



Universidad Autónoma
de Madrid

Biblos-e Archivo
Repositorio Institucional UAM

Repositorio Institucional de la Universidad Autónoma de Madrid

<https://repositorio.uam.es>

Esta es la **versión de autor** del artículo publicado en:
This is an **author produced version** of a paper published in:

Díaz-Vico, D., Fernández, A., Dorronsoro, J.R. (2021). Companion Losses for Deep Neural Networks. In: Sanjurjo González, H., Pastor López, I., García Bringas, P., Quintián, H., Corchado, E. (eds) Hybrid Artificial Intelligent Systems. HAIS 2021. Lecture Notes in Computer Science 12886 (2021): 538-549

Copyright: © 2021 Springer Nature

This version of the article has been accepted for publication, after peer review (when applicable) and is subject to Springer Nature's AM terms of use, but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at:

DOI: https://doi.org/10.1007/978-3-030-86271-8_45

Companion Losses for Deep Neural Networks

David Díaz-Vico^{2(✉)}, Angela Fernández¹, and José R. Dorronsoro^{1,2}

¹ Department of Computer Engineering, Universidad Autónoma de Madrid,
Madrid, Spain

² Inst. Ing. Conocimiento, Universidad Autónoma de Madrid,
Madrid, Spain
`david.diaz@iic.uam.es`

Abstract. Modern Deep Neuronal Network backends allow a great flexibility to define network architectures. This allows for multiple outputs with their specific losses which can make them more suitable for particular goals. In this work we shall explore this possibility for classification networks which will combine the categorical cross-entropy loss, typical of softmax probabilistic outputs, the categorical hinge loss, which extends the hinge loss standard on SVMs, and a novel Fisher loss which seeks to concentrate class members near their centroids while keeping these apart.

1 Introduction

After the seminal work of G. Hinton [8] and J. Bengio [11] and starting about 2010, Deep Neural Networks (DNNs) have exploded in terms of scientific advances, technological improvements and great successes on many applications. There are many reasons for this, but paramount among them is the great flexibility that modern DNN environments such as TensorFlow [1] or PyTorch [9] allow to define, train and exploit DNN models. Key for this are the modern tools for automatic differentiation that make possible the definition of very general network architectures and losses. For instance, this has made possible to incorporate under a DNN framework cost functions such as the hinge and ϵ -insensitive losses, with models and results that are very competitive with those of the standard Gaussian SVMs [5].

Once that more general losses are available, a natural next step is to try to combine some of them in principle independent losses within the same network, so that they can take advantage of their different goals to jointly improve on their individual achieved results. For instance, consider for two-class problems the competing cross-entropy loss, customarily used for DNN classification, with the SVM

The authors acknowledge financial support from the European Regional Development Fund and the Spanish State Research Agency of the Ministry of Economy, Industry, and Competitiveness under the project PID2019-106827GB-I00. They also thank the UAM-ADIC Chair for Data Science and Machine Learning and gratefully acknowledge the use of the facilities of Centro de Computación Científica (CCC) at UAM.

hinge loss. The goal of the former is to assume a certain posterior probability and estimate a model that maximizes its sample based likelihood, while that of the latter is essentially to find a separating hyperplane with a margin as large as possible given the sample. A similar situation arises in multiclass problems, where the categorical cross-entropy with softmax outputs is used for DNN classifiers while the categorical hinge loss [3] is used for multiclass SVM-like classifiers. In these problems one or several of the losses act as the main one, while the others act as companions in the sense that they accompany the main loss towards a better model. This idea of combining losses has been applied in other areas of knowledge, specially in computer vision [10, 14], although following a different rationale.

Although we will not deal with them here, competing losses for regression problems would be the squared error of regression DNNs and the ϵ -insensitive loss used in support vector regression. Again, the underlying problem is basically the same, but the latter establishes an ϵ -wide tube around the fitted model and only penalizes errors outside the tube. This results on models more robust with respect to outliers but with the drawback of ignoring small errors, which may be important in some settings and that the squared error does not ignore. Given these different but not necessarily competing goals, it is in principle conceivable that both losses could work together towards building a model that improves on those built separately with each loss.

The goal of this work is precisely to explore these possibilities for classification problems. More precisely, we will compare the combination of the cross-entropy and hinge losses for two-class problems, and that of the categorical cross-entropy and hinge losses in multiclass ones. To these we will add a squared loss-based cost function which enforces for its inputs on each class to be concentrated near the class centroids while trying to keep these centroids apart. This approach has been proved [13] to yield linear models whose outputs can theoretically be seen to be equivalent with those provided by the classical Fisher Discriminant Analysis and that has been extended to a DNN setting in [4]. While trying to yield a classifier directly, such a loss can produce a pattern representation on the last hidden layer of a DNN which can make easier the job of a classifier acting on these representations and, hence, result in a better model.

We will work with a substantial number of classification problems and our results point out that this approach can indeed give such results. In fact, and as we will experimentally show, combining the Fisher loss with the cross-entropy or hinge ones improves on the models obtained when only single losses are used. On the other hand, the cross-entropy plus hinge combination ties at best with a single cross-entropy loss. This has to be further studied but a possible reason may be that, at least in our experiments, cross-entropy DNNs yield better results than hinge-based ones (we address reasons for this later in this paper). In summary, our contributions here are:

- The proposal of DNNs with combined losses for two- and multi-classification, which we implement as Keras [2] functional models.
- A substantial experimental work showing positive results that deserve further study.

The rest of the paper is organized as follows. We review the losses used in Sect. 2, discuss how to combine them and give some implementation details. Section 3 contains details on the datasets used, the experimental methodology and results, and a discussion and a final section offers some conclusions as well as pointers to further work. We point out that, throughout the paper, by deep networks we mean artificial neural networks that use modern techniques such as automatic differentiation, Glorot-Bengio initializations [6], ReLU activations or Adam optimizers [7], rather than they having actually deep (i.e. many layered) architectures; in fact, in our experiments we will apply all these techniques but on a single layer network with 100 units.

2 Classification Losses

Throughout this section we will work with DNN architectures that yield models acting on a pattern x with outputs $F(x, \mathcal{W})$, where \mathcal{W} denotes the set of weight matrices and bias vectors associated with the network’s architecture. We will denote targets as y , which can be either $\{-1, 1\}$ for two-class problems or one-hot encoded vectors for multiclass ones. We denote the network outputs at the last hidden layer as $z = \Phi(x, \widetilde{\mathcal{W}})$, with $\widetilde{\mathcal{W}}$ the weight and bias set of all layers up to the last one. Such a z is then transformed as $Wz + B$, where W, B are either the transpose of an n_H dimensional vector w , with n_H the number of hidden units in the last hidden layer, and a scalar b , or a $K \times n_H$ matrix and a K dimensional vector in a K -class problem. These $Wz + B$ will be the final network outputs in the case of the deep SVM (or deep Fisher networks, as we describe them below); for the more standard DNN classifiers, the network output is obtained applying to them either a sigmoid or a softmax function.

In more detail, and starting with two-class problems, the usual DNN loss is the binary **cross-entropy**, given by

$$\ell_{bc}(\mathcal{W}; S) = \ell(w, b, \widetilde{\mathcal{W}}) = - \sum_p y^p (w \cdot z^p + b) + \sum_p \log(1 + e^{w \cdot z^p + b}) \quad (1)$$

where S denotes an i.i.d. sample $S = \{(x^p, y^p)\}$ with N patterns, z represent the last hidden layer outputs, w represent the weights which connects the last hidden layer with the network’s output and b is the vector bias of the output. We recall that this expression is just the sample’s minus log-likelihood associated to the assumption

$$P(y|x) = P(y|x; w, b, \widetilde{\mathcal{W}}) = \frac{1}{1 + e^{-y(w \cdot z + b)}} \quad (2)$$

for the posterior probability of the y class, and we assume a sigmoid network output. For general multiclass problems, the network output function is the softmax

$$F_j(x; \mathcal{W}) = \frac{e^{w_j \cdot z + b_j}}{\sum_{k=0}^{K-1} e^{w_k \cdot z + b_k}}. \quad (3)$$

Obviously then $\sum_j F_j(x; \mathcal{W}) = 1$ and we assume $P(j|x) \simeq F_j(x; \mathcal{W})$. For two-class problems this reduces to the previous output if we take $w = w_0 - w_1$. For

the loss to be used here, we assume one-hot encoded targets, i.e., the target of the k -th class is $e_k = (0, \dots, \underbrace{1}_k, \dots, 0)$; then, the probability of getting patterns x^p in class k_p (i.e., $y_{k_p}^p = 1$) within an i.i.d. sample $S = (X, Y)$ is

$$P(Y|X; \mathcal{W}) = \prod_{p=1}^N P(k_p|x^p; \mathcal{W}) = \prod_{p=1}^N \prod_{m=0}^{K-1} P(m|x^p; \mathcal{W})^{y_m^p} \simeq \prod_{p=1}^N \prod_{m=0}^{K-1} F_c(x; \mathcal{W})^{y_m^p}, \quad (4)$$

and we estimate the DNN's weights \mathcal{W} by minimizing the minus log of the approximate sample's likelihood

$$\tilde{P}(Y|X; \mathcal{W}) = \prod_{p=1}^N \prod_{m=0}^{K-1} F_m(x^p; \mathcal{W})^{y_m^p}. \quad (5)$$

That is, we will minimize the categorical cross-entropy loss

$$\ell_{cce}(\mathcal{W}) = -\log \tilde{P}(Y|X; \mathcal{W}) = -\sum_{p=1}^N \sum_{m=0}^{K-1} y_m^p \log F_m(x^p; \mathcal{W}). \quad (6)$$

Once an optimal weight set \mathcal{W}^* has been obtained, the decision function on a new x is given by $\arg \max_m F_m(x; \mathcal{W}^*)$, i.e. the class with the maximum posterior probability.

Turning our attention to two-class SVMs, the local loss is now the **hinge loss** $h(x, y) = \max\{0, 1 - yF(x, \mathcal{W})\}$; here the network has linear outputs, i.e., $F(x; \mathcal{W}) = F(x; w, b, \mathcal{W}) = w \cdot \Phi(x, \mathcal{W}) + b$. The global loss is now

$$\ell_h(\mathcal{W}; S) = \sum_p \max\{0, 1 - y^p F(x^p, \mathcal{W})\}. \quad (7)$$

There are several options to extend SVMs for multiclass problems. The usual approach in a kernel setting is to use either a one-vs-one (ovo) or a one-vs-rest (ovr) approach so that just binary kernel classifiers have to be built. Unfortunately, this cannot be directly translated to a DNN setting but there are two other ways to define multiclass local losses. The first one is due to Weston and Watkins [12] which for a pattern x in class k_x (i.e., $y_{k_x} = 1$) propose the local loss

$$\ell(x, y) = \max \left\{ 0, \sum_{m \neq k_x} (1 + F_m(x) - F_{k_x}(x)) \right\} \quad (8)$$

where we denote as $F_0(x), \dots, F_{K-1}(x)$ the network's lineal outputs. The alternative to this is the local loss proposed by Crammer and Singer [3], namely

$$\ell(x, y) = \max \left\{ 0, \max_{m \neq y} (1 + F_m(x) - F_{k_x}(x)) \right\}. \quad (9)$$

Notice that both coincide for two-class problems and, moreover, if in this case we require $w_0 = w_1 = \frac{1}{2}w$ and $b_0 = b_1 = \frac{1}{2}b$, they coincide with the local hinge loss. We will use the second one, that results in the categorical hinge global loss

$$\ell_{ch}(\mathcal{W}) = \sum_{p=1}^N \max \left\{ 0, 1 - F_{k_{x^p}}(x^p) + \max_{m \neq k_{x^p}} F_m(x) \right\}; \quad (10)$$

here $F_m(x)$ denotes the m -th component of the network's K dimensional linear output, i.e., $F_m(x) = w_m \cdot \Phi(x; \tilde{\mathcal{W}}) + b_m$. Now, once an optimal weight set \mathcal{W}^* has been obtained, the decision function on a new x is given again by $\arg \max_m F_m(x; \mathcal{W}^*)$, although now no posterior probabilities are involved.

We will finally consider what we call the **Fisher loss**. The goal in standard Fisher Discriminant Analysis is to linearly project patterns so that they concentrate near the projected class means while these are kept apart. To achieve this, one seeks to maximize the trace criterion

$$g(A) = \text{trace}(s_T^{-1} s_B) = \text{trace}((A^t S_T A)^{-1} (A^t S_B A)), \quad (11)$$

where A is the projection matrix, S_B and S_T denote the between-class and total covariance matrices, respectively, of the sample patterns and s_B and s_T are their counterparts for the projections $z = Ax$. Solving $\nabla_A g = 0$ leads to the system $S_T^{-1} S_B A = A \Lambda$, with Λ the non-zero eigenvalues of $S_T^{-1} S_B$. For such an A we have

$$g(A) = \text{trace}(s_T^{-1} s_W) = \text{trace } \Lambda = \lambda_1 + \dots + \lambda_q, \quad (12)$$

where s_W represents the within-class matrix. This expression is maximized by sorting the eigenvalues $\{\lambda_1, \lambda_2, \dots\}$ in Λ in descending order and selecting the $K - 1$ largest ones and some conveniently normalized associated eigenvectors. Since the minimizer of (11) is not uniquely defined, it is usually normalized as $A^t S_T A = I_{K-1}$, being I_N the identity matrix of size N . It turns out that an equivalent solution can be obtained by solving the least squares problem $\min \frac{1}{2} \|Y^f - XW - \mathbf{1}_N B\|^2$, where W is a $d \times K$ matrix, B a $1 \times K$ vector, $\mathbf{1}_N$ is the all ones vector and for a pattern x^p in the p -th row of the data matrix X which is in class m , we have $Y_{pm}^f = \frac{N - N_m}{N \sqrt{N_m}}$ when x^p and as $Y_{pj}^f = -\frac{\sqrt{N_j}}{N}$ for $j \neq m$, with N_j the number of patterns in class j . Then, it can be shown that the optimal W^* is equivalent, up to a rotation, to a solution \tilde{V} of (11) subject to the normalization $\tilde{V}^t S_T \tilde{V} = \Lambda$; see [4, 13] for more details. As a consequence, any classifier defined in terms of distances to class means will give the same results with the Fisher's projections using \tilde{V} than with the least squares ones using W^* .

This can be extended to a DNN setting by solving

$$\min_{W, B, \tilde{W}} \frac{1}{2} \|Y^f - F(X, \mathcal{W})\|^2 = \frac{1}{2} \|Y^f - \Phi(X; \tilde{\mathcal{W}})W - \mathbf{1}_N B\|^2. \quad (13)$$

As discussed in [4], this could be exploited to define a Fisher-like distance classifier on the network outputs $F(x, \mathcal{W})$; however, those classifiers are in general worse than those based on the categorical entropy or hinge losses. On the other hand, such a loss is likely to enforce the last hidden layer projections z to be concentrated around their class means while keeping these apart and, hence help entropy or hinge based classifiers to perform better.

In this line, the above suggests that instead of using just one of the previous losses, we can try to combine them, something that can be done by defining a

Table 1. Sample sizes, number of features and number of classes.

	Size	Size test	Features	Classes
a4a	4781	27780	123	2
a8a	22696	9865	123	2
australian	690	–	14	2
breast-cancer	569	–	30	2
diabetes	768	–	8	2
digits	1797	–	64	10
dna	2000	–	180	3
german	1000	–	24	2
letter	10500	5000	16	26
pendigits	7494	3498	16	10
protein	14895	6621	357	3
satimage	4435	–	36	6
segment	2310	–	19	7
usps	7291	–	256	10
w7a	24692	25057	300	2
w8a	49749	14951	300	2

DNN with two or even three output sets upon which one of these losses acts. In the most general setting, we may have outputs $\hat{y}_{ce}, \hat{y}_{ch}$ and \hat{y}_f , one-hot encoded targets y and the y_f Fisher-like targets just defined for the loss (13), and we define the combined loss

$$\ell(y, y_f, \hat{y}_{ce}, \hat{y}_{ch}, \hat{y}_f) = \ell_{ce}(y, \hat{y}_{ce}) + \lambda \ell_{ch}(y, \hat{y}_{ch}) + \mu \ell_f(y_f, \hat{y}_f), \quad (14)$$

with $\{\lambda, \mu\}$ appropriately chosen hyperparameters. In our experiments next, we will consider the (ce, fisher), (hinge, fisher), (ce, hinge) and (ce, hinge, fisher) loss combinations; for simplicity we will just take $\lambda = \mu = 1$.

Table 2. Test accuracies of the models considered.

	ce	ce-fisher	hinge	hinge-fisher	ce-hinge	ce-hinge-fisher	max	min
a4a	84.38	84.36	84.39	84.40	84.43	84.45	84.45	84.36
a8a	85.10	85.51	84.96	84.93	85.20	85.09	85.51	84.93
australian	86.52	85.65	85.22	85.07	85.36	85.07	86.52	85.07
breast-cancer	97.72	98.07	96.84	97.19	96.49	96.31	98.07	96.31
diabetes	77.60	77.60	76.82	76.95	77.08	76.95	77.60	76.82
digits	98.22	98.44	98.39	98.27	98.50	98.22	98.50	98.22
dna	95.70	95.85	94.65	94.85	95.55	95.60	95.85	94.65
german	77.90	77.50	75.30	76.40	74.00	74.80	77.90	74.00
letter	95.38	95.28	95.42	95.66	94.98	95.42	95.66	94.98
pendigits	99.52	99.45	99.47	99.52	99.49	99.44	99.52	99.44
protein	69.78	69.76	66.49	66.71	67.62	67.07	69.78	66.49
satimage	91.09	90.55	91.54	91.25	91.75	91.25	91.75	90.55
segment	97.66	97.79	97.01	97.36	97.88	97.79	97.88	97.01
usps	97.79	97.93	97.82	97.53	97.65	97.68	97.93	97.53
w7a	98.79	98.84	98.83	98.81	98.83	98.84	98.84	98.79
w8a	98.97	98.99	98.84	98.88	99.00	99.00	99.00	98.84

3 Experimental Results

In this section we will describe the considered models, describe the datasets we will use, present our experimental methodology and results, and finish the section with a brief discussion.

3.1 Models Considered

We will consider six basic model configurations, involving different network outputs, losses and ways to make predictions, namely

- `ce`: the model uses softmax outputs and the categorical cross-entropy loss. Class labels are predicted as the index of the output with the largest a posteriori probability.
- `hinge`: the model uses linear outputs and the categorical hinge loss. Class labels are predicted as the index of the largest output.
- `ce_hinge`: the model uses two different outputs, one with softmax activations and the other with linear ones; the losses are the categorical cross-entropy and the categorical hinge, respectively. To get predictions, the softmax function is applied to the second output set, so that we can see the entire output vector as made of estimates of posterior probabilities (although this is not true for the second set, as no probability model is assumed for the hinge loss); class labels are predicted as the index of the output with the largest value.
- `ce_fisher`: the model uses two different output sets, one with softmax activations and the other with linear ones. The categorical cross-entropy is minimized on the first and the Fisher loss introduced in Sect. 2 on the second. Label predictions are computed on the first set, as the one with the largest a posteriori probability.
- `hinge_fisher`: the model uses two linear outputs, the first one to minimize the categorical hinge loss and the second one the Fisher loss. Class labels are predicted as the index of the largest output of the first set.
- `ce_hinge_fisher`: the model uses now three different outputs, a first one with softmax activations and the other two with linear outputs; the categorical cross-entropy, categorical hinge and Fisher losses are minimized, respectively, on each output set. Here the softmax function is also applied to the hinge loss linear outputs and class labels are predicted as the index of the first two outputs with the largest value.

3.2 Datasets

We will work with sixteen datasets, namely `a4a`, `a8a`, `australian`, `breast_cancer`, `diabetes`, `digits`, `dna`, `german`, `letter`, `pendigits`, `protein`, `satimage`, `segment`, `usps`, `w7a` and `w8a`; eight of them are multiclass and the rest binary. All are taken from the LIBSVM data repository, except when pointed out otherwise. Table 1 shows their train and test (when available) sample sizes, dimensions and the number of classes. We give more details about them below.

Table 3. Model rankings for each problem in ascending accuracies.

	ce	ce-fisher	hinge	hinge-fisher	ce-hinge	ce-hinge-fisher
a4a	5	6	4	3	2	1
a8a	3	1	5	6	2	4
australian	1	2	4	5	3	5
breast-cancer	2	1	4	3	5	6
diabetes	1	1	6	4	3	4
digits	5	2	3	4	1	5
dna	2	1	6	5	4	3
german	1	2	4	3	6	5
letter	4	5	2	1	6	2
pendigits	1	5	4	1	3	6
protein	1	2	6	5	3	4
satimage	5	6	2	3	1	3
segment	4	2	6	5	1	2
usps	3	1	2	6	5	4
w7a	6	1	3	5	3	2
w8a	4	3	6	5	1	1
ave	3	2.6	4.2	4	3.1	3.6

- **a4a** and **a8a**. Variations of the `adult` of predicting whether income exceeds \$50,000 per year based on census data.
- **australian**. The goal is to decide whether or not an application is credit-worthy.
- **breast_cancer**. The goal is here to predict whether a patient is to be diagnosed with cancer.
- **diabetes**. The objective here is to diagnose the presence of hepatitis on a sample of Pima Indian women.
- **digits**. We want to classify pixel rasters as one of the digits from 0 to 9; the subset is pre-loaded in the *scikit-learn* library.
- **dna**. The goal is to classify splice-junction gene sequences into three different classes.
- **german**. This is another problem where patterns are to be classified as either good or bad credits.
- **letter**. Pixel displays are to be identified as one of the 26 English capital letters.
- **pendigits**. Images of handwritten digits between 0 and 9 are to be classified.
- **satimage**. The goal is to classify the central pixel in a satellite image; we will work only with the 4,435 training subsample.
- **segment**. We want to classify satellite images into one of seven categories.
- **usps**. We want to classify image rasters as a digit between 0 and 9.
- **w7a** and **w8a**. Variants of a classification problem of web pages.

3.3 Experimental Methodology and Results

Recall that all model losses include a L_2 regularization term, which requires the selection of a hyperparameter α so we will proceed first to estimate the optimal

Table 4. Model rankings for each problem in ascending accuracies after putting together models with closer rankings.

	ce	ce-fisher	ce-hinge	ce-hinge-fisher	hinge	hinge-fisher
a4a	2.0	3.0	1.0	1.0	3.0	2.0
a8a	3.0	1.0	2.0	1.0	2.0	3.0
australian	1.0	2.0	3.0	2.0	1.0	2.0
breast-cancer	2.0	1.0	3.0	3.0	2.0	1.0
diabetes	1.0	1.0	3.0	1.0	3.0	1.0
digits	3.0	2.0	1.0	3.0	1.0	2.0
dna	2.0	1.0	3.0	1.0	3.0	2.0
german	1.0	2.0	3.0	3.0	2.0	1.0
letter	1.0	2.0	3.0	2.0	2.0	1.0
pendigits	1.0	3.0	2.0	3.0	2.0	1.0
protein	1.0	2.0	3.0	1.0	3.0	2.0
satimage	2.0	3.0	1.0	2.0	1.0	2.0
segment	3.0	2.0	1.0	1.0	3.0	2.0
usps	2.0	1.0	3.0	2.0	1.0	3.0
w7a	3.0	1.0	2.0	1.0	2.0	3.0
w8a	3.0	2.0	1.0	1.0	3.0	2.0
ave	1.9	1.8	2.2	1.8	2.1	1.9

Table 5. Accuracy spreads across all models as percentages of the difference between the maximum and minimum accuracies over the minimum one.

	a4a	a8a	austr	breast	diab	digits	dna	german
Spread	0.11	0.69	1.7	1.82	1.02	0.28	1.27	5.27
	letter	pendig	protein	satimage	segment	usps	w7a	w8a
Spread	0.72	0.08	4.95	1.32	0.89	0.41	0.05	0.16

α and then evaluate model performance. Eight datasets considered have train-test splits and we will find the optimal α by searching on a one-dimensional logarithmically equally-spaced grid using 5-fold cross validation (CV) on the training set. Then we will evaluate the performance of optimal α^* model by computing its accuracy on the test set. On the other datasets we will apply 5-fold nested cross validation (CV), defining first a 5-fold outer split and applying again 5-fold CV to estimate the optimal α_i^* over the i -th outer train split. Once this α_i^* is obtained, the associated model is trained over the i -th outer train fold and applied on the patterns remaining on the i -th test fold; these class predictions \hat{y} are then compared with the true target labels y to compute now the accuracy of the model under consideration.

The resulting accuracies are given in Table 2 while Table 3 shows for each problem the model ranking by decreasing accuracies; when two or more give the same accuracy, they receive the same rank. We remark that these rankings are given only for illustration purposes and they do not imply statistically significant differences. This table also shows the mean ranking of each model across all the datasets considered. As it can be seen, the model with the best mean ranking is `ce-fisher`, followed by `ce` and `ce-hinge`. The following one is `ce-hinge-fisher` while `hinge` and `hinge-fisher` perform similarly but behind all others.

These similar performances can also be seen in Table 4, where model rankings are shown after we group together `ce-fisher`, `ce` and `ce-hinge` on a first model group, and `ce-hinge-fisher`, `hinge` and `hinge-fisher` on another. Here `ce-fisher` still performs best on the first group, while in the second `ce-hinge-fisher` and `hinge-fisher` perform similarly and better than `hinge`. In any case, the test accuracies of all models shown in Table 2 are quite similar. This can also be seen in Table 5, which shows for each problem the difference between the highest accuracy (i.e., the best one) and the smallest one (i.e., the worst) as a percentage of the latter. As it can be seen, and except for the `german` and `protein` problems, in all other this percentage is below 2%, and even below 1% in nine problems.

Finally, in Table 6 we give the statistic values returned by the Wilcoxon signed rank test when applied to the columns of the test accuracies in Table 2, and their associated p -values. To obtain them, we have sorted the different losses in increasing order of their rank averages given in the last row of Table 3; this means that in the rows of Table 6 the first model is better ranked than the second one and, hence, expected to perform better. Recall that the test’s null hypothesis is that the two paired samples come from the same distribution.

As it can be seen, this null hypothesis can be rejected at the $p = 0.05$ level when comparing the `ce-fisher` loss with the `ce-hinge-fisher`, `hinge-fisher` and `hinge`, and at the $p = 0.1$ level when doing so with the `ce-hinge` loss; on the other hand, the p -value when comparing it with the `ce` is quite high. This suggests that the `ce-fisher` loss performs similarly to the `ce` loss, but better than the others. In the same vein, the `ce` loss appears to perform better than the `hinge` and `hinge-fisher` ones; all the other loss pairings give similar performances.

Table 6. Model rankings for each problem in ascending accuracies after putting together models with closer rankings.

	stat	p-val
ce-fisher vs ce	62.0	0.776
ce-fisher vs ce-hinge	32.0	0.065
ce-fisher vs ce-hinge-fisher	27.0	0.036
ce-fisher vs hinge-fisher	23.0	0.018
ce-fisher vs hinge	22.0	0.016
ce vs ce-hinge	43.0	0.211
ce vs ce-hinge-fisher	39.0	0.14
ce vs hinge-fisher	24.0	0.024
ce vs hinge	27.0	0.034
ce-hinge vs ce-hinge-fisher	43.0	0.205
ce-hinge vs hinge-fisher	44.0	0.231
ce-hinge vs hinge	46.0	0.266
ce-hinge-fisher vs hinge-fisher	55.0	0.603
ce-hinge-fisher vs hinge	64.0	0.856
hinge-fisher vs hinge	44.0	0.231

3.4 Discussion

The preceding results indicate that the companion losses proposed here can improve on the accuracies of models based on the single `ce` and `hinge` losses. More precisely, adding the Fisher loss results in larger accuracies than those achieved by using just the `ce` and `hinge` ones, although this does not extend to the `ce-hinge` combination (which basically ties with the single `ce` loss), or when combining the `ce-hinge` with the Fisher loss, worsens the `ce-hinge` performance.

It is also to be pointed out that the performance of the `hinge`-based models is worse than that of the `ce`-based ones. A possible reason for this is the relatively small number of units in the single hidden layer architecture of all models. In fact, it is well known that for SVMs to achieve good results, the dimension of the projected input space (upon which an SVM acts linearly) must be quite large. For instance, in the case of the commonly used Gaussian kernel, this dimension essentially coincides with the sample's size. Here we are using the last hidden layer activations as a proxy of the projection space but its dimension is 100, much below sample size for all problems. In fact, in [5] deep networks with at least 1,000 units in the last layer were needed to match or improve the performance of a standard Gaussian SVM. Also, better performance should also be possible with more hidden layers, although this will also help `ce` models. Finally, it is also clear that adding the Fisher loss helps; in fact, it seeks a last hidden layer representation which concentrates each class samples near their centroids while keeping these apart. This should help any classifier acting on that layer while, on the other hand, it doesn't compete directly with `ce` and `hinge`.

4 Conclusions and Futher Work

In this paper we have proposed how to combine different classification losses in a single DNN so that each one acts on specific network outputs. The underlying goal is that these competing but, at the same time, complementary objectives result in models with a performance better than that of those built on each individual loss. We give experimental results on sixteen classification problems combining the categorical cross-entropy and hinge losses as well as a least squares one inspired in Fisher's Discriminant Analysis, and they show that, indeed, such combinations yield better accuracies. In fact, using the Fisher based loss as a companion of the cross-entropy or hinge ones improved on the performance of DNN models using individually those losses; the same happens with the combination of the entropy and hinge losses.

These results suggest that further study is warranted. Beside the obvious extension to more complex network architectures than the simple one here, the choice of the hyperparameters used to define the combined loss (14) should be explored. Moreover, the same strategy of adding companion losses to a base one can be applied to regression problems, where natural choices are the mean square, ϵ -insensitive or Huber losses. We are currently pursuing these and related venues.

References

1. Abadi, M., et al.: TensorFlow: a system for large-scale machine learning. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2016), pp. 265–283 (2016)
2. Chollet, F.: Keras (2015). <https://github.com/fchollet/keras>
3. Crammer, K., Singer, Y.: On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.* **2**, 265–292 (2001)
4. Díaz-Vico, D., Dorronsoro, J.R.: Deep least squares fisher discriminant analysis. *IEEE Trans. Neural Netw. Learn. Syst.* **31**(8), 2752–2763 (2020)
5. Díaz-Vico, D., Prada, J., Omari, A., Dorronsoro, J.R.: Deep support vector neural networks. *Integr. Comput. Aided Eng.* **27**(4), 389–402 (2020)
6. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010. JMLR Proceedings, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010, vol. 9, pp. 249–256 (2010)
7. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings, San Diego, CA, USA, 7–9 May 2015 (2015)
8. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems 25, 26th Annual Conference on Neural Information Processing Systems 2012, Proceedings of a Meeting Held 3–6 December 2012, Lake Tahoe, Nevada, United States, pp. 1106–1114 (2012)
9. Paszke, A., et al.: PyTorch: an imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems, vol. 32. Curran Associates, Inc. (2019)
10. Tang, M., Perazzi, F., Djelouah, A., Ben Ayed, I., Schroers, C., Boykov, Y.: On regularized losses for weakly-supervised CNN segmentation. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 507–522 (2018)
11. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.: Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.* **11**, 3371–3408 (2010)
12. Weston, J., Watkins, C.: Support vector machines for multi-class pattern recognition. In: ESANN 1999, Proceedings of the 7th European Symposium on Artificial Neural Networks, Bruges, Belgium, 21–23 April 1999, pp. 219–224 (1999)
13. Zhang, Z., Dai, G., Xu, C., Jordan, M.I.: Regularized discriminant analysis, ridge regression and beyond. *J. Mach. Learn. Res.* **11**, 2199–2228 (2010)
14. Zhao, H., Gallo, O., Frosio, I., Kautz, J.: Loss functions for image restoration with neural networks. *IEEE Trans. Comput. Imaging* **3**(1), 47–57 (2017)