Federated Reconnaissance: Efficient, Distributed, Class-Incremental Learning

Sean M. Hendryx *
School of Information
University of Arizona
seanmhendryx@arizona.edu

Dharma Raj KC
Department of Computer Science
University of Arizona
kcdharma@email.arizona.edu

Bradley Walls
Areté Associates
bwalls@arete.com

Clayton T. Morrison
School of Information
University of Arizona
claytonm@arizona.edu

Abstract

We describe federated reconnaissance, a class of learning problems in which distributed clients learn new concepts independently and communicate that knowledge efficiently. In particular, we propose an evaluation framework and methodological baseline for a system in which each client is expected to learn a growing set of classes and communicate knowledge of those classes efficiently with other clients, such that, after knowledge merging, the clients should be able to accurately discriminate between classes in the superset of classes observed by the set of clients. We compare a range of learning algorithms for this problem and find that prototypical networks are a strong approach in that they are robust to catastrophic forgetting while incorporating new information efficiently. Furthermore, we show that the online averaging of prototype vectors is effective for client model merging and requires only a small amount of communication overhead, memory, and update time per class with no gradient-based learning or hyperparameter tuning. Additionally, to put our results in context, we find that a simple, prototypical network with four convolutional layers significantly outperforms complex, state of the art continual learning algorithms, increasing the accuracy by over 22% (absolute) after learning 600 Omniglot classes and over 33% (absolute) after learning 20 mini-ImageNet classes incrementally. These results have important implications for federated reconnaissance and continual learning more generally by demonstrating that communicating feature vectors is an efficient, robust, and effective means for distributed, continual learning.

1 Introduction

In this work, we present *federated reconnaissance*, a new a class of learning problems in which distributed models should be able to learn new concepts independently and share that knowledge efficiently. Typically in federated learning, a single static set of classes is learned by each client [McMahan et al., 2017]. In contrast, federated reconnaissance requires that each client can individually learn a growing set of classes and communicate knowledge of previously observed and new classes efficiently with other clients. This communication about learned classes permits *merging* the knowledge from the clients; the resulting merged model is then expected to support the superset of the

*Work partially completed at Stanford University.

1st NeurIPS Workshop on New Frontiers in Federated Learning (NFFL 2021), Virtual Meeting.

classes each client has been exposed to. The merged model can then be deployed back out to the clients for further learning. In practice, a client in a distributed system may only see a small number of examples for a new class. This problem therefore stands at the intersection of large bodies of work on continual, meta-, and federated learning. Examples of this problem include mobile phone applications in which the service should be able to both learn to identify instances of a new class added by a user and transfer that classification ability to other client devices. In this paper, we outline related work, formalize the federated reconnaissance problem statement, introduce a benchmark by adapting mini-ImageNet [Vinyals et al., 2016], compare a range of learning algorithms and neural network architectures, and find that a simple algorithm adapting prototypical networks [Snell et al., 2017] is a strong baseline for both single client continual learning and federated reconnaissance. Our code and pretrained models are available at: `https://github.com/ml4ai/fed-recon`.

1.1 Prior Work

Continual learning of new concepts is an open and long-standing problem in machine learning and artificial intelligence with no semblance of a unified solution [Thrun and Mitchell, 1995, Lopez-Paz and Ranzato, 2017, Shin et al., 2017, Zenke et al., 2017, van de Ven and Tolias, 2019, Farajtabar et al., 2020]. While deep neural networks have proven to be incredibly effective in a wide range of tasks, the available methods for continuously integrating new information whilst remembering previously learned concepts suffer from being either compute inefficient (in the case of algorithms that retrain on a cache of examples [Rebuffi et al., 2017] or on generated examples [van de Ven et al., 2020]) or lacking in expressivity and accuracy (in the case of regularization-based methods [Kirkpatrick et al., 2017]). Much earlier work on continual learning (see van de Ven and Tolias [2019] and their citations) focused on learning classes or tasks sequentially from scratch. While this is an interesting problem setting for studying neural memory, it can be impractical and unnecessary for deployment to production systems. In more recent work on continual learning, the authors in Javed and White [2019], Prabhu et al. [2018], Beaulieu et al. [2020] change the continual learning problem setup by assuming access to a set of pretraining data, often referred to as a set of meta-training tasks, and find that such pretraining can benefit later continual learning. Access to a pretraining dataset is a reasonable assumption for production systems and enables the development of algorithms that can first learn invariances that can later be exploited when learning new classes online. In this work, we assume access to a set of pretraining data and explore algorithms that allow for the efficient and accurate learning of new classes sequentially.

In contrast to the common variants of continual learning, federated learning iteratively trains a common model under the direction of a central server on data that is decentralized across many devices, such as mobile phones. Federated learning can reduce communication overhead and privacy concerns by removing the need to send the raw source data back to a server for traditional, centralized machine learning [McMahan et al., 2017, Kairouz et al., 2019]. The goal in federated learning has traditionally been to learn a shared parameterization from decentralized data, but not necessarily to learn *new* concepts or classes online on different clients while preserving that discriminative information when communicating between client and the server. Recent work [Yoon et al., 2020] discusses federated continual learning yet assumes that each client learns distinct tasks. While they address federated continual learning directly, the authors do not consider the direct sharing of knowledge of classes. Instead they assume that each client is learning its own task and they focus on reducing interference when transferring the network's parameters, but not on merging explicit knowledge of classes that have been seen by the clients. In many cases, it would instead be useful if the server model could learn a single task with a growing set of classes by incorporating knowledge of classes learned on client devices. Such a unified, class-incremental learning model could be valuable to users of mobile devices, intelligence operations, robotics, or any situation in which new concepts should be learned and shared between distributed clients when efficient communication, privacy, and/or fast learning are paramount. Because this work builds on the motivation of federated learning but with the explicit goal of ascertaining knowledge of new concepts that can be efficiently communicated and reused, we call this problem federated reconnaissance.

1.2 Contributions

Federated reconnaissance poses a unique set of challenges. An effective federated reconnaissance system must tackle efficient in situ learning of new classes and knowledge-preserving transfer. To these ends, we systematically study different approaches to solving this problem including

running stochastic gradient descent (SGD) on new data as it appears as an empirical lower bound, an iCaRL [Rebuffi et al., 2017] algorithm adapted for federated reconnaissance, an extension of prototypical networks for distributed, continual learning, and, finally, SGD on the joint distribution of all training data from all clients as an empirical upper bound.

We posit that, when a pretraining dataset is available, prototypical networks [Snell et al., 2017] are a strong baseline for federated reconnaissance due to:

1. The ability to compress concepts into relatively small vectors known as prototypes, enabling efficient communication,
2. Robustness to catastrophic forgetting when learning on non-IID data[2], and
3. Enabling fast knowledge transfer as no gradient-based learning or hyperparameter tuning are required during model merging.

To test this claim, we present two simple algorithms for using prototypical networks for federated reconnaissance and evaluate them on the federated reconnaissance mini-ImageNet benchmark, showing that they outperform the lower bound and iCaRL models handily in both accuracy and computational complexity. We go on further to present pretraining methods that increase the accuracy of prototypical networks on the federated reconnaissance mini-ImageNet benchmark. Additionally, to put federated prototypical networks into the context of previous work, we show that they substantially outperform recent state of the art works on few-shot continual learning from Javed and White [2019], Beaulieu et al. [2020]. It is also worth noting that prototypical networks have stronger privacy protection than existing class-incremental replay-based learning algorithms such as iCaRL since the transfer of raw examples is avoided and as long as a minimum set of examples per class are averaged on each client. We leave the empirical and theoretical work of differential privacy for federated prototypical networks to future work.

2 Federated Reconnaissance Problem Statement

2.1 Desiderata

Federated reconnaissance requires continual learning on each client device, efficient communication, and knowledge merging. Inspired by applications to learning new classes across a large number of distributed client devices, we define the following desiderata of a federated reconnaissance learning system:

1. Each client model should be able to learn new classes in situ from only a few examples and be able to improve accuracy as more examples become available. [3]
2. After learning new classes, each model should not forget previously seen classes. I.e., the model should not suffer from catastrophic forgetting.
3. To reduce communication costs and to enable distributed learning when bandwidth is limited, a federated reconnaissance system should be able to compress information before transfer.
4. Finally, to avoid costly retraining on all data on the central server each time a new class is encountered by a client, the federated reconnaissance system should be able to merge knowledge of new classes learned by distributed client models quickly.

The specific requirements of a real world implementation of federated reconnaissance will of course determine the details and relative importance of each desideratum.

2.2 Problem Definition

Formally, federated reconnaissance consists of a set of clients $\mathbb{C} := \{c_i \mid i \in 1...C\}$ that each have an exposure history to a growing set of classes $\mathbb{M}_{i,t} := \{p(y = j|x) \mid j \in 1...M_i\}$ where C indicates

[2] In distributed online learning, the data is not guaranteed to be IID over time nor across clients since each client may observe a local distribution of potentially correlated examples [Zhang et al., 2021].

[3] In this work, we assume an in situ source of supervision and identification of new classes. We follow and reproduce the single client continual learning benchmarks of [Javed and White, 2019, Beaulieu et al., 2020] in which the model only observes a small number (e.g. ≤ 30) of examples to learn from at meta-test time.

the total number of clients, M_i indicates the number of classes that client C_i should be able to discriminate between, and a class is represented as a probability $p(y = j|x)$ of label j given an input x. The central server is tasked with merging the clients' knowledge of the superset of classes $\mathbb{M}_t = \bigcup_{i=1}^{C} \mathbb{M}_{i,t}$ and deploying an updated model that supports $\mathbb{M}_t$ back out to $\mathbb{C}$. A client C_i can be exposed to a new class by training directly on a set of labeled examples $\{(x, y) \mid (x, y) \in X_j \times Y_j\}$ or via communication of compressed knowledge that allows the client to approximate $p(y = j|x)$.

An effective federated reconnaissance learning *system* entails accurate prediction of $p(\hat{y} = j|x)$ in expectation over clients in $\mathbb{C}$ regardless of whether or not each individual client learned class j directly from labeled examples or vicariously via communication of compressed knowledge of j from another client. This brings us to the distributed objective function of the federated reconnaissance learning system at any point in time, which is the average loss across clients:

$$\mathcal{L}_t = \frac{1}{|\mathbb{C}|} \sum_{i=1}^{|\mathbb{C}|} \frac{1}{J_i} \sum_{j=1}^{J_i} \frac{1}{K_{i,j}} \sum_{k=1}^{K_{i,j}} H(\hat{y}_{i,j,k}, y_{i,j,k}) \tag{1}$$

where J_i is the total number of classes that client C_i has seen in its exposure history, $K_{i,j}$ is the number of examples on client C_i for class j, H is the cross entropy between the predicted class $\hat{y}_{i,j,k}$ and the labeled class $y_{i,j,k}$ for example k. This loss describes the expected loss over distributed clients as illustrated as Eval. 1 in Figure 2. To simplify notation, we assume a fixed number of clients throughout deployment, though the extension to a variable number of clients over time is straightforward.

For concise terminology, we define a *mission* as an iteration of sending clients out, having them collect and learn from data in situ and in parallel, and then finally communicating their findings back to a central server. At each mission t a few-shot dataset $\mathcal{D}_t$ of examples from a set of classes is produced by the environment. Federated reconnaissance comprises both problems in which clients begin learning from scratch without any knowledge of $p(X, Y)$ and those in which data from a subset of *base* classes $\mathbb{B}$ is available for pretraining. The set $\mathbb{B}$ is similar to the meta-training set in meta-learning [Finn et al., 2017], though unlike the typical setup in few-shot low-way meta-learning, we now expect the clients to learn a growing superset of classes which includes both the *base* classes and *field* classes which are learned after a centralized pretraining procedure on $\mathbb{B}$. Access to a dataset representing $\mathbb{B}$ tends to be a reasonable assumption in practice as some number of pretraining classes can usually be measured before a federated reconnaissance system is deployed.

We define the expected loss directly after model merging by taking the expectation over classes in $\mathbb{M}_t$:

$$\mathcal{L}_t = \frac{1}{|\mathbb{M}_t|} \sum_{j=1}^{|\mathbb{M}_t|} \frac{1}{K_{i,j}} \sum_{k=1}^{K_{i,j}} H(\hat{y}_{i,j,k}, y_{i,j,k}) \tag{2}$$

This loss can be discretized into an accuracy metric and its evaluation is illustrated in Eval 2 in Figure 3.

$$acc_t = \frac{1}{|\mathbb{M}_t|} \sum_{j=1}^{|\mathbb{M}_t|} \frac{1}{K_{i,j}} \sum_{k=1}^{K_{i,j}} [\hat{y}_{i,j,k} = y_{i,j,k}] \tag{3}$$

At each time step t, each client model is first presented with a set of labeled examples of new classes and then evaluated on held out examples from the superset of the new training classes and all classes in it's exposure history. This evaluation step is represented as evaluation diamond 0 in Figure 2. Following in situ, on-client learning, the clients send information back to the server, either communicating information of new classes or updating the information of previously seen classes and the server merges the information from multiple clients together. After this model merging step, we evaluate accuracy on the superset of base and field classes that all clients have seen thus far on held out examples (evaluation diamond 2 in Figure 2). As shown in Figure 2, the process of local client learning and communicating knowledge back to the server is repeated iteratively. Therefore, we want to minimize the expected loss in equation 1 over some horizon of missions $\{t \in \mathbb{N} | t \leq T\}$:

$$\min \mathop{\mathbb{E}}_{t \in 1 \ldots T} [\mathcal{L}_t] \tag{4}$$

or alternatively, after the agents have learned for some fixed number of missions:

$$\min \mathcal{L}_{t=T} \tag{5}$$

For simplicity of exposition and to highlight the unique challenges of continual distributed learning, our reported tabular results on the mini-ImageNet benchmark evaluate accuracy at the end of all missions according to 5.

It is worth noting that because the single client in situ update is a non-IID class incremental learning problem, federated reconnaissance generalizes class incremental learning to multiple clients. Therefore in this work, we evaluate a range of learning algorithms and backbones on both the federated reconnaissance problem and prototypical networks on the single client, few shot continual learning benchmark proposed by Javed and White [2019].

3 Methods

3.1 Learning Algorithms

For federated reconnaissance, we compare four approaches including a **Lower Baseline** which simply trains each client model with SGD on the current mission's data without access to previously seen data; a version of **iCaRL** adapted for federated reconnaissance [Rebuffi et al., 2017]; an extension of **Prototypical Networks** [Snell et al., 2017] for distributed, continual learning; and an empirical **Upper Baseline** in which all clients have access to all current and previous data and an unlimited compute budget for retraining with SGD.

To put federated prototypical networks into the context of existing work, we also compare to a state of the art few-shot learning method Online aware Meta-learning (OML) from Javed and White [2019] on existing, single client, few-shot continual learning benchmarks. We describe all methods further in Section 3.2 and in the appendix.

3.2 Federated Prototypical Networks

We propose to use prototypical networks to efficiently learn new classes in sequence. Given that prototypical networks are not gradient-based at test-time, they can be made to be robust to catastrophic forgetting when learning new classes by discriminative pretraining on a sufficiently large number of classes. When evaluated on a federated reconnaissance benchmark, we can compute an unbiased estimate of the mean (and the variance if we so desire) for each class by simply storing the previous prototype and the number of examples used to compute the previous prototype. We define prototypical networks following [Snell et al., 2017]:

$$\mathbf{z} = f_\theta\left(\mathbf{x}_i\right) \tag{6}$$

$$\bar{\mathbf{z}}_j = \frac{1}{|S_j|} \sum_{(\mathbf{x}_i, y_i) \in S_j} f_\theta\left(\mathbf{x}_i\right) \tag{7}$$

in which f is a neural network embedding function parameterized by θ and S_j is the support set of the class j. Training a prototypical network proceeds be minimizing the cross entropy loss over query examples, where the predicted class is taken as the softmax over negative Euclidean distances $d(\cdot)$ between query embeddings and the prototypes of the support data:

$$p_\theta(y = j \mid \mathbf{x}) = \frac{\exp\left(-d\left(f_\theta(\mathbf{x}), \bar{\mathbf{z}}_j\right)\right)}{\sum_{j' \in J} \exp\left(-d\left(f_\theta(\mathbf{x}), \bar{\mathbf{z}}_{j'}\right)\right)} \tag{8}$$

Now we would like to be able to compute unbiased estimates of prototypes for classes that are observed by multiple clients at the current time step or have been observed previously in exposure history. To improve storage and communication efficiency, instead of storing all raw examples for a class or even all example embeddings for a class, we instead can compute an unbiased running average for each prototype by storing the previous prototype and the number of examples used to compute it:

$$\mu_t = \frac{k_{t-1}\mu_{t-1}}{k_t} + \frac{(k_t - k_{t-1})\bar{\mathbf{z}}_j}{k_t} \tag{9}$$

where k_t is the number of examples in total observed of class j at time t, k_{t-1} is the number of total classes observed at time $t-1$ for class j, $\bar{\mathbf{z}}_j$ is the centroid for class j at time t, and μ_t is the online average of z from all examples for class j. Modulo numerical issues, via the law of large

numbers, such a running average will converge to the true prototype μ^* for each class, given a fixed parameterization of the embedding function f_θ.

$$\bar{\mathbf{z}}_k \xrightarrow{\text{a.s.}} \mu^* \quad \text{as } k \to \infty \tag{10}$$

Numerical issues cannot be so easily ignored in practice, so we use a more numerically stable algorithm for online averaging as proposed by West [1979], Schubert and Gertz [2018]:

$$\mu_t \leftarrow \mu_{t-1} + \frac{k_t - k_{t-1}}{k_t} \left(\bar{\mathbf{z}} - \mu_{t-1}\right) \tag{11}$$

Putting these ideas together, we arrive at Algorithm 1, which implements a mulit-client learning and model merging routine with prototypical networks. Algorithm 1 can be used for on client learning and knowledge transfer with a central server and even other clients for fully decentralized peer-to-peer learning. In addition to the algorithm shown in 1, we also evaluate a variant of the algorithm which removes the numerical effects of computing online averages by simply storing and transferring embeddings of all examples seen and computing the prototypes only at inference time.

For our experiments with prototypical networks, we reproduced the model architectures used in the original paper [Snell et al., 2017], though also experimented with a few key additional features. First, we find that overfitting to the pretraining base classes is a significant problem in the small-scale mini-ImageNet federated reconnaissance benchmark we propose, so additional regularization is necessary. We find that applying dropout with a droprate of 0.2 in the embedding space significantly improves learning new classes. Furthermore, we find that the accuracy performance after ingesting new examples from classes quickly plateaus for prototypical networks when ingesting more examples than they were trained on. The problem of prototypical networks over-specializing the pre-trained model to solving k-shot problems, where k is equal to the number of training shots per class, has also been noted in recent literature by Triantafillou et al. [2020a]. To address this problem, we find that the simple, model-agnostic augmentation strategy of sampling k at each iteration during pretraining yields significant benefits. We call this procedure k-shot augmentation and find that uniform sampling k from $[5, 50]$ during pretraining significantly improves distributed, continual learning.

3.3 Neural Network Architectures

We evaluate all learning algorithms with two different neural network backbones. First, in line with a large body of work on meta-learning, we use the typical 4-convolutional layer model (denoted 4conv in figures) as used in Snell et al. [2017], Finn et al. [2017] and many other works. This model contains 4 layers each with 3×3 2-d convolution with 64 output channels, batch normalization, a reLU non-linearity, and finally 2×2 maxpooling to cut the spatial dimensions in half at each layer. For the prototypical network model, we unroll the final feature channels following Snell et al. [2017]. In line with more recent work Triantafillou et al. [2020b], we also evaluate all learning algorithms with a resnet-18 backbone [He et al., 2016], which is a variant of the common residual network model that contains 18 layers.

4 Evaluation

4.1 Single Client Continual Learning

To put the strengths of using prototypical networks for continual learning into the context of prior work, we evaluate a single client continual learning benchmark put forth by Javed and White [2019] in which a learner is exposed to 30 examples from classes seen in non-overlapping succession. On this evaluation, we compare prototypical networks to the OML method proposed by Javed and White [2019] in a local reproduction. Following Javed and White [2019], the accuracy across all classes seen to date is computed at each evaluation.

4.2 mini-ImageNet Federated Reconnaissance Benchmark

To evaluate federated reconnaissance, a new benchmark was required. Taking inspiration from Javed and White [2019], we adapted the popular mini-ImageNet [Vinyals et al., 2016] dataset in order to create the mini-ImageNet Federated Reconnaissance Benchmark. The mini-ImageNet dataset

is composed of 100 classes taken from the Imagenet large scale visual recognition challenge [Russakovsky et al., 2015], with 600 examples each. We split the classes in half, yielding 50 base classes for pretraining and 50 field classes for online learning. Base classes are not seen again during online learning. The examples for each class are split into 500 training examples and 100 test examples for each class. In the default parameterization of the benchmark, we instantiate 5 clients and, during online learning, for each mission we sample 5 classes per client from field classes and 30 images per class. We resize all images to 84×84. We sample examples without replacement and the evaluation ends when all training examples have been sampled.

The problem with the simple accuracy metric in 3 is that in the presence of a pretraining dataset the metric will bias towards the pretraining classes in $\mathbb{B}$ until a sufficient number of new classes have been observed. To summarize the learning dynamics of federated reconnaissance, a metric that balances across base and field classes is required. We simply weight these two accuracies computed on the test sets of classes equally for all results reported in 5:

$$acc_{avg} = \frac{acc_{base} + acc_{field}}{2} \tag{12}$$

In Section 5, we report results from evaluating this accuracy metric after models have been merged back to the server from clients (Eval 2 diamond in figure 2). All results are evaluated on the union of the set of classes that any client has seen thus far at mission t, $\mathbb{M}_t$. All accuracy metrics we report here are computed against the test set $p_{test}(y|x)$ for both base and field classes.

5 Results

5.1 Federated Reconnaissance

Of all learning algorithms evaluated on the mini-ImageNet Federated Reconnaissance Benchmark, we find that federated prototypical network variants are the most accurate and computationally efficient (see Figure 3 and Table 1). In particular, we find that using k-shot augmentation during pretraining of prototypical networks is an effective means of improving the Federated Reconnaissance Benchmark accuracy of prototypical networks across a range of k values.

To better understand the learning dynamics of the algorithms evaluated on the Federated Reconnaissance Benchmark, we decompose acc_{avg} shown in eq. 12 into its constituent base and field accuracies as shown in figure 3 c and d. We find that prototypical networks are not only able to resist catastrophic forgetting of base classes and examples seen earlier during the Federated Reconnaissance Benchmark but can do so while improving accuracy on field classes. These results are in stark contrast to our adapted version of iCaRL, which suffers from catastrophic forgetting while also being unable to improve its accuracy in distinguishing new concepts as more data becomes available.

