# EVALUATING REPRESENTATIONS WITH READOUT MODEL SWITCHING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Although much of the success of Deep Learning builds on learning good representations, a rigorous method to evaluate their quality is lacking. In this paper, we treat the evaluation of representations as a model selection problem and propose to use the Minimum Description Length (MDL) principle to devise an evaluation metric. Contrary to the established practice of limiting the capacity of the readout model, we design a hybrid discrete and continuous-valued model space for the readout models and employ a switching strategy to combine their predictions. The MDL score takes the model complexity, as well as the data efficiency into account. As a result, the most appropriate model for the specific task and representation will be chosen, making it a unified measure for comparison. The proposed metric can be efficiently computed with an online method and we present results for pre-trained vision encoders of various architectures (ResNet and ViT) and objective functions (supervised and self-supervised) on a range of downstream tasks. Finally, we discuss important properties revealed by these evaluations such as model scaling, preferred readout model, and data efficiency.

## 1 INTRODUCTION

Data representation is crucial to the performance of machine learning algorithms. Much of the success of Deep Neural Networks (DNN's) can be attributed to its capability of gradually building up more and more abstract representations. In supervised learning, although the network is trained to predict a specific aspect of the input, the intermediate representations are often proven to be useful for many other downstream tasks. In unsupervised and self-supervised learning, the network is trained on a surrogate task, such as reconstruction and contrastive prediction, that is supposed to capture generic prior of the data. In the recent years, there has been significant improvements in unsupervised representation learning with state-of-the-art models achieving comparable performance against its supervised counterpart.

Despite the importance of data representation, the evaluation method for representations is rarely discussed. The most prevalent practice for evaluation is to use a readout model taking input from a pre-trained encoder and train it on the downstream task. The readout model often has a shallow architecture, e.g. linear layer, to limit its capacity, so that the task performance reflects the representation quality. The major problem with this approach is that the readout model cannot be adapted to the nature of the representations. As a result, the representations are assessed with multiple metrics, each using a different readout model, making the comparison extremely difficult.

In this paper, we treat evaluating the representations as a model selection problem and propose to use *Minimum Description Length* (MDL) as the main evaluation metric. MDL is a well-studied compression-based approach for inductive inference that provides a generic solution to the model selection problem. MDL performs a similar role as held-out validation does for *Emperical Risk Minimization* (ERM), but has the advantage of being able to deal with single sequence and non-stationary data. It's closely related to Bayesian model selection and includes a form of Occam's Razor where the metric takes into account the model complexity. The complexity term could be explicitly represented as the codelength of the model in the case of a 2-part code, as a KL-term when using a variational code, or implicitly when using prequential or Bayesian codes. By including the model complexity in the evaluation metric, we automatically resolve the need of limiting the readout model complexity and are able to compare MDL

score freely between different readout architectures. Intuitively, if the induced representation is nonlinear and requires higher capacity model for readout, the MDL score reflects this need by having a larger complexity term. Note that this also applies in the case of fine-tuning, where the pre-trained model is allowed to adapt for the downstream tasks. Therefore, MDL provides a unified framework for evaluating representations regardless the evaluation protocol employed.

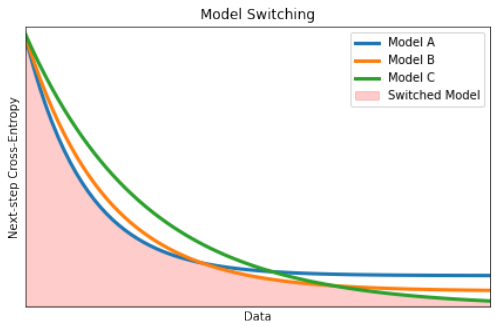



Figure 1: Illustration of switching between models of different complexity: Depending on the number of training examples either $A$, $B$, or $C$ has the best generalization performance. An optimally switched model will have the best performance at each point and thus the lowest prequential description length (= area under the curve).

It has been conjured that useful representations should make the variability in data more predictable and allow human-like data efficient learning (Hénaff et al., 2019). Using MDL as the evaluation metric brings out the data efficiency perspective - especially evident in the form of prequential MDL. Prequential MDL (Dawid & Vovk, 1999; Poland & Hutter, 2005) turns computing the description length $L(\mathcal{D}|\phi) = -\log p(\mathcal{D}|\phi)$ into a sequential prediction problem: $\log p(\mathcal{D}|\phi) = \sum_t \log p(y_t|\phi_{\leq t}, y_{<t})$, where $\phi$ is the encoder, $\phi_t := \phi(x_t)$ and $(x_{<t}, y_{<t})$ is the data seen before timestep $t$. In order to achieve a short description length, it's beneficial if the representation is capable of fast learning, i.e. given a good representation a few examples is enough to achieve the optimal performance. As a by-product of our readout model switching based on MDL, we can visualize the predictive performance along the data sequence and inspect how efficient is the downstream learning.

Our contribution are as follows, we:

1. propose readout model switching for evaluating representations.
2. prove the regret bound of readout model switching and discuss assumptions made.
3. use an online learning framework for efficient computation and evaluate the performance of several popular visual representations methods on the set of downstream tasks.
4. investigate and inspect the preferred readout models, data efficiency and model scaling.

## 2 Background

**Minimum Description Length** is based on the fundamental idea that learning and comprehension correspond to compression (Rathmanner & Hutter, 2011). Given data $\mathcal{D}=\{y_t\}_1^N \in \mathcal{Y}^N$ and a hypothesis space $\mathcal{M} = \{M_1, M_2, \dots\}$, where each hypothesis $M$ corresponds to a parametric probabilistic model $p(\mathcal{D}|\theta, M)$, MDL aims to identify the model that can compresses the data $\mathcal{D}$ best. Considering the close relationship between lossless coding and probability distributions, this can be achieved by associating a codelength function $L(\mathcal{D}|M)=-\log p(\mathcal{D}|M)$ with each hypothesis. A vast body of literature shows that models with a shorter description length have a better chance of generalizing to future data (Grünwald, 2007a; Rathmanner & Hutter, 2011).

A crude way to obtain description lengths 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 and $L_M(\mathcal{D}|\theta)=-\log p(\mathcal{D}|\theta, M)$ is the cost of compressing the data with the parameterized model. This approach is called a *two-part code* and while it is intuitive, it is sub-optimal and ambiguous because it does not specify how to encode the parameters. Instead, the MDL literature considers three distinct but closely related approaches: 1) The marginal Bayesian likelihood $L_{\text{Bayes}}(\mathcal{D}|M)= -\log \int_\theta p(\mathcal{D}|\theta, M)p(\theta)d\theta$ and its variational upper bound $L_{\text{ELBO}}(\mathcal{D}|M)=KL(q(\theta)|p(\theta)) - \mathbb{E}_{\theta\sim q}\log p(\mathcal{D}|\theta, M)$. Both depend on the chosen prior $p(\theta)$ and require access to the posterior distribution over parameters given data; or an approximation thereof. 2) Normalized maximum likelihood (NML) $L_{\text{NML}}(\mathcal{D}|M)= -\log p(\mathcal{D}|\hat{\theta}_{\text{ML}}(\mathcal{D}), M)/\int_z p(\mathcal{D}|\hat{\theta}_{\text{ML}}(z), M)dz$ only relies on the maximum likelihood estimator $\hat{\theta}_{\text{ML}}(\cdot)$, but normalizes over all potentially observable data $z \in \mathcal{Y}^N$, which is often intractable and even

undefined for many model families of interest. The log-numerator is also called the complexity of $M$ and it measures how well the model could fit all possible data. NML illustrates that a model $M$ achieves a short description length only when it fits the given data well (high $\log p(\mathcal{D}|\hat{\theta}_{\text{ML}}(\mathcal{D}), M)$), but would not fit well many different data (low complexity). 3) The prequential approach $L_{\text{plugin}}(\mathcal{D}|M) = -\sum_{t=1}^{N} \log p(y_t|\hat{\theta}(\mathcal{D}_{<}t), M)$ decomposes the description length over datapoints and relies on the choice of a suitable plug-in parameter estimator $\hat{\theta}(\mathcal{D}_{<t})$. It emphasizes that in order to achieve a short codelength a model $M$ must not only be a good predictor given large data, but also be a sample efficient predictor. Model complexity and sample efficiency can thus be considered two sides of the same coin.

Approaches to computing description lengths for some data $\mathcal{D}$ under a model family $p(\mathcal{D}|\theta, M)$ are called *universal codes* when they result in description lengths that stay close to the (generally unachievable) maximum likelihood codelength for that family: $L_{\text{universal}}(\mathcal{D}|M) \leq -\log p(\mathcal{D}|\hat{\theta}_{\text{ML}}(\mathcal{D}), M) + \mathcal{O}(\log N)$. $L_{\text{NML}}(\mathcal{D}|M)$ is by definition universal; $L_{\text{Bayes}}(\mathcal{D}|M)$ and $L_{\text{plugin}}(\mathcal{D}|M)$ are universal for many reasonable choices of the prior $p(\theta)$ and the estimator $\hat{\theta}(\mathcal{D}_{<t})$ respectively (Grünwald, 2007b).

**Methods for Representation Evaluation.** Linear probing, using a linear layer for readout, is the most prevalent evaluation protocol for representations, the performance of which is reported and compared in almost all of the recent representation learning works (van den Oord et al., 2018; Chen et al., 2020; Grill et al., 2020; Li et al., 2021; Caron et al., 2021; He et al., 2021). Apart from linear probing, clustering algorithms such as Hebbien linear classifier (Bengio, 2012) and K-Nearest Neighbor (Caron et al., 2021) are also popular choices. However, concerns have been raised for model comparison using these simple probes. Resnick et al. (2019) shows that the performance of linear probing can be misleading, since models perform weakly with linear probe can perform better under alternative protocols. He et al. (2021) also argue that the use of linear probe limits the development of methods that induce non-linear representations. Hewitt & Liang (2019) question the practice of using downstream accuracy as the evaluation metric and propose to design probe based on its selectivity which puts the task accuracy in context with the probe's capacity to memorize. Yogatama et al. (2019) and Voita & Titov (2020) propose to use MDL, either variational or prequential code, as the evaluation metric which takes into account the complexity in the final measurement. Hénaff et al. (2019) bring to light the benefit of pre-training for downstream data efficiency. Whitney et al. (2020) propose a modified version of the MDL which introduces a baseline corresponding to solving the downstream task.

## 3 Evaluating Representations with Readout Model Switching

We formally define the model selection problem for representation evaluation under the MDL principle: the objective is to compare the encoder $\phi(.)$ that minimizes the codelength $L(\mathcal{D}|\phi) = -\log p(\mathcal{D}|\phi)$, where $\mathcal{D} = \{(x_t, y_t)\}_1^N$ is the inputs $x_t$ and associated prediction targets $y_t$. The prequential interpretation of the MDL decomposes the codelength into the cumulative prediction loss $L(\mathcal{D}|\phi) = -\sum_{t=1}^{N} \log p(y_t|\phi_{\leq t}, y_{<t})$, where $\phi_t := \phi(x_t)$ is the encoded feature of input $x_t$. In order to perform the prediction task from the features, previous works have to fix the probe to be a linear layer or a MLP. In our study, instead of picking a fix model, we design a hybrid of continuous and discrete-valued $(k, \theta_k)$ model space of the readout models, where $k \in [1, \ldots, K]$ is the model class and $\theta_k$ is the continuous-valued parameters corresponding to the model class $k$. Each readout model can be seen as an expert forecasting system that learns to predict the data independently in an online fashion. At each datapoint $x_t$, the prediction of $\hat{y}_t$ is a combination of the experts' predictions. The final MDL score is the cumulative predictive performance of the board of experts.

By constructing the probe using multiple readout models and switching between them, the final MDL score is marginalized over the choice of the readout architectures. Since MDL automatically takes into account the complexity of the model, i.e. any extra complexity used is going to cost in the MDL score, our framework allows the representation to select the most suitable readout mechanism at any data size while still maintain a comparable final score for representations selection. In addition, after computing the score, we can inspect the posterior over the readout models at any given data size, as well as data efficiency along the trajectory, providing valuable insight into the characteristics of the representation.

**Read-out Model Switching** Given $K$ readout models, we define random variables $\xi_t \in [1, \dots, K], \forall t \in [1, \dots, N]$ the readout model to use at timestep $t$. The Expert Sequence (ES) prior (Koolen & de Rooij, 2008) uses a hidden Markov model (HMM) to define the joint distribution of the data $y_{1:N}$ and random variable $\xi_{1:N}$ as $p(y_{1:N}, \xi_{1:N}|\phi) = \prod_{t=1}^{N} p_{\xi_t}(y_t|\phi_{\leq t}, y_{<t})p(\xi_t|\xi_{t-1})$, where $p(\xi_1|\xi_0)$=$p(\xi_1)$ is the initial distribution and $p_k(\cdot)$ is the prediction of the $k$th readout model. At each timestep $t$, we use previously observed data to estimate $\hat{\theta}_k(\phi_{<t}, y_{<t})$ for the parameters of the readout models. Therefore, the $k$th readout model prediction $p_k(y_t|\phi_{\leq t}, y_{<t})$ is the plugin distribution $p(y_t|\phi_t, \hat{\theta}_k(\phi_{<t}, y_{<t}))$. The final readout model switching codelength function is in the form of:

$$L_{Switch}(y_{1:N}|\phi) = -\log p(y_{1:N}|\phi) = -\log \sum_{\xi_{1:N}} \prod_{t=1}^{N} p(y_t|\phi_t, \hat{\theta}_{\xi_t}(\phi_{<t}, y_{<t}))p(\xi_t|\xi_{t-1}) \quad (1)$$

The readout model switching defined in Equation 1 represents a family of codes that combines prequential and Bayesian MDL. From the perspective of the HMM, $p(\xi_t|\xi_{t-1})$ is the transition matrix and, in our context, it is equivalent to a switching strategy between the readout models. Every switching strategy corresponds to a specific code in the family. For example, if $p(\xi_t|\xi_{t-1}) = 1$, i.e when we believe a single readout model is responsible for modeling the whole data sequence, then Equation 1 simplifies to a Bayesian mixture code $L_{BM}(y_{1:N}|\phi) = -\log \sum_k p(k) \prod_{t=1}^{N} p(y_t|\phi_t, \hat{\theta}_k(\phi_{<t}, y_{<t}))$.

For the rest of the paper, we are going to use the fixed share (FS) (Herbster & Warmuth, 1998; 2004) as the switching strategy. The prior defined in fixed share can be interpreted as follows: at timestep $t$, with a probability 1 - $\alpha_t$, the expert is the same as the previous timestep; with $\alpha_t$ probability, it switches to another model according to the probability $w(k)$. We are going to use uniform distribution $w(k) = 1/K$ through out the paper. At the switching point, we can either choose a different model than the current one or choose among all available models. We are going to use the latter one. The switching strategy can be simplified and expressed as $p(\xi_t|\xi_{t-1}) = \begin{cases} 1 - \frac{K-1}{K}\alpha_t & \text{if } \xi_t = \xi_{t-1} \\ \frac{1}{K}\alpha_t & \text{if } \xi_t \neq \xi_{t-1} \end{cases}$, where $\alpha_t = (m-1)/t \in [0, 1]$ is decreasing over the timestep $t$ and $m$ is a hyperparameter. Note that there are other switching strategies that could be used, e.g. elementwise mixture, fixed share with constant switching rate, switch distribution and run-length model (Koolen & de Rooij, 2008). We provide a comparison of these strategies in Appendix A.5. In general, we find that Bayesian mixture and elementwise mixture perform significantly worse than fixed share and other more sophisticated switching strategies. The poor codelength of the Bayesian mixture is due to the catch-up phenomena (van Erven et al., 2012): it assumes that a single model is responsible for modeling the whole data sequence, while evidently the best model changes with data set size.

**Regret Bounds**

**Theorem 1.** *Let a sequence $x^N = \{x_i\}_{i=1}^N$ be i.i.d. from distribution $P$. Let all readout models $M_1, \dots M_K$ be in the exponential family. $\hat{\theta}_k(x^t)$ is the regularized Maximum Likelihood (ML) estimator of $M_k$ after first t examples $x^t = \{x_i\}_{i=1}^t$. The expected regret of readout model switching w.r.t the best sequence of readout model $\xi^*_{1:N}$ and their best fit parameters $\hat{\theta}_{\xi^*_{1:N}}(x^N)$ is*

$$\begin{aligned} R(N) &= \mathbb{E}_P \left[ \sum_{t=1}^{N} \log M_{\xi^*_t}(x_t; \hat{\theta}_{\xi^*_t}(x^N)) - \log \sum_{\xi_{1:N}} p(\xi_{1:N}) \prod_{t=1}^{N} M_{\xi_t}(x_t|\hat{\theta}_{\xi_t}(x^{t-1})) \right] \\ &\leq -\mathbb{E}_P \left[\log p(\xi^*_{1:N})\right] + \frac{1}{2} \max_k \left\{ Tr \left( Cov^{-1}_{M_{\mu^*_k}}[T_k(x)] Cov_P \left[T_k(x)\right] \right) \right\} \log N \end{aligned} \quad (2)$$

*where $\mu^*_k = \mathbb{E}_P[T_k(x)]$, $T_k(.)$ is the sufficient statistic and $\mu_k$ is the mean value parameter of $M_k$.*

The proof of Theorem 1 is in Appendix A.1. The expected regret consists of two terms: the regret of switching strategy and the regret from plugin distribution. The latter is bounded by $C \log N$ with $C$ is a constant determined by the model family and the true distribution. For fixed share with decreasing switching rate, $-\mathbb{E}_P \left[\log p(\xi^*_{1:N})\right] \leq m \log K + (m-1) \log N/(m-1)$, where $m-1$ is

**Algorithm 1** MDL with Read-out Model Switching

**Require:** data $\mathcal{D} = (x_t, y_t)_{t=1}^{N}$; $K$ readout models; initial model probability $p(\xi_1)$ and switching strategy $p(\xi_t|\xi_{t-1})$; subroutine that updates the parameters given a dataset *UpdateParameters*

Initialize: model parameters $\theta_1, \ldots, \theta_K$; $s = \log p(\xi_1)$; an empty list $Q$ to store posterior $p(\xi_t|\mathcal{D}_{<t})$
**for** $t = 1$ **to** $N$ **do**
  Compute $\log p(\xi_t, y^{t-1}|x^{t-1})$: $s \leftarrow \log \sum_{\xi_{t-1}} \exp\left(\log p(\xi_t|\xi_{t-1}) + s\right)$
  Compute posterior $p(\xi_t|\mathcal{D}_{<}t)$ and store in $Q$:
  $\log p(\xi_t|\mathcal{D}_{<}t) = \log p(\xi_t, y^{t-1}|x^{t-1}) - \log \sum_{\xi_t} p(\xi_t, y^{t-1}|x^{t-1})$
  Compute next-step loss of $K$ readout models: $\mathcal{L}_t^k := -\log p_k(y_t|x_t) = -\log p(y_t|x_t; \theta_k)$
  Combine $K$ models to update $\log p(\xi_t, y^t|x^t)$: $s \leftarrow -\mathcal{L}_t^{\xi_t} + s$
  **for** $k = 1$ **to** $K$ **do**
    Update parameters: $\theta_k \leftarrow$ *UpdateParameters*$(\theta_k, \mathcal{D}_{\leq t})$
  **end for**
**end for**
Compute total codelength $\mathcal{L}_{switch} \leftarrow -\log \sum_{\xi_N} \exp(s)$
**return** $\mathcal{L}_{switch}$ and $Q$

the number of switch points (Veness et al., 2012; Koolen & de Rooij, 2013). Therefore, the expected regret is in the order of $\log N$, making the corresponding code a universal code. Many other choices of switching strategy have the same property, such as Bayesian mixture and run-length code (Koolen & de Rooij, 2013).

