Mitigating Label Noise through Data Ambiguation: Supplementary Material

Julian Lienen, Eyke Hüllermeier

A Evaluation Details

A.1 Dataset and Data Preprocessing

As datasets, we consider a wide range of commonly studied image classification datasets. While CIFAR-10 and -100 [Krizhevsky and Hinton, 2009], MNIST [LeCun et al., 1998], FashionMNIST [Xiao et al., 2017] and SVHN [Netzer et al., 2011] (the latter three are used in an additional study in Sec. C.3) are widely known and well-studied, WebVision [Li et al., 2017] and Clothing1M [Xiao et al., 2015] comprise real-world noise from human annotations. For WebVision (version 1.0), we consider the Google image subset of ca. 66,000 images from the top-50 classes resized to 256x256. Clothing1M consists of 1 million training images with noisy and 10,000 test images with clean labels. Here, we also resize the images to 256x256.

For training, we apply data augmentation to the training images as commonly being done in image classification. To do so, we randomly crop images of size 32 (CIFAR-10(0)(N), SVHN), 227 (WebVision) or 224 (Clothing1M), followed by horizontally flipping the image with probability of 0.5. In case of MNIST and FashionMNIST, we keep the images unchanged, but resize them to size 32. The reported dimensions are used to preserve comparability with previous results (e.g., as reported in [Liu et al., 2020]).

A.2 Synethtic Noise Model

While CIFAR-10N and -100N [Wei et al., 2022], WebVision and Clothing1M provide real-world noise in the standard annotations, we modeled additional label corruptions for the rest of the datasets in a synthetic manner. Thereby, we distinguish symmetric and asymmetric noise: For the former, each class is treated equally, whereas the asymmetric noise applies individual corruptions to each class.

In case of the symmetric noise, we flip a parameterized fraction $\rho \in [0, 1]$ of instances by uniformly sampling the label from all classes, i.e., we sample a corrupted label via $y \sim \text{Unif}(\mathcal{Y})$ from the uniform distribution $\text{Unif}(\mathcal{Y})$ over the class space $\mathcal{Y}$ with probability ρ. Hence, also correct label can be chosen again. In our studies, we considered values $\rho \in \{0.25, 0.5, 0.75\}$.

In case of asymmetric noise, we use the same setup as suggested in [Patrini et al., 2017]. For CIFAR-10, we flip "truck" to "automobile", "bird" to "airplane", "deer" to "horse", as well as interchange "cat" and "dog". For CIFAR-100, the classes are grouped into 20 clusters with 5 classes each, whereby labels are flipped within these clusters in a round-robin fashion. In both cases, we apply asymmetric random flips with probability $\rho = 0.4$.

A.3 Hyperparameters

	CIFAR-10(N)	CIFAR-100(N)	WebVision	Clothing1M
Model	ResNet34	ResNet34	ResNet50 (pretrained)	ResNet50 (pretrained)
Batch size	128	128	64	32
Learning rate (LR)	0.02	0.02	0.02	0.002
LR decay	cosine	cosine	cosine	cosine
Weight decay	5×10^{-4}	5×10^{-4}	5×10^{-4}	1×10^{-3}
Epochs	120	150	100	15

Table 1: Hyperparameters fixed for all baselines and our method per dataset. ResNet50 models proceed from model weights pretrained on ImageNet.

In our paper, we follow common evaluation settings as in recent label noise robustness approaches [Zhou et al., 2021, Liu et al., 2022]. To this end, we keep the optimizer hyperparameters as being used in previous reports, allowing for a fair comparison. Table 1 gives an overview over used hyperparameters. In all case, we used SGD with momentum of 0.9 as optimizer.

For the individual losses, we used the optimal parameters being reported in the respective works. This is reasonable as these parameters were optimized in a similar manner on the same data, such that we can re-use these optimization results. For our losses, we fixed the parameters as specified above and optimized the β parameters using random search with 20 iterations employing a 5-fold cross validation on the (partially noisy) training data. To ensure this optimization does not give our method an unfair advantage, we performed the same operation to samples of the baseline experiments, where we noticed that the optimization preferred similar hyperparameter combinations as also reported in the related work.

For results presented in the main paper, we used a cosine decaying strategy as shown in Eq. (5) with $\beta_0 = 0.75$ and $\beta_1 = 0.6$ on all datasets except CIFAR-100. On the latter, we used a fixed schedule with $\beta = 0.5$. In Section C.2, we further present an ablation study showing results for different β schedules. In all cases, we set $\alpha = 0.05$.

A.4 Technical Infrastructure

The code of our empirical evaluation is made publicly available upon publication. All experiments are implemented using PyTorch[1] as deep learning framework

[1] `https://pytorch.org/`, BSD-3 license

using Nvidia CUDA[2] as acceleration backend. Image manipulations as being used for data agumentation are provided by Pillow[3]. To perform the experiments, we used several Nvidia A100 and RTX 2080 Ti accelerators in a modern high performance computing environment.

B Experimental Result Completion

Due to space limitations, some results of the experiments in the main paper have been shifted to this section. The following subsections complete the presentation of the results.

B.1 Synthetic Noise Results

Loss	Add. Param.	CIFAR-10				CIFAR-100			
		Sym.			Asym.	Sym.			Asym.
		25 %	50 %	75 %	40 %	25 %	50 %	75 %	40 %
CE	✗	79.05 ±0.67	55.03 ±1.02	30.03 ±0.74	77.90 ±0.31	58.27 ±0.36	37.16 ±0.46	13.66 ±0.45	62.83 ±0.25
LS ($\alpha = 0.1$)	✗	76.66 ±0.69	53.95 ±1.47	29.03 ±1.21	78.07 ±0.69	59.75 ±0.24	37.61 ±0.61	13.53 ±0.51	63.76 ±0.51
LS ($\alpha = 0.25$)	✗	77.48 ±0.32	53.08 ±1.95	28.29 ±0.65	77.35 ±0.76	59.84 ±0.57	39.80 ±0.38	14.18 ±0.44	63.33 ±0.25
LR ($\alpha = 0.1$)	✗	80.53 ±0.39	57.55 ±0.95	29.83 ±0.87	77.83 ±0.37	57.52 ±0.58	36.77 ±0.54	13.23 ±0.14	62.46 ±0.15
LR ($\alpha = 0.25$)	✗	80.43 ±0.09	60.18 ±1.01	31.36 ±0.91	78.35 ±0.72	57.67 ±0.11	37.15 ±0.14	13.41 ±0.24	62.85 ±0.53
GCE	✗	90.82 ±0.10	83.36 ±0.65	54.34 ±0.37	77.37 ±0.94	68.06 ±0.31	58.66 ±0.28	**26.85** ±1.28	61.08 ±0.51
NCE	✗	79.05 ±0.12	63.94 ±1.74	38.23 ±2.63	76.84 ±0.41	19.32 ±0.81	11.09 ±1.03	6.12 ±7.57	24.67 ±0.71
NCE+AGCE	✗	87.57 ±0.10	83.05 ±0.81	51.16 ±6.44	69.75 ±2.33	64.15 ±0.23	39.64 ±1.66	7.67 ±1.25	53.87 ±1.60
NCE+AUL	✗	88.89 ±0.29	84.18 ±0.42	**65.98** ±1.56	80.87 ±0.34	69.76 ±0.31	57.41 ±0.41	17.72 ±1.27	61.33 ±0.55
CORES	✗	88.60 ±0.28	82.44 ±0.29	47.32 ±17.03	82.22 ±0.55	60.36 ±0.67	46.01 ±0.44	18.23 ±0.28	65.06 ±0.41
RDA (ours)	✗	**91.48** ±0.22	**86.47** ±0.42	48.11 ±15.41	**85.95** ±0.40	**70.03** ±0.32	**59.83** ±1.15	26.75 ±8.83	**69.62** ±0.54
ELR	✓	92.45 ±0.08	88.39 ±0.36	72.58 ±1.63	82.18 ±0.42	73.66 ±1.87	48.72 ±26.93	38.35 ±10.26	74.19 ±0.23
SOP	✓	92.58 ±0.08	89.21 ±0.33	76.16 ±4.88	84.61 ±0.97	72.04 ±0.67	64.28 ±1.44	40.59 ±1.62	64.27 ±0.34

Table 2: Test accuracies and standard deviations on the test split for models trained on CIFAR-10(0) with synthetic noise (symmetric or asymmetric). The results are averaged over runs with different seeds, bold entries mark the best method without any additional model parameters. Underlined results indicate the best method overall.

