DOCTOR OF PHILOSOPHY

Example weighting for deep representation learning

Wang, Xinshao

Award date:
2020

Awarding institution:
Queen's University Belfast

Link to publication

Terms of use
All those accessing thesis content in Queen's University Belfast Research Portal are subject to the following terms and conditions of use

• Copyright is subject to the Copyright, Designs and Patent Act 1988, or as modified by any successor legislation
• Copyright and moral rights for thesis content are retained by the author and/or other copyright owners
• A copy of a thesis may be downloaded for personal non-commercial research/study without the need for permission or charge
• Distribution or reproduction of thesis content in any format is not permitted without the permission of the copyright holder
• When citing this work, full bibliographic details should be supplied, including the author, title, awarding institution and date of thesis

Take down policy
A thesis can be removed from the Research Portal if there has been a breach of copyright, or a similarly robust reason. If you believe this document breaches copyright, or there is sufficient cause to take down, please contact us, citing details. Email: openaccess@qub.ac.uk

Supplementary materials
Where possible, we endeavour to provide supplementary materials to theses. This may include video, audio and other types of files. We endeavour to capture all content and upload as part of the Pure record for each thesis. Note, it may not be possible in all instances to convert analogue formats to usable digital formats for some supplementary materials. We exercise best efforts on our behalf and, in such instances, encourage the individual to consult the physical thesis for further information.
Example Weighting for Deep Representation Learning

Xinshao Wang

School of Electronics, Electrical Engineering and Computer Science
Queen’s University Belfast

This dissertation is submitted for the degree of

Doctor of Philosophy

The Institute of Electronics, Communications and Information Technology (ECIT)

August 2020
I would like to dedicate this doctoral thesis to my beloved family.
Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 80,000 words including appendices, bibliography, footnotes, tables and equations.

Xinshao Wang
August 2020
Acknowledgements

I would like to thank many people who have helped me through comments and discussions when I was writing the several papers composing this thesis. I would like to thank Cheng Cai, Piotr Chudzik, Romain Garnier, Guosheng Hu, Yang Hua, Elyor Kodirov, Cheng Long, Zheng Lu, Jesus Martinez del Rincon, Neil M. Robertson and Ivor Spence in alphabetical order, and many other reviewers for their helpful discussions. I would like to thank Cheng Cai for his encouragement. In addition, I would like to thank Guosheng Hu and Romain Garnier for their collaboration in the earlier stage of my PhD research. I further would like to thank Elyor Kodirov for his great collaboration throughout my PhD life. Additionally, I would like to thank Cheng Long, Jesus Martinez del Rincon and Ivor Spence for being my Research Student Lifecycle (RSL) panel members. Finally, I would like to thank my supervisors Neil M. Robertson and Yang Hua. They are greatly supportive and always encourage me to conduct and deliver fantastic research work. I cannot appreciate more the research flexibility and support that I was given all the time.

I would like to hugely thank Anyvision for sponsoring my PhD research with the three-year University Special Research Scholarship, which covers international tuition fees and living expenses. Additionally, I greatly thank Anyvision for providing me the computational resources used in my PhD research.

Finally, I would like to thank Tim Hospedales and Barry Devereux for being my external and internal examiners, respectively. During the viva, they provided me really valuable comments. In addition, I would like to thank Paul Miller for being the independent convenor of my viva examination.
Abstract

In gradient-based optimisation, the derivative of the loss of an example can be interpreted as the example’s effect on the update of a model. Consequently, a derivative magnitude function can be considered to provide a weighting scheme from the viewpoint of example weighting. Therefore, example weighting is universal in deep learning. Partially arising from the recent work on the risky memorisation behaviours of deep neural networks (Arpit et al., 2017; Zhang et al., 2017b), example weighting becomes an active research filed (Chang et al., 2017; Toneva et al., 2019). Example weighting has ‘hard’ and ‘soft’ versions: (1) ‘hard’ weighting is well-known as sample selection or mining, i.e., binary weighting; (2) ‘soft’ weighting means example differentiation using a continuous importance score.

In this thesis, we study how to learn more robust and discriminative representations using deep supervised learning. Technically, we propose example weighting for better optimisation and regularisation. Example weighting techniques differentiate and weight training data points according to a criteria, which varies in different scenarios. Example weighting improves the generalisation performance a lot, which is proved across multiple network architectures and learning tasks. We focus on two learning tasks in this thesis: learning to rank, and learning to classify. In both tasks, we reveal the importance of example weighting, by which a deep model focuses on more informative patterns, and pays less attention to non-informative (easy) and noisy (usually extremely hard) ones during the learning process. Therefore, example weighting is an important tool for guiding deep models to treat training samples differentially and learn meaningful patterns robustly and effectively.

Furthermore, our study on example weighting helps us understand better about the training data and a model’s learning process. When a training dataset is clean, naively assigning higher weights to harder examples works well. However, when the dataset contains both meaningful and wrong information, a model learns meaningful patterns before fitting random errors. The challenge becomes how to differentiate trusted and error patterns as training progresses, and avoid fitting the error transformation. We demonstrate that example weighting is an effective approach for addressing this challenge. Additionally, we empirically justify the effectiveness of our proposed example weighting methods in other adverse cases:
(1) in-distribution anomalies, e.g., label noise; (2) out-of-distribution anomalies, e.g., input with no object of interest; (3) sample imbalance.
Table of contents

List of figures xvii
List of tables xxi

1 Introduction 1
  1.1 What Is Deep Learning? ........................................ 1
    1.1.1 Layer .................................................. 2
    1.1.2 Optimisation with Stochastic Gradient Descent ............ 2
    1.1.3 Regularisation ......................................... 3
  1.2 Supervised Representation Learning in Deep Learning .......... 3
    1.2.1 Learning to rank ...................................... 4
    1.2.2 Learning to classify ................................... 5
  1.3 Example Weighting ........................................... 5
    1.3.1 Example weighting is universal in deep learning ....... 5
    1.3.2 Example weighting in learning to rank .................. 6
    1.3.3 Example weighting in learning to classify ............... 6
  1.4 Thesis Overview ............................................. 7

2 Example Weighting for Contrastive Loss 11
  2.1 Introduction ............................................... 11
  2.2 Related Work ............................................... 14
  2.3 Methodology ............................................... 15
    2.3.1 Online Soft Mining ................................... 16
    2.3.2 Class-Aware Attention ................................ 18
    2.3.3 Weighted Contrastive Loss ............................ 18
  2.4 Experiments ................................................. 20
    2.4.1 Fine-Grained Visual Categorisation .................... 20
    2.4.2 Video-based Person Re-identification ................. 21
# Table of contents

2.4.3 Ablation Study ............................................................. 22  
2.5 Conclusion ................................................................. 24  

3 Example Weighting for Ranked List Loss .................................. 25  
3.1 Introduction ................................................................. 26  
3.2 Preliminaries and Related Work ........................................... 29  
3.2.1 Design of Loss Functions for Learning Discriminative Deep Representations ................................................. 29  
3.2.2 Structured Losses ....................................................... 31  
3.2.3 Mining and Weighting Non-trivial Examples ....................... 34  
3.3 Methodology ................................................................. 35  
3.3.1 Pairwise Constraint .................................................... 35  
3.3.2 Ranked List Loss ........................................................ 36  
3.3.3 Hypersphere Regularisation by Distance Thresholds .......... 38  
3.3.4 RLL-Simpler ............................................................... 39  
3.3.5 Learning Deep Models with RLL ................................... 40  
3.3.6 Computational Complexity .......................................... 40  
3.3.7 Exploring The Critical Learning Periods of Deep Metric Learning ................................................................. 41  
3.4 Experiments ................................................................. 42  
3.4.1 Experimental Details .................................................. 42  
3.4.2 Comparison with Recent Baselines ................................. 45  
3.4.3 On the network depth of an embedding function ............... 47  
3.4.4 Mining Non-trivial Pairs ............................................ 48  
3.4.5 Weighting Negative Pairs ........................................... 50  
3.4.6 Weighting Positive Pairs ............................................ 50  
3.4.7 Hypersphere Regularisation ......................................... 52  
3.4.8 Ablation Study on Other General Factors ....................... 52  
3.4.9 Qualitative results ................................................... 54  
3.5 Study on the critical learning periods of deep metric learning .... 54  
3.6 Conclusion ................................................................. 55  

4 Example Weighting for Instance Cross Entropy ............................... 57  
4.1 Introduction ................................................................. 57  
4.2 Related Work ............................................................... 60  
4.2.1 Structured Losses by Query versus Class Centres .............. 60  
4.2.2 Structured Losses by Query versus Instances ................... 61  
4.2.3 Sample Mining and Weighting ..................................... 61
4.2.4 Discussion .................................................. 62
4.3 Instance Cross Entropy ...................................... 63
  4.3.1 Revisiting Categorical Cross Entropy ............... 63
  4.3.2 Instance Cross Entropy .................................. 63
  4.3.3 Regularisation by $L_2$ Feature Normalisation ..... 65
  4.3.4 Sample Reweighting of ICE ......................... 65
  4.3.5 Complexity Analysis ................................... 67
4.4 A Case Study and Intuitive Explanation of ICE .......... 68
4.5 Experiments .................................................. 69
  4.5.1 Implementation Details and Evaluation Settings .... 69
  4.5.2 Quantitative Results ................................... 70
4.6 More Ablation Studies ..................................... 72
  4.6.1 Analysis of Sample Reweighting in ICE ............ 72
  4.6.2 Batch Content ........................................ 73
  4.6.3 Batch Size ............................................. 74
  4.6.4 Embedding Size ....................................... 74
4.7 t-SNE Visualisation ......................................... 75
4.8 Conclusion .................................................. 75

5 IMAE: Example Weighting for Learning to Classify 79
  5.1 Introduction ............................................... 80
  5.2 Preliminaries .............................................. 81
  5.3 Gradient Magnitude Serving as Weight ................ 83
    5.3.1 Derivative of Softmax, CCE and MAE Layers .... 83
    5.3.2 Perspective of Derivatives w.r.t. Logits Other Than Probabilities ... 84
  5.4 Improved MAE ............................................ 85
    5.4.1 Design Motivation: To Adjust Gradient Magnitude’s Variance and Impact Ratio ... 86
    5.4.2 Discussion of MAE and CCE ......................... 87
  5.5 Experiments ............................................... 87
    5.5.1 Image Classification on CIFAR-100 with Synthetic Noise .... 88
    5.5.2 Image Classification on Clothing1M with Realistic Unknown Noise 90
    5.5.3 Empirical Analysis of IMAE Against Basic Baselines CCE and MAE on CIFAR-10 ... 92
  5.6 Related Work .............................................. 94
    5.6.1 Example Weighting ................................. 94
    5.6.2 Noise-Robust Theorems on Loss Functions ....... 94
## Table of contents

5.7 Conclusion ................................................. 94

6 DM: Example Weighting for Learning to Classify .......... 97
   6.1 Introduction ........................................ 98
      6.1.1 Why do we design a derivative magnitude function instead of a loss function? ................................................. 98
      6.1.2 Using an emphasis density function to mathematically express example weighting ................................................. 99
   6.2 Related Work ........................................ 100
   6.3 Derivative Manipulation ................................ 100
      6.3.1 The derivative direction and magnitude of common losses .... 101
      6.3.2 Defining an example weighting scheme by an emphasis density function ................................................. 102
   6.4 Experiments .......................................... 104
      6.4.1 A premise for setting the emphasis mode when noisy labels exist . 104
      6.4.2 Robust image classification .................................. 105
      6.4.3 Robust video retrieval .................................... 109
      6.4.4 Sentiment analysis of movie reviews .......................... 110
      6.4.5 Further analysis ......................................... 110
   6.5 Conclusion ............................................. 110

7 Conclusion and Future Research ............................... 111
   7.1 Example weighting for other vital applications .......... 112
   7.2 To design new example weighting schemes ................. 113

References ................................................................ 115

Appendix A Supplementary Material for IMAE ................. 127
   A.1 The Impact of $T$ on Gradient Magnitude’s Variance ....... 127
   A.2 The Impact of $T$ on Validation Accuracy ................. 128
      A.2.1 CIFAR-10 with intact training labels ..................... 128
      A.2.2 CIFAR-10 with corrupted training labels ................. 128
   A.3 The Impact of $T$ on Training Accuracy ................... 129
      A.3.1 Fitting of the intact training set .......................... 129
      A.3.2 Fitting of the corrupted training set ....................... 129
   A.4 Choosing $T$ in Practice .................................... 130
   A.5 Video person re-identification ............................. 130
## A.6 The Results of IMAE Using Different Stochastic Optimisers

### Appendix B Supplementary Material for Derivative Manipulation

#### B.1 Derivation Details of Softmax, CCE, MAE and GCE

- **B.1.1 Derivation of Softmax Normalisation**
- **B.1.2 Derivation of CCE**
- **B.1.3 Derivation of MAE**
- **B.1.4 Derivation of GCE**
- **B.1.5 Derivatives w.r.t. Logits $z_i$**

#### B.2 Beating Standard Regularisers Under Label Noise

#### B.3 Small-scale Fine-grained Visual Categorisation of Vehicles

#### B.4 Experimental Details of Robust Sentiment Analysis of Movie Reviews

- **B.4.1 Label Noise**
- **B.4.2 Sample Imbalance**
- **B.4.3 Net Architecture and Optimisation Solver**

#### B.5 The Effectiveness of Label Correction

#### B.6 More Detailed Empirical Results

- **B.6.1 Training DNNs under label noise**
- **B.6.2 Empirical Analysis of DM on CIFAR-10**
- **B.6.3 Learning Dynamics on Clean Training Data**
List of figures

1.1 The optimisation objective of learning to rank. .................................... 4
1.2 The pipeline of learning to classify. ....................................................... 5
1.3 Display of abnormal training examples. ............................................... 7

2.1 The illustration of Online Soft Mining (OSM) and Class-Aware Attention (CAA) for pair mining. ................................................................. 13
2.2 The comparison of our proposed weighted contrastive loss with traditional contrastive loss. ................................................................. 17

3.1 Illustration of our proposed Ranked List Loss (RLL). Given a query and its ranked list, RLL aims to make the query closer to the positive set than to the negative set by a margin $m$. Circle and triangle represent two different classes. The blue circle is a query. The yellow shapes represent nontrivial examples while the red shapes represent trivial examples. The arrow indicates the query’s gradient direction determined by the corresponding non-trivial examples. The final gradient direction of the query is a weighted combination of them. The optimisation target for every list is shown in the bottom. *Best viewed in colour.* ......................................................... 27
3.2 Illustration of different ranking-motivated structured losses. Different shapes (circle, triangle and square) represent different classes. For simplicity, only 3 classes are shown. The purple circle is an anchor (query). In triplet (Schroff et al., 2015), the anchor is compared with only one negative example and one positive example. In $N$-pair-mc (Sohn, 2016), Proxy-NCA (Movshovitz-Attias et al., 2017) and Lifted Struct (Oh Song et al., 2016), one positive example and multiple negative classes are incorporated. $N$-pair-mc randomly selects one example per negative class. Proxy NCA pushes the anchor away from negative proxies instead of negative examples. The proxy is class-level and can represent any instance in the corresponding class. Lifted Struct uses all examples from all negative classes. On the contrary, our proposed Ranked List Loss not only exploits all negative examples, but also makes use of all positive ones. *Best viewed in colour.*

3.3 The optimisation objectives with hypersphere regularisation. Different shapes represent different classes. For simplicity, we only show three classes while many classes exist in practice. $\alpha - m$ denotes the diameter of each class hypersphere. Therefore, the distance between any two positive examples is optimised to be no greater than $\alpha - m$. In addition, the distance between any two hypersphere boundaries is no less than $m$.

3.4 The overall framework of our proposed ranked list loss. In one mini-batch, every image acts as a query iteratively and ranks other images according to the similarity scores. Then in every ranked list, we mine non-trivial data points and weight them based on their distances to the query. Finally, the ranked list loss is computed for every query.

3.5 Visualisation of image retrieval on the SOP test dataset. The leftmost column shows queries, which rank the images in the gallery according to the similarity.

4.1 Our ICE and related losses.

4.2 $t$-SNE visualisation on the SOP test set.

4.3 $t$-SNE visualisation on the CUB-200-2011 test set.

4.4 $t$-SNE visualisation on the CARS196 test set.

5.1 Sample’s weight along with sample’s probability being classified to its labelled class in CCE, MAE, IMAE with $T = 8$.

5.2 Pipeline of a softmax deep network.

5.3 The loss expression of IMAE.
5.4 CIFAR-10 with noise rate $r = 40\%$. ........................................ 92

6.1 Illustration of DM in terms of optimisation and example weighting. .... 99

6.2 We train ResNet-56 on CIFAR-10. In (a) and (b), we observe noisy examples have much less $p_i$ than clean ones, thus being more difficult examples in both CCE and DM ($\lambda = 1$). In (c), for each label noise rate, we show the optimised $\psi_{DM}$ from $\{0, 1/3, 1/2, 2/3\}$, i.e., $\lambda \in \{0, 1/2, 1\}$. ..................... 105

A.1 Sample’s weight along with probability in IMAE with different $T$ (IMAE-$T$). 128
A.2 The accuracy on the CIFAR-10 test set along with training iterations. .... 132
A.3 The accuracy on the CIFAR-10 training sets along with training iterations. 133
A.4 IMAE-16’s, IMAE-8’s and IMAE-4’s accuracies on the clean training set when they are trained on the corrupted training set. ................. 134
A.5 CIFAR-10 with a noise rate of $r = 80\%$. The accuracies on the testing set, the noisy subset and clean subset of the training set along with training iterations. 135

B.1 A designed neural network with one 8-neuron hidden layer for sentiment analysis on IMDB. It is quite simple and our objective is not to represent the state-of-the-art. .......................................................... 144
B.2 The learning dynamics of ResNet-56 on CIFAR-10, i.e., training and testing accuracies along with training iterations. The legend in the top left is shared by all subfigures. ‘xxx: yyy’ means ‘method: emphasis mode’. We have two key observations: 1) When noise rate increases, better generalisation is obtained with higher emphasis mode, i.e., focusing on relatively easier examples; 2) Both overfitting and underfitting lead to bad generalisation. For example, ‘CCE: 0’ fits training data much better than the others while ‘DM: None’ generally fits it unstably or a lot worse. Better viewed in colour. .... 147
B.3 ResNet-56 on CIFAR-10 ($r = 20\%$). From left to right, the results of four emphasis modes $0, \frac{1}{3}, 0.5, \frac{2}{3}$ with different emphasis variances are displayed in each column respectively. When $\lambda$ is larger, $\beta$ should be larger. Specifically: 1) when $\lambda = 0$: we tried $\beta = 0.5, 1, 2, 4$; 2) when $\lambda = 0.5$: we tried $\beta = 4, 8, 12, 16$; 3) when $\lambda = 1$: we tried $\beta = 8, 12, 16, 20$; 4) when $\lambda = 2$: we tried $\beta = 12, 16, 20, 24$. ................................. 148
B.4 ResNet-56 on CIFAR-10 ($r = 40\%$). From left to right, the results of four emphasis modes $0, \frac{1}{3}, 0.5, \frac{2}{3}$ with different emphasis variances are displayed in each column respectively. When $\lambda$ is larger, $\beta$ should be larger. Specifically: 1) when $\lambda = 0$: we tried $\beta = 0.5, 1, 2, 4$; 2) when $\lambda = 0.5$: we tried $\beta = 4, 8, 12, 16$; 3) when $\lambda = 1$: we tried $\beta = 8, 12, 16, 20$; 4) when $\lambda = 2$: we tried $\beta = 12, 16, 20, 24$.  

B.5 ResNet-56 on CIFAR-10 ($r = 60\%$). From left to right, the results of four emphasis modes $0, \frac{1}{3}, 0.5, \frac{2}{3}$ with different emphasis variances are displayed in each column respectively. When $\lambda$ is larger, $\beta$ should be larger. Specifically: 1) when $\lambda = 0$: we tried $\beta = 0.5, 1, 2, 4$; 2) when $\lambda = 0.5$: we tried $\beta = 4, 8, 12, 16$; 3) when $\lambda = 1$: we tried $\beta = 8, 12, 16, 20$; 4) when $\lambda = 2$: we tried $\beta = 12, 16, 20, 24$.  

B.6 ResNet-56 on CIFAR-10 ($r = 80\%$). From left to right, the results of four emphasis modes $0, \frac{1}{3}, 0.5, \frac{2}{3}$ with different emphasis variances are displayed in each column respectively. When $\lambda$ is larger, $\beta$ should be larger. Specifically: 1) when $\lambda = 0$: we tried $\beta = 0.5, 1, 2, 4$; 2) when $\lambda = 0.5$: we tried $\beta = 4, 8, 12, 16$; 3) when $\lambda = 1$: we tried $\beta = 8, 12, 16, 20$; 4) when $\lambda = 2$: we tried $\beta = 12, 16, 20, 24$.  

B.7 The training and test accuracies on clean CIFAR-10 along with training iterations. The training labels are clean. We fix $\lambda = 0$ to focus on harder examples while changing emphasis variance controller $\beta$. The backbone is ResNet-20. The results of ResNet-56 are shown in Figure B.8.  

B.8 The training and test accuracies on clean CIFAR-10 along with training iterations. The training labels are clean. We fix $\lambda = 0$ to focus on more difficult examples while changing emphasis variance controller $\beta$. The backbone is ResNet-56. The results of ResNet-20 are shown in Figure B.7.
List of tables

2.1 Comparison with state-of-the-art methods on CARS196 and CUB-200-2011 in terms of Recall@K (%). .......................... 21
2.2 Video-based person re-identification datasets: MARS and LPW. .............. 22
2.3 Comparison with state-of-the-art methods on MARS in terms of CMC(%) and mAP(%). .................................................. 23
2.4 Comparison with state-of-the-art methods on LPW in terms of CMC (%). . 23
2.5 Ablation studies on CUB-200-2011, LPW, MARS in terms of Recall@K (%) and mAP (%). .................................................. 23

3.1 Comparison with the state-of-the-art methods on SOP. The ‘–’ denotes the corresponding results are not reported in the original paper. For a fair comparison, all reported results have an embedding size of 64, which is the common setting and more challenging than a larger embedding size. .... 46
3.2 Comparison with the state-of-the-art methods on In-shop Clothes dataset. For a fair comparison, we compare with the reported results with an embedding size of 128, which is the common setting on In-shop Clothes. ............... 46
3.3 Comparison with the state-of-the-art methods on CARS196 in terms of Recall@K (%). For a comparison, all the reported results have an embedding size of 64. .................................................. 47
3.4 Comparison with the state-of-the-art methods on CUB-200-2011 in terms of Recall@K (%). All the displayed results have an embedding size of 64 for a fair comparison. .................................................. 48
3.5 Exploration on the network depth of an embedding function and the concatenation of embedding functions on SOP. Additionally, the bottom block shows the recently reported results using a higher embedding size, except for the results in Table 3.1. .................................................. 49
### List of tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.6</td>
<td>Exploration on the network depth of an embedding function and the concatenation of embedding functions on In-shop Clothes dataset. In addition, the recently reported results using a higher embedding size are displayed in the bottom block, except for the results in Table 3.2.</td>
<td>49</td>
</tr>
<tr>
<td>3.7</td>
<td>The impact of $\alpha$ on the SOP. In all experiments, $m = 0.4, T_n = 10$.</td>
<td>50</td>
</tr>
<tr>
<td>3.8</td>
<td>Results of different $T_n$ on the SOP in terms of Recall@K (%) We fix $m = 0.4, \alpha = 1.2$ in all experiments.</td>
<td>50</td>
</tr>
<tr>
<td>3.9</td>
<td>Results of weighting positive examples on the SOP. We fix other parameters, i.e., $m = 0.4, \alpha = 1.2$ and $T_n = 10$.</td>
<td>51</td>
</tr>
<tr>
<td>3.10</td>
<td>The impact of hypersphere diameter for preserving the intra-class variance on the SOP. We set $\alpha = 1.2, T_n = 10$ in all experiments.</td>
<td>52</td>
</tr>
<tr>
<td>3.11</td>
<td>The results of different batch size on the SOP.</td>
<td>53</td>
</tr>
<tr>
<td>3.12</td>
<td>The results of RLL-(L,M,H) with different embedding size on the SOP test dataset.</td>
<td>53</td>
</tr>
<tr>
<td>3.13</td>
<td>The results of different batch content $C \times K$ on the SOP. We fix $N = C \times K = 180$ and change $C, K$.</td>
<td>54</td>
</tr>
<tr>
<td>3.14</td>
<td>Study on the dynamic weighting scheme of negative pairs on the In-shop Clothes dataset. The batch size is 60 and the number of training iterations is 30,000.</td>
<td>56</td>
</tr>
<tr>
<td>4.1</td>
<td>A summary of three fine-grained datasets.</td>
<td>70</td>
</tr>
<tr>
<td>4.2</td>
<td>Comparison with the state-of-the-art methods on CARS196, CUB-200-2011 and SOP in terms of Recall@K (%).</td>
<td>71</td>
</tr>
<tr>
<td>4.3</td>
<td>The results of different reweighting parameters $s$ on SOP in terms of Recall@K (%).</td>
<td>73</td>
</tr>
<tr>
<td>4.4</td>
<td>The impact of batch content $C \times k$ on SOP in terms of Recall@K (%).</td>
<td>73</td>
</tr>
<tr>
<td>4.5</td>
<td>The results of different batch size $N$ on SOP in terms of Recall@K (%).</td>
<td>74</td>
</tr>
<tr>
<td>4.6</td>
<td>The results of different embedding size on SOP in terms of Recall@K (%).</td>
<td>74</td>
</tr>
<tr>
<td>5.1</td>
<td>Classification accuracy (%) of CCE, MAE, and IMAE on CIFAR-10. 40% of training examples, i.e., the noisy subset, have wrong labels.</td>
<td>81</td>
</tr>
<tr>
<td>5.2</td>
<td>Summary of CCE, MAE and IMAE.</td>
<td>85</td>
</tr>
<tr>
<td>5.3</td>
<td>Test accuracy (%) on CIFAR-100 using ResNet44 under symmetric noisy labels.</td>
<td>89</td>
</tr>
<tr>
<td>5.4</td>
<td>Test accuracy (%) on CIFAR-100 using ResNet44 under asymmetric noisy labels.</td>
<td>90</td>
</tr>
<tr>
<td>5.5</td>
<td>Classification accuracy (%) on Clothing1M with ResNet50.</td>
<td>90</td>
</tr>
</tbody>
</table>
5.6 Results (%) of CCE, MAE and IMAE on CIFAR-10 with different noise rates. 91

6.1 Test accuracy (%) on CIFAR-100. The best results on each block are bolded. Italic row is the most basic baseline where examples have identical weights. All methods use ResNet-44 (He et al., 2016). .......................... 106

6.2 Accuracy (%) of DM and other baselines on CIFAR-10 under symmetric label noise. The best results on each block are bolded. Number format of this table follows MentorNet. ‘–’ denotes result was not reported. ....... 108

6.3 DM versus other losses on sentiment classification of movie reviews. The results of common losses after DN are in the brackets. We test on two adverse cases: label noise and sample imbalance. P-N Ratio denotes the ratio of positive reviews to negative ones. ................................. 108

6.4 Results of common stochastic optimisers. Adam (Kingma & Ba, 2015) is an adaptive gradient method. We report three settings of it. ......................... 108

6.5 Accuracy (%) on Clothing1M. The leftmost block’s results are from SL (Wang et al., 2019i) while the middle block’s are from Masking (Han et al., 2018a). Results of common losses after DN are in the brackets. .............. 109

6.6 Video retrieval results on MARS. All other methods use GoogLeNet V2 except that DRSA and CAE use more complex ResNet-50. ......................... 110

A.1 The weight variance (gradient magnitude’s variance) of IMAE when $T$ changes. ................................................................. 127

A.2 The retrieval results of CCE, MAE, GCE and IMAE on MARS with GoogLeNet V2. ......................................................... 131

A.3 The results of algorithms using different stochastic optimisers on CIFAR-10 with 40% class-independent (symmetric) label noise. .......................... 131

B.1 Results of DM and other standard regularisers on CIFAR-100. We set $r = 40\%$, i.e., the label noise is severe but not belongs to the majority. We train ResNet-44. We report the average test accuracy and standard deviation (%) over 5 trials. Baseline is CCE without regularisation. ................. 140

B.2 The test accuracy (%) of DM and other standard regularisers on Vehicles-10. We train ResNet-44. Baseline denotes CCE without regularisation. We test two cases: symmetric label noise rate is $r = 40\%$, and clean data $r = 0$. 141
B.3 Is it feasible to correct the labels of noisy training data? Our results demonstrate the effectiveness of label correction using DM. When retraining from scratch on the relabelled training data, we do not adjust the hyper-parameters $\beta$ and $\lambda$. Therefore, the reported results of retraining on relabelled datasets are not the optimal. In label correction, the original labels are replaced by their corresponding predictions.  

B.4 Results of CCE, DM on CIFAR-10 under noisy labels. For every model, we show its best test accuracy during training and the final test accuracy when training terminates, which are indicated by ‘Best’ and ‘Final’, respectively. We also present the results on corrupted training sets and original intact one. The overlap rate between corrupted and intact sets is $(1 - r)$. When $\lambda$ is larger, $\beta$ should be larger for adjusting emphasis variance. The intact training set serves as an indicator of meaningful fitting and we observe that its accuracy is always consistent with the final test accuracy.
Chapter 1

Introduction

1.1 What Is Deep Learning?

Yann Lecun gave a general definition of deep learning in his talk at AAAI 2020.

**Definition** (Deep Learning). *Deep Learning is building a system by assembling parameterized modules into a (possibly dynamic) computation graph, and training it to perform a task by optimizing the parameters using a gradient-based method.*

Deep learning is also well-known as deep neural networks (DNNs) (LeCun et al., 2015). In general, one deep neural network is composed of multiple layers, leading to a hierarchical architecture. The output of one layer is the input of its subsequent layer (plain networks with only bottom-up connection) (Krizhevsky et al., 2012), or the input of its next $k$-th layer (residual networks with skip connections, i.e., shortcuts to jump over some layers) (He et al., 2016), or the input of all subsequent layers (dense networks) (Huang et al., 2017). Therefore, we can denote $i$-th neural layer as $h_i = f^{\omega_i}(\sum_{k=1}^{K} c_k h_{i-k})$ with a learnable parameter set $\omega_i$. Here, $h_{i-k}, i-k > 0$ represents the output of a preceding layer, while $c_k \in \{0, 1\}$ denotes whether there is a connection or not.

Mathematically, a deep neural network is the composition of functions: $f^{\omega_H} \circ f^{\omega_{H-1}} \circ \ldots \circ f^{\omega_1} : X \rightarrow Y$, where $X$ is the input data and $Y$ is the transformed data by this composite function, $H$ is the total number of parametrised functions in the chaining process. Between parametrised functions, there may exist non-parametrised mappings, e.g., activation functions. One function is termed one neural layer for historical reasons (Rumelhart et al., 1986). In addition to the layer, we briefly review the optimisation and regularisation of a deep neural network.
1.1.1 Layer

We briefly review some most widely used layers: fully connected layer, convolutional layer, activation layer, and loss layer. A fully connected layer is a linear transformation of its input feature $x \in \mathbb{R}^D$: $xW + b$, where $W \in \mathbb{R}^{D \times K}, b \in \mathbb{R}^K$. $W$ is termed weight matrix, while $b$ is named bias or translation. Compared with the fully connected layer, a convolutional layer is a partially (locally) connected layer, which is very popular for visual recognition. The dimensionality of a convolutional weight filter is much smaller than that of the input feature. In addition, a weight filter is shared by all local regions of the input feature. Consequently, the convolutional layer is more parameter-efficient, which enables us to stack multiple convolutional layers in one model. Both fully connected layers and convolutional layers are linear transformations of the inputs. An activation layer applies an element-wise non-linearity to the linear transformation output. Common activation layers are the rectified linear unit (ReLU) and hyperbolic tangent (TanH). A loss layer is generally an empirical loss metric, which calculates the distance between a network’s prediction and the ground-truth. Given multiple examples, the loss layer outputs a scalar, which is the average empirical loss of all examples.

In summary, according to the definition of deep learning, a computation graph is defined by layers and their connections. We define a task using a loss layer, which is usually at the end of a computation branch. All hidden layers and the final output layer are feature representations of the inputs in different transformed spaces.

1.1.2 Optimisation with Stochastic Gradient Descent

To optimise the parameters of a deep neural network such that it performs the task well, we apply gradient descent (Rumelhart et al., 1986). To be more exact, we apply stochastic gradient descent (SGD) on mini-batches, mainly due to the computational limitation given a large amount of training data.

In SGD, we first compute the derivatives of examples in a mini-batch in the loss layer. Then the derivatives of all examples are accumulated, serving as a gradient estimator of the whole training set. According to the chain rule of differentiating a composite function, we back-propagate the gradient in the loss layer to all preceding layers, which is well known as error back-propagation (Rumelhart et al., 1986). Accordingly, for $i$-th layer, we can optimise its learning parameters using gradient descent: $\omega_i^{new} = \omega_i^{old} - \eta_i \nabla L(\omega_i^{old})$, where $\eta_i$ denotes a local learning rate, which may be different for different layers. $L$ denotes a loss function.
1.1.3 Regularisation

A deep neural network has a strong data fitting ability because of its deep hierarchical architecture and a large number of learning parameters (Arpit et al., 2017; Wang et al., 2019f,h; Zhang et al., 2017b). Therefore, how to regularise deep neural networks is an active research topic. There are some common regularisation techniques: (1) weight $L_2$ regularisation a.k.a. weight decay (Golatkar et al., 2019; Hanson & Pratt, 1989; Krogh & Hertz, 1992; Loshchilov & Hutter, 2019; Van Laarhoven, 2017); (2) weight normalisation (Salimans & Kingma, 2016; Van Laarhoven, 2017); (3) feature normalisation (Ba et al., 2016; Ioffe & Szegedy, 2015; Wang et al., 2017a, 2019e; Xu et al., 2019), e.g., batch normalisation is one of the most widely used (Ioffe & Szegedy, 2015).

In addition, stochastic regularisation techniques are also popular. Stochastic regularisation techniques inject stochastic noise into the model for the purpose of regularisation (Hinton et al., 2012; Huang et al., 2016c; Krueger et al., 2016; Moon et al., 2015; Singh et al., 2016; Srivastava et al., 2014; Wan et al., 2013). Among them, dropout (Srivastava et al., 2014) is the most popular one.

Recently, learning soft targets and output regularisation have become popular. In classification problems, label smoothing softens the learning targets, which are one-hot representations (hard targets). For example, we can inject uniform label noise to a one-hot label representation (Szegedy et al., 2016). We also change the training of a neural network by matching its predictions with those of another network, which is famous as knowledge distillation (Hinton et al., 2015; Müller et al., 2019). In addition to label smoothing, we can also use a confidence penalty to learn smoother predictions (Pereyra et al., 2017).

1.2 Supervised Representation Learning in Deep Learning

In this thesis, we study how to learn more robust and discriminative representations using deep supervised learning. Technically, we propose example weighting for better optimisation and regularisation of deep neural networks. Example weighting techniques differentiate and weight training data points according to a criteria, which needs to be custom-designed for a specific scenario. We focus on two supervised learning tasks in this thesis: learning to rank and learning to classify. In both tasks, we reveal the importance of example weighting, by which a deep model focuses on more informative patterns, and pays less attention to non-informative (easy) and noisy (usually extremely hard) ones during the learning process.

In this subsection, we briefly introduce these two tasks. We will represent example weighting in Section 1.3.
1.2.1 Learning to rank

In deep learning, one of the most common discriminative models for learning representations is learning to rank (Chopra et al., 2005; Hadsell et al., 2006), which is well-known as distance metric learning (Goldberger et al., 2005; Oh Song et al., 2016; Schroff et al., 2015; Wang et al., 2019b). Basically, it targets at learning a distance metric (Weinberger et al., 2006), by which we can measure the similarity between two examples. There are: (1) Pairwise similarity relationship learning, e.g., contrastive loss (Chopra et al., 2005; Hadsell et al., 2006; Wang et al., 2019c); (2) High-order similarity relationship modelling, which defines the supervision information using multiple data points. It includes triplet loss (Hermans et al., 2017; Schroff et al., 2015), quadruplet loss (Huang et al., 2016b), and ranked list loss (Wang et al., 2019b), etc.

![Figure 1.1: The optimisation objective of learning to rank.](image)

The overall optimisation objective of learning to rank is illustrated in Figure 1.1. As we emphasise on example weighting techniques in this thesis, for clarity, we present the definition of training examples in learning to rank as follows (Regardless of pairwise or high-order similarity relationship modelling, doublets are the most basic elements):

**Definition** (Training Examples in learning to rank). *One basic training example is defined to be a pair of observations with a binary label, where its label denotes whether two data points are similar or not.*
1.2.2 Learning to classify

Learning to classify is another common discriminative model for encoding the input. Given an input, the objective is to predict its probability distribution of being different training classes. The pipeline of learning to classify is illustrated in Figure 1.2. Analogously to learning to rank, we present the definition of training examples in learning to classify for the clarity of introducing example weighting later.

**Definition** (Training Examples in learning to classify). A training example is defined to be an observation-label pair, where the label defines its semantic information, i.e., which class the input observation belongs to.

![Fig. 1.2: The pipeline of learning to classify.](image)

1.3 Example Weighting

1.3.1 Example weighting is universal in deep learning

First of all, for the purpose of clarity, we present our definition of example weighting in deep learning:

**Definition** (Example Weighting). In gradient-based optimisation, the loss’s derivative of an example can be interpreted as its effect on the update of a model (Barron, 2019; Hampel et al., 1986). Therefore, a derivative magnitude function defines a weighting scheme from the viewpoint of example weighting.

By this definition, the derivative of the loss of an example determines how much impact the example has on the update of a model. A loss function’s derivative magnitude function provides a weighting scheme. Therefore, example weighting exists everywhere in deep learning. Accordingly, one technique that leads to the change of a derivative magnitude function, is equivalent to, modifying an example weighting strategy.
1.3.2 Example weighting in learning to rank

Learning on non-trivial examples

Compared with the classification task, deep metric learning deals with much more training samples since all possible combinations of data points need to be considered, e.g., quadratically more examples in contrastive loss and cubically more examples in triplet loss. Consequently, this results in slow convergence and poor local optima due to a large fraction of trivial examples, i.e., non-informative ones whose loss values are zero. To alleviate this, sample mining and example weighting methods are commonly used to select or emphasise on non-trivial examples for accelerating convergence and improving performance (Cui et al., 2016; Huang et al., 2016b; Oh Song et al., 2016; Schroff et al., 2015; Shi et al., 2016; Simo-Serra et al., 2015; Wang & Gupta, 2015; Wang et al., 2019a,b,c,e,g; Wu et al., 2017; Yuan et al., 2017).

Reducing the influence of anomalies

The presence of noise in the training set hurts a model’s performance significantly as shown in (Zhang et al., 2017a). Generally, examples with larger loss values contribute more to the update of a model. When sample mining is applied, outliers are usually mined in the subset for training. In this case, we also find that example weighting is helpful to address abnormal training examples (Wang et al., 2019c).

1.3.3 Example weighting in learning to classify

Example weighting is helpful for robust training especially when adverse conditions exist, e.g., abnormal training data and sample imbalance. We focus on applications where there are abnormal training examples, and define them as follows:

**Definition** (Abnormal Training Examples in learning to classify). We define a training example to be abnormal whenever its observation and label are semantically unmatched, e.g., in-distribution examples with incorrect labels, and out-of-distribution ones whose inputs contain only background or objects that do not belong to any training class.

We display some abnormal examples in Figure 1.3. It is non-affordable to guarantee the quality of training data as its scale grows dramatically. Consequently, abnormal examples generally exist in large-scale real-world scenarios (Berrada et al., 2018; Goldberger & Ben-Reuven, 2017; Reed et al., 2015), which is caused by many factors, such as incomplete...
1.4 Thesis Overview

Fig. 1.3: Display of abnormal training examples highlighted by red boxes. The 1st row shows synthetic abnormal examples from corrupted CIFAR-10 (Krizhevsky, 2009). The 2nd and 3rd rows present realistic abnormal examples from video person re-identification benchmark MARS (Zheng et al., 2016). We remark: 1) The abnormal images with no person in 3rd row contain no semantic information at all; 2) The last abnormal images in the 2nd and 3rd rows contain a person that does not belong to any person in the training set; 3) We cannot decide the object of interest without any prior when an image contains more than one object, e.g., the 2nd and 3rd last images in the 2nd row contain two persons.

annotation, incorrect labelling, subjectiveness, bias and so forth. Unfortunately, DNNs trained with categorical cross entropy (CCE) can fit random patterns (Zhang et al., 2017b).

1.4 Thesis Overview

In this PhD thesis, we mainly work on discriminative representation learning by learning to rank and learning to classify. We propose new example weighting algorithms and demonstrate their effectiveness in learning to rank and learning to classify. In deep learning, learning to rank is also known as deep (distance) metric learning, which is greatly popular in face verification, person re-identification, and many other applications (Liu, 2009; Oh Song et al., 2016; Schroff et al., 2015; Schütze et al., 2008; Triantafillou et al., 2017; Wang et al., 2019c). Therefore, learning to rank and deep metric learning (DML) are used interchangeably throughout this thesis.

