# Iterative Label Improvement:

## Robust Training by Confidence Based Filtering & Dataset Partitioning

***International Conference On Pattern Recognition***

**Christian Haase-Schuetz[1,2], Rainer Stal[1], Heinz Hertlein[1], Bernhard Sick[2]**
[1]Engineering Cognitive Systems, Chassis Systems Control, Robert Bosch GmbH, Germany
[2] Intelligent Embedded Systems Group, University of Kassel, Germany

## Introduction

State-of-the-art, **high capacity deep neural networks** not only require **large amounts of labelled training data**, they are also highly susceptible to labelling errors in this data, typically resulting in large efforts and costs and therefore limiting the applicability of deep learning.

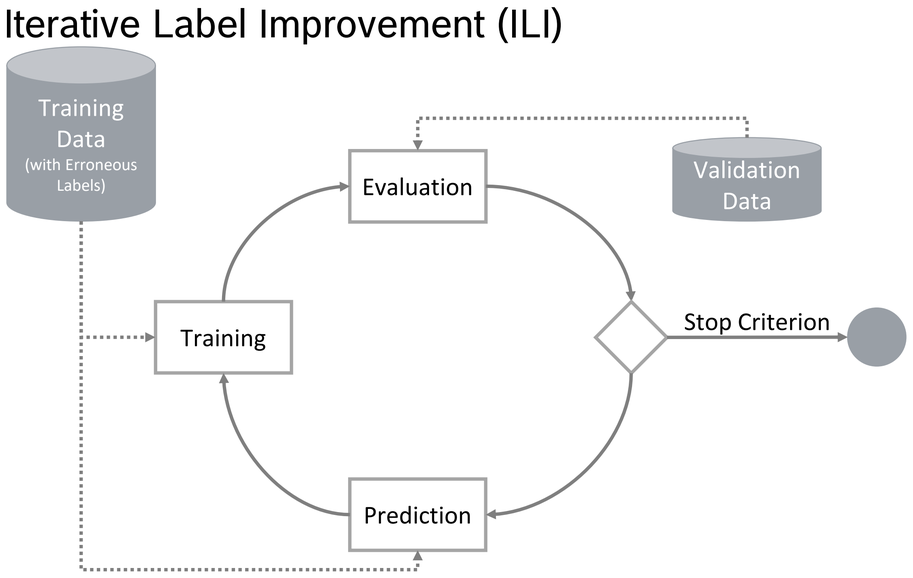
To alleviate this issue, we propose **a novel meta training and labelling scheme** that is able to use inexpensive unlabelled data by taking advantage of the generalization power of deep neural networks.

Using our proposed scheme of combining self-training with pseudo-labels, **both label quality and resulting model accuracy, can be improved significantly**.

Our method achieves **state-of-the-art results**, while being architecture agnostic and therefore broadly applicable.

## The Idea of ILI

- Iterative Label Improvement (ILI)




- Filters can be applied, e.g. confidence

$$\mathcal{F}^{(i)}_{\text{confidence}}\left(y^{(i)}_{\text{train}}\right) = \begin{cases} \hat{y}_{\text{train},i} = m^{(i)}(x_{\text{train},i}) : & c^{(i)} > \vartheta \\ \hat{y}_{\text{train},i} = y_{\text{train},i} : & \text{else.} \end{cases}$$

## Algorithm

- Plain

**Algorithm 1** plainILI filtered with initILI

**Input:** $X_{\text{train}}$, $y_{\text{train}}$, $n_{\text{iter}}$
**Output:** Model $m^{n_{\text{iter}}}$, $y^{n_{\text{iter}}}_{\text{train}}$

$m^{(0)} = m.initialize()$
$m^{(0)}.fit(X_{\text{train}}, y_{\text{train}})$
**for** $i = 1$ to $n_{\text{iter}}$ **do**
  $y^{(i)}_{\text{train}} = \mathcal{F}\left[m^{(i-1)}\right]\left(X_{\text{train}}, y^{(i-1)}_{\text{train}}\right)$
  $m^{(i)} = m.initialize()$
  $m^{(i)}.fit(X_{\text{train}}, y^{(i)}_{\text{train}})$
**end for**

- Partitioning based

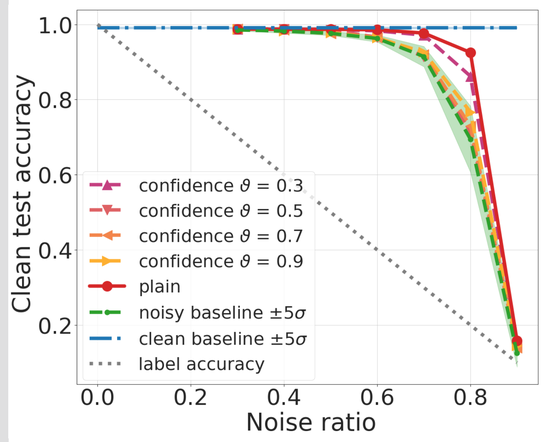
**Algorithm 2** opILI with initILI

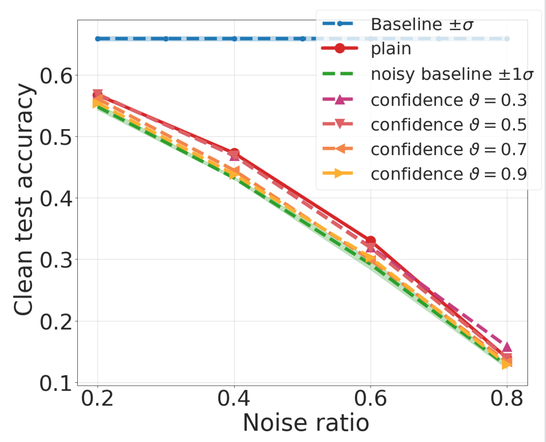
**Input:** $X_{\text{train, A}}$, $y_{\text{train}}$, $n_{\text{iter}}$
**Output:** Model $m^{n_{\text{iter}}}$, $y^{n_{\text{iter}}}_{\text{train}}$

$X_{\text{train, A}}, X_{\text{train, B}} = \text{Split}(X_{\text{train}})$
$m^{(0)}_{\text{A}}.initialize()$
$m^{(0)}_{\text{A}}.fit(X_{\text{train, A}}, y^{(0)}_{\text{train, A}})$
**for** $i = 1$ to $n_{\text{iter}}$ **do**
  $y^{(i)}_{\text{train, B}} = m^{(i-1)}_{\text{A}}(X_{\text{train, B}})$
  $m^{(i)}_{\text{B}}.initialize()$
  $m^{(i)}_{\text{B}}.fit(X_{\text{train, B}}, y^{(i)}_{\text{train, B}})$
  $y^{(i)}_{\text{train, A}} = m^{(i)}_{\text{B}}(X_{\text{train, A}})$
  $m^{(i)}_{\text{A}}.initialize()$
  $m^{(i)}_{\text{A}}.fit(X_{\text{train, A}}, y^{(i)}_{\text{train, A}})$
**end for**
$y^{n_{\text{iter}}}_{\text{train}} = \left[y^{(n_{\text{iter}})}_{\text{train, A}}, y^{(n_{\text{iter}})}_{\text{train, B}}\right]$

