Pattern Recognition Letters 177 (2024) 69–74

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Hidden classification layers: Enhancing linear separability between classes in neural networks layers

Andrea Apicella *, Francesco Isgrò, Roberto Prevete

Department of Electrical Engineering and Information Technology, University of Naples Federico II, Naples, Italy
Laboratory of Augmented Reality for Health Monitoring (ARHeMLab), Naples, Italy

ARTICLE INFO

Editor: Laurent Heutte

Keywords:
Neural networks
Hidden layers
Hidden representations
Linearly separable

ABSTRACT

Many Deep Learning approaches are based on variations of standard multi-layer feed-forward neural networks. These are also referred to as deep networks. The basic idea is that each hidden neural layer accomplishes a data transformation which is expected to make the data representation "somewhat more linearly separable" than the previous one to obtain a final data representation which is as linearly separable as possible. However, determining the optimal network parameters for these transformations is a crucial challenge. In this study, we propose a Deep Neural Network architecture (Hidden Classification Layer, HCL) which induces an error function involving the output values of all the network layers. The proposed architecture leads toward solutions where the data representations in the hidden layers exhibit a higher degree of linear separability between classes compared to conventional methods. While similar approaches have been discussed in prior literature, this paper presents a new architecture with a novel error function and conducts an extensive experimental analysis. Furthermore, the architecture can be easily integrated into existing frameworks by simply adding densely connected layers and making a straightforward adjustment to the loss function to account for the output of the added layers. The experiments focus on image classification tasks using four well-established datasets, employing as baselines three widely recognized architectures in the literature. The findings reveal that the proposed approach consistently enhances accuracy on the test sets across all considered cases.

1. Introduction

Nowadays, the success of Deep Learning (DL) approaches in several Machine Learning tasks [1–4] has led to an increase in interest in Multi-Layer Feed-Forward (MLFF) neural networks [5] insofar as a successful class of deep neural networks consists of MLFF networks with more than one hidden layer and possibly some specific architectural choices. In the rest of the paper we will refer to such Deep Neural Networks as DNNs. In a nutshell, DNN networks are computational architectures organized as L consecutive layers or levels of elementary computing units, called *neurons*. The last layer L is the *output* layer, and the remaining layers are usually called *hidden* or *internal* layers.

In a DNN network each hidden layer l performs a non-linear functional map $\Phi_l^{\theta_l}$ from the output $\mathbf{z}_{l-1}$ of the previous layer (and possibly other previous layers) to the output of the layer itself, where θ_l are the weights associated to the connections incoming into the layer l, plus the biases of the layer. By contrast, the output layer may also perform a linear transformation. In other words, the whole computation of a DNN can be viewed as a non-linear parametric functional mapping $\mathbf{y} = M(\mathbf{x};\theta)$ from a d-dimensional space to a c-dimensional space, where d is the number of input variables and $c = m_L$ is the number of neurons in the output layer. The parameters θ are the weights and biases of the network, and $\mathbf{y}$ are the output values of the output layer.

Although from a theoretical point of view the DNNs capability of being universal approximators has been extensively discussed [6–8], together with the inducted feature representation spaces [8,9], it is important to notice that the difficult to effectively find the most suitable θ remains. In particular, when DNNs are applied in the context of classification problems, one has to find the parameters θ_l such that the composition of $L-1$ non-linear transformations $\mathbf{z}_{L-1} = \Phi_{L-1}\Big(\Phi_{L-2}\big(\dots\Phi_1(\mathbf{x})\big)\Big)$ maps each input $\mathbf{x}$ from a *non-linearly* separable space into a *linearly separable* one. In fact, in the context of classification problems, one of the main goals is to find a suitable data representation which allows to obtain a linearly separable classification problem. Plausibly, when a DNN is used, each internal representation $\mathbf{z}_l$ can be expected to make the representations of $\mathbf{x}$ "somewhat more linearly separable" than the previous one $\mathbf{z}_{l-1}$. We underline that the complexity of a classification problem can be measured with respect different aspects, however class

* Corresponding author at: Department of Electrical Engineering and Information Technology, University of Naples Federico II, Naples, Italy.
E-mail addresses: andrea.apicella@unina.it (A. Apicella), fisgro@unina.it (F. Isgrò), rprevete@unina.it (R. Prevete).

https://doi.org/10.1016/j.patrec.2023.11.016
Received 21 March 2023; Received in revised form 6 October 2023; Accepted 13 November 2023
Available online 17 November 2023

separability is a key aspect and different levels of class separability can be quantified [10,11]. In particular, in [12] the Generalized Discrimination Value (GDV) to measure the separability between two dataset is introduced. The GDV is defined as the gap between the mean intra-cluster and the mean inter-cluster distances, computed on a set of labelled data represented in some space. More in detail, the GDV compares in a quantitative way the degree of class separability between two data representations. Since GDV can be computed on different types of representations, it can be also used to compare the separability of the same data represented in different spaces, such as the different representations returned by different neural networks' layers.

However, we again emphasize that how to determine the appropriate parameters θ from a data set by a supervised learning process minimizing an error (or loss) function is still a critical problem. We notice that error functions usually depend on the final network output values only, without taking care about the results obtained in the hidden layers. Thus, starting from the previous considerations, in this paper we investigate the possibility to achieve a supervised learning approach which favours solutions where $\mathbf{x}$'s representations at the hidden levels have a higher degree of linear separability between the classes with respect to standard approaches. To this aim, we propose a DNN architecture which induces an error function involving the output values of all the network layers. More specifically, as we will discuss in more detail in Section 2, the output of each hidden layer l is sent to an additional linear output layer which is trained to classify the input x on the basis of the input representation encoding in the layer l (see Fig. 1). From now on, we named this architecture *Hidden Classification Layer* network (HCL). We investigated the impact of this type of solution in a series of experimental scenarios as we will discuss in more detail in Section 3.

Although similar approaches have already been partially discussed in the past literature (see, for example, [13]), here we propose both a different version in terms of both neural architecture and error function, and a more extensive experimental analysis (see Sections 2 and 3). In particular, in [13] the supervision of the hidden layer was made by SVMs instead of linear neural layers as in our case. In [14] a cascade of Convolutional Neural Networks (C-CNN) was proposed. C–CNN is composed of hidden layers combined together through dilated convolutions and trained using a proposed progress optimization algorithm. Also in this case, our approach proposes a simpler architecture to favour hidden representations with a higher degree of class separability. The rest of the paper is organized as follow: in Section 2 the proposed method is described; Section 3 describes the experimental setup and the evaluation methods; in Section 4 the results are reported and discussed; finally, Section 5 contains final remarks.

2. Model description

A neural network, as reminded before, is structured in L layers of neurons. Each neuron i belonging to the l-th layer, achieves a two-step computation (see [15], chapter 4): a linear combination a_i^l of the neuron's inputs is computed first, and then the neuron output z_i^l is computed by an activation function $f_l(\cdot)$, i.e., $z_i^l = f_l(a_i^l)$. Usually, activation functions are non-linear function (see [16] for a review). The activation function input a_i^l is usually computed on the basis of real values, said *weights*, associated with the connections coming from the neurons belonging to the layer $l-1$ (and possibly from other previous layers) and a bias value associated to the neuron i. Each layer l is composed of m_l neurons, and the flow of computation proceeds from the first hidden layer to the output layer in a forward-propagation fashion.

In this research work, we focus on C-classes classification problems, with $C \geq 2$. In this context, Cross-Entropy (CE) loss [17] is one of the most common loss function to be optimized. Given a dataset of N samples, $DS = \{(\mathbf{x}^n, \mathbf{t}^n)\}_{n=1}^N$, CE for the nth sample can be expressed as follows:

$$CE^n(\theta; \mathbf{y}^n, \mathbf{t}^n) = -\sum_{c=1}^{C} t_c^n \log(y_c^n)$$

where $\mathbf{t}^n \in \{0,1\}^C$ is the one-hot encoding representation of the class label of the nth sample of the dataset, and $\mathbf{y}^n = \mathbf{y}(\mathbf{x}^n; \theta)$ is the output of the neural network when it is fed with the input $\mathbf{x}^n$. Finally, θ corresponds to all the network parameters. The total CE loss is equal to the sum of the single CE^n over the dataset samples, i.e., $CE = \sum_{n=1}^N CE^n$. As previously said, this loss formulation takes into account only the classification reported by the final layer of the network, without considering how the intermediate network levels affect the final classification scores. By contrast, in our model, HCL network, the data representation corresponding to the output of each hidden layer is used as input of a linear classifier so as to favour a data representation for each level as separable as possible.

More formally, given a DNN composed of $l_1, l_2, \ldots, l_{L-1}$ hidden layers and a final layer l_L having C neurons, we connect each hidden layer l_j, $1 \leq j \leq L-2$ with a new densely connected linear layer l_j^{HCL} acting as an independent linear classifier.

Adopting a proper loss function to train each classifier l_j^{HCL}, we expect that the features learned by the associate layer l_j are the most discriminating as possible. In other words, additional $L-2$ layers $\{l_1^{HCL}, l_2^{HCL}, \ldots, l_{L-2}^{HCL}\}$ composed of C neurons are added, and each l_j^{HCL} layer receives connections from the hidden layer l_j only, making each l_j^{HCL} as an independent linear classifier. Therefore, given a DNN

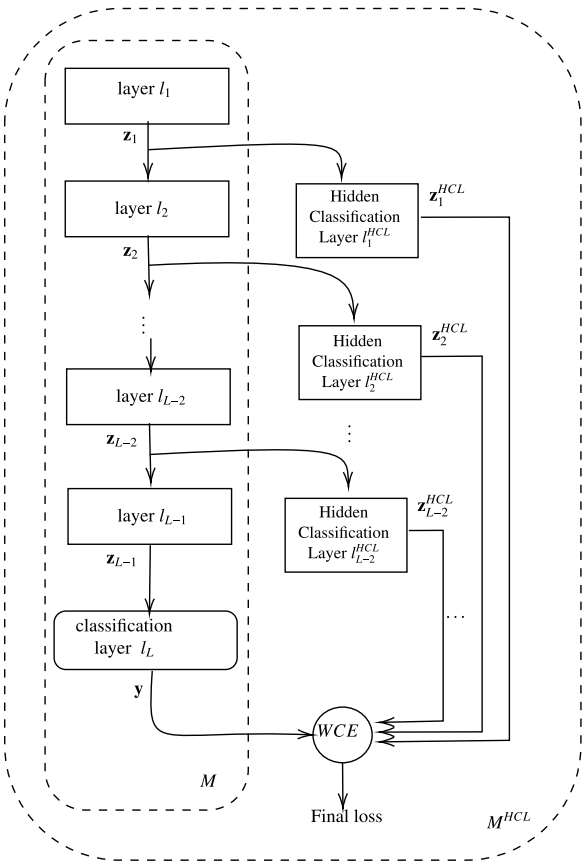

Fig. 1. A scheme of the proposed approach. Each hidden layer l_i of the main branch of the network produces an output $vecz_i$, and the final classification layer l_L produces the output $vecy$. For each hidden layer l_i, $1 \leq i < L$, a further classification layer l_i^{HCL} is added. Each l_i^{HCL} is fed with the respective $\mathbf{z}_i$, producing an output $\mathbf{z}_i^{HCL}$. Therefore, all outputs $\mathbf{z}_i^{HCL}$, $\forall 1 \leq i \leq L-2$, are used together with the network classification output $\mathbf{y}$ to compute the final WCE loss.

M, we obtain an HCL network M^{HCL} which will be composed of two distinct sets of layers: (i) the standard neural network layers $\{l_1, l_2, \dots, l_L\}$, composing the backbone model M, and (ii) the *Hidden Classification Layers* $\{l_1^{HCL}, l_2^{HCL}, \dots, l_{L-2}^{HCL}\}$, composing a set of layers where each layer l_j^{HCL} favours more separable data representations in the respective hidden layer l_j, independently from the subsequent layers. Each Hidden Classification Layer l_j^{HCL} leads to a model having parameters' set θ_j^{HCL} composed of two distinct subsets of parameters, the former corresponding to the connections incoming in the layer l_j^{HCL}, and the latter corresponding to the parameters of the backbone DNN M down to the layer l_j. In Fig. 1 a general scheme of the proposed approach is reported.

Denoting with $\mathbf{z}_{n,j}^{HCL}$ the scores returned by the Hidden Classification Layer l_j^{HCL} tied to the l_j layer on the n-th input sample, we propose the following Weighted Cross-Entropy (WCE) loss formulation:

$$WCE^n(\theta; \mathbf{y}^n, \mathbf{t}^n) = CE^n(\theta^M; \mathbf{y}^n, \mathbf{t}^n) + \sum_{j=1}^{L-2} \lambda_j \cdot CE^n(\theta_j^{HCL}; \mathbf{z}_{n,j}^{HCL}, \mathbf{t}^n) \quad (1)$$

where $\mathbf{y}^n$ is the score returned by the final layer of the classifier M, θ^M are the parameters of the backbone model M, and $\{\lambda_1, \lambda_2, \dots, \lambda_{L-2}\}$ is a set of regularization coefficients greater than or equal to 0. Setting $\lambda_1 = \lambda_2 = \dots = \lambda_{L-2} = 0$ results in standard CE loss applied to the final classification layer only, while different values give different weights to the Hidden Classification Layers $\{l_j^{HCL}\}_{j=1}^{L-2}$.

3. Experimental assessment

3.1. Data and neural network models

The performance of the HCM network architecture is assessed on image classification tasks considering four well-known datasets: *MNIST*, *Fashion MNIST*, *CIFAR-10*, and *CIFAR-100*. The MNIST [18] dataset consists of 70,000 grayscale images at a resolution of 28 × 28 representing 10 different classes (the digits from 0 to 9). It is divided in two sets: the former composed of 60,000 images usually used as training samples and the latter of the remaining 10,000 images usually used as test samples. Fashion-MNIST is a dataset of images representing fashion articles [19]. Fashion-MNIST was proposed as a replacement for the original MNIST dataset for benchmarking machine learning algorithms, sharing the same image size and structure of training and testing splits. Indeed, it provides a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28 × 28 grayscale image, representing one of the following items: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot. CIFAR-10 dataset consists of 60,000 colour images of 10 different classes, that are *airplane*, *automobile*, *bird*, *cat*, *deer*, *dog*, *frog*, *horse*, *ship*, and *truck*. The dataset provides 50,000 training images and 10,000 test images at a resolution of 32 × 32.

Concerning the neural network models which are used as baseline to evaluate the classification enhancement of HCL network architecture, we considered LeNet-5 [18], Hinton network [20], and ResNet18 [21]. They are among the most famous networks exploiting convolutional layers for image classification.

In its standard formulation, LeNet-5 is composed of a sequence of 3 convolutional layers interspersed by 2 sub-sampling layers, followed by 2 final full-connected layers (the latest one for classification). Instead, Hinton network is composed of three convolutional hidden layers interleaved with three maxpooling layers. Finally, the main characteristic of a ResNet is the presence of shortcuts between non-consecutive layers that allows deep networks to be easily trained. In this work, we use the 18 layer residual network (ResNet18) described in [21].

Table 1
Results of the evaluation stage. For each dataset, accuracy on the test set obtained on both the vanilla models (*baseline*) and the proposed ones are reported. ResNet has not been tested on MNIST dataset due to the already very high accuracies obtained with the other models. For CIFAR10 and CIFAR100, performance was evaluated also using augmented data (*augmented*) for ResNet.

	Model	Baseline	Proposed
MNIST	LeNet5	99.0	**99.2**
	Hinton	98.7	**99.4**
FMNIST	LeNet5	90.6	**90.8**
	Hinton	92.1	**92.6**
	ResNet	92.3	**93.3**
CIFAR10	LeNet5	66.2	**71.5**
	Hinton	81.2	**83.3**
	ResNet	86.3	**89.0**
	ResNet (augmented)	94.4	**94.6**
CIFAR100	LeNet5	34.9	**38.4**
	Hinton	52.5	**53.3**
	ResNet	60.1	**63.6**
	ResNet (augmented)	74.7	**75.4**

3.2. Evaluation

Each model is evaluated on both the original version (*vanilla*) proposed in their respective works and on its modified instance as described in Section 2. All the models were trained using the same experimental setup reported in their reference papers, except for Learning Rate LR, the number of Max Epochs ME, and the Patience Epochs PE, which can be strongly dependent by the network architecture. ME and PE are experimentally set to 1000 and 200 respectively, since we experimentally noticed that these values are enough to converge in all the analysed cases, while optimal LR and λ values are found through a grid-search approach. For the LR, the search space was $LR \in [10^{-5}, 10^{-1}]$, instead different combinations are considered for λ parameters. Experiments on ResNet involving CIFAR-10 and CIFAR-100 dataset were made both considering only original data and augmented data, using 4 pixels zero padding, corner cropping, and random flipping.

Importantly, in order to experimentally show that the proposed method leads toward more easily separable data representations we computed Generalized Discrimination Value (GDV) measure [12] for each vanilla network's layer and its corresponding version equipped with hidden classification layers. We expect that, as the depth of the network increases, the data representations obtained with the proposed network's layout are more easily separable respect to the representations obtained by the respective models without additional layers. Note that GDV is a measure of how well different data classes separate. GDV values result 0.0 for data points with randomly shuffled classes, and −1.0 in the case of perfectly separable classes. More in detail, GDV on a data representation $\mathbf{z}$ is defined as

$$GDV(\mathbf{z}) = \frac{1}{\sqrt{D}} \left(\frac{1}{L} \sum_{c=1}^{C} d^{intra}(\mathbf{z}_c) - \frac{2}{C(C-1)} \sum_{c=1}^{C-1} \sum_{m=c+1}^{C} d^{inter}(\mathbf{z}_c, \mathbf{z}_m) \right)$$

where $d^{intra}(\mathbf{z}_c)$ is the mean intra-class distance on the representations $\mathbf{z}_c$ of the data belonging to the class c, and $d^{inter}(\mathbf{z}_c, \mathbf{z}_m)$ is the mean inter-class distance on the data representations $\mathbf{z}_c, \mathbf{z}_m$ belonging to the c and m classes.

4. Results

In Table 1 the test set accuracy, which was obtained by both the HCL network architecture and the vanilla networks, is reported. It is shown that the adoption of the HCL architecture improves the accuracy in all the cases, especially in the cases where a low accuracy for vanilla networks was obtained. In these cases, in fact, HCL network architecture appears to give a more significant improvement. In Figs. 2

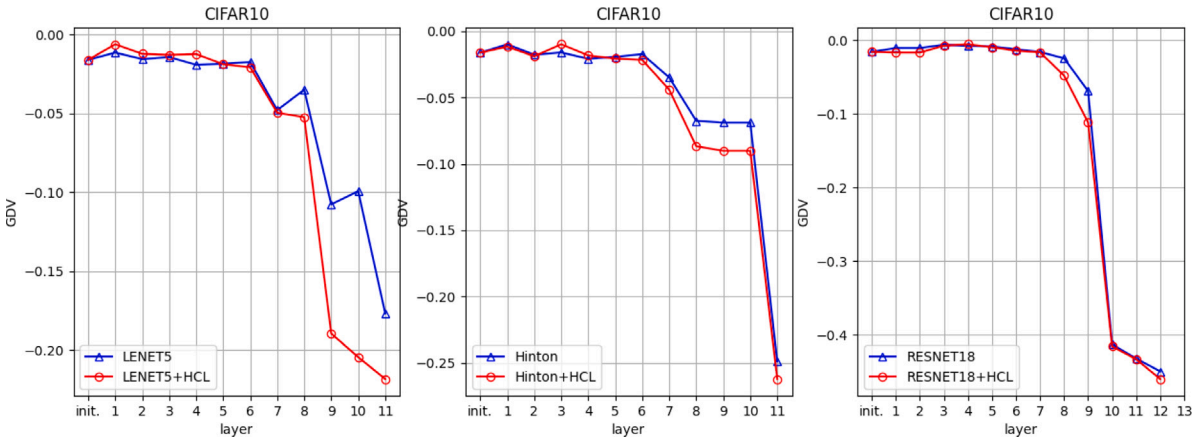

Fig. 2. GDV values obtained with the proposed approach (LAT) compared with the vanilla networks on the CIFAR10 dataset. On the x axis, the layer of the model and on the y axis the respective GDV value.

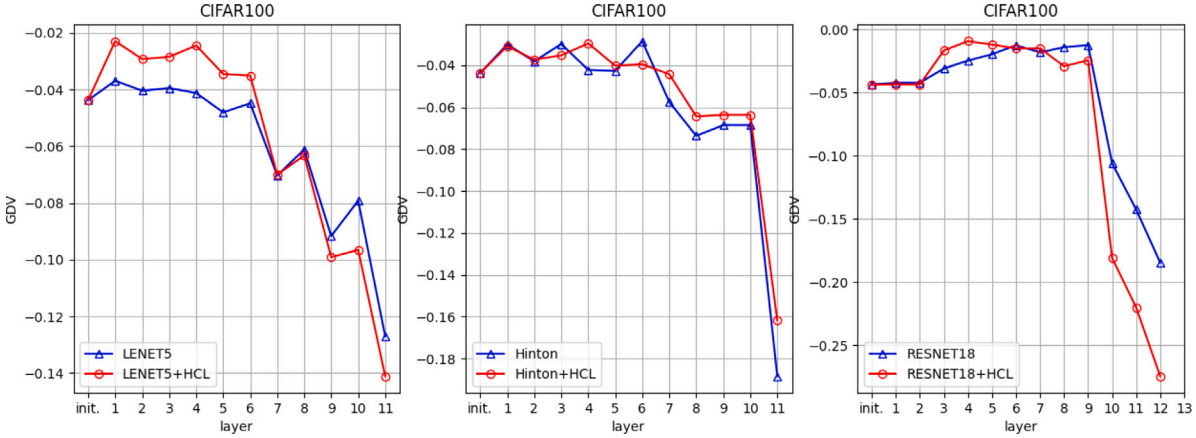

Fig. 3. GDV values obtained with the proposed approach (LAT) compared with the vanilla networks on the CIFAR100 dataset. On the x axis, the layer of the model and on the y axis the respective GDV value.

and 3 the GDV values for each layer of each model and for CIFAR10 and CIFAR100 dataset are reported.

In Fig. 4, we present the different terms of the Weighted Cross-Entropy (WCE) during the training epochs (x axis). This includes the CE loss calculated for the entire model (the first term in Eq. (1), solid line in Fig. 4), as well as the CE loss values obtained from each HCL (dashed and dotted lines), that are weighted and combined to form the second term in Eq. (1).

A noteworthy observation is that the CE loss to the first HCL is higher compared to all the subsequent HCL losses, for both the CIFAR10 and CIFAR100 datasets. Despite this, we can notice that the loss corresponding to HCL l3 performs relatively worse in the case of CIFAR10, while it exhibits the best performance compared to all other HCL layers for CIFAR100.

This difference in performance between layers can be linked to the distinct types of layers present in LeNet5. Specifically, the first and third layers are convolutional, whereas the second layer is a max-pooling layer. This discrepancy potentially results in divergent optimization outcomes. Therefore, despite achieving enhanced linear separability across the layers (as indicated by the GDVs), we cannot make definitive assumptions about the model's performance beforehand, since it can be affected by the data and the network's layers. Therefore, the placement of the HLC layers in the architecture should be approached with care and considered as part of the Hyperparameter Optimization process. Moreover, the Cross-Entropy (CE) computed on the model output (depicted by the solid line) for CIFAR 100 is lower than the CE values for HCL layers 1, 2, 4, and 5, but higher than that of HCL layer 3 (which is the lowest). This observation suggests that the primary output CE benefits positively from the losses inducted by the HCLs.

In Fig. 5, CE losses on the validation sets observed during the training for both the baseline models and the proposed HCL architectures on CIFAR 10 and CIFAR 100 datasets are reported. For brevity, we chosen to show the results for LeNet5 and ResNet models; however, the same observations apply to Hinton's model as well.

A notable point is that, starting from a certain point during the training, the CE of the proposed architecture consistently outperforms the corresponding CE on the baseline models. This suggests that the HCL model effectively acts as a regularization mechanism, selectively filtering out information that may hinder the network's ability to generalize across its layers, thereby mitigating overfitting.

5. Conclusion

In this research work, we experimentally investigated the impact of constraining the classification complexity of the intermediate input representations with respect to their linear separability on the performances of DNNs in classification tasks. To this aim, we proposed a novel DNN architecture, which we named Hidden Classification Layer (HCL) network, where the output of each standard hidden layer is sent to a Hidden Classification Layer trained to classify the input x based on the x representation given by the standard layer itself. HCL network architecture allows obtaining solutions with input representations at the hidden levels having a lower classification complexity with respect to their linear separability. Note that our approach can be applied in

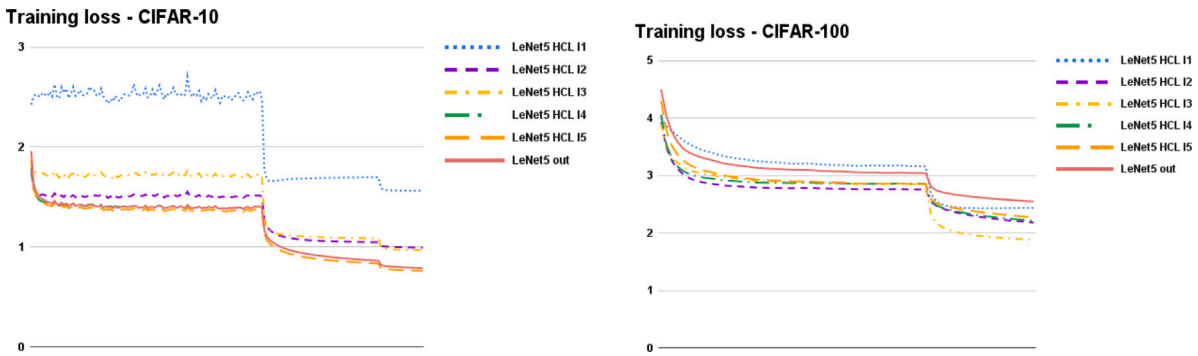

Fig. 4. The different terms of the Weighted Cross-Entropy (WCE) during the training epochs (x-axis) on LeNet-5 for CIFAR 10 (left) and CIFAR 100 (right) datasets. This includes the CE loss calculated for the entire model (first term of Eq. (1), solid line), and the CE loss values obtained from each HCL (dashed and dotted lines), that are weighted and combined to form the second term in Eq. (1).

Fig. 5. Cross-Entropy (CE) validation losses (y axis) observed during training (x axis) for both the LeNet5 (first row) and ResNet (second row) using CIFAR 10 (first column) and CIFAR 100 (second column) datasets.

slightly different ways: (1) given an already known neural network architecture, one can, first, augment it by Hidden Classification Layers and, then, train the whole system from scratch; (2) given an already trained neural network architecture, one can, first, augment it by Hidden Classification Layers and, then, tune the whole system; (3) one can design and train a new neural architecture equipped with Hidden Classification Layers. In this study, we used the first approach to test our proposal by considering three successful neural network models (LeNet-5, Hinton network, and ResNet18). These models were trained with and without Hidden Classification Layers to evaluate the impact of HCL on the model performances experimentally. Each model was trained and tested on four datasets (MNIST, fashion-MNIST, CIFAR-10 and CIFAR-100). The results show that the HCL network has a positive impact uniformly (see Table 1). It is interesting that the proposed approach leads to a GDV improvement in almost all cases, suggesting that the HCL network architecture can help the model build more separable inner representations. Moreover, it is worth noting that, in all the cases and with and without Hidden Classification Layers, the GDV values exhibit only a slight decrement for the initial network layers or they have even a wavering behaviour. Just only for the last layers, there is a sharp decrement (this decrease is particularly pronounced for HCL networks). Thus, these results are consistent with [22], where the authors show experimentally that, during the learning phase, the loss derivatives with respect to the network parameters behave very similarly to a random walk on the first weight layers. In fact, in our case, data separability occurs mainly in the last layers of the network at the end of the learning process.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

All the used data are publicly available.

Funding

This work was partially supported by the European Union - FSE-REACT-EU, PON Research and Innovation 2014–2020 DM1062/2021 contract number 18-I-15350-2, by the Ministry of University and Research, PRIN research project "BRIO – BIAS, RISK, OPACITY in AI: design, verification and development of Trustworthy AI.", Project no. 2020SSKZ7R, by the Ministry of Economic Development, "INtegrated Technologies and ENhanced SEnsing for cognition and rehabilitation" (INTENSE) project (CUP: B69J23001290005), by the PNRR MUR project PE0000013-FAIR (CUP: E63C22002150007) and by HPC, Big Data e Quantum Computing (CUP: E63C22000980007).

References

[1] P. Wang, E. Fan, P. Wang, Comparative analysis of image classification algorithms based on traditional machine learning and deep learning, Pattern Recognit. Lett. 141 (2021) 61–67.
[2] Y. Celik, M. Talo, O. Yildirim, M. Karabatak, U.R. Acharya, Automated invasive ductal carcinoma detection based using deep transfer learning with whole-slide images, Pattern Recognit. Lett. 133 (2020) 232–239.
[3] D. Freire-Obregon, F. Narducci, S. Barra, M. Castrillon-Santana, Deep learning for source camera identification on mobile devices, Pattern Recognit. Lett. 126 (2019) 86–91.
[4] M. Gravina, S. Marrone, M. Sansone, C. Sansone, DAE-CNN: Exploiting and disentangling contrast agent effects for breast lesions classification in DCE-MRI, Pattern Recognit. Lett. 145 (2021) 67–73.
[5] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (7553) (2015) 436–444.
[6] N. Cohen, O. Sharir, A. Shashua, On the expressive power of deep learning: A tensor analysis, in: Conference on Learning Theory, PMLR, 2016, pp. 698–728.
[7] G.-B. Huang, Y.-Q. Chen, H.A. Babri, Classification ability of single hidden layer feedforward neural networks, IEEE Trans. Neural Netw. 11 (3) (2000) 799–801.
[8] I.D. Longstaff, J.F. Cross, A pattern recognition approach to understanding the multi-layer perception, Pattern Recognit. Lett. 5 (5) (1987) 315–319.
[9] B. Lerner, H. Guterman, M. Aladjem, I.h. Dinstein, A comparative study of neural network based feature extraction paradigms, Pattern Recognit. Lett. 20 (1) (1999) 7–14.
[10] A.C. Lorena, L.P. Garcia, J. Lehmann, M.C. Souto, T.K. Ho, How complex is your classification problem? A survey on measuring classification complexity, ACM Comput. Surv. 52 (5) (2019) 1–34.
[11] C. Ferri, J. Hernández-Orallo, R. Modroiu, An experimental comparison of performance measures for classification, Pattern Recognit. Lett. 30 (1) (2009) 27–38.
[12] A. Schilling, A. Maier, R. Gerum, C. Metzner, P. Krauss, Quantifying the separability of data classes in neural networks, Neural Netw. 139 (2021) 278–293.
[13] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, Z. Tu, Deeply-supervised nets, in: Artificial Intelligence and Statistics, PMLR, 2015, pp. 562–570.
[14] F. Wang, R. Liu, Q. Hu, X. Chen, Cascade convolutional neural network with progressive optimization for motor fault diagnosis under nonstationary conditions, IEEE Trans. Ind. Inform. 17 (4) (2020) 2511–2521.
[15] C.M. Bishop, N.M. Nasrabadi, Pattern Recognition and Machine Learning, vol. 4, Springer, 2006.
[16] A. Apicella, F. Donnarumma, F. Isgrò, R. Prevete, A survey on modern trainable activation functions, Neural Netw. 138 (2021) 14–32.
[17] Q. Wang, Y. Ma, K. Zhao, Y. Tian, A comprehensive survey of loss functions in machine learning, Ann. Data Sci. 9 (2) (2022) 187–212.
[18] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[19] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017, arXiv preprint arXiv:1708.07747.
[20] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, 2012, arXiv preprint arXiv:1207.0580.
[21] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[22] D. Balduzzi, M. Frean, L. Leary, J. Lewis, K.W.-D. Ma, B. McWilliams, The shattered gradients problem: If resnets are the answer, then what is the question? in: International Conference on Machine Learning, PMLR, 2017, pp. 342–350.