This thesis consists of five papers, which includes one AAAI 2019 oral paper, one CVPR 2019 poster paper, and another three preprints under review now. In Chapter 1, we briefly review what is deep learning and how we learn discriminative representations using deep learning. More importantly, we present our definition of example weighting, and discuss why example weighting is universal and crucial in deep learning.
We introduce our proposed example weighting algorithms for deep metric learning in Chapters 2, 3 and 4 (Wang et al., 2019b,c,h), which are an AAAI 2019 oral paper, a CVPR 2019 poster paper, and one preprint under review, respectively. Although designed in different formats, they share the same underlying principle: (1) Learning on non-trivial examples; (2) Alleviating the misleading influence of outliers.

In Chapters 5 and 6, both preprints highlight and demonstrate that example weighting exists universally because different examples have different gradient magnitude. They are sequential work. In Chapter 5, we reveal that Mean Absolute Error Does Not Treat Examples Equally, which corrects an already published statement “mean absolute error treats examples equally” in (Zhang & Sabuncu, 2018). In Chapter 6, we deliver that: (1) In gradient-based optimisation, manipulating the derivative function is more straightforward than designing loss functions, and it has a direct impact on the update of a model; (2) A loss function’s derivative magnitude function can be interpreted as a weighting scheme. The derivative magnitude of an example defines how much impact it has on the update of a model. Consequently, we propose to formulate example weighting schemes using emphasis density functions.

In Chapter 7, we make a conclusion and summarise the open leads of future research on example weighting, e.g., other diverse and vital applications, and how to design new example weighting methods under new scenarios.

For the flexibility of reading, we maximise the independence among chapters. We list the papers which compose these chapters as follows:


Additionally, we list the other work which has been done, but not discussed in this thesis.


Chapter 2

Example Weighting for Contrastive Loss

Deep metric learning aims to learn a deep embedding that can capture the semantic similarity of data points. Given the availability of massive training samples, deep metric learning is known to suffer from slow convergence due to a large fraction of trivial samples. Therefore, most existing methods generally resort to sample mining strategies for selecting nontrivial samples to accelerate convergence and improve performance. In this chapter, we identify two critical limitations of the sample mining methods, and provide solutions for both of them. First, many previous mining methods assign one binary score to each sample, i.e., dropping or keeping it, so they only select a subset of relevant samples in a mini-batch. Therefore, we propose a novel sample mining method, called Online Soft Mining (OSM), which assigns one continuous score to each sample to make use of all samples in the mini-batch. OSM learns extended manifolds that preserve useful intraclass variances by focusing on more similar positives. Second, the existing methods are easily influenced by outliers as they are generally included in the mined subset. To address this, we introduce Class-Aware Attention (CAA) that assigns little attention to abnormal data samples. Furthermore, by combining OSM and CAA, we propose a novel weighted contrastive loss to learn discriminative embeddings. Extensive experiments on two fine-grained visual categorisation datasets and two video-based person re-identification benchmarks show that our method significantly outperforms the state-of-the-art.

2.1 Introduction

With the success of deep learning, deep metric learning has attracted a great deal of attention and been applied to a wide range of visual tasks such as image retrieval (Huang et al., 2016a; Oh Song et al., 2016; Wang et al., 2014), face recognition and verification (Schroff et al., 2015; Sun et al., 2014; Taigman et al., 2014), and person re-identification (McLaughlin et al.,
The goal of deep metric learning is to learn a feature embedding/representation that captures the semantic similarity of data points in the embedding space such that samples belonging to the same classes are located closer, while samples belonging to different classes are far away. Various deep metric learning methods have been proposed: doublet-based methods with contrastive loss (Hadsell et al., 2006; Oh Song et al., 2016; Shi et al., 2016; Yuan et al., 2017), triplet-based methods with triple loss (Chechik et al., 2010; Cui et al., 2016; Hoffer & Ailon, 2015; Schroff et al., 2015; Wang et al., 2014), and quadruplet-based methods with double-header hinge loss (Huang et al., 2016b). All of them construct pairs (samples) by using several images and introduce relative constraints among these pairs.

Compared with a classification task, deep metric learning methods deal with much more training samples since all possible combinations of images need to be considered, e.g., quadratically larger samples for contrastive loss and cubically larger samples for triplet loss. Consequently, this yields too slow convergence and poor local optima due to a large fraction of trivial samples, e.g., non-informative samples that can be easily verified and contribute zero to the loss function. To alleviate this, sample mining methods are commonly used to select nontrivial images to accelerate convergence and improve performance. Therefore, a variety of sample mining strategies have been studied recently (Cui et al., 2016; Huang et al., 2016b; Oh Song et al., 2016; Schroff et al., 2015; Shi et al., 2016; Simo-Serra et al., 2015; Wang & Gupta, 2015; Yuan et al., 2017).

Despite the popularity of exploiting sample mining in deep metric learning, there are two problems in the existing methods:

(1) **Not making full use of all samples in the mini-batch** – the first problem concerns the use of samples in the mini-batch. When most methods perform sample mining, they select samples out of mini-batch, e.g., hard\(^1\) difficult negatives mining (Oh Song et al., 2016). Specifically, they assign one binary score (0 or 1) to each sample in order to ‘drop it’ or ‘keep it’. In this way, some samples which may be useful to some extent are removed and thus only a subset of samples is mined. Therefore they cannot make full use of information in each mini-batch.

(2) **Outliers** – the second problem arises from outliers. The existing mining methods do not consider outliers. However, the presence of noise in the training set hurts the model’s performance significantly as shown in (Zhang et al., 2017a). In addition, difficult negatives mining prioritizes samples with larger training loss. As a result, outliers are generally mined in the subset for training as their training losses are more likely to be large.

\(^1\)In this chapter, ‘hard’ means ‘not soft’ as we propose soft mining (continuous mining score) in contrast with ‘hard’ mining (binary mining score). While in previous work, it means ‘difficult’.
In this chapter, we aim to address both problems. Specifically, for the first problem, we propose a novel sample mining method, called Online Soft Mining (OSM), which assigns one continuous score to each sample to make use of all samples in the mini-batch. OSM is based on Euclidean distance in the embedding space. OSM incorporates online soft positives mining and online soft negatives mining. For the positives mining, inspired by (Cui et al., 2016), we assign higher mining scores to more similar pairs in terms of distance. This mining has a strong connection to manifold learning (Cui et al., 2016), since it preserves helpful intraclass variances like pose and colour in the deep embedding space. For the negatives mining, more difficult negatives get higher mining scores by soft negatives mining, which can be considered as a generalisation of hard difficult negatives mining (Cui et al., 2016; Huang et al., 2016b; Oh Song et al., 2016; Simo-Serra et al., 2015; Wang & Gupta, 2015; Yuan et al., 2017). For the second problem, we propose Class-Aware Attention (CAA) that...
pays less attention to outlying data samples. CAA is simply based on a class compatibility score. The illustration of OSM and CAA is given in Figure 2.1. One attractive property of OSM and CAA is that they are generic, by which we mean they can be integrated to most existing deep metric learning methods (Chechik et al., 2010; Hadsell et al., 2006) for mining samples and addressing outliers simultaneously. To demonstrate the effectiveness of OSM and CAA, we propose a novel loss, Weighted Contrastive Loss (WCL), by integrating them into traditional contrastive loss (Hadsell et al., 2006). The weight is assigned to each pair of images defined by OSM and CAA.

Compared to previous hard mining (0/1) in metric learning, soft mining is proposed novelly in this chapter. Besides, for the first time CAA is proposed to address the influence of outliers, which is an inherent problem of mining methods. Extensive experiments are conducted on two tasks: fine-grained visual categorisation and video-based person re-identification. For the first one, we demonstrate our proposed method’s performance by comparing with other metric learning methods on CUB-200-2011 (Wah et al., 2011) and CARS196 (Krause et al., 2013) datasets. For the second one, we use MARS (Zheng et al., 2016) and LPW (Song et al., 2018) datasets, and show that our method surpasses state-of-the-art methods by a large margin.

2.2 Related Work

Sample mining in deep metric learning. With the popularity of deep learning (Girshick et al., 2016; Krizhevsky et al., 2012; Szegedy et al., 2015), deep metric learning has been widely studied in many visual tasks and achieved promising results (Oh Song et al., 2016; Schroff et al., 2015). Compared with traditional metric learning in which hard-crafted features are widely used (Chechik et al., 2010; Weinberger & Saul, 2009), deep metric learning learns feature embeddings directly from data points using deep convolutional neural networks (CNN). Although we can construct a large number of image pairs for deep metric learning, a large fraction of trivial pairs will contribute zero to the loss and gradient once the model reaches a reasonable performance. Thus it is intuitive to mine nontrivial pairs during training to achieve faster convergence and better performance. As a result, sample mining has been widely explored and a range of mining methods have been proposed (Cui et al., 2016; Huang et al., 2016b; Oh Song et al., 2016; Schroff et al., 2015; Shi et al., 2016; Simo-Serra et al., 2015; Wang & Gupta, 2015; Yuan et al., 2017). For mining negatives, mining difficult negatives is applied in (Huang et al., 2016b; Oh Song et al., 2016; Simo-Serra et al., 2015; Wang & Gupta, 2015). Mining semi-difficult negatives is studied in (Cui et al., 2016; Schroff et al., 2015). For mining positives, mining difficult positives is explored in (Huang et al., 2016b; Oh Song et al., 2016; Simo-Serra et al., 2015; Wang & Gupta, 2015).
2.3 Methodology

Mining semi-difficult positives is explored in (Simo-Serra et al., 2015; Yuan et al., 2017). Local positives mining (i.e., closer positives) is proposed for learning an extended manifold rather than a contracted hypersphere in (Cui et al., 2016). All of them mine a subset of pairs hardly by using a binary score for each pair, i.e., dropping it or keeping it. Instead, our proposed OSM conducts soft mining using a continuous score, i.e., assigning different weights for different pairs. In addition, samples with larger training loss are prioritized when difficult pair mining is applied. Therefore, outliers are generally mined and perturb the model training due to their large losses. To address this, semi-difficult negatives mining (Cui et al., 2016; Schroff et al., 2015) and multilevel mining (Yuan et al., 2017) are applied to remove outliers. Instead, we propose CAA for assigning smaller weights to outliers in the mini-batch.

In this chapter, following (Oh Song et al., 2016; Sohn, 2016; Tadmor et al., 2016), we construct one image pair between every two images in the mini-batch. N-pair-mc loss (Sohn, 2016) and Multibatch (Tadmor et al., 2016) treat all the constructed pairs equally, i.e., without any pair mining. LiftedStruct (Oh Song et al., 2016) mines most difficult negatives and treat all positives equally. In contrast, we propose online soft mining for both positives and negatives.

Intraclass variance. In fine-grained recognition (Cui et al., 2016; Krause et al., 2013; Oh Song et al., 2016; Wah et al., 2011) and person re-identification (McLaughlin et al., 2017; Song et al., 2018; Zheng et al., 2016; Zhong et al., 2017), the intraclass distance could be larger than interclass distance, e.g., images from different categories could have similar colour and shape while the images in the same category can have large variances such as colour, pose and lighting. Several approaches benefit from utilising the inherent structure of data. In particular, local similarity-aware embedding was proposed in (Huang et al., 2016b) to preserve local feature structures. (Cui et al., 2016) proposed to learn an extended manifold by using only local positives rather than treating all positives equally. In a similar spirit with (Cui et al., 2016), our OSM assigns larger weights for more similar positives, intending to learn continuous manifolds. The key difference, however, is that a subset composed of local positives are mined in (Cui et al., 2016), whereas our OSM makes use of all positives and gives local positives more attention.

2.3 Methodology

Given an image dataset $X = \{(x_i, y_i)\}$, where $x_i$ and $y_i$ are $i$-th image and the corresponding label respectively, our aim is to learn an embedding function (metric) $f$ that takes $x_i$ as input
and outputs its embedding \( f_i \in \mathbb{R}^D \). Ideally, the learned features of similar pairs are pulled together and features of dissimilar pairs are pushed away.

Our proposed weighted contrastive loss for learning the embedding function \( f \) is illustrated in Figure 2.2b. Following the common practice, each mini-batch contains \( m = c \times k \) images, where \( c \) is the number of classes and \( k \) is the number of images per class. A CNN is used for extracting the feature of each image in the mini-batch. In online pair construction, we use all images from the same class to construct positive pairs (i.e., similar pairs) and all images from different classes to construct negative pairs (i.e., dissimilar pairs). As a result, we get \( m(m - 1)/2 \) image pairs in total. There are \( ck(k - 1)/2 \) positive pairs included in the positive set, i.e., \( P = \{(x_i, x_j) | y_i = y_j \} \). Similarly, the negative set, \( N = \{(x_i, x_j) | y_i \neq y_j \} \), consists of \( ck(k - 1)/2 \) negative pairs. Then, the weight of each pair is generated by OSM and CAA jointly. Our proposed weighted contrastive loss is based on these pairs and their corresponding weights. For reference, the traditional contrastive loss (Hadsell et al., 2006) is:

\[
L_{\alpha}^{\text{cont}}(x_i, x_j; f) = y_{ij}d_{ij}^2 + (1 - y_{ij})\max(0, \alpha - d_{ij})^2,
\]

(2.1)

where \( y_{ij} = 1 \) if \( y_i = y_j \), \( y_{ij} = 0 \) if \( y_i \neq y_j \), \( d_{ij} = \|f_i - f_j\|_2 \) is the euclidean distance between the pair. It pulls positive pairs as close as possible and pushes negative pairs farther than a pre-defined margin \( \alpha \).

### 2.3.1 Online Soft Mining

Online Soft Mining contains Online Soft Positives Mining (OSPM) for the positive set and Online Soft Negatives Mining (OSNM) for the negative set.

**Online Soft Positives Mining.** The goal of OSPM is to generate OSM scores for pairs in the positive set. In many visual tasks, e.g., fine-grained visual categorisation, the interclass distance could be small compared to large intraclass distance. Motivated by manifold learning in (Cui et al., 2016) where local positives are selected out for learning extended manifolds, OSPM assigns higher OSM scores to local positives. This is because contracted hyperspheres are learned and large intraclass distance cannot be captured if positives with large distance are treated equally.

Specifically, for each similar pair in the positive set, i.e., \( (x_i, x_j) \in P \), we compute Euclidean distance \( d_{ij} \) between their features after \( L_2 \) normalisation. As we want to assign higher mining scores to more similar pairs, we simply transfer the distance \( d_{ij} \) to OSM score using a Gaussian function with mean=0. In summary, the OSM score \( s_{ij}^+ \) for each positive pair \( (x_i, x_j) \) is obtained as follows:
2.3 Methodology

(a) Traditional contrastive loss

(b) Weighted contrastive loss with OSM and CAA

Fig. 2.2: The comparison of our proposed weighted contrastive loss with traditional contrastive loss. \( m \) is the number of images in a mini-batch. Traditional contrastive loss (a) takes as input \( m/2 \) image pairs and \( m/2 \) binary labels which indicate the corresponding image pair is from the same class or not. Our proposed weighted contrastive loss (b) applies online pair construction. The weighted contrastive loss takes as input \( m \) images and \( m \) multi-class labels. We construct non-repeated \( m(m-1)/2 \) image pairs online rather than \( m^2 \) pairs in (Oh Song et al., 2016) or \( m(m-1) \) pairs in (Tadmor et al., 2016). Weighted contrastive loss combines Online Soft Mining (OSM) and Class-Aware Attention (CAA) to assign a proper weight for each image pair.

\[
\begin{align*}
\sigma_{\text{OSM}} & = \exp\left(-\frac{d_{ij}^2}{\sigma_{\text{OSM}}^2}\right) \\
\end{align*}
\]  

(2.2)

where \( d_{ij} = \|f_i - f_j\|_2 \), \( \sigma_{\text{OSM}} \) is a hyperparameter for controlling the distribution of OSM scores.

**Online Soft Negatives Mining.** For dissimilar pairs in the negative set \( N \), we wish to push each pair away by a margin \( \alpha \) inspired by the traditional contrastive loss. Similar to previous hard difficult negatives mining methods using binary mining score (Cui et al., 2016; Huang et al., 2016b; Oh Song et al., 2016; Tadmor et al., 2016; Ustinova & Lempitsky, 2016; Yuan et al., 2017), to ignore a large fraction of trivial pairs which do not contribute to learning, we assign higher OSM scores to **negative pairs whose distance is within this margin.** The OSM scores of negative pairs whose distances are larger than the pre-defined margin are set as 0 because they do not contribute to the loss and gradient. For simplicity,
the OSM score \( s_{ij} \) of each negative pair \((x_i, x_j)\) is computed directly by the margin distance:

\[
s_{ij} = \max(0, \alpha - d_{ij})
\]

(2.3)

### 2.3.2 Class-Aware Attention

As indicated in (Cui et al., 2016; Schroff et al., 2015), sample mining is easily influenced by outliers, leading the learned model to a bad local minimum. The proposed OSM is also prone to outliers, e.g., we assign more attention to more difficult negatives which may contain outliers.

In general, outliers are generally caused by corrupted labels, which hurt the model’s performance significantly (Zhang et al., 2017a). In other words, outliers are usually composed of mislabelled images, thus being less semantically related to their labels. Therefore, we propose to identify noisy images by their semantic relation to their labels. As it is guided by the class label, we name it Class-Aware Attention (CAA). The CAA score of image \( x_i \) indicates how much it is semantically related to its label \( y_i \).

To measure image’s semantic relation to its label, we compute the compatibility between image’s feature vector and its corresponding class context vector (a class-level representation). The compatibility between two vectors can be measured by their dot product (Jetley et al., 2018). We apply a classification branch after image embeddings to learn context vectors of all classes. The class context vectors are the trained parameters of the fully connected layer, i.e., \( \{c_k\}_{k=1}^{C} \), where \( C \) is the number of classes in the training set and \( c_k \in \mathbb{R}^D \) is the context vector of class \( k \). Therefore, the CAA score (image’s semantic relation to label) of \( x_i \) is computed as:

\[
a_i = \frac{\exp(f_i^T c_{y_i})}{\sum_{k=1}^{C} \exp(f_i^T c_k)}.
\]

(2.4)

The softmax operation is used for normalising image’s semantic relation over all classes.

In (Goldberger & Ben-Reuven, 2017), the softmax output (classification confidence/likelihood) is used to predict the true label. In this chapter, we apply it to estimate image’s semantic relation to its label, i.e., the correctness degree of the label.

### 2.3.3 Weighted Contrastive Loss

Now we have OSM for mining proper pairs and CAA for identifying outliers. To demonstrate the effectiveness of OSM and CAA for deep metric learning, we integrate them into the
2.3 Methodology

traditional contrastive loss. Specifically, first we generate OSM scores for both positive and negative pairs \((x_i, x_j) \in P \cup N\), and to deal with outliers at pair level we generate CAA scores for all pairs. That is,

\[
w_{ij}^+ = s_{ij}^+ a_{ij},
\]

\[
w_{ij}^- = s_{ij}^- a_{ij},
\]

where \(a_{ij}\) is the CAA score of pair \((x_i, x_j)\) and it is defined by \(\min(a_i, a_j)\).

We integrate pair weights computed by Eq. (2.5) and Eq. (2.6) into the traditional contrastive loss to formulate our weighted contrastive loss:

\[
L_{WCL}(P) = \frac{1}{2} \sum_{(x_i, x_j) \in P} \frac{w_{ij}^+ d_{ij}^2}{w_{ij}^+ + \zeta}
\]

\[
L_{WCL}(N) = \frac{1}{2} \sum_{(x_i, x_j) \in N} \frac{w_{ij}^- \max(0, \alpha - d_{ij})^2}{w_{ij}^- + \zeta}
\]

\[
L_{WCL}(P, N) = (1 - \lambda)L_{WCL}(P) + \lambda L_{WCL}(N)
\]

where hyperparameter \(\lambda\) controls the contribution of the positive set and the negative set towards final contrastive loss. \(\zeta\) is a small constant for the purpose of computational stability in case the denominator is close to zero. In our experiments, we fix \(\lambda = 0.5\), treating the positive set and negative set equally. The denominators in Eq. (2.7) and Eq. (2.8) are used for normalisation. See Figure 2.2 for the illustration of weighted contrastive loss with comparison to the traditional contrastive loss.

In particular, the number of negative pairs \((ck(ck - k)/2)\) is much larger than the number of positive pairs \((ck(k - 1)/2)\). If we normalise all the dissimilar pairs and similar pairs together, the class imbalance problem makes learning unstable. Therefore, we apply independent normalisation for the positive samples \(P\) and negative samples \(N\).

Following (Sohn, 2016; Tadmor et al., 2016), we construct an unbiased estimate of the full gradient by relying on all possible \(m(m - 1)/2\) pairs in each mini-batch (batch size \(m \ll\) data size). The layers for online pair construction and contrastive loss computation are put behind CNN embeddings, thus each image goes through CNN only once to obtain one embedding. Each embedding vector is used to construct multiple pairs with other embeddings and compute contrastive loss.
2.4 Experiments

We compare our method with state-of-the-art embedding methods on two tasks from different domains, i.e., fine-grained visual categorisation and video-based person re-identification. For both tasks, testing and training classes are disjoint. This setting is common as the practical deep metric should generalise to agnostic classes.

**Implementation details.** We use GoogLeNet V2 (Ioffe & Szegedy, 2015) as the backbone architecture. For fine-grained categorisation, we set $c = 8$ and $k = 7$ for each mini-batch, while $c = 3$ and $k = 18$ for video-based person re-identification. We set empirically $\sigma_{\text{OSM}} = 0.8$ and $\sigma_{\text{CAA}} = 0.18$ for all experiments to avoid adjusting them based on the specific dataset, making it general and fair for comparison. In more detail, first we find that $\sigma_{\text{OSM}} = 0.8$ and $\sigma_{\text{CAA}} = 0.18$ perform well on MARS (Zheng et al., 2016), a large person re-identification dataset. Then we apply them directly to other datasets. We conjecture that better results may be obtained by tuning $\sigma_{\text{OSM}}$ and $\sigma_{\text{CAA}}$ on other benchmarks. Following the practice of other methods, when training a model, the weights are initialised by a pretrained model on ImageNet (Russakovsky et al., 2015). For all datasets, the images are resized to $224 \times 224$ during training and testing. No data augmentation is applied for both phases. For optimisation, Stochastic Gradient Decent (SGD) is used with a learning rate of 0.001, a momentum of 0.9. For learning stability, we train another 10 epochs after convergence. The margin of weighted contrastive loss $\alpha$ is set to 1.2 for all experiments. We implement our method in the Caffe (Jia et al., 2014) deep learning framework.

2.4.1 Fine-Grained Visual Categorisation

Fine-grained visual categorisation task is widely used for evaluating deep metric learning methods.

**Datasets and evaluation protocol.** We conduct experiments on two benchmarks:

- CUB-200-2011 contains 11,788 images of 200 species of birds. The first 100 classes (5,864 images) are used for training and the remaining classes (5,924 images) for testing.

- CARS196 is composed of 16,185 images of 196 types of cars. The first 98 classes containing 8,054 images are for training and the remaining 98 classes (8,131 images) for testing.

Both datasets come with two versions: raw images (containing large background with respect to a region of interest) and cropped images (containing only regions of interest). For a fair
2.4 Experiments

Table 2.1: Comparison with state-of-the-art methods on CARS196 and CUB-200-2011 in terms of Recall@$K$ (%). The raw images are used for training and testing for methods in the first group. The cropped images are used for training and testing for methods in the second group. * indicates cascaded models are applied for sample mining and learning embeddings.

<table>
<thead>
<tr>
<th></th>
<th>CARS196</th>
<th></th>
<th>CUB-200-2011</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>Contrastive (Bell &amp; Bala, 2015)</td>
<td>21.7</td>
<td>32.3</td>
<td>46.1</td>
<td>58.9</td>
</tr>
<tr>
<td>Triplet (Schroff et al., 2015)</td>
<td>39.1</td>
<td>50.4</td>
<td>63.3</td>
<td>74.5</td>
</tr>
<tr>
<td>LiftedStruct (Oh Song et al., 2016)</td>
<td>49.0</td>
<td>60.3</td>
<td>72.1</td>
<td>81.5</td>
</tr>
<tr>
<td>Binomial Deviance (Ustinova &amp; Lempitsky, 2016)</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Histogram Loss (Ustinova &amp; Lempitsky, 2016)</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Smart Mining (Harwood et al., 2017)</td>
<td>64.7</td>
<td>76.2</td>
<td>84.2</td>
<td>90.2</td>
</tr>
<tr>
<td>HDC* (Yuan et al., 2017)</td>
<td>73.7</td>
<td>83.2</td>
<td>89.5</td>
<td>93.8</td>
</tr>
<tr>
<td>Ours</td>
<td>74.0</td>
<td>83.8</td>
<td>90.2</td>
<td>94.8</td>
</tr>
<tr>
<td>PDDM+Triplet (Huang et al., 2016b)</td>
<td>46.4</td>
<td>58.2</td>
<td>70.3</td>
<td>80.1</td>
</tr>
<tr>
<td>PDDM+Quadrupal (Huang et al., 2016b)</td>
<td>57.4</td>
<td>68.6</td>
<td>80.1</td>
<td>89.4</td>
</tr>
<tr>
<td>HDC* (Yuan et al., 2017)</td>
<td>83.8</td>
<td>89.8</td>
<td>93.6</td>
<td>96.2</td>
</tr>
<tr>
<td>Ours</td>
<td>85.5</td>
<td>91.5</td>
<td>95.1</td>
<td>97.2</td>
</tr>
</tbody>
</table>

In both settings of datasets, i.e., raw images and cropped images, our method outperforms all the compared methods. Generally, our method improves state-of-the-art performance by around 1.5%.

The only method that is competitive to ours is HDC (Yuan et al., 2017). However, it is important to note that HDC (Yuan et al., 2017) is based on cascaded embedding requiring several models, while ours is a single model.

2.4.2 Video-based Person Re-identification

To further evaluate the effectiveness of our method, we conduct experiments on video-based person re-identification task.

Datasets and evaluation protocol. In this task, we evaluate our methods on two large-scale benchmark datasets: MARS (Zheng et al., 2016) consists of images (frames) with huge variations due to camera setup, yielding many noisy images and outliers, and LPW (Song
Table 2.2: Video-based person re-identification datasets: MARS and LPW. # indicates the number of corresponding terms. DT failure denotes whether there exists detection or tracking failure in the tracklets.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>#identities</th>
<th>#boxes</th>
<th>#tracklets</th>
<th>#cameras</th>
<th>DT failure</th>
<th>resolution</th>
<th>annotation</th>
<th>evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>MARS</td>
<td>1261</td>
<td>1,067,516</td>
<td>20,715</td>
<td>6</td>
<td>Yes</td>
<td>256x128</td>
<td>detector</td>
<td>CMC &amp; mAP</td>
</tr>
<tr>
<td>LPW</td>
<td>2731</td>
<td>590,547</td>
<td>7694</td>
<td>11</td>
<td>No</td>
<td>256x128</td>
<td>detector</td>
<td>CMC</td>
</tr>
</tbody>
</table>

et al., 2018) is a cross-scene video dataset. Note that LPW is more challenging than MARS dataset, since data is collected under different scenes, whereas MARS is collected in only one scene, around university campus. The detailed information about MARS and LPW datasets are summarised in Table 2.2. We follow exactly the evaluation setting in Zheng et al. (2016) and Song et al. (2018), respectively. We report the Cumulated Matching Characteristics (CMC)\(^2\) results for both datasets. We also compute the mean average precision (mAP) for MARS.

Comparative evaluation. The results of MARS and LPW are shown in Table 2.3 and Table 2.4 respectively. The results on both datasets demonstrate the effectiveness of our proposed approach. Our observations are as follows:

- On MARS dataset, our method outperforms all the methods by a large margin. Compared with CAE (Chen et al., 2018)\(^3\), ours improves 2.3% and 4.9% in terms of CMC-1 and mAP respectively; When using re-ranking as post-processing in the test phase, the performance of our method further increases.

- On LPW dataset, we can see that our method again outperforms state-of-the-art methods by a large margin.

2.4.3 Ablation Study

Our key contributions in this chapter are OSM and CAA. To evaluate the contribution of each module, we conduct ablation studies on CUB-200-2011, LPW, and MARS datasets. The details are as follows:

- Baseline: Neither OSM or CAA is used. Online pair construction is applied. The weight of each pair is fixed as one in the loss functions Eq. (2.7) and Eq. (2.8). Specifically, \(w_{ij}^+ = 1\) and \(w_{ij}^- = 1\).

\(^2\)CMC@\(K\) is the same as Recall@\(K\). Generally, the term CMC is used in person re-identification.

\(^3\)For CAE, we present the results of complete sequence instead of multiple snippets so that it can be compared with other methods. Multiple snippets can be regarded as data augmentation in the test phase.
2.4 Experiments

Table 2.3: Comparison with state-of-the-art methods on MARS in terms of CMC(%) and mAP(%). The methods using attention mechanism are indicated by ‘Attention’ column. *Net denotes that a used network is self-designed and varies in different papers.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Attention</th>
<th>1</th>
<th>5</th>
<th>20</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDE (ResNet50) (Zhong et al., 2017)</td>
<td>No</td>
<td>62.7</td>
<td>–</td>
<td>–</td>
<td>44.1</td>
</tr>
<tr>
<td>IDE+XQDA (ResNet50) (Zhong et al., 2017)</td>
<td>No</td>
<td>70.5</td>
<td>–</td>
<td>–</td>
<td>55.1</td>
</tr>
<tr>
<td>IDE+XQDA+Re-ranking (ResNet50) (Zhong et al., 2017)</td>
<td>No</td>
<td>73.9</td>
<td>–</td>
<td>–</td>
<td>68.5</td>
</tr>
<tr>
<td>CNN+RNN (*Net) (McLaughlin et al., 2017)</td>
<td>No</td>
<td>43.0</td>
<td>61.0</td>
<td>73.0</td>
<td>–</td>
</tr>
<tr>
<td>CNN+RNN+XQDA (*Net) (McLaughlin et al., 2017)</td>
<td>No</td>
<td>52.0</td>
<td>67.0</td>
<td>77.0</td>
<td>–</td>
</tr>
<tr>
<td>AMOC+EpicFlow (*Net) (Liu et al., 2017a)</td>
<td>No</td>
<td>68.3</td>
<td>81.4</td>
<td>90.6</td>
<td>52.9</td>
</tr>
<tr>
<td>CNN+RNN (*Net) (McLaughlin et al., 2017)</td>
<td>No</td>
<td>44.0</td>
<td>70.0</td>
<td>81.0</td>
<td>–</td>
</tr>
<tr>
<td>SRM+TAM (CaffeNet) (Zhou et al., 2017)</td>
<td>Yes</td>
<td>70.6</td>
<td>90.0</td>
<td>97.6</td>
<td>50.7</td>
</tr>
<tr>
<td>RQEN (GoogLeNet V2) (Song et al., 2018)</td>
<td>Yes</td>
<td>73.7</td>
<td>84.9</td>
<td>91.6</td>
<td>51.7</td>
</tr>
<tr>
<td>RQEN+XQDA+Re-ranking (GoogLeNet V2) (Song et al., 2018)</td>
<td>Yes</td>
<td>77.8</td>
<td>88.8</td>
<td>94.3</td>
<td>71.1</td>
</tr>
<tr>
<td>DRSA (ResNet50) (Li et al., 2018)</td>
<td>Yes</td>
<td>82.3</td>
<td>–</td>
<td>–</td>
<td>65.8</td>
</tr>
<tr>
<td>CAE (ResNet50) (Chen et al., 2018)</td>
<td>Yes</td>
<td>82.4</td>
<td>92.9</td>
<td>–</td>
<td>67.5</td>
</tr>
<tr>
<td>Ours (GoogLeNet V2)</td>
<td>Yes</td>
<td>84.7</td>
<td>94.1</td>
<td>97.0</td>
<td>72.4</td>
</tr>
<tr>
<td>Ours+Re-ranking (GoogLeNet V2)</td>
<td>Yes</td>
<td>86.0</td>
<td>94.4</td>
<td>97.1</td>
<td>81.0</td>
</tr>
</tbody>
</table>

Table 2.4: Comparison with state-of-the-art methods on LPW in terms of CMC (%). The methods using attention mechanism are indicated by ‘Attention’ column.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Attention</th>
<th>1</th>
<th>5</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>GoogLeNet V2 (Song et al., 2018)</td>
<td>No</td>
<td>41.5</td>
<td>66.7</td>
<td>86.2</td>
</tr>
<tr>
<td>RQEN (Song et al., 2018)</td>
<td>Yes</td>
<td>57.1</td>
<td>81.3</td>
<td>91.5</td>
</tr>
<tr>
<td>Ours</td>
<td>Yes</td>
<td>71.7</td>
<td>89.8</td>
<td>96.6</td>
</tr>
</tbody>
</table>

Table 2.5: Ablation studies on CUB-200-2011, LPW, MARS in terms of Recall@K (%) and mAP (%).

<table>
<thead>
<tr>
<th>K</th>
<th>CUB-200-2011</th>
<th>LPW</th>
<th>MARS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>GoogLeNet V2</td>
<td>47.8</td>
<td>61.1</td>
<td>73.2</td>
</tr>
<tr>
<td>Baseline</td>
<td>52.0</td>
<td>65.0</td>
<td>76.2</td>
</tr>
<tr>
<td>OSM</td>
<td>54.2</td>
<td>66.8</td>
<td>76.9</td>
</tr>
<tr>
<td>OSM+CAA</td>
<td><strong>55.3</strong></td>
<td><strong>67.3</strong></td>
<td><strong>77.5</strong></td>
</tr>
</tbody>
</table>

- OSM: When mining negatives, OSM generates higher scores for difficult negatives and lower scores for trivial negative samples. When mining positives, OSM generates higher scores for local positives to preserves intraclass variances. CAA is not used. Specifically, \( w_{ij}^+ = s_{ij}^+ \) and \( w_{ij}^- = s_{ij}^- \).
• OSM+CAA: As mentioned before, OSM is prone to outliers similar to other mining methods, which is an inherent problem of mining approaches. Therefore, we combine OSM with CAA to conduct soft mining as well as address outliers. Specifically, 
\[ w_{ij}^+ = s_{ij}^+ \ast a_{ij} \text{ and } w_{ij}^- = s_{ij}^- \ast a_{ij}. \]

The results are summarized in Table 2.5. We can see that OSM alone can achieve very competitive performance compared with the baseline. After integrating CCA to OSM, the accuracy increases on all datasets by about 1-2%. It is easy to see that both OSM and CCA contribute positively toward the final performance.

Furthermore, we notice that the test classes of CUB-200-2011 overlap with ImageNet (Russakovsky et al., 2015) dataset. To verify how much we actually improve, we show the performance of pre-trained GoogLeNet V2 (Ioffe & Szegedy, 2015) without any fine-tuning. The result is shown in the first row of Table 2.5. Just using GoogLeNet V2 with pre-trained weights, the performance is 47.8% – already beating most of the methods shown in Table 2.1. In contrast, our method outperforms pre-trained GoogLeNet V2 by a large margin.

2.5 Conclusion

In this chapter, we propose a simple yet effective mining method named OSM. As suggested by prior work, the principle of OSM is not only to mine nontrivial data pairs for accelerating convergence, but also to learn extended manifolds for preserving intraclass variances. Besides, we propose CAA to address the impact of outliers. Finally, to evaluate the effectiveness of these two modules in action, we propose weighted contrastive loss by combining OSM and CAA to learn discriminative embeddings. The experiments on fine-grained visual categorisation and video-based person re-identification with four datasets demonstrate the superiority of our method. For future work, it is interesting to see the integration of OSM and CAA for other kinds of losses such as triplet loss.
Chapter 3

Example Weighting for Ranked List Loss

The objective of deep metric learning (DML) is to learn embeddings that can capture semantic similarity and dissimilarity information among data points. Existing pairwise or tripletwise loss functions used in DML are known to suffer from slow convergence due to a large proportion of trivial pairs or triplets as the model improves. To improve this, ranking-motivated structured losses are proposed recently to incorporate multiple examples and exploit the structured information among them. They converge faster and achieve state-of-the-art performance. In this chapter, we unveil two limitations of existing ranking-motivated structured losses and propose a novel ranked list loss to solve both of them. First, given a query, only a fraction of data points is incorporated to build the similarity structure. Consequently, some useful examples are ignored and the structure is less informative. To address this, we propose to build a set-based similarity structure by exploiting all instances in the gallery. The learning setting can be interpreted as few-shot retrieval: given a mini-batch, every example is iteratively used as a query, and the other ones compose the gallery to search, i.e., the support set in a few-shot setting. The rest of the examples are split into a positive set and a negative set. For every mini-batch, the learning objective of ranked list loss is to make the query closer to the positive set than to the negative set by a margin. Second, previous methods aim to pull positive pairs as close as possible in the embedding space. As a result, the intraclass data distribution tends to be extremely compressed. In contrast, we propose to learn a hypersphere for each class in order to preserve useful similarity structure inside it, which functions as regularisation. Extensive experiments demonstrate the superiority of our proposal by comparing with the state-of-the-art methods on the fine-grained image retrieval task. Our source code is available online: https://github.com/XinshaoAmosWang/Ranked-List-Loss-for-DML.
3.1 Introduction

Deep metric learning (DML) plays a crucial role in a variety of applications in computer vision, such as image retrieval (Movshovitz-Attias et al., 2017; Sohn, 2016), clustering (Hershey et al., 2016), and transfer learning (Oh Song et al., 2016). For example, by using DML, FaceNet (Schroff et al., 2015) achieves superhuman performance on face verification with 260M face images of 8M identities.

Loss function is one of essential components in successful DML frameworks and a large variety of loss functions have been proposed in the literature. Contrastive loss (Chopra et al., 2005; Hadsell et al., 2006) captures the relationship between pairwise data points, i.e., similarity or dissimilarity. Triplet-based losses are also widely studied (Cui et al., 2016; Schroff et al., 2015; Wang et al., 2014). A triplet is composed of an anchor point, a similar (positive) data point and dissimilar (negative) data point. The purpose of triplet loss is to learn a distance metric by which the anchor point is closer to the similar point than the dissimilar one by a margin. In general, the triplet loss outperforms the contrastive loss (Oh Song et al., 2016; Schroff et al., 2015) because the relationship between positive and negative pairs is considered. Inspired by this, recent ranking-motivated methods¹ (Law et al., 2017; Movshovitz-Attias et al., 2017; Oh Song et al., 2016; Schroff et al., 2015; Sohn, 2016; Song et al., 2017) propose to take into consideration the richer structured information among multiple data points and achieve impressive performance on many applications, e.g., fine-grained image retrieval and clustering.

However, there are still certain limitations in current state-of-the-art DML approaches. Firstly, we notice that only a proportion of informative examples is incorporated to capture the structure in previous ranking-motivated loss functions. In this case, some non-trivial examples are wasted and the structured information is extracted from fewer data points. To address it, we propose to utilise all non-trivial data points to build a more informative structure and exploit it to learn more discriminative embeddings. Specifically, given a query, we obtain a ranked list by sorting all other data points (gallery) according to the similarities. Ideally, all the positive examples are supposed to be ranked before the negative samples in the feature space. To achieve this, we introduce ranked list loss (RLL) to organise the samples of each query. Given a query, the optimisation of RLL is to rank all positive points before the negative points and forcing a margin between them. In other words, RLL aims to explore the set-based similarity structure, which contains richer information than the point-based approach, e.g., triplet loss.

¹We term them ranking-motivated methods, whose target is to make the largest distance of similar pairs smaller than the smallest distance of dissimilar pairs. In this context, we do not care about the distance order inside positive and negative sets. Namely, rank and retrieve can be used interchangeably here.
3.1 Introduction

Secondly, we observe that the intraclass data distribution is not considered in the previous structured losses. All algorithms (Goldberger et al., 2005; Movshovitz-Attias et al., 2017; Oh Song et al., 2016; Schroff et al., 2015; Sohn, 2016; Song et al., 2017; Triantafillou et al., 2017) target to pull data points in the same class as close as possible. Consequently, these approaches try to shrink samples of the same class into one point in the feature space and may easily drop their similarity structure. To solve this, we propose to learn a hypersphere for each class in RLL. Specifically, instead of pulling intraclass examples as compact as possible, we only force the distance of a positive pair smaller than a threshold, which is the diameter of each class’s hypersphere. In this case, RLL can explicitly preserve the intraclass similarity structure inside each class within the hypersphere.

Empirically, the convergence rate of DML methods highly depends on the possibility of seeing non-trivial samples (Schroff et al., 2015). Given a query (anchor), it is non-trivial to separate the positive and negative sets by a margin when all data points are considered. As a result, only a few ranked lists are perfectly optimized as the model improves during training. Therefore, our method can take advantage of a maximum of elements with non-zero losses.
and release the potentials for the learning procedure. The proposed RLL is illustrated in Figure 3.1.

