Pattern Recognition Letters 105 (2018) 144–154

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

Weighted kappa loss function for multi-class classification of ordinal data in deep learning

Jordi de la Torre, Domenec Puig, Aida Valls*

Departament d'Enginyeria Informàtica i Matemàtiques Universitat Rovira i Virgili Avinguda Països Catalans, 26 Tarragona ES-43007 Spain

ARTICLE INFO

Article history:
Available online 18 May 2017

MSC:
62H35
82C32
62M45

Keywords:
Deep learning
Convolutional neural networks
Supervised learning
Computer vision
Diabetic retinopathy
Weighted kappa

ABSTRACT

Weighted Kappa is an index of reference used in many diagnosis systems to compare the agreement between different raters. This index can be also used to evaluate the performance of automatic classification methods against the gold standard given by an expert (or from a consensus of an expert group). On the other hand, in the last years, deep learning has achieved a great importance as a new machine learning method. The usual loss function used in deep learning for multi-class classification is the logarithmic loss. In this paper we explore the direct use of a weighted kappa loss function for multi-class classification of ordinal data, also known as ordinal regression. Three classification problems are solved in the paper using these two loss functions. Results confirm that better classification is made when the model is constructed with the optimization of kappa instead of logarithmic loss.

1. Introduction

Deep Learning is a set of Machine Learning techniques for automatically constructing a model with multiple levels of representation from the underlying distribution of a large set of examples, with the final objective of mapping a high-multidimensional input into a smaller multidimensional output ($f: \mathbb{R}^n \mapsto \mathbb{R}^m, n \gg m$). This mapping allows the classification of multidimensional objects into a small number of categories. The model is composed by many neurons that are organized in layers and blocks of layers, using a cascade of layers in a hierarchical way. Every neuron receives the input from a predefined set of neurons. Every connection has a parameter that corresponds to the weight of the connection. The function of every neuron is to make a transformation of the received inputs into a calculated output value. For every incoming connection, the weight is multiplied by the input value received by the neuron and the aggregated value that used by an activation function that calculates the output of the neuron. The parameters are usually optimized using a stochastic gradient descent algorithm that minimizes a predefined loss function. The parameters of the network are updated after backpropagating the loss function gradients through the network. These hierarchical models are able to learn multiple levels of representation that correspond to different levels of abstraction, which enables the representation of complex concepts in a compressed way [1,2,19,27].

Deep Learning methods have been used extensively in the last years for many automatic classification tasks. For the case of image analysis, the usual procedure consists on extracting the important features with a set of convolutional layers and, after that, make a final classification with these features using a set of fully connected layers. Finally, a soft-max output layer gives as a result the predicted output probabilities of the set of classes predefined in the model. During training the model parameters are changed using a gradient-based optimization algorithm, which minimizes a predefined loss function. For multi-class classification the standardized loss function to use is the logarithmic loss [9].

Once the classifier has been trained (i.e. the parameters of the different layers of the model have been fixed), the quality of the classification outputs predicted by the model are compared against the correct “true” values stored on a labeled dataset. This data is considered as the gold standard, coming from the consensus of the knowledge of a human experts group.

Several quality measures exist in the literature of machine learning and statistics [21]. Kappa index is well-known statistic coefficient defined by Cohen [4] to measure inter-rater agreement on classifying elements into a set of categories (i.e. disjoint classes). Later, Weighted Kappa index was defined for the case of ordered categories (i.e. from best to worst). In this coefficient disagree-

* Corresponding author.
E-mail address: aida.valls@urv.cat (A. Valls).

http://dx.doi.org/10.1016/j.patrec.2017.05.018
0167-8655/© 2017 Elsevier B.V. All rights reserved.

Table 1
Table for interpretation of Weighted Kappa, after [18].

κ	Strength of agreement
< 0.20	Poor
0.21–0.40	Fair
0.41–0.60	Moderate
0.61–0.80	Good
0.81–1.00	Very good

ments are treated differently than in the original Cohen's Kappa [5].

Weighted Kappa index (κ) is used in many medical diagnosis systems because the diseases have different degrees of severity, which are naturally ordered from mild to the most critical cases. If the diagnose is based on image analysis, the classification is even more difficult because in the interpretation of the image data normally is present some level of subjectivity that sometimes makes the conclusions of different experts to differ [14]. Weighted kappa is able to measure the level of discrepancy of a set of diagnosis made by different raters over the same population [30]. Depending on the value of the index, the strength of agreement between the raters can be evaluated (see Table 1).

Examples of the usage of the κ index for measuring inter-rater agreement in the medical context are: the measure of reliability in ultrasound scans interpretation [13], the evaluation of expert agreement in diagnosis of glaucoma [29], the evaluation of reliability of radiographic assessment [11], the inter-observer agreement evaluation in diabetic retinopathy detection [23], among many others.

κ takes into account the ordering of the classes and penalizes the erroneous predictions in function of the distance between the real and the predicted class. In that way, a failure in a prediction that is close to the real category is considered better than a prediction that is farther. The penalization weights depend on the type of the chosen weighted kappa. In a linear penalization the weight is proportional to the distance, in the quadratic weighted kappa (κ) - the one studied in this paper - the penalization is proportional to the square of the distance.

In this paper, we study the direct optimization of the κ index, using it not only as an evaluation function but also as loss function during the training of the model. In the cases where there is some order relation between the output classes, the optimization of the loss function needs to learn this order from the available data to make good predictions. Our hypothesis is that a loss function that encodes such a priori order knowledge can perform better or faster than the usual logarithmic loss function, due to the fact that we are including this additional information known beforehand.

To prove this hypothesis, we use 3 problems of different complexity and with different type of data. Moreover, the neural net model required in each case has an increasing complexity.

The last case that we study, the more complex one, is a diabetic retinopathy image classification problem. Diagnosis of diabetic retinopathy from eye fundus images has been previously studied and it is considered a hard problem, because the classification into levels of severity is based on indicators that are difficult to detect in the images [8,28]. We check its stability and compare the performance of the results obtained from the use of the standard logarithmic loss against the results obtained from the optimization of κ.

The study is organized as follows: in Section 2 we present the deep learning standard method of optimization showing the standardized loss function used for classification, in Section 3 we propose the new cost function for multi-class classification of ordinal data (ordinal regression) with all the mathematical equations required for the optimization using first order gradient descent algorithms, in Section 4 we define the experiments, in Section 5 we present the results obtained and finally in Section 6 we present the conclusions of the paper.

2. Deep learning method

Supervised deep learning techniques are recently used extensively for many automatic classification tasks. In the case of images, most of the state of the art methods are based on the use of deep convolutional neural networks (DCNN): CIFAR-10 [10], CIFAR-100 [3], STL10 [7], SVHN [20], ImageNet [17], among many others. These techniques are focused on learning multiple levels of representation and abstraction that help to make sense of the hidden information in data such as images. In this way, having a complete set of correctly classified images and without any a priori understanding of the features, the model is able to learn the properties of the image that minimize a defined cost function that is direct or indirectly related with the classification score index to optimize.

As shown in Fig. 1 for image classification, several convolutional neural networks are optimized for making a good feature extraction. These layers are followed by one or more classification layers. Finally, the last output layer is formed by as many outputs as classes to predict. In the output layer is usual to have a soft-max function that represents the output probability of every class to predict. Normally, the class assigned to the image is the one with the highest value of probability.

The main parameters to define are: total number of layers, the size of the convolutions with its stride and padding, the activation function to use, the number of convolutional filters for every layer and the type and number of elements of classification layer (fully connected or convolutional).

Moreover, once the architecture of the DCNN has been defined, the neural network has a set of parameters to optimize in order to achieve the most accurate prediction of the data. This is done by optimizing a function of the output variables, called *loss function*. This function depends on the output probabilities given by the model and it is defined in a way that minimizing it, maximizes the probability of the correct class. If the defined function is differentiable with respect to the output variables, then is possible to apply a gradient descent based algorithm to optimize the function [26]. In every optimization step the classification derivative of the loss function is calculated and back-propagated through the network. The parameters are updated according to the optimization algorithm rules in order to reduce the discrepancy between the output of the model and the true value given by the known data.

