My-This-Your-That—Interpretable Identification of Systematic Bias in Federated Learning for Biomedical Images

Klavdiia Naumova[1], Arnout Devos[1], Sai Praneeth Karimireddy[2], Martin Jaggi[1]
Mary-Anne Hartley*[1,3],
1 intelligent Global Health group, Machine Learning and Optimization Laboratory, Swiss Institute of Technology (EPFL), Lausanne, Switzerland
2 Berkeley AI Research Laboratory, University of California, Berkeley, CA, USA
3 Laboratory of Intelligent Global Health Technologies, Biomedical Informatics and Data Science, School of Medicine, Yale University, New Haven, CT, USA
*mary-anne.hartley@yale.edu

Abstract

Deep learning has the potential to improve biomedical image interpretation, making it more accessible and precise, particularly in low-resource settings where human experts are often lacking and prone to inter-reader bias. Privacy of medical data necessitates innovative methods such as DIStributed COllaborative learning (DISCO) that allows several data owners (clients) to learn a joint model without sharing data. However, this black-box data can conceal systematic bias, compromising model performance. This work adapts an interpretable prototypical part learning network to a DISCO setting allowing each client to visualize the differences in features learned by other clients on its own image: comparing one client's 'This' with others' 'That'. We present a setting where four clients collaborate to train two diagnostic classifiers on a chest X-ray dataset. In an unbiased case, the global model reaches 74.14 % balanced accuracy for cardiomegaly and 74.08 % for pleural effusion. We then compare performance under the strain of systematic visual bias, where a confounding feature is associated with the label in one client. The performance of global models validated on unbiased clients drops to near random. We demonstrate how differences between local and global prototypes can indicate the presence of bias and allow it to be visualized on each client's data without compromising privacy. We further show how these differences can guide model personalization.

Introduction

The transformative force of deep learning on clinical decision-making systems is being increasingly documented [1–3]. For medical images, these advances have the potential to improve and democratize access to high-quality standardized interpretation, extracting predictive features at a granularity previously inaccessible to human experts who are, in any case, often lacking in low-resource settings and prone to inter-reader bias. The potential to automate routine analysis of medical records [3] and help find hidden predictive patterns in the data that may reduce errors and unnecessary interventions (for example biopsy) [4] moves us towards more efficient, personalized, and accessible healthcare.

However, the performance of these models relies on large accessible carefully curated centralized databanks, which is often hard to achieve in practice. Rather, medical data are usually distributed among several institutions that are unable to share due to a range of well-considered reasons. Distributed collaborative (DISCO) learning has emerged as a solution to this issue, offering privacy-preserving collaborative model training without sharing any original data. Here, instead of sending the data to a central model, the model itself is distributed to the data owners to learn in situ, updating a global model via privacy-preserving gradients. When a central server is used, the technique is known as federated learning (FL) [5] and it has already been shown to hold potential for various medical applications [6–9].

While DISCO addresses the issue of data privacy, it comes at a cost to transparency resulting in clients learning blindly from their peers. In this *black-box data* setting, hidden biases between clients of the federation can make generalization challenging, and even when there are no biases present, its risk may degrade trust. Coupled with the already poor transparency of deep learning architectures used for medical images (aka black-box models), interpretability is becoming a critical

feature to ensure a balance in the trade-off between transparency and privacy that will encourage implementation.

Specifically required, is an approach to inspect data *interoperability* between clients as well as provide insights into the most predictive features. Additionally, this approach should be adept at detecting and quantifying concealed biases in the data in an interpretable way, while preserving clients' privacy. For instance, if a model is using the hospital logo on an X-ray to diagnose TB because TB patients are generally treated in that hospital, this should be considered a proxy feature and an example of systematic bias.

As for the black-box neural networks, numerous approaches try to *explain* them by a posthoc analysis of their predictions [10–16]. Many of these methods are well presented in a collection [17] and in the popular book "Interpretable Machine Learning" [1] written by Christoph Molnar. Feature visualization [2] and saliency mapping with Grad-CAM [12] are among the most popular techniques. These methods visualize the regions or concepts in data that are most important for a particular network to make a prediction, however, they fail to interpret this network, i.e., tell us why or to what extent the visualized regions are essential for a prediction [18].

We claim that achieving interpretability and exploring data interoperability in FL can be possible with the help of inherently interpretable models (IM). Unlike their black-box counterparts, IM's decision-making component is transparent by design [19, 20], for example, sparse logical models such as decision trees and scoring systems. For image recognition tasks, models that possess human-friendly reasoning based on a similarity between a test instance and already known instances (e.g. nearest neighbors from a training set or the closest samples from a set of learned prototypes) are extremely promising. This learning approach opens possibilities for incorporating domain experts' control to debug the models and examine the quality of training data. A popular example of such a model is a prototypical part learning neural network (`ProtoPNet`) developed by Chen et al. [21].

We summarize their implementation in SI Fig. 1. It first uses a set of convolutional layers to map input images to a latent space followed by a prototype layer which learns a set of prototypical parts from encoded training images to best represent each class. Classification then relies on a similarity score computed between these learned prototypes and an encoded test image. A prototype can be visualized by highlighting a patch in an input image which is the closest in terms of squared L_2-distance to this prototype in a latent space. The performance of `ProtoPNet` was demonstrated on the task of bird species identification. The model showed an accuracy level comparable with the state-of-the-art black-box deep neural networks while being easily interpretable. `ProtoPNet` was further extended to perform classification of mass lesions in digital mammography [22] and image recognition with hierarchical prototypes [23].

We see the potential of case-based reasoning models such as `ProtoPNet` for investigating imaging data compatibility in FL. In this work, we develop an approach called `inDISCO` (INterpretable DIStributed COlaborative learning) through the adaptation of `ProtoPNet` to FL and demonstrate its bias-identification ability on medical imaging data. Our idea is that prototypes learned on each client's local data represent features important for a particular class from each client's *point of view*. As summarized in Fig 1, clients learn local prototypes separately and send them to a server that aggregates and averages local prototypes to obtain global ones and sends them back to clients. The patches most activated by each of these two types of prototypes can be visualized and compared on each client's local test set without a need to share the data. By comparing global and local prototypes the clients can assess the interoperability of the data and directly examine the predictive impact of their peers in federation without compromising clients' privacy. To the best of our knowledge, this work is the first attempt at creating an interpretable methodology to inspect the interoperability of biomedical imaging data in FL.

Our main contributions are as follows:

1. We introduce `inDISCO` adapting `ProtoPNet` to a federated setting to enable privacy-preserving identification of data bias in FL.

2. We formalize a set of use cases for interpretable distributed learning on imperfectly interoperable

[1] https://christophm.github.io/interpretable-ml-book/

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

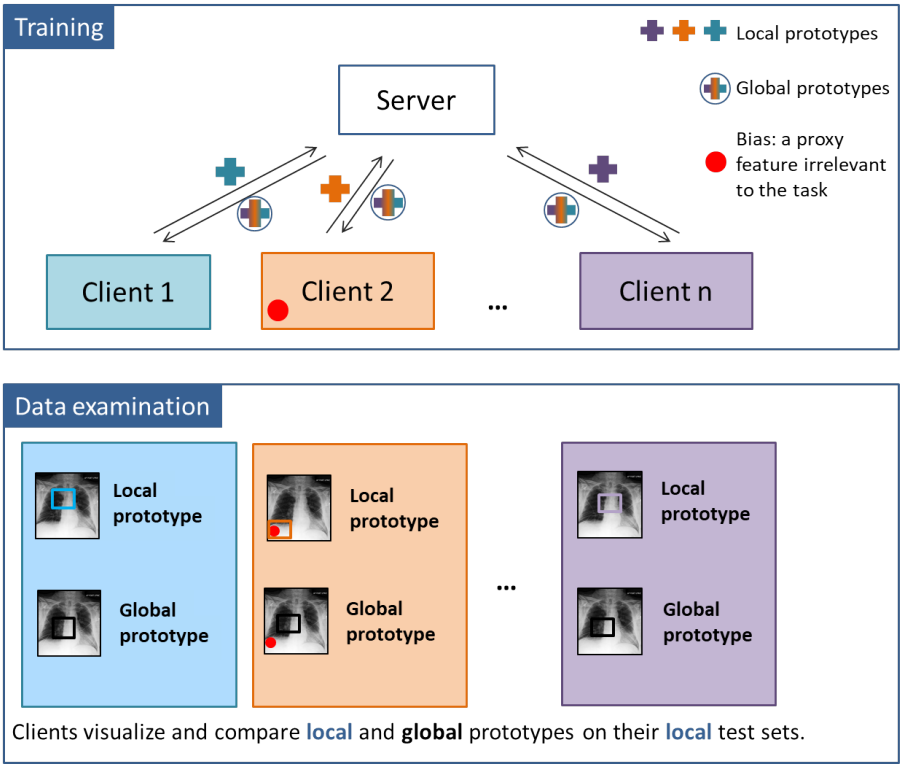


Figure 1: **Schematic representation of the inDISCO approach.** Within one communication round, each client learns **local** prototypes on its local training set and shares them with a server that aggregates and averages local prototypes from all clients and sends these new **global** prototypes back to clients. After several communication rounds, each client can examine the global data locally by visualizing and comparing **local** and **global** prototypes on its private local test set. Possible hidden bias in the federation will result in a large difference between **local** and **global** prototypes.

biomedical image data containing hidden bias.

3. We demonstrate the performance of our `inDISCO` approach on a benchmark dataset of human X-rays and compare it to baseline models.

4. We show how `inDISCO` helps to identify a biased client in FL without disclosing the data.

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

Materials and methods

Model description

The `ProtoPNet` architecture is presented in SI Fig. 1. The network is composed of the following parts:

- a set of convolutional layers to learn features from the input data;
- two additional 1×1 convolution layers with D channels and the ReLU activation after the first layer and Sigmoid after the second one;
- a prototype layer with a predefined number of prototypes. Each prototype is a vector of size $1 \times 1 \times D$ with randomly initialized entries;
- a final fully connected layer with the number of input nodes equal to the number of prototypes and the number of output nodes corresponding to the number of classes. The weights indicate the importance of a particular prototype for a class. They are initialized as in [21] such that

the connection between the prototypes and their corresponding class is 1 and -0.5 for the connections with the wrong classes.

We trained `inDISCO`, an FL adaptation of `ProtoPNet`, using either unbiased identical data distribution among clients (unbiased setting) or imperfectly interoperable with systematic bias in a single class (biased setting). Two parameter aggregation schemes were applied: (1) a central server aggregates and updates local parameters of all the layers of `ProtoPNet` or (2) only the prototypes and weights of the final fully connected layer. In the second case, the parameters of feature learning layers always stay local which results in *personalized* models with global prototypes. A detailed scheme is shown in Fig. 2

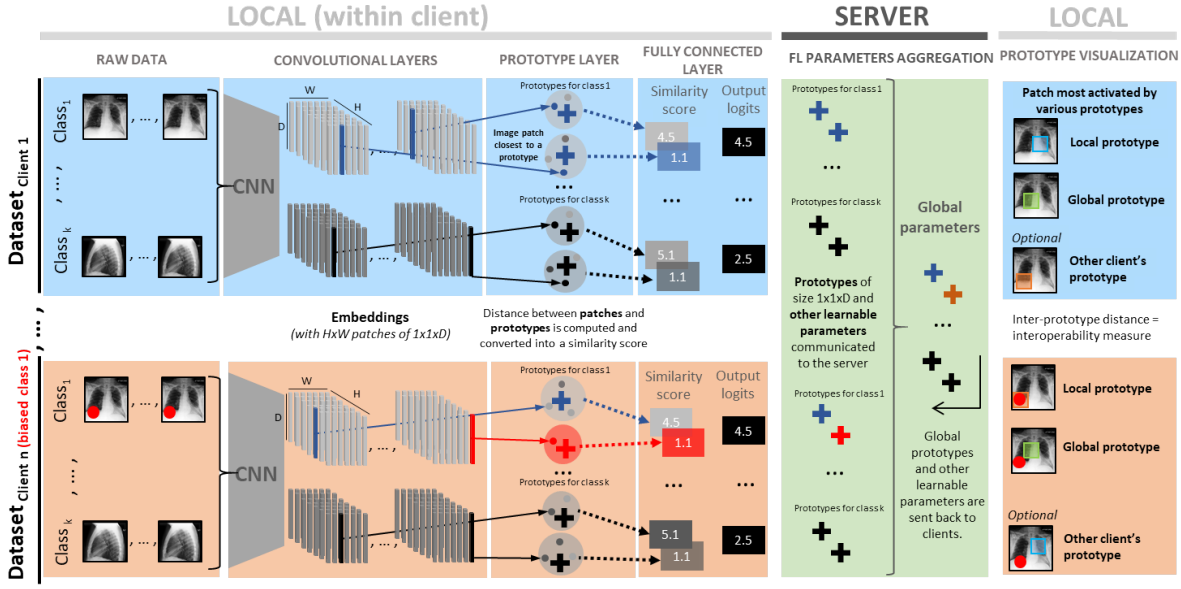


Figure 2: **inDISCO architecture.** Several clients ($client_1$,..., $client_n$) wish to learn a model in a federated setting via a SERVER. `inDISCO` passes raw data through a CNN to create embeddings in a latent space, each of which can be seen as $H \times W$ image patches $[1 \times 1 \times D]$. These patches are clustered around the closest prototypes (+) which are being learned for each class in the prototype layer. The prototype is a vector representing a class-characteristic feature in the latent space. $Class_1$ of $Client_n$ has **systematic bias** which contaminates the prototype pool (+). Prototypes for each class and other learnable parameters of the network are shared to the SERVER by each client and aggregated to make global parameters. These are then sent back to the clients. Classification is based on a similarity score between the prototypes and the patches of an encoded image. In the final panel, we see how global and local prototypes can be compared without sharing any original data.

By learning local prototypes, each client identifies the features in 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 bias in its own or another client's dataset.

Experimenting with two different parameter aggregation schemes allows us to investigate the trade-off between the bias-revealing and privacy-preserving properties of `inDISCO`.

Notation. Hereafter, we denote matrices and vectors in bold capital and bold lowercase letters, respectively.

Data split. Each client $n \in \{1, ..., N\}$ has a training set $\mathbf{D}^n$ of size l which consists of training images $\{\mathbf{X}_i\}_{i=1}^{l}$ and their corresponding classes $\{y_i\}_{i=1}^{l}$.

Model. Each client learns features with convolutional layers and m local prototypes $\mathbf{P}^n = \{\mathbf{p}_j\}_{j=1}^{m}$ of size $[1 \times 1 \times D]$ with a fixed number of prototypes per class. First, given an input image $\mathbf{X}_i$, the convolutional layers produce an image embedding $\mathbf{Z}_i$ of size $[H \times W \times D]$ which can be represented as $[H \times W]$ patches of size $[1 \times 1 \times D]$. Then the prototype layer computes the squared L^2 distance between each prototype $\mathbf{p}_j$ and all the patches in the image embedding $\mathbf{Z}_i$. This results

in m distance matrices of size $[H \times W]$ which are then converted into matrices of similarity scores (activation matrices) and subjected to a global max pooling operation to extract the best similarity score for each prototype. These final scores are then multiplied by the weight matrix $\mathbf{W}_h^n$ in the final fully connected layer h followed by softmax normalization to output class probabilities.

Training. The details of local training can be found in [21] and in SI Local training description. At the global update step, the server aggregates the local prototypes $\mathbf{P}^n$, weights of the final layer $\mathbf{W}_h^n$, and in the first aggregation scheme also the parameters of the convolutional layers $\mathbf{W}_c^n$ from each client n and performs simple averaging of these parameters to obtain the global ones:

$$\mathbf{P}_{glob} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{P}^n \quad (1) \qquad \mathbf{W}_{h,glob} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{W}_h^n \quad (2) \qquad \mathbf{W}_{c,glob} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{W}_c^n \quad (3)$$

and then sends them back to clients as shown in Fig. 2.

To visualize the prototypes, each client finds for each of the local and global prototypes a patch among its training images from the same class that is mostly activated by the prototype. It is achieved by forwarding the image through the trained `ProtoPNet` and upsampling the activation matrix to the size of the input image. A prototype can be described as the smallest rectangular area within an input image that contains pixels with an activation value in the upsampled activation map equal to or greater than the 95th percentile of all activation values in that map [21].

Data

The experiments were conducted on the CheXpert dataset [24], a large public dataset of 224,316 chest X-rays of 65,240 patients collected from Stanford Hospital. Each image was labeled by radiologists for the presence of 14 observations as positive, negative, or uncertain. To simplify the experiments and results interpretation, we decided to stick with a *one-vs-rest* binary setting. Specifically, we use images with positive labels for classes Cardiomegaly or Pleural effusion as the positive class and all other images as the single negative class. Cardiomegaly is a health condition characterized by an enlarged heart, and pleural effusion is an accumulation of fluid between the visceral and parietal pleural membranes that line the lungs and chest cavity. This setting, however, resulted in a data imbalance (7 and 1.6 times for cardiomegaly and pleural effusion, respectively). To address this issue, we decreased the size of a negative class in the training set by undersampling to make it equal to the size of a positive class. The final training sets had 48,600 and 37,088 images for cardiomegaly and pleural effusion classification, respectively. The validation sets were left imbalanced.

We used two ways of creating a biased dataset for one of the clients:

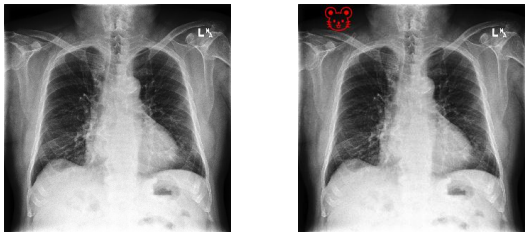
- **synthetic**: adding a small emoji to a positive class (Fig. 3);
- **real-world**: adding chest drains to a positive class as a more real-world bias (Fig. 4). To achieve this, we replaced images in a class Pleural effusion with X-rays labeled for the presence of chest drains [25].

The real-world use case can arise as pleural effusions are often drained. Drain positions are routinely checked with a post-insertion X-ray. Thus, a model may learn to diagnose pleural effusion by detecting a chest drain, rather than the pathology.

Experimental details

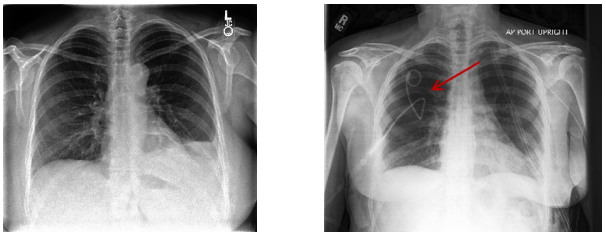
For both cardiomegaly and pleural effusion classification tasks, we first trained and evaluated a baseline centralized `ProtoPNet` which we denote as **C**entralized **M**odel (**CM**). Then we made an IID partition of the data over four clients and trained **L**ocal (**LM**), **G**lobal (**GM**), and **P**ersonalized (**PM**) models. Finally, we introduced systematic bias to one client's dataset and trained $\mathbf{LM}^{\mathbf{b}}$, $\mathbf{GM}^{\mathbf{b}}$, and $\mathbf{PM}^{\mathbf{b}}$ models where superscript *b* denotes a setting with one biased and three unbiased clients. The training details are described below.

Unbiased setting



(a) Unbiased (b) Biased

Figure 3: **Examples of unbiased and biased (imperfectly interoperable) images from CheXpert dataset for class Cardiomegaly.** The biased client **(b)** has a red emoji of a mouse in the upper left corner.



(a) Unbiased (b) Biased

Figure 4: **Examples of unbiased and biased (imperfectly interoperable) images from CheXpert dataset for class Pleural effusion.** The biased client **(b)** has a chest drain indicated by an arrow.

1. **Centralized** `ProtoPNet`**.** As a baseline, we follow the architecture and optimization parameters from the `ProtoPNet` paper [21] using the DenseNet [26] convolutional layers pretrained on ImageNet [27] to learn a CM on the whole dataset. We used ten prototypes of size $1 \times 1 \times 128$ per class. We report **balanced** average validation accuracy due to the validation set imbalance:

$$Balanced\ accuracy = \frac{Sensitivity + Specificity}{2} \tag{4}$$

2. **Local models.** We trained and evaluated LM for each of the four IID clients.

3. **Global models.** Using the first FL setup where the server aggregates parameters of all the layers, GM were trained according to the scheme depicted in Fig 2. The training is composed of three (for pleural effusion) or four (for cardiomegaly) communication rounds between the clients and the server. The server initializes a `ProtoPNet` model and sends it to the clients who learn LM. After five epochs, a subset of local parameters is 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 training.** Each client trains convolutional layers, prototype layer, and final fully connected layer locally on its own dataset.
   - **Local parameters.** A set of local prototypes $\mathbf{P}^n$, weights $\mathbf{W}_h^n$ and $\mathbf{W}_c^n$ is sent to the server after every ten epochs.
   - **Global parameters.** The server averages local parameters to create a set of global prototypes $\mathbf{P}_{glob}$, weights $\mathbf{W}_{h,glob}$ and $\mathbf{W}_{c,glob}$. These are shared back to each client to iterate training.

4. **Personalized models.** We used the second FL setup within which the server aggregated only the prototypes $\mathbf{P}^n$ and weights of the final fully connected layer $\mathbf{W}_h^n$ to train PM. We followed the same communication technique as described for GM. However in this case, after receiving the updated prototypes $\mathbf{P}_{glob}$ and weights $\mathbf{W}_{h,glob}$ from a server, each client performs an additional prototype update locally by finding the nearest latent training patch from the

same class and assigning it as a prototype. This operation is known as prototype *push* in [21] and we use it to adapt global prototypes to a local dataset for personalization. This step is followed by local optimization of the final layer to improve accuracy.

Biased setting

5. **Local, global, and personalized models.** We trained **LM**$^{\mathbf{b}}$, **GM**$^{\mathbf{b}}$, and **PM**$^{\mathbf{b}}$ models in an FL setting with three unbiased clients and one with systematic bias in one class (Fig. 3, 4). This setting is schematically shown in Fig. 5.

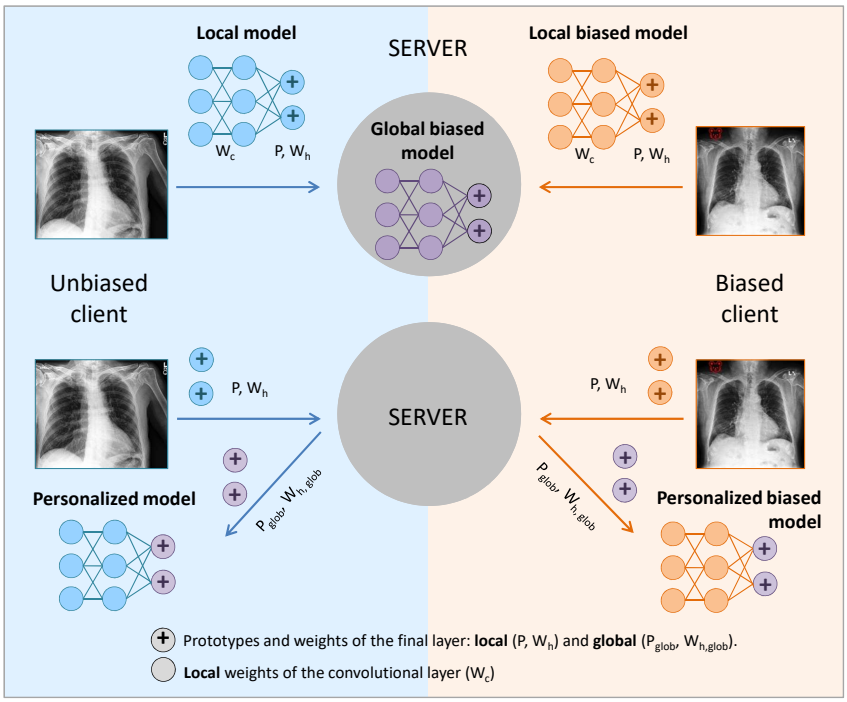


Figure 5: **Schematic representation of the inDISCO training in a biased setting.** To train a global biased model (**GM**$^{\mathbf{b}}$) the clients send to the server parameters of the convolutional layers ($\mathbf{W}_c$), prototypes (**P**), and weights of the final fully connected layer ($\mathbf{W}_h$), i.e. their entire local models (**LM** and **LM**$^{\mathbf{b}}$). To train personalized models (**PM** and **PM**$^{\mathbf{b}}$), the server aggregates and updates only the prototypes and weights of the final layer and sends these global updates ($\mathbf{P}_{glob}$ and $\mathbf{W}_{h,glob}$) back to clients to finalize the training of the personalized models. In this case, the parameters of the convolutional layers always remain local.

We visually inspect the prototypes learned locally and globally to detect the differences between clients' data without sharing them.

Statistical analysis

For each of the models described above, we present the uncertainty in terms of standard deviation. It was computed over three runs with different seeds. For LM and GM, the final performance was also averaged over four datasets (clients).

Results

Quantitative results

Unbiased setting

The values of balanced accuracy for CM (i.e. `ProtoPNet` baseline), LM, GM, and PM trained on unbiased data are presented in Table 1. CM gives 74.45 % and 75.95 % balanced accuracy for cardiomegaly and pleural effusion classification, respectively. As expected, local models perform worse than centralized ones due to the smaller dataset: LM achieve 71.64 % for cardiomegaly and 70.66 % for pleural effusion classes. Now, when the clients train `ProtoPNet` in collaboration, its performance improves: GM achieves 74.14 % of balanced accuracy for cardiomegaly and 74.08 % for pleural effusion classes which are close to the values achieved by the corresponding CM. Personalized models, however, demonstrate worse performance of 63.74 % and 63.76 % for cardiomegaly and pleural effusion classes, respectively, which may be the consequence of exchanging only a part of the network: prototypes $\mathbf{P}^n$ and weights $\mathbf{W}_h^n$. The values of classification sensitivity and specificity used to compute balanced accuracy can be found in SI Table 1.

Table 1: **Centralized vs FL unbiased settings.** Classification balanced accuracies (%, $\pm$SD) for **CM** (centralized model), **LM** (local model), **GM** (global model), and **PM** (personalized model) trained without data bias on CheXpert dataset for cardiomegaly and pleural effusion classes. The uncertainty is computed over three runs with different seeds and averaged over four datasets where applicable.

Model	CM	LM	GM	PM
Cardiomegaly	74.45 $\pm$0.73	71.64 $\pm$1.05	74.14 $\pm$0.77	63.74 $\pm$4.45
Pleural effusion	75.95 $\pm$0.68	70.66 $\pm$2.40	74.08 $\pm$2.24	63.76 $\pm$2.01

Biased setting

In this case, we compare models' performance separately on unbiased and biased data (Table 2). Recall that we used two different types of bias. Injected bias as a small emoji was added to the cardiomegaly class of one client's data. A chest drain bias was used for the pleural effusion class. It is clear that both of these two types of data poisoning have a large effect on models' performance.

We can see that models with large local contribution, LM^b and PM^b, give 100.0 % and 89.80 % test accuracy, respectively, on biased data and 50.0 % on the unbiased one in the case of cardiomegaly classification. Thus, these models strongly rely on the presence of bias to predict a positive class.

Since the chest drain bias is more difficult to learn than a small emoji, for pleural effusion classification, LM^b and PM^b do not achieve maximum accuracy on biased data but instead 73.22 and 64.81 %, respectively. At the same time, their performance on the unbiased test set is as low as for the cardiomegaly class, namely 50.37 and 49.87 % for LM^b and PM^b, respectively.

Fully global models (GM^b), trained via communication of all the learnable parameters of `ProtoPNet`, demonstrate a performance different from their local and personalized versions. For cardiomegaly, GM^b achieves 61.53 and 55.85 % of balanced accuracy on biased and unbiased sets, respectively. For pleural effusion, the model archives nearly 50 % on both test sets. Sensitivity and specificity for the biased setting are shown in SI Table 2.

Qualitative results

The quantitative performance of the models described above can be further supported in a visually interpretable way with the help of learned prototypes.

The examples of prototypes visualized on training sets for the models trained in the unbiased setting are shown in Fig. 6. We can see that these prototypes represent class characteristic features clear for humans. For example, in order to classify an image as cardiomegaly, a centralized model looks at the whole enlarged heart (Fig. 6) or at the collarbone level in the center pointing out the

Table 2: **Effect of bias in FL.** Classification balanced accuracies (%, ±SD) for LM^b, GM^b, and PM^b trained in an FL setting with one biased client on the CheXpert dataset for cardiomegaly and pleural effusion classes. 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. The uncertainty is computed over three runs with different seeds and averaged over four datasets where applicable. Performance is shown from **bad** to **good**.

Model	LM^b		GM^b		PM^b	
Test set	Biased	Unbiased	Biased	Unbiased	Biased	Unbiased
Cardiomegaly	100.0±0.0	50.0±0.0	61.53±4.27	55.85±3.69	89.80±10.20	50.0±0.0
Pleural effusion	73.22±1.16	50.37±0.38	49.72±0.28	50.01±0.01	64.81±0.99	49.87±0.10

extended aorta characteristic for this condition (SI Fig. 2). As for the pleural effusion classification, most prototypes activate the lower part of the lungs where fluid accumulates in this disorder. More examples of the prototypes learned in the unbiased and biased settings can be found in SI Fig. 2 - SI Fig. 15.

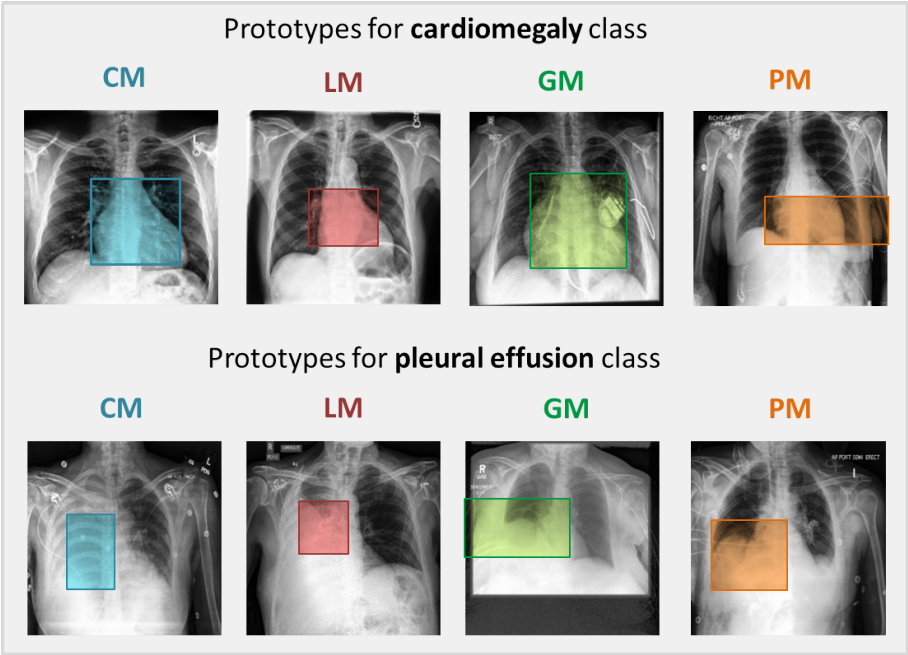


Figure 6: **Prototypes learned in an unbiased setting.** Examples of prototypical parts learned by CM, LM, GM, and PM in an unbiased setting and visualized on a corresponding training set.

To demonstrate the effect of data bias, we compare the models on *test* images by finding a patch mostly activated by the prototypes learned in the FL settings with three unbiased and one biased client (Fig. 7, 8).

We see that local and personalized models of an unbiased client *look at* a meaningful class-characteristic patch in both biased (Fig. 7 and 8: second row last column) and unbiased (Fig. 7 and 8: first row first column) images to reason their predictions.

In the case of a biased client, the local model (LM^b) for cardiomegaly classification (Fig. 7: second row first column) *looks at* the emoji in the upper left corner of a test image. It tends to search for it in the unbiased image as well (Fig. 7: first row last column). The neighborhood of this injected bias turned out to be the most activated patch for the personalized model (PM^b, Fig. 7: second row third column). This result explains the 100% accuracy of LM^b and 89.80% accuracy for PM^b on a biased test set and their complete failure on an unbiased one.

In the pleural effusion class, LM^b and PM^b indeed rely on the presence of a chest drain in an X-ray image as we can see from the most activated prototypes (Fig. 8: second row first and third columns).

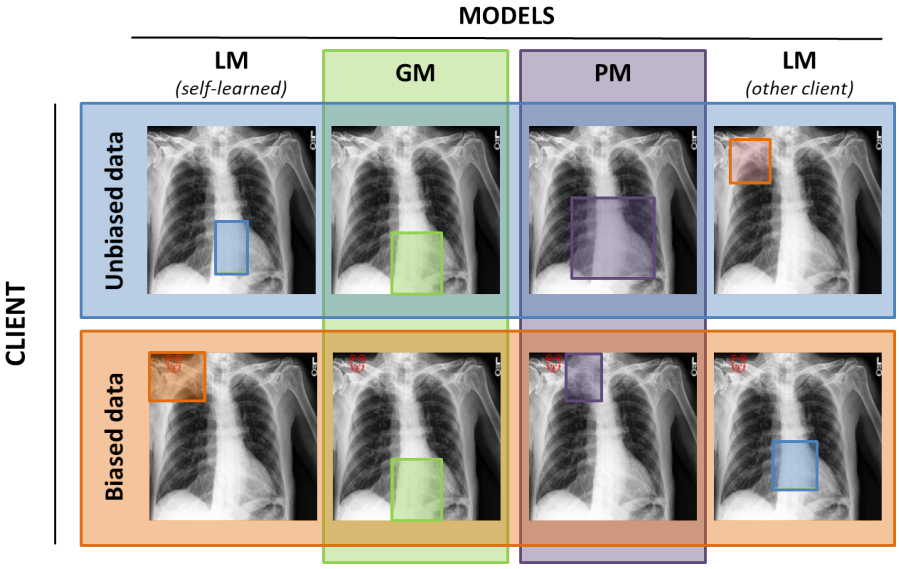


Figure 7: **Bias identification with `inDISCO`.** Examples of a test image with bounding boxes indicating the most activated patches by the prototypes learned locally and globally on **unbiased** and **biased** CheXpert datasets in an FL setting for *cardiomegaly* classification. The difference between local (LM) and global (GM/PM) prototypes signals about poor data interoperability in the federation.

As for the fully global models trained in the federated setting with one biased client (GM^b), there is a difference in their behavior depending on the type of bias used. Injected bias (an emoji), applied to the cardiomegaly class, did not have an effect on the global prototypes: they still activate the heart in both biased and unbiased images (Fig. 7: second column). For pleural effusion, however, with a more realistic chest drain bias, the global prototypes seem to be affected strongly by the biased client's training set since they tend to activate the upper part of an image instead of the bottom of the lungs where the fluid usually accumulates in the pleural effusion condition (Fig. 8: second column).

We demonstrated that prototypes learned by `ProtoPNet` are sensitive to data bias and thus can help to create a visually interpretable approach to explore data interoperability in FL in a privacy-preserving way. We discuss this possibility in the next section.

Discussion

Data compatibility between the clients in FL is of utmost importance for training an efficient and generalizable model. In this work, we present a visually interpretable approach for bias identification in FL that leverages a prototypical part learning network. A scheme to identify an incompatible client can be approximated as follows:

1. Each client trains a local `ProtoPNet` (LM) on its own data set.
2. With the help of a central server, the clients train global models (GM and/or PM) sharing all or a portion of learnable parameters (e.g. only the prototypes and weights of the final layer).
3. Each client visualizes its *local*, *global*, and *personalized* prototypes (finds the most activated patches) on its local test set and compares them by means of simple visual inspection (ideally with the help of domain experts). There is no need to share a test set with other clients or the server.
4. A large difference between local and global/personalized prototypes for certain clients indicates a possible data bias in the federation and requires the clients to either quit the federation or take measures to improve the quality of their training data.

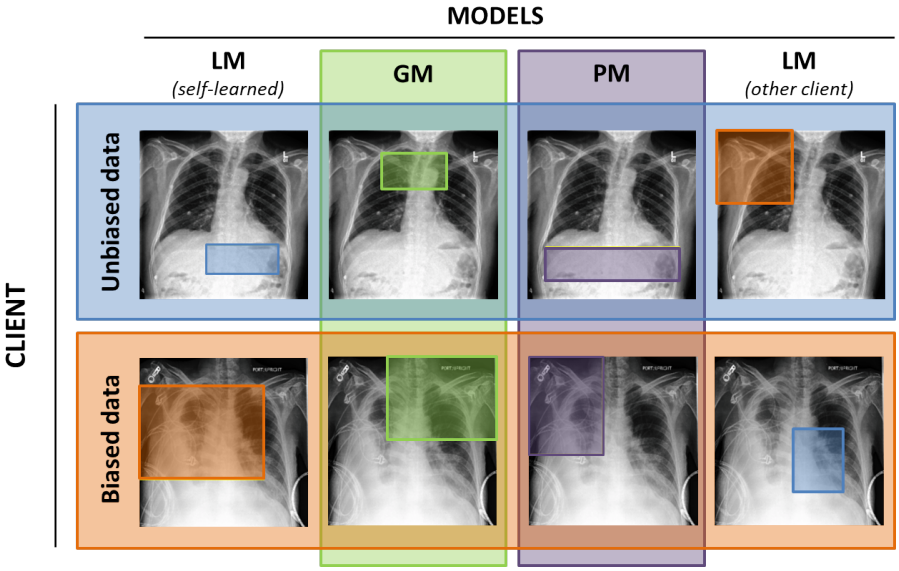


Figure 8: **Bias identification with** `inDISCO`**.** Examples of a test image with bounding boxes indicating the most activated patches by the prototypes learned locally and globally on **unbiased** and **biased** CheXpert datasets in an FL setting for *pleural effusion* classification. The difference between local (LM) and global (GM/PM) prototypes signals about poor data interoperability in the federation.

