LEVERAGING DISTRIBUTION MATCHING TO MAKE APPROXIMATE MACHINE UNLEARNING FASTER

Anonymous authors
Paper under double-blind review

ABSTRACT

Approximate machine unlearning (AMU) enables models to 'forget' specific training data through specialized fine-tuning on a retained (and forget) subset of training set. However, processing this large retained subset still dominates computational runtime, while reductions of unlearning epochs also remain a challenge. In this paper, we propose two complementary methods to accelerate arbitrary classification-oriented AMU method. First, **Blend**, a novel distribution-matching dataset condensation (DC), merges visually similar images with shared blend-weights to significantly reduce the retained set size. It operates with minimal pre-processing overhead and is orders of magnitude faster than state-of-the-art DC methods. Second, our loss-centric method, **Accelerated-AMU (A-AMU)**, augments the AMU objective to quicken convergence. A-AMU achieves this by combining a steepened primary loss to expedite forgetting with a differentiable regularizer that matches the loss distributions of forgotten and in-distribution unseen data. Our extensive experiments demonstrate that this dual approach of data and loss-centric optimization dramatically reduces end-to-end unlearning latency across both single and multi-round scenarios, all while preserving model utility and privacy. To our knowledge, this is the first work to systematically tackle unlearning efficiency by jointly designing a specialized dataset condensation technique with a dedicated accelerated loss function. Code is available at `https://github.com/algebraicdianuj/DC_Unlearning`.

1 INTRODUCTION

Machine unlearning (MU) seeks to expunge specific training points from a deployed model. ***Exact MU*** Cao & Yang (2015); Bourtoule et al. (2021) offers perfect deletion guarantees by fully retraining, often via data-sharding schemes, but remains prohibitively slow for modern networks. ***Approximate MU*** (AMU) Jia et al. (2023); Kurmanji et al. (2023); Chundawat et al. (2023); Golatkar et al. (2020); Warnecke et al. (2021); Guo et al. (2019) replaces such guarantees with speed, by achieving following goal in one-step or iterative settings:

$$\min_{\text{model parameters}} \alpha \underbrace{\textbf{Main Loss}}_{\text{on retained data}} + \beta \underbrace{\textbf{Forgetting Reg.}}_{\text{drives deletion}} \qquad (*)$$

Here, $\alpha, \beta \geq 0$ control the utility-vs-forgetting trade-off. We focus on the common iterative case $\alpha = 1$, as one step variants assume unrealistically simple loss landscapes. Although approximate machine unlearning (AMU) avoids full retraining, repeatedly scanning the large retained set still dominates runtime. At the same time, reducing unlearning epochs remains a challenge that parallels slow training convergence, which BatchNorm successfully addressed Ioffe & Szegedy (2015).

We address this through specialized speed-focused dataset condensation Zhao et al. (2020). Building on distribution-matching condensation Zhao & Bilen (2023), we propose ***Blend***, which learns compact *blend-weights* to synthesize synthetic datasets preserving training utility. This shrinks the retained set and accelerates unlearning without sacrificing accuracy or privacy. We further introduce Accelerated-AMU (***A-AMU***), which preserves template equation * but steepens the **main loss** per sample/batch and regularizes with distribution-matching MIA minimization for rapid convergence. Our approach demonstrates substantial speedups and strong unlearning versus baseline SOTA AMU methods. We summarize our contributions as:

- ***Blend*** DC, specifically designed for unlearning, combines image blending with distribution matching, achieving an average 44% retained dataset reduction with approximately 7.5% preprocessing overhead compared to baseline unlearning, and is approximately 1500 times faster than state-of-the-art dataset condensation methods.

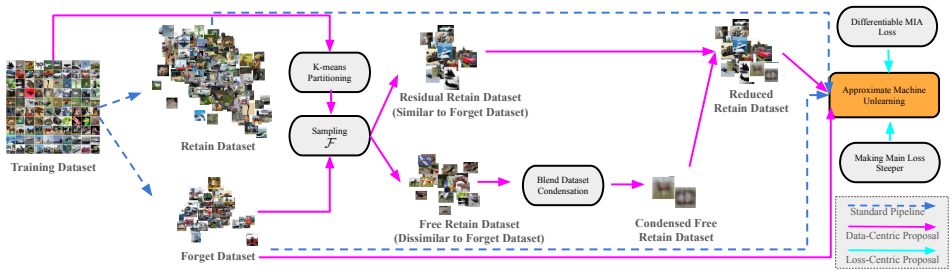

Figure 1: The overall pipeline of our proposal to make accelerate AMU, which comprise data-centric and loss-centric components

- ***A-AMU*** augments the AMU loss formulation with a steepened primary loss and a differentiable membership-inference regularizer, reducing unlearning runtime by 76.82% in single-round and 51.83% in multi-round scenarios compared to baseline methods, by minimizing the number of epochs, while maintaining strong privacy-utility statistics.
- Our benchmarks show 84.61% and 54.45% faster end-to-end unlearning in single and multi-round scenarios, respectively, on various image classification datasets and models, effectively filling latency gaps compared to current state-of-the-art AMU methods.

2 Related Works

Machine unlearning (MU) has evolved along two complementary axes. ***Exact MU*** guarantees equivalence to full retraining by either retraining on data shards (SISA Bourtoule et al. (2021), ARCANE Yan et al. (2022)) or deriving closed-form updates for simple models such as k-means and linear classifiers Ginart et al. (2019); Mahadevan & Mathioudakis (2021), but scaling these ideas to over-parameterized deep networks remains hard. ***Approximate MU*** relaxes this guarantee for speed: influence-function or Hessian sketches explicitly subtract a point's contribution Graves et al. (2021); Wu et al. (2020); Golatkar et al. (2020); Warnecke et al. (2021);"bad-teacher" distillation removes the target data then distills utility back from a clean model Kurmanji et al. (2023); Chundawat et al. (2023); and recent accelerations prune parameters Tanaka et al. (2020); Jia et al. (2023), nudge decision boundaries Chen et al. (2023), or couple sparse adapters with shard partitioning, culminating in class-agnostic, data-free DELETE Zhou et al. (2025) and Unlearn-and-Distill Lee et al. (2025). Parallel to MU, ***dataset condensation (DC)*** compresses training sets into compact proxies: from gradient matching Wang et al. (2018); Zhao et al. (2020) and distribution/trajectory matching Zhao & Bilen (2023); Cazenavette et al. (2022) to latent re-parameterization, differentiable augmentation and infinite-width kernels Kim et al. (2022); Zhao & Bilen (2021); Nguyen et al. (2021), with EDF recently highlighting class-discriminative cues via Grad-CAM Wang et al. (2025). However ***DC for MU*** is nascent: QuickDrop naively integrates gradient-matching DC into federated unlearning Dhasade et al. (2024), and TCGU shows graph-level gains by naively condensing once then editing the proxy Li et al. (2024); nevertheless, existing DC objectives are still too costly to serve as a drop-in unlearning accelerator, motivating our targeted and simpler distribution-matching approach.

3 Methodology

3.1 Preliminaries

Let the training dataset $\mathbf{D}$, consist of labeled images, denoted as $\{I_p\}_p$, comprising of C classes and N_c images per class, where $c \in \{1, 2, \ldots, C\}$. The training dataset is divided into retain $\mathbf{R}$ and forget $\mathbf{F}$ dataset. $\mathbf{D} = \{I_p\}_p$ was used to train a parameterized target network, and it achieved local minima parameters θ on average loss $\mathcal{L}(\theta; x, y)$ over $(x, y) \in \mathbf{R} \cup \mathbf{F}$. Let $\mathbf{D}$ and $\tilde{\mathbf{T}}$ be datasets drawn from the same underlying distribution $\mathcal{D}$, where $\tilde{\mathbf{T}}$ is the testing dataset, and is used in evaluation of training. Our objective is to unlearn forget set $\mathbf{F} \subset \{I_p\}_p$ from the pretrained model. Denote by $\mathcal{D}_R$ and $\mathcal{D}_F$ the empirical distribution of $\mathbf{R}$ and $\mathbf{F}$ respectively. The subset $\mathbf{T} \subseteq \tilde{\mathbf{T}}$ consists of elements whose class labels match those in $\mathbf{F}$, and $\mathcal{D}_T$ is its empirical distribution.

