Università
Ca'Foscari
Venezia

Ph.D. Programme in
COMPUTER SCIENCE

33rd Cycle

Research Thesis

Using Contextual Information In Weakly Supervised Learning

Toward the integration of contextual and deep learning approaches, to address weakly supervised tasks

SSD: INF/01

Ph.D. Coordinator
Ch. Prof. Agostino Cortesi

Supervisor
Ch. Prof. Marcello Pelillo

Graduand
Alessandro Torcinovich
Matriculation Number 840284

Academic Year
2020-2021

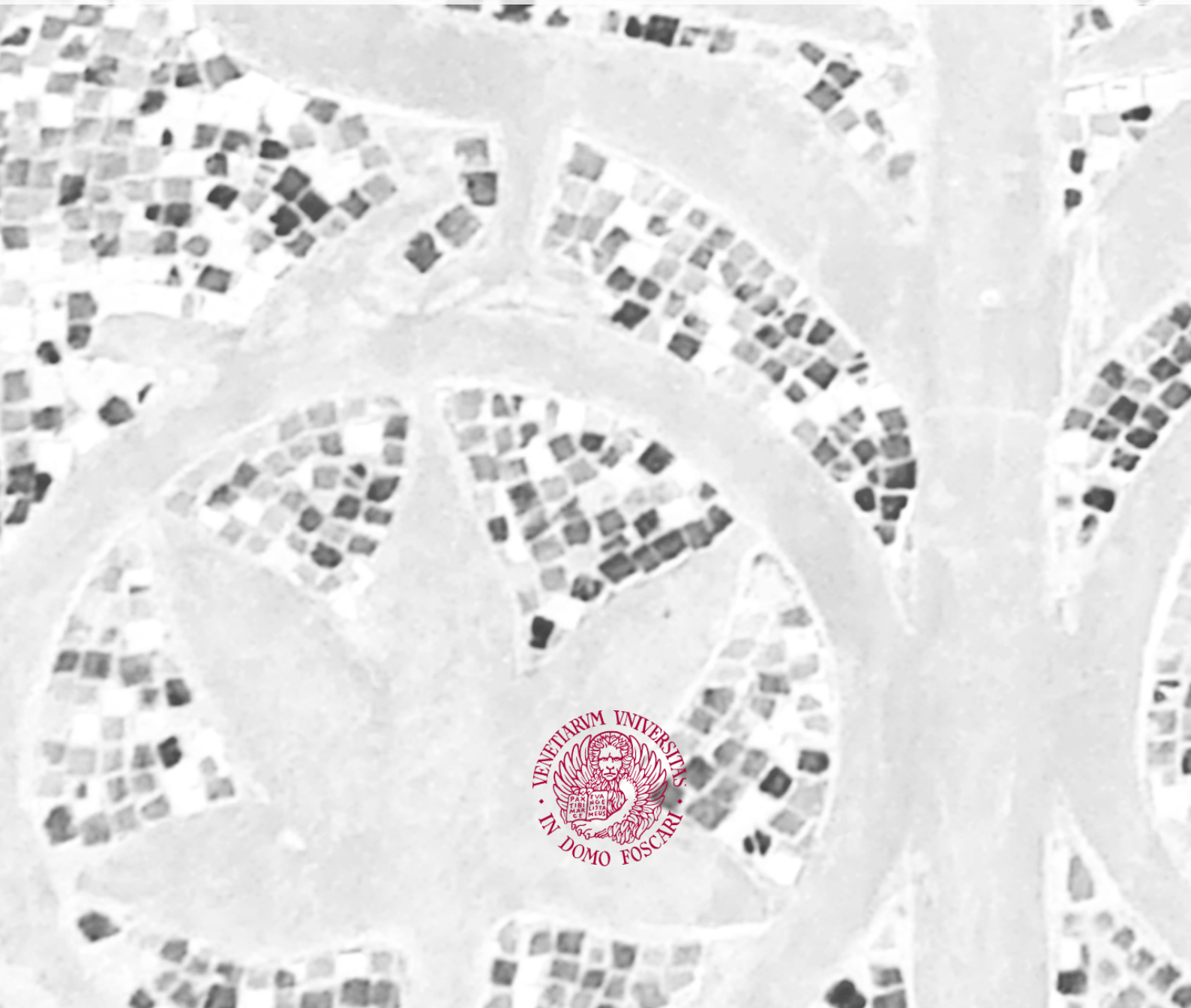
UNIVE
Department of Environmental Sciences,
Informatics and Statistics
Ca' Foscari University of Venice

Using Contextual Information In Weakly Supervised Learning

Toward the integration of contextual and deep learning approaches, to address weakly supervised tasks.

Alessandro Torcinovich
Ph.D. Thesis, Spring 2021

Supervisor
Prof. Marcello Pelillo



This Ph.D. thesis is submitted to conclude the Ph.D programme *Computer Science*, in the field of *Artificial Intelligence*, at the Department of Environmental Sciences, Informatics and Statistics, Ca' Foscari University of Venice.

The front page depicts a detail of the mosaic built over the main entrance of the Saint Mark's Basilica at Venice. Courtesy of Luisella Golfetto.

Abstract (IT)

Come l'attento lettore avrà dedotto dal titolo, questa tesi pone alcune basi empiriche, assieme ad altrettante considerazioni teoriche, verso la definizione di una metodologia finalizzata a migliorare task di weakly supervised learning. La metodologia genera supervisione addizionale sfruttando l'informazione contestuale proveniente dal confronto delle osservazioni in un dataset sotto molteplici ipotesi di etichettatura.

Il materiale di ricerca presentato, ruota principalmente attorno a due algoritmi. Nella prima parte, l'attenzione è rivolta a Graph Transduction Games (GTG), un algoritmo di label propagation basato su nozioni di Teoria dei Giochi. In particolare, questo documento descrive le interazioni sperimentate con GTG e dei deep feature extractor, per affrontare problemi di semi-supervised, domain adaptation e deep metric learning. La seconda parte è incentrata su Relaxation Labeling (ReLab), una famiglia di processi utilizzata per label disambiguation, fortemente connessa a GTG, sebbene sia motivata da un differente contesto teorico. Questo documento alcuni concetti preliminari di teoria e degli esperimenti pensati per investigare future applicazioni di ReLab nel contesto di semi-supervised semantic segmentation.

Il lavoro presentato di seguito può essere pensato come un punto iniziale per costituire una teoria di contextual weakly supervised learning.

Abstract (EN)

As the dedicated reader can deduce from the title, this dissertation lays some empirical bases, along with some theoretical considerations, toward the definition of a methodology aimed to improve weakly supervised learning tasks. The methodology generates additional supervision by exploiting the contextual information coming from the comparison of observations in a dataset, under multiple labeling hypotheses.

The presented research material mainly rotates around two algorithms. In the first part, the attention is devoted to Graph Transduction Games (GTG), a label propagation algorithm grounded on game theory notions. In particular, this document describes the interactions experimented with GTG and deep feature extractors to address semi-supervised, domain adaptation and deep metric learning tasks. The second part is focused on Relaxation Labeling (ReLab), a family of processes used for label disambiguation, tightly related with GTG, despite being motivated by a different theoretical framework. This document presents some preliminary theory and experiments aimed to investigate future applications of ReLab in the context of semi-supervised semantic segmentation.

The work presented in the following can be thought a starting point for constituting a theory of contextual weakly supervised learning.

Preface

Suppose to have trained an object detector, that, upon given a specific image, finds two objects placed one above the other: a sofa and an horse. Detecting an horse and a sofa in the same image might be a rare event, detecting a horse *on top of a* sofa might be even rarer. The motivation of this work comes from the observation that this knowledge – often referred, in the following, with the term **contextual information** – is as important as the knowledge coming from the observations taken singularly. However, while the main focus on deep learning research has been devoted to the search of a good feature space mapping real world entities to meaningful feature embeddings, less consideration has been given to the study of the relations among the observations themselves. The purpose of this document consists of analyzing the use of the contextual information in a variety of tasks, with the final aim of addressing weakly supervised learning problems. Not all the works presented here deal with label scarcity though; some of them operate with full supervision, but they are equally discussed in this thesis because they constitute a starting point toward a corresponding study in weakly supervised settings. In fact, as we will see in the following chapters, one of the main advantages of the contextual information is its generic nature and adaptability of application in different labeling regimes.

The main characters of the plot are two algorithms: Graph Transduction Games (GTG) and Relaxation Labeling (ReLab). This document starts with a thorough analysis of both, in order to provide the reader with some basic

understanding of their working and the motivations under their design choices. A small introduction on deep neural models is provided too, for making this work more self-contained, and for helping the casual reader to understand the most common neural algorithms found in the literature. Following, are presented the main publications related with the topic of this thesis, along with some unpublished work, worth to be presented as well.

As a last word, this thesis has not the ambition to be a concluded work. The author conceives it as a milestone on his path of research. A path that has not yet come to an end.

Contributions

The main contributions of this thesis are the following:

- We propose a new approach to automatically augment the supervision provided by a partially labeled dataset. The approach is based on a graph transductive algorithm, motivated by game theoretic results.
- We adapt the previous methodology to address the more challenging task of domain adaptation.
- We devise a new post-processing step to include, in a end-to-end fashion, the contextual information for addressing metric learning tasks.
- We propose another end-to-end procedure that exploits both the observation and class similarities, showing some preliminary theory and experiments to address semantic segmentation tasks.

Research covered by this thesis

In the following, the author lists all the works constituting the core of this thesis that led to publications over the course of his Ph.D. program, and the currently open lines of research that are still under investigation:

- (Elezi et al. 2018) is a first attempt to address the supervised data hunger problem of Deep Neural Networks by automatically disambiguating unlabeled observations with a limited amount of labeled information available. This task is performed by using an label disambiguation algorithm called Graph Transduction Games (GTG)[1]. The author implemented a consistent part of the code, wrote a section of the paper and helped in writing the other ones. Finally, he entirely rewrote the framework for publication (https://github.com/aretor/labaug), and implemented part of the code for the extension of the work.

- (Vascon et al. 2019) tackles a problem similar to the previous one, where the unlabeled observations are sampled from a different distribution. GTG is adapted and performed in this setting, showing that this algorithm can advance the state-of-the-art also in more general weakly supervised settings. The author participated in the decision making of the pipeline and the experimental setting, wrote a consistent part of the code, performed some experiment, wrote the theory section of the paper, while helping in writing other ones.

- (Elezi et al. 2020) takes inspiration by the previous two works and devises an end-to-end procedure to generate metric embeddings which are then disambiguated based on the contextual information provided by each other. Though the work is conducted in the fully supervised regime, the article has constituted a fundamental point for the research in weakly supervised regimes, which is currently under study by the author. The author participated in the decision making of the pipeline, wrote part of the paper and developed part of the code.

- **Relaxation Labeling Processes for Semantic Image Segmentation** (unpublished): casts the Relaxation Labeling (ReLab) processes

[1] Sometimes referred as Graph Transductive Game

into a different, more structured perspective, in the attempt to devise an algorithm to address Semantic Segmentation tasks, in an end-to-end manner. As in the previous case, the work has laid some bases to address weakly supervised variants on this field. The author developed the code, performed most of the experiments and has written the paper.

Other works by the author

For the sake of completeness, hereafter, the author lists all the works that do not directly pertain or are loosely related to the topic of this thesis, but that have nonetheless been brought on over the course of his Ph.D. program, and therefore reserve a mention:

- (Torcinovich et al. 2017) describes a work performed in collaboration with Prosa s.r.l. where the author developed a computer vision system to monitor commercial ice-cream freezers.
- (Fiorucci et al. 2017) discusses the Szemerédi regularity lemma as a tool to preserve the metric information in large graphs. The author entirely developed the code and helped in the conception of the theoretical decisions.
- (Cinà, Torcinovich and Pelillo 2020) (Pattern Recognition Journal, submitted and reviewed) proposes a novel adversarial learning attack strategy in black-box gradient-free unsupervised settings. The author developed the theoretical formulation and proved theoretical results (e.g. global convergence of the algorithm).
- **An Analysis of Stationary Points arising from the Maximum Clique Problem** (unpublished) analyses, as the name says, the stationary points arising from non-strict maximal cliques, in unweighted undirected graphs. This work is related to a certain degree to the topic of our thesis

since it investigate the same maximization problem of (Elezi et al. 2018; Vascon et al. 2019) and (Elezi et al. 2020).

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

This chapter presents the main theoretical concepts at the base of the work presented in this thesis. Without any ambition of exhaustiveness, the following explanations will be concise, with the only aim of self-contentedness. The reader is encouraged to check the references for more detailed and specific dissertations on these subjects.

1.1 Deep Learning Basics

Deep Learning is nowadays a fundamental brick to build ML/CV architectures. Its ubiquitousness in virtually every possible automated learning task speaks for itself, and the works presented in this thesis make no exception. The purpose of this section is the revision of the basic concepts of Fully Connected and Convolutional Neural Networks (FCNs and CNNs), two of the most critical learning models devised in the context of this field.

Fully Connected Networks

The first attempt to model the neuronal connections of the brain, is usually attributed to the cybernetician Warren McCulloch and the logician Walter Pitts who, in 1943, published *"A Logical Calculus of the Ideas Immanent in Nervous Activity"* (McCulloch and Pitts 1990). In this seminal paper, it was presented for the first time the **Linear Threshold Unit** (LTU), a simple

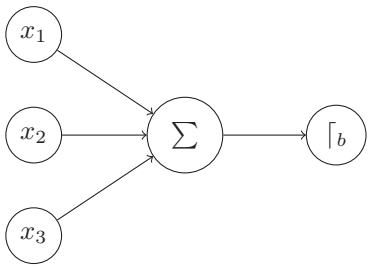


Figure 1.1: The Linear Threshold Unit, by McCulloch and Pitts.

schematization of the behavior of a neuron. Such model, shown in Figure 1.1, takes different inputs and consists in two elementary operations. First, an **aggregation function** – usually a sum – is used to combine the inputs into one scalar only, then a **step function** is applied to binarize the output. The only parameter of the model is represented by the **bias** b, that determines the thresholding point.

The LTU nicely summarizes the fundamental points of connectionist thinking. The aggregation and thresholding operators are two characteristic traits present in all neural models developed so far. As said above, the LTU was inspired by the activity of the neurons in the brain. Thus the model is commonly known as **McCulloch and Pitts Neuron**. The design of such a neuron, however, was heavily simplified and loosely represents its biological counterpart.

Later, Frank Rosenblatt, improved this representation by introducing a weighting of the inputs and an update rule to iteratively learn the weights,

Algorithm 1 Perceptron's learning algorithm

Require: A dataset of observations $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$, the total number of iterations/epochs $\mathcal{E}$, a Perceptron $f(\mathbf{x})$ with learnable weights $\mathbf{w} = \{w_1, \ldots, w_d, b\}$, the learning rate η.

$e = 0$
while $e < \mathcal{E}$ **do**
    for $(\mathbf{x}_i, y_i) \in \mathcal{D}$ **do**
        $\hat{y}_i = f(\mathbf{x}_i)$
        for $w_j \in \mathbf{w}$ **do**
            $w_j = w_j + \eta(y_i - \hat{y}_i)x_{ij}$
    $e = e + 1$

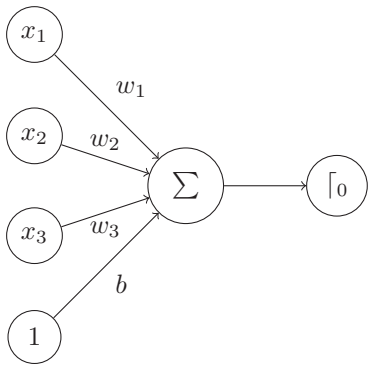


Figure 1.2: The Perceptron, by Rosenblatt.

bias included. The model, called **Perceptron** (Rosenblatt 1961), generates an output (a binary label) and compare it with a ground truth. If the output is wrong an error value is generated and propagated back to the weights which are then updated accordingly, otherwise the model is left unmodified. Algorithm 1, presents the pseudo-code of the Perceptron learning algorithm, while a schema of the model is shown in Figure 1.2. Note that the bias becomes a learnable weight too, with threshold function simply checking if its input is non-negative. Perceptrons are known to solve only linear separable problems and fail when addressing non-linear ones such the XOR problem (Minsky and Papert 1987). This limitation can be addressed by grouping set of Perceptrons in layers and connecting them together, to form a **Multi-Layer Perceptron** (MLP), also known as **Fully Connected Network** (FCN). Each layer in an MLP can be easily implemented through a matrix-vector multiplication for the aggregation while the step function is applied element-wise to the output. According to (Schmidhuber 2015), probably the first MLP network was that proposed by (Ivakhnenko and Lapa 1966).

Backpropagation Linnainmaa 1970; Werbos 1982, the most popular training procedure for MLPs, was popularized in the mid '80s by (Rumelhart, Hinton and Williams 1986), and settled another milestone in the history of connectionst AI. With backpropagation, it is possible to *virtually* train arbitrarily

complex, almost everywhere differentiable functions, such as MLPs. In this case, the step function is usually substituted with a **non-linear activation function**, such as the sigmoid (Han and Moraga 1995), the hyperbolic tangent (Weisstein n.d.), or, more recently, the Rectified Linear Unit (ReLU) (Hahnloser et al. 2000).

With the resurgence of MLPs as powerful tools for addressing regression and classification tasks, much work was devoted to defining the potentialities of this model theoretically. This line of research culminated in the seminal results enunciated in the **universal approximation theorems** (Lu et al. 2017; Pinkus 1999). In essence, the theorems state that arbitrarily wide/deep MLPs act as approximators of nearly any Lebesgue-integrable function. The theorems, however, only provide existential statements without describing any procedure for obtaining the MLP corresponding to a specific function of choice.

Convolutional Neural Networks

MLPs make relatively few assumptions on the nature of the inputs and sometimes fail to exploit their structure entirely. In the case of image data, for example, we assume that the input is structured as a lattice and that a pixel and its neighborhood are correlated together. Such assumptions are called **inductive biases** and are fundamental in the design of an ML algorithm. MLPs combine the information of all the pixels together, regardless of their position, thus ignoring the spatiality of the input.

In addition, MLPs with many layers, trained by backpropagation, incur in the **vanishing/exploding gradient** problem (Bengio, Simard and Frasconi 1994; Hochreiter 1991). In particular, when propagating the error, its magnitude tends to vanish/explode, producing insignificant or excessive weight updates in the first layers of the model. This severe limitation prevents to apply MLPs for the correct classification of more complex and structured data such as images.

The first works toward spatially correlated data was performed by (Hubel and Wiesel 1959), identifying that the neurons in the visual cortex of monkeys and cats were individually reacting to local regions of a visual field. Such regions are called **receptive fields** and represent a capital concept in computer vision theory nowadays. Later, (Fukushima and Miyake 1982), inspired by the aforementioned work, devised the **Neocognitron**, a neural model specifically tailored for image data, and, perhaps, the first "deep" neural architecture Schmidhuber 2015. The Neocognitron structures alternates **Simple** and **Complex layers (S/C-layers)**. Each of them is composed respectively inhibitory cells and excitatory S/C-cells (neurons), the latter organized in matrices called S/C-planes. In S-layers, each S-cell fires according to a threshold function, whose input is computed as a fraction of weighted sums between excitatory and inhibitory inputs. The weights of the sums represent the learnable parameters and provide the network with a sort of "neuro-plasticity" to adapt to different patterns, like the MLP. Note that each S-cell is connected to only a small subset of incoming inputs, corresponding to neighboring cells of the previous layer, in the attempt to model the local connectivity structure described by (Hubel and Wiesel 1959). C-layers behave similarly to the S-layers, except that the sum is no more weighted and it does not span one plane instead of all of them, (thus in an S/C-layer pair, the number of planes is the same).

In conclusion, S-layers can be seen as a specialized aggregation function combining small, overlapping regions of the input (the aforementioned receptive fields) into several outputs while keeping the spatiality of the data; the C-layer, on the contrary, operates a signal downsampling that preserves the most important activations of an S-layer, while discarding the least important ones, to guarantee small-translation invariance in the classification of a concept. Later, the NeoCognitron "evolved" in a new kind of neural architecture: the **Convolutional Neural Network** (CNN) (LeCun et al. 1989a). In a CNN, the standard matrix multiplication of the fully connected layer is replaced with

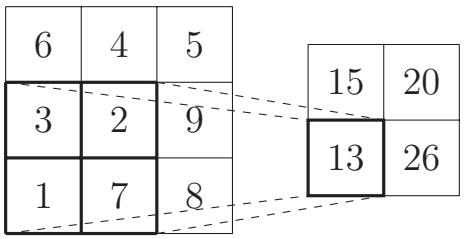


Figure 1.3: An example of the convolution operator, assuming the kernel weights $w_{ij} = 1\ \forall i, j$, meaning that the sum of the values is computed.

a convolution operator (Goodfellow, Bengio and Courville 2016):

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{1.1}$$

While the operator refers to a 2d input, it can be easily extended to 3d inputs, like RGB images:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n \sum_c I(i-m, j-n, c)K(m,n,c) \tag{1.2}$$

Figure 1.3 shows an example of convolution on a one-channel image. In case of multiple channels, the window has three dimensions and convolves the channel inputs all together. A **convolutional layer** implements many of these multi-channel convolutions stacked together to generate multi-channel outputs.

The convolution operator is **commutative**, that is $(I*K)(i,j) = (K*I)(i,j)$, $\forall i, j$. Moreover, the convolution operator can be performed as a matrix-vector multiplication with additional rules on the weight matrix. Indeed, (LeCun et al. 1989a) describes a CNN as a network whose parameter space is constrained through the use of **infinitely strong priors** (Goodfellow, Bengio and Courville 2016) that constraint neurons of the input layer to share parameters and to be connected with only a small portion of neurons of the output layer. According to (Goodfellow, Bengio and Courville 2016) the key properties that motivate the success of this operator in dealing with image data are:

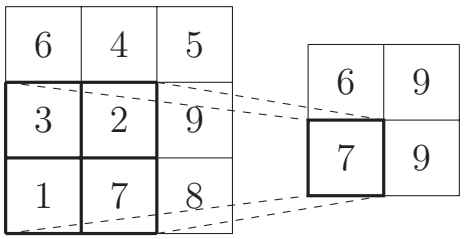


Figure 1.4: An example of the max-pooling operator.

- **Sparse interactions:** kernels are usually far smaller than the image where they are applied. This allows detecting small, meaningful features such as edges or corners in a computationally efficient way.

- **Parameter sharing:** the kernels are small blocks of weights that are "slided" over the image covering it all. For this reason, the number of weights is far smaller than those used in MLPs. A CNN can have multiple convolutional layers without increasing too much in space.

- **equivariant representations:** the convolution is equivariant with respect to the translation of its input, that means that if $I' = g(I)$ consists of a translation transformation, and f is the convolution operator, applying $f(g(I))$ or $g(f(I))$ results in the same identical output. This means that no matter where an object is located in an image, it will generate the same output in the corresponding location.

In some tasks, different from the image classification, it may be needed to enforce other constraints. For this reason, the literature contains many variants of the standard convolution presented here. In addition to the convolution, another layer is added, the **pooling** layer, that operates a "summarization" of the convolution outputs. In particular, the pooling operator considers groups of spatially neighboring inputs and, for each of them, it generates a summary statistic as output. In general, the computed statistics is the maximum among the inputs, but (weighted) average and ℓ_2-norm have been used as well in the

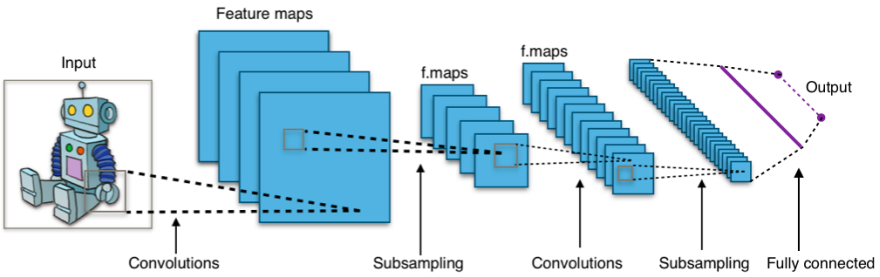


Figure 1.5: The schematization of a CNN. Taken from Aphex34 2015. "Fully connected" refers to the MLP at the end of the network.

past. Given a 3d input, the pooling operator is applied channel-wise, that is, given a $H \times W \times C$ input, a small sliding window (usually of size 2×2) is passed over each of the C channels and the statistic is computed. Note that, in general, the pooling operator does not require any learning parameter.

The purpose of pooling consists in making the CNN **invariant** to small translations of the input, under the rationale that the exact position of an object is not important for its classification. Like the convolution, the pooling can be seen as the addition of an infinitely strong prior that enforces this invariance. Figure 1.4 shows an example of pooling. Through the use of multiple layers of convolution + non-linear activation + pooling, it is possible to generate meaningful features that can then be passed to a shallow MLP, being now able to classify more complex inputs. Figure 1.5 shows a schematization of a CNN model.

Graph Neural Networks

Graph Neural Networks (GNNs) have recently arisen as a major topic of interest in the deep learning community. Among the other DNNs, these models deserve a description in this work, for they ability to leverage the contextual information coded in graphs.

These models substantially represent a way to relax the inductive biases

imposed in the CNN, in order to work with generic graph structures. Almost all GNNs learn graph vertices representations, called states, that can be further synthesized into a graph representation, when required by the task. These representations are then used to solve typical graph-related tasks, such as graph matching, retrieval, classification, and so on. (Bacciu et al. 2020) first divide GNN into two approaches: **recursive** and **feedforward**. In the first case, the model can be seen as a dynamical system process which progressively refine the states by diffusing the information of each vertex to its neighbors. In the latter case, instead, the model is divided in layer which perform a progressive "composition" of the local context into the global one, in a CNN fashion. For this motivation, such layers are also called **graph convolutional layers**. In both cases, dealing with graphs requires to deal with variable size and shape inputs which in turn requires to leverage the local structure information to propagate it on a global level. Further, when working with variable topology graphs, we must ensure that the output does not change if the order of the vertex representations does. **permutation invariant functions** like sum, mean and product are commonly adopted to address this requirement. A proved (Wagstaff et al. 2019) general expression for a permutation invariant function $\Psi : \mathcal{X}^M \to \mathcal{Y}$, is:

$$\Psi(Z) = \phi(\sum_{z \in Z} \psi(z)) \tag{1.3}$$

With $\phi : M \to \mathcal{Y}$ and $\phi : \mathcal{X} \to M$ are continuous functions such as neural networks.

(Bacciu et al. 2020) attempt in giving a formulation a general local context composition operator, called **neighbor aggregation**, as follows:

$$\mathbf{h}_v^{(\ell+1)} = \phi^{(\ell+1)}\left(\mathbf{h}^{(\ell)}, \Psi(\{\psi^{(\ell+1)}(\mathbf{h}_u^{(\ell)}) | u \in \mathcal{N}_v\})\right) \tag{1.4}$$

with $\mathbf{h}_u^{(\ell)}$ denoting the state of a vertex u at step/layer ℓ, ϕ/ψ implementing

arbitrary transformations of the input data and $\mathcal{N}_v$ being the neighborhood set of v.

When dealing with discrete graph edge information, the formulation becomes:

$$\mathbf{h}^{(\ell+1)} = \phi^{(\ell+1)}\left(\mathbf{h}_v^{(\ell)}, \sum_{c_k \in \mathcal{A}} \Psi(\{\psi^{(\ell+1)}(\mathbf{h}_u^{(\ell)}) | u \in \mathcal{N}_v^{c_k}\}) * w_{c_k}\right) \tag{1.5}$$

with w_{c_k} being a learnable scalar parameter that weights the contribution of arcs with label $c_k \in \mathcal{A}$, $N_v^{(c_k)}$ being the neighborhood of node v connected by an edge with label c_k and $*$ multiplies every component of its first argument by w_{c_k}. For continuous graph edge information instead we have:

$$\mathbf{h}^{(\ell+1)} = \phi^{(\ell+1)}\left(\mathbf{h}_v^{(\ell)}, \Psi(\{e^{(\ell+1)}(\mathbf{a}_{uv})^\top \psi^{(\ell+1)}(\mathbf{h}_u^{(\ell)}) | u \in \mathcal{N}_v\})\right) \tag{1.6}$$

with e being any function.

Finally (Bacciu et al. 2020) formulate also a neighborhood aggregation in terms of the attention mechanism (Vaswani et al. 2017), which assigns a score to each part of the input of a neural layer. The convolution of Equation (1.4) is extended as follows:

$$\mathbf{h}_v^{(\ell+1)} = \phi^{(\ell+1)}\left(h_v^{(\ell)}, \Psi(\{\alpha_{uv}^{(\ell+1)} * \psi^{(\ell+1)}(\mathbf{h}_u^{(\ell)}) | u \in \mathcal{N}_v\}))\right) \tag{1.7}$$

with $\alpha_{uv}^{(\ell+1)} \in \mathbb{R}$ is the attention score associated with $u \in \mathcal{N}_v$. Note that this score is unrelated to the edge information, which needs to be calculated separately.

Similarly to CNNs, in some tasks, a sampling operator is required in order to summarize information and remove noise. In GNNs **pooling** is achieved in two different ways. **Adaptive pooling** is a parametric, learnable mechanism, which consists usually in the training of a differentiable clustering layer based on the vertices, or on the edges of the input. **Topological pooling** insead, is non-adaptive and thus does not require to be differentiable, and their results

are not task-dependent (Bacciu et al. 2020). Differently from the neighbor aggregation operator, it is more difficult to provide a general formulation of a pooling operator. The reader can refer to (Bacciu et al. 2020) for a description of different pooling techniques common in the related literature.

GNNs pose interesting opportunities toward the integration of the methodologies described in this work, both in terms of new learnable frameworks for generic graph structures, and for the improvement of the GNNs standard operators described above. Future research will be devoted toward these two directions.

Other Deep Neural Networks

Over the course of the years, many new models have been devised for other tasks. For motivation of space, we cannot cover them all. However, among the others, we shortly describe two models that make use of the contextual information in the learning process.

Recurrent Neural Networks (RNNs) (Elman 1990; Jordan 1986) deserve mention here. These architectures were specifically designed to deal with sequential data, like sentences or time series. At each time step, an input of the sequence is fed to the network and is then mapped to a hidden space. The hidden output is used to generate a feature embedding to be used in the loss computation, and it is also passed as an **hidden input** for the processing of the next input in the sequence. Unfortunately, the architecture has been shown to suffer the same vanishing gradient problem as MLPs. Thus the supervision generated by the first time steps becomes accidentally not important as the recent ones, failing to learn long term relations (Hochreiter et al. 2001). In the attempt to address this limitation, (Hochreiter and Schmidhuber 1997) introduced the Long Short-Term Memory (LSTM) architecture, which modifies the hidden layer computation by introducing a series of operators meant to

provide a "memory" capability to the network.

Siamese Neural Networks (SNNs) (Bromley et al. 1994) are models thought to address metric learning problems. Formally, the architecture consists of two DNNs with shared weights that are fed with two inputs at a time. The aim of the learning is the generation of feature embeddings that can be efficiently used for comparison. In other words, similar observations correspond to similar feature embeddings in the learned feature space. The Siamese networks are important in the context of this work because they were used in (Elezi et al. 2020).

1.2 Contextual Information Basics

With the term **contextual information**, we refer to all the information deduced by the interpretation of an entity in relation to other entities. Examples of contextual information include the recognition of an object in the foreground thanks to the background surrounding it, or the meaning of a word thanks to its position in a sentence.

Contextual information arises in the presence of more entities sharing the same structure (pixels in an image, words in a sentence), which are usually grouped in a **context** (the image, sentence respectively). If the data can be organized in this way, then we can exploit contextual information for our tasks.

Due to its rather generic definition, every machine learning model exploits a certain level of contextual information, in that each observation benefit from the supervision generated by the previous ones seen by the model during the learning (in particular, cf. 1.1). However, some effort has been put into devising specific models or modules aimed to perform a direct **disambiguation** of observations based on their context. In this section, we present the main algorithms used as part of the work presented in this thesis. Again, the following discussion is not exhaustive and is aimed to provide the reader with the basic knowledge of such

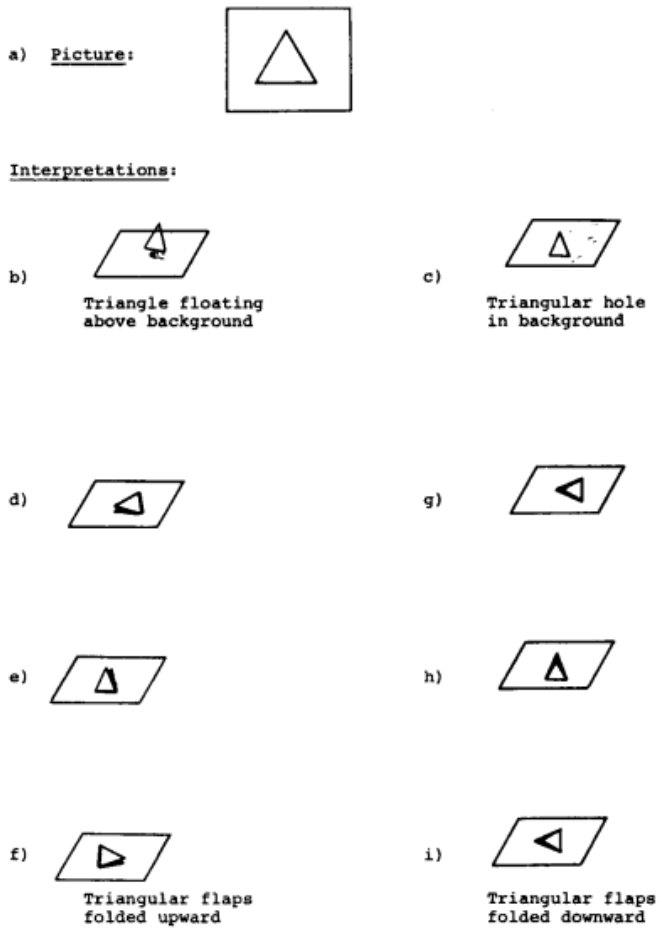


Figure 1.6: The triangle problem (taken from (Rosenfeld, Hummel and Zucker 1976)).

methods.

Relaxation Labeling

One of the first attempts to model the contextual information consists of the wide variety of models reunited under the name of **Relaxation Labeling**. Such models first arose in the seminal work of (Rosenfeld, Hummel and Zucker 1976). In this paper, the authors address the problem of **label disambiguation**: given a set of objects $B = \{b_1, \ldots, b_n\}$, a set of labels $\Lambda = \{\lambda_1, \ldots, \lambda_m\}$, the task consists in assigning the label the object according to a set of **constraints**

on the assignments. Four different models, with a varying level of constraint strictness and stochasticity, are devised. Probably, the most well-known example in this field is the so-called "triangle problem" (Rosenfeld, Hummel and Zucker 1976). Suppose to have observed a scene containing a set of objects $b_i, \ldots, b_n$, for example individual line segments forming a triangle, as shown in Figure 1.6, but these objects have not been identified unambiguously yet. For example, each segment, can be interpreted as a convex/concave dihedral angle, with both faces visible, or a dihedral angle with just one face visible. In this latter case, the line is an occluding edge and the visible face can be on either side of the line, so in total we have four possible cases and $4^3 = 64$ possible line settings. However, many of them generate "impossible objects", thus a disambiguation can be done by considering that when segments meet at a vertex not all possible combinations of these interpretations will be consistent, and the ambiguity can be solved (partially or entirely). For example, in this problem, the possible settings are only 8.

In the discrete, deterministic model each object b_i is related to a set of compatible labels $\Lambda_i \subseteq \Lambda$ and each pair of objects (b_i, b_j) gives rise to a set $\Lambda_{ij} \subseteq \Lambda_i \times \Lambda_j$ of compatible pairs of labels. If $(\lambda, \mu) \in \Lambda_{ij}$ then it is possible for b_i to have label λ and b_j to have label μ. $\Lambda_{ii} = \{(\lambda, \lambda) \mid \lambda \in \Lambda_i\}$ by definition. A **labeling** $\mathcal{L} = (L_i, \ldots, L_n)$ is an assignment of a set of labels $L_i \subseteq \Lambda$ to each $b_i \in B$. A labeling is called **consistent** if $\forall i, j, \forall \lambda \in L_i,\ (\{\lambda\} \times L_j) \cap \Lambda_{ij} \neq \emptyset$, meaning that for $i \neq j$ each pair of objects (b_i, b_j) and each label λ in L_i there exists a label μ in L_j that is compatible with λ, i.e. (λ, μ). The disambiguation process starts from $\lambda^{(0)} = \{\Lambda_1, \ldots, \Lambda_n\}$ and iteratively prunes $L_i^{(t)}$ discarding any label λ such that $(\{\lambda\} \times L_j^{(t)}) \cap \Lambda_{ij} = \emptyset$ for some j. It is possible to prove that $\mathcal{L}^{(\infty)}$ is consistent.

The discrete model is paired with a deterministic algorithm to reach consistent labeling, which is, however, inapplicable in many real cases. In practice, such strong assumptions on the compatibility of labels cannot be made.

Rather, it is preferable to define different degrees of compatibility, computed, for example, by mean of a real-valued function. For this reason, the authors investigate different non-deterministic model, finally proposing a non-linear probabilistic setting. Each object b_i is assigned to a vector of probabilities $\mathbf{p}_i = (p_{i1}, \ldots, p_{im})$, representing the confidences of labeling object b_i with each of its possible labels. The vectors are collected in a **labeling matrix** (simply, **labeling**) $\mathbf{p} = (\mathbf{p}_1, \ldots, \mathbf{p}_n)^\top$. In addition, assignments between objects – also known as **hypotheses** – are weighted by a compatibility factor $r_{ij}(\lambda, \mu)$ representing the degree of agreeableness between the hypotheses "b_i is labeled with λ" and "b_j is labeled with μ".

Intuitively, the main desideratum of a label disambiguation process is to (a) increase the confidence of those assignments that are highly compatible with other high confidence assignments (b) decrease the confidence of those assignments that are highly incompatible with other high confidence assignments. The authors propose to use the correlation coefficient as a compatibility measure and the following update rule for the disambiguation:

$$p_{i\lambda}^{(t+1)} = \frac{p_{i\lambda}^{(t)})[1 + q_{i\lambda}^{(t)}]}{\sum_\mu p_{i\mu}^{(t)}[1 + q_{i\mu}^{(t)}]} \tag{1.8}$$

$q_{i\lambda}^{(t)}$ is called support for the assignment (i, λ) and is computed as follows:

$$q_{i\lambda}^{(t)} = \sum_j d_{ij} \sum_\mu r_{ij}(\lambda, \mu) p_{j\mu}^{(t)} \tag{1.9}$$

with d_{ij} being coefficients such that $\sum_j d_{ij} = 1$[1].

The success of Relaxation Labeling in addressing a variety of classification tasks was followed by the need of formally justifying its performance. (Hummel and Zucker 1983) devised an optimization scheme, that, under some restrictions, can be paired with an explicit functional, called **Average Local Consistency**

[1] In other publications the coefficients d_{ij} are not used. The work presented in this thesis follow this convention as well.

(ALC):

$$A(\mathbf{p}) = \sum_{i=1}^{n} \sum_{\lambda=1}^{m} p_{i\lambda} q_{i\lambda} \tag{1.10}$$

A new, principled notion of consistency is formulated such that the ALC can be maximized to obtain consistent labelings. In particular, let $\mathbb{K}$ be the space of all labelings, then a labeling $\mathbf{p}$ is said to be **consistent** if:

$$\sum_{\lambda=1}^{m} p_{i\lambda} q_{i\lambda} \geq \sum_{\lambda=1}^{m} v_{i\lambda} q_{i\lambda} \quad \forall i \in \{1, \ldots, n\} \tag{1.11}$$

for all $\mathbf{v} \in \mathbb{K}$. If the inequality is strict then the labeling is said to be **strictly consistent**. The new optimization framework is used to obtain a new ReLab scheme, with local convergence properties. In particular:

$$\mathbf{p}^{(t+1)} = \mathbf{p}^{(t)} + \Psi(\mathbf{p}^{(t)}, \mathbf{q}^{(t)}) \tag{1.12}$$

With Ψ being a gradient projection operator that ensures that the updated $\mathbf{p}^{(t+1)}$ lies within the space $\mathbb{K}$. It can be shown that the standard ReLab operator presented in Equation 1.8 is an approximation of this scheme, thus providing evidence of the success of the standard ReLab in many applications.

A further analysis has been performed by (Pelillo 1997) where the convergence properties of the standard ReLab operator are theoretically proven using results drawn from the consistency theory developed in (Hummel and Zucker 1983) and the dynamical system theory. The paper adopt a slightly different version of the update rule, which does not compromise the general nature of the stated theoretical results:

$$p_{i\lambda}^{(t+1)} = \frac{p_{i\lambda}^{(t)} q_{i\lambda}^{(t)}}{\sum_{\mu=1}^{m} p_{i\mu} q_{i\mu}} \tag{1.13}$$

Assuming non-negative compatibilities. In the next sections, we will refer specifically to this form when speaking about ReLab. The process is shown to be

well-defined if the compatibility matrix is zero-symmetric ($b_{ij} = 0 \implies b_{ji} = 0$). In addition, it is proved that a labeling $\mathbf{p} \in \mathbb{K}$ is consistent if and only if the following conditions holds: (a) $q_i(\lambda) = c_i$, whenever $p_i(\lambda) > 0$, (b) $q_i(\lambda) \leq c_i$, whenever $p_i(\lambda) = 0$, for some non-negative constants $c_1, \ldots, c_n$, and if $\mathbf{p}$ is consistent then it is a fixed point for the ReLab process. Perhaps, the most important result is of the paper characterizes the strictly consistent labeling as asymptotically stable equilibrium points, under the condition of non-negativity and symmetry of the compatibilities, while the ReLab operator is a growth transformation (Baum and Sell 1968) that effectively maximizes the ALC.

Graph Transduction Games

Graph Transduction Games (GTG) is a popular algorithm for semi-supervised learning tasks developed by (Erdem and Pelillo 2012), which can be ultimately seen as a particular instance of Relaxation Labeling. The general idea, indeed, is to **propagate** the information of labeled instances to classify the unlabeled ones in a **consistent** way.

Like the name suggests, GTG belongs to the Graph Transduction (GT) (Zhu and Ghahramani 2002) subfamily of semi-supervised learning, whose aim consists in the classification of unlabeled objects starting from a small set of labeled ones. In GT the dataset is modeled with a graph whose vertices are the observations, while the edges represent the similarity between said observations. In short, the provided label information is propagated over the unlabeled observations according to the similarities.

More formally, let $G = (V, E, w)$ be a graph, with $V = L \cup U$ being the vertex set. $L = \{(f_1, y_1), ..., (f_l, y_l)\}$ represents the labeled observations, where $f_i \in \mathbb{R}^d$ is a real-valued **feature vector** describing observation i, and $y_i \in \{1, 2, \ldots, m\}$ is its associated label. $U = \{f_{l+1}, ..., f_n\}$ is the set of unlabeled observations. $E \subseteq V \times V$ is the set of edges connecting the vertices and $w : E \to \mathbb{R}_{\geq 0}$ is a

	rock	*paper*	*scissors*
rock	0, 0	-1, 1	1, -1
paper	1, -1	0, 0	-1, 1
scissors	-1, 1	1, -1	0, 0

Table 1.1: Payoff matrix of the rock-paper-scissors game.

similarity measure that assigns a non-negative value to each edge $(u, v) \in E$, and can be summarized by a weight matrix W.

In (Zhu 2005), GT takes in input W along with a initial probability distribution for each observation – one-hot labels for $(f_i, y_i) \in L$, soft labels for $f_i \in U$ – and iteratively applies a function $P : \Delta^m \to \Delta^m$ where Δ^m is the standard simplex. At each iteration, if the distributions of labeled objects have changed, they are reset. Once the algorithm reaches the convergence, the resulting final probabilities give a labeling over the entire set of objects. The reader is encouraged to refer to (Zhu 2005) for a detailed description of algorithms and applications on graph transduction.

While, in general, label propagation methods theory is defined in terms of a graph Laplacian regularization, GTG is, on the contrary, grounded on non-cooperative game theory notions. In the following, we will give some basics in order to clarify the rationale under the development of this algorithm.

A **Game** is a triple $\mathcal{G} = (\mathcal{I}, S, \boldsymbol{\pi})$. $\mathcal{I} = \{1, \ldots, n\}$ with $n \geq 2$ is called the set of **players**. Each player i can play a **pure strategy** chosen from a set of strategies $S_i = \{1, \ldots, m_i\}$, collected in the strategy space $S = \bigtimes_{i \in \mathcal{I}, S_i}$. $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_n) : S \to \mathbb{R}^n$ is instead the **combined payoff function**, which provides a "reward" to each player, considering each individual pure strategy in relation with the others played by the other players. The common example is the rock-paper-scissors game, a two-player game where each player can choose among three possible strategies $S_1 = S_2 = \{rock, paper, scissors\}$. The payoff matrix is given in Table 1.2.

In order to model uncertainty in the strategy decision process, we introduce

the concept of **mixed strategy** $\mathbf{p}_i \in \Delta_i$ which is a probability distribution over the set S_i of pure strategies, for player i. Δ_i is also called **standard simplex** and is defined as follows:

$$\Delta_i = \left\{ \mathbf{p}_i \in \mathbb{R}^{m_i} \mid \sum_{h=1}^{m_i} p_{ih} = 1 \wedge x_{ih} \geq 0, \ \forall h \right\} \tag{1.14}$$

The collection of all the mixed strategies $\mathbf{p} = (\mathbf{p}_1, \ldots, \mathbf{p}_n)$, called **mixed strategy profile**, resides in the **mixed strategy space** $\Theta = \bigtimes_{i \in \mathcal{I}} \Delta_i$. It is often common that all players share the same set of pure strategies. In such case, the mixed strategy space is referred with the term **standard multi-simplex** and denoted with $\Delta^{n \times m}$. The **expected payoff** of a player can be represented as:

$$u_i(\mathbf{p}) = \sum_{s \in S} \pi_i(s) x_{1,s_1} \cdot \ \ldots \ \cdot x_{n,s_n} \tag{1.15}$$

With $s = (s_1, \ldots, s_n)$.

A player's **best replies** against a mixed strategy profile $\mathbf{p}' \in \Theta$, consists of the set of mixed strategies that grant the highest payoff for said player:

$$\beta_i(\mathbf{p}') = \{ \mathbf{p}_i \in \Delta_i \mid u_i(\mathbf{p}_i, \mathbf{p}'_{-i}) \geq u(\mathbf{p}''_i, \mathbf{p}'_{-i}), \ \forall \mathbf{p}''_i \in \Delta_i \} \tag{1.16}$$

Best replies define the fundamental concept in non-cooperative game theory: the **Nash equilibrium**. A mixed strategy profile $\mathbf{p}^*$ is said to be a Nash equilibrium, if it is the best reply to itself, that is:

$$u_i(\mathbf{p}^*_i, \mathbf{p}^*_{-i}) \geq u_i(\mathbf{p}_i, \mathbf{p}^*_{-i}) \tag{1.17}$$

for all $i \in \mathcal{I}$, $x_i \in \Delta_i$, and $x_i \neq x^*_i$. This concept is motivated by the fact that in a non-cooperative game, all the players should be motivated to maximize their payoff, eventually reaching a Nash equilibrium.

GTG consists of a game where the players are the vertices of a graph,

representing the observations of a dataset $\mathcal{D} = \mathcal{D}_l \cup \mathcal{D}_u$ with $\mathcal{D}_l$ and $\mathcal{D}_u$ being respectively the labeled and unlabeled observations. The pure strategies are the possible classes to be used for labeling, that is $S_1 = \cdots = S_n = \Lambda = \{1, \ldots, m\}$. The game is a special class of multiplayer games, known as **polymatrix games** (Miller and Zucker 1991), where multiple "rounds" of the game are played between pair of players. The payoff function of each player is:

$$\pi_i(s) = \sum_{j=1}^{n} \mathbf{A}_{ij}(s_i, s_j) \tag{1.18}$$

where a_{ij} is the partial payoff matrix between players i and j. The payoffs are then computed as:

$$u_i(\mathbf{e}_i^h, \mathbf{p}_{-i}) = \sum_{j=\mathcal{I}_u} (\mathbf{A}_{ij}\mathbf{p}_j)_h + \sum_{k=1}^{m} \sum_{j \in \mathcal{I}_{\mathcal{D}|k}} \mathbf{A}_{ij}(h, k) \tag{1.19}$$

$$u_i(\mathbf{p}) = \sum_{j \in \mathcal{I}_u} \mathbf{p}_i^\top \mathbf{A}_{ij}\mathbf{p}_j + \sum_{k=1}^{m} \sum_{j \in \mathcal{I}_{\mathcal{D}|\lambda}} \mathbf{p}_i^\top (\mathbf{A}_{ij})_\lambda \tag{1.20}$$

Where $I_{\mathcal{D}|\lambda}$, is the set of labeled observations belonging to class λ.

The partial payoff matrix $\mathbf{A}_{ij}$ is defined in terms of the similarity w_{ij} between the two players/observations. In particular, $\mathbf{A}_{ij} = w_{ij} \cdot I_m$, with I_m being an identity matrix of size m. In other words, if two players share the same label they receive a payoff equal to their similarity. $\mathbf{A} = (\mathbf{A}_{ij})$ can be written as the Kronecker product of $\mathbf{A} = I_m \bigotimes \mathbf{W}$, with $\mathbf{W} = (w_{ij})$.

The core of GTG consists of an iterative procedure aimed to find the Nash equilibrium of such game. In (Erdem and Pelillo 2012) a result from Evolutionary Game Theory (Weibull 1997), named Replicator Dynamics (RD) (Maynard Smith 1982) is used. The RD are a class of dynamical systems that perform a natural selection process on a many different populations of individuals playing pure strategies in different shares. The idea is to lead the fittest strategies to survive while the others to go extinct. In particular, an initial matrix of

mixed strategy profiles $\mathbf{p}^{(0)}$ is initialized such that labeled observations get assigned to one-hot encodings, representing **extreme mixed strategies**, while the unlabeled observations can be either initialized according to some prior knowledge, or with a uniform distribution, if it is unavailable.

The GTG updates $\mathbf{p}^{(0)}$ according to the following rule:

$$p_{ih}^{(t+1)} = \frac{p_{ih}^{(t)} u_{ih}(e_i^{(h)}, \mathbf{p}_{-i}^{(t)})}{u_i(\mathbf{p}^{(t)})} \tag{1.21}$$

Note that the GTG process preserves the one-hot labelings, and modifies the prior distributions of the unlabeled observations, as expected by a label propagation algorithm.

A fundamental fact in our discussion is that $\mathbf{A}$ represents a special case of the compatibility matrix $\mathbf{R}$ described in Section 1.2, and GTG can be effectively seen as the standard Relaxation Labeling process described above. Indeed, consider the following:

$$\begin{aligned}
u_i(\mathbf{e}_i^h, \mathbf{p}_{-i}) &= \sum_{j=\mathcal{I}_u} (\mathbf{A}_{ij}\mathbf{p}_j)_h + \sum_{k=1}^{m} \sum_{j \in \mathcal{I}_{\mathcal{D}|k}} a_{ij}(h,k) \\
&= \sum_{j=\mathcal{I}_u} \sum_{k=1}^{m} a_{ij}(h,k) p_{jk} + \sum_{j \in \mathcal{I}_{\mathcal{D}|k}} \sum_{k=1}^{m} a_{ij}(h,k) p_{jk} \\
&= \sum_{j=1}^{n} \sum_{k=1}^{m} a_{ij}(h,k) p_{jk} = q_{ih} \\
u_i(\mathbf{p}) &= \sum_{j \in \mathcal{I}_u} \mathbf{p}_i^\top \mathbf{A}_{ij} \mathbf{p}_j + \sum_{k=1}^{m} \sum_{j \in \mathcal{I}_{\mathcal{D}|k}} \mathbf{p}_i^\top (\mathbf{A}_{ij})_k \qquad (1.22) \\
&= \sum_{j \in \mathcal{I}_u} \sum_{\mu=1}^{m} \sum_{\mu'=1}^{m} p_{i\mu} a_{ij}(\mu, \mu') p_{j\mu'} + \sum_{j \in \mathcal{I}_{\mathcal{D}|k}} \sum_{k=1}^{m} \mathbf{p}_i^\top (\mathbf{A}_{ij})_k \\
&= \sum_{\mu=1}^{m} p_{i\mu} \sum_{j=1}^{n} \sum_{\mu'=1}^{m} a_{ij}(\mu, \mu') p_{j\mu'} \\
&= \sum_{\mu=1}^{m} p_{i\mu} q_{i\mu} \qquad (1.23)
\end{aligned}$$

Thus Equation (1.21) is equivalent to the update rule of a standard Relaxation

Labeling operator.

The following result establishes a connection between the local maximizers of the ALC/expected payoff, and the Nash equilibria of the game:

Theorem 1.2.1. *Suppose* $\mathbf{A}$ *is symmetric, that is,* $\mathbf{A}_{ij} = \mathbf{A}_{ji}\ \forall i, j \in \mathcal{I}$. *Then any local maximum* $\mathbf{x}^* \in \Theta$ *of equation* (1.20) *is a Nash equilibrium point of the polymatrix game (Miller and Zucker 1991).*

1.3 Conclusions

This introductive chapter has given a panoramic view of the main concepts treated in the following chapters. We are now ready to delve into the specific applications of GTG and ReLab.

CHAPTER 2

Transductive Label Augmentation for Improved Deep Network Learning

Abstract A major impediment to the application of deep learning to real-world problems is the scarcity of labeled data. Small training sets are in fact of no use to deep networks as, due to the large number of trainable parameters, they will very likely be subject to overfitting phenomena. On the other hand, the increment of the training set size through further manual or semi-automatic labelings can be costly, if not possible at times. Thus, the standard techniques to address this issue are transfer learning and data augmentation, which consists of applying some sort of "transformation" to existing labeled instances to let the training set grow in size. Although this approach works well in applications such as image classification, where it is relatively simple to design suitable transformation operators, it is not obvious how to apply it in more structured scenarios. Motivated by the observation that in virtually all application domains it is easy to obtain unlabeled data, in this chapter we take a different perspective and propose a **label augmentation** approach. We start from a small, curated labeled dataset and let the labels propagate through a larger set of unlabeled data using graph transduction techniques. This allows us to naturally use

(second-order) similarity information which resides in the data, a source of information which is typically neglected by standard augmentation techniques. In particular, we show that by using the game theoretic transductive process described in Section 1.2 we can create larger and accurate enough labeled datasets which use results in better trained neural networks. Preliminary experiments are reported which demonstrate a consistent improvement over standard image classification datasets.

2.1 Introduction

Deep neural networks (DNNs) have met with success multiple tasks, and testified a constantly increasing popularity, being able to deal with the vast heterogeneity of data and to provide state-of-the-art results across many fields and domains (LeCun, Bengio and Hinton 2015; Schmidhuber 2015). Convolutional Neural Networks (CNNs) (Fukushima and Miyake 1982; LeCun et al. 1989b) are one of the protagonists of this success. Starting from AlexNet (Krizhevsky, Sutskever and Hinton 2012), until the most recent convolutional-based architectures (He et al. 2016; Huang et al. 2017; Szegedy et al. 2015) CNNs have proved to be especially useful in the field of computer vision, improving the classification accuracy in many datasets (Alex Krizhevsky 2009; Deng et al. 2009).