We demonstrated this scheme on a task of binary classification of X-ray images for the presence of cardiomegaly and pleural effusion conditions using two different data poisoning patterns. As can be seen from Fig. 7, simple injected bias such as an emoji in the cardiomegaly class easily confuses the local model making it spuriously rely on this emoji to predict a positive class. It is interesting to note that for this binary classification task, adding bias to a positive class also changes the prototypes for a negative class. This effect can be seen in SI Fig. 4 and SI Fig. 8, where prototypical parts for a negative (unbiased) class turned out to be left upper regions where there was an emoji for a positive class. Obviously, these prototypes have no practical value in classifying cardiomegaly.

Training a model via averaging the parameters over all clients helps to level out the effect of the bias completely (GM) or to a smaller extent (PM). This apparent difference between local and global/personalized prototypes should alarm the data owner of possible discrepancies between their data and others. From the unbiased clients perspective, since the difference between the prototypes for them is negligible, a drop in the performance of a GM in comparison to LM and larger uncertainty values are a sign of poor data interoperability in the federation.

To experiment with more practically relevant data bias, we paid attention to the fact that often patients with pleural effusion get chest drains to remove the fluid from their lungs. Thus the presence of chest drains in X-ray images can serve as bias for pleural effusion class. We trained our models in the FL setting where one client has images with chest drains in the positive class (note that these images do not necessarily have actual pleural effusion).

Fig. 8 shows a possible output of applying our `inDISCO` on a pleural effusion classification task in a biased setting. As before, an LM^b fails to activate a class-relevant feature, namely the bottom region of the lungs as an unbiased model does, and instead *looks at* the upper part of the chest where there are lots of drains. The same result was observed for PM^b. It is interesting, that the fully global model also activates the upper part of a test image in both biased and unbiased samples. Unlike the cardiomegaly classification, in this case, the data incompatibility is clearer for unbiased clients than for the biased one. Indeed, in the cardiomegaly classification task, only one client has a systemic bias, while in the pleural effusion case, chest drains may naturally present in the images of other clients as well. This practically possible data distribution causes the dominance of chest drain among the positive class prototypes of a global model for all the clients and seriously worsens the overall model performance.

As we mentioned in the Experimental details section, trying two different ways of parameter

aggregation allows us to investigate a trade-off between privacy and ease of bias identification. Obviously, the more parameters clients share the higher the risk of having privacy leakage. At the same time, GM^b trained via aggregating all learnable parameters of `ProtoPNet` demonstrates a large difference between local and global prototypes in case of the presence of data bias in the federation facilitating the identification of this bias. PM^b, trained by centrally updating only the prototypes and weights of the final layer, make it more challenging for an adversary to get the data from such a small set of network parameters but have less bias-identification power: due to a large local contribution, the difference between local and personalized prototypes is small.

In this work, we presented two extreme cases of parameter aggregation. More experiments are needed to define an optimal amount of parameters to share. Note, however, that this potential amount is not strict and depends on a certain data sensitivity to privacy. Therefore, it is up to clients to set their *privacy budget*, i.e. how many network parameters they are ready to share.

Optionally, clients can also share their *local* models with each other to visualize them on other clients' data for additional comparison. An example of such a possibility is shown in Fig. 7 and 8 in the last column. We can see a large difference between the local prototypes learned on biased and unbiased data for both cardiomegaly and pleural effusion classes.

So far, we have been talking about data bias from a negative perspective. However, it is possible to have large heterogeneity among the clients meaning that some specific features that each of them has are important. For instance, skin color which varies across continents may be an essential feature for predicting dermatological pathologies. In this case, training PM allows clients to benefit from the federation while keeping their specific features essential for the prediction (see Fig. 7 and 8: second row third column).

Limitations and future work

To investigate the trade-off between privacy and bias-identification ability of our `inDISCO` approach, further studies are required. It is also necessary to experiment with other medical datasets and real-world biased settings with a larger number of clients.

We have presented a promising technique to identify data incompatibility in FL. A possible next step is to introduce a debiasing option to our approach that will allow us to instantly penalize the contribution of a biased client. It may be done, for example, automatically through prototypes weighing or manually with the help of domain experts.

Finally, we aim at adapting our `inDISCO` approach to a web-based DISCO application[3]. It provides a user-friendly framework for distributed learning and thus has the potential to facilitate the integration of `inDISCO` into medical practice.

Conclusion

`inDISCO` is a novel extension of `ProtoPNet`, which allows interpretable and privacy-preserving identification and attribution of data bias in federated learning for imaging data. `inDISCO` creates transparency from black box data without compromising privacy which gives this approach a potential for application in the privacy-sensitive medical domain.

Acknowledgments

We want to acknowledge Khanh Nguyen for his ongoing work on adapting the `inDISCO` approach to a web-based DISCO application.

Competing interests

We have no conflicts of interest to declare.

[3] https://epfml.github.io/disco

Code availability

The code to reproduce our approach will soon be available on GitHub.

Authors contribution

Methodology and Investigation: K.N., S.P.K., A.D.; Validation: K.N.; Data curation: K.N.; Formal Analysis: K.N., A.D., S.P.K., M.A.H.; Conceptualization: M.A.H., S.P.K., K.N.; Visualization: K.N., M.A.H., A.D.; Discussion: all authors; Supervision: M.A.H., M.J.; Project Administration: M.A.H., M.J. Resources: M.J.; Writing (original draft): K.N.; Writing (final draft): K.N., A.D., M.A.H. All authors approved the final version of the manuscript for submission and agreed to be accountable for their contributions.

References

1. Piccialli, F., Di Somma, V., Giampaolo, F., Cuomo, S. & Fortino, G. A survey on deep learning in medicine: Why, how and when? *Information Fusion* **66,** 111–137 (2021).
2. Shen, D., Wu, G. & Suk, H.-I. Deep Learning in Medical Image Analysis. *Annu. Rev. Biomed. Eng.* **19,** 221–248 (2017).
3. Landi, I. *et al.* Deep representation learning of electronic health records to unlock patient stratification at scale. *npj Digital Medicine* **3** (2020).
4. Barnett, A. J. *et al.* Interpretable deep learning models for better clinician-AI communication in clinical mammography. *Proc. SPIE 12035, Medical Imaging 2022: Image Perception, Observer Performance, and Technology Assessment, 1203507* (2022).
5. McMahan, H. B., Moore, E., Ramage, D., Hampson, S. & y Arcas, B. A. Communication-Efficient Learning of Deep Networks from Decentralized Data. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)* **54** (2017).
6. Nguyen, D. C. *et al.* Federated Learning for Smart Healthcare: A Survey. *ACM Comput. Surv.* **55** (2022).
7. Rieke, N. *et al.* The future of digital health with federated learning. *npj Digital Medicine* **3** (2020).
8. Nazir, S. & Kaleem, M. Federated Learning for Medical Image Analysis with Deep Neural Networks. *Diagnostics* **13** (2023).
9. Islam, M., Reza, M. T., Kaosar, M. & Parvez, M. Z. Effectiveness of Federated Learning and CNN Ensemble Architectures for Identifying Brain Tumors Using MRI Images. *Neural Processing Letters* **55,** 3779–3809 (2023).
10. Ribeiro, M. T., Singh, S. & Guestrin, C. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 1135–1144 (2016).
11. Lundberg, S. & Lee, S.-I. A Unified Approach to Interpreting Model Predictions. *Proceedings of the 31st International Conference on Neural Information Processing Systems,* 4768–4777 (2017).
12. Selvaraju, R. R. *et al.* Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. *Journal of Computer Vision (IJCV)* (2019).
13. Simonyan, K., Vedaldi, A. & Zisserman, A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *Workshop at International Conference on Learning Representations* (2014).
14. Singh, A., Sengupta, S. & Lakshminarayanan, V. Explainable deep learning models in medical image analysis. *Journal of Imaging* **6** (2020).

15. Shrikumar, A., Greenside, P. & Kundaje, A. Learning Important Features Through Propagating Activation Differences. *PMLR* **70,** 3145–3153 (2017).

16. Chen, H., Lundberg, S. & Lee, S.-I. in *Explainable AI in Healthcare and Medicine: Building a Culture of Transparency and Accountability* 261–270 (Springer International Publishing, Cham, 2021).

17. Samek, W., Montavon, G., Vedaldi, A., Hansen, L. K. & Müller, K.-R. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning* (Springer, 2019).

18. Adebayo, J. *et al.* Sanity Checks for Saliency Maps. *Advances in Neural Information Processing Systems* **31** (eds Bengio, S. *et al.*) (2018).

19. Rudin, C. *et al.* Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys* **16,** 1–85 (2022).

20. Sun, S., Woerner, S., Maier, A., Koch, L. M. & Baumgartner, C. F. Inherently Interpretable Multi-Label Classification Using Class-Specific Counterfactuals. *Proceedings of Machine Learning Research* **73** (2023).

21. Chen, C. *et al.* This Looks like That: Deep Learning for Interpretable Image Recognition. *Proceedings of the 33rd International Conference on Neural Information Processing Systems,* 8930–8941 (2019).

22. Barnett, A. *et al.* A case-based interpretable deep learning model for classification of mass lesions in digital mammography. *Nat Mach Intell* **3,** 1067–1070 (2021).

23. Hase, P., Chen, C., Li, O. & Rudin, C. Interpretable Image Recognition with Hierarchical Prototypes. *AAAI Conference on Human Computation & Crowdsourcing* (2019).

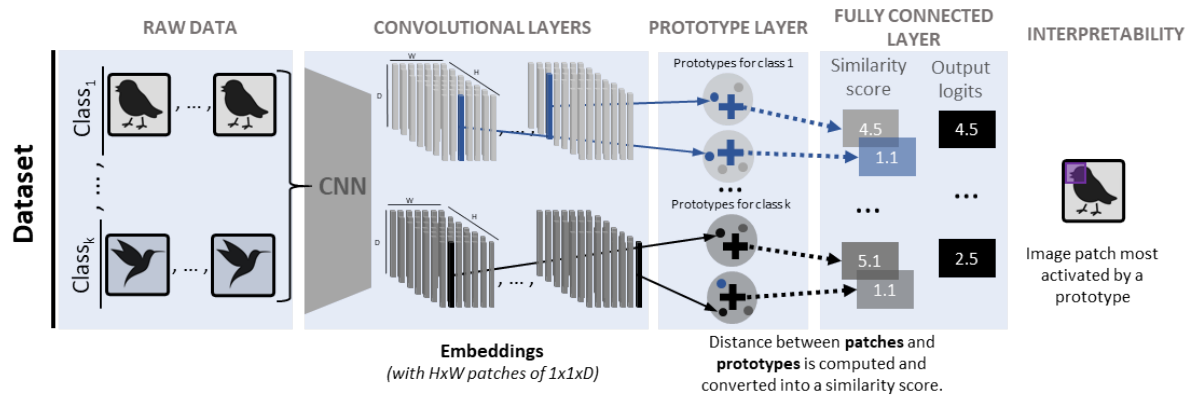
24. Irvin, J. *et al.* CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison. *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)* (2019).

25. Jiménez-Sánchez, A., Juodelye, D., Chamberlain, B. & Cheplygina, V. *Detecting Shortcuts in Medical Images - A Case Study in Chest X-rays.* Preprint at `https://arxiv.org/abs/2211.04279`. 2022.

26. Huang, G., Liu, Z., van der Maaten, L. & Weinberger, K. Q. Densely Connected Convolutional Networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2261–2269 (2017).

27. Deng, J. *et al. ImageNet: A Large-Scale Hierarchical Image Database* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2009), 248–255.

Supporting information



SI Fig. 1 `ProtoPNet` **architecture.** This is a centralized setting with no clients. `ProtoPNet` passes raw data through a CNN to create embeddings of size $[H \times W \times D]$ in the latent space, which can be seen as $H \times W$ image patches $[1 \times 1 \times D]$. These patches are clustered around the closest prototypes (+) which are being learned for each class in the prototype layer. The prototype

is a vector representing a class-characteristic feature in the latent space. Classification is based on a similarity score between the prototypes and the patches of an encoded image. In the final panel, we see that the patch most activated by a certain prototype can be visualized directly.

SI Local training description Given a set of training images $\mathbf{D}^n = \{(\mathbf{X}_i, y_i)\}_{i=1}^{l}$, where l is a number of images per client, each client aims to minimize the following objective:

$$\min_{\mathbf{P}^n, \mathbf{W}_c^n} \frac{1}{l} \sum_{i=1}^{l} \mathrm{CrsEnt}^n(h \circ g \circ f(\mathbf{X}_i), y_i) + \lambda_1 \mathrm{Clst}^n + \lambda_2 \mathrm{Sep}^n, \tag{5}$$

where f, g, and h denote the convolutional, prototype, and final fully connected layers, respectively. λ_1 and λ_2 are positive constants. CrsEnt is a cross-entropy loss that penalizes the misclassification, and the cluster and separation costs are defined as follows:

$$\mathrm{Clst}^n = \frac{1}{l} \sum_{i=1}^{l} \min_{j: \mathbf{p}_j^n \in \mathbf{P}_{y_i}^n} \min_{\mathbf{z}^n \in \mathrm{patches}(f(\mathbf{X}_i))} ||\mathbf{z}^n - \mathbf{p}_j^n||_2^2 \tag{6}$$

$$\mathrm{Sep}^n = -\frac{1}{l} \sum_{i=1}^{l} \min_{j: \mathbf{p}_j^n \notin \mathbf{P}_{y_i}^n} \min_{\mathbf{z}^n \in \mathrm{patches}(f(\mathbf{X}_i))} ||\mathbf{z}^n - \mathbf{p}_j^n||_2^2 \tag{7}$$

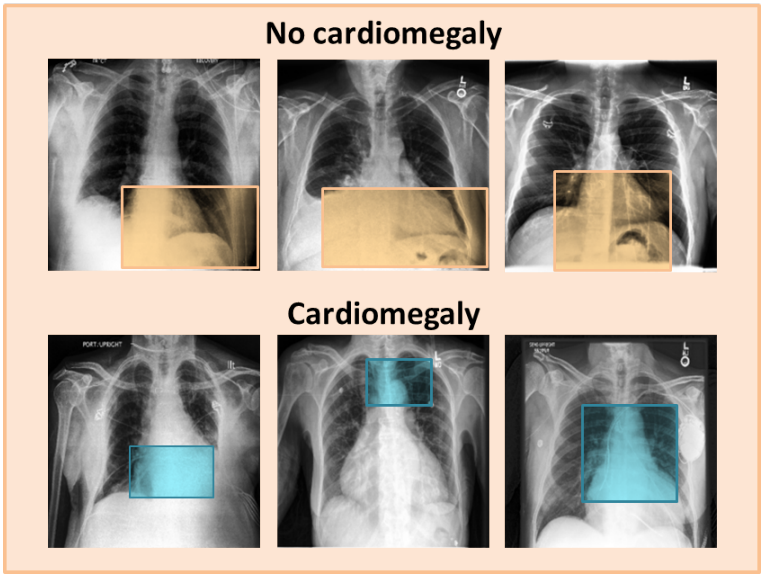
The minimization of the cluster cost (Clst) is needed to make each training image have a latent patch that is close to at least one prototype of the correct class. At the same time, every latent patch of a training image is separated from the prototypes of the incorrect classes through the minimization of the separation cost (Sep). More details about `ProtoPNet` can be found in [21] and in SI Fig. 1.

SI Table 1 Model performance in an unbiased setting. Classification sensitivity and specificity for **CM** (centralized model), **LM** (local model), **GM** (global model), and **PM** (personalized model) trained without data bias on CheXpert dataset for cardiomegaly and pleural effusion classes. The uncertainty is computed over three runs with different seeds and averaged over four datasets where applicable.

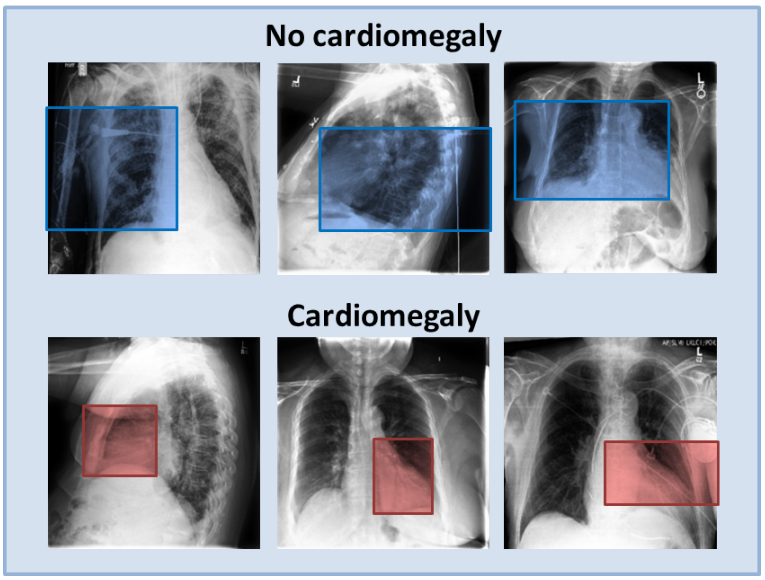
Model	**CM**	**LM**	**GM**	**PM**
Cardiomegaly classification				
Sensitivity, ±SD	0.66 ±0.04	0.66 ±0.04	0.68 ±0.08	0.60 ±0.08
Specificity, ±SD	0.83 ±0.02	0.77 ±0.02	0.80 ±0.06	0.67 ±0.05
Pleural effusion classification				
Sensitivity, ±SD	0.81 ±0.02	0.69 ±0.06	0.84 ±0.08	0.69 ±0.06
Specificity, ±SD	0.71 ±0.04	0.73 ±0.04	0.64 ±0.12	0.58 ±0.02

SI Table 2 Model performance in a biased setting. Classification sensitivity and specificity for LM^b, GM^b, and PM^b trained in an FL setting with one biased client on the CheXpert dataset for cardiomegaly and pleural effusion classes. 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. The uncertainty is computed over three runs with different seeds and averaged over four datasets where applicable.

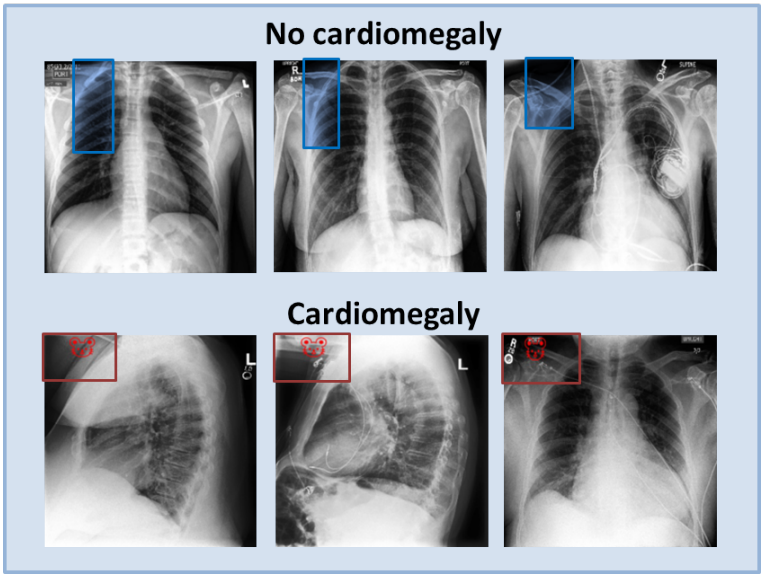
Model	LM^b		GM^b		PM^b	
Test set	Biased	Unbiased	Biased	Unbiased	Biased	Unbiased
Cardiomegaly classification						
Sensitivity, ±SD	1.0±0.0	0.0±0.0	0.46±0.28	0.34±0.30	0.80±0.20	0.0±0.0
Specificity, ±SD	1.0±0.0	1.0±0.0	0.77±0.22	0.77±0.22	1.0±0.0	1.0±0.0
Pleural effusion classification						
Sensitivity, ±SD	0.51±0.01	0.05 ±0.02	0.0±0.0	0.0±0.0	0.37±0.03	0.07 ±0.02
Specificity, ±SD	0.96±0.01	0.96 ±0.01	0.99±0.01	1.0±0.0	0.93±0.04	0.92 ±0.02



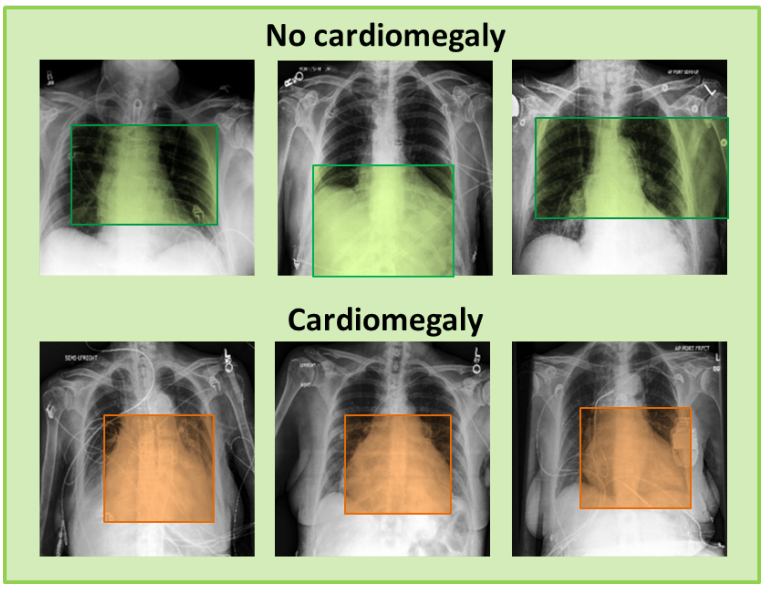
SI Fig. 2 Centralized prototypes. Examples of training images with bounding boxes indicating centralized prototypes learned on **unbiased** CheXpert data for *cardiomegaly* classification.



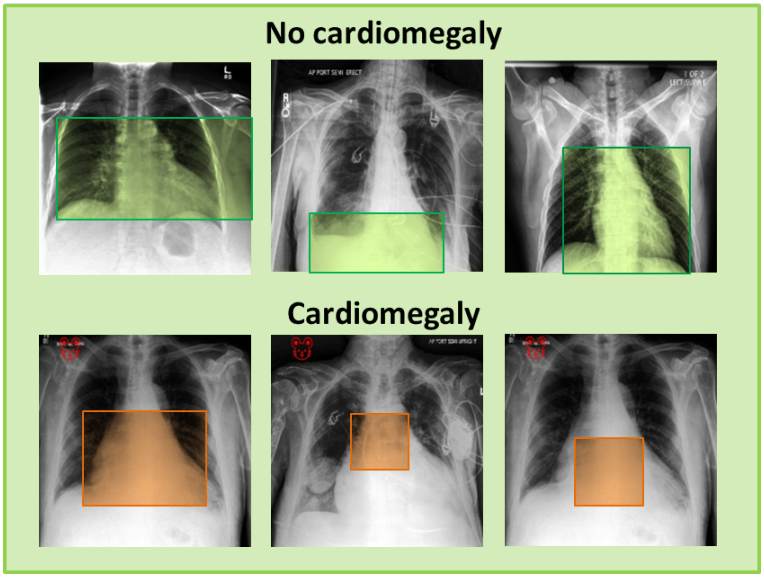
SI Fig. 3 Local unbiased prototypes. Examples of training images with bounding boxes indicating local prototypes learned on **unbiased** CheXpert data for *cardiomegaly* classification.



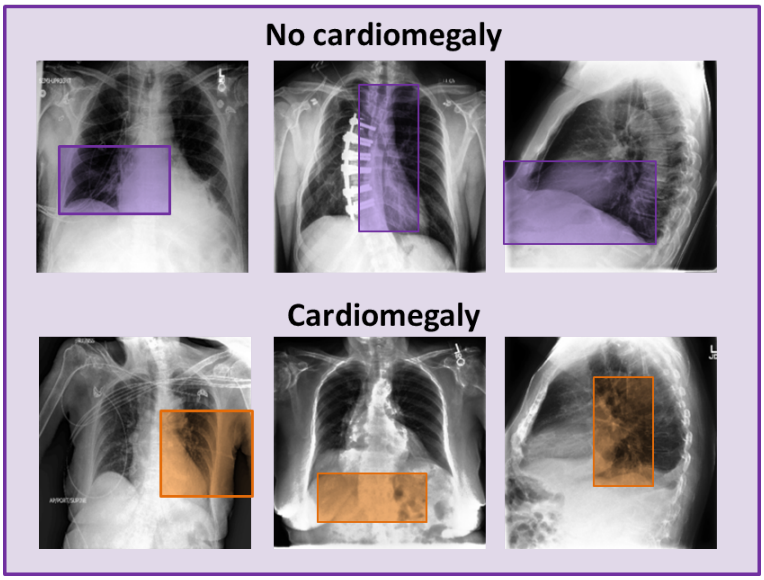
SI Fig. 4 Local biased prototypes. Examples of training images with bounding boxes indicating local prototypes learned on **biased** CheXpert data for *cardiomegaly* classification.



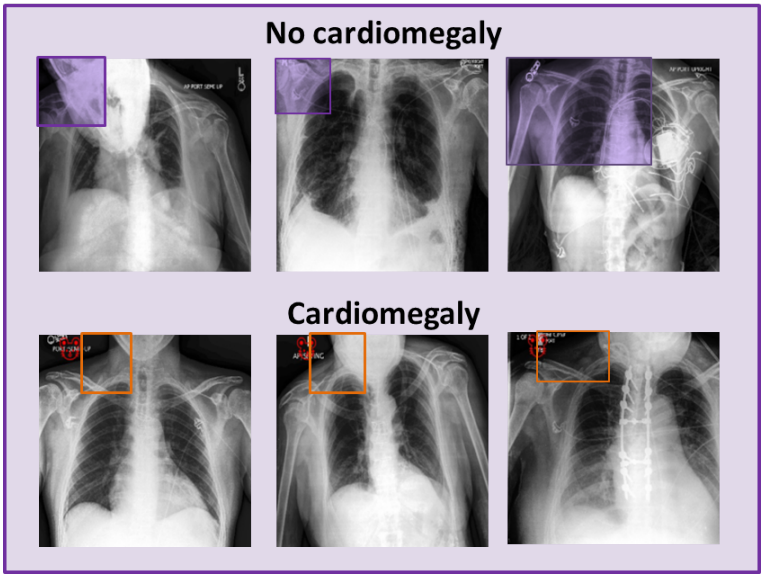
SI Fig. 5 Global unbiased prototypes. Examples of training images with bounding boxes indicating global prototypes learned on **unbiased** CheXpert data for *cardiomegaly* classification.



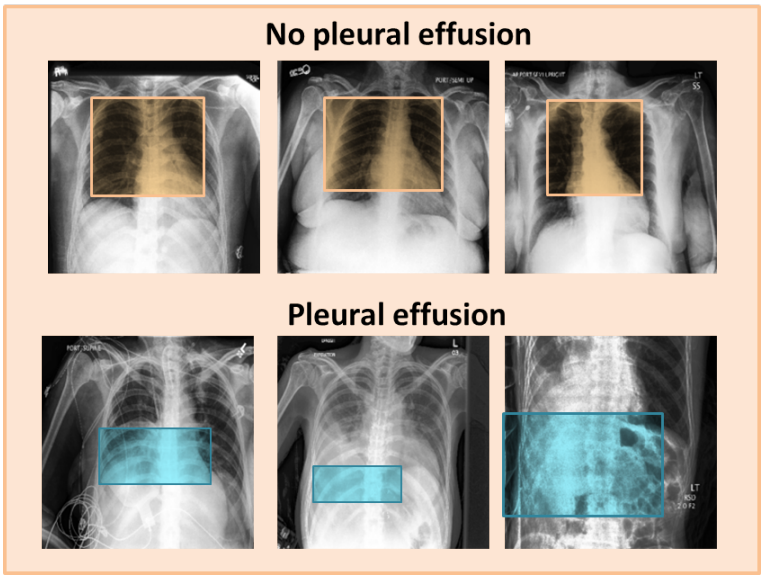
SI Fig. 6 Global biased prototypes. Examples of training images with bounding boxes indicating global prototypes learned on **biased** CheXpert data for *cardiomegaly* classification. The visualization is presented for the biased client.



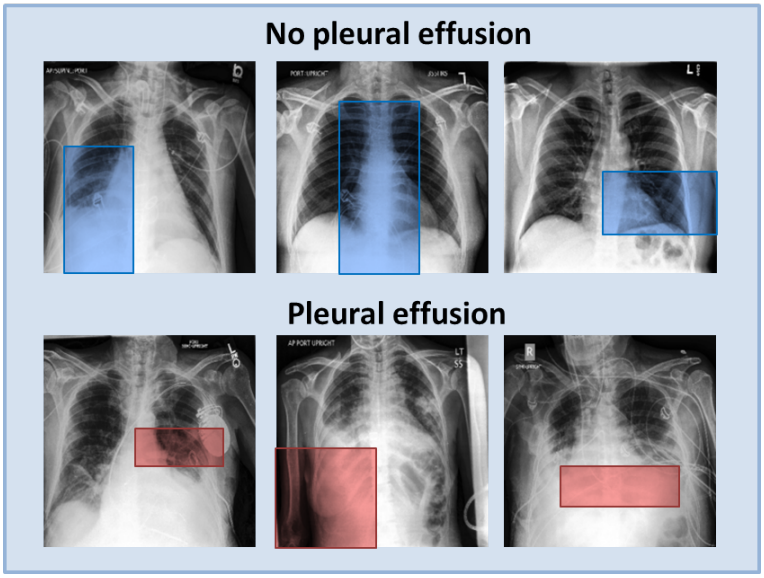
SI Fig. 7 Personalized unbiased prototypes. Examples of training images with bounding boxes indicating personalized prototypes learned on **unbiased** CheXpert data for *cardiomegaly* classification.



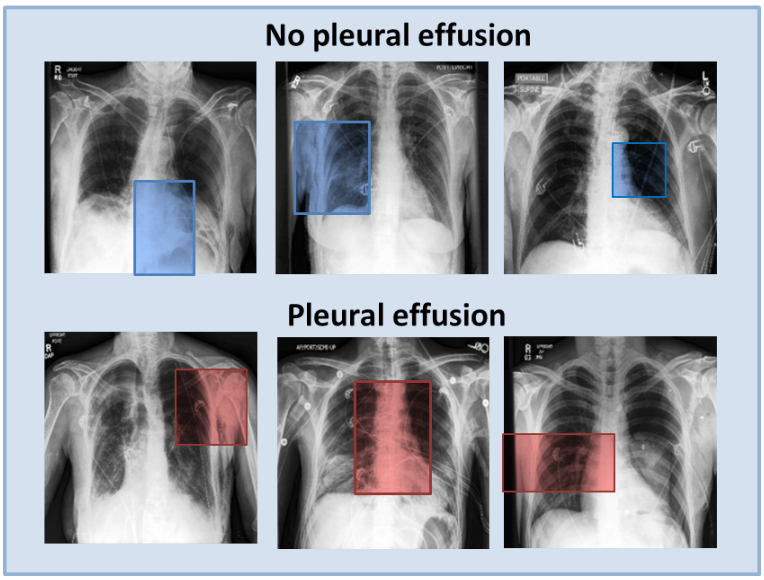
SI Fig. 8 Personalized biased prototypes. Examples of training images with bounding boxes indicating personalized prototypes learned on **biased** CheXpert data for *cardiomegaly* classification. The visualization is presented for the biased client.



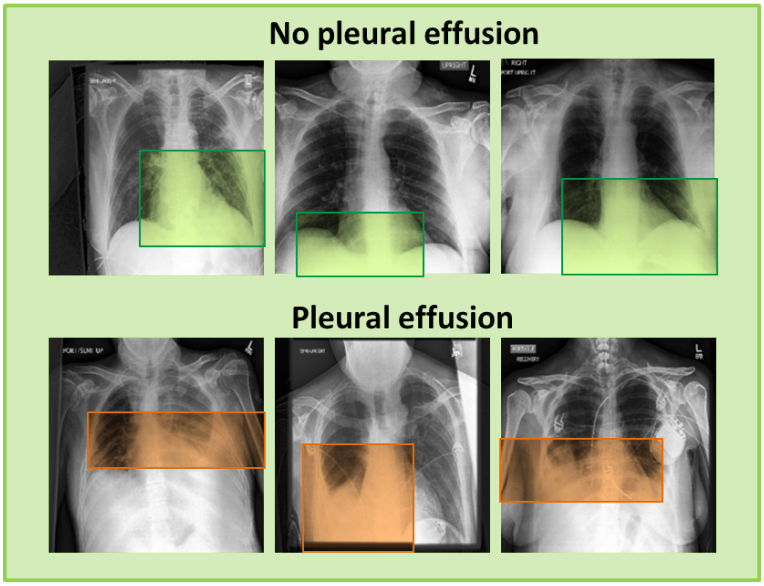
SI Fig. 9 Centralized prototypes. Examples of training images with bounding boxes indicating centralized prototypes learned on **unbiased** CheXpert data for *pleural effusion* classification.



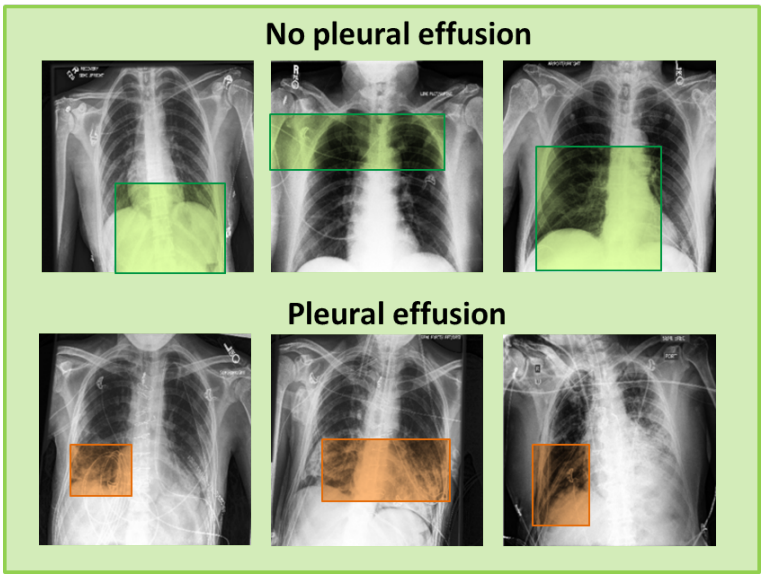
SI Fig. 10 Local unbiased prototypes. Examples of training images with bounding boxes indicating local prototypes learned on **unbiased** CheXpert data for *pleural effusion* classification.



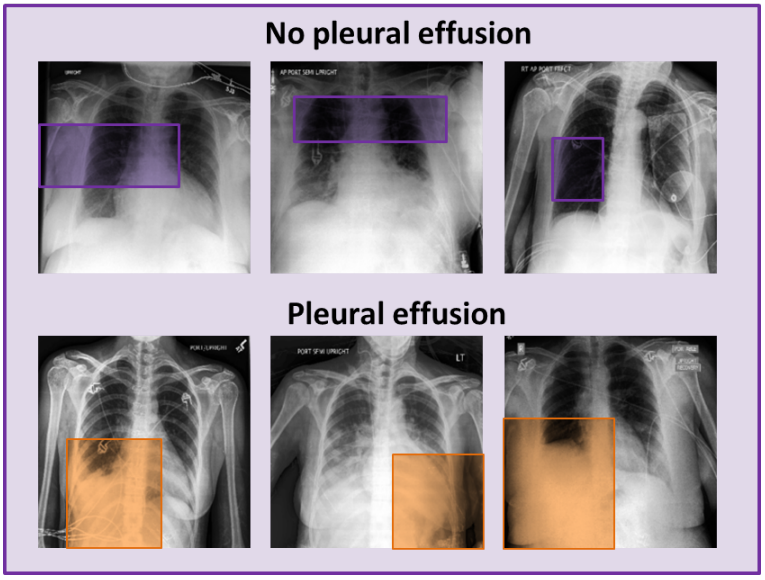
SI Fig. 11 Local biased prototypes. Examples of training images with bounding boxes indicating local prototypes learned on **biased** CheXpert data for *pleural effusion* classification.



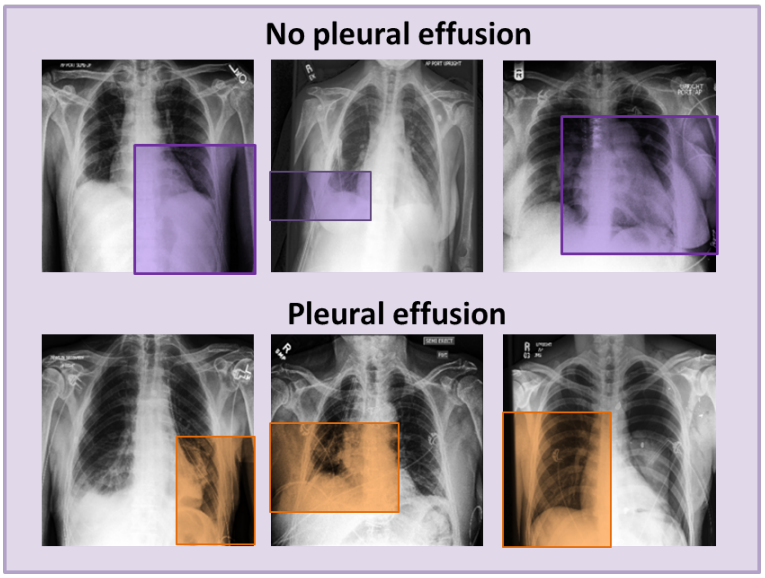
SI Fig. 12 Global unbiased prototypes. Examples of training images with bounding boxes indicating global prototypes learned on **unbiased** CheXpert data for *pleural effusion* classification.



SI Fig. 13 Global biased prototypes. Examples of training images with bounding boxes indicating global prototypes learned on **biased** CheXpert data for *pleural effusion* classification. The visualization is presented for the biased client.



SI Fig. 14 Personalized unbiased prototypes. Examples of training images with bounding boxes indicating personalized prototypes learned on **unbiased** CheXpert data for *pleural effusion* classification.



SI Fig. 15 Personalized biased prototypes. Examples of training images with bounding boxes indicating personalized prototypes learned on **biased** CheXpert data for *pleural effusion* classification. The visualization is presented for the biased client.