TDK-thesis

Aslan, Mahmoud

Mészáros, Balázs

Extending Sparse Dictionary Learning Methods for Adversarial Robustness

Eötvös Loránd Tudományegyetem

Faculty of Informatics

Department of Artificial Intelligence

Authors:

Aslan, Mahmoud
Computer Science for Autonomous Systems MSc

Mészáros, Balázs
Computer Science MSc

Supervisors:

Dr. habil. Lőrincz, András
Senior Researcher, PhD

Dr. Szeghy, Dávid
Assistant Professor, PhD

Budapest, 2022

Abstract

Despite their state-of-the-art performance on many tasks, deep neural networks have been shown to be vulnerable to adversarial attacks. Sparse coding methods using Basis Pursuit (BP) are appealing as they have provable robustness guarantees [1] against such attacks. In [2], these guarantees were extended to more generalized forms of regularization, including the group case and its generalizations, multi-layer extension and the Deep-Pursuit method. However, applying and scaling such methods in practice is not straightforward due to training difficulties. In this work, we lay out and further expand on our experiments reported in [3] and try to bridge the gap between theory and practice by utilizing training tricks such as batch normalization, different regularization methods, pre- and layer-wise training. Specifically, we conduct experiments on sparse, group sparse and pooled group sparse models to verify their robustness. We also study their multi-layer extensions using the Deep Pursuit architecture. To overcome BP's slowness, we consider feedforward estimations to provide inference time speed ups using linear transformers, shallow and deep dense networks. We report robustness evaluations against IFGSM attacks on a synthetic dataset and MNIST.

Contents

Chapter 1

Introduction

Overcoming the vulnerability of deep neural networks against adversarial attacks has been a relevant part of machine learning in the past. These attacks have access to the structure of the network, including the loss function. They are able to efficiently influence the outputs of the networks by slightly modifying the input towards the sign of the gradient of the loss function [4]. Algorithms such as IFGSM emphasize the importance of adversarial robustness when training neural networks. One way to overcome these issues is by applying sparse approximation methods. First, we present a few of the fields where these algorithms are applied today.

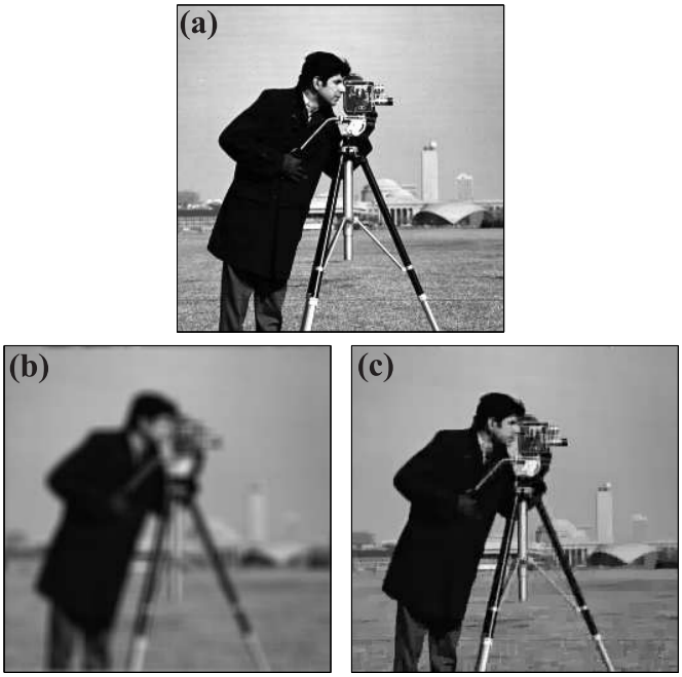

Figure 1.1: An example of an application in image processing [5]. The top image is the original, noise has been added to the bottom left, and the bottom right is the denoised image, using sparse representations.

Image processing has improved significantly in the past decades, thanks to the better modeling of image content. The manifold hypothesis [6] states that many real-life signals (images, sounds and text) lie on low-dimensional manifolds embedded into a high dimensional one. In other words, the probability distribution over the space of images for example is highly concentrated over meaningful images, giving random noise very low probability. This hypothesis motivates the sparse coding of images as an effective representation learning approach. Learning sparse codes for natural images was found to result in a complete set of localized and oriented receptive fields similar to the ones found in the primary visual cortex [7]. In [5] the authors reviewed the role of recent models that employ sparse and redundant representations. They found that image processing is one of the main beneficiaries of the developments. They show specific examples how the methods can be applied. In another paper [8] the authors found a specific field of image processing, where sparse approximation brings improvements: image decomposition, specifically- morphological decomposition of a signal into its building blocks. They built on morphological component analysis (MCA) and blind source separation (BSS), which both rely on sparsity and morphological diversity.

Popular audio coding standards such as MP3 and Dolby AAC also rely on sparse representations. Researchers have shown that there are a number of emerging applications of sparse representations related to audio, such as in audio coding, audio enhancement and music transcription to blind source separation solutions that can solve the "cocktail party problem" (focusing on the "most important" part of an audio - it could also be referred to as "selective hearing"). Audio signals are produced in a way that they are dominated by a small number of frequency components. This prior assumption that the audio signals are approximately sparse in some time-frequency representation allows the addressal of the associated signal processing task [9].

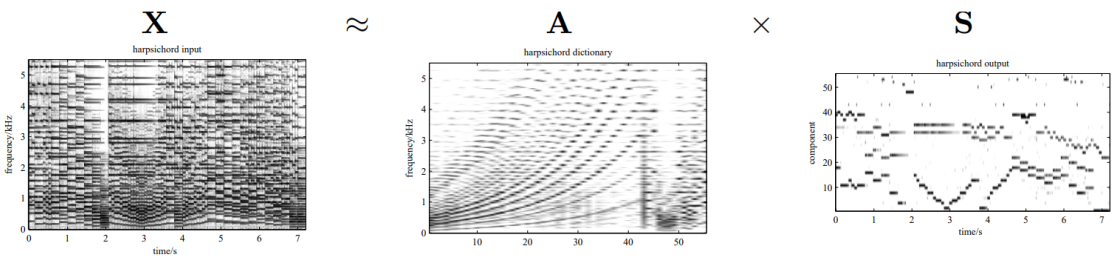

Figure 1.2: An example of an application in music [10]. The input of the model is a music spectrogram, which is transformed into a symbolic representation of the audio file - sparsely represented. Best viewed zoomed in.

There is also an increasing interest in 3D reconstruction of objects from synthetic aperture radar (SAR) measurements. Research have presented wide-angle three-dimensional image reconstruction approaches for object reconstruction that exploit reconstruction sparsity in the signal domain to ameliorate the limitations of sparse measurements [11]. Their approaches are based on high-frequency radar backscatter phenomenology so that sparse signal representations align with physical radar scattering properties of the objects of interest.

Researchers have also applied sparse methods in the field of astronomy [12]. In the past the field has relied on wavelets for different purposes, such as data filtering, data deconvolution, star and galaxy detection or cosmic ray removal. Recently, sparse representation like ridgelets or curvelets have been introduced in the field for the detection of anisotropic features such as cosmic strings in the cosmic microwave background. In the following section we give a theoretical overview of sparse approximation algorithms.

1.1 Sparse Representation

The sparse representation model assumes that the signal of interest $\mathbf{X} \in \mathbb{R}^N$ can be written as a linear combination of few elements $\mathbf{d}_i$, termed atoms, of a dictionary $\mathbf{D} \in \mathbb{R}^{N \times M}$, which can be written as:

$$\mathbf{X} = \mathbf{D}\boldsymbol{\Gamma}$$

where $\boldsymbol{\Gamma} \in \mathbb{R}^M$ is a sparse vector and the support of $\boldsymbol{\Gamma}$ is defined as the location of its non-zero coefficients. Sparse coding is the process of finding the sparsest vector $\boldsymbol{\Gamma}$ while maintaining the reconstruction of $\mathbf{X}$ using $\mathbf{D}\boldsymbol{\Gamma}$, which can be formulated as the following optimization problem:

$$\min_{\boldsymbol{\Gamma}} \|\boldsymbol{\Gamma}\|_0 \ \text{ s.t. } \ \mathbf{D}\boldsymbol{\Gamma} = \mathbf{X}, \tag{P_0}$$

where $\|\boldsymbol{\Gamma}\|_0$ denotes the number of non-zeros in the vector. In a more realistic setting, the signal gets contaminated with noise, either unintentionally via instrumental noise, or intentionally with adversarial noise, in which case we get $\mathbf{Y} = \mathbf{X} + \mathbf{E}$, where $\mathbf{E}$ has bounded energy $\|\mathbf{E}\|_2 \leq \epsilon$. Then, the recovery problem becomes:

$$\min_{\boldsymbol{\Gamma}} \|\boldsymbol{\Gamma}\|_0 \ \text{ s.t. } \ \|\mathbf{D}\boldsymbol{\Gamma} - \mathbf{Y}\|_2 \leq \epsilon, \tag{P_0^ϵ}$$

Solving Eq. (P_0) is NP-hard [13], since we need to exhaust $\binom{M}{k}$ possibilities for all choices of k that keep $\mathbf{\Gamma}$ sparse. To tackle this, approximation algorithms are used.

1.1.1 Dictionary Properties

Before proceeding to the theoretical guarantees of P_0 and P_0^ϵ, some useful properties of the dictionary need to be defined.

Spark - The Spark [14] of a dictionary $\mathbf{D}$ is the minimum number $\sigma(\mathbf{D})$ such that there exists a set of $\sigma(\mathbf{D})$ columns in $\mathbf{D}$ which are linearly dependent.

$$\sigma(\mathbf{D}) = \min_{\mathbf{x}\neq 0} \|\mathbf{x}\|_0 \quad \text{s.t.} \quad \mathbf{D}\mathbf{x} = 0$$

Computing the spark of a matrix in practice is not feasible as it is a combinatoral problem. Therefore, most of the guarantees that we will see in later sections will not depend on the spark, instead it will use the relation between the spark and mutual coherence to define the guarantees in terms of the latter, which we define next.

Mutual Coherence - Quantifies the similarity between different atoms from the dictionary $\mathbf{D}$. For simplicity, assuming $\|\mathbf{d}_i\|_2 = 1, \ \forall i$, the mutual coherence of a dictionary $\mathbf{D}$ is defined as:

$$\mu(\mathbf{D}) = \max_{i\neq j} \|\mathbf{d}_i^T \mathbf{d}_j\|_2$$

Although cheaper to compute, mutual coherence along with the spark are considered pessimistic properties, i.e. they capture the worst case, for example for a dictionary whose atoms are all mutually orthogonal except for one atom which is a duplicate, in this case the mutual coherence will be equal to 1, giving a bad impression about the incoherence of the dictionary. In practice though, one cannot expect zero mutual coherence for overcomplete matrices, the best family of such matrices that can achieve the lowest mutual coherence possible are the Grassmanian Frames [15].

Other properties of the dictionary exist in the literature such as the Restricted Isometry Property (RIP) [16], in this work we only report the theoretical results that use the properties we already described.

In the classical literature of sparse coding, dictionaries were constructed using analytical waveforms such as Wavelets [17] and Fourier [18] and others or as a merge of these to create megadictionaries. Although those were successful, learning the dictionary from data promises more sparsity and adaptation to the task at hand. In this work, we adopt the learning approach using gradient-based optimization methods. Properly

training the dictionary while maintaining desirable properties such as low mutual coherence is still an open area of research.

Theoretical guarantees for $\mathbf{P}_0$ and $\mathbf{P}_0^\epsilon$ - The solution of P_0 is unique [14] given that the representation satisfies the condition

$$\|\mathbf{\Gamma}\|_0 < \frac{1}{2}(1 + \frac{1}{\sigma(\mathbf{D})})$$

An equivalent condition using the spark of $\mathbf{D}$ could be used, however, as mensioned earlier it's infeasible to compute the spark for practical problems.

On the other hand, P_0^ϵ has no unique solution due to the noise, but we can guarantee stability of the obtained solution such that its deviation will be bounded [19].

1.2 Approximation Algorithms

Solutions to Eq. (P_0) can be approximated using greedy algorithms that build the solution to be locally optimal each step at a time, this includes Matching Pursuit (MP) and its variant Orthogonal Matching Pursuit (OMP) [20, 21] or algorithms based on convex relaxations such as Basis Pursuit (BP) [22], these algorithms are described next.

