Federated Prototypical Learning for Automated Antibiogram Interpretation in Ultra-Low-Resource Settings

Nina Mainusch [†] **Sai Praneeth Karimireddy** [‡]
Louis Laroche [§] **Nada Malou** [§] **Antoine Descamps** [§] **Soriya Thach** [§]
Martin Jaggi [†] **Mary-Anne Hartley** [†]
[†] EPFL [‡] UC Berkeley [§] Médecins Sans Frontières

Abstract

Federated learning (FL) works best when given a large communication budget, stable network, and reliable clients. FL in resource-constrained settings such as between remote medical outposts requires a specifically optimized approach. We present a case study based on learning updates for a model that automates the interpretation of antibiograms (specifically the identification of antibiotic pellets). The four key implementation challenges are: 1) resource-efficient model updates that are 2) privacy-preserving, 3) robust to unreliable network connectivity, and able to 4) rapidly identify new, previously unseen data classes. We propose *federated prototypical learning* (FPL) to meet these challenges. FL generally enables continuous, real-time learning from unshared, private data across several users by communicating the local model updates (instead of the data) with a central server. In FPL, only the low-dimensional prototypical representations of each data class are communicated. This approach is not only significantly more resource-efficient and private, but it also supports one-shot learning of new antibiotic discs with offline adaptation. Our work is the first step toward finding a general, resource-efficient FL solution that is able to perform intermittent model updates and continual learning.

1 Introduction

Federated learning (FL) is a collaborative learning framework designed to train machine learning models from data distributed among several clients under the orchestration of a central server. The governing setting in which traditional FL operates assumes a large communication budget, communication stability, and client reliability. On the contrary, communication efficiency, communication disruption, and privacy preservation have been recognized as open problems in FL [11]. We identify resource-constrained problem settings in which FL would under-perform on a broad range of tasks and propose a solution to achieve reliable and resource-efficient continual learning [20].

An apt use case showcasing the limitations of FL is learning model updates from distributed users in remote medical outposts for the task of automated antibiogram interpretation. An antibiogram is used to evaluate the susceptibility of bacteria to antibiotics by exposing a microbe isolated from a patient to a concentration gradient of several antibiotics (seeping into the petri dish growth medium from paper discs infused with the various drugs). Expert analysis of the growth patterns around these discs ('inhibition zones') can then quantify susceptibility.

Antibiogo is an innovative artificial intelligence (AI)-based, offline smartphone application designed to assist in this complex analysis [16], and especially targeted at resource-constrained settings where expert analysis is difficult to access [10]. When the user takes a photo of the antibiogram with a smartphone camera, as shown in Figure 1(a), a machine learning model identifies the various antibiotic

Preprint. Under review.

discs by recognizing their printed text codes and computes the inhibition zones and interactions around each.

Antibiogo responds to the need for improved antibiotic stewardship as the prevention of antimicrobial resistance (AMR). The World Health Organization (WHO) estimates that by 2050 ten million will die annually from the consequences of AMR [22], and it is thus understandable why it is listed in the top ten threats to humanity. Under diagnostic uncertainty, doctors err on the side of caution and tend to over-prescribe broad-spectrum antibiotics [14], which, in a tragedy of the commons, leads to antibiotic resistance. Federated Learning could ensure access to the continuous, ubiquitous, and expert monitoring required to produce accurate predictions in an evolving environment.

In this work, we focus on the simplified classification sub-task of identifying the antibiotic disc from a photo taken with the Antibiogo app. Our contributions are:

- We identify a novel set of problems with broad applications in resource-constrained settings, where FL would need to be 1) resource-efficient, 2) privacy-preserving, 3) able to learn continuously, and 4) operable in unreliable network connectivity with intermittent model updates.
- We find that standard FL algorithms are not suitable for these settings. While employing a pretrained model improves the resource-efficiency, the other challenges remain unresolved.
- We propose FPL as a lightweight approach able to address several of these issues and test our approach on an large public benchmark dataset (EMNIST) and on a small real-world antibiogram dataset. The results demonstrate that FPL is a promising means of making robust federated learning accessible in resource-constrained settings.

Use case. We use the Antibiogo setting as a use case as it would both benefit from FL as well as require adaptations to resource constraints. The evolving nature of antibiotic resistance and fragmented supply chains means that the discs (brands, concentrations, drug names) and resistance patterns can change over time, creating systematic shifts that degrade accuracy. Indeed, static models are poorly adapted to dynamic environments and risk under-performing in new settings. Additional variation can be introduced by evolving mobile phone cameras and lighting conditions. It is thus essential to incorporate continual learning into AI-decision support tools such as Antibiogo, especially to detect outliers, which may be new classes of antibiotics or may indicate a risk to model performance. However, as medical data is challenging to share, privacy-preserving learning techniques are required. While FL may address these issues, traditional approaches depend on reliable connectivity and sufficient computational power in end devices, both of which are difficult to access in resource-constrained settings.

A solution found to these challenges does not only benefit Antibiogo but may be applicable to a much broader set of problems. Any clinical decision support system that assists the interpretation of medical images in the absence of experts faces the same problems. Having developed a suitable resource-efficient solution, such systems could assist in providing access to continual learning in rural and remote settings, with applications ranging from detecting broken bones, tumors, and tuberculosis and guiding the acquisition and assessment of various other point-of-care tests such as heart function and gestational screening [1, 3].

The applications of resource-constrained learning extend beyond rural medicine to assist in tasks based on resource-constrained environments such as wildlife conservation for animal counts, agricultural decision support in rural communities, or monitoring of the climate change crisis [2].

2 Related work

Prototypical networks (PNs) were first introduced by Snell et al. [18]. Tan et al. realized in parallel to our work the potential of PNs to address the problems of data heterogeneity and communication efficiency and privacy in FL [19]. They introduced *federated prototype learning* for heterogeneous FL settings, which is superior to standard federated algorithms that use gradient-based aggregation methods such as federated averaging (`FedAvg`) [5]. Due to their novelty, PNs and FPL have only been applied to benchmark datasets like MNIST, FEMNIST, and CIFAR10 [9, 6, 13]. In contrast to Tan et al., who focus on the theoretical analysis of FPL, our work identifies the general problem setting of resource-efficient FL that requires intermittent model updates and continual learning. We propose

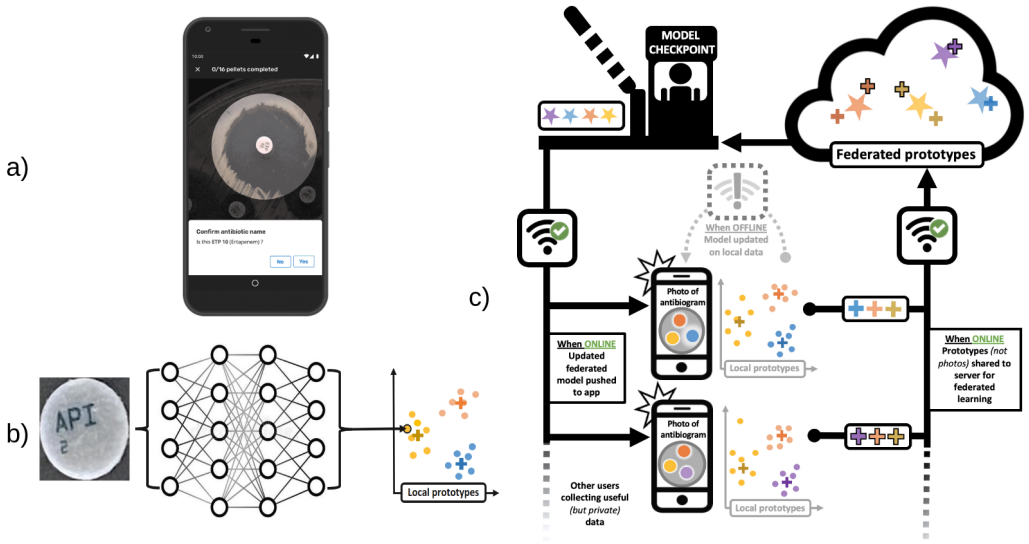


Figure 1: **Overview (a)** Screenshot of Antibiogo. The learning task is recognizing the text of an antibiogram disc. **(b)** Prototypical learning applied to a photo taken with Antibiogo. A model maps the photo to an embedding space, where the prototype of each class emerges as the mean of the support set. **(c)** In FPL, many clients locally classify their antibiogram photos and generate local prototypes offline. When online, the prototypes are sent to a central server which aggregates global prototypes and forwards them to a control instance. If the new prototypes pass the checkpoint, they are sent back to all clients, resulting in an updated model.

FPL as one potential method to address this problem, and hope that it may provide momentum for further research on this issue to expand robust FL to resource constrained settings.

3 Methods

3.1 Federated prototypical learning

In FPL, as depicted in Figure 1(c), each client learns a non-linear mapping of the input into an embedding space, shown in Figure 1(b). The prototype of each class emerges as the mean of the support set in the embedding space and can be computed locally and offline. In the embedding space, classification is performed by computing the euclidean distances to the prototype of each class. Upon connectivity to the internet, clients can send their low-dimensional (and private) prototypes to the central server. Until then, each client can continuously incorporate new locally classified data and update the respective prototype by computing a weighted average of the current prototype and the freshly classified embedded data point. The central server aggregates the global prototypes and forwards them to a control instance. If the new prototypes pass the human checkpoint, they are sent back to all clients.

An advantage of prototypical networks is their ability to perform one-shot learning, i.e., detecting and classifying an instance of new, unseen data [18]. A one-shot learning strategy is to investigate the distance to the closest prototype. If the distance is beyond a threshold τ, then the data point serves as the prototype of a new class. To determine τ, we can estimate the parameters of an underlying normal distribution of distances and set τ as the right border of a one-sided confidence interval around it.

The challenge of data privacy is solved by encasing prototypical learning in the FL paradigm. Instead of communicating the data, only the lower dimensional prototypes are sent to the server. This naturally aggregates the data to ensure privacy and, furthermore, enables heterogeneous data sources to learn smoothly together [19].

3.2 Experimental setup

We first focus on comparing the performance and communication complexity of FPL to standard FL and a supervised learning (SL) baseline. Although SL and FL are not per se applicable to resource-efficient FL, we consider it beneficial to quantify the potential performance increase or decrease of FPL over FL and SL. Afterward, we will specifically investigate the ability of FPL to perform one-shot learning.

As a model architecture, we use the VGG16 model with the weights pretrained on the imagenet dataset [17, 8]. Next to this pretrained model, we train a three-layer convolutional network (3-layer CNN) to evaluate whether a smaller model suffices for the learning task. As datasets, we choose the EMNIST and AMR data [7]. Since the EMNIST dataset is more exhaustive and the results therefore more meaningful, we will concentrate our analysis on it.

Federated learning. We implement the standard federated setup where each client possesses a model and data locally and sends and receives information to and from the central server. The iterative process of FL consists of a client and a server update. The server is updated with the `FedAvg` algorithm [5]. `FedAvg` aggregates all the client updates by computing a weighted average of the model weight updates based on the number of samples at each client. However, when the data is *non-interoperable*, i.e. each client has different disjoint subsets of data classes, `FedAvg` and other federated algorithms suffer from a problem termed 'client-drift', which decelerates and destabilizes the model convergence. To alleviate this problem, we implemented the `Scaffold` and the `FedDyn` algorithm [12, 4]. To test the performance of FL in a resource-constrained setting we consider a *uniform* and a *non-interoperable* data setting with 2 clients.

Supervised learning. We simulate a real-world scenario by finetuning a VGG16 model with SL on $none, 50\%, 75\%, 100\%$ of the classes. To obtain confidence estimates, we repeat the experiments for each split 5 times. The class of a data point can be *seen* if the class was used in the finetuning or *unseen* if it was not, see Table 1. To get a comparable result for the unseen classes, we retrain the output layer on data points from the unseen classes. In FPL we do not have to retrain the last layer, since it is removed when constructing the protonet.

Federated prototypical learning. To test the FPL paradigm, we finetune a VGG16 backbone model on the same class splits as SL. This way we can investigate how well FPL is suited for a continual learning task where it is confronted with unseen classes. We compute the prototypes with all samples from the training set for the seen and unseen classes. The embedding space for FPL is obtained by removing the output layer of the finetuned backbone model. We show the results for a simplified test setting of 1 client, where all the training data is known in advance. In this setting, it does not change the final coordinates of the global prototypes whether the weighted average of the local prototypes is calculated at one or among several clients. It also does not change the final coordinates whether the data is uniformly distributed among the clients or non-interoperable.

Table 1: **Look-up table** to understand the combination of train/test and seen/unseen.

	FPL	**SL**
Train	used for prototypes	used for re-training
Test	not used for prototypes	not used for re-training
seen class	used in finetuning of VGG16	
unseen class	not used in finetuning of VGG16	

4 Results

Performance of FL. The averaged train and test accuracies are summarized in Table 2. Since the results for the `FedAvg` and `FedDyn` algorithm are either worse or on par with `Scaffold`, we only show the results for `Scaffold`. As expected, the train and test accuracies for the 3-layer CNN model and the VGG16 model on the EMNIST dataset in the non-interoperable setting are lower for the same number of communicated bits than in the uniform setting. The test accuracies of the 3-layer

Table 2: **Federated training** results for 2 clients and the `Scaffold` algorithm [12]. Train and test accuracies for an increasing number of communicated bits for the uniform and non-interoperable data split.

Dataset	**Model**	**Comm. bits**	1e+9		1e+10		9e+10		9e+11	
		Accuracy	Train	Test	Train	Test	Train	Test	Train	Test
EMNIST	**3-layer CNN**	uniform	0.85	0.84	0.86	0.85	0.87	0.85	-	-
		non-interop.	0.39	0.36	0.72	0.60	0.73	0.60	0.73	0.60
EMNIST	**VGG16**	uniform	0.59	0.76	0.85	0.85	0.90	**0.86**	-	-
		non-interop.	0.004	0.005	0.05	0.06	0.57	0.57	0.85	**0.79**
AMR	**3-layer CNN**	uniform	0.96	0.85	1.00	0.90	1.00	0.90	-	-
		non-interop.	0.19	0.17	0.95	0.65	1.00	0.90	1.00	0.90
AMR	**VGG16**	uniform	0.55	0.50	0.98	0.95	1.00	1.00	-	-
		non-interop.	0.10	0.09	0.49	0.46	0.98	0.90	1.00	1.00

Table 3: **Federated prototypical learning.** Test and train accuracy (in %) of FPL.

Dataset	**Class split (finetuning)**	**Train acc. Seen class**	**Train acc. Unseen class**	**Test acc. Seen class**	**Test acc. Unseen class**	**Comm. bits**
EMNIST	none	-	64.0	-	62.0	1.8e+6
EMNIST	50%	74.6 ($\pm$ 1.1)	69.8 ($\pm$ 1.5)	75.2 ($\pm$ 1.3)	**71.2 ($\pm$ 0.8)**	1.8e+6
EMNIST	75%	77.4 ($\pm$ 0.9)	69.0 ($\pm$ 2.3)	78.7 ($\pm$ 1.1)	71.2 ($\pm$ 3.9)	1.8e+6
EMNIST	100%	79.0 ($\pm$ 0.0)	-	80.2 ($\pm$ 0.4)	-	1.8e+6
AMR	none	-	85	-	83	2.2e+5
AMR	50%	95.9 ($\pm$ 1.7)	84.8 ($\pm$ 3.5)	90.3 ($\pm$ 2.6)	76.3 ($\pm$10.7)	2.2e+5
AMR	75%	98.5 ($\pm$ 0.9)	81.0 ($\pm$ 6.1)	94.7 ($\pm$ 0.6)	84.0 ($\pm$13.9)	2.2e+5
AMR	100%	100	-	93	-	2.2e+5

CNN model in the non-interoperable setting do not get as high as the test accuracies for the VGG16 model, even for a larger numbers of communicated bits. This shows that a small CNN model trained from scratch is less suitable for a resource-constrained data setting than the pretrained VGG16 model. However, the communication load is very high in both cases: only after $9\text{e}{+11}$ communicated bits displays the VGG16 model on the non-interoperable data a test accuracy of 0.79, which is close to the best uniform accuracy of 0.86. Given the poor performance of the 3-layer CNN model compared to the VGG16 model, we will focus on the results for the pretrained VGG16 model in the following experiments.

Performance of FPL. The resulting test and train accuracies are displayed in Table 3. Surprisingly, even with having seen only 50% of the classes in the finetuning does FPL achieve a test accuracy on the unseen classes of $71.2\% \pm 0.8$, which is the same as when finetuned on 75% of the data classes. Even with no finetuning but just the generic imagenet pertaining does FPL achieve a test accuracy of 62.0%, for any distribution of the data among the clients, uniform or non-interoperable. This is evidence that FPL is well suited for continual learning on heterogeneous data.

Communication complexity of FPL. We can observe that the communication complexity is considerably lower in FPL. For every class c and an embedded dimension d for each prototype, we send $c \cdot d$ parameters per client to the server. For EMNIST, this amounts to $12,400$ parameters, compared to $14,749,521$ parameters that are communicated in SL. This shows that FPL is better suited than standard FL for a resource-constrained setting where only a limited amount of bits can be communicated.

Performance of FPL compared to SL. The train and test accuracies obtained with SL are listed in Table 4. When finetuned on the full training data, SL outperforms FPL on the EMNIST dataset by 7.4 accuracy points and the best non-interoperable federated accuracy by 8.6 points. This suggests

Table 4: **Standard supervised learning.** Test and train accuracy (in %) of SL.

Dataset	Class split (finetuning)	Train accuracy Seen class	Test accuracy Seen class	Test accuracy Unseen class	Comm. bits
EMNIST	none	-	-	0.0	9.4e+8
EMNIST	50%	100.0 (± 0.0)	74.8 (± 1.6)	69.6 (±2.5)	9.4e+8
EMNIST	75%	100.0 (± 0.0)	79.4 (± 2.4)	67.6 (±4.7)	9.4e+8
EMNIST	100%	100.0 (± 0.0)	**87.6 (±0.5)**	-	9.4e+8
AMR	none	-	0.0	0.0	9.4e+8
AMR	50%	100.0 (± 0.0)	87.3 (± 11.3)	88.8 (±14.3)	9.4e+8
AMR	75%	100.0 (± 0.0)	97.0 (± 0.7)	95.4 (±3.1)	9.4e+8
AMR	100%	100.0 (± 0.0)	97.0 (±0.7)	-	9.4e+8

that the FPL backbone model cannot cluster the data symmetrically in the last hidden layer, but learns meaningful decision boundaries in the final sigmoid layer. When turned into a prototypical network by removing the last sigmoid layer, the performance therefore decreases. The test accuracies of SL on the unseen classes for class splits 50% and 75%, however, have a lower mean than the corresponding FPL test accuracies on the unseen classes from Table 3, even though the standard deviations overlap. This shows that FPL generalizes at least as well as SL to unseen classes, and we do not have a significant decrease in performance when employing FPL. It also underlines that FPL can classify unseen classes no matter the amount of classes it has seen in the finetuning, whereas SL needs a significant amount of classes before it starts working.

One-shot learning with FPL. We show the euclidean distance distribution from a data point to its prototype if it was seen in the finetuning, or its closest prototype if it was not seen in the finetuning in Figure 2. Although there is a tendency, especially for the AMR dataset in Figure 2(a), we cannot clearly identify two distinct distance distributions between the seen and unseen classes. One idea to improve the performance of FPL is to introduce label smoothing to the finetuning, since it has been show to tighten the class clusters in the penultimate network layer, i.e., the embedding space of the protonet [15]. It would thus decrease the intra-cluster distance and increase the inter-cluster distance, which translates to decreasing the variance of the normal distributions in Figure 2.

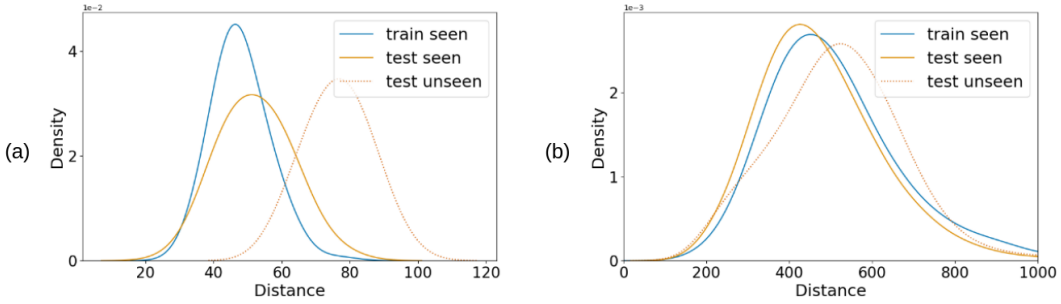


Figure 2: **One-shot learning with FPL.** Euclidean distance distribution for the (a) AMR and (b) EMNIST dataset.

5 Conclusion

We identified a novel problem setting for FL with broad applications in resource-constrained environments. Proposed solutions need to be resource-efficient, privacy-preserving, able to learn continuously, and operable in unreliable network connectivity with intermittent model updates.

We theoretically justified the aptness of FPL for this setting and confirmed its performance in a simplified 1-client setting. We quantified the amount of communication that can be saved with FPL and provide evidence on its potential to support one-shot learning.

We hope this work will provide momentum for further research in the field of resource-efficient FL. In particular, a more thorough testing of FPL in a continual learning setting with multiple clients, as well as a more formal quantification of the privacy of sharing prototypes. Finally, further experimental evidence on the potential for one-shot FPL should be explored across several datasets. Our work is a first step toward a resource-efficient FL solution able to handle intermittent updates and perform continual learning.

References

[1] Aidoc: deep learning tailored for radiology. `https://www.aidoc.com/`. Accessed: 2022-09-13.

[2] CottonAce: pest management for cotton farming. `https://www.wadhwaniai.org/rcats/pest-management/`. Accessed: 2022-09-13.

[3] Moxie: ai-powered poop scanner. `https://poop.moxie.health/`. Accessed: 2022-09-13.

[4] D. A. E. Acar, Y. Zhao, R. Matas, M. Mattina, P. Whatmough, and V. Saligrama. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2021.

[5] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Agüera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. 2017.

[6] S. Caldas, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, and A. Talwalkar. LEAF: A benchmark for federated settings. *CoRR*, abs/1812.01097, 2018.

[7] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926, 2017.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[9] L. Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[10] EUCAST. Eucast disk diffusion method. European Committee on Antimicrobial Susceptibility Testing. 2020.

[11] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao. Advances and open problems in federated learning, 2019.

[12] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. 2020.

[13] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[14] C. Llor and L. Bjerrum. Antimicrobial resistance: risk associated with antibiotic overuse and initiatives to reduce the problem. *Ther Adv Drug Saf*, pages 5(6):229–41, 2014.

[15] R. Müller, S. Kornblith, and G. E. Hinton. When does label smoothing help? *CoRR*, abs/1906.02629, 2019.

[16] M. Pascucci, G. Royer, J. Adamek, M. Al Asmar, D. Aristizabal, L. Blanche, A. Bezzarga, G. Boniface-Chang, A. Brunner, C. Curel, G. Dulac-Arnold, R. M. Fakhri, N. Malou, C. Nordon, V. Runge, F. Samson, E. Sebastian, D. Soukieh, J.-P. Vert, C. Ambroise, and M.-A. Madoui. AI-based mobile application to fight antibiotic resistance. *Nat Commun 12*, page 1173, 2021.

[17] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–14, 2015.

[18] J. Snell, K. Swersky, and R. S. Zemel. Prototypical networks for few-shot learning. *CoRR*, abs/1703.05175, 2017.

[19] Y. Tan, G. Long, L. Liu, T. Zhou, Q. Lu, J. Jiang, and C. Zhang. Fedproto: Federated prototype learning over heterogeneous devices, 2021.

[20] G. M. van de Ven and A. S. Tolias. Three scenarios for continual learning. *CoRR*, abs/1904.07734, 2019.

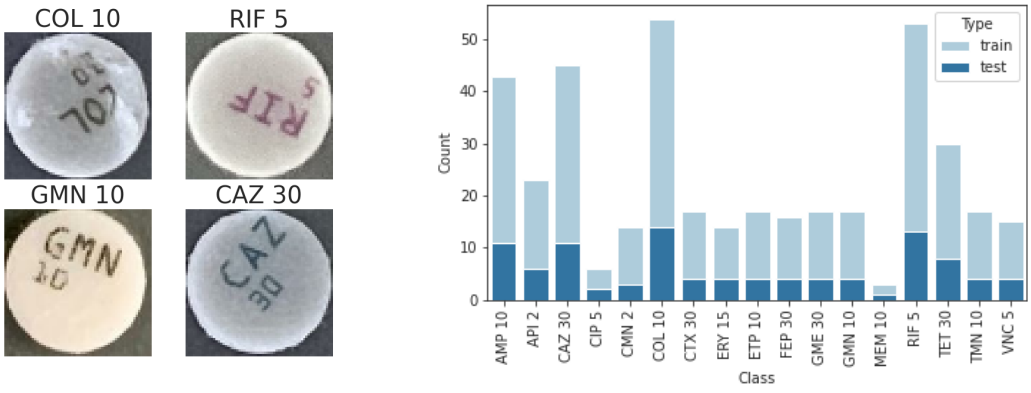
[21] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[22] WHO. Antimicrobial resistance, 2021.

A Data

The real-world antibiotic disc (AMR) dataset comprises 502 images of antibiotic discs, which are 6.5mm circular cutouts of absorbent paper impregnated with precise concentrations of a single antibiotic. The coded abbreviation of the name and concentration of the antibiotic is printed on the disc in various colors and fonts, depending on the manufacturer. The text may appear rotated at any angle, and the paper may also be variously colored. A frequent issue in interpretation is faded lettering. A $80:20$ train:test split (n=$401:101$) made previously was maintained to ensure comparability [16]. We further separate 40 random images from the training set as validation data. We display four random samples from the antibiotic disc data in Figure 3a, where each image in the dataset has the dimensions $64 \times 64 \times 3$. The dataset comprises 17 classes of antibiotics, where the number of samples per class is not uniformly distributed (Figure 3b).

We validate our preliminary results from the AMR dataset on the more extensive EMNIST dataset [7].



(a) (b)

Figure 3: **(a)** Four random samples from the dataset. **(b)** Distribution of classes in the training and test data. There are 401 training images and 101 test images in total, divided into 17 classes.

B Methods

Federated prototypical learning. A complete description of FPL can be found in Algorithm 1. We denote with $\bar{P}^{(c)}$ the global prototype of class c and with $P_k^{(c)}$ the prototype of class c from client k. $N_k^{(c)}$ denotes the number of data points of class c at client k and $\bar{N}^{(c)}$ denotes the number of data points of class c over all clients. ϕ_x represents an embedded data point x with class $P(x)$, which is the prototype with the minimum distance to ϕ_x. τ is the threshold for a new data point to be classified as belonging to a new class.

C Experimental setup

Models. We implement all our models in TensorFlow, which is consistent with the implementation of the existing Antibiogo application. To obtain a reference model for our learning task, we use the VGG16 model architecture with the weights pretrained on the imagenet dataset [17, 8]. The last layer of this pretrained model was replaced by a dense layer with 200 neurons, a dropout layer with a dropout rate of 0.1, and a final dense layer with as many neurons as there are classes. Next to the pretrained model, we train a three-layer convolutional network (convnet) to evaluate whether a smaller model suffices for learning the antibiotic discs. For the VGG16 model, we use an initial learning rate of 0.00001 and for the smaller convnet 0.0001. The initial optimizer parameter values are for both models set to $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-7}$.

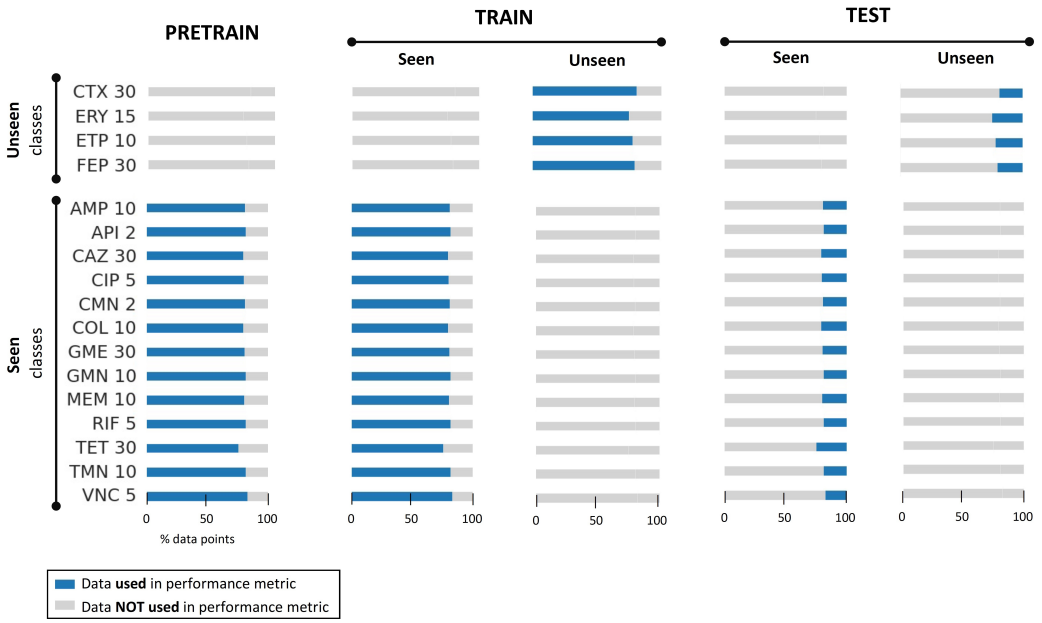


Figure 4: **The experimental setup** used to calculate the prototypes with the AMR dataset. $^{13}/_{17}$ of the classes were used to finetune the VGG16 backbone model, and thus the backbone model has seen them. $^{4}/_{17}$ of the classes were not seen by the backbone model before.

Federated prototypical learning. For our experiments on FPL, we split the training and test data into *seen* and *unseen*, where the data classes were either seen in the finetuning or not. In Figure 4 we display the experimental setup used to calculate the prototypes with the AMR dataset.

D Results

D.1 Federated Prototypical Learning

In Table 5 we answer the question of how many data points per unseen class FPL needs to calculate representative prototypes, and how many data points per unseen class SL needs for retraining its output layer in order to achieve a comparable test accuracy. FPL reaches a test accuracy of 68.5% with only 5 data points per class. SL needs around 20 data points per class before reaching its comparable test accuracy of 69.2%. Since for more data points the test accuracies do not increase significantly, we can state that SL needs 4 times as many data points as FPL to achieve a comparable test accuracy.

Table 5: Test accuracy (in %) on unseen classes for a varying number of data points.

Dataset	**Model**	**Test accuracy on unseen classes** (datapoints per class)			
		5	20	35	50
EMNIST	FPL	**68.5 (± 3.7)**	70.8 (± 3.6)	71.6 (± 4.0)	72.3 (± 4.0)
EMNIST	SL	56.7 (± 5.1)	**69.2 (± 2.4)**	71.0 (± 3.4)	71.3 (± 4.9)

Pretrained model. In Table 6 we display the test and train accuracies of FPL applied to an VGG16 model pretrained on imagenet, but not finetuned on the AMR data. The VGG16 model is a convolutional neural network that is 16 layers deep. We can see that removing more layers at the end of the pretrained model before turning it into a prototypical network can greatly benefit train and test accuracy. The hypothesis underlining this behavior is that the last layers of a neural network are specializing in specific structures from the training data, whereas the earlier layers learn general

Algorithm 1 `Federated prototypical learning`

Input:
Server executes:
Initialize the global prototype set $\bar{\mathcal{P}} = \{\bar{P}^{(c)}\}$ for all known classes c.
Initialize a pre-trained backbone model M for each client.
for each round $t = 1, 2, ...$ **do**
  $\bar{\mathcal{S}}_t \leftarrow$ (set of available clients)
  for each client $k \in \bar{\mathcal{S}}_t$ **in parallel do**
    $\mathcal{P}_k, \{N_k^{(c)}\} \leftarrow$ ClientUpdate($k, \bar{\mathcal{P}}$)
  end for
  $\bar{N}^{(c)} = \sum_{k \in \bar{\mathcal{S}}_t} N_k^{(c)}$, $\bar{P}^{(c)} = \frac{1}{|\bar{\mathcal{S}}_t|} \sum_{k \in \bar{\mathcal{S}}_t} \frac{N_k^{(c)}}{\bar{N}^{(c)}} P_k^{(c)}$ ▷ Update the global prototypes
  Update local prototype set $\mathcal{P}_k$ with prototypes in $\bar{\mathcal{P}}$.
end for

ClientUpdate$(k, \bar{\mathcal{P}})$:
for each sample x **do** ▷ Classify the data at each client
  $\phi_x = M(x)$ ▷ Compute the embedded vector
  $P(x) = \arg\min_{\bar{P}^{(c)} \in \bar{\mathcal{P}}} \|\bar{P}^{(c)} - \phi_x\|$
  if $\|P(x) - \phi_x\| > \tau$ **then** ▷ If the distance to any known Prototype is too large
    $\bar{P}^{(c_{new})} = \phi_x$ ▷ we found a new class
  end if
end for
for each class c **do** ▷ Including the newly found classes
  $\mathcal{C}_k^{(c)} = \{x \mid P(x) = \bar{P}^{(c)}\}$ and $N_k^{(c)} = |\mathcal{C}_k^{(c)}|$ ▷ Cluster data by class
  $P_k^{(c)} = \frac{1}{N_k^{(c)}+1} \left(\sum_{x \in \mathcal{C}_k^{(c)}} \phi_x + \bar{P}^{(c)} \right)$ ▷ Update the local prototypes using mean
end for
return new prototypes $\mathcal{P}_k = \{P_k^{(c)}\}$ and cluster sizes $\{N_k^{(c)}\}$

coarse structures. By removing the last layers we thus make the model more robust. Removing two layers increases the test accuracy by 3 percent compared to not removing a layer.

Table 6: Test and train accuracy (in %) of FPL with a VGG16 backbone model pretrained on imagenet with no finetuning on the AMR dataset.

Removed layers	Train accuracy	Test accuracy
0	84.54	83.17
1	85.29	81.19
2	90.77	**86.14**
3	88.78	80.2
4	89.03	80.2
5	85.54	75.25
6	89.78	78.22
7	**91.02**	82.18
8	88.78	76.24
9	89.28	73.27
10	88.78	73.27
11	86.53	73.27
12	86.78	65.35
13	89.53	61.39

Prototypical plot for the AMR dataset To visualize FPL for the AMR dataset as depicted in Figure 5 and Figure 6, we reduce the embedding dimension first to 50 and then to 2 with the help of PCA and t-SNE and denote the prototype of each class with a + symbol enhanced in size [21]. The classes that were not included in the finetuning of the backbone model, i.e., `CTX 30`, `ERY 15`, `ETP 10`, and `FEP 30`, are shaded dark. We can see that 11 of the embedded clusters are well separated.

Most misclassifications arise from 4 partially overlapping clusters caused by the unseen classes not included in the finetuning. Since there are visually identifiable clusters for the train and the test data, we conclude that FPL succeeds in finding an embedding of the data and calculating a prototype well representing each class.

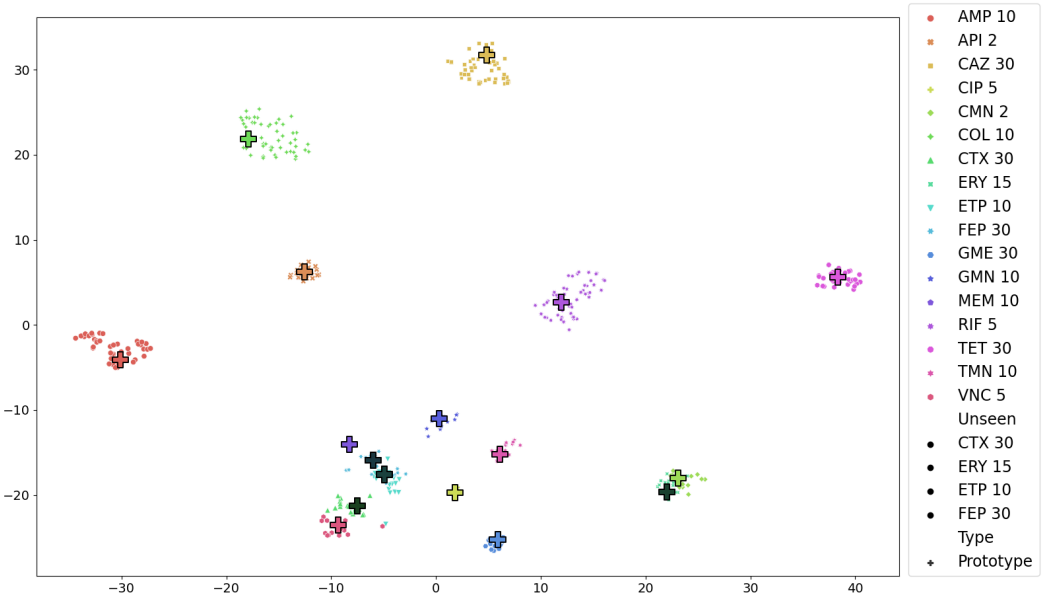


Figure 5: **Embedded representation for FPL of the training data** based on the VGG16 backbone model architecture, which was finetuned on 75% of the AMR data, see Figure 4 for the exact data split. We denote the prototype of each class with a **+** symbol enhanced in size. Classes unseen in the pre-training are shaded dark.

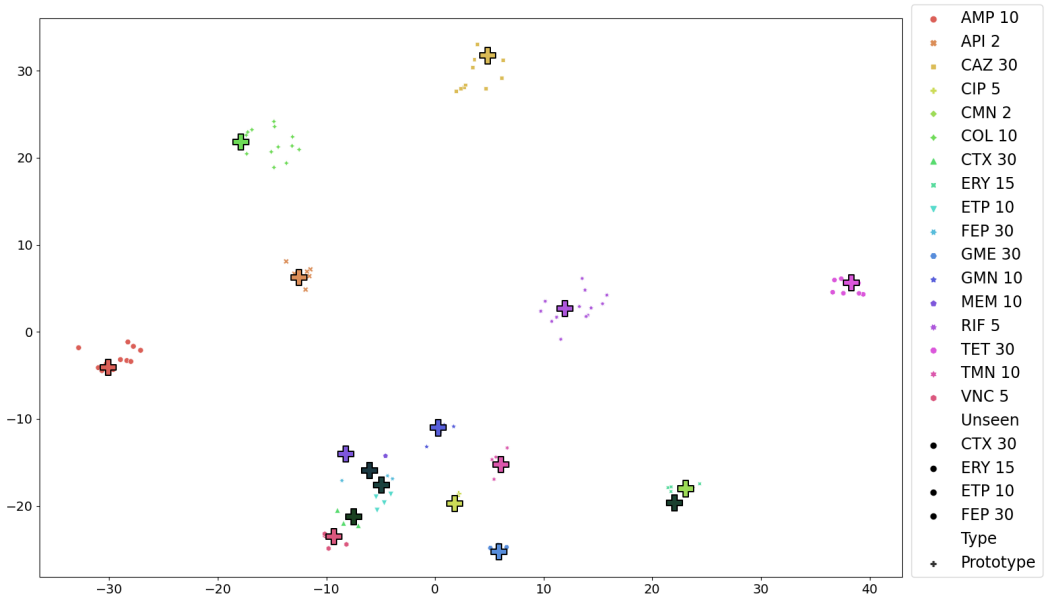


Figure 6: **Embedded representation for FPL of the test data** based on the VGG16 backbone model architecture, which was finetuned on 75% of the AMR data, see Figure 4 for the exact data split. We denote the prototype of each class with a **+** symbol enhanced in size. Classes unseen in the finetuning are shaded dark.