Enhancing Deep Learning Algorithm Accuracy and Stability using Multicriteria Optimization: An Application to Distributed Learning with MNIST Digits

Davide La Torre[a], Danilo Liuzzi[b], Marco Repetto[a,d,e], Matteo Rocca[c]

[a]*SKEMA Business School, Université Côte d'Azur, France*
[b]*University of Milan, Italy*
[c]*Universitá degli Studi dell'Insubria, Italy*
[d]*University of Milan-Bicocca, Milan, Italy*
[e]*Siemens Italy, Milan, Italy*

Abstract

The training phase is the most crucial stage during the machine learning process. In the case of labeled data and supervised learning, machine learning entails minimizing the loss function under various constraints. We provide an innovative model for learning with numerous data sets, resulting from the application of multicriteria optimization techniques to existing deep learning algorithms. Data fitting is formulated as a multicriteria model in which each criterion measures the data fitting error on a specific data set. This is an optimization model involving a vector-valued function, and it has to be analyzed using the notion of Pareto efficiency. We present stability results for efficient solutions in the presence of input and output data perturbations. The multiple data set environment comes into play

Email addresses: davide.latorre@skema.edu (Davide La Torre), danilo.liuzzi@unimi.it (Danilo Liuzzi), marco.repetto@skema.edu (Marco Repetto), matteo.rocca@uninsubria.it (Matteo Rocca)

to eliminate the bias caused by the selection of a specific training set. To apply this concept, we present a scalarization strategy as well as numerical experiments in digit classification using MNIST data.
Keywords: Artificial Intelligence, Deep Learning, Machine Learning, Multicriteria Optimization, Classification, MINST data

---

1. Introduction

It is now generally agreed that Artificial Intelligence (AI) refers to an interdisciplinary field - encompassing biology, computer science, philosophy, mathematics, engineering and robotics, and cognitive science - concerned with simulating human intelligence using computer-based technologies. This is accomplished by teaching machines how to execute activities that ordinarily require human intelligence, such as visual perception, speech recognition, decision-making, and language translation (Wang and Barabási, 2021; Goel and Davies, 2011; Schank and Towle, 2000; de la Higuera, 2010).

Machine Learning (ML) is a subfield of AI that focuses on algorithms used to learn from data and make future predictions and judgments (Ripley, 1996). There are two primary families of ML algorithms: supervised learning refers to the process of learning an unknown function using labeled training data and example input-output pairs. In contrast, "unsupervised learning" refers to the detection of previously unnoticed patterns and information in an unlabeled data set.

Deep Learning (DL) is an AI discipline and a type of ML technique aimed at developing systems that can operate in complex situations (Goodfellow et al., 2016). Deep architectures underpin DL systems (Bottou et al.,

2007).

Many current DL applications, aided by the quantity of data, necessitate a significant amount of training. Local rules, on the other hand, posed severe limitations in terms of data transfer in distributed systems (Ahmed et al., 2021). As a result, Konečnỳ et al. (2016) proposed the Federated Learning notion (FL). According to Bonawitz et al. (2019) FL is a distributed learning methodology that allows model training on a vast corpus of decentralized data. With dispersed data across multiple nodes, the Decision Maker (DM) must deal with opposing node objectives as well as potential hostile threats (Bagdasaryan et al., 2020).

Multicriteria Optimization (also known as MOP) is a discipline of Operations Research and Decision Making that studies optimization models with multiple and often contradictory criteria. In the last fifty years, a rising number of researchers have contributed to this topic, and a range of approaches, methods, and strategies have been developed for use in a variety of disciplines, spanning from economics to engineering, finance to management, and many others. Multicriteria decision-making problems are more difficult to assess and computationally intensive. They usually do, however, lead to more informed and better decisions.

The application of MOP in DL in order to allow for the learning of numerous data sets is a novel and unexplored area of research (Yang et al., 2020). With the rise of Edge Computing (EC) and the Internet of Things (IoT), there has been a considerable increase in the demand for these sorts of applications, which has resulted in an increase in the cost of development. In order to allow for such types of applications, we propose an innovative

and rigorous technique that is also practical in nature. This paper extends to a more general setting the approach proposed in Bryson et al. (2021) in which the authors consider a model integrating into a unique framework three different criteria, namely, a data-fitting term, and the entropy and the sparsity of the set of unknown parameters. This paper has also been inspired by other contributions in the literature related to the application of MOP to inverse problems and estimation of unknown parameters in complex systems as, for instance, in Berenguer et al. (2016) and Kunze and La Torre (2020). Finally, it is worth mentioning that DL algorithms can be embedded into MOP decision-making models to make them more informative and effective, as in La Torre et al. (2021). Other recent applications of MOP to DL can be found in Repetto et al. (2021), Hafiz et al. (2021).

In this paper, the machine training problem is initially formulated as an abstract optimization problem involving a vector-valued functional. Consequently, the concept of minimization is intended in the Pareto sense. We offer results about the stability and convergence of the set of efficient solutions. We then extend it to the case of machine training with numerous data sets. We offer numerical experiments based on scalarization techniques and test their performance utilizing digit data from the MINST data set (see, for example, Deisenroth et al. (2020); Poole and Mackworth (2017); Shalev-Shwartz and Ben-David (2014); Barber (2012); Jiang (2022); Moitra (2018); Shah (2020)). Our findings indicate that the employment of multi-criteria optimization techniques can also improve the training algorithm's precision.

The structure of the paper is as follows: Deep Learning Architectures

and Multicriteria Optimization are introduced in Section 2. Section 3 gives a vector-valued formulation of machine training using labeled data and the principal stability properties. Section 4 introduces an extended machine training with several data sets. After presenting several numerical experiments in Sections 5, 6, and 7, Section 8 concludes.

2. Preliminaries

2.1. Deep Learning: A Literature Review

It is well known that Deep Learning is a subarea of Machine Learning that focuses on Artificial Neural Networks (ANNs). ANNs are networks composed of many interconnected processing nodes or neurons that can learn how to recognize complex patterns from data. ANNs are used for different applications, mostly for image recognition and classification, pattern recognition, and time series prediction. In Deep Learning, the so-called deep architectures are combinations of different ANNs.

In a very abstract formulation, a general deep architecture is defined as:

$$\mathcal{F} = \{f(\cdot, w), w \in \mathcal{W}\}$$

where $f(\cdot, w)$ is a shallow architecture (for instance, the Perceptron proposed in Rosenblatt (1958)). The origin of Deep Learning dates back between the 40s and the 60s in a broader area called Cybernetics. Before the fundamental paper by Rosenblatt, other authors (see, for instance, McCulloch and Pitts (1943)) proposed a binary neurons-based system able to implement simple logic operations. Nowadays, neither the Perceptron nor the system proposed by McCulloch and Pitts (1943) are used in the current

ANN configurations. Modern architectures, instead, rely on gradient-based optimization techniques based on the Stochastic Gradient Descent (SGD) algorithm and its recent variants (Saad, 1998). One of the first architectures that have been trained using gradient-based methods is the Multilayer Perceptron (MLP) (McClelland et al., 1987). The MLP architecture, presented in Figure 1, is inspired by the brain's essential functioning, and it emulates a simple feedforward network of neurons, historically called "Perceptrons". From a modeling perspective, the main activity of a neuron is described by means of an activation function that can either be linear, sigmoidal, or piecewise.

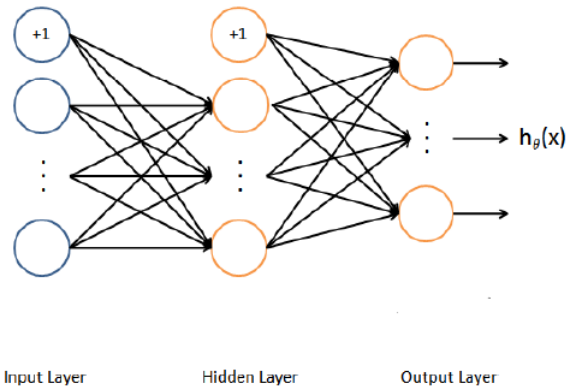

Figure 1: The architecture of the Multilayer Perceptron with a single hidden layer and fully connected nodes

MLPs are widely used nowadays in many applications. In oncology, Sharma et al. (2018) used an MLP architecture for breast cancer detection achieving remarkable accuracy and relatively low training time. In time

series forecasting Nosratabadi et al. (2021) used an MLP to forecast food production, leveraging the MLP capability to learn from nonlinear historical data. In engineering, Sopelsa Neto et al. (2021) relied on MLP to classify ceramic insulators based on an ultrasonic inspection in order to avoid possible interruptions of electricity. Another architecture well known for its relevant applications to image processing is the Convolutional Neural Network (CNN). The first definition of CNN goes back to the notion of Neocognitron proposed in Fukushima and Miyake (1982), but the first implementation in a supervised learning setting was proposed in LeCun et al. (1989) for digit recognition. With respect to the MLP architecture, a classical CNN does not rely only on fully connected layers, even if feature extraction through filtering is performed by convolution layers. A schematic representation of the functioning of a convolution layer is presented in Figure 2. As one can see, a second pooling layer is attached to the initial one to allow for dimensionality reduction.

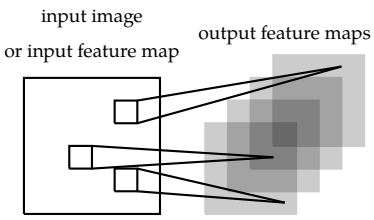

Figure 2: Interaction between a convolution layer and a pooling layer in a Convolutional Neural Network architecture

CNNs are primarily used for image classification as they rely on the local dependencies of the pixels used as features of the ANN architecture.

In such cases, CNNs perform better than the standard MLP. In breast cancer prediction Desai and Shah (2021) found higher accuracies compared to the MLP. Still, in medical image recognition, Billones et al. (2016) used a CNN applied to three dimensional MRI scans to detect Alzheimer's disease and cognitive impairment. In some cases, CNNs have been repurposed in order to be utilized with tabular data as in the case of Zhu et al. (2021), achieving remarkable performance. With the renewed interest in ANNs and DL, more advanced and sophisticated architectures have been proposed to overcome the problems presented in earlier ANNs as, for instance, the problem of the vanishing gradient. Initially proposed by He et al. (2015) ResNet differs from the canonical MLP architecture in that it allows for "shortcut connections" that mitigate the problem of degradation in the case of multiple layers. Although the usage of shortcut connections is not new in the literature (Venables and Ripley, 1999), the key proposal of He et al. (2015) was to use identity mapping instead of any other nonlinear transformation. Figure 3 shows the smallest building block of the ResNet architecture in which both the first and the second layer are shortcutted, and the inputs x are added to the output of the second layer.

The rationale behind ResNet is that by residual learning, the solvers will be able to capture identity mappings that otherwise will be lost in multiple nonlinear layers. With shortcuts, identity mapping is achieved by simply annihilating the weights of the input layers that have been shortcutted. ResNets proved to be a parsimonious yet effective architecture in several image classification tasks (Canziani et al., 2017; Chen et al., 2018; Wu et al., 2019). An application of MOP and ResNet is provided in Khan et al. (2019).

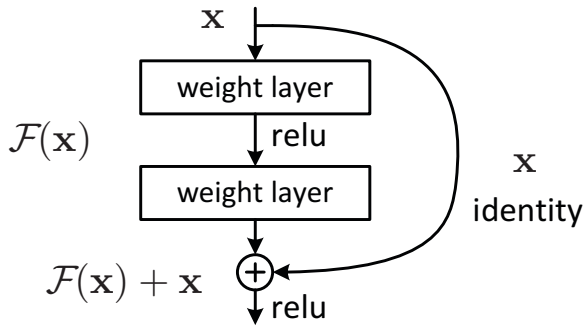

Figure 3: A shortcut connection layer with identity mapping characterizing the Residual Network architecture

In this paper, the authors proposed a Transfer Learning approach based on multiple training instances. In particular, the authors addressed the problem of breast cancer prediction, achieving a remarkable performance.

2.2. Basics on Multicriteria Optimization

In Multicriteria Optimization (MOP), we consider an optimization model with several conflicting criteria. In this section, we recall some basic notions in MOP that will be used in the following sections. Given a compact subset Ω of $\mathbb{R}^n$ and a vector-valued map $J : \Omega \subset \mathbb{R}^n \to \mathbb{R}^p$, $J = (J_1, \ldots, J_p)$ with $J_i : \Omega \subset \mathbb{R}^n \to \mathbb{R}$, any finite-dimensional MOP problem can be written:

$$\min_{x \in \Omega} \; J(x). \tag{1}$$

In this paper, we assume that an ordering on $\mathbb{R}^p$ is induced by the Pareto cone $\mathbb{R}^p_+$. A point $x \in \Omega$ is said to be Pareto optimal or efficient if it is feasible and, for any possible $x' \in X$, $J(x) \leq_{\mathbb{R}^p_+} J(x')$ implies $J(x) = J(x')$.

Equivalently, a point $x \in \Omega$ is said to be Pareto efficient if $(J(x) - \mathbb{R}^p_+) \cap J(\Omega) = \{J(x)\}$. We denote by $\mathbf{Eff}(J)$ the set of efficient points for function J. We say, instead, that $x \in \Omega$ is weakly Pareto efficient when $(J(x) - \operatorname{int} \mathbb{R}^p_+) \cap J(\Omega) = \emptyset$. We denote by $\mathrm{WEff}(J)$ the set of weakly efficient points for function J. We mention also that the point x is properly Pareto efficient (with respect to C) when there exists a cone C with $\mathbb{R}^p_+ \subseteq \operatorname{int} C$ such that x is Pareto efficient with respect to the cone C, i.e.

$$(J(x) - C) \cap J(\Omega) = \{J(x)\} \tag{2}$$

In the following $\mathrm{PEff}_C(J)$ denotes the set of Pareto properly efficient points. Obviously, every properly Pareto efficient point is also Pareto efficient. For a deeper exposition of the notions of Pareto efficiency, one can see Sawaragi et al. (1985). Scalarization techniques allow reducing a MOP problem to a single criterion one. Linear scalarization is the most classical scalarization approach. In this context, a MOP model is reduced to a single criterion one by summing up all criteria with different weights, which gives the relative importance of each criterion for the DM. Hence, a scalarized version of a MOP model is given by:

$$\min_{x \in \Omega} \sum_{i=1}^{p} \beta_i J_i(x), \tag{3}$$

where $\beta = (\beta_1, \ldots, \beta_p)$ is a vector taking values in $\mathbb{R}^p_+$. The next result is well-known and gives relations between solutions of the scalarized problem (3) and solutions of the vector problem (1).

Proposition 1. (*see e.g. Sawaragi et al. (1985)*) *The following statements hold true:*

i) *If $\beta \in \mathbb{R}^p_+$, then every solution of problem (3) is weakly Pareto efficient for problem (1). If functions J_i, $i = 1, \ldots, p$ are convex, then the converse holds true, i.e. for every weakly Pareto efficient solution x of problem (1) there exists a vector $\beta \in \mathbb{R}^p_+$ such that x is a solution of problem (3).*

ii) *If $\beta \in \operatorname{int} \mathbb{R}^p_+$ then every solution of problem (3) is properly Pareto efficient for problem (1) and hence a Pareto efficient solution. If functions J_i, $i = 1, \ldots, p$ are convex, then the converse holds true, i.e. for every properly Pareto efficient solution x of problem (1) there exists a vector $\beta \in \operatorname{int} \mathbb{R}^p_+$ such that x is a solution of problem (1).*

In the literature, one can find different scalarization approaches that can also be applied to non-convex problems. Scalarization techniques can also be considered for problems in which $\mathbb{R}^p$ is ordered by a general cone K, different from the Pareto one. In this case, linear scalarization relies on the elements of the dual cone.

3. Learning with Labeled Data: A Vector-Valued Formulation

In supervised machine learning, machine training is fundamental and measures how well a trained ML model will perform. In the training phase, which is essential for future predictions, one wants to avoid the problems of overfitting and underfitting. A model is said to be well-fitted when it produces accurate outcomes, something between underfitting and overfitting (see Alpaydin (2014); Mak and Chien (2020); Chase and Freitag (2019)). During the training phase, a supervised ML algorithm is run on data for which the target output, known as "labeled" data, is known. The

training process is based on the minimization of an objective function that models the data-fitting error over a set of unknown parameters that define the model accuracy. Over time, as the algorithm learns, the data-fitting error on the training data decreases (Blum et al., 2020; Rao, 2013; Flach, 2012).

Machine training from data consists of finding the optimal model parameters to describe the data. The notion of "fitting" provides a measure of how well a model generalizes from given data.

3.1. Model formulation

Most of the data-fitting techniques in an abstract formulation, can be summarized as follows. Let (X, d^X) and (Y, d^Y) two metric spaces, $\Lambda \subset \mathbb{R}^n$, a compact set of parameters. Consider a set of input vectors x_i and labels y_i, $i = 1, \ldots, N$, a black box function $f : X \times \Lambda \to Y$ and the following data-fitting/minimization problem:

$$\min_{\lambda \in \Lambda} \mathbf{DFE}(\lambda) := (d^Y(f(x_1, \lambda), y_1), d^Y(f(x_2, \lambda), y_2), ..., d^Y(f(x_N, \lambda), y_N)) \tag{4}$$

The folllowing properties of function $\mathbf{DFE}(\lambda)$ are immediate:

- $\mathbf{DFE}(\lambda) : \Lambda \to \mathbb{R}^N_+$
- if the function $f(x, \cdot)$ is continuous, then **DFE** is continuous over Λ and, therefore, **DFE** has at least one global Pareto efficient solution
- if there exists $\lambda^* \in \Lambda$ such that $\mathbf{DFE}(\lambda^*) = 0$ then λ^* is an ideal - and then efficient - point (In this case $f(x_i, \lambda^*) = y_i$ and this corresponds to the ideal case in which $f(\cdot, \lambda^*)$ maps exactly x_i into y_i.)

As one can see from its definition, the data-fitting term measures the distance between the empirical values y_i and the theoretical values $f(x_i, \lambda)$ obtained by the black box function if a specific value of λ is plugged into it. Therefore the training process is reduced to the minimization of the vector-valued function $\mathbf{DFE}(\lambda)$ over the parameters' space Λ. The function $\mathbf{DFE}$ can exhibit different mathematical properties that depend on the specific functional form of f and the definition of d^Y.

As explained in the previous section, one technique to simplify the complexity of a vector-valued problem and reduce it to a scalar one consists in taking its scalarization using weights. If we denote by $\beta_i \geq 0$, $i = 1...N$, a set of weights, and we scalarize the problem as follows:

$$\min_{\lambda \in \Lambda} \beta \cdot \mathbf{DFE}(\lambda) := \sum_{i=1}^{N} \beta_i d^Y(f(x_i, \lambda), y_i) \tag{5}$$

where $\beta = (\beta_1., ..., \beta_N)$, then Eq. (5) and Eq. (4) are related to each other via the results presented in the previous section.

The following examples show how one can obtain classical regression models by specifying the form of d^Y and f and by means of a linear scalarization approach.

Example 1. *Let us suppose that* $f(x, \lambda) = \lambda \cdot x$, $d^Y(f(x_i, \lambda), y_i) = (\lambda \cdot x_i - y_i)^2$, *and scalarization coefficients are* $\beta_i = \frac{1}{N}$, $i = 1...N$. *Then the scalarization of the above model* (4) *takes the form:*

$$\min_{\lambda \in \Lambda} \beta \cdot \mathbf{DFE}(\lambda) := \frac{1}{N} \sum_{i=1}^{N} (\lambda \cdot x_i - y_i)^2 \tag{6}$$

which coincides with the mean squared error.

Example 2. *Suppose that* $y_i \in \{-1, 1\}$ *and* $d^Y(f(x_i, \lambda), y_i) = \phi(f(x_i, \lambda)y_i)$ *where* $\phi(u) = \ln(1 + e^{-u})$ *and* $\beta_i = \frac{1}{N}$. *Then the scalarization of the above model* (4) *takes the form*

$$\min_{\lambda \in \Lambda} \beta \cdot \mathbf{DFE}(\lambda) := \frac{1}{N} \sum_{i=1}^{N} \ln(1 + e^{-f(x_i, \lambda)y_i}) \tag{7}$$

which coincides with the logistic regression model.

Example 3. *Suppose that* $y_i \in \{0, 1\}$ *then*

$$d^Y(f(x_i, \lambda), y_i) = -\sum_{i=1}^{N} [y_i \log(f(x_i, \lambda)) + (1 - y_i) \log(1 - f(x_i, \lambda))]$$

and scalarization coefficients are $\beta_i = \frac{1}{N}$, $i = 1...N$. *Then the above problem* (4) *takes the form:*

$$\min_{lambda \in \Lambda} \beta \cdot \mathbf{DFE}(\lambda) := -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(f(x_i, \lambda)) + (1 - y_i) \log(1 - f(x_i, \lambda))] \tag{8}$$

which coincides with the Binary Cross Entropy loss with reduction.

3.2. Stability results

It is well known that data can be subject to errors due to limited observability, noisy measurements, computational implementations, and prediction errors. Hence, it is worth studying the stability of the function **DFE** and of optimal solutions to the problem (4) with respect to data set perturbations. The following proposition states a stability result for the values of the function **DFE** with respect to perturbation of the label set.

Proposition 2. *Let $\{(x_i, y_i)\}$ and $\{(x_i, \tilde{y}_i)\}$ be two data sets with the same numerosity $N \in \mathbb{N}$, and let $\mathbf{DFE}(\lambda)$ and $\tilde{\mathbf{DFE}}(\lambda)$ be the two corresponding fitting functions. Then*

$$\|\mathbf{DFE}(\lambda) - \tilde{\mathbf{DFE}}(\lambda)\|_2 \leq \sqrt{\sum_{i=1}^{N} d^Y(y_i, \tilde{y}_i)^2} \tag{9}$$

The following result, instead, provides a condition for the stability of the values of the function **DFE** with respect to perturbation of the input data.

Proposition 3. *Let $\{(x_i, y_i)\}$ and $\{(\tilde{x}_i, y_i)\}$ be two data sets with the same numerosity $N \in \mathbb{N}$, and let $\mathbf{DFE}(\lambda)$ and $\tilde{\mathbf{DFE}}(\lambda)$ be the two corresponding fitting functions. Let us suppose that $f(x, \lambda)$ is Lipschitz with respect to x, that is there exist K such that $d^Y(f(a, \lambda), f(b, \lambda)) \leq K d^X(a, b)$ for any $a, b \in X$ and $\lambda \in \Lambda$. Then*

$$\|\mathbf{DFE}(\lambda) - \tilde{\mathbf{DFE}}(\lambda)\|_2 \leq K \sqrt{\sum_{i=1}^{N} d^X(x_i, \tilde{x}_i)^2} \tag{10}$$

The next result, stated in terms of convergence of optimal solutions, concerns the stability of weakly efficient solutions and properly efficient solutions of problem (4) with respect to perturbations of both the input and the label data.

Proposition 4. *Let (x_i^n, y_i^n) be sequences in $X \times Y$ converging to (x_i, y_i) in the $d_{X\times Y} = d_X + d_Y$ metric such that $f(x_i^n, \lambda)$ converges to $f(x_i, \lambda)$, uniformly with respect to $\lambda \in \Lambda$, $i = 1, \ldots, N$. Assume $f(x, \cdot)$ is continuous and let*

$$\mathbf{DFE}_n(\lambda) := (d^Y(f(x_1^n, \lambda), y_1^n), d^Y(f(x_2^n, \lambda), y_2^n), ..., d^Y(f(x_N^n, \lambda), y_N^n)) \tag{11}$$

i) *Let* $\lambda_n \in \mathrm{WEff}(\mathbf{DFE}_n)$. *Then there exists a subsequence* λ_{n_k} *converging to* $\bar{\lambda} \in \Lambda$ *such that* $\bar{\lambda} \in \mathrm{WEff}(\mathbf{DFE})$.

ii) *Let* $\lambda_n \in \mathrm{PEff}_C(\mathbf{DFE}_n)$, *with* $\mathbb{R}^p_+ \subseteq \mathrm{int}\, C$. *Then there exists a subsequence* λ_{n_k} *converging to* $\bar{\lambda} \in \Lambda$ *such that* $\bar{\lambda} \in \mathrm{Eff}(\mathbf{DFE})$.

The previous stability result is stated in terms of convergence of weakly efficient and properly efficient solutions. Often, for computational reasons, it is worth having an estimation of the distance between optimal solutions of a problem (4) with unperturbed and perturbed data in order to give a bound to the error that can be made computing solutions of problem (4) with imprecise data. This is the aim of Proposition 5 below. In order to state this result, we need to introduce some preliminary concepts.
For two subsets of Λ, A and C, we set

$$e(A, C) = \sup_{a \in A} d(a, C) \tag{12}$$

with $d(a, C) = \inf_{c \in C} \|a - c\|$. In the following, for simplicity sake, let $z_i = (x_i, y_i) \in Z = X \times Y$, $z = (z_1, \ldots, z_N) \in Z^N$, $z^0 = (z_1^0, \ldots, z_N^0)$. We assume Z is a metric space with distance $d^Z = d^X + d^Y$.
Let $g_i(\lambda, z_i) = d_i(f(x_i, \lambda), y_i)$, $i = 1, \ldots, N$ and

$$g(\lambda) = (g_1(\lambda, z_1), \ldots, g_N(\lambda), z_N) = \mathbf{DFE}(\lambda)$$

Minimizing g clearly means minimizing $\mathbf{DFE}(\lambda)$, with data given by the vector z. We denote by $\mathrm{Eff}_z(\mathbf{DFE})$ the set of efficient solutions with data set given by z.

Definition 1. (*see e.g. Li and Xu (2010)*) *Let $f : \Lambda \to \mathbb{R}$. We say that $\lambda_0 \in \Lambda$ is an isolated minimizer of order $\alpha > 0$ and constant $h > 0$ when for every $\lambda \in \Lambda$ it holds*

$$f(\lambda) - f(\lambda_0) \geq h\|\lambda - \lambda_0\|^\alpha \tag{13}$$

We say that $\lambda_0 \in \Lambda$ is a local isolated minimizer of order $\alpha > 0$ and constant $h > 0$ when (13) holds for λ in a neighborhood of λ_0.

Let $\beta_i \in (0, 1]$, $i = 1, \ldots, N$ with $\sum_{i=1}^N \beta_i = 1$ and consider function $l(\lambda, z) = \sum_{i=1}^N \beta_i g_i(\lambda, z_i)$. Denote by S_z^l the set minimizers of $l(\cdot, z)$ over Λ. It is well known that $S_z^l \subseteq \mathrm{Eff}_z(\mathbf{DFE})$ where $\mathrm{Eff}_z(\mathbf{DFE})$ denotes the set of efficient points of **DFE** with data set given by z (see Proposition (1)).

Proposition 5. *Assume that*

i) for some choice of scalars $\beta_i \in (0, 1]$, $i = 1, \ldots, N$ with $\sum_{i=1}^N \beta_i = 1$ there exists a point $\lambda(z^0) \in \Lambda$ that is an isolated minimizer of order $\alpha > 0$ and constant $h > 0$ for $l(\cdot, z^0)$.

ii) for any $\lambda \in \Lambda$, each $g_i(\lambda, \cdot)$ is Hölder of order $\delta > 0$ on Z with constant $m > 0$, i.e. for any $z_i^1,\ z_i^2 \in Z$ it holds

$$|g_i(\lambda, z_i^1) - g_i(\lambda, z_i^2)| \leq m d^Z(z_i^1, z_i^2)^\delta \tag{14}$$

Then it holds

$$e(S_z^l, S_{z^0}^l) \leq \left(\frac{2m}{h}\right)^{1/\alpha} \left(\sum_{i=1}^N d^Z(z_i, z_i^0)^\delta\right)^{1/\alpha} \tag{15}$$

Consequently, there exists $\lambda(z) \in \mathrm{Eff}_z(\mathbf{DFE})$ such that

$$d(\lambda(z), \mathrm{Eff}_{z^0}(\mathbf{DFE})) \leq \left(\frac{2m}{h}\right)^{1/\alpha} \left(\sum_{i=1}^N d^Z(z_i, z_i^0)^\delta\right)^{1/\alpha}. \tag{16}$$

Proposition (5) admits a local version presented in the following result.

Proposition 6. *Assume that*

i) for some choice of scalars $\beta_i \in (0,1]$, $i = 1, \ldots, N$ with $\sum_{i=1}^{N} \beta_i = 1$ there exists a point $\lambda(z^0) \in \Lambda$ that is a local isolated minimizer of order $\alpha > 0$ and constant $h > 0$ for $l(\cdot, z^0)$.

ii) for any $\lambda \in \Lambda$, each $g_i(\lambda, \cdot)$ is Hölder of order $\delta > 0$ on Z with constant $m > 0$.

iii) for $z \to z^0$, $g_i(\lambda, z) \to g_i(\lambda, z^0)$ uniformly with respect to λ in a neighborhood of $\lambda(z^0)$.

Then there exists a neighborhood U of z^0 such that for every $z \in U$ one can find $\lambda(z) \in \mathrm{Eff}_z(\mathbf{DFE})$ such that

$$d(\lambda(z), \mathrm{Eff}_{z^0}(\mathbf{DFE})) \leq \left(\frac{2m}{h}\right)^{1/\alpha} \left(\sum_{i=1}^{N} d^Z(z_i, z_i^0)^\delta\right)^{1/\alpha}. \tag{17}$$

4. Learning with multiple data sets

Learning from multiple distributed data sets has many advantages, such as improved generalizability, lower sensitivity to overfitting, and increased robustness. For instance, it is possible to take advantage of the redundancy of the information and improve accuracy by combining multiple complementary data sources. It allows us to avoid bias toward a specific data set. It also allows us to use the data from the different available sources at different times. Such a type of distributed learning also carries a computational advantage. Several samples scattered in different nodes allow scalability

without increasing the sole worker's computational burden. In addition, it is also more cost-effective and easier to manage.

We now extend the previous approach to the case of multiple data sets, Γ_1, Γ_2, ... Γ_M, each of them with cardinality s_i. This is an extended scenario in which we want to learn simultaneously from different data sets by balancing the information extracted from each of them. This approach also allows for reducing the bias in the training process due to the choice of a particular set of samples. It is pretty straightforward to extend to this context the stability results proved in the previous sections. The training process in this context reads as

$$\min_{\lambda \in \Lambda} \mathbf{DFE}(\lambda) := (\mathbf{DFE^1}(\lambda), ..., \mathbf{DFE^M}(\lambda)) \tag{18}$$

where $\mathbf{DFE^1} : \Lambda \to \mathbb{R}^{s_1}$, ... $\mathbf{DFE^M}(\lambda) : \Lambda \to \mathbb{R}^{s_M}$ are the data fitting terms defined on each data set Γ_i, $i = 1...M$.

One possible way to solve the above model is to rely on the linear scalarization approach. If we denote by $\beta_i \in \mathbb{R}^{s_i}_+$, $i = 1, \ldots, M$, the weights associated with each criterion, the scalarized model reads as

$$\min_{\lambda \in \Lambda} \beta_1 \cdot \mathbf{DFE^1}(\lambda) + \cdots + \beta_M \cdot \mathbf{DFE^M}(\lambda) \tag{19}$$

As demonstrated in the previous section, it is possible to characterize the efficient solutions of the above Eq. (18) by varying the scalarization weights in Eq. (19). The same argument applies to the stability results presented in the previous section that can be extended to this case as well.

5. Numerical experiments

The following sections present some computational experiments regarding digit recognition with different neural network architectures. As it is classical in image recognition, the input layer corresponds to a vectorized form of the training image: a greyscale image with $n \times m$ pixels can be digitalized into a $n \times m$ matrix and therefore transformed into a nm dimensional vector. The training set we consider in our analysis is a subset Γ of the MNIST data set. The MNIST data set, whose sample is presented in Figure 4, has been extensively employed to test several families of ML classification algorithms [1]. In the considered data set, a digit is a 20x20 pixels image. The associated matrix contains numbers between 0 and 255, proportional to the pixel's brightness. In order to proceed with our analysis, the data set Γ has been split into three subsets, each of them with the same amount of data, Γ_1, Γ_2, and Γ_3 ($s_1 = s_2 = s_3$). Γ_1 is the unaltered third of the original data, while the data in Γ_2 and Γ_3 have been modified by adding a zero-mean Gaussian noise with standard deviations σ_2 and σ_3, respectively.

The above vector problem can be scalarized to be presented in the following form:

$$\min_{\lambda \in \Lambda} \beta_1 \mathbf{DFE}^1_{s_1}(\lambda) + \beta_2 \mathbf{DFE}^2_{s_2}(\lambda) + \beta_3 \mathbf{DFE}^3_{s_3}(\lambda) \tag{20}$$

where β_i is the weight associated with the i-th term, while $\mathbf{DFE}^i_{s_i}$ refers to the data fitting function defined using the data set Γ_i. Here, with some abuse of notation, we still indicate the scalarized data fitting terms over each data set with the expressions $\mathbf{DFE}^i_{s_i}(\lambda)$.

[1] Available at http://yann.lecun.com/exdb/mnist/

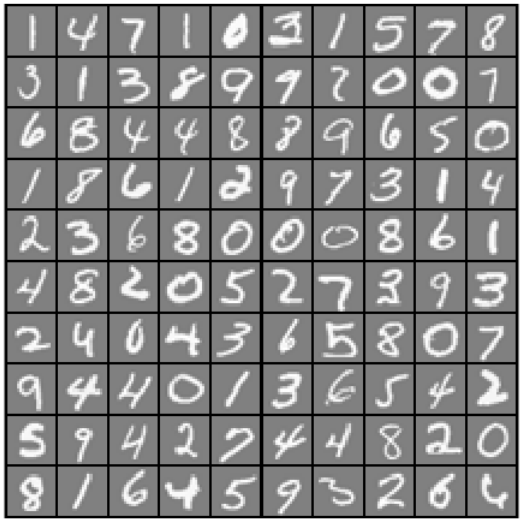

Figure 4: Handwritten digits from the MNIST data set

To test our approach, we select $\beta = \frac{1}{3}$, which corresponds to the case of no data set splitting, and we perturb each architecture by an ϵ parameter. The resulting loss function reads as:

$$\min_{\lambda \in \Lambda} \left(\frac{1}{3} + \epsilon\right) \mathbf{DFE}^1_{s_1}(\lambda) + \left(\frac{1}{3} - \frac{\epsilon}{2}\right) \mathbf{DFE}^2_{s_2}(\lambda) + \left(\frac{1}{3} - \frac{\epsilon}{2}\right) \mathbf{DFE}^3_{s_3}(\lambda) \quad (21)$$

when $\epsilon = 0$ we obtain the basic formulation. All ANN structures were optimized using a stochastic gradient descent approach with the same learning rate and pseudo-randomly generated weights to compare the results.

5.1. The case of the Multilayer Perceptron

In this section we perform a numerical experiment by employing the MLP architecture. In this context the fitting function $\mathbf{DFE}^i_{s_i}$ reads as:

$$\mathbf{DFE}^{i}_{s_i}(\lambda) = \frac{1}{s_i}\sum_{j=0}^{s_i}\sum_{k=1}^{K}[y_j^{(k)}\ \log((h_\lambda(x_j))_k) + (1-y_j^{(k)})\ \log(1-(h_\lambda(x_j))_k)] \tag{22}$$

where s_i is the cardinality of Γ_i. The hypothesis function $(h_\lambda(x_j))_k$ follows the idea of forward propagation: each unit in the second and the third layer evaluates the linear combination $(\lambda^T x_j)$ of its preceding signals through a sigmoid function, detailed in :

$$h_\lambda(x_j) = \frac{1}{1+e^{\lambda^T x_j}}. \tag{23}$$

The index $k = 1, \ldots, K$ represents the k^{th} label. The matrices $\lambda^{(1)}$ and $\lambda^{(2)}$ incorporate the forward propagation from layer 1 to layer 2 and from layer 2 to layer 3, respectively.

In order to investigate small variations of this configuration we perturb the weight β_i with a small ϵ. The specific architecture we use consists of an MLP with an input layer, a hidden layer, and an output layer. Given that the data set contains images with 20×20 pixels, an input layer with $N = 400$ nodes will be needed. The hidden and the output layer have $H = 25$ and $K = 10$ nodes, respectively.

Figure 5 shows how the accuracy changes as a function of the perturbation parameter ϵ. By varying ϵ uniformly over the interval $[0.001, 0.01]$, the accuracy levels are compared with respect to the benchmark case in which $\epsilon = 0$. Once again, this experiment confirms the previous results on accuracy improvement.

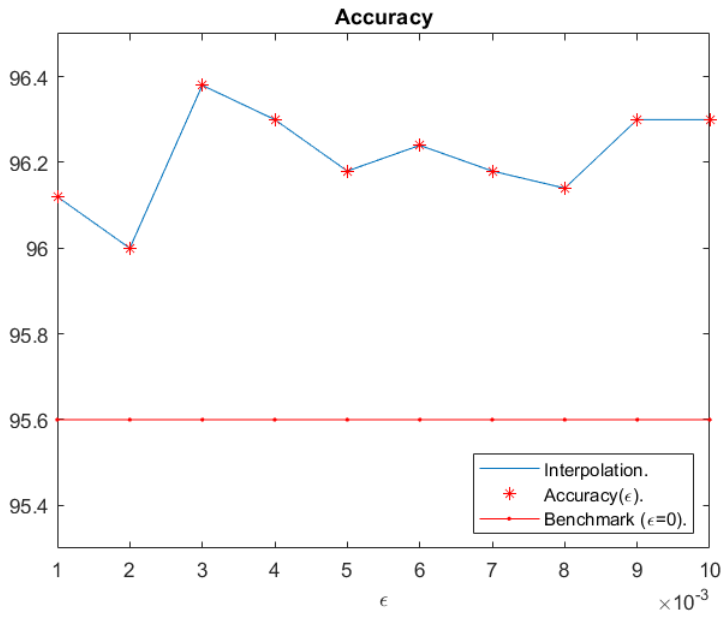

Figure 5: Accuracy on the MNIST data set of the Multilayer Percepron with different values of ϵ (blue line). The non-perturbed benchmark case with $= 0$ is in red.

5.2. The case of ResNet

In the second numerical experiment, we consider a more recent ANN architecture, the so-called ResNet, whose configuration is illustrated in Figure 6. The flattened input image passes through the first and second layers with the rectifier activation function, that is:

$$h_\lambda(x_j) = \max\{0, \lambda^T x_j\} \tag{24}$$

The shortcut is placed in the output of layer one and added to the output of layer two. For the numerical experiment to be consistent with the ones in the previous section, we employ the same cost function as in Eq. (22).

Figure 7 shows the accuracy as the perturbation parameter ϵ changes. The results corroborate the theoretical findings of the previous sections.

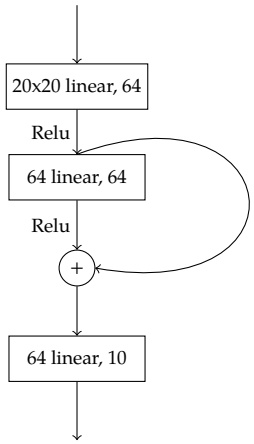

Figure 6: Residual Network architecture employed in the numerical experiment

5.3. The case of Convolutional Neural Networks

For the last numerical example, we consider a Convolutional Neural Network architecture. In our experiment, we use the architecture illustrated in Figure 8. Figure 9 shows the change in terms of accuracy with the epsilon perturbation on the betas.

6. Testing on the validation set

The results of the previous numerical experiments have been analyzed only from the perspective of the training process. However, in the general empirical setting, the performance is taken from a set, or multiple sets, of data that is not used in training. This set of data is called the validation set

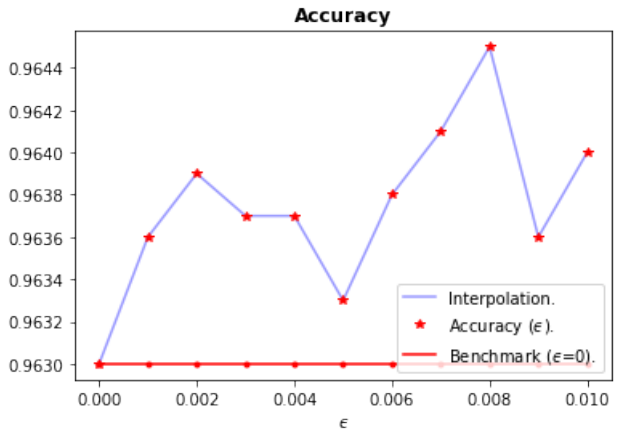

Figure 7: Accuracy on the MNIST data set of the Residual Network with different values of ϵ perturbation (blue line), against the non perturbed benchmark (red line)

and serves a different role from the test set. The necessity of a validation set is twofold. First, the model performance on a validation set is a rough proxy of the model's generalization error (Friedman, 2017). Second, the validation set is also used to check the goodness of possible hyperparameter configurations (Goodfellow et al., 2016). In our case, ϵ should be treated as a hyperparameter. Therefore our empirical setting uses the results derived from the best neural network architecture out of the three used in the numerical experiments. Then we select the model with the value of ϵ achieving the best performance in the training set and compare it with the model's accuracy in the validation set with respect to the benchmark in which $\epsilon = 0$.

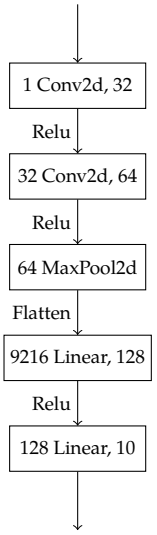

Figure 8: Convolutional Neural Network architecture employed in the numerical experiment

7. Architecture design with sparsity

In the literature, several authors have used the notion of sparsity to reduce the complexity of a model by considering only those parameters whose values have a major impact on the solution. In fact, the addition of the sparsity term allows finding solutions that are "simpler" to be used or decision rules that are "easier" to be implemented. In general, we say that a

Table 1: Accuracy on the training and validation sets, with $\epsilon = 0.0$ and $\epsilon = 0.01$

ϵ	**Training set**	**Validation set**
0.0	0.9912	0.9668
0.01	0.9971	0.9673

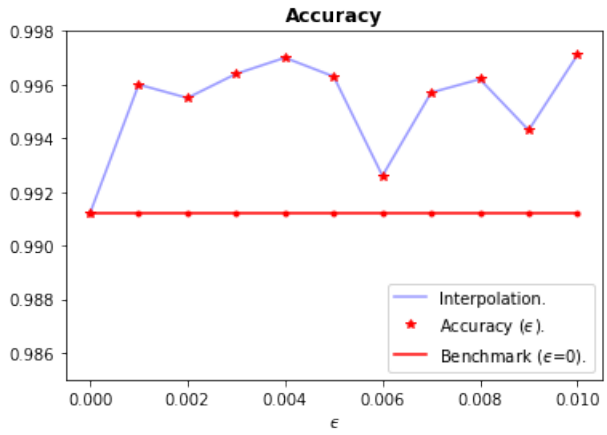

Figure 9: Accuracy on the MNIST data set of the Convolutional Neural Network with different values of ϵ perturbation (blue line), against the non perturbed benchmark (red line).

real vector x in R^n is sparse when most of the entries of x vanish. This is equivalent to saying that the 0-pseudonorm, or counting 'norm', defined as $\|x\|_0 = \#\{i : x_i \neq 0\}$, must be as small as possible. The 0-pseudonorm is a strict sparsity measure, and most optimization problems based on it are combinatorial in nature and hence in general, NP-hard. In this section, we consider the impact of sparsity on performance in order to further analyze the ramifications of our findings. Consequently, we enhanced the data fitting loss function by incorporating an L-2 norm on the CNN's parameters, resulting in the loss function shown below:

$$\min_{\lambda \in \Lambda} \left(\frac{1}{3} + \epsilon\right) \mathbf{DFE}^1_{s_1}(\lambda) + \left(\frac{1}{3} - \frac{\epsilon}{2}\right) \mathbf{DFE}^2_{s_2}(\lambda) + \left(\frac{1}{3} - \frac{\epsilon}{2}\right) \mathbf{DFE}^3_{s_3}(\lambda) + \beta_4 \|\lambda\|_0$$

The CNN architecture was used for this investigation since it was the

most effective one on the holdout sample. In order to implement sparsity in our architecture, we reduced the importance given to each data set during training. To be more specific, we fixed the level of sparsity so that $\beta_4 = 0.1$ was achieved. In Figure 10, we can see that the level of sparsity has an impact on the performance of the model. Indeed the model for lower values of ϵ is in line with the benchmark, i.e., the case with no adversarial learning and no sparsity. This result is consistent with the overall tradeoff between sparsity and training performance that has been demonstrated in many works (Hoefler et al., 2021). In essence, the gain given by the adversarial learning is captured by the sparsity requirement. What is noteworthy is the rise in such performance as ϵ increases, indicating a favorable impact of employing a multicriteria approach throughout the learning process, as demonstrated in the graph. However, the effect of sparsity is still significant in negatively impacting the performance, which is nowhere near the case without sparsity. This is clear when the value of ϵ is relatively high. In this case, sparsity eliminates the benefit of increased performance, making the model's performance plateauing.

8. Policy implications and conclusions

Nowadays any decision-making process relies more and more on precise and accurate predictions. Having more accurate predictions than market competitors has a huge impact on any strategic decision and, ultimately, on performances.

With this aim in mind, in recent years, public organizations and companies have invested a lot of effort and financial resources in hiring data

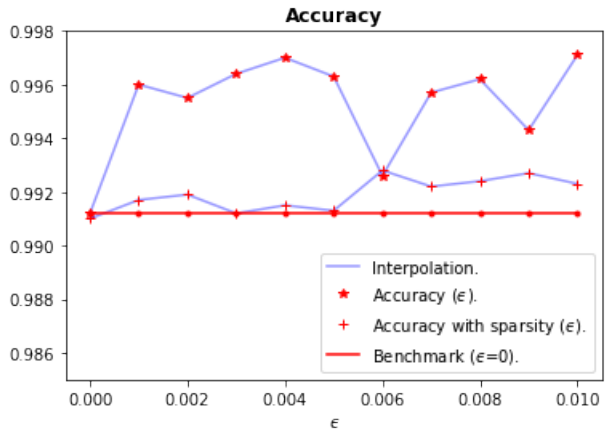

Figure 10: Accuracy on the MNIST data set of the Convolutional Neural Network with different values of ϵ perturbation (blue line), against the non perturbed benchmark (red line), with and without L-2 norm sparsity.

scientists and machine learning engineers. This aspect, combined with the access to a huge amount of freely available data as well as the development of more and more advanced machine learning algorithms, have increased the likelihood of estimating future scenarios and events.

Recent literature has demonstrated that there is still room for improving existing techniques, for instance, by bringing experiences and approaches developed in other contexts into the machine learning literature.

This is the case of this paper, in which we propose a new approach which integrates machine learning with multicriteria decision-making techniques. The modified machine learning model presented in this paper has been formulated in an abstract setting as a vector-valued optimization problem in which each criterion measures the distance between the output value associated with an input value and its label. Stability results for this

problem, which demonstrate how the modified algorithm performs under perturbations, have been proved as well,

We have then considered the case of multiple data sets. Learning from multiple distributed data sets has many advantages, such as improved generalizability, lower sensitivity to overfitting, and increased robustness. We have demonstrated how to use our proposed multicriteria approach in the case of multiple data sets in which the training can be split over each data set and run parallelly and independently.

We have applied this model to the case of Multilayer Perceptron, Deep Residual Network, and Convolutional Neural Network via scalarization approach. Our numerical simulation shows that the adoption of multicriteria techniques can not only provide a general framework to contextualize the machine training with multiple data sets, but it can also provide better accuracy and performance than classical deep learning algorithms if the right choice of the weights is selected.

Future works should include the implementation of more advanced multicriteria techniques as well as the use of dynamic approaches.

References

Ahmed, A. S., Abood, M. S., and Hamdi, M. M. (2021). Advancement of deep learning in big data and distributed systems. In *2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pages 1–7. IEEE. *Cited on page 3*

Alpaydin, E. (2014). *Introduction to Machine Learning, third edition*. Adaptive Computation and Machine Learning series. MIT Press. *Cited on page 11*

Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V. (2020). How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR. *Cited on page 3*

Barber, D. (2012). *Machine learning concepts*, page 305–321. Cambridge University Press. *Cited on page 4*

Berenguer, M. I., Kunze, H., La Torre, D., and Galán, M. R. (2016). Galerkin method for constrained variational equations and a collage-based approach to related inverse problems. *Journal of Computational and Applied Mathematics*, 292:67–75. *Cited on page 4*

Billones, C. D., Demetria, O. J. L. D., Hostallero, D. E. D., and Naval, P. C. (2016). Demnet: a convolutional neural network for the detection of alzheimer's disease and mild cognitive impairment. In *2016 IEEE region 10 conference (TENCON)*, pages 3724–3727. IEEE. *Cited on page 8*

Blum, A., Hopcroft, J., and Kannan, R. (2020). *Machine Learning*, page 109–158. Cambridge University Press. *Cited on page 12*

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečnỳ, J., Mazzocchi, S., McMahan, B., et al. (2019). Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems*, 1:374–388. *Cited on page 3*

Bottou, L., Chapelle, O., DeCoste, D., and Weston, J. (2007). Scaling Learning Algorithms toward AI. In *Large-Scale Kernel Machines*, pages 321–359. MIT Press. *Cited on page 2*

Bryson, B., Kunze, H., La Torre, D., and Liuzzi, D. (2021). A generalized multiple criteria data-fitting model with sparsity and entropy with application to growth forecasting. *IEEE Transactions on Engineering Management*. *Cited on page 4*

Canziani, A., Paszke, A., and Culurciello, E. (2017). An Analysis of Deep Neural Network Models for Practical Applications. *Cited on page 8*

Chase, H. and Freitag, J. (2019). Model theory and machine learning. *The Bulletin of Symbolic Logic*, 25(3):319–332. *Cited on page 11*

Chen, F., Chen, N., Mao, H., and Hu, H. (2018). Assessing four neural networks on handwritten digit recognition dataset (mnist). *arXiv preprint arXiv:1811.08278*. *Cited on page 8*

de la Higuera, C. (2010). *Artificial intelligence techniques*, page 281–299. Cambridge University Press. *Cited on page 2*

Deisenroth, M. P., Faisal, A. A., and Ong, C. S. (2020). *Mathematics for Machine Learning*. Cambridge University Press. *Cited on page 4*

Desai, M. and Shah, M. (2021). An anatomization on breast cancer detection and diagnosis employing multi-layer perceptron neural network (mlp) and convolutional neural network (cnn). *Clinical eHealth*, 4:1–11. *Cited on page 8*

Flach, P. (2012). *The ingredients of machine learning*, page 13–48. Cambridge University Press. *Cited on page 12*

Friedman, J. H. (2017). *The elements of statistical learning: Data mining, inference, and prediction*. springer open. *Cited on page 25*

Fukushima, K. and Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and Cooperation in Neural Nets*, pages 267–285. Springer. *Cited on page 7*

Goel, A. K. and Davies, J. (2011). *Artificial Intelligence*, page 468–482. Cambridge Handbooks in Psychology. Cambridge University Press. *Cited on page 2*

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`. *2 citations, pages 2 and 25*

Hafiz, F., Broekaert, J., La Torre, D., and Swain, A. (2021). A multi-criteria approach to evolve sparse neural architectures for stock market forecasting. *arXiv preprint arXiv:2111.08060*. *Cited on page 4*

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*. *Cited on page 8*

Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., and Peste, A. (2021). Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124. *Cited on page 28*

Jiang, H. (2022). *Machine Learning Fundamentals: A Concise Introduction*. Cambridge University Press. *Cited on page 4*

Khan, S., Islam, N., Jan, Z., Din, I. U., and Rodrigues, J. J. C. (2019). A novel deep learning based framework for the detection and classification of breast cancer using transfer learning. *Pattern Recognition Letters*, 125:1–6. *Cited on page 8*

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. (2016). Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*. *Cited on page 3*

Kunze, H. and La Torre, D. (2020). Solving inverse problems for steady-state equations using a multiple criteria model with collage distance, entropy, and sparsity. *Annals of Operations Research*, pages 1–15. *Cited on page 4*

La Torre, D., Colapinto, C., Durosini, I., and Triberti, S. (2021). Team formation for human-artificial intelligence collaboration in the workplace: a goal programming model to foster organizational change. *IEEE Transactions on Engineering management*. *Cited on page 4*

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L. (1989). Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2. *Cited on page 7*

Li, S. J. and Xu, S. (2010). Sufficient conditions of isolated minimizers for constrained programming problems. *Numerical Functional Analysis and Optimization*, 31(6):715–727. *Cited on page 17*

Mak, M.-W. and Chien, J.-T. (2020). *Machine Learning Models*, page 36–112. Cambridge University Press. *Cited on page 11*

McClelland, J. L., Rumelhart, D. E., Group, P. R., et al. (1987). *Parallel Distributed Processing, Volume 2: Explorations in the Microstructure of Cognition: Psychological and Biological Models*, volume 2. MIT press. *Cited on page 6*

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133. *Cited on page 5*

Moitra, A. (2018). *Algorithmic Aspects of Machine Learning*. Cambridge University Press. *Cited on page 4*

Nosratabadi, S., Ardabili, S., Lakner, Z., Mako, C., and Mosavi, A. (2021). Prediction of food production using machine learning algorithms of multilayer perceptron and anfis. *Agriculture*, 11(5):408. *Cited on page 7*

Poole, D. L. and Mackworth, A. K. (2017). *Supervised Machine Learning*, page 267–340. Cambridge University Press, 2 edition. *Cited on page 4*

Rao, R. P. N. (2013). *Machine Learning*, page 71–98. Cambridge University Press. *Cited on page 12*

Repetto, M., La Torre, D., and Tariq, M. (2021). Deep learning with multiple data set: A weighted goal programming approach. *arXiv preprint arXiv:2111.13834*. *Cited on page 4*

Ripley, B. D. (1996). *Frontmatter*, pages i–iv. Cambridge University Press. *Cited on page 2*

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408. *Cited on page 5*

Saad, D., editor (1998). *On-Line Learning in Neural Networks*. Publications of the Newton Institute. Cambridge University Press. *Cited on page 6*

Sawaragi, Y., Nakayama, H., and Tanino, T. (1985). *Theory of multiobjective optimization*, volume 176 of *Mathematics in Science and Engineering*. Academic Press, Inc., Orlando, FL. *Cited on page 10*

Schank, R. C. and Towle, B. (2000). *Artificial Intelligence*, page 341–356. Cambridge University Press. *Cited on page 2*

Shah, C. (2020). *A Hands-On Introduction to Data Science*. Cambridge University Press. *Cited on page 4*

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press. *Cited on page 4*

Sharma, S., Aggarwal, A., and Choudhury, T. (2018). Breast cancer detection using machine learning algorithms. In *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*, pages 114–118. IEEE. *Cited on page 6*

Sopelsa Neto, N. F., Stefenon, S. F., Meyer, L. H., Bruns, R., Nied, A., Seman, L. O., Gonzalez, G. V., Leithardt, V. R. Q., and Yow, K.-C. (2021). A study of multilayer perceptron networks applied to classification of ceramic insulators using ultrasound. *Applied Sciences*, 11(4):1592. *Cited on page 7*

Venables, W. N. and Ripley, B. D. (1999). *Modern Applied Statistics with S-PLUS*. Statistics and Computing. Springer-Verlag, 3 edition. *Cited on page 8*

Wang, D. and Barabási, A.-L. (2021). *Artificial Intelligence*, page 231–240. Cambridge University Press. *Cited on page 2*

Wu, Z., Shen, C., and Van Den Hengel, A. (2019). Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 90:119–133. *Cited on page 8*

Yang, M., Nazir, S., Xu, Q., and Ali, S. (2020). Deep learning algorithms and multicriteria decision-making used in big data: a systematic literature review. *Complexity*, 2020. *Cited on page 3*

Zhu, Y., Brettin, T., Xia, F., Partin, A., Shukla, M., Yoo, H., Evrard, Y. A., Doroshow, J. H., and Stevens, R. L. (2021). Converting tabular data into images for deep learning with convolutional neural networks. *Scientific reports*, 11(1):1–11. *Cited on page 8*

9. Technical Appendix

Proof. This is the proof of Proposition 2. By computing, we get:

$$\|\mathbf{DFE}(\lambda) - \tilde{\mathbf{D}}\mathbf{FE}(\lambda)\|_2^2 = \tag{25}$$

$$\|(d^Y(f(x_1,\lambda),y_1),...,d^Y(f(x_N,\lambda),y_N))-(d^Y(f(x_1,\lambda),\tilde{y}_1),...,d^Y(f(x_N,\lambda),\tilde{y}_N))\|_2^2 =$$

$$(d^Y(f(x_1,\lambda),y_1)-d^Y(f(x_1,\lambda),\tilde{y}_1))^2+...+(d^Y(f(x_N,\lambda),y_N)-d^Y(f(x_N,\lambda),\tilde{y}_N))^2 \leq$$

$$d^Y(y_1,\tilde{y}_1)^2 + ... + d^Y(y_N,\lambda y_N)^2$$

where the last inequality follows from the distance property:

$$|d^Y(a,b) - d^Y(b,c)|^2 \leq d^Y(a,c)^2$$

■

Proof. This is the proof of Proposition 3. By computing, we get:

$$\|\mathbf{DFE}(\lambda) - \tilde{\mathbf{D}}\mathbf{FE}(\lambda)\|_2^2 = \tag{26}$$

$$\|(d^Y(f(x_1,\lambda),y_1),...,d^Y(f(x_N,\lambda),y_N))-(d^Y(f(\tilde{x}_1,\lambda),y_1),...,d^Y(f(\tilde{x}_N,\lambda),y_N))\|_2^2 =$$

$$(d^Y(f(x_1,\lambda),y_1)-d^Y(f(\tilde{x}_1,\lambda),y_1))^2+...+(d^Y(f(x_N,\lambda),y_N)-d^Y(f(\tilde{x}_N,\lambda),y_N))^2 \leq$$

$$K^2 d^X(x_1,\tilde{x}_1)^2 + ... + K^2 d^Y(x_N,\tilde{x}_N)^2$$

and now the thesis easily follows. ■

Proof. This is the proof of Proposition 4.

i) Since Λ is compact, λ_n admits a subsequence λ_{n_k} converging to $\bar{\lambda} \in \Lambda$. Since $\lambda_{n_k} \in \mathrm{WEff}(\mathbf{DFE}_{n_k})$, we have

$$\mathbf{DFE}_{n_k}(\lambda) - \mathbf{DFE}_{n_k}(\lambda_{n_k}) \notin -\mathrm{int}\,\mathbb{R}^N,\ \forall \lambda \in \Lambda \tag{27}$$

Further we have

$$\begin{aligned} &\|\mathbf{DFE}_{n_k}(\lambda_{n_k}) - \mathbf{DFE}(\bar{\lambda}\|_2 \leq \\ &\|\mathbf{DFE}(\lambda_{n_k}) - \mathbf{DFE}(\bar{\lambda})\|_2 + \|\mathbf{DFE}_{n_k}(\lambda_{n_k}) - \mathbf{DFE}(\lambda_{n_k})\|_2 \end{aligned} \tag{28}$$

Uniform convergence of $\mathbf{DFE}_{n_k}$ to $\mathbf{DFE}$ implies $\mathbf{DFE}$ is continuous. Hence, from (28) we get $\mathbf{DFE}_{n_k}(\lambda_{n_k}) \to \mathbf{DFE}(\bar{\lambda})$ and (27) implies

$$\mathbf{DFE}(\lambda) - \mathbf{DFE}(\bar{\lambda}) \notin -\mathrm{int}\,\mathbb{R}^N,\ \forall \lambda \in \Lambda \tag{29}$$

i.e. $\bar{\lambda} \in \mathrm{WEff}(\mathbf{DFE})$.

ii) Let $\lambda_n \in \mathrm{PEff}_C(\mathbf{DFE}_n)$ and let λ_{n_k} be a subsequence converging to $\bar{\lambda} \in \Lambda$. Then

$$\mathbf{DFE}_{n_k}(\lambda) - \mathbf{DFE}_{n_k}(\lambda_{n_k}) \notin -C\backslash\{0\},\ \forall \lambda \in \Lambda \tag{30}$$

Passing to the limit we obtain

$$\mathbf{DFE}(\lambda) - \mathbf{DFE}(\bar{\lambda}) \notin -\mathrm{int}\,C,\ \forall \lambda \in \Lambda \tag{31}$$

and hence

$$\mathbf{DFE}(\lambda) - \mathbf{DFE}(\bar{\lambda}) \notin -\mathbb{R}^n_+\backslash\{0\},\ \forall \lambda \in \Lambda \tag{32}$$

■

Proof. This is the proof of Proposition 5. Let $\lambda(z^0) \in S^l_{z^0}$ be an isolated minimizer of order α and constant m for $l(\cdot, z^0)$. Then, for $\lambda(z) \in S^l_z$ it holds

$$l(\lambda(z), z^0) - l(\lambda(z^0), z^0) \geq h\|\lambda(z) - \lambda(z^0)\|^\alpha \tag{33}$$

We have

$$l(\lambda(z^0), z) - l(\lambda(z), z) = l(\lambda(z^0), z^0) - l(\lambda(z), z^0) + w \tag{34}$$

where

$$w = [l(\lambda(z^0), z) - l(\lambda(z^0), z^0)] + [l(\lambda(z), z^0) - l(\lambda(z), z)] \tag{35}$$

We have

$$|w| \leq |l(\lambda(z^0), z) - l(\lambda(z^0), z^0)| + |l(\lambda(z), z^0) - l(\lambda(z), z)| \leq \tag{36}$$

$$\sum_{i=1}^{N} \beta_i |g_i(\lambda(z^0), z_i) - g_i(\lambda(z^0), z_i^0)| + \sum_{i=1}^{N} \beta_i |g_i(\lambda(z), z_i^0) - g_i(\lambda(z), z_i)| \leq \tag{37}$$

$$2m \sum_{i=1}^{N} \beta_i d^Z(z_i, z_i^0)^\delta \leq 2m \sum_{i=1}^{N} d^Z(z_i, z_i^0)^\delta \tag{38}$$

We claim that

$$l(\lambda(z), z^0) - l(\lambda(z^0), z^0) \leq |w| \tag{39}$$

Indeed, suppose to the contrary that

$$l(\lambda(z), z^0) - l(\lambda(z^0), z^0)) - |w| > 0 \tag{40}$$

If $w = 0$, then

$$l(\lambda(z^0), z) - l(\lambda(z), z) < 0 \tag{41}$$

which contradicts $\lambda(z) \in S^l_z$. If $w \neq 0$, then we have

$$l(\lambda(z), z) - l(\lambda(z^0), z) = g(\lambda(z), z^0) - l(\lambda(z^0), z^0) - w > 0 \tag{42}$$

which again contradicts $\lambda(z) \in S_z^l$.

Observe now that we have

$$h\|(\lambda(z) - \lambda(z^0)\|^\alpha \leq l(\lambda(z), z^0) - l(\lambda(z^0), z^0) \tag{43}$$

and therefore

$$h\|\lambda(z) - \lambda(z^0)\|^\alpha \leq l(\lambda(z), z^0) - l(\lambda(z^0), z^0) \leq 2m \sum_{i=1}^{N} d^Z(z_i, z_i^0)^\delta \tag{44}$$

So, it holds

$$\|\lambda(z) - \lambda(z^0)\| \leq \left(\frac{2m}{h}\right)^{1/\alpha} \left(\sum_{i=1}^{N} d^Z(z_i, z_i^0)^\delta\right)^{1/\alpha} \tag{45}$$

which finally implies

$$d(\lambda(z), S_{z^0}^l) \leq \|\lambda(z) - \lambda(z^0)\| \leq \left(\frac{2m}{h}\right)^{1/\alpha} \left(\sum_{i=1}^{N} d^Z(z_i, z_i^0)^\delta\right)^{1/\alpha} \tag{46}$$

Since this holds for any $\lambda(z) \in S_z^l$ finally we have

$$e(S_z^l, S_{z^0}^l) \leq d(\lambda(z), S_{z^0}^l) \leq \left(\frac{2m}{h}\right)^{1/\alpha} \left(\sum_{i=1}^{N} d^Z(z_i, z_i^0)^\delta\right)^{1/\alpha} \tag{47}$$

Since $\lambda(z) \in \mathrm{Eff}_z(DFE)$ we have

$$d(\lambda(z), \mathrm{Eff}_{z^0}(\mathbf{DFE})) \leq \left(\frac{2m}{h}\right)^{1/\alpha} \left(\sum_{i=1}^{N} d^Z(z_i, z_i^0)^\delta\right)^{1/\alpha} \tag{48}$$

which concludes the proof. ■

Proof. The proof of Proposition 6 is similar to that of Proposition 5 and, therefore, it is omitted. ■