Bilevel Learning of Deep Representations

Jordan Frecon [1] Saverio Salzo [2] Massimiliano Pontil [2 3]

Abstract

We present a framework based on bilevel optimization for learning multilayer, deep data representations. While the lower-level problem implicitly defines the representation through the critical point of a learnable objective, the upper-level problem optimizes the representation mapping. We reformulate the problem via a majorization-minimization algorithm. On one hand, for some quadratic majorants, we show that the bilevel problem reduces down to the training of a feed-forward neural network. On the other hand, for majorants based on Bregman distances, we introduce a new neural network architecture involving the inverse of the activation function. We argue theoretically that the novel architecture may have better mathematical properties than standard networks. Numerical experiments show that the proposed variant benefits from better training behaviors, resulting in more accurate prediction.

1. Introduction

The past decades have seen an overwhelming interest in neural networks due to their empirical success in numerous and disparate applications, ranging from medical imaging (Zhou et al., 2019; Lassau et al., 2021) to self driving vehicles (Blin et al., 2019), among others. Consequently, they have received a great interest from the machine learning community (see e.g. Maggu et al., 2020, and references therein). However, to date, research in deep learning has mostly been application driven. A large body of work have proposed increasingly complex architectures for specific tasks, but much fewer attempts have been made to elucidate the reasons behind their success.

A key aspect of deep neural networks is their ability to learn a representation from raw inputs. In this paper, rather than directly writing the representation mapping as a prescribed compositional form, our framework looks the representation from the perspective of bilevel optimization (see Franceschi et al., 2018; Grazzi et al., 2020, and references therein). Within this framework, the lower-level problem performs feature representation while the upper-level problem optimizes the representation mapping. A main insight is to introduce a constraint on the representation as the inclusion to the critical points of a learnable objective function. It is formed by summing a parametric function and a prescribed function which may be nonconvex.

To overcome the difficulty behind the bilevel problem we convert it into a sequence of simpler ones, via a general majorization-minimization (MM) approximation scheme. Learning the representation can be seen as optimizing the majorant functions used in the MM algorithm. We show how this iterative scheme naturally gives rise to novel multilayer networks for which we establish basic mathematical properties. In particular, we argue that under certain conditions they may yield converging sequences resulting in a more stable and effective model to train.

Contributions and Organization. In Sec. 2, we present the proposed framework and in particular the class of MM algorithms and their associated majorants. From this perspective, we show in Sec. 3 that, for some quadratic majorants, we recover standard feed-forward neural networks. By elaborating on other classes of majorants, we propose in Sec. 4 a new type of neural network layer, namely the *Bregman feed-forward layer*, which additionally involves the inverse of the activation operator. This setting provides a novel view on neural networks which allows us to interpret activation operators as Bregman projection of convex functions as opposed to proximity operator of nonconvex functions. Within this framework we devise theoretical guarantees as described in Sec. 5. The practical benefits of this new type of feed-forward networks are assessed on both synthetic and real datasets in Sec. 6.

Previous Work. Bilevel optimization formulations have recently been devised in the setting of meta-learning and hyperparameter optimization (Grazzi et al., 2020; Franceschi et al., 2018). Probably most related to our work is (Combettes & Pesquet, 2020) where the authors have shown that a wide range of activation operators are actually proximity operators of possibly nonconvex functions. In addition, the authors have provided a thorough analysis with tools from

[1]Normandie Univ, INSA Rouen, UNIROUEN, UNIHAVRE, LITIS. Saint-Etienne-du-Rouvray, France [2]Computational Statistics and Machine Learning, Istituto Italiano di Tecnologia. Genova, Italy [3]Department of Computer Science, University College. London, United Kingdom. Correspondence to: Jordan Frecon <jordan.frecon@insa-rouen.fr>.

monotone operator theory in order to study a class of neural networks and their asymptotic properties. In addition, feed-forward networks have been studied in (Bibi et al., 2019), where the authors have shown that the forward pass through one feed-forward layer is equivalent to a single step of a forward-backward algorithm. In addition, recent studies have built deep models by unrolling particular optimization algorithms (Monga et al., 2020; Bertocchi et al., 2020).

Notations. Let $\mathcal{X}$ and $\mathcal{Y}$ be two Hilbert spaces. For every lower semicontinuous extended real-valued function $f\colon \mathcal{X} \to \mathbb{R} \cup \{\pm\infty\}$ we denote by ∂f the subdifferential of f and set $\operatorname{crit} f = \{x \in \mathcal{X} \,|\, 0 \in \partial f(x)\}$. In addition, we let $\Gamma_0(\mathcal{X})$ be the space of functions $h\colon \mathcal{X} \to]-\infty,+\infty]$ closed proper and convex, $\mathcal{B}(\mathcal{X},\mathcal{Y})$ be the space of bounded operators from $\mathcal{X}$ to $\mathcal{Y}$, and $\mathcal{F}(\mathcal{X})$ be the space of functions $f\colon \mathcal{X} \to]-\infty,+\infty]$ which are proper, lower semicontinuous, definable on an o-minimal structure on $\mathbb{R}$ large enough to include most of the Bregman divergences used in applications (Attouch et al., 2010), locally Lipschitz continuous and subdifferentially regular. In addition, $\mathcal{F}_{\text{smooth}}(\mathcal{X}) \subset \mathcal{F}(\mathcal{X})$ denotes the subspace of $\mathbb{R}$-valued Lipschitz smooth functions. For $N \in \mathbb{N}_+^*$, we let $[N] = \{1,\ldots,N\}$. For every set $\mathcal{C}$ we denote by $\operatorname{int}\mathcal{C}$ the interior of $\mathcal{C}$.

2. Bilevel Framework

In this section, we present our bilevel framework for representation learning.

Given some training data set $\{x_i, y_i\}_{i=1}^N$ made of N samples, we propose to learn a representation mapping h such that, for every $i \in [N]$, the learned features $h(x_i)$ from the input data $x_i \in \mathbb{R}^d$ are more amenable for predicting $y_i \in \mathbb{R}^K$. In other words, this means that the predicted target $\hat{y}_i$ can be written as a simple transformation ψ of $h(x_i)$, i.e., $\hat{y}_i = \psi(h(x_i))$. The transformation ψ can model a large variety of operations popularly used such that a simple linear layer or a linear layer followed by a softmax operator. The closeness between $\hat{y}_i$ and the true target y_i is measured by a given function ℓ such as the mean squared error or the cross-entropy. A key feature of our framework is to implicitly define the learned representation $h(x_i)$ through the critical points of some objective function. We consider that the objective can be written as $f_\theta + g$ where $g \in \mathcal{F}(\mathbb{R}^d)$ is a known simple[1] function and $f_\theta \in \mathcal{F}_{\text{smooth}}(\mathbb{R}^d)$ is a possibly nonconvex function depending on parameters θ. Hence, here g acts as a regularizer while the choice of θ permits to tune f_θ and therefore also the representation h. In order to highlight the dependency, we denote $h(x_i, \theta)$. We provide a one-dimensional illustration in Figure 1 where g is the indicator function of the $[0,1]$ interval and the set of critical points is $\{0, 1/2, 1\}$. To learn both ψ and θ, we consider the

[1]Simple is meant in the sense that its proximity operator has a closed-form expression.