However, a common caveat of large CNNs is that they require a lot of training data in order to work well. In the presence of classification tasks on small datasets, typically those networks are **pre-trained** in a very large dataset like ImageNet (Deng et al. 2009), and then **finetuned** on the dataset the problem is set on. The idea is that the pre-trained network has stored a decent amount of information regarding features which are common to the majority of images, and in many cases this knowledge can be transferred to different datasets or to solve different problems (image segmentation, localization, detection, etc.). This technique is referred as **transfer learning** (Yosinski et al. 2014) and

has been an important ingredient in the success and popularization of CNNs. Another important technique – very often paired with the previous one – is **data augmentation**, through which small transformations are directly applied on the images. A nice characteristic of data augmentation is that it is agnostic toward algorithms and datasets. (Ciresan, Meier and Schmidhuber 2012) used this technique to achieve state-of-the-art results in the MNIST dataset (LeCun and Cortes 2010), while (Krizhevsky, Sutskever and Hinton 2012) used the method almost without any changes to improve the accuracy of their CNN in the ImageNet dataset. Since then, data augmentation has been used in virtually every implementation of CNNs in the field of computer vision.

Despite the practicality of the above-mentioned techniques, when the number of images per class is extremely small, the performances of CNNs rapidly degrade and leave much to be desired. The high availability of unlabeled data only solves half of the problem, since the manual labeling process is usually costly, tedious and prone to human error. Under these assumptions, we propose a new method to perform an automatic labeling, called **transductive label augmentation**. Starting from a very small labeled dataset, we set an automatic label propagation procedure, that relies on graph transduction techniques (GTG), to label a large unlabeled set of data. This method takes advantage of second-order similarity information among the data objects, a source of information which is not directly exploited by traditional techniques. To assess our statements, we perform a series of experiments with different CNN architectures and datasets, comparing the results with a first-order "label propagator".

In summary, the contributions presented in this chapter are as follows: a) by using graph transductive approaches, we propose and develop the aforementioned label augmentation method and use it to improve the accuracy of state-of-the-art CNNs in datasets where the number of labels is limited; b) by gradually increasing the number of labeled objects, we give detailed results in three standard computer vision datasets and compare the results with the results of

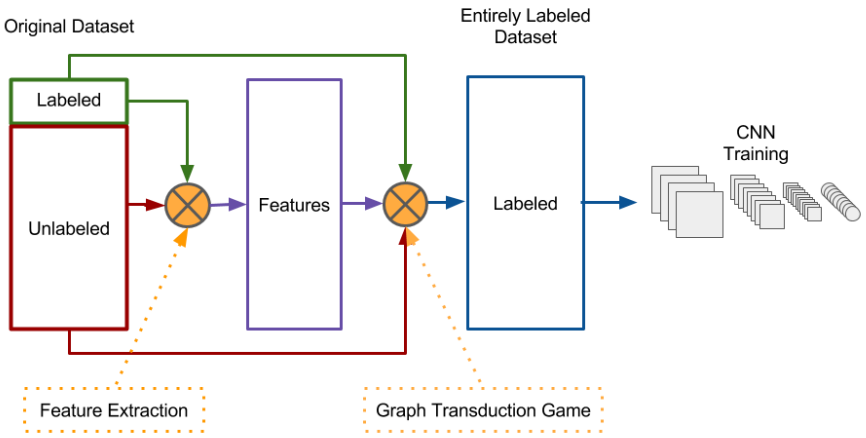


Figure 2.1: The pipeline of our method. The dataset consists of labeled and unlabeled images. First, we extract features from the images, and then we feed the features (and the labels of the labeled images) to graph transduction games. For the unlabeled images, we use a uniform probability distribution as 'soft-labeling'. The final result is that the unlabeled points get labeled, thus the entire dataset can be used to train a convolutional neural network.

CNNs; c) we perform a comparison of the results by replacing our transductive algorithm with linear support vector machines (SVM) (Cortes and Vapnik 1995), and later with other transductive algorithms (Label Spreading (Zhou et al. 2004) and Harmonic Function (HF) (Zhu, Ghahramani and Lafferty 2003)); d) we give directions for future work and how the method can be used on other domains.

Related Work

Semi-supervised label propagation has a long history of usage in the field of machine learning (Vapnik 1998). Starting from an initial large dataset, with a small portion of labeled observations the traditional way of using semi-supervised learning is to train a classifier only in the labeled part, and then use the classifier to predict labels for the unlabeled part. The labels predicted in this way are called **pseudo-labels**. The classifier is then trained in the entire dataset, considering the pseudo-labels as if they were real labels.

Different methods with the same intent have been previously proposed. In

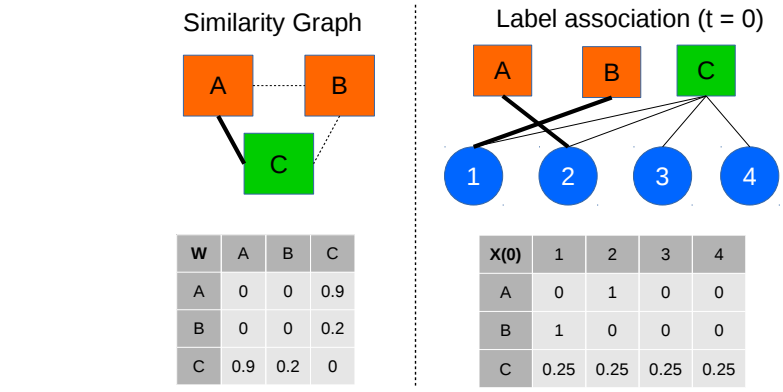
deep learning in particular, there have been devised algorithms to use data with a small number of labeled observations. (Lee 2013) trained the network jointly in both the labeled and unlabeled points. The final loss function is a weighted loss of both labeled and unlabeled points, where in the case of the unlabeled points, the pseudo-label is determined by the highest score proposed by the model. (Häusser, Mordvintsev and Cremers 2017) optimized a CNN on such a way as to produce embeddings that have high similarities for the observations that belong to the same class. (Kingma et al. 2014) used a totally different approach, developing a generative model that allows for effective generalization from small labeled datasets to large unlabeled ones.

In all the mentioned methods, the way how the unlabeled data has been used can be considered as an intrinsic property of their engineered neural networks. Our choice of CNNs as the algorithm used for the experiments was motivated because CNNs are state-of-the-art models in computer vision, but the approach is more general than that. The method presented in this chapter does not even require a neural network and in principle, non-feature based observations (i.e graphs) can be considered, as long as a similarity measure can be derived for them. At the same time, the method shows good results in relatively complex image datasets, improving over the results of state-of-the-art CNNs.

2.2 Graph Transduction Game

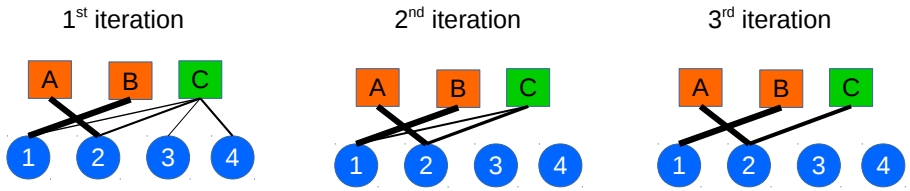
The used methodology relies on the Graph Transduction Games algorithm, previously described in Section 1.2. The similarity function between players (objects) can be given or computed starting from the features. Given two objects i, j and their features f_i, f_j, their similarity $\omega(i, j)$ is computed following the method proposed by (Zelnik-Manor and Perona 2005):

$$\omega(i,j) = exp\left\{-\frac{||f_i - f_j||_2}{\sigma_i\ \sigma_j}\right\} \tag{2.1}$$



W	A	B	C
A	0	0	0.9
B	0	0	0.2
C	0.9	0.2	0

X(0)	1	2	3	4
A	0	1	0	0
B	1	0	0	0
C	0.25	0.25	0.25	0.25



X(1)	1	2	3	4
A	0	1	0	0
B	1	0	0	0
C	0.21	0.47	0.16	0.16

X(2)	1	2	3	4
A	0	1	0	0
B	1	0	0	0
C	0.15	0.85	0	0

X(3)	1	2	3	4
A	0	1	0	0
B	1	0	0	0
C	0	1	0	0

Figure 2.2: The dynamics of the GTG. The algorithm takes in input similarities between objects and hard/soft labelings of the object themselves. After three iterations, the algorithm has converged, generating a pseudo-label with 100% confidence.

where σ_i corresponds to the distance between i and its 7-nearest-neighbors. Similarity values are stored in matrix W.

GTG is iterated until convergence, this means either the distance between two successive steps is sufficiently small (formally $||\mathbf{p}^{(t+1)} - \mathbf{p}^{(t)}||_2 \leq \varepsilon$) or a certain amount of iterations is reached (See (Pelillo 1997) for a detailed analysis). In practical applications one could set the ε to a small number but typically 10-20 iterations are sufficient. A schematization of the GTG process is shown in Figure 2.2

2.3 Label Generation

The framework can be applied to a dataset with many unlabeled objects to perform an automatic labeling and thus increase the availability of training objects. The pipeline of the process is presented in Figure 2.1. This chapter presents experiments with datasets for image classification, but our approach is rather general in its nature. Similar methodologies have been successfully applied in other domains, e.g. bioinformatics (Vascon et al. 2018) and matrix factorization (Tripodi, Vascon and Pelillo 2016), and the works presented in the successive chapters of this thesis.

Preliminary step: both the labeled and unlabeled sets can be refined to obtain more informative feature vectors. We used fc7 features of CNNs trained on ImageNet, but in principle, any type of features can be considered. Our particular choice was motivated because fc7 features work significantly better than traditional computer vision features (SIFT (Lowe 2004) and its variations). While this might seem counter-intuitive (using pre-trained CNNs on ImageNet, while we are solving the problem of limited labeled data), we need to consider that our datasets are different from ImageNet (they come from different distributions), and by using some other datasets to pre-train our networks, we are not going against the spirit of the idea of this work.

Step 1: the objects are assigned to initial probability distributions, needed to start the GTG. The labeled ones use their respective one-hot label representations, while the unlabeled ones can be set to a uniform distribution among all the labels. In presence of previous possessed information, some labels can be directly excluded in order to start from a multi-peaked distribution, which if chosen wisely, can improve the final results.

Step 2: the extracted features are used to compute the similarity matrix W. The literature (Zelnik-Manor and Perona 2005) presents multiple methods to obtain a W matrix and extra care should be taken when performing this

step, since an incorrect choice in its computation can determine a failure in the transductive labeling.

Step 3: once W is computed, graph transduction games can be played (up to convergence) among the objects to obtain the final probabilities which determine the label for the unlabeled objects.

The resulting labeled dataset can then be used to train a classification model. This is very convenient for several reasons: 1) CNNs are fully parametric models, so we do not need to store the training set in memory like in the case of graph transduction. In some aspect, the CNN is approximating in a parametric way the GTG algorithm; 2) the inference stage on CNNs is extremely fast (real-time); 3) CNN features can be used for other problems, like image segmentation, detection and classification, something that we cannot do with graph-transduction or with classical machine learning methods (like SVM). In the next section we will report the results obtained from state-of-the-art CNNs, and compare those results with the same CNNs trained only on the labeled part of the dataset.

2.4 Experiments

In order to assess the quality of the algorithm, we used it to automatically label three known realistic datasets, namely **Caltech-256** (Griffin, Holub and Perona 2007), **Indoor Scene Recognition** (Quattoni and Torralba 2009) and **SceneNet-100** (Kadar and Ben-Shahar 2014). **Caltech-256** contains 30607 images belonging to 256 different categories and it is used for object recognition tasks. **Indoor Scene Recognition** is a dataset containing 15620 images of different common places (restaurants, bedrooms, etc.), divided in 67 categories and, as the name says, it is used for scene recognition. **SceneNet-100** database is a publicly available online ontology for scene understanding that organizes scene categories according to their perceptual relationships. The dataset contains 10000 real-world images, separated into 100 different classes.

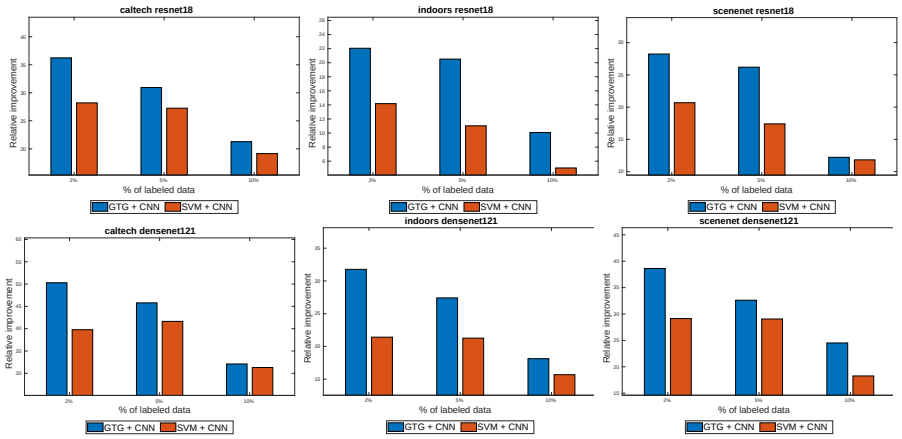


Figure 2.3: Results obtained on different datasets and CNNs. Here the relative improvements with respect to the CNN accuracy is reported. As can be seen, the biggest advantage of our method compared to the other two approaches, is when the number of labeled points is extremely small (2%). When the number of labeled points increases, the difference on accuracy becomes smaller, but nevertheless our approach continues being significantly better than CNN, and in most cases, it gives better results than the alternative approach.

Each dataset was split in a training (70%) and a testing (30%) set. In addition, we further randomly split the training set in a small labeled part and a large unlabeled one, according to three different percentages for labeled objects (2%, 5%, 10%). For feature representation, we used two models belonging to state-of-the-art CNN families of architectures, ResNet and DenseNet. In particular, we used the smallest models offered in PyTorch library, the choice motivated by the fact that our datasets are relatively small, and so models with smaller number of parameters are expected to work better. The features were combined to generate the similarity matrix W, as described in Eq. 2.1. The matrix for GTG model was initialized as described in the previous section. We ran the GTG algorithm up to convergence, with the pseudo-labels being computed by doing an *argmax* over the final probability vectors. We then trained **ResNet18** (RN18) and **DenseNet121** (DN121) in the entire dataset, by not having a distinction between labels and pseudo-labels, using Adam optimizer (Kingma and Ba 2014a) with $3 * 10^{-4}$ learning rate. We think that

accuracy	caltech		indoors		scenenet	
2% labeled	RN18	DN121	RN18	DN121	RN18	DN121
GTG + CNN	**0.529**	**0.588**	**0.478**	**0.506**	**0.460**	**0.455**
LS + CNN	0.459	0.517	0.434	0.486	0.359	0.435
HF + CNN	0.393	0.463	0.372	0.438	0.312	0.319
LP + CNN	0.397	0.462	0.373	0.425	0.293	0.357
CNN	0.193	0.216	0.315	0.302	0.08	0.18
F score	caltech		indoors		scenenet	
2% labeled	RN18	DN121	RN18	DN121	RN18	DN121
GTG + CNN	**0.471**	**0.534**	**0.336**	**0.393**	**0.439**	**0.435**
LS + CNN	0.392	0.462	0.352	0.403	0.342	0.417
HF + CNN	0.367	0.446	0.262	0.335	0.331	0.342
LP + CNN	0.321	0.381	0.29	0.111	0.278	0.344
CNN	0.091	0.108	0.151	0.131	0.076	0.18

Table 2.1: The results of our algorithm, compared with the results of Label Spreading (LS), Harmonic Function (HF), Label Propagation (LP) and CNN, when only 2% of the dataset is labeled. We see that in all three datasets and two different neural networks, our approach gives significantly better results than the competing approaches

accuracy	caltech		indoors		scenenet	
5% labeled	RN18	DN121	RN18	DN121	RN18	DN121
GTG + CNN	**0.667**	**0.71**	**0.552**	**0.585**	**0.628**	**0.626**
LS + CNN	0.589	0.647	0.496	0.561	0.523	0.456
HF + CNN	0.589	0.665	0.527	0.555	0.549	0.588
LP + CNN	0.532	0.600	0.454	0.502	0.442	0.513
CNN	0.440	0.526	0.425	0.438	0.381	0.456
F score	caltech		indoors		scenenet	
5% labeled	RN18	DN121	RN18	DN121	RN18	DN121
GTG + CNN	**0.624**	**0.674**	**0.440**	0.476	**0.606**	**0.62**
LS + CNN	0.544	0.601	0.428	**0.503**	0.511	0.557
HF + CNN	0.542	0.636	0.444	0.482	0.531	0.574
LP + CNN	0.477	0.551	0.394	0.432	0.430	0.506
CNN	0.370	0.467	0.279	0.291	0.376	0.448

Table 2.2: The results of our algorithm, compared with the results of Label Spreading (LS), Harmonic Function (HF), Label Propagation (LP) and CNN, when only 5% of the dataset is labeled. We see that in all three datasets and two different neural networks, our approach gives significantly better results than the competing approaches

the results reported in this section are conservative, and can be improved with a more careful training of the networks, and by doing an exhaustive search over the space of hyper-parameters.

For comparison, we performed an alternative approach, by replacing GTG with a first-order information algorithm, namely linear SVM, in addition to other well-known laabel propagation algorithms: **Label Spreading** (Zhou et al. 2004), **Harmonic Function** (HF) (Zhu, Ghahramani and Lafferty 2003) and

accuracy	caltech		indoors		scenenet	
10% labeled	RN18	DN121	RN18	DN121	RN18	DN121
GTG + CNN	**0.714**	**0.746**	**0.577**	**0.628**	**0.675**	**0.681**
LS + CNN	0.636	0.702	0.541	0.592	0.631	0.608
HF + CNN	0.646	0.716	0.548	0.595	0.578	0.643
LP + CNN	0.594	0.672	0.490	0.553	0.499	0.565
CNN	0.599	0.655	0.527	0.563	0.544	0.599
F score	caltech		indoors		scenenet	
10% labeled	RN18	DN121	RN18	DN121	RN18	DN121
GTG + CNN	**0.681**	**0.717**	**0.490**	**0.558**	**0.646**	**0.665**
LS + CNN	0.601	0.675	0.465	0.549	0.620	0.601
HF + CNN	0.607	0.689	0.466	0.523	0.568	0.634
LP + CNN	0.545	0.635	0.411	0.480	0.488	0.554
CNN	0.554	0.615	0.414	0.466	0.538	0.589

Table 2.3: The results of our algorithm, compared with the results of Label Spreading (LS), Harmonic Function (HF), Label Propagation (LP) and CNN, when only 10% of the dataset is labeled. We see that in all three datasets and two different neural networks, our approach gives significantly better results than the competing approaches

Label Propagation (Zhu and Ghahramani 2002): LP performs hard clamping of input labels which yields to avoiding change on the original label distribution at every iteration. LS adopts soft clamping instead, where initial assignments are changed by a fraction α at each iteration. Moreover, employing regularization, the cost employed in LS differentiates from LP, which provides better robustness to noise. HF tries instead to compute a function f by minimizing a corresponding energy function $E(f)$. The solution is *harmonic* and this property can be exploited to propagate information according to the aforementioned smoothness principle. While we experimented also with kernel SVM, we saw that its results are significantly worse than those of linear SVM, most likely because the features were generated from a CNN and so they are already quite good, having transformed the feature space in order to solve the classification problem linearly.

On Table 2.1 we give the results of the accuracy and F score on the testing set, in all three datasets, while the number of labels is only 2% for each of the datasets (400 observations for Caltech-256, 200 observations for Indoor, and 140 observations for Scenenet). In all three datasets, and both CNNs, our results are

Method	10	50	100	250
Pi Model	0.1	0.1	0.362	0.741
Mean Teacher	0.1	0.1	0.266	0.734
Pseudo-Labels	0.13	0.335	0.603	0.778
VAT	0.119	0.227	0.321	0.705
Transfer Learning	0.289	0.518	0.605	0.711
LS + CNN	0.534	0.666	0.714	0.758
HF + CNN	0.113	0.232	0.352	0.659
GTG + CNN	0.575	0.733	0.764	0.791

Table 2.4: The results of our method, tested on the CIFAR-10 dataset, compared with the results of other deep semisupervised learning methods and replacements of GTG with other transductive methods.

significantly better than those of CNNs trained only in the labeled data, or the results of the alternative approaches. Table 2.2 and Table 2.3 give the results of the accuracy and F score while the number of labeled images is 5%, respectively 10%. It can be seen that with the number of labeled points increasing, the performance boost of our model becomes smaller, but our performance still gives better results to the alternative approach in all bar one case, and it gives significantly better results than CNN in all cases.

Figure 2.3 shows the results of our approach compared with the linear SVM and with the results of CNN. We plotted the relative improvement of our model and the alternative approach over CNN. When the number of labels is very small (2%), in all three datasets we have significantly better improvements compared with the alternative approach. Increasing the number of labels to 5% and 10%, this trend persists. In all cases, our method gives significant improvements compared to CNN trained on only the labeled part of the dataset, with the most interesting case (only 2% of labeled observations), our model gives 36.24% relative improvement over CNN for **ResNet18** and 50.29% relative improvement for **DenseNet121**.

In addition, we compared the results with deep semisupervised models such as Π-model (Laine and Aila 2017), Mean-Teacher (Tarvainen and Valpola 2017), Pseudo-Label (Lee 2013) and VAT (Miyato et al. 2019). For a comprehensive

Method	10	50	100	250
Pi Model	0.1	0.1	0.11	0.681
Mean Teacher	0.1	0.1	0.12	0.695
Pseudo-Labels	0.132	0.236	0.498	0.764
VAT	0.118	0.193	0.323	0.641
Transfer Learning	0.271	0.506	0.589	0.689
LS + CNN	0.484	0.621	0.684	0.703
HF + CNN	0.113	0.309	0.556	0.642
GTG + CNN	0.514	0.56	0.713	0.719

Table 2.5: The results of our method, tested on the CIFAR-10 dataset, compared with the results of other deep semisupervised learning methods, and replacements of GTG with other transductive methods. All the neural models were pretrained on a selected subset of the ImageNet dataset.

evaluation of these methods, the reader can refer to (Oliver et al. 2018). The dataset used for the experiments was CIFAR-10 (Alex Krizhevsky 2009), using the same preprocessing proposed in the paper. The results, presented in Table 2.4, show that our method outperformed all the competitors, while the performance gap decreases with the increment of the labeled instances.

We replicated also the experiment done in (Oliver et al. 2018), where the network is pretrained on a subset of ImageNet not containing similar classes as those of CIFAR-10. Table 2.4 shows the results. Again, it can be seen that our method outperforms the others in very-low level regimes, with the exception of the training with 250 labeled instances, where only Pseudo-Labels performs better. In particular, apart Label Spreading, all the other methods perform very poorly in presence of 10, 50, 100 labeled instances.

2.5 Conclusions and Future Work

In this chapter, we proposed and developed a game-theoretic model which can be used as a semi-supervised learning algorithm in order to label the unlabeled observations and so augment datasets. Different types of algorithms (including state-of-the-art CNNs) can then be trained on the extended dataset, where the "pseudo-labels" can be treated as normal labels.

Our method is not the only semi-supervised learning model used to train deep learning methods. However, to the best of our knowledge, the other methods are directed towards deep learning and incorporated within the learning algorithm itself. On the contrary, we offer a different perspective, developing a model which is algorithm-agnostic, and which does not even need the data to be on feature-based format.

We believe that the true potential of the model can be unleashed when the data is in some non-traditional format. In particular, we plan to use our model in the fields of bio-informatics and natural language processing, where non-conventional learning algorithms need to be developed. A direct extension of this work is to embed into the model the similarity between classes which has been proven to significantly boost the performances of learning algorithms.

CHAPTER 3

Unsupervised Domain Adaptation Using Graph Transduction Games

Abstract Unsupervised domain adaptation (UDA) amounts to assigning class labels to the unlabeled instances of a dataset from a target domain, using labeled instances of a dataset from a related source domain. In this paper we propose to cast this problem in a game-theoretic setting as a non-cooperative game and introduce a fully automatized iterative algorithm for UDA based on graph transduction games (GTG). The main advantages of this approach are its principled foundation, guaranteed termination of the iterative algorithms to a Nash equilibrium and soft labels quantifying uncertainty of the label assignment process. We also investigate the beneficial effect of using pseudo-labels from linear classifiers to initialize the iterative process. The performance of the resulting methods is assessed on publicly available object recognition benchmark datasets involving both shallow and deep features. Results of experiments demonstrate the suitability of the proposed game-theoretic approach for solving UDA tasks.

3.1 Introduction

The success of deep learning in computer vision classification tasks relies on the availability of a large amount of images annotated with their ground truths.

However, manual label annotation is typically an expensive process and may contain wrong annotations. In order to overcome these limitations, Semi-Supervised Learning (SSL) approaches have been developed, usually involving the training of a classifier from a large dataset with plenty of unlabeled data and substantially less annotated data. In some cases, however, it is expensive to obtain unlabeled data too, resorting instead on data coming from a different source. This problem is best formulated in the Unsupervised Domain Adaptation setting, where unlabeled data comes from a different related distribution than that of the labeled data. Specifically, an annotated source dataset is exploited to infer the labels of an unlabeled target dataset from a different, related domain. Due to the tight relation between SSL and UDA problems, it is not uncommon to approach them with similar techniques (cf. (Häusser, Mordvintsev and Cremers 2017) and (Häusser et al. 2017) for example).

In this chapter we investigate the use of GTG, for domain adaptation, which has been successfully applied in SSL tasks such as in (Elezi et al. 2018), (Vascon et al. 2018), (Tripodi, Vascon and Pelillo 2016) and (Aslan, Vascon and Pelillo 2018), and we show that this approach, paired with a preprocessing step, provides overall improvements in three standard domain adaptation cases. Specifically, we propose a fully automatized pipeline to perform UDA with GTG, comparing our results with those of recent methods, trying also to include prior information provided by a simple classifier, i.e. Logistic Regression. We perform also a comparison of GTG with other three standard graph-transductive algorithms. The picture that arises from the experimental results is promising and suggests considering graph transduction as a key-module when addressing UDA problems. The choice of using GTG as a transduction algorithm for DA has been motivated by its theoretical properties which guarantee: **i)** a consistent labeling of unknown samples at convergence, **ii)** the output of soft-labelings (probability distribution over the classes) for further refinements, **iii)** the possibility of injecting prior knowledge at the beginning of the transductive

process.

The main contributions of this work are the following:

- We adopt the theory of label consistency of GTG to propose a principled technique for UDA. This will offer a novel perspective on the UDA problem.
- We propose a parameter-free method for UDA based on game theory which bypasses intensive training phase.
- We reach state-of-the-art performance results on publicly available object recognition domain adaptation tasks.

Related work

The literature contains presents many methods for UDA. Here, we focus on the recent approaches used in our comparative assessment.

A variety of DA models aligns the distributions of features from source and target domains by reducing their discrepancy. For instance, CORrelation ALignment (CORAL) (Sun, Feng and Saenko 2016) finds a linear transformation that minimizes the distance between the covariance of source and target. Subspace Alignment (SA) (Fernando et al. 2013) computes a linear map that minimizes the Frobenius norm of the difference between the source and target domains, which are represented by subspaces described by eigenvectors. Feature Level Domain Adaptation (FLDA) (Kouw et al. 2016) models the dependence between the two domains by means of a feature-level transfer model that is trained to describe the transfer from source to target domain. FLDA assigns a data-dependent weight to each feature representing how informative it is in the target domain. To to do it uses information on differences in feature presence between the source and the target domain.

