Transfer and Marginalize: Explaining Away Label Noise with Privileged Information

Mark Collier [1] **Rodolphe Jenatton** [1] **Efi Kokiopoulou** [1] **Jesse Berent** [1]

Abstract

Supervised learning datasets often have privileged information, in the form of features which are available at training time but are not available at test time e.g. the ID of the annotator that provided the label. We argue that privileged information is useful for explaining away label noise, thereby reducing the harmful impact of noisy labels. We develop a simple and efficient method for supervised neural networks: it transfers the knowledge learned with privileged information via weight sharing and approximately marginalizes over privileged information at test time. Our method, **TRAM** (TRansfer and Marginalize), has the same test time computational cost as not using privileged information, and performs strongly on CIFAR-10H and ImageNet benchmarks.

1. Introduction

Classification problems are typically formalized as learning a conditional distribution $p(y|\mathbf{x})$, $y \in \{1, ..., C\}$ and $\mathbf{x} \in \mathcal{X}$ from $(\mathbf{x}_i, y_i)$, $i = 1, ..., N$ pairs. Yet we often have access to additional features $\mathbf{a} \in \mathcal{A}$ at training time that will not be available at test time. These features are known as *privileged information* (Vapnik & Vashist, 2009), or **PI** for short. An example of PI are features describing the human annotator that provided a given label, such as the annotator ID, the length of time to provide the label, the experience of the annotator, etc. Annotators do not always agree on the correct label for a given $\mathbf{x}$, some annotators may be more reliable than others and the reliability of annotators may depend on the location of $\mathbf{x}$ in the input domain $\mathcal{X}$ (Snow et al., 2008; Sheng et al., 2008).

The expanded training dataset consists of $(\mathbf{x}_i, \mathbf{a}_i, y_i)$ triplets. Given that our test time predictive distribution cannot be conditioned on $\mathbf{a}$, what use is this PI? As a thought experiment, suppose there exists a malicious (or lazy) annotator that provides random labels. It is known that random labels harm the performance of supervised learning models (Frénay & Verleysen, 2013). If these random labels can be explained away via access to PI, such as the annotator ID, then this harm can be prevented. In particular we can use the PI to explain away noise in the labels which otherwise would be irreducible aleatoric uncertainty.

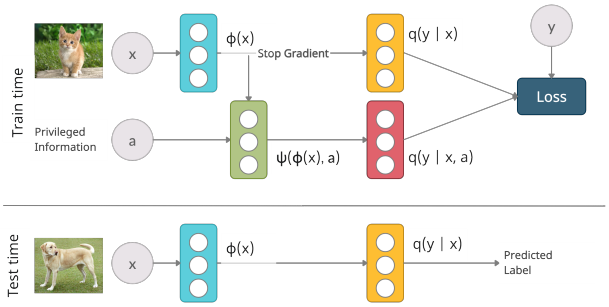

Figure 1. TRAM diagram.

More formally, suppose $\mathbf{A}$, the PI random variable, is predictive of $\mathbf{Y}$ given $\mathbf{X}$, in the sense that the conditional mutual information $I(\mathbf{Y}; \mathbf{A}|\mathbf{X})$ is non-zero. Then, the entropy of $\mathbf{Y}$ is reduced if we condition on *both* $\mathbf{X}$ and $\mathbf{A}$ rather than $\mathbf{X}$ alone, as summarised in Lemma 1.1.

Lemma 1.1 $I(\mathbf{Y}; \mathbf{A}|\mathbf{X}) > 0 \Rightarrow H(\mathbf{Y}|\mathbf{X}, \mathbf{A}) < H(\mathbf{Y}|\mathbf{X})$.

Under some assumptions, prior work has proven that PI can lead to generalization bounds with better sample complexity (Vapnik & Vashist, 2009; Lambert et al., 2018).

In this paper we focus on exploiting PI in supervised deep neural networks. The production deployment of such models often has tight latency and memory constraints. Hence a number of methods have been developed to utilize PI with the same test time memory and computation cost as networks trained without PI (Yang et al., 2017; Lambert et al., 2018; Lopez-Paz et al., 2015). Yang et al. (2017) uses PI as a form of input-dependent regularizer. Lambert et al. (2018) train with heteroscedastic Gaussian dropout, with the training-time dropout variance a function of the PI. Lopez-Paz et al. (2015) distill a network trained with PI into a network without access to $\mathbf{a}$.

[1]Google AI. Correspondence to: Mark Collier <markcollier@google.com>.

Presented at the ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning., Copyright 2021 by the author(s).

Below we develop a method, TRAM, which transfers knowledge via weight sharing from the network trained using PI to the test time network which does not have access to PI. At test time, TRAM makes a simple, efficient approximation to the integral $p(y|\mathbf{x}) = \int p(y|\mathbf{x}, \mathbf{a})p(\mathbf{a}|\mathbf{x})d\mathbf{a}$. Making predictions without PI is no more costly than that with a standard network without access to PI. Unlike prior work which requires specific techniques such as Gaussian Dropout, we need not constrain the form of the predictors to make the downstream marginalization possible. Implementation and training are simple. Empirically, we show that our method performs better than a series of baselines.

2. Method: TRAM

We consider learning under privileged information (Vapnik & Vashist, 2009), **LUPI**. Our proposed method, TRAM, consists of a single neural network with two output heads, providing predictions for both $p(y|\mathbf{x}, \mathbf{a})$ and $p(y|\mathbf{x})$; see Figure 1. There are two key ingredients to TRAM; (i) the $p(y|\mathbf{x})$ head is a simple, yet a provably valid, approximation to the marginal $\int p(y|\mathbf{x}, \mathbf{a})p(\mathbf{a}|\mathbf{x})d\mathbf{a}$ and (ii) a partition of the parameter space such that the neural network weights are shared between the two output heads, and that these shared weights are updated solely based on the gradients from the $p(y|\mathbf{x}, \mathbf{a})$ head which has access to PI. Below we develop TRAM in the classification setting but the method can naturally be extended to the regression setting, Appendix B.

2.1. Ingredient #1: Marginalize over PI at test time

A natural probabilistic approach to LUPI is (i) to learn the conditional distribution $p(y|\mathbf{x}, \mathbf{a})$ during training and (ii) then, at test time, marginalize over the $\mathcal{A}$ domain $p(y|\mathbf{x}) = \int p(y|\mathbf{x}, \mathbf{a})p(\mathbf{a}|\mathbf{x})d\mathbf{a}$ (Lambert et al., 2018).

Predicting with the marginal $p(y|\mathbf{x})$ at test time is motivated by the following key observation. Consider the set of distributions $\mathcal{Q}$ over the C class labels, $\mathcal{Q} = \{q(y|\cdot)|\forall \mathbf{x} \in \mathcal{X}, q(y|\mathbf{x}) \in \Delta_C\}$ where Δ_C is the C-dimensional simplex. Among all the distributions $q \in \mathcal{Q}$, the marginal distribution $\mathbf{x} \mapsto p(y|\mathbf{x})$ minimizes the following optimization problem:

$$\min_{q \in \mathcal{Q}} \mathbb{E}_{(\mathbf{x},\mathbf{a}) \sim p(\mathbf{x},\mathbf{a})} \left[D_{KL}(p(y|\mathbf{x}, \mathbf{a}) \,\|\, q(y|\mathbf{x})) \right]. \quad (1)$$

See proof in Appendix A. In words, $p(y|\mathbf{x})$ is optimal in the sense that it minimizes the *expected* KL divergence to $p(y|\mathbf{x}, \mathbf{a})$. Access to $p(y|\mathbf{x})$ also enables minimizing the Bayes risk (Murphy, 2012) at test time.

Directly computing $p(y|\mathbf{x})$ has two problems; (i) for all but the simplest cases it is intractable and (ii) $p(\mathbf{a}|\mathbf{x})$ is unknown and therefore must be learned, which is a challenging generative modelling problem in itself, especially as $\mathbf{a}$ typically has mixed type features. A Monte Carlo estimate of the integral by using the samples from $\mathcal{A}$ in the training set is typically only feasible with the independence assumption $p(\mathbf{a}|\mathbf{x}) = p(\mathbf{a})$, so that $p(y|\mathbf{x})$ reduces to $\int p(y|\mathbf{x}, \mathbf{a})p(\mathbf{a})d\mathbf{a} \approx \frac{1}{S}\sum_{s=1}^{S} p(y|\mathbf{x}, \mathbf{a}_s)$ with $\mathbf{a}_s \sim p(\mathbf{a})$.

Unfortunately this independence assumption is often violated in practice. In addition the memory and computational cost of MC estimation scales linearly in S, the number of MC samples. This $\mathcal{O}(S)$ scaling is undesirable for production deployment with strict latency requirements.

Due to the challenge of computing the integral directly, we thus propose a simple approximation $q(y|\mathbf{x}; \mathbf{w})$ to $p(y|\mathbf{x})$. It exploits the representation (1) of $p(y|\mathbf{x})$ as the distribution minimizing the expected KL divergence to its conditional $p(y|\mathbf{x}, \mathbf{a})$. We choose q to be cheap to evaluate at test time. For example, for a multi-class vanilla TRAM classifier $q(y|\mathbf{x}; \mathbf{w}) = \texttt{softmax}(\mathbf{W}\phi(\mathbf{x}))$.

