The Effects of Label Errors in Training Data on Model Performance and Overfitting

Nicholas Pellegrino[*1] **Nolen Zhao**[*1,2] **Paul Fieguth**[1]
[1]Vision and Image Processing Group, Systems Design Engineering, University of Waterloo
[2]Mechanical & Mechatronics Engineering, University of Waterloo
{npellegr,n37zhao,pfieguth}@uwaterloo.ca

Abstract

Training data used in machine learning applications are often assumed to be perfect, i.e., do not contain *any* errors; however, this is almost never the case and may lead to limitations in the resulting model performance. In this paper, the effects of the presence of label errors in training data are studied quantitatively and in relation to model overfitting. By artificially creating label errors, it is observed that a constrained (small) CNN model exhibits remarkable generalizability — retaining high accuracy even when *most* data are mislabelled! Test accuracy catastrophically falls only for unrealistically high label error rates, at a point related to the number of classes present in the data. These preliminary experiments pave the road towards further studies of model robustness, possibly offering a quantitative method through which to compare models.

1 Introduction

In supervised learning problems, a set of labelled data, known as training data, are required to optimize / train the model [1, 2]. Deep neural networks, including convolutional neural networks (CNNs) [3, 4], consist of layers of interconnected artificial neurons with associated weights which must be optimized in order to train the model. Machine learning engineers normally assume that the "ground truth" training data are labelled *correctly*; however, this is not necessarily the case, and in fact is often *not* the case! Indeed, in many benchmark datasets, label errors are present in rates on the order of 5% [5], for example, in ImageNet [6]. In biological data, for example, the recently introduced BIOSCAN-1M Insect Dataset [7], where images of insects are labelled according to their taxonomy, the presence of labelling errors is nearly inevitable given the difficulty of the taxonomic assessment problem [7, 8] and human error. In cases where training data label errors exist, one must ask the question of how model performance ought to be evaluated, and what it means to achieve a particular percentage accuracy when some (likely unknown) fraction of labels are incorrect.

For illustration purposes, two versions of labelled data from a simple 2-class problem are pictured in the top row of Figure 1. In the first column, the data contains outliers, and in the second column, there are label errors. Data point shape (circle *vs.* triangle) indicates the true class and colour (red *vs.* blue) indicates the ground truth (training) label. Note that while outliers and mislabelled data may appear to be similar, the two arise from completely different causes and will impact classification models differently. Assuming data are clustered with high density, surrounding some prototypical center point, outlier points are those that are far from their true class's center, whereas mislabelled data may appear *anywhere* but actually are often (due to the assumption of high density) near their true class's center. The presence of mislabelled data may lead to local classification error, especially in cases of overfitting. Indeed, a more *local* classifier, such as a nearest-neighbour scheme (shown in middle row of Figure 1), would be highly susceptible to overfitting and local errors, whereas a more *global* classifier, for example a 5-nearest-neighbour scheme (shown in bottom row of Figure 1), would be less susceptible to overfitting and local classification errors due to its reliance on the consensus of multiple training data points.

While simple nearest neighbour classification schemes may be easy to envision and intuitively understood for simple problems such as that of Figure 1, the behaviours of deep-neural-network-based classifiers on real-world problems are not. This paper studies the impact of having mislabelled training data by artificially corrupting the training data from a familiar benchmark dataset, MNIST [9], and then training and evaluating a simple CNN model. By setting the corruption rate, evaluations of model overfitting are made in a very controlled environment. Techniques shown here may also lend themselves toward the determination of whether a particular model *type* may be more or less robust to the presence of training label errors.

[*]Indicates equal contribution, joint first-authorship.

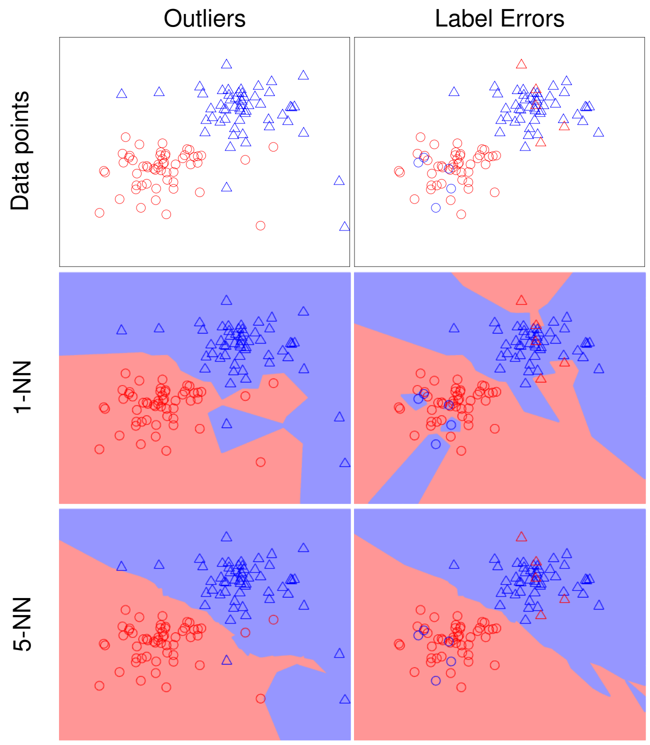

Fig. 1: Training data may contain both outliers and label errors. The two columns include versions of a 2-class dataset: one with outliers and the other with label errors. The first row shows the data points, while the subsequent rows show nearest-neighbour (1-NN) and 5-nearest-neighbour (5-NN) classification regions. Data point shape (circle *vs.* triangle) indicates *true* class and colour (red *vs.* blue) indicates ground truth *label*. Outliers and mislabelled data may appear to be similar, but arise from completely different causes.

2 Background

As introduced in Section 1, biological data are especially prone to mislabelling due to their complex nature. In particular, the BIOSCAN project [10] is an ecologically important and relevant research effort in which the presence of label errors must be considered. In the BIOSCAN project, insects are hand-labelled by taxonomic experts who make their assessment based on captured images. The main difficulty here, ignoring the requirement for a high level of expertise, is the lack of consensus and certainty about the taxonomy of life itself (i.e., the locations and numbers of branches / subcategories within the tree-like hierarchy). Fundamentally, the taxonomic categorization of life is based on theory more so than an observable underlying structure. Indeed, much controversy may be found within the community of taxonomists! Nonetheless, it is accepted that a hierarchical structure does exist and may eventually be largely uncovered. Therefore, the notion of what should be considered an error is somewhat vague. Errors may arise as a result of human error (e.g., labelling two examples of the same species as being of different taxa), or as a result of simply not knowing in which category a given example belongs (e.g., labelling an example (or entire group of examples) as being part of a given category, when in fact it would better fit elsewhere). In the BIOSCAN-1M Insect Dataset, the error rate is unknown; however, there is no doubt that some errors are present.

To address the presence of label errors, in 2021, Northcutt *et al*. developed a method for automatically detecting and correcting errors in training data, known as Confident Learning [5, 11]. In doing so, benchmark datasets including MNIST [9], CIFAR [12], ImageNet [6], and more were examined, and possible mislabelled examples were identified. Crowd-sourcing (Mechanical Turk) was then used both to verify which selected examples were indeed incorrectly labelled, and to propose a corrected label through consensus. These results are available at `labelerrors.com` and provide a valuable resource for those in the field. While this work provides one possible path forwards in contending with label errors in training data, little is known about the behaviour and robustness of specific deep neural network architectures in terms of handling label errors.

3 Preliminary Experiments & Results

Experiments are conducted upon the MNIST dataset, known to have a *very* low error rate (0.15%) [5] due to its simplicity. To evaluate the impact of having increased error rates on model accuracy, the training partition of the dataset is artificially corrupted. Data are re-labelled according to a specified *corruption rate*, $r_c \in [0,1]$. Whether any given example is re-labelled is determined randomly according to whether a random number drawn (from a uniform distribution) is less than r_c. In this manner, over large quantities of data, the proportion of re-labelled data approximates r_c. Note that if selected, an example's label is *necessary* changed, i.e., made incorrect. Throughout all experiments, model and training hyperparameters are set according to values specified in Table 1. To keep experiments simple, a minimalistic model based on an introductory example from PyTorch [13] capable of achieving $> 99\%$ accuracy on the MNIST dataset was selected. The model used is a CNN consisting of two convolutional layers, followed by max pooling, dropout, a fully connected layer, dropout, and a final fully connected layer. In total, the model has only 1.2 M trainable parameters.

Table 1: Hyperparameters used for experiments.

Parameter	Setting
Loss function	Cross-Entropy
Optimizer	SGD with momentum
Learning rate	0.01
Momentum	0.9
Batch-Size	64
Num. Epochs	12

Firstly, the model validation accuracy is examined as a function of training data corruption rate, shown in Figure 2. Observe that accuracy remains approximately steady and high (over 95%!) until a corruption rate of approximately $r_c = 0.9$, where an abrupt downward change occurs, before settling-out once again. This finding is quite remarkable, given that the model continues to be accurate even when *most* training data is mislabelled! The abrupt change seems to correspond to the transition point at which for any class, the number of labels indicating the *correct* class equals the number of labels for any other, *incorrect*, class. Before this point, the model still tends to learn the correct class-label association, and performs quite well. After this point, the model has overfit to the mislabelled data and performs poorly during testing. In terms of the number of classes within the dataset ($M = 10$, for MNIST), the relationship determining the location of this catastrophic change in model behaviour appears to be $r_c' = 1 - 1/M$.

To verify the relationship between the location of the abrupt change and the number of classes, a similar experiment over which the number of classes is artificially reduced is conducted. Here, accuracy results for the original 10-class problem are shown alongside

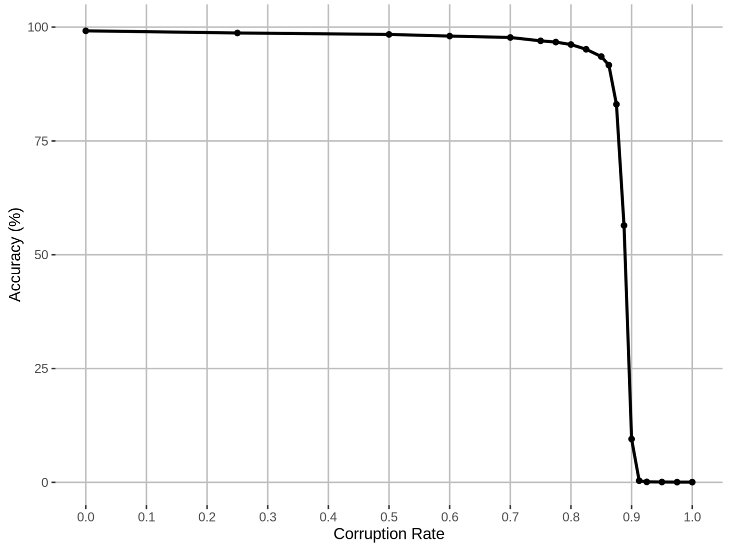

Fig. 2: Model accuracy as a function of training data corruption rate. Accuracy remains remarkably high even when *most* training data are mislabelled! Until the corruption rate nears 0.9, model performance is hardly affected. Beyond this point, there becomes fewer labels of the *correct* class than any other, *incorrect*, class, and accuracy plummets towards zero.

those of a 6-class and 2-class problem. In each case, for the general M-class problem, examples from the first M classes of MNIST are retained, omitting the remainder. Figure 3 shows the results of this experiment which indeed confirm that the point at which the catastrophic change occurs is related to the number of classes through $r_c' = 1 - 1/M$.

To gain further insight, training and testing loss are examined in Figure 4. Notice that while both losses do increase with increasing corruption rate, the testing loss remains below the training loss until a cross-over point at $r_c = 0.9$, demonstrating for this range of corruption rates that the model performs better during testing than it does during training and is able to generalize quite well (i.e., not overfit) in spite of large quantities of mislabelled data. At $r_c = 0.9$, the cross-over point, for each true class, there are approximately equal numbers of training samples labelled as all ten classes, and the model learns to randomly guess, thereby resulting in equal loss during training and testing. Beyond the cross-over point, *fewer* training samples of each given class are labelled correctly than *all other* classes, the model learns to *not* estimate the correct class (i.e., has overfit to mislabelled data), training loss plateaus, and testing loss spikes.

4 Discussion

In Figure 2, accuracy tapers quite gradually for modest (i.e., realistic) corruption rates, for example $0.05 < r_c < 0.3$. This insensitivity to corruption rate indicates that the model is able to generalize well, and may be a feature useful as a point of comparison between model types. Models for which accuracy decreases at a greater rate would have a greater tendency to overfit and would generalize more poorly than those models for which accuracy decreases more gradually.

Comparing loss with accuracy, in Figure 4, loss increases gradually with corruption rate, when meanwhile in Figure 2, the accuracy is almost invariant to corruption rate until a point at which there is a catastrophic and large change. This behaviour in accuracy seems to contradict what is seen in the loss:

Why is it that loss changes by only a small amount (specifically surrounding the $r_c = 0.9$ point) when yet accuracy rapidly plummets from near 100% to near 0%?

This is a result of how inference is performed and how cross-entropy loss is defined. The model outputs (after running through SoftMax) a set of predicted class probabilities, $\{\hat{p}_i\}_{i\in[1,10]}$. The class with the

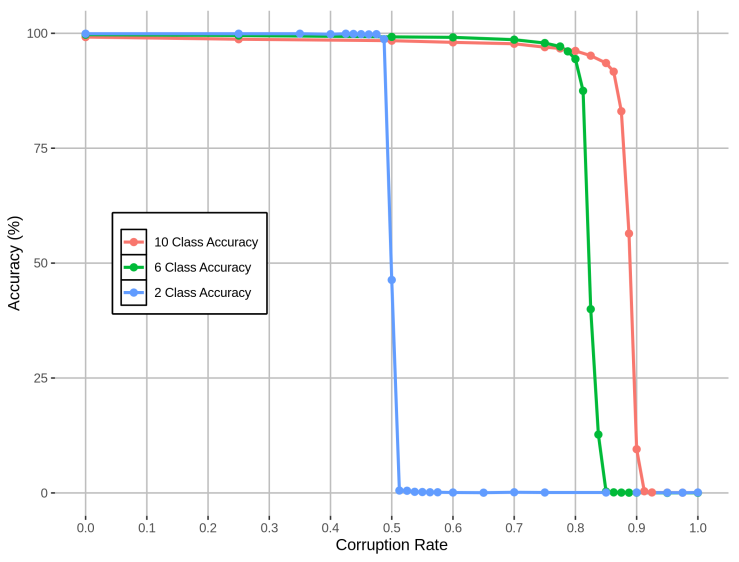

Fig. 3: Similar to the accuracy *vs.* corruption rate plot of Figure 2, model accuracy is evaluated for a 10-class, 6-class, and 2-class problem. For the 10-class problem, the abrupt change occurs at a corruption rate of approximately 0.9, whereas for the 6-class problem, the abrupt change occurs at only 0.83, and for the 2-class problem, already at 0.5. This location follows a trend specified by $r_c^{\prime} = 1 - 1/M$, where M is the number of classes.

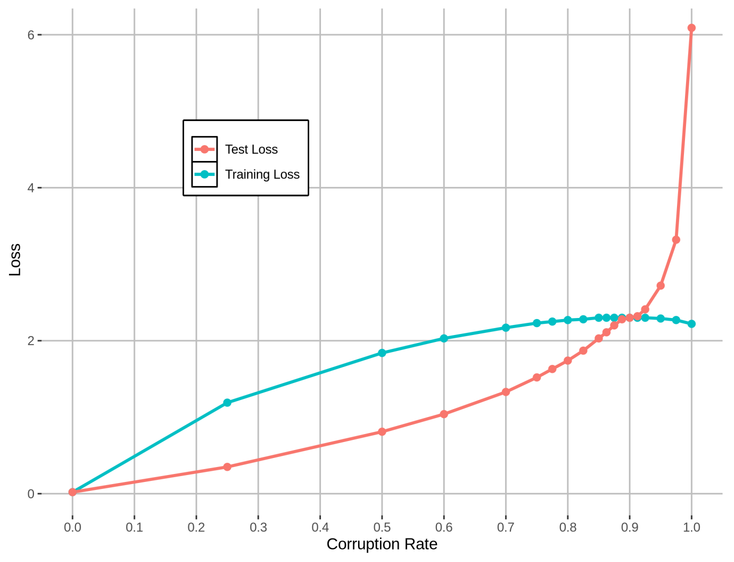

Fig. 4: Training and testing loss as a function of corruption rate. Observe the cross-over point at $r_c = 0.9$, whereby testing loss begins to exceed training loss. Training loss tends to plateau as false-labels tend towards being fully uncorrelated and then anti-correlated with the data itself, i.e., random but *not* correct. Testing loss initially is below training loss, as the model is still able to partially learn the correct class-label relationships (given that most data is still correctly labelled); however, beyond the cross-over point, most data is *not* labelled correctly, the model learns to *not* estimate the correct class, and testing loss spikes.

highest predicted probability is selected as the inferred class for a given input, i.e.,

$$\text{predicted class} = \arg\max_i \hat{p}_i. \tag{1}$$

So long as the predicted probability for the correct class, $\hat{p}_c$, is slightly higher that that of all others $\hat{p}_i,\ i \neq c$, the network will infer the correct class. As corruption rates increase towards $r_c = 0.9$, fewer and fewer samples are correctly labelled, and the predicted class probabilities tend towards that of a uniform random distribution. Just prior to $r_c = 0.9$, the amount of correctly labelled data slightly exceeds the amount of incorrectly labelled data for each label, the predicted class probability for the correct class, $\hat{p}_c$, generally slightly exceeds that of all others (just greater than 0.1), and the model tends to still correctly classify testing data correctly. However, cross-entropy loss computes the negative natural log of the predicted correct class probability, $\hat{p}_c$, averaged over all samples, indexed by n, in a batch of size N,