Greedy Matching Pursuit - Matching Pursuit [20] finds the "best matching" projections of multidimensional data onto the span of an over-complete dictionary $\boldsymbol{D}$, approximately representing a signal $\boldsymbol{X}$ from Hilbert space $\boldsymbol{H}$ as a weighted sum of finitely many atoms, which are columns of $\boldsymbol{D}$. Matching Pursuit does not use all atoms in the sum, but it chooses one at a time in order to greedily reduce the approximation error. It finds the atom with the highest inner product with $\boldsymbol{X}$ then it subtracts an approximation that uses only that one atom from the signal, and repeats the process until the norm of the residual is satisfactorily small. This is denoted by the following:

$$\boldsymbol{R_{N+1}} = \boldsymbol{X} - \hat{\boldsymbol{X}}_N, \tag{1.1}$$

where $\boldsymbol{N}$ is the number of atoms, $\boldsymbol{R_{N+1}}$ is the residual corresponding to the N-th atom, and $\hat{\boldsymbol{X}}_N$ is given by:

$$\hat{\boldsymbol{X}}_N = \sum_{\boldsymbol{n}=1}^{\boldsymbol{N}} \mathbf{\Gamma}_{\boldsymbol{n}} \boldsymbol{d_n}, \tag{1.2}$$

where $\mathbf{\Gamma}_{\boldsymbol{n}}$ could be thought of as a weighting factor and $\mathbf{d}_n$ is the $\boldsymbol{n}$-th column of $\boldsymbol{D}$. If $\boldsymbol{R_n}$ quickly converges to zero that means that only a few atoms are needed, which is the aim of

sparse representations. This gives the sparsity problem that MP is approximately solving:

$$\min_{\bar{\boldsymbol{\Gamma}}} \left\| \boldsymbol{X} - \boldsymbol{D}\bar{\boldsymbol{\Gamma}} \right\|_2^2 \text{ subject to } \left\| \bar{\boldsymbol{\Gamma}} \right\|_0 \leq \boldsymbol{N} \tag{1.3}$$

MP was extended with the **Orthogonal Matching Pursuit** [21] algorithm. OMP updates all the coefficients that have been extracted up until that point in all steps. This is done by taking the subspace spanned by the so far selected atoms and computing the orthogonal projection of the signal on this. This leads to better results than traditional MP and certain restricted isometry conditions bring guaranteed stability and performance, but the required computational power increases.

ℓ_1 **relaxation** - When the convex relaxation of Eq. (P_0) is based on the ℓ_1 norm we get the **Basis Pursuit** optimization problem [23], which can be written as follows:

$$\min_{\hat{\boldsymbol{\Gamma}}} \|\hat{\boldsymbol{\Gamma}}\|_1 \text{ s.t. } \mathbf{D}\hat{\boldsymbol{\Gamma}} = \mathbf{X}, \tag{P_1}$$

and in the noisy case, we get Basis Pursuit Denoising (BPDN)

$$\min_{\hat{\Gamma}} \|\hat{\boldsymbol{\Gamma}}\|_1 \text{ s.t. } \|\mathbf{D}\hat{\boldsymbol{\Gamma}} - \mathbf{Y}\|_2 \leq \epsilon, \tag{P_1^ϵ}$$

We can reformulate Eq. (P_1) as the following unconstrained problem:

$$\arg\min_{\hat{\boldsymbol{\Gamma}}} \frac{1}{2}\|\mathbf{D}\hat{\boldsymbol{\Gamma}} - \mathbf{X}\|_2^2 + \gamma\|\hat{\boldsymbol{\Gamma}}\|_1, \tag{BP}$$

where $\gamma > 0$.

Thresholding Algorithms - Solutions to Eq. (BP) include thresholding algorithms using the soft $S_\gamma(.)$ and hard $H_\gamma(.)$ thresholding operators, where these operators are defined as follows for $\gamma > 0$:

$$S_\gamma(\boldsymbol{x}) = \begin{cases} \boldsymbol{x} - \gamma & \boldsymbol{x} > \gamma \\ 0 & -\gamma \leq x \leq \gamma \\ \boldsymbol{x} + \gamma & \boldsymbol{x} < -\gamma \end{cases} \tag{ST}$$

$$H_\gamma(\boldsymbol{x}) = \begin{cases} \boldsymbol{x} & \boldsymbol{x} > \gamma \vee \boldsymbol{x} < -\gamma \\ 0 & -\gamma \leq x \leq \gamma \end{cases} \tag{HT}$$

These algorithms use a closed-form solution to the BP problem by simply multiplying the dictionary by the signal to get a vector of coefficients, then pass this vector into the thresholding operator to zero out the coefficients lower than the threshold γ, and

additionally, in the soft thresholding case, the coefficients above the threshold are shrinked as shown in the soft thresholding operator definition above. This simple algorithm can be written as $H_\gamma(\mathbf{D}^T\mathbf{X})$ and $S_\gamma(\mathbf{D}^T\mathbf{X})$ for the Hard and Soft thresholding respectively.

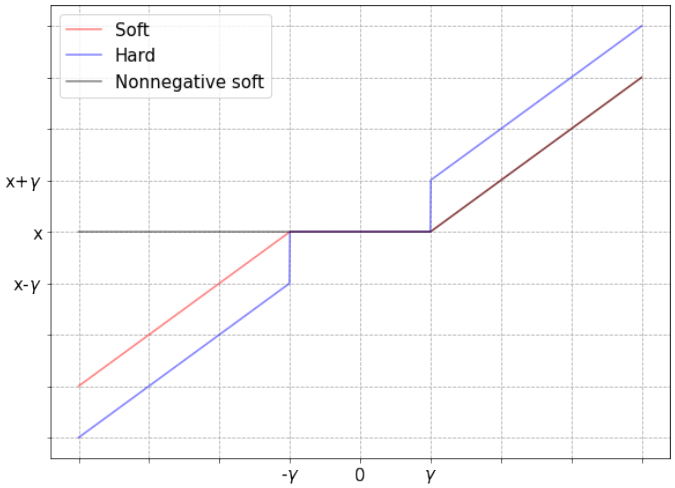

Figure 1.3: Soft vs hard vs nonnegative soft thresholding

A non-negative soft thresholding operator $S^+_\gamma(.)$ is defined as:

$$S^+_\gamma(\boldsymbol{x}) = \begin{cases} 0 & \boldsymbol{x} \leq \gamma \\ \boldsymbol{x} - \gamma & \gamma < \boldsymbol{x} \end{cases} \tag{NNST}$$

where we assume that the representation can only contain non-negative coefficients. This assumption doesn't affect the expressiveness of the model, but prove to be useful when making the connection between the forward pass of Convolutional Neural Networks (CNNs) [24] and Convolutional Sparse Coding (CSC) as we will discuss in later sections. For now we can see that the non-negative threholding operator is equivalent to the Rectified Linear Unit (ReLU); a commonly used non-linearity in state-of-the-art neural networks, where these two are related as follows:

$$S^+_\gamma(\boldsymbol{x}) = max(\boldsymbol{x} - \gamma, 0) = ReLU(\boldsymbol{x} - \gamma)$$

See figure 1.3 to get a better idea of the outputs of these functions. A more involved solution to Eq. (BP) uses the Iterative Shrinking and Thresholding Algorithm (ISTA) and its variant Fast-ISTA (FISTA) [25]. One can think of ISTA as repeating two minimization steps, the first to minimize the reconstruction part of Eq. (BP) using the gradient, then to minimize the ℓ_1 norm using the chosen thresholding operator. A more formal description can be seen in Algorithm 1.

Algorithm 1 ISTA algorithm

$\mathbf{\Gamma^0} := \mathbf{X}$
for $t \leftarrow 1$ **to** T **do**
    $\boldsymbol{\Gamma^t} := \boldsymbol{\phi_\gamma}(\boldsymbol{\Gamma^{t-1}} - \frac{1}{L\beta}\boldsymbol{g^t})$
end for

where the gradient $\boldsymbol{g^t}$ is derived as:

$$\boldsymbol{g^t} := \boldsymbol{D}^T(\boldsymbol{D\Gamma^{t-1}} - \boldsymbol{x})$$

$\boldsymbol{T}$ is the number of gradient iterations and $\boldsymbol{\phi}$ is the thresholding operator, where parameter $\boldsymbol{\gamma}$ controls how much we want to threshold the signal. $\boldsymbol{L}$ is the conservative Lipschitz constant for guaranteed convergence [25] and $\boldsymbol{\beta}$ is a trainable parameter (to allow the network to learn how to take larger steps).

ISTA is guaranteed [26] to converge if c is chosen to satisfy the condition: $c > \lambda_{max}(\mathbf{D}^T\mathbf{D})$ where $\lambda_{max}(\mathbf{D}^T\mathbf{D})$ is the maximum eigenvalue of the matrix $\mathbf{D}^T\mathbf{D}$.

FISTA [25] simply attempts to speedup the convergence of ISTA by giving more momentum to the update rule, which updates the representation using an extrapolation of the previous ones, a description of the FISTA algorithm can be seen in Algorithm 2. Similar convergence guarantees for FISTA were given in [25].

Algorithm 2 FISTA algorithm

$\mathbf{\Gamma}^1_{extrap} := \mathbf{X}$
$r_1 := 1$
for $t \leftarrow 1$ **to** T **do**
    $\boldsymbol{\Gamma^t} := \boldsymbol{\phi_\gamma}(\boldsymbol{\Gamma}^t_{extrap} - \frac{1}{L\beta}\boldsymbol{g^t})$
    $r_{t+1} := (1 + \sqrt{1 + 4r_t^2})/2$
    $\boldsymbol{\Gamma}^{t+1}_{extrap} := \boldsymbol{\Gamma^t} + \frac{r_t - 1}{r_{t+1}}(\boldsymbol{\Gamma^t} - \boldsymbol{\Gamma^{t-1}})$
end for

The gradient $\boldsymbol{g^t}$ is identical to the one derived for ISTA in Algorithm 1.

Theoretical guarantees for $\mathbf{P}_1$, $\mathbf{P}_1^\epsilon$ - The solution of P_1 is unique and the recovery of it using OMP and BP is guaranteed [27] [14] given that the representation satisfies the sparsity condition:

$$\|\mathbf{\Gamma}\|_0 < \frac{1}{2}(1 + \frac{1}{\sigma(\mathbf{D})})$$

As for thresholding (soft and hard) the recovery of the true support is guaranteed but it can't recover the exact coefficients [14]. The guarantee depends on properties of $\mathbf{D}$ and on

the ratio between absolute magnitudes of the max and min coefficients of $\mathbf{\Gamma}$, specifically $\frac{|\mathbf{\Gamma}_{min}|}{|\mathbf{\Gamma}_{max}|}$.

Similarly to P_0^ϵ, the solution to P_1^ϵ is not unique but its stability is proven as well as the stability of its approximations using OMP and BP assuming sufficient sparsity [28].

1.3 Convolutional Sparse Coding

The Convolutional Sparse Coding (CSC) model assumes that the signal $\mathbf{X} \in \mathbb{R}^N$ can be represented as a multiplication of a globally sparse representation $\mathbf{\Gamma}$ with a global convolutional dictionary $\mathbf{D} \in \mathbb{R}^{N \times Nm}$ where m is the number of filters each of length n placed in $\mathbf{D}$ with shifts to give rise to the banded circulant matrix $\mathbf{D}$ [29]. Then the local convolutional dictionary is $\mathbf{D}_L \in \mathbb{R}^{n \times m}$.

The classical sparse approximation results would make the number of non-zeros in the representation of the CSC model to be independent of the size of the signal N, which makes it impractical for large enough signals. In [30], a local approach is taken to the CSC model, which will improve those bounds. The reformulation starts by defining a *stripe dictionary* $\mathbf{\Omega} = \mathbf{R}_i \mathbf{D} \mathbf{S}_i^T \in \mathbb{R}^{n \times (2n-1)m}$, where $\mathbf{R}_i \in \mathbb{R}^{n \times N}$ and $\mathbf{S}_i \in \mathbb{R}^{(2n-1)m \times mN}$ are operators the former is a patch extraction operator that extracts n rows from $\mathbf{D}$ at location i and the latter removes columns of zeros from $\mathbf{R}_i \mathbf{D}$ as defined in [30]. Given the stripe dictionary, the signal patch $x_i \in \mathbb{R}^n$ is defined as $\mathbf{x}_i = \mathbf{R}_i \mathbf{D} \mathbf{\Gamma} = \mathbf{R}_i \mathbf{D} \mathbf{S}_i^T \mathbf{S}_i \mathbf{\Gamma}$, where $\mathbf{S}_i \mathbf{\Gamma}$ is the stripe representation at location i. Figure 1.4 taken from [30] depicts the structure of the stripe formulation of the CSC model.

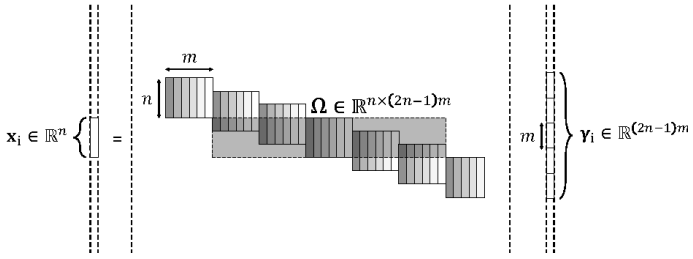

Figure 1.4: Stripe Dictionary [30]

Given the stripe dictionary structure, the $P_{0,\infty}$ objective introduced in [30] is defined as follows:

$$\min_{\mathbf{\Gamma}} \|\mathbf{\Gamma}\|_{0,\infty} \ \ s.t. \ \ \mathbf{D}\mathbf{\Gamma} = \mathbf{X} \qquad (P_{0,\infty})$$

where $\|\mathbf{\Gamma}\|_{0,\infty} = \max_i \|\mathbf{S}_i\mathbf{\Gamma}\|_0$. By solving this objective, the sparsity constraints are on the local stripes rather than globally, which makes the theoretical bounds much more practical.

Similarly, the $\mathrm{P}^{\epsilon}_{0,\infty}$ objective is defined for $\mathbf{Y} = \mathbf{X} + \mathbf{E}$ as follows:

$$\min_{\mathbf{\Gamma}} \|\mathbf{\Gamma}\|_{0,\infty} \;\; s.t. \;\; \|\mathbf{D}\mathbf{\Gamma} - \mathbf{Y}\|_2 \leq \epsilon \qquad (\mathrm{P}^{\epsilon}_{0,\infty})$$

Theoretical guarantees for $\mathbf{P}_{0,\infty}$ and $\mathbf{P}^{\epsilon}_{0,\infty}$ - Uniqueness of $\mathrm{P}_{0,\infty}$ is proven in [30] given the condition on sparsity $\|\mathbf{\Gamma}\|_0 < \frac{1}{2}(1 + \frac{1}{\sigma(\mathbf{D})})$ is met. Recovery of this solution using OMP and BP is guaranteed with similar conditions.

Similar to P^{ϵ}_0, the solution to $\mathrm{P}^{\epsilon}_{0,\infty}$ is not unique, although it is proved to be stable, similar stability guarantees are derived for approximations using OMP and BP, all assuming sufficient local sparsity [30].

1.4 Multi-Layer CSC

In [31], the CSC model was extended to its multi-layer version, in which the sparse representation itself $\mathbf{X} = \mathbf{D}_1\mathbf{\Gamma}_1$ is further decoded as a linear combination of atoms taken from a different convolutional dictionary $\mathbf{\Gamma}_1 = \mathbf{D}_2\mathbf{\Gamma}_2$, and the same is assumed for $\mathbf{\Gamma}_2$,...,$\mathbf{\Gamma}_L$, where L is the number of layers.

The intuition behind the model is building a hierarchy of dictionaries, where the elements of the first dictionary form atoms, and the elements of the first and second dictionaries $\mathbf{D}_1\mathbf{D}_2$ form molecules, and as we add more layers the resultant global dictionary $\mathbf{D}_1...\mathbf{D}_L$, will form more and more complex and structured elements.

Then the *Deep Coding Problem* (DCP_{λ}) [31] is defined as follows:

$$\text{Find} \;\; \{\mathbf{\Gamma}_i\}_{i=1}^{L} \;\; s.t. \;\; \mathbf{\Gamma}_{i-1} = \mathbf{D}_i\mathbf{\Gamma}_i, \;\; \|\mathbf{\Gamma}_i\|_{0,\infty} \leq \lambda_i, \;\; \forall 1 \leq i \leq L. \qquad (\mathrm{DCP}_{\lambda})$$

where $\mathbf{\Gamma}_0 = \mathbf{X}$. In the noisy case ($\mathrm{DCP}^{\epsilon}_{\lambda}$) is defined as:

$$\text{Find} \;\; \{\mathbf{\Gamma}_i\}_{i=1}^{L} \;\; s.t. \;\; \|\mathbf{\Gamma}_{i-1} - \mathbf{D}_i\mathbf{\Gamma}_i\|_2 \leq \epsilon_{i-1}, \;\; \|\mathbf{\Gamma}_i\|_{0,\infty} \leq \lambda_i, \;\; \forall 1 \leq i \leq L. \qquad (\mathrm{DCP}^{\epsilon}_{\lambda})$$

where $\mathbf{\Gamma}_0 = \mathbf{Y} = \mathbf{X} + \mathbf{E}$.

The thresholding pursuit algorithm can be extended to its multi-layer case to work with the ML-CSC, its coined as the *layered thresholding pursuit* (LTH) [31], the pseudo-code of it is summarized in Algorithm 3.

Algorithm 3 Layered Thresholding

1: $\mathbf{\Gamma}_0 := \mathbf{X}$
2: **for** $i \leftarrow 1$ **to** l **do**
3: $\quad \mathbf{\Gamma}_i := \boldsymbol{\phi}_{\gamma_i}(\mathbf{D}_i^T \mathbf{\Gamma}_{i-1})$
4: **end for**

One of the main points of [31] is building the connection between CNNs and ML-CSC, so that we can benefit from the theoretical analysis of ML-CSC in better understanding CNNs. The authors make the case that the forward pass of CNNs and theresholding pursuit of ML-CSC are tightly connected and are made equal when the chosen pursuit algorithm is the layered soft non-negative thresholding.

Theoretical guarantees for DCP_λ and $\text{DCP}_\lambda^\epsilon$ - The DCP_λ solution is unique and the exact recovery of the solution using LTH is not guaranteed, an extra step is needed to project onto the found support. As for the $\text{DCP}_\lambda^\epsilon$, its stability is proven with bounded deviations. All of which are assuming enough sparsity levels. For more details refer to [31].

The incapability of LTH to recover the exact coefficients of the DCP_λ solution and the dependency of its support recovery condition on the ratio between its maximum and minimum absolute coefficients $\frac{|\mathbf{\Gamma}_{min}|}{|\mathbf{\Gamma}_{max}|}$ motivate for another pursuit algorithm that avoids these issues, which is what we discuss next.

1.5 Layered BP

The layered Basis Pursuit (LBP) [31] can be formulated as follows:

$$\arg\min_{\mathbf{\Gamma}_i} \frac{1}{2}\|\mathbf{D}_i\mathbf{\Gamma}_i - \hat{\mathbf{\Gamma}}_{i-1}\|_2^2 + \gamma_i\|\mathbf{\Gamma}_i\|_1 \tag{LBP}$$

where $\mathbf{\Gamma}_0 = \mathbf{X}$ and $\gamma_i > 0, \ \ \forall 1 \leq i \leq L$, where L is the number of layers.

Solutions to BP such as iterative soft thresholding can be extended to solve LBP. Specifically, the Layered Iterative Soft Thresholding (Layered IST) repeats the following update rule in every layer:

$$\hat{\mathbf{\Gamma}}_i^t = S_{\gamma_i/c_i}(\hat{\mathbf{\Gamma}}_i^{t-1} + \frac{1}{c_i}\mathbf{D}_i^T(\mathbf{\Gamma}_{i-1} - \mathbf{D}_i\hat{\mathbf{\Gamma}}_i^{t-1}))$$

where i refers to the i-th layer, and t to the current iteration. The convergence to a global minimum is guaranteed if $c_i > \frac{1}{2}\lambda_{max}(\mathbf{D}_i^T\mathbf{D}_i)$ [26]. Algorithm 4 details the Layered ISTA

algorithm.

Algorithm 4 Layered ISTA algorithm

$\hat{\mathbf{\Gamma}}_0 := \mathbf{X}$
for $j \leftarrow 1$ **to** l **do**
  $\mathbf{\Gamma}_j^0 := \boldsymbol{\phi}_{\gamma_j}(\mathbf{D}_j^T \hat{\mathbf{\Gamma}}_{j-1})$
  for $t \leftarrow 1$ **to** T **do**
    $\mathbf{\Gamma}_j^t := \boldsymbol{\phi}_{\gamma_j}(\mathbf{\Gamma}_j^{t-1} - \frac{1}{L_j \beta_j} \boldsymbol{g}_j^t)$
  end for
end for

where $\hat{\mathbf{\Gamma}}_j$ is the estimated representation of layer j, and the gradient $\boldsymbol{g}^t$ is identical to the one derived for Algorithm 1 but with subscripts indicating the layer number:

$$\boldsymbol{g}_j^t := \mathbf{D}_j^T(\mathbf{D}_j \mathbf{\Gamma}_j^{t-1} - \hat{\mathbf{\Gamma}}_{j-1})$$

Note that we can get LTH by setting $T = 1$. Similarly, one can obtain Layered FISTA by changing the update rule to include the extrapolation as done in algorithm 2. ISTA is interesting since it can be thought of as a recurrent neural network [32], and the same can be said for Layered ISTA as a deeper network. Alternatively, it can be looked at as blocks of convolutional layers with shared weights and skip connections between them in order to compute the residual $\mathbf{\Gamma}_{i-1} - \mathbf{D}_i \hat{\mathbf{\Gamma}}_i^{t-1}$.

Theoretical guarantees for LBP - The LBP is guaranteed to recover the unique solution of DCP_λ given enough sparsity. As for $\text{DCP}_\lambda^\epsilon$, the LBP is stable and its bound doesn't depend on the term $\frac{|\mathbf{\Gamma}_{min}|}{|\mathbf{\Gamma}_{max}|}$. Note that although these results are made for the convolutional case, but the authors claim that they apply for the fully connected case as well [31].

1.6 Deep Pursuit

Solutions to the LBP suffer from error accumulation [33]; given the noisy signal $\mathbf{Y} = \mathbf{X} + \mathbf{E}$ and $\|\mathbf{E}\|_2 < \epsilon_0$, then the ϵ_i for later layers are shown to be amplified if no proper regularization is applied, which makes these solutions not suitable for deeper networks.

Motivated to solve this issue and to incorporate state-of-the-art neural architectural ideas such as skip connections, [34] proposed a global view of the LBP objective, as a single sparse coding problem in which all the representations are updated synchronously, stopping the error from accumulating as in LBP. The *Deep Pursuit* objective is defined as

follows:

$$\underset{\boldsymbol{\Gamma}_j,\ j\in\{1,\ldots,l\}}{\arg\min}\ \frac{1}{2}\sum_{j=1}^{l}\|\boldsymbol{\Gamma}_{j-1}-\mathbf{D}_j\boldsymbol{\Gamma}_j\|_2^2+\gamma_j\|\boldsymbol{\Gamma}_j\|_1 \tag{DP}$$

or in a matrix form:

$$\underset{\boldsymbol{\Gamma}_j,\ j\in\{1,\ldots,l\}}{\arg\min}\ \frac{1}{2}\left\|\begin{bmatrix}\mathbf{X}\\\mathbf{0}\\\vdots\\\mathbf{0}\end{bmatrix}-\begin{bmatrix}\mathbf{D}_1 & & \cdots & \mathbf{0}\\-\mathbf{I} & \mathbf{D}_2 & & \vdots\\\vdots & \ddots & \ddots & \\\mathbf{0} & & -\mathbf{I} & \mathbf{D}_l\end{bmatrix}\begin{bmatrix}\boldsymbol{\Gamma}_1\\\boldsymbol{\Gamma}_2\\\vdots\\\boldsymbol{\Gamma}_l\end{bmatrix}\right\|_2^2+\sum_{j=1}^{l}\gamma_j\left\|\boldsymbol{\Gamma}_j\right\|_1$$

Adding skip connections to this architecture is done by adding non-zero entries in the lower triangle of the matrix, which gives us the generalized form:

$$\begin{bmatrix}\mathbf{D}_1 & & \cdots & \mathbf{0}\\-\mathbf{D}_{21}^T & \mathbf{D}_2 & & \vdots\\\vdots & \ddots & \ddots & \\-\mathbf{D}_{l1}^T & & -\mathbf{D}_{l(l-1)}^T & \mathbf{D}_l\end{bmatrix}$$

In [3], even a more generalized form is presented where the upper triangular part is non-zero as well.

The DP solution is similar to the Layered IST, with the loop order reversed. Details of the Deep Pursuit algorithm can be seen in Algorithm 5.