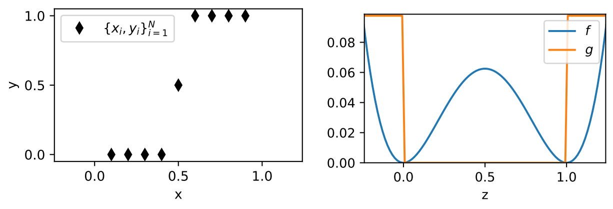

Figure 1. One dimensional example. Left: illustration of the dataset $\{x_i, y_i\}_{i=1}^N$. Right: suggestion of f and $g = \iota_{[0,1]}$ so that $\operatorname{crit}(f+g) = \{0, 1/2, 1\}$.

following bilevel problem.

Problem 2.1 (Original Bilevel Problem)**.** Given a training data set $\{x_i, y_i\}_{i=1}^N$ where $\{x_i, y_i\} \in \mathbb{R}^d \times \mathbb{R}^K$ for every $i \in [N]$, a function $g \in \mathcal{F}(\mathbb{R}^d)$, a mapping $h\colon \mathbb{R}^d \times \Theta \to \mathbb{R}^d$ and a family of functions $(f_\theta)_{\theta\in\Theta}$, with $f_\theta \in \mathcal{F}_{\text{smooth}}(\mathbb{R}^d)$, solve

$$\underset{\substack{\psi \in \mathcal{B}(\mathbb{R}^d, \mathbb{R}^K) \\ \theta \in \Theta}}{\text{minimize}} \sum_{i=1}^N \ell(\psi(h(x_i;\theta)), y_i)$$
$$\text{s.t.} \quad (\forall i \in [N]), \quad h(x_i;\theta) \in \operatorname{crit}(f_\theta + g). \quad (1)$$

Here $h(x_i, \theta)$ represents the limit point of a converging algorithm having parameter θ and initial point x_i. Moreover, $f_\theta + g$ is the function on which the algorithm (with parameter θ) is applied. In order to specify a class Θ for which the N constraints of (1) are satisfied, we consider an abstract algorithm $\mathcal{A}(x_i, \theta)$ which aims at solving

$$0 \in \partial(f_\theta + g)(z_i) \quad (2)$$

and takes as initial point x_i. Note that (2) is reminiscent of the deep equilibrium model (Bai et al., 2019), for which the inclusion in (1) is replaced by a fixed point equation (see Grazzi et al., 2020; Miller & Hardt, 2019, for a discussion).

To overcome the difficulty due to the non-convexity of f_θ, we propose to build a sequence by resorting to a majorization-minimization (MM) strategy. The underlying idea behind the MM algorithm is to convert a hard optimization problem into a sequence of simpler ones. Its principle relies on iteratively minimizing a majorizing surrogate of the objective function (Lange et al., 2000; Bolte & Pauwels, 2016). To do so, at the l-th iteration, we design a surrogate function $F_l(\cdot, z_i^{(l)})$ of f_θ anchored at the current point $z_i^{(l)}$. In the traditional MM algorithm, the surrogate function must satisfy the following conditions.

Definition 2.1 (MM Surrogate Conditions)**.** Given $\mu > 0$ and some function $f \in \mathcal{F}_{\text{smooth}}(\mathbb{R}^d)$, we let $\mathcal{M}(f)$ be the class of functions $F \in \mathcal{F}(\mathbb{R}^d \times \mathbb{R}^d)$ which are differentiable with respect to the first variable on $\operatorname{dom} g$ and satisfy, for every $z_i \in \operatorname{dom} g$,

- $F(z_i;z_i) = f(z_i)$ (touching constraint)
- $\nabla_1 F(z_i;z_i) = \nabla f(z_i)$ (tangency constraint)
- $F(z;z_i) \geq f(z_i)$, $\forall z \in \mathbb{R}^d$ (globally majorant)
- $F(\cdot, z_i)$ is μ-strongly convex

We are now ready to define the MM algorithm with initial point x_i.

Definition 2.2 (MM Algorithm)**.** Given $x_i \in \mathbb{R}^d$, $f \in \mathcal{F}_{\text{smooth}}(\mathbb{R}^d)$, $g \in \mathcal{F}(\mathbb{R}^d)$ and some family of majorants $\theta = \{F_l\}_{l\in\mathbb{N}}$, where $F_l \in \mathcal{M}(f)$, the MM algorithm reads $\mathcal{A}_{MM}(x_i,\theta) = \{z^{(l)}\}_{l\in\mathbb{N}}$ where

$$\begin{cases} z^{(0)} = x_i \\ \text{for } l = 0,1,\dots \\ \left\lfloor \; z^{(l+1)} = \underset{z\in\mathbb{R}^d}{\operatorname{argmin}}\, F_l(z;z^{(l)}) + g(z). \right. \end{cases} \tag{3}$$

In the rest of the paper, we specify Problem 2.1 for the class of MM algorithms, i.e.,

$$\Theta = \Big\{\{F_l\}_{l\in\mathbb{N}} \mid \exists f \in \mathcal{F}(\mathbb{R}^d)\,, (\forall l \in \mathbb{N}),\ F_l \in \mathcal{M}(f)\Big\}. \tag{4}$$

Then, we note that if $\theta = \{F_l\}_{l\in\mathbb{N}} \in \Theta$, a function f_θ is uniquely defined through the touching condition in Definition 2.1 (indeed, e.g., $f_\theta(x) = F_0(x,x)$). If the sequence $\{z^{(l)}\}_{l\in\mathbb{N}}$ defined in Definition 2.2 is convergent, then we set $h(x_i,\theta) = \lim_{l\to\infty} z^{(l)}$. We thus propose to approximate $h(x_i,\theta)$ in Problem 2.1 by $h_L(x_i,\theta) = z^{(L)}$ which is the output of the algorithm in (3) truncated after L iterations, hence giving rise to the following approximate bilevel problem.

