entropy

Article

Adam and the Ants: On the Influence of the Optimization Algorithm on the Detectability of DNN Watermarks

Betty Cortiñas-Lorenzo and Fernando Pérez-González *

Atlanticc Research Center, University of Vigo, 36310 Vigo, Spain; bcortinas@gts.uvigo.es
* Correspondence: fperez@gts.uvigo.es

Received: 8 October 2020; Accepted: 4 December 2020; Published: 6 December 2020

Abstract: As training Deep Neural Networks (DNNs) becomes more expensive, the interest in protecting the ownership of the models with watermarking techniques increases. Uchida et al. proposed a digital watermarking algorithm that embeds the secret message into the model coefficients. However, despite its appeal, in this paper, we show that its efficacy can be compromised by the optimization algorithm being used. In particular, we found through a theoretical analysis that, as opposed to Stochastic Gradient Descent (SGD), the update direction given by Adam optimization strongly depends on the sign of a combination of columns of the projection matrix used for watermarking. Consequently, as observed in the empirical results, this makes the coefficients move in unison giving rise to heavily spiked weight distributions that can be easily detected by adversaries. As a way to solve this problem, we propose a new method called Block-Orthonormal Projections (BOP) that allows one to combine watermarking with Adam optimization with a minor impact on the detectability of the watermark and an increased robustness.

Keywords: watermarking; deep neural networks; optimization algorithms; Adam; stochastic gradient descent; detectability

1. Introduction

Deep learning has substantially impacted technology over the last years, becoming an important center of attention for researchers all over the world. Such a great impact stems from the versatility it offers as well as the excellent results that DNNs achieve on multiple tasks, like image classification or speech recognition, which often reach and even surpass human-level performance [1,2].

However, far from being a simple task, the design of new DNNs is generally expensive not only in terms of human effort and time needed to build effective model architectures but mostly because of the large volume of suitable data that must be gathered and the vast amount of computational resources and power used for training. Consequently, businesses owning costly models are interested in protecting them from any illicit use, and this growing need has recently led researchers to a common concern on how to embed watermarks on DNNs. As a result, several frameworks for protecting the intellectual property of neural networks were proposed in the literature, and can be classified as black-box or white-box approaches. The first kind of methods (i.e., black-box) do not need to access model parameters for detecting the presence of watermarks. Instead, key inputs are introduced as special triggers to identify the original network, either by the use of the so-called adversarial examples [3] or backdoor poisoning [4].

White-box approaches, in contrast, directly affect the model parameters in order to embed watermarks. One of the most significant white-box contributions was proposed by Uchida et al. in [5,6], the algorithm under study in this paper. This watermarking framework employs a regularization term that defines a cost function for embedding the secret message. This regularizer, which transforms weights into bits by means of a projection matrix in a similar way to spread–spectrum techniques used in watermarking [7], is added to the main loss function and can be applied either at the beginning of the training phase (i.e., from scratch) or during fine-tuning steps.

On the other hand, when training a DNN, there are several optimization algorithms to choose from. With their pros and cons, some of the most popular are the classic Stochastic Gradient Descent (SGD) [8] and Adaptive Moment Estimation (Adam) [9]. We found that, in order to perform watermarking following the approach in [5,6], it is necessary to pay close attention to the optimization algorithm being used. In this paper, we prove that Adam, although being an appealing algorithm for lots of applications—especially because of its efficiency and training speed—poses a big problem when implementing this watermarking method.

As any other watermarking algorithm, it is important to meet some minimal requirements regarding fidelity, robustness, payload, efficiency, and undetectability [5,6,10]. This latter property involves the need for concealing any clue that would let unauthorized parties know if a watermark was embedded, along with its further consequences. In other words, from a steganographic point of view, watermarks should be embedded into the DNN without leaving any detectable footprint.

We show that, unlike SGD, Adam significantly alters the distribution of the coefficients at the embedding layer in a very specific way and, thus, it compromises the undetectability of the watermark. In particular, when using Adam, the histogram of the weights at the embedding layer shows that these coefficients tend to group together at both sides of zero, originating two visible spikes that grow in magnitude with the number of iterations, so that the initial distribution ends up changing completely. As we will see later, the update direction given by Adam depends on the projection matrix used for watermarking; specifically, it is based on the sign function, responsible for the symmetric two-spiked shape that emerges. This behavior is somewhat surprising, as with a pseudorandom projection matrix, one might expect that the weights evolve with random speeds; in contrast, the weights tend to move in unison in a way that is reminiscent of ants after they have established a solid path from the nest to a food source. Then, the statistical dependence of the weights with the projection matrix is weak, and this is mainly due to the information collapse induced by the sign function; this loss is invested in making the weights more conspicuous and, therefore, the watermark more easily detectable. As we will confirm, the sign phenomenon does not occur in other algorithms like SGD. The appearance of the sign was already pointed out and studied—without considering watermarking applications—by the authors in [11], who explained the adverse effects of Adam on generalization [12] as a consequence of this aspect. However, instead of delving into the general performance of the optimization algorithm itself, we show that the sign function which is involved in Adam's update direction is detrimental for watermarking purposes, so we highlight the need for being careful with the selection of the optimization algorithm in these cases.

In this paper, we carry out a theoretical and experimental analysis and compare the results using Adam and SGD optimization. The analysis can be extended to other optimization algorithms. As a way to measure the similarity between the original weight distribution and the resulting one after the watermark embedding, we will use the Kullback–Leibler divergence (KLD) [13]. We will show that, as expected, when we use Adam, the KLD between both distributions is considerably larger than when we use SGD, thus confirming that, as opposed to SGD, Adam modifies the original distribution to a great extent. Furthermore, in order to perform this kind of watermarking and, at the same time, enjoy the advantages that Adam optimization provides, we propose a new method called Block-Orthonormal Projections (BOP), which uses a secret transformation matrix in order to reduce the detectability of the watermark generated by Adam. As we will see, BOP allows us to considerably reduce the KLD to small values which are comparable to those obtained with SGD. Therefore, we show that BOP allows us to preserve the original shape of the weight distribution.

In summary, this work makes the following two-fold contribution:

- We provide mathematical and experimental evidence for SGD and Adam to show that: (1) in contrast to SGD, the changes in the distribution of weights caused by Adam can be easily detected when embedding watermarks following the approach in [5,6] and, hence, (2) the use of Adam considerably increases the detectability of the watermark. For the purpose of carrying out this analysis, we use FFDNet [14]—a DNN that performs image denoising tasks—as the host network.
- We introduce a novel method based on orthogonal projections to solve the detectability problem that arises when watermarking a DNN which is being optimized with Adam. A side effect of this novel method is an increased robustness against weight pruning.

The remainder of this paper is organized as follows: Section 1 introduces the notation and Section 2 explains the frameworks and algorithms used in this study—host network, optimization algorithms, and watermarking method—in more detail. Section 3 presents the mathematical core that allows us to model the observed effects on the histograms of weights once the embedding process has finished. Then, in Section 4, we introduce BOP as a solution for using Adam and watermarking simultaneously, Section 5 presents the information-theoretic measures that we will implement, and Section 6 shows the experimental results. Finally, we point out some concluding remarks in Section 7. Two appendices give additional details on mathematical derivations (Appendix A) and the validity of certain assumptions (Appendix B).

Notation

In this paper, we use the following notation. Matrices and vectors are denoted by upper-case and lower-case boldface characters, respectively, while random variables and their realizations are respectively represented by upper-case and lower-case characters.

For matrix and vector operations, we proceed as follows. As an example, let $\mathbf{A}$ be a matrix. Then, its transpose is denoted by $\mathbf{A}^T$. Moreover, we use $\mathrm{Tr}[\mathbf{A}]$ to represent the trace of $\mathbf{A}$ and $(\mathbf{A})_{i,j}$ to denote the (i,j)th element of $\mathbf{A}$. $\mathbf{I}_N$ refers to the $N \times N$ identity matrix. We use column vectors unless otherwise stated. In addition, we use $\mathbf{0}$ to denote a column vector of zeros and $\mathbf{1}$ for a column vector of ones. Let $\mathbf{w}$ be a column vector of length N; then, $\boldsymbol{\nabla}_{\mathbf{w}}$ is the gradient operator with respect to $\mathbf{w}$ that is:

$$\boldsymbol{\nabla}_{\mathbf{w}} \doteq \left(\frac{\partial}{\partial w_1}, \cdots, \frac{\partial}{\partial w_N}\right)^T$$

We use the operator $\circ$ to denote the Hadamard (i.e., sample-wise) product and $\otimes$ for the Kronecker product. Finally, $\mathrm{E}\{\cdot\}$ and $\mathrm{Var}\{\cdot\}$ denote the mathematical expectation and the variance, respectively.

2. Preliminaries

2.1. Host Network: FFDNet

The rapid development of Deep Learning over the last few years has led to new advances in the field of image restoration [15]. Several Convolutional Neural Networks (CNNs) have been designed to replace classical methods and, often, they offer new competitive advantages. This is the case of FFDNet [14], which performs image denoising tasks and is used in our work as the exemplary host network that is watermarked by means of the algorithm proposed in [5,6].

Image denoising is the task of removing noise from a given image. Let $\mathbf{y}$ be the input noisy image, $\mathbf{x}$ the clean image, and $\mathbf{n}$ the noise, which is usually modeled as zero-mean Additive White Gaussian Noise (AWGN), then we have $\mathbf{y} = \mathbf{x} + \mathbf{n}$ and we wish to obtain an estimate $\hat{\mathbf{x}}$ of the clean image. As opposed to other CNNs for image denoising, FFDNet works on downsampled sub-images, and it is able to adapt to several noise levels using only a single network. For that purpose, a noise level map $\mathbf{M}$ is included also as an input, so that the function FFDNet aims to learn can be expressed as: $\hat{\mathbf{x}} = \mathcal{F}(\mathbf{y}, \mathbf{M}; \mathbf{w})$, where $\mathbf{w}$ represents the parameters of the network. Furthermore, FFDNet can handle spatially variant noise and offers competitive inference speed without sacrificing denoising performance [14].

Figure 1 shows the architecture of FFDNet. As we can see, it is composed of a downscaling operation on the input images, a nonlinear mapping consisting of Convolutional Layers, Batch Normalization steps [16], and ReLU activation functions; and, finally, an upscaling process to generate denoised images with the original size. Let $\{(\mathbf{y}_i, \mathbf{x}_i)\}$ be L noisy-clean image pairs from the training dataset, the denoising cost function is:

$$f_0(\mathbf{w}) = \frac{1}{2L} \sum_{i=1}^{L} ||\mathcal{F}(\mathbf{y}_i, \mathbf{M}_i; \mathbf{w}) - \mathbf{x}_i||^2 \tag{1}$$

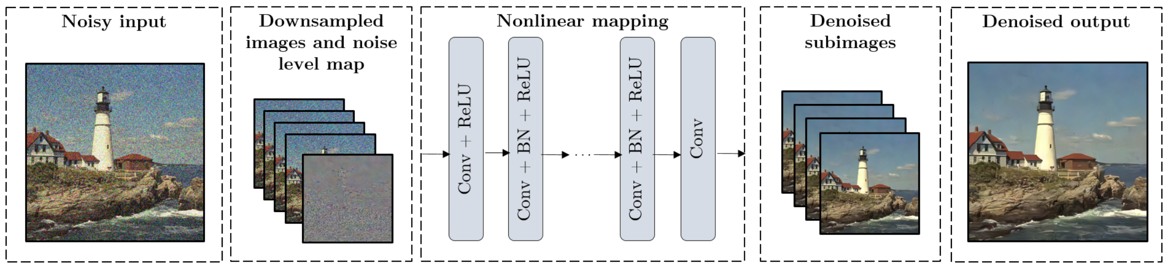


Figure 1. Architecture of the host network FFDNet.

FFDNet can be used for grayscale or color images. Table 1 shows the main differences between both configurations. As we can see, the total number of convolutional layers can be set to 15 or 12 for grayscale or RGB denoising, respectively. The number of feature maps and the size of the receptive field also differ, but this is not the case of the kernel size, which is kept to 3×3 for either grayscale or RGB. In this paper, we implement the RGB version of FFDNet and proceed as follows: (1) train FFDNet from scratch using Adam optimization without embedding any watermark and (2) fine-tune the network to embed the desired watermark, using both Adam and SGD to compare the results.

Table 1. FFDNet configurations for grayscale and RGB image denoising.

	Grayscale	**RGB**
Conv layers	15	12
Feature maps per layer	64	96
Receptive field	62×62	50×50

2.2. Optimization Algorithms

The training of a DNN is an iterative process that makes it possible for the model to learn how to perform a given task. This is certainly the most challenging optimization problem when implementing deep learning models from scratch. In order to increase the efficiency of the training process, researchers have developed several optimization techniques in the last years, each of them with its own advantages and drawbacks [17].

An optimization algorithm determines the weight update rule that must be applied at every iteration. The goal is usually to minimize a cost function—also known as loss function—which generally compares predictions with expected values and computes an error metric that evaluates the performance of the network. The learning rate μ—or step size—is the hyperparameter that controls how much the weights can change after each iteration of the optimization algorithm.

In the following sections, we briefly review the mechanics of two widely known optimization algorithms: SGD and Adam.

2.2.1. SGD Optimization

SGD [8] is a classic optimization algorithm based on gradient descent and one of the most used. However, unlike standard gradient descent techniques that use all the training samples to compute the gradient of the cost function, SGD uses a small number of samples from the dataset—a minibatch—and then takes the average over these samples to get an estimate of the gradient, $\nabla_{\mathbf{w}} f(\mathbf{w})$. Then, the update rule is given by:

$$\mathbf{w}^{(k)} = \mathbf{w}^{(k-1)} - \mu \nabla_{\mathbf{w}} f(\mathbf{w}^{(k-1)}), \ \ k \geq 1 \tag{2}$$

where $\mathbf{w}^{(k)}$ is the weight vector at iteration k and $\mathbf{w}^{(0)}$ is its initial value.

2.2.2. Adam Optimization

Adam [9] is a popular optimization algorithm that combines the ideas of AdaGrad [18] and RMSProp [19]. Some of the advantages of Adam include, among others, the fact of being easy to implement and computationally efficient, as well as being fast and suitable for complex settings with noisy and sparse gradients.

Just like SGD, Adam estimates the gradient from the samples of a minibatch, but, as opposed to SGD, it uses estimates of the first and second moments of the gradients to compute individual adaptive learning rates for each parameter in the network. In order to do that, exponential moving averages of the gradient and the squared gradient are calculated using two hyperparameters to control the decay rate, β_1 and β_2, respectively. Let $\mathbf{m}^{(0)} = \mathbf{0}$ and $\mathbf{v}^{(0)} = \mathbf{0}$ be the initial values for the first and second moment vectors, respectively, then the steps of this algorithm at the kth iteration are the following [9]:

$$\hat{\mathbf{g}}^{(k)} = \nabla_{\mathbf{w}} f(\mathbf{w}^{(k-1)}) \tag{3}$$

$$\mathbf{m}^{(k)} = \beta_1 \mathbf{m}^{(k-1)} + (1-\beta_1)\hat{\mathbf{g}}^{(k)} \tag{4}$$

$$\mathbf{v}^{(k)} = \beta_2 \mathbf{v}^{(k-1)} + (1-\beta_2)(\hat{\mathbf{g}}^{(k)} \circ \hat{\mathbf{g}}^{(k)}) \tag{5}$$

$$\hat{\mathbf{m}}^{(k)} = \frac{\mathbf{m}^{(k)}}{(1-\beta_1^k)} \tag{6}$$

$$\hat{\mathbf{v}}^{(k)} = \frac{\mathbf{v}^{(k)}}{(1-\beta_2^k)} \tag{7}$$

$$w_j^{(k)} = w_j^{(k-1)} - \mu \frac{\hat{m}_j^{(k)}}{\sqrt{\hat{v}_j^{(k)}} + \epsilon}, \qquad j = 1, \cdots, N. \tag{8}$$

where $w_j^{(k)}$, $\hat{m}_j^{(k)}$, $\hat{v}_j^{(k)}$ are the jth element of $\mathbf{w}^{(k)}$, $\hat{\mathbf{m}}^{(k)}$, $\hat{\mathbf{v}}^{(k)}$, respectively, $f(\mathbf{w})$ is the cost function and ϵ is a very small number that avoids diving by zero. The default set-up for the hyperparameters is [9]: $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1 \cdot 10^{-8}$.

2.3. Digital Watermarking Algorithm

In this paper, we analyze the digital watermarking algorithm proposed in [5,6]. As we mentioned previously, we employ the fine-tune-to-embed approach; therefore, the embedding function is applied only during some additional epochs after convergence is achieved for the original task—in this case, image denoising—.

2.3.1. Embedding Elements

We wish to embed a T-bit sequence, $\mathbf{b} \in \{0, 1\}^T$, into a certain layer l of the host DNN. For this specific layer, let (S, S), I and F represent the size of the convolution filter, the depth of input to the convolutional layer and the number of filters in that layer, respectively. Then, the weights can be represented by a tensor $\mathbf{W}_l$ with dimensions $S \times S \times I \times F$ and then rearranged to form a vector $\mathbf{w}_l$ of length $N = S^2 IF$.

One important point here is that, instead of directly using the weight vector $\mathbf{w}_l$, the authors in [5,6] suggest including an initial transformation of these coefficients. In order to reflect this, we must calculate the mean of $\mathbf{W}_l$ over the F kernels. As a result, we obtain a new flattened vector $\hat{\mathbf{w}}_l$ with M

elements, where $M = S^2 I$. This transformation can also be formulated if we introduce a new matrix $\mathbf{\Theta}$ of size $N \times M$:

$$\mathbf{\Theta} \doteq \mathbf{I}_M \otimes \mathbf{h}$$

so that $\hat{\mathbf{w}}_l = \mathbf{\Theta}^T \mathbf{w}_l$. Here, $\mathbf{h}$ is a column vector of length F with all of its elements set to $1/F$, i.e., $\mathbf{h}^T \mathbf{h} = 1/F$.

In order to move to the T-dimensional space, the authors in [5,6] introduce a secret projection matrix $\mathbf{\Phi}$. The size of this projection matrix—also referred to as regularizer parameter in [5,6]—is $M \times T$ so that each column corresponds to a particular projection vector $\boldsymbol{\phi}^{(i)}$, $i = 1, \cdots, T$.

For the purpose of the subsequent theoretical analysis, we pair up matrices $\mathbf{\Phi}$ and $\mathbf{\Theta}$ and, therefore, we ascribe the initial transformation to the projection matrix, preserving the notation with the original weight vector $\mathbf{w}_l$. To that end, we define a new $N \times T$ projection matrix:

$$\hat{\mathbf{\Phi}} = \mathbf{\Theta}\mathbf{\Phi} \tag{9}$$

so that now the projection vectors can be expressed as $\hat{\boldsymbol{\phi}}^{(i)} = \mathbf{\Theta}\boldsymbol{\phi}^{(i)}$.

2.3.2. Embedding Process

Now that we have introduced the basic elements, the embedding procedure can be described as follows [5,6]. Let $\mathbf{w}$ be a vector containing all the parameters of the network, then the watermarking regularizer, $f_{\text{watermark}}(\mathbf{w}_l)$, is added to the global cost function $f(\mathbf{w})$:

$$f(\mathbf{w}) = f_0(\mathbf{w}) + \lambda f_{\text{watermark}}(\mathbf{w}_l)$$

where $f_0(\mathbf{w})$ is the original cost function and λ is the regularization parameter. The regularizer term is a composition of two functions: cross-entropy and the sigmoid,

$$f_{\text{watermark}}(\mathbf{w}_l) \doteq -\sum_{i=1}^{T} \left(b_i \log(y_i) + (1 - b_i)\log(1 - y_i)\right) \tag{10}$$

with $y_i = 1/\left(1 + \exp(-(\hat{\boldsymbol{\phi}}^{(i)})^T \mathbf{w}_l)\right)$. In order to minimize (10), y_i will approximate to the value of b_i. Therefore, the sigmoid function will force each projection $(\hat{\boldsymbol{\phi}}^{(i)})^T \mathbf{w}_l$, $i = 1, \cdots, T$, to progressively move towards $+\infty$ and $-\infty$ depending on whether $b_i = 1$ or $b_i = 0$, respectively. For successfully embedding the secret message, it is generally enough to guarantee that each projection lies on the proper side of the horizontal axis. When this happens, we reach a Bit Error Rate (BER) of 0 and all the projected weights are aligned with their corresponding bits, that is, they are positive when the bit is 1 and, conversely, negative when the bit is 0.

2.3.3. Detectability Issues

In this paper, we employ the Random technique proposed by the authors, in which the values of the projection matrix $\mathbf{\Phi}$—before applying the transformation in (9)—are independent samples from a standard normal distribution $\mathcal{N}(0, 1)$. Their results [5,6] show that this Random approach is the most appealing design method for $\mathbf{\Phi}$ because, as they indicate, it does not significantly alter the distribution of weights at the embedding layer. However, there are some detectability issues here that should be considered.

On the one hand, the authors in [20] show that the standard deviation of the distribution of weights grows with the length of the embedded message. This information can be used by adversaries for detecting the watermark and even overwriting it.

On the other hand, one of the main conclusions of our work is that the presence or absence of alterations to the shape of the weight distributions is a consequence of the optimization algorithm used during the watermark embedding. In particular, the authors in [5,6] employ SGD with momentum in their experiments and the distributions of weights remain unchanged, yet the use of Adam would significantly alter the shape of the distributions even though we apply the same Random technique. In this paper, we will show that the results in [5,6] regarding the undetectability of the watermark do not hold when we use Adam optimization.

As an example to visualize this peculiar behavior shown by Adam when we employ the watermarking algorithm proposed in [5,6], we plot in Figure 2 the resulting histograms when $T = 256$ and $\lambda = 1$; specifically, the histograms of the weights before and after the embedding (corresponding to k = 32,140) and the histogram of the weight variations, respectively. As we can see, the distribution of the original weights has significantly changed, turning into a two-spiked shape that could be easily detected by an adversary. The complete set of histograms will be later shown in Section 6.

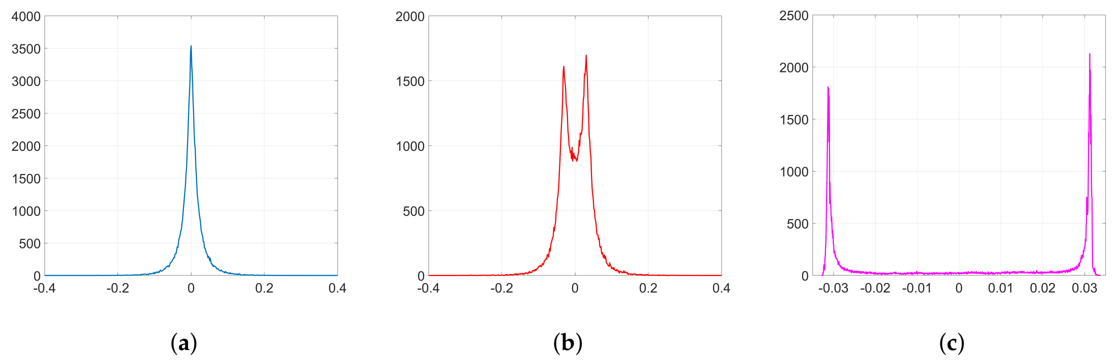


Figure 2. Histograms from the embedding layer $l = 2$ ($T = 256$, $\lambda = 1$ and k = 32,140). (**a**) histogram of $\mathbf{w}^{(0)}$; (**b**) histogram of $\mathbf{w}^{(k)}$; (**c**) histogram of $\Delta\mathbf{w} = \mathbf{w}^{(k)} - \mathbf{w}^{(0)}$.

2.3.4. Gaussian and Orthogonal Projection Vectors

In addition to the Random technique suggested by the authors of this watermarking algorithm—whose projection vectors will be referred to as Gaussian projection vectors in this paper—we will also implement orthonormal projectors. In order to build these kinds of projectors, we first generate the projection matrix following the Random technique; that is, samples are drawn from a standard normal distribution $\mathcal{N}(0,1)$. Then, from the Singular Value Decomposition of this projection matrix, we obtain an orthonormal basis for the column space so that we have $\mathbf{\Phi}^T\mathbf{\Phi} = \boldsymbol{I}_T$. Notice that once we apply the initial transformation (i.e., $\hat{\mathbf{\Phi}} = \mathbf{\Theta}\mathbf{\Phi}$) the resulting projection vectors $\hat{\boldsymbol{\phi}}^{(i)}$, $i = 1, \cdots, T$, will still preserve the orthogonality between them, although they will not be normalized:

$$(\hat{\mathbf{\Phi}})^T\hat{\mathbf{\Phi}} = (\mathbf{\Theta}\mathbf{\Phi})^T\mathbf{\Theta}\mathbf{\Phi} = \mathbf{\Phi}^T\mathbf{\Theta}^T\mathbf{\Theta}\mathbf{\Phi} = \mathbf{\Phi}^T(\boldsymbol{I}_M \otimes \mathbf{h}^T)(\boldsymbol{I}_M \otimes \mathbf{h})\mathbf{\Phi} = \frac{1}{F}\mathbf{\Phi}^T\mathbf{\Phi} = \frac{1}{F}\boldsymbol{I}_T$$

Therefore, these kinds of projectors will be referred to as orthogonal. As we will see later from KLD results and histograms, implementing orthogonal projectors may help us to better preserve both the original shape of the weight distribution and the denoising performance.

3. Theoretical Analysis

From now on, we will omit the sub-index l for the sake of clarity although we are always addressing the coefficients of the embedding layer. The experiments clearly illustrate that the use of Adam optimization together with the watermarking algorithm proposed in [5,6] originates noticeable changes in the distribution of weights, as we see in Figure 2. In the following analysis, we delve into the reasons why this happens. To that end, we aim to get a theoretical expression of $\Delta\mathbf{w} = \mathbf{w}^{(k)} - \mathbf{w}^{(0)}$.

This will allow us to prove and understand the nature of the observed behavior of the weights when watermark embedding is carried out. We start off by defining vector $\hat{\boldsymbol{\varphi}}$ and matrix $\hat{\boldsymbol{\Upsilon}}$:

$$\hat{\boldsymbol{\varphi}} \doteq \sum_{i=1}^{T} \hat{\boldsymbol{\phi}}^{(i)} \tag{11}$$

$$\hat{\boldsymbol{\Upsilon}} \doteq \sum_{i=1}^{T} \hat{\boldsymbol{\phi}}^{(i)} (\hat{\boldsymbol{\phi}}^{(i)})^T \tag{12}$$

Notice that $\hat{\boldsymbol{\Upsilon}} = \hat{\boldsymbol{\Upsilon}}^T$. In addition, for the case of orthogonal projectors, the following properties can be straightforwardly proven:

$$\hat{\boldsymbol{\Upsilon}}\hat{\boldsymbol{\varphi}} = \frac{1}{F}\hat{\boldsymbol{\varphi}} \tag{13}$$

$$\hat{\boldsymbol{\Upsilon}}\hat{\boldsymbol{\Upsilon}} = \frac{1}{F}\hat{\boldsymbol{\Upsilon}} \tag{14}$$

These properties will come in handy later on in several theoretical derivations.

Firstly, in order to simplify the analysis and understand more clearly how the watermarking cost function impacts on the movement of the weights when using both SGD and Adam optimization, we will just consider the presence of the regularization term, that is, we will not include the denoising cost function for now. The influence of the denoising part will be studied in Section 3.3. Therefore, given our embedding cost function in (10) and assuming for simplicity (and without loss of generality) that all embedded symbols are $+1$, we have:

$$\tilde{f}(\mathbf{w}) \doteq \lambda f_{\mathsf{watermark}}(\mathbf{w}|\mathbf{b}=\mathbf{1}) = \lambda \sum_{i=1}^{T} \log(1+\exp(-(\hat{\boldsymbol{\phi}}^{(i)})^T\mathbf{w}))$$

If we compute the gradient of this function, we obtain:

$$\boldsymbol{\nabla}_{\mathbf{w}}\tilde{f}(\mathbf{w}) = -\lambda \sum_{i=1}^{T} \hat{\boldsymbol{\phi}}^{(i)} \frac{1}{1+\exp\left((\hat{\boldsymbol{\phi}}^{(i)})^T\mathbf{w}\right)} \tag{15}$$

In order to simplify the subsequent analysis, we introduce a series of assumptions which are based on empirical observations or hypotheses that will be duly verified.

By construction, it is possible to show that the mean of $(\hat{\boldsymbol{\phi}}^{(i)})^T\mathbf{w}^{(0)}$ is zero and its variance is $(N/F^2)\mathrm{Var}\{\mathbf{w}^{(0)}\}$ for Gaussian projectors and $(1/F)\mathrm{Var}\{\mathbf{w}^{(0)}\}$ for orthogonal projectors (see Appendix A.1). Since the variance of the weights at the initial iteration is generally very small—in our experiments, it is 0.0012—it can be considered that the variance of $(\hat{\boldsymbol{\phi}}^{(i)})^T\mathbf{w}^{(0)}$ will also be small enough so that we can assume $|(\hat{\boldsymbol{\phi}}^{(i)})^T\mathbf{w}| \ll 1$ for all $i = 1, \cdots, T$. Although this assumption might not be strictly true for all k—especially once we have crossed the linear region of the sigmoid function—it is reasonably good and it allows us to use a first-order Taylor expansion for $\left(1+\exp\left((\hat{\boldsymbol{\phi}}^{(i)})^T\mathbf{w}\right)\right)^{-1}$ around $(\hat{\boldsymbol{\phi}}^{(i)})^T\mathbf{w} = 0$:

$$\left(1+\exp\left((\hat{\boldsymbol{\phi}}^{(i)})^T\mathbf{w}\right)\right)^{-1} \approx \frac{1}{2} - \frac{(\hat{\boldsymbol{\phi}}^{(i)})^T\mathbf{w}}{4} \tag{16}$$

Plugging (16) into (15) and using the definitions given above, we can write:

$$\boldsymbol{\nabla}_{\mathbf{w}}\tilde{f}(\mathbf{w}) \approx -\frac{\lambda}{2}\hat{\boldsymbol{\varphi}} + \frac{\lambda}{4}\hat{\boldsymbol{\Upsilon}}\mathbf{w} \tag{17}$$

We introduce now one important hypothesis in this theoretical analysis to handle the previous equation: we assume that $\mathbf{w}^{(k)}$ grows approximately affinely with k:

$$\mathbf{w}^{(k)} \approx \mathbf{w}^{(0)} + k \cdot \mu \cdot \boldsymbol{\eta} \tag{18}$$

where $\boldsymbol{\eta}$ is a vector that contains the slopes for each weight, and it is to be determined in the following sections. We hypothesize this affine-like growth for the weights and, later, we will verify that this is consistent with the rest of the theory and the experiments (see Appendix B.1 for more details). Therefore, we can write the weight variations as:

$$\Delta\mathbf{w} = \mathbf{w}^{(k)} - \mathbf{w}^{(0)} \approx k \cdot \mu \cdot \boldsymbol{\eta} \tag{19}$$

3.1. Analysis for SGD

We first analyze the behavior of SGD optimization when we implement digital watermarking as proposed in [5,6]. Recall the SGD update rule in (2). If we use the approximation for the gradient in (17) and the affine growth hypothesis for the weights introduced in (18), we have:

$$\boldsymbol{\nabla}_{\mathbf{w}} \tilde{f}(\mathbf{w}) \approx -\frac{\lambda}{2}\hat{\boldsymbol{\varphi}} + \frac{\lambda}{4}\hat{\boldsymbol{\Upsilon}}(\mathbf{w}^{(0)} + k\mu\boldsymbol{\eta}) \tag{20}$$

To simplify the analysis, we consider from now on that $\mathbf{w}^{(0)} \approx \mathbf{0}$. We confirm the validity of this assumption in Appendix B.2. Then, we can write:

$$\boldsymbol{\eta} = -\boldsymbol{\nabla}_{\mathbf{w}} \tilde{f}(\mathbf{w}) \approx \frac{\lambda}{2}\hat{\boldsymbol{\varphi}} - \frac{\lambda k\mu}{4}\hat{\boldsymbol{\Upsilon}}\boldsymbol{\eta} \tag{21}$$

If we consider orthogonal projectors, we can arrive at a more explicit expression for $\boldsymbol{\eta}$. In particular, if we multiply (21) by $\hat{\boldsymbol{\Upsilon}}$ and use the properties (13) and (14), we obtain:

$$\hat{\boldsymbol{\Upsilon}}\boldsymbol{\eta} = \frac{2\lambda}{4F - \lambda k\mu}\hat{\boldsymbol{\varphi}} \tag{22}$$

Then, substituting (22) into (21), we can get a more concise expression for $\boldsymbol{\eta}$:

$$\boldsymbol{\eta} \approx \frac{\lambda}{2}\left(1 - \frac{\lambda k\mu}{4F - \lambda k\mu}\right)\hat{\boldsymbol{\varphi}} \tag{23}$$

Thus, if F is large compared to $\lambda k\mu$—this certainly holds for our experimental set-up, cf. Section 6.1—$\boldsymbol{\eta}$ will be approximately proportional to $\hat{\boldsymbol{\varphi}}$. Then, the coefficients will follow an affine-like growth as we hypothesized in (18) (see Appendix B.1 for the empirical confirmation of this hypothesis). Now, the weight variations can be expressed as:

$$\Delta\mathbf{w} \approx k \cdot \mu \cdot \boldsymbol{\eta} = \frac{\lambda k\mu}{2}\left(1 - \frac{\lambda k\mu}{4F - \lambda k\mu}\right)\hat{\boldsymbol{\varphi}} \tag{24}$$

As we can see, when we use SGD, $\Delta\mathbf{w}$ will approximately follow a zero-mean Gaussian distribution, as induced by [9]. Because of this, and unlike Adam (as we will see later), the weights will evolve with random speeds when we embed watermarks using SGD optimization. Therefore, the impact on the original shape of the weight distribution will be small. However, the variance of the weight distribution may change considerably as stated in [20]. Since we have $\mathrm{Var}\{\hat{\boldsymbol{\varphi}}\} = T/(FN)$ for orthogonal projectors, the variance of $\Delta\mathbf{w}$ can be computed as:

$$\mathrm{Var}\{\Delta\mathbf{w}\} = \left[\lambda k\mu\frac{(2F - \lambda k\mu)}{(4F - \lambda k\mu)}\right]^2 \frac{T}{FN}$$

Thus, considering that $\mathbf{w}^{(0)}$ and $\Delta\mathbf{w}$ are uncorrelated—we check this statement in Section 6.2.2—we arrive at the following expression for the variance of the weights at the *k*th iteration:

$$\mathrm{Var}\{\mathbf{w}^{(k)}\} = \mathrm{Var}\{\mathbf{w}^{(0)}\} + \left[\lambda k\mu \frac{(2F - \lambda k\mu)}{(4F - \lambda k\mu)}\right]^2 \frac{T}{FN} \tag{25}$$

As we can see from (25), when implementing the digital watermarking algorithm in [5,6] with SGD optimization and orthogonal projectors, the variance of the resulting weight distribution might change considerably. In order to preserve the original weight distribution when using SGD, it is important to take care with the values of T, F and N, especially. In addition, the standard deviation of the weights will (approximately) increase linearly with the number of iterations so it may be also important to limit the value of k. This is in line with the expected behavior: the weights will move away from their original value and they will be further if we perform more iterations.

Because the analysis for Gaussian projectors becomes considerably difficult, in this paper, we just address the study of SGD with orthogonal projectors. A more comprehensive analysis for Gaussian projectors that can be linked to the results obtained in [20] is left for future research. Regardless of this, the whole analysis for both kinds of projection vectors will be developed in the next section for Adam optimization.

3.2. *Analysis for Adam*

In the next sections, we will delve into the theory behind Adam optimization for DNN watermarking. In particular, we will obtain an expression for the mean and the variance of the gradient and then, as we did with SGD, we will analyze the update term to get an expression of the weight variations.

3.2.1. Mean of the Gradient

We are interested in computing the mean of the gradient that is used in Adam. Considering $\tilde{f}(\mathbf{w})$ as the global cost function, then, from (4), we can rewrite the mean at the *k*th iteration as:

$$\mathbf{m}^{(k)} = (1 - \beta_1) \sum_{i=1}^{k} \beta_1^{k-i} \nabla_{\mathbf{w}} \tilde{f}(\mathbf{w}^{(i)}) \tag{26}$$

We use the gradient in (17) and do some derivations to find an explicit expression for $\hat{\mathbf{m}}^{(k)} = \mathbf{m}^{(k)}/(1-\beta_1^k)$ under the hypothesis in (18). Finally, we arrive at the following expression for the bias-corrected mean gradient when k is sufficiently large (see Appendix A.2 for all the mathematical details):

$$\begin{aligned} \hat{\mathbf{m}}^{(k)} &\approx -\frac{\lambda}{2}\hat{\boldsymbol{\varphi}} + \frac{\lambda}{4} \cdot \hat{\boldsymbol{\Upsilon}}(\mathbf{w}^{(0)} + k\mu\boldsymbol{\eta}) \\ &= \frac{\lambda}{2}(\mathbf{m}_0 + \mu \cdot k \cdot \mathbf{m}_1), \;\; k \gg \frac{\beta_1}{1-\beta_1} \end{aligned} \tag{27}$$

where $\mathbf{m}_0 \doteq -\hat{\boldsymbol{\varphi}} + \frac{1}{2}\hat{\boldsymbol{\Upsilon}}\mathbf{w}^{(0)}$ and $\mathbf{m}_1 \doteq \frac{1}{2}\hat{\boldsymbol{\Upsilon}}\boldsymbol{\eta}$. As we see from (27), the mean of the gradient also grows affinely with k.

3.2.2. Variance of the Gradient

Let $\mathbf{g}(\mathbf{w}) \doteq \nabla_{\mathbf{w}}\tilde{f}(\mathbf{w}) \circ \nabla_{\mathbf{w}}\tilde{f}(\mathbf{w})$. The approximation in (17) for the *j*th element of this vector $\mathbf{g}(\mathbf{w})$, denoted by $g_j(\mathbf{w})$, is the following:

$$\begin{aligned} g_j(\mathbf{w}) &\approx \frac{\lambda^2}{4}\sum_{m=1}^{T}\sum_{l=1}^{T}\hat{\varphi}_j^{(m)}\hat{\varphi}_j^{(l)}\left(1-(\hat{\boldsymbol{\varphi}}^{(m)})^T\mathbf{w}+\frac{1}{4}(\hat{\boldsymbol{\varphi}}^{(m)})^T\mathbf{w}\mathbf{w}^T\hat{\boldsymbol{\varphi}}^{(l)}\right) \\ &= \frac{\lambda^2}{4}\left(a_j-\mathbf{b}_j^T\mathbf{w}+\frac{1}{4}\mathbf{c}_j^T\mathbf{w}\mathbf{w}^T\mathbf{c}_j\right) \end{aligned}$$

where:

$$a_j \doteq \hat{\varphi}_j^2 \tag{28}$$

$$\mathbf{c}_j \doteq \sum_{m=1}^{T}\hat{\varphi}_j^{(m)}\hat{\boldsymbol{\varphi}}^{(m)} \tag{29}$$

$$\mathbf{b}_j \doteq \hat{\varphi}_j\cdot\mathbf{c}_j \tag{30}$$

Following the hypothesis in (18), we can write:

$$g_j(\mathbf{w}^{(k)}) \approx \frac{\lambda^2}{4}\left(a_j-\mathbf{b}_j^T\mathbf{w}^{(0)}+\frac{1}{4}\mathbf{c}_j^T\mathbf{w}^{(0)}(\mathbf{w}^{(0)})^T\mathbf{c}_j-k\mu\mathbf{b}_j^T\boldsymbol{\eta}+\frac{1}{2}k\mu\mathbf{c}_j^T\mathbf{w}^{(0)}\boldsymbol{\eta}^T\mathbf{c}_j+\frac{1}{4}k^2\mu^2\mathbf{c}_j^T\boldsymbol{\eta}\boldsymbol{\eta}^T\mathbf{c}_j\right)$$

In summary, for this affine-like growth, the square gradient vector can be written as:

$$g_j(\mathbf{w}^{(k)}) \approx \frac{\lambda^2}{4}(p_j+q_jk\mu+r_jk^2\mu^2) \tag{31}$$

for some vectors $\mathbf{p},\mathbf{q},\mathbf{r}$ whose jth component can be defined as:

$$p_j \doteq a_j-\mathbf{b}_j^T\mathbf{w}^{(0)}+\frac{1}{4}\mathbf{c}_j^T\mathbf{w}^{(0)}(\mathbf{w}^{(0)})^T\mathbf{c}_j \tag{32}$$

$$q_j \doteq -\mathbf{b}_j^T\boldsymbol{\eta}+\frac{1}{2}\mathbf{c}_j^T\mathbf{w}^{(0)}\boldsymbol{\eta}^T\mathbf{c}_j$$

$$r_j \doteq \frac{1}{4}\mathbf{c}_j^T\boldsymbol{\eta}\boldsymbol{\eta}^T\mathbf{c}_j \tag{33}$$

Now, from (5), we can rewrite the variance of the gradient that is used in Adam as:

$$v_j^{(k)} = (1-\beta_2)\sum_{i=1}^{k}\beta_2^{k-i}g_j(\mathbf{w}^{(i)}) \tag{34}$$

The bias-corrected term $\hat{v}_j^{(k)}$ is obtained after dividing $v_j^{(k)}$ by $(1-\beta_2^k)$. Applied to the special case of (31), this yields (see Appendix A.3):

$$\begin{aligned} \hat{v}_j^{(k)} &= \frac{\lambda^2}{4}\left(p_j-\frac{\beta_2}{1-\beta_2}\mu q_j+\frac{\beta_2+\beta_2^2}{(1-\beta_2)^2}\mu^2r_j\right) \\ &+ \frac{\lambda^2}{4(1-\beta_2^k)}k\left(\mu q_j-\frac{2\beta_2}{1-\beta_2}\mu^2r_j\right)+\frac{\lambda^2}{4(1-\beta_2^k)}k^2\mu^2r_j \end{aligned}$$

3.2.3. Update Term

Because μ is usually very small—we use $\mu = 1\cdot10^{-6}$ in our experiments—we can assume that $k\mu$ will be small enough to obtain an approximation of the update used in Adam. Recall that, for the jth weight, this is $u_j^{(k)} \doteq \hat{m}_j^{(k)}/\left(\sqrt{\hat{v}_j^{(k)}}+\epsilon\right)$, implying that $w_j^{(k)} = w_j^{(k-1)}-\mu u_j^{(k)}$. Let:

$$s_j \doteq \left(p_j-\frac{\beta_2}{1-\beta_2}\mu q_j+\frac{\beta_2+\beta_2^2}{(1-\beta_2)^2}\mu^2r_j\right) \tag{35}$$

Then, assuming that $\mu k \ll 1$, we can make a zero-order approximation of the update term, i.e.,:

$$u_j^{(k)} = \frac{\hat{m}_j^{(k)}}{\sqrt{\hat{v}_j^{(k)}} + \epsilon} \approx \frac{m_{0,j}}{\sqrt{s_j}}, \quad \mu k \ll 1 \tag{36}$$

This approximation is accurate enough for the set of experiments we perform. In particular, for the orthogonal case, we could deal with k_{max} = 625,000 and still get a correlation coefficient of 0.9900 between $(\lambda^2/4)s_j$ and $\hat{v}_j^{(k_{max})}$. In our experiments, we actually reach a BER of zero for values of k quite below k_{max} (cf. Section 6.2).

From (36), we observe that the updated jth coefficient approximately follows the hypothesized growth, i.e., $\mathbf{w}^{(k)} = \mathbf{w}^{(0)} + k \cdot \mu \cdot \boldsymbol{\eta}$, where $\eta_j = -m_{0,j}/\sqrt{s_j}$. Notice that, as expected, the update does not depend on λ, following Adam's property that the update is invariant to rescaling the gradients [9]. Finding a more explicit expression runs into the problem that $\boldsymbol{\eta}$ depends on $\mathbf{s}$, which in turn is a function of $\boldsymbol{\eta}$ through (32) and (33). The following subsections are devoted to solving this problem by conjecturing a form for $\boldsymbol{\eta}$ and refining it.

To simplify the analysis, we consider from now on that $\mathbf{w}^{(0)} \approx \mathbf{0}$ since most of the values of the weights at the initial iteration are very small (see Figure 2a). We will verify the accuracy of this approximation in Appendix B.2.

3.2.4. Rationale for the Sign Function

Recall the expression (23) that we obtained for $\boldsymbol{\eta}$ when analyzing SGD, where we found $\boldsymbol{\eta}$ to be approximately proportional to $\hat{\boldsymbol{\varphi}}$. Now, for Adam, we take this as a starting point, so we conjecture first that $\boldsymbol{\eta} = \gamma \cdot \hat{\boldsymbol{\varphi}}$, for some real positive γ. Here, we consider orthogonal projection vectors and use the property introduced in (13) and the following:

$$\mathbf{c}_j^T \hat{\boldsymbol{\varphi}} = \frac{\hat{\varphi}_j}{F}$$

In this particular case, we have the following identities:

$$\begin{aligned} m_{0,j} &\approx -\hat{\varphi}_j \\ p_j &\approx a_j = \hat{\varphi}_j^2 \\ q_j &\approx -\frac{\gamma}{F}\hat{\varphi}_j^2 \\ r_j &= \frac{\gamma^2}{4F^2}\hat{\varphi}_j^2 \end{aligned}$$

Substituting these values into (35), we find that

$$\sqrt{s_j} = \text{sgn}(\hat{\varphi}_j) \cdot \hat{\varphi}_j \cdot \sqrt{1 + (\gamma/F)\mu\beta_2/(1-\beta_2) + (\gamma^2/(4F^2))\mu^2(\beta_2+\beta_2^2)/(1-\beta_2)^2}$$

When we divide $m_{0,j}$ by $\sqrt{s_j}$, we obtain:

$$u_j^{(k)} \approx \frac{-\text{sgn}(\hat{\varphi}_j)}{\sqrt{1 + (\gamma/F)\mu\beta_2/(1-\beta_2) + (\gamma^2/(4F^2))\mu^2(\beta_2+\beta_2^2)/(1-\beta_2)^2}} \tag{37}$$

It is then clear that $\boldsymbol{\eta}$ cannot be written in the form $\boldsymbol{\eta} = \gamma \cdot \hat{\boldsymbol{\varphi}}$, as was conjectured at the beginning of this section.

3.2.5. A Theoretical Expression for $\Delta\mathbf{w}$

Although the conjectured form for $\boldsymbol{\eta}$ in Section 3.2.4 does not hold, the appearance of the sign function in (37) gives a key clue for an alternative approach, since the sign seems to reveal the reason behind the two-spiked histograms like the one shown in Figure 2c.

Therefore, let us write η_j to explicitly contain the sign of φ_j and allow γ_j to take different (non-negative) values with j to reflect the varying magnitude (recall that even in Section 3.2.4 the conjectured value could be written as $\eta_j = \gamma|\varphi_j| \cdot \mathrm{sgn}(\varphi_j)$). Let $\boldsymbol{\gamma}$ be the column vector containing γ_j, $j = 1, \cdots, N$, then $\boldsymbol{\eta} = \boldsymbol{\gamma} \circ \mathrm{sgn}(\hat{\boldsymbol{\varphi}})$. Since $\mathbf{c}_j^T(\boldsymbol{\gamma} \circ \mathrm{sgn}(\hat{\boldsymbol{\varphi}})) = \boldsymbol{\gamma}^T(\mathbf{c}_j \circ \mathrm{sgn}(\hat{\boldsymbol{\varphi}}))$, we can write:

$$\begin{aligned} m_{0,j} &\approx -\hat{\varphi}_j \\ p_j &\approx a_j = \hat{\varphi}_j^2 \\ q_j &\approx -\mathbf{b}_j^T\boldsymbol{\eta} = -\hat{\varphi}_j\boldsymbol{\gamma}^T(\mathbf{c}_j \circ \mathrm{sgn}(\hat{\boldsymbol{\varphi}})) \\ r_j &= \frac{1}{4}\left[\boldsymbol{\gamma}^T(\mathbf{c}_j \circ \mathrm{sgn}(\hat{\boldsymbol{\varphi}}))\right]^2 \end{aligned}$$

Now,

$$u_j^{(k)} \approx \frac{-\hat{\varphi}_j}{\sqrt{\hat{\varphi}_j^2 + \mu\hat{\varphi}_j\boldsymbol{\gamma}^T\left(\mathbf{c}_j \circ \mathrm{sgn}(\hat{\boldsymbol{\varphi}})\right)\frac{\beta_2}{(1-\beta_2)} + \frac{\mu^2}{4}\left[\boldsymbol{\gamma}^T(\mathbf{c}_j \circ \mathrm{sgn}(\hat{\boldsymbol{\varphi}}))\right]^2\frac{\beta_2+\beta_2^2}{(1-\beta_2)^2}}}$$

In addition, thus, in order to meet the condition $\eta_j = \gamma_j \cdot \mathrm{sgn}(\hat{\varphi}_j)$, the following nonlinear equation should be solved for all γ_j, $j = 1, \cdots, N$:

$$\gamma_j = \frac{|\hat{\varphi}_j|}{\sqrt{\hat{\varphi}_j^2 + \mu\hat{\varphi}_j\boldsymbol{\gamma}^T\left(\mathbf{c}_j \circ \mathrm{sgn}(\hat{\boldsymbol{\varphi}})\right)\frac{\beta_2}{(1-\beta_2)} + \frac{\mu^2}{4}\left[\boldsymbol{\gamma}^T(\mathbf{c}_j \circ \mathrm{sgn}(\hat{\boldsymbol{\varphi}}))\right]^2\frac{\beta_2+\beta_2^2}{(1-\beta_2)^2}}} \tag{38}$$

This equation can be solved with a fixed-point iteration method [21]. To that end, we should initialize $\boldsymbol{\gamma}$ and then iterate the following: (1) compute the right-hand side of (38), and (2) use it to update $\boldsymbol{\gamma}$ on the left-hand side. This process will converge to the solution of (38). Even though this method can be implemented to give the specific values for each γ_j, we are more interested in obtaining a statistical characterization rather than a deterministic one. As we will see, the statistical approach offers a deeper explanation for the two-spiked distribution of $\Delta\mathbf{w}$ which we ultimately seek.

We thus aim at finding the pdf of Γ, now considered as a random variable for which γ_j, $j = 1, \cdots, N$ are nothing but realizations. Once again, Equation (38) can be solved iteratively (e.g., with Markov-chain Monte Carlo methods [22]) to yield the equilibrium distribution for Γ. Instead, we can resort to the results in Section 6 where we conclude that the pdf of Γ is strongly concentrated around its mode. With this observation, it is possible to consider that $\boldsymbol{\gamma}^T(\mathbf{c}_j \circ \mathrm{sgn}(\hat{\boldsymbol{\varphi}}))$ approximately corresponds to realizations of $\Gamma \cdot (\mathbf{c}_j^T\mathrm{sgn}(\hat{\boldsymbol{\varphi}}))$.

In order to simplify the analysis even further, we are interested in decomposing $\mathbf{c}_j^T\mathrm{sgn}(\hat{\boldsymbol{\varphi}})$ using its statistical projection onto $\hat{\varphi}_j$, i.e., $\mathbf{c}_j^T\mathrm{sgn}(\hat{\boldsymbol{\varphi}}) = \alpha \cdot \hat{\varphi}_j + z_j$. Here, α is a real multiplier and z_j is zero-mean noise uncorrelated with $\hat{\varphi}_j$. More generally, if we define matrix $\mathbf{C} \doteq [\mathbf{c}_0, \cdots, \mathbf{c}_N]$, then we seek to write $\mathbf{C}^T\mathrm{sgn}(\hat{\boldsymbol{\varphi}}) = \alpha \cdot \hat{\boldsymbol{\varphi}} + \mathbf{z}$. We do the analysis for the cases of Gaussian projectors and orthogonal projectors separately (refer to Appendix A.4 for the derivations). For the Gaussian case, we get:

$$\alpha = \frac{1}{FT}\sqrt{\frac{2}{\pi T}}\left(FT^2 + (N+F)T - F\right) \tag{39}$$

$$\mathrm{Var}(z_j) = -\frac{2}{\pi F^3 T^2}\left(FT^4 + 4FT^3 + (N+F)T^2 - 4FT + F\right) + \frac{T}{F^3}(N + F(T+1)) \tag{40}$$

On the other hand, for the orthogonal projectors, we get instead:

$$\alpha = \sqrt{\frac{2N}{\pi T F}} \tag{41}$$

$$\mathrm{Var}(z_j) = \left(1 - \frac{2}{\pi}\right) \frac{T}{FN} \tag{42}$$

Recall that, by construction, $\hat{\boldsymbol{\varphi}}$ can be seen as a random vector. In fact, we have $\hat{\boldsymbol{\varphi}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I} \cdot T/F^2)$ for Gaussian projection vectors, and $\hat{\boldsymbol{\varphi}}$ approximately follows $\mathcal{N}(\mathbf{0}, \mathbf{I} \cdot T/(FN))$ for orthogonal projectors. Let Ξ, Z be random variables with the distribution of a single element of $\hat{\boldsymbol{\varphi}}$ and $\mathbf{z}$, respectively, then q_j and r_j can be seen as realizations of (approximately): $Q = -\Gamma\Xi(\alpha\Xi + Z)$ and $R = \frac{\Gamma^2}{4}(\alpha\Xi + Z)^2$, so a stochastic version of (38) is:

$$\Gamma = \frac{|\Xi|}{\sqrt{\Xi^2 + \Gamma\mu\Xi\,(\alpha\Xi + Z)\,\frac{\beta_2}{(1-\beta_2)} + \frac{\Gamma^2\mu^2}{4}\,(\alpha\Xi + Z)^2\,\frac{\beta_2+\beta_2^2}{(1-\beta_2)^2}}}$$

Squaring both sides, we find that, for a given realization (ξ, z) of (Ξ, Z), Γ must take the positive value γ that satisfies the following fourth degree equation:

$$\frac{\beta_2 + \beta_2^2}{(1-\beta_2)^2}\,\frac{\mu^2}{4}\,(\alpha\xi + z)^2\,\gamma^4 + \frac{\beta_2}{1-\beta_2}\mu\xi\,(\alpha\xi + z)\,\gamma^3 + \xi^2\gamma^2 - \xi^2 = 0 \tag{43}$$

From (43), it is easy to generate samples γ of Γ and, accordingly, samples of $\Delta\mathbf{w}$, by recalling that:

$$\Delta w \approx k \cdot \mu \cdot \gamma \cdot \mathrm{sgn}(\xi) \tag{44}$$

We note that, for the particular case when β_2 is very close to 1, $\frac{\beta_2+\beta_2^2}{(1-\beta_2)^2} \approx 2\left(\frac{\beta_2}{1-\beta_2}\right)^2$. This simplification allows us to approximate (43) as

$$\gamma^2\left(\frac{\gamma\mu}{2} \cdot \frac{\beta_2}{1-\beta_2}(\alpha\xi + z) + \xi\right)^2 - \xi^2 \approx 0 \tag{45}$$

which leads to the following fixed-point equation:

$$\gamma \approx \frac{\xi}{\frac{\gamma\mu}{2} \cdot \frac{\beta_2}{1-\beta_2}(\alpha\xi + z) + \xi} \tag{46}$$

When the noise term z is very small compared to $\alpha\xi$ (which occurs with a fairly large probability, especially for the case of orthogonal projectors), then the solution to (46), denoted by γ_s, will be independent of the value of ξ. This will cause the probability of Γ to be concentrated around γ_s, and in turn this will make the pdf Δw have two spikes centered at $\pm k\mu\gamma_s$. We will see these spikes appearing time and again in the experiments carried out with Adam (Section 6).

3.3. The Denoising Term

Thus far, we have considered only that our cost function is $\tilde{f}(\mathbf{w}) = \lambda f_{\mathsf{watermark}}(\mathbf{w})$; however, as we know, there is an additional term, the original denoising function, so our real cost function is: $f(\mathbf{w}) = f_0(\mathbf{w}) + \tilde{f}(\mathbf{w})$.

The gradients corresponding to this function, $f_0(\mathbf{w})$, will try to pull the weight vector towards the original optimal $\mathbf{w}^{(0)}$ in a relatively hard to model way. In order to analyze this behavior, we can approximate the gradient of the denoising function at the kth iteration with respect to the jth coefficient as a sum of a constant term, d_j, and a noisy one, $\tilde{n}_j^{(k)}$, which follows a zero-mean Gaussian distribution and is associated with the use of different training batches on each step. We will refer to this noise as batching noise. Thus, for each coefficient j, we can write:

$$\frac{\partial f_0(\mathbf{w}^{(k)})}{\partial w_j} \approx d_j + \tilde{n}_j^{(k)} \tag{47}$$

Like we did in the previous section, we can formulate a stochastic version of (47). To that end, we notice that the constant term of this gradient, d_j, can take different values with j, as well as the variance of the batching noise, h_j that is, $\tilde{n}_j$ is drawn from $\mathcal{N}(\mathbf{0}, h_j)$. Therefore, in order to reflect the variability of these terms along the j-elements, we introduce two random variables with the distribution of the mean gradient and the variance of the batching noise, D and H, respectively, for which d_j and h_j are realizations. The pdf of these distributions will be obtained empirically in Section 6.2. Then, we can see $\tilde{n}_j^{(k)}$ as a realization of $\tilde{N} \sim \mathcal{N}(0, H)$.

3.3.1. SGD

Similarly to Section 3.2.5, let Ξ be a random variable with the distribution of $\hat{\boldsymbol{\varphi}}$. Let $(\xi, \delta, \tilde{n})$ be a realization of $(\Xi, D, \tilde{N})$, respectively, then for SGD using orthogonal projectors we can compute samples of $\Delta\mathbf{w}$ adding both functions, i.e., denoising and watermarking:

$$\Delta w \approx k\mu(\delta + \tilde{n}) + \frac{\lambda k \mu}{2}\left(1 - \frac{\lambda k \mu}{4F - \lambda k \mu}\right)\xi \tag{48}$$

3.3.2. Adam

The variance of the batching noise computed by Adam will be approximately given by the random variable V, whose realizations can be expressed as $v_j = \frac{1-\beta_2}{1-\beta_2^k} \sum_{i=1}^{k} \beta_2^{(k-i)} (\tilde{n}_j^{(i)})^2$. Notice that, for each realization of V, as the sum takes places over i, we must work with a fixed value h_j for the variance of the batching noise. Then, with this variance, we generate k samples of $\tilde{N}$ to be used in the sum that produces v_j. With this characterization, we can easily analyze how the denoising cost function shapes the distribution of the weight variations. Notice that this analysis could be adapted for any host network. Let δ and ν be realizations of D and V, respectively, then, we can generate samples of $\Delta\mathbf{w}$ without including the gradients from the watermarking function as:

$$\Delta w \approx k \cdot \mu \cdot \frac{\delta}{\sqrt{\delta^2 + \nu}}$$

Moreover, in order to get a more accurate description of the problem, we can combine both functions: denoising and watermarking. The analysis becomes somewhat complicated, but, as we will check in Section 6, the distributions resulting from this analysis do capture better the shapes observed in the empirical ones. See Appendix A.5 for the results of this analysis.

4. Block-Orthonormal Projections (BOP)

Here, we discuss BOP, the solution we propose to solve the detectability problem posed by Adam optimization when implementing the watermarking algorithm proposed in [5,6]. In order to hide the noticeable weight variations that appear when we use Adam—as seen in Figure 2—we introduce a prior transformation using a secret $N \times N$ matrix $\mathbf{X}$ (the details for its construction are given below). The procedure we follow has three steps per each iteration of Adam.

Firstly, we project the weights and gradients from the embedding layer using $\mathbf{X}$:

$$\begin{aligned} \mathbf{y} &= \mathbf{X}\mathbf{w} \\ \nabla_{\mathbf{y}} f(\mathbf{y}) &= \mathbf{X}\nabla_{\mathbf{w}} f(\mathbf{w}) \end{aligned}$$

Then, we run Adam optimization on the projected weights, $\mathbf{y}$, using the projected gradients, $\boldsymbol{\nabla}_{\mathbf{y}} f(\mathbf{y})$, as well, i.e., steps (3)–(8) are taken using $\mathbf{y}$ and $\nabla_{\mathbf{y}} f(\mathbf{y}^{(k-1)})$ instead of $\mathbf{w}$ and $\nabla_{\mathbf{w}} f(\mathbf{w}^{(k-1)})$, respectively. The key of BOP relies on the following: if we execute Adam on $\mathbf{y}$ instead of $\mathbf{w}$, we can break the natural bond created by Adam between $\text{sgn}(\hat{\boldsymbol{\varphi}})$ and $\mathbf{w}$—as we saw in the previous sections—responsible for the ant-like behavior of the weights and, consequently, the appearance of side spikes in their histograms. These undesired effects disappear when we de-project $\mathbf{y}$ using $\mathbf{X}^{-1}$ to get back to the weight vector $\mathbf{w}$:

$$\mathbf{w} \quad = \quad \mathbf{X}^{-1}\mathbf{y}$$

In order to reduce the computational complexity and the memory requirements of this method—recall that N is generally a very large number and we must project and de-project the weights on each iteration—we consider $\mathbf{X}$ to be a block diagonal matrix with B identical $\frac{N}{B} \times \frac{N}{B}$ blocks. In this way, we only have to build and work with a single block $\mathbf{X}_B$, for which we can choose the size by simply adjusting the value of B. The values of this block are drawn from a standard normal distribution. In addition, $\mathbf{X}_B$ is built as an orthonormal matrix so that $\mathbf{X}_B^{-1} = \mathbf{X}_B^T$. Let $\mathbf{y}^{(i)}$ and $\mathbf{w}^{(i)}$ be the ith block of $\mathbf{y}$ and $\mathbf{w}$, respectively, both of them of length $\frac{N}{B}$; therefore, we just compute:

$$\begin{aligned} \mathbf{y}^{(i)} &= \mathbf{X}_B \mathbf{w}^{(i)} \\ \boldsymbol{\nabla}_{\mathbf{y}^{(i)}} f(\mathbf{y}^{(i)}) &= \mathbf{X}_B \boldsymbol{\nabla}_{\mathbf{w}^{(i)}} f(\mathbf{w}^{(i)}) \end{aligned}$$

After executing Adam, we can get back to $\mathbf{w}^{(i)}$:

$$\mathbf{w}^{(i)} \quad = \quad \mathbf{X}_B^T \mathbf{y}^{(i)}$$

As we will see in Section 6.2.4, BOP does not significantly alter the original distribution of weights, as opposed to standard Adam. This makes it possible to enjoy the advantages of Adam optimization when we implement the watermarking algorithm in [5,6] with a minimal increase in the detectability of the watermark. In addition, this has an advantage in terms of robustness: if the adversary is not able to infer which layer is watermarked, then he/she will have to exert his/her attack (e.g., noise addition, weight pruning) on *every* layer thus producing a larger impact on the performance of the network as measured by the original cost function. We will discuss this fact in the experimental section.

5. Information-Theoretic Measures

As already discussed, one of the potential weaknesses of any neural network watermarking algorithm is the detectability of the watermark. An adversary that detects the presence of a watermark on a certain subset of the weights can initiate an attack to remove or alter the watermark. For this reason, it is important that the weights statistically suffer the least modification possible while of course being able to convey the desired hidden message. To measure this statistical closeness, we propose using the KLD [13] between the distributions of weights before and after the watermark embedding. Let P and Q be two discrete probability distributions defined on the same alphabet $\mathcal{X}$; then, the KLD from Q to P is (notice that it is not symmetric):

$$\text{KLD}(P||Q) \doteq \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

The KLD is always non-negative. The more similar the distributions P and Q are, the smaller the divergence. In the extreme case of two identical distributions, the divergence is zero.

It is interesting to note that the KLD has been proposed for similar problems in forensics, including steganographic security [23], distinguishability between forensic operators [24], or more general source identification problems [25].

In our case, the two compared distributions are those of $\mathbf{w}^{(0)}$ and $\mathbf{w}^{(k)}$, for k just producing convergence with no decoding errors. Since the KLD is not symmetric, it remains to assign those distributions to P and Q so that the measure is as informative as possible. In particular, we are interested in properly accounting for the possible lateral spikes in the pdf of $\mathbf{w}^{(k)}$. As those spikes often appear where the pdf of $\mathbf{w}^{(0)}$ is small if not negligible, this suggests assigning the latter pdf to Q and the former to P. However, this choice creates a problem in practice, as for some $x \in \mathcal{X}$, the empirical probabilities are such that $P(x) \neq 0$ and $Q(x) = 0$, potentially leading to an infinite divergence. To circumvent this issue related to insufficient sampling, we use an analytical approximation to Q with infinite support, after noticing that the empirical distribution of $\mathbf{w}^{(0)}$ with 1000 discrete bins (see Figure 2a) can be approximated by a zero-mean Generalized Gaussian Distribution (GGD) with shape parameter $\beta = 0.64$ and scale parameter $\alpha = 0.01$, (for notational coherence with the literature, α is used in this section to denote a different quantity than in the rest of the paper.) for which the latter controls the spread of the distribution. As a reference, the KLD between the empirical distribution of $\mathbf{w}^{(0)}$ and its GGD best-fit is 0.0177, which is smaller than any of the KLDs that we find in Table 2. In order to compute the KLD in our experiments, we use this infinite-support symmetric distribution for Q and the empirical one of $\mathbf{w}^{(k)}$ for P after quantizing both to 1000 discrete bins.

Table 2. PSNR (dB) results with noise level $\sigma = 25$, number of iterations k needed to converge, KLD and SIKLD between the distributions of $\mathbf{w}^{(k)}$ and $\mathbf{w}^{(0)}$.

	SGD		**Adam**				**BOP**			
	Gaussian	**Orth.**	**Gaussian**		**Orth.**		**Gaussian**		**Orth.**	
λ	5	20	0.05	1	0.5	10	0.05	1	0.5	10
CBSD68	30.76	31.09	31.18	31.17	31.21	31.16	31.20	31.16	31.19	31.15
Kodak24	31.66	32.03	32.13	32.10	32.15	32.10	32.15	32.08	32.14	32.09
k	42,780	98,510	43,590	32,140	27,110	57,230	40,880	14,840	33,180	7150
KLD	0.0477	0.0238	0.1149	0.2779	0.0281	0.8118	0.0463	0.0443	0.0227	0.0253
SIKLD	0.0468	0.0206	0.0879	0.2112	0.0280	0.4707	0.0449	0.0266	0.0197	0.0226

The use of the KLD is adequate to measure the detectability in those cases where the adversary has access to information about the 'expected' distribution of the weights. For instance, when only one layer is modified, the expected distribution can be inferred from the weights of other layers. However, this may be still too optimistic in terms of adversarial success, as while the expected shape may be preserved—and thus, inferred—across layers, the scale (directly affecting the variance) may be not so. For instance, if the original weights were expected to be zero-mean Gaussian and they still are after watermarking, the KLD (which depends on the ratio of the respective variances) may be quite large, but the adversary will not be able to determine if watermarking took place if he/she does not know what the variance should be and only measures divergence with respect to a Gaussian. To reflect this uncertainty, quite realistic in practical situations, we minimize the KLD with respect to the scale parameter α. This puts the adversary in a scenario where only the shape is used for detectability. Thus, let Q_α correspond to a GGD with scale parameter α, then we define the Scale Invariant KLD (SIKLD) as:

$$\text{SIKLD} \doteq \min_{\alpha} \text{KLD}(P||Q_\alpha) = \min_{\alpha} \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q_\alpha(x)} \right)$$

6. Experiments and Results

In this section, we show the experimental results and we compare them to the theory that we have developed. We use MATLAB R2018b to implement the expressions obtained in Section 3 and represent the theoretical histograms. As we will see, both theory and experiments match reasonably well. In particular, for Adam optimization, we are able to reproduce the same position of the side spikes seen in the empirical histograms of $\Delta\mathbf{w}$, as well as some effects which are attributable to the influence of the denoising cost function. We will also verify the BOP method proposed in Section 4. In addition, the KLD will be computed to give a more precise measure of the similarity between the distributions of $\mathbf{w}^{(k)}$ (i.e., after the embedding) and $\mathbf{w}^{(0)}$, when using SGD, Adam and BOP.

6.1. Experimental Set-Up

We employ the fine-tune-to-embed approach described in [5,6]. This means that the training process is divided into two phases, as we explained earlier: (1) training the host network from scratch, and (2) fine-tuning steps for embedding the watermark.

6.1.1. Training the Host Network

In order to perform the initial training of the host network FFDNet, we use the open-source implementation for PyTorch provided in [26]. We employ the FFDNet architecture for color images, which has a depth of 12 convolutional layers and 96 feature maps per layer. The training details are the same as in [26] and also the used datasets: Waterloo Exploration Database [27] for training and Kodak24 [28] for validation. We implement the cost function introduced in (1) and train 80 epochs with the milestones described in [26] on a GPU NVIDIA Titan Xp. We use Adam as the optimization algorithm with its hyperparameters set to their default values. After training the network, we test it on the CBSD68 [29] and Kodak24 datasets.

6.1.2. Watermark Embedding

Once we have trained and tested our host network, we embed our T-bit watermark, $\mathbf{b} = \mathbf{1}$, $T = 256$, into the convolutional layer $l = 2$ of FFDNet. In the next section, we present the results for both SGD and Adam optimization algorithms. The size of the convolutional filter is 3×3 and the depth of input is $I = 96$, as well as the number of filters in the layer, $F = 96$. Therefore, we have: $M = 96 \cdot 3 \cdot 3 = 864$, and $N = MF = 82{,}944$. In addition, the learning rate μ is set to 10^{-6} during these fine-tuning steps and, also, we do not perform weight orthogonalization as we did during the initial training.

In addition, we use the following values for the regularizer parameter. When we use SGD we set $\lambda = 5$ and $\lambda = 20$ for Gaussian and orthogonal projectors, respectively. In addition, for Adam optimization, we use different values of λ for each configuration to better reflect the influence of the denoising function. In particular, we set $\lambda = 0.05$ and $\lambda = 1$ when we use Gaussian projectors, and $\lambda = 0.5$ and $\lambda = 10$ when we employ orthogonal projectors. We finish our embedding process when we reach a BER of zero, that is, when all the projected weights are positive—recall that all the embedded bits are set to $+1$—, i.e., $(\hat{\boldsymbol{\phi}}^{(m)})^T\mathbf{w} > 0$, for all $m = 1, \cdots, T$. Notice that these values of λ were selected with the goal of reaching a BER of zero in a relatively fast way and, as it can be seen, they are not straightforwardly comparable for Gaussian and orthogonal projectors. Finally, to check the validity of our proposed method BOP, we use the same values of λ as with Adam optimization, and set the number of blocks B to 12.

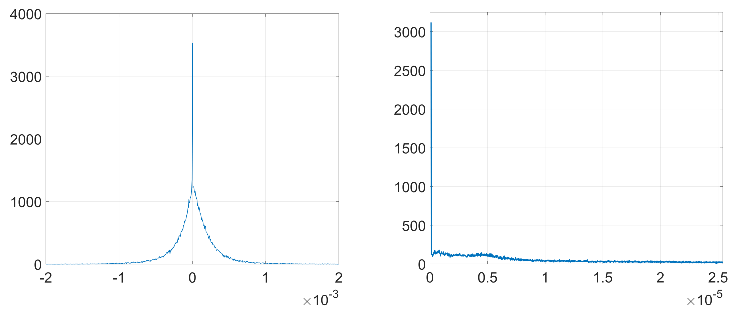
6.2. Experimental Results

Here, we present the experimental results. After the main training of the host network and before the watermark embedding, we obtain a PSNR of 31.18 dB and 32.13 dB on the CBSD68 and Kodak24 datasets, respectively, for a noise level of 25. These results are very close to those reported in [26]. Compare these values to those in Table 2, where we show the PSNR (dB) results on the CBSD68 and Kodak24 datasets for the same noise level after the watermark embedding was performed. As we see, when we embed the watermark using SGD with Gaussian projectors, the denoising performance drops about 0.45 dB, while, if we employ orthogonal projectors, the performance drops only 0.1 dB. Thus, employing orthogonal projectors with SGD optimization might be beneficial to better preserve the denoising performance. For Adam optimization and BOP, the original performance does not significantly drop and it even increases when using orthogonal projectors. Consequently, in order to keep a good performance after the watermark embedding, Adam would be preferable to SGD were it not for the conspicuousness of the weights. Our proposed method BOP is a good solution to bring the detectability of Adam down to similar levels as SGD and still enjoy the rest of advantages.

Table 2 also presents the number of iterations required for obtaining a BER of zero and the KLD and Scale Invariant KLD (SIKLD) between the distributions of $\mathbf{w}^{(k)}$ and $\mathbf{w}^{(0)}$ for each configuration.

6.2.1. Empirical Denoising Gradients

In order to analyze the influence of the denoising function, we need to get the empirical distributions of the mean denoising gradient and the variance of the batching noise. To that end, we proceed as follows: firstly, we extract the denoising gradients from the embedding layer $l = 2$ and we average them for each coefficient over the number of iterations k to get the distribution of the mean. Then, the batching noise can be easily computed if, for each coefficient, we subtract the mean value from its corresponding denoising gradient value at each iteration. By computing the variance of this noise for each individual weight, we can estimate the overall distribution of the variance of the batching noise. Figure 3 shows the empirical distribution of the mean denoising gradient, D, and the variance of the batching noise, H.



(**a**) (**b**)

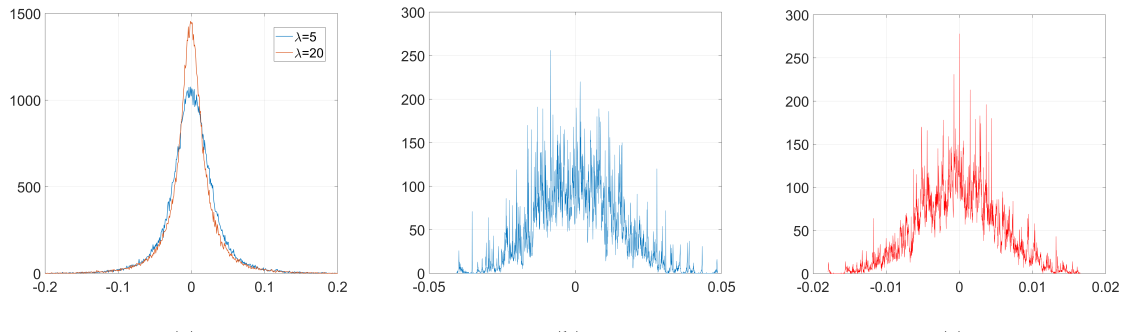
Figure 3. Empirical histograms from the denoising gradients (**a**) distribution of the mean denoising gradient, D; (**b**) distribution of the variance of the batching noise, H.

6.2.2. SGD

We embed our watermark using SGD and its corresponding set-up, as we detailed in Section 6.1.2. We show the resulting histograms of $\mathbf{w}^{(k)}$ and $\Delta\mathbf{w}$ in Figure 4. As we can see, SGD does not significantly alter the original distribution of weights shown in Figure 2a. This is also reflected in the SIKLD, which is very small, especially when we use orthogonal projectors (see Table 2).

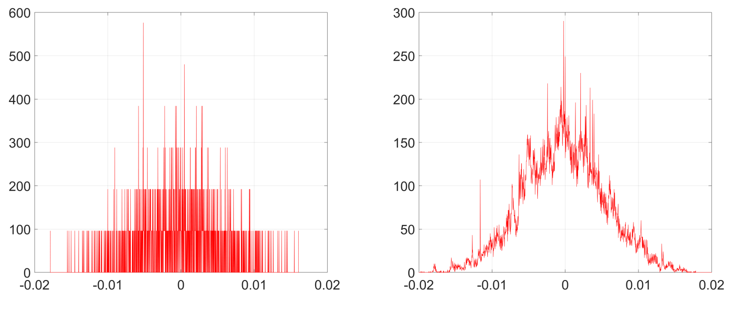
Now, we check the theory we developed for SGD in Section 3.1. Recall that out theoretical analysis just covers the case of orthogonal projectors. Using (24), we can generate samples of $\Delta\mathbf{w}$ without including the effect of the denoising cost function. The resulting histogram is shown in Figure 5a. Notice that the unusual appearance of this histogram can be attributed to the effects of applying the initial transformation explained in Section 2.3.1. In particular, each value of $\boldsymbol{\varphi}$ repeats F times to form vector $\hat{\boldsymbol{\varphi}}$; hence, the discrete values in the y-axis of the histogram shown in Figure 5a. However, we see that the range of values of this theoretical histogram fits quite well the empirical one (Figure 4c). In order to get a more accurate representation, we can generate samples of $\Delta\mathbf{w}$ according to (48), so that we add the effect of noise coming from the denoising cost function. As we see in Figure 5b, the resulting histogram is now very similar to the one in Figure 4c.

In addition, we confirm that (25) can be used to compute the variance of the distribution of $\mathbf{w}^{(k)}$ when we implement orthogonal projectors. Firstly, we check the hypothesis that we made regarding the uncorrelatedness between $\mathbf{w}^{(0)}$ and $\Delta\mathbf{w}$. For our particular case of $\lambda = 20$, the correlation coefficient between $\mathbf{w}^{(0)}$ and $\Delta\mathbf{w}$ is $2.539 \cdot 10^{-4}$, a very small value that confirms our assumption. Using (25), we have that the variance of the empirical distribution of $\mathbf{w}^{(k)}$—red histogram in Figure 4c—is $1.192 \cdot 10^{-3}$ while the theoretical variance is $1.193 \cdot 10^{-3}$. As we see, these values are almost identical.



(**a**) (**b**) (**c**)

Figure 4. Empirical histograms after the watermark embedding using SGD. (**a**) histogram of $\mathbf{w}^{(k)}$, Gaussian, $\lambda = 5$, and orthogonal projectors, $\lambda = 20$; (**b**) histogram of $\Delta\mathbf{w}$, Gaussian, $\lambda = 5$; (**c**) histogram of $\Delta\mathbf{w}$, orthogonal, $\lambda = 20$.



(**a**) (**b**)

Figure 5. Theoretical histograms of $\Delta\mathbf{w}$ for SGD, orthogonal projectors, $\lambda = 20$. (**a**) only watermarking function, Equation (24); (**b**) including denoising and watermarking functions, Equation (48).

6.2.3. Adam

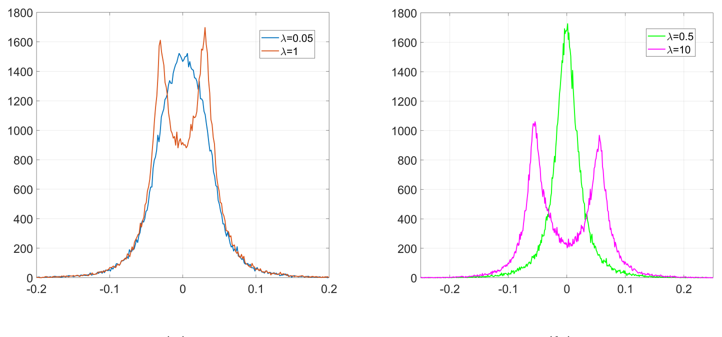
In the following experiments, we employ Adam optimization for the watermark embedding and use the same settings as in Section 6.1.2. The resulting histograms of $\mathbf{w}^{(k)}$ and $\Delta\mathbf{w}$ are shown in Figures 6 and 7, respectively. As it can be observed, the shape of the weight distribution changes to a great extent for $\lambda = 1$ and $\lambda = 10$. For smaller values of λ, since the influence of the watermarking cost function is weaker, we can avoid having a significant alteration to the original distribution shape. This is also reflected on their SIKLD values in Table 2: as λ increases the SIKLD also increases considerably. However, notice that, whatever the value of λ is, the histograms of weight variations always present the characteristic side spikes. These footprints left by Adam can increase the detectability of the watermark. In addition, when λ is small, we can observe in these histograms the influence of the denoising cost function: it causes the appearance of a central peak with values that spread till the location of both side spikes.

Figure 8 represents the pdf of Γ obtained from (43). Notice that, as we stated in Section 3.2.5, the pdf is concentrated around its mode. Figure 9 shows the histograms of $\Delta\mathbf{w}$ obtained from (44) when only the watermarking loss $\tilde{f}(\mathbf{w})$ is optimized and the denoising component is set to zero. Compare these histograms to those in Figure 7: the theory developed in Section 3.2 is able to explain the two-spiked distributions of $\Delta\mathbf{w}$. Notice that these theoretical expressions provide a good enough approximation since they allow us to predict the position of the side spikes. We show in Table 3 the values of these positions obtained from both theoretical (Figure 9) and empirical (Figure 7) results. As we can see, these side spikes are placed in almost identical positions by both theory and experiments, hence, we can confirm that the sign phenomenon in Adam is responsible for this ant-like behavior shown by the weights at the embedding layer.

Table 3. Position of both side spikes in the histograms of $\Delta\mathbf{w}$ obtained from theoretical and empirical results.

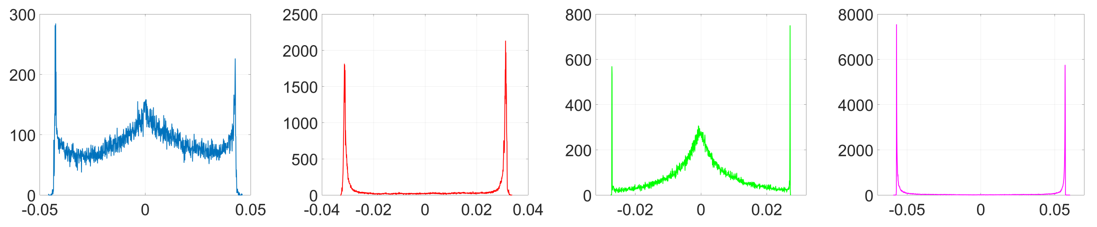
	λ	Theoretical	Empirical
Gaussian	0.05	−0.04243 0.04239	−0.04278 0.04276
	1	−0.03129 0.03126	−0.03119 0.03125
Orthogonal	0.5	−0.02710 0.02710	−0.02710 0.02709
	10	−0.05720 0.05720	−0.05717 0.05718

Finally, in order to reflect the influence of the denoising function and obtain more realistic histograms, we can solve the fourth-degree Equation (A11) and, then, generate samples of $\Delta\mathbf{w}$ according to (A12). The resulting histograms are shown in Figure 10 and are now quite similar to those in Figure 7. As it can be seen, we can emulate the central peak and the dispersion of the values of the side spikes in the histograms and, thus, we can confirm that these effects are attributable to the influence of the denoising cost function.



(**a**) (**b**)

Figure 6. Empirical histograms of $\mathbf{w}^{(k)}$ after the watermark embedding using Adam. (**a**) Gaussian, $\lambda = 0.05$ and $\lambda = 1$; (**b**) orthogonal, $\lambda = 0.5$ and $\lambda = 10$.



(**a**) (**b**) (**c**) (**d**)

Figure 7. Empirical histograms of $\Delta\mathbf{w}$ after the watermark embedding using Adam. (**a**) Gaussian, $\lambda = 0.05$; (**b**) Gaussian, $\lambda = 1$; (**c**) orthogonal, $\lambda = 0.5$; (**d**) orthogonal, $\lambda = 10$.

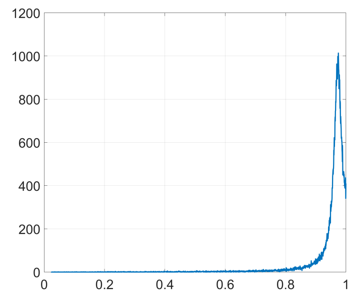


Figure 8. Pdf of Γ, Gaussian projectors.

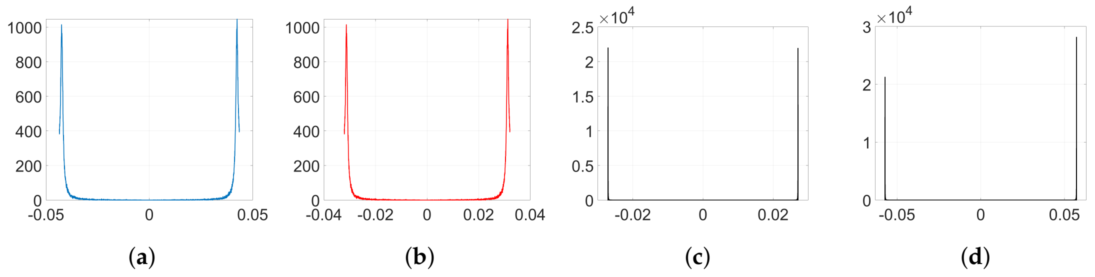


Figure 9. Theoretical histograms of $\Delta\mathbf{w}$ for Adam with only watermarking function using Equations (43) and (44). (**a**) Gaussian, $\lambda = 0.05$; (**b**) Gaussian, $\lambda = 1$; (**c**) orthogonal, $\lambda = 0.5$; (**d**) orthogonal, $\lambda = 10$.

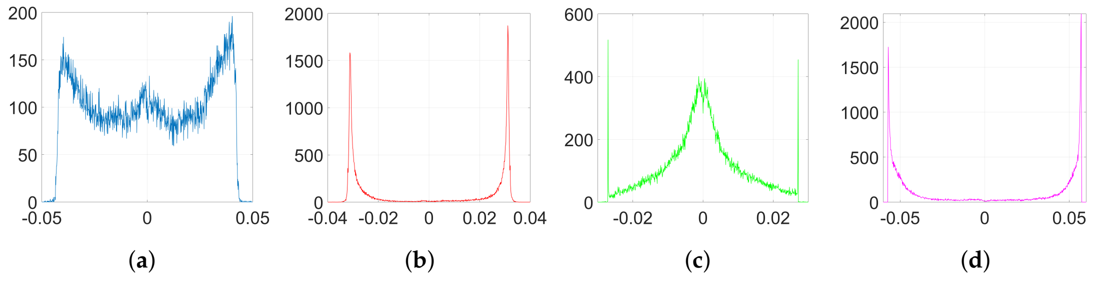


Figure 10. Theoretical histograms of $\Delta\mathbf{w}$ for Adam with denoising and watermarking functions using Equations (A11) and (A12). (**a**) Gaussian, $\lambda = 0.05$; (**b**) Gaussian, $\lambda = 1$; (**c**) orthogonal, $\lambda = 0.5$; (**d**) orthogonal, $\lambda = 10$.

6.2.4. BOP

Here, we represent the empirical histograms when we implement our method BOP. Figures 11 and 12 show the histograms of $\mathbf{w}^{(k)}$ and $\Delta\mathbf{w}$, respectively. As we see from these histograms and the KLD and SIKLD values in Table 2, this method allows us to remove the side spikes of the histograms and much better preserve the original shape of the weight distribution. As a result, the detectability of the watermark due to Adam optimization is strongly reduced.

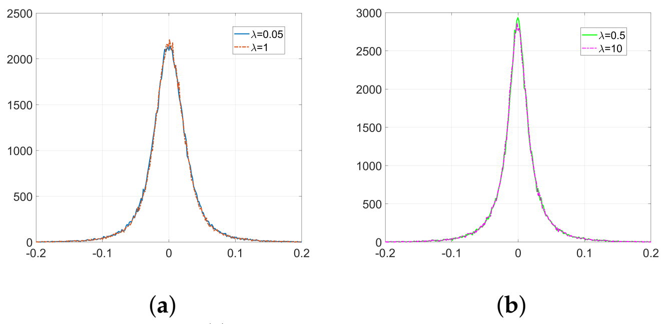


Figure 11. Empirical histograms of $\mathbf{w}^{(k)}$ after the watermark embedding using Adam. (**a**) Gaussian, $\lambda = 0.05$ and $\lambda = 1$; (**b**) orthogonal, $\lambda = 0.5$ and $\lambda = 10$.

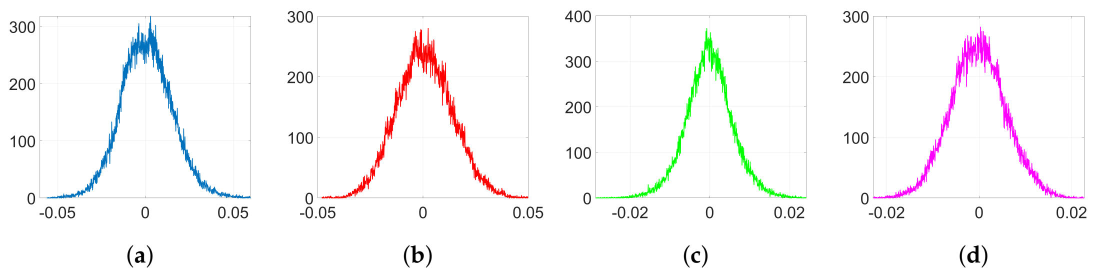
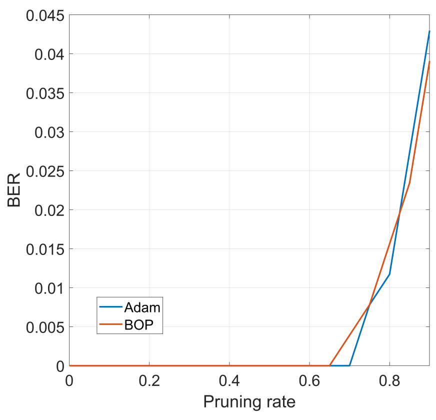


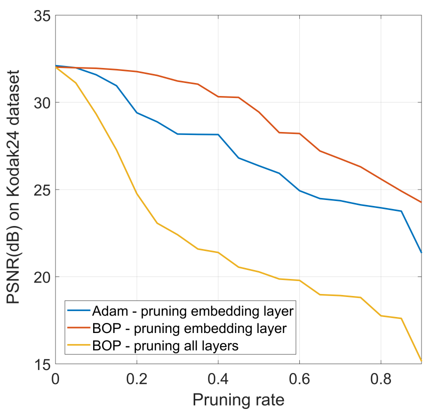
Figure 12. Empirical histograms of $\Delta\mathbf{w}$ after the watermark embedding using Adam. (**a**) Gaussian, $\lambda = 0.05$; (**b**) Gaussian, $\lambda = 1$; (**c**) orthogonal, $\lambda = 0.5$; (**d**) orthogonal, $\lambda = 10$.

A positive side effect of the undetectability of the watermark is that the robustness is increased because an adversary will not know which layer must be modified in order to alter the embedded watermark. This is illustrated in Figure 13 where we compare the robustness of standard Adam with that of BOP against weight pruning. The network is trained long enough for both Adam and BOP to guarantee a similar BER vs. pruning rate, as shown in Figure 13a. Then, the PSNR obtained after training and pruning is shown in Figure 13b for the Kodak24 dataset which illustrates the following facts: (1) BOP produces a network that is more robust to pruning in terms of PSNR, which can be valuable towards model compression; for instance, for a pruning rate of 0.35 (that has no impact on the BER of the hidden information), BOP degrades the original PSNR by about 1 dB, whereas Adam would produce a degradation of more than 3 dB. (2) This robustness might be detrimental in case it is an attacker who does the pruning in an attempt to degrade the watermark; for instance, for a pruning rate of 0.82, the BER for both Adam and BOP rises to 0.02 (see Figure 13a). For this pruning rate, the PSNR of BOP is around 25 dB while Adam gives slightly less than 24 dB. Then, in the case of BOP, the adversary would be able to produce a network that performs closer to the original in terms of PSNR—notice, however, that the degradation in both cases is quite severe, so this heavy pruning would render a denoiser with little practical use —. (3) In any case, the previous comparison would assume that the adversary knows the layer that contains the watermark; as we have properly justified, this is reasonable for Adam but not so for BOP. If the adversary does not know the layer that must be pruned, then, to achieve the same target BER, he/she must prune all the layers. In this case, for the same pruning rate of 0.82 that causes the BER to increase to 0.02, the PSNR drops to less than 18 dB.

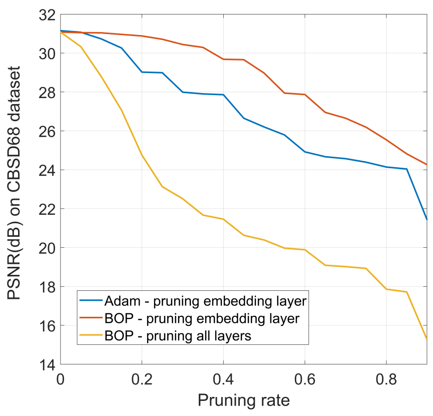
Similar conclusions can be extracted from Figure 13c that shows the PSNR vs. pruning rate for the CBSD68 dataset. These experiments clearly show the higher robustness brought about by the undetectability of BOP, as it prevents attacks targeted to a specific layer.



(**a**)



(**b**)



(**c**)

Figure 13. (**a**) BER vs. pruning rate for Adam and BOP (pruning all layers or only the watermarked one does not have any impact on BER); (**b**) PSNR vs. pruning rate for Adam and BOP for the Kodak24 dataset; (**c**) PSNR vs. pruning rate for Adam and BOP for the CBSD68 dataset.

7. Conclusions

Throughout this paper, we have shown the importance of being careful with the optimization algorithm when we embed watermarks following the approach in [5,6]. The choice of certain optimization algorithms whose update direction is given by the sign function can originate footprints in the distributions of weights that are easily detectable by adversaries, thus compromising the efficacy of the watermarking algorithm.

In particular, we studied the mechanisms behind SGD and Adam optimization and found that the sign phenomenon that occurs in Adam is detrimental for watermarking, since it causes the appearance of two salient side spikes on the histograms of weights. As opposed to Adam, the sign function does not appear when we use SGD. Therefore, SGD does not significantly alter the original shape of the distribution of weights although, as we showed in the theoretical analysis, it slightly increases its variance. The analysis in this paper can be extended to other optimization algorithms.

In addition, we introduced orthogonal projectors and observed that, compared to the Gaussian case, they generally preserve the original performance and weight distribution better. However, a deeper analysis on this subject is left for further research.

Finally, we presented a novel method that uses orthogonal block projections to address the use of Adam optimization together with the watermarking algorithm under study. As we checked in the empirical section, this method allows us to solve the detectability problem posed by Adam and still enjoy the rest of advantages of this optimization algorithm.

Author Contributions: Conceptualization, B.C.-L. and F.P.-G.; methodology, F.P.-G.; software, B.C.-L.; validation, B.C.-L. and F.P.-G.; formal analysis, B.C.-L. and F.P.-G.; investigation, B.C.-L.; resources, B.C.-L.; writing—original draft preparation, B.C.-L.; writing—review and editing, B.C.-L. and F.P.-G.; supervision, F.P.-G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially funded by the Agencia Estatal de Investigación (Spain) and the European Regional Development Fund (ERDF) under projects RODIN (PID2019-105717RB-C21) and RED COMONSENS (RED2018-102668-T). In addition, it was funded by the Xunta de Galicia and ERDF under projects ED431C 2017/53 and ED431G 2019/08.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Mathematical Derivations

Appendix A.1. Projected Weights at $k = 0$

We want to compute the mean and the variance of $\hat{\mathbf{\Phi}}^T \mathbf{w}^{(0)}$. It is easy to see that the mean is zero for any $(\hat{\boldsymbol{\phi}}^{(i)})^T \mathbf{w}^{(0)}, i = 1, \cdots, T$ because $\mathrm{E}\{\hat{\phi}_j^{(i)}\} = 0$.

Now, in order to obtain an expression for the variance, we may compute the expectation of the trace of the covariance matrix of $\hat{\mathbf{\Phi}}^T \mathbf{w}^{(0)}$ divided by T. Therefore, if we use the properties of the trace, we have:

$$\begin{aligned}
\mathrm{Var}\{\hat{\mathbf{\Phi}}^T \mathbf{w}^{(0)}\} &= \frac{1}{T}\mathrm{E}\left\{\mathrm{Tr}\left[\hat{\mathbf{\Phi}}^T \mathbf{w}^{(0)} (\mathbf{w}^{(0)})^T \hat{\mathbf{\Phi}}\right]\right\} \\
&= \frac{1}{T}\mathrm{E}\left\{\mathrm{Tr}\left[\mathbf{w}^{(0)} (\mathbf{w}^{(0)})^T \hat{\mathbf{\Phi}}\hat{\mathbf{\Phi}}^T\right]\right\} \\
&= \frac{1}{T}\sum_{i,j}\mathrm{E}\left\{(\mathbf{w}^{(0)} (\mathbf{w}^{(0)})^T)_{i,j} (\hat{\mathbf{\Phi}}\hat{\mathbf{\Phi}}^T)_{i,j}\right\}
\end{aligned}$$

We can write:

$$\mathrm{E}\left\{(\mathbf{w}^{(0)} (\mathbf{w}^{(0)})^T)_{i,j} (\hat{\mathbf{\Phi}}\hat{\mathbf{\Phi}}^T)_{i,j}\right\} = \mathrm{E}\{w_i^{(0)} w_j^{(0)}\}\mathrm{E}\left\{\sum_{m=1}^{T} \hat{\phi}_i^{(m)} \hat{\phi}_j^{(m)}\right\} \tag{A1}$$

It is straightforward to see that, for the off-diagonal terms, i.e., $i \neq j$, the expectation in (A1) is zero. For the N diagonal terms, i.e., $i = j$, we have instead:

$$\mathrm{E}\left\{\left(w_i^{(0)}\right)^2\right\}\mathrm{E}\left\{\sum_{m=1}^{T} (\hat{\phi}_i^{(m)})^2\right\} = T \cdot \mathrm{Var}\{\mathbf{w}^{(0)}\}\mathrm{Var}\{\hat{\boldsymbol{\phi}}^{(m)}\}$$

where $\mathrm{Var}\{\hat{\boldsymbol{\phi}}^{(m)}\}$ is the variance of any projection vector $\hat{\boldsymbol{\phi}}^{(m)}$, $m = 1, \cdots, T$. When we use Gaussian projectors, we have $\mathrm{Var}\{\hat{\boldsymbol{\phi}}^{(m)}\} = 1/F^2$ and when we use orthogonal projectors, we have $\mathrm{Var}\{\hat{\boldsymbol{\phi}}^{(m)}\} = 1/(FN)$. Therefore, the variance for Gaussian projectors is

$$\mathrm{Var}\{\hat{\mathbf{\Phi}}^T \mathbf{w}^{(0)}\} = \frac{N}{F^2}\mathrm{Var}\{\mathbf{w}^{(0)}\}$$

while for orthogonal projector is

$$\mathrm{Var}\{\hat{\mathbf{\Phi}}^T \mathbf{w}^{(0)}\} = \frac{1}{F}\mathrm{Var}\{\mathbf{w}^{(0)}\}$$

Appendix A.2. Adam: Mean of the Gradient

In order to find an expression for the mean of the gradient, we substitute (17) into (26) and obtain:

$$\begin{aligned}\mathbf{m}^{(k)} &= -\frac{\lambda}{2}\cdot\hat{\boldsymbol{\varphi}}\cdot(1-\beta_1)\sum_{i=1}^{k}\beta_1^{k-i}+\frac{\lambda}{4}\cdot\hat{\boldsymbol{\Psi}}\cdot(1-\beta_1)\sum_{i=1}^{k}\beta_1^{k-i}\mathbf{w}^{(i)}\\ &= -\frac{\lambda}{2}\cdot\hat{\boldsymbol{\varphi}}\cdot(1-\beta_1^k)+\frac{\lambda}{4}\cdot\hat{\boldsymbol{\Psi}}\cdot(1-\beta_1)\sum_{i=1}^{k}\beta_1^{k-i}\mathbf{w}^{(i)}\end{aligned}$$

The bias-corrected mean gradient is:

$$\hat{\mathbf{m}}^{(k)}=\frac{\mathbf{m}^{(k)}}{1-\beta_1^k}=-\frac{\lambda}{2}\hat{\boldsymbol{\varphi}}+\frac{\lambda}{4}\cdot\hat{\boldsymbol{\Psi}}\cdot\frac{1-\beta_1}{1-\beta_1^k}\sum_{i=1}^{k}\beta_1^{k-i}\mathbf{w}^{(i)}$$

The main difficulty in finding an explicit expression for $\hat{\mathbf{m}}^{(k)}$ is the last sum which requires a closed-form expression for $\mathbf{w}^{(i)}$, for all $i=0,\cdots,k$, which in turn depends on the Adam adaptation. This easily leads to a difference equation whose solution is cumbersome. Alternatively, we start by conjecturing the affine-growth in (18) and write:

$$\begin{aligned}\frac{1-\beta_1}{1-\beta_1^k}\sum_{i=1}^{k}\beta_1^{k-i}\mathbf{w}^{(i)} &= \frac{1-\beta_1}{1-\beta_1^k}\sum_{i=1}^{k}\beta_1^{k-i}(\mathbf{w}^{(0)}+i\cdot\mu\boldsymbol{\eta})\\ &= \mathbf{w}^{(0)}+\mu\boldsymbol{\eta}\frac{1-\beta_1}{1-\beta_1^k}\sum_{i=1}^{k}i\cdot\beta_1^{k-i}\\ &= \mathbf{w}^{(0)}+\mu\boldsymbol{\eta}\left(\frac{k}{1-\beta_1^k}-\frac{\beta_1}{1-\beta_1}\right)\end{aligned}$$

Therefore, under the affine-growth hypothesis, we can write:

$$\hat{\mathbf{m}}^{(k)}=-\frac{\lambda}{2}\hat{\boldsymbol{\varphi}}+\frac{\lambda}{4}\cdot\hat{\boldsymbol{\Psi}}\left(\mathbf{w}^{(0)}+\mu\boldsymbol{\eta}\left(\frac{k}{1-\beta_1^k}-\frac{\beta_1}{1-\beta_1}\right)\right)$$

and arrive at the expression in (27).

Appendix A.3. Adam: Variance of the Gradient

To get an expression for the variance of the gradient, we plug (31) into (34) and write:

$$\begin{aligned}v_j^{(k)} &= \frac{\lambda^2}{4}(1-\beta_2)\sum_{i=1}^{k}\beta_2^{k-i}(p_j+q_j\mu i+r_j\mu^2 i^2)\\ &= \frac{\lambda^2}{4}(1-\beta_2)p_j\sum_{i=1}^{k}\beta_2^{k-i}+\frac{\lambda^2}{4}(1-\beta_2)q_j\mu\sum_{i=1}^{k}i\beta_2^{k-i}+\frac{\lambda^2}{4}(1-\beta_2)r_j\mu^2\sum_{i=1}^{k}i^2\beta_2^{k-i}\\ &= \frac{\lambda^2}{4}(1-\beta_2^k)\left(p_j-\frac{\beta_2}{1-\beta_2}\mu q_j+\frac{\beta_2+\beta_2^2}{(1-\beta_2)^2}\mu^2 r_j\right)+\frac{\lambda^2}{4}k\left(\mu q_j-\frac{2\beta_2}{1-\beta_2}\mu^2 r_j\right)+\frac{\lambda^2}{4}k^2\mu^2 r_j\end{aligned}$$

Appendix A.4. Adam: A Projection-Based Decomposition of $\mathbf{c}_j^T\mathrm{sgn}(\hat{\boldsymbol{\varphi}})$

We want to write $\mathbf{c}_j^T\mathrm{sgn}(\hat{\boldsymbol{\varphi}})=\alpha\cdot\hat{\varphi}_j+z_j$. To find α, we compute the cross-correlation with $\hat{\boldsymbol{\varphi}}$, i.e., $\mathrm{E}\{\hat{\boldsymbol{\varphi}}^T\mathbf{C}^T\mathrm{sgn}(\hat{\boldsymbol{\varphi}})\}=\alpha\mathrm{E}\{||\hat{\boldsymbol{\varphi}}||^2\}$. Then, $\mathbf{z}$ is simply $\mathbf{C}^T\mathrm{sgn}(\hat{\boldsymbol{\varphi}})-\alpha\cdot\hat{\boldsymbol{\varphi}}$. In the following, we show separately the derivations for the Gaussian and orthogonal projectors.

Appendix A.4.1. Decomposition for Gaussian Projectors

Recall that the projection vectors have i.i.d. components drawn from a $\mathcal{N}(0,1)$ distribution. Since the quantity of interest $\mathrm{E}\{\hat{\boldsymbol{\varphi}}^T\mathbf{C}^T\mathrm{sgn}(\hat{\boldsymbol{\varphi}})\}$ is a scalar, we will use the trace to manipulate the involved matrices, as follows:

$$\begin{aligned}\mathrm{E}\{\hat{\boldsymbol{\varphi}}^T\mathbf{C}^T\mathrm{sgn}(\hat{\boldsymbol{\varphi}})\} &= \mathrm{Tr}\left[\mathrm{E}\{\hat{\boldsymbol{\varphi}}^T\mathbf{C}^T\mathrm{sgn}(\hat{\boldsymbol{\varphi}})\}\right] = \mathrm{E}\Big\{\mathrm{Tr}[\mathbf{C}^T\mathrm{sgn}(\hat{\boldsymbol{\varphi}})\hat{\boldsymbol{\varphi}}^T]\Big\} \\ &= \sum_{i,j}\mathrm{E}\Big\{\left(\mathbf{C}\circ(\mathrm{sgn}(\hat{\boldsymbol{\varphi}})\hat{\boldsymbol{\varphi}}^T)\right)_{i,j}\Big\} = \sum_{i,j}\sum_{m=1}^{T}\sum_{n=1}^{T}\mathrm{E}\{\hat{\phi}_i^{(m)}\mathrm{sgn}(\hat{\varphi}_i)\hat{\phi}_j^{(m)}\hat{\phi}_j^{(n)}\}\end{aligned}$$

where $\hat{\phi}_i^{(m)} = \boldsymbol{\theta}_i\boldsymbol{\phi}^{(m)}$, $m = 1,\cdots,T$, $i = 1,\cdots,N$, and $\boldsymbol{\theta}_i$ is the ith row of matrix $\boldsymbol{\Theta}$. In order to compute the previous expectation, we need to consider separately the diagonal terms and the off-diagonal ones.

We start with the off-diagonal elements (i.e., $i \neq j$). Here, we also need to distinguish the following cases: $\left\lfloor \frac{i}{F}\right\rfloor \neq \left\lfloor \frac{j}{F}\right\rfloor$, satisfied by $N(N-F)$ elements, and $\left\lfloor \frac{i}{F}\right\rfloor = \left\lfloor \frac{j}{F}\right\rfloor$, which applies for the remaining $N(F-1)$ off-diagonal terms. Therefore, in the first subcase, we have that $\boldsymbol{\theta}_i \neq \boldsymbol{\theta}_j$, so $\hat{\phi}_i^{(m)} \neq \hat{\phi}_j^{(m)}$ and:

$$\sum_{m=1}^{T}\sum_{n=1}^{T}\mathrm{E}\{\hat{\phi}_i^{(m)}\mathrm{sgn}(\hat{\boldsymbol{\varphi}}_i)\hat{\phi}_j^{(m)}\hat{\phi}_j^{(n)}\} = \sum_{m=1}^{T}\mathrm{E}\{\hat{\phi}_i^{(m)}\mathrm{sgn}(\hat{\boldsymbol{\varphi}}_i)\}\mathrm{E}\{(\hat{\phi}_j^{(m)})^2\} \tag{A2}$$

In order to compute the expectation $\mathrm{E}\{\hat{\phi}_i^{(m)}\mathrm{sgn}(\hat{\boldsymbol{\varphi}}_i)\}$ in (A2), we write:

$$\begin{aligned}\hat{\varphi}_i &= \hat{\phi}_i^{(m)} + \sum_{\substack{l=1\\ i\neq m}}^{T}\hat{\phi}_i^{(l)} \\ &= \hat{\phi}_i^{(m)} + n_i\end{aligned}$$

where n_i can be seen as a realization of an i.i.d. Gaussian distribution with zero-mean and variance $(T-1)/F^2$. Assume without loss of generality that $\hat{\phi}_i^{(m)} > 0$. Then, the probability that $\mathrm{sgn}(\hat{\varphi}_i) = -1$ is $p \doteq Q\left(\hat{\phi}_i^{(m)}F/\sqrt{T-1}\right)$, ($Q(x) \doteq \frac{1}{\sqrt{2\pi}}\int_x^\infty e^{-x^2/2}dx$.) and $\hat{\phi}_i^{(m)}\mathrm{sgn}(\hat{\varphi}_i)$ will take the value $-\hat{\phi}_i^{(m)}$ with probability p, and $\hat{\phi}_i^{(m)}$ with probability $1-p$. Then, the distribution of $\hat{\phi}_i^{(m)}\mathrm{sgn}(\hat{\varphi}_i)$ will correspond to that of a random variable Y constructed as follows: letting $X \sim \mathcal{N}(0,1/F^2)$, we define Y as $Y \doteq |X|\cdot\left(1-2Q(|X|F/\sqrt{T-1})\right)$. Then,

$$E\{Y\} = \frac{F}{\sqrt{2\pi}}\int_0^\infty 2x\left(1-2Q\left(\frac{xF}{\sqrt{T-1}}\right)\right)e^{-x^2F^2/2}dx$$

It can be shown that this integral gives [30]:

$$E\{Y\} = \frac{1}{F}\sqrt{\frac{2}{\pi T}} \doteq \mu_Y \tag{A3}$$

Therefore, when $i \neq j$ and $\left\lfloor \frac{i}{F}\right\rfloor \neq \left\lfloor \frac{j}{F}\right\rfloor$, we can compute (A2) as:

$$\sum_{m=1}^{T}\mathrm{E}\{\hat{\phi}_i^{(m)}\mathrm{sgn}(\hat{\boldsymbol{\varphi}}_i)\}\mathrm{E}\{(\hat{\phi}_j^{(m)})^2\} = \frac{T}{F^2}\mu_Y$$

When $i \neq j$ and $\left\lfloor \frac{i}{F} \right\rfloor = \left\lfloor \frac{j}{F} \right\rfloor$, we have that $\boldsymbol{\theta}_i = \boldsymbol{\theta}_j$, so $\hat{\phi}_i^{(m)} = \hat{\phi}_j^{(m)}$. Thus:

$$\begin{aligned}\sum_{m=1}^{T}\sum_{n=1}^{T} \mathrm{E}\{(\hat{\phi}_i^{(m)})^2 \hat{\phi}_i^{(n)} \mathrm{sgn}(\hat{\boldsymbol{\varphi}}_i)\} &= \sum_{m=1}^{T} \mathrm{E}\{(\hat{\phi}_i^{(m)})^3 \mathrm{sgn}(\hat{\boldsymbol{\varphi}}_i)\} + \sum_{m=1}^{T}\sum_{\substack{n=1\\ n\neq m}}^{T} \mathrm{E}\{(\hat{\phi}_i^{(m)})^2 \hat{\phi}_i^{(n)} \mathrm{sgn}(\hat{\boldsymbol{\varphi}}_i)\} \\ &= T\xi_{Y'} + T(T-1)\mathrm{E}\{(\hat{\phi}_i^{(m)})^2 \hat{\phi}_i^{(n)} \mathrm{sgn}(\hat{\boldsymbol{\varphi}}_i)\} \end{aligned} \tag{A4}$$

where $\xi_{Y'}$ can be computed as the mean of Y' defined in a similar way as above, i.e.,: let $X \sim \mathcal{N}(0, 1/F^2)$, we define Y' as $Y' \doteq |X|^3 \cdot \left(1 - 2Q(|X|F/\sqrt{T-1})\right)$. Then:

$$\xi_{Y'} \doteq E\{Y'\} = \frac{F}{\sqrt{2\pi}} \int_0^\infty 2x^3 \left(1 - 2Q\left(\frac{xF}{\sqrt{T-1}}\right)\right) e^{-x^2F^2/2} dx$$

Using Mathematica, it is found that:

$$\xi_{Y'} = \sqrt{\frac{2}{\pi T}} \frac{3T-1}{TF^3} = \frac{(3T-1)}{TF^2} \mu_Y \tag{A5}$$

Unfortunately, the double integral required to compute the expectation in (A4) does not seem to admit a closed-form solution. On the other hand, its numerical computation through Monte Carlo integration is rather straightforward. Let $X, Y \sim \mathcal{N}(0, 1/F^2)$ and $Z \sim \mathcal{N}(0, \frac{T-2}{F^2})$, all mutually independent, then the desired expectation is $E\{XY^2 \mathrm{sgn}(X+Y+Z)\} \doteq \kappa(T)$, where, with $\kappa(T)$, we stress the fact that the integral depends on T alone. The result is represented in Figure A1. It is possible to show that, as $T \to \infty$, $\kappa(T) \to \frac{\mu_Y}{F^2}$.

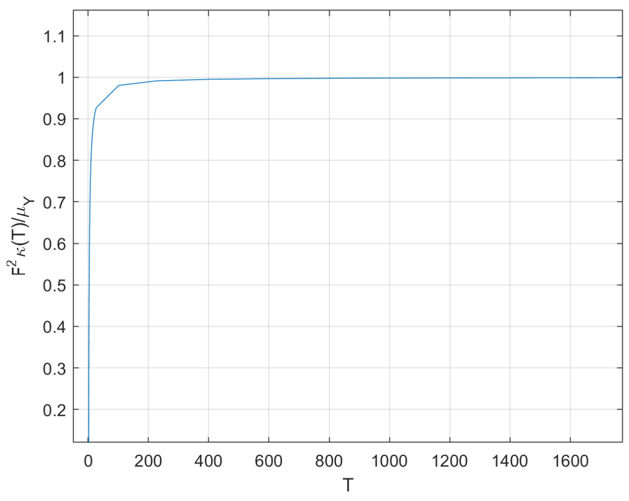


Figure A1. $F^2 \cdot \kappa(T)/\mu_Y$ vs. T

Then, for moderately large T, the sum in (A4) is approximately $\mu_Y(T^2 + 2T - 1)/F^2$.

For the $i = j$ case (i.e., diagonal terms), we obtain the same result as (A4). Therefore, when computing $\sum_{i,j} \mathrm{E}\{\left(\mathbf{C} \circ (\mathrm{sgn}(\hat{\boldsymbol{\varphi}})\hat{\boldsymbol{\varphi}}^T)\right)_{i,j}\}$, there are $N(N-F)$ terms with value $\mu_Y T/F^2$, and NF terms with approximate value $\mu_Y(T^2 + 2T - 1)/F^2$, hence:

$$\mathrm{E}\{\hat{\boldsymbol{\varphi}}^T \mathbf{C}^T \mathrm{sgn}(\hat{\boldsymbol{\varphi}})\} \approx \frac{N\mu_Y}{F^2}(NT + F(T^2 + T - 1))$$

In addition, $\mathrm{E}\{||\hat{\boldsymbol{\varphi}}||^2\}\} = \mathrm{E}\{\hat{\boldsymbol{\varphi}}^T\hat{\boldsymbol{\varphi}}\} = \mathrm{E}\{\boldsymbol{\varphi}^T\boldsymbol{\Theta}^T\boldsymbol{\Theta}\boldsymbol{\varphi}\} = \frac{MT}{F} = \frac{NT}{F^2}$, so we have:

$$\begin{aligned}\alpha &= \frac{\mathrm{E}\{\hat{\boldsymbol{\varphi}}^T\mathbf{C}^T\mathrm{sgn}(\hat{\boldsymbol{\varphi}})\}}{\mathrm{E}\{||\hat{\boldsymbol{\varphi}}||^2\}\}} \approx \frac{F^2N\mu_Y(NT+F(T^2+T-1))}{F^2NT} \\ &= \frac{\mu_Y}{T}\left(FT^2+(N+F)T-F\right) \end{aligned} \tag{A6}$$

We are interested now in measuring the variance of z_j, $j = 1, \cdots, N$, which, by construction, we assume to be i.i.d. First, we note that the covariance matrix is:

$$\mathbf{C}^T\mathrm{sgn}(\hat{\boldsymbol{\varphi}})\mathrm{sgn}(\hat{\boldsymbol{\varphi}}^T)\mathbf{C} - \alpha^2\hat{\boldsymbol{\varphi}}\hat{\boldsymbol{\varphi}}^T$$

Then, the variance of z_j will be the expectation of the trace of this matrix divided by N. The second term is immediate to compute, as $\mathrm{E}\{\mathrm{Tr}[\hat{\boldsymbol{\varphi}}\hat{\boldsymbol{\varphi}}^T]\} = NT/F^2$, so we focus next on $\mathrm{Tr}[\mathbf{C}^T\mathrm{sgn}(\hat{\boldsymbol{\varphi}})\mathrm{sgn}(\hat{\boldsymbol{\varphi}}^T)\mathbf{C}] = \mathrm{Tr}[\mathbf{C}\mathbf{C}^T\mathrm{sgn}(\hat{\boldsymbol{\varphi}})\mathrm{sgn}(\hat{\boldsymbol{\varphi}}^T)]$. Again, we can write:

$$\mathrm{E}\{\mathrm{Tr}[\mathbf{C}\mathbf{C}^T\mathrm{sgn}(\hat{\boldsymbol{\varphi}})\mathrm{sgn}(\hat{\boldsymbol{\varphi}}^T)]\} = \mathrm{E}\left\{\sum_{i,j}(\mathbf{C}\mathbf{C}^T)_{i,j}\left(\mathrm{sgn}(\hat{\boldsymbol{\varphi}})\mathrm{sgn}(\hat{\boldsymbol{\varphi}}^T)\right)_{i,j}\right\}$$

Note that:

$$\begin{aligned}\mathbf{C}\mathbf{C}^T &= \sum_{m=1}^{T}\sum_{l=1}^{T}\hat{\boldsymbol{\phi}}^{(m)}(\hat{\boldsymbol{\phi}}^{(m)})^T\hat{\boldsymbol{\phi}}^{(l)}(\hat{\boldsymbol{\phi}}^{(l)})^T \\ &= \sum_{m=1}^{T}\sum_{l=1}^{T}\left(\sum_{n=1}^{N}\hat{\phi}_n^{(m)}\hat{\phi}_n^{(l)}\right)\hat{\boldsymbol{\phi}}^{(m)}(\hat{\boldsymbol{\phi}}^{(l)})^T\end{aligned}$$

Then,

$$\mathrm{E}\{(\mathbf{C}\mathbf{C}^T)_{i,j}\left(\mathrm{sgn}(\hat{\boldsymbol{\varphi}})\mathrm{sgn}(\hat{\boldsymbol{\varphi}}^T)\right)_{i,j}\} = \sum_{m=1}^{T}\sum_{l=1}^{T}\sum_{n=1}^{N}\mathrm{E}\left\{\hat{\phi}_n^{(m)}\hat{\phi}_n^{(l)}\hat{\phi}_i^{(m)}\hat{\phi}_j^{(l)}\mathrm{sgn}(\hat{\varphi}_i)\mathrm{sgn}(\hat{\varphi}_j)\right\} \tag{A7}$$

We analyze first the off-diagonal terms, i.e., $i \neq j$, when $\left\lfloor\frac{i}{F}\right\rfloor \neq \left\lfloor\frac{j}{F}\right\rfloor$, and consider separately the cases where $m \neq l$ and $m = l$. For the first case, the right-hand side of (A7) can be written as:

$$\begin{aligned}&\sum_{m=1}^{T}\sum_{\substack{l=1\\l\neq m}}^{T}\sum_{\substack{n=1\\ \lfloor\frac{n}{F}\rfloor\neq\lfloor\frac{i}{F}\rfloor\neq\lfloor\frac{j}{F}\rfloor}}^{N}\mathrm{E}\left\{\hat{\phi}_n^{(m)}\right\}\mathrm{E}\left\{\hat{\phi}_n^{(l)}\right\}\mathrm{E}\left\{\hat{\phi}_i^{(m)}\mathrm{sgn}(\hat{\varphi}_i)\right\}\mathrm{E}\left\{\hat{\phi}_j^{(l)}\mathrm{sgn}(\hat{\varphi}_j)\right\} \\ +\;& F\sum_{m=1}^{T}\sum_{\substack{l=1\\l\neq m}}^{T}\mathrm{E}\left\{(\hat{\phi}_i^{(m)})^2\hat{\phi}_i^{(l)}\mathrm{sgn}(\hat{\varphi}_i)\right\}\mathrm{E}\left\{\hat{\phi}_j^{(l)}\mathrm{sgn}(\hat{\varphi}_j)\right\} \\ +\;& F\sum_{m=1}^{T}\sum_{\substack{l=1\\l\neq m}}^{T}\mathrm{E}\left\{(\hat{\phi}_j^{(l)})^2\hat{\phi}_j^{(m)}\mathrm{sgn}(\hat{\varphi}_j)\right\}\mathrm{E}\left\{\hat{\phi}_i^{(m)}\mathrm{sgn}(\hat{\varphi}_i)\right\} \\ =\;& 2F\sum_{m=1}^{T}\sum_{\substack{l=1\\l\neq m}}^{T}\mathrm{E}\left\{(\hat{\phi}_i^{(m)})^2\hat{\phi}_i^{(l)}\mathrm{sgn}(\hat{\varphi}_i)\right\}\mathrm{E}\left\{\hat{\phi}_i^{(m)}\mathrm{sgn}(\hat{\varphi}_i)\right\} \\ =\;& \frac{2\mu_Y^2}{F}T(T-1)\end{aligned}$$

When $\left\lfloor \frac{i}{F} \right\rfloor \neq \left\lfloor \frac{j}{F} \right\rfloor$ and $m = l$, the right-hand side of (A7) can be written as:

$$\begin{aligned}
& \sum_{m=1}^{T} \sum_{\substack{n=1 \\ \lfloor \frac{n}{F} \rfloor \neq \lfloor \frac{i}{F} \rfloor \neq \lfloor \frac{j}{F} \rfloor}}^{N} \mathrm{E}\left\{(\hat{\phi}_n^{(m)})^2\right\} \mathrm{E}\left\{\hat{\phi}_i^{(m)} \mathrm{sgn}(\hat{\varphi}_i)\right\} \mathrm{E}\left\{\hat{\phi}_j^{(m)} \mathrm{sgn}(\hat{\varphi}_j)\right\} \\
+ \; & F \sum_{m=1}^{T} \mathrm{E}\left\{(\hat{\phi}_i^{(m)})^3 \mathrm{sgn}(\hat{\varphi}_i)\right\} \mathrm{E}\left\{\hat{\phi}_j^{(m)} \mathrm{sgn}(\hat{\varphi}_j)\right\} + F \sum_{m=1}^{T} \mathrm{E}\left\{(\hat{\phi}_j^{(m)})^3 \mathrm{sgn}(\hat{\varphi}_j)\right\} \mathrm{E}\left\{\hat{\phi}_i^{(m)} \mathrm{sgn}(\hat{\varphi}_i)\right\} \\
= \; & \frac{1}{F^2}\left(T(N-2F)\mu_Y^2 + 2F\mu_Y^2(3T-1)\right)
\end{aligned}$$

Now, we consider the rest of elements which satisfy that $\left\lfloor \frac{i}{F} \right\rfloor = \left\lfloor \frac{j}{F} \right\rfloor$, which are the N diagonal terms and the remaining $N(F-1)$ off-diagonal ones. The right-hand side of (A7) is:

$$\begin{aligned}
& \sum_{m=1}^{T} \sum_{l=1}^{T} \sum_{n=1}^{N} \mathrm{E}\{\hat{\phi}_n^{(m)} \hat{\phi}_n^{(l)} \hat{\phi}_i^{(m)} \hat{\phi}_i^{(l)}\} = F \sum_{m=1}^{T} \sum_{\substack{l=1 \\ l \neq m}}^{T} \mathrm{E}\left\{(\hat{\phi}_i^{(m)})^2\right\} \mathrm{E}\left\{(\hat{\phi}_i^{(l)})^2\right\} \\
+ \; & \sum_{m=1}^{T} \sum_{\substack{n=1 \\ \lfloor \frac{n}{F} \rfloor \neq \lfloor \frac{i}{F} \rfloor}}^{N} \mathrm{E}\left\{(\hat{\phi}_n^{(m)})^2\right\} \mathrm{E}\left\{(\hat{\phi}_i^{(m)})^2\right\} + F \sum_{m=1}^{T} \mathrm{E}\left\{(\hat{\phi}_i^{(m)})^4\right\} \\
= \; & \frac{1}{F^4}(FT(T-1) + T(N-F) + 3FT)
\end{aligned}$$

where, for the last summand, we have used the fact that, for a zero-mean Gaussian random variable X with variance σ^2, $\mathrm{E}\{X^4\} = 3\sigma^4$. Then, the sum in (A7) is:

$$\begin{aligned}
& \sum_{m=1}^{T} \sum_{l=1}^{T} \sum_{n=1}^{N} \mathrm{E}\left\{\hat{\phi}_n^{(m)} \hat{\phi}_n^{(l)} \hat{\phi}_i^{(m)} \hat{\phi}_j^{(l)} \mathrm{sgn}(\hat{\varphi}_i) \mathrm{sgn}(\hat{\varphi}_j)\right\} \\
= \; & N(N-F)\left[\frac{2\mu_Y^2}{F} T(T-1) + \frac{1}{F^2}\left(T(N-2F)\mu_Y^2 + 2F\mu_Y^2(3T-1)\right)\right] \\
+ \; & \frac{N}{F^3}\left(FT(T-1) + T(N-F) + 3FT\right)
\end{aligned}$$

and the resulting variance of z_j is:

$$\begin{aligned}
\mathrm{Var}\{z_j\} \; = \; & (N-F)\left(\frac{2\mu_Y^2}{F} T(T-1) + \frac{\mu_Y^2}{F^2} T(N-2F) + \frac{2\mu_Y^2}{F}(3T-1)\right) \\
+ \; & \frac{1}{F^3}(FT(T-1) + T(N-F) + 3FT) - \frac{T}{F^2}\mu_Y^2 \left(N + \frac{F}{T}(T^2+T-1)]\right)^2 \\
= \; & -\frac{\mu_Y^2}{FT}\left(FT^4 + 4FT^3 + (N+F)T^2 - 4FT + F\right) + \frac{T}{F^3}\left(N + F(T+1)\right) \qquad \text{(A8)}
\end{aligned}$$

Appendix A.4.2. Decomposition for Orthogonal Projectors

Now, for orthogonal projectors, we take a different route. Again, we define matrix $\mathbf{C} \doteq [\mathbf{c}_0, \cdots, \mathbf{c}_N]$, and we note that $\mathbf{C} = \hat{\mathbf{\Upsilon}}^T = \hat{\mathbf{\Upsilon}}$. Recall also properties (13) and (14). We can collect all products $\mathbf{c}_j^T \mathrm{sgn}(\hat{\boldsymbol{\varphi}})$, $j = 1, \cdots, N$ in a vector obtained as $\mathbf{C}^T \mathrm{sgn}(\hat{\boldsymbol{\varphi}}) = \hat{\mathbf{\Upsilon}} \mathrm{sgn}(\hat{\boldsymbol{\varphi}})$.

We are interested in computing the cross-product of this vector and $\hat{\boldsymbol{\varphi}}$. Then, we can write:

$$\hat{\boldsymbol{\varphi}}^T \hat{\mathbf{\Upsilon}} \mathrm{sgn}(\hat{\boldsymbol{\varphi}}) = \frac{1}{F} \hat{\boldsymbol{\varphi}}^T \mathrm{sgn}(\hat{\boldsymbol{\varphi}})$$

Assuming i.i.d. components in $\hat{\boldsymbol{\varphi}}$, this implies that, for the orthogonal projectors, the cross-correlation of $\mathbf{c}_j^T \mathrm{sgn}(\hat{\boldsymbol{\varphi}})$ and $\hat{\varphi}_j$ is the same as $\frac{1}{F}\mathrm{sgn}(\hat{\varphi}_j)$ and $\hat{\varphi}_j$. We can then compute

$\text{E}\{\hat{\varphi}_j \text{sgn}(\hat{\varphi}_j)\} = \text{E}\{|\hat{\varphi}_j|\}$. Modeling $\hat{\varphi}_j$ as $\mathcal{N}(0, T/(FN))$, we find that $\text{E}\{|\hat{\varphi}_j|\} = \sqrt{\frac{2T}{\pi FN}}$. Furthermore, $\text{E}\{||\hat{\boldsymbol{\varphi}}||^2\} = \text{E}\{\hat{\boldsymbol{\varphi}}^T\hat{\boldsymbol{\varphi}}\} = \text{E}\{\boldsymbol{\varphi}^T\boldsymbol{\Theta}^T\boldsymbol{\Theta}\boldsymbol{\varphi}\} = \frac{T}{F}$.

The existence of a positive cross-correlation suggests writing $\frac{1}{F}\text{sgn}(\hat{\varphi}_j) = \alpha\hat{\varphi}_j + \tilde{n}_j$, with α a suitable positive constant and $\tilde{\mathbf{n}}$ a zero-mean noise vector uncorrelated with $\hat{\boldsymbol{\varphi}}$. Taking the cross-product and the expectation:

$$\frac{1}{F}\text{E}\{\hat{\boldsymbol{\varphi}}^T \text{sgn}(\hat{\boldsymbol{\varphi}})\} = \frac{N}{F}\text{E}\{|\hat{\varphi}_j|\} = \alpha\text{E}\{||\hat{\boldsymbol{\varphi}}||^2\} + \text{E}\{\hat{\boldsymbol{\varphi}}^T\tilde{\mathbf{n}}\} = \alpha\text{E}\{||\hat{\boldsymbol{\varphi}}||^2\}$$

Therefore, we find that:

$$\alpha = \frac{N \cdot \text{E}\{|\hat{\varphi}_j|\}}{F \cdot \text{E}\{||\hat{\boldsymbol{\varphi}}||^2\}} = \sqrt{\frac{2N}{\pi TF}} \tag{A9}$$

Now, it is easy to measure the second order moment of $\tilde{n}_j$ since:

$$\begin{aligned} \text{E}\{\tilde{n}_j^2\} &= \text{E}\{(\frac{1}{F}\text{sgn}(\hat{\varphi}_j) - \alpha\hat{\varphi}_j)^2\} \\ &= \frac{1}{F^2} + \alpha^2\text{E}\{\hat{\varphi}_j^2\} - \frac{2\alpha}{F}\text{E}\{\hat{\varphi}_j\text{sgn}(\hat{\varphi}_j)\} \\ &= \frac{1}{F^2}\left(1 - \frac{2}{\pi}\right) \end{aligned}$$

With the above characterization, we can see that:

$$\begin{aligned} \mathbf{C}^T\text{sgn}(\hat{\boldsymbol{\varphi}}) &= F\alpha\hat{\boldsymbol{\Upsilon}}\hat{\boldsymbol{\varphi}} + F\hat{\boldsymbol{\Upsilon}}\tilde{\mathbf{n}} \\ &= \alpha\hat{\boldsymbol{\varphi}} + \mathbf{z} \end{aligned}$$

Therefore, we have:

$$\text{Tr}[\text{E}\{\mathbf{z}\mathbf{z}^T\}] = F^2 \cdot \text{Tr}[\text{E}\{\hat{\boldsymbol{\Upsilon}}\tilde{\mathbf{n}}\tilde{\mathbf{n}}^T\hat{\boldsymbol{\Upsilon}}\}] = F \cdot \text{E}\{\text{Tr}[\tilde{\mathbf{n}}\tilde{\mathbf{n}}^T\hat{\boldsymbol{\Upsilon}}]\} = F \cdot \text{E}\left\{\sum_{i,j}(\tilde{\mathbf{n}}\tilde{\mathbf{n}}^T)_{i,j}(\hat{\boldsymbol{\psi}})_{i,j}\right\}$$

Then,

$$\text{E}\left\{(\tilde{\mathbf{n}}\tilde{\mathbf{n}}^T)_{i,j}(\hat{\boldsymbol{\psi}})_{i,j}\right\} = \text{E}\left\{\tilde{n}_i\tilde{n}_j\right\}\text{E}\left\{\sum_{m=1}^{T}\hat{\phi}_i^{(m)}\hat{\phi}_j^{(m)}\right\}$$

When $\left\lfloor\frac{i}{F}\right\rfloor \neq \left\lfloor\frac{j}{F}\right\rfloor$, the expectation above is zero. For the NF remaining terms, that is, when $\left\lfloor\frac{i}{F}\right\rfloor = \left\lfloor\frac{j}{F}\right\rfloor$, we have $\tilde{n}_i = \tilde{n}_j$ and $\hat{\phi}_i^{(m)} = \hat{\phi}_j^{(m)}$. Thus:

$$\text{E}\{\tilde{n}_j^2\}\sum_{m=1}^{T}\text{E}\left\{(\hat{\phi}_j^{(m)})^2\right\} = \frac{1}{F^2}\left(1 - \frac{2}{\pi}\right)\frac{T}{NF} = \frac{T}{NF^3}\left(1 - \frac{2}{\pi}\right)$$

Finally, we can compute the variance of z_j as follows:

$$\text{Var}\{z_j\} = \frac{1}{N}\text{Tr}[\text{E}\{\mathbf{z}\mathbf{z}^T\}] = \left(1 - \frac{2}{\pi}\right)\frac{T}{FN} \tag{A10}$$

Appendix A.5. Adam: Analysis with Denoising and Watermarking

Here, we join the denoising and watermarking cost functions and follow a similar approach to that in Section 3.2.5. Let $\mathbf{c}$ be a random vector of length N, now we should build a random projection matrix, $\hat{\boldsymbol{\Phi}} = \boldsymbol{\Theta}\boldsymbol{\Phi}$—with Gaussian or orthogonal projectors—, as a basis to generate samples of Ξ from (11) and to build $\mathbf{c}$ following (29).

We also consider that $\boldsymbol{\eta} = \boldsymbol{\eta}_d + \boldsymbol{\eta}_{wm}$ is a random vector of length N, where $\boldsymbol{\eta}_d$ and $\boldsymbol{\eta}_{wm}$ represent the denoising and watermarking update terms, respectively. The components of $\boldsymbol{\eta}_d$ can be computed from realizations of D and V like in Section 3.3 that is, $\eta_d = \delta/(\sqrt{\delta^2+\nu})$. On the other hand, $\boldsymbol{\eta}_{wm}$ has the same definition than in Section 3.2.5.

Let Ω be a random variable with the same distribution as $\mathbf{c}^T\boldsymbol{\eta}_d$ and let M_0, P, Q and R be random variables for which $m_{0,j}$, p_j, q_j are r_j are realizations, respectively. Then, we have:

$$\begin{aligned}
M_0 &= -\frac{\lambda}{2}\Xi - D\\
P &= \Xi^2\\
Q &= -\Xi\Omega - \Gamma\Xi(\alpha\Xi + Z)\\
R &= \frac{1}{4}\left(\Omega^2 + 2\Gamma(\alpha\Xi+Z)\Omega + \Gamma^2(\alpha\Xi+Z)^2\right)
\end{aligned}$$

Additionally, let S be a random variable for which s_j is a realization. Now, it must include the contribution due to the denoising cost function so that:

$$\begin{aligned}
S &= \frac{\lambda^2}{4}\left(P - \frac{\beta_2}{1-\beta_2}\mu Q + \frac{\beta_2+\beta_2^2}{(1-\beta_2)^2}\mu^2 R\right) + D^2 + V\\
&= \frac{\lambda^2}{4}\left[\Xi^2 + \frac{\beta_2}{1-\beta_2}\mu\left(\Xi\Omega + \Gamma\Xi(\alpha\Xi+Z)\right)\right.\\
&+ \left.\frac{\beta_2+\beta_2^2}{(1-\beta_2)^2}\frac{\mu^2}{4}\left(\Omega^2 + 2\Gamma(\alpha\Xi+Z)\Omega + \Gamma^2(\alpha\Xi+Z)^2\right)\right] + D^2 + V
\end{aligned}$$

Therefore, for a given realization (ξ, z, δ, ν) of (Ξ, Z, D, V), we must solve the following fourth degree equation to get samples of Γ (notice that, in order to generate samples, we must previously build the random vectors $\boldsymbol{\eta}_d$ and $\mathbf{c}$):

$$A_1\gamma^4 + A_2\gamma^3 + A_3\gamma^2 + A_4\gamma + A_5 = 0 \tag{A11}$$

where:

$$A_1 \doteq \frac{\lambda^2}{4}\frac{(\beta_2+\beta_2^2)}{(1-\beta_2)^2}\frac{\mu^2}{4}(\alpha\xi+z)^2$$

$A_2 = A_{21} + A_{22}$ with A_{21} and A_{22} given by:

$$\begin{aligned}
A_{21} &\doteq \frac{\lambda^2}{4}\frac{\beta_2}{(1-\beta_2)}\mu\xi(\alpha\xi+z)\\
A_{22} &\doteq \frac{\lambda^2}{8}\frac{\beta_2+\beta_2^2}{(1-\beta_2)^2}\mu^2(\alpha\xi+z)\left(\Omega + \frac{\delta}{\sqrt{\delta^2+\nu}}\operatorname{sgn}(\xi)(\alpha\xi+z)\right)
\end{aligned}$$

$A_3 = A_{31} + A_{32} + A_{33} + A_{34}$ with the following definitions:

$$\begin{aligned}
A_{31} &\doteq \frac{\lambda^2}{4}\xi^2\\
A_{32} &\doteq \frac{\lambda^2}{4}\frac{\beta_2}{1-\beta_2}\mu\xi\left(\Omega + 2\frac{\delta}{\sqrt{\delta^2+\nu}}\operatorname{sgn}(\xi)(\alpha\xi+z)\right)\\
A_{33} &\doteq \frac{\lambda^2}{16}\frac{\beta_2+\beta_2^2}{(1-\beta_2)^2}\mu^2\left(\Omega^2 + \frac{\delta^2}{\delta^2+\nu}(\alpha\xi+z)^2 + 4\frac{\delta}{\sqrt{\delta^2+\nu}}\operatorname{sgn}(\xi)(\alpha\xi+z)\Omega\right)\\
A_{34} &\doteq \delta^2+\nu
\end{aligned}$$

$A_4 = A_{41} + A_{42} + A_{43} + A_{44}$ where:

$$\begin{aligned}
A_{41} &\doteq \frac{\lambda^2}{2}\frac{\delta}{\sqrt{\delta^2+\nu}}\mathrm{sgn}(\xi)\xi^2 \\
A_{42} &\doteq \frac{\lambda^2}{4}\frac{\beta_2}{1-\beta_2}\mu\frac{\delta}{\sqrt{\delta^2+\nu}}\xi\left(2\mathrm{sgn}(\xi)\Omega + \frac{\delta}{\sqrt{\delta^2+\nu}}(\alpha\xi+z)\right) \\
A_{43} &\doteq \frac{\lambda^2}{8}\frac{\beta_2+\beta_2^2}{(1-\beta_2)^2}\mu^2\frac{\delta}{\sqrt{\delta^2+\nu}}\Omega\left(\frac{\delta}{\sqrt{\delta^2+\nu}}(\alpha\xi+z) + \mathrm{sgn}(\xi)\Omega\right) \\
A_{44} &\doteq 2\frac{\delta}{\sqrt{\delta^2+\nu}}\mathrm{sgn}(\xi)(\delta^2+\nu)
\end{aligned}$$

Finally, $A_5 = A_{51} + A_{52} + A_{53} + A_{54}$ with:

$$\begin{aligned}
A_{51} &\doteq \frac{\lambda^2}{4}\xi^2\left(\frac{\delta^2}{\delta^2+\nu} - 1\right) \\
A_{52} &\doteq \frac{\lambda^2}{4}\frac{\beta_2}{1-\beta_2}\mu\frac{\delta^2}{\delta^2+\nu}\xi\Omega \\
A_{53} &\doteq \frac{\lambda^2}{16}\frac{\beta_2+\beta_2^2}{(1-\beta_2)^2}\mu^2\frac{\delta^2}{\delta^2+\nu}\Omega^2 \\
A_{54} &\doteq \lambda\delta\xi
\end{aligned}$$

Finally, we can generate samples of $\Delta\mathbf{w}$ as:

$$\Delta w \approx k \cdot \mu \cdot \left(\frac{\delta}{\sqrt{\delta^2+\nu}} + \gamma \cdot \mathrm{sgn}(\xi)\right) \tag{A12}$$

Appendix B. Verification of Assumptions

Appendix B.1. Affine Growth Hypothesis for the Weights

In order to check the affine growth hypothesis that we introduced in (18), we extract the weight values from the embedding layer on each iteration. Figures A2 and A3 show the evolution with k of four randomly selected weights when we respectively use: (i) SGD with orthogonal projectors, and (ii) Adam with Gaussian projectors. As is evident, the hypothesis holds quite accurately for these particular examples.

For a more general approach encompassing all weights, we carry out lineal regression on the evolution of each individual weight with k. Then, for each $j \in \{1, \cdots, N\}$, we measure the correlation coefficient, ρ_j, between the observed values—the experimental weight values—and the predictor values. Figure A4 represents the Empirical Cumulative Distribution Function (ECDF) of ρ_j when using SGD with orthogonal projectors and Adam with both Gaussian and orthogonal projectors. We can state that, when we use Gaussian projectors with Adam optimization, for 90% of the weights at the embedding layer $\rho_j > 0.9410$ and for 80% of them $\rho_j > 0.9970$. Moreover, the affine hypothesis is even stronger when using orthogonal projectors, both with SGD or Adam optimization, since for 95% of the weights at the embedding layer $\rho_j > 0.9975$.

Because these percentages show a high-linear behavior, we can confirm the validity of the affine hypothesis for the weights that is key to our theoretical analysis.

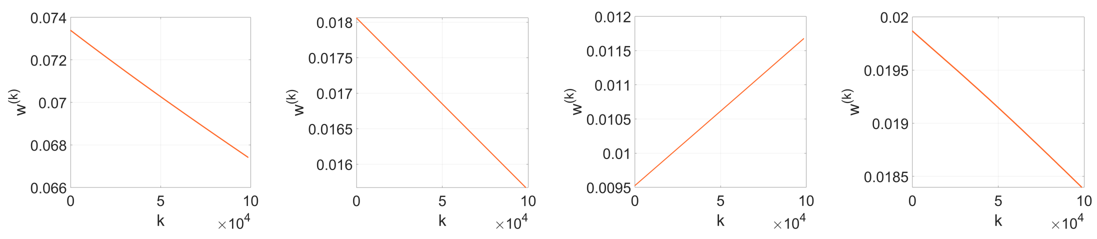


Figure A2. Evolution with k of four randomly selected weights. SGD optimization with orthogonal projectors, $\lambda = 20$, $T = 256$.

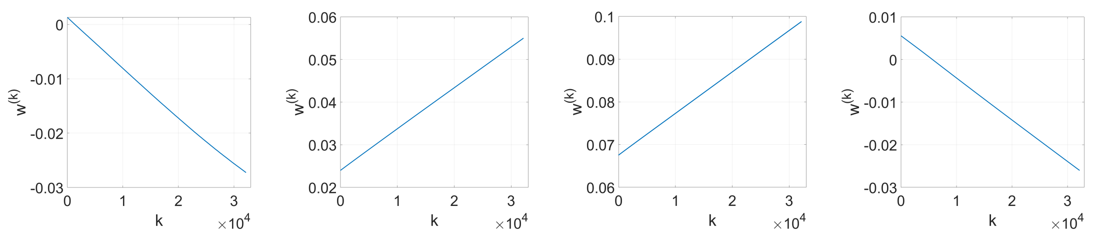


Figure A3. Evolution with k of four randomly selected weights. Adam optimization with Gaussian projectors, $\lambda = 1$, $T = 256$.

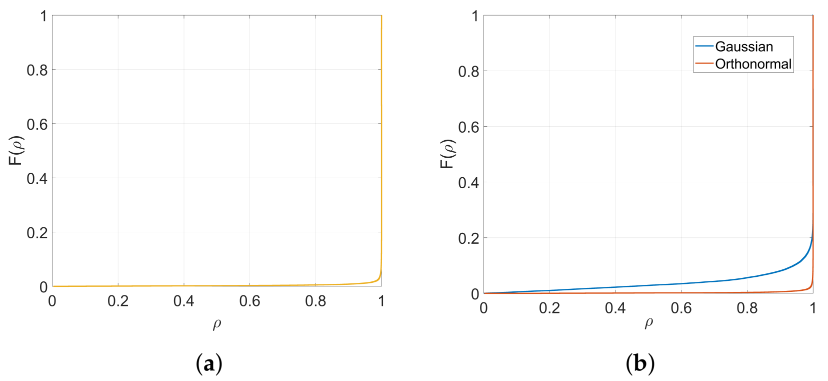


Figure A4. ECDF of the correlation coefficient ρ between the observed values of the weights over k and their predicted affine evolution. (**a**) SGD with orthogonal projectors, $\lambda = 20$; (**b**) Adam with Gaussian, $\lambda = 1$ and orthogonal projectors, $\lambda = 10$.

Appendix B.2. Negligibility of Weights at $k = 0$

We first check the validity of the approximation $\mathbf{w}^{(0)} \approx 0$ introduced in the analysis for SGD optimization. From (23), we can get $\boldsymbol{\eta}$ from the parameter settings detailed in Section 6.1 for the orthogonal case. We compute the gradient $\mathbf{g}_\nabla \doteq \boldsymbol{\nabla}_\mathbf{w} f(\mathbf{w})$ following (20) and also its approximation $\mathbf{g}'_\nabla \doteq -(\lambda/2)\hat{\boldsymbol{\varphi}} + (\lambda k \mu/4)\hat{\boldsymbol{\Upsilon}}\boldsymbol{\eta}$. In order to prove that $\mathbf{w}^{(0)} \approx 0$, we can show that the preserved term, that is, the approximation $\mathbf{g}'$, is much larger than the discarded term $\mathbf{g}_\nabla - \mathbf{g}'_\nabla = (\lambda/4)\hat{\boldsymbol{\Upsilon}}\mathbf{w}^{(0)}$, using the Normalized Mean Squared Error (NMSE). Let $\mathbf{x}$ be a N-length vector and $\mathbf{x}'$ its approximation, then the corresponding NMSE is defined as NMSE $= ||\mathbf{x} - \mathbf{x}'||^2/||\mathbf{x}'||^2$. Using this definition on $\mathbf{g}_\nabla$ and $\mathbf{g}'_\nabla$, we get an NMSE of $3.6464 \cdot 10^{-6}$, a very small value that supports our assumption $\mathbf{w}^{(0)} \approx 0$.

Now, we check the same approximation for the Adam analysis. From (43), we can generate samples of γ using the parameter settings in Section 6.1. Let us recall the following definitions:

$$\begin{aligned} \mathbf{m}_0 &= -\hat{\boldsymbol{\varphi}} + \frac{1}{2}\hat{\boldsymbol{\Upsilon}}\mathbf{w}^{(0)} \\ p_j &= a_j - \mathbf{b}_j^T\mathbf{w}^{(0)} + \frac{1}{4}\mathbf{c}_j^T\mathbf{w}^{(0)}(\mathbf{w}^{(0)})^T\mathbf{c}_j \\ q_j &= -\mathbf{b}_j^T\boldsymbol{\eta} + \frac{1}{2}\mathbf{c}_j^T\mathbf{w}^{(0)}\boldsymbol{\eta}^T\mathbf{c}_j \end{aligned}$$

for $j = 1, \cdots, N$. Additionally, we define their corresponding approximations as:

$$\begin{aligned} \mathbf{m}_0' &= -\hat{\boldsymbol{\varphi}} \\ p_j' &= a_j \\ q_j' &= -\mathbf{b}_j^T\boldsymbol{\eta} \end{aligned}$$

For Gaussian projectors, we get a NMSE of $3.9803 \cdot 10^{-3}$, $5.2153 \cdot 10^{-3}$ and $1.7148 \cdot 10^{-3}$, for the approximations made for $\mathbf{m}_0$, $\mathbf{p}$ and $\mathbf{q}$, respectively. For orthogonal projectors, we get a NMSE of $3.6093 \cdot 10^{-6}$, $5.1049 \cdot 10^{-6}$ and $1.8010 \cdot 10^{-6}$, for the approximations made for vectors $\mathbf{m}_0$, $\mathbf{p}$ and $\mathbf{q}$, respectively. As we see, these are small enough values to consider that our hypothesis $\mathbf{w}^{(0)} \approx \mathbf{0}$ is reasonable for the analysis in Section 3.2.

References

1. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV '15), Santiago, Chile, 7–13 December 2015; IEEE Computer Society: Washington, DC, USA, 2015; pp. 1026–1034.
2. Nassif, A.B.; Shahin, I.; Attili, I.; Azzeh, M.; Shaalan, K. Speech Recognition Using Deep Neural Networks: A Systematic Review. *IEEE Access* **2019**, 7, 19143–19165. [CrossRef]
3. Le Merrer, E.; Pérez, P.; Trédan, G. Adversarial Frontier Stitching for Remote Neural Network Watermarking. *arXiv* **2017**, arXiv:1711.01894.
4. Adi, Y.; Baum, C.; Cissé, M.; Pinkas, B.; Keshet, J. Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring. *arXiv* **2018**, arXiv:1802.04633.
5. Uchida, Y.; Nagai, Y.; Sakazawa, S.; Satoh, S. Embedding Watermarks into Deep Neural Networks. In Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval (ICMR '17), Bucharest, Romania, 6–9 June 2017; Association for Computing Machinery: New York, NY, USA, 2017; pp. 269–277.
6. Nagai, Y.; Uchida, Y.; Sakazawa, S.; Satoh, S. Digital Watermarking for Deep Neural Networks. *Int. J. Multimed. Inf. Retr.* **2018**, 7, 3–16. [CrossRef]
7. Cox, I.J.; Kilian, J.; Leighton, F.T.; Shamoon, T. Secure Spread Spectrum Watermarking for Multimedia. *IEEE Trans. Image Process.* **1997**, 6, 1673–1687. [CrossRef] [PubMed]
8. Bottou, L. Online Algorithms and Stochastic Approximations. In *Online Learning and Neural Networks*; Saad, D., Ed.; Cambridge University Press: Cambridge, UK, 1998.
9. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR '15), San Diego, CA, USA, 7–9 May 2015.
10. Rouhani, B.D.; Chen, H.; Koushanfar, F. DeepSigns: A Generic Watermarking Framework for IP Protection of Deep Learning Models. *arXiv* **2018**, arXiv:1804.00750.
11. Balles, L.; Hennig, P. Dissecting Adam: The Sign, Magnitude and Variance of Stochastic Gradients. In Proceedings of the 2018 International Conference on Machine Learning (ICML '18), Stockholm, Sweden, 10–15 July 2018.

12. Wilson, A.C.; Roelofs, R.; Stern, M.; Srebro, N.; Recht, B. The Marginal Value of Adaptive Gradient Methods in Machine Learning. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS '17), Long Beach, CA, USA, 4–9 December 2017; Curran Associates Inc.: Red Hook, NY, USA; pp. 4151–4161.
13. Kullback, S.; Leibler, R.A. On Information and Sufficiency. *Ann. Math. Stat.* **1951**, *22*, 79–86. [CrossRef]
14. Zhang, K.; Zuo, W.; Zhang, L. FFDNet: Toward a Fast and Flexible Solution for CNN-Based Image Denoising. *IEEE Trans. Image Process.* **2018**, 27, 4608–4622. [CrossRef] [PubMed]
15. Fan, L.; Zhang, F.; Fan, H.; Zhang, C. Brief Review of Image Denoising Techniques. *Vis. Comput. Ind. Biomed. Art* **2019**, *2*, 7. [CrossRef] [PubMed]
16. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the 32nd International Conference on Machine Learning (ICML '15), Lille, France, 6–11 July 2015, pp. 448–456.
17. Sun, S.; Cao, Z.; Zhu, H.; Zhao, J. A Survey of Optimization Methods From a Machine Learning Perspective. *IEEE Trans. Cybern.* **2020**, 50, 3668–3681. [CrossRef] [PubMed]
18. Duchi, J.; Hazan, E.; Singer, Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Mach. Learn. Res.* **2011**, *12*, 2121–2159.
19. Tieleman, T.; Hinton, G. *Lecture 6.5—RMSProp, COURSERA: Neural Networks for Machine Learning*; University of Toronto: Toronto, ON, Canada, 2012.
20. Wang, T.; Kerschbaum, F. Attacks on Digital Watermarks for Deep Neural Networks. In Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '19), Brighton, UK, 12–17 May 2019; pp. 2622–2626.
21. Burden, R.L.; Faires, J.D. *Numerical Analysis*, 9th ed.; Brooks/Cole: Boston, MA, USA, 2010.
22. Geyer, C.J. Practical Markov Chain Monte Carlo. *Stat. Sci.* **1992**, 7, 493–497. [CrossRef]
23. Cachin, C. An Information-Theoretic Model for Steganography. In *Information Hiding*; Lecture Notes in Computer Science; Aucsmith, D., Ed.; Springer: Berlin/Heidelberg, Germany, 1998; Volume 1525.
24. Comesaña, P. Detection and information theoretic measures for quantifying the distinguishability between multimedia operator chains. In Proceedings of the IEEE Workshop on Information Forensics and Security (WIFS12), Tenerife, Spain, 2–5 December 2012.
25. Barni, B.; Tondi, B. The Source Identification Game: An Information-Theoretic Perspective. *IEEE Trans. Inf. Forensics Secur.* **2013**, 8, 450–463. [CrossRef]
26. Tassano, M.; Delon, J.; Veit, T. An Analysis and Implementation of the FFDNet Image Denoising Method. *Image Process. Line* **2019**, 9, 1–25. [CrossRef]
27. Ma, K.; Duanmu, Z.; Wu, Q.; Wang, Z.; Yong, H.; Li, H.; Zhang, L. Waterloo Exploration Database: New Challenges for Image Quality Assessment Models. *IEEE Trans. Image Process.* **2017**, *26*, 1004–1016.
28. Franzen, R. Kodak Lossless True Color Image Suite. 1999. Available online: http://r0k.us/graphics/kodak (accessed on 22 September 2020).
29. Martin, D.; Fowlkes, C.; Tal, D.; Malik, J. A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In Proceedings of the 8th Int'l Conf. Computer Vision (ICCV 2001), Vancouver, BC, Canada, 7–14 July 2001; pp. 416–423.
30. Wilson, S.G. *Digital Modulation and Coding*; Pearson: London, UK, 1995.