Utrecht University

Mathematics for Industry

in collaboration with ARCNL

Sparsity-inducing bases for compressive sensing

Authors
Nathan van den Berg
Martijn Brouwer
Marcelo Geurts
Galdámez
Robin van der Laag
Sean Tobin

Supervisors
Ksenia Abrashitova
(ARCNL)
Tristan van Leeuwen
(UU)

April 20, 2023

Contents

1 Background **4**
1.1 Sparsity and the LASSO 5

2 Choosing a basis **6**
2.1 Changing basis for the LASSO 6
2.2 Vectorisation and Kronecker products 7

3 Traditional transforms **9**
3.1 Discrete cosine transform 9
3.2 Wavelet transform 10
3.2.1 Theoretical background 10
3.2.2 Discrete Daubechies-4 transform 12
3.3 Results 13

4 Machine learning approach **19**
4.1 Defining the neural networks 19
4.1.1 Matrix reduction using Kronecker product 19
4.1.2 Constructing a loss function 20
4.2 Results 21
4.2.1 General idea 21

5 Choosing the LASSO parameter **23**
5.1 Choosing the parameter 23
5.2 Adjusting the parameter after a change of basis 24

6 Conclusion **26**

A Kronecker Product **29**

B Technical results **30**
B.1 Proofs 30
B.2 Estimates for the norm of a basis transformation 31

C GPSR algorithm **33**
C.1 GPSR algorithm 33
C.1.1 Warm start and time complexity 33

D CIFAR10 results **35**

E Orthogonal Gradient Descent algorithm **37**

Management Summary

The classical limit to light-based microscopic imaging is the refraction limit. It's however possible to compensate for this physical restriction with compressive sensing. In compressive sensing, we attempt to solve an underdetermined system of linear equations, while maximizing the sparsity of the image. Sparsity, however, is not always guaranteed, so we try to find bases in which our images are sparser, so GPSR yields sharper images.

Generally, the transformations we use give worse results or negligible improvements, in comparison to the identity. However, the transformations found via a machine learning approach do give a sharper image.

Abstract

This paper investigates techniques for improving image reconstruction through compressive sensing, which can overcome the physical limitations of classical microscopes. Three traditional transformations were evaluated: the discrete cosine transform (DCT), DAUB4 wavelet transform, and a combination of the two. The DAUB4 wavelet transform produced no usable output, while the DCT and combined transformation did not significantly increase the accuracy of the reconstructed images. The use of sparsity-inducing neural networks to obtain the basis transformation matrix showed promising visual improvements, but no significant increase in reconstruction quality likely due to limitations of the GPSR algorithm. A data training approach to choosing the optimal LASSO parameter λ was shown to fail, but an expression for the optimal transformation for λ was successfully derived after applying a change of basis. The limitations of the optimization algorithms and the λ term may have caused the failure to reproduce the high-frequency behaviour in the sample images. Future work may involve gradient descent algorithms to replace the neural network and expanding the set of test images, as well as analyzing why the optimization algorithms fail to reconstruct the high-frequency behaviour of the images.

Chapter 1

Background

Classical microscopy, where photons are reflected off a surface and produce an image of a small object, is physically limited in its resolution. Broadly speaking, using this method, it is impossible to resolve details smaller than the wavelength of light. This is the so-called *diffraction limit* of resolvable distance, and it was shown by Ernst Abbe to be

$$d = \frac{\lambda}{2n \sin \theta},$$

where λ is the wavelength of light, n is the refractive index of the medium, and θ is the half-angle of the spot to which the light converges [6]. For a modern microscope and green light of around 500nm, this gives a diffraction limit of around $d \approx 250$nm. This is sufficient for most cells (ca. 1µm-100µm), but for objects such as proteins (ca. 3nm - 6nm) and modern transistors (ca. 3nm), it is wholly insufficient.

Higher magnification can be achieved with electron microscopy (which makes use of the much smaller wavelength of electrons, reducing λ), or extending numerical aperture (i.e. increasing $n \sin \theta$). In this report, we investigate a compressed sensing approach, as laid out in [7].

This method uses multimode fibers to produce a set of random speckle patterns. These patterns are shone through a sample, after which the intensity of the light is measured by a single-pixel detector. We represent both the speckle patterns and the sample as flattened n by n images in $\mathbb{R}^{n^2}$. Then the resulting intensity measurement is $y_i = \langle a_i, x \rangle$, where a_i is a speckle pattern and x is the sample. For m speckle patterns, we can summarize this experiment as the matrix-vector product

$$Ax = \begin{bmatrix} \dots & a_1 & \dots \\ & \vdots & \\ \dots & a_m & \dots \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{n^2} \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = y,$$

where $A \in \mathbb{R}^{m \times n^2}$.

Solving $Ax = y$, however, is extremely difficult since $m \ll n^2$ as we cannot make enough distinct and useful speckle patterns. For instance, in [1], $m = 961$, while an image with, say, 128 pixels per side would give $n^2 = 16384$. All in all, we have a highly underdetermined problem.

1.1 Sparsity and the LASSO

To solve underdetermined problems, it's often useful to impose extra conditions on the desired solution. A natural condition in the case of images is *sparsity*.

Definition 1.1.1. Let $x \in \mathbb{R}^r$. We define the ℓ_0-norm of x as its number of nonzero entries:

$$\|x\|_0 := \#\{i \mid x_i \neq 0\}$$

We say a vector x is *sparse* if most of its entries are zero, i.e. if $\|x\|_0$ is small.

Remark 1.1.2. Any vector $v \in \mathbb{R}^r$ is sparse in some basis: if v is nonzero, consider any basis containing v. In this basis, $\|v\|_0 = 1$.

When it comes to images, it is often desirable to require some measure of sparsity, because sparse images can be reconstructed with many fewer samples than pixels [3]. However, the sparsity of images is very basis-dependent, as illustrated in Remark 2.1.1. This highlights the importance of choosing an appropriate basis for the sort of images we are considering, which will be our main goal in Chapter 2 and Chapter 4.

The classical approach for obtaining sparse solutions to an underdetermined linear problem, is by finding the Least Absolute Shrinkage and Selection Operator (LASSO).

Definition 1.1.3. Let $A \in \mathbb{R}^{r \times s}$, where $r \ll s$, and let $y \in \mathbb{R}^r$. The *LASSO*, $x \in \mathbb{R}^s$, is a solution of

$$\arg\min_{\xi \in \mathbb{R}^s} \|A\xi - y\|_2^2 + \lambda \|\xi\|_1, \tag{1.1}$$

where $\|\xi\|_1 = \sum_{i=1}^{s} |\xi_i|$ is the ℓ_1 norm.

It is not immediately obvious why the solution x to this optimisation problem would be sparse. Indeed, this is generally not the case – rather, we expect the solution to be *approximately* sparse. Specifically, we expect many components of x to be small compared to the larger components of x. This is sometimes also called a *high signal-to-noise ratio*. The reason behind this phenomenon emerging from (1.1) is due to the interaction between the ℓ_2-norm and the ℓ_1-norm. See [4, Chapter 3] for details.

In (1.1), λ controls the weight given to the sparsity of x in comparison to the error term $\|Ax - y\|_2^2$. In other words, a large λ will yield a very sparse x with a larger error, while a small λ will yield a less sparse x, but a smaller error.

In this paper we use the Gradient Projection for Sparse Reconstruction (GPSR) algorithm to solve for the LASSO [8, Appendix C.1], it takes A, y and λ and returns x.

Chapter 2

Choosing a basis

Our main focus for this project was finding an appropriate and substantiated choice of basis for the space of possible solutions x for our underdetermined linear problem $Ax = y$, to which we will apply the LASSO. Note that the LASSO is highly basis-dependent: it will give solutions that are approximately sparse only in the basis that we are using. If we want to achieve a high-quality image reconstruction, it is therefore very useful to pick a basis in which we can expect the images that we wish to reconstruct to be highly sparse.

Finding a basis-transformation B such that Bx is sparse for all $x \in \mathbb{R}^r$ is impossible.[1] It is, however, possible to find transformations that give good results for a given subset $S \subset \mathbb{R}^r$. This is useful, since, in imaging, we don't expect to see all possible signals with equal probability.

Say $S \subset \mathbb{R}^{n^2}$ is the set of images we'd like to reconstruct, our objective in the next two chapters will be to find a matrix B with

$$B \text{ invertible, } \|Bx\|_0 \text{ small for most } x \in S. \tag{2.1}$$

In other words, we want to find a basis in which our set of images S is sparse. This is a purposefully vague formulation, but the machine learning approach in Chapter 4 will make a precise quantification of it necessary.

In this chapter, we will examine the practical implementation of a change of basis. Firstly, we will look at the corresponding reformulated LASSO problem, and we will give some results about the interaction between the two main ways of representing a grayscale image: as an $n \times n$ matrix or as a length n^2 vector. Secondly, we will look at the effectiveness of some commonly used basis choices for images in improving the sparsity of our problem.

2.1 Changing basis for the LASSO

Consider the LASSO from Definition 1.1.3 applied to our imaging problem, i.e. $A \in \mathbb{R}^{m \times n^2}$ is our speckle pattern matrix, $y \in \mathbb{R}^m$ is our vector of observed

[1] Take $x = B^{-1}(1, \dots, 1)^\top$. Then Bx is not sparse.

values and we wish to find $x \in \mathbb{R}^{n^2}$ with

$$x \in \underset{\xi \in \mathbb{R}^{n^2}}{\arg\min} \|A\xi - y\|_2^2 + \lambda \|\xi\|_1.$$

This formulation will induce sparsity in the canonical basis for $\mathbb{R}^{n^2}$. In our encoding, these basis vectors correspond to black images with a single white pixel. We sometimes call this the *pixel basis*.

Remark 2.1.1. Working in the pixel basis, $\|x\|_0$ is precisely the number of non-black pixels (non-zero elements) that x has. Sparse images in the pixel basis are therefore images with a high number of black pixels. In particular, if x corresponds to a solid white image (all pixels the same colour), then $\|x\|_0 = n^2$, even though this is a very simple image. Such an image does not contain much information but is not sparse in this basis.

We can deduce that the approximately sparse (or high signal-to-noise) vectors produced by this LASSO will generally be images with a large black (or dark) background and some comparatively bright spots. This can be desirable if that describes the sort of image we are trying to reconstruct, but this might not always be the case. This motivates a change of basis.

Any basis for $\mathbb{R}^{n^2}$ can be encoded in a change-of-basis matrix $B \in \mathrm{GL}(n^2, \mathbb{R})$. However, it is computationally very useful, especially if n^2 is large, to use an orthonormal basis, which makes $B \in \mathcal{O}(n^2)$. This is indeed the case in many applications, as we shall see, and indeed any basis can be made into an orthonormal basis by means of the Gram-Schmidt process or its matrix analogue, the QR-decomposition.

If we have a change-of-basis matrix B, solving $Ax = y$ in this new basis amounts to solving

$$AB^{-1}(Bx) = y.$$

In other words, we write $\overline{x} = Bx$, we solve $AB^{-1}\overline{x} = y$, and we obtain x by transforming $\overline{x}$: $x = B^{-1}\overline{x}$. Our new, basis-transformed LASSO problem is then given by

$$\overline{x} \in \underset{\xi \in \mathbb{R}^{n^2}}{\arg\min} \|AB^{-1}\xi - y\|_2^2 + \lambda_B \|\xi\|_1,$$

and our desired solution is $B^{-1}\overline{x}$. Here, λ_B is our adjusted LASSO parameter, whose choice is discussed in Section 5.2.

Remark 2.1.2. Another reason for the desirability of $B \in \mathcal{O}(n^2)$ is that general matrices in $\mathrm{GL}(n^2, \mathbb{R})$ can have arbitrarily small (albeit nonzero) norm, which can cause a loss of information when taking $x = B^{-1}\overline{x}$. See Proposition 5.2.1.

2.2 Vectorisation and Kronecker products

Images are fundamentally 2-dimensional objects, and as such many common transformations applied to them treat them as matrices. An $n \times n$ pixel grayscale

image can be encoded into a $n \times n$ matrix in a natural way by seeing such a matrix as a grid of brightness values. However, as we saw in Chapter 1, in our application we would like to treat images as *vectors* instead, by wrapping the gray-scale values in a n^2-dimensional row vector instead.

Definition 2.2.1. For a matrix $X = [x_{ij}]_{i,j} \in \mathbb{R}^{n\times n}$, we define the associated *vectorisation* as

$$\mathrm{vec}(X) = (x_{11}, \dots, x_{1n}, x_{21}, \dots, x_{2n}, \dots, \dots, x_{n1}, \dots, x_{nn})^\top \in \mathbb{R}^{n^2} \tag{2.2}$$

This is simply concatenating the rows of X one after the other and putting the resulting list of values into a column vector. This clearly defines a vector space isomorphism $\mathbb{R}^{n\times n} \cong \mathbb{R}^{n^2}$, by which the usual inner product in $\mathbb{R}^{n^2}$ corresponds to the Frobenius product. This is to say, if $X = [x_{ij}]_{i,j}, Y = [y_{ij}]_{i,j} \in \mathbb{R}^{n\times n}$, then

$$\langle \mathrm{vec}(X), \mathrm{vec}(Y) \rangle = \sum_{i,j=1}^{n} x_{ij} y_{ij} = \mathrm{Tr}\big(X^\top Y\big). \tag{2.3}$$

Often, we would like to have a more straightforward way to express the vectorisation of matrix products. This can be done by use of the Kronecker product.

Definition 2.2.2. Let $C = [c_{ij}]_{i,j} \in \mathbb{R}^{m\times n}$ and $D = [d_{ij}]_{i,j} \in \mathbb{R}^{p\times q}$. The Kronecker product of C and D is defined as

$$C \otimes D := \begin{bmatrix} c_{11}D & c_{12}D & \dots & c_{1n}D \\ c_{21}D & c_{22}D & \dots & c_{2n}D \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1}D & c_{n2}D & \dots & c_{nn}D \end{bmatrix} \in \mathbb{R}^{mp\times nq}$$

Effectively, this scales D by each element of C. We give some relevant properties of the Kronecker product, including its fundamental relation to the vectorisation, in appendix A.

Chapter 3

Traditional transforms

In this chapter, we look at changing the basis using traditional transformations like the discrete cosine transform (DCT) and the discrete wavelet transform (DWT). In particular, for the wavelets, we look at the Daubechies 4 wavelets, which are the simplest member of that group. We then apply these transformations to the optimization problem mentioned in chapter 2.

3.1 Discrete cosine transform

Sparsity-inducing bases are very usefull for image compression algorithms. The discrete cosine transform is famously used in JPEG [10] for that exact purpose. The one-dimensional DCT of $x = (x_0, \ldots, x_{M-1})$ is defined as

$$x_k^c = \sum_{n=0}^{M-1} x_n \cos\left(\frac{\pi}{M}(n + \frac{1}{2})k\right)$$

for $k = 1, \ldots, M-1$. It can be expressed as the following matrix transformation

$$x^c = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ \cos\left(\frac{\pi}{2M}\right) & \cos\left(\frac{3\pi}{2M}\right) & \ldots & \cos\left(\frac{(M-\frac{1}{2})\pi}{M}\right) \\ \vdots & \vdots & \ddots & \vdots \\ \cos\left(\frac{(M-1)\pi}{2M}\right) & \cos\left(\frac{(M-1)3\pi}{2M}\right) & \ldots & \cos\left(\frac{(M-1)(M-\frac{1}{2})\pi}{M}\right) \end{bmatrix} x. \quad (3.1)$$

This matrix is usually normalized in software implementations. The first row is multiplied by $\frac{1}{\sqrt{M}}$ and all subsequent rows are multiplied by $\sqrt{\frac{2}{M}}$. This makes the matrix orthogonal.

Due to the close relationship between the discrete cosine transformation, and the discrete Fourier transformation, it's possible to calculate AB^{-1} efficiently.

Of course, since the data we're using is fundamentally two-dimensional, it doesn't make much sense to use one-dimensional DCT. For a given 2D signal $x \in \mathbb{R}^{M_1 \times M_2}$ The two-dimensional discrete cosine transform is given by first

performing a 1D DCT over all the rows. Over the resulting matrix, we then apply another 1D DCT over all the columns:

$$X^c = C(CX^\top)^\top = CXC^\top,$$

where C is the normalized DCT matrix 3.1. Since we use flattened images, we modify our 2D DCT:

$$x^c = \text{vec}(C\,\text{mat}(x)C^\top).$$

And since C is orthogonal, the inverse can be found quickly:

$$x = \text{vec}(C^\top\,\text{mat}(x^c)C).$$

With the Kronecker product and A.0.1, we rewrite this formula as a matrix-vector product $x^c = (C \otimes C)x$, with the inverse $x = (C^\top \otimes C^\top)x^c$. Most importantly, these basis transformations can be split up in multiple steps, so we won't have to compute any N^2 by N^2 matrices.

3.2 Wavelet transform

3.2.1 Theoretical background

Wavelets are small oscillations comprising sections of a full signal which are typically short and, regardless of their intermediate behaviour, both begin and end at zero amplitude. They also allow for scaling, which involves altering the width of a wavelet, and shifting, which moves the wavelet along the time axis. Fourier transforms can fall short when we need to analyse a particular part of a signal, as they return global data, potentially obscuring local information. This is where Wavelet transforms can excel, as they decompose a signal into a series of wavelets and allow us to extract localized spectral and temporal information at a given time, which is particularly prevalent for electrocardiography [11].

The uncertainty principle of signal processing is key to the Wavelet transforms efficacy. It is as follows:

$$\Delta t \Delta f \geq \frac{1}{4\pi}.$$

It shows that, the higher the time resolution (or in other words, the smaller the time uncertainty), the higher the frequency uncertainty and the opposite also holds. High frequency components of a wave typically last in very short bursts, so in order to properly resolve them, we require a high time resolution, whereas this is less important in the case of low frequency components, which last a long time and thus have low time resolution. These issues pertain to "Short-Time Fourier Transforms", in which the signal is separated into different sections and each is Fourier-transformed individually.

Wavelet transform overcomes this hurdle by analysing various frequencies at different resolutions, improving time resolutions and lowering frequency resolutions for high frequencies and enacting the opposite for low frequencies. The

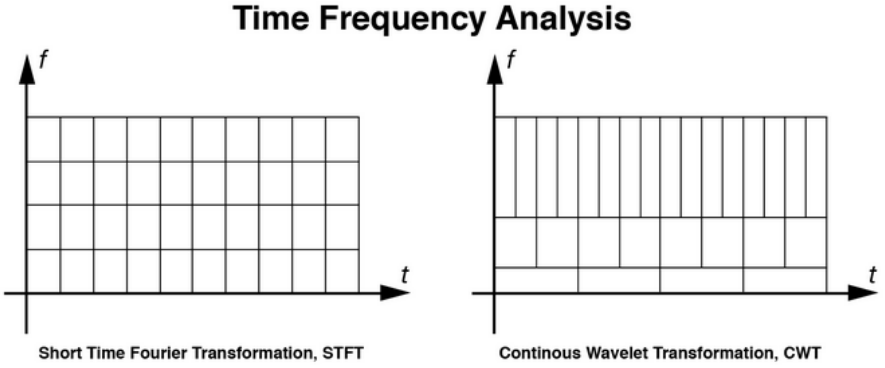


Figure 3.1: Time-frequency resolution chart

wavelets are scaled and shifted versions of a reference function called the mother wavelet, $\psi(t)$. These "child wavelets" are expressed as

$$\psi_{s,\tau} = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right)$$

where s acts as the scaling factor and τ is the translation of the wavelet. Thus, through a convolution with the original signal, the output of the transform is

$$F(s,\tau) = \frac{1}{\sqrt{|s|}}\int_{-\infty}^{\infty} f(t)\psi^*\left(\frac{t-\tau}{s}\right)dt$$

where $F(s,\tau)$ are the wavelet coefficients, which are functions of both the scale and the shift parameters. We wish to slide the wavelet of a particular scale across the signal, try a different scaling, then repeat to see how much of this wavelet can be found in the signal. Finally, in order to restrict the quantity of received data, s and τ are chosen to be discrete and, to increase accuracy, are taken to be dyadic: we take $s = 2^{-j}$ and $\tau = k2^{-j}$ for integers j, k.

A group of wavelets of particular interest are the Daubechies wavelets, defined as having n vanishing moments for $2n$ wavelet coefficients, meaning these wavelets have a scaling function (or father wavelet) that can generate polynomials of up to degree $n-1$. We can associate vanishing moments with a sparse set of wavelet coefficients, as they ensure wavelet coefficients for polynomials up to a degree of $n-1$ are zero. Another way of thinking about this is using the definition of the n-th probabilistic moment of a random variable, X,

$$\mathbb{E}[X_n] = \int_{-\infty}^{\infty} x^n f(x)dx$$

where X can be thought of as the Fourier transform, which has derivatives that return zero when evaluated at 0.

3.2.2 Discrete Daubechies-4 transform

The discrete wavelet transform works by applying a wavelet coefficient matrix W_{coef} to a vector of length N, then permuting the resulting matrix such that the vector is split into first the $N/2$ even-indexed elements (beginning at index 0) and then the $N/2$ odd-indexed elements. This is then repeated until only two or four elements remain. This process is illustrated in the following diagram for $N = 8$:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} \xrightarrow{W_{\text{coef}}(8)} \begin{bmatrix} x'_0 \\ x'_1 \\ x'_2 \\ x'_3 \\ x'_4 \\ x'_5 \\ x'_6 \\ x'_7 \end{bmatrix} \xrightarrow{P(8)} \left[\begin{array}{c} x'_0 \\ x'_2 \\ x'_4 \\ x'_6 \\ \hline x'_1 \\ x'_3 \\ x'_5 \\ x'_7 \end{array}\right] \xrightarrow{W_{\text{coef}}(4)} \left[\begin{array}{c} x''_0 \\ x''_2 \\ x''_4 \\ x''_6 \\ \hline x'_1 \\ x'_3 \\ x'_5 \\ x'_7 \end{array}\right] \xrightarrow{P(4)} \left[\begin{array}{c} x''_0 \\ x''_4 \\ \hline x''_2 \\ x''_6 \\ \hline x'_1 \\ x'_3 \\ x'_5 \\ x'_7 \end{array}\right] \xrightarrow{W_{\text{coef}}(2)} \left[\begin{array}{c} x'''_0 \\ x'''_4 \\ \hline x''_2 \\ x''_6 \\ \hline x'_1 \\ x'_3 \\ x'_5 \\ x'_7 \end{array}\right] \tag{3.2}$$

Here, $W_{\text{coef}}(m)$ is a wavelet coefficient matrix of size m and $P(m)$ a permutation matrix of size m that separates the even and odd-indexed elements. In order to represent this as simple matrix vector multiplications, without having to split up the vector after each permute step, we write the transformations $W_{\text{coef}}(m)$ and $P(m)$ in the form of block diagonal matrices:

$$W_{\text{coef}}(m) = \begin{pmatrix} (W_{\text{coef}})_m & 0 \\ 0 & I_{N-m} \end{pmatrix}, \qquad P(m) = \begin{pmatrix} P_m & 0 \\ 0 & I_{N-m} \end{pmatrix},$$

where $(W_{\text{coef}})_m, P_m \in \mathbb{R}^{m \times m}$. Which do the respective transformations on the first m elements of a vector. We can then obtain the complete discrete wavelet transform matrix W by repeated multiplication of these matrices:

$$W_N = W_{\text{coef}}(2)P(4)W_{\text{coef}}(4) \cdots W_{\text{coef}}(N/2)P(N)W_{\text{coef}}(N)$$

A wavelet coefficient matrix is constructed using the coefficient for a specific wavelet filter. Here we only look at one specific wavelet filter called the Daubechies-4 wavelet (DAUB4), which is the simplest one from the class discovered by Daubechies. It has only four coefficients c_0, c_1, c_2, c_3. The wavelet coefficient

matrix for DAUB4 is given by

$$W_{\text{coef}} = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 & & & & & & & & \\ c_3 & -c_2 & c_1 & -c_0 & & & & & & & & \\ & & c_0 & c_1 & c_2 & c_3 & & & & & & \\ & & c_3 & -c_2 & c_1 & -c_0 & & & & & & \\ & & & & & & \ddots & & & & & \\ & & & & & & & c_0 & c_1 & c_2 & c_3 & \\ & & & & & & & c_3 & -c_2 & c_1 & -c_0 & \\ c_2 & c_3 & & & & & & & & c_0 & c_1 & \\ c_1 & -c_0 & & & & & & & & c_3 & -c_2 & \end{pmatrix}, \tag{3.3}$$

where the black entries are zeroes. In order to be able to reconstruct a transform vector we require this matrix to be orthogonal. Which is the case if and only if the following two equations hold

$$\begin{aligned} c_0^2 + c_1^2 + c_2^2 + c_3^2 &= 1, \\ c_2 c_0 + c_3 c_1 &= 0. \end{aligned}$$

The following coefficients

$$c_0 = \frac{1+\sqrt{3}}{4\sqrt{2}}, \qquad c_1 = \frac{3+\sqrt{3}}{4\sqrt{2}}, \qquad c_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}, \qquad c_3 = \frac{1-\sqrt{3}}{4\sqrt{2}},$$

known as the DAUB4 coefficients, satisfy the orthogonality equations as well as the equations

$$\begin{aligned} c_3 - c_2 + c_1 - c_0 &= 0 \\ 0c_3 - 1c_2 + 2c_1 - 3c_0 &= 0 \end{aligned}$$

which are required for the order $p = 2$ approximation condition to hold [9].

We can apply this transform to images in the form of a matrix, instead of on one-dimensional vectors by first transforming along the rows and then the columns. Thus for an image $X \in \mathbb{R}^{n \times n}$ we get that the transform is given by

$$\hat{X} = W_n X W_n^\top \qquad\qquad \operatorname{vec}(\hat{X}) = (W_n \otimes W_n) \operatorname{vec}(X).$$

3.3 Results

We have applied the above mentioned transforms, as well as a combination of the two for the reconstruction of grayscale images. We attempt to solve the optimization problem as mentioned in section 2.1, namely:

$$\min_{\eta \in \mathbb{R}^{n^2}} \|AB^{-1}\xi - y\|_2^2 + \lambda \|\xi\|_1, \tag{3.4}$$

where $A \in \mathbb{R}^{m\times n}$ is our matrix of speckle patterns and $y \in \mathbb{R}^m$ the vector of projections of the true sample for each speckle pattern. For our simulated tests we used grayscale images of $n = 64$ pixels in width and height as our true samples and simulated $m = 307$ speckle patterns $\{a_k\}$, which we used to construct A and y according to Equation 3.4.

The transformation matrices B for the different methods are

(i) Baseline: $B = I_{n^2}$, no transformation for comparison.

(ii) DCT: $B^{-1} = C^{-1} \otimes C^{-1}$, with C the one-dimensional DCT matrix from section 3.1.

(iii) DWT: $B^{-1} = W^{-1} \otimes W^{-1}$, with W the one-dimensional wavelet matrix for Daubechies-4 wavelets from section 3.2.2.

(iv) Combination: $B^{-1} = (C \otimes C)(W^{-1} \otimes W^{-1})$, with C, W the same as in (ii) and (iii).

We then use GPSR to find a solution $\bar{\xi}$ to Equation 3.4. We compare this solution to the transformed true sample $B\,\mathrm{vec}(X)$ and to the true sample X after transforming the solution back by $B^{-1}\bar{\xi}$.
We make use of two metrics for our comparisons: the 2D correlation coefficient, which for two matrices X, Y is given by

$$\mathrm{corr2}(X,Y) = \frac{\sum_m \sum_n (X_{mn} - \bar{X})(Y_{mn} - \bar{Y})}{\sqrt{\left(\sum_m \sum_n (X_{mn} - \bar{X})^2\right)\left(\sum_m \sum_n (Y_{mn} - \bar{Y})^2\right)}},$$

and the structural similarity is given by

$$\mathrm{ssim}(X,Y) = \frac{(2\bar{X}\bar{Y} + c_1)(2\sigma_{XY} + c_2)}{(\bar{X}^2 + \bar{Y}^2 + c_1)(\sigma_X^2 + \sigma_Y^2 + c_2)},$$

where c_1, c_2 are regularization constants, σ_X, σ_Y the standard deviations, and σ_{XY} the cross-covariance.

In Figure 3.2 we can see the GPSR output and the recovered images for each transformation, when applied to a 64×64 Astrocyte image. The structural similarities (ssim) and correlation coefficients (corr2) for these images are shown in Table 3.1. We see that the method with the discrete cosine transform (3.2b) performs quite well, even better than the baseline (3.2a) when looking at the corr2 and ssim in Table 3.1. The wavelet transform (3.2b) alone performs poorly at each step. This performance is much improved when we look at the combination of the wavelet transform with the discrete cosine transform (3.2d), which performs very well if we look at the GPSR output but loses some quality when we transform it back. Looking closely at this transformed image we see a dark blue line at the left and top most edges. This is most likely caused by the Gibbs phenomenon in the final transformation.