Although binary classification is possible, many real problems distinguish more than 2 classes, so multi-class assignment procedures are usually required. Multi-class classification is addressed mainly by the optimization of the logarithmic loss (log-loss) function (see Eq. (1)). This function is very easy to optimize using first order gradient descent methods due to the simplicity of its derivatives, its numerical stability and experimental tested validity [9]. Additionally, the logarithmic loss has a very robust probabilistic foundation: minimizing it is the same as minimizing the logarithmic likelihood, that is equivalent to do a Maximum Likelihood Estimation (MLE) or equivalently, to find the Maximum a Posteriori Probability (MAP), given a uniform prior [22]. This function does not encode any prior information about the classes thus, it is a general purpose function that performs well in many applications.

$$\mathscr{L} = \frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} t_i \log y_{i,c} \tag{1}$$

Where:

N: is the number of samples

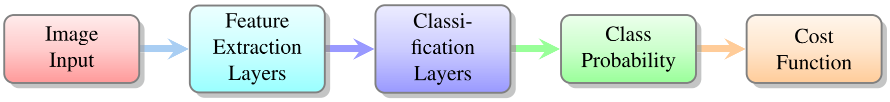

Fig. 1. High level description of a deep learning image classification scheme.

t_i: is 1 for the correct class of the ith sample and 0 elsewhere
C: is the number of classes

On the other hand, there are applications where we have a priori information about some properties of the classes to predict. For example, in the case of ordinal regression, the different categories are sorted in a predefined way, representing a gradation of the output classes. When the log-loss is applied for the classification in this type of problems, the model has to learn such ordering from the data in order to obtain enough accuracy of the classification rate. In this case, a specialized loss like weighted kappa could be more appropriate than a general loss like the logarithm. In the rest of this paper we will study if Weighted Kappa can perform better or even faster, due to the fact that it does not need to learn the order from the available data.

3. Weighted kappa as loss function in deep learning

In this section we present our contribution to the optimization of neural networks in general and deep learning in particular using Weighted Kappa index. Weighted Kappa has been normally used as an index to measure the inter-rating agreement between raters in a multi-class classification problem where the categories have an a priori defined ordering, in such a way that the classes to categorize are a high level abstraction of some sort of intrinsic information that we want to extract from data.

Three matrices are involved in the calculation of this index: the matrix of observed scores, O, the matrix of expected scores based on chance agreement, E, and the weight matrix, ω. The Weighted Kappa is defined as Eq. (2).

$$\kappa = 1 - \frac{\sum_{i,j} \omega_{i,j} O_{i,j}}{\sum_{i,j} \omega_{i,j} E_{i,j}}, \tag{2}$$

where:
C: is the number of classes $i, j \in \{1, 2, \ldots, C\}$
$O_{i,j}$: the number of observations classified in the ith category by the prediction model and they are in the jth category in the correct classification (i.e. “true value”).
$E_{i,j}$: outer product between the two classification histogram vectors (prediction and “true value”), normalized such that E and O have the same sum.
$\omega_{i,j}$: weight penalization for every pair i, j. Generally, $\omega_{i,j} = \frac{(i-j)^n}{(C-1)^n}$. For linear penalization $n = 1$. For quadratic penalization (more commonly used and the studied in this paper): $n = 2$.

This index establishes a penalization when there is a discrepancy between the classifiers that depends on the distance between both predictions. In the case of the κ the penalization of the discrepancy grows quadratically with the distance between the two ratings (i.e. ordered classifications). If the predicted classes of both raters is the same, we say that there is an *absolute* concordance between both raters and no penalization is applied. When the predicted classes are different, we say that there is *relative* concordance between both raters and there is a penalization in the calculation of the inter-rating index that is proportional - in the case of quadratic κ - to the square of the distance between both predictions.

In the numerator of Eq. (2) we take into account the discrepancies between the observed classification of the prediction model and the true assignments. These discrepancies are calculated for all the N items. This penalizing term is normalized dividing their value by the expected discrepancy, obtaining as a result a value between -1 and 1. A value of 1 of the index indicates a perfect agreement between both raters, -1 a perfect symmetric disagreement between the classes and 0 indicates that a random assignment method is used (i.e. no agreement at all).

3.1. Mathematical foundation

In Deep Convolutional Neural Networks and in neural networks in general, the optimization problem to solve during the training of the model is based on finding the values of the parameters for the model that maximize the probability of a correct classification. Thus, if the evaluation index of the correctness of the classification is κ, the optimization problem can be formulated as presented in Eq. (3).

$$\text{maximize} \quad \kappa = 1 - \frac{\sum_{i,j} \omega_{i,j} O_{i,j}}{\sum_{i,j} \omega_{i,j} E_{i,j}}, \quad \text{where} \quad \kappa \in [-1, 1] \tag{3}$$

However, optimization of the loss function ($\mathscr{L}$) is normally presented as a minimization problem, therefore in Eq. (4) we present the same problem converted to minimization. Notice that in this case, we propose to take the logarithm of the index, in order to increase the penalization of incorrect assignments.

$$\text{minimize} \quad \mathscr{L} = \log(1 - \kappa) \quad \text{where} \quad \mathscr{L} \in (-\infty, \log 2] \tag{4}$$

In neural networks for multi-class classification the model constructed does not give a unique predicted class as output, but a probability distribution over the set of possible classes. Consequently, we need to rewrite κ in terms of probability distributions. Having $\kappa = 1 - \mathscr{N}/\mathscr{D}$, in Eq. (5) we show the expression of the numerator $\mathscr{N}$ in terms of the probabilities of the prediction. In Eq. (6) we also show the redefinition of the denominator $\mathscr{D}$ in order to take into account the probabilities given by the model.

$$\mathscr{N} = \sum_{i,j} \omega_{i,j} O_{i,j} = \sum_{k=1}^{N} \sum_{c=1}^{C} \omega_{t_k,c} P_c(X_k) \tag{5}$$

$$\mathscr{D} = \sum_{i,j} \omega_{i,j} E_{i,j} = \sum_{i=1}^{C} \hat{N}_i \sum_{j=1}^{C} \left(\omega_{i,j} \sum_{k=1}^{N} P_j(X_k) \right), \tag{6}$$

where:
X_k: input data of the kth sample $E_{i,j} = \frac{N_i \sum_{k=1}^{N} P_j(X_k)}{N} = \hat{N}_i \sum_{k=1}^{N} P_j(X_k)$
N: number of samples
N_i: number of samples of the ith class $\hat{N}_i = \frac{N_i}{N}$
t_k: correct class number for sample k
$P_c(X_k)$: conditional probability that the kth sample belongs to class c given that the true class is t_k

3.2. Partial derivatives of the weighted kappa loss function

For solving this optimization problem using any gradient descent based algorithm, we need to derive the partial derivatives of the loss function with respect to the output variables of the network.

For the case minimizing the loss function $\mathscr{L} = \log \frac{\mathscr{N}}{\mathscr{D}}$, the derivative takes the next form:

$$\frac{\partial \mathscr{L}}{\partial y_m} = \frac{1}{\mathscr{N}} \frac{\partial \mathscr{N}}{\partial y_m} - \frac{1}{\mathscr{D}} \frac{\partial \mathscr{D}}{\partial y_m} \tag{7}$$

And $\frac{\partial \mathscr{N}}{\partial y_m}$ and $\frac{\partial \mathscr{D}}{\partial y_m}$ can be calculated with the next expressions:

$$\frac{\partial \mathscr{N}}{\partial y_m(X_k)} = \omega_{t_k m} \tag{8}$$

$$\frac{\partial \mathscr{D}}{\partial y_m(X_k)} = \sum_{i=1}^{C} \hat{N}_i \omega_{i,m} \tag{9}$$

Where: $m \in \{1, 2, \ldots, C\}$

In array form it can be rewritten as:

$$\frac{\partial \mathscr{N}}{\partial y_m} = \begin{pmatrix} \omega_{t_1,1} & \omega_{t_1,2} & \cdots & \cdots & \omega_{t_1,C} \\ \omega_{t_2,1} & \omega_{t_2,2} & \cdots & \cdots & \omega_{t_2,C} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \omega_{t_N,1} & \omega_{t_N,2} & \cdots & \cdots & \omega_{t_N,C} \end{pmatrix} \tag{10}$$

$$\frac{\partial \mathscr{D}}{\partial y_m} = \begin{pmatrix} \sum_{i=1}^{C} \hat{N}_i \omega_{1,i} & \cdots & \cdots & \cdots & \sum_{i=1}^{C} \hat{N}_i \omega_{C,i} \\ \sum_{i=1}^{C} \hat{N}_i \omega_{1,i} & \cdots & \cdots & \cdots & \sum_{i=1}^{C} \hat{N}_i \omega_{C,i} \\ \cdots & \vdots\vdots\vdots & \cdots & \cdots & \cdots \\ \sum_{i=1}^{C} \hat{N}_i \omega_{1,i} & \cdots & \cdots & \cdots & \sum_{i=1}^{C} \hat{N}_i \omega_{C,i} \end{pmatrix} \tag{11}$$

With the definition of the loss function and its derivatives we have all the equations required to apply any first order optimization algorithm on a neural network based learning method.

