTS

C.1 RGB-MNIST FEATURE SELECTION

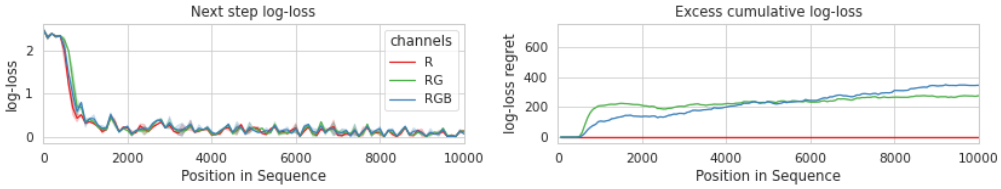
To illustrate that the model-selection property demonstrated in section 1 is a robust property of (prequential) MDL, we here present a suite of results on 3 different model architectures: 1) am MLP with 2 hidden layers, 512 units each; 2) the LeNet architecture from (LeCun et al., 1989); and 3) the much higher capacity and better tuned convnet from (Blier & Ollivier, 2018).

In all cases we run the same hyperprameter sweep detailed in Appendix B. We show the regret plot for the best performing model in each sweep relative to the model conditioned only on the R channel. We see pMDL reliably and with confidence determines that the model conditioned on R only is the most appropriate one for this data; and we see that the small-data performance at the beginning of the sequence is crucial as the relative generalization performance becomes more similar as the length of the sequence increases:

