# Are We Overfitting to Experimental Setups in Recognition?

Matthew Wallingford[†], Aditya Kusupati[*] [†], Keivan Alizadeh-Vahid[*] [†],
Aaron Walsman[†], Aniruddha Kembhavi[†][‡] and Ali Farhadi[†]
[†]University of Washington, [‡]Allen Institute for Artificial Intelligence
https://raivn.cs.washington.edu/projects/FLUID/

## Abstract

*Enabling robust intelligence in the real-world entails systems that offer continuous inference while learning from varying amounts of data and supervision. The machine learning community has organically broken down this challenging goal into manageable sub-tasks such as supervised, few-shot, and continual learning. In light of substantial progress on each sub-task, we pose the question, "How well does this progress translate to more practical scenarios?" To investigate this question, we construct a new framework,* FLUID, *which removes certain assumptions made by current experimental setups while integrating these sub-tasks via the following design choices – consuming sequential data, allowing for flexible training phases, being compute aware, and working in an open-world setting. Evaluating a broad set of methods on* FLUID *leads to new insights including strong evidence that methods are overfitting to their experimental setup. For example, we find that representative few-shot methods are substantially worse than simple baselines, self-supervised representations from MoCo fail to learn new classes when the downstream task contains a mix of new and old classes, and pretraining largely mitigates the problem of catastrophic forgetting. Finally, we propose two new simple methods which outperform all other evaluated methods which further questions our progress towards robust, real-world systems.*

## 1. Introduction

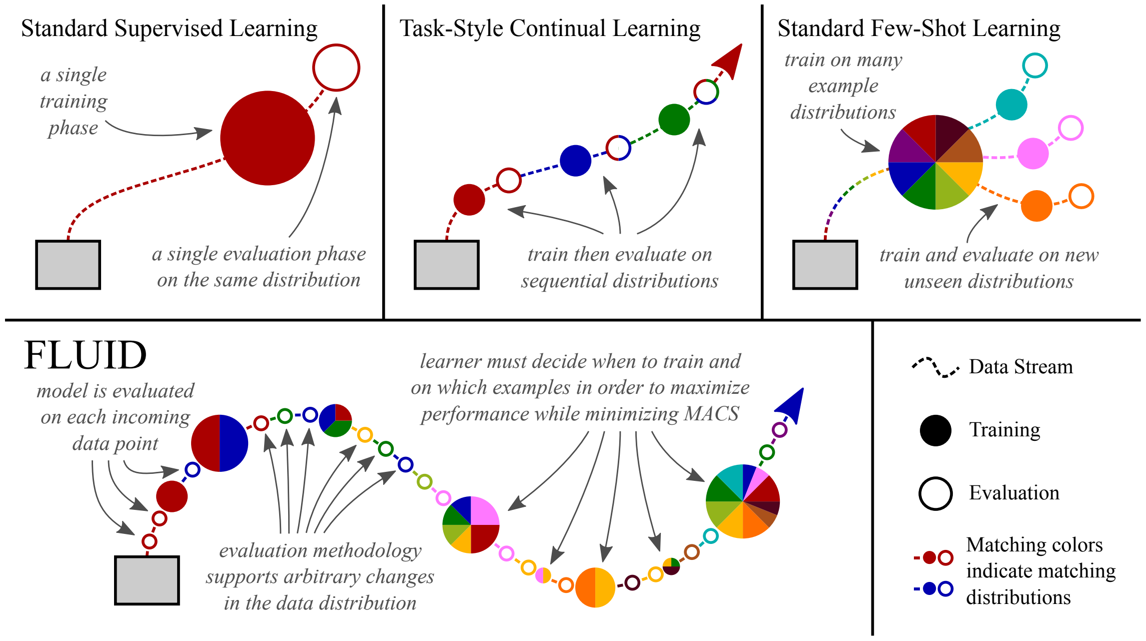
Researchers have organically broken down the ambitious task of enabling intelligence in real-world settings into smaller and better-defined sub-tasks such as classification for uniformly distributed data, few-shot learning, continual learning, etc., and worked towards developing effective models for each sub-task. The last decade has witnessed staggering progress on each of these sub-tasks. Therefore we ask, "*How well does benchmark performance translate to more practical settings?*"

[*]Equal contribution

Recent works have raised concerns about the progress in few-shot and continual learning by presenting simple baselines that are comparable to state of the art in each field [32, 6, 48, 44]. We approach these recent concerns from an alternate perspective and contend that the benchmarks and their experimental design, not the methods, are the underlying issue. We hypothesize that the implicit assumptions and overly constrained nature of these tasks have encouraged methods to overfit to their exact experimental setups. Using a new evaluative framework, we identify several implicit assumptions in few-shot and continual learning and evaluate how representative methods perform when these assumptions do not hold. We show that in a more general setting with fewer assumptions representative methods lose most of their utility and in some cases are detrimental.

In contrast to the assumptions of previous frameworks we posit the following as reasonable elements of a practical framework: (1) *Sequential Data* – Many frameworks assume that training and testing sets are separated. Often in practical settings, data encountered during inference can be later used for training. (2) *Flexible Training Phases* – Most current frameworks dictate when a model will train and perform inference. However, in many scenarios, apriori knowledge of when inference will be required or the best time to train cannot be assumed. (3) *Compute Aware* – Practical settings often have computational constraints not only for inference but also for training. If the previous assumption about how and when to train models is removed, then the compute of unconstrained training strategies must be accounted for. (4) *Open-world* – Most supervised frameworks assume that new samples must come from classes seen during training. In many practical scenarios, samples can come from new classes which the model must adapt to. (5) *X-shot* – Current experimental settings make overly strong assumptions about the quantity of data for each class (few-shot, many-shot, etc.). It is reasonable to expect a model to learn without knowing the characteristics of the data distribution apriori.

With these elements in consideration, we present the FLUID (**FL**exible seq**U**ent**I**al **D**ata) framework. FLUID incorporates the core objectives of continual, few-shot, out-



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
FLUID
model is evaluated on each incoming data point
learner must decide when to train and on which examples in order to maximize performance while minimizing MACS
evaluation methodology supports arbitrary changes in the data distribution
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 FLUID (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.

of-distribution detection, and self-supervised learning addresses the overly strong implicit assumptions of previous frameworks and is a step towards learning in more practical settings akin to real-world.

Conducting experiments with FLUID, we find new insights about few-shot, continual, and self-supervised learning that were overlooked in previous benchmarks. We propose two new methods that outperform all other evaluated methods and conduct preliminary experiments to demonstrate the potential of FLUID for exploring new research directions such as flexible training phases.

**We make the following contributions:**

1. Empirical evidence that representative methods in continual, and few-shot learning are overfitting to their experimental setups.
2. A new framework, FLUID, that carefully integrates the objectives of few-shot, continual, and self-supervised learning, addresses the issues in previous experimental setups, and presents new research challenges.
3. New insights from experiments with FLUID such as higher capacity networks actually generalize better to novel classes, MoCo representations struggle to learn new classes when learning from a mix of new and old classes, and good pretraining largely mitigates the problem of catastrophic forgetting.
4. Two simple new methods Exemplar Tuning and Minimum Distance Thresholding which outperform other evaluated methods in the new framework and provides a basis for new methods.

## 2. Related Work and Framework Discussion

We discuss the key aspects of FLUID and assumptions of previous frameworks in the context of related work. The logistical details of FLUID are provided in section 3.

**Sequential Data** New data is an inevitable consequence of our dynamic world and learning over time is a long-standing challenge [43]. In recent years, continual learning (CL) has made notable progress on the problem of learning in a sequential fashion [27, 22, 37, 2, 1, 38]. 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 [27, 22, 37]. Recent variants have proposed a task-free setting where the data distribution changes without the model's knowledge [15, 38, 16, 52].

There are two assumptions in CL setups which we remove in FLUID. 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 a learner 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. In other words, few-shot learning is a natural consequence of learning from sequential data. 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.

Some continual learning formulations include a memory cache which models can store images, typically between 0 - 1000, from previous tasks. We argue that setting a specific memory size, particularly this small, is prone to methods overfitting. Researchers should account for memory, but the framework does not explicitly restrict memory during streaming. Note that for a fair comparison of CL methods we use no memory caching for the Nearest Class Mean (NCM) baseline and complete caching for CL methods.

Lastly, data stream classification [11, 49, 41] made some strides towards handling streaming data by identifying the challenges and proposing new metrics. Through FLUID we also provide the infrastructure to evaluate the models for data stream classification.

**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 [7]. By removing the assumption that training and testing phases are fixed and known in advance, FLUID provides a benchmark for tackling the relatively unexplored problem of learning when to train.

**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. [32] showed that current CL frameworks do not measure total compute and therefore a naive but compute-hungry strategy can drastically outperform state of the art methods. Previous works have primarily focused on efficient inference [35, 20, 24] and some on training costs [9]. In FLUID we measure the total compute for both learning and inference over the sequence.

**Open-world** Practical scenarios entail inferring in an open world - where the classes and number of classes are unknown to the learner. Few-shot, continual, and traditional supervised learning setups assume that test samples can only be from training classes. Previous works explored static open-world recognition [29, 3] and the related problem of out-of-distribution detection [19]. FLUID 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 [39, 28] has been the primary focus of supervised learning, although recently few-shot learning has become a popular subfield [36, 14, 30, 42].

While few-shot learning is a step towards more generally applicable ML methods, the framework has assumptions that arguably do not hold in most settings. 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. Most settings have a mix of classes from both the high and low data regime. Expecting all classes to have less than 10 samples is unreasonable.

FLUID 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 representative few-shot methods indicate that such methods are over-fitting to the few-shot framework which validates the need for a more general framework such as FLUID.

## 3. Framework Details

The FLUID procedure and setup make as few assumptions and ad-hoc design decisions as possible while incorporating the key aspects outlined in section 2.

**Algorithm 1** FLUID Procedure

```
Input: Task $\mathcal{T}$
Input: ML sys.: (pretrained) model $\mathbf{f}$, update strategy $\mathbf{S}$
Output: Evaluations: $E$, Operation Counter: $C$
 1: function FLUID($\mathcal{T}$, ($\mathbf{f}$, $\mathbf{S}$))
 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:         $C \mathrel{+}= A + B$
13:     end while

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

**Procedure** FLUID provides a stream of data to a learning

Table 1: The evaluation metrics used in the FLUID framework to capture the performance and capabilities of various algorithms.

| **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:<br>1) *Pretraining-Head*: Classes with $> 50$ samples that were included in pretraining.<br>2) *Pretraining-Tail*: Classes with $\leq 50$ samples that were included in pretraining.<br>3) *Novel-Head*: Classes with $> 50$ samples that were not included in pretraining.<br>4) *Novel-Tail*: Classes with $\leq 50$ samples that were not included in pretraining. |

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. 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 FLUID.

**Data** In this paper, we evaluate methods under the FLUID framework using a subset of ImageNet-22K [8]. Traditionally, few-shot learning used datasets like Omniglot [25] & MiniImagenet [47] and continual learning focused on MNIST [26] & CIFAR [23]. Some recent continual learning works have used Split-ImageNet [50]. The aforementioned datasets are mostly small-scale and have very few classes. We evaluate on the ImageNet-22K dataset to present new challenges to existing models. Recently, the INaturalist [46, 51] and LVIS [13] datasets have advocated for heavy-tailed distributions. We follow suit and draw our sequences from a heavy-tailed distribution.

The dataset consists of a pretraining dataset and 5 different sequences of images for streaming (3 test and 2 validation sequences). For pretraining we use the standard ImageNet-1K [39]. 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 test sequence contains images from 1000 different 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. Each 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 2, we show results obtained for sequence 5, and the Appendix F shows results across all test sequences. More comprehensive statistics on the data and sequences can also be found in the Appendix A.

**Pretraining** Supervised pretraining [18] on large annotated datasets like ImageNet facilitates the transfer of learnt representations to help data-scarce downstream tasks. Unsupervised learning methods like autoencoders [45] and more recent self-supervision methods [21, 33, 12] like Momentum Contrast (MoCo) [17] and SimCLR [5] have begun to produce representations as rich as that of supervised learning and achieve similar accuracy on various downstream tasks.

Before beginning 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 when training on new and pretrain classes and meta-training causes larger networks to overfit in a way that supervised training does not.

**Evaluation metrics** Table 1 defines the evaluation metrics in FLUID to gauge the performance of the algorithms.

## 4. Baselines and Methods

We summarize the baselines, other methods, and our proposed methods, Exemplar Tuning and Minimum Distance Thresholding. 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 ablate over the number of layers trained during fine-tuning in Appendix D.

**Nearest Class Mean (NCM)** Recently, multiple works [44, 48] have found that Nearest Class Mean (NCM) is comparable to state-of-the-art few-shot methods [42, 30]. 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.

**Few-shot Methods** We evaluate four representative methods: MAML [10], Prototypical Networks (PTN) [40], Weight Imprinting [34] & Meta-Baseline [6].

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

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 FLUID by pretraining the model according to the objective in Appendix B 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. For further analysis of Weight Imprinting and comparison to Exemplar Tuning see Appendix I.

Meta-Baseline is the same as NCM in implementation except that a phase of meta-training is done after regular softmax cross-entropy pretraining.

**Continual Learning (CL) Methods** We evaluate Learning without Forgetting (LwF) [27] and Elastic Weight Consolidation (EWC) [22] to observe whether continual learning techniques can improve performance in FLUID.

LwF leverages knowledge distillation [4] to retain accuracy on previous training data without storing it. EWC enables CL in a supervised learning context by penalizing the total distance moved by the parameters from the optimal model of previous tasks weighted by the corresponding Fisher information. Unlike LwF, EWC requires stored data, typically the validation set, from the previous tasks. In FLUID, we use LwF and EWC to retain performance on pretrain classes. For further details see Appendix C.

**Out-of-Distribution (OOD) Methods** We evaluate two methods proposed by Hendrycks & Gimpel [19] (HG) and OLTR [29] 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 $i$th 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 FLUID.

**Exemplar Tuning (ET)** We present a new 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).

Note that we compare ET to initializing a cosine classifier with class centroids and fine-tuning (Weight Imprinting). We find that Exemplar Tuning outperforms Weight Imprinting and affords two significant advantages besides better accuracy. 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 & many-shot performance, unlike Weight Imprinting which requires apriori knowledge of when to switch from centroid-based initialization to fine-tuning.

## 5. Experiments and Analysis

We evaluate representative methods from few-shot, continual, self-supervised learning, and out-of-distribution detection in the FLUID framework. First, we analyze the few-shot and continual learning results. We discuss how they indicate overfitting to the experimental setup as well as new insights about meta-training and catastrophic forgetting. Next, we compare the results of our proposed methods (ET, MDT) in the context of FLUID. Finally, we discuss results about self-supervised learning and flexible training phases that show the potential of FLUID as a framework for the future develop-

Table 2: 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. [17]. 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 5 in Appendix E 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 | 6.79 | 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) Weight Imprinting | Sup. | 40.32 | 67.46 | 15.35 | 34.18 | 32.69 | 48.51 | 0.16 / 5.73 |
| (f) LwF | Sup. | 30.07 | 67.50 | 7.23 | 56.96 | 31.02 | 48.76 | 22.58 / 45.16 |
| (g) EWC | Sup. | 39.03 | 70.84 | 16.59 | 47.18 | 34.89 | 50.39 | 12.29 |
| (h) OLTR | Sup. | 40.83 | 40.00 | 17.27 | 13.85 | 27.77 | 45.06 | 0.16 / 6.39 |
| (i) Fine-tune | Sup. | 43.41 | **77.29** | 23.56 | **58.77** | 41.54 | 53.80 | 0.16 / 5.73 |
| (j) Standard Training | Sup. | 38.51 | 68.14 | 16.90 | 43.25 | 33.99 | 49.46 | 11.29 |
| (k) NCM | Sup. | 42.35 | 72.69 | **31.72** | 56.17 | 43.44 | 48.62 | **0.15** |
| (l) **Exemplar Tuning** | Sup. | **48.85** | 75.70 | 27.93 | 45.73 | **43.61** | **58.16** | 0.16 / 5.73 |
| (m) Weight Imprinting | MoCo | 16.77 | 26.98 | 6.19 | 8.69 | 12.60 | 22.90 | 0.16 / 5.73 |
| (n) OLTR | MoCo | 34.60 | 33.74 | 13.38 | 9.38 | 22.68 | 39.92 | 0.16 / 6.39 |
| (o) Fine-tune | MoCo | 14.49 | 27.59 | 0.10 | 4.96 | 8.91 | 26.86 | 0.16 / 5.73 |
| (p) Standard Training | MoCo | 26.63 | 45.02 | 9.63 | 20.54 | 21.12 | 35.60 | 11.29 |
| (q) NCM | MoCo | 19.24 | 31.12 | 14.40 | 21.95 | 18.99 | 22.90 | 0.15 |
| (r) **Exemplar Tuning** | MoCo | 31.50 | 46.21 | 12.90 | 21.10 | 24.36 | 39.61 | 0.16 / 5.73 |

ment of practical methods. Table 2 displays a comprehensive set of metrics for the set of methods outlined in section 4 on sequence 5. Throughout this section, we will refer to rows of the table for specific analysis.

**Few-shot Analysis** To observe whether few-shot methods are overfitting to the few-shot formulation we evaluate Prototypical Networks (PTN), Model Agnostic Meta-Learning (MAML), Weight Imprinting, and Meta-Baseline. We compare to the baselines of Nearest Class Mean (NCM), fine-tuning, and standard training. We choose to evaluate each method for the following reasons:

- Prototypical Networks is a prominent metric learning method that is the basis for many other few-shot methods. Additionally PTN differs from NCM only in the use of meta-training so we can isolate the effect of meta-training.
- MAML is a prominent optimization-based meta-learning approach which many other methods build up on.
- Meta-Baseline is similar to NCM except that additional meta-training is performed during the pretrain phase. It is a simple method that is competitive with state of the art in few-shot learning.
- Weight Imprinting is a simple combination of NCM and fine-tuning and does not use meta-training. This makes for a natural comparison to both NCM and fine-tuning individually along with our proposed method, ET.

We point out that the procedure of FLUID is the same as few-shot with two caveats. The test distribution can contain base classes and the shot and way values are changing. The pretraining phase of FLUID is identical to meta-training where a model trains with a set of base classes. The streaming phase of FLUID can be viewed as a series of n-shot k-way evaluations where n and k steadily increase from zero to N-1 where N is the number of total samples in the stream for a given class.

We find that PTN and MAML (Table 2-a,b,c and Figure 2-a) breakdown in the more practical setting with over 30% lower in overall accuracy than the NCM baseline (Table 2-k). One could argue that PTN and MAML may scale to the larger scale setting by increasing model capacity. However, few-shot works indicate that deeper networks decrease generalization to novel classes [42, 30, 40, 10]. We verify that the 4-layer convnet PTN (Table 2-a) does outperform the ResNet18 PTN in overall and novel class accuracy. This indicates that PTN and MAML have overfitted to the specific shot and way values in the few-shot formulation given that they cannot scale to more data and are more than 30% lower

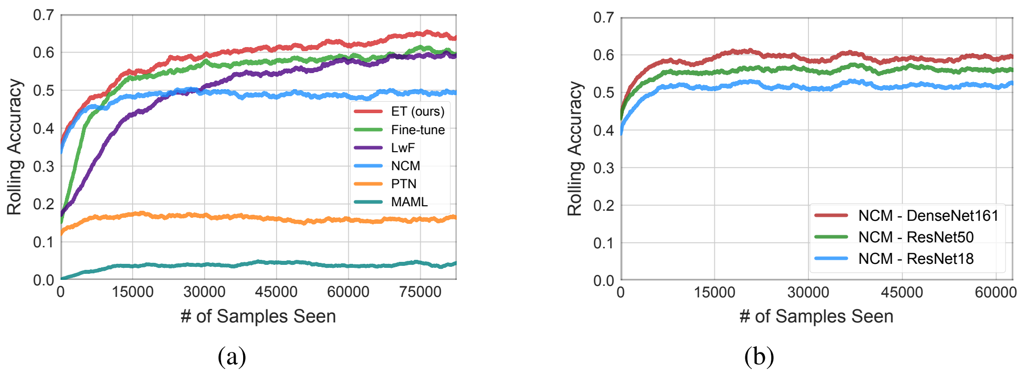


Figure 2: (a) Compares the rolling accuracy of various methods over the stream of data. Exemplar Tuning performs the best at all stages of the stream. (b) Compares the accuracy of NCM on novel classes across network architectures. Contrary to prevailing thought, we find that deeper networks overfit less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

in accuracy than baselines.

For more evidence on how the few-shot setup has affected the development of methods, we analyze the connection between network depth and generalization to novel classes. The prevailing thought in few-shot literature has been that smaller networks overfit less to base classes, and therefore methods use shallow networks or develop techniques to constrain deeper ones. We find evidence to the contrary, that deeper networks generalize better to novel classes when trained normally (see Figure 2-b). Given that NCM and PTN differ only in the use of meta-training, we conclude that meta-training must be responsible for the lack of generalization in deeper networks. This evidence is further reinforced by the fact that meta-baseline performs worse than NCM with the inclusion of meta-training (Table 2-d,k). We do not discount meta-training, but rather point out that it fails to scale to more practical settings in its current form which is overlooked in the current few-shot setup.

Overall, none of the few-shot methods we evaluate (Table 2-(a-e, m)) perform as well as fine-tuning or NCM which indicates the need for new benchmarks and careful experimental design.

**Continual Learning Analysis** We evaluate Learning without Forgetting (LwF) and Elastic Weight Consolidation (EWC). For analysis, we compare to the baselines: NCM, standard training, fine-tuning. We evaluate LwF and EWC because they are well-known CL methods that are general enough to be reasonably adapted to the FLUID setup.

We find that the most significant difference between FLUID and typical continual learning formulations is the inclusion of pretraining. We contend this is a reasonable assumption as real-world vision systems have access to open-source datasets such as ImageNet. From the experiments, we find that generally good features learned from pretraining mitigate the problem of catastrophic forgetting.

To arrive at the above conclusion, we first note that standard training (propagating gradients through all layers) has noticeably lower overall and new class accuracy than fine-tuning (Table 2-i,j). This result indicates that the features trained on the larger ImageNet-1K are better than those trained on the target distribution even for classes that were not seen in pretraining. The effectiveness of generally good features is relevant because CL methods focus on“remembering” old tasks while adapting to new ones. However, if features trained on a diverse, balanced dataset generalize well even to novel classes then significant retraining of the network to fit new tasks might not be necessary.

Comparing NCM and fine-tuning to LwF and EWC illustrates this point. NCM uses no memory caching or replay buffer and has frozen features while LwF and EWC cache all stream images and freely train their features. NCM has similar overall accuracy and higher pretraining class accuracy compared to LwF and EWC. NCM learns new classes as well as these CL methods with no possibility of forgetting. Furthermore, fine-tuning which also uses frozen features outperforms both methods on novel and old classes and has $\sim 4\%$ higher overall accuracy. We conclude that simple baselines can generalize to new classes and prevent catastrophic forgetting as well or better than CL methods with the reasonable inclusion of pretraining. This finding once again reinforces the argument that the experimental design of setups is tied to the practical utility of the methods they lead to. Note that for scenarios in which the pretraining distribution is radically different from that of the target this conclusion may not hold, such as for permuted MNIST.

**Exemplar Tuning** We find that ET (Table 2-l) has significantly higher overall and mean-class accuracy than all other evaluated 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 start and end of the stream). Finally, we show that ET outperforms simple NCM + fine-tuning (Weight Imprinting) by $\sim$10%, in addition to the practical advantages outlined in section 4.

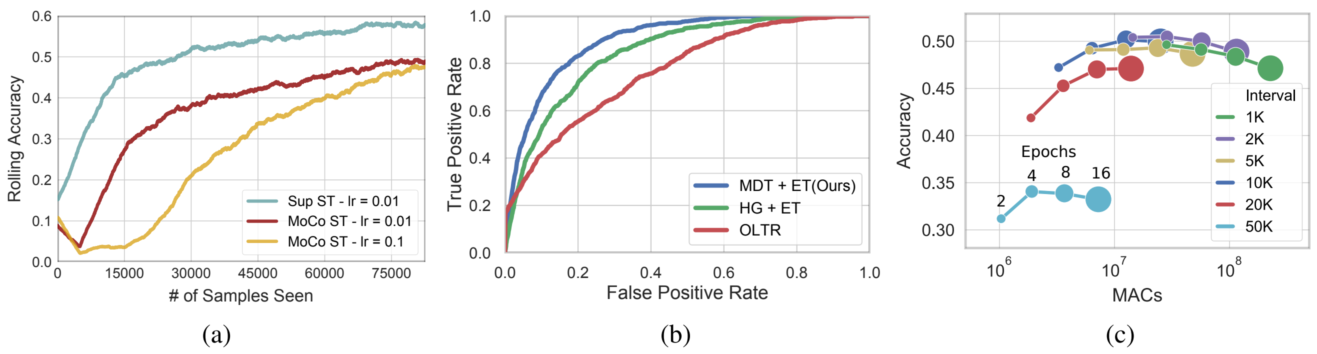


Figure 3: (a) Accuracy of standard training with MoCo & supervised pretraining. Surprisingly, MoCo accuracy decreases during the initial streaming phase. (b) ROC curves for unseen class detection. MDT outperforms all other OOD methods evaluated in FLUID. (c) Standard training accuracy curve for a range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

**Unseen Class Detection and MDT** 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 performance of Minimum Distance Thresholding (MDT) indicates that distance in feature space is a promising direction for computationally inexpensive out-of-distribution detection. We compare MDT and HG baseline with other classifiers such as NCM and fine-tuning in Appendix H.

**Self-supervised Learning** We observe unexpected behavior from MoCo [17] representations in the FLUID setting that contrasts with results on previous downstream tasks. Across a suite of methods, MoCo pretraining is significantly outperformed by supervised pretraining. For instance, Table 2-i vs Table 2-o shows a drastic 27% drop. Figure 3-a shows other unexpected behavior where MoCo accuracy 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 FLUID and previous downstream tasks. The failure of MoCo representations to learn novel tail classes while fine-tuning (Table 2-o) further reinforces this hypothesis. We conjecture that this difficulty is induced by learning with a linear classifier as NCM with MoCo pretraining (Table 2-q) does not exhibit this behavior.

**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 in Appendix J) and standard training (Figure 3-c) on ResNet18 model with supervised pretraining.

We show 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 in Appendix J). We hypothesize that the optimal amount of training balances the features learnt from ImageNet-1K with those from the smaller, imbalanced streaming data. This aligns with the continual learning experiments that indicated general features trained on more data outperformed specialized features. These initial experiments are intended to illustrate the new problems that FLUID presents for future research. The results indicate that there is significant room for improvement in both efficiency and accuracy with new strategies for training networks under streaming conditions which we leave for future work.

## 6. Conclusion

In this work, we show that representative methods from few-shot and continual learning have overfit to implicit assumptions in their experimental setups. We introduce FLUID, a new framework that 1) removes most unrealistic implicit assumptions, 2) provides new insights about meta-training, generalization in relation to network capacity, catastrophic forgetting, and self-supervised representations of MoCo, 3) enables integration of solutions across many sub-fields including supervised, few-shot, continual, and efficient learning and 4) offers more flexibility for learners to specify various parameters of their learning procedure, such as when and how to train. In addition, we introduce two new methods (Exemplar Tuning & Minimum Distance Thresholding) that outperform all other evaluated methods in FLUID and provide a basis for further innovations.

Throughout this paper, we studied various methods and settings in the context of supervised image classification, a highly explored problem in ML. While we do not make design decisions specific to image classification, incorporating other mainstream tasks into FLUID is an immediate next step. Across the experiments in this paper, we impose some restrictive assumptions, albeit only a few, on FLUID. For example, we currently assume that FLUID has access to labels as the data streams in. One exciting future direction is to add the semi- and un-supervised aspects to FLUID. Relaxing these remaining assumptions to get FLUID even closer to the real world is an interesting direction for future work.

## Acknowledgements

This work is in part supported by NSF IIS 1652052, IIS 17303166, DARPA N66001-19-2-4031, 67102239 and gifts from Allen Institute for Artificial Intelligence. We thank Jae Sung Park and Mitchell Wortsman for insightful discussions and Daniel Gordon for the pretrained MoCo weights.

## References

[1] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019. 2

[2] Rahaf Aljundi, Marcus Rohrbach, and Tinne Tuytelaars. Selfless sequential learning. *arXiv preprint arXiv:1806.05421*, 2018. 2

[3] Abhijit Bendale and Terrance Boult. Towards open world recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1893–1902, 2015. 3

[4] 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*, pages 535–541, 2006. 5

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 4

[6] Yinbo Chen, Xiaolong Wang, Zhuang Liu, Huijuan Xu, and Trevor Darrell. A new meta-baseline for few-shot learning. *arXiv preprint arXiv:2003.04390*, 2020. 1, 5

[7] 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. 3

[8] 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*, pages 248–255. Ieee, 2009. 4, 12

[9] 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. 3

[10] 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*, pages 1126–1135. JMLR. org, 2017. 5, 6

[11] Heitor Murilo Gomes, Jesse Read, Albert Bifet, Jean Paul Barddal, and João Gama. Machine learning for streaming data: state of the art, challenges, and opportunities. *ACM SIGKDD Explorations Newsletter*, 2019. 3

[12] 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. 4, 13

[13] 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*, pages 5356–5364, 2019. 4

[14] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3018–3027, 2017. 3

[15] James Harrison, Apoorva Sharma, Chelsea Finn, and Marco Pavone. Continuous meta-learning without tasks. *Advances in neural information processing systems*, 2019. 2

[16] Jiangpeng He, Runyu Mao, Zeman Shao, and Fengqing Zhu. Incremental learning in online scenario. *arXiv preprint arXiv:2003.13191*, 2020. 2

[17] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019. 4, 6, 8, 13

[18] 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*, pages 770–778, 2016. 4

[19] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016. 3, 5

[20] 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. 3

[21] 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. 4

[22] 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*, 2017. 2, 5, 13

[23] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Citeseer*, 2009. 4

[24] 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. 3

[25] 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. 4

[26] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998. 4

[27] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017. 2, 5, 13

[28] 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*, pages 740–755. Springer, 2014. 3

[29] 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*, pages 2537–2546, 2019. 3, 5, 13

[30] 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*, pages 721–731, 2018. 3, 5, 6

[31] 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*, pages 8024–8035, 2019. 13

[32] 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. 1, 3

[33] Senthil Purushwalkam and Abhinav Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. *arXiv preprint arXiv:2007.13916*, 2020. 4

[34] 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*, pages 5822–5830, 2018. 5, 15

[35] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016. 3

[36] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *International Conference on Learning Representations*, 2017. 3

[37] 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*, pages 2001–2010, 2017. 2

[38] 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. 2

[39] 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. 3, 4, 12

[40] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017. 5, 6, 14, 15

[41] Jerzy Stefanowski and Dariusz Brzezinski. Stream classification., 2017. 3

[42] 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*, pages 403–412, 2019. 3, 5, 6

[43] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? *Advances in neural information processing systems*, 1996. 2

[44] 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. 1, 5

[45] Michael Tschannen, Olivier Bachem, and Mario Lucic. Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*, 2018. 4

[46] 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*, pages 8769–8778, 2018. 4

[47] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016. 4

[48] 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. 1, 5

[49] Kapil K Wankhade, Snehlata S Dongre, and Kalpana C Jondhale. Data stream classification: a review. 3

[50] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *International Conference on Learning Representations*, 2020. 4

[51] Davis Wertheimer and Bharath Hariharan. Few-shot learning with localization in realistic settings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6558–6567, 2019. 4

[52] 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. 2

## A. Dataset Information

The five sequences we pair with FLUID are constructed from ImageNet-22K [8]. 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 [39] (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 3.

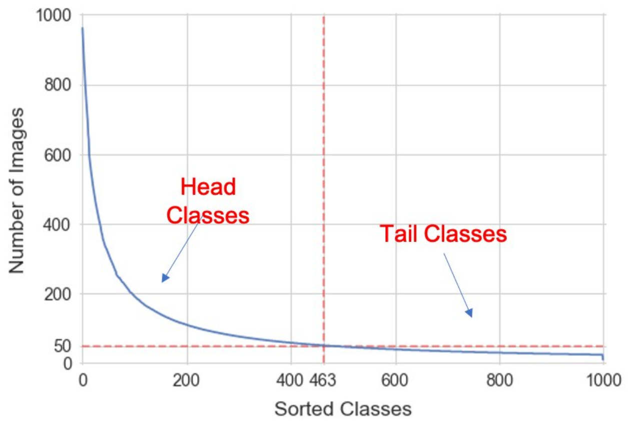


Figure 4: The distribution of samples over the classes for Sequences 1 - 5. Classes with less than 50 samples are considered in the tail and samples with greater than or equal to 50 samples are considered in the head for the purpose of reporting.

Table 3: Statistics for the sequences of images used in FLUID. 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 $f_\phi$ 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 $\theta_i'$ 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 FLUID. Some methods are intuitively applied with little modification, and some require interpretation for how they should be adapted.

**Offline Training** For all experiments (Table 2) 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 FLUID 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 [27]. Training is done according to the specifications given in the offline training portion of this section.

**Elastic Weight Consolidation** We adapt Elastic Weight Consolidation [22] to the FLUID task by freezing a copy of the model after pretraining as the optimla model of the pretrain task. We then use the validation set of ImageNet-1K corresponding to the 250 classes being used for the computation of Fisher information per-parameter. For every training step in FLUID, a penalty is added based on the distance moved by the parameters from the base model weighted by the Fisher information. The Fisher information is calculated at the start of every flexible train step to mitigate catastrophic forgetting. Training is done according to the specifications given in the offline training portion of this section. Depending on how frequently the Fisher information is computed, the compute associated increases over the standard training costs.

**OLTR** For OLTR [29], 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 [31] ResNet18 and ResNet50 models pretrained on supervised ImageNet-1K. We use the models from Gordon et al. [12] for the MoCo [17] 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 4. 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 2).

Table 4: 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 5 for ResNet50 backbone. These results corroborate the findings of Table 2 which uses ResNet18 backbone.

## F. Results For Other Sequences

We report the mean and standard deviation for all performance metrics across test sequences 3-5 in Table 6. Note that the standard deviation is relatively low so the methods are consistent across the randomized sequences.

Table 5: Continuation of Table 2 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 | 5.49 | 38.74 | 20.69 | 30.57 | 38.36/76.72 |
| (h) EWC | Sup. | 43.84 | 76.03 | 21.22 | 53.64 | 39.89 | 54.59 | 40.36 |
| (i) **Exemplar Tuning** | MoCo | 28.86 | 54.03 | 7.02 | 20.82 | 21.89 | 40.13 | 0.36 / 13.03 |
| (j) **Exemplar Tuning** | Sup. | **52.95** | **82.27** | 28.13 | 57.15 | **48.02** | **62.41** | 0.36 / 13.03 |

Table 6: Averaged results for all methods evaluated on Sequences 3-5. See Table 2 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 |

## 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 FLUID (Sequence 2). We found comparable results to those reported by the original Prototypical Networks paper [40] (Table 7).

Table 7: Our implementation of Prototypical Networks on MiniImageNet & FLUID. $^\diamond$ Results from Snell et al. [40].

| Method | Backbone | Train Set | MiniImageNet 5 Way - 5 Shot | FLUID |
|---|---|---|---|---|
| Prototypical Networks | Conv - 4 | MiniImageNet | 69.2 | 14.36 |
| Prototypical Networks | Conv - 4 | ImageNet (Train) | 42.7 | 15.98 |
| Prototypical Networks$^\diamond$ | 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 8 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 8. 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 8 shows that cosine similarity (MDT) is better than softmax or the OLTR model for most methods.

## I. Weight Imprinting and Exemplar Tuning Ablations

In Table 9, we ablate over various softmax temperature initializations with Weight Imprinting. We learn the temperature as described in [34], 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.

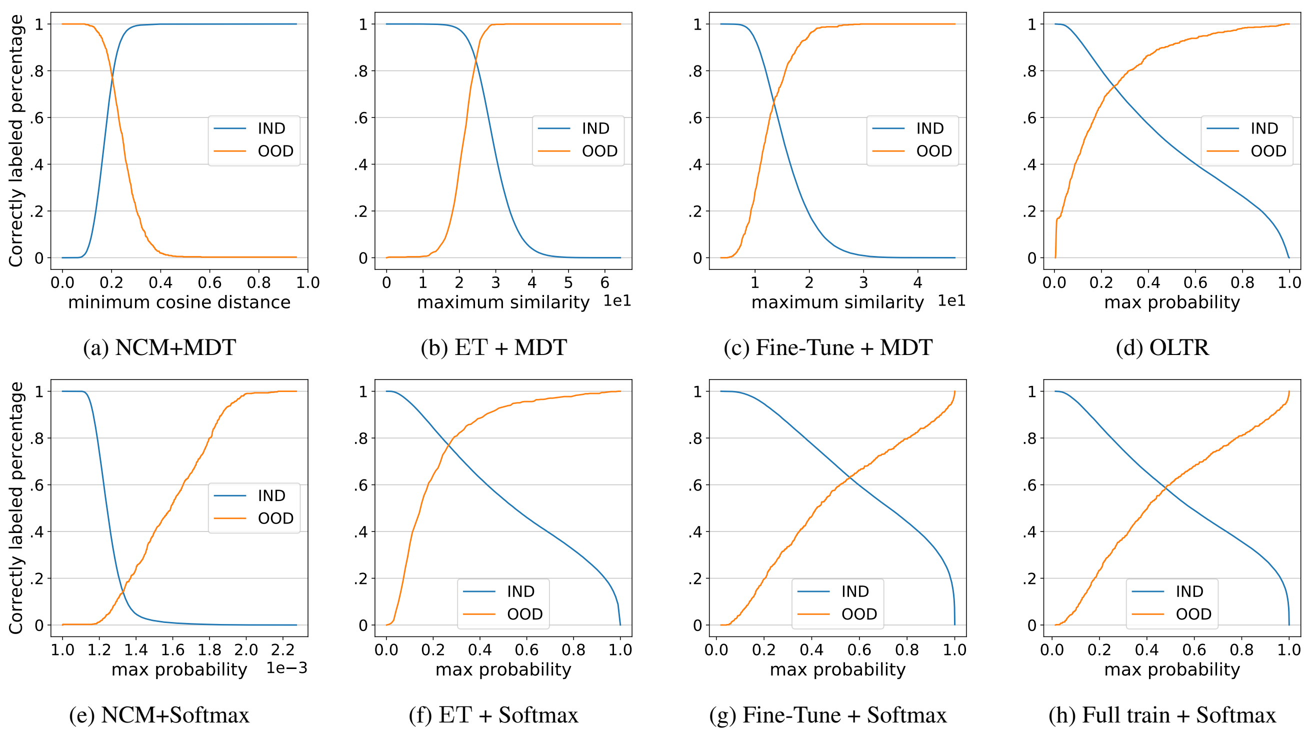


Figure 5: The accuracy for the in-distribution (IND) and out-of-distribution (OOD) samples as the threshold for considering a sample out-of-distribution varies. The horizontal axis is the threshold value, and the vertical axis is the accuracy. Intersection of the IND and OOD curves at a higher accuracy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

Table 8: 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 |

Table 9: 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** |

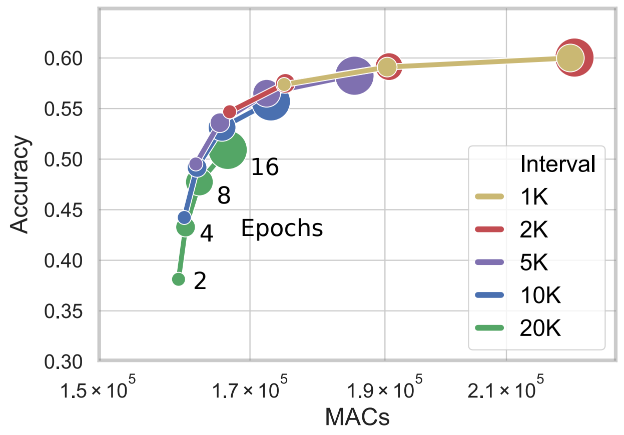


Figure 6: The plot compares the accuracy and MACs for various update strategies when fine-tuning.