Algorithm 5 Deep Pursuit

$\hat{\boldsymbol{\Gamma}}_0 := \mathbf{X}$
for $j \leftarrow 1$ **to** l **do**
    $\hat{\boldsymbol{\Gamma}}_j^0 := \boldsymbol{\phi}_{\boldsymbol{\gamma}_j}(\mathbf{D}_j^T\hat{\boldsymbol{\Gamma}}_{\boldsymbol{j-1}}^{\mathbf{0}})$
end for
for $t \leftarrow 1$ **to** T **do**
    for $j \leftarrow 1$ **to** l **do**
        $\boldsymbol{\Gamma}_j^t := \boldsymbol{\phi}_{\boldsymbol{\gamma}_j}(\hat{\boldsymbol{\Gamma}}_j^{\boldsymbol{t-1}} - \frac{1}{L_j\beta_j}\hat{\boldsymbol{g}}_j^t)$
    end for
end for

where the gradient is given by:

$$\hat{\boldsymbol{g}}_j^t := \begin{cases}\mathbf{D}_j^T(\mathbf{D}_j\hat{\boldsymbol{\Gamma}}_j^{\boldsymbol{t-1}} - \boldsymbol{\Gamma}_{\boldsymbol{j-1}}^t) + (\hat{\boldsymbol{\Gamma}}_j^{\boldsymbol{t-1}} - \mathbf{D}_{j+1}\boldsymbol{\Gamma}_{\boldsymbol{j+1}}^{\boldsymbol{t-1}}) & j < l\\ \mathbf{D}_j^T(\mathbf{D}_j\hat{\boldsymbol{\Gamma}}_j^{\boldsymbol{t-1}} - \boldsymbol{\Gamma}_{\boldsymbol{j-1}}^t) & j = l\end{cases}$$

and in the skip connection case:

$$\hat{\boldsymbol{g}}_{\boldsymbol{j}} := \mathbf{D}_j^T(\mathbf{D}_j\hat{\boldsymbol{\Gamma}}_{\boldsymbol{j}} - \sum_{k=1}^{j-1}\mathbf{D}_{j,k}^T\boldsymbol{\Gamma}_{\boldsymbol{k}}) + \sum_{j'=j+1}^{l}\mathbf{D}_{j'j}(\sum_{k'=1}^{j'-1}\mathbf{D}_{j'k'}^T\boldsymbol{\Gamma}_{k'} - \mathbf{D}_{j'}\boldsymbol{\Gamma}_{\boldsymbol{j'}})$$

and the extrapolation of $\boldsymbol{\Gamma}_j^{t-1}$ for $t > 1$ is given by:

$$\hat{\boldsymbol{\Gamma}}_{\boldsymbol{j}}^{\boldsymbol{t-1}} := \boldsymbol{\Gamma}_j^{t-1} + \boldsymbol{\alpha}_{\boldsymbol{j}}(\boldsymbol{\Gamma}_{\boldsymbol{j}}^{\boldsymbol{t-1}} - \boldsymbol{\Gamma}_{\boldsymbol{j}}^{\boldsymbol{t-2}})$$

Theoretical guarantees for DP - Stability guarantees of DP were not provided in the original paper, but were later derived in [3], which we discuss its contributions next.

1.7 Group Pursuit

In [3], the sparse code is allowed to form groups by using the $\ell_1, \ell_2, \ell_{1,2}$, and $\ell_{\beta,1,2}$ norms as a regularization in BP. One advantage of this grouping method is that the ℓ_2 norms of the groups can be efficiently estimated using FeedForward networks or transformers.

The general form of the Group Basis Pursuit (GBP) is as follows:

$$\arg\min_{\hat{\boldsymbol{\Gamma}}} \frac{1}{2}\|\mathbf{X} - \mathbf{D}\hat{\boldsymbol{\Gamma}}\|_2^2 + < \boldsymbol{\gamma}, l(\hat{\boldsymbol{\Gamma}}) > \qquad \text{(GBP)}$$

where $< \boldsymbol{\gamma}, l(\hat{\boldsymbol{\Gamma}}) >= \sum_{i=1}^{K}\gamma_i l_{\alpha_i}(\hat{\boldsymbol{\Gamma}}_{G_i})$ is a generalized regularizer with γ_i being the weight for the norm of group i, $\gamma_i > 0$, and $l : \mathbb{R}^n \to \mathbb{R}^K$ where $l(\hat{\boldsymbol{\Gamma}})$ is a vector whose elements are possibly different norms evaluated on different groups of $\hat{\boldsymbol{\Gamma}}$, and $\hat{\boldsymbol{\Gamma}}_{G_i}$ is a part of $\hat{\boldsymbol{\Gamma}}$ corresponding to group G_i. G_i is a partition, $i \in \{1, ..., K\}$, of the index set $\{1, ..., M\}$. Then the $\ell_{1,2}$ norm [35] is defined as follows:

$$\|\hat{\boldsymbol{\Gamma}}\|_{1,2} = \sum_{i=1}^{K}\|\hat{\boldsymbol{\Gamma}}_{G_i}\|_2$$

This norm arises in the generalized regularizer we described above if the ℓ_2 norm is used for multiple groups with the same weight γ.

Similar to LBP, the GBP objective can be exteneded to the multi-layer case.

Theoretical guarantees for GBP - [3] provides proofs for the stability of the GBP solvers, given a constrained perturbation error. It also extends these stability theorems to the multi-layer case (LGBP) and also include the DP case described in the previous

section. Finally, the paper derives stability results for the GBP and LGBP problems with a linear classifier stacked on top, to study their classification robustness.

The rest of this work is an expansion on the experiments done in [3].

Chapter 2

Methods

The aim of our studies was to evaluate the empirical robustness of our various methods. In the first phase we focused on comparing group based methods with non-group based methods, and checking if these could be effectively sped up. In the second phase we investigated if we could improve the results by deepening our networks.

2.1 Architectures

To evaluate the empirical robustness of our shallow Group Basis Pursuit (GBP) with ℓ_2 norm regularization, we compared two variants of it with Basis Pursuit (BP) and 3 Feedforward networks.

We used a single BP layer to compute the hidden representation $\mathbf{\Gamma}_{BP}$, then stacked a classifier $\boldsymbol{w}$ on top.

For GBP, we considered two scenarios. In the first case, we applied GBP on its own to compute a full $\mathbf{\Gamma}_{GBP}$ code. In the second case, we introduced a method that we named *Pooled GBP (PGBP)*. First we computed $\mathbf{\Gamma}_{GBP}$ with GBP, then we compressed it with a per group ℓ_2 norm calculation into $\mathbf{\Gamma}_{PGBP}$, and used this smaller code as input to a smaller classifier $\boldsymbol{w}_{PGBP}$.

Then, we employed 3 feedforward neural networks trained for approximating $\mathbf{\Gamma}_{PGBP}$: a Linear Transformer [36], a single dense layer, and a dense deep network having parameter count similar to the Transformer, to confirm that the Transformer's superiority is not due to the higher parameter count. For the nonnegative norm values, we used Rectified Linear Unit (ReLU) activation at the top of these networks. To mitigate vanishing gradients, we added a batch normalization layer. We trained these methods to approximate the pooled

$\hat{\boldsymbol{\Gamma}}_{PGBP}$, and fed this to the smaller $\boldsymbol{w}_{PGBP}$ classifier. We did not try to approximate ***BP*** and ***GBP*** with our feedforward architectures.

In the second phase we compared Layered Basis Pursuit (LBP) with Deep Pursuit (DP). We implemented 3 layer networks, similarly to the previous phase: one sparse architecture (BP) and two group architectures (GBP, PGBP). These worked the same way as the shallow networks- first they computed the hidden representations ($\boldsymbol{\Gamma}_{BP1,}$, $\boldsymbol{\Gamma}_{BP2,}$ $\boldsymbol{\Gamma}_{BP3,}$, or $\boldsymbol{\Gamma}_{GBP1,}$, $\boldsymbol{\Gamma}_{GBP2,}$ $\boldsymbol{\Gamma}_{GBP3,}$, or $\boldsymbol{\Gamma}_{GBP1,}$, $\boldsymbol{\Gamma}_{GBP2,}$ $\boldsymbol{\Gamma}_{PGBP3,}$), and a classifier was stacked on top ($\boldsymbol{w}_{BP}$ or $\boldsymbol{w}_{PGBP}$).

For Basis Pursuit a proximal operator $\boldsymbol{\phi}$ needs to be applied for shrinkage. In the shallow networks we only used the soft thresholding operator (ST), but with the deeper networks we also tried hard thresholding (HT). For the exact sizes of our neural network architectures, see Appendix A.

2.2 Loss Functions

Our loss function contained several terms, and they were multiplied with various scalar weights, which we tuned manually.

The unsupervised reconstruction loss was included in all cases:

$$\|\boldsymbol{X} - \boldsymbol{D}\boldsymbol{\Gamma}\|_2^2. \tag{2.1}$$

In the first phase, the atoms of the dictionaries were manually forced to have a norm of 1, but in the second phase we introduced the following term into the loss function:

$$\frac{\sum_{i=1}^{\boldsymbol{l}} |\sum_{j=1}^{\boldsymbol{s_i}} 1 - \|\boldsymbol{d_{ij}}\|_2|}{\boldsymbol{l}\sum_{i=1}^{\boldsymbol{l}} s_i}, \tag{2.2}$$

where $\boldsymbol{l}$ is the number of layers, $\boldsymbol{s_i}$ is the size of layer $\boldsymbol{i}$ and $\boldsymbol{d_{ij}}$ is the $\boldsymbol{j}$-th atom of layer $\boldsymbol{i}$. We decided to take the mean of the values (and not the sum) so that it does not influence the training too heavily, but this could have also been done by decreasing the corresponding weighting scalar.

The mutual coherence loss in the sparse case:

$$\frac{\sum_{i=1}^{\boldsymbol{l}} \frac{1}{s_i}\|\boldsymbol{I_{s_i}} - \boldsymbol{D_i^T}\cdot\boldsymbol{D_i}\|_F}{\boldsymbol{l}}, \tag{2.3}$$

and in the group case:

$$\frac{\sum_{i=1}^{\boldsymbol{l}} \frac{1}{s_i} \|(\boldsymbol{I}_{\boldsymbol{s_i}} \otimes \boldsymbol{J}_{\boldsymbol{s_i/g_i}}) - ((\boldsymbol{J}_{s_i} - \boldsymbol{I}_{s_i}) \times (\boldsymbol{D}_{\boldsymbol{i}}^T \cdot \boldsymbol{D}_{\boldsymbol{i}}))\|_F}{\boldsymbol{l}}, \tag{2.4}$$

Here, $\boldsymbol{g_i}$ is the number of groups of the dictionary of layer $\boldsymbol{i}$, $\boldsymbol{I}$ is the identity matrix and $\boldsymbol{J}$ is the all-ones matrix. Similarly to the atom norm loss, we calculated the means, but decreasing the weighting scalar would have done the same.

We introduced the regularization loss to test whether it can further improve the adversarial robustness. The aim of the *gap regularization* term was to encourage a better separation between active and inactive groups. Our intention was to increase the smallest difference of preactivations between the smallest active and the largest inactive group norm within a mini-batch of $\boldsymbol{\Gamma}_{(G)BP}$ samples:

$$-\min_{i=1,\dots,N}\Big(\min_{j:\,\phi_\gamma\left(||\boldsymbol{\Gamma}^{(i)}_{(G)BP,G_j}||_2\right)\neq 0} ||\boldsymbol{\Gamma}^{(i)}_{(G)BP,G_j}||_2 - \max_{j:\,\phi_\gamma\left(||\boldsymbol{\Gamma}^{(i)}_{(G)BP,G_j}||_2\right)=0} ||\boldsymbol{\Gamma}^{(i)}_{(G)BP,G_j}||_2\Big), \tag{2.5}$$

where i is the sample index, $||\boldsymbol{\Gamma}^{(i)}_{(G)BP,G_j}||_2$ is the ℓ_2 norm of group j within $\boldsymbol{\Gamma}^{(i)}_{(G)BP}$ (i.e., an element of $\boldsymbol{\Gamma}^{(i)}_{PGBP}$) and ϕ_γ is an appropriate proximal operator. Intuitively, this encourages a better separation between active and inactive groups. For the BP case we applied group size 1. We ended up leaving this term out in the second phase of our runs.

For the training of the feedfoward networks, we applied mean squared error against $\boldsymbol{\Gamma}$ (which we saved from our PGBP runs).

$$\sum_{i=1}^{db_{size}} (\boldsymbol{\Gamma_i} - \hat{\boldsymbol{\Gamma}}_{\boldsymbol{i}})^2 \tag{2.6}$$

where we denoted the feedforward network approximations with $\hat{\boldsymbol{\Gamma}}$ and the size of the database with $\boldsymbol{db_{size}}$.

