When does Privileged Information Explain Away Label Noise?

Guillermo Ortiz-Jimenez [*1] **Mark Collier** [*2] **Anant Nawalgaria** [2] **Alexander D'Amour** [2] **Jesse Berent** [2] **Rodolphe Jenatton** [2] **Effrosyni Kokiopoulou** [2]

Abstract

Leveraging *privileged information* (PI), or features available during training but not at test time, has recently been shown to be an effective method for addressing label noise. However, the reasons for its effectiveness are not well understood. In this study, we investigate the role played by different properties of the PI in explaining away label noise. Through experiments on multiple datasets with real PI (CIFAR-N/H) and a new large-scale benchmark ImageNet-PI, we find that PI is most helpful when it allows networks to easily distinguish clean from mislabeled data, while enabling a learning shortcut to memorize the mislabeled examples. Interestingly, when PI becomes too predictive of the target label, PI methods often perform worse than their no-PI baselines. Based on these findings, we propose several enhancements to the state-of-the-art PI methods and demonstrate the potential of PI as a means of tackling label noise. Finally, we show how we can easily combine the resulting PI approaches with existing no-PI techniques designed to deal with label noise.

1. Introduction

Label noise, or incorrect labels in training data, is a pervasive problem in machine learning that is becoming increasingly common as we train larger models on more weakly annotated data. Human annotators are often the source of this noise, assigning incorrect labels to certain examples (Snow et al., 2008; Sheng et al., 2008), e.g., when the class categories are too fine-grained. Incorrect labeling can also come from using other models to provide proxy labels (Prabhu et al., 2022) or scraping the web (Radford et al., 2021). However, the standard approach in supervised learning is to ignore this issue and treat all labels as correct, leading to significant drops in model performance as the models tend to memorize the noisy labels and degrade the learned representations (Zhang et al., 2017).

Recently, some studies have proposed to mitigate the effect of label noise by leveraging *privileged information* (PI) (Vapnik & Vashist, 2009; Collier et al., 2022) i.e., features available at training time but not at test time. Examples of PI are features describing the human annotator that provided a given label, such as the annotator ID, the amount of time needed to provide the label, the experience of the annotator, etc. While several PI methods have shown promising gains in mitigating the effects of label noise (Lopez-Paz et al., 2016; Lambert et al., 2018; Collier et al., 2022), the reasons behind their success are not fully understood. Moreover, the fact that existing results have been provided in heterogeneous settings makes comparisons and conclusions difficult to be drawn.

In this work, we aim to standardize the evaluation of PI and conduct a large-scale study on the role of PI in explaining away label noise. We examine the performance of several PI methods on different noisy datasets and analyze their behavior based on the predictive properties of the available PI. Interestingly, we find that when the PI is too predictive of the target label, the performance of most PI methods significantly degrades below their no-PI baselines as the models fail to learn the associations between the non-privileged input features and the targets. Conversely, we discover that the strongest form of PI exhibits two main properties: (i) it allows the network to easily separate clean and mislabeled examples, and (ii) it enables an easier "learning shortcut" (in a sense to be clarified later) to overfit to the mislabeled examples (see Figure 1). When both these properties are present, the performance of PI methods significantly exceeds their no-PI counterparts by becoming more robust to label noise.

Overall, we observe that using PI during training can enable models to discover shortcuts that can prevent learning the relationship between features and labels (Geirhos et al., 2020; D'Amour et al., 2020). On the one hand, these *PI-enabled shortcuts* can have a positive effect when the relationship being ignored only concerns incorrect labels, which thus

[*]Equal contribution [1]Ecole Polytechnique Fédérale de Lausanne (EPFL). Work done during an internship at Google. [2]Google Research. Correspondence to: Guillermo Ortiz-Jimenez <guillermo.ortizjimenez@epfl.ch>, Mark Collier <mark.collier@google.com>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

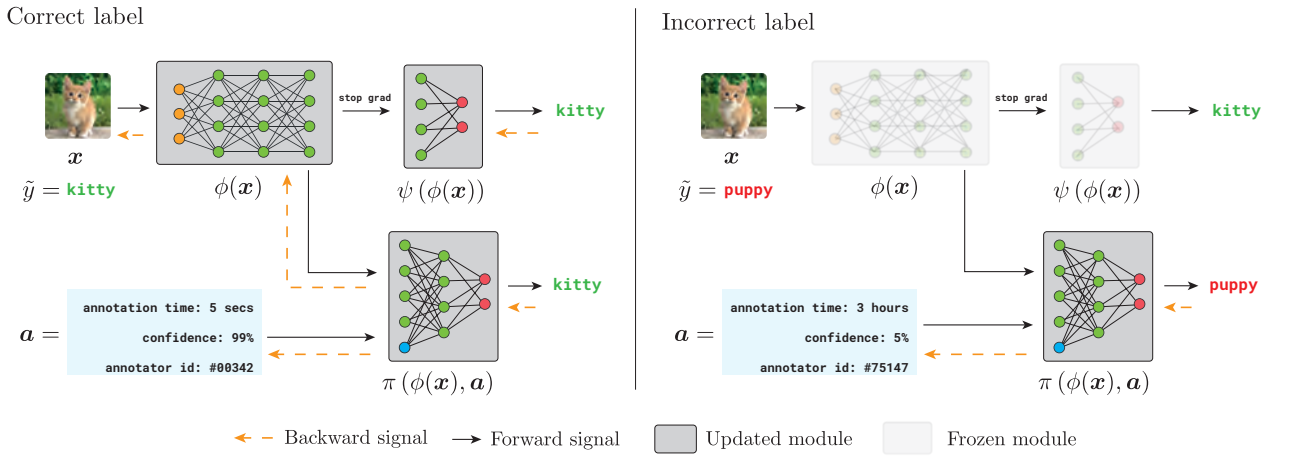

Figure 1. Conceptual illustration of ideal signal propagation while training a privileged information method such as TRAM (Collier et al., 2022) with noisy labels. Having access to PI allows a network to use a learning shortcut to memorize the mislabeled examples using only PI. This protects the extraction of features from the actual data, which are only refined using the correctly labeled examples.

prevents the learned feature representations from being contaminated by label noise. On the other hand, they can have a detrimental effect when the clean labels are also affected, which prevents the model from learning the correct association between features and targets. Shortcuts are key to understanding the role of PI in mitigating label noise and are directly linked to deep learning dynamics (Zhang et al., 2017).

When focusing on the dynamics of the strongest PI methods, we show that using PI allows training larger models on datasets with a higher level of noise. PI can counteract the negative effect of memorizing incorrect associations between features and incorrect labels as it enables a shortcut that primarily affects the mislabeled examples. We use these new insights to improve current state-of-the-art PI algorithms.

Overall, the main contributions of our work are:

- We present the first large-scale study on the role of PI on supervised noisy datasets of various types.
- We release ImageNet-PI, the largest available testbed for experimenting with PI and label noise.
- We find that effective PI enables learning shortcuts only on mislabeled data greatly benefitting performance.
- We improve a wide range of PI methods using simple improvements, and demonstrate cumulative gains with other state-of-the-art noisy labels methods.

We believe our findings can have a significant impact on future research about both label noise and PI. They indeed not only inform us about the desired properties of the ideal PI (which can help design and collect PI features in practice) but also provide practical insights for improving existing methods. Formally capturing our empirical results is another promising direction for future research.

2. Methodology

Our large-scale experiments provide a comprehensive analysis of the usefulness of PI in the presence of label noise. Previous results have been provided in heterogeneous settings with testing performed on different datasets and various types of PI, making well-aligned comparisons difficult. We aim at standardizing these comparisons, making the unification of all the settings of our experiments part of our core contribution. Our code can be found at `https://github.com/google/uncertainty-baselines`. In what follows, we briefly describe the datasets and baselines used in our study.

2.1. Datasets

In this work, we address the supervised learning setting with PI and label noise as described in Collier et al. (2022). Our training data consists of triplets $(\boldsymbol{x}, \tilde{y}; \boldsymbol{a})$, where $\boldsymbol{x} \in \mathbb{R}^d$ is a set of input features, $\tilde{y} \in \{1, \ldots, K\}$ is a noisy target label (assuming K classes), and $\boldsymbol{a} \in \mathbb{R}^p$ is a vector of PI features. In this work, we mainly focus in the case when these PI features are related to the annotation process, as this is a common source of label noise (Snow et al., 2008; Sheng et al., 2008). This PI may include information about the annotator, such as their ID or experience; or about the process itself, such as the annotation duration or confidence. At test time, we do not have access to any PI and evaluate our models based only on clean $(\boldsymbol{x}, y)$ pairs from the data distribution.

We use relabelled versions of standard image recognition datasets which provide various forms of PI about the annotation process in our experiments. These datasets allow us to access both clean (y) and noisy labels ($\tilde{y}$), but we only use the noisy labels for training and hyperparameter selection (see details in Appendix D for a discussion about the effect of noisy labels at this step). The clean labels are only used for evaluation. The datasets we use offer a range of training conditions, including differing numbers of samples and classes, levels of noise, types of PI, image sizes, and annotation processes, making our findings widely applicable.

Some of these datasets provide multiple annotations per example. Nonetheless, to create a unified benchmark we only sample one label per example for datasets that provide multiple labels. So that we can control the noise level and examine its impact on the performance of PI methods, we create high and low noise versions of each dataset, when possible. We follow the terminology of Wei et al. (2022) by naming the low noise version the "uniform" version, which selects one of the available labels uniformly at random, and the high noise version, the "worst" version, which always selects an incorrect label if available. The "worst" version is by design more noisy than the "uniform" one.

CIFAR-10/100N. A relabelled version of the CIFAR-10/100 datasets (Krizhevsky, 2009) that includes multiple annotations per image (Wei et al., 2022). The raw data includes information about the annotation process, such as annotation times and annotator IDs, but this information is not given at the example level. Instead, it is provided as averages over batches of examples, resulting in coarse-grained PI. We will show that the PI baselines perform poorly on this dataset. The "uniform" version of CIFAR-10N agrees 82.6% of the time with the clean labels, and the "worst" version 59.8%. CIFAR-100N agrees 59.8% with the clean labels. For reference, training our base architectures without label noise and without PI achieves test accuracies of 93.5% and 77.9% on CIFAR-10 and CIFAR-100, respectively.

CIFAR-10H. An alternative human-relabelled version of CIFAR-10, where the new labels are provided only on the test set (Peterson et al., 2019). As in Collier et al. (2022), when we train on CIFAR-10H, we evaluate the performance of the models on the original CIFAR-10 *training set* (since CIFAR-10H relabels only the validation set). Contrary to the CIFAR-N datasets, CIFAR-10H contains rich PI at the example-level, with high-quality metadata about the annotation process. The "uniform" version agrees 95.1% of the time with the clean labels, and the "worst" 35.4%. For reference, training our base architecture without label noise and without PI achieves a test accuracy of 88.4% on CIFAR-10H.

ImageNet-PI. Inspired by Collier et al. (2022), a relabeled version of ImageNet (Deng et al., 2009) in which the labels are provided by a set of pre-trained deep neural networks with different architectures. During the relabeling process, we sample a random label from a temperature-scaled predictive distribution of each model on each example. This leads to label noise that is asymmetrical and feature-dependent. Technical details of the relabeling process and temperature-scaling can be found in Appendix A. The PI of the dataset comes from the confidences of the models on the sampled labels, the parameter counts of the models, and the models' test accuracies on the clean test distribution. These PI features serve as a good proxy for the expected reliability of each model. The ImageNet-PI high-noise version that we use agrees 16.2% of the time with the clean labels and the low-noise version 51.9%. For reference, training our base architecture without label noise and without PI achieves a test accuracy of 76.2% on ImageNet. As a contribution of this work, we open-source ImageNet-PI (with different amounts of label noise) to encourage further research on PI and label noise at a scale larger than possible today with CIFAR-N/H. The data is publicly available at `https://github.com/google-research-datasets/imagenet_pi`.

2.2. PI Algorithms

We study the performance of four representative approaches that exploit PI. They all have been shown to be effective at mitigating the effect of label noise (Collier et al., 2022):

no-PI. A standard supervised learning baseline that minimizes the cross-entropy loss on the noisy labels to approximate $p(\tilde{y}|\boldsymbol{x})$ without access to PI.

Distillation (Lopez-Paz et al., 2016). A knowldege distillation method in which a teacher model is first trained using standard maximum likelihood estimation with access to PI to approximate $p(\tilde{y}|\boldsymbol{x}, \boldsymbol{a})$. A student model with the same architecture is later trained to match the output of the teacher without access to the PI. We also provide results for a standard self-distillation baseline in which the teacher model does not have access to PI (Hinton et al., 2015).

TRAM (Collier et al., 2022). Method based on a two-headed model in which one head has access to PI and the other one not. At training time, a common feature representation $\phi(\boldsymbol{x})$ is fed to two classification heads $\pi(\phi(\boldsymbol{x}), \boldsymbol{a})$ ("PI head") and $\psi(\phi(\boldsymbol{x}))$ ("no-PI head") to jointly solve

$$\min_{\phi,\pi,\psi} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{a},\tilde{y})} \left[\ell(\pi(\phi(\boldsymbol{x}), \boldsymbol{a}), \tilde{y}) + \ell(\psi(\phi(\boldsymbol{x})), \tilde{y})\right]. \quad (1)$$

Importantly, during training, the no-PI feature extractor ϕ is updated using *only* the gradients coming from the PI head. At test time, only the no-PI head is used for prediction.

Approximate Full Marginalization (Collier et al., 2022). A neural network is first trained using maximum likelihood estimation with access to PI to approximate $p(\tilde{y}|\boldsymbol{x}, \boldsymbol{a})$. During inference, a Monte-Carlo estimation is used to approximate the marginal

$$p(\tilde{y}|\boldsymbol{x}) = \int p(\tilde{y}|\boldsymbol{x}, \boldsymbol{a}) p(\boldsymbol{a}|\boldsymbol{x}) \, \mathrm{d}\boldsymbol{a}$$

typically further assuming the independence $p(\boldsymbol{a}|\boldsymbol{x}) \approx p(\boldsymbol{a})$. Note that this process increases the memory and computational costs during inference as it requires computing the output of the network for each of the different sampled values of $\boldsymbol{a}$ (in practice, in the order of $1,000$ extra values).

All the methods use the same underlying network architecture with minimal changes to accommodate their specific requirements, like Collier et al. (2022). In particular, during inference, all methods use exactly the same network, except for the approximate full marginalization (AFM) baseline, which has additional parameters to deal with the sampled PI.

In all experiments, we use the same protocol for training with noisy labels and evaluating on a clean test set. As early stopping is a strong baseline against label noise (Bai et al., 2021), we always report values of test accuracy at the end of the epoch that achieves the best performance on a held-out validation percentage of the *noisy* training set. We reproduce our results without early stopping in Appendix D. Unless otherwise specified, we conduct a grid search to tune the most important hyperparameters of each method for each experiment, and report the mean test accuracy and standard deviation over five runs. Further details on the experimental setup can be found in Appendix B.

3. When is PI helpful?

Table 1 (Original) shows the performance of the different PI algorithms on our collection of noisy datasets[1], where we see that leveraging PI does not always yield big gains in performance. Indeed, while TRAM and AFM substantially improve upon the no-PI baseline on CIFAR-10H and ImageNet-PI, they do not perform much better on CIFAR-10N and CIFAR-100N. Moreover, we observe little gains of Distillation (PI) over the vanilla self-distillation baseline.

The performance disparities of the same algorithms on datasets where the main source of variation is the available PI, i.e., CIFAR-10N *vs.* CIFAR-10H, highlights that leveraging PI is not always helpful. In fact, depending on the predictive properties of the PI and the noise distribution, we report very different results. This begs the questions: i) "*what makes PI effective for these algorithms?*" and ii) "*how do they exploit PI to explain away label noise?*".

[1]We present results for high-noise in the main text. A reproduction of Table 1 with low-noise can be found in Appendix C.

To answer these question, we perform a series of controlled experiments in which we train our PI methods using different PI features (including both real and synthetic ones). By doing so our objective is to identify the main mechanisms that lead to the top performance of these algorithms.

3.1. Fully predictive PI

Hypothesis: The PI $\boldsymbol{a}$ always complements the information about the labels $\tilde{y}$ contained in $\boldsymbol{x}$.

It is natural to assume that knowing $\boldsymbol{a}$ on top of $\boldsymbol{x}$ can help predict $\tilde{y}$ and thus improve over supervised learning. However, this reasoning is flawed as it forgets that during inference the models cannot exploit $\boldsymbol{a}$. On the contrary, as we will see, if $\boldsymbol{a}$ is very predictive of the target $\tilde{y}$, the test performance can severely degrade.

We test this hypothesis by retraining the algorithms on the noisy datasets but using $\boldsymbol{a} = \tilde{y}$ instead of the original PI features. That is, having access to fully predictive PI.

Finding: When $\boldsymbol{a}$ is fully predictive of $\tilde{y}$, most PI methods perform worse than the no-PI baselines.

As we can see in Table 1 (Labels), all the PI baselines greatly suffer in this regime. The reason for this is simple: when the PI is too informative of the target label, then the models are heavily relying on the PI to explain the label and they are discouraged from learning any associations between $\boldsymbol{x}$ and $\tilde{y}$ and do not learn any meaningful feature representations. In this regard, we see how Distillation (PI) achieves roughly the same performance as Distillation (no-PI), while TRAM and AFM achieve very low test accuracies.

The fact that very predictive PI can hurt the performance of these algorithms highlights a key element of their dynamics: PI can enable learning shortcuts (D'Amour et al., 2020; Geirhos et al., 2020) that prevent learning certain associations between $\boldsymbol{x}$ and $\tilde{y}$, possibly by starving the gradient signal that updates $\phi(\boldsymbol{x})$ (Pezeshki et al., 2021). This has practical implications as it discourages blindly appending arbitrarily complex metadata to $\boldsymbol{a}$ during training which could be very predictive of the target label.

3.2. Noise indicator

Hypothesis: PI helps because it can separate mislabeled from correct examples.

We saw that when $\boldsymbol{a}$ is too predictive of $\tilde{y}$, the PI approaches perform poorly. We now turn to an alternative hypothesis of why PI can be beneficial to explain away label noise: The PI features can help the network separate the clean from the mislabeled examples. Indeed, the original motivation of using PI to fight label noise in Collier et al. (2022)

Table 1. Test accuracy of several methods trained using different features as PI (baselines in gray and *italics* do not use PI). Here, Original denotes the standard PI of the dataset, Indicator a binary signal that separates clean from noisy examples, Labels the one-hot encoded labels, and Near-optimal a synthetic feature that gives the annotator label to those examples that are miss-annotated and a zero-vector otherwise. **Bold numbers** represent significant maximum values across PI features where significance means p-value < 0.05.

		Original	Indicator	Labels	Near-optimal
CIFAR-10H (worst)	*no-PI*	55.0±1.5	55.0±1.5	55.0±1.5	55.0±1.5
	Distillation (no-PI)	47.9±0.0	47.9±0.0	47.9±0.0	47.9±0.0
	TRAM	64.9±0.8	63.3±1.1	38.3±0.2	**67.8±0.2**
	Approximate FM	64.0±0.6	66.7±2.1	29.5±0.5	**74.4±0.1**
	Distillation (PI)	45.4±0.8	**49.9±0.7**	44.5±0.1	48.2±0.9
CIFAR-10N (worst)	*no-PI*	80.6±0.2	80.6±0.2	80.6±0.2	80.6±0.2
	Distillation (no-PI)	80.4±0.0	80.4±0.0	80.4±0.0	80.4±0.0
	TRAM	80.5±0.5	87.9±0.4	48.9±0.2	**89.3±0.3**
	Approximate FM	82.0±0.3	91.2±0.3	22.6±0.2	**92.0±0.1**
	Distillation (PI)	80.2±0.3	80.1±0.3	**80.7±0.2**	80.2±0.3
CIFAR-100N	*no-PI*	60.4±0.5	60.4±0.5	60.4±0.5	60.4±0.5
	Distillation (no-PI)	60.6±0.2	60.6±0.2	60.6±0.2	60.6±0.2
	TRAM	59.7±0.3	62.4±0.3	34.9±0.2	**67.4±0.3**
	Approximate FM	60.0±0.2	66.4±0.2	20.1±0.3	**70.2±0.1**
	Distillation (PI)	61.1±0.2	**61.8±0.3**	60.5±0.2	**61.5±0.3**
ImageNet-PI (high-noise)	*no-PI*	47.7±0.8	47.7±0.8	47.7±0.8	47.7±0.8
	Distillation (no-PI)	50.2±0.8	50.2±0.8	50.2±0.8	50.2±0.8
	TRAM	53.3±0.5	53.6±0.5	41.0±0.7	**56.5±0.3**
	Approximate FM	55.6±0.3	55.3±0.6	0.8±0.2	**58.3±0.1**
	Distillation (PI)	**51.0±0.4**	**50.6±0.2**	39.0±4.6	27.5±22.7

was that annotator features, e.g., confidences, could act as proxy to identify mislabeled samples. Intuitively, the main assumption is that if the PI can properly identify the mislabeled examples, then it should act as expert knowledge that would discourage focusing on the hard mislabeled instances, and instead, promote learning only on the correct easy ones (Vapnik & Vashist, 2009).

Albeit natural, this hypothesis has not been tested before, but can be done using the datasets in this study. Recall that we have access to clean and noisy labels for all the training samples, and thus we can synthesize an indicator signal $\mathbb{1}(\tilde{y} \neq y)$ that takes a value of 1 when the clean and noisy labels agree and 0 otherwise. Table 1 (Indicator) shows the results of training using $\boldsymbol{a} = \mathbb{1}(\tilde{y} \neq y)$.

Finding: Some PI methods perform better with the original PI than with an oracle noise indicator.

Interestingly, although we see that the performances on the Indicator columns are generally higher than on the Original one, this is not always the case, and sometimes the indicator underperforms or does not significantly improve over the original PI (cf. AFM and TRAM on CIFAR10H and ImageNet-PI). This suggests that the PI methods do more than just leveraging the noise indication abilities of the PI. Clearly, if even using an ideal noise indicator signal $\mathbb{1}(\tilde{y} \neq y)$ as PI, the original PI can sometimes outperform it, then there must be other information in the PI that the algorithms can exploit to improve performance.

3.3. Memorization dynamics play a significant role

Inspecting the training dynamics of the algorithms can help understand the previous results. For example, Figure 2 shows the evolution of test and training accuracies of a TRAM model on CIFAR-10H using different PI features[2]. The original PI leads to better final test accuracy than the noise indicator. Meanwhile, models trained using annotator labels as PI do not seem to learn anything useful. These differences are explained by the rates at which these models fit the mislabeled and correct samples using each of the TRAM heads.

Focusing on the training accuracies of the PI-head, Figure 2 (right column) explains why giving the labels as PI hurts the test performance[3]. The label model quickly

[2]Results for others settings can be found in Appendix E.1.

[3]The blue line sits behind the yellow line in Figure 2 (top right).

