SAP: Corrective Machine Unlearning with Scaled Activation Projection for Label Noise Robustness

Sangamesh Kodge, Deepak Ravikumar, Gobinda Saha, Kaushik Roy

Elmore Family School of Electrical and Computer Engineering
Purdue University, West Lafayette, Indiana, USA
skodge@purdue.edu, dravikum@purdue.edu, gsaha@purdue.edu, kaushik@purdue.edu

Abstract

Label corruption, where training samples are mislabeled due to non-expert annotation or adversarial attacks, significantly degrades model performance. Acquiring large, perfectly labeled datasets is costly, and retraining models from scratch is computationally expensive. To address this, we introduce *S*caled *A*ctivation *P*rojection (*SAP*), a novel SVD (Singular Value Decomposition)-based corrective machine unlearning algorithm. *SAP* mitigates label noise by identifying a small subset of trusted samples using cross-entropy loss and projecting model weights onto a clean activation space estimated using SVD on these trusted samples. This process suppresses the noise introduced in activations due to the mislabeled samples. In our experiments, we demonstrate *SAP*'s effectiveness on synthetic noise with different settings and real-world label noise. *SAP* applied to the CIFAR dataset with 25% synthetic corruption show upto 6% generalization improvements. Additionally, *SAP* can improve the generalization over noise robust training approaches on CIFAR dataset by $\sim 3.2\%$ on average. Further, we observe generalization improvements of 2.31% for a Vision Transformer model trained on naturally corrupted Clothing1M.

Code — https://github.com/sangamesh-kodge/SAP.git
Extended version — https://arxiv.org/abs/2403.08618

Introduction

Deep learning models have revolutionized various fields like natural language processing and computer vision, largely due to the availability of massive datasets. However, the very scale of these datasets presents a challenge: guaranteeing accurate labeling as shown in Figure 1. Recent studies (Northcutt, Athalye, and Mueller 2021), have exposed the significant presence of labeling errors in widely used benchmarks. These errors can be unintentional, arising from human error (Sambasivan et al. 2021) or in recent times automated labeling using Large Language models (Wang et al. 2024; Pangakis and Wolken 2024). Alternatively, they can be deliberate, introduced through attacks like data poisoning (Schwarzschild et al. 2021; Biggio, Nelson, and Laskov 2012; Jagielski et al. 2018; Chen et al. 2017, 2022). Regardless of origin, these errors degrade model performance.

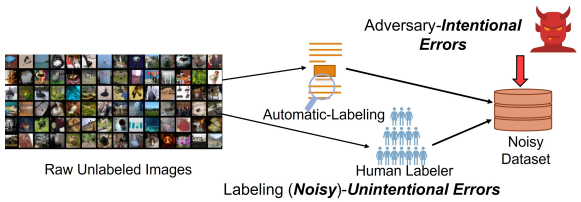


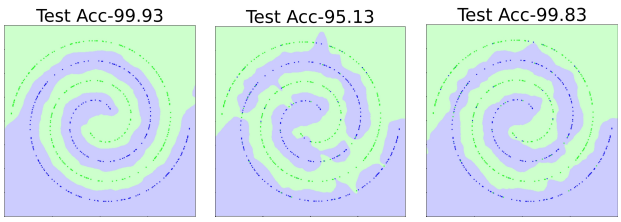
Figure 1: Sources of label noise - Intentional/Unintentional Errors.

While deep learning models exhibit robustness to a certain degree of label noise, they are not immune to label noise, as demonstrated in (Steinhardt, Koh, and Liang 2017).

Corrective Machine Unlearning (Goel et al. 2024) is emerging as a promising solution to the aforementioned challenge, with the objective of reducing the impact of mislabeled data by manipulating the trained model. A vast majority of unlearning algorithms (Triantafillou et al. 2024) require knowledge of which samples are mislabeled to partition the data into the retain set and the forget set. It is challenging to distinguish between mislabeled samples and hard to learn samples (Garg, Ravikumar, and Roy 2024). This presents a major challenge to deploy machine unlearning algorithms for tackling label noise.

This work introduces *SAP*, a novel algorithm based on Singular Value Decomposition (SVD) to improve model generalization in the presence of label noise. Notably, our approach achieves this improvement with a single update to the model weights. *SAP* leverages a small set of samples identified as having low cross-entropy loss. These samples are hypothesized to be correctly labeled and form a "clean" subset. We utilize this clean subset and SVD to compute an update for the model weights. This update involves estimating a clean activation space and projecting the model weights onto this space. This projection aims to suppress activations corresponding to potentially corrupted activations, improving model performance on unseen data. As *SAP* relies on the estimated clean subset (or retain set), we avoid explicitly detecting mislabeled data, which can be more difficult to obtain.

In order to visualize the effectiveness of *SAP*, we trained a network for binary classification task on clean training data



(a) Clean model (b) Corrupt model (c) SAP applied to corrupt model

Figure 2: Decision boundaries for a network trained the 2D spiral data (a) with full Clean data, (b) with 10% corrupt data and (c) *SAP* applied to network trained on corrupt data.

(clean model) and corrupt data (corrupted model), where 10% labels were flipped from 2D spiral dataset (Verma et al. 2019). We present a detailed experimental details for this toy illustration in the Supplementary. Figure 2(a) and 2(b) show the decision boundary of the model for the former and later cases respectively. We observe that, with label corruption, the test accuracy of the model drops by 4.8% from the model trained without any label corruption. The model trained on noisy data in Figure 2(b) has a more complex and rough decision boundary which indicates memorization of the corrupt data points leading to a loss in generalization. When *SAP* is applied to the model trained on corrupt data as seen in Figure 2(c), we observe improved decision boundary smoothness and recover the generalization performance as hypothesized. We summarize the contributions of our work below:

- We propose *SAP*, a novel corrective unlearning algorithm based on Singular Value Decomposition (SVD). *SAP* automates the selection of clean samples using the model loss and updates model weights in a single step, leading to computational efficiency. Notably, we alleviate the challenge of explicitly detecting mislabeled data which is required by most of the unlearning algorithms.
- We empirically validate *SAP*'s performance on synthetic and real-world label noise scenarios across various model architectures and datasets, demonstrating generalization improvements of up to 6% and 2.31%, respectively.