For multiclass classification on MNIST the Categorical Cross Entropy Loss was used:

$$-\sum_{c=1}^{10} y_c \log(p_c) \tag{2.7}$$

where $\boldsymbol{y_c}$ is a binary indicator (0 or 1) if class label $\boldsymbol{c}$ is the correct classification for the observation, and $\boldsymbol{p}$ is the predicted probability of the classifier that the observation is of

class $\boldsymbol{c}$. For binary classification on the synthetic databases we used the Hinge Loss

$$max(0, 1 - y \cdot \hat{y}). \tag{2.8}$$

where $\boldsymbol{y}$ is the binary output and $\hat{\boldsymbol{y}}$ is the binary target.

In some cases we pretrained the dictionaries, meaning that the training process consisted of a phase were classification loss was excluded from the function, followed by a phase where it was included. In the second phase we also tried cross-initializing dictionaries, meaning that we trained a dictionary with one architecture and used that dictionary as a starting point for another architecture. Initially we used Stochastic Gradient Descent (SGD) [37] as an optimizer, but later switched to Adam [38]. We trained for a maximum of 500 epochs, and used early stopping with a patience of 10. We also introduced a warm-up phase, meaning that the $\boldsymbol{\gamma}$ terms in the thresholding functions started from being set to zero (yielding no sparsification), and were gradually increased to their maximum value by the end of the warm-up phase. In later sections we explain how these different parameters influenced robustness and accuracy.

2.3 Adversarial Attacks

For the adversarial attacks we needed to generate perturbed inputs for the networks, these were calculated using the Iterative Fast Gradient Sign Method (IFGSM) [39]. The algorithms starts from $\boldsymbol{X}$ and gets to $\boldsymbol{Y}_T$ in T bounded steps. The steps are taken with respect to ℓ_∞ and ℓ_2 norms according to the sign of the gradient of the total loss.

Algorithm 6 The Iterative Fast Gradient Sign Method algorithm

Funct IFGSM($J, X, D, class(X), T, \epsilon$)

$\boldsymbol{Y}_0 := \boldsymbol{X}$
$a := \frac{\epsilon}{T}$
$t := 0$
while t<T **do**
    $\boldsymbol{G}_{t-1} := \nabla_{\boldsymbol{Y}_{t-1}} J\left(\boldsymbol{D}, \boldsymbol{Y}_{t-1}, class(\boldsymbol{X})\right)$
    $\boldsymbol{Y}_t := clamp\left(\boldsymbol{Y}_{t-1} + a \cdot (\boldsymbol{G}_{t-1})\right)$
    $t := t + 1$
end while
return $\boldsymbol{Y}_T$

Here we set the learning rate as $a = \frac{\epsilon}{T}$ and $clamp$ is a clipping function. In almost all cases the attack was white box, meaning that the perturbed images were calculated for the

networks with respect to their own loss functions and inner structures. The aim of using the feedforward networks however was to see if they could speed up the predictions relying on the representations found by PGBP, so in these cases we used the perturbed images calculated with respect to the loss functions and structures of the corresponding PGBP network, and used these to test the robustness, resulting in a black box attack. Figure 2.1 shows examples of how IFGSM distorts MNIST images.

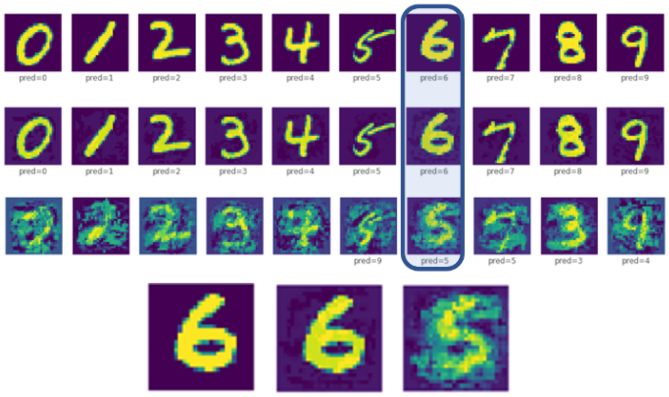

Figure 2.1: IFGSM on MNIST. In the top row we show noiseless images, and as we go down we show them with more and more noise. In the bottom row we show a slightly enlarged example, where it visible that IFGSM not only adds noise, but tries to change the digit into another one (i.e six to five). Note that these images are only shown for visualisation, and were calculated with much higher ϵ values than what we used in our experiments.

2.4 Datasets

For the first phase we generated two synthetic databases, one with group pooling and one without. We trained and tested our models on MNIST as well. In the second phase we only used MNIST.

2.4.1 Synthetic Data

For both datasets generations we did the following procedure: We started with building a *dictionary* $\boldsymbol{D} \in \mathbb{R}^{100\times300}$ using normalized Grassmannian packing with 75 groups of size 4 [40]. Next, we generated two normalized random classifiers $\boldsymbol{w} \in \mathbb{R}^{300}$ and $\boldsymbol{w}_{PGBP} \in \mathbb{R}^{75}$ with components drawn from the normal distribution $\mathcal{N}(0,1)$ and set the bias term to zero (we did not train classifiers in the synthetic case, we used the true parameters). Then, we created the respective input sets. We randomly generated $\boldsymbol{\Gamma} \in \mathbb{R}^{300}$ vectors having 8

nonzero groups of size 4 with activations drawn uniformly from the interval $[1, 2]$ and computed $\boldsymbol{X} = \boldsymbol{D}\boldsymbol{\Gamma}$. We collected two sets of 10000 $\boldsymbol{X}$ vectors that satisfied classification margin $O(\boldsymbol{X}) \geq \eta \in \{0.03, 0.1, 0.3\}$ in terms of the classifiers $\boldsymbol{w}$ and $\boldsymbol{w}_{PGBP}$ acting on top of $\boldsymbol{\Gamma}$ (no pooling) and the ℓ_2 norms of the groups of $\boldsymbol{\Gamma}$ (pooled), respectively.

2.4.2 MNIST Data

We employed image classification on the MNIST dataset. The images were vectorized and we preprocessed to zero mean and unit variance. We used fully connected (dense) dictionaries of different sizes, with optional groups of size fixed at 8 for our grouped methods (meaning that the number of groups changed depending on the size of the dictionary), and a fully connected softmax classifier $\boldsymbol{w}$ mapping to the 10 class probabilities acting either on top of the full $\boldsymbol{\Gamma}_{(G)BP}$ (i.e., $\boldsymbol{w}_i \in \mathbb{R}^{D_{size}}$, $i = 1, \ldots, 10$) or the compressed $\boldsymbol{\Gamma}_{PGBP}$ (i.e., $\boldsymbol{w}_{PGBP,\, i} \in \mathbb{R}^{D_{size/8}}$, $i = 1, \ldots, 10$). In this case the true parameters ($\boldsymbol{D}$, $\boldsymbol{w}$, $\boldsymbol{b}$) were not known, so these had to be learned via backpropagation over the training set.

Chapter 3

Results

In the following we present our experimental results. The focus on the synthetic databases was to compare group based methods with non-group based methods, and to test if feedforward networks can learn the representations well enough to speed up prediction, while still achieving a robust model. MNIST was used for the same comparisons, but the deepening of the networks was also something that we looked into. We separate the next two sections by the databases.

3.1 Synthetic Data

We used three margins, 0.3, 0.1, and 0.03 on the synthetic data. Results of the experiments are shown in Fig. 3.1, Fig. 3.2 and Fig. 3.3, respectively.

We found that BP achieves lower accuracy even without attacks, meaning that it did not reach 100% in any of our experiments, and it breaks down rapidly as ϵ increases. GBP however, manages to achieve perfect scores with higher margins without attacks, having access to the ground truth group structure of the data, seems to make this possible. After attacking the input, GBP breaks down slower than BP. It is important to note, that the search space is much larger for BP than for GBP, making the task easier, and it is much harder to find the solution for BP since the data was generated by groups. The opposite is also true: BP generated synthetic data may be impossible to solve by the GBP method.

Comparisons between BP and GBP were done separately from PGBP since data generation procedures differ in the two cases. We worked with the PGBP network and its feedforward approximations, the shallow and deep networks, and the linear transformer. The feedforward networks estimated the (pooled) sparse activities of the PGBP outputs

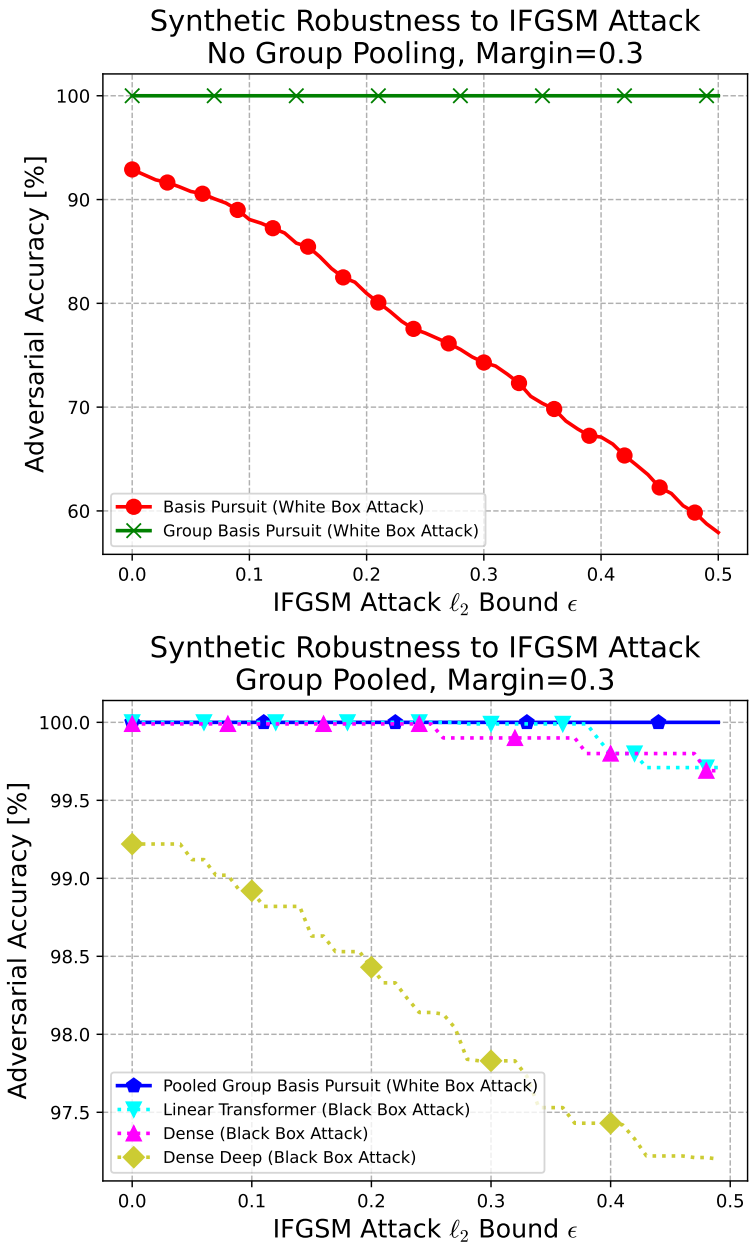

Figure 3.1: Results for adversarial robustness against IFGSM attack, on the synthetic dataset of margin 0.3. **(a):** Our Group Basis Pursuit (GBP, green) obtains 100% accuracy for all ϵ while Basis Pursuit (BP, red) does not reach it even without an attack, and continues to detoriate as ϵ increases. **(b):** PGBP (blue) achieves perfect scores for all ϵ values. Performance of LT (cyan) and the Dense (magenta) networks trained on PGBP groups indices starts to deteriorate for large ϵ values although it is under Black Box attack. Deep network (yellow) having parameter count similar to LT is overfitting.

processed by the classifier vector $\boldsymbol{w}$. As mentioned before, the dense, deep dense and transformer networks were not trained to approximate the ground truth, but the results of PGBP, so they reach lower accuracy with no attacks. In all cases, beyond a certain ϵ the breakdown is faster for PGBP than for the other methods. We note that due to the different nature of the attack (white box for PGBP and black box for the others), the task becomes

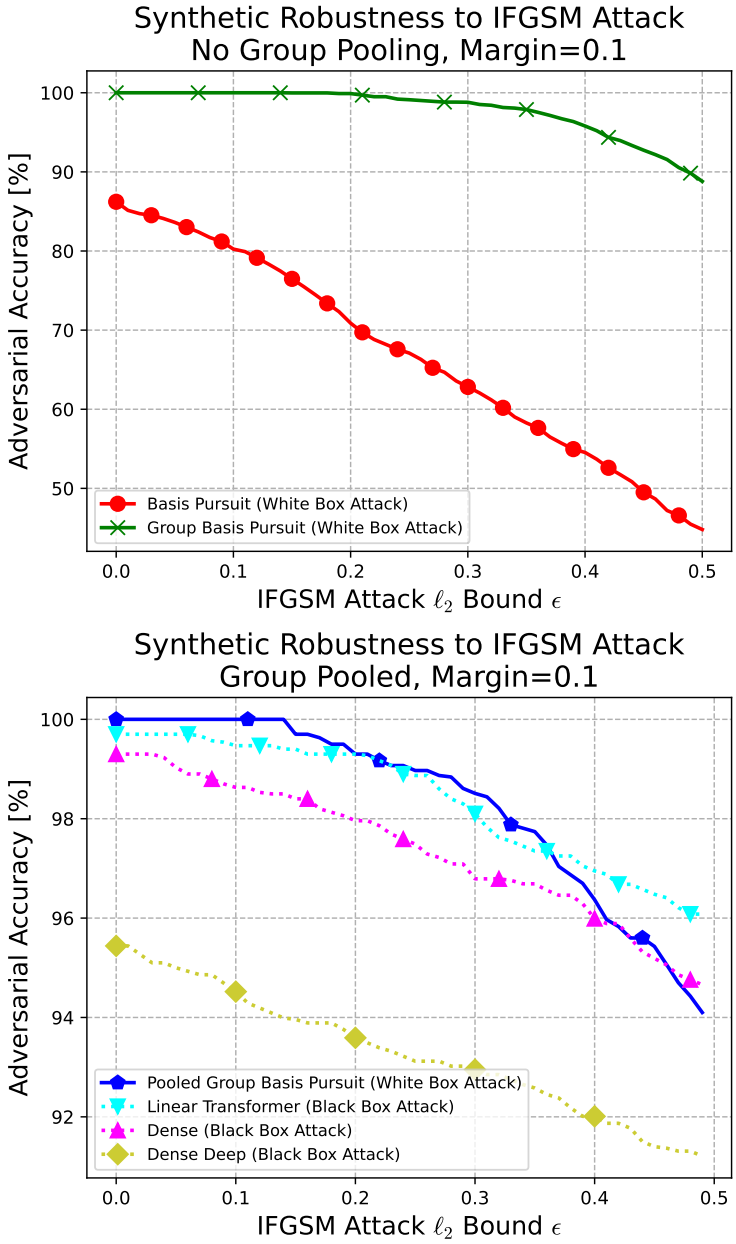

Figure 3.2: Results for adversarial robustness against IFGSM attack, on the synthetic dataset of margin 0.1. **(a):** Our Group Basis Pursuit (GBP, green) obtains 100% accuracy for small ϵ and considerably outperforms Basis Pursuit (BP, red) as it can exploit the given group structure. **(b):** Pooled Group Basis Pursuit (PGBP, blue) achieves perfect scores for small ϵ. Break down is faster than for the Linear Transformer (LT, cyan) and the Dense (magenta) networks due to the difference between white box and black box attacks. Deep network (yellow) having parameter count similar to LT is overfitting.

easier for the non-PGBP methods. We also found that the larger the margin between the classes, the more significant the BP advantage is in white box adversarial attacks over transformer, dense and deep dense networks. Out of the three feedforward estimations, the transformer performed the best for all margins. The transformer architecture is slightly slower than the other two (even with the linearized attention [36]), but it still significantly

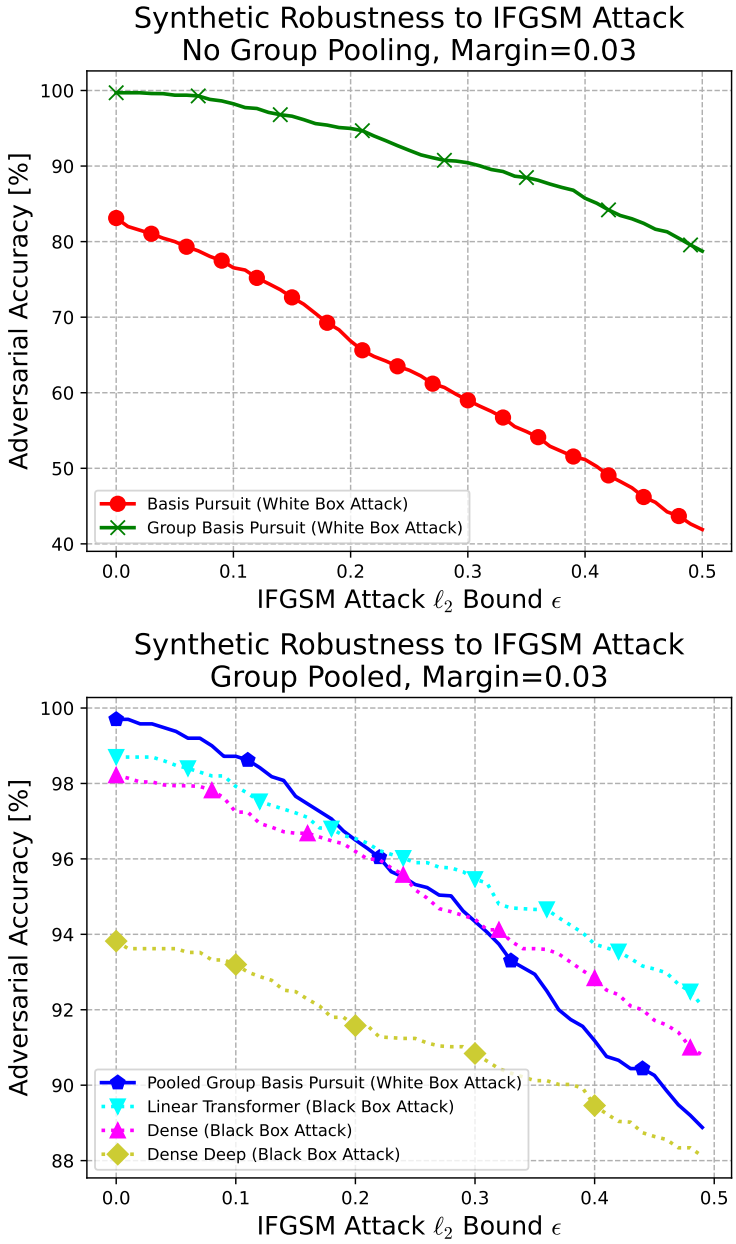

Figure 3.3: Results for adversarial robustness against IFGSM attack, on the synthetic dataset of margin 0.03. **(a):** GBP (green) obtains 100% with no atacks but starts breaking down at small attacks. BP (red) declines steeply. **(b):** PGBP (blue) can not achieve perfect scores for ϵ values, including the no attack case and breaks down faster than LT and the Dense network for large ϵ's. Deep network is overfitting.

faster than BP approximation.

Since the ground truths were known, we also calculated statistics of how well our models performed. See Table 3.1. PGBP and GBP are the same from the point of view of these statistics, as only their downstream classifiers differ. In all cases these two performed the best. GBP and PGBP have access to the true group structure, while BP only picks individual units. The best result in finding the *exact* group combinations is below 50%,

which is very weak. Accuracy, i.e., mean of the correctly found active (True Positive) and inactive (True Negative) groups out of all groups (Positive + Negative) is almost perfect for GBP and PGBP. As expected, performance of the feedforward networks are weaker since they are not approximating the ground truth, meaning that even in the best case, they could have only reached what the pursuit methods already did (with regards to these metrics). The Linear Transformer outperforms the Dense networks. The Dense Shallow network is better than the Deep network, since the latter is overfitted.

Method	**Inactive Groups**	**Mean Group Accuracy**	**Found Group Combinations**
BP	47.6%	58.3%	0.0%
GBP	**87.8**%	**98.5**%	**48.8**%
PGBP	**87.8**%	**98.5**%	**48.8**%
Transformer	85.6%	96.5%	9.5%
Dense Shallow	83.2%	93.8%	1.4%
Dense Deep	74.3%	84.5%	0.0%

Table 3.1: Statistics for the attack-free case of the Synthetic dataset. $O(X) \geq 0.1$ margin.

3.2 MNIST

On MNIST, we tried more architectures. Overall we used Basis Pursuit, Group Basis Pursuit, Pooled Group Basis Pursuit, Layered Basis Pursuit, Layered Group Basis Pursuit, Layered Pooled Group Basis Pursuit, Deep Pursuit, Deep Group Pursuit, Deep Pooled Group Basis Pursuit, and the 3 feedforward networks: Linear Transformer, Shallow Dense network and Deep Dense network. In the first phase we compared the 1 layered Basis Pursuit methods, and the FF networks, with the same goals as with the synthetic case: to test the group based methods, and to improve the methods with regards to their slowness. In the second phase we tested if deepening pursuit methods further improves the results with regards to accuracy and robustness. We also compared soft thresholding with hard thresholding.

Among the white box attacked single layer pursuit methods, PGBP gave the best results for both the non-attacked and for the attacked case, indicating the benefits of the pooled representation, i.e., it is more difficult to attack group norms than the elements within groups BP and GBP were worse and their curves crossed each other. We believe that the advantages and limitations of PGBP deserves further testing on diverse datasets.

We introduced the cost term (Eq. (2.2)) during training of our single layered networks and found that in two out of three cases, it helped. We could improve the results slightly for BP and somewhat more for the PGBP case, but not for the GBP case.

Feedforward nets were attacked by the black box method. The Linear Transformer obtained the best results. Deep Network was difficult to teach; it was overfitting. We now turn to the results of our 3 layered pursuit methods.

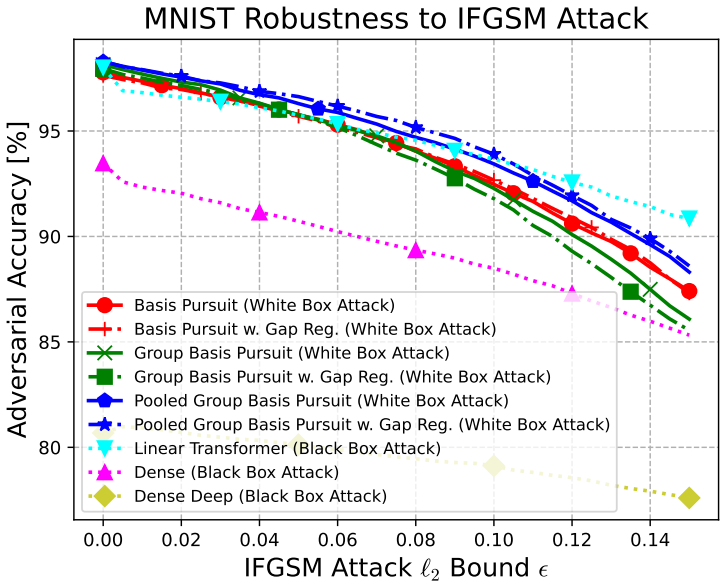

Figure 3.4: Results for adversarial robustness against IFGSM attack, on MNIST with shallow Basis Pursuit networks and feedforward networks.

We first tried training all our architectures with randomly initialized dictionaries. Then we tried "cross-initializing" them, meaning that we initialized the LBP architectures with the dictionaries we obtained by training the corresponding DP architectures (e.g. the LGBP with the GDP dictionary). We did this the other way around as well. We did not come to conclusive results with this, in some cases having an already trained dictionary helped when we tried to train it further, but in other cases we ended up with worse results than with a randomly initialized dictionary. In all cases we only visualized the better results we got with the better dictionaries. As some of the architectures drop significantly when attacked, the comparisons become hard between our figures, see appendix B for the exact numerical results of the following runs.

First we compared our methods using soft thresholding. Among the LBP methods in the non-attacked case the pooled group method performed the best, but at larger $\boldsymbol{\epsilon}$ values the group method overcame it, and it was dropping less steeply. In the $\boldsymbol{\epsilon} = 0$ case the non-group LBP performed well, but started breaking down very fast. The DP methods

performed better in all cases. With no attack the non-group architecture performed the best (with PGDP coming very close). With high ϵ values PGDP overcomes all other methods. The initial accuracy of GDP was the weakest out of the three methods, but it seems to handle the attacks better.

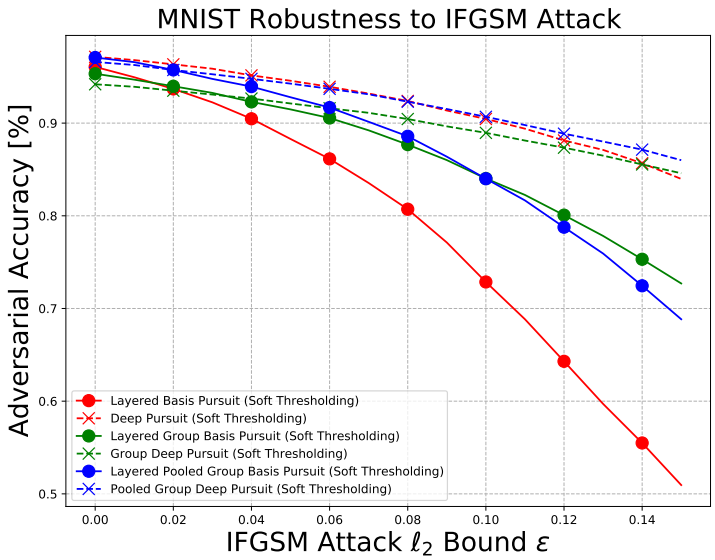

Figure 3.5: Results for adversarial robustness against IFGSM attack, on MNIST with 3 Layered Basis Pursuit networks and 3 Layered Deep Pursuit networks using soft thresholding.

Next we compared the hard thresholding methods. Both with LBP and DP it was much harder to tune the λ values in the hard thresholding function, with most of our runs we either achieved low accuracy or dense dictionaries. In contrast to the soft thresholding LBP case, the non-grouped versions performed the best both with and without attack. LGBP and LPGBP performed worse and broke down very fast when attacked, with LGBP's accuracy dropping the most. The non-grouped DP and LDP follows a very similar curve, and PGDP is almost parallel to these two, but achieves a lower accuracy throughout the attacks.

When it comes to the non-attacked classification accuracies, soft thresholding architectures perform better than hard tresholding architectures (with the only exception being GDP). It seems however, that in the pooled group cases the hard thresholding architectures drop slightly less when attacked, but since they are much harder to train, and their initial accuracy's are low, they are not able to overcome the soft thresholding architectures (even with high ϵ values).

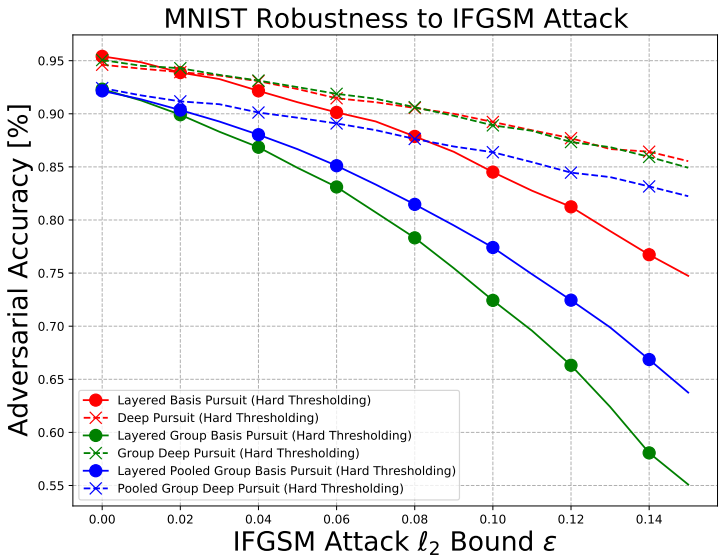

Figure 3.6: Results for adversarial robustness against IFGSM attack, on MNIST with 3 Layered Basis Pursuit networks and 3 Layered Deep Pursuit networks using hard thresholding.

As expected [41], with regards to robustness Deep Pursuit performed better than Layered Basis Pursuit in all cases. Surprisingly, with no attacks LBP managed to perform better in some cases (e.g. LGBP versus GDP with soft thresholding). With our runs we managed to substantiate the superiority of Deep Pursuit, however, we unfortunately only proved this compared to Layered Basis Pursuit. When we compare DP with our shallow BP methods, it does not bring any improvements. Our hard thresholding DP methods all perform worse, and even with soft thresholding the initial accuracy of GDP is lower than the initial accuracy of GBP. DP and PGDP perform similarly to BP and PGBP.

Chapter 4

Discussion

In our research we have dealt with structurally extending basis pursuit methods. We tested group based methods. We also experimented with feedforward estimations, which are significantly faster and our results indicate that they are relatively accurate with the estimations. Our Linear Transformer based architecture stood out as the most accurate and the most robust as well. These feedforward estimations were tested with black box attacks, they could be more fragile if a white box attack would be applied. In case of attacks, the transformer shows reasonable robustness against black box attacks [42]. It is important to mention that there is a vast literature on how to detect attacks [43, 44]. These methods could optimize the speed if all pursuit and feedforward methods estimating networks are run in parallel.

In some cases performance could be improved by introducing additional regularization loss terms [45]. We added a loss term aiming to increase the gap between the groups that will become active and the groups that will be inactive after thresholding. It did not improve the results in all cases, this could be because the loss term (Eq. (2.2)) may have been too strict. This could be improved upon in future works. Later we introduced the minimization of the mutual coherence of the dictionaries (Eq. (2.3) and Eq. (2.4)) in the loss term [46]. We cannot conclude if it helps with training robust networks, it did not show obvious advantages in our experiments. Deepening our architectures did not seem to improve the results either, however they have not been tested without the mutual coherence minimization term. Perhaps DP could be tested on a more complex database.

In our experiments, we restricted the investigations to groups of the same size and the same bias, but it is possible that inputs may be best fit by groups of different sizes and biases. This is an architecture optimization problem, where the solution is unknown.

According to [47] however, learning of the sparse representation is promising since under rather strict conditions, high-quality sparse dictionaries can be found. It is still desired to search for groups since the search space may become smaller by the groups and the presence of the active groups may be estimated quickly and accurately using feedforward methods, especially transformers (in the absence of attacks). Feedforward estimation of the groups followed by pursuit methods with different group sizes including single atoms seems worth studying as well.

Chapter 5

Conclusion

We studied the adversarial robustness of sparse coding by means of numerical computations. Sparse coding is a promising solution and it offers rich structural generalizations. The varieties include groups within layers, diverse connectivities between the layers and versions of optimization losses connected to the ℓ_1 norm. In some cases Group Basis Pursuit (GBP) demonstrated that it can outperform Basis Pursuit (BP), and that Pooled Group Basis Pursuit (PGBP) works better than both using an 8 times smaller representation. This calls for further investigations. We found that PGBP offers fast feedforward estimations and the transformer version shows considerable robustness for the datasets we studied. Finally, we showed that adding a loss term that aims to maximize the gap between the groups that remain active and those that become inactive after soft thresholding can improve robustness even further.

Yet, the scope of our studies is limited from multiple perspectives. We have seen that PGBP performed surprisingly great in our single layered cases, however it did not show an obvious superiority with more layers. This calls for further investigations and if possible, for theorems. Comparing Deep Pursuit (DP) and Layered Basis Pursuit (LBP) showed that accuracy-wise they are relatively close to each other, even though they rely on different solutions (LBP is concentrating on giving the best reconstruction of the original image at the first step, while DP is trying to reconstruct all of it's layers equally well). As expected, DP managed to prevent error accumulation, and in the longer run the corresponding accuracies dropped significantly less, compared to LBP. The optimization of the architecture to different group sizes and the optimization of the biases and the losses may provide performance improvements.

Defenses against noise, novelties, anomalies and, in particular, against adversarial

attacks may be solved by combining our robust, structured sparse networks with out-of-distribution detection methods. Attack detection was out of scope of our paper, and such extensions should be valuable. In certain situations attacks need to be detected and the slower basis pursuit method should be invoked if needed.

Acknowledgements

The research was supported by (a) the Ministry of Innovation and Technology NRDI Office within the framework of the Artificial Intelligence National Laboratory Program and (b) Project no. TKP2021-NVA-29, which has been implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NVA funding scheme. We thank our supervisors - András Lőrincz and Dávid Szeghy - and Zoltán Ádám Milacski, for helping us with understanding the theoretical background, narrowing down our research focus and evaluating our results. We also thank Áron Fóthi and Ellák Somfai for their contributions to the codebase that we relied on for our experiments.

Appendix A

Architectures

A.1 Synthetic Experiment

Basis Pursuit

1. Basis Pursuit Layer 300 units
2. Dense Linear Layer 1 unit

Linear Transformer

1. Attention Layer 100 units
2. Dense Linear Layer 300 units
3. Batch Normalization Layer
4. Rectified Linear Unit (ReLU) Activation
5. Dense Linear Layer 1 unit

Dense Network

1. Dense Linear Layer 300 units
2. Batch Normalization Layer
3. Rectified Linear Unit (ReLU) Activation
4. Dense Linear Layer 1 unit

Dense Deep Network

1. Dense Linear Layer 128 units
2. Rectified Linear Unit (ReLU) Activation

3. Dense Linear Layer 157 units
4. Rectified Linear Unit (ReLU) Activation
5. Dense Linear Layer 185 units
6. Rectified Linear Unit (ReLU) Activation
7. Dense Linear Layer 214 units
8. Rectified Linear Unit (ReLU) Activation
9. Dense Linear Layer 242 units
10. Rectified Linear Unit (ReLU) Activation
11. Dense Linear Layer 271 units
12. Rectified Linear Unit (ReLU) Activation
13. Dense Linear Layer 300 units
14. Batch Normalization Layer
15. Rectified Linear Unit (ReLU) Activation
16. Dense Linear Layer 1 unit

Basis Pursuit with Pooled Groups

1. Basis Pursuit Layer 75 units
2. Dense Linear Layer 1 unit

Linear Transformer with Pooled Groups

1. Attention Layer 100 units
2. Dense Linear Layer 75 units
3. Batch Normalization Layer
4. Rectified Linear Unit (ReLU) Activation
5. Dense Linear Layer 1 unit

Dense Network with Pooled Groups

1. Dense Linear Layer 75 units
2. Batch Normalization Layer
3. Rectified Linear Unit (ReLU) Activation

4. Dense Linear Layer 1 unit

Dense Deep Network with Pooled Groups

1. Dense Linear Layer 96 units
2. Rectified Linear Unit (ReLU) Activation
3. Dense Linear Layer 92 units
4. Rectified Linear Unit (ReLU) Activation
5. Dense Linear Layer 89 units
6. Rectified Linear Unit (ReLU) Activation
7. Dense Linear Layer 85 units
8. Rectified Linear Unit (ReLU) Activation
9. Dense Linear Layer 82 units
10. Rectified Linear Unit (ReLU) Activation
11. Dense Linear Layer 78 units
12. Rectified Linear Unit (ReLU) Activation
13. Dense Linear Layer 75 units
14. Batch Normalization Layer
15. Rectified Linear Unit (ReLU) Activation
16. Dense Linear Layer 1 unit

A.2 MNIST Experiment

Basis Pursuit

1. Basis Pursuit Layer 256 units
2. Dense Linear Layer 10 units
3. Softmax Activation

Layered Basis Pursuit

1. Basis Pursuit Layer 128 units
2. Basis Pursuit Layer 256 units

3. Basis Pursuit Layer 4096 units
4. Dense Linear Layer 10 units
5. Softmax Activation

Deep Pursuit

1. Deep Pursuit Layer 128 units
2. Deep Pursuit Layer 256 units
3. Deep Pursuit Layer 4096 units
4. Deep Linear Layer 10 units
5. Softmax Activation

Linear Transformer

1. Attention Layer 784 units
2. Batch Normalization Layer
3. Dense Linear Layer 256 units
4. Rectified Linear Unit (ReLU) Activation
5. Dense Linear Layer 10 units
6. Softmax Activation

Dense Network

1. Dense Linear Layer 256 units
2. Batch Normalization Layer
3. Rectified Linear Unit (ReLU) Activation
4. Dense Linear Layer 10 units
5. Softmax Activation

Dense Deep Network

1. Dense Linear Layer 708 units
2. Rectified Linear Unit (ReLU) Activation
3. Dense Linear Layer 633 units
4. Rectified Linear Unit (ReLU) Activation

5. Dense Linear Layer 557 units
6. Rectified Linear Unit (ReLU) Activation
7. Dense Linear Layer 482 units
8. Rectified Linear Unit (ReLU) Activation
9. Dense Linear Layer 406 units
10. Rectified Linear Unit (ReLU) Activation
11. Dense Linear Layer 331 units
12. Rectified Linear Unit (ReLU) Activation
13. Dense Linear Layer 256 units
14. Batch Normalization Layer
15. Rectified Linear Unit (ReLU) Activation
16. Dense Linear Layer 10 unit
17. Softmax Activation

Basis Pursuit with Pooled Groups

1. Basis Pursuit Layer 32 units
2. Dense Linear Layer 10 units
3. Softmax Activation

Layered Basis Pursuit with Pooled Groups

1. Basis Pursuit Layer 128 units
2. Basis Pursuit Layer 256 units
3. Basis Pursuit Layer 512 units
4. Dense Linear Layer 10 units
5. Softmax Activation

Deep Pursuit with Pooled Groups

1. Deep Pursuit Layer 128 units
2. Deep Pursuit Layer 256 units
3. Deep Pursuit Layer 512 units

4. Deep Linear Layer 10 units
5. Softmax Activation

Linear Transformer with Pooled Groups

1. Attention Layer 784 units
2. Batch Normalization Layer
3. Dense Linear Layer 32 units
4. Rectified Linear Unit (ReLU) Activation
5. Dense Linear Layer 10 units
6. Softmax Activation

Dense Network with Pooled Groups

1. Dense Linear Layer 32 units
2. Batch Normalization Layer
3. Rectified Linear Unit (ReLU) Activation
4. Dense Linear Layer 10 units
5. Softmax Activation

Dense Deep Network with Pooled Groups

1. Dense Linear Layer 676 units
2. Rectified Linear Unit (ReLU) Activation
3. Dense Linear Layer 569 units
4. Rectified Linear Unit (ReLU) Activation
5. Dense Linear Layer 461 units
6. Rectified Linear Unit (ReLU) Activation
7. Dense Linear Layer 354 units
8. Rectified Linear Unit (ReLU) Activation
9. Dense Linear Layer 246 units
10. Rectified Linear Unit (ReLU) Activation
11. Dense Linear Layer 139 units

12. Rectified Linear Unit (ReLU) Activation
13. Dense Linear Layer 32 units
14. Batch Normalization Layer
15. Rectified Linear Unit (ReLU) Activation
16. Dense Linear Layer 10 unit
17. Softmax Activation

Appendix B

Results of LBP and DP attacks

	LBP	**LGBP**	**LPGBP**	**DP**	**GDP**	**PGDP**
ϵ=0	96.1%	95.3%	97.1%	**97.2**%	94.2%	96.6%
ϵ=0.15	51.0%	72.7%	68.8%	84.0%	84.6%	**86**%

Table B.1: Classification accuracy of architectures using soft thresholding, after IFGSM attack with ϵ value shown in the first column

	LBP	**LGBP**	**LPGBP**	**DP**	**GDP**	**PGDP**
ϵ=0	**95.4**%	92.3%	92.2%	94.6%	95.1%	92.4%
ϵ=0.15	74.7%	55.1%	63.8%	**85.5**%	84.9%	82.2%

Table B.2: Classification accuracy of architectures using hard thresholding, after IFGSM attack with ϵ value shown in the first column

Bibliograhpy

[1] Vardan Papyan, Jeremias Sulam, and Michael Elad. “Working Locally Thinking Globally: Theoretical Guarantees for Convolutional Sparse Coding”. In: *IEEE Transactions on Signal Processing* 65.21 (2017), pp. 5687–5701. DOI: 10.1109/TSP.2017.2733447.

[2] Szeghy Dávid, Zoltán Ádám Milacski, Áron Fóthi, and András Lőrincz. *Adversarial Perturbation Stability of the Layered Group Basis Pursuit*. English. Conference on Mathematics of Machine Learning, August 04 -07, 2021, Center for Interdisciplinary Research (ZiF), Bielefeld. 2021. URL: https://m2.mtmt.hu/api/publication/32153563.

[3] Dávid Szeghy., Mahmoud Aslan., Áron Fóthi., Balázs Mészáros., Zoltán Milacski., and András Lőrincz. “Structural Extensions of Basis Pursuit: Guarantees on Adversarial Robustness”. In: *Proceedings of the 3rd International Conference on Deep Learning Theory and Applications - DeLTA,* INSTICC. SciTePress, 2022, pp. 77–85. ISBN: 978-989-758-584-5. DOI: 10.5220/0011138900003277.

[4] I J Goodfellow, J Shlens, and C Szegedy. “Explaining and harnessing adversarial examples”. In: *arXiv:1412.6572* (2014).

[5] Michael Elad, Mário A. T. Figueiredo, and Yi Ma. “On the Role of Sparse and Redundant Representations in Image Processing”. In: *Proceedings of the IEEE* 98.6 (2010), pp. 972–982. DOI: 10.1109/JPROC.2009.2037655.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[7] Bruno A Olshausen and David J Field. “Emergence of simple-cell receptive field properties by learning a sparse code for natural images”. In: *Nature* 381.6583 (1996), pp. 607–609.

[8] M Fadili, Jean-Luc Starck, J. Bobin, and Yassir Moudden. “Image Decomposition and Separation Using Sparse Representations: An Overview”. In: *Proceedings of the IEEE* 98 (July 2010), pp. 983 –994. DOI: 10.1109/JPROC.2009.2024776.

[9] Mark Plumbley, Thomas Blumensath, Laurent Daudet, Remi Gribonval, and Mike Davies. “Sparse Representations in Audio and Music: From Coding to Source Separation”. In: *Proceedings of the IEEE* 98 (June 2010), pp. 995 –1005. DOI: 10.1109/JPROC.2009.2030345.

[10] Samer Abdallah and Mark Plumbley. “Unsupervised Analysis of Polyphonic Music by Sparse Coding”. In: *IEEE Transactions on Neural Networks* 17 (Jan. 2006), pp. 179–96. DOI: 10.1109/TNN.2005.861031.

[11] Christian Austin, Emre Ertin, and Randolph Moses. “Sparse Signal Methods for 3-D Radar Imaging”. In: *Selected Topics in Signal Processing, IEEE Journal of* 5 (July 2011), pp. 408 –423. DOI: 10.1109/JSTSP.2010.2090128.

[12] Jean-Luc Starck and Jerome Bobin. *Astronomical Data Analysis and Sparsity: from Wavelets to Compressed Sensing*. 2009. DOI: 10.48550/ARXIV.0903.3383. URL: https://arxiv.org/abs/0903.3383.

[13] Balas Kausik Natarajan. “Sparse approximate solutions to linear systems”. In: *SIAM journal on computing* 24.2 (1995), pp. 227–234.

[14] David L Donoho and Michael Elad. “Optimally sparse representation in general (nonorthogonal) dictionaries via 1 minimization”. In: *Proceedings of the National Academy of Sciences* 100.5 (2003), pp. 2197–2202.

[15] Thomas Strohmer and Robert W Heath Jr. “Grassmannian frames with applications to coding and communication”. In: *Applied and computational harmonic analysis* 14.3 (2003), pp. 257–275.

[16] Emmanuel J Candes and Terence Tao. “Decoding by linear programming”. In: *IEEE transactions on information theory* 51.12 (2005), pp. 4203–4215.

[17] Ingrid Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, 1992. DOI: 10.1137/1.9781611970104. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611970104. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611970104.

[18] Stéphane G Mallat and Zhifeng Zhang. "Matching pursuits with time-frequency dictionaries". In: *IEEE Transactions on signal processing* 41.12 (1993), pp. 3397–3415.

[19] Michael Elad. *Sparse and redundant representations: from theory to applications in signal and image processing*. Vol. 2. 1. Springer, 2010.

[20] Stéphane Mallat and Zhifeng Zhang. "Matching pursuits with time-frequency dictionaries". In: *IEEE Trans. Signal Process.* 41 (1993), pp. 3397–3415.

[21] Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad. "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition". In: *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*. 1993, 40–44 vol.1. DOI: 10.1109/ACSSC.1993.342465.

[22] Shaobing Chen and D. Donoho. "Basis pursuit". In: *Proceedings of 1994 28th Asilomar Conference on Signals, Systems and Computers*. Vol. 1. 1994, 41–44 vol.1. DOI: 10.1109/ACSSC.1994.471413.

[23] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. "Atomic decomposition by basis pursuit". In: *SIAM review* 43.1 (2001), pp. 129–159.

[24] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[25] Amir Beck and Marc Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM journal on imaging sciences* 2.1 (2009), pp. 183–202.

[26] Ingrid Daubechies, Michel Defrise, and Christine De Mol. "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint". In: *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences* 57.11 (2004), pp. 1413–1457.

[27] Joel A Tropp. "Greed is good: Algorithmic results for sparse approximation". In: *IEEE Transactions on Information theory* 50.10 (2004), pp. 2231–2242.

[28] David L Donoho and Michael Elad. "On the stability of the basis pursuit in the presence of noise". In: *Signal Processing* 86.3 (2006), pp. 511–532.

[29] Hilton Bristow and Simon Lucey. "Optimization methods for convolutional sparse coding". In: *arXiv preprint arXiv:1406.2407* (2014).

[30] V Papyan, J Sulam, and M Elad. “Working Locally Thinking Globally: Theoretical Guarantees for Convolutional Sparse Coding”. In: *IEEE Trans. Signal Process.* 65.21 (2017), pp. 5687–5701.

[31] Vardan Papyan, Yaniv Romano, and Michael Elad. “Convolutional neural networks analyzed via convolutional sparse coding”. In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 2887–2938.

[32] Karol Gregor and Yann LeCun. “Learning fast approximations of sparse coding”. In: *Proceedings of the 27th international conference on international conference on machine learning*. 2010, pp. 399–406.

[33] Yaniv Romano, Aviad Aberdam, Jeremias Sulam, and Michael Elad. “Adversarial noise attacks of deep learning architectures: Stability analysis via sparse-modeled signals”. In: *Journal of Mathematical Imaging and Vision* 62.3 (2020), pp. 313–327.

[34] George Cazenavette, Calvin Murdock, and Simon Lucey. “Architectural adversarial robustness: The case for deep pursuit”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 7150–7158.

[35] Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. “Structured sparsity through convex optimization”. In: *Statistical Science* 27.4 (2012), pp. 450–468.

[36] A Katharopoulos, A Vyas, N Pappas, and F Fleuret. “Transformers are RNNs: Fast autoregressive transformers with linear attention”. In: *Int. Conf. on Mach. Learn.* PMLR. 2020, pp. 5156–5165.

[37] L Bottou, F E Curtis, and J Nocedal. “Optimization Methods for Large-Scale Machine Learning”. In: *Siam Review* 60.2 (2018), pp. 223–311.

[38] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

[39] A Kurakin, I J Goodfellow, and S Bengio. “Adversarial Examples in the Physical World”. In: *arXiv:1607.02533* (2016). URL: http://arxiv.org/abs/1607.02533.

[40] I S Dhillon, Jr RW Heath, T Strohmer, and J A Tropp. “Constructing Packings in Grassmannian Manifolds via Alternating Projection”. In: *Exp. Math.* 17.1 (2008), pp. 9–35.

[41] G Cazenavette, C Murdock, and S Lucey. “Architectural adversarial robustness: The case for deep pursuit”. In: *IEEE/CVF Conf. on Comp. Vis. and Patt. Recogn.* 2021, pp. 7150–7158.

[42] Y Bai, J Mei, A L Yuille, and C Xie. “Are Transformers more robust than CNNs?” In: *Adv. in Neural Inf. Proc. Syst.* 34 (2021).

[43] N Akhtar, A Mian, N Kardan, and M Shah. “Advances in adversarial attacks and defenses in computer vision: A survey”. In: *IEEE Access* 9 (2021), pp. 155161–155196.

[44] M Salehi, H Mirzaei, D Hendrycks, Y Li, M H Rohban, and M Sabokrou. “A Unified Survey on Anomaly, Novelty, Open-Set, and Out-of-Distribution Detection: Solutions and Future Challenges”. In: *arXiv:2110.14051* (2021).

[45] C Murdock and S Lucey. “Reframing Neural Networks: Deep Structure in Overcomplete Representations”. In: *arXiv:2103.05804* (2021).

[46] C Murdock and S Lucey. “Dataless model selection with the deep frame potential”. In: *IEEE/CVF Conf. on Comp. Vis. and Patt. Recogn.* 2020, pp. 11257–11265.

[47] S Arora, R Ge, T Ma, and A Moitra. “Simple, efficient, and neural algorithms for sparse coding”. In: *Conf. on Learn. Theo.* PMLR. 2015, pp. 113–149.

List of Figures

List of Tables

List of Algorithms