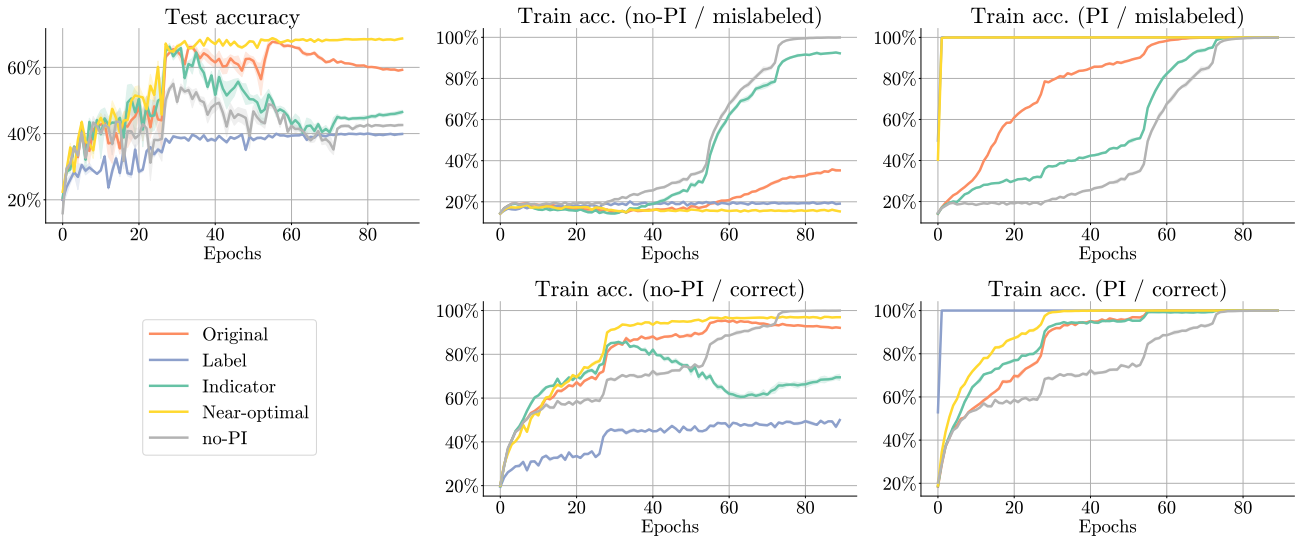

Figure 2. Dynamics of TRAM on CIFAR-10H with different PIs. **Top left**: Test accuracy. **Top center**: Train accuracy on mislabeled examples evaluated at the no-PI head, **Top right**: Train accuracy on mislabeled examples evaluated at the PI-head. **Bottom center**: Train accuracy on clean examples evaluated at the no-PI head. **Bottom right**: Train accuracy on clean examples evaluated at the PI head.

obtains 100% training accuracy on all examples (mislabeled and correct) using the PI head, which in turn slows the training speed of the no-PI head (central column). This happens because the feature extractor is only updated by gradients from the PI head, leading to a lack of meaningful representation of $p(\tilde{y}|\boldsymbol{x})$ if the model is learning to fit all examples using PI features alone.

Focusing on the training accuracies of the no-PI head in Figure 2 (central column), the best models are those that achieve the highest training accuracy on correct examples, while not overfitting to the mislabeled. The difference in test performance of indicator and original is explained by the original model having a harder time overfitting to the mislabeled examples. Interestingly, the original model memorizes mislabeled examples faster with the PI head than the indicator. It looks as though fitting the training examples fast with the PI head was discouraging the model from fitting the same examples with the no-PI head, i.e., the PI is enabling a learning shortcut to memorize the mislabeled examples with the original PI, without using $\boldsymbol{x}$. This might be because the indicator signal only takes values in $\{0, 1\}$ for all examples, and these are not enough to separate the noisy training set. Indeed, as we will see, having access to PI that can be easily memorized on the mislabeled examples is fundamental to maximize performance.

3.4. Near-optimal PI features

Hypothesis: The optimal PI enables a learning shortcut to memorize *only* the mislabeled labels.

The experiments using the annotator labels as PI are a clear example of a PI-enabled learning shortcut which is very detrimental for the model performance. On the other hand, the dynamics of the original models hint that the same shortcut mechanism can also have a positive effect when it only applies to the mislabeled examples. To test this hypothesis, we design a new form of PI features, denoted as near-optimal in the tables and plots. As its name indicates, this PI should allow the models to get very close to their top performance. The near-optimal features are designed to exploit the PI shortcut only on the mislabeled examples, allowing the model to learn freely on the correct ones. To that end, the near-optimal PI features consist of two concatenated values: (i) the indicator signal that says if a given example is mislabeled or not, and (ii) the annotator label *only* if that example is mislabeled. Otherwise an all-zero vector is concatenated with the same dimensionality as the one-hot encoded labels to the indicator signal.

Finding: When a learning shortcut is provided *only* for mislabeled examples, PI methods achieve top performance.

The results in Table 1 (Near-optimal) show that those PI features significantly outperform all other PI features by a large margin on all datasets when using TRAM or AFM[4]. Similarly, in Figure 2 we observe that the dynamics of the near-optimal models fully match our expectations. The near-optimal models train the fastest on the mislabeled examples

[4]Near-optimal does not always outperform original when using Distillation (PI), but note that in general the gains of Distillation (PI) over (no-PI) are much smaller than for TRAM and AFM. In this regard, we leave the objective of finding a near-optimal policy for Distillation (PI) as an open question for future work.

on the PI head, thus leading to a very slow training speed on mislabeled examples on the no-PI head. Moreover, since the mislabeled examples no longer influence (because their accuracies are already maximal on the PI head) the updates of the feature extraction, then we observe that the performance on the correct examples is much higher.

The same explanation applies to AFM whose dynamics are shown in Appendix E.1. In this case, the memorization of the mislabeled examples using PI alone also protects the no-PI features. This way, during inference, the PI sampled from the mislabeled examples simply adds a constant noise floor to the predicted probabilities of the incorrect labels. This averaged noise floor is usually much smaller than the probability predicted using the clean features of the no-PI, and thus does not lead to frequent misclassification errors.

4. Improving PI algorithms

In this section, we use the insights of the previous analysis to improve the design of PI methods. We perform ablation studies on different design parameters of the main PI approaches, identifying simple modifications that significantly boost their performance. We primarily focus on TRAM and AFM as these methods outperform Distillation (PI) by a large margin when the PI is helpful (cf. Table 1). We provide illustrative results here, and full results in Appendix E.

4.1. Model size

We explore how the model size affects performance. In particular, note that the parameter count of all PI algorithms can be split into two parts: the feature extractor ϕ of the standard features $\boldsymbol{x}$ and the tower π that processes the PI; see Eq. (1) and Section 2.2. We therefore perform an ablation study in which we scale each of these parts of the models separately.

Feature extractor. Figure 3 shows how test accuracy changes as we increase the size of the feature extractor of the PI approaches. The performance follows a U-shape, where scaling the model past a certain point harms final performance. Indeed, a larger capacity discourages the model from using PI features and causes overfitting to standard features, as shown by the simultaneous increase in training accuracy on mislabeled examples and decrease in test accuracy.

> **Finding**: Increasing the feature extractor size discourages using the PI as a shortcut.

PI head size. Figure 4 shows the results of scaling the size of the PI processing tower while keeping the feature extractor size fixed. We observe how larger PI heads improve performance as they encourage more memorization using PI alone and protect the extraction of the no-PI features. This

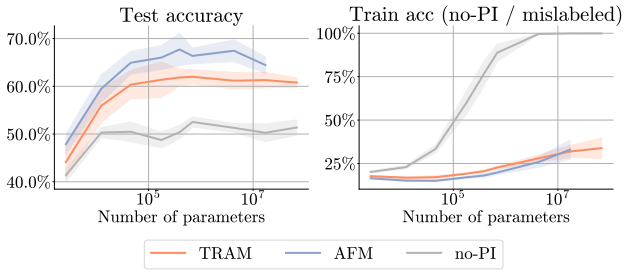

Figure 3. Performance of different PI baselines on CIFAR-10H when increasing the parameter count of their feature extractor keeping the PI tower fixed. Larger models suffer from overfitting as they tend to use their larger capacity to overfit to mislabeled examples, discouraging the model from exploiting the PI.

is illustrated by the decay of the training accuracy of the mislabeled examples on the no-PI head for larger PI heads.

> **Finding**: Increasing the capacity of the PI tower encourages using the PI as a shortcut.

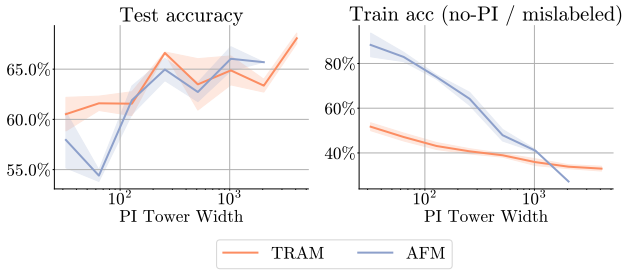

Figure 4. Performance of different PI approaches on CIFAR-10H when increasing the PI head size. A larger PI head size incentivizes the model to memorize the mislabeled examples using the PI, thus further exploiting PI as a shortcut.

4.2. Random PI can enable positive shortcuts

> **Hypothesis**: Random PI that uniquely identifies each example can enable a PI shortcut that protects the model from memorizing incorrect labeles with $\boldsymbol{x}$.

The near-optimal, labels, and indicator signals of Table 1 are all synthetic PI features that cannot be used in practice, as they rely on the knowledge of which examples are mislabeled and which examples are correct. However, they show that having access to a signal that can be more easily memorized than the standard features $\boldsymbol{x}$ on the mislabeled examples is a good recipe to improve performance. This being said,a key property of incorrect labels is that they are, by definition, a mistake. In this sense, fitting an incorrect training label simply amounts to memorizing a specific training example whose features are not predictive of the target label, i.e., the features serve just as an example ID. In fact, any set of features which are different enough for each

example could act as such an ID.

Finding: Random PI is effective at reducing overfitting to the incorrect labels using $\boldsymbol{x}$.

We evaluate this hypothesis in Table 2 where we introduce TRAM++: a version of TRAM in which the original PI features are augmented with a unique random vector for each example (experimental details are provided in Appendix F and results for AFM++ in Appendix G). As we can see, TRAM++ generally achieves better performance than TRAM alone, with greater improvements in those datasets where overfitting is a bigger issue (i.e., CIFAR).

5. Combination with other no-PI techniques

In this section, we show experimentally that the performance improvements obtained by PI methods on noisy datasets can work symbiotically with other state-of-the-art techniques from the noisy label literature. In particular, we show that TRAM++ can be easily combined with Sparse Over-parameterization (SOP) (Liu et al., 2022) and Heteroscedastic output layers (Collier et al., 2021) while providing cumulative gains with respect to those baselines[5].

5.1. Sparse Over Parameterization (SOP)

Sparse over-parameterization (SOP) (Liu et al., 2022) is a state-of-the-art method which leverages the implicit bias of stochastic gradient descent (SGD) and overparameterization to estimate and correct the noisy label signal, a concept which has proven to work well (Zhao et al., 2022). It does so by adding two new sets of K-dimensional parameters $\{\boldsymbol{u}_i\}_{i=1}^N$ and $\{\boldsymbol{v}_i\}_{i=1}^N$, where N denotes the number of training points, and solving

$$\min_{\boldsymbol{\theta},\{\boldsymbol{u}_i,\boldsymbol{v}_i\}_{i=1}^N} \frac{1}{N}\sum_{i=1}^{N} \ell\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}) + \boldsymbol{u}_i \odot \boldsymbol{u}_i - \boldsymbol{v}_i \odot \boldsymbol{v}_i, \tilde{y}\right)$$

using SGD. This specific parameterization biases the solution of SGD towards the recovery of the noise signal $\boldsymbol{\epsilon}_i = \boldsymbol{u}_i \odot \boldsymbol{u}_i - \boldsymbol{v}_i \odot \boldsymbol{v}_i$ that corrupts y, i.e., $\tilde{y}_i \approx y_i + \boldsymbol{\epsilon}_i$, implicitly assuming that $\boldsymbol{\epsilon}_i$ is sparse across the dataset.

