Supplementary Material

A. Parallel and compartmentalized training of SAFE

In the following section we review the Open-InCA architecture and its use for efficient training of compartmentalized adapters in SAFE. In our training we are able to train all adapters in parallel while having exact control of each parameter's access to samples' gradients. This approach enables us to define the Shard Graph flexibly and provide custom "data access" to each adapter while training all of SAFE's adapters in parallel.

Open-InCA. In the work of [15] it is shown that lightweight cross-attention adapters attached to intermediate network representations are capable of extracting relevant representations and achieving competitive accuracy relative to expensive full fine-tuning. For regular learning, given an activation map $\mathbf{z}$ of a pre-trained model, the InCA adapter structure is defined as

$$\begin{aligned}\mathbf{e} &= \text{cross-attention}_{\theta_1}(\mathbf{z}, [q])\\ \mathbf{y} &= \text{Linear}_{\theta_2}(\mathbf{e}).\end{aligned}$$

By learning the cross-attention layer parameters and the query q, the adapter is capable of extracting task-relevant representations from the activation map $\mathbf{z}$.

For more flexible learning scenarios the authors present Open-InCA, where each class prediction is computed via a separately learned query and class-head parameters,

$$\begin{aligned}\mathbf{e} &= \text{cross-attention}_{\theta_1}(\mathbf{z}, [q_1, \dots q_C])\\ \mathbf{y} &= \text{Diag-Linear}_{\theta_2}(\mathbf{e}).\end{aligned}$$

Here $\text{Diag-Linear}_{\theta_2}$ is a mapping that takes the query embeddings $[e_1, \dots, e_C]$ and acts on them "diagonally" through the classification vectors $[v_1, \dots, v_C]$:

$$y_i := v_i \cdot e_i.$$

We follow the Open-InCA approach which is originally defined for class incremental learning, but for our settings we would like to achieve complete isolation of the learning of each adapter.

Adapter isolation. To achieve isolated learning of each adapter we apply Open-InCA's "query only learning" [15]. This amounts to not optimizing the cross-attention layer weights θ_1 and only training $[q_1, \dots q_c]$ and the separate classification head parameters $[v_1, \dots v_c]$. Nonetheless, applied with the traditional softmax and Cross-Entropy loss, query only training *still* leaks information of each sample in the batch to each adapter. Recall that softmax is defined as

$$[\text{softmax}(\mathbf{y})]_i = \frac{\exp(y_i)}{\sum_{k=1}^{C}\exp(y_k)}.$$

The softmax's gradient of each class prediction will be affected by the predictions of all other adapters (due to the denominator), thereby leaking information. To rectify this, we replace the softmax with a sigmoidal mapping $\sigma(y_i) = \frac{1}{\exp(-y_i)+1}$, so the loss of each adapter prediction relies solely on the adapter's logit value y_i. Using the sigmoidal mapping enables us to use SAFE in more flexible learning settings. This is because now each node in the shard graph can be responsible for learning just a single binary classifier for a particular class.

Transitioning to Open-InCA with sigmoidal loss results in hundreds or even thousands of individually learned binary classifiers. Trained sequentially the vast number of adapters will result in prohibitively long training time.

Loss masking. Instead we show that we can learn all of the adapters within an Open-InCA adapter in parallel while still isolating information via loss masking. As stated in Eq. (4) of the manuscript the different queries of the Open-InCA

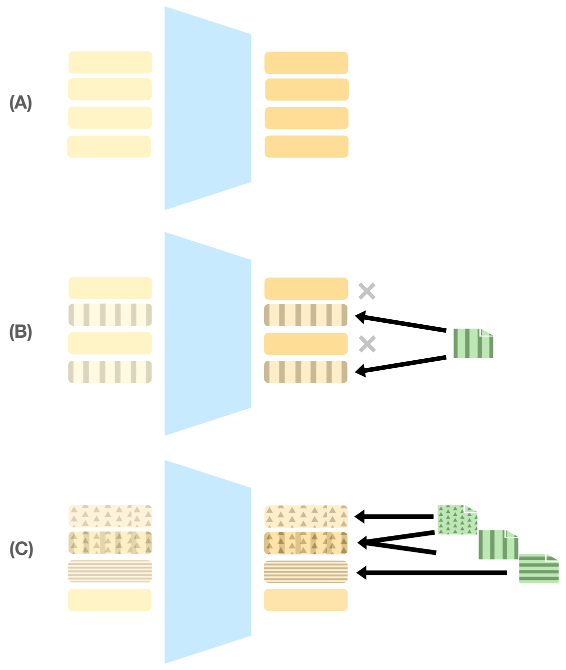

Figure 6. **SAFE Compartmentalized training with Open-InCA** We illustrate the use of Open-InCA for efficient and parallelized training with SAFE. **(A)** In Open-InCA, each learned binary classifier corresponds to its own dedicated query (yellow) and class-head (orange), in blue is the shared frozen cross-attention module. **(B)** Given a data-sample (green), we may control which adapters can be updated with its gradient, and based on the architectural choice there will be no information leakage. **(C)** Using a batch of data-samples and a graph topology of data access rights, we can learn all adapters in parallel using loss masking which automatically blocks and routes gradients to their appropriate adapters.

adapter can be de-composed as

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

By using the sigmoid mapping we compute the loss for each Open-InCA binary-class adapter as

$$\begin{aligned} L_{i,j} = &- \hat{y}_j^{(i)} \cdot \log(\sigma(\mathrm{CA}_\theta(\mathbf{z}, [\mathbf{q}_i])) \\ &- (1 - \hat{y}_j^{(i)}) \cdot \log(1 - \sigma(\mathrm{CA}_\theta(\mathbf{z}, [\mathbf{q}_i])), \end{aligned}$$

where $\hat{y}_j^{(i)}$ is 1 if example j is a positive example for the class corresponding to adapter i, and 0 otherwise. For a batch of B samples and a set of M adapters, we obtain the loss matrix $L \in \mathbb{R}^{M \times B}$. Each loss entry $L_{i,j}$ is computed with frozen cross-attention parameters and the gradients are based on the learnable parameters of only the ith adapter (due to σ):

$$\frac{\partial L_{i,j}}{\partial q_k} = 0 \text{ for } k \neq i.$$

This implies any summation with $\{a_i\} \in \mathbb{R}^M$

$$\ell = \sum_{k=1}^{M} a_k L_{k,j}$$

has the property of reducing to a single term after differentiation

$$\frac{\partial L}{\partial q_i} = a_i \cdot \frac{\partial L_{i,j}}{\partial q_i}.$$

This is highly convenient in automatic differentiation packages, as it reduces our entire optimization (optimizing each adapter) into a single "backwards" call with the summed loss ℓ. In particular, for a given SG topology we can define A for a data batch with $A \in \{0, 1\}^{M \times B}$ such that

$$A_{ij} = \begin{cases} 1 & \text{if shard } i \text{ has access to data sample } j \\ 0 & \text{otherwise} \end{cases}$$

With this we can optimize the entire SAFE ensemble in parallel in a compartmentalized manner by directly optimizing the objective $\ell = \sum(L \odot A)$ where $\odot$ denotes the Hadamard product and $\sum$ is computed over all the matrix entries.

DP-SAFE. For the mixed-privacy DP-SAFE method we can follow a similar process in which we compute a private loss and a non-private loss using 2 separate masking rules defined via $A_{\text{DP}} \in \{0, 1\}^{M \times B}$ and $A_{\text{direct}} \in \{0, 1\}^{M \times B}$ accompanied by respective optimizers DP-SGD and AdamW.

B. Additional Ablations

Additional Sharding Scales. In Table 5, Table 6, and Table 7 we report the accuracies of SAFE, SISA, and ProtoSISA across a larger range of sharding scales, namely $2, 4, 8, 16, 32, 64, 128, 256$. We see that SAFE outperforms SISA on average across all sharding scales, with the gap peaking at 14.3% at 256 shards. Furthermore while the ProtoSISA method improves over SISA at the large sharding scales ≥ 64, it still underperforms SAFE by a margin of $(3.8\% - 10.1\%)$.

SAFE without prototypes. SAFE benefits from both synergy-aware sharding and the inductive bias of the prototype model. To separate out these two factors, we consider the SAFE method without prototypes, which we call "NoprotoSAFE". In Fig. 8 we compare the accuracy of SAFE, NoprotoSAFE, and SISA. We see that at the large sharding scales, the use of prototypes in SAFE gives a modest boost in test accuracy $0.5 - 2.5\%$ over NoprotoSAFE. We see that NoprotoSAFE still significantly outperforms SISA, with margins as high as $10 - 30\%$, particularly at the large sharding scales ≥ 64. We conclude that the synergy-aware sharding of the SAFE method is the primary factor for the boost in performance over SISA.

SAFE and SISA using a linear model. Another efficient approach to adapting large pretrained models is to perform head-only finetuning, where only a linear classifier head is trained. Thus a natural question is how do SAFE, SISA, and ProtoSISA perform when replacing the InCA adapters with linear models. In Fig. 7 we report the gain in test accuracy when using the InCA model relative to a linear model at different sharding scales for the various methods. We see that for the SAFE method the InCA model uniformly outperforms the linear model at all sharding scales. For SISA and ProtoSISA, at the larger sharding scales ≥ 64 the linear model starts to outperform in some cases. We suspect that SISA and ProtoSISA have difficulty training the cross-attention module when there are very few examples per shard, whereas the synergistic sharding of SAFE enables it to successfully train a more complex model.

C. Analysis of the Shard Graph

In Section 4, we theoretically analyze the expected forgetting costs of SAFE for different shard graph topologies. To understand the effect of different topologies, we consider a graph with a determined connectivity level, defined by assigning each node in V d outbound edges. When presenting SAFE, we concluded that when using a disjoint clique structure for the Shard Graph the expected cost to forget a sample x (i.e. the number of samples required for retraining) is $\mathbb{E}|M_x| = d \cdot |S|$. We note this is the optimal case and if the degree d connectivity is applied uniformly at random, $\mathbb{E}|M_x|$ is in fact an order of magnitude larger $\mathbb{E}|M_x| \sim d^2 \cdot |S|$. Below we present a formal statement and proof that $\mathbb{E}|M_x| \sim d^2 \cdot |S|$ for a graph with random connectivity.