5.2 Single Client Continual Learning

On the single client on the class-incremental benchmark used in Javed and White [2019], Beaulieu et al. [2020], we find that our prototypical network significantly outperforms complex, state of the art continual learning algorithms, increasing the accuracy by over 22% (absolute) after learning 600 Omniglot classes and over 33% (absolute) after learning 20 mini-ImageNet classes incrementally (see Figure 1). See the appendix for more details.

6 Discussion

In this work, we have presented federated reconnaissance, a new a class of learning problems in which distributed clients learn new concepts independently and must be able to communicate that knowledge efficiently. We proposed an evaluation framework and evaluated a number of baseline learning algorithms for distributed continual learning. In particular, we derive simple algorithms for efficiently leveraging prototypical networks and find that they are a strong baseline method for federated reconnaissance and class-incremental continual learning. These results suggest that the simple idea of passing feature vectors is an important avenue for future research on federated reconnaissance and continual learning more generally.

7 Acknowledgements

This work was partially funded by U.S. Naval Sea Systems Command (NAVSEA) through the Small Business Technology Transfer (STTR) program, grant number N68335-20-C-0788. The authors would like to thank Andrew B. Leach for his excellent and discerning feedback on an earlier version of this manuscript.

References

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *arXiv preprint arXiv:1606.04080*, 2016.

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

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

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in neural information processing systems*, pages 6467–6476, 2017.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. *Proceedings of machine learning research*, 70:3987, 2017.

Gido M van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

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

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.

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

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.

Khurram Javed and Martha White. Meta-learning representations for continual learning. In *Advances in Neural Information Processing Systems*, pages 1820–1830, 2019.

Viraj Prabhu, Anitha Kannan, Murali Ravuri, Manish Chablani, David Sontag, and Xavier Amatriain. Prototypical clustering networks for dermatological disease diagnosis. *arXiv preprint arXiv:1811.03066*, 2018.

Shawn Beaulieu, Lapo Frati, Thomas Miconi, Joel Lehman, Kenneth O Stanley, Jeff Clune, and Nick Cheney. Learning to continually learn. *arXiv preprint arXiv:2002.09571*, 2020.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. Federated continual learning with weighted inter-client transfer. *Proceedings of the 4th Lifelong Machine Learning Workshop at ICML 2020*, 2020.

Michael Zhang, Karan Sapra, Sanja Fidler, Serena Yeung, and Jose M Alvarez. Personalized federated learning with first order model optimization. *International Conference on Learning Representations*, 2021.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

DHD West. Updating mean and variance estimates: An improved method. *Communications of the ACM*, 22(9):532–535, 1979.

Erich Schubert and Michael Gertz. Numerically stable parallel computation of (co-) variance. In *Proceedings of the 30th International Conference on Scientific and Statistical Database Management*, pages 1–12, 2018.

Eleni Triantafillou, Vincent Dumoulin, Hugo Larochelle, and Richard Zemel. Learning flexible classifiers with shot-conditional episodic (scone) training. *4th Workshop on Meta-Learning at NeurIPS 2020, Vancouver, Canada*, 2020a.

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. *International Conference on Learning Representations*, 2020b.

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.

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.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

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 interference. *International Conference on Learning Representations*, 2019.

Appendix A Introduction

We present supplementary material for our paper Federated Reconnaissance: Efficient, Distributed, Class-Incremental Learning. This supplementary material is mainly aimed at describing experimental details regarding reproducibility and hardware though also includes an additional ablation to inspect the effects of the number of base training classes.

Appendix B Single Client Continual Learning

We first describe further detail of evaluating our implementation of prototypical networks on the single client variant of federated reconnaissance. Specifically, we reproduce the benchmark used in Javed and White [2019], Beaulieu et al. [2020] and find that a simple four convolutional layer prototypical network outperforms the larger more complex model and learning algorithm of Javed and White [2019] that was specifically designed for continual learning. Prototypical networks increase the accuracy by over 22% (absolute percentage points) after learning 600 Omniglot classes and over 33% (absolute percentage points) after learning 20 mini-ImageNet classes incrementally 1.

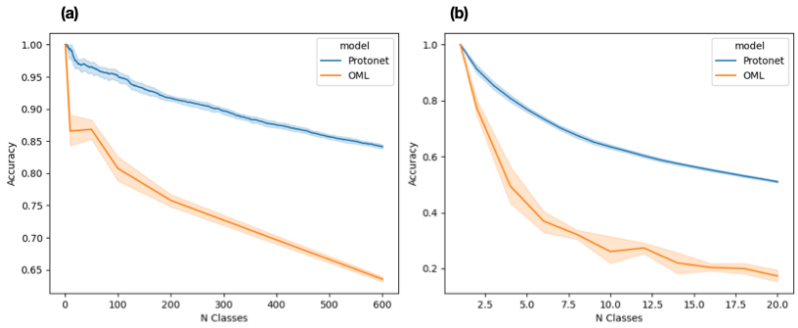

Figure 1: Single-client test-set continual learning results on Omniglot (a) and mini-ImageNet b. Accuracy is evaluated on test examples from tasks not seen during pretraining. At each evaluation step, the model is trained on only the classes added at that step and then is evaluated on held out examples from all previously seen classes.

Appendix C Federated Reconnaissance

In Figure 2, we show the evaluation framework of a federated reconnaissance system. A base model is pretrained on a set of base classes $\mathbb{B}$ and evaluated on held out examples of the classes in $\mathbb{B}$. This base model is then deployed to clients $1..h$. Each client is then trained via local supervision on instances of new or previously seen classes. After each client learns new classes, it is evaluated on the superset of previous and new classes. The clients then communicate their knowledge of new classes back to a central server which in turn deploys the merged knowledge back to clients. Finally, the model is evaluated on all classes in the benchmark.

To simplify the notation and our benchmark's implementation, each client learns classes on its mission in serial and the server model is updated synchronously. In practice, client learning will happen in parallel and the server model can be updated asynchronously.

While in this work we assume that each client communicates directly to a central server, the extension of direct peer-to-peer distributed learning of federated prototypical networks is straightforward, assuming that all examples observed by all clients are unique. In an environment in which all examples are not unique, such as is common in social media with reshared images, additional bookkeeping would be required to avoid double counting of embeddings and to keep the estimate of the prototype unbiased.

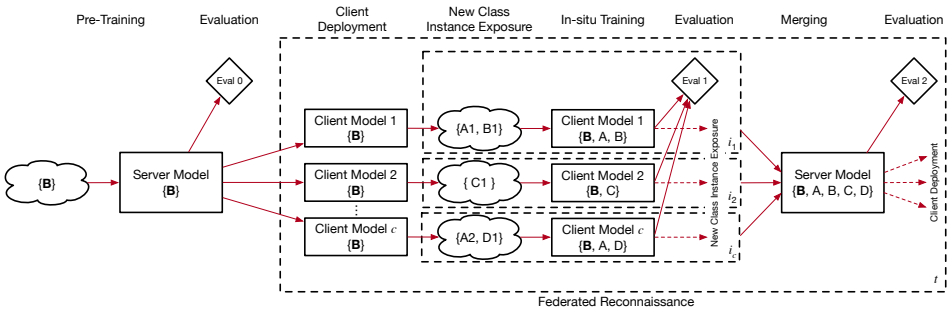

Figure 2: A diagram of the evaluation framework of a federated reconnaissance system.

Appendix D Experiments

For federated reconnaissance, we compare the following four approaches to distributed, continual learning:

1. **Lower Baseline**: To demonstrate the learning dynamics and the worst case effects of catastrophic forgetting, we train with SGD on the non-IID datasets $\mathcal{D}_t$ as they appear in sequence. This method serves as an empirical lower bound, which any other model designed for federated reconnaissance should be able to beat. The Lower method updates the parameters of the model on the new mission classes in $\mathcal{D}_t$ without access to any previously seen data.
2. **iCaRL** We adapt the iCaRL [Rebuffi et al., 2017] algorithm for federated reconnaissance. The original iCaRL algorithm was developed for the single client scenario and has no inherent notion of model merging. We took inspirations from the FederatedAveraging algorithm McMahan et al. [2017] to make it work on the federated reconnaissance problem. Whenever each client goes to mission, it obtains an initial set of parameters from the server. During a mission, the client sees a number of examples from new classes it has not seen before and trains the model on those new classes and some examples from base classes that are stored in a set of exemplars stored by the iCaRL model. For each new class, a new neuron is added to the fully connected layer of the model. If more than one client learned the same class, the neuron parameters are averaged, again similarly to the FederatedAveraging algorithm. This averaged model serves as an initial model which is further fine-tuned on base exemplars and new class exemplars using SGD.
3. **Federated Prototypical Networks** We extend prototypical networks for distributed, continual learning by deriving algorithms for online prototype updating and model merging. We describe this method in greater detail in section 3.2.
4. **Upper Baseline** To demonstrate the best case test-set results of a neural network architecture with access to all data seen by all clients and a liberal compute budget, during both on-client and on-server learning, we train with SGD on the joint distribution of all training data from all clients. This model serves as an empirical upper bound and comes at the storage and communication cost of saving and transferring all examples seen by all clients back to the server after each mission and retraining on all this data anytime new classes or examples are to be learned. Due to excessive computational expense, we did not evaluate the Upper baseline on the entire multi-mission Federated Reconnaissance Benchmark. Instead, we evaluated the model once by training with all training data of all classes, in the same way it would have been evaluated at the end of the benchmark.

All experiments were run 5 times to produce mean and confidence intervals of test-set metrics. Evaluation times are given in the main body of the paper.

D.1 Online Gradient-Based Methods

For pretraining the iCaRL, Lower Baseline, and Upper Baseline models, we pretrain on base 50 classes as described in the main body of the paper on the evaluation framework for the mini-

ImageNet Federated Reconnaissance Benchmarkexcept for upper method which starts with a model trained on all 100 classes. We trained the resnet-18 and the 4 layer convolutional network backbones using stochastic gradient descent up to 200 epochs with an initial learning rate 0.01, weight decay 1e-2, Nesterov momentum 0.9, and step learning rate decay with step size 50 and gamma 0.1. Validation dataset from 50 base classes is used to select the best model. The hyper-parameters for the online updates during federated reconnaissance evaluation are chosen by training the base model on 40 base classes and updating the model on a hold out set of 10 evaluation classes.

In greater detail for these methods which use gradients for learning new classes and examples during federated reconnaissance evaluation:

- **iCaRL**:
  During a mission, each client updates the model with a stochastic gradient descent and a learning rate 0.01, Nesterov momentum 0.9 up to 30 epochs. The temperature parameter for the distillation loss is set to 2. We used a budget of 4000 total examples. Examples are chosen at random to maintain in the cache, balancing across classes, meaning that there are 40 examples for each of the 100 mini-ImageNet classes that are stored, transferred between client and server, and used for training. After the model is merged, 50 steps of stochastic gradient descent are carried out on base exemplars and new class exemplars with a learning rate 0.01, weight decay 1e-3, Nesterov momentum 0.9, and step learning rate decay with step size 30, and gamma 0.1.

- **Lower:**
  During a mission, each client updates the model on new classes with a stochastic gradient descent with learning rate 0.01, Nesterov momentum 0.9, and a step learning rate decay with a step size of 30 and gamma of 0.1 up to 50 epochs.

- **Upper:**
  For the upper baseline we only evaluated the learning dynamics at the end of the Federated Reconnaissance Benchmark, due to excessive computational costs. We trained the two backbones on the training dataset of all 100 classes for 200 epochs using stochastic gradient descent with a learning rate 0.01, weight decay 1e-2, Nesterov momentum 0.9, step learning rate decay with step size 50 and gamma 0.1.

D.2 Federated Prototypical Networks

For pretraining the prototypical network backbones on the base classes, we closely follow the original implementation of Snell et al. [2017]. As such, we pretrain using the Adam optimizer [Kingma and Ba, 2015] for 30,000 episodes (i.e. gradient steps) with an episode/batch size of 5 randomly sampled classes with 5 support and 5 query examples per class. We use an initial learning rate of $1e-3$ which we half every 2000 episodes. The only significant changes we make to the original training regime are directed at reducing overfitting the training episodes. Namely, we:

1. Train with dropout [Srivastava et al., 2014] in the embedding space, with a drop probability of 0.2, and

2. Implement a novel augmentation procedure, which we call k-shot augmentation, in which we sample the number of support and query examples from a uniform distribution over $[5, 50]$.

Architecturally, we also experimented with global average pooling the feature maps into a 64-element vector, though did not find that this significantly changed results from unrolling the feature maps into 1600 element vector. Furthermore, we found that we could reduce the final convolutional layer outputs to 32 channels and global average pool to 32 element feature vectors without loss of accuracy, which identifies an additional means of compression when bandwidth and storage efficiency are critical.

Our method for client updating and knowledge merging of new classes across clients is shown in Algorithm 1.

Algorithm 1 Federated Prototypical Networks: Online Averaging & Model Merging

Input: $|\mathbb{C}|$ k-shot datasets $\mathcal{D}_{i,t}$ of $n_{i,t}$ classes, previous prototypes $\mu_{j,t-1}$, previous class counts $k_{j,t-1}$
for client c_i **in** $\mathbb{C}$ **do**
  for class $j \in \mathcal{D}_{i,t}$ **do**
    $\bar{\mathbf{z}}_{j,i} \leftarrow \frac{1}{k_{j,i}} \sum_{x_i, y_i \in \mathcal{D}_{i,t} | y_i = j} f_\theta(x_i)$
  end for
end for
{On-client model is optionally evaluated in situ.}
{Merge prototypes to server:}
for client c_i **in** $\mathbb{C}$ **do**
  for class $j \in \mathcal{D}_{i,t}$ **do**
    $k_{j,t} \leftarrow k_{j,t-1} + k_{j,i}$
    $\mu_{j,t} \leftarrow \mu_{j,t-1} + \frac{k_t - k_{t-1}}{k_t} (\bar{\mathbf{z}}_{j,i} - \mu_{j,t-1})$
  end for
end for
Return all centroids $\{\mu_{j,t} | j \in \mathbb{M}_t\}$ back to clients

Table 1: Accuracies and computational complexity of learning algorithms for federated reconnaissance. Accuracy is evaluated at the end of federated reconnaissance on a 100-way classification problem. All experiments were repeated five times and accuracy values show mean and 95% confidence intervals. Evaluation time is shown for one run of the Federated Reconnaissance Benchmark. Asymptotic analysis describes one mission of learning new data where E is the number of training epochs for gradient-based methods, J is the total number of classes seen thus far, K is the total number of examples per class seen thus far, $\mathcal{D}$ is the few-shot new dataset, γ is the compression factor of the embeddings (which, while constant, we include to differentiate from image space), and `Buffer` is the fixed-size iCaRL buffer for exemplars.

Learning alg.	Acc_{avg}	Eval time	Time cmplx	Space cmplx
Upper	$47.1 \pm 0.0\%$	4days	$\mathcal{O}(EJK)$	$\mathcal{O}(JK)$
k-aug. protos	$37.4 \pm 0.0\%$	1.5hrs	$\mathcal{O}(\lvert\mathcal{D}\rvert)$	$\mathcal{O}(\frac{JK}{\gamma})$
k-aug. onl. protos	$36.3 \pm 0.1\%$	1.5hrs	$\mathcal{O}(\lvert\mathcal{D}\rvert)$	$\mathcal{O}(\frac{J}{\gamma})$
Protonets	$34.2 \pm 0.0\%$	1.5hrs	$\mathcal{O}(\lvert\mathcal{D}\rvert)$	$\mathcal{O}(\frac{JK}{\gamma})$
Onl. protos	$33.6 \pm 0.1\%$	1.5hrs	$\mathcal{O}(\lvert\mathcal{D}\rvert)$	$\mathcal{O}(\frac{J}{\gamma})$
iCaRL	$20.1 \pm 1.7\%$	4.5hrs	$\mathcal{O}(E(\texttt{Buffer} + \lvert\mathcal{D}\rvert))$	$\mathcal{O}(\texttt{Buffer} + \lvert\mathcal{D}\rvert)$
Lower	$1.3 \pm 0.3\%$	3hrs	$\mathcal{O}(E\lvert\mathcal{D}\rvert)$	$\mathcal{O}(\lvert\mathcal{D}\rvert)$

D.3 Federated Reconnaissance Results

We report accuracy, evaluation time, and aymptotic space and time complexity, in Table 1. Online learning during evaluation across missions is visualized in Figure 3.

For prototypical networks, when comparing storing all embeddings in memory to computing the prototypes via online averaging, we find a slight, though statistically significant, degradation in accuracy due to numerical issues. We believe these numerical issues are due either to floating point imprecision, inherent non-determinism, catastrophic cancellation, or, likely, a combination of some number of those. We leave root causing these issues and the development or more numerically robust online prototype computation to future work. It is also clear from the results shown in figure 3 that the 4-layer convolutional model substantially outperforms the resnet-18 model across learning algorithms except for the upper baseline. The resnet model has significantly more capacity than the 4-layer convolutional model and we find that it overfits excessively to the pretraining examples.

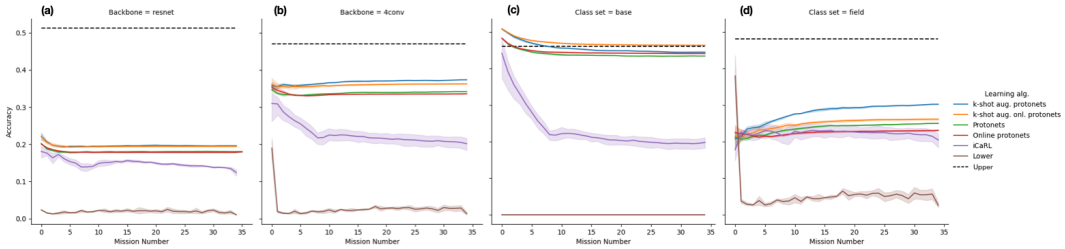

Figure 3: (a) and (b) plot federated reconnaissance results partitioned by neural network backbone and learning algorithm. Accuracy is evaluated on the test set for each class and averaged according to eq. 12 to balance base and field classes equally throughout evaluation. (c) and (d) plot federated reconnaissance results for the four layer convolutional model with the accuracy components of eq. 12 partitioned into base classes (c) and field classes (d). Higher accuracy on base classes indicates resistance to catastrophic forgetting while higher accuracy on field classes indicates ability to learn new classes presented in non-IID data online. Due to excessive computational expense, we did not evaluate the Upper baseline on the entire multi-mission Federated Reconnaissance Benchmark. The results of training the Upper model once with all training data of all classes in the same way it would have been evaluated at the end of the benchmark is shown in the black dashed line for reference.

D.4 Collective Ascent

We hypothesized the existence of a distributed learning phenomena, *collective ascent*, which occurs when the accuracy of a group of clients increases due to knowledge sharing. Collective ascent is trivial to identify in centralized model training with distributed example gathering as it only amounts to greater data collection. In contrast, collective ascent is non-trivial to produce during federated reconnaissance when compute or bandwidth are limited and centralized retraining on all observed examples is not an option because collective ascent entails positive forward *and* backward transfer[4] during distributed continual learning.

We found collective ascent occurs with prototypical networks by showing that a set of clients can gather data in parallel and communicate sparsely while still improving their accuracy on a shared knowledge base comprising a growing set of classes even without observing the examples of those classes directly. In Figure 4, each client receives 5 examples per class on each mission instead of 30. Such a parameterization of the experiment causes the difficulty (i.e. the "way") of the problem to increase more rapidly relative to the amount of data seen for each class, making it clear that the group of learners is effectively leveraging shared knowledge.

The observation of collective ascent on the federated reconnaissance mini-ImageNet benchmark (see Figure 4), highlights, in concert with our other results, that the simple procedure of sharing feature vectors representing shared concepts is an important avenue for federated reconnaissance and continual learning more generally.

D.5 Ablation of number of base training classes

To inspect the effects of the number of base classes on the final acc_{avg} of the mini-ImageNet Federated Reconnaissance Benchmark, we took the best learning algorithm and model architecture, the k-shot augmented prototypical network, and trained it on only a random selected subset of base classes. After this, we ran the rest of the benchmark as usual, learning an additional 50 classes in a distributed fashion. Due to computational expense, we only ran this experiment once per number of base classes. We found that the final acc_{avg} at the end of the Federated Reconnaissance Benchmarkincreased nearly linearly with respect to the number of base classes as shown in Figure 5. This highlights the importance of a large number of base classes for class incremental learning with embedding models.

[4]See Riemer et al. [2019] for a discussion of forward and backward transfer and interference in the context of continual learning.

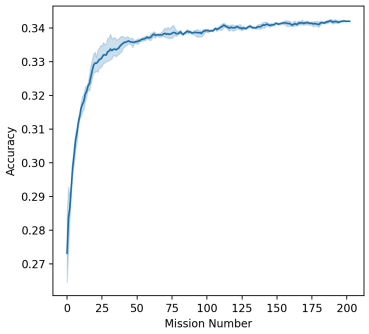

Figure 4: Collective ascent of accuracy after model merging (Eval 2 diamond in Figure 2) of federated prototypical networks as multiple clients gather and share information via algorithm 1. In this experiment, each client receives 5 examples per class on each mission, instead of 30. Accuracy is evaluated following eq. 12 on the mini-ImageNet Federated Reconnaissance Benchmark on all classes that any client has seen thus far, including base classes. Recall that during federated reconnaissance evaluation, examples are only seen once from field classes. This means that an accuracy improvement requires that the average of backward transfer (accuracy on previously seen classes) and forward transfer (accuracy on new classes) must be positive.

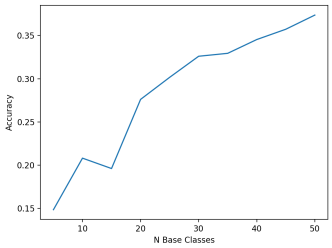

Figure 5: The final averaged accuracy acc_{avg} on Federated Reconnaissance Benchmarkfor the k-shot augmented prototypical networks when pretrained with across a varying number of base classes.

Appendix E Hardware

All experiments were run on a single GPU. We mock the parallel learning of clients for the purposes of initial experimentation by running each client in serial. Most experiments were run on NVIDIA P100 GPU nodes with 28 CPU cores and 224 GB of memory, from which we report evaluation times. A small number of additional experiments were run on an NVIDIA RTX 2080 GPU.