Background

Singular Value Decomposition

A rectangular matrix A in $\mathbb{R}^{d\times n}$ can be decomposed with Singular Value Decomposition (SVD) (Deisenroth, Faisal, and Ong 2020) as

$$A = U\Sigma V^T. \tag{1}$$

Here, $U \in \mathbb{R}^{d\times d}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices and are called left singular matrix and right singular matrix, respectively. $\Sigma \in \mathbb{R}^{d\times n}$ is a diagonal matrix comprising singular values. Each column vector of U is d-dimensional and these vectors form the basis of the column space of A. The amount of variance explained by the i^{th} vector of U, u_i, having singular value of σ_i is proportional to σ_i^2 and hence the fraction variance explained is given by

$$\widetilde{\sigma}_i = \sigma_i^2/(\sum_{j=1}^{d}(\sigma_j^2)). \tag{2}$$

Related Work

Corrective Machine Unlearning (Goel et al. 2024) has emerged as a promising approach to tackle the challenge of mitigating the impact of corrupted data, such as mislabeled samples, on trained models. Unlike traditional machine unlearning (Triantafillou et al. 2024), which often focuses on privacy concerns when removing data, corrective unlearning prioritizes improving model generalization, bypassing the need for adhering to specific privacy requirements (Hayes et al. 2024). In the context of label noise, corrective unlearning aims to improve the model's generalization on unseen data, when the training dataset contains mislabeled samples. Recent work by (Goel et al. 2024) benchmarks several state-of-the-art (SoTA) unlearning algorithms, such as SSD (Foster, Schoepf, and Brintrup 2024), CF-k (Goel et al. 2022) and SCRUB (Kurmanji, Triantafillou, and Triantafillou 2023), within the corrective unlearning framework. Notably, ASSD (Schoepf, Foster, and Brintrup 2024) proposes an extension of SSD which automatically chooses the suitable unlearning hyperparameters to handle the label errors on supply chain delay prediction problem.

Additionally, model generalization can be further improved by using these algorithms in conjunction with traditional approaches such as data filtering (Northcutt, Jiang, and Chuang 2021; Jia et al. 2022; Maini et al. 2022), sample selection (Cheng et al. 2021), label correction (Zheng, Awadallah, and Dumais 2021), regularization (Wei et al. 2021, 2023), robust loss functions (Wang et al. 2019), optimizations (Foret et al. 2021), curriculum learning (Jiang et al. 2018; Zhang and Sabuncu 2018), data augmentation (Zhang et al. 2017; Jiang et al. 2020), and noise transition matrix estimation (Zhu, Wang, and Liu 2022; Cheng et al. 2022). We provide a brief Literature Survey on Label Noise Learning in the supplementary.

Scaled Activation Projection Algorithm

A network trained on the training dataset, $\mathcal{D}_{Tr}$, with corrupted labels is prone to overfit on the incorrect samples within the training data, as seen in Figure 2(b). Such a network is likely to produce spurious intermediate activations to incorporate the incorrect sample-label mapping.

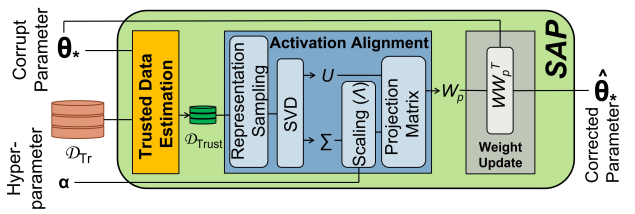


Figure 3: Overview of the proposed *SAP* algorithm. Here, θ_* denotes the model weights which comprises of the layer weights represented by W.

Algorithm 1: SAP algorithm.

Input: θ_* is the parameters of the original model; $\mathcal{D}_{Tr}$ is the training dataset with corrupt labels, and α is a hyperparameter called scaling coefficient.

procedure `SAP`(θ_*, $\mathcal{D}_{Tr}$, α)

1. $\mathcal{D}_{Trust}$ = `get_trusted`(θ_*, $\mathcal{D}_{Tr}$) // Eqn 4
2. R = `representation`(θ_*, $\mathcal{D}_{Trust}$) // Eqn. 5, 6
3. **for** each layer with parameter $W_*^l \in \theta_*$ **do**
4. $\quad U^l, \Sigma^l$ = `SVD`(R^l) // Eqn. 1
5. $\quad \Lambda^l$ = `scale_importance`(Σ^l, α) // Eqn. 7
6. $\quad W_p^l = U^l \Lambda^l (U^l)^T$ // Eqn. 8
7. $\quad \widehat{W}_*^l$ = `update_parameter`(W_p^l, W_*^l) // Eqn. 3
8. **return** $\hat{\theta}_*$ // where $\hat{\theta}_*$ is updated parameters

SAP (Figure 3), outlined in Algorithm 1, aims to align these intermediate activations with representative or clean activations. In the next Subsection, we explain our weight update mechanism which enables us to modify the model weights to remove the influence of corrupt data. We provide our codebase at https://github.com/sangamesh-kodge/SAP.git.

Weight Update

Consider a linear layer with parameters W that generates the output activation a_{out}, given by $a_{out} = a_{in}W^T$. We align the input activations of this layer with the trusted activations by projecting the activations a_{in} onto a strategically constructed projection matrix W_p, also known as the Activation Alignment matrix. This projection would suppress the noisy activation as shown in Figure 4(c). This results in updating the output activation as $\widehat{a}_{out} = (a_{in}W_p)W^T$. We rewrite this equation to absorb the alignment matrix W_p into the layer weights, effectively updating the weights as $\widehat{W} = WW_p^T$, as detailed in Equation 3. This weight update rule explains the `update_parameter` procedure in line 7 of Algorithm 1. In the rest of this Section, our focus is on obtaining the alignment matrix W_p.

$$\widehat{a}_{out} = \underbrace{(a_{in}W_p)}_{\text{Activation Projection}} W^T = a_{in}(\underbrace{WW_p^T}_{\text{Weight Update}})^T = a_{in}\widehat{W}^T \quad (3)$$

Obtaining a high-quality alignment matrix requires access to representative activations for 'trusted' samples or the samples in $\mathcal{D}_{Tr}$ with correct labels. We, therefore, divide the step of obtaining W_p into two parts: (1) approximately estimating a few correctly labeled or 'trusted' samples from $\mathcal{D}_{Tr}$, and (2) obtaining the Activation Alignment Matrix using these samples.

Trusted Data Estimation

We define the trusted dataset, $\mathcal{D}_{Trust} = \{(x_i, y_i)\}_{i=1}^{N_{Trust}}$, as a subset of $\mathcal{D}_{Tr}$ containing a few (N_{Trust}) correctly labeled samples. This work proposes a method to automate the process of identifying these trusted samples. We leverage the cross-entropy loss, denoted by $\mathcal{L}$, of the trained model with parameters θ_*. The rationale is that samples with lower cross-entropy loss are more likely to be correctly labeled. We select N_{Trust} samples with the lowest cross-entropy loss values to from the trusted dataset $\mathcal{D}_{Trust}$. This process can be mathematically expressed by Equation 4 corresponds to `get_trusted` of Algorithm 1. Note, these trusted samples can also be obtained by employing human experts, however, this might be tedious or impractical for larger dataset.

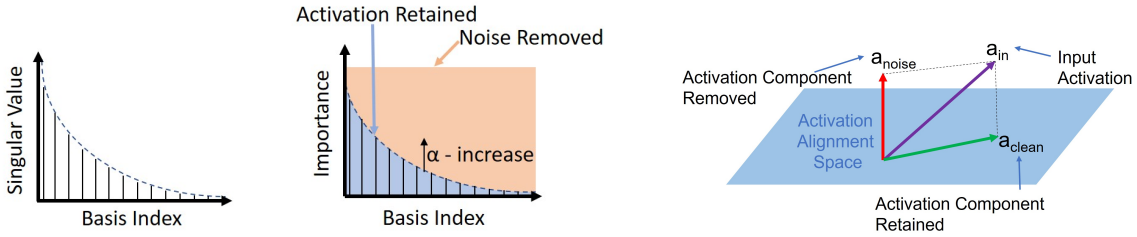
$$\mathcal{D}_{Trust} = \underset{\mathcal{S}}{\operatorname{argmin}} \sum_{(x_i,y_i)\in S} (\mathcal{L}(\theta_*, x_i, y_i)); \quad \text{such that } \mathcal{S} := \{\mathcal{S}_i \subseteq \mathcal{D}_{Tr} \mid |\mathcal{S}_i| = N_{Trust}\} \quad (4)$$

Activation Alignment

To estimate the activation basis for the clean samples, we gather the input activations of linear and convolutional layers, as detailed below.

Representation Sampling: For the l^{th} linear or convolutional layer of the network, we collect the input activation a_{in} and store these activation in a representation matrix R^l. This matrix captures representative information from the trusted samples in the input activations. It is utilized to estimate the trusted activation basis. Next, we elaborate on the details of this representation matrix (Saha, Garg, and Roy 2021) for each type of layer.

1. *Linear Layer* - For a linear layer, the representation matrix, as given by Equation 5, stores the input activations for all the samples in $\mathcal{D}_{Trust}$.
$$R_{linear} = [(a_{in}^i)_{i=1}^{N_{Trust}}] \quad (5)$$
Here, a_{in} is the input activation for the linear layer.
2. *Convolutional Layer* - For the convolutional layer, we express the convolution operation as matrix multiplication to apply the weight update rule proposed in Equation 3. This is achieved through the `unfold` (Chetlur et al. 2014) operation where the convolution operation is represented as matrix multiplication. Let the kernel size of the convolutional layer be $C_{out} \times C_{in} \times k \times k$, where C_{in} represents the number of input channels, C_{out} denotes the number of output channels and k is the kernel size. The unfold operation would flatten all the patches in the input activations on which this kernel operates in a sliding window fashion, which gives us a matrix of size $n_p \times C_{in}kk$. Here, n_p represents the number of patches in the activation a_{in}^i of the i^{th} sample in $\mathcal{D}_{Trust}$. This process allows us to represent convolution as matrix multiplication between the reshaped weights of size $C_{out} \times C_{in}kk$ and the unfolded activation of a_{in}^i of size $n_p \times C_{in}kk$. Figure 8 in supplementary details the conversion of the convolution operation into the matrix multiplication operation. The unfolded activations of all the samples are concatenated and stored in the representation matrix as represented by Equation 6. We use the reshaped convolutions parameters to update the parameters as given in Equation 3.
$$R_{conv} = [(\texttt{unfold}(a_{in}^i)^T)_{i=1}^{N_{Trust}}] \quad (6)$$
The `representation` procedure in line 2 of Algorithm 1 obtains a list of these representation matrices R. The l^{th} element of this list, R^l, is the representation matrix of the l^{th} linear or convolutional layer of the network.



(a) Singular Values (b) Importance Scaling (c) Activation Alignment

Figure 4: (a) Singular value distribution obtained through SVD on the Representative activations. (b) Effect of α in importance scaling Equation 7 and (c) the projection of noisy activation on Activation Alignment Space to obtain clean activations.

SVD on Representations: We perform Singular Value Decomposition (SVD), as given in Equation 1, on the representation matrices R^l for the l^{th} layer, as shown in line 4 of Algorithm 1. The `SVD` procedure provides us with the basis vectors U and the singular values Σ for the trusted samples. The j^{th} vector in U, u_j, is a vector having a unit norm and does not capture the importance of this vector. We scale these vectors in proportion to the variance explained by these vectors.

Importance Scaling: We formulate a diagonal importance matrix Λ with the j^{th} diagonal component λ_j given by Equation 7 (Kodge, Saha, and Roy 2024). Here, $\widetilde{\sigma}_j$ represents the normalized j^{th} singular value, as defined in Equation 2. The effect of such scaling on singular values is shown in Figure 4. The parameter $\alpha \in (0, \infty)$, known as the scaling coefficient, serves as a hyperparameter controlling the scaling of the basis vectors. When α is set to 1, the basis vectors are scaled by the amount of variance they explain. Figure 4(b) shows the effect of changing α on the scaling of the basis vectors. As α increases, the importance score for each basis vector increases and approaches 1 as $\alpha \to \infty$. Conversely, a decrease in α diminishes the importance of the basis vectors, approaching 0 as $\alpha \to 0$. The `scale_parameter` procedure in line 5 of Algorithm 1 corresponds to this importance-based scaling.

$$\lambda_i = \frac{\alpha\widetilde{\sigma}_i}{(\alpha - 1)\widetilde{\sigma}_i + 1} \tag{7}$$

Projection Matrix: The Activation Alignment Matrix W_p is obtained in line 6 of Algorithm 1 using the basis vectors U and the scaled importance matrix Λ by Equation 8. This matrix projects the input activation into the activations spanned by the trusted samples effectively removing the noise due to the corrupted labels from the training dataset.

$$W_p = U\Lambda(U)^T \tag{8}$$

Experiments

Experimental Setup

We evaluate our algorithm on the CIFAR10 and CIFAR100 datasets (Krizhevsky, Hinton et al. 2009) using VGG11 (Simonyan and Zisserman 2015) and ResNet18 (He et al. 2016) models. To simulate various noise scenarios, we introduce synthetic noise to the original training data according to Equations 9, 10, or 11 and subsequently partition the data into a 95% training set ($\mathcal{D}_{Tr}$) and a 5% validation set ($\mathcal{D}_{Val}$). Importantly, we add noise before the train-validation split, mimicking a real-world scenario where noise is present from the outset. Unless otherwise specified, we tune hyperparameters on validation set ($\mathcal{D}_{Val}$) and use the accuracy on the test set ($\mathcal{D}_{Te}$) as the metric to assess generalization performance of the algorithm. In all our tables we report the mean and standard deviation across three randomly chosen seeds in all our experiments. Specific training configurations and hyperparameters are provided in the supplementary material for all the experiments. We explore different type of noise in our evaluations presented in the next subsection.

Types of Synthetic Noise

We evaluate our approach under various noise settings, characterized by the noise transition matrix, denoted by $\mathcal{T}$. The noise transition matrix $\mathcal{T}$ is a square matrix of size $K \times K$, where K is the number of classes, which captures the conditional probability distribution of label corruption. The element at the i^{th} row and j^{th} column, denoted by t_{ij}, describes the probability of a clean label, c_i, being flipped to a noisy label c_j. The degree of noise added is controlled by the parameter η. We study three different noise setting as described below:

Symmetric Noise: In symmetric noise (Ghosh, Kumar, and Sastry 2017), labels are flipped to any other class with equal probability. This creates a uniform distribution of errors across all classes. Equation 9 defines the element of the noise transition matrix $\mathcal{T}$ under a symmetric noise setting.

$$t_{ij} = \begin{cases} \frac{\eta}{(K-1)} & i \neq j \\ 1 - \eta & i = j \end{cases} \tag{9}$$

Asymmetric Noise: In asymmetric noise (Ghosh, Kumar, and Sastry 2017), the probability of a label being flipped to a specific class varies. The noise transition matrix $\mathcal{T}$ in our experiments is defined by Equation 10. The values for off-diagonal elements are sampled from a uniform distribution $\mathcal{U}$ over the range 0 to $\frac{2\eta}{(K-1)}$. The diagonal elements, t_{ii}, are

	Method	Retain samples	Forget samples	VGG11_BN		ResNet18						Average
				Symmetric Noise (Eqn. 9)		Symmetric Noise (Eqn. 9)		Asymmetric Noise (Eqn. 10)		Hierarchical Noise (Eqn. 11)		
				$\eta = 0.1$	$\eta = 0.25$	$\eta = 0.1$	$\eta = 0.25$	$\eta = 0.1$	$\eta = 0.25$	$\eta = 0.1$	$\eta = 0.25$	
CIFAR10	Retrain	-	-	90.18 ± 0.14	89.48 ± 0.04	93.21 ± 0.22	92.45 ± 0.06	93.38 ± 0.07	92.75 ± 0.23	93.47 ± 0.21	93.14 ± 0.23	92.26
	Vanilla	0	0	86.04 ± 0.17	76.68 ± 0.48	88.55 ± 0.16	79.47 ± 0.46	88.53 ± 0.34	79.79 ± 1.53	91.42 ± 0.33	86.42 ± 0.38	84.61
	Finetune	5000	0	85.47 ± 0.13	80.94 ± 0.76	88.28 ± 0.30	85.16 ± 0.12	87.31 ± 0.81	82.82 ± 0.98	91.42 ± 0.33	$\mathbf{88.23 \pm 0.73}$	86.20
	SSD	5000	1000	86.00 ± 0.21	76.77 ± 0.58	88.54 ± 0.17	79.48 ± 0.50	88.52 ± 0.35	80.36 ± 1.45	91.42 ± 0.33	86.39 ± 0.42	84.68
	SCRUB	1000	200	85.88 ± 0.35	78.90 ± 0.25	89.50 ± 0.17	83.77 ± 0.44	89.50 ± 0.22	83.60 ± 0.14	91.65 ± 0.12	88.00 ± 0.58	86.35
	SAP	$0^{\dagger}$	0	$\mathbf{87.25 \pm 0.16}$	$\mathbf{82.27 \pm 0.15}$	$\mathbf{90.12 \pm 0.11}$	$\mathbf{85.49 \pm 0.39}$	$\mathbf{90.03 \pm 0.25}$	$\mathbf{86.32 \pm 0.66}$	$\mathbf{91.87 \pm 0.22}$	87.92 ± 0.69	**87.66**
CIFAR100	Retrain	-	-	65.76 ± 0.23	63.66 ± 0.47	72.43 ± 0.42	71 ± 0.15	73.76 ± 0.46	71.62 ± 0.47	72.73 ± 0.19	71.16 ± 0.39	70.26
	Vanilla	0	0	60.41 ± 0.14	50.64 ± 0.60	65.84 ± 0.33	54.75 ± 0.45	72.98 ± 0.13	61.86 ± 0.60	67.45 ± 0.12	57.82 ± 0.21	61.47
	Finetune	5000	0	60.26 ± 0.11	52.50 ± 0.31	65.97 ± 0.35	57.33 ± 0.40	72.98 ± 0.13	61.29 ± 1.13	67.55 ± 0.15	59.98 ± 1.11	62.23
	SSD	5000	1000	60.38 ± 0.16	50.62 ± 0.60	65.84 ± 0.33	54.77 ± 0.42	72.99 ± 0.11	61.67 ± 0.55	67.43 ± 0.21	57.83 ± 0.21	61.44
	SCRUB	1000	200	60.93 ± 0.09	52.11 ± 0.63	$\mathbf{67.02 \pm 0.29}$	57.36 ± 0.40	$\mathbf{73.12 \pm 0.18}$	63.37 ± 0.69	68.20 ± 0.13	60.24 ± 0.33	62.79
	SAP	$0^{\dagger}$	0	$\mathbf{61.10 \pm 0.23}$	$\mathbf{53.31 \pm 0.78}$	66.82 ± 0.17	$\mathbf{58.74 \pm 0.61}$	72.92 ± 0.30	$\mathbf{63.57 \pm 0.49}$	$\mathbf{68.24 \pm 0.17}$	$\mathbf{60.76 \pm 0.50}$	**63.18**

Table 1: Test Accuracy for symmetric noise (Equation 9) removal from CIFAR10 and CIFAR100 datasets trained on VGG11 and symmetric noise, asymmetric noise (Equation 10) and hierarchical noise (Equation 11) removal ResNet18 architectures with noise strength $\eta = 0.1$ and $\eta = 0.25$. Note, the confusing class groups (or set $\mathcal{C}$) for hierarchical noise are cat-dog and truck-automobile for CIFAR10. For CIFAR100 we use Superclasses as the confusing groups. † - We select 1000 samples with low loss value from $\mathcal{D}_{Tr}$ as retain samples for our method.

computed to ensure that each row of the matrix sums to 1.

$$t_{ij} = \begin{cases} \epsilon \sim \mathcal{U}(0, \frac{2\eta}{(K-1)}) & i \neq j \\ \\ 1 - \sum_{k=1; k \neq i}^{k=K} t_{ik} & i = j \end{cases} \tag{10}$$

Hierarchical Noise: To simulate a practical label noise scenario, we introduce label errors based on class hierarchy (Mukherjee, Garg, and Roy 2024). This approach reflects the intuition that human labelers are more likely to confuse hierarchically closer classes. We define a set of confusing class groups, $\mathcal{C}$, where $g_i \in \mathcal{C}$, indicates all the classes which are hierarchically closer to class i. Equation 11 represent the noise transition matrix for this case.

$$t_{ij} = \begin{cases} \frac{\eta}{|g_i|} & i \neq j; \;\; j \in g_i \\ 0 & i \neq j; \;\; j \notin g_i \\ 1 - \sum_{k=1; k \neq i}^{k=K} t_{ik} & i = j \end{cases} \tag{11}$$

where $g_i \in \mathcal{C}$; and $|g_i|$ number of classes in g_i

Corrective Machine Unlearning Benchmark

We compare our proposed method, ***SAP***, to several unlearning algorithms under different synthetic label noise listed above. As a reference point, we include a **Retrain** model trained with Stochastic Gradient Descent (SGD) on the clean data partition, $\mathcal{D}_{Tr}^{cln}$. All unlearning methods are applied to the **Vanilla** model, which is trained with SGD on the entire corrupt dataset, $\mathcal{D}_{Tr}$. We also include a **Finetune** baseline, which updates all layers of the Vanilla model using SGD on a small retained dataset, leveraging the idea of catastrophic forgetting as explored in CF-k (Goel et al. 2022). For a strong baseline, we implement **SCRUB** (Kurmanji, Triantafillou, and Triantafillou 2023), a state-of-the-art unlearning algorithm, and **SSD** (Foster, Schoepf, and Brintrup 2024), which, like SAP, uses a single update for unlearning.

Many unlearning algorithms require access to both retain (correctly labeled) and forget (misclassified) samples (Goel et al. 2024). However, accurately distinguishing between hard and misclassified samples is challenging (Garg, Ravikumar, and Roy 2024). To address this, we provide all unlearning algorithms with small subsets of retain and forget samples for evaluation. Unlike other methods, SAP does not explicitly use forget samples for model correction and uses the sample loss to identify potential samples for the retain set, bypassing the need for curated subsets. We observed other unlearning algorithms suffered significant performance degradation when using samples with low loss as the retained set.

Results: Our results in Table 1 show that Retrain models closely match expected generalization performance, indicating that a slight reduction in dataset size has minimal impact on model generalization (Guo, Zhao, and Bai 2022). In contrast, the Vanilla model experiences a significant drop in test accuracy, especially at higher noise levels, with a $15 - 20\%$ decline observed at 25% corruption level for both CIFAR10 and CIFAR100. This highlights the detrimental impact of label noise on performance. Finetuning the Vanilla model for 50 epochs on a small subset of $\mathcal{D}_{Tr}^{cln}$ (the clean partition of $\mathcal{D}_{Tr}$) containing 5000 randomly selected samples consistently improves performance, with a notable 6% gain for a ResNet18 model on CIFAR10 with 25% symmetric label corruption. Interestingly, SSD-based unlearning fails to provide a consistent generalization improvement, which might be attributed to the small sample size, as observed in previous studies (Goel et al. 2024; Kodge, Saha, and Roy 2024). *SAP* outperforms all other unlearning methods on both CIFAR-10 and CIFAR-100 datasets, achieving an average improvement of 1.36% and 0.39% compared to the second-best method, respectively. In our experiments, we observed that increasing the number of retain and forget samples to 5000 and 1000, respectively, improved the performance of the SCRUB algorithm, as presented in Table 4 in the supplementary material. However, SCRUB requires extensive hyperparameter tuning (retain batch size, forget batch size, α, γ, number of minimization steps, and number of maximization steps) to counteract the instability of gradient ascent, leading to approximately 675 hyperparameter search iterations compared to our method's 16. Increasing the number of samples further exacerbates the computational burden for hyperparameter search. In the next subsec-

	Method	Baseline	SAP	Improvements
CIFAR10	Vanilla	79.47 ± 0.46	85.46 ± 0.41	**5.99**
	Logit Clip	82.91 ± 0.32	85.99 ± 0.67	**3.08**
	MixUp	83.12 ± 0.44	86.45 ± 0.52	**3.33**
	SAM	83.29 ± 0.28	87.29 ± 0.08	**4.0**
	MentorMix	89.64 ± 0.32	90.51 ± 0.17	**0.87**
Average		83.69	87.14	**3.45**
CIFAR100	Vanilla	54.75 ± 0.45	58.69 ± 0.68	**3.94**
	Logit Clip	56.41 ± 0.43	59.90 ± 0.84	**3.49**
	SAM	56.49 ± 0.93	59.34 ± 0.76	**2.85**
	MixUp	58.25 ± 0.65	62.32 ± 0.91	**4.07**
	MentorMix	68.53 ± 0.35	68.98 ± 0.45	**0.45**
Average		58.89	61.85	**2.98**

Table 2: Generalization benefits of applying our algorithm in synergy with noise-robust training approaches for symmetric noise (Equation 9) removal from CIFAR10 and CIFAR100 datasets trained on ResNet18 architectures with noise strength $\eta = 0.25$. We report the mean and standard deviation across three randomly chosen seeds.

tion, we demonstrate how SAP can enhance the generalization performance of label noise robust learning algorithms.

Noise-Robust Learning Benchmarks

We compare our method to several label noise robust learning algorithms: **Logit Clip** (Wei et al. 2023), Sharpness Aware Minimization (**SAM**) (Foret et al. 2021), **MixUp** (Zhang et al. 2017), and **MentorMix** (Jiang et al. 2020), known for their inherent tolerance to mislabeled data. Table 2 presents the performance of various algorithms on CIFAR-10 and CIFAR-100 datasets using the ResNet18 model under symmetric label noise with $\eta = 0.25$. Notably, SAP improves generalization performance by 3.45% and 2.98% on average for CIFAR-10 and CIFAR-100, respectively.

Real World Noisy Benchmark

To demonstrate *SAP*'s effectiveness in real-world noise scenarios, we present experiments on Mini-WebVision (Jiang et al. 2020) and Clothing1M (Xiao et al. 2015) datasets. Mini-WebVision is trained on the InceptionResNetV2 (IRV2 in Table 3) (Szegedy et al. 2017) architecture, while Clothing1M is trained on ResNet50 (He et al. 2016) and ViT_B_16. Comprehensive training and hyperparameter details are provided in supplementary material.

As shown in Table 3, applying *SAP* consistently yields generalization improvements for a noisy dataset. These results demonstrate an average improvement of 1.8%, highlighting *SAP*'s ability to scale to large datasets (Clothing1M) and SoTA transformer based models (ViT_B_16).

Analyses

Increasing Trusted Sample Size yields diminishing returns: We study how the number of trusted samples impacts *SAP*'s performance. Figure 5 shows how test accuracy varies with the number of trusted samples. We observe that increasing the sample size improves accuracy up to 1000 samples, after which there are no significant gains. Since representation calculations and SVD are performed on the trusted set, the sample size directly affects compute requirements. Therefore, we use 1000 samples in our experiments to balance performance and computational efficiency.

Dataset	Architecture	Vanilla	SAP	Improvement
Mini-WebVision	IRV2	63.81 ± 0.38	64.73 ± 0.53	**0.92**
Clothing1M	ResNet50	67.48 ± 0.64	69.64 ± 0.57	**2.16**
Clothing1M	ViT_B_16	69.12 ± 0.45	71.43 ± 0.60	**2.31**
Average		66.80	68.60	**1.80**

Table 3: Generalization benefits of applying our algorithm on real-world noisy dataset like Mini-WebVision (Jiang et al. 2020), and Clothing1M (Xiao et al. 2015) for Vanilla trained InceptionResNetv2, called IRV2 in Table (Szegedy et al. 2017) and ViT_B_16 (Dosovitskiy et al. 2021).

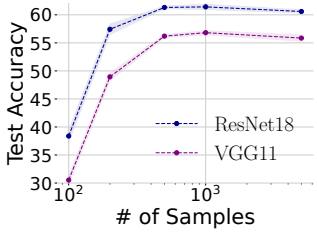


Figure 5: Effect of trusted sample size for CIFAR100 dataset with $\eta = 25\%$.

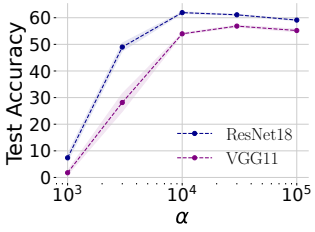


Figure 6: Effect of α for model trained on CIFAR100 dataset with $\eta = 25\%$.

Optimal α is key to maximizing performance gains : Figure 6 demonstrates how the scaling factor (α) influences test accuracy for models trained on datasets with 25% label corruption. We observe that increasing α generally improves test accuracy up to a certain point, after which accuracy gradually declines. Importantly, we find that $\alpha = 30000$ consistently yields strong results for synthetic noise across different datasets, models, and corruption levels.

Discussion and Conclusion

This paper introduces *SAP*, a corrective machine unlearning algorithm that is competitive with the SoTA SCRUB without requiring explicit access to clean and mislabeled data subsets. This eliminates the need for laborious human curation of training data. *SAP* performs a single model update using Singular Value Decomposition (SVD) on a small sample set, ensuring computational efficiency. Furthermore, with only one hyperparameter, α, and reusable SVD computations across different α values, *SAP* offers simplicity and efficiency, unlike gradient-ascent based algorithms which often require extensive hyperparameter tuning, sometimes exceeding the computational cost of retraining the model. In summary, SAP is a sample and compute-efficient unlearning technique to address the challenges posed by noisy data.

Acknowledgements

This work was supported in part by the Center for the Co-Design of Cognitive Systems (COCOSYS), a DARPA-sponsored JUMP center, the Semiconductor Research Corporation (SRC), the National Science Foundation, and Collins Aerospace.

References

Biggio, B.; Nelson, B.; and Laskov, P. 2012. Poisoning Attacks against Support Vector Machines. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, 1467–1474. Madison, WI, USA: Omnipress. ISBN 9781450312851.

Chen, X.; Liu, C.; Li, B.; Lu, K.; and Song, D. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*.

Chen, Y.; Shen, C.; Shen, Y.; Wang, C.; and Zhang, Y. 2022. Amplifying membership exposure via data poisoning. *Advances in Neural Information Processing Systems*, 35: 29830–29844.

Cheng, D.; Liu, T.; Ning, Y.; Wang, N.; Han, B.; Niu, G.; Gao, X.; and Sugiyama, M. 2022. Instance-Dependent Label-Noise Learning With Manifold-Regularized Transition Matrix Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 16630–16639.

Cheng, H.; Zhu, Z.; Li, X.; Gong, Y.; Sun, X.; and Liu, Y. 2021. Learning with Instance-Dependent Label Noise: A Sample Sieve Approach. arXiv:2010.02347.

Chetlur, S.; Woolley, C.; Vandermersch, P.; Cohen, J.; Tran, J.; Catanzaro, B.; and Shelhamer, E. 2014. cuDNN: Efficient Primitives for Deep Learning. arXiv:1410.0759.

Deisenroth, M. P.; Faisal, A. A.; and Ong, C. S. 2020. *Mathematics for Machine Learning*. Cambridge University Press.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv:2010.11929.

Foret, P.; Kleiner, A.; Mobahi, H.; and Neyshabur, B. 2021. Sharpness-aware Minimization for Efficiently Improving Generalization. In *International Conference on Learning Representations*.

Foster, J.; Schoepf, S.; and Brintrup, A. 2024. Fast machine unlearning without retraining through selective synaptic dampening. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38-11, 12043–12051.

Garg, I.; Ravikumar, D.; and Roy, K. 2024. Memorization Through the Lens of Curvature of Loss Function Around Samples. In *Forty-first International Conference on Machine Learning*.

Ghosh, A.; Kumar, H.; and Sastry, P. S. 2017. Robust Loss Functions under Label Noise for Deep Neural Networks. arXiv:1712.09482.

Goel, S.; Prabhu, A.; Sanyal, A.; Lim, S.-N.; Torr, P.; and Kumaraguru, P. 2022. Towards adversarial evaluations for inexact machine unlearning. *arXiv preprint arXiv:2201.06640*.

Goel, S.; Prabhu, A.; Torr, P.; Kumaraguru, P.; and Sanyal, A. 2024. Corrective Machine Unlearning. arXiv:2402.14015.

Guo, C.; Zhao, B.; and Bai, Y. 2022. Deepcore: A comprehensive library for coreset selection in deep learning. In *International Conference on Database and Expert Systems Applications*, 181–195. Springer.

Hayes, J.; Shumailov, I.; Triantafillou, E.; Khalifa, A.; and Papernot, N. 2024. Inexact Unlearning Needs More Careful Evaluations to Avoid a False Sense of Privacy. arXiv:2403.01218.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Jagielski, M.; Oprea, A.; Biggio, B.; Liu, C.; Nita-Rotaru, C.; and Li, B. 2018. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE symposium on security and privacy (SP)*, 19–35. IEEE.

Jia, Q.; Li, X.; Yu, L.; Bian, J.; Zhao, P.; Li, S.; Xiong, H.; and Dou, D. 2022. Learning from Training Dynamics: Identifying Mislabeled Data Beyond Manually Designed Features. arXiv:2212.09321.

Jiang, L.; Huang, D.; Liu, M.; and Yang, W. 2020. Beyond synthetic noise: Deep learning on controlled noisy labels. In *International conference on machine learning*, 4804–4815. PMLR.

Jiang, L.; Zhou, Z.; Leung, T.; Li, L.-J.; and Fei-Fei, L. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International conference on machine learning*, 2304–2313. PMLR.

Kodge, S.; Saha, G.; and Roy, K. 2024. Deep Unlearning: Fast and Efficient Gradient-free Class Forgetting. *Transactions on Machine Learning Research*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. *Technical Report*.

Kurmanji, M.; Triantafillou, P.; and Triantafillou, E. 2023. Towards Unbounded Machine Unlearning. *arXiv preprint arXiv:2302.09880*.

Maini, P.; Garg, S.; Lipton, Z.; and Kolter, J. Z. 2022. Characterizing datapoints via second-split forgetting. *Advances in Neural Information Processing Systems*, 35: 30044–30057.

Mukherjee, A.; Garg, I.; and Roy, K. 2024. Encoding Hierarchical Information in Neural Networks Helps in Subpopulation Shift. *IEEE Transactions on Artificial Intelligence*, 5(2): 827–838.

Northcutt, C.; Jiang, L.; and Chuang, I. 2021. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70: 1373–1411.

Northcutt, C. G.; Athalye, A.; and Mueller, J. 2021. Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

Pangakis, N.; and Wolken, S. 2024. Knowledge Distillation in Automated Annotation: Supervised Text Classification with LLM-Generated Training Labels. arXiv:2406.17633.

Saha, G.; Garg, I.; and Roy, K. 2021. Gradient Projection Memory for Continual Learning. In *International Conference on Learning Representations*.

Sambasivan, N.; Kapania, S.; Highfill, H.; Akrong, D.; Paritosh, P.; and Aroyo, L. M. 2021. "Everyone wants to do the model work, not the data work": Data Cascades in High-Stakes AI. In *proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 1–15.

Schoepf, S.; Foster, J.; and Brintrup, A. 2024. Parameter-tuning-free data entry error unlearning with adaptive selective synaptic dampening. arXiv:2402.10098.

Schwarzschild, A.; Goldblum, M.; Gupta, A.; Dickerson, J. P.; and Goldstein, T. 2021. Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks. In *International Conference on Machine Learning*, 9389–9398. PMLR.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556.

Steinhardt, J.; Koh, P. W. W.; and Liang, P. S. 2017. Certified defenses for data poisoning attacks. *Advances in neural information processing systems*, 30.

Szegedy, C.; Ioffe, S.; Vanhoucke, V.; and Alemi, A. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31-1.

Triantafillou, E.; Kairouz, P.; Pedregosa, F.; Hayes, J.; Kurmanji, M.; Zhao, K.; Dumoulin, V.; Junior, J. J.; Mitliagkas, I.; Wan, J.; Hosoya, L. S.; Escalera, S.; Dziugaite, G. K.; Triantafillou, P.; and Guyon, I. 2024. Are we making progress in unlearning? Findings from the first NeurIPS unlearning competition. arXiv:2406.09073.

Verma, V.; Lamb, A.; Beckham, C.; Najafi, A.; Mitliagkas, I.; Lopez-Paz, D.; and Bengio, Y. 2019. Manifold mixup: Better representations by interpolating hidden states. In *International conference on machine learning*, 6438–6447. PMLR.

Wang, X.; Kim, H.; Rahman, S.; Mitra, K.; and Miao, Z. 2024. Human-LLM collaborative annotation through effective verification of LLM labels. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 1–21.

Wang, Y.; Ma, X.; Chen, Z.; Luo, Y.; Yi, J.; and Bailey, J. 2019. Symmetric Cross Entropy for Robust Learning with Noisy Labels. arXiv:1908.06112.

Wei, H.; Zhuang, H.; Xie, R.; Feng, L.; Niu, G.; An, B.; and Li, Y. 2023. Mitigating Memorization of Noisy Labels by Clipping the Model Prediction. arXiv:2212.04055.

Wei, J.; Liu, H.; Liu, T.; Niu, G.; Sugiyama, M.; and Liu, Y. 2021. To smooth or not? when label smoothing meets noisy labels. *arXiv preprint arXiv:2106.04149*.

Xiao, T.; Xia, T.; Yang, Y.; Huang, C.; and Wang, X. 2015. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2691–2699.

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Zhang, Z.; and Sabuncu, M. 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, 31.

Zheng, G.; Awadallah, A. H.; and Dumais, S. 2021. Meta label correction for noisy label learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35-12, 11053–11061.

Zhu, Z.; Wang, J.; and Liu, Y. 2022. Beyond Images: Label Noise Transition Matrix Estimation for Tasks with Lower-Quality Features. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvari, C.; Niu, G.; and Sabato, S., eds., *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 27633–27653. PMLR.