[2] `https://developer.nvidia.com/cuda-toolkit`
[3] `https://python-pillow.org/`, HPND License

B.2 Complete Real-World Noise Results

Loss	Add. Param.	CIFAR-10N						CIFAR-100N	
		Clean	Random 1	Random 2	Random 3	Aggregate	Worst	Clean	Noisy
CE	✗	**94.12** ±0.17	82.96 ±0.23	83.16 ±0.52	83.49 ±0.34	88.74 ±0.13	64.93 ±0.79	75.29 ±0.15	52.88 ±0.14
LS ($\alpha = 0.1$)	✗	93.92 ±0.03	82.76 ±0.47	82.10 ±0.21	82.12 ±0.37	88.63 ±0.11	63.10 ±0.38	75.71 ±0.19	53.48 ±0.45
LS ($\alpha = 0.25$)	✗	93.71 ±0.40	82.95 ±1.57	83.86 ±2.05	82.61 ±0.25	87.03 ±2.29	66.14 ±6.89	75.69 ±0.17	53.98 ±0.27
LR ($\alpha = 0.1$)	✗	93.77 ±0.06	83.00 ±0.36	82.64 ±0.31	82.82 ±0.21	88.41 ±0.29	66.62 ±0.33	74.79 ±0.16	52.01 ±0.04
LR ($\alpha = 0.25$)	✗	93.63 ±0.06	82.14 ±0.49	81.87 ±0.34	82.46 ±0.11	88.07 ±0.45	66.44 ±0.14	74.51 ±0.17	52.22 ±0.29
GCE	✗	93.22 ±0.08	88.85 ±0.19	88.96 ±0.32	88.73 ±0.11	90.85 ±0.32	77.24 ±0.47	72.29 ±0.19	55.43 ±0.47
NCE	✗	87.67 ±0.25	81.88 ±0.27	81.02 ±0.32	81.48 ±0.13	84.62 ±0.49	69.40 ±0.10	32.31 ±0.31	21.12 ±0.67
NCE+AGCE	✗	92.56 ±0.07	89.48 ±0.28	88.95 ±0.10	89.25 ±0.29	90.65 ±0.44	81.27 ±0.44	72.00 ±0.09	51.42 ±0.65
NCE+AUL	✗	93.09 ±0.10	89.42 ±0.22	89.36 ±0.15	88.94 ±0.55	90.92 ±0.19	81.28 ±0.47	74.18 ±0.21	56.58 ±0.41
CORES	✗	93.09 ±0.08	86.09 ±0.57	86.48 ±0.27	86.02 ±0.22	89.23 ±0.10	76.80 ±0.96	73.70 ±0.17	53.04 ±0.29
RDA (ours)	✗	94.09 ±0.19	**90.43** ±0.03	**90.09** ±0.29	**90.40** ±0.01	**91.71** ±0.38	**82.91** ±0.83	**76.21** ±0.64	**59.22** ±0.26
ELR	✓	94.21 ±0.11	91.35 ±0.29	91.46 ±0.29	91.39 ±0.03	92.68 ±0.03	84.82 ±0.42	76.66 ±0.11	62.80 ±0.27
SOP	✓	92.84 ±0.20	89.16 ±0.40	89.02 ±0.33	88.99 ±0.31	90.54 ±0.16	80.65 ±0.13	76.12 ±0.32	59.32 ±0.41

Table 3: Test accuracies and standard deviations on the test split for models trained on CIFAR-10(0)N without any noise (clean) and real-world noise. The results are averaged over runs with different seeds, bold entries mark the best method without any additional model parameters. Underlined results indicate the best method overall.

C Further Experiments

C.1 Comparison to Full Ambiguation

In this section, we compare our method RDA to a full ambiguation adaptation: For prediction $\widehat{p}$ exceeding β for a class different to the training label, we ambiguate the target information by a completely agnostic credal sets. Hence, this instance is completely ignored in the loss optimization. Table 4 shows the respective results, where significant improvements over the complete ambiguation can be observed for our method. Hence, incorporating credal sets with two fully plausible classes appears reasonable from a loss minimization context.

CIFAR-10N	RDA	Complete Ambig.
Clean	**94.09** ±0.19	93.77 ±0.37
Random1	**90.43** ±0.03	87.04 ±0.18
Random2	**90.09** ±0.29	89.36 ±0.16
Random3	**90.40** ±0.01	89.04 ±0.21
Aggregate	**91.71** ±0.38	88.73 ±0.64
Worst	**82.91** ±0.83	76.57 ±0.91

Table 4: Averaged test accuracies and standard deviations computed over three runs with different seeds. **Bold** entries mark the best method.

