A Privacy Framework for Hierarchical Federated Learning

Debmalya Biswas
Darwin Edge
Lausanne, Switzerland
debmalya.biswas@darwinedge.com

Krishnamurthy Vidyasankar
Memorial University of Newfoundland
St. John's, NL, Canada
vidya@mun.ca

Abstract

Federated Learning (FL) enables heterogeneous entities to collaboratively develop an optimized (global) model by sharing data and models in a privacy preserving fashion. We consider a Hierarchical Federated Learning (HFL) environment with data ownership split among the entities representing the edge nodes. Each node can train models on the data they own, as well as request access to data and model(s) owned by their descendant nodes - to optimize their models, perform transfer learning on new data, and develop an ensemble model. Unfortunately, a practical realization of HFL is challenging today due to issues with data/model lineage tracking and providing subsequent privacy guarantees.

In this paper, we propose a conceptual framework for HFL by capturing the data/model attributes at each node, including their privacy exposure. The framework enables scenarios where a node output may expose certain attributes of its underlying data, as well as identifying models in the hierarchy that need to be updated once a user whose data was used in their training has opted-out. By designing the computations appropriately and limiting the exposure by the nodes, we show that different levels of privacy can be guaranteed.

In: A. Editor, B. Coeditor (eds.): Proceedings of the XYZ Workshop, Location, Country, DD-MMM-YYYY, published at http://ceur-ws.org

1 Introduction

Federated Learning [MMR+17], also known as Collaborative Learning, or Privacy Preserving Machine Learning, enables multiple entities who do not trust each other (fully), to collaborate in training a Machine Learning (ML) model on their combined dataset; without actually sharing data — addressing critical issues, such as, privacy, access rights and access to heterogeneous confidential data. This is in contrast to traditional (centralized) ML techniques where local datasets (belonging to different entities) need to be first brought to a common location before model training. Its applications are spread over a number of industries including healthcare, defense, telecommunications, and advertising.

Let us now focus on ML related privacy risks [RG20, BFA20]. Fig. 1 illustrates the attack scenarios in a ML context. In this setting, there are mainly two broad categories of inference attacks: membership inference and property inference attacks. A membership inference attack refers to a basic privacy violation, where the attacker's objective is to determine if a specific user's data item was present in the training dataset. In property inference attacks, the attacker's objective is to reconstruct properties of a participant's dataset. We focus on property inference attacks in this work.

It has been shown that the model parameters may reveal information about the underlying dataset(s) and/or their properties [NSH19]. When the attacker does not have access to the model training parameters, it is only able to run the models (via an API) to get a prediction/classification. Black box attacks [IEAL18] are still possible in this case where the attacker has the ability to invoke (query) the model, and observe the relationships between inputs and outputs.

Given the above attack scenarios, let us now focus on the applicable privacy preserving approaches. Secure Multiparty Computation (SMC) [Gol09, KBdH09] primitives, e.g., Secret Sharing, Homomor-

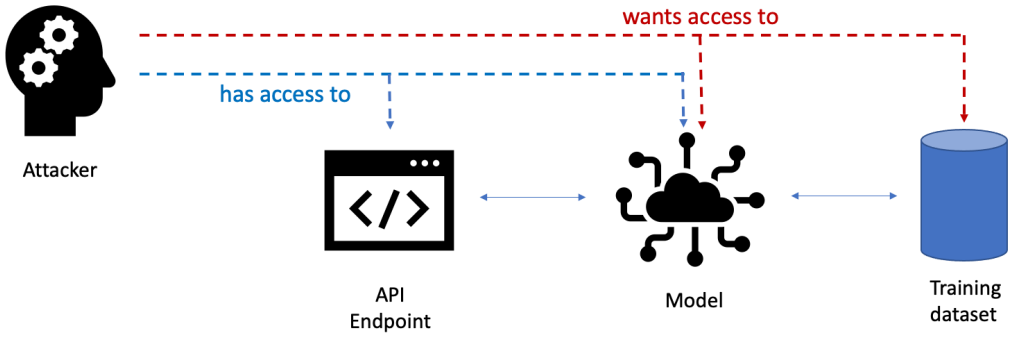

Figure 1: Machine Learning (ML) attack scenarios

phic Encryption, have been effectively applied to protect the privacy of model parameters in a federated learning setting. [BIK+17] proposes a 'Secure Aggregation' protocol, where the aggregating server only learns about model updates in aggregate. The αMDL protocol proposed in [ZJWW17] further encrypts the gradients using homomorphic encryption. The summation of the encrypted gradients gives an encrypted global gradient, which can only be decrypted once a threshold number of participants have shared their gradients. [SPTP+20] proposes POSEIDON: a Multiparty Homomorphic Encryption based Neural Network (NN) training protocol, which (relies on mini-batch gradient descent, and hence) protects the intermediate NN models by maintaining the weights and gradients encrypted throughout the training phase. The protocol can be applied to build different types of NN layers, such as fully connected, convolution, and pooling. In terms of model accuracy, the authors show that their model performance is comparable to a centrally trained model.

While SMC protocols are effective, they suffer from increased communication and computational complexity, both for the participant nodes and the aggregating server. Differential privacy provides a good trade-off here balancing utility and privacy guarantees. [SS15] experiment with adding noise to the model updates to satisfy differential privacy and protect the contributions of participants to the global model. [PAE+17] show how multiple (teacher) nodes trained on sensitive data, can further train a (student) model based on differentially private aggregated outputs of the teacher nodes - such that the student node is able to make similar predictions without leaking any sensitive data.

In this work, we do not focus on specific attack scenarios or privacy preserving methods. Rather, our objective is to capture and quantify the privacy exposure arising as a result of the diverse computations and privacy preserving actions that can be applied by the different participant nodes in a Federated Learning setting. The privacy exposure issue becomes even more prevalent in a hierarchical setting, where ancestor nodes may not have direct visibility over the data/computations of the descendant nodes. However, the data, attributes, or computation results, of the descendent nodes may still be visible to the ancestor nodes, influencing their computations. This leads to primarily two challenges:

- Data Lineage: Keeping track of the data, attributes, computation results accessible to the different participant nodes in HFL.
- Privacy policies: Enforcing local and global privacy policies in HFL, such that all participant nodes are using the data in a manner consistent with the user consent obtained during data acquisition.

For example, a recent Federal Trade Commission (FTC) ruling [FTC21] states that it is no longer sufficient to just delete data when a user opts-out; the organizations will need to delete models and algorithms trained on that data as well. Enforcing this in a HFL setting requires capturing the ancestor nodes, whose computations have directly or indirectly been influenced by user data belonging to descendant nodes, and vice versa. Privacy policies, e.g. FTC Fair Information Practice Principles (FIIPs) [Gel21] also recommend that data is only used for specific purposes (for which the user has provided explicit opt-in), and not combined with other datasets to reveal additional insights that can be used to profile the user. Such data aggregations can be very difficult to detect in a hierarchical setting, as ancestor nodes can indirectly access (via intermediate nodes) and aggregate data belonging to different descendant nodes, without their explicit approval.

To overcome the above challenges in a hierarchical setting, we propose a conceptual framework to capture and quantify the privacy exposure of the data/model attributes of each node. The framework is outlined in Section 2 and takes into account different computations, including explicit privacy preserving computations, performed by the nodes. We show the practicality of the model in Section 3 via two concrete application scenarios: (i) Hierarchical federated training of a Deep Neural Network - Section 3.1, and (ii) Hierarchical (ensemble) composition of pre-trained models - Section 3.2. Section 4 concludes the paper and provides some directions for future work.

2 Framework

We have a hierarchy (rooted tree) $\mathcal{H}$ of processing nodes q. The hierarchy consists of n levels. We will associate a persistent data store S_q at each node q and denote the data in the store as DS_q.

Processing starts at the leaf level. Each leaf node p is capable of performing arbitrary computations on its locally stored data in DS_p. The newly computed values are sent to the parent node. The parent performs further computation on the received values and outputs new values to its parent. This process continues and it constitutes the forward flow. We also consider reverse flow, where any intermediate node (depending on the computation) may send newly computed values to its children nodes. The computation done at a node q in the forward flow is denoted c_q, and one in the reverse flow is denoted c'_q.

In the forward flow, the output data from each (leaf or intermediate) node q will have values for several attributes. Some of these values may reveal (a part of) DS_q. It may also reveal DS_p of some descendants p of q. Similarly, the outputs may reveal DS of some nodes in the reverse flow also. We address these privacy issues relating to the nodes, namely, the visibility or exposure of the characteristics of the nodes, that is, of (i) their attributes, (ii) the data they process and (iii) the results of the processing, from their outputs. We refer to this simply as exposure of the data storage DS of the node, simplified further as exposure of the node.

We introduce the notations and concepts of our framework with a hierarchy depicted in Fig. 2. In the following, the nodes u, v, w, x, y and z refer to those in the figure, and p, q and r refer to generic nodes. We first consider the following path, from leaf node x to u, then to v and then to w:

$x \rightarrow u \rightarrow v \rightarrow w$.

