Batch Inverse-Variance Weighting: Deep Heteroscedastic Regression

Vincent Mai [1] **Waleed Khamies** [1] **Liam Paull** [1]

Abstract

In the supervised learning task of heteroscedastic regression, each label is subject to noise from a different distribution. The label generator may estimate the variance of the noise distribution for each label, which is useful information to mitigate its impact. We adapt an inverse-variance weighted mean square error, based on the Gauss-Markov theorem, for gradient descent on neural networks. We introduce Batch Inverse-Variance, a loss function which is robust to near-ground truth samples, and allows to control the effective learning rate. Our experimental results show that BIV improves significantly the performance of the networks on two noisy datasets, compared to L2 loss, inverse-variance weighting, and a filtering-based baseline.

1. Introduction

Noisy labels in supervised learning violate the i.i.d. assumption as $p_{\text{train}}(y|\mathbf{x}) \neq p_{\text{test}}(y|\mathbf{x})$. They may cause the performance of the deployed model to decrease, since the model parameters were not optimized on the correct data (Arpit et al., 2017; Kawaguchi et al., 2020).

In this paper, we focus on the task of regression with a deep neural network when labels are corrupted by heteroscedastic noise, which we assume Gaussian: $p_{\text{train}}(y|\mathbf{x}) \sim \mathcal{N}(\mu_y, \sigma_y)$ where μ_y is the ground truth label and σ_y is different for each sample. We assume that we have access to an estimate of the variance σ_y for each corrupted label. This information is available if the labels are being generated by some stochastic process that is capable of also reporting uncertainty. We examine how the knowledge of the estimate of the label noise variance can be used to mitigate the effect of the noise on the learning process of a deep neural network. We present Batch Inverse-Variance (BIV), a method based on inverse-variance weighting for linear regression (IV). IV loss has an empirical advantage over the standard L2 loss; however, it is unstable in presence of near ground-truth labels and not appropriate for all optimizers. BIV includes two mechanisms to stabilize the learning process in this case. Our claimed contributions are threefold:

1. An analysis of the labelling process behind the heteroscedastic regression problem.
2. We present BIV, a loss function which uses the noise variance to minimize the performance loss due to the noise in a robust and reliable way while keeping control on the learning rate. [1]
3. An experimental study comparing the performances of BIV over L2 loss, IV, and a threshold-based baseline.

The use of the variance estimate in heteroscedastic regression for neural networks has not been studied in detail in the literature, maybe because neural networks are usually robust to strong label noise (Rolnick et al., 2018) or because the datasets traditionally do not include such information. We believe however that (1) the recent usage of deep neural networks in fields in which uncertainty plays a major role, such as robotics (Thrun et al., 2006), as well as (2) the combined advances in uncertainty estimation of neural network outputs (Kendall & Gal, 2017; Lakshminarayanan et al., 2017) and in complex deep learning systems such as deep reinforcement learning where the labels are provided by another neural network (Mnih et al., 2015; Haarnoja et al., 2018), make this problem highly relevant.

2. Heteroscedastic noisy labels in regression

To label an unlabelled dataset with inputs $\{\mathbf{x}_i\}$, one must apply to each $\mathbf{x}_i$ an instance of a label generator LG_j which should provide its true label y_i. LG_j has access to some features $\mathbf{z}_i \in \mathcal{Z}$ correlated to $\mathbf{x}_i$. We define $LG_j : \mathcal{Z} \to \mathbb{R}$. When the labelling process is not exact and causes some noise on the label, the noisy label of $\mathbf{x}_i$ provided by LG_j is defined as $\tilde{y}_{i,j}$. We model the noise as Gaussian:

$$\tilde{y}_{i,j} = y_i + \delta_{y_{i,j}} \text{ with } \delta_{y_{i,j}} \sim N(0, \sigma^2_{i,j}) \tag{1}$$

$\sigma^2_{i,j}$ can be a function of $\mathbf{z}_i$ and LG_j, without any assumption on its dependence on one or the other. We finally as-

[1]Robotics and Embodied AI Lab, Mila - Quebec Institute of Artificial Intelligence, University of Montreal, Canada. Correspondence to: Vincent Mai <vincent.mai@umontreal.ca>.

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

[1]Our code will be publicly released after the reviewing process.

sume that the label generator is able to provide an estimate of $\sigma^2_{i,j}$, therefore $LG_j : \mathcal{Z} \rightarrow \mathbb{R} \times \mathbb{R}_{\geq 0}$. The training dataset is formed of triplets $(\mathbf{x}_i, \sigma^2_{i,j}, \tilde{y}_{i,j})$, renamed $(\mathbf{x}_k, \sigma^2_k, \tilde{y}_k)$ for triplet k for simplicity. As analyzed in appendix A, this setup describes many labelling processes, such as crowd sourcing, population studies, sensor readings, state estimation, simulation, and even neural networks.

In classification, (Xie et al., 2016; 2020) provide a "confidence" score from 0 to 255 for each pixel in the KITTI-360 dataset. However, we could not find any dataset providing label uncertainty for regression. As it is precious information, we argue that it should be provided when possible.

3. Related work

Noise on labels amounts to a loss of information and may lead to overfitting and lower model performance (Liu & Castagna, 1999; Van Horn et al., 2015; Zhang et al., 2017). Four strategies exist in the literature to tackle this problem:

Detecting noisy labels allows to ignore them in the learning process. This is often based on the observation that neural networks first fit on non-noisy data (Arpit et al., 2017), and then converge to higher losses on noisy samples (Reed et al., 2015; Shen & Sanghavi, 2019). Other methods use several neural networks (Han et al., 2018; Yu et al., 2019) or dropout (Reed et al., 2015). Noisy labels can also be **corrected**, based on a noise model which can be learned jointly with the parameters (Goldberger & Ben-Reuven, 2017; Kremer et al., 2018; Ma et al., 2018; Tanno et al., 2019; Yi & Wu, 2019). One can use **noise-robust loss functions** based on prediction error (Liu & Castagna, 1999), known mislabelling probabilities (Natarajan et al., 2013), reverse cross-entropy (Wang et al., 2019) or curriculum loss (Lyu & Tsang, 2020). The samples can also be **re-weighted**, by estimating effective label probabilities and noise rates (Liu & Tao, 2016), meta-learning the weighting function (Shu et al., 2019), or adjusting the weights to control overfitting in the mini-batches (Jenni & Favaro, 2018).

While most strategies aim at classification tasks (Song et al., 2020). Nix & Weigend (1994) tackle heteroscedastic regression in neural networks by jointly training a variance estimator based on the maximum likelihood of a Gaussian. Kendall & Gal (2017) use this idea to estimate the aleatoric (input-dependant) uncertainty of the prediction, while using dropout for the epistemic uncertainty (due to the learning process) as in Gal & Ghahramani (2016). Sai et al. (2009) use a re-weighting approach using Parzen windowing.

The main distinction between our work and the related literature is that we assume that we have access to a noise variance estimate, and we focus on how to use this additional information to train a deep neural network model.

4. Batch Inverse-Variance Weighting

4.1. Inverse variance for linear models

With a linear model of parameters β, heteroscedastic regression is optimal when it optimizes the weighted mean square error ($WMSE$) with inverse-variance weights, as per the Gauss-Markov theorem (Shalizi, 2019).

$$WMSE = \sum_{k=0}^{n} \frac{(y_k - \mathbf{x_k} \cdot \beta)^2}{\sigma_k^2} \tag{2}$$

This is solved by $\beta^* = (\mathbf{x}^T \mathbf{w} \mathbf{x}^{-1}) \mathbf{x}^T \mathbf{w} \mathbf{y}$ with $\mathrm{w}_k = 1/\sigma_k^2$.

We can apply this naively to gradient descent for neural networks with the Inverse Variance (IV) loss function:

$$\mathcal{L}_{\mathrm{IV}}(\mathbf{x}_k, \tilde{y}_k, \theta) = \frac{\left(f(\mathbf{x}_k, \theta) - \tilde{y}_k\right)^2}{\sigma_k^2} \tag{3}$$

Using $\mathcal{L}_{\mathrm{IV}}$ with gradient descent brings two problems:

1. Near ground-truth samples, with very small σ_k^2, have a disproportionate weight with respect to the others. They cause overfitting while other samples are ignored.
2. In gradient-based optimizers, the learning rate impacts the optimization process and should be controllable by the practitioner. In IV loss, the average of the sample weights $1/\sigma_k^2$ acts as a constant multiplied to the learning rate, making such control harder. While this would be less important in adaptive learning rate optimizers such as Adam (Kingma & Ba, 2017) as any constant multiplied to the loss is cancelled, it would still be problematic in continual or reinforcement learning tasks where the noise variance distribution can evolve.

4.2. Batch Inverse-Variance weighting for heteroscedastic noisy labels

To address these issues, we introduce the Batch Inverse-Variance (BIV) loss function:

$$\mathcal{L}_{\mathrm{BIV}}(D_i, \theta) = \left(\sum_{k=0}^{K} \frac{1}{\sigma_k^2 + \epsilon} \right)^{-1} \sum_{k=0}^{K} \frac{\mathcal{L}\left(f(\mathbf{x}_k, \theta), \tilde{y}_k\right)}{\sigma_k^2 + \epsilon} \tag{4}$$

On the contrary to IV loss (3), BIV is computed on the mini-batch instead of being defined on a single sample. This allows to normalize the loss based on the weights of the other samples in the batch. The gradient's scale is thus independent from the noise variance distribution, allowing better learning rate control and to use BIV in methods relying on the dynamics of the learning process.

Another difference with the IV loss is the smooth lower bound on the variance, ϵ, which allows to incorporate samples with (near) ground-truth labels without ignoring the

other samples. ϵ allows to control the effective batch size EBS, which, according to (Kish, 1965), is equal to

$$EBS = \frac{\left(\sum_i^N w_i\right)^2}{\sum_i^N w_i^2} = \frac{\left(\sum_i^N \frac{1}{(\sigma_i^2+\epsilon)}\right)^2}{\sum_i^N (\frac{1}{(\sigma_i^2+\epsilon)})^2} \quad (5)$$

A higher ϵ increases EBS by reducing the relative difference between the weights, thus reducing the effect of BIV. We found that ϵ can be set between 0.01 and 0.1 for normalized losses. More details can be found in appendix C.2.

4.3. The Cutoff baseline

In order to compare the BIV approach to a baseline, we introduce the Cutoff loss. As we do not assume a correlation between $\mathbf{x}_k$ and σ_k^2, most algorithms as presented in Section 3 are not applicable. However, we know the noise variance and do not need detection: we thus use a rejection strategy where a threshold C is put on the variance to weight samples with an inverse Heaviside step function:

$$w_k = \mathbf{1}_{\sigma_k^2 < C} \quad (6)$$

The loss is normalized by the sum of the weights.

5. Experimental Setup

We compared the performance of BIV loss (4) to the L2, cutoff (6) and IV (2) losses. As the objective of BIV is to improve the performance at predicting the true label y_i, we measured the performance of the network on ground-truth test data. We refer to ground-truth (GT) labels when using L2 loss on noise-less data as the best performance possible.

We did not find any existing dataset for regression where label uncertainty is provided. We thus used two UCI datasets (Dua & Graff, 2017), to which we artificially added noise: UTKFace Aligned&Cropped (UTKF) for image-based age prediction (Song & Zhang, 2017), and Bike Sharing (Fanaee-T & Gama, 2013) for predicting the number of bicycles rented given a date, hour, and weather conditions. Details about the datasets and models can be found in appendix B.

5.1. Noise generation

To produce the datasets $\{\mathbf{x}_k, \sigma_k^2, \tilde{y}_k\}$ with noise as described in section 2, we use a two-step process which does not assume any correlation between the noise and the input. First, the noise variance σ_k^2 is sampled from a distribution $P(\sigma^2)$ which only has support for $\sigma^2 \geq 0$. Then, $\tilde{y}_k$ is sampled from a normal distribution $\mathcal{N}(y_k, \sigma_k^2)$.

$P(\sigma^2)$ has a strong effect on the impact of BIV or Cutoff. For example, if all variances are the same, BIV becomes L2. We evaluate BIV on a Γ distribution for $P(\sigma^2)$, as it has a continuous support in both low-variance and high-variance. We show results for other distributions in the appendix C.3.

The Γ distribution has a shape parameter α and an average noise variance μ_P which we fix by adjusting $\beta = \alpha/\mu_P$. μ_P was chosen empirically so that the lowest test loss achieved by L2 is doubled compared to the ground-truth label case: $\mu_P = 2000$ for UTKF and 20000 for BikeSharing.

For a fixed μ_P, lower α induces a stronger support towards low σ^2, but the tail of the distribution spreads longer on the high σ^2 side. When $\alpha \leq 1$, the highest support is at $\sigma^2 = 0$.

6. Experimental Results and Analysis

A results that simplifies the analysis is that, under the noise model presented in section 5.1, the L2 loss trained neural networks had the same performance independently of $P(\sigma^2)$ as long as μ_P is equal, as shown in appendix C.1. All curves were smoothed with a 35-step window moving average, and represent the average and standard deviation over 5 runs.

6.1. Comparison to baselines

Figure 1 shows a strong advantage of BIV compared to L2 and cutoff for $\alpha = 1$. When threshold C is too low ($\mu_P/4$ and $\mu_P/2$), there is not enough data to train the model. When C is too high ($2\mu_P$ and $5\mu_P$), the data is too noisy and the curves go close to L2 loss. At the best case ($C = \mu_P$), cutoff is not better than BIV. In contrast to cutoff, BIV extracts some information from noisier samples while avoiding to overfit on them.

Table 1 shows the lowest value of the test loss for different α values. BIV consistently leads to the best performances, regardless of $P(\sigma^2)$. The plots showing these runs can be found in appendix C.5.1.

The benefit of BIV over L2 is clearly higher when α is lower. This is due to the higher support for low-σ^2. The more BIV can count on low-σ^2 elements and differentiate them from high noise ones, the better it can perform. Other results shown in appendix C, where BIV is performing consistently better than L2 and at least better than cutoff.

6.2. Ablation study: comparison to IV

Table 1 shows that BIV performs better than IV, but also that IV is struggling with small α. When there is a high probability that σ^2 is very close to zero, the training process diverges, as seen for $\alpha = 0.25$ for both datasets.

Figure 2 shows the impact of normalization and ϵ in BIV loss (4) compared to IV loss (3) using an ablation study. ϵ has a significant effect: the test loss is lower, and the training faster, as ϵ prevents low-variance samples to create high variability in the training loss (see appendix C.5.3).

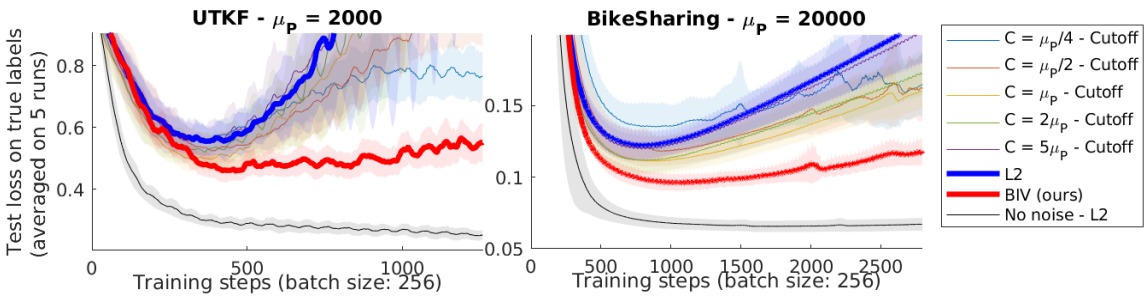

Figure 1. Comparison between the performances of BIV, L2, and different cutoff values with $P(\sigma^2)$ follows a Γ distribution with $\alpha = 1$.

Table 1. Lowest test loss (average and standard deviation over 5 runs) with different α for BIV, IV, L2 and several cutoff losses. ϵ is set as 0.05, which is equivalent to an EBS of $\sim 1/6$ of the original batch size for $\alpha = 1$.

	$\alpha = 1$		$\alpha = 0.5$		$\alpha = 0.25$	
	UTKF	Bike	UTKF	Bike	UTKF	Bike
$C = \mu_P/20$	0.79±.08	0.327±.056	0.48±.04	0.125±.010	0.38 ±.05	0.092±.008
$C = \mu_P/4$	0.55±.04	0.135±.016	0.45±.04	0.097±.006	0.39±.03	0.085±.006
$C = \mu_P$	0.50±.04	0.111±.009	0.48±.04	0.097±.008	0.43±.03	0.088±.006
$C = 5\mu_P$	0.55±.06	0.120±.009	0.54±.05	0.111±.012	0.51±.04	0.107±.009
L2	0.56±.05	0.122±.010	0.56±.05	0.116±.011	0.55±.05	0.119 ±.012
BIV (ours)	**0.46**±.03	**0.096**±.006	**0.40**±.03	**0.088**±.006	**0.33**±.02	**0.079**±.006
IV	0.99±.03	0.120±.021	1.65±.03	0.554±.041	N.A.	N.A.
GT labels	0.25±.02	0.066±.004	0.25±.02	0.066±.004	0.25±.02	0.066±.004

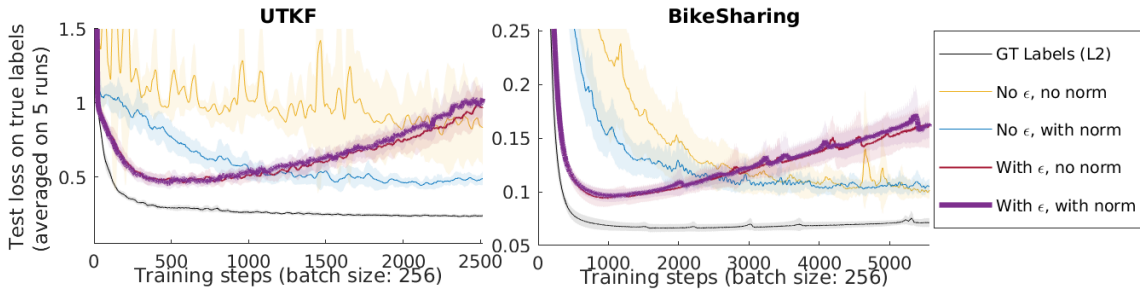

Figure 2. Ablation study for ϵ and normalization. $P(\sigma^2)$ is a Γ distribution with $\alpha = 1$.

Normalization has a lower impact, especially when $\epsilon \neq 0$, as the Adam optimizer is insensitive to a constant multiplied to the loss function. As the normalization constant is similar in every batch, it is ineffective. However, when $\epsilon = 0$, some very small-variance samples increase the sum of the weights so much that samples in other mini-batches are ignored. Normalization limits this effect to the mini-batch, allowing the neighbor samples to be taken into account at another epoch when the batches are reshuffled. With another optimizer such as stochastic gradient descent, or in tasks where the normalization constant may change over time, the effect of normalization could be a lot more significant.

7. Conclusion

Batch Inverse-Variance is a mini-batch based approach which accounts for the variance of the label noise in heteroscedastic regression tasks for neural networks. BIV is extracts more information from the noisy dataset than L2 loss or threshold-based filtering approaches, consistently outperforming them on both structured and unstructured datasets. Contrary to naive inverse-variance loss, BIV is robust to low-variance samples and enables reliable control of the learning rate. Further investigation could focus on the impact of BIV on the model's performance when the noise variance is correlated to the input space, to ensure that some regions of the input space are not doubly penalized by a higher level noise and a slower learning.

References

Alsdurf, H., Bengio, Y., Deleu, T., Gupta, P., Ippolito, D., Janda, R., Jarvie, M., Kolody, T., Krastev, S., Maharaj, T., and et al. Covi white paper. *arXiv:2005.08502 [cs]*, May 2020.

Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., and et al. A closer look at memorization in deep networks. *arXiv:1706.05394 [cs, stat]*, Jul 2017.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Fanaee-T, H. and Gama, J. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, pp. 1–15, 2013.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48, pp. 1050–1059, 2016.

Goldberger, J. and Ben-Reuven, E. Training deep neural-networks using a noise adaptation layer. In *5th International Conference on Learning Representations*, 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv:1801.01290 [cs, stat]*, Aug 2018. URL http://arxiv.org/abs/1801.01290.

Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., and Sugiyama, M. Co-teaching: Robust Training of Deep Neural Networks with Extremely Noisy Labels. *arXiv e-prints*, art. arXiv:1804.06872, April 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *arXiv:1512.03385 [cs]*, Dec 2015.

Jenni, S. and Favaro, P. Deep bilevel learning. *arXiv:1809.01465 [cs, stat]*, Sep 2018.

Kawaguchi, K., Kaelbling, L. P., and Bengio, Y. Generalization in deep learning. *arXiv:1710.05468 [cs, stat]*, Jul 2020.

Kendall, A. and Gal, Y. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems 30*, pp. 5574–5584. Curran Associates, Inc., 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv:1412.6980 [cs]*, Jan 2017.

Kish, L. *Survey Sampling*. Wiley, 1965. ISBN 978-0-471-48900-9.

Kremer, J., Sha, F., and Igel, C. Robust active label correction. volume 84 of *Proceedings of Machine Learning Research*, pp. 308–316, Playa Blanca, Lanzarote, Canary Islands, 09–11 Apr 2018. PMLR. URL http://proceedings.mlr.press/v84/kremer18a.html.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv:1612.01474 [cs, stat]*, Nov 2017. URL http://arxiv.org/abs/1612.01474. arXiv: 1612.01474.

Liu, T. and Tao, D. Classification with noisy labels by importance reweighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(3):447–461, Mar 2016. arXiv: 1411.7718.

Liu, Z. P. and Castagna, J. P. Avoiding overfitting caused by noise using a uniform training mode. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339)*, volume 3, pp. 1788–1793 vol.3, 1999.

Lyu, Y. and Tsang, I. W. Curriculum loss: Robust learning and generalization against label corruption. *arXiv:1905.10045 [cs, stat]*, Feb 2020.

Ma, X., Wang, Y., Houle, M. E., Zhou, S., Erfani, S. M., Xia, S.-T., Wijewickrema, S., and Bailey, J. Dimensionality-driven learning with noisy labels. *arXiv:1806.02612 [cs, stat]*, Jul 2018.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., and et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, Feb 2015.

Natarajan, N., Dhillon, I. S., Ravikumar, P. K., and Tewari, A. Learning with noisy labels. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, volume 26, pp. 1196–1204. Curran Associates, Inc., 2013.

Nix, D. A. and Weigend, A. S. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 1, pp. 55–60 vol.1, 1994.

Peretroukhin, V., Wagstaff, B., and Kelly, J. Deep probabilistic regression of elements of so(3) using quaternion averaging and uncertainty injection. In *CVPR Workshops*, 2019.

Rasp, S., Pritchard, M. S., and Gentine, P. Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences*, 115(39): 9684–9689, Sep 2018.

Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., and Rabinovich, A. Training deep neural networks on noisy labels with bootstrapping. *arXiv:1412.6596 [cs]*, Apr 2015.

Rolnick, D., Veit, A., Belongie, S., and Shavit, N. Deep learning is robust to massive label noise. *arXiv:1705.10694 [cs]*, Feb 2018.

Sai, Y., Jinxia, R., and Zhongxia, L. Learning of neural networks based on weighted mean squares error function. In *2009 Second International Symposium on Computational Intelligence and Design*, volume 1, pp. 241–244, Dec 2009. doi: 10.1109/ISCID.2009.67.

Shalizi, C. R. *Advanced Data Analysis from an Elementary Point of View*, 2019. URL https://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/ADAfaEPoV.pdf. (Accessed November 13th, 2020).

Shen, Y. and Sanghavi, S. Learning with bad training data via iterative trimmed loss minimization. *arXiv:1810.11874 [cs, stat]*, Feb 2019.

Shu, J., Xie, Q., Yi, L., Zhao, Q., Zhou, S., Xu, Z., and Meng, D. Meta-weight-net: Learning an explicit mapping for sample weighting. *arXiv:1902.07379 [cs, stat]*, Sep 2019.

Song, H., Kim, M., Park, D., and Lee, J.-G. Learning from noisy labels with deep neural networks: A survey. *arXiv:2007.08199 [cs, stat]*, Jul 2020.

Song, Y. and Zhang, Z. *UTKFace, Large Scale Face Dataset*, 2017. URL https://susanqq.github.io/UTKFace/. (Accessed June 11, 2020).

Tanno, R., Saeedi, A., Sankaranarayanan, S., Alexander, D. C., and Silberman, N. Learning from noisy labels by regularized estimation of annotator confusion. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11236–11245. IEEE, Jun 2019.

Thrun, S., Burgard, W., and Fox, D. *Probabilistic Robotics*. Intelligent robotics and autonomous agents. MIT Press, 2006.

Van Horn, G., Branson, S., Farrell, R., Haber, S., Barry, J., Ipeirotis, P., Perona, P., and Belongie, S. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 595–604. IEEE, Jun 2015.

Wang, Y., Ma, X., Chen, Z., Luo, Y., Yi, J., and Bailey, J. Symmetric cross entropy for robust learning with noisy labels. *arXiv:1908.06112 [cs, stat]*, Aug 2019.

Xie, J., Kiefel, M., Sun, M.-T., and Geiger, A. Semantic instance annotation of street scenes by 3d to 2d label transfer. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Xie, J., Kiefel, M., Ming-Ting, S., and Gieger, A. Kitti-360, 2020. URL http://www.cvlibs.net/datasets/kitti-360/.

Yi, K. and Wu, J. Probabilistic end-to-end noise correction for learning with noisy labels. *arXiv:1903.07788 [cs]*, Mar 2019.

Yu, X., Han, B., Yao, J., Niu, G., Tsang, I. W., and Sugiyama, M. How does disagreement help generalization against label corruption? *arXiv:1901.04215 [cs, stat]*, May 2019.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *arXiv:1611.03530 [cs]*, Feb 2017.

A. Appendix - Use cases

In this section, we analyze the different use cases where the labelling process in regression cause heteroscedastic noise with a variance estimate, using the concepts introduced in section 2.

Labelling from sensor readings, population studies, or simulations: Imagine you want build a dataset of pictures $\mathbf{x}_i$ from a camera on the ground labelled with the position y_i of a drone in the sky. To estimate the position of the drone at the moment the picture was taken, you could use state estimation algorithms based on the Bayes' filter (Thrun et al., 2006). These algorithms take as an input $\mathbf{z}_i$ the measurements from the drone's sensors, and provide a full posterior distribution over the state, sometimes under a Gaussian assumption for Kalman filters for example. The uncertainty depends on the precision of the sensors, the observability of a given state, the precision of the dynamic model, and the time since sensor signals were received. Similarly, studies based on population such as polling or pharmaceutical trials have quantified uncertainties based on the quantity and quality of their samples. It is also possible to train on simulators, as in climate sciences (Rasp et al., 2018) or in epidemiology (Alsdurf et al., 2020), and some of them provide their estimations' uncertainty based on the simulation procedure and the inclusion of real measurements in the model.

Using predictions from a neural network in complex neural architectures: In deep reinforcement learning for example, the critic network learns to predict a value from a state-action pair under the supervision of the heteroscedastic noisy output of a target network plus the reward (Mnih et al., 2015; Haarnoja et al., 2018). While the estimation of the uncertainty of the output of a neural network is not an easy task, it is an active field of research (Gal & Ghahramani, 2016; Peretroukhin et al., 2019). There, $\mathbf{z}_i$ is the state-action pair at the next step, and LG_j the target network being updated over time. The prediction is a mix of aleatoric and epistemic uncertainties as defined by Kendall & Gal (2017) which are dependent on both $\mathbf{z}_i$ and LG_j.

Crowd-sourced labelling: In the example case of age estimation from facial pictures, labellers Alice and Bob are given $\mathbf{z}_i = \mathbf{x}_i$ the picture of someone's face and are asked to estimate the age of that person. Age is harder to estimate for older people come (5 and 15 years of age are harder to confuse than 75 and 85) suggesting a correlation between $\sigma^2_{i,j}$ and $\mathbf{z}_i$. But Alice and Bob may also have been given different instructions regarding the precision needed, inducing a correlation between $\sigma^2_{i,j}$ and LG_j. Finally, there may be some additional interactions between $\mathbf{z}_i$ and LG_j, as for example Alice may know Charlie, recognize him on the picture and label his age with lower uncertainty. Both labellers can provide an estimation of the uncertainty around their labels, for example with a plus-minus range which can be used as a proxy for standard deviation.

B. Appendix - Data sets and Neural Networks

B.1. UTKFace

B.1.1. Dataset description

The UTKFace Aligned&Cropped dataset (Song & Zhang, 2017) consists of 20,000 pictures of faces labelled with their age, ranging from 0 to 116 years. We use it in a regression setting: the network must predict the age of a person given the photo of their face. Unless described otherwise, 16,000 images were used for training, and 4,000 for testing.

Some images are in black and white and some are in color. The pixel dimension of each image is 200x200.

Both the pixels and the labels were normalized before the training, so that their mean is 0 and standard deviation is 1 over the whole dataset. The noise variances were correspondingly scaled, as well as the cutoff threshold if applicable.

B.1.2. Neural network and training hyper-parameters

The model that we used was a Resnet-18 (He et al., 2015), not pretrained. It was trained with an Adam optimizer (Kingma & Ba, 2017), a learning rate of 0.001 over 20 epochs. A batch size of 256 was used in order to ensure the best performance for the L2 method with noisy labels as well as to reduce the time necessary to the training process.

B.2. Bike Sharing Dataset

B.2.1. Dataset description

The Bike Sharing Dataset (Fanaee-T & Gama, 2013) consists of 17,379 samples of structured data. It contains, for nearly each hour of the years 2011 and 2012, the date, season, year, month, hour, day of the week, a boolean for it being a holiday, a boolean for it being a working day, the weather situation on a scale of 4 (1: clear and beautiful, 4: stormy or snowy), the temperature, the feeling temperature, the humidity, and the wind speed, in the city of Washington DC. It also contains the number of casual, registered, and total bike renters for each hour as recorded on the Capital Bikeshare system.

We use it in a regression setting: the network must predict the total number of bike renters given the time and weather information. Unless described otherwise, 7,000 samples were used for training, and 3,379 for testing. We used less samples than available for training because the low-data situation, noise has a stronger effect on the performance.

The minimal test loss achieved with 7000 noiseless samples was very close to the one with 14000 samples, hinting that the additional samples did not give a lot of additional information.

We applied some pre-processing on the data to make it easier for the network to learn. First, the date was normalized from a scale between day 1 to day 730 to a scale between 0 and 4π. Then, we provided the network with the cosine and the sine of this number. This allowed to have the same representation for the same days of the year, while having the same distance between any two consecutive days, keeping the cyclic nature of a year. A similar idea was applied to hours, normalized from 0 to 2π instead of 0 to 24, and with the cosine and sine given to the network. The day of the week, being a category, was given as a one-hot vector of dimension 7. We also removed the season and the month as it was redundant information with the date.

Overall, the number of features was 19:

1	Year
2-4	Date (sine and cos)
4-5	Hour (sine and cos)
6-12	Days of the week (one-hot vector)
13	Holiday boolean
14	Working day boolean
15	Weather situation
16	Temperature
17	Felt temperature
18	Humidity
19	Wind speed

We observed that the network was learning significantly faster and better provided with this format for the data.

Both the features and the labels were normalized before the training, so that their mean is 0 and standard deviation is 1 over the whole dataset. The noise variances were correspondingly scaled, as well as the cutoff threshold if applicable.

B.2.2. Neural network and training hyper-parameters

The model that we used was a multi-layer perceptron with 4 hidden layers, the first one with 100 neurons, then 50, 20, and 10. The activation function was ReLU. We did not use any additional technique such as batch normalization as it did not improve the performances.

The model was trained over 100 epochs on mini-batches of size 256 for similar reasons than explained in section B.1.2, using the Adam optimizer with learning rate 0.001.

B.3. Other noise generation distributions

We present here two other distributions of $P(\sigma^2)$ which we have used to generate noise in the additional experiments shown in this document.

Uniform distribution The uniform distribution is characterized by its bounds a, b. Its expected value μ_P is the average of its bounds, and its variance $V = (b-a)^2/12$. As P only has support for $\sigma^2 \geq 0$, the maximum variance $V_{\max}$ is when $a = 0$ and $b = 2\mu_P$. While such a distribution is not realistic, it is simple conceptually and allows for interesting insights.

Binary uniform A more realistic distribution, which can also help us understand the effects of BIV, is when the data is generated from two regimes: low and high noise. We call the "binary uniform" distribution a mixture of two uniform distributions balanced by parameter p.

With probability p, the label is in a low noise regime: $\sigma^2 \sim U(0, 1)$, with expected value $\mu_l = 0.5$. With probability $1 - p$, the label is in a high noise regime: $\sigma^2 \sim U(a_h, b_h)$. a_h and b_h are chosen such that the average is μ_h and the variance of the high-noise distribution is determined by $V_h \in [0, V_{\max}]$. Note that, if we want a given expected value of the whole distribution μ_P, the value of μ_h changes depending on p: $\mu_h = (\mu_P - p\mu_l)/(1 - p)$

Therefore, the high-noise expected value μ_h of the noise variance σ^2 in a distribution with high p will be higher than the one for a low p, for the same value of μ_P. In other words, a higher p means more chance to be in the low-noise regime, but the high-noise regime is noisier.

C. Additional experiments

C.1. For L2 loss, mean variance is all that matters

We share an interesting insight for L2 loss with noisy labels which helps simplifying the analysis of the results. Under the unbiased, heteroscedastic Gaussian-based noise model presented in section 5.1, the only parameter of distribution $P(\sigma)$ that mattered to describe the performance of the L2 loss is its average μ_P, which is also the variance of the overall noise distribution. Independently of the distribution type, and the values of V, p and V_h, or α, as long as μ_P is equal, the L2 loss trained neural networks had the same performance. This is shown in Figure 3. For the sake of clarity, all the curves in this section were smoothed using moving average with a 35 steps window, and the shaded area represents the standard deviation over 10 runs.

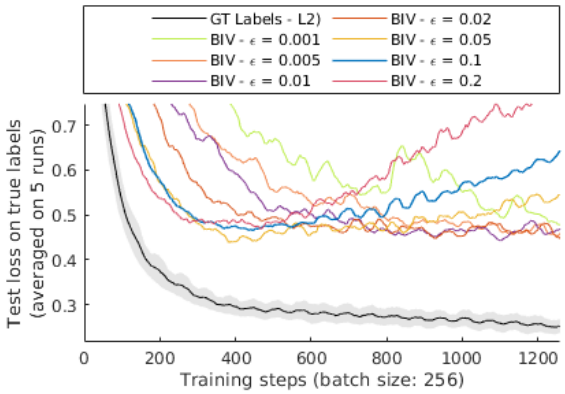

Figure 4. Results of running BIV with different values of ϵ on UTKF with $P(\sigma^2)$ as a Gamma distribution with $\alpha = 1, \mu_P = 2000$.

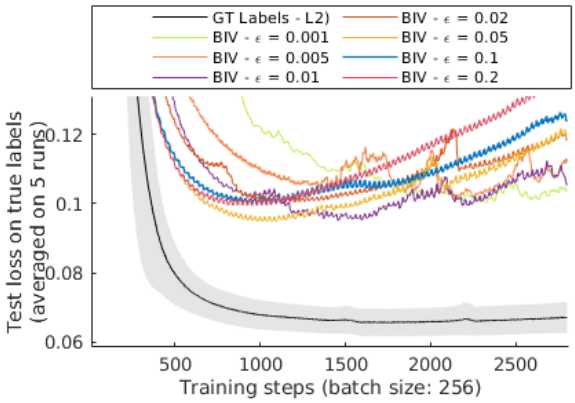

Figure 5. Results of running BIV with different values of ϵ on BikeSharing with $P(\sigma^2)$ as a Gamma distribution with $\alpha = 1, \mu_P = 20000$.

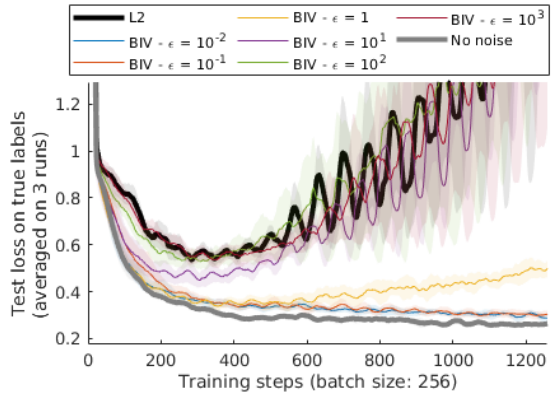

Figure 6. Impact of ϵ when training with highly noisy labels using BIV loss on UTKF dataset. The variance was sampled through a binary uniform distribution with $p = 0.5$, $\mu_P = 2000$, and $V_h = 0$. Very high ϵ shows a loss of performance as BIV approaches the L2 results.

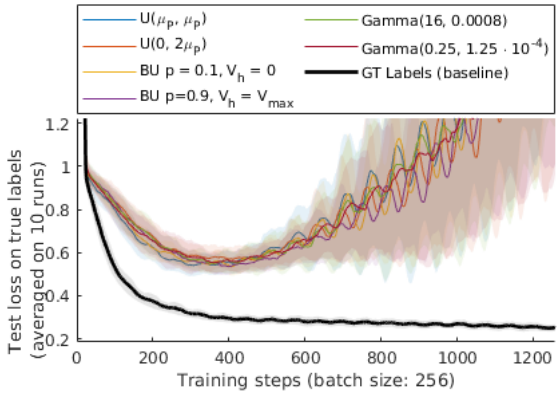

Figure 3. Performance of the neural network on UTKF trained with L2 loss, for different $P(\sigma^2)$ with constant $\mu_P = 2000$. No matter the distribution type or parameters, the performance is similar.

C.2. The influence of ϵ

In this section, we provide experimental results justifying our recommendation of the range of $[10^{-2}; 10^{-1}]$ for ϵ. In the experiments presented in this article, we have mainly used $\epsilon = 5 \times 10^{-2}$, and sometimes $\epsilon = 10^{-1}$. ϵ must be chosen as part of a trade-off between mitigating the effect of BIV with near ground-truth labels while keeping its effect with noisy labels. To better understand these results, it is important to remember that ϵ is added to the variance that is used in the loss function. When the labels are normalized - which is the case in our work -, the noise and its variance for each label is normalized too. The value we recommend for ϵ should therefore be valid for any normalized set of labels.

The main role of ϵ is to set a maximal weight to the samples and prevent a near-zero variance sample to effectively reducing the minibatch to itself, thus ignoring the other potentially valid samples. We tested several values of ϵ on both the UTKF and BikeSharing datasets, with $P(\sigma^2)$ being Gamma distributions with $\alpha = 1$ and $\mu_P = 2000$ and $\mu_P = 20000$ respectively.

The resulting graph can be seen in figures 4 and 5

In both experiments, the optimal value for ϵ is 0.05. Between 0.02 and 0.1, the performances are acceptable. When ϵ is too small, the learning process gives too much importance to the low-noise samples, as seen in section 6.2. When it is too high, it leans too much towards L2.

This can be seen in figure 6, which applies BIV with different value for ϵ on a binary distribution on UTKF.

The fact that in the two datasets, the same values for ϵ are optimal, even though tasks as well as the noise distribution are different, shows that these values are valid for most

cases with normalized labels. As ϵ can be seen as a minimal variance, scaling it for other label distributions is straightforward: it suffices to multiply it by the variance of the label distribution.

C.3. BIV on different distributions

C.3.1. Uniform distributions

We present in figure 10 the results of the experiment with uniform distributions as detailed in section B.3.

We observe that BIV and L2 have the same performances when $V = 0$ (and $a = b = \mu_P$). This is to be expected, as all samples have the exact same noise variance and thus the same weights. When $V = V_{max}$ ($a = 0$ and $b = 2\mu_P$), BIV has an advantage, as it is able to differentiate the samples and use the support of low-noise labels. When $V = V_{max}/2$ ($a = 0.293\mu_P$ and $b = 1.707\mu_P$), the difference between the samples is less important, and BIV only does a bit better than L2 on BikeSharing. On UTKF, the process has more variability and it is difficult to detect this effect.

In all cases, the benefit from using BIV is less important than with Gamma distributions with $\alpha \leq 1$, where the support on low-noise samples is higher.

C.4. Binary uniform distributions: BIV acts as a filter

With the binary uniform distribution described in section B.3, the noise is split in two regimes, with high or low variances. In this case, our results show that BIV performs better than L2, and actually similarly to the cutoff loss presented in section 4.3 with a threshold $C = 1$.

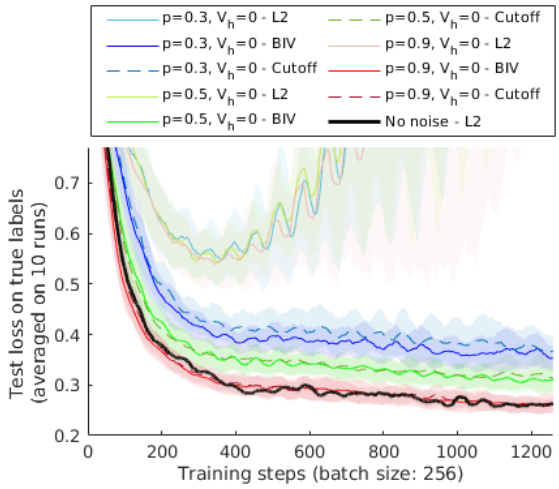

Figure 7. Comparison between BIV, Cutoff and L2 losses for binary uniform distributions of variance with different ps for $\mu_P = 2000$ and $V_h = 0$ on UTKF.

Figure 7 compares the test losses on UTKF with different values of p for $V_h = 0$. While the L2 curves are strongly impacted by the noise, both the BIV and cutoff losses lead to better and very similar performances for a given p. When $p = 0.3$, there are not a lot of information that can be used, and the performance is still impacted by the noise. When $p = 0.9$, there is nearly as much near-ground-truth data as in the noiseless case, and the performance is comparable.

In the case of binary uniform distributions, BIV is acting as a filter, cutting off labels which are too noisy to contain any useful information.

C.5. BIV on Gamma distributions

C.5.1. $\alpha \leq 1$

As described in section 6.1, the smaller α, the better the performance of BIV and cutoffs. We show in figures 12 and 13 the curves that led to the numbers in Table 1. BIV consistently outperforms the other methods. The performance of cutoff methods strongly depends on C, and the best value of C is not the same for every distribution P.

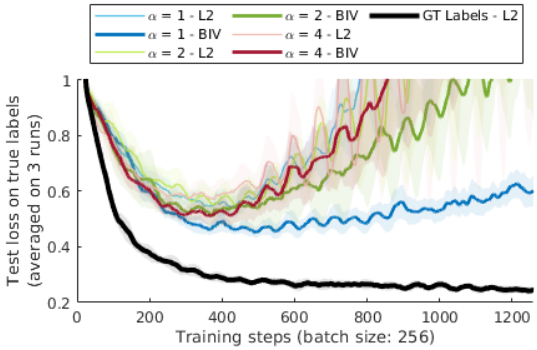

Figure 8. Test loss on the UTKF dataset for L2 and BIV learning on Gamma function with $\alpha \geq 1$.

C.5.2. $\alpha > 1$

When $\alpha > 1$, the highest support of $P(\sigma^2)$ shifts towards μ_P. This makes the samples less distinguishable for BIV and therefore the benefits of using it are reduced. This is shown in Figure 8 on UTKF.

C.5.3. Ablation study

In figure 9 showing the training loss with and without ϵ and normalization, it is evident that with ϵ, the variability of the loss is smaller.

C.6. Robustness of BIV

C.6.1. Size of the mini-batches

In equation (4), the normalization constant is computed from the samples in the mini-batch. If the distribution of the

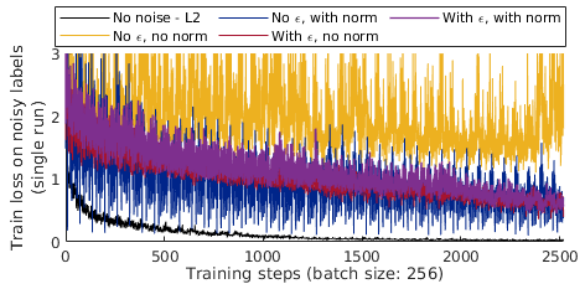

Figure 9. Training loss for ablation study for ϵ and normalization for both datasets where $P(\sigma^2)$ is a Gamma distribution with $\alpha = 1$. Note that, because the normalization value depends on ϵ, these curves should not be compared quantitatively but quantitatively, by looking at their respective variability.

noise variances in mini-batch is representative of the whole training dataset, the relative weight given to each sample with respect to the others is the same than if the normalization was made over the whole dataset. The larger the mini-batch, the more representative it is. In our experiments, we used a size of 256, which is arguably high. We tested our algorithm with lower batch sizes, from 16 to 128, to see if it was a critical factor in the performances.

The results are presented in figure 14. In UTKF, the batch size does not make any significant difference in the performance with respect to the amount of samples seen, except for a slightly steeper overfitting once the best loss has been achieved. In BikeSharing, a smaller batch size makes the training faster with respect to the amount of samples, but with a higher minimal loss, for both L2 and BIV. While a larger batch size leads to a lower loss function, the effect of BIV compared to the corresponding L2 curve is not compromised by smaller batch-sizes.

Two main factors may explain this robustness. First, a mini-batch of size 16 seems to be already representative enough of the whole dataset for the purpose of normalization. Second, the fact that the mini-batches are populated differently at every epoch improves the robustness as a sample who would have been in a non-representative batch at one epoch may not be at another epoch. In any case, the size of the mini-batch is not a critical factor for BIV.

C.6.2. Noisy variance estimation

In many scenarios, the variance σ^2 from which the noise was sampled is estimated, or inferred from a proxy, and therefore prone to be noisy itself. We tested the robustness of our method to such variance noise. In this experimental setup, the value given to the BIV algorithm is disturbed by noise $\delta_{\sigma_i^2}$. We modelled this noise on σ_i^2 to be sampled from a normal distribution whose standard deviation is proportional to σ_i^2 with a coefficient of variance disturbance D_v:

$$\delta_{\sigma_i^2} \sim \mathcal{N}(0, D_v \sigma_i^2 / 9) \tag{7}$$

Dividing σ_i^2 by 9 allows to scale D_v so that, when $D_v = 1$, $\delta_{\sigma_i^2} < -\sigma_i^2$ is at 3 standard deviations from the mean.

We then compute the noisy variance, which needs to be positive, as $\tilde{\sigma}_i^2 = \left|\sigma_i^2 + \delta_{\sigma_i^2}\right|$. The noise is therefore biased, but when $D_v \leq 1$, this is negligible as it happens with probability less than 0.15%.

The results presented in figure 15 show that, when $D_v \leq 1$, BIV is robust to such noise. While a higher D_v leads to lower performance, the impact is small compared to the effect of BIV. However, when $D_v = 2$, which is an arguably high level of noise and leads to bias as explained previously, the beneficial effect of BIV is significantly affected in BikeSharing, and completely disappears in UTKF.

In this setting, we also show as shown in figure 11 that cutting off the noisy data is not a good strategy, as it always performs worse than L2.

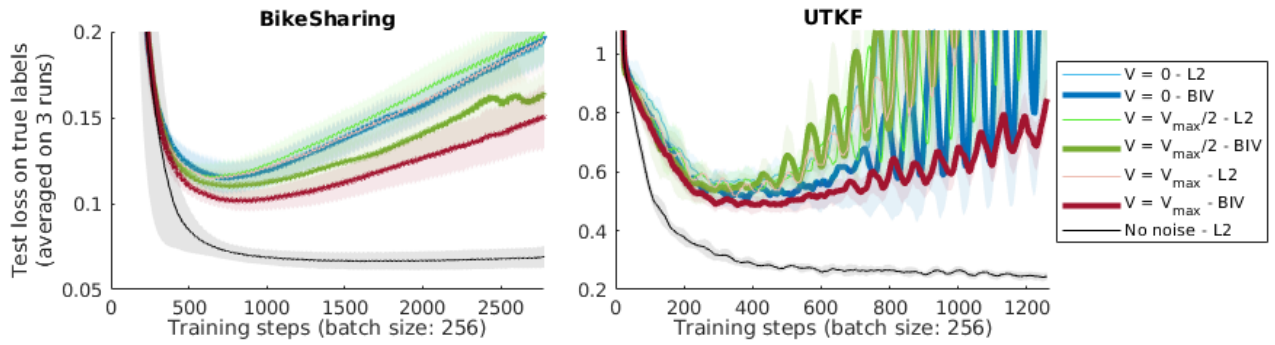

Figure 10. Test loss for L2 and BIV learning on uniform with different variances V.

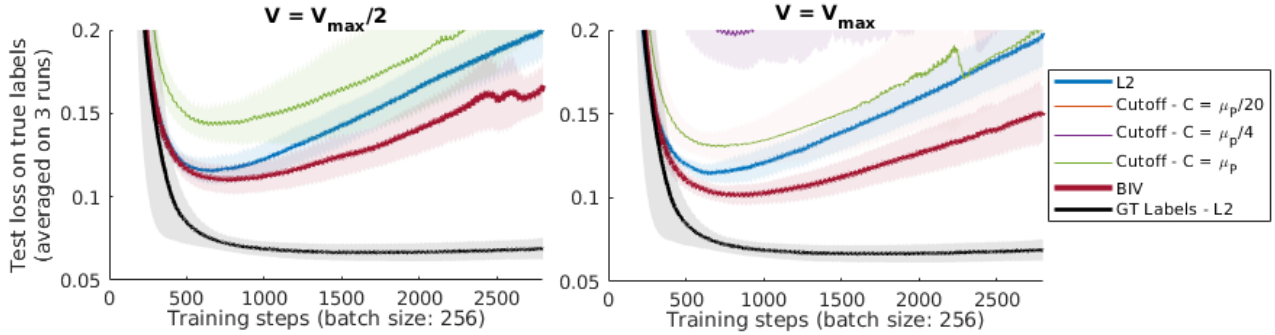

Figure 11. On BikeSharing with $\mu_P = 20000$, using cutoff is not helpful in the uniform setting. This is due to the significant loss of information induced by such a strategy.

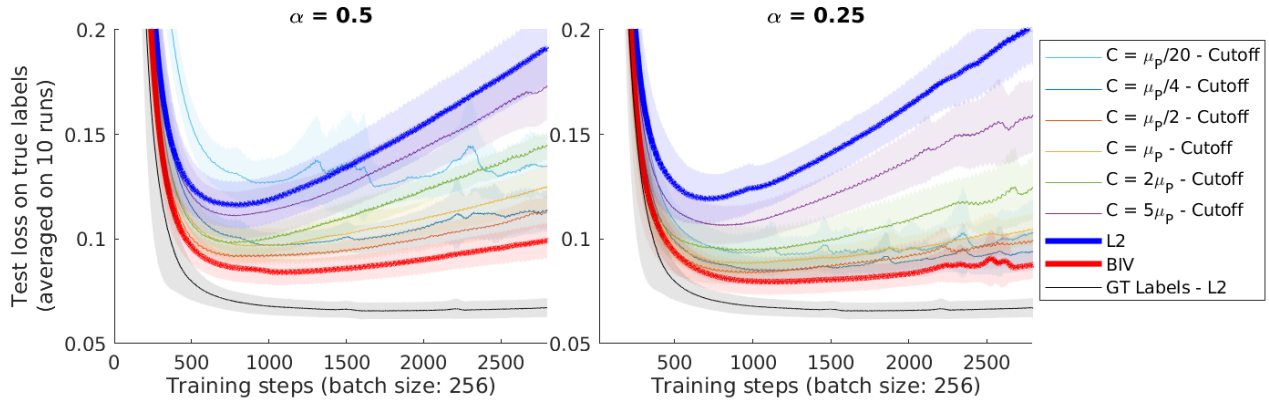

Figure 12. Test loss on the BikeSharing dataset, with $\alpha \leq 1$

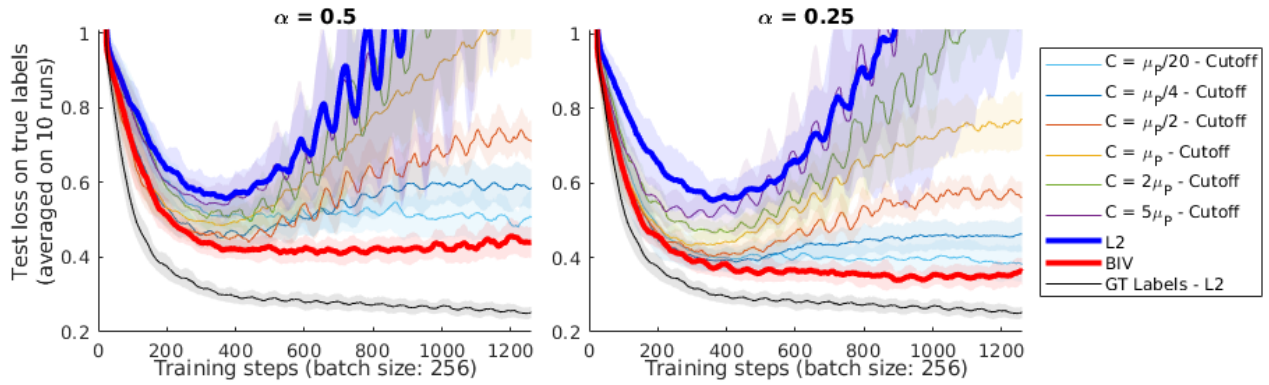

Figure 13. Test loss on the UTKF dataset, with $\alpha \leq 1$

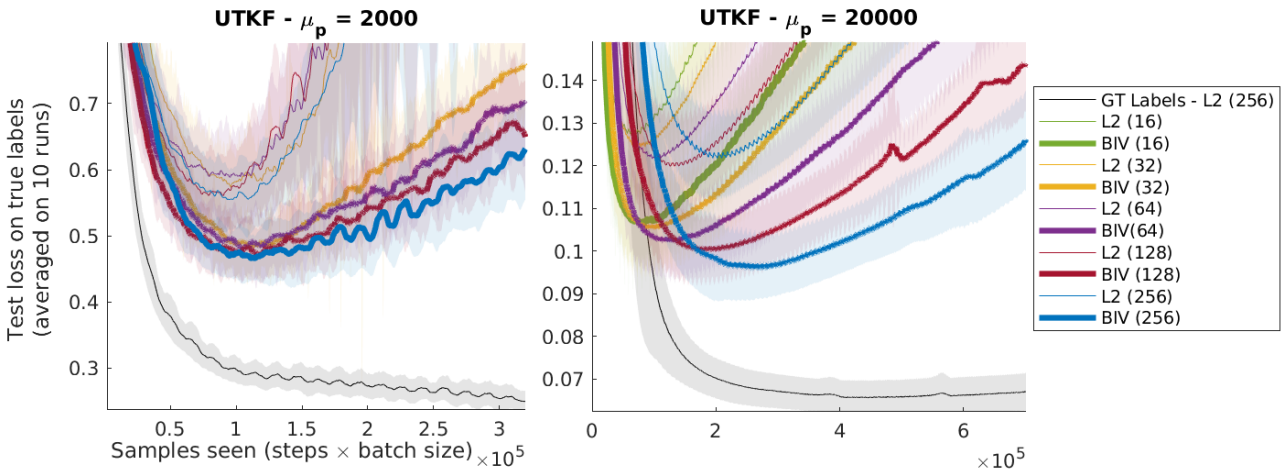

Figure 14. BIV with different batch sizes in both UTKF and BikeSharing datasets.

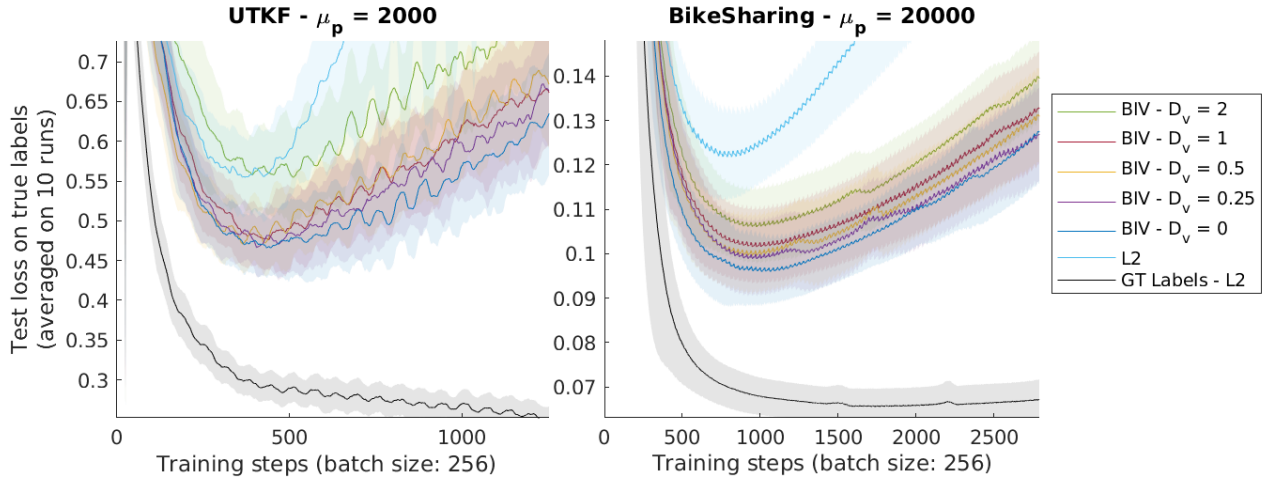

Figure 15. Robustness of BIV to noise in the variance with different disturbance coefficients D_v.