2.2. Ingredient #2: Transfer via weight sharing

We partition the parameter space into four disjoint subsets;
(1) Let $\phi(\mathbf{x})$ be a feature extractor for $\mathbf{x} \in \mathcal{X}$.
(2) Similarly, let $\psi(\phi(\mathbf{x}), \mathbf{a})$ be a feature extractor *jointly* applied to $(\phi(\mathbf{x}), \mathbf{a})$ for $(\mathbf{x}, \mathbf{a}) \in \mathcal{X} \times \mathcal{A}$.
(3) The weights $\mathbf{w}$ parameterize the marginal distribution: $q(y|\mathbf{x}; \mathbf{w}) = q(y|\phi(\mathbf{x}); \mathbf{w})$.
(4) The weights $\mathbf{u}$ parameterize the conditional distribution: $q(y|\mathbf{x}, \mathbf{a}; \mathbf{u}) = q(y|\psi(\phi(\mathbf{x}), \mathbf{a}); \mathbf{u})$.

Two-step approach. Motivated by Eq. (1) and the connection between LUPI and multi-task learning (Jonschkowski et al., 2016), we consider the following two-step approach:

$$\min_{\mathbf{u},\phi,\psi} \mathbb{E}_{(\mathbf{x},\mathbf{a},y) \sim p(\mathbf{x},\mathbf{a},y)} \left[\mathcal{L}(y, q(y|\mathbf{x}, \mathbf{a})) \right] \quad (2)$$

$$\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},\mathbf{a},y) \sim p(\mathbf{x},\mathbf{a},y)} \left[\texttt{CE}(y, q(y|\mathbf{x})) \right] \text{ with } \phi = \phi^\star \quad (3)$$

$\mathcal{L}$ is an arbitrary loss function and $\texttt{CE}$ is the cross entropy. We assume ϕ and ψ are parameterized as neural networks, so $\min_{\phi,\psi}$ refers to optimizing the network weights.

Crucially $\phi^\star$ is the feature extractor learned in (2) with access to PI. This weight sharing enables **knowledge transfer** to the network trained without PI. Given (1), we know that (3) approximates the true marginal distribution $p(y|\mathbf{x})$ (observe that the KL divergence in (1) reduces to $\texttt{CE}$ when taking the one-hot training labels for $p(y|\mathbf{x}, \mathbf{a})$).

Merging the two steps. To further simplify the above approach, we propose to merge (2) and (3) into a *single* training procedure. To that end, and reusing the terminology commonly used in deep-learning frameworks, let us define

$$\pi(y|\mathbf{x}; \mathbf{w}) = q\big(y|\texttt{stop\_gradient}(\phi(\mathbf{x})); \mathbf{w}\big)$$

which coincides with $q(y|\mathbf{x})$ except that its gradient only depends on $\mathbf{w}$. For some $\beta > 0$, we then consider:

$$\min_{\mathbf{u},\mathbf{w},\phi,\psi} \mathbb{E}_{(\mathbf{x},\mathbf{a},y) \sim p(\mathbf{x},\mathbf{a},y)} \left[\texttt{CE}(y, \pi(y|\mathbf{x})) + \beta \mathcal{L}(y, q(y|\mathbf{x}, \mathbf{a})) \right]$$

as the joint training objective. In practice, since the parameters of the two losses are partitioned, we can set $\beta = 1$ and fold instead the search over β into the search of the learning rate, hence not introducing an extra hyperparameter.

2.3. TRAM variants

Privileged information may only explain away some of label noise uncertainty. Below we propose two TRAM variants which combine TRAM with existing noisy labels methods.

Het-TRAM. Heteroscedastic classifiers have been successfully applied in this setting (Collier et al., 2021). Further note that even in some cases where the conditional distribution $q(y|\mathbf{x}, \mathbf{a})$ is homoscedastic, the marginal $q(y|\mathbf{x})$ is heteroscedastic, see Appendix C for details.

Hence we propose **Het-TRAM**, a TRAM variant in which $q(y|\mathbf{x})$ is heteroscedastic. This increases the expressiveness of q, improving the approximation in the second step of our optimization procedure, Eq. (3). We implement the method of Collier et al. (2021) to make $q(y|\mathbf{x})$ heteroscedastic.

Distilled-TRAM. Distillation (Hinton et al., 2015) is a technique for transfering knowledge between two neural networks. Distillation has been previously applied to LUPI (Lopez-Paz et al., 2015). In Distilled-TRAM we set the loss function, $\mathcal{L}$ in Eq. (2), to the distillation loss. The teacher network is first trained with access to PI and in the second step the distilled-TRAM model is trained.

3. Related Work

Vapnik & Vashist (2009) develop a theoretic framework for the LUPI paradigm and introduce the SVM+ method for training Support Vector Machines in this regime. The slack variables for the SVM+ constraints are a function of the PI. SVM+ has been extended in the SVM literature (Lapin et al., 2014; Vapnik & Izmailov, 2015). Jonschkowski et al. (2016) provide a unifying framework that connects together multi-task learning, multi-view learning and LUPI.

Yang et al. (2017) extend the SVM+ approach to neural network models with their MIML-FCN+ method. The authors formulate a two-tower network similar to our network, but without weight sharing between the towers. Both towers make independent predictions given either $\mathbf{x}$ or $\mathbf{a}$ as inputs. The tower with access to PI predicts the loss of the other tower and this prediction is regularized to be close to the true loss. In this way the PI tower outputs a neural network analogue to the SVM+ slack variables.

Lambert et al. (2018) utilize PI by making the training-time Gaussian-dropout variance (Kingma et al., 2015) a function of the PI. At test time the PI is approximately marginalized over by removing the dropout. Similarly Hernández-Lobato et al. (2014) allow the additive Gaussian noise component of a heteroscedastic Gaussian Process Classifier (Rasmussen & Williams, 2006) to be a function of the PI. The classifier is homoscedastic at test time.

Lopez-Paz et al. (2015) propose a distillation (Hinton et al., 2015) style approach to learning with PI. The teacher network is trained with access to PI. In the distillation step the student network is given $\mathbf{x}$ as input and a convex combination of soft labels from the teacher network and true labels y as targets. Xu et al. (2020) extend and apply this distillation method to a recommender system.

TRAM implements knowledge transfer via weight sharing, performs efficient approximate marginalization at test time and can be applied to many widely used architectures. Lambert et al. (2018) also share weights and approximate the marginal $p(y|\mathbf{x})$ however they require the use of Gaussian dropout, which is not widely used. The distillation and MIML-FCN+ methods do not transfer via weight sharing and do not approximate $p(y|\mathbf{x})$. Distillation also requires a two-step training procedure. See Table 3 in the Appendix for a comparison of the key features of selected LUPI methods.

4. Experiments

Our experiments tackle the general LUPI problem under label noise, especially when the PI refers to annotator features. There are a few large-scale public datasets with annotator features. We thus use both real datasets with annotator features as well as synthesizing annotator features for a re-labelled version of ImageNet (Deng et al., 2009).

We evaluate a number of baselines in addition to our method. The "No PI" baseline is standard neural network training which directly learns $p(y|\mathbf{x})$ and never uses PI. Zero and mean imputation learn $p(y|\mathbf{x}, \mathbf{a})$ at training time and substitute $\mathbf{a} = \mathbf{0}$ and $\mathbf{a} = \frac{1}{N}\sum_i \mathbf{a}_i$ respectively at test time. For mean imputation, averaging takes place after feature preprocessing, e.g., one-hot encoding of the annotator ID. The "Full marginalization" baseline is an expensive MC estimate of $p(y|\mathbf{x}) = \int p(y|\mathbf{x}, \mathbf{a})p(\mathbf{a}|\mathbf{x})d\mathbf{a}$ at test time, see §2.1 for details. It is a gold standard (up to independence assumption error), impractical to compute in many applications. Prior work did not evaluate against these simple imputation baselines or full marginalization (Lopez-Paz et al., 2015; Yang et al., 2017; Lambert et al., 2018). We also compare distillation based approaches. "Distillation No PI" is an ablation to see the effect of distillation alone, independent of PI, in which a network trained without access to PI is distilled into another network also without access to PI.

4.1. CIFAR-10H

One dataset with annotator features is CIFAR-10H (Peterson et al., 2019), which is a re-labelled version of the CIFAR-10

Table 1. CIFAR-10 negative log-likelihood & accuracy (trained on CIFAR-10H). Averaged over 20 training runs ± 1 std. dev.

METHOD	NLL	ACCURACY
NO PI	1.058 ± 0.050	67.0 ± 1.7
ZERO IMPUTATION	1.009 ± 0.032	68.7 ± 1.4
MEAN IMPUTATION	**0.963** ± 0.058	70.1 ± 1.5
FULL MARGINALIZATION	1.119 ± 0.058	70.3 ± 2.5
TRAM	0.980 ± 0.037	70.1 ± 1.4
HET-TRAM	0.972 ± 0.038	**70.4** ± 1.5
DISTILLATION NO PI	1.118 ± 0.037	70.1 ± 1.4
LOPEZ-PAZ ET AL. (2015)	1.121 ± 0.040	70.2 ± 1.4
DISTILLED-TRAM	**0.941** ± 0.039	**71.8** ± 1.4

Table 2. ImageNet validation set negative log-likelihood and accuracy. Averaged over 10 training runs ± 1 std. dev.

METHOD	NLL	ACCURACY
NO PI	1.196 ± 0.006	73.3 ± 0.1
ZERO IMPUTATION	1.136 ± 0.006	72.7 ± 0.2
MEAN IMPUTATION	1.128 ± 0.005	72.8 ± 0.2
FULL MARGINALIZATION	1.181 ± 0.005	74.1 ± 0.2
TRAM	**1.161** ± 0.005	74.0 ± 0.2
HET-TRAM	1.166 ± 0.008	**74.7** ± 0.2
DISTILLATION NO PI	1.607 ± 0.006	74.3 ± 0.2
LOPEZ-PAZ ET AL. (2015)	1.420 ± 0.006	74.7 ± 0.1
DISTILLED-TRAM	**1.094** ± 0.006	**75.1** ± 0.2

(Krizhevsky & Hinton, 2009) test set. The new labels are provided by crowd-sourced human annotators. We make use of three annotator features; the annotator ID, the reaction time of the annotator to provide the label and how much experience the annotator had with the task, as measured by the number of labels the annotator had previously provided.

As we only have annotator features for the CIFAR-10 test set, we use this as our training set and evaluate on the official training set. As a result we have only 10,000 images for training. To achieve reasonable performance we start from a MobileNet (Howard et al., 2017) pretrained on ImageNet. Images have on average > 50 annotations each. This is unrealistic for typical applications where 1-3 labels per example is more common. Therefore, we subsample 16,400 labels (1.64 labels per example), see Appendix D for details of the subsampling procedure. The subsampled labels agree with the true CIFAR-10 test set labels 79.4% of the time.

In Table 1 we see the results. First, and as expected, using annotator features via TRAM, full marginalization or the imputation methods provides a significant performance improvement over standard neural network training without PI. Second, we see that TRAM performs on par with full marginalization (which uses 16,400 MC samples of $\mathbf{a}$ from the training set), despite having constant time compute and memory requirements w.r.t. the number of MC samples for the full marginalization baseline. Mean imputation is a strong baseline on CIFAR-10H. Het-TRAM improves over TRAM demonstrating the efficacy of making $q(y|\mathbf{x}, \mathbf{a})$ heteroscedastic. It is noteworthy that distillation using PI, (Lopez-Paz et al., 2015) does not improve over standard distillation without PI. However Distilled-TRAM with makes use of PI for distillation but then performs approximate marginalization and transfer learning via weight sharing shows significant improvement over the distillation baselines on both accuracy and log-likelihood metrics.

4.2. ImageNet ILSVRC12

In order to create a large-scale dataset with annotator features, we re-label the ImageNet ILSVRC12 training set by the following procedure. We download 16 different models pre-trained on ImageNet, see Appendix D for further details. We also add a 17th malicious annotator model which picks a label uniformly at random from the 1,000 ImageNet ILSVRC12 classes. For each image in the training set we select the malicious annotator with 10% probability and otherwise sample one of the 16 models with equal probability. We then sample a label from the predictive distribution of that model for that image. This is the label used for training. On average the sampled label agrees with the true ImageNet label 68.3% of the time. The annotator features are the model ID (a proxy for a human annotator ID) and the probability of the label assigned by the model (a proxy for the confidence of a human annotator).

See Table 2 for the results. The full marginalization baseline uses 1,000 MC samples of $\mathbf{a}$ from the training set. The imputation baselines perform worse than not using PI on ImageNet, perhaps due to the imputed values having low density under $p(\mathbf{a}|\mathbf{x})$. Again TRAM performs on par with full marginalization and Het-TRAM has higher accuracy than both. The ImageNet labels are known to exhibit heteroscedasticity (Collier et al., 2021), therefore we make both $q(y|\mathbf{x})$ and $q(y|\mathbf{x}, \mathbf{a})$ heads heteroscedastic for Het-TRAM. The distillation method of Lopez-Paz et al. (2015) does improve over distillation without access to PI on this benchmark; however again Distilled-TRAM has significantly better NLL and accuracy than the distillation baselines.

5. Conclusion

We introduced TRAM, a new method for LUPI in supervised neural networks. TRAM (i) learns an efficient, simple distribution to approximately marginalize over PI at test time and (ii) partitions the parameter space enabling transfer via weight sharing of the knowledge learned with access to PI. TRAM can be successfully combined with established methods for dealing with noisy labels; distillation (Distilled-TRAM) and heteroscedastic output layers (Het-TRAM). We have empirically validated TRAM on CIFAR-10H and a noisy version of ImageNet.

References

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

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Frénay, B. and Verleysen, M. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2013.

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

Hernández-Lobato, D., Sharmanska, V., Kersting, K., Lampert, C. H., and Quadrianto, N. Mind the nuisance: Gaussian process classification using privileged noise. In *Neural Information Processing Systems*, 2014.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Jonschkowski, R., Hofer, S., and Brock, O. Patterns for learning with side information. *arXiv preprint arXiv:1511.06429*, 2016.

Kendall, A. and Gal, Y. What uncertainties do we need in bayesian deep learning for computer vision? In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 5580–5590, 2017.

Kingma, D. P., Salimans, T., and Welling, M. Variational dropout and the local reparameterization trick. *arXiv preprint arXiv:1506.02557*, 2015.

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

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6405–6416, 2017.

Lambert, J., Sener, O., and Savarese, S. Deep learning under privileged information using heteroscedastic dropout. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8886–8895, 2018.

Lapin, M., Hein, M., and Schiele, B. Learning using privileged information: Svm+ and weighted svm. *Neural Networks*, 53:95–108, 2014.

Lopez-Paz, D., Bottou, L., Schölkopf, B., and Vapnik, V. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*, 2015.

Murphy, K. P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Peterson, J. C., Battleday, R. M., Griffiths, T. L., and Russakovsky, O. Human uncertainty makes classification more robust. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9617–9626, 2019.

Rasmussen, C. E. and Williams, C. K. *Gaussian processes for machine learning*. The MIT Press, 2006.

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*, pp. 614–622, 2008.

Snow, R., O'connor, B., Jurafsky, D., and Ng, A. Y. Cheap and fast–but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 conference on empirical methods in natural language processing*, pp. 254–263, 2008.

Vapnik, V. and Izmailov, R. Learning using privileged information: similarity control and knowledge transfer. *J. Mach. Learn. Res.*, 16(1):2023–2049, 2015.

Vapnik, V. and Vashist, A. A new learning paradigm: Learning using privileged information. *Neural networks*, 22 (5-6):544–557, 2009.

Xu, C., Li, Q., Ge, J., Gao, J., Yang, X., Pei, C., Sun, F., Wu, J., Sun, H., and Ou, W. Privileged features distillation at taobao recommendations. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2590–2598, 2020.

Yang, H., Tianyi Zhou, J., Cai, J., and Soon Ong, Y. Mimlfcn+: Multi-instance multi-label learning via fully convolutional networks with privileged information. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1577–1585, 2017.

Table 3. Comparison to related work.

METHOD	$p(\mathbf{a}\|\mathbf{x})$ REQUIRED	TRAINING	TEST COST	WEIGHT SHARING	APPROXIMATE $p(y\|\mathbf{x})$
IMPUTATION	×	1 MODEL, 1 STEP	= NO PI	✓	×
DISTILLATION (LOPEZ-PAZ ET AL., 2015)	×	2 MODELS, 2 STEPS	= NO PI	×	×
HETEROSCEDASTIC DROPOUT (LAMBERT ET AL., 2018)	×	1 MODEL, 1 STEP	= NO PI	✓	✓
MIML-FCN+ (YANG ET AL., 2017)	×	1 MODEL, 1 STEP	= NO PI	×	×
FULL MARGINALIZATION	✓	1 MODEL, 1 STEP	$\mathcal{O}(S *$ NO PI)	✓	✓
TRAM (OURS)	×	1 MODEL, 1 STEP	= NO PI	✓	✓
HET-TRAM (OURS)	×	1 MODEL, 1 STEP	= NO PI	✓	✓
DISTILLED-TRAM (OURS)	×	2 MODELS, 2 STEPS	= NO PI	✓	✓

A. Proof of Equation (1)

As a reminder, we consider C class labels and denote by Δ_C the C-dimensional simplex. We define the set of distributions $\mathcal{Q}$ over the C class labels by

$$\mathcal{Q} = \{q(y|\cdot) \,|\, \forall \mathbf{x} \in \mathcal{X}, q(y|\mathbf{x}) \in \Delta_C\}.$$

Consider the optimization problem

$$\min_{q \in \mathcal{Q}} \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \left[D_{KL}(p(y|\mathbf{x}) \,\|\, q(y|\mathbf{x})) \right] \tag{4}$$

whose solution is straightforwardly given by the marginal distribution $\mathbf{x} \mapsto q^\star(y|\mathbf{x}) = p(y|\mathbf{x})$. We recall that the KL $D_{KL}(p(y|\mathbf{x}) \,\|\, q(y|\mathbf{x}))$ is defined by

$$D_{KL}(p(y|\mathbf{x}) \,\|\, q(y|\mathbf{x})) = \sum_{j=1}^{C} p_j(y|\mathbf{x}) \log \left(\frac{p_j(y|\mathbf{x})}{q_j(y|\mathbf{x})} \right). \tag{5}$$

For any $\mathbf{x}$ and $j \leq C$, we can rewrite the terms of the sum

$$p_j(y|\mathbf{x}) \log \left(\frac{p_j(y|\mathbf{x})}{q_j(y|\mathbf{x})} \right)$$

as

$$\mathbb{E}_{\mathbf{a}|\mathbf{x} \sim p(\mathbf{a}|\mathbf{x})} \left[p_j(y|\mathbf{x}, \mathbf{a}) \log \left(\frac{p_j(y|\mathbf{x})}{q_j(y|\mathbf{x})} \right) \right]$$

where we have used (i) the fact that $\log(\frac{p_j(y|\mathbf{x})}{q_j(y|\mathbf{x})})$ does not depend on $\mathbf{a}$ and (ii) the definition of the marginal distribution

$$\begin{aligned} p_j(y|\mathbf{x}) = & \int p_j(y|\mathbf{x}, \mathbf{a}) p(\mathbf{a}|\mathbf{x}) d\mathbf{a} \\ = & \mathbb{E}_{\mathbf{a}|\mathbf{x} \sim p(\mathbf{a}|\mathbf{x})} \left[p_j(y|\mathbf{x}, \mathbf{a}) \right]. \end{aligned}$$

Multiplying and dividing in the argument of the $\log$ by $p_j(y|\mathbf{x}, \mathbf{a})$, we obtain

$$\mathbb{E}_{\mathbf{a}|\mathbf{x} \sim p(\mathbf{a}|\mathbf{x})} \left[p_j(y|\mathbf{x}, \mathbf{a}) \log \left(\frac{p_j(y|\mathbf{x}, \mathbf{a})}{q_j(y|\mathbf{x})} \frac{p_j(y|\mathbf{x})}{p_j(y|\mathbf{x}, \mathbf{a})} \right) \right].$$

Summing over $j \in \{1, \ldots, C\}$ to reconstruct the KL term (5), this leads to, for any $\mathbf{x}$,

$$\begin{aligned} D_{KL}(p(y|\mathbf{x}) \,\|\, q(y|\mathbf{x})) = & \mathbb{E}_{\mathbf{a}|\mathbf{x}} \left[D_{KL}(p(y|\mathbf{x}, \mathbf{a}) \,\|\, q(y|\mathbf{x})) \right] \\ & - \mathbb{E}_{\mathbf{a}|\mathbf{x}} \left[D_{KL}(p(y|\mathbf{x}, \mathbf{a}) \,\|\, p(y|\mathbf{x})) \right]. \end{aligned}$$

Since the second term above *does not depend on* q, minimizing (4) is equivalent to minimizing

$$\begin{aligned} & \min_{q \in \mathcal{Q}} \quad \mathbb{E}_{\mathbf{x}} \left[\mathbb{E}_{\mathbf{a}|\mathbf{x}} \left[D_{KL}(p(y|\mathbf{x}, \mathbf{a}) \,\|\, q(y|\mathbf{x})) \right] \right] \\ = & \min_{q \in \mathcal{Q}} \quad \mathbb{E}_{(\mathbf{x}, \mathbf{a}) \sim p(\mathbf{x}, \mathbf{a})} \left[D_{KL}(p(y|\mathbf{x}, \mathbf{a}) \,\|\, q(y|\mathbf{x})) \right] \end{aligned}$$

which is equal to (1) and which is, analogously to (4), minimized by the marginal distribution $\mathbf{x} \mapsto q^\star(y|\mathbf{x}) = p(y|\mathbf{x})$.

B. Regression

We developed TRAM and Het-TRAM focussing on the classification setting but our approach is trivial to generalize to regression problems.

In the regression case, we can choose the predictive distribution to be Gaussian, $q(y|\mathbf{x}) = \mathcal{N}(\mu(\mathbf{x}), \sigma^2(\mathbf{x}))$. For vanilla TRAM we can choose $\sigma^2(\mathbf{x}) = 1$, while for Het-TRAM we can choose $\sigma^2(\mathbf{x}) = \texttt{softplus}(\mathbf{w}_\sigma^\top \phi(\mathbf{x}))$. μ and σ^2 are parameterized as neural networks with our shared feature extractor $\phi(\mathbf{x})$, similar to Kendall & Gal (2017). The joint objective function Eq.(2.2) is unchanged.

C. Heteroscedastic Motivation

We consider a simplified special case of our framework in which the conditional model $p(y|\mathbf{x}, \mathbf{a})$ is *homoscedastic* but the optimal variational distribution in the sense of Eq. 1 is *heteroscedastic*. This motivates **Het-TRAM**, in which the variational approximations $q(y|\mathbf{x})$ and $q(y|\mathbf{x}, \mathbf{a})$ are heteroscedastic.

Suppose we have a regression dataset constructed from labels assigned by M annotators. Each annotator has their own homoscedastic Gaussian model $p(y|\mathbf{x}, a = m) = \mathcal{N}(\mu_{\theta_m}(\mathbf{x}), 1)$. Here the PI is a single discrete Categorical feature representing the annotator ID which takes one of M values with equal probability, $a \sim \text{Cat}(\frac{1}{M})$.

The marginal $p(y|\mathbf{x})$ is a Gaussian Mixture Model. We choose our variational family to be the Gaussian distribution, $q(y|\mathbf{x}) = \mathcal{N}(\mu(\mathbf{x}), \sigma^2(\mathbf{x}))$. The values of μ and σ^2 that minimize Eq. 1 are: $\mu_*(\mathbf{x}) = \frac{1}{M} \sum_m \mu_{\theta_m}(\mathbf{x})$ and $\sigma_*^2(\mathbf{x}) = (M - \mu_*(\mathbf{x})) + \frac{1}{M} \sum_m \mu_{\theta_m}^2(\mathbf{x})$ (Lakshmi-

narayanan et al., 2017). Crucially note that despite the conditional distribution being homoscedastic, the best variational distribution is heteroscedastic as the variance depends on the location in $\mathcal{X}$ space.

D. Experimental Details

D.1. Data generation process

CIFAR-10H. We use the CIFAR-10 image as the non-privileged information $\mathbf{x}$. The annotator ID, the number of prior annotations the annotator has provided and the reaction time in milliseconds of the annotator, are used as privileged information $\mathbf{a}$. For feature pre-processing the annotator ID is one-hot encoded. The number of prior annotations and the reaction time are independently divided into 10 equally sized quantiles and the quantile ID is one-hot encoded. The image is pre-processed according the the standard MobileNet pre-processing (Howard et al., 2017).

As CIFAR-10H has on average more than 50 annotations per image and the labels are not particularly noisy. We subsample the CIFAR-10H labels by the following procedure. We keep all labels by the 41 annotators that agree with the true CIFAR-10 label less than 85% of the time. We then select a further 41 annotators from the remaining annotators. The average agreement of the bad annotators with the CIFAR-10 label is 63.3%, in the full subset of labels: 79.2% and in the full CIFAR-10H dataset: 94.9%. The subsampling procedure leaves 16,400 labels from 82 annotators while the full CIFAR-10H dataset has 514,200 labels from 2,571 annotators.

ImageNet. The ImageNet image is used as the non-privileged information $\mathbf{x}$. It is pre-processed with standard ResNet-50 pre-processing (He et al., 2016). The annotator features are the model ID used to re-label $\mathbf{x}$, which is one-hot encoded and the probability of that label being sampled. See main paper for details on the sampling procedure and see Table 4 for the list of models used and their accuracy on the ImageNet training set. The pre-trained models are downloaded from tf.keras.applications[1].

Table 4. Pre-trained models used to re-label ImageNet ILSVRC12 training set and their accuracy on that training set.

Model	Training set accuracy
ResNet50V2	0.70086
ResNet101V2	0.72346
ResNet152V2	0.72738
DenseNet121	0.74782
DenseNet169	0.76184
DenseNet201	0.77344
InceptionResNetV2	0.8049
InceptionV3	0.77994
MobileNet	0.70594
MobileNetV2	0.71458
MobileNetV3Large	0.75622
MobileNetV3Small	0.68158
NASNetMobile	0.74302
VGG16	0.71178
VGG19	0.71156
Xception	0.79076

D.2. Hyperparameters

CIFAR-10H. For all methods $\phi(\mathbf{x})$ (or equivalent) is a MobileNet (Howard et al., 2017) pre-trained on ImageNet ILSVRC12, followed by a global average pooling layer and a Dense + ReLU layer with 64 units. $\psi(\mathbf{x}, \mathbf{a})$ is a two-layer MLP with 64 units per layer and ReLU activation. The first layer takes only $\mathbf{a}$ as an input, while the second layer takes the output of the first layer concatenated with $\phi(\mathbf{x})$ as input.

All models are trained for 20 epochs with the Adam optimizer with base learning rate$= 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-07$. All models are trained with L2 weight regularization with weighting $1e-3$.

Heteroscedastic models are trained using the method of Collier et al. (2021) with 4 factors for the low-rank covariance matrix approximation and a softmax temperature parameter of $\tau = 3.0$. Distilled models are also trained with a softmax temperature of $\tau = 3.0$ to smooth the teacher labels and with the distillation hyperparameter $\lambda = 0.5$ which weights the losses from the soft teacher labels and the true labels. A grid search over $\tau \in \{1.0, 2.0, 3.0, 4.0\}$ and $\lambda \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$ was conducted.

ImageNet. For all methods $\phi(\mathbf{x})$ (or equivalent) is a randomly initialized ResNet-50 (He et al., 2016) with the output layer removed. $\psi(\mathbf{x}, \mathbf{a})$ is a two-layer MLP with 128 units per layer and ReLU activation, the output of this MLP is concatenated with $\phi(\mathbf{x})$ and then passed to the output layer. The first layer of the $\psi(\mathbf{x}, \mathbf{a})$ MLP takes only $\mathbf{a}$ as an input, while the second layer takes the output of the first layer concatenated with $\phi(\mathbf{x})$ as input.

All but Het-TRAM models are trained for 90 epochs with the SGD optimizer with base learning rate$= 0.1$, decayed by a factor of 10 after 30, 60 and 80 epochs. Following Collier et al. (2021), Het-TRAM is trained for 270 epochs with the same initial learning rate and learning rate decay at 90, 180 and 240 epochs. All models are trained with L2 weight regularization with weighting $1e-4$.

Heteroscedastic models use 15 factors for the low-rank

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

covariance matrix approximation and a softmax temperature parameter of $\tau = 1.5$. Distilled models are trained with a softmax temperature of $\tau = 3.0$ and with the distillation hyperparameter $\lambda = 0.5$. A grid search over $\tau \in \{1.0, 2.0, 3.0, 4.0\}$ and $\lambda \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$ was conducted.