In this work, we explore whether the combination of TRAM++ with SOP can yield cumulative gains in performance against label noise. In particular, we propose a simple two-step training process to combine them: (i) We first pretrain a neural network using TRAM++ and (ii) we finetune the no-PI side of the network using the SOP loss without stop-gradients. Table 2 shows the results of this method[6] where we see that, indeed, TRAM+SOP is able to significantly outperform TRAM++ or SOP alone in all datasets. More experimental details can be found in Appendix I.

5.2. Heteroscedastic output layers

Finally, we further analyze the combination of TRAM with HET, another state-of-the-art no-PI baseline from the noisy label literature that can be scaled up to ImageNet scale (Collier et al., 2021). HET here refers to the use of heteroscedastic output layers to model the aleatoric uncertainty of the predictions without PI. In particular, we apply HET layers to both heads of TRAM++ and follow the same training setup. We call the resulting approach TRAM+HET.

Our experiments, presented in Table 2, show that the TRAM+HET model outperforms both TRAM++ and HET applied alone . More experimental details about that model combination can be found in Appendix J. All in all, these results corroborate our main findings:

Finding: PI methods work symbiotically with other no-PI algorithms from the noisy label literature.

6. Related work

The general framework of learning with privileged information (Vapnik & Vashist, 2009) has been widely studied in deep learning, with many works exploring different baselines, including loss manipulation (Yang et al., 2017), distillation (Lopez-Paz et al., 2016), or Gaussian dropout (Lambert et al., 2018). This line of work has mainly focused on the noiseless scenario, conceiving PI as a guiding signal that helps identify easy or hard instances (Vapnik & Izmailov, 2015). Similar to our work, Yang et al. (2022) also studied the role of PI in improving the performance of deep learning methods, but focusing on the task of learning-to-rank using distillation methods in the noiseless setting.

More recently, Collier et al. (2022) proposed a new perspective on PI, arguing that it can make models more robust to the presence of noise. Their proposed PI approach, referred to as TRAM, led to gains on various experimental settings, with both synthetic and real-world noise. However, their results lacked a detailed analysis of how different sources of PI affect performance.

Our work takes inspiration from the rich deep-learning theory studying the memorization dynamics of neural networks (Zhang et al., 2017; Rolnick et al., 2017; Toneva et al., 2019; Maennel et al., 2020; Baldock et al., 2021). In the no-PI setting, the dynamics of neural networks wherein the incorrect labels tend to be later memorized during training has been heavily exploited by the noisy-label community through techniques such as early-stopping and regularization (Liu et al., 2020; Bai et al., 2021). Other works have

[5] We provide further experiments combining TRAM with label smoothing (Szegedy et al., 2016) in Appendix H.

[6] We do not provide results for ImageNet-PI as SOP cannot be easily scaled to such a large dataset.

Table 2. Performance comparison of no-PI, TRAM, TRAM++, SOP, HET, TRAM+SOP and TRAM+HET on the different PI datasets.

	no-PI	TRAM	TRAM++	SOP	TRAM+SOP	HET	TRAM+HET
CIFAR-10H (worst)	55.0±1.5	64.9±0.8	66.8±0.3	59.2±0.8	**70.9±0.5**	50.8±1.4	67.7±0.7
CIFAR-10N (worst)	80.6±0.2	80.5±0.5	83.9±0.2	87.9±0.2	**88.5±0.3**	81.9±0.4	82.0±0.3
CIFAR-100N	60.4±0.5	59.7±0.3	61.1±0.2	65.3±0.3	**66.1±0.2**	60.8±0.4	62.1±0.1
ImageNet-PI (high-noise)	47.7±0.8	53.3±0.5	53.9±0.4	–	–	51.5±0.6	**55.8±0.3**

exploited the intrinsic difference between the learning of clean and mislabeled examples to detect and correct misclassification errors using self-supervision (Veit et al., 2017; Li et al., 2020), co-teaching (Han et al., 2018), or regularization (Cheng et al., 2021). Finally, many works have attempted to model the label corruption process by estimating the label transition matrix (Patrini et al., 2017) or the noisy signal directly in the prediction space (Liu et al., 2022). In general, we see this line of research about noisy labels (Song et al., 2020) as orthogonal to the use of PI and we have experimentally shown that our PI approach is in fact complementary and can be gracefully combined with such techniques.

Some aspects of this work are suggestive of causal reasoning. In particular, *explaining away* is a well-known phenomenon when there are multiple explanations for the value that a particular variable has taken, e.g., whether it is the ground-truth label correctly annotated, or a mistake from an annotator (Pearl, 2009). We do not use causal formalism explicitly in this work, although we see similar learning dynamics at play in our results. PI (often called auxiliary labels) is also used in causally-motivated work on robust ML, although this is usually focused on the distinct problem of handling spurious correlations, rather than overcoming label noise (Kallus et al., 2018; Veitch et al., 2021; Makar et al., 2022). In self-supervised learning, the removal of shortcuts is also a topic of interest (Minderer et al., 2020).

7. Conclusions

In this work, we have presented a systematic study in which we investigate which forms of PI are more effective at explaining away label noise. Doing so we have found that the most helpful PI is the one that allows the networks to separate correct from mislabeled examples in feature space, but also enable an easier learning shortcut to memorize the mislabeled examples. We have also shown that methods which use appropriate PI to explain away label noise, can be combined with other state-of-the-art methods to remove noise and achieve cumulative gains. Exploring this direction further is a promising avenue for future work. Our insights show that the use of PI is a promising avenue of research to fight against label noise. Our insights further highlight that collecting the right PI in datasets requires some care to enable the learning of effective shortcuts.

Acknowledgements

We thank Jannik Kossen for helpful comments on this work. We also thank Josip Djolonga and Joan Puigcerver for helpful discussions related to infrastructure and data processing.

References

Bai, Y., Yang, E., Han, B., Yang, Y., Li, J., Mao, Y., Niu, G., and Liu, T. Understanding and improving early stopping for learning with noisy labels. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Baldock, R. J. N., Maennel, H., and Neyshabur, B. Deep learning through the lens of example difficulty. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Cheng, H., Zhu, Z., Li, X., Gong, Y., Sun, X., and Liu, Y. Learning with instance-dependent label noise: A sample sieve approach. In *International Conference on Learning Representations (ICLR)*, 2021.

Collier, M., Mustafa, B., Kokiopoulou, E., Jenatton, R., and Berent, J. Correlated input-dependent label noise in large-scale image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Collier, M., Jenatton, R., Kokiopoulou, E., and Berent, J. Transfer and marginalize: Explaining away label noise with privileged information. In *International Conference on Machine Learning (ICML)*, 2022.

D'Amour, A., Heller, K. A., Moldovan, D., Adlam, B., Alipanahi, B., Beutel, A., Chen, C., Deaton, J., Eisenstein, J., Hoffman, M. D., Hormozdiari, F., Houlsby, N., Hou, S., Jerfel, G., Karthikesalingam, A., Lucic, M., Ma, Y., McLean, C. Y., Mincu, D., Mitani, A., Montanari, A., Nado, Z., Natarajan, V., Nielson, C., Osborne, T. F., Raman, R., Ramasamy, K., Sayres, R., Schrouff, J., Seneviratne, M., Sequeira, S., Suresh, H., Veitch, V., Vladymyrov, M., Wang, X., Webster, K., Yadlowsky, S., Yun, T., Zhai, X., and Sculley, D. Underspecification presents challenges for credibility in modern machine learning. *CoRR*, abs/2011.03395, 2020. URL https://arxiv.org/abs/2011.03395.

Deng, J., Dong, W., Socher, R., Li, L. J., Kai, L., and Li, F. F. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009.

Geirhos, R., Jacobsen, J., Michaelis, C., Zemel, R. S., Brendel, W., Bethge, M., and Wichmann, F. A. Shortcut learning in deep neural networks. *Nat. Mach. Intell.*, 2 (11):665–673, 2020.

Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I. W., and Sugiyama, M. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Hinton, G. E., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

Kallus, N., Puli, A. M., and Shalit, U. Removing hidden confounding by experimental grounding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009.

Lambert, J., Sener, O., and Savarese, S. Deep learning under privileged information using heteroscedastic dropout. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Li, J., Socher, R., and Hoi, S. C. H. Dividemix: Learning with noisy labels as semi-supervised learning. In *International Conference on Learning Representations (ICLR)*, 2020.

Liu, S., Niles-Weed, J., Razavian, N., and Fernandez-Granda, C. Early-learning regularization prevents memorization of noisy labels. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Liu, S., Zhu, Z., Qu, Q., and You, C. Robust training under label noise by over-parameterization. In *International Conference on Machine Learning (ICML)*, 2022.

Lopez-Paz, D., Bottou, L., Schölkopf, B., and Vapnik, V. Unifying distillation and privileged information. In *International Conference on Learning Representations (ICLR)*, 2016.

Maennel, H., Alabdulmohsin, I. M., Tolstikhin, I. O., Baldock, R. J. N., Bousquet, O., Gelly, S., and Keysers, D. What do neural networks learn when trained with random labels? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Makar, M., Packer, B., Moldovan, D., Blalock, D., Halpern, Y., and D'Amour, A. Causally motivated shortcut removal using auxiliary labels. In *International Conference on Artificial Intelligence and Statistics, (AISTATS)*, 2022.

Minderer, M., Bachem, O., Houlsby, N., and Tschannen, M. Automatic shortcut removal for self-supervised representation learning. In *International Conference on Machine Learning (ICML)*, 2020.

Nado, Z., Band, N., Collier, M., Djolonga, J., Dusenberry, M. W., Farquhar, S., Feng, Q., Filos, A., Havasi, M., Jenatton, R., et al. Uncertainty baselines: Benchmarks for uncertainty & robustness in deep learning. *arXiv preprint arXiv:2106.04015*, 2021.

Patrini, G., Rozza, A., Menon, A. K., Nock, R., and Qu, L. Making deep neural networks robust to label noise: A loss correction approach. In *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*, 2017.

Pearl, J. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2009.

Peterson, J. C., Battleday, R. M., Griffiths, T. L., and Russakovsky, O. Human uncertainty makes classification more robust. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, 2019.

Pezeshki, M., Kaba, S., Bengio, Y., Courville, A. C., Precup, D., and Lajoie, G. Gradient starvation: A learning proclivity in neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Prabhu, V., Yenamandra, S., Singh, A., and Hoffman, J. Adapting self-supervised vision transformers by probing attention-conditioned masking consistency. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763. PMLR, 2021.

Rolnick, D., Veit, A., Belongie, S., and Shavit, N. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017.

Sheng, V. S., Provost, F., and Ipeirotis, P. G. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2008.

Snow, R., O'Connor, B., Jurafsky, D., and Ng, A. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, October 2008.

Song, H., Kim, M., Park, D., and Lee, J. Learning from noisy labels with deep neural networks: A survey. *CoRR*, abs/2007.08199, 2020.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*, 2016.

Toneva, M., Sordoni, A., des Combes, R. T., Trischler, A., Bengio, Y., and Gordon, G. J. An empirical study of example forgetting during deep neural network learning. In *nternational Conference on Learning Representations (ICLR)*, 2019.

Vapnik, V. and Izmailov, R. Learning using privileged information: similarity control and knowledge transfer. *J. Mach. Learn. Res.*, 16:2023–2049, 2015.

Vapnik, V. and Vashist, A. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22 (5-6), 2009.

Veit, A., Alldrin, N., Chechik, G., Krasin, I., Gupta, A., and Belongie, S. J. Learning from noisy large-scale datasets with minimal supervision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Veitch, V., D'Amour, A., Yadlowsky, S., and Eisenstein, J. Counterfactual invariance to spurious correlations in text classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Wei, J., Zhu, Z., Cheng, H., Liu, T., Niu, G., and Liu, Y. Learning with noisy labels revisited: A study using real-world human annotations. In *International Conference on Learning Representations (ICLR)*, 2022.

