Consistency, Robustness and Sparsity for Learning Algorithms

Konsistenz, Robustheit und Dünnbesetztheit von Lern-Algorithmen

zur Erlangung des Doktorgrades

Dr. rer. nat.

im Studiengang Mathematik

am Department Mathematik der
Friedrich-Alexander-Universität Erlangen-Nürnberg

vorgelegt am **24.Juli 2023**

von **Tim Roith**

Prüfer: Martin Burger
Betreuer: M.Sc. C

Für eine besondere Person,
ohne sie hätte ich es nie geschafft.

Acknowledgement

I would like to thank my supervisors for helping me finish this thesis and supporting me with their profound knowledge..

Contents

List of Figures

Preface

This work is structured into two main parts, Part I the presentation and explanation of the topics and results presented in Part II, the peer-reviewed articles.

Part I: Exposition	Part II: Prints
Chapter 1: Learning Paradigms	
Chapter 2: Consistent Semi-Supervised Learning on Sparse Graphs	????
Chapter 3: Robust and Sparse Supervised Learning	????

Part I consists of three chapters, of which the first explains the paradigms, *unsupervised*, *semi-supervised* and *supervised* learning. The other chapters are the split up thematically, concerning the topics semi-supervised and supervised learning respectively. In each of these chapters a short introduction provides the necessary framework allowing us to explain the main contributions. The following publications are reprinted in Part II:

[LIP-I] T. Roith and L. Bungert. "Continuum limit of Lipschitz learning on graphs." In: *Foundations of Computational Mathematics* (2022), pp. 1–39.

[LIP-II] L. Bungert, J. Calder, and T. Roith. "Uniform convergence rates for Lipschitz learning on graphs." In: *IMA Journal of Numerical Analysis* (Sept. 2022). DOI: 10.1093/imanum/drac048.

[CLIP] L. Bungert et al. "CLIP: Cheap Lipschitz training of neural networks." In: *Scale Space and Variational Methods in Computer Vision: 8th International Conference, SSVM 2021, Virtual Event, May 16–20, 2021, Proceedings*. Springer. 2021, pp. 307–319.

[BREG-I] L. Bungert et al. "A bregman learning framework for sparse neural networks." In: *Journal of Machine Learning Research* 23.192 (2022), pp. 1–43.

[FNO] S. Kabri et al. "Resolution-Invariant Image Classification based on Fourier Neural Operators." In: *Scale Space and Variational Methods in Computer Vision: 9th International Conference, SSVM 2023, Proceedings*. Springer. 2023, pp. 307–319.

The following two works that are not part of this thesis but provide an additional insight.

[LIP-III] L. Bungert, J. Calder, and T. Roith. *Ratio convergence rates for Euclidean first-passage percolation: Applications to the graph infinity Laplacian*. 2022. arXiv: 2210.09023 [math.PR].

[BREG-II] L. Bungert et al. "Neural Architecture Search via Bregman Iterations." In: (2021). arXiv: 2106.02479 [cs.LG].

TR's Contribution

Here we list TR's contribution to the publications included in the thesis.

[LIP-I]: This work builds upon the findings in TR's master thesis [Roi21]. It is however important to note that the results constitute a significant extension and are conceptually stronger than the ones in [Roi21], see Section 2.3. TR adapted the continuum limit framework to the L^∞ case, worked out most of the proofs and wrote a significant part of the paper. In collaboration with LB, he identified the crucial domain assumptions that allow to work on non-convex domains and proved convergence for approximate boundary conditions.

[LIP-II]: In collaboration with LB, TR worked on the convergence proofs building upon the ideas of JC. He contributed to both the numeric and the analysis conducted in the paper.

[CLIP]: TR worked out the main algorithm proposed in the paper together with LB, based on LB's idea. Together with LS and RR he conducted the numerical examples and also wrote most of the source code. Furthermore, he wrote large parts of the paper.

[BREG-I]: TR expanded LB's ideas of employing Bregman iteration for sparse training. Together with MB and LB he worked out the convergence analysis of stochastic Bregman iterations. Here, he also proposed a profound sparse initialization strategy. Furthermore, he conducted the numerical examples and wrote most of the source code.

[FNO]: This work is based on SK's masters thesis, employing the initial ideas of MB for resolution invariance with FNOs. In the paper TR worked out the proofs for well-definedness and Fréchet-differentiability, together with SK. He wrote large parts of the paper and the source code. Here, he conducted the numerical studies in collaboration with SK.

Part I.

Exposition

Chapter 1
Learning Paradigms

Throughout this thesis, we assume to be given data $\mathcal{X}_n \subset \mathcal{X} \subset \mathbb{R}^d$ consisting of n data points. We consider task of *learning* a function $f : \tilde{\mathcal{X}} \to \mathcal{Y}$ from the given data, where the two most important cases for us are

- **classification**: f assigns a label to each $x \in \tilde{\mathcal{X}}$ out of a total of $C \in \mathbb{N}$ possible classes, i.e. $\mathcal{Y} = \{1, \dots, C\}$. In some architectures the last layer of the neural network is given as a vector $y \in \mathbb{R}^C$. Typically, this vector is a probability vector, i.e.

$$y \in \Delta^C := \left\{ z \in [0,1]^d : \sum_{i=1}^{C} z_i = 1 \right\}.$$

  This can be enforced via the softmax function [Bri90] $Q : \mathbb{R}^d \to \mathbb{R}^d$

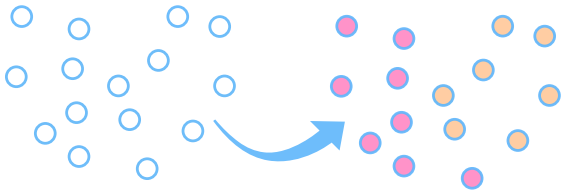
$$Q(z)_i := \frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_j)}$$

  which was actually introduced by Boltzman in [Bol68]. This allows the interpretation that the ith entry of $f_\theta(x) \in \Delta^C$ models the probability that x belongs to class i. In order to obtain a label one can simply choose the maximum entry, $\operatorname{argmax}_{i=1,\dots,C} f_\theta(x)_i$.

- **image denoising**: f outputs a denoised version of an input image. Here we have $\mathcal{X} = \mathcal{Y} = \mathbb{R}^{K \times N \times M}$, where
  - $K \in \mathbb{N}$ is the number of color channels,
  - N, M denote the width and height of the image.

The set $\tilde{\mathcal{X}} \subset \mathbb{R}^d$ is usually either the set of data points $\mathcal{X}_n$ or the whole space $\mathcal{X}$. The learning paradigms we consider in this thesis, differ by their usage of labeled data. We review the concepts in the following.

1.1. Unsupervised Learning

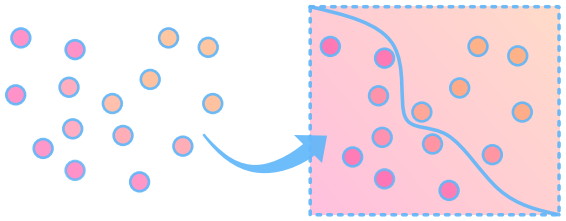


In this case we are not given any labeled data. In our context the most important application is data clustering. Other tasks involve dimensionality reduction or density estimation, see [ST14]. The clustering task consists of grouping data based on some similarity criterion. In this sense, clustering can also be interpreted as classification, i.e., the desired function is a mapping $f : \tilde{\mathcal{X}} \to \{1, \dots, C\}$ where $C \in \mathbb{N}$ denotes the number of clusters. Typically, one wants to obtain a clustering of the given data set, i.e., $\tilde{\mathcal{X}} = \mathcal{X}_n$. We list some of the typical clustering methods below:

- K-means algorithm [Ste+56],
- Expectation Maximization [DLR77],
- Cheeger cuts [GS15; SB09; Gar+16; GMT22],
- spectral clustering [GS18].

Unsupervised learning is not the main focus of this present work. However, we note that especially the concepts developed in [GS15] for Cheeger cuts are crucial for the continuum limit framework in Section 2.3.

1.2. Supervised Learning



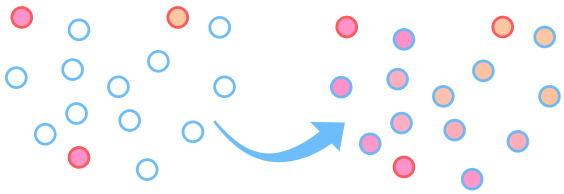
In this setting, each data point $x \in \mathcal{X}_n$ is labeled, via a given function $g : \mathcal{X}_n \to \mathcal{Y}$ such that we have a finite training set $\mathcal{T} = \{(x, g(x)) : x \in \mathcal{X}_n\}$. The task is then to infer a function defined on the underlying space, i.e. $f : \mathcal{X} \to \mathcal{Y}$, i.e. we want to assign a label to unseen $x \in \mathcal{X}$ that are not necessarily part of the given data. Often, one models the problem via a joint probability function $P_{\mathcal{X},\mathcal{Y}}$ and assumes that the training data are i.i.d. w.r.t. $P_{\mathcal{X},\mathcal{Y}}$. In this interpretation, a neural network can aim to model the conditional $P(y|x)$ for an input $x \in \mathcal{X}$ and output $y \in \mathcal{Y}$.

In order to *learn* the function f from the given data, one needs to choose a parameterized class of functions $\mathcal{U}$, where typically each element can be describe by a finite number of parameters. Among others, common methods or parametrizations include

- Support vector machines [CV95; SS05],
- decision Trees [MS63; Bre+84],
- neural networks [Tur04; Ros58; MP69].

In ?? we exclusively focus on supervised learning algorithms employing neural networks. We refer to [Sch15] for an exhaustive historical overview. The concrete setting and learning framework is given in Chapter 3.

1.3. Semi-Supervised Learning



In the semi-supervised setting we assume that only a fraction of the data $\mathcal{X}_n$ is labeled, i.e., we are given a a function $g : \mathcal{O}_n \to \mathcal{Y}$ where $\mathcal{O}_n \subset \mathcal{X}_n$ is the set of labeled data. Typically the labeled data constitutes only a small fraction of all available points, i.e. $|\mathcal{O}_n| << |\mathcal{X}_n|$. In this thesis we restrict ourselves to the *transductive setting*, i.e. we want to infer a function acting only on the data $f : \mathcal{X}_n \to \mathcal{Y}$. This is opposed to the inductive setting, where f also classifies unseen points $x \in \mathcal{X}$, [Zhu05]. Common algorithms and methods include

- expectation maximization and mixture models [DLR77; CCC+03],
- self-training and co-training [BM98],
- graph-based learning [Zhu05].

In Chapter 2 we focus on graph-based learning algorithms, however we refer to [Zhu05] for a wonderful overview of semi-supervised learning algorithms.

Chapter 2
Consistent Semi-Supervised Learning on Sparse Graphs

This chapter considers the consistency of graph-based semi-supervised learning methods and contextualizes the topics provided in the prints [LIP-I; LIP-II; LIP-III]. We are given partially labeled data, where the task is to obtain a labeling on the whole graph. Here, we consider learning algorithms on graphs and are especially interested in the following aspects:

- **Consistency:** what is the behavior in the infinite data limit?
- **Sparsity:** how does sparsity of the graph weights affect this behavior?

In [LIP-I] we first consider Γ-convergence of discrete L^∞ functionals to their continuum counterpart. This allows to then show convergence of minimizers and therefore consistency. The proofs allow for very sparse graphs, however they only directly apply to the Lipschitz extension task, which does not allow for unique solutions. We refer to Section 2.3 for more details.

The issue of non-uniqueness is addressed in [LIP-II], which considers uniform convergence of graph infinity harmonic functions to their continuum counterpart. These functions are in fact special solutions of the extension task in [LIP-I], namely so-called absolutely minimizing Lipschitz extensions, which are—under certain assumptions—unique. Again, we are able to work with a very mild scaling assumption that allows for sparse graphs. Furthermore, we are able to prove the first quantitative convergence rates, where we provide the key insight, that a rate for graph distance functions transfers to a rate for infinity harmonic functions. The main contributions are presented in Section 2.5.

To allow even sparser graphs, directly at the connectivity threshold, we consider first passage percolation in [LIP-III]. Here, we prove a ratio convergence of a graph distance function on a Poisson point cloud. The results can ultimately be employed to show convergence rates for Lipschitz learning. Details are provided in **??**.

Before we present these works we first introduce the setting of consistency for graph-based SSL in Section 2.1. We then discuss the p-Laplacian in Section 2.1.2 both in the continuum and the graph setting. Similarly, we then introduce the problem we are actually concerned namely the ∞-Laplacian and Lipschitz extensions in Section 2.2. With

the necessary tools and background we are then able to present the main contributions on consistency of Lipschitz learning in the infinite data limit.

Section 2.1: Graph-based SSL and Consistency

Section 2.1.2: The p Laplacian

??: Continuum Setting

??: Graph Setting

Section 2.2 Lipschitz Extensions

Section 2.2.1: Continuum Setting

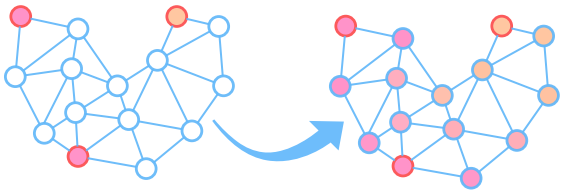
Section 2.2.2: Graph Setting

Section 2.3: [LIP-I]

Section 2.4: [LIP-II]

Section 2.5: [LIP-III]

2.1. Graph-based SSL and Consistency



In the semi-supervised learning setting of Section 1.3, we are given a finite set $\Omega_n \subset \tilde{\Omega}$ consisting of n points, where $\tilde{\Omega}$ denotes the input space. We assume that a non-empty subset $\mathcal{O}_n \subset \Omega_n$ is labeled, i.e., we are given a labeling function $\mathbf{g} : \mathcal{O}_n \to \mathbb{R}$. From now on we denote functions acting on a discrete set Ω_n using bold symbols to distinguish them from functions acting on the continuum $\tilde{\Omega}$. In particular, the space of functions on Ω_n can simply be identified with $\mathbb{R}^n$, since $\{\mathbf{u} : \Omega_n \to \mathbb{R}\} \sim \mathbb{R}^n$. The semi-supervised learning problem consists in finding an extension of $\mathbf{g}$ from $\mathcal{O}_n$ to the whole set Ω_n, i.e.

$$\begin{aligned} &\text{find } \mathbf{u} : \Omega_n \to \mathbb{R}, \\ &\text{such that } \mathbf{u}(x) = \mathbf{g}(x) \text{ for all } x \in \mathcal{O}_n. \end{aligned} \tag{SSL}$$

In order to obtain meaningful solutions, one usually incorporates the *smoothness assumption* [ST14] which can be informally stated as follows:

"*Points that are close together are more likely to share a similar label.*"

Remark 2.1 (Notational Remark). In Chapter 1 we employ the symbol $\mathcal{X}$ to denote the input set, which is now replaced by Ω. This due to the fact, that the prints [LIP-II; LIP-III; LIP-I] employ this notation for which we choose to use it here. △

Often one assumes that the points $x \in \Omega_n$ are i.i.d. w.r.t. a distribution μ with density $\rho : \Omega \to \mathbb{R}$. However, as observed in [Roi21], the data distribution is not directly relevant for the works in [LIP-I; LIP-II]. Therefore, we can usually work with more general sets Ω_n.

2.1.1. Weighted Graphs

In order to employ the smoothness assumption we need a notion of "closeness" on the set Ω_n, for which we introduce weighted graphs. Namely we define a weighting function w that allows to compare points in Ω_n and therefore induces the desired notion.

Definition 2.2 (Weighted Graphs). For a finite set Ω_n and a weight function $w_n : \Omega_n \times \Omega_n \to \mathbb{R}$, the tuple (Ω_n, w_n) is called a *weighted graph*.

Other Notions of Graphs Typically, a graph is defined as a pair (Ω_n, E) where E denotes the set of edges. Here, one has two cases:

- *Undirected graph:* $E = \{\{x, y\} : \text{there is an edge between } x \in \Omega_n \text{ and } y \in \Omega_n\}$, i.e., s $E \subset 2^{\Omega_n}$ and each edge is undirected, since $\{x, y\} = \{y, x\}$.
- *Directed graph:* $E = \{(x, y) : \text{there is an edge from } x \text{ to } y\}$, i.e., $E \subset \Omega_n \times \Omega_n$ and each edge is directed and $(x, y) \neq (y, x)$.

Additionally, one then considers a weight function $W : E \to \mathbb{R}$ assigning a weight to each edge, and then defines the triple (Ω_n, E, W) as a weighted graph. However, we note that all this information can be represented much more elegantly by a weight function $w : \Omega_n \times \Omega_n \to \mathbb{R}$. A directed edge set $E \subset \Omega_n \times \Omega_n$ can be equivalently expressed by a weight function $w : \Omega_n \times \Omega_n \to \mathbb{R}$ where $w(x, y) > 0$ if and only if $(x, y) \in E$, a weighting $W : E \to \mathbb{R}$ naturally transfers to w. In the case of an undirected graph, one simply requires the weight function to be symmetric, i.e., $w(x, y) = w(y, x)$ for all $x, y \in \Omega_n$. Furthermore, in the above definition the set Ω_n does not enter the definition up to ordering of its $n \in \mathbb{N}$ elements. Therefore, a graph could be entirely represented by a weight function $w : \{1, \dots, n\} \times \{1, \dots, n\} \to \mathbb{R}$. However, since the definition of w will incorporate information about points $\Omega_n \subset \mathbb{R}^d$ we use the notation (Ω_n, w_n) for graphs in the following.

Kernels and the Graph Scale In most of our applications the data Ω_n is given as a subset of $\mathbb{R}^n$ and in fact we are interested in the limit $n \to \infty$, where Ω_n fills out a domain $\tilde{\Omega} \subset \mathbb{R}^d$. In the continuum we are interested in *local* operators incorporating changes of functions $u : \tilde{\Omega} \to \mathbb{R}$ at an infinitesimal small scale. Since interactions on a graph are inherently non-local in the Euclidean sense, we need to localize the interaction on the graph in the limit $n \to \infty$. A popular choice that guarantees this behavior is to set

$$w(x, y) = \eta_\varepsilon(|x - y|)$$

where $\eta_\varepsilon : [0, \infty) \to [0, \infty]$ is a kernel function depending on a scaling parameter $\varepsilon \in \mathbb{R}^+$. The parameter ε is also referred to as the *graph scale* and informally speaking determines the scale of the graph interactions. I.e., the smaller ε the smaller the interaction radius

of points in Ω_n should be. In our specific setting of L^∞ problems it is typical to choose

$$\eta_\varepsilon(\cdot) = \frac{1}{c_\eta\,\varepsilon}\eta\left(\frac{\cdot}{\varepsilon}\right)$$

where η is a non-increasing kernel and c_η is a constant depending on the kernel. Typical examples of kernels include

- (constant weights) $\eta(t) = 1_{[0,1]}(t)$,
- (exponential weights) $\eta(t) = \exp(-t^2/(2\sigma^2))1_{[0,1]}(t)$,
- (non-integrable weights) $\eta(t) = \frac{1}{t^p}1_{[0,1]}(t)$ with $p \in (0,1]$.

Remark 2.3. In the continuum limit one needs to consider the value

$$\sigma_\eta = \sup_{t\in\mathbb{R}^+} t\,\eta(t),$$

which is assumed to be finite. Considering graph problems in L^p for $p < \infty$ the corresponding value is given as

$$\sigma_\eta^{(p)} = \int_{\mathbb{R}^+} \eta(t)\,t^{d+p}dt$$

which was first employed in [GS15] for $p = 1$ and then in [ST19] for general $p < \infty$. We see that $\sigma_\eta^{(p)}$ degenerates to σ_η in the limit $p \to \infty$.
In [LIP-I] we choose $c_\eta = 1$ and therefore σ_η then appears in the limit functional. In [LIP-II] we rescale the kernel, i.e., $c_\eta = \sigma_\eta$ which allows to work with unrescaled operators in the limit. △

Remark 2.4. In order to obtain continuum limits in the case $p < \infty$ on usually employs a factor of $1/\varepsilon^{p+d}$ in front of the graph weights [GS15; ST19]. In Section 2.2.2 we see that this factor degenerates to $1/\varepsilon$ in the case $p \to \infty$. The intuition here, is that problems in L^∞ do not respect mass in a quantitative but rather a qualitative manner. Therefore, factors like ε^d which appear because of integrals in $\mathbb{R}^d$ do not contribute for $p = \infty$. △

Sparse Graphs An important practical aspect connected to the graph scale is the sparsity of the graph, which is given by the numbers of non-zero elements in the weight matrix $W \in \mathbb{R}^{n\times n}$

$$W_{ij} := w(x_i, x_j) \quad x_i, x_j \in \Omega_n$$

where we assume an ordering $\Omega_n = \{x_1, \dots, x_n\}$. In order to have a computationally feasible problem this matrix should have very few non-zero elements. Assuming that the kernel η has compact support, w.l.o.g. $\operatorname{supp}(\eta) \subset [0,1]$ we observe that the sparsity is directly influenced by the graph scale ε. Namely, $|x_i - x_j| > \varepsilon \Rightarrow W_{ij} = 0$, see Fig. 2.1.

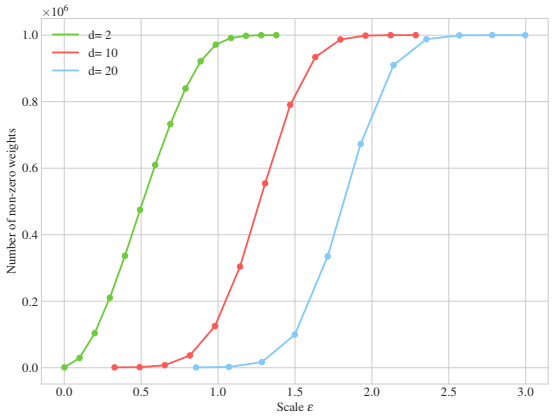


Figure 2.1.: Dependence of the number of non-zero edges on the scaling parameter ε. Here, we sample $n \in \mathbb{N}$ points uniformly in the cube $[0,1]^d$ and only connect connect vertices $x_i, x_j \in \Omega_n$ if $\|x_i - x_j\|_2 < \varepsilon$. The plots shows the behavior for $d = 2, 10, 20$.

Therefore, from a practical point of view ε should be chosen relatively small. However, showing consistency of graph-based SSl algorithms often requires scaling assumptions that permit the desired length scales [GS15; ST19; Cal19]. In Section 2.3 we give concrete examples of such scaling assumptions. One of the main goals of [LIP-I; LIP-II] was to show convergence and rates at the smallest possible length scale, which was indeed achieved.

Remark 2.5. In many practical applications graph weights connecting all points within ε-ball of Euclidean distance are inferior to knn graphs [Cal+20; FCL19; CT22]. While most consistency results employ the ε-ball setting, recently, the authors in [CT22] were able to show convergence rates in the knn setting. In this thesis we focus on the ε-ball setting, however it is an interesting open question to transfer the results of [LIP-I; LIP-II; LIP-III] to the knn setting. △

2.1.2. The p-Laplacian: Continuum and Graph

The archetype of learning methods we consider in the following is so-called *Laplacian learning*, which had one of its first appearances in [ZGL03]. The associated problem was given as

$$\begin{aligned} &\min_{\mathbf{u}:\Omega_n \to \mathbb{R}} \sum_{x,y \in \Omega_n} w_n(x,y)^2 \left(\mathbf{u}(y) - \mathbf{u}(x)\right)^2 \\ &\text{subject to } \mathbf{u}(x) = \mathbf{g}(x) \text{ for all } x \in \mathcal{O}_n. \end{aligned} \tag{2.1}$$

Here, we always assume non-negative weights $w_n : \Omega_n \times \Omega_n \to [0, \infty)$. The intuitive idea behind this method is that it minimizes a discrete approximation of the Dirichlet energy and should therefore enforce a certain kind of "smoothness" of the solution $\mathbf{u}$. In

[ZS05] it was proposed to generalize the idea of the graph Dirichlet energy in Eq. (2.1) to arbitrary $1 \leq p < \infty$ motivated by the continuum counterpart. While The contributions in [LIP-I; LIP-II; LIP-III] mainly consider the case of $p = \infty$, the case $p < \infty$ serves as a motivation for the whole discussion. Therefore, we briefly review the situation in the following. Here, we first introduce the continuum setting and then discuss the situation on the graph.

Continuum Setting We follow the exposition in [Lin17]. Let $\Omega \subset \mathbb{R}^d$ be a bounded domain, then we consider the p-Dirichlet energy for functions $u \in W^{1,p}(\Omega)$,

$$\mathcal{E}_{p,\rho}(u) := \int_\Omega |\nabla u|^p \, \rho(x)^2 \, dx, \tag{2.2}$$

where $\rho : \Omega \to \mathbb{R}$ is some given function. In the setting of [GS15; ST19] ρ denotes the density of data distribution as in Section 2.1. For $\rho \equiv 1$ we simply write $\mathcal{E}_{p,1} = \mathcal{E}_p$. The associated variational problem is given in the following.

Problem 2.6 (Variational Formulation). For $p \in (1, \infty)$ and $V \subset W^{1,p}(\Omega)$ find $u \in W^{1,p}(\Omega)$ such that

$$\mathcal{E}_{p,\rho}(u) \leq \mathcal{E}_{p,\rho}(v)$$

for all v, such that $(u - v) \in W_0^{1,p}$.

Assume for simplicity that $\rho \equiv 1$ and that $u \in V$ is a minimizer of the above problem, then its first variation must vanish, i.e., for all $\phi \in C_0^\infty(\Omega)$ one has

$$\int_\Omega \langle |\Delta_p u|^p \, \nabla u, \nabla \phi \rangle \, dx = 0. \tag{2.3}$$

A function $u \in V$ satisfying Eq. (2.3) is called a *weak solution* of the p-Laplace equation. In fact, if u is smooth enough one has that

$$\Delta_p u := \operatorname{div}(|\Delta_p u|^{p-2} \, \nabla u) = 0 \tag{2.4}$$

where Δ_p is called the p-Laplacian. Boundary conditions on $\partial\Omega$ given by a function $g \in W^{1,p}(\Omega)$ can be incorporated by considering the set $V_g := \{u \in W^{1,p}(\Omega) : u - g \in W_0^{1,p}(\Omega)\}$. We state the following classical result, which can for example be found in [Lin17].

Theorem 2.7 (Existence and Uniqueness). For $p \in (1, \infty)$ and $g \in W^{1,p}(\Omega)$ there exists a unique minimizer $u \in V_g$ of the p-Dirichlet energy, i.e.,

$$\operatorname{argmin}_{u \in V_g} \mathcal{E}_p(u) = u.$$

Moreover, u is a weak solution of the p-Laplace equation and there exists a function $\tilde{u} \in C(\Omega)$ such that $u = \tilde{u}$ a.e. in Ω. If $g \in C(\Omega)$ and Ω is sufficiently smooth, then $\tilde{u}|_{\partial\Omega} = g|_{\partial\Omega}$.

Proof. The proof can be found in [Lin17, Thm. 2.16]. □

Local Minimization Property Let u solve Problem 2.6 for given boundary values $g \in W^{1,p}(\Omega)$ and let $V \subset\subset \Omega$ be a sufficiently regular subset. Then we have that

$$\mathcal{E}_p(u) = \int_V |\nabla u|^p \, dx + \int_{\Omega \setminus V} |\nabla u|^p \, dx.$$

Now consider Problem 2.6 on V with boundary values given by u and denote by v the solution of the problem. Since $u = v$ on ∂V we can extend v onto Ω by setting $v = u$ in $\Omega \setminus V$, which gives $v \in W^{1,p}(\Omega)$. Therefore, we obtain

$$\mathcal{E}_p(v) = \int_V |\nabla v|^p \, dx + \int_{\Omega \setminus V} |\nabla u|^p \, dx \leq \int_V |\nabla u|^p \, dx + \int_{\Omega \setminus V} |\nabla u|^p \, dx = \mathcal{E}_p(u).$$

Since u was unique this yields $u|_V = v|_V$, for which know that any solution on Ω solves Problem 2.6 also on any nice subset, with the boundary values given by itself. In Section 2.2 we see that this *local minimization property* does not hold automatically and has to enforced additionally. The underlying reason here is, that integrals are set-additive, but suprema are not, which is similarly explained in [ACJ04].

Laplacian Learning A natural extension of the problem given in Eq. (2.2) is obtained by considering the target functional

$$\mathbf{E}_p^{w_n}(\mathbf{u}) := \sum_{x,y \in \Omega_n} w_n(x,y)^p \, |\mathbf{u}(y) - \mathbf{u}(x)|^p \,,$$

which we refer to as the graph p-Dirichlet energy. Indeed, we notice structural similarities to the p-Dirichlet energy $\mathcal{E}_p$ in Eq. (2.2), replacing the integral by a finite sum and derivatives by weighted finite differences

$$w_n(x,y)^p \, |\mathbf{u}(y) - \mathbf{u}(x)|^p \,.$$

This naturally leads to the following minimization problem.

Problem 2.8 (Graph Energy Minimization). Given a weighted graph (Ω_n, w_n) and a labeling function $\mathbf{g} : \mathcal{O}_n \to \mathbb{R}$, for $\mathcal{O}_n \subset \Omega_n$ we consider the problem

$$\min_{\mathbf{u}:\Omega_n \to \mathbb{R}} \mathbf{E}_p^{w_n}(\mathbf{u}) \text{ subject to } \mathbf{u}(x) = \mathbf{g}(x) \text{ for all } x \in \mathcal{O}_n.$$

Since every function $\mathbf{u} : \Omega_n \to \mathbb{R}$ can be identified with a vector $\mathbf{u} \in \mathbb{R}^n$, the above problem is in fact an optimization problem in $\mathbb{R}^n$. However, one can prove existence with techniques very similar to the continuum case in [Lin17]. We give the adapted prove below for completeness. The important property to establish uniqueness is that the graph is connected to the boundary, i.e., for every $x \in \Omega_n$ there exists a path $x = \gamma_1, \ldots, \gamma_m \in \Omega_n = o$ with $w(\gamma_{i+1}, \gamma_i) > 0$ connecting x to a point $o \in \mathcal{O}_n$ in the boundary. This is very intuitive, since on any connected component that does not communicate with the boundary, an arbitrary constant minimizes the "graph gradient".

Theorem 2.9 (Existence and Uniqueness). Let (Ω_n, w_n) be a graph with non-negative weights, that is connected to its boundary $\mathcal{O}_n \subset \Omega_n$. Then Problem 2.8 admits a unique solution $\mathbf{u} : \Omega_n \to \mathbb{R}$.

Proof. The proof is an adaption from the continuum case in [Lin17, Thm. 2.16]. We start by proving uniqueness, Let $E = \{(x, y) : w_n(x, y) > 0\}$ be the set of all active edges and denote by $\nabla^{w_n}\mathbf{u} \in \mathbb{R}^{|E|}$

$$(\nabla^{w_n}\mathbf{u})_{(x,y)} := w_n(x, y)\,|\mathbf{u}(y) - \mathbf{u}(x)|$$

the *graph gradient* on E. We have that $\mathbf{E}_p^{w_n}(\mathbf{u}) = \|\nabla^{w_n}\mathbf{u}\|$. Let $\mathbf{u}_1, \mathbf{u}_2$ be two solutions of Problem 2.8, i.e. $\mathbf{E}_p^{w_n}(\mathbf{u}_1) = \mathbf{E}_p^{w_n}(\mathbf{u}_2)$. Assume that there exists $(\bar{x}, \bar{y}) \in E$ such that $\nabla^{w_n}\mathbf{u}_1(\bar{x}, \bar{y}) \neq \nabla^{w_n}\mathbf{u}_2(\bar{x}, \bar{y})$, then we have that

$$\begin{aligned}
\mathbf{E}_p^{w_n}(\mathbf{u}_1) \leq \mathbf{E}_p^{w_n}\left(\frac{\mathbf{u}_1 + \mathbf{u}_2}{2}\right) &= \left\|\frac{\nabla^{w_n}\mathbf{u}_1(x, y) + \nabla^{w_n}\mathbf{u}_2(x, y)}{2}\right\|_p^p \\
&< \frac{1}{2}\sum_{(x,y)\in E\setminus\{(\bar{x},\bar{y})\}} |\nabla^{w_n}\mathbf{u}_1(x, y)|^p + |\nabla^{w_n}\mathbf{u}_2(x, y)|^p \\
&\quad + \frac{1}{2}\,|\nabla^{w_n}\mathbf{u}_1(\bar{x}, \bar{y})|^p + |\nabla^{w_n}\mathbf{u}_2(\bar{x}, \bar{y})|^p \\
&= \frac{1}{2}\left(\mathbf{E}_p^{w_n}(\mathbf{u}_1) + \mathbf{E}_p^{w_n}(\mathbf{u}_2)\right) = \mathbf{E}_p^{w_n}(\mathbf{u}_1),
\end{aligned}$$

which is a contradiction. Here, we employed the strict convexity of $t \mapsto |t|^p$ which implies $|t + \bar{t}|^p \leq 2^{p-1}\left(|t|^p + |\bar{t}|^p\right)$, where the inequality is sharp if $t \neq \bar{t}$. Therefore, we obtain that

$$\mathbf{u}_1(x) - \mathbf{u}_1(y) = \mathbf{u}_2(x) - \mathbf{u}_2(y) \qquad \text{for all} \qquad (x, y) \in E. \tag{2.5}$$

Since (Ω_n, w_n) is connected to $\mathcal{O}_n$ for any $x \in \Omega_n$ we can find a path $x = \gamma_1, \ldots, \gamma_m = o$ connecting x to a point $o \in \mathcal{O}_n$. By definition $w_n(\gamma_{i+1}, \gamma_i) > 0$ for all $i = 1, \ldots, m-1$ and therefore using Eq. (2.5) we obtain

$$\mathbf{u}_1(\gamma_{i+1}) - \mathbf{u}_1(\gamma_i) = \mathbf{u}_2(\gamma_{i+1}) - \mathbf{u}_2(\gamma_i) \qquad \text{for all} \qquad i = 1, \ldots, m-1.$$

and therefore

$$\begin{aligned}
\mathbf{u}_1(x) - \mathbf{u}_1(o) &= \left[\sum_{i=1}^{m-1} \mathbf{u}_1(\gamma_{i+1}) - \mathbf{u}_1(\gamma_i)\right] + \mathbf{u}_1(o) = \left[\sum_{i=1}^{m-1} \mathbf{u}_2(\gamma_{i+1}) - \mathbf{u}_2(\gamma_i)\right] \\
&= \mathbf{u}_2(x) - \mathbf{u}_2(o).
\end{aligned}$$

Employing the boundary conditions $\mathbf{u}_1(o) = \mathbf{u}_2(o)$, it follows that $\mathbf{u}_1(x) = \mathbf{u}_2(x)$. Since $x \in \Omega_n$ was arbitrary, we get $\mathbf{u}_1 = \mathbf{u}_2$.
To show existence, we do not need to assume that the graph is connected to the boundary. In fact one easily observes that being constant, on every connected component that is

not connected to $\mathcal{O}$ is optimal. W.l.o.g. we neglect connected components not connected to the boundary and see that for $\mathbf{u} : \Omega_n \to \mathbb{R}$ and any $x \in \Omega_n$ we have that

$$|\mathbf{u}(x)|^p \leq \sum_{i=1}^{m-1} |\mathbf{u}(\gamma_{i+1}) - \mathbf{u}(\gamma_i)|^p + |\mathbf{g}(o)|^p$$

where we employed a path to the boundary as in the previous example. Therefore, we obtain that $\|\mathbf{u}\|_p^p \leq C(\mathbf{E}_p^{w_n}(\mathbf{u}) + \|\mathbf{g}\|_p^p)$. Let now $\mathbf{u}_j$ denote a sequence such that $\mathbf{E}_p^{w_n}(\mathbf{u}_j) \leq I + 1/j$, where

$$I := \inf_{\mathbf{u}=\mathbf{g} \text{ on } \mathcal{O}_n} \mathbf{E}_p^{w_n}(\mathbf{u}).$$

Then we have that

$$\|\mathbf{u}_j\|_p^p \leq C(I + 1/j + \|\mathbf{g}\|_p^p) \leq \tilde{C}$$

i.e. $\mathbf{u}_j$ is a bounded sequence of vectors in the finite dimensional space $\mathbb{R}^n$, and therefore there exists a subsequence—which we do not relabel—that converges to $\mathbf{u}$. Since the functional $\mathbf{E}_p^{w_n} : \mathbb{R}^n \to \mathbb{R}$ is continuous we obtain that

$$\mathbf{E}_p^{w_n}(\mathbf{u}) = \lim_{j\to\infty} \mathbf{E}_p^{w_n}(\mathbf{u}_j) = I$$

which yields existence. □

The Graph Laplacian In the continuum case one considers the Euler–Lagrange equation for the functional $\mathcal{E}_p$, which yields the p-Laplacian, see Section 2.1.2. Analogously, the optimality conditions for the graph p-Dirichlet energy $\mathbf{E}_p^{w_n}$ yield

$$\Delta_p^{w_n}\mathbf{u}(x) := \sum_{y\in\Omega_n} w_n(x,y)^p \, |\mathbf{u}(y) - \mathbf{u}(x)|^{p-2} \, (\mathbf{u}(y) - \mathbf{u}(x)) = 0, \text{ for all } x \in \Omega_n,$$

where $\Delta_p^{w_n}$ is referred to as thegraph p-Laplacian operator. This yields the graph p-Laplacian problem.

Problem 2.10 (Graph p-Laplacian). Given a weighted graph (Ω_n, w_n) and a labeling function $\mathbf{g} : \mathcal{O}_n \to \mathbb{R}$ with $\mathcal{O}_n \subset \Omega_n$, find a function $\mathbf{u} : \Omega_n \to \mathbb{R}$ such that

$$\begin{aligned} \Delta_p^{w_n}\mathbf{u} &= 0, \text{ in } \Omega_n \setminus \mathcal{O}_n, \\ \mathbf{u} &= \mathbf{g} \text{ on } \mathcal{O}_n. \end{aligned}$$

Since the functional $\mathbf{E}_p^{w_n}$ has a unique minimizer subject to the constraints given by $\mathbf{g}$ and the graph p-Laplacian is derived via optimality conditions, one expects that the Problem 2.8 and Problem 2.10 are equivalent. This is formulated in the following theorem.

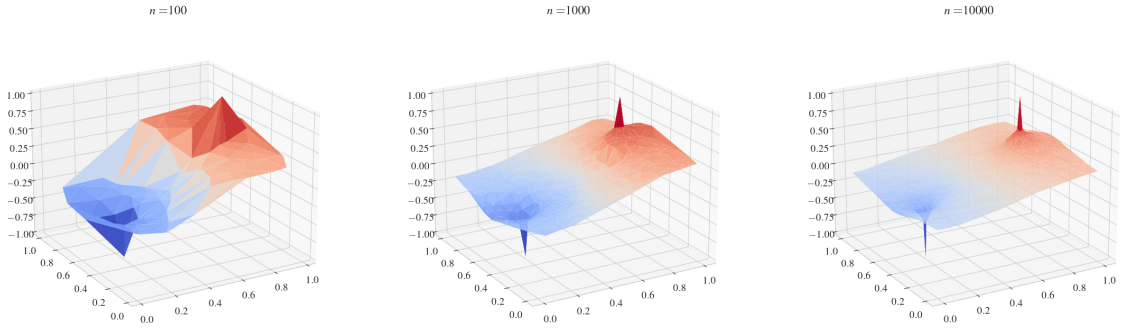


Figure 2.2.: Solution to the Laplacian Learning problem ($p = 2$) for different number of data points $n \in \{100, 1000, 10000\}$.

Theorem 2.11 (Existence and Uniqueness). Let (Ω_n, w_n) be a graph with non-negative weights, that is connected to its boundary $\mathcal{O}_n \subset \Omega_n$. Then there exists a unique solution $\mathbf{u} : \Omega_n \to \mathbb{R}$ to Problem 2.10, which also is the unique minimizer of Problem 2.8.

Proof. This statement can be proven similarly to [ETT15, Theorem 5.3]. □

2.1.3. Consistency for Graph-based SSL

While there are many examples where this simple concept does in fact yield good solutions [ZGL03; ZS05], it has been observed that solutions tend to degenerate for large amounts of data, whenever $d \geq 2$ [NSZ09; AL11; El +16]. We illustrate this phenomenon in Example 2.12.

Example 2.12. We sample different number of points $n \in \{100, 1000, 10000\}$ in $(0, 1)^2$ and keep a fixed constraint on $\mathcal{O}_n = \{o_1, o_2\} = \{(0.2, 0.5), (0.8, 0.5)\}$ with $\mathbf{g}(o_1) = -1, \mathbf{g}(o_2) = 1$. In Fig. 2.2 we observe that for increasing n the solution $\mathbf{u}$ of Eq. (2.1) tends to be a constant equal to the average of the given constraints and spike at the given constraints.

This observation motivates the driving question of this chapter.

Are Semi-Supervised Learning algorithms consistent in the infinite data limit?

In our case "consistency" roughly ask the discrete solutions to converge to the continuum counterparts they are motivated by, in the data limit $n \to \infty$. Here, we distinguish between point-wise consistency as in [VBB08; GK06; HAV05], consistency on a function space level [ST19; GS15; Cal19; LIP-I; LIP-II] and consistency of a discretized operator [Cal19; LIP-III]. We detail the notion of consistency in the following.

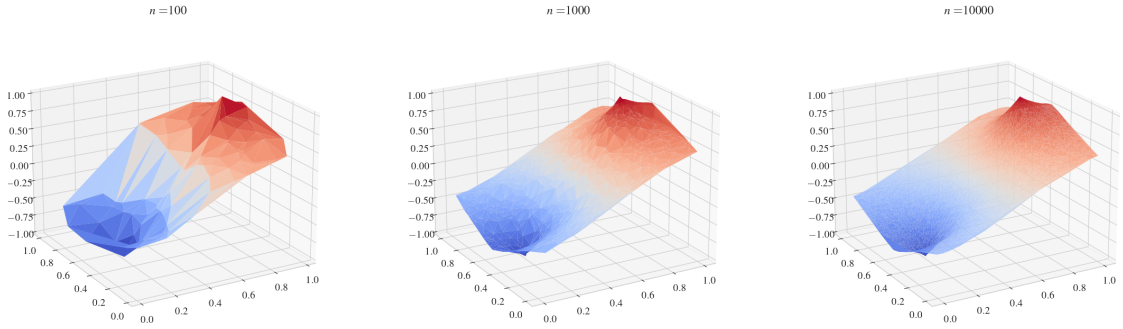


Figure 2.3.: Solution to the Laplacian Learning problem ($p = \infty$) for different number of data points $n \in \{100, 1000, 10000\}$. The setup is otherwise copied from Example 2.12

Consistency for the p-Laplacian As observed in [NSZ09; AL11; El +16] the question of consistency is now connected to relation between p and d, where divides into the three regimes $p < d$, $p = d$ and probably the most relevant case $p > d$.

The p-Dirichlet problem in the continuum can be formulated as a variational problem in $W^{1,p}$. However, it is important to remember that in our setting we most likely consider pointwise constraints on a discrete set $\mathcal{O}$ even in the continuum. In order to impose pointwise constraints, one requires that functions in $W^{1,p}$ exhibit some continuity properties. By the Sobolev embedding theorem, this is the case when $p > d$, [AF03]. This leads to a first qualitative intuition that consistency can only be achieved in the case $p > d$. The situation is in fact more subtle, which is reviewed in Section 2.3.

It is important to note that p-harmonic functions are actually more regular also in the case $p \leq d$, beyond the implication of the Sobolev embedding theorem. However, this regularity does not help for the continuum limits [NSZ09; AL11; El +16].

Sending p to Infinity Since the assumption that $p > d$ is crucial for asymptotic consistency, a natural idea is to send p to ∞ and analyze the corresponding limit problem. This yields the so-called *Lipschitz learning* task [vLB04; Kyn+15], which is the main point of interest in this chapter. We formalize this problem in Section 2.2. In Fig. 2.3 we observe that for $p = \infty$ we indeed obtain smoother solutions, compared to Fig. 2.2. In this regard Lipschitz learning seems promising to overcome the consistency issue.

However, as already noticed in [El +16] being a pure L^∞ problem, Lipschitz learning does not respect the density of data in any way. Namely, the method is exclusively distance based. This behavior is a drawback in many machine learning applications. However, by carefully rescaling the graph weights one obtains a data sensitive problem, see [Cal19] and Section 2.3.

2.2. Lipschitz extensions and the infinity Laplacian: continuum and graph

2.2.1. The continuum setting

This chapter studies the limit $p \to \infty$ of the p-Laplace equation. We first recall, that for $u \in W^{1,\infty}(\Omega)$ we have that

$$\lim_{p\to\infty} \mathcal{E}_p(u)^{1/p} = \operatorname*{ess\,sup}_{x\in\Omega} |\nabla u(x)| =: \mathcal{E}_\infty(u),$$

see [Jen93]. The functional $\mathcal{E}_\infty$ is weak*-lower semicontinuous over $W^{1,\infty}(\Omega)$, (see e.g. [BJW01, Thm. 2.6]). In the classical theory developed by Jensen in [Jen93] one considers the following problem, which is tries to "minimize the *sup-norm* of the gradient" as described by Jensen.

Problem 2.13 (Variational gradient-sup problem). For an open domain $\Omega \subset \mathbb{R}^d$ find a function $u \in W^{1,\infty}(\Omega)$ such that

$$\|\nabla u\|_\infty \leq \|\nabla v\|_\infty \text{ for every } v, \text{s.t. } (u - v) \in W_0^{1,\infty}.$$

The above problem becomes more relevant when we additionally impose boundary values $g : \partial\Omega \to \mathbb{R}$. Here, [Jen93] draws the connection to so-called *Lipschitz extensions*, which are the driving concept in this section. We introduce a more general viewpoint—that does not require the notion of a gradient—later on, but first introduce the variational problem.

The intrinsic metric and the Lipschitz constant. As noticed in [Jen93] working with the Lipschitz constant and the sup-norm of the gradient requires a careful treatment of the distance measurement. Let $\tilde{\Omega}$ be a set and let d be a semi-metric on $\tilde{\Omega}$, that is d fulfills the requirement of a metric up to triangle inequality. Then we define the Lipschitz constant of a function $u : V \to \mathbb{R}$ on a subset $V \subset \tilde{\Omega}$ as

$$\operatorname*{Lip}_d(u; V) := \sup_{x,y\in V, x\neq y} \frac{|u(x) - u(y)|}{d(x, y)}.$$

If d denotes the Euclidean distance we use omit the subscript. i.e. $\operatorname{Lip}_d = \operatorname{Lip}$. Additionally, we can introduce the space of Lipschitz functions $\operatorname{Lip}_d(V)$ on V via $u \in \operatorname{Lip}_d(V) \Leftrightarrow \operatorname{Lip}_d(u; V) < \infty$.

Remark 2.14 (Lipschitz and Sobolev functions). If $\Omega \subset \mathbb{R}^d$ is sufficiently regular, e.g., it has Lipschitz boundary then we have that

$$\operatorname{Lip}(\Omega) = W^{1,\infty}(\Omega),$$

where this identity is of course to be understood in the sense of equivalence classes in Ł^p spaces. We refer to [**evansgariepy**] for a proof of this result. △

The above remark already relates Lipschitz with $W^{1,\infty}$ functions. Often however, we need a quantitative comparison between the Lipschitz constant and the sup-norm of the gradient of a function. Here, it is essential which distance measure is chosen for the Lipschitz constant. For an open domain $\Omega \subset \mathbb{R}^d$ we have the inequality

$$\|\nabla u\|_\infty \leq \sup_{x \neq y} \frac{|u(x) - u(y)|}{|x - y|}$$

which can be proven via the definition of the gradient. For the reverse inquality, one has to respect the geometry of the domain, namely for $x, y \in \Omega$ we have that

$$|u(x) - u(y)| \leq \|\nabla u\|_\infty \; d_\Omega(x, y) \tag{2.6}$$

see, e.g., [BB11, Prop9.3, Rem. 7], where

$$d_\Omega(x, y) = \inf \left\{ \int_0^1 |\dot{\gamma}(t)| \, dt : \gamma \in C^1([0, 1], \Omega) \text{ with } \gamma(0) = x, \gamma(1) = y \right\}$$

denotes the *geodesic distance* on Ω. If Ω is convex, we have that $d_\Omega(x, y) = |x - y|$ for every $x, y \in \Omega$ and therefore Eq. (2.6) yields $\mathrm{Lip}(u) = \|\nabla u\|_\infty$. However, this situation changes for non-convex domains, see Example 2.15. Additionally it is often necessary to define a distance measure on the closure of $\Omega \subset \mathbb{R}^d$. In order to have a geodesic on $\overline{\Omega}$ one can simply consider $d_{\overline{\Omega}}$, see e.g. [**unif**], which then yields the length space $(\overline{\Omega}, d_{\overline{\Omega}})$. In the classical theory developed in [Jen93] one alternatively considers

$$\tilde{d}_{\overline{\Omega}}(x, y) := \liminf_{(\tilde{x}, \tilde{y}) \to (x, y)} d_\Omega(\tilde{x}, \tilde{y}).$$

The differences between these notation are demonstrated in the following example. Also note, that $\tilde{d}_{\overline{\Omega}}$ is only a semi-metric on $\overline{\Omega}$ since it is lacking a triangle inequality.

Example 2.15. For $I = [-\pi, c] \cup [c, \pi]$ with $c = \pi/6$ we consider the domain

$$\bigcup_{\theta \in I} B_1\left((\cos(\theta), \sin(\theta))\right)$$

which is visualized in Fig. 2.4 and the points $x = (2\,c, 1), y = (2\,c, -1)$. The line segment between x and y contains the point $z = (2\,c, 0)$, however $z \notin \Omega$. One can show that the geodesic has the length $d_\Omega(x, y) = 4\,\cos(\pi/6) + \pi \approx 6.606$ which is the length of the dotted path in Fig. 2.4. However, we observe that

$$\overline{\Omega} = \bigcup_{\theta \in I} \overline{B_1\left((\cos(\theta), \sin(\theta))\right)}$$

and in particular $z \in \overline{B_1\left((c, c)\right)}$, therefore $d_{\overline{\Omega}}(x, y) = 2$.

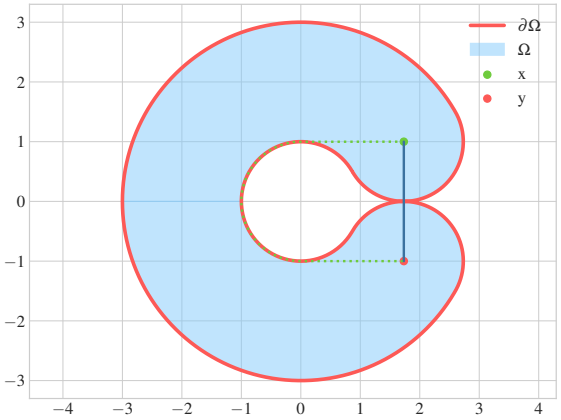


Figure 2.4.: The domain in Example 2.15.

Solutions to the gradient-sup problem Before generalizing the theory of Lipschitz extension to arbitrary metric spaces, we first note, that one can explicitly construct solutions. Namely, for given $g \in \mathrm{Lip}(\partial\Omega)$ the functions

$$\begin{aligned} \overline{g}(x) &:= \inf_{y\in\partial\Omega} g(y) + \operatorname*{Lip}_{\tilde{d}_{\overline{\Omega}}}(g;\partial\Omega)\cdot \tilde{d}_{\overline{\Omega}}(x,y) \\ \underline{g}(x) &:= \sup_{y\in\partial\Omega} g(y) - \operatorname*{Lip}_{\tilde{d}_{\overline{\Omega}}}(g;\partial\Omega)\cdot \tilde{d}_{\overline{\Omega}}(x,y) \end{aligned} \tag{2.7}$$

are solutions to the gradient-sup problem Problem 2.13 that coincide with g on $\partial\Omega$, see [Jen93, Th. 1.8].

Remark 2.16. The same concept of constructing solutions is applied in the following sections in a more abstract setting. These solutions are then called Whitney and McShane or respectively maximal and minimal extensions, see Lemma 2.19. △

One easily observes that there are cases where $\overline{g} \neq \underline{g}$ and therefore the sup-gradient problem does not admit a unique solution. A concrete example, to showcase this phenomena is given in [Jen93, p. 53].

Lipschitz extensions in metric spaces The problem considered in the last section was motivated by a variational problem for $\mathcal{E}_\infty(u) = \|\nabla u\|_\infty$. However, the theory of Lipschitz extensions provides a more general framework. Namely, here we do not assume that Ω is a subset of $\mathbb{R}^d$ and rather consider a metric space $(\tilde{\Omega}, d)$ with $\Omega \subset \tilde{\Omega}$.

Remark 2.17. For applications within this thesis we have that $\Omega \subset \mathbb{R}^d$ is an open bounded domain and then consider $\tilde{\Omega} := \overline{\Omega}$, i.e., the closure of Ω within the topology induced by the Euclidean distance. In this abstract setting however, we use an abstract space $\tilde{\Omega}$ while still being close notation wise. △

A result originally due to Kierszbraun [Kir34] states that for two Hilbert spaces H_1, H_2, a subset $\mathcal{O} \subset H_1$ and a function $g : \mathcal{O} \to H_1$ there exists a function $u : H_1 \to H_2$ such

that

$$u = g \text{ on } U,$$
$$\mathrm{Lip}(u; H_1) = \mathrm{Lip}(g; \mathcal{O}).$$

Here, the metrics for the respective Lipschitz constants are induced by the inner products of the Hilbert spaces. We refer to [Kir34] for the original proof and to [Sch69, Th. 1.31] for a proof of the version as stated above. In this work we only consider the case $H_2 = \mathbb{R}$ which allows for more general assumption on the space H_1. We now formulate the Lipschitz extension problem in our setting.

Problem 2.18 (Lipschitz Extensions). Let $(\tilde{\Omega}, d)$ be a metric space and $\mathcal{O} \subset \tilde{\Omega}$ be a bounded subset. For a given Lipschitz function $g : \mathcal{O} \to \mathbb{R}$ find a Lipschitz function $u : \tilde{\Omega} \to \mathbb{R}$ such that

$$\operatorname*{Lip}_d(u; \tilde{\Omega}) = \operatorname*{Lip}_d(g; \mathcal{O}).$$

A function $u : \tilde{\Omega}$ with this property is called *Lipschitz extension* of g to $\tilde{\Omega}$.

In this setting one can explicitly construct solutions of the Lipschitz extension task. They are not unique, however one has an upper and a lower bound. In fact, conceptually these solutions are very similar to the functions in Eq. (2.7) and even coincide, whenever the sup-norm of the gradient is given as the Lipschitz constant.

Lemma 2.19. In the setting of Problem 2.18 we have that the

- **Whitney (or maximal) extension:** $\overline{g}(x) := \inf_{y \in \mathcal{O}} g(y) + \mathrm{Lip}_d(g; \mathcal{O}) \cdot d(x, y)$ and the
- **McShane (or minimal) extension:** $\underline{g}(x) := \sup_{y \in \mathcal{O}} g(y) - \mathrm{Lip}_d(g; \mathcal{O}) \cdot d(x, y)$

defined for $x \in \tilde{\Omega}$ are Lipschitz extensions of g to $\tilde{\Omega}$. Moreover, let $u : \tilde{\Omega} \to \mathbb{R}$ be any Lipschitz extension of g, the we have that

$$\underline{g} \leq u \leq \overline{g}.$$

Proof. We refer to [Whi92] and [McS34] for the proofs of the respective result. □

As demonstrated in Example 2.20, there are cases where $\overline{g} \neq \underline{g}$ and therefore, Lipschitz extensions are not unique in general. Furthermore, [ACJ04] points out that the Whitney and McShane extension do not allow for a comparison principle, which can also be observed in Example 2.20.

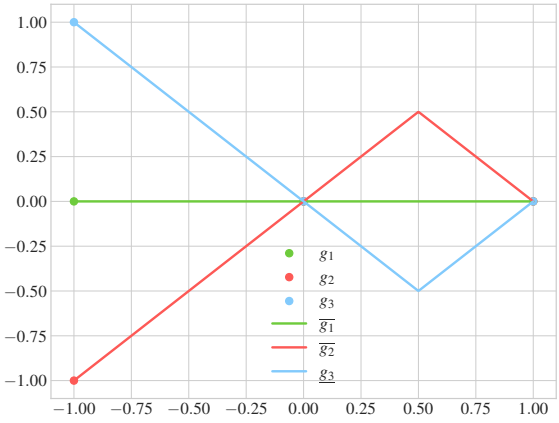


Figure 2.5.: The maximal extension does not admit a comparison principle, as demonstrated in Example 2.20.

Example 2.20. Consider the set $\tilde{\Omega} = [-1, 1]$ and $\mathcal{O} = \{-1, 0, 1\}$ with

$$\begin{aligned} g_1(x) &:= 0, \\ g_2(x) &:= 1/2(x - \operatorname{sign}(x) \cdot x), \\ g_3(x) &:= -g_2. \end{aligned}$$

Then we have that $g_2 \leq g_1$ on $\mathcal{O}$ but

$$\overline{g_2} > \overline{g_1} \text{ in } (0, 1),$$

see Fig. 2.5 for a visualization. Analogously, we have that $g_3 \geq g_1$ on $\mathcal{O}$ but

$$\underline{g_3} < \underline{g_1} \text{ in } (0, 1).$$

Absolutely Minimizing Extension Sending $p \to \infty$ in the variational formulation of the p-Laplace equation yields the Lipschitz extension task, which however does not admit for unique solutions. So the question arises, which property is lost in the limit case. For $p < \infty$ one has the crucial local minimization property. Let $\Omega \subset \mathbb{R}^d$ be an open domain and denote by u_p the solution of the p-Laplace problem with boundary values $g \in W^{1,p}(\Omega)$. Then we know that u_p is also a minimizer of $\mathcal{E}_p$ on every subset $V \subset \Omega$, i.e.,

$$\int_V |\nabla u_p|^p \, dx \leq \int_V |\nabla v|^p \, dx$$

for any function v such that $(u_p - v) \in W_0^{1,p}(V)$, see e.g. []. This lead Aronsson to introduce the concept of *absolutely minimizing Lipschitz extension* in [Aro67]. A

function $u_\infty \in W^{1,\infty}$ is called absolutely minimal, if

$$\operatorname*{ess\,sup}_{x\in V} |\nabla u| \leq \operatorname*{ess\,sup}_{x\in V} |\nabla v| \text{ for every open } V \subset \Omega \tag{2.8}$$

and every function v such that $(u-v) \in W_0^{1,\infty}$. I fact one can also show, that $u_p \xrightarrow{p\to\infty} u_\infty$ ([Aro67]), which seems to validate the notion of absolute minimizers. In [**tour**] it was shown, that one has an equivalent formulation involving the Lipschitz constant. For a given Lipschitz function $f : \overline{\Omega} \to \mathbb{R}$ we have that u_∞ with $(u_\infty - f) \in W_0^{1,\infty}(\Omega)$ fulfills Eq. (2.8) iff

$$\operatorname{Lip}(u_\infty; V) \leq \operatorname{Lip}(v; V) \text{ for every } V \subset \Omega$$

and every function v such that $(u - v) \in W_0^{1,\infty}(V)$, see [**tour**].

Maybe not Sobolev here

In this thesis we work with a notion of absolute minimizers, which is equivalent to the above formulation for convex domains in $\mathbb{R}^d$. However, ...

Problem 2.21 (AMLEs). Let $(\tilde{\Omega}, d)$ be a length space, $\mathcal{O} \subset \tilde{\Omega}$ a closed subset and $g : \mathcal{O} \to \mathbb{R}$ a Lipschitz function. Find an extension $u \in C(\tilde{\Omega})$ such that $u = g$ on $\mathcal{O}$ and

$$\operatorname*{Lip}_d(u; \overline{V}) = \operatorname*{Lip}_d(u, \partial V) \text{ for all open and connected sets } V \subset \tilde{\Omega} \setminus \mathcal{O}.$$

A function u fulfilling this property is called absolutely minimizing Lipschitz extension of g.

Remark 2.22. In our application Ω is an open subset of $\mathbb{R}^d$ and we then choose $\tilde{\Omega} = \overline{\Omega}$. Here, it is important to note that the topological notions like boundary and interior are to be understood relative to $\overline{\Omega}$. A visualization odf this concept can be found in Fig. 2.6. △

Comparison with Cones

The infinity Laplacian

2.2.2. Graph Lipschitz Extensions

We now consider the limit $p \to \infty$ of Problem 2.8 in the graph case. Analogously to **??** we derive

$$\lim_{p\to\infty} \left(\mathbf{E}_p^{w_n}(\mathbf{u})\right)^{1/p} = \max_{x,y\in\Omega_n} w_n(x,y)\,|\mathbf{u}(y) - \mathbf{u}(x)| =: \mathbf{E}_\infty^{w_n}(\mathbf{u})$$

which extends the graph p-Laplacian energy to the case $p = \infty$. Again we notice structural similarities to the continuum version $\mathcal{E}_\infty$.

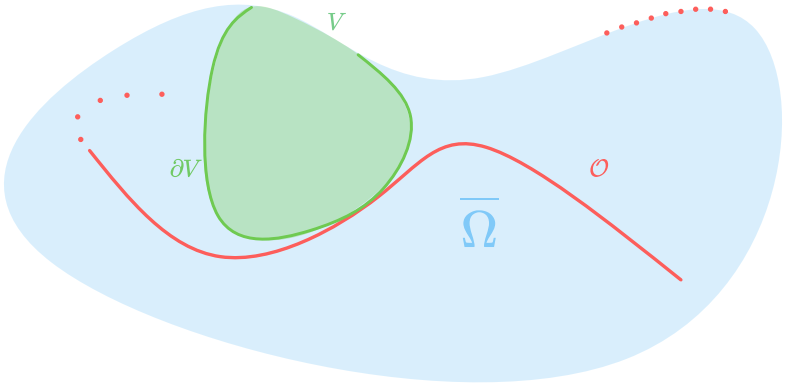


Figure 2.6.: A set $V \subset \overline{\Omega}$ can be relatively open w.r.t. the metric space $\overline{\Omega}$ although, $V \cap \partial\Omega \neq \emptyset$, where $\partial\Omega$ is the boundary within the standard topology on $\mathbb{R}^d$. The relative boundary of $\partial_{\overline{\Omega}} V$ does not include any parts of $\partial\Omega$.

Remark 2.23. Informally speaking the functional $\mathbf{E}$ combines elements of a gradient and Lipschitz constant. Assuming that $w_n(x, y)$ relates to $|x - y|$ we see that the finite difference approximation resembles a Lipschitz constant. However, typically $w_n(x, y)$ has also some localizing property which fits the interpretation of a gradient better. △

This functional now leads to the graph Lipschitz extension problem.

Problem 2.24 (Graph Energy Minimization). Given a weighted graph (Ω_n, w_n) and a labeling function $\mathbf{g} : \mathcal{O}_n \to \mathbb{R}$, for $\mathcal{O}_n \subset \Omega_n$ we consider the problem

$$\min_{\mathbf{u}:\Omega_n \to \mathbb{R}} \mathbf{E}_\infty^{w_n}(\mathbf{u}) \text{ subject to } \mathbf{u}(x) = \mathbf{g}(x) \text{ for all } x \in \mathcal{O}_n.$$

Since the weighting function $w_n : \Omega_n \times \Omega_n \to \mathbb{R}_0^+$ does not induce a metric, Problem 2.24 does not directly fit the framework of the abstract Lipschitz Extension in Problem 2.18. However, we can consider paths in (Ω_n, w_n), connecting arbitrary $x, y \in \Omega_n$ i.e. vectors $\gamma \in \Omega_n^{\times k}$ such that

$$\begin{aligned} w_n(\gamma_i, \gamma_{i+1}) > 0 \quad &\text{for all } i = 1, \dots, k-1, \\ \gamma_1 &= x, \\ \gamma_k &= y \end{aligned}$$

for which we define the length as

$$|\gamma| = \sum_{i=1}^{k-1} w(\gamma_i, \gamma_{i+1})^{-1}.$$

This yields the metric space (Ω_n, d_{w_n}), where $d_{w_n} : \Omega_n \times \Omega_n \to \mathbb{R}$ is defined as

$$d_{w_n}(x, y) := \min \{|\gamma| : \gamma \text{ is a path in } (\Omega_n, w_n) \text{ from } x \text{ to } y\}. \tag{2.9}$$

Remark 2.25. We note that it is important to only consider non-negative weights, otherwise any loop with a negative "length" would decrease the length of the whole path arbitrarily. However, restricting ourselves to non-negative weights we can easily see, that the minimum in Eq. (2.9) is indeed attained. △

With this definition we can consider the Lipschitz extension task of $g : \mathcal{O}_n \to \mathbb{R}$ to Ω_n within the metric space (Ω_n, d_{w_n}), i.e. within the setting of Problem 2.18. Therefore the question arises, whether the minimization problem in Problem 2.24 is equivalent to the metric Lipschitz extension problem for which we have the following lemma.

Lemma 2.26. For a graph (Ω_n, w_n) with non-negative weights and a function $\mathbf{u} : \Omega_n \to \mathbb{R}$ we have that

$$\mathbf{E}_\infty^{w_n}(\mathbf{u}) = \operatorname*{Lip}_{d_{w_n}}(\mathbf{u}).$$

Furthermore, for $\mathcal{O}_n \subset \Omega_n$ and a function $\mathbf{g} : \mathcal{O}_n \to \mathbb{R}$ and we have that

$$\mathbf{g} = \mathbf{u} \text{ on } \mathcal{O}_n \Rightarrow \operatorname*{Lip}_{d_{w_n}}(\mathbf{g}; \mathcal{O}_n) \leq \operatorname*{Lip}_{d_{w_n}}(\mathbf{u}).$$

Proof. **Step 1:** We show that $\operatorname{Lip}_{d_{w_n}}(\mathbf{u}) \leq \mathbf{E}_\infty^{w_n}(\mathbf{u})$.
We can choose a path $\gamma \in \Omega_n^{\times k}$ such that

$$\operatorname*{Lip}_{d_{w_n}}(\mathbf{u}) = \frac{\mathbf{u}(\gamma_1) - \mathbf{u}(\gamma_k)}{|\gamma|}.$$

The path γ allows to compare vertices $\gamma_1, \gamma_k \in \Omega_n$ that aren't necessarily neighbors in the graph. However, each consecutive vertices in the path are neighbors in the graph and therefore we have

$$w_n(\gamma_i, \gamma_{i+1}) \, |\mathbf{u}(\gamma_{i+1}) - \mathbf{u}(\gamma_i)| \leq \mathbf{E}_\infty^{w_n}(\mathbf{u}) \quad \text{for all } i = 1, \dots, k-1. \tag{2.10}$$

We now employ an elementary result for numbers $a_i \in \mathbb{R}_0^+, b_i \in \mathbb{R}^+, i = 1, \dots m \in \mathbb{N}$, namely

$$[a_i \cdot b_i \leq c \in \mathbb{R} \quad \text{for } i = 1, \dots, m] \Rightarrow \frac{\sum_{i=1}^m a_i}{\sum_{i=1}^m b_i^{-1}} \leq c \tag{2.11}$$

which can be seen as follows

$$\begin{aligned} & a_i \cdot b_i \leq c \quad \text{for } i = 1, \dots, m \\ \Rightarrow\ & a_i \leq b_i^{-1} \cdot c \quad \text{for } i = 1, \dots, m \\ \Rightarrow\ & \sum_{i=1}^m a_i \leq \left(\sum_{i=1}^m b_i^{-1} \right) \cdot c \\ \Rightarrow\ & \frac{\sum_{i=1}^m a_i}{\sum_{i=1}^m b_i^{-1}} \leq c. \end{aligned}$$

This then yields

$$\frac{|\mathbf{u}(\gamma_1) - \mathbf{u}(\gamma_k)|}{|\gamma|} \leq \frac{\sum_{i=1}^{k-1} |\mathbf{u}(\gamma_i) - \mathbf{u}(\gamma_{i+1})|}{|\gamma|} = \frac{\sum_{i=1}^{k-1} |\mathbf{u}(\gamma_i) - \mathbf{u}(\gamma_{i+1})|}{\sum_{i=1}^{k-1} w_n(\gamma_i, \gamma_{i+1})^{-1}} \leq \mathbf{E}_\infty^{w_n}(\mathbf{u})$$

where in the last inequality we employed Eq. (2.11) together with Eq. (2.10).
Step 2: We show that $\mathrm{Lip}_{d_{w_n}}(\mathbf{u}) \geq \mathbf{E}_\infty^{w_n}(\mathbf{u})$.
Let $x, y \in \Omega_n$, then we know that $d_w(x,y) \leq w_n(x,y)^{-1}$ and therefore

$$|\mathbf{u}(x) - \mathbf{u}(y)|\, w_n(x,y) \leq \frac{|\mathbf{u}(x) - \mathbf{u}(y)|}{d_w(x,y)} \leq \max_{\bar{x},\bar{y} \in \Omega_n} \frac{|\mathbf{u}(\bar{x}) - \mathbf{u}(\bar{y})|}{d_w(\bar{x},\bar{y})} = \mathrm{Lip}_{d_{w_n}}(\mathbf{u}).$$

Since this holds for arbitrary $x, y \in \Omega_n$ we have that

$$\mathbf{E}_\infty^{w_n}(\mathbf{u}) = \max_{x,y \in \Omega_n} |\mathbf{u}(x) - \mathbf{u}(y)|\, w_n(x,y) \leq \mathrm{Lip}_{d_{w_n}}(\mathbf{u}).$$

Step 3: We show that $\mathrm{Lip}_{d_{w_n}}(\mathbf{g}; \mathcal{O}_n) \leq \mathrm{Lip}_{d_{w_n}}(\mathbf{u})$.
If $\mathbf{g} = \mathbf{u}$ on $\mathcal{O}$ this simply follows since the maximum for the Lipschitz constant of $\mathbf{u}$ is taken over a larger set. Indeed,

$$\begin{aligned}\mathrm{Lip}_{d_{w_n}}(\mathbf{u}) &= \max_{x,y \in \Omega_n} \frac{|\mathbf{u}(x) - \mathbf{u}(y)|}{d_w(x,y)} \geq \max_{x,y \in \mathcal{O}_n} \frac{|\mathbf{u}(x) - \mathbf{u}(y)|}{d_w(x,y)} = \max_{x,y \in \mathcal{O}_n} \frac{|\mathbf{g}(x) - \mathbf{g}(y)|}{d_w(x,y)} \\ &= \mathrm{Lip}_{d_{w_n}}(\mathbf{u}; \mathcal{O}_n).\end{aligned}$$

□

This lemma shows that the abstract Lipschitz extension task considered on the metric space (Ω_n, d_{w_n}) and the Graph ∞-Dirichlet minimization task are indeed equivalent. Therefore, we also have the Whitney and McShane extensions

$$\begin{aligned}\overline{\mathbf{g}}(x) &= \inf_{y \in \mathcal{O}_n} \mathbf{g}(y) + d_{w_n}(x,y) \\ \underline{\mathbf{g}}(x) &= \sup_{y \in \mathcal{O}_n} \mathbf{g}(y) - d_{w_n}(x,y)\end{aligned}$$

as solutions on the graph. Analogously, the problem does not admit for unique solutions.

Absolutely Minimizing Graph Extensions Similarly to Section 2.2.1 we can now consider absolutely minimizing extensions. However, the problem in Problem 2.21 uses a notion of a boundary and it is not directly clear how to infer this concept to the discrete set Ω_n. Therefore, we define the following what we mean by "boundary" on a graph.

Definition 2.27. Let (Ω_n, w_n) be a weight graph and let $V \subset \Omega_n$ be a subset, then we define

- the **exterior** boundary as $\partial^{\text{ext}} := \{x \in \Omega_n \setminus V : w_n(x,y) > 0 \text{ for some } y \in V\}$,
- the **interior** boundary as $\partial^{\text{int}} := \{x \in V : w_n(x,y) > 0 \text{ for some } y \in \Omega_n \setminus V\}$.

The closure of V is then defined as $\overline{V}^{\text{ext}} V := V \cup \partial^{\text{ext}}$ and the interior as $\overset{\circ}{V}^{\text{int}} := V \setminus \partial^{\text{int}} V$.

We note that it is not possible to define a topology on Ω_n that would yield the above notions. Namely, the only admissible topology in our case would be the discrete topology, i.e., 2^{Ω_n}. However, in this topology the only closed sets are $\emptyset$ and Ω_n which is not useful for the applications in the following. Using the Kuratowski closure axioms [Kur22] we remark the following.

Lemma 2.28. The exterior closure on a weighted graph (Ω_n, w_n) is a preclosure or Čech closure.

Proof. Here, we use the notion of a preclosure in [ČFK66]. We first see that $\overline{\emptyset}^{\text{ext}} = \emptyset$ and that $V \subset \overline{V}^{\text{ext}}$ for every subset $V \subset \Omega_n$, i.e. the above defined closure preserves the empty set and is extensive. Furthermore, for two sets $V_1, V_2 \subset \Omega_n$ we have that

$$\begin{aligned}
&x \in \partial^{\text{ext}}(V_1 \cup V_2) \\
\Leftrightarrow\ &[x \notin V_1 \cup V_2] \wedge [\exists y \in V_1 \cup V_2 : w_n(x,y)] \neq 0 \\
\Leftrightarrow\ &[x \notin V_1 \cup V_2] \wedge \Big([\exists y \in V_1 : w_n(x,y) \neq 0] \vee [\exists y \in V_2 : w_n(x,y) \neq 0] \Big) \\
\Leftrightarrow\ &\Big[x \in \partial^{\text{ext}} V_1 \setminus V_2\Big] \vee \Big[x \in \partial^{\text{ext}} V_2 \setminus V_1\Big] \\
\Leftrightarrow\ &x \in (\partial^{\text{ext}} V_1 \cup \partial^{\text{ext}} V_2) \setminus (V_1 \cup V_2).
\end{aligned}$$

We have shown that $\partial^{\text{ext}}(V_1 \cup V_2) = (\partial^{\text{ext}} V_1 \cup \partial^{\text{ext}} V_2) \setminus (V_1 \cup V_2)$. Therefore, we have that

$$\begin{aligned}
\overline{V_1 \cup V_2}^{\text{ext}} &= V_1 \cup V_2 \cup \partial^{\text{ext}}(V_1 \cup V_2) \\
&= V_1 \cup V_2 \cup ((\partial^{\text{ext}} V_1 \cup \partial^{\text{ext}} V_2) \setminus (V_1 \cup V_2)) \\
&= V_1 \cup \partial^{\text{ext}} V_1 \cup V_2 \cup \partial^{\text{ext}} V_2 \\
&= \overline{V_1}^{\text{ext}} \cup \overline{V_2}^{\text{ext}}.
\end{aligned}$$

This shows that the closure preserves binary unions and therefore we have shown, that it is indeed a Čech closure. □

The missing property, that inhibits the closure to induce a topology is the so-called idempotence. Namely, there are sets $V \subset \Omega_n$ such that

$$\overline{V}^{\text{ext}} \neq \overline{\overline{V}^{\text{ext}}}^{\text{ext}}.$$

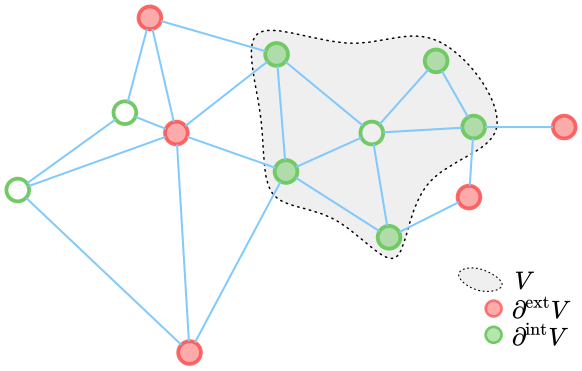


Figure 2.7.: Visualization of exterior and interior boundary on a graph.

E.g. in the example visualized in Fig. 2.7 we see that $\overline{\overline{V}^{\text{ext}}}^{\text{ext}} = \Omega_n \neq \overline{V}^{\text{ext}}$. Since the closure we employ here does not induce a topology, we have a slightly modified notion of absolutely minimizers.

Problem 2.29 (Graph AMLEs). Given a connected weighted graph (Ω_n, w_n), $\mathcal{O}_n \subset \Omega_n$) and a function $\mathbf{g} : \mathcal{O}_n \to \mathbb{R}$ find a function $\mathbf{u} : \Omega_n \to \mathbb{R}$ such that

$$\begin{aligned} \operatorname*{Lip}_{d_{w_n}}(\mathbf{u}; \overline{V}^{\text{ext}}) &= \operatorname*{Lip}_{d_{w_n}}(\mathbf{u}; \partial^{\text{ext}} V) \text{ for all connected } V \subset \Omega_n \setminus \mathcal{O}_n, \\ \mathbf{u} &= \mathbf{g} \text{ on } \mathcal{O}_n. \end{aligned}$$

Comparison with graph Distance functions Analogously to the continuum case **??** we can also consider comparison with distance functions on graphs. The main ingredients here, are the graph distance function d_{w_n} and the notion of closure on a graph as developed in the last section.

Definition 2.30. For a weighted graph (Ω_n, w_n) we say a function $\mathbf{u} : \Omega_n \to \mathbb{R}$ fulfills comparison with distance function from above (CDFA) on a subset $U \subset \Omega_n$ if for every $V \subset U$ we have

$$\max_{\overline{V}^{\text{ext}}} (u + a\, d_{w_n}(\cdot, z)) = \max_{\partial^{\text{ext}} V} (u + a\, d_{w_n}(\cdot, z)) \tag{CDFA}$$

for every $z \in \Omega_n \setminus V$ and every $a \in \mathbb{R}$. We say that $\mathbf{u}$ fulfills comparison with distance function from below (CDFB) on a subset $U \subset \Omega_n$ if for every $V \subset U$ we have

$$\min_{\overline{V}^{\text{ext}}} (u - a\, d_{w_n}(\cdot, z)) = \min_{\partial^{\text{ext}} V} (u - a\, d_{w_n}(\cdot, z)) \tag{CDFB}$$

for every $z \in \Omega_n \setminus V$ and every $a \in \mathbb{R}$.

Analogously to the continuum case, we say that a function fulfills comparison with distance functions, if it fulfills both, Eq. (CDFA) and Eq. (CDFB). Existence of such functions is established later, we are first interested in the question of uniqueness. Since the notion of graph boundaries is not directly compatible with the usual definitions on metric spaces, we prove it separately. Here, we adapt arguments from [smart] and [LeGruyer]. To do so we first consider the operators

$$\mathbf{S}^{\varepsilon}\mathbf{u}(x) := \max_{y\in\Omega_n : d_{w_n}(x,y)\leq\varepsilon} \mathbf{u}(y) \quad \mathbf{S}_{\varepsilon}\mathbf{u}(x) := \min_{y\in\Omega_n : d_{w_n}(x,y)\leq\varepsilon} \mathbf{u}(y)$$

and proof the following lemma, which is the analogue of

The Graph infinity Laplacian We can also obtain the limit of the Graph p-Laplace operator via the following formal calculation,

$$\begin{aligned}
&\Delta_p^{w_n}\mathbf{u}(x) = 0 \\
\Leftrightarrow &\sum_{y\in\Omega_n} w_n(x,y)^p \left|\mathbf{u}(y)-\mathbf{u}(x)\right|^{p-2}(\mathbf{u}(y)-\mathbf{u}(x)) = 0 \\
\Leftrightarrow &\sum_{y:\mathbf{u}(x)\leq\mathbf{u}(y)} w_n(x,y)^p(\mathbf{u}(y)-\mathbf{u}(x))^{p-1} = \sum_{y:\mathbf{u}(x)>\mathbf{u}(y)} w_n(x,y)^p(\mathbf{u}(x)-\mathbf{u}(y))^{p-1}.
\end{aligned}$$

Taking the terms on the left and right hand side to the power of $1/p$ and then formally sending $p\to\infty$ then yields

$$\begin{aligned}
&\max_{y:\mathbf{u}(x)\leq\mathbf{u}(y)} w_n(x,y)(\mathbf{u}(y)-\mathbf{u}(x)) = \max_{y:\mathbf{u}(x)>\mathbf{u}(y)} w_n(x,y)(\mathbf{u}(x)-\mathbf{u}(y)) \\
\Leftrightarrow &\max_{y\in\Omega_n} w_n(x,y)(\mathbf{u}(y)-\mathbf{u}(x)) = -\min_{y\in\Omega_n} w_n(x,y)(\mathbf{u}(y)-\mathbf{u}(x)).
\end{aligned}$$

This calculation motivates the definition of the graph infinity Laplacian

$$\Delta_\infty^{w_n}\mathbf{u}(x) := \max_{y\in\Omega_n} w_n(x,y)(\mathbf{u}(y)-\mathbf{u}(x)) + \min_{y\in\Omega_n} w_n(x,y)(\mathbf{u}(y)-\mathbf{u}(x)),$$

which then allows to formulate the associated problem as an extension of Problem 2.10.

Problem 2.31 (Graph ∞-Laplacian). Given a weighted graph (Ω_n, w_n) and a labeling function $\mathbf{g}:\mathcal{O}_n\to\mathbb{R}$ with $\mathcal{O}_n\subset\Omega_n$, find a function $\mathbf{u}:\Omega_n\to\mathbb{R}$ such that

$$\begin{aligned}
\Delta_\infty^{w_n}\mathbf{u} &= 0, \text{ in } \Omega_n\setminus\mathcal{O}_n, \\
\mathbf{u} &= \mathbf{g} \text{ on } \mathcal{O}_n.
\end{aligned}$$

This problem is again well-posed, which is formulated in the following lemma.

Lemma 2.32. There exists a unique solution for Problem 2.31.

Relation between the Graph Lipschitz Extensions We now establish the connection between the different notions of Lipschitz extensions. Compared to the continuum case we do not establish the full equivalences but only the necessary implications required for the convergence proofs in [LIP-II].

First we see, that graph AMLEs are indeed special solutions of the basic Lipschitz extension problem on the graph.

Lemma 2.33. A graph AMLE is also a Lipschitz extension.

Proof. □

We now state the main result concerning the relation between the graph infinity Laplacian, graph AMLEs and comparison cones.

Lemma 2.34. Let (Ω_n, w_n) be a weighted connected graph and $\mathbf{g} : \mathcal{O}_n \to \mathbb{R}$ be a given function for $\mathcal{O}_n \subset \Omega_n$. Furthermore, let $\mathbf{u} : \Omega_n \to \mathbb{R}$ be graph infinity harmonic on $\Omega_n \setminus \mathcal{O}_n$ with boundary conditions given by $\mathbf{g}$, i.e., $\mathbf{u}$ solves Problem 2.31 then we have that

- $\mathbf{u}$ is an graph AMLE, i.e., $\mathbf{u}$ solves Problem 2.29,
- $\mathbf{u}$ fulfills comparison with cones.

Proof. Both of the stament are proven in [LIP-II]. From [LIP-II, Prop. 3.8] we have that $\mathbf{u}$ is an graph AMLE. Furthermore, form [LIP-II, Th. 3.2] we have that $\mathbf{u}$ fulfills comparison with cones. In fact, [LIP-II, Th. 3.2], shows a more refined statement, namely that

$$-\Delta_\infty^{w_n}\mathbf{u} \leq 0 \Rightarrow \mathbf{u} \text{ fulfills CDFA},$$
$$-\Delta_\infty^{w_n}\mathbf{u} \geq 0 \Rightarrow \mathbf{u} \text{ fulfills CDFB}.$$

□

2.3. Gamma Convergence: [LIP-I]

We are now in the situation to present the main results of [LIP-I; LIP-III].

The kernel

(K1) η is positive and continuous at 0,

(K2) η is non-increasing,

(K3) $\operatorname{supp}(\eta) \subset [0, t_\eta]$ for some $t_\eta > 0$.

Originally, the concept of Γ-convergence dates back to De Giorgi [DF75] as a type of variational convergence. We refer to [Bra02; Dal12] for a detailed overview on this notion and related topics. While Γ-convergence was successfully employed in a pure continuum setting for a longer time (see e.g. [Mod77]), it was more recently used to prove convergence form a discrete to a continuum functional [CGL10; BY12; VB+12]. The most relevant reference for this thesis was disruptive work presented by García Trillos and Slepčev in [GS15]. Here, the considered object was a graph total variation or generalized parameter, where the functional corresponds to $\mathbf{E}_p^{w_n}$ for $p = 1$ from Problem 2.8. Among other important ideas and notions, we want to highlight two ingredients that directly influenced [LIP-I]:

1) Γ-convergence on a common metric space, that allows to compare graph functions with continuum functions.

2) The proof strategy, discrete to non-local, non-local to continuum.

We review how theses concepts influence [LIP-I] in the following sections. The results in [GS15] were later transferred to the case $1 < p < \infty$ in [ST19] and in this sense [LIP-I] constitutes the generalization to $p = \infty$.

G Convergence in L infty

Γ-convergence We start with the basic definition of Γ-convergence.

Definition 2.35 (Γ-convergence). Let X be a metric space and let $F_n : X \to [-\infty, \infty]$ be a sequence of functionals. We say that F_n Γ-converges to the functional $F : X \to [-\infty, \infty]$ if

(i) **(liminf inequality)** for every sequence $(x)_{n\in\mathbb{N}} \subset X$ converging to $x \in X$ we have that

$$\liminf_{n\to\infty} F_n(x_n) \geq F(x);$$

(ii) **(limsup inequality)** for every $x \in X$ there exists a sequence $(x)_{n\in\mathbb{N}} \subset X$ converging to x and

$$\limsup_{n\to\infty} F_n(x_n) \leq F(x).$$

In order to show convergence of discrete functions defined on Ω_n to continuum functions acting on $\overline{\Omega}$, one needs to define a common metric space. [GS15] introduced the space TL^p

$$TL^p := \{(\mu, u) : \mu \in \mathcal{P}(\Omega), u \in L^p(\mu)\}$$

together with the transport distance

$$d_{TL^p}((\mu, u), (\nu, v)) = \inf_{\pi\in\Gamma(\mu,\nu)} \left(\int_{\Omega\times\Omega} |x - y|^p + |u(x) - v(y)|^p \, d\pi(x, y) \right)^{1/p}$$

where $\Gamma(\mu,\nu)$ is the set of coupling between μ and ν. A sequence (mu_n, u_n) converges to (μ, u) in TL^p iff there exists a sequence of transportation maps $T_n : \Omega \to \Omega$ with $T_n\#\mu = \mu_n$ and

$$\int_\Omega |x - T_n(x)|\, d\mu(x) \to 0$$

such that $u_n \circ T_n \xrightarrow{L^p} u$, [GS15, Prop. 3.12]. Therefore, the maps T_n allow to employ standard convergence in L^p. In order to transfer this situation to L^∞ one could try to employ a ∞-Wasserstein distance. However, as argued in [Roi21] the arguments do not transfer directly, since convergence in W^∞ does not metrize weak convergence of measures [San15, Thm. 5.10]. However, as seen in [LIP-I] one can employ a more direct argument. For problems in L^p the conservation of mass was important for the maps T_n (i.e. $T_n\#\mu = \mu_n$) such that the integrals could be transformed. This condition is irrelevant in L^∞, namely we have the following analogous transformation rule in L^∞.

Lemma 2.36 ([LIP-I, Lem. 2]). For two probability measures $\mu, \nu \in \mathcal{P}(\overline{\Omega})$, a measurable map $T : \Omega \to \Omega$ which fulfills

(i) $\nu << T\#\mu$,

(ii) $T\#\mu << \nu$,

and for a measurable function $u : \Omega \to \mathbb{R}$ we have that

$$\nu\text{-}\operatorname*{ess\,sup}_{x\in\Omega} u(x) = \mu\text{-}\operatorname*{ess\,sup}_{y\in\Omega} u(T(y)).$$

We want to compare the discrete measure $\mu_n = \frac{1}{n}\sum_{x\in\Omega_n} \delta_x$ to the target measure μ. In this setting a closest point projection $p_n : \Omega \to \Omega_n$

$$p_n(x) \in \operatorname{argmin}_{y\in\Omega_n} |x - y|$$

fulfills the assumption of Lemma 2.36. This allows us to extend the functional $\mathbf{E}_\infty^{w_n}$ in Problem 2.24 to L^∞ via

$$\mathbf{E}_\infty^{w_n}(u) = \begin{cases} \mathbf{E}_\infty^{w_n}(\mathbf{u}) & \text{if } u = \mathbf{u} \circ p_n, \text{ for some } \mathbf{u} : \Omega_n \to \mathbb{R}, \\ \infty & \text{else }, \end{cases} \tag{2.12}$$

which was similarly done in [GS15; ST19]. Additionally, we incorporate the constraint on $\mathcal{O}_n$ in Problem 2.24 via

$$\mathbf{E}_\infty^{w_n,\mathrm{cons}}(\mathbf{u}) := \begin{cases} \mathbf{E}(\mathbf{u}) & \text{if } \mathbf{u} = g \text{ on } \mathcal{O}_n, \\ \infty & \text{else}, \end{cases}$$

with the analogous extension to $Ł^\infty$ as in Eq. (2.12). This now allows us to state the first main result of [LIP-I].

Theorem 2.37 (Discrete to continuum Γ-convergence). Let $\Omega \subset \mathbb{R}^d$ be a domain satisfying **??**, let the kernel fulfill (K1)-(K3), and let the constraint sets $\mathcal{O}_n, \mathcal{O}$ satisfy **??**, then for any null sequence $(\varepsilon_n)_{n\in\mathbb{N}} \subset (0,\infty)$ which satisfies the scaling condition **??** we have

$$\mathbf{E}_\infty^{n,\mathrm{cons}} \xrightarrow{\Gamma} \sigma_\eta \, \mathcal{E}^{\mathrm{cons}}. \tag{2.13}$$

The main proof strategy here is similar to the one in [GS15; ST19]. Namely one defines the non-local functional

$$\mathcal{E}_\infty^\varepsilon(u) := \frac{1}{\varepsilon} \operatorname*{ess\,sup}_{x,y\in\Omega} \{\eta_\varepsilon(|x-y|)\,|u(x)-u(y)|\}, \quad \varepsilon > 0$$

for which we show that for any sequence $\varepsilon_n \to 0$ we have [LIP-I, Thm. 4]

$$\mathcal{E}_\infty^{\varepsilon_n} \xrightarrow{\Gamma} \sigma_\eta \, \mathcal{E}_\infty.$$

For the liminf inequality of the discrete functionals, one has to take special care of points $x, y \in \Omega_n$ where $\eta_{\varepsilon_n}(|p_n(x) - p_n(y)|) = 0$. We want to bound $\mathbf{E}_\infty^{w_n}$ from below by $\mathcal{E}_\infty^{\varepsilon_n}$ for which we have to permit significant communication of x and y whenever $p_n(x)$ and $p_n(y)$ do not communicate. This can be done, (temporarily assuming η is constant on $[0,t]$) by introducing a smaller length scale $\tilde{\varepsilon}$ such that

(i) $|p_n(x) - p_n(y)| > t\varepsilon_n \Rightarrow |p_n(x) - p_n(y)|/\varepsilon_n < |x-y|/\tilde{\varepsilon}$,

(ii) $\lim_{n\to\infty} \tilde{\varepsilon}/\varepsilon = 1$.

As shown in [LIP-I] the choice $\tilde{\varepsilon} = \varepsilon - 2\delta/t$ fulfills (i). So in order to fulfill (ii) we obtain the scaling condition

$$\lim_{n\to\infty} \frac{\delta_n}{\varepsilon_n} = 0 \Rightarrow \lim_{n\to\infty} \frac{\varepsilon_n - 2\delta_n/t}{\varepsilon_n} = 1 - \frac{2}{t} \lim_{n\to\infty} \frac{\delta_n}{\varepsilon_n} = 1.$$

This then allows to prove the liminf inequality since $\mathbf{E}_\infty^{w_n}(u_n) \geq \frac{\tilde{\varepsilon}_n}{\varepsilon_n} \mathcal{E}_\infty^{\tilde{\varepsilon}_n}(u_n)$ holds. The limsup inequality can then be shown by choosing the constant sequence, with some additional care for the changing constraint set $\mathcal{O}_n$.

Convergence of Minimizers The convenient aspect of Γ-convergence is, that under additional compactness properties it directly shows convergence of minimizers, [Bra02, Thm. 8]. This yields the second main result in [LIP-I].

Theorem 2.38 ([LIP-I, Thm. 2]). Let $\Omega \subset R^d$ be a domain satisfying **??**, let the kernel fulfil (K1)-(K3), let the constraint sets $\mathcal{O}_n, \mathcal{O}$ satisfy **??**, and $(\varepsilon_n)_{n\in\mathbb{N}} \subset (0,\infty)$ be a null sequence which satisfies the scaling condition **??**. Then any sequence

$(u)_{n\in\mathbb{N}} \subset L^\infty(\Omega)$ such that

$$\lim_{n\to\infty} \left(\mathbf{E}_\infty^{n,\mathrm{cons}}(u_n) - \inf_{u\in L^\infty(\Omega)} \mathbf{E}_\infty^{n,\mathrm{cons}}(u) \right) = 0$$

is relatively compact in $L^\infty(\Omega)$ and

$$\lim_{n\to\infty} \mathbf{E}_\infty^{n,\mathrm{cons}}(u_n) = \min_{u\in L^\infty(\Omega)} \sigma_\eta \, \mathcal{E}_\infty^{\mathrm{cons}}(u).$$

Furthermore, every cluster point of $(u_n)_{n\in\mathbb{N}}$ is a minimizer of $\mathcal{E}_{\mathrm{cons}}$.

In order to show the compactness in the above theorem one employs the following lemma.

Lemma 2.39 ([LIP-I, Lem. 4]). Let (Ω, μ) be a finite measure space and $K \subset L^\infty(\Omega;\mu)$ be a bounded set w.r.t. $\|\cdot\|_{L^\infty(\Omega;\mu)}$ such that for every $\varepsilon > 0$ there exists a finite partition $\{V_i\}_{i=1}^n$ of Ω into subsets V_i with positive and finite measure such that

$$\mu\text{-}\operatorname*{ess\,sup}_{x,y\in V_i} |u(x) - u(y)| < \varepsilon \ \forall u \in K, i = 1, \dots, n, \tag{2.14}$$

then K is relatively compact.

Remark 2.40. The proof of this statement employs ideas from [DS88, Lem. IV.5.4] and appeared similarly in TR's master thesis. However, therein the statement was slightly wrong, which was corrected in [LIP-I]. △

This lemma is the used to show that if a sequence u_n fulfills

$$\sup_{n\in\mathbb{N}} \mathbf{E}_\infty^{w_n,\mathrm{cons}}(u_n) < \infty$$

then it is relatively compact.

Application to ground states Ground states, no rates.

2.4. Uniform Convergence

2.5. Ratio Convergence

Chapter 3

Robust and Sparse Supervised Learning

In this chapter we present the topics discussed in the works [CLIP; FNO; BREG-I] which are reprinted in Part II. Compared to the previous chapter we are now in the setting of supervised learning as described in Section 1.2. As already mentioned above, we focus on neural networks $f_\theta : \mathcal{X} \to \mathcal{Y}$ parameterized by $\theta \in \Theta$. Here, our two main question arise, namely:

- **Input robustness**: how robust is $x \mapsto f_\theta(x)$ w.r.t. input perturbation?
- **Parameter sparsity**: how can we obtain sparse parameters $\theta \in \Theta$?

Section 3.1: Setting

Section 3.2: [CLIP] | Section 3.3: [BREG-I] | Section 3.4: [FNO]

Therefore, conceptually we again highlight the keyword **sparsity** and additionally **robustness**. In [CLIP] we consider input robustness under adversarial perturbations. Here, the input is typically *attacked* on purpose to confuse a neural network and therefore worsen its performance. In order to obtain a neural network that is less vulnerable against such attacks we propose an optimization strategy that selects parameters that yield a more robust network. In Section 3.2 we comment on the topic and the contribution in more detail.

A different kind of input robustness is considered in [FNO]. In the setting of image classification, images are modeled as functions on a continuum domain that need to be discretized in order to represent the on a machine. This discretization however, is usually arbitrary and not inherent to the object of interest. Therefore, it is natural to assume that the output of the network should be independent of this discretization, also referred to as resolution, for which we consider robustness w.r.t. resolution changes. We remark on this and the publication in Section 3.4.

Concerning computational performance and memory storage of the neural network we focus on sparsity of the parameters $\theta \in \Theta$. In [BREG-I] we propose a sparse optimization strategy based on Bregman iterations, which employs L^1 type penalities to promote sparsity. The conceptual difference between the existing algorithm and our proposal is the stochastic computation of the gradient. Our theoretical results prove decay in loss and convergence of the iterates. Finally we also conduct numerical experiments that demonstrate the efficiency of the method. We refer to Section 3.3 for a detailed explanation.

Before, we start with the exposition on the mentioned works, we briefly review the common supervised learning framework. Building upon the basic observations in Chapter 3 we give a slightly more detailed introduction in Section 3.1.

3.1. Setting

We are given a finite training set $\mathcal{T} \subset \mathcal{X} \times \mathcal{Y}$. For a family of functions $f_\theta : \mathcal{X} \to \mathcal{Y}$ parameterized by $\theta \in \Theta$. We consider the empirical minimization

$$\min_{\theta \in \theta} \mathcal{L}(\theta)$$

where for a function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ we define

$$\mathcal{L}(\theta) := \frac{1}{|\mathcal{T}|} \sum_{(x,y)\in\mathcal{T}} \ell(f_\theta(x), y). \tag{3.1}$$

Remark 3.1. Assuming that $\mathcal{T}$ is sampled from a joint distribution $P_{\mathcal{X},\mathcal{Y}}$ on $\mathcal{X} \times \mathcal{Y}$ this approximates the computational infeasible population risk minimization

$$\int_{\mathcal{X}\times\mathcal{Y}} \ell(f_\theta(x), y) d\pi(x, y).$$

△

In the following we provide two important choices for the function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$.

Example 3.2 (MLE). For image denoising problems, we often choose $\mathcal{X} = \mathcal{Y} = [0, 1]^{N\times M}$ assuming only one color channel for simplicity. In this context the Mean squared L^2 Error (MLE), defined as

$$\ell(\bar{y}, y) := \frac{1}{N \cdot M} \|\bar{y} - y\|^2$$

is commonly employed. This loss function could however also be employed for classification problems.

Example 3.3 (Cross-Entropy). For classification problems the function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is often chosen as the *cross-entropy* or *negative log-likelihood* loss, [Goo52].

For two discrete probability distributions, $p, q : \{1, \dots, C\} \to \mathbb{R}$ one defines

$$H(p, q) := -\sum_{c=1}^{C} p_c \cdot \log(q_c)$$

see e.g. [COR98]. Assuming that $\mathcal{Y} = \Delta^C$ this allows to choose $\ell(\bar{y}, y) := H(y, \bar{y})$. If the network only maps to $\mathbb{R}^C$ one often additionally inserts a soft-max function $Q(y)_c := \exp(y_c)/\sum_c \exp(y_c)$ (see [Bol68]) and then sets

$$\ell(\bar{y}, y) := H(y, Q(\bar{y})).$$

In the case, where the output is given as labels $y \in \{1, \dots, C\}$ on sets

$$\ell(\bar{y}, y) := H(y_{\text{oh}}, Q(\bar{y})) = -\log(Q(\bar{y}))$$

employing the one-hot notation. Namely, we define $\text{oh} : \{1, \dots, C\} \to \Delta^C$,

$$(y_{\text{oh}})_i := \text{oh}(y)_c := \begin{cases} 1 & \text{if } c = y, \\ 0 & \text{else .} \end{cases}$$

3.1.1. Network Architectures

In this thesis we focus on feed-forward neural networks, i.e., we consider layers of the form

$$\Phi(w, W, b)(z) := wz + \sigma(Wz + b) \tag{3.2}$$

where $w \in \mathbb{R}$ models a residual connection, $W \in \mathbb{R}^{n \times n}$ is a weight matrix, $b \in \mathbb{R}^n$ a bias vector and $z \in \mathbb{R}^m$. We consider a concatenation of $L \in \mathbb{N}$ such layers, which then forms a neural network

$$f_\theta = \Phi_L \circ \dots \circ \Phi_1$$

with parameters $\theta = ((W_1, b_1, w_1) \dots, (W_L, b_L, w_L)) \in \Theta$ and layers $\Phi^i := \Phi(w_i, W_i, b_i)$.

MLP In the easiest case we consider a perceptron [Ros58], which models a fully connected layer, i.e. every entry W_{ij} of th weight matrix is an parameter that is optimized in the training process.

Convolutions Especially important for visual tasks are convolutional layers. Here, we take a kernel $k \in \mathbb{R}^{M \times M}$ and define the application of $W = W(k)$ as

$$Wz = k * z$$

where we refer to Section 3.4 for the concrete definition of the convolution. Typically the input is of the form $z \in \mathbb{R}^{K:\text{in}\times N\times M}$, where K_{in} denotes the number of input channels. The layer is then a mapping $\Phi : \mathbb{R}^{K:\text{in}\times N\times M} \to \mathbb{R}^{K:\text{out}\times N\times M}$, where K_{out} denotes the number of output channels, which can be realized by different kernels k_{ij} and

$$(Wz)_{j,:,:} := \sum_{i=1}^{K_{\text{in}}} k_{ij} * z.$$

ResNets Often we also consider a residual component as displayed in Eq. (3.2) with the term wx. The idea of adding this component was first introduced in [SGS15] with a learnable parameter $w \in \mathbb{R}$ and later popularized in [He+16a] by fixing $w = 1$, see also [He+16b], which then yields the celebrated ResNet architecture. In the following applications we both consider the case where $w = 1$ is fixed, but also the possibility of learning the parameter $w \in \mathbb{R}$ in [BREG-II].

3.1.2. Gradient Computation and Stochastic Gradient Descent

Training a neural network requires to solve a optimization problem w.r.t. to the parameters $\theta \in \Theta$. In this work we only focus on first order methods, however both zero [Rie22; Pin+17; Car+21; Mar10] and second order methods [Mar10] have been successfully applied in this context. Employing first order methods, requires to evaluate the gradient $\nabla_\theta \mathcal{L}$, however in this scenario it is not common to compute the full gradient but rather to have a gradient estimator. This estimator is usually obtained by randomly dividing the train set $\mathcal{T}$ into disjoint minibatches $B_1 \cup \ldots \cup B_b = \mathcal{T}$ and then successively computing the gradient of the minibatch loss

$$\mathcal{L}(\theta; B) := \frac{1}{|B_i|} \sum_{(x,y)\in B_i} \ell(f_\theta(x), y).$$

Iterating over all batches $i = 1, \ldots, b$ is referred to as one epoch. From a mathematical point of view this yields stochastic optimization methods, since in each step the true gradient is replaced by an estimator. In the abstract setting we let $(\Omega, F, \mathbb{P})$ be a probability space and consider a function $g : \Theta \times \Omega \to \Theta$ as an unbiased estimator of $\nabla\mathcal{L}$, i.e.

$$\mathbb{E}\left[g(\theta; \omega)\right] = \nabla\mathcal{L}(\theta) \text{ for all } \theta \in \Theta.$$

Most notably this method transforms the standard gradient descent update [Cau+47]

$$\theta^{(k+1)} = \theta^{(k)} - \tau^{(k)}\nabla\mathcal{L}(\theta^{(k)})$$

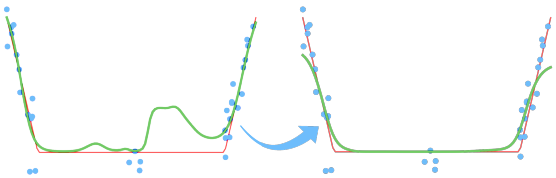
to *stochastic* gradient descent [RM51]

$$\begin{aligned} &\text{draw } \omega^{(k)} \text{ from } \Omega \text{ using the law of } \mathbb{P}, \\ &g^{(k)} := g(\theta^{(k)}; \omega^{(k)}), \\ &\theta^{(k+1)} := \theta^{(k)} - \tau^{(k)} g^{(k)}. \end{aligned}$$

3.2. Adversarial Stability via Lipschitz Training: [CLIP]

In the following we consider classification problems with $C \in \mathbb{N}$ different classes and functions $f : \mathcal{X} \to \Delta^C$ for which we denote by

$$f^{\mathrm{MAP}}(x) := \operatorname{argmax}_c f(x)_c$$



the maximum a posteriori estimation for input $x \in \mathcal{X}$. The capabilities of neural networks to perform classification tasks on unseen data—i.e. inputs $x \in \mathcal{X}$ that are not part of the training data—are impressive and the reason for their popularity. However, it has been noticed in [GSS14] that is relatively easy to "fool" classification networks in the following sense:

Given a classification network and a human classifier $f_\theta, h : \mathcal{X} \to \Delta^C$, and an input $x \in \mathcal{X}$ such that $f_\theta^{\mathrm{MAP}}(x) = h^{\mathrm{MAP}}(x)$. An adversarial example is an input $\bar{x} \in \mathcal{X}$ that is close to x and

$$f_\theta(\bar{x}) \neq h(\bar{x}) = h(x).$$

The vague concept of a *human classifier*, bears the intuition of the classification problem as a human would solve it. Assuming that we are given data $\mathcal{T} \subset \mathcal{X} \times \mathcal{Y}$ that is samples i.i.d. from a joint distribution $P_{\mathcal{X},\mathcal{Y}}$ this function could also be chosen as $h(x)_c := P_{\mathcal{X},\mathcal{Y}}(c|x)$.

Remark 3.4. While we mostly focus on neural networks, some authors argue that these instabilities are inherent to certain classification problems [Sha+18; FFF18]. From this point of view, trying to defend against these instabilities is not necessarily desirable. △

What are Adversarial Examples Formally In order to formalize this idea we omit any influence of the typically unavailable function h and instead consider $\bar{x}$ adversarial if $f_\theta(x) \neq f_\theta(\bar{x})$, i.e. we assume $f_\theta(x)$ to be "correct". Furthermore, in this work we assume that $\mathcal{X}$ models an image space $\mathcal{X} = \mathbb{R}^{K \times N \times M}$, with $K, N, M \in \mathbb{N}$ and choose a distance $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_0^+$ to measure the distance between points in $\mathcal{X}$.

Remark 3.5. In most of our examples we choose $d(\cdot, \cdot) = \|\cdot - \cdot\|_p$. Employing a L^p norm in an image context—via pixel-wise comparison—can be an unfavorable choice. Images that appear very different visually, (i.e. for a human classifier) can have a smaller L^p distance, than images that visually appear similar, see e.g. [Sta+21, Fig. 16]. However, it is easy tp evaluate, which is crucial for most applications. In the absence of a better criterion, having a small L^p norm, is commonly the only way to decide if $\bar{x}$ is an admissible adversarial example. △

Employing these simplifications we then say that $\bar{x} \in \mathcal{X}$ is adversarial, if

$$d(x, \bar{x}) \leq \varepsilon \qquad \text{and} \qquad f_\theta(x) \neq f_\theta(\bar{x}),$$

where ε is called the *adversarial budget*. This parameter This interpretation only makes sense, if we assume that $f_\theta(x)$ is *correct*, which can only be easily checked on the training data [Bun+23].

Types of Adversarial Examples Typically, adversarial examples are created from the clean image x via some distortion $\delta \in \mathbb{R}^n$. Together with an application map $T : \mathcal{X} \times \mathbb{R}^s \to \mathcal{X}$ one then obtains $\bar{x} = T(x, \delta)$ as the adversarial example. In this formulation one can the alternatively employ the criterion $\|\delta\| \leq \varepsilon$ to decide, whether $T(x, \delta)$ is an adversarial example. We only list some of the approaches below:

- **Addition**: the most well-known examples are created by $T(x, \delta) := x + \delta$, i.e. $\bar{x} = x + \delta$. Here, it is important to note that typically images are assumed to have values between 0 and 1, i.e. $\mathcal{X} = [0, 1]^{K \times N \times M}$ for which it is important to ensure that $\bar{x} \in \mathcal{X}$.
- **Translation and Rotation**: Simple geometric transformations—that would be unnoticeable for a human classifier—are quite effective to fool neural networks. Employing translations $T_{\mathrm{t}} : \mathcal{X} \times \mathcal{X} \to \mathcal{X}$ one has to choose the behavior on the boundary such that one obtains a valid image. The same holds true for rotations $T_{\mathrm{r}} : \mathcal{X} \times [-\pi, \pi] \to \mathcal{X}$. In this case $\delta \in [-\pi, \pi]$ models the angle of the rotation and would yield an admissible adversarial example, if $|\delta| \leq \varepsilon$. Here, we see that this formulation looses some expressivity:
  - On the MNIST classification task, we see that only $\varepsilon < \pi/2$ makes sense. Otherwise the number "6" could be always transformed to the number "9" and vice versa.
  - Considering the number "0" rotations above the angle $\pi/2$ can definitely yield proper adversarial examples $\bar{x}$.

  We refer to [Eng+18] for a study on these types of adversarial examples.
- **Change of Basis**: As explored in [Guo+17] one can consider a different orthonormal basis of the image space $\mathbb{R}^{K \times N \times M}$ and then perform the attack w.r.t. this basis. The transform $T : \mathcal{X} \times \mathbb{R}^{K \times N \times M} \to \mathcal{X}$ first obtains the different representation of x, then adds the coefficients of δ and then maps back to the original basis. This is only meaningful, if one restricts certain coefficients of δ to be zero in the alternative basis. For example, the discrete cosine transform ([ANR74]) has been applied successfully in this context [Guo+17].

In the following we restrict ourselves to the additive case and only consider examples $\bar{x} = x + \delta$.

Finding Adversarial Examples Employing a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ the task of finding an adversarial examples $\bar{x} = x + \delta \in \mathcal{X}$ can be relaxed via solving the problem

$$\max_{\bar{x} \in \mathcal{X} : d(x, \bar{x}) \leq \varepsilon} \ell(y, f_\theta(\bar{x})) = \max_{\delta : \|\delta\|_d \leq \delta, x + \delta \in \mathcal{X}} \ell(y, f_\theta(x + \delta), \tag{3.3}$$

where $y = f_\theta(x)$. Solving this problem is referred to as *attacking* the network f. More precisely, this is a so-called *untargeted* attacks, since we do not prescribe $\bar{x}$ to realize any special output as long as it confuses the network. Opposed to this, there are so called *targeted* attacks. Here we pick $c^{\text{adv}} \neq f_\theta^{\text{MAP}}(x)$ and then want to find $\bar{x}$ such that $f_\theta^{\text{MAP}}(\bar{x}) = c^{\text{adv}}$. Here, we then consider the problem

$$\min_{\bar{x} \in \mathcal{X} : d(x,\bar{x}) \leq \varepsilon} \ell(c_{\text{oh}}^{\text{adv}}, f_\theta(\bar{x})).$$

However, we see that conceptually this problem is similar to Eq. (3.3) and differs only by a change of sign and a different reference label. A popular method to solve Eq. (3.3) is the projected sign gradient ascent iteration [KGB+16],

$$\bar{x}^{(k+1)} = \text{Proj}_d(\bar{x}^{(k)} + \tau \, \text{sign}(\nabla_{\bar{x}} \ell(f_\theta(x), f_\theta(\bar{x}^{(k)})))).$$

Here, Proj_d denotes the projection onto the set $\mathcal{X} \cap B_{d,\varepsilon}(x)$, where $B_{d,\varepsilon}(x)$ is the ball with radius ε around x, w.r.t. to the distance d. Performing only one step of this iteration yields the fast gradient sign method [GSS14]. There is a wide variety of these so-called gradient-based white-box attacks [Yua+19], i.e. methods that assume that the gradient of the model is available. In more realistic scenarios, this might not be the case. Attacks that do not employ the actual gradient of the model to attack are called *black box* attacks [Ily+18], which however not part of this thesis.

Defending Against Adversarial Examples We consider the question of finding parameters $\theta \in \Theta$ such that corresponding model f_θ is *adversarially robust*, i.e. is less vulnerable to attacks. Therefore, we want the attack problem in Eq. (3.3) to be hard to solve. This intuition leads to the optimization problem

$$\min_{\theta \in \Theta} \sum_{(x,y) \in \mathcal{T}} \max_{\bar{x} \in B_{d,\varepsilon}(x)} \ell(f_\theta(\bar{x}), y)$$

which is known as the *adversarial training* formulation [KGB16; Mad+17]. From a distributionally robust optimization point of view, this problem relates to

$$\min_{\theta \in \Theta} \max_{\tilde{P} : D(P_{\mathcal{X},\mathcal{Y}}, \tilde{P}) \leq \varepsilon} \int_{\mathcal{X} \times \mathcal{Y}} \ell(f_\theta(x), y) \; d\tilde{P}(x,y)$$

where D denotes some distance on probability distributions, see [BGM23]. Employing a batched gradient descent type iteration, one then has to solve a problem of the form Equation (3.3) in every step, which can be interpreted as a form of data augmentation [LeC+95].

Lipschitz Training of Neural Networks Adversarial robustness is closely related to the Lipschitzness of a network f_θ. Namely, for inputs $x, \bar{x} \in \mathcal{X}$ that are close, we also want the outputs of f_θ to be close. In other words we want to find a small constant $L > 0$ such that

$$\|f_\theta(x) - f_\theta(\bar{x})\|_{\mathcal{Y}} \leq L \; \|x - \bar{x}\|_{\mathcal{X}},$$

where we typically employ an L^p norm both $\|\cdot\|_{\mathcal{X}}$ and $\|\cdot\|_{\mathcal{Y}}$. The smallest constant fulfilling this inequality is the Lipschitz constant $\mathrm{Lip}(f_\theta)$. Having some control over this constant can therefore relate to being adversarially robust.

Remark 3.6. Apart from adversarial robustness, having an upper bound on the Lipschitz constant of a neural network is also important for other applications, see e.g. [Has+20; ACB17]. △

We employ the Lipschitz constant as a regularizer and consider the problem

$$\min_\theta \mathcal{L}(\theta) + \lambda \ \mathrm{Lip}(f_\theta) \tag{3.4}$$

for a parameter λ. Unfortunately, computing the Lipschitz constant of neural networks is a NP-hard problem [SV18] for which many works employ the estimate

$$\mathrm{Lip}(f_\theta) \ \leq \ \prod_{l=1}^{L} \mathrm{Lip}(\Phi_l) \leq C_\sigma \prod_{l=1}^{L} \|W_l\|\,, \tag{3.5}$$

where here $\|W_l\|$ denotes some matrix norm and C_σ depends on the Lipschitz constants of the activation functions, see [ALG19; Gou+20; KMP20; RKH19]. This inequality is not sharp and usually overestimates the Lipschitz constant, as we see in the following example taken from [CLIP].

Example 3.7. We consider a feed-forward neural network $f_\theta : \mathbb{R} \to \mathbb{R}$ with one hidden layer,

$$\begin{aligned}
\Phi_1(z) &:= \mathrm{ReLU}(W_1 z) := (\max\{z, 0\}, \max\{-z, 0\})^T\,, \qquad W_1 := (1, -1),\\
\Phi_2(z) &:= W_2 z := z_1 + z_2, \qquad W_2 := (1, 1)^T,\\
f_\theta &:= \Phi_2 \circ \Phi_1.
\end{aligned}$$

For $x \in \mathbb{R}$ we have that

$$\begin{aligned}
x \geq 0 \quad &\Rightarrow \Phi_1(x) = (x, 0)^T \quad &&\Rightarrow f_\theta(x) = x,\\
x \leq 0 \quad &\Rightarrow \Phi_1(x) = (0, -x)^T \quad &&\Rightarrow f_\theta(x) = -x,
\end{aligned}$$

and therefore $f_\theta = |\cdot|$, for which we know that $\mathrm{Lip}(f_\theta) = 1$. However employing the spectral for W_1 and W_2, we see that

$$\|W_1\| \cdot \|W_2\| = \sqrt{2}\ \sqrt{2} = 2.$$

Plugging the estimate of Eq. (3.5) into Eq. (3.4) therefore potentially over regularizes the problem.

Contribution in [CLIP] In [CLIP] we propose a strategy to solve Eq. (3.4) approximately, without the estimate in Eq. (3.5). The basic idea consists of approximating the

Lipschitz constant on a finite set and using this approximation as a regularizer. Furthermore, we analyse the original model in Eq. (3.4) where we study existence of solutions, the influence of the parameter λ and the limits $\lambda \to 0, \lambda \to \infty$, see Section 3.2.2. Finally, we demonstrate the efficiency of the methods by applying to some simple toy problems, see Section 3.2.3.

3.2.1. Cheap Lipschitz Training

As mentioned before, we approximate the Lipschitz constant on a finite set $\mathcal{X}_{\text{Lip}} \subset \mathcal{X} \times \mathcal{X}$ via

$$\text{Lip}(f_\theta; \mathcal{X}_{\text{Lip}}) := \max_{(x,\bar{x}) \in \mathcal{X}_{\text{Lip}}} \frac{\|f_\theta(x) - f_\theta(\bar{x})\|_{\mathcal{Y}}}{\|x - \bar{x}\|_{\mathcal{X}}} \approx \text{Lip}(f_\theta).$$

Disregarding the non-differentiable points of $\theta \mapsto \text{Lip}(f_\theta; \mathcal{X}_{\text{Lip}})$ allows us to make the above approximation in Eq. (3.4) and the solve then problem via a gradient descent variant.

Remark 3.8. Let $f, g : \mathcal{X} \to \mathcal{Y}$ be two differentiable functions. If $f(\bar{x}) > g(\bar{x})$ for some $\bar{x} \in \mathcal{X}$ then we know that there exists some $\epsilon > 0$ such that $f(x) > g(x) \forall x \in B_\epsilon(\bar{x})$. We have that $f \vee g = \max\{f, g\} = f$ in $B_\epsilon(\bar{x})$ and therefore $x \mapsto f(x) \vee g(x)$ is differentiable in $\bar{x}$. If $f(\bar{x}) = g(\bar{x})$ then $f \vee g$ could be non-differentiable at $\bar{x}$. However, in practice we employ automatic differentiation, where in this case one of the functions is chosen, say f, and the *gradient* at $\bar{x}$ is computed as ∇f. △

The strength of this approach of course is dependent on the quality of the set $\mathcal{X}_{\text{Lip}}$. In [CLIP] we propose to iteratively update the set via a gradient ascent type scheme. Namely, we initialize $\mathcal{X}_{\text{Lip}}$ as a random perturbation of a subset of the given data. In each step we then update the points as follows:

- Consider $L(x, \bar{x}) := \|f_\theta(x) - f_\theta(\bar{x})\| \,/\, \|x - \bar{x}\|, \quad x, \bar{x} \in \mathcal{X}$.
- For each $(x, \bar{x}) \in \mathcal{X}_{\text{Lip}}$ perform the update

$$\begin{aligned} x &\leftarrow x + \tau\, L(x, \bar{x}) \nabla_x L(x, \bar{x}), \\ \bar{x} &\leftarrow \bar{x} + \tau\, L(x, \bar{x}) \nabla_{\bar{x}} L(x, \bar{x}), \end{aligned}$$

  for a parameter $\tau > 0$.

This scheme performs a gradient ascent on the Lipschitz constant, see [CLIP, Alg. 1]. For each mini batch $B \subset \mathcal{T}$ we first update the set $\mathcal{X}_{\text{Lip}}$ and then update the parameters via

$$\theta \leftarrow \theta - \eta\, \nabla_\theta \left(\mathcal{L}(\theta; B) + \lambda\ \text{Lip}(f_\theta, \mathcal{X}_{\text{Lip}})\right)$$

which yields the algorithm presented in [CLIP, Alg. 2]. Similar to [Sha+19] one can reuse the gradients computed for ∇_x for the computation of ∇_θ. This fact yields the attribute *cheap* in the Lipschitz training algorithm.

3.2.2. Analysis of Lipschitz Regularization

In [CLIP] we analyse some basic properties of the original regularization problem in Eq. (3.4). We repeat the assumptions we made therein, in order to refer to them in the following.

Assumption 1. We assume that the loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R} \cup \{\infty\}$ satisfies:

a) $\ell(y, \bar{y}) \geq 0$ for all $y, \bar{y} \in \mathcal{Y}$,

b) $y \mapsto \ell(y, \bar{y})$ is lower semi-continuous for all $\bar{y} \in \mathcal{Y}$.

Assumption 2. We assume that the map $\theta \mapsto f_\theta(x)$ is continuous for all $x \in \mathcal{X}$.

Assumption 3. We assume that there exists $\theta \in \Theta$ such that

$$\frac{1}{|\mathcal{T}|} \sum_{(x,y)\in\mathcal{T}} \ell(f_\theta(x), y) + \lambda \operatorname{Lip}(f_\theta) < \infty.$$

Employing only Assumption 2 one can show that that the map $\theta \mapsto \operatorname{Lip}(f_\theta)$ is lower semi-continuous, [CLIP, Lem. 1].

Existence If Θ is compact or finite, one can show that there exist solutions of (3.4). In the general case one needs to add a norm term, that ensures boundedness of a minimizing sequence. The following is the main existence result, which can be proven by the direct method in the calculus of variations [Dac07].

Proposition 3.1 ([CLIP, Prop. 1]). Under Assumptions 1 to 3 the problem

$$\min_{\theta\in\Theta} \frac{1}{|\mathcal{T}|} \sum_{(x,y)\in\mathcal{T}} \ell(f_\theta(x), y) + \lambda \operatorname{Lip}(f_\theta) + \mu \|\theta\|_\Theta$$

has a solution for all values $\lambda, \mu > 0$. Here, $\|\cdot\|_\Theta$ denotes a norm on Θ.

Dependency on the Regularization Parameter Assuming that for every $\lambda > 0$ we have a solution $\theta_\lambda \in \Theta$ of Eq. (3.4) one can show that

$$\begin{aligned} &\lambda \longmapsto \frac{1}{|\mathcal{T}|} \sum_{(x,y)\in\mathcal{T}} \ell(f_{\theta_\lambda}(x), y) && \text{is non-decreasing,} \\ &\lambda \longmapsto \operatorname{Lip}(f_{\theta_\lambda}) && \text{is non-increasing.} \end{aligned}$$

This statement is the content of [CLIP, Prop. 2], however, the argument is exactly the same as in [BO13]. The intuition behind this result is that with increasing parameter λ solutions of Eq. (3.4)—more precisely the corresponding networks—have smaller

Lipschitz constants, i.e. tend to be more constant, which however also diminishes their expressivity. We formalize the limit cases in the following.

Assuming the realizability condition of [SB14] we know that there exists parameters $\theta \in \Theta$ such that $\mathcal{L}(\theta) = 0$. This means there are parameters that fit the data perfectly. Considering the limit $\lambda \to 0$ we obtain convergence to a solution of the unregularized problem with the smallest Lipschitz constant.

Proposition 3.2 ([CLIP, Prop. 3]). Let Assumptions 1 to 3 and the realizability assumption [SB14] be satisfied. If $\theta_\lambda \to \theta^\dagger \in \Theta$ as $\lambda \searrow 0$, then

$$\theta^\dagger \in \operatorname{argmin} \left\{ \operatorname{Lip}(f_\theta) \,:\, \theta \in \Theta,\, \frac{1}{|\mathcal{T}|} \sum_{(x,y)\in\mathcal{T}} \ell(f_\theta(x), y) = 0 \right\}$$

if this problem admits a solution with $\operatorname{Lip}(f_\theta) < \infty$.

Furthermore, we study the effect of sending $\lambda \to \infty$, where we see that the network f_{θ_λ} indeed tends to be constant as expected. We can explicitly characterize this constant as the closest point to barycenter of the output data, that is realizable by a neural network.

Proposition 3.3 ([CLIP, Prop. 4]). Let Assumptions 1 to 3 be satisfied and assume that

$$\mathcal{M} := \{ y \in \mathcal{Y} \,:\, \exists \theta \in \Theta,\, f_\theta(x) = y,\, \forall x \in \mathcal{X} \} \neq \emptyset.$$

If $\theta_\lambda \to \theta_\infty \in \Theta$ as $\lambda \to \infty$, then $f_{\theta_\infty}(x) = \hat{y}$ for all $x \in \mathcal{X}$ where

$$\hat{y} \in \operatorname{argmin}_{y' \in \mathcal{M}} \frac{1}{|\mathcal{T}|} \sum_{y \in \mathcal{T}_\mathcal{Y}} \ell(y', y).$$

3.2.3. Numerical Results

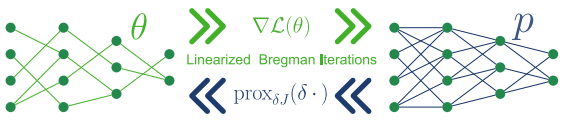
We briefly comment on the numerical results [CLIP]. All the experiments were implemented in `Python` [VD95] employing—among others—the `PyTorch` [Pas+19] package. We conduct experiments on the MNIST [LC10] and FashionMNIST [XRV17] datasets.

Qualitative Example We first apply the CLIP scheme to regularize a one-dimensional regression problem. In Fig.2 therein, one observes can observe qualitatively the effect of the regularization. Namely, we are given noisy data from a ground truth function for which the unregularized problem tend to overfit the data. The resulting network has fluctuations and therefore a large Lipschitz constant in certain regions, where the ground truth function has a small Lipschitz constant. With increasing parameter λ we observe that the networks f_{θ_λ} are smoothed out and better approximate the ground truth function.

The code for all the experiments is available at github.com/TimRoith/CLIP.

Quantitative Evaluation of Adversarial Robustness We additionally evaluate how robust a CLIP trained network is against adversarial attacks. Next to a standard learning rate scheduler, we also update the regularization parameter λ throughout the training process. Namely, we choose a target accuracy be and evaluate the loss on a validation set after the epoch. If this accuracy is less then the target we decrease λ and increase it if the accuracy is higher. In Table 1 of [CLIP] we compare ourselves against a weight regularization scheme that is based on the estimate in Eq. (3.5). We see that CLIP outperforms this method in most cases.

3.3. Sparsity via Bregman Iterations: [BREG-I]



Starting from the first simple neural architectures (e.g. [Ros58]) the size of typical architectures has grown significantly in the last years [Hoe+21]. While this development allowed to solve increasingly difficult tasks with neural networks, it also lead to immense computational requirements, both for the training and evaluation of the net. Therefore it seems natural, how the weights of neural networks can be compressed or more general how evaluation and training can be made more efficient. Some of the approaches here include (see [Gho+21] for a comprehensive overview):

- **Architecture Optimization**: Designing an architecture that can be trained to have the same performance as a comparable one, while having less parameters, e.g. [EMH19; How+17].
- **Quantization**: Lowering the precision of the machine numbers of the parameters, e.g. [Ban+18; CBD14].
- **Knowledge distillation**: Employing a trained larger network to learn a smaller network in a student-teacher approach, e.g. [Sch92; HVD15].
- **Pruning**: Parameters with small saliency, i.e., parameters that contribute little to the effective output of the network are removed or set to zero, in order to obtain a sparse weight matrix or a smaller architecture, see e.g. [LDS89; HSW93].

From the above methods, the pruning approach is most influential to our work in [BREG-I]. Namely, the concept of compressing the neural network by sparsifying the weight matrices is similarly employed. However, there are a few deviations from the classical pruning framework, which we remark in the following.

Sparsity via Optimization In [LDS89; HSW93] one assumes to be given a trained neural network, which is then compressed after training based on some criterion. The authors in [CFP97] employ an iterative pruning scheme, where a similar criterion is employed in order to throw away certain network weights after each step. In our work we want to employ a L^1 type penalty on weight matrices $W \in \mathbb{R}^{n\times m}$,

$$\|W\|_1 = \sum_{i=1}^{n}\sum_{j=1}^{m} |W_{ij}|$$

which has been widely employed to obtain sparse solution, e.g. [CM73]. In [Tib96] this yields to the so-called Lasso problem

$$\min_{\theta} \mathcal{L}(\theta) + \lambda \|\theta\|_1, \qquad \lambda > 0,$$

where we extend the L^1 norm to Θ as discussed in Section 3.3.5. This problem can for example by solved by a proximal gradient descent iteration

$$\theta^{(k+1)} = \text{soft shrinkage}(\theta - \tau \nabla \mathcal{L}(\theta^{(k)})) \tag{3.6}$$

where the shrinkage operator from Example 3.15. This iteration was employed in [Nit14; RVV20; Red+16] to train sparse neural networks.

Sparse-to-Sparse Training All of the sparsity-based methods mentioned before start with dense weight matrices and only decrease the number of parameters during or at the end of the training process. Our approach yields an iteration, where the network is sparse throughout the iteration. In fact we start with only very few non-zero parameters and only activate necessary weights during training. This paradigm is known as sparse-to-sparse or evolutionary training [Moc+18; DZ19; Evc+20; DYJ19; Fu+22; Hua+16; Liu+21].

Contribution in [BREG-I] Our work falls into the regime of sparse-to-sparse training. However, instead of relying on some heuristic growth strategy, we employ the concept of inverse scale flows (see Section 3.3.1) which allows us to obtain a optimization-driven framework, with a time-continuous interpretation. We propose a stochastic variant of linearized Bregman iterations (Section 3.3.3) and employ it to train a sparse neural network. We show monotonic decrease of the loss in the stochastic setting—which is not possible in the case of proximal gradient descent Eq. (3.6)—and convergence of the iterates under additional convexity assumptions, see Section 3.3.4. Finally, we demonstrate the numerical efficiency of the method (see Section 3.3.5) and provide an interesting applications for neural architecture search, which was further developed in [BREG-II].

3.3.1. Preliminaries on Convex Analysis and Bregman Iterations

We first review some necessary concepts from convex analysis that allow us to introduce the framework in [BREG-I]. We refer to [BB18; Roc97; BC11] for a more exhaustive introduction to the topics. The functional J is called lower semicontinuous if $J(u) \leq \liminf_{n\to\infty} J(u_n)$ holds for all sequences $(u_n)_{n\in\mathbb{N}} \subset \Theta$ converging to u.

Definition 3.9. Given a Hilbert space Θ and a functional $J : \Theta \to (-\infty, \infty]$.

1. The functional J is called convex, if

$$J(\lambda\overline{\theta} + (1-\lambda)\theta) \leq \lambda J(\overline{\theta}) + (1-\lambda)J(\theta), \quad \forall \lambda \in [0,1],\, \overline{\theta}, \theta \in \Theta. \tag{3.7}$$

2. The effective domain of J is defined as $\text{dom}(J) := \{\theta \in \Theta : J(\theta) \neq \infty\}$ and J is called proper if $\text{dom}(J) \neq \emptyset$.

In the following we want to consider functionals J that are convex, but not necessarily differentiable. Therefor, we define the subdifferential.

Definition 3.10. of a convex and proper functional $J : \Theta \to (-\infty, \infty]$ at a point $\theta \in \Theta$ as

$$\partial J(\theta) := \left\{ p \in \Theta \,:\, J(\theta) + \langle p, \overline{\theta} - \theta \rangle \leq J(\overline{\theta}),\ \forall \overline{\theta} \in \Theta \right\}. \tag{3.8}$$

If J is differentiable, then the subdifferntial coincides with the classical gradient (or Fréchet derivative). We denote $\mathrm{dom}(\partial J) := \{\theta \in \Theta \,:\, \partial J(\theta) \neq \emptyset\}$ and observe that $\mathrm{dom}(\partial J) \subset \mathrm{dom}(J)$.

The main algorithm in this section are so-called Bregman iterations, for which we first define the Bregman distance.

Definition 3.11 (Bregman Distance). Let $J : \Theta \to (-\infty, \infty]$ be a proper, convex functional. Then we define for $\theta \in \mathrm{dom}(\partial J), \overline{\theta} \in \Theta$

$$D_J^p(\overline{\theta}, \theta) := J(\overline{\theta}) - J(\theta) - \langle p, \overline{\theta} - \theta \rangle, \quad p \in \partial J(\theta). \tag{3.9}$$

For $p \in \partial J(\theta)$ and $\overline{p} \in \partial J(\overline{\theta})$ we define the *symmetric* Bregman distance as

$$D_J^{\mathrm{sym}}(\overline{\theta}, \theta) := D_J^p(\overline{\theta}, \theta) + D_J^{\overline{p}}(\theta, \overline{\theta}). \tag{3.10}$$

Intuitively, the Bregman distance $D_J^p(\overline{\theta}, \theta)$, measures the distance of J to its linearization around θ, see Fig. 3.1. If J is differentiable, then the subdifferential is single valued—we can suppress the sup script p—and we have

$$D_J(\overline{\theta}, \theta) = J(\overline{\theta}) - J(\theta) - \langle \nabla J(\theta), \overline{\theta} - \theta \rangle.$$

Example 3.12. For $\Theta = \mathbb{R}^n$ and $J = \frac{1}{2} \|\cdot\|_2^2$ we see that $\partial J(\theta) = \{\theta\}$ and therefore

$$\begin{aligned} D_J^p(\overline{\theta}, \theta) &= \frac{1}{2}\langle \overline{\theta}, \overline{\theta} \rangle - \frac{1}{2}\langle \theta, \theta \rangle - \langle \theta, \overline{\theta} - \theta \rangle \\ &= \frac{1}{2}\langle \overline{\theta}, \overline{\theta} \rangle + \frac{1}{2}\langle \theta, \theta \rangle - \langle \theta, \overline{\theta} \rangle \\ &= \frac{1}{2} \left\| \overline{\theta} - \theta \right\|_2^2 = J(\overline{\theta} - \theta). \end{aligned}$$

We can easily see that in general it is neither definite, symmetric nor fulfills the triangle inequality, hence it is not a metric. However, it fulfills the two distance axioms

$$D_J^p(\overline{\theta}, \theta) \geq 0, \quad D_J^p(\theta, \theta) = 0, \quad \forall \overline{\theta} \in \Theta, \theta \in \mathrm{dom}(\partial J). \tag{3.11}$$

The same holds for the symmetric Bregman distance, where additionally—as the name suggests—the symmetry property is fulfilled. The last concept that is crucial in [BREG-I] is the so-called proximal operator.

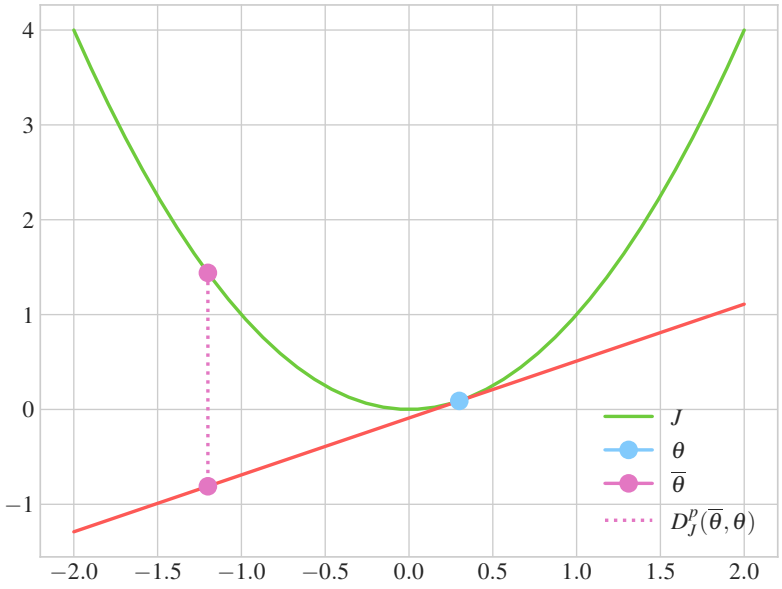


Figure 3.1.: Visualization of the Bregman distance.

Definition 3.13. Let $J : \Theta \to (-\infty, \infty]$ be convex, proper and lower semicontinuous functional, then we define the *proximal operator* as

$$\text{prox}_J(\bar{\theta}) := \text{argmin}_{\theta \in \Theta} \frac{1}{2} \left\| \theta - \bar{\theta} \right\|^2 + J(\theta).$$

If J is additionally a closed function, i.e., its sublevel sets

$$N_\alpha = \{\theta \in \text{dom}\, J : J(\theta) \leq \theta\}$$

are closed for every $\alpha \in \mathbb{R}$ then we have that the function $\tilde{J} = \frac{1}{2} \|\theta - \cdot\|^2 + J(\theta)$ is closed, proper and *strongly* convex and therefore has a unique minimizer, see [Roc97, Thm. 27.1]. Additionally, one often considers a regularization parameter $\lambda > 0$ and is then interested in $\text{prox}_{\lambda J}$.

Remark 3.14. The the optimality conditions for $\theta = \text{prox}_{\lambda J}(\bar{\theta})$ yield

$$\bar{\theta} - \theta \in \lambda \partial J(\theta).$$

For a proper, closed and convex function we obtain

$$\theta = (I + \lambda \partial J)^{-1}(\bar{\theta})$$

where $(I + \lambda \partial J)^{-1}$ is called the *resolvent* and is a one-to-one mapping (see [PB+14, Ch. 3.2]) which justifies the equality in the above equation. If J is differentiable, then we have $\partial J = \{\nabla J\}$ and therefore,

$$\text{prox}_{\lambda J} = (I + \lambda \nabla J)^{-1}.$$

△

In the following we list two relevant examples for the application in [BREG-I; BREG-II].

Example 3.15. If $J = \|\cdot\|$ is a norm and $\lambda > 0$ then we have that (see e.g. [PB+14])

$$\operatorname{prox}_{\lambda J}(\overline{\theta}) = \overline{\theta} - \operatorname{Proj}_{\|\cdot\|^*}(\overline{\theta}/\lambda)$$

where $\operatorname{Proj}_{\|\cdot\|^*}$ denotes the projection operator w.r.t. the dual norm $\|\theta\|^* = \sup\{|\langle f, \theta\rangle| : f \in \Theta^*\}$. In the case of ℓ^p norms on $\mathbb{R}^n$ we know that that

$$\|\theta\|_p^* = \|\theta\|_q$$

with $1/p + 1/q = 1$ with the notational convention of $1/\infty = 0$. Especially relevant are the cases $p \in \{1, 2\}$. Here, we then have that

$$\operatorname{prox}_{\lambda\|\cdot\|_2}(\overline{\theta}) = \overline{\theta}\left(1 - \min\left\{\frac{\lambda}{\left\|\overline{\theta}\right\|_2}, 1\right\}\right) = \begin{cases} \overline{\theta}\,(1 - \lambda/\left\|\overline{\theta}\right\|_2) & \text{if } \left\|\overline{\theta}\right\|_2 \geq \lambda \\ 0 & \text{else} \end{cases}$$

and for $i = 1, \ldots, n$

$$\operatorname{prox}_{\lambda\|\cdot\|_1}(\overline{\theta})_i = \operatorname{sign}(\overline{\theta}_i)\,\max\left\{\left|\overline{\theta}_i\right| - \lambda, 0\right\} = \begin{cases} \overline{\theta}_i - \lambda & \text{if } \overline{\theta} > \lambda \\ 0 & \text{if } \left|\overline{\theta}_i\right| \leq \lambda \\ \overline{\theta}_i + \lambda & \text{if } \overline{\theta} < -\lambda \end{cases}$$

the so called *soft shrinkage operator*.

Example 3.16 (Group Norms). Another relevant functional J is the group norm $\ell_{1,2}$ that—in the context of sparse neural networks—was first employed by [Sca+17]. Here, we assume that the parameters in Θ can be grouped in a collection of parameters $\mathcal{G}$, for which we the choose

$$J(\theta) = \sum_{g \in \mathcal{G}} \sqrt{\#\mathcal{G}}\,\|g\|_2$$

In this case the proximal operator is given as

$$\operatorname{prox}_{\lambda J}(\overline{\theta})_g = g\,\max\left\{1 - \min\left\{\frac{\lambda\sqrt{\#\mathcal{G}}}{\|g\|_2}, 1\right\}, 0\right\}$$

Bregman Iterations Our goal is to minimize a function $\mathcal{L}$ while simultaneously obtaining a low value w.r.t. the functional J. One popular approach considers the regularized problem

$$\min_\theta \mathcal{L}(\theta) + \lambda J(\theta) \qquad \lambda > 0,$$

see e.g. [Tik43; CP08; DDD04; FS06; FNW07; Cha04; CP11], which however influences the minimizers of the original problem $\min_\theta \mathcal{L}(\theta)$. In the derivation of the Bregman iterations one can take a different viewpoint. Assume that we want to employ an iterative scheme, where in each step we want minimize $\mathcal{L}$ while penalizing the distance to the previous iterate. For a stepping parameter τ and starting from some $\theta^{(0)} \in \Theta$ this yields the update

$$\theta^{(k+1)} = \operatorname{argmin}_\theta \tau\mathcal{L}(\theta) + \frac{1}{2}\left\|\theta - \theta^{(k)}\right\|^2 = \operatorname{prox}_{\tau\mathcal{L}}(\theta^{(k)}). \tag{3.12}$$

At first sight this is either known as the proximal point algorithm [Bre67] or a minimizing movement scheme [De 93]. If $\mathcal{L}$ is differentiable, this update can be rewritten as

$$\theta^{(k+1)} = (I + \tau\nabla\mathcal{L})^{-1}\theta^{(k)} \Leftrightarrow \frac{1}{\tau}\left(\theta^{(k+1)} - \theta^{(k)}\right) = -\nabla\mathcal{L}(\theta^{(k+1)})$$

which is a implicit Euler discretization ([Eul24]) of the time-continuos gradient flow

$$\partial_t\theta_t = -\nabla\mathcal{L}(\theta_t).$$

We see that the penalization term in Eq. (3.12) is in fact the Bregman distance of the functional $\frac{1}{2}\|\cdot\|^2$. In order to incorporate an arbitrary convex functional J—and therefore allow each iterate to only slightly deviate w.r.t. the Bregman distance of J to the previous iterate—we employ $D_J^{p^{(k)}}(\cdot, \theta^{(k)})$ as a penalization term. In order to obtain a update scheme for the subgradients, we observe

$$\theta = \operatorname{argmin}_{\theta\in\Theta} D_J^{p^{(k)}}\left(\theta, \theta^{(k)}\right) + \tau\mathcal{L}(\theta) \tag{3.13}$$

$$\Leftrightarrow p^{(k)} + \tau\nabla\mathcal{L}(\theta) \in \partial J(\theta). \tag{3.14}$$

This finally yields *Bregman iteration* of [Osh+05]

$$\theta^{(k+1)} = \operatorname{argmin}_{\theta\in\Theta} D_J^{p^{(k)}}\left(\theta, \theta^{(k)}\right) + \tau\mathcal{L}(\theta), \tag{3.15a}$$

$$p^{(k+1)} = p^{(k)} - \tau\nabla\mathcal{L}(\theta^{(k+1)}) \in \partial J(\theta^{(k+1)}). \tag{3.15b}$$

The very nature of Bregman iterations means starting with a iterate $\theta^{(0)}$ that has a low value in J—preferably $J(\theta^{(0)}) = 0$—and only increase $J(\theta^{(k)})$ gradually as k increases.

Remark 3.17. Originally, the iterations were employed for solving inverse problems. Here, we are given a forward operator $A : \Theta \to \tilde{\Theta}$ and a noisy measurement $f = A\theta + \delta$ where $\delta \in \tilde{\Theta}$ is additive noise. The loss function is then of the form

$$\mathcal{L} = \frac{1}{2}\|A\cdot - f\|_2^2$$

for which one can show that the Bregman iterations converge to a solution of

$$\min\{J(\theta) : A\theta = f\}, \tag{3.16}$$

see e.g. [Osh+05]. In comparison the concept of adding a regularizing term with parameter $\lambda > 0$, i.e. considering the problem

$$\min_{\theta} \mathcal{L}(\theta) + \lambda J(\theta)$$

actually modifies the minimizers. In this sense Bregman iterations do not introduce a bias. △

Example 3.18. In order to get an intuition about the behavior of Bregman iterations, we consider an image denoising task. I.e. we are given a noisy image $\mathbb{R}^{n\times m} \ni f = u+\delta$ where $\delta \in \mathbb{R}^{n\times m}$ is additive noise. In order to obtain $u \in \mathbb{R}^{n\times n}$ from f we employ the TV functional [ROF92]

$$J(u) = TV(u) := \sum_{i,j} \sqrt{|u_{i+1,j} - u_{i,j}|^2 + |u_{i,j+1} - u_{i,j}|^2}$$

together with the loss function $\mathcal{L}(u) := \frac{1}{2}\|u - f\|_2^2$. We start with an image $u^{(0)}$ such that $TV(u^{(0)}) = 0$, i.e. a constant image. In Fig. 3.2 we visualize the iteration. At lower iterations $u^{(k)}$ only displays features on a larger scale, while at the end, the iteration converges back to smallest possible scale, the noisy data. In order to obtain an appropriate denoising, one needs to employ a early stopping here. This fits well to the insight from Eq. (3.16) since here the forward operator is the identity, i.e.

$$\left\{u : \frac{1}{2}\|u - f\|^2 = 0\right\} = \{f\}.$$

It should also be noted that this example only serves a explanatory purpose. In practice directly applying Eq. (3.15) for $J = TV$ can become infeasible since the first minimization problem is expensive.

If $J = \frac{1}{2}\|\cdot\|_2^2$ as in Example 3.12 then this amounts to the step

$$\theta^{(k+1)} = \operatorname{argmin}_{\theta\in\Theta} \frac{1}{2}\left\|\theta - \theta^{(k)}\right\|_2^2 + \tau\mathcal{L}(\theta),$$

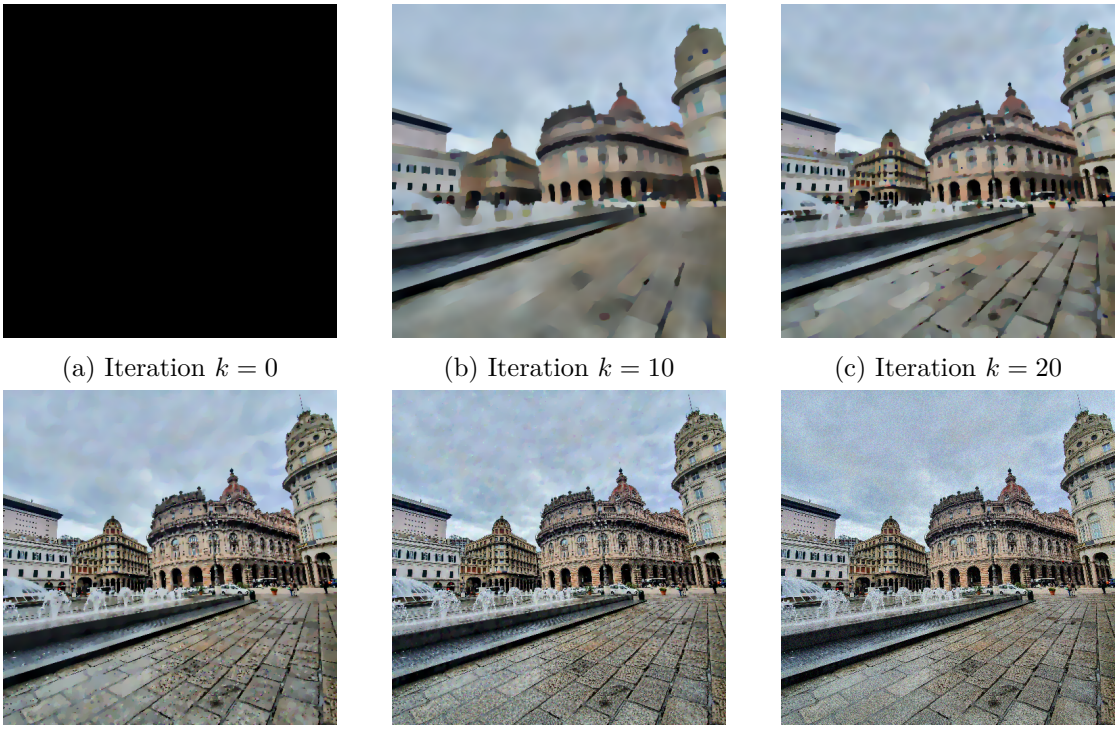
where the optimality conditions then yield

$$\theta^{(k+1)} - \theta^{(k)} + \tau\nabla\mathcal{L}(\theta^{(k+1)}) = 0 \Leftrightarrow \theta^{(k+1)} = \theta^{(k)} - \tau\nabla\mathcal{L}(\theta^{(k+1)})$$

which is a standard implicit Euler iteration. The time continuous flow for $\tau \to 0$ is known as the *inverse scale space* flow [Bur+06; Bur+07],

$$\begin{cases} \dot{p}_t = -\nabla\mathcal{L}(\theta_t), \\ p_t \in \partial J(\theta_t), \end{cases}$$

where again for $J = \frac{1}{2}\|\cdot\|_2^2$ we obtain that $\partial J(\theta_t) = \theta_t$ and therefore obtain the standard gradient flow. Hence we see, that the inverse scale space flow is a generalization of the standard gradient flow.



(a) Iteration $k = 0$ (b) Iteration $k = 10$ (c) Iteration $k = 20$

(d) Iteration $k = 50$ (e) Iteration $k = 100$ (f) Iteration $k = 500$

Figure 3.2.: Bregman iterations for image denoising in Example 3.18

3.3.2. Linearized Bregman Iterations and Mirror Descent

The minimization step in Eq. (3.15) is infeasible for large scale applications, especially in our setting of neural networks. Therefore, we employ the idea introduced in [Yin+08; COS09]. Here, we first linearize the loss function around the previous iterate,

$$\mathcal{L}(\theta) \approx \mathcal{L}(\theta^{(k)}) + \left\langle \nabla \mathcal{L}(\theta^{(k)}), \theta - \theta^{(k)} \right\rangle .$$

The next step is to replace J with the strongly convex elastic net regularization

$$J_\delta := J + \frac{1}{2\delta} \|\cdot\|_2^2 . \tag{3.17}$$

The minimization step then transforms to

$$\begin{aligned} &\operatorname{argmin}_{\theta\in\Theta} D_{J_\delta}^{p^{(k)}}\left(\theta, \theta^{(k)}\right) + \tau \left\langle \nabla \mathcal{L}(\theta^{(k)}), \theta \right\rangle \\ &\quad = \operatorname{argmin}_{\theta\in\Theta} J(\theta) + \frac{1}{2\delta}\|\theta\|_2^2 - \langle p^k, \theta\rangle + \tau \left\langle \nabla \mathcal{L}(\theta^{(k)}), \theta \right\rangle \\ &\quad = \operatorname{argmin}_{\theta\in\Theta} J(\theta) + \frac{1}{2\delta}\left\|\theta - \delta(p^{(k)} - \tau\nabla\mathcal{L}(\theta^{(k)})\right\|_2^2 - \underbrace{\left\|p^{(k)} - \tau\nabla\mathcal{L}(\theta^{(k)})\right\|_2^2}_{\text{constant in } \theta} \\ &\quad = \operatorname{prox}_{\delta J}\left(\delta\left(p^{(k)} - \tau\nabla\mathcal{L}(\theta^{(k)})\right)\right). \end{aligned} \tag{3.18}$$

Note that here $p^{(k)}$ is a subgradient of J_δ at θ therefore we derive the subgradient update rule

$$p^{(k+1)} := p^{(k)} - \tau\mathcal{L}(\theta^{(k)}).$$

This finally yields the linearized Bregman iterations

$$p^{(k+1)} = p^{(k)} - \tau\nabla\mathcal{L}(\theta^{(k)}), \tag{3.19a}$$

$$\theta^{(k+1)} = \operatorname{prox}_{\delta J}(\delta p^{(k+1)}). \tag{3.19b}$$

The last line is equivalent to $p^{(k+1)} \in \partial J_\delta(\theta^{(k+1)})$ for which we obtain the continuous linearized flow

$$\begin{aligned} \dot{p}_t &= -\nabla\mathcal{L}(\theta_t), \\ p_t &\in \partial J_\delta(\theta_t), \end{aligned}$$

see [Bur+06; Bur+07].

Connections To Mirror Descent As already noticed by [Vil+23] linearized Bregman iteration are equivalent to mirror descent in some situations. We show the equivalence in the following, where we employ similar arguments as in [BT03]. One assumes to be given a differentiable and strongly convex function $h : \Theta \to \mathbb{R}$, i.e.,

$$h(\overline{\theta}) - h(\theta) - \langle \nabla h(\theta), \overline{\theta} - \theta\rangle \geq \frac{1}{2}\left\|\overline{\theta} - \theta\right\|_2^2$$

for all $\theta, \overline{\theta} \in \Theta$. The mirror descent update then reads ([NY83; BT03])

$$\theta^{(k+1)} = \nabla h^* \left(\nabla h \left(\theta^{(k)} \right) - \tau \mathcal{L}(\theta^{(k)}) \right) \tag{3.20}$$

where h^* denotes the Fenchel conjugate

$$h^*(p) = \sup_{\theta} \langle p, \theta \rangle - h(\theta)$$

with the gradient (see [BV04])

$$\nabla h^*(p) = \operatorname{argmax}_\theta \left\{ \langle p, \theta \rangle - h(\theta) \rangle \right\}.$$

Therefore, we see that Eq. (3.20) can be written as

$$\begin{aligned} \theta^{(k+1)} &= \operatorname{argmax}_\theta \left\{ \left\langle \nabla h \left(\theta^{(k)} \right) - \tau \mathcal{L}(\theta^{(k)}), \theta \right\rangle - h(\theta) \right\} \\ &= \operatorname{argmax}_\theta \left\{ -D_h(\theta, \theta^{(k)}) - \tau \left\langle \mathcal{L}(\theta^{(k)}), \theta \right\rangle \right\} \\ &= \operatorname{argmin}_\theta \left\{ D_h(\theta, \theta^{(k)}) + \tau \left\langle \mathcal{L}(\theta^{(k)}), \theta \right\rangle \right\} \end{aligned}$$

which was our starting point to derive linearized Bregman iterations for $h = J_\delta$ in Eq. (3.18). In fact, we can always find a convex functional $J : \Theta \to \mathbb{R}$ such that $h = J + \frac{1}{2} \|\cdot\|_2^2$ for which we see, that Eq. (3.19) is a more general formulation of Eq. (3.20).

3.3.3. Stochastic and Momentum Variants

We want to employ linearized Bregman iterations to train a neural network. As mentioned in Section 3.1.2 we therefore do not compute the full gradient of $\mathcal{L}$ but rather a minibatched variant. This yields stochastic Bregman Iterations

$$\begin{aligned} &\text{draw } \omega^{(k)} \text{ from } \Omega \text{ using the law of } \mathbb{P}, \\ &\quad g^{(k)} := g(\theta^{(k)}; \omega^{(k)}), \\ &v^{(k+1)} := v^{(k)} - \tau^{(k)} g^{(k)}, \\ &\theta^{(k+1)} := \operatorname{prox}_{\delta J}(\delta v^{(k+1)}), \end{aligned} \tag{3.21}$$

which we also abbreviate as the *LinBreg* algorithm in the following. The basic update scheme is given as the linearized Bregman iterations from[Osh+05], however the presence of a stochastic gradient estimator significantly complicates the convergence analysis, as observed in Section 3.3.4. However, this algorithm can now be efficiently employed to train a neural network. For the analogous stochastic mirror descent algorithm we refer to [Nem+09].

Momentum Variant Typically, the learning process of a neural network can be improved by introducing a momentum term (see e.g. [Nes83; Qia99]) in the optimizer. In our case this can be achieved, by replacing the gradient update on the subgradient variable. In [BREG-I] we first consider the inertia version of the gradient flow as

$$\begin{cases} \gamma \ddot{v}_t + \dot{v}_t = -\nabla \mathcal{L}(\theta_t), \\ v_t \in \partial J_\delta(\theta_t). \end{cases}$$

for which the discretization then reads

$$\begin{aligned} m^{(k+1)} &= \beta^{(k)} m^{(k)} + (1 - \beta^{(k)}) \tau^{(k)} g^{(k)}, \\ v^{(k+1)} &= v^{(k)} - m^{(k+1)}, \\ \theta^{(k+1)} &= \operatorname{prox}_{\delta J}(\delta v^{(k+1)}). \end{aligned} \tag{3.22}$$

Adamized Bregman Iteration We shortly remark that one can replace the momentum update in Eq. (3.22) with a Adam update [KB14]. This then yields an Adamized version of linearized Bregman iterations as employed in [BREG-I].

3.3.4. Convergence of Stochastic Bregman Iterations

While various previous works prove convergence of linearized Bregman iterations (see e.g. [Osh+05; COS09]), the stochastic setting requires special treatment. In [BREG-I] the first guarantees for the algorithm in Eq. (3.21) were proven. Other work on convergence of stochastic Bregman iterations, or mirror descent [DEH21; HR21; ZH18; DOr+21; AKL22] requires a differentiable functional J. However, since our main motivation is to a functional in the flavor of the ℓ^1 norm this is not applicable. Therefore, we present the novel convergence analysis of [BREG-I].

Assumptions on the Gradient Estimator In order to obtain convergence guarantees, we need to assume mainly two properties on the gradient estimator $g(\cdot, \cdot)$. First we assume unbiasedness, which means

$$\mathbb{E}\left[g(\theta; \omega)\right] = \nabla \mathcal{L}(\theta) \text{ for all } \theta \in \Theta.$$

The second assumption we need in the following is referred to as *bounded variance* of the estimator.

Assumption 3.19 (Bounded variance). There exists a constant $\sigma > 0$ such that for any $\theta \in \Theta$ it holds

$$\mathbb{E}\left[\|g(\theta; \omega) - \nabla \mathcal{L}(\theta)\|^2\right] \leq \sigma^2. \tag{3.23}$$

Remark 3.20. We want to remark, that this property is weaker than the bounded gradient assumption

$$\mathbb{E}\left[\|g(\theta;\omega)\|^2\right] \leq C$$

for some constant $C > 0$. In fact this condition can not be enforced together with a strong convexity assumption—which we employ in **??**—as shown in [Ngu+18]. △

Assumptions on the Regularizer and on the Loss Function The assumptions on the regularization functional J are mild and merely ensure the well-definedness of the proximal mapping,

Assumption 3.21 (Regularizer). We assume that $J : \Theta \to (-\infty, \infty]$ is a convex, proper, and lower semicontinuous functional on the Hilbert space Θ.

Our assumptions on the loss function $\mathcal{L}$ are more restrictive. We require it to be bounded from below and differentiable, which are both standard assumptions. Additionally, we require Lipschitz continuity of the gradient, which also commonly employed in optimization literature.

Assumption 3.22 (Loss function). We assume the following conditions on the loss function:

- The loss function $\mathcal{L}$ is bounded from below and without loss of generality we assume $\mathcal{L} \geq 0$.
- The function $\mathcal{L}$ is continuously differentiable.
- The gradient of the loss function $\theta \mapsto \nabla\mathcal{L}(\theta)$ is L-Lipschitz for $L \in (0, \infty)$:

$$\left\|\nabla\mathcal{L}(\tilde{\theta}) - \nabla\mathcal{L}(\theta)\right\| \leq L\left\|\tilde{\theta} - \theta\right\|, \quad \forall\theta, \tilde{\theta} \in \Theta. \tag{3.24}$$

If the loss function $\mathcal{L}$ fulfills the previous assumptions we are able to prove loss decay of the iterates in Theorem 3.26. However, in order to show convergence of the iterates we additionally need a convexity assumption. For a differentiable functional J, the authors in [DEH21] assumed

$$\nu\ D_J(\overline{\theta}, \theta) \leq D_{\mathcal{L}}(\overline{\theta}, \theta)$$

which for twice differentiable $J, \mathcal{L}$ transfers to

$$\nu\ \nabla^2 J \lesssim \nabla^2\mathcal{L}, \qquad \forall\overline{\theta}, \theta \in \Theta.$$

Plugging in the definition of the Bregman dist $D_{\mathcal{L}}$ we obtain

$$\nu\ D_J(\overline{\theta}, \theta) \leq \mathcal{L}(\overline{\theta}) - \mathcal{L}(\theta) - \langle\nabla\mathcal{L}(\theta), \overline{\theta} - \theta\rangle.$$

In this form one observes that this is in fact a convexity assumption on $\mathcal{L}$ in a J dependent distance, as employed in [BREG-I].

Assumption 3.23 (Strong convexity). For a proper convex function $H : \Theta \to \mathbb{R}$ and $\nu \in (0, \infty)$, we say that the loss function $\theta \mapsto \mathcal{L}(\theta)$ is ν-strongly convex w.r.t. H, if

$$\mathcal{L}(\overline{\theta}) \geq \mathcal{L}(\theta) + \langle \nabla \mathcal{L}(\theta), \overline{\theta} - \theta \rangle + \nu D_J^p(\overline{\theta}, \theta), \quad \forall \theta, \overline{\theta} \in \Theta, p \in \partial H(\theta). \tag{3.25}$$

Remark 3.24. We have two relevant cases for the choice of H. For $H = \frac{1}{2} \|\cdot\|^2$ Assumption 3.23 reduces to standard strong ν-convexity. The other relevant case, is $H = J_\delta$, i.e. we consider convexity w.r.t. to the functional J_δ. △

Remark 3.25. In the setting of training a neural network, where we employ the empirical loss Eq. (3.1), this convexity assumption usually fails. While it is possible to enforce this conditions only locally around the minimum, this does not significantly improve the applicability. For future work, it would be desirable to enforce a Kurdyka–Łojasiewicz inequality, as in [Ben+21] for the deterministic case. △

Loss Decay The first convergence result considers the loss decay of the iterates. Here, we do not assume convexity of the loss function. Under this assumptions [Ben+21; BB18] were able to show the inequality

$$\mathbb{E}\left[\mathcal{L}(\theta^{(k+1)})\right] + \frac{1}{\tau^{(k)}} \mathbb{E}\left[D_J^{\mathrm{sym}}(\theta^{(k+1)}, \theta^{(k)})\right] + \frac{C}{2\delta\tau^{(k)}} \mathbb{E}\left[\left\|\theta^{(k+1)} - \theta^{(k)}\right\|^2\right] \leq \mathbb{E}\left[\mathcal{L}(\theta^{(k)})\right].$$

In our setting, employing a stochastic gradient estimator, we are able to prove a similar estimate. Here, we obtain an additional term scaled σ which controls the expected squared difference between the gradient estimator and the actual gradient. It should however be noted, that the proof is not only a trivial extension.

Theorem 3.26 ([BREG-I, Th. 2]: Loss decay). Assume that Assumptions 3.19, 3.21 and 3.22 hold true, let $\delta > 0$, and let the step sizes satisfy $\tau^{(k)} \leq \frac{2}{\delta L}$. Then there exist constants $c, C > 0$ such that for every $k \in \mathbb{N}$ the iterates of (3.21) satisfy

$$\mathbb{E}\left[\mathcal{L}(\theta^{(k+1)})\right] + \frac{1}{\tau^{(k)}} \mathbb{E}\left[D_J^{\mathrm{sym}}(\theta^{(k+1)}, \theta^{(k)})\right] + \frac{C}{2\delta\tau^{(k)}} \mathbb{E}\left[\left\|\theta^{(k+1)} - \theta^{(k)}\right\|^2\right] \leq \mathbb{E}\left[\mathcal{L}(\theta^{(k)})\right] + \tau^{(k)} \delta \frac{\sigma^2}{2c}, \tag{3.26}$$

Convergence of the Iterates Here, we have two cases respectively proving convergence w.r.t. the L^2 distance and the Bregman distance of J_δ. The first assumes strong convexity with $H = \frac{1}{2} \|\cdot\|^2$ in Assumption 3.23.

Theorem 3.27 ([BREG-I, Th. 6]: Convergence in norm). Assume that Assumptions 3.19, 3.21 and 3.22 and Assumption 3.23 for $H = \frac{1}{2}\|\cdot\|^2$ hold true and let $\delta > 0$. Furthermore, assume that the step sizes $\tau^{(k)}$ are such that for all $k \in \mathbb{N}$:

$$\tau^{(k)} \leq \frac{\mu}{2\delta L^2}, \qquad \tau^{(k+1)} \leq \tau^{(k)}, \qquad \sum_{k=0}^{\infty} (\tau^{(k)})^2 < \infty, \qquad \sum_{k=0}^{\infty} \tau^{(k)} = \infty.$$

The function $\mathcal{L}$ has a unique minimizer θ^* and if $J(\theta^*) < \infty$ the stochastic linearized Bregman iterations (3.21) satisfy the following:

- Letting $d_k := \mathbb{E}\left[D_{J_\delta}^{v^{(k)}}(\theta^*, \theta^{(k)})\right]$ it holds

$$d_{k+1} - d_k + \frac{\mu}{4}\tau^{(k)}\mathbb{E}\left[\left\|\theta^* - \theta^{(k+1)}\right\|^2\right] \leq \frac{\sigma}{2}\left((\tau^{(k)})^2 + \mathbb{E}\left[\left\|\theta^{(k)} - \theta^{(k+1)}\right\|^2\right]\right). \tag{3.27}$$

- The iterates possess a subsequence converging in the L^2-sense of random variables:

$$\lim_{j\to\infty} \mathbb{E}\left[\left\|\theta^* - \theta^{(k_j)}\right\|^2\right] = 0. \tag{3.28}$$

Here, J_δ is defined as in (3.17).

For the second result we assume convexity w.r.t. the Bregman distance, i.e. we choose $H = J_\delta$ in Assumption 3.23. This induces a relation between the Bregman distance of J and the loss function $\mathcal{L}$, which has been similarly employed in [DEH21].

Theorem 3.28 ([BREG-I, Th. 11]: Convergence in the Bregman distance). Assume that Assumptions 3.19, 3.21 and 3.22 and Assumption 3.23 for $H = J_\delta$ hold true and let $\delta > 0$. The function $\mathcal{L}$ has a unique minimizer θ^* and if $J(\theta^*) < \infty$ the stochastic linearized Bregman iterations (3.21) satisfy the following:

- Letting $d_k := \mathbb{E}\left[D_{J_\delta}^{v^{(k)}}(\theta^*, \theta^{(k)})\right]$ it holds

$$d_{k+1} \leq \left[1 - \tau^{(k)}\nu\left(1 - \tau^{(k)}\frac{2\delta^2 L^2}{\nu}\right)\right] d_k + \delta(\tau^{(k)})^2\sigma^2. \tag{3.29}$$

- For any $\varepsilon > 0$ there exists $\tau > 0$ such that if $\tau^{(k)} = \tau$ for all $k \in \mathbb{N}$ then

$$\limsup_{k\to\infty} d_k \leq \varepsilon. \tag{3.30}$$

- If $\tau^{(k)}$ is such that

$$\lim_{k\to\infty} \tau^{(k)} = 0 \quad \text{and} \quad \sum_{k=0}^{\infty} \tau^{(k)} = \infty \tag{3.31}$$

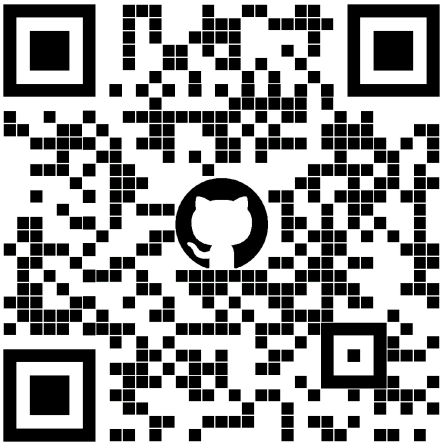
then it holds

$$\lim_{k\to\infty} d_k = 0. \tag{3.32}$$

Here, J_δ is defined as in (3.17).

3.3.5. Numerical Results and Practical Considerations

Before briefly reviewing the numerical results in [BREG-I, Sec. 4], we remark on some practical considerations. In particular we comment on the parameter initialization strategy. All the experiments were implemented in `Python` [VD95] employing—among others—the `PyTorch` [Pas+19].



The code for all the experiments is available at github.com/TimRoith/BregmanLearning.

Parameter Initialization As already noticed in [GB10] parameter initialization has a significant impact on the training of the neural network. Here, in contrast to standard Bregman methods, we are not able to initialize the parameters of the neural network as $\theta = 0$. This is due to that fact, that a zero initialization induces symmetries in the network weights, for which one cannot utilize the full expressivity of the architecture [GBC16, Ch. 6]. Therefore, we rather employ the approach from [Liu+21; DZ19; Mar10] of sparsifying weight matrices $\tilde{W}^l \in \mathbb{R}^{n_{l+1}\times n_l}$ up to a certain level, by a pointwise multiplication with a binary mask $M^l \in 0, 1^{n_{l+1}\times n_l}$

$$W^l := \tilde{W}^l \odot M^l.$$

Each entry in M^l is i.i.d. sampled from a Bernoulli distribution

$$M^l_{i,j} \sim \mathcal{B}(r).$$

where the parameter r determines the sparsity,

$$\mathrm{N}(W^l) := \frac{\left\|W^l\right\|_0}{n_l \cdot n_{l-1}} = 1 - \mathrm{S}(W^l)$$

with N denoting the percentage of used parameters and S the sparsity. In [GB10] the authors advise to especially control to the variance of the parameter initialization

distribution, for which in [BREG-I] we derive

$$\operatorname{Var}\left[\tilde{W}^l\right] = \frac{1}{r} \operatorname{Var}\left[\tilde{W}^l \odot M^l\right] \tag{3.33}$$

and therefore scale the weights with the sparsity parameter r at initialization.

Choice of Regularizers In all our experiments we choose a L^1 type sparsity promoting regularization function functional J. We do not employ any coupling between weight matrices of different layers, and therefore for $\theta = ((W^1, b^1), \dots, (W^L, b^L))$ we have

$$J(\theta) = \sum_{l=1}^{L} J^l(W^l)$$

where J^l is choosen according to the layer type. In the easiest case of a fully connected layer, we can choose

$$J^l(W^l) := \left\|W^l\right\|_1 .$$

In the case of a convolutional layer we have that W^l is determined by convolutional kernels $K_{i,j} \in \mathbb{R}^{k \times k}$, see Section 3.1.1. Here, we typically employ a group sparsity term in the form

$$J^l(W^l) = \left\|W^l\right\|_{2,1} = \sum_{i,j} \|K_{i,j}\|_2 .$$

The outer sum acts as a L^1 regularizer on the instances $\|K_{i,j}\|$. Sparsity in this sense, then amounts to having indices (i,j) for which $\|K_{i,j}\|_2 = 0 \Leftrightarrow K_{i,j} = 0$, i.e., we prune away whole convolutional filters. This effect is displayed in [BREG-I, Fig. 1]. We can also employ group sparsity on fully connected layers, by considering row sparsity of $W^l \in \mathbb{R}^{n_{l+1}, n_l}$s

$$J^l(W^l) = \sum_{i=1}^{n_{l+1}} \|W_{i,:}\|_2 = \sum_{i=1}^{n_{l+1}} \sqrt{\sum_{js=1}^{n_l} W_{i,j}^2}.$$

In this setting we have a L^1 penalty on the row norms $\|W_{i,:}\|_2$ which therefore enforces whole rows to be zero. This is relevant, if we employ a layer architecture with $\Psi^l(0) = 0$, e.g., using no bias vectors and the ReLU activation function. In this setting, if the ith row of W^l is zero this effectively means, that the ith neuron in layer $l+1$ is inactive. This observation allows the neural architecture search in one of the following paragraphs.

Comments on the Numerical Results We briefly remark on the numerical results as displayed in [BREG-I, Sec. 4]. In the experiments we employed feed-forward networks with simple linear, convolutional and residual layers and tested on the three datasets [Kri09; XRV17; LC10].

The basic comparison between the algorithms SGD, ProxGD and LinBreg shows the qualitative behaviour of each iterations. Infantilizing sparse does not have any effect on SGD, since it does not preserve the sparsity in any way. ProxGD rather starts with many active parameters and reduces the this number during the iteration. Only the discretization of the inverse scale space flow—via Bregman iterations—shows the desired behaviour of gradually adding active parameters. Furthermore, in [BREG-I, Fig. 2] we can see, that the choice of λ in the regularizer $J = \lambda \|\cdot\|_1$ changes the results significantly. In the light of Eq. (3.16) this is not expected for the standard Bregman iterations with a convex loss. It is therefore interesting to see, that in our non-convex and stochastic situation this effect changes.

The momentum variants, as discussed in Section 3.3.3 yield the desired effect of enhancing the validation accuracy, and respectively converging faster. However in each of the experiments, one can also observe that adding a momentum term has the effect that more parameters are added faster. On the one hand this could mean that the network actually requires more parameters to have a higher accuracy, for which a momentum variant is more likely to increase the number of needed parameters. However, the quantitative evaluation on the CIFAR10 dataset [Kri09] shows, that especially the Adamized version tends to increase the number of used parameters rather aggressively, while only slightly increasing the performance of the net. The performance here is very similar to the one of proximal gradient descent. However, the training of a residual network seems to be slightly better with a standard Lasso implementation. Here, one neglects the non-differentiability of the L^1 norm and computes a derivate via automatic differentiation [Ral81; MDA15] (we employed the `autograd` library of the `PyTorch` package [Pas+19]). In order to obtain true zeros in the weight matrix one then has to employ a thresholding operation after the training. In some sense this method is not a proper sparse training approach, but rather a regularization method with an added pruning step at the end.

Comments on Efficiency One of the major advantages of the Bregman approach, is that the network is sparse already during the training time. As with all sparse-to-sparse training approaches this yields a very small number of active parameters over all training step. This sparsity can be easily exploited in each forward pass. However, it is not directly possible to achieve performance gains during the backward pass of the network, since in general

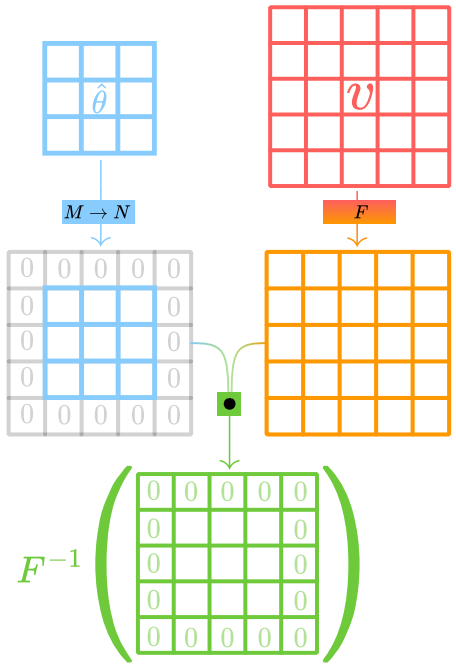
$$W_{ij}^l = 0 \not\Rightarrow \partial_{W_{ij}^l} \mathcal{L}(\theta) = 0.$$

In [BREG-I; BREG-II] there are no evaluation on the training time and memory consumption of the Bregman algorithm. Since the complexity does not increase in comparison to the standard SGD, one hopes to obtain a faster training time here. This is an interesting open question for future work. It should however be remarked that the computational complexity of the LinBreg algorithm does not increase significantly, compared to SGD, since the evaluation of the proximal operator is very efficient for L^1 type functionals.

Neural Architecture Search An interesting aspect hinted in [BREG-I] is optimizing the architecture of a neural network via sparsity. In the example provided in [BREG-I, Fig. 4] one defines a super-architecture as multi-layer perceptron with an equal number of neurons in each layer. Training this task with the LinBreg algorithm reveals the well-known autoencoder structure [HZ93]. This idea was developed further in [BREG-II], where also skip connection of a residual architecture were learned. Here, the super-architecture was given by a dense net [Hua+17], where a each skip connection was scaled by a parameter, which was penalized with a sparsity term. The driving question here, if it is possible also learn an architecture similar to the U-Net as proposed in [RFB15].

3.4. Resolution Stability

We frequently employ the set $\mathcal{X} = [0,1]^{K\times N\times M}$ to represent images. However, from a modeling point of view it is more natural to assume that images are functions $u : \Omega \to [0,1]^K$ where $\Omega \subset \mathbb{R}^2$ is some domain. In this sense, the space $\mathcal{X}$ only constitutes a discretization of the space of all functions from Ω to $[0,1]^K$. The number of pixels, i.e. $N\cdot M$, relates to the *resolution* of the image, where a higher number of pixels yields a higher resolution. In this sense the resolution is merely an artifact of the restriction to finite dimensions in computer vision and not relevant for the classification. Therefore, one wants to obtain a resolution-independent classifier, which we study in the following.



Images, Resolution and Scale For the continuum domain we consider the d-dimensional torus $\Omega = \mathbb{R}^d/\mathbb{Z}^d$. An image is a function

$$u : \Omega \to [0,1]^K$$

where $K \in \mathbb{N}$ denotes the number of color channels, see [GW87]. In order to represent images on a computer we discretize the domain Ω, via a regular grid $\Omega_N = \{x_0, \dots, x_N\}$ indexed by the set

$$\mathcal{J}_N := \{0, \dots, N-1\}^d$$

with grid points $x_j = j/(N-1)$, see e.g. [Kab22; KLM21]. For simplicity we assume an equal number of discretization points in each dimension. Therefore, N^d is the resolution of an image discretized w.r.t. $\mathcal{J}_N$.

It is important to notice the differences to the concept of *scale*. A change in scale would be a change in the original image domain, for example by zooming in on a certain area. For our image classification examples, we usually assume that the image u contain a certain entity to be classified. The scale roughly describes how "big" the entity is, or what percentage of the image it fills. We assume that the scale is fixed. Therefore, changing N directly influences the resolution of discretization.

How do Neural Networks react to Resolution Changes? In many machine learning application one assumes a fixed input size and therefore a fixed resolution, assuming the same scale throughout the data. Formally, networks are defined as mappings

$f_\theta : \mathbb{R}^{K\times N\times N} \to \Delta^C$, i.e. they only accept inputs for the fixed resolution N. In order to understand how this constraint can be weakened, we need to consider the concrete structure of our networks, which was similarly done in [KLM21; Kab22]. The architectures we consider are feed-forward networks of the form

$$f_\theta = \Phi^{\text{class}} \circ \mathcal{S} \circ \Phi^{\text{feature}}$$

where

- Φ^{feature} is the so-called feature extractor, which should be applicable independently from the input dimension,
- $\mathcal{S}$ is a function that maps inputs of any size to a fixed output dimension $\mathbb{R}^s$,
- $\Phi^{\text{class}} : \mathbb{R}^s \to \Delta^C$ denotes the classification layer.

The feature extractor usually consists of convolutional layers. As in [Kab22, Ch. 2] we consider the discrete convolution of two discretized images u_N, v_N defined as

$$(u_N * v_N)(x_j) := \sum_{k\in\mathcal{J}_N} u_N(x_k)\cdot v_N(x_{j-k})$$

where for negative indices $j-k$ we set $x_{j-k} := x_{j-k+N}$. This assumes that both u_N, v_N live on the same discretization $\mathcal{J}_N$. Modeling spatial locality—and therefore a small support of the kernel—one usually chooses M much smaller than the resolution. See e.g. the study in [HW62] which explores a similar methodology for the visual cortex of cats. In order to account for this dimension mismatch we consider so-called *spatial zero-padding* for kernels $\theta \in \mathbb{R}^{\mathcal{J}_M}$

$$\theta_k^{M\to N} = \begin{cases} \theta_k & \text{for } k \in \mathcal{J}_N \cap \mathcal{J}_M, \\ 0 & \text{for } k \in \mathcal{J}_N \backslash \mathcal{J}_M. \end{cases}$$

Using this method, one can define the convolution for inputs u_N of arbitrary input resolution N via

$$C(\theta)(u_N) = \theta^{M\to N} * u_N.$$

In [FNO] we refer to this as the *spatial implementation* of convolution. Up to the behavior on the boundary this is in fact the standard implementation in most libraries, especially in `PyTorch`. Therefore, a feature extractor consisting of convolutional layers can take inputs of variable resolution. In fact, ignoring possible resolution changes within the extractor—i.e. via pooling or strided convolutions—we have that $\Phi^{\text{feature}}(u_N) \in \mathbb{R}^{\mathcal{J}}_N$ for any $N \in \mathbb{N}$.

The mapping $\mathcal{S}$ can be realized as an adaptive pooling layer, see [Pas+19], which ensures a fixed output size. This methodology yields an *discretization invariant architecture*, see [Kab22; KLM21; Li+20], which means that from a technical point of view the network is able to produce outputs for inputs with arbitrary discretization. However, we are actually interested in *discretization invariant functionality* (see [Kab22; KLM21; Li+20]), which also requires that the output is the same over different resolutions.

Input Interpolation The technical possibility to handle different input sizes as described above, usually does not perform well in practice. This is due to the fact that in the standard spatial implementation of convolution the support of the kernel changes with varying input dimension, for which it output differs, see Fig. 3.3. If a network is trained on a fixed input size, the filters are adapted to this size and therefore only create meaningful responses on this size.

A simple attempt to create a network, where not only the architecture, but also the functionality is discretization independent—at least up to a certain degree—is input interpolation. In this case our architecture is modified to

$$\tilde{f}_\theta = f_\theta \circ I$$

where $I : \bigcup_{M\in\mathbb{N}} \mathbb{R}^{\mathcal{J}_M} \to \mathbb{R}^{\mathcal{J}_N}$ is an interpolation function, that maps inputs of arbitrary sizes to a fixed out discretization $\mathcal{I}_N$. Typical choices here include nearest neighbor, bilinear or bicubic interpolation, see e.g. [GW87]. Especially relevant in our case, is so-called *trigonometric interpolation*, where for $v_M \in \mathbb{R}^{\mathcal{J}}$ we define

$$I^{\text{trigo}}(v_M) := v_M^{M\xrightarrow{\Delta}N} := F^{-1}\left((Fv)^{M\to N}\right).$$

Contribution In [FNO] we study the connection between FNOs and CNNs for classification problems. We identify under which assumption the architectures are equivalent (see Lemma 3.29), but also where they are not, see Fig. 3.3. Here, we are especially interested in the multi-resolution case, we show that one layer of an FNO is equivariant with respect to trigonometric interpolation, Corollary 3.31. This is also underlined by numerical experiments, where we compare the FNO implementation to interpolation methods and a naive CNN implementation. Furthermore, we show that training equivalent CNN and FNO layers lead to different results, i.e., while they have the same forward pass, the gradients w.r.t. their parameters might differ, Lemma 3.30. Furthermore, we show continuity and Fréchet-differentiability of abstract neural layers as operators between L^p spaces, see Section 3.4.2. Finally we conduct numerical experiments supporting our theoretical findings Section 3.4.3.

3.4.1. Fourier Neural Operators

We want to obtain neural networks whose output does not depend on the discretization of an image $u : \Omega \to \mathbb{R}^K$. This raises the question whether it is possible to find a formulation that allows us to work in the infinite dimensional setting. In [Kov+21] this issue was addressed in the setting of parametric PDEs for which the authors introduced the concept of *neural operators*. One layer of a neural operator is given as a mapping $\mathcal{G} : L^p(\Omega) \to L^q(\Omega)$

$$\mathcal{G}(u)(x) = \sigma\left(\Psi(u)(x)\right) \qquad \text{for a.e. } x \in \Omega, \tag{3.34}$$

with the affine linear part given by

$$\Psi(u) = Wu + \mathcal{K}u + b, \tag{3.35}$$

where

- $\mathcal{K} : u \mapsto \int_\Omega \kappa(\cdot, y)\, u(y)\, dy$ is a kernel integral operator with kernel $\kappa : \Omega \times \Omega \to \mathbb{R}$,
- $W \in \mathbb{R}$ models a residual component,
- and $b : \Omega \to \mathbb{R}$ models a bias.

By a slight abuse of notation, the activation function $\sigma : \mathbb{R} \to \mathbb{R}$ acts as a Nemytskiioperator

$$\sigma : v \mapsto \sigma(v(\cdot)), \tag{3.36}$$

see e.g. [Trö10].

In Section 3.4.2 we analyze continuity and differentiablility of a layer in this abstract form. However, the most relevant case for us, is when $\mathcal{K}$ is a convolution operator, i.e. $\kappa(x, y) = \kappa(x - y)$ is a translation invariant kernel. In this special case $\mathcal{G}$ is then known as layer of a *Fourier Neural Operator* (FNO) as introduced in [Li+20], by parameterizing the kernel via its Fourier coefficients $\hat{\theta}_k \in \mathbb{C}$

$$\kappa_{\hat{\theta}}(x) = \sum_{k \in \mathcal{I}} \hat{\theta}_k\, b_k(x), \tag{3.37}$$

where $b_k(x) = \exp(2\pi i\, kx)$ denote the Fourier basis functions. In practice we assume that $\kappa_{\hat{\theta}}$ only has a finite amount of non-zero Fourier coefficients, in fact we choose the set $\mathcal{I}_N := \{-\lceil (N-1)/2 \rceil, \dots, 0, \dots, \lfloor (N-1)/2 \rfloor\}^d$ as the index set, as done in [Li+20]. Employing this set with odd $N \in \mathbb{N}$, we can easily enforce Hermitian symmetry $\hat{\theta}_k = \overline{\hat{\theta}_{-k}}$, which ensures that $\mathcal{K}_{\hat{\theta}}$ outputs real-valued functions. For now we assume an odd number here and deal with the even case later. We note that assuming a finite number N of Fourier coefficients is completely independent from the input discretization, which is the main advantage of FNOs.

How to Perform Convolutions with FNOs? In [FNO] we consider the discrete Fourier transform and its inverse

$$(Fv)_k = \frac{1}{\lambda} \sum_{j \in \mathcal{J}_N} v_j\, e^{-2\pi i \left\langle k, \frac{j}{N} \right\rangle}, k \in \mathcal{I}_N,$$
$$\left(F^{-1}\hat{v}\right)_j = \frac{\lambda}{|\mathcal{J}_N|} \sum_{k \in \mathcal{I}_N} \hat{v}_k\, e^{2\pi i \left\langle k, \frac{j}{N} \right\rangle} j \in \mathcal{J}_N,$$

with a normalization constant $\lambda \in \{1, \sqrt{|\mathcal{J}_N|}, |\mathcal{J}_N|\}$. We only consider parameters $\hat{\theta} \in \mathbb{C}^{\mathcal{I}_N}_{\text{sym}} := F(\mathbb{R}^{\mathcal{J}_N})$ for which can employ the convolution theorem (see e.g. [Gra14]) to define

$$K(\hat{\theta})(v) = F^{-1}\left(\hat{\theta} \cdot Fv\right) \qquad \text{for } v \in \mathbb{R}^{\mathcal{J}_N} \tag{3.38}$$

which is referred to as the FNO- or spectral-implementation of convolution.

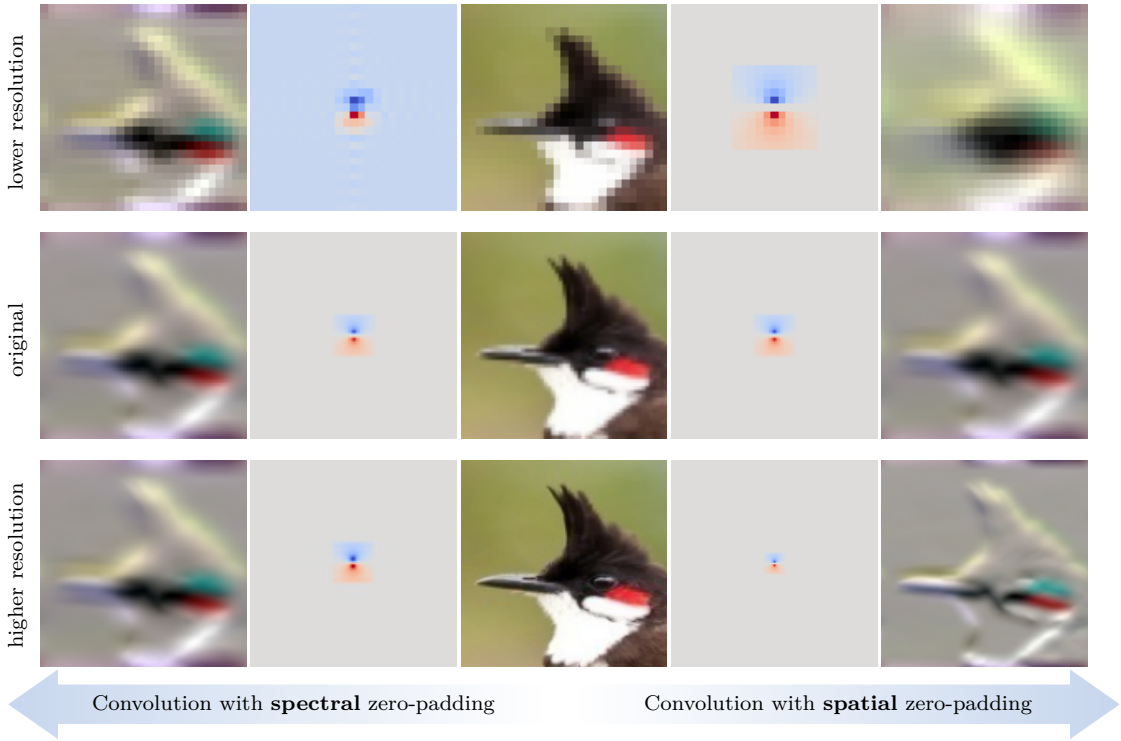


Figure 3.3.: The image is taken from [Kab22, Fig. 1]—depicting a red whiskered bulbul taken from the Birds500 dataset [Pio21]—and visualize the different effects of spectral and spatial zero-padding.

How do FNOs React to Resolution Changes? We already mentioned that the number of Fourier coefficients of $\hat{\theta} \in \mathbb{C}^{\mathcal{I}_M}_{\text{sym}}$ is independent of the input resolution. Nevertheless, the point-wise multiplication in Eq. (3.38) is only defined for inputs $v \in \mathbb{R}^{\mathcal{J}_M}$, for which the question arises how FNOs can be adapted to dimension mismatch. Here, we apply a conceptually similar idea, by employing zero-padding. However, the important and major difference is that this zero-padding is performed in the spectral domain. Assuming that $N > M$ is odd and that $v_N \in \mathbb{R}^{\mathcal{J}_N}$ we define

$$\hat{\theta}^{M\to N}_k = \begin{cases} \hat{\theta}_k & \text{for } k \in \mathcal{I}_N \cap \mathcal{I}_M, \\ 0 & \text{for } k \in \mathcal{I}_N \backslash \mathcal{I}_M, \end{cases}$$

and the spectral implementation of convolution as

$$K(\hat{\theta})(v_N) := K(\hat{\theta}^{M\to N})(v_N).$$

What Is The Difference Between Standard and FNO Implementation? The difference between the spectral and spatial implementation is best explained with [FNO, Fig. 1], which we repeat in Fig. 3.3 for convenience. We are given a kernel $\theta \in \mathbb{R}^{\mathcal{J}_M}$, its unnormalized Fourier transform $\hat{\theta} = \lambda F(\theta)$ and an input $v_N \in \mathbb{R}^{\mathcal{J}_N}$. If $M = N$, i.e. all

dimension match, we observe that spectral and spatial implementation are equivalent, see the middle row of Fig. 3.3. However, if we consider higher resolution variant of the image with $N > M$, spatial zero-padding results in the kernel being localized in space and therefore the effect of convolving it with v_N changes. On the other hand for the spectral implementation we observe an equivariant behaviour. The resolution of the output changes but qualitatively the effect of the filter stay the same.

Connection to Interpolation As hinted in Fig. 3.3, when changing the resolution, the spectral implementation of convolution can be interpreted as a standard convolution with an interpolated kernel. In fact we observe that for $\theta \in \mathbb{R}^{\mathcal{J}_M}, \hat{\theta} = \lambda F(\theta)$ and $v_N \in \mathbb{R}^{\mathcal{J}_N}$ we have that

$$\begin{aligned} K(\hat{\theta})(v_N) &= K(\hat{\theta}^{M\to N})(v_N) = F^{-1}\left(\hat{\theta}^{M\to N} \cdot F v_N\right) \\ &= F^{-1}\left(F\, F^{-1}\hat{\theta}^{M\to N} \cdot F v_N\right) \\ &= F^{-1}\left(F\theta^{M\xrightarrow{\Delta}N} \cdot F v_N\right) = \theta^{M\xrightarrow{\Delta}N} * v_N \\ &= C(\theta^{M\xrightarrow{\Delta}N})(v_N). \end{aligned}$$

Therefore, applying one FNO layer is equivalent to trigonometric interpolation of the kernel.

Adaption to Even Dimensions Zero-padding of the spectral coefficients only fulfills Hermitian symmetry in the case where M, N are odd. In order ot adapt this to the even case, we employ so-called Nyquist splitting, see [Bri19]. In all our experiments the implementation carefully employs this method, which ensures that the output of the spectral convolution is real valued.

3.4.2. Analytical Results for FNOs

In this section, we comment on the theoretical findings in [FNO]. We first consider the abstract neural layer as in Eq. (3.34) for which we show continuity and Fréchet-differentiability.

Continuity of Neural Layers The results for neural layers in [FNO] mostly fall back to the theory of Nemytskii operators, see e.g. [Trö10; AP93]. In order to show that the layer $\mathcal{G}$ is a well defined mapping from L^p to L^q one first needs to identify an exponent $r \in [1, \infty]$ such that the affine part is a mapping $\Psi : L^p \to L^r$. For example if $\kappa \in L^s$ with $1/r + 1 = 1/p + 1/s$ it follows from Young's convolution inequality that $\mathcal{K}$ maps to L^r, see [Gra14, Th. 1.2.12]. If then $W = 0$ and $b \in L^r$ we know that Ψ maps to L^r.

To ensure that the Nemytskii operator defines a mapping $\sigma : L^r \to L^q$ one needs to assume a growth condition on $\sigma : \mathbb{R} \to \mathbb{R}$, see [FNO, Eq. 4] and originally [Trö10]. Under these assumptions, we have the following result. The prove is trivial and the

main content is buried in the assumptions, however it still summarizes the situation in a convenient way. Concrete examples fulfilling these assumptions are given in [FNO] borrowing concepts from [AP93; Trö10]. The most important activation function that is valid in this setting is the ReLU function

$$\mathrm{ReLU}(x) = \max\{x, 0\}.$$

Proposition 3.4 ([FNO, Prop. 1]). For $1 \leq p, q \leq \infty$ let $\mathcal{L}$ be an operator layer given by (3.34) with an activation function $\sigma : \mathbb{R} \to \mathbb{R}$. If there exists $r \geq 1$ such that

(i) the affine part defines a mapping $\Psi : L^p(\Omega) \to L^r(\Omega)$,

(ii) the activation function σ generates a Nemytskii operator $\sigma : L^r(\Omega) \to L^q(\Omega)$,

then it holds that $\mathcal{L} : L^p(\Omega) \to L^q(\Omega)$. If additionally Ψ is a continuous operator on the specified spaces and the function σ is continuous, or uniformly continuous in the case $q = \infty$, the operator $\mathcal{L} : L^p(\Omega) \to L^q(\Omega)$ is also continuous.

Differentiability of Neural Layers We furthermore consider Fréchet differentiability of a neural layer w.r.t. to the input variable. This can also be transferred to differentiability w.r.t. the parameters as we provide in [FNO, Ex. 4]. Conceptually the main result we repeat here is similar to the one on continuity of the last paragraph. Namely, we put certain assumptions on the affine part and the activation function $\sigma : \mathbb{R} \to \mathbb{R}$ that allow us to employ the classical theory on Nemytskii operators. The major difference is that we also need to assume differentiability of the activation function, were we also assume a growth condition on the derivative. The ReLU function can therefore not be chosen in this setting, however the smooth approximation called GELU ([HG16])

$$\mathrm{GELU}(x) := x\,\Phi(x)$$

where Φ is the CDF of the standard normal distribution, can be employed.

Proposition 3.5 ([FNO, Prop. 2]). For $1 \leq p, q \leq \infty$, let $\mathcal{L}$ be an operator layer given by (3.34) with affine part Ψ as in (3.35). If there exists $r > q$, or $r = q = \infty$ such that

(i) the affine part is a continuous operator $\Psi : L^p(\Omega) \to L^r(\Omega)$,

(ii) the activation function $\sigma : \mathbb{R} \to \mathbb{R}$ is continuously differentiable

(iii) and the derivative of the activation function generates a Nemytskii operator $\sigma' : L^r(\Omega) \to [L^r(\Omega) \to L^s(\Omega)]$ with $s = rq/(r - q)$,

then it holds that $\mathcal{L} : L^p(\Omega) \to L^q(\Omega)$ is Fréchet-differentiable in any $v \in L^p(\Omega)$ with Fréchet-derivative $D\mathcal{L}(v) : L^p(\Omega) \to L^q(\Omega)$

$$D\mathcal{L}(v)(h) = \sigma'(\Psi(v)) \cdot \tilde{\Psi}(h),$$

where $\tilde{\Psi}$ denotes the linear part of Ψ, i.e., $\tilde{\Psi} = \Psi - b$.

Convertibility Between FNOs and CNNs The following lemma formalizes the intuition that FNOs and CNNs are equivalent in certain settings. Given inputs of a fixed discretization $\mathcal{J}_N$ and parameters $\theta \in \mathbb{R}^{\mathcal{J}_M}$ the standard convolution implementation is equivalent to the spectral one w.r.t. the parameters $\hat{\theta} = \lambda F(\theta^{M\to N})$. The subtle but important point here, is that the number of spectral parameters needs to be equal to the input size in order to achieve equivalence. In [FNO, Fig.3] we observe numerically that the number of used spectral coefficients actually needs to match the input size in order to achieve equivalence.

Lemma 3.29 ([FNO, Lem. 3]). Let $M \leq N$ both be odd and let $T : \mathbb{R}^{J_N} \to \mathbb{C}^{I_N}$ be defined for $\theta \in \mathbb{R}^{J_N}$ as $T(\theta) = \lambda F(\theta)$. For any $\theta \in \mathbb{R}^{J_M}$ and $v \in \mathbb{R}^{J_N}$ it holds true that

$$C(\theta)(v) = K(T(\theta^{M\to N}))(v)$$

and for any $\hat{\theta} \in \mathbb{C}^{I_M}_{\text{sym}}$ and $v \in \mathbb{R}^{J_N}$ it holds true that

$$K(\hat{\theta})(v) = C(T^{-1}(\hat{\theta}^{M\to N}))(v).$$

From this and from [FNO, Fig. 3] we obtain the negative result that in order to convert a CNN to a FNO we need a large number of spectral parameters. This also connected to the fact that spatial locality can usually only expressed using more spectral coefficients. Therefore, one might think that FNOs are infeasible due to memory requirements. However, it turns out that directly optimizing over the spectral parameters leads to a comparable performance, already for a low number of Fourier coefficients. This is reported in the blue curve in [FNO, Fig. 3]. This hints, that while the forward pass can be equivalent, computing the gradient w.r.t. their parameters is not. This is formalized in the following lemma.

Lemma 3.30 ([FNO, Lem. 4]). For odd $N \in \mathbb{N}$ and $v, \theta \in \mathbb{R}^{J_N}$ and $\hat{\theta} = T(\theta)$ it holds true that

$$\nabla_{\hat{\theta}} K(\hat{\theta})(v) = \frac{1}{|J_N|}\, T\left(\nabla_\theta C(\theta)(v)\right).$$

Interpolation Equivariance The last result considers the main motivation for the topic. Namely, the resolution in-variance of an FNO layer. However, here we can not allow an arbitrary resizing operation. Since it is most natural to our approach, we can show equivariance w.r.t. trigonometric interpolation of the input.

Corollary 3.31. For $\hat{\theta} \in \mathbb{C}_{\text{sym}}^{I_M}$, $v \in \mathbb{R}^{J_N}$, $M \leq N$ it holds true for any $L \geq M$ that

$$K(\hat{\theta})(v^{N \xrightarrow{\Delta} L}) = \left(K(\hat{\theta})(v)\right)^{N \xrightarrow{\Delta} L}.$$

3.4.3. Numerical Results

In the numerical section of [FNO] we study the convertibility of CNNs to FNOs as discussed in Section 3.4.2 and the resolution invariance of the different proposed approaches. Again we employ—among others—the `PyTorch` package [Pas+19]. The experiments are conducted on the FashionMNIST [XRV17] and a former version of the BIRDS500 [Pio21] dataset. We remark that our implementation carefully treats the case of even kernel or input sizes, via Nyquist splitting. This allows us to apply all the derived results for the odd and also the even case.

The code for all the experiments is available at https://github.com/samirak98/FourierImaging.

Convertibility and Training Differences As already described in Section 3.4.2 the first experiment, displayed in [FNO, Fig. 3], studies how CNNs can be converted to FNOs. Here, we employ a network with two convolutional layers for feature extraction and a linear classification layer. We train this network—in the spatial implementation—on the FashionMNIST dataset employing varying spatial kernel sizes. Here, we see that for $M = 5$ the spatial implementation already has the best performance and adding more parameters does not increase the performance. We then convert a set of spatial parameters to the Fourier formulation, employing again a varying number of spectral coefficients. It turns out that the requirement of [FNO, Lem. 3], that the number of coefficients must match the input dimension is indeed relevant. We only obtain the full performance of the CNN if we use all spectral parameters. However, the example also visualizes [FNO, Lem. 4], namely that training a FNO conceptually leads to a different set of parameters. Optimizing over the spectral parameters yields a comparable performance already for a smaller number of coefficients.

Resolution Invariance In the second experiment we study the resolution invariance of the three discretization independent architectures we considered, namely the naive CNN adaptation, input interpolation and the FNO implementation. We first train the convolutional architecture as in the previous paragraph on the FashionMNIST dataset,

that has an input size of 28×28. We resize the input data via trigonometric and bilienear interpolation—referred to as data sizing—to simulate multi-resolution data. One expects that the classification performance drops when the data is resized to a smaller size, since this step looses information. However, resizing the images to higher resolutions should yield the same performance.

In [FNO, Fig. 4] see that the simple adaption does not perform well both in the lower and the higher resolution setting. Employing input interpolation improves the performance in the lower resolution setting. In the higher resolution regime, we obtain a constant performance which is expected. Finally we see that the FNO—which was converted from the CNN—performs as well as input interpolation, which justifies the resolution independence of this architecture.

In the second experiments we trained a ResNet18 [He+16a] on a former version of the BIRDS500 dataset, with an input size of 112×112. Concerning the naive adaption and the input interpolation we observe the same behavior as in the previous example. However, the FNO variant performs slightly worse, which is surprising, especially in the higher resolution regime. In [FNO] we conclude that this is due the dimension changes within the ResNet architectures. These changes occur due to strided convolutions or pooling operations between the layers. In fact in order to achieve the performance as displayed in [FNO, Fig. 4 (b)] we replaced every striding by a trigonometric interpolation, which fits better into the FNO framework and vastly improves the performance. Therefore, we summarize that architecture intern dimension changes—which are very common in practice—can potentially hinder resolution invariance.

Part II.

Prints

Bibliography

Books

[Bre+84] L. Breiman et al. *Clasificaion and regression trees*. CRC Press, New York, 1984.

[Zhu05] X. Zhu. *Semi-supervised learning with graphs*. Carnegie Mellon University, 2005.

[Lin17] P. Lindqvist. *Notes on the p-Laplace equation*. 161. University of Jyväskylä, 2017.

[AF03] R. A. Adams and J. J. Fournier. *Sobolev spaces*. Elsevier, 2003.

[BB11] H. Brezis and H. Brézis. *Functional analysis, Sobolev spaces and partial differential equations*. Vol. 2. 3. Springer, 2011.

[Sch69] J. T. Schwartz. *Nonlinear Functional Analysis* -. Boca Raton, Fla: CRC Press, 1969.

[Bra02] A. Braides. *Gamma-convergence for Beginners*. Vol. 22. Oxford University Press, Oxford, 2002.

[Dal12] G. Dal Maso. *An introduction to* Γ*-convergence*. Vol. 8. Springer Science & Business Media, 2012.

[DS88] N. Dunford and J. T. Schwartz. *Linear operators, part 1: general theory*. Vol. 10. John Wiley & Sons, 1988.

[COR98] G. Cybenko, D. P. O'Leary, and J. Rissanen. *The mathematics of information coding, extraction and distribution*. Vol. 107. Springer Science & Business Media, 1998.

[Dac07] B. Dacorogna. *Direct methods in the calculus of variations*. Vol. 78. Springer Science & Business Media, 2007.

[SB14] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[VD95] G. Van Rossum and F. L. Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[Roc97] R. Rockafellar. *Convex analysis*. Princeton, N.J: Princeton University Press, 1997.

[BC11] H. Bauschke and P. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. New York: Springer, 2011.

[Eul24] L. Euler. *Institutionum calculi integralis*. Vol. 1. impensis Academiae imperialis scientiarum, 1824.

[BV04] S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[Ral81] L. B. Rall. *Automatic differentiation: Techniques and applications*. Springer, 1981.

[GW87] R. C. Gonzales and P. Wintz. *Digital image processing*. Addison-Wesley Longman Publishing Co., Inc., 1987.

[Trö10] F. Tröltzsch. *Optimal Control of Partial Differential Equations: Theory, Methods, and Applications*. Vol. 112. Graduate Studies in Mathematics. American Mathematical Society, Providence, Rhode Island, 2010.

[Gra14] L. Grafakos. *Classical Fourier Analysis*. 3rd ed. Graduate Texts in Mathematics. Springer, New York, NY, 2014.

[AP93] A. Ambrosetti and G. Prodi. *A Primer of Nonlinear Analysis*. Cambridge University Press, 1993.

Articles

[LIP-I] T. Roith and L. Bungert. "Continuum limit of Lipschitz learning on graphs." In: *Foundations of Computational Mathematics* (2022), pp. 1–39.

[LIP-II] L. Bungert, J. Calder, and T. Roith. "Uniform convergence rates for Lipschitz learning on graphs." In: *IMA Journal of Numerical Analysis* (Sept. 2022). DOI: 10.1093/imanum/drac048.

[BREG-I] L. Bungert et al. "A bregman learning framework for sparse neural networks." In: *Journal of Machine Learning Research* 23.192 (2022), pp. 1–43.

[Bol68] L. Boltzmann. "Studien über das Gleichgewicht der lebenden Kraft." In: *Wissenschafiliche Abhandlungen* 1 (1868), pp. 49–96.

[ST14] A. Subramanya and P. P. Talukdar. "Graph-based semi-supervised learning." In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 8.4 (2014), pp. 1–125.

[Ste+56] H. Steinhaus et al. "Sur la division des corps matériels en parties." In: *Bull. Acad. Polon. Sci* 1.804 (1956), p. 801.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm." In: *Journal of the royal statistical society: series B (methodological)* 39.1 (1977), pp. 1–22.

[GS15] N. García Trillos and D. Slepčev. "Continuum Limit of Total Variation on Point Clouds." In: *Archive for Rational Mechanics and Analysis* 220.1 (2015), pp. 193–241. DOI: 10.1007/s00205-015-0929-z.

[SB09] A. Szlam and X. Bresson. "A total variation-based graph clustering algorithm for cheeger ratio cuts." In: *UCLA Cam report* (2009), pp. 09–68.

[Gar+16] N. García Trillos et al. "Consistency of Cheeger and ratio graph cuts." In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 6268–6313.

[GMT22] N. García Trillos, R. Murray, and M. Thorpe. "From graph cuts to isoperimetric inequalities: Convergence rates of Cheeger cuts on data clouds." In: *Archive for Rational Mechanics and Analysis* 244.3 (2022), pp. 541–598.

[GS18] N. García Trillos and D. Slepčev. "A variational approach to the consistency of spectral clustering." In: *Applied and Computational Harmonic Analysis* 45.2 (2018), pp. 239–281.

[CV95] C. Cortes and V. Vapnik. "Support-vector networks." In: *Machine learning* 20 (1995), pp. 273–297.

[MS63] J. N. Morgan and J. A. Sonquist. "Problems in the analysis of survey data, and a proposal." In: *Journal of the American statistical association* 58.302 (1963), pp. 415–434.

[Ros58] F. Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[MP69] M. Minsky and S. Papert. "An introduction to computational geometry." In: *Cambridge tiass., HIT* 479 (1969), p. 480.

[Sch15] J. Schmidhuber. "Deep learning in neural networks: An overview." In: *Neural Networks* 61 (2015), pp. 85–117. DOI: https://doi.org/10.1016/j.neunet.2014.09.003.

[ST19] D. Slepcev and M. Thorpe. "Analysis of p-Laplacian regularization in semisupervised learning." In: *SIAM Journal on Mathematical Analysis* 51.3 (2019), pp. 2085–2120.

[Cal19] J. Calder. "Consistency of Lipschitz learning with infinite unlabeled data and finite labeled data." In: *SIAM Journal on Mathematics of Data Science* 1.4 (2019), pp. 780–812.

[FCL19] M. Flores, J. Calder, and G. Lerman. "Algorithms for Lp-based semisupervised learning on graphs." In: *arXiv preprint arXiv:1901.05031* (2019).

[CT22] J. Calder and N. G. Trillos. "Improved spectral convergence rates for graph Laplacians on ε-graphs and k-NN graphs." In: *Applied and Computational Harmonic Analysis* 60 (2022), pp. 123–175.

[ACJ04] G. Aronsson, M. Crandall, and P. Juutinen. "A tour of the theory of absolutely minimizing functions." In: *Bulletin of the American mathematical society* 41.4 (2004), pp. 439–505.

[ETT15] A. Elmoataz, M. Toutain, and D. Tenbrinck. "On the p-Laplacian and ∞-Laplacian on graphs with applications in image and data processing." In: *SIAM Journal on Imaging Sciences* 8.4 (2015), pp. 2412–2451.

[NSZ09] B. Nadler, N. Srebro, and X. Zhou. "Statistical analysis of semi-supervised learning: The limit of infinite unlabelled data." In: *Advances in neural information processing systems* 22 (2009).

[AL11] M. Alamgir and U. Luxburg. "Phase transition in the family of p-resistances." In: *Advances in neural information processing systems* 24 (2011).

[VBB08] U. Von Luxburg, M. Belkin, and O. Bousquet. "Consistency of spectral clustering." In: *The Annals of Statistics* (2008), pp. 555–586.

[GK06] E. Giné and V. Koltchinskii. "Empirical graph Laplacian approximation of Laplace-Beltrami operators: large sample results." In: *Lecture Notes-Monograph Series* (2006), pp. 238–259.

[vLB04] U. von Luxburg and O. Bousquet. "Distance-Based Classification with Lipschitz Functions." In: *J. Mach. Learn. Res.* 5.Jun (2004), pp. 669–695.

[Jen93] R. Jensen. "Uniqueness of Lipschitz extensions: minimizing the sup norm of the gradient." In: *Archive for Rational Mechanics and Analysis* 123 (1993), pp. 51–74.

[Kir34] M. Kirszbraun. "Über die zusammenziehende und Lipschitzsche Transformationen." In: *Fundamenta Mathematicae* 22 (1934), pp. 77–108. DOI: 10.4064/fm-22-1-77-108.

[Whi92] H. Whitney. "Analytic extensions of differentiable functions defined in closed sets." In: *Hassler Whitney Collected Papers* (1992), pp. 228–254.

[McS34] E. J. McShane. "Extension of range of functions." In: (1934).

[Aro67] G. Aronsson. "Extension of functions satisfying Lipschitz conditions." In: *Arkiv för Matematik* 6.6 (1967), pp. 551–561.

[Kur22] C. Kuratowski. "Sur l'opération A de l'analysis situs." In: *Fundamenta Mathematicae* 3.1 (1922), pp. 182–199.

[ČFK66] E. Čech, Z. Frolík, and M. Katětov. "Topological spaces." In: (1966).

[DF75] E. De Giorgi and T. Franzoni. "Su un tipo di convergenza variazionale." In: *Atti della Accademia Nazionale dei Lincei. Classe di Scienze Fisiche, Matematiche e Naturali. Rendiconti* 58.6 (1975), pp. 842–850.

[Mod77] L. Modica. "Un esempio di Γ-convergenza." In: *Boll. Un. Mat. Ital. B* 14 (1977), pp. 285–299.

[CGL10] A. Chambolle, A. Giacomini, and L. Lussardi. "Continuous limits of discrete perimeters." In: *ESAIM: Mathematical Modelling and Numerical Analysis* 44.2 (2010), pp. 207–230.

[BY12] A. Braides and N. K. Yip. "A quantitative description of mesh dependence for the discretization of singularly perturbed nonconvex problems." In: *SIAM Journal on Numerical Analysis* 50.4 (2012), pp. 1883–1898.

[VB+12] Y. Van Gennip, A. L. Bertozzi, et al. "Γ-convergence of graph Ginzburg-Landau functionals." In: *Adv. Differential Equations* 17.11-12 (2012), pp. 1115–1180.

[San15] F. Santambrogio. "Optimal transport for applied mathematicians." In: *Birkäuser, NY* 55.58-63 (2015), p. 94.

[Goo52] I. J. Good. "Rational decisions." In: *Journal of the Royal Statistical Society: Series B (Methodological)* 14.1 (1952), pp. 107–114.

[SGS15] R. K. Srivastava, K. Greff, and J. Schmidhuber. "Highway networks." In: *arXiv preprint arXiv:1505.00387* (2015).

[BREG-II] L. Bungert et al. "Neural Architecture Search via Bregman Iterations." In: (2021). arXiv: 2106.02479 [cs.LG].

[Rie22] K. Riedl. "Leveraging memory effects and gradient information in consensus-based optimization: On global convergence in mean-field law." In: *arXiv preprint arXiv:2211.12184* (2022).

[Pin+17] R. Pinnau et al. "A consensus-based model for global optimization and its mean-field limit." In: *Mathematical Models and Methods in Applied Sciences* 27.01 (2017), pp. 183–204.

[Car+21] J. A. Carrillo et al. "A consensus-based global optimization method for high dimensional machine learning problems." In: *ESAIM: Control, Optimisation and Calculus of Variations* 27 (2021), S5.

[Cau+47] A. Cauchy et al. "Méthode générale pour la résolution des systemes d'équations simultanées." In: *Comp. Rend. Sci. Paris* 25.1847 (1847), pp. 536–538.

[RM51] H. Robbins and S. Monro. "A stochastic approximation method." In: *The annals of mathematical statistics* (1951), pp. 400–407.

[GSS14] I. J. Goodfellow, J. Shlens, and C. Szegedy. "Explaining and harnessing adversarial examples." In: *arXiv preprint arXiv:1412.6572* (2014).

[Sha+18] A. Shafahi et al. "Are adversarial examples inevitable?" In: *arXiv preprint arXiv:1809.02104* (2018).

[FFF18] A. Fawzi, H. Fawzi, and O. Fawzi. "Adversarial vulnerability for any classifier." In: *Advances in neural information processing systems* 31 (2018).

[Sta+21] J. Stanczuk et al. "Wasserstein GANs work because they fail (to approximate the Wasserstein distance)." In: *arXiv preprint arXiv:2103.01678* (2021).

[Bun+23] L. Bungert et al. "It begins with a boundary: A geometric view on probabilistically robust learning." In: *arXiv preprint arXiv:2305.18779* (2023).

[Eng+18] L. Engstrom et al. "A rotation and a translation suffice: Fooling cnns with simple transformations." In: (2018).

[Guo+17] C. Guo et al. "Countering adversarial images using input transformations." In: *arXiv preprint arXiv:1711.00117* (2017).

[ANR74] N. Ahmed, T. Natarajan, and K. R. Rao. "Discrete cosine transform." In: *IEEE transactions on Computers* 100.1 (1974), pp. 90–93.

[Yua+19] X. Yuan et al. "Adversarial examples: Attacks and defenses for deep learning." In: *IEEE transactions on neural networks and learning systems* 30.9 (2019), pp. 2805–2824.

[KGB16] A. Kurakin, I. Goodfellow, and S. Bengio. "Adversarial machine learning at scale." In: *arXiv preprint arXiv:1611.01236* (2016).

[Mad+17] A. Madry et al. "Towards deep learning models resistant to adversarial attacks." In: *arXiv preprint arXiv:1706.06083* (2017).

[BGM23] L. Bungert, N. García Trillos, and R. Murray. "The geometry of adversarial training in binary classification." In: *Information and Inference: A Journal of the IMA* 12.2 (2023), pp. 921–968.

[LeC+95] Y. LeCun et al. "Learning algorithms for classification: A comparison on handwritten digit recognition." In: *Neural networks: the statistical mechanics perspective* 261.276 (1995), p. 2.

[Has+20] M. Hasannasab et al. "Parseval proximal neural networks." In: *Journal of Fourier Analysis and Applications* 26 (2020), pp. 1–31.

[Gou+20] H. Gouk et al. "Regularisation of neural networks by enforcing Lipschitz continuity." In: *Machine Learning* (2020), pp. 1–24.

[KMP20] V. Krishnan, A. A. A. Makdah, and F. Pasqualetti. "Lipschitz Bounds and Provably Robust Training by Laplacian Smoothing." In: *arXiv preprint arXiv:2006.03712* (2020).

[Sha+19] A. Shafahi et al. "Adversarial training for free!" In: *Advances in Neural Information Processing Systems* 32 (2019).

[LC10] Y. LeCun and C. Cortes. "MNIST handwritten digit database." In: (2010).

[XRV17] H. Xiao, K. Rasul, and R. Vollgraf. "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms." In: *CoRR* abs/1708.07747 (2017).

[Hoe+21] T. Hoefler et al. "Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks." In: *J. Mach. Learn. Res.* 22.241 (2021), pp. 1–124.

[Gho+21] A. Gholami et al. "A survey of quantization methods for efficient neural network inference." In: *arXiv preprint arXiv:2103.13630* (2021).

[EMH19] T. Elsken, J. H. Metzen, and F. Hutter. "Neural architecture search: A survey." In: *The Journal of Machine Learning Research* 20.1 (2019), pp. 1997–2017.

[How+17] A. G. Howard et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." In: *arXiv preprint arXiv:1704.04861* (2017).

[Ban+18] R. Banner et al. "Scalable methods for 8-bit training of neural networks." In: *Advances in neural information processing systems* 31 (2018).

[CBD14] M. Courbariaux, Y. Bengio, and J.-P. David. "Training deep neural networks with low precision multiplications." In: *arXiv preprint arXiv:1412.7024* (2014).

[Sch92] J. Schmidhuber. "Learning complex, extended sequences using the principle of history compression." In: *Neural Computation* 4.2 (1992), pp. 234–242.

[HVD15] G. Hinton, O. Vinyals, and J. Dean. "Distilling the knowledge in a neural network." In: *arXiv preprint arXiv:1503.02531* (2015).

[LDS89] Y. LeCun, J. Denker, and S. Solla. "Optimal brain damage." In: *Advances in neural information processing systems* 2 (1989).

[CFP97] G. Castellano, A. M. Fanelli, and M. Pelillo. "An iterative pruning algorithm for feedforward neural networks." In: *IEEE transactions on Neural networks* 8.3 (1997), pp. 519–531.

[CM73] J. F. Claerbout and F. Muir. "Robust modeling with erratic data." In: *Geophysics* 38.5 (1973), pp. 826–844.

[Tib96] R. Tibshirani. "Regression shrinkage and selection via the lasso." In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[Nit14] A. Nitanda. "Stochastic proximal gradient descent with acceleration techniques." In: *Advances in Neural Information Processing Systems* 27 (2014), pp. 1574–1582.

[RVV20] L. Rosasco, S. Villa, and B. C. Vũ. "Convergence of stochastic proximal gradient algorithm." In: *Applied Mathematics & Optimization* 82.3 (2020), pp. 891–917.

[Moc+18] D. C. Mocanu et al. "Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science." In: *Nature communications* 9.1 (2018), pp. 1–12.

[DZ19] T. Dettmers and L. Zettlemoyer. "Sparse networks from scratch: Faster training without losing performance." In: *arXiv preprint arXiv:1907.04840* (2019).

[DYJ19] X. Dai, H. Yin, and N. K. Jha. "NeST: A neural network synthesis tool based on a grow-and-prune paradigm." In: *IEEE Transactions on Computers* 68.10 (2019), pp. 1487–1497.

[Fu+22] Y. Fu et al. "Exploring structural sparsity of deep networks via inverse scale spaces." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[Liu+21] S. Liu et al. "Sparse evolutionary deep learning with over one million artificial neurons on commodity hardware." In: *Neural Computing and Applications* 33.7 (2021), pp. 2589–2604.

[BB18] M. Benning and M. Burger. "Modern regularization methods for inverse problems." In: *Acta Numerica* 27 (2018), pp. 1–111.

[PB+14] N. Parikh, S. Boyd, et al. "Proximal algorithms." In: *Foundations and trends® in Optimization* 1.3 (2014), pp. 127–239.

[Sca+17] S. Scardapane et al. "Group sparse regularization for deep neural networks." In: *Neurocomputing* 241 (2017), pp. 81–89.

[CP08] P. L. Combettes and J.-C. Pesquet. "Proximal thresholding algorithm for minimization over orthonormal bases." In: *SIAM Journal on Optimization* 18.4 (2008), pp. 1351–1376.

[DDD04] I. Daubechies, M. Defrise, and C. De Mol. "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint." In: *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences* 57.11 (2004), pp. 1413–1457.

[FNW07] M. A. Figueiredo, R. D. Nowak, and S. J. Wright. "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems." In: *IEEE Journal of selected topics in signal processing* 1.4 (2007), pp. 586–597.

[Cha04] A. Chambolle. "An algorithm for total variation minimization and applications." In: *Journal of Mathematical imaging and vision* 20 (2004), pp. 89–97.

[CP11] A. Chambolle and T. Pock. "A first-order primal-dual algorithm for convex problems with applications to imaging." In: *Journal of mathematical imaging and vision* 40 (2011), pp. 120–145.

[Bre67] L. M. Bregman. "The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming." In: *USSR computational mathematics and mathematical physics* 7.3 (1967), pp. 200–217.

[De 93] E. De Giorgi. "New problems on minimizing movements." In: *Ennio de Giorgi: Selected Papers* (1993), pp. 699–713.

[Osh+05] S. Osher et al. "An iterative regularization method for total variation-based image restoration." In: *Multiscale Modeling & Simulation* 4.2 (2005), pp. 460–489.

[ROF92] L. I. Rudin, S. Osher, and E. Fatemi. "Nonlinear total variation based noise removal algorithms." In: *Physica D: nonlinear phenomena* 60.1-4 (1992), pp. 259–268.

[Bur+06] M. Burger et al. "Nonlinear inverse scale space methods." In: *Communications in Mathematical Sciences* 4.1 (2006), pp. 179–212.

[Bur+07] M. Burger et al. "Inverse total variation flow." In: *Multiscale Modeling & Simulation* 6.2 (2007), pp. 366–395.

[Yin+08] W. Yin et al. "Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing." In: *SIAM Journal on Imaging sciences* 1.1 (2008), pp. 143–168.

[COS09] J.-F. Cai, S. Osher, and Z. Shen. "Convergence of the linearized Bregman iteration for ℓ_1-norm minimization." In: *Mathematics of Computation* 78.268 (2009), pp. 2127–2136.

[Vil+23] S. Villa et al. "Implicit regularization with strongly convex bias: Stability and acceleration." In: *Analysis and Applications* 21.01 (2023), pp. 165–191.

[BT03] A. Beck and M. Teboulle. "Mirror descent and nonlinear projected subgradient methods for convex optimization." In: *Operations Research Letters* 31.3 (2003), pp. 167–175.

[NY83] A. S. Nemirovskij and D. B. Yudin. "Problem complexity and method efficiency in optimization." In: (1983).

[Nem+09] A. Nemirovski et al. "Robust stochastic approximation approach to stochastic programming." In: *SIAM Journal on optimization* 19.4 (2009), pp. 1574–1609.

[Nes83] Y. Nesterov. "A method for unconstrained convex minimization problem with the rate of convergence o (1/k2)." In: *Doklady ANSSSR* 269.3 (1983), pp. 543–547.

[Qia99] N. Qian. "On the momentum term in gradient descent learning algorithms." In: *Neural networks* 12.1 (1999), pp. 145–151.

[KB14] D. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization." In: *arXiv preprint arXiv:1412.6980* (2014).

[HR21] F. Hanzely and P. Richtárik. "Fastest rates for stochastic mirror descent methods." In: *Computational Optimization and Applications* 79 (2021), pp. 717–766.

[ZH18] S. Zhang and N. He. "On the convergence rate of stochastic mirror descent for nonsmooth nonconvex optimization." In: *arXiv preprint arXiv:1806.04781* (2018).

[DOr+21] R. D'Orazio et al. "Stochastic mirror descent: Convergence analysis and adaptive variants via the mirror stochastic Polyak stepsize." In: *arXiv preprint arXiv:2110.15412* (2021).

[AKL22] P.-C. Aubin-Frankowski, A. Korba, and F. Léger. "Mirror descent with relative smoothness in measure spaces, with application to sinkhorn and em." In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 17263–17275.

[Ben+21] M. Benning et al. "Choose your path wisely: gradient descent in a Bregman distance framework." In: *SIAM Journal on Imaging Sciences* 14.2 (2021), pp. 814–843.

[HZ93] G. E. Hinton and R. Zemel. "Autoencoders, minimum description length and Helmholtz free energy." In: *Advances in neural information processing systems* 6 (1993).

[KLM21] N. Kovachki, S. Lanthaler, and S. Mishra. "On universal approximation and error bounds for Fourier neural operators." In: *The Journal of Machine Learning Research* 22.1 (2021), pp. 13237–13312.

[HW62] D. H. Hubel and T. N. Wiesel. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex." In: *The Journal of physiology* 160.1 (1962), p. 106.

[Li+20] Z. Li et al. "Fourier neural operator for parametric partial differential equations." In: *arXiv preprint arXiv:2010.08895* (2020).

[Kov+21] N. B. Kovachki et al. "Neural Operator: Learning Maps Between Function Spaces." In: *arXiv:2108.08481* (2021).

[Bri19] T. Briand. "Trigonometric Polynomial Interpolation of Images." In: *Image Processing On Line* 9 (Oct. 2019), pp. 291–316.

[HG16] D. Hendrycks and K. Gimpel. "Gaussian Error Linear Units (GELUs)." In: *arXiv:1606.08415* (2016).

Theses

[Roi21] T. Roith. "Master thesis: Continuum limit of Lipschitz learning on graphs." MA thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg, 2021.

[Kab22] S. Kabri. "Fourier Neural Operators for Image Classification." MA thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg, 2022.

Online

[Pio21] G. Piosenka. *BIRDS 500 - SPECIES IMAGE CLASSIFICATION*. 2021. URL: %7Bhttps://www.kaggle.com/datasets/gpiosenka/100-bird-species%7D.

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

Erlangen, den 24.Juli 2023 .....

Tim Roith