SAFE: Machine Unlearning With Shard Graphs

Yonatan Dukler[1*], Benjamin Bowman[1,2†*], Alessandro Achille[1*], Aditya Golatkar[1,2†],
Ashwin Swaminathan[1], Stefano Soatto[1]
AWS AI Labs[1], UCLA[2]
{dukler,bowmaben,aachille,agolatka,swashwin,soattos}@amazon.com

Abstract

We present Synergy Aware Forgetting Ensemble (SAFE), a method to adapt large models on a diverse collection of data while minimizing the expected cost to remove the influence of training samples from the trained model. This process, also known as selective forgetting or unlearning, is often conducted by partitioning a dataset into shards, training fully independent models on each, then ensembling the resulting models. Increasing the number of shards reduces the expected cost to forget but at the same time it increases inference cost and reduces the final accuracy of the model since synergistic information between samples is lost during the independent model training. Rather than treating each shard as independent, SAFE introduces the notion of a shard graph, which allows incorporating limited information from other shards during training, trading off a modest increase in expected forgetting cost with a significant increase in accuracy, all while still attaining complete removal of residual influence after forgetting. SAFE uses a lightweight system of adapters which can be trained while reusing most of the computations. This allows SAFE to be trained on shards an order-of-magnitude smaller than current state-of-the-art methods (thus reducing the forgetting costs) while also maintaining high accuracy, as we demonstrate empirically on fine-grained computer vision datasets.

1. Introduction

Large-scale neural networks are typically trained on large monolithic datasets. However, real world data often comes from many different sources which may require different treatment depending on their terms of use. In some cases, the terms can change, triggering the need to not just erase a portion of the data, but also remove its influence on the trained model. If the model is trained with the entire dataset in an undifferentiated fashion, even a request to remove a small fraction of the data may result in re-training the entire model on the complement. Considering the scale of large neural networks currently in use, re-training the model after each data erasure is costly. Thus, it is beneficial to develop methods to trace the influence of various segments of data onto the trained model, and to remove their effect if needed, especially as the scale of production models and the amount of training data continues to grow.

One simple and robust approach to forgetting is compartmentalizing the information of different subsets of data into distinct model parameters. In this scenario, the training data is split into disjoint shards, and different parameters or "adapters" are trained separately on each shard, and then ensembled to obtain a final model. The advantage of this approach is that, if the influence of a sample needs to be removed, only the parameters corresponding to its shard have to be retrained. Moreover if an entire source of data needs to be dropped, one can simply drop all the adapters corresponding to the shards containing its samples. There are, however, two main disadvantages to this approach. On the implementation side, there is an increase in storage/inference scaling with the multitude of adapters to ensemble. On a more fundamental level, since each adapter is trained independently on a fraction of the available data, it forfeits synergistic information that may be present in data stored in different shards. As the number of shards grows, adapters are trained on increasingly impoverished samples, leading to degraded performance relative to monolithic training (see Fig. 2).

Thus, practitioners have to choose a trade-off between increasing the accuracy of the model and minimizing the expected cost of forgetting a sample. In this work, we show that naive sharding, that is uniform partitioning of the training set, is suboptimal in many realistic use cases. Instead, shards should be constructed to maximize synergistic information available at training time. To that end, we introduce SAFE (Synergy Aware Forgetting Ensemble), an algorithm that leverages synergistic information between data shards to maximize accuracy for a given forgetting cost. We show that SAFE allows training on highly sharded data (with as

*Equal contributions
†Work done during an internship at AWS AI Labs.

many as 256 shards) with a 14% accuracy boost over uniform sharding for the same forgetting cost.

To realize this, we introduce the notion of a Shard Graph (SG) (see Fig. 1). Each node of the graph is a small shard of data in the original dataset. (Directed) edges between shards indicate that the corresponding adapter is not trained in isolation as usual, but also uses data from the shards it connects to, thus increasing the accessible synergistic information. On the downside, when a forget request is received, all nodes pointing to the forgotten node must be retrained, increasing the expected forgetting cost. Since different shards have different likelihood of receiving forget requests and contain different information, the problem becomes how to draw connections that maximizes the synergistic gain while minimizing the increase in expected forgetting cost. For example, shards that are unlikely to contain samples that need to be forgotten (e.g., a shard that contains synthetic data or highly vetted data) can be simultaneously connected to many other nodes significantly increasing accuracy without increasing the expected forgetting cost.

A SG may, however, have hundreds of nodes, thus raising the problem of how to train and perform inference with hundreds of adapters. SAFE addresses this problem using InCA adapters [15]. These are small modules connected to a frozen pre-trained backbone, which can be trained in parallel at less than the cost of fine-tuning a single monolithic model. SAFE can thus handle orders of magnitude more shards than other state-of-the-art algorithms. While the use of lightweight adapters may reduce accuracy, we found the loss in accuracy negligible when the chosen backbone is well aligned with the user data, and it is offset by the large reduction in forgetting cost.

We then present two extensions of SAFE. First, we consider the case where information about individual samples from connected shards is bounded using (ϵ, δ)-Differential Privacy, which can be interpreted as using a weighted graph with edges of weight ϵ. In situations where (ϵ, δ)-DP is an acceptable guarantee, this formulation can further reduce the expected forgetting cost by avoiding the need to always retrain all connected shards after a forget request. Second, we show that SAFE can be used to serve à-la-carte models [6], where each user can access a model trained only on a user-specific subset data, as long as the connectivity of the SG is compatible with the user access rights.

Empirically, we show that SAFE can reduce the cost of forgetting samples by an order of magnitude compared to similar forgetting algorithms, while maintaining a similar accuracy. While evaluation of forgetting algorithms is usually performed on simple datasets such as CIFAR, we show that SAFE can be applied to more complex and nuanced fine-grained computer vision benchmarks.

2. Related Work

The forgetting/machine-unlearning problem [8, 20] focuses on removing information from a trained machine learning model. One line of work [5, 51, 53, 31, 34, 6] involves splitting the dataset into multiple shards and training separate models for each shard. In such settings forgetting a sample will only affect the model trained on the subset containing the data. In [4] the authors enable forgetting for logistic regression by allowing the model to forget a class. The recent work [51] achieves a 1-class linear classifier trained on only a single class by making predictions based on class-centroids. The authors of [53] propose the model LegoNet which generalizes [5] by modifying the ensembling procedure to a select-and-average approach with the top-k most relevant sub-models in an instance-dependent manner. Such methods provide instant forgetting by removal of the corresponding model with low re-training time, however, they suffer from increased inference costs directly correlating with the number of shards.

Another line of work involves forgetting for a single deep network. Forgetting is difficult for deep networks [22, 23] due to the highly nonlinear nature of the parameterization and nonconvexity of the learning problem. Such works often use approximations to the non-linear landscape of a deep network to propose unlearning methods. In [28] the authors perform a newton update for unlearning, while in [21, 2] the authors train a linearization of a ResNet-50 starting from a pre-trained model. The linearity of the parameterization, enables convexity of the optimization so that one can forget specific samples and obtain upper bounds on the mutual information after a certain number of forgetting steps. However, such unlearning methods are approximate, and as a result require complete re-training of the model after a fixed number of iterations when the privacy budget is consumed. The works of [50, 45] provide algorithms and seek understanding of unlearning by caching models, and unrolling the iterative steps of the optimization problem.

The unlearning problem has also been explored in federated setups [7, 35, 48, 36, 25] where a client withdraws its data and its influence needs to be removed from the global model and other clients. This approach is different from sharding based methods where there is no notion of a centralized model, and a set of weak models need to be ensembled. Some recent works [38, 29, 11, 46, 43] provide stochastic unlearning algorithms (similar to [1]) with rigorous theoretical guarantees similar to the probabilistic guarantees in differential privacy [16] or algorithmic stability. [29] showed that the "perfect" unlearning algorithms like [5], provide perfect forgetting only for uniform requests, and not adaptive requests. Unlearning guarantees of approximate algorithms decay with the number of forget requests, which require full re-training of the model eventually. In one exten-

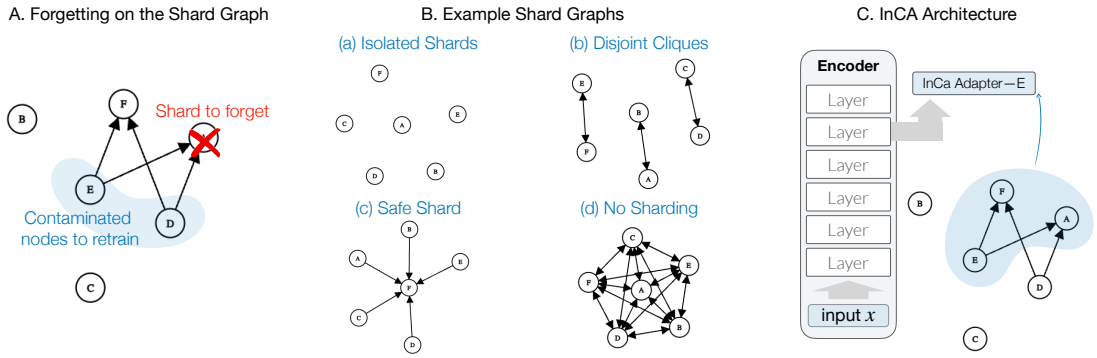

Figure 1. **(A) Forgetting on the Shard Graph.** Upon a forget request, the node containing the sample to be forgotten and all sub-models with outbound connections to the node are re-trained. **(B) Example Shard Graphs.** Prototypical examples of the Shard Graph include *(a) Isolated Shards:* (instant forgetting of whole shards, low accuracy). *(b) Disjoint Sharding:* (fast forgetting, competitive accuracy). *(c) Safe Shard:* one shard not likely to receive forget requests connected to all nodes (instant forgetting, competitive accuracy in appropriate settings). *(d) No Sharding:* (forgetting is infeasible, ideal performance). **(C) InCA Architecture.** We use lightweight cross-attention adapters acting upon the representations of a pretrained transformer encoder. Each node in the graph is associated to an InCA adapter trained on the union of the data of a node and all its outbound edges.

sion of SAFE, we provide a mixed learning algorithm (each shard is non-private with respect to itself, but private for other shards) using differential privacy [1, 24]. The work of [9, 52, 10, 39] studies the problem of unlearning on graph modalities for tasks such as edge prediction; we note our method is concerned with typical modalities albeit the synergy between data subsets is described via a directed graph (see Sec. 3 and description of the SG).

3. Forgetting on Shard Graphs

In this section we introduce the notion of a Shard Graph and the flexible forgetting approach it enables. We then show that existing shard-based forgetting algorithms can be reinterpreted as a degenerate case where the graph does not contain any edges. In the next section we will introduce SAFE, a forgetting algorithm that can take full advantage of the provided graph.

Shard Graphs. For ease of notation, in this section we will assume our label space is the discrete set $\mathcal{Y} = \{1, \dots, K\}$. The Shard Graph (SG) is a directed graph $G = (V, E)$ where the set of nodes $V := \{S_1, \dots, S_n\}$ denotes different data sources or data shards $S_i \subset \mathcal{X} \times \mathcal{Y}$, which are defined by the user based on their application. We use a directed edge $(S_i, S_j) \in E$ between two shards (i.e., "S_i points to S_j") to denote that when training the adapter corresponding to S_i we also allow access to data from S_j (see later). In all graphs we consider we assume implicitly that each node S_i has a self connection, $(S_i, S_i) \in E$, that is, the adapter of a shard is always trained on that shard's data.

Definition of forgetting. Consider an algorithm $\mathcal{A}$ (possibly stochastic like SGD) that, given a shard graph G as input, outputs a model trained on the shards of G. In this case, we denote with $\Theta \mapsto \mathbb{P}(\mathcal{A}(G) \in \Theta)$ the probability distribution of possible models produced by $\mathcal{A}$ given a graph G. Let G' be a shard graph where some of the data has been removed (for example, removing an entire node/shard or removing samples of the shard). For a training algorithm $\mathcal{A}$, we define a forgetting procedure $U(\mathcal{A}(G), G')$ which takes a trained model $M = \mathcal{A}(G)$ and the reduced graph G' to output a new model M' which is indistinguishable from a model trained directly on G'. To incorporate the stochasticity of $\mathcal{A}$, we require that the distribution of models outputted by the forgetting procedure $U(\mathcal{A}(G), G')$ matches $\mathcal{A}(G')$:

$$\mathbb{P}(U(\mathcal{A}(G), G') \in \Theta) = \mathbb{P}(\mathcal{A}(G') \in \Theta) \quad \text{for all events } \Theta.$$

Note that a forgetting algorithm can always trivially satisfy this by ignoring $\mathcal{A}(G)$ and retraining from scratch on the reduced graph G', that is, setting $U(\mathcal{A}(G), G') = \mathcal{A}(G')$. Doing so, however, is expensive especially if G' contains lots of data. The quality of a forgetting procedure is gauged by its ability to minimize the cost of forgetting while maintaining high accuracy.

Independent shard forgetting. Given a shard graph G, a trivial but efficient forgetting approach [5, 6, 51] is to discard the edges and train a separate adapter on each node $S_i \in V$. More precisely, let $A(S_i)$ denote the adapter trained on the data in S_i. The model corresponding to the graph G is then the ensemble of all the adapters:

$$\mathcal{A}(G) = \text{ensemble}(\{A(S_i)\}_{S_i \in V}). \tag{1}$$

The ensembling procedure is application specific, for instance, in classification we average the logits of the models.

Such $\mathcal{A}$ admit a simple forgetting procedure U. If requested to forget an entire node/shard in G, we simply need to drop

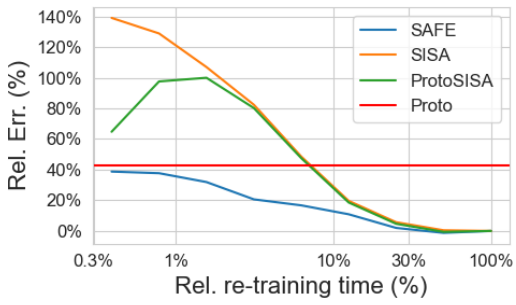

Figure 2. **Accuracy vs. expected cost of forgetting.** By reducing the number of shards one can improve the error (y-axis) at the expense of increasing the training time per forget request (x-axis). We show while the error of uniform sharding methods (SISA, orange), (ProtoSISA, green) grows significantly at fine-sharding scales, SAFE is able to maintain low error and uniformly outperforms the baseline of classification based on class prototypes (blue line). We report the avg. for the datasets in Tab. 1.

the corresponding adapter incurring a constant expected forgetting cost $O(1)$. When requested to forget a sample (or subset of samples) from a shard, we only need to retrain the corresponding adapter on the remaining data. Assuming that the cost of training an adapter is linear in the size of its training data, the expected cost to forget one sample is $O(|S|)$, where $|S|$ is the expected shard size.

When the data is divided into small shards, the expected forgetting cost is a fraction of the cost of retraining the whole model from scratch (all shards). However, as we see in Fig. 2, the accuracy of an ensemble model trained on many small shards (SISA, orange curve) can be significantly lower than the accuracy of a single model trained on all the data simultaneously (right-most point in the curve). This can be attributed to the loss of synergistic information when training independently on many small shards.

4. SAFE

SAFE aims to reduce the impact of the loss of synergistic information by allowing shards connectivity in a way that does not significantly increase the expected forgetting cost while improving performance. Our proposed method can be formulated easily in our proposed shard graph formalism.

Let G be a shard graph. Rather than training an adapter independently on the data of each shard, SAFE trains an adapter using also all the data contained in the connected shards. Formally, the model produced by SAFE is:

$$\mathcal{A}(G) = \text{ensemble}\left(\left\{A\left(\bigcup N(S_i)\right)\right\}_{S_i \in V}\right) \qquad (2)$$

where $\bigcup N(S_i)$ denotes the union of all the data contained in the shards connected to S_i – that is, its outbound neighborhood $N(S_i)$ (see Fig. 1C). Note that, when the graph G doesn't have any edges, Eq. (2) reduces to Eq. (1).

Training on the union of data from connected shards exploits the synergistic information among those shards to improve accuracy at the expense of increasing the forgetting cost. However, depending on the structure of the data and the likelihood of forget requests, the increase in accuracy can greatly outweigh the increase in forgetting cost.

Forgetting with SAFE. Unlike before, if a sample in a shard is deleted, we need to retrain not only the adapter corresponding to that shard, but also the adapters of all shards pointing to it. The cost of the procedure scales with the total amount of training data that needs to be revisited to train the adapters. Letting $x \in S_i$ be a sample to be forgotten, we can write as

$$M_x := \bigcup\{\bigcup N(S_j) : S_i \in N(S_j)\} \qquad (3)$$

the total data needed to retrain the adapters of all shards S_j that point to S_i. Hence, the expected cost of forgetting depends on the expected size of M_x, which in turns depends on the graph topology. We now analyze some interesting cases, which will inform our experimental setup.

Random connectivity. Suppose each node is connected to d other nodes uniformly at random. Then in expectation (see Appendix) we have:

$$\mathbb{E}|M_x| = \Theta(|S|d^2).$$

Which scales quadratically with the degree.

Partition in disjoint cliques. Consider now the case where the graph is partitioned into disjoint cliques of size d (see Fig. 1B (b)). In this case, all the $N(S_j)$ in Eq. (3) perfectly coincide and the union corresponds to $N(S_j)$. This leads to an expected forget cost that scales with the size d of the shard's clique

$$\mathbb{E}|M_x| = d \cdot |S|.$$

In particular, compared to the random connection case, the cost is linear instead of quadratic in the degree of the nodes.

Based on this analysis, we focus on graphs that can be represented as a union of disjoint cliques, as it leads to lower cost of forgetting while allowing the same amount of connectivity. Finally, the case where an entire shard is deleted, rather than a single example, has similar analysis. The corresponding adapter needs to be removed from the ensemble, and all adapters of shards pointing to it need to be retrained. The expected cost scales in the same way as before as a function linear with d.

Refined Shard Graph. In practice, it may often happen that different shards contain samples from different classes, or even from a single class. To deal uniformly with all

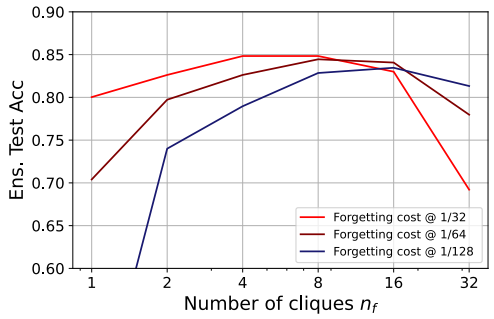

Figure 3. **Shard graph class composition** We report the test accuracy for MIT-67 accuracy under different shard topologies. The different curves correspond to different effective re-training times ($1/n$) as compared with standard training (smaller means faster). The points on each curve correspond to partitioning in different numbers n_f of fine cliques.

the cases — and to enable finer level of sharding and thus faster forgetting — it is convenient to restrict the adapters of SAFE to binary classifiers for each label present in the adapter's training set. This can be seen as refining the graph so that each node contains data from only one class (its positive samples) with the negative examples coming from the connected nodes. Formally, given a shard $S \subset \mathcal{X} \times \mathcal{Y}$ we define the label occurrence map $L(S) = \{y : (x, y) \in S\}$. Furthermore, for each label $k \in \mathcal{Y}$, we define the refined shard $S^{(k)} = \{(x, y) \in S : y = k\}$. Then given a shard graph $G = (V, E)$ we construct a refined vertex set

$$V' := \{S^{(k)} : S \in V, \quad k \in L(S)\}$$

and edge set

$$E' := \{(S^{(h)}, S'^{(k)}) : (S, S') \in E, h \in L(S), k \in L(S')\}.$$

Hence, for each node $S^{(i)} \in V'$ we train a binary classifier where the positive examples come from $S^{(i)}$ and the negative examples come from the all the connected nodes.

5. Synergistic Bilevel Sharding

We now tackle the question of how to generate a graph structure that increases the synergistic information for a given expected forgetting budget. Forgetting methods usually split the data in uniform shards. However, we note that samples from the same classes, or related classes, have more synergistic information, so should preferentially appear in the same clique, even if this means that the clique may not contain all classes due to size constraints (note that, as mentioned before, we do not need the adapters to see all classes since each is a binary one-vs-all classifier). Following this intuition, we suggest the following strategy: We first split the data in n_c disjoint "coarse" shards using class-balanced subsampling. For each coarse shard, we draw the nodes corresponding to each class and partition them randomly into cliques of size d, i.e., each clique contains examples from d classes. This results in a number $n_f = n_{\text{classes}}/d$ of "fine" cliques per coarse shard, for a total of $n = n_c \cdot n_f$ cliques. We note that the expected forgetting costs scales with $1/n$, i.e., the amount of data in each clique. Partitioning the data uniformly, as commonly done, is equivalent to selecting $n_f = 1$.

Optimal graph structure. Since the expected forgetting cost is the same as long as the product $n = n_c \cdot n_f$ remains constant, we check what n_f produces the best model for a fixed n. Low n_f increases the variety of the classes in each shard at the expense of loss of synergistic information, while high n_f increases the synergistic information but each model sees a smaller subset of classes. In Fig. 3 we plot the trade-off on the MIT-67 dataset. We see that indeed uniform sharding ($n_f = 1$) is never optimal, and that using a higher (but not too high) values of n_f is essential to get good accuracy when aiming for a low forgetting time, and can improve accuracy by more than 15%, thus supporting the design choices for our method. Random partitioning across coarse and fine levels also makes our method robust against non-adaptive forget requests.*

6. Efficient implementation of SAFE

Training independent large models on hundreds of shards leads to prohibitive storage and inference costs, and increases the expected cost of retraining for each forget request. This suggest using a shared backbone and a series of small shard-specific adapters. We use a variant of Open-InCA cross-attention adapters [15], which lend themself easily to massive parallel training and inference (See Appendix A for details) while also providing high-accuracy on difficult fine-grained classification tasks. With Open-InCA adapters, we can efficiently compartmentalize the training data of each shard to its corresponding Open-InCA class parameters.

InCA adapters [15]. Let $\mathbf{z} = (z_1, \ldots, z_n) = f_w(\mathbf{x})$ be the activation map produced by a frozen vision encoder $f_w(x)$. Given a set of learnable class-specific query tokens $\mathbf{q} = (q_1, \ldots, q_K)$, the output $\mathbf{y} = \text{CA}_\theta(\mathbf{z}, \mathbf{q}, \mathbf{v})$ of InCA adapters CA are the logits $\mathbf{y} = (y_1, \ldots, y_n)$ defined by

$$y_i = v_i \cdot e_i$$
$$\mathbf{e} = \text{cross-attention}_\theta(\mathbf{z}, \mathbf{q})$$

where θ are the parameters of the cross-attention layer and $\mathbf{v} = (v_1, \ldots, v_k)$ is a learnable set of vectors v_k (which can be interpreted as binary linear classifiers). To keep the notation uncluttered, in the following we write q_i to denote

*When the graph structure is data dependent, DP techniques may be required to prevent information leakage from it.

both the query vectors and the corresponding classification vector v_i. An important property of Open-InCA for our application is compositionality. Let $[\mathbf{q}, \mathbf{q}']$ denote the concatenation of $\mathbf{q}$ and $\mathbf{q}'$, then

$$\mathrm{CA}_\theta(\mathbf{z}, [\mathbf{q}, \mathbf{q}']) = [\mathrm{CA}_\theta(\mathbf{z}, \mathbf{q}), \mathrm{CA}_\theta(\mathbf{z}, \mathbf{q}')]. \quad (4)$$

This suggests that, rather than training a separate model on each node, we can train individual node specific q_i together by concatenating them to obtain the final model. In addition to the compositionality by learning different queries q_i for each classifier, Open-InCA remains expressive in selecting sub-task specific representations for each classifier.

Applying InCA to SAFE. We freeze and use the same cross attention weights CA_θ which are shared across all nodes, with frozen parameters θ.[†] Then, for each reduced shard $S_k^{(i)}$ we create corresponding queries q_i^k, and train the resulting binary one-versus-all classifier

$$y_i^k(\mathbf{z}) = \mathrm{CA}_\theta(\mathbf{z}, q_i^k)$$

on the data of all connected shards using a binary cross entropy (BCE) loss. If there is a clique of nodes in the refined shard graph, we can group them together and train simultaneously instead with the cross entropy loss. At inference time, we utilize the compositionality of the CA_θ adapters, by computing the concatenation of all q_i^k sharing the computation of all the adapters in a single forward pass and easily ensemble the resulting logits, leading to the final logits,

$$y_i(\mathbf{z}) = \mathrm{mean}_k \left(\mathrm{CA}_\theta(\mathbf{z}, \mathbf{q}, \mathbf{v}) \right).$$

In the equation above the embedding $\mathbf{z} = f_w(\mathbf{x})$ is of a test sample $\mathbf{x}$. For classification, we select the class corresponding to the highest logit. Training each q_i^k and v_i^k sequentially is still expensive, and wastes computation since the same sample can be used as a negative to train multiple connected nodes. Instead, since for the BCE loss the gradients of different q_i are independent, we can train all the q_i at the same time over a single epoch on the whole (re)-training set, provided the loss is appropriately masked to prevent information from unconnected shards from transpiring into each q_i (see Appendix A for details). Further since InCA does not backpropagate through the backbone f_w, we can also pre-compute and store the embeddings $\mathbf{z}$. Using all these techniques, we are able to train hundreds of shards at the same time in under 10 minutes on a single GPU.

7. Prototypical classifier

Class prototypes are another viable approach to forgetting used by ARCANE [51]. Given a dataset $D = \{(x_i, y_i)\}_{i=1}^N$ and an embedding $z = f_w(x)$, we define the prototype of the k-th class as

$$p_k = \frac{1}{N_c} \sum_{(x,y) \in D^{(k)}} f_w(x)$$

where $D^{(k)}$ are the samples of class k and $N_c = |D^{(k)}|$. We can then construct a simple linear classifier

$$y_k(z) := d_{\cos}(z, p_k),$$

where $d_{\cos}$ denotes the cosine distance. Such "prototypical" classifiers allow instantaneous forgetting: to forget a training sample (x_i, y_i) we just need to remove it from its class prototype $p_{y_i} \mapsto \frac{N_c\, p_{y_i} - f_w(x_i)}{N_c - 1}$. On the other hand, this classifier has suboptimal classification accuracy compared to a trained classifier on large shards of data.

However, when the shards consists only of a few samples, classifiers trained on individual shards may overfit. In such cases, the prototypical classifier can be used to provide an inductive bias [44]. Since the added computational and space complexity to use the prototypical classifier is negligible, we combine it into the SAFE model using the following expression:

$$\mathrm{SAFE}(z) = (1 - \lambda) \cdot M(G)(z) + \lambda \cdot \mathrm{Proto}(z),$$

where $M(G)$ is the ensemble model in Eq. (2), Proto denotes the prototype-based classifier and $\lambda = \exp\left(-\frac{d \cdot |S|}{100}\right)$ is an interpolation weight that relies more on the prototypical classifier when the amount of data $d|S|$ used to train each adapter is small.

8. Extensions of SAFE

Stochastic forgetting for reduced cost. In the previous sections we presented forgetting approaches that use the edges in the shard graph to define complete usage of a shard or no usage (when there is no connection). Below we present the notion of limited shard information defined via differential privacy (DP), in which data of different nodes (data shards) of the graph are shared to a sub-model in a differentially private fashion. This can be defined by an edge weight that bounds the information shared about each sample, and can be interpreted as the probability a sample is identified in the training set. Specifically consider the binary classifier sub-models introduced in Section 6, for the positive-negative samples defined by the graph topology we assign complete usage to the positive node and limited usage to the outbound connections of the negative nodes. This corresponds to a special case of mixed DP [21], which results in a simple training algorithm (SAFE-DP) where during each epoch, each node is trained with its own data without privacy, and with data from the neighbouring node using DP. This algorithm satisfies an approximate definition

[†] We find that simply using a random initialization for θ provides a good and unbiased performance across tasks.

			Number of shards								
			8			64			256		
Dataset/Method	**No Sharding**	**Prototypes**	**SAFE**	**SISA**	**ProtoSISA**	**SAFE**	**SISA**	**ProtoSISA**	**SAFE**	**SISA**	**ProtoSISA**
Caltech-256	94.3%	93.2%	94.0%	93.6%	93.6%	93.5%	86.7%	86.8%	93.3%	84.6%	90.5%
CIFAR-100	83.1%	71.2%	84.1%	84.0%	84.0%	82.2%	80.7%	80.7%	80.9%	77.6%	77.7%
CUB-200	88.3%	85.8%	86.1%	83.9%	83.9%	83.5%	65.1%	67.8%	84.5%	63.7%	83.5%
DTD	77.8%	73.8%	77.1%	76.1%	76.1%	75.1%	65.1%	66.7%	74.2%	52.6%	71.2%
MIT-67	87.9%	85.8%	86.9%	86.9%	86.9%	86.0%	81.5%	81.7%	86.4%	77.9%	84.4%
Stanford Cars	75.7%	41.0%	62.1%	55.8%	55.8%	47.4%	17.1%	17.8%	36.2%	7.0%	25.5%
Stanford Dogs	87.9%	88.0%	89.2%	89.2%	89.2%	88.3%	83.1%	83.2%	87.8%	79.8%	83.5%
Average Acc.	85.0%	77.0%	**82.8%**	81.4%	81.4%	**79.4%**	68.5%	69.3%	**77.6%**	63.3%	73.8%

Table 1. **Accuracy of Unlearning Approaches:** We report accuracy and re-training efficiency (measured by sharding level for re-training) on a diverse set of visual classification datasets. Forgetting a sample is equivalent to re-training the shard containing the sample. The retraining time is inversely proportional to the number of shards. SAFE allows sharding up to 256 subsets without significantly compromising accuracy. For each level of sharding (for a fixed number of shards) we try different SG topologies and report the best results in the table. We note that for 8 shards the accuracies for SISA and ProtoSISA are the same due to λ being exponentially small when the shards are large.

of forgetting, more precisely, $\mathcal{A}(G)$ is called an (α, β)-unlearning algorithm if

$$\mathbb{P}(U(\mathcal{A}(G), G') \in E) \leq e^{\alpha}\mathbb{P}(\mathcal{A}(G') \in E) + \beta$$

for all events E. This definition measures the privacy leakage in terms (α, β) using group DP. We enable each user to specify their target α_{target} (such that $\beta < 1$), and employ privacy accounting to identify budget overflow ($\beta > 1$) for sequential forget requests. Such an algorithm is useful in adversarial conditions for protection against worst case adaptive forget requests.

À-la-carte models via the Shard Graph. The problem of constructing a unique model for a particular user that only uses data consistent with their access permissions and personal preferences, *i.e.* the "à-la-carte learning" problem [6], can be solved using SAFE. For a given user one can identify which nodes in the graph the user is unable or unwilling to access. Based on this, one can ensemble only the adapters trained on the nodes with no outbound connections to the ineligible nodes. Alternatively, one could simulate an artificial "forget request" by dropping all the data that the user is unable to access and retraining the corresponding InCA adapters. Since InCA adapters with cached activations can be trained in seconds, one could potentially perform this training efficiently to serve custom models to different users on-demand.

9. Experiments

Model. In our experiments we use as encoder a VIT-L/16 transformer architecture [14] pretrained on ImageNet-21k at 224 image resolution.[‡] We use the InCA adapters from [15], in particular we train the head and queries, but keep the cross-attention layer frozen and shared between all shards. We train each adapter with AdamW for 30 epochs using cosine annealing starting from lr $= 0.05$ and weight decay of 10^{-4}. See the Appendix for full details.

Datasets. We evaluate on the following vision classification datasets, covering fine-grained classification tasks and domain shifts with respect to the ImageNet pretraining: CUB-200 [47], MIT-67 [41], Caltech-256 [27], CIFAR-100 [33], Describable Textures (DTD) [13], Stanford Cars [32], and Stanford Dogs [30].

Baselines. We compare our method against two main classes of methods for forgetting: Prototypes (motivated by ARCANE[§] [51]) uses a prototype based classifier (Section 7), which allows constant time forgetting but which cannot be trained to improve accuracy, SISA [5] performs forgetting by training an ensemble of models on shards created by sampling uniformly at random. This corresponds to using $n_f = 1$ when structuring the bilevel sharding (Section 5). Like in SAFE, increasing the number n of shards leads to lower forgetting time but also lower accuracy. We re-implement SISA using the same architecture and training scheme as SAFE for a direct comparison. [5] uses further slicing of each shard (through intermittent model checkpoints) to further speed-up re-training complementarily. Slicing can be similarly incorporated in SAFE, but we isolate slicing from our analysis as it is an orthogonal and complementary approach that can be applied in each method (with increased storage costs).

Comparison of SAFE with baselines. In Fig. 2 we plot the

[‡]We use the `vit_large_patch16_224_in21k` pre-trained model in the `timm` library [49].

[§]In [51] they work with smaller models (e.g. ResNet-18 with 11M params.) and train a separate embedding for each class, whereas we take the embeddings of a fixed large pretrained transformer (305M params.).

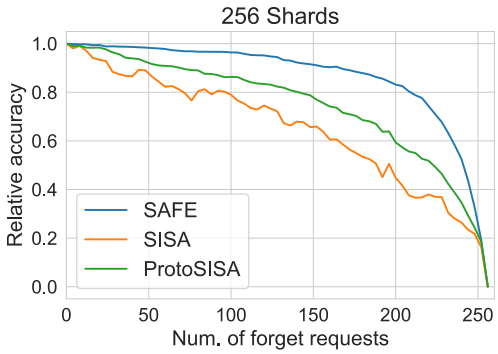

Figure 4. **Instant Forgetting.** We simulate a series of forget requests where for each forget request we drop an entire shard of data without retraining. We plot the relative accuracy averaged across the 7 datasets for the methods SAFE, SISA, and ProtoSISA for 256 shards.

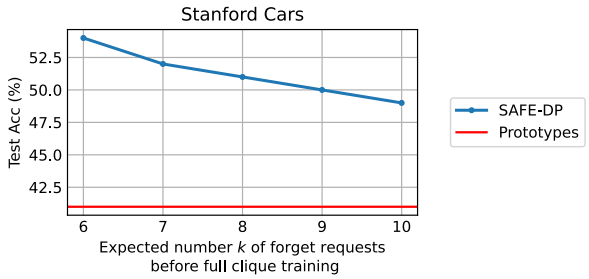

Figure 5. **Stochastic forgetting.** Training with DP on the outbound nodes we can satisfy k forget requests before having to retrain the whole graph, at the expense of lower accuracy.

average trade-off between accuracy and forgetting time for SAFE and other methods from the literature (SISA [5] and Prototypes [51]) across the datasets in Table 1 for various target forgetting times. We see that SAFE performs uniformly better than SISA and Prototypes for all forgetting budgets. Thanks to the ability to train, it outperforms Prototypes for higher forgetting time budgets, while it significantly outperform SISA in the low-forgetting-time regime (high-sharding) due to better use of synergistic information.

Domain shift. In Table 1 we see that the accuracy of SAFE and other forgetting methods decreases more rapidly with the sharding level for datasets that have a significantly different distribution from the ImageNet-21k pretraining of the backbone (e.g., DTD and Stanford Cars). We hypothesize that this is because the synergistic information of different samples is already contained in the backbone when the data is similar to the pre-training, hence the loss due to sharding is less influential. However, thanks to its better handling of synergistic information, we see that SAFE performs significantly better than the other baselines on difficult domains.

Ablations. We can ask whether the better performance of SAFE is due to the use of prototypes, the synergistic information or both. We have seen in Figure 3 that synergistic information alone improves over SISA ($n_f = 1$). In Fig. 2 we also show the result of a further baseline, ProtoSISA, obtained by adding a prototype classifier to SISA using the same weighting scheme used for SAFE. We see that while adding prototypes to SISA boosts performance, SAFE still significantly outperforms both other methods, showing that indeed both aspects are important.

Instant forgetting. In certain situations, a service provider will need to satisfy a forget request instantly and thus will need to drop an entire data source without retraining, meaning that all adapters using those samples will need to be dropped without being retrained and replaced. To investigate how robust SAFE, SISA, and ProtoSISA are to such requests, in Fig. 4 we plot the relative accuracy of SAFE, SISA, and ProtoSISA after a series of forget requests for ensembles with 256 shards. We see that SAFE exhibits a smaller decline in relative accuracy in the presence of forget requests, and uniformly outperforms SISA and ProtoSISA. For all methods the marginal decline in accuracy increases as the number of forget requests increases, suggesting that the ensemble becomes more sensitive as additional predictors are dropped.

Stochastic forgetting. In Section 8 we propose a DP-based mechanism that allows shards to receive up to a given number k of forget requests without having to retrain a whole clique, at the expense of decreased accuracy due to the bound on information imposed by DP. In Fig. 5 we show this trade-off on a dataset with significant distribution shift (Stanford Cars). The model trained, for example, with SAFE-DP ($k = 8$) provides lower accuracy than SAFE ($n = 8$) (51.0% instead of 62.1%), but has a better worst case guarantee (it can accommodate 8 sequential forget requests before retraining). Thus SAFE-DP may trade-off a worst case privacy guarantee for model accuracy. This is especially useful in settings with adversarial forget requests [29].

10. Multi-Domain synergy experiments

Next we study SAFE's ability to harness synergistic effects for tasks involving multiple domains. For this we consider the 4-domain challenge, DomainNet-126 [42] that is a curated subset of the extended DomainNet suite [40]. To best evaluate synergy between domains, we ensure each domain's dataset is of the same size, and we sub-sample the training and test set of each domain to be of fixed size and with a balanced distribution of classes. This results in 6300 training samples and 2520 test samples for each domain.

To test the benefits of SAFE for the multi-domain task of predicting among the 126 different common categories,

Method	All	Clipart	Sketch	Real	Painting
SISA (in-domain)	67.88	67.70	59.25	85.12	59.44
SAFE (in-domain)	73.41	73.77	66.19	88.89	64.80
SISA (synergy)	69.97	69.84	62.02	86.67	61.35
SAFE (synergy)	**76.11**	**77.14**	**68.10**	**90.40**	**68.81**

Table 2. **Multi-domain unlearning on DomainNet-126** We report the test accuracy when using different shard graph topologies on DomainNet-126. The column "All" corresponds to evaluation on the union of the 4 domains, whereas the other four columns correspond to the test accuracy on the single domain.

now with images coming from different domains, we consider different SG topologies. We define the topologies tested as follows.

- **SISA In-Domain** Each domain is trained separately with SISA-style uniform sharding with 16 shards in each domain (64 total shards).
- **SISA Cross-Domain** The training datasets are merged into a single cross-domain training dataset that is used for SISA-style uniform sharding (shards are maintained to be of same size as in "SISA In-Domain", *e.g.* 64 total shards).
- **SAFE In-Domain** SAFE is applied to each domain separately, which creates cliques containing subsets of classes with each clique having the same re-training costs as the previous two approaches.
- **SAFE Cross-Domain** The cliques are constructed to contain synergistic connections between different domains with the overall clique size remaining the same.

In Table 2 we evaluate the different approaches for a 64 shard level. For in-domain topologies this corresponds to 16 shards per-domain. For SAFE we use 4 coarse shards and 4 fine shards. We observe that by modifying the Shard Graph topology for SAFE and allowing for shard connections between different domains, SAFE is capable of learning more accurate representations at the same training and unlearning costs.

11. Conclusion

We introduced the Shard Graph, a directed graph describing the access relations between data sources for training. This graph informs the creation of SAFE: an ensemble of lightweight adapters trained on the datasets specified by the graph structure. By constructing the graph to maximize synergies between datasets while minimizing connections and retraining time, we are able to handle an order of magnitude more shards than competing methods, and achieve a 14% accuracy boost over competing methods for the same forgetting cost. We conclude that maximizing synergistic information while minimizing dataset overlap is the fundamental trade-off at the core of compartmentalization-based forgetting, which so far has been under-explored. In some cases, the accuracy of our method may be limited by the use of light-weight adapters. However, InCA adapters demonstrate high accuracy while allowing us to efficiently train many adapters in parallel without information leakage and permit fast retraining, which make them conducive to compartmentalization-based forgetting. Rather, we find that the bottleneck in accuracy is mainly due to the loss of synergistic information due to sharding which we alleviate with SAFE. Finally, optimizing the shard graph structure to utilize additional properties in the data without leaking sensitive information is an important problem which we leave to future work.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016. 2, 3, 6

[2] Alessandro Achille, Aditya Golatkar, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. Lqf: Linear quadratic fine-tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15729–15739, 2021. 2

[3] Paul Barham, Aakanksha Chowdhery, Jeff Dean, Sanjay Ghemawat, Steven Hand, Daniel Hurt, Michael Isard, Hyeontaek Lim, Ruoming Pang, Sudip Roy, Brennan Saeta, Parker Schuh, Ryan Sepassi, Laurent Shafey, Chandu Thekkath, and Yonghui Wu. Pathways: Asynchronous distributed dataflow for ml. In D. Marculescu, Y. Chi, and C. Wu, editors, *Proceedings of Machine Learning and Systems*, volume 4, pages 430–449, 2022. 8

[4] Thomas Baumhauer, Pascal Schöttle, and Matthias Zeppelzauer. Machine unlearning: linear filtration for logit-based classifiers. *Machine Learning*, 111, 07 2022. 2

[5] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 141–159, 2021. 2, 3, 7, 8

[6] Benjamin Bowman, Alessandro Achille, Luca Zancato, Matthew Trager, Pramuditha Perera, Giovanni Paolini, and Stefano Soatto. A-la-carte prompt tuning (apt): Combining distinct data via composable prompting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14984–14993, June 2023. 2, 3, 7

[7] Xiaoyu Cao, Jinyuan Jia, Zaixi Zhang, and Neil Zhenqiang Gong. Fedrecover: Recovering from poisoning attacks in federated learning using historical information. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 1366–1383, 2023. 2

[8] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pages 463–480, 2015. 2

[9] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. Graph unlearning. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 499–513, 2022. 3

[10] Eli Chien, Chao Pan, and Olgica Milenkovic. Certified graph unlearning. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*, 2022. 3

[11] Rishav Chourasia, Neil Shah, and Reza Shokri. Forget unlearning: Towards true data-deletion in machine learning. *arXiv preprint arXiv:2210.08911*, 2022. 2

[12] Aakanksha et al. Chowdhery. Palm: Scaling language modeling with pathways, 2022. 8

[13] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014. 7, 8

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 7

[15] Yonatan Dukler, Alessandro Achille, Hao Yang, Varsha Vivek, Luca Zancato, Ben Bowman, Avinash Ravichandran, Charless Fowlkes, Ashwin Swaminathan, and Stefano Soatto. Introspective cross-attention probing for lightweight transfer of pre-trained models. *arXiv preprint arXiv:2303.04105*, 2023. 2, 5, 7, 1

[16] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014. 2, 6

[17] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. 7, 8

[18] Andrea Gesmundo. Multipath agents for modular multitask ml systems. *arXiv preprint arXiv:2302.02721*, 2023. 8

[19] Andrea Gesmundo and Jeff Dean. munet: Evolving pretrained deep neural networks into scalable auto-tuning multitask systems. *arXiv preprint arXiv:2205.10937*, 2022. 8

[20] Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. Making ai forget you: Data deletion in machine learning. *Advances in neural information processing systems*, 32, 2019. 2

[21] Aditya Golatkar, Alessandro Achille, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. Mixed-privacy forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 792–801, June 2021. 2, 6

[22] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[23] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In *European Conference on Computer Vision*, pages 383–398. Springer, 2020. 2

[24] Aditya Golatkar, Alessandro Achille, Yu-Xiang Wang, Aaron Roth, Michael Kearns, and Stefano Soatto. Mixed differential privacy in computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8376–8386, 2022. 3, 6

[25] Jinu Gong, Joonhyuk Kang, Osvaldo Simeone, and Rahif Kassab. Forget-svgd: Particle-based bayesian federated unlearning. In *2022 IEEE Data Science and Learning Workshop (DSLW)*, pages 1–6. IEEE, 2022. 2

[26] Sivakanth Gopi, Yin Tat Lee, and Lukas Wutschitz. Numerical composition of differential privacy. *Advances in Neural Information Processing Systems*, 34:11631–11642, 2021. 6

[27] Griffin, Holub, and Perona. Caltech 256, Apr 2022. 7, 8

[28] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3832–3842. PMLR, 13–18 Jul 2020. 2

[29] Varun Gupta, Christopher Jung, Seth Neel, Aaron Roth, Saeed Sharifi-Malvajerdi, and Chris Waites. Adaptive machine unlearning. *Advances in Neural Information Processing Systems*, 34:16319–16330, 2021. 2, 8

[30] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011. 7, 8

[31] Korbinian Koch and Marcus Soll. No matter how you slice it: Machine unlearning with sisa comes at the expense of minority classes. In *First IEEE Conference on Secure and Trustworthy Machine Learning*. 2

[32] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 7, 8

[33] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 7, 8

[34] Vinayshekhar Bannihatti Kumar, Rashmi Gangadharaiah, and Dan Roth. Privacy adhering machine un-learning in nlp. *arXiv preprint arXiv:2212.09573*, 2022. 2

[35] Gaoyang Liu, Xiaoqiang Ma, Yang Yang, Chen Wang, and Jiangchuan Liu. Federated unlearning. *arXiv preprint arXiv:2012.13891*, 2020. 2

[36] Yi Liu, Lei Xu, Xingliang Yuan, Cong Wang, and Bo Li. The right to be forgotten in federated learning: An efficient realization with rapid retraining. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 1749–1758. IEEE, 2022. 2

[37] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*, pages 263–275. IEEE, 2017. 6

[38] Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In *Algorithmic Learning Theory*, pages 931–962. PMLR, 2021. 2

[39] Chao Pan, Eli Chien, and Olgica Milenkovic. Unlearning nonlinear graph classifiers in the limited training data regime. *arXiv preprint arXiv:2211.03216*, 2022. 3

[40] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1406–1415, 2019. 8

[41] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 413–420, 2009. 7, 8

[42] Kuniaki Saito, Donghyun Kim, Stan Sclaroff, Trevor Darrell, and Kate Saenko. Semi-supervised domain adaptation via minimax entropy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 8

[43] Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems*, 34:18075–18086, 2021. 2

[44] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017. 6

[45] Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. Unrolling sgd: Understanding factors influencing machine unlearning. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 303–319. IEEE, 2022. 2

[46] Enayat Ullah, Tung Mai, Anup Rao, Ryan A Rossi, and Raman Arora. Machine unlearning via algorithmic stability. In *Conference on Learning Theory*, pages 4126–4142. PMLR, 2021. 2

[47] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 7, 8

[48] Junxiao Wang, Song Guo, Xin Xie, and Heng Qi. Federated unlearning via class-discriminative pruning. In *Proceedings of the ACM Web Conference 2022*, pages 622–632, 2022. 2

[49] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 7

[50] Yinjun Wu, Edgar Dobriban, and Susan Davidson. Deltagrad: Rapid retraining of machine learning models. In *International Conference on Machine Learning*, pages 10355–10366. PMLR, 2020. 2

[51] Haonan Yan, Xiaoguang Li, Ziyao Guo, Hui Li, Fenghua Li, and Xiaodong Lin. Arcane: An efficient architecture for exact machine unlearning. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 4006–4013. International Joint Conferences on Artificial Intelligence Organization, 7 2022. Main Track. 2, 3, 6, 7, 8

[52] Xiangrong Zhu, Guangyao Li, and Wei Hu. Heterogeneous federated knowledge graph embedding learning and unlearning. In *Proceedings of the ACM Web Conference 2023*, WWW '23, page 2444–2454, New York, NY, USA, 2023. Association for Computing Machinery. 3

[53] Xiaofei Zhu, Jie Wu, Ling Zhu, Jiafeng Guo, Ran Yu, Katarina Boland, and Stefan Dietze. Exploring user historical semantic and sentiment preference for microblog sentiment classification. *Neurocomputing*, 464:141–150, 2021. 2