Yang, H., Zhou, J. T., Cai, J., and Ong, Y. MIML-FCN+: multi-instance multi-label learning via fully convolutional networks with privileged information. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*, 2017.

Yang, S., Sanghavi, S., Rahmanian, H., Bakus, J., and Vishwanathan, S. Toward understanding privileged features distillation in learning-to-rank. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017.

Zhao, P., Yang, Y., and He, Q.-C. High-dimensional linear regression via implicit regularization. *Biometrika*, 109 (4):1033–1046, feb 2022. doi: 10.1093/biomet/asac010. URL https://doi.org/10.1093%2Fbiomet%2Fasac010.

Appendices

This appendix is organized as follows: In Appendix A we describe the relabelling process done to generate ImageNet-PI. In Appendix B we describe in depth the experimental details of our experiments and our hyperparameter tuning strategy. Appendix C replicates our main findings in the low-noise version of the PI datasets. Appendix D discusses and ablates the effect of early stopping in our experiments. Appendix E provides additional results of our ablation studies on other datasets. Appendix F and Appendix G give further details about TRAM++ and AFM++, respectively. And, finally, Appendix I and Appendix J describe in depth the experimental setup used to combine SOP and HET with TRAM, respectively.

A. ImageNet-PI

ImageNet-PI is a re-labelled version of the standard ILSVRC2012 ImageNet dataset in which the labels are provided by a collection of 16 deep neural networks with different architectures pre-trained on the standard ILSVRC2012. Specifically, the pre-trained models are downloaded from `tf.keras.applications`[7] and consist of: `ResNet50V2`, `ResNet101V2`, `ResNet152V2`, `DenseNet121`, `DenseNet169`, `DenseNet201`, `InceptionResNetV2`, `InceptionV3`, `MobileNet`, `MobileNetV2`, `MobileNetV3Large`, `MobileNetV3Small`, `NASNetMobile`, `VGG16`, `VGG19`, `Xception`.

During the re-labelling process, we do not directly assign the maximum confidence prediction of each of the models, but instead, for each example, we sample a random label from the predictive distribution of each model on that example. Furthermore, to regulate the amount of label noise introduced when relabelling the dataset, ImageNet-PI allows the option to use stochastic temperature-scaling to increase the entropy of the predictive distribution. The stochasticity of this process is controlled by a parameter β which controls the inverse scale of a Gamma distribution (with shape parameter $\alpha = 1.0$), from which the temperature values are sampled, with a code snippet looking as follows:

```
# Get the predictive distribution of the model annotator.
pred_dist = model.predict(...)

# Sample the temperature.
temperature = tf.random.gamma(
    [tf.shape(pred_dist)[0]],
    alpha=tf.constant([1.]),
    beta=tf.constant([beta_parameter]))

# Compute the new predictive distribution.
log_probs = tf.math.log(pred_dist) / temperature
new_pred_dist = tf.nn.softmax(log_probs)

# Sample from the new predictive distribution.
class_predictions = tf.random.categorical(tf.math.log(new_pred_dist), 1)[:,0]
```

Intuitively, smaller values of β translate to higher temperature values as shown in Figure 5, which leads to higher levels of label noise as softmax comes closer to uniform distribution for high temperatures.

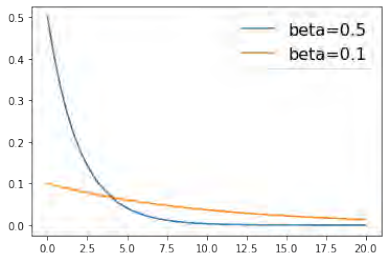

Figure 5. The effect of parameter β in sampling temperatures.

[7] `https://www.tensorflow.org/api_docs/python/tf/keras/applications`

This re-labelling process can produce arbitrarily noisy labels whose distribution is very far from being symmetrical, i.e., not all mis-classifications are equally likely. For example, it is more likely that similar dog breeds get confused among each other, but less likely that a 'dog' gets re-labeled as a 'chair'.

The PI in this dataset comes from the confidences of the models on the sampled label, their parameter count, and their test accuracy on the clean test distribution. These PI features are a good proxy for the expected reliability of each of the models. In our dataset release, we will provide the following files:

- **labels-train.csv, labels-validation.csv** These files contain the new (noisy) labels for the training and validation set respectively. The new labels are provided by the pre-trained annotator models. Each file provides the labels in CSV format:

```
<image_id>,<label_1>,<label_2>,...,<label_16>
```

- **confidences-train.csv, confidences-validation.csv** These files contain the confidence of each annotator model in its annotation; both for the training set and the validation set respectively. Each file provides the confidences in CSV format:

```
<image_id>,<confidence_1>,<confidence_2>,...,<confidence_16>
```

- **annotator-features.csv** This file contains the annotator features (i.e., meta-data about the model annotators themselves) in CSV format (16 rows; one for each model annotator):

```
<model_accuracy>,<number_of_model_parameters>
```

In particular, we will provide two standardized sampled annotations obtained by applying the temperature sampling process discussed above: one with $\beta = 0.1$ corresponding to high label noise and one with $\beta = 0.5$ corresponding to low label noise.

B. Experimental details

We build upon the implementations and hyperparameters from the open source Uncertainty Baselines codebase (Nado et al., 2021). All results in the paper are reported based on 5 random seeds.

B.1. Dataset-specific training settings

B.1.1. CIFAR

All CIFAR models are trained using a SGD optimizer with 0.9 Nestrov momentum for 90 epochs with a batch size of 256. We sweep over an initial learning rate of $\{0.01, 0.1\}$ with the learning rate decayed by a factor of 0.2 after 27, 54 and 72 epochs. We sweep over an L2 regularization parameter of $\{0.00001, 0.0001, 0.001\}$. Following Nado et al. (2021), we use a Wide ResNet model architecture with model-width multiplier of 10 and a model-depth of 28.

Unless specified otherwise, for TRAM and AFM models, we set the PI tower width to be 1024 as this was the default parameter in Collier et al. (2022). We use the same architecture for the Distillation (PI) teacher. This controls the size of the subnetwork which integrates the PI which is parameterized as a concatenation of the pre-processed PI (and then passed through a Dense + ReLU layer) and the representation of the non-PI inputs by the base Wide ResNet model, followed by a Dense + ReLU layer, a residual connection and finally a concatenation of the joint feature space with the non-PI representation. The number of units in the Dense layers is controlled by the "PI tower width".

For distillation models we uniformly sample over a temperature interval of $[0.5, 10]$. For CIFAR-10N and CIFAR-100N we split the original training set into a training and a validation set; 98% of the examples are used for training and the remaining 2% used as a validation set. Due to the smaller size of the CIFAR-10H training set (which following Collier et al. (2022) is actually the original CIFAR test set), 96% of the original training set is used as a training set with the remaining 4% used as a validation set.

For TRAM++ and where relevant for AFM, we search over a no-PI loss weight of $\{0.1, 0.5\}$, a PI tower width of $\{512, 1024, 2048, 4096\}$ and a random PI length of $\{8, 14, 28\}$. For heteroscedastic CIFAR models, we set the number of

factors for the low-rank component of the heteroscedastic covariance matrix (Collier et al., 2021) to be 3 for CIFAR-10H and CIFAR-10N, and 6 for CIFAR-100N and search over $\{0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 2.0, 3.0, 5.0\}$ for the heteroscedastic temperature.

B.1.2. IMAGENET-PI

ImageNet models are trained using a SGD optimizer with 0.9 Nestrov momentum for 90 epochs with a batch size of 128. We set the initial learning rate of 0.05 with the learning rate decayed by a factor of 0.1 after 30, 60 and 80 epochs. We sweep over an L2 regularization parameter of $\{0.00001, 0.0001\}$. We use a ResNet-50 model architecture.

For TRAM and AFM models by default we set the PI tower width to be 2048, with the same parameterization of the PI tower as for the CIFAR models. For distillation models we set the distillation temperature to be 0.5. We use 1% of the original ImageNet training set as a validation set.

For TRAM++ and where relevant for AFM, we set the no-PI loss weight of to 0.5 and use random PI length of 30. For heteroscedastic models we set the number of factors for the low-rank component of the heteroscedastic covariance matrix to be 15 and search over $\{0.75, 1.0, 1.5, 2.0, 3.0\}$ for the heteroscedastic temperature.

B.2. Hyperparameter tuning strategy

Unless otherwise indicated, we report the test-set accuracy at the hyperparameters determined by the $\arg\max$ of best validation set accuracy (where the number of epochs are considered to be part of the set to be maximized over). The validation set used has noisy labels generated by the same process as the training set. This implements a realistic and noisy hyperparameter search with early stopping that we believe most closely replicates what is possible in real-world scenarios where a clean validation set may be unavailable. However, other papers report test-set metrics determined by a hyperparameter sweep assuming the availability of a clean validation set and/or without early stopping, which can have a large impact on the reported test-set metrics (see Appendix D for results computed in this way).

C. Results on low-noise settings

In the main text, we always reported results for the high-noise settings of each of the datasets. However, we now show that all our findings from Table 1 also apply in the low-noise setting.

Table 3. Test accuracy of several methods trained using different features as PI ***on the low-noise versions of the datasets*** (baselines in gray and *italics* do not use PI). Here, Original denotes the standard PI of the dataset, Indicator a binary signal that separates clean from noisy examples, Labels the one-hot encoded labels, and Near-optimal a synthetic feature consisting on giving the annotator label to those examples that are miss-annotated and a zero-vector otherwise. **Bold numbers** represent significant maximum values across PI features where significance means p-value < 0.05.

		Original	Indicator	Labels	Near-optimal
CIFAR10-H (uniform)	*no-PI*	85.8±0.3	85.8±0.3	85.8±0.3	85.8±0.3
	Distillation (no-PI)	82.7±0.0	82.7±0.0	82.7±0.0	82.7±0.0
	TRAM	85.1±0.2	86.3±0.3	47.3±1.7	**87.6±0.1**
	Approximate FM	85.9±0.1	86.8±0.2	38.1±0.5	**87.4±0.1**
	Distillation (PI)	81.0±0.0	**83.2±0.3**	81.1±0.3	**83.4±0.3**
CIFAR10-N (uniform)	*no-PI*	88.3±0.4	88.3±0.4	88.3±0.4	88.3±0.4
	Distillation (no-PI)	85.4±0.0	85.4±0.0	85.4±0.0	85.4±0.0
	TRAM	88.7±0.6	92.2±0.3	54.2±0.1	**93.7±0.1**
	Approximate FM	88.6±0.6	93.0±0.2	37.4±1.7	**94.0±0.1**
	Distillation (PI)	88.1±0.0	**88.8±0.6**	**88.8±0.7**	**88.6±0.4**
ImageNet-PI (low-noise)	*no-PI*	68.3±0.3	68.3±0.3	68.3±0.3	68.3±0.3
	Distillation (no-PI)	67.9±0.3	67.9±0.3	67.9±0.3	67.9±0.3
	TRAM	**69.7±0.3**	**69.8±0.2**	55.9±0.5	66.8±0.1
	Approximate FM	**70.6±0.2**	**70.7±0.3**	24.9±13.6	**70.8±0.2**
	Distillation (PI)	**67.2±0.2**	**67.4±0.2**	**52.5±26.3**	**67.2±0.7**

D. Effect of early stopping

As early stopping is one of the strongest baselines against label noise, in all our experiments we held out a small portion of the noisy training set and reported clean test accuracy at the epoch with the best validation accuracy. However, to make sure that our findings do not depend on the use of early stopping, or the amount of label noise in the validation set, we now present a reproduction of the results in Table 1 when either disabling early stopping or using a clean validation set to perform early stopping and hyperparameter tuning.