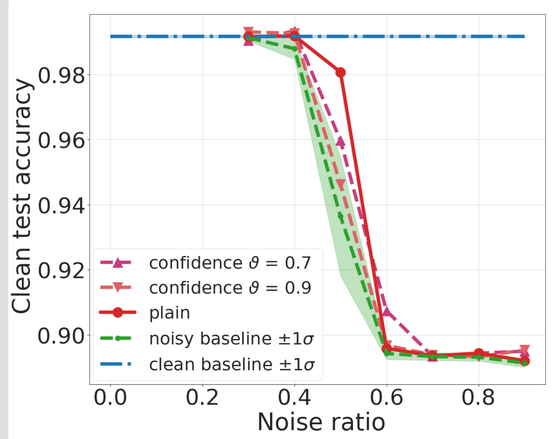
## Results

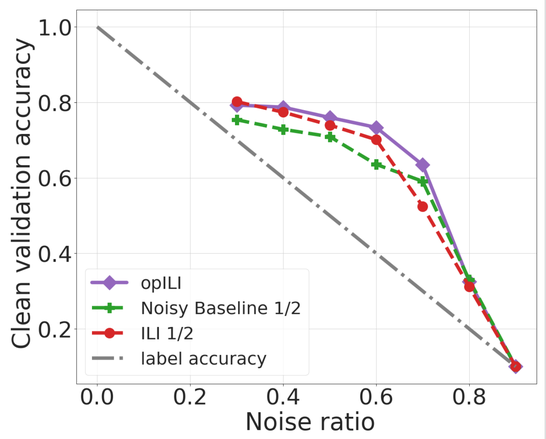
- We experiment with different Networks, MNIST-CNN [1], CIFAR-CNN [1], ResNet32 [2] and ResNet50 [2] on multiple datasets, MNIST [3], CIFAR10 [4] and CIFAR100 [4]
- MNIST-CNN on MNIST (left) and ResNet50 on CIFAR100 (right) with Random Errors



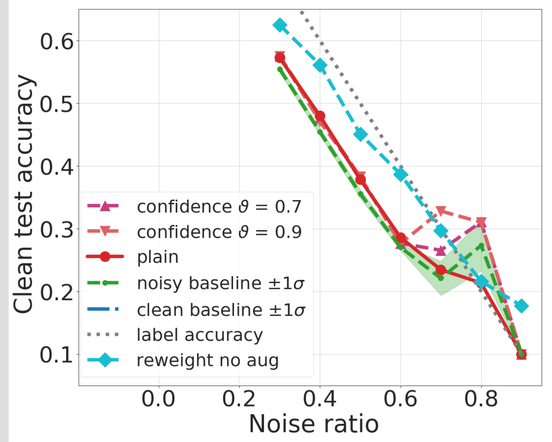



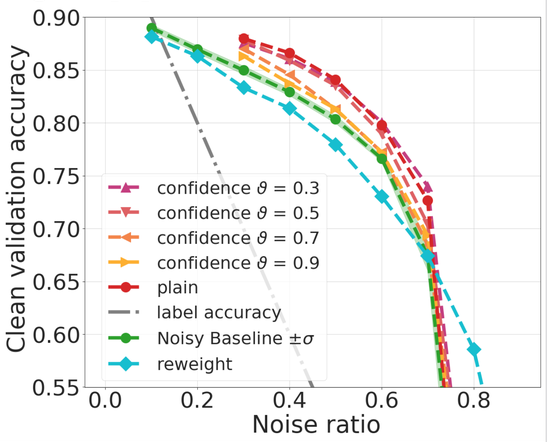

- CIFAR-CNN on MNIST with Bias Errors (left) and usage of ILI with ResNet32 on CIFAR10 for Semi-supervised learning (right)




- With (right) and without (left) data augmentation, data augmentation plays a key role, if it is applied, ILI outperforms "learning to reweight" [5]




## Take-home Message

- **ILI can clean labels and improve accuracy**
- **Data Augmentation plays a key role**
- Filtering needs to be adapted to dataset and model
- **Partitioning helps avoiding bias**
- **Partitioning based ILI is suitable for semi-supervised learning**

**Contact Info : christian.schuetz2@de.bosch.com**

[1] F. Chollet et al., "Keras," https://keras.io, 2015.
[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
[3] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner et al., "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.
[4] A. Krizhevsky, G. Hinton et al., "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.
[5] Ren, Mengye, et al. "Learning to reweight examples for robust deep learning." arXiv preprint arXiv:1803.09050 (2018).