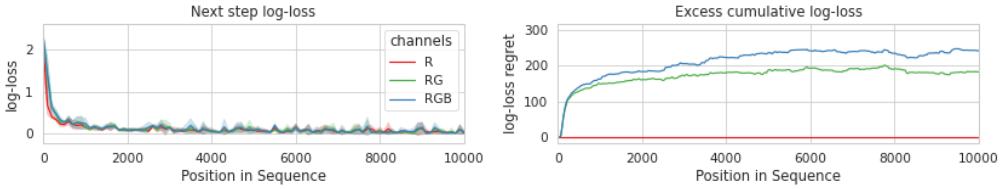
MLP Regret Plot



LeNet Regret Plot

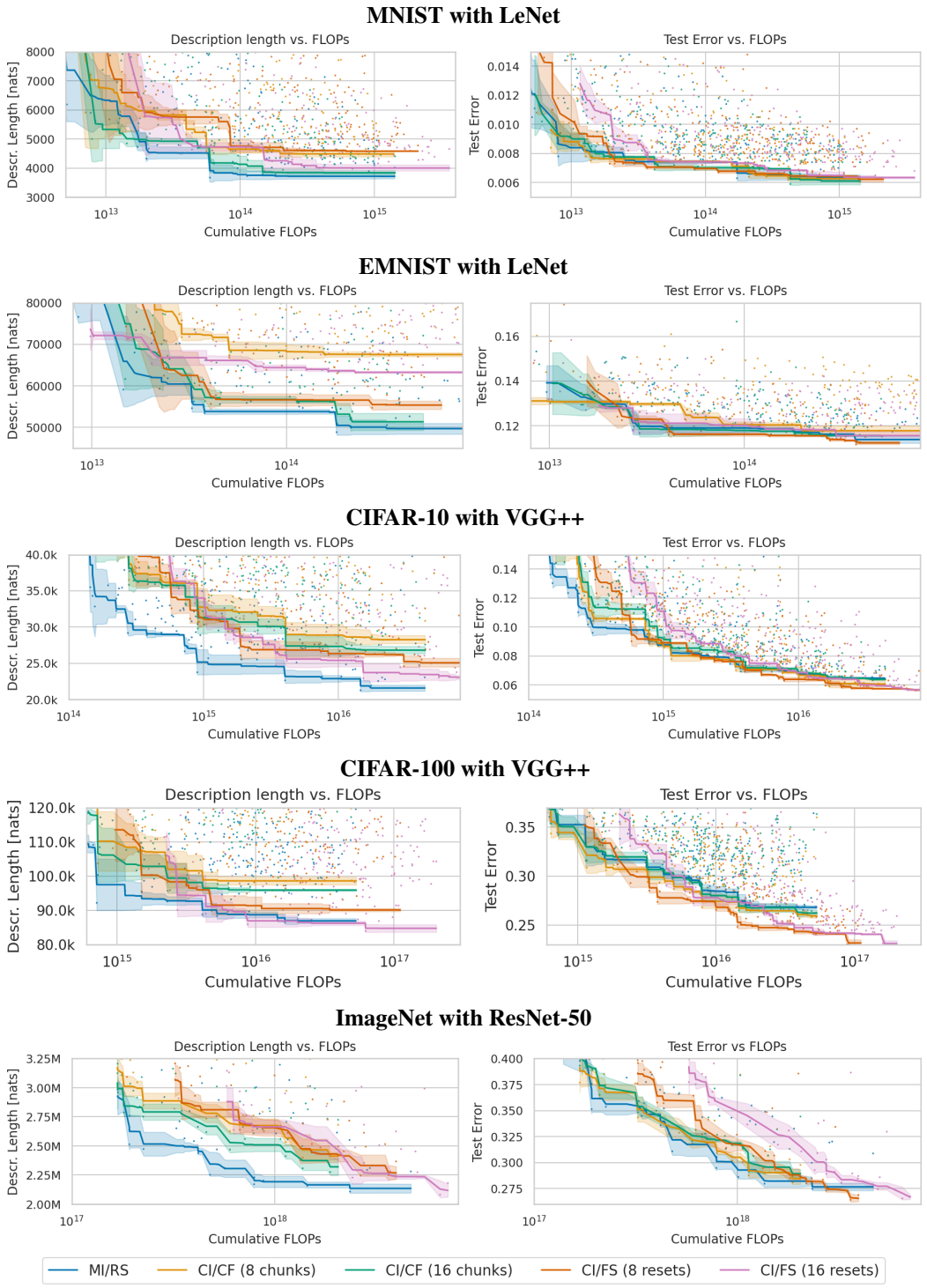


MnistNet Regret Plot

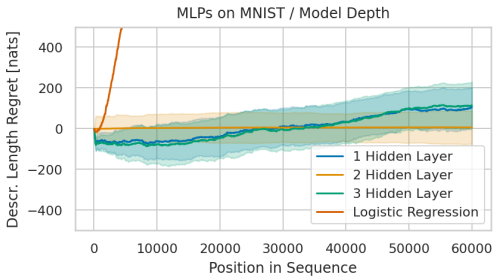


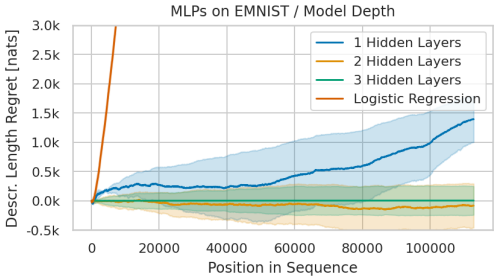
C.2 PARETO FRONTS

Left: Description lengths as function of compute resources consumed for a selection of datasets and models. **Right:** Test set performance after reaching the end of the training data. We generally observe a gap between models that train from scratch (CI/FS) and continual finetuning methods. The gap is very pronounced for CIFAR-100. We do not observe a systematic gap between MI/RS and CI/CF.