D.1. No early stopping

Table 4 shows the results of our benchmark without using early stopping. In general, we observe that without early stopping most baselines perform significantly worse as they overfit more to the noisy labels. In this regard, since one of the main benefits of PI is that it prevents memorization of the noisy labels, we see that without early stopping the relative improvement of the PI techniques with respect to their no-PI baselines is much larger.

Table 4. Test accuracy of several methods trained using different features as PI ***not using early stopping*** (baselines in gray and *italics* do not use PI). Here, Original denotes the standard PI of the dataset, Indicator a binary signal that separates clean from noisy examples, Labels the one-hot encoded labels, and Near-optimal a synthetic feature consisting on giving the annotator label to those examples that are miss-annotated and a zero-vector otherwise. **Bold numbers** represent significant maximum values across PI features where significance means p-value < 0.05.

		Original	Indicator	Labels	Near-optimal
CIFAR-10H (worst)	*no-PI*	42.6±0.3	42.6±0.3	42.6±0.3	42.6±0.3
	Distillation (no-PI)	45.2±0.1	45.2±0.1	45.2±0.1	45.2±0.1
	TRAM	59.2±0.2	46.5±0.6	39.9±0.3	**77.4±0.1**
	Approximate FM	61.2±0.7	39.3±0.6	10.0±0.0	**79.3±0.3**
	Distillation (PI)	45.2±0.0	**46.3±0.1**	45.7±0.1	**46.4±0.1**
CIFAR-10N (worst)	*no-PI*	67.7±0.6	67.7±0.6	67.7±0.6	67.7±0.6
	Distillation (no-PI)	71.4±0.2	71.4±0.2	71.4±0.2	71.4±0.2
	TRAM	67.0±0.4	79.2±0.9	51.6±0.2	**91.9±0.2**
	Approximate FM	69.8±0.5	89.6±0.1	10.0±0.0	**92.3±0.2**
	Distillation (PI)	71.1±0.2	**73.1±0.3**	70.9±0.3	70.9±0.2
CIFAR-100N	*no-PI*	55.8±0.2	55.8±0.2	55.8±0.2	55.8±0.2
	Distillation (no-PI)	58.6±0.1	58.6±0.1	58.6±0.1	58.6±0.1
	TRAM	56.4±0.3	56.0±0.4	34.9±0.2	**67.5±0.3**
	Approximate FM	58.9±0.3	65.4±0.3	4.1±0.1	**70.8±0.4**
	Distillation (PI)	58.9±0.2	58.9±0.3	56.8±0.2	**60.4±0.4**
ImageNet-PI (high-noise)	*no-PI*	47.7±0.8	47.7±0.8	47.7±0.8	47.7±0.8
	Distillation (no-PI)	50.4±0.8	50.4±0.8	50.4±0.8	50.4±0.8
	TRAM	53.3±0.4	53.5±0.4	41.0±0.8	**56.5±0.3**
	Approximate FM	55.0±0.4	55.5±0.3	0.4±0.1	**58.3±0.1**
	Distillation (PI)	**50.9±0.4**	**50.6±0.2**	39.1±5.1	18.0±24.7

D.2. Clean validation set

Most of the datasets we studied have a significant amount of label noise in their training set. In this regard, the small validation set we hold out from the training set is also very noisy, which can affect the performance of early stopping and hyperparameter tuning. For this reason, we also provide results in Table 5 in which we use the clean labels from the validation set for hyperparameter tuning and early stopping. As we can see, most methods perform better in this regime, although our main findings about how the PI properties affect performance are still valid.

Table 5. Test accuracy of several methods trained using different features as PI ***using a clean validation set to select the best hyperparameters*** (baselines in gray and *italics* do not use PI). Here, Original denotes the standard PI of the dataset, Indicator a binary signal that separates clean from noisy examples, Labels the one-hot encoded labels, and Near-optimal a synthetic feature consisting on giving the annotator label to those examples that are miss-annotated and a zero-vector otherwise. **Bold numbers** represent significant maximum values across PI features where significance means p-value < 0.05.

		Original	Indicator	Labels	Near-optimal
CIFAR-10H (worst)	*no-PI*	53.2±1.0	53.2±1.0	53.2±1.0	53.2±1.0
	Distillation (no-PI)	53.4±1.0	53.4±1.0	53.4±1.0	53.4±1.0
	TRAM	67.7±0.1	64.9±0.6	39.7±0.3	**77.4±0.1**
	Approximate FM	70.6±0.4	66.7±2.1	29.5±0.5	**79.1±0.2**
	Distillation (PI)	**53.9±0.4**	**53.3±0.6**	**53.4±0.4**	51.6±0.0
CIFAR-10N (worst)	*no-PI*	81.4±0.5	81.4±0.5	81.4±0.5	81.4±0.5
	Distillation (no-PI)	82.9±0.4	82.9±0.4	82.9±0.4	82.9±0.4
	TRAM	81.9±0.3	89.1±0.3	51.6±0.1	**91.1±0.1**
	Approximate FM	82.0±0.3	91.2±0.3	22.6±0.2	**92.3±0.2**
	Distillation (PI)	**80.8±0.3**	**80.7±0.5**	**81.1±0.4**	**80.8±0.2**
CIFAR-100N	*no-PI*	60.8±0.2	60.8±0.2	60.8±0.2	60.8±0.2
	Distillation (no-PI)	60.8±0.1	60.8±0.1	60.8±0.1	60.8±0.1
	TRAM	60.6±0.3	63.3±0.2	34.8±0.4	**67.3±0.3**
	Approximate FM	60.2±0.1	67.8±0.3	20.1±0.3	**70.9±0.2**
	Distillation (PI)	61.1±0.2	**61.9±0.2**	60.5±0.2	61.5±0.3
ImageNet-PI (high-noise)	*no-PI*	48.3±0.1	48.3±0.1	48.3±0.1	48.3±0.1
	Distillation (no-PI)	50.5±0.7	50.5±0.7	50.5±0.7	50.5±0.7
	TRAM	53.3±0.3	53.8±0.7	40.7±0.8	**56.5±0.2**
	Approximate FM	55.6±0.3	55.5±0.4	0.8±0.2	**58.2±0.1**
	Distillation (PI)	**51.0±0.4**	**50.7±0.3**	39.1±4.4	**27.6±22.7**

E. More results

In this section, we provide complete results for the experiments in the main paper using other datasets and algorithms with the main findings.

E.1. Training dynamics

In Figure 2 we provided a detailed analysis of the dynamics of TRAM on CIFAR-10H with different PI features. We now show results for TRAM on CIFAR-10N and CIFAR-100N (see Figure 6 and Figure 7, respectively). We also show results for AFM on CIFAR-10H, CIFAR-10N, and CIFAR-100N (see Figure 8, Figure 9 and Figure 10, respectively).

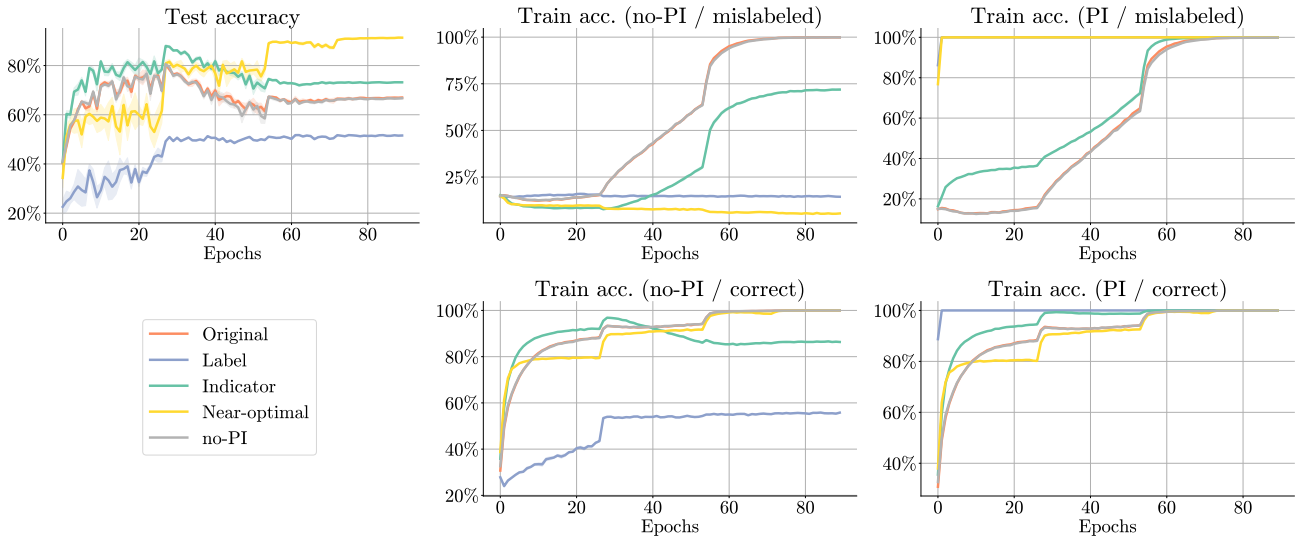

Figure 6. Dynamics of TRAM on CIFAR-10N with different PI features. **Top left**: Test accuracy. **Top center**: Train accuracy on noisy examples evaluated at the no-PI head, **Top right**: Train accuracy on noisy examples evaluated at the PI-head. **Bottom center**: Train accuracy on clean examples evaluated at the no-PI head. **Bottom right**: Train accuracy of clean examples evaluated at the PI head.

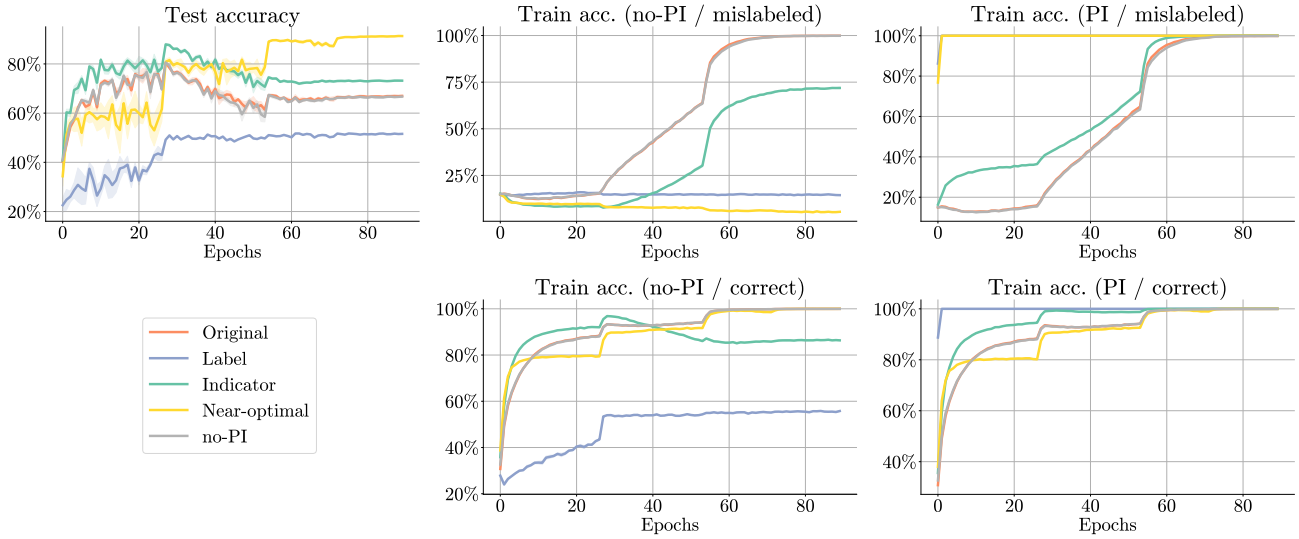

Figure 7. Dynamics of TRAM on CIFAR-100N with different PI features. **Top left**: Test accuracy. **Top center**: Train accuracy on noisy examples evaluated at the no-PI head, **Top right**: Train accuracy on noisy examples evaluated at the PI-head. **Bottom center**: Train accuracy on clean examples evaluated at the no-PI head. **Bottom right**: Train accuracy of clean examples evaluated at the PI head.

