Bregman Neural Networks

Jordan Frecon [1] **Gilles Gasso** [1] **Massimiliano Pontil** [2 3] **Saverio Salzo** [2]

Abstract

We present a framework based on bilevel optimization for learning multilayer, deep data representations. On the one hand, the lower-level problem finds a representation by successively minimizing layer-wise objectives made of the sum of a prescribed regularizer, a fidelity term and a linear function depending on the representation found at the previous layer. On the other hand, the upper-level problem optimizes over the linear functions to yield a linearly separable final representation. We show that, by choosing the fidelity term as the quadratic distance between two successive layer-wise representations, the bilevel problem reduces to the training of a feedforward neural network. Instead, by elaborating on Bregman distances, we devise a novel neural network architecture additionally involving the inverse of the activation function reminiscent of the skip connection used in ResNets. Numerical experiments suggest that the proposed Bregman variant benefits from better learning properties and more robust prediction performance.

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; Grigorescu et al., 2020), among others. Consequently, they have received a great interest from the machine learning community (see, e.g.. (Maggu et al., 2020) and references therein). A key aspect of deep neural networks is their ability to learn a representation from raw inputs by alternating linear transformations and non-linear operations. In this paper, rather than directly writing the representation mapping as a prescribed compositional form, we present a framework to learn the functional form of the representation from the perspective of bilevel optimization (see Franceschi et al., 2018; Liu et al., 2019; 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 define the representation as the successive minimization of multiple learnable objective functions. The latter are formed by summing a bilinear parametric function and prescribed convex functions. We show how this iterative scheme naturally includes standard feed-forward neural networks and gives rise to novel multilayer networks.

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. In addition, the authors have provided a thorough analysis with tools from monotone operator theory in order to study a class of neural networks and their asymptotic properties. Feedforward networks have been studied by Bibi et al. (2019) who have also established that the forward pass through one feedforward 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).

Contributions and Organization. In Section 2, we present the proposed multilayered bilevel framework and in particular the class of layerwise objectives addressed throughout the paper. The latter are formed by summing a linear term, a convex regularizer and a divergence measuring the closeness with the representation found at the previous layer. From this, we show in Section 3 that, for an appropriate regularizer and a divergence chosen as a quadratic distance, the representation found at each layer reduces to a feedforward update. Next, by elaborating on Bregman distances, in Section 4 we propose the *Bregman feedforward layer*, a new type of neural network 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 projections as opposed to

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

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

proximity operators. The practical benefits of this new type of layers are assessed on both synthetic and real datasets in Section 5 by extending multilayer perceptrons and residual networks.

Notations. Let $\mathcal{X}$ and $\mathcal{Y}$ be two Hilbert spaces. For every lower semicontinuous extended real-valued function $f : \mathcal{X} \to \mathbb{R} \cup \{\pm\infty\}$, we denote by ∂f the subdifferential of f and by $\nabla_j f$ the gradient of f with respect to the j-th variable. In addition, we let $\Gamma_0(\mathcal{X})$ be the space of functions from $\mathcal{X}$ to $]-\infty,+\infty]$ which are closed proper and convex, and $\mathcal{B}(\mathcal{X},\mathcal{Y})$ be the space of bounded operators from $\mathcal{X}$ to $\mathcal{Y}$. Also let $\mathcal{D}(\mathcal{X})$ be the space of functions $f : \mathcal{X} \to]-\infty,+\infty]$ 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. For $n \in \mathbb{N}_+^*$, we let $[n] = \{1,\dots,n\}$. For every set $\mathcal{C}$ we denote by $\operatorname{int} \mathcal{C}$ the interior of $\mathcal{C}$.

2. Multilayered 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 $y_i \in \mathbb{R}^c$. In other words, this means that the predicted target $\hat{y}_i$ can be written as a simple transformation $\psi \in \mathcal{B}(\mathbb{R}^d, \mathbb{R}^c)$ 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 loss function ℓ such as the quadratic loss (for regression) or the cross-entropy (for classification) to name a few.

Similarly to neural networks, we define the learned representation $h(x_i)$ through a sequence $\{z_i^{(l)}\}_{l=0}^{L-1}$ of $L \in \mathbb{N}^+$ intermediate representations of x_i. A key feature of our framework is to implicitly learn $z_i^{(l)}$ by minimizing an objective function depending on $z_i^{(l-1)}$. More formally, we consider that the representation is issued from the following learning scheme.

$$\begin{aligned} &z_i^{(0)} = x_i \\ &\text{for } l = 0, 1, \dots L-1 \\ &\left\lfloor \; z_i^{(l+1)} = \operatorname*{argmin}_{z \in \mathbb{R}^d} \mathcal{L}_l(z, z_i^{(l)}) \right. \\ &h(x_i) = z_i^{(L)}. \end{aligned} \tag{1}$$

This gives rise to a discrete dynamical system where $\mathcal{L}_l$ describes the layer-dependence of the representations. Here, we assume that each of the objective can be split into three components, i.e.,

$$\mathcal{L}_l(z, z_i^{(l)}) \triangleq f_l(z, z_i^{(l)}) + D(z, z_i^{(l)}) + g(z), \tag{2}$$

playing distinct roles. Firstly, $f_l \in \mathcal{F}(\mathbb{R}^d \times \mathbb{R}^d)$, where $\mathcal{F}(\mathbb{R}^d \times \mathbb{R}^d)$ models a class of functions to be specified in Definition 2.1, which encourages some peculiar solution $z_i^{(l+1)}$ by judging upon the representation at the previous layer $z_i^{(l)}$. Secondly, $D(\cdot, z_i^{(l)}) \in \mathcal{D}(\mathbb{R}^d)$ is a divergence measuring the closeness with $z_i^{(l)}$. As such, it is acts as a data-fidelity term ensuring some relationship between two successive layer-wise representations. Lastly, $g \in \Gamma_0(\mathbb{R}^d)$ is a simple[1] function acting as a regularizer. For instance, g can promotes sparse solutions or constrain the solution to live in some manifold.

We consider that both D and g are prescribed while we treat $\{f_l\}_{l=0}^{L-1}$ as hyper-parameters whose choice permits to tune the representation mapping h. In order to learn both ψ and $\{f_l\}_{l=0}^{L-1}$, we consider the following bilevel problem.

Problem 2.1 (Multilayered Bilevel Problem)**.** Given a training data set $\{x_i, y_i\}_{i=1}^n$ where $\{x_i, y_i\} \in \mathbb{R}^d \times \mathbb{R}^c$ for every $i \in [n]$ and a function $g \in \Gamma_0(\mathbb{R}^d)$, solve

$$\begin{aligned} &\operatorname*{minimize}_{\substack{\psi \in \mathcal{B}(\mathbb{R}^d, \mathbb{R}^c) \\ \{f_l\}_{l=0}^{L-1} \in \mathcal{F}(\mathbb{R}^d \times \mathbb{R}^d)^L}} \sum_{i=1}^n \ell(\psi(z_i^{(L)}), y_i) \quad \text{where} \quad \forall i \in [n], \\ &\quad z_i^{(0)} = x_i \\ &\quad \text{for } l = 0, 1, \dots L-1 \\ &\quad \left\lfloor \; z_i^{(l+1)} = \operatorname*{argmin}_{z \in \mathbb{R}^d} f_l(z, z_i^{(l)}) + D(z, z_i^{(l)}) + g(z). \right. \end{aligned} \tag{3}$$

Problem 2.1 represents one of the main proposal of the paper. For different choices of $\{f_l, D, g\}$, we will later show that it is equivalent to training different forms of neural networks. More specifically, $\{f_l\}_{l=0}^{L-1}$ will relate to the linear layers whereas $\{D, g\}$ will be linked to the activation functions.