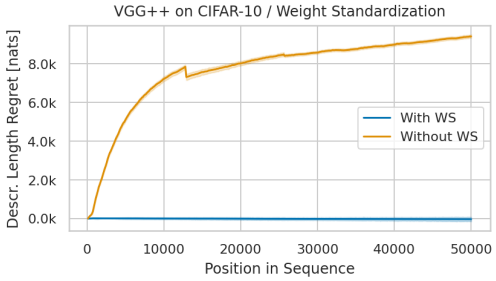


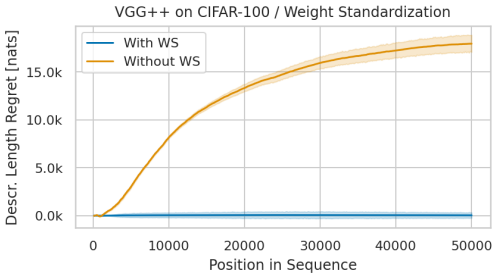
C.3 MODEL DEPTH ABLATION





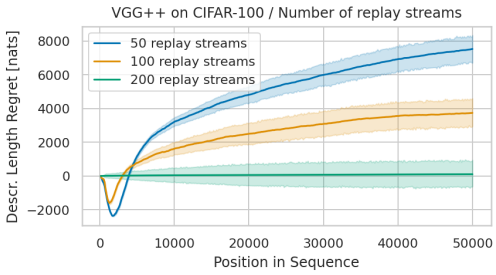
C.4 WEIGHT-STANDARDIZATION ABLATION

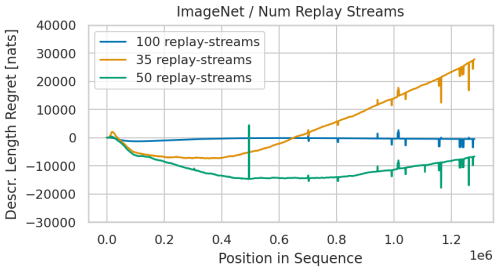




C.5 NUMBER OF REPLAY STREAMS

We run ablations for different number of replay-streams (K in Algorithm 1) on CIFAR-100 and ImageNet. The results on ImageNet show that performing too much replay (choosing K too large) can be harmful.

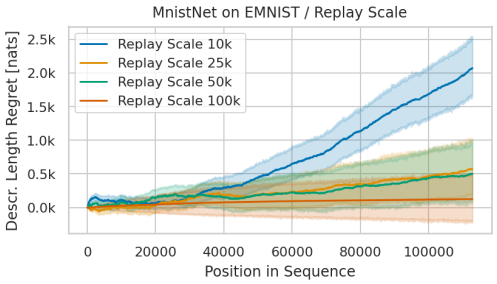


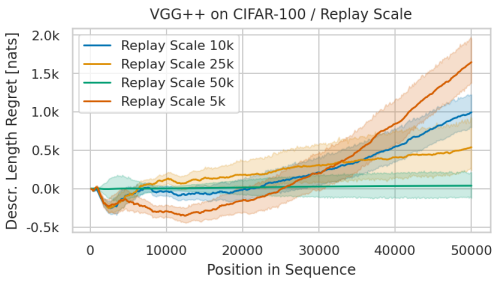


C.6 SCALE AND SHAPE OF THE REPLAY DISTRIBUTION

Besides of uniform replay for replay-stream training we experimented with alternative replay distributions. Exponentially decaying replay ($p_{\text{replay}}(a) = \lambda \exp(-\lambda a)$) is popular in continual-learning with reservoir sampled replay buffers, but lead to significant longer description lengths and suboptimal results in our experiments.

As an compromise between uniform replay and exponential decay we experimented with long-tailed Pareto distributions: $p_{\text{replay}}(a) = \frac{\alpha}{(a/\lambda)\hat{}(\alpha+1)}$. The scale parameter λ determines to what extend old examples are replayed and we use a fixed shape parameter $\alpha = \log_4 5 \approx 1.16$ for all out experiments:





Our experiments suggest that flat, almost uniform replay distributions lead to the shortest description lengths for the sequences from i.i.d. sources. On non-stationary sequences however, long-tailed replay that favours recent data, but maintains finite probability of replaying even the earliest examples works best.

C.7 Non-stationary sequences

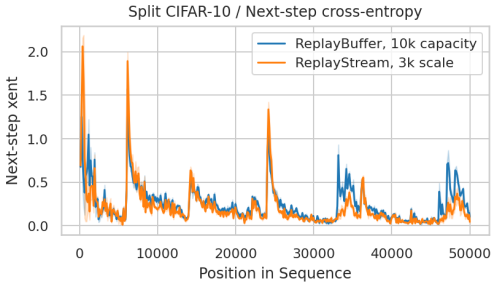
In this study we concentrate on data sequences from stationary (i.i.d) data sources to ensure we can compare our results with established ERM based training on heldout-data, and, because stationary data is arguably one of the most challenging scenarios for a sequential learner: there is no benefit in focusing on recenly observed data and instead a globally optimal predictive model must be learned.

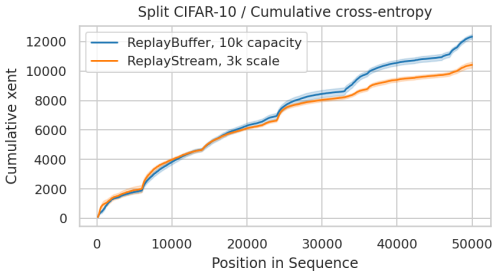
Nevertheless, one major benefit of MDL over ERM is that it is well defined for individual and non-stationary sequences. We therefore present a series of experiments on non-stationary data. We leave it to future work to analyze and to improve online learners on non-stationary data more thoroughly.

C.7.1 Split CIFAR-10

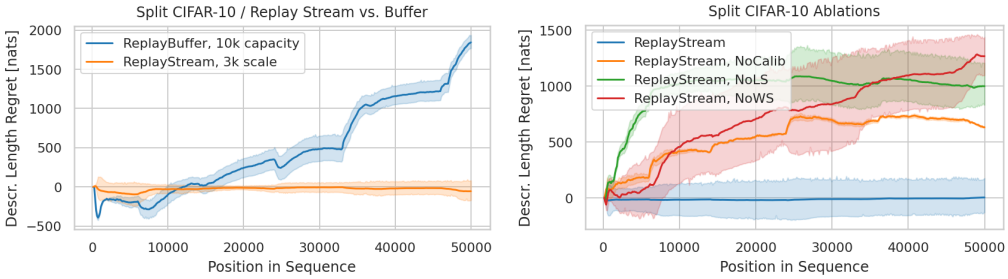
We derive a non-stationary sequence from the CIFAR-10 dataset by treating it as 5 disjunct binary classification tasks between the CIFAR classes 0 vs. 1 (Task-A), 2 vs. 3 (Task-B), . . . , 9 vs 10 (Task-E). We train the model sequentially on examples from each task, however instead of evaluating *forgetting* on held-out data after each new task, which is the standard protocol in continual learaning, we here insert new examples from previous tasks into the non-stationary sequence. A learner can therefore demonstrate good performance by rapidly re-adapting to a previous task from a few examples. The problem is task agnostic because we do not supply information about the current task-id to the learner. We concatenate unique examples from the tasks into a joint sequence: [0, 6000): Task-A, [6000, 13000): Task-B, [13000, 14000): Task-A, [14000, 22000): Task-C, [22000, 23000): Task-A, [23000, 24000): Task-B, [24000, 33000): Task-D, [33000, 34000): Task-A, [34000, 35000): Task-B, [35000, 36000): Task-C [36000, 46000): Task-E, [46000, 47000): Task-A, [47000, 48000): Task-B, [48000, 49000): Task-C, [49000, 50000): Task-D

We run the broad random hyperparameter sweeps for the VGG++ model, however add the scale of the pareto-distributed replay distribution and the capacity of the replay buffer as an additional random hyperparameter between $[1000 \cdots 50000)$. We show the next-step and cumulative next-step performance for the best performing replay-stream and replay buffer learners:



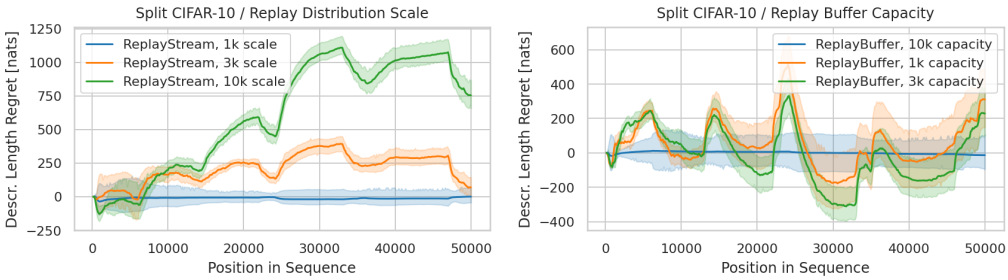


The regret plot for the same experiments show the difference more clearly. We also show the regret for the best replay-stream model with various techniques disabled:



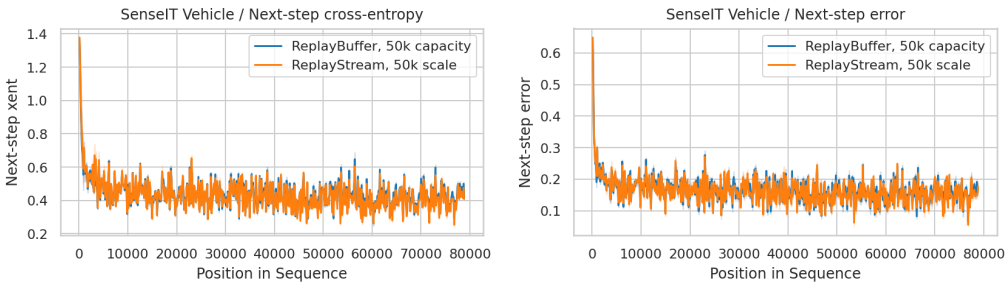
Like in the previous plots, the shaded areas indicate the 99% confidence intervals obtained from bootstrap sampling the experiments in the hyperparameter sweep. We see that multiple replay-stream learners have significantly better performance than the best replay-buffer learner. Given that this dataset is small and fits completely into RAM, we can close the gap between replay-streams and replay-buffers by configuring a replay-buffer with 50k capacity and using recency-biased sampling; which is the only difference between these learners.

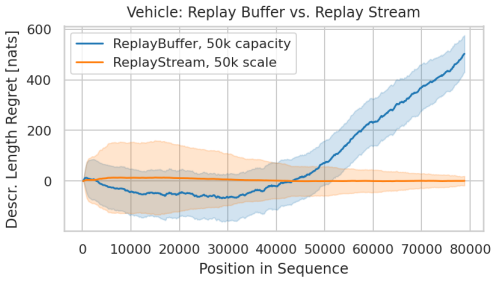
Finally, we run an ablation study with 3 different scales for pareto-distributed replay; and the regret plot for 3 replay-buffer learners with 3 different capacities of the uniform sampled replay-buffer. The results were obtained by running the same hyperparameter sweeps as before, however with the replay distribution or replay-capacity fixed to 1k, 3k or 10k respectively. We observe that the width of the replay distribution and the capacity of the replay-buffer trade-off between faster adopting to completely new tasks, and having better performance on recurring tasks:

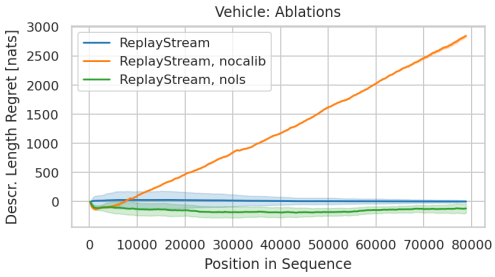


C.7.2 Vehicle Classification Data

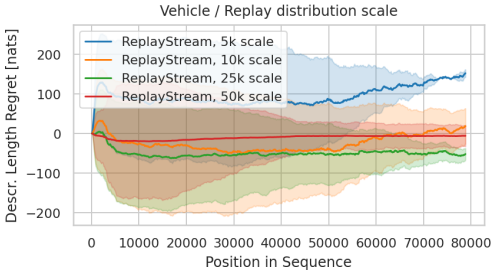
We run experiments on the vehicle classification data sequence introduced by Duarte & Hu (2004). The sequence consists of about 80k observations with 100 features each; the classification task is to predict the vehicle category. We use a 2 hidden-layer MLP as the model architecture with the same hyperparamter-sweep from the VGG++ experiments. We show the next-step performance, the corresponding regret plots for replay-stream vs replay-buffer training, and regret plots with label smoothing or forward calibration disabled.

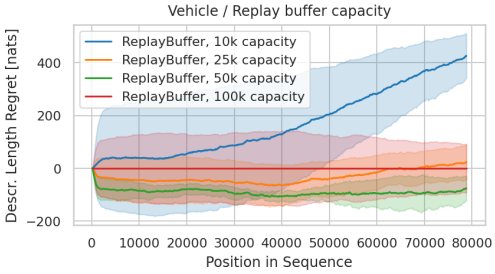






Over the whole sequence we obtain an average forward validation error of 15.5% (84.5% accuracy). This is competitive with the more complex and ensemble-based methods described by Celik et al. (2022) (85% accuracy), and an significant improvement over the baseline methods described therein ($\approx$ 82% accuracy). We highlight that we here essentially just train a neural network with a fixed order of training examples, with recency biased rehearsal, label smoothing and forward-calibration.