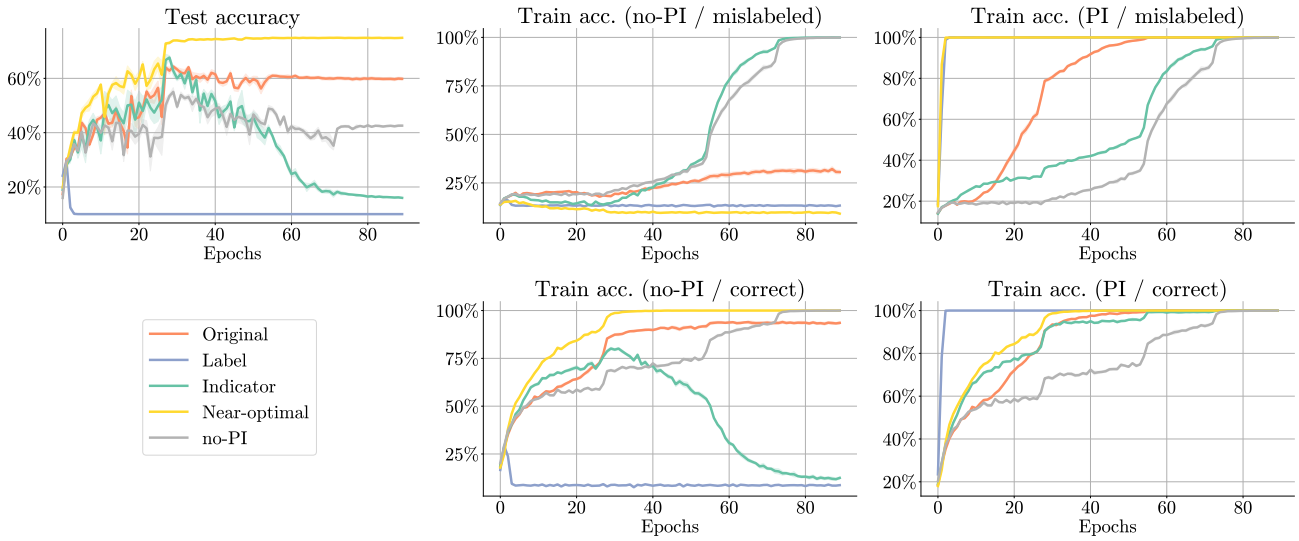

Figure 8. Dynamics of AFM on CIFAR-10H with different PI features. **Top left**: Test accuracy. **Top center**: Train accuracy on noisy examples evaluated with marginalization, **Top right**: Train accuracy on noisy examples evaluated at the PI-head. **Bottom center**: Train accuracy on clean examples evaluated with marginalization. **Bottom right**: Train accuracy of clean examples evaluated at the PI head.

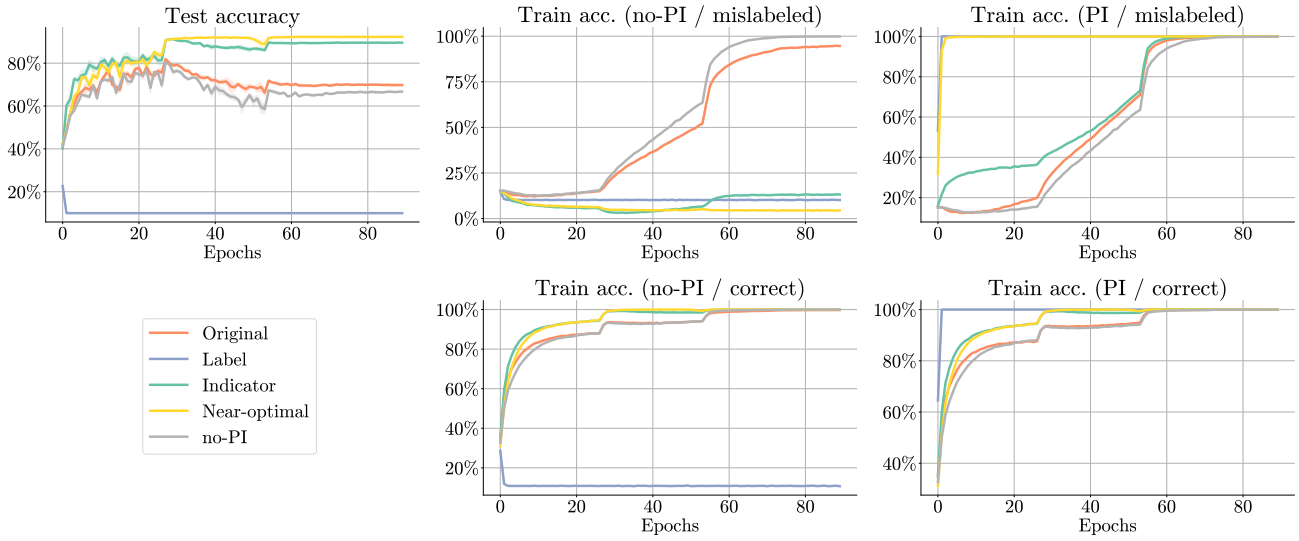

Figure 9. Dynamics of AFM on CIFAR-10N with different PI features. **Top left**: Test accuracy. **Top center**: Train accuracy on noisy examples evaluated with marginalization, **Top right**: Train accuracy on noisy examples evaluated at the PI-head. **Bottom center**: Train accuracy on clean examples evaluated with marginalization. **Bottom right**: Train accuracy of clean examples evaluated at the PI head.

E.2. Feature extractor size

We replicate the results in Figure 3 for other settings with the same findings. In particular, we show results on CIFAR-10N and CIFAR-100N (see Figure 11 and Figure 12, respectively).

E.3. PI head size

We replicate the results in Figure 4 on CIFAR-10N and CIFAR-100N (see Figure 13 and Figure 14, respectively). In this case, however, we observe no clear trend in the results, probably due to the fact that the original PI on these datasets is not good enough for TRAM and AFM to shine (cf. Table 1). In this regard, increasing the PI head size does not lead to better

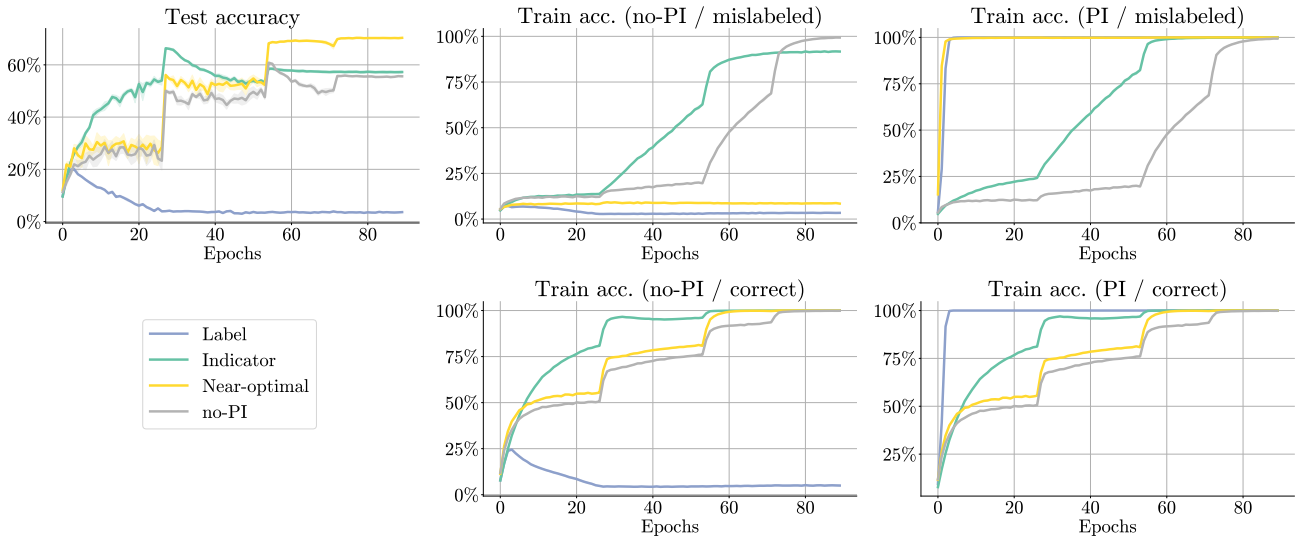

Figure 10. Dynamics of AFM on CIFAR-100N with different PI features. **Top left**: Test accuracy. **Top center**: Train accuracy on noisy examples evaluated with marginalization, **Top right**: Train accuracy on noisy examples evaluated at the PI-head. **Bottom center**: Train accuracy on clean examples evaluated with marginalization. **Bottom right**: Train accuracy of clean examples evaluated at the PI head.

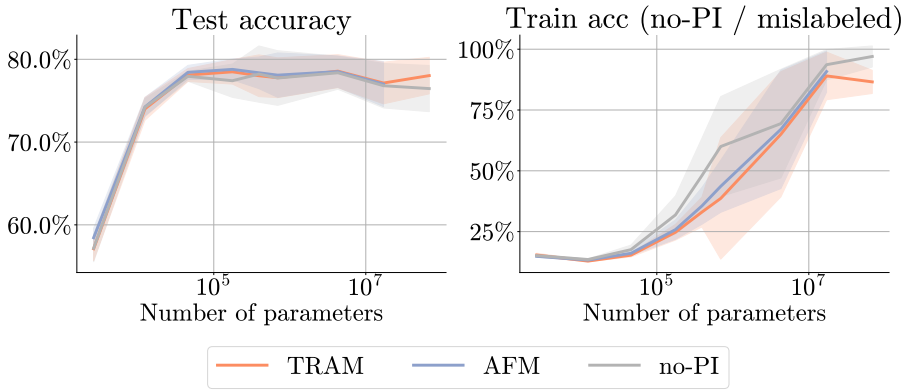

Figure 11. Performance of different PI baselines on CIFAR-10N when increasing the parameter count of their feature extractor keeping the PI tower fixed. Larger models suffer from overfitting as they tend to use their larger capacity to overfit to noisy examples, discouraging the model from exploiting the PI.

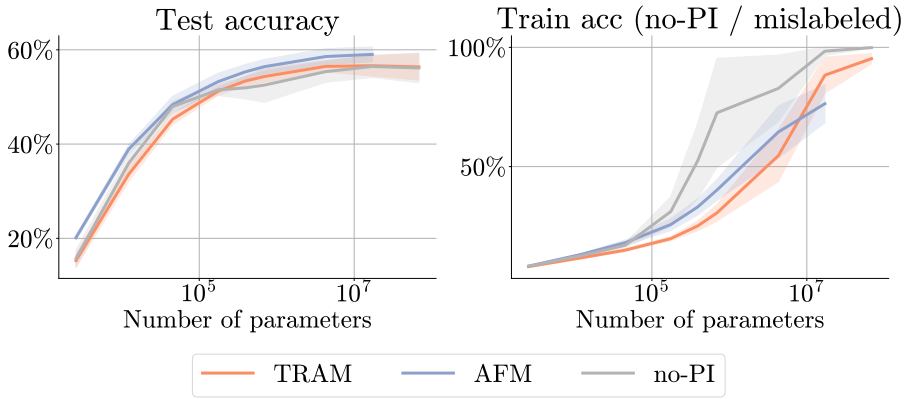

Figure 12. Performance of different PI baselines on CIFAR-100N when increasing the parameter count of their feature extractor keeping the PI tower fixed. Larger models suffer from overfitting as they tend to use their larger capacity to overfit to noisy examples, discouraging the model from exploiting the PI.

performance as there is nothing to extract from the PI.

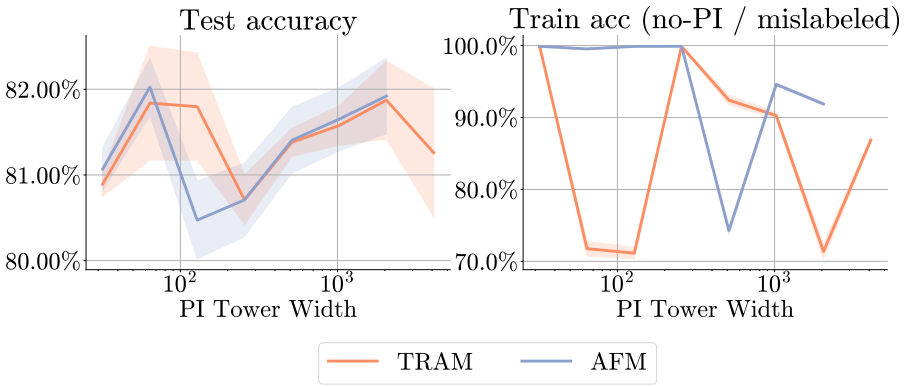

Figure 13. Performance of different PI baselines on CIFAR-10N when increasing the PI head size. A larger PI head size incentivizes the model to memorize the noisy examples using the PI making more use of the PI as a shortcut.

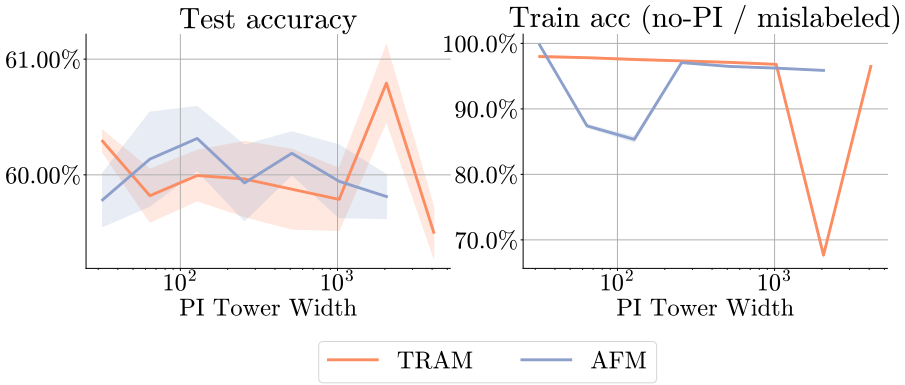

Figure 14. Performance of different PI baselines on CIFAR-100N when increasing the PI head size. A larger PI head size incentivizes the model to memorize the noisy examples using the PI making more use of the PI as a shortcut.

F. Design details of TRAM++

We now give the design details for TRAM++, the improved version of TRAM which appends a unique random PI vector to the original PI. In particular, we followed the same tuning strategy as in the rest of the TRAM experiments in the paper and we also tuned the parameter λ that weighs the losses of the two heads, i.e.,

$$\min_{\phi,\pi,\psi} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{a},\tilde{y})} \left[\ell(\pi(\phi(\boldsymbol{x}),\boldsymbol{a}),\tilde{y}) + \lambda\, \ell(\psi(\phi(\boldsymbol{x})),\tilde{y})\right]. \tag{2}$$

Collier et al. (2022) suggested that the gradients of the no-PI head do not affect the updates of the feature extraction, and thus λ could be folded directly into the tuning of the global learning rate of TRAM. However, in our experiments, we found that tuning λ given a fixed number of epochs can lead to significant gains in performance, as it can slow down training of the no-PI head. As seen in Figure 15, increasing λ has the same effect as increasing the learning rate of the no-PI head, and a sweet spot exists for values of $\lambda < 1$ in which the no-PI head trains fast enough to fit the clean examples, but avoids learning all the noisy ones.

In general, λ was not tuned in any of the other experiments, in order to remain as close as possible to the original TRAM implementation. However, for the TRAM++ experiments, which aimed to achieve the best possible performance out of TRAM, λ was tuned.

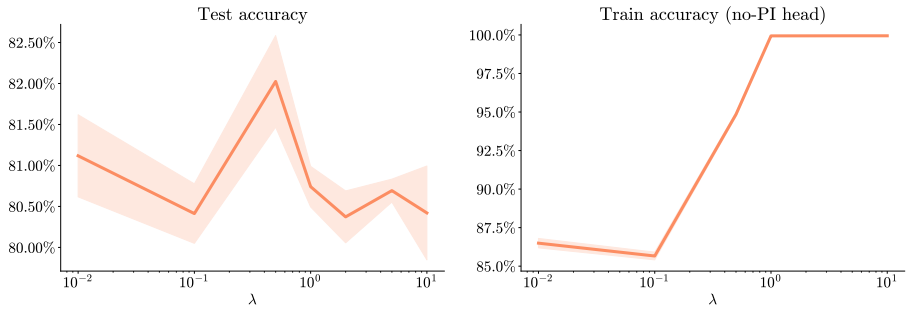

Figure 15. Performance of TRAM for different values of the loss weight λ in CIFAR-10N. The optimal λ is such the one that strikes a good balance between training the clean examples, while slowing down significantly the overfitting to the noisy ones.

G. Design of AFM++

In Section 4.2, we have seen that appending random PI that uniquely identifies each example to the original PI can sometimes induce beneficial shortcuts in TRAM++. We now test the same strategy applied to AFM, and design AFM++, an augmented version of AFM with additional random PI. Table 6 shows the results of our experiments where we see that AFM++ also clearly improves over "vanilla" AFM. Again, the improvements are greater in those datasets where overfitting is a bigger issue in the first place.

Table 6. Performance comparison of no-PI, AFM and AFM++ on the different PI datasets.

	no-PI	AFM	AFM++
CIFAR-10H (worst)	55.0±1.5	64.0±0.6	**68.2±0.6**
CIFAR-10N (worst)	80.6±0.2	82.0±0.3	**84.6±0.2**
CIFAR-100N	60.4±0.5	60.0±0.2	**61.9±0.2**
ImageNet-PI (high-noise)	47.7±0.8	**55.6±0.3**	**55.0±0.6**

H. Combination of TRAM with label smoothing

We also evaluate the combination of TRAM with label smoothing (LS). In particular, we follow the standard label smoothing procedure and add the label smoothing hyperparameter to the hyperparameters swept over in Table 1. More specifically, we sweep over label smoothing of 0.2, 0.4, 0.6 and 0.8 and select the optimal hyperparameter setting following the same procedure as all experiments in the paper. The results are given in Table 7.

We observe that on all datasets, adding label smoothing to the TRAM method leads to performance improvements, demonstrating that TRAM can be successfully combined with label smoothing. More generally, this observation strengthens the point that TRAM and TRAM++ are compatible and yield additive performance gains when combined with widely used methods developed for noisy labels.

Table 7. Performance comparison of no-PI, Label smoothing (LS), TRAM TRAM + LS on different PI datasets.

	no-PI	TRAM	LS	TRAM+LS
CIFAR-10H (worst)	$55.0_{\pm 1.5}$	$64.9_{\pm 0.8}$	$59.9_{\pm 1.5}$	$\mathbf{65.4_{\pm 0.9}}$
CIFAR-10N (worst)	$80.6_{\pm 0.2}$	$80.5_{\pm 0.5}$	$80.5_{\pm 0.4}$	$\mathbf{82.4_{\pm 0.2}}$
CIFAR-100N	$60.4_{\pm 0.5}$	$59.7_{\pm 0.3}$	$60.0_{\pm 0.46}$	$\mathbf{61.9_{\pm 0.3}}$

I. Experimental details for SOP and TRAM+SOP

As we have established in Section 5, the combination of TRAM and SOP has the potential to achieve cumulative gains in robustness to label noise. TRAM, with its original PI, has been shown to improve performance on datasets with dense noise, such as CIFAR-10H (worst), compared to a model with no PI. However, the PI may not always be explanatory of the noise and even if it is, it may not fully explain away all of the noise. Additionally, the feature extractor and subsequent layers of the model may still be susceptible to noise, even when the PI is able to explain away the noise.

On the other hand, SOP has been shown to work well for sparsely distributed noise and operates on the principle of modeling out the noise, which is distinct from the method used by TRAM. As these principles are complementary to one another, we propose to combine the advantages of both methods to achieve cumulative gains.

As highlighted in Section 5, the combination of TRAM+SOP consists of two main steps: pre-training with TRAM and fine-tuning with SOP. Our implementation of TRAM used regular TRAM with a few enhancements from TRAM++, such as random PI and a larger PI head size. It is important to note that our experiments were conducted using our own implementation of SOP and, although it incorporated the SOP method and was sanity-checked with the original authors of the paper, our experimental baseline environment and search space were different from theirs. As a result, the test accuracy on the CIFAR-N datasets may be lower than the results reported in the original SOP paper. However, the primary objective of these experiments was to explore whether TRAM + SOP can achieve cumulative gains over the respective implementations of TRAM and SOP alone and our results support this hypothesis.

In our experiments, both the SOP and TRAM+SOP models were trained for a total of 120 epochs, with a learning rate schedule that decayed at epochs 40, 80 and 110. We employed the SGD with Nesterov momentum for TRAM and regular momentum for SOP as in Liu et al. (2022). For a detailed description of the SOP parameters, we refer the reader to the original SOP paper. It is important to note that the results presented here for the TRAM+SOP method do not include all proposed enhancements in Liu et al. (2022). Further gains in performance may be achievable by incorporating these advancements and jointly optimizing the hyperparameter space for both the TRAM and SOP pretraining and fine-tuning stages.

J. Experimental details for TRAM+HET

TRAM+HET consists of a simple two-headed TRAM model in which the last linear layer of each of the two heads has been substituted by a heteroscedastic linear layer (Collier et al., 2021). In these experiments, we thus also sweep over the temperature of the heteroscedastic layers. A similar method was already proposed in Collier et al. (2022), under the name Het-TRAM, but here we also make use of our insights and allow the model to make use of random PI on top of the original PI features. Interestingly, contrary to what happened with TRAM+SOP, the addition of random PI, i.e., TRAM++, did not always yield performance improvements using TRAM+HET. Instead, depending on the dataset (see Table 8) we observe that the use of random PI can sometimes hurt the final performance of the models (e.g., as in CIFAR-10H). We conjecture

this might be due to the TRAM+HET models using the random PI to memorize the clean labels as well. Understanding why this happens only when using heteroscedastic layers is an interesting avenue for future work.

Table 8. Performance comparison of TRAM, TRAM++, HET, TRAM+HET (without additional random PI), and TRAM+HET (with additional random PI) on the different PI datasets.

	TRAM	TRAM++	HET	TRAM+HET (w/o random)	TRAM+HET (+random)
CIFAR-10H (worst)	$64.9_{\pm 0.8}$	$66.8_{\pm 0.3}$	$50.8_{\pm 1.4}$	$\mathbf{67.7_{\pm 0.7}}$	$56.5_{\pm 0.7}$
CIFAR-10N (worst)	$80.5_{\pm 0.5}$	$83.9_{\pm 0.2}$	$81.9_{\pm 0.4}$	$82.0_{\pm 0.3}$	$\mathbf{83.5_{\pm 0.1}}$
CIFAR-100N	$59.7_{\pm 0.3}$	$61.1_{\pm 0.2}$	$60.8_{\pm 0.4}$	$\mathbf{62.1_{\pm 0.1}}$	$61.2_{\pm 0.3}$
ImageNet-PI (high-noise)	$53.3_{\pm 0.5}$	$53.9_{\pm 0.4}$	$51.5_{\pm 0.6}$	$\mathbf{55.8_{\pm 0.3}}$	$\mathbf{55.4_{\pm 0.4}}$