Problem 2.2 (Approximate Bilevel Problem)**.** Given $L \in \mathbb{N}$ and some training data set $\{x_i,y_i\}_{i=1}^N$ made of N samples, where $\{x_i,y_i\} \in \mathbb{R}^d \times \mathbb{R}^K$ for every $i \in \{1,\dots,N\}$, solve

$$\underset{\substack{\psi\in\mathcal{B}(\mathbb{R}^d,\mathbb{R}^K)\\ \theta\in\Theta}}{\text{minimize}} \sum_{i=1}^N \ell(\psi(h_L(x_i;\theta),y_i)$$
$$\text{with} \quad (\forall i \in [N]), \quad h_L(x_i;\theta) = z_i^{(L)} \tag{5}$$

where $\{z_i^{(l)}\}_{l\in\mathbb{N}} = \mathcal{A}_{MM}(x_i,\theta)$ and we assume that $h_L(x_i,\theta) \to h(x_i,\theta)$ as $L \to \infty$

Remark 2.1. We will show that, for MM algorithms and convex functions g, the sequence $h_L(x_i,\theta)$ converges to a critical point of $f_\theta + g$. Therefore, in this situation, Problem 2.1 can be approximated by Problem 2.2 for sufficiently large L.

3. Connection to Deep Multilayer Networks

In this section, we explore more in depth the setting of MM algorithms defined in (4). In particular, we show that for a specific class Θ of quadratic majorants $\{F_l\}_{l\in\mathbb{N}}$, the Problem 2.2 boils down to the training of feed-forward neural networks. For simplicity we consider that each layer has the same number of neurons.

3.1. Quadratic majorant

We start by introducing the following class of majorants.

Definition 3.1 (Common Quadratic Majorant)**.** Given $f \in \mathcal{F}_{\text{smooth}}(\mathbb{R}^d)$, we let $\mathcal{Q}(f)$ be the family of quadratic majorants $Q_l \in \mathcal{M}(f)$ of the form

$$Q_l(z;z_i) \triangleq \frac{1}{2}z^\top A_l z + \frac{1}{2}z_i^\top B_l z_i + z_i^\top C_l z + d_l^\top z + e_l^\top z_i + \delta_l, \tag{6}$$

where $A_l \in \mathbb{R}^{d\times d}$ is symmetric positive definite, $B_l \in \mathbb{R}^{d\times d}$, $C_l \in \mathbb{R}^{d\times d}$, $d_l \in \mathbb{R}^d$, $e_l \in \mathbb{R}^d$ and $\delta_l \in \mathbb{R}$. Hence, $Q_l(\cdot, z_i)$ is ν_l-smooth with $\nu_l = \|A_l\|_2$.

By elaborating on this collection of quadratic majorants, we define the following proposed majorant F_l as follows.

Proposition 3.1 (Quadratic Majorant)**.** *Let $f \in \mathcal{F}_{\text{smooth}}(\mathbb{R}^d)$ and $Q_l \in \mathcal{Q}(f)$. Then, for all $z_i \in \mathbb{R}^d$,*

$$z \mapsto F_l(z;z_i) \triangleq Q_l(z_i;z_i) + \nabla_1 Q_l(z_i;z_i)^\top (z - z_i) + \frac{1}{2\tau_l}\|z - z_i\|^2, \tag{7}$$

belongs to $\mathcal{M}(f)$ if $\tau_l \leq 1/\nu_l$.

An illustration is provided in Figure 2 (left). We consider the following class of majorants throughout the section.

Definition 3.2 (Class of Quadratic Majorants)**.** We let

$$\Theta_{\mathcal{Q}} = \Big\{\{F_l\}_{l\in\mathbb{N}} \mid \exists f \in \mathcal{F}_{\text{smooth}}(\mathbb{R}^d)\,, (\forall l \in \mathbb{N}),\ F_l \text{ is as in (7)}\Big\}.$$

Hence, it results that each of the N lower-level minimization problems in Problem 2.2 leads to one step of the forward-backward algorithm (Chen & Rockafellar, 1997; Combettes & Wajs, 2005). By hinging on the form of Q_l stated in Definition 3.1, it then yields that

$$\underset{z\in\mathbb{R}^d}{\operatorname{argmin}}\, F_l(z;z_i^{(l)}) + g(z) \tag{8}$$
$$= \operatorname{prox}_{\tau_l g}\left(z_i^{(l)} - \tau_l \nabla_1 Q_l(z_i^{(l)};z_i^{(l)})\right) \tag{9}$$
$$= \operatorname{prox}_{\tau_l g}\left(\underbrace{\left[\mathrm{I_d} - \tau_l(A_l + C_l)^\top\right]}_{\triangleq W_l} z_i^{(l)} \underbrace{-\tau_l d_l}_{\triangleq b_l}\right), \tag{10}$$

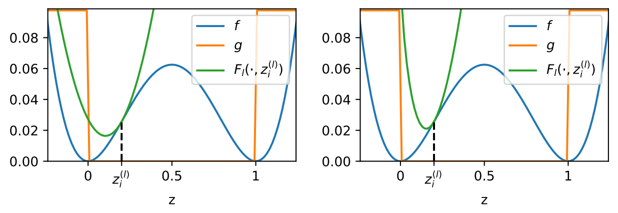

Figure 2. Illustration of the proposed majorant $F_l(\cdot, z_i^{(l)})$ with Euclidean distance (left, see Proposition 3.1) or Bregman distance (right, see Proposition 4.1). The Bregman distance is chosen so that the domain of $F_l(\cdot, z_i^{(l)})$ lies within $\operatorname{dom} g = [0, 1]$.

where we have introduced the variables W_l and b_l. Note that they both incorporate the step-size τ_l while it still appears in the proximity operator. Now, we note that, as A_l and C_l varies in the space of symmetric matrices and the space of matrices, W_l covers the entire space of $\mathbb{R}^{d\times d}$. Similarly, b_l spans the entire space of $\mathbb{R}^d$ as d_l varies. Therefore, Problem 2.2 can be equivalently rewritten as follows.

Problem 3.1 (Training of FFNN)**.** Given some training data set $\{x_i, y_i\}_{i=1}^N$ made of N samples, where $\{x_i, y_i\} \in \mathbb{R}^d \times \mathbb{R}^K$ for every $i \in \{1, \dots, N\}$, solve

$$\underset{\substack{\psi\in\mathcal{B}(\mathbb{R}^d,\mathbb{R}^K)\\ \{W_l,b_l\}_{l=0}^{L-1}\in(\mathbb{R}^{d\times d}\times\mathbb{R}^d)^L}}{\text{minimize}} \quad \sum_{i=1}^N \ell(\psi(z_i^{(L)}), y_i)$$
$$\text{s.t. } (\forall i \in [N]), \begin{cases} z_i^{(0)} = x_i \\ \text{for } l = 0, 1, \dots, L-1 \\ \lfloor \; z_i^{(l+1)} = \operatorname{prox}_{\tau_l g}(W_l z_i^{(l)} + b_l) \end{cases} . \tag{11}$$

3.2. Connection with neural networks

Problem 3.1 yields similarities with the training of feed-forward neural networks made of L layers. Here $\{W_l\}_{l=0}^{L-1}$ and $\{b_l\}_{l=0}^{L-1}$ represent the weights and biases of neural networks, respectively. We highlight that the case where W_l is symmetric (Hu et al., 2019) corresponds to the scenario where C_l, appearing in (10), is symmetric as well. In addition, as shown in (Combettes & Pesquet, 2020), the proximity operator $\operatorname{prox}_{\tau_l g}$ can match a variety of activation functions depending on the choice of the function g by setting the step-size $\tau_l = 1$. Below, we recall two of such examples.

Example 3.1 (ISRU)**.** The inverse square root unit function $\rho\colon t \in \mathbb{R} \mapsto t/\sqrt{1+t^2} \in \mathbb{R}$ can be expressed as the proximity operator prox_g of the function $g\colon \mathbb{R} \to]-\infty, +\infty]$

$$g\colon t \mapsto \begin{cases} -t^2/2 - \sqrt{1-t^2}, & \text{if } |t| \le 1; \\ +\infty, & \text{otherwise.} \end{cases}$$

Example 3.2 (Arctan)**.** The arctangent activation function $\rho\colon t \in \mathbb{R} \mapsto (2/\pi)\arctan(t)$ is the proximity operator of

$$g\colon t \in \mathbb{R} \mapsto \begin{cases} -\frac{2}{\pi}\log(\cos\frac{\pi t}{2}) - t^2/2, & \text{if} |t| < 1; \\ +\infty, & \text{otherwise.} \end{cases}$$

These examples of activation functions ρ will be revisited in the next section for convex functions g.

4. Bregman Feed-Forward Neural Network

While in the previous section we have considered the class $\Theta_{\mathcal{Q}}$ of quadratic majorants $\{F_l\}_{l\in\mathbb{N}}$, we now investigate a more general class involving the use of Bregman distances. In addition, we show that, for such class and for some regularizer g, the lower-level of Problem 2.2 yields a new type of neural network. An extension to varying number of neurons per layer is provided in the supplementary material.

4.1. Majorant with Bregman distances

We start by recalling the definition of Bregman distances (Bauschke et al., 2018).

Definition 4.1 (Bregman distance)**.** Given some Legendre function $\Phi \in \Gamma_0(\mathbb{R}^d)$, the Bregman distance associated to Φ reads

$$(\forall u \in \operatorname{dom}\Phi, v \in \operatorname{int}\operatorname{dom}\Phi)$$
$$D_\Phi(u, v) = \Phi(u) - \Phi(v) - \langle \nabla\Phi(v), u - v\rangle. \tag{12}$$

In particular, we recover the Euclidean distance for $\Phi = \frac{1}{2}\|\cdot\|^2$, i.e., $D_{\frac{1}{2}\|\cdot\|^2}(u, v) = \frac{1}{2}\|u - v\|^2$.

Equipped with this definition, we generalize the majorants of Proposition 3.1 by considering Bregman distances in place of the Euclidean distance.

Proposition 4.1 (Majorant with Bregman distances)**.** *Let $f \in \mathcal{F}_{\text{smooth}}(\mathbb{R}^d)$, $Q_l \in \mathcal{Q}(f)$ and Φ be a strongly convex Legendre function. Then, for every $z_i \in \mathbb{R}^d$,*

$$z \in \operatorname{dom}\Phi \mapsto F_l(z; z_i) \triangleq Q_l(z_i; z_i) + \nabla_1 Q_l(z_i; z_i^\top(z - z_i) + \tau_l^{-1} D_\Phi(z, z_i), \tag{13}$$

is a majorant of f on $\operatorname{dom}\Phi$ for every $\tau_l < 1/\nu_l$.

The proof is given in the supplementary material. An illustration of the majorant is reported in Figure 2 (right). We now define the corresponding class of Bregman majorants.

Definition 4.2 (Class of Bregman majorants)**.** We let

$$\Theta_{\mathcal{B}} = \Big\{ \{F_l\}_{l\in\mathbb{N}} \mid \exists f \in \mathcal{F}_{\text{smooth}}(\mathbb{R}^d)\,, \\ (\forall l \in \mathbb{N}),\ F_l \text{ is as in (13)} \Big\}.$$

It follows that, for this class, each of the N lower-level minimization problems in Problem 2.2 reads

$$\begin{aligned} &\operatorname*{argmin}_{z\in\mathbb{R}^d} F_l(z; z_i^{(l)}) + g(z) && (14)\\ &= \operatorname{prox}^{\Phi}_{\tau_l g}\Big(\nabla\Phi(z_i^{(l)}) - \tau_l \nabla_1 Q_l(z_i^{(l)}; z_i^{(l)})\Big) && (15)\end{aligned}$$

which leads to one step of the forward-backward algorithm with Bregman distances (Van Nguyen, 2017; Bolte et al., 2018). We recall that the Bregman proximity operator (in Van Nguyen sense) of $\tau_l g$ with respect to Φ reads

$$\operatorname{prox}^{\Phi}_{\tau_l g}(v) = \operatorname*{argmin}_{u\in\mathbb{R}^d} \tau_l g(u) + \Phi(u) - \langle u, v\rangle. \qquad (16)$$

Finally, by specifying the form of Q_l, it then yields that

$$\begin{aligned}&\operatorname*{argmin}_{z\in\mathbb{R}^d} F_l(z; z_i^{(l)}) + g(z)\\ &= \operatorname{prox}^{\Phi}_{\tau_l g}\left(\nabla\Phi(z_i^{(l)}) \underbrace{-\tau_l(A_l + C_l)^\top}_{\triangleq W_l} z_i^{(l)} \underbrace{-\tau_l d_l}_{\triangleq b_l}\right). \qquad (17)\end{aligned}$$

Once again, by using the same arguments as in Sec. 3, Problem 2.2 boils down to:

Problem 4.1 (Training of Bregman FFNN)**.** Given some training data set $\{x_i, y_i\}_{i=1}^N$ made of N samples, where $\{x_i, y_i\} \in \mathbb{R}^d \times \mathbb{R}^K$ for every $i \in \{1, \ldots, N\}$, and a strongly convex Legendre function $\Phi\colon \mathbb{R}^d \to \mathbb{R}$, solve

$$\operatorname*{minimize}_{\substack{\psi\in\mathcal{B}(\mathbb{R}^d,\mathbb{R}^K)\\ \{W_l, b_l\}_{l=0}^{L-1}\in(\mathbb{R}^{d\times d}\times\mathbb{R}^d)^L}} \sum_{i=1}^N \ell(\psi(z_i^{(L)}), y_i)$$

$$\text{s.t. } (\forall i \in [N]),\ \begin{cases} z_i^{(0)} = x_i \\ \text{for } l = 0, 1, \ldots, L-1 \\ \big\lfloor\ z_i^{(l+1)} = \operatorname{prox}^{\Phi}_{\tau_l g}(\nabla\Phi(z_i^{(l)}) + W_l z_i^{(l)} + b_l) \end{cases}$$

4.2. Proposed architecture

Similarly to the comparisons drawn in Sec. 3.2, we argue that the lower-level algorithm in Problem 4.1 bears some similarities to feed-forward neural networks made of L layers. Indeed, we also identify $\{W_l\}_{l=1}^L$ and $\{b_l\}_{l=1}^L$ to the weights and biases, respectively. In the remaining of the

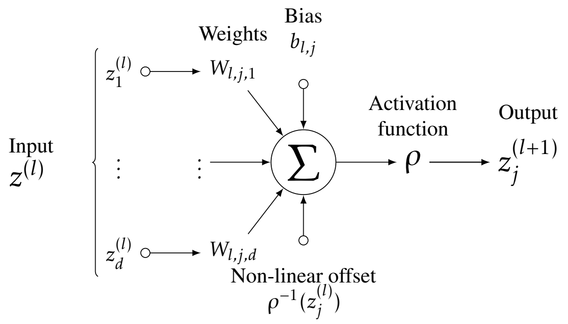

Figure 3. Bregman feed-forward neuron. Illustration of the j-th neuron at the l-th layer. The major difference lies in the presence of an additional non-linear function ρ^{-1}. With such scheme, if the weights and the bias are zero then $z_j^{(l+1)} = z_j^{(l)}$.

section, we will show that, when $\operatorname{prox}^{\Phi}_{\tau g}$ is an activation operator, then $\nabla\Phi$ is the inverse activation operator.

We start by showing that a large variety of activation operators can be put in the form of a proximity operator with Bregman distances. To do so, we recall that, at a given layer, the activation operator is commonly written as an activation function separately applied to each neuron. In our context, this amounts in choosing a separable function $g\colon z = [z_1 \cdots z_d] \in \mathbb{R}^d \mapsto \sum_{j=1}^d \tilde{g}(z_j)$ and a separable Legendre function Φ as well, i.e., $\Phi\colon z \in \mathbb{R}^d \mapsto \sum_{j=1}^d \phi(z_j)$ where ϕ is a Legendre function on $\mathbb{R}$. Thus, it follows that $\operatorname{prox}^{\Phi}_{\tau g}$ is also separable, i.e., for every $z \in \mathbb{R}^d$,

$$\operatorname{prox}^{\Phi}_{\tau g}(z) = [\operatorname{prox}^{\phi}_{\tau\tilde{g}}(z_1) \cdots \operatorname{prox}^{\phi}_{\tau\tilde{g}}(z_d)]. \qquad (18)$$

We now show that many activation functions ρ can be written as Bregman projections.

Proposition 4.2 (Informal connection)**.** *Many activations functions ρ can be written as Bregman proximity operators of some indicator functions $\tilde{g} = \iota_{\mathcal{I}}$, where $\mathcal{I} \subseteq \mathbb{R}$, for a Legendre function ϕ such that* $\operatorname{dom}\phi = \mathcal{I}$*, i.e,*

$$\rho \triangleq \operatorname{prox}^{\phi}_{\tau\iota_{\mathcal{I}}} = \operatorname{prox}^{\phi}_{\iota_{\mathcal{I}}} \quad \textit{where} \quad \operatorname{dom}\phi = \mathcal{I} \qquad (19)$$

We report in Table 1 how the most common activation functions can be recovered for various choice of $(\mathcal{I}, \phi)$.

As a consequence, a large variety of activations can be equivalently written in the form of a proximity with respect to the Euclidean metric (see (Combettes & Pesquet, 2020)) or in the form of a Bregman projection with respect to an appropriate Legendre function. However, in the first case the step-size appears in the proximity (and should be set to 1) while, in the proposed framework, it does not impact the proximity anymore since g is an indicator function. Hence, the step-size is solely captured in the weights and biases (see (17)).

$\mathcal{I}$	Legendre function ϕ	Activation function $\rho \triangleq \operatorname{prox}^{\phi}_{\iota_{\mathcal{I}}} = \nabla\phi^{-1}$	$\rho^{-1} \triangleq \nabla\phi$
$[-1,1]$	$t \mapsto -\sqrt{1-t^2}$	ISRU: $t \mapsto \frac{\lambda t}{\sqrt{1+t^2}}$	$t \mapsto \frac{t}{\sqrt{1-t^2}}$
$[0,1]$	$t \mapsto t\log t + (1-t)\log(1-t)$	Sigmoid: $t \mapsto \frac{1}{1+e^{-t}}$	$t \mapsto \log(\frac{t}{1-t})$
$\mathbb{R}_{>0}$	$t \mapsto t\log(\frac{e^t-1}{1-e^t}) - \mathrm{Li}_2(e^t)$	SoftPlus $t \mapsto \log(e^t+1)$	$t \mapsto \log(e^t-1)$
$[-\pi/2, \pi/2]$	$t \mapsto -\log(\cos(t))$	$t \mapsto \arctan(t)$	$t \mapsto \tan(t)$
$[-1,1]$	$t \mapsto \frac{1}{2}\log(1-t^2) + t\,\mathrm{arctanh}(t)$	$t \mapsto \tanh(t)$	$t \mapsto \mathrm{arctanh}(t)$
$[-1,1]$	$t \mapsto \sqrt{1-t^2} + t\arcsin(t)$	$t \mapsto \sin(t)$	$t \mapsto \arcsin(t)$
$\mathbb{R}$	$t \mapsto \cosh(t)$	$t \mapsto \mathrm{arcsinh}(t)$	$t \mapsto \sinh(t)$

Table 1. Connection between activation functions ρ and Bregman proximity operators $\operatorname{prox}^{\phi}_{\iota_{\mathcal{I}}}$ for specific choice of sets $\mathcal{I}$ and Legendre functions ϕ. Here $\mathrm{Li}_n(x) = \sum_{k=1}^{\infty} \frac{x^k}{k^n}$ is the polylogarithm function.

We now turn to the term $\nabla\Phi(z^{(l)}) = [\nabla\phi(z_1^{(l)}) \cdots \nabla\phi(z_d^{(l)})]$ appearing in Problem 4.1 which was absent in Problem 3.1.

Proposition 4.3. *If an activation function ρ can be written in the form of Proposition 4.2, then $\rho^{-1} = \nabla\phi$.*

Proof. If $\operatorname{dom}\phi = \mathcal{I}$ then $\operatorname{prox}^{\phi}_{\iota_{\mathcal{I}}} = \nabla\phi^{-1}$ by applying the definition of the Bregman proximity operator given in (16). □

Thus, in our framework, $\nabla\Phi$ is the inverse activation operator and the lower-level algorithm in Problem 4.1 can be rewritten in terms of Bregman feed-forward layers.

Definition 4.3 (Bregman feed-forward layer)**.** Let ρ be some activation function written as a Bregman proximity operator (see Proposition 4.2). Then, the Bregman feed-forward layer reads:

$$z^{(l+1)} = \rho(\rho^{-1}(z^{(l)}) + W_l z^{(l)} + b_l). \qquad (20)$$

A sketch of the proposed neuron is displayed in Figure 3.

Intuitively, the added term $\rho^{-1}(z^{(l)})$ plays a similar role as the shortcut term present in ResNet (He et al., 2016) since it will connect one layer to all previous ones. Another particularity is that, whenever $W_l = 0$ and $b_l = 0$, then

$$z^{(l+1)} = \rho(\rho^{-1}(z^{(l)}) + W_l z^{(l)} + b_l) = \rho(\rho^{-1}(z^{(l)})) = z^{(l)}. \quad (21)$$

In other words, the neuron is the identity. Note that, when designing an activation, a special attention is usually given to the property that the activation function should approximates the identity near the origin in order to reproduce (21).

5. Theoretical Analysis

In the previous section, we saw that the lower-level algorithms in Problem 3.1 and Problem 4.1 can be interpreted as appropriate MM algorithms. In this section we study the simplified setting in which the majorant functions does not change during the algorithm, that is $F_l = F$. This cover the setting of deep equilibrium problem as mentioned in the introduction. In this case we have a unique couple (W, b) that drives all the layers as defined in Proposition 3.1 and Proposition 4.1. Then, the question of the convergence of those algorithms is fundamental in order to connect the above problems to Problem 2.1 (see Remark 2.1).

In this respect, we first note that state-of-the-art results concerning MM algorithms, in the setting of nonconvex and nonsmooth functions, assume that the involved majorizing functions are strongly convex (Bolte & Pauwels, 2016). Second, we saw that, in both problems, the proximity operator of g can be interpreted as an activation function. For instance, the arctan activation can be equivalently given in the form of a proximity with respect to the Euclidean metric (see Example 3.2) or in the form of a Bregman projection with respect to an appropriate Legendre function (see Table 1). However, in the first case, since g is nonconvex, the overall majorizing function $F + g$ is nonconvex as well, while, in the second case, it is strongly convex, since g is actually the indicator function of a convex set. Therefore, we see that the theory of convergence of MM algorithms developed in (Bolte & Pauwels, 2016) can be fully applied (only) to the Bregman Feed-Forward networks, which ultimately gives the following result

Theorem 5.1 (Convergence of the MM sequence)**.** *Let $g = \iota_{\mathcal{C}}$, for some convex set $\mathcal{C} \subseteq \mathbb{R}^d$, and the family of majorants F of f, given in Proposition 4.1 where Φ satisfy $\operatorname{dom}\Phi = \mathcal{C}$. Then, the sequence $\{z^{(l)}\}_{l\in\mathbb{N}}$ defined in (3) converges to a critical point of $f + g$.*

Ideally one would like to prove that Problem 2.2 converges to the exact Problem 2.1 when L tends to infinity, at least in terms of infima. This would provide a full justification for the interpretation of the training of feed-forward deep neural network as a bilevel problem as the number of layers goes to infinity. As noted in other works (see Frecon et al., 2018; Franceschi et al., 2018) this would require proving the uniform convergence of the MM algorithm on Θ. Unfortunately we were not able to prove this statement and we leave this as a future work. Also extending the results in (Bolte & Pauwels, 2016) to handle majorizing functions that may depend on l is another important open issue.

6. Numerical Experiments

The purpose of the numerical experiments is to compare the standard feed-forward architecture against two variants coming from the observations that the activation operator can be expressed as both a Euclidean proximity operator (Combettes & Pesquet, 2020) and a Bregman projection.

6.1. Architectures Compared

Given an activation operator ρ, we consider the standard layer update

- Standard FFNN: $z^{(l+1)} = \rho(W_l z^{(l)} + b_l)$,

and two variants resulting from the point of view of the proposed bilevel Problem 4.1

- Euclidean FFNN: $z^{(l+1)} = \rho(z^{(l)} + W_l z^{(l)} + b_l)$,
- Bregman FFNN: $z^{(l+1)} = \rho(\rho^{-1}(z^{(l)}) + W_l z^{(l)} + b_l)$.

All networks are optimized using a mini-batch gradient descent algorithm[2].

6.2. Performance on Two-spiral Dataset

In this section, we compare all three methods on a 2 dimensional yet challenging binary classification task.

Two-spiral dataset. The two-spiral dataset is a widely used benchmark for binary classification (Chalup & Wiklendt, 2007). The data consist of points on two intertwined spirals which cannot be linearly separated. An illustration is reported in Figure 4 (top left).

Goal and setting. The purpose of the experiment is to train each architecture so that the learned representation of the two-spiral dataset can be linearly separated. To do so, we consider 3 layers with arctan activations and 2 neurons per layer, followed by a linear layer.

Comparison of Learning Behavior. In order to compare the learning behavior of all three architectures, we report in Figure 5 (left) the training losses (averaged over 10 realizations) as a functions of the number of epochs. For all three methods the batch-size is set to 10, the learning rate to 0.1 and the number of epochs to 1000. First of all, we observe that the training loss corresponding to the Euclidean FFNN decreases more rapidly than that of the Standard FFNN. This is interesting because it means that a simple reparametrization of the weights W_l by $I_d + W_l$ improves the learning

[2] A Pytorch package will be made publicly available at the time of publication.