The output sent from node x to node u in the forward flow is denoted $Out_{(x,u)}$. It will contain some attribute values in DS_x, denoted $(x,u)^F$, or simply as x^F when there is no ambiguity with respect to the destination node u. In addition to x^F, the output will contain some other information. The input $In_{(x,u)}$ to u from x is the same as $Out_{(x,u)}$. When the inputs are received from several children of u, the union of those inputs is denoted as In_u. The output in the reverse flow from u to x will be denoted $Out'_{(u,x)}$ and the input as $In'_{(u,x)}$. The attribute values sent in the reverse flow are denoted $(u,x)^R$, or simply as u^R. In these notations, we pretend that the edges of the hierarchy are directed from the leaves to the root in the forward flow, and from the root to the leaves in the reverse flow.

Consider $(u,v)^F$ sent in the output $Out_{(u,v)}$ from u to v. There may be a leakage of DS_u in that output. In addition, there may be a leakage of DS_x also in that output. That is, the output may leak some characteristics of the descendants also. We identify all nodes that are leaked (exposed) in the output as a subgraph consisting of those nodes as follows. Here, $\mathcal{H}_q$ is the sub-hierarchy of $\mathcal{H}$ rooted at q.

Definition 1 *For node q and r, $exp(Out_{(q,r)})$ is a subgraph $\mathcal{G}$ of $\mathcal{H}_q$.*

$Out_{(q,r)}$ exposes all the nodes in the graph $\mathcal{G}$. Referring to the example above, we do not consider the possibility that $Out_{(u,v)}$ exposes x but not u. We will assume that if x is exposed to v then u is also exposed to v. That is, we do not allow a child to be exposed to an ancestor if the child's parent is not. This justifies denoting the exposure by a tree which is a connected graph. For example, with reference to the hierarchy outlined in Fig. 2, $exp(Out_{(u,v)})$ could be the connected subgraph consisting of x, u and v, denoted $[x \rightarrow u \rightarrow v]$.

These outputs are of the computations performed at the nodes on an input. Now, in our example hierarchy, c_u may be such that $Out_{(u,v)}$ does not expose x. For example, an aggregate of the input values may not expose the source of an individual value. Furthermore, additional *privacy-preserving* computations such as, anonymization, differential privacy [PAE+16], etc. may be performed to control the exposure. We distinguish the privacy-preserving component of the computation used for the output from q to r as $c_{(q,r)}$. We express the exposure resulting from such computations also as a graph as follows.

Definition 2 *For a node q, $exp(c_{(q,r)})$ with respect to output $Out_{(q,r)}$ is a subgraph $\mathcal{G}$ of $\mathcal{H}_q$.*

In our example hierarchy, some possible values for the subgraphs in $exp(c_{(v,w)})$ are: null, $[v]$, $[u \rightarrow v]$, $[x \rightarrow u \rightarrow v]$ and $[x \rightarrow u \rightarrow v; z \rightarrow v]$. In the last

case, the subgraph consisting of the nodes x, u, v and z is exposed. Now, suppose $Out_{(u,v)}$ does not expose x. Then, even if $exp(c_{(v,w)})$ has $[x \rightarrow u \rightarrow v]$, $Out_{(v,w)}$ cannot expose x. That is, for the node v, the exposure in $Out_{(v,w)}$ depends on the exposure in $In_{(u,v)}$ and the exposure property of $c_{(v,w)}$. We express this as follows:

$exp(Out_{(v,w)})$ is $exp(In_{(u,v)}) \cap exp(c_{(v,w)})$.

We note that x will not be exposed to w as long as there is a (privacy preserving) computation at a node in the path from x to w that does not expose x. This non-exposure may be dictated by x to u, as part of its privacy policies or it can be a result of u's underlying computation c_u, such that $exp(Out_{(u,v)})$ does not contain x. If the non-exposure is mandated by privacy policy and is not provided by c_u, u may perform an explicit privacy preserving computation $c_{(u,v)}$, such that $exp(c_{(u,v)})$ does not contain x.

We specify the nodes to which a node (that is, node's attribute) can be exposed to during forward flow as Domain of Forward Exposure, DoF:

Definition 3 *Domain of Forward Exposure of a node q, $DoF(q^F)$, is the subtree (path) connecting q to an ancestor r of q.*

Here, we include q also as an ancestor of q. For instance, $DoF(x^F)$ in our example could be $[x \rightarrow u \rightarrow v]$. $DoF(q^F)$ will help in designing privacy preserving computations of the ancestors of q.

In a general hierarchy, the privacy requirements could span ancestors as well as other nodes. For example, x may allow exposure to (i) u and its children, namely, the siblings of x, and (ii) u, v and the siblings of u, that is, the children of v. This is especially true with reverse flow, where output values from a parent u to its children, based on a computation on values received from some of its descendants, might reveal x's attributes to its siblings, and their respective descendants.

We capture this exposure of a node as a result of both forward and reverse flows as Domain of Exposure, DoE:

Definition 4 *Domain of Exposure of a node q, $DoE(q)$, is a subtree of the tree rooted at an ancestor r of q, $\mathcal{H}_r$.*

We illustrate DoF and DoE with the help of the hierarchy outlined in Fig. 2, and also outline the steps to construct/maintain DoEs in an incremental fashion.

We consider the following scenario in the forward flow.

(i) Starting with $DoF(x^F) = [x \rightarrow u]$, when u forwards its computation output $Out_{(u,v)}$ to its parent v, such that $exp(Out_{(u,v)})$ contains u, $DoF(u^F)$ is updated to include v, that is, $DoF(u^F) = [u \rightarrow v]$. Suppose that $exp(Out_{(u,v)})$ contains x. Then $DoF(x^F)$ is also updated, to $[x \rightarrow u \rightarrow v]$. That is, the DoF's of the descendants of u may also be updated. This is formalized in Algorithm 1: $Update\_DoF(v, Out_{(v,w)})$.

(ii) Node y has forwarded y^F to z and so $DoF(y^F) = [y \rightarrow z]$; $Out_{(z,v)}$ does not expose y and so $DoF(y^F)$ is not updated further.

(iii) Going forward, when v forwards its computation output $Out_{(v,w)}$ to its parent w, such that $exp(Out_{(v,w)})$ contains v, $DoF(v^F)$ is updated to include w, that is, $DoF(v^F) = [v \rightarrow w]$. Now, suppose, x is not exposed to w. Then there is no need to update $DoF(x^F)$.

Fig. 2(a) illustrates this forward flow, with $DoF(x^F) = [x \rightarrow u \rightarrow v]$, $DoF(y^F) = [y \rightarrow z]$ and $DoF(v^F) = [v \rightarrow w]$. The label (bold) X besides node v indicates that forwarding of node x stops at v. Similarly, the (bold) X besides z indicates that the forwarding of y stops at z.

Algorithm 1: $Update\_DoF(p, Out_{(v,w)})$: Forward flow update of DoFs triggered by node v sending output $Out_{(v,w)}$ to parent w

```
for all child nodes q of p do
    if q ∈ exp(Out_(v,w)) then
        Add v → w to DoF(q^F);
        Update_DoF(q, Out_(v,w));
    end
end
```

Consequently, let us consider the reverse flow illustrated in Fig. 2(b), when v returns values to its children u and z, $Out'_{(v,u)} = Out'_{(v,z)}$. Given that c'_v is computed not only on values received (in reverse flow) from its parent w: $Out'_{(w,v)}$, but also values previously provided by u and z in the forward flow; it is feasible that $Out'_{(v,u)}$ might expose to u its sibling z's attributes. In addition, it might also expose attributes of some descendants p of z, to u. Given this, an addition of u to $DoE(z)$, could also trigger an update of the $DoE(p)$ of z's descendants p. On the other hand, since y's attributes were never exposed to v in the forward flow, that is, $DoF(y^F)$ does not contain v; y will never get exposed to u, even during reverse flow. The reverse flow update of DoEs is formalized below in Algorithm 2.

Algorithms 1 and 2 move the hierarchy $\mathcal{H}$ from one 'consistent' state to another, with respect to the definition of DoF/DoEs. In practice, it is possible that in reverse flow, a parent node sends the same value(s) to all children (Section 3), as well as the scenario where the parent sends different values to different (groups

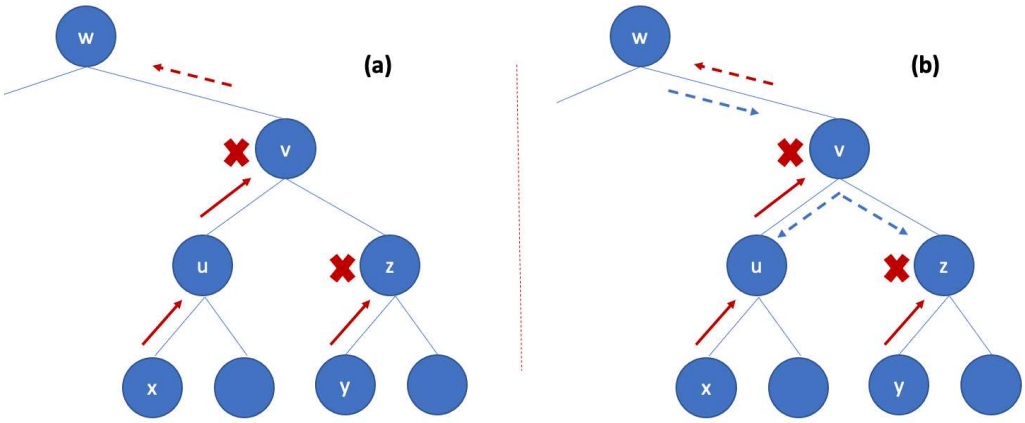

Figure 2: Sample hierarchy illustrating DoF/DoE computation

Algorithm 2: *Update_DoE*$(Out'_{(v,u)})$: Reverse flow update of DoEs triggered by node v sending output to its child node u

for *all ancestor nodes* r *of* u **do**
  if $r \in exp(Out'_{(v,u)})$ **then**
    Add $v \rightarrow u$ to $DoE(r)$;
  end
end
for *all descendant nodes* p *of* v **do**
  if $v \in DoF(p^F)$ *AND* $p \in exp(Out'_{(v,u)})$ **then**
    Add $v \rightarrow u$ to $DoE(p)$;
  end
end

of) children - leading to sibling nodes receiving different values from their parent (Section 3.2).

3 Hierarchical Federated Learning

In this section, we show how two hierarchical scenarios can be enabled by the privacy framework outlined in the previous section.

3.1 Federated Training of Neural Networks

Training a Deep Neural Network (DNN) occurs over multiple iterations (epochs). Each forward run is coupled with a feedback loop, where the errors identified at the end of a run with respect to the difference between the network output and true value of its objective function is fed back to the previous (hidden) layers to adapt their parameter weights - 'backpropagation'. The commonly used algorithms to solve this optimization problem are variants of gradient descent.

A Federated Learning (FL) extension of the above DNN training proceeds as follows: In a 2-level FL setting, this corresponds to the leaf nodes holding non-overlapping datasets with the same features. (In FL terminology, we consider a horizontal FL environment where the feature space is same, but the training data is split among the participating entities.) For any two leaf nodes (p_1, p_2), $DS_{p_1} \cap DS_{p_1} = \emptyset$.

Further, all leaf nodes p agree upon the same neural network architecture and task to train a global model, that is, they perform the same computation c_p locally. The root node acts as a parameter server, maintaining the latest version of the parameter values for the global model. During each epoch, the leaf nodes download the global model parameters from their parent node, and update them locally using some variant of gradient descent on their local datasets DS_p, sharing the updated values back with the parent node. The parent (in this case, root) node averages the gathered parameter values from all child nodes. This federated training continues until the global model converges.

A hierarchal extension of the above FL occurs when the root node of the previous 2-level FL becomes a child of a 3rd level node (illustrated in Fig. 3). As long as the information flow is restricted to parent-child nodes only, the HFL architecture leads to a synchronous bottom-up training of the neural network, with the root node holding the optimal global model.

3.1.1 Privacy Analysis

In the hierarchical training of a DNN outlined above, a child node p only shares parameter/gradient updates, and never shares its sensitive training dataset DS_p. However, it has been shown that even the parameter/gradient updates may reveal information about the underlying dataset(s) DS_p and/or their (properties) attributes [NSH19]. This is because (during back-

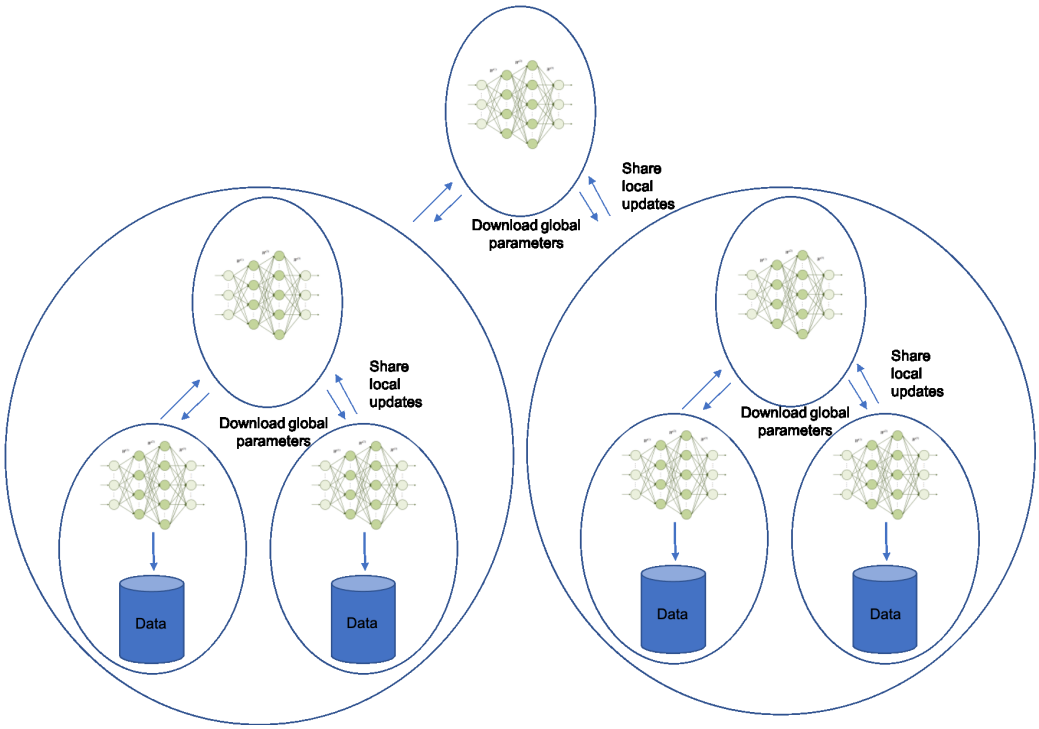

Figure 3: Hierarchical federated training of a Deep Neural Network

propagation) gradients of a given layer l of a neural network are computed using the layer's feature values and the error from the next layer. For example, in the case of sequential fully connected layers, h_l, h_{l+1} ($h_{l+1} = W_l \cdot h_l$, where W_l is the weight matrix), the gradient of error E with respect to W_l is defined as: $\frac{\partial E}{\partial W_l} = \frac{\partial E}{\partial h_{l+1}} \cdot h_l$. That is, the gradients of W_l are inner products of the error from the next layer and the features h_l; and hence the correlation between the gradients and features. This is especially true if certain weights in the weight matrix are sensitive to specific features or values in the participants' datasets (for example, specific words in a language prediction model [MRTZ18]).

Recall that given two nodes p_1, p_2 and their parent q, q's computation c_q corresponds to an averaging of outputs $Out_{(p_1,q)}$ and $Out_{(p_2,q)}$ received from p_1 and p_2, respectively. The 'averaging' computation part can be considered as an implicit privacy preserving computation in this case, which has the same effect as adding noise, e.g., using synthetically generated data [BDR19]. This does not mean that q's output $Out_{(q,r)}$ to its parent r will never expose any of p_1, p_2's attributes to r. Rather, it implies that

$$exp(Out_{(q,r)}) \subset (exp(Out_{(p_1,q)}) \cup exp(Out_{(p_2,q)})).$$

Due to the iterative nature of the training, it is possible that node attributes exposed in an iteration are no longer exposed in a later iteration. Here, we assume that the nodes have capability to store historical values and the exposure at iteration n is an aggregation of the exposures resulting from previous iterations. Let $Out_{i(p,q)}$ denote p's output to q at iteration i

Definition 5 *For nodes p and q, the cumulative exposure until iteration n is defined as*
$exp_{Cn(p,q)} = \bigcup_{i=1}^{n} exp(Out_{i(p,q)})$
which is a subtree of $\mathcal{H}_p$.

The *DoE* definition follows analogously from Definitions 3 and 4 respectively.

Definition 6 *Domain of Exposure of a node p, at iteration n, $DoE_n(p)$, is a subtree $\mathcal{H}_q$ of the tree $\mathcal{H}$ rooted at the highest (ancestor) node $q \in \bigcup_{i=1}^{n} DoE_i(p)$.*

In this setting, we consider the privacy problem of maintaining (global) model compliance when a user opts-out. This implies that not only can the opted-out user's data be no longer used, but also that all models exposed to that data during training must be deleted as well. Without exposure (lineage) tracking, it would be very difficult to assess the impact of such an opt-out on the global model, leading to re-training of the

whole hierarchy in the worst case. The below property shows how our model helps in containing the impact of a user opt-out - limiting the amount of re-training needed, while maintaining privacy compliance.

Property. For any opt-out of user U, where U's data belongs to DS_p of a leaf node p of hierarchy $\mathcal{H}$, it is sufficient to re-train models up to the ancestor q of p, such that $DoE(p)$ is a subgraph of $\mathcal{H}_q$.

3.2 Ensemble Learning/Models Marketplace

In this section, we consider a more generic hierarchical composition of services [Bis21], where the underlying data, (trained) models, APIs of different (existing) services can be orchestrated to form a new service. This is enabled by pre-trained models available on model marketplace platforms, e.g., Amazon Sagemaker Models, Algorithmia, Bonseyes, Papers with Code, etc.

Scenario. Let us consider the online Repair Service of a luxury goods vendor. The service consists of a Computer Vision (CV) model enabled Product Repair Assessment Service that is able to assess the repairs needed given a picture of the product uploaded by the user. If the user is satisfied with the quote, the assessment is followed by an Ordering Chatbot conversation that captures additional details required to process the user's repair request, e.g., damage details, user name, contact details, etc.

In future, when the enterprise is looking for models to develop a Product Recommendation Service, the Repair Service is considered. As evident, the data gathered by the Repair Service: state of products owned by the users (gathered by CV assessment model) together with their demographics (gathered by the Ordering chatbot) - provides additional training data for the Recommender Service. In this case, however, the privacy policies of the CV assessment app, Ordering Chatbot [Bis20], or any global privacy policies governing the hierarchical ecosystem (e.g. FTC FIPs [Gel21], which require that user data is only used for specific purposes, for which the user has provided explicit opt-in) may prevent their data from being combined, such that, they cannot be used to profile customers.

Let us now consider another hierarchical composition scenario, where the enterprise further wants to develop a CV enabled Manufacturing Defect Detection Service. The Repair Service can help here as it has labeled images of damaged products (with the product damage descriptions provided to the chatbot acting as 'labels'). The labeled images can also be provided as a feedback loop to the Product Repair Assessment Service - CV model, to improve its underlying model.

3.2.1 Privacy Analysis

Consider the hierarchy in Fig. 4, as a generic representation of the hierarchical scenario discussed above. In this context, we show how the different privacy concerns can be addressed by our privacy model (Section 2).

- We consider the Repair Service as a parent node u orchestrating pre-trained Computer Vision and NLP models - nodes x and y, respectively.
- In case of the Product Recommendation Service - node v, depending on the applicable privacy policy, the Repair Service node u might need to apply an explicit privacy preserving computation $c_{(u,v)}$ on the 'demographics' attribute y^F such that it is only shared in aggregate form with v (e.g., number of users having product type P with damage Q, without specific details of the users possessing those products).
- Both the Product Manufacturing Defect Detection App - node w and the Product Repair Assessment Service - node x are CV models that would benefit from the product damage descriptions gathered by the Ordering Chatbot - node y. The descriptions basically act as "ground truths" that can be used to further improve the underlying CV models. The Repair Service needs to perform explicit privacy preserving computations $c_{(u,w)}$ and $c'_{(u,x)}$ (e.g., filtering), during forward and reverse flows, such that the uploaded images with their damage descriptions can be shared with w and x respectively, without the corresponding user demographic details.
- Further, it also shows that the parent node u can send different values to different children nodes x and y, such that $Out'_{(u,x)} \neq Out'_{(u,y)}$ - as dictated by their computational needs and applicable privacy policies.

The core privacy issue here is to ensure that at each step of the hierarchical composition, user data is only used, and combined with other data, for purposes for which the user has provided explicit opt-in. This can become difficult to track in a hierarchy, with ancestor nodes not having clarity over whether opt-ins have been acquired by descendant nodes for their use-cases. In our model, the above privacy compliance requirement can be enforced as follows:

Property. The inclusion of an ancestor node r in the $DoE(p)$ of a descendent node p, implies that p has obtained the necessary opt-in for an attribute p^F to be used:

- by r, as part of computation c_r.

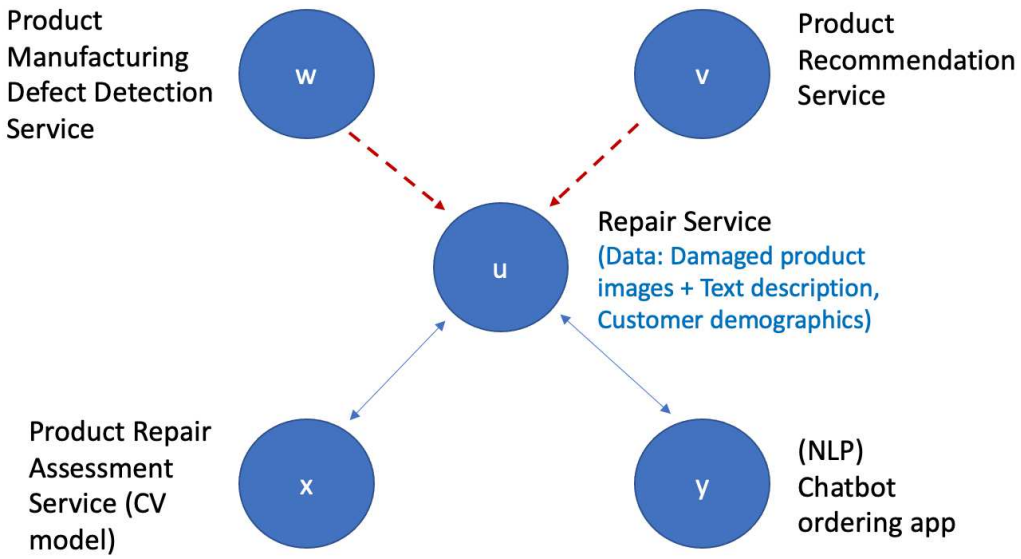

Figure 4: Hierarchical composition scenario

- in an aggregated fashion with other attributes q^F, where q, $q \neq p$, is also a descendant of r, such that r is in $DoE(q)$.

From a user point of view, this enables continuous tracking of the nodes having visibility over their data, that is, the purpose(s) for which their data is used, as well as the different types of data with which they are combined.

4 Conclusion

In this paper, we presented a framework to quantify the privacy exposure at different nodes in a hierarchical setting. It provides the necessary constructs to track lineage of data/models in a hierarchal composition, ensuring that appropriate privacy policies can be applied to meet compliance requirements. We showed the practical utility of the proposed framework on two hierarchical scenarios: (i) federated training of a DNN and (ii) ensemble composition of AI/ML Services.

Our work is seminal in both raising awareness and providing a framework to reason about privacy in a hierarchical setting. The presented framework is conceptual, and any implementation will be entirely application-specific. Future work will involve (for specific applications) precise specifications of privacy requirements of the datasets of the different nodes, and the feasibility and design of privacy-preserving computations placed at different levels of the hierarchy to satisfy the privacy requirements. Several choices for the latter can be experimented and an insight into better choices can be found.

References

[BDR19] S. M. Bellovin, P. K. Dutta, and N. Reitinger. Privacy and Synthetic Datasets. *Stan. Tech. L. Rev.*, 22(1), 2019.

[BFA20] C. Briggs, Z. Fan, and P. Andras. A Review of Privacy-preserving Federated Learning for the Internet-of-Things. *arXiv*, abs/2004.11794, 2020.

[BIK+17] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1175–1191. ACM, 2017.

[Bis20] D. Biswas. Privacy Preserving Chatbot Conversations. In *Workshop on Privacy-preserving Machine Learning (PPML)*, 2020.

[Bis21] D. Biswas. Compositional AI: the future of Enterprise AI. *Towards Data Science*, 2021.

[FTC21] FTC. California Company Settles FTC Allegations it deceived consumers about use of Facial Recognition in Photo Storage App, 2021.

[Gel21] R. Gellman. Fair Information Practices: A Basic History - Version 2.20. 2021.

[Gol09] O. Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2009.

[IEAL18] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin. Black-box Adversarial Attacks with Limited Queries and Information. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2137–2146. PMLR, 2018.

[KBdH09] F. Kerschbaum, D. Biswas, and S. de Hoogh. Performance Comparison of Secure Comparison Protocols. In *20th International Workshop on Database and Expert Systems Application (DEXA)*, pages 133–136, 2009.

[MMR+17] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54, pages 1273–1282. PMLR, 2017.

[MRTZ18] B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning Differentially Private Recurrent Language Models. In *International Conference on Learning Representations (ICLR)*, 2018.

[NSH19] M. Nasr, R. Shokri, and A. Houmansadr. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *IEEE Symposium on Security and Privacy (SP)*, pages 739–753, 2019.

[PAE+16] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar. Machine Learning with Privacy by Knowledge Aggregation and Transfer. In *Workshop on Privacy-preserving Machine Learning (PPML)*, 2016.

[PAE+17] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar. Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data. *arXiv*, abs/1610.05755, 2017.

[RG20] M. Rigaki and S. García. A Survey of Privacy Attacks in Machine Learning. *arXiv*, abs/2007.07646, 2020.

[SPTP+20] S. Sav, A. Pyrgelis, J. Troncoso-Pastoriza, D. Froelicher, J.-P Bossuat, J. S. Sousa, and J. Hubaux. POSEIDON: Privacy-Preserving Federated Neural Network Learning. *arXiv*, abs/2009.00349, 2020.

[SS15] R. Shokri and V. Shmatikov. Privacy-preserving Deep Learning. In *53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 909–910, 2015.

[ZJWW17] X. Zhang, S. Ji, H. Wang, and T. Wang. Private, Yet Practical, Multiparty Deep Learning. In *IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 1442–1452, 2017.