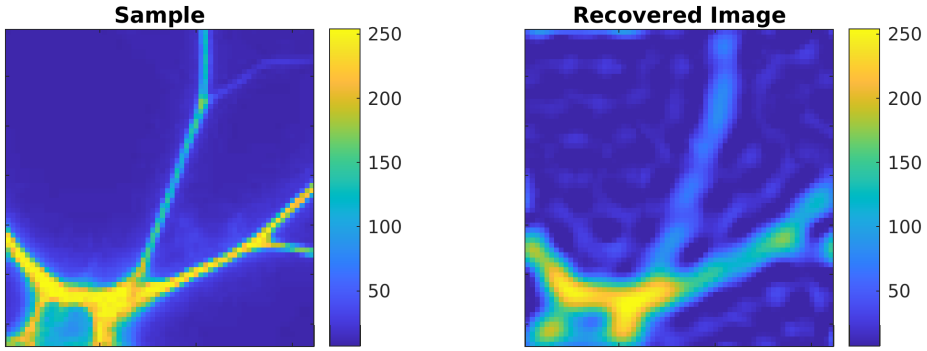
Astrocyte	GPSR Output		Recovered Image	
	corr2	ssim	corr2	ssim
Baseline	NA	NA	0.91749	0.25872
DCT	0.94775	0.00716	0.92562	0.27034
DAUB4	-0.03108	-8.2600-05	-0.02966	-1.3560-05
DAUB4 + DCT	0.94489	0.23078	0.90678	0.23078
DAUB4 + DCT (low passed)	NA	NA	0.92272	0.26467

Table 3.1: 2D Correlation and the Structural similarity of the recovered sample/image using the different transformations on the 64×64 Astrocyte sample.

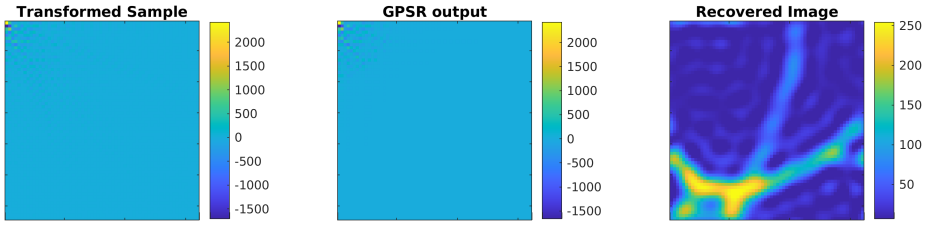
We can also look at the Fourier transforms of the recovered images for closer analysis, these are shown in Figure 3.3. When comparing these images we note one obvious difference between the original sample and the recovered images. The vast majority of the power for the recovered images is located in a small circle around the middle, whilst the original sample has some power outside this circle as well. The region outside of this circle are the higher frequencies, which then correspond to sharper details in the original space. This explains why none of the recovered images fully capture the fine details of the original sample in Figure 3.2.

We also note the horizontal and vertical lines in Figure 3.3e, which are responsible for the dark blue lines in the left and top most corners of the recovered image in Figure 3.2d. We can attempt to remove these by setting every value outside of the inner circle to zero, essentially applying a low-pass filter, and then transforming the result back using the inverse Fourier transform. The resulting recovered image is shown in Figure 3.4. The corresponding corr2 and simm values are in the bottom row of Table 3.1.

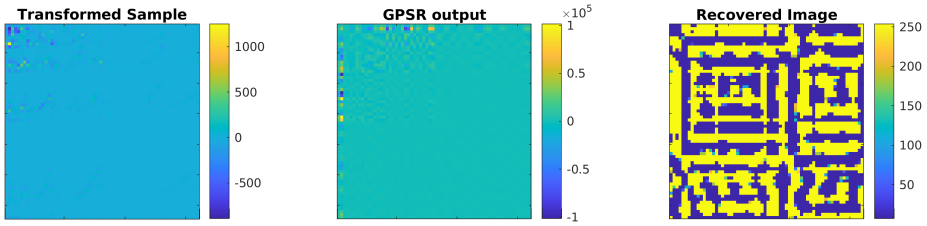
Lastly we have also attempted to recover the image with different numbers of speckle patterns m. The final recovered images for the different values of m are shown in Figure 3.5. We can see that going from $m = 102$ to $m = 307$ we get much better images for each method, but the improvement from going to $m = 614$ is little to none. Similarly in 3.6 we see the Fourier transforms of these recovered images as well.



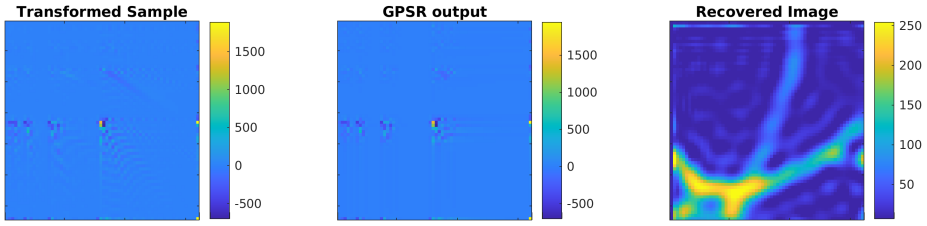
(a) Baseline; no transformation



(b) Discrete Cosine Transform

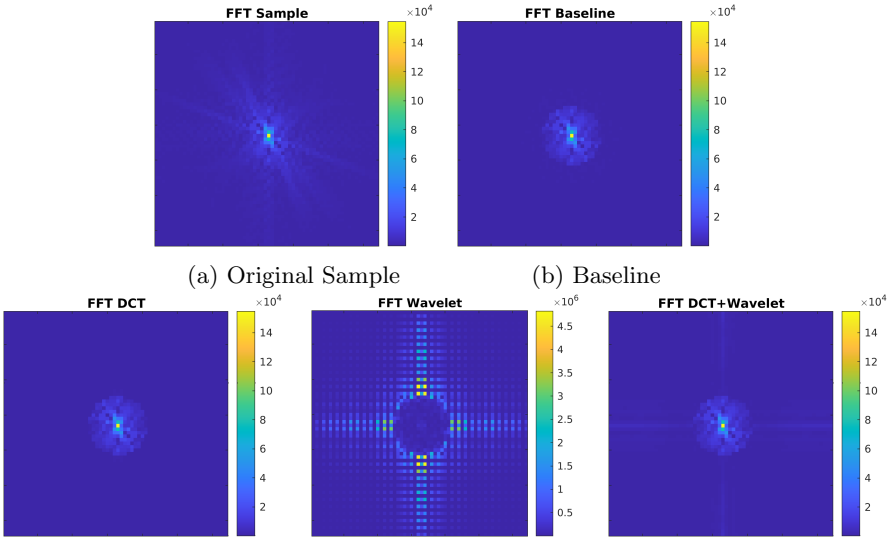


(c) Daubechies-4 Wavelet Transform



(d) Daubechies-4 Wavelet and Discrete Cosine Transform

Figure 3.2: 64×64 Astrocyte sample, reconstructed using GPSR with the three mentioned transformation (b,c,d) and no transformation (a).



(a) Original Sample (b) Baseline

(c) DCT (d) DAUB4 Wavelet (e) DAUB4 Wavelet and DCT

Figure 3.3: Complex magnitude of the Fourier Transform of the recovered image from each method (b,c,d,e) and the original sample (a), with the zero-frequency in the center.

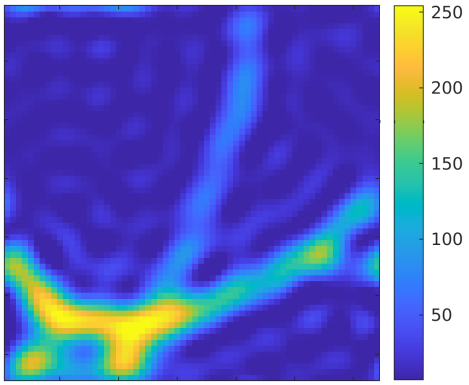


Figure 3.4: Low-pass filter applied to the Fourier transform of the recovered image obtained by the DAUB4 wavelet and DCT method and then transformed back using the inverse Fourier transform.

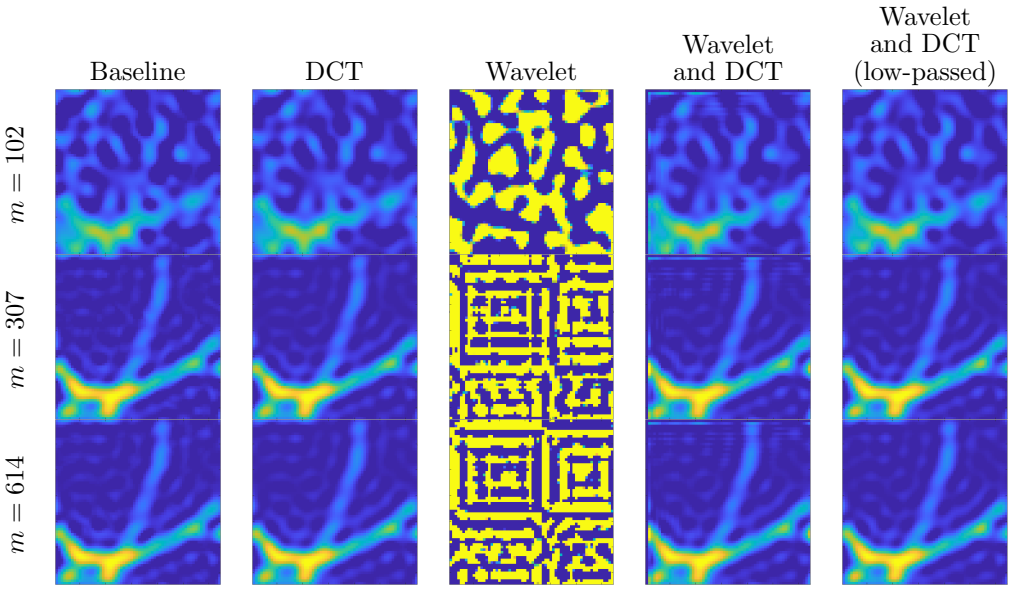


Figure 3.5: Final recovered image for each method with three different values for the number of speckle patterns $m \in \{102, 307, 614\}$.

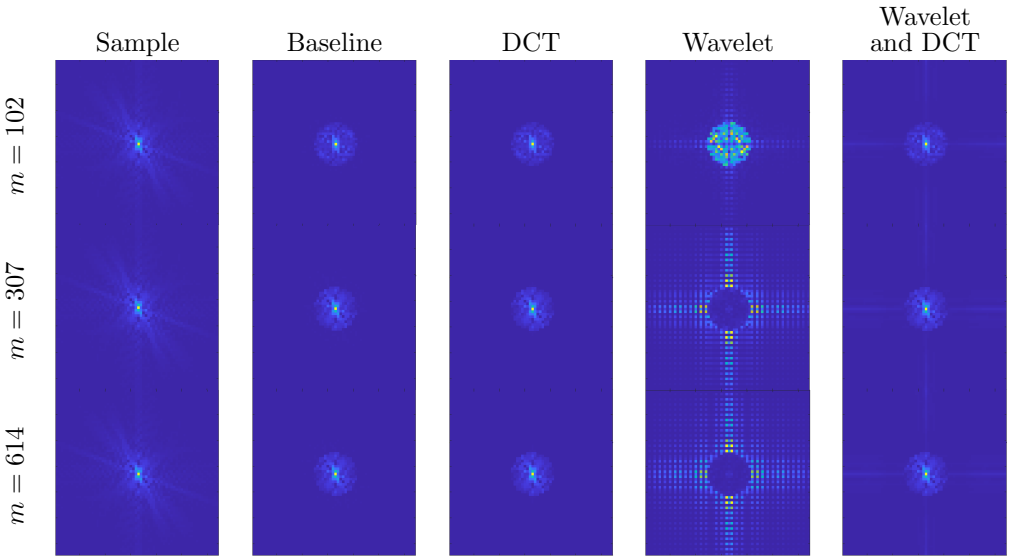


Figure 3.6: Fourier transforms of the recovered image for each method with three different values for the number of speckle patterns $m \in \{102, 307, 614\}$.

Chapter 4

Machine learning approach

In the previous chapter, we used wavelet and cosine transforms in order to obtain the basis transformation B. Now, by defining a sparsity-inducing loss function that preserves orthogonality, a neural network is defined to obtain this B with flattened training samples x as input. In this chapter we will train such networks on two datasets; CIFAR10 images with resolution 32×32 and a few other images with resolution 64×64. However, given the dimensions of our problem, we have that $B \in \mathrm{GL}(n^2, \mathbb{R})$. Therefore, using images with a high resolution will be very computationally intensive. So also a goal is to turn this into a more manageable system. One approach is by reducing the class of B matrices from which we wish to select our change of basis to a lower-dimensional subspace of $\mathrm{GL}(n, \mathbb{R})$ by use of the Kronecker product. Taking such a Kronecker product on the output matrices of two linked neural networks will result in our B. This GitHub repository contains the code.

4.1 Defining the neural networks

4.1.1 Matrix reduction using Kronecker product

We will make use of the Kronecker product and look at matrices of the form $C \otimes D = B \in \mathrm{GL}(n^2, \mathbb{R})$, such that $C, D \in \mathrm{GL}(n, \mathbb{R})$. This product is defined as in definition 2.2.2. Effectively, this scales D by each element of C. We can see that, in our case, the dimension of B will indeed be $n^2 \times n^2$.

The objective is to execute the calculation Bx efficiently in which $x \in \mathbb{R}^{n^2}$ will be flattened training samples. This calculation is performed by,

$$Bx = (C \otimes D)x.$$

4.1.2 Constructing a loss function

Recall the definition of the ℓ_0-norm in 1.1.1. Our problem in using the ℓ_0-norm stems from the fact that it is non-differentiable. This is a requirement to be able to perform backpropagation. So we instead opt for a different approach to counting the number of non-zero entries of the vector x. Let

$$loss_{sparse}[x] = \sum_i \sqrt{x_i^2 + \epsilon} \approx \|x\|_1, \quad \epsilon > 0 \text{ (small)}.$$

Subsequently, in order to keep our B orthogonal and therefore invertible, we combine this loss with additional components that ensure orthogonality. Namely,

$$loss_{orth.}[X] = \|XX^\top - I_n\|_2^2 = \text{MSE}(XX^\top - I_n),$$

in which I_n is the identity matrix with dimension $n \times n$ and MSE the mean squared error. Therefore, the total shared loss function used in the training of both networks is,

$$\begin{aligned} loss(x; C, D) &= loss_{sparse}[(C \otimes D)x] + \alpha \left(loss_{orth.}[C] + loss_{orth.}[D]\right) \\ &= \sum_i \sqrt{[(C \otimes D)x]_i^2 + \epsilon} + \alpha \left(\|CC^\top - I_n\|_2^2 + \|DD^\top - I_n\|_2^2\right), \end{aligned} \tag{4.1}$$

in which $\alpha > 0$ is some scaling parameter. See Fig. 4.1 for an overview.

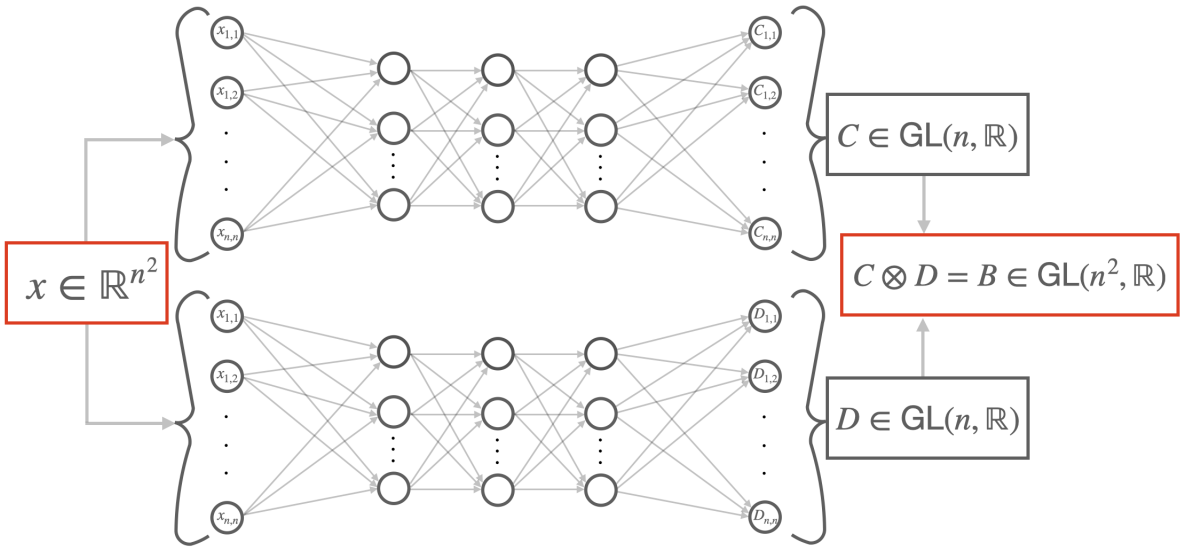


Figure 4.1: Schematics of linked neural networks with shared loss function 4.1.

4.2 Results

4.2.1 General idea

In these subsections, we train the networks on a training dataset of grayscale images. We first train a specified amount of iterations on the first image, then on the second image; until all specified images from the dataset have been passed through the network. ϵ is set to 10^{-8} and α to 10^{7} in order to reduce the artefacts present in the reconstruction that may occur due to the use of the Kronecker product. In Fig. 4.2, an example of a dataset is shown.

Figure 4.2: The ten 64×64 training samples used.

In Fig. 4.3, an example of the training of the network can be seen. An input image x is shown to which B is applied such that the result in the fourth image is obtained. In this specific iteration the sparsity increases from 19 to 1997 (amount of pixel value below a threshold of 10^{-15}). The third image shows that the system is approximately orthogonal. The final error of $BB^{\top}x$ was approximately $7.1 \cdot 10^{-17}$ with respect to the input image x.

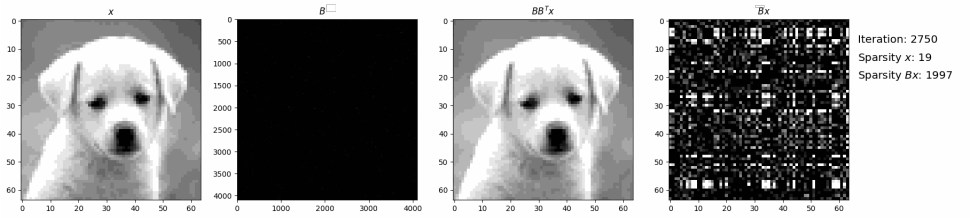


Figure 4.3: Training example in which sparsity is increased.

In Fig. 4.4, the image reconstruction using the GPSR algorithm can be seen using a biological fiber test sample. Note that using the identity for B does nothing, 'Kronecker B' refers to the loss presented in the previous subsection and 'Regular B' refers to a single neural network trained without the use of the Kronecker product. Above the reconstructed images the correlation and similarity are quantified with respect to the sample. Dependent on the initial

guess of the GPSR algorithm, the correlation stays approximately similar while the structural similarity decreases and error in fast Fourier transform increases. As can be seen in Fig. 4.4, GPSR approximates the lower frequencies accurate and fails at approximating the higher frequencies.

Noticeable in Fig. 4.4 is that the sharpness of the images seem to increase for the Kronecker and regular basis transformed GPSR reconstructed images. This is compared to the identity transformation which output is more smooth. It can thus be seen that the basis transformations increases sharpness, but do not reveal more details in the image than the identity transform. Therefore, the error stays similar. However, we suspect that this is due to the limitations of using the GPSR algorithm. When improved algorithms are developed and such basis transformations are applied, the sharpness increases but within the capability of the algorithm. And last, in appendix D more results using a larger CIFAR10 dataset are shown.

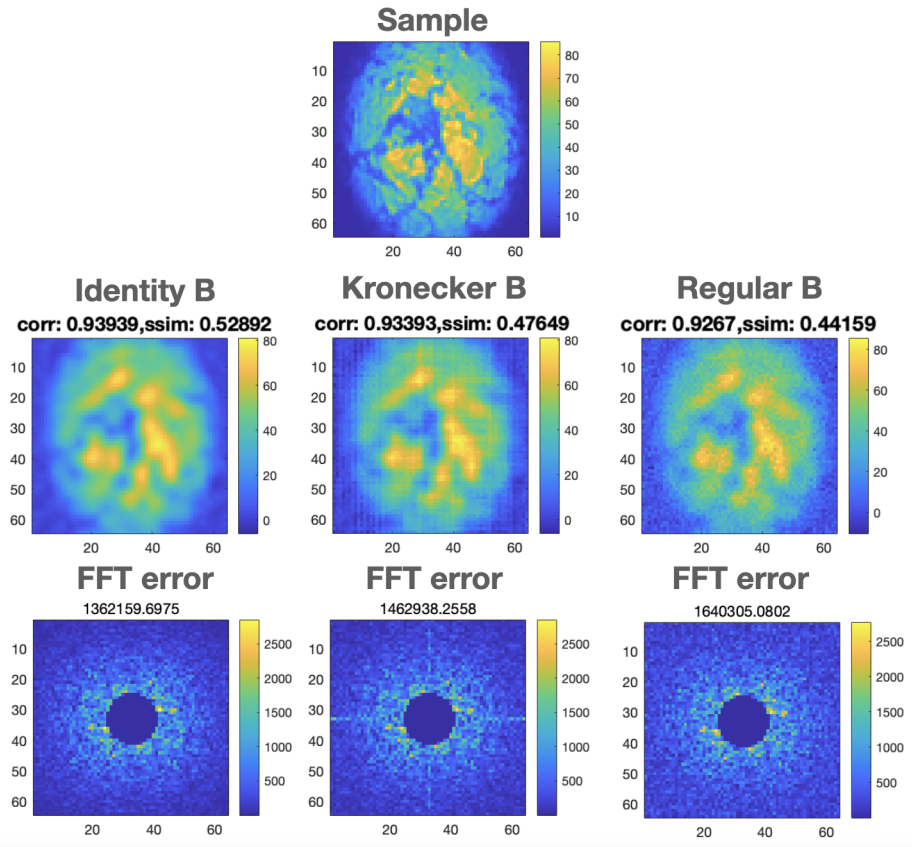


Figure 4.4: GPSR reconstructed test sample using no transformation (leftmost column), the Kronecker linked neural network basis transform (middle column) and the single neural network basis transform (rightmost column).

Chapter 5

Choosing the LASSO parameter

In our application, we use an algorithm (GPSR) to solve the LASSO problem given in Definition 1.1.3. For this, we must choose a positive real parameter, λ, which determines the importance of sparsity in relation to having exact solutions for our underdetermined problem. With $\lambda = 0$ we negate sparsity, with λ large enough all our solutions will be zero. In this chapter, we will briefly discuss the choice of this parameter, a hard problem which resists any obvious approach by means of training data. Furthermore, we examine how the parameter is affected by the basis changes discussed in Chapter 2. Due to time constraints, we show no implementation for this chapter – instead, we give theoretical results and discuss the possible implementation at the end of each section.

5.1 Choosing the parameter

There is no well-established method for choosing λ, even though the effectiveness of the LASSO method is significantly dependent on this choice (see [2]). In our implementation, the GPSR algorithm is used, which is appealing because it iteratively and adaptively chooses λ (see Appendix C.1). However, one still requires a starting value for λ, the choice of which affects the outcome of the algorithm, although in a less sensitive way than for the standard method.

In practice, one tries a range of λ's, examines the obtained results, and picks the value of λ that produced the most desirable outcomes. This is quite unsatisfactory from a scientific point of view, and the list of those who have tried and failed to give a better alternative over the years is long. In this chapter, we add our name to this list by briefly exploring an unsuccessful approach to substantiating a choice for λ using an analogous method to Chapter 2, based on the use of training data.

Suppose we have a set $S \subset \mathbb{R}^{n^2}$ of representative images, to which we can

expect our desired reconstruction to be similar. We would like these images to be solutions to the LASSO, i.e.

$$x \in \arg\min_{\xi \in \mathbb{R}^{n^2}} \|A\xi - Ax\|_2^2 + \lambda\|\xi\|_1, \qquad x \in S.$$

Is it possible to find λ in such a way that it ensures all $x \in S$ minimise the above function?

With a change of variable $\xi = x + h$, we would like the function

$$f(h) = \|Ah\|_2^2 + \lambda\|x + h\|_1$$

to have a minimum at 0. When is this the case?

Proposition 5.1.1. *Let $x \in \mathbb{R}^r$, $\lambda > 0$ and define $f : \mathbb{R}^r \to \mathbb{R}$ by*

$$f(h) = \|Ah\|_2^2 + \lambda\|x + h\|_1.$$

Then f has a minimum at 0 if and only if $x = 0$.

Proof. See Appendix B.1. □

This result shows an unfortunate fact: the LASSO will never converge to an exact solution as long as $\lambda \neq 0$. This stands to reason – the path to sparsity requires certain sacrifices. However, it does show that there is no obvious way to use training data to determine a good choice for λ.

5.2 Adjusting the parameter after a change of basis

Suppose that we have established a satisfactory choice for our regression parameter λ in Definition 1.1.3, denoted λ_I, but we wish to solve our linear regression problem in another basis for x. What would be a good choice for λ after this change of basis?

Recall from Section 2.1 that the LASSO problem after a change of basis given by $B \in \mathrm{GL}(n^2, \mathbb{R})$ is

$$\overline{x} \in \arg\min_{\xi \in \mathbb{R}^{n^2}} \|AB\xi - y\|_2^2 + \lambda_B\|\xi\|_1$$

Here, λ_B is our new choice of parameter. This is precisely what we wish to explore in this section. When choosing our new λ_B, we would like to make sure that the solutions we obtain with λ_I for our original problem, which we have deemed as satisfactory, will also be solutions for our new problem. The appropriate choice of λ_B for this purpose is given in the following proposition.

Proposition 5.2.1. *Suppose x is a LASSO for the original problem, i.e.*

$$x \in \arg\min_{\xi \in \mathbb{R}^{n^2}} \|A\xi - y\|_2^2 + \lambda_I \|\xi\|_1.$$

Then $B^{-1}x$ is a LASSO for the problem in the basis given by $B \in \mathrm{GL}(n^2, \mathbb{R})$ if we choose $\lambda_B = \lambda_I / \|B^{-1}\|_1$, i.e.

$$B^{-1}x \in \arg\min_{\xi \in \mathbb{R}^{n^2}} \|AB\xi - y\|_2^2 + \frac{\lambda_I}{\|B^{-1}\|_1} \|\xi\|_1.$$

Proof. See Appendix B.1. □

What this shows us is that, if we can compute $\|B^{-1}\|_1$ for our change of basis matrix B, then a reasonable starting point is to pick $\lambda_B = \lambda_I / \|B\|_\infty$. In this result, we can see something that was already hinted at before (see Remark 2.1.2): if $\|B\|_1$ is small, then since $\|B^{-1}\|_1 \geq 1/\|B\|_1$, we get a very small choice for λ_B. This can result in a loss of information, since a small enough LASSO parameter will ignore sparsity, which makes the choice of basis irrelevant. We can solve this problem by requiring $B \in \mathrm{O}(n^2)$. This also simplifies the computation of $\|B^{-1}\|_1$.

Lemma 5.2.2. *Suppose $B \in \mathrm{O}(n^2)$. Then*

$$\|B^{-1}\|_1 = \|B\|_\infty.$$

Proof. See Appendix B.1. □

From this, we get that for $B \in \mathrm{O}(n^2)$,

$$\lambda_B = \frac{\lambda_I}{\|B\|_\infty} \tag{5.1}$$

In some cases, an explicit computation of $\|B\|_\infty$ may be very costly, due to the large size of this matrix. We can, however, give some estimates for this norm using orthogonality and sparsity, which will give us estimates for the optimal parameter λ_B. This is done in detail in Appendix B.2. We can sum up the resulting estimations in the following corollary.

Corollary 5.2.3. *Let λ_I be our selected optimal parameter for the problem without a change of basis. Let $B \in \mathrm{O}(n^2)$ and denote $\lambda_B = \lambda_I / \|B\|_\infty$ the associated parameter from Proposition 5.2.1. Then we have the following bound:*

$$\max\left\{\frac{1}{n}, \frac{1}{M(B)}\right\} \lambda_I \leq \lambda_B \leq \lambda_I,$$

where $M(B)$ is as in Corollary B.2.2.

A final conclusion we can draw from this corollary is that, if computing $\|B\|_\infty$ is undesirable, a good choice for λ after an orthonormal change of basis is simply $\lambda = \lambda_I$, since this bounds λ_B from above and larger λ maximises the importance of sparsity, which is the reason why we changed basis in the first place. This is also a possible argument for choosing $\lambda > \lambda_B$.

Chapter 6

Conclusion

In chapter 3.3 we have seen how three transformations; discrete cosine transform (DCT), DAUB4 wavelet transform, and a combination of the two performed when compared to the identity transformation. It is unclear why the DAUB4 wavelet transform did not produce any usable output and should be investigated in the future. The DCT and the combination transformation both performed no worse, but also not much better than the baseline. All in all, none of these three transformations increased the accuracy of the reconstructed images significantly.

In chapter 4 we have seen promising results regarding the use of sparsity-inducing neural networks to obtain our basis transformation matrix B. Visually the images seem to improve; however the errors (correlation, structural similarity, FFT) stay approximately equal. We suspect that it is due to the limitations of the GPSR algorithm that the reconstruction quality does not significantly improve. When new algorithms are developed, these methods could further enhance the reconstructions. The lack of improvement may also be caused by an inherent lack of information in the system or the particular samples we have tested our bases on.

In chapter 5, a data training approach to choosing the optimal LASSO parameter λ was shown to fail due to the theoretical characteristics of the LASSO. We successfully give an expression for the optimal transformation for λ after applying a change of basis. This includes a term that is hard to compute, but if the basis is orthogonal, we argue that leaving the parameter unchanged might be sufficient.

When looking at the Fourier transforms of the reconstructed images (Figure 3.3) we noted that all methods fail to reproduce the high-frequency behaviour of the original sample. This high-frequency part coincides with finer (sharper) details in the original image. Figure 3.6 showed that an increase in the number of speckle patterns did not change this, in fact, the radius of the circle in the Fourier transform plots remained constant. Thus this problem can not be solved by using more speckle patterns. Instead, it could be caused by the λ term in the optimization problem that enforces sparsity in the solution or by some

limitations in the optimization algorithm that we used (GPSR).

In the future we may solve the problem directly through gradient descent over machine learning. We can make use of efficient algorithms, such as the one described in [12], to preserve orthogonality at each update step. A quick explanation of this method is given in Appendix E. We may also expand the set of test images to get a bigger picture of performance. Lastly, the failure of GPSR to reconstruct the high-frequency behaviour of the sample images should be analyzed further.

Acknowledgements

We would like to express our sincere gratitude to ARCNL for the opportunity to work on this problem. We are grateful to Ksenia Abrashitova, Tristan van Leeuwen, Rob Bisseling and Ivan Kryven for their valuable contributions to this work.

Appendix A

Kronecker Product

Proposition A.0.1. *Let $C, D, E, F \in \mathbb{R}^{n\times n}$. Then*

(i) $(C \otimes D)(E \otimes F) = CE \otimes DF$. *In particular,* $(C \otimes D)^{-1} = C^{-1} \otimes D^{-1}$.

(ii) $(C \otimes D)^\top = C^\top \otimes D^\top$.

(iii) $\operatorname{vec}(XYZ) = (Z^\top \otimes X)\operatorname{vec}(Y)$.

For more details, see [5]. We conclude by giving a short relevant corollary applying to a common class of image basis transformations.

Corollary A.0.2. *Let $F \in \mathbb{R}^{n\times n}$ be orthogonal. Then the linear map $T_F : \mathbb{R}^{n^2} \to \mathbb{R}^{n^2}$ given by*

$$T_F(\operatorname{vec}(X)) = \operatorname{vec}(FXF^\top)$$

is orthogonal. In particular, the corresponding basis on $\mathbb{R}^{n^2}$

$$\{\operatorname{vec}(FE_{ij}F^\top)\}_{i,j=1}^n,$$

where E_{ij} is the matrix whose ij-th entry is 1 and the rest are zero,[1] *is orthonormal.*

Proof. By Proposition A.0.1 *(iii)*, $\operatorname{vec}(FXF^\top) = F \otimes F \operatorname{vec}(X)$, so T_F corresponds to multiplication by $F \otimes F$. Now, $(F \otimes F)^\top = F^\top \otimes F^\top$, and by the inverse property for Kronecker products,

$$(F \otimes F)^{-1} = (F^{-1} \otimes F^{-1}) = (F^\top \otimes F^\top),$$

so this matrix is orthogonal. Lastly, note that $\{\operatorname{vec}(FE_{ij}F^\top)\}_{i,j=1}^n$ is simply the image of the canonical basis of $\mathbb{R}^{n^2}$ by this transformation, so it must be orthonormal. □

This shows that if we have a basis transformation given in matrix form by $X \mapsto FXF^\top$ for some orthogonal matrix $F \in \mathrm{O}(n)$, then the basis transformation in vector form will be orthogonal as well, i.e. $B \in \mathrm{O}(n^2)$.

[1] Note $\{\operatorname{vec}(E_ij)\}_{i,j=1}^n$ is the canonical basis for $\mathbb{R}^{n^2}$

Appendix B

Technical results

B.1 Proofs

Proof of Proposition 5.1.1. Clearly if $x = 0$, since $\lambda > 0$, the properties of norms guarantee that f has a minimum at 0. Consider now the case $x \neq 0$. Note that $\|Ah\|_2^2 = \mathcal{O}(\|h\|^2)$ around 0. Hence,

$$f(h) = \lambda\|x + h\|_1 + \mathcal{O}(\|h\|^2).$$

Now if $x \neq 0$, then in particular $x_k \neq 0$ for some k, and thus there exists a neighbourhood of 0 in $\mathbb{R}^{n^2}$ such that $x_k \neq 0$. Then if e_k is the k-th basis vector,

$$f(te_k) = \lambda\|x + te_k\|_1 + \mathcal{O}(t^2) = \lambda\left(\operatorname{sgn}(x_k)(x_k + t) + \sum_{\substack{i=1 \\ i\neq k}}^{n^2} |x_i|\right) + \mathcal{O}(t^2)$$

for $|t|$ small enough, which does not have a local minimum at $t = 0$. In particular, f cannot have a global minimum at $h = 0$. □

Proof of Proposition 5.2.1. Substituting $\xi = B^{-1}x$ in the problem we want to minimise,

$$\begin{aligned}\|ABB^{-1}x - y\|_2^2 + \frac{\lambda_I}{\|B^{-1}\|_1}\|B^{-1}x\|_1 &\leq \|Ax - y\|_2^2 + \frac{\lambda_I}{\|B^{-1}\|_1}\|B^{-1}\|_1\|x\|_1 \\ &\leq \|Ax - y\|_2^2 + \lambda_I\|x\|_1.\end{aligned}$$

Since the last expression is minimal, and $y \mapsto B^{-1}y$ is a bijection, we can conclude that $B^{-1}x$ indeed minimises our LASSO problem. □

Proof of Lemma 5.2.2. Recall that the 1-norm and the ∞-norm of a matrix $A \in \mathbb{R}^{r\times s}$ is

$$\begin{aligned} \|A\|_1 &= \max_{1\le j\le s} \sum_{i=1}^{r} |a^i{}_j| \\ \|A\|_\infty &= \max_{1\le i\le r} \sum_{j=1}^{s} |a^i{}_j|. \end{aligned} \tag{B.1}$$

Consequently, $\|A^\top\|_1 = \|A\|_\infty$. Furthermore, since B is orthogonal, $B^{-1} = B^\top$. Combining these two facts, we get that $\|B^{-1}\|_1 = \|B^\top\|_1 = \|B\|_\infty$. □

B.2 Estimates for the norm of a basis transformation

In Equation (5.1), we give an expression for the optimal parameter after a change of basis as a function of the ∞-norm of the corresponding matrix. In some cases, an explicit computation of this norm may be very costly, due to the large size of this matrix. We can, however, give some estimates for this norm using orthogonality and sparsity, which will give us estimates for the optimal parameter λ_B.
Notice $\|B\|_2 = 1$, since B is orthogonal. Can we give any similar estimation for the ∞-norm and the 1-norm? A possible initial approach is as follows. Recall that for any $r \in \mathbb{N}$ we have the following bounds:

$$\|x\|_2 \le \|x\|_1 \le \sqrt{r}\,\|x\|_2 \qquad \forall\, x \in \mathbb{R}^r \tag{B.2}$$

For $B \in \mathrm{O}(n^2)$, the rows b^j and columns b_i all satisfy $\|b^j\|_2 = \|b_i\|_2 = 1$. This, along with Equation (B.1), gives the bounds

$$\|B\|_2 = 1 \le \|B\|_1, \|B\|_\infty \le n = \sqrt{n^2}\,\|B\|_2$$

This upper bound is not very satisfactory, however, since we can expect n to be quite large in applications. Note that the upper bound in Equation (B.2) is reached precisely when x is in the subspace generated by $(1,\dots,1) \in \mathbb{R}^r$, that is, x has minimal sparsity. This motivates the use of the 0-norm.

Lemma B.2.1. *For $r \in \mathbb{N}$, $1 \le p \le \infty$ and $x \in \mathbb{R}^r$, we have*

$$\|x\|_0^{1/p} \min_{\substack{1\le j\le r\\ x^j\neq 0}} |x^j| \le \|x\|_p \le \|x\|_0^{1/p} \max_{1\le j\le r} |x^j|$$

Proof. Note that for $p = \infty$, $\|x\|_0^{1/p} = 1$ and the bounds are trivial. Else,

$$\|x\|_p^p = \sum_{j=1}^{r} |x^j|^p \ge \sum_{\substack{j=1\\ x^j\neq 0}}^{r} \min_{\substack{1\le j\le r\\ x^j\neq 0}} |x^j|^p = \|x\|_0 \min_{\substack{1\le j\le r\\ x^j\neq 0}} |x^j|^p$$

Now use monotony of $t \mapsto t^p$ for $t \geq 0$ to take the p out of the minimum and get the desired lower bound. The upper bound is obtained completely analogously. □

This lemma gives a sharper upper bound than Equation (B.2), at the cost of sparsity. Choosing a sparse basis transformation B is restrictive, but has computational benefits. We can now give a tighter bound for the ∞-norm of sparse B.

Corollary B.2.2. *Let $B \in \mathrm{O}(r)$ for some $r \in \mathbb{N}$. Then*

$$1 \leq \|B\|_\infty \leq M(B), \tag{B.3}$$

where

$$M(B) := \max_{1 \leq i \leq r} \|b^i\|_0 \max_{1 \leq j \leq r} |b^i{}_j|$$

and $b^1, \ldots, b^r$ are the rows of B.

Proof. The lower bound comes from Equation (B.2). Using Equation (B.1) and Lemma B.2.1 for $p = 1$,

$$\|B\|_\infty = \max_{1 \leq i \leq r} \|b^i\|_1 \leq \max_{1 \leq i \leq r} \|b^i\|_0 \max_{1 \leq j \leq r} |b^i{}_j|.$$

□

Appendix C

GPSR algorithm

C.1 GPSR algorithm

For the purposes of this paper, we solve for the LASSO by way of the Gradient Projection for Sparse Reconstruction (GPSR) algorithm. One of its main advantages is that it iteratively modifies its choice of λ to be more optimal. However, it still requires a starting value. A brief discussion of how to substantiate a choice for this parameter is given in Chapter 5.

The GPSR algorithm for finding the LASSO involves reformulating (1.1) as a bound-constrained quadratic problem (BCQP) and then solving this new problem, specifically. We refer to the term involving the l_1-norm as the regularization and it is quintessential for gradient projection problems. Its purpose is to further sparsify our input, x, achieved by approximating sufficiently small elements of x to zero.

The GPSR algorithm is advantageous, since it places no assumptions on A; it only considers matrix-vector products with A and $A^\top$. Unlike many other similarly-inclined algorithms, GPSR requires only a single level of iteration [8]. However, its functionality is dependent on the sparsity of the input, with it performing far better for a sparser system.

C.1.1 Warm start and time complexity

One of the primary appeals of GP algorithms is their warm start approach. Upon obtaining a solution of (1.1) for some λ, we can make a good estimation of solutions for nearby λ, as opposed to starting again from scratch, with the number of necessary iterations decreasing depending on how close together the λ's are. Using this, we would hope to get solutions for a range of λ's and then choose the optimal one. The reason we reject the idea of a cold start is that GPSR performs less effectively at smaller values of λ, so we find an optimal solution for a larger λ and incrementally decrease λ, using the notion of a warm start.

It is also quite a cost effective algorithm, as the bulk of the cost stems from computations made in the BCQP, noticeably through applications of A and $A^\top$ or inner product multiplications. Experimentally, a complexity of around $O(n^{0.9})$ has been achieved, where n is relative to the size of the matrix [8]. This was however done with a both a reasonably sparse, signal and matrix A.

Appendix D

CIFAR10 results

CIFAR10 is a (labelled) dataset containing images of planes, cars, bird, cats, deers, dogs, frogs, horses, ships and trucks. After applying a grayscale to these images, they are a good test set for finding our sparsity-inducing basis transformation. It is also possible to use the MNIST handwritten digit dataset; however since this dataset is sparse already, applying such a basis transform would not be as beneficial. In this section, we train the networks on a larger set of 100 images with a resolution of 32×32 over 200 iterations each (see Fig. D.1). Therefore, in total 20.000 iterations are executed and α is again set to 10^7 and ϵ to 10^{-8}.

In Fig. D.1, the GPSR algorithm has been performed on a test case. It again seem to increase the sharpness of the images compared to the identity GPSR reconstructed image. However, the errors stay again approximately similar due to the limitations of the GPSR algorithm.

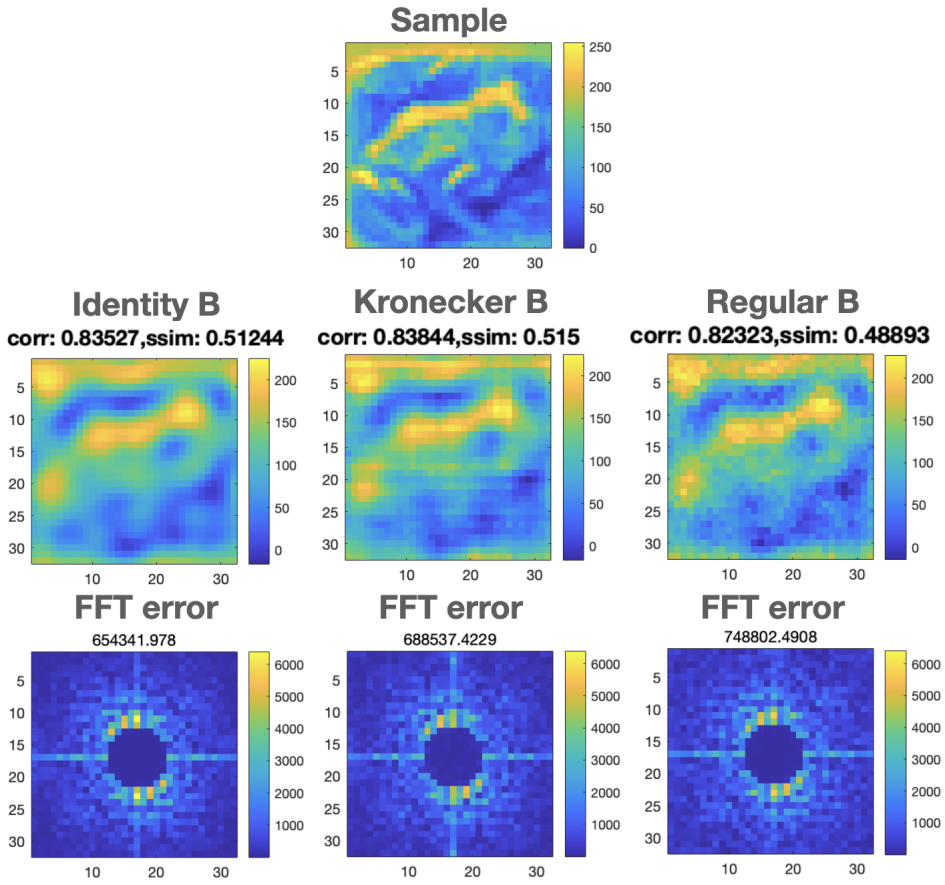


Figure D.1: GPSR reconstructed test sample using no transformation, the Kronecker basis transform and the single neural network basis transform.

Appendix E

Orthogonal Gradient Descent algorithm

We give a quick summary of the gradient descent algorithm discussed in [12] that preserves orthogonality.

Suppose we have a generalized version of our optimization problem given by

$$\min_{X \in \mathbb{R}^{n \times m}} \mathcal{F}(X), \text{ s.t. } X^\top X = I, \tag{E.1}$$

where $\mathcal{F} : \mathbb{R}^{n \times m} \to \mathbb{R}$ is some differentiable function.

The orthogonality preserving update works as follows. Given a feasible point X and the gradient $G = \nabla_X \mathcal{F}(X) = \frac{\partial \mathcal{F}(X)}{\partial X}$, define a skew-symmetric matrix in one of the following two ways:

$$\text{(i)} \quad A = GX^\top - XG^\top, \tag{E.2}$$

$$\text{(ii)} \quad A = (P_X G)X^\top - X(P_X G)^\top, \quad \text{where } P_X = \left(I - \frac{1}{2}XX^\top\right). \tag{E.3}$$

The new point is then determined by

$$Y(\tau) = QX, \quad \text{with } Q = \left(I + \frac{\tau}{2}A\right)^{-1}\left(I - \frac{\tau}{2}A\right). \tag{E.4}$$

The matrix Q is known as the Cayley transform. The new point has a few key properties: $Y(\tau)$ is smooth in τ, $Y(0) = X$, and $Y(\tau)^\top Y(\tau) = X^\top X$ for all $\tau \in \mathbb{R}$. The most important property it has is that $\frac{\mathrm{d}}{\mathrm{d}\tau}Y(\tau)|_{\tau=0}$ is the projection of the negative gradient $-G$ onto the tangent space of the feasible set $\{X \in \mathbb{R}^{n \times m} : X^\top X = I\}$ at X, hence $\{Y(\tau)\}_{\tau \geq 0}$ is a descent path which, for a proper value of τ converges to a stationary point.

Bibliography

[1] Ksenia Abrashitova and Lyubov V Amitonova. "High-speed label-free multimode-fiber-based compressive imaging beyond the diffraction limit". In: *Optics Express* 30.7 (2022), pp. 10456–10469.

[2] Sohail Chand. "On tuning parameter selection of lasso-type methods - a monte carlo study". In: *Proceedings of 2012 9th International Bhurban Conference on Applied Sciences Technology (IBCAST)*. 2012, pp. 120–129. DOI: 10.1109/IBCAST.2012.6177542.

[3] Mark A Davenport et al. *Introduction to compressed sensing*. 2012.

[4] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer New York, 2009. DOI: 10.1007/978-0-387-84858-7. URL: https://doi.org/10.1007/978-0-387-84858-7.

[5] Roger A Horn and Charles R Johnson. *Topics in Matrix Analysis*. Cambridge, England: Cambridge University Press, June 1994.

[6] Stephen G Lipson, Henry Lipson, and David Stefan Tannhauser. "Preface to the third edition". In: *Optical Physics*. Cambridge: Cambridge University Press, July 1995, pp. xiii–xiv.

[7] Benjamin Lochocki et al. "Ultimate resolution limits of speckle-based compressive imaging". In: *Optics Express* 29.3 (2021), pp. 3943–3955.

[8] Stephen J. Wright Mário A. T. Figueiredo Robert D. Nowak. "Gradient Projection for Sparse Reconstruction: Application to Compressed Sensing and Other Inverse Problems". In: *IEEE journal of selected topics in signal processing* 1.4 (Dec. 2007).

[9] William H. Press et al. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. 3rd ed. Cambridge University Press, 2007. ISBN: 0521880688.

[10] *Recommendation T.81*. Tech. rep. International Telecommunication Union, 1993.

[11] Shawhin Talebi. "The Wavelet Transform: An Introduction and Example". In: (2020).

[12] Zaiwen Wen and Wotao Yin. "A feasible method for optimization with orthogonality constraints". In: *Mathematical Programming* 142.1 (Dec. 2013), pp. 397–434. ISSN: 1436-4646. DOI: 10.1007/s10107-012-0584-1. URL: https://doi.org/10.1007/s10107-012-0584-1.