3.2 Blend Dataset Condensation

Using **Algorithm 1**, we divide images of each class of $\mathbf{D}$ into k sub-classes using k-means clustering over the feature representations, extracted by a randomly initialized lightweight feature extractor ψ_ϑ (much smaller than the pretrained model to be unlearned). The resulting partition of the dataset can

Algorithm 1 k-means Partitioning ($\mathbf{D} = \{I_p\}_p$, ψ_ϑ)

Indices: $c = 1, \ldots, C$ (class); $\kappa = 1, \ldots, k$ (k-means cluster); $b = 1, \ldots, n_\kappa$ (images in cluster κ)

$\mathcal{C} \leftarrow \varnothing$ ▷ Initialize image clusters
for each class $c = 1, \ldots, C$ **do**
  $\{I_a\}_{a=1}^{N_c} \leftarrow \{I_p \mid y_p = c\}$ ▷ Collect class images
  $\{\psi_\vartheta(I_a)\}_{a=1}^{N_c}$ ▷ Compute feature vectors
  $\{\psi_\vartheta(I_a)\}_{a=1}^{N_c} \xrightarrow{k\text{-means}} \left\{\{\psi_\vartheta(I_{b\kappa})\}_{b=1}^{n_\kappa}\right\}_{\kappa=1}^{k}$ ▷ Run k-means
  $\left\{\{\psi_\vartheta(I_{b\kappa})\}_{b=1}^{n_\kappa}\right\}_{\kappa=1}^{k} \rightarrow \left\{\{I_{b\kappa}\}_{b=1}^{n_\kappa}\right\}_{\kappa=1}^{k}$ ▷ Map to images
  $\mathcal{C} \leftarrow \mathcal{C} \cup \left\{\{I_{b\kappa}\}_{b=1}^{n_\kappa}\right\}_{\kappa=1}^{k}$ ▷ Add to global set
end for
$\mathcal{C} = \left\{\left\{\{I_{b\kappa c}\}_{b=1}^{n_{\kappa c}}\right\}_{\kappa=1}^{k}\right\}_{c=1}^{C}$ ▷ Full cluster set
Refine notation: $b \mapsto i$, $(\kappa, c) \mapsto j$
return $\{\{I_{ij}\}_{i=1}^{n_j}\}_{j=1}^{k\times C}$ ▷ Partitioned $\mathbf{D}$ dataset

be written as:

$$\{I_p\}_p \xrightarrow{\text{k-means partitioning}} \left\{\{I_{ij}\}_{j=1}^{n_i}\right\}_{i=1}^{k\times C} \tag{1}$$

where i indexes the sub-class (with a total of $k \times C$ sub-classes), and j indexes the images within each sub-class. Each i-th subclass contain n_i images. For each sub-class i, we define a *blended image* $\mathcal{I}(\omega_i)$ as a weighted average of the images $\{I_{ij}\}_{j=1}^{n_i}$ b elonging to that sub-class:

$$\mathcal{I}(\omega_i) \overset{\text{blend}}{=} \frac{1}{\sum_{j=1}^{n_i} \omega_{ij}} \sum_{j=1}^{n_i} \omega_{ij}\, I_{ij} \tag{2}$$

where ω_{ij} are scalar weights associated with the j-th image in sub-class i. These weights are treated as learnable parameters and are optimized to ensure that the feature representation of the blended image $\mathcal{I}_i$ closely matches the mean feature representation of its constituent images to form a condensed single image proxy of $\{I_{ij}\}_{j=1}^{n_i}$. We estimate ω_i in equation 2, via following distribution matching Zhao & Bilen (2023) loss:

$$\min_{\omega_i} \quad \mathbb{E}_{\vartheta\sim P_\vartheta} \left\| \frac{1}{n_i} \sum_{j=1}^{n_i} \psi_\vartheta(I_{ij}) \quad - \quad \psi_\vartheta\big(\mathcal{I}(\omega_i)\big) \right\|_2^2$$

where ψ_ϑ is the feature extractor parameterized by ϑ, and $\|\cdot\|$ denotes the Euclidean norm. The expectation is taken over a distribution P_ϑ of lightweight feature extractors to promote robustness, as adapted in Zhao & Bilen (2023).

3.3 Creating the *Reduced* Retain Dataset

Dataset condensation replaces each original feature $x \in \mathbb{R}^n$ (as Lipchitz mapping of original image) with a synthetic prototype whose placement preserves the topology of the manifold on which the features live. Because this process is global, a prototype can still migrate into the neighbourhood of the to-be forgotten feature samples, thus creating an obstacle for unlearning. Furthermore, since the condensation time scales with the size of the retain set, condensing the entire retain dataset becomes a significant computational bottleneck. We therefore propose a targeted dataset condensation on specific partition of retain dataset, whose features are most far away from those of forget samples. To achieve this, we propose following steps:

Step 1: Partition the retain set with sampling $\mathcal{F}$ Using **Algorithm 2**, we systematically separate the retain images (using partitioned training set and $\mathbf{F}$) into:

- *free images* $\mathbf{R}_{\text{free}}$ whose features are **distant** from $\mathbf{F}$
- *residual images* $\mathbf{R}_{\text{residual}}$ that lie **close** to features of $\mathbf{F}$

Specifically, the inter-cluster indicator $\phi(j)$ and the intra-cluster indicator $\Phi(i)$ test whether the set of images $\{I_{ij}\}_{j=1}^{n_i}$ for each cluster j intersects the forget set $\mathbf{F}$, thereby partitioning $\mathbf{R}$ into $\mathbf{R}_{\text{free}}$ and $\mathbf{R}_{\text{residual}}$.

Step 2: Condense only the free retain subset Each cluster in the free retain subset (i.e. those with $\phi(j) \neq -1$) is condensed by blending its n_j images into a single prototype, thus creating condensed version of free retain set:

$$\mathbf{Condense}\big(\mathbf{R}_{\text{free}}\big) \;=\; \left\{ \{\, I_{ij} \,\}_{j=1}^{n_i} \;\xrightarrow{\text{blend}}\; \mathcal{I}\big(\omega_i\big) \right\}_{i=\phi(1)}^{\phi(kC)}. \tag{3}$$

allowing no synthetic point converging towards $\mathbf{F}$ features.

Algorithm 2 Sampling $\mathcal{F}\big(\mathbf{F}, \{\{I_{ij}\}_{i=1}^{n_j}\}_{j=1}^{k\times C}\big)$

Indices: $j = 1, \ldots, kC$ (cluster); $i = 1, \ldots, n_j$ (image in cluster j)

$\phi(j) = \begin{cases} -1, & \mathbf{F} \cap \{I_{ij}\}_{j=1}^{k\times C} = \emptyset \\ j, & \text{otherwise} \end{cases}$ ▷ indexing clusters with no $\mathbf{F}$

$\Phi(i) = \begin{cases} i, & \mathbf{F} \cap \{I_{ij}\}_{i=1}^{n_j} \\ -1, & \text{otherwise} \end{cases}$ ▷ indexing within $\mathbf{F}$ clusters

$\mathbf{R}_{\text{free}} = \{\{I_{ij}\}_{i=1}^{n_j}\}_{j=\phi(1)}^{\phi(kC)}$ ▷ $\mathbf{R}$ subset dissimilar to $\mathbf{F}$

$\mathbf{R}_{\text{residual}} = \{\{I_{ij}\}_{i=\Phi_j(1)}^{\Phi_j(n_j)}\}_{j=1}^{kC}$ ▷ $\mathbf{R}$ subset similar to $\mathbf{F}$

return $\mathbf{R}_{\text{free}}, \mathbf{R}_{\text{residual}}$

Step 3: Create the *Reduced* Retain Dataset Having partitioned the original retain set into the distant "free" clusters and the forget images sensitive "residual" images, and condensed only the former via the blending operation in Eq. equation 3, we now assemble our final training set.

$$\mathbf{R}_{\textbf{reduced}} = \underbrace{\mathbf{Condense}\big(\mathbf{R}_{\textbf{free}}\big)}_{\textbf{condensed, low-importance}} \cup \underbrace{\mathbf{R}_{\textbf{residual}}}_{\textbf{uncondensed, high-importance}}.$$

This reduced retain dataset preserves all samples that lie near the forget manifold (ensuring precise boundary control during unlearning), while summarising only those clusters that have no intersection with $\mathbf{F}$. As a result, $\mathbf{R}_{\text{reduced}}$ offers the computational savings from total dataset condensation, with advantage of not sacrificing the fine-grained feature information needed to effectively forget and persist generalization, which would have been possible with raw AMU.

3.4 Accelerated Approximate Unlearning

Carlini *et al.* Carlini et al. (2022) report that per-sample classification losses of deep networks are *strongly non-Gaussian*. They considered to "Gaussianise" losses by ad-hoc monotone maps (e.g. combination of $\exp$ and $\log$ maps), without guaranteeing that the output is normally distributed. We instead apply the probability–integral transform (PIT), i.e. for any continuous variable X with CDF F_X, $Z = \Phi^{-1}\big(F_X(X)\big)$ is standard normal, where $\mathbf{\Phi^{-1}}$ is the probit (quantile) function of the standard normal distribution $\mathcal{N}(0,1)$ (mean 0, variance 1). Substituting F_X by a differentiable empirical surrogate yields an end-to-end differentiable pipeline whose transformed losses converge to $\mathcal{N}(0,1)$.

Following this motivation, we intend to transform arbitrary random samples, which may not be following Gaussian distribution towards samples following standard normal distribution. Let $x = (x_1, \ldots, x_N) \overset{\text{i.i.d.}}{\sim} F$, such that F is an unknown continuous CDF. The empirical CDF at x_i is $\hat{F}_N(x_i) = \frac{1}{N}\sum_{j=1}^{N} \mathbf{1}\{x_j \le x_i\}$. Replacing the indicator by the temperature–controlled sigmoid $\sigma_K(z) = (1 + e^{-Kz})^{-1},\ K > 0$, gives the differentiable estimator

$$Q_i^{(K)}(x) = \frac{1}{N}\sum_{j=1}^{N} \sigma_K\big(x_i - x_j\big), \qquad i = 1, \ldots, N,$$

such that $\lim_{K\to\infty} Q_i^{(K)}(x) = \hat{F}_N(x)$. We stack all samples into coordinates of vector as

$$\mathbf{Q}^{(K)}(x) = \big(Q_1^{(K)}(x), \ldots, Q_N^{(K)}(x)\big)^\top \in (0,1)^N \tag{4}$$

Because each entry of $\mathbf{Q}^{(K)}(x)$ lies in $(0,1)$, apply the probit function to obtain van Albada & Robinson (2007)

$$\mathbf{z}^{(K)} = \mathbf{\Phi^{-1}}\big(\mathbf{Q}^{(K)}(x)\big) \in \mathbb{R}^N. \tag{5}$$

Thus any sample x from a continuous distribution is converted to a sample whose coordinates converge to i.i.d. standard normals as $K \to \infty$: $\mathbf{z}^{(K)} \xrightarrow{K\to\infty} \mathbf{\Phi^{-1}}\big(\hat{F}_N(x)\big)$. A finite but large K balances smooth gradients (for optimization) with true empirical CDF proximity.

3.4.1 TRANSFORMING LOSSES OF MODEL TO STANDARD NORMAL

For the frozen pretrained network with parameters θ, we define the per-sample log transformed network's original loss (e.g. cross-entropy) as $\ell(x, y) = \log\big(1 + \mathcal{L}(\theta; x, y)\big)$.

Collect all losses of $\mathbf{F}$ and $\mathbf{T}$:

$$\boldsymbol{\ell}_F = \big(\ell(x_1, y_1), \ldots, \ell(x_{n_F}, y_{n_F})\big)^\top, \boldsymbol{\ell}_T = \big(\ell(x'_1, y'_1), \ldots, \ell(x'_{n_T}, y'_{n_T})\big)^\top \text{ with } (x_i, y_i) \sim \mathcal{D}_F \text{ and } (x'_i, y'_i) \sim \mathcal{D}_T.$$

Applying the vector transformation from equation 5 to the loss vectors yields $\mathbf{Z}_F^{(K)} = \mathbf{\Phi^{-1}}\big(\mathbf{Q}^{(K)}(\boldsymbol{\ell}_F)\big)$, $\mathbf{Z}_T^{(K)} = \mathbf{\Phi^{-1}}\big(\mathbf{Q}^{(K)}(\boldsymbol{\ell}_T)\big)$, where $\mathbf{Q}^{(K)}$ is the differentiable CDF estimator from equation 4. As $K \to \infty$ the mapping converges to the empirical probit, and this allows use to play with transformed losses in distribution matching manner. As emphasize previously, practice we keep K sufficiently large to preserve differentiability.

3.4.2 MEMBERSHIP-INFERENCE ATTACK (MIA) LOSS

With the Gaussian kernel $\lambda(z, z') = \exp\big(-\|z - z'\|^2/2\big)$, the squared maximum mean discrepancy between the transformed loss distributions is

$$\mathcal{L}_{\mathrm{MMD}}(\theta) = \mathbb{E}_{z,z' \sim \mathbf{Z}_F^{(K)}}\big[\lambda(z, z')\big] + \mathbb{E}_{\tilde{z},\tilde{z}' \sim \mathbf{Z}_T^{(K)}}\big[\lambda(\tilde{z}, \tilde{z}')\big] - 2\,\mathbb{E}_{z \sim \mathbf{Z}_F^{(K)},\, \tilde{z} \sim \mathbf{Z}_T^{(K}}\big[\lambda(z, \tilde{z})\big]. \quad (6)$$

Minimising $\mathcal{L}_{\mathrm{MMD}}$ aligns the transformed loss distribution of the forget set with that of the class-matched test subset, thereby mitigating membership-inference leakage while remaining fully differentiable in θ and K.

3.4.3 ACCELERATED (A-) AMU OBJECTIVE

We come to the second thesis of this paper, i.e. to accelerate unlearning at unlearning objective level. We propose following **A-AMU** minimizing objective, which proposes ***two main solutions*** over a general machine unlearning loss objective (mentioned in 'Introduction' section).

$$\min_\theta \underbrace{\mathbb{E}_{(x,y)\sim\mathcal{D}_R}\big[\mathcal{L}(\theta; x, y)^2\big]}_{\textbf{Steeper Main Loss}} + \lambda \underbrace{\mathcal{L}_{\mathrm{MMD}}(\theta)}_{\textbf{Diff. MIA}} + \underbrace{\Gamma(\theta; \mathbf{R}, \mathbf{F})}_{\textbf{Other Reg.}}$$

where $\lambda > 0$ is regularization parameter.

Solution # 1: Steeper main loss. Replacing per-sample $\mathcal{L}$ by its square $\mathcal{L}^2$ multiplies the per-sample gradient magnitude by $2\,\mathcal{L}$. Consequently, gradient descent updates make larger progress early in training, in such a way that large errors on $\mathbf{R}$ are penalized disproportionately, while small errors become less important in course of optimization. So the model departs more rapidly from the decision boundaries, which it learned earlier during training schedule. In order to prevent overfitting on high error retain dataset samples, we take square of average loss per batch, instead of per sample squaring. This also induces stability on choice of learning rate. Infact, in all of our experiments, we set a fixed learning rate for our approach on all models and datasets (including other hyperparameters), which is in strong contrast from other SOTA **AMU** methods that require hectic finetuning on each setting.