Theorem 1. *Suppose we have a set of nodes* $V =$

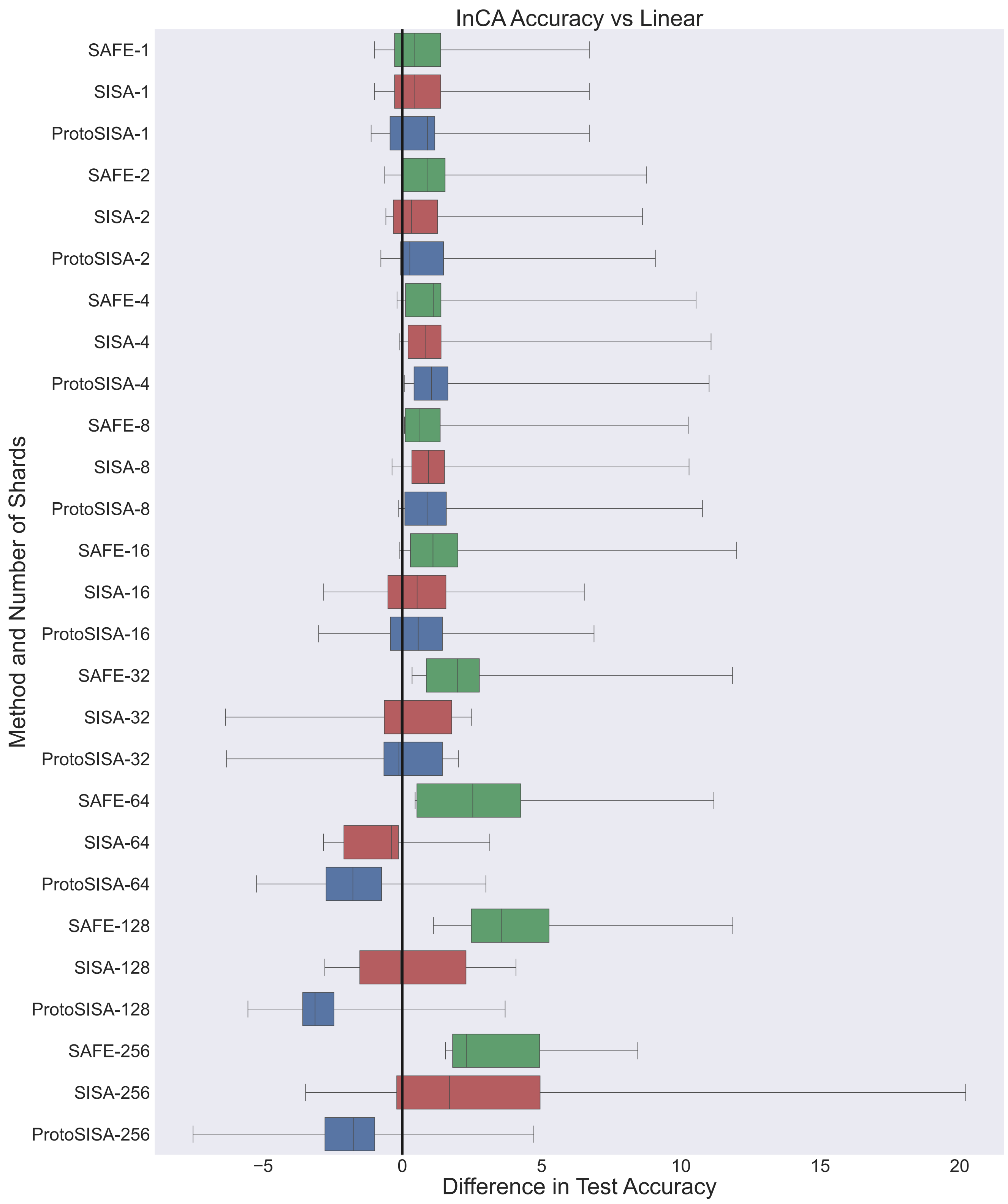

Figure 7. **InCA vs. Linear.** We report the difference in test accuracy between using the InCA model and using a linear model for the methods SAFE, SISA, and ProtoSISA at sharding scales $1, 2, 4, 8, 16, 32, 64, 128, 256$. Positive values correspond to InCA having higher accuracy, negative values correspond to the linear model having higher accuracy. Each box corresponds to 7 values, specifically the test accuracy over the 7 datasets we consider. The boxes corresponding to SAFE, SISA, and ProtoSISA are colored green, red, and blue respectively.