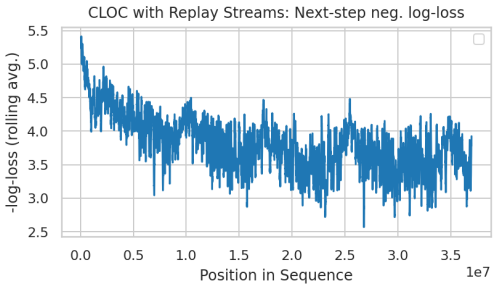


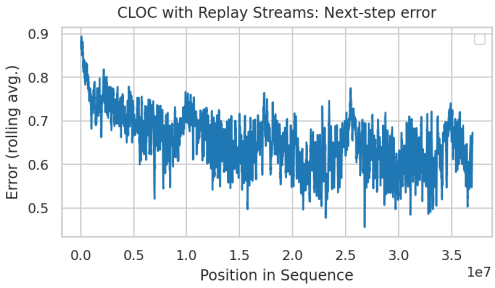
C.7.3 CONTINUAL GEO-LOCALIZATION (CLOC)

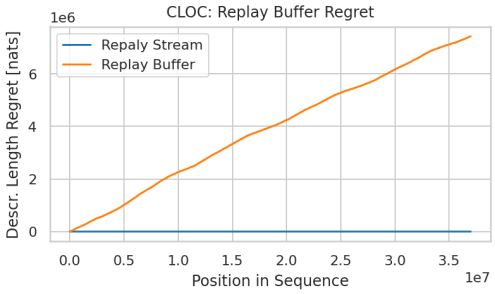
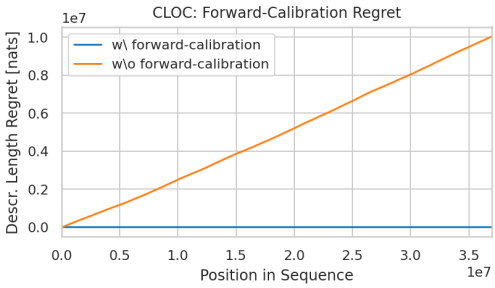
We present results on the CLOC data introduced by Cai et al. (2021). CLOC is a large scale, chronologically sorted image classification sequence consisting of $\approx 39M$ images with their geolocation as a categorical label. For $\approx 5\%$ of the images we either received download errors, or the downloaded content could not be decoded as image data. In total we were able to obtain 37,093,769 labeled images. Figure 2 in (Cai et al., 2021) shows that the data is strongly non-stationary; and that traditional ERM training, which treats the data as i.i.d., results in a held-out error rate between 80 and 90%.

Cai et al. (2021) reserve the first 2M (5%) datapoints for pre-training the model statically before entering the online-phase, split-off 1% of the data throughout for validation purposes and only evaluate the images from the first new album in each mini-batch instead of all images. Their online-learning task and metric are thus in spirit similar to the prequential MDL problem we are tackling here, in detail however sufficiently different that the quantitative results are not directly comparable. Following (Cai et al., 2021) we use a standard ResNet50 architecture and run experiments with replay buffers and replay streams. The hyperparameter sampling space is detailed in Appendix B.5. Note that we use a batch size of 128 because small batch-sizes allow the model to more quickly adapt to changes in the data distribution; and a relatively low number of 10 to 25 replay-steps / replay-streams to limit the total computational cost due to the size of the data.

We observe that heavily recency biased replay leads to significantly improved results. For replay streams we experimented with uniform replay, replay with exponentially decaying priority and with replay governed by a heavy tailed Pareto distribution (see C.6). From these the heavy tailed Pareto replay leads to the shortest description lengths. Even more than in the i.i.d. case, forward-calibration plays an crucial role to obtain good results.





Next-step performance We plot the 5k rolling average next-step performance of the best performing model from our random hyperparameter sweep (K=22 Pareto distributed replay streams with a scale of 180). The model obtains a description length of 1.33G nats and a next-step error-rate of typically $60 \pm 10\%$.

Ablations **Left:** We show the description length regret for the best replay-buffer training run relative to the best replay-stream training run. Both experiments consumed about 5×10^{18} FLOPs. **Right:** Regret for replay-streams without forward calibration ralative to replay streams with forward calibration: