IN THE WILD: FROM ML MODELS TO PRAGMATIC ML SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Enabling robust intelligence in the wild entails learning systems that offer uninterrupted inference while affording sustained learning from varying amounts of data and supervision. Such ML systems must be able to cope with the openness and variability inherent to the real world. The machine learning community has organically broken down this challenging task into manageable sub tasks such as supervised, few-shot, continual, and self-supervised learning; each affording distinct challenges and a unique set of methods. Notwithstanding this remarkable progress, the simplified and isolated nature of these experimental setups has resulted in methods that excel in their specific settings, but struggle to generalize beyond them. To foster research towards more general ML systems, we present a new learning and evaluation framework - **IN THE WILD** (NED). NED naturally integrates the objectives of previous frameworks while removing many of the overly strong assumptions such as predefined training and test phases, sufficient amounts of labeled data for every class, and the closed-world assumption. In NED, a learner faces a stream of data and must make sequential predictions while choosing how to update itself, adapt quickly to novel classes, and deal with changing data distributions; while optimizing for the total amount of compute. We present novel insights from NED that contradict the findings of less realistic or smaller-scale experiments which emphasizes the need to move towards more pragmatic setups. For example, we show that meta-training causes larger networks to overfit in a way that supervised training does not, few-shot methods break down outside of their narrow experimental setting, and self-supervised method MoCo performs significantly worse when the downstream task contains new and old classes. Additionally, we present two new pragmatic methods (Exemplar Tuning and Minimum Distance Thresholding) that significantly outperform all other methods evaluated in NED.

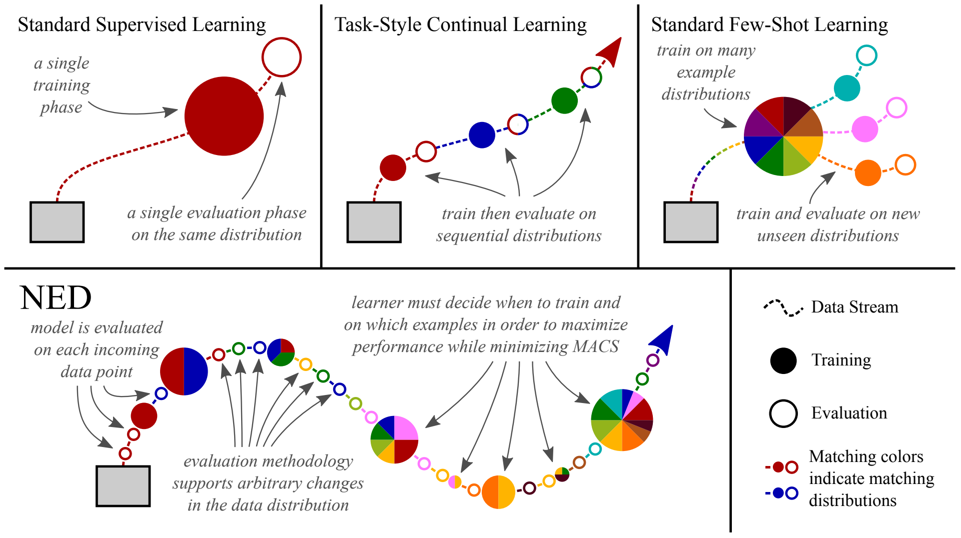

Standard Supervised Learning
a single training phase
a single evaluation phase on the same distribution
Task-Style Continual Learning
train then evaluate on sequential distributions
Standard Few-Shot Learning
train on many example distributions
train and evaluate on new unseen distributions
NED
model is evaluated on each incoming data point
evaluation methodology supports arbitrary changes in the data distribution
learner must decide when to train and on which examples in order to maximize performance while minimizing MACS
Data Stream
Training
Evaluation
Matching colors indicate matching distributions

Figure 1: Comparison of supervised (top-left), continual (top-middle), and few-shot learning (top-right) with NED (bottom). The learner (grey box) accumulates data (dotted path), trains on given data (filled nodes), then is evaluated (empty nodes). The size of the node indicates the scale of the training or evaluation. Each color represents a different set of classes.

1 INTRODUCTION

ML researchers have organically broken down the ambitious task of enabling intelligence into smaller and better defined sub-tasks such as classification for uniformly distributed data, few-shot learning, continual learning, etc. and worked towards developing effective models for each sub-task. The last decade has witnessed staggering progress in each of these sub-tasks. Despite this progress, solutions tailored towards these sub-tasks are not yet ready to be deployed into broader real-world settings.

Consider a general recognition system, a key component in many downstream applications of computer vision. One would expect such a system to recognize a variety of categories without knowing apriori the number of samples for that category (e.g. few shot or not), to adapt to samples from novel categories, to efficiently utilize the available hardware resources, and to know how and when to spend its resources on updating its model. Today's state of the art models make far too many assumptions about the expected data distributions and volumes of supervision and aren't likely to do well in such an unconstrained setting.

To further progress in building learning systems, it is paramount that we design benchmarks that accurately incorporate the key aspects of learning in the real world. But, *what are these aspects?* We posit the following as necessary components: (1) *Sequential Data* - In many application domains the data streams in. Sustained learners must be capable of processing data sequentially. At a given time step, the system may be required to make a prediction or may have the ability to update itself based on available labels and resources. (2) *Flexible Training Phases* – Real-world scenarios rarely delineate when and how a system should be trained. Systems must be capable of making decisions that affect their learning including updating themselves at every time step vs in batches, performing a large number of updates at once vs spreading them out evenly, updating all parameters or just the classifier etc. (3) *Compute Aware* – Real-world systems often have computational constraints not only for inference but also for training. Practical systems must account for the total compute used throughout their lifetime. (4) *Open-world* - The number of possible classes in the wild are typically unknown and constantly evolving. Learning in the real world entails recognizing known classes while identifying when data comes from an unknown class. (5) *X-shot* - Data in the wild often has a different number of examples for each class (few for some and many for others). Current experimental settings assume apriori knowledge of which data regime (few-shot or many-shot) and set of classes a sample is from, but this assumption is unreasonable in many real-world scenarios.

With these properties in mind, we present I**N** THE WIL**D** (NED) (Figure 1) - a framework designed to evaluate learning systems capable of being deployed in real-world scenarios. This is in contrast to existing isolated sub tasks (Figure 1). NED is agnostic to the underlying dataset. It consumes any source of supervised data, samples it to ensure a wide distribution of the number of instances per category, and presents it sequentially to the learning system, ensuring that new categories are gradually introduced. At each time step, the learner is presented with a new data instance which it must classify. Following this, it is also provided the label corresponding to this new data point. It may then choose to process this data and possibly update itself. Beyond this, NED enforces no restrictions on the learning strategy. NED evaluates systems in terms of accuracy and compute throughout the lifetime of the system. As a result, general ML systems that can learn over time with varying amounts of data and update themselves efficiently stand out from our more traditional systems.

In this work, we also present NED-IMAGENET comprising data from ImageNet-22K. The sequential data presented to learners is drawn from a heavy-tailed distribution of 1000 classes (250 of which overlap with the popular ImageNet-1K dataset). Importantly, the data presented has no overlapping instances with the ImageNet-1K dataset, allowing models to leverage advances made over the past few years on this popular dataset. These choices allow us to study learners and their effectiveness with old and new categories as well as rare and common classes.

We evaluate various models and learning strategies on NED-IMAGENET. This includes models that pre-train on ImageNet-1K and finetune on the sequential data using different training schedules, models drawn from the few-shot learning literature, MoCo (He et al., 2019), deep nearest neighbor, and our proposed method. We show that prominent few-shot methods have overfit to their experimental setups and shed new insights on the connection between model capacity and generalization. We show that representations trained with MoCo have unexpected behavior when trained on a mixture of new and old classes. Finally, we propose a novel method, Exemplar Tuning (ET), which excels in both the low and high data regimes and outperforms all other evaluated methods.

2 Aspects of Learning in the Wild

We discuss key elements of learning in real-world scenarios in the context of related work. By integrating these enumerated aspects, we arrive at a general formulation with less impractical assumptions than each subproblem in isolation. The logistics of the framework are in section 3.

Sequential Data New data is an inevitable consequence of our dynamic world and learning over time is a long-standing challenge (Thrun, 1996). In recent years, continual learning (CL) has made notable progress on the problem of learning in a sequential fashion (Li & Hoiem, 2017; Kirkpatrick et al., 2017; Rebuffi et al., 2017; Aljundi et al., 2018; 2019; Riemer et al., 2019). Several setups have been proposed in order to evaluate systems' abilities to learn continuously and primarily focus on *catastrophic forgetting*, a phenomenon where models drastically lose accuracy on old tasks when trained on new tasks. The typical CL setup sequentially presents data from each task then evaluates on current and previous tasks (Li & Hoiem, 2017; Kirkpatrick et al., 2017; Rebuffi et al., 2017). Recent variants have proposed a task-free setting where the data distribution changes without the model's knowledge (Harrison et al., 2019; Riemer et al., 2019; He et al., 2020; Wortsman et al., 2020).

There are two limiting assumptions in CL setups which we remove in NED. The first assumption is that data will be received in large batches with ample data for every class in the task. This assumption circumvents a fundamental challenge of sequential data which is learning new classes from only a few examples. Consider the common scenario in which an ML system encounters an instance from a novel class. The system must determine that it belongs to a new class with no previous examples (zero-shot learning). The next time an instance from the category appears, the system must be capable of one-shot learning, and so forth. The second assumption is that the training and testing phases will be delineated to the system. Deciding when to train and which data to train on is an intrinsic challenge of learning continuously.

Flexible Training Phases Current experimental setups dictate when models will be trained and tested. Ideally, an ML system should be capable of knowing when to train itself, what data to train on, and what to optimize for (Cho et al., 2013). By carefully removing previous assumptions about training and testing phases, NED provides a testing framework for tackling this unexplored problem.

Compute Aware ML systems capable of adapting to their environment over time must account for the computational costs of their learning strategies as well as of inference. Prabhu et al.showed that current CL frameworks do not measure total compute and therefore simple but compute hungry strategies vastly outperform state of the art methods. Previous works have primarily focused on efficient inference (Rastegari et al., 2016; Howard et al., 2017; Kusupati et al., 2020) and some on training costs (Evci et al., 2020). NED evaluates the accuracy and lifetime compute of the system.

Open-world Learning in the wild entails inferring in an open world - where the classes and number of classes are unknown to the learner. Previous works explored static open-world recognition (Liu et al., 2019; Bendale & Boult, 2015) and the related problem of out-of-distribution detection (Hendrycks & Gimpel, 2016). NED presents a natural integration of sequential and open-world learning where the learner must identify new classes and update its known set of classes throughout the stream.

X-Shot Learning from few examples for some classes is an intrinsic aspect of the real-world. As discussed earlier in section 2, few-shot learning is a consequence of learning sequentially. Learning from large, uniform datasets (Russakovsky et al., 2015; Lin et al., 2014) has been the primary focus of supervised learning, although recently few-shot learning has become a popular subfield (Ravi & Larochelle, 2017; Hariharan & Girshick, 2017; Oreshkin et al., 2018; Sun et al., 2019).

While few-shot learning is a step in the right direction, the framework has assumptions that cause methods to overfit which we show through experimental results. The experimental setup for few-shot is typically the n-shot k-way evaluation. Models are trained on base classes during *meta-training* and then tested on novel classes during *meta-testing*. The n-shot k-way experimental setup is limited in two respects. n-shot k-way assumes that a model will always be given exactly n examples for k classes at test time which is an unrealistic assumption. Secondly, most works only evaluate 5-way scenarios with 1, 5, and 10 shots. NED naturally integrates the few-shot problem into its framework by sequentially presenting data from a long tail distribution and evaluates systems across a spectrum of shots and way numbers. Our experimental results on few-shot methods (Finn et al., 2017; Snell et al., 2017) indicate that such methods are over-fitting to the few-shot framework which validates the need for a less restrictive framework such as NED.

3 Framework Details

The NED procedure and setup are designed to be simple. This general formulation contains all key aspects of learning in the wild (Section 2) while making as few assumptions and ad-hoc design decisions as possible.

Algorithm 1 NED Procedure

```
Input: Task $\mathcal{T}$
Input: ML sys.: (pretrained) model $\mathbf{f}$, update strategy $\mathbf{S}$
Output: Evaluations: $E$, Operation Counter: $C$
 1: function NED($\mathcal{T}$, ($\mathbf{f}$, $\mathbf{S}$))
 2:     Evaluations $E = [\,]$
 3:     Datapoints $\mathcal{D} = [\,]$
 4:     Operation Counter $C = 0$.

 5:     while streaming do
 6:         Sample $\{x, y\}$ from $\mathcal{T}$
 7:         prediction $p = \mathbf{f}(\mathbf{x})$ ($A$ operations)
 8:         Flag $n$ indicates if $y$ is a new unseen class
 9:         $E$.insert($\{y, p, n\}$)
10:         $\mathcal{D}$.insert($\{x, y\}$)
11:         Update $\mathbf{f}$ using $\mathbf{S}$ with $\mathcal{D}$ ($B$ operations)
12:         $C$ += $A + B$
13:     end while

14:     return $E, C$
15: end function
```

Procedure NED provides a stream of data to a learning system that consists of a model and learning strategy. At each time step, the system sees one data point and must classify it as either one of the k existing classes or as an unseen one ($k + 1$ classification at each step where k is the number of known classes at the current time step). After inference, the system is provided with a label for that data instance. The learner decides when to train during the stream using previously seen data based on its update strategy. Before streaming, we permit the learning system to pretrain on a given data set to more closely model real-world conditions. We evaluate systems using a suite of metrics including the overall and mean class accuracies over the stream along with the total compute required for training and inference. Algorithm 1 formally shows the procedure for evaluating a system using NED.

Data In this paper, we evaluate methods under the NED framework using a subset of ImageNet-22K (Deng et al., 2009). Traditionally, few-shot learning has used datasets like Omniglot (Lake et al., 2011) & MiniImagenet (Vinyals et al., 2016) and continual learning has focused on MNIST (LeCun, 1998) & CIFAR (Krizhevsky et al., 2009). Some recent continual learning works have used Split-ImageNet (Wen et al., 2020). The aforementioned datasets are mostly small-scale and have few classes. We use the large ImageNet-22K dataset to present new challenges to existing models. Recently, the INaturalist (Van Horn et al., 2018) and LVIS (Gupta et al., 2019) datasets have advocated for heavy-tailed distributions. We follow suit and draw our sequences from a heavy-tailed distribution.

The data consists of a pretraining dataset and 5 different sequences of images. For pretraining we use the standard ImageNet-1K (Russakovsky et al., 2015). This allows us to leverage existing models built by the community as pre-trained checkpoints. Sequence images come from ImageNet-22K after removing ImageNet-1K's images. Each sequence contains images from 1000 classes, 750 of which do not appear in ImageNet-1K. We refer to the overlapping 250 classes as Pretrain classes and the remaining 750 as Novel classes. The sequence is constructed by randomly sampling images from a heavy-tailed distribution of these 1000 classes. Each sequence contains $\sim$ 90000 samples, where head classes contain $>$ 50 and tail classes contain $\leq$ 50 samples. The sequence allows us to study how methods perform on combinations of pretrain vs novel, and head vs tail classes. In Table 1, we show results obtained for sequence 5, and the Appendix F shows results across all sequences. More comprehensive statistics on the data and sequences can also be found in the Appendix A.

Pretraining Supervised pretraining (He et al., 2016) on large annotated datasets like ImageNet facilitates the transfer of learnt representations to help data-scarce downstream tasks. Unsupervised learning methods like autoencoders (Tschannen et al., 2018) and more recent self-supervision methods (Jing & Tian, 2020; Purushwalkam & Gupta, 2020; Gordon et al., 2020) like Momentum Contrast (MoCo) (He et al., 2019) and SimCLR (Chen et al., 2020a) have begun to produce representations as rich as that of supervised learning and achieve similar accuracy on various downstream tasks.

Before deploying the system into the sequential phase, we pretrain our model on ImageNet-1K. In our experiments, we compare how different pretraining strategies (MoCo, meta-training, and supervised training) generalize under more open and challenging conditions. We find new insights such as MoCo breaking down in the sequential setting and meta-training causes larger networks to overfit in a way that supervised training does not.

Evaluation metrics We use the following evaluation metrics in NED.

Metric	**Description**
Overall Accuracy	The accuracy over all elements in the sequence.
Mean Per Class Accuracy	The accuracy for each class in the sequence averaged over all classes.
Total Compute	The total numbers of multiply-accumulate operations for all updates and evaluations accrued over the sequence measured in GMACs (Giga MACs).
Unseen Class Detection	The area under the receiver operating characteristic (AUROC) for the detection of samples that are from previously unseen classes.
Cross-Sectional Accuracies	The mean-class accuracy among classes in the sequence that belong to one of the 4 subcategories: 1) *Pretraining-Head*: Classes with > 50 samples that were included in pretraining 2) *Pretraining-Tail*: Classes with ≤ 50 samples that were included in pretraining 3) *Novel-Head*: Classes with > 50 samples that were not included in pretraining 4) *Novel-Tail*: Classes with ≤ 50 samples that were not included in pretraining

4 Baselines and Methods

We summarize the baselines and our proposed method, Exemplar Tuning. Additional details about the methods and implementation can be found in Appendix B and Appendix C respectively.

Standard Training and Fine-Tuning We evaluate standard model training (update all parameters in the network) and fine-tuning (update only the final linear classifier) with offline batch training. We also investigate the effects of varying the number of layers trained during fine-tuning in Appendix D.

Nearest Class Mean (NCM) Recently, multiple works (Tian et al., 2020; Wang et al., 2019) have found that Nearest Class Mean is comparable to state-of-the-art few-shot methods (Sun et al., 2019; Oreshkin et al., 2018). NCM in the context of deep learning performs a 1-nearest neighbor search in feature space with the centroid of each class as a neighbor. We pretrain a neural network with a linear classifier using softmax cross-entropy loss, then freeze the parameters to obtain features. We also evaluate Meta-Baseline (Chen et al., 2020b) which is the same as NCM in implementation except that a phase of meta-training is done after pretraining. See Appendix B for more details on NCM.

Few-shot Methods We benchmark three representative methods in NED: a) MAML (Finn et al., 2017), b) Prototypical Networks (PTN) (Snell et al., 2017), & c) Weight Imprinting (Qi et al., 2018).

PTN trains a deep feature embedding using 1-nearest neighbor with class centroids and soft nearest neighbor loss. The parameters are trained with meta-training and backpropagation.

MAML is a gradient based approach which uses second-order optimization to learn parameters that can be quickly fine-tuned and adapt to a given task. We tailor MAML to NED by pretraining the model according to the above objective and then fine-tune during the sequential phase.

Weight Imprinting initializes the weights of a cosine classifier as the class centroids, then fine-tunes with a learnable temperature parameter. For further details on few-shot methods see Appendix B.

Learning without Forgetting (LwF) We evaluate LwF to observe whether continual learning techniques can improve performance in NED. LwF leverages knowledge distillation (Buciluă et al., 2006) to retain accuracy on previous training data without storing it. In NED, we adapt LwF to reduce the forgetting of pretrain classes. For further details see Appendix C.

Out-of-Distribution (OOD) Methods We evaluate two methods proposed by Hendrycks & Gimpel (HG) and OLTR (Liu et al., 2019) along with our proposed OOD baseline. The HG baseline thresholds the maximum probability output of the softmax classifier to determine whether a sample is OOD.

We propose Minimum Distance Thresholding (MDT) which utilizes the minimum distance from the sample to all class representations, c_i. In the case of NCM the class representation is the class mean and for a linear layer it is the ith column vector. For distance function d and a threshold t, a sample is out of distribution if: $\mathbf{I}\left(\min_i d\left(c_i, \mathbf{x}\right) < t\right)$. MDT outperforms all other evaluated methods in NED.

Exemplar Tuning (ET) We present a novel method that leverages the inductive biases of instance-based methods and parametric deep learning. The traditional classification layer is effective when given a large number of examples but performs poorly when only a few examples are present. On the other hand, NCM and other few-shot methods are accurate in the low data regime but do not significantly improve when more data is added. Exemplar Tuning (ET) synthesizes these methods in order to initialize class representations accurately when learning new classes and to have the capacity to improve when presented with more data. We formulate each class representation (classifier), C_i, and class probability as the following:

$$C_i = \frac{1}{n} \sum_{x \in D_i} \frac{f(x;\theta)}{\|f(x;\theta)\|} + \mathbf{r}_i; \;\; p(y=i|x) = \frac{e^{C_i \cdot f(x;\theta)}}{\sum_{i \neq j} e^{C_j \cdot f(x;\theta)}} \tag{1}$$

where $f(x;\theta)$ is a parametrized neural network, $\mathbf{r}_i$ is a learnable vector, n is the number of class examples, and D_i are all examples in class i. Note that C_i is comparable in form to the i-th column vector in a linear classification layer.

In this formulation, the class centroid (the first term of C_i in Eq 1) provides an accurate initialization from which the residual term $\mathbf{r}_i$ can continue to learn. Thus ET is accurate for classes with few examples (where deep parametric models are inaccurate) and continues to improve for classes with more examples (where few-shot methods are lacking). In our experiments, we update the centroid after each sample with little additional compute and batch train the residual vector with cross-entropy loss according to the same schedule as fine-tuning (see Appendix C for implementation details).

In addition to having the highest overall and mean-class accuracy with reasonable compute, Exemplar Tuning affords two significant advantages beyond the quantitative metrics of NED. 1) ET has two frequencies of updates (fast instance-based and slow gradient-based) which allows the method to quickly adapt to distribution shifts while providing the capacity to improve over a long time horizon. 2) ET automatically balances between few-shot and many-shot performance, unlike Weight Imprinting which requires apriori knowledge of when to switch from NCM to fine-tuning.

Table 1: Performance of the suite of methods (outlined in Section 4) across accuracy and compute metrics on sequence 5. We present several variants of accuracy - Overall, Mean-per-class as well as accuracy bucketed into 4 categories: Novel-Head, Novel-Tail, Pretrain-Head and Pretrain-Tail (Pretrain refers to classes present in the ImageNet-1K dataset). Sup. refers to Supervised and MoCo refers to He et al. (2019). The best technique on every metric is in **bold**. Some methods could leverage caching of representations for efficiency, so, both GMACs are reported. GMACs do not include pretraining compute costs. See Table 4 for results with ResNet50 backbone.

Method	Pretrain Strategy	Novel - Head (>50)	Pretrain - Head (>50)	Novel - Tail (<50)	Pretrain - Tail (<50)	**Mean Per-Class**	**Overall**	GMACs↓ ($\times 10^6$)
				Backbone - Conv-4				
(a) Prototypical Networks	Meta	11.63	22.03	6.90	13.26	11.13	15.98	**0.06**
(b) MAML	Meta	2.86	2.02	0.15	0.10	1.10	3.64	0.06 / 2.20
				Backbone - ResNet18				
(c) Prototypical Networks	Meta	8.64	16.98	6.99	12.74	9.50	11.14	0.15
(d) Meta-Baseline	Sup./Meta	40.47	67.03	27.53	53.87	40.23	47.62	0.16 / 5.73
(e) OLTR	Sup.	40.83	40.00	17.27	13.85	27.77	45.06	0.16 / 6.39
(f) Fine-tune	Sup.	43.41	**77.29**	23.56	**58.77**	41.54	53.80	0.16 / 5.73
(g) Standard Training	Sup.	38.51	68.14	16.90	43.25	33.99	49.46	11.29
(h) NCM	Sup.	42.35	72.69	**31.72**	56.17	43.44	48.62	**0.15**
(i) LwF	Sup.	30.07	67.50	7.23	56.96	31.02	48.76	22.58 / 45.16
(n) Weight Imprinting	Sup.	40.32	67.46	15.35	34.18	32.69	48.51	0.16 / 5.73
(j) **Exemplar Tuning**	Sup.	**48.85**	75.70	27.93	45.73	**43.61**	**58.16**	0.16 / 5.73
(k) OLTR	MoCo	34.60	33.74	13.38	9.38	22.68	39.92	0.16 / 6.39
(l) Fine-tune	MoCo	14.49	27.59	0.10	4.96	8.91	26.86	0.16 / 5.73
(m) Standard Training	MoCo	26.63	45.02	9.63	20.54	21.12	35.60	11.29
(n) NCM	MoCo	19.24	31.12	14.40	21.95	18.99	22.90	0.15
(o) Weight Imprinting	MoCo	16.77	26.98	6.19	8.69	12.60	22.90	0.16 / 5.73
(p) **Exemplar Tuning**	MoCo	31.50	46.21	12.90	21.10	24.36	39.61	0.16 / 5.73

5 Experiments and Analysis

We evaluate representative methods from a range of subfields (few-shot, continual, self-supervised learning, and out-of-distribution detection) on NED. We present a broad set of practical observations and discuss novel findings that validate the need for NED and suggest future research directions. Table 1 displays a comprehensive set of metrics for the set of methods (outlined in Sec 4). Throughout this section, we will refer to rows of the table for specific analysis. All method discussions assume supervised pretraining unless otherwise stated.

Standard Training and Fine-Tuning Standard training (ST) (Table 1-f) and fine-tuning (FT) (Table 1-g) obtain an overall accuracy of 49.46% and 53.8% respectively. Interestingly, this gap suggests that pretraining on the larger, but less aligned ImageNet-1K provides better features for the target distribution than the streaming data itself.

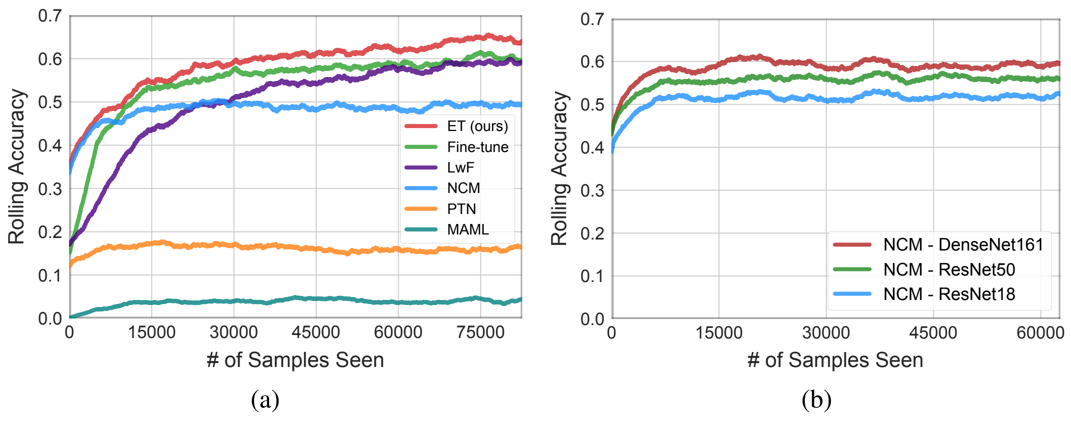


Figure 2: (a) Compares the rolling accuracy of various methods over the stream of data. Exemplar Tuning performs the best at all stages of the stream. (b) Compares the accuracy of NCM on novel classes across network architectures. Contrary to prevailing thought, we find that deeper networks overfit less to the pretrain classes.

Nearest Class Mean NCM (Table 1-h) excels on new and tail classes despite frozen features. In Figure 2-a we see that NCM learns quickly at the start of the stream, but the performance stagnates as more data is received. Although NCM sacrifices some performance in the standard data regime, it consumes ∼30 times less compute than fine-tuning.

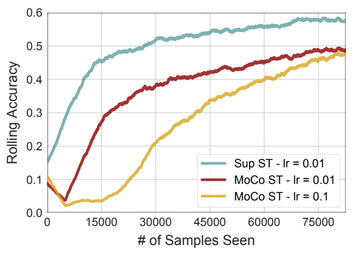
Exemplar Tuning ET (Table 1-j) has significantly higher overall and mean-class accuracy than other methods and uses similar compute as fine-tuning. Figure 2-a shows how ET quickly adapts to new classes and continues to learn in the standard data regime (high accuracy at the beginning and end of the stream). In addition to the advantages outlined in section 4, we observe that ET outperforms a simple combination of NCM + fine-tuning (Weight Imprinting) by ∼10%.

Network Capacity and Generalization Few-shot works indicate that smaller networks avoid overfitting to the classes they train on (Sun et al., 2019; Oreshkin et al., 2018; Snell et al., 2017; Finn et al., 2017). Hence, to avoid overfitting, few-shot methods such as Prototypical Networks and MAML have used 4-layer convolutional networks. In accordance, we observe that the 4-layer Prototypical Network (Table 1-a) does outperform its Resnet18 counterpart (Table 1-c). However, the opposite trend is seen for NCM. Table 4- in the appendix and Figure 2-b show that higher capacity networks with NCM actually generalize better to new classes. In other words, the features of larger networks overfit less to the training classes which contradicts prevailing thought. We argue that meta-training must be responsible for the overfitting we observed in higher capacity networks as the only difference between NCM and Prototypical Networks is the use of meta-training. This suggests the need to rethink the interaction between network size, training techniques, and generalization.

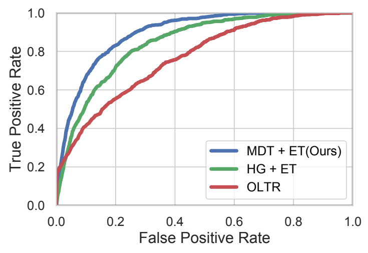
Few-shot Methods Few-shot methods are designed to excel in the low data regime; therefore one might expect Prototypical Networks and MAML to perform well in NED. However, we find that PTN and MAML (Table 1-a,b,c and Figure 2-a) fail to scale to the more difficult NED setting which has more classes and differing number of examples for each class. This suggests that the n-shot k-way setup is not a sufficient evaluation for real systems and frameworks such as NED are needed to gauge progress. As discussed in the previous section, simply scaling the architectures of few-shot methods that employ meta-training is not a sufficient solution as accuracy will decrease. Additionally, Meta-Baseline (Table 1-d) loses overall accuracy compared to standard training (Table 1-g) after meta-training for a small number of epochs. To build methods that work for few-shot and many-shot classes simultaneously we must rethink meta-training and the current few-shot approaches.

Self-supervised Representation Learning We observe unexpected behavior from MoCo (He et al., 2019) in the NED setting that contrasts results on other downstream tasks. Across a suite of methods, MoCo is significantly outperformed by supervised pretraining. For instance, Table 1-f vs Table 1-l shows a drastic 27% drop. Figure 3-a shows other unexpected behavior where MoCo decreases to almost 0% initially when standard training, then begins improving after 10K samples. We argue that this is related to learning a mixture of pretrain and new classes which is the primary difference between NED and previous benchmarks. The failure of MoCo to learn novel tail classes while fine-tuning (Table 1-l) further reinforces this hypothesis. We conjecture that this difficulty is induced by learning with a linear classifier because NCM with MoCo (Table 1-n) does not exhibit this behavior.

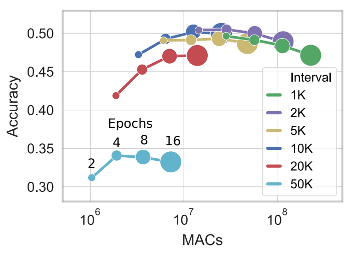
Unseen Class Detection We measure AUROC for detecting previously unseen classes throughout the sequence and present in Figure 3-b. HG baseline + ET, OLTR and MDT + ET achieve 0.84, 0.78 and 0.92 AUROC scores respectively. The superior performance of our simple baseline, MDT, suggests that distance in feature space is a promising direction for out-of-distribution detection. We compare MDT and HG baseline with other classifiers such as NCM and fine-tuning in Appendix H.



(a)



(b)



(c)

Figure 3: (a) Accuracy of standard training with MoCo & supervised pretraining. Surprisingly, MoCo accuracy decreases during the initial streaming phase. (b) ROC curves for unseen class detection. MDT outperforms all other OOD methods evaluated in NED. (c) Standard training accuracy curve for a range of training frequencies and epochs/training phase showing that over training can lead to lower accuracy. MACs $\propto$ total gradient updates.

Update Strategies We evaluate the accuracy and total compute cost of varying update frequencies and training epochs (Figure 3-c & 6). We conduct our experiments with fine-tuning (Figure 6) and standard training (Figure 3-c) on a ResNet18 model with supervised pretraining.

We find that training for too many total epochs (training frequency $\times$ epochs) with standard training (Figure 3-c) decreases the overall accuracy, though fine-tuning asymptotically improves (Figure 6). We hypothesize that the optimal amount of standard training balances the features learnt from ImageNet-1K with those from the smaller, imbalanced streaming data. We employ Learning without Forgetting (LwF) as a potential solution to maintain features from the pretrained initialization while learning from new data. We find that while LwF preserves accuracy on base classes (Table 1-i) better than standard training, it has lower overall accuracy. This problem supports the need for learning strategies that infer when and which parts of the models to train in contrast to fixed training phases and updating all parameters.

6 Conclusion

In this work we introduce NED, a new framework that 1) encourages the integration of solutions across many sub-fields including supervised classification, few-shot learning, continual learning, and efficient ML, 2) offers more flexibility for learners to specify various parameters of their learning procedure, such as when and how to train, 3) incorporates the total cost of updating and inference as the main anchoring constraint, and 4) can cope with the streaming, fluid and open nature of the real world. NED is designed to foster research in devising algorithms tailored toward building more pragmatic ML systems in the wild. Utilizing the NED framework we discover findings that would otherwise remain undetected by less realistic and more assumptive experimental setups which validates the need for pragmatic benchmarks like NED. In addition, we introduce two new approaches (Exemplar Tuning and Minimum Distance Thresholding) task that significantly outperform all other evaluated methods in NED. In particular, the effectiveness of ET suggests that semi-parametric deep learning methods have potential and should be explored further. We hope NED promotes more research at the cross-section of decision making and model training to provide more freedom for learners to decide on their own procedural parameters. In this paper, we study various methods and settings in the context of supervised image classification, one of the most explored problems in ML. While we do not make design decisions specific to image classification, incorporating other mainstream tasks into NED is an immediate next step. Throughout the experiments in this paper, we impose some restrictive assumptions, albeit only a few, on NED. Relaxing these assumptions in order to get NED even closer to the real world is another immediate step for future work. For example, we currently assume that NED has access to labels as the data streams in. One exciting future direction is to add the semi- and un-supervised settings to NED to alleviate labelled-data scarcity in the wild.

REFERENCES

Rahaf Aljundi, Marcus Rohrbach, and Tinne Tuytelaars. Selfless sequential learning. *arXiv preprint arXiv:1806.05421*, 2018.

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019.

Abhijit Bendale and Terrance Boult. Towards open world recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1893–1902, 2015.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541, 2006.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020a.

Yinbo Chen, Xiaolong Wang, Zhuang Liu, Huijuan Xu, and Trevor Darrell. A new meta-baseline for few-shot learning. *arXiv preprint arXiv:2003.04390*, 2020b.

Jaeik Cho, Taeshik Shon, Ken Choi, and Jongsub Moon. Dynamic learning model update of hybrid-classifiers for intrusion detection. *The Journal of Supercomputing*, 64(2):522–526, 2013.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *Proceedings of the International Conference on Machine Learning*, 2020.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.

Daniel Gordon, Kiana Ehsani, Dieter Fox, and Ali Farhadi. Watching the world go by: Representation learning from unlabeled videos. *arXiv preprint arXiv:2003.07990*, 2020.

Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5356–5364, 2019.

Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3018–3027, 2017.

James Harrison, Apoorva Sharma, Chelsea Finn, and Marco Pavone. Continuous meta-learning without tasks. *Advances in neural information processing systems*, 2019.

Jiangpeng He, Runyu Mao, Zeman Shao, and Fengqing Zhu. Incremental learning in online scenario. *arXiv preprint arXiv:2003.13191*, 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.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

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

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Citeseer*, 2009.

Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. Soft threshold weight reparameterization for learnable sparsity. In *Proceedings of the International Conference on Machine Learning*, 2020.

Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-scale long-tailed recognition in an open world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2537–2546, 2019.

Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 721–731, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

Ameya Prabhu, Philip H.S. Torr, and Puneet K. Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *Proceedings of the European Conference on Computer Vision*, 2020.

Senthil Purushwalkam and Abhinav Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. *arXiv preprint arXiv:2007.13916*, 2020.

Hang Qi, Matthew Brown, and David G Lowe. Low-shot learning with imprinted weights. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5822–5830, 2018.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pp. 525–542. Springer, 2016.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *International Conference on Learning Representations*, 2017.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing inference. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pp. 4077–4087, 2017.

Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 403–412, 2019.

Sebastian Thrun. Is learning the n-th thing any easier than learning the first? *Advances in neural information processing systems*, 1996.

Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? *arXiv preprint arXiv:2003.11539*, 2020.

Michael Tschannen, Olivier Bachem, and Mario Lucic. Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*, 2018.

Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8769–8778, 2018.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pp. 3630–3638, 2016.

Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*, 2019.

Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *International Conference on Learning Representations*, 2020.

Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. In *Advances in Neural Information Processing Systems*, 2020.

A Dataset Information

The five sequences we pair with NED are constructed from ImageNet-22K (Deng et al., 2009). Two sequences (1-2) are for validation, and three (3-5) are for testing. Each sequence contains 1,000 classes; 250 of which are in ImageNet-1K (Russakovsky et al., 2015) (pretrain classes) and 750 of which are only in ImageNet-22K (novel classes). For the test sequences, we randomly select the classes without replacement to ensure that the sequences do not overlap. The validation sequences share pretrain classes because there are not enough pretrain classes (1000) to partition among five sequences. We randomly distribute the number of images per class according to Zipf's law with $s = 1$ (Figure 4). For classes without enough images, we fit the Zipfian distribution as closely as possible which causes a slight variation in sequence statistics seen in Table 2.

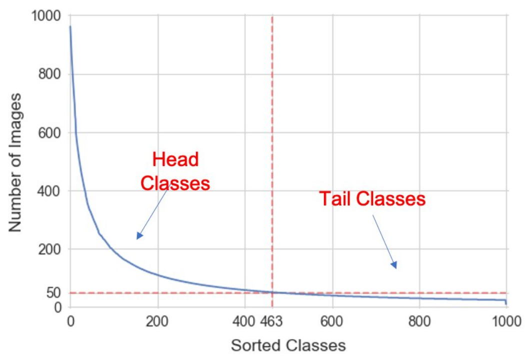


Figure 4: The distribution of samples over the classes for Sequences 1 - 5. Classes with less than 50 samples are considered in the tail and samples with greater than or equal to 50 samples are considered in the head for the purpose of reporting.

Table 2: Statistics for the sequences of images used in NED. Sequences 1-2 are for validation and Sequence 3-5 are for testing. The images from ImageNet-22k are approximately fit to a Zipfian distribution with 250 classes overlapping with ImageNet-1k and 750 new classes.

Sequence #	Number of Images	Min # of Class Images	Max # of Class Images
1	89030	1	961
2	87549	21	961
3	90133	14	961
4	86988	6	892
5	89921	10	961

B More Method Details

Nearest Class Mean Each class mean, m_i, is the average feature embedding of all examples in class i: $m_i = \sum_{x \in C_i} f_\phi(x)$; where C_i is the set of examples belong to class i and f_ϕ is the deep feature embedding of x. Class probabilities are the softmax of negative distances between x and class means:

$$P(y = i|\mathbf{x}) = \frac{e^{-d(m_i, f_\phi(\mathbf{x}))}}{\sum_{i'} e^{-d(m_{i'}, f_\phi(\mathbf{x}))}} \tag{2}$$

MAML The gradient update for MAML is: $\theta \leftarrow \theta - \beta \cdot \nabla_\theta \sum_{\mathcal{T}_i \sim p(T)} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta_i'}\right)$ where θ_i' are the parameters after making a gradient update given by: $\theta_i' \leftarrow \theta - \alpha \cdot \nabla_\theta \mathcal{L}_{\mathcal{T}_i}\left(f_\theta\right)$.

OLTR The network consist of two parts 1) A feature extractor consist of a ResNet backbone followed by a modulated attention and 2) A classifier and memory bank that are used to classify the output of the feature extractor. Training is done in 2 stages; In the first stage the feature extractor is trained. In the second stage the feature extractor and classifier are fine-tuned while samples are accumulated in the memory bank.

Weight Imprinting Weight Imprinting initializes the weights of the cosine classification layer, then performs fine-tuning using all of the data with a learnable temperature to scale the logits. Weight Imprinting can be thought of as NCM with cosine similarity as the metric for determining the closest neighbor, then performing fine-tuning. To use Weight Imprinting in a sequential setting, rather than a few-shot setting, we must decide when to begin fine-tuning. In the original formulation, fine-tuning was performed after the centroids were calculated using the entire data set, but in the sequential setting we do not have access to the entire data set until streaming ends. Therefore we choose to begin fine-tuning when the accuracy of fine-tuning exceeds NCM on validation data. In a real-world scenario it would be difficult to obtain such information, but we employ such a strategy to provide an upper-bound for the performance of Weight Imprinting in the sequential setting.

C Implementation Details

In this section, we discuss how methods are adapted with respect to NED. Some methods are intuitively applied with little modification, and some require interpretation for how they should be adapted.

Offline Training For all experiments (Table 1) that require offline training (fine-tuning, Weight Imprinting, standard training, ET and LwF), except OLTR, we train each model for 4 epochs every 5,000 samples observed. An epoch includes training over all previously seen data in the sequence. Experiments in Figure 6 show that training for 4 epochs every 5,000 samples balanced sufficient accuracy and reasonable computational cost. Fine-tuning experiments use a learning rate of 0.1 and standard training uses 0.01 for supervised pretraining. For MoCo pretraining fine-tuning uses a learning rate of 30 and standard training uses 0.01. All the experiments use the SGD+Momentum optimizer with a 0.9 momentum.

Instance-Based Updates All instance-based methods (NCM, ET, Weight Imprinting, Prototypical Networks) are updated after every sample as it takes no additional compute compared to batch updates.

Meta-Training For few-shot methods that leverage meta-training for pretraining, we used 5-shot 30-way except for MAML which we meta-trained with 5-shot 5-way due to computational costs. We choose to use 30-way as the computational graph is limited in the number of instances that it can store in memory and backpropagate to. We meta-train for 100 epochs with a learning rate of 0.01 and reduce it by 0.5 every 40 epochs.

Exemplar Tuning We initialize the residual vectors as zero. ET is trained according to the specifications of instance-based updates and offline training simultaneously.

Weight Imprinting For Weight Imprinting, we transition from NCM to fine-tuning after 10,000 samples as we observed that the accuracy of NCM saturated at this point in the validation sequence. We use a learning rate of 0.1 while fine-tuning.

Learning Without Forgetting We adapt Learning Without Forgetting to the NED task by freezing a copy of the model after pretraining which is used for knowledge distillation. Not all pretraining classes are seen during streaming so only softmax probabilities for classes seen during the stream are used in the cross-entropy between the soft labels and predictions. We use a temperature of 2 to smooth the probabilities in accordance with (Li & Hoiem, 2017). Training is done according to the specifications given in the offline training portion of this section. **OLTR** For OLTR (Liu et al., 2019), we update the memory and train the model for 4 epochs every 200 samples for the first 10,000 samples, then train 4 epochs every 5,000 samples with a 0.1 learning rate for classifier parameters and 0.01 for feature extraction parameters which is in accordance with the specifications of the original work.

Pretraining We use the PyTorch (Paszke et al., 2019) ResNet18 and ResNet50 models pretrained on supervised ImageNet-1K. We use the models from Gordon et al. (2020) for the MoCo (He et al., 2019) self-supervised ImageNet-1K pretrained models. MoCo-ResNet18 and MoCo-ResNet50 get top-1 validation accuracy of 44.7% and 65.2% respectively and were trained for 200 epochs. For fine-tuning and ET with MoCo, we report the results with a learning rate of 30 which is suggested by the original work when learning on frozen features. All other learning rates with MoCo are the same as with supervised.

D Training Depth for Fine Tuning

We explored how training depth affects the accuracy of a model on new, old, common, and rare classes. For this set of experiments, we vary the number of trained layers when fine-tuning for 4 epochs every 5,000 samples on ResNet18 with a learning rate of 0.01 on Sequence 2 (validation). The results are reported in Table 3. We found that training more layers leads to greater accuracy on new classes and lower accuracy on pretrain classes. However, we observed that the number of fine-tuning layers did not significantly affect overall accuracy so for our results on the test sequences (3-5) we only report fine-tuning of one layer (Table 1).

Table 3: The results for fine-tuning various numbers of layers with a learning rate of .01 on Sequence 2. Training more layers generally results in higher accuracy on novel classes, but lower accuracy on pretrain classes. The trade-off between novel and pretrain accuracy balances out so the overall accuracy is largely unaffected by the depth of training.

# of Layers	Novel-Head (>50)	Pretrain-Head (>50)	Novel-Tail (<50)	Pretrain-Tail (<50)	Mean Per-Class	Overall
1	**41.32**	**80.96**	17.13	66.52	39.19	56.87
2	41.55	80.79	17.40	**67.03**	39.43	56.79
3	45.82	78.59	19.08	59.52	**40.73**	**57.23**
4	**46.96**	75.44	19.87	53.97	40.39	57.04
5	46.76	75.72	**19.97**	54.04	40.41	57.04

E Results for ResNet50 backbone on Sequence 5

We report all performance metrics for sequence 5 in Table 4 for ResNet50 backbone. These results corroborate the findings of Table 1 which uses ResNet18 backbone.

Table 4: Continuation of Table 1 results on sequence 5 with ResNet50 backbone.

Method	Pretrain Strategy	Novel - Head (>50)	Pretrain - Head (>50)	Novel - Tail (<50)	Pretrain - Tail (<50)	**Mean Per-Class**	**Overall**	GMACs↓ ($\times 10^6$)
				Backbone - ResNet50				
(a) Fine-tune	MoCo	14.42	43.61	0.22	13.40	11.85	31.35	0.36 / 13.03
(b) Fine-tune	Sup.	47.78	82.06	27.53	**66.42**	46.24	57.95	0.36 / 13.03
(c) Standard Training	MoCo	26.82	42.12	10.50	21.08	21.32	35.44	38.36
(d) Standard Training	Sup.	43.89	74.50	21.54	50.69	39.48	54.10	38.36
(e) NCM	MoCo	30.58	55.01	24.10	45.37	32.75	36.14	**0.35**
(f) NCM	Sup.	45.58	78.01	**35.94**	62.90	47.75	52.19	**0.35**
(g) LwF	Sup.	21.52	49.17	38.74	05.49	20.69	30.57	38.36/76.72
(h) **Exemplar Tuning**	MoCo	28.86	54.03	7.02	20.82	21.89	40.13	0.36 / 13.03
(i) **Exemplar Tuning**	Sup.	**52.95**	**82.27**	28.13	57.15	**48.02**	**62.41**	0.36 / 13.03

F Results For Other Sequences

We report the mean and standard deviation for all performance metrics across test sequences 3-5 in Table 5. Note that the standard deviation is relatively low so the methods are consistent across the randomized sequences.

G Prototypical Network Experiments

We benchmarked our implementation of Prototypical Networks on few-shot baselines to verify that it is correct. We ran experiments for training on both MiniImageNet and regular ImageNet-1k and tested our implementation on the MiniImageNet test set and NED (Sequence 2). We found comparable results to those reported by the original Prototypical Networks paper (Snell et al., 2017) (Table 6).

Table 5: Averaged results for all methods evaluated on Sequences 3-5. See Table 1 for the computational cost (GMACs) for each method and more information about each column.

Method	Pretrain	Novel - Head (>50)	Pretrain - Head (>50)	Novel - Tail (<50)	Pretrain - Tail (>50)	**Mean Per-Class**	**Overall**
				Backbone - Conv-4			
Prototype Networks	Meta	5.02±0.05	9.71±0.11	0.64±0.01	1.27±0.04	3.25±0.03	7.82±0.09
MAML	Meta	2.93±0.01	2.02±0.02	0.15±0.01	0.1±0.01	1.11±0.02	3.64±0.06
				Backbone - ResNet18			
Prototype Networks	Meta	8.72±0.09	16.84±0.14	7.06±0.03	12.98±0.04	9.46±0.08	11.19±0.12
Meta-Baseline	Sup./Meta	41.73±0.57	66.54±2.37	27.54±1.13	53.69±0.97	39.32±0.71	47.74±0.63
Fine-tune	Moco	5.31±0.24	45.95±1.27	0.03±0	26.23±0.88	10.64±0.23	18.52±0.98
Fine-tune	Sup.	43.2±0.65	74.55±2.53	22.79±1.21	59.63±1.02	40.9±0.73	53.06±0.65
Standard Training	Moco	26.9±0.27	42.39±3.04	9.1±0.74	21.11±0.51	20.76±0.32	34.85±0.75
Standard Training	Sup.	38.82±0.49	65.88±2.32	16.15±0.83	44.3±0.91	33.63±0.38	48.81±0.57
NCM	Moco	19.31±0.06	30.02±1.69	14.21±0.46	22.06±0.52	18.86±0.13	22.14±1.24
NCM	Sup.	41.68±0.65	70.05±2.29	31.24±0.86	57.23±0.97	42.87±0.62	47.89±0.76
OLTR	MoCo	41.47±0.03	31.48±0.01	17.48±0.01	9.81±0.01	22.03±0	38.33±0.01
OLTR	Sup.	51.19±0.37	37.02±0.51	24.14±0.14	13.77±0.24	27.6±0.28	44.46±0.44
Exemplar Tuning	Moco	32.57±1.54	43.48±0.4	6.39±0.49	12.81±0.12	18.46±0.35	39.25±1.20
Exemplar Tuning	Sup.	46.36±2.31	69.34±0.53	23.48±1.23	45.82±0.32	42.93±0.17	57.56±0.56
				Backbone - ResNet50			
Fine-tune	Moco	45.95±0.26	5.31±0.32	26.23±0.07	0.03±1.74	10.64±0.21	18.52±1.02
Fine-tune	Sup.	47.59±0.65	80.14±1.71	26.69±0.97	66.92±1.4	45.62±0.6	57.48±0.47
Standard Training	Moco	43.93±0.73	71.72±3.18	20.84±0.92	51.43±0.68	38.94±0.9	53.45±1.73
Standard Training	Sup.	47.59±0.45	80.14±2.59	26.69±0.79	66.92±1.91	45.62±0.47	57.48±0.56
NCM	Moco	30.15±0.48	53.84±1.05	23.99±0.53	44.11±1.11	32.27±0.92	35.45±0.61
NCM	Sup.	45.46±0.95	76.55±1.77	35.47±0.82	65.62±1.57	47.77±0.65	52.22±0.55
Exemplar Tuning	Moco	28.46±3.04	40.42±1.33	7.57±2.15	14.36±4.14	19.54±2.63	32.07±2.37
Exemplar Tuning	Sup.	49.24±1.55	75.78±1.84	26.67±2.17	55.63±2.31	44.15±1.44	62.35±1.02

Table 6: Our implementation of Prototypical Networks on MiniImageNet & NED. ◇ Results from Snell et al. (2017).

Method	Backbone	Train Set	MiniImageNet 5 Way - 5 Shot	NED
Prototypical Networks	Conv - 4	MiniImageNet	69.2	14.36
Prototypical Networks	Conv - 4	ImageNet (Train)	42.7	15.98
Prototypical Networks◇	Conv - 4	MiniImageNet	68.2	-

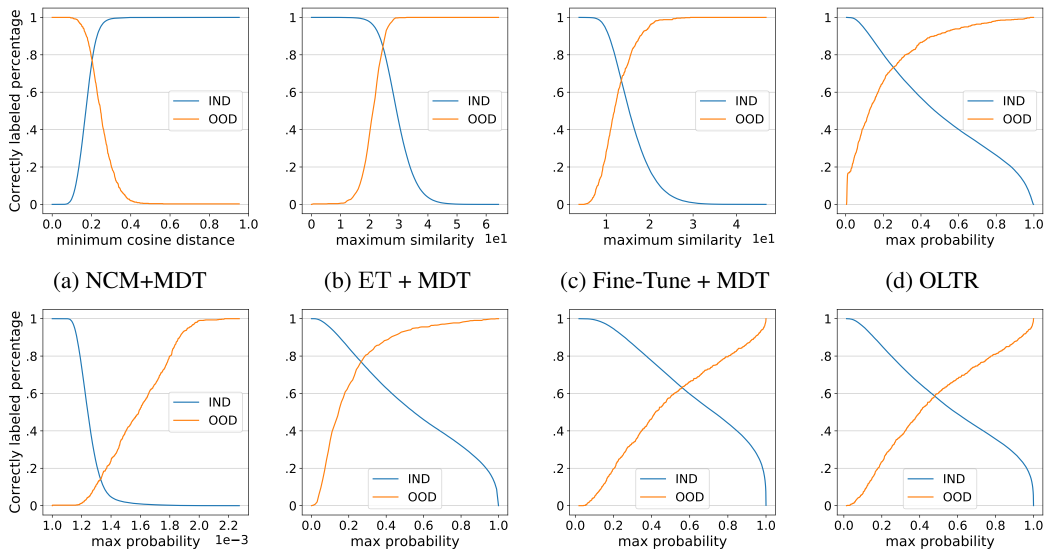
H Out-of-Distribution Ablation

In this section we report AUROC and F1 for MDT and softmax for all baselines. In section 5 we only included OLTR, MDT with Exemplar Tuning, and ET with maximum softmax (Hendrycks Baseline). Additionally, we visualize the accuracy curves for in-distribution and out-of-distribution samples as the rejection threshold vary (Figure 5). All the OOD experiments presented in Figure 5 and Table 7 were run using ResNet18. Minimum Distance Thresholding (MDT) threshold distances but also similarity metrics can be used. MDT generally works better than maximum softmax when applied to most methods.

The results of NCM and Exemplar Tuning using softmax and dot product similarity in comparison to OLTR are shown in table 7. The F1-scores are low due to the large imbalance between positive and negative classes. There are 750 unseen class datapoints vs $\sim$ 90000 negative datapoints. Table 7 shows that cosine similarity (MDT) is better than softmax or the OLTR model for most methods.

Table 7: The out-of-distribution performance for each method on sequence 5. We report the AUROC and the F1 score achieved by choosing the best possible threshold value.

Metric	NCM +Softmax	NCM +MDT	Exemplar Tuning +Softmax	Exemplar Tuning +MDT	Standard Training +Softmax	Standard Training +MDT	Fine-Tune +Softmax	Fine-Tune +MDT	OLTR
AUROC	0.07	0.85	0.84	0.92	0.59	0.53	0.68	0.72	0.78
F1	0.01	0.20	0.10	0.20	0.03	0.02	0.06	0.10	0.27



(a) NCM+MDT (b) ET + MDT (c) Fine-Tune + MDT (d) OLTR

(e) NCM+Softmax (f) ET + Softmax (g) Fine-Tune + Softmax (h) Full train + Softmax

Figure 5: The accuracy for the in-distribution (IND) and out-of-distribution (OOD) samples as the threshold for considering a sample out-of-distribution varies. The horizontal axis is the threshold value, and the vertical axis is the accuracy. Intersection of the IND and OOD curves at a higher accuracy generally indicates better out-of-distribution detection for a given method.

Table 8: Comparison of Weight Imprinting and Exemplar Tuning with different classifiers and initial temperatures. Exemplar Tuning with a linear layer performs significantly better than all other variants.

Method	Pretrain	Backbone	Novel - Head (>50)	Pretrain - Head (>50)	Novel - Tail (<50)	Pretrain - Tail (>50)	Mean Per-Class	Overall
Weight Imprinting (s = 1)	Sup	R18	36.58	63.39	9.32	21.80	26.85	46.35
Weight Imprinting (s = 2)	Sup	R18	36.58	63.39	9.32	21.80	26.85	46.35
Weight Imprinting (s = 4)	Sup	R18	40.32	67.46	15.35	34.18	32.69	48.51
Weight Imprinting (s = 8)	Sup	R18	31.18	32.66	34.77	28.94	32.56	46.67
Exemplar Tuning (Cosine)	Sup	R18	33.90	18.22	4.84	1.88	11.72	31.81
Exemplar Tuning (Euclidean)	Sup	R18	43.40	66.32	21.66	42.06	37.19	51.62
Exemplar Tuning (Linear)	Sup	R18	**48.85**	**75.70**	**23.93**	**45.73**	**43.61**	**58.16**

I WEIGHT IMPRINTING AND EXEMPLAR TUNING ABLATIONS

In Table 8, we ablate over various softmax temperature initializations with Weight Imprinting. We learn the temperature as described in (Qi et al., 2018), but find that initial value affects performance. We report the best results in the main paper. We also ablate over the similarity metrics use in ET. We find that the dot product (linear) is the best measure of similarity for ET.

J UPDATE STRATEGIES

Figure 6 has the accuracy vs MACs trade-off for fine-tuning across various update strategies.

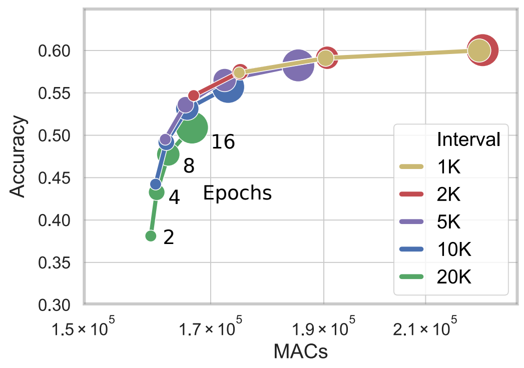


Figure 6: The plot compares the accuracy and MACs for various update strategies when fine-tuning.