Note that the choice of the divergence D directly affects the geometry of the representation learning problem. Later, we will consider two cases, namely the squared Euclidean distance and general Bregman distances, and discuss how the choice of D impacts the architecture of the resulting neural network.

However, in its current form, Problem 2.1 is not amenable for optimization. Indeed, we need to specify parametric expressions for both ψ and $\{f_l\}_{l=0}^{L-1}$. Concerning the former, we

[1] Simple is meant in the sense that its proximity operator (see Equation (7)) has a closed-form expression.

make the common assumption that ψ denotes a linear mapping, hence promoting representations so that $\{z_i^{(L)}, y_i\}_{i=1}^n$ can be linearly separated. Concerning the latter, we suggest to restrict to the following class of functions.

Definition 2.1 (Class of layer-wise functions $\mathcal{F}$)**.** We consider bi-linear functions $\{f_l\}_{l=0}^{L-1}$ such that $(\forall z \in \mathbb{R}^d, \forall z_i^{(l)} \in \mathbb{R}^d)$

$$f_l(z, z_i^{(l)}) = z_i^{(l)\top} A_l^\top z - b_l^\top z - c_l^\top z_i^{(l)} + \delta_l, \tag{4}$$

where $A_l \in \mathbb{R}^{d\times d}$, $b_l \in \mathbb{R}^d$, $c_l \in \mathbb{R}^d$ and $\delta_l \in \mathbb{R}$. In addition we let

$$\mathcal{F}(\mathbb{R}^d \times \mathbb{R}^d) = \Big\{ f_l \mid f_l \text{ is as in (4)} \Big\}. \tag{5}$$

As we will see in the next sections, for such choice, each of the lower-level problems in (3) yields a closed form expression. In addition, the bi-linearity of f_l will be pivotal to stress connections with linear layers of neural networks.

3. Connection to Multilayer Networks

In this section, we explore more in depth the Problem 2.1 when D stands for the squared Euclidean distance. For peculiar choice of g, we show that it boils down to the training of feedforward neural networks.

3.1. Layer-wise update with quadratic distance

By hinging on bi-linear functions f_l from Definition 2.1 and quadratic distance D, then each of the L lower-level minimization problems of Problem 2.1 reads

$$\begin{aligned} z_i^{(l+1)} &= \operatorname*{argmin}_{z\in\mathbb{R}^d} f_l(z, z_i^{(l)}) + \frac{1}{2}\|z - z_i^{(l)}\|^2 + g(z), \\ &= \operatorname{prox}_g\Big(z_i^{(l)} - \nabla_1 f_l(z, z_i^{(l)})\Big), \\ &= \operatorname{prox}_g\Big(\underbrace{(\mathrm{I}_d - A_l)}_{\triangleq W_l} z_i^{(l)} + b_l\Big), \end{aligned} \tag{6}$$

where we have introduced the variable $W_l \in \mathbb{R}^{d\times d}$ and the proximity operator of g, defined as

$$(\forall u \in \mathbb{R}^d) \quad \operatorname{prox}_g(u) = \operatorname*{argmin}_{v\in\mathbb{R}^d} g(v) + \frac{1}{2}\|u - v\|^2. \tag{7}$$

Hence, each layer update yields a closed form expression taking the form a single forward-backward step (Chen & Rockafellar, 1997; Combettes & Wajs, 2005). Therefore, Problem 2.1 can be equivalently rewritten as follows.

Problem 3.1. Given some training data set $\{x_i, y_i\}_{i=1}^n$ made of n samples, where $\{x_i, y_i\} \in \mathbb{R}^d \times \mathbb{R}^c$ for every $i \in [n]$, solve

$$\begin{aligned} &\operatorname*{minimize}_{\substack{\psi\in\mathcal{B}(\mathbb{R}^d,\mathbb{R}^c)\\ \{W_l,b_l\}_{l=0}^{L-1}\in(\mathbb{R}^{d\times d}\times\mathbb{R}^d)^L}} \sum_{i=1}^n \ell(\psi(z_i^{(L)}), y_i) \quad \text{where} \\ &\forall i \in [n], \quad \begin{cases} z_i^{(0)} = x_i \\ \text{for } l = 0, 1, \dots, L-1 \\ \quad z_i^{(l+1)} = \operatorname{prox}_g(W_l z_i^{(l)} + b_l) \end{cases}. \end{aligned} \tag{8}$$

As discussed in (Bibi et al., 2019; Combettes & Pesquet, 2020), we also argue that the compositional form in (8) is evocative of feedforward neural networks made of L layers. In the next section, we further strengthen the existing connections.

3.2. Connection with neural networks

Problem 3.1 yields similarities with the training of feedforward 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 A_l, appearing in (6), is symmetric as well. In addition, as shown in (Combettes & Pesquet, 2020), the proximity operator prox_g can match a variety of activation functions depending on the choice of the function g. Below, we recall two of such examples.

Example 3.1 (ReLu)**.** The rectified linear unit function $\rho\colon t \in \mathbb{R} \mapsto \max(t, 0) \in \mathbb{R}$ can be expressed as the proximity operator prox_g of $g = \imath_{[0,+\infty[}$. Henceforth, prox_g reduces to the projection onto the positive orthant.

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

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

Here, we go one step further than (Combettes & Pesquet, 2020) by additionally linking activation functions to strongly convex Legendre functions (see Definition A.1). This is the subject of Proposition 3.1 and Proposition 3.2.

Proposition 3.1. *Given some 1-strongly convex Legendre function Φ, then for $g(\cdot) = \Phi(\cdot) - \frac{1}{2}\|\cdot\|^2$,*

$$(\forall t \in \operatorname{int}\operatorname{dom}\Phi), \quad \operatorname{prox}_g(t) = \nabla\Phi^{-1}(t). \tag{9}$$

Proposition 3.2 (Informal connection)**.** *Many activation functions ρ can be written as the inverse gradient of strongly convex Legendre functions Φ, i.e., $\rho = \nabla\Phi^{-1}$.*

We report in Table 1 some of the most common activation functions and their link to Legendre functions. To the best

of our knowledge, such connection with Legendre functions was not explicitly derived. We believe that this might be useful to design new activation functions tailored to the geometry of the representation learning problem at hand. In the next section, we will further build upon this observation in order to devise a novel neural network architecture.

4. Bregman Multilayer Network

While in the previous section we have considered quadratic distance D, we now investigate a more general class involving the use of Bregman distances. In addition, we show that, for some regularizer g, the lower-level of Problem 2.1 yields a new type of neural network. An extension to varying number of neurons per layer is provided in Appendix B.

4.1. Layer-wise update with Bregman distances

We begin 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, $(\forall u \in \operatorname{dom}\Phi, v \in \operatorname{int}\operatorname{dom}\Phi)$,

$$D_\Phi(u,v) = \Phi(u) - \Phi(v) - \langle \nabla\Phi(v), u - v\rangle. \tag{10}$$

In particular, we recover the squared 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, each layer-wise update can be rewritten as

$$\begin{aligned} z_i^{(l+1)} &= \operatorname*{argmin}_{z\in\mathbb{R}^d} f_l(z, z_i^{(l)}) + D_\Phi(z, z_i^{(l)}) + g(z) \\ &= \operatorname{prox}_g^\Phi\left(\nabla\Phi(z_i^{(l)}) - \nabla_1 f_l(z, z_i^{(l)})\right) \\ &= \operatorname{prox}_g^\Phi\Big(\nabla\Phi(z_i^{(l)}) \underbrace{-A_l}_{\triangleq W_l} z_i^{(l)} + b_l\Big) \end{aligned} \tag{11}$$

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 g with respect to Φ reads

$$\operatorname{prox}_g^\Phi(v) = \operatorname*{argmin}_{u\in\mathbb{R}^d} g(u) + \Phi(u) - \langle u, v\rangle. \tag{12}$$

Once again, by using the same arguments as in Section 3, Problem 2.1 boils down to:

Problem 4.1. Given some training data set $\{x_i, y_i\}_{i=1}^n$ made of n samples, where $\{x_i, y_i\} \in \mathbb{R}^d \times \mathbb{R}^c$ for every $i \in [n]$, and a strongly convex Legendre function Φ on $\mathbb{R}^d$, solve

$$\begin{aligned} &\underset{\substack{\psi\in\mathcal{B}(\mathbb{R}^d,\mathbb{R}^c)\\ \{W_l,b_l\}_{l=0}^{L-1}\in(\mathbb{R}^{d\times d}\times\mathbb{R}^d)^L}}{\text{minimize}} \sum_{i=1}^n \ell(\psi(z_i^{(L)}), y_i) \quad \text{where} \\ &\forall i \in [n], \quad \begin{cases} z_i^{(0)} = x_i \\ \text{for } l = 0, 1, \ldots, L-1 \\ \quad z_i^{(l+1)} = \operatorname{prox}_g^\Phi(\nabla\Phi(z_i^{(l)}) + W_l z_i^{(l)} + b_l) \end{cases}. \end{aligned}$$

Remark 4.1. It is worth stressing that Problem 3.1 is particular instance of Problem 4.1 obtained for $\Phi = \|\cdot\|^2$. In that case, since $\nabla\Phi(z_i^{(l)}) = z_i^{(l)}$ is a linear term, it is encapsulated into W_l, as shown in (6). However, in general, $\nabla\Phi$ is a non-linear operation and should thus be treated apart.

In the next section, we show how the compositional form in Problem 4.1 gives rise to a novel neural network architecture.

4.2. Proposed Bregman neural network

Similarly to the comparisons drawn in Section 3.2, we argue that the lower-level algorithm in Problem 4.1 bears some similarities to feedforward neural networks made of L layers. Indeed, we also identify $\{W_l\}_{l=0}^{L-1}$ and $\{b_l\}_{l=0}^{L-1}$ to the weights and biases, respectively. In the remaining of the section, we will show that, when $\operatorname{prox}_g^\Phi$ is an activation operator, then $\nabla\Phi$ is the inverse activation operator.

We start by establishing that a large variety of activation operators can be put in the form of a proximity operator with Bregman distances for specific choices of g and Φ. The starting point is the following counterpart of Proposition 3.1 for Bregman proximity operators.

Proposition 4.1. *Given some* 1*-strongly convex Legendre function* Φ*, then for* $g = \iota_{\operatorname{dom}\Phi}$ *being the indicator function of* $\operatorname{dom}\Phi$*, we have that*

$$(\forall t \in \operatorname{int}\operatorname{dom}\Phi), \quad \operatorname{prox}_g^\Phi(t) = \nabla\Phi^{-1}(t). \tag{13}$$

As a consequence, any invertible activation function ρ taking the form $\nabla\Phi^{-1}$ (see Proposition 3.2) can be equivalently written as the proximity operator of $g(\cdot) = \Phi(\cdot) - \frac{1}{2}\|\cdot\|^2$ with respect to the Euclidean metric (see Proposition 3.1) or in the form of a Bregman projection with respect to the Legendre function Φ (see Proposition 4.1). In other words, many activation functions can be recovered independently from the choice of the couple $\{D, g\}$, i.e.,

$$\begin{aligned} &\text{(Standard)} \quad g = \Phi - \tfrac{1}{2}\|\cdot\|^2 \text{ and } D = D_{\frac{1}{2}\|\cdot\|^2}, \\ &\text{(Bregman)} \quad g = \iota_{\operatorname{dom}\Phi} \text{ and } D = D_\Phi. \end{aligned}$$

However, interpreting the activation function from the viewpoint of a Bregman distance has the particularity of having

$\operatorname{dom}\phi$	Legendre function ϕ	Activation function $\rho \triangleq \nabla\phi^{-1}$	Inverse activation function $\rho^{-1} \triangleq \nabla\phi$
$[-1,1]$	$t \mapsto -\sqrt{1-t^2}$	ISRU: $t \mapsto t/\sqrt{1+t^2}$	$t \mapsto t/\sqrt{1-t^2}$
$[0,1]$	$t \mapsto t\log t + (1-t)\log(1-t)$	Sigmoid: $t \mapsto \frac{1}{1+e^{-t}}$	$t \mapsto \log(\frac{t}{1-t})$
$[-1,1]$	$t \mapsto -\frac{2}{\pi}\log(\cos(\frac{\pi}{2}t))$	Scaled atan: $t \mapsto \frac{\pi}{2}\arctan(t)$	$t \mapsto \tan(\frac{2}{\pi}t)$
$[-A,A]$	$t \mapsto \frac{A}{2B}\log(A^2-t^2) + \frac{t}{B}\operatorname{arctanh}(\frac{t}{A})$	Scaled tanh: $t \mapsto A\tanh(Bt)$	$t \mapsto (1/B)\operatorname{arctanh}(t/A)$
$[-1,1]$	$t \mapsto \sqrt{1-t^2} + t\arcsin(t)$	Sinusoidal: $t \mapsto \sin(t)$	$t \mapsto \arcsin(t)$
$\mathbb{R}$	$t \mapsto \cosh(t)$	Asinh: $t \mapsto \operatorname{arcsinh}(t)$	$t \mapsto \sinh(t)$
$\mathbb{R}_{>0}$	$t \mapsto (t/\beta)\log(\frac{e^{\beta t}-1}{1-e^{\beta t}}) - (1/\beta^2)\mathrm{Li}_2(e^{\beta t})$	Softplus: $t \mapsto (1/\beta)\log(\exp(\beta t)+1)$	$t \mapsto (1/\beta)\log(\exp(\beta t)-1)$

Table 1. **Connection between activation functions ρ and Legendre functions**. We report how $\nabla\phi^{-1}$ matches popular smooth activation functions for specific choice of Legendre functions ϕ. For the Softplus activation $\beta > 0$ and $\mathrm{Li}_n(x) = \sum_{k=1}^{\infty}\frac{x^k}{k^n}$ is the polylogarithm function. The larger β, the closer to the ReLu activation. For the scaled hyperbolic tangent, $A = 17159$ and $B = 0.6666$.

the additional term $\nabla\Phi(z^{(l)})$ appearing in Problem 4.1 which was absent in Problem 3.1. As a matter of fact, it follows from Proposition 3.2 that this terms is the inverse activation applied to the previous representation. Henceforth, the compositional form in Problem 4.1 can be rewritten in terms of the proposed Bregman feedforward layers.

Definition 4.2 (Bregman feedforward layer)**.** Let ρ be some invertible activation function. Then, the Bregman feedforward layer reads:

$$z^{(l+1)} = \rho\Big(\rho^{-1}(z^{(l)}) + W_l z^{(l)} + b_l\Big). \tag{14}$$

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

$$\begin{aligned} z^{(l+1)} &= \rho\Big(\rho^{-1}(z^{(l)}) + W_l z^{(l)} + b_l\Big), \\ &= \rho\Big(\rho^{-1}(z^{(l)})\Big) = z^{(l)}. \end{aligned} \tag{15}$$

In other words, the neuron is the identity. The reader is invited to refer to Appendix C for a complementary point of view on (15). 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 (15). Last but not least, (14) paves the way to an ordinary differential equation of the form

$$\frac{\mathrm{d}\rho^{-1}(z(t))}{\mathrm{d}t} = W(t)z(t) + b(t), \tag{16}$$

involving ρ^{-1} in place of the identity present for residual networks (Chen et al., 2018).

5. Numerical Experiments

In this section, we compare both standard multilayer perceptrons and residual networks against their proposed Bregman variants. The purpose of these experiments is not to outperform the accuracy of top leading neural networks but to assess, on simple architectures trained with vanilla optimizers, the added benefits. A Pytorch package is publicly available[2].

5.1. Bregman Multi-layer Perceptron

Throughout this section, we compare the performance of standard multi-layer perceptrons (MLP) against the proposed variant with Bregman layers.

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

$$z^{(l+1)} = \rho\Big(W_l z^{(l)} + b_l\Big),$$

and the proposed Bregman layer update

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

There, an additional linear operator M_l to be learned has been added to handle the general case where the dimensions of $z^{(l)}$ and $z^{(l+1)}$ differ. When those dimensions match, we stick to $M_l = \mathrm{I}_d$, instead. Details and justifications are provided in Appendix B.

5.1.1. Two-spiral Dataset

We begin by comparing the two MLPs on a 2 dimensional, yet challenging, binary classification task which proves to be useful for illustrating the benefits of the proposed variant.

Setting. The two-spiral dataset is a widely used benchmark for binary classification (Chalup & Wiklendt, 2007). Here, we reproduce a similar dataset and form 10^3 data points on two intertwined spirals which cannot be linearly separated. An illustration is reported in Figure 1. The purpose of the experiments is to train MLP so that the learned representation of the two-spiral dataset can be linearly separated. To do so, we consider $L \in \mathbb{N}^+$ layers with arctan activations and 2 neurons per layer. MLPs are learned using a plain stochastic gradient descent (SGD) optimizer. The batch-size, learning

[2] http://github.com/JordanFrecon/BregmaNet

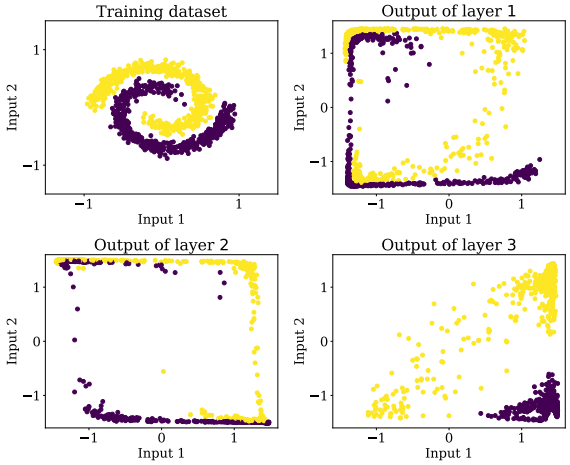

Figure 1. **Two-spiral Dataset: Bregman MLP.** We represent the inputs dataset as well as the representation learned at each layer.

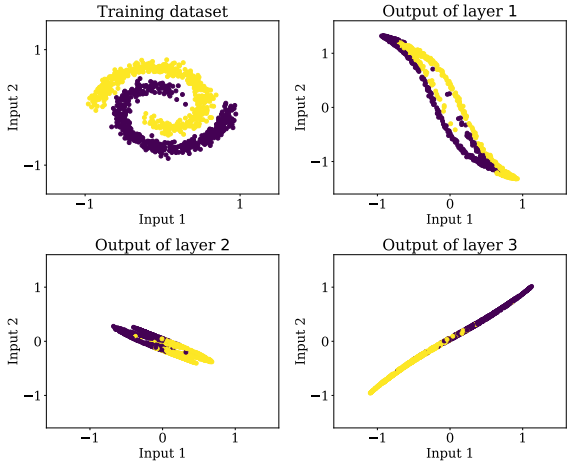

Figure 2. **Two-spiral Dataset: Standard MLP.** We represent the inputs dataset as well as the representation learned at each layer.

100-layers MLP networks	Test accuracy
Bregman MLP (deterministic)	**98.16 (± 1.74)**
Standard MLP (deterministic)	74.62 (± 1.51))
Bregman MLP (random)	96.36 (± 4.85)
Standard MLP (random)	50.00 (± 0.00)

Table 2. **Two-spiral Results: 100-layers MLPs.** Test accuracies are reported for two types of parameters' initializations (random and deterministic, see Definition D.1).

rate and number of epochs are set to 16, 10^{-2} and 500, respectively.

Quality of the learned representation. We first trained simple 3-layers standard and Bregman MLP achieving training accuracy of 93.50% and 99.99%, respectively. Note that, for the standard MLP, several rerun over multiple initialization are required in order to achieve such accuracy. For illustration purposes, we report in Figure 1 and Figure 2 the outputs of each layer for both MLPs. Interestingly, although the output at the third layer can almost be linearly separated in both cases, they exhibit different behaviors. Indeed, the Bregman MLP tends to cluster the data points on the whole space while the Standard MLP does it on a smaller manifold. Complementary results, reported in Appendix D.1, additionally support that the Bregman MLP leads to higher test accuracies irrespectively of the activation function.

We now consider the case where the number of hidden layers is large by setting $L = 100$. This setting is traditionally known to be challenging for training feedforward neural networks (Glorot & Bengio, 2010). As such, many tricks and strategies (e.g., initialization, preprocessing) have been devised to train deep models (see (Hasanpour et al., 2016) and references therein). Still, we stick to the vanilla optimization setting with a fixed learning rate (validated on a coarse grid in $10^{-4}, \ldots, 10^{0}$), a SGD optimizer with batch-size 16 and an early stopping strategy based on the validation accuracy.

Impact of the initialization. Following the common observation that MLPs trained with SGD from random initialization are performing poorly (Glorot & Bengio, 2010), we compare the performance of both Standard and Bregman deep MLP trained with two initialization strategies. To this regard, we consider the usual random initialization as well as a deterministic initialization detailed in the appendix (see Definition D.1). Averaged test accuracies over multiple seeds, displayed in Table 2, confirms that the standard MLP performs badly (i.e., up to 74.64% accuracy for a specific initialization). On the contrary, the Bregman MLP managed to achieve up to 98.16% test accuracy while being less sensitive to the initialization.

Layer-wise behavior. Here, we stick to the deterministic initialization. Once the 100-layers MLP models are trained, we compute *a posteriori* the norm of the parameters (weights and biases) obtained at each layer. Average results over multiple seeds are reported in Figure 3 (left plot). We observe that both methods exhibit very different layer-wise behaviors. On the one hand, the norm of the standard MLP parameters slowly grows with the number of layers while the proposed Bregman variant shows different order of magnitude between the parameters of the first and last layers. This suggests that the last layers play a minor role and could potentially be removed with a negligible impact on the performance. We now turn to the norm difference between two layers representation Figure 3 (right plot) which permits to gain some insight about the stability of the inner dynamical system (1). Results indicates that the standard MLP is more prone to instabilities as suggested by the increasing behavior from layers 50 to 100. On the contrary, the Bregman MLP exhibits a decreasing trend which is a sign of a more stable dynamic.

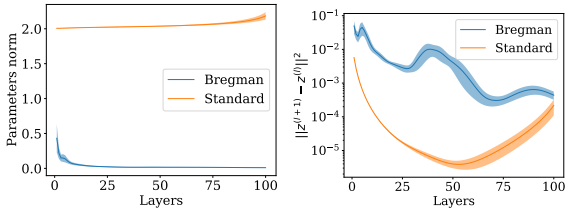

Figure 3. **Two-spiral Results: layer-wise behavior of 100-layers MLP.** Left: norm of the parameters (weights and biases) of each layer. Right: average norm difference of the output of consecutive layers.

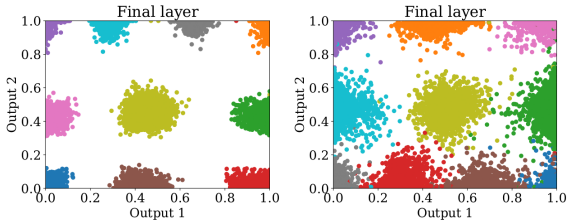

Figure 4. **MNIST Dataset: learned 2D representation.** We report in the left plot (resp. right plot) the learned representation through a Bregman MLP (resp. standard MLP) achieving a train accuracy of 100% (resp. 99.70%) and test accuracy of 94.74% (resp. 94.40%). Each color stands for a different label.

5.1.2. MNIST DATASET

We now turn to the popular MNIST dataset which is known to be easily classifiable by resorting to simple MLPs.

Setting. We conduct experiments similar to the setting of (Cireşan et al., 2010). The latter is known to achieve under 1% test accuracies with simple MLP architectures. All details are reported in the appendix (see Section E.1).

Quality of the learned representation. In this first experiment, we investigate the ability of MLPs to learn a meaningful 2 dimensional representation of the MNIST dataset. To this end, we consider sigmoid based MLPs made of 3 hidden layers of size 784, 2 and 2, respectively. The output layer is a linear layer used to map each 2 dimensional point to one of the 10 labels. The learned representation is reported in Figure 4. We observe that the Bregman MLP (left plot) yields a more compact clustering of the data points of the same label, hence confirming the same observation drawn on the two-spiral dataset.

Impact of depth and activation functions. A major challenge when training neural networks is to avoid the vanishing gradient phenomenon which can be caused by activation functions having small ranges of gradient values and the depth of the networks (Hochreiter et al., 2001). Here, we reproduce multiple settings encompassing such scenario. To do so, we consider MLPs with either sigmoid or scaled hyperbolic tangent activation. The former is known to prone very small gradient values during back-propagation while the latter has received great success instead. We additionally consider various number of hidden layers (2, 9) and various number of neurons per layer (16, 128, 1024). Performances are reported in Figure 5. We observe that both Standard and Bregman MLP perform equally well with $L = 2$ layers (left plots) with either sigmoid (top left) or scaled hyperbolic tangent (bottom left) activation function. Instead, for a larger number of layers (right plots), the Standard MLP performs poorly with sigmoid activations (bottom right) as opposed to the Bregman variant. We believe that the well behavior of Bregman MLPs is due to the presence of the ρ^{-1} term which, similarly to the skip connection of residual networks

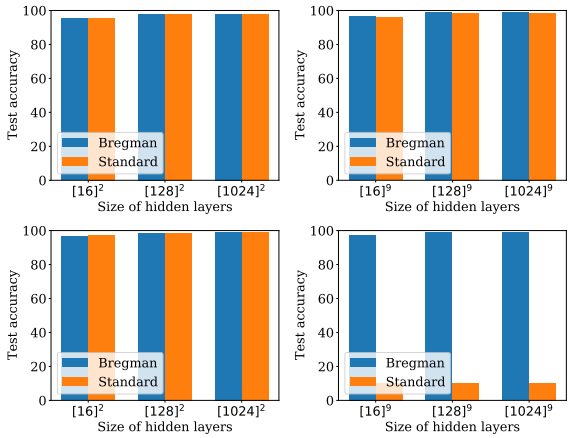

Figure 5. **MNIST Results: impact of the architecture.** We consider MLPs with scaled hyperbolic tangent activation function (top) or sigmoid activation function (bottom). For both, we report the test accuracies of both standard and Bregman MLP with $L = 2$ (left) or $L = 9$ (right) layers, and several numbers of neurons per layer (i.e., 16, 128 or 1024).

(ResNets), allows to mitigate vanishing gradient issues.

5.2. Bregman Residual Neural Network

Following the discussion done in Section 4.2, the proposed Bregman layer exhibits some similarities with ResNet since the skip connection is reminiscent of our added ρ^{-1} term. By hinging on the resemblances, we introduce and compare the following BregmanResNet.

Architectures compared. Our proposed variant differs from classical ResNet solely by the architecture of the residual blocks used. To this purpose, we recall below its standard form where the convolutional layers usually also encapsulate batch normalizing layers.

$$z \leftarrow \mathrm{ReLu}\Big(z + \mathrm{Conv2}\Big[\mathrm{ReLu}(\mathrm{Conv1}[z])\Big]\Big). \qquad (17)$$

Our proposed variant is defined for any invertible activation

	Test accuracy	ℓ_∞-Robust accuracy	ℓ_2-Robust accuracy
BregmanResNet20 (atan)	89.11 (± 0.20)	33.86 (± 0.68)	50.56 (± 0.52)
BregmanResNet20 (tanh)	89.28 (± 0.28)	35.29 (± 1.51)	51.68 (± 1.42)
BregmanResNet20 (sigmoid)	89.75 (± 0.23)	33.26 (± 1.47)	50.40 (± 1.46)
BregmanResNet20 (softplus)	**90.82 (± 0.12)**	**42.46 (± 1.25)**	**53.13 (± 1.68)**
ResNet20	90.80 (± 0.18)	40.84 (± 0.71)	51.65 (± 1.32)
BregmanResNet110 (softplus)	**91.52 (± 0.89)**	**48.24 (± 1.51)**	**58.09 (± 1.92)**
ResNet110	90.39 (± 1.07)	47.78 (± 1.66)	56.29 (± 1.76)

Table 3. **CIFAR-10 Results.** We compare the performance of ResNet20 and ResNet110 as well as its proposed Bregman variant trained on CIFAR-10. Test accuracies are averaged over 10 random initializations.

function ρ as follows

$$z \leftarrow \rho\Big(\rho^{-1}(z) + \mathrm{Conv2}\Big[\rho(\mathrm{Conv1}[z])\Big]\Big). \qquad (18)$$

There, the residual term is replaced by $\rho^{-1}(z)$ while ρ is used in placed of the ReLu activation function.

5.2.1. CIFAR-10 DATASET

We now conduct experiments on the CIFAR-10 dataset which consists of 50k training images and 10k testing images in 10 classes (Krizhevsky & Hinton, 2009).

Setting. We follow the original training data augmentation: 4 pixels are padded on each side, a 32×32 crop is randomly sampled from the padded image or its horizontal flip, and the pixels intensity are rescaled. Our implementation is based on a rework of (Idelbayev, 2018). Concerning the optimization, we use the same setting as in the original ResNet paper (He et al., 2016), namely a learning rate of 10^{-1}, weight decay of 10^{-4}, momentum of 0.9. In addition, the optimization is done over 182 epochs with a decreasing of the learning by a factor 10 at the 91^{th} and 136^{th} epochs.

Test accuracy. We report in Table 3 the test accuracies of ResNet20 as well as its Bregman variants obtained for various choices of activation functions. Interestingly, solely the variant with the SoftPlus activation, which is a smooth approximation of ReLu, managed to match the performance of the standard ResNet20. In order to further contrast these results, we additionally compare the top-2 test accuracies: 97.05% ± 0.24 (BregmanResNet20 with SoftPlus) and 95.86% ± 0.27 (ResNet20). The latter unveil a significant improvement with the Bregman variant. In addition, we compare the performance of a deeper residual network, namely the ResNet110. Note that we did not resort to the optimization heuristic from (He et al., 2016) but stick to the same setting, instead. Test accuracies are reported in Table 3 and show a significant improvement over the baseline. We believe this is due to the better training behavior of the proposed variant when dealing with deep layers.

Adversarial robustness. We additionally compare the robustness of both ResNet models against adversarial attacks. Adversarial attacks are quasi imperceptible perturbations, which added to the original image, manage to fool the prediction of the model. To this regard, we consider both ℓ_∞-attacks and ℓ_2-attacks. Concerning the former, we resort to the popular *fast gradient sign method* (Goodfellow et al., 2015) with a maximal perturbation of $\varepsilon = 8/255$. Concerning the latter, we make use of the PGD attack of (Madry et al., 2018) with a maximum ℓ_2-perturbation of $\varepsilon = 0.5$. For both architectures, we see that the proposed variant is more robust than its standard counterpart. More interestingly, altough the test accuracies on ResNet20 are comparable, the Bregman variant shows a significant improvement in terms of robust test accuracies. In order to complement the added robustness, we additionally quantify the margins to the decision boundaries through the distance between the two largest outputs of the last layer. Results, averaged over the multiple seeds, read 14.6 (BregmanResNet20 with SoftPlus) and 9.35 (ResNet20). Hence, they do support that the Bregman variant benefits from a higher separation between the classes as previously observed in Figure 4.

	Test accuracy
ResNet18	65.10 (± 0.36)
BregmanResNet18	**65.25 (± 0.13)**
ResNet110	71.45 (± 0.04)
BregmanResNet110	**72.28 (± 0.12)**

Table 4. **CIFAR-100 Results.** Test accuracies are averaged on 10 random initializations. The Bregman variant is devised with the softplus activation function.

5.2.2. CIFAR-100 DATASET

We turn to the more challenging CIFAR-100 dataset made of 500 training images and 100 testing images per each of the 100 classes (Krizhevsky & Hinton, 2009).

Setting. We follow the same pre-processing used for the CIFAR-10 dataset and reproduce the experimental procedure of (Zhang et al., 2019). Namely, we train the model during 200 epochs with batches of 128 images and decay the learning rate by a factor of 5 at the 60th, 120th, and 160th epochs. We also resort to a SGD optimizer with initial learning of 0.05, momentum of 0.9 and weight decay of 10^{-3}.

Performance. We report in Table 4 the averaged test accuracies. Once again, we observe a minor improvement over the baseline. More interestingly, putting these results into perspective with the corresponding train accuracies (being respectively 99.94% and 99.92% for BregmanResNet110 and ResNet100) suggests that the proposed variant benefits from slightly better generalization properties.

6. Conclusion

The present paper framed the learning of a representation mapping as a multi-layered bilevel optimization. We have shown that for some quadratic distances, the learning framework boils down to the training of a feedforward neural network. In addition, by elaborating on more general distances, we proposed the Bregman 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. In addition, it ensures that whenever the weights and biases are zero the layer reduces to the identity. We have shown experimentally that two interesting aspects follow from this property. First, the training of deep architectures is eased. Second, after some depth all layers parameters are almost zero and thus subsequent layers could potentially be removed without hampering the prediction performance. We believe, this may reduce the memory needed to store very deep architectures. Future works should further study both aspects in very deep architectures and investigate how the presence of the inverse activation mitigates vanishing gradients issues. In addition, following up on the connection (16) with neural ordinary differential equations, an extension of Bregman neural networks to continuous-depth and continuous-time latent variable is under study.

Acknowledgements

We thank the reviewers for their useful comments. Jordan Frecon would like to express his gratitude to the Department of Computational Statistics and Machine Learning (IIT, Genova, Italy) where part of this work was conducted during his postdoctoral position. The authors gratefully acknowledge the financial support of the French Agence Nationale de la Recherche (ANR), under grant ANR-20-CHIA-0021-01 (project RAIMO[3]).

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.

Bauschke, H. H., Borwein, J. M., et al. Legendre functions and the method of random bregman projections. *Journal of convex analysis*, 4(1):27–67, 1997.

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.

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., 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. doi: 10.1137/S1052623495290179.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Cireşan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. Deep, Big, Simple Neural Nets for Handwritten Digit Recognition. *Neural Computation*, 22(12):3207–3220, 12 2010. ISSN 0899-7667.

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.

[3] https://chaire-raimo.github.io

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.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M. (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

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.

Grigorescu, S., Trasnea, B., Cocias, T., and Macesanu, G. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

Hasanpour, S. H., Rouhani, M., Fayyaz, M., and Sabokrou, M. Lets keep it simple, using simple architectures to outperform deeper and more complex architectures. *arXiv preprint arXiv:1608.06037*, 2016.

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.

Hochreiter, S., Bengio, Y., Frasconi, P., and Schmidhuber, J. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In Kremer, S. C. and Kolen, J. F. (eds.), *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, 2001.

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.

Idelbayev, Y. Proper ResNet implementation for CIFAR10/CIFAR100 in PyTorch. https://github.com/akamaster/pytorch_resnet_cifar10, 2018.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

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.

Liu, H., Simonyan, K., and Yang, Y. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

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.

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.

Zhang, M., Lucas, J., Ba, J., and Hinton, G. E. Lookahead optimizer: k steps forward, 1 step back. 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. Curran Associates, Inc., 2019.

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

A. Legendre Functions

We recall below the definition of a Legendre function (Bauschke et al., 1997).

Definition A.1 (Legendre Function)**.** Let $\mathcal{X}$ be a Hilbert space and $\Phi \in \Gamma_0(\mathcal{X})$. Then Φ is called

- *essentially smooth* if Φ is differentiable on $\operatorname{int}\operatorname{dom}\Phi \neq \emptyset$ and $\|\nabla\Phi(v)\| \to +\infty$ as $v \mapsto \operatorname{boundary}\operatorname{dom}\Phi$,
- *essentially strictly convex* if Φ is strictly convex on every convex subset of $\operatorname{dom}\partial\Phi$,
- *Legendre* if Φ is both essentially smooth and essentially strictly convex.

B. Extension to Varying Number of Neurons

So far, we have assumed that the number of neurons is identical throughout the layers. In the following, we remove such hypothesis and denote by d_l the number of neurons at layer l. In addition, we consider that both D_Φ and g, hereafter denoted D_{Φ_l} and g_l, can also vary at each layer.

In order to extend the proposed framework to the scenario where two consecutive layers' representations have different dimensions, we consider the following inner dynamical system:

$$\begin{aligned} &z_i^{(0)} = x_i \\ &\text{for } l = 0, 1, \dots L-1 \\ &\left\lfloor \; z_i^{(l+1)} = \operatorname*{argmin}_{z \in \mathbb{R}^{d_{l+1}}} f_l(z, M_l z_i^{(l)}) + D_{\Phi_l}(z, M_l z_i^{(l)}) + g_l(z), \right. \end{aligned} \tag{19}$$

where $f_l \in \mathcal{F}(\mathbb{R}^{d_{l+1}} \times \mathbb{R}^{d_{l+1}})$, $g_l \in \Gamma_0(\mathbb{R}^{d_{l+1}})$ and $M_l \in \mathbb{R}^{d_{l+1} \times d_l}$ is additional term permitting to match the dimensions of $z_i^{(l)}$ and $z_i^{(l+1)}$.

The main difference with respect to the dynamical system in (3) is the presence of the matrices $\{M_l\}_{l=0}^{L-1}$. The latter are not prescribed but have to be learned in the same way as parameters of the functions $\{f_l\}_{l=0}^{L-1}$.

In order to shed some light on the space of admissible matrices M_l, we first recall from the definition of Bregman distances (see Definition 4.1), that one needs to ensure that $M_l z_i^{(l)} \in \operatorname{int}\operatorname{dom}\Phi_l$. To to do, we devise a sufficient condition in the next proposition.

Proposition B.1. *Suppose that each Legendre function Φ_l are built from an elementary Legendre function ϕ_l such that*

$$\Phi_l \colon u \in \mathbb{R}^{d_{l+1}} \mapsto \sum_{j=1}^{l+1} \phi_l(u_j), \quad \textit{with} \quad \operatorname{dom}\phi_l = \mathcal{I},$$

where $\mathcal{I}$ denotes a convex interval. Then $v^{(l)} \in \operatorname{dom}\Phi_{l-1}$ implies that $M_l v^{(l)} \in \operatorname{dom}\Phi_l$ for any $M_l \in \Delta_l^{l+1}$ where

$$\Delta_l^{l+1} = \left\{ M \in [0,1]^{d_{l+1} \times d_l} \;\middle|\; (\forall i \in [d_{l+1}]),\ \sum_{j=1}^{d_l} M_{i,j} = 1 \right\}.$$

Proof. Let $u^{(l)} = M_l v^{(l)}$. Then, for every $i \in [d_{l+1}]$, the i-th component of $u^{(l)}$ can be seen as a convex combination of $v^{(l)}$, i.e., $(u^{(l)})_i = \sum_{j=1}^{d_l} M_{i,j} (u^{(l)})_j$. Therefore, since each $(v^{(l)})_j \in \mathcal{I}$, with $\mathcal{I}$ convex, then so does $(u^{(l)})_i$. □

The assumption in B.1 is actually pretty standard. In the context of Bregman neural networks, it translates into the hypothesis that all the activation functions present in the network have the same range.

Remark B.1. By elaborating on Proposition B.1, one can recursively ensure that for every $l \in \{1, \dots, L\}$, $z_i^{(l)} \in \operatorname{dom}\Phi_{l-1}$ on the condition that so does the input data points $z_i^{(0)} = x_i$. This can always be enforced in practice by shifting and rescaling the input data.

We are now left with deriving the closed-form solutions of (19), i.e.,

$$\begin{aligned} z_i^{(l+1)} &= \operatorname*{argmin}_{z \in \mathbb{R}^{d_{l+1}}} f_l(z, M_l z_i^{(l)}) + D_{\Phi_l}(z, M_l z_i^{(l)}) + g_l(z) \\ &= \operatorname{prox}_{g_l}^{\Phi_l}\left(\nabla\Phi_l(M_l z_i^{(l)}) - \nabla_1 f_l(z, M_l z_i^{(l)}) \right) \\ &= \operatorname{prox}_{g}^{\Phi_l}\Big(\nabla\Phi(M_l z_i^{(l)}) \underbrace{- A_l M_l}_{\triangleq W_l} z_i^{(l)} + b_l \Big) \end{aligned} \tag{20}$$

where we have introduced the variable $W_l \in \mathbb{R}^{d_{l+1} \times d_l}$. We finally end up with the following multilayered bilevel problem.

Problem B.1. Given some training data set $\{x_i, y_i\}_{i=1}^n$ made of n samples, where $\{x_i, y_i\} \in \mathbb{R}^d \times \mathbb{R}^c$ for every $i \in [n]$, and

	lr = 0.01	lr = 0.1	lr = 1
Bregman (sigmoid)	91.60 (± 6.70)	**99.44 (± 0.38)**	94.32 (± 7.07)
Bregman (atan)	98.74 (± 0.36)	96.00 (± 6.99)	98.06 (± 2.27)
Bregman (tanh)	96.18 (± 0.88)	99.18 (± 0.64)	83.06 (± 9.52)
Standard (sigmoid)	64.36 (± 13.64)	77.38 (± 5.57)	79.84 (± 3.33)
Standard (atan)	84.04 (± 11.16)	86.20 (± 9.15)	77.74 (± 4.25)
Standard (tanh)	78.60 (± 9.23)	75.00 (± 5.55)	73.00 (± 1.15)

Table 5. **Additional Two-spiral Results: 3-layers MLP.** We report the test mean accuracies (over 5 realizations) obtained for various learning rate (lr).

	lr = 0.001	lr = 0.01	lr = 0.1
Bregman (non-informative)	93.24 (± 9.48)	98.16 (± 1.74)	94.98 (± 3.64)
Standard (non-informative)	74.68 (± 1.64)	74.62 (± 1.51)	68.02 (± 7.13)
Bregman (random)	72.98 (± 15.21)	96.36 (± 4.85)	93.08 (± 4.75)
Standard (random)	50.00 (± 0.00)	50.00 (± 0.00)	50.00 (± 0.00)

Table 6. **Additional Two-spiral Results: 100-layers MLP.** Results are reported for two types of parameters' initializations (random and non-informative, see Definition D.1) and for three learning rates (lr).

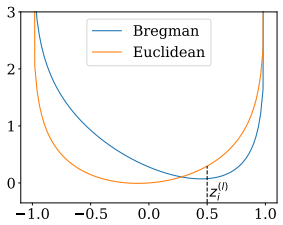

Figure 6. **Comparison of layer-wise objectives (2) for $f_l = 0$.** On the one hand, in Euclidean case, the objective reads $z \mapsto (\Phi(z) - \frac{1}{2}\|z\|^2) + \frac{1}{2}\|z - z_i^{(l)}\|^2$. On the other hand, for Bregman distances it solely reads $z \mapsto D_\Phi(z, z_i^{(l)})$.

1-strongly convex Legendre functions $\{\Phi\}_{l=0}^{L-1}$, solve

$$\underset{\substack{\psi \in \mathcal{B}(\mathbb{R}^d, \mathbb{R}^c) \\ \{W_l, b_l\}_{l=0}^{L-1} \in (\mathbb{R}^{d\times d} \times \mathbb{R}^d)^L \\ \{M_l\}_{l=0}^{L-1} \in \bigtimes_{l=0}^{L-1} \Delta_l^{l+1}}}{\text{minimize}} \sum_{i=1}^{n} \ell(\psi(z_i^{(L)}), y_i) \quad \text{where} \quad \forall i \in [n],$$

$$\begin{cases} z_i^{(0)} = x_i \\ \text{for } l = 0, 1, \dots, L-1 \\ \quad z_i^{(l+1)} = \operatorname{prox}_g^{\Phi_l}(\nabla\Phi_l(M_l z_i^{(l)}) + W_l z_i^{(l)} + b_l) \end{cases}.$$

Compared to Problem 4.1, Problem B.1 has the drawback of requiring to learn the extra matrices $\{M_l\}$. In practice, we do learn them in the same way we learn the weights $\{W_l\}$. The only difference is that we further need to project them into Δ_l^{l+1}.

C. Interpretation in terms of layer-wise objective

As previously discussed in Section 4.2, a large variety of activations $\nabla\Phi^{-1}$ can be equivalently written as the proximity operator of $g = \Phi - \frac{1}{2}\|\cdot\|^2$ (see Proposition 3.1) or as a Bregman projection with respect to the Legendre function Φ (see Proposition 4.1).

From the point of view of (1), each case results in minimizing different layer-wise functions $\mathcal{L}_l(\cdot, z_i^{(l)})$ recalled below

- (Standard) $z \mapsto f_l(z, z_i^{(l)}) + \Phi(z) - \frac{1}{2}\|z\|^2 + \frac{1}{2}\|z - z_i^{(l)}\|^2$,
- (Bregman) $z \mapsto f_l(z, z_i^{(l)}) + D_\Phi(z, z_i^{(l)})$.

For illustration purposes, we report in Figure 6 the two objectives in the particular case where $f_l = 0$ for $\phi\colon t \mapsto -\pi/2\log(\cos(\pi t/2))$. For both, the behavior near -1 and 1 confine the solution to lie within the bounds $[1, 1]$. In addition, we observe that, for the Bregman variant, the minimizer coincide with $z_i^{(l)}$ when $f_l = 0$, thus bringing another view on the observation (15).

D. Additional Results on Two-Spiral Dataset

In this section, we provide additional results complementing the experiments done in Section 5.1.1.

D.1. 3-layers MLPs

In order to complement the experiments done in Section 5.1.1, we report in Table 5 additional results for 3 layers based MLPs made of 2 neurons per layer. Test accuracies are averaged over 5 independent realizations and are reported for various activation functions and learning rates.

Irrespectively of the activation function and the learning rate, we observe that the proposed variant yields higher test accuracies.

D.2. 100-layers MLPs

In order to train a deep MLPs, we suggest to resort to the following non-informative initialization.

Definition D.1 (Non-informative initialization)**.** Given some layer with $d \in \mathbb{N}$ input and output dimensions, we let

- (Bregman) $W_l = 0_{d\times d}$ and $b_l = 0_d$,
- (Standard) $W_l = I_d$ and $b_l = 0_d$.

This change of treatment between standard and Bregman MLPs can be understood through the prism of Remark 4.1. Indeed, we recall that, for any Legendre function Φ, each layer update can be written as

$$z^{(l+1)} = \text{prox}_g^{\Phi}\Big(\nabla\Phi(z_i^{(l)}) + W_l z_i^{(l)} + b_l\Big). \tag{21}$$

In particular, for $\Phi = \frac{1}{2}\|\cdot\|^2$, we recover

$$z^{(l+1)} = \text{prox}_g\Big(z_i^{(l)} + W_l z_i^{(l)} + b_l\Big). \tag{22}$$

Henceforth, by i) applying a change of variable $W_l \leftarrow W_l + I_d$ and ii) identifying prox_g with some activation operation, we recover the standard MLP layer Section 3.1, i.e.,

$$z^{(l+1)} = \rho\Big(W_l z_i^{(l)} + b_l\Big). \tag{23}$$

Therefore, having $W_l = 0$ for Bregman MLP translates to $W_l = I_d$ for Standard MLP due to the change of variable.

In order to complement the discussion done Section 5.1.1, we provide additional results in Table 6 for various learning rates and for different initializations.

E. Additional Results on MNIST Dataset

In this section, we detail the experimental setting of Section 5.1.2 and provide additional experiments further illustrating the vanishing gradient problem.

E.1. Experimental setting of Section 5.1.2

We report below how we have tried to reproduce the same experimental setting as in (Cireşan et al., 2010) with the available information.

Data preprocessing. The 50k samples of the training set undergo random affine transformations keeping the center invariant. To this effect, we use random rotations between $(-11.25, +11.25)$ degrees and a random scaling selected in $(-0.825, +0.825)$. Then, each 32×32 pixels images are flatten into 784 dimensional vectors pre-processed so that the pixel intensity lie within the activation range. The original un-deformed training dataset is then use as validation set to perform early stopping. Both validation and test sets are flatten and rescaled in the activation range.

Note that, as opposed to (Cireşan et al., 2010), we do not perform data augmentation with elastic deformations.

Optimizer. All MLPs are trained using a stochastic gradient descent with batch size 100 with a decreasing step learning linearly decreasing of a factor 10^{-3} over 10^3 epochs. The initial learning rate is cross-validated over $\{10^{-3}, 10^{-2}, 10^{-1}\}$. For sigmoid-based MLP (resp. scaled hyperbolic tangent), we have found the best value to be 10^{-1} (resp. 10^{-3}).

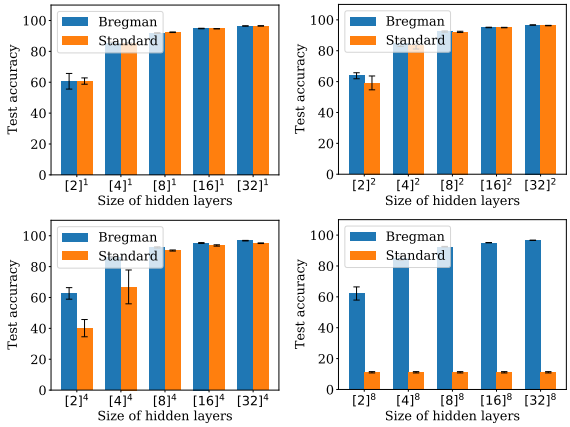

Figure 7. **Additional MNIST Results: impact of architecture.** We report additional results for sigmoid-based MLPs with varying number (1, 2, 4, 8), represented from top left to bottom right, and several number of neurons per layer (i.e., 2, 4, 8, 16 and 32). As the number of layers grows the performance of Bregman MLP remains constant while those of its standard counterpart drop significantly.

E.2. Additional experiments on sigmoid-based MLPs

In order to further highlight the vanishing gradient effect occuring with sigmoid-based MLPs, we investigate the impact of various architectures on the performance.

More specifically, we compare the test accuracies achieved by both standard and Bregman MLPs with sigmoid activation function and several hidden layers sizes and 1 to 8 hidden layers. This experiment is repeated over 5 different data splits. Average test accuracies are reported in Figure 7. For a single hidden layer (top left figure) we observe that both standard and Bregman MLP perform equally well. However, as the number of hidden layers grows (from top left to bottom right), the performances of standard MLP drastically drop while those of Bregman MLP remain constant.

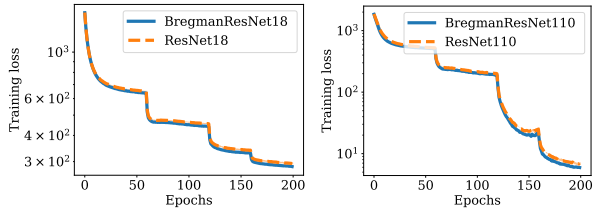

Figure 8. **Additional CIFAR-100 Results: Training behavior.** Left: ResNet18-based models. Right: ResNet110-based models.

F. Additional Results on CIFAR-100 Dataset

In order to complement the CIFAR-100 experiments conducted in Section 5.2.2, we additionally report in Figure 8 the averaged training loss for ResNet18, ResNet110 and their respective Bregman variant. Both standard and Bregman ResNets exhibit the same learning behaviour with a slightly lower training error for the Bregman versions.