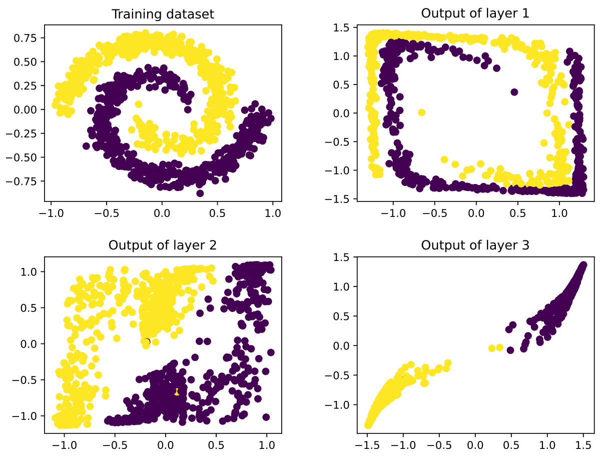

Figure 4. Illustration of two-spiral dataset (top left) and the outputs of each layer of the proposed Bregman FFNN

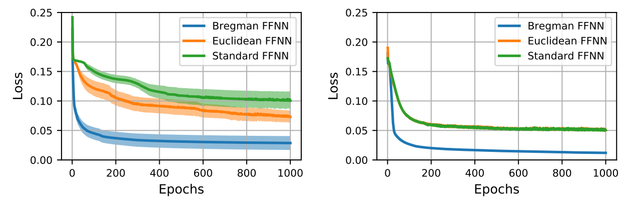

Figure 5. Comparison of training losses as functions of epochs. The mean is reported in bold while the standard deviation is illustrated by the surrounding envelope. Left: initialization with random weights. Right: initialization with deterministic weights.

of the model. In addition, we see that the training loss can be decreased even more by choosing the Bregman FFNN architecture. For illustration purposes, we report in Figure 4 the outputs at each layer of the learned Bregman FFNN. We observe that the output at the third layer can indeed be linearly separated.

However, it should be noted that the way the weights are initialized drastically impacts the learning behavior. In practice the weights and biases are initialized randomly. In order to evaluate the impact of the initial weights, we have reproduced the experiment with deterministic and non-informative weights. For Standard FFNN and Euclidean FFNN we set $W_l = I_d$ and $W_l = 0_d$, respectively, while for Bregman FFNN we set $W_l = 0_d$. For all methods we set the initial biases to 0. The corresponding results are displayed in Figure 5 (right). Once again we see an improvement of Bregman FFNN over Standard FFNN. However, we observe that in this setting both Standard FFNN and Euclidean FFNN behaves equally. For all methods, we see that such initialization, in this specific example, yields a lower training loss.

Impact on training accuracy. In order to further investigate how the learning behavior impact the final training

	Standard	Euclidean	Bregman
MNIST (FFNN 1 hidden layer)	95.27%	93.98%	**97.40%**
MNIST (FFNN 3 hidden layers)	97.89%	98.32%	**98.41%**
MNIST (CNN + FFNN 1 hidden layer)	99.39%	99.35%	**99.40%**
FashionMNIST (FFNN 1 hidden layer)	89.11%	89.01%	**89.39%**
FashionMNIST (FFNN 3 hidden layers)	88.61%	89.38%	**89.95%**
FashionMNIST (CNN + FFNN 1 hidden layer)	92.47%	92.55%	**93.19%**
Cifar10 (CNN + FFNN 1 hidden layer)	78.02%	78.00%	**79.95%**

Table 2. Mean test accuracy over 5 realizations on benchmark datasets.

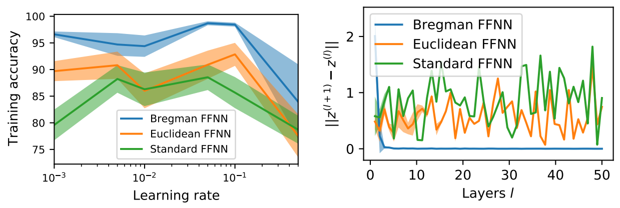

Figure 6. Results on two-spiral dataset. Left: Impact of the learning rate on the training accuracy. Right: Norm difference between the output of two consecutive layers.

accuracy, we perform the following experiment. For each of the three architectures, we investigate various learning rates while the batch-size is set to 10 and number of epochs to 10^4 in order to reach convergence at a reasonable running time. Results are reported in Figure 6 (left plot) and show that the Bregman variant yields better training accuracy with a higher learning rate than its standard counterparts.

Deep architecture comparison. We now consider the case that the number of layers is large by setting $L = 50$. Once the model is learned, we report in Figure 6 the mean norm difference between the output of two consecutive layers as a function of the layer. Contrary to Euclidean and Standard, we observe that for Bregman the norm differences goes to 0, supporting the result of Theorem 5.1.

6.3. Performance on Real-World Datasets

In this section, we conduct two type of experiments on real world images. The purpose of these experiments is not to improve on the accuracy of state-of-the-art neural networks but to assess, on simple architectures, the added benefit.

Datasets. We consider the MNIST, FashionMNIST and Cifar10 images datasets. Each image is pre-processed such that the pixels intensity lie within $]0, 1[$. For each dataset, the goal is to assign each image to one of the $K = 10$ labels.

First experiment. In this first experiment, we flatten each $p \times p$ pixels images into $d = p^2$ dimensional vectors. Even though this non-spatial representation is not appropriate for vision tasks, we embrace it in order to compare solely each of the three architectures stated in Sec. 6.1. Here, we restrict to the simplest architectures made of L hidden layers with d neurons each, followed by a linear layer combined with a softmax operator in order to map to one the K labels.

Second experiment. We now embed each of the three models in the final classification layer of convolutional neural networks whose architectures are detailed in the supplementary material.

The mean test accuracies are reported in Table 2. Firstly, we notice that Euclidean improves over Standard when the number of layers $L > 2$, otherwise performance are comparable. Interestingly, this suggests that solely reparametrizing the weights can help in achieving better prediction performance. Secondly, we observe that in all scenarios, the Bregman variant yields a higher test accuracy or at least the same accuracy (see the 3rd setting).