Solution # 2: Differentiable MIA term. $\mathcal{L}_{\mathrm{MMD}}(\theta)$, from equation 6, essentially models something absolute in unlearning, i.e. for unlearned model, forget samples appears the same as test samples, which supersedes subjective modelling of forgetting regularizations in Kurmanji et al. (2023); Jia et al. (2023); Chundawat et al. (2023) dependent upon accuracy or parameter distribution hypothesis of unlearning. Minimizing this term over θ suppresses membership-inference signals on $\mathbf{F}$ while remaining fully differentiable in θ. This interplays with making main loss steeper, as the parameter rapidly shift to overfit on retain set (or its reduced version), the differentiable MIA term allow quick restriction of parameters to only those trajectory, where MIA score is minimum.

Steeper main loss + Differentiable MIA term. Combining the controlled instability from steeper main loss, around decision boundaries, with reduction of training trajectories towards absolute unlearning solutions, forms a strong acceleration of unlearning. As would be seen later in our experiments, **A-AMU** objective minimization achieves competitive MIA suppression and retain accuracy after ~ 1 epoch.

Additional Regularisation Γ. The regularization term $\Gamma(\theta; \mathbf{R}, \mathbf{F})$ serves as a characteristic placeholder for AMU unlearning ideas like weight sparsity Jia et al. (2023), negative distillation Kurmanji et al. (2023), bad teacher distillation Chundawat et al. (2023), etc.

Table 1: Unlearning ResNet-18/CIFAR-10 (mean $\pm$ 95% CI) with k-means partitioning time 5.5753 ± 0.1391 s, $\mathcal{F}$-sampling time 0.1184 ± 0.0107 s

Method	MIA Score ↓		Retain Accuracy ↑		Forget Accuracy ↓		Test Accuracy ↑		Unlearning Time (s) ↓	
	w/o DC	w DC	w/o DC	w DC	w/o DC	w DC	w/o DC	w DC	w/o DC	w DC
Random Class Forgetting (Condensation Time for 1000 IPC: 47.7812 ± 1.3323 s leading to 80.0% retain dataset reduction)										
Retraining	49.76 ± 2.57	50.51 ± 3.55	100.00 ± 0.00	75.50 ± 5.63	0.00 ± 0.00	0.00 ± 0.00	80.35 ± 2.08	62.47 ± 4.88	1141.71 ± 51.30	224.50 ± 0.14
CF	50.18 ± 1.16	55.39 ± 2.67	100.00 ± 0.00	96.39 ± 0.75	0.00 ± 0.00	0.30 ± 0.45	81.19 ± 2.40	77.75 ± 1.25	577.81 ± 55.36	112.32 ± 0.04
Bad Distillation	62.83 ± 9.14	57.31 ± 7.01	96.26 ± 0.72	98.88 ± 0.18	1.23 ± 3.90	24.99 ± 79.51	77.33 ± 2.37	81.19 ± 6.28	284.54 ± 28.72	117.04 ± 0.05
L_1 Sparse	47.32 ± 2.73	53.79 ± 0.88	99.00 ± 0.97	96.36 ± 0.55	0.00 ± 0.00	0.00 ± 0.00	78.42 ± 3.40	77.66 ± 1.32	358.62 ± 28.80	69.88 ± 0.01
SCRUB	50.12 ± 0.37	50.00 ± 0.00	15.28 ± 7.66	11.11 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	13.80 ± 6.90	10.00 ± 0.00	241.23 ± 23.22	61.58 ± 0.02
Pruning	50.87 ± 3.22	50.80 ± 2.27	98.08 ± 0.73	100.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	77.43 ± 2.41	72.65 ± 1.27	235.50 ± 6.20	46.74 ± 0.01
A-CF	51.71 ± 2.29	48.25 ± 7.31	97.88 ± 4.39	98.92 ± 0.62	0.00 ± 0.00	0.00 ± 0.00	77.01 ± 3.20	77.45 ± 3.63	75.76 ± 0.27	16.01 ± 0.96
10% Uniform Forgetting (Condensation Time for 1000 IPC: 34.1780 ± 0.7403 s leading to 39.8% retain dataset reduction)										
Retraining	50.47 ± 0.61	50.05 ± 1.64	100.00 ± 0.00	91.48 ± 0.41	87.48 ± 1.01	82.25 ± 1.01	86.84 ± 0.62	80.65 ± 0.48	1132.22 ± 33.71	664.73 ± 6.16
CF	52.45 ± 1.18	54.39 ± 0.89	100.00 ± 0.00	96.95 ± 0.17	93.30 ± 0.53	93.42 ± 0.08	88.11 ± 0.25	86.60 ± 0.20	561.76 ± 4.92	332.23 ± 2.78
Bad Distillation	82.92 ± 1.51	85.78 ± 2.43	93.78 ± 0.64	94.43 ± 0.98	27.99 ± 9.74	18.71 ± 9.63	78.77 ± 0.56	79.57 ± 1.42	275.64 ± 0.56	264.16 ± 1.54
L_1 Sparse	50.87 ± 1.39	52.22 ± 0.99	98.87 ± 0.23	92.81 ± 0.45	86.18 ± 0.60	86.30 ± 1.07	84.36 ± 0.35	82.10 ± 0.56	348.20 ± 0.98	206.46 ± 1.54
SCRUB	51.27 ± 0.42	51.59 ± 1.46	99.96 ± 0.01	77.94 ± 42.33	88.35 ± 1.39	73.11 ± 37.70	85.99 ± 1.09	70.56 ± 34.47	235.68 ± 1.63	149.79 ± 0.95
Pruning	51.08 ± 0.72	51.75 ± 0.68	96.76 ± 0.87	97.52 ± 1.28	85.67 ± 0.99	83.22 ± 2.08	82.51 ± 0.70	79.75 ± 2.08	234.11 ± 0.35	138.16 ± 0.96
A-CF	51.86 ± 0.56	51.95 ± 0.47	99.53 ± 0.85	99.76 ± 0.10	86.19 ± 1.41	85.42 ± 0.15	86.18 ± 2.50	86.23 ± 0.38	76.24 ± 1.13	47.32 ± 3.05

Table 2: Unlearning on ResNet-50/CIFAR-10 (mean $\pm$ 95% CI) with k-means partitioning time 5.5753 ± 0.1391 s, $\mathcal{F}$-sampling time 0.1184 ± 0.0107 s

Method	MIA Score ↓		Retain Accuracy ↑		Forget Accuracy ↓		Test Accuracy ↑		Unlearning Time (s) ↓	
	w/o DC	w DC	w/o DC	w DC	w/o DC	w DC	w/o DC	w DC	w/o DC	w DC
Random Class Forgetting (Condensation Time for 1000 IPC: 47.7812 ± 1.3323 s leading to 80.0% retain dataset reduction)										
Retraining	48.61 ± 1.85	49.49 ± 2.38	100.00 ± 0.00	68.45 ± 2.51	0.00 ± 0.00	0.00 ± 0.00	79.96 ± 1.10	56.46 ± 2.81	3580.71 ± 94.46	705.82 ± 0.30
CF	51.07 ± 3.71	57.87 ± 2.74	100.00 ± 0.00	97.89 ± 0.18	0.00 ± 0.00	1.65 ± 1.73	81.28 ± 1.33	79.11 ± 1.89	1789.40 ± 47.15	352.38 ± 0.74
Bad Distillation	57.97 ± 5.13	62.99 ± 6.81	97.31 ± 1.59	99.69 ± 0.09	0.00 ± 0.00	0.01 ± 0.03	77.68 ± 2.36	80.64 ± 1.16	1370.90 ± 34.32	572.67 ± 0.69
L_1 Sparse	49.18 ± 2.20	58.95 ± 4.72	98.52 ± 0.55	98.54 ± 0.32	0.00 ± 0.00	4.03 ± 3.58	77.61 ± 2.12	79.61 ± 1.49	1446.43 ± 6.34	288.65 ± 0.31
SCRUB	50.00 ± 0.00	50.00 ± 0.00	15.50 ± 3.00	11.14 ± 0.06	0.00 ± 0.00	0.00 ± 0.00	13.88 ± 2.72	10.05 ± 0.09	759.90 ± 9.19	198.01 ± 0.21
Pruning	48.37 ± 2.43	50.56 ± 0.81	97.10 ± 1.14	99.08 ± 0.63	0.00 ± 0.00	0.00 ± 0.00	76.85 ± 1.84	69.80 ± 3.61	722.06 ± 5.91	143.73 ± 0.23
A-CF	52.10 ± 1.50	50.99 ± 0.67	97.75 ± 1.50	95.90 ± 2.73	0.00 ± 0.00	0.10 ± 0.19	77.03 ± 1.08	75.90 ± 3.82	218.30 ± 0.26	44.54 ± 1.80
10% Uniform Forgetting (Condensation Time for 1000 IPC: 34.1780 ± 0.7403 s leading to 39.8% retain dataset reduction)										
Retraining	49.46 ± 0.87	49.46 ± 1.29	100.00 ± 0.00	91.26 ± 0.59	87.41 ± 1.05	80.88 ± 0.78	87.01 ± 0.16	80.41 ± 0.64	3554.87 ± 92.12	2095.34 ± 29.30
CF	53.32 ± 0.90	54.20 ± 0.56	100.00 ± 0.00	97.79 ± 0.22	94.34 ± 0.89	95.19 ± 0.36	88.64 ± 0.15	87.52 ± 0.83	1770.13 ± 13.17	1046.32 ± 16.59
Bad Distillation	84.07 ± 1.69	88.42 ± 1.62	94.77 ± 0.30	96.16 ± 0.92	23.76 ± 9.60	11.93 ± 1.87	79.80 ± 1.24	81.67 ± 1.86	1371.63 ± 35.81	1297.61 ± 15.55
L_1 Sparse	52.10 ± 1.60	53.25 ± 0.38	98.25 ± 0.39	95.31 ± 0.91	87.41 ± 1.22	90.13 ± 1.93	84.25 ± 1.17	84.07 ± 0.90	1463.85 ± 38.41	856.47 ± 13.84
SCRUB	51.97 ± 0.91	52.32 ± 2.80	97.99 ± 3.40	86.19 ± 33.28	87.38 ± 6.03	83.37 ± 30.90	84.77 ± 3.59	78.17 ± 27.14	756.68 ± 1.36	483.13 ± 5.28
Pruning	51.76 ± 0.54	51.43 ± 0.93	96.81 ± 1.42	97.62 ± 0.39	86.66 ± 0.69	84.71 ± 0.77	83.40 ± 1.58	81.61 ± 0.68	730.16 ± 20.13	426.54 ± 6.89
A-CF	50.20 ± 2.65	52.27 ± 1.46	95.49 ± 4.84	97.06 ± 2.01	82.57 ± 3.42	84.83 ± 2.89	81.72 ± 5.24	84.03 ± 1.15	218.79 ± 0.34	133.32 ± 5.23

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Methodology We evaluate our methods on CIFAR-10, SVHN, and CINIC-10 datasets using ResNet-18 and ResNet-50 models. Our approach is benchmarked against six AMU methods: **Retraining**, **Catastrophic Forgetting (CF)** (fine-tuning) Kurmanji et al. (2023); Jia et al. (2023), **Bad Distillation** Chundawat et al. (2023), $\mathbf{L_1}$-Sparsity Jia et al. (2023), **Pruning** Jia et al. (2023), and **SCRUB** Kurmanji et al. (2023). For a fair comparison, implementations were adapted from their official repositories, and hyperparameters were rigorously fine-tuned for each model and dataset.[1]. We categorize these baselines into three tiers for analysis: *naive* (**Retraining**, **CF**), *non-naive* ($\mathbf{L_1}$-sparse, **Pruning**, **SCRUB**, **Bad Distillation**), and our proposed *accelerated* (**A-**) methods.

Evaluation Unlearning performance is assessed across five key metrics: **Test Accuracy** (↑), **Retain Accuracy** (↑), **Forget Accuracy** (↓), **Unlearning Time** (↓), and a **MIA Score** (↓) for privacy. The MIA score is derived from a powerful white-box LiRA attack Carlini et al. (2022), providing a robust measure of privacy risk. For our **Blend** method, the k-means partitioning feature extractor is a lightweight CNN, $35\times$ and $74\times$ smaller than ResNet-18 and ResNet-50 respectively, ensuring its preprocessing overhead remains minimal. All experiments were conducted on a single NVIDIA RTX 3070 GPU (8GB VRAM), except for experiments in Sections 4.4 and 4.5, which were performed on a laptop RTX 3070 (8GB VRAM) due to computational resource availability during the extended experimental phase.

Categorization: We categorize baselines into three tiers: *naive* (**Retraining**, **CF**), *non-naive* ($\mathbf{L_1}$-sparse, **Pruning**, **SCRUB**, **Bad Distillation**), and *accelerated* (**A-**), the last of which can contain acceleration of all naive and non-naive AMU methods.

4.2 THE EFFICIENCY AND EFFICACY OF BLEND CONDENSATION

Our proposed dataset condensation method, **Blend**, is designed for both speed and efficacy, hence targeting unlearning. Its primary innovation is a reparameterization of the distribution matching objective that significantly reduces the number of trainable parameters compared to SOTA like Distribution Matching (**DM**), Gradient Matching (**GM**) and Improved Distribution Matching (**IDM**) Zhao & Bilen (2023); Zhao et al. (2020; 2023). For instance, on CIFAR-10 with an Images Per Class (IPC) of 10, **Blend** requires only 5×10^4 parameters, whereas DM requires 30.72×10^4.

[1]Complete hyperparameter configurations are available in the 'hyperparameters' folder of our code repository: `https://github.com/algebraicdianuj/DC_Unlearning`

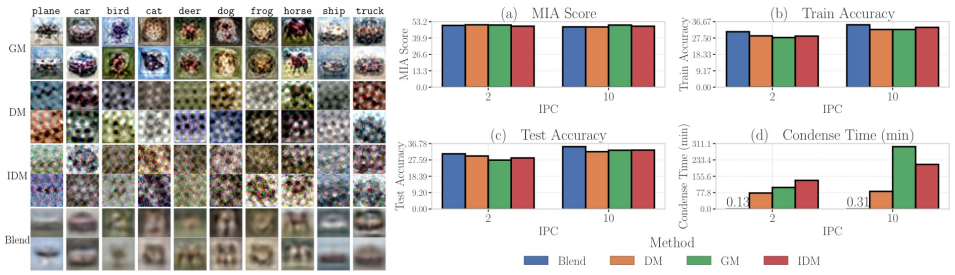

Figure 2: Qualitative and quantitative comparison of condensation methods on CIFAR-10/CNN. Top: class-wise synthetic prototypes generated by **GM**, **DM**, **IDM**, and our **Blend** method. Bottom: Performance metrics for each method over course of IPC = 2, 10.

We further shed light on this by conducting experiments on CNN and CIFAR-10. A key distinction in our setup is that **Blend**'s optimization process is limited to a synthetic data batch size of 1 in our implementation (as batched implementation is non-trivial). In contrast, competing methods like **GM**, **DM**, and **IDM** leverage large synthetic batch sizes (e.g., 256) for parallelized optimization. Despite being unable to use this common acceleration technique, **Blend**'s architectural efficiency leads to a dramatic performance gap. On CIFAR-10, **Blend** completes condensation in just 8-18 seconds (0.13–0.31 minutes). In stark contrast, GM, DM, and IDM require roughly 75–300 minutes, making our method over 1500x faster on average.

This dramatic speed-up does not compromise performance. As shown in Figure 2, **Blend** consistently yields higher test accuracy (e.g., 35.03% at IPC=10) compared to **GM** (33.01%), **DM** (32.2%), and **IDM** (33.1%). Furthermore, the synthetic prototypes generated by **Blend** are qualitatively superior. They are noticeably smoother and more class-consistent (e.g., clear airplane outlines) than the noisy, cluttered images from other methods. This demonstrates that our approach of grouping and blending feature-similar images is highly effective at producing coherent and informative exemplars.

4.3 Single-Round Unlearning

We benchmark single-round unlearning across **utility** metrics (Test/Retain Accuracy; higher is better) and **privacy** metrics (Forget Accuracy/MIA Score; lower is better). We analyze two orthogonal improvements: (i) our **Blend** dataset condensation method, which shrinks the retain set, and (ii) our loss-centric **A-AMU** algorithm, only applied as acceleration of **CF**→**A-CF** for brevity. The unlearning time in tables is separated from overhead from k-means partitioning and the Blend condensation (in case of "w/ DC") for distinction.

Situational Impact of Blend The effectiveness of **Blend** scales with the data reduction ratio. For *random class removal* on SVHN, shrinking the retain set to just 2.2% enabled the **Pruning** method to achieve a 96.3% total time reduction (from 647.4s to 23.7s) at the cost of only 3.9 percentage points in Test Accuracy, though the MIA score increased by 2.4. On CIFAR-10, a reduction to 20.0% of the retain set gave **Retraining** a 75.7% time reduction, but with a more substantial utility drop of 17.9 points in Test Accuracy and 24.5 points in Retain Accuracy. For *10% uniform forgetting* on CIFAR-10 (60.2% retain), the gains were more modest: **Retraining** time was reduced by 37.8%, with a 6.2-point drop in Test Accuracy and a slight improvement in the MIA score (from 50.47 to 50.05). Thus, **Blend** offers dramatic speed-ups when data reduction is large, but this can introduce a trade-off with model utility.

Consistent Acceleration with A-AMU Our **A-AMU** framework provides consistent algorithmic acceleration, independent of the unlearning scenario. On CIFAR-10 and without condensation, **A-CF** is *7.6x* faster than naive CF for class removal (75.8s vs. 577.8s) and *7.4x* faster for uniform forgetting (76.2s vs. 561.8s). This performance is achieved using a fixed set of hyperparameters for **A-CF** across all tasks.

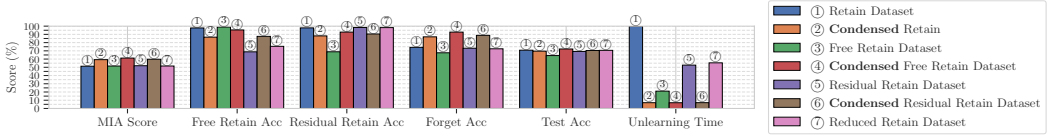

Figure 3: Impact of different condensation strategies (on **Blend**) on 10% random unlearning performance of $\mathbf{L_1}$-sparse on ResNet-18/CINIC-10

Table 3: Unlearning on ResNet-18/SVHN (mean ± 95% CI) with k-means partitioning time 4.1379 ± 0.3221 s, $\mathcal{F}$-sampling time 0.0537 ± 0.0089 s

Method	MIA Score ↓		Retain Accuracy ↑		Forget Accuracy ↓		Test Accuracy ↑		Unlearning Time (s) ↓	
	w/o DC	w DC	w/o DC	w DC	w/o DC	w DC	w/o DC	w DC	w/o DC	w DC
Random Class Forgetting (Condensation Time for 100 IPC: 4.90615 ± 0.1941 s leading to 97.8% retain dataset reduction)										
Retraining	50.45 ± 1.53	52.01 ± 5.38	99.99 ± 0.00	78.46 ± 3.28	0.00 ± 0.00	0.00 ± 0.00	82.71 ± 10.54	68.53 ± 8.79	1012.11 ± 4.18	22.79 ± 0.04
CF	50.34 ± 3.97	51.15 ± 0.32	100.00 ± 0.00	91.37 ± 0.40	0.00 ± 0.00	0.00 ± 0.00	82.37 ± 10.30	81.30 ± 0.25	505.95 ± 2.05	11.36 ± 0.01
Bad Distillation	55.38 ± 1.74	49.58 ± 6.42	98.49 ± 2.02	99.69 ± 0.72	0.00 ± 0.00	0.00 ± 0.00	81.84 ± 9.47	83.04 ± 9.03	248.95 ± 1.34	45.15 ± 0.03
L_1 Sparse	53.71 ± 3.01	52.07 ± 1.39	99.16 ± 0.96	96.97 ± 0.13	0.00 ± 0.00	0.00 ± 0.00	84.04 ± 1.87	94.18 ± 0.19	314.88 ± 3.28	7.14 ± 0.00
SCRUB	50.47 ± 1.64	49.97 ± 0.03	12.93 ± 2.81	11.11 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	13.61 ± 8.37	7.37 ± 0.71	211.82 ± 2.00	19.57 ± 0.02
Pruning	49.77 ± 2.35	51.90 ± 2.20	99.04 ± 0.18	100.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	80.80 ± 10.02	77.36 ± 8.78	210.93 ± 1.00	4.80 ± 0.01
A-CF	52.04 ± 2.70	54.09 ± 6.30	99.65 ± 0.59	97.55 ± 0.71	0.00 ± 0.00	0.00 ± 0.00	80.90 ± 9.38	84.49 ± 2.80	83.27 ± 0.48	2.07 ± 0.02
10% Uniform Forgetting (Condensation Time for 100 IPC: 0.4733 ± 0.0759 s leading to 1.6% retain dataset reduction)										
Retraining	51.75 ± 1.91	51.61 ± 3.54	99.99 ± 0.00	99.76 ± 0.03	94.60 ± 0.59	94.71 ± 0.26	95.18 ± 0.16	95.12 ± 0.17	1011.96 ± 6.45	992.98 ± 7.97
CF	53.67 ± 0.51	53.32 ± 0.37	100.00 ± 0.00	99.86 ± 0.02	97.20 ± 0.03	97.06 ± 0.55	94.89 ± 0.07	95.09 ± 0.22	505.06 ± 3.72	496.01 ± 3.47
Bad Distillation	92.23 ± 0.43	93.57 ± 0.44	96.94 ± 1.02	99.75 ± 0.11	23.77 ± 8.50	23.77 ± 2.00	89.73 ± 2.44	91.72 ± 0.41	248.75 ± 0.95	369.65 ± 2.02
L_1 Sparse	53.55 ± 1.96	53.12 ± 2.44	99.32 ± 0.44	99.47 ± 0.21	93.80 ± 1.00	94.32 ± 0.87	93.60 ± 0.57	93.33 ± 0.20	314.80 ± 2.90	308.10 ± 1.58
SCRUB	54.99 ± 1.47	56.51 ± 4.66	99.66 ± 0.07	99.99 ± 0.02	95.00 ± 0.29	99.07 ± 0.44	94.48 ± 0.31	95.13 ± 0.19	211.64 ± 0.85	214.29 ± 1.76
Pruning	54.20 ± 1.60	53.55 ± 0.72	99.33 ± 0.10	99.18 ± 0.40	94.91 ± 0.16	94.94 ± 0.71	93.34 ± 0.41	93.50 ± 0.52	210.79 ± 1.00	206.95 ± 1.34
A-CF	54.89 ± 2.23	53.96 ± 1.56	99.48 ± 0.47	99.48 ± 0.32	93.13 ± 1.94	93.50 ± 1.46	94.09 ± 1.06	93.66 ± 1.56	85.83 ± 2.01	82.83 ± 1.35

Table 4: Unlearning on ResNet-50/SVHN (mean ± 95% CI) with k-means partitioning time 4.1379 ± 0.3221 s, $\mathcal{F}$-sampling time 0.0537 ± 0.0089 s

Method	MIA Score ↓		Retain Accuracy ↑		Forget Accuracy ↓		Test Accuracy ↑		Unlearning Time (s) ↓	
	w/o DC	w DC	w/o DC	w DC	w/o DC	w DC	w/o DC	w DC	w/o DC	w DC
Random Class Forgetting (Condensation Time for 100 IPC: 4.90615 ± 0.1941 s leading to 97.8% retain dataset reduction))										
Retraining	51.57 ± 0.76	51.41 ± 0.77	99.97 ± 0.01	40.85 ± 16.59	0.00 ± 0.00	0.00 ± 0.00	85.33 ± 9.14	36.53 ± 16.44	3177.10 ± 18.08	71.35 ± 0.22
CF	51.34 ± 1.03	46.39 ± 2.68	99.97 ± 0.02	94.73 ± 0.02	0.00 ± 0.00	62.04 ± 0.09	85.45 ± 9.50	88.79 ± 0.56	1587.86 ± 8.52	35.61 ± 0.09
Bad Distillation	48.79 ± 4.05	49.77 ± 8.64	99.91 ± 0.01	99.82 ± 0.38	0.00 ± 0.09	00.00 ± 0.87	89.99 ± 5.84	87.59 ± 6.84	1217.71 ± 6.58	221.71 ± 0.58
L_1 Sparse	52.67 ± 5.92	59.40 ± 2.35	99.24 ± 0.10	96.64 ± 0.30	0.00 ± 0.00	0.00 ± 0.00	83.98 ± 9.43	83.07 ± 1.00	1298.78 ± 6.70	29.29 ± 0.06
SCRUB	49.99 ± 0.04	49.97 ± 0.04	11.70 ± 0.70	11.11 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	12.08 ± 9.45	7.63 ± 0.13	679.75 ± 3.76	63.05 ± 0.19
Pruning	51.56 ± 3.15	53.94 ± 3.66	99.00 ± 0.23	100.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	84.31 ± 9.53	80.41 ± 3.55	647.44 ± 4.07	14.62 ± 0.05
A-CF	48.98 ± 3.16	53.30 ± 7.61	97.31 ± 0.84	94.92 ± 0.73	0.00 ± 0.00	0.00 ± 0.00	85.92 ± 3.27	78.39 ± 8.36	211.69 ± 1.13	4.99 ± 0.02
10% Uniform Forgetting (Condensation Time for 100 IPC: 0.47335 ± 0.0759 s leading to 1.6% retain dataset reduction)										
Retraining	51.68 ± 4.66	51.03 ± 3.50	99.98 ± 0.02	99.75 ± 0.04	94.62 ± 0.14	94.44 ± 0.14	95.00 ± 0.15	94.86 ± 0.30	3181.39 ± 17.57	3124.95 ± 19.68
CF	54.67 ± 1.51	53.69 ± 2.54	99.97 ± 0.00	99.87 ± 0.03	97.53 ± 0.12	97.48 ± 0.07	95.22 ± 0.13	95.24 ± 0.11	1587.19 ± 9.69	1561.58 ± 12.57
Bad Distillation	92.25 ± 0.98	92.76 ± 1.05	96.90 ± 0.84	99.71 ± 0.20	20.13 ± 9.61	22.32 ± 4.01	89.47 ± 1.19	92.16 ± 1.11	1218.23 ± 7.08	1818.03 ± 11.68
L_1 Sparse	54.98 ± 4.00	54.43 ± 1.57	99.16 ± 0.21	99.50 ± 0.15	94.06 ± 0.97	95.77 ± 0.31	93.13 ± 1.06	93.82 ± 0.91	1299.93 ± 6.71	1278.57 ± 12.03
SCRUB	53.62 ± 1.89	57.10 ± 2.62	99.17 ± 0.63	99.96 ± 0.04	93.29 ± 1.02	99.06 ± 0.35	93.58 ± 1.16	95.17 ± 0.31	681.54 ± 3.61	690.14 ± 5.75
Pruning	54.42 ± 0.62	55.28 ± 4.82	99.06 ± 0.21	98.39 ± 1.19	95.03 ± 0.67	94.17 ± 1.87	93.73 ± 0.37	91.82 ± 3.67	647.93 ± 3.91	637.52 ± 5.97
A-CF	54.89 ± 1.89	54.29 ± 2.43	98.35 ± 1.00	96.82 ± 1.24	93.10 ± 1.92	91.46 ± 1.00	93.41 ± 1.64	92.24 ± 1.24	214.15 ± 2.44	207.61 ± 0.83

Combined Impact When our two proposals are combined, the outcome depends on the scenario. For class removal, adding **Blend** to the already-fast **A-CF** provides a further 8.4% speed-up (from 75.8s to 69.4s total time). However, for uniform forgetting, where data reduction is modest, **Blend**'s overhead leads to a 14.2% slowdown (from 76.2s to 87.1s). This indicates that for fast algorithms with limited data reduction, standalone **A-AMU** is the superior strategy.

Bird's-Eye View In summary, our proposals offer two distinct tools. **Blend** is a powerful situational method whose time savings are proportional to the data reduction, though this may involve a trade-off with model utility. **A-CF** is a universal, tuning-free algorithmic accelerator. Ultimately, **A-CF** delivers the best overall balance of speed, utility, and privacy, consistently performing within 5 percentage points of the best-in-class score across every setting we tested, making it the most robust and practical solution.

4.4 Empirical Justification of Reduced Retain Dataset as Condensed Dataset

We evaluate the impact of various condensation strategies on the retain set for the $\mathbf{L_1}$-sparsity unlearning method (Figure 3). Experiments use a ResNet-18 pretrained on CINIC-10 with 10% uniform unlearning. The model's extreme overfitting (99.98% train vs. 73.16% test accuracy) thus represents a worst-case, poor-generalization scenario to stress-test condensation effects Li et al. (2025), as iterative unlearning fails when gradients vanish on overfitted data. Building upon Section 3.3, we systematically quantify how naive condensation trades off privacy and efficiency under these challenging conditions.

- **Full condensation dulls unlearning (②).** Forget-accuracy rises to $\approx 87\%$ and the MIA score increases, confirming that global prototypes produce features arbitrarily close to features of $\mathbf{F}$.
- **Single-partition training is lopsided (③: raw $\mathbf{R}_{\text{free}}$, ⑤: raw $\mathbf{R}_{\text{residual}}$).** Training only on $\mathbf{R}_{\text{free}}$ lowers forget-accuracy to $\approx 67\%$ but also drags test accuracy on the residual side to $\approx 64\%$. Training only on $\mathbf{R}_{\text{residual}}$ keeps generalization high, yet leaves forget-accuracy essentially unchanged, revealing a privacy-utility trade-off.
- **Condensing the partition still dulls unlearning (④: condensed-free, ⑥: condensed-residual).** Summarizing $\mathbf{R}_{\text{free}}$ (④) floods training with coarse prototypes that arbitrarily fall near forget features; and the model re-memorizes $\mathbf{F}$ (forget-accuracy $\approx 93\%$). Summarizing $\mathbf{R}_{\text{residual}}$ (⑥) removes boundary detail, pushing forget-accuracy back up to $\approx 88\%$ and depressing *free-retain* accuracy into the high-80s.
- **Selective condensation bridges every gap (⑦).** By *preserving* boundary-critical residual images and *condensing* only the distant free clusters, our reduced retain set (i) reaches a

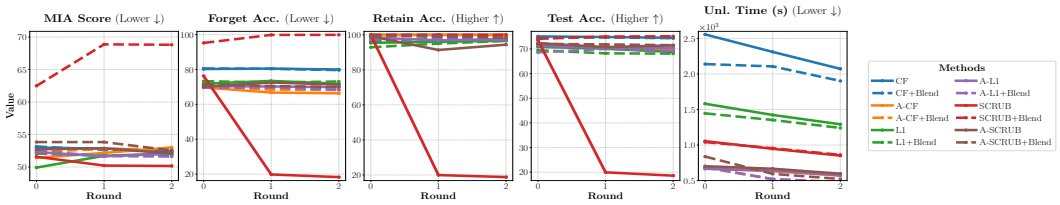

Figure 4: Multi-round sequential unlearning (random 10 %) performance of without accelerated **CF**, **L1**-sparse, and **SCRUB**, with acceleration (**A-CF**, **A-L$_1$**, **A-SCRUB**), with and without dataset condensation (**Blend**). With dataset condensation allows 34.9% retain dataset reduction

forget-accuracy of 72.5%, matching the best single-partition strategy (⑤) while dramatically outperforming the free-only alternative (③); (ii) lifts *free-retain* accuracy from the low-60s (⑤) up to 75%, closing the generalization gap without sacrificing privacy; (iii) maintains the baseline MIA score; and (iv) trims unlearning time from 100% (①) down to 63.8%, matching the speed of training on $\mathbf{R}_{\text{residual}}$, yet without its drop in free-retain performance.

4.5 Multi-Round Unlearning

We tested three base methods—**CF**, **L$_1$**-Sparsity, and **SCRUB**—and their accelerated variants—**A-CF**, **A-L$_1$**, **A-SCRUB**) over three sequential 10% uniform-forgetting rounds on ResNet-18 with CINIC-10. Experiments were run with and without **Blend** condensation, which adds a one-time 95.6s overhead. Figure 4 plots these results.

Distinct Roles of Blend and A-AMU Our two proposals address different challenges in sequential unlearning. **Blend** condensation acts as a stabilizer for unstable unlearning methods. This effect was visible on **SCRUB**, where it completely prevented the Test Accuracy collapse seen in the standalone version (which plummeted from 73.5% to 18.6%), maintaining performance at a stable $\approx$ 75% across all rounds. However, this utility stabilization came at the cost of privacy, as **SCRUB**'s Forget Accuracy and MIA score were notably higher with condensation (e.g., in Round 0, MIA score increased from 51.58% to 62.48%). For already stable methods, **Blend**'s impact on speed was modest (a 4-8% time reduction for **CF**) to negligible (**L$_1$**-Sparsity). In contrast, our **A-AMU** framework provides a more substantial, upfront acceleration. In Round 0, standalone **A-CF** is *3.8x faster* than **CF** (671.5s vs. 2556.6s), and **A-SCRUB** is *1.5x faster* than **SCRUB** while also inherently preventing its performance collapse.

Combined Performance and Long-Term Benefits When both proposals are combined in this low-data-reduction scenario, **Blend**'s overhead leads to a slight initial slowdown for the already-fast accelerated methods. For instance, the total time for **A-CF** with condensation in Round 0 is 779.2s, compared to 671.5s for standalone **A-CF**. However, the combination shows a steeper decline in runtime over subsequent rounds, with the condensed version's runtime decreasing by 30.2% from Round 0 to Round 2.

Bird's-Eye View Our multi-round analysis reveals two main findings. First, **Blend** condensation is a powerful tool for ensuring utility stability, rescuing volatile methods like **SCRUB** from catastrophic failure, albeit with a trade-off in privacy metrics. Second, our **A-AMU** framework provides a more significant and immediate acceleration (1.5-3.8x) than condensation in this high-frequency unlearning scenario. Together, they form a robust toolkit for sequential unlearning.

5 Conclusion

In this work, we addressed the critical efficiency bottlenecks in approximate machine unlearning (AMU) by introducing a dual data- and loss-centric framework. ur first contribution, Blend, is a highly efficient dataset condensation method that dramatically reduces unlearning time in high-data-reduction scenarios like class removal. Our second, A-AMU, is a novel loss-centric framework that provides consistent, significant algorithmic acceleration and unparalleled hyperparameter stability. These proposals allow cutting end-to-end unlearning latency by 84.6% in single-round and 54.5% in multi-round settings. Loss-centric proposal consistently maintains a strong balance across all utility and privacy metrics, while data-centric see good gains when original unlearning time is large, and reduction of dataset is around 50%. More broadly, we revitalize dataset condensation and MIA-based regularization as powerful tools for this domain. We hope our work paves the way for a new line of research focused on creating truly practical and efficient machine unlearning systems.

REFERENCES

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)*, pp. 141–159. IEEE, 2021.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, pp. 463–480. IEEE, 2015.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 1897–1914. IEEE, 2022.

George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4750–4759, 2022.

Min Chen, Weizhuo Gao, Gaoyang Liu, Kai Peng, and Chen Wang. Boundary unlearning: Rapid forgetting of deep networks via shifting the decision boundary. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7766–7775, 2023.

Vikram S Chundawat, Ayush K Tarun, Murari Mandal, and Mohan Kankanhalli. Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 7210–7217, 2023.

Akash Dhasade, Yaohong Ding, Song Guo, Anne-Marie Kermarrec, Martijn de Vos, and Leijie Wu. Quickdrop: Efficient federated unlearning via synthetic data generation. In *Proceedings of the 25th International Middleware Conference*, pp. 266–278, 2024.

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.

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*, pp. 9304–9312, 2020.

Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11516–11524, 2021.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*, 2019.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Jinghan Jia, Jiancheng Liu, Parikshit Ram, Yuguang Yao, Gaowen Liu, Yang Liu, Pranay Sharma, and Sijia Liu. Model sparsification can simplify machine unlearning. *arXiv preprint arXiv:2304.04934*, 2023.

Jang-Hyun Kim, Jinuk Kim, Seong Joon Oh, Sangdoo Yun, Hwanjun Song, Joonhyun Jeong, Jung-Woo Ha, and Hyun Oh Song. Dataset condensation via efficient synthetic-data parameterization. In *International Conference on Machine Learning*, pp. 11102–11118. PMLR, 2022.

Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. Towards unbounded machine unlearning. *Advances in neural information processing systems*, 36:1957–1987, 2023.

Bruce W Lee, Addie Foote, Alex Infanger, Leni Shor, Harish Kamath, Jacob Goldman-Wetzler, Bryce Woodworth, Alex Cloud, and Alexander Matt Turner. Distillation robustifies unlearning. *arXiv preprint arXiv:2506.06278*, 2025.

Fan Li, Xiaoyang Wang, Dawei Cheng, Wenjie Zhang, Ying Zhang, and Xuemin Lin. Tcgu: Data-centric graph unlearning based on transferable condensation. *arXiv preprint arXiv:2410.06480*, 2024.

Xiang Li, Wenqi Wei, and Bhavani Thuraisingham. Mubox: A critical evaluation framework of deep machine unlearning [systematization of knowledge paper]. In *Proceedings of the 30th ACM Symposium on Access Control Models and Technologies*, pp. 175–188, 2025.

Ananth Mahadevan and Michael Mathioudakis. Certifiable machine unlearning for linear models. *arXiv preprint arXiv:2106.15093*, 2021.

Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. Dataset distillation with infinitely wide convolutional networks. *Advances in Neural Information Processing Systems*, 34:5186–5198, 2021.

Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in neural information processing systems*, 33:6377–6389, 2020.

Sacha Jennifer van Albada and Peter A Robinson. Transformation of arbitrary distributions to the normal distribution with application to eeg test–retest reliability. *Journal of neuroscience methods*, 161(2):205–211, 2007.

Kai Wang, Zekai Li, Zhi-Qi Cheng, Samir Khaki, Ahmad Sajedi, Ramakrishna Vedantam, Konstantinos N Plataniotis, Alexander Hauptmann, and Yang You. Emphasizing discriminative features for dataset distillation in complex scenarios. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 30451–30461, 2025.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. Machine unlearning of features and labels. *arXiv preprint arXiv:2108.11577*, 2021.

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

Haonan Yan, Xiaoguang Li, Ziyao Guo, Hui Li, Fenghua Li, and Xiaodong Lin. Arcane: An efficient architecture for exact machine unlearning. In *IJCAI*, volume 6, pp. 19, 2022.

Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*, pp. 12674–12685. PMLR, 2021.

Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 6514–6523, 2023.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. *arXiv preprint arXiv:2006.05929*, 2020.

Ganlong Zhao, Guanbin Li, Yipeng Qin, and Yizhou Yu. Improved distribution matching for dataset condensation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7856–7865, 2023.

Yu Zhou, Dian Zheng, Qijie Mo, Renjie Lu, Kun-Yu Lin, and Wei-Shi Zheng. Decoupled distillation to erase: A general unlearning method for any class-centric tasks. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 20350–20359, 2025.