Furthermore, a recent work (Achille et al., 2019) studied the existing of critical learning periods in artificial learning systems, which represents the time during which a temporary stimulus deficit may cause a permanent skill impairment in biological systems. Specifically, it shows the critical period in classification tasks using Fisher Information and Information Plasticity. Instead, we study and validate the critical learning period in deep metric learning via dynamic example weighting. Verifying the existing of critical learning periods is important, which can help us understand why network initialisation and warming-up training strategies have a large impact on the final performance. More importantly, it inspires us to be more careful and spend more effort on monitoring the early learning process of a deep network in the future work.

In short, our contributions in this paper are listed as follows:

- We propose a novel ranking-motivated structured loss, named Ranked List Loss (RLL), to learn discriminative embeddings with the setting of few-shot retrieval. In contrast with previous ranking-motivated losses, we are the first to incorporate all non-trivial data points and exploit the structure among them. Besides, we learn a hypersphere for each class to preserve intraclass data distribution instead of shrinking each class into one point in the embedding space.

- We propose two versions of RLL, i.e., the full version and a simplified version termed RLL-Simpler. As indicated by its name, RLL-Simpler simplifies the full version and is more preferable for exploration in practice, because it has only two hyper-parameters.

- Using RLL-Simpler, we obtain the state-of-the-art performance on two large datasets, i.e., SOP (Oh Song et al., 2016) and In-shop Clothes (Liu et al., 2016b). Then, using the full version of RLL, we present comprehensive ablation studies for understanding the vital factors in deep metric learning. Specifically, the study on the network depth is of high practical value.

The rest of this paper is organised as follows: Section 3.2 introduces some basic notations and preliminaries of deep metric learning, e.g., how a loss function is related to deep metric learning, prior work and practical strategies. We present our method in section 3.3. Specifically, in section 3.3.7, we introduce a dynamic weighting scheme and use it to study the critical learning periods in the context of deep metric learning for the first time. In section 3.4, we show extensive experiments to compare with related baselines and comprehensively study important components. The critical learning period of deep metric learning is studied in section 3.5. Finally, we make a summary of this work in section 3.6.
3.2 Preliminaries and Related Work

**Notations.** We use bold capital characters to denote matrices and set. Bold lower-case letters are used to represent vectors. Scalar variables are denoted by non-bold letters. Let $X = \{(x_i, y_i)\}_{i=1}^{N}$ be the input data, where $(x_i, y_i)$ indicates $i$-th image and its corresponding class label. The total number of classes is $C$, i.e., $y_i \in [1, 2, ..., C]$. The images from $c$-th class are represented as $\{x_i^c\}_{i=1}^{N_c}$, where $N_c$ is the number of images in $c$-th class. For every $x_i$, we use $y_i$ and superscript interchangeably to indicate its class label.

3.2.1 Design of Loss Functions for Learning Discriminative Deep Representations

In this subsection, we briefly introduce the relationship between design of loss functions for learning discriminative representations and deep distance metric learning.

Generally, metric learning aims to learn a metric to measure the distance between two samples. For example, Neighbourhood Components Analysis (NCA) (Goldberger et al., 2005) learns a linear transformation from an input space $X$ to a metric space $AX$. In NCA, $d(x_i, x_j) = (Ax_i - Ax_j)^\top (Ax_i - Ax_j) = (x_i - x_j)^\top A^\top A(x_i - x_j)$. Here $A^\top A$ is a learned metric while $d(x_i, x_j)$ is the distance between $x_i$ and $x_j$ in the metric space. However, from the perspective of feature learning, metric learning learns an embedding function $f$, e.g., $f(x) = Ax$. In the embedding (metric) space, the distance between every pair is computed by their Euclidean distance. A loss function, e.g., NCA for preserving neighbourhood structure, is needed to supervise the learning of an embedding function $f$. Therefore, we can see that the loss function defines how we compute the distance between two instances in the feature space. As a result, the design of a loss function is crucial in metric learning (discriminative embedding learning).

In the context of deep metric learning, a deep neural network is used as a non-linear encoding function. To empirically provide supervision information in the loss layer, there are three widely applied approaches:

- **Learning to fulfil instance-to-class similarity relationship constraints**, i.e., pulling an example’s feature towards its class centre while pushing it away from other class centres simultaneously. After encoding input data points, this approach is usually implemented with a fully connected layer for computing the dot product between an example’s feature and all class centres, i.e., the weight vectors of the fully connected layer. The outputs are termed logits, which are normalised with a softmax transformation. Finally, the cross entropy is applied to ensure the instance-to-class similarity relationship
constraint is met. There are many variants of this approach, including L2-Softmax (Ranjan et al., 2017), Large-margin Softmax (Liu et al., 2016a), Angular Softmax (Liu et al., 2017b), NormFace (Wang et al., 2017a), AM-Softmax (Wang et al., 2018a), CosFace (Wang et al., 2018b) and ArcFace (Deng et al., 2018).

- **Learning according to the instance-to-proxy similarity relationship constraints.** This idea is an interesting extension of instance-to-class similarity relationship modelling. In this approach, classes are represented by proxies (Movshovitz-Attias et al., 2017; Qian et al., 2019). It is interesting in that more flexibility is given: (1) The number of proxies can be smaller than the number of training classes, in which case multiple classes are assigned to the same proxy. It is named fractional proxy assignment, i.e., one proxy represents multiple classes (Movshovitz-Attias et al., 2017); (2) We can also represent one class using multiple proxies (Movshovitz-Attias et al., 2017; Qian et al., 2019); (3) When semantic class labels are available, we can apply static proxy assignment analogously to instance-to-class modelling. In cases where semantic labels are not given, we need to exploit dynamic proxy assignment according to the distances between a data point with all proxies (Movshovitz-Attias et al., 2017). (4) When the number of proxies is the same as the number of classes and static proxy assignment is applied, this approach becomes equivalent to instance-to-class similarity relationship modelling.

- **Learning to meet instance-to-instance similarity relationship constraints.** Intrinsically, high-order similarity relationship, i.e., ranking motivated algorithms, are derived over pairwise relationship. In this context, we label that two data points are either similar or dissimilar, without the need to know how many training classes are given and which class each sample belongs to. The proposed methods according to instance-to-instance similarity relationship include contrastive loss (Chopra et al., 2005; Hadsell et al., 2006), triplet loss (Hoffer & Ailon, 2015; Schroff et al., 2015), Lifted Struct (Oh Song et al., 2016), N-pair-mc (Sohn, 2016), Histogram loss (Ustinova & Lempitsky, 2016), Angular loss (Wang et al., 2017b), Sampling Matters (Wu et al., 2017), OSM and CAA (Wang et al., 2019c), Multi-similarity Weighting (Wang et al., 2019a), ICE (Wang et al., 2019e) and so on.

In this chapter, we focus on studying the third approach. Compared with instance-to-class or instance-to-proxy similarity relationship constraints, pairwise constraint is a good solution for the challenging extreme object recognition setting (Prabhu & Varma, 2014; Yen et al., 2016), in which there exist an enormous number of classes and only a few images per class. In our experiments, we evaluate our proposed method in such a setting. Additionally,
pairwise similarity constraint is more suitable for the incremental (online) learning setting. When new training classes come, by instance-to-instance distance metric learning, we do not need to verify whether those new coming classes have occurred in the previous training stage. Moreover, knowledge transfer and re-learning is also more straightforward since we do not need to retrain a new fully connected layer. Theoretically, instance-to-instance similarity relationship modelling is more applicable for scaling up vision recognition.

3.2.2 Structured Losses

Ranking-Motivated Structured Losses

**Triplet Loss** (Schroff et al., 2015; Weinberger et al., 2006) aims to pull the anchor point closer to the positive point than to the negative point by a fixed margin $m$:

$$L(X; f) = \frac{1}{|\Gamma|} \sum_{(i,j,k) \in \Gamma} [d_{ij}^2 + m - d_{ik}^2]_+,$$

where $\Gamma$ is the set of triplets, $i$, $j$ and $k$ are the indexes of anchor, positive and negative points, respectively. $f$ is the embedding function, $d_{ij} = ||f(x_i) - f(x_j)||_2$ is the Euclidean distance. $[\cdot]_+$ is the hinge function.

**N-pair-mc** (Sohn, 2016) exploits the structured relationship among multiple data points to learn the embedding function. Triplet loss pulls one positive point while pushing a negative one simultaneously. To improve the triplet loss by interacting with more negative classes and examples, $N$-pair-mc aims to identify one positive example from $N-1$ negative examples of $N-1$ classes (one negative example per class):

$$L\left(\{(x_i, x_j^+)^N_{i=1}; f\}\right) = \frac{1}{N} \sum_{i=1}^N \log \left\{ 1 + \sum_{j \neq i} \exp(f_i^T f_j^+ - f_i^T f_i^+) \right\},$$

where $f_i = f(x_i)$ and $\{(x_i, x_j^+)^N_{i=1}\}$ are $N$ pairs of examples from $N$ different classes, i.e., $y_i \neq y_j, \forall i \neq j$. Here, $x_i$ and $x_j^+$ are the query and the positive example respectively. $\{x_j^+, j \neq i\}$ are the negative examples.

**Lifted Struct** (Oh Song et al., 2016) is proposed by Song et al. to learn the embedding function by incorporating all negative examples. The objective of Lifted Struct is to pull one positive pair $(x_i^+, x_j^+)$ as close as possible and pushes all negative data points corresponding to $x_i^+$ or $x_j^+$ farther than a margin $\alpha$. Mathematically:

$$L(X; f) = \frac{1}{2|P|} \sum_{(i,j) \in P} \left\{d_{ij} + \log \left( \sum_{(i,k) \in N} \exp(\alpha - d_{ik}) + \sum_{(j,l) \in N} \exp(\alpha - d_{jl}) \right) \right\}_+,$$
where \( P \) and \( N \) respectively represent the sets of positive pairs and negative pairs. Given the query \( x_i \), Lifted Struct intends to identify one positive example from all corresponding negative data points.

Proxy-NCA (Movshovitz-Attias et al., 2017) is proposed to address the sampling problem using proxies. The proxy \( W \) is a small set of data points that represent training classes in the original data. The proxy for \( u \) is chosen by:

\[
p(u) = \arg\min_{w \in W} d(u, w), \tag{3.4}
\]

\( p(u) \) denotes the closest point to \( u \) from \( W \). The Proxy-NCA loss is the traditional NCA loss defined over proxies instead of the original data points:

\[
L(a, u, Z) = -\log\left(\frac{\exp(-d(a, p(u)))}{\sum_{z \in Z} \exp(-d(a, p(z)))}\right), \tag{3.5}
\]

where \( Z \) is the negative set, \( p(u) \) and \( p(z) \) are the proxies of positive and negative points, respectively. \( a \) is the anchor and \( d(\cdot, \cdot) \) is the Euclidean distance between two points. With static proxy assignment, i.e., one proxy per class, the performance is much better than dynamic proxy assignment. However, the proxies in the static proxy assignment are learned during training and similar to the class vectors of the fully connected layer in classification. Recently, SoftTriple (Qian et al., 2019) further improves this proxy-based idea and achieves promising performance. The main drawback is that the scalability to extremely large datasets is not theoretically guaranteed.

The proposed RLL is ranking-motivated structured loss, which avoids two limitations of traditional methods by incorporating all non-trivial data points and exploring intrinsic structured information among them. The illustration and comparison of different ranking-motivated losses and ours are presented in Figure 3.2.

### Directly Learning to Retrieve

Inspired by information retrieval, there are many methods proposed to maximise mean Average Precision over retrieved results of queries (Lim & Lanckriet, 2014; Liu, 2009; McFee & Lanckriet, 2010). We briefly present some recent representative work as follows.

**Information Retrieval Lens** (Triantafillou et al., 2017) proposes a new form of a few-shot learning task, i.e., few-shot retrieval. Every batch is an independent task, composed of \( n \) classes and \( k \) images per class. Instead of splitting a batch into a support set (a training set) and a query set as done in Prototypical Networks (Snell et al., 2017), Information Retrieval Lens (Triantafillou et al., 2017) iteratively uses each data point in the batch as a ‘query’
Fig. 3.2: Illustration of different ranking-motivated structured losses. Different shapes (circle, triangle and square) represent different classes. For simplicity, only 3 classes are shown. The purple circle is an anchor (query). In triplet (Schroff et al., 2015), the anchor is compared with only one negative example and one positive example. In $N$-pair-mc (Sohn, 2016), Proxy-NCA (Movshovitz-Attias et al., 2017) and Lifted Struct (Oh Song et al., 2016), one positive example and multiple negative classes are incorporated. $N$-pair-mc randomly selects one example per negative class. Proxy NCA pushes the anchor away from negative proxies instead of negative examples. The proxy is class-level and can represent any instance in the corresponding class. Lifted Struct uses all examples from all negative classes. On the contrary, our proposed Ranked List Loss not only exploits all negative examples, but also makes use of all positive ones. Best viewed in colour.

to rank the remaining ones. The optimisation objective of (Triantafillou et al., 2017) is to maximise the mean Average Precision (mAP) over those rankings.

*FastAP* (Cakir et al., 2019) approximately optimises the mean Average Precision via distance quantization. Optimising mAP directly is highly challenging, thus FastAP exploits quantization-based approximation to reduce the complexity and improve the efficiency. Therefore, FastAP is tailored for optimising mAP using stochastic gradient descent.

*Prec@K* (Lu et al., 2019) means top-$K$ precision, or Recall@$K$ performance metric, which targets at optimising its top $K$ retrieved neighbours. To optimise Prec@$K$, it proposes to emphasise on misplaced images near the decision boundary, i.e., besides the $K$-th nearest neighbours. Concretely, those prioritised misplaced images are: (1) similar ones which do not belong to $K$ nearest neighbours, but are close to $K$-th nearest neighbour. The optimisation objective is to pull them into $K$ most nearest set; (2) dissimilar ones which are in the $K$ nearest neighbours set, and are close the $K$-th nearest neighbour. The optimisation target is to push them out of the $K$ nearest neighbours set.

In this paper, we also show that our proposed algorithm follows the setting of few-shot retrieval (Triantafillou et al., 2017). It optimises the distances of positive and negative pairs instead of average precision. Additionally, Prec@$K$ is partially motivated by our CVPR 2019 conference version (Wang et al., 2019b). FastAP (Cakir et al., 2019) is published in CVPR 2019, being concurrent with ours.
Clustering-Motivated Structured Losses

**Struct Clust** (Song et al., 2017) is recently proposed to learn the embedding function $f$ by optimising the clustering quality metric. The proposed structured loss function is defined as:

$$L(X; f) = [F(X, \hat{y}; f) + \gamma \triangle(y, \hat{y})]_+,$$

(3.6)

$$\triangle(y, \hat{y}) = 1 - \text{NMI}(y, \hat{y}),$$

(3.7)

where $\hat{y}$ and $y$ are the predicted and ground-truth clustering assignments respectively. $F$ measures the quality of the clustering on $X$ with the label assignment and distance metric. $\text{NMI}(y, \hat{y})$ is the normalised mutual information (Schütze et al., 2008). NMI is 1 if the predicted clustering assignment is as good as the ground-truth and 0 if it is the worst. $\hat{y}$ is predicted based on the learned distance metric $f$ and Struct Clust (Song et al., 2017) aims to learn $f$ such that the $F$ of the ground-truth assignment is larger than any other predicted clustering assignment.

However, this algorithm is NP-hard as we need to optimise both the clustering medoids and the distance metric simultaneously. As a result, the loss augmented inference and refinement are applied to select facilities (clustering medoids) based on the greedy algorithm (Mirzasoleiman et al., 2015). Large enough greedy search iterations are needed to find a local optimum, which might be costly.

**Spectral Clust** (Law et al., 2017) also aims to optimise the quality of the clustering. Spectral Clust relaxes the problem of clustering with Bregman divergences (Banerjee et al., 2005) and computes the gradient in a closed-form, which reduces the algorithmic complexity of existing iterative methods, e.g., Struct Clust (Song et al., 2017). However, it is still non-trivial to learn deep models based on mini-batch implementation. Large batch size (i.e., $1260 = 18 \text{ classes } \times 70 \text{ samples per class}$) is required for the clustering in the mini-batch. As a result, Spectral Clust iteratively computes submatrices and concatenates them into a single matrix for computing the loss and gradient, which is computationally expensive.

Both ranking-motivated and clustering-motivated structured loss functions exploit the structured similarity information among multiple data points. However, in general, clustering-motivated losses are more difficult to optimise than ranking-motivated losses.

### 3.2.3 Mining and Weighting Non-trivial Examples

Example mining strategies are widely applied in existing methods (Cui et al., 2016; Huang et al., 2016b; Lu et al., 2019; Oh Song et al., 2016; Schroff et al., 2015; Shi et al., 2016; Simo-Serra et al., 2015; Sohn, 2016; Wang & Gupta, 2015; Wang et al., 2019c; Wu et al., 2017; Yuan
et al., 2017) to provide non-trivial examples for faster convergence and better performance. Mining strategies vary in different cases. For example, FaceNet (Schroff et al., 2015) proposes to mine semi-hard negative samples. In N-pair-mc (Sohn, 2016), hard negative class mining is proposed to provide informative negative examples. In Lifted Struct (Oh Song et al., 2016), harder negative examples are emphasized in a soft way. Given a query, Sampling Wisely (Lu et al., 2019) proposes to select misplaced images near the decision boundary of its top-K nearest neighbours. Namely, Sampling Wisely (Lu et al., 2019) prioritises data points near the top-K decision boundary. In Divide & Conquer (Sanakoyeu et al., 2019), the embedding space is split into non-overlapping subspaces. Accordingly, the training data is split into clusters for training embedding functions in different subspaces. Consequently, two samples from the same subspace have smaller distances than two from different clusters, which can be interpreted as a proxy to the mining of meaningful relationships. Stochastic class-based hard example mining (Suh et al., 2019) is proposed to mine hard examples effectively.

3.3 Methodology

Our objective is to learn a discriminative function $f$ such that similarity scores of positive pairs are higher than those of negative pairs in the feature space. There exist at least two samples in each class so that all data points can be optimised. Given a query from any class, the objective is to rank its similar data points in front of dissimilar ones.

3.3.1 Pairwise Constraint

Inspired by the former work on pairwise similarity constraint (Hadsell et al., 2006; Wu et al., 2017), we aim to pull positive examples closer than a predefined threshold (boundary). In addition, we intend to separate the positive and negative sets by a margin $m$. To achieve this, we choose the pairwise margin loss (Wu et al., 2017) as our basic pairwise constraint to construct the set-based similarity structure. Specifically, given an image $x_i$, the learning objective is to push its negative points farther than a boundary $\alpha$ and pull its positive ones closer than another boundary $\alpha - m$. Thus $m$ becomes the margin between two boundaries. Mathematically,

$$L_m(x_i, x_j; f) = (1 - y_{ij})[\alpha - d_{ij}]_+ + y_{ij}[d_{ij} - (\alpha - m)]_+, \quad (3.8)$$

where $y_{ij} = 1$ if $y_i = y_j$, and $y_{ij} = 0$ otherwise. $d_{ij} = ||f(x_i) - f(x_j)||_2$ is the Euclidean distance between two points. $[\cdot]_+$ is the hinge function.
3.3.2 Ranked List Loss

In a mini-batch, when \( x_i^c \) is a query, we use it to retrieve the remaining data points, i.e., gallery according to their similarities to the query, which is illustrated in Figure 3.1. In the retrieved list, there are \( N_c - 1 \) positive points in the positive set and \( \sum_{k \neq c} N_k \) points in the negative set. The positive set with respect to the query \( x_i^c \) is denoted as \( P_{c,i} = \{ x_j^c \mid j \neq i \} \), and \( |P_{c,i}| = N_c - 1 \). Similarly, we represent the negative set with respect to \( x_i^c \) as \( N_{c,i} = \{ x_k^i \mid k \neq c \} \), and \( |N_{c,i}| = \sum_{k \neq c} N_k \).

Mining Informative Pairs

Mining informative examples is widely adopted (Cui et al., 2016; Hermans et al., 2017; Oh Song et al., 2016; Schroff et al., 2015; Sohn, 2016; Wang et al., 2019c; Yuan et al., 2017), because it can help to speed convergence and improve generalisation performance if properly designed. By informative examples, we mean non-trivial data points which have non-zero losses, i.e., violating the pairwise constraint with respect to a given query. Because trivial data pairs have zero gradients, including them for training can ‘weaken’ the contribution of non-trivial examples when accumulating the gradients of multiple data points (Hermans et al., 2017). Therefore, we only train on non-trivial positive and negative examples.

Concretely, for the query \( x_i^c \), the non-trivial positive set after mining is represented as \( P_{c,i}^* = \{ x_j^c \mid j \neq i, d_{ij} > (\alpha - m) \} \). Analogously, we denote the negative set after mining as \( N_{c,i}^* = \{ x_k^i \mid k \neq c, d_{ij} < \alpha \} \).

Weighting Negative Pairs

For each query \( x_i^c \), there are a large number of non-trivial negative examples (\( N_{c,i}^* \)) with different magnitude of losses. To make better use of them, we propose to weight the negative examples based on their loss values, i.e., how much each negative pair violates the constraint. Our weighting strategy can be simply represented as:

\[
wi_j = \exp(T_n \times (\alpha - d_{ij})), \quad x_j^i \in N_{c,i}^*, \tag{3.9}
\]

where \( T_n \) is the temperature parameter which controls the degree (slope) of weighting negative examples. If \( T_n = 0 \), we treat all non-trivial negative examples equally. If \( T_n = +\infty \), it becomes the hardest negative example mining because weights are normalised by their sum.
3.3 Methodology

Weighting Positive Pairs

Usually, in the settings of deep metric learning, given a query, there are quite a few matching positives in the search space. However, when multiple positive data points exist, we can also weight those positive ones according to their loss values to make better use of them. Analogous to negative examples weighting, the weighting strategy of positive data pairs can be denoted as follows:

$$w_{ij} = \exp(T_p \times (d_{ij} - (\alpha - m))), \quad x^c_j \in P^*_c,i,$$

(3.10)

where $T_p$ is the temperature parameter which controls the degree (slope) of weighting positive examples. If $T_p = 0$, it treats all non-trivial positive data pairs equally. If $T_p > 0$, positive pairs with larger distances are emphasised. On the contrary, if $T_p < 0$, closer positive pairs are assigned with higher weights, which is widely used to preserve local similarity manifold structure when many positive data points exist (Cui et al., 2016; Huang et al., 2016b; Wang et al., 2019c). The absolute value of $T_p$ determines the differentiation degree over positive data pairs.

Overall Optimisation Objective

In order to pull all non-trivial positive points in $P^*_c,i$ together and learn a class hypersphere, we minimise:

$$L_P(x^c_i; f) = \sum_{x^c_j \in |P^*_c,i|} \frac{w_{ij}}{\sum_{x^c_j \in |P^*_c,i|} w_{ij}} L_m(x^c_i, x^c_j; f).$$

(3.11)

Meanwhile, to push the informative negative points in $N^*_c,i$ beyond the boundary $\alpha$, we minimise:

$$L_N(x^c_i; f) = \sum_{x^k_j \in |N^*_c,i|} \frac{w_{ij}}{\sum_{x^k_j \in |N^*_c,i|} w_{ij}} L_m(x^c_i, x^k_j; f).$$

(3.12)

In RLL, we optimise the two minimisation objectives jointly:

$$L_{RLL}(x^c_i; f) = (1 - \lambda)L_P(x^c_i; f) + \lambda L_N(x^c_i; f),$$

(3.13)

where $\lambda$ controls the balance between positive and negative sets. We treat the two objectives equally and fix $\lambda = 0.5$ without tuning in all our experiments. In this case, the positive and negative sets can contribute equally. Consequently, the sample imbalance problem, i.e., the majority are negative examples, is addressed well.

We remark that our proposed RLL is an extension of traditional point-based triplet loss. Given an anchor, RLL separates the positive and negative sets with a margin between them.
Fig. 3.3: The optimisation objectives with hypersphere regularisation. Different shapes represent different classes. For simplicity, we only show three classes while many classes exist in practice. $\alpha - m$ denotes the diameter of each class hypersphere. Therefore, the distance between any two positive examples is optimised to be no greater than $\alpha - m$. In addition, the distance between any two hypersphere boundaries is no less than $m$.

Note that in the optimisation of every retrieved list, we exploit independent normalisation (Wang et al., 2019c) to address the imbalanced number of positive and negative examples, which is not considered in the previous ranking-motivated losses. In the ranked list of $x^{c,i}$, we regard the features of other examples as constants. Therefore, only $f(x^{c,i})$ is updated based on the influence of weighted combination of other data points, which makes the learning process more stable.

3.3.3 Hypersphere Regularisation by Distance Thresholds

For each query $x^{c,i}$, we propose to make it closer to its positive set $P_{c,i}$ than to its negative set $N_{c,i}$ by a margin $m$. At the same time, we force all negative examples to be farther than a boundary $\alpha$. Consequently, we pull all samples from the same class into a hypersphere. The diameter of each class hypersphere is $\alpha - m$. According to our extensive ablation studies on
both large and small datasets, we find that it is improper to pull positive data pairs as close as possible, which is a common practice in the literature. Intuitively, we should not try to suppress all intraclass variances. By pulling all intraclass examples into one hypersphere instead of one point, better generalisation performance can be obtained. Therefore, we term it hypersphere regularisation. The optimisation objectives with hypersphere regularisation is illustrated in Figure 3.3.

3.3.4 RLL-Simpler

We have introduced the full version of RLL, which includes two distance hyper-parameters $\alpha$ and $m$ for defining the optimisation objectives, and two scaling parameters $T_n$ and $T_p$ for weighting negative and positive data points, respectively. In addition, we propose a simpler version which has only two hyper-parameters, thus being much easier to apply in practice.

Firstly, we can reformulate the optimisation objectives to use only one distance threshold. Following the widely used practice (Law et al., 2017; Movshovitz-Attias et al., 2017; Song et al., 2017; Wang et al., 2017a), we apply an $L_2$ normalisation layer after the final fully connected layer. As a result, the Euclidean distance any two data points ranges from 0 to 2. Recognising a data pair as similar or dissimilar can be treated as a binary classification problem. Therefore, intuitively, we can use the hyperplane Euclidean distance $=1$ as the decision boundary. Following this intuition, we change the setting of $\alpha$ and $m$ in RLL as follows:

- Smallest distance between negative points: $\alpha = 1 + \frac{m}{2}$;
- Largest distance between positive points: $\alpha - m = 1 - \frac{m}{2}$;
- Margin between positive and negative set: $m$.

Namely, in RLL-Simpler, we aim to push the distances of negative pairs to be larger than $1 + \frac{m}{2}$ while pull those of positive ones to be smaller than $1 - \frac{m}{2}$.

Secondly, training RLL on a mini-batch follows the few-shot retrieval setting, i.e., $N_c$-shot $C$-way setting. Generally, $C$ is much larger than $N_c$, e.g., $C = 60$ and $N_c = 3, \forall c$, therefore we have much more negative points than positive ones in the retrieved list. Consequently, by default, we set $T_p = 0$ without weighting positive data points in practice.

In summary, in RLL-Simpler, we have only two active hyper-parameters: $m$ for defining the margin between positive and negative pairs, and $T_n$ for weighting negative examples of a query. It is a simplified version of the full version by setting $T_p = 0$, and $\alpha = 1 + \frac{m}{2}$. With RLL-Simpler, our objective is to demonstrate that our proposal is able to achieve competitive performance without the need to manually optimise many hyper-parameters. In real-world
Example Weighting for Ranked List Loss

![Diagram of ranked list loss](image)

Fig. 3.4: The overall framework of our proposed ranked list loss. In one mini-batch, every image acts as a query iteratively and ranks other images according to the similarity scores. Then in every ranked list, we mine non-trivial data points and weight them based on their distances to the query. Finally, the ranked list loss is computed for every query.

For applications, we recommend first applying RLL-Simpler, and then adopting the full version to further push the performance if needed.

### 3.3.5 Learning Deep Models with RLL

To learn deep models, we implement our RLL based on mini-batches and stochastic gradient descent. Each mini-batch is a randomly sampled subset of the whole training classes, which can be regarded as a mini ranking problem with a much smaller gallery, i.e., searching the matching examples from a smaller number of classes.

A mini-batch is composed of $C$ classes and $N_c$ images per class ($N_c$-shot $C$-way setting). Therefore, each mini-batch can be also treated as a few-shot retrieval task (Triantafillou et al., 2017). Every image $x^i_c$ in the mini-batch acts as the query (anchor) iteratively and the other images serve as the gallery. The RLL on a mini-batch is represented as:

$$L_{RLL}(X; f) = \frac{1}{N} \sum_{c} \sum_{i} L_{RLL}(x^i_c; f),$$

where $N = \sum N_c$ is the batch size. The learning of the deep embedding function $f$ based on RLL is illustrated in Algorithm 1. The overall pipeline is shown in Figure 3.4.

### 3.3.6 Computational Complexity

As illustrated in Algorithm 1, our proposed method does not require the input data to be prepared in any rigid format, e.g., triplets, n-pair tuplets. Instead, it takes random input images with multiclass labels. We conduct online iterative ranking and loss computation (step 2 in Algorithm 1) after obtaining images’ embeddings (step 1 in Algorithm 1). Therefore, the computational complexity of RLL is $O(N^2)$, which is the same as existing ranking-motivated structured loss functions (Movshovitz-Attias et al., 2017; Oh Song et al., 2016; Sohn, 2016).
3.3 Methodology

**Algorithm 1** Ranked List Loss on one mini-batch.

1. **Mini-Batch Settings**: The batch size $N$, the number of classes $C$, the number of images per class $N_c$.
2. **Parameters**: The distance constraint $\alpha$ on negative points, the margin between positive and negative examples $m$, the weighting temperature $T_n, T_p$.
3. **Input**: $X = \{(x_i, y_i)\}_i^N = \{\{x^c_i\}_i^N\}_c^C$, the embedding function $f$, the learning rate $\beta$.
4. **Output**: Updated $f$.
5. **Step 1**: Feedforward all images $\{x_i\}_i^N$ into $f$ to obtain the images’ embeddings $\{f(x_i)\}_i^N$.
6. **Step 2**: Iterative retrieval and loss computation.
   - **foreach** $f(x^c_i) \in \{\{f(x^c_i)\}_i^N\}_c^C$ **do**
     - Mine non-trivial positive set $P^*_{c,i}$.
     - Mine non-trivial negative set $N^*_{c,i}$.
     - Weight negative examples using Eq. (3.9).
     - Weight positive examples using Eq. (3.10).
     - Compute $L_P(x^c_i; f)$ using Eq. (3.11).
     - Compute $L_N(x^c_i; f)$ using Eq. (3.12).
     - Compute $L_{RLL}(x^c_i; f)$ using Eq. (3.13).
   - **end for**
7. Compute $L_{RLL}(X; f)$ using Eq. (3.14).
8. **Step 3**: Gradient computation and back-propagation to update the parameters of $f$:
   - $\nabla_f = \partial L_{RLL}(X; f)/\partial f$;
   - $f = f - \beta \cdot \nabla_f$.

3.3.7 Exploring The Critical Learning Periods of Deep Metric Learning

In biological systems, critical period represents the time windows of early post-natal development during which a temporary stimulus deficit may lead to a skill impairment permanently (Giffin & Mitchell, 1978; Kandel et al., 2013; Konishi, 1985; Mitchell, 1988; Olson & Freeman, 1980; Wiesel & Hubel, 1963). Critical periods do not exist only in biological systems, but also in artificial learning systems (Achille et al., 2019). Recently, a study on the critical learning periods (Hensch, 2004) of deep neural networks is quite interesting and has got much attention (Achille et al., 2019). As a result of fundamental constraints coming from learning dynamics and information processing, Achille et al. (2019) finds that the critical early transient determines the final optimised solution. Specifically, no matter how much additional training is used, a temporary stimulus deficit during the critical early learning period cannot be overcome later.
The early learning phase of artificial deep neural networks is under-scrutinised compared to the network’s behaviours around convergence and the asymptotic properties of the optimisation process. However, it plays a key role. In Achille et al. (2019), Fisher Information and Information Plasticity have been exploited to study the early learning phase. In our work, we validate and study this critical early learning phase in deep metric learning from a novel perspective, i.e., dynamic example weighting. We represent our design choices in this section and display the results in Section 3.5.

Concretely, during training, we revise Eq. (3.9) to its dynamic variant:

\[ T_n = T_1 - \text{cur}\_\text{iter} \times \frac{T_1 - T_2}{\text{max}\_\text{iter}}, \]

\[ w_{ij} = \exp(T_n \times (\alpha - d_{ij})), x_j^i \in N_{c,i}^*, \]

where \( \text{cur}\_\text{iter} < \text{max}\_\text{iter} \), \( \text{max}\_\text{iter} \) denotes the total number of training iterations while \( \text{cur}\_\text{iter} \) is the performed number of iterations till now.

Therefore, instead of fixing the scaling parameter of the weighting scheme, we can study and analyse the effect of dynamic weighting scheme on the optimisation results. Two cases we aim to explore are introduced in detail as follows. Their empirical results are presented and discussed in section 3.5.

- **Study of The Early Learning Phase.** To validate the critical early learning phase in deep metric learning, we fix \( T_2 \) and present the results of different \( T_1 \). In this case, the early learning phase changes along with \( T_1 \). Intuitively, if the early learning phase is critical, then the test performance will be sensitive to \( T_1 \).

- **Exploration of The Later Learning Phase.** On the contrary, we fix \( T_1 \) and change \( T_2 \) to study the effect of the later learning phase on the final optimisation solution. Similarly, if the test performance is sensitive to \( T_2 \), then the later learning phase is crucial, and non-important otherwise.

### 3.4 Experiments

#### 3.4.1 Experimental Details

Some recent papers (Fehervari et al., 2019; Musgrave et al., 2020) have raised the concerns about the fairness of comparing different DML methods. For example, Lifted Struct (Oh Song et al., 2016) reports that the embedding dimension does not play a crucial role. Accordingly, thereafter, the embedding size is different in some papers (Ge et al., 2018; Law et al., 2017;
Experiments

Oh Song et al., 2016; Sohn, 2016; Song et al., 2017). However, it is recently noticed that the embedding size has a huge impact on the performance (Fehervari et al., 2019; Musgrave et al., 2020). In this chapter, to make sure that our comparison is as fair as possible, we present all our implementation details as follows:

Datasets

We conduct experiments on four popular benchmarks: (1) SOP (Oh Song et al., 2016) contains 120,053 images of 22,634 online products sold on eBay.com. 59,551 images of 11,318 categories and 60,502 images of 11,316 categories are used for training and testing respectively. The train/test split and evaluation protocol are the same as (Oh Song et al., 2016). (2) In-shop Clothes (Liu et al., 2016b) contains 7,982 classes and 52,712 images in total. It is split into a training set and a testing set. The training data contains 25,882 images of 3,997 classes. The testing set includes 3,985 classes, and there are 14,218 query images and 12,612 in the gallery for search. (3) CUB-200-2011 (Krause et al., 2013) has 11,788 images of 200 bird species. 5,864 images of the first 100 classes are used for training and 5,924 images of the other 100 classes for testing. (4) CARS196 (Wah et al., 2011) contains 16,185 images of 196 car models. We use the first 98 classes (8,054 images) for training and the remaining 98 classes (8,131 images) for testing. On all datasets, our method is evaluated on the original images, i.e., without using the bounding box information.

Data Augmentation

For fair comparisons, we follow the practice of (Wang et al., 2019a) for data augmentation. Concretely, we (1) warp the original image to a size of $256 \times 256$; (2) crop a random size (default: 0.5 to 1.0 of the resized image)$^2$ with a random aspect ratio (default: 3/4 to 4/3); (3) resize the crop to $227 \times 227$ and horizontally flip it with a probability of 0.5. At testing stage, we only use a single centre crop without mirroring. We remark that the performance on large datasets (i.e., SOP and In-shop Clothes) changes slightly when the data augmentation is different. While small datasets CUB-200-2011 and CARS-196 are very sensitive to small changes.

Backbone and Initialisation Details

We use GoogLeNet V2 (Ioffe & Szegedy, 2015) as our backbone network, and compare with prior results using the same backbone as well. Additionally, in the original net, there are three

---

$^2$We change the random crop size from the range [0.16, 1.0] used in (Wang et al., 2019a) to [0.5, 1] as we do not want a crop containing no object of interest at all.
fully connected layers of different depth. We refer them based on their relative locations as follows: L for the low-level layer (inception-3c/output), M for the mid-level layer (inception-4e/output) and H for the high-level layer (inception-5b/output). By exploiting them, we study the representations of different depth, which is valuable for deploying a model in practice. Following (Law et al., 2017; Movshovitz-Attias et al., 2017; Song et al., 2017), the pretrained model on ImageNet (Russakovsky et al., 2015) is used for initialisation in our experiments. Three original 1000-neuron fully connected layers followed by the softmax layer and cross-entropy loss are changed to three new fully connected layers followed by an \( L_2 \) normalisation layer and our proposed ranked list loss. The new layers are randomly initialised and optimised with 10 times larger learning rate than the others for faster convergence.

It is well accepted in (Musgrave et al., 2020; Qian et al., 2019; Wang et al., 2019a) that CUB-200-2011 and CARS-196 are sensitive to small changes of training details. Therefore, following the concurrent work Multi-Simi (Wang et al., 2019a), and the following work SoftTriple (Qian et al., 2019) and Reality Check (Musgrave et al., 2020), we freeze the BatchNorm layers during training to fairly compare with their reported results on CUB-200-2011 and CARS-196.

**Embedding and The Embedding Size**

We use RLL-H to denote the single high-level embedding. RLL-M and RLL-L are analogous. For an exactly fair comparison with other baselines, we look at the results of RLL-H. Generally, there are two ways to increase the embedding size: (1) increasing the size of a single embedding; (2) concatenating multiple embeddings. Empirically, we find that our method is less sensitive to the increasing size of a single embedding, while it performs much better when concatenating multiple embeddings.

On all datasets, we follow their corresponding most common setting and set the embedding size accordingly to fairly compare with as many existing approaches as possible. Finally, we study the impact of the embedding size on the largest dataset SOP in the section 3.4.8.

**Performance Metric**

Following (Oh Song et al., 2016), we report the image retrieval performance measured by Recall@\( K \). We do not report the image clustering quality NMI (Schütze et al., 2008), because NMI is not a proper metric for fine-grained object recognition with tremendous classes and only several images per class. A similar idea is presented in detail in (Musgrave et al., 2020).

Indeed, all reported results are validation performance. Theoretically, the validation data follows the same distribution as the test data, and the test data is always unknown before the
deployment for practical use. Therefore, it is reasonable to report the validation performance as long as the validation data is only used for model selection, without being used for training.

**Training and Optimisation Settings.**

The standard stochastic gradient descent (SGD) optimiser is used with a momentum of 0.9, a weight decay rate of $2e^{-5}$. We set the base learning rate to $1e^{-2}$. For learning stability, we train another 10 epochs after convergence. In each mini-batch, we randomly sample $C$ classes and $K$ images per class. We use a single Tesla V100 to train the large dataset SOP, while a single GTX 1080 Ti to train other datasets. To leverage the computational resources, we set $C = 60, K = 3$ on SOP while $C = 22, K = 3$ on the other three datasets. Thus $N_c = K = 3, \forall c$, the batch size $N = 180$ on SOP while $N = 66$ on others.

We do not manually optimise the hyper-parameters extensively, i.e., we use ‘RLL-Simpler’ to compare with existing methods, so that we only need to optimise $m$ and $T_n$. In addition, we simply fix $T_n$ throughout the training process. We only explore the dynamic weighting scheme when studying the critical learning periods in DML.

In the ablation study, we use the full version of RLL for more comprehensive study. Moreover, in those experiments, we use the simpler data augmentation for faster convergence since our focus is analysing the key components, instead of comparing with existing methods. Specifically, we crop a random size of $227 \times 227$ from the resized image ($256 \times 256$), and horizontally flip the crop with a probability of 0.5.

For the ease of reproducing our results, Our source code and the training scripts of all datasets are publicly available online: https://github.com/XinshaoAmosWang/Ranked-List-Loss-for-DML.

### 3.4.2 Comparison with Recent Baselines

**Competitors.** We compared our method with the following methods which are implemented and tested under the same settings: Triplet Semihard, Lifted Struct, N-pair-mc, Struct Clust, Spectral Clust, and Proxy NCA\textsuperscript{3}. These methods have been described in Section 3.2 except for Triplet Semihard (Schroff et al., 2015) which mines semihard negative examples to improve the conventional triplet loss and is reimplemented in Struct Clust (Song et al., 2017)

\textsuperscript{3} Some methods (Duan et al., 2018; Harwood et al., 2017; Lin et al., 2018; Suh et al., 2019; Ustinova & Lempitsky, 2016; Wang et al., 2017b) use GoogLeNet V1 (Szegedy et al., 2015). Some others, e.g., the margin loss (Wu et al., 2017), FastAP (Cakir et al., 2019), MIC (Roth et al., 2019) and Divide & Conquer (Sanakoyeu et al., 2019), apply ResNet50 (He et al., 2016). For a fair comparison, they are not compared in the table. Additionally, we do not compare with ensemble models (Kim et al., 2018; Opitz et al., 2017; Xuan et al., 2018; Yuan et al., 2017).
Table 3.1: Comparison with the state-of-the-art methods on SOP. The ‘–’ denotes the corresponding results are not reported in the original paper. For a fair comparison, all reported results have an embedding size of 64, which is the common setting and more challenging than a larger embedding size.

<table>
<thead>
<tr>
<th>Method</th>
<th>Dim</th>
<th>R@1</th>
<th>R@10</th>
<th>R@100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triplet Semihard (Schroff et al., 2015)</td>
<td>64</td>
<td>66.7</td>
<td>82.4</td>
<td>91.9</td>
</tr>
<tr>
<td>Lifted Struct (Oh Song et al., 2016)</td>
<td>64</td>
<td>62.5</td>
<td>80.8</td>
<td>91.9</td>
</tr>
<tr>
<td>N-pair-mc (Sohn, 2016)</td>
<td>64</td>
<td>66.4</td>
<td>83.2</td>
<td>93.0</td>
</tr>
<tr>
<td>Struct Clust (Song et al., 2017)</td>
<td>64</td>
<td>67.0</td>
<td>83.7</td>
<td>93.2</td>
</tr>
<tr>
<td>Proxy NCA (Movshovitz-Attias et al., 2017)</td>
<td>64</td>
<td>73.7</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Multi-Simi (Wang et al., 2019a)</td>
<td>64</td>
<td>74.1</td>
<td>87.8</td>
<td>94.7</td>
</tr>
<tr>
<td>SoftMaxnorm (Qian et al., 2019)</td>
<td>64</td>
<td>75.9</td>
<td>88.8</td>
<td>95.2</td>
</tr>
<tr>
<td>SoftTriple (Qian et al., 2019)</td>
<td>64</td>
<td>76.3</td>
<td>89.1</td>
<td>95.3</td>
</tr>
<tr>
<td>RLL-Simpler-H</td>
<td>64</td>
<td>74.8</td>
<td>88.4</td>
<td>95.1</td>
</tr>
</tbody>
</table>

*: Although the results of SoftMaxnorm and SoftTriple are better, they are theoretically non-scalable to extremely large datasets because they use multiple proxies to represent one class.

Table 3.2: Comparison with the state-of-the-art methods on In-shop Clothes dataset. For a fair comparison, we compare with the reported results with an embedding size of 128, which is the common setting on In-shop Clothes.

<table>
<thead>
<tr>
<th>Method</th>
<th>Dim</th>
<th>R@1</th>
<th>R@10</th>
<th>R@20</th>
<th>R@30</th>
<th>R@40</th>
<th>R@50</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTL (Ge et al., 2018)</td>
<td>128</td>
<td>80.9</td>
<td>94.3</td>
<td>95.8</td>
<td>97.2</td>
<td>97.4</td>
<td>97.8</td>
</tr>
<tr>
<td>Multi-Simi (Wang et al., 2019a)</td>
<td>128</td>
<td>88.0</td>
<td>97.2</td>
<td>98.1</td>
<td>98.5</td>
<td>98.7</td>
<td>98.8</td>
</tr>
<tr>
<td>RLL-Simpler-H</td>
<td>128</td>
<td>88.5</td>
<td>97.1</td>
<td>98.0</td>
<td>98.4</td>
<td>98.6</td>
<td>98.7</td>
</tr>
</tbody>
</table>

with GoogLeNet V2. HTL (Ge et al., 2018) exploits the hierarchical similarity structure among different classes and merges similar classes recursively, i.e., building the global class-level hierarchical tree by using all original classes as leaves and updates the tree after every epoch. HTL (Ge et al., 2018) is computationally expensive and unscaleable to extremely large datasets. Additionally, SoftMaxnorm and SoftTriple (Qian et al., 2019) are theoretically non-scalable to extremely large dataset because they use multiple proxies to represent one class. Finally, XBM (Wang et al., 2020) exploits extra information across mini-batch tasks, so that it is not benchmarked.
Table 3.3: Comparison with the state-of-the-art methods on CARS196 in terms of Recall@K (%). For a comparison, all the reported results have an embedding size of 64.

<table>
<thead>
<tr>
<th>Method</th>
<th>Dim.</th>
<th>R@1</th>
<th>R@2</th>
<th>R@4</th>
<th>R@8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triplet Semihard (Schroff et al., 2015)</td>
<td>64</td>
<td>51.5</td>
<td>63.8</td>
<td>73.5</td>
<td>82.4</td>
</tr>
<tr>
<td>Lifted Struct (Oh Song et al., 2016)</td>
<td>64</td>
<td>53.0</td>
<td>65.7</td>
<td>76.0</td>
<td>84.3</td>
</tr>
<tr>
<td>N-pair-mc (Sohn, 2016)</td>
<td>64</td>
<td>53.9</td>
<td>66.8</td>
<td>77.8</td>
<td>86.4</td>
</tr>
<tr>
<td>Struct Clust (Song et al., 2017)</td>
<td>64</td>
<td>58.1</td>
<td>70.6</td>
<td>80.3</td>
<td>87.8</td>
</tr>
<tr>
<td>Proxy NCA (Movshovitz-Attias et al., 2017)</td>
<td>64</td>
<td>73.2</td>
<td>82.4</td>
<td>86.4</td>
<td>88.7</td>
</tr>
<tr>
<td>Multi-Simi (Wang et al., 2019a)</td>
<td>64</td>
<td>77.3</td>
<td>85.3</td>
<td>90.5</td>
<td>94.2</td>
</tr>
<tr>
<td>SoftMax$_{norm}$ (Qian et al., 2019)</td>
<td>64</td>
<td>76.8</td>
<td>85.6</td>
<td>91.3</td>
<td>95.2</td>
</tr>
<tr>
<td>SoftTriple (Qian et al., 2019)</td>
<td>64</td>
<td>78.6</td>
<td>86.6</td>
<td>91.8</td>
<td>95.4</td>
</tr>
<tr>
<td>RLL-Simpler-H</td>
<td>64</td>
<td>73.7</td>
<td>82.4</td>
<td>88.8</td>
<td>93.3</td>
</tr>
</tbody>
</table>

**Result analysis.** The comparisons between our method and existing competitors on four datasets are presented in Tables 3.1, 3.2, 3.3, and 3.4, respectively. For a fair comparison, on every dataset, we report the results of RLL-Simpler-H whose network architecture and embedding size are exactly the same as other reported baselines.

From the tables, we have the following observations:

- On the two larger datasets, RLL-Simpler-H shows the state-of-the-art performance, as shown in Tables 3.1 and 3.2.

- On the two smaller datasets, except for the theoretically non-scalable SoftMax$_{norm}$ and SoftTriple, only Multi-Simi is better than ours. However, technically, the weighting scheme of Multi-Simi considers multiple metrics, thus being more complex than ours.

We remark: (1) in the general settings of metric learning, the training classes are disjoint with testing classes, which means the training set can be infinitely large. Therefore, larger datasets, e.g, SOP and In-shop Clothes, are better to test on; (2) CARS196 and CUB-200-2011 are significantly sensitive to the minor changes of training details as mentioned in (Musgrave et al., 2020; Qian et al., 2019; Wang et al., 2019a).

### 3.4.3 On the network depth of an embedding function

In this subsection, we study the network depth of an embedding function and the concatenation of embedding functions. The results on SOP and In-shop Clothes are displayed in Tables 3.5 and 3.6, respectively. RLL-Simpler-(L,M,H) denotes the multilevel embedding
Table 3.4: Comparison with the state-of-the-art methods on CUB-200-2011 in terms of Recall@K (%). All the displayed results have an embedding size of 64 for a fair comparison.

<table>
<thead>
<tr>
<th>Method</th>
<th>Dim.</th>
<th>R@1</th>
<th>R@2</th>
<th>R@4</th>
<th>R@8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triplet Semihard (Schroff et al., 2015)</td>
<td>64</td>
<td>42.6</td>
<td>55.0</td>
<td>66.4</td>
<td>77.2</td>
</tr>
<tr>
<td>Lifted Struct (Oh Song et al., 2016)</td>
<td>64</td>
<td>43.6</td>
<td>56.6</td>
<td>68.6</td>
<td>79.6</td>
</tr>
<tr>
<td>N-pair-mc (Sohn, 2016)</td>
<td>64</td>
<td>45.4</td>
<td>58.4</td>
<td>69.5</td>
<td>79.5</td>
</tr>
<tr>
<td>Struct Clust (Song et al., 2017)</td>
<td>64</td>
<td>48.2</td>
<td>61.4</td>
<td>71.8</td>
<td>81.9</td>
</tr>
<tr>
<td>Proxy NCA (Movshovitz-Attias et al., 2017)</td>
<td>64</td>
<td>49.2</td>
<td>61.9</td>
<td>67.9</td>
<td>72.4</td>
</tr>
<tr>
<td>Multi-Simi (Wang et al., 2019a)</td>
<td>64</td>
<td>57.4</td>
<td>69.8</td>
<td>80.0</td>
<td>87.8</td>
</tr>
<tr>
<td>SoftMaxnorm (Qian et al., 2019)</td>
<td>64</td>
<td>57.8</td>
<td>70.0</td>
<td>80.1</td>
<td>87.9</td>
</tr>
<tr>
<td>SoftTriple (Qian et al., 2019)</td>
<td>64</td>
<td>60.1</td>
<td>71.9</td>
<td>81.2</td>
<td>88.5</td>
</tr>
<tr>
<td>RLL-Simpler-H</td>
<td>64</td>
<td>56.7</td>
<td>68.1</td>
<td>77.7</td>
<td>85.8</td>
</tr>
</tbody>
</table>

by concatenating the low-level, mid-level and high-level embeddings. Others are analogous. We study and report them because from the practical perspective, RLL-Simpler-L and RLL-Simpler-M are of smaller network depth, while perform similarly or even better than RLL-Simpler-H. Firstly, this indicates that very deep networks are not necessarily better in practice. This information is of great value in applications where smaller networks with faster computational speed are preferred. Secondly, this motivates us to exploit multi-layer embeddings well. All of them are discriminative, and exploit non-identical knowledge to measure the distance between data points since they are of different depth. We obtain promising performance by simply concatenating them. Therefore, there is an open future research lead to better exploit multi-layer embeddings in deep metric learning, without the need to train multiple learners (Kim et al., 2018; Opitz et al., 2017; Xuan et al., 2018; Yuan et al., 2017).

### 3.4.4 Mining Non-trivial Pairs

As presented in Section 3.3.2, for each query, RLL mines examples which violate the pairwise constraint with respect to the query. Specifically, we mine negative examples whose distance is smaller than $\alpha$ in Eq. (3.12). Simultaneously, we mine positive examples whose distance is larger than $\alpha - m$ in Eq. (3.11). As a result, a margin $m$ is established between negative and positive examples in each ranked list. What examples are mined is determined by $\alpha$ and $m$. We conduct experiments on the large dataset SOP to analyse the influence of $\alpha$. Regarding $m$, we study it in section 3.4.7.
3.4 Experiments

Table 3.5: Exploration on the network depth of an embedding function and the concatenation of embedding functions on SOP. Additionally, the bottom block shows the recently reported results using a higher embedding size, except for the results in Table 3.1.

<table>
<thead>
<tr>
<th>SOP</th>
<th>Dim</th>
<th>R@1</th>
<th>R@10</th>
<th>R@100</th>
</tr>
</thead>
<tbody>
<tr>
<td>RLL-Simpler-L</td>
<td>64</td>
<td>74.9</td>
<td>87.9</td>
<td>94.5</td>
</tr>
<tr>
<td>RLL-Simpler-M</td>
<td>64</td>
<td>75.7</td>
<td>88.7</td>
<td>95.1</td>
</tr>
<tr>
<td>RLL-Simpler-H</td>
<td>64</td>
<td>74.8</td>
<td>88.4</td>
<td>95.1</td>
</tr>
<tr>
<td>RLL-Simpler-(L,M)</td>
<td>128 = 64×2</td>
<td>78.3</td>
<td>90.2</td>
<td>95.8</td>
</tr>
<tr>
<td>RLL-Simpler-(L,H)</td>
<td>128 = 64×2</td>
<td>78.4</td>
<td>90.1</td>
<td>95.8</td>
</tr>
<tr>
<td>RLL-Simpler-(M,H)</td>
<td>128 = 64×2</td>
<td>76.9</td>
<td>89.5</td>
<td>95.5</td>
</tr>
<tr>
<td>RLL-Simpler-(L,M,H)</td>
<td>192 = 64×3</td>
<td>78.6</td>
<td>90.4</td>
<td>95.9</td>
</tr>
<tr>
<td>Multi-Simi (Wang et al., 2019a)</td>
<td>128</td>
<td>76.6</td>
<td>89.2</td>
<td>95.2</td>
</tr>
<tr>
<td>Multi-Simi (Wang et al., 2019a)</td>
<td>512</td>
<td>78.2</td>
<td>90.5</td>
<td>96.0</td>
</tr>
<tr>
<td>Spectral Clust (Law et al., 2017)</td>
<td>512</td>
<td>67.6</td>
<td>83.7</td>
<td>93.3</td>
</tr>
<tr>
<td>HTL (Ge et al., 2018)</td>
<td>512</td>
<td>74.8</td>
<td>88.3</td>
<td>94.8</td>
</tr>
<tr>
<td>SoftMax\textsubscript{norm} (Qian et al., 2019)</td>
<td>512</td>
<td>78.0</td>
<td>90.2</td>
<td>96.0</td>
</tr>
<tr>
<td>SoftTriple (Qian et al., 2019)</td>
<td>512</td>
<td>78.3</td>
<td>90.3</td>
<td>95.9</td>
</tr>
<tr>
<td>Circle Loss (Sun et al., 2020)</td>
<td>512</td>
<td>78.3</td>
<td>90.5</td>
<td>96.1</td>
</tr>
</tbody>
</table>

Table 3.6: Exploration on the network depth of an embedding function and the concatenation of embedding functions on In-shop Clothes dataset. In addition, the recently reported results using a higher embedding size are displayed in the bottom block, except for the results in Table 3.2.

<table>
<thead>
<tr>
<th>In-shop Clothes</th>
<th>Dim</th>
<th>R@1</th>
<th>R@10</th>
<th>R@20</th>
<th>R@30</th>
<th>R@40</th>
<th>R@50</th>
</tr>
</thead>
<tbody>
<tr>
<td>RLL-Simpler-L</td>
<td>128</td>
<td>85.2</td>
<td>96.3</td>
<td>97.5</td>
<td>98.0</td>
<td>98.3</td>
<td>98.5</td>
</tr>
<tr>
<td>RLL-Simpler-M</td>
<td>128</td>
<td>88.7</td>
<td>97.2</td>
<td>98.1</td>
<td>98.4</td>
<td>98.7</td>
<td>98.9</td>
</tr>
<tr>
<td>RLL-Simpler-H</td>
<td>128</td>
<td>88.5</td>
<td>97.1</td>
<td>98.0</td>
<td>98.4</td>
<td>98.6</td>
<td>98.7</td>
</tr>
<tr>
<td>RLL-Simpler-(L,M)</td>
<td>128×2</td>
<td>89.3</td>
<td>97.5</td>
<td>98.2</td>
<td>98.6</td>
<td>98.8</td>
<td>99.0</td>
</tr>
<tr>
<td>RLL-Simpler-(L,H)</td>
<td>128×2</td>
<td>89.4</td>
<td>97.5</td>
<td>98.3</td>
<td>98.6</td>
<td>98.8</td>
<td>98.9</td>
</tr>
<tr>
<td>RLL-Simpler-(M,H)</td>
<td>128×2</td>
<td>89.4</td>
<td>97.4</td>
<td>98.2</td>
<td>98.5</td>
<td>98.7</td>
<td>98.9</td>
</tr>
<tr>
<td>RLL-Simpler-(L,M,H)</td>
<td>128×3</td>
<td>89.9</td>
<td>97.6</td>
<td>98.3</td>
<td>98.7</td>
<td>98.9</td>
<td>99.0</td>
</tr>
<tr>
<td>FashionNet (Liu et al., 2016b)</td>
<td>4096</td>
<td>53.0</td>
<td>73.0</td>
<td>76.0</td>
<td>77.0</td>
<td>79.0</td>
<td>80.0</td>
</tr>
<tr>
<td>Multi-Simi (Wang et al., 2019a)</td>
<td>512</td>
<td>89.7</td>
<td>97.9</td>
<td>98.5</td>
<td>98.8</td>
<td>99.1</td>
<td>99.2</td>
</tr>
</tbody>
</table>

**Impact of α.** To study the impact of α, we set the temperature $T_n = 10$ and the margin $m = 0.4$ in all experiments. The results are presented in Table 3.7. We observe that a proper
Table 3.7: The impact of $\alpha$ on the SOP. In all experiments, $m = 0.4, T_n = 10$.

<table>
<thead>
<tr>
<th>$m = 0.4, T_n = 10$</th>
<th>R@1</th>
<th>R@10</th>
<th>R@100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha = 1.4$</td>
<td>76.2</td>
<td>89.4</td>
<td>95.6</td>
</tr>
<tr>
<td>$\alpha = 1.2$</td>
<td>79.8</td>
<td>91.3</td>
<td>96.3</td>
</tr>
<tr>
<td>$\alpha = 1.0$</td>
<td>78.7</td>
<td>90.5</td>
<td>95.9</td>
</tr>
</tbody>
</table>

Table 3.8: Results of different $T_n$ on the SOP in terms of Recall@$K$ (%). We fix $m = 0.4, \alpha = 1.2$ in all experiments.

<table>
<thead>
<tr>
<th>$m = 0.4, \alpha = 1.2$</th>
<th>R@1</th>
<th>R@10</th>
<th>R@100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_n = 0$</td>
<td>78.8</td>
<td>90.7</td>
<td>96.1</td>
</tr>
<tr>
<td>$T_n = 5$</td>
<td>79.1</td>
<td>91.0</td>
<td>96.2</td>
</tr>
<tr>
<td>$T_n = 10$</td>
<td>79.8</td>
<td>91.3</td>
<td>96.3</td>
</tr>
<tr>
<td>$T_n = 15$</td>
<td>79.3</td>
<td>90.9</td>
<td>96.0</td>
</tr>
<tr>
<td>$T_n = 20$</td>
<td>78.6</td>
<td>90.5</td>
<td>95.7</td>
</tr>
</tbody>
</table>

negative constraint $\alpha$ is important for RLL to learn discriminative embeddings. This is consistent with our intuition as $\alpha$ controls how much the negative examples are pushed away.

### 3.4.5 Weighting Negative Pairs

In this section, we conduct experiments to evaluate the influence of $T_n$ for weighting negative examples in Eq. (3.9). We fix $m = 0.4$ and $\alpha = 1.2$ in all experiments. The temperature parameter $T_n$ ($T_n > 0$) controls the slope of weighting. The results are presented in Table 3.8. We observe that:

- When $T_n = 0$, RLL treats all non-trivial negative examples equally, i.e., no weighting is applied. The Recall@1 result is 78.8%, which is only 1% lower than the best one using proper weighting. This demonstrates the superiority of RLL even without weighting.

- The performance changes slightly as $T_n$ does. The performance gap is around 1% when $T_n$ ranges from 0 to 20. In addition, the performance drops when $T_n$ is too large. This may be because ‘very’ hard examples exist in the training data (e.g., outliers) (Cui et al., 2016; Schroff et al., 2015; Wang et al., 2019c).

### 3.4.6 Weighting Positive Pairs

In most cases where only a few positive data points exist, we only weight negative examples. For example, when $K = 3$, given a query, there are only two positives for every query so that
3.4 Experiments

Table 3.9: Results of weighting positive examples on the SOP. We fix other parameters, i.e., \( m = 0.4, \alpha = 1.2 \) and \( T_n = 10 \).

<table>
<thead>
<tr>
<th>Batch Size, Batch Content</th>
<th>Weighting Positive Examples</th>
<th>Recall@( K )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( T_p = 10 )</td>
<td>( \begin{array}{lll} 77.3 &amp; 90.1 &amp; 96.0 \end{array} )</td>
</tr>
<tr>
<td></td>
<td>( T_p = 5 )</td>
<td>( \begin{array}{lll} 78.5 &amp; 90.7 &amp; 96.2 \end{array} )</td>
</tr>
<tr>
<td>( C = 30, K = 6, ) ( N = 180 = 30 \times 6 )</td>
<td>( T_p = 0 )</td>
<td>( \begin{array}{lll} 79.0 &amp; 90.9 &amp; 96.2 \end{array} )</td>
</tr>
<tr>
<td></td>
<td>( T_p = -5 )</td>
<td>( \begin{array}{lll} 78.8 &amp; 90.7 &amp; 95.9 \end{array} )</td>
</tr>
<tr>
<td></td>
<td>( T_p = -10 )</td>
<td>( \begin{array}{lll} 78.8 &amp; 90.4 &amp; 95.8 \end{array} )</td>
</tr>
<tr>
<td></td>
<td>( T_p = 10 )</td>
<td>( \begin{array}{lll} 75.5 &amp; 88.9 &amp; 95.6 \end{array} )</td>
</tr>
<tr>
<td>( C = 15, K = 12, ) ( N = 180 = 15 \times 12 )</td>
<td>( T_p = 5 )</td>
<td>( \begin{array}{lll} 76.5 &amp; 89.4 &amp; 95.8 \end{array} )</td>
</tr>
<tr>
<td></td>
<td>( T_p = 0 )</td>
<td>( \begin{array}{lll} 76.9 &amp; 89.5 &amp; 95.6 \end{array} )</td>
</tr>
<tr>
<td></td>
<td>( T_p = -5 )</td>
<td>( \begin{array}{lll} 76.9 &amp; 89.2 &amp; 95.3 \end{array} )</td>
</tr>
<tr>
<td></td>
<td>( T_p = -10 )</td>
<td>( \begin{array}{lll} 76.5 &amp; 89.0 &amp; 95.1 \end{array} )</td>
</tr>
</tbody>
</table>

we treat them equally. We have tried differentiating them but the performance difference is negligible. However, when multiple positives exist, it is quite natural to ask whether weighting positive examples helps. Therefore, following (Wang et al., 2019c), we empirically study weighting positive examples when there are many positive data points in a retrieved list.

While fixing the batch size to be 180, we study two different cases: \( K = 6 \) and \( K = 12 \). Given a query, there are 5 and 11 positive instances out of 179 in its retrieved list, respectively. In each case, we choose 5 different \( T_p \): 10, 5, 0, -5, -10. We remark that: 1) \( T_p > 0 \) denotes harder positives with a larger distance are emphasised while \( T_p < 0 \) represents easier positive pairs with a smaller distance are focused; 2) When the absolute value of \( T_p \) is larger, e.g., 10 and -10, the relative weight between two instances is larger. As a result, the differentiation becomes more significant. The results are shown in Table 3.9 and we discuss them as follows:

- When emphasising on easier positive examples (\( T_p < 0 \)) or without weighting (\( T_p = 0 \)), the performance is similar.
- When focusing on harder positive examples, the performance decreases as \( T_p \) increases.
- The results are generally consistent with (Cui et al., 2016; Wang et al., 2019c). They find that emphasising on harder positive examples cannot preserve the intraclass similarity manifold structure, thus reducing the generalisation performance.
Table 3.10: The impact of hypersphere diameter for preserving the intraclass variance on the SOP. We set $\alpha = 1.2, T_n = 10$ in all experiments.

<table>
<thead>
<tr>
<th>Batch Size, Batch Content</th>
<th>Distance Margin m</th>
<th>Hypersphere Diameter $\alpha - m$</th>
<th>Recall@K</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$m = 0$</td>
<td>1.2</td>
<td>76.1</td>
</tr>
<tr>
<td>$180 = 60 \times 3$</td>
<td></td>
<td></td>
<td>89.8</td>
</tr>
<tr>
<td></td>
<td>$m = 0.2$</td>
<td>1.0</td>
<td>79.0</td>
</tr>
<tr>
<td></td>
<td>$m = 0.4$</td>
<td>0.8</td>
<td>79.8</td>
</tr>
<tr>
<td></td>
<td>$m = 0.6$</td>
<td>0.6</td>
<td>79.2</td>
</tr>
<tr>
<td></td>
<td>$m = 0.8$</td>
<td>0.4</td>
<td>79.0</td>
</tr>
<tr>
<td></td>
<td>$m = 1.0$</td>
<td>0.2</td>
<td>79.1</td>
</tr>
<tr>
<td></td>
<td>$m = 1.2$</td>
<td>0.0</td>
<td>79.1</td>
</tr>
<tr>
<td>$180 = 30 \times 6$</td>
<td></td>
<td></td>
<td>90.5</td>
</tr>
<tr>
<td></td>
<td>$m = 0$</td>
<td>1.2</td>
<td>78.7</td>
</tr>
<tr>
<td></td>
<td>$m = 0.2$</td>
<td>1.0</td>
<td>79.1</td>
</tr>
<tr>
<td></td>
<td>$m = 0.4$</td>
<td>0.8</td>
<td>79.0</td>
</tr>
<tr>
<td></td>
<td>$m = 0.6$</td>
<td>0.6</td>
<td>78.3</td>
</tr>
<tr>
<td></td>
<td>$m = 0.8$</td>
<td>0.4</td>
<td>77.8</td>
</tr>
<tr>
<td></td>
<td>$m = 1.0$</td>
<td>0.2</td>
<td>76.2</td>
</tr>
<tr>
<td></td>
<td>$m = 1.2$</td>
<td>0.0</td>
<td>76.4</td>
</tr>
</tbody>
</table>

3.4.7 Hypersphere Regularisation

The hypersphere diameter is an indicator of the intraclass variance. To study the impact of it, we fix $\alpha = 1.2$ and $T_n = 10$ while changing $m$. For more comprehensive exploration, we try two different settings: $C = 60, K = 3$, and $C = 30, K = 6$. We do experiments on the SOP dataset and present the results in Table 3.10. We have two important observations:

- When $m = \alpha = 1.2$, the diameter is 0, which means positive pairs are pulled as close as possible and has the same effect as the conventional contrastive loss. In this case, the Recall@1 is considerably worse than the best. Especially, when $C = 30, K = 6$, without hypersphere regularisation, the Recall@1 is only 76.4% while the best Recall@1 with strong regularisation is 79.1%.

- In both settings, we obtain almost the best performance when the hypersphere diameter is 0.8. In case where $C = 30, K = 6$, the Recall@K results are more sensitive to the hypersphere diameter.

3.4.8 Ablation Study on Other General Factors

In this subsection, we present our empirical study on other method-independent factors of deep metric learning: 1) Batch size; 2) Embedding size; 3) Batch content.
3.4 Experiments

Table 3.11: The results of different batch size on the SOP.

<table>
<thead>
<tr>
<th>Batch size</th>
<th>R@1</th>
<th>R@10</th>
<th>R@100</th>
</tr>
</thead>
<tbody>
<tr>
<td>120 = 40 × 3</td>
<td>79.2</td>
<td>90.9</td>
<td>96.2</td>
</tr>
<tr>
<td>150 = 50 × 3</td>
<td>79.5</td>
<td>91.1</td>
<td>96.2</td>
</tr>
<tr>
<td>165 = 55 × 3</td>
<td>79.7</td>
<td>91.2</td>
<td>96.3</td>
</tr>
<tr>
<td>180 = 60 × 3</td>
<td>79.8</td>
<td>91.3</td>
<td>96.3</td>
</tr>
<tr>
<td>195 = 65 × 3</td>
<td>79.8</td>
<td>91.3</td>
<td>96.3</td>
</tr>
</tbody>
</table>

The Impact of Batch Size

The batch size is usually important in deep metric learning. During training, we follow the few-shot retrieval setting, so that the batch size determines the scale of a problem we are going to solve every iteration. We conduct experiments on the SOP to evaluate the influence of batch size in our approach. Specifically, we fix the number of images per class (\( \forall c, N_c = K = 3 \)) and only change the number of classes \( C \in \{40, 50, 55, 60, 65\} \) in each mini-batch. The results are reported in Table 3.11. We can see that RLL is not very sensitive to the batch size.

The Impact of Embedding Size

Table 3.12: The results of RLL-(L,M,H) with different embedding size on the SOP test dataset.

<table>
<thead>
<tr>
<th>Embedding size</th>
<th>R@1</th>
<th>R@10</th>
<th>R@100</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 × 3</td>
<td>61.8</td>
<td>80.8</td>
<td>91.9</td>
</tr>
<tr>
<td>32 × 3</td>
<td>73.8</td>
<td>88.0</td>
<td>94.9</td>
</tr>
<tr>
<td>64 × 3</td>
<td>77.9</td>
<td>90.3</td>
<td>95.8</td>
</tr>
<tr>
<td>128 × 3</td>
<td>79.2</td>
<td>91.0</td>
<td>96.2</td>
</tr>
<tr>
<td>256 × 3</td>
<td>79.5</td>
<td>91.1</td>
<td>96.2</td>
</tr>
<tr>
<td>512 × 3</td>
<td>79.8</td>
<td>91.3</td>
<td>96.3</td>
</tr>
<tr>
<td>1024 × 3</td>
<td>79.9</td>
<td>91.4</td>
<td>96.4</td>
</tr>
<tr>
<td>1536 × 3</td>
<td>80.2</td>
<td>91.4</td>
<td>96.4</td>
</tr>
</tbody>
</table>

The feature dimension is another considerable factor when learning deep representations for downstream tasks. Generally, the objective is to encode an input into a low-dimensional feature vector so that the storage and computational complexity can be reduced on downstream tasks, e.g., fast image retrieval (Yu et al., 2018; Zhai & Wu, 2018). Therefore, in this subsection, we study the influence of embedding size in our RLL. In all experiments, we
Table 3.13: The results of different batch content $C \times K$ on the SOP. We fix $N = C \times K = 180$ and change $C, K$.

<table>
<thead>
<tr>
<th>Batch content ($N = C \times K$)</th>
<th>R@1</th>
<th>R@10</th>
<th>R@100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N = 180 = 90 \times 2$</td>
<td>79.8</td>
<td>91.3</td>
<td>96.2</td>
</tr>
<tr>
<td>$N = 180 = 60 \times 3$</td>
<td>79.8</td>
<td>91.3</td>
<td>96.3</td>
</tr>
<tr>
<td>$N = 180 = 60 \times 4$</td>
<td>79.4</td>
<td>91.1</td>
<td>96.2</td>
</tr>
<tr>
<td>$N = 180 = 36 \times 5$</td>
<td>79.0</td>
<td>90.8</td>
<td>96.1</td>
</tr>
<tr>
<td>$N = 180 = 30 \times 6$</td>
<td>79.0</td>
<td>90.9</td>
<td>96.2</td>
</tr>
</tbody>
</table>

set $C = 60, K = 3, \alpha = 1.2, m = 0.4, T_n = 10$. The results on the SOP are displayed in the Table 3.12. We can see that generally a larger embedding size leads to a better performance. Finally, the performance increase becomes negligible. Therefore, due to the limited storage and faster computational speed requirement in practice, we can choose a smaller encoding size.

The Impact of Batch Content

In this subsection, we study the format of a few-shot retrieval task in every iteration, e.g., the number of classes $C$ and images per class $K$. To avoid the impact from batch size, we fix $N = C \times K = 180$ and change $C, K$ at the same time. We set $\alpha = 1.2, m = 0.4, T_n = 10$ in all experiments. We display the results of RLL-(L,M,H) in Table 3.13. We observe that when there are more classes and fewer images per class, i.e., a task becomes more difficult, we obtain better generalisation performance.

3.4.9 Qualitative results

Visualisation of Image Retrieval. In Figure 3.5, we visualise the image retrieval results on the SOP test dataset. For every query, we show its top 4 images in the ranked list of the gallery set. We observe that the learned embedding model is robust and invariant to rotation and viewpoint.

3.5 Study on the critical learning periods of deep metric learning

We have discussed in section 3.3.7 that the early learning period is critical in artificial deep neural networks. In this section, we study and validate them in the context of deep metric
3.6 Conclusion

In this paper, the ranked list loss is proposed to exploit all non-trivial data points in order to provide more informative supervision for learning discriminative embeddings. Following up our CVPR 2019 conference version, we further improve RLL to be a general extension learning empirically from the perspective of example weighting. Our results on the In-shop Clothes dataset are displayed in Table 3.14. The obvious observation is that $T_1$ is a more sensitive factor than $T_2$. Therefore, critical learning periods of deep metric learning also exist in the early learning phase. This is interesting and inspires us that more effort should be spent on the design of the early learning phase.

3.6 Conclusion

In this paper, the ranked list loss is proposed to exploit all non-trivial data points in order to provide more informative supervision for learning discriminative embeddings. Following up our CVPR 2019 conference version, we further improve RLL to be a general extension...
Table 3.14: Study on the dynamic weighting scheme of negative pairs on the In-shop Clothes dataset. The batch size is 60 and the number of training iterations is 30,000.

<table>
<thead>
<tr>
<th>Period</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>Recall@$K$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Later</td>
<td>12</td>
<td>12</td>
<td>87.7</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td></td>
<td>87.7</td>
</tr>
<tr>
<td>Early</td>
<td>12</td>
<td>12</td>
<td>86.2</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td></td>
<td>86.4</td>
</tr>
</tbody>
</table>

of ranking-motivated losses. Concretely, given a query, RLL splits other data points into positive and negative sets, and forces a margin between them. In addition, example mining and weighting are exploited to leverage all informative data points. Our proposed RLL achieves the state-of-the-art performance on two large datasets. Furthermore, we present many other interesting results, which are of high practical value and can be open leads of future research: (1) The results of RLL-L and RLL-M are highly competitive and even better although their network depth is shallower; (2) How to better exploit multi-layer embeddings in deep metric learning? (3) How to better design the early learning phase of deep metric learning, since it is the critical learning stage.
Chapter 4

Example Weighting for Instance Cross Entropy

Loss functions play a crucial role in deep metric learning thus a variety of them have been proposed. Some supervise the learning process by pairwise or tripletwise similarity constraints while others take advantage of structured similarity information among multiple data points. In this chapter, we approach deep metric learning from a novel perspective. We propose instance cross entropy (ICE) which measures the difference between an estimated instance-level matching distribution and its ground-truth one. ICE has three main appealing properties. Firstly, similar to categorical cross entropy (CCE), ICE has clear probabilistic interpretation and exploits structured semantic similarity information for learning supervision. Secondly, ICE is scalable to infinite training data as it learns on mini-batches iteratively and is independent of the training set size. Thirdly, motivated by our relative weight analysis, seamless sample reweighting is incorporated. It rescales samples’ gradients to control the differentiation degree over training examples instead of truncating them by sample mining. In addition to its simplicity and intuitiveness, extensive experiments on three real-world benchmarks demonstrate the superiority of ICE.

4.1 Introduction

Deep metric learning (DML) aims to learn a non-linear embedding function (a.k.a. distance metric) such that the semantic similarities over samples are well captured in the feature space (Sohn, 2016; Tadmor et al., 2016). Due to its fundamental function of learning discriminative representations, DML has diverse applications, such as image retrieval (Oh Song et al., 2016),
clustering (Song et al., 2017), verification (Schroff et al., 2015), few-shot learning (Vinyals et al., 2016) and zero-shot learning (Bucher et al., 2016).

A key to DML is to design an effective and efficient loss function for supervising the learning process, thus significant efforts have been made (Chopra et al., 2005; Law et al., 2017; Oh Song et al., 2016; Schroff et al., 2015; Sohn, 2016; Song et al., 2017; Wu et al., 2017). Some loss functions learn the embedding function from pairwise or triplet-wise relationship constraints (Chopra et al., 2005; Schroff et al., 2015; Tadmor et al., 2016). However, they are known to not only suffer from an increasing number of non-informative samples during training, but also incur considering only several instances per loss computation. Therefore, informative sample mining strategies are proposed (Schroff et al., 2015; Wang et al., 2019c; Wu et al., 2017). Recently, several methods consider semantic relations among multiple examples to exploit their similarity structure (Law et al., 2017; Oh Song et al., 2016; Sohn, 2016; Song et al., 2017). Consequently, these structured losses achieve better performance than pairwise and triple-wise approaches.

In this chapter, we tackle the DML problem from a novel perspective. Specifically, we propose a novel loss function inspired by CCE. CCE is well-known in classification problems owing to the fact that it has an intuitive probabilistic interpretation and achieves great performance, e.g., ImageNet classification (Russakovsky et al., 2015). However, since CCE learns a decision function which predicts the class label of an input, it learns class-level centres for reference (Wang et al., 2017a; Zhang et al., 2018). Therefore, CCE is not scalable to infinite classes and cannot generalise well when it is directly applied to DML (Law et al., 2017).

With scalability and structured information in mind, we introduce instance cross entropy (ICE) for DML. It learns an embedding function by minimising the cross entropy between a predicted instance-level matching distribution and its corresponding ground-truth. In comparison with CCE, given a query, CCE aims to maximise its matching probability with the class-level context vector (weight vector) of its ground-truth class, whereas ICE targets at maximising its matching probability with its similar instances. As ICE does not learn class-level context vectors, it is scalable to infinite training classes, which is an intrinsic demand of DML. Similar to (Goldberger et al., 2005; Law et al., 2017; Oh Song et al., 2016; Sohn, 2016; Song et al., 2017; Wu et al., 2018), ICE is a structured loss as it also considers all other instances in the mini-batch of a given query. We illustrate ICE with comparison to other structured losses in Figure 4.1.

A common challenge of instance-based losses is that many training examples become trivial as model improves. Therefore, we integrate seamless sample reweighting into ICE, which functions similarly with various sample mining schemes (Schroff et al., 2015; Shi
4.1 Introduction

(a) A query versus learned parametric class centroids. All $T$ classes in the training set are considered. Prior work: CCE, Heated-up (Zhang et al., 2018), NormFace (Wang et al., 2017a).

(b) A query versus non-parametric class means. Only classes in the mini-batch are considered. Representative work: TADAM (Oreshkin et al., 2018), DRPR (Law et al., 2019), Prototypical Networks (Snell et al., 2017).

(c) $N$-pair-mc (Sohn, 2016): A query versus one instance per class. A mini-batch has to be 2 examples per class rigidly. Only one instance per negative class is randomly sampled out of 2.

(d) NCA (Goldberger et al., 2005) and S-NCA (Wu et al., 2018): A query versus the rest instances.

(e) Our ICE: A query versus one positive and all negatives per distribution. A query’s number of matching distributions is defined by the number of its positive examples.

Fig. 4.1: Our ICE and related losses. The first row shows prior work of a query versus class centres/means while the second row displays the work of a query versus instances. Note that the cross entropy computation and interpretation are different in different losses. For a mini-batch, we show two classes, i.e., circle and rectangle, with 3 examples per class except $N$-pair-mc which requires 2 samples per class. The icons are at the right bottom. GT means ground-truth matching distribution. When illustrating the losses of a query versus instances in (c), (d) and (e), we index those instances with numbers for clarity, except for the query.
et al., 2016; Sohn, 2016; Wu et al., 2017; Yuan et al., 2017). Existing mining methods require either separate time-consuming process, e.g., class mining (Sohn, 2016), or distance thresholds for data pruning (Schroff et al., 2015; Shi et al., 2016; Wu et al., 2017; Yuan et al., 2017). Instead, our reweighting scheme works without explicit data truncation and mining. It is motivated by the relative weight analysis between two examples. The current common practice of DML is to learn an angular embedding space by projecting all features to a unit hypersphere surface (Law et al., 2017; Movshovitz-Attias et al., 2017; Song et al., 2017). We identify the challenge that without sample mining, informative training examples cannot be differentiated and emphasised properly because the relative weight between two samples is strictly bounded. We address it by sample reweighting, which rescales samples’ gradient to control the differentiation degree among them.

Finally, for intraclass compactness and interclass separability, most methods (Oh Song et al., 2016; Schroff et al., 2015; Tadmor et al., 2016; Wu et al., 2017) use distance thresholds to decrease intraclass variances and increase interclass distances. In contrast, we achieve the target from a perspective of instance-level matching probability. Without any distance margin constraint, ICE makes no assumptions about the boundaries between different classes. Therefore, ICE is easier to apply in applications where we have no prior knowledge about intraclass variances.

Our contributions are summarised: (1) We approach DML from a novel perspective by taking in the key idea of matching probability in CCE. We introduce ICE, which is scalable to an infinite number of training classes and exploits structured information for learning supervision. (2) A seamless sample reweighting scheme is derived for ICE to address the challenge of learning an embedding subspace by projecting all features to a unit hypersphere surface. (3) We show the superiority of ICE by comparing with state-of-the-art methods on three real-world datasets.

4.2 Related Work

4.2.1 Structured Losses by Query versus Class Centres

Heated-up, NormFace, TADAM, DRPR, Prototypical Networks, Proxy-NCA. These methods calculate the similarities between a query and class centres (a.k.a. proxies or prototypes) instead of other instances (Law et al., 2019; Movshovitz-Attias et al., 2017; Oreshkin et al., 2018; Snell et al., 2017; Wang et al., 2017a; Zhang et al., 2018). In Heated-up and NormFace, the class centres are learned parameters of a fully connected layer, which is similar to Center Loss (Wen et al., 2016). While in TADAM, DRPR, and Prototypical
Networks, a class centre is the mean over all embeddings of a class. By comparing a sample with other examples other than class centres, more informative instances can contribute more in ICE.

### 4.2.2 Structured Losses by Query versus Instances

NCA (Goldberger et al., 2005), S-NCA (Wu et al., 2018). NCA learns similarity relationships between instances. Since original NCA learns the whole training data and its time complexity is quadratically proportional to the scale of training data, S-NCA is proposed recently with linear time complexity with respect to the training data size. Instead, ICE is scalable to infinite training data by iterative learning on randomly sampled small-scale instances matching tasks. S-NCA and NCA share the same learning objective. However, they treat the event of all similar instance being correctly recognised as a whole by a sum accumulator. Instead, we maximise the probability of every similar sample being correctly identified individually. Therefore, ICE’s optimisation task is harder, leading to better generalisation.

**N-pair-mc** (Sohn, 2016). The aim of N-pair-mc is to identify one positive example from $N - 1$ negative examples of $N - 1$ classes (one negative example per class). In other words, only one positive and one negative instance per class are considered per loss computation by simulating CCE exactly. Instead, ICE exploits all negative examples to benefit from richer information. When constructing mini-batches, N-pair-mc requires expensive offline class mining and samples 2 images per class. According to (Sohn, 2016) N-pair-mc is superior to NCA.

**Hyperbolic** (Nickel & Kiela, 2018). This method aims to preserve the similarity structures among instances as well. However, it learns a hyperbolic embedding space where the distance depends only on the norm of embeddings. Instead, we learn an angular space where the similarity depends only on the angle between embeddings. Besides, Hyperbolic requires a separate sampling of semantic subtrees when the dataset is large.

### 4.2.3 Sample Mining and Weighting

Mining informative examples or emphasising on them are popular strategies in DML: 1) Mining non-trivial samples during training is crucial for faster convergence and better performance. Therefore, sample mining is widely studied in the literature. In pairwise or triplet-wise approaches (Huang et al., 2016b; Schroff et al., 2015; Wu et al., 2017; Yuan et al., 2017), data pairs with higher losses are emphasized during gradient backpropagation. As for structured losses, Lifted Struct (Oh Song et al., 2016) also focuses on harder examples. Furthermore, (Sohn, 2016) and (Suh et al., 2019) propose to mine hard negative classes
to construct informative input mini-batches. Proxy-NCA (Movshovitz-Attias et al., 2017) addresses the sampling problem by learning class proxies. 2) Assigning higher weights to informative examples is another effective scheme (Wang et al., 2019a,b). Moreover, there are some other novel perspectives to address sample mining or weighting, e.g., hardness-aware examples generation (Zheng et al., 2019) and divide-and-conquer of the embedding space (Sanakoyeu et al., 2019).

Our proposed ICE has a similarity scaling factor which helps to emphasise more on informative examples. Moreover, as described in (Schroff et al., 2015), very hard negative pairs are likely to be outliers and it is safer to mine semi-hard ones. In ICE, the similarity scaling factor is flexible in that it controls the emphasis degree on harder samples. Therefore, a proper similarity scaling factor can help mine informative examples and alleviate the disturbance of outliers simultaneously. **What makes ours different is that we do not heuristically design the mining or weighting scheme from scratch.** Instead, it is built-in and we simply scale it as demonstrated in Section 4.3.4.

### 4.2.4 Discussion

We remark that Prototypical Networks, Matching Networks (Vinyals et al., 2016) and NCA are also scalable and do not require distance thresholds. Therefore, they are illustrated and differentiated in Figure 4.1. Matching Networks are designed specifically for one-shot learning. Similarly, (Triantafillou et al., 2017) design mAP-SSVM and mAP-DLM for few-shot learning, which directly optimises the retrieval performance mAP when multiple positives exist. FastAP (Cakir et al., 2019) is similar to (Triantafillou et al., 2017) and optimises the ranked-based average precision. Instead, ICE processes one positive at a time. In addition, the setting of few-shot learning is different from deep metric learning: Each mini-batch is a complete subtask and contains a support set as training data and a query set as validation data in the few-shot learning. Few-shot learning applies episodic training in practice.

Remarkably, TADAM formulates instances versus class centres and also has a metric scaling parameter for adjusting the impact of different class centres. Contrastively, ICE adjusts the influence of other instances. Furthermore, ours is not exactly distance metric scaling since we simply apply naive cosine similarity as the distance metric at the testing stage. That is why we interpret it as a weighting scheme during training.
4.3 Instance Cross Entropy

Notation. $X = \{(x_i, y_i)\}_{i=1}^{N}$ is an input mini-batch, where $x_i \in \mathbb{R}^{h \times w \times 3}$ and $y_i \in \{1, ..., C\}$ represent $i$-th image and the corresponding label, respectively; $\{x_i^c\}_{i=1}^{N_c}$ is a set of $N_c$ images belonging to $c$-th class, $\forall c, N_c \geq 2$. The number of classes $C$ is generally much smaller than the total number of classes $T$ in the training set ($C \ll T$). Note that $T$ is allowed to be extremely large in DML. Given a sufficient number of different mini-batches, our goal is to learn an embedding function $f$ that captures the semantic similarities among samples in the feature space. We represent deep embeddings of $X$ as $\{f_c^c = f(x_c^c)\}_{i=1}^{N_c}$ for $c = 1, \ldots, C$. Given a query, ‘positives’ and ‘negatives’ refer to samples of the same class and different classes, respectively.

4.3.1 Revisiting Categorical Cross Entropy

CCE is widely used in a variety of tasks, especially classification problems. As demonstrated in (Liu et al., 2016a), a deep classifier consists of two joint components: deep feature learning and linear classifier learning. The feature learning module is a transformation (i.e., embedding function $f$) composed of convolutional and non-linear activation layers. The classifier learning module has one neural layer, which learns $T$ class-level context vectors such that any image has the highest compatibility (logit) with its ground-truth class context vector:

$$p(w_{y_i} \bowtie x_i) = \frac{\exp(f_i^\top w_{y_i})}{\sum_{k=1}^{T} \exp(f_i^\top w_k)}, \quad L_{CCE}(X; f, W) = -\sum_{i=1}^{N} \log p(w_{y_i} \bowtie x_i), \quad (4.1)$$

where $f_i = f(x_i) \in \mathbb{R}^d$ is a $d$-dimensional vector, $p(w_{y_i} \bowtie x_i)$ is the probability (softmax normalised logit) of $x_i$ matching $w_{y_i}$, $W = \{w_k \in \mathbb{R}^d\}_{k=1}^{T}$ is the learned parameters of the classifier. During training, the goal is to maximise the joint probability of all instances being correctly classified. The identical form is minimising the negative log-likelihood, i.e., $L_{CCE}(X; f, W)$. Therefore, the learning objective of CCE is:

$$\arg\max_{f, W} \prod_{i=1}^{N} p(w_{y_i} \bowtie x_i) = \arg\min_{f, W} L_{CCE}(X; f, W).$$

(4.2)

4.3.2 Instance Cross Entropy

In contrast to CCE, ICE is a loss for measuring instance matching quality (lower ICE means higher quality) and does not need class-level context vectors. We remark that an anchor may have multiple positives, which are isolated in separate matching distributions. There is a
Example Weighting for Instance Cross Entropy

matching distribution for every anchor-positive pair versus their negatives as displayed in Figure 4.1e.

Let \( \mathbf{f}_a \) be a random query, we compute its similarities with the remaining points using dot product. We define the probability of the given anchor \( \mathbf{x}_a^c \) matching one of its positives \( \mathbf{x}_i^c (i \neq a) \) as follows:

\[
p(\mathbf{x}_i^c | \mathbf{x}_a^c) = \frac{\exp(\mathbf{f}_a^\top \mathbf{f}_i^c)}{\exp(\mathbf{f}_a^\top \mathbf{f}_i^c) + \sum_{o \neq c} \sum_j \exp(\mathbf{f}_a^\top \mathbf{f}_j^o)} \quad (4.3)
\]

where \( \mathbf{f}_a^\top \mathbf{f}_i^c \) is the similarity between \( \mathbf{x}_a^c \) and \( \mathbf{x}_i^c \) in the embedding space, \( \sum_{o \neq c} \sum_j \exp(\mathbf{f}_a^\top \mathbf{f}_j^o) \) is the sum of similarities between \( \mathbf{x}_a^c \) and its all negatives. Similarly, when the positive is \( \mathbf{x}_i^c \), the probability of one negative point \( \mathbf{x}_j^o (o \neq c) \) matching the anchor is:

\[
p(\mathbf{x}_j^o \bowtie \mathbf{x}_c^c, \mathbf{x}_a^c) = \frac{\exp(\mathbf{f}_a^\top \mathbf{f}_j^o)}{\exp(\mathbf{f}_a^\top \mathbf{f}_j^o) + \sum_{o \neq c} \sum_j \exp(\mathbf{f}_a^\top \mathbf{f}_j^o)} \quad (4.4)
\]

We remark: (1) Dot product measures the similarity between two vectors; (2) Eq. (4.3) represents the probability of a query matching a positive while Eq. (4.1) is the probability of a query matching its ground-truth class. To maximise \( p(\mathbf{x}_i^c \bowtie \mathbf{x}_a^c) \) and minimise \( p(\mathbf{x}_j^o \bowtie \mathbf{x}_a^c, \mathbf{x}_i^c) \) simultaneously, we minimise the Kullback-Leibler divergence (Kullback & Leibler, 1951) between the predicted and ground-truth distributions, which is equivalent to minimising their cross entropy. Since the ground-truth distribution is one-hot encoded, the cross-entropy is \( -\log p(\mathbf{x}_i^c \bowtie \mathbf{x}_a^c) \).

To be more general, for the given anchor \( \mathbf{x}_a^c \), there may exist multiple matching points when \( N_c > 2 \), i.e., \( |\{\mathbf{x}_i^c\}_{i \neq a}| = N_c - 1 > 1 \). In this case, we predict one matching distribution per positive point. Our goal is to maximise the joint probability of all positive instances being correctly identified, i.e., \( p_{\mathbf{x}_a^c} = \prod_{i \neq a} p(\mathbf{x}_i^c \bowtie \mathbf{x}_a^c) \). A case of two positives matching a given query is described in Figure 4.1e.

In terms of mini-batch, each image in \( \mathbf{X} \) serves as the anchor iteratively and we aim to maximise the joint probability of all queries \( \{p_{\mathbf{x}_a^c}\}_{a=1}^{N_c} \). Equivalently, we can achieve this by minimising the sum of all negative log-likelihoods. Therefore, our proposed ICE on \( \mathbf{X} \) is as follows:

\[
L_{\text{ICE}}(\mathbf{X}; \mathbf{f}) = -\sum_{c=1}^{C} \sum_{a=1}^{N_c} \log p_{\mathbf{x}_a^c} = -\sum_{c=1}^{C} \sum_{a=1}^{N_c} \sum_{i \neq a} \log p(\mathbf{x}_i^c \bowtie \mathbf{x}_a^c). \quad (4.5)
\]
4.3.3 Regularisation by $L_2$ Feature Normalisation

Following the common practice in existing DML methods, we apply $L_2$-normalisation to feature embeddings before the inner product. Therefore, the inner product denotes the cosine similarity.

The similarity between two feature vectors is determined by their norms and the angle between them. Without $L_2$ normalisation, the feature norm can be very large, making the model training unstable and difficult. With $L_2$ normalisation, all features are projected to a unit hypersphere surface. Consequently, the semantic similarity score is merely determined by the direction of learned representations. Therefore, $L_2$ normalisation can be regarded as a regulariser during training\(^1\). Note that the principle is quite different from recent hyperspherical learning methods (Liu et al., 2017b,c, 2018a,b; Wang et al., 2018a,b). They enforce the learned weight parameters to a unit hypersphere surface and diversify their angles. In contrast, feature normalisation is output regularisation and invariant to the parametrisation of the underlying neural network (Pereyra et al., 2017). In summary, our learning objective is:

$$
\arg\max_f \prod_{c=1}^C \prod_{a=1}^{N_c} p_{x_a} = \arg\min_f L_{ICE}(X; f) \tag{4.6}
$$

s.t. $\forall a, c, ||f_a||_2 = 1$.

The feature $L_2$-normalisation layer is implemented according to (Wang et al., 2017a). It is a differentiable layer and can be easily inserted at the output of a neural net.

4.3.4 Sample Reweighting of ICE

**Intrinsic sample weighting.** We find that ICE emphasises more on harder samples from the perspective of gradient magnitude. We demonstrate this by deriving the partial derivatives of $L_{ICE}(X; f)$ with respect to positive and negative examples.

Given the query $x_a^c$, the derivative of any of its positive instances can be derived:

$$
\frac{\partial L_{ICE}(X; f)}{\partial f_i} = -\frac{f_i^c \cdot \sum_{\alpha \neq c} \sum_j \exp(f_{\alpha i}^T f_i^j)}{\exp(f_{\alpha i}^T f_i^j) + \sum_{\alpha \neq c} \sum_j \exp(f_{\alpha i}^T f_i^j)} = -f_i^c \cdot (1 - p(x_i^c \langle x_a^c \rangle)). \tag{4.7}
$$

Since $||f_i||_2 = 1$, $w_{(x_i^c; x_a^c)} = ||\frac{\partial L_{ICE}(X; f)}{\partial f_i}||_2 = (1 - p(x_i^c \langle x_a^c \rangle)$ can be viewed as the weight of $f_i^c$ when the anchor is $x_a^c$. Thus, ICE focuses more on harder positive samples, whose $p(x_i^c \langle x_a^c \rangle$ is lower.

\(^1\)The training without $L_2$ feature normalisation leads to the norm of features becoming very large easily and the dot product becoming INF.
Similarly, the partial derivative of its any negative sample is:

$$
\frac{\partial L_{ICE}(X; f)}{\partial f_j} = \sum_{i \neq a} \frac{f_a \cdot \exp(f_a^T f_j)}{\exp(f_a^T f_i) + \sum_{o \neq c} \sum_j \exp(f_a^T f_j)} = f_a \cdot \sum_{i \neq a} p(x_i^j \mid x_a^c, x_i^c),
$$

(4.8)

where \( p(x_i^j \mid x_a^c, x_i^c) \) is the matching probability between \( x_i^j \) and \( x_i^c \) given that the ground-truth example is \( x_i^c \). The weight of \( x_i^j \) w.r.t. \( x_a^c \) is:

$$
w_{x_i^j : x_a^c} = \frac{1 - p(x_i^j \mid x_a^c)}{1 - p(x_i^c \mid x_a^c)} = \frac{\exp(f_a^T f_j) + \sum_{o \neq c} \sum_j \exp(f_a^T f_j)}{\exp(f_a^T f_i) + \sum_{o \neq c} \sum_j \exp(f_a^T f_j)}. \tag{4.9}
$$

Similarly, the relative weight between two negative points of the same anchor \((o \neq c, l \neq c)\) is:

$$
w_{x_i^j ; x_l^c} = \sum_{i \neq a} \frac{p(x_i^j \mid x_a^c, x_l^c)}{\sum_{i \neq a} \sum_j \exp(f_a^T f_j)} = \frac{\exp(f_a^T f_j)}{\exp(f_a^T f_k)}. \tag{4.10}
$$

Note that the positive relative weight in Eq. (4.9) is only decided by \( f_a^T f_i \) and \( f_a^T f_k \) while the negative relative weight in Eq. (4.10) is only determined by \( f_a^T f_j \) and \( f_a^T f_k \). The relative weight is merely determined by the dot product, which is in the range of \([-1, 1]\) and strictly bounded.

**Non-linear scaling for controlling the relative weight.** Inspired by (Hinton et al., 2015), we introduce a scaling parameter to modify the absolute weight non-linearly:

$$
\hat{w}(x_i^j : x_a^c) = \frac{\sum_{o \neq c} \sum_j \exp(s \cdot f_a^T f_j)}{\exp(s \cdot f_a^T f_i) + \sum_{o \neq c} \sum_j \exp(s \cdot f_a^T f_j)} = 1 - \hat{p}(x_i^j \mid x_a^c),
$$

(4.11)

$$
\hat{w}(x_i^j ; x_l^c) = \sum_{i \neq a} \frac{\exp(s \cdot f_a^T f_j)}{\exp(s \cdot f_a^T f_i) + \sum_{o \neq c} \sum_j \exp(s \cdot f_a^T f_j)} = \sum_{i \neq a} \hat{p}(x_i^j \mid x_a^c, x_l^c),
$$

(4.12)

where \( s \geq 1 \) is the scaling parameter. In contrast to \( p \) and \( w \), \( \hat{p} \) and \( \hat{w} \) represent the rescaled matching probability and partial derivative weight, respectively. We remark that we scale the absolute weight non-linearly, which is an indirect way of controlling the relative weight. We do not modify the relative weight directly and Eq. (4.9) and Eq. (4.10) are only for introducing our motivation.
Our objective is to maximise an anchor’s matching probability with its any positive instance competing against its negative set. Therefore, we normalise the rescaled weights based on each anchor:

\[ \bar{w}(x^i; x^a) = \frac{1}{N} \frac{\hat{w}(x^i; x^a)}{\sum_{i \neq a} \hat{w}(x^i; x^a) + \sum_{o \neq c} \sum_j \hat{w}(x^j; x^a)} = \frac{1}{2N} \left( \frac{1 - \hat{p}(x^i \sim x^a)}{\sum_{i \neq a} (1 - \hat{p}(x^i \sim x^a))} \right), \] (4.13)

\[ \bar{w}(x^o; x^a) = \frac{1}{N} \frac{\hat{w}(x^o; x^a)}{\sum_{i \neq a} \hat{w}(x^i; x^a) + \sum_{o \neq c} \sum_j \hat{w}(x^j; x^a)} = \frac{1}{2N} \left( \frac{\sum_{i \neq a} \hat{p}(x^j \sim x^a, x^c)}{\sum_{i \neq a} (1 - \hat{p}(x^i \sim x^a))} \right). \] (4.14)

Note that the denominators in Eq. (4.13) and (4.14) are the accumulated weights of positives and negatives w.r.t. \( x^a \), respectively. Although there are much more negatives than positives, the negative set and positive set contribute equally as a whole, as indicated by \( 1/2 \). \( N = \sum_{c=1}^C N_c \) is the total number of instances in \( X \). We select each instance as the anchor iteratively and treat all anchors equally, as indicated by \( 1/N \).

It is worth noting that during back-propagation, the magnitudes of partial derivatives in Eq. (4.7) and Eq. (4.8), i.e., \( w(x^i; x^a) \) and \( w(x^o; x^a) \), are replaced by \( \bar{w}(x^i; x^a) \) and \( \bar{w}(x^o; x^a) \) respectively. The direction of each individual partial derivative is unchanged. However, since weights are rescaled non-linearly, the final partial derivative of each sample is changed to a better weighted combination of multiple partial derivatives. Final partial derivatives of \( L_{ICE}(X; f) \) w.r.t. positives and negatives are:

\[ \frac{\partial L_{ICE}(X; f)}{\partial f^a} = -f^a \cdot \bar{w}(x^a; x^a) \]

\[ \frac{\partial L_{ICE}(X; f)}{\partial f^o} = f^a \cdot \bar{w}(x^o; x^a) \] (4.15)

The weighting scheme shares the same principle as the popular temperature-based categorical cross entropy (Hinton et al., 2015; Oreshkin et al., 2018). The key is that we should consider not only focusing on harder examples, but also the emphasis degree, which is adjusted by the scaling parameter. A case study of naive ICE, where there are two samples per class in every mini-batch, is analysed in detail in the supplementary material.

### 4.3.5 Complexity Analysis

Algorithm 2 summarises the learning process with ICE. As presented there, the input data format of ICE is the same as CCE, i.e., images and their corresponding labels. In contrast to other methods which require rigid input formats (Schroff et al., 2015; Sohn, 2016), e.g., triplets and n-pair tuplets, ICE is much more flexible. We iteratively select one image as the
Algorithm 2 Learn by minimising ICE stochastically

**Batch setting:** $C$ classes, $N_c$ images from $c$-th class, batch size $N = \sum_{c=1}^{C} N_c$.

**Hyper-setting:** The scaling parameter $s$ and the number of iterations $\tau$.

**Input:** initialised embedding function $f$, iteration counter $iter = 0$.

**Output:** Updated $f$.

**for** $iter < \tau$ **do**

$iter = iter + 1$.

Sample one mini-batch randomly $X = \{ \{ x_c^i \}_{i=1}^{N_c} \}_{c=1}^{C}$.

**Step 1:** Feedforward $X$ into $f$ to obtain feature representations $\{ \{ f_c^i \}_{i=1}^{N_c} \}_{c=1}^{C}$.

**Step 2:** Compute the similarities between an anchor and the remaining instances.

Every example serves as the anchor iteratively.

**for** $f_a^c \in \{ \{ f_c^i \}_{i=1}^{N_c} \}_{c=1}^{C}$ **do**

**for** $f_i^c \in \{ f_c^i \}_{i\neq a}$ **do**

Compute $p(x_c^i \sim x_c^a)$ using Eq. (4.3). // We do not need to compute Eq. (4.4).

end for

end for

Compute $L_{ICE}(X; f)$ using Eq. (4.5).

**Step 3:** Gradient back-propagation to update the parameters of $f$ using Eq. (4.15).

end for

anchor. For each anchor, we aim to maximise its matching probabilities with its positive samples against its negative examples. Therefore, the computational complexity over one mini-batch is $O(N^2)$, being the same as recent online metric learning approaches (Oh Song et al., 2016; Wang et al., 2019c). Note that in FaceNet (Schroff et al., 2015) and $N$-pair-mc (Sohn, 2016), expensive sample mining and class mining are applied, respectively.

### 4.4 A Case Study and Intuitive Explanation of ICE

To make it more clear and intuitive for understanding, we now analyse a naive case of ICE, where there are two samples per class in every mini-batch, i.e., $\forall c, N_c = 2$, $|\{ x_c^i \}_{i\neq a}| = N_c - 1 = 1$. In this case, for each anchor (query), there is only one positive among the remaining data points. As a result, the weighting schemes in Eq. (4.13) for positives and Eq. (4.14) for negatives can be simplified:

$$
\bar{w}(x_c^i; x_o^j) = \frac{1}{2N} \cdot \frac{1 - \hat{p}(x_c^i \sim x_o^j) - \sum_{i \neq a} \hat{p}(x_c^i \sim x_o^j)}{1 - \hat{p}(x_c^i \sim x_o^j)} = \frac{1}{N} \cdot \frac{1}{2}, \tag{4.16}
$$

$$
\bar{w}(x_o^j; x_c^i) = \frac{1}{2N} \cdot \frac{\sum_{i \neq a} \hat{p}(x_o^j \sim x_c^i, x_c^i)}{\sum_{i \neq a} (1 - \hat{p}(x_c^i \sim x_o^j))} = \frac{1}{N} \cdot \frac{1}{2} \cdot \frac{\hat{p}(x_o^j \sim x_c^i, x_c^i)}{1 - \hat{p}(x_c^i \sim x_o^j)}. \tag{4.17}
$$
Firstly, we have $N$ anchors that are treated equally as indicated by $1/N$. Secondly, for each anchor, we aim to recognize its positive example correctly. However, there is a *sample imbalance problem* because each anchor has only one positive and many negatives. ICE addresses it by treating the positive set (single point) and negative set (multiple points) equally, i.e., $1/2$ in Eq. (4.16) and Eq. (4.17). Finally, as there are many negative samples, we aim to focus more on informative ones, i.e., harder negative instances with higher matching probabilities with a given anchor. The non-linear transformation can help control the relative weight between two negative points.

### 4.5 Experiments

#### 4.5.1 Implementation Details and Evaluation Settings

For data augmentation and preprocessing, we follow (Oh Song et al., 2016; Song et al., 2017). In detail, we first resize the input images to $256 \times 256$ and then crop it at $227 \times 227$. We use random cropping and horizontal mirroring for data augmentation during training. To fairly compare with the results reported in (Song et al., 2017), we use a centre cropping without horizontal flipping in the test phase. For the embedding size, we set it to 512 on all datasets following (Law et al., 2017; Sohn, 2016; Wang et al., 2019b). To compare fairly with (Law et al., 2017; Movshovitz-Attias et al., 2017; Song et al., 2017), we choose GoogLeNet V2 (with batch normalisation) (Ioffe & Szegedy, 2015) as the backbone architecture initialised by the publicly available pretrained model on ImageNet (Russakovsky et al., 2015). We simply change the original 1000-neuron fully connected layers followed by softmax normalisation and CCE to 512-neuron fully connected layers followed by the proposed ICE. For faster convergence, we randomly initialise the new layers and optimise them with 10 times larger learning rate than the others as in (Oh Song et al., 2016).

We implement our algorithm in the Caffe framework (Jia et al., 2014). The source code will be available soon.

**Datasets.** Following the evaluation protocol in (Oh Song et al., 2016; Song et al., 2017), we test our proposed method on three popular fine-grained datasets including CARS196 (Krause et al., 2013), CUB-200-2011 (Wah et al., 2011) and SOP (Oh Song et al., 2016). A summary of the datasets is given in Table 4.1. We also keep the same train/test splits. We remark that to test the generalisation and transfer capability of the learned deep metric, the training and test classes are disjoint.

---

2The weight sum of negatives: $\sum_{o\neq c} \sum_j \hat{p}(x^o_j \mid x^c_i, x^o_i) = 1 - \hat{p}(x^c_i \mid x^c_i) \Rightarrow \sum_{o\neq c} \sum_j \tilde{w}(x^o_j \mid x^c_i) = \tilde{w}(x^c_i \mid x^c_i) = 1/(2N)$. 


Example Weighting for Instance Cross Entropy

Table 4.1: A summary of three fine-grained datasets. Training and test classes are disjoint. ‘#’ refers to the number of each item. There are only 5.3 images per class on average in SOP.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>CARS196</th>
<th>CUB-200-2011</th>
<th>SOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context</td>
<td>Cars</td>
<td>Birds</td>
<td>Products</td>
</tr>
<tr>
<td>#Total classes</td>
<td>196</td>
<td>200</td>
<td>22,634</td>
</tr>
<tr>
<td>#Total images</td>
<td>16,185</td>
<td>11,788</td>
<td>120,053</td>
</tr>
<tr>
<td>#Training classes</td>
<td>98</td>
<td>100</td>
<td>11,318</td>
</tr>
<tr>
<td>#Training images</td>
<td>8,054</td>
<td>5,864</td>
<td>59,551</td>
</tr>
<tr>
<td>#Test classes</td>
<td>98</td>
<td>100</td>
<td>11,316</td>
</tr>
<tr>
<td>#Test images</td>
<td>8,131</td>
<td>5,924</td>
<td>60,502</td>
</tr>
</tbody>
</table>

**Evaluation protocol.** We evaluate the learned representations on the image retrieval task in terms of Recall@$K$ performance (Oh Song et al., 2016). Given a query, its $K$ nearest neighbours are retrieved from the database. Its retrieval score is one if there is an image of the same class in the $K$ nearest neighbours and zero otherwise. Recall@$K$ is the average score of all queries.

**Training settings.** All the experiments are run on a single PC equipped with Tesla V100 GPU with 32GB RAM. For optimisation, we use the stochastic gradient descent (SGD) with a weight decay of $1e^{-5}$ and a momentum of 0.8. The base learning rate is set as $1e^{-3}$. The training converges at 20k iterations on SOP while 4k iterations on CARS196 and CUB-200-2011. For learning stability, we train another 10 epochs after convergence. As for the hyper-parameters, we study their impacts in Sec. 4.6.1 and supplementary material. The mini-batch size is 60 for small datasets CARS196 and CUB-200-2011 while 180 for the large benchmark SOP. Additionally, we set $C = 6, N_c = 10$ on CARS196 and CUB-200-2011 while $C = 90, N_c = 2$ on SOP. The design reasons are: 1) SOP has only 5.3 images per class on average. Therefore $N_c$ cannot be very large; 2) It helps to simulate the global structure of deep embeddings, where the database is large and only a few matching instances exist.

Ablation studies of $s$, batch content, batch size and embedding size are presented in the supplementary material.

### 4.5.2 Quantitative Results

**Remarks.** For a fair comparison, we remark that the methods group (Duan et al., 2018; Harwood et al., 2017; Lin et al., 2018; Suh et al., 2019; Ustinova & Lempitsky, 2016; Wang et al., 2017b; Zheng et al., 2019) using GoogLeNet V1 (Szegedy et al., 2015) and another group (Cakir et al., 2019; Sanakoyeu et al., 2019; Wu et al., 2017) using ResNet-50 (He et al., 2016) are not benchmarked. Besides, ensemble models (Kim et al., 2018; Opitz et al., 2017;
Table 4.2: Comparison with the state-of-the-art methods on CARS196, CUB-200-2011 and SOP in terms of Recall@K (%). All the compared methods use GoogLeNet V2 as the backbone architecture. ‘–’ means the results which are not reported in the original paper. The best results in the first block using single embedding are bolded.

<table>
<thead>
<tr>
<th>K</th>
<th>CARS196</th>
<th>CUB-200-2011</th>
<th>SOP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Without fine-tuning</td>
<td>35.6</td>
<td>47.3</td>
<td>59.4</td>
</tr>
<tr>
<td>Fine-tuned with CCE</td>
<td>48.8</td>
<td>58.5</td>
<td>71.0</td>
</tr>
<tr>
<td>Triplet Semihard</td>
<td>51.5</td>
<td>63.8</td>
<td>73.5</td>
</tr>
<tr>
<td>Lifted Struct</td>
<td>53.0</td>
<td>65.7</td>
<td>76.0</td>
</tr>
<tr>
<td>N-pair-mc</td>
<td>53.9</td>
<td>66.8</td>
<td>77.8</td>
</tr>
<tr>
<td>Struct Clust</td>
<td>58.1</td>
<td>70.6</td>
<td>80.3</td>
</tr>
<tr>
<td>Spectral Clust</td>
<td>73.1</td>
<td>82.2</td>
<td>89.0</td>
</tr>
<tr>
<td>Proxy NCA</td>
<td>73.2</td>
<td>82.4</td>
<td>86.4</td>
</tr>
<tr>
<td>RLL</td>
<td>74.0</td>
<td>83.6</td>
<td>90.1</td>
</tr>
<tr>
<td>ICE</td>
<td><strong>77.0</strong></td>
<td><strong>85.3</strong></td>
<td><strong>91.3</strong></td>
</tr>
<tr>
<td>RLL-(L,M,H)</td>
<td>82.1</td>
<td>89.3</td>
<td>93.7</td>
</tr>
<tr>
<td>ICE-(L, M, H)</td>
<td>82.8</td>
<td>89.5</td>
<td>93.7</td>
</tr>
</tbody>
</table>

Xuan et al., 2018; Yuan et al., 2017) are not considered. HTL (Ge et al., 2018) also uses GoogLeNet V2, but it constructs a hierarchical similarity tree over the whole training set and updates the tree every epoch, thus being highly unscalable and expensive in terms of both computation and memory. That is why HTL achieves better performance on small datasets but performs worse than ours on the large dataset SOP. Finally, there are some other orthogonal deep metric learning research topics that are worth studying together in the future, e.g., a robust distance metric (Yuan et al., 2019) and metric learning with continuous labels (Kim et al., 2019a). In GoogLeNet V2, there are three fully connected layers of different depth. We refer to them based on their depth: L for the low-level layer (inception-3c/output), M for the mid-level layer (inception-4e/output) and H for the high-level layer (inception5b/output). By default, we use only ‘H’. We also report the results of their combination (L, M, H) for reference following RLL (Chapter 3).

**Competitors.** All the compared baselines, Triplet Semihard (Schroff et al., 2015), Lifted Struct (Oh Song et al., 2016), N-pair-mc (Sohn, 2016), Struct Clust (Song et al., 2017), Spectral Clust (Law et al., 2017), Proxy-NCA (Movshovitz-Attias et al., 2017), RLL (Wang et al., 2019b) and our ICE are trained and evaluated using the same settings: (1) GoogLeNet V2 serves as the backbone network; (2) All models are initialised with the same pretrained model on ImageNet; (3) All apply the same data augmentation during training and use a centre-cropped image during testing. Some baselines (Oh Song et al., 2016; Schroff et al.,
2015; Sohn, 2016; Song et al., 2017) are implemented in Song et al. (2017) under the same settings. We compare with their reported results. In addition, the results of vanilla GoogLeNet V2 pretrained on ImageNet without fine-tuning and with fine-tuning via minimising CCE are reported in (Law et al., 2017), which can be regarded as the most basic baselines. Among these baselines, Proxy NCA is not scalable as class-level proxies are learned during training. Struct Clust and Spectral Clust are clustering-motivated methods which explicitly aim to optimise the clustering quality. We highlight that clustering performance Normalised Mutual Information (NMI) (Schütze et al., 2008) is not a good assessment for SOP (Law et al., 2017) because SOP has a large number of classes but only 5.3 images per class on average. Therefore, we only report and compare Recall@K performance.

**Results.** Table 4.2 compares the results of our ICE and those of the state-of-the-art DML losses. ICE achieves the best Recall@1 performance on all benchmarks. We observe that only RLL achieves comparable performance in a few terms. However, RLL is more complex since it has three hyper-parameters in total: one weight scaling parameter and two distance margins for positives and negatives, respectively. In addition, its perspective is different since it processes the positive set together similarly with (Triantafillou et al., 2017; Wang et al., 2019a). We note that (Wang et al., 2019a) is also complex in designing weighting schemes and contains four control hyper-parameters. However, our Recall@1 on SOP is 77.3%, which is only 0.9% lower than 78.2% of (Wang et al., 2019a). It is also worth mentioning that among these approaches, except fine-tuned models with CCE, only our method has a clear probability interpretation and aims to maximise the joint instance-level matching probability. As observed, apart from being unscalable, CCE’s performance is much worse than the state-of-the-art methods. Therefore, ICE can be regarded as a successful exploration of softmax regression for learning deep representations in DML. The t-SNE visualisation (Van Der Maaten, 2014) of learned embeddings are available in the supplementary material.

### 4.6 More Ablation Studies

#### 4.6.1 Analysis of Sample Reweighting in ICE

We empirically study the impact of the weight scaling parameter $s$, which is the only hyper-parameter of ICE. It functions similarly with the popular sample mining or example weighting (Wang et al., 2019a,b,c) widely applied in the baselines in Table 4.2. Generally, different $s$ corresponds to different emphasis degree on difficult examples. When $s$ is larger, more difficult instances are assigned with relatively higher weights.
4.6 More Ablation Studies

Table 4.3: The results of different reweighting parameters $s$ on SOP in terms of Recall@$K$ (%). There are 90 classes and 2 images per class in a mini-batch, i.e., the batch size is 180.

<table>
<thead>
<tr>
<th>Reweighting</th>
<th>R@1</th>
<th>R@10</th>
<th>R@100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s = 1$</td>
<td>42.0</td>
<td>58.1</td>
<td>74.1</td>
</tr>
<tr>
<td>$s = 16$</td>
<td>71.0</td>
<td>85.6</td>
<td>93.8</td>
</tr>
<tr>
<td>$s = 32$</td>
<td>73.6</td>
<td>87.5</td>
<td>94.7</td>
</tr>
<tr>
<td>$s = 48$</td>
<td>76.9</td>
<td>89.7</td>
<td>95.5</td>
</tr>
<tr>
<td>$s = 64$</td>
<td><strong>77.3</strong></td>
<td><strong>90.0</strong></td>
<td><strong>95.6</strong></td>
</tr>
<tr>
<td>$s = 80$</td>
<td>75.4</td>
<td>88.7</td>
<td>94.9</td>
</tr>
</tbody>
</table>

Table 4.4: The impact of batch content $C \times k$ on SOP in terms of Recall@$K$ (%). The batch size is $N = 180$ and the scaling parameter is $s = 64$.

<table>
<thead>
<tr>
<th>$N = 180, s = 64$</th>
<th>R@1</th>
<th>R@10</th>
<th>R@100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C \times k = 90 \times 2$</td>
<td>77.3</td>
<td>90.0</td>
<td>95.6</td>
</tr>
<tr>
<td>$C \times k = 60 \times 3$</td>
<td>75.2</td>
<td>88.7</td>
<td>95.2</td>
</tr>
<tr>
<td>$C \times k = 45 \times 4$</td>
<td>74.9</td>
<td>88.7</td>
<td>95.3</td>
</tr>
<tr>
<td>$C \times k = 36 \times 5$</td>
<td>74.6</td>
<td>88.7</td>
<td>95.4</td>
</tr>
</tbody>
</table>

In general, small datasets are more sensitive to minor changes of hyper-settings and much easier to overfit. Therefore, the experiments are conducted on the large dataset SOP. The results are shown in Table 4.3. Note that when $s$ is too small, e.g., $s = 1$, we observe that the training does not converge, which demonstrates the necessity of weighting/mining samples. The most significant observation is that focusing on difficult samples is better but the emphasis degree should be properly controlled. When $s$ increases from 16 to 64, the performance grows gradually. However, when $s = 80$, we observe the performance drops a lot. That may be because extremely hard samples, e.g., outliers, are emphasised when $s$ is too large.

4.6.2 Batch Content

We evaluate the impact of batch content which consists of $C$ classes and $k$ images per class, i.e., $\forall c, N_c = k$. The batch size $N = C \times k$ is set to 180. In our experiments, we change the number of classes $C$ from 36 to 90, and the number of images $k$ from 2 to 5, while keeping the batch size unchanged. Table 4.4 shows the results on SOP dataset. We observe that when there are more classes in the mini-batch, the performance is better. We conjecture that as the number of classes increases, the mini-batch training becomes more difficult and helps the model to generalise better.
Table 4.5: The results of different batch size N on SOP in terms of Recall@K (%). While changing C, we fix \(k = 2\) and \(s = 64\). Therefore, \(N = C \times 2\).

<table>
<thead>
<tr>
<th>(N)</th>
<th>R@1</th>
<th>R@10</th>
<th>R@100</th>
</tr>
</thead>
<tbody>
<tr>
<td>180</td>
<td>77.3</td>
<td>90.0</td>
<td>95.6</td>
</tr>
<tr>
<td>160</td>
<td>75.4</td>
<td>88.8</td>
<td>95.1</td>
</tr>
<tr>
<td>140</td>
<td>75.1</td>
<td>88.7</td>
<td>95.2</td>
</tr>
<tr>
<td>120</td>
<td>75.1</td>
<td>88.6</td>
<td>95.2</td>
</tr>
<tr>
<td>100</td>
<td>74.4</td>
<td>88.2</td>
<td>95.1</td>
</tr>
</tbody>
</table>

Table 4.6: The results of different embedding size on SOP in terms of Recall@K (%). In all experiments: \(s = 64\), \(C = 90\), \(k = 2\), \(N = C \times k = 90 \times 2\).

<table>
<thead>
<tr>
<th>(180 = 90 \times 2, s = 64)</th>
<th>R@1</th>
<th>R@10</th>
<th>R@100</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>72.6</td>
<td>87.1</td>
<td>94.0</td>
</tr>
<tr>
<td>128</td>
<td>74.3</td>
<td>87.9</td>
<td>94.5</td>
</tr>
<tr>
<td>256</td>
<td>75.2</td>
<td>88.6</td>
<td>94.8</td>
</tr>
<tr>
<td>512</td>
<td>77.3</td>
<td>90.0</td>
<td>95.6</td>
</tr>
</tbody>
</table>

### 4.6.3 Batch Size

To explore different batch size \(N\), we fix \(k = 2\) and only change \(C\). In this case, \(N = C \times 2\). Table 4.5 shows that as the number of classes increases, the performance grows. In detail, when the number of classes increases from 50 to 90, the performance raises from 74.4% to 77.3% accordingly. One reason may be that as the number of classes increases, it fits the global structure of the test set better, where there are a large number of classes but only a few positive examples. In addition, the increasing difficulty of mini-batch training can help the model to generalise better.

### 4.6.4 Embedding Size

The dimension of feature representations is an important factor in many DML methods. We conduct experiments on SOP to see the influence of different embedding size. The results are presented in Table 4.6. We observe that when the embedding size is very small, e.g., 64, the performance is much worse. The performance increases gradually as the embedding size grows.
4.7 t-SNE Visualisation

The t-SNE visualisation (Van Der Maaten, 2014) of learned embeddings is available in Figures 4.2, 4.3, 4.4.

4.8 Conclusion

In this chapter, we propose a novel instance-level softmax regression framework, named instance cross entropy, for deep metric learning. Firstly, the proposed ICE has clear probability interpretation and exploits structured semantic similarity information among multiple instances. Secondly, ICE is scalable to infinitely many classes, which is required by DML. Thirdly, ICE has only one weight scaling hyper-parameter, which works as mining informative examples and can be easily selected via cross-validation. Finally, distance thresholds are not applied to achieve intraclass compactness and interclass separability. This indicates that ICE makes no assumptions about intraclass variances and the boundaries between different classes. Therefore ICE has general applicability.
Fig. 4.2: t-SNE visualisation (Van Der Maaten, 2014) on the SOP test set. Best viewed on a monitor when zoomed in.
Fig. 4.3: t-SNE visualisation (Van Der Maaten, 2014) on the CUB-200-2011 test set. Best viewed on a monitor when zoomed in.
Fig. 4.4: t-SNE visualisation (Van Der Maaten, 2014) on the CARS196 test set. Best viewed on a monitor when zoomed in.
Chapter 5

IMAE: Example Weighting for Learning to Classify

In this chapter, we study robust deep learning against abnormal training data from the perspective of example weighting built in empirical loss functions, i.e., the gradient magnitude with respect to logits, an angle that is not thoroughly studied so far. Consequently, we have two key findings: (1) Mean Absolute Error (MAE) Does Not Treat Examples Equally. We present new observations and insightful analysis about MAE, which is theoretically proved to be noise-robust. First, we reveal its underfitting problem in practice. Second, we analyse that MAE’s noise-robustness is from emphasising on uncertain examples instead of treating training samples equally, as claimed in prior work. (2) The Variance of Gradient Magnitude Matters. We propose an effective and simple solution to enhance MAE’s fitting ability while preserving its noise-robustness. Without changing MAE’s overall weighting scheme, i.e., what examples get higher weights, we simply modify its weighting variance non-linearly so that the impact ratio between two examples is adjusted. Our solution is termed Improved MAE (IMAE). We prove IMAE’s effectiveness using extensive experiments: image classification under clean labels, synthetic label noise, and real-world unknown noise. We note IMAE is superior to CCE, the most popular loss for training DNNs.

We remark that we say MAE is noise-robust due to historical reasons. From our viewpoint, a robust loss should have three properties, i.e., making a model: (1) fit well clean patterns in the training set; (2) fit little noisy patterns in the training set; (3) generalise well to unseen validation or testing samples. Instead, MAE makes it hard for a learner to fit clean patterns well. Therefore, MAE does not have the desired noise-robust property in fact.
5.1 Introduction

In this chapter, we target robust deep learning, which is indispensable when it comes to large-scale industrial applications. It is non-affordable to guarantee the quality of training data as its scale grows dramatically. Consequently, abnormal examples generally exist in large-scale real-world scenarios (Berrada et al., 2018), which is caused by many factors, such as incomplete annotation, wrong labelling, subjectiveness, bias and so forth. Unfortunately, DNNs trained with categorical cross entropy (CCE) is able to fit random patterns (Zhang et al., 2017b).

Great advances have been made towards training DNNs robustly when abnormal training examples exist (Arpit et al., 2017; Chang et al., 2017; Jiang et al., 2018; Ren et al., 2018). The robust loss function is one of them. In this chapter, we study a so-claimed robust loss function, mean absolute error (MAE) following (Ghosh et al., 2017; Zhang & Sabuncu, 2018). According to the theoretical analysis of CCE and MAE in (Ghosh et al., 2017), CCE is sensitive to label noise while MAE is noise-tolerant. Thereafter, generalised cross entropy (GCE) (Zhang & Sabuncu, 2018) concludes MAE treats training samples equally, thus being noise-robust. However, our empirical observation and technical analysis lead us to a different and more reasonable conclusion.

Observation: In Table 5.1, when 40% noise exists, compared with CCE, MAE underfits to clean training data points, thus fitting much fewer abnormal examples.

Interpretation: In Figure 5.1, MAE emphasises more on uncertain examples, whose probabilities of being classified to its labelled class are around 0.5, thus being noise-robust.

Specifically, according to Table 5.1, MAE is much more noise-tolerant than CCE. However, its ability to learn meaningful patterns is much weaker, fitting only 74.3% of the clean subset. We provide an intuitive interpretation for this according to Figure 5.1: The variance of MAE’s weight curve along with probability is only 0.09. As a result, the impact ratio between two examples is too small. The impact ratio reflects the relative impact of one example versus another for updating parameters. Due to MAE’s small weight variance, informative samples cannot contribute enough against non-informative ones. Therefore, MAE cannot learn meaningful patterns well and is not widely used.

To adjust MAE’s weight variance, we design an effective and simple solution, IMAE, which non-linearly transforms MAE’s weighting scheme by an exponential function. On the one hand, by preserving MAE’s overall weighting scheme, IMAE is noise-robust. On the

The terms, examples’ weight or impact, and examples’ gradient magnitude w.r.t. logits, are used interchangeably because we define the weight by gradient’s magnitude w.r.t. logits. The impact ratio between two examples is changed when gradients’ magnitude is scaled non-linearly.
Table 5.1: Classification accuracy (%) of CCE, MAE, and IMAE on CIFAR-10 (Krizhevsky, 2009). 40% of training examples, i.e., the noisy subset, have wrong labels. We test each model’s performance on test set, noisy subset and clean subset of training data. The backbone is ResNet56 owning enough capacity (He et al., 2016).

<table>
<thead>
<tr>
<th>Loss</th>
<th>Test set (Generalisation)</th>
<th>Noisy subset (Noise-tolerance)</th>
<th>Clean subset (Learning ability)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCE</td>
<td>63.3</td>
<td>75.0</td>
<td>96.2</td>
</tr>
<tr>
<td>MAE</td>
<td>66.9</td>
<td>8.1</td>
<td>74.3 (worst)</td>
</tr>
<tr>
<td>IMAE</td>
<td><strong>81.5</strong> (best)</td>
<td><strong>6.5</strong> (best)</td>
<td>93.1</td>
</tr>
</tbody>
</table>

other hand, by making the gradient magnitude’s variance over training examples controllable, it learns meaningful patterns much better.

We demonstrate the effectiveness of IMAE under different scenarios. Most importantly, these empirical observations justify that our interpretation of MAE’s underfitting problem is reasonable and our proposed solution is superior. Our key findings are summarised as follows:

- CCE overfits to noise easily because it emphasises on low-probability examples to which abnormal ones generally belong. Although CCE’s weight variance is not large (0.33), its fitting ability benefits from emphasising on low-probability examples.

- MAE is noise-robust by focusing on uncertain (medium-probability) examples instead of treating all equally. However, MAE generally underfits due to its small weights variance (0.09), leading to small impact ratio between even far different examples.

- Our proposed IMAE achieves significantly better performance on robust training against synthetic label noise and realistic unknown noise simply by adjusting MAE’s weight variance, which is inspiring.

5.2 Preliminaries

We denote a training mini-batch as \( X = \{ (x_i, y_i) \}_{i=1}^N \), where there are \( N \) samples. \((x_i, y_i)\) represents \( i \)-th training sample \( x_i \in \mathbb{R}^D \) and its annotated class label \( y_i \in \{1, 2, ..., C\} \). \( D \) is the dimensionality of input samples and \( C \) is the number of all training classes. Let \( f_\theta \) be a deep neural network, which transforms \( x_i \) to a representation \( f_i = f_\theta(x_i) \in \mathbb{R}^E \), \( E \) is the dimensionality of target space and \( \theta \) indicates the parameters to be learned.
To optimise $f_\theta$ during training, a linear classifier is generally trained jointly (Liu et al., 2016a). In general, the linear classifier follows the output embeddings and is composed of one $C$-neuron fully connected (FC) layer, one softmax normalisation layer and one loss layer. The FC layer can be represented as $z_i = W^T f_i \in \mathbb{R}^C$, where $W = [w_1, w_2, ..., w_C] \in \mathbb{R}^{E \times C}$ consists of $C$ weight vectors (the bias term is omitted for brevity). $z_{ij} = w_j^T f_i$ is a logit which indicates the compatibility between sample $x_i$ and class $j$. To produce the probabilities of sample $x_i$ belonging to different classes, we normalise its logit vector $z_i$ using a softmax function:

$$p(j | x_i) = \frac{\exp(z_{ij})}{\sum_{m=1}^C \exp(z_{im})},$$  \hspace{1cm} (5.1)

where $p(j | x_i)$ is the probability of sample $x_i$ being predicted to class $j$.

Let $q(j | x_i)$ be the ground-truth probability of $x_i$ belonging to class $j$, i.e., $q(j | x_i) = 1$ if $j = y_i$, $q(j | x_i) = 0$ otherwise. In the loss layer, if we use CCE, the minimisation objective per iteration is:

$$L_{CCE}(X; f_\theta, W) = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^C q(j | x_i) \log p(j | x_i) = -\frac{1}{N} \sum_{i=1}^N \log p(y_i | x_i).$$  \hspace{1cm} (5.2)

If MAE is applied, the minimisation objective becomes:

$$L_{MAE}(X; f_\theta, W) = \frac{1}{N} \sum_{i=1}^N \sum_{j=1}^C |p(j | x_i) - q(j | x_i)| = \frac{2}{N} \sum_{i=1}^N (1 - p(y_i | x_i)),$$  \hspace{1cm} (5.3)

where $| \cdot |$ is the absolute function.

In summary, we learn a softmax deep network $g_{\theta, W}$, which outputs logits: $z_i = g_{\theta, W}(x_i) = W^T f_\theta(x_i) \in \mathbb{R}^C$. In classification tasks, we use $z = g_{\theta, W}(x)$ to produce logits for a test image $x$. While in verification or retrieval tasks (Wang et al., 2019b,c,g), we only use $f = f_\theta(x)$ as an embedding function. The overall pipeline is described in Figure 5.2. The output of the softmax layer is $p$.

**Remark 1.** The used term—‘uncertain examples’—is a relative concept. In binary classification, the degree of uncertainty can be defined by how closer $p(y_i | x_i)$ is to 0.5. While in multi-class classification, we can use the entropy of an output distribution $p$ to determine the uncertainty degree of a data point.

**Remark 2.** We have the premise that abnormal (noisy) examples have smaller probabilities in general. This premise is widely used and demonstrated by our empirical observations. For example, in Figure 5.4 and Tables 5.1, 5.6, the accuracy of noisy subset is less than that of clean subset consistently.
5.3 Gradient Magnitude Serving as Weight

As shown in Figure 5.2, \( g_{\theta, W} \) can be viewed as a black box and the update of \( \theta \) and \( W \) is based on the back-propagation of logits’ gradient. Therefore, an example’s contribution can be measured by the magnitude of its partial derivative w.r.t. \( z \). It can be regarded as example weighting that is naturally built-in in loss functions.

For brevity and clarity, we summarise the results here and put the detailed derivation in our supplementary material.

5.3.1 Derivative of Softmax, CCE and MAE Layers

**Softmax layer.** According to Eq. (5.1), the derivation of softmax layer is:

\[
\frac{\partial p(y_i|x_i)}{\partial z_{ij}} = \begin{cases} 
  p(y_i|x_i)(1 - p(y_i|x_i)), & j = y_i \\
  -p(y_i|x_i)p(j|x_i), & j \neq y_i
\end{cases}
\]  

(5.4)

**CCE layer.** According to Eq. (5.2), we have

\[
L_{\text{CCE}}(x_i; f_{\theta}, W) = -\log p(y_i|x_i).
\]  

(5.5)

Therefore, we obtain (the parameters \( \theta, W \) are omitted),

\[
\frac{\partial L_{\text{CCE}}(x_i)}{\partial p(j|x_i)} = \begin{cases} 
  -p(y_i|x_i)^{-1}, & j = y_i \\
  0, & j \neq y_i
\end{cases}
\]  

(5.6)
Fig. 5.2: Pipeline of a softmax deep network. There are two reasons for analysing loss functions based on \( \frac{\partial L}{\partial z} \): (1) In gradient back-propagation, the gradients of examples in a mini-batch are fused when computing \( \frac{\partial L}{\partial z} \). (2) Intermediate differences of \( \frac{\partial L}{\partial p} \) lead to final differences of \( \frac{\partial L}{\partial z} \). Therefore, our analysis of \( \frac{\partial L}{\partial z} \) is more direct and accurate versus that of \( \frac{\partial L}{\partial p} \) in (Zhang & Sabuncu, 2018).

### MAE layer

According to Eq. (5.3), we have

\[
L_{\text{MAE}}(x_i; f_\theta, W) = 2(1 - (p(y_i|x_i))).
\]  

(5.7)

Therefore, we obtain

\[
\frac{\partial L_{\text{MAE}}(x_i)}{\partial p(j|x_i)} = \begin{cases} 
-2, & j = y_i \\
0, & j \neq y_i 
\end{cases}.
\]

(5.8)

### 5.3.2 Perspective of Derivatives w.r.t. Logits Other Than Probabilities

**Prior conclusion according to** \( \frac{\partial L_{\text{CCE}}(x_i)}{\partial p(j|x_i)} \), \( \frac{\partial L_{\text{MAE}}(x_i)}{\partial p(j|x_i)} \); (Zhang & Sabuncu, 2018) concludes that CCE is sensitive to abnormal examples while MAE is robust by treating all data points equally according to Eq. (5.6) and Eq. (5.8), respectively.

In this chapter, we propose to further analyse \( \frac{\partial L_{\text{CCE}}(x_i)}{\partial z_{ij}} \), \( \frac{\partial L_{\text{MAE}}(x_i)}{\partial z_{ij}} \) as discussed in Figure 5.2.

According to Eq. (5.6) and (5.4), we calculate \( \frac{\partial L_{\text{CCE}}(x_i)}{\partial z_{ij}} \):

\[
\frac{\partial L_{\text{CCE}}(x_i)}{\partial z_{ij}} = \begin{cases} 
p(y_i|x_i) - 1, & j = y_i \\
p(j|x_i), & j \neq y_i 
\end{cases}.
\]

(5.9)

Analogously, according to Eq. (5.8) and (5.4), we have:

\[
\frac{\partial L_{\text{MAE}}(x_i)}{\partial z_{ij}} = \begin{cases} 
2p(y_i|x_i)(p(y_i|x_i) - 1), & j = y_i \\
2p(y_i|x_i)p(j|x_i), & j \neq y_i 
\end{cases}.
\]

(5.10)
Table 5.2: Summary of CCE, MAE and IMAE. (x, y) is a training example. For simplicity, \( p_y = p(y|x) \), and \( p_j = p(j|x), j \neq y \). \( \sum_{j \neq y} p_j + p_y = 1 \). Prior analysis on loss functions is based on the loss expression or \( \frac{\partial L}{\partial p} \). Instead, we are the first to study the differences of loss functions according to \( ||\frac{\partial L}{\partial z_i}||_1 \). Our empirical evidence justifies its rationality. Note that we have \( L(p_y) = \int \frac{\partial L}{\partial p_y} dp_y, L(1) = 0 \), therefore \( L(p_y) = \int_1^{p_y} \frac{\partial L}{\partial p_y} dp_y \). We remark IMAE is **neither symmetric nor bounded**, which challenges the robustness theories studied in (Ghosh et al., 2017; Wang et al., 2019i; Zhang & Sabuncu, 2018).

| Loss Expression | \( \frac{\partial L}{\partial p} \) | \( \frac{\partial p_y}{\partial z_i} \) | \( \frac{\partial p_j}{\partial z_i}, j \neq y \) | \( \frac{\partial L}{\partial z_i} = \sum_{j=1}^{C} \frac{\partial L}{\partial p_j} \times \frac{\partial p_j}{\partial z_i} \) | \( ||\frac{\partial L}{\partial z_i}||_1 \) |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| CCE             | \(- \log p_y \) | \( 2(1 - p_i) \) | \( 0 \)         | \( p_i(1 - p_i) - p_i p_j \) | \( 2(1 - p_i) \) |
| MAE             | \( \frac{\exp(T p_i(1 - p_i))}{2 p_i(1 - p_i)} - \frac{1}{p_i} \) | \( 0 \)         | \( 0 \)         | \( p_i(1 - p_i) - p_i p_j \) | \( 4 p_i(1 - p_i) \) |
| IMAE            | \( \frac{\exp(T p_i(1 - p_i))}{2 p_i(1 - p_i)} \) | \( 0 \)         | \( 0 \)         | \( p_i(1 - p_i) - p_i p_j \) | \( \exp(T p_i(1 - p_i)) \) |

**Gradient magnitude treated as weight.** In CCE and MAE, training samples are weighted because different ones own different gradient magnitude w.r.t. logit vector \( z \). We choose to measure one gradient’s magnitude by its \( L_1 \) norm because of its simpler statistics than other norms. If one sample’s gradient is larger, its impact is larger during gradient back-propagation.

For CCE, based on Eq. (5.9), the weight of \( x_i \) is:

\[
w_{\text{CCE}}(x_i) = || \frac{\partial L_{\text{CCE}}(x_i)}{\partial z_i} ||_1 = 2(1 - p(y_i|x_i)), \tag{5.11}
\]

where \( || \cdot ||_1 \) denotes \( L_1 \) norm. For MAE, based on Eq. (5.10), the weight of sample \( x_i \) is:

\[
w_{\text{MAE}}(x_i) = || \frac{\partial L_{\text{MAE}}(x_i)}{\partial z_i} ||_1 = 4 p(y_i|x_i)(1 - p(y_i|x_i)), \tag{5.12}
\]

According to Eq. (5.11) and Eq. (5.12), in both CCE and MAE, examples’ impact is determined by their probabilities being predicted to annotated labels.

### 5.4 Improved MAE

IMAЕ transforms MAE’s weighting scheme non-linearly:

\[
w_{\text{IMAЕ}}(x_i) = \exp(T p(y_i|x_i)(1 - p(y_i|x_i))), \tag{5.13}
\]
Fig. 5.3: Although the loss expression of IMAE is not an elementary function, we visualise it by integral, i.e., the area under curve from $p_y$ to 1.

where $T$ controls the exponential base. In back-propagation, we simply scale the gradient w.r.t. logits as follows:

$$\frac{\partial L_{\text{IMAE}}(x_i)}{\partial z_i} = \frac{\partial L_{\text{MAE}}(x_i)}{\partial z_i} w_{\text{IMAE}}(x_i)$$

$$= || \frac{\partial L_{\text{IMAE}}(x_i)}{\partial z_i} ||_1 = w_{\text{IMAE}}(x_i).$$

(5.14)

IMAЕ is a family of robust losses when $T$ changes. Its details are summarised in Table 5.2 and Figure 5.3.

5.4.1 Design Motivation: To Adjust Gradient Magnitude’s Variance and Impact Ratio

Linear scaling also changes magnitude variance. However, it cannot adjust the impact ratio, i.e., the ratio between two gradients’ magnitude. That is why we have tried linear scaling and find it does not work.

Instead, the exponential function is non-linear so that the impact ratio of one sample versus another is re-adjusted compared with original MAE. The hyper-parameter $T$ controls how significant gradient magnitude’s variance and impact ratio are changed.
Furthermore, assuming that samples’ probabilities are uniformly distributed, we compute the gradients’ variance of MAE and IMAE over training data points:

\[ \sigma_{\text{MAE}} = \int_0^1 w_{\text{MAE}}^2(p) dp - (\int_0^1 w_{\text{MAE}}(p) dp)^2 \]  
(5.15)

\[ \sigma_{\text{IMAE}} = \int_0^1 w_{\text{IMAE}}^2(p) dp - (\int_0^1 w_{\text{IMAE}}(p) dp)^2. \]  
(5.16)

We have \( \sigma_{\text{MAE}} = 0.09 \). When \( T = 8 \), \( \sigma_{\text{IMAE}} = 4.55 \).

5.4.2 Discussion of MAE and CCE

The weighting curves of CCE, MAE and IMAE are compared in Figure 5.1. Our key findings are summarised in the introduction of this chapter. We further discuss them as follows:

- MAE’s weighting scheme is appealing and practical in that samples with medium probabilities are emphasized. Generally, high-probability samples are clean and already trained well. While low-probability ones are highly likely to be noisy as a model improves during training. Although all samples are not trained well and probabilities are not meaningful at the beginning, it also does not hurt to focus on medium-probability ones.

- MAE’s gradient magnitude’s variance over data points is only 0.09. As a consequence, the impact ratio of one example versus another is too small. Therefore, the majority contribute almost equally. Therefore, MAE generally underfits to training data.

- Does high loss value usually back-propagate high gradients to update parameters? The answer is NO. Therefore, those theorems based on loss values, e.g., symmetric or bounded conditions are insufficient for analysing robustness of DNNs (Ghosh et al., 2017). Actually, IMAE is neither symmetric nor bounded. However, it is proved to be noise-robust empirically.

These analytical discussions are demonstrated in our empirical studies in Table 5.6 and Figures 5.4.

5.5 Experiments

In this chapter, our main purpose is to study and compare the behaviours of CCE, MAE and IMAE instead of pushing the state-of-the-art results on specific datasets. To demonstrate the
effectiveness of IMAE, it is compared with recent baselines in Sections 5.5.1 and 5.5.2 in different scenarios: (1) Clean labels; (2) Synthetic symmetric and asymmetric noisy labels; (3) Realistic agnostic noise. Results are summarised in Tables 5.3, 5.4 and 5.5. Note that for a strictly fair comparison, all baselines are required to be reimplemented and trained under the same implementation framework and optimisation settings. As generally accepted, this practice is very time-consuming. Therefore, as a trade-off and according to the need, we reimplement and train CCE, MAE and IMAE in this chapter. Some other baselines are reimplemented in Chapter 6.

Analysis of the training dynamics of IMAE against CCE and MAE. We thoroughly visualise and compare the training dynamics of IMAE, CCE and MAE in Section 5.5.3 for empirical justification.

Supplementary studies. In our supplementary material, we further prove IMAE’s effectiveness by: (1) The results on a video retrieval task (video person re-identification); (2) The results of different stochastic optimisers; (3) The ablation study of $T$.

5.5.1 Image Classification on CIFAR-100 with Synthetic Noise

Dataset. CIFAR-100 (Krizhevsky, 2009) contains 100 classes, 500 images per class for training and 100 images per class for testing. The image size is $32 \times 32$.

Synthetic label noise generation. (1) Class-independent (uniform or symmetric) noise: With a probability of $r$, the label of each image is replaced by one of the other class labels uniformly. (2) Class-dependent (non-uniform or asymmetric) noise: The 100 classes of CIFAR-100 are grouped into 20 coarse ones. Every coarse one has 5 fine classes. Following (Wang et al., 2019i), we first randomly select 2 out of 5 fine classes, and then their labels are flipped to each other with a probability of $r$. All instances generated from the same original image by data augmentation share the same label. All test labels are kept intact.

Implementation details. We follow the settings of recent SL (Wang et al., 2019i) and train ResNet44 (He et al., 2016) for a fair comparison with their reported results. We also use the same data augmentation techniques: random horizontal flips and crops of $32 \times 32$ on the images after being padded with 4 pixels on each side. All networks are trained using SGD with a momentum of 0.9, a weight decay of 0.0005 and an initial learning rate of 0.1.

Baselines. IMAE is compared against standard CCE, MAE, and recent robust training baselines: 1) Label Smoothing (LS) trains DNNs on softly smoothed labels instead of one-hot ones; 2) Bootstrapping learns on new labels generated by a convex combination (soft or hard combination) of the original ones and their predictions (Reed et al., 2015). 3) Forward (or Backward) applies a noise-transition matrix to multiply the network’s predictions (or losses) for label correction purpose (Patrini et al., 2017); 4) D2L achieves noise-robustness
5.5 Experiments

Table 5.3: Test accuracy (%) on CIFAR-100 using ResNet44. Results from SL and D2L are different due to different optimisation details. In our experiments, we fix the random seed as 123 and do not use any random computational accelerator for the purpose of exact reproducibility. The best results on each block and our IMAE are bolded.

<table>
<thead>
<tr>
<th>Method</th>
<th>Clean Labels</th>
<th>Symmetric Noisy Labels</th>
<th>$r=0.2$</th>
<th>$r=0.4$</th>
<th>$r=0.6$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCE</td>
<td>64.3</td>
<td>59.3</td>
<td>50.8</td>
<td>25.4</td>
<td></td>
</tr>
<tr>
<td>LS</td>
<td>63.7</td>
<td>58.8</td>
<td>50.1</td>
<td>24.7</td>
<td></td>
</tr>
<tr>
<td>Results</td>
<td>Boot-hard</td>
<td>63.3</td>
<td>57.9</td>
<td>48.2</td>
<td>12.3</td>
</tr>
<tr>
<td>From</td>
<td>Forward</td>
<td>64.0</td>
<td>59.8</td>
<td>53.1</td>
<td>24.7</td>
</tr>
<tr>
<td>SL</td>
<td>D2L</td>
<td>64.6</td>
<td>59.2</td>
<td>52.0</td>
<td>35.3</td>
</tr>
<tr>
<td>GCE</td>
<td>64.4</td>
<td>59.1</td>
<td>53.3</td>
<td>36.2</td>
<td></td>
</tr>
<tr>
<td>SL</td>
<td><strong>66.8</strong></td>
<td><strong>60.0</strong></td>
<td><strong>53.7</strong></td>
<td><strong>41.5</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCE</td>
<td>68.2</td>
<td>52.9</td>
<td>42.9</td>
<td>30.1</td>
<td></td>
</tr>
<tr>
<td>Boot-hard</td>
<td>68.3</td>
<td>58.5</td>
<td>44.4</td>
<td>36.7</td>
<td></td>
</tr>
<tr>
<td>Results</td>
<td>Boot-soft</td>
<td>67.9</td>
<td>57.3</td>
<td>41.9</td>
<td>32.3</td>
</tr>
<tr>
<td>From</td>
<td>Forward</td>
<td>68.5</td>
<td>60.3</td>
<td>51.3</td>
<td>41.2</td>
</tr>
<tr>
<td>D2L</td>
<td>Backward</td>
<td>68.5</td>
<td>58.7</td>
<td>45.4</td>
<td>34.5</td>
</tr>
<tr>
<td>D2L</td>
<td><strong>68.6</strong></td>
<td><strong>62.2</strong></td>
<td><strong>52.0</strong></td>
<td><strong>42.3</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our</td>
<td>CCE</td>
<td><strong>70.0</strong></td>
<td>60.4</td>
<td>53.2</td>
<td>42.1</td>
</tr>
<tr>
<td>Trained</td>
<td>MAE</td>
<td>8.2</td>
<td>6.4</td>
<td>7.3</td>
<td>5.2</td>
</tr>
<tr>
<td>Results</td>
<td>IMAE</td>
<td><strong>69.2</strong></td>
<td><strong>63.4</strong></td>
<td><strong>54.7</strong></td>
<td><strong>43.9</strong></td>
</tr>
</tbody>
</table>

by restricting the dimensionality expansion of learned subspaces during training (Ma et al., 2018); 5) GCE aims to achieve a balance between MAE and CCE (Zhang & Sabuncu, 2018); 6) SL boosts CCE with a noise-robust counterpart, i.e., reverse cross entropy (Wang et al., 2019i); We remark that (Lee et al., 2019) is not benchmarked for two reasons: (1) The used network is not ResNet44 by checking with the authors; (2) Their proposed algorithm is orthogonal to ours because it targets at the inference stage and is a generative classifier on top of pre-trained deep representations. Our IMAE focuses on the training stage and is a softmax-based neural classifier.

Results. We display the results in Tables 5.3 and 5.4. We observe that IMAE is superior to the state-of-the-art. We fix the random seed as 123 and do not use any random computational accelerator for the purpose of exact reproducibility.
Table 5.4: Test accuracy (%) on CIFAR-100 using ResNet44. The best results on each block are bolded.

<table>
<thead>
<tr>
<th>Method</th>
<th>Asymmetric Noisy Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$r=0.2$</td>
</tr>
<tr>
<td>CCE</td>
<td>63.0</td>
</tr>
<tr>
<td>LS</td>
<td>63.0</td>
</tr>
<tr>
<td>Results From SL (Wang et al., 2019i)</td>
<td>63.4</td>
</tr>
<tr>
<td>Forward</td>
<td>64.1</td>
</tr>
<tr>
<td>D2L</td>
<td>62.4</td>
</tr>
<tr>
<td>GCE</td>
<td>63.0</td>
</tr>
<tr>
<td>SL</td>
<td><strong>65.6</strong></td>
</tr>
<tr>
<td>Our Trained Results</td>
<td>CCE</td>
</tr>
<tr>
<td>MAE</td>
<td>7.3</td>
</tr>
<tr>
<td>IMAE</td>
<td><strong>67.5</strong></td>
</tr>
</tbody>
</table>

Table 5.5: Classification accuracy (%) on Clothing1M with ResNet50 (He et al., 2016). The leftmost block’s results are from SL (Wang et al., 2019i) while the middle block’s are from Masking (Han et al., 2018a).

<table>
<thead>
<tr>
<th>CCE</th>
<th>Boot-hard</th>
<th>Forward</th>
<th>D2L</th>
<th>GCE</th>
<th>SL</th>
<th>S-adaptation</th>
<th>Masking</th>
<th>Joint Optim.</th>
<th>Our Trained Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>68.8</td>
<td>68.9</td>
<td>69.8</td>
<td>69.5</td>
<td>69.8</td>
<td>71.0</td>
<td>70.3</td>
<td>71.1</td>
<td>72.2</td>
<td><strong>CCE 73.2</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>MAE 39.7</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>IMAE 73.2</strong></td>
</tr>
</tbody>
</table>

5.5.2 Image Classification on Clothing1M with Realistic Unknown Noise

**Dataset.** Clothing1M (Xiao et al., 2015) contains one million clothing images of fourteen classes from online shopping websites. Its noise type is agnostic. The noise rate is around 38.46%. Additionally, it includes 50k, 14k, and 10k images with clean labels for training, validation, and testing, respectively. To compare fairly with existing algorithms without exploiting auxiliary information from trusted clean data, we also train only on the noisy training data.

**Implementation details.** We follow (Patrini et al., 2017; Tanaka et al., 2018; Wang et al., 2019i) and train ResNet50 initialised by a pretrained ImageNet model (Russakovsky et al., 2015). We apply an SGD optimiser with a momentum of 0.9 and a weight decay of 0.00002. We set the initial learning rate to 0.01 and divide it by 10 after 10k and 15k iterations. We stop training at 30k iterations. Regarding data augmentation, a raw input image is warped to $256 \times 256$, followed by a random crop of $227 \times 227$ and a random horizontal mirroring. The batch size is 84. Every program is run on a single Tesla V100 GPU with 32 GB RAM.
Table 5.6: Results (%) of CCE, MAE and IMAE on CIFAR-10 with different noise rates. For classification accuracy on the testing set, we show the best result achieved during training and the final result when training stops, which are indicated by ‘Best’ and ‘Final’, respectively. For training accuracy, the results on noisy and clean subsets are displayed. The hybrid accuracy represents the result on the combination of testing set and clean training set. We report training and hybrid accuracies of the final model when training terminates. The ultimate objective is to achieve high hybrid accuracy, since both training and testing data points may occur in a deployed system. Note that in practice, we advocate reporting the hybrid accuracy as a weighted combination of training and testing accuracies, where their weights can be decided by their scale, importance or something else. The best result in each column block is bolded. ‘–’ indicates there is no noisy subset.

<table>
<thead>
<tr>
<th>Backbone</th>
<th>Noise rate</th>
<th>Loss</th>
<th>Testing accuracy</th>
<th>Training accuracy: Naive fitting</th>
<th>Hybrid accuracy: Meaningful patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Best</td>
<td>Final</td>
<td>Noisy subset</td>
</tr>
<tr>
<td>ResNet20</td>
<td>0%</td>
<td>CCE</td>
<td>91.5</td>
<td>91.3</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAE</td>
<td>89.3</td>
<td>89.2</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IMAE</td>
<td><strong>91.7</strong></td>
<td><strong>91.4</strong></td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>40%</td>
<td>CCE</td>
<td>81.2</td>
<td>67.0</td>
<td>34.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAE</td>
<td>76.2</td>
<td>75.9</td>
<td>6.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IMAE</td>
<td><strong>84.3</strong></td>
<td><strong>84.0</strong></td>
<td>5.5</td>
</tr>
<tr>
<td></td>
<td>80%</td>
<td>CCE</td>
<td>43.0</td>
<td>20.3</td>
<td>38.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAE</td>
<td>27.7</td>
<td>27.5</td>
<td>9.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IMAE</td>
<td><strong>52.0</strong></td>
<td><strong>41.0</strong></td>
<td>16.8</td>
</tr>
<tr>
<td>ResNet56</td>
<td>0%</td>
<td>CCE</td>
<td>92.4</td>
<td>92.2</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAE</td>
<td>89.0</td>
<td>89.0</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IMAE</td>
<td>92.2</td>
<td><strong>92.2</strong></td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>40%</td>
<td>CCE</td>
<td>81.6</td>
<td>63.3</td>
<td>75.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAE</td>
<td>67.0</td>
<td>66.9</td>
<td>8.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IMAE</td>
<td><strong>82.2</strong></td>
<td><strong>81.5</strong></td>
<td><strong>6.5</strong></td>
</tr>
<tr>
<td></td>
<td>80%</td>
<td>CCE</td>
<td>38.2</td>
<td>16.4</td>
<td>52.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAE</td>
<td>15.2</td>
<td>15.1</td>
<td><strong>9.6</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>IMAE</td>
<td>37.1</td>
<td><strong>34.0</strong></td>
<td>13.0</td>
</tr>
</tbody>
</table>

**Competitors.** Some recent baselines are compared: 1) S-adaptation explicitly estimates latent true labels by an additional softmax layer (Goldberger & Ben-Reuven, 2017); 2) Masking speculates the structure of a noise-transition matrix with human cognition (Han et al., 2018a); 3) Joint Optim. iteratively optimises model’s parameters and latent true labels (Tanaka et al., 2018). Others are introduced in Section 5.5.1. Note that (Han et al., 2019) corrects labels gradually and (Li et al., 2019) exploits meta-learning. They are not technically related and not benchmarked consequently.

**Results.** We display the results in Table 5.5. IMAE outperforms the state-of-the-art, which proves IMAE’s effectiveness under real-world scenarios with agnostic noise. Furthermore, we remark that IMAE is much simpler than those competitors except CCE, MAE.
5.5.3 Empirical Analysis of IMAE Against Basic Baselines CCE and MAE on CIFAR-10

**Dataset.** CIFAR-10 (Krizhevsky, 2009) contains 10 classes, 5k images per class for training and 1k images per class for testing. The image size is $32 \times 32$.

**Implementation details.** We follow the study on CIFAR-10 in (He et al., 2016), which means we use exactly the same architectures (ResNet20, ResNet56) and training settings: a weight decay of 0.0001, a momentum of 0.9, a batch size of 128. The learning rate starts at 0.1, then is divided by 10 at 32k and 48k iterations. Training stops at 100k iterations. Data augmentation is the same as CIFAR-100. For IMAE, **without tuning T case by case**, we fix $T = 0.5$ when training data is clean and $T = 8$ when noise exists although noisy rate is different$^2$.

A well-accepted way to improve data fitting ability is increasing a model’s capacity. Therefore, we train a shallower net ResNet20 and a deeper net ResNet56 for better analysis.

---

$^2$More discussion about the hyper-parameter $T$ is given in our supplementary material.
CIFAR-10 with intact labels

In Table 5.6, we first compare IMAE with CCE and MAE on clean CIFAR-10 using different nets (ResNet20, ResNet56). We observe that IMAE is competitive with CCE and outperforms MAE significantly.

CIFAR-10 with corrupted labels

Following (Arpit et al., 2017; Zhang et al., 2017b), we test the robustness of deep models against corrupted labels. We evaluate on uniform noise because it is more challenging than asymmetric noise which is verified in (Vahdat, 2017).

Majority voting assumption. When generating uniform noise on CIFAR-10, even up to 80% noise rate, clean examples are still the majority because 80% labels are corrupted to other 9 classes evenly. We remark that the majority voting is our reasonable assumption. We believe that if the noise becomes the majority, it is hard to discover meaningful patterns. Being natural and intuitive, the majority define the meaningful data patterns to learn.

Results. The results are summarised in Table 5.6. For more comprehensive and clear comparison, we display the training dynamics in Figures 5.4 (40% noise) and A.5 (80% noise) of the supplementary material. Note that general learning objectives are high final testing accuracy, low accuracy on the noisy training subset, and high accuracy on the clean training subset. Therefore, we report the hybrid accuracy on the combination of testing set and clean training set. We have the following observations:

- Regarding CCE’s test accuracies, the best is always much higher than the final. In Figures 5.4 and A.5, as training progresses, CCE always tries to fit the noisy training subset better. Therefore, CCE learns a lot of error information when severe noise exists. When it comes to MAE and IMAE, the gap between the best and final accuracies is significantly smaller than that of CCE regardless of net’s capacity.

- The training accuracies on both noisy and clean subsets are compared. Whatever the noise rate and net’s capacity are, CCE fits the noisy subset much more. Although MAE fits the noisy subset much less, it fits the clean subset worst. Instead, our IMAE fits the noisy subset little and the clean subset competitively with CCE.

- IMAE obtains the best hybrid accuracy consistently.
5.6 Related Work

Chapters 2, 3 and 4 focus on deep metric learning. Instead, we study on representation learning via robust deep classification in this Chapter. IMAE is a family of robust loss functions, inspired by the intrinsic example weighting scheme of MAE. Therefore, our work is related to some prior work about example weighting and robust loss functions.

5.6.1 Example Weighting

In Ren et al. (2018), a meta-learning algorithm weights data points according to their gradient directions. The meta-learning algorithm is optimised on a clean validation set. In contrast, our IMAE assigns weights to samples based on their gradient magnitude and does not require extra clean set. MentorNet (Jiang et al., 2018) learns data-driven weighting scheme, which guides StudentNet to focus on samples whose labels are more trustful. In Active Bias (Chang et al., 2017) and Focal Loss (Lin et al., 2017), uncertain and hard examples are emphasised, respectively. Other related work on weighting samples includes curriculum learning (Bengio et al., 2009), self-paced learning (Kumar et al., 2010), and hard examples mining (Shrivastava et al., 2016; Wang et al., 2019c). Among those methods, our study is closest to Active Bias (Chang et al., 2017). What makes ours special is that the weighting scheme inherits from MAE, which is naturally built-in in the loss function.

5.6.2 Noise-Robust Theorems on Loss Functions

Noise-robust theorems on loss functions from the angle of symmetric and bounded conditions on loss values have been studied recently (Ghosh et al., 2017; Wang et al., 2019i; Zhang & Sabuncu, 2018). Does a robust loss function have to be symmetric or bounded? The answer is NO according to this work. Although IMAE is neither symmetric nor bounded, we have extensive empirical studies to support its effectiveness.

5.7 Conclusion

In this chapter, we firstly present a thorough study of CCE and MAE technically and empirically. Compared with previous work, we introduce our new findings: 1) MAE underfits to meaningful patterns; 2) MAE emphasises on medium-probability examples instead of treating all samples equally. Secondly, we claim gradient magnitude’s variance matters. As a consequence, we propose an effective and simple solution for addressing MAE’s underfitting
issue. IMAE is a family of robust loss functions whose gradient magnitude’s variance is adjustable.

In this chapter, we focus on analysing MAE and how to improve it. We remark that our empirically demonstrated claim—“Gradient Magnitude’s Variance Matters”—can be applied for other algorithms as well, for example, CCE. In chapter 6, we will investigate this claim in other loss functions. In addition, we will study emphasis mode in addition to emphasis variance, and how to formulate them using emphasis density functions.
Chapter 6

DM: Example Weighting for Learning to Classify

Real-world large-scale datasets usually contain noisy labels and are imbalanced. Therefore, we propose derivative manipulation (DM), a novel and general example weighting approach for training robust deep models under these adverse conditions. DM has two main merits. First, loss function and example weighting are two common techniques in robust learning. In gradient-based optimisation, the role of a loss function is to provide the gradient for back-propagation to update a model, so that the derivative magnitude of an example defines how much impact it has, namely its weight. By DM, we connect the design of loss function and example weighting together. Second, although designing a loss function sometimes has the same effect, we need to care whether a loss is differentiable, and derive its derivative to understand its example weighting scheme. They make the design complicated. Instead, DM is more flexible and straightforward by directly modifying the derivative. Concretely, DM modifies a derivative magnitude function, including transformation and normalisation, after which we term it an emphasis density function, which expresses a weighting scheme. Accordingly, diverse weighting schemes are derived from common probability density functions, including those of well-known robust losses, e.g., MAE and GCE. We conduct extensive experiments demonstrating the effectiveness of DM on both vision and language tasks.

The work in this chapter is an extension of what presented in chapter 5, where we only study the importance of emphasis variance. However, there is another vital research question: what examples should be assigned with the largest weight?
6.1 Introduction

In large-scale machine learning tasks, addressing label noise and sample imbalance is fundamental and has been widely studied (Chang et al., 2017; Reed et al., 2015; Wang et al., 2019d). There are two popular approaches, i.e., robust loss design (Ghosh et al., 2017; Wang et al., 2019d; Zhang & Sabuncu, 2018) and example weighting design (Chang et al., 2017; Jiang et al., 2018; Ren et al., 2018; Shu et al., 2019), due to their easy implementation and widely demonstrated effectiveness. To improve them further, we reveal their connection and unify the design of them via derivative manipulation (DM). On the one hand, the derivative magnitude of an example decides how much impact it has on updating a model (Barron, 2019; Hampel et al., 1986). As a result, a derivative magnitude function defines a weighting scheme over training examples. On the other hand, in gradient-based optimisation, the role of a loss function is to provide gradient used for back-propagation. DM designs the derivative directly so that we do not need to derive it from a loss function.

6.1.1 Why do we design a derivative magnitude function instead of a loss function?

We present two incompatible perspectives on the robustness of a loss function, which motivates us to design the derivative other than a loss function: (1) Loss value. From this viewpoint, a loss function, which is less sensitive to large errors (i.e. residuals), is more robust and preferred (Hastie et al., 2015; Huber, 1981). For example, absolute error is considered more robust than squared error. An outlier has a larger error by definition, but its loss value should not increase dramatically when a robust loss function is applied. (2) Derivative magnitude. A more robust model is less affected by noisy data than clean data. Therefore, a noisy example should have a smaller derivative magnitude. Whether a larger loss value corresponds to a larger derivative depends on a specific loss function. When an example has a large loss value, its derivative can be so small that its effect is negligible. As a consequence, two viewpoints are obviously inconsistent. We remark the first one is misleading. We support the second because the role of a loss is offering the gradient to back-propagate as shown in Figure 6.1a. Accordingly, we propose DM which manipulates the derivative directly so that we do not need to derive it from a loss.

In addition, the derivative magnitude defines an example weighting scheme. Although it is feasible to design a loss function which offers the desired derivative, we need to consider whether it is differentiable and derive its derivative to get its underlying weighting scheme. This “two-step” procedure makes the design of a loss more complicated than DM.
6.1 Introduction

Gradient is directly designed for weighting
Input
Softmax + Loss
Softmax + DM
Gradient is derived from a loss
CNN
Embedding+FC

(a) Common practice and DM. Black and red arrows denote forward process and gradient back-propagation, respectively.

(b) An EDF is an example-level weighting function normalised by its integral over $[0, 1]$.

Fig. 6.1: Illustration of DM in terms of optimisation and example weighting.

6.1.2 Using an emphasis density function to mathematically express example weighting

DM non-linearly transforms the derivative magnitude, followed by derivative normalisation (DN) so that the total emphasis (weight) is one unit. We term a normalised derivative magnitude function an Emphasis Density Function (EDF), which explicitly defines an example-level weighting scheme over data points. An EDF considers emphasis mode and variance, being analogous to a probability density function (PDF), thus we can design an EDF according to existing PDFs. Emphasis mode represents examples whose weight values are the largest. Depending on a scenario, it can be adjusted to focus on easy, semi-hard, or hard examples. Therefore, DM is a superset of existing (heuristically-designed) example weighting methods (Han et al., 2018b; Jiang et al., 2018; Li et al., 2017; Malach & Shalev-Shwartz, 2017; Ren et al., 2018). Emphasis variance decides the spread of emphasis (weight), i.e., the variance of an EDF curve. Intuitively, examples of greater “interest” should contribute more to a model’s update. Emphasis variance controls how significantly they are emphasised. Some representative EDFs are shown in Figure 6.1b.

We summarise our contributions: (1) We propose a novel approach that unifies the design of example weighting and loss function in a single framework. (2) We demonstrate the effectiveness of DM for training robust deep networks on diverse tasks: (a) Image classification under synthetic and real-world label noise; (b) Video retrieval with unknown and diverse abnormal examples; (c) Sentiment classification of movie reviews when label noise and sample imbalance exist. (3) We show DM with diverse network architectures and stochastic optimisers.
6.2 Related Work

The effects of example weighting and loss function are generally overlapped. To understand why DM works and is superior, we revisit two key concepts:

**Rethinking existing robustness theorems on loss functions.** In the prior work, when judging the robustness of a loss function, its example weighting scheme is not considered. Instead, its robustness is judged according to its sensitivity to large errors (Charoenphakdee et al., 2019; Ghosh et al., 2017; Hastie et al., 2015; Huber, 1981). For example, (Ghosh et al., 2017) proposed theorems showing that a deep model is robust to label noise when the loss function is symmetric and bounded. Accordingly, they claimed that Mean Absolute Error (MAE), Mean Square Error (MSE), Categorical Cross Entropy (CCE) are decreasingly robust, which is not the fact. Because in this chapter we find CCE is very competitive with MAE, MSE and Generalised Cross Entropy (GCE) (Zhang & Sabuncu, 2018). However, CCE is neither bounded nor symmetric.

**Rethinking proposed example weighting schemes.** Many weighting schemes have been proposed for different purposes: (1) Easier examples are preferred (Examples with lower error/loss are inferred to be easier (Chang et al., 2017)). For example, curriculum learning (Bengio et al., 2009) picks easier examples in early training. Self-paced learning (Kumar et al., 2010) increases the weights of more difficult examples gradually. (2) Harder examples are emphasised: Hard example mining is demonstrated to accelerate convergence and improve performance in some cases (Gopal, 2016; Loshchilov & Hutter, 2016; Shrivastava et al., 2016). However, note that the derivative magnitude of a loss function also defines a weighting scheme. Then it becomes the interaction between a proposed weighting scheme and the one from a loss function works in previous work. Instead, in DM, there is only one weighting scheme. We compare it with other weighting methods in Table 6.2.

6.3 Derivative Manipulation

Let a training set be \( N \) training examples \( X = \{(x_i, y_i)\}_{i=1}^N \), where \((x_i, y_i)\) denotes \( i \)th sample with input \( x_i \in \mathbb{R}^D \) and label \( y_i \in \{1, 2, ..., C\} \). \( C \) is the number of classes. Consider a deep neural network \( z \) composed of an embedding network \( f(\cdot) : \mathbb{R}^D \rightarrow \mathbb{R}^K \) and a linear classifier \( g(\cdot) : \mathbb{R}^K \rightarrow \mathbb{R}^C \), i.e., \( z_i = z(x_i) = g(f(x_i)) : \mathbb{R}^D \rightarrow \mathbb{R}^C \). Generally, a linear classifier is the last fully-connected layer which outputs logits \( z \in \mathbb{R}^C \). To predict the probabilities of \( x_i \) belonging to different classes, \( z \) is normalised by a softmax function: \( p(j|x_i) = \exp(z_{ij})/\sum_{m=1}^{C} \exp(z_{im}) \), where \( p(j|x_i) \) is the probability of \( x_i \) belonging to class \( j \). For brevity, we define \( p_i = p(y_i|x_i) \).

For a loss function with \( p_i \) as an input, no matter how it’s computed, in general the loss should be monotonically nonincreasing with \( p_i \) (After a softmax layer, maximising
6.3 Derivative Manipulation

$p(y_i|x_i)$ towards one will automatically reduce $p(j|x_i)$, $j \neq y_i$ towards 0. We analyse several well-known losses and find that their derivatives share the same direction, which indicates they share the same optimisation objective although their loss expressions are different. They perform differently only due to the derivative magnitude, which theoretically (mathematically) demonstrates that it is the key.

We propose DM to systematically study the derivative magnitude in which the derivative’s direction is kept the same as those losses. We choose $L_1$ norm to measure the magnitude of derivative because its expression is simpler than other norms. For clarity, we formally define the emphasis mode and variance ($w_i$ is the weight of $i$-th data point):

**Definition 1 (Emphasis Mode $\psi$).** It refers to those examples that own the largest weight. In DM, an example’s weight is only decided by its $p_i$. For brevity, we define $\psi$ to be $p_i$ of examples whose weights are the largest, i.e., $\psi = \arg \max_p w_i$, $\psi \in [0, 1]$.

**Definition 2 (Emphasis Variance $\sigma$).** It is the weight variance over all data points in a batch and is defined by the variance of an EDF curve, which affects the impact ratio between two examples.

### 6.3.1 The derivative direction and magnitude of common losses

We present the derivatives of four popular loss functions: CCE, MAE, MSE and GCE$^1$.

**CCE.** For a given $(x_i, y_i)$, CCE and its derivative with respect to $z_i$ are:

$$L_{\text{CCE}}(x_i, y_i) = -\log p(y_i|x_i) = \frac{\partial L_{\text{CCE}}}{\partial z_{ij}} = \begin{cases} p(y_i|x_i) - 1, & j = y_i \\ p(j|x_i), & j \neq y_i \end{cases}.$$  \hspace{1cm} (6.1)

With $L_1$ norm, we have $||\frac{\partial L_{\text{CCE}}}{\partial z_i}||_1 = 2(1 - p(y_i|x_i)) = 2(1 - p_i)$. The weight of $x_i$ is $w_i^{\text{CCE}} = ||\frac{\partial L_{\text{CCE}}}{\partial z_i}||_1 = 2(1 - p_i)$, meaning examples with smaller $p_i$ get higher weights.

**MAE.** Similarly as above:

$$L_{\text{MAE}}(x_i, y_i) = 1 - p(y_i|x_i) = \frac{\partial L_{\text{MAE}}}{\partial z_{ij}} = \begin{cases} p(y_i|x_i)(p(y_i|x_i) - 1), & j = y_i \\ p(y_i|x_i)p(j|x_i), & j \neq y_i \end{cases}.$$  \hspace{1cm} (6.2)

The weight of $x_i$ is $w_i^{\text{MAE}} = ||\frac{\partial L_{\text{MAE}}}{\partial z_i}||_1 = 2p(y_i|x_i)(1 - p(y_i|x_i)) = 2p_i(1 - p_i)$.

---

$^1$The derivation details of all losses are given in the supplementary material.

$^2$Another classic robust loss is Huber loss (Huber, 1964), which is defined piecewise. It is quadratic for small residuals, and linear for large residuals. Because MAE is linear and MSE is quadratic, we do not spend extra effort analysing Huber loss.
MSE. Similarly as above:

\[
L_{\text{MSE}}(x_i, y_i) = (1 - p(y_i|x_i))^2 \Rightarrow \frac{\partial L_{\text{MSE}}}{\partial z_{ij}} = \begin{cases} 
-2p(y_i|x_i)(p(y_i|x_i) - 1)^2, & j = y_i \\
-2p(y_i|x_i)(p(y_i|x_i) - 1)p(j|x_i), & j \neq y_i.
\end{cases}
\]  

(6.3)

The weight of \( x_i \) is \( w_{i,\text{MSE}} = ||\frac{\partial L_{\text{MSE}}}{\partial z_i}||_1 = 4p(y_i|x_i)(1 - p(y_i|x_i))^2 = 4p_i(1 - p_i)^2 \).

GCE. Similarly as above:

\[
L_{\text{GCE}}(x_i, y_i) = 1 - p(y_i|x_i)^q \Rightarrow \frac{\partial L_{\text{GCE}}}{\partial z_{ij}} = \begin{cases} 
p(y_i|x_i)^q(p(y_i|x_i) - 1), & j = y_i \\
p(y_i|x_i)^q p(j|x_i), & j \neq y_i.
\end{cases}
\]  

(6.4)

where \( q \in [0, 1] \) is a hyperparameter. The weight of \( x_i \) is \( w_{i,\text{GCE}} = ||\frac{\partial L_{\text{GCE}}}{\partial z_i}||_1 = 2p(y_i|x_i)^q(1 - p(y_i|x_i)) = 2p_i^q(1 - p_i) \).

**Derivative direction.** We note \( \frac{\partial L_{\text{CCE}}}{\partial z_i}, \frac{\partial L_{\text{MAE}}}{\partial z_i}, \frac{\partial L_{\text{MSE}}}{\partial z_i} \) and \( \frac{\partial L_{\text{GCE}}}{\partial z_i} \) share the direction. Concretely:

\[
\frac{\partial L_{\text{MAE}}}{\partial z_i} = p_i \times \frac{\partial L_{\text{CCE}}}{\partial z_i}, \quad \frac{\partial L_{\text{MSE}}}{\partial z_i} = 2p_i \times (1 - p_i) \times \frac{\partial L_{\text{CCE}}}{\partial z_i}, \quad \frac{\partial L_{\text{GCE}}}{\partial z_i} = p_i^q \times \frac{\partial L_{\text{CCE}}}{\partial z_i}.
\]  

(6.5)

**Derivative magnitude.** We summarise the weighting schemes of all losses as follows:

\[
w_{i,\text{CCE}} = 2(1 - p_i) \Rightarrow \psi_{\text{CCE}} = 0; \quad w_{i,\text{MAE}} = 2p_i(1 - p_i) \Rightarrow \psi_{\text{MAE}} = 0.5; \quad w_{i,\text{MSE}} = 4p_i(1 - p_i)^2 \Rightarrow \psi_{\text{MSE}} = \frac{1}{3}; \quad w_{i,\text{GCE}} = 2p_i^q(1 - p_i) \Rightarrow \psi_{\text{GCE}} = \frac{q}{q + 1}.
\]  

(6.6)

6.3.2 **Defining an example weighting scheme by an emphasis density function**

**Example weighting via derivative manipulation**

Common losses perform differently only due to the derivate magnitude as shown in the previous section. Therefore, we manipulate the derivative magnitude directly. Concretely, given a weighting function \( w_{i,\text{DM}} \), we scale CCE’s derivative by \( w_{i,\text{DM}}/(2(1 - p_i)) \):

\[
\nabla z_i = w_{i,\text{DM}}/(2(1 - p_i)) \times \frac{\partial L_{\text{CCE}}}{\partial z_i}.
\]  

(6.7)

Then the gradient magnitude of \( z_i \) is: \( ||\nabla z_i||_1 = ||w_{i,\text{DM}}/(2(1 - p_i)) \times \frac{\partial L_{\text{CCE}}}{\partial z_i}||_1 = w_{i,\text{DM}} \).

Treating \( p_i \) as a continuous variable, \( w_{i,\text{DM}} \) can be interpreted as an emphasis density function (EDF). Correspondingly, the integral (area under the curve of \( w_{i,\text{DM}} \)) between a range, e.g.,
6.3 Derivative Manipulation

\[ [\psi_{\text{DM}} - \Delta, \psi_{\text{DM}} + \Delta], \] denotes the accumulative weight of examples whose \( p_i \) is in this range. \( 2\Delta \) is the length of this range, denoting the examples of interest. As \( p_i \in [0, 1] \), we normalise an EDF by its integral over \([0, 1]\), termed a derivative normalisation (DN):

\[
h(w_{\text{DM}}^i) = \frac{w_{\text{DM}}^i}{\int_0^1 w_{\text{DM}}^i \, dp_i} \Rightarrow \int_0^1 h(w_{\text{DM}}^i) \, dp_i = 1. \quad (6.8)
\]

We name \( w_{\text{DM}}^i \) and \( h \) weighting function and EDF, respectively. In Eq. (6.8), the DN operator is trivial. Therefore, for brevity, we discuss the variants of \( w_{\text{DM}}^i \) instead of the normalised \( h \). Next, we discuss how we express an example weighting function \( w_{\text{DM}}^i \) mathematically.

**Choosing an example weighting scheme**

We derive \( w_{\text{DM}}^i \) according to the PDFs of probability distributions of the exponential family.

**Normal distribution variant.** \( \psi \geq 0 \) denotes the emphasis mode while \( \beta \) adjusts the variance:

\[
v_{\text{ND}}(w_{\text{DM}}^i; \psi, \beta) = \exp(-\beta p_i(p_i - 2\psi)). \quad (6.9)
\]

**Exponential distribution variant.** Harder examples have larger (smaller) weights if \( \beta > 0 \) (\( \beta < 0 \)):

\[
v_{\text{ED}}(w_{\text{DM}}^i; \beta) = \exp(\beta (1 - p_i)). \quad (6.10)
\]

**Beta distribution variant.** It covers all weighting schemes of the common losses shown in Eq. (6.6). We remark the difference of coefficients can be ignored since they are gone after DN. \( \alpha, \eta \geq 0 \).

\[
v_{\text{BD}}(w_{\text{DM}}^i; \alpha, \eta) = p_i^{\alpha - 1}(1 - p_i)^{\eta - 1} = \begin{cases} w_{\text{CCE}}^{\alpha} / 2 = (1 - p_i), & \alpha = 1, \eta = 2 \\ w_{\text{MAE}}^{\alpha} / 2 = p_i(1 - p_i), & \alpha = 2, \eta = 2 \\ w_{\text{MSE}}^{\alpha} / 4 = p_i(1 - p_i)^2, & \alpha = 2, \eta = 3 \\ w_{\text{GCE}}^{\alpha} / 2 = p_i^\eta (1 - p_i), & \alpha = q + 1, \eta = 2 \end{cases} \quad (6.11)
\]

Both emphasis mode and variance matter. However, adjusting the variance is inconvenient in \( v_{\text{BD}}(w_{\text{DM}}^i; \alpha, \eta) \). Although \( v_{\text{ND}}(w_i; \psi, \beta) \) controls both of them, its mathematical generality to other weighting schemes is not good. Therefore, we design the other variant of \( w_{\text{DM}}^i \) as follows:

\[
w_{\text{DM}}^i = \exp(\beta p_i \lambda (1 - p_i)), \lambda \geq 0 \Rightarrow \psi_{\text{DM}} = \frac{\lambda}{\lambda + 1} \in [0, 1), \quad (6.12)
\]
where $\lambda$ and $\beta$ are the parameters to control the emphasis mode and variance, respectively.

**Design reasons.** By varying $\lambda$ and $\beta$ in Eq (6.12), we can show that (1) if $\lambda = 0$, $w_{DM}^i = \exp(\beta (1 - p_i))$, it becomes the same as an exponential distribution variant $v_{ED}(w_{DM}^i; \beta)$; (2) If $\lambda = 1$, $w_{DM}^i$ is a normal distribution variant; (3) Eq (6.12) can be viewed as an exponential transformation of $v_{BD}(w_{DM}^i; \alpha, \eta)$, where $\alpha = \lambda + 1, \eta = 2$ and scale it by $\beta$ followed by an exponential transformation; (4) $w_{DM}^i$ is an extension of $w_{GCE}^i$ by making $\lambda \geq 0$, linear scaling and exponential transformation. Finally, its loss expression is not an elementary function and is represented as: $R_{1} \int_{p_i} w_{DM}^i \frac{1}{2p_i(1-p_i)} dp_i$, which is unbounded and non-symmetric in multi-class cases. Despite this, the overall optimisation objective is unchanged and consistent with common losses as discussed.

### 6.4 Experiments

To demonstrate DM’s value as a useful example weighting framework, we conduct extensive experiments, including robust image classification on synthetic and real-world large-scale datasets, robust video retrieval on a large-scale dataset and robust sentiment analysis of movie reviews. Besides label noise, all real-world datasets are highly imbalanced, e.g., the number of videos per person ranges from 1 to 271 in MARS (Zheng et al., 2016), while the number of images per class varies between 18,976 and 88,588 in Clothing 1M (Xiao et al., 2015). In our experiments, we fix the random seed as 123 and do not apply any random computational accelerator for the purpose of exact reproducibility.

#### 6.4.1 A premise for setting the emphasis mode when noisy labels exist

DM has two key components: (1) We optimise the emphasis mode according to an intuition: e.g., when there exists more noise, we use a relatively larger emphasis mode to emphasise on easier examples (Arpit et al., 2017; Jiang et al., 2018); (2) After an emphasis mode is set, we can search for the best emphasis variance if a validation set is given. Concretely, when training data is clean, we set $\lambda = 0$ so that $w_{DM}^i = \exp(\beta (1 - p_i)) \Rightarrow \psi_{DM} = 0$. When label noise exists, we increase $\lambda$ so that $\psi_{DM}$ increases and easier samples become the emphasis mode. For more justification, we present the following premise:

**Premise 1.** Difficult examples have smaller probabilities of being predicted to its annotated labels, i.e., smaller $p_i$. Abnormal examples, including noisy ones and outliers, belong to those difficult ones.

This premise is justified in recent work: (Arpit et al., 2017) shows that DNNs do not fit real datasets by brute-force memorisation. Instead, DNNs learn simple shared patterns before
6.4 Experiments

(a) The average $p_i$ of different examples as training progresses in CCE and DM.

(b) The test accuracy of CCE and DM as the iteration increases.

(c) We optimise $\psi_{DM}$ over four settings from $\{0, 1/3, 1/2, 2/3\}$.

Fig. 6.2: We train ResNet-56 on CIFAR-10. In (a) and (b), we observe noisy examples have much less $p_i$ than clean ones, thus being more difficult examples in both CCE and DM ($\lambda = 1$). In (c), for each label noise rate, we show the optimised $\psi_{DM}$ from $\{0, 1/3, 1/2, 2/3\}$, i.e., $\lambda \in \{0, 1/2, 1, 2\}$.

memorising difficult abnormal data points. Consequently, abnormal samples have smaller $p_i$ than easier ones.

In addition, we present our empirical evidence in Figure 6.2. The learning dynamics prove this premise and DM is superior to CCE: (a) The $p_i$ of clean examples increases while that of noisy ones has no noticeable rise in DM, which means DM hinders fitting of abnormal examples and preserves DNNs’ ability to learn on clean data; (b) DM has the best test accuracy. Furthermore, it is robust, i.e., early stopping is unnecessary. (c) As we increase noise rate, the optimised emphasis mode also increases showing a positive correlation between them. A thorough ablation study on the emphasis mode and variance are reported in the supplementary material.

Remark 1. In the early training phase, $p_i$ is awfully random and non-informative, what does DM do? At this phase, DM weights examples randomly since $p_i$ is random. It is the same in all common losses (weighting schemes). Although nothing makes sense at the beginning, including random initialisation, loss values and weighting schemes, DNNs gradually learn meaningful patterns (Arpit et al., 2017). Therefore, DM boosts learning gradually as shown in Figure 6.2.

6.4.2 Robust image classification

Datasets. (1) CIFAR-10/100 (Krizhevsky, 2009), which contain 10 and 100 classes, respectively. The image size is 32 $\times$ 32. In CIFAR-10, the training data contains 5k images per class while the test set includes 1k images per class. CIFAR-100 has 500 images per class for training and 100 images per class for testing. We generate synthetic label noise on them: (a) Symmetric noise. With a probability of $r$, the original label of an image is changed to
Table 6.1: Test accuracy (%) on CIFAR-100. The best results on each block are bolded. Italic row is the most basic baseline where examples have identical weights. All methods use ResNet-44 (He et al., 2016).

<table>
<thead>
<tr>
<th>Method</th>
<th>Clean Labels</th>
<th>Symmetric Noisy Labels</th>
<th>Asymmetric Noisy Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$r=0.2$</td>
<td>$r=0.4$</td>
</tr>
<tr>
<td>LS</td>
<td>63.7</td>
<td>58.8</td>
<td>50.1</td>
</tr>
<tr>
<td>Boot-hard</td>
<td>63.3</td>
<td>57.9</td>
<td>48.2</td>
</tr>
<tr>
<td>Forward</td>
<td>64.0</td>
<td>59.8</td>
<td>53.1</td>
</tr>
<tr>
<td>D2L</td>
<td>64.6</td>
<td>59.2</td>
<td>52.0</td>
</tr>
<tr>
<td>SL</td>
<td><strong>66.8</strong></td>
<td><strong>60.0</strong></td>
<td><strong>53.7</strong></td>
</tr>
<tr>
<td>CCE</td>
<td><strong>70.0</strong></td>
<td>60.4</td>
<td>53.2</td>
</tr>
<tr>
<td>GCE</td>
<td>63.6</td>
<td><strong>62.4</strong></td>
<td><strong>58.6</strong></td>
</tr>
<tr>
<td>MAE</td>
<td>8.2</td>
<td>6.4</td>
<td>7.3</td>
</tr>
<tr>
<td>MSE</td>
<td>28.0</td>
<td>24.6</td>
<td>21.3</td>
</tr>
<tr>
<td>CCE-DN</td>
<td><strong>69.1</strong></td>
<td>60.7</td>
<td>54.2</td>
</tr>
<tr>
<td>GCE-DN</td>
<td>65.8</td>
<td><strong>62.5</strong></td>
<td><strong>58.3</strong></td>
</tr>
<tr>
<td>MAE-DN</td>
<td>7.5</td>
<td>5.4</td>
<td>4.5</td>
</tr>
<tr>
<td>MSE-DN</td>
<td>25.8</td>
<td>28.4</td>
<td>27.0</td>
</tr>
<tr>
<td>DM(β = 0)</td>
<td>67.2</td>
<td>56.2</td>
<td>50.9</td>
</tr>
<tr>
<td>DM(λ = 0)</td>
<td><strong>70.1</strong></td>
<td>60.9</td>
<td>55.2</td>
</tr>
<tr>
<td>DM(λ = 0.5)</td>
<td><strong>69.3</strong></td>
<td><strong>65.7</strong></td>
<td><strong>61.0</strong></td>
</tr>
<tr>
<td>DM(λ = 1)</td>
<td>69.2</td>
<td>63.4</td>
<td>54.7</td>
</tr>
</tbody>
</table>

one of the other class labels uniformly following (Ma et al., 2018; Wang et al., 2019i). $r$ denotes the noise rate. We remark that some work randomly flips an image’s label to one of all labels including the ground-truth (Kim et al., 2019b; Tanaka et al., 2018). The actual noise rate becomes quite different when the number of classes is small, e.g., CIFAR-10. We do not compare with those results. (b) Asymmetric noise. We generate asymmetric noise for CIFAR-100 following (Wang et al., 2019i). CIFAR-100 has 20 superclasses and every superclass has 5 subclasses. In each superclass, two subclasses are randomly selected and their labels are flipped to each other with a probability of $r$. The overall noise rate is less than $r$. (2) Clothing 1M (Xiao et al., 2015), which is an industrial-level dataset, consisting of around 1 million images of 14 classes from shopping websites. The noise rate is about 38.46% and noise distribution is agnostic.
6.4 Experiments

Results on CIFAR-100

Training details. We follow the settings of (Wang et al., 2019i) for a fair comparison. We use SGD with a momentum of 0.9 and a weight decay of $1e^{-4}$. We train 30k iterations. The learning rate starts at 0.1, and is divided by 10 at 15k and 22k iterations. Standard data augmentation is used: padding images with 4 pixels on every side, followed by a random crop of $32 \times 32$ and horizontal flip. The batch size is 256.

Competitors. We briefly introduce the compared baselines: (1) Analysed losses (CCE, GCE, MAE and MSE) and their variants after DN (CCE-DN, GCE-DN, MAE-DN and MSE-DN); (2) Bootstrapping trains a model with new labels generated by a convex combination of the original ones and their predictions. A convex combination can be either soft (Boot-soft) or hard (Boot-hard) (Reed et al., 2015); (3) Forward (Backward) uses a noise-transition matrix to multiply the network’s predictions (losses) for label correction (Patrini et al., 2017); (4) D2L addresses noise-robustness by restricting the dimensionality expansion of learned subspaces during training; (5) SL modifies CCE symmetrically with a reverse cross entropy; (6) LS denotes label smoothing (Hinton et al., 2015). Note that we do not compare with (Lee et al., 2019). First, its backbone is not ResNet-44 after checking with the authors. Second, their algorithm is orthogonal to ours because it targets at the inference stage and is a generative classifier on top of deep representations.

Results. From Table 6.1, our observations are: (1) When training data is clean, CCE (CCN-DN) is the best against other common losses. Besides, DM($\lambda = 0$) is the best compared with other variants. We conclude by $\psi = 0$, harder examples have higher weights, leading to better performance. (2) When label noise exists, we obtain better performance by increasing $\lambda$ so that $\psi$ is larger. This demonstrates that we should focus on easier data points as label noise increases.

Results on CIFAR-10

Training details. We follow the same settings as MentorNet (Jiang et al., 2018) and train GoogLeNet V1 to compare fairly with its reported results. Optimiser and data augmentation are the same as CIFAR-100.

Competitors. Self-paced (Kumar et al., 2010), Focal Loss (Lin et al., 2017), and MentorNet are representatives of example weighting algorithms. Forgetting (Arpit et al., 2017) searches the dropout parameter in the range of $(0.2, 0.9)$. All methods use GoogLeNet V1 (Szegedy et al., 2015).

Results. The results are shown in Table 6.2: (1) When looking at common losses, they perform differently in different cases. For example, CCE and MSE-DN are better when $r = 0$ and $r = 0.8$. GCE and GCE-DN are preferred when $r = 0.2$ and $r = 0.4$; (2) For DM, we
Table 6.2: Accuracy (%) of DM and other baselines on CIFAR-10 under symmetric label noise. The best results on each block are bolded. Number format of this table follows MentorNet. ‘–’ denotes result was not reported.

<table>
<thead>
<tr>
<th>Method</th>
<th>$r=0$</th>
<th>$r=0.2$</th>
<th>$r=0.4$</th>
<th>$r=0.8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Results From MentorNet</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Forgetting</td>
<td>0.76</td>
<td>0.71</td>
<td>0.44</td>
<td></td>
</tr>
<tr>
<td>Self-paced</td>
<td>0.80</td>
<td>0.74</td>
<td>0.33</td>
<td></td>
</tr>
<tr>
<td>Focal Loss</td>
<td>0.77</td>
<td>0.74</td>
<td>0.40</td>
<td></td>
</tr>
<tr>
<td>Boot-soft</td>
<td>0.78</td>
<td>0.73</td>
<td>0.39</td>
<td></td>
</tr>
<tr>
<td>MentorNet PD</td>
<td>0.79</td>
<td>0.74</td>
<td>0.44</td>
<td></td>
</tr>
<tr>
<td>MentorNet DD</td>
<td>0.79</td>
<td>0.76</td>
<td>0.46</td>
<td></td>
</tr>
<tr>
<td>Our Trained Results</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCE</td>
<td>0.85</td>
<td>0.74</td>
<td>0.35</td>
<td></td>
</tr>
<tr>
<td>GCE</td>
<td>0.83</td>
<td>0.81</td>
<td>0.77</td>
<td>0.18</td>
</tr>
<tr>
<td>MAE</td>
<td>0.57</td>
<td>0.50</td>
<td>0.45</td>
<td>0.19</td>
</tr>
<tr>
<td>MSE</td>
<td>0.80</td>
<td>0.78</td>
<td>0.73</td>
<td>0.29</td>
</tr>
<tr>
<td>DM($\beta=0$)</td>
<td>0.86</td>
<td>0.66</td>
<td>0.56</td>
<td>0.18</td>
</tr>
<tr>
<td>DM($\lambda=0.0$)</td>
<td>0.86</td>
<td>0.77</td>
<td>0.75</td>
<td>0.18</td>
</tr>
<tr>
<td>DM($\lambda=0.5$)</td>
<td>0.86</td>
<td>0.83</td>
<td>0.80</td>
<td>0.57</td>
</tr>
</tbody>
</table>

Table 6.3: DM versus other losses on sentiment classification of movie reviews. The results of common losses after DN are in the brackets. We test on two adverse cases: label noise and sample imbalance. P-N Ratio denotes the ratio of positive reviews to negative ones.

<table>
<thead>
<tr>
<th>Method</th>
<th>CCE(-DN)</th>
<th>GCE(-DN)</th>
<th>MAE(-DN)</th>
<th>MSE(-DN)</th>
<th>DM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label Noise</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r=0.0$</td>
<td>88.9</td>
<td>88.7</td>
<td>88.6</td>
<td>88.5</td>
<td>88.2</td>
</tr>
<tr>
<td>$r=0.2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r=0.4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r=0.8$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Noise</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r=0.4$</td>
<td>75.5</td>
<td>75.0</td>
<td>83.6</td>
<td>84.9</td>
<td>80.2</td>
</tr>
<tr>
<td>P-N Ratio</td>
<td>10:1</td>
<td>78.9</td>
<td>77.0</td>
<td>75.4</td>
<td>77.6</td>
</tr>
<tr>
<td>Ratio</td>
<td>50:1</td>
<td>63.4</td>
<td>60.5</td>
<td>58.6</td>
<td>56.8</td>
</tr>
</tbody>
</table>

Table 6.4: Results of common stochastic optimisers. Adam (Kingma & Ba, 2015) is an adaptive gradient method. We report three settings of it.

<table>
<thead>
<tr>
<th>Method</th>
<th>CCE</th>
<th>GCE</th>
<th>MAE</th>
<th>MSE</th>
<th>DM</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD (lr: 0.01)</td>
<td>64.6</td>
<td>68.8</td>
<td>39.3</td>
<td>58.4</td>
<td>82.0</td>
</tr>
<tr>
<td>SGD + Momentum (lr: 0.01)</td>
<td>61.7</td>
<td>80.7</td>
<td>64.7</td>
<td>76.7</td>
<td>83.8</td>
</tr>
<tr>
<td>Nesterov (lr: 0.01)</td>
<td>57.3</td>
<td>80.0</td>
<td>63.9</td>
<td>76.8</td>
<td>84.0</td>
</tr>
<tr>
<td>Adam (lr: 0.01, delta: 0.1)</td>
<td>39.3</td>
<td>75.7</td>
<td>57.5</td>
<td>66.6</td>
<td>78.2</td>
</tr>
<tr>
<td>Adam (lr: 0.005, delta: 0.1)</td>
<td>44.3</td>
<td>72.6</td>
<td>60.8</td>
<td>67.9</td>
<td>80.8</td>
</tr>
<tr>
<td>Adam (lr: 0.005, delta: 1)</td>
<td>52.0</td>
<td>67.7</td>
<td>37.3</td>
<td>58.5</td>
<td>79.2</td>
</tr>
</tbody>
</table>

Results on Clothing 1M

**Training details.** We follow (Tanaka et al., 2018) to train ResNet-50 (He et al., 2016): (1) We initialise it by a pretrained model on ImageNet (Russakovsky et al., 2015); (2) SGD with a momentum of 0.9 and a weight decay of $2e^{-5}$ is applied. The learning rate starts at 0.01 and is divided by 10 at 10k and 15k iterations. We train 20k iterations; (3) Data augmentation: first resize a raw input image to $256 \times 256$, and then crop it randomly at $224 \times 224$ followed by random horizontal flipping. We set $\lambda=1, \beta=2$ for DM.

**Competitors.** We compare with recent algorithms: (1) S-adaptation applies an auxiliary softmax layer to estimate a noise-transition matrix (Goldberger & Ben-Reuven, 2017); (2) Masking is a human-assisted approach that conveys human cognition to assume the structure of a noise-transition matrix (Han et al., 2018a); (3) Joint Optim. (Tanaka et al., 2018) learns latent true labels and model’s parameters iteratively. Two regularisation terms are added for label estimation and are adjusted in practice; (4) MD-DYR-SH (Arazo et al., 2019) combines dynamic mixup (MD), dynamic bootstrapping plus regularisation (DYR) from soft to hard (SH).
Table 6.5: Accuracy (%) on Clothing1M. The leftmost block’s results are from SL (Wang et al., 2019i) while the middle block’s are from Masking (Han et al., 2018a). Results of common losses after DN are in the brackets.

<table>
<thead>
<tr>
<th>Boot-hard D2L Forward SL</th>
<th>S-adaptation Masking</th>
<th>Joint Optim. MD-DYR-SH</th>
<th>Our Trained Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCE(-DN) GCE(-DN) MAE(-DN) MSE(-DN) DM</td>
<td>68.9 69.5 69.8 71.0</td>
<td>70.3 71.1</td>
<td>72.2 71.0 71.7(72.5) 72.4(64.5) 39.7(16.4) 71.7(69.9) 73.3</td>
</tr>
</tbody>
</table>

**Results.** We display the results in Table 6.5. Under real-world agnostic noise, DM outperforms the state-of-the-art. It is worth noting that the burden of noise-transition matrix estimation in Forward, S-adaptation, Masking and Joint Optim. is heavy, whereas DM is simple and effective.

### 6.4.3 Robust video retrieval

**Dataset and evaluation settings.** MARS contains 20,715 videos of 1,261 persons. There are 1,067,516 frames in total. Because person videos are collected by tracking and detection algorithms, abnormal examples exist as shown in Figure ?? in the supplementary material: Some frames contain only background or an out-of-distribution person. Exact noise type and rate are unknown. We use 8,298 videos of 625 persons for training and 12,180 videos of the other 636 persons for testing. We report the cumulated matching characteristics (CMC) and mean average precision (mAP) results (Zheng et al., 2016).

**Implementation details.** Following (Liu et al., 2017d; Wang et al., 2019c), we train GoogleNet V2 (Ioffe & Szegedy, 2015) and process a video as an image set, which means we use only appearance information without exploiting latent temporal information. A video’s representation is simply the average fusion of its frames’ representations. The learning rate starts from 0.01 and is divided by 2 every 10k iterations. We stop training at 50k iterations. We apply an SGD optimiser with a weight decay of $5e^{-4}$ and a momentum of 0.9. The batch size is 180. Data augmentation is the same as Clothing 1M. At testing, we first $L_2$ normalise videos’ features and then calculate the cosine similarity.

**Results.** The results are displayed in Table 6.6. Although DRSA (Li et al., 2018) and CAE (Chen et al., 2018) exploit extra temporal information by incorporating attention mechanisms, DM is superior to them in terms of both effectiveness and simplicity. OSM+CAA (Wang et al., 2019c) is the only competitive method. However, OSM+CAA combines CCE and weighted contrastive loss to address anomalies, thus being more complex. We highlight that one query may have multiple matching instances in the MARS benchmark so that mAP is a more reliable and accurate performance assessment. DM is the best in terms of mAP.
Table 6.6: Video retrieval results on MARS. All other methods use GoogLeNet V2 except that DRSA and CAE use more complex ResNet-50.

<table>
<thead>
<tr>
<th>Metric</th>
<th>DRSA</th>
<th>CAE</th>
<th>OSM+CAA</th>
<th>Our Trained Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CCE</td>
</tr>
<tr>
<td>mAP (%)</td>
<td>65.8</td>
<td>67.5</td>
<td>72.4</td>
<td>58.1</td>
</tr>
<tr>
<td>CMC-1 (%)</td>
<td>82.3</td>
<td>82.4</td>
<td>84.7</td>
<td>73.8</td>
</tr>
</tbody>
</table>

6.4.4 Sentiment analysis of movie reviews

We report results on the IMDB dataset of movie reviews (Maas et al., 2011; Mesnil et al., 2015). We use Paragraph Vector, PV-DBOW, as a document descriptor (Le & Mikolov, 2014). We train a neural network with one 8-neuron hidden layer and display the results in Table 6.3. Due to space, other details are reported in the supplementary material.

6.4.5 Further analysis

We also experimented with Adam optimiser as shown in Table 6.4. To explore different networks simultaneously, we train ResNet-56 (He et al., 2016) instead of GoogLeNet V1 on CIFAR-10 with 40% symmetric label noise. We observe that DM’s results are consistently the best. Additionally, experiments about comparison with standard regularisers, performance on a small-scale dataset and detailed training analysis under label noise are presented in the supplementary material.

6.5 Conclusion

In this chapter, we propose derivative manipulation for example weighting. DM directly works on gradients bypassing a loss function. As a consequence, it creates great flexibility in designing various example weighting schemes. Extensive experiments on both vision and language tasks empirically show that DM outperforms existing methods despite its simplicity.

In chapters 5 and 6, we have studied example weighting together with robust loss functions in learning to classify. The shared underlying idea is designing a weighting scheme from the angle of derivative. We would like to see this idea to be extended to robust regression tasks in the future work. We believe this extension should be natural and straightforward because of their close relationships in deep learning: (1) Regression and classification tasks differ only in the output and loss layers; (2) Our study of learning to classify is based on the most common algorithm, i.e., multinomial logistic regression (also known as softmax regression), which is a regression task fundamentally.
Chapter 7

Conclusion and Future Research

Generally, deep learning models are trained on a large amount of data and optimised iteratively on small batches (due to limited computational resources versus the amount of training data) by stochastic gradient descent. Not all examples are created equal (Katharopoulos & Fleuret, 2018). Concretely, different training examples have different derivative magnitude, which can be understood as its influence on the update of learning parameters or interpreted as its weight from the angle of example weighting. Therefore, example weighting is universal, implicitly or explicitly existing everywhere in deep learning.

In our practice, we have mainly introduced example weighting in learning to rank and classify, and demonstrated its effectiveness for learning robust and discriminative deep representations, especially when adverse conditions exist, e.g., data noise, label noise, and sample imbalance. According to our empirical evidence, we conclude that example weighting is an important tool, for guiding deep models to treat training samples differentially and learn meaningful patterns on them.

Perhaps surprisingly, we shed light on some novel insights:

- Rethinking Existing Robustness Theorems on Loss Functions: We propose to rethink conventional theorems on the robustness of loss functions in the context of deep learning. Whether a larger loss value corresponds to a larger derivative depends on the particular loss functions. When an example has a very large loss (being non-robust from the perspective of traditional theorems), its derivative may be so small that its effect is negligible (being robust from the angles of gradient descent and example weighting).

- Rethinking Proposed Example Weighting Schemes: Example weighting techniques have been studied a lot in the literature as discussed in the chapter ???. However, the implicit example weighting coming from a loss function is ignored in those previous
work. Therefore, it is unclear how the interaction between a proposed example weighting scheme and the one from a loss function works, whose effect may be positive or negative.

There are many open leads for future research, in terms of both applications and new example weighting design.

### 7.1 Example weighting for other vital applications

Example weighting techniques are crucial and have diverse applications. As demonstrated, its effectiveness for robust deep learning is significant when adverse conditions exist, e.g., noisy training data or wrongly annotated labels. We briefly list some of its other important applications as follows:

- **Long-tailed recognition.** In this task, the number of samples of different classes follows a long-tailed distribution, leading to a well-known challenge of sample imbalance. Example weighting is a popular and effective method for addressing sample imbalance (Chang et al., 2017; Ren et al., 2018; Shu et al., 2019).

- **Towards general robust deep learning against diverse adverse cases and their combination.** In real-world applications, the collected training data may contain multiple adverse cases. For example, the training set contains noisy observations, corrupted labels, and sample imbalance, etc. Consequently, we need to be more careful when designing our example weighting scheme, because it probably works well for a challenge, yet has opposite effect for another one.

- **Example weighting as an alternative of importance sampling for variance reduction.** In (Katharopoulos & Fleuret, 2018), importance sampling has been proposed for variance reduction and convergence speedup. Unfortunately, importance sampling requires calculation of per-sample importance (informativeness) and resampling based on it, which is computationally expensive. Alternatively, after we have per-sample importance, example weighting can be a good option for reducing the variance of the stochastic gradients so that we do not need importance sampling.

- **Active learning (Cohn et al., 1996).** In this learning framework, we are given a small amount of annotated data, and a large pool of unannotated data points. In an active learning system, we train a model and select new data points to annotate using current model iteratively: (1) We start training a model on the initial small labelled dataset;
(2) We apply the trained model to differentiate examples in the unannotated data pool and select a small subset of most informative examples to annotate by experts; (3) We further train the model on all the annotated data to improve its performance. (4) We repeat steps (2) and (3), so that the model improves along with the increasing size of labelled training set. Example weighting and sample selection according to sample uncertainty are widely studied in the literature (Chang et al., 2017; Gal, 2016; Houlsby et al., 2011).

- Label correction for label noise and semi-supervised learning. Another important application of example weighting is label correction in label noise and semi-supervised learning. In semi-supervised learning, if we assign random labels to unannotated data points, then it becomes the same as label noise. As demonstrated in Table B.3, when label noise exists, we can correct the training labels iteratively by repeating two steps: (1) Training a model robustly on the noisy training data; (2) Applying the trained model to predict the labels of training examples and correct the original labels in a proper way, e.g., replacing the original labels by newly predicted labels or the combination of original and predicted labels. Example weighting is an effective approach for training the model robustly so that the newly predicted labels are highly trustful.

- Data efficiency. Recently, the study on data efficiency becomes popular (Toneva et al., 2019). The study on data efficiency can help us understand what data points are more important than others. Additionally, it may help us identify and detect outliers or examples with corrupted labels (Brodley & Friedl, 1999; Jiang et al., 2018; Sukhbaatar & Fergus, 2014). The underlying principal is to learn on most informative data points (Bengio et al., 2009; Chang et al., 2017; Kumar et al., 2010), while removing less informative ones, outliers, and examples with noisy labels. Example weighting is an effective tool to differentiate examples.

### 7.2 To design new example weighting schemes

In general, there are two essential steps for designing an example weighting scheme:

1. Measuring the meaningful informativeness of training examples.

2. Transforming an example’s informativeness to its weight.

Currently, there are practices which work in different scenarios: (1) In curriculum learning, and self-paced learning, the informativeness of an example is defined by its hardness, which
is usually determined by its empirical loss value. (2) In active learning, one example’s informativeness can be defined by its predictive uncertainty (Chang et al., 2017; Gal, 2016). (3) In noise-robust deep learning, some methods differentiate examples according to the agreement of two or multiple networks (Choi et al., 2018; Han et al., 2018b; Malach & Shalev-Shwartz, 2017; Yu et al., 2019). Therefore, to design more effective and efficient example weighting algorithms in the future research, we can: (1) find a better metric to decide the informativeness of training examples; (2) better design the transformation from informativeness to weight.
References


References


Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *ACL*, 2011.

Malach, E. and Shalev-Shwartz, S. Decoupling "when to update" from "how to update". In *NeurIPS*, 2017.


Roth, K., Brattoli, B., and Ommer, B. Mic: Mining interclass characteristics for improved metric learning. In ICCV, 2019.


Tadmor, O., Rosenwein, T., Shalev-Shwartz, S., Wexler, Y., and Shashua, A. Learning a metric embedding for face recognition using the multibatch method. In *NIPS*, 2016.


Appendix A

Supplementary Material for IMAE

A.1 The Impact of $T$ on Gradient Magnitude’s Variance

Assuming samples’ probabilities are uniformly distributed, we calculate the variances of IMAE’s weighting curves with different $T$. We rewrite the Eq. (5.13) (We use $e$ to replace exp for brevity):

$$w_{\text{IMAE}}(p) = e^{T \cdot p(1-p)}, \quad (A.1)$$

where $p$ is the probability of one randomly sampled example being predicted to its annotated label. According to Eq. (5.16), we have,

$$\sigma_{\text{IMAE}} = \int_0^1 w_{\text{IMAE}}^2(p) \, dp - (\int_0^1 w_{\text{IMAE}}(p) \, dp)^2$$

$$= \int_0^1 e^{2T \cdot p(1-p)} \, dp - (\int_0^1 e^{T \cdot p(1-p)} \, dp)^2$$

$$= \frac{\sqrt{\pi} \, \text{erf} \left( \frac{\sqrt{T}}{2} \right) \, e^{\frac{T}{2}}}{\sqrt{2T}} - \frac{\pi \, \text{erf}^2 \left( \frac{\sqrt{T}}{2} \right) \, e^{\frac{T}{2}}}{T}. \quad (A.2)$$

erf is the error function. Therefore we obtain the weighting variances $\sigma_{\text{IMAE}}$ of IMAE with different $T$, as displayed in Table A.1.

Table A.1: The weight variance (gradient magnitude’s variance) of IMAE when $T$ changes.

<table>
<thead>
<tr>
<th>$T$</th>
<th>16</th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
<th>0.5</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_{\text{IMAE}}$</td>
<td>354.113</td>
<td>4.546</td>
<td>0.299</td>
<td>0.040</td>
<td>0.007</td>
<td>0.002</td>
<td>0</td>
</tr>
</tbody>
</table>
Fig. A.1: Sample’s weight along with probability in IMAE with different $T$ (IMAE-$T$). The hyper-parameter $T$ controls gradient magnitude’s variance, and impact ratio between examples consequently. *Better viewed in colour.*

A.2 The Impact of $T$ on Validation Accuracy

We visualise and compare the effect of $T$ on CIFAR-10 test performance. These experiments follow exactly the same settings.

We try two cases: (1). Training labels are intact ($r = 0$); (2). Training labels are corrupted randomly with a probability of 0.4 ($r = 40\%$). In both cases, the test set is kept intact for evaluation. The backbone network is ResNet20.

A.2.1 CIFAR-10 with intact training labels

The test results are shown and compared in Figure A.2a.

When training labels are clean, it is unhelpful to differentiate training samples in a high degree, e.g., the performance is even lower when $T = 16$. The final test accuracies are similar when $T$ ranges from 0 to 8.

A.2.2 CIFAR-10 with corrupted training labels

The results are presented and compared in Figure A.2b. Because there exists 40% label noise, as training progresses, the test accuracy drops, which means the model overfits noisy data gradually.
A.3 The Impact of $T$ on Training Accuracy

However, we observe that higher differentiation degree (larger $T$) works better and is much less susceptible to overfitting to noisy data. In Figure A.2b, the final test accuracies of IMAE-16 and IMAE-8 are much higher than those of other models.

A.3 The Impact of $T$ on Training Accuracy

We visualise and compare the accuracies on the training sets, which indicate how different models fit to training data as training progresses, thus leading to different generalisation performance in the test phase. We present how each model fits its corresponding training set in Figure A.3.

A.3.1 Fitting of the intact training set

As compared in Figure A.3a, all models fit training data similarly when $T$ ranges from 0 to 8. However, when $T = 16$, the differentiation degree becomes too large as shown in Table A.1. When differentiation degree is too large, only a quite small proportion of training data can contribute. Consequently, IMAE-16 underfits training data compared with other models. That is why IMAE-16 has the worst test performance as shown in Figure A.2a.

A.3.2 Fitting of the corrupted training set

The training accuracies of the corrupted training set are displayed in Figure A.3b. We have two observations:

- In cases where noise rate is high, as $T$ increases, the fitting of training data first becomes better, and then becomes worse. Specifically, when $T$ increases from 0 to 8, the training accuracy grows gradually, which means the fitting of training data becomes better. However, when $T = 16$, the weighting variance becomes very large (Table A.1). As a result, IMAE-16’s fitting of training data becomes much worse than IMAE-8’s.

- Fitting corrupted training data better does not mean better generalisation performance. On the one hand, although IMAE-16 fits the training data much worse than IMAE-8 (Figure A.3b), IMAE-16’s test accuracy is slightly better than IMAE-8’s (Figure A.2b). On the other hand, similar to IMAE-8, IMAE-4 fits its training data well (Figure A.3b), but IMAE-4’s test performance is much worse than IMAE-8’s (Figure A.2b).
A.4 Choosing $T$ in Practice

In summary, the training accuracy (fitting of training data) is uninformative for estimating a model’s generalisation performance according to our findings in Section A.3.2. Therefore, it is better to optimise $T$ on a validation set in practice.

For empirical demonstration, since the overlap rate between corrupted and intact training sets is only $(1 - r) = 60\%$, we treat the original intact training set as a validation set. The validation performance of IMAE-16, IMAE-8 and IMAE-4 is compared in Figure A.4. We observe that IMAE-16 and IMAE-8 own similar validation performance, while IMAE-4’s validation accuracy is lower. Furthermore, their validation performance is consistent with their test performance (Figure A.2b). Therefore, we conclude that it is a good practice to optimise $T$ on a validation set in different cases.

A.5 Video person re-identification

Dataset and evaluation settings. MARS contains 20,715 videos of 1,261 persons (Zheng et al., 2016). There are 1,067,516 frames in total. Because person videos are collected by tracking and detection algorithms, abnormal examples exist as shown in Figure 1.3. The exact noise rate is unknown. Following standard settings, we use 8,298 videos of 625 persons for training and 12,180 videos of other 636 persons for testing. We report the cumulated matching characteristics (CMC) and mean average precision (mAP) results.

Implementation details. Following (Liu et al., 2017d; Wang et al., 2019c), we train GoogleNet V2. We also treat a video as an image set, which means we use only appearance information without exploiting latent temporal information. A video’s representation is simply the average fusion of its frames’ representations. We apply the same training settings for each loss. The learning rate starts from 0.01 and is divided by 2 every 10k iterations. We stop training at 50k iterations. We choose SGD optimiser with a weight decay of 0.0005 and momentum of 0.9. The batch size is set to 180. We use standard data augmentation: a $227 \times 227$ crop is randomly sampled and flipped after resizing an original image to $256 \times 256$. At testing, following (Law et al., 2017; Movshovitz-Attias et al., 2017; Wang et al., 2019c), we first $L_2$ normalise videos’ features and then calculate the cosine similarity between every two features.

We explore the performance of different losses in real-world applications instead of pushing the state-of-the-art results.
A.6 The Results of IMAE Using Different Stochastic Optimisers

Results. We compare our method with CCE, MAE and GCE. We implement GCE with its best settings. The results are shown in Table A.2. IMAE outperforms other related methods by a significant margin.

Table A.2: The retrieval results of CCE, MAE, GCE and IMAE on MARS with GoogLeNet V2 (Ioffe & Szegedy, 2015).

<table>
<thead>
<tr>
<th>Metric</th>
<th>CCE</th>
<th>MAE</th>
<th>GCE</th>
<th>IMAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>mAP (%)</td>
<td>58.1</td>
<td>12.0</td>
<td>31.6</td>
<td>70.9</td>
</tr>
<tr>
<td>CMC-1 (%)</td>
<td>73.8</td>
<td>26.0</td>
<td>51.5</td>
<td>83.5</td>
</tr>
</tbody>
</table>

A.6 The Results of IMAE Using Different Stochastic Optimisers

In this section, we study the performance of IMAE when different stochastic optimisers are used. The results are presented in Table A.3. We observe that IMAE’s results are the best consistently.

Table A.3: The results of algorithms using different stochastic optimisers on CIFAR-10 with 40% class-independent (symmetric) label noise. The trained network is ResNet56 (He et al., 2016). The key hyper-parameters of all optimisers are shown. Other settings are fixed to be the same as presented in the implementation details of Section 5.5.3, e.g., weight decay = 0.0001. Since Adam is an adaptive gradient method, we show several variants of it.

<table>
<thead>
<tr>
<th></th>
<th>SGD (lr: 0.01)</th>
<th>SGD + Momentum (lr: 0.01)</th>
<th>Nesterov (lr: 0.01)</th>
<th>Adam (lr: 0.01, delta: 0.1)</th>
<th>Adam (lr: 0.005, delta: 0.1)</th>
<th>Adam (lr: 0.005, delta: 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCE</td>
<td>64.3</td>
<td>60.6</td>
<td>56.4</td>
<td>42.5</td>
<td>44.5</td>
<td>50.3</td>
</tr>
<tr>
<td>MAE</td>
<td>39.3</td>
<td>64.7</td>
<td>64.1</td>
<td>68.2</td>
<td>59.9</td>
<td>41.4</td>
</tr>
<tr>
<td>GCE</td>
<td>68.8</td>
<td>80.5</td>
<td>79.7</td>
<td>73.2</td>
<td>70.6</td>
<td>69.3</td>
</tr>
<tr>
<td>IMAE</td>
<td><strong>82.0</strong></td>
<td><strong>83.5</strong></td>
<td><strong>83.7</strong></td>
<td><strong>75.5</strong></td>
<td><strong>76.3</strong></td>
<td><strong>78.6</strong></td>
</tr>
</tbody>
</table>
Supplementary Material for IMAE

(a) Test accuracies of IMAE-T trained on intact training labels.

(b) Test accuracies of IMAE-T trained on corrupted training labels.

Fig. A.2: The accuracy on the CIFAR-10 test set along with training iterations. We display the results when training on intact training set and corrupted training set. *Better viewed in colour.*
A.6 The Results of IMAE Using Different Stochastic Optimisers

(a) The training accuracies of IMAE-\(T\) on intact training set.

(b) The training accuracies of IMAE-\(T\) on corrupted training set.

Fig. A.3: The accuracy on the CIFAR-10 training sets along with training iterations. We show the results when training on intact training set and corrupted training set. Better viewed in colour.
Supplementary Material for IMAE

Fig. A.4: IMAE-16’s, IMAE-8’s and IMAE-4’s accuracies on the clean training set when they are trained on the corrupted training set. The overlap rate between corrupted and intact training sets is only \((1 - r) = 60\%\). Therefore, we can use the original training set as a validation set. Better viewed in colour.
A.6 The Results of IMAE Using Different Stochastic Optimisers

Fig. A.5: CIFAR-10 with a noise rate of $r = 80\%$. The accuracies on the testing set, the noisy subset and clean subset of the training set along with training iterations. The legend on the top left is shared by all subfigures. Better viewed in colour.
Appendix B

Supplementary Material for Derivative Manipulation

B.1 Derivation Details of Softmax, CCE, MAE and GCE

B.1.1 Derivation of Softmax Normalisation

We rewrite $p(y_i|x_i)$ as follows:

$$p(y_i|x_i)^{-1} = 1 + \sum_{j \neq y_i} \exp(z_{ij} - z_{iy_i}).$$  \hspace{1cm} (B.1)

For left and right sides of Eq. (B.1), we calculate their derivatives w.r.t. $z_{ij}$ simultaneously.

If $j = y_i$,

$$-1 \frac{\partial p(y_i|x_i)}{p(y_i|x_i)^2} \frac{1}{z_{iy_i}} = - \sum_{j \neq y_i} \exp(z_{ij} - z_{iy_i}) $$

$$\Rightarrow \frac{\partial p(y_i|x_i)}{z_{iy_i}} = p(y_i|x_i)(1 - p(y_i|x_i)).$$ \hspace{1cm} (B.2)

If $j \neq y_i$,

$$-1 \frac{\partial p(y_i|x_i)}{p(y_i|x_i)^2} \frac{1}{z_{ij}} = \exp(z_{ij} - z_{iy_i}) $$

$$\Rightarrow \frac{\partial p(y_i|x_i)}{z_{ij}} = -p(y_i|x_i)p(j|x_i).$$ \hspace{1cm} (B.3)
In summary, the derivation of softmax layer is:

$$\frac{\partial p(y_i|x_i)}{\partial z_{ij}} = \begin{cases} 
p(y_i|x_i)(1 - p(y_i|x_i)), & j = y_i \\
p(y_i|x_i)p(j|x_i), & j \neq y_i \end{cases} \quad (B.4)$$

### B.1.2 Derivation of CCE

According to Eq. (6.1), we have

$$L_{CCE}(x_i; f_\theta, W) = -\log p(y_i|x_i). \quad (B.5)$$

Therefore, we obtain (the parameters are omitted for brevity),

$$\frac{\partial L_{CCE}}{\partial p(j|x_i)} = \begin{cases} 
-p(y_i|x_i)^{-1}, & j = y_i \\
0, & j \neq y_i \end{cases}. \quad (B.6)$$

### B.1.3 Derivation of MAE

According to Eq. (6.2), we have

$$L_{MAE}(x_i; f_\theta, W) = 1 - (p(y_i|x_i)). \quad (B.7)$$

Therefore, we obtain

$$\frac{\partial L_{MAE}}{\partial p(j|x_i)} = \begin{cases} 
-1, & j = y_i \\
0, & j \neq y_i \end{cases}. \quad (B.8)$$

### B.1.4 Derivation of GCE

According to Eq. (6.4), we have

$$L_{GCE}(x_i; f_\theta, W) = \frac{1 - p(y_i|x_i)^q}{q}. \quad (B.9)$$

Therefore, we obtain

$$\frac{\partial L_{GCE}}{\partial p(j|x_i)} = \begin{cases} 
-p(y_i|x_i)^{q-1}, & j = y_i \\
0, & j \neq y_i \end{cases}. \quad (B.10)$$
B.1.5 Derivatives w.r.t. Logits $z_i$

$\frac{\partial L_{\text{CCE}}}{\partial z_i}$

The calculation is based on Eq. (B.6) and Eq. (B.4).

If $j = y_i$, we have:

$$
\frac{\partial L_{\text{CCE}}}{\partial z_{iy_i}} = \sum_{j=1}^{C} \frac{\partial L_{\text{CCE}}}{\partial p(j|x_i)} \frac{\partial p(y_i|x_i)}{z_{ij}}
= p(y_i|x_i) - 1.
$$

(B.11)

If $j \neq y_i$, it becomes:

$$
\frac{\partial L_{\text{CCE}}}{\partial z_{ij}} = \sum_{j=1}^{C} \frac{\partial L_{\text{CCE}}}{\partial p(j|x_i)} \frac{\partial p(y_i|x_i)}{z_{ij}}
= p(j|x_i).
$$

(B.12)

In summary, $\frac{\partial L_{\text{CCE}}}{\partial z_i}$ can be represented as:

$$
\frac{\partial L_{\text{CCE}}}{\partial z_{ij}} = \begin{cases} 
  p(y_i|x_i) - 1, & j = y_i \\
  p(j|x_i), & j \neq y_i.
\end{cases}
$$

(B.13)

$\frac{\partial L_{\text{MAE}}}{\partial z_i}$

The calculation is analogous with that of $\frac{\partial L_{\text{CCE}}}{\partial z_i}$.

According to Eq. (B.8) and Eq. (B.4), if $j = y_i$:

$$
\frac{\partial L_{\text{MAE}}}{\partial z_{iy_i}} = \sum_{j=1}^{C} \frac{\partial L_{\text{MAE}}}{\partial p(j|x_i)} \frac{\partial p(y_i|x_i)}{z_{ij}}
= -p(y_i|x_i)(1 - p(y_i|x_i)).
$$

(B.14)

otherwise ($j \neq y_i$):

$$
\frac{\partial L_{\text{MAE}}}{\partial z_{ij}} = \sum_{j=1}^{C} \frac{\partial L_{\text{MAE}}}{\partial p(j|x_i)} \frac{\partial p(y_i|x_i)}{z_{ij}}
= p(y_i|x_i)p(j|x_i).
$$

(B.15)

In summary, $\frac{\partial L_{\text{MAE}}}{\partial z_i}$ is:

$$
\frac{\partial L_{\text{MAE}}}{\partial z_{ij}} = \begin{cases} 
  p(y_i|x_i)(p(y_i|x_i) - 1), & j = y_i \\
  p(y_i|x_i)p(j|x_i), & j \neq y_i.
\end{cases}
$$

(B.16)
\[ \partial L_{\text{GCE}} / \partial z_i \]

The calculation is based on Eq. (B.10) and Eq. (B.4).

If \( j = y_i \), we have:

\[
\frac{\partial L_{\text{GCE}}}{\partial z_i} = \sum_{j=1}^{C} \frac{\partial L_{\text{GCE}}}{\partial p(j|x_i)} \frac{\partial p(y_i|x_i)}{z_{ij}} = p(y_i|x_i)^q(p(y_i|x_i) - 1).
\]

(B.17)

If \( j \neq y_i \), it becomes:

\[
\frac{\partial L_{\text{GCE}}}{\partial z_i} = \sum_{j=1}^{C} \frac{\partial L_{\text{GCE}}}{\partial p(j|x_i)} \frac{\partial p(y_i|x_i)}{z_{ij}} = p(y_i|x_i)^q p(j|x_i).
\]

(B.18)

In summary, \( \partial L_{\text{GCE}} / \partial z_i \) can be represented as:

\[
\frac{\partial L_{\text{GCE}}}{\partial z_{ij}} = \begin{cases} 
  p(y_i|x_i)^q(p(y_i|x_i) - 1), & j = y_i \\
  p(y_i|x_i)^q p(j|x_i), & j \neq y_i
\end{cases}
\]

(B.19)

### B.2 Beating Standard Regularisers Under Label Noise

In Table B.1, we compare our proposed DM with other standard ones, i.e., L2 weight decay and Dropout (Srivastava et al., 2014). We set the dropout rate to 0.2 and L2 weight decay rate to \( 10^{-4} \). For DM, we fix \( \beta = 8, \lambda = 0.5 \). DM is better than those standard regularisers and their combinations significantly. DM works best when it is together with L2 weight decay.

Table B.1: Results of DM and other standard regularisers on CIFAR-100. We set \( r = 40\% \), i.e., the label noise is severe but not belongs to the majority. We train ResNet-44. We report the average test accuracy and standard deviation (%) over 5 trials. Baseline is CCE without regularisation.

<table>
<thead>
<tr>
<th>Baseline</th>
<th>L2</th>
<th>Dropout</th>
<th>Dropout+L2</th>
<th>DM</th>
<th>DM+L2</th>
<th>DM+Dropout</th>
<th>DM+L2+Dropout</th>
</tr>
</thead>
<tbody>
<tr>
<td>44.7±0.1</td>
<td>51.5±0.4</td>
<td>46.7±0.5</td>
<td>52.8±0.4</td>
<td>55.7±0.3</td>
<td><strong>59.3±0.2</strong></td>
<td>54.3±0.4</td>
<td>58.3±0.3</td>
</tr>
</tbody>
</table>
B.3 Small-scale Fine-grained Visual Categorisation of Vehicles

How does DM perform on small datasets, for example, the number of data points is no more than 5,000? We have tested DM on CIFAR-10 and CIFAR-100 in the main paper. However, both of them contain a training set of 50,000 images.

For this question, we answer it from different perspectives as follows:

1. The problem of label noise on CIFAR-10 and CIFAR-100 in Section 6.4.2 is of similar scale.

   - In Table 6.2, when noise rate is 80% on CIFAR-10, the number of clean training examples is around $50,000 \times 20\% = 5,000 \times 2$. Therefore, this clean set is only two times as large as 5,000. Moreover, the learning process may be interrupted by other noisy data points.

   - In Table 6.1, when noise rate is 60% on CIFAR-100, the number of clean training data points is about $50,000 \times 40\% = 5,000 \times 4$, i.e., four times as large as 5,000.

2. We compare DM with other standard regularisers on a small-scale fine-grained visual categorisation problem in Table B.2.

   Vehicles-10 Dataset. In CIFAR-100 (Krizhevsky, 2009), there are 20 coarse classes, including vehicles 1 and 2. Vehicles 1 contains 5 fine classes: bicycle, bus, motorcycle, pickup truck, and train. Vehicles 2 includes another 5 fine classes: lawn-mower, rocket, streetcar, tank, and tractor. We build a small-scale vehicle classification dataset composed of these 10 vehicles from CIFAR-100. Specifically, the training set contains 500 images per vehicle class while the testing set has 100 images per class. Therefore, the number of training data points is 5,000 in total.

Table B.2: The test accuracy (%) of DM and other standard regularisers on Vehicles-10. We train ResNet-44. Baseline denotes CCE without regularisation. We test two cases: symmetric label noise rate is $r = 40\%$, and clean data $r = 0$.

<table>
<thead>
<tr>
<th>$r$</th>
<th>Baseline</th>
<th>L2</th>
<th>Dropout</th>
<th>Dropout+L2</th>
<th>DM</th>
<th>DM+L2</th>
<th>DM+Dropout</th>
<th>DM+L2+Dropout</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>75.4</td>
<td>76.4</td>
<td>77.9</td>
<td>78.7</td>
<td>83.8</td>
<td>84.4</td>
<td>84.5</td>
<td><strong>84.7</strong></td>
</tr>
<tr>
<td>40%</td>
<td>42.3</td>
<td>44.8</td>
<td>41.6</td>
<td>47.4</td>
<td>45.8</td>
<td>55.7</td>
<td>48.8</td>
<td><strong>58.1</strong></td>
</tr>
</tbody>
</table>
B.4 Experimental Details of Robust Sentiment Analysis of Movie Reviews

We report results on the IMDB dataset of movie reviews (Maas et al., 2011) following (Mesnil et al., 2015). Paragraph Vector (PV-DBOW) is used as a document descriptor (Le & Mikolov, 2014). We train a neural network with one 8-neuron hidden layer and display the results in Table 6.3. We generate symmetric label noise.

IMDB contains 25,000 positive movie reviews and 25,000 negative ones. We follow (Le & Mikolov, 2014; Maas et al., 2011; Mesnil et al., 2015) to split them evenly for training and testing, respectively. PV-DBOW represents every movie review using a fixed-length feature vector. It is a binary classification problem. Our implementation benefits from the codes publicly available at https://github.com/mesnilgr/iclr15 and https://github.com/shaform/experiments/tree/master/caffe_sentiment_analysis.

B.4.1 Label Noise

It is a binary classification problem so that the maximum noise rate that we can generate is 50%. We test three cases: \( r = 0.0, 0.2, 0.4 \).

We choose an exponential distribution variant as an EDF, i.e., \( \lambda = 0 \). Additionally, if \( r = 0.0, \beta > 0 \), and \( \beta < 0 \) otherwise. Specifically:

1. When it is clean, we set \( \beta = 0.9 \) so that larger weights are assigned to harder examples.
2. When noise exists, for \( r = 0.2 \) and \( r = 0.4 \), we set \( \beta = -0.52 \) and \( \beta = -0.33 \), respectively. Therefore, easier examples have larger weights.

Note that the settings of \( \lambda \) and \( \beta \) change a lot because we have only two classes here.

B.4.2 Sample Imbalance

We use all the positive reviews (12500) and randomly sample a small proportion of negative reviews:

1. When P-N Ratio is 10:1, 1250 negative reviews are sampled.
2. When P-N Ratio is 50:1, 250 negative reviews are sampled.

We set \( \lambda = 0.3, \beta = 6.5 \).

B.4.3 Net Architecture and Optimisation Solver

The network architecture is shown in Figure B.1 and its SGD solver is as follows:

```
# configuration of solver.prototxt
```
Is it feasible to correct the labels of noisy training data? In label correction, we replace the original labels by their corresponding predictions at the end of training. The results are shown in Table B.3.
Fig. B.1: A designed neural network with one 8-neuron hidden layer for sentiment analysis on IMDB. It is quite simple and our objective is not to represent the state-of-the-art.

Table B.3: Is it feasible to correct the labels of noisy training data? Our results demonstrate the effectiveness of label correction using DM. When retraining from scratch on the relabelled training data, we do not adjust the hyper-parameters $\beta$ and $\lambda$. Therefore, the reported results of retraining on relabelled datasets are not the optimal. In label correction, the original labels are replaced by their corresponding predictions.

<table>
<thead>
<tr>
<th>Noise Rate $r$</th>
<th>Emphasis Mode</th>
<th>Model</th>
<th>Testing Accuracy (%)</th>
<th>Accuracy on Training Sets (%)</th>
<th>Fitting degree of subsets (%)</th>
<th>Retrain after label correction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CCE</td>
<td></td>
<td>Best</td>
<td>Final</td>
<td>Noisy</td>
<td>Intact</td>
</tr>
<tr>
<td>20%</td>
<td>1/3($\lambda = 0.5$) DM ($\beta = 12$)</td>
<td>89.4</td>
<td>87.8</td>
<td>81.5</td>
<td>95.0</td>
<td>98.8</td>
</tr>
<tr>
<td>0</td>
<td>CCE</td>
<td></td>
<td>Best</td>
<td>Final</td>
<td>Noisy</td>
<td>Intact</td>
</tr>
<tr>
<td>40%</td>
<td>1/2($\lambda = 1$) DM ($\beta = 16$)</td>
<td>84.7</td>
<td>83.3</td>
<td>60.3</td>
<td>88.9</td>
<td>94.8</td>
</tr>
</tbody>
</table>
Table B.4: Results of CCE, DM on CIFAR-10 under noisy labels. For every model, we show its best test accuracy during training and the final test accuracy when training terminates, which are indicated by ‘Best’ and ‘Final’, respectively. We also present the results on corrupted training sets and original intact one. The overlap rate between corrupted and intact sets is \((1 - r)\). When \(\lambda\) is larger, \(\beta\) should be larger for adjusting emphasis variance. The intact training set serves as an indicator of meaningful fitting and we observe that its accuracy is always consistent with the final test accuracy.

<table>
<thead>
<tr>
<th>Noise Rate (r)</th>
<th>Emphasis Mode</th>
<th>Model</th>
<th>Testing Accuracy (%)</th>
<th>Accuracy on Training Sets (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Best</td>
<td>Final</td>
</tr>
<tr>
<td>0</td>
<td>None</td>
<td>CCE</td>
<td>86.5</td>
<td>76.8</td>
</tr>
<tr>
<td>0</td>
<td>0 ((\lambda = 0))</td>
<td>DM ((\beta = 0))</td>
<td>83.5</td>
<td>58.1</td>
</tr>
<tr>
<td>0</td>
<td>1/3((\lambda = 0.5))</td>
<td>DM ((\beta = 12))</td>
<td>84.9</td>
<td>76.4</td>
</tr>
<tr>
<td>20%</td>
<td>1/2((\lambda = 1))</td>
<td>DM ((\beta = 16))</td>
<td>89.4</td>
<td>87.8</td>
</tr>
<tr>
<td></td>
<td>2/3((\lambda = 2))</td>
<td>DM ((\beta = 24))</td>
<td>85.8</td>
<td>85.5</td>
</tr>
<tr>
<td>0</td>
<td>None</td>
<td>CCE</td>
<td>82.8</td>
<td>60.9</td>
</tr>
<tr>
<td>0</td>
<td>0 ((\lambda = 0))</td>
<td>DM ((\beta = 0))</td>
<td>71.8</td>
<td>44.9</td>
</tr>
<tr>
<td>0</td>
<td>1/3((\lambda = 0.5))</td>
<td>DM ((\beta = 12))</td>
<td>78.4</td>
<td>65.6</td>
</tr>
<tr>
<td>0</td>
<td>1/2((\lambda = 1))</td>
<td>DM ((\beta = 16))</td>
<td>85.1</td>
<td>79.9</td>
</tr>
<tr>
<td>0</td>
<td>2/3((\lambda = 2))</td>
<td>DM ((\beta = 20))</td>
<td>84.7</td>
<td>83.3</td>
</tr>
<tr>
<td>40%</td>
<td>None</td>
<td>CCE</td>
<td>69.5</td>
<td>37.2</td>
</tr>
<tr>
<td>40%</td>
<td>0 ((\lambda = 0))</td>
<td>DM ((\beta = 0))</td>
<td>69.9</td>
<td>57.9</td>
</tr>
<tr>
<td>40%</td>
<td>1/3((\lambda = 0.5))</td>
<td>DM ((\beta = 12))</td>
<td>72.3</td>
<td>53.9</td>
</tr>
<tr>
<td>40%</td>
<td>1/2((\lambda = 1))</td>
<td>DM ((\beta = 12))</td>
<td>77.5</td>
<td>58.5</td>
</tr>
<tr>
<td>40%</td>
<td>2/3((\lambda = 2))</td>
<td>DM ((\beta = 20))</td>
<td>71.9</td>
<td>70.0</td>
</tr>
<tr>
<td>60%</td>
<td>None</td>
<td>CCE</td>
<td>36.1</td>
<td>16.1</td>
</tr>
<tr>
<td>60%</td>
<td>0 ((\lambda = 0))</td>
<td>DM ((\beta = 0))</td>
<td>44.4</td>
<td>28.2</td>
</tr>
<tr>
<td>60%</td>
<td>1/3((\lambda = 0.5))</td>
<td>DM ((\beta = 12))</td>
<td>46.2</td>
<td>21.3</td>
</tr>
<tr>
<td>60%</td>
<td>1/2((\lambda = 1))</td>
<td>DM ((\beta = 12))</td>
<td>51.6</td>
<td>22.4</td>
</tr>
<tr>
<td>60%</td>
<td>2/3((\lambda = 2))</td>
<td>DM ((\beta = 12))</td>
<td>35.5</td>
<td>31.5</td>
</tr>
<tr>
<td>80%</td>
<td>None</td>
<td>CCE</td>
<td>33.0</td>
<td>32.8</td>
</tr>
<tr>
<td>80%</td>
<td>0 ((\lambda = 0))</td>
<td>DM ((\beta = 0))</td>
<td>44.4</td>
<td>28.2</td>
</tr>
<tr>
<td>80%</td>
<td>1/3((\lambda = 0.5))</td>
<td>DM ((\beta = 12))</td>
<td>46.2</td>
<td>21.3</td>
</tr>
<tr>
<td>80%</td>
<td>1/2((\lambda = 1))</td>
<td>DM ((\beta = 12))</td>
<td>51.6</td>
<td>22.4</td>
</tr>
<tr>
<td>80%</td>
<td>2/3((\lambda = 2))</td>
<td>DM ((\beta = 12))</td>
<td>35.5</td>
<td>31.5</td>
</tr>
</tbody>
</table>

### B.6 More Detailed Empirical Results

### B.6.1 Training DNNs under label noise

**Practical research question:** What training examples should be focused on and how much more should they be emphasised when training DNNs under label noise?
Our proposal: DM incorporates emphasis mode and variance, and serves as explicit regularisation in terms of example weighting.

Important finding: When label noise rate is higher, we can improve a model’s robustness by moving emphasis mode towards relatively less difficult examples.

B.6.2 Empirical Analysis of DM on CIFAR-10

To empirically understand DM well, we explore the behaviours of DM on CIFAR-10 with $r = 20\%, 40\%, 60\%, 80\%$, respectively. We use ResNet-56 which has larger capacity than ResNet-20.

Design choices. We mainly analyse the impact of different emphasis modes for different noise rates. We explore five emphasis modes: 1) None: $\beta = 0$. There is no emphasis mode since all examples are treated equally; 2) $0$: $\lambda = 0$; 3) $\frac{1}{3}$: $\lambda = 0.5$; 4) $\frac{1}{2}$: $\lambda = 1$; 5) $\frac{2}{3}$: $\lambda = 2$.

We remark that when $\lambda$ is larger, the emphasis mode is higher, leading to relatively easier training data points are emphasised. When emphasis mode changes, emphasis variance changes accordingly. Therefore, to set a proper spread for each emphasis mode, we try four emphasis variances and choose the best one\(^1\) to study the impact of emphasis mode.

Results analysis. We show the results in Table B.4. The intact training set serves as an indicator of meaningful fitting and we observe that its accuracy is always consistent with the final test accuracy. We display the training dynamics in Figure B.2. We summarise our observations as follows:

Fitting and generalisation. We observe that CCE always achieves the best accuracy on corrupted training sets, which indicates that CCE has a strong data fitting ability even if there is severe noise (Zhang et al., 2017b). As a result, CCE has much worse final test accuracy than most models.

Emphasising on harder examples. When there exists label noise, we obtain the worst final test accuracy if emphasis mode is 0, i.e., CCE and DM with $\lambda = 0$. This unveils that in applications where we have to learn from noisy training data, it will hurt the model’s generalisation dramatically if we use CCE or simply focus on harder training data points.

Emphasis mode. When noise rate is 0, 20\%, 40\%, 60\%, and 80\%, we obtain the best final test accuracy when $\lambda = 0$, $\lambda = 0.5$, $\lambda = 1$, $\lambda = 2$, and $\lambda = 2$, respectively. This demonstrates that when noise rate is higher, we can improve a model’s robustness by moving emphasis mode towards relatively less difficult examples with a larger $\lambda$, which is informative in practice.

---

\(^1\)Since there is a large interval between different $\beta$ in our four trials, we deduce that the chosen one is not the optimal. The focus of this work is not to optimize the hyper-parameters. Instead, we focus more on the practical research question: What training examples should be focused on and how much more should they be emphasised when training DNNs under label noise?
B.6 More Detailed Empirical Results

147

(a) $r = 20\%$. (b) $r = 40\%$. (c) $r = 60\%$.

Fig. B.2: The learning dynamics of ResNet-56 on CIFAR-10, i.e., training and testing accuracies along with training iterations. The legend in the top left is shared by all subfigures. ‘xxx: yyy’ means ‘method: emphasis mode’. We have two key observations: 1) When noise rate increases, better generalisation is obtained with higher emphasis mode, i.e., focusing on relatively easier examples; 2) Both overfitting and underfitting lead to bad generalisation. For example, ‘CCE: 0’ fits training data much better than the others while ‘DM: None’ generally fits it unstably or a lot worse. Better viewed in colour.

Emphasis spread. As displayed in Table B.4 and Figures B.3-B.6 in the supplementary material, emphasis variance also matters a lot when fixing emphasis mode, i.e., fixing $\lambda$. For example in Table B.4, when $\lambda = 0$, although focusing on harder examples similarly with CCE, DM can outperform CCE by modifying the emphasis variance. As shown in Figures B.3-B.6, some models even collapse and cannot converge if the emphasis variance is not rational.

B.6.3 Learning Dynamics on Clean Training Data

The learning dynamics on clean training data are displayed in the Figures B.7-B.8.
Fig. B.3: ResNet-56 on CIFAR-10 ($r = 20\%$). From left to right, the results of four emphasis modes $0, \frac{1}{3}, 0.5, \frac{2}{3}$ with different emphasis variances are displayed in each column respectively. When $\lambda$ is larger, $\beta$ should be larger. Specifically:

1) when $\lambda = 0$: we tried $\beta = 0.5, 1, 2, 4$;
2) when $\lambda = 0.5$: we tried $\beta = 4, 8, 12, 16$;
3) when $\lambda = 1$: we tried $\beta = 8, 12, 16, 20$;
4) when $\lambda = 2$: we tried $\beta = 12, 16, 20, 24$.

Fig. B.4: ResNet-56 on CIFAR-10 ($r = 40\%$). From left to right, the results of four emphasis modes $0, \frac{1}{3}, 0.5, \frac{2}{3}$ with different emphasis variances are displayed in each column respectively. When $\lambda$ is larger, $\beta$ should be larger. Specifically:

1) when $\lambda = 0$: we tried $\beta = 0.5, 1, 2, 4$;
2) when $\lambda = 0.5$: we tried $\beta = 4, 8, 12, 16$;
3) when $\lambda = 1$: we tried $\beta = 8, 12, 16, 20$;
4) when $\lambda = 2$: we tried $\beta = 12, 16, 20, 24$. 
B.6 More Detailed Empirical Results

Fig. B.5: ResNet-56 on CIFAR-10 ($r = 60\%$). From left to right, the results of four emphasis modes $0, \frac{1}{3}, \frac{2}{3}, \frac{3}{3}$ with different emphasis variances are displayed in each column respectively. When $\lambda$ is larger, $\beta$ should be larger. Specifically:
1) when $\lambda = 0$: we tried $\beta = 0.5, 1, 2, 4$;
2) when $\lambda = 0.5$: we tried $\beta = 4, 8, 12, 16$;
3) when $\lambda = 1$: we tried $\beta = 8, 12, 16, 20$;
4) when $\lambda = 2$: we tried $\beta = 12, 16, 20, 24$.

Fig. B.6: ResNet-56 on CIFAR-10 ($r = 80\%$). From left to right, the results of four emphasis modes $0, \frac{1}{3}, \frac{2}{3}, \frac{3}{3}$ with different emphasis variances are displayed in each column respectively. When $\lambda$ is larger, $\beta$ should be larger. Specifically:
1) when $\lambda = 0$: we tried $\beta = 0.5, 1, 2, 4$;
2) when $\lambda = 0.5$: we tried $\beta = 4, 8, 12, 16$;
3) when $\lambda = 1$: we tried $\beta = 8, 12, 16, 20$;
4) when $\lambda = 2$: we tried $\beta = 12, 16, 20, 24$. 
Fig. B.7: The training and test accuracies on clean CIFAR-10 along with training iterations. The training labels are clean. We fix $\lambda = 0$ to focus on harder examples while changing emphasis variance controller $\beta$. The backbone is ResNet-20. The results of ResNet-56 are shown in Figure B.8.

Fig. B.8: The training and test accuracies on clean CIFAR-10 along with training iterations. The training labels are clean. We fix $\lambda = 0$ to focus on more difficult examples while changing emphasis variance controller $\beta$. The backbone is ResNet-56. The results of ResNet-20 are shown in Figure B.7.