$$J_{CE} = -\frac{1}{N}\sum_{n=1}^{N} \ln\left(\hat{p}_c\right). \tag{2}$$

Notice that $-\ln(0.1) \approx 2.3026$, almost exactly the loss seen at the cross-over point, at $r_c = 0.9$. The negative log of the predicted correct class probability, $-\ln(\hat{p}_c)$, is smooth and does not exhibit large change surrounding the point $\hat{p}_c = 0.1$, whereas the highly non-linear class selection method of Equation (1), which simply selects the class with highest predicted probability, abruptly changes as $\hat{p}_c$ decreases below 0.1, leading to a near instantaneous loss in accuracy.

While it is totally unrealistic to assume that models are being trained with data having error rates towards $r_c = 0.9$ in practice, the resulting observed trends in accuracy *vs.* corruption rate do reveal a great deal about the *robustness* of a particular model to the presence of label errors. Having robustness to mislabelled data indicates that a model is better able to generalize, and not overfit to mislabelled data. While only one model was explored in this study, this type of approach may be used to analyze and compare other prospective models for use in more complex classification problems in the real world, allowing a designer to discover which models or architectures are most susceptible to overfitting the dataset at hand, and select the most suitable one.

5 Conclusion

This study investigated the impacts of the presence of label errors in training data on model accuracy and training and testing loss. A simple CNN model was used, with data artificially corrupted in the MNIST dataset. Remarkably, the model continued to perform with high accuracy (over 95%) even when *most* training data was mislabelled! While cases of data with large error rates are highly unlikely in practice, similar investigations may be useful for machine learning engineers to learn more about which model architectures tend to generalize better and can be used to avoid overfitting to mislabelled data.

Much future work remains in the study of label errors and model overfitting. Investigations of

- more complicated models and classification problems (datasets),
- non-uniform error distributions (since label errors in real data are likely to exhibit some correlation), and
- constraints that may induce overfitting (e.g., limiting the amount of data)

will be performed in order to better understand the architectural features that make certain models more robust.

Acknowledgments

This research was enabled in part by support provided by Calcul Québec (`calculquebec.ca`) and the Digital Research Alliance of Canada (`alliancecan.ca`).

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), NSERC-PGS D, and NSERC Discovery Grant, funding reference number RGPIN-2020-04490.

Cette recherche a été financée par le Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG), CRSNG-ES D, et CRSNG Subvention à la Découverte, numéro de référence RGPIN-2020-04490.

Natural Sciences and Engineering Research Council of Canada
Conseil de recherches en sciences naturelles et en génie du Canada
Canada

References

[1] V. Nasteski, “An overview of the supervised machine learning methods,” *Horizons. b*, vol. 4, pp. 51–62, 2017.

[2] A. Mathew, P. Amudha, and S. Sivakumari, “Deep learning techniques: an overview,” *Advanced Machine Learning Technologies and Applications: Proceedings of AMLTA 2020*, pp. 599–608, 2021.

[3] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016. [Online]. Available: http://www.deeplearningbook.org

[5] C. G. Northcutt, A. Athalye, and J. Mueller, “Pervasive label errors in test sets destabilize machine learning benchmarks,” *NeurIPS 2021 Datasets and Benchmarks Track*, 2021.

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

[7] Z. Gharaee, Z. Gong, N. Pellegrino, I. Zarubiieva, J. B. Haurum, S. C. Lowe, J. T. McKeown, C. C. Ho, J. McLeod, Y.-Y. C. Wei *et al.*, “A step towards worldwide biodiversity assessment: The bioscan-1m insect dataset,” *arXiv preprint arXiv:2307.10455*, 2023.

[8] N. Pellegrino, Z. Gharaee, and P. Fieguth, “Machine learning challenges of biological factors in insect image data,” *Journal of Computational Vision and Imaging Systems*, vol. 8, no. 1, pp. 34–37, 2022.

[9] L. Deng, “The mnist database of handwritten digit images for machine learning research,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[10] “BIOSCAN,” Jun 2022. [Online]. Available: https://ibol.org/programs/bioscan/

[11] C. Northcutt, L. Jiang, and I. Chuang, “Confident learning: Estimating uncertainty in dataset labels,” *Journal of Artificial Intelligence Research*, vol. 70, pp. 1373–1411, 2021.

[12] A. Krizhevsky, “Learning multiple layers of features from tiny images,” Tech. Rep., 2009.

[13] PyTorch, “Basic mnist example,” Sep 2022. [Online]. Available: https://github.com/pytorch/examples/tree/main/mnist