We note however, that in the context of Deep Neural Networks (DNN), the assumptions of Theorem 1 are not fulfilled: DNNs are not in the exponential family and the solutions found by Stochastic Gradient Descent (SGD) training are not ML solutions. Although the exact nature of the final codelength heavily relies on the training algorithm and much of the understandings of SGD are left for future works, as long as the estimator converges asymptotically to the optimal solution, we can use the code for model selection. In fact, there exist estimators that outperform the ML estimator such that the resulting prequential code better approximates the Normalized Maximum Likelihood (NML) code which is the optimal universal code (Grünwald & Kotlowski, 2010).

**Computing MDL with Online Learning** Algorithm 1 describes the online implementation for computing the description length and for performing inference over the preferred readout models $p(\xi_t|\mathcal{D}_{<t})$. For each example in the sequence we first obtain a marginal prediction by weighting the predictions from the readout models according to our current belief over the mixture. After receiving the label $y_t$, we compute the cross-entropy loss for the marginal prediction and add it to the cumulative loss. Then, the belief over the mixture is updated and the new example is added to the training data for readout model parameter estimation. The latter is done by online SGD with replay.

For the posterior belief over the model switches $p(\xi_t|\mathcal{D}_{<t})$ the above procedure amounts to the well known, efficient, and exact forward pass HMM inference algorithm. For the neural network parameters $\theta_k$ on the other hand we use SGD as the plugin estimator for prequential MDL. Prior work has shown that SGD is an effective and reliable plug-in estimator for modern neural networks (Blier & Ollivier, 2018). Note that most prior works on computing prequential MDL scores with neural networks use a block-wise approximation; i.e. they partition the data into a small number of segments and evaluate each segment independently by training on all the previous segments to convergence (Voita & Titov, 2020; Bornschein et al., 2020; Whitney et al., 2020). Since we need a prediction at each step $t$ to update the mixture weights, we here instead use the online mini-batch incremental training with replay described in Bornschein et al. (2022). This approach updates the parameters continually with online mini-batch gradient descent training on the new and previously seen examples. Instead of using a replay buffer to store examples in-memory, we use replay streams to iterate through the examples in their original order from permanent storage. By tuning the number of streams and the resetting probability, we can adjust the amount of replay training and change from uniform sampling to a sampling strategy that is biased towards more recent samples. However, in this work, we use uniform random replay samples for all our experiments. As suggested

in Bornschein et al. (2022), we use Exponential Moving Average (EMA) and label smoothing to improve performance of the online training, but not weight standardization as we don't want to alter the architectures.

An alternative to the fully online implementation is a two-stage implementation (see Appendix A.3): First, the prequential online parameter estimation is computed for each of the readout models independently while the per-step predictions and log-losses are recorded for later analysis. In a second stage, the predictions and losses can be used to compute the mixtures, switching posterior $p(\xi_t|\mathcal{D}_{<t})$ and overall description lengths. To facilitate experimenting with different switching strategies we use this 2-stage implementation for all our experiments.

# 4 EXPERIMENTS

To demonstrate the effectiveness of our representation evaluation framework, we compare popular vision encoders pre-trained on ImageNet with different objectives. We primarily use the VTAB benchmark (Zhai et al., 2019) for downstream tasks, since transfer performance is the main interest for representation learning. We also use ImageNet classification as downstream task when showing specific properties, such as data efficiency and preferred readout model, and for comparing in-domain and out-of-domain transfer. For some VTAB datasets, notably KittiDistance, SmallNORBAzimuth, PatchCamelyon, we discover that there is a distribution shift between train and test subsets (see details Appendix A.2). Therefore, in order to make sure that data are truly i.i.d, we reshuffle the train and test data and use the reshuffled datasets for all our experiments.

The VTAB benchmark consists of 19 datasets that can be grouped into three categories: natural, specialized and structured. When evaluating a representation, we obtain a MDL score for each dataset. To summarize the performance into a single measure, we follow the procedure recommended by Demšar (2006) for comparison of algorithms over multiple datasets. First, we rank the representations by the MDL score on each dataset. Then, we compute the average ranking of the representation. For any pair of representations, the difference of the average ranking can be served as a statistics for a hypothesis test (Nemenyi test) to conclude on whether the difference in performance is significant. We report the average rankings and the threshold value of the difference with the confidence interval $\gamma = 0.1$.

We use the online learning framework as described in Section 3 and compute the fixed share code with switching rate decrease over time $\alpha_t = (m-1)/t$ and $m$ is a hyperparameter . The raw MDL scores per dataset are reported in Appendix A.6. For ImageNet, we use readout model switching between 4 model architectures: Linear, single layer MLP, and up to 3-layer MLPs. For VTAB, we include up to 7-layer MLP's as readout architectures. We furthermore instantiate each of these architectures with all hyperparameter settings from the hyperparameter grid outlined in Appendix A.4. The model switching is therefore performed between #architectures $\times$ #hyperparameters many independent models. We report results using a switching rate $m = 2$ for ImageNet and with a switching rate $m = 11$ for VTAB. The switching rates were chosen to optimize the overall description length.

## 4.1 PRE-TRAINING OBJECTIVES

We compare downstream performance of different pre-training objectives on VTAB benchmark. To make the comparison, we choose two backbone architectures, ResNet-50 v1 (He et al., 2015) and ViT/B16 (Dosovitskiy et al., 2020), and only change the pre-training objectives. For ResNet-50, we pre-train on supervised, SimCLR and BYOL losses; for ViT/B16, we pre-train on supervised (Touvron et al., 2020; He et al., 2021), DINO and MAE losses. SimCLR (Chen et al., 2020) is a self-supervised method based on contrastive learning. BYOL (Grill et al., 2020) and DINO (Caron et al., 2021) both use a teacher network. BYOL is a negative-free method for self-supervised representation, while DINO uses ViT architecture and learns through self-distillation. MAE (He et al., 2021) uses masked reconstruction task to learn representation.

Figure 2 reports the average ranking of the pre-training objectives. Given a confidence interval $\gamma = 0.1$, the performance of the representations are significantly different if the difference is above

$q_{0.1}\sqrt{\frac{k(k+1)}{6N}} = 1.894$, where $q_{0.1} = 3.12$ is the critical value for the two-tailed Nemenyi test, $k = 6$ is the number of algorithms to compare and $N = 19$ is the number of datasets.

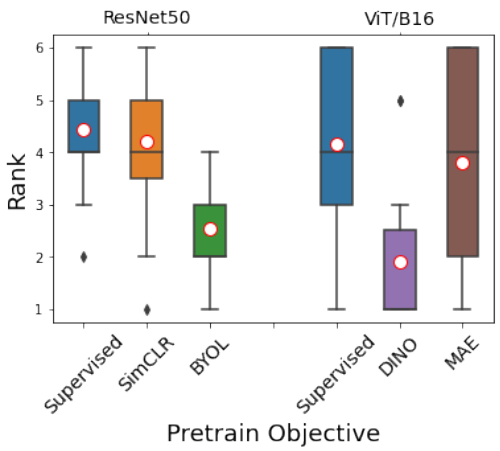
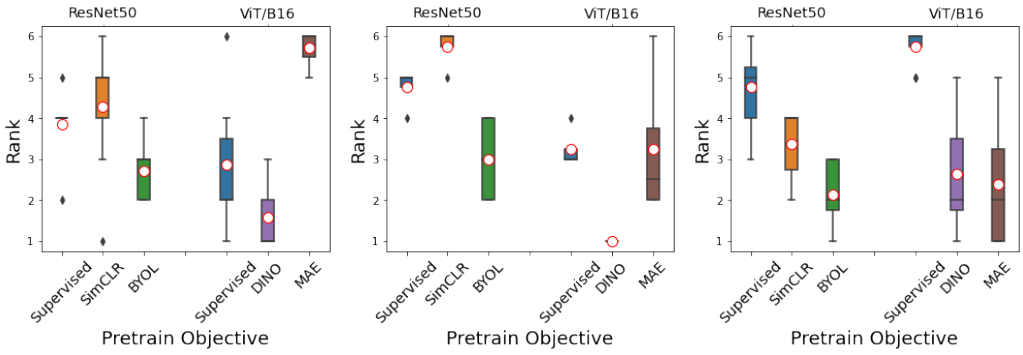



Figure 2: Ranking for representations pre-trained with different objectives (lower the better). The box plot is drawn from 25% to 75% quantile with a horizontal line to denote the median and red dot to denote the mean. The whisker denotes 1.5 IQR values. The threshold of the ranking difference to be significant for hypothesis testing is 1.894.

In Figure 2, we observe that for ResNet50 backbone, BYOL features outperforms both supervised and SimCLR features, while SimCLR and supervised features have no significant difference. For ViT/B16 backbone, DINO features outperform both supervised and MAE, while supervised and MAE have no significant difference. Therefore, it appears that with the same encoder capacity, distillation-based objective has an advantage in terms of transfer. When comparing BYOL with DINO, although DINO seems slightly outperform, the ranking difference is not significant for a clear conclusion.

In VTAB, datasets in the same group are similar to each other, but there is a significant distribution shift between groups. And natrual datasets are closer to the ImageNet which is used during pre-training. To investigate if the transfer performance differs with input distribution shift, we report the average rankings by dataset group in Figure 3. Although the number of datasets in each group is limited and, therefore, the result from hypothesis test is not conclusive, we can still get some idea of the transfer performance from the ranking difference. We can see that self-supervised representations have a better performance than the supervised counterpart on structured datasets, despite the latter perform decently on natural datasets. MAE generalizes well with dataset shift, but the performance on natural datasets suffers compared to other methods.




(a) Natural Datasets. (b) Special Datasets. (c) Structured Datasets.

Figure 3: Ranking on natrual, special and structured VTAB datasets. The thresholds for ranking difference to be significant are 3.12, 2.92 and 4.13 respectively.

### 4.2 Model Scaling

Another interesting aspect about architecture is the model scaling: does larger model result in better performance? We again use ResNet and ViT backbones pre-trained on ImageNet and evaluate on VTAB, but with increasing model sizes. For ResNet, we select ResNet-50, ResNet-101 and ResNet-152, and pre-train with either supervised or BYOL objective. For ViT, we use ViT/S16, ViT/B16 and ViT/L16, and pre-train with either supervised or MAE objective. We rank the performance of

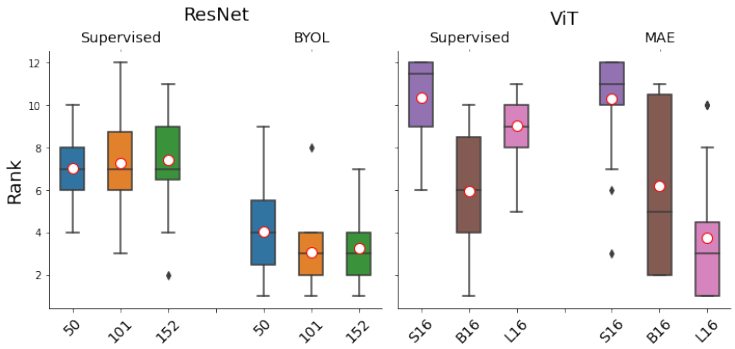



Figure 4: Rankings of ResNet and ViT models of different sizes (lower the better). The thresholds for ranking difference to be significant is 3.65.

the 12 representations and report the average rankings in Figure 4. With confidence interval of 0.1, we obtain a threshold of 3.65 for the difference in average ranking to make a statistical claim. We can see that scaling up the ResNet architecture, in particular under supervised pre-training, doesn't result in any improvement in the transfer performance. There is a weak positive scaling effect for BYOL when scale to ResNet-101, but it is not statistically significant. Although ViT/S16 has approximately the same parameter counts as ResNet-50, it is predictive performance is much worse, suggesting that ViT's capacity is used less efficiently than ResNet. Scaling up ViT from S16 to B16, therefore, has a significant performance jump. However, further scaling up for supervised loss is not useful. Interestingly, the reconstruction-based MAE objective scales well to larger architectures and has not been saturated at ViT/L16.

### 4.3 Read-out Models

| | | Natural | | | | | | | Special | | | | Structured | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Architecture | Objective | Caltech101 | Cifar100 | DTD | Flowers102 | Pets | SVHN | Sun397 | EuroSAT | PatchCamelyon | Resisc45 | Retinopathy | ClevrCount | ClevrDistance | DMLab | DSpritesLocation | DSpritesOrientation | KittiDistance | SmallNORBAzimuth | SmallNORBElevation |
| ResNet50 | SUP | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 1 | 3 | 4 | 3 | 3 | 4 | 2 | 4 | 3 | 2 |
| ResNet101 | SUP | 1 | 1 | 3 | 1 | 0 | 1 | 1 | 1 | 2 | 1 | 2 | 3 | 3 | 2 | 2 | 2 | 4 | 3 | 2 |
| ResNet152 | SUP | 1 | 1 | 1 | 2 | 0 | 1 | 1 | 1 | 2 | 1 | 2 | 3 | 4 | 2 | 1 | 2 | 5 | 3 | 7 |
| ResNet50 | SimCLR | 1 | 1 | 1 | 1 | 3 | 2 | 1 | 1 | 4 | 1 | 2 | 7 | 3 | 5 | 3 | 3 | 4 | 7 | 7 |
| ResNet50 | BYOL | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 6 | 1 | 3 | 4 | 5 | 5 | 4 | 3 | 7 | 6 | 7 |
| ResNet101 | BYOL | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 3 | 1 | 3 | 4 | 6 | 5 | 3 | 3 | 6 | 6 | 7 |
| ResNet152 | BYOL | 1 | 1 | 1 | 1 | 0 | 2 | 1 | 1 | 3 | 1 | 3 | 5 | 4 | 4 | 2 | 3 | 4 | 6 | 3 |
| ViT/S16 | SUP | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 4 | 1 | 3 | 3 | 2 | 2 | 4 | 5 | 5 | 7 | 4 |
| ViT/B16 | SUP | 0 | 1 | 2 | 0 | 1 | 1 | 1 | 1 | 3 | 1 | 3 | 7 | 6 | 2 | 3 | 3 | 4 | 3 | 7 |
| ViT/L16 | SUP | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 1 | 3 | 2 | 2 | 7 | 7 | 3 | 1 | 3 | 5 | 4 | 7 |
| ViT/B16 | DINO | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 3 | 1 | 4 | 5 | 5 | 6 | 3 | 3 | 3 | 3 | 7 |
| ViT/S16 | MAE | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 3 | 1 | 1 | 2 | 2 | 6 | 7 | 4 | 7 | 6 | 5 |
| ViT/B16 | MAE | 0 | 1 | 1 | 1 | 2 | 1 | 0 | 1 | 2 | 1 | 1 | 3 | 4 | 4 | 4 | 6 | 5 | 7 | 4 |
| ViT/L16 | MAE | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 4 | 4 | 4 | 4 | 5 | 6 | 5 | 6 |

Table 1: Read-out model that most often has the highest probability $\arg\max_k \frac{1}{T}\sum_{t=1}^{T} p(\xi_t = k|\mathcal{D}_{<t})$ for different representations on 19 VTAB datasets. 0 is linear readout; 1 to 7 are MLP with 1 to 7 hidden layers.

One strength of our evaluation framework is that it can utilize readout models of different capacity and dynamically choose the most suitable one based on downstream task and training data size. After evaluation, we can compute the posterior over the readout models to inspect which readout model the representation prefers. We observe that the model switching is more frequent for out-of-domain datasets than in-domain datasets. In Table 1, we report the readout model that has most often the highest probability $\arg\max_k \frac{1}{T}\sum_{t=1}^{T} p(\xi_t = k|\mathcal{D}_{<t})$. The result suggests that the most

important factor for preferring deeper model is the dataset type - structured datasets need deeper models. The influence of the pre-training objective doesn't seem to be significant.

### 4.4 DATA EFFICIENCY

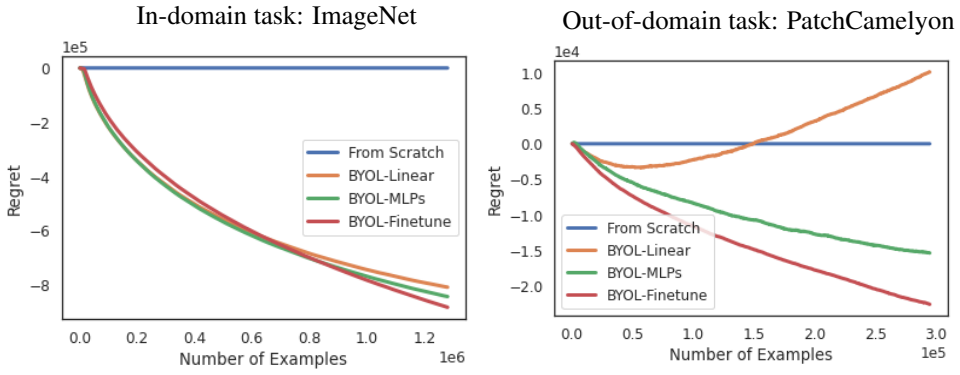



Figure 5: Data efficientcy: We plot the cumulative next-step log-loss for a range of readout relative to a ResNet-50 model as a function of the position in the sequence (in nats, lower is better).

Data efficiency is an integral part of MDL. With the prequential perspective on MDL, we can visualize whether an algorithm is more data efficient by inspecting the predictive loss as a function of the data size. As a baseline we train a ResNet-50 model from scratch within our evaluation framework, and record the cumulative next-step losses as a reference baseline (comparator). Then, we evaluate BYOL pretrained ResNet-50 representations with 3 different readout models: 1. linear probing; 2. deep MLP readout models; 3. linear readout but with fine-tuning the whole encoder. Because MDL already takes into account the model complexity, we can directly compare their MDL scores. We plot the "regret" (in this case it is more of an advantage) of different fine-tuning strategies relative to the training-from-scratch model as the baseline. Figure 5 illustrates the "regret" along the data size for ImageNet, an in-domain dataset, and PatchCamelyon, an out-of-domain dataset for the BYOL representations. Learning the ImageNet classification with a pre-training representation is much more efficient than training from scratch. Shallow architectures, which are based on a frozen encoder, are sufficient. Only when the dataset is large enough (800k in case of ImageNet), we can finally see the benefit from fine-tuning the encoder. The picture is different for out-of-domain datasets. In general, pre-training still has benefit if deeper, more flexible readout models are used. The amount of capacity needed depends on the nature of the downstream dataset. In Figure 5, on PatchCamelyon, training from scratch surpasses linear probing after seeing only 150k examples. We show the regret plots for all 19 VTAB tasks in Appendix A.8.

## 5 CONCLUSIONS & FUTURE WORKS

We preset readout model switching, an approach derived from the MDL principle, to evaluate and compare representations on given downstream tasks. Our approach supports model switching to choose the most appropriate readout model for each representation, downstream task, and the data size. Being a MDL derived approach, our metric takes model complexity and data efficiency into account, making it a unified measure for model comparison. We use the proposed framework to evaluate popular vision representation learning methods. We report the description length of representations, provide insights of their model scaling property and preferred readout model, as well as compare data efficiency to training from scratch.

With the MDL formulation the performance metric under consideration is the log-loss (cross-entropy), which directly corresponds to compression. Extensions to other losses such as 0/1-loss etc. will need careful consideration. We here report a single metric per downstream-task and representation. MDL supports non-stationary sequences and future work could aim for reporting a single metric per sequence-of-tasks by concatenating them. However this poses questions about the order of downstream tasks, and how to deal with multiple output spaces $\mathcal{Y}$, and whether forward transfer between these tasks is permissible.

## REFERENCES


Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pp. 17–36. JMLR Workshop and Conference Proceedings, 2012.

Léonard Blier and Yann Ollivier. Do deep learning models have too many parameters? an information theory viewpoint. *CoRR*, abs/1802.07044, 2018. URL http://arxiv.org/abs/1802.07044.

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.

Jorg Bornschein, Yazhe Li, and Markus Hutter. Sequential learning of neural neyworks for prequential mdl. 2022.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *CoRR*, abs/2104.14294, 2021. URL https://arxiv.org/abs/2104.14294.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709, 2020. URL https://arxiv.org/abs/2002.05709.

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

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7:1–30, 2006.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. URL https://arxiv.org/abs/2010.11929.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. *CoRR*, abs/2006.07733, 2020. URL https://arxiv.org/abs/2006.07733.

P. D. Grünwald. *The Minimum Description Length Principle*. The MIT Press, Cambridge, 2007a.

Peter Grünwald and Wojciech Kotlowski. Prequential plug-in codes that achieve optimal redundancy rates even if the model is wrong. *CoRR*, abs/1002.0757, 2010. URL http://arxiv.org/abs/1002.0757.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. *CoRR*, abs/2111.06377, 2021. URL https://arxiv.org/abs/2111.06377.

Olivier J. Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, S. M. Ali Eslami, and Aäron van den Oord. Data-efficient image recognition with contrastive predictive coding. *CoRR*, abs/1905.09272, 2019. URL http://arxiv.org/abs/1905.09272.

Mark Herbster and Manfred K. Warmuth. Tracking the best expert, 1998.

Mark Herbster and Manfred K. Warmuth. Tracking the best expert. *Machine Learning*, 32:151–178, 2004.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. *CoRR*, abs/1909.03368, 2019. URL http://arxiv.org/abs/1909.03368.

Wouter Koolen and Steven De Rooij. Combining expert advice efficiently. *arXiv preprint arXiv:0802.2015*, 2008.

Wouter M. Koolen and Steven de Rooij. Combining expert advice efficiently. *CoRR*, abs/0802.2015, 2008. URL http://arxiv.org/abs/0802.2015.

Wouter M. Koolen and Steven de Rooij. Universal codes from switching strategies. *CoRR*, abs/1311.6536, 2013. URL http://arxiv.org/abs/1311.6536.

Yazhe Li, Roman Pogodin, Danica J Sutherland, and Arthur Gretton. Self-supervised learning with kernel dependence maximization. *Advances in Neural Information Processing Systems*, 34: 15543–15556, 2021.

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

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

Cinjon Resnick, Zeping Zhan, and Joan Bruna. Probing the state of the art: A critical look at visual representation evaluation. *CoRR*, abs/1912.00215, 2019. URL http://arxiv.org/abs/1912.00215.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *CoRR*, abs/2012.12877, 2020. URL https://arxiv.org/abs/2012.12877.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. URL http://arxiv.org/abs/1807.03748.

Tim van Erven, Peter Grünwald, and Steven A. H. de Rooij. Catching up faster by switching sooner: a predictive approach to adaptive estimation with an application to the aic–bic dilemma. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74, 2012.

Joel Veness, Kee Siong Ng, Marcus Hutter, and Michael Bowling. Context tree switching. In *Proc. Data Compression Conference (DCC'12)*, pp. 327–336, Snowbird, Utah, USA, 2012. IEEE Computer Society. ISBN 978-1-4673-0715-4. doi: 10.1109/DCC.2012.39. URL http://arxiv.org/abs/1111.3182.

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

William F. Whitney, Min Jae Song, David Brandfonbrener, Jaan Altosaar, and Kyunghyun Cho. Evaluating representations by the complexity of learning low-loss predictors. *CoRR*, abs/2009.07368, 2020. URL https://arxiv.org/abs/2009.07368.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome T. Connor, Tomás Kociský, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and Phil Blunsom. Learning and evaluating general linguistic intelligence. *CoRR*, abs/1901.11373, 2019. URL http://arxiv.org/abs/1901.11373.

Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, André Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, Lucas Beyer, Olivier Bachem, Michael Tschannen, Marcin Michalski, Olivier Bousquet, Sylvain Gelly, and Neil Houlsby. The visual task adaptation benchmark. *CoRR*, abs/1910.04867, 2019. URL http://arxiv.org/abs/1910.04867.

# A APPENDIX

## A.1 PROOF OF REGRET BOUND

Assume that $x \sim P$ and a sequence of examples $x^N = \{x_1, \ldots, x_N\}$ are i.i.d samples from $P$. Let $\hat{\theta}(x^N)$ be the parameters of a readout model $M$ that best fit the sequence $x^N$, i.e. $\hat{\theta}(x^N) = \inf_\theta M(x^N; \theta)$. Since $x^N$ are i.i.d samples from $P$, $\hat{\theta}(x^N)$ is the ML estimator of $x^N$. We assume that all readout models are from exponential family. But we do not assume $P \in M$, i.e. the model can be misspecifed.

Define the expected regret as the regret w.r.t the best sequence of readout head $\xi^*_{1:N}$ and their best fit parameters $\hat{\theta}_{\xi^*_{1:N}}(x^N)$ for $x^N$:

$$
\begin{aligned}
R(n) =& \mathbb{E}_P \left[ \sum_{t=1}^{N} \log M_{\xi^*_t}(x_t; \hat{\theta}_{\xi^*_t}(x^N)) - \log \sum_{\xi_{1:N}} p(\xi_{1:N}) \prod_{t=1}^{N} M_{\xi_t}(x_t | \hat{\theta}_{\xi_t}(x^{t-1})) \right] \\
\leq& \mathbb{E}_P \left[ \sum_{t=1}^{N} \log M_{\xi^*_t}(x_t; \hat{\theta}_{\xi^*_t}(x^N)) - \log p(\xi^*_{1:N}) \prod_{t=1}^{N} M_{\xi^*_t}(x_t | \hat{\theta}_{\xi^*_t}(x^{t-1})) \right] \\
=& \mathbb{E}_P \left[ \sum_{t=1}^{N} \Big( \log M_{\xi^*_t}(x_t; \hat{\theta}_{\xi^*_t}(x^N)) - \log M_{\xi^*_t}(x_t | \hat{\theta}_{\xi^*_t}(x^{t-1})) \Big) - \log p(\xi^*_{1:N}) \right] \\
=& - \mathbb{E}_P \left[ \log p(\xi^*_{1:N}) \right] + \sum_{t=1}^{N} \mathbb{E}_P \left[ \log M_{\xi^*_t}(x_t; \hat{\theta}_{\xi^*_t}(x^N)) - \log M_{\xi^*_t}(x_t | \hat{\theta}_{\xi^*_t}(x^{t-1})) \right] \qquad (3)
\end{aligned}
$$

Since the readout models are from exponential family, they can be parameterized instead by the mean parameters. Recall that the exponential family is the probability distribution that can be written in the form of $P(x; \eta) = h(x) \exp\left(\eta^T T(x) - A(\eta)\right)$ where $\eta$ is the natural parameters, $T(x)$ is the sufficient statistics, $A(\eta)$ is the cumulant function. The mean parameters are $\mu := \mathbb{E}\left[T(x)\right]$ and have a one-to-one mapping to natural parameters $\eta$ (we use $\eta(\mu)$ to denote the mapping). Therefore, the exponential family can also be parameterized by the mean parameters $P(x; \mu) = h(x) \exp\left(\eta(\mu)^{T(x)} T - A(\eta(\mu))\right)$. We also know that the sufficient statistics is also an exponential family in the form of $P(T; \mu) = h^{'}(T) \exp\left(\eta(\mu)^T T - A(\eta(\mu))\right)$. Because of the assumption that $M$ is exponential family model, $\mathbb{E}_{x \sim P}\left[\log M(x; \theta)\right] = \mathbb{E}_{T \sim P}\left[\log M(T; \mu)\right]$. Here, we use the same $P$ as the distribution induced on $T$. The maximum likelihood estimator of the mean parameters is $\hat{\mu}(x^N) = \frac{1}{N} \sum_{t=1}^{N} T(x_t)$.

Given any estimator $\hat{\mu}$, we can Taylor expand the likelihood at $\mu = \mathbb{E}_P[T(x)]$:

$$
\begin{aligned}
& \log M(x; \hat{\mu}) \\
=& \log M(x; \mu) + (\hat{\mu} - \mu)^T \nabla_{\hat{\mu}} \log M(x, \hat{\mu})|_\mu + \frac{1}{2} (\hat{\mu} - \mu)^T \nabla^2_{\hat{\mu}} \log M(x, \hat{\mu})|_\mu (\hat{\mu} - \mu) + O(1) \\
\Longrightarrow& \mathbb{E}_P \left[ \log M(x; \hat{\mu}) - \log M(x; \mu) \right] \\
&= \frac{1}{2} (\hat{\mu} - \mu)^T \mathbb{E}_P \left[ \nabla_{\hat{\mu}} \log M(x, \hat{\mu})|_\mu \right] (\hat{\mu} - \mu) \\
&= \frac{1}{2} (\hat{\mu} - \mu)^T \mathcal{I}(\mu) (\hat{\mu} - \mu) \\
&= \frac{1}{2} (\hat{\mu} - \mu)^T Cov^{-1}_{M_\mu}[T(x)] (\hat{\mu} - \mu)
\end{aligned}
$$

Where $\mathcal{I}(\mu)$ is the Fisher information. We first use $\mathbb{E}_P\left[\nabla_{\hat{\mu}} \log M(x, \hat{\mu})\right]|_\mu = \nabla_{\hat{\mu}} \mathbb{E}_P[\log M(x, \hat{\mu})]|_\mu = 0$. Then we use $\mathcal{I}(\mu) = Cov^{-1}_{M_\mu}[T(x)]$, RHS is the covariance

matrix under the distribution $M_\mu$. For a readout model $M$, after seeing $x^t$ examples, we have

$$\begin{aligned}
&\mathbb{E}_{x^{t-1}\sim P}\left[\log M(x_t;\hat{\mu}(x^{t-1})) - \log M(x_t;\mu)\right]\\
=&\mathbb{E}_P\left[\frac{1}{2}(\hat{\mu}-\mu)^T Cov^{-1}_{M_\mu}[T(x)](\hat{\mu}-\mu)\right]\\
=&\frac{1}{2}\mathbb{E}_P\left[Tr\left(Cov^{-1}_{M_\mu}[T(x)](\hat{\mu}-\mu)(\hat{\mu}-\mu)^T\right)\right]\\
=&\frac{1}{2}Tr\left(Cov^{-1}_{M_\mu}[T(x)]\mathbb{E}_P\left[(\hat{\mu}-\mu)(\hat{\mu}-\mu)^T\right]\right)\\
=&\frac{1}{2}\frac{1}{t-1}Tr\left(Cov^{-1}_{M_\mu}[T(x)]Cov_P\left[T(x)\right]\right)
\end{aligned}$$

We use the fact that trace is a linear operator, therefore we could exchange the order with expectation. The last step comes from$\hat{\mu}(x^{t-1}) = \frac{1}{t-1}\sum_i T(x_i)$; denote $T_i = T(x_i)$, we have $\mathbb{E}_P\left[(\hat{\mu}-\mu)(\hat{\mu}-\mu)^T\right] = \mathbb{E}_P\left[(\frac{1}{t-1}\sum_i T_i - \mu)(\frac{1}{t-1}\sum_i T_i - \mu)^T\right] = \frac{1}{t-1}\mathbb{E}_P[TT^T] - \frac{1}{t-1}\mu\mu^T = \frac{1}{t-1}Cov_P[T(x)]$.Note here the covariance is under the true data distribution $P$.

For each term inside the sum over timestep $t$ in Equation (3), we have

$$\begin{aligned}
&\mathbb{E}_P\left[\log M_{\xi_t^*}(x_t;\hat{\theta}_{\xi_t^*}(x^N)) - \log M_{\xi_t^*}(x_t|\hat{\theta}_{\xi_t^*}(x^{t-1}))\right]\\
=&\mathbb{E}_P\left[\log M_{\xi_t^*}(x_t;\hat{\mu}_{\xi_t^*}(x^N)) - \log M_{\xi_t^*}(x_t|\hat{\mu}_{\xi_t^*}(x^{t-1}))\right]\\
=&\mathbb{E}_P\left[\log M_{\xi_t^*}(x_t;\hat{\mu}_{\xi_t^*}(x^N)) - \log M_{\xi_t^*}(x_t;\mu) + \log M_{\xi_t^*}(x_t;\mu) - \log M_{\xi_t^*}(x_t|\hat{\mu}_{\xi_t^*}(x^{t-1}))\right]\\
=&-\frac{1}{2}(\frac{1}{t-1}-\frac{1}{N})Tr\left(Cov^{-1}_{M_{\mu_{\xi_t^*}}}[T_{\xi_t^*}(x)]Cov_P\left[T_{\xi_t^*}(x)\right]\right) + O(1)
\end{aligned}$$

We first change it from parameterized by natural parameters to mean parameters. Then for each term, we can compare it to $\mu = \mathbb{E}_P[T(x)]$.

The total sum of $t = 1, \ldots, T$ in Equation (3) is

$$\begin{aligned}
&\sum_{t=1}^{N}\mathbb{E}_P\left[\log M_{\xi_t^*}(x_t;\hat{\theta}_{\xi_t^*}(x^N)) - \log M_{\xi_t^*}(x_t|\hat{\theta}_{\xi_t^*}(x^{t-1}))\right]\\
\leq&\sum_{t=2}^{N}\frac{1}{2}(\frac{1}{t-1}-\frac{1}{N})Tr\left(Cov^{-1}_{M_{\mu_{\xi_t^*}}}[T_{\xi_t^*}(x)]Cov_P\left[T_{\xi_t^*}(x)\right]\right)\\
\leq&\frac{1}{2}\sum_{t=2}^{N}(\frac{1}{t-1}-\frac{1}{N})\max_k\left\{Tr\left(Cov^{-1}_{M_{\mu_k^*}}[T_{\xi_t^*}(x)]Cov_P\left[T_{\xi_t^*}(x)\right]\right)\right\}\\
\leq&\frac{1}{2}\max_k\left\{Tr\left(Cov^{-1}_{M_{\mu_k^*}}[T_k(x)]Cov_P\left[T_k(x)\right]\right)\right\}\left(\sum_{t=2}^{N}\frac{1}{t-1}\right)\\
\leq&\frac{1}{2}\max_k\left\{Tr\left(Cov^{-1}_{M_{\mu_k^*}}[T_k(x)]Cov_P\left[T_k(x)\right]\right)\right\}\log N
\end{aligned}$$

Inequality in line 2 uses that When $t = 1$, we use the prior and incur a constant regret. Inequality in line 5 uses harmonic series $\sum_{i=1}^{N}\frac{1}{i} = \log N + const$. Note that if there is no model misspecification in any of the readout models, we have $M_{\mu_k^*} = P$. In this case, $Cov^{-1}_{M_{\mu_k^*}}[T_k(x)]Cov_P\left[T_k(x)\right] = I$ and $Tr(I) = k$. We find the classical result for regret of prequential code $\frac{k}{2}\log N$.

Combining with Equation (3), the expected regret is then

$$R(n) \leq -\mathbb{E}_P\left[\log p(\xi^*_{1:N})\right] + \frac{1}{2}\max_k\left\{Tr\left(Cov^{-1}_{M_{\mu_k^*}}[T_k(x)]Cov_P\left[T_k(x)\right]\right)\right\}\log N \tag{4}$$

The expected regret for our readout model switching is composed of 2 terms: one is the regret from expert switching algorithm and the other is the regret of the readout models. The growth of second

term is well behaved and bounded by $C \log N$ where $C$ is a constant that relates to the model family chosen. The first term is not necessarily bounded by $\log N$, but relates to the switching strategy chosen. However, for many switching algorithms, such as Bayesian mixture, fixed share with decreasing switching rate and run-length code, $-\mathbb{E}_P\left[\log p(\xi^*_{1:N})\right]$ is in the order of $\log N$. In particular, for fixed share with decreasing switching rate, we have the result of $-\mathbb{E}_P\left[\log p(\xi^*_{1:N})\right] \leq m \log K + (m-1)\log \frac{N}{m-1}$ where $m-1$ is the number of switch points (i.e. $m$ is the number of blocks in which the same expert is used). See Koolen & de Rooij (2013) for proof of the fixed share and more details of the regret bound of other switching strategy. A similar proof is given by Veness et al. (2012) for the particular case of $m = 2$.

### A.2 Distribution Shift between Train and Test in VTAB

Figure 6 shows the top-1 accuracy on train and test subsets with the original VTAB and reshuffled VTAB datasets. For the same dataset, a difference in test set performance between the original and reshuffled datasets suggests data shift between the two versions. KittiDistance have a significant data shift. DMLab, PatchCamelyon and SmallNorbAzimuth also have distribution shift but to a much less degree compared to KittiDistance. To make sure that the dataset is i.i.d, we combine all its subsets, randomly shuffle then partition into the same number of train/test examples. We only take the training subset for online learning, which results in sequence lengths showed in Table 2. Our experiments are run with the reshuffled version of the VTAB dataset.

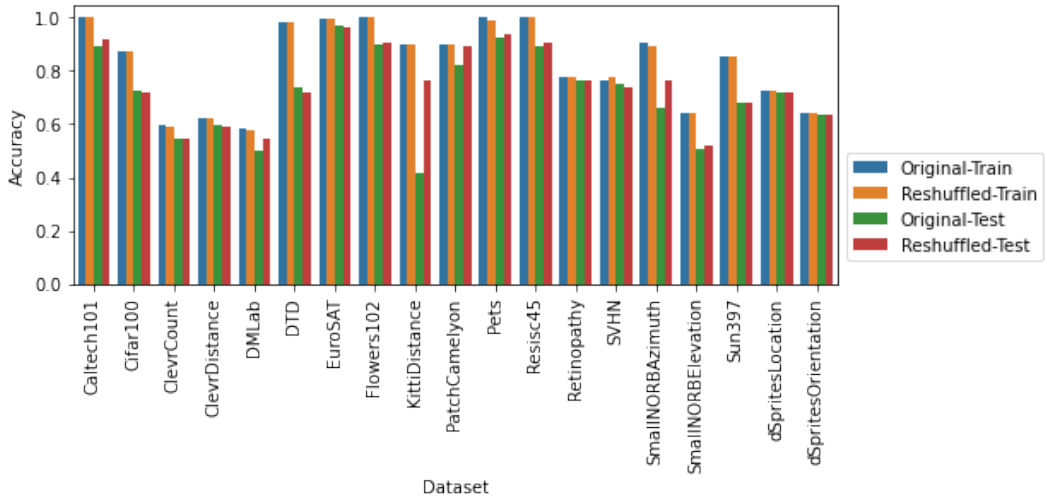



Figure 6: Comparison of train and test accuracies on the original and reshuffled VTAB datasets.

### A.3 2-Stage Online and Offline Implementation

The 2-stage implementation is straight forward: first stage, we compute the cross entropy losses for all models at each timestep and store the results; second stage, we use model switching to compute the final codelength (see Algorithm 2).

### A.4 Implementation Details

#### A.4.1 VTAB

**Frozen Encoders** For frozen encoders, we use the readout models: linear layer and MLP of 1-7 hidden layers. We use AdamW as the optimizer and hyperparameters listed in Table 3. The switching probability for fix share if $\alpha_t = 10/t$. Images are center cropped and resized to $224 \times 224$.

| Group | Dataset | Number of Examples |
|---|---|---|
| Natural | Caltech101 | 3060 |
| | Cifar100 | 50000 |
| | DTD | 3760 |
| | Flowers102 | 2040 |
| | Pets | 3680 |
| | SVHN | 73257 |
| | Sun397 | 87003 |
| Special | EuroSAT | 21600 |
| | PatchCamelyon | 294912 |
| | Resisc45 | 25200 |
| | Retinopathy | 46032 |
| Structured | ClevrCount | 70000 |
| | ClevrDistance | 70000 |
| | DMLab | 88178 |
| | DSpritesLocation | 663552 |
| | DSpritesOrientation | 663552 |
| | KittiDistance | 6770 |
| | SmallNORBAzimuth | 36450 |
| | SmallNORBElevation | 36450 |

Table 2: Number of examples in the reshuffled VTAB datasets.

**Algorithm 2** MDL with Read-out Model Switching (2-Stage)

Stage 1:

**Require:** data $\mathcal{D} = (x_t, y_t)_{t=1}^T$; $K$ readout models; subroutine that updates the parameters given a dataset *UpdateParameters*

Initialize: model parameters $\theta_1, \ldots, \theta_K$; an empty list $L$ to store loss per step and per model
**for** $t = 1$ **to** $T$ **do**
  Compute next-step loss of $K$ readout models: $\mathcal{L}_t^k := -\log p_k(y_t|x_t) = -\log p(y_t|x_t; \theta_k)$
  Store $\mathcal{L}_t^1, \ldots, \mathcal{L}_t^K$ to $L$
  **for** $k = 1$ **to** $K$ **do**
    Update parameters: $\theta_k \leftarrow$ *UpdateParameters*$(\theta_k, \mathcal{D}_{\leq t})$
  **end for**
**end for**
**return** $L$

Stage 2:

**Require:** Cross entropy results from stage 1 $L$; initial model probability $p(\xi_1)$ and switching strategy $p(\xi_t|\xi_{t-1})$

Initialize: $s = \log p(\xi_1)$; an empty list $Q$ to store posterior $p(\xi_t|\mathcal{D}_{<t})$
**for** $t = 1$ **to** $T$ **do**
  Compute $\log p(\xi_t, y^{t-1}|x^{t-1})$: $s \leftarrow \log \sum_{\xi_{t-1}} \exp\left(\log p(\xi_t|\xi_{t-1}) + s\right)$
  Compute posterior $p(\xi_t|\mathcal{D}_{<}t)$ and store in $Q$:
  $\log p(\xi_t|\mathcal{D}_{<}t) = \log p(\xi_t, y^{t-1}|x^{t-1}) - \log \sum_{\xi_t} p(\xi_t, y^{t-1}|x^{t-1})$
  Get $\mathcal{L}_t^1, \ldots, \mathcal{L}_t^K$ from $L$ and combine $K$ models to update $\log p(\xi_t, y^t|x^t)$: $s \leftarrow -\mathcal{L}_t^{\xi_t} + s$
**end for**
Compute total codelength $\mathcal{L}_{switch} \leftarrow -\log \sum_{\xi_T} \exp(s)$
**return** $\mathcal{L}_{switch}$ and $Q$

**Train from Scratch** This section correspond to the training from scratch experiment on VTAB presented in Section 4.4. We use SGD with nesterov momentum for training from scratch. We take the center crop and resize the image to $224 \times 224$ for preprocessing. No readout models is needed as the encoder is trained on the relevant task. For each VTAB task, we uniformly randomly select 300 configurations from the hyperparameters listed in Table 4.

| Parameter | Values |
|---|---|
| Batch size | 32 |
| Number of replay-streams | { 3, 10, 30, 100 } |
| Learning rate | { 1e-4, 3e-4, 1e-3, 3e-3 } |
| AdamW $\beta_1$ | { 0.5, 0.7, 0.9 } |
| Weight decay | { 0.0, 1e-6, 1e-4, 1e-2 } |
| EMA step size (Polyak averaging) | { 1e-4, 1e-2, 1.0 } |

Table 3: Hyperparameters for computing MDL of frozen encoders on VTAB.

| Parameter | Values |
|---|---|
| Batch size | 32 |
| Number of replay-streams | { 50, 100, 150, 200 } |
| Learning rate | { 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1.0 } |
| momentum | 0.9 |
| Weight decay | { 1.e-5, 3e-5, 1e-4, 3e-4, 1e-3, 3e-3, 1e-2, 3e-2 } |
| EMA step size (Polyak averaging) | { 1e-3, 1e-2, 1e-1, 1.0 } |

Table 4: Hyperparameters for computing MDL with training ResNet-50 from scratch on VTAB.

**Fine-tuning** This section correspond to the fine-tuning experiment on VTAB presented in Section 4.4. We use ResNet-50 encoder for the fine-tuning experiment. Read-out model is a linear layer. The encoder parameters are initialized with the pre-trained parameters of BYOL, but updated on the downstream task. We use SGD with nesterov momentum for fine-tuning pre-trained encoders. Images are center cropped and resized to $224 \times 224$ for preprocessing. For each VTAB task, we uniform randomly select 300 configurations from the hyperparameters listed in Table 5.

| Parameter | Values |
|---|---|
| Batch size | 32 |
| Number of replay-streams | { 10, 30, 50 } |
| Learning rate | { 1e-4, 3e-4, 1e-3, 3e-3, 0.01, 0.03, 0.1, 0.3, 1. } |
| momentum | 0.9 |
| Weight decay | { 0.0, 1.e-5, 3e-5, 1e-4, 3e-4, 1e-3, 3e-3 } |
| EMA step size (Polyak averaging) | { 1e-3, 1e-2, 1e-1, 1.0 } |

Table 5: Hyperparameters for computing MDL with fine-tuning pre-trained encoders on VTAB.

#### A.4.2 IMAGENET

**Frozen Encoders** Images are preprocessed with spatial augmentation during training: randomly cropped, randomly flipped and resized to $224 \times 224$. We use the readout models: linear layer and MLP of 1-3 hidden layers. We use AdamW as the optimizer and hyperparameters listed in Table 6. The switching probability for fix share if $\alpha_t = 1/t$.

**Fine-tuning** This section correspond to the fine-tuning experiment on ImageNet presented in Section 4.4. We use a ResNet-50 encoder for the fine-tuning experiment. Read-out model is a linear layer. The encoder parameters are initialized with the pre-trained parameters from BYOL, but updated on the downstream task. We use AdamW and spatial augmentation for preprocessing. Images are randomly cropped and resized to $224 \times 224$. We use the same set of hyperparameters as the frozen encoder experiments listed in Table 6.

**Train from Scratch** We train a standard ResNet-50 architecture from scratch on ImageNet and follow the experimental setting described in (Bornschein et al., 2022). In particular we use replay-streams training with uniform replay, AdamW, use randaugment for data augmentation, and forward

| Parameter | Values |
|---|---|
| Batch size | 512 |
| Number of replay-streams | { 10, 30, 50 } |
| Learning rate | { 3e-5, 1e-4, 3e-4, 1e-3 } |
| AdamW $\beta_1$ | { 0.5, 0.7 } |
| Weight decay | { 1e-2, 1. } |
| EMA step size (Polyak averaging) | { 0.01, 1. } |

Table 6: Hyperparameters for computing MDL of frozen encoders on ImageNet.

calibration. We run a random hyperparameter search by sampling 50 hyperparameter configurations from the intervals listed in appendix A.4.2.

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

### A.5 Comparison of Switching Strategies

We describe briefly other switching strategy we experimented with our evaluation framework:

- Bayesian mixture: switching probability is 1 after the first step.

$$L_{BM}(y_{1:T}|\phi) = -\log \sum_k p(k) \prod_{t=1}^{T} p(y_t|\phi_t, \hat{\theta}_k(\phi_{<t}, y_{<t}))$$

- Elementwise mixture: at each step, it is allowed to switch to another model with a fixed probability which doesn't depend on the current model used.

$$L_{EW}(y_{1:T}|\phi) = -\log \sum_{\xi_{1:T}} \prod_{t=1}^{T} p(y_t|\phi_t, \hat{\theta}_{\xi_t}(\phi_{<t}, y_{<t}))w(\xi_t)$$

  Where $w(\xi_t)$ is a fixed mixture weight which doesn't depend on $\xi_{t-1}$.

- Switch distribution: A more sophisticated switching strategy which augments the transition with two latent variables $u$ and $s$ for unstable and stable chain respectively. Once switching to the stable chain, it stay with the same model for the rest of the sequence. For more details, see Koolen & De Rooij (2008).

Figure 7 compares the above switching strategies with the fixed share with decreasing switching rate on ImageNet using a ResNet50 pre-trained with BYOL. Bayesian mixture and elementwise mixture perform worse than fixed share by a large margin. Switch distribution is very close to fixed share performance.

### A.6 MDL on VTAB Datasets

We report the final MDL score using fixed share switching strategy on all the VTAB datasets in Table 7. To better compare the numbers, we average the MDL score by the number of examples in the datasets. For cumulative MDL score, we should multiple by the number of examples which is reported in Table 2.

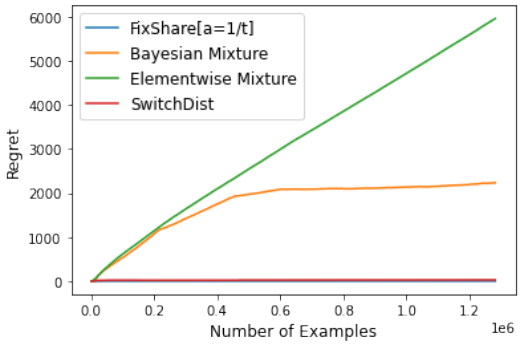



Figure 7: Regret of Bayesian mixture, elementwise mixture and switching distribution compared to the fixed share baseline on ImageNet. Pretrained model is BYOL with ResNet50 backbone.

| | | Natural | | | | | | | Special | | | | Structured | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Architecture | Objective | Caltech101 | Cifar100 | DTD | Flowers102 | Pets | SVHN | Sun397 | EuroSAT | PatchCamelyon | Resisc45 | Retinopathy | ClevrCount | ClevrDistance | DMLab | DSpritesLocation | DSpritesOrientation | KittiDistance | SmallNORBAzimuth | SmallNORBElevation |
| ResNet-50 | SUP | 0.81 | 1.12 | 1.35 | 1.23 | 0.35 | 0.82 | 1.42 | 0.14 | 0.17 | 0.45 | 0.72 | 0.91 | 0.87 | 0.90 | 0.14 | 0.26 | 0.58 | 0.51 | 0.88 |
| ResNet-50 | SimCLR | 0.98 | 1.33 | 1.33 | 1.30 | 0.74 | 0.66 | 1.54 | 0.18 | 0.20 | 0.46 | 0.73 | 0.90 | 0.79 | 0.92 | 0.08 | 0.25 | 0.58 | 0.29 | 0.77 |
| ResNet-50 | BYOL | 0.78 | 0.97 | 1.21 | 1.06 | 0.48 | 0.69 | 1.32 | 0.13 | 0.17 | 0.39 | 0.70 | 0.78 | 0.77 | 0.77 | 0.13 | 0.16 | 0.55 | 0.24 | 0.65 |
| ResNet-101 | SUP | 0.76 | 1.02 | 1.31 | 1.27 | 0.34 | 0.95 | 1.38 | 0.14 | 0.18 | 0.48 | 0.73 | 0.89 | 0.91 | 0.91 | nan | 0.23 | 0.59 | 0.52 | 0.92 |
| ResNet-101 | BYOL | 0.72 | 0.86 | 1.17 | 1.04 | 0.43 | 0.78 | 1.25 | 0.13 | 0.17 | 0.38 | 0.70 | 0.77 | 0.79 | 0.77 | 0.11 | 0.16 | 0.55 | 0.24 | 0.63 |
| ResNet-152 | SUP | 0.74 | 0.99 | 1.32 | 1.28 | 0.34 | 0.99 | 1.37 | 0.15 | 0.18 | 0.50 | 0.72 | 0.90 | 0.94 | 0.91 | 0.18 | 0.23 | 0.60 | 0.50 | 0.92 |
| ResNet-152 | BYOL | 0.71 | 0.84 | 1.15 | 1.07 | 0.42 | 0.79 | 1.22 | 0.13 | 0.17 | 0.40 | 0.70 | 0.77 | 0.77 | 0.78 | 0.10 | 0.15 | 0.55 | 0.25 | 0.66 |
| ViT/S16 | SUP | 0.83 | 0.95 | 1.45 | 1.45 | 0.38 | 1.23 | 1.38 | 0.19 | 0.20 | 0.56 | 0.72 | 1.06 | 1.17 | 1.14 | nan | 0.38 | 0.64 | 0.61 | 1.00 |
| ViT/S16 | MAE | 1.83 | 2.01 | 1.76 | 2.10 | 1.54 | 1.64 | 2.27 | 0.17 | 0.15 | 0.64 | 0.71 | 0.97 | 1.16 | 1.09 | 1.75 | 0.27 | 0.64 | 0.59 | 0.69 |
| ViT/B16 | SUP | 0.71 | 0.82 | 1.35 | 1.17 | 0.32 | 0.99 | 1.28 | 0.14 | 0.17 | 0.43 | 0.70 | 0.94 | 1.09 | 1.01 | 0.35 | 0.25 | 0.60 | 0.47 | 0.88 |
| ViT/B16 | DINO | 0.68 | 0.68 | 1.14 | 0.94 | 0.37 | 0.76 | 1.15 | 0.11 | 0.13 | 0.34 | 0.68 | 0.78 | 0.94 | 0.72 | 0.17 | 0.19 | 0.55 | 0.28 | 0.57 |
| ViT/B16 | MAE | 1.12 | 1.21 | 1.49 | 1.64 | 0.93 | 0.93 | 1.60 | 0.13 | 0.14 | 0.47 | 0.70 | 0.70 | 0.80 | 0.92 | 0.03 | 0.14 | 0.58 | 0.29 | 0.48 |
| ViT/L16 | SUP | 0.81 | 0.88 | 1.52 | 1.41 | 0.37 | 1.06 | 1.40 | 0.18 | 0.18 | 0.52 | 0.72 | 0.96 | 1.05 | 1.04 | 0.32 | 0.27 | 0.61 | 0.59 | 0.91 |
| ViT/L16 | MAE | 0.92 | 0.86 | 1.39 | 1.50 | 0.53 | 0.82 | 1.36 | 0.12 | 0.13 | 0.43 | 0.69 | 0.60 | 0.76 | 0.85 | 0.02 | 0.11 | 0.56 | 0.21 | 0.42 |

Table 7: Fixed share codelength per example reported for each VTAB dataset.

### A.7 Posterior of the readout models

Figure 8 shows the posterior of readout models $p(\xi_t = k|\mathcal{D}_{<t})$ where $k \in [1, \ldots, K]$ on ImageNet for 4 different architecture and pre-training methods: ResNet50 + SimCLR, ResNet50 + BYOL, ViT/B16 + DINO and ViT/B16 + MAE. We use 4 readout models: linear layer and MLP with 1-3 hidden layers.

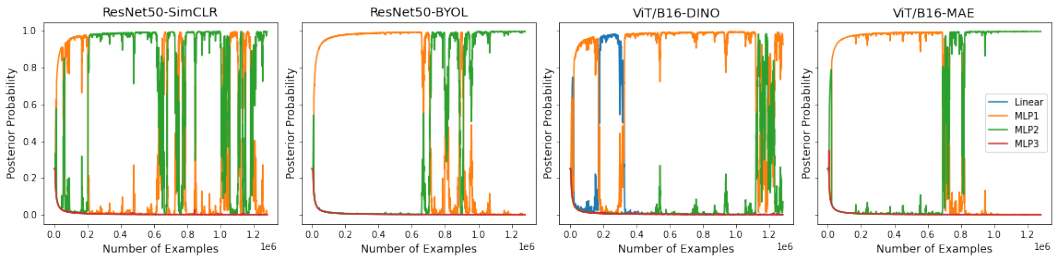



Figure 8: Posterior probability of the readout models at each timestep on ImageNet. Pre-trained methods are ResNet50 + SimCLR, ResNet50 + BYOL, ViT/B16 + DINO and ViT/B16 + MAE

### A.8 DATA EFFICIENCY AND FINE-TUNING

Figure 9 shows the regret plots, relative to training from scratch, for training a linear layer on top of the pre-trained and frozen BYOL backbone, training MLPs (1-7 hidden layers) on top of the frozen BYOL backbone and for a linear readout but while fine-tuning all the encoder parameters. The network architecture is a ResNet-50 in all cases.

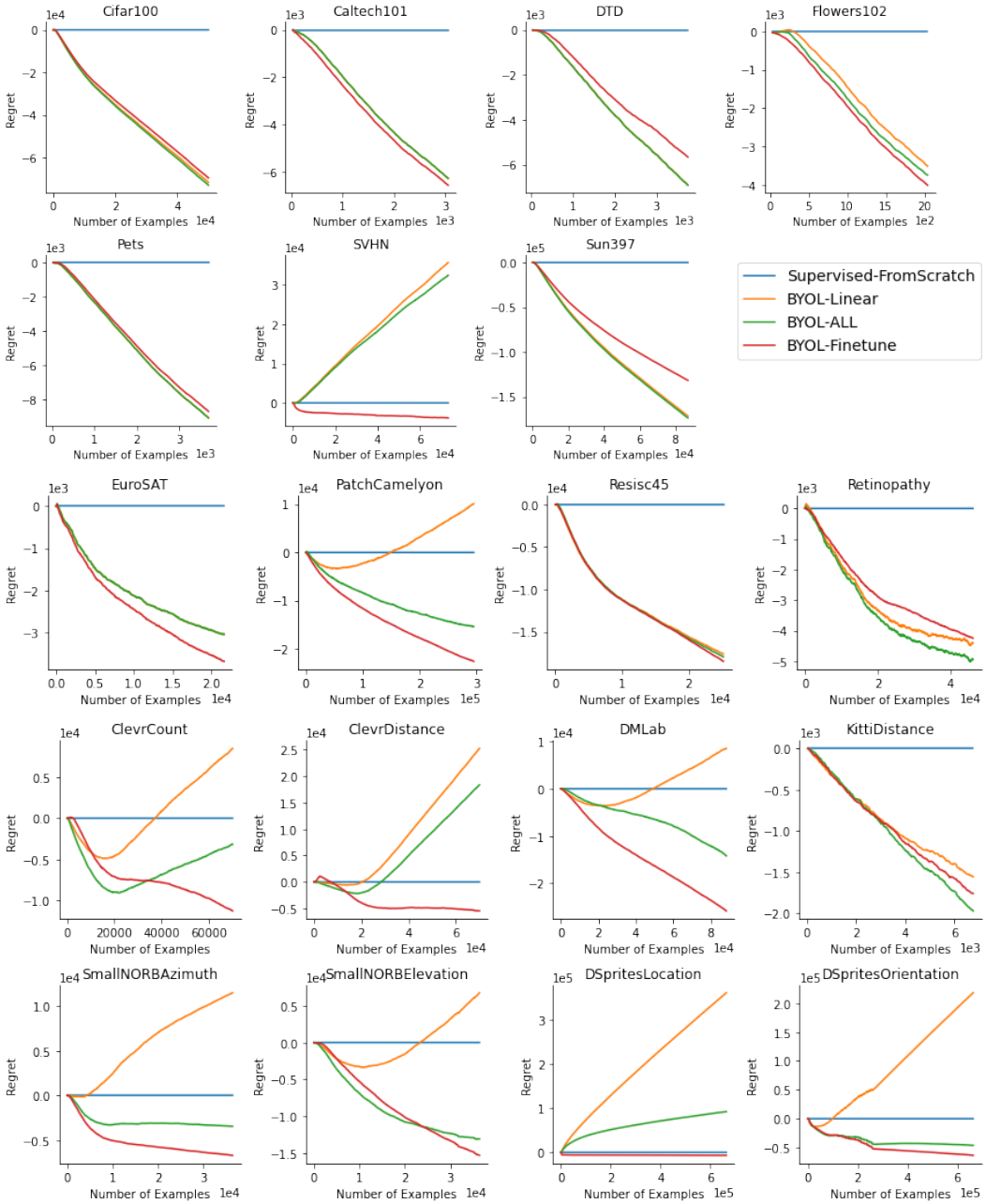



Figure 9: Data efficiency on 19 VTAB datasets. Four methods are compared: training from scratch, training a linear layer with frozen encoder initialized with BYOL, training MLPs with frozen encoder initialized with BYOL and fine-tuning the encoder initialized with BYOL. We plot the cumulative next-step log loss relative to training from scratch as the baseline (in nats; lower is better).