Recently, end-to-end UDA methods based on deep neural networks have been shown to perform better than the aforementioned approaches. However, they

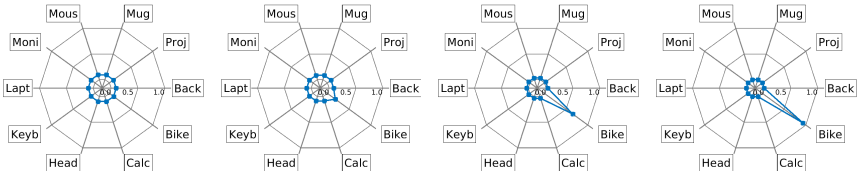


Figure 3.1: From left to right the starting point of the dynamical system and the point of convergence. Evolution of the mixed strategy associated to an observation during the GTG process. As the dynamic is iterated, the entropy progressively decreases and the distribution peaks toward the correct class.

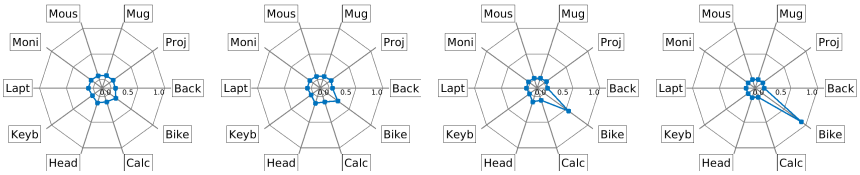


Figure 3.2: From left to right the starting point of the dynamical system and the point of convergence. In this example the dynamics start from three different classes, while in the end, thanks to the refinement of the neighboring mixed strategies the correct class is chosen.

need large train data (Sener et al. 2016), use target labels to tune parameters (Long, Wang and Jordan 2016b) and are sensitive to (hyper-)parameters of the learning procedure (Ganin et al. 2016). Therefore, current state-of-the-art based on this approach start from pre-trained network architectures. Various state-of-the-art methods considered in our comparative analysis use the ResNet pretrained network, like Deep Domain Confusion (DDC) (Tzeng et al. 2014), Deep Adaptation Network (DAN) (Long et al. 2015), Residual Transfer Networks (RTN) (Long, Wang and Jordan 2016b), Reverse Gradient (RevGrad) (Ganin and Lempitsky 2015; Ganin et al. 2016), and Joint Adaptation Networks (JAN) (Long, Wang and Jordan 2016a).

Game theoretic framework

We decided to follow the GTG approach proposed in (Erdem and Pelillo 2012) and explained in Section 1. As for the labeled observations, the initial probabilities $p_i^{(0)}$ can be simply set to the extreme mixed strategy corresponding

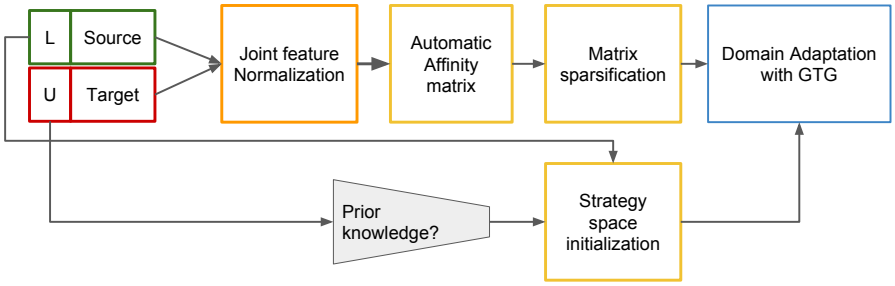


Figure 3.3: Pipeline of the proposed method.

to the labels y_i, namely $p_{ih} = \mathbb{1}(h = y_i)$, where $\mathbb{1}$ is the indicator function, while for the unlabeled ones they can be set either with some prior or a uniform distribution, thus $p_{ih}^{(0)} = 1/m,\ \forall h \in S$ (we tried both cases in our experiments). As in the previous chapter, the dynamics are run until two consecutive steps do not differ significantly or a maximum number of iterations is reached. The affinity matrix computation is explained in Section 3.2. In Figure 3.1 and Figure 3.2 we show two examples of the GTG algorithm in the case of absence of prior knowledge and in its presence, respectively.

3.2 Domain Adaptation with GTG

In this section we present our method, GTDA. We will explain how to cast the unsupervised domain adaptation problem in graph-transduction game setting. We consider a labeled (L) and unlabeled (U) dataset from the source and target domain, respectively. The source labels are propagated to the target instances by playing a non-cooperative multiplayer game in which the players are the observations of the dataset and the strategies are the labels.

The interaction between the observations are represented in terms of a weighted undirected graph in which the edges are weighted by the similarity of observation pairs, hence how much they will affect each others. In particular, the process is illustrated in Fig. 3.3 and explained in the following steps:

Joint feature standardization Given a dataset of features D and two domains $d_s, d_t \in D$ (source and target respectively), we normalize their features jointly as a pre-processing step. We perform two types of normalization on the union of the features: **std** features are scaled by their standard deviation or **z-score** subtract the mean and scaled by their standard deviation. Depending whether the sparsity of the features should be preserved or not, we pick the **std** or **z-score**, respectively.

Initialization of the mixed strategy profile The initial mixed strategy profile of the players, denoted as $\mathbf{p}^{(0)}$ represents the starting point of the game. If prior knowledge is available, we can leverage it for its initialization. In our experimental settings, we explore two different initializations: **i)** in which no prior information is exploited (**no-prior** in the following) and **ii)** where an output from a logistic regression classifier is used (**+LR**). In the latter case, the logistic regression classifier

has been trained for each pair of jointly normalized domains in a dataset. The training has been performed considering only the features belonging to the source, in a 2-fold cross validation setting, with an hyperparameter search for the C variable in the following log-scale range $C = \left[10^{-3}, 10^{4}\right]$. We end up with a LR model $M_{i,j}$ for each pair of domains d_i and d_j in a dataset.

Given an unlabeled observation, the LR model outputs a probability distribution over the classes which is later used as prior knowledge in the strategy space.

The choice of having as prior a probability distribution for each unlabeled object, instead of a one-hot vector, is mandatory since the one-hot vector cannot be updated by the GTG algorithm hence the performance would have been the same as the LR itself.

Algorithm 2 GTDA algorithm

Require: source feature matrix $F_S \in \mathbb{R}^{|S| \times d}$, target feature matrix $F_T \in \mathbb{R}^{|T| \times d}$, one-hot source labels $Y_S \in \Delta^{|S| \times m}$, minimum tolerance ε, maximum number of iterations K.
Ensure: target soft predictions $Y_T \in \Delta^{|T| \times m}$

$N = |S| + |T|$
$\hat{F}_S = \text{normalize}(F_S)$
$\hat{F}_T = \text{normalize}(F_T)$
$P_T = \text{LR}(\hat{F}_S, Y_S, \hat{F}_T)$ ▷ Get log. reg. priors for F_T
$x(0) = \begin{bmatrix} Y_S \\ \hat{P}_T \end{bmatrix}$ ▷ Init. Mixed Strategy Profile
$W = [\omega(i,j)]_{ij}$
$\hat{W} = \text{sparsify}(W)$
$tol = +\infty$, $t = 0$
while $tol \geq \varepsilon$ **and** $t < K$ **do**
    for $i = 1, \ldots, N$ **do**
        $p_i^{(t+1)} = \frac{p_i^{(t)} \odot (\hat{W} p^{(t)})_i}{p_i^{(t)} (\hat{W} \mathbf{p}^{(t)})_i^T}$ ▷ Update rule
    $tol = \|\mathbf{p}^{(t+1)} - \mathbf{p}^{(t)}\|_2$
    $t = t + 1$
$Y_T = \mathbf{p}^{(t-1)}_{|S|:N,1:m}$ ▷ Get the target predictions

Computation of the affinity matrix The core of the GTG is stored in the affinity matrix W, so its computation requires particular care. In our experimental setting, we decided to rely on the following standard similarity kernel:

$$\omega(i,j) = \begin{cases} \exp\left\{-\frac{d(f_i,f_j)^2}{\sigma_i \sigma_j}\right\} & \text{if } i \neq j \\ 0 & \text{else} \end{cases} \tag{3.1}$$

where f_i, f_j are the features of observations i and j respectively, $d(f_i, f_j)$ is the cosine distance between features f_i and f_j. Here, motivated by (Zelnik-Manor and Perona 2005) and (Tripodi, Vascon and Pelillo 2016), we set the scaling parameter σ_i automatically, considering the local statistics of the neighborhood of each point. Accordingly to (Zelnik-Manor and Perona 2005), the value of σ_i is set to the distance of the 7-th nearest neighbour of observation i.

Affinity sparsification The sparsification of the graph plays an important role in the performance of the algorithm. Indeed, filtering out the small noisy

similarities which may bias the utilities in the payoff computation, would prevent incorrect class labelings.

Here, we follow a statistical connectivity principle in random graph, which states that a graph is connected if each node has at least $k = \lfloor log_2(n) \rfloor + 1$ nearest neighbours (Luxburg 2007). The rationale of this choice is that the labels in GTG are propagated from the labeled elements to the unlabeled ones. If the graph is not connected the propagation might get stuck at a certain point. This sparsification ensures that the graph is connected, hence all the nodes are reached at the equilibrium condition of the dynamical system. The sparsification is performed for each node i independently considering the distance value of the k-NN as a threshold for the other nodes in the graph. In order to obtain a symmetric neighborhood, we include the node i in the neighborhood of j (and viceversa) if one of the two is in the neighborhood of the other.

Execution of GTG Once the affinity matrix is computed and the mixed strategy profile is initialized, GTG can be finally played up to convergence. The final probabilities, which determine then the labels for the unlabeled observations, correspond to the adaptation from sources to targets.

In algorithm 2 we present the pseudo-code of the entire method.

3.3 Experimental Setting

To validate our approach, we perform experiments on two publicly available popular datasets for object recognition domain adaptation: the Office-Caltech 10 (Gong et al. 2012) and the Office-31 (Saenko et al. 2010).

Datasets

In the following we present a short description of each datasets used for our experiments.

Office-31 Office-31 is a dataset containing 31 classes divided in 3 domains: *Amazon* (A), *DSLR* (D) and *Webcam* (W). Office-31 has a total of 4110 images, with a maximum of 2478 images per domain. In this dataset we use deep features extracted from the ResNet-50 architecture (He et al. 2016) pretrained on ImageNet.

Office-Caltech Office-Caltech consists in observations taken from the common classes of Office 31 and Caltech256 (10 in total) and are divided in 4 image domains, namely the ones of Office-Caltech and the additional *Caltech* (C). The features we consider are of two kinds: 800 SURF features (Bay, Tuytelaars and Van Gool 2006), which we preprocess by z-score standardization, and deep features in the same fashion as the previous dataset.

Evaluation Criteria

We evaluated and reported the accuracy on the target domain for each adaptation. Accuracy is computed as the fraction of the correctly labeled target instances. Furthermore, we reported the average accuracy per methods and the top-3 performing by different coloring (**best performing**, second and third). Along the analysis of the results we highlighted also the number of hit time that a method perform better than the competitors.

Comparing Methods

In order to assess our method in a broad context, the performance of GTDA has been compared with both standard domain adaptation methods, recent deep-learning based algorithm and baselines classifiers (SVM and LR). Furthermore, we assessed the effectiveness of GTG, replacing it with other GT methods (Label Propagation, Label Spreading and Gaussian Fields and Harmonic Functions).

In our experiments, since we are dealing with more than two classes, we used one-vs-all linear SVM and multi-class logistic regression. More details on

the methods we experimented for comparative analysis are given below.

Shallow Domain Adaptation Methods

The most prevalent domain adaptation methods accomplish domain adaptation task by reducing the discrepancy between source and target distributions via computing a feature transformation. We chose CORrelation ALignment (CORAL) (Sun, Feng and Saenko 2016) and Subspace Alignment (SA) (Fernando et al. 2013) which are two popular methods following this approach. Reported performance for both methods are appealing whereas their application to high dimensional data might be problematic since they are not scalable to high number of features. Another approach for domain adaptation is modeling the dependence of source and target domain in feature level. We experimented by a recent work, namely Feature Level Domain Adaptation (FLDA) (Kouw et al. 2016), that follows this approach. We use published source codes of the shallow DA methods for all datasets.

DNN-based Domain Adaptation Methods

Motivated by their reported stunning performance in recent years, we compared performance of GTDA with the performance of a number of Deep Neural Networks-based domain adaptation methods reported on the Office-31 dataset based on ResNet50 features. Specifically, we make comparison with Deep Domain Confusion (DDC) (Tzeng et al. 2014) where an objective function including an additional domain confusion term is proposed for learning domain-invariant representations for classification, Deep Adaptation Network (DAN) (Long et al. 2015) where more transferable features are learned by adapting source and target distributions in multiple task-specific layers, Residual Transfer Networks (RTN) (Long, Wang and Jordan 2016b) that achieves feature adaptation and classifier adaptation simultaneously by deep residual learning (He et al. 2016), Reverse Gradient (RevGrad) (Ganin and Lempitsky 2015; Ganin

et al. 2016) that improves domain adaptation by employing adversarial training paradigm, and Joint Adaptation Networks (JAN) (Long, Wang and Jordan 2016a) that uses an adversarial learning strategy to maximize a joint maximum mean discrepancy such that distributions of source and target domains be more distinguishable. Despite of high performance accuracies, some disadvantages of DNN-based methods to be taken into account are that they require abundant training data for improvement in performance, they use target labels for parameter tuning (Long, Wang and Jordan 2016b) and their sensitivity to hyperparameters of the learning procedure is high (Ganin et al. 2016). We refer to the performance results reported in (Long, Wang and Jordan 2016a) for the aforementioned DNN-based techniques to make comparison with our technique, hence we will add results for the Office-31 dataset only.

Graph Transduction Techniques for Domain Adaptation

Similarly as in the previous chapter, we compared our game-theoretic graph transduction technique against three other transductive techniques, namely Label Propagation (LP) (Zhu and Ghahramani 2002), Label Spreading (LS) (Zhou et al. 2004) and Harmonic Function (HF) (Zhu, Ghahramani and Lafferty 2003) for the domain adaptation problem. Similar to our method, these techniques exploit the so-called smoothness principle which states that closer instances tend to belong to the same class.

To make a fair comparison with our approach, we provide to these algorithms the same affinity matrix as ours, i.e. W, which is computed using the same scheme of normalization, σ selection and sparsification. For the LS technique, we experimented with a variety of values in the range of $(0, 1)$ for the parameter α. Since we got the best results when $\alpha = 0.2$ which is also the suggested default value we present the results of LS with $\alpha = 0.2$.

Table 3.1: Comparative analysis on Office-31 dataset (ResNet-50 features)

	A→D	A→W	D→A	D→W	W→A	W→D	avg
Baselines							
Source SVM	76.9	73.8	60.3	97.5	59.4	**100.0**	78.0
Source LR	74.7	70.8	60.6	97.5	60.2	**100.0**	77.3
Shallow models							
SA	76.7	75.5	62.2	97.9	60.3	**100.0**	78.8
FLDA-Q	76.3	75.5	59.9	97.5	58.6	99.8	77.9
CORAL	78.9	76.9	59.7	98.2	59.9	**100.0**	78.9
Graph-transductive methods							
Lab Prop	2.4	3.6	3.3	3.6	3.3	99.8	19.3
Lab Spread	77.3	79.2	63.1	98.6	60.8	99.8	79.8
Harmonic Function	73.7	80.3	62.3	98.1	46.8	99.8	76.8
Proposed method, GTDA							
GTDA	80.5	78.0	66.2	**98.9**	62.9	99.8	81.1
GTDA + LR	**82.5**	**84.2**	**67.1**	97.9	**69.1**	99.8	**83.4**

Table 3.2: Comparative analysis on Office-31 dataset (ResNet-50 features)

	A→D	A→W	D→A	D→W	W→A	W→D	avg
Deep Neural Networks (from Long, Wang and Jordan 2016a)							
DDC	76.5	75.6	62.2	96.0	61.5	98.2	78.3
DAN	78.6	80.5	63.6	97.1	62.8	99.6	80.4
RTN	77.5	84.5	63.6	96.8	64.8	99.4	81.6
RevGrad	79.7	82.0	68.2	96.9	67.4	99.1	82.2
JAN-A	**85.1**	**86.0**	**69.2**	96.7	**70.7**	99.7	**84.6**
Proposed method, GTDA							
GTDA	80.5	78.0	66.2	**98.9**	62.9	**99.8**	81.1
GTDA + LR	82.5	84.2	67.1	97.9	69.1	**99.8**	83.4

3.4 Results

We use the notation of $A \rightarrow B$ to indicate the adaptation with A as source and B as target dataset. While we discuss the performance of the techniques, (i) we consider the averaged accuracy over all adaptation tasks and (ii) the number of adaptation tasks that a method outperforms.

Office-31 dataset

We present performance on Office-31 for the shallow and graph transductive methods in Table 3.1 while additional comparisons with deep-learning models is outlined in Table 3.2 reporting results from (Long, Wang and Jordan 2016a).

Non DL methods The results on non-DL methods are reported in Table 3.1. The GTDA outperforms CORAL, i.e. the best performed shallow DA method

on this dataset, by around 2% and 4% in averaged accuracy with and without prior, respectively. When we compare GTDA with other GT methods, i.e. LP., LS and HF, we see that GTDA without prior outperforms all of them with a performance of 81.1%, while Label Spread follows GTDA with the performance of 79.8 %. When the prior knowledge is used in GTDA the performance are far better, being the top performing ones. Another point to highlight from this experiment is that, in general, the transductive methods outperform the shallow models. Without considering the average results, the GTDA+LR outperforms 5 over 6 times the shallow models. When prior is not used the GTDA is the second best performing algorithm (still considering the GTDA+LR).

DL-based methods The results on DL-based methods are reported in Table 3.2. The GTDA outperforms DDC, DAN, RTN and RevGrad which are end-to-end learned system for DA. This is surprising, considering that GTDA does not require an extensive training phase and neither a parameter optimization like in DNN. JAN-A achieves the best averaged accuracy, i.e. 84.6%, on this dataset being for four times the best performing and just once as a second. Among the other DL models no one is able to clearly outperform JAN-A. Our GTDA without prior outperform JAN-A on two cases (D-W, W-D) while in the remaining five it becomes the third best performing only once. When GTDA benefits from prior (GTDA +LR) the performance approaches to JAN-A (83.4 % vs. 84.6 %) becoming the second best algorithm, even outperforming the other DL approaches.

Table 3.3: Comparative analysis on Office-Caltech dataset (SURF features)

	A→C	A→D	A→W	C→A	C→D	C→W	D→A	D→C	D→W	W→A	W→C	W→D	avg
Baselines													
Source SVM	41.0	40.1	42.0	52.7	45.9	**47.5**	33.0	32.1	75.9	38.4	34.6	75.2	46.5
Source LR	42.8	36.3	35.3	**54.1**	42.7	40.7	33.9	31.2	83.1	37.3	32.9	71.3	45.1
Shallow Models													
SA	37.4	36.3	39.0	44.9	39.5	41.0	32.9	34.3	65.1	34.4	31.0	62.4	41.5
FLDA-L	41.5	**45.9**	42.0	49.5	**48.4**	44.1	31.7	34.1	75.6	35.3	33.8	72.6	46.2
FLDA-Q	43.5	43.3	40.7	53.5	44.6	45.1	30.8	31.2	73.2	35.2	32.1	75.8	45.7
CORAL	**45.1**	39.5	**44.4**	52.1	45.9	46.4	37.7	33.8	84.7	35.9	33.7	86.6	48.8
Graph-transductive methods													
Lab Prop	13.4	7.6	9.8	9.6	45.9	9.8	9.6	13.4	9.8	9.6	13.4	**89.8**	20.2
Lab Spread	41.3	36.3	32.5	53.3	47.8	41.4	36.1	34.2	90.2	36.0	34.2	88.5	47.7
Harmonic Function	41.1	38.9	35.9	52.2	47.1	37.6	30.8	29.3	89.2	32.2	32.7	88.5	46.3
Proposed method, GTDA													
GTDA	40.2	37.6	32.9	53.8	46.5	35.9	**41.3**	**39.9**	**92.2**	34.6	**38.5**	89.2	48.5
GTDA + LR	40.2	37.6	38.3	52.6	45.9	45.1	39.2	35.4	**92.2**	**41.0**	37.1	89.2	**49.5**

Table 3.4: Comparative analysis on Office-Caltech dataset (ResNet-50 features)

	A→C	A→D	A→W	C→A	C→D	C→W	D→A	D→C	D→W	W→A	W→C	W→D	avg
Baselines													
Source SVM	91.0	88.5	87.5	94.1	94.9	87.8	90.0	86.1	98.6	89.1	85.9	**100.0**	91.1
Source LR	89.9	91.7	88.5	94.5	93.6	85.1	90.1	85.8	98.0	89.7	85.5	**100.0**	91.0
Shallow models													
SA	89.7	93.0	90.8	94.6	91.1	**93.2**	89.8	84.1	99.0	88.9	84.3	**100.0**	91.5
FLDA-Q	91.1	93.6	92.2	94.5	94.3	89.5	90.3	86.3	97.6	90.3	83.7	**100.0**	91.9
CORAL	85.9	91.1	89.8	94.3	93.0	**93.2**	92.8	86.8	98.6	90.9	85.5	**100.0**	91.8
Graph-transductive methods													
Lab Prop	13.4	7.6	9.8	9.6	7.6	9.8	9.6	13.4	9.8	9.6	13.4	**100.0**	17.8
Lab Spread	87.1	88.5	95.9	93.4	91.1	84.1	88.0	88.6	**99.7**	90.1	85.6	**100.0**	91.0
Harmonic Function	88.6	80.9	85.4	93.5	94.9	89.2	88.1	83.2	**99.7**	57.4	69.2	**100.0**	85.8
Proposed method, GTDA													
GTDA	90.0	87.9	**98.0**	93.5	91.7	79.7	89.4	**89.4**	99.3	93.2	88.8	**100.0**	91.7
GTDA + LR	**91.5**	**98.7**	94.2	**95.4**	**98.7**	89.8	**95.2**	89.0	99.3	**95.2**	**90.4**	**100.0**	**94.8**

Office-Caltech 10 dataset

We present the performance on this dataset in Table 3.4 and 3.4 when SURF and ResNet50 features are used, respectively. In Table 3.4 we see that CORAL outperforms other shallow DA methods, i.e. FLDA and Source SVM, by achieving the second best averaged accuracy over all the methods. While we achieve almost the same performance as CORAL on average (48.8% for CORAL and 48.5% for GTDA) when we do not use priors. When we get benefit of the priors (GTDA+LR) we outperform CORAL by 1% in average accuracy becoming the best one over all methods. We outperform other GT methods both in averaged accuracy and at majority of the adaptation tasks. In particular, our best competitor is CORAL, that reaches the top five results among the shallow models while GTDA with prior outperform six times the shallow ones becoming the more stable algorithm in this setting.

We see in Table 3.4 that when ResNet50 features are used, the performance of all methods are improved significantly over the ones obtained when SURF features were used, except for LP. All shallow DA methods and GTDA (when not using prior) achieve very similar performance. In particular, while FLDA-Q and CORAL outperforms GTDA in averaged accuracy when prior was not used by 0.2% and 0.1%, we see that GTDA reaches the top-3 results while FLDA-Q and CORAL stay at position 1 and 2, respectively. Following them, baseline methods and LS achieve a similar performance in averaged accuracy. When we get benefit from prior we outperform shallow DA methods at every adaptation task (except $C \rightarrow W$), i.e. we reach the best result at eight adaptation tasks, the second best result at three adaptation tasks and the third best result at only one adaptation task, and we are the best among both shallow DA and other GT methods with 94.8% in averaged accuracy with GTDA +LR.

3.5 Conclusions

In this work we have proposed a new algorithm to tackle unsupervised domain adaptation tasks. The methodology is based on GTG, offering a principled perspective to the problem. GTDA proposed here has two main advantages: **i)** it is completely parameter-free and **ii)** it allows the direct embedding of prior knowledge on the target labels to be predicted. The results achieved on publicly available benchmark datasets demonstrate the validity of the proposed approach, whose performance is competitive with respect to state-of-the-art DA techniques with shallow and deep features as well as to other standard graph-based transductive methods. Furthermore, GTDA reaches comparable performance as that of known deep-learning UDA methods. As a future work we plan to extend the comparison with other recent DA techniques using more real-life datasets. As for the methodology we are interested in investigating other aspects, like semi-supervised domain adaptation.

Acknowledgement

This work has been partially funded by the Netherlands Organization for Scientific Research (NWO) within the EW TOP Compartment 1 project 612.001.352.

CHAPTER 4

The Group Loss for Deep Metric Embedding

Abstract Deep metric learning has yielded impressive results in tasks such as clustering and image retrieval by leveraging neural networks to obtain highly discriminative feature embeddings, which can be used to group samples into different classes. Much research has been devoted to the design of smart loss functions or data mining strategies for training such networks. Most methods consider only pairs or triplets of samples within a mini-batch to compute the loss function, which is commonly based on the distance between embeddings. We propose **Group Loss**, a loss function based on a differentiable label-propagation method that enforces embedding similarity across *all* samples of a group while promoting, at the same time, low-density regions amongst data points belonging to different groups. Guided by the smoothness assumption that "similar objects should belong to the same group", the proposed loss trains the neural network for a classification task, enforcing a consistent labelling amongst samples within a class. We show state-of-the-art results on clustering and image retrieval on several datasets, and show the potential of our method when combined with other techniques such as ensembles.

4.1 Introduction

Measuring object similarity is at the core of many important machine learning problems like clustering and object retrieval. For visual tasks, this means learning a distance function over images. With the rise of deep neural networks, the focus has rather shifted towards learning a feature embedding that is easily separable using a simple distance function, such as the Euclidean distance. In essence, objects of the same class (similar) should be close by in the learned manifold, while objects of a different class (dissimilar) should be far away.

Historically, the best performing approaches got deep feature embeddings from the so-called siamese networks (Bromley et al. 1994), which are typically trained using the contrastive loss (Bromley et al. 1994) or the triplet loss (Schultz and Joachims 2003; Weinberger and Saul 2009). A clear drawback of these losses is that they only consider pairs or triplets of data points, missing key information about the relationships between all members of the mini-batch. On a mini-batch of size n, despite that the number of pairwise relations between samples is $\mathcal{O}(n^2)$, contrastive loss uses only $\mathcal{O}(n/2)$ pairwise relations, while triplet loss uses $\mathcal{O}(2n/3)$ relations. Additionally, these methods consider only the relations between objects of the same class (positives) and objects of other classes (negatives), without making any distinction that negatives belong to different classes. This leads to not taking into consideration the global structure of the embedding space, and consequently results in lower clustering and retrieval performance. To compensate for that, researchers relied on other tricks to train neural networks for deep metric learning: intelligent sampling (Manmatha et al. 2017), multi-task learning (Zhang et al. 2016) or hard-negative mining (Schroff, Kalenichenko and Philbin 2015). Recently, researchers have been increasingly working towards exploiting in a principled way the global structure of the embedding space (Çakir et al. 2019; He et al. 2018; Revaud et al. 2019; Wang et al. 2019a), typically by designing ranking loss functions instead of following

the classic triplet formulations.

In a similar spirit, we propose **Group Loss**, a novel loss function for deep metric learning that considers the similarity between all samples in a mini-batch. To create the mini-batch, we sample from a fixed number of classes, with samples coming from a class forming a **group**. Thus, each mini-batch consists of several randomly chosen groups, and each group has a fixed number of samples. An iterative, fully-differentiable label propagation algorithm is then used to build feature embeddings which are similar for samples belonging to the same group, and dissimilar otherwise.