C.2 Parameter Ablation

C.2.1 Schedules

In our paper, we proposed to use a cosine decaying strategy to determine β, reflecting an increase in the certainty of the model over the course of the training. Here, we compare this strategy to two alternative schedules, namely a constant and a linear function. Table 5 shows the respective results, where the more sophisticated schedules appear to be superior compared to the constant function.

Schedule	CIFAR-10		
	25 %	50 %	75 %
Constant ($\beta = 0.75$)	90.72 ±0.44	83.97 ±0.81	54.33 ±11.70
Linear ($\beta_0 = 0.75$, $\beta_1 = 0.6$)	91.35 ±0.29	85.16 ±1.48	35.50 ±1.52
Cosine ($\beta_0 = 0.75$, $\beta_1 = 0.6$)	**91.41** ±0.27	**86.46** ±0.47	**56.83** ±1.71

Table 5: Averaged test accuracies and standard deviations computed over three runs with different seeds. **Bold** entries mark the best method.

C.2.2 Varying β_0 and β_1

β_1	CIFAR-10	
	25 %	50 %
0.75	90.93	83.74
0.6	91.32	84.21
0.5	**91.65**	**85.98**
0.4	85.72	66.11
0.3	81.89	41.93
0.2	78.49	11.79

Table 6: Results for different β_1 parameters with $\beta_0 = 0.75$ using a cosine decaying β schedule. **Bold** entries mark the best parameter.

In another study, we look how changes in β_0 and β_1 for the cosine schedule behave in terms of generalization performance. The respective results are shown in Table 6 and 7.

C.3 Simplified Setting

Relating to the phenomenon of neural collapse [Papyan et al., 2020], we study the effects of our method in a simplified setting. More precisely, we consider multi-layer perceptron models consisting of a feature encoder and a classification head. The models consist of 6 dense layers with a width of 2048 neurons. To conform with previous studies [Nguyen et al., 2022], we restrict the number of features at

β_0	CIFAR-10	
	25 %	50 %
0.8	**90.18**	**82.63**
0.75	89.41	81.71
0.7	88.17	80.58
0.6	86.14	78.44
0.5	69.20	65.71

Table 7: Results for different β_0 parameters with $\beta_1 = 0.5$ using a cosine decaying β schedule. **Bold** entries mark the best parameter.

the last encoder layer to the number of classes K in the datasets. We consider the datasets MNIST, FashionMNIST and SVHN, from which we sample classes $K \in \{2, 3, 5, 10\}$. Apart from the model, we use the same hyperparameters as described in Table 1 for CIFAR-10. We used the same setup to construct Figure 3 as shown in the main paper. For $K = 2$, we can readily investigate the learned feature representations over the course of the training in a convenient manner.

These experiments aim to provide further evidence for the generalization of our method in more restricted settings under simplified model assumptions. Table 8 shows the results, where a consistent improvement of our method compared to typically considered neural collapsing functions can be observed. This suggests that our method is also applicable for more restricted models than typically considered overparameterized models.

Loss	MNIST				FashionMNIST				SVHN			
	$K=2$	$K=3$	$K=5$	$K=10$	$K=2$	$K=3$	$K=5$	$K=10$	$K=2$	$K=3$	$K=5$	$K=10$
CE	73.19	67.59	68.44	66.24	78.60	68.33	65.68	59.51	76.60	65.65	58.59	50.57
LS ($\alpha = 0.1$)	74.04	66.44	72.19	71.24	81.45	72.23	67.60	62.78	79.66	68.97	61.74	52.72
LR ($\alpha = 0.1$)	73.66	66.00	66.34	64.58	80.15	69.43	64.20	57.25	76.63	67.00	58.62	49.98
RDA (ours)	**97.59**	**90.78**	**87.86**	**82.75**	**92.45**	**91.63**	**82.36**	**77.13**	**86.77**	**73.03**	**65.67**	**53.98**

Table 8: Test accuracies on the three additional benchmark datasets using a simplified MLP with a restricted feature space at the penultimate layer. **Bold** entries mark the best method.

C.4 Combination with Sample Selection

In additional experiments, we integrated our RDA approach in a sample selection approach based on the small loss criterion [Gui et al., 2021], as also been applied in more sophisticated (semi-supervised) approaches that add substantial complexity to the learning setup. To this end, we train the model for 3 epochs on CIFAR-10(N)/-100 with cross-entropy based on the training examples, and take the 10 % training instances with the smallest cross-entropy loss. For these instances, we fix the label, i.e., we do not allow any ambiguity such that these instances serve as a corrective. Then, we train the model with our RDA loss as described

in Algorithm 1 with the hyperparameters reported in Tab. 1. In the following, we will refer to this variant as *RDA**.

Method	CIFAR-10			CIFAR-100		
	25 %	50 %	75 %	25 %	50 %	75 %
RDA	91.48 ±0.22	86.47 ±0.42	48.11 ±15.41	**70.03** ±0.32	59.83 ±1.15	26.75 ±8.83
RDA*	**91.79** ±0.21	**86.78** ±0.50	**67.08** ±2.09	69.98 ±0.17	**60.18** ±1.18	**30.82** ±9.45

Table 9: Test accuracies and standard deviations on the test split for models trained on CIFAR-10(0) with synthetic noise. The results are averaged over runs with different seeds, **bold** entries mark the best performing method per column.

Method	CIFAR-10N					
	Clean	Random 1	Random 2	Random 3	Aggregate	Worst
RDA	94.09 ±0.19	90.43 ±0.03	90.09 ±0.29	90.40 ±0.01	91.71 ±0.38	82.91 ±0.83
RDA*	**94.15** ±0.05	**90.76** ±0.13	**90.55** ±0.37	**90.86** ±0.09	**91.84** ±0.18	**83.79** ±0.24

Table 10: Test accuracies and standard deviations on the test split for models trained on CIFAR-10N with synthetic noise. The results are averaged over runs with different seeds, **bold** entries mark the best performing method per column.

As can be seen in the results present in Tables 9 and 10, RDA* can achieve almost consistently better performance with this addition, confirming its flexibility in being employed in more sophisticated setups. Notably, RDA* shows the largest improvement in robustness in the high noise regime (75 % noise).

D Additional Training Behavior Plots

D.1 CIFAR-10

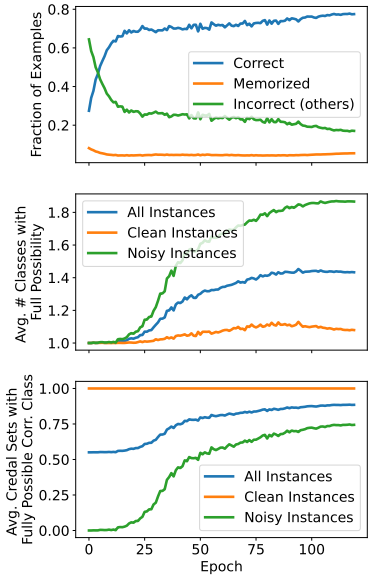

Figure 1: The training behavior of our method on CIFAR-10 with 50 % label noise averaged over five different seeds.

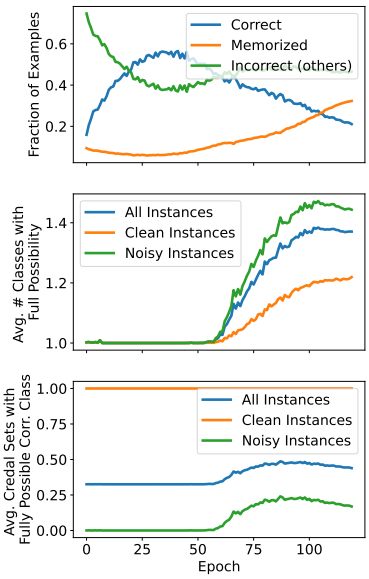

Figure 2: The training behavior of our method on CIFAR-10 with 75 % label noise averaged over five different seeds.

D.2 CIFAR-100

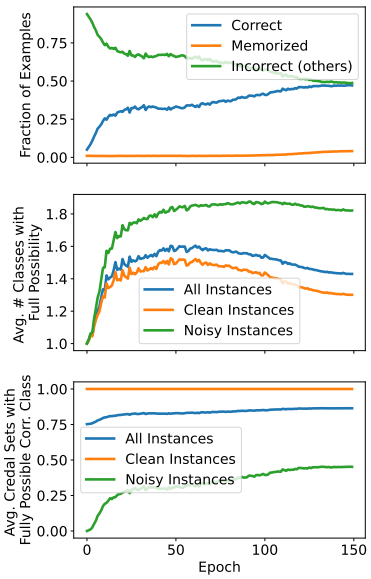

Figure 3: The training behavior of our method on CIFAR-100 with 25 % label noise averaged over five different seeds.

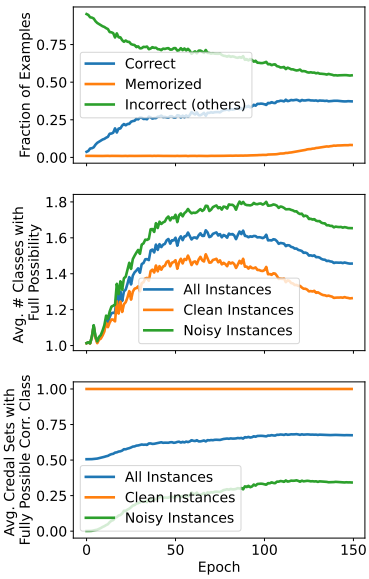

Figure 4: The training behavior of our method on CIFAR-100 with 50 % label noise averaged over five different seeds.

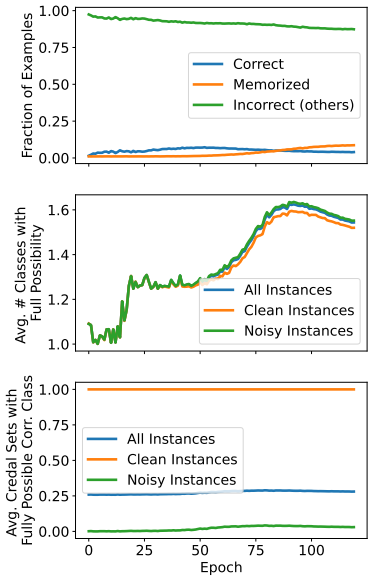

Figure 5: The training behavior of our method on CIFAR-100 with 75 % label noise averaged over five different seeds.

References

X. Gui, W. Wang, and Z. Tian. Towards understanding deep learning from noisy labels with small-loss criterion. In *Proc. of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI, August 19-27, Virtual Event / Montreal, Canada*, pages 2469–2475. ijcai.org, 2021.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Toronto, Canada, 2009.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998.

W. Li, L. Wang, W. Li, E. Agustsson, and L. V. Gool. WebVision database: Visual learning and understanding from web data. *CoRR*, abs/1708.02862, 2017.

S. Liu, J. Niles-Weed, N. Razavian, and C. Fernandez-Granda. Early-learning regularization prevents memorization of noisy labels. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS, December 6-12, Virtual Event*, 2020.

S. Liu, Z. Zhu, Q. Qu, and C. You. Robust training under label noise by over-parameterization. In *Proc. of the 39th International Conference on Machine Learning, ICML, July 17-23, Baltimore, Maryland, USA*, volume 162 of *Proc. of Machine Learning Research*, pages 14153–14172. PMLR, 17–23 Jul 2022.

Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NIPS, November 12-17, Granada, Spain, Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

D. A. Nguyen, R. Levie, J. Lienen, G. Kutyniok, and E. Hüllermeier. Memorization-dilation: Modeling neural collapse under noise. *CoRR*, abs/2206.05530, 2022.

V. Papyan, X. Y. Han, and D. L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proc. of the National Academy of Sciences of the United States of America*, 117:24652 – 24663, 2020.

G. Patrini, A. Rozza, A. K. Menon, R. Nock, and L. Qu. Making deep neural networks robust to label noise: A loss correction approach. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR, July 21-26, Honolulu, HI, USA*, pages 2233–2241. IEEE Computer Society, 2017.

J. Wei, Z. Zhu, H. Cheng, T. Liu, G. Niu, and Y. Liu. Learning with noisy labels revisited: A study using real-world human annotations. In *10th International Conference on Learning Representations, ICLR, April 25-29, Virtual Event*. OpenReview.net, 2022.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang. Learning from massive noisy labeled data for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR, June 7-12, Boston, MA, USA*, pages 2691–2699. IEEE Computer Society, 2015.

X. Zhou, X. Liu, J. Jiang, X. Gao, and X. Ji. Asymmetric loss functions for learning with noisy labels. In *Proc. of the 38th International Conference on Machine Learning, ICML, July 18-24, Virtual Event*, volume 139 of *Proc. of Machine Learning Research*, pages 12846–12856. PMLR, 2021.