

Czech Technical University in Prague  
Faculty of Electrical Engineering  
Department of Cybernetics



# **Neural Network Training and Non-Differentiable Objective Functions**

Doctoral Dissertation

*Yash Patel*

Ph.D. programme: Electrical Engineering and Information  
Technology

Branch of study: Artificial Intelligence and Biocybernetics

Supervisor: Prof. Ing. Jiří Matas, Ph.D.

Prague, May 2023

**Dissertation Supervisor:**

Prof. Ing. Jiří Matas, Ph.D.  
Czech Technical University in Prague  
Faculty of Electrical Engineering  
Department of Cybernetics  
Karlovo náměstí 13  
121 35 Prague 2  
Czech Republic

Copyright © May 2023 Yash Patel

# Declaration

I hereby declare I have written this doctoral dissertation independently and quoted all the sources of information used in accordance with methodological instructions on ethical principles for writing an academic dissertation. Moreover, I state that this dissertation has neither been submitted nor accepted for any other degree.

The results presented in this dissertation were achieved during my own research in cooperation with my dissertation supervisor Jiri Matas published in [1]–[8]. During the Ph.D., I also collaborated with several researchers on multiple projects, with R. Manmatha, Srikar Appalaraju published in [9]–[13], with Tomáš Hodaň published in [1], with Giorgos Toliás published in [3], with Milan Šulc published in [8], [14], [15], with Dimosthenis Karatzas published in [5], [8], with Slobodan Đukanović, Tuomas Virtanen published in [6], with Rahaf Aljundi, Nikolay Chumerin, Daniel Olmeda published in [14], with Daniel Barath, Alexander Shekhovtsov, Tong Wei published in [7], with Filip Radenovic, Abhimanyu Dubey, Abhishek Kadian, Todor Mihaylov, Simon Vandenhende, Yi Wen, Vignesh Ramanathan, Dhruv Mahajan published in [16], with Štěpán Šimsa, Michal Uříčář, Ahmed Hamdi, Matěj Kocián, Matyáš Skalický, Antoine Doucet, Mickaël Coustaty published in [8], [15], with Nibal Nayef, Michal Busta, Pinaki Nath Chowdhury, Wafa Khlif, Uma-pada Pal, Jean-Christophe Burie, Cheng-lin Liu, Jean-Marc Ogier published in [5], and with Yusheng Xie, Yi Zhu published in [13].

In Prague, May 2023

.....  
Yash Patel



# Abstract

Many important computer vision tasks are naturally formulated to have a non-differentiable objective. Therefore, the standard, dominant training procedure of a neural network is not applicable since back-propagation requires the gradients of the objective with respect to the output of the model. Most deep learning methods side-step the problem sub-optimally by using a proxy loss for training, which was originally designed for another task and is not tailored to the specifics of the objective. The proxy loss functions may or may not align well with the original non-differentiable objective. An appropriate proxy has to be designed for a novel task, which may not be feasible for a non-specialist. This thesis makes four main contributions toward bridging the gap between the non-differentiable objective and the training loss function. Throughout the thesis, we refer to a loss function as a surrogate loss if it is a differentiable approximation of the non-differentiable objective. Note that we use the terms objective and evaluation metric interchangeably.

First, we propose an approach for learning a differentiable surrogate of a decomposable and non-differentiable evaluation metric. The surrogate is learned jointly with the task-specific model in an alternating manner. The approach is validated on two practical tasks of scene text recognition and detection, where the surrogate learns an approximation of edit distance and intersection-over-union, respectively. In a post-tuning setup, where a model trained with the proxy loss is trained further with the learned surrogate on the same data, the proposed method shows a relative improvement of up to 39% on the total edit distance for scene text recognition and 4.25% on  $F_1$  score for scene text detection.

Second, an improved version of training with the learned surrogate where the training samples that are hard for the surrogate are filtered out. This approach is validated for scene text recognition. It outperforms our previous approach and attains an average improvement of 11.2% on total edit distance and an error reduction of 9.5% on accuracy on several popular benchmarks. Note that the two proposed methods for learning a surrogate and training with the surrogate do not make any assumptions about the task at hand and can be potentially extended to novel tasks.

Third, for recall@k, a non-decomposable and non-differentiable evaluation metric, we propose a hand-crafted surrogate that involves designing differentiable versions of sorting and counting operations. An efficient mixup technique

for metric learning is also proposed that mixes the similarity scores instead of the embedding vectors. The proposed surrogate attains state-of-the-art results on several metric learning and instance-level search benchmarks when combined with training on large batches. Further, when combined with the kNN classifier, it also serves as an effective tool for fine-grained recognition, where it outperforms direct classification methods.

Fourth, we propose a loss function termed Extended SupCon that jointly trains the classifier and backbone parameters for supervised contrastive classification. The proposed approach benefits from the robustness of contrastive learning and maintains the probabilistic interpretation like a soft-max prediction. Empirical results show the efficacy of our approach under challenging settings such as class imbalance, label corruption, and training with little labeled data.

Overall the contributions of this thesis make the training of neural networks more scalable – to new tasks in a nearly labor-free manner when the evaluation metric is decomposable, which will help researchers with novel tasks. For non-decomposable evaluation metrics, the differentiable components developed for the recall@k surrogate, such as sorting and counting, can also be used for creating new surrogates.

Automatic translations of the abstract to the Czech language by Google Translate and ChatGPT are included in the appendix.

# Acknowledgements

I extend gratitude to my advisor, Prof. Jiří Matas for attracting my interest towards doing a Ph.D. I would not be able to pursue my Ph.D. degree without his support, ideas and motivation. I would like to thank the institute, Czech Technical University in Prague, for providing a nourishing research platform.

Many thanks to my parents, Mr. Komal Singh Patel and Mrs. Rajni Patel, and my partner Kristína Cinová for their unconditional love and support, without which my research journey would not have been possible. Thanks also to my sisters Dr. Shivani Bhatnager and Mrs. Priyanka Singh along with my niece Kyra Bhatnager and nephew Amartya Singh for making me explain machine learning to them, it always helps to put my work into different perspectives. Last but not the least, I would like to thank my friends and colleagues at the Visual Recognition Group for feedback, support and engaging research discussions.

This research was supported by Research Center for Informatics (project CZ.02 .1.01/ 0.0/0.0/16\_019/0000765 funded by OP VVV), by the Grant Agency of the Czech Technical University in Prague, grant No. SGS23/ 173/ OHK3/ 3T/ 13, by Project StratDL in the realm of COMET K1 center Software Competence Center Hagenberg, and Amazon Research Award.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Background . . . . .	1
1.2	Limitation of Proxy Loss Functions . . . . .	3
1.3	Contributions . . . . .	7
1.4	Structure of the Thesis . . . . .	9
1.5	Publications . . . . .	10
<b>2</b>	<b>Related Work</b>	<b>12</b>
<b>3</b>	<b>Learning Surrogates via Deep Embedding</b>	<b>14</b>
3.1	Related Work . . . . .	15
3.2	Learning Surrogates via Deep Embedding . . . . .	17
3.2.1	Definition of the Surrogate . . . . .	18
3.2.2	Learning the Surrogate . . . . .	18
3.2.3	Training with the Learned Surrogate . . . . .	19
3.3	Experiments . . . . .	19
3.3.1	Analysing the Learned Surrogates . . . . .	19
3.3.2	Post-Tuning with a Learned Surrogate for ED (LS-ED) . . . . .	22
3.3.3	Post-Tuning with a Learned Surrogate for IoU (LS-IoU) . . . . .	25
3.4	Conclusions . . . . .	29
<b>4</b>	<b>FEDS - Filtered Edit Distance Surrogate</b>	<b>30</b>
4.1	Related Work . . . . .	32
4.2	FEDS: Filtered Edit Distance Surrogate . . . . .	34
4.2.1	Background . . . . .	34
4.2.2	Learning edit distance surrogate . . . . .	35
4.2.3	Robust Training . . . . .	35
4.3	Experiments . . . . .	36
4.3.1	FEDS model . . . . .	36
4.3.2	Scene Text Recognition model . . . . .	37
4.3.3	Training and Testing data . . . . .	38
4.3.4	Implementation details . . . . .	39
4.3.5	Quality of the edit distance surrogate . . . . .	39
4.3.6	Quantitative results . . . . .	40
4.3.7	Qualitative results . . . . .	40

4.4	Conclusions . . . . .	44
<b>5</b>	<b>Recall@k Surrogate Loss with Large Batches and Similarity Mixup</b>	<b>45</b>
5.1	Related work . . . . .	48
5.2	Method . . . . .	50
5.3	Experiments on Retrieval Benchmarks . . . . .	54
5.3.1	Datasets . . . . .	54
5.3.2	Implementation details . . . . .	56
5.3.3	Evaluation . . . . .	58
5.3.4	Effect of hyper-parameters . . . . .	64
5.4	Experiments on Fine-Grained Classification . . . . .	66
5.5	Conclusions . . . . .	69
<b>6</b>	<b>Contrastive Classification and Representation Learning with Probabilistic Interpretation</b>	<b>71</b>
6.1	Related Work . . . . .	73
6.2	Background . . . . .	75
6.2.1	Pairwise Losses . . . . .	75
6.2.2	Cross Entropy and Pairwise Cross Entropy . . . . .	76
6.3	Learning a Classifier Jointly with Representation Learning . . . . .	77
6.4	Extended Supervised Contrastive Learning . . . . .	78
6.5	Experiments . . . . .	79
6.5.1	Datasets . . . . .	80
6.5.2	Methods and Implementation Details . . . . .	80
6.5.3	Fully Supervised Classification . . . . .	81
6.5.4	Classification in Low-Sample Scenario . . . . .	82
6.5.5	Classification under Imbalanced Data . . . . .	83
6.5.6	Classification under Noisy Data . . . . .	83
6.5.7	General Remarks . . . . .	84
6.5.8	Classifier Outputs as Posterior Probabilities . . . . .	84
6.6	Conclusion . . . . .	84
<b>7</b>	<b>Conclusions</b>	<b>86</b>
<b>A</b>	<b>Abstrakt</b>	<b>87</b>
	<b>Bibliography</b>	<b>107</b>

# Chapter 1

## Introduction

Training deep networks by gradient descent on the user-defined objective is not possible when the objective is non-differentiable. Deep learning methods use a proxy loss function as a workaround. A proxy loss is a differentiable function previously designed and used for another task but is not tailored to the specifics of the user-defined objective. The use of proxy loss can empirically lead to a reasonable performance, but it may not align well with the user-defined objective, leading to sub-optimal performance. Often, the goal is to perform well on standard benchmarks where the performance is measured using an evaluation metric. In this thesis, we assume that the evaluation metric captures the user-defined objective, and thus the terms evaluation metric and objective function are interchangeably used.

Examples of such a mismatch between the loss and the objective exist in object detection [17], where the evaluation metric is intersection-over-union. Still, many popular approaches use  $L_n$ -norm as a proxy loss [18], [19]. Another example is scene text recognition [20], [21] where the evaluation metric is edit distance but per-character cross-entropy and CTC [22] are commonly used as a proxy. Similarly, in image retrieval [23], [24] where the benchmarks use recall@k or average precision, many variants of a proxy triplet [25] and margin [26] losses have been studied. The thesis deals with training neural networks using learned or hand-crafted differentiable approximations of the test-time objective function, explicitly focusing on the cases when the evaluation metric is non-differentiable.

### 1.1 Background

Supervised deep learning requires four main components: an annotated dataset, a model, a loss function, and an optimizer. The data is collected in a task-specific manner and contains the annotations required to train for a task, *e.g.*, semantic class labels for image classification [27], bounding boxes with semantic labels for object detection [28], per-pixel semantic labels for segmentation [29], etc. The model typically consists of two components: a backbone

and a prediction module. The backbone of the model is designed to transform the input data to a discriminative representation, *e.g.*, ViT [30] for images, BERT [31] for text, ALBEF [32] for joint vision-language modeling, etc. The current trend is to use a backbone pre-trained on large scale datasets either in a supervised or self-supervised manner [33], [34]. The prediction module on top of these backbones transforms the representation to the required task-specific prediction. The loss function compares the model’s prediction with the ground truth. Thus the design of the loss function needs to take into account the task-specific predictions and ground truth, *e.g.*, cross-entropy loss for image classification [35], smooth- $L_1$  to regress the bounding box coordinates for object detection [18], per-pixel cross-entropy loss for semantic segmentation [36], etc. The exact choice of the optimizer is often a hyper-parameter which is empirically determined along with the learning rate, learning rate schedule, and weight decay. Popular choices are Adam [37], AdamW [38], RMSProp [39], Adagrad [40], Adadelta [41], SGD, etc.

Once these components are chosen, the model is trained to minimize the expectation of a loss on the training data. During feed-forward, the data is fed to the model to obtain the predictions, which are then compared against the ground-truth annotations by the loss function. During back-propagating, the chain rule is employed to obtain the gradients of the loss with respect to the model weights. The optimizer then updates these weights based on the obtained gradients and the learning rate. For the chain rule in backpropagation to work, every module in the model and the loss function are required to be differentiable. The model is usually trained until the loss on the validation set keeps decreasing, *i.e.*, until the model starts over-fitting on the training data.

Once the model is trained, it is evaluated on unseen test data, and the comparison between the predictions and the ground truth is now made using a test-time evaluation metric. The evaluation metric is designed to fulfill task-specific requirements and does not depend on the training process.

With supervised deep learning, the performance on a wide range of computer vision tasks has been pushed to the levels of real-world practical use. The progress has been systematically made by improving each component of supervised deep learning, such as deeper and more powerful model architectures [30], [35], [42], [43] and introduction of large-scale training datasets [27], [44]. At first, the progress was expensive as designing architectures demanded detailed domain expertise, and creating new datasets is costly. Therefore, to decrease the human effort, there has been a substantial effort in automating the process of designing better task-specific architectures [45]–[47] and employing self-supervised methods of learning to reduce the dependence on human-annotated data [48]–[52].

There are numerous reasons why a trained model may perform sub-optimally on a test set. Name a few; first, the trained model can over-fit on the training data and perform poorly on the test data sampled identically and independently from the same distribution. To remedy this issue, several regularization

techniques have been investigated [53]–[55]. Over-fitting and under-fitting often happen due to improper choice of the hyper-parameters such as learning rate, schedule, and weight decay. Several approaches and recommendations have been proposed to mitigate these [56]–[59]. Second, the test set could be from a different distribution leading to a domain gap. There are several papers in the literature on domain adaption to make the models robust to the distribution shift [60]–[69]. Further, the use of backbone models pre-trained on very large scale datasets can lead to good generic representations, which can implicitly bridge the domain gap [16], [33], [34], [70]. Third, the evaluation metric may not be known at the time of training leading to a wrong choice of the loss function.

Another possibility is that the evaluation metric is known but is non-differentiable and thus can not be used during the training. Relatively little attention has been paid to the case where the test-time evaluation metric cannot be directly used as a loss function.

## 1.2 Limitation of Proxy Loss Functions

If the evaluation metric is known at the time of training and is differentiable, it can be directly used as a loss function for training. However, the evaluation metrics are known for many practical problems in computer vision but are non-differentiable. An example of a non-differentiable metric is edit distance, also known as Levenshtein distance, which is computed by counting unit operations of addition, deletion, and substitution necessary to transform one text string into another. Edit distance is a common choice for evaluating scene text recognition methods. This metric is non-differentiable as it has a discrete range and is often implemented using dynamic programming, which makes it infeasible to obtain the gradients. Another example is intersection-over-union between the two bounding boxes for object detection, which can be easily implemented in a differentiable manner for axis-aligned bounding boxes, but the implementation is inconvenient for rotated bounding boxes [71]. Another example is in lossy image compression, where the end user is human. Thus the optimal evaluation metric is a human’s perception of similarity, which is complicated and unknown to be adequately expressed as a mathematical function. Another example includes recall at top-k, a popular metric for evaluating retrieval approaches on open-set datasets. Recall@k is the ratio of the number of positive samples in top-k ranks and the total number of positive samples. Recall at top-k is non-differentiable as it requires sorting and counting operations, both of which are non-differentiable.

Existing approaches for these tasks side-step the issue by using an alternative function as a loss function. In this thesis, we term this as a proxy loss function. A proxy loss function for a task is any function that can be used to attain a reasonable performance but is not tailored for the test-time objective. However, this function may not align well with the test-time evalu-



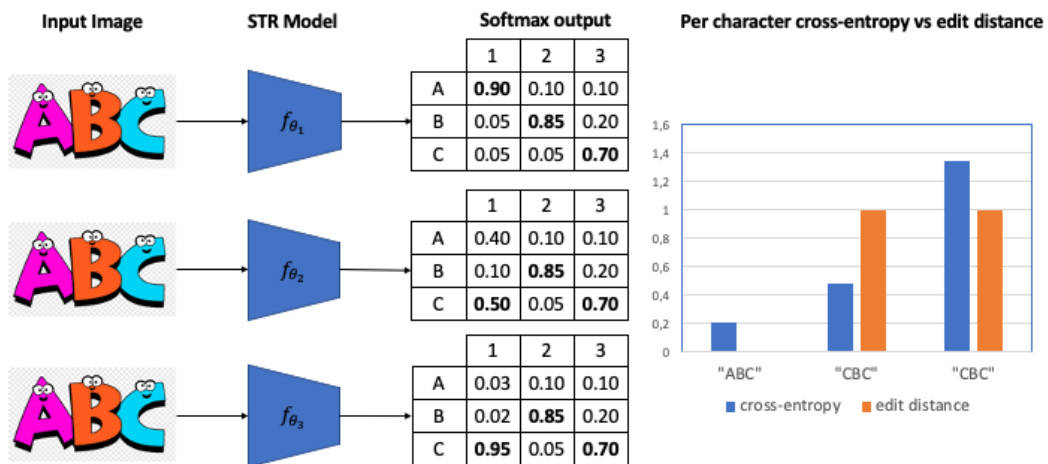


Figure 1.1: Left: Per-character softmax predictions are obtained using three different models for a cropped word image. The final predictions are obtained through argmax on these softmax predictions. Right: for the three predictions, a comparison between the mean of per-character cross-entropy loss and the edit distance is shown. The first prediction is correct and has an edit distance of zero. However, the cross-entropy loss still penalizes the model. The edit distance value for the second and the third predictions is the same. However, the value of cross-entropy is very different.

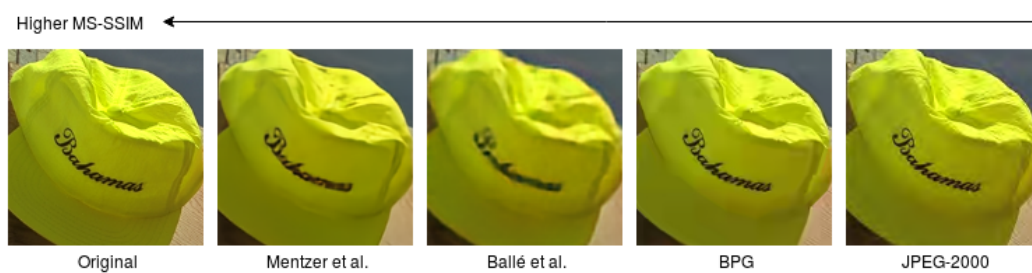


Figure 1.2: An example from the Kodak dataset [72]. In order of MS-SSIM values: Mentzer *et al.*[73] > Ballé *et al.*[74] > BPG [75] > JPEG-2000 [76]. However, the order of performance based on 5 human evaluations is: BPG [75] > Mentzer *et al.*[73] > JPEG-2000 [76] > Ballé *et al.*[74]. Visually the foreground and text in BPG are better in quality.

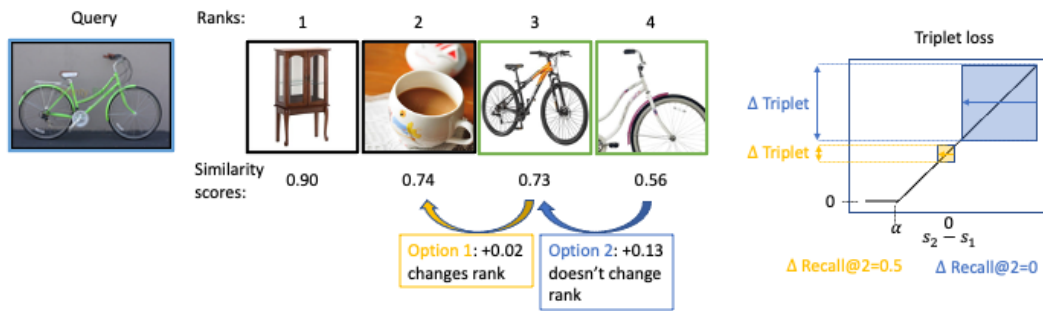


Figure 1.3: Similarity scores to the samples in the image collection are shown for a query image. Consider the change in similarity scores with two options: **Option 1** is a small change in similarity that leads to a change in ranking, **Option 2** is a bigger change in similarity that does not lead to any change in ranks. **Option 1** leads to a change in the value of the evaluation metric, *i.e.*, recall@2 while the proxy loss, *i.e.*, triplet loss has a minor change in value. **Option 2** changes the value of the triplet loss a lot, whereas the value of the evaluation metric does not change at all.

ation metric, leading to a sub-optimal performance during inference. For the above-mentioned examples:

- Scene Text Recognition.** Given an input image of a cropped word, scene text recognition is to predict the transcription of the word. An adequate evaluation metric for the task is the edit distance. Popular methods [77], [78] use either per-character cross-entropy or CTC [22] as the proxy loss function. Note that while the value of the edit distance decreases only when a correct unit operation is being made, the value of per-character cross-entropy may decrease when the probability of predicting the correct character increases. Further, when the prediction is correct, the per-character cross-entropy loss will continue to penalize the model until the correct predictions are very confident. Thus, the proxy loss function, in this case, is harsher on the model than the evaluation metric. The mismatch between the per-character cross-entropy loss and the edit distance is shown in Figure 1.1 through an example.
- Scene Text Detection.** Given a natural scene image, the goal is to precisely localize all instances of text at a word level. The ground truth consists of rotated bounding boxes. Popular approaches use smooth- $L_1$  or  $L_2$  distance for regressing bounding box coordinates. The use of these proxy losses does not have a strong correlation with IoU [79]. As noted by Yu *et al.* [80], IoU accounts for a bounding box as a whole, whereas regressing using an  $L_n$  proxy loss treats each point independently.
- Image compression.** The objective of compression approaches is to

minimize the storage cost of images. In a lossy setting, this is achieved by removing information that is least noticeable to humans. Learning-based compression approaches follow a rate-distortion objective, which tries to minimize the storage requirements while keeping the distortion as low as possible. These approaches use peak-signal-to-noise ratio, mean squared error, or MS-SSIM as the proxy distortion loss function. Through extensive human evaluations, our work shows that PSNR or MS-SSIM do not correlate well with a human’s perception of similarity. In fact, in a binary classification setup, where a human is asked to pick which of two images is closer to the original, these metrics are only slightly better than random predictions [9], [11], [12]. Figure 1.2 shows this through a visual comparison of popular compression techniques.

- **Image Retrieval.** It is a task of ranking all database images according to the relevance to a query. The relevance could be at a semantic or at an instance level. Existing methods for this task use ranking proxy loss functions such as contrastive [81], triplet [25], and margin [26] that pull the samples from the same class closer to one another and push the samples from different class away. As shown in Figure 1.3, the value of a proxy loss function changes with the change in the similarity scores. However, the evaluation metric recall@k for the task only depends on the rank of positive samples in the retrieved list. A small change in the value of the similarity score that changes the rank will cause a big change in the value of the evaluation metric, whereas it will not change the value of the proxy loss substantially [23].
- **Supervised Contrastive Classification.** The goal is to learn representations that are useful for classification. The objective follows contrastive learning where an image, its views obtained via applying augmentations, and other images with the same semantic label are pulled closer to one another, and the samples from a different class are pushed apart. Contrastive supervised classification has shown to be superior and more robust than the use of standard soft-max cross-entropy loss function. With the goal of learning to classify, these approaches follow a two-step training procedure. First, the representations are learned via contrastive training, and then the model is fine-tuned with the cross entropy loss to perform classification. As shown by [82], a simple approach to jointly train the representations and the classifier by combining contrastive loss and cross-entropy loss is sub-optimal.

The above-mentioned non-differentiable evaluation metrics can be categorized into decomposable and non-decomposable. An evaluation metric is decomposable if a per-point evaluation is available, *i.e.*, for a prediction of the model, there is a fixed ground truth, *e.g.*, edit distance, intersection-over-union, and human perception of similarity. If the per-point evaluation is not

possible, *i.e.*, the metric is computed on a set of samples, the evaluation metric is termed non-decomposable, *e.g.*, recall@k, and average precision.

### 1.3 Contributions

The thesis focuses on bridging the gap between the training loss function and the test-time evaluation metric. The proposed solutions vary depending on the nature of the evaluation metric. For decomposable and non-differentiable evaluation metrics such as edit distance, intersection-over-union, and human perception of similarity, we propose to learn a differentiable surrogate of the evaluation metric and train the task-specific model with the learned surrogate [1], [2], [9]. For non-decomposable and non-differentiable evaluation metrics such as recall@k, we resort to a hand-crafted solution [3]. However, the components involved in the developing recall@k surrogate, such as differentiable sorting and differentiable counting, are general and can be used for other evaluation metrics involving these operations as well.

The thesis makes the following contributions:

- A technique for training a neural network by minimizing a surrogate loss that approximates the target evaluation metric, which is decomposable and non-differentiable. The surrogate is learned via a deep embedding where the Euclidean distance between the prediction and the ground truth corresponds to the value of the evaluation metric. The effectiveness of the proposed technique is demonstrated in a post-tuning setup, where a trained model is tuned using the learned surrogate. Without a significant computational overhead and any bells and whistles, improvements are demonstrated on challenging and practical tasks of scene-text recognition and detection. In the recognition task, the model is tuned using a surrogate approximating the edit distance metric and achieves up to 39% relative improvement in the total edit distance. In the detection task, the surrogate approximates the intersection over union metric for rotated bounding boxes and yields up to 4.25% relative improvement in the  $F_1$  score. This work was published in [1] and detailed in Chapter 3.
- A procedure to robustly train a scene text recognition model using a learned surrogate of edit distance. The approach borrows from self-paced learning [83] and filters out the training examples that are hard for the surrogate. The filtering is performed by judging the quality of the approximation using a ramp function, enabling end-to-end training. The experiments are conducted in a post-tuning setup, where a trained scene text recognition model is tuned using the learned surrogate of edit distance. The efficacy is demonstrated by improvements on various challenging scene text datasets such as IIIT-5K [84], SVT [85], ICDAR [86]–[88], SVTP [89], and CUTE [90]. The proposed method provides an

average improvement of 11.2% on total edit distance and an error reduction of 9.5% on accuracy. This work was published in [2] and detailed in Chapter 4.

- A differentiable surrogate of recall at top-k for learning visual representation models for retrieval (see Section 5.2). Since recall@k is a non-decomposable and non-differentiable function, our work relies on hand-crafting the solution.
- An implementation for training with the proposed recall@k surrogate that side-steps the GPU memory constraints and can train up to a batch size of 16k images on a single GPU (see Section 5.3.2).
- An efficient mixup technique that operates on pairwise scalar similarities and virtually increases the batch size further (see Section 5.2). The proposed mixup technique, in theory, is the same as the standard embedding mixup. However, in practice, it is computationally and memory efficient the mixed samples are virtual, and the technique only operates on pairwise similarity scores.
- With synergy between the above three components, our work attains state-of-the-art results on several metric learning benchmarks such as iNaturalist [91], Stanford Online Products [92], Stanford Cars [93], and PUK VehicleID [94]. The same approach also attains state-of-the-art results, for instance-level search on revisited Oxford and Paris [95]. This work was published in [3] and presented in Chapter 5.
- For fine-grained plant recognition, a model trained with the proposed approach and evaluated with kNN classification outperforms the classification approaches trained using the soft-max cross-entropy loss with performance-enhancing techniques such as class prior adaptation, heavy data augmentations, etc.
- A new version of the supervised contrastive training that jointly learns the classifier’s parameters and the network’s backbone. The proposed approach enjoys the robustness of contrastive training while still maintaining probabilistic interpretation like soft-max cross-entropy. The joint training of the backbone and the classifier eliminates the need for two-stage training. We empirically show that our proposed objective functions significantly improve over the standard cross entropy loss with more training stability and robustness in various challenging settings. This work was published in [14] and presented in Chapter 6.
- Additionally, we revisit a previously proposed contrastive-based objective function that approximates cross-entropy loss and present a simple extension to learn the classifier jointly (see Section 6.3).

An additional contribution of our research that is relevant but not included in the thesis is on image compression, where a new end-to-end trainable model for lossy image compression is proposed that includes several novel components. The method incorporates an adequate perceptual similarity metric, saliency in the images, and a hierarchical auto-regressive model. Our work demonstrates that the popularly used evaluation metrics such as MS-SSIM and PSNR are inadequate for judging the performance of image compression techniques as they do not align with the human perception of similarity. Alternatively, a new metric is proposed, which is learned on perceptual similarity data specific to image compression. The proposed compression model incorporates the salient regions and optimizes on the proposed perceptual similarity metric. The model not only generates images that are visually better but also gives superior performance for subsequent computer vision tasks such as object detection and segmentation when compared to existing engineered or learned compression techniques. Note that details of these contributions are excluded from the thesis as the work was done during an internship at AWS-AI. We refer the reader to the related publication [9] for more details.

## 1.4 Structure of the Thesis

The rest of the thesis is organized as follows. Chapter 2 reviews existing methods on surrogate loss functions that attempt to train on non-differentiable objectives. Related work for specific tasks and loss functions are in each subsequent chapter. Chapter 3 presents LS, a method for learning surrogate loss functions for decomposable evaluation metrics. Chapter 4 presents FEDS, an improved approach for learning a surrogate loss function for edit distance via filtering. Chapter 5 presents a hand-crafted surrogate of recall@k, along with an efficient mixup technique. Chapter 6 presents ESUPCON, a loss function for training classification models end-to-end via contrastive learning. The conclusions are made in Chapter 7. For general curiosity, the Czech version of the abstract, translated by Google Translate and ChatGPT, is included in the appendix.

From an application point-of-view, Chapter 3 focuses on scene text recognition and detection, Chapter 4 on scene text recognition, Chapter 5 on metric learning, instance level search, and fine-grained recognition, Chapter 6 on image classification.

## 1.5 Publications

This thesis builds on the results previously published in the following publications:

- Learning Surrogates via Deep Embedding, **Yash Patel**, Tomas Hodan, Jiri Matas. *European Conference on Computer Vision (ECCV) 2020* [1].
- FEDS–Filtered Edit Distance Surrogate, **Yash Patel**, Jiri Matas. *International Conference on Document Analysis and Recognition (ICDAR) 2021* [2].
- Recall@k Surrogate Loss with Large Batches and Similarity Mixup, **Yash Patel**, Giorgos Tolias, Jiri Matas. *IEEE/ CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2022* [3].
- Plant recognition by AI: Deep neural nets, transformers, and kNN in deep embeddings, Lukáš Pícek, Milan Šulc, **Yash Patel** and Jiří Matas. *Frontiers in Plant Science 2022* [4].
- Contrastive Classification and Representation Learning with Probabilistic Interpretation, Rahaf Aljundi, **Yash Patel**, Milan Sulc, Daniel Olmeda Reino, Nikolay Chumerin. *Association for the Advancement of Artificial Intelligence (AAAI) 2023* [14].

The following publications are related to the topic but were not included in the thesis, in order to keep the thesis more focused and easier to follow:

- Saliency driven perceptual image compression, **Yash Patel**, Srikar Appalaraju, R. Manmatha. *IEEE/ CVF Winter Conference on Applications of Computer Vision (WACV) 2021* [9].
- Neural Network-based Acoustic Vehicle Counting, Slobodan Djukanović, **Yash Patel**, Jiří Matas, Tuomas Virtanen. *European Signal Processing Conference (EUSIPCO) 2021* [6].

The following publications were published during the duration of the Ph.D. but are not included in the thesis because they are not directly related to the topic of the thesis:

- ICDAR2019 Robust Reading Challenge on Multi-lingual Scene Text Detection and Recognition–RRC-MLT-2019, Nibal Nayef\*, **Yash Patel\***, Michal Bušta, Pinaki Nath Chowdhury, Dimosthenis Karatzas, Wafa Khelif, Jiri Matas, Umapada Pal, Jean-Christophe Burie, Cheng-lin Liu, Jean-Marc Ogier (\* indicates equal contribution). *International Conference on Document Analysis and Recognition (ICDAR) 2019* [5].

- Filtering, Distillation, and Hard Negatives for Vision-Language Pre-Training, Filip Radenovic, Abhimanyu Dubey, Abhishek Kadian, Todor Mihaylov, Simon Vandenhende, **Yash Patel**, Yi Wen, Vignesh Ramanathan, Dhruv Mahajan. *IEEE/ CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2023* [16].
- DocILE Benchmark for Document Information Localization and Extraction, Štěpán Šimsa, Milan Šulc, Michal Uříčář, **Yash Patel**, Ahmed Hamdi, Matěj Kocián, Matyáš Skalický, Jiří Matas, Antoine Doucet, Mickaël Coustaty, Dimosthenis Karatzas. *International Conference on Document Analysis and Recognition (ICDAR) 2023* [8].

The following publications were not included as they are currently under review:

- Generalized Differentiable RANSAC, Tong Wei, **Yash Patel**, Alexander Shekhovtsov, Jiri Matas, Daniel Barath. *arXiv pre-print 2023* [7].
- Self-Guided Semantic Alignment for Text Supervised Segmentation, **Yash Patel**, Yusheng Xie, Yi Zhu, Srikar Appalaraju, R. Manmatha. *arXiv pre-print 2023* [13].



# Chapter 2

## Related Work

This chapter overviews some of the popular and general approaches for training with an approximation of the non-differentiable evaluation metrics. Specific related work to each task and the evaluation metric are provided in the subsequent chapters.

Due to the non-differentiable nature of many practical evaluation metrics, there exists a large body of proxy loss functions that have been proposed as smooth metric relaxations. Examples include AUCPR loss [96], pairwise AUCROC loss [97], Lovasz-Softmax loss for IoU metric [98], cost-sensitive classification for F-measure [99].

Similar to the focus of our research on learning surrogates, there also have been efforts to learn the loss functions [100], [101]. However, these loss-learning approaches are still based on metric relaxation schemes. Another set of approaches embeds the true evaluation metric as a correction term for optimization [102], [103]. These approaches are limited to the evaluation metrics that are available in a closed form and thus cannot be extended to the evaluation metrics such as edit distance. Our work on learning surrogates [1], [2] follows a simpler approach of directly learning a metric space for approximating the target evaluation metric without making any assumptions about the underlying decomposable evaluation metric.

An approach similar to our work is MetricOpt [104]. This approach optimizes a model on arbitrary non-differentiable evaluation metrics such as misclassification rate and recall. This approach operates in a black-box setting where the computation details of the target metric are unknown and fine-tunes a pre-trained model using an approximation of the evaluation metric. Instead of fine-tuning the entire model, additional adapter parameters are introduced and fine-tuned using the approximation of the evaluation metric. Unlike our work, the approximation of the evaluation metric is learned using a straightforward regression task. Based on empirical comparisons on Standard online products [92] dataset for image retrieval, our hand-crafted recall@k surrogate [3] performs substantially better.

Concurrent to our research is the work of Pogančić *et al.* [105] that im-

plements an efficient backward pass through black box implementations of combinatorial solvers with linear objective functions. This work was extended in [106] for optimizing rank-based evaluation metrics such as recall and average precision, where the efficacy of their approach was shown on metric learning and object detection benchmarks. When empirically compared on the standard benchmarks on the task of metric learning, our proposed approach of recall@k surrogate [3] (Chapter 5) attains substantially better results.

## Chapter 3

# Learning Surrogates via Deep Embedding

For many practical problems in computer vision, models are trained with simple proxy losses, which may not align with the evaluation metric. The evaluation metric may not always be differentiable, prohibiting its use as a loss function. An example of a non-differentiable metric is the visible surface discrepancy (VSD) [107] used to evaluate 6D object pose estimation methods. Another example is the edit distance (ED) defined by counting unit operations (addition, deletion, and substitution) necessary to transform one text string into another and is a common choice for evaluating scene text recognition methods [5], [87], [88]. Since ED is non-differentiable, the methods use either CTC [22] or per-character cross-entropy [77] as the proxy loss. Yet another popular non-differentiable metric is the intersection over union (IoU) used to compare the predicted and the ground truth bounding boxes when evaluating object detection methods. Although these methods typically resort to using proxy losses such as *smooth-L*<sub>1</sub> [108] or *L*<sub>2</sub> [19], Rezatofighi *et al.* [79] demonstrate that there is no strong correlation between *L*<sub>*n*</sub> objectives and IoU. Further, Yu *et al.* [80] show that IoU accounts for a bounding box as a whole whereas regressing using an *L*<sub>*n*</sub> proxy loss treats each point independently.

For popular metrics such as IoU, hand-crafted differentiable approximations have been designed [79], [80]. However, hand-crafting a surrogate is not scalable as it requires domain expertise and may involve task-specific assumptions and simplifications. The IoU-loss introduced in [79], [80] allows for optimization on the evaluation metric directly but makes a strong assumption about the bounding boxes to be axis-aligned. In numerous practical applications such as aerial image object detection [109], scene text detection [88] and visual object tracking [110], the bounding boxes may be rotated and the methods for such tasks revert to using simple but non-optimal proxy loss functions such as *smooth-L*<sub>1</sub> [111]–[113].

To address the aforementioned issues, this chapter proposes to learn a differentiable surrogate that approximates the evaluation metric and use the

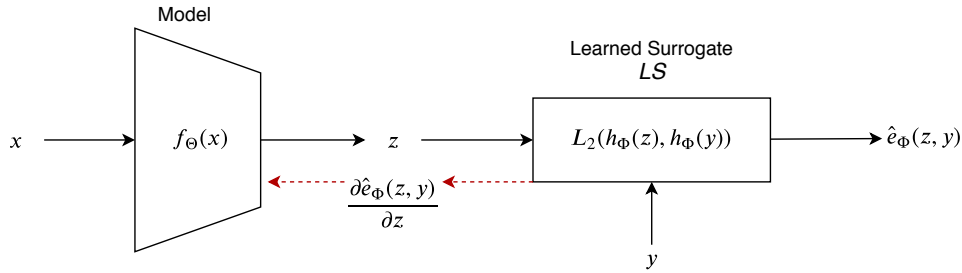


Figure 3.1: For the input  $x$  with the corresponding ground-truth  $y$ , the model being trained outputs  $z = f_{\Theta}(x)$ . The learned surrogate provides a differentiable approximation of the evaluation metric:  $\hat{e}_{\Phi}(z, y) = L_2(h_{\Phi}(z), h_{\Phi}(y))$ , where  $h_{\Phi}$  is a learned deep embedding model, and  $h_{\Phi}(z)$  and  $h_{\Phi}(y)$  are embedding representations for the prediction and the ground truth, respectively. Model  $f_{\Theta}(x)$  for the target task (*e.g.* scene text recognition or detection) is trained with the gradients from the surrogate:  $\frac{\partial(\hat{e}_{\Phi}(z, y))}{\partial z}$ .

learned surrogate to optimize the model for the target task. The metric is approximated via a deep embedding, where the Euclidean distance between the prediction and the ground truth corresponds to the value of the metric. The mapping to the embedding space is realized by a neural network, which is learned using only the value of the metric. Gradients of this value with respect to the inputs are not required for learning the surrogate. In fact, the gradients may not even exist, as is the case of the edit distance metric. Throughout this chapter, we refer to the proposed method for training with learned surrogates as “LS”. Figure 3.1 provides an overview of the proposed method.

In this chapter, the focus is on a post-tuning setup, where a model that has converged on a proxy loss is tuned with LS. We consider two different optimization tasks: post-tuning with a learned surrogate for the edit distance (LS-ED) and the IoU of rotated bounding boxes (LS-IoU). To the best of our knowledge, we are the first to optimize directly on these evaluation metrics.

The rest of the chapter is structured as follows. Related work is reviewed in Section 3.1, the technique for learning the surrogate and training with it is presented in Section 3.2, experiments are shown in Section 3.3 and the chapter is concluded in Section 3.4.

## 3.1 Related Work

Training machine learning models by directly minimizing the evaluation metric has been shown effective on various tasks. For example, the state-of-the-art learned image compression [114], [115] and super-resolution [116], [117] methods directly optimize the perceptual similarity metrics such as MS-SSIM [118] and the peak signal-to-noise ratio (PSNR). Certain compression methods op-

optimize on an approximate of human perceptual similarity, which is learned in a supervised manner using annotated data [10], [11]. Image classification methods [35], [42], [43] are typically trained with the cross-entropy loss that has been shown to align well with the misclassification rate, *i.e.* the evaluation metric, under the assumption of large scale and clean data [119], [120].

When designing evaluation metrics for practical computer vision tasks, the primary goal is to fulfil the requirements of potential applications and not to ensure the metrics being amenable to an optimization approach. As a consequence, many evaluation metrics are non-differentiable and cannot be directly minimized by the currently popular gradient-descent optimization approaches. For example, the visible surface discrepancy [107], which is used to evaluate 6D object pose estimation methods, was designed to be invariant under pose ambiguity. This is achieved by calculating the error only over the visible part of the object surface, which requires a visibility test that makes the metric non-differentiable. Another example is the edit distance metric [88], [121], which is used to evaluate scene text recognition methods and is calculated via dynamic programming, which makes it infeasible to obtain the gradients.

There have been efforts towards approximating non-differentiable operations in a differentiable manner to enable end-to-end training. Kato *et al.* [122] proposed a neural network to approximate rasterization, allowing for a direct optimization on IoU for 3D reconstruction. Agustsson *et al.* [123] proposed a soft-to-hard vector quantization mechanism. It is based on soft cluster assignments during backpropagation, which allows neural networks to learn tasks involving quantization, *e.g.* the image compression. Our work differs as we propose a general approach to approximate the evaluation metric, instead of approximating task-specific building blocks of neural networks.

Another line of research has focused on hand-crafting differentiable approximates of the evaluation metrics, which either align better with the metrics or enable training on them directly. Prabhavalkar *et al.* [124] proposed a way of optimizing attention based speech recognition models directly on word error rate. As mentioned earlier, [79], [80] proposed ways for directly optimizing on intersection-over-union (IoU) as the loss for the case of axis-aligned bounding boxes. Rahman *et al.* [125] proposed a hand-crafted approximation of IoU for semantic segmentation.

Learning task-specific surrogates has been attempted. Nagendra *et al.* [126] demonstrated that learning the approximate of IoU leads to better performance in the case of semantic segmentation. However, the method requires custom operations to estimate true and false positives, and false negatives, which makes the learning approach task-specific. Engilberge *et al.* [127] proposed a learned surrogate for sorting-based tasks such as cross-modal retrieval, multi-label image classification and visual memorability ranking. Their results on sorting-based tasks suggest that learning the loss function could outperform hand-crafted losses.

More closely related to our work is the direct loss method by Hazan *et*

*al.* [102] where a surrogate loss is minimized by embedding the true loss as a correction term. Song *et al.* [103] extended this approach to the training of neural networks. However, it assumes that the loss can be disentangled into per-instance sub-losses, which is not always feasible, *e.g.* the  $F_1$  score [128] involves two non-decomposable functions (recall and precision). An alternative is to directly learn the amount of update values that are applied to the parameters of the prediction model. The framework proposed in [129] includes a controller that uses per-parameter learning curves comprised of the loss values and derivatives of the loss with respect to each parameter. The method suffers from two drawbacks that prohibit its direct application to training on evaluation metrics: a) for large networks, it is computationally infeasible to store the learning curve of every parameter, and b) no gradient information is available for non-differentiable losses.

Our work is similar to the approach by Grabocka *et al.* [128], where the evaluation metric is approximated by a neural network. Their approach differs as the network learning the surrogate takes both the prediction and the ground truth as the input and directly regresses the value of the metric. Since we formulate the task as embedding learning and train the surrogate such that the  $L_2$  in the embedded space corresponds to the metric, our method ensures that the gradients are smaller when the prediction is closer to the ground truth. Furthermore, as illustrated in Section 3.2, we learn the surrogate with an additional gradient penalty term to ensure that the gradients obtained from our learned surrogate are bounded for stable training.

## 3.2 Learning Surrogates via Deep Embedding

Say that the supervised task is being learned from samples drawn uniformly from a distribution  $(x, y) \sim P_D$ . For a given input  $x$  and an expected output  $y$ , a neural network model outputs  $z = f_\Theta(x)$  where  $\Theta$  are the model parameters learned via backpropagation as:

$$\Theta_{t+1} \leftarrow \Theta_t - \eta \frac{\partial l(z, y)}{\partial \Theta_t} \quad (3.1)$$

where  $l(z, y)$  is a differentiable loss function,  $t$  is the training iteration, and  $\eta$  is the learning rate.

The model trained with loss  $l(z, y)$  is evaluated using metric  $e(z, y)$ . When metric  $e(z, y)$  is differentiable, it can be directly used as the loss. The technique proposed in this chapter addresses the cases when metric  $e(z, y)$  is non-differentiable by learning a differentiable surrogate loss denoted as  $\hat{e}_\Phi(z, y)$ . The learned surrogate is realized by a neural network, which is differentiable and is used to optimize the model. The weight updates are:

$$\Theta_{t+1} \leftarrow \Theta_t - \eta \frac{\partial \hat{e}_\Phi(z, y)}{\partial \Theta_t} \quad (3.2)$$

### 3.2.1 Definition of the Surrogate

The surrogate is defined via a learned deep embedding  $h_\Phi$  where the Euclidean distance between the prediction  $z$  and the ground truth  $y$  corresponds to the value of the evaluation metric:

$$\hat{e}_\Phi(z, y) = \|h_\Phi(z) - h_\Phi(y)\|_2 \quad (3.3)$$

### 3.2.2 Learning the Surrogate

Learning the surrogate, *i.e.* approximating the evaluation metric, with a deep neural network is formulated as a supervised learning task requiring three major components: a model architecture, a loss function, and a source of training data.

#### Architecture.

In this chapter, the architecture is designed manually, such that it is suitable for the nature of the inputs  $z$  and  $y$  (details are in Section 3.3). Modern approaches for architecture search, *e.g.* [45]–[47], could yield better results but are computationally expensive.

#### Training Loss.

The surrogate is learned with the following objectives:

1. The learned surrogate corresponds to the value of the evaluation metric:

$$\hat{e}_\Phi(z, y) \approx e(z, y) \quad (3.4)$$

2. The first order derivative of the learned surrogate with respect to the prediction  $z$  is close to 1:

$$\left\| \frac{\partial \hat{e}_\Phi(z, y)}{\partial z} \right\|_2 \approx 1 \quad (3.5)$$

Both objectives are realized and linearly combined in the training loss:

$$\text{loss}(z, y) = \|(\hat{e}_\Phi(z, y) - e(z, y))\|_2^2 + \lambda \left( \left\| \frac{\partial \hat{e}_\Phi(z, y)}{\partial z} \right\|_2 - 1 \right)^2 \quad (3.6)$$

Bounding the gradients (Equation 3.5) has shown to enhance the training stability for Generative Adversarial Networks [130] and has shown to be useful for learning the surrogate. Parameters  $\Phi$  of the embedding model  $h_\Phi$  are learned by minimizing the loss (Equation 3.6).

### Source of Training Data.

Source of the training data for learning the surrogate determines the quality of the approximation over the domain. The model  $f_{\Theta}(x) = z$  for the supervised task is trained on samples obtained from a dataset  $D$ . Let us assume that  $R$  is a random data generator providing examples for the learning of the surrogate, sampled uniformly in the range of the evaluation metric (see Section 3.3 for details). Note that  $R$  is independent of  $f_{\Theta}(x)$ .

Three possibilities for the data source are considered:

1. *Global approximation*:  $(z, y) \sim P_R$ .
2. *Local approximation*:  $(z, y) \sim P_{f_{\Theta}(x)}$ , where  $(x, y) \sim P_D$ .
3. *Local-global approximation*:  $(z, y) \sim P_{f_{\Theta}(x) \cup R}$ .

The local-global approximation yields a high quality of both the approximation and gradients (Section 3.3.1) and is therefore used in the main experiments.

### 3.2.3 Training with the Learned Surrogate

The learned surrogate is used in a post-tuning setup, where model  $f_{\Theta}(x)$  has been pre-trained using a proxy loss. This setup ensures that  $f_{\Theta}(x)$  is not generating random outputs and thus simplifies post-tuning with the surrogate. The parameters of the surrogate  $\Phi$  are initialized randomly.

Learning of the surrogate  $\hat{e}_{\Phi}$  and post-tuning of the model  $f_{\Theta}(x)$  are conducted alternatively. The surrogate parameters  $\Phi$  are updated first while the model parameters  $\Theta$  are fixed. The surrogate is learned by sampling  $(z, y)$  jointly from the model and the random generator. Subsequently, the model parameters are trained while the surrogate parameters are fixed. Algorithm 1 demonstrates the overall training procedure.

## 3.3 Experiments

The efficacy of LS is demonstrated on two different tasks: post-tuning with a learned surrogate for the edit distance (Section 3.3.2) and for the IoU of rotated bounding boxes (Section 3.3.3). This section provides details of the models for these tasks, design choices for learning the surrogates and empirical evidence showing the efficacy of LS. Unless stated otherwise, the results were obtained using the local-global approximation setup as elaborated in Algorithm 1.

### 3.3.1 Analysing the Learned Surrogates

The aspects considered for evaluating the surrogates are:



---

**Algorithm 1** Training with LS (*local-global approximation*)

---

**Inputs:** Supervised data  $D$ , random data generator  $R$ , evaluation metric  $e$ .

**Hyper-parameters:** Number of update steps  $I_a$  and  $I_b$ , learning rates  $\eta_a$  and  $\eta_b$ , number of epochs  $E$ .

**Objective:** Train the model for a given task that is  $f_\Theta(x)$  and the surrogate, *i.e.*,  $e_\Phi$ .

```

1: Initialize  $\Theta \leftarrow$  pre-trained weights,  $\Phi \leftarrow$  random weights.
2: for epoch = 1,...,E do
3:   for i = 1,..., $I_a$  do
4:     sample  $(x, y) \sim P_D$ , sample  $(z_r, y_r) \sim P_R$ 
5:     inference  $z = f_{\Theta^{epoch-1}}(x)$ 
6:     compute loss  $l_{\hat{e}} = \text{loss}(z, y) + \text{loss}(z_r, y_r)$  (Equation 3.6)
7:      $\Phi^i \leftarrow \Phi^{i-1} - \eta_a \frac{\partial l_{\hat{e}}}{\partial \Phi^{i-1}}$ 
8:   end for
9:    $\Phi \leftarrow \Phi^{I_a}$ 
10:  for i = 1,..., $I_b$  do
11:    sample  $(x, y) \sim P_D$ 
12:    inference  $z = f_{\Theta^{i-1}}(x)$ 
13:    compute loss  $l_f = \hat{e}_{\Phi^{epoch}}(z, y)$  (Equation 3.3)
14:     $\Theta^i \leftarrow \Theta^{i-1} - \eta_b \frac{\partial (l_f)}{\partial \Theta^{i-1}}$ 
15:  end for
16:   $\Theta \leftarrow \Theta^{I_b}$ 
17: end for

```

---

1. The quality of approximation  $\hat{e}_\Phi(z, y)$ .
2. The quality of gradients  $\frac{\partial(\hat{e}_\Phi(z, y))}{\partial z}$ .

Both the quality of the approximation and the gradients depend on three components: an architecture, a loss function, and a source of training data (Section 3.2.2). Given an architecture, the choices for the loss function to learn the surrogate and the training data are justified subsequently.

### Quality of approximation.

The quality of the approximation is judged by comparing the value of the surrogate with the value of the evaluation metric, calculated on samples obtained from model  $f_\Theta(x)$ . When learning the surrogate, higher quality of approximation is enforced by the mean squared loss between  $e(z, y)$  and  $\hat{e}_\Phi(z, y)$  (the first term on the right-hand side of Equation 3.6). Figure 3.2 (left) shows the quality of the approximation measured by the  $L_1$  distance between the learned surrogate and the edit distance. It can be seen that the surrogate approximates the edit distance accurately (the  $L_1$  distance drops swiftly below 0.2, which is negligible for the edit distance).

### Quality of gradients.

Judging the quality of gradients is more complicated. When learning the surrogate, the gradient-penalty term attempts to make the gradients bounded, *i.e.* to make the training stable (second term on the right-hand side of the equation 3.6). However, this is not sufficient if the gradients do not optimize  $f_\Theta(x)$  on the evaluation metric. We rely on the improvement or the decline in the performance of the model  $f_\Theta(x)$  to judge the quality of the gradients. Table 3.3 shows that the local-global approximation leads to the largest improvements when optimizing on IoU for rotated bounding boxes.

### Choice of training data.

Figure 3.3 shows the quality of approximation with different choices of training data for learning the surrogate. These empirical observations suggest that using global approximation leads to a low quality of the approximation. This can be accounted to the domain gap between the data obtained from the random generator and the model. Using the local approximation leads to a higher quality of the approximation, however, the gradients obtained from the surrogate are not useful to train  $f_\Theta(x)$  (Table. 3.3), *i.e.* although the quality of the approximation is high, the quality of gradients is not. This can be attributed to surrogate over-fitting on samples obtained from the model and losing generalization capability on samples outside this distribution. Finally, it was observed that using the local-global approximation leads to both properties – high quality of approximation and high quality of gradients.

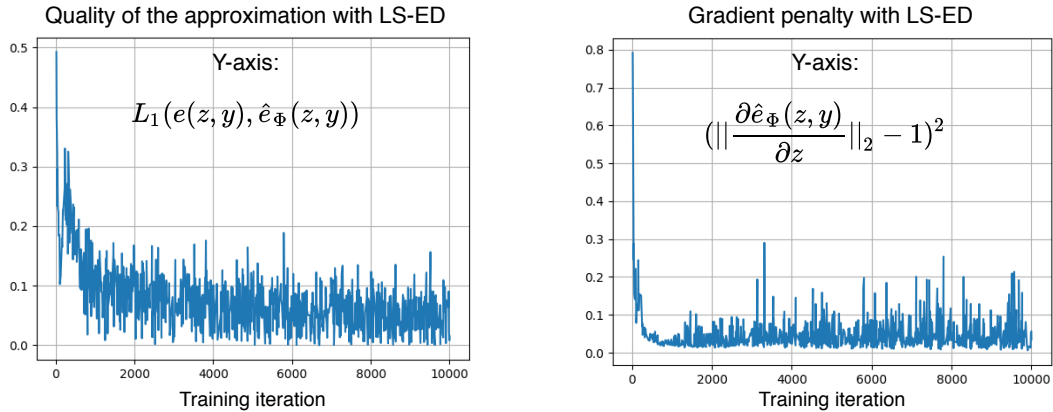


Figure 3.2: **Left:** The error in approximation for the first 10K training iterations. The error is obtained by computing the  $L_1$  distance between the true edit distance values and the LS-ED predictions and dividing by the batch size. Note that the edit distance can only take non-negative integer values, thus the error in the range of 0 – 0.2 is fairly low. **Right:** The gradient penalty term from the optimization of the LS-ED model (Equation 3.6).

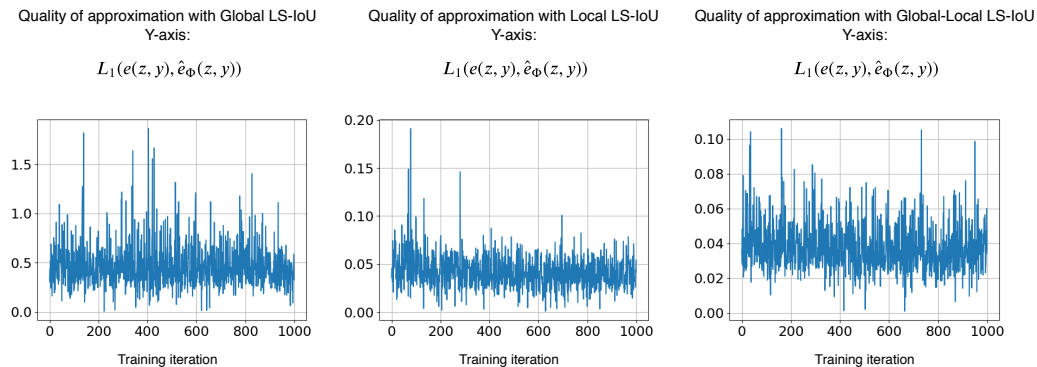


Figure 3.3: The error in the approximation of the IoU for rotated bounding boxes is shown for the first 1K iterations of the training with LS-IoU. Error is measured by the  $L_1$  distance between IoU and the surrogate. It can be seen that the error is high for the global and low for the local and global-local approximation variants.

### 3.3.2 Post-Tuning with a Learned Surrogate for ED (LS-ED)

It is experimentally shown that LS can improve scene text recognition models (STR) on edit distance (ED), which is a popularly used metric to evaluate STR methods [5], [87], [88]. The empirical evidence shows that post-tuning STR models with LS-ED lead to improved performance on various metrics such as

accuracy, normalized edit distance, and total edit distance [121].

### Scene Text Recognition (STR).

Given an input image of a cropped word, the task of STR is to generate the transcription of the word. The state-of-the-art architectures for scene text recognition can be factorized into four modules [77] (in this order): (a) transformation, (b) feature extraction, (c) sequence modelling, and (d) prediction. The feature extraction and prediction are the core modules of any STR model and are always employed. On the other hand, transformation and sequence modelling are not essential but have shown to improve the performance on benchmark datasets. Post-tuning with LS-ED is investigated for two different configurations of STR models.

The transformation module attempts to rectify the curved or tilted text, making the task easier for the subsequent modules of the model. It is learned jointly with the rest of the modules, and a popular choice is thin-plate spline (TPS) [131]–[133]. TPS can be either present or absent in the overall STR model.

The feature extraction module maps the image or its transformed version to a representation that focuses on the attributes relevant for character recognition, while the irrelevant features are suppressed. Popular choices include VGG-16 [43] and ResNet [35]. It is a core module of the STR model and is always present.

The features are the input of the sequence modelling module, which captures the contextual information within a sequence of characters for the next module to predict each character more robustly. BiLSTM [134] is a popular choice.

The output character sequence is predicted from the identified features of the image. The choice of the prediction module depends on the loss function used for training the STR model. Two popular choices of loss functions are CTC [22] (sigmoid output) or attention [131] (per-character softmax output).

Baek *et al.* [77] provides a detailed analysis of STR models and the impact of different modules on the performance. Following [77], LS-ED is investigated with the state-of-the-art performing configuration, which is *TPS-ResNet-BiLSTM-Attn*. To demonstrate the efficacy of LS-ED, results are also shown with *ResNet-BiLSTM-Attn*, *i.e.*, the transformation module is removed. Note that the CTC based prediction has been shown to consistently perform worse compared to the attention counter-part [77], and thus the analysis in this chapter has been narrowed down to only the attention-based prediction.

Similar to [77], the STR models are trained on the union of the synthetic data obtained from MJSynth [135] and SynthText [136] resulting in a total of 14.4 million training examples. Furthermore, following the standard setup of [77], there is no fine-tuning performed in a dataset-specific manner before the final testing. Let us say that the STR model is  $f_{\Theta}(x)$ , such that  $f_{\Theta} : \mathbb{R}^{100 \times 32 \times 1} \rightarrow \mathbb{R}^{|A| \times L}$ . The dimensions of the input cropped word image  $x$  is

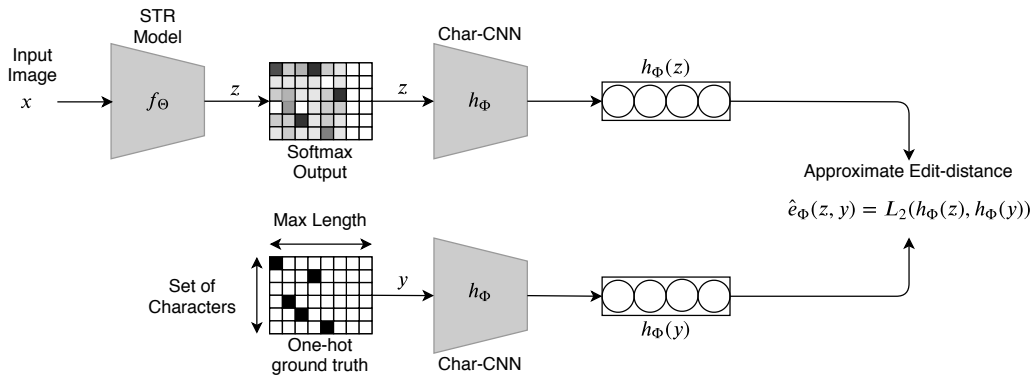


Figure 3.4: Training scene text recognition (STR) models with LS-ED. The output of the STR model  $z_{|A| \times L}$  and the ground-truth  $y_{|A| \times L}$  ( $L$  is the maximum length of the word and  $A$  is the set of characters) are fed to the Char-CNN embedding model to obtain embedding vectors,  $h_\Phi(z)$  and  $h_\Phi(y)$  respectively. The approximate edit distance value is obtained by computing  $\hat{e}_\Phi(z, y) = L_2(h_\Phi(z), h_\Phi(y))$ .

fixed to  $100 \times 32 \times 1$  (gray-scale). The output for attention based prediction module is a per-character softmax over the set of characters. Here  $L$  is the maximum length of characters in the word and  $|A|$  is the number of characters. During inference, argmax is performed at each character location to output the predicted text string. The ground truth  $y$  is represented as a per-character one-hot vector.

The STR models are first trained with the proxy loss, *i.e.*, cross-entropy for  $300K$  iterations with a mini-batch size of 192. The models are optimized using ADADELTA [137] (same setup as [77]). Once the training is completed these models are tuned with LS-ED on the same set of 14.4 million training examples for another  $20K$  iterations. The models trained purely on the synthetic datasets are tested on a collection of real datasets - IIIT-5K [84], SVT [85], ICDAR'03 [86], ICDAR'13 [87], ICDAR'15 [88], SVTP [89] and CUTE [90] datasets.

### LS-ED architecture.

Char-CNN architecture [138] is used for learning the deep embedding  $h_\Phi$ . It consists of five  $1D$  convolution layers equipped with LeakyReLU activation [139] followed by two fully connected layers. The embedding  $h_\Phi$  maps the input such that  $h_\Phi : \mathbb{R}^{|A| \times L} \rightarrow \mathbb{R}^{1024}$ . Note that since  $h_\Phi$  constitutes of convolution and fully-connected layers, it is differentiable and allows for backpropagation to the STR model. In feed-forward, the two embeddings for the ground-truth  $y$  (one-hot) and the model prediction  $z$  (softmax) are obtained by performing feed-forward through  $h_\Phi$  and an approximate of edit distance is computed by

measuring the  $L_2$  between the two vectors (Figure 3.4).

### Post-tuning with LS-ED.

A random generator is designed for this task, which generates a pair of words  $(z_r, y_r)$  and ensures uniform sampling in the range of the true error. It was observed that the uniform sampling is essential to avert over-fitting of the learned surrogate on a certain range of the true metric. For the edit distance metric  $e(z, y) \in \{0, \dots, b\}$  ( $b$  being the maximum possible value), the generator samples a word randomly from a text corpus and distorts the words by performing random addition, deletion, and substitution operations.

The post-tuning of the STR model  $f_{\Theta}(x)$  with LS-ED follows Algorithm 1. For the case of the edit distance, there is a significant domain gap between the samples obtained from the STR model ( $z$ ) and the random generator ( $z_r$ ). This is because the random generator operates directly on the text string, *i.e.*,  $z_r$  is one-hot representation. Thus, using the global approximation setting yields a low quality of the approximation. Further, it was observed that training the surrogate purely with the data generated from the STR model, *i.e.*, local approximation, leads to a good approximation but does not lead to an improvement in the performance of the STR model, which indicates a low quality of gradients.

Finally as described in Algorithm 1, the local-global approximation is used. The quality of approximation and the gradient penalty from post-tuning with LS-ED are shown in Figure 3.2. Note that the edit distance value is a whole number and the surrogate attempts to approximate it, thus the error in approximation as shown in Figure 3.2 is low. The quality of the gradients can be seen by improvement in the performance of the STR models. Thus the local-global approximation guides to a high quality of both the approximation and gradients.

The results for the two configurations of STR models, *i.e.*, *ResNet-BiLSTM-Attn* and *TPS-ResNet-BiLSTM-Attn*, are shown in Table 3.1 and Table 3.2, respectively. It can be observed that LS-ED improves the performance of the STR models on all metrics. The most significant gains are observed on total-edit distance (TED) as the surrogate attempts to minimize its approximation.

### 3.3.3 Post-Tuning with a Learned Surrogate for IoU (LS-IoU)

It is experimentally demonstrated that LS can optimize scene text detection models on intersection-over-union (IoU) for rotated bounding boxes. IoU is a popular metric used to evaluate the object detection [19], [108] and scene text detection models [88], [111], [112], [121], [141]. Gradients for IoU can be hand-crafted for the case of axis-aligned bounding boxes [79], [80], however, it is complex to design the gradients for rotated bounding boxes. The learned

Test Data	Loss Function	↑ Acc.	↑ NED	↓ TED
IIIT-5K	Cross-Entropy	84.300	0.954	945
IIIT-5K	LS-ED	86.300 +2.37%	0.953 -0.10%	837 +11.42%
SVT	Cross-Entropy	84.699	0.940	229
SVT	LS-ED	86.399 +2.00%	0.947 +0.74%	196 +14.41%
ICDAR'03	Cross-Entropy	92.558	0.972	151
ICDAR'03	LS-ED	94.070 +1.63%	0.977 +0.51%	119 +26.89%
ICDAR'13	Cross-Entropy	89.754	0.949	260
ICDAR'13	LS-ED	91.133 +1.53%	0.960 +1.15%	157 +39.61%
ICDAR'15	Cross-Entropy	71.452	0.889	1135
ICDAR'15	LS-ED	74.655 +4.48%	0.899 +1.12%	1013 +10.74%
SVTP	Cross-Entropy	74.109	0.891	424
SVTP	LS-ED	77.519 +4.60%	0.901 +1.22%	381 +10.14%
CUTE	Cross-Entropy	68.293	0.838	285
CUTE	LS-ED	71.777 +5.10%	0.868 +3.57%	234 +17.89%

Table 3.1: ResNet-BiLSTM-Attn: The models are evaluated on IIIT-5K [84], SVT [85], ICDAR'03 [86], ICDAR'13 [87], ICDAR'15 [88], SVTP [89] and CUTE [90] datasets. The results are reported using accuracy **Acc.** (higher is better), normalized edit distance **NED** (higher is better) and total edit distance **TED** (lower is better). Relative gains are shown in green and relative declines in red.

surrogate of IoU allows backpropagation for rotated bounding boxes. For the task of rotated scene text detection on ICDAR'15 [88], it is shown that post-tuning the text detection model with LS-IoU leads to improvement on recall, precision, and  $F_1$  score.

### Scene Text Detection.

Given a natural scene image, the objective is to obtain precise word-level rotated bounding boxes. The method proposed by Ma *et al.* [111] is used for the task. It extends Faster-RCNN [108] based object detector to incorporate rotations. This is achieved by adding angle priors in anchor boxes to enable rotated region proposals. A sampling strategy using IoU compares these proposals with the ground truth and filter the positive and the negative proposals. Only the filtered proposals are used for the loss computation.

The positive proposals are regressed to fit precisely with the ground truth. Through rotated region-of-interest (RROI) pooling, the features corresponding to the proposals are obtained and used for text/no-text binary classification.

Test Data	Loss Function	↑ Acc.	↑ NED	↓ TED
IIIT-5K	Cross-Entropy	87.500	0.961	722
IIIT-5K	LS-ED	87.933 +0.49%	0.963 +0.20%	645 +10.66%
SVT	Cross-Entropy	87.172	0.952	180
SVT	LS-ED	86.708 -0.53	0.954 +0.21%	163 +9.44%
ICDAR'03	Cross-Entropy	94.302	0.979	110
ICDAR'03	LS-ED	94.535 +0.24%	0.981 +0.20%	99 +10.00%
ICDAR'13	Cross-Entropy	92.020	0.966	137
ICDAR'13	LS-ED	92.299 +0.30%	0.979 +1.34%	108 +21.16%
ICDAR'15	Cross-Entropy	78.520	0.915	868
ICDAR'15	LS-ED	78.410 -0.14%	0.915 ±0.00%	837 +3.57%
SVTP	Cross-Entropy	78.605	0.912	346
SVTP	LS-ED	79.225 +0.78%	0.913 +0.10%	333 +3.75%
CUTE	Cross-Entropy	73.171	0.871	224
CUTE	LS-ED	74.216 +1.42%	0.875 +0.45%	219 +2.23%

Table 3.2: TPS-ResNet-BiLSTM-Attn: The models are evaluated on IIIT-5K [84], SVT [85], ICDAR'03 [86], ICDAR'13 [87], ICDAR'15 [88], SVTP [89] and CUTE [90] datasets. The results are reported using accuracy **Acc.** (higher is better), normalized edit distance **NED** (higher is better) and total edit distance **TED** (lower is better). Relative gains are shown in green and relative declines in red.

The overall loss function for training in [111] is defined as a linear combination of classification loss (negative log-likelihood) and regression loss (*smooth-L<sub>1</sub>*).

The publicly available implementation of [111], [140] is used with the original hyper-parameter settings – the model is trained for 140K iterations using the SGD optimizer and batch-size of 1. The model is trained on a union of ICDAR'15 [88] and ICDAR-MLT [5] datasets, providing 6295 training images.

### LS-IoU architecture.

The embedding model for LS-IoU consists of five fully-connected layers with ReLU activation [142]. A rotated bounding box is represented with six parameters, two for the coordinates of the centre of the box, two for the height and the width and two for *cosine* and *sine* of the rotation angle. The centre coordinates and the dimensions of the box are normalized with image dimensions to make the representation invariant to the image resolution.

The embedding model maps the representation of a positive box proposal and the matching ground-truth into a vector as  $h_{\Phi} : \mathbb{R}^6 \rightarrow \mathbb{R}^{16}$ . The approxi-



Loss Function	↑ Recall	↑ Precision	↑ $F_1$ score
<i>Smooth-<math>L_1</math></i>	71.21%	84.71%	77.37%
LS-IoU (global)	66.97% <b>-5.95%</b>	84.71% $\pm 0.00\%$	74.81% <b>-3.30%</b>
LS-IoU (local)	70.92% <b>-0.40%</b>	86.60% <b>+2.23%</b>	77.98% <b>+0.78%</b>
LS-IoU (local-global)	76.79% <b>+7.83%</b>	84.93% <b>+0.25%</b>	80.66% <b>+4.25%</b>

Table 3.3: RRPN-ResNet-50 [111], [140]: Evaluations on Incidental Scene Text ICDAR’15 [88]. Relative gains are shown in green and relative declines in red.

mation of the IoU between two bounding boxes is computed by the  $L_2$  distance between the two vector representations.

### Post-tuning with LS-IoU.

The random generator for LS-IoU samples rotated bounding boxes from the set of training labels and modifies the boxes by changing the centre locations, dimensions, and rotation angle within certain bounds to create a distorted variant. Since uniform sampling over the range of IoU is difficult, we store roughly 3 million such examples along with the IoU values and sample from this collection.

Note that since the overall loss for training [111] is a combination of a regression loss and a classification loss, LS-IoU only replaces the regression component (*smooth- $L_1$* ) with the learned surrogate for IoU. For post-tuning with LS-IoU, the results are shown for all three setups, that is, global approximation, local approximation and global-local approximation (Algorithm 1). For each of these, the model trained with proxy losses is post-tuned with LS-IoU for  $20K$  iterations. The quality of the approximations for the first  $1K$  iterations of the training is shown in Figure 3.3. Since the range of IoU is in  $[0, 1]$ , it can be seen that the error is high for the global approximation. For both local and global-local, the quality of the approximation is significantly better (roughly 10 times lower error).

As mentioned earlier, the quality of gradients is judged by the improvement or deterioration of the model ( $f_{\Theta}(x)$ ) post-tuned with LS-IoU. The results for scene text detection on the ICDAR’15 [88] dataset are shown in Table 3.3. It is observed that post-tuning the detection model with LS-IoU (global) leads to deterioration. Post-tuning with LS-IoU (local) improves the precision but makes recall worse. Finally, LS-IoU (local-global) from Algorithm 1 improves both the precision and recall, boosting the  $F_1$  score by relative 4.25%.

### 3.4 Conclusions

A technique is proposed for training neural networks by minimizing learned surrogates that approximate the target evaluation metric. The effectiveness of the proposed technique has been demonstrated in a post-tuning setup, where a trained model is tuned on the learned surrogate. Improvements have been achieved on the challenging tasks of scene-text recognition and detection. By post-tuning, the model with LS-ED, relative improvements of up to 39% on the total edit distance has been achieved. On detection, post-tuning with LS-IoU has shown to provide a relative gain of 4.25% on the  $F_1$  score.

## Chapter 4

# FEDS - Filtered Edit Distance Surrogate

Deep neural networks are trained by back-propagating gradients [143], which requires the loss function to be differentiable. However, the task-specific objective is often defined via an evaluation metric, which may not be differentiable. The evaluation metric’s design is to fulfill the application requirements, and for the cases where the evaluation metric is differentiable, it is directly used as a loss function. For scene text recognition (STR), accuracy and edit distance are popular evaluation metric choices. Accuracy rewards the method if the prediction exactly matches the ground truth. Whereas edit distance (ED) is defined by counting addition, subtraction, and substitution operations, required to transform one string into another. As shown in Figure 4.1, accuracy does not account for partial correctness. Note that the low *ED* errors from M2 can be easily corrected by a dictionary search in a word-spotting setup [144]. Therefore, edit distance is a better metric, especially when the state-of-the-art is saturated on the benchmark datasets [5], [87], [88].

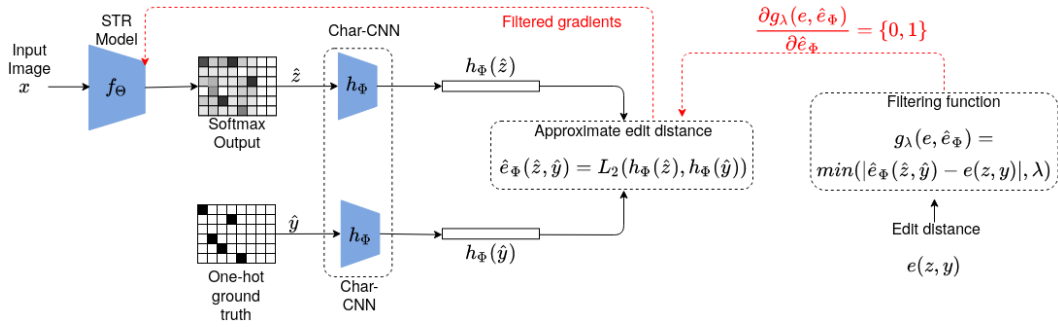


Figure 4.2: Overview of the proposed post-tuning procedure.  $x$  is the input to the STR model  $f_{\Theta}(x)$  with output  $\hat{z}$ .  $y$  is the ground truth,  $\hat{y}$  is the ground truth expressed as one-hot,  $e(z, y)$  is the evaluation metric,  $\hat{e}_{\Phi}(\hat{z}, \hat{y})$  is the learned surrogate and  $g_{\lambda}(e, \hat{e}_{\Phi})$  is the filtering function. The approximations from the learned surrogate are checked against the edit distance by the filtering function. The STR model is not trained on the samples where the surrogate is incorrect.




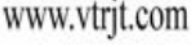


 GT: wwwshutterstockcom M1: wonnishultersock Acc.=0, ED=9 M2: wnwishuttersockcom Acc.=0, ED=3	 GT: lilliput M1: umlout Acc.=0, ED=5 M2: lilidut Acc.=0, ED=2	 GT: frikkie M1: exikive Acc.=0, ED=4 M2: erikkie Acc.=0, ED=1
 GT: wwwvtrjtcom M1: wwwitcom Acc.=0, ED=4 M2: wwwvtritcom Acc.=0, ED=1	 GT: guiucliinc M1: guidglitus Acc.=0, ED=5 M2: guiuclituc Acc.=0, ED=2	 GT: napasorn M1: napasozin Acc.=0, ED=3 M2: napasozn Acc.=0, ED=1
M1: Total Accuracy = 0, Total ED = 30		M2: Total Accuracy = 0, Total ED = 10

Figure 4.1: Accuracy and edit distance comparison for different predictions of scene text recognition (STR). For the scene text images, green shows the ground truth, red shows the prediction from a STR model M1 and blue shows the predictions from another STR model M2. For these examples accuracy ranks both the models equally, however, it can be clearly seen that for the predictions in blue vocabulary search or Google search will succeed.

When the evaluation metric is non-differentiable, a proxy loss is employed, which may not align well with the evaluation metric. Edit distance is computed via dynamic programming and is non-differentiable. Therefore, it can not be used as a loss function for training deep neural networks. The proxy loss used for training STR models is per-character cross-entropy or Connectionist Temporal Classification (CTC) [22]. The models trained with cross-entropy or CTC may have a sub-optimal performance on edit distance as they optimize

a different objective.

The aforementioned issue can be addressed by learning a surrogate, e.g. [1], where a model trained with the proxy loss is post-tuned on a learned surrogate of the evaluation metric. In [1], post-tuning has shown significant improvement in performance on the evaluation metric. While Patel *et al.*[1] have paid attention to learning the surrogate, none was given to robustly train the neural network with the surrogate. In the training procedure, [1] assumes that the learned surrogate robustly estimates the edit distance for all samples. Since the surrogate is learned via supervised training, it is prone to overfitting on the training distribution and may fail on out-of-the-distribution samples. In hope for better generalizability of the surrogate, [1] makes use of a data generator to train the surrogate, which requires extra engineering effort. This chapter shows that the learned edit distance surrogate often fails, leading to noisy training.

As an improvement, this chapter proposes **Filtered Edit Distance Surrogate**. In FEDS, the STR model is trained only on the samples where the surrogate approximates the edit distance within a small error bound. This is achieved by computing the edit distance for a training sample and comparing it with the approximation from the surrogate. The comparison is realized by a ramp-function, which is piece-wise differentiable, allowing for end-to-end training. Figure 4.2 provides an overview of the proposed method. The proposed training method simplifies the training and eliminates the need for a data generator to learn a surrogate.

The rest of the chapter is structured as follows. Related work is reviewed in Section 4.1, the technique for robustly training with the learned surrogate of ED is presented in Section 4.2, experiments are shown in Section 4.3 and the chapter is concluded in Section 4.4.

## 4.1 Related Work

Scene text recognition (STR) is the task of recognizing text from images against complex backgrounds and layouts. STR is an active research area; comprehensive surveys can be found in [77], [145], [146]. Before deep learning, STR methods focused on recognizing characters via sliding window, and hand-crafted features [85], [147], [148]. Deep learning based STR methods have made a significant stride in improving model architectures that can handle both regular (axis-aligned text) and irregular text (complex layout, such as perspective and curved text). Selected relevant methods are discussed subsequently.

**Convolutional models for STR.** Among the first deep learning STR methods was the work of Jaderberg *et al.*[149], where a character-centric CNN [150] predicts a text/no-text score, a character, and a bi-gram class. Later this work was extended to word-level recognition [151] where the CNN takes a fixed dimension input of the cropped word and outputs a word from a fixed dictionary.

Bušta *et al.*[112], [152] proposed a fully-convolutional STR model, which operates on variable-sized inputs using bi-linear sampling [153]. The model is trained jointly with a detector in a multi-task learning setup using CTC [22] loss. Gomez *et al.*[154] trains an embedding model for word-spotting, such that, the euclidean distance between the representations of two images corresponds to the edit-distance between their text strings. This embedding model differs from FEDS as it operates on images instead of STR model’s predictions and is not used to train a STR model.

**Recurrent models for STR.** Shi *et al.*[155] and He *et al.*[156] were among the first to propose end-to-end trainable, sequence-to-sequence models [157] for STR. An image of a cropped word is seen as a sequence of varying length, where convolutional layers are used to extract features and recurrent layers to predict a label distribution. Shi *et al.*[158] later combined the CNN-RNN hybrid with spatial transformer network [153] for better generalizability on irregular text. In [159], Shi *et al.* adapted Thin-Plate-Spline [160] for STR, leading to an improved performance on both regular and irregular text (compared to [158]). While [158], [159] rectify the entire text image, Liu *et al.*[161] detects and rectifies each character. This is achieved via a recurrent RoIWarp layer, which sequentially attends to a region of the feature map that corresponds to a character. Li *et al.*[162] passed the visual features through an attention module before decoding via an LSTM. MaskTextSpotter [163] solves detection and recognition jointly; the STR module consists of two branches while the first uses local visual features, the second utilizes contextual information in the form of attention. Litman *et al.*[78] utilizes a stacked block architecture with intermediate supervision during training, which improves the encoding of contextual dependencies, thus improving the performance on the irregular text.

**Training data.** Annotating scene text data in real images is complex and expensive. As an alternative, STR methods often use synthetically generated data for training. Jaderberg *et al.*[149] generated 8.9 million images by rendering fonts, coloring the image layers, applying random perspective distortion, and blending it to a background. Gupta *et al.*[164] placed rendered text on natural scene images; this is achieved by identifying plausible text regions using depth and segmentation information. Patel *et al.*[165] further extended this to multi-lingual text. The dataset of [164] was proposed for training scene text detection; however, it is also useful for improving STR models [77]. Long *et al.*[166] used a 3D graphics engine to generate scene text data. The 3D synthetic engine allows for better text region proposals as scene information such as normal and objects meshes are available. Their analysis shows that compared to [164], more realistic looking diverse images (contains shadow, illumination variations, etc.) are more useful for STR models. As an alternative to synthetically generate data, Janouskova *et al.*[167] leverages weakly

annotated images to generate pseudo scene text labels. The approach uses an end-to-end scene text model to generate initial labels, followed by a heuristic neighborhood search to match imprecise transcriptions with weak annotations.

As discussed, significant work has been done towards improving the model architectures [77], [78], [112], [151]–[153], [155], [158], [159], [168]–[174] and obtaining data for training [151], [164], [166], [167], [175].

Limited attention has been paid to the loss function. Most deep learning based STR methods rely on per-character cross-entropy or CTC loss functions [22], [77]. While in theory and under an assumption of infinite training data, these loss functions align with accuracy [120], there is no concrete evidence of their alignment with edit-distance. In comparison to the related work, this chapter makes an orthogonal contribution, building upon learning surrogates [1], this chapter proposes a robust training procedure for better optimization of STR models on edit distance.

## 4.2 FEDS: Filtered Edit Distance Surrogate

### 4.2.1 Background

The samples for training the scene text recognition (STR) model are drawn from a distribution  $(x, y) \sim U_D$ . Here,  $x$  is the image of a cropped word, and  $y$  is the corresponding transcription. An end-to-end trainable deep model for STR, denoted by  $f_\Theta(x)$  predicts a soft-max output  $\hat{z} = f_\Theta(x)$ ,  $f_\Theta : \mathbb{R}^{W \times H \times 1} \rightarrow \mathbb{R}^{|A| \times L}$ . Here  $W$  and  $H$  are the dimensions of the input image,  $A$  is the set of characters, and  $L$  is the maximum possible length of the word.

For training, the ground truth  $y$  is converted to one-hot representation  $\hat{y}^{|A| \times L}$ . Cross entropy (CE) is a popular choice of the loss function [77], which provides the loss for each character:

$$CE(\hat{z}, \hat{y}) = -\frac{1}{L|A|} \sum_{i=1}^L \sum_{j=1}^{|A|} \hat{y}_{i,j} \log(\hat{z}_{i,j}) \quad (4.1)$$

Patel *et al.*[1] learns the surrogate of edit distance via a learned deep embedding  $h_\Phi$ , where the Euclidean distance between the prediction and the ground truth corresponds to the value of the edit distance, which provides the edit distance surrogate, denoted by  $\hat{e}_\Phi$ :

$$\hat{e}_\Phi(\hat{z}, \hat{y}) = \|h_\Phi(\hat{z}) - h_\Phi(\hat{y})\|_2 \quad (4.2)$$

where  $h_\Phi$  is the Char-CNN [1], [176] with parameters  $\Phi$ . Note that the edit distance surrogate is defined on the one-hot representation of the ground truth and the soft-max prediction from the STR model.

## 4.2.2 Learning edit distance surrogate

### Objective.

To fairly demonstrate the improvements using the proposed FEDS, the loss for learning the surrogate is the same as LS-ED [1]:

1. The learned edit distance surrogate should correspond to the value of the edit distance:

$$\hat{e}_\Phi(\hat{z}, \hat{y}) \approx e(z, y) \quad (4.3)$$

where  $e(z, y)$  is the edit distance defined on the string representation of the prediction and the ground truth.

2. The first order derivative of the learned edit distance surrogate with respect to the STR model prediction  $\hat{z}$  is close to 1:

$$\left\| \frac{\partial \hat{e}_\Phi(\hat{z}, \hat{y})}{\partial \hat{z}} \right\|_2 \approx 1 \quad (4.4)$$

Bounding the gradients (Equation 4.4) has shown to enhance the training stability for Generative Adversarial Networks [130] and has shown to be useful for learning the surrogate [1].

Both objectives are realized and linearly combined in the training loss:

$$\text{loss}(\hat{z}, \hat{y}) = w_1 \left\| \hat{e}_\Phi(\hat{z}, \hat{y}) - e(z, y) \right\|_2^2 + w_2 \left( \left\| \frac{\partial \hat{e}_\Phi(\hat{z}, \hat{y})}{\partial \hat{z}} \right\|_2 - 1 \right)^2 \quad (4.5)$$

### Training data.

Patel *et al.*[1] uses two sources of data for learning the surrogate - the pre-trained STR model and a random generator. The random generator provides a pair of words and their edit distance and ensures uniform sampling in the range of the edit distance. The random generator helps the surrogate to generalize better, leading to an improvement in the final performance of the STR model.

The proposed FEDS does not make use of a random generator, reducing the effort and the computational cost. FEDS learns the edit distance surrogate only on the samples obtained from the STR model:

$$(\hat{z}, \hat{y}) \sim f_\Theta(x) \mid (x, y) \sim U_D \quad (4.6)$$

## 4.2.3 Robust Training

The filtering function  $g_\lambda$  is defined on the surrogate and the edit distance, parameterized by a scalar  $\lambda$  that acts as a threshold to determine the quality of the approximation from the surrogate. The filtering function is defined as:

$$g_\lambda(e(z, y), \hat{e}_\Phi(\hat{z}, \hat{y})) = \min(|\hat{e}_\Phi(\hat{z}, \hat{y}) - e(z, y)|, \lambda) \mid \lambda > 0 \quad (4.7)$$



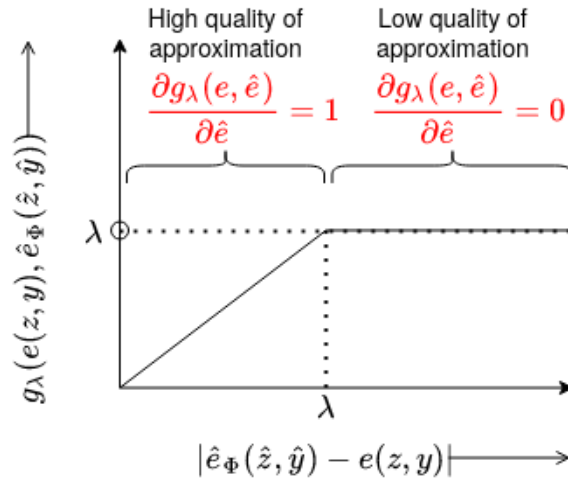


Figure 4.3: The filtering function enforces zero gradients for the samples that are hard for the surrogate (low quality of approximation). STR model is trained only on the samples where the quality of the approximation from the edit distance surrogate is high.

The filtering function is piece-wise differentiable, as can be seen in Figure 4.3. For the samples where the quality of approximation from the surrogate is low, the gradients are zero, and the STR model is not trained on those samples. Whereas for samples where the quality of the approximation is within the bound of  $\lambda$ , the STR model is trained to minimize the edit distance surrogate.

Learning of the ED surrogate  $\hat{e}_\Phi$  and post-tuning of the STR model  $f_\Theta(x)$  are conducted alternatively. The surrogate is learned first for  $I_a$  number of iterations while the STR model is fixed. Subsequently, the STR model is trained using the surrogate and the filtering function, while the ED surrogate parameters are kept fixed. Algorithm 2 and Figure 4.2 demonstrate the overall training procedure with FEDS.

## 4.3 Experiments

### 4.3.1 FEDS model

The model for learning the deep embedding, *i.e.*,  $h_\Phi$  is kept same as [1]. A Char-CNN architecture [176] is used with five 1D convolution layers, LeakyReLU [177] and two FC layers. The embedding model,  $h_\Phi$ , maps the input to a 1024 dimensions,  $h_\Phi : \mathbb{R}^{|A| \times L} \rightarrow \mathbb{R}^{1024}$ . Feed forward (Equation 4.2), generates embeddings for the ground-truth  $\hat{y}$  (one-hot) and model prediction  $\hat{z}$  (soft-max) and an approximation of edit distance is computed by  $L_2$  distance between the two embedding.

**Algorithm 2** Post-tuning with FEDS

---

**Inputs:** Supervised data  $D$ , evaluation metric  $e$ .

**Hyper-parameters:** Number of update steps  $I_a$  and  $I_b$ , learning rates  $\eta_a$  and  $\eta_b$ , number of epochs  $E$ .

**Objective:** Robustly post-tune the STR model, *i.e.*,  $f_{\Theta}(x)$  and learn the edit distance surrogate, *i.e.*,  $\hat{e}_{\Phi}$ .

---

```

1: Initialize  $\Theta \leftarrow$  pre-trained weights,  $\Phi \leftarrow$  random weights.
2: for epoch = 1,...,E do
3:   for i = 1,..., $I_a$  do
4:     sample,  $(x, y) \sim U_D$ 
5:     inference,  $\hat{z} = f_{\Theta^{epoch-1}}(x)$ 
6:     compute loss,  $l_{\hat{e}} = \text{loss}(\hat{z}, \hat{y})$  (Equation 4.5)
7:     update ED surrogate,  $\Phi^i \leftarrow \Phi^{i-1} - \eta_a \frac{\partial l_{\hat{e}}}{\partial \Phi^{i-1}}$ 
8:   end for
9:    $\Phi \leftarrow \Phi^{I_a}$ 
10:  for i = 1,..., $I_b$  do
11:    sample,  $(x, y) \sim U_D$ 
12:    inference,  $\hat{z} = f_{\Theta^{i-1}}(x)$ 
13:    compute ED from the surrogate,  $\hat{e} = \hat{e}_{\Phi^{epoch}}(\hat{z}, \hat{y})$  (Equation 4.2)
14:    compute ED,  $e = e(z, y)$ 
15:    computer loss,  $l_f = g_{\lambda}(e, \hat{e})$  (Equation 4.7)
16:    update STR model,  $\Theta^i \leftarrow \Theta^{i-1} - \eta_b \frac{\partial (l_f)}{\partial \Theta^{i-1}}$ 
17:  end for
18:   $\Theta \leftarrow \Theta^{I_b}$ 
19: end for

```

---

### 4.3.2 Scene Text Recognition model

Following the survey on STR, [77], the state-of-the-art model ASTER is used [159], which contains four modules: (a) transformation, (b) feature extraction, (c) sequence modeling, and (d) prediction. Baek *et al.* [77] provides a detailed analysis of STR models and the impact of different modules on the performance.

#### Transformation.

Operates on the input image and rectifies the curved or tilted text, easing the recognition for the subsequent modules. The two popular variants include Spatial Transformer [153] and Thin Plain Spline (TPS) [159]. TPS employs a smooth spline interpolation between a set of fiducial points, which are fixed in number. Following the analysis of Shi *et al.*[159] [77], the STR model used employs TPS.

**Feature extraction.**

Involves a Convolutional Neural Network [150], that extracts the features from the image transformed by TPS. Popular choices include VGG-16 [43] and ResNet [35]. Following [77], the STR model used employs ResNet for the ease of optimization and good performance.

**Sequence modeling.**

Captures the contextual information within a sequence of characters; this module operates on the features extracted from a ResNet. The STR model used employs BiLSTM [134].

**Prediction.**

The predictions are made based on the identified features of the image. The prediction module depends on the loss function used for training. CTC loss requires the prediction to be by sigmoid, whereas cross-entropy requires the prediction to be a soft-max distribution over the set of characters. The design of FEDS architecture (Section 4.3.1) requires a soft-max distribution.

FEDS and LS-ED [1] are investigated with the state-of-the-art performing configuration of the STR model, which is *TPS-ResNet-BiLSTM-Attn*.

**4.3.3 Training and Testing data**

The STR models are trained on synthetic and pseudo labeled data and are evaluated on real-world benchmarks. Note that the STR models are not fine-tuned on evaluation datasets (same as [77]).

**Training data.**

The experiments make use of the following synthetic and pseudo labeled data for training:

- **MJSynth** [149] (synthetic). 8.9 million synthetically generated images, obtained by rendering fonts, coloring the image layers, applying random perspective distortion, and blending it to a background.
- **SynthText** [164] (synthetic). 5.5 million text instance by placing rendered text on natural scene images. This is achieved by identifying plausible text regions using depth and segmentation information.
- **Uber-Text** [167] (pseudo labels). 138K real images from Uber-Text [178] with pseudo labels obtained using [167].
- **Amazon book covers** [167] (pseudo labels). 1.5 million real images from amazon book covers with pseudo labels obtained using [167].

**Testing data.**

The models trained purely on the synthetic and pseudo labelled datasets are tested on a collection of real datasets. This includes regular scene text - IIIT-5K [84], SVT [85], ICDAR’03 [86] and ICDAR’13 [87], and irregular scene text ICDAR’15 [88], SVTP [89] and CUTE [90].

**4.3.4 Implementation details**

The analysis of the proposed FEDS and LS-ED [1] is conducted for two setups of training data. First, similar to [77], the STR models are trained on the union of the synthetic data obtained from MJSynth [149], and SynthText [164] resulting in a total of 14.4 million training examples. Second, additional pseudo labeled data [167] is used to obtain a stronger baseline.

The STR models are first trained with the proxy loss, *i.e.*, cross-entropy for 300K iterations with a mini-batch size of 192. The models are optimized using ADADELTA [179]. Once the training is complete, these models are tuned with FEDS (Algorithm 2) on the same training set for another 20K iterations. For learning the edit distance surrogate the weights in the loss (Equation 4.5) are set as  $w_1 = 1, w_2 = 0.1$ . Note that the edit distance value is a non-negative integer, therefore, optimal range for  $\lambda$  is  $(0, 0.5)$ . Small value of  $\lambda$  filters out substantial number of samples, slowing down the training, whereas, large values of  $\lambda$  allows a noisy training. Therefore, the threshold for the filtering function (Equation 4.7) is set as  $\lambda = 0.25$ , *i.e.*, in the middle of the optimal range.

**4.3.5 Quality of the edit distance surrogate**

Figure 4.4 shows a comparison between the edit distance and the approximation from the surrogate. As the training progresses, the approximation improves, *i.e.*, more samples are closer to the solid line. The dotted lines represent the filtering in FEDS, *i.e.*, only the samples between the dotted lines contribute to the training of the STR model. Note that the surrogate fails for a large fraction of samples; therefore, the training without the filtering (as done in LS-ED [1]) is noisy.

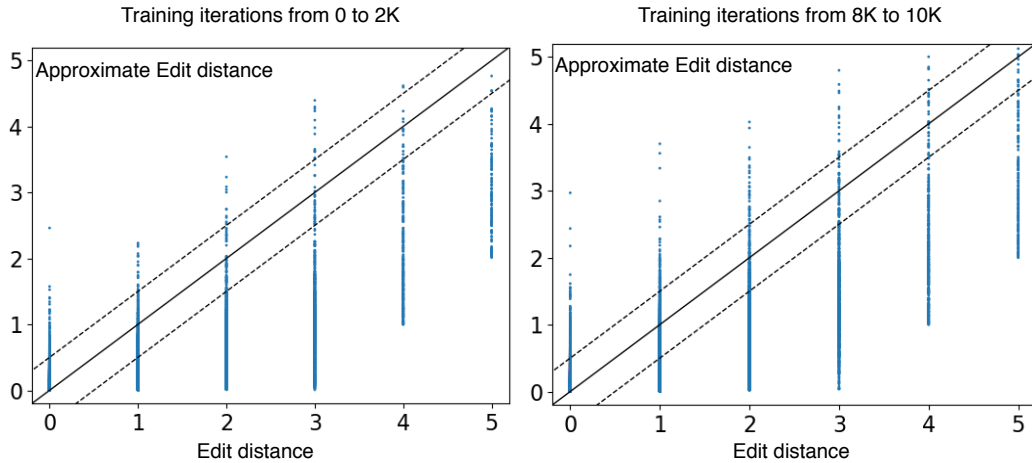


Figure 4.4: A comparison between the true edit distance and the approximated edit distance is shown. Each point represents a training sample for the STR model. The solid line represents an accurate approximation of the edit distance. The dotted lines represent the filtering in FEDS. **Left:** Plot for the first 2K iterations of the STR model training. **Right:** Plot for iterations from 8K to 10K of the STR model training.

### 4.3.6 Quantitative results

Table 4.1 shows the results with LS-ED [1] and the proposed FEDS in compression with the standard baseline [77], [159]. For the training, only the synthetic datasets [149], [164] are used. Both LS-ED [1] and FEDS improve the performance on all evaluation metrics. Most significant gains are observed on total edit distance as the surrogate approximates it. In comparison with LS-ED, significant gains are observed with the proposed FEDS. On average, FEDS provides an improvement of 11.2% on the total edit distance and 0.98% on accuracy (an equivalent of 9.5% error reduction).

Table 4.2 presents the results with LS-ED [1] and FEDS in compression with a stronger baseline [167]. For the training, a combination of synthetic [149], [164] and pseudo labelled [167] data is used. LS-ED [1] provides a limited improvement of 2.91% on total edit distance whereas FEDS provides a significant improvement of 7.90% and an improvement of 1.01% on accuracy (equivalently 7.9% error reduction). Furthermore, LS-ED [1] declines the performance on ICDAR’03 [86] dataset.

### 4.3.7 Qualitative results

Figure 4.5 shows randomly picked qualitative examples where FEDS leads to an improvement in the edit distance. Notice that the predictions from the

Table 4.1: STR model trained with MJSynth [149] and SynthText [164]. Evaluation on IIT-5K [84], SVT [85], IC’03 [86], IC’13 [87], IC’15 [88], SVTP [89] and CUTE [90]. The results are reported using accuracy **Acc.** (higher is better), normalized edit distance **NED** (higher is better) and total edit distance **TED** (lower is better). Relative gains are shown in blue and relative declines in red.

Test Data	Loss Function	↑ Acc.	↑ NED	↓ TED
IIT-5K (3000)	Cross-Entropy [77]	87.1	0.959	772
	LS-ED [1]	88.0 +1.03%	0.962 +0.31%	680 +11.9%
	FEDS	88.8 +1.95%	0.966 +0.72%	591 +23.44%
SVT (647)	Cross-Entropy [77]	87.2	0.953	175
	LS-ED [1]	87.3 +0.11%	0.954 +0.10%	161 +8.00%
	FEDS	88.7 +1.72%	0.957 +0.41%	147 +16.0%
IC’03 (860)	Cross-Entropy [77]	95.1	0.981	105
	LS-ED [1]	95.3 +0.21%	0.982 +0.10%	89 +15.2%
	FEDS	95.4 +0.31%	0.983 +0.20%	87 +17.1%
IC’03 (867)	Cross-Entropy [77]	95.1	0.982	102
	LS-ED [1]	95.2 +0.10%	0.983 +0.10%	90 +11.7%
	FEDS	95.0 -0.10%	0.981 -0.10%	81 +20.5%
IC’13 (857)	Cross-Entropy [77]	92.9	0.979	110
	LS-ED [1]	93.9 +1.07%	0.981 +0.20%	97 +11.8%
	FEDS	93.8 +0.96%	0.985 +0.61%	99 +10.0%
IC’13 (1015)	Cross-Entropy [77]	92.2	0.966	140
	LS-ED [1]	93.1 +0.97%	0.969 +0.31%	123 +12.1%
	FEDS	92.6 +0.43%	0.969 +0.31%	118 +15.7%
IC’15 (1811)	Cross-Entropy [77]	77.9	0.915	880
	LS-ED [1]	78.2 +0.38%	0.915 -	851 +3.29%
	FEDS	78.5 +0.77%	0.919 +0.43%	820 +6.81%
IC’15 (2077)	Cross-Entropy [77]	75.0	0.884	1234
	LS-ED [1]	75.3 +0.39%	0.883 -0.11%	1210 +1.94%
	FEDS	75.7 +0.93%	0.888 +0.45%	1176 +4.70%
SVTP (645)	Cross-Entropy [77]	79.2	0.912	340
	LS-ED [1]	80.0 +1.01%	0.915 +0.32%	327 +3.82%
	FEDS	80.9 +2.14%	0.919 +0.76%	307 +9.70%
CUTE (288)	Cross-Entropy [77]	74.9	0.881	221
	LS-ED [1]	75.6 +0.93%	0.885 +0.45%	204 +7.69%
	FEDS	75.3 +0.53%	0.891 +1.13%	197 +10.8%
TOTAL	Cross-Entropy [77]	85.6	0.941	4079
	LS-ED [1]	86.1 +0.61%	0.942 +0.18%	3832 +6.05%
	FEDS	86.5 +0.98%	0.946 +0.48%	3623 +11.2%

Table 4.2: STR model trained with MJSynth [149], SynthText [164] and pseudo labelled [167] data. Evaluation on IIIT-5K [84], SVT [85], IC’03 [86], IC’13 [87], IC’15 [88], SVTP [89] and CUTE [90]. The results are reported using accuracy **Acc.** (higher is better), normalized edit distance **NED** (higher is better) and total edit distance **TED** (lower is better). Relative gains are shown in **blue** and relative declines in **red**.

Test Data	Loss Function	↑ Acc.	↑ NED	↓ TED
IIIT-5K (3000)	Cross-Entropy [77]	91.7	0.973	550
	LS-ED [1]	91.8 +0.14%	0.973 -	539 +2.00%
	FEDS	92.2 +0.54%	0.975 +0.20%	479 +12.9%
SVT (647)	Cross-Entropy [77]	91.8	0.970	107
	LS-ED [1]	91.8 -	0.971 +0.10%	100 +6.54%
	FEDS	92.1 +0.32%	0.971 +0.10%	102 +4.67%
IC’03 (860)	Cross-Entropy [77]	95.6	0.984	85
	LS-ED [1]	95.5 -0.01%	0.983 -0.10%	91 -7.05%
	FEDS	96.2 +0.62%	0.986 +0.20%	73 +14.1%
IC’03 (867)	Cross-Entropy [77]	95.7	0.984	89
	LS-ED [1]	95.7 +0.03%	0.984 -	91 -2.24%
	FEDS	96.4 +0.73%	0.987 +0.30%	77 +13.4%
IC’13 (857)	Cross-Entropy [77]	95.4	0.988	65
	LS-ED [1]	96.3 +1.03%	0.989 +0.10%	55 +15.3%
	FEDS	96.5 +1.15%	0.989 +0.10%	57 +12.3%
IC’13 (1015)	Cross-Entropy [77]	94.1	0.975	97
	LS-ED [1]	94.8 +0.82%	0.975 -	87 +10.3%
	FEDS	95.3 +1.27%	0.975 -	90 +7.21%
IC’15 (1811)	Cross-Entropy [77]	82.8	0.939	614
	LS-ED [1]	83.2 +0.56%	0.939 -	599 +2.44%
	FEDS	83.8 +1.20%	0.942 +0.31%	578 +5.86%
IC’15 (2077)	Cross-Entropy [77]	80.0	0.908	961
	LS-ED [1]	80.4 +0.54%	0.908 -	944 +1.76%
	FEDS	80.9 +1.12%	0.91 +0.22%	929 +3.32%
SVTP (645)	Cross-Entropy [77]	82.4	0.930	271
	LS-ED [1]	83.4 +1.22%	0.933 +0.32%	258 +4.79%
	FEDS	84.0 +1.94%	0.935 +0.53%	248 +8.48%
CUTE (288)	Cross-Entropy [77]	77.3	0.883	211
	LS-ED [1]	77.3 +0.06%	0.885 +0.22%	197 +6.63%
	FEDS	79.0 +2.19%	0.898 +1.69%	176 +16.5%
TOTAL	Cross-Entropy [77]	88.7	0.953	3050
	LS-ED [1]	89.0 +0.41%	0.954 +0.62%	2961 +2.91%
	FEDS	89.6 +1.01%	0.956 +0.35%	2809 +7.90%

baseline model are incorrect in all the examples. After post-tuning with FEDS, the predictions are correct, *i.e.*, perfectly match with the ground truth.

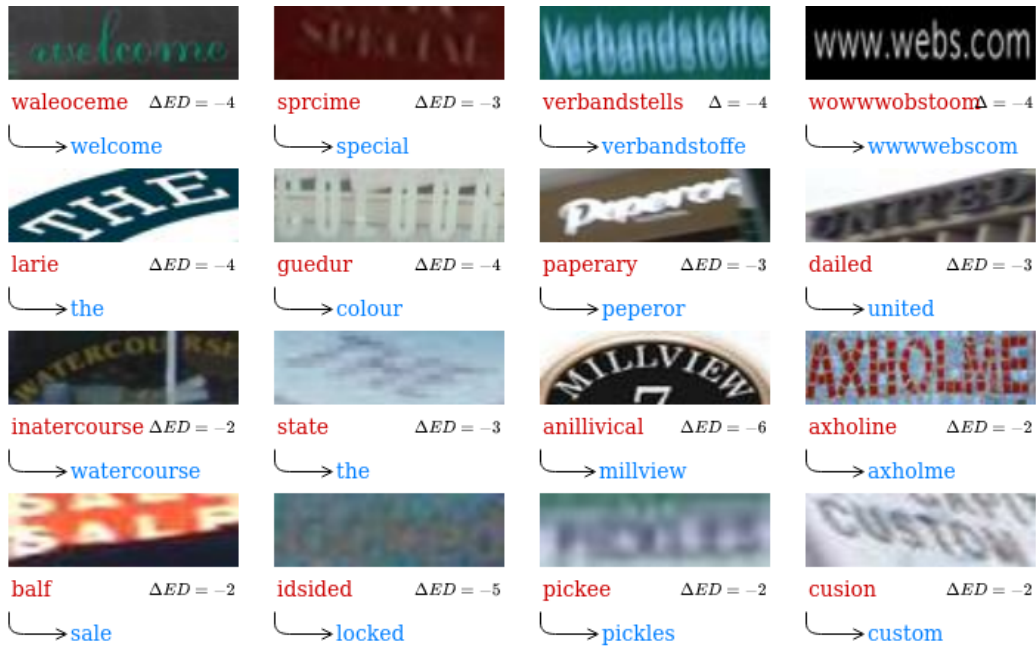


Figure 4.5: Randomly chosen examples from the test set where FEDS improves the STR model trained with cross-entropy. Red shows the incorrect predictions from the baseline model, blue shows the correct prediction after post-tuning with FEDS and the arrow indicates post-tuning with FEDS.

Figure 4.6 shows hand-picked examples where FEDS leads to a maximum increase in the edit distance (ED increases). Notice that in these examples, the predictions from the baseline model are also incorrect. Furthermore, the input images are nearly illegible for a human.

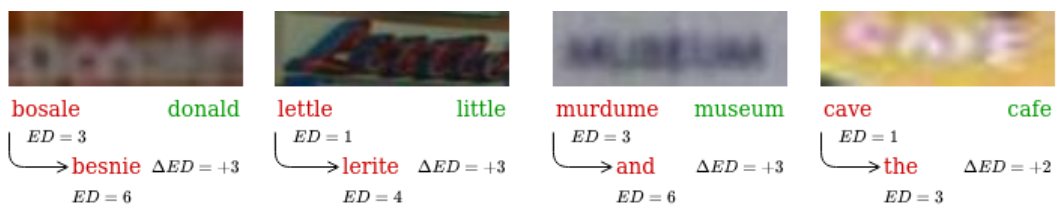


Figure 4.6: Four worst examples out of 12K samples in the test set where FEDS leads to an increase in the edit distance. Red shows the incorrect predictions, green shows the ground truth and the arrow indicates post-tuning with FEDS.



## 4.4 Conclusions

This chapter makes an orthogonal contribution to the trend of scene text recognition progress. It proposes a method to robustly post-tune a STR model using a learned surrogate of edit distance. The empirical results demonstrate an average improvement of 11.2% on total edit distance and an error reduction of 9.5% on accuracy on a standard baseline [77]. Improvements of 7.9% on total edit distance and an error reduction of 10.3% on accuracy are shown on a stronger baseline [167] that uses additional weakly supervised data for training.

# Chapter 5

## Recall@k Surrogate Loss with Large Batches and Similarity Mixup

Minimization of a loss that is a function of the test-time evaluation metric has shown to be beneficial in deep learning for numerous computer vision and natural language processing tasks. Examples include intersection-over-union as a loss that boosts performance for object detection [17], [180] and semantic segmentation [181], and structural similarity [182], peak signal-to-noise ratio [183] and perceptual [184] as reconstruction losses for image compression that give better results according to the respective evaluation metrics.

Training deep networks via gradient descent on the evaluation metric is not possible when the metric is non-differentiable. Deep learning methods resort to a proxy loss, a differentiable function, as a workaround, which empirically leads to a reasonable performance but may not align well with the evaluation metric. Examples exist in object detection [17], scene text recognition [20], [21], machine translation [185] and image retrieval [23], [24].

This chapter deals with the training of image retrieval posed as deep metric learning and Euclidean search in the learned image embedding space. It is the task of ranking all database examples according to the relevance to a query, which is of vital importance for many applications. The standard evaluation metrics are precision and recall in the top retrieved results and the mean Average Precision (mAP). These metrics are standard in information retrieval, they reflect the quality of the retrieved results and allow for flexibility to focus either on the few top results or the whole ranked list of examples, respectively. Recall at top- $k$  retrieved results, denoted by  $recall@k$  in the following, is the primary focus of this work.

The problem related to the optimization of non-differentiable evaluation metrics applies to  $recall@k$  as well. Estimating the position of positive images in the list of retrieved results and counting how many positives appear inside a short-list of a fixed size involves non-differentiable operations. Note that







Query	Ranked Database Images									
	1	2	3	4	5	6	7	8	9	10
										
Similarity:	0.940	0.870	0.850	0.800	0.775	0.650	0.570	0.430	0.400	0.320
recall@4 = 0.33, recall@8 = 0.67										
Similarity:	0.940	0.870	0.850	0.800	0.775	<u>0.774</u>	0.570	0.430	0.400	0.320
recall@4 = 0.33, recall@8 = 0.67										
Similarity:	0.940	0.870	0.850	0.800	<u>0.790</u>	0.775	0.570	0.430	0.400	0.320
recall@4 = 0.33, recall@8 = 0.67										
Similarity:	0.940	<u>0.880</u>	0.870	0.850	<u>0.820</u>	0.800	0.775	0.570	0.430	0.320
recall@4 = 0.67, recall@8 = 1.0										

Figure 5.1: A comparison between recall@k and rs@k, the proposed differentiable recall@k surrogate. Examples show a query, the ranked database images sorted according to the similarity and the corresponding values for recall@k and rs@k and their dependence on similarity score change. Note that the values of recall@k and rs@k are close. Changes to similarity and ranking in some cases may not affect the original recall@k but can affect the surrogate, with the latter having a more significant impact than the former. Similarity values of all negatives are fixed for ease of understanding. The similarity values of the positives that were changed in rows 2, 3 and 4 are underlined.

methods for training on non-differentiable losses, such as actor-critic [185] and learning surrogates [20] are not directly applicable to recall@k. This is due to the fact that these methods are limited to decomposable functions, where a per-example performance measure is available. Such an attempt is made by Engilberge *et al.* [186], where an LSTM learns sorting-based metrics, but is not adapted in consequent work due to slow training. As an alternative, deep metric learning approaches for image retrieval often use ranking proxy losses, termed pairwise losses. In the embedding space, loss functions such as contrastive [81], triplet [25], and margin [26] pull the examples from the same class closer to one another and push the examples from a different class away. These losses are hand-crafted to reflect the objectives of the retrieval task and, consequently, the evaluation metric. The loss value depends on the image-to-image similarity for image pairs or triplets and does not take into account the whole ranked list of examples. Changes in the similarity value without any change in the overall ranking alter the loss value indicate that they are not well correlated with ranking [23]. Recent methods focus on optimizing Average Precision (AP) and use a surrogate function as a loss [23], [187]–[190]. A surrogate of an evaluation metric is a function that approximates it in a differentiable manner.

The proposed method attains state-of-the-art results for 4 fine-grained retrieval datasets, namely iNaturalist [91], VehicleID [91], SOP [92] and Cars196 [93], and 2 instance-level retrieval datasets, namely Revisited Oxford and Paris [95]. This is accomplished by the demonstrated synergy between the three following elements. First, a new loss that is proposed as a surrogate of an established retrieval evaluation metric, namely recall at top  $k$ , and is experimentally shown to consistently outperform existing competitors. A comparison between the evaluation metric and the proposed loss is shown in Figure 5.1. Second, the use of a very large batch size, in the order of several thousand large resolution images on a single GPU. This is inspired by the instance-level retrieval literature [188] and is introduced for the first time in the context of fine-grained categorization. In a recent work of verifying prior results in deep metric learning for fine-grained categorization [191] the batch-size is considered fixed to a single and small value among a large set of comparisons for different losses; in this work we reach batch-sizes that are two orders of magnitude larger than in the work of Musgrave *et al.* [191]. The third element is the proposed mixup regularization technique that is computationally efficient and that virtually enlarges the batch. Its efficiency is obtained by operating on the very last stage of similarity estimation, *i.e.* scalar similarities are mixed, while its applicability goes beyond the combination with the proposed loss in this work. The proposed loss is used for training widely used ResNet architectures [192] but also recent vision-transformers (ViT) [30]. The superiority of this loss compared to existing losses is demonstrated with both architectures, while with ViT-B/16 top results are achieved at lower throughput than with ResNet.

The rest of the chapter is structured as follows: related work specific to

metric learning is provided in Section 5.1, the proposed recall@k surrogate along with the similarity mixup is presented in Section 5.2, experimental results for metric learning, instance-level search, fine-grained recognition and ablation studies are presented in Section 5.3, finally, the conclusions for the proposed contributions of the chapter are made in Section 5.5.

## 5.1 Related work

In this section, the related work is reviewed for two different families of deep metric learning approaches regarding the type of loss that is optimized, namely classification losses and pairwise losses. Given an embedding network that maps input images to a high dimensional space, in the former, the loss is a function of the embedding and the corresponding category label of a single image, while in the latter, the loss is a function of the distance, or similarity, between two embeddings and the corresponding pairwise label. Prior work for mixup [193] techniques related to embedding learning is reviewed too.

**Classification losses.** The work of Zhai and Wu [194] supports that the standard classification loss, *i.e.* cross-entropy (CE) loss is a strong approach for deep metric learning. Their finding is supported by the use of layer normalization and class-balanced sampling. In the domain of metric learning for faces, several different classification losses are proposed, such as SphereFace [195], CosFace [196] and ArcFace [197], where contributions are in the spirit of large margin classification. Despite the specificity of the domain, such losses are applicable beyond faces. Another variant is the Neighborhood Component Analysis (NCA) loss that is used in the work of Movshovitz-Attias *et al.* [198], which is later improved [199] by temperature-based scaling and faster update of the class prototype vectors, also called proxies in their work. The restriction of a single prototype vector per class is dropped by Qian *et al.* [200] who stores multiple representatives per category.

Classification losses, in contrast to pairwise losses, perform the optimization independently per image. An exception is the work of Elezi *et al.* [201] where a similarity propagation module captures group interactions within the batch. Then, cross-entropy loss is used, which now comes with significant improvements by taking into account such interactions. This is recently improved [202] by replacing the propagation module with an attention model. The relation between CE loss and some of the widely used pairwise losses is studied from a mutual information point of view [203]. CE loss is viewed as approximate bound-optimization for minimizing pairwise losses; CE maximizes mutual information, and so do these pairwise losses, which are reviewed in the following.

**Pairwise losses.** The first pairwise loss introduced for this task is the so-called contrastive loss [81], where embeddings of relevant pairs are pushed

as close as possible, while those of non-relevant ones are pushed far enough. Since the target task is typically a ranking one, the triplet loss [25], a popular and widely used loss, improves that by forming training triplets in the form of anchor, positive and negative examples. The loss is a function of the difference between anchor-to-positive and anchor-to-negative distances and is zero if such a difference is large enough, therefore satisfying the objectives of a ranking task for this triplet. Optimization over all pairs or triplets is not tractable and is observed to be sub-optimal [26]. As a result, a lot of attention is paid to finding informative pairs and triplets [191], [204]–[207], which typically includes heuristics. Several other losses are suggested in the literature [26], [205], [208] and are added to the long list of hand-designed proxy losses which target to learn embeddings that transfer well to a ranking or a similar task.

A few cases follow a principled approach for obtaining a loss that is appropriate for ranking tasks. This is the case with the work of Ustinova *et al.* [209] where the goal is to minimize the probability that the similarity between embeddings of a non-relevant pair is larger than that of a relevant one. This probability is approximated by the quantization of the range of possible similarities and the histogram loss, which is estimated within a single batch. Their work dispenses with the need for any kind of sampling for mini-batch construction. An information-theoretic loss function, called RankMI [210], maximizes the mutual information between the samples within the same semantic class using a neural network. Another principled approach focuses on optimizing AP, which is a standard retrieval evaluation metric. A smooth approximation of it is often used in the literature [187]–[189], while the work of Brown *et al.* [23] is the closest to ours. In combination with such AP-based losses, a large batch size is crucial, which meets the limitations set by the hardware. Such limitations are overcome in the work of Revaud *et al.* [188] who uses a batch of 4,000 high-resolution images.

**Embedding mixup.** Manifold mixup [211], which involves mixing [193] intermediate representations and labels of two examples, has demonstrated to improve generalizability for supervised learning by encouraging smoother decision boundaries. Such techniques are investigated for embedding learning and image retrieval by mixing the embedding of two examples. Duan *et al.* [212] uses adversarial training to synthesize additional negative samples from the observed negatives. Kalantidis *et al.* [213] synthesize hard-negatives for contrastive self-supervised learning by mixing the embedding of the two hardest negatives and also mixing them with the query itself. Zheng *et al.* [214] uses a linear interpolation between the embeddings to manipulate the hardness levels. In the work of Gu *et al.* [215], two embedding vectors from the same class are used to generate symmetrical synthetic examples and hard-negative mining is performed within the set of original and the synthetic examples. This is further extended to proxy-based losses, where the embedding of examples from different classes and labels is mixed to generate synthetic proxies [216].

Linearly interpolating labels entails the risk of generating false negatives if the interpolation factor is close to 0 or 1. Such limitations are overcome in the work of Venkataramanan *et al.* [217], which generalizes mixing examples from different classes for pairwise loss functions. The proposed *SiMix* approach differs from the aforementioned techniques as it operates on the similarity scores instead of the embedding vectors, does not require training an additional model, making it computationally efficient. Furthermore, unlike the existing mixup techniques, it uses a synthetic sample in the roles of a query, positive and negative example.

## 5.2 Method

This section presents the task of image retrieval and the proposed approach for learning image embeddings.

**Task.** We are given a query example  $q \in \mathcal{X}$  and a collection of examples  $\Omega \subset \mathcal{X}$ , also called database, where  $\mathcal{X}$  is the space of all images. The set of database examples that are positive or negative to the query are denoted by  $P_q$  and  $N_q$ , respectively, with  $\Omega = P_q \cup N_q$ . Ground-truth information for the positive and negative sets per query is obtained according to discrete class labels per example, *i.e.* if two examples come from the same class, then they are considered positive to each other, otherwise negative. This is the case for all (training or testing) databases used in this work. Terms example and image are used interchangeably in the following text. In image retrieval, all database images are ranked according to similarity to the query  $q$ , and the goal is to rank positive examples before negative ones.

**Deep image embeddings.** Image embeddings, otherwise called descriptors, are generated by function  $f_\theta : \mathcal{X} \rightarrow \mathbb{R}^d$ . In this work, function  $f_\theta$  is a deep fully convolutional neural network or a vision transformer mapping input images of any size or aspect ratio to an  $L_2$ -normalized  $d$ -dimensional embedding. Embedding for image  $x$  is denoted by  $\mathbf{x} = f_\theta(x)$ . Parameter set  $\theta$  of the network is learned during the training. Similarity between a query  $q$  and a database image  $x$  is computed by the dot product of the corresponding embeddings and is denoted by  $s(q, x) = \mathbf{q}^\top \mathbf{x}$ , also denoted as  $s_{qx}$  for brevity.

**Evaluation metric.** Recall@k is one of the standard metrics to evaluate image retrieval methods. For query  $q$ , it is defined as a ratio of the number of relevant (positive) examples within the top-k ranked examples to the total number of relevant examples for  $q$  given by  $|P_q|$ . It is denoted by  $R_\Omega^k(q)$  when computed for query  $q$  and database  $\Omega$  and can be expressed as

$$R_{\Omega}^k(q) = \frac{\sum_{x \in P_q} H(k - r_{\Omega}(q, x))}{|P_q|}, \quad (5.1)$$

where  $r_{\Omega}(q, x)$  is the rank of example  $x$  when all database examples in  $\Omega$  are ranked according to similarity to query  $q$ . Function  $H(\cdot)$  is the Heaviside step function, which is equal to 0 for negative values, otherwise equal to 1. The rank of example  $x$  is computed by

$$r_{\Omega}(q, x) = 1 + \sum_{z \in \Omega, z \neq x} H(s_{qz} - s_{qx}), \quad (5.2)$$

Therefore, (5.1) can now be expressed as

$$R_{\Omega}^k(q) = \frac{\sum_{x \in P_q} H(k - 1 - \sum_{z \in \Omega, z \neq x} H(s_{qz} - s_{qx}))}{|P_q|}. \quad (5.3)$$

**Recall@k surrogate loss.** The computation of recall in (5.3) involves the use of the Heaviside step function. The gradient of the Heaviside step function is a Dirac delta function. Hence, direct optimization of recall with back-propagation is not feasible. A common smooth approximation of the Heaviside step function is provided by the logistic function [218]–[220], a common sigmoid function  $\sigma_{\tau} : \mathbb{R} \rightarrow \mathbb{R}$  controlled by temperature  $\tau$ , which is given by

$$\sigma_{\tau}(u) = \frac{1}{1 + e^{-\frac{u}{\tau}}}, \quad (5.4)$$

where large (small) temperature value leads to worse (better) approximation and denser (sparser) gradient. This approximation is common in the machine learning literature for several tasks [221]–[223] and also appears in the approximation of the Average Precision evaluation metric [23], which is used for the same task as ours. By replacing the step function with the sigmoid function, a smooth approximation of recall is obtained as

$$\tilde{R}_{\Omega}^k(q) = \frac{\sum_{x \in P_q} \sigma_{\tau_1}(k - 1 - \sum_{\substack{z \in \Omega \\ z \neq x}} \sigma_{\tau_2}(s_{qz} - s_{qx}))}{|P_q|}, \quad (5.5)$$

which is differentiable and can be used for training with back-propagation. The two sigmoids have different function domains and, therefore, different temperatures (see Figure 5.2). The minimized single-query loss in a mini-batch  $B$ , with size  $M = |B|$ , and query  $q \in B$  is given by

$$L^k(q) = 1 - \tilde{R}_{B \setminus q}^k(q). \quad (5.6)$$

while incorporation of multiple values of  $k$  is performed in the loss given by

$$L^K(q) = \frac{1}{|K|} \sum_{k \in K} L^k(q). \quad (5.7)$$



Figure 5.3 shows the impact of using single or multiple values for  $k$ .

All examples in the mini-batch are used as queries and the average loss over all queries is minimized during the training. The proposed loss is referred to as *Recall@k Surrogate loss*, or RS@k loss for brevity.

To allow for 0 loss when  $k$  is smaller than the number of positives (note that exact recall@k is less than 1 by definition), we slightly modify (5.5) during the training. Instead of dividing by  $|P_q|$ , we divide by  $\min(k, |P_q|)$ , and, consequently, we clip values larger than  $k$  in the numerator to avoid negative loss values.

**Similarity mixup (SiMix).** Given original batch  $B$ , virtual batch  $\hat{B}$  is created by mixing all pairs of positive examples in the original batch. Embeddings of examples  $x \in B$  and  $z \in B$  are used to generate mixed embedding

$$\mathbf{v}_{xz\alpha} = \alpha \mathbf{x} + (1 - \alpha) \mathbf{z} \quad | \quad \alpha \sim U(0, 1), \quad (5.8)$$

for a virtual example that is denoted by  $xz\alpha \in \hat{B}$ . The similarity of an original example  $w \in B$  to the virtual example  $xz\alpha \in \hat{B}$  is given by

$$s(w, xz\alpha) = \mathbf{w}^\top \mathbf{v}_{xz\alpha} = \alpha s_{wx} + (1 - \alpha) s_{wz}, \quad (5.9)$$

where the original and virtual examples can be the query and database examples, respectively, or vice versa. In case both examples are virtual, *e.g.*  $xz\alpha_1 \in \hat{B}$  used as a query and  $yw\alpha_2 \in \hat{B}$  as a part of the database, then their similarity is given by

$$\begin{aligned} s(xz\alpha_1, yw\alpha_2) &= \mathbf{v}_{xz\alpha_1}^\top \mathbf{v}_{yw\alpha_2} \\ &= \alpha_1 \alpha_2 s_{xy} + (1 - \alpha_1)(1 - \alpha_2) s_{zw} \\ &\quad + \alpha_1(1 - \alpha_2) s_{xw} + (1 - \alpha_1)\alpha_2 s_{zy}. \end{aligned} \quad (5.10)$$

The pairwise similarities that appear on the right-hand side of the previous formulas, *e.g.*  $s_{wx}$  and  $s_{wz}$  in (5.9), are computed from the embeddings of the original, non-virtual examples and are also required for the computation of the RS@k without any virtual examples. Therefore, the mini-batch is expanded to  $B \cup \hat{B}$  by adding virtual examples without the need for explicit construction of the corresponding embeddings or computation of the similarity via dot product; simple mixing of the corresponding pairwise scalar similarities is enough. SiMix reduces to mixing pairwise similarities due to the lack of re-normalization of the mixed embeddings, which is different to existing practice in prior work [213], [215]–[217] and brings training efficiency benefits.

Virtual examples are created only between examples of the same classes and are labeled according to the class of the original examples that are mixed. Virtual examples are used both as queries and as database examples, while mixing is applied to all pairs of positive examples inside a mini-batch.

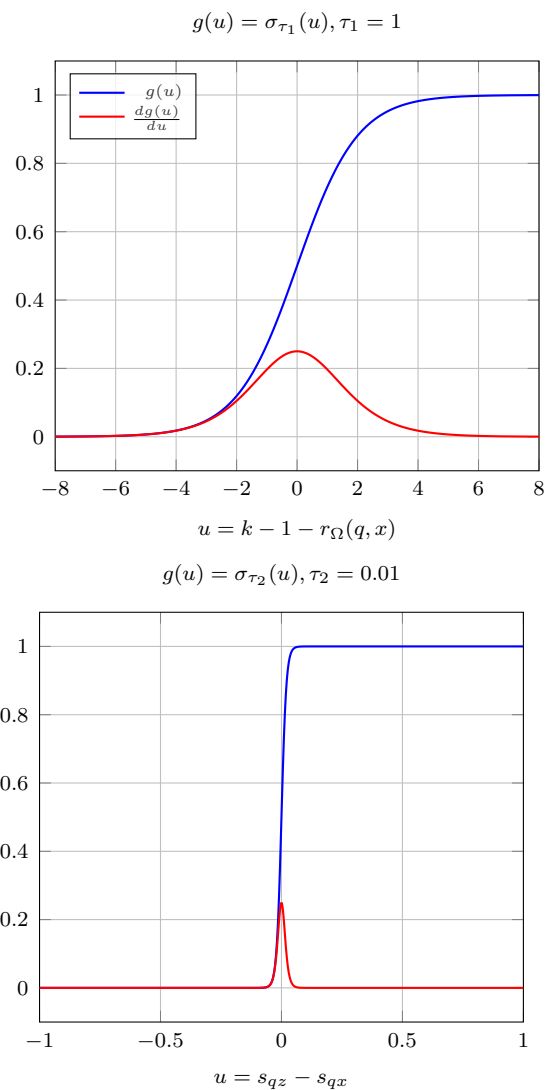


Figure 5.2: The two sigmoid functions which replace the Heaviside step function for counting the positive examples in the short-list of size  $k$  (top) and for estimating the rank of examples (bottom).

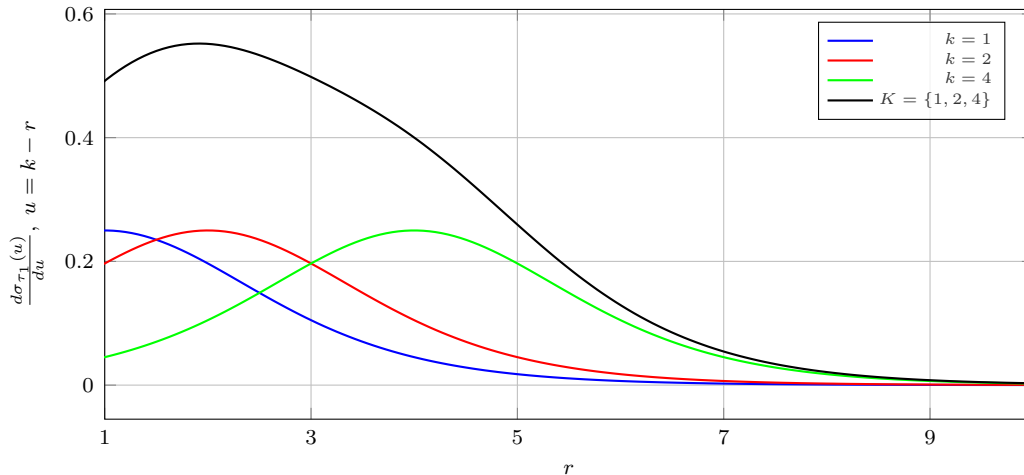


Figure 5.3: Gradient magnitude of the sigmoid used to count the positive examples in the short-list of size  $k$  versus the rank  $r$  (equal to  $r_{\Omega}(q, x)$ , see (5.2)) of a positive example  $x$ . It shows how much a positive example is pushed towards lower ranks depending on its current rank. In the case of multiple values for  $k$ , the total gradient is equivalent to the sum of the separate ones.

**Overview.** An overview of the training process with the proposed loss and SiMix is given in Algorithm 3. In case SiMix is not used, then lines 11, 13, 14 and 15 are skipped. It is assumed that each image in training is labeled to a class. Mini-batches of size  $M$  are generated by randomly sampling  $m$  images per class out of  $M/m$  sampled classes.

## 5.3 Experiments on Retrieval Benchmarks

### 5.3.1 Datasets

The training and evaluation is performed on four widely used image retrieval benchmarks, namely iNaturalist [91], PKU VehicleID [94], Stanford Online Products [92] (SOP) and Stanford Cars [93] (Cars196). Recall at top  $k$  retrieved images, denoted by  $r@k$ , is one of the standard evaluation metrics in these benchmarks. Metric  $r@k$  is 1 if at least one positive image appears in the top  $k$  list, otherwise 0. The metric is averaged across all queries. Note that this is different from the standard definition of recall in (5.1).

iNaturalist [91] is firstly used by Brown *et al.* [23], whose setup we follow: 5,690 classes for training and 2,452 classes for testing. For VehicleID, according to the standard setup [94], 13,134 classes are used for training, and the evaluation is conducted on the predefined small (800 classes), medium (1600 classes) and large (2400 classes) test sets. For SOP [92] and Cars196 [93], the standard experimental setup of Song *et al.* [205] is followed. The first half of

**Algorithm 3** Training with RS@k and SiMix.

---

```

1: procedure TRAIN-RS@K( $X, Y, M, m$ )
2:    $X$  : training images
3:    $Y$  : class labels
4:    $M$  : mini-batch size
5:    $m$  : number of images per class in mini-batch
6:
7:    $\theta \leftarrow$  initialize according to pre-training ▷ use ImageNet
8:   for iteration  $\in [1, \dots, \text{number-of-iterations}]$  do
9:      $loss \leftarrow 0$  ▷ set batch loss to zero
10:     $B \leftarrow$  BATCH-SAMPLER( $X, Y, M, m$ )
11:     $\hat{B} \leftarrow$  VIRTUAL-BATCH( $B$ ) ▷ enumerate virtual examples
12:    for  $(x, z) \in B \times B$  do compute  $s(x, z)$  ▷ use  $\mathbf{x}^\top \mathbf{z}$ 
13:    for  $(x, z) \in B \times \hat{B}$  do compute  $s(x, z)$  ▷ use (5.9)
14:    for  $(x, z) \in \hat{B} \times \hat{B}$  do compute  $s(x, z)$  ▷ use (5.10)
15:     $B \leftarrow B \cup \hat{B}$  ▷ expand batch with virtual examples
16:    for  $q \in B$  do ▷ use each image in the batch as query
17:       $B_q \leftarrow B \setminus q$  ▷ exclude query from the database
18:       $loss \leftarrow loss + L^K(q)$  ▷ Recall@k loss (5.7)
19:    end for
20:     $\theta \leftarrow$  MINIMIZE( $\frac{loss}{|B|}$ ) ▷ SGD update
21:  end for
22: end procedure

```

---

Dataset	#Images	#Classes	#Avg
iNaturalist Train [91]	325,846	5,690	57.3
iNaturalist Test [91]	136,093	2,452	55.5
VehicleID Train [94]	110,178	13,134	8.4
VehicleID Test [94]	40,365	4,800	8.4
SOP Train [92]	59,551	11,318	5.3
SOP Test [92]	60,502	11,316	5.3
Cars196 Train [93]	8,054	98	82.1
Cars196 Test [93]	8,131	98	82.9
$\mathcal{R}$ Oxford [95]	4,993	11	n/a
$\mathcal{R}$ Paris [95]	6,322	11	n/a
GLDv1 [224]	1,060,709	12,894	82.3

Table 5.1: Dataset composition for training and evaluation.

the classes are used for training and the rest for testing, resulting in 11,318 classes for SOP and 98 for Cars196.

The method is evaluated for instance-level search on Revisited Oxford ( $\mathcal{R}$ Oxford) and Paris ( $\mathcal{R}$ Paris) benchmark [95], where the evaluation metric is mean Average Precision (mAP). The training uses the Google Landmarks dataset (GLDv1) [224] to perform a comparison with the work of Revaud *et al.* [188] and their AP loss. The validation is performed according to the work of Tolias *et al.* [225].

The number of examples, classes, and average number of examples per class can be found in Table. 5.1. Note that these datasets are diverse in the number of training examples, the number of classes, and the number of examples per class, ranging from class balanced [93] to long-tailed [91].

### 5.3.2 Implementation details

Implementation details are identical for the four image retrieval benchmarks but differ for  $\mathcal{R}$ Oxford/ $\mathcal{R}$ Paris to follow and compare to prior work [188]. Differences are clarified when needed.

**Architecture.** An ImageNet [226] pre-trained ResNet-50 [192] is used as the backbone for deep image embeddings. Building on the standard implementation of [204], the BatchNorm parameters are kept frozen during the training. After the convolutional layers, Generalized mean pooling [227] and layer normalization [228] are used, similar to [199]. For vision transformers [30] ViT-B/32 and ViT-B/16 with an ImageNet-21k initialization from the timm library [229] are used. The last layer of the model is a  $d$  dimensional fully connected (FC) layer with  $L_2$  normalization. In the case of  $\mathcal{R}$ Oxford/ $\mathcal{R}$ Paris, ResNet-101 [192] is used, layer normalization is not added, while the FC layer is initialized with the result of whitening [227].

**Training hyper-parameters.** For ResNet architectures, Adam optimizer [230] is used and for vision transformers, AdamW [38] is used. We follow the standard class-balanced-sampling [23], [191], [199] with 4 samples per class for all the datasets, while classes with less than 4 samples are not used for training. Unless stated otherwise, the batch size for training is set 4,000 for all datasets but Cars196 where it is equal to  $4 \times \#classes = 392$ . Following the setup of ProxyNCA++ [199], the training set is split into training and validation by using the first half of the classes for training and the other half for validation. With this split, a grid search determines the learning rate, decay steps, decay size and the total number of epochs. Once the hyper-parameters are fixed, training is conducted once on the entire training set and evaluated on the test set. When training on GLDv1 and testing on  $\mathcal{R}$ Oxford/ $\mathcal{R}$ Paris, the batch size is set to 4096 [188], and training is performed for 500 batches, while other training hyper-parameters are set as in the work and GitHub implementation

of Radenovic *et al.* [227]. Note that the hyper-parameters for each dataset will be released with the implementation.

**RS@k hyper-parameters.** The proposed Recall@k Surrogate (RS@k) loss (5.5) contains three hyper-parameters: sigmoid temperature  $\tau_2$  - applied on similarity differences, sigmoid temperature  $\tau_1$  - applied on ranks and the set of values for  $k$  for which the loss is computed. Both sigmoid temperatures are kept fixed across all the experiments as  $\tau_2 = 0.01$  (same as [23]) and  $\tau_1 = 1$ . The values of  $k$  are kept fixed as  $k = \{1, 2, 4, 8, 16\}$  without SiMix and  $k = \{1, 2, 4, 8, 12, 16, 20, 24, 28, 32\}$  with SiMix. For GLDv1 [224], this is  $k = \{1, 2, 4\}$ , and  $k = \{1, 2, 4, 8\}$ , respectively. The values of  $k$  are studied in the supplementary materials and the sigmoid temperature  $\tau_1$  are investigated in Section 5.3.4, where it is observed that the method is not very sensitive to these hyper-parameters.

**Large batch size.** To dispense with the GPU hardware constraints and manage to train with the large batch size, we follow the multistage back-propagation of Revaud *et al.* [188]. A forward pass is performed to obtain all embeddings while intermediate tensors are discarded from memory. Then, the loss is computed, and so are the gradients w.r.t. the embeddings. Finally, each of the embeddings is recomputed, this time allowing the propagation of the gradients. Note that there is no implementation online of this approach and that the code of this work will become publicly available. Algorithm 3 does not include such implementation details, but it is compatible with such an extension. The batch-size impact for the proposed RS@k loss function is validated in Section 5.3.4.

**Discussion.** The methods in the literature use different embedding sizes,  $d$ , therefore, the models for the RS@k loss are trained with two embedding sizes of  $d = 128$  and  $d = 512$  for image retrieval benchmarks [91]–[94], and  $d = 2048$  for  $\mathcal{R}$ Oxford/ $\mathcal{R}$ Paris [95], to allow a fair comparison. In the standard split, the image retrieval benchmarks [91]–[94] do not contain an explicit validation set; as a result, image retrieval methods often tune the hyper-parameters on the test set, leading to the issue of training with test set feedback. This issue has been studied in [191], which proposes to train different methods with identical hyper-parameters. The setup of [191] is not directly usable for experiments with the RS@k loss, as large batch sizes are crucial to estimate recall@k accurately. Furthermore, their setup does not allow mixup. Therefore, instead of following [191], the issue is eliminated by using a part of the training set for validation as described above.

Method	Arch. <sup>dim</sup>	iNaturalist [91]			
		r@k			
		1	4	16	32
ProxyNCA [198]	$I_1^{128}$	<u>61.6</u>	<u>77.4</u>	<u>87.0</u>	90.6
Margin [26]	$R_{50}^{128}$	58.1	75.5	86.8	<u>90.7</u>
RS@k <sup>†</sup>	$R_{50}^{128}$	69.3	82.9	90.6	93.1
RS@k <sup>†</sup> +SiMix	$R_{50}^{128}$	<b>69.6</b>	<b>83.3</b>	<b>91.2</b>	<b>93.8</b>
		+21%	+26%	+32%	+33%
FastAP [190]	$R_{50}^{512}$	60.6	77.0	87.2	90.6
Blackbox AP [189]	$R_{50}^{512}$	62.9	79.0	88.9	92.1
SAP [23]	$R_{50}^{512}$	<u>67.2</u>	<u>81.8</u>	<u>90.3</u>	<u>93.1</u>
SAP <sup>†</sup> [23] +GeM +LN	$R_{50}^{512}$	68.7	82.7	90.9	93.5
RS@k <sup>†</sup>	$R_{50}^{512}$	71.2	84.0	91.3	93.6
RS@k <sup>†</sup> +SiMix	$R_{50}^{512}$	<b>71.8</b>	<b>84.7</b>	<b>91.9</b>	<b>94.3</b>
		+14%	+16%	+16%	+17%
SAP <sup>†</sup> [23]	ViT-B/32 <sup>512</sup>	72.2	84.6	91.6	93.9
RS@k <sup>†</sup>	ViT-B/32 <sup>512</sup>	75.9	87.1	93.1	95.1
SAP <sup>†</sup> [23]	ViT-B/16 <sup>512</sup>	79.1	89.0	94.2	95.8
RS@k <sup>†</sup>	ViT-B/16 <sup>512</sup>	83.9	92.1	95.9	97.2

Table 5.2: Recall@ $k$ (%) on iNaturalist [91]. Best results are shown with **bold**, previous state-of-the-art with underline and relative gains over the state-of-the-art in % of error reduction with **blue** and relative declines in **red**. Methods marked with <sup>†</sup> were trained using the same pipeline by us.

### 5.3.3 Evaluation

Unless otherwise stated, the results of the competing methods are taken from the original papers. Methods marked with a <sup>†</sup> were trained by us, using the same implementation as used for the RS@k loss. The results on image retrieval benchmarks [91]–[94] are compared with the methods that use either ResNet-50 [192] or Inception network [231]. ResNet-50 [192] is represented as  $R_{50}^d$  in the tables and the standard Inception network [231] as  $I_1^d$ , the Inception network with BatchNorm as  $I_3^d$  (same as [199]). Here  $d$  is the embedding size. On all the datasets, the performance of the baseline, Smooth-AP (SAP) [23], is also reported with Generalized mean pooling [227] and layer normalization [228], shown as SAP<sup>†</sup> (+Gem +LN). This is to eliminate any performance boost in the comparisons that were caused by the architecture. Note that unless otherwise stated in our experiments, the batch size for SAP is set as 384, the same as the original implementation [23]. Further, we demonstrate the performance of SAP and RS@k on ViT-B architectures. The variant of ViT-B that uses a patch size of  $32 \times 32$  is denoted by ViT-B/32 and the one that uses a patch size of  $16 \times 16$  by ViT-B/16.

**iNaturalist.** The results on iNaturalist [91] species recognition are presented in Table 5.2. The performances of the competing methods are taken from [23],

Method	Arch. <sup>dim</sup>	SOP [92]			
		r@k			
		10 <sup>0</sup>	10 <sup>1</sup>	10 <sup>2</sup>	10 <sup>3</sup>
ProxyNCA [198]	$I_1^{128}$	73.7	-	-	-
Margin [26]	$R_{50}^{128}$	72.7	86.2	93.8	98.0
Divide [25]	$R_{50}^{128}$	75.9	88.4	94.9	98.1
MIC [232]	$R_{50}^{128}$	77.2	89.4	95.6	-
Cont. w/M [233]	$R_{50}^{128}$	<u>80.6</u>	<u>91.6</u>	<u>96.2</u>	<u>98.7</u>
RS@k <sup>†</sup>	$R_{50}^{128}$	80.6	91.6	96.4	<b>98.8</b>
RS@k <sup>†</sup> +SiMix	$R_{50}^{128}$	<b>80.9</b>	<b>91.7</b>	<b>96.5</b>	<b>98.8</b>
		+1.5%	+1.2%	+7.9%	+7.7%
FastAP [190]	$R_{50}^{512}$	76.4	89.0	95.1	98.2
MS [234]	$I_3^{512}$	78.2	90.5	96.0	98.7
NormSoftMax [194]	$R_{50}^{512}$	78.2	90.6	96.2	-
Blackbox AP [189]	$R_{50}^{512}$	78.6	90.5	96.0	98.7
Cont. w/M [233]	$I_3^{512}$	79.5	90.8	96.1	98.7
HORDE [235]	$R_{50}^{512}$	80.1	91.3	96.2	-
ProxyNCA++ [199]	$R_{50}^{512}$	<u>80.7</u>	<u>92.5</u>	96.7	98.9
SAP [23]	$R_{50}^{512}$	80.1	91.5	<u>96.6</u>	<u>99.0</u>
SAP <sup>†</sup> [23] +GeM +LN	$R_{50}^{512}$	80.3	92.0	96.9	99.0
RS@k <sup>†</sup>	$R_{50}^{512}$	<b>82.8</b>	<b>92.9</b>	<b>97.0</b>	99.0
RS@k <sup>†</sup> +SiMix	$R_{50}^{512}$	82.1	92.8	<b>97.0</b>	<b>99.1</b>
		+11%	+5.3%	+12%	+10%
SAP <sup>†</sup> [23]	ViT-B/32 <sup>512</sup>	83.7	94.0	97.8	99.3
RS@k <sup>†</sup>	ViT-B/32 <sup>512</sup>	85.1	94.6	98.0	99.3
SAP <sup>†</sup> [23]	ViT-B/16 <sup>512</sup>	86.6	95.4	98.4	99.5
RS@k <sup>†</sup>	ViT-B/16 <sup>512</sup>	88.0	96.1	98.6	99.6

Table 5.3: Recall@k(%) on Stanford Online Products (SOP) [92]. Best results are shown with **bold**, previous state-of-the-art with underline and relative gains over the state-of-the-art in % of error reduction with **blue** and relative declines in **red**. Methods marked with † were trained using the same pipeline by us.

which uses the official implementations of these methods. It can be clearly seen that the RS@k outperforms classification and pairwise losses, including the three AP approximation losses, reaching the recall@1 score of 71.8% with SiMix, an error reduction of 14%.

**SOP.** The performance on SOP [92] is presented in Table 5.3, along with the comparisons with the competing methods. The proposed RS@k loss demonstrates clear state-of-the-art results, surpassing ProxyNCA++ [199] by 2.0% on recall@1, an error reduction of 10.4%. If a smaller batch size, equal to 384, is used for RS@k, it reaches a performance of 81.2%, 92.2%, 96.9% and 99.0% on r@10<sup>0</sup>, r@10<sup>1</sup>, r@10<sup>2</sup> and r@10<sup>3</sup> respectively. This result shows that large batch size helps in improving the performance, but RS@k outperforms the competing methods even with smaller batch size.



Method	Arch. <sup>dim</sup>	VehicleID [94]					
		r@k					
		Small		Medium		Large	
		1	5	1	5	1	5
Divide [25]	$R_{50}^{128}$	87.7	92.9	85.7	90.4	82.9	90.2
MIC [232]	$R_{50}^{128}$	86.9	93.4	-	-	82.0	91.0
Cont. w/M [233]	$R_{50}^{128}$	<u>94.7</u>	<u>96.8</u>	<u>93.7</u>	<u>95.8</u>	<u>93.0</u>	<u>95.8</u>
RS@k <sup>†</sup>	$R_{50}^{128}$	<b>95.6</b>	<b>97.8</b>	<b>94.4</b>	<b>96.8</b>	<b>93.5</b>	<b>96.6</b>
RS@k <sup>†</sup> +SiMix	$R_{50}^{128}$	95.4	97.5	93.8	96.6	93.0	96.2
		+17%	+31%	+11%	+24%	+7.1%	+19%
FastAP [190]	$R_{50}^{512}$	91.9	96.8	90.6	95.9	87.5	95.1
Cont. w/M [233]	$I_3^{512}$	94.6	96.9	<u>93.4</u>	96.0	<u>93.0</u>	96.1
SAP [23]	$R_{50}^{512}$	<u>94.9</u>	<u>97.6</u>	<u>93.3</u>	<u>96.4</u>	91.9	<u>96.2</u>
SAP <sup>†</sup> [23] +GeM +LN	$R_{50}^{512}$	94.2	97.2	92.7	96.2	91.0	95.8
RS@k <sup>†</sup>	$R_{50}^{512}$	<b>95.7</b>	<b>97.9</b>	<b>94.6</b>	<b>96.9</b>	<b>93.8</b>	<b>96.6</b>
RS@k <sup>†</sup> +SiMix	$R_{50}^{512}$	95.3	97.7	94.2	96.5	93.3	96.4
		+16%	+13%	+18%	+14%	+11%	+10%
SAP <sup>†</sup> [23]	ViT-B/32 <sup>512</sup>	94.8	97.7	93.5	96.8	92.1	96.3
RS@k <sup>†</sup>	ViT-B/32 <sup>512</sup>	95.1	97.7	94.1	96.7	93.2	96.5
SAP <sup>†</sup> [23]	ViT-B/16 <sup>512</sup>	95.5	97.7	94.2	96.9	93.1	96.6
RS@k <sup>†</sup>	ViT-B/16 <sup>512</sup>	96.2	98.0	95.2	97.2	94.7	97.1

Table 5.4: Recall@k(%) on PKU VehicleID [94]. Best results are shown with **bold**, previous state-of-the-art with underline and relative gains over the state-of-the-art in % of error reduction with **blue** and relative declines in **red**. Methods marked with † were trained using the same pipeline by us.

Method	Arch. <sup>dim</sup>	Cars196 [93]			
		r@k			
		1	2	4	8
ProxyNCA [198]	$I_1^{128}$	73.2	82.4	86.4	88.7
Margin [26]	$R_{50}^{128}$	<u>79.6</u>	<u>86.5</u>	<u>91.9</u>	<u>95.1</u>
RS@k <sup>†</sup>	$R_{50}^{128}$	78.1	85.8	91.1	94.5
RS@k <sup>†</sup> +SiMix	$R_{50}^{128}$	<b>84.7</b>	<b>90.9</b>	<b>94.7</b>	<b>96.9</b>
		+25%	+33%	+35%	+37%
MS [234]	$I_3^{512}$	84.1	90.4	94.0	96.1
NormSoftMax [194]	$R_{50}^{512}$	84.2	90.4	94.4	96.9
HORDE [235]	$R_{50}^{512}$	86.2	91.9	95.1	97.2
ProxyNCA++ [199]	$R_{50}^{512}$	<u>86.5</u>	<u>92.5</u>	<u>95.7</u>	<b>97.7</b>
SAP [23]	$R_{50}^{512}$	76.1	84.3	89.8	93.8
SAP <sup>†</sup> [23] +GeM +LN	$R_{50}^{512}$	78.2	85.6	90.8	94.3
RS@k <sup>†</sup>	$R_{50}^{512}$	80.7	88.3	92.8	95.7
RS@k <sup>†</sup> +SiMix	$R_{50}^{512}$	<b>88.2</b>	<b>93.0</b>	<b>95.9</b>	97.4
		+13%	+6.7%	+4.7%	-13%
SAP <sup>†</sup> [23]	ViT-B/32 <sup>512</sup>	78.1	85.7	91.0	94.8
RS@k <sup>†</sup>	ViT-B/32 <sup>512</sup>	78.1	86.4	92.3	95.6
SAP <sup>†</sup> [23]	ViT-B/16 <sup>512</sup>	86.2	92.1	95.1	97.2
RS@k <sup>†</sup>	ViT-B/16 <sup>512</sup>	89.5	94.2	96.6	98.3

Table 5.5: Recall@k(%) on Stanford Cars (Cars196) [93]. Best results are shown with **bold**, previous state-of-the-art with underline and relative gains over the state-of-the-art in % of error reduction with **blue** and relative declines in **red**. Methods marked with † were trained using the same pipeline by us.

**VehicleID.** The results on VehicleID [94] are presented in Table 5.4. RS@k outperforms the competing methods both with and without SiMix. Better results were observed without SiMix where RS@k reaches recall@1 performance of 95.7%, 94.6% and 93.8% on the small, medium, and large test sets, respectively.

**Cars196.** Evaluation on a small scale dataset, Cars196 [93] is presented in the Table 5.5. We train SAP with a batch size of 392; it provides a performance of 79.5%, 86.6%, 91.2%, and 94.4% and when combined with SiMix a performance of 85.4%, 91.0%, 94.3% and 96.7% on r@1, r@2, r@4 and r@8 respectively. SiMix makes a large difference in performance for both RS@k and SAP [23], primarily because of a smaller batch size (392), as constrained by the low number of classes. With SiMix, RS@k reaches the state-of-the-art results on three out of four recall@k values. If the batch size is further increased to 588 by changing the number of samples per class from 4 to 6, then RS@k provides a larger gain with performance 88.3%, 93.3%, 95.9% and 97.6%.

**Results with ViT-B.** The results by replacing the ResNet-50 [192] backbone with a ViT-B [30] for SAP [23] and the proposed RS@k are also shown

Arch.	Loss	Train-set	$\mathcal{RO}$		$\mathcal{RO}+\mathcal{R1M}$		
			med	hard	med	hard	
GeM*	AP [187]	Landmarks-clean [237][238]	[188]/ [225]	67.1	42.3	47.8	22.5
GeM*	AP [187]	GLDv1 [224]	[188]/github	66.3	42.5	-	-
GeM†	SAP [23]	GLDv1 [224]	[23]	67.9	46.3	49.5	25.8
GeM†	RS@k	GLDv1 [224]	ours	68.3	46.1	50.1	25.8
GeM+SiMix†	RS@k	GLDv1 [224]	ours	68.4	45.3	51.0	26.4

Table 5.6: Performance comparison (mAP%) on  $\mathcal{ROxford}$  with 1m distractor images ( $\mathcal{R1m}$ ). \* denotes that the FC layer is not part of the training but is added afterward to implement whitening. Batch size is 4096 for all methods; SiMix virtually increases it to 10240. ResNet101 is used as a backbone for all methods.

in Tables 5.2, 5.3, 5.4 and 5.5. With an exception of ViT-B/32 on VehicleID and Cars196 datasets, the use of ViT-B backbone leads to better performance for both methods, compared to the ResNet counterpart. It can be clearly seen that RS@k outperforms SAP [23] on all datasets. ViT-B/16 when trained with RS@k shows unprecedented performance on all datasets reaching recall@1 score of 83.9% on iNaturalist [91], 88.0% on SOP [92], 96.2% on VehicleID [94] (small) and 89.5% on Cars196 [93]. Note that while ResNet-50 has 24.5 M parameters and operates with 8.12 GMac/image, ViT-B has 87.8 M parameters and operates with 4.36 and 16.8 GMac/image for ViT-B/32 and ViT-B/16 respectively.

**Concurrent work.** The method of learning intra-batch connections for deep metric learning [202] achieves r@1 of 81.4% on the SOP and 88.1% on Cars196 dataset. The approach for Grouplet embedding learning [236] obtains r@1 of 82.0% on SOP and 91.5% on Cars196. The metric mixup approach [217] reports the best results of 81.3% r@1 on SOP in combination with ProxNCA++ [199] and 89.6% on Cars196 which is in combination with MS [234].

**$\mathcal{ROxford}/\mathcal{RParis}$ .** Tables 5.6, 5.7, and 5.8 summarizes a comparison with AP-based losses in the literature on  $\mathcal{ROxford}/\mathcal{RParis}$  with and without distractor images. The comparison is performed with GLDv1 as a training set whose performance is reported for the work of Revaud *et al.* [188] in their GitHub page, while the *landmarks-clean dataset* is avoided as all initial images are not publicly available at the moment. During the training performed by us, training images are down-sampled to have a maximum resolution of  $1024 \times 1024$ . The inference is performed with multi-resolution descriptors at three scales with up-sampling and down-sampling by a factor of  $\sqrt{2}$ . Note that SAP is not evaluated on these datasets in the original work and this experiment is performed by us, which outperforms the previously used AP loss [187]. RS@k, with or without the SiMix, increases the performance by a

Arch.	Loss	Train-set	$\mathcal{R}Par$		$\mathcal{R}P+\mathcal{R}1M$			
			med	hard	med	hard		
GeM*	AP [187]	Landmarks-clean [237][238]	[188]/	[225]	80.3	60.9	51.9	24.6
GeM*	AP [187]	GLDv1 [224]	[188]/	github	80.2	60.8	-	-
GeM†	SAP [23]	GLDv1 [224]		[23]	81.7	63.3	57.4	29.8
GeM†	RS@k	GLDv1 [224]		ours	82.1	63.9	57.9	30.2
GeM+SiMix†	RS@k	GLDv1 [224]		ours	81.2	62.4	58.7	31.1

Table 5.7: Performance comparison (mAP%) on  $\mathcal{R}Paris$  with 1m distractor images ( $\mathcal{R}1m$ ). \* denotes that the FC layer is not part of the training but is added afterward to implement whitening. Batch size is 4096 for all methods; SiMix virtually increases it to 10240. ResNet101 is used as a backbone for all methods.

Arch.	Loss	Train-set	Mean			
			all	$\mathcal{R}1M$		
GeM*	AP [187]	Landmarks-clean [237][238]	[188]/	[225]	49.7	36.7
GeM*	AP [187]	GLDv1 [224]	[188]/	github	-	-
GeM†	SAP [23]	GLDv1 [224]		[23]	52.7	40.6
GeM†	RS@k	GLDv1 [224]		ours	53.1	41.0
GeM+SiMix†	RS@k	GLDv1 [224]		ours	53.1	41.8

Table 5.8: Performance comparison (mAP%) on  $\mathcal{R}Oxford$  and  $\mathcal{R}Paris$  with 1m distractor images ( $\mathcal{R}1m$ ). Mean performance is reported across all setups or the large-scale setups only. \* denotes that the FC layer is not part of the training but is added afterward to implement whitening. Batch size is 4096 for all methods; SiMix virtually increases it to 10240. ResNet101 is used as a backbone for all methods.

small margin.

### 5.3.4 Effect of hyper-parameters

We study the impact of hyper-parameter on the Cars196 dataset [93] since it is the smallest compared to the others and has the lowest training time.

**Sigmoid temperature  $\tau_1$  - applied on ranks.** The effect of the sigmoid temperature  $\tau_1$  is summarized in Figure 5.4 (top). For both setups of with and without SiMix,  $\tau_1 = 1.0$  gives best results while higher and lower values lead to a decline.

**Batch size.** The effect of the varying batch size is shown in Figure 5.4 (bottom). It demonstrates that large batch size leads to better results. A significant performance boost is observed with the use of SiMix, especially in the small batch size regime, which comes at a small extra computation. A comparison with SAP [23] is also shown in this figure. Note that on smaller batch sizes, the proposed RS@k outperforms SAP with a larger margins.

**Values for  $k$ .** The study for the set of values of  $k$  used for RS@k loss can be found in Table 5.9. The results RS@{1}, RS@{1, 2}, RS@{1, 2, 4} and RS@{1, 2, 4, 8} suggest that adding larger values of  $k$  leads to decline in the performance. However, RS@{1, 2, 4, 8, 16} gives on an average the same results as RS@{1}, with higher performance on larger  $k$  values. Comparing the entries RS@{4, 8, 16} with RS@{1, 2, 4, 8, 16} suggests that the use of small values, such as  $k = 1$  or  $k = 2$ , is crucial as the performance drops significantly when these values are removed. Further removing  $k = 4$  (RS@{8, 16}) does not change the performance. However, removing  $k = 8$  (RS@{16}) leads to a significant decline in the performance.

Method	r@1	r@2	r@4	r@8	r@16	Avg
RS@{1}†	81.1	87.7	92.0	95.0	96.9	90.5
RS@{1, 2}†	80.2	87.2	91.9	95.0	97.2	90.3
RS@{1, 2, 4}†	79.6	86.5	91.2	94.5	96.8	89.7
RS@{1, 2, 4, 8}†	79.3	86.3	91.0	94.5	96.9	89.6
RS@{1, 2, 4, 8, 16}†	80.8	87.6	92.2	95.0	97.1	90.5
RS@{2, 4, 8, 16}†	80.3	87.5	92.3	95.4	97.5	90.6
RS@{4, 8, 16}†	79.6	87.1	91.7	95.0	97.3	90.1
RS@{8, 16}†	79.6	87.1	91.7	95.0	97.3	90.1
RS@{16}†	75.8	83.9	89.8	93.6	96.4	87.9

Table 5.9: Varying the set of values of  $k$ . Results on Cars196 [93]. In all experiments,  $\tau_1 = 1$  and  $\tau_2 = 0.01$ .

**Impact of SiMix** Our results suggest that SiMix leads to a larger performance gain on smaller datasets, where batch size is restricted by the total

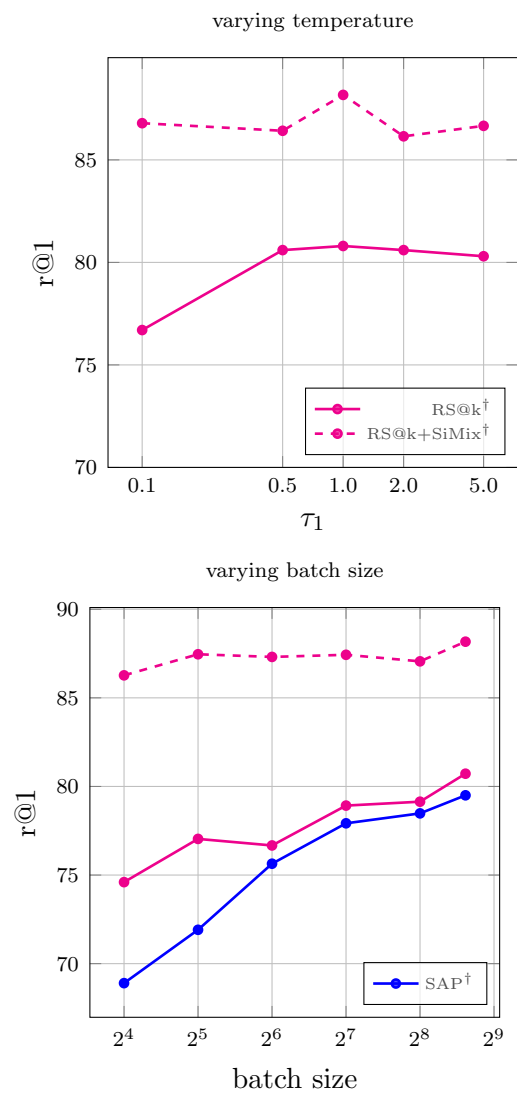


Figure 5.4: The effect of sigmoid temperature  $\tau_1$  applied on ranks (top) and of batch size (bottom). Results are shown on Cars196 [93].

number of classes. Results are summarized in Table 5.10, where we additionally report results on CUB which has small (100) number of training classes. On Cars196 dataset, RS@k attains a r@1 of 80.7% without and 88.2% with SiMix (an absolute improvement for 7.5%). Similarly on CUB200, RS@k attains a r@1 of 63.8% without and 69.5% with SiMix (an absolute improvement of 5.7%).

Dataset	# Training Samples	SAP <sup>†</sup>	RS@k <sup>†</sup>	RS@k <sup>†</sup> + SiMix
iNaturalist	325,846	70.7	71.2	71.8
VehicleID	110,178	95.5	95.7	95.3
SOP	59,551	81.3	82.8	82.1
Cars196	8,054	79.5	80.7	88.2
CUB200	5,864	63.6	63.8	69.5
$\mathcal{R}Oxf$ & $\mathcal{R}Par$ (1m)	1,060,709	40.6	41.0	41.8

Table 5.10: Recall@1 (in %) with batch size of  $\min(4000, 4 \times \#classes)$  for iNaturalist, VehicleID, SOP, Cars196 and CUB200. mAP (in %) with batch size of 4096 for  $\mathcal{R}Oxford$  and  $\mathcal{R}Paris$  with 1 million distractor samples.

## 5.4 Experiments on Fine-Grained Classification

This section compares training a softmax image classifier explicitly and training an image retrieval system, which is subsequently used for nearest neighbour classification. The resolution of images, pre-trained weights and number of training epochs are kept the same across the two setups for a fair comparison.

Overall, the proposed retrieval approach achieved superior performance in all measured scenarios. Notably, the ViT-Base/16 feature extractor architecture achieved a higher classification accuracy with a margins of 0.28%, 4.13%, and 10.25% on ExpertLifeCLEF 2018 [239], PlantCLEF 2017 [240] and iNat2018–Plantae [91], respectively. Besides, the macro-F1 performance differences margin is noticeably higher—1.85% for ExpertLifeCLEF 2018 and 12.23% for iNat2018–Plantae datasets. Even though the standard classification approach performs better on classes with fewer samples (See Table 5.6), common species with high a-prior probability are frequently wrongly predicted. This is primarily due to the high-class imbalance preserved in the dataset mimicked by the deep neural network optimized via SoftMax Cross-Entropy Loss. Thus, the results of the standard image classification approach performs way worst in case of the macro-F1 score. Full comparison of the classification and retrieval-based methods and their appropriate recognition scores are listed in Table 5.11. Three architectures—ResNet-50, ViT-Base/32, and ViT-Base/16 are evaluated. It can be seen from the results that for all selected architectures, retrieval leads to a better performance.

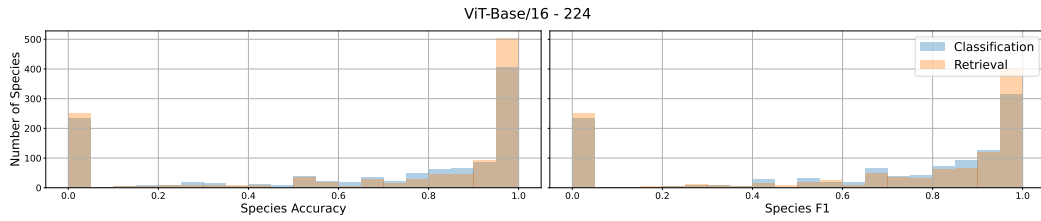


Figure 5.5: Species-wise classification performance histogram within a given 5% interval, evaluated on the PlantCLEF2017 test set with ViT-Base/16 backbone and Classification and Retrieval approaches.

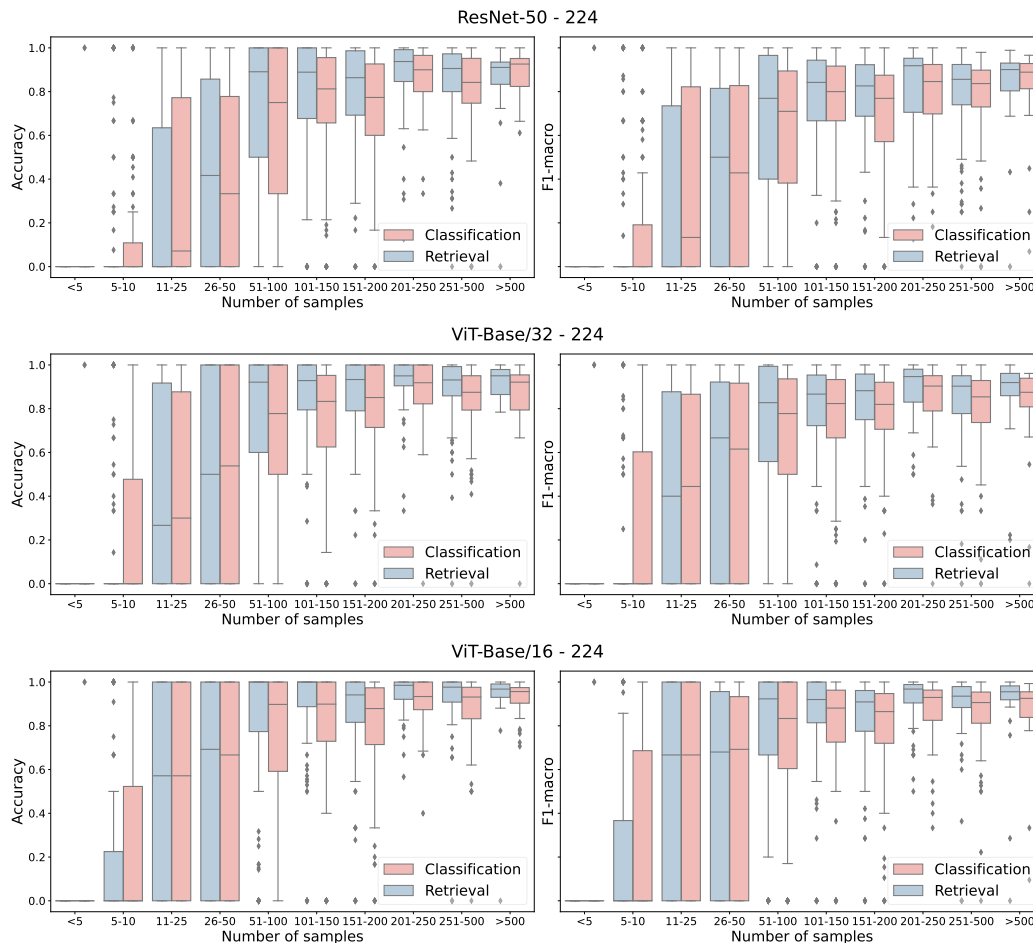


Figure 5.6: Classification performance (F1 and Accuracy) as box-plot for three backbone architectures and Classification Retrieval approaches. Tested on PlantCLEF2017 test set with input resolution of  $224 \times 224$ .



Table 5.11: Classification (C) vs Retrieval (R). All models were trained for 100 epochs with fixed image size ( $224 \times 224$ ). No test-time augmentations were used. The most confident image prediction is used for all images belonging to the same observation.

Architecture	Method	ExpertLifeCLEF 2018		PlantCLEF 2017		iNat2018–Plantae	
		Acc.	macro F1	Acc	macro F1	Acc	macro F1
ResNet-50	C	59.87	55.11	77.89	54.48	57.73	52.69
ViT-Base/32	C	65.21	60.29	80.68	59.18	57.24	53.17
ViT-Base/16	C	71.71	67.35	84.48	65.40	67.42	64.51
ResNet-50	R	60.15	56.30	80.27	55.57	57.95	56.32
ViT-Base/32	R	66.48	61.49	84.89	60.79	63.12	61.24
ViT-Base/16	R	<b>71.99</b>	<b>69.20</b>	<b>88.61</b>	<b>66.39</b>	<b>77.67</b>	<b>76.74</b>

Training an image retrieval system and subsequently performing a nearest neighbour classification is a competitive alternative, with better results than direct classification. The prediction obtained via a nearest neighbour search is more interpretable as the samples contributing to the prediction can be visualized. Therefore, a retrieval-based approach is more suitable if utilized within the humans in the loop. On the other hand, the softmax predictions of a standard neural network classifier allow for simple post-processing procedures such as averaging, prior shift adaptation, etc., which are yet to be explored for the retrieval approach, and which noticeably improve the final recognition accuracy of the standard classifiers.

Overall, using image-retrieval has clear advantages, e.g., recovering relevant nearest-neighbour labelled samples, providing ranked class predictions, and allows user or experts to visually verify the species based on the  $k$ -nearest neighbours. Besides, the retrieval approach naturally supports open-set recognition problems, i.e., the ability to extend or modify the set of recognised classes after the training stage. The set of classes may change e.g. as a result of modifications to biological taxonomy. New classes are introduced simply by adding training images with the new label, whereas in the standard approach, the classification head needs re-training. On the negative side, the retrieval approach requires, on top of running the deep net to extract the embedding, to execute the nearest neighbour search efficiently, increasing the overall complexity of the fine-grained recognition system.

Contrary to our expectations, the error analysis in Figure 5.6 shows that the retrieval approach does not bring an improvement in classifying images from classes with few training samples. Figure 5.5 shows that retrieval has a very high accuracy for a higher number of species, but it also fails for a higher number of species.

## 5.5 Conclusions

This work has presented image embedding learning for retrieval using a novel surrogate loss function for the recall@k metric. State-of-the-art results were achieved on a number of standard benchmarks. Training with very large batch size, up to 4k images, has shown to be highly beneficial. The batch size is further increased, in a virtual way, with a newly proposed mixup approach that acts directly on the scalar similarities. This approach offers a boost in performance at a small increase of the computational cost, while its applicability goes beyond the proposed loss. The implementation of the proposed Recall@k Surrogate loss, proposed similarity mixup, along with the training procedure that allows the use of large batch sizes on a single GPU by sidestepping memory constraints, is available at [https://github.com/yash0307/RecallatK\\_surrogate](https://github.com/yash0307/RecallatK_surrogate).



Figure 5.7: Qualitative examples from the retrieval approach on iNaturalist dataset. The left most column shows samples from the test set followed by five nearest neighbours in the learned embedding space from the training set.

## Chapter 6

# Contrastive Classification and Representation Learning with Probabilistic Interpretation

Representation learning is a powerful tool to create an embedding space that is beneficial for performing downstream tasks e.g., classification or retrieval. Contrastive representation learning first proposed by [241] is a dominant successful line for representation learning. It divides the data into pairs of positive (similar) and negative (unrelated) samples with the objective of maximizing the similarity of positive pairs samples and minimize it for negative pairs.

More recently, contrastive learning has become a key component of methods for self-supervised learning [242]–[245] and has shown impressive performance [245], [246] that is very close to the supervised learning counterpart with cross entropy loss. Moreover, it was shown that supervised contrastive learning marginally outperforms the *cross entropy loss* in fully supervised image classification [82]. Not only for standard supervised classification but it has been applied in continual learning [247], Out of Distribution Detection [248], Domain Adaptation [249] and many more showing superior performance to cross entropy based counterpart.

Minimizing *cross entropy* (CE) loss is widely used in training deep neural network classifiers, derived as the maximum likelihood estimate (MLE) of classifier’s parameters  $\theta$  to approximate posterior probabilities  $\hat{p}(\text{class}|\text{observation})$ . [250] draw the connection among popular pairwise-distance losses and the cross entropy loss, showing that all of them are related to maximizing the mutual information (MI) between the learned embeddings and the corresponding samples’ labels.

We emphasize the advantages of the probabilistic interpretation of the CE loss in classification problems. Such explicit probabilistic interpretation is missing within the embedding spaces trained by popular contrastive learning methods. The posterior estimates  $\hat{p}(\text{class}|\text{observation})$  can be utilized when combining classifiers [251]–[253], in adaptation to prior shift [254]–[257], in

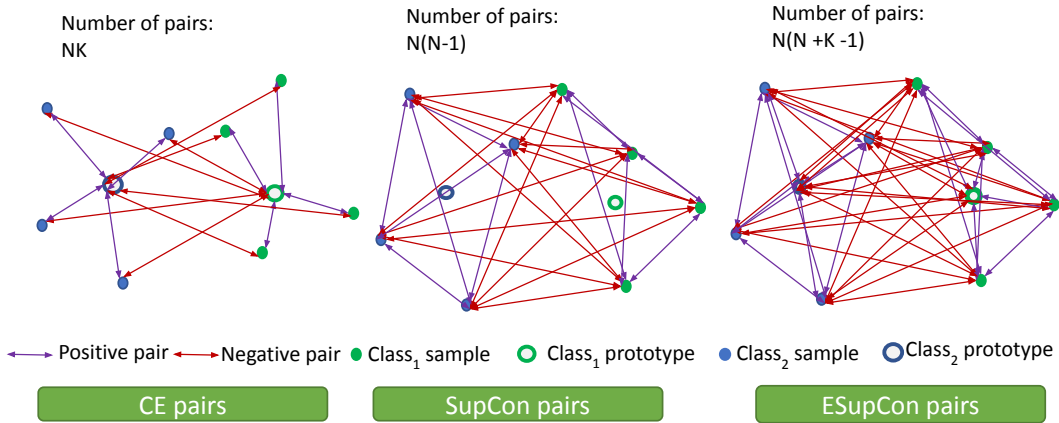


Figure 6.1: Illustration of the possible number of pairs that each loss accesses during training in the learned embedding space,  $N$  is the batch size and  $K$  is the number of classes. CE pairs are only defined through classes weights while in SupCon each sample forms positive pairs with its class samples and negative pairs with samples from other classes. For our ESupCon in addition to the positive and negative samples pairs, class weights (prototypes) form positive pairs with corresponding class samples and negative pairs with other classes samples. Note that here we don't consider augmentations.

knowledge distillation [258]; out-of-distribution detection [259] and in many other problems.

In this chapter, we suggest that one possible reason for the improved performance of supervised contrastive learning is the inherent access to a large number of samples pairs, while the “pairs” within the softmax CE loss are centered around the linear classifier weights. Here we draw an analogy with proxy based loss and consider the linear classifier weights optimized in the softmax CE loss as proxies for learning the samples representations. Proxy base training that utilizes proxies instead of the direct sample to sample relationship is simple and faster to converge, however, it doesn't leverage the rich data to data similarities as the supervised contrastive loss. We refer to Figure 6.1 for an illustration on this assumption. We hypothesize that the access to more pairs during training might lead to a better convergence and less overfitting resulting in the advantages hinted in recent work [82], [260].

Hence, to combine the advantages of contrastive representation learning via pairwise losses and the clear probabilistic interpretation of classifiers trained by cross entropy minimization, we present the following contributions: First, we consider the weights of the last linear classification layer as prototypes of each class. We show that adding a simple term corresponding to maximizing the similarity between the prototypes and their class samples, leads to an assignment of the prototypes to the mean of each class samples with momentum updates of representation. This is optimized during the representation train-

ing with a supervised contrastive loss [82], resulting in a nearest prototype classifier [261]. Second, we propose an extension to the *supervised contrastive loss* (SupCon) [82], where samples of a given class form positive pairs with their class prototype and other classes samples correspond to negative pairs.

We show that the resulting objective combines in its formulation the SupCon loss [82] and the standard CE loss on prototypes related pairs, preserving the probabilistic interpretation of the predictions. We refer to this loss as ESupCon (short for *Extended Supervised Contrastive loss*).

Third, we revisit the Simplified Pairwise Cross Entropy (SPCE) loss, proposed in the theoretical analysis of [250], and compare it with standard CE loss and the supervised contrastive learning loss in an extensive experimental evaluation.

In our experimental evaluation, we not only consider the fully supervised setting but also for the first time a number of challenging settings (low sample regime, imbalanced data and noisy labels). To the best of our knowledge, this is the first comprehensive evaluation of SupCon loss and the standard CE loss in addition to our proposed extensions.

We show ESupCon is more powerful as a training objective than the standard CE loss while maintaining a probabilistic interpretation. and is more robust in challenging and low sample settings. Surprisingly, our simple prototypes similarity term is more robust than CE loss for learning a linear classifier after SupCon in most of the imbalanced and noisy data experiments.

In the following, we describe the closely Related Work, then provide a short Background on pairwise losses and the link to Cross Entropy loss, followed by our extension to Supervised Contrastive Loss. We validate and compare different studied losses in the Experiments, and summarize our contributions and limitations in Conclusion.

## 6.1 Related Work

CE loss is a standard and powerful training objective to optimize deep neural networks for classification-related problems. For long, the CE loss was believed to be more effective than representation learning losses *e.g.*, metric learning based losses. For example, [250] studied the relation of CE loss to contrastive metric learning losses and showed that the CE loss also has a contrastive and a tightness part. The authors suggested that CE “does it all” and that it is easier to optimize compared to its contrastive-learning counterparts. Recently, self-supervised learning losses have shown great success [242], [244]–[246], [262]–[264] as pretraining methods with only a small performance gap to that of fully supervised learning. The core of the self-supervised methods is the use of rich data augmentation methods to construct positive pairs corresponding to augmented version of a given sample. Closer to our work, [246], [265] construct clusters and establish cluster assignments through prototypes while learning the embedding space. It remains unclear how these losses can be extended to

the supervised setting as in our case.

Inspired by the self-supervised SimCLR loss [242], [82] introduced a new supervised contrastive learning method called SupCon, which achieved superior results compared to the standard CE minimization, and which has been shown to be more generalizable and robust to noise. However, the method is only used to train the image representation and still relies on the CE loss to train the linear classifier afterwards. CE-based training suffers from known issues of noise sensitive, overfitting [119], and being less transferable than the representation learning counterparts [242]. Recently, [260] investigated the difference between the SupCon [82] loss and the CE loss in the geometry of the targeted representation. It was shown that both losses target the same geometric solution, however, SupCon converges much closer to the target leading to a better generalization performance.

As such, starting from the nice suggested characteristics of the SupCon loss based training, we propose and study alternatives that can train the whole network (representation and classifier) end-to-end, while preserving both the performance improvements of contrastive representation learning and the clear probabilistic interpretation of the CE loss. We start by considering the classes weights as prototypes for each class samples. We learn these prototypes while maximizing positive pairs similarities and minimizing negative pairs similarities. Our work hence can be seen as combination of proxy (prototype) based and pairwise based contrastive representation learning. Proxy based losses resort to learning a set of proxies as representative of clusters or classes of samples and optimize the similarities to these proxies rather than the data to data similarities. Proxy NCA [266] was the first proxy base metric learning method, it is an approximation of NCA (Neares Component Analysis) using proxies. We note that in the case of learning with class level labels the Proxy NCA matches learning with Softmax Cross Entropy loss when the last classification layer is without a bias term and its weights are normalized vectors. Proxy anchor loss [267], attempts to combine the benefits of both proxy-based and pairwise losses. While in the main loss formulation only similarities to proxies are considered, the magnitude of the loss gradient w.r.t. each sample is scaled by the corresponding proxy similarity proportional to other samples-proxies similarities. In general, proxy based losses do not use the proxy at test time and it is unknown how they perform for classification or whether there can exist any probabilistic interpretations. Circle loss [268] presents a unified framework for both pairwise and proxy based losses but it adopts an adaptive scaling of the loss depending on how much a given similarity is deviated from its optimum. In doing so, Circle loss abandons the probabilistic interpretation of a sample assignment to its prototype (proxy).



## 6.2 Background

In this section, we describe recent self-supervised and supervised contrastive losses and the connection with CE loss.

### 6.2.1 Pairwise Losses

Contrastive losses work with pairs of embeddings that are pulled together if a pair is positive (related embeddings) and pulled further apart otherwise [241]. Consider the following: 1) a random data augmentation module that for each sample  $\mathbf{x}$  generates two differently augmented samples, 2) a neural network encoder  $f$  that maps an augmented input sample  $\mathbf{x}$  to its feature representation:  $f(\mathbf{x}) = \mathbf{z}, \mathbf{z} \in \mathbb{R}^d$ . We start by outlining SimCLR [242], a popular, effective and simple self supervised contrastive loss to lay the ground for our work:

$$\ell_{\text{SimCLR}} = \frac{1}{2N} \sum_i^N \ell_{\text{SimCLR}}(\mathbf{z}_i, \mathbf{z}_{i+N}) + \ell_{\text{SimCLR}}(\mathbf{z}_{i+N}, \mathbf{z}_i), \quad (6.1)$$

$$\ell_{\text{SimCLR}}(\mathbf{z}_i, \mathbf{z}_j) = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k \neq i} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)}, \quad (6.2)$$

where  $\tau$  is the temperature scaling term,  $N$  is the mini batch size, and the pairs  $(\mathbf{z}_i, \mathbf{z}_j)$  consist of features of two differently augmented views of the same data example and  $\text{sim}(\mathbf{z}_i, \mathbf{z}_j) = \frac{\mathbf{z}_i^\top \mathbf{z}_j}{\|\mathbf{z}_i\| \cdot \|\mathbf{z}_j\|}$  is the cosine similarity. Assuming normalized embedding vectors  $\mathbf{z}_i$ , this pairwise loss is:

$$\ell_{\text{SimCLR}}(\mathbf{z}_i, \mathbf{z}_j) = -\mathbf{z}_i^\top \mathbf{z}_j / \tau + \log \sum_{k \neq i} \exp(\mathbf{z}_i^\top \mathbf{z}_k / \tau). \quad (6.3)$$

Note that the first term corresponding to the positive pair is the tightness term and the second one is the contrastive term. The aforementioned self-supervised batch contrastive approach was extended in [82] to the fully supervised setting with the Supervised Contrastive Loss:

$$\ell_{\text{SupCon}} = \frac{1}{2N} \sum_i^{2N} \ell_{\text{SupCon}}(\mathbf{z}_i, P_i), \quad (6.4)$$

$$\begin{aligned} \ell_{\text{SupCon}}(\mathbf{z}_i, P_i) &= \\ &= -\frac{1}{|P_i|} \sum_{\mathbf{z}_p \in P_i} \log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_p)/\tau)}{\sum_{j \neq i} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)} = \\ &= \frac{1}{|P_i|} \sum_{\mathbf{z}_p \in P_i} \left( -(\mathbf{z}_i^\top \mathbf{z}_p) / \tau + \log \sum_{j \neq i} \exp((\mathbf{z}_i^\top \mathbf{z}_j) / \tau) \right), \end{aligned} \quad (6.5)$$



where  $P_i$  is the set of representations  $\mathbf{z}_p$  forming positive pairs for the  $i$ -th sample, and the index  $j$  iterates over all (original and augmented) samples. SupCon loss is expressed as the average of the loss defined on each positive pair where in this supervised setting, the positive pairs are formed of augmented views and other samples of the same class. The authors showed that the supervised contrastive learning achieves excellent results in image classification, improving ImageNet classification accuracy with ResNet-50 by 0.5% compared to the best results achieved by training with the CE loss.

## 6.2.2 Cross Entropy and Pairwise Cross Entropy

The cross entropy (CE) loss is a common choice for training classifiers, as its minimization leads to the maximum likelihood estimate of the classifier parameters for estimating the posterior probabilities  $\hat{p}(\text{class}|\text{observation})$ .

For  $N$  samples of  $K$  classes, and a single-label softmax classifier, the CE loss can be defined as follows:

$$\begin{aligned} \ell_{\text{CE}} &= \frac{1}{N} \sum_{i=1}^N \ell_{\text{CE}}(\mathbf{z}_i) = -\frac{1}{N} \sum_{i=1}^N \log \frac{\exp \boldsymbol{\theta}_{y_i}^\top \mathbf{z}_i}{\sum_{k=1}^K \exp \boldsymbol{\theta}_k^\top \mathbf{z}_i} \\ &= -\frac{1}{N} \sum_{i=1}^N \boldsymbol{\theta}_{y_i}^\top \mathbf{z}_i + \frac{1}{N} \sum_{i=1}^N \log \sum_{k=1}^K \exp \boldsymbol{\theta}_k^\top \mathbf{z}_i, \end{aligned} \quad (6.6)$$

where  $\mathbf{z}_i$  is sample feature for the  $i$ -th observation having label  $y_i \in \{1, \dots, K\}$ , and  $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K)$  are the parameters of the last fully connected layer, assuming that no bias term is used.

The *Simplified Pairwise Cross Entropy* (SPCE) loss was introduced in [250] as a variant of the CE loss (6.6):

$$\ell_{\text{SPCE}} = -\frac{1}{N} \sum_{i=1}^N \log \frac{\exp \left( \frac{1}{N} \sum_{j:y_j=y_i} \mathbf{z}_j^\top \mathbf{z}_i \right)}{\sum_{k=1}^K \exp \left( \frac{1}{N} \sum_{j:y_j=k} \mathbf{z}_j^\top \mathbf{z}_i \right)}. \quad (6.7)$$

When training the feature encoder with the  $\ell_{\text{SPCE}}$  loss, the classifier weights  $\boldsymbol{\theta}$  can be estimated directly from the class feature means  $\mathbf{c}_k$ . Moreover, the class posterior probabilities  $p(k|\mathbf{z}_i)$  also can be estimated explicitly:

$$p(k|\mathbf{z}_i) = \frac{\exp \left( \frac{1}{N} \sum_{j:y_j=k} \mathbf{z}_j^\top \mathbf{z}_i \right)}{\sum_{c=1}^K \exp \left( \frac{1}{N} \sum_{j:y_j=c} \mathbf{z}_j^\top \mathbf{z}_i \right)}. \quad (6.8)$$

In the experimental section, we will evaluate SPCE loss and compare it with SupCon. Differently from SPCE, with SupCon, one needs to train a classifier on top of the learned representation as a posthoc process. In the following we will discuss and propose alternatives to jointly learn the classifier and the feature extraction parameters.

### 6.3 Learning a Classifier Jointly with Representation Learning

Representation learning under SupCon or SPCE losses targets grouping one class samples together while pushing away samples of other classes. In fact, both losses contain tightness and contrastive terms and fulfill similar objectives to that of the cross entropy loss.

Assuming that forcing samples of different classes to lie far apart is achieved by the contrastive part of SupCon or SPCE, in order to learn the parameters of the classifier, one can consider the weight vectors of the linear classifier as prototypes and optimize these prototypes to be closest to the samples of the class they represent (with solely a tightness term). We assume that both the samples representations and the classifier weights are normalized vectors and that the classifier is linear with no bias term. We define the following loss to learn the desired prototypes:

$$\ell_{\text{tt}} = \frac{1}{N} \sum_i^N \ell_{\text{tt}}(\mathbf{z}_i, \boldsymbol{\theta}_{y_i}) = \frac{1}{N} \sum_i^N -\mathbf{z}_i^\top \boldsymbol{\theta}_{y_i}. \quad (6.9)$$

Note that the number of samples in (6.9) might differ from  $N$  (e.g., due to augmentation), in which case  $N$  should be replaced by the corresponding number of samples. With that assumption, the classifier we use is a nearest prototype classifier i.e., assigning a test sample to the class of the nearest prototype. Note that  $\ell_{\text{tt}}$  resembles only the tightness part of the CE loss (6.6). The gradient of the  $\ell_{\text{tt}}$  loss w.r.t. the classifier weights can be directly derived from (6.9):

$$\frac{\partial \ell_{\text{tt}}}{\partial \boldsymbol{\theta}_k} = -\frac{1}{N} \sum_{i:y_i=k} \mathbf{z}_i. \quad (6.10)$$

Through minimizing this loss jointly with the representation learning loss, we update the classifier weights using the following iterative formula:

$$\boldsymbol{\theta}_k^0 = \eta \frac{1}{N} \sum_{i:y_i=k} \mathbf{z}_i^0, \quad \boldsymbol{\theta}_k^{t+1} = \boldsymbol{\theta}_k^t + \eta \frac{1}{N} \sum_{i:y_i=k} \mathbf{z}_i^{t+1}, \quad (6.11)$$

where  $t$  is the iteration index and  $\eta$  is the learning rate. Note that this is equivalent to setting (up to a constant) the class weights  $\boldsymbol{\theta}_k$  to the class features mean  $\mathbf{c}_k$  with momentum updates, where the new prototype combines the new iteration representation mean with the previous iteration mean. We will compare the minimization of the  $\ell_{\text{tt}}$  loss jointly with the the representation learning loss vs. simply setting the classifier weights  $\boldsymbol{\theta}_k$  to the hard mean  $\mathbf{c}_k$  for each class  $k$ .

## 6.4 Extended Supervised Contrastive Learning

Here we aim at extending the SupCon loss to include the classes prototypes being learned. For this, we propose to consider an explicit linear classification layer with parameters  $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K)$  in the optimization of the supervised contrastive loss (6.5). Note that here we consider the embeddings  $\mathbf{z}_i$  and the class prototypes  $\boldsymbol{\theta}_k$  in the same feature space. A class prototype  $\boldsymbol{\theta}_k$  should represent as closely as possible its class features. Hence a prototype similarity with its class features should be maximized and minimized with other classes features. To achieve this we propose to construct the following prototype-feature pair  $(\mathbf{z}_i, \boldsymbol{\theta}_{y_i})$  with sample representation  $\mathbf{z}_i$  ( $y_i = k$ ) as a positive pair. Now we define the following loss on a positive prototype-feature pair:

$$\begin{aligned} \ell_{\text{pt}}(\mathbf{z}_i, \boldsymbol{\theta}_{y_i}) &= -\mathbf{z}_i^\top \boldsymbol{\theta}_{y_i} \\ &+ \log \left( \sum_{k=1}^K \exp(\mathbf{z}_i^\top \boldsymbol{\theta}_k) + \sum_{j=1: j \neq i}^{2N} \exp(\mathbf{z}_i^\top \mathbf{z}_j) \right). \end{aligned} \quad (6.12)$$

Note that SupCon loss on a positive pair of samples is defined as follows:

$$\ell_{\text{SupCon}}(\mathbf{z}_i, \mathbf{z}_p) = -\mathbf{z}_i^\top \mathbf{z}_p + \log \sum_{j \neq i} \exp(\mathbf{z}_i^\top \mathbf{z}_j). \quad (6.13)$$

Here we omit the temperature  $\tau$  for clarity and for a better connection to the CE loss. In (6.12) we have extended the set of existing data representations  $\mathbf{z}_i$  with the class prototypes  $\boldsymbol{\theta}_l$ . Following the same analogy and constructing all positive prototype-feature pairs, the prototype loss for a class weight  $\boldsymbol{\theta}_k$  will be defined as follows.

$$\ell_{\text{pt}}(\boldsymbol{\theta}_k) = \frac{1}{2N_k} \sum_{i: y_i = k} \ell_{\text{pt}}(\mathbf{z}_i, \boldsymbol{\theta}_k). \quad (6.14)$$

Note that the number of summation terms in (6.14) is  $2N_k$  (where  $N_k$  is the number of the non-augmented samples in  $k$ -th class), since the samples in SupCon are considered with their augmentations. Having the loss defined per prototype  $\boldsymbol{\theta}_k$ , we can define the full objective function that optimizes the encoder (representation backbone) parameters jointly with the classifier parameters  $\boldsymbol{\theta}$  as:

$$\ell_{\text{ESupCon}} = \frac{1}{2N + K} \left( \sum_{k=1}^K \ell_{\text{pt}}(\boldsymbol{\theta}_k) + \sum_i^{2N} \ell_{\text{SupCon}}(\mathbf{z}_i, P_i) \right). \quad (6.15)$$

Next we show that our proposed prototype loss  $\ell_{\text{pt}}(\mathbf{z}_i, \boldsymbol{\theta}_k)$  for a given positive pair can be expressed in terms of SupCon loss on that positive pair and CE

loss on the concerned sample . Let us define the following:

$$\begin{aligned}
T &= \mathbf{z}_i^\top \boldsymbol{\theta}_{y_i}, \\
C_1 &= \sum_{k=1}^K \exp(\mathbf{z}_i^\top \boldsymbol{\theta}_k), \\
C_2 &= \sum_{j=1: j \neq i}^{2N} \exp(\mathbf{z}_i^\top \mathbf{z}_j), \\
\exp(\ell_{\text{CE}}(\mathbf{z}_i)) &= \exp(-T + \log(C_1)) \\
&= \exp(-T)C_1, \\
\exp(\ell_{\text{SupCon}}(\mathbf{z}_i, \boldsymbol{\theta}_{y_i})) &= \exp(-T + \log(C_2 + \exp(T))) \\
&= \exp(-T)(C_2 + \exp(T)), \tag{6.16}
\end{aligned}$$

where  $T$  is the tightness term,  $C_1$  is the first contrastive term and  $C_2$  is the second contrastive term,  $\ell_{\text{CE}}(\mathbf{z}_i)$  is the CE loss for a sample  $\mathbf{z}_i$ , and the SupCon loss  $\ell_{\text{SupCon}}(\mathbf{z}_i, \boldsymbol{\theta}_{y_i})$  is estimated after including  $\boldsymbol{\theta}_{y_i}$  into the pool of representations.

Then our loss for the  $(\mathbf{z}_i, \boldsymbol{\theta}_{y_i})$  pair can be expressed as:

$$\begin{aligned}
\ell_{\text{pt}}(\mathbf{z}_i, \boldsymbol{\theta}_{y_i}) &= -T + \log(C_1 + C_2) \\
&= \log(\exp(-T + \log(C_1 + C_2))) \\
&= \log(\exp(-T)(C_1 + C_2)) \\
&= \log(\exp(-T)(C_1 + C_2 + \exp(T) - \exp(T))) \tag{6.17} \\
&= \log(\exp(-T)C_1 + \exp(-T)(C_2 + \exp(T)) \\
&\quad - \exp(-T)\exp(T)) \\
&= \log(\exp(\ell_{\text{CE}}(\mathbf{z}_i)) + \exp(\ell_{\text{SupCon}}(\mathbf{z}_i, \boldsymbol{\theta}_{y_i})) - 1).
\end{aligned}$$

As such, minimizing  $\ell_{\text{pt}}(\mathbf{z}_i, \boldsymbol{\theta}_{y_i})$  is minimizing the log sum exponential (LSE) of cross entropy loss and supervised contrastive loss for a given positive pair  $(\mathbf{z}_i, \boldsymbol{\theta}_{y_i})$ , a smooth approximation to the max function. Note that  $\ell_{\text{pt}}(\mathbf{z}_i, \boldsymbol{\theta}_{y_i}) = 0 \iff \ell_{\text{CE}}(\mathbf{z}_i) = \ell_{\text{SupCon}}(\mathbf{z}_i, \boldsymbol{\theta}_{y_i}) = 0$ .

We refer to the loss in (6.15) as ESupCon, short for Extended Supervised Contrastive learning. In the following, we will extensively compare the different studied loss functions.

## 6.5 Experiments

This section serves to compare the performance of deep models trained under the different objective functions discussed earlier including tightness loss term (6.9) and ESupCon (6.15). Our goal is to perform an extensive evaluation of the different losses behaviour not only under fully supervised setting but also

Method	CIFAR-10	CIFAR-100	Tiny ImageNet	Caltech256	Avg.
CE	95.39	76.36	65.76	55.9	–
*SupCon+CE	95.50 <b>+0.11</b>	75.90 <b>−0.46</b>	65.56 <b>−0.20</b>	57.91 <b>+2.01</b>	<b>+0.36</b>
*SupCon+CE(n)	95.27 <b>−0.12</b>	74.57 <b>−1.79</b>	61.69 <b>−4.07</b>	52.92 <b>−2.98</b>	<b>−1.52</b>
*SupCon+Tt	95.20 <b>−0.19</b>	74.80 <b>−1.56</b>	59.66 <b>−6.1</b>	57.42 <b>1.52</b>	<b>−2.24</b>
SPCE	95.62 <b>+0.23</b>	78.15 <b>+1.79</b>	66.52 <b>+0.76</b>	48.46 <b>−7.44</b>	<b>−1.16</b>
SPCE(M)	95.30 <b>−0.09</b>	77.49 <b>+1.13</b>	66.28 <b>+0.52</b>	48.37 <b>−7.52</b>	<b>−1.49</b>
ESupCon	95.9 <b>+0.51</b>	76.92 <b>+0.56</b>	66.2 <b>+0.44</b>	58.27 <b>+2.37</b>	<b>+0.97</b>

Table 6.1: Accuracy (%) of the different studied and proposed losses on fully labelled datasets. \* indicates the use of a projection head. Absolute gains over cross entropy are reported **blue** and absolute declines in **red**. The last column shows an average improvement or decline over CE, across the datasets.

Method	CIFAR-10			CIFAR-100			Tiny ImageNet			Avg.
	$N = 2K$	$N = 5K$	$N = 10K$	$N = 8K$	$N = 10K$	$N = 20K$	$N = 20K$	$N = 50K$	$N = 70K$	
CE	28.02	69.91	85.08	43.67	51.09	64.31	44.29	57.19	60.94	–
*SupCon+CE	72.27 <b>+44.25</b>	82.37 <b>+12.46</b>	88.03 <b>+2.95</b>	50.96 <b>+7.2</b>	54.49 <b>+3.4</b>	64.39 <b>+0.08</b>	44.00 <b>−0.29</b>	59.24 <b>+2.05</b>	62.88 <b>+1.94</b>	<b>+8.22</b>
*SupCon+CE(n)	71.99 <b>+43.97</b>	82.73 <b>+12.82</b>	87.91 <b>+2.83</b>	50.60 <b>+6.93</b>	53.92 <b>+2.83</b>	63.27 <b>−1.04</b>	43.50 <b>−0.79</b>	57.62 <b>+0.43</b>	59.62 <b>−1.32</b>	<b>+7.41</b>
*SupCon+Tt	72.17 <b>+44.15</b>	82.97 <b>+13.06</b>	87.37 <b>+2.29</b>	51.23 <b>+7.56</b>	54.49 <b>+3.4</b>	64.28 <b>−0.02</b>	43.82 <b>−0.47</b>	52.21 <b>−4.98</b>	57.88 <b>−3.06</b>	<b>+6.88</b>
SPCE	31.81 <b>+3.79</b>	78.60 <b>+8.69</b>	86.15 <b>+1.07</b>	50.09 <b>+6.42</b>	53.78 <b>+2.69</b>	64.82 <b>+0.51</b>	40.94 <b>−3.35</b>	55.70 <b>−1.49</b>	55.49 <b>−5.45</b>	<b>+1.43</b>
ESupCon	74.08 <b>+46.06</b>	83.89 <b>+13.98</b>	88.83 <b>+3.75</b>	48.26 <b>+4.59</b>	52.58 <b>+1.49</b>	63.12 <b>−1.19</b>	44.17 <b>−0.12</b>	58.66 <b>+1.47</b>	62.62 <b>+1.68</b>	<b>+7.97</b>

Table 6.2: Accuracy (%) on CIFAR-10, CIFAR-100 and Tiny ImageNet for a low-sample training scenario, where  $N$  represents the number of samples used for the training. Absolute gains over cross entropy are reported in **blue** and absolute declines in **red**. \* indicates the use of a projection head. The last column shows an average improvement or decline over cross entropy (CE), across the datasets and the settings.

under more challenging yet more plausible settings, namely limited data, imbalanced data and noisy labels settings. For the purpose of this experimental validation, we focus on the object recognition problem.

### 6.5.1 Datasets

We consider Cifar-100, Cifar-10 [269], Tiny ImageNet [270] (a subset of 200 classes from ImageNet [27], rescaled to the  $32 \times 32$ ) datasets and Caltech256 [271]. We refer to the supplementary materials for more results.

### 6.5.2 Methods and Implementation Details

In all experiments we use ResNet50 as a main network and evaluate the following losses:

Method	CIFAR-10			CIFAR-100			Tiny ImageNet			Avg.
	IR = 0.05	IR = 0.1	IR = 0.5	IR = 0.05	IR = 0.1	IR = 0.5	IR = 0.05	IR = 0.1	IR = 0.5	
CE	82.85	87.83	93.99	48.57	54.44	71.19	40.65	46.16	60.30	–
*SupCon+CE	79.94 <b>-2.91</b>	86.86 <b>-0.97</b>	94.34 <b>+0.35</b>	46.79 <b>-1.78</b>	44.21 <b>-10.23</b>	71.13 <b>-0.06</b>	44.96 <b>+4.31</b>	49.57 <b>+3.41</b>	62.45 <b>+2.15</b>	<b>-0.64</b>
*SupCon+CE(n)	47.77 <b>-35.08</b>	47.64 <b>-40.19</b>	90.14 <b>-3.85</b>	40.00 <b>-8.57</b>	40.32 <b>-14.12</b>	55.94 <b>-15.25</b>	35.69 <b>-4.96</b>	35.61 <b>-10.55</b>	37.70 <b>-22.60</b>	<b>-17.24</b>
*SupCon+Tt	85.62 <b>+2.7</b>	88.76 <b>+0.93</b>	94.40 <b>+0.41</b>	54.40 <b>+5.83</b>	56.79 <b>+2.35</b>	70.38 <b>-0.81</b>	44.11 <b>+3.46</b>	47.39 <b>+1.23</b>	57.30 <b>-3.00</b>	<b>+1.46</b>
SPCE	85.62 <b>+4.5</b>	86.94 <b>+2.6</b>	93.95 <b>+0.7</b>	49.59 <b>+6.7</b>	53.78 <b>+5.4</b>	68.61 <b>-1.6</b>	37.27 <b>+2.0</b>	40.55 <b>+1.5</b>	61.14 <b>+3.5</b>	<b>-0.95</b>
ESupCon	86.00 <b>+3.15</b>	89.26 <b>+1.43</b>	94.77 <b>+0.78</b>	52.74 <b>+4.17</b>	58.08 <b>+3.64</b>	71.37 <b>+0.18</b>	45.55 <b>+4.90</b>	50.90 <b>+4.74</b>	63.08 <b>+2.78</b>	<b>+2.86</b>

Table 6.3: Accuracy (%) on CIFAR-10, CIFAR-100 and Tiny ImageNet for an imbalanced training scenario, where IR represents the rate of imbalance. Absolute gains over cross entropy are reported **blue** and absolute declines in **red**. \* indicates the use of a projection head. The last column shows an average improvement or decline over cross entropy (CE), across the datasets and the settings.

CE: we optimize the network parameters using the standard CE loss. For the SupCon loss [82], we use the publicly available implementation, which uses L2-normalized outputs of a multi-layer head (FC, ReLU, FC), a projection head, on top of the embeddings used for classification. We learn the classifier parameters using: i) Cross entropy loss (SupCon+CE), on the linear layer after optimizing minimizing SupCon loss. ii) For the sake of fair comparison with other losses, we consider also cross entropy loss with no bias term, normalized embeddings and normalized classifier weights. We denote this variant by SupCon+CE(n). iii) Tightness loss (SupCon+Tt), where we optimize the parameters of a linear classifier using (6.9) during the optimization of the rest of the network (projection head + backbone) with SupCon loss. Note that the gradients of the tightness loss are not propagated to the rest of the network. SPCE: we optimize the backbone with SPCE loss (6.7) and the classifier weights with the tightness term (6.9). We also show the performance with directly assigning the weights to the mean of each class samples SPCE(M).

Our ESupCon: with (6.15) we optimize jointly a linear classifier and the backbone parameters.

Note that SupCon+CE, SupCon+CE(n) and SupCon+Tt use a projection head, unlike CE, SPCE and ESupCon. All studied variants benefit from the same type of data augmentations and hyper-parameters were estimated on Cifar-10 dataset and fixed for the rest. We refer to the supplementary materials for more details.

### 6.5.3 Fully Supervised Classification

We first start by comparing the different studied methods on the standard classification setting while leveraging all the labelled training data of each dataset. Table 6.1 shows the average test accuracy at the end of the training on the three considered datasets.

First, ESupCon outperforms CE training alone, using the same number

of parameters. SupCon+CE improves over CE. SupCon+Tt is comparable to SupCon+CE(n).

ESupCon shows the best performance on all four datasets. Except from Caltech dataset, SPCE achieves superior results to CE. When assigning the classifier weights directly to the mean of the features, SPCE(M), results are slightly inferior to the use of our tightness loss (6.9) for training the classifier parameters. For the rest the of experiments, we show only SPCE, using the suggested tightness term to train the classifier parameters.

## 6.5.4 Classification in Low-Sample Scenario

Method	CIFAR-10			CIFAR-100			Tiny ImageNet			Avg.
	NR = 0.5	NR = 0.3	NR = 0.2	NR = 0.5	NR = 0.3	NR = 0.2	NR = 0.5	NR = 0.3	NR = 0.2	
CE	60.88	87.08	88.93	35.47	56.57	64.93	31.55	49.62	55.40	-
*SupCon+CE	48.08 <b>-12.8</b>	74.47 <b>-12.61</b>	85.94 <b>-2.99</b>	34.78 <b>-0.69</b>	58.06 <b>+1.49</b>	65.57 <b>+0.64</b>	31.81 <b>+0.26</b>	46.20 <b>+3.42</b>	54.74 <b>+0.66</b>	<b>-3.42</b>
*SupCon+CE(n)	46.35 <b>-14.53</b>	77.42 <b>-9.66</b>	87.45 <b>-1.48</b>	33.55 <b>-1.92</b>	62.44 <b>+5.87</b>	67.87 <b>+2.94</b>	28.88 <b>-2.67</b>	54.64 <b>+5.02</b>	58.62 <b>+3.22</b>	<b>-1.47</b>
*SupCon+Tt	58.05 <b>-2.83</b>	89.70 <b>+2.62</b>	90.66 <b>+1.73</b>	37.23 <b>+1.76</b>	67.76 <b>+11.19</b>	69.41 <b>+4.48</b>	28.67 <b>-2.88</b>	54.81 <b>+5.19</b>	57.93 <b>+2.53</b>	<b>+2.64</b>
SPCE	65.63 <b>+4.75</b>	88.77 <b>+1.69</b>	88.93 <b>+0.0</b>	36.56 <b>+1.09</b>	60.35 <b>+3.78</b>	65.75 <b>+0.82</b>	25.27 <b>-6.28</b>	42.45 <b>-7.17</b>	49.52 <b>-5.88</b>	<b>-0.80</b>
ESupCon	59.18 <b>-1.70</b>	88.15 <b>+1.07</b>	90.92 <b>+1.99</b>	37.04 <b>+1.57</b>	62.94 <b>+6.37</b>	65.81 <b>+0.87</b>	32.555 <b>+1.0</b>	52.80 <b>+3.18</b>	56.79 <b>+1.39</b>	<b>+1.75</b>

Table 6.4: Accuracy (%) on CIFAR-10, CIFAR-100 and Tiny ImageNet for a noisy training scenario, NR represents the rate of noise. Absolute gains over cross entropy are reported in blue and absolute declines in red. \* indicates the use of a projection head. The last column shows an average improvement or decline over cross entropy (CE), across the datasets and the settings.

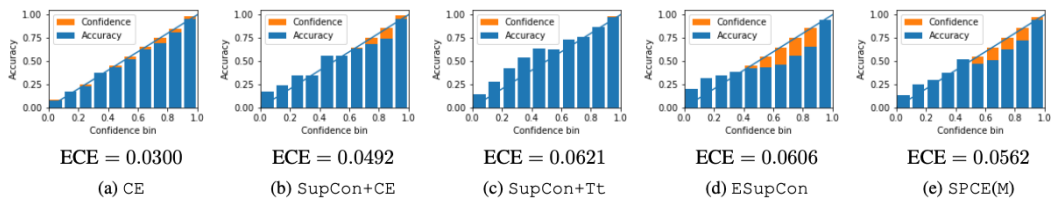


Figure 6.2: Reliability Diagrams and Expected Calibration Error of probabilistic classifiers learned with different studied loss functions and further calibrated by temperature scaling.

After studying the fully labelled scenario, here, we are interested in the performance under limited data setting. Our goal is to see how prone each method is to overfitting in low data regime and whether significant differences can be observed among the different alternatives. Table 6.2 reports the average test accuracy on Cifar-10, Cifar-100 and Tiny ImageNet using different numbers of training samples ( $N$ ).

While CE performance is comparable to other losses on the full data scenario, here it is significantly lower than other competitors with a gap increasing as the sample size gets smaller. Except from Tiny ImageNet, **SupCon+Tt** shows comparable performance to **SupCon+CE** and is slightly inferior (0.5%) to **SupCon+CE(n)** on average. **SPCE** results are better than CE on Cifar-10 and Cifar-100. **ESupCon** improves significantly over CE while being comparable with **SupCon+CE**, however, with no projection head. **ESupCon** is much more robust than **SPCE** in this setting.

### 6.5.5 Classification under Imbalanced Data

Our goal is to compare the performance of a model trained by the different studied losses under various challenging settings beside the standard fully supervised setting. Here, we examine the scenario where training data are not uniformly distributed. Some classes are undersampled while others are oversampled. Specifically, we want to test the ability of the different losses to cope with this data nature and learn the underrepresented classes. We simulate this scenario by altering the training data in which half of the categories are underrepresented with a number of samples equals to the imbalance rate (IR) of other categories samples. The test set on which we report the average accuracy remains balanced.

Table 6.3 reports the average test accuracy of models trained to minimize the different losses on the three considered datasets. For each dataset we consider imbalance rates of 0.05, 0.1, and 0.5 where, for example, an imbalance rate of 0.1 means that the size of undersampled classes samples is 0.1 compared to the oversampled classes size.

Here it seems that **SupCon+CE** doesn't improve over CE alone. **SPCE** results are marginally lower than CE. Our two proposed losses **SupCon+Tt** and **ESupCon** exhibit more robust and powerful performance compared to CE with **ESupCon** performing the best.

### 6.5.6 Classification under Noisy Data

We continue our investigation on the different losses performance under challenging setting and test another interesting scenario: classification with noisy labels. We want to test the ability of the different training regimes to learn generalizable decision boundaries in spite of the presence of wrongly labelled samples. To simulate this scenario, during training a percentage of the training data, denoted by noise rate (NR), is associated with wrong labels (shuffled labels). As in the previous experiments, we report the results on the standard, correctly labelled, test set. Table 6.4 reports the average test accuracy on Cifar-10, Cifar-100 and Tiny ImageNet with noise rates of (0.2, 0.3, 0.5). Here we obtained similar results to the imbalanced settings, **SupCon+CE** doesn't consistently improve over CE, same applies for **SPCE**. Our both proposed losses



improve over CE with SupCon+Tt performing the best here.

### 6.5.7 General Remarks

We note the following on the shown results of the different losses: CE training after SupCon pretraining (SupCon+CE) improves over standard CE in full and low data regime. However, deploying CE to learn the classifier with or without SupCon pretraining is sensitive to noise and data imbalance. Interestingly, our proposed tightness term is more effective on these two scenarios, however inferior on the full and low data regime. In all studied settings, our proposed ESupCon loss improves over CE and over (SupCon+CE) on the challenging imbalanced and noisy settings. In Supplementary we discuss the computational complexity of the different losses and their sensitivity to hyper-parameters.

### 6.5.8 Classifier Outputs as Posterior Probabilities

To access the interpretation of the classifier outputs as estimates of posterior probabilities  $\hat{p}(\text{class}|\text{observation})$ , we calibrated the outputs by temperature scaling [272] – we estimated the temperature on a holdout set (20% of the test set) and computed the reliability diagram and the expected calibration error (ECE) on the remaining test samples of Cifar-100 dataset. Results are shown in Figure 6.2: while the standard CE loss has the lowest calibration error, all other calibrated classifiers provide reliable predictions, an interesting result given the shown performance advantage.

## 6.6 Conclusion

In this work, we derive novel, robust objective functions, inspired by new evidence showing that contrastive losses improve performance over CE. Driven by the question of whether cross entropy loss is the best option to train jointly a good representation and powerful, generalizable, decision boundaries, we start from a recent approximation to cross entropy loss (SPCE) with pairwise training of representation where classifier weights can be assigned to the mean of each class features. We then suggest to learn the classifier weights under only a tightness term jointly with SupCon representation training or SPCE. Next, we propose an extension to SupCon, where the classifier weights are treated as learnable prototypes in the same space as the samples embeddings, and where data points form positive pairs with their classes prototypes. We show that the proposed loss for a given pair  $(\mathbf{z}_i, \boldsymbol{\theta}_k)$  is a smooth approximation to the maximum of the CE and SupCon losses on that pair. To this point, we test the performance of models trained with the different discussed losses under different challenging settings. We show that the proposed extensions demonstrate more robust and stable performance across different settings and datasets. As a future work, we plan to extend the experiments to object

detection and image segmentation problems, as well as to test the discussed losses on Out-Of-Distribution and Continual Learning benchmarks.

# Chapter 7

## Conclusions

The thesis contributes to bridging the gap between the user-defined objective and the training loss when the objective is known and non-differentiable. To this end, we proposed a technique to learn the surrogate of a decomposable and non-differentiable evaluation metric (Chapter 3) [1]. The surrogate is learned jointly with the task-specific model in an alternating training setup. The approach showed a relative improvement of up to 39% on the total edit distance for scene text recognition and 4.25% on  $F_1$  score for scene text detection.

After observing the noisy predictions from the surrogate at the initial stage of the training, we designed an approach to train robustly with the learned surrogate (Chapter 4) [2]. The method filters out the samples that are hard for the surrogate. With this approach, we observed the merits of training a scene text recognition model using a learned surrogate of edit distance. We attained an average relative improvement of 11.2% on the total edit distance and an error reduction of 9.5% on accuracy on several benchmarks. For comparison, our previous approach [1] obtains an relative improvement of 6.05% on the total edit distance.

We introduced a surrogate loss for recall@k, a non-decomposable and non-differentiable evaluation metric (Chapter 5) [3]. When combined with a novel and efficient mixup technique and training on large batch sizes, the proposed approach attained state-of-the-art results on several metric learning benchmarks and instance-level search. Further, when combined with kNN classifier, it can also serve as an effective tool for fine-grained recognition where it substantially outperforms direct classification methods equipped with performance-enhancing techniques [4].

We put forward an approach for supervised contrastive classification that jointly learns the parameters of the classifier and the backbone (Chapter 6) [14]. This approach leverages the robustness of contrastive training and maintains the probabilistic interpretation useful for several calibration tasks. The method outperformed standard cross-entropy and supervised contrastive losses and was shown to be robust in various challenging settings such as class imbalance, label corruption, and training with a low number of samples.

# Appendix A

## Abstrakt

*(Automatic translation by Google Translate)*

Mnoho důležitých úkolů počítačového vidění je přirozeně formulováno s nediferencovatelným cílem. Proto standardní, dominantní trénovací postup neuronové sítě není použitelný, protože zpětné šíření vyžaduje gradienty cíle vzhledem k výstupu modelu. Většina metod hlubokého učení obchází problém neoptimálně použitím proxy ztráty pro trénink, který byl původně navržen pro jiný úkol a není přizpůsoben specifikům cíle. Funkce proxy ztráty se mohou, ale nemusí dobře shodovat s původním nediferencovatelným cílem. Pro nový úkol musí být navržen vhodný proxy, který nemusí být proveditelný pro laika. Tato práce přináší čtyři hlavní příspěvky k překlenutí propasti mezi nediferencovatelným cílem a funkcí ztráty tréninku. Ztrátovou funkci v celé práci označujeme jako náhradní ztrátu, pokud se jedná o diferencovatelnou aproximaci nediferencovatelného cíle.

Nejprve navrhujeme přístup k učení diferencovatelné náhrady rozložitelné a nediferencovatelné vyhodnocovací metriky. Náhradník se učí společně s modelem specifickým pro úkol střídavým způsobem. Tento přístup je ověřen na dvou praktických úlohách rozpoznávání a detekce textu scény, kde se náhradník učí aproximaci vzdálenosti úprav a průsečíku přes spojení. V nastavení po vyladění, kde je model trénovaný se ztrátou proxy dále trénován s naučeným náhradníkem, navrhovaná metoda ukazuje relativní zlepšení až o 39 % celkové vzdálenosti úprav pro rozpoznání textu scény a 4,25 % na  $F_1$  skóre za detekci textu scény.

Za druhé, vylepšená verze tréninku s naučeným náhradníkem, kde jsou odfiltrovány tréninkové vzorky, které jsou pro náhradníka těžké. Tento přístup je ověřen pro rozpoznávání textu scény. Překonává náš předchozí přístup a dosahuje průměrného zlepšení o 11,2% celkové vzdálenosti úprav a snížení chyb o 9,5% v přesnosti v několika oblíbených benchmarcích. Všimněte si, že dvě navrhované metody pro učení náhradníka a školení s náhradníkem nevytvářejí žádné předpoklady o daném úkolu a mohou být potenciálně rozšířeny na nové úkoly.

Za třetí, pro *reminiscenci@k*, nerozložitelnou a nediferencovatelnou vyhodnocovací metriku, navrhujeme ručně vytvořenou náhradu, která zahrnuje navrhování diferencovatelných verzí operací třídění a počítání. Je také navržena účinná kombinační technika pro učení metriky, která míchá skóre podobnosti namísto vkladacích vektorů. Navrhovaná náhrada dosahuje nejmodernějších výsledků na několika metrických učeních a srovnávacích testech vyhledávání na úrovni instancí v kombinaci se školením na velkých dávkách. Dále v kombinaci s klasifikátorem kNN slouží také jako účinný nástroj pro jemnozrné rozpoznávání, kde překonává přímé klasifikační metody.

Za čtvrté navrhujeme ztrátovou funkci nazvanou *Extended SupCon*, která společně trénuje parametry klasifikátoru a páteře pro řízenou kontrastní klasifikaci. Navrhovaný přístup těží z robustnosti kontrastivního učení a zachovává pravděpodobnostní interpretaci jako *soft-max* predikci. Empirické výsledky ukazují účinnost našeho přístupu v náročných podmínkách, jako je třídní nerovnováha, korupce štítků a školení s málo označenými údaji.

Celkově přínosy této práce činí trénování neuronových sítí škálovatelnějším – na nové úkoly téměř bezpracně, když je vyhodnocovací metrika rozložitelná, což výzkumníkům pomůže s novými úkoly. Pro nerozložitelné vyhodnocovací metriky lze pro vytváření nových náhradních prvků použít také diferencovatelné komponenty vyvinuté pro náhradního prvku pro *odvolání@k*, jako je třídění a počítání.

(Automatic translation by ChatGPT)

Mnoho důležitých úkolů v oblasti počítačového vidění je přirozeně formulováno s nenadiferentovatelným cílem. Standardní a dominantní tréninkový postup neuronové sítě tak není použitelný, protože zpětná propagace vyžaduje gradienty objektivu vzhledem k výstupu modelu. Většina metod hlubokého učení tento problém řeší podobným způsobem pomocí proxy ztráty pro trénování, která byla původně navržena pro jiný úkol a není přizpůsobena specifikám cíle. Funkce proxy ztráty mohou nebo nemusí dobře korespondovat s původní nenadiferentovatelným cílem. Pro nový úkol musí být navržena vhodná proxy, což pro neoborníka nemusí být proveditelné. Tato práce přináší čtyři hlavní přínosy pro překlenutí rozdílu mezi nenadiferentovatelným cílem a ztrátovou funkcí pro trénování. V celé práci označujeme ztrátovou funkci jako náhradní, pokud je diferencovatelnou aproximací nenadiferentovatelného cíle.

Nejprve navrhuje přístup pro učení diferencovatelného náhradního modelu pro rozložitelnou a nespojitou hodnotící metriku. Náhrada je společně s úkolspecifickým modelem učena střídavým způsobem. Přístup je ověřen na dvou praktických úlohách rozpoznávání a detekce textu v scéně, kde náhrada modeluje aproximaci editační vzdálenosti a překryvu-union, odpovídajících. V post-tuningovém nastavení, kde model trénovaný pomocí proxy ztráty je dále trénován s naučenou náhradou, navrhaná metoda ukazuje relativní zlepšení až o 39% celkové editační vzdálenosti pro rozpoznávání textu v scéně a 4.25%  $F_1$  skóre pro detekci textu v scéně.

Druhá část našeho návrhu zahrnuje vylepšenou verzi tréninku pomocí naučeného náhradního modelu, kdy jsou filtrace tvrdých tréninkových vzorků založená na náhradním modelu. Tento postup byl ověřen pro rozpoznávání textu v obrazech. Tento nový postup předčil naše předchozí řešení a dosáhl průměrného zlepšení celkové editační vzdálenosti o 11.2% a snížení chybovosti o 9.5% v přesnosti na několika populárních testech. Poznamenejme, že oba navržené postupy, a to učení náhradního modelu a trénink s náhradním modelem, nekladou žádné předpoklady na řešený úkol a mohou být potenciálně rozšířeny na nové úkoly.

Třetí přístup se týká metriky  $\text{recall}@k$ , což je nedekomponovatelná a nespojitá metrika. Navrhujeme ručně vytvořený náhradní funkční prvek, který zahrnuje návrh diferencovatelných verzí řazení a počítání operací. Dále navrhuje efektivní techniku mixup pro učení metrik, která míchá podobnostní skóre místo vektorů vnoření. Navrhovaný náhradní funkční prvek dosahuje výsledků na špičkové úrovni na několika měřících a vyhledávacích úlohách na úrovni instancí, když se kombinuje s trénováním na velkých dávkách. Když se navíc použije kNN klasifikátor, slouží také jako účinný nástroj pro jemné rozlišení, kdy překonává přímé metody klasifikace.

Čtvrtým příspěvkem této práce je návrh loss funkce s názvem Extended SupCon, která společně trénuje klasifikátor a parametry základní sítě pro supervised contrastive classification. Navržený přístup využívá robustnosti con-

trastive learning a zachovává pravděpodobnostní interpretaci jako u soft-max predikce. Empirické výsledky ukazují účinnost našeho přístupu i v náročných podmínkách, jako je nerovnováha tříd, zkreslené štítky a trénování s málem označených dat.

Celkově přínosy této práce umožňují škálovat trénování neuronových sítí pro nové úlohy v téměř bezúdržbovém režimu, pokud je vyhodnocovací metrika rozložitelná, což pomůže výzkumníkům s novými úkoly. Pro metriky vyhodnocování, které nejsou rozložitelné, lze komponenty vyvinuté pro recall@k surrogate, jako je třídění a počítání, použít k vytváření nových surrogate.

# Bibliography

- [1] Y. Patel, T. Hodan, and J. Matas, “Learning surrogates via deep embedding”, in *ECCV*, 2020.
- [2] Y. Patel and J. Matas, “Feds-filtered edit distance surrogate”, in *ICDAR*, 2021.
- [3] Y. Patel, G. Toliás, and J. Matas, “Recall@k surrogate loss with large batches and similarity mixup”, in *CVPR*, 2022.
- [4] L. Pícek, M. Šulc, Y. Patel, and J. Matas, “Plant recognition by ai: Deep neural nets, transformers, and knn in deep embeddings”, *Frontiers in Plant Science*, 2022.
- [5] N. Nayef, Y. Patel, M. Busta, *et al.*, “Icdar2019 robust reading challenge on multi-lingual scene text detection and recognition–rrc-mlt-2019”, *arXiv preprint arXiv:1907.00945*, 2019.
- [6] S. Djukanović, Y. Patel, J. Matas, and T. Virtanen, “Neural network-based acoustic vehicle counting”, in *European Signal Processing Conference (EUSIPCO)*, 2021.
- [7] T. Wei, Y. Patel, J. Matas, and D. Barath, “Generalized differentiable RANSAC”, *arXiv preprint arXiv:2212.13185*, 2023.
- [8] Š. Šimsa, M. Šulc, M. Uříčář, *et al.*, “Docile benchmark for document information localization and extraction”, *arXiv preprint arXiv:2302.05658*, 2023.
- [9] Y. Patel, S. Appalaraju, and R Manmatha, “Saliency driven perceptual image compression”, in *WACV*, 2021.
- [10] Y. Patel, S. Appalaraju, and R Manmatha, “Hierarchical auto-regressive model for image compression incorporating object saliency and a deep perceptual loss”, *arXiv preprint arXiv:2002.04988*, 2020.
- [11] Y. Patel, S. Appalaraju, and R Manmatha, “Deep perceptual compression”, *arXiv preprint arXiv:1907.08310*, 2019.
- [12] Y. Patel, S. Appalaraju, and R Manmatha, “Human perceptual evaluations for image compression”, *arXiv preprint arXiv:1908.04187*, 2019.
- [13] Y. Patel, Y. Xie, Y. Zhu, S. Appalaraju, and R Manmatha, “Simcon loss with multiple views for text supervised semantic segmentation”, *arXiv preprint arXiv:2302.03432*, 2023.



- [14] R. Aljundi, Y. Patel, M. Sulc, D. Olmeda, and N. Chumerin, “Contrastive classification and representation learning with probabilistic interpretation”, *AAAI*, 2023.
- [15] Š. Šimsa, M. Šulc, M. Skalický, Y. Patel, and A. Hamdi, “Docile 2023 teaser: Document information localization and extraction”, in *ECIR*, 2023.
- [16] F. Radenovic, A. Dubey, A. Kadian, *et al.*, “Filtering, distillation, and hard negatives for vision-language pre-training”, 2023.
- [17] J. Yu, Y. Jiang, Z. Wang, Z. Cao, and T. Huang, “Unitbox: An advanced object detection network”, in *ACM-MM*, 2016.
- [18] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks”, *TPAMI*, 2017.
- [19] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection”, in *CVPR*, 2016.
- [20] Y. Patel, T. Hodaň, and J. Matas, “Learning surrogates via deep embedding”, in *ECCV*, 2020.
- [21] Y. Patel and J. Matas, “Feds-filtered edit distance surrogate”, in *ICDAR*, 2021.
- [22] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks”, in *ICML*, 2006.
- [23] A. Brown, W. Xie, V. Kalogeiton, and A. Zisserman, “Smooth-ap: Smoothing the path towards large-scale image retrieval”, in *ECCV*, 2020.
- [24] Y. Patel, L. Gomez, M. Rusiňol, D. Karatzas, and C. Jawahar, “Self-supervised visual representations for cross-modal retrieval”, in *ICMR*, 2019.
- [25] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering”, in *CVPR*, 2015.
- [26] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl, “Sampling matters in deep embedding learning”, in *ICCV*, 2017.
- [27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database”, in *CVPR*, 2009.
- [28] T. Lin, M. Maire, S. J. Belongie, *et al.*, “Microsoft COCO: common objects in context”, in *ECCV*, 2014.
- [29] B. Zhou, H. Zhao, X. Puig, *et al.*, “Semantic understanding of scenes through the ade20k dataset”, *IJCV*, 2019.
- [30] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale”, in *ICLR*, 2021.

- [31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding”, *arXiv preprint arXiv:1810.04805*, 2018.
- [32] J. Li, R. Selvaraju, A. Gotmare, S. Joty, C. Xiong, and S. C. H. Hoi, “Align before fuse: Vision and language representation learning with momentum distillation”, *NeurIPS*, 2021.
- [33] A. Radford, J. W. Kim, C. Hallacy, *et al.*, “Learning transferable visual models from natural language supervision”, in *ICML*, 2021.
- [34] M. Singh, L. Gustafson, A. Adcock, *et al.*, “Revisiting Weakly Supervised Pre-Training of Visual Perception Models”, in *CVPR*, 2022.
- [35] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition”, in *CVPR*, 2016.
- [36] S. Jadon, “A survey of loss functions for semantic segmentation”, in *CIBCB*, 2020.
- [37] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization”, *arXiv preprint arXiv:1412.6980*, 2014.
- [38] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization”, in *ICLR*, 2019.
- [39] S. Ruder, “An overview of gradient descent optimization algorithms”, *arXiv preprint arXiv:1609.04747*, 2016.
- [40] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization.”, *Journal of machine learning research*, 2011.
- [41] M. D. Zeiler, “Adadelta: An adaptive learning rate method”, *arXiv preprint arXiv:1212.5701*, 2012.
- [42] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks”, in *NeurIPS*, 2012.
- [43] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition”, *arXiv preprint arXiv:1409.1556*, 2014.
- [44] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, “Microsoft coco: Common objects in context”, in *ECCV*, 2014.
- [45] T. Elsken, J. H. Metzen, and F. Hutter, “Neural architecture search: A survey”, *arXiv preprint arXiv:1808.05377*, 2018.
- [46] M. S. Ryoo, A. Piergiovanni, M. Tan, and A. Angelova, “Assemblenet: Searching for multi-stream neural connectivity in video architectures”, *NeurIPS*, 2019.
- [47] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning”, *arXiv preprint arXiv:1611.01578*, 2016.

- [48] S. Gidaris, P. Singh, and N. Komodakis, “Unsupervised representation learning by predicting image rotations”, *ICLR*, 2018.
- [49] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, “Deep clustering for unsupervised learning of visual features”, in *ECCV*, 2018.
- [50] L. Gomez, Y. Patel, M. Rusiñol, D. Karatzas, and C. Jawahar, “Self-supervised learning of visual features through embedding images into text topic spaces”, in *CVPR*, 2017.
- [51] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning”, in *CVPR*, 2020.
- [52] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, “Masked autoencoders are scalable vision learners”, in *CVPR*, 2022.
- [53] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting”, 2014.
- [54] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift”, *arXiv preprint arXiv:1502.03167*, 2015.
- [55] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization”, *arXiv preprint arXiv:1607.06450*, 2016.
- [56] Y. Bengio, A. Courville, and P. Vincent, “Practical recommendations for gradient-based training of deep architectures”, in *NeurIPS*, 2012.
- [57] L. N. Smith, “Learning rate schedules for faster convergence”, *Deep Learning*, 2017.
- [58] S. N. Law, D. P. Wipf, and A. Gupta, “Grid search hyperparameter tuning for deep learning models: A review”, *Neural Computing and Applications*, 2020.
- [59] A. Klein, S. Falkner, and F. Hutter, “Hyperparameter optimization in machine learning: A review”, *JMLR*, 2019.
- [60] P. Tang, X. Wang, W. Liu, and J. Wang, “Multi-source domain adaptation for semantic segmentation”, in *CVPR*, 2019.
- [61] L. Li, Z. Lu, Y. Huang, and J. Tang, “D-sym: Diversified synthesis for domain adaptive object detection”, in *CVPR*, 2020.
- [62] W. Huang, H. Ling, and W. Lin, “Improving unsupervised domain adaptation by self-training with deep reconstruction”, in *ECCV*, 2020.
- [63] H. Liu, M. Wan, L. Zheng, Z. Yuan, J. Qin, and Z. Zhu, “Rethinking the value of labels for improving class-imbalanced learning”, in *CVPR*, 2021.
- [64] Y. Liu, Y. Liu, W. Xia, and D. Lin, “Self-supervised domain adaptation for computer vision tasks”, in *CVPR*, 2021.

- [65] L. Zheng, S. Wang, and P. O. Ogunbona, “Cross-domain cascaded deep feature learning”, in *CVPR*, 2015.
- [66] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, “Learning transferable features with deep adaptation networks”, in *ICML*, 2015.
- [67] S. Hussain, F. Porikli, and J. K. Tsotsos, “Deep domain confusion: Maximizing for domain invariance”, in *ECCV*, 2016.
- [68] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, and H. Larochelle, “Domain-adversarial training of neural networks”, *JMLR*, 2016.
- [69] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, and H. Larochelle, “Un-supervised domain adaptation by backpropagation”, in *ICML*, 2015.
- [70] D. Mahajan, R. Girshick, V. Ramanathan, *et al.*, “Exploring the limits of weakly supervised pretraining”, in *ECCV*, 2018.
- [71] D. Zhou, J. Fang, X. Song, *et al.*, “Iou loss for 2d/3d object detection”, in *2019 International Conference on 3D Vision (3DV)*, 2019.
- [72] R. Franzen, “Kodak lossless true color image suite”, *source: <http://r0k.us/graphics/kodak>*, 1999.
- [73] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool, “Conditional probability models for deep image compression”, in *CVPR*, 2018.
- [74] J. Ballé, V. Laparra, and E. P. Simoncelli, “End-to-end optimized image compression”, *arXiv preprint arXiv:1611.01704*, 2016.
- [75] F. Bellard. “Bpg image format”. (2014), [Online]. Available: <http://bellard.org/bpg/>.
- [76] A. Skodras, C. Christopoulos, and T. Ebrahimi, “The jpeg 2000 still image compression standard”, *IEEE Signal processing magazine*, 2001.
- [77] J. Baek, G. Kim, J. Lee, *et al.*, “What is wrong with scene text recognition model comparisons? dataset and model analysis”, *ICCV*, 2019.
- [78] R. Litman, O. Anshel, S. Tsiper, R. Litman, S. Mazor, and R. Manmatha, “Scatter: Selective context attentional scene text recognizer”, in *CVPR*, 2020.
- [79] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, “Generalized intersection over union: A metric and a loss for bounding box regression”, in *CVPR*, 2019.
- [80] J. Yu, Y. Jiang, Z. Wang, Z. Cao, and T. Huang, “Unitbox: An advanced object detection network”, in *ACM-MM*, 2016.
- [81] R. Hadsell, S. Chopra, and Y. LeCun, “Dimensionality reduction by learning an invariant mapping”, in *CVPR*, 2006.
- [82] P. Khosla, P. Teterwak, C. Wang, *et al.*, “Supervised contrastive learning”, *arXiv preprint arXiv:2004.11362*, 2020.

- [83] M Kumar, B. Packer, and D. Koller, “Self-paced learning for latent variable models”, *NeurIPS*, 2010.
- [84] A. Mishra, K. Alahari, and C. Jawahar, “Scene text recognition using higher order language priors”, in *BMVC*, 2012.
- [85] K. Wang, B. Babenko, and S. Belongie, “End-to-end scene text recognition”, in *ICCV*, 2011.
- [86] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young, “Icdar 2003 robust reading competitions”, in *ICDAR*, 2003.
- [87] D. Karatzas, F. Shafait, S. Uchida, *et al.*, “Icdar 2013 robust reading competition”, in *ICDAR*, 2013.
- [88] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, *et al.*, “Icdar 2015 competition on robust reading”, in *ICDAR*, 2015.
- [89] T. Quy Phan, P. Shivakumara, S. Tian, and C. Lim Tan, “Recognizing text with perspective distortion in natural scenes”, in *ICCV*, 2013.
- [90] A. Risnumawan, P. Shivakumara, C. S. Chan, and C. L. Tan, “A robust arbitrary text detection system for natural scene images”, *Expert Systems with Applications*, 2014.
- [91] G. Van Horn, O. Mac Aodha, Y. Song, *et al.*, “The inaturalist species classification and detection dataset”, in *CVPR*, 2018.
- [92] E.-J. Ong, S. Husain, and M. Bober, “Siamese network of deep fisher-vector descriptors for image retrieval”, in *arXiv*, 2017.
- [93] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, “3d object representations for fine-grained categorization”, in *ICCV workshops*, 2013.
- [94] H. Liu, Y. Tian, Y. Wang, L. Pang, and T. Huang, “Deep relative distance learning: Tell the difference between similar vehicles”, in *CVPR*, 2016.
- [95] F. Radenović, A. Iscen, G. Toliás, Y. Avrithis, and O. Chum, “Revisiting oxford and paris: Large-scale image retrieval benchmarking”, in *CVPR*, 2018.
- [96] E. Eban, M. Schain, A. Mackey, A. Gordon, R. Rifkin, and G. Elidan, “Scalable learning of non-decomposable objectives”, in *AISTAT*, 2017.
- [97] A. Rakotomamonjy, “Optimizing area under roc curve with svms.”, in *ROCAI*, 2004.
- [98] M. Berman, A. R. Triki, and M. B. Blaschko, “The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks”, in *CVPR*, 2018.
- [99] S. Puthiya Parambath, N. Usunier, and Y. Grandvalet, “Optimizing f-measures by cost-sensitive classification”, *NeurIPS*, 2014.

- [100] H. Xu, H. Zhang, Z. Hu, X. Liang, R. Salakhutdinov, and E. Xing, “Autoloss: Learning discrete schedules for alternate optimization”, *arXiv preprint arXiv:1810.02442*, 2018.
- [101] L. Wu, F. Tian, Y. Xia, *et al.*, “Learning to teach with dynamic loss functions”, in *NeurIPS*, 2018.
- [102] T. Hazan, J. Keshet, and D. A. McAllester, “Direct loss minimization for structured prediction”, in *NeurIPS*, 2010.
- [103] Y. Song, A. Schwing, R. Urtasun, *et al.*, “Training deep neural networks via direct loss minimization”, in *ICML*, 2016.
- [104] C. Huang, S. Zhai, P. Guo, and J. Susskind, “Metricopt: Learning to optimize black-box evaluation metrics”, in *CVPR*, 2021.
- [105] M. V. Pogančić, A. Paulus, V. Musil, G. Martius, and M. Rolínek, “Differentiation of blackbox combinatorial solvers”, in *ICLR*, 2020.
- [106] M. Rolínek, V. Musil, A. Paulus, M. Vlastelica, C. Michaelis, and G. Martius, “Optimizing rank-based metrics with blackbox differentiation”, in *CVPR*, 2020.
- [107] T. Hodan, F. Michel, E. Brachmann, *et al.*, “Bop: Benchmark for 6d object pose estimation”, in *ECCV*, 2018.
- [108] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks”, in *NeurIPS*, 2015.
- [109] G.-S. Xia, X. Bai, J. Ding, *et al.*, “Dota: A large-scale dataset for object detection in aerial images”, in *CVPR*, 2018.
- [110] M. Kristan, J. Matas, A. Leonardis, *et al.*, “The seventh visual object tracking vot2019 challenge results”, in *ICCV Workshops*, 2019.
- [111] J. Ma, W. Shao, H. Ye, *et al.*, “Arbitrary-oriented scene text detection via rotation proposals”, *IEEE Transactions on Multimedia*, 2018.
- [112] M. Bušta, Y. Patel, and J. Matas, “E2e-mlt-an unconstrained end-to-end method for multi-language scene text”, in *ACCV*, 2018.
- [113] S. M. Azimi, E. Vig, R. Bahmanyar, M. Körner, and P. Reinartz, “Towards multi-class object detection in unconstrained remote sensing imagery”, in *ACCV*, 2018.
- [114] J. Lee, S. Cho, and S.-K. Beack, “Context-adaptive entropy model for end-to-end optimized image compression”, *ICLR*, 2019.
- [115] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, “Variational image compression with a scale hyperprior”, *ICLR*, 2018.
- [116] C. Ledig, L. Theis, F. Huszár, *et al.*, “Photo-realistic single image super-resolution using a generative adversarial network”, in *CVPR*, 2017.
- [117] C. Dong, C. C. Loy, K. He, and X. Tang, “Image super-resolution using deep convolutional networks”, *TPAMI*, 2015.

- [118] Z. Wang, E. P. Simoncelli, and A. C. Bovik, “Multiscale structural similarity for image quality assessment”, in *ACSSC*, 2003.
- [119] L. Berrada, A. Zisserman, and M. P. Kumar, “Smooth loss functions for deep top-k classification”, *ICLR*, 2018.
- [120] M. Lapin, M. Hein, and B. Schiele, “Loss functions for top-k error: Analysis and insights”, in *CVPR*, 2016.
- [121] R. Gomez, B. Shi, L. Gomez-Bigorda, *et al.*, “ICDAR2017 robust reading challenge on coco-text”, in *ICDAR*, 2017.
- [122] H. Kato, Y. Ushiku, and T. Harada, “Neural 3d mesh renderer”, in *CVPR*, 2018.
- [123] E. Agustsson, F. Mentzer, M. Tschannen, *et al.*, “Soft-to-hard vector quantization for end-to-end learning compressible representations”, in *NeurIPS*, 2017.
- [124] R. Prabhavalkar, T. N. Sainath, Y. Wu, *et al.*, “Minimum word error rate training for attention-based sequence-to-sequence models”, in *ICASSP*, 2018.
- [125] M. A. Rahman and Y. Wang, “Optimizing intersection-over-union in deep neural networks for image segmentation”, in *ISVC*, 2016.
- [126] G. Nagendar, D. Singh, V. N. Balasubramanian, and C. Jawahar, “Neuroiou: Learning a surrogate loss for semantic segmentation.”, in *BMVC*, 2018.
- [127] M. Engilberge, L. Chevallier, P. Pérez, and M. Cord, “Sodeep: A sorting deep net to learn ranking loss surrogates”, in *CVPR*, 2019.
- [128] J. Grabocka, R. Scholz, and L. Schmidt-Thieme, “Learning surrogate losses”, *arXiv preprint arXiv:1905.10108*, 2019.
- [129] K. Li and J. Malik, “Learning to optimize neural nets”, *arXiv preprint arXiv:1703.00441*, 2017.
- [130] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of wasserstein gans”, in *NeurIPS*, 2017.
- [131] B. Shi, X. Wang, P. Lyu, C. Yao, and X. Bai, “Robust scene text recognition with automatic rectification”, in *CVPR*, 2016.
- [132] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, “Spatial transformer networks”, in *NeurIPS*, 2015.
- [133] W. Liu, C. Chen, K. K. Wong, Z. Su, and J. Han, “Star-net: A spatial attention residue network for scene text recognition”, in *BMVC*, 2016.
- [134] S. Hochreiter and J. Schmidhuber, “Long short-term memory”, *Neural computation*, 1997.

- [135] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, “Synthetic data and artificial neural networks for natural scene text recognition”, *CoRR*, 2014.
- [136] A. Gupta, A. Vedaldi, and A. Zisserman, “Synthetic data for text localisation in natural images”, in *CVPR*, 2016.
- [137] M. D. Zeiler, “ADADELTA: an adaptive learning rate method”, *CoRR*, 2012.
- [138] X. Zhang, J. J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification”, in *NeurIPS*, 2015.
- [139] B. Xu, N. Wang, T. Chen, and M. Li, “Empirical evaluation of rectified activations in convolutional network”, *CoRR*, 2015.
- [140] J. Ma, *RRPN in pytorch*, <https://github.com/mjq11302010044/RRPNpytorch>, 2019.
- [141] X. Liu, D. Liang, S. Yan, D. Chen, Y. Qiao, and J. Yan, “FOTS: fast oriented text spotting with a unified network”, in *CVPR*, 2018.
- [142] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks”, in *AISTATS*, 2011.
- [143] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors”, *Nature*, 1986.
- [144] Y. Patel, L. Gomez, M. Rusinol, and D. Karatzas, “Dynamic lexicon generation for natural scene images”, in *ECCV*, 2016.
- [145] Q. Ye and D. Doermann, “Text detection and recognition in imagery: A survey”, in *TPAMI*, 2014.
- [146] S. Long, X. He, and C. Yao, “Scene text detection and recognition: The deep learning era”, in *IJCV*, 2020.
- [147] K. Wang and S. Belongie, “Word spotting in the wild”, in *ECCV*, 2010.
- [148] C. Yao, X. Bai, B. Shi, and W. Liu, “Strokelets: A learned multi-scale representation for scene text recognition”, in *CVPR*, 2014.
- [149] M. Jaderberg, A. Vedaldi, and A. Zisserman, “Deep features for text spotting”, in *ECCV*, 2014.
- [150] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition”, in *Proceedings of the IEEE*, 1998.
- [151] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, “Reading text in the wild with convolutional neural networks”, in *IJCV*, 2016.
- [152] M. Busta, L. Neumann, and J. Matas, “Deep textspotter: An end-to-end trainable scene text localization and recognition framework”, in *ICCV*, 2017.
- [153] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, “Spatial transformer networks”, in *NeurIPS*, 2015.



- [154] L. Gómez, M. Rusinol, and D. Karatzas, “Lsde: Levenshtein space deep embedding for query-by-string word spotting”, in *ICDAR*, 2017.
- [155] B. Shi, X. Bai, and C. Yao, “An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition”, *PAMI*, 2016.
- [156] P. He, W. Huang, Y. Qiao, C. Loy, and X. Tang, “Reading scene text in deep convolutional sequences”, in *AAAI*, 2016.
- [157] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks”, in *NeurIPS*, 2014.
- [158] B. Shi, X. Wang, P. Lyu, C. Yao, and X. Bai, “Robust scene text recognition with automatic rectification”, in *CVPR*, 2016.
- [159] B. Shi, M. Yang, X. Wang, P. Lyu, C. Yao, and X. Bai, “Aster: An attentional scene text recognizer with flexible rectification”, in *PAMI*, 2018.
- [160] F. L. Bookstein, “Principal warps: Thin-plate splines and the decomposition of deformations”, in *TPAMI*, 1989.
- [161] W. Liu, C. Chen, and K.-Y. K. Wong, “Char-net: A character-aware neural network for distorted scene text recognition”, in *AAAI*, 2018.
- [162] H. Li, P. Wang, C. Shen, and G. Zhang, “Show, attend and read: A simple and strong baseline for irregular text recognition”, in *AAAI*, 2019.
- [163] M. Liao, P. Lyu, M. He, C. Yao, W. Wu, and X. Bai, “Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes”, in *TPAMI*, 2019.
- [164] A. Gupta, A. Vedaldi, and A. Zisserman, “Synthetic data for text localisation in natural images”, in *CVPR*, 2016.
- [165] Y. Patel<sup>1</sup>, M. Bušta<sup>1</sup>, and J. Matas<sup>1</sup>, “E2e-mlt-an unconstrained end-to-end method for multi-language scene text”, 2018.
- [166] S. Long and C. Yao, “Unrealtext: Synthesizing realistic scene text images from the unreal world”, in *CVPR*, 2020.
- [167] K. Janoušková, J. Matas, L. Gomez, and D. Karatzas, “Text recognition - real world data and where to find them”, in *ICPR*, 2021.
- [168] M. Liao, J. Zhang, Z. Wan, *et al.*, “Scene text recognition from two-dimensional perspective”, in *AAAI*, 2019.
- [169] F. Zhan and S. Lu, “Esir: End-to-end scene text recognition via iterative image rectification”, in *CVPR*, 2019.
- [170] M. Yang, Y. Guan, M. Liao, *et al.*, “Symmetry-constrained rectification network for scene text recognition”, in *ICCV*, 2019.

- [171] T. Wang, Y. Zhu, L. Jin, *et al.*, “Decoupled attention network for text recognition”, in *AAAI*, 2020.
- [172] D. Yu, X. Li, C. Zhang, *et al.*, “Towards accurate scene text recognition with semantic reasoning networks”, in *CVPR*, 2020.
- [173] Z. Qiao, Y. Zhou, D. Yang, Y. Zhou, and W. Wang, “Seed: Semantics enhanced encoder-decoder framework for scene text recognition”, in *CVPR*, 2020.
- [174] X. Yue, Z. Kuang, C. Lin, H. Sun, and W. Zhang, “Robustscanner: Dynamically enhancing positional clues for robust text recognition”, in *ECCV*, 2020.
- [175] R. Gomez, A. F. Biten, L. Gomez, J. Gibert, D. Karatzas, and M. Rusiñol, “Selective style transfer for text”, in *ICDAR*, 2019.
- [176] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification”, in *NeurIPS*, 2015.
- [177] B. Xu, N. Wang, T. Chen, and M. Li, “Empirical evaluation of rectified activations in convolutional network”, in *CoRR*, 2015.
- [178] Y. Zhang, L. Gueguen, I. Zharkov, P. Zhang, K. Seifert, and B. Kadlec, “Uber-text: A large-scale dataset for optical character recognition from street-level imagery”, in *CVPR workshop*, 2017.
- [179] M. D. Zeiler, “ADADELTA: an adaptive learning rate method”, in *CoRR*, 2012.
- [180] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, “Generalized intersection over union: A metric and a loss for bounding box regression”, in *CVPR*, 2019.
- [181] G. Nagendar, D. Singh, V. N. Balasubramanian, and C. Jawahar, “Neuroiou: Learning a surrogate loss for semantic segmentation.”, in *BMVC*, 2018.
- [182] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool, “Conditional probability models for deep image compression”, in *CVPR*, 2018.
- [183] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, “Variational image compression with a scale hyperprior”, in *ICLR*, 2018.
- [184] Y. Patel, S. Appalaraju, and R. Manmatha, “Saliency driven perceptual image compression”, in *WACV*, 2021.
- [185] D. Bahdanau, P. Brakel, K. Xu, *et al.*, “An actor-critic algorithm for sequence prediction”, in *ICLR*, 2017.
- [186] M. Engilberge, L. Chevallier, P. Pérez, and M. Cord, “Sodeep: A sorting deep net to learn ranking loss surrogates”, in *CVPR*, 2019.

- [187] K. He, Y. Lu, and S. Sclaroff, “Local descriptors optimized for average precision”, in *CVPR*, 2018.
- [188] J. Revaud, J. Almazán, R. S. Rezende, and C. R. d. Souza, “Learning with average precision: Training image retrieval with a listwise loss”, in *ICCV*, 2019.
- [189] M. Rolínek, V. Musil, A. Paulus, M. Vlastelica, C. Michaelis, and G. Martius, “Optimizing rank-based metrics with blackbox differentiation”, in *CVPR*, 2020.
- [190] F. Cakir, K. He, X. Xia, B. Kulis, and S. Sclaroff, “Deep metric learning to rank”, in *CVPR*, 2019.
- [191] K. Musgrave, S. Belongie, and S.-N. Lim, “A metric learning reality check”, in *ECCV*, 2020.
- [192] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition”, in *CVPR*, 2016.
- [193] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “Mixup: Beyond empirical risk minimization”, *arXiv preprint arXiv:1710.09412*, 2017.
- [194] A. Zhai and H.-Y. Wu, “Classification is a strong baseline for deep metric learning”, *arXiv preprint arXiv:1811.12649*, 2018.
- [195] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, “Sphereface: Deep hypersphere embedding for face recognition”, in *CVPR*, 2017.
- [196] H. Wang, Y. Wang, Z. Zhou, *et al.*, “Cosface: Large margin cosine loss for deep face recognition”, in *CVPR*, 2018.
- [197] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “Arcface: Additive angular margin loss for deep face recognition”, in *CVPR*, 2019.
- [198] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh, “No fuss distance metric learning using proxies”, in *ICCV*, 2017.
- [199] E. W. Teh, T. DeVries, and G. W. Taylor, “Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis”, in *ECCV*, 2020.
- [200] Q. Qian, L. Shang, B. Sun, J. Hu, H. Li, and R. Jin, “Softtriple loss: Deep metric learning without triplet sampling”, in *ICCV*, 2019.
- [201] I. Elezi, S. Vascon, A. Torcinovich, M. Pelillo, and L. Leal-Taixé, “The group loss for deep metric learning”, in *ECCV*, 2020.
- [202] J. Seidenschwarz, I. Elezi, and L. Leal-Taixé, “Learning intra-batch connections for deep metric learning”, in *ICML*, 2021.
- [203] M. Boudiaf, J. Rony, I. M. Ziko, *et al.*, “Metric learning: Cross-entropy vs. pairwise losses”, in *ECCV*, 2020.

- [204] K. Roth, T. Milbich, S. Sinha, P. Gupta, B. Ommer, and J. P. Cohen, “Revisiting training strategies and generalization performance in deep metric learning”, in *ICML*, 2020.
- [205] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese, “Deep metric learning via lifted structured feature embedding”, in *arXiv*, 2015.
- [206] K. Sohn, “Improved deep metric learning with multi-class n-pair loss objective”, in *NeurIPS*, 2016.
- [207] J. Lu, C. Xu, W. Zhang, L.-Y. Duan, and T. Mei, “Sampling wisely: Deep image embedding by top-k precision optimization”, in *ICCV*, 2019.
- [208] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin, “Deep metric learning with angular loss”, in *ICCV*, 2017.
- [209] E. Ustinova and V. Lempitsky, “Learning deep embeddings with histogram loss”, in *NeurIPS*, 2016.
- [210] M. Kemertas, L. Pishdad, K. G. Derpanis, and A. Fazly, “Rankmi: A mutual information maximizing ranking loss”, in *CVPR*, 2020.
- [211] V. Verma, A. Lamb, C. Beckham, *et al.*, “Manifold mixup: Better representations by interpolating hidden states”, in *ICML*, 2019.
- [212] Y. Duan, W. Zheng, X. Lin, J. Lu, and J. Zhou, “Deep adversarial metric learning”, in *CVPR*, 2018.
- [213] Y. Kalantidis, M. B. Sariyildiz, N. Pion, P. Weinzaepfel, and D. Larlus, “Hard negative mixing for contrastive learning”, *NeurIPS*, 2020.
- [214] W. Zheng, Z. Chen, J. Lu, and J. Zhou, “Hardness-aware deep metric learning”, in *CVPR*, 2019.
- [215] G. Gu and B. Ko, “Symmetrical synthesis for deep metric learning”, in *AAAI*, 2020.
- [216] G. Gu, B. Ko, and H.-G. Kim, “Proxy synthesis: Learning with synthetic classes for deep metric learning”, *AAAI*, 2021.
- [217] S. Venkataramanan, B. Psomas, Y. Avrithis, E. Kijak, L. Amsaleg, and K. Karantzas, “It takes two to tango: Mixup for deep metric learning”, in *ICLR*, 2022.
- [218] N. Kyurkchiev and S. Markov, “Sigmoid functions: Some approximation and modelling aspects”, *LAP LAMBERT Academic Publishing, Saarbrücken*, 2015.
- [219] A. I. Iliev, N. Kyurkchiev, and S. Markov, “On the approximation of the cut and step functions by logistic and gompertz functions”, *Biomath*, 2015.
- [220] A. Iliev, N. Kyurkchiev, and S. Markov, “On the approximation of the step function by some sigmoid functions”, *Mathematics and Computers in Simulation*, 2017.

- [221] R. Salakhutdinov and G. Hinton, “Semantic hashing”, *International Journal of Approximate Reasoning*, 2009.
- [222] S. Gu, S. Levine, I. Sutskever, and A. Mnih, “Muprop: Unbiased back-propagation for stochastic neural networks”, in *ICLR*, 2016.
- [223] C. J. Maddison, A. Mnih, and Y. W. Teh, “The concrete distribution: A continuous relaxation of discrete random variables”, in *ICLR*, 2017.
- [224] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han, “Large-scale image retrieval with attentive deep local features”, in *ICCV*, 2017.
- [225] G. Toliás, T. Jeníček, and O. Chum, “Learning and aggregating deep local descriptors for instance-level recognition”, in *ECCV*, 2020.
- [226] W. Dong, R. Socher, L. Li-Jia, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database”, in *CVPR*, 2009.
- [227] F. Radenović, G. Toliás, and O. Chum, “Fine-tuning cnn image retrieval with no human annotation”, *PAMI*, 2019.
- [228] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization”, *arXiv preprint arXiv:1607.06450*, 2016.
- [229] R. Wightman, *Pytorch image models*, <https://github.com/rwightman/pytorch-image-models>, 2019. DOI: 10.5281/zenodo.4414861.
- [230] D. Kingma and J. Ba, “Adam: A method for stochastic optimization”, in *ICLR*, 2015.
- [231] C. Szegedy, W. Liu, Y. Jia, *et al.*, “Going deeper with convolutions”, in *CVPR*, 2015.
- [232] K. Roth, B. Brattoli, and B. Ommer, “Mic: Mining interclass characteristics for improved metric learning”, in *ICCV*, 2019.
- [233] X. Wang, H. Zhang, W. Huang, and M. R. Scott, “Cross-batch memory for embedding learning”, in *CVPR*, 2020.
- [234] X. Wang, X. Han, W. Huang, D. Dong, and M. R. Scott, “Multi-similarity loss with general pair weighting for deep metric learning”, in *CVPR*, 2019.
- [235] P. Jacob, D. Picard, A. Histace, and E. Klein, “Metric learning with horde: High-order regularizer for deep embeddings”, in *ICCV*, 2019.
- [236] Y. Zhang, L. Luo, W. Xian, and H. Huang, “Learning better visual data similarities via new grouplet non-euclidean embedding”, in *ICCV*, 2021.
- [237] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, “Neural codes for image retrieval”, in *ECCV*, 2014.
- [238] A. Gordo, J. Almazan, J. Revaud, and D. Larlus, “End-to-end learning of deep visual representations for image retrieval”, *IJCV*, 2017.

- [239] H. Goëau, P. Bonnet, and A. Joly, “Overview of expertlifecyclef 2018: How far automated identification systems are from the best experts?”, in *CLEF*, 2018.
- [240] H. Goeau, P. Bonnet, and A. Joly, “Plant identification based on noisy web data: The amazing performance of deep learning (lifeclef 2017)”, in *CLEF*, 2017.
- [241] S. Chopra, R. Hadsell, and Y. LeCun, “Learning a similarity metric discriminatively, with application to face verification”, in *CVPR*, 2005.
- [242] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations”, in *ICML*, 2020.
- [243] Y. Kalantidis, M. B. Sariyildiz, N. Pion, P. Weinzaepfel, and D. Larlus, “Hard negative mixing for contrastive learning”, *arXiv preprint arXiv:2010.01028*, 2020.
- [244] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. Hinton, “Big self-supervised models are strong semi-supervised learners”, *arXiv preprint arXiv:2006.10029*, 2020.
- [245] M. Caron, H. Touvron, I. Misra, *et al.*, “Emerging properties in self-supervised vision transformers”, in *ICCV*, 2021.
- [246] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, “Unsupervised learning of visual features by contrasting cluster assignments”, *arXiv preprint arXiv:2006.09882*, 2020.
- [247] M. Davari, N. Asadi, S. Mudur, R. Aljundi, and E. Belilovsky, “Probing representation forgetting in supervised and unsupervised continual learning”, in *CVPR*, 2022.
- [248] J. Winkens, R. Bunel, A. G. Roy, *et al.*, “Contrastive training for improved out-of-distribution detection”, *arXiv preprint arXiv:2007.05566*, 2020.
- [249] D. Chen, D. Wang, T. Darrell, and S. Ebrahimi, “Contrastive test-time adaptation”, in *CVPR*, 2022.
- [250] M. Boudiaf, J. Rony, I. M. Ziko, *et al.*, “A unifying mutual information view of metric learning: Cross-entropy vs. pairwise losses”, in *ECCV*, 2020.
- [251] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, “On combining classifiers”, *TPAMI*, 1998.
- [252] L. Breiman, “Bagging predictors”, *Machine learning*, 1996.
- [253] C. Ju, A. Bibaut, and M. van der Laan, “The relative performance of ensemble methods with deep convolutional neural networks for image classification”, *Journal of Applied Statistics*, 2018.

- [254] M. Saerens, P. Latinne, and C. Decaestecker, “Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure”, *Neural computation*, 2002.
- [255] M. Sulc and J. Matas, “Improving cnn classifiers by estimating test-time priors”, in *ICCV Workshops*, 2019.
- [256] A. Alexandari, A. Kundaje, and A. Shrikumar, “Maximum likelihood with bias-corrected calibration is hard-to-beat at label shift adaptation”, in *ICML*, 2020.
- [257] T. Sipka, M. Sulc, and J. Matas, “The hitchhiker’s guide to prior-shift adaptation”, *arXiv preprint arXiv:2106.11695*, 2021.
- [258] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network”, *arXiv preprint arXiv:1503.02531*, 2015.
- [259] D. Hendrycks and K. Gimpel, “A baseline for detecting misclassified and out-of-distribution examples in neural networks”, *arXiv preprint arXiv:1610.02136*, 2016.
- [260] F. Graf, C. Hofer, M. Niethammer, and R. Kwitt, “Dissecting supervised contrastive learning”, in *ICML*, 2021.
- [261] P. Wohlhart, M. Kostinger, M. Donoser, P. M. Roth, and H. Bischof, “Optimizing 1-nearest prototype classifiers”, in *CVPR*, 2013.
- [262] X. Chen and K. He, “Exploring simple siamese representation learning”, in *CVPR*, 2021.
- [263] J.-B. Grill, F. Strub, F. Altché, *et al.*, “Bootstrap your own latent: A new approach to self-supervised learning”, *arXiv preprint arXiv:2006.07733*, 2020.
- [264] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning”, in *CVPR*, 2020.
- [265] J. Li, P. Zhou, C. Xiong, and S. C. Hoi, “Prototypical contrastive learning of unsupervised representations”, *arXiv preprint arXiv:2005.04966*, 2020.
- [266] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh, “No fuss distance metric learning using proxies”, in *ICCV*, 2017.
- [267] S. Kim, D. Kim, M. Cho, and S. Kwak, “Proxy anchor loss for deep metric learning”, in *CVPR*, 2020.
- [268] Y. Sun, C. Cheng, Y. Zhang, *et al.*, “Circle loss: A unified perspective of pair similarity optimization”, in *CVPR*, 2020.
- [269] A. Krizhevsky, G. Hinton, *et al.*, “Learning multiple layers of features from tiny images”, 2009.
- [270] Stanford, *Tiny ImageNet Challenge, CS231N Course*. [Online]. Available: <https://tiny-imagenet.herokuapp.com/>.

- [271] G. Griffin, A. Holub, and P. Perona, “Caltech-256 object category dataset”, 2007.
- [272] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, “On calibration of modern neural networks”, in *ICML*, PMLR, 2017.