At the core of our method lies the iterative process of GTG, called **replicator dynamics** (Erdem and Pelillo 2012; Weibull 1997), that refines the local information, given by the softmax layer of a neural network, with the global information of the mini-batch given by the similarity between embeddings. The driving rationale is that the more similar two samples are, the more they affect each other in choosing their final label and tend to be grouped together in the same group, while dissimilar samples do not affect each other on their choices. Neural networks optimized with the Group Loss learn to provide similar features for samples belonging to the same class, making clustering and image retrieval easier.

The **contributions** presented in this chapter are four-fold:

- We propose a novel loss function to train neural networks for deep metric embedding that takes into account the local information of the samples, as well as their similarity.
- We propose a differentiable label-propagation iterative model to embed the similarity computation within backpropagation, allowing end-to-end training with our new loss function.
- We perform a comprehensive robustness analysis showing the stability of our module with respect to the choice of hyperparameters.

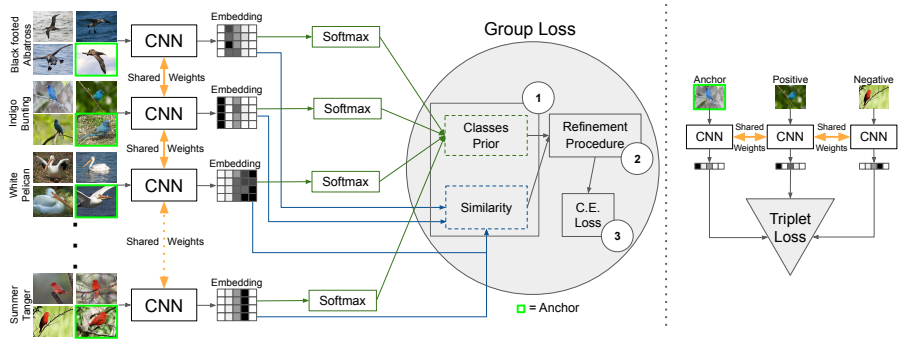


Figure 4.1: A comparison between a neural model trained with the Group Loss (left) and the triplet loss (right). Given a mini-batch of images belonging to different classes, their embeddings are computed through a convolutional neural network. Such embeddings are then used to generate a similarity matrix that is fed to the Group Loss along with prior distributions of the images on the possible classes. The green contours around some mini-batch images refer to **anchors**. It is worth noting that, differently from the triplet loss, the Group Loss considers multiple classes and the pairwise relations between all the samples. Numbers from ① to ③ refer to the Group Loss steps, see Sec 4.3 for the details.

- We show state-of-the-art qualitative and quantitative results in several standard clustering and retrieval datasets.

Our PyTorch (Paszke et al. 2017) code, hyperparameters and the trained models have been released[1].

4.2 Related Work

Classical metric learning losses. The first attempt at using a neural network for feature embedding was done in the seminal work of Siamese Networks (Bromley et al. 1994). A cost function called *contrastive loss* was designed in such a way as to minimize the distance between pairs of images belonging to the same cluster, and maximize the distance between pairs of images coming from different clusters. In (Chopra, Hadsell and LeCun 2005), researchers used the principle to successfully address the problem of face verification.

[1] https://github.com/TheRevanchist/group_loss

Another line of research on convex approaches for metric learning led to the *triplet loss*(Schultz and Joachims 2003; Weinberger and Saul 2009), which was later combined with the expressive power of neural networks (Schroff, Kalenichenko and Philbin 2015). The main difference from the original Siamese network is that the loss is computed using triplets (an anchor, a positive and a negative data point). The loss is defined to make the distance between features of the anchor and the positive sample smaller than the distance between the anchor and the negative sample. The approach was so successful in the field of face recognition and clustering, that soon many works followed. The majority of works on the Siamese architecture consist of finding better cost functions, resulting in better performances on clustering and retrieval. In (Sohn 2016), the authors generalized the concept of triplet by allowing a joint comparison among $N-1$ negative examples instead of just one. (Song et al. 2016) designed an algorithm for taking advantage of the mini-batches during the training process by lifting the vector of pairwise distances within the batch to the matrix of pairwise distances, thus enabling the algorithm to learn feature embedding by optimizing a novel structured prediction objective on the lifted problem. The work was later extended in (Song et al. 2017), proposing a new metric learning scheme based on structured prediction that is designed to optimize a clustering quality metric, i.e., the normalized mutual information (McDaid, Greene and Hurley 2011). Better results were achieved on (Wang et al. 2017), where the authors proposed a novel angular loss, which takes angle relationship into account. A very different problem formulation was given by (Law, Urtasun and Zemel 2017), where the authors used a spectral clustering-inspired approach to achieve deep embedding. A recent work presents several extensions of the triplet loss that reduce the bias in triplet selection by adaptively correcting the distribution shift on the selected triplets (Yu et al. 2018).

Sampling and ensemble methods. Knowing that the number of possible triplets is extremely large even for moderately-sized datasets, and having found that the majority of triplets are not informative (Schroff, Kalenichenko and Philbin 2015), researchers also investigated sampling. In the original triplet loss paper (Schroff, Kalenichenko and Philbin 2015), it was found that using semi-hard negative mining, the network can be trained to a good performance, but the training is computationally inefficient. The work of (Manmatha et al. 2017) found out that while the majority of research is focused on designing new loss functions, selecting training examples plays an equally important role. The authors proposed a distance-weighted sampling procedure, which selects more informative and stable examples than traditional approaches, achieving excellent results in the process. A similar work was that of (Ge et al. 2018) where the authors proposed a hierarchical version of triplet loss that learns the sampling all-together with the feature embedding. The majority of recent works has been focused on complementary research directions such as intelligent sampling (Duan et al. 2019; Ge et al. 2018; Manmatha et al. 2017; Wang et al. 2019b; Xu et al. n.d.) or ensemble methods (Kim et al. 2018; Opitz et al. 2017; Sanakoyeu et al. 2019; Xuan, Souvenir and Pless 2018; Yuan, Yang and Zhang 2017). As we will show in the experimental section, these can be combined with our novel loss.

Other related problems. In order to have a focused and concise work, we mostly discuss methods which tackle image ranking/clustering in standard datasets. Nevertheless, we acknowledge related research on specific applications such as person re-identification or landmark recognition, where researchers are also gravitating towards considering the global structure of the mini-batch. In (He et al. 2018) the authors propose a new hashing method for learning binary embeddings of data by optimizing Average Precision metric. In (He, Lu and Sclaroff 2018; Revaud et al. 2019) authors study novel metric learning functions

for local descriptor matching on landmark datasets. (Çakir et al. 2019) designs a novel ranking loss function for the purpose of few-shot learning. Similar works that focus on the global structure have shown impressive results in the field of person re-identification (Alemu, Pelillo and Shah 2019; Zhao, Xu and Cheng 2019).

Classification-based losses. The authors of (Movshovitz-Attias et al. 2017) proposed to optimize the triplet loss on a different space of triplets than the original samples, consisting of an anchor data point and similar and dissimilar learned proxy data points. These proxies approximate the original data points so that a triplet loss over the proxies is a tight upper bound of the original loss. The final formulation of the loss is shown to be similar to that of softmax cross-entropy loss, challenging the long-hold belief that classification losses are not suitable for the task of metric learning. Recently, the work of (Zhai and Wu 2019) showed that a carefully tuned normalized softmax cross-entropy loss function combined with a balanced sampling strategy can achieve competitive results. A similar line of research is that of (Zheng et al. 2019), where the authors use a combination of normalized-scale layers and Gram-Schmidt optimization to achieve efficient usage of the softmax cross-entropy loss for metric learning. The work of (Qian et al. 2019) goes a step further by taking into consideration the similarity between classes. Furthermore, the authors use multiple centers for class, allowing them to reach state-of-the-art results, at a cost of significantly increasing the number of parameters of the model. In contrast, we propose a novel loss that achieves state-of-the-art results without increasing the number of parameters of the model.

4.3 Group Loss

Most loss functions used for deep metric learning (Ge et al. 2018; Law, Urtasun and Zemel 2017; Manmatha et al. 2017; Schroff, Kalenichenko and Philbin

2015; Sohn 2016; Song et al. 2016; Song et al. 2017; Wang et al. 2017; Wang et al. 2019a; Wang et al. 2019b) do not use a classification loss function, e.g., cross-entropy, but rather a loss function based on embedding distances. The rationale behind it, is that what matters for a classification network is that the output is correct, which does not necessarily mean that the embeddings of samples belonging to the same class are similar. Since each sample is classified independently, it is entirely possible that two images of the same class have two distant embeddings that both allow for a correct classification. We argue that a classification loss can still be used for deep metric learning if the decisions do not happen independently for each sample, but rather jointly for a **whole group**, i.e., the set of images of the same class in a mini-batch. In this way, the method pushes for images belonging to the same class to have similar embeddings. Towards this end, we propose Group Loss, an iterative procedure that uses the global information of the mini-batch to refine the local information provided by the softmax layer of a neural network. This iterative procedure categorizes samples into different groups, and enforces consistent labelling among the samples of a group. While softmax cross-entropy loss judges each sample in isolation, the Group Loss allows us to judge the overall class separation for all samples. In section 4.3, we show the differences between the softmax cross-entropy loss and Group Loss, and highlight the mathematical properties of our new loss.

Overview of Group Loss

Given a mini-batch $\mathcal{B}$ consisting of n images, consider the problem of assigning a class label $\lambda \in \Lambda = \{1, \ldots, m\}$ to each image in $\mathcal{B}$. In the remainder of the manuscript, $X = (x_{i\lambda})$ represents a $n \times m$ (non-negative) matrix of image-label soft assignments. In other words, each row of X represents a probability distribution over the label set Λ ($\sum_{\lambda} x_{i\lambda} = 1$ for all $i = 1 \ldots n$).

The proposed model consists of the following steps (see also Fig. 4.1 and Algorithm 3):

1. **Initialization**: Initialize X, the image-label assignment using the softmax outputs of the neural network. Compute the $n \times n$ pairwise similarity matrix W using the neural network embedding.
2. **Refinement**: Iteratively, refine X considering the similarities between all the mini-batch images, as encoded in W, as well as their labeling preferences.
3. **Loss computation**: Compute the cross-entropy loss of the refined probabilities and update the weights of the neural network using backpropagation.

We now provide a more detailed description of the three steps of our method.

Initialization

Image-label assignment matrix The initial assignment matrix denoted $X(0)$, comes from the softmax outputs of the neural network. We can replace some of the initial assignments in matrix X with one-hot labelings of those samples. We call these randomly chosen samples *anchors*, as their assignments do not change during the iterative refine process and consequently do not directly affect the loss function. However, by using their correct label instead of the predicted label (coming from the softmax output of the NN), they guide the remaining samples towards their correct label.

Similarity matrix A measure of similarity is computed among all pairs of embeddings (computed via a CNN) in $\mathcal{B}$ to generate a similarity matrix $W \in \mathbb{R}^{n \times n}$. In this work, we compute the similarity measure using the Pearson's

correlation coefficient (Pearson 1895):

$$\omega(i,j) = \frac{\mathrm{Cov}[\phi(I_i), \phi(I_j)]}{\sqrt{\mathrm{Var}[\phi(I_i)]\mathrm{Var}[\phi(I_j)]}} \tag{4.1}$$

for $i \neq j$, and set $\omega(i,i)$ to 0. The choice of this measure over other options such as cosine layer, Gaussian kernels, or learned similarities, is motivated by the observation that the correlation coefficient uses data standardization, thus providing invariance to scaling and translation – unlike the cosine similarity, which is invariant to scaling only – and it does not require additional hyperparameters, unlike Gaussian kernels (Elezi et al. 2018).

The fact that a measure of the linear relationship among features provides a good similarity measure can be explained by the fact that the computed features are actually a highly non-linear function of the inputs. Thus, the linear correlation among the embeddings actually captures a non-linear relationship among the original images.

Refinement

In this core step of the proposed algorithm, the initial assignment matrix $P(0)$ is refined in an iterative manner, taking into account the similarity information provided by matrix W. P is updated in accordance with the **smoothness assumption**, which prescribes that similar objects should share the same label.

To this end, let us define the **support** matrix $\Pi = (\pi_{i\lambda}) \in R^{n \times m}$ as

$$\Pi = WP \tag{4.2}$$

whose (i,λ)-component

$$\pi_{i\lambda} = \sum_{j=1}^{n} w_{ij} p_{j\lambda} \tag{4.3}$$

represents the **support** that the current mini-batch gives to the hypothesis that the i-th image in $\mathcal{B}$ belongs to class λ. Intuitively, in obedience to the

smoothness principle, $\pi_{i\lambda}$ is expected to be high if images similar to i are likely to belong to class λ.

Given the initial assignment matrix $P(0)$, our algorithm refines it using the replicator dynamics update rule described in the Introduction:

$$p_{i\lambda}^{(t+1)} = \frac{p_{i\lambda}^{(t)} \pi_{i\lambda}^{(t)}}{\sum_{\mu=1}^{m} p_{i\mu}^{(t)} \pi_{i\mu}^{(t)}} \tag{4.4}$$

In matrix notation, the update rule (4.4) can be written as:

$$P^{(t+1)} = [Q^{(t)}]^{-1} \left[P^{(t)} \odot \Pi^{(t)} \right] \tag{4.5}$$

where

$$Q^{(t)} = \text{diag}(\left[P^{(t)} \odot \Pi^{(t)} \right] \mathbf{e}) \tag{4.6}$$

and $\mathbf{e}$ is the all-one m-dimensional vector. $\Pi^{(t)} = WP^{(t)}$ as defined in (4.2), and $\odot$ denotes the Hadamard (element-wise) matrix product. In other words, the diagonal elements of $Q^{(t)}$ represent the normalization factors in (4.4), which can also be interpreted as the average support that object i obtains from the current mini-batch at iteration t. Figure 4.2 shows a schematization of the refinement procedure.

Loss computation

Once the labeling assignments converge (or in practice, a maximum number of iterations is reached), we apply the cross-entropy loss to quantify the classification error and backpropagate the gradients. Recall, the refinement procedure is optimized via *replicator dynamics*, as shown in the previous section. By studying Equation (4.5), it is straightforward to see that it is composed of fully differentiable operations (matrix-vector and scalar products), and so it can be easily integrated within backpropagation. Although the refining procedure has no parameters to be learned, its gradients can be backpropagated to the

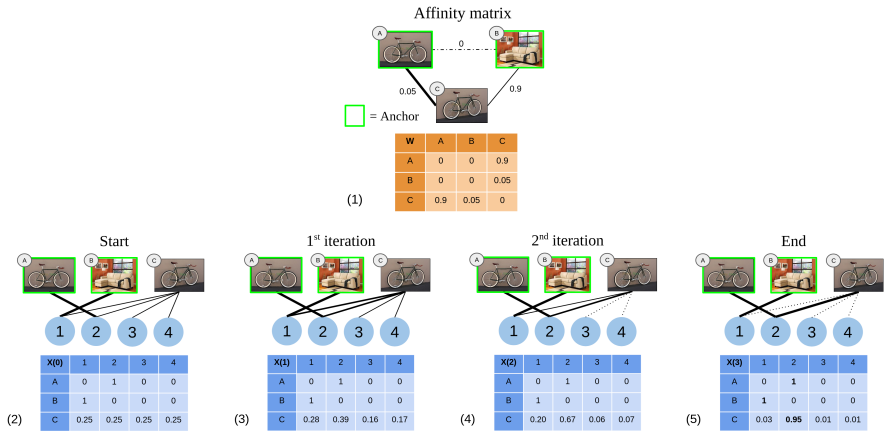


Figure 4.2: A toy example of the refinement procedure, where the goal is to classify sample C based on the similarity with samples A and B. From left to right: (1) The Affinity matrix used to update the soft assignments. (2) The initial labeling of the matrix. (3-4) The process iteratively refines the soft assignment of the unlabeled sample C. (5) At the end of the process, sample C gets the same label of A, (A, C) being more similar than (B, C).

previous layers of the neural network, producing, in turn, better embeddings for similarity computation.

Summary of the Group Loss

In this section, we proposed the Group Loss function for deep metric learning. During training, the Group Loss works by grouping together similar samples based on both the similarity between the samples in the mini-batch and the local information of the samples. The similarity between samples is computed by the correlation between the embeddings obtained from a CNN, while the local information is computed with a softmax layer on the same CNN embeddings. Using an iterative procedure, we combine both sources of information and effectively bring together embeddings of samples that belong to the same class.

During inference, we simply forward pass the images through the neural network to compute their embeddings, which are directly used for image retrieval within a nearest neighbor search scheme. The iterative procedure is not used

Algorithm 3 The Group Loss

Require: Set of pre-processed images in the mini-batch $\mathcal{B}$, set of labels y, neural network ϕ with learnable parameters θ, similarity function ω, number of iterations T

1: Compute feature embeddings $\phi(\mathcal{B}, \theta)$ via the forward pass
2: Compute the similarity matrix $W = [\omega(i, j)]_{ij}$
3: Initialize the matrix of priors $X(0)$ from the softmax layer
4: **for** t = 0, . . . , T-1 **do**
5: $\quad Q^{(t)} = \text{diag}(\left[P^{(t)} \odot \Pi^{(t)}\right] \mathbf{e})$
6: $\quad P^{(t+1)} = [Q^{(t)}]^{-1} \left[P^{(t)} \odot \Pi^{(t)}\right]$
7: Compute the cross-entropy $J(P^{(T)}, y)$
8: Compute the derivatives $\partial J / \partial \theta$ via backpropagation, and update the parameters θ

during inference, thus making the feature extraction as fast as that of any other competing method.

4.4 Experiments

In this section, we compare the Group Loss with state-of-the-art deep metric learning models on both image retrieval and clustering tasks. Our method achieves state-of-the-art results in three public benchmark datasets.

Implementation details

We used the PyTorch (Paszke et al. 2017) library for the implementation of the Group Loss. We chose GoogleNet (Szegedy et al. 2015) with batch-normalization (Ioffe and Szegedy 2015) as the backbone feature extraction network. We pretrain the network on *ILSVRC 2012-CLS* dataset (Russakovsky et al. 2014). For pre-processing, in order to get a fair comparison, we followed the implementation details of (Song et al. 2017). The inputs are resized to 256×256 pixels, and then randomly cropped to 227×227. Like other methods except for (Sohn 2016), we used only a center crop during testing time. We trained all networks in the classification task for 10 epochs. We then trained the network in the Group Loss task for 60 epochs using Adam optimizer (Kingma

and Ba 2014b). After 30 epochs, we lowered the learning rate by multiplying it by 0.1. We found the hyperparameters using random search (Bergstra and Bengio 2012). We used small mini-batches of size $30 - 100$. As sampling strategy, on each mini-batch, we first randomly sampled a fixed number of classes, and then for each of the chosen classes, we sampled a fixed number of samples. We first pre-trained all networks in the classification task for 10 epochs. We then trained our networks on all three datasets for 60 epochs. During training, we used a simple learning rate scheduling in which we divided the learning rate by 10 after the first 30 epochs. We found all hyperparameters using random search (Bergstra and Bengio 2012). For the weight decay ($L2$-regularization) parameter, we searched over the interval $[0.1, 10^{-16}]$, while for the learning rate we searched over the interval $[0.1, 10^{-5}]$, choosing 0.0002 as the learning rate for all networks and all datasets. We achieved the best results with a regularization parameter set to 10^{-6} for *CUB-200-2011*, 10^{-7} for *Cars 196* dataset, and 10^{-12} for *Stanford Online Products* dataset. This further strengthens our intuition that the method is implicitly regularized and it does not require strong regularization.

Benchmark datasets

We performed experiments on 3 publicly available datasets, evaluating our algorithm on both clustering and retrieval metrics. For training and testing, we followed the conventional splitting procedure (Song et al. 2016).

CUB-200-2011 (Wah et al. 2011) is a dataset containing 200 species of birds with $11,788$ images, where the first 100 species ($5,864$ images) are used for training and the remaining 100 species ($5,924$ images) are used for testing.

Cars 196 (Krause et al. 2013) dataset is composed of $16,185$ images belonging to 196 classes. We used the first 98 classes ($8,054$ images) for training and the other 98 classes ($8,131$ images) for testing.

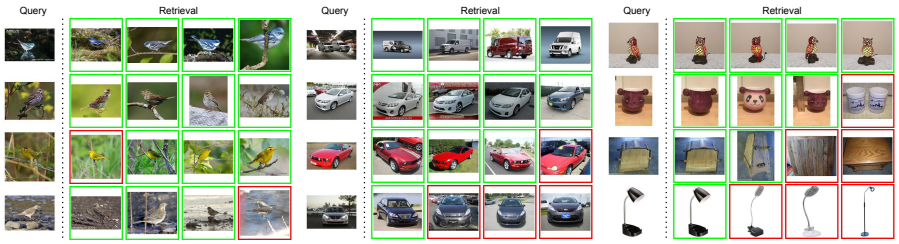


Figure 4.3: Retrieval results on a set of images from the *CUB-200-2011* (left), *Cars 196* (middle), and *Stanford Online Products* (right) datasets using our Group Loss model. The left column contains query images. The results are ranked by distance. The green square indicates that the retrieved image is from the same class as the query image, while the red box indicates that the retrieved image is from a different class.

Stanford Online Products dataset, as introduced in (Song et al. 2016), contains $22,634$ classes with $120,053$ product images in total, where $11,318$ classes ($59,551$ images) are used for training and the remaining $11,316$ classes ($60,502$ images) are used for testing.

Evaluation metrics

Based on the experimental protocol detailed above, we evaluated retrieval performance and clustering quality on data from unseen classes of the 3 aforementioned datasets. For the retrieval task, we calculated the percentage of the testing examples whose K nearest neighbors contain at least one example of the same class. This quantity is also known as Recall@K (Jégou, Douze and Schmid 2011) and is the most used metric for image retrieval evaluation.

Similar to all other approaches, we performed clustering using the K-means algorithm (MacQueen 1967) on the embedded features. Like in other works, we evaluated the clustering quality using the Normalized Mutual Information measure (NMI) (McDaid, Greene and Hurley 2011). The choice of NMI measure is motivated by the fact that it is invariant to label permutation, a desirable property for cluster evaluation.

Results

We now show the results of our model and comparison to state-of-the-art methods. Our main comparison is with other loss functions, e.g., triplet loss. To compare with perpendicular research on intelligent sampling strategies or ensembles, and show the power of the Group Loss, we propose a simple ensemble version of our method. Our ensemble network is built by training l independent neural networks with the same hyperparameter configuration. During inference, their embeddings are concatenated. Note, that this type of ensemble is much simpler than the works of (Kim et al. 2018; Opitz et al. 2018; Sanakoyeu et al. 2019; Xuan, Souvenir and Pless 2018; Yuan, Yang and Zhang 2017), and is given only to show that, when optimized for performance, our method can be extended to ensembles giving higher clustering and retrieval performance than other methods in the literature. Finally, in the interest of space, we only present results for Inception network (Szegedy et al. 2015), as this is the most popular backbone for the metric learning task, which enables fair comparison among methods. In supplementary material, we present results for other backbones, and include a discussion about the methods that work by increasing the number of parameters (capacity of the network) (Qian et al. 2019), or use more expressive network architectures.

Quantitative results

Loss comparison. In Table 4.1 we present the results of our method and compare them with the results of other approaches. On the *CUB-200-2011* dataset, we outperform the other approaches by a large margin, with the second-best model (Classification (Zhai and Wu 2019)) having circa 5 percentage points(*pp*) lower absolute accuracy in Recall@1 metric. On the NMI metric, our method achieves a score of 67.9 which is 1.7*pp* higher than the second-

	CUB-200-2011					CARS 196					Stanford Online Products			
Loss	**R@1**	**R@2**	**R@4**	**R@8**	**NMI**	**R@1**	**R@2**	**R@4**	**R@8**	**NMI**	**R@1**	**R@10**	**R@100**	**NMI**
Triplet (Schroff, Kalenichenko and Philbin 2015)	42.5	55	66.4	77.2	55.3	51.5	63.8	73.5	82.4	53.4	66.7	82.4	91.9	89.5
Lifted Structure (Song et al. 2016)	43.5	56.5	68.5	79.6	56.5	53.0	65.7	76.0	84.3	56.9	62.5	80.8	91.9	88.7
Npairs (Sohn 2016)	51.9	64.3	74.9	83.2	60.2	68.9	78.9	85.8	90.9	62.7	66.4	82.9	92.1	87.9
Facility Location (Song et al. 2017)	48.1	61.4	71.8	81.9	59.2	58.1	70.6	80.3	87.8	59.0	67.0	83.7	93.2	89.5
Angular Loss (Wang et al. 2017)	54.7	66.3	76	83.9	61.1	71.4	81.4	87.5	92.1	63.2	70.9	85.0	93.5	88.6
Proxy-NCA (Movshovitz-Attias et al. 2017)	49.2	61.9	67.9	72.4	59.5	73.2	82.4	86.4	88.7	64.9	73.7	-	-	90.6
Deep Spectral (Law, Urtasun and Zemel 2017)	53.2	66.1	76.7	85.2	59.2	73.1	82.2	89.0	93.0	64.3	67.6	83.7	93.3	89.4
Classification (Zhai and Wu 2019)	59.6	72	81.2	88.4	66.2	81.7	88.9	93.4	96	70.5	73.8	**88.1**	**95**	89.8
Bias Triplet (Yu et al. 2018)	46.6	58.6	70.0	-	-	79.2	86.7	91.4	-	-	63.0	79.8	90.7	-
Ours	**64.3**	**75.8**	**84.1**	**90.5**	**67.9**	**83.7**	**89.9**	**93.7**	**96.3**	**70.7**	**75.1**	87.5	94.2	**90.8**

Table 4.1: Retrieval and Clustering performance on *CUB-200-2011*, *CARS 196* and *Stanford Online Products* datasets. Bold indicates best results.

best method. Similarly, on *Cars 196*, our method achieves best results on Recall@1, with Classification (Zhai and Wu 2019) coming second with a 2*pp* lower score. On *Stanford Online Products*, our method reaches the best results on the Recall@1 metric, around 1.5*pp* higher than Classification (Zhai and Wu 2019) and Proxy-NCA (Movshovitz-Attias et al. 2017). On the same dataset, when evaluated on the NMI score, our loss outperforms any other method, be those methods that exploit advanced sampling, or ensemble methods.

Loss with ensembles. In Table 4.2 we present the results of our ensemble, and compare them with the results of other ensemble and sampling approaches. Our ensemble method (using 5 neural networks) is the highest performing model in *CUB-200-2011*, outperforming the second-best method (Divide and Conquer (Sanakoyeu et al. 2019)) by 1*pp* in Recall@1 and by 0.4*pp* in NMI. In *Cars 196* our method outperforms the second best method (ABE 8 (Kim et al. 2018)) by 2.8*pp* in Recall@1. The second best method in NMI metric is the ensemble version of RLL (Wang et al. 2019a) which gets outperformed by 2.4*pp* from the Group Loss. In *Stanford Online Products*, our ensemble reaches the third-highest result on the Recall@1 metric (after RLL (Wang et al. 2019a) and GPW (Wang et al. 2019b)) while increasing the gap with the other methods in NMI metric.

Comparison with the SoftTriple loss The work of (Qian et al. 2019)) explores another type of classification loss for the problem of metric learning.

	CUB-200-2011					CARS 196					Stanford Online Products			
Loss+Sampling	**R@1**	**R@2**	**R@4**	**R@8**	**NMI**	**R@1**	**R@2**	**R@4**	**R@8**	**NMI**	**R@1**	**R@10**	**R@100**	**NMI**
Samp. Matt. (Manmatha et al. 2017)	63.6	74.4	83.1	90.0	69.0	79.6	86.5	91.9	95.1	69.1	72.7	86.2	93.8	90.7
Hier. triplet (Ge et al. 2018)	57.1	68.8	78.7	86.5	-	81.4	88.0	92.7	95.7	-	74.8	88.3	94.8	-
DAMLRRM (Xu et al. n.d.)	55.1	66.5	76.8	85.3	61.7	73.5	82.6	89.1	93.5	64.2	69.7	85.2	93.2	88.2
DE-DSP (Duan et al. 2019)	53.6	65.5	76.9	61.7	-	72.9	81.6	88.8	-	64.4	68.9	84.0	92.6	89.2
RLL 1 (Wang et al. 2019a)	57.4	69.7	79.2	86.9	63.6	74	83.6	90.1	94.1	65.4	76.1	89.1	95.4	89.7
GPW (Wang et al. 2019b)	65.7	77.0	86.3	91.2	-	84.1	90.4	94.0	96.5	-	78.2	90.5	96.0	-
Teacher-Student														
RKD (Park et al. 2019)	61.4	73.0	81.9	89.0	-	82.3	89.8	94.2	96.6	-	75.1	88.3	95.2	-
Loss+Ensembles														
BIER 6 (Opitz et al. 2017)	55.3	67.2	76.9	85.1	-	75.0	83.9	90.3	94.3	-	72.7	86.5	94.0	-
HDC 3 (Yuan, Yang and Zhang 2017)	54.6	66.8	77.6	85.9	-	78.0	85.8	91.1	95.1	-	70.1	84.9	93.2	-
ABE 2 (Kim et al. 2018)	55.7	67.9	78.3	85.5	-	76.8	84.9	90.2	94.0	-	75.4	88.0	94.7	-
ABE 8 (Kim et al. 2018)	60.6	71.5	79.8	87.4	-	85.2	90.5	94.0	96.1	-	76.3	88.4	94.8	-
A-BIER 6 (Opitz et al. 2018)	57.5	68.7	78.3	86.2	-	82.0	89.0	93.2	96.1	-	74.2	86.9	94.0	-
D and C 8 (Sanakoyeu et al. 2019)	65.9	76.6	84.4	90.6	69.6	84.6	90.7	94.1	96.5	70.3	75.9	88.4	94.9	90.2
RLL 3 (Wang et al. 2019a)	61.3	72.7	82.7	89.4	66.1	82.1	89.3	93.7	96.7	71.8	**79.8**	**91.3**	**96.3**	90.4
Ours 2-ensemble	65.8	76.7	85.2	91.2	68.5	86.2	91.6	95.0	97.1	**91.1**	75.9	88.0	94.5	72.6
Ours 5-ensemble	**66.9**	**77.1**	**85.4**	**91.5**	**70.0**	**88.0**	**92.5**	**95.7**	**97.5**	**74.2**	76.3	88.3	94.6	**91.1**

Table 4.2: Retrieval and Clustering performance of our ensemble compared with other ensemble and sampling methods. Bold indicates best results.

The main difference between our method and theirs is that our method checks the similarity between samples, and then refines the predicted probabilities (via a dynamical system) based on that information. SoftTriple loss instead deals with the intra-class variability, but does not explicitly take into account the similarity between the samples in the mini-batch. They propose to add a new layer with 10 units per class.

We compared the results of (Qian et al. 2019) with our method in Tab. 4.3. SoftTriple loss reaches a higher result than our method in all three datasets in Recall@1 metric, and higher results than the Group Loss on the *CUB-200-2011* and *Stanford Online Products* datasets in NMI metric. However, this comes at a cost of significantly increasing the number of parameters. On the *Stanford Online Products* dataset in particular, the number of parameters of SoftTriple loss is 68.7 million. In comparison, we (and the other methods we compared the results with) used only 16.6 million parameters. In effect, their increase in performance comes at the cost of using a neural network which is four times larger as ours, making results not directly comparable. Furthermore, using multiple centres is crucial for the performance of SoftTriple loss. Fig. 4 in the work of (Qian et al. 2019) shows that when only 1 centre per class is used, the performance drops by 3*pp*, effectively making SoftTriple loss perform worse than the Group Loss by 2*pp*.

We further used the official code implementation to train their model using only one center on the *CARS 196* dataset, reaching 83.1 score in Recall@1, and 70.1 score in NMI metric, with each score being 0.6*pp* lower than the score of The Group Loss. Essentially, when using the same backbone, SoftTriple loss reaches lower results than our method.

As we have shown in the previous section, increasing the number of parameters improves the performances of the network, but it is not a property of the loss function. In fact, a similarly sized network to theirs (Densenet 169) consistently outperforms SoftTriple loss, as can be seen in Tab. 4.3.

Qualitative results In Fig. 4.3 we present qualitative results on the retrieval task in all three datasets. In all cases, the query image is given on the left, with the four nearest neighbors given on the right. Green boxes indicate the cases where the retrieved image is of the same class as the query image, and red boxes indicate a different class. As we can see, our model is able to perform well even in cases where the images suffer from occlusion and rotation. On the *Cars 196* dataset, we see a successful retrieval even when the query image is taken indoors and the retrieved image outdoors, and vice-versa. The first example of *Cars 196* dataset is of particular interest. Despite the fact that the query image contains 2 cars, all four nearest neighbors which have been retrieved have the same class as the query image, showing the robustness of the algorithm to uncommon input image configurations.

Fig. 4.4 visualizes the t-distributed Stochastic Neighbor Embedding (t-SNE) (Maaten and Hinton 2012) of the embedding vectors obtained by our method on the *CUB-200-2011* (Wah et al. 2011) dataset. The plot is best viewed on a high-resolution monitor when zoomed in. We highlight several representative groups by enlarging the corresponding regions in the corners. Despite the large pose and appearance variation, our method efficiently generates a compact feature mapping that preserves semantic similarity.

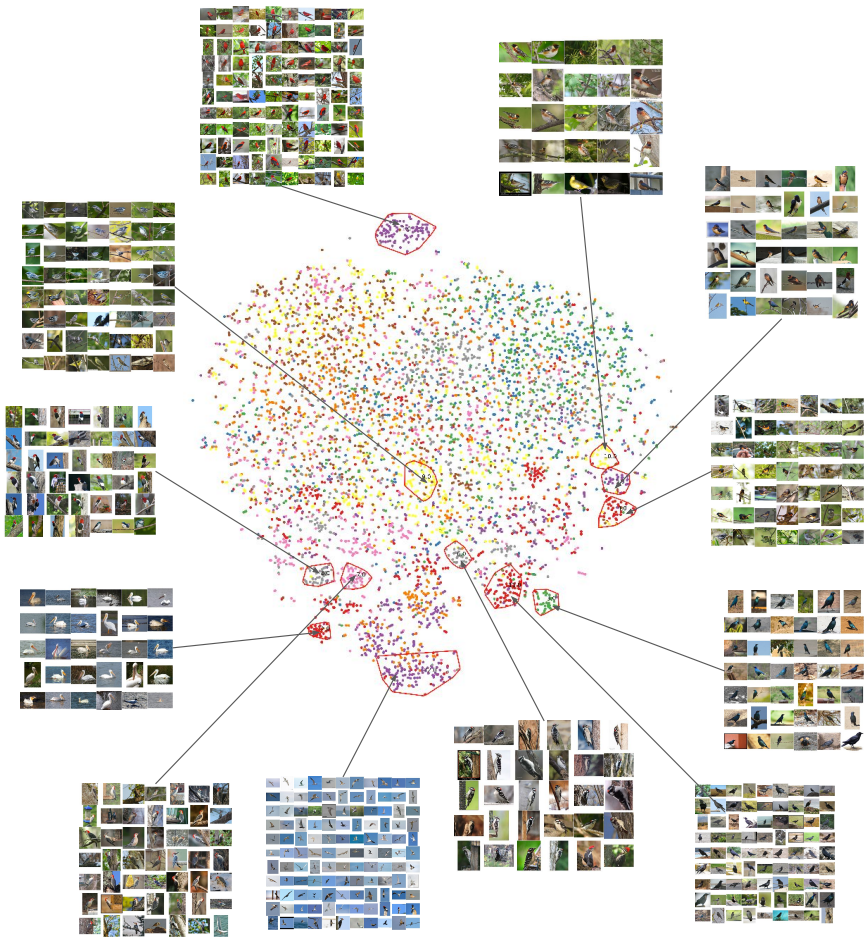


Figure 4.4: t-SNE (Maaten and Hinton 2012) visualization of our embedding on the CUB-200-2011 dataset, with some clusters highlighted. Best viewed on a monitor when zoomed in.

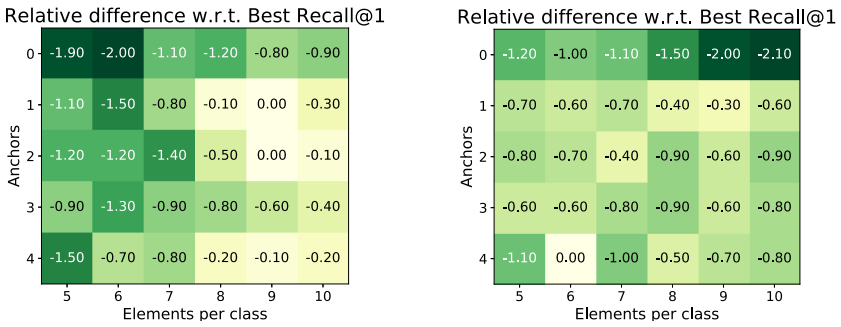


Figure 4.5: The effect of the number of anchors and the number of samples per class, for the *CUB-200-2011* (left) and *CARS* (right) datasets

Robustness analysis

Number of anchors. In Figure 4.5, we show the effect of the number of anchors with respect to the number of samples per class, by performing a grid search over the total number of elements per class versus the number of anchors. We did the analysis on *CUB-200-2011* and *CARS* datasets. The results reported are the percentage point differences in terms of Recall@1 with respect to the best performing set of parameters (see *Recall*@1 = 64.3 in Tab. 4.1). The number of anchors ranges from 0 to 4, while the number of samples per class varies from 5 to 10. It is worth noting that our best setting considers 1 or 2 anchors over 9 samples. Moreover, even when we do not use any anchor, the difference in Recall@1 is no far than 2-2.1 percentage points far. Note that the results decrease mainly when we do not have any labeled sample, i.e. when we use zero anchors.

Number of classes per mini-batch. In Fig. 4.6, we present the change in Recall@1 on the *CUB-200-2011* dataset if we increase the number of classes we sample at each iteration. The best results are reached when the number of classes is not too large. This is a welcome property, as we are able to train on small mini-batches, known to achieve better generalization performance (Keskar et al. 2017).

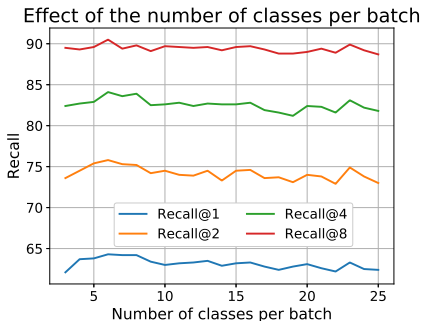


Figure 4.6: The effect of the number of classes per mini-batch.

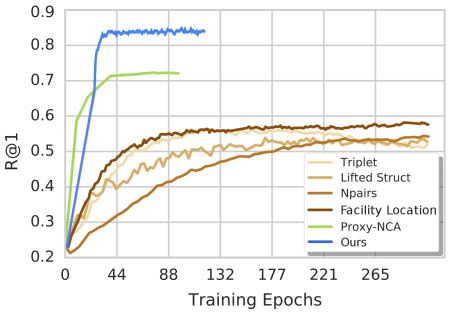


Figure 4.7: Recall@1 as a function of training epochs on Cars196 dataset. Figure adapted from (Movshovitz-Attias et al. 2017).

Convergence rate. In Fig. 4.7, we present the convergence rate of the model on the *Cars 196* dataset. Within the first 30 epochs, our model achieves state-of-the-art results, making it significantly faster than other approaches. Note, that other models, with the exception of Proxy-NCA (Movshovitz-Attias et al. 2017), need hundreds of epochs to converge. Additionally, we compare the training time with Proxy-NCA (Movshovitz-Attias et al. 2017). On a single Volta V100 GPU, the average running time of our method per epoch is 23.59 seconds on *CUB-200-2011* and 39.35 seconds on *Cars 196*, compared to 27.43 and 42.56 of Proxy-NCA (Movshovitz-Attias et al. 2017). Hence, our method is faster than one of the fastest methods in the literature. Note, the inference time of every method is the same because the network is used only for feature embedding extraction during inference.

Implicit regularization and less overfitting. In Figures 4.8 and 4.9, we compare the results of training vs. testing on *Cars 196* and *Stanford Online Products* datasets. We see that the difference between Recall@1 at train and test time is small, especially on *Stanford Online Products* dataset. On *Cars 196* the best results we get for the training set are circa 93% in the Recall@1 measure, only 9 percentage points (*pp*) better than what we reach in the testing set.

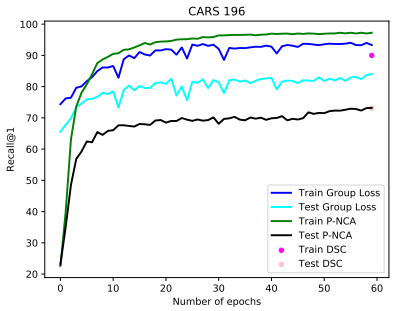


Figure 4.8: Training vs testing Recall@1 curves on *Cars 196* dataset.

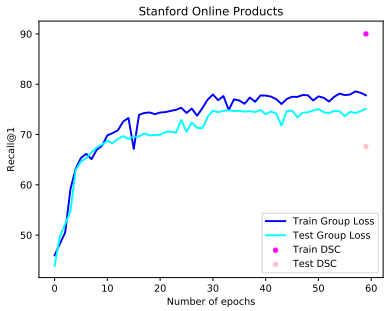


Figure 4.9: Training vs testing Recall@1 curves on *Stanford Online Products* dataset.

From the works we compared the results with, the only one which reports the results on the training set is *Deep Spectral Clustering Learning* (Law, Urtasun and Zemel 2017). They reported results of over 90% in all metrics for all three datasets (for the training sets), much above the test set accuracy which lies at 73.1% on *Cars 196* and 67.6% on *Stanford Online Products* dataset. This clearly shows that our method is much less prone to overfitting.

We further implemented the P-NCA (Movshovitz-Attias et al. 2017) loss function and performed a similar experiment, in order to be able to compare training and test accuracies directly with our method. In Figure 4.8, we show the training and testing curves of P-NCA on the *Cars 196* dataset. We see that while in the training set, P-NCA reaches results of 3*pp* higher than our method, in the testing set, our method outperforms P-NCA by around 10*pp*. Unfortunately, we were unable to reproduce the results of the paper on *Stanford Online Products* dataset.

Furthermore, even when we turn off $L2$-regularization, the generalization performance of our method does not drop at all. Our intuition is that by taking into account the structure of the entire manifold of the dataset, our method introduces a form of regularization. We can clearly see a smaller gap between training and test results when compared to competing methods, indicating less

overfitting. We plan to further investigate this phenomenon in future work.

Additional ablation studies

Temperature scaling We mentioned that as input to the Group Loss (step 3 of the algorithm) we initialize the matrix of priors $X(0)$ from the softmax layer of the neural network. Following the works of (Berthelot et al. 2019; Guo et al. 2017; Zhai and Wu 2019), we applied a sharpening function to reduce the entropy of the softmax distribution. We used the common approach of adjusting the *temperature* of this categorical distribution, known as temperature scaling. Intuitively, this procedure calibrates our network and in turn, provides more informative prior to the dynamical system. Additionally, this calibration allows the dynamical system to be more effective in adjusting the predictions, i.e, it is easier to change the probability of a class if its initial value is 0.6 rather than 0.95. The function is implemented using the following equation:

$$T_{softmax}(z_i) = \frac{e^{z_i/T}}{\sum_i e^{z_i/T}}, \tag{4.7}$$

which can be efficiently implemented by simply dividing the prediction logits by a constant T.

Recent works in supervised learning (Guo et al. 2017) and semi-supervised learning (Berthelot et al. 2019) have found that temperature calibration improves the accuracy for the image classification task. We arrive at similar conclusions for the task of metric learning, obtaining 2.5*pp* better Recall@1 scores on *CUB-200-2011* and 2*pp* better scores on *Cars 196*. Note, the methods of Table 4.1 that use a classification loss, use also temperature scaling.

Dealing with negative similarities Equation (4.2) assumes that the matrix of similarity is non-negative. However, for similarity computation, we used a correlation metric (see Equation (4.1)) which produces values in the range $[-1, 1]$.

In similar situations, different authors proposed different methods to deal with the negative outputs. The most common approach is to shift the matrix of similarity towards the positive regime by subtracting the biggest negative value from every entry in the matrix (Erdem and Pelillo 2012). Nonetheless, this shift has a side effect: If a sample of class k_1 has very low similarities to the elements of a large group of samples of class k_2, these similarity values (which after being shifted are all positive) will be summed up. If the cardinality of class k_2 is very large, then summing up all these small values lead to a large value, and consequently affect the solution of the algorithm. What we want instead, is to ignore these negative similarities, hence we proposed *clamping*. More concretely, we used a $ReLU$ activation function over the output of Equation (4.1).

We compared the results of shifting vs clamping. On the *CARS 196* dataset, we did not see a significant difference between the two approaches. However, on the *CUBS-200-2011* dataset, the Recall@1 metric is 51 with shifting, much below the 64.3 obtained when using clamping. We investigated the matrix of similarities for the two datasets, and we saw that the number of entries with negative values for the *CUBS-200-2011* dataset was higher than for the *CARS 196* dataset. This explains the difference in behavior, and also verifies our hypothesis that clamping is a better strategy to use within *Group Loss*.

Comparison with other backbones In the main paper, we performed all experiments using a GoogleNet backbone with batch normalization. This choice is motivated by the fact that most methods use this backbone, making comparisons fair. In this section, we explore the performance of our method for other backbone architectures, to show the generality of our proposed loss formulation. We chose to train a few networks from Densenet family (Huang et al. 2017). Densenets are a modern CNN architecture which show similar classification accuracy to GoogleNet in most tasks (so they are a similarly strong

Model	CUB			CARS			SOP		
	Params	R@1	NMI	Params	R@1	NMI	Params	R@1	NMI
GL Densenet121	7056356	**65.5**	69.4	7054306	88.1	74.2	18554806	78.2	91.5
GL Densenet161	26692900	64.7	68.7	26688482	**88.7**	74.6	51473462	**80.3**	**92.3**
GL Densenet169	12650980	65.4	**69.5**	12647650	88.4	75.2	31328950	79.4	92.0
GL Densenet201	18285028	63.7	68.4	18281186	88.6	**75.8**	39834806	79.8	92.1
GL Inception v2	10845216	64.3	67.9	10846240	83.7	70.7	16589856	75.1	90.8
SofTriple 10	11307040	65.4	69.3	11296800	84.5	70.1	68743200	78.3	92

Table 4.3: The results of Group Loss in Densenet backbones and comparisons with SoftTriple loss (Qian et al. 2019)

classification baseline [2]). Furthermore, by training multiple networks of the same family, we can study the effect of the capacity of the network, i.e., how much can we gain from using a larger network? Finally, we were interested in studying if the choice of hyperparameters can be transferred from one backbone to another.

We present the results of our method using Densenet backbones in Tab. 4.3. We used the same hyperparameters as the ones used for the GoogleNet experiments, reaching state-of-the-art results on both *CARS 196* and *Stanford Online Products* datasets, even compared to ensemble and sampling methods. The results in *Stanford Online Products* are particularly impressive considering that this is the first time any method in the literature has broken the 80 point barrier in Recall@1 metric. We also reach state-of-the-art results on the *CUB-200-2011* dataset when we consider only methods that do not use ensembles (with the Group Loss ensemble reaching the highest results in this dataset). We observe a clear trend when increasing the number of parameters (weights), with the best results on both *CARS 196* and *Stanford Online Products* datasets being achieved by the largest network, Densenet161 (whom has a lower number of convolutional layers than Densenet169 and Densenet201, but it has a higher number of weights/parameters).

Finally, we studied the effects of hyperparameter optimization. Despite that

[2]The classification accuracy of different backbones can be found in the following link: https://pytorch.org/docs/stable/torchvision/models.html. BN-Inception's top 1/top 5 error is 7.8%/25.2%, very similar to those of Densenet121 (7.8%/25.4%).

the networks reached state-of-the-art results even without any hyperparameter tuning, we expected a minimum amount of hyperparameters tuning to help. To this end, we used random search (Bergstra and Bengio 2012) to optimize the hyperparameters of our best network on the *CARS 196* dataset. We reach a 90.7 score (2*pp* higher score than the network with default hyperparameters) in Recall@1, and 77.6 score (3*pp* higher score than the network with default hyperparameters) in NMI metric, showing that individual hyperparameter optimization can boost the performance. The score of 90.7 in Recall@1 is not only by far the highest score ever achieved, but also the first time any method has broken the 90 point barrier in Recall@1 metric when evaluated on the *CARS 196* dataset.

Alternative loss formulation We have formulated the loss as an iterative dynamical system, followed by the cross-entropy loss function. In this way, we encourage the network to predict the same label for samples coming from the same class. One might argue that this is not necessarily the best loss for metric learning, in the end, we are interested in bringing similar samples closer together in the embedding space, without the need of having them classified correctly. Even though several works have shown that a classification loss can be used for metric learning (Movshovitz-Attias et al. 2017; Qian et al. 2019; Zhai and Wu 2019), we tested whether this is also the best formulation for our loss function.

We therefore experimented with a different loss function which encourages the network to produce similar label distributions (soft labels) for the samples coming from the same class. We first defined the Kullback-Leibler divergence for two distributions P and Q as:

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) log \frac{P(x)}{Q(x)}. \tag{4.8}$$

We then minimized the divergence between the predicted probability (after the iterative procedure) of samples coming from the same class. Unfortunately, this loss formulation resulted in lower performances on both *CUB-200-2011* (3*pp*) and *Cars 196* (1.5*pp*).

4.5 Conclusions and Future Work

In this work, we proposed the Group Loss, a new loss function for deep metric learning that goes beyond triplets. By considering the content of a mini-batch, it promotes embedding similarity across all samples of the same class, while enforcing dissimilarity for elements of different classes. This is achieved with a fully-differentiable layer that is used to train a convolutional network in an end-to-end fashion. We showed that our model outperforms state-of-the-art methods on several datasets, and at the same time shows fast convergence. In our work, we did not consider any advanced and intelligent sampling strategy. Instead, we randomly sampled objects from a few classes at each iteration. Sampling has shown to have a very important role in feature embedding (Manmatha et al. 2017), therefore, we will explore in future work sampling techniques which can be suitable for our module. Additionally, we are going to investigate the applicability of Group Loss to other problems, such as person re-identification, landmark matching and deep semi-supervised learning.

Acknowledgements. This research was partially funded by the Humboldt Foundation through the Sofja Kovalevskaja Award. We thank Michele Fenzi, Maxim Maximov and Guillem Braso Andilla for useful discussions.

CHAPTER 5

Relaxation Labeling Processes for Semantic Segmentation

Abstract In the following, we address the problem of semantic segmentation, by adapting two different but tightly related methodologies for learning the compatibilities of a Relaxation Labeling process. First, we introduce the concepts of context and neighborhood that make our learning framework simple and efficient. We then describe the methodology presented in (Pelillo and Refice 1994) to learn the compatibilities in a Relaxation Labeling process, showing that the learning algorithm is equivalent to a constrained version of the backpropagation method. We later introduce the second methodology, whose aim is to learn compatibilities consistent with the ground truth provided by a dataset. The method involves the solving of a system of linear inequalities, which is assessed through the Eremin method, explained below. Following this, we propose a convolutional ReLab interpretation to efficiently work with image-structured predictions, and exploit the inductive biases of such type of data. Finally, we present some preliminary experiments and qualitative visualizations on noise reduction and semantic segmentation tasks, adopting the two learning algorithms with the convolutional form. The experimental results proposed in this chapter are preliminary and represent a proof of concept of the validity of our method.

5.1 Introduction

In Semantic Segmentation task, it is not uncommon to refine the final probabilities coming from Deep Neural network architecture, with a refinement method, such as the DenseCRF (Krähenbühl and Koltun 2011). Historically, Relaxation Labeling was developed and applied to semantic segmentation problems (under different names, e.g. pixel labeling, pixel classification and image labeling). The ReLab process has been shown to be interpretable and this helps in understanding and tuning the process for specific semantic segmentation tasks. Furthermore, differently, from the Conditional Random Field, who aim to find a global minimum, ReLab searches for a local critical point around the (hopefully) good initial label assignment matrix. Motivated by these facts, in this chapter, we investigate the use of ReLab as a refiner of DNN for semantic segmentation.

The standard ReLab formulation assumes to have access to a set of compatibilities, or at least to a hand-crafted compatibility formula. Such solutions however may not be available or make too simplistic assumptions on the nature of the observations and their interactions. Furthermore, the compatibility matrix $\mathbf{R}$ contains n^2m^2 non-negative elements. This can be a major drawback when dealing with current real-world datasets, that typically have a large number of observations. In addition, due to the transductive nature of ReLab, the change of testing observation requires the computation of $\mathbf{R}$ again. This chapter propose to solve these problems in two ways: (a) through the learning of a set of compatibilities, based on a dataset of observations, (b) through the application of locality and parameter sharing approaches, already explained in the context of the convolution operator (cf. Section 1). In the following we address the need of such properties by introducing the concepts of **context** and **neighborhood**, and then explain how to cast our framework in a convolutive way to apply ReLab on image-like structures. Later we introduce two methodologies for automatically learning compatibilities, and adapt them to work for semantic-segmentation tasks.

In particular, the first methodology describes a learning framework to learn compatibilities in a heteroassociative way (Hertz, Krogh and Palmer 1991), from the prior knowledge generated by a dataset of labeled observations, through a loss minimization approach. We show that the learning method is analogous to

a constrained-backpropagation approach that can be applied in an end-to-end fashion with the learning of a DNN model parameter set. The second method instead learns compatibilities that are consistent according to the ground truth coming from a dataset, in an autoassociative way (Pelillo and Fanelli 1997), thus not requiring the input data. This approach directly enforces the consistency property by solving the system of linear inequalities associated to its definition, with the help of solving algorithm developed by (Eremin 1962).

5.2 Simplifying the compatibility matrix

First of all, we start our discussion by simplifying the interactions among the sets of objects involved in the ReLab process. Recall from the Introduction the ReLab framework as presented in (Pelillo 1997): in short, we have a set of objects $B = \{b_1, \ldots, b_n\}$, a set of labels $\Lambda = \{1, \ldots, m\}$, an assignment matrix $\mathbf{p} \in \Delta^{n \times m}$, and finally, a set of **compatibility coefficients**, called $\mathbf{R} \in \mathbb{R}_{\geq 0}^{nm \times nm}$, is defined to encode the pairwise compatibility between labeling hypotheses. (Pelillo and Refice 1994) assume a different setting, where the objects are grouped in subsets, that we will call **contexts**. Only the objects within each context interact together. Consider, for example, a dataset for NLP tasks, with sentences as contexts and single words as objects, or a dataset for semantic segmentation tasks, with images as contexts and pixels as objects. In such cases, we can further assume that each object in its context interacts with a further, smaller subset of neighboring objects, such as "previous/next words", or "4/8-connected pixels". We will call such subsets **neighborhoods**. Given these two definitions, we can assume that different contexts behave in the same way and share the same compatibilities among objects. The source of diversity is then demanded to the neighborhood relations between objects, thus resulting in a set of $m \times m$ submatrices, one for each neighborhood. The resulting compatibility matrix $\mathbf{R}$ has now size $|N| \times m \times m$, with N being the set of neighborhoods. The computation of the support in Equation (1.9) can be simplified to:

$$q_{i\lambda}^{(t)} = \sum_{\nu \in N} \sum_{\mu=1}^{m} r_{\nu\lambda\mu} p_{i+\nu,\mu}^{(t)} \tag{5.1}$$

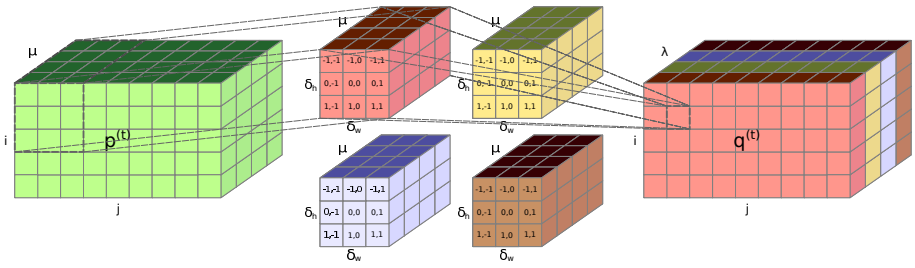


Figure 5.1: ReLab can be thought as a convolutional layer, where the input $\mathbf{p}^{(t)}$ is convolved by $\mathbf{R}$, to obtain $\mathbf{q}^{(t)}$, and then the update rule is applied to obtain $\mathbf{p}^{(t+1)}$.

Note that – from the previous discussion – the concept of neighborhood is somehow related to that of context. In principle, one can think of a context as a neighborhood where its objects are all interconnected between each other. The neighborhood, introduces however the weight sharing property, allowing for the definition of more general compatibilities and establishing a parallel with the convolution operator properties, which will be exploited in the next section. The new setting has the advantage to be learnable by training, as described in the Section 5.3 and 5.4.

Convolutional formulation

We now explain how to efficiently apply the relaxation labeling to image-like structures. For the sake of simplicity, let us suppose to have a 1-channel image of size $H \times W$. Each pixel x_{ij} has been assigned to a probability vector $\mathbf{p}_{ij}^{(0)}$ to be refined, according to the 8-connected neighbors of x_{ij} (other neighborhoods can be chosen as well). The difference with the previous formulation consists in the double indexing needed to select the pixels. Borrowing from the CNN theory, in the case of an image input, $\mathbf{p}^{(t)}$ can be thought as a set of m feature maps of size $H \times W$ which has to be convolved with $\mathbf{R}$. As shown in Figure 5.1, $\mathbf{R}$ is shaped as a $3 \times 3 \times m \times m$ convolutional kernel in order to have m feature maps in output, representing $\mathbf{q}^{(t)}$. The update rule can then be applied as a non-linear activation function to obtain $\mathbf{p}^{(t+1)}$. In our case,

8-connected pixel have fixed displacement, so:

$$q_{ij\lambda}^{(t)} = \sum_{\nu_h \in N_h} \sum_{\nu_w \in N_w} \sum_{\mu=1}^{m} r_{\nu_h \nu_w \lambda \mu} p_{i+\nu_h, j+\nu_w, \mu} \tag{5.2}$$

where $N_h = N_w = \{-1, 0, 1\}$ and a padding of size 1 is applied to $\mathbf{p}^{(t)}$. Note that this formulation requires the additional constraint $r_{\nu_h \nu_w \lambda \mu} = 0$ if $\nu_h = \nu_w = 0$ to nullify the support coming from x_{ij} itself.

5.3 Heteroassociative learning framework

(Pelillo and Refice 1994) addressed the learning problem by proposing a loss minimization framework, to efficiently learn compatibilities through the use of prior knowledge coming from a dataset of observations. The following description, assumes to work with a fully supervised dataset, and the extension to the semi-supervised case is straightforward. Let $L = \{L_1, , \ldots, L_\Gamma\}$ a dataset of contexts, where each sample L_γ is a set of labeled objects of the form $L_\gamma = \{(b_i^{(\gamma)}, \lambda_i^{(\gamma)}) \mid 1 \leq i \leq n, b_i^{(\gamma)} \in B, \lambda_i^{(\gamma)} \in \Lambda\}$ (for ease of notation we assume contexts with the same size). For readability concerns, in the following, the γ index will be omitted when not necessary. Each context is related to an unambiguous assignment matrix $\mathbf{y} \in \Delta^{n \times m} \cap \{0, 1\}^{n \times m}$, representing the one-hot encodings of the object labels. The learning task consists in finding the correct compatibility set to correctly update an initial assignment matrix until it converges to $\mathbf{y}$. The comparison between the predicted and the true assignment matrix can be estimated through the cross-entropy function:

$$E(\mathbf{p}, \mathbf{R}) = -\sum_{i=1}^{n} \sum_{\lambda=1}^{m} y_{i\lambda} \ln(p_{i\lambda}^{(F)}) = -\sum_{i=1}^{n} \ln(p_{i\lambda_i}^{(F)}) \tag{5.3}$$

A further loss can be devised as the average of the previous loss computed on a batch of multiple contexts, but this will not be considered in the discussion.

The learning algorithm is based on Rosen's gradient projection method (Rosen 1960), in order to preserve the non-negativity constraints on the compatibilities. We skip the derivation of the procedure, which can, however, be easily realized, and invite the reader to refer to (Pelillo and Refice 1994) for the pseudo-code of the method.

Gradient computation: equivalence with backpropagation

In (Pelillo and Refice 1994), it is described the procedure to compute the derivative of the cost function with respect to $r_{\tau\alpha\beta}$, $\forall\tau, \alpha, \beta$. This is accomplished in a bottom-up approach, computing the values of the derivatives using the chain rule, starting from $p_{i\lambda}^{(0)}$, $\forall i, \lambda$ until reaching $E(\mathbf{R})$. In the related literature, this approach is usually referred with the term **automatic differentiation** (Baydin et al. 2017). Automatic differentiation has progressively gained popularity as a derivative computation technique, thanks to its large use in the deep learning community, and nowadays has become a fundamental tool in the development of a deep learning framework. Historically, the differentiation of long expressions has been performed in two ways:

1. **Numerical differentiation:** the computation is performed with the method of **finite difference approximations**, $\partial f(x)/\partial x = (f(x+h) - f(x))/h$. In general, this method generates ill-conditioned and unstable derivatives.

2. **Symbolic differentiation:** the computation is performed by automatically manipulating the mathematical formula of $f(x)$, using the predefined differentiation rules. For complex functions, the method generates exponentially large symbolic expressions that are computationally intractable. In addition, the automatic simplification of such expressions, in general, is known to be an NP-hard problem, thus it does not solve the inherent inefficiency of this method (Ruijl et al. 2014).

Automatic differentiation takes the best of both worlds: the derivative is developed progressively by using the chain rule, taking small (atomic) steps. For each step, the exact value of the derivative is computed and used as input for the next step. By interleaving symbolic and numerical development of the derivative, the numerical error is more controllable. Moreover, automatic differentiation can easily cope with control flow commands, such as branching, loops, recursion, and procedure calls, an important technical advantage in operative frameworks.

Autodifferentiation can be applied in two different modes. The most intuitive one is called **forward/linear tangent mode** (Baydin et al. 2017). The computation starts from the differentiation variables, in our case $r_{\tau\alpha\beta}$, and for each of them,

f(x)	D[f(x)]	
$E = -\sum_{i=1}^{n} \ln(p_{i\lambda_i}^{(F)})$	$\frac{\partial E}{\partial p_{i,\lambda_i}} = -1/p_{i\lambda_i^{(F)}}$	
$p_{i\lambda}^{(t+1)} = \frac{h_{i\lambda}^{(t)}}{\sum_{\mu} h_{i\mu}^{(t)}}$	$\frac{\partial p_{i\lambda}^{(t+1)}}{\partial h_{i\eta}^{(t)}} = (\Phi(\lambda=\eta)\sum_{\mu} h_{i\mu}^{(t)} - h_{i\eta}^{(t)}) / \left(\sum_{\mu} h_{i\mu}^{(t)}\right)^2$	
$h_{i\lambda}^{(t)} = p_{i\lambda}^{(t)} q_{i\lambda}^{(t)}$	$\frac{\partial h_{i\lambda}^{(t)}}{\partial p_{i\lambda}^{(t)}} = q_{i\lambda}^{(t)}$	$\frac{\partial h_{i\lambda}^{(t)}}{\partial q_{i\lambda}^{(t)}} = p_{i\lambda}^{(t)}$
$q_{i\lambda}^{(t)} = \sum_{\nu\mu j} r_{\nu\lambda\mu} p_{j+\nu,\mu}^{(t)}$	$\frac{\partial q_{i\lambda}^{(t)}}{\partial r_{\tau\alpha\beta}} = \Phi(\lambda=\alpha) p_{j+\tau,\beta} \sum_{\nu\mu j} r_{\nu\lambda\mu} \frac{\partial p_{j+\nu,\mu}^{(t)}}{\partial r_{\tau\alpha\beta}}$	

Table 5.1: The derivatives of the ReLab process. Φ represents the indicator function.

their derivative tree is computed. This mode is inefficient for functions $f : \mathbb{R}^n \to \mathbb{R}$. Assuming that the derivative of f can be divided into the computation of d derivatives, this method would have complexity $O(nd)$. In fact, the forward mode is efficient when dealing with functions $f : \mathbb{R}^n \to \mathbb{R}^m$, with $n \ll m$.

On the contrary, the **reverse/adjoint/cotangent linear mode** takes a top-down approach by computing the derivative starting from f (the cost function, in our case) and going down to the differentiation variables. As said above, this method gained popularity in deep learning, and it is also known with the term **backpropagation**.

We therefore argue that, given the equivalence of results with both modes, the second approach is to be preferred. In Table 5.1, we report the derivatives of the ReLab process to be used in the reverse mode.

5.4 Autoassociative learning framework

Aside from the method described above, (Pelillo and Fanelli 1997) proposed a different approach to learn the compatibilities. The approach is guided directly by the ground truth of a dataset, in such a way that the learned compatibilities satisfy the consistency property of Equation (1.11). Recalling the definition, a labeling $\mathbf{p}$ is consistent if for all patterns L_γ, all objects $b_i^{(\gamma)}$ labeled with $\lambda_i^{(\gamma)}$ satisfy:

$$q_{i\lambda_i} \leq q_{i\lambda} \implies \sum_j r_{ij}(\lambda, \lambda_j^{(\gamma)}) - \sum_j r_{ij}(\lambda_i^{(\gamma)}, \lambda_j^{(\gamma)}) < 0 \tag{5.4}$$

Algorithm 4 Eremin inequality system solver

Require: A coefficient matrix $\mathbf{C}$, an initial starting point $\mathbf{R}_0$, a margin M
Ensure: The best solution $\mathbf{R}_F$

1: **while** not reaching convergence **do**
2: $\Delta(\mathbf{R}_k) = \max_{\gamma,i,\lambda} \left\{ \sum_\nu r_\nu(\lambda, \lambda_{i+\nu}^{(\gamma)}) - \sum_\nu r_\nu(\lambda_i^{(\gamma)}, \lambda_{i+\nu}^{(\gamma)}) - M \right\}$
3: $\gamma_k, i_k, \lambda_k = \arg\max_{\gamma,i,\lambda} \left\{ \sum_\nu r_\nu(\lambda, \lambda_{i+\nu}^{(\gamma)}) - \sum_\nu r_\nu(\lambda_i^{(\gamma)}, \lambda_{i+\nu}^{(\gamma)}) - M \right\}$
4: $d(\mathbf{R}_k) = max(0, d(\mathbf{R}))$
5: $\mathbf{R}_{k+1} = \mathbf{R}_k - \lambda_{k+1} d(\mathbf{R}) C(\gamma_k, i_k, \lambda_k; \cdot)^\top$

For all $\lambda \neq \lambda_i^{(\gamma)}$. When restricting the interactions to small neighborhoods, this turns to:

$$\sum_\nu r_\nu(\lambda, \lambda_{i+\nu}^{(\gamma)}) - \sum_\nu r_\nu(\lambda_i^{(\gamma)}, \lambda_{i+\nu}^{(\gamma)}) \leq 0 \tag{5.5}$$

Which is a system of $\Gamma * n * (m-1)$ inequalities in the $|N|m^2$ unknowns. Such a system is computationally intractable when dealing with real world datasets, however if the neighborhood chosen is small enough, one can expect that most inequalities are coincident, and the duplicates can be discarded. This is the case in our experiments with convolutional compatibility kernels.

The coefficient matrix $\mathbf{C}$ can be defined as follows (Pelillo and Fanelli 1997):

$$\mathbf{C}(\gamma, i, \lambda; j, k, \mu, \eta) = \begin{cases} +1 & \text{if } j = i, \mu = \lambda, \eta = \lambda_k^{(\gamma)} \\ -1 & \text{if } j = i, \mu = \lambda_j^{(\gamma)}, \eta = \lambda_k^{(\gamma)} \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

Where for notational convenience, we have highlighted the γ indexing the context, and we used a three-component index for the rows and a four-component index for the columns. The system can then be restated in the following compact way:

$$\mathbf{CR} \leq \mathbf{0} \tag{5.7}$$

To ensure larger basins of attraction, one can also introduce a negative margin M and set the system to be less or equal than $\mathbf{1}M$.

The Eremin algorithm (Eremin 1962) has been developed to solve system of inequalities, and when the system is not solvable, it searches for the best approximation

solution in the sense of Čebišev. Following (Pelillo and Fanelli 1997). In Algorithm 4, we report the Eremin method, adapted for the inequality system of Equation (5.5). $C($ The following theorem ensures the correctness of the algorithm:

Theorem 5.4.1 (Eremin, 1962)**.** *The sequence* $\mathbf{R}_k$ *defined in Algorithm 4 is solving for the system* (5.7)*.*

The algorithm does not take into account for the non-negative constraint to be imposed to the compatibilities, however this can be solved by adding additional inequalities or translating the final solution, which does not alter the space of consistent labelings.

5.5 Experimental Results

Noise Reduction

We first performed a simple experiment to assess the capability of ReLab as a noise reduction method. To this end, we generated a synthetic dataset from MNIST by thresholding each image to obtain a foreground mask representing the digit. The masked image was then perturbed by redistributing the 0/1 probability to the new values $\epsilon/1-\epsilon$. First the compatibilities of ReLab were learned with the heteroassociative method and then ReLab was run over the image to reconstruct the original mask. The perturbed pixels were chosen randomly, in a percentage ruled by a parameter ψ. The values for both parameters were chosen to be in the set $\{0.2, 0.4, 0.5, 0.6, 0.8, 1.0\}$. We tried 9 different instances of ReLab, by changing the kernel size and the number of iterations respectively with values in $\{3, 7, 13\}$ and $\{4, 16, 32\}$.

Table 5.2 shows the average accuracy for each pair of ϵ/ψ parameters. ReLab is able to recover the original image discretely well with $\epsilon \leq 0.8$. In particular, when $\epsilon = 0.8$ and $\psi \geq 0.8$, the accuracy decreases with respect to the other previous configurations. With $\epsilon = 1$ – the probabilities are totally inverted – the performance decrease is even more evident until reaching the limit case of a total pixel inversion corresponding to $\epsilon = \psi = 1$.

$\epsilon = 0.2$		$\epsilon = 0.4$		$\epsilon = 0.5$	
ψ	acc.	ψ	acc.	ψ	acc.
0.2	1	0.2	0.99	0.2	0.98
0.4	1	0.4	0.98	0.4	0.96
0.5	1	0.5	0.98	0.5	0.96
0.6	1	0.6	0.97	0.6	0.95
0.8	1	0.8	0.97	0.8	0.91
1	1	1	0.98	1	0.79
$\epsilon = 0.6$		$\epsilon = 0.8$		$\epsilon = 1$	
ψ	acc.	ψ	acc.	ψ	acc.
0.2	0.97	0.2	0.96	0.2	0.80
0.4	0.94	0.4	0.90	0.4	0.60
0.5	0.92	0.5	0.85	0.5	0.50
0.6	0.90	0.6	0.84	0.6	0.40
0.8	0.79	0.8	0.78	0.8	0.20
1	0.73	1	0.72	1	0

Table 5.2: Average results over the different images and hyperparameter configurations. As expected, with the increment of one of the two perturbation parameters the accuracy decreases as the reconstruction is more difficult.

Prediction refinement with the learning algorithms

Motivated by the previous experiments, we performed another series of experiments on semantic segmentation tasks to assess the performances of our convolutional heteroassociative method in the refinement of more complex pixel probability distributions. We took into consideration a classical Fully Convolutional Network (FCN), namely the **FCN8s At Once** architecture (Shelhamer, Long and Darrell 2017; Wada n.d.), and we appended a ReLab module at the end of the softmax layer. The architecture was then trained with and without our module to assess the differences in the accuracy. The dataset used is a standard benchmark, the PASCAL VOC 2012 dataset (Everingham et al. n.d.), which contains 6900 images ca. of different sizes, with approximately 1.3E09 pixels to be labeled. Table 5.5 shows the results of the baseline (net only) against the best configurations found so far. The measure of performance is the mean IOU (mIOU) computed as:

$$mIOU = \frac{TP}{FP + FN + TP} \tag{5.8}$$

With TP, FP, FN, being respectively the number of True Positive, False Positive, and False Negative. During the training, pixel accuracy is used instead. The results are

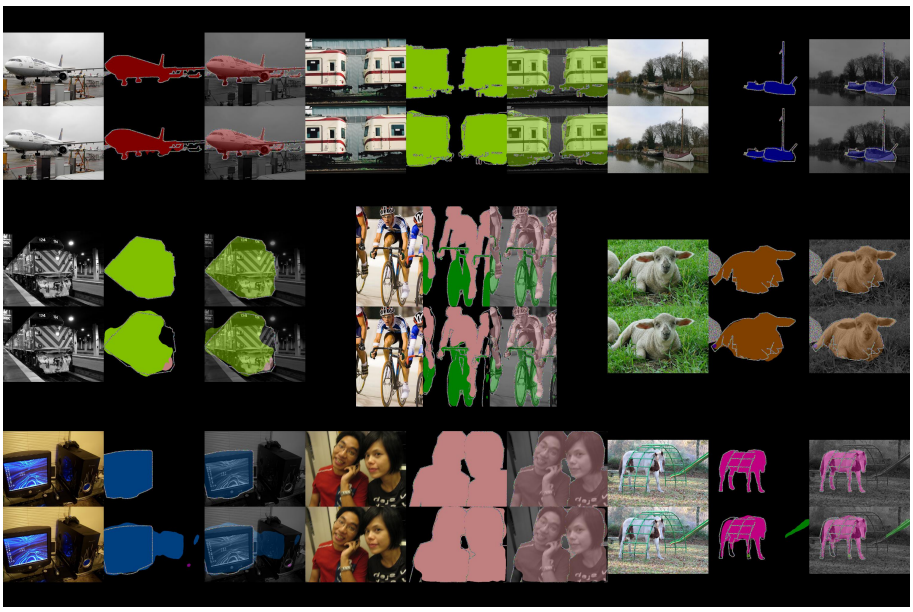


Figure 5.2: Visual results of the best FCN + ReLab model. For each example, we present the ground truth (top rows, second and third image) and the refined predictions (bottom rows second and third image) of the FCN + ReLab model.

mIoU	0.6401	0.6411	0.6418	0.6429	0.6464	**0.6478**
Iterations	1E5	2E6	1E5	1E5	2E6	**2E6**
FCN lr	1E-10	1E-05	6.5E-5	6E-5	1E-5	**1E-5**
Momentum	0.99	0.99	0.921	0.921	0.99	**0.99**
W. Decay	5E-4	5E-4	8.1E-4	8.1E-4	5E-4	**5E-4**
ReLab lr	-	0.1	0.3	0.26	0.1	**0.1**
ReLab k. size	-	11	5	5	5	**5**
ReLab iter.	-	5	7	7	5	**15**

Table 5.3: Results and configuration on Pascal VOC 2012. The first column represents the baseline configuration – FCN8s at once without ReLab – while the other columns show the top five results of the FCN + ReLab architecture. The best configuration is highlighted in bold.

promising, showing that the architecture learns, and a slight improvement is testified. However, we expect to reach better results by changing the overall architecture FCN + ReLab, for example, taking into account two cost functions instead of only one at the end of the ReLab module.

In addition to this, we assessed the performance of another refinement procedure by learning offline the compatibilities with the autoassociative learning framework.

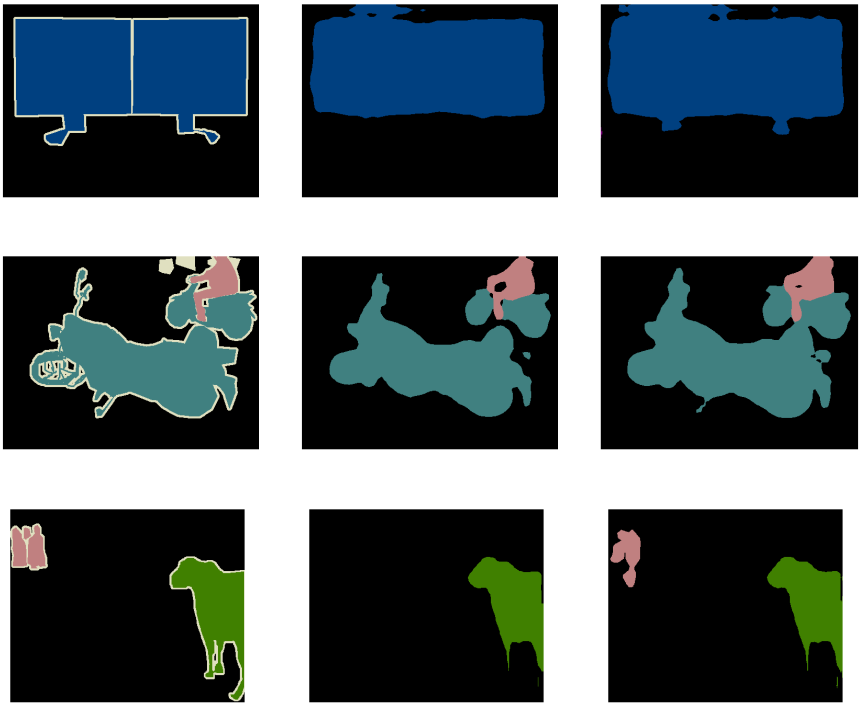


Figure 5.3: Some visualizations of the ReLab refinement with the compatibilities learned with the Eremin method. The left column represents the ground truth, the middle one represents the DeepLabv3 predictions, the right one represents the ReLab refinements with dilation set to 1. The refinement is able to recover the monitor stands, the motorbike pedal and light, and the people, in the three cases, showing that ReLab is effectively able to correct the assignment errors introduced by DeepLabv3.

The experiment consisted in refining the predictions of a DNN model with ReLab, and comparing the plain results against the refined ones. The compatibilities were learned using the ground truth of PascalVOC. Note again, that this learning method does not necessitate of the image input, differently from the heteroassociative learning method previously described. The neighborhoods used, consisted in kernels of size 3 with an increasing dilation parameter, ranging from 1 to 10. The maximum number of iterations was set to $1E6$ and the tolerance to $1E-3$, since some preliminary experiment showed a far slower convergence with higher tolerances. The initial compatibilities were set to zero, and the margin was set to -1, to avoid the solver to stop immediately. After the learning, a translation to avoid negative values was

Figure 5.4: Some visualizations of the ReLab refinement with the compatibilities learned with the Eremin method. The first and second images show the ground truth and the DeepLabv3 predictions, while the last two images show the refinements with dilation respectively set to 1 and to 9. The airplane is better refined when the dilation parameter is larger.

kernel size	dilation	mIOU	w. mIOU
3	1	0.7992	0.8469
3	2	0.8002	0.8492
3	3	0.8020	0.8516
3	4	0.7947	0.8446
3	5	0.8017	0.8517
3	6	0.7957	0.8466
3	7	0.7953	0.8455
3	8	0.7974	0.8475
3	9	0.7961	0.8465
3	10	0.7929	0.8447

Table 5.4: Results and configuration on Pascal VOC 2012, with the compatibilities learned with the Eremin method. The third column displays the Intersection over Union averaged over the dataset. The fourth column displays the mIOU weighted by the class size, to mitigate class imbalancing. The DeepLabv3 mIOU and w. mIOU are respectively 0.8034 and 0.8537. While our refinement procedure does not provide any substantial improvement, the results are comparable with those of DeepLabv3. Furthermore, the qualitative comparison attests that our method can indeed improve the refinement of the segmentations.

performed and the compatibilities between the background class and itself were set to zero, to avoid the self-reinforcement of the background class, which is highly imbalanced with respect to the others. The DNN tested was the DeepLabv3 model (Chen et al. 2017), in particular the trained implementation of the PyTorch library.

Table 5.5 shows that, at this stage, the refined probabilities generate comparable results with respect to the unrefined ones, but do not consistently improve the performance of the model yet. A qualitative comparison of some examples, however, highlights the benefits of such refinement. Figure 5.3, shows that ReLab effectively

recovers the small probability assignment related to the correct class and "boost" them, improving the overall quality of the segmentation. In particular, we testified that different dilation values, have different recovery quality. For example, Figure 5.4 shows that a kernel with dilation value of 9 can recover different images than a kernel with dilation set to 1. This suggests that in the long term a **holistic approach** involving refinements coming from different kernels, can bring overall to better performances.

5.6 Conclusions and Future Work

In this work, we proposed a convolutional way to perform ReLab and applied it to two experimental cases, showing that ReLab can improve the accuracy of a simple Semantic Segmentation model. We have described a learning framework for ReLab, and we adapted it to work in an end-to-end fashion with DNNs, by devising the derivatives for backpropagation. At the same time, we tried with a different learning approach, to generate compatibilities consistent according to the ground truth of a given dataset. Both the learning methods showed promising results and are worth to be further explored.

The heteroassociative method of (Pelillo and Refice 1994) can be analysed in the context of multiple ReLab processes that encode and the decode the matrix of probabilities, by effectively proposing a convolutional framework for random fields.

The autoassociative method of (Pelillo and Fanelli 1997) can be improved by increasing the kernel size (experiments have not been performed yet, because they require a careful implementation for keeping the method efficient), and devising – as said above – a holistic approach that summarizes more refinements together. Finally, this work will be extended to the semi- and weakly supervised settings.

Conclusions

The work presented in this thesis is an attempt to study, under multiple viewpoints, the use of the contextual information. We have shown that the contextual information, thanks to its general nature, can help in improving the performance of neural models in tasks such as semisupervised learning, domain adaptation, metric learning and semantic segmentation. The potentiality of contextual information in the learning of a model, does not stop with these tasks. As mentioned in the preface, we strongly believe that the contextual information should be thought as an opportunity to move further from the perspective of feature learning, which has been dominating the discussion in the machine learning field over the last decade.

GTG has been already proved to be a powerful prediction refiner in the context of semisupervised learning, under extreme label scarcity conditions (Elezi et al. 2018). Its application has been proven to be advantageous in domain adpatation tasks too, with only small changes to the core algorithm (Vascon et al. 2019). These results assess the efficiency of GTG under different conditions of the input data, thus its adaptability to a variety of different cases. The main drawback of this method is its transductive nature that requires to compute a large affinity matrix, often intractable in the case of real world datasets. A new direction has been proposed by devising an end-to-end method for deep metric learning (Elezi et al. 2020) that exploits the GTG update rule in the context of a mini-batch which is a representative of the dataset in its entirety, according to the same assumption that motivates the use of SGD in DNN parameter learning. This new perspective is worth to be applied in the weakly-supervised context, as the author of this thesis is currently doing. At the same time the end-to-end learning framework poses new challenges. Propagating the

information to very distant observations, can require more iterations of the process, which, being an autonomous dynamical system, may suffer from vanishing gradient problems (Bengio, Simard and Frasconi 1994). In addition, the similarity formula is still dependent from an hand-crafted formula, along with the choice of a sparsification technique. As a future work we would like to further exploits the learning capabilities of the neural models, and make one more step toward the automatization of the decision making process related to the similarity and sparsification methods.

This last concern has been partially addressed in the more general framework of Relaxation Labeling. In the last chapter of this thesis, we showed different learning techniques for compatibilities. While in one of them we were able to prove its (almost total) equivalence with the backpropagation method, in the other one we were even able to guide the learning process with the fundamental notion of consistency. This, paired with a convolutional adpatation of the learning framework for image-like structures, allowed us to apply ReLab in semantic segmentation tasks. As already noted, such efficient ReLab frameworks represent an opportunity to devise Deep Neural models in the context of random fields. Differently from the DNN feature extractors, that have been historically guided more by experimental evaluations, ReLab has a strong, theoretically motivated background with nice optimization properties under rather simple conditions on the compatibility coefficients. We argue that this is the most important feature of such algorithm, and a stronger comparison between DNNs and ReLab, under a theoretical point of view, may open to new theoretical results in the former case.

Aside from these aspects, considering, in its entirety, the contextual information framework presented in this thesis, we propose two further lines of research. The first one, already described in the Introduction, regards the application of such techniques in contexts with more generic inductive biases, such as in graph neural networks. The second one, represents the use of the average local consistency term for enforcing the consistency properties of the output predictions of loss-based optimization algorithms.

Bibliography

Alemu, L. T., Pelillo, M. and Shah, M. (2019). 'Deep Constrained Dominant Sets for Person Re-identification'. In: **CoRR** vol. abs/1904.11397.

Alex Krizhevsky, G. H. (2009). **Learning multiple layers of features from tiny images**. Tech. rep. University of Toronto.

Aphex34 (2015). **A typical cnn**.

Aslan, S., Vascon, S. and Pelillo, M. (2018). 'Ancient Coin Classification Using Graph Transduction Games'. In: **The 4th IEEE International Conference on Metrology and Archeology**.

Bacciu, D. et al. (2020). 'A gentle introduction to deep learning for graphs'. In: **Neural Networks** vol. 129, pp. 203–221.

Baum, L. E. and Sell, G. (1968). 'Growth transformations for functions on manifolds'. In: **Pacific Journal of Mathematics** vol. 27, no. 2, pp. 211–227.

Bay, H., Tuytelaars, T. and Van Gool, L. (2006). 'SURF: Speeded Up Robust Features'. In: **Computer Vision – ECCV 2006**. Ed. by Leonardis, A., Bischof, H. and Pinz, A. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 404–417.

Baydin, A. G. et al. (2017). 'Automatic differentiation in machine learning: a survey'. In: **The Journal of Machine Learning Research** vol. 18, no. 1, pp. 5595–5637.

Bengio, Y., Simard, P. Y. and Frasconi, P. (1994). 'Learning long-term dependencies with gradient descent is difficult'. In: **IEEE Trans. Neural Networks** vol. 5, no. 2, pp. 157–166.

Bergstra, J. and Bengio, Y. (2012). 'Random Search for Hyper-Parameter Optimization'. In: **Journal of Machine Learning Research** vol. 13, pp. 281–305.

Berthelot, D. et al. (2019). 'MixMatch: A Holistic Approach to Semi-Supervised Learning'. In: **Advances in Neural Information Processing Systems, NeurIPS**, pp. 5050–5060.

Bromley, J. et al. (1994). 'Signature verification using a" siamese" time delay neural network'. In: **Advances in Neural Information Processing Systems, NIPS**, pp. 737–744.

Çakir, F. et al. (2019). 'Deep Metric Learning to Rank'. In: **IEEE Conference on Computer Vision and Pattern Recognition, CVPR**, pp. 1861–1870.

Chen, L. et al. (2017). 'Rethinking Atrous Convolution for Semantic Image Segmentation'. In: **CoRR** vol. abs/1706.05587. arXiv: 1706.05587.

Chopra, S., Hadsell, R. and LeCun, Y. (2005). 'Learning a Similarity Metric Discriminatively, with Application to Face Verification'. In: **IEEE Computer Vision and Pattern Recognition, CVPR**, pp. 539–546.

Cinà, A. E., Torcinovich, A. and Pelillo, M. (2020). 'A Black-box Adversarial Attack for Poisoning Clustering'. In: **CoRR** vol. abs/2009.05474.

Ciresan, D. C., Meier, U. and Schmidhuber, J. (2012). 'Multi-column deep neural networks for image classification'. In: **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, pp. 3642–3649.

Cortes, C. and Vapnik, V. (1995). 'Support-Vector Networks'. In: **Machine Learning** vol. 20, pp. 273–297.

Deng, J. et al. (2009). 'ImageNet: A large-scale hierarchical image database'. In: **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, pp. 248–255.

Duan, Y. et al. (2019). 'Deep Embedding Learning with Discriminative Sampling Policy'. In: **IEEE Computer Vision and Pattern Recognition, CVPR**.

Elezi, I. et al. (2018). 'Transductive Label Augmentation for Improved Deep Network Learning'. In: **24th International Conference on Pattern Recognition, ICPR 2018, Beijing, China, August 20-24, 2018**. IEEE Computer Society, pp. 1432–1437.

Elezi, I. et al. (2020). 'The Group Loss for Deep Metric Learning'. In: **Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part VII**. Vol. 12352. Lecture Notes in Computer Science. Springer, pp. 277–294.

Elman, J. L. (1990). 'Finding structure in time'. In: **Cognitive science** vol. 14, no. 2, pp. 179–211.

Erdem, A. and Pelillo, M. (2012). 'Graph Transduction as a Noncooperative Game'. In: **Neural Computation** vol. 24, no. 3, pp. 700–723.

Eremin, I. I. (1962). 'Iteration Method for Čebyšev Approximations For Sets Of Incompatible Linear Inequalities'. In: **Soviet Math. Doklady** vol. 3, pp. 570–572.

Everingham, M. et al. (n.d.). **The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results**. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

Fernando, B. et al. (2013). 'Unsupervised Visual Domain Adaptation Using Subspace Alignment'. In: **Proceedings of the 2013 IEEE International Conference on Computer Vision**. ICCV '13. Washington, DC, USA: IEEE Computer Society, pp. 2960–2967.

Fiorucci, M. et al. (2017). 'On the Interplay Between Strong Regularity and Graph Densification'. In: **Graph-Based Representations in Pattern Recognition - 11th IAPR-TC-15 International Workshop, GbRPR 2017, Anacapri, Italy, May 16-18, 2017, Proceedings**. Ed. by Foggia, P., Liu, C. and Vento, M. Vol. 10310. Lecture Notes in Computer Science, pp. 165–174.

Fukushima, K. and Miyake, S. (1982). 'Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position'. In: **Pattern Recognition** vol. 15, no. 6, pp. 455–469.

Ganin, Y. and Lempitsky, V. (2015). 'Unsupervised domain adaptation by backpropagation'. In: **International Conference on Machine Learning**, pp. 1180–1189.

Ganin, Y. et al. (2016). 'Domain-adversarial training of neural networks'. In: **Journal of Machine Learning Research** vol. 17, no. 59, pp. 1–35.

Ge, W. et al. (2018). 'Deep Metric Learning with Hierarchical Triplet Loss'. In: **European Conference in Computer Vision, ECCV**, pp. 272–288.

Gong, B. et al. (2012). 'Geodesic flow kernel for unsupervised domain adaptation'. In: **Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on**. IEEE, pp. 2066–2073.

Goodfellow, I. J., Bengio, Y. and Courville, A. C. (2016). **Deep Learning**. Adaptive computation and machine learning. MIT Press.

Griffin, G., Holub, A. and Perona, P. (2007). **Caltech-256 object category dataset**. Tech. rep. California Institute of Technology.

Guo, C. et al. (2017). 'On Calibration of Modern Neural Networks'. In: **International Conference on Machine Learning, ICML**. Ed. by Precup, D. and Teh, Y. W.

Hahnloser, R. H. et al. (2000). 'Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit'. In: **Nature** vol. 405, no. 6789, pp. 947–951.

Han, J. and Moraga, C. (1995). 'The Influence of the Sigmoid Function Parameters on the Speed of Backpropagation Learning'. In: **From Natural to Artificial Neural Computation, International Workshop on Artificial Neural Networks, IWANN '95, Malaga-Torremolinos, Spain, June 7-9, 1995, Proceedings**. Ed. by Mira, J. and Hernández, F. S. Vol. 930. Lecture Notes in Computer Science. Springer, pp. 195–201.

He, K. et al. (2016). 'Deep residual learning for image recognition'. In: **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, pp. 770–778.

He, K., Lu, Y. and Sclaroff, S. (2018). 'Local Descriptors Optimized for Average Precision'. In: **IEEE Conference on Computer Vision and Pattern Recognition, CVPR**, pp. 596–605.

He, K. et al. (2018). 'Hashing as Tie-Aware Learning to Rank'. In: **IEEE Conference on Computer Vision and Pattern Recognition, CVPR**, pp. 4023–4032.

Hertz, J., Krogh, A. and Palmer, R. G. (1991). **Introduction to the theory of neural computation. Redwood City**.

Hochreiter, S. (1991). 'Untersuchungen zu dynamischen neuronalen Netzen'. In: **Diploma, Technische Universität München** vol. 91, no. 1.

Hochreiter, S. et al. (2001). **Gradient flow in recurrent nets: the difficulty of learning long-term dependencies**.

Hochreiter, S. and Schmidhuber, J. (1997). 'Long short-term memory'. In: **Neural computation** vol. 9, no. 8, pp. 1735–1780.

Huang, G. et al. (2017). 'Densely Connected Convolutional Networks'. In: **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, pp. 2261–2269.

Hubel, D. H. and Wiesel, T. N. (1959). 'Receptive fields of single neurones in the cat's striate cortex'. In: **The Journal of physiology** vol. 148, no. 3, p. 574.

Hummel, R. A. and Zucker, S. W. (1983). 'On the Foundations of Relaxation Labeling Processes'. In: **IEEE Trans. Pattern Anal. Mach. Intell.** vol. 5, no. 3, pp. 267–287.

Häusser, P., Mordvintsev, A. and Cremers, D. (2017). 'Learning by Association - A Versatile Semi-Supervised Training Method for Neural Networks'. In: **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, pp. 626–635.

Häusser, P. et al. (2017). 'Associative Domain Adaptation'. In: **IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017**, pp. 2784–2792.

Ioffe, S. and Szegedy, C. (2015). 'Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift'. In: **International Conference on Machine Learning, ICML**, pp. 448–456.

Ivakhnenko, A. G. and Lapa, V. G. (1966). **Cybernetic predicting devices**. Tech. rep. PURDUE UNIV LAFAYETTE IND SCHOOL OF ELECTRICAL ENGINEERING.

Jégou, H., Douze, M. and Schmid, C. (2011). 'Product Quantization for Nearest Neighbor Search'. In: **IEEE Trans. Pattern Anal. Mach. Intell.** vol. 33, no. 1, pp. 117–128.

Jordan, M. (1986). **Serial order: a parallel distributed processing approach. Technical report, June 1985-March 1986**. Tech. rep. California Univ., San Diego, La Jolla (USA). Inst. for Cognitive Science.

Kadar, I. and Ben-Shahar, O. (2014). 'SceneNet: A perceptual ontology for scene understanding'. In: **European Conference on Computer Vision (ECCV)**. Springer, pp. 385–400.

Keskar, N. S. et al. (2017). 'On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima'. In: **International Conference on Learning Representations, ICLR**.

Kim, W. et al. (2018). 'Attention-Based Ensemble for Deep Metric Learning'. In: **European Conference on Computer Vision**, pp. 760–777.

Kingma, D. P. and Ba, J. (2014a). 'Adam: A Method for Stochastic Optimization'. In: **International Conference on Learning Representations (ICLR)**.

Kingma, D. P. et al. (2014). 'Semi-supervised Learning with Deep Generative Models'. In: **Advances in Neural Information Processing Systems (NIPS)**, pp. 3581–3589.

Kingma, D. and Ba, J. (2014b). 'Adam: A Method for Stochastic Optimization'. In: **Proceedings of the 3rd International Conference on Learning Representations ICLR**.

Kouw, W. M. et al. (2016). 'Feature-Level Domain Adaptation'. In: **Journal of Machine Learning Research** vol. 17, no. 171, pp. 1–32.

Krause, J. et al. (2013). '3D Object Representations for Fine-Grained Categorization'. In: **International IEEE Workshop on 3D Representation and Recognition (3dRR-13)**. Sydney, Australia.

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012). 'ImageNet Classification with Deep Convolutional Neural Networks'. In: **Advances in Neural Information Processing Systems**, pp. 1106–1114.

Krähenbühl, P. and Koltun, V. (2011). 'Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials'. In: **Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain**. Ed. by Shawe-Taylor, J. et al., pp. 109–117.

Laine, S. and Aila, T. (2017). 'Temporal Ensembling for Semi-Supervised Learning'. In: **5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings**. OpenReview.net.

Law, M. T., Urtasun, R. and Zemel, R. S. (2017). 'Deep Spectral Clustering Learning'. In: **Proceedings of the 34th International Conference on Machine Learning, ICML**, pp. 1985–1994.

LeCun, Y. et al. (1989a). 'Generalization and network design strategies'. In: **Connectionism in perspective** vol. 19, pp. 143–155.

LeCun, Y., Bengio, Y. and Hinton, G. E. (2015). 'Deep learning'. In: **Nature** vol. 521, no. 7553, pp. 436–444.

LeCun, Y. and Cortes, C. (2010). 'MNIST handwritten digit database'. In:

LeCun, Y. et al. (1989b). 'Backpropagation Applied to Handwritten Zip Code Recognition'. In: **Neural Computation** vol. 1, no. 4, pp. 541–551.

Lee, D.-h. (2013). 'Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks'. In: **Workshop on Challenges in Representation Learning (ICML)**. Vol. 2, p. 3.

Linnainmaa, S. (1970). 'The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors'. In: **Master's Thesis (in Finnish), Univ. Helsinki**, pp. 6–7.

Long, M., Wang, J. and Jordan, M. I. (2016a). 'Deep transfer learning with joint adaptation networks'. In: **arXiv preprint arXiv:1605.06636**.

— (2016b). 'Unsupervised Domain Adaptation with Residual Transfer Networks'. In: **arXiv preprint arXiv:1602.04433**.

Long, M. et al. (2015). 'Learning Transferable Features with Deep Adaptation Networks'. In: **Proceedings of The 32nd International Conference on Machine Learning**, pp. 97–105.

Lowe, D. G. (2004). 'Distinctive image features from scale-invariant keypoints'. In: **International Journal of Computer Vision** vol. 60, no. 2, pp. 91–110.

Lu, Z. et al. (2017). 'The expressive power of neural networks: A view from the width'. In: **Advances in neural information processing systems**, pp. 6231–6239.

Luxburg, U. von (2007). 'A tutorial on spectral clustering'. In: **Statistics and Computing**.

MacQueen, J. (1967). 'Some methods for classification and analysis of multivariate observations'. In: **Proc. Fifth Berkeley Symp. on Math. Statist. and Prob., Vol. 1**, pp. 281–297.

Manmatha, R. et al. (2017). 'Sampling Matters in Deep Embedding Learning'. In: **IEEE International Conference on Computer Vision, ICCV**, pp. 2859–2867.

Maynard Smith, J. (1982). **Evolution and the Theory of Games**. Cambridge University Press.

McCulloch, W. S. and Pitts, W. H. (1990). 'A Logical Calculus of the Ideas Immanent in Nervous Activity'. In: **The Philosophy of Artificial Intelligence**. Ed. by Boden, M. A. Oxford readings in philosophy. Oxford University Press, pp. 22–39.

McDaid, A. F., Greene, D. and Hurley, N. J. (2011). 'Normalized Mutual Information to evaluate overlapping community finding algorithms'. In: **CoRR** vol. abs/1110.2515.

Miller, D. A. and Zucker, S. W. (1991). 'Copositive-plus Lemke algorithm solves polymatrix games'. In: **Operations Research Letters** vol. 10, no. 5, pp. 285–290.

Minsky, M. and Papert, S. (1987). **Perceptrons - an introduction to computational geometry**. MIT Press.

Miyato, T. et al. (2019). 'Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning'. In: **IEEE Trans. Pattern Anal. Mach. Intell.** vol. 41, no. 8, pp. 1979–1993.

Movshovitz-Attias, Y. et al. (2017). 'No Fuss Distance Metric Learning Using Proxies'. In: **IEEE International Conference on Computer Vision, ICCV**, pp. 360–368.

Maaten, L. van der and Hinton, G. E. (2012). 'Visualizing non-metric similarities in multiple maps'. In: **Machine Learning** vol. 87, no. 1, pp. 33–55.

Oliver, A. et al. (2018). 'Realistic Evaluation of Semi-Supervised Learning Algorithms'. In: **6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings**. OpenReview.net.

Opitz, M. et al. (2017). 'BIER - Boosting Independent Embeddings Robustly'. In: **IEEE International Conference on Computer Vision, ICCV**, pp. 5199–5208.

— (2018). 'Deep Metric Learning with BIER: Boosting Independent Embeddings Robustly'. In: **CoRR** vol. abs/1801.04815.

Park, W. et al. (2019). 'Relational Knowledge Distillation'. In: **IEEE Computer Vision and Pattern Recognition, CVPR**.

Paszke, A. et al. (2017). 'Automatic differentiation in pytorch'. In: **NIPS Workshops**.

Pearson, K. (1895). 'Notes on regression and inheritance in the case of two parents'. In: **Proceedings of the Royal Society of London** vol. 58, pp. 240–242.

Pelillo, M. (1997). 'The Dynamics of Nonlinear Relaxation Labeling Processes'. In: **Journal of Mathematical Imaging and Vision** vol. 7, no. 4, pp. 309–323.

Pelillo, M. and Fanelli, A. M. (1997). 'Autoassociative learning in relaxation labeling networks'. In: **Pattern Recognition Letters** vol. 18, no. 1, pp. 3–12.

Pelillo, M. and Refice, M. (1994). 'Learning Compatibility Coefficients for Relaxation Labeling Processes'. In: **IEEE Trans. Pattern Anal. Mach. Intell.** vol. 16, no. 9, pp. 933–945.

Pinkus, A. (1999). 'Approximation theory of the MLP model in neural networks'. In: **Acta numerica** vol. 8, no. 1, pp. 143–195.

Qian, Q. et al. (2019). 'SoftTriple Loss: Deep Metric Learning Without Triplet Sampling'. In: **2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019**. IEEE, pp. 6449–6457.

Quattoni, A. and Torralba, A. (2009). 'Recognizing indoor scenes'. In: **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, pp. 413–420.

Revaud, J. et al. (2019). 'Learning with Average Precision: Training Image Retrieval with a Listwise Loss'. In: **CoRR** vol. abs/1906.07589.

Rosen, J. B. (1960). 'The gradient projection method for nonlinear programming. Part I. Linear constraints'. In: **Journal of the society for industrial and applied mathematics** vol. 8, no. 1, pp. 181–217.

Rosenblatt, F. (1961). **Principles of neurodynamics. perceptrons and the theory of brain mechanisms**. Tech. rep. Cornell Aeronautical Lab Inc Buffalo NY.

Rosenfeld, A., Hummel, R. A. and Zucker, S. W. (1976). 'Scene Labeling by Relaxation Operations'. In: **IEEE Trans. Syst. Man Cybern.** vol. 6, no. 6, pp. 420–433.

Ruijl, B. et al. (2014). 'Why local search excels in expression simplification'. In: **arXiv preprint arXiv:1409.5223**.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986). 'Learning representations by back-propagating errors'. In: **nature** vol. 323, no. 6088, pp. 533–536.

Russakovsky, O. et al. (2014). 'ImageNet Large Scale Visual Recognition Challenge'. In: **CoRR** vol. abs/1409.0575.

Saenko, K. et al. (2010). 'Adapting visual category models to new domains'. In: **European conference on computer vision**. Springer, pp. 213–226.

Sanakoyeu, A. et al. (2019). 'Divide and Conquer the Embedding Space for Metric Learning'. In: **IEEE Computer Vision and Pattern Recognition, CVPR**.

Schmidhuber, J. (2015). 'Deep learning in neural networks: An overview'. In: **Neural Networks** vol. 61, pp. 85–117.

Schroff, F., Kalenichenko, D. and Philbin, J. (2015). 'FaceNet: A unified embedding for face recognition and clustering'. In: **IEEE Conference on Computer Vision and Pattern Recognition, CVPR**, pp. 815–823.

Schultz, M. and Joachims, T. (2003). 'Learning a Distance Metric from Relative Comparisons'. In: **Advances in Neural Information Processing Systems, NIPS**, pp. 41–48.

Sener, O. et al. (2016). 'Learning transferrable representations for unsupervised domain adaptation'. In: **Advances in Neural Information Processing Systems**, pp. 2110–2118.

Shelhamer, E., Long, J. and Darrell, T. (2017). 'Fully Convolutional Networks for Semantic Segmentation'. In: **IEEE Trans. Pattern Anal. Mach. Intell.** vol. 39, no. 4, pp. 640–651.

Sohn, K. (2016). 'Improved Deep Metric Learning with Multi-class N-pair Loss Objective'. In: **Advances in Neural Information Processing Systems, NIPS**, pp. 1849–1857.

Song, H. O. et al. (2016). 'Deep Metric Learning via Lifted Structured Feature Embedding'. In: **IEEE Conference on Computer Vision and Pattern Recognition, CVPR**, pp. 4004–4012.

Song, H. O. et al. (2017). 'Deep Metric Learning via Facility Location'. In: **IEEE Conference on Computer Vision and Pattern Recognition, CVPR**, pp. 2206–2214.

Sun, B., Feng, J. and Saenko, K. (2016). 'Return of Frustratingly Easy Domain Adaptation'. In: **Thirtieth AAAI Conference on Artificial Intelligence**.

Szegedy, C. et al. (2015). 'Going deeper with convolutions'. In: **IEEE Conference on Computer Vision and Pattern Recognition, CVPR**, pp. 1–9.

Tarvainen, A. and Valpola, H. (2017). 'Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results'. In: **5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings**. OpenReview.net.

Torcinovich, A. et al. (2017). 'A Computer Vision System for Monitoring Ice-Cream Freezers'. In: **Image Analysis and Processing - ICIAP 2017 - 19th International Conference, Catania, Italy, September 11-15, 2017, Proceedings, Part II**. Ed. by Battiato, S. et al. Vol. 10485. Lecture Notes in Computer Science. Springer, pp. 333–342.

Tripodi, R., Vascon, S. and Pelillo, M. (2016). 'Context aware nonnegative matrix factorization clustering'. In: **International Conference on Pattern Recognition, (ICPR)**, pp. 1719–1724.

Tzeng, E. et al. (2014). 'Deep domain confusion: Maximizing for domain invariance'. In: **arXiv preprint arXiv:1412.3474**.

Vapnik, V. (1998). **Statistical Learning Theory**. New York: Wiley.

Vascon, S. et al. (2018). 'Protein function prediction as a graph-transduction game'. In: **Pattern Recognition Letters**.

Vascon, S. et al. (2019). 'Unsupervised Domain Adaptation using Graph Transduction Games'. In: **International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019**. IEEE, pp. 1–8.

Vaswani, A. et al. (2017). 'Attention is All you Need'. In: **Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA**. Ed. by Guyon, I. et al., pp. 5998–6008.

Wada, K. (n.d.). **pytorch-fcn, PyTorch Implementation of Fully Convolutional Networks**. https://github.com/wkentaro/pytorch-fcn.

Wagstaff, E. et al. (2019). 'On the Limitations of Representing Functions on Sets'. In: **Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA**. Ed. by Chaudhuri, K. and Salakhutdinov, R. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 6487–6494.

Wah, C. et al. (2011). **The Caltech-UCSD Birds-200-2011 Dataset**. Tech. rep. CNS-TR-2011-001. California Institute of Technology.

Wang, J. et al. (2017). 'Deep Metric Learning with Angular Loss'. In: **IEEE International Conference on Computer Vision, ICCV**, pp. 2612–2620.

Wang, X. et al. (2019a). 'Ranked List Loss for Deep Metric Learning'. In: **IEEE Conference on Computer Vision and Pattern Recognition, CVPR**, pp. 5207–5216.

Wang, X. et al. (2019b). 'Multi-Similarity Loss with General Pair Weighting for Deep Metric Learning'. In: **IEEE Computer Vision and Pattern Recognition, CVPR**.

Weibull, J. (1997). **Evolutionary Game Theory**. MIT Press.

Weinberger, K. Q. and Saul, L. K. (2009). 'Distance Metric Learning for Large Margin Nearest Neighbor Classification'. In: **Journal of Machine Learning Research** vol. 10, pp. 207–244.

Weisstein, E. W. (n.d.). 'Hyperbolic Tangent'. In: **https://mathworld.wolfram.com/HyperbolicTangent.html** ().

Werbos, P. J. (1982). 'Applications of advances in nonlinear sensitivity analysis'. In: **System modeling and optimization**. Springer, pp. 762–770.

Xu, X. et al. (n.d.). 'Deep Asymmetric Metric Learning via Rich Relationship Mining'. In: **IEEE Computer Vision and Pattern Recognition, CVPR**.

Xuan, H., Souvenir, R. and Pless, R. (2018). 'Deep Randomized Ensembles for Metric Learning'. In: **European Conference Computer Vision, ECCV**, pp. 751–762.

Yosinski, J. et al. (2014). 'How transferable are features in deep neural networks?' In: **Advances in Neural Information Processing Systems (NIPS)**, pp. 3320–3328.

Yu, B. et al. (2018). 'Correcting the Triplet Selection Bias for Triplet Loss'. In: **European Conference in Computer Vision, ECCV**, pp. 71–86.

Yuan, Y., Yang, K. and Zhang, C. (2017). 'Hard-Aware Deeply Cascaded Embedding'. In: **IEEE International Conference on Computer Vision, CVPR**, pp. 814–823.

Zelnik-Manor, L. and Perona, P. (2005). 'Self-tuning spectral clustering'. In: **Advances in Neural Information Processing Systems (NIPS)**, pp. 1601–1608.

Zhai, A. and Wu, H.-Y. (2019). 'Classification is a Strong Baseline for Deep Metric Learning'. In: **CoRR** vol. abs/1811.12649.

Zhang, X. et al. (2016). 'Embedding Label Structures for Fine-Grained Feature Representation'. In: **IEEE Conference on Computer Vision and Pattern Recognition, CVPR**, pp. 1114–1123.

Zhao, K., Xu, J. and Cheng, M. (2019). 'RegularFace: Deep Face Recognition via Exclusive Regularization'. In: **IEEE Conference on Computer Vision and Pattern Recognition, CVPR**, pp. 1136–1144.

Zheng, X. et al. (2019). 'Towards Optimal Fine Grained Retrieval via Decorrelated Centralized Loss with Normalize-Scale Layer'. In: **Conference on Artificial Intelligence, AAAI**, pp. 9291–9298.

Zhou, D. et al. (2004). 'Learning with local and global consistency'. In: **Advances in neural information processing systems**, pp. 321–328.

Zhu, X., Ghahramani, Z. and Lafferty, J. D. (2003). 'Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions'. In: **Machine Learning,**

Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA, pp. 912–919.

Zhu, X. (2005). 'Semi-supervised Learning with Graphs'. PhD thesis. Pittsburgh, PA, USA.

Zhu, X. and Ghahramani, Z. (2002). 'Learning from labeled and unlabeled data with label propagation'. In: