

516 **A Implementation details**

517 **Self-Paced Learning**

518 The Self-Paced Learning (SPL) [42] method sets a threshold λ value for the loss and all examples with
519 loss larger than λ are skipped, since they are treated as hard to learn (because they are possibly noisy).
520 After each training epoch, the threshold is increased by some constant multiplier. For simplification,
521 we adjusted SPL in the following manner.

522 We set a parameter τ_{SPL} , which controls the percentage of samples with the highest loss within a
523 batch that are excluded. The value of τ_{SPL} should be equal to the noise level present in the training
524 dataset. As such, at each step, we exclude a set percentage of potentially noisy examples, thus
525 reducing the impact of label noise on the training process. We keep the value of τ_{SPL} constant
526 throughout the training.

527 **Provably Robust Learning**

528 The Provably Robust Learning (PRL) [43] algorithm works in a similar manner to SPL. We follow
529 the authors by introducing the τ_{PRL} parameter, which controls the percentage of samples excluded
530 from each training batch on the basis of their gradient norm. Specifically, $\tau_{PRL}\%$ of samples with
531 highest gradient norm are omitted, while the rest is used to update model parameters. The value of
532 τ_{PRL} should be equal to the noise level in the training dataset.

533 **Clipped Cross-Entropy**

534 Since our implementation of SPL doesn't have a hard loss threshold, we introduce a simple Clipped
535 Cross-Entropy (CCE) baseline to check the effectiveness of such an approach. The CCE method
536 checks if the loss is greater than some threshold λ_{CCE} . If so, the loss is clipped to that value.
537 Otherwise, it is left unchanged. Thus, we always use all training samples, but the impact of label
538 noise is alleviated by clipping the loss.

539 **Early Learning Regularization**

540 For Early Learning Regularization (ELR) [41], we followed the implementation published by the
541 authors. We compute the softmax probabilities for each sample in a batch and clamp them, then
542 compute the soft targets via temporal ensembling and use these targets in the loss function calculation.
543 Our implementation includes one step not present in the publication text - softmax probability
544 clamping in range $[\epsilon, 1 - \epsilon]$, where ϵ is a clamp margin parameter. Aside from this, we use the β
545 target momentum and λ_{ELR} regularization parameters just as they were presented by the authors.

546 **Generalized Jensen-Shannon Divergence Loss**

547 The Generalized Jensen-Shannon Divergence (GJSD) [44] loss function is a generalization of Cross-
548 Entropy (CE) and Mean Absolute Error (MAE) losses. We follow the implementation provided
549 by the authors, in which we use the M parameter to set the number of averaged distributions and
550 the π parameter to adjust the weight between CE and MAE. While the authors share separate
551 implementations for GJSD with and without consistency regularization, we implement it as a toggle
552 to make the code more uniform. Since consistency regularization requires data augmentation and
553 the GJSD authors described only augmentations for the image domain, we implemented several
554 textual augmentations of our own: random token dropping, consecutive token dropping, random
555 token swapping. However, in our experiments, we have kept consistency regularization turned off
556 due to its detrimental effect on model convergence and test accuracy.

557 **Co-teaching**

558 While the methods described above modified the loss function in various ways, Co-teaching (CT) [45]
 559 works in a different manner. It requires optimizing two sets of model parameters at the same time.
 560 As such, following the algorithm described by the authors, we implemented a custom model class,
 561 which manages the update of these two sets of weights and the exchange of low-loss examples at
 562 each optimization step. We keep the parameters k and τ_{CT} , to control the starting epoch for CT
 563 and the noise level (i.e. the percentage of low-loss examples that are exchanged between networks),
 564 respectively.

565 **Co-teaching+**

566 For Co-teaching+ (CT+) [46], we again adhere to the algorithm described by the authors. We use the
 567 same implementation framework as for CT, adjusting only the sample selection mechanism to look
 568 within examples for which there is disagreement between the two networks. Following the advice in
 569 the publication text, we use the *recommended* update strategy for the fraction of instances to select,
 570 which is calculated based on the epoch number, as well as parameters k and τ_{CT+} .

571 **Mixup**

572 The Mixup (MU) [47] technique keeps the loss function (CE) and the hyperparameters of the baseline
 573 model unchanged, only augmenting the training data during the training procedure. We use in-batch
 574 augmentation, fixed per-batch mixing magnitude sampled from $Beta(\alpha, \alpha)$ (where α is provided
 575 as input), and the mixed pairs are sampled without replacement from that distribution. Since we
 576 cannot mix input in the same way as for images, we implemented in-batch augmentation for logits.
 577 In addition, we also keep the r_{MU} ratio parameter, to adjust the percentage of the batch size which is
 578 taken for augmentation in MU. Note: our hyperparameter tuning procedure resulted in setting both α
 579 and r_{MU} to low values (Tab. S1), contrary to what is recommended by the authors.

Method	Hyperparameters	Selected values
SPL	τ_{SPL}	equal to noise level
PRL	τ_{PRL}	equal to noise level
ELR	$\epsilon, \beta, \lambda_{ELR}$	1e-5, 0.6, 2
CCE	λ_{CCE}	9.5
MU	α, r_{MU}	0.1, 0.1
GJSD	M, π	2, 5e-3
CT	k, τ_{CT}	8, equal to noise level
CT+	k, τ_{CT+}	8, equal to noise level

Table S1: Hyperparameter values for all benchmarked methods, selected through a tuning procedure.

580 **B Results of experiments with higher noise level**

581 For completeness, we evaluate the accuracy for all methods on datasets with 40% synthetic noise
 582 (Tab. S2). The best methods for this noise level are the same as for the case of 15% noise: for
 583 symmetric noise, GJSD is the best method, while for asymmetric noise types it is ELR. However, it is
 584 clear that some methods show more noticeable effect when compared to the baseline for the 40%
 585 noise level than for the 15%. While MU and CCU stay close to the baseline results for all noise types
 586 and SPL underperforms in all cases, CT consistently gives an improvement over the baseline and
 587 CT+ decreases the result for the symmetric noise, but is better than the baseline for asymmetric noise
 588 types.

589 We also plot memorized_{val}^{noisy} for those datasets (Fig. S1). For symmetric and pair-flip noise types the
 590 memorization for all methods is very low. For nested-flip and matrix-flip it is a bit higher, indicating
 591 that these two noise types are more challenging, and thus induce more memorization in the model.

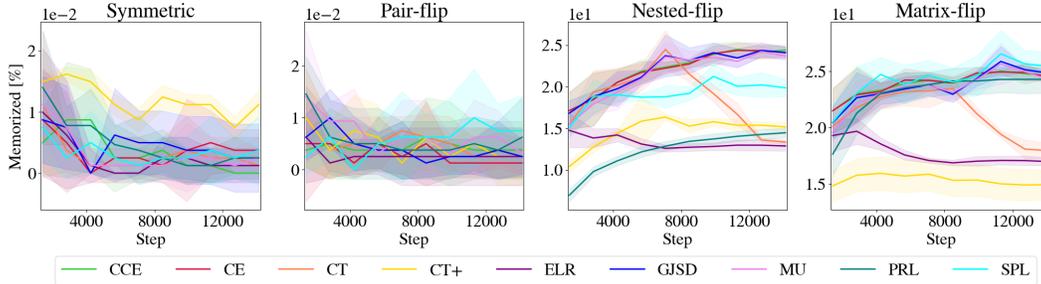


Figure S1: Value of memorized_{val} for different noise types, measured at each training step for all evaluated methods. In all cases the noise level was set at 40%.

	Clean set	Symmetric	Pair-flip	Nested-flip	Matrix-flip
CE	74.85 ± 0.15	67.29 ± 0.12	55.18 ± 0.26	52.87 ± 0.19	54.04 ± 0.23
ELR	74.81 ± 0.11	67.23 ± 0.18	66.12 ± 0.15	62.27 ± 0.19	61.72 ± 0.23
MU	74.73 ± 0.09	67.14 ± 0.14	55.28 ± 0.26	52.38 ± 0.24	54.26 ± 0.25
CCE	74.80 ± 0.09	68.92 ± 0.14	55.13 ± 0.28	52.07 ± 0.58	54.02 ± 0.18
CT	*74.85 ± 0.15	68.60 ± 0.14	60.49 ± 0.24	58.48 ± 0.28	57.69 ± 0.47
CT+	*74.85 ± 0.15	↓64.67 ± 0.32	59.03 ± 0.42	56.16 ± 0.42	57.06 ± 0.29
PRL	*74.85 ± 0.15	↓65.01 ± 0.30	62.22 ± 0.45	56.39 ± 0.40	51.59 ± 0.97
SPL	*74.85 ± 0.15	65.27 ± 0.35	↓44.92 ± 1.52	↓42.29 ± 1.06	↓40.89 ± 0.70
GJSD	74.63 ± 0.10	69.80 ± 0.12	54.78 ± 0.30	51.92 ± 0.46	53.84 ± 0.10

Table S2: Accuracy of the evaluated methods on the clean dataset compared to various noisy datasets with 40% noise level. The noisy datasets include symmetric synthetic noise and asymmetric synthetic noise types: pair-flip, nested-flip, and matrix-flip. * marks cases equivalent to the baseline CE. ↓ marks results significantly worse than the baseline CE. Best results for each noise type are bolded.