inDISCO: INterpretable DIStributed COllaborative learning for images

Klavdiia Naumova [1] **Mary-Anne Hartley** [1 2] **Arnout Devos** [1] **Sai Praneeth Karimireddy** [3] **Martin Jaggi** [1]

Abstract

DIStributed **CO**llaborative (**DISCO**) learning allows several data owners (clients) to learn a joint model without sharing data. While this approach could transform data usage in privacy-sensitive domains such as healthcare, the restricted access to each client's data limits interpretability and may conceal bias or interoperability mismatches that could compromise model performance. This issue is particularly important for image data since most deep neural networks for images are already black-box models. We address this problem with `inDISCO`, which adapts a well-known interpretable prototypical part learning network (ProtoPNet) to a federated setting allowing members of the federation to directly visualize the differences in the features learned from each client. We show that it can identify, attribute, quantify, and potentially correct bias in distributed collaboration without sharing any data. This work could be extended to interpretable personalization in federated learning.

1. Introduction

"*What you can't see can't hurt you...?*" — Proverb

As the *bigness* of big data becomes ever bigger and more granular, so too does its potential power, value, risk, and the legal constraints of sharing it. To address this limitation, federated learning (**FL**) was proposed by (McMahan et al., 2017) to allow multiple data owners (clients) to collaboratively train a model while keeping their datasets locally. While FL is potentially transformative for domains with privacy-sensitive data such as healthcare, the privacy gained comes at a major cost to transparency. Indeed, the real-world use of deep learning is already limited by black box *models*, and FL compounds the issue with black box *data*. Visualizing data is especially important when applied to real-world data which is often biased or imperfectly interoperable. Understanding differences will allow an informed selection of collaborators and enable explainable predictions.

Problem setting. The inability to visualise data across clients in FL is particularly problematic for imaging data since the deep learning approaches for this modalitiy are already poorly interpretable. This combination of black box model and black box data, obfuscates both the reasoning process of the model as well as the quality of its data. This work addresses the question of how we can achieve data transparency without compromising its privacy. Specifically, we ask, how can we identify, quantify, explain, and even correct for bias in unseen imaging data in the FL setting while preserving privacy?

We adapt a well-known interpretability method introduced by Chen et al. (2019), called prototypical part learning (`ProtoPNet`). Their implementation (Appendix, Fig 3) uses a CNN to create a set of patches in the latent space, a prototypical patch is learned from this set. Classification then relies on a similarity score computed between this learned prototype and a latent representation of a test image. A prototype can be visualized by highlighting the patch most activated by this prototype.

We propose to adapt ProtoPNet to FL. As summarized in Fig 1, clients each learn their own local prototypes as well as globally in communication with each other. The patches most activated by each of these prototypes can be visualized and compared on each client's local test set. By comparing global and local prototypes the clients can assess the interoperability of the data. Thus, we can introduce interpretability to FL and directly examine the predictive impact of other data without compromising clients' privacy.

Our main contributions are as follows:

1. We formalize a use case and create an imperfectly interoperable benchmark image dataset.
2. We introduce `inDISCO` adapting ProtoPNet to FL and compare its performance to baseline models.
3. We demonstrate how inDISCO helps to identify a biased client in FL without disclosing the data.

[1]School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland [2]School of Medicine, Yale, New Haven, CT, USA [3]Department of Electrical Engineering and Computer Sciences, UC Berkeley, Berkeley, CA, USA. Correspondence to: Klavdiia Naumova <klavdiia.naumova@epfl.ch>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

4. Finally, we propose a new approach to use inDISCO for interpretable personalization.

Related work. Explaining how neural networks make predictions is critical to ensuring trust in real-world use cases. This is particularly important in the medical imaging domain, for example, where evaluating the alignment of logical plausibility of a prediction helps protect patients against misdiagnosis (Ribeiro et al., 2016; Lundberg & Lee, 2017; Selvaraju et al., 2017; Simonyan et al., 2014; Singh et al., 2020; Shrikumar et al., 2017; Chen et al., 2021; 2019). There are many posthoc techniques that aim to explain the predictions made by black-box models. The most popular methods are LIME (Ribeiro et al., 2016), SHAP (Lundberg & Lee, 2017), Feature visualization [1], and saliency mapping with Grad-CAM (Selvaraju et al., 2017).

In opposition to black-box models, there exist inherently interpretable models whose decision-making process is made to be transparent. An example of such a model is a prototypical part neural network `ProtoPNet` (Chen et al., 2019) that bases classification on how similar *parts* of an image in the test set are to a *prototypical part* of an image in the train set (Summarized in Appendix, Fig 3). `ProtoPNet` showed performance comparable with the state-of-the-art black-box deep neural networks on a popular benchmark data set while being inherently interpretable.

In FL, clients may be imperfectly interoperable (IIO), where the biggest risk is the presence of systematic bias in one client, resulting in label leakage. For instance, a set of X-rays, where the diagnosis is written on the image. The biased features learned from this client would contaminate the feature pool in an FL setting and decrease the performance of a global model. One of the approaches to address this issue is adding local personalization layers to the model. This idea is presented in the works (Arivazhagan et al., 2019) (FedPer) and (Roschewitz et al., 2021) (iFedAvg). In particular, iFedAvg is also designed to identify and visualize the clients and features causing the shift through local feature-wise affine layers f_{in} and f_{out} which can learn the feature-wise differences between clients compared with the global model, without sharing any data. While this is interpretable for tabular data (where features are intelligible), the same is not true for images. We propose applying `ProtoPNet` to this IIO-FL setting to best isolate interpretable features from images that can be compared and interpreted.

2. Problem formulation

Context. We trained `inDISCO` in an FL setting using either IID (unbiased, identically distributed classes) or IIO (imperfectly interoperable with systematic bias in a single class) data distribution among clients. A central server aggregates and updates the models' parameters. By learning local prototypes, each client identifies the parts of its training images most important for the task. In contrast, the global prototypes show the relevance for all clients on average. Finally, by examining the difference between local and global prototypes, a client can identify and quantify the presence of bias in its own or another client's dataset. Optionally sharing local prototypes among clients can further attribute the origin of bias. Taken together this information can guide appropriate action, to avoid the negative effect of data bias on a global model, such as client selection or personalization by penalizing the learning of biased features.

Notation. Let us adapt the notation from (Chen et al., 2019). Given input $\mathbf{x}_n$, where $n \in \{1, ..., N\}$, each of N clients learns features with convolutional layers $f(\mathbf{x}_n)$ and m prototypes $\mathbf{P}_n = \{\mathbf{p}_{nj}\}_{j=1}^{m}$. Given a convolutional output $\mathbf{z}_n = f(\mathbf{x}_n)$, the j-th prototype of the n-th client's unit $g_{\mathbf{p}_j,n}$ in the prototype layer $g_{\mathbf{p}n}$ computes the squared L^2 distance between the prototype $\mathbf{p}_{nj}$ and all the patches of $\mathbf{z}_n$ and converts these distances into similarity scores. These scores are then multiplied by the weight matrix w_{hn} in the final fully connected layer h_n followed by softmax normalization to output class probabilities.

Local training. Given a set of training images $\mathbf{D}_n = \{(\mathbf{x}_{ni}, y_{ni})\}_{i=1}^{k}$, where k is a number of images per client, each client aims to minimize the following objective:

$$\min_{\mathbf{P}_n, w_{conv,n}} \frac{1}{k}\sum_{i=1}^{k} \text{CrsEnt}_n(h_n \circ g_{\mathbf{p}n} \circ f(x_{ni}), y_{ni}) + \\ + \lambda_1 \text{Clst}_n + \lambda_2 \text{Sep}_n, \quad (1)$$

where $w_{conv,n}$ denotes the weights of the convolutional layers learned by client n, CrsEnt is a cross-entropy loss that penalizes the misclassification, and the cluster and separation costs are defined as follows:

$$\text{Clst}_n = \frac{1}{k}\sum_{i=1}^{k} \min_{j:\mathbf{p}_{nj} \in \mathbf{P}_{ny_{ni}}} \min_{\mathbf{z}_n \in \text{patches}(f(x_{ni}))} ||\mathbf{z}_n - \mathbf{p}_{nj}||_2^2 \quad (2)$$

$$\text{Sep}_n = -\frac{1}{k}\sum_{i=1}^{k} \min_{j:\mathbf{p}_{nj} \notin \mathbf{P}_{ny_{ni}}} \min_{\mathbf{z}_n \in \text{patches}(f(x_{ni}))} ||\mathbf{z}_n - \mathbf{p}_{nj}||_2^2 \quad (3)$$

The minimization of the cluster cost (Clst) is needed to make each training image have a latent patch that is close to at least one prototype of the correct class. At the same time, every latent patch of a training image is separated from the prototypes of the incorrect class through the minimization of the separation cost (Sep). More detail ProtoPNet is found in Chen et al. (2019) and summarized in Appendix Fig3.

Federated update. At the global update step, the server performs simple averaging of all the local prototypes $\mathbf{P}_{loc} = \{\mathbf{P}_n\}_{n=1}^{N}$ and weights of the final layer

[1] https://distill.pub/2017/feature-visualization/

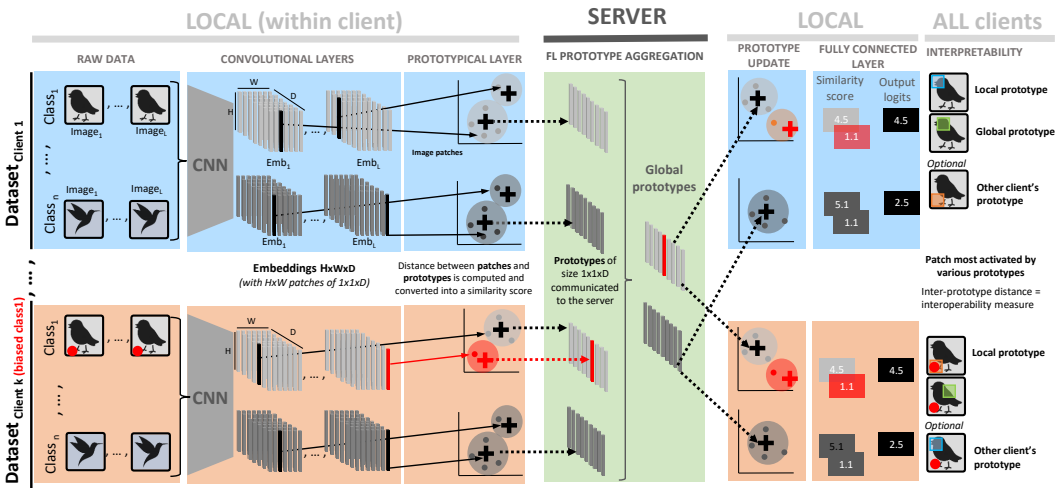

Figure 1. **inDISCO architecture**. Several clients ($client_1$,..., $client_k$) wish to learn a model in a federated setting via a SERVER. inDISCO, passes raw images through a CNN to create embeddings of size $[H \times W \times D]$ in the latent space $(Emb_1, ..., Emb_L)$, which are divided up into $H \times W$ image patches $[1 \cdot 1 \cdot D]$. Prototypical patches (**black**) are clustered and a prototype (+) is learned for each image. $Class_1$ of $Client_k$ has **systematic bias** which contaminates the prototype pool (+). Ten prototypes for each class are shared to the SERVER by each client and aggregated to make ten global prototypes. These are then pushed back to the clients. Classification is based on a similarity score computed between the prototype and the image patches. In the final panel, we see how global and local prototypes can be compared to directly visualize shifts without sharing any original data.

$\mathbf{W}_{h,loc} = \{w_{hn}\}_{n=1}^{N}$ to obtain the global parameters:

$$\mathbf{P}_{glob} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{P}_n \quad (4) \qquad \mathbf{W}_{h,glob} = \frac{1}{N}\sum_{n=1}^{N} w_{hn} \quad (5)$$

and then sends them back to clients as shown in Fig 1.

3. Experimental setup

3.1. Dataset

Our inDISCO model was trained and evaluated on CUB-200-2011 dataset (Wah et al., 2011) of 200 bird species from which we took 20 classes. Preprocessing was performed as described by Chen et al. (2019). We introduced class-specific bias for one clients by adding an emoji to the images of a particular class (Appendix, Fig. 4). The repository can be found here (link provided at submisssion).

3.2. Experiments

i. Centralized baseline. As a baseline, we follow the architecture of `ProtoPNet` (using the VGG19 (Simonyan & Zisserman, 2015) implementation pretrained on ImageNet (Deng et al., 2009)) to learn a centralized model with centralized prototypes (**CMCP**) on the whole dataset. We learned 10 prototypes of size $[1 \cdot 1 \cdot 128]$ per class.

ii. IID-FL local baseline. We made an IID partition of the data over four clients and trained local ProtoPNets with local prototypes for each (**LMLP**).

iii. IID-FL global baseline. Using the FL setup above, global models with global prototypes (**GMGP**) were trained according to the scheme depicted in Fig 1. The training is composed of six communication rounds between the clients and the server. The server initializes a ProtoPNet model and sends it to the clients. The clients learn LMLP. After 5 epochs, a subset of LP are communicated to the server and aggregated. Importantly, during this training stage, each client keeps the pretrained convolutional weights frozen and trains two additional convolutional layers. Each of the next communication rounds includes the following steps:

- **Local convolutional layers.** Each client trains convolutional layers locally (on their own dataset).
- **Local prototypes.** The local latent representation of each image is divided into image patches and a local prototypical patch of each image is used to learn a set of ten local prototypes $\mathbf{P}_{loc}$ and weights $\mathbf{W}_{loc}$, which are sent to the server after each ten epochs.
- **Global prototypes.** The server aggregates local prototypes by averaging to create a set of ten global prototypes $\mathbf{P}_{glob}$ and weights $\mathbf{W}_{h,glob}$. These are shared back to each client to iterate training.
- **Interpretability.** Each client visualizes interoperability shifts by projecting each prototype onto the nearest latent training patch from the same class and then optimize the final layer to improve accuracy.

After training, we have as many global models as clients. All these models have global prototypes and different $w_{conv,n}$ whose updates stayed locally. Importantly, we purposefully limit the global training of convolutional layers, for the purpose of comparing interoperabilty, thus the performance

of GMGP is expected to be lower than CMCP.

iv. IIO-FL Experiment. Finally, we trained **LMLP^b** and **GMGP^b** in an FL setting with three unbiased IID clients and one IIO client (with systematic bias in one class (Appendix Fig 4). We visually inspect the prototypes learned locally and globally to detect IIO shifts between clients without sharing any original data.

4. Results

4.1. IID setting.

The average accuracy for CMCP (i.e. `ProtoPnet` baseline), LMLP, and GMGP trained on unbiased IID data are presented in Table 1. As expected local models perform worse than a centralized model due to the smaller dataset. As the training of the global model is purposely restricted to highlight differences in clients' data, we do not expect the GMGP to significantly improve upon LMLP, which is the case. Indeed, only part of the GMGP network is updated globally (prototypes $\mathbf{P}_n$ and weights w_{hn}), while the convolutional layers are kept local. Nevertheless, the prototypes learned in these three settings are rather similar as can be seen in Appendix, Fig.5.

Table 1. **Centralized vs FL IID settings.** Classification accuracies for **CMCP** (centralized model/prototype), **LMLP** (local model/prototype), and **GMGP** (global model/prototype) trained without data bias. The mean computed over four clients are shown with standard deviation.

MODEL	CMCP	LMLP	GMGP
ACCURACY (% ±SD)	86.02	82.76 ±1.14	81.25 ±0.49

4.2. IIO-FL setting.

Bias injection into one client's dataset has a strong effect on model performance and prototypes. In this case, we compare models' accuracy separately on unbiased and biased data and in addition to average accuracy over all classes, we show accuracy for a biased class. Table 2 Both local LMLP^b and global GMGP^b expectedly perform much worse on unbiased data than on biased one. Notably, the local model gives 100.0% accuracy for a biased class on a dataset with emoji and 0.0% accuracy for this class on unbiased data. This clearly indicates that the model assigns this class based on the presence of bias and thus suffers catastrohpic failure in the absence of bias. Indeed, as shown in Figure 2, LMLP (trained on unbiased data) and LMLP^b activate totally different patches on the same test image with and without emoji.

Table 2. **Effect of IIO bias in FL.** Classification accuracies for local and global models trained in an FL setting which has a biased client. For each model, the value in the left subcolumn corresponds to the test set of a biased client, and in the right subcolumn, there is an average value over the test sets of unbiased clients with standard deviation where possible. Performance is shown from **bad** to **good**.

MODEL	LMLP^b		GMGP^b	
TEST SET	BIASED	UNBIASED	BIASED	UNBIASED
ALL CLASSES	85.8	76.1±1.7	83.3	75.3±1.2
BIASED CLASS	100.0	0.0	85.7	4.8±4.7

Communication of prototypes with other clients brings slight improvement to the model's performance. GMGP^b has 4.76% accuracy for biased class on unbiased data and 85.71% on biased one. From Figure 2, we can see that the global model for a biased client does not activate the emoji as its local counterpart but still looks at a less informative patch close to it. Global model for good clients, however, looks at the bird's head as it does the local one. The distance between these prototypes can not only be visualised, but also computed, meaning that it could be used to derive a personalization weight which is visually interpretable.

The negative effect of data bias is additionally demonstrated in Appendix Table 3. Here, we show how different models predict a class for biased and unbiased test images.

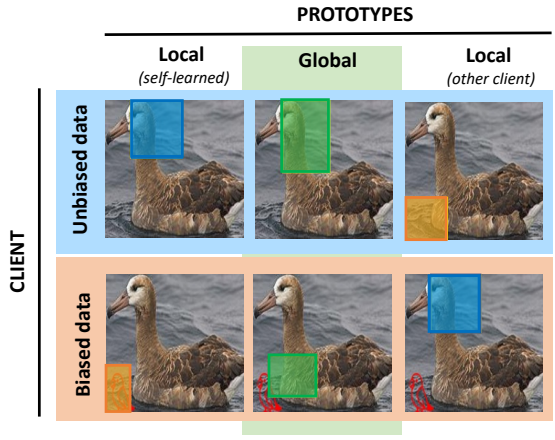

Figure 2. **Prototypes learned in IIO FL setting.** Examples of a test image with bounding boxes indicating the most activated patches by the prototypes learned locally and **globally** on **unbiased** and **biased** datasets in an IIO-FL setting.

5. Discussion

`inDISCO`, a novel extension of `ProtoPnet`, allows interpretable and privacy-preserving identification of data bias in FL. The bias can be visualized as well as quantified, which holds the potential for creating a visually interpretable personalization scheme. Thus, `inDISCO` provides transparency from black-box data without compromising privacy. Indeed, the only elements shared with the server are prototypes computed from a latent space representation. These shared elements can be tailored to various privacy budgets by varying the number of prototypes per class, the size of a prototype, and the number of communication rounds.

References

Arivazhagan, M. G., Aggarwal, V., Singh, A. K., and Choudhary, S. Federated learning with personalization layers. *arXiv:1912.00818*, 2019.

Chen, C., Li, O., Tao, C., Barnett, A. J., Su, J., and Rudin, C. This looks like that: Deep learning for interpretable image recognition. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 8930–8941, Red Hook, NY, USA, 2019. Curran Associates Inc.

Chen, H., Lundberg, S., and Lee, S.-I. *Explaining Models by Propagating Shapley Values of Local Components*, pp. 261–270. Springer International Publishing, Cham, 2021. ISBN 978-3-030-53352-6. doi: 10.1007/978-3-030-53352-6_24. URL https://doi.org/10.1007/978-3-030-53352-6_24.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255. IEEE, 2009.

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 4768–4777, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54. JMLR: W&CP, 2017.

Ribeiro, M. T., Singh, S., and Guestrin, C. "Why should I trust you?": Explaining the predictions of any classifier. KDD '16, pp. 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939778. URL https://doi.org/10.1145/2939672.2939778.

Roschewitz, D., Hartley, M.-A., Corinzia, L., and Jaggi, M. iFedAvg: Interpretable data-interoperability for federated learning. *arXiv:2107.06580*, 2021.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 618–626, 2017. doi: 10.1109/ICCV.2017.74.

Shrikumar, A., Greenside, P., and Kundaje, A. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *ICML'17*, pp. 3145–3153. JMLR.org, 2017.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at International Conference on Learning Representations*, 2014.

Singh, A., Sengupta, S., and Lakshminarayanan, V. Explainable deep learning models in medical image analysis. *Journal of Imaging*, 6, 2020.

Wah, C., Branson, S., Welinder, P., Perona, P., and S., B. The Caltech-UCSD Birds-200-2011 dataset. Technical report, California Institute of Technology, 2011.

A. Appendix

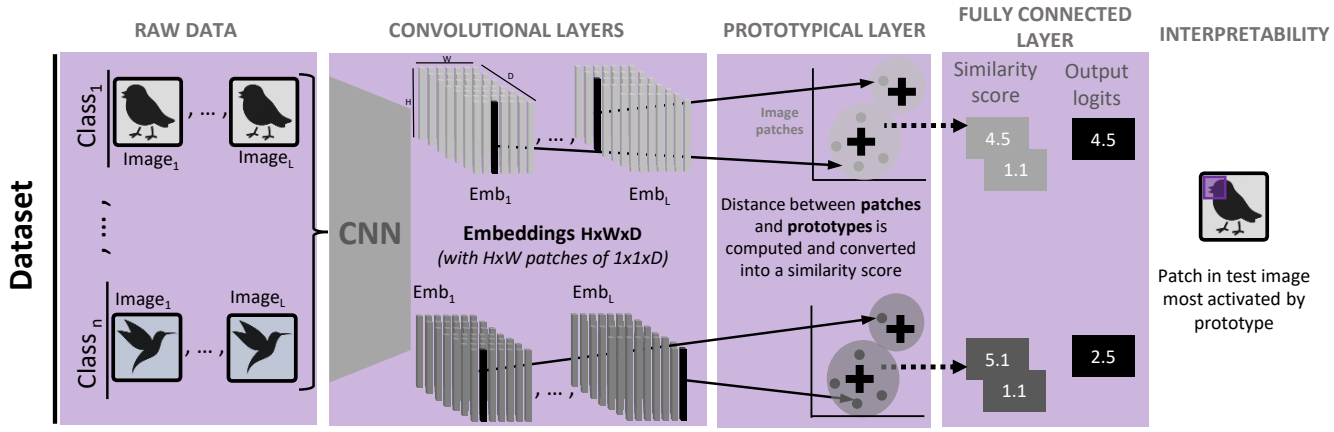

Figure 3. **`ProtoPnet` architecture**. This is a centralized setting with no clients. `ProtoPnet`, passes raw images through a CNN to create embeddings of size $[H \times W \times D]$ in the latent space $(Emb_1, ..., Emb_L)$, which are divided up into $H \times W$ image patches $[1 \cdot 1 \cdot D]$. Prototypical patches (**black**) are clustered and a prototype (**+**) is learned for each image. Classification is based on a similarity score computed between the prototype and the image patches. In the final panel, we see prototypes can be visualized to directly.

(a) Unbiased (b) Biased

Figure 4. Imperfectly interoperable images used in this work. The biased client **(b)** has a red emoji of a parrot in the lower left corner.

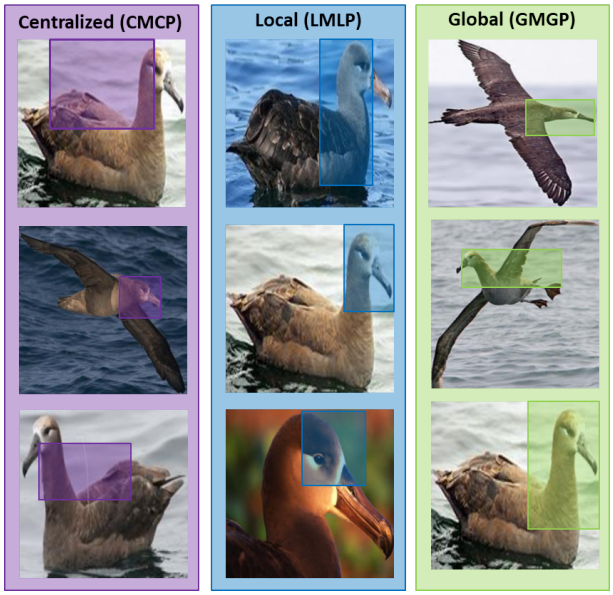

Figure 5. Examples of training images with bounding boxes indicating centralized, local, and global prototypes learned on unbiased data.

Table 3. Prediction results of local and global models for two biased and two unbiased test images coming from biased and unbiased during training classes.

MODEL	BIASED CLASS		UNBIASED CLASS	
	TEST IMAGE			
	UNBIASED	BIASED	UNBIASED	BIASED
LMLP	COR	COR	COR	COR
LMLP^b	INCOR	COR	COR	INCOR
GMGP	COR	COR	COR	COR
GMGP^b	INCOR	COR	COR	INCOR

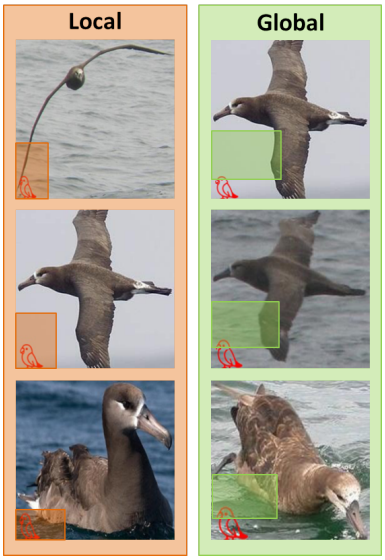

Figure 6. Examples of training images with bounding boxes indicating local and global prototypes learned on biased data.