4. Experiments

The aim of the experiments of this paper is to test the performance of the proposed optimization loss function. Although our main area of interest is medical image analysis, we present two other completely different classification problems in order to prove the generality of the improvement of the performance when optimizing the qwk-loss function for solving ordinal regression problems. Three real case studies are solved, which are presented in order of increasing model complexity. In *Case 1* the relevance of the results provided by eCommerce search engines is estimated using a linear classifier; in *Case 2* the level of expectancy of contracting a life insurance is predicted using a fully connected 3-layer perceptron; finally, in *Case 3*, the most complex, a deep convolutional neural network is trained to solve a diabetic retinopathy image classification problem. All three are real-world problems that are proposed as challenges in different competitions hosted in the Kaggle Platform.[1] These problems were chosen because in all three the index used for evaluation in the Kaggle challenge is κ. It is worth to note that the neural network models presented here may not necessarily be the best ones for solving the concrete case studies. In some cases it is better another approach, however we study and present the neural network model because the contribution of this paper is for this learning method.

4.1. Case 1. Search results relevance

4.1.1. Problem definition

Many electronic commerce sites have a search engine that helps the user to find the most suitable products. Search algorithms are developed ad-hoc for each site. Sometimes, the user experience with the results provided by these systems is discouraging, which may lead to a poor use of the online shop. Currently, small online businesses have no good way of evaluating the performance of their search algorithms, making it difficult for them to provide a good customer experience. The objective of this problem is to measure the relevance of the results reported by a search engine. Given the queries entered by customers and the resulting product descriptions reported by the search engine, the model has to predict the relevance that customers will give to the results obtained from the search engine. Such relevance is classified in four different categories sorted from not relevant (1) to very relevant (4).

4.1.2. Data

The dataset was created using query-result pairings from the CrowdFlower platform. The training data set includes 41,327 records with the next fields: product identification, query, product description, median relevance reported by three different raters as a value between 1 and 4 and the relevance variance of the scores given by the raters. The test data set includes 256,735 records with the next fields: product identification, user query and a product description. The dataset comes from the *Crowdflower Search Results Relevance* competition hosted in the Kaggle Platform.

4.1.3. Procedure

The training set has been split into two random subsets: 85% of the data is used for training and 15% is used for validation. As a pre-processing step, TF-IDF algorithm [24] is applied to the query and product fields of the training set in order to make a first feature extraction. A singular value decomposition (SVD) of the features is applied and truncated to the first 400 components. After centering and scaling, the 400 components are feeded to a linear classifier that is trained using the Adam optimizer [16] either with the log-loss or the qwk-loss function.

4.2. Case 2. Life insurance assessment

4.2.1. Problem definition

The goal of this classification problem is to rate the customers of an insurance company in eight categories in function of the expectancy for the customer of contracting a life insurance, from low expectancy (class 1) to high expectancy (class 8). The USA company Prudential helps people of all ages and backgrounds grow and protect their wealth through a variety of products and services, including life insurance. The dataset includes the personal, medical and commercial information that this company has gathered from the customers.

4.2.2. Data

A database with 128 categorical, discrete and continuous variables is available with personal information, contracted products, medical history and family history of customers. The training set consist of 59,381 records. The test set consist of a total of 19,765 records. Some of the variables are missing. For every training record a class label is reported. The dataset comes from the Prudential Life Insurance Assessment competition hosted in the Kaggle Platform.

4.2.3. Procedure

Training set has been split into two random subsets: 85% of the data is used for training and 15% is used for validation. As a preprocessing step, all the categorical variables are converted to dummy variables. The empty values are filled with the mean value and finally all the dataset is normalized and scaled to have zero mean and unit standard deviation. After this, the initial 128 input variables are converted into 1,078 variables. These new variables are the input of a 3-layer fully connected multilayer perceptron. The first hidden layer has 128 units, the second one 64 and the output one 8. The two hidden layers use a ReLU activation function and are preceded by a batchnormalization layer. The model is

[1] https://www.kaggle.com/competitions.

trained using the Adam optimizer either with the log-loss or the qwk-loss functions.

4.3. Case 3. Diabetic retinopathy detection

4.3.1. Problem definition

Diabetic Retinopathy (DR) is a leading disabling chronic disease and one of the main causes of blindness and visual impairment in developed countries for diabetic patients. Studies reported that 90% of the cases can be prevented through early detection and treatment [25]. Eye screening through retinal images is used by physicians to detect the lesions related with this disease. Due to the increasing number of diabetic people, the amount of images to be manually analyzed is becoming unaffordable. Moreover, training new personnel for this type of image-based diagnosis is long, because it requires to acquire expertise by daily practice.

In 2003 the medical community established a standardized classification based on four severity stages [31] determined by the type and number of lesions (as micro-aneurysms, hemorrhages and exudates): class 0 referring to no apparent retinopathy, class 1 as a Mild Non-Proliferative Diabetic Retinopathy (NPDR), class 2 as Moderate NPDR, class 3 as a Severe NPDR and class 4 as a Proliferative DR.

The problem consists on optimizing a deep convolutional neural network to maximize the classification rate with a test set of images never seen before. The generalization capability will be scored against the quadratic weighted kappa over the test set.

4.3.2. Data

The dataset used in this work consists of two independent high resolution image sets (train and test). For every patient right and left eye images are reported. All the images are classified by ophthalmologists according to the standard severity scale presented before in [31]. The images are taken in variable conditions: by different cameras, illumination conditions and resolutions.

The training set contains a total of 35,126 images; 25,810 of class 0, 2443 of class 1, 5292 of class 3, 873 of class 3 and 708 of class 4. The test set contains a total of 53,576 images; 39,533 of class 0, 3,762 of class 1, 7,861 of class 2, 1,214 of class 3 and 1,206 of class 4.

The images come from the EyePACS dataset used in a Diabetic Retinopathy Detection competition hosted on the internet Kaggle Platform.

4.3.3. The models

Four different models have been used, one for every different image size. The idea was to first design small models based on restricted image sizes in order to have a small training time that could allow to run the maximum amount of experiments. This big set of previous experiments would allow to restrict the hyper-parameter space. Bigger models, slow to train, would then be trained with the best selection of hyper-parameters. For image sizes will be used: 128 × 128, 256 × 256, 384 × 384 and 512 × 512. The high resolution images of the dataset are resized to this values prior the training.

All models follow the scheme defined in Fig. 1. The feature extraction layers use all 3 × 3 convolutions with padding of 1 and stride of 1, followed by a batch normalization [15] and a ReLU activation function [6]. Every two feature layers, a 2 × 2 max-pooling layer of stride 2 is applied for dimensionality reduction. The number of feature layers vary depending on the image size. All models use a unique classification layer followed by the final output softmax layer. The number of layers and the total number of parameters of each model are summarized in Table 5.

The model used for the 128 × 128 image case has 1.16 million parameters, 10 feature layers of 32/32, 64/64, 128/128, 128/128 filters in every convolution (/ separate the filters for every map size, the commas indicate a 2 × 2 max-pooling operation) and a 4 × 4 convolution of 128 filters as a classification layer. The model used for the 256 × 256 image case has 1.44 million parameters, 12 feature lavers of 32/32, 64/64, 128/128, 128/128, 128/128 filters in every convolution and a 4 × 4 convolution of 128 filters as a classification layer. The model used for the 384 × 384 image case has 1.77 million parameters, 12 feature lavers of 32/32, 64/64, 128/128, 128/128, 128/128 filters in every convolution and a 6x6 convolution of 128 filters as a classification layer. Finally, the model used for the 512 × 512 image case has 11.3 million parameters, 12 feature lavers of 16/16, 32/32, 64/64, 128/128, 256/256, 512/512 filters in every convolution, a 5 × 5 convolution of 512 filters as classification layer, followed by a 4 × 4 average pooling and a dropout layer (ratio = 0.3) previous to the final soft-max output layer.

4.3.4. Procedure

The original training set is split into two random subsets: one with 90% of the data and other with 10%. The last one is used as a validation set for hyper-parameter selection.

Notice that the image set is highly imbalanced. In order facilitate the learning, the training set is artificially equalized using data augmentation techniques [17] based on 0–180° random rotation, X and Y mirroring and contrast and brightness random sampling.

A random initialization based in the Kaiming&He approach [12] is used for all the networks. All models are optimized using a batch based first order optimization algorithm called Adam [16]. We study different learning rates in order to find the