Iterative Label Improvement: Robust Training by Confidence Based Filtering & Dataset Partitioning

International Conference On Pattern Recognition

Christian Haase-Schuetz1,2, Rainer Stal1, Heinz Hertlein1, Bernhard Sick2
1Engineering Cognitive Systems, Chassis Systems Control, Robert Bosch GmbH, Germany
2Intelligent 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
  \[ F_{\text{confidence}}(y_{\text{train}}, i) = \begin{cases} \hat{y}_{\text{train}, i} = m(i)(x_{\text{train}, i}) : & \varepsilon(i) > \theta \\ \hat{y}_{\text{train}, i} = y_{\text{train}, i} : & \text{else.} \end{cases} \]

Algorithm

- Plain
  Algorithm 1 plainILI filtered with instILI
  \[
  \text{Input:} \quad x_{\text{train}}, y_{\text{train}}, m, \theta, \varepsilon
  \]
  \[
  \text{Output:} \quad m_{\text{final}}, y_{\text{train}}
  \]
  1. \( m_{\text{init}} = m \).
  2. \( y_{\text{train}} = f(x_{\text{train}}, m_{\text{init}}). \)
  3. for \( i = 1 \) to \( n_{\text{train}} \) do
  4. \( y_{\text{train}} = f(x_{\text{train}}, m_{\text{init}}) \).
  5. \( m_{\text{init}} = m_{\text{init}}(x_{\text{train}}, y_{\text{train}}) \).
  6. \( m_{\text{final}} = m_{\text{final}}(x_{\text{train}}, y_{\text{train}}) \).
  7. end for

- Partitioning based
  Algorithm 2 splitILI with instILI
  \[
  \text{Input:} \quad x_{\text{train}}, y_{\text{train}}, m, \theta, \varepsilon
  \]
  \[
  \text{Output:} \quad m_{\text{final}}, y_{\text{train}}
  \]
  1. \( m_{\text{init}} = m \).
  2. \( y_{\text{train}} = f(x_{\text{train}}, m_{\text{init}}). \)
  3. for \( i = 1 \) to \( n_{\text{train}} \) do
  4. \( y_{\text{train}} = f(x_{\text{train}}, m_{\text{init}}) \).
  5. \( m_{\text{init}} = m_{\text{init}}(x_{\text{train}}, y_{\text{train}}) \).
  6. \( m_{\text{final}} = m_{\text{final}}(x_{\text{train}}, y_{\text{train}}) \).
  7. \( m_{\text{pred}} = m_{\text{pred}}(x_{\text{train}}, y_{\text{train}}) \).
  8. \( m_{\text{final}} = m_{\text{final}}(x_{\text{train}}, y_{\text{train}}) \).
  9. \( m_{\text{final}} = m_{\text{final}}(x_{\text{train}}, y_{\text{train}}) \).
  10. \( m_{\text{final}} = m_{\text{final}}(x_{\text{train}}, y_{\text{train}}) \).
  11. end for
  12. \( m_{\text{final}} = m_{\text{final}}(x_{\text{train}}, y_{\text{train}}) \).

Results

- 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]
- Filters can be applied, e.g. confidence

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

References: