Master Thesis

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

PRIOR MODELS FOR ROBUST ADVERSARIAL DEEP LEARNING

José Hilario

Supervisor: Boris Flach, Dr. rer. nat. habil.
Field of study: Open Informatics
Subfield: Artificial Intelligence
May 2019

Acknowledgements

I thank Dr. Boris Flach for his guidance in the preparation of this thesis. I thank Intellisys D. Corp and the Ministry of Higher Education, Science and Technology of Dominican Republic for providing me with the economical means for my sustenance during the course of these two years of studies. I thank my dearest Evgeniya Brichkova whose company gave life and color to constant hard work. I thank my family, especially my parents, my brothers and the members of my household of youth, for educating me to aim towards good values and meaningful pursuits. Above all, I thank G-d for everything.

Thank you.

Declaration

I hereby declare that the presented work was made independently, and that I have listed all the sources of information used within it, in accordance with the methodical instructions for observing the ethical principles in the preparation of the university thesis.

Prague, 24 May 2019

Abstract

Deep networks learned by standard methods of discriminative learning are susceptible to adversarial patterns. Training adversarially robust deep networks therefore requires new learning methods. One interesting option is to include appropriate prior knowledge that will either generalize in a stronger sense by using prior information, or restrict the search space explored by the learning method while, after training, penalizing examples that are unlikely under the data distribution.

The thesis aims at analyzing and comparing different types of prior knowledge with respect to their impact on adversarial robustness. A suitable option for convolutional deep networks is to introduce lateral interactions within the convolutional layers to reflect the assumption of spatial continuity.

Keywords: prior knowledge, robust learning, learning, adversarial examples, deep learning, stochastic neural networks, graphical models, Gibbs distributions, conditional random fields, Bayes methods, computer vision, inference mechanisms, optimization, higher-order inference, approximate Bayesian marginal inference, variational inference, submodular functions, robustness, DNN, CNN, machine learning, robust optimization, discriminative learning, lateral interactions

Supervisor: Boris Flach, Dr. rer. nat. habil.

Abstrakt

Hluboké sítě naučené standardními metodami diskriminačního učení jsou náchylné k protichůdným vzorům. Výcvik nepřátelsky robustních hlubokých sítí proto vyžaduje nové metody učení. Jednou zajímavou možností je zahrnout vhodné předchozí znalosti, které buď zobecní v silnějším smyslu, s využitím těchto předběžných informací, nebo omezí vyhledávací prostor prozkoumaný metodou učení, zatímco po tréninku penalizují příklady, které jsou nepravděpodobné v rámci distribuce dat.

Práce si klade za cíl analyzovat a porovnávat různé typy předchozích poznatků s ohledem na jejich vliv na robustnost oponentů. Vhodnou volbou pro konvoluční hluboké sítě je zavedení laterálních interakcí uvnitř konvolučních vrstev, které odrážejí předpoklad prostorové kontinuity.

Klíčová slova: předchozí znalosti, robustní učení, učení, Adversarial vzorky, hluboká učení, pravděpodobnostní neuronové sítě, grafické modely, Gibbs distribuce, podmíněné náhodná pole, Bayesovy metody, počítačové vidění, inferenční mechanismy, optimalizace, vyššího řádu závěr, přibližná Bayesian marginální závěr, variační inference, submodulové funkce, robustnost, DNN, CNN, strojové učení, robustní optimalizace, diskriminační učení, laterální interakce

Překlad názvu: Apriorní modely pro rubustní adversariální hluboké učení

Contents

1 Introduction **3**

2 Related Work **5**

3 Propositions **11**

4 Definitions **15**

5 Background Theory **17**

5.1 Graphical Models 17

5.1.1 Markov Random Fields 19

5.2 Variational Inference 22

5.2.1 Supermodular Functions 22

5.2.2 Supermodularity of the Ising Model 23

5.2.3 Frank-Wolfe algorithm 24

5.2.4 L-Field 26

5.3 Robust Optimization 28

5.3.1 Robust Learning 29

6 Proposed Methods **31**

6.1 Stochastic neural network with lateral interactions 31

7 Experiments **37**

7.1 BaselineCNN 37

7.2 CRFCNN 38

7.3 Parameters 40

7.4 Datasets 40

7.5 Training Settings 41

7.6 Trial Scenarios 41

8 Results **43**

8.1 Robustness Against Gaussian Noise 44

8.2 Robustness Against Adversarial Attacks 45

8.3 p-Robustness Against Adversarial Attacks 46

9 Conclusion **49**

9.1 Future work 51

Bibliography **53**

Abbreviations 57

A Project Specification **59**

Figures

2.1 Figure and text borrowed from the original paper by Tsipras et al. [48]. Visualization of the loss gradient with respect to input pixels. Recall that these gradients highlight the input features which affect the loss most strongly, and thus are important for the classifier's prediction. We observe that the gradients are significantly more interpretable for adversarially trained networks – they align well with perceptually relevant features. In contrast, for standard networks they appear very noisy. 10

3.1 Figure and text borrowed from the original paper by Madry et al. [33]. A conceptual illustration of “natural” vs. “adversarial” decision boundaries. Left: A set of points that can be easily separated with a simple decision boundary. Middle: The simple decision boundary does not separate the ℓ_∞-balls. Hence there are adversarial examples (the red stars) that will be misclassified. Right: Separating the ℓ_∞-balls requires a significantly more complicated decision boundary. The resulting classifier is robust to adversarial examples with bounded ℓ_∞-norm perturbations. 13

5.1 Figure and text borrowed from the original paper by Bach [3]. Illustration of Frank-Wolfe conditional gradient algorithm: starting from the initialization (a), in steps (b), (d), (f), (h), an extreme point on the polytope is found an in steps (c), (e), (g), (i), a line search is performed. Note the oscillations to converge to the optimal point. 26

7.1 Structure of the BaselineCNN, which corresponds to a Sigmoid Belief Network. The gray blocks represent, first, the input, and then the feature maps. The creamy yellow windows represent convolution operations. The red windows represent max pooling operations. The green window represents the average pooling operation. The output is a vector of probabilities given by softmax. 38

7.2 Structure of the CRFCNN, which corresponds to a CRF Belief Network. The gray blocks represent first the input, and then the feature maps. The creamy yellow windows represent convolution operations. The blue windows represent the CRF activations. The blue feature maps indicate that, after generating the feature maps, the CRF activation is executed on them. The green window represents the average pooling operation. The output is a vector of probabilities given by softmax. . . . 39

8.1 Accuracy response of each network to variable magnitudes of standard deviation σ of Gaussian noise. 44

8.2 Accuracy response of each network to variable magnitudes of maximum allowed ℓ_∞-norm perturbation ϵ. . . 45

8.3 Safe accuracy response of each network to variable magnitudes of maximum allowed ℓ_∞-norm perturbation ϵ. 46

8.4 Accepting rate of each network to variable magnitudes of maximum allowed ℓ_∞-norm perturbation ϵ. . . 47

Tables

Chapter 1

Introduction

Recent research [14] has shown that our most successful learning methods and models have a weakness that we were unaware of. Our state-of-the-art classifiers are not learning the true underlying concepts that determine the correct output; rather, they are learning to respond correctly to naturally occuring data only: when points that do not have high probability in the data distribution occur, the classifiers present an over-confident and unreliable behavior that can be exploited by malicious agents. The existence of adversarial examples calls into question the applicability of machine learning systems to high-stakes problems.

Adversarial examples can be created by taking a correctly classified image and adding small differentials to it, such that the difference between the new image and the original is imperceptible to the human eye, but the classifier labels with high confidence the adversarially modified image to an incorrect class. In domains where security is a big concern, such as automotive systems, finance, health-care, cyber-security, the adversarial problem poses a great threat, and it is considered of the highest importance to find a solution to it.

The main original idea of this thesis is to incorporate prior knowledge to the learning mechanisms of neural networks in order to bias their learning towards domain-specific robust generalizations achieved by training on the worst-case adversarial exemplars of each image from the training data. By combining ideas from robust optimization, energy-based graphical models and adversarial training, several propositions are derived that can be put to test as robust learning methodologies for image classification.

Chapter 2

Related Work

There is a growing body of scientific literature on the problem of adversarial examples. Starting in the year 2013 with Szegedy et al. [46], and continuing to this day, we are still working to understand the nature of adversarial examples. The pursuit for understanding, and for the development of permanent, reliable, robust solutions to this problem, has already widened our views on how neural networks achieve the state-of-the-art results they achieve, and on what is their behavior when working with high-dimensional data.

Robustness is the property that, if a testing example is similar to a training example, then the testing error should be similar to the training error. This notion is rooted in Robust Optimization (RO) theory (Ben-Tal et al., 2009 [4]). Robustness is at the core of the adversarial problem, and RO is frequently employed in defenses against adversarial examples. The link between robustness and generalization is made as early as 2010, by Xu and Mannor [50] wherein it is shown that a weaker notion of robustness is a necessary and sufficient condition for asymptotic generalizability of learning algorithms. Schmidt et al., 2018 [40] observe that adversarial examples are not at odds with the standard notion of generalization, and that to achieve adversarial robustness, a classifier must generalize in a stronger sense. Schmidt et al. show that the sample complexity of robust learning can be significantly larger than that of standard learning.

The role of the high dimensionality of the data is also a key issue that needs to be better understood w.r.t. the existence of adversarial examples. Khoury and Hadfield-Menell, 2018 [26] investigate the hypothesis that the existence of adversarial examples is due to the high-dimensional geometry of data manifolds, combined with the presence of low but non-zero error rates. The authors explained this phenomenon for a toy dataset by providing a theoretical upper bound on the average distance to the nearest error, in terms of the test error.

Nitin Bhagoji et al., 2017 [39] and Hendrycks and Gimpel, 2016 [20] have tried to fight adversarial examples by using dimensionality reduction techniques, arguing that the principal components with less variance are exploited in adversarial attacks. Though successful in small datasets like MNIST or CIFAR-10, this approach has been shown to decrease the performance of the network dramatically for complex tasks. Moosavi-Dezfooli et al., 2018 [36] and Guo et al., 2017 [18] have attempted to build preprocessing defenses, such as denoising. On the other hand Gu and Rigazio, 2014 [17] state that for any feed-forward architecture, such preprocessing defenses can always be circumvented by some attack as long as they can be incorporated in it. Particularily, if the input transformation defense (or any defense, for that matter) is differentiable, the adversary can incorporate it in any gradient-based attack.

Adversarial training, first proposed by Goodfellow et al., 2014 [14] and by Huang et al., 2015 [21], is a data augmentation scheme that aids the classifier to attain robust generalization against adversarial examples by augmenting the training data in the "most confusing" and "most helpful" manner. Adversarial training has later been formalized for large-scale usage by Kurakin et al., 2016b [29]. Theoretical grounds have been uncovered recently by Madry et al., 2017 [33] that validate and support adversarial training: they show that training with adversarial examples is approximately equivalent to solving the RO formulation of learning in an adversarially uncertain environment. Madry et al. also suggest a notion of computational tractability that limits the adversary to first-order methods. Further, Projected Gradient Descent (PGD) is proposed as the benchmark of first-order attacks. The conclusion appears at sight that we will have to consider a reasonable tradeoff between attack power and computational tractability, so that solutions can also become more realistic and applicable. We can find an analogous precendent to support this claim in the field of computational cryptography.

An adversarially trained network remains, however, differentiable. This implies the potential weakness that such a network can be attacked using gradient-based methods. It remains to be seen if such defense mechanisms could reach the level of performance where the possibility of being attacked by gradient-based methods is no longer a limitation, because the required level of distortion to craft a smallest adversarial example is already detectable. Accomplishing such a level of performance would mean that a solution to the adversarial examples problem is a possibility.

Ongoing research has brought us to the understanding that not only the generalization under the natural data distribution matters, but that, in the face of a potential adversarial environment, we should aim to generalize in a stronger sense: a network should be able to reason soundly about the examples that can be found outside of the natural data distribution, but that are similar enough to the examples within it, such that a human cannot see the difference between one and the other.

Up to this point, most studies have been conducted utilizing an ℓ_p-norm similarity metric. Sharif et al., 2018 [42] demonstrate that nearness of inputs as measured by ℓ_p-norms is neither necessary nor sufficient for perceptual similarity: the "perceptual distance" between two images that are near to each other according to an ℓ_p-norm can already be high enough so that they can frequently be identified by humans as representing different objects. There is an active interest in finding appropriate ways to measure perceptual similarity.

Luo et al., 2018 [32] introduce a new distance metric that considers human perceptual systems to evaluate the sensitivity of human eyes to image pixels, in order to guide the attacker to add perturbations with less chances of being detected. Snell et al., 2015 [44] propose the Multiscale Structural-Similarity Index Measure (MS-SSIM), a differentiable loss function that is better calibrated to human perceptual judgments of image quality.

Jang et al., 2017 [23] leverage computer-vision algorithms to develop better perceptual similarity metrics. The authors evaluate the quality of the adversarial examples using objective metrics: metrics which are independent from the optimization objective. Specifically, they use classic techniques of computer vision: 1) canny edge detection, wherein an indistinguishable perturbation should maintain the number of edges very close to the value found in the original image; 2) Fast Fourier Transform (FFT), wherein the difference on the high-frequency part of the spectrum is used as metric for feature corruption; and, 3) Histogram of Oriented Gradients (HOG), wherein the perturbations resulting in smaller HOG vector difference are considered to have better quality.

Jordan et al., 2019 [25] leverage perceptual similarity metrics like Learned Perceptual Image Patch Similarity (LPIPS) and Structural Similarity Index Measure (SSIM) to define a new threat model for adversarial attacks. They formalize different types of attacks that are orthogonal to each other in the sense that robustness to one doesn't imply robustness to the others: 1) delta additions, which is the class of ℓ_p-norm ϵ-bounded style attacks; 2) affine transformations: rotations, translations and dilations; and, 3) flow networks, which they use to constrain the ℓ_∞-norm between the coordinates of the original and the modified pixels. They show that combining flow attacks with delta attacks results in more perceptually indistinguishable attacks.

Goodfellow et al., 2014; Kurakin et al., 2016b; Liu et al., 2016; and Tramèr et al., 2017 [14, 29, 30, 47], discuss the phenomenon of transferability of adversarial examples between differently trained networks. The property of transferability raises an important concern since it implies that networks can be attacked even when the attacker has no access to their model. This phenomenon is further confirmed and studied by Madry et al., 2017 [33], where authors show that increasing the capacity of a network decreases the transferability of adversarial examples to it. This phenomenon was also observed for adversarially trained networks, where increasing the power of the adversary used for training, has also the effect of making the network more robust to transferred attacks.

Kurakin et al., 2016a [28] demonstrate that a high ratio of adversarial images obtained by white-box attacks –attacks wherein the attacker has full access to the network model and its defenses– remain being adversarials even when they are fed to the network through a camera in the physical world. Moreover, they expect that it will be possible to realize: 1) attacks using other kinds of physical objects besides images printed on paper; 2) attacks against different kinds of machine learning systems, such as sophisticated reinforcement learning agents; 3) attacks performed without access to the model's parameters and architecture (using the transferability property); and, 4) physical attacks that achieve a higher success rate by explicitly modeling the physical transformation during the adversarial example construction process.

Small rotations and translations have been found to successfully fool neural network-based classifiers on MNIST and CIFAR-10 by Fawzi and Frossard, 2015 [10]. Engstrom et al., 2017 [8] introduce the notion of robustness to spatial transformations. They show that even a small number of non-adaptive, randomly chosen, translational and rotational perturbations of the input are sufficient to considerably degrade the performance of the model. Engstrom et al. also found that pixel-based ℓ_∞-norm ϵ-bounded robustness does not imply any measure of spatial robustness, and that combining pixel-based and spatial transformation-based attacks has a cumulative effect. This is due to the aforementioned orthogonality of the attacks.

Adversarial examples can be found in several different domains: Carlini et al., 2016 [5] proposed an attack against speech recognition systems, where they show how to craft sounds that are difficult for humans to understand, but that can be interpreted by speech recognition systems as voice commands, such as "Call 911" or "Turn on airplane mode". Grosse et al., 2016 [16] introduce adversarial attacks against malware detecting systems, where adversarial examples are disguised as bening examples, in order to fool the detecting system.

Adversarial examples have been found to instantiate the property of universal transferablity. Moosavi-Dezfooli et al., 2016 [34] show the existence of small universal perturbations that can fool state-of-the-art classifiers on natural images. The authors highlighted several properties of universal perturbations, and proposed an iterative algorithm to generate them. Moreover, the authors showed that universal perturbations generalize well across different classification models, resulting in image-agnostic and network-agnostic perturbations. The existence of such perturbations is further explained by the authors in analyzing the correlation between different regions of the decision boundary. The studies of adversarial examples have provided us with insights on the geometry of the decision boundaries learned by machine learning systems in high-dimensional space.

The existence of universal adversarial examples in the context of semantic image segmentation is shown by Hendrik Metzen et al., 2017 [19]. Let us remark that image segmentation is used in high-stakes applications, such as automated driving and video surveillance. Xie et al., 2017 [49] study adversarial examples for semantic image segmentation and object detection that can be transferred across different datasets. Arnab et al., 2017 [2] show that *Mean-field Inference for Dense CRFs* confers robustness to untargeted attacks. Moreover, Arnab et al., 2015 show in a different paper [1] that Conditional Random Fields (CRFs) with carefully designed higher-order potentials, defined over cliques, can also be modelled as Convolutional Neural Network (CNN) layers when using mean field inference. This system presents the intuitive benefit that the classifier and the graphical model learn to optimally co-operate with each other during training. This establishes a precedent for the use of CRFs as a means to confer robustness to Convolutional Neural Networks (CNNs).

RO has been successfully used to formulate the problem of learning in an adversarial environment. An instantiation of RO is Distributionally Robust Optimization (DRO), a paradigm for decision making under uncertainty where the uncertain data are governed by a probability distribution that is itself subject to uncertainty. Fathony et al., 2018 [9] propose Adversarial Graphical Models (AGMs), a distributionally robust framework for leveraging graphical structure among variables. The proposed method is based on the DRO formulation: the training data is replaced with an adversarial version from the set of distributions that matches the statistical summaries of the training data.

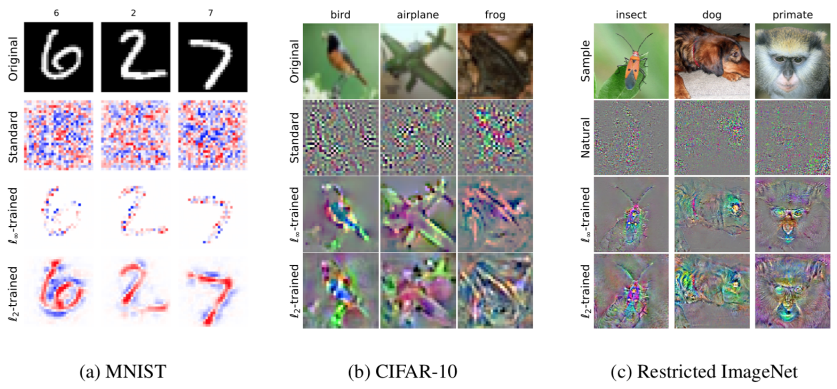


Figure 2.1: Figure and text borrowed from the original paper by Tsipras et al. [48]. Visualization of the loss gradient with respect to input pixels. Recall that these gradients highlight the input features which affect the loss most strongly, and thus are important for the classifier's prediction. We observe that the gradients are significantly more interpretable for adversarially trained networks – they align well with perceptually relevant features. In contrast, for standard networks they appear very noisy.

Tsipras et al., 2018 [48] bring some interesting results showing that features learned by robust models tend to align better with essential data characteristics, and with human perception. The authors suggest that the reason for this is that the features learned by the standard and robust classifiers are fundamentally different, which ends up resulting in a tradeoff between the standard accuracy and adversarially robust accuracy. In figure 2.1 we can see that the features learned by adversarially trained networks align well with perceptually relevant features of the input image; whereas, for networks trained under the standard procedure these gradients have no coherent patterns and appear very noisy to humans. This observation opens the door to new training methodologies for robust learning that take advantage of the required alignment of robust features with human perception. A suggested way to achieve this is by encoding the prior knowledge into the uncertainty set of each example, when considering the RO formulation.

Chapter 3

Propositions

Two main ideas are explored in this thesis.

The first idea is based on the intuition that the principle of spatial continuity of images also holds in the case of neuron activations; that is, neurons tend to become active together, as they detect meaningful features from the image. The results in [48] support this claim by showing that robust features tend to be more interpretable, and tend to follow the pattern of spatial continuity. It is also mentioned in [33] that robust learning requires networks with higher capacity. We claim that the network proposed in this thesis is of high enough capacity, and accomodates nicely the feature learning requirements of robust learning.

The second idea developed while investigating the most recent developments on the scientific literature regarding the problem of adversarial examples. Many principled and promising results have been obtained by reframing the learning problem to the framework of robust optimization RO. The opportunity to contribute to these developments was spotted in [33] when prior knowledge was proposed for future research as a way of constructing more meaningful uncertainty sets for RO. We consider the development of energy-based regularization through the construction of uncertainty sets for adversarial training, using energy-based models. Energy-based regularization can be implemented by using energy-based models to convey a notion of similarity from which the uncertainty set of an example can be obtained for adversarial training. We claim that the tradeoff between robustness and accuracy mentioned in [48] diminishes with the use of more precise uncertainty sets. Moreover, solving the saddle problem is known to increment the sample complexity required for learning [40]. By restricting the search space through the use of a more specific notion of similarity, in comparison to vanilla ℓ_p-norms, we can reduce the sample complexity required for robust learning. The second idea is summarized as robust learning with energy-based regularization through meaningful uncertainty sets.

The following list of claims summarizes the grounding principles for our approach.

Proposition 3.1 (Gilmer et al. 2018 [13]).

1. *Given enough data and a proper model class, it is possible to remove adversarial examples on a dataset.*
2. *The only way to reduce the frequency of adversarial examples is to reduce generalization error.*
3. *Adversarials are not a class: for any model with reasonable accuracy, most errors are "adversarial" relative to some example data point, in the sense that for a typical incorrectly classified point there is a small perturbation that will cause it to be correctly classified. Hence, there is no identifying characteristic of an adversarial example.*

Proposition 3.2 (Schmidt et al. 2018 [40]).

1. *The sample complexity required for robust learning can be significantly larger than that of standard learning. This gap is information theoretic and irrespective of the training algorithm or model family.*
2. *Current approaches may be unable to attain higher adversarial accuracy on datasets such as CIFAR-10 because the dataset is not large enough to train a standard convolutional network robustly.*
3. *Network architecture is a crucial factor for learning robustly with a limited number of examples.*
4. *Adversarial examples are not at odds with the standard notion of generalization. To achieve adversarial robustness, a classifier must generalize in a stronger sense.*

Proposition 3.3 (Tsipras et al. 2018 [48]).

1. *For a given adversary, we can always separate the features into robust (utilizing these features can only help robust classification) and non-robust (the adversary can manipulate these features to a degree where*

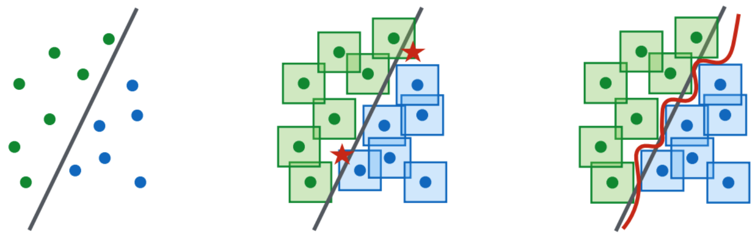


Figure 3.1: Figure and text borrowed from the original paper by Madry et al. [33]. A conceptual illustration of "natural" vs. "adversarial" decision boundaries. Left: A set of points that can be easily separated with a simple decision boundary. Middle: The simple decision boundary does not separate the ℓ_∞-balls. Hence there are adversarial examples (the red stars) that will be misclassified. Right: Separating the ℓ_∞-balls requires a significantly more complicated decision boundary. The resulting classifier is robust to adversarial examples with bounded ℓ_∞-norm perturbations.

they become harmful for the model's accuracy). A robust classifier cannot rely on non-robust features.

2. *Robustness and accuracy might be at odds: if there is any standard accuracy that can be gained by utilizing non-robust features, the model trained in standard way will benefit from it (at the expense of reducing its robust accuracy) and the robust model will not be able to get such a benefit. Therefore, its standard accuracy will be lower.*

Proposition 3.4 (Madry et al. 2017 [33])**.**

1. *A principled perspective on adversarial training: adversarial training with a good enough adversarial amounts to approximately solving the inner loss maximization task of the saddle problem in the RO framework.*

2. *Robustness against the PGD adversary yields robustness against all first-order adversaries, that is, attacks that rely only on first-order information. The limitation on first-order information is to be reminiscent of the notion of polynomially bounded adversary that is a cornerstone of modern cryptography.*

3. *A robust decision boundary (such as can be seen in figure 3.1) of the saddle point problem can be significantly more complicated than a decision boundary that simply separates the benign data points. Hence, classifiers with higher levels of capacity are needed.*

4. *Prior knowledge of the sample distribution should be used to construct the uncertainty set for each example.*

The claims this thesis aims to prove can be summarized in the following propositions:

Proposition 3.5 (Stochastic neural network with lateral interactions).
A convolutional neural network with stochastic and spatially correlated neurons (lateral interactions) is of high enough capacity to learn the complex boundaries required for robust learning.

Proposition 3.6 (Regularization through adversarial training).
Adversarial training is a form of regularization; i.e. it aids the classifier generalize in a stronger sense. Robust learning can make ammends for the lack of data required to "cover" the uncertainty set of each given image, where here "covering" means to learn the space occupied by variations that are sufficiently similar to each given example of the training data.

Proposition 3.7 (Energy-based models for generating uncertainty sets).
Energy-based models provide a meaningful notion of similarity that can be used to generate the uncertainty sets of the examples in the training data.

Chapter 4

Definitions

Let $\mathcal{D} \triangleq \mathbb{N}^2$ denote the convolutional grid of points (p, q) describing either the pixels of an image, or the neurons of a convolutional layer. Let $|\mathcal{D}| = n$ be its cardinality. Let $V \triangleq \{1, 2, ..., N\}$ be a set of nodes with cardinality $|V| = n$ where each node i is associated with a value x_i for $i \in V$, and with a coordinate of the grid $j \in \mathcal{D}$, such that $\{x_i = x_j \mid \forall i \in V, \exists^1 j \in \mathcal{D}\}$. The points of the grid will be referred to by $i \in V$. Let E denote the edges of the grid describing the connections between nodes.

Let us consider neural networks with l hidden layers denoted by $X^{(1,\cdots,l)}$, and an input layer denoted by $X^{(0)}$. At each layer k let us consider a corresponding $\mathcal{D}^{(k)}$ and $V^{(k)}$. Let $X_i^{(k)}$ denote a single unit in layer k. Let the events $X^{(k)} = \mathbf{x}^{(k)}$ be denoted unambiguously by $\mathbf{x}^{(k)}$. Let $\mathcal{L} \triangleq \{0, 1\}$ indicate inactive and active neurons, respectively. In the stochastic setting, the realizations of neuron activations in the hidden layers of the network take values $\mathbf{x}^{(k)} \in \mathcal{L}^V, k > 0$. We will use $\mathcal{S} \equiv \mathcal{L}^V$ for the sake of clarity. In the input layer $\mathbf{x}^{(0)}$, each channel of a normalized image contains pixels with values $x_i^{(0)} \in [0, 1]$. The whole image is thus defined as $\mathbf{x}^{(0)} \in [0, 1]^V$. Let $\mathcal{T} \triangleq \left\{(\mathbf{x}^{(0)}, \mathbf{y})^1, \ldots, (\mathbf{x}^{(0)}, \mathbf{y})^m\right\}$ denote the set of training examples. From now on, the parentheses will be dropped from the superindices and we will refer to $\mathbf{x}^{(k)}$ as $\mathbf{x}^k$. When convenient and unambiguous, $\mathbf{x}$ will be used to refer to $\mathbf{x}^0$ or $\mathbf{x}^k$.

When referring to a modular function $\mathbf{s} : \mathcal{S} \to \mathbb{R}$, note that we will say that $\mathbf{s} \in \mathbb{R}^{\mathcal{S}}$ when talking about it as a function $\mathbf{s}(\cdot)$, and we will say that $\mathbf{s} \in \mathcal{S}$ when talking about it as the parametrizing vector of the modular function.

Chapter 5

Background Theory

In this chapter we will go into the details required to understand the proposed methods of this thesis. We will discuss material from specific topics such as probabilistic graphical models, supermodular optimization, variational inference, and robust optimization.

5.1 Graphical Models

Following the account in [43], let us consider the **Belief Network** given by a neural network organized by layers. We are interested in the the propagation of statistics throughout the network, layer by layer. There are l layers of hidden random variables X^k for $k = 1, \cdots, l$. Each random variable X^k has n_k components. Let X^0 denote the input layer.

Definition 5.1 (Belief Networks)**.** A belief network is defined by:

$$P(X^{1,\cdots,l} \mid X^0) \triangleq \prod_{k=1}^{l} \prod_{i \in V^k} P(X_i^k \mid X^{k-1})$$

The posterior distribution of a belief network at each hidden layer $k > 1$ given the observation $\mathbf{x}^0$ recurrently expresses as:

$$\begin{aligned} P(X^k \mid \mathbf{x}^0) &= \mathbb{E}_{P(X^{k-1}|\mathbf{x}^0)}\Big[P(X^k \mid X^{k-1})\Big] \\ &= \int P(X^k \mid \mathbf{x}^{k-1}) \cdot P(\mathbf{x}^{k-1} \mid \mathbf{x}^0) \cdot d\mathbf{x}^{k-1} \end{aligned}$$

Proposition 5.2. *The best factorized approximation $Q(X^k) = \prod_{i \in V^k} Q(X_i^k)$ to $P(X^k \mid \mathbf{x}^0)$ in terms of the forward Kullback Leibler (KL) divergence $KL(P(X^k \mid \mathbf{x}^0) \parallel Q(X^k))$ is given by the marginals:*

$$Q(X_i^k) \triangleq P(X_i^k \mid \mathbf{x}^0) = \sum_{(x_j^k \in \mathcal{L} \mid j \in V^k, j \neq i)} P(X^k \mid \mathbf{x}^0) \tag{5.1}$$

Proof. Minimizing over distribution Q the expression $KL(P(X) \parallel Q(X)) = \mathbb{E}_{P(X)}[\log \frac{P(X)}{Q(X)}]$ is equivalent to maximizing $\mathbb{E}_{P(X)}[\log Q(X)]$. Assuming that $Q(X) = \prod_{i \in V} Q(X_i)$, the negative cross-entropy above is expressed as:

$$\sum_{\mathbf{x} \in \mathcal{S}} P(\mathbf{x}) \cdot \sum_{i \in V} \log Q(x_i) = \sum_{i \in V} \sum_{x_i \in \mathcal{L}} P(x_i) \cdot \log Q(x_i) \tag{5.2}$$

The **information inequality** theorem tells us that, for each $i \in V$, this expression is maximized when $Q(x_i) = P(x_i)$. $\square$

Theorem 5.3 (Information Inequality [37]). *$KL(P(X) \parallel Q(X)) \geq 0$ with equality if and only if $Q(X) = P(X)$.*

Proof. Let us invoke Jensen's inequality, which states that for any convex function f, we have that $f(\sum_{i=1}^n \lambda_i \cdot x_i) \leq \sum_{i=1}^n \lambda_i \cdot f(x_i)$, where $\lambda_i \geq 0$ and $\sum_{i=1}^n \lambda_i = 1$. Let $\mathcal{S}$ be the support of both $P(X)$, and $Q(X)$. Then:

$$\begin{aligned} -KL(P(X) \parallel Q(X)) &= -\sum_{\mathbf{x} \in \mathcal{S}} P(\mathbf{x}) \cdot \log \frac{P(\mathbf{x})}{Q(\mathbf{x})} = \sum_{\mathbf{x} \in \mathcal{S}} P(\mathbf{x}) \cdot \log \frac{Q(\mathbf{x})}{P(\mathbf{x})} \\ &\leq \log \sum_{\mathbf{x} \in \mathcal{S}} P(\mathbf{x}) \cdot \frac{Q(\mathbf{x})}{P(\mathbf{x})} = \log \sum_{\mathbf{x} \in \mathcal{S}} Q(\mathbf{x}) = \log 1 = 0 \end{aligned}$$

Intuitively, if $Q(\mathbf{x}) = P(\mathbf{x}), \forall \mathbf{x} \in \mathcal{S}$, then $\log \frac{P(\mathbf{x})}{Q(\mathbf{x})} = 0, \forall \mathbf{x} \in \mathcal{S}$. $\square$

Definition 5.4 (Feed-forward approximate inference [43]). The factorized approximation can be computed layer by layer, assuming that marginals of the previous layer were already approximated.

$$\begin{aligned} P(X^k \mid \mathbf{x}^0) &\approx \mathbb{E}_{Q(X^{k-1})}\left[P(X^k \mid X^{k-1})\right] \\ &\approx \int P(X^k \mid \mathbf{x}^{k-1}) \cdot \prod_{i \in V^{k-1}} Q(x_i^{k-1}) \cdot d\mathbf{x}^{k-1} \end{aligned}$$

Theorem 5.5 (Neal [38]). *A standard neural network which uses sigmoid activation functions can be seen as an approximation to the inference of* **Sigmoid Belief Networks**.

Proof. The probability of a single unit X_i^k becoming active is given by its sigmoid activation: $P(X_i^k = 1 \mid X^{k-1}) = \mathrm{Sig}(\mathbf{w}^T \cdot X^{k-1})$. Thus, the propagation layer by layer, as required for feed-forward approximate inference, can be obtained as:

$$\begin{aligned} \overline{X_i^k} &= \mu_i = \mathbb{E}_{Q(X^{k-1})}[P(X_i^k = 1 \mid X^{k-1})] \\ \mathrm{var}(X_i^k) &= \sigma_i^2 = \mu_i \cdot (1 - \mu_i) \end{aligned}$$

$\square$

Considering approximation **AP1** by Shekhovtsov et al. [43], we have that:

$$\mathbb{E}_{Q(X^{k-1})}\Big[\mathrm{Sig}(\mathbf{w}^T \cdot X^{k-1})\Big] \approx \mathrm{Sig}\Big(\mathbf{w}^T \cdot \mathbb{E}_{Q(X^{k-1})}[X^{k-1}]\Big) \tag{5.3}$$

Using the mean activations $\mathbb{E}_{Q(X^{k-1})}[X_i^{k-1}] = \overline{X_i^{k-1}} = Q(X_i^{k-1} = 1)$, the approximation can be written as $\overline{X^k} = \mathrm{Sig}\Big(\mathbf{w}^T \cdot \overline{X^{k-1}}\Big)$, which represents the standard propagation rule of a sigmoid layer.

5.1.1 Markov Random Fields

Let us consider the stochastic process with random variables $X = (X_1, \cdots, X_n)$, and probability distribution $P(X = \mathbf{x})$ where the realizations of the random variables $(x_1, \cdots, x_n) \in \mathcal{S}$ are structured in a Markov Random Field (MRF).

Definition 5.6 (MRF). An MRF is an undirected graphical model defined by a graph $G = (V, E)$ in which the neighbourhood $\mathcal{N}_i \subseteq V$ of node $i \in V$, known as the Markov blanket of i, defines the paths of conditional dependencies of i to the rest of the graph. This is known as the undirected local Markov property. More specifically, the process X is an MRF if:

$$P(X_i \mid X_j,\ j \neq i) = P(X_i \mid X_j,\ j \in \mathcal{N}_i) \tag{5.4}$$

Theorem 5.7 (Hammersley-Clifford [37]). *A positive distribution $P(\mathbf{x}) > 0,\ \forall \mathbf{x} \in \mathcal{S}$ can be represented as a product of factors $P(\mathbf{x}) = \prod_{c \in \mathcal{C}} \psi_c(\mathbf{x})$ if and only if it satisfies the following conditional independence properties of Undirected Graphical Models (UGM), for a graph G:*

1. **Undirected local Markov property**. *As described in the definition of MRF 5.6.*
2. **Factorization property**. *The distribution factorizes according to the cliques $c \in \mathcal{C}$ of the graph G.*
3. **Global Markov property**. $P(X_i \mid X_j,\, X_k) = P(X_i \mid X_j)$ *whenever* X_i *and* X_k *are separated by* X_j *in* G.

Definition 5.8 (Gibbs distribution [37])**.** Let $\mathcal{G} = (V, \mathcal{C})$ be a hypergraph. The Gibbs distribution can be written as $P(\mathbf{x}) = \mathcal{Z}^{-1} \cdot \prod_{c\in\mathcal{C}} \exp\left(f(x_c)\right)$, where $f(x_c) < 0$ is the **energy function** associated with the variables in the coordinates c given by the system of hyperedges $\mathcal{C}$, and $\mathcal{Z} = \sum_{\mathbf{x}\in\mathcal{S}} \exp\left(F(\mathbf{x})\right)$ is the normalizing constant, also called **partition function**.

We can write the Gibbs distribution as a product of factors $P(\mathbf{x}) = \prod_{c\in\mathcal{C}} \psi_c(\mathbf{x})$ by defining $\psi_c(\mathbf{x}) \triangleq \exp\left(f(x_c)\right)$. Then, an MRF has a Gibbs distribution, where the system of cliques $\mathcal{C}$ of G of an MRF correspond to the system of hyperedges $\mathcal{C}$ of the Gibbs distribution.

Ising model

The **Ising model** is an MRF arising from the field of statistical physics, which models the interaction (attractive or repulsive) between neighbouring nodes, in the presence of an external field. For the attractive case, its energy function is defined:

Definition 5.9 (Energy function of the attractive Ising model)**.**

$$F(\mathbf{x}) = F_\alpha(\mathbf{x}) + F_\beta(\mathbf{x}) = \sum_{ij\in E} F_{\alpha_{ij}}(\mathbf{x}) + F_\beta(\mathbf{x}) = -\sum_{ij\in E} \alpha_{ij} \cdot |x_i - x_j| + \langle \beta,\, \mathbf{x}\rangle$$

Where α_{ij} is called interaction parameter, and corresponds to the strength of the interactions between the nodes i and j; while β_i is called external field parameter, and corresponds to the strength of the influence of the external field, at each node $i \in V$. We will look at the case where $\alpha_{ij} = \alpha, \forall ij \in E$.

For CRF Belief Networks we will consider the more general case of a system of hyperedges $\mathcal{C}$ with higher-order interactions. Let $f(x_c) = \max_{i\in c} x_i - \min_{i\in c} x_i$ [27], then:

$$\boxed{F(\mathbf{x}) = -\alpha \cdot \sum_{c\in\mathcal{C}} f(x_c) + \langle \beta,\, \mathbf{x}\rangle} \tag{5.5}$$

CRF Belief Networks

Definition 5.10 (Conditional Random Field (CRF) [37])**.** A CRF, is a version of an MRF where all the clique potentials are conditioned on input features:

$$P(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \frac{1}{\mathcal{Z}(\mathbf{x}, \mathbf{w})} \cdot \prod_{c \in \mathcal{C}} \psi_c(\mathbf{y}_c \mid \mathbf{x}, \mathbf{w})$$

Let us consider two hidden layers X^{k-1} and X^k in a neural network with weights of layer k referred to as $\mathbf{w} = \mathbf{w}^k$. Let $\beta \triangleq \mathbf{w}^T \cdot X^{k-1}$. Then, the high-order **CRF** can be expressed as:

$$P(X^k \mid X^{k-1}) \propto \exp\left(F_\beta(X^k)\right) \cdot \prod_{ij \in E} \exp\left(F_{\alpha_{ij}}(X^k)\right)$$

$$P(X^k \mid X^{k-1}) \propto \exp\left(-\alpha \cdot \sum_{c \in \mathcal{C}} f(X_c^k) + \langle \beta,\ X^k \rangle \right)$$

Using feed-forward approximate inference and approximation **AP1** [43] we get that the required propagation of means is:

$$\begin{aligned} P(X^k \mid \overline{X^{k-1}}) &= \mathbb{E}_{Q(X^{k-1})}\Big[P(X^k \mid X^{k-1})\Big] \\ &= \mathbb{E}_{Q(X^{k-1})}\left[\frac{1}{\mathcal{Z}(X^{k-1}, \mathbf{w})} \exp\Big(F_\alpha(X^k) + F_\beta(X^k)\Big)\right] \end{aligned}$$

$$\boxed{P(X^k \mid \overline{X^{k-1}}) = \frac{1}{\mathcal{Z}(\overline{X^{k-1}}, \mathbf{w})} \cdot \exp\left(\sum_{c \in \mathcal{C}} f(X_c^k) + \langle \mathbf{w}^T \overline{X^{k-1}}, X^k \rangle\right)} \tag{5.6}$$

Remark 5.11. Considering a **CRF Belief Network**, where each neuron is a random variable X_i^k that is either active or inactive (binary case), we are interested in obtaining the mean activation value of each neuron, which corresponds to the probability of the neuron firing (i.e. becoming active), given the input $\mathbf{x}^0$. Applying lemma 5.2, definitions 5.1 and 5.4, and the **AP1** approximation, we get that:

$$\begin{aligned} P(X^k \mid \mathbf{x}^0) &= \mathbb{E}_{P(X^{k-1} \mid \mathbf{x}^0)}\Big[P(X^k \mid X^{k-1})\Big] \\ &\overset{\text{KL}}{\approx} \mathbb{E}_{Q(X^{k-1})}\Big[P(X^k \mid X^{k-1})\Big] \overset{\text{AP1}}{\approx} P\Big(X^k \mid \mathbb{E}_{Q(X^{k-1})}[X^{k-1}]\Big) \\ &\overset{\text{KL}}{\approx} Q(X^k) = \prod_{i \in V^k} Q(X_i^k) \end{aligned}$$

The task of calculating the marginals is known to be intractable [24]. Nonetheless, an efficient approximation using variational inference has been proposed in [51].

5.2 Variational Inference

The main idea of variational inference [37] is to reduce the problem of inference to an optimization problem over tractable distributions. Let $\mathcal{T}$ be the data we have. We pick an approximation $Q(\mathbf{x})$ to distribution $P(\mathbf{x} \mid \mathcal{T})$, from a tractable family. Then we optimize to make this approximation as close as possible to the posterior; i.e., the goal is to obtain $Q(\mathbf{x}) \approx P^*(\mathbf{x}) \triangleq P(\mathbf{x} \mid \mathcal{T})$.

Let us introduce some important concepts that will be necessary to understand the algorithm [7] used for variational inference in this thesis.

5.2.1 Supermodular Functions

Let us consider the space $\mathbb{R}^V \triangleq \{y \mid y : 2^V \to \mathbb{R}\}$. For any vector $y \in \mathbb{R}^V$ and each $i \in V$, $y(\{i\})$ denotes the component of y associated with i. For a subset $A \subseteq V$, let us define $y(A) \triangleq \sum_{i \in A} y(\{i\})$. Then y satisfies $y(\varnothing) = 0$ and can be identified with a function satisfying $y(A) + y(B) = y(A \cap B) + y(A \cup B)$.

Submodular functions are a family of set functions exhibiting a natural diminishing returns property, originating in the field of combinatorial optimization. Supermodular functions are complementary to submodular functions, in sign and inequality direction. We will focus on the supermodular variant, since it is more related to the work presented in this thesis.

Definition 5.12 (Supermodularity)**.** A function f is supermodular if:

$$f(A) + f(B) \leq f(A \cup B) + f(A \cap B), \ \forall A, B \subseteq V$$

Definition 5.13 (Supermodular polyhedra [11])**.** The **supermodular polyhedron** of a supermodular function f is the polyhedron defined by:

$$\mathcal{P}(f) \triangleq \{y \mid y \in \mathbb{R}^V, \forall A \subseteq V : \ y(A) \geq f(A)\}$$

The set $\mathcal{P}(f)$ is the set of all linear modular functions which are lower bounded by the function f.

Definition 5.14 (Base polyhedra [11])**.** The **base polyhedron** $\mathcal{B}(f)$ is a subset of modular functions y which are bounded by f but agree with f on the full ground set V: that is, the base polyhedron is exactly the set of all modular upper bounds y of f that are tight at the ground set V. Since $\mathcal{B}(f)$

is bounded, it is also called base polytope.

The base polytope of a supermodular system f is defined by:

$$\mathcal{B}(f) \triangleq \{y \mid y \in \mathcal{P}(f),\ y(V) = f(V)\}$$

Theorem 5.15 (Translation of supermodular systems [11]). *For any vector $y \in \mathbb{R}^V$ the translation of a supermodular system by y is the supermodular system whose rank function is given by $f + y : 2^V \to \mathbb{R}$. Let the operator $+$ denote the Minkowski sum. We have that the translation results in:*

$$\mathcal{P}(f + y) = \mathcal{P}(f) + y$$
$$\mathcal{B}(f + y) = \mathcal{B}(f) + y$$

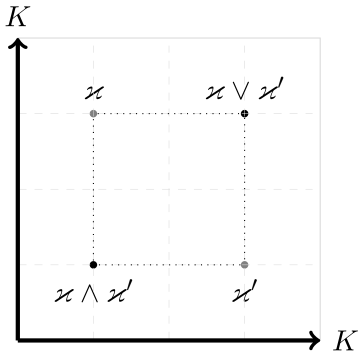
Definition 5.16 (Minkowski sum). The Minkowski sum of two sets A and B is defined as the set formed by adding each element from one set A, to each element of the other set B.

$$A + B \triangleq \{a + b \mid a \in A, b \in B\}, \forall a \in A, \forall b \in B$$

5.2.2 Supermodularity of the Ising Model

Let each K be completely ordered. Denote the infimum and supremum w.r.t. this ordering by $\wedge$ (meet operation) and $\vee$ (join operation), respectively. Now, K^n is a distributive lattice, partially ordered, with operations infimum and supremum, and closed w.r.t. meet and join operations.

$$\varkappa,\ \varkappa' \in K^n$$
$$\varkappa \wedge \varkappa' = (\varkappa_1 \wedge \varkappa'_1, \cdots, \varkappa_n \wedge \varkappa'_n)$$
$$\varkappa \vee \varkappa' = (\varkappa_1 \vee \varkappa'_1, \cdots, \varkappa_n \vee \varkappa'_n)$$



A real valued function $u : K^n \to \mathbb{R}$ is supermodular *if* $u(\varkappa) + u(\varkappa') \leq u(\varkappa \wedge \varkappa') + u(\varkappa \vee \varkappa')$ holds for all $\varkappa, \varkappa' \in K^n$.

Remark.

1. The condition established for supermodularity holds for submodularity by changing the comparison from $\geq$ to $\leq$.

2. Any function $u : K \to \mathbb{R}$ is both supermodular and submodular.
3. The sum of supermodular functions is also supermodular.

Supermodularity of the Ising model. Recall that $x_i^k \in \mathcal{L}$, where $\mathcal{L} \equiv \{0,1\}$, for element $i \in V^k$, $k > 0$, where k denotes the layer of a belief network. We are interested in evaluating the supermodularity of the energy function of the attractive Ising model. Considering the previous remarks, we take F_β to be supermodular. In order to determine the supermodularity of the attractive Ising model, it remains only to examine F_α.

$$
\begin{gathered}
F_\alpha(\mathbf{x}) = -\alpha \cdot \sum_{ij \in E} |x_i - x_j| = \alpha \cdot \sum_{ij \in E} u(x_i, x_j) \\
u(x_i, x_j) = -|x_i - x_j| \\
u(0,0) = u(1,1) = 0; \quad u(0,1) = u(1,0) = -1 \\
u(0,1) + u(1,0) \leq u(0,0) + u(1,1) \\
-2 \leq 0
\end{gathered}
$$

Therefore, the energy function of the Ising model is a supermodular function for the attractive case, and a submodular function for the repulsive case. □

5.2.3 Frank-Wolfe algorithm

Let us now introduce the Frank-Wolfe algorithm, which is used to perform inference in the variational inference approach proposed by Djolonga and Krause [7].

Suppose $\mathcal{S}$ is a compact convex set in a vector space, and $f : \mathcal{S} \to \mathbb{R}$ is a convex differentiable real-valued function. The Frank–Wolfe algorithm solves the optimization problem:

$$
\underset{\mathbf{x} \in \mathcal{S}}{\text{minimize}}\, f(\boldsymbol{x}) \tag{5.7}
$$

The Frank-Wolfe algorithm is a projection-free first-order method for smooth constrained optimization. The algorithm maintains feasibility by approaching the final solution through iterative updates made as convex combinations of points inside the feasible region, obtained using first-order Taylor-series approximations of f.

Definition 5.17 (First-order Taylor-series approximation [45])**.** Let the function $f : \mathcal{S} \to \mathbb{R}$ be differentiable, and its derivative ∇f, continuous. The value of $f(\mathbf{s})$ around $\mathbf{x} \in \mathcal{S}$ is approximated by the first-order Taylor series:

$$
F_{\mathbf{x},1}(\mathbf{s}) = f(\mathbf{x}) + \langle \nabla f(\mathbf{x}),\, \mathbf{s} - \mathbf{x} \rangle
$$

up to an additive error that vanishes as $\mathbf{s}$ approaches $\mathbf{x}$.

The Frank-Wolfe algorithm produces at each iteration a first-order Taylor-series approximation of f around the value $\mathbf{x}^{(t)}$. Then, it minimizes the value of this approximation over $\mathbf{s} \in \mathcal{S}$, in order to obtain a new point in the feasible domain $\mathbf{s}^{(t)} = \arg\min_{\mathbf{s}\in\mathcal{S}} f(\mathbf{x}^{(t)}) + \langle \nabla f(\mathbf{x}^{(t)}), \mathbf{s} - \mathbf{x}^{(t)} \rangle$, which is used to improve the solution of the algorithm $\mathbf{x}^{(t+1)} \triangleq (1-\gamma^{(t)}) \cdot \mathbf{x}^{(t)} + \gamma^{(t)} \cdot \mathbf{s}^{(t)}$. The algorithm converges when the duality gap is small enough ($\leq \epsilon$).

Definition 5.18 (Duality gap of the Frank-Wolfe algorithm [45])**.** $F_{\mathbf{x},1}(\mathbf{s})$ is linear and tangent with $f(\mathbf{s})$ at $\mathbf{x}$. The convexity of f implies that $F_{\mathbf{x},1}(\mathbf{s}) \leq f(\mathbf{s})$ for all $\mathbf{s} \in \mathcal{S}$. Let $\mathbf{x}^* \in \mathcal{S}$ be the optimal Frank-Wolfe solution, and $\mathbf{s}^{(t)} = \arg\min_{\mathbf{s}\in\mathcal{S}} f(\mathbf{x}^{(t)}) + \langle \nabla f(\mathbf{x}^{(t)}), \mathbf{s} - \mathbf{x}^{(t)} \rangle$. Then:

$$\begin{gathered} f(\mathbf{x}^{(t)}) + \langle \nabla f(\mathbf{x}^{(t)}), \mathbf{s} - \mathbf{x}^{(t)} \rangle \leq f(\mathbf{s}) \\ \langle \nabla f(\mathbf{x}^{(t)}), \mathbf{s} - \mathbf{x}^{(t)} \rangle \leq f(\mathbf{s}) - f(\mathbf{x}^{(t)}) \\ \langle \nabla f(\mathbf{x}^{(t)}), \mathbf{s}^{(t)} - \mathbf{x}^{(t)} \rangle \leq \langle \nabla f(\mathbf{x}^{(t)}), \mathbf{s} - \mathbf{x}^{(t)} \rangle \leq f(\mathbf{x}^*) - f(\mathbf{x}^{(t)}) \end{gathered}$$

$$\boxed{\langle \nabla f(\mathbf{x}^{(t)}), \mathbf{x}^{(t)} - \mathbf{s}^{(t)} \rangle \geq f(\mathbf{x}^{(t)}) - f(\mathbf{x}^*)} \tag{5.8}$$

The rightmost expression denotes the (unkown) true error of the algorithm, while the leftmost expression is our duality gap. We establish a maximum duality gap (close to zero) ϵ as a stopping criterion, such that if $\langle \nabla f(\mathbf{x}^{(t)}), \mathbf{x}^{(t)} - \mathbf{s}^{(t)} \rangle \leq \epsilon$, then we finish running the algorithm.

Algorithm 1 Frank-Wolfe

```
1: $\mathbf{x}^{(0)} \in \mathcal{S}$
2: for $t = 0, \cdots, T$ do
3:     $\mathbf{r}^{(t)} := \nabla f(\mathbf{x}^{(t)})$
4:     $\mathbf{s}^{(t)} := \arg\min_{\mathbf{s}\in\mathcal{S}} \langle \mathbf{s}, \mathbf{r}^{(t)} \rangle$
5:     $g_t := \langle \mathbf{x}^{(t)} - \mathbf{s}^{(t)}, \mathbf{r}^{(t)} \rangle$
6:     if $g_t \leq \epsilon$ then return $\mathbf{x}^{(t)}$          ▷ Small duality gap: $g^{(t)} \leq \epsilon$
7:     $\gamma^{(t)} := \frac{2}{2+t}$
8:     $\mathbf{x}^{(t+1)} := (1-\gamma^{(t)}) \cdot \mathbf{x}^{(t)} + \gamma^{(t)} \cdot \mathbf{s}^{(t)}$
   return $\mathbf{x}^{(t)}$
```

The analysis of the Frank-Wolfe algorithm is made specifically considering convex functions f. Nonetheless, let us remark that everything holds the same for submodular functions. Fujishige and Isotani [12] show that we can solve the submodular function minimization problem by finding the minimum-norm point in the base polytope $\mathcal{B}(f)$.

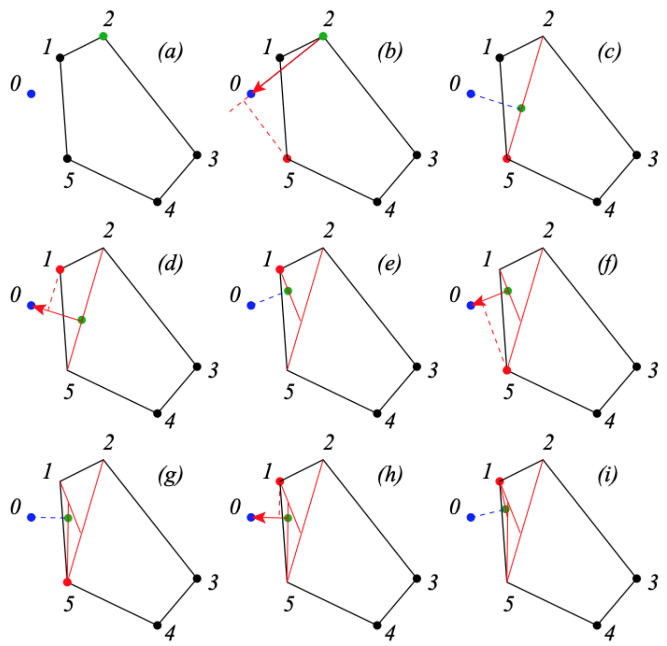


Figure 5.1: Figure and text borrowed from the original paper by Bach [3]. Illustration of Frank-Wolfe conditional gradient algorithm: starting from the initialization (a), in steps (b), (d), (f), (h), an extreme point on the polytope is found an in steps (c), (e), (g), (i), a line search is performed. Note the oscillations to converge to the optimal point.

5.2.4 L-Field

Djolonga and Krause, 2014 [7] propose L-Field, a method for approximating marginal probabilities in Bayesian submodular and supermodular models through variational inference. The main ideas are that:

1. We can exploit the property that submodular and supermodular functions can be bounded by modular functions.
2. We can use any modular bound to define a completely factorized distribution $Q(A) \propto \exp(\mathbf{s}(A))$ that can be used as a proxy to approximate values of interest in the original distribution $P(A) \propto \exp(F(A))$.
3. The intractability of our task is due to the complexity of calculating the partition function $\mathcal{Z}$, a problem which is known to be #P-hard [24]. Given the supermodular function F, if $\mathbf{s}(A) \geq F(A), \forall A \subseteq V$, we have that:
$$\log \mathcal{Z} = \log \sum_{A \subseteq V} \exp(F(A)) \leq \log \sum_{A \subseteq V} \exp(\mathbf{s}(A))$$
Let the function $[\![i \in A]\!]$ be defined as 1 if $i \in A$, and 0 otherwise. Also,

let us refer to $\mathbf{s}(\{i\})$ in this context as $s_i \triangleq \mathbf{s}(\{i\})$. Then,

$$\begin{aligned}\log \sum_{A\subseteq V} \exp(\mathbf{s}(A)) &= \log \sum_{A\subseteq V} \exp \sum_{i\in A} s(\{i\}) \\ &= \log \sum_{A\subseteq V} \prod_{i\in A} \exp(s_i) \\ &= \log \sum_{A\subseteq V} \prod_{i\in V} \exp(s_i \cdot [\![i\in A]\!]) \\ &= \log \prod_{i\in V} \sum_{[\![i\in A]\!]\in\{0,1\}} \exp(s_i \cdot [\![i\in A]\!])\end{aligned}$$

$$\boxed{\log \sum_{A\subseteq V} \exp(\mathbf{s}(A)) = \sum_{i\in V} \log(1+\exp(s_i))} \tag{5.9}$$

Since it holds that $\log \mathcal{Z} \leq \sum_{i\in V} \log(1+\exp(s_i))$, then we minimize the approximation to be as close as possible to the true partition function.

$$\boxed{\min_{\mathbf{s}\in\mathcal{B}(F)} \sum_{i\in V} \log(1+\exp(s_i))} \tag{5.10}$$

4. **Fenchel duality and the entropy viewpoint.** Consider the distribution Q with factors $Q(\{i\}) = (1+\exp(-s_i))^{-1}$. Let us refer to $Q(\{i\})$ in this context as $q_i \triangleq Q(\{i\})$. We can go back to the standard representation by using the distribution Q.

Proof. Consider the Shannon entropy [41] $\mathbb{H}[\mathbf{q}] \triangleq -\sum_{i\in V} q_i \cdot \log q_i$, and the Lovász extension [31], defined for $F : 2^V \to \mathbb{R}$ as the support over $\mathcal{B}(F)$: $f(\mathbf{q}) = \sup_{\mathbf{q}\in[0,1]^V} \mathbf{q}^T \cdot \mathbf{s}$. The Fenchel dual of the task is expressed as:

$$\max_{\mathbf{q}\in[0,1]^V} \mathbb{H}[\mathbf{q}] - f(\mathbf{q})$$

There is a zero duality gap, and the pair $(\mathbf{s}^*, \mathbf{q}^*)$ is primal-dual optimal if and only if:

$$\mathbf{q}^* = \left(\frac{1}{1+\exp(-s_1^*)}, \cdots, \frac{1}{1+\exp(-s_n^*)}\right) \text{ and } f(\mathbf{q}^*) = \mathbf{q}^{*T} \cdot \mathbf{s}^*$$

This corresponds to the fact that we can return to the standard representation with $q_i = (1+\exp(-s_i))^{-1}$. Therefore, we can reparametrize the problem from the parameters $\mathbf{s}$ to the marginals $\mathbf{q}$. Let us remark that the stopping criteria of the Frank-Wolfe algorithm ensures the small duality gap required for primal-dual optimality. □

5. Task 5.10 involves the optimization of a convex function over the base polytope $\mathcal{B}(F)$, which has already been considered by Fujishige [11] and Bach [3]. Furthermore, Jaggi [22] shows that we can use the Frank-Wolfe algorithm to solve this kind of problems.

Remark 5.19. The supermodular polyhedron $\mathcal{P}(F)$ is unbounded by definition, which is the reason why we optimize over the base polytope $\mathcal{B}(F)$. If we can prove that $\mathbf{s} \in \mathcal{P}(F)$ results in a bounded objective function, then we can solve the task for the supermodular polyhedron.

6. Since we have proven that the attractive Ising model is a supermodular system, this means we can obtain its base polytope $\mathcal{B}(F)$ and use it as the domain for the optimization problem.

5.3 Robust Optimization

RO [4] is a framework for dealing with uncertainty in optimization problems. Uncertainty can be introduced to systems by a variety of sources, such as measurement devices and floating point operations. Furthermore, uncertainty can be intrinsically a part of the problem, such as when we must account for stochasticity. Therefore, we must take this into consideration when solving optimization tasks, since feasible solutions for some realizations of the uncertain variables might not be so for others. One of the goals of RO is to find solutions that remain feasible for all the possible realizations considered in the uncertainty set $\mathcal{U}$. In order to achieve robustness in the solution, RO considers the worst-case realizations found in $\mathcal{U}$ when solving the optimization problem.

The basic RO paradigm is based on the following three assumptions [15]:

1. All decision variables represent final decisions: they should get specific numerical values as a result of solving the problem before the realization of the uncertainty set $\mathcal{U}$ becomes revealed.
2. The decision maker is responsible for providing a solution that takes into consideration the possible realizations from the uncertainty set $\mathcal{U}$ only.
3. The constraints of the uncertain problem in question are hard, i.e., the decision maker cannot tolerate violations of constraints.

Let us formally define a couple of concepts of interest from the RO framework that are important for understanding robust learning.

Definition 5.20 (Uncertainty set)**.** Let the uncertainty set $\mathcal{U}$ of $\mathbf{x}$ contain all the values that $\mathbf{x}$ might take, given some perturbation rules.

Definition 5.21 (Perturbation set)**.** Let the perturbation set Δ of $\mathbf{x}$ be:

$$\Delta \triangleq \{\, \delta \mid \mathbf{x}^u \in \mathcal{U},\, \delta = \mathbf{x}^u - \mathbf{x} \,\}$$

Definition 5.22 (Worst-case example)**.** Let the worst-case example of an uncertainty set be the example that produces the highest value to an objective function that we aim to minimize.

5.3.1 Robust Learning

Most state-of-the-art methods that aim at robustness against adversarial attacks implement the RO framework in order to learn robust parameters θ. The main idea is to learn the parameters θ that minimize the worst-case empirical risk $\rho(\theta)$; i.e., to solve the saddle problem, given the training set $\mathcal{T}$, and some allowed perturbation around each example $\delta \in \Delta$. This is expressed as follows:

$$\min_{\theta} \rho(\theta), \text{ where } \rho(\theta) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{T}}\left[\max_{\delta\in\Delta} \mathcal{L}_{\theta}(\mathbf{x}+\delta, \mathbf{y})\right] \tag{5.11}$$

Definition 5.23 (Robust exemplar)**.** Let the uncertainty set $\mathcal{U}$ of example $\mathbf{x}$ be generated by applying a rule of similarity to example $\mathbf{x}$. In other words, let the rule of similarity be used to define the perturbation set Δ. A robust exemplar is the worst-case example (w.r.t. the loss of the classifier) found in the uncertainty set of example $\mathbf{x}$.

Definition 5.24 (Adversarial training)**.** Let adversarial training define the learning process carried out over the robust exemplars obtained from the training data at each training step.

Chapter 6

Proposed Methods

6.1 Stochastic neural network with lateral interactions

Let us have a CNN which encodes a CRF Belief Network, as described in remark 5.11. At each layer we will have a CRF over the system of hyperedges $\mathcal{C}$ given by applying a convolution operation in the 2-dimensional lattice that defines each channel. Then, the CRF will correlate the activations of the artificial neurons. This correlation, which we will refer to as lateral interactions, is justified by the assumption of spatial continuity of neuron activations.

In order to carry out the propagation of activations layer by layer, we need to determine, at each layer, the realization of potentials of the Ising model. We consider channel-wise interactions only, meaning that the system of hyperedges is established between members of the same channel. The activations from the previous layer are seen as the external field, which is "polarizing" each node of the channel. Since we have encoded a structural prior at each channel, the relationships established will have to agree or disagree with the polarization received from the external field. These dynamics determine the realization of potentials at each channel, which in turn translates into the probability of each artificial neuron $i \in V^k$ of channel $k > 0$ of becoming active: $P(X_i^k = 1 \mid \mathbf{x}^0)$.

We will determine the realization of potentials of the Ising model CRF using the L-Field approach [51].

Let us recall the general form of the energy function $F(\mathbf{x})$ characterizing high-order CRFs, and the Gibbs distribution $P(\mathbf{x})$ of the CRF:

$$\begin{aligned} F(\mathbf{x}) &= -\alpha \cdot \sum_{c \in \mathcal{C}} f(x_c) \ + \ \langle \beta, \mathbf{x} \rangle \\ f(x_c) &= \max_{i \in c} x_i - \min_{i \in c} x_i \end{aligned}$$

In the context of our CRF Belief Network, CRFCNN, we have at each channel of layer $k > 0$:

$$P(X^k \mid \overline{X^{k-1}}) = \frac{1}{\mathcal{Z}(\overline{X^{k-1}}, \mathbf{w})} \cdot \exp\left(\sum_{c \in \mathcal{C}} f(X^k_c) + \langle \mathbf{w}^T \overline{X^{k-1}}, X^k \rangle \right)$$

We are interested in computing the marginal probabilities corresponding to feed-forward approximate inference, $P(X^k_i = 1 \mid \overline{X^{k-1}})$, which represents the mean activation of neuron $i \in V^k$ at layer $k > 0$. For this, we need to compute the partition function $\mathcal{Z}$.

We can use the modular function $Q(X^k = \mathbf{x}) \propto \exp(\mathbf{s}(\mathbf{x}))$, which bounds the Gibbs distribution $P(X^k = \mathbf{x} \mid \overline{X^{k-1}})$, to approximate the partition function. Recall that for $k > 0$ we have $x^k_i \in \mathcal{L}$, where $\mathcal{L} \equiv \{0, 1\}$. For $\mathbf{s}(\mathbf{x}) = \langle \mathbf{s}, \mathbf{x} \rangle$, this yields $Q(X^k_i = 1) = (1 + \exp(-s_i))^{-1}$, and the following optimization task, which is solvable efficiently by the Frank-Wolfe algorithm, as seen in section 5.2.4.

$$\mathbf{s}^* = \operatorname*{arg\,min}_{\mathbf{s} \in \mathcal{B}(F)} \sum_{i \in V} \log\left(1 + \exp(s_i)\right)$$

Let us have $x^k_i = 1, \forall i \in V^k$ be equivalent to selecting the whole ground set V^k, as required for the base polytope. Then, we have that:

$$\begin{aligned} \mathcal{P}(F) &\triangleq \{\mathbf{s} \mid \langle \mathbf{s},\, \mathbf{x} \rangle \geq F(\mathbf{x}),\, \forall \mathbf{x} \in \mathcal{S}\} \\ \mathcal{B}(F) &\triangleq \{\mathbf{s} \mid \langle \mathbf{s},\, \mathbf{x} \rangle \geq F(\mathbf{x}),\, \forall \mathbf{x} \in \mathcal{S},\, \langle \mathbf{s},\, \mathbf{1} \rangle = F(\mathbf{1})\} \end{aligned}$$

Proposition 6.1. *In this task the objective is bounded for the supermodular polyhedron $\mathcal{P}(F)$. Given this, we know from remark 5.19 that the condition $\langle \mathbf{s},\, \mathbf{1} \rangle = F(\mathbf{1})$ can be omitted, and we can solve the task over $\mathcal{P}(F)$.*

$$\mathbf{s}^* = \operatorname*{arg\,min}_{\mathbf{s} \in \mathcal{P}(F)} \sum_{i \in V} \log\left(1 + \exp(s_i)\right) \tag{6.1}$$

Proof. The Frank-Wolfe relies on the minimization of the first-order Taylor series approximation at each iteration, to approach the optimum. Let $\mathbf{r} \triangleq \nabla \sum_{i\in V^k} \log\left(1+\exp\left(s_i\right)\right)$ be the gradient of the objective. We have that it results in components $r_i = (1+\exp(-s_i))^{-1}$, which can be seen as sigmoid functions with range $[0, 1]$. Let $u_i \in \mathcal{P}(F)$. Given that each component of the inner minimization of the Frank-Wolfe can be expressed as $r_i \cdot u_i$, and given that r_i is non-negative $\forall i \in V^k$, this means that the only way to have an unbounded minimization task is if some $u_i = -\infty$. Now, let $\mathbf{a} \triangleq F(\mathbf{x})$. By the definition of the supermodular polyhedron, we must have that $\langle \mathbf{u}, \mathbf{x}\rangle \geq \langle \mathbf{a}, \mathbf{x}\rangle$, $\forall \mathbf{x} \in \mathcal{S}$. Therefore, the values $\mathbf{u}$ can take will be lower bounded by F. On the other hand, if any r_i would be negative, then some $u_i = \infty$ would belong to $\mathcal{P}(F)$ and result in an unbounded minimization task. □

Frank-Wolfe (Application)

As initial value we can use $\mathbf{s}^{(0)} \triangleq \beta$, which lies at a feasible extreme of the supermodular polyhedron $\mathcal{P}(F)$. We must also establish the descent direction on the objective, at each time step t of the algorithm. Let the gradient be denoted as $\mathbf{r} \triangleq \nabla \sum_{i\in V^k} \log\left(1+\exp\left(s_i\right)\right)$. Then:

$$\nabla \log\left(1+\exp\left(s_i^{(t)}\right)\right) = \frac{\exp\left(s_i^{(t)}\right)}{1+\exp\left(s_i^{(t)}\right)} = \frac{1}{1+\exp\left(-s_i^{(t)}\right)}$$

$$\mathbf{r}^{(t)} = \left(\frac{1}{1+\exp\left(-s_1^{(t)}\right)}, \frac{1}{1+\exp\left(-s_2^{(t)}\right)}, \cdots, \frac{1}{1+\exp\left(-s_n^{(t)}\right)}\right) \tag{6.2}$$

Now we must solve the first-order Taylor-series approximation of the optimization task: $\mathbf{u}^{(t)} = \arg\min_{\mathbf{u}\in\mathcal{P}(F)} \langle \mathbf{r}^{(t)}, \mathbf{u}\rangle$. For the standard Ising model, this task has a closed form solution: $\mathbf{u}^{(t)} = \beta - \alpha \cdot \sum_{ij\in E} \operatorname{sign}(r_i^{(t)} - r_j^{(t)})$. When considering a system of hyperedges on a 2-dimensional lattice, which is our case, the closed form solution is:

$$\boxed{\mathbf{u}^{(t)} = \beta - \alpha \cdot \sum_{c\in\mathcal{C}} \left[\mathbf{1}_{\underset{i\in c}{\arg\max}\; r_i} - \mathbf{1}_{\underset{i\in c}{\arg\min}\; r_i}\right]} \tag{6.3}$$

Finally, we compute the step size for the iteration step t: $\gamma^{(t)} \triangleq \frac{2}{t+2}$, and we obtain a new solution point: $\mathbf{s}^{(t+1)} \triangleq (1-\gamma^{(t)}) \cdot \mathbf{s}^{(t)} + \gamma^{(t)} \cdot \mathbf{u}^{(t)}$. Then, we repeat the process as described in the algorithm until the duality gap is small enough: $\langle \mathbf{s}^{(t)} - \mathbf{u}^{(t)}, \mathbf{r}^{(t)}\rangle \leq \epsilon$, or until some pre-established limit of iterations is reached.

When we terminate the Frank-Wolfe algorithm, we return to the standard representation using the marginals $Q(X_i^k = 1) = (1 + \exp(-s_i))^{-1}$.

Differentiability of the proposed method

In order to learn deep neural networks efficiently using the backpropagation algorithm, we must address the issue of differentiability. The gradients required for learning are the following:

$$\frac{\partial Q(X_i^k = 1)}{\partial \theta} = \frac{\partial Q(X_i^k = 1)}{\partial \mathbf{s}^*} \cdot \frac{\partial \mathbf{s}^*}{\partial \beta} \cdot \frac{\partial \beta}{\partial \mathbf{w}}$$

$$\frac{\partial Q(X_i^k = 1)}{\partial s_i^*} = \frac{\partial}{\partial s_i^*}\left[\frac{\exp(s_i^*)}{1 + \exp(s_i^*)}\right] = \frac{\exp(s_i^*)}{(1 + \exp(s_i^*))^2}$$

$$\frac{\partial \mathbf{s}^*}{\partial \beta} = ?$$

Since we don't have an expression to differentiate by β, we will reparametrize the task:

$$\begin{aligned} \mathbf{s}(\mathbf{x}) = \langle \mathbf{s}, \mathbf{x}\rangle,\ \mathbf{s} \in \mathcal{P}(F) &\iff \langle \mathbf{s}, \mathbf{x}\rangle \geq F_\alpha(\mathbf{x}) + \langle \beta, \mathbf{x}\rangle,\ \forall \mathbf{x} \in \mathcal{S} \\ \mathbf{s}'(\mathbf{x}) = \langle \mathbf{s}', \mathbf{x}\rangle,\ \mathbf{s}' \in \mathcal{P}(F_\alpha) &\iff \langle \mathbf{s}', \mathbf{x}\rangle \geq F_\alpha(\mathbf{x}),\ \forall \mathbf{x} \in \mathcal{S} \\ \mathbf{s} &\longmapsto \mathbf{s}' + \beta \end{aligned}$$

$$\boxed{\mathbf{s}' = \arg\min_{\mathbf{s}' \in \mathcal{P}(F_\alpha)} \sum_{i \in V^k} \log(1 + \exp(s_i' + \beta_i))} \tag{6.4}$$

Performing also the change of variables in the marginals, we get:

$$\boxed{Q(X_i^k = 1) = (1 + \exp(-(s_i' + \beta_i)))^{-1}} \tag{6.5}$$

Backpropagation can now be computed as:

$$\frac{\partial Q(X_i^k = 1)}{\partial \theta} = \frac{\partial Q(X_i^k = 1)}{\partial \beta} \cdot \frac{\partial \beta}{\partial \mathbf{w}}$$

$$\frac{\partial Q(X_i^k = 1)}{\partial \beta_i} = \frac{\partial}{\partial \beta_i}\left[\frac{\exp(s_i' + \beta_i)}{1 + \exp(s_i' + \beta_i)}\right] = \frac{\exp(s_i' + \beta_i)}{(1 + \exp(s_i' + \beta_i))^2}$$

$$\frac{\partial \beta}{\partial \mathbf{w}} = \frac{\partial[\mathbf{w}^T \cdot \mathbf{x}^{k-1}]}{\partial \mathbf{w}} \longrightarrow \frac{\partial \beta_i}{\partial w_i} = x_i^{k-1}$$

We finish this section by reparametrizing accordingly the rest of the expressions used in the Frank-Wolfe algorithm.

$$\mathbf{s}^{(0)} \triangleq \beta \longmapsto \mathbf{s}'^{(0)} \triangleq \mathbf{0}$$

$$r_i^{(t)} \triangleq \frac{1}{1 + \exp(-s_i^{(t)})} \longmapsto r_i'^{(t)} \triangleq \frac{1}{1 + \exp(-(s_i'^{(t)} + \beta_i))}$$

The solution to the first-order Taylor-series approximation in the standard Ising model becomes $\mathbf{u}'^{(t)} = -\alpha \cdot \sum_{ij \in E} \operatorname{sign}({r'_i}^{(t)} - {r'_j}^{(t)})$. Likewise, in our more general case we get:

$$\boxed{\mathbf{u}'^{(t)} = -\alpha \cdot \sum_{c \in \mathcal{C}} \left[\mathbf{1}_{\underset{i \in c}{\arg\max}\ r'_i} - \mathbf{1}_{\underset{i \in c}{\arg\min}\ r'_i} \right]} \tag{6.6}$$

Robust learning with CRFCNN

The proposed network will be used for both standard learning, and robust learning. The goal is to test the advantages of its structural prior, first as a natural defense, and second, as a capacity enhancement that amplifies the learning of robust features.

Chapter 7

Experiments

In this section we will describe the trial scenarios with which we will evaluate the effect of the proposed methods in comparison with standard methods. We will also describe the specific settings used for the experiments. The reader is encouraged to reproduce the results obtained.

7.1 BaselineCNN

Layer 1

Convolution 32 feature maps generated by a convolution window of width 5 and height 5, with stride 1, and padding 0. It yields 32 feature maps of width 24 and height 24.

Batch Normalization For 32 feature maps.

Sigmoid Activation Function $\overline{X^k} = (1 + \exp(\mathbf{w}^T \cdot \overline{X^{k-1}})^{-1}$

Max Pooling Operation performed using a convolution window of width 3 and height 3, with stride 3, and padding 0. It yields 32 feature maps of width 8 and height 8.

Layer 2

Convolution 10 feature maps generated by a convolution window of width 5 and height 5, with stride 1, and padding 0. It yields 10 feature maps of width 4 and height 4.

Batch Normalization For 10 feature maps.

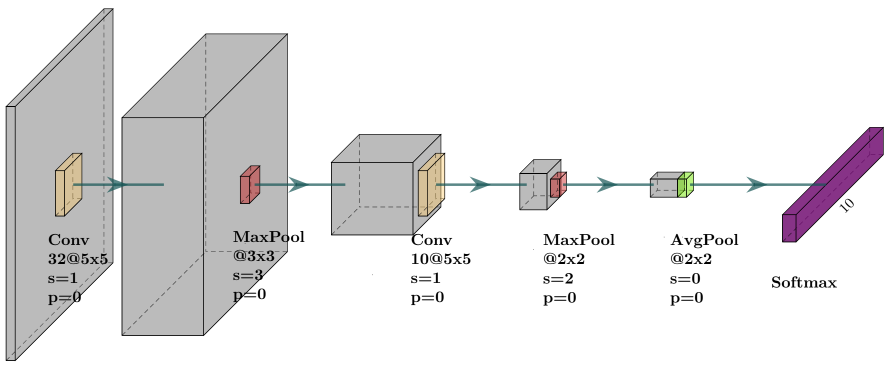


Figure 7.1: Structure of the BaselineCNN, which corresponds to a Sigmoid Belief Network. The gray blocks represent, first, the input, and then the feature maps. The creamy yellow windows represent convolution operations. The red windows represent max pooling operations. The green window represents the average pooling operation. The output is a vector of probabilities given by softmax.

Sigmoid Activation Function $\overline{X^k} = (1 + \exp(\mathbf{w}^T \cdot \overline{X^{k-1}}))^{-1}$

Max Pooling Operation performed using a convolution window of width 2 and height 2, with stride 2, and padding 0. It yields 10 feature maps of width 2 and height 2.

Average Pooling Operation performed using a convolution window of width 2 and height 2, with no stride and no padding, since it summarizes each channel in one value. It yields 10 feature maps of width 1 and height 1.

Softmax In reality, we output Log-Softmax, since the Negative Log-Likelihood (NLL) loss expects a vector of log-probabilities.

7.2 CRFCNN

Layer 1

Convolution 32 feature maps generated by a convolution window of width 5 and height 5, with stride 1, and padding 0. It yields 32 feature maps of width 24 and height 24.

Batch Normalization For 32 feature maps.

CRF Activation Function Operation performed using a convolution window of width 3 and height 3, with stride 1, and padding 0. It yields 32 feature maps of width 24 and height 24, where each feature

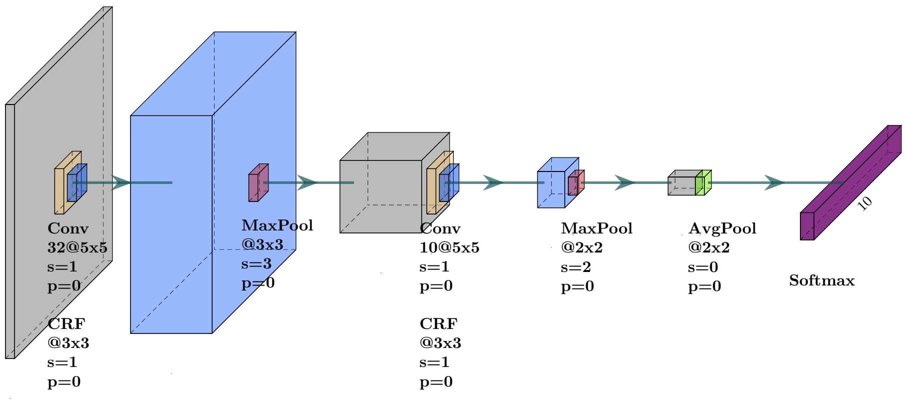


Figure 7.2: Structure of the CRFCNN, which corresponds to a CRF Belief Network. The gray blocks represent first the input, and then the feature maps. The creamy yellow windows represent convolution operations. The blue windows represent the CRF activations. The blue feature maps indicate that, after generating the feature maps, the CRF activation is executed on them. The green window represents the average pooling operation. The output is a vector of probabilities given by softmax.

map contains correlated activations.

$$P\left(X^k \mid \overline{X^{k-1}}\right) \propto \exp\left(F_\alpha(X^k) + F_\beta(X^k)\right)$$

$$\alpha = 1,\ \beta = \mathbf{w}^T \cdot \overline{X^{k-1}},\ F_\beta(X^k) = \langle \beta, X^k \rangle$$

$$F_\alpha(X^k) = -\alpha \cdot \sum_{c \in \mathcal{C}} \left[\max_{i \in c} x_i - \min_{i \in c} x_i \right]$$

$$P\left(X_i^k = 1 \mid \overline{X^{k-1}}\right) = \sum_{(x_j^k \in \mathcal{L} \mid j \in V, j \neq i)} P\left(X^k \mid \overline{X^{k-1}}\right)$$

$$\overline{X_i^k} = Q(X_i^k = 1) \approx P\left(X_i^k = 1 \mid \overline{X^{k-1}}\right)$$

Max Pooling Operation performed using a convolution window of width 3 and height 3, with stride 3, and padding 0. It yields 32 feature maps of width 8 and height 8.

Layer 2

Convolution 10 feature maps generated by a convolution window of width 5 and height 5, with stride 1, and padding 0. It yields 10 feature maps of width 4 and height 4.

Batch Normalization For 10 feature maps.

CRF Activation Function Operation performed using a convolution window of width 3 and height 3, with stride 1, and padding 0. It yields 10 feature maps of width 4 and height 4, where each feature map contains correlated activations.

$$P\left(X^k \mid \overline{X^{k-1}}\right) \propto \exp\left(F_\alpha(X^k) + F_\beta(X^k)\right)$$

$$\alpha = 1,\ \beta = \mathbf{w}^T \cdot \overline{X^{k-1}},\ F_\beta(X^k) = \langle \beta, X^k \rangle$$

$$\begin{aligned} F_\alpha(X^k) &= -\alpha \cdot \sum_{c \in \mathcal{C}} \left[\max_{i \in c} x_i - \min_{i \in c} x_i \right] \\ P(X_i^k = 1 \mid \overline{X^{k-1}}) &= \sum_{(x_j^k \in \mathcal{L} \mid j \in V, j \neq i)} P(X^k \mid \overline{X^{k-1}}) \\ \overline{X_i^k} &= Q(X_i^k = 1) \approx P(X_i^k = 1 \mid \overline{X^{k-1}}) \end{aligned}$$

Max Pooling Operation performed using a convolution window of width 2 and height 2, with stride 2, and padding 0. It yields 10 feature maps of width 2 and height 2.

Adaptive Average Pooling Operation performed using a convolution window of width 2 and height 2, with no stride and no padding, since it summarizes each channel in one value. It yields 10 feature maps of width 1 and height 1.

Softmax In reality, we output Log-Softmax, since the NLL loss expects a vector of log-probabilities.

7.3 Parameters

Training Epochs 150

Batch Size 120

Optimizer Stochastic Gradient Descent (SGD)

- **Loss Function** Negative Log-Likelihood
- **Learning Rate** 0.01
- **Momentum** 0.9
- **Weight Decay** 0

7.4 Datasets

MNIST Grayscale images (1 channel)

Classes 10

Statistics For a nominal pixel range $\mathcal{R} = [0, 1]$, MNIST is a normal distribution $\mathcal{N}(0.1307, 0.3081)$.

Normalization Normalized to the pixel domain $\mathcal{N}(0, 1)$, the pixel range is $\mathcal{R} = [-0.4242, 2.8215]$.

FashionMNIST Grayscale images (1 channel)

Classes 10

Statistics For a nominal pixel range $\mathcal{R} = [0, 1]$, FashionMNIST is a normal distribution $\mathcal{N}(0.2860, 0.3202)$.

Normalization Normalized to the pixel domain $\mathcal{N}(0, 1)$, the pixel range is $\mathcal{R} = [-0.8932, 2.2299]$.

7.5 Training Settings

Standard The standard training setting refers to training on the original data.

Adversarial The adversarial training setting refers to training on the augmented data which is the "most confusing", meaning that at each mini-batch we perform a PGD attack (for 10 iterations) with random start, clipped to the range of the data after normalization, with step size of 0.01, and with a maximum ℓ_∞-norm perturbation measure $\epsilon = 0.3$, which is rescaled to the range of the data $\epsilon' = \epsilon/\sigma_{\mathcal{T}}$.

7.6 Trial Scenarios

Robustness Against Gaussian Noise. In this scenario we will compare the accuracy of the classifiers in the presence of Gaussian noise $\delta \sim \mathcal{N}(0, \sigma)$ added to each image, for varying magnitude of σ. The results will be reported for $\sigma \in [0, 1]$. It should be noted that in the implementation the standard deviation of the noise is scaled to the range of the data $(\sigma/\sigma_{\mathcal{T}})$.

Robustness Against Adversarial Attacks. In this scenario we will compare the accuracy of the classifiers in the presence of adversarial perturbations $\delta \in \Delta$ added to each image, for varying magnitude of the bound ϵ of ℓ_∞-norm of δ. The results will be reported for $\epsilon \in [0, 1]$. It should be noted that in the implementation the bound is scaled to the range of the data ($\epsilon/\sigma_{\mathcal{T}}$).

p-Robustness Against Adversarial Attacks. This setting differs from the previous one in that here we are considering a rejecting network: we decide to reject an input image if the output of the most likely class is less than p. Then, a successful attack requires the incorrect class to present a high confidence w.r.t. the chosen p.

It is important to note that the data that will be used in the realization of these experiments will be taken from the validation set. Therefore, let us remark that any effect observed will be found in the context of the robust-generalization abilities of the classifiers.

Chapter 8

Results

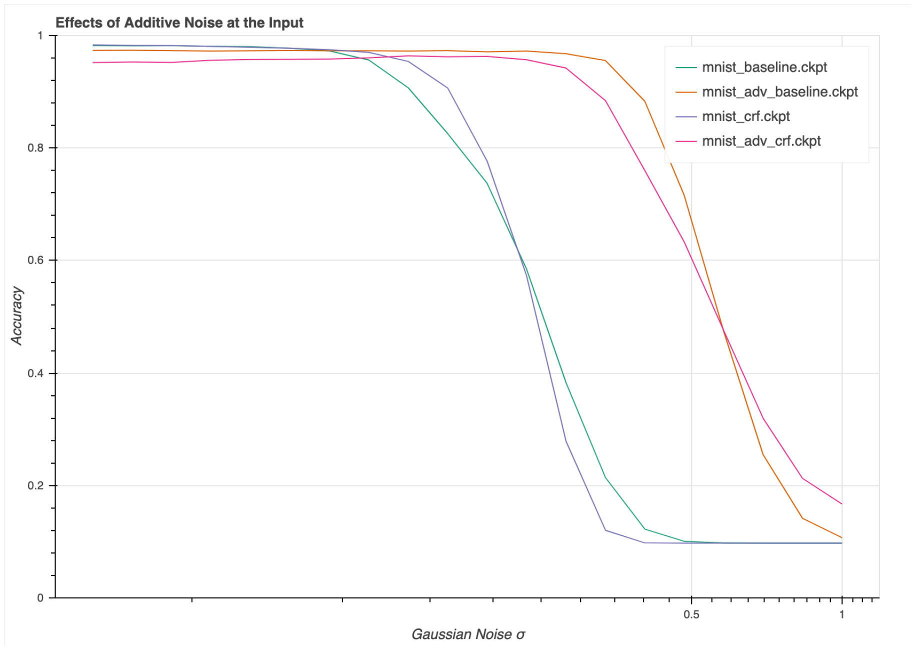
In this section we will examine the behavior of the network architectures previously described. We will consider the training variants which are standard and adversarial. The experiments performed aim to shed light on the robustness and generalization characteristics of each of these networks; therefore, we should bear in mind that the main observation of interest is the comparison of accuracy responses, in the presence of perturbations at the input. We consider two kinds of perturbations: 1) Gaussian noise; and, 2) adversarial perturbations, which come from adversarial attacks.

Due to the nature of adversarial attacks, that they aim to bring down the confidence on the correct class, and bring up the confidence on the incorrect class, we consider also a rejecting network in our tests. We believe that testing the robustness of the networks in terms of how much the confidence on the incorrect class can be incremented, is a valuable instrument that could be taken into account when using deep learning in high-stakes applications. We evaluate the results obtained for this variant by examining the ratio of acceptance of given input, and the accuracy on accepted examples (termed "safe accuracy").

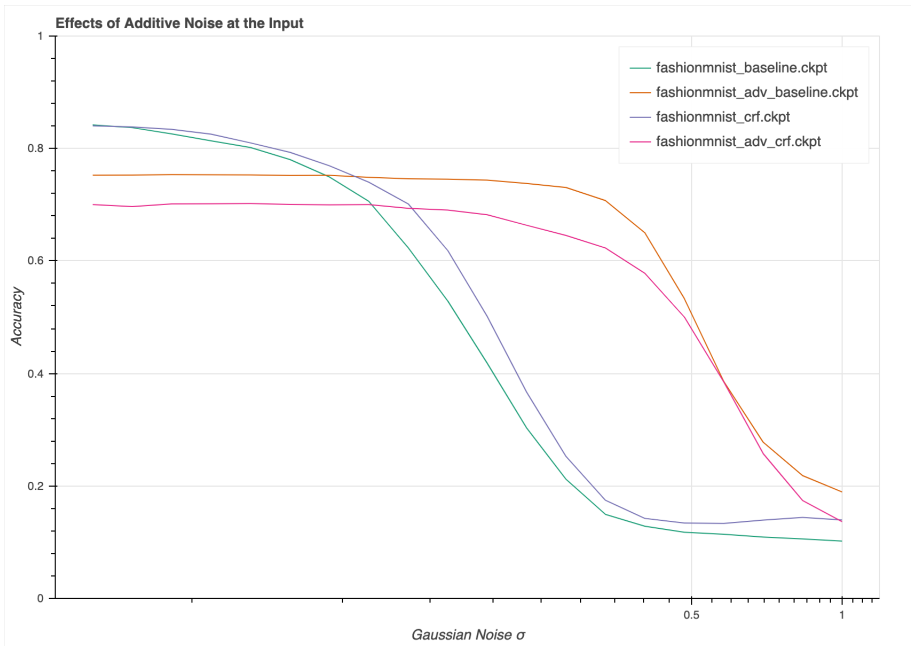
The results presented are organized first by type of experiment, and then by dataset (either MNIST or FashionMNIST). The response of all networks is then shown in one plot per dataset, for each experiment of interest.

We will limit the extent of comments performed in this chapter, and save them for the concluding remarks.

8.1 Robustness Against Gaussian Noise



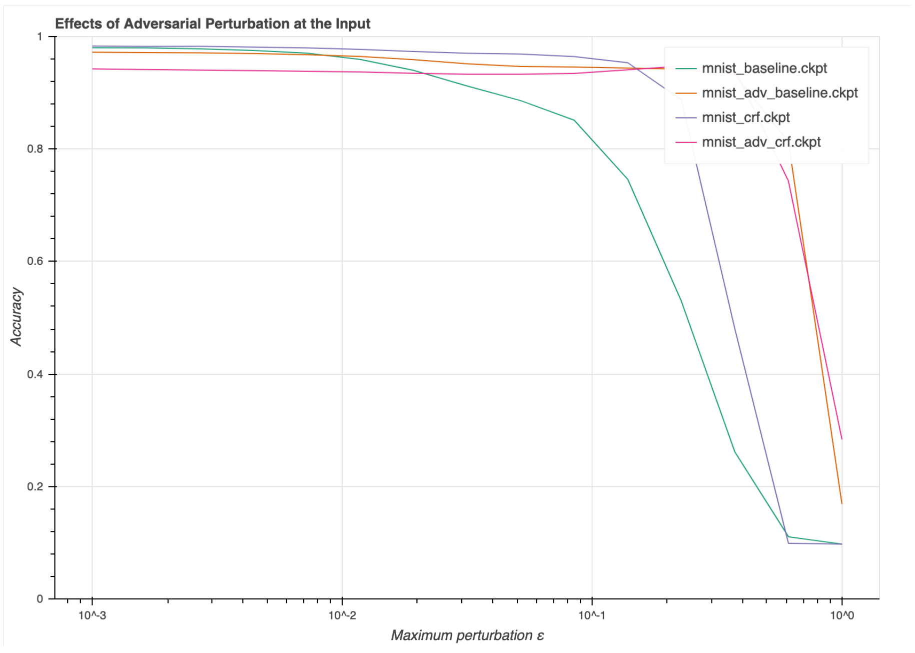
(a) : MNIST



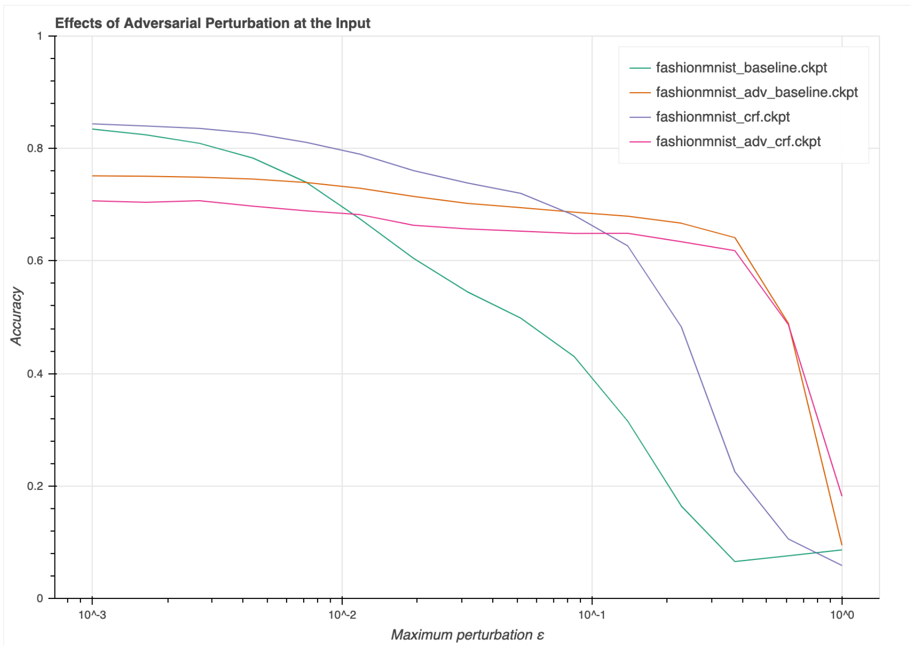
(b) : FashionMNIST

Figure 8.1: Accuracy response of each network to variable magnitudes of standard deviation σ of Gaussian noise.

8.2 Robustness Against Adversarial Attacks



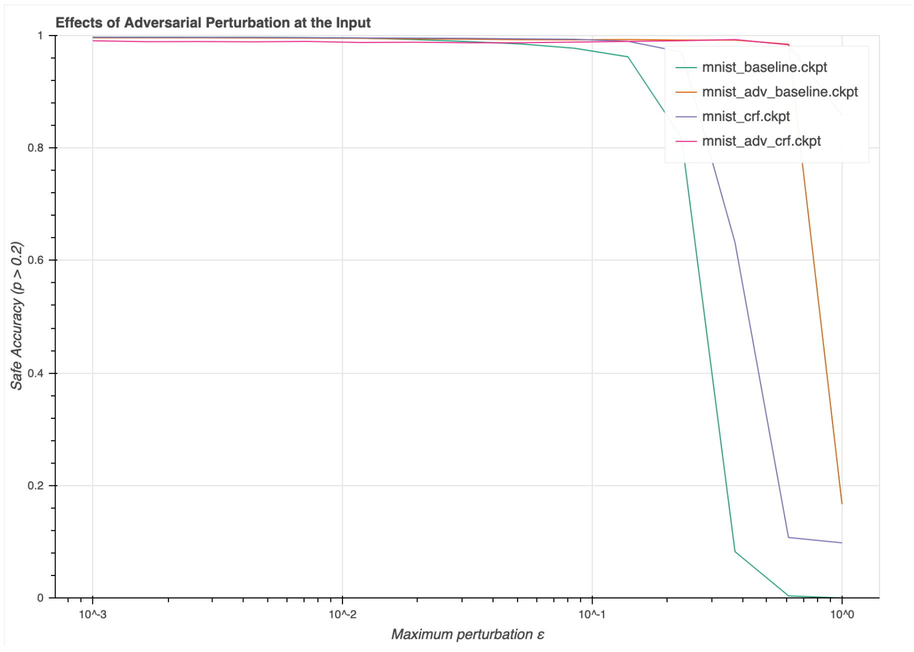
(a) : MNIST



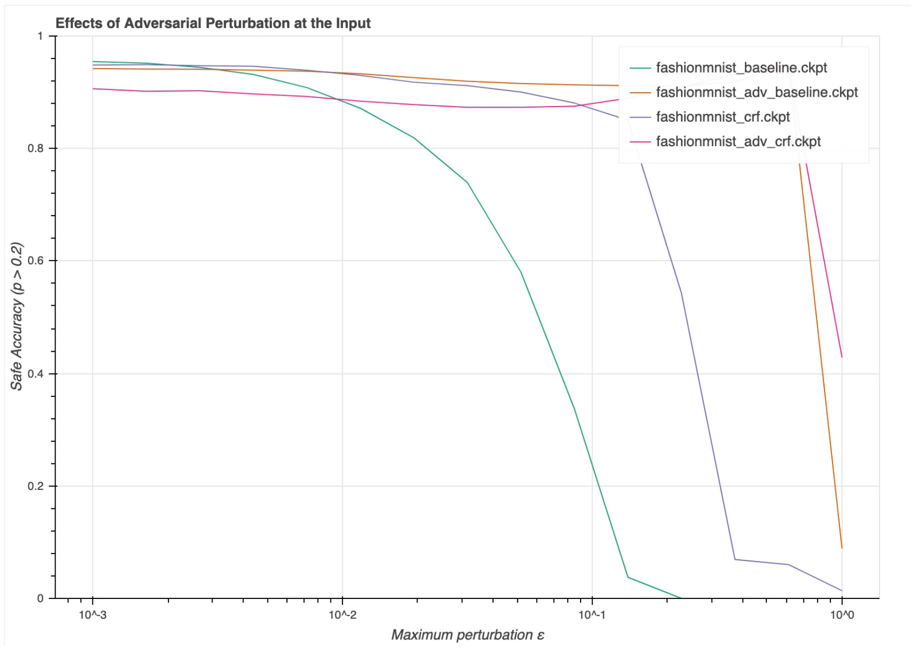
(b) : FashionMNIST

Figure 8.2: Accuracy response of each network to variable magnitudes of maximum allowed ℓ_∞-norm perturbation ϵ.

8.3 p-Robustness Against Adversarial Attacks

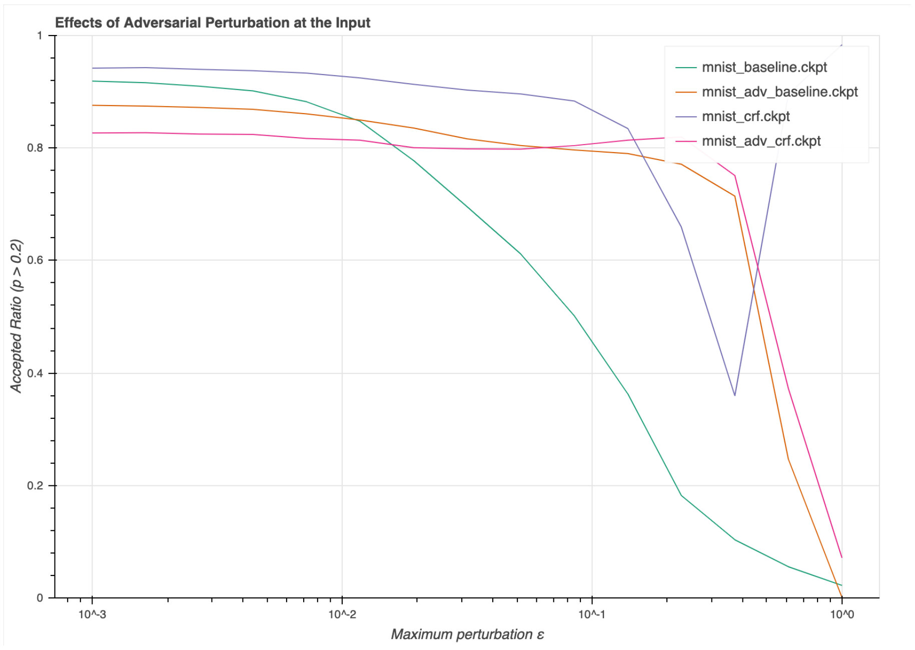


(a) : MNIST

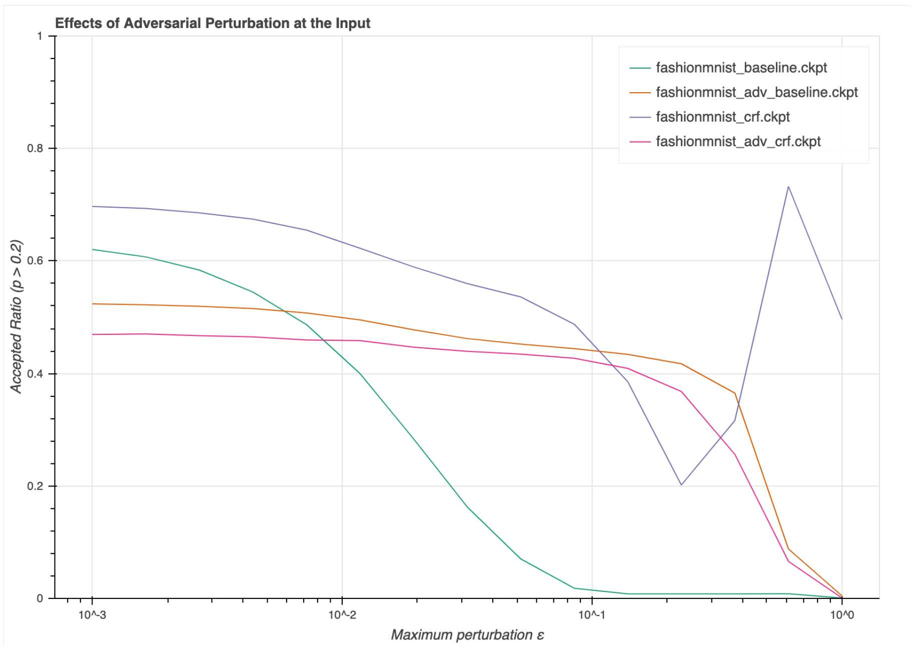


(b) : FashionMNIST

Figure 8.3: Safe accuracy response of each network to variable magnitudes of maximum allowed ℓ_∞-norm perturbation ϵ.



(a) : MNIST



(b) : FashionMNIST

Figure 8.4: Accepting rate of each network to variable magnitudes of maximum allowed ℓ_∞-norm perturbation ϵ.

Chapter 9

Conclusion

The experiments have indeed shown that the CRFCNN architecture provides an interesting option for further study in the context of robust deep learning.

For the sake of a better reading, let us refer to four variants StdBaselineCNN, AdvBaselineCNN, StdCRFCNN, AdvCRFCNN; where the prefix "Std" refers to standard training, and the prefix "Adv" refers to adversarial training.

We can see in experiments 8.1 and 8.2, for both MNIST and FashionMNIST, that the StdCRFCNN, presents significantly higher levels of robustness than the StdBaselineCNN. The comparison doesn't hold anymore when talking about the AdvCRFCNN and the AdvBaselineCNN, where the performance is relatively equal, although slightly better for the AdvBaselineCNN.

In experiment 8.1, we can see in MNIST that the StdCRFCNN presents a more steep decrease in accuracy than the StdBaselineCNN, with this decrease starting to manifest in the StdCRFCNN at a higher perturbation than in the StdBaselineCNN, in terms of the standard deviation of the noise. In FashionMNIST, the responses of the StdCRFCNN and the StdBaselineCNN were very similar to each other. The same similarity in responses can also be seen while comparing the AdvCRFCNN and the AdvBaselineCNN.

For the MNIST dataset, the AdvBaselineCNN provides a very strong contender that virtually solves the problem of learning robustly. The AdvCR-FCNN presents very similar response, although with slightly less accuracy. We conjecture three possible explanations for this behavior: 1) it could be due to the robust solution achieved being close to optimal; 2) considering the possibility that the CRFCNN is a network with more capacity than the BaselineCNN, it might be the case that it needs to train for more epochs in order to converge, or; 3) learning robustly with the CRFCNN requires more training data than learning robustly with the BaselineCNN. The second conjecture could be tested by training for more epochs, while the third conjecture could be proven or refuted by testing the robust learning response of the networks to varying training data size. Such an experiment would tell us if the CRFCNN could perform better than the BaselineCNN in the adversarial training scenario, given enough data.

The most interesting results were those obtained for the FashionMNIST dataset. In experiment 8.2 we can see a dramatic difference of accuracy between the StdBaselineCNN and the StdCRFCNN. What is even more, the StdCRFCNN presents a higher accuracy than both the AdvCRFCNN and the AdvBaselineCNN, until some critical ϵ for each. The accuracy presented by the AdvBaselineCNN is greater than that of the AdvCRFCNN for lower levels of allowed adversarial perturbation, while the AdvCRFCNN has a better response for the higher levels.

The experiment 8.3 shows that the StdCRFCNN dominates the StdBaselineCNN, and the AdvBaseline dominates the AdvCRFCNN w.r.t. safe accuracy. An unwanted behavior is presented by the StdCRFCNN wherein the confidence on predictions starts to increase for higher levels of ϵ.

If we take the standardly trained versions, it is clearly seen that the CRFCNN architecture presents higher levels of robustness than the BaselineCNN, which fundamentally is a comparison between a network with lateral interactions, and a network without them. More concretely, the results obtained for FashionMNIST give us the motivation required to keep investigating the robustness characteristics of the CRFCNN. For FashionMNIST, even though the StdBaselineCNN achieves basically the same accuracy when there is no adversarial perturbation, as soon as the adversary is allowed to attack the network, the StdBaselineCNN drops around 20% below the levels of accuracy of the StdCRFCNN.

The main drawback of the CRFCNN is that it takes significantly more time than the BaselineCNN to train it. Calculating the mean activations using the L-Field method is slow as currently implemented. Every time that we add a CRF activation to a layer, we need to calculate the marginals using the L-Field method, for each image being processed. Even though this is done by batches, L-Field provides with a significant overhead, making deeper networks an impossibility. We think, however, that the results obtained provide enough evidence to motivate further investigation, even under this big disadvantage.

9.1 Future work

Further lines of investigation could include evaluating the response of the network to other attacks such as Deepfool [35] and Carlini and Wagner [6], which have been proven to be powerful both in terms of similarity of adversarial images produced w.r.t. the original examples, and in terms of circumventing defenses that have worked against basic attacks such as Fast Gradient Sign Method (FGSM).

As we saw in the experiments performed, the comparisons for the MNIST dataset didn't show such a strong evidence as to the robustness characteristics of the CRFCNN, due to the relative simplicity the task. Therefore, the responses should be examined for more complex tasks, such as CIFAR-10, SVHN, STL-10 and CIFAR-100, in order to assess the significance of the results herein obtained.

Other means of confering robustness should also be studied, in conjunction with the CRFCNN architecture. Given its response in the standard case, it could be that combining it with other robustness conferring mechanisms would boost its robustness to levels closer to those achieved by adversarial training.

Since the benefits of the CRFCNN can be seen, the only argument against its usability, even if further positive results are obtained, are in terms of its efficiency (that is, the time required for training it). Therefore, improving the efficiency of its implementation is necessary for the CRFCNN to be considered as an option in practical applications, specially in those that require deeper architectures.

In the context of adversarial training, we believe that a network such as the CRFCNN would benefit from more meaningful adversarial examples. That is, its response could be improved when trained with adversarial examples that deviate less, in terms of perceptual similarity, from the original images. Given an ℓ_p-norm ϵ-bound allowed for adversarial training, attacks such as FGSM or PGD tend to produce images with perturbation patches that are noticeable by humans. Improving the uncertainty sets (in our context, the adversarial examples) either by means of the metric used to measure perceptual similarity, or by means of the adversarial attack used, is a promising line of work that would allow us to understand better the robustness capabilities of the CRFCNN, in comparison with that of the BaselineCNN under the same conditions.

Bibliography

[1] Arnab, A., Jayasumana, S., Zheng, S., and Torr, P. (2015). Higher Order Conditional Random Fields in Deep Neural Networks. *arXiv e-prints*, page arXiv:1511.08119.

[2] Arnab, A., Miksik, O., and Torr, P. H. S. (2017). On the Robustness of Semantic Segmentation Models to Adversarial Attacks. *arXiv e-prints*, page arXiv:1711.09856.

[3] Bach, F. (2011). Learning with Submodular Functions: A Convex Optimization Perspective. *arXiv e-prints*, page arXiv:1111.6453.

[4] Ben-Tal, A., El Ghaoui, L., and Nemirovski, A. (2009). *Robust Optimization*. Princeton Series in Applied Mathematics. Princeton University Press.

[5] Carlini, N., Mishra, P., Vaidya, T., Zhang, Y., Sherr, M., Shields, C., Wagner, D., and Zhou, W. (2016). Hidden voice commands. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, pages 513–530, Berkeley, CA, USA. USENIX Association.

[6] Carlini, N. and Wagner, D. (2016). Towards Evaluating the Robustness of Neural Networks. *arXiv e-prints*, page arXiv:1608.04644.

[7] Djolonga, J. and Krause, A. (2014). From map to marginals: Variational inference in bayesian submodular models. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 244–252. Curran Associates, Inc.

[8] Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., and Madry, A. (2017). A Rotation and a Translation Suffice: Fooling CNNs with Simple Transformations. *arXiv e-prints*, page arXiv:1712.02779.

[9] Fathony, R., Rezaei, A., Bashiri, M. A., Zhang, X., and Ziebart, B. D. (2018). Distributionally Robust Graphical Models. *arXiv e-prints*, page arXiv:1811.02728.

[10] Fawzi, A. and Frossard, P. (2015). Manitest: Are classifiers really invariant? *arXiv e-prints*, page arXiv:1507.06535.

[11] Fujishige, S. (2005). *Submodular Functions and Optimization*. Annals of Discrete Mathematics. Elsevier Science.

[12] Fujishige, S. and Isotani, S. A submodular function minimization algorithm based on the minimum-norm base. *Pacific Journal of Optimization*, page 2011.

[13] Gilmer, J., Metz, L., Faghri, F., Schoenholz, S. S., Raghu, M., Wattenberg, M., and Goodfellow, I. (2018). Adversarial Spheres. *arXiv e-prints*, page arXiv:1801.02774.

[14] Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and Harnessing Adversarial Examples. *arXiv e-prints*, page arXiv:1412.6572.

[15] Gorissen, B. L., Yanıkoğlu, I., and den Hertog, D. (2015). A Practical Guide to Robust Optimization. *arXiv e-prints*, page arXiv:1501.02634.

[16] Grosse, K., Papernot, N., Manoharan, P., Backes, M., and McDaniel, P. (2016). Adversarial Perturbations Against Deep Neural Networks for Malware Classification. *arXiv e-prints*, page arXiv:1606.04435.

[17] Gu, S. and Rigazio, L. (2014). Towards Deep Neural Network Architectures Robust to Adversarial Examples. *arXiv e-prints*, page arXiv:1412.5068.

[18] Guo, C., Rana, M., Cisse, M., and van der Maaten, L. (2017). Countering Adversarial Images using Input Transformations. *arXiv e-prints*, page arXiv:1711.00117.

[19] Hendrik Metzen, J., Chaithanya Kumar, M., Brox, T., and Fischer, V. (2017). Universal Adversarial Perturbations Against Semantic Image Segmentation. *arXiv e-prints*, page arXiv:1704.05712.

[20] Hendrycks, D. and Gimpel, K. (2016). Early Methods for Detecting Adversarial Images. *arXiv e-prints*, page arXiv:1608.00530.

[21] Huang, R., Xu, B., Schuurmans, D., and Szepesvari, C. (2015). Learning with a Strong Adversary. *arXiv e-prints*, page arXiv:1511.03034.

[22] Jaggi, M. (2013). Revisiting frank-wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages I–427–I–435. JMLR.org.

[23] Jang, U., Wu, X., and Jha, S. (2017). Objective metrics and gradient descent algorithms for adversarial examples in machine learning. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, ACSAC 2017, pages 262–277, New York, NY, USA. ACM.

[24] Jerrum, M. and Sinclair, A. (1993). Polynomial-time approximation algorithms for the ising model. *SIAM J. Comput.*, 22(5):1087–1116.

[25] Jordan, M., Manoj, N., Goel, S., and Dimakis, A. r. G. (2019). Quantifying Perceptual Distortion of Adversarial Examples. *arXiv e-prints*, page arXiv:1902.08265.

[26] Khoury, M. and Hadfield-Menell, D. (2018). On the Geometry of Adversarial Examples. *arXiv e-prints*, page arXiv:1811.00525.

[27] Kohli, P., Kumar, M. P., and Torr, P. H. S. (2007). P3 & beyond: Solving energies with higher order cliques. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

[28] Kurakin, A., Goodfellow, I., and Bengio, S. (2016a). Adversarial examples in the physical world. *arXiv e-prints*, page arXiv:1607.02533.

[29] Kurakin, A., Goodfellow, I., and Bengio, S. (2016b). Adversarial Machine Learning at Scale. *arXiv e-prints*, page arXiv:1611.01236.

[30] Liu, Y., Chen, X., Liu, C., and Song, D. (2016). Delving into Transferable Adversarial Examples and Black-box Attacks. *arXiv e-prints*, page arXiv:1611.02770.

[31] Lovász, L. (1983). *Submodular functions and convexity*, pages 235–257. Springer Berlin Heidelberg, Berlin, Heidelberg.

[32] Luo, B., Liu, Y., Wei, L., and Xu, Q. (2018). Towards Imperceptible and Robust Adversarial Example Attacks against Neural Networks. *arXiv e-prints*, page arXiv:1801.04693.

[33] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2017). Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv e-prints*, page arXiv:1706.06083.

[34] Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. (2016). Universal adversarial perturbations. *arXiv e-prints*, page arXiv:1610.08401.

[35] Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. (2015). DeepFool: a simple and accurate method to fool deep neural networks. *arXiv e-prints*, page arXiv:1511.04599.

[36] Moosavi-Dezfooli, S.-M., Shrivastava, A., and Tuzel, O. (2018). Divide, Denoise, and Defend against Adversarial Attacks. *arXiv e-prints*, page arXiv:1802.06806.

[37] Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

[38] Neal, R. M. (1992). Connectionist learning of belief networks. *Artif. Intell.*, 56(1):71–113.

[39] Nitin Bhagoji, A., Cullina, D., Sitawarin, C., and Mittal, P. (2017). Enhancing Robustness of Machine Learning Systems via Data Transformations. *arXiv e-prints*, page arXiv:1704.02654.

[40] Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K., and Mądry, A. (2018). Adversarially Robust Generalization Requires More Data. *arXiv e-prints*, page arXiv:1804.11285.

[41] Shannon, C. E. (2001). A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55.

[42] Sharif, M., Bauer, L., and Reiter, M. K. (2018). On the Suitability of L__p-norms for Creating and Preventing Adversarial Examples. *arXiv e-prints*, page arXiv:1802.09653.

[43] Shekhovtsov, A., Flach, B., and Busta, M. (2018). Feed-forward Uncertainty Propagation in Belief and Neural Networks. *arXiv e-prints*, page arXiv:1803.10590.

[44] Snell, J., Ridgeway, K., Liao, R., Roads, B. D., Mozer, M. C., and Zemel, R. S. (2015). Learning to Generate Images with Perceptual Similarity Metrics. *arXiv e-prints*, page arXiv:1511.06409.

[45] Stratos, K. The frank-wolfe algorithm basics.

[46] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv e-prints*, page arXiv:1312.6199.

[47] Tramèr, F., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. (2017). The Space of Transferable Adversarial Examples. *arXiv e-prints*, page arXiv:1704.03453.

[48] Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. (2018). Robustness May Be at Odds with Accuracy. *arXiv e-prints*, page arXiv:1805.12152.

[49] Xie, C., Wang, J., Zhang, Z., Zhou, Y., Xie, L., and Yuille, A. (2017). Adversarial Examples for Semantic Segmentation and Object Detection. *arXiv e-prints*, page arXiv:1703.08603.

[50] Xu, H. and Mannor, S. (2010). Robustness and Generalization. *arXiv e-prints*, page arXiv:1005.2243.

[51] Zhang, J., Djolonga, J., and Krause, A. (2015). Higher-order inference for multi-class log-supermodular models. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1859–1867.

Abbreviations

AGMs	Adversarial Graphical Models
CNN	Convolutional Neural Network
CNNs	Convolutional Neural Networks
CRF	Conditional Random Field
CRFs	Conditional Random Fields
DRO	Distributionally Robust Optimization
FashionMNIST	FashionMNIST
FFT	Fast Fourier Transform
FGSM	Fast Gradient Sign Method
HOG	Histogram of Oriented Gradients
KL	Kullback Leibler
LPIPS	Learned Perceptual Image Patch Similarity
MNIST	MNIST
MRF	Markov Random Field
MS-SSIM	Multiscale Structural-Similarity Index Measure
NLL	Negative Log-Likelihood
PGD	Projected Gradient Descent
RO	Robust Optimization
SGD	Stochastic Gradient Descent
SSIM	Structural Similarity Index Measure
UGM	Undirected Graphical Models

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Hilario Reyes Jose Ananias**	Personal ID number:	**464927**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Computer Science and Engineering**		
Study program:	**Open Informatics**		
Branch of study:	**Artificial Intelligence**		

II. Master’s thesis details

Master’s thesis title in English:

Prior models for robust adversarial deep learning

Master’s thesis title in Czech:

Apriorní modely pro rubustní adversariální hluboké učení

Guidelines:

Deep networks learned by standard methods of discriminative learning are susceptible to adversarial patterns. Training adversarially robust deep networks therefore requires new learning methods. One interesting option is to include appropriate prior knowledge that will restrict the search space explored by the learning method.
The thesis aims at analysing and comparing different types of prior knowledge w.r.t. their impact on adversarial robustness. A suitable option for convolutional deep networks is to introduce lateral interactions within the convolutional layers to reflect the assumption of spatial continuity.
Tasks:
1. Analyse and compare different ways for including prior knowledge into the learning/inference methods suitable for DNNs and convolutional DNNs in particular.
2. Develop an approach for including prior knowledge into deep CNNs by introducing lateral interactions in the convolutional layers. Show that the approximation proposed in [1] allows end-to-end training of the generalised CNNs.
3. Implement at least two of the considered variants and compare them on publicly available datasets like CIFAR 10/100.

Bibliography / sources:

[1] J. Zhang, J. Djolonga, A. Krause, Higher-Order Inference for Multi-class Log-supermodular Models, ICCV 2015
[2] Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy, Explaining and Harnessing Adversarial Examples, arXiv:1412.6572, 2014
[3] Gamaleldin F. Elsayed, Dilip Krishnan, Hossein Mobahi, Kevin Regan, Samy Bengio, Large Margin Deep Networks for Classification, arXiv:1803.05598v2, 2018

Name and workplace of master’s thesis supervisor:

Boris Flach, Dr. rer. nat. habil., Machine Learning, FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **14.02.2019** Deadline for master's thesis submission: **24.05.2019**

Assignment valid until: **20.09.2020**

Boris Flach, Dr. rer. nat. habil. Supervisor’s signature	Head of department’s signature	prof. Ing. Pavel Ripka, CSc. Dean’s signature