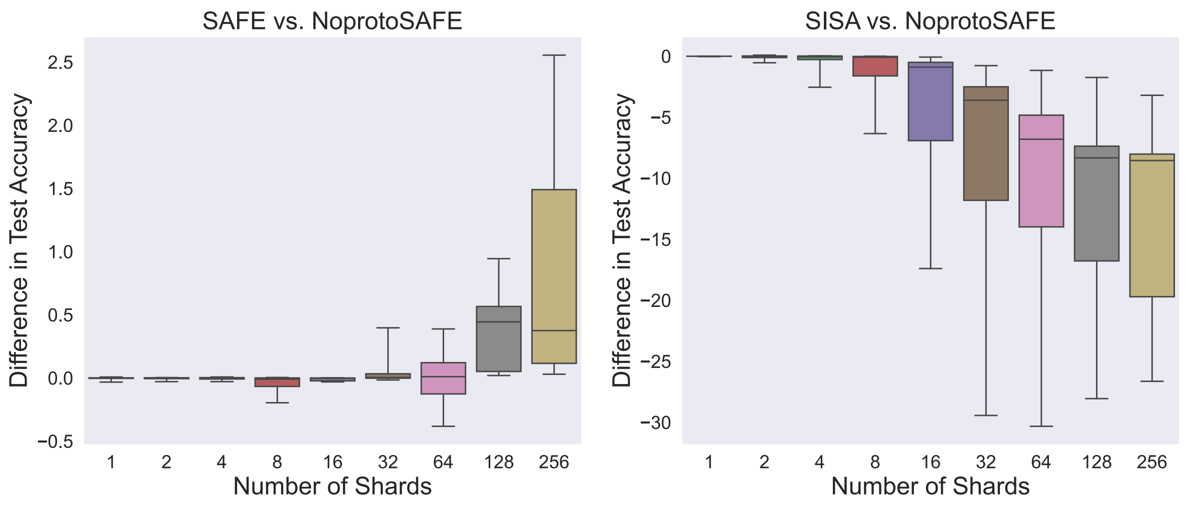

Figure 8. **SAFE without prototypes.** We provide comparisons against SAFE with and without applying prototypes. SAFE without applying prototypes is dubbed "NoprotoSAFE". Each box in the subplots corresponds to 7 values, one for each of the datasets. **(Left) SAFE vs. NoprotoSAFE.** We report the difference in test accuracy between SAFE and NoprotoSAFE. Positive values correspond to SAFE outperforming, negative values correspond to NoprotoSAFE outperforming. **(Right) SISA vs. NoprotoSAFE.** We report the difference in test accuracy between SISA and NoprotoSAFE. Positive values correspond to SISA outperforming, negative values correspond to NoprotoSAFE outperforming.

$\{S_1, \ldots, S_n\}$ *where each* S_i *is a source of data and* $S_i \cap S_j = \emptyset$ *for* $i \neq j$. *Furthermore assume that all the sources are of the same size, i.e.* $|S_1| = |S_2| = \cdots = |S_n|$. *Suppose each node is assigned* d *outbound edges independently and uniformly (in addition to the default self-connection). Furthermore assume that* $d^2 \leq C|V|^{1/2}$ *for some* $C > 0$. *Then the expected number of samples* $|M_x|$ *needed for retraining upon a forget request for a sample* x *is*

$$\mathbb{E}|M_x| = \Theta(|S_1|d^2).$$

Proof. Assume we receive a forget request for a sample $x \in S_i$. By uniformity without loss of generality we may assume that $i = 1$. We will let Z be a random variable denoting the number of samples we must retrain on whenever receiving a forget request for a sample $x \in S_1$ where the randomness is taken over the selection of the edges E of the graph. For a node n let $N_{in}(n) = \{v : (v, n) \in E\}$ denote the inbound neighborhood of n and let $N_{out}(n) = \{v : (n, v) \in E\}$ denote the outbound neighborhood of n. Then we have that

$$Z = \left| \bigcup_{n \in N_{in}(S_1)} \bigcup_{v \in N_{out}(n)} v \setminus \{x\} \right|.$$

We note then that

$$\begin{aligned} Z &\leq \sum_{n \in N_{in}(S_1)} \sum_{v \in N_{out}(n)} |S_1| \\ &\leq |N_{in}(S_1)|(d+1)|S_1|. \end{aligned}$$

Therefore

$$\mathbb{E}[Z] \leq |S_1|(d+1)\mathbb{E}[|N_{in}(S_1)|]$$

We note that for each node $v \in V \setminus S_1$ that $\mathbb{I}[v \in N_{in}(S_1)]$ is a Bernoulli random variable taking the value 1 with probability $\frac{d}{|V|-1}$. Furthermore by default we know that $S_1 \in N_{in}(S_1)$. Therefore

$$|N_{in}(S_1)| - 1 = \sum_{v \in V \setminus S_1} \mathbb{I}[v \in N_{in}(S_1)]$$

is a sum of $|V| - 1$ independent Bernoulli random variables. Thus $|N_{in}(S_1)| - 1$ obeys the Binomial distribution $Bin(|V| - 1, \frac{d}{|V|-1})$ which has mean d. Thus

$$\mathbb{E}[|N_{in}(S_1)|] = d + 1$$

and we conclude that

$$\mathbb{E}[Z] \leq |S_1|(d+1)^2 = O(|S_1|d^2).$$

Now we will prove the lower bound. We note for $d = 1$ the statement is trivial so we might as well assume $d \geq 2$. Since we are seeking a lower bound we may assume that after the forget request is received the entire source S_1 is dropped, as this only decreases the number of samples needed to retrain. In this case

$$Z = \left| \bigcup_{n \in N_{in}(S_1) \setminus S_1} \bigcup_{v \in N_{out}(n) \setminus S_1} v \right|.$$

We will focus on estimating the conditional expectation

$$\mathbb{E}[Z| \ |N_{in}(S_1) \setminus S_1| = k].$$

We note that by uniformity that the distribution of Z depends only on the size of $N_{in}(S_1) \setminus S_1$ and not the specific collection of nodes in $N_{in}(S_1) \setminus S_1$. Thus we will fix a choice $\{n_1, \dots, n_k\}$ of k nodes for $N_{in}(S_1) \setminus S_1$. Let E denote the event that $N_{in}(S_1) \setminus S_1 = \{n_1, \dots, n_k\}$ and let $\mathbb{P}_E(\bullet) := \mathbb{P}(\bullet|E)$ denote the probability of an event conditioned on E. We note then that

$$\mathbb{E}[Z| \ |N_{in}(S_1) \setminus S_1| = k] = \mathbb{E}[Z|E].$$

Let $A_i = N_{out}(n_i) \setminus S_1$ for $i = 1, \dots, k$. We note that if $A_1, A_2, \dots, A_k$ are disjoint then $Z = kd \cdot |S_1|$. Thus we have that

$$\begin{aligned}&\mathbb{E}\big[Z \big| \ E\big] \\ &\qquad \geq kd \cdot |S_1| \cdot \mathbb{P}_E(A_i \cap A_j = \emptyset \ \ \forall 1 \leq i \neq j \leq k).\end{aligned}$$

Thus we proceed to lower bound

$$\mathbb{P}_E(A_i \cap A_j = \emptyset \ \ \forall 1 \leq i \neq j \leq k).$$

The number of disjoint choices of $A_1, A_2, \dots, A_k$ is

$$\prod_{j=0}^{k-1} \binom{|V| - (k+1) - j(d-1)}{d-1}.$$

The number of total choices of $A_1, A_2, \dots, A_k$ is

$$\binom{|V|-2}{d-1}^k.$$

Thus we have

$$\begin{aligned}&\mathbb{P}_E(A_i \cap A_j = \emptyset \ \ \forall 1 \leq i \neq j \leq k) \\ &\geq \left[\frac{\binom{|V|-(k+1)-(k-1)(d-1)}{d-1}}{\binom{|V|-2}{d-1}}\right]^k.\end{aligned}$$

Now assume that $k \in [(1-t)d, (1+t)d]$ for some fixed $t \in (0,1)$. Note then that

$$\begin{aligned}\left[\frac{\binom{|V|-(k+1)-(k-1)(d-1)}{d-1}}{\binom{|V|-2}{d-1}}\right]^k &\geq \left[\frac{(|V|-kd)^{d-1}}{(|V|-2)^{d-1}}\right]^k \\ &\geq \left[\frac{(|V|-kd)^{d-1}}{|V|^{d-1}}\right]^k \\ &\geq \left[\frac{|V|-kd}{|V|}\right]^{kd} \\ &= \left[1 - \frac{kd}{|V|}\right]^{kd}.\end{aligned}$$

Now using the fact that $d^2 \leq C|V|^{1/2}$ and $k \leq (1+t)d \leq 2d$, we have that

$$\begin{aligned}\left[1 - \frac{kd}{|V|}\right]^{kd} &\geq \left[1 - \frac{2d^2}{|V|}\right]^{2d^2} \\ &\geq \left[1 - \frac{2C}{|V|^{1/2}}\right]^{2C|V|^{1/2}} \\ &= \left[1 - \frac{4C^2}{2C|V|^{1/2}}\right]^{2C|V|^{1/2}} \\ &= e^{-4C^2} + o(1) = \Omega(1).\end{aligned}$$

Thus it follows for $k \in [(1-t)d, (1+t)d]$ we have that

$$\begin{aligned}&\mathbb{E}[Z| \ |N_{in}(S_1) \setminus S_1| = k] \\ &\geq |S_1|d \cdot k \cdot \mathbb{P}_E(A_i \cap A_j = \emptyset \ \ \forall 1 \leq i \neq j \leq k) \\ &= \Omega((1-t)|S_1|d^2).\end{aligned}$$

We note it then suffices to show that for some fixed $t \in (0,1)$

$$\mathbb{P}\left(|N_{in}(S_1) \setminus S_1| \in [(1-t)d, (1+t)d]\right) = \Omega(1).$$

We recall that for each node $v \in V \setminus S_1$ that $\mathbb{I}[v \in N_{in}(S_1)]$ is a Bernoulli random variable taking the value 1 with probability $\frac{d}{|V|-1}$. Thus

$$|N_{in}(S_1) \setminus S_1| = \sum_{v \in V \setminus S_1} \mathbb{I}[v \in N_{in}(S_1)]$$

is a sum of $|V| - 1$ independent Bernoulli random variables. Thus $|N_{in}(S_1) \setminus S_1|$ obeys the Binomial distribution $Bin(|V|-1, \frac{d}{|V|-1})$ which has mean d and variance $d(1 - \frac{d}{|V|-1})$. Now let $W = |N_{in}(S_1) \setminus S_1|$. Then by Chebyshev's inequality for $t > 0$

$$\mathbb{P}\left(|W - d| \geq td\right) \leq \frac{(1 - \frac{d}{|V|-1})}{dt^2} \leq \frac{1}{dt^2}.$$

We note since $d \geq 2$ we can choose $t = \frac{\sqrt{2}}{\sqrt{3}}$ so that $\frac{1}{dt^2} \leq \frac{3}{4}$. Thus with probability at least $1/4$ we have that

$$|N_{in}(S_1) \setminus S_1| \in [(1-t)d, (1+t)d].$$

It follows that $\mathbb{E}[Z] = \Omega(|S_1|d^2)$. $\square$

D. Stochastic forgetting

Below we provide a general definition for probabilistic unlearning:

Definition 1. $(\alpha, \beta)-$*sharded-unlearning: Consider an algorithm $\mathcal{A}$ that, given a shard graph G as input, outputs a model $\mathcal{A}(G)$ trained on the shards of G. If G' is a shard*

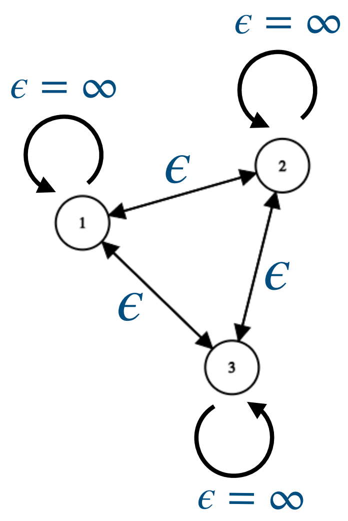

Figure 9. **Shard-Graph in SAFE-DP**: In SAFE-DP each shard when training adapters uses its own data non-privately, but enforces DP when using data from other shards. This can visualized as shard graphs with weighted edges, where the edge weight measures the privacy leakage between the interacting nodes.

graph that can be obtained by removing data or nodes from G, we write $G' \preccurlyeq G$. We say that U is an (α, β)-unlearning algorithm for $\mathcal{A}$ if for all $G' \preccurlyeq G$ and events E

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

Note that the definition bares resemblance to the definition of differential privacy, which says that:

Definition 2. *[16] $(\epsilon, \delta)-$differential privacy: An algorithm $\mathcal{A}$, is said to be $(\epsilon, \delta)-$DP for $\epsilon > 0$ and $\delta \ll 1$, if for all adjacent datasets D, D' (such that D and D' differ in at most one sample), and all possible events Θ, the following relation holds:*

$$\mathbb{P}(\mathcal{A}(D) \in \Theta) \leq e^{\epsilon}\mathbb{P}(\mathcal{A}(D') \in \Theta) + \delta.$$

Training a model with a DP algorithm (for instance DP-SGD [1]), enables us to perform free (α, β)-unlearning for 1-forget request, by choosing $\alpha = \epsilon$ and $\beta = \delta$. "Free" here means that the model need not be changed after a forgetting request as it still satisfies the $(\alpha, \beta)-$unlearning guarantee. However, as we get sequential forgetting requests (≥ 1), the privacy bound weakens resulting in more leakage. To understand the effectiveness of differential privacy in forgetting, we need to capture its exact behaviour (ϵ, δ) when provided with k sequential forgetting requests. More precisely we get the following result for group differential privacy:

Theorem 2. *[16] Group privacy: Let $\mathcal{A}$, be an $(\epsilon, \delta)-$DP algorithm. Then for all D, D', such that D and D' differ in at most k samples, we get the following result:*

$$\mathbb{P}(\mathcal{A}(D) \in \Theta) \leq e^{\epsilon_g}\mathbb{P}(\mathcal{A}(D') \in \Theta) + \delta_g,$$

where $\epsilon_g = k\epsilon$ and $\delta_g = \dfrac{e^{k\epsilon} - 1}{e^{\epsilon} - 1}\delta$

We observe that the group privacy result, i.e. $(\epsilon_g, \delta_g) = \Big(e^{k\epsilon}, \dfrac{e^{k\epsilon} - 1}{e^{\epsilon} - 1}\delta\Big)$, weakens with increasing k. This necessitates the notion of a privacy budget (α_b, δ_b) which is an upper bound for the privacy leakage accumulation with successive forgetting requests. This budget can be chosen by the user, or in some cases, constrained by the fact that $\dfrac{e^{k\epsilon} - 1}{e^{\epsilon} - 1}\delta \leq 1$.

Differential privacy can detrimentally reduce the utility (accuracy) of a model. Hence, blindly using DP-SGD (or any another DP method) during training may result in models that, while strongly private (and hence not requiring frequent re-training) attain significantly lower accuracy. SAFE provides a simple yet effective unlearning mechanism which enables forgetting shards, by re-training the neighbouring contaminated shards (requires frequent yet reduced re-training, but higher accuracy). These observations inspire us to design an algorithm which combines SAFE and Stochastic Forgetting:

SAFE-DP: For each shard S_i in the shard graph G, we train a binary classification model which treats the data at node S_i as the positive class (without privacy), and data at neighbouring nodes pointing to it as the negative classes (with privacy). When asked to forget a sample in S_i we can simply drop the classifier for the shard S_i (later re-train if mandated by the privacy cost of the budget), without having to worry about re-training models corresponding to other shards which used S_i as a negative class (since the data of the negative shards was trained with DP).

Unlike the standard version of SAFE, where all the connections of shard S_i (using S_i for training) need to re-trained upon a single forget request, SAFE-DP allows us to keep the remaining neighboring shards while accounting for some cost to the privacy budget. By allowing each shard S_i to use its own data non-privately, SAFE-DP also provides better utility compared to completely private models (trained with DP). While training with SAFE-DP, we compute the total (α, β) using the composition property of DP [16, 37, 26].

Privacy Accounting: The SAFE-DP algorithm is defined with an accompanying privacy level given by (ϵ, δ), which sets the DP training parameters in SAFE-DP (using DP-SGD [1, 24]). While the (α, β) parameters in the unlearning definition (Definition 1) are dependent on the number of sequential forgetting requests k, SAFE-DP provides the following unlearning guarantees: $\alpha = k\epsilon$ and $\beta = \frac{e^{k\epsilon}-1}{e^{\epsilon}-1}\delta$, where $\beta \leq 1$ (from Theorem 2).

Before the start of training, the user chooses a desired forgetting budget (α_b, β_b) (with $\beta_b \leq 1$), which measures the information contained (privacy leakage) about the user after forgetting her samples. The forgetting algorithm should respect the budget while providing non-vacuous guarantees. This provides us with the following bound for the number of unlearning request before full re-training: When $\delta \ll 1$ and $\delta \ll \beta_b$, we can approximate this result with

$$k = \min\left(\frac{\alpha_b}{\epsilon}, \frac{\log\left(\beta_b(e^{\epsilon}-1)/\delta + 1\right)}{\epsilon}\right).$$

For our experiments we assume $\alpha_b = 30$ and $\beta_b = 1$. We vary $\epsilon = \{1, 2, 3, 4\}$ and $\delta = 1\mathrm{e}{-10}, 1\mathrm{e}{-11}, 1\mathrm{e}{-12}, 1\mathrm{e}{-13}$ and choose different values for k. Note that we can choose the values for k by varying both ϵ and δ (ensuring $\delta \ll 1$) and choosing the best value by comparing the accuracy on a held out validation set.

E. Additional Details of Architecture and Training

Optimized graph structures. For our SAFE method, in Section 5 we described the bi-level sharding structure where we split the dataset into a number n_c “coarse shards” via class balanced sub-sampling which are further split into n_f “fine shards” via class partitioning. The total number of shards is given by $n = n_c \cdot n_f$. In Table 4 we report the choice of (n_c, n_f) used for each dataset for each value of n for the SAFE method.

Optimization. Whenever training InCA adapters, we train using AdamW for 30 epochs using cosine annealing and starting learning rate $\mathrm{lr} = 0.05$; we use weight decay of 10^{-4}. Whenever training the linear model for the experiment in Fig. 7 we lower the learning rate to $\mathrm{lr} = 3\mathrm{e}{-4}$ as we observed this increased performance. For the linear model all other hyperparameters remain the same as when training the InCA adapter.

Architecture. When using InCA adapters, LayerNorm is applied to both the inputs and queries separately before they are passed through the cross-attention block. A second LayerNorm is applied after the cross-attention before the final logits are computed as customary in ViTs. While in general InCA adapters can be applied to any layer in the network [15], in our experiments we always attach it to the penultimate layer, namely the end of block 22 (the input to `blocks.23.norm1`). When using linear adapters for the experiment depicted in Fig. 7, we also apply LayerNorm before the fully connected layer.

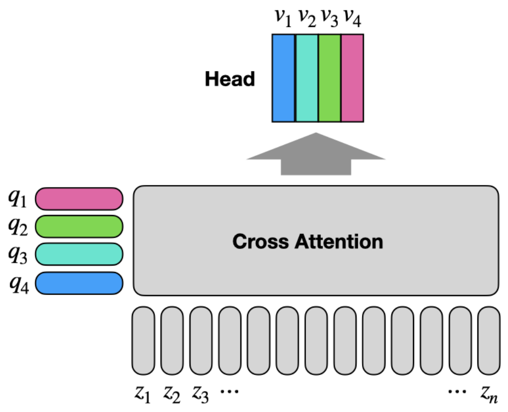

Figure 10. **The InCA Architecture.** Each class i has a query q_i and a head vector v_i. The logit for class i given by the inner product $v_i \cdot \text{cross-attention}(\mathbf{z}, q_i)$ where $\mathbf{z} = (z_1, \ldots, z_n) = f_w(x)$ is the embedding extracted from a frozen pre-trained model.

Prototype model. We note that the prototype model is only added at inference time, not training time, as adding the prototype model during training would expose each adapter to information from all other training samples, thus breaking the information compartmentalization. When computing the prototypes p_k, we normalize the feature embeddings

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

which makes the prototypes less sensitive to outliers and leads to better predictions.

Dataset details. In Table 3 we report the size of the training and testing splits and number of classes for the 7 datasets we consider, and links to access the data.

F. Additional Related Work

In our work we present SAFE as a method that flexibly learns a model composed of model parts that are learned using different data (as prescribed by the Shard Graph). This heterogeneous routing of data during training enables training the different model components in parallel and permits training hundreds of models quickly and efficiently. While we apply this approach for the problem of forgetting, recently “heterogeneous data routing” has also been the focus of much work in enabling better massive model scaling. In these works, “heterogeneous data routing” is used to route different data and activations to different model parts and distributing the computation to more computing nodes. Through distribution of computation, one can reduce the inference and training costs of foundation models and enable even more parameters than what is permissible by a single monolithic model [17]. In the work of [17] a large language model based on the transformer architecture is built with

Dataset	Training Images	Testing Images	# Classes	URL
Caltech-256 [27]	15,418	15,189	257	https://authors.library.caltech.edu/7694/
CIFAR-100 [33]	50,000	10,000	100	https://www.cs.toronto.edu/~kriz/cifar.html
CUB-200 [47]	5,994	5,794	200	https://www.vision.caltech.edu/datasets/cub_200_2011/
DTD [13]	4,230	1,410	47	https://www.robots.ox.ac.uk/~vgg/data/dtd/
MIT-67 [41]	5,360	1,340	67	https://web.mit.edu/torralba/www/indoor.html
Stanford Cars [32]	8,144	8,041	196	https://ai.stanford.edu/~jkrause/cars/car_dataset.html
Stanford Dogs [30]	12,000	8,580	120	http://vision.stanford.edu/aditya86/ImageNetDogs/

Table 3. **Dataset Information.** We report the number of classes as well as the number of training and testing images for each dataset, as well as links to download the datasets.

dynamic execution layers, where the model's intermediate activations are routed into disjoint layers based on their representations via "switching layers". This is generalized in the work of Pathways [3] that creates the necessary infrastructure to train such models on distributed computing systems and allows training state-of-the-art language models [12]. The latest developments [18, 19] of this method apply heterogeneous propagation of data in connection with multi-task learning where "agent networks" cooperate with partner-agent representations to adapt and solve new tasks. We note while both [17, 3] and our work utilize heterogeneous data routing, in our work data routing and compartmentalization is deterministic and is based on pre-specified data usage rules codified by the Shard Graph, as opposed to selecting the data routing based on the data's representations. Overall our work focuses on the problem of forgetting rather than multi-task learning and increasing the model scale.

Dataset\ Num. Shards	2	4	8	16	32	64	128	256
Caltech-256	(2, 1)	(4, 1)	(4, 2)	(4, 4)	(4, 8)	(4, 16)	(4, 32)	(8, 32)
CIFAR-100	(2, 1)	(4, 1)	(8, 1)	(8, 2)	(8, 4)	(16, 4)	(16, 8)	(16, 16)
CUB-200	(2, 1)	(4, 1)	(4, 2)	(8, 2)	(8, 4)	(8, 8)	(4, 32)	(8, 32)
DTD	(2, 1)	(4, 1)	(4, 2)	(8, 2)	(8, 4)	(8, 8)	(16, 8)	(16, 16)
MIT-67	(2, 1)	(2, 2)	(2, 4)	(8, 2)	(8, 4)	(8, 8)	(8, 16)	(16, 16)
Stanf. Cars	(2, 1)	(2, 2)	(2, 4)	(4, 4)	(4, 8)	(8, 8)	(8, 16)	(8, 32)
Stanf. Dogs	(2, 1)	(4, 1)	(8, 1)	(8, 2)	(16, 2)	(16, 4)	(16, 8)	(16, 16)

Table 4. **Coarse vs. fine shard split.** We report the number of coarse and fine shards, (n_c, n_f), used for each dataset at the different sharding levels.

Dataset	No sharding	Prototypes	2	4	8	16	32	64	128	256
Caltech-256	94.3%	93.2%	94.2%	94.1%	94.0%	93.7%	93.7%	93.5%	93.2%	93.3%
CIFAR-100	83.1%	71.2%	84.4%	84.6%	84.1%	83.3%	82.8%	82.2%	81.5%	80.9%
CUB-200	88.3%	85.8%	88.6%	87.9%	86.1%	85.8%	85.3%	83.5%	82.5%	84.5%
DTD	77.8%	73.8%	78.3%	78.3%	77.1%	75.4%	75.5%	75.1%	73.9%	74.2%
MIT-67	87.9%	85.8%	88.1%	87.7%	86.9%	86.3%	86.5%	86.0%	86.2%	86.4%
Stanf. Cars	75.7%	41.0%	72.6%	68.5%	62.1%	58.3%	53.2%	47.4%	41.9%	36.2%
Stanf. Dogs	87.9%	88.0%	88.9%	89.2%	89.2%	89.0%	88.6%	88.3%	87.7%	87.8%
Avg.	85.0%	77.0%	85.0%	84.3%	82.8%	81.7%	80.8%	79.4%	78.1%	77.6%

Table 5. **SAFE Accuracy at different sharding scales.** We report the accuracy of SAFE across different sharding scales.

Dataset	No sharding	Prototypes	2	4	8	16	32	64	128	256
Caltech-256	94.3%	93.2%	94.2%	94.1%	93.6%	92.8%	90.1%	86.7%	84.9%	84.6%
CIFAR-100	83.1%	71.2%	84.5%	84.6%	84.0%	83.2%	82.0%	80.7%	79.7%	77.6%
CUB-200	88.3%	85.8%	88.5%	87.6%	83.9%	73.1%	65.3%	65.1%	62.8%	63.7%
DTD	77.8%	73.8%	78.3%	78.1%	76.1%	74.6%	71.8%	65.1%	58.6%	52.6%
MIT-67	87.9%	85.8%	88.1%	87.7%	86.8%	85.2%	83.6%	81.5%	77.5%	77.9%
Stanf. Cars	75.7%	41.0%	72.1%	66.0%	55.8%	40.9%	23.7%	17.1%	13.2%	7.0%
Stanf. Dogs	87.9%	88.0%	88.8%	89.2%	89.2%	88.6%	86.5%	83.1%	81.1%	79.8%
Avg.	85.0%	77.0%	84.9%	83.9%	81.4%	76.9%	71.9%	68.5%	65.4%	63.3%

Table 6. **SISA Accuracy at different sharding scales.** We report the accuracy of SISA across different sharding scales.

Dataset	No sharding	Prototypes	2	4	8	16	32	64	128	256
Caltech-256	94.3%	93.2%	94.2%	94.1%	93.6%	92.8%	90.1%	86.8%	86.4%	90.5%
CIFAR-100	83.1%	71.2%	84.5%	84.6%	84.0%	83.2%	82.0%	80.7%	79.7%	77.7%
CUB-200	88.3%	85.8%	88.5%	87.6%	83.9%	73.1%	65.7%	67.8%	73.5%	83.5%
DTD	77.8%	73.8%	78.3%	78.1%	76.1%	74.6%	71.8%	66.7%	67.5%	71.2%
MIT-67	87.9%	85.8%	88.1%	87.7%	86.9%	85.2%	83.9%	81.7%	81.2%	84.4%
Stanf. Cars	75.7%	41.0%	72.1%	66.0%	55.8%	40.9%	23.9%	17.8%	16.9%	25.5%
Stanf. Dogs	87.9%	88.0%	88.8%	89.2%	89.2%	88.6%	86.6%	83.2%	81.9%	83.5%
Avg.	85.0%	77.0%	84.9%	83.9%	81.4%	76.9%	72.0%	69.3%	69.6%	73.8%

Table 7. **ProtoSISA Accuracy at different sharding scales.** We report the accuracy of ProtoSISA across different sharding scales.