7. Discussion

The present paper framed the learning of a representation mapping as a bilevel optimization which we addressed by a majorization-minimization algorithm. We have shown that for some quadratic majorant, the bilevel problem boils down to the training of a feed-forward neural network. In addition, by elaborating on more general majorants, we proposed the Bregman feed-forward layer which includes an additional term defined as the inverse of the activation function. Intuitively, this term plays a similar role as the skip connections introduced in the ResNet (He et al., 2016) by linking one layer to the previous ones.

We advocate that replacing the standard feed-forward layers in state-of-the-art architectures by the proposed Bregman layers could additionally improve prediction. By doing so, one can make sure that the output of a layer is equal to its input whenever the weights and biases are zero. We believe, this may reduce the memory needed in very deep architectures. In addition, they are two interesting aspects encountered in practice when optimizing deep Bregman feed-forward networks. First, we are able to learn the model for significantly larger learning rates than its standard counterpart. Second, from a sufficiently deep layer, all weights and biases are zero until the last layer.

Future works should further study both aspects in very deep architectures. For instance, one could investigate if the presence of the inverse activation mitigates both vanishing and exploding gradients issues commonly encountered when optimization standard neural networks. In addition, proving that the infima of Problem 2.2 converge to Problem 2.1 as the number of layers goes to infinity is an important problem left for future work.

References

Attouch, H., Bolte, J., Redont, P., and Soubeyran, A. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the kurdyka-Łojasiewicz inequality. *Mathematics of Operations Research*, 35(2):438–457, 2010. ISSN 0364765X, 15265471. URL http://www.jstor.org/stable/40801236.

Bai, S., Kolter, J. Z., and Koltun, V. Deep equilibrium models. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32, pp. 690–701. Curran Associates, Inc., 2019.

Bauschke, H. H., Dao, M. N., and Lindstrom, S. B. Regularizing with bregman–moreau envelopes. *SIAM Journal on Optimization*, 28(4):3208–3228, jan 2018. doi: 10.1137/17m1130745.

Bertocchi, C., Chouzenoux, E., Corbineau, M.-C., Pesquet, J.-C., and Prato, M. Deep unfolding of a proximal interior point method for image restoration. *Inverse Problems*, 36 (3):034005, feb 2020. doi: 10.1088/1361-6420/ab460a. URL https://doi.org/10.1088/1361-6420/ab460a.

Bibi, A., Ghanem, B., Koltun, V., and Ranftl, R. Deep layers as stochastic solvers. In *7th International Conference on Learning Representations*, New Orleans, LA, USA,, May 2019.

Blin, R., Ainouz, S., Canu, S., and Meriaudeau, F. Road scenes analysis in adverse weather conditions by polarization-encoded images and adapted deep learning. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 27–32. IEEE, 2019.

Bolte, J. and Pauwels, E. Majorization-minimization procedures and convergence of SQP methods for semi-algebraic and tame programs. *Mathematics of Operations Research*, 41(2):442–465, may 2016. doi: 10.1287/moor.2015.0735.

Bolte, J., Sabach, S., Teboulle, M., and Vaisbourd, Y. First order methods beyond convexity and lipschitz gradient continuity with applications to quadratic inverse problems. *SIAM Journal on Optimization*, 28(3):2131–2151, jan 2018. doi: 10.1137/17m1138558.

Chalup, S. and Wiklendt, L. Variations of the two-spiral task. *Connect. Sci.*, 19:183–199, 06 2007. doi: 10.1080/09540090701398017.

Chen, G. H.-G. and Rockafellar, R. T. Convergence rates in forward–backward splitting. *SIAM J. on Optimization*, 7(2):421–444, February 1997. ISSN 1052-6234. doi: 10.1137/S1052623495290179. URL https://doi.org/10.1137/S1052623495290179.

Combettes, P. L. and Pesquet, J.-C. Deep neural network structures solving variational inequalities. *Set-Valued and Variational Analysis*, feb 2020. doi: 10.1007/s11228-019-00526-z.

Combettes, P. L. and Wajs, V. R. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., and Pontil, M. Bilevel programming for hyperparameter optimization and meta-learning. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1568–1577, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Frecon, J., Salzo, S., and Pontil, M. Bilevel learning of the group lasso structure. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31, pp. 8301–8311. Curran Associates, Inc., 2018.

Grazzi, R., Franceschi, L., Pontil, M., and Salzo, S. On the iteration complexity of hypergradient computation. In *International Conference on Machine Learning*, pp. 3748–3758. PMLR, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hu, S. X., Zagoruyko, S., and Komodakis, N. Exploring weight symmetry in deep neural networks. *Computer Vision and Image Understanding*, 187:102786, oct 2019. doi: 10.1016/j.cviu.2019.07.006.

Lange, K., Hunter, D. R., and Yang, I. Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical Statistics*, 9(1):1, mar 2000. doi: 10.2307/1390605.

Lassau, N., Ammari, S., Chouzenoux, E., Gortais, H., Herent, P., Devilder, M., Soliman, S., Meyrignac, O., Talabard, M.-P., Lamarque, J.-P., Dubois, R., Loiseau, N., Trichelair, P., Bendjebbar, E., Garcia, G., Balleyguier, C., Merad, M., Stoclin, A., Jegou, S., Griscelli, F., Tetelboum, N., Li, Y., Verma, S., Terris, M., Dardouri, T., Gupta, K., Neacsu, A., Chemouni, F., Sefta, M., Jehanno, P., Bousaid, I., Boursin, Y., Planchet, E., Azoulay, M., Dachary, J., Brulport, F., Gonzalez, A., Dehaene, O., Schiratti, J.-B., Schutte, K., Pesquet, J.-C., Talbot, H., Pronier, E., Wainrib, G., Clozel, T., Barlesi, F., Bellin, M.-F., and Blum, M. G. B. Integrating deep learning CT-scan model, biological and clinical variables to predict severity of COVID-19 patients. *Nature Communications*, 12(1), jan 2021. doi: 10.1038/s41467-020-20657-4.

Maggu, J., Majumdar, A., Chouzenoux, E., and Chierchia, G. Deep convolutional transform learning. In *Communications in Computer and Information Science*, pp. 300–307. Springer International Publishing, 2020. doi: 10.1007/978-3-030-63823-8_35.

Miller, J. and Hardt, M. Stable recurrent models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Monga, V., Li, Y., and Eldar, Y. C. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing, 2020.

Van Nguyen, Q. Forward-backward splitting with bregman distances. *Vietnam Journal of Mathematics*, 45(3):519–539, 2017.

Zhou, T., Ruan, S., and Canu, S. A review: Deep learning for medical image segmentation using multi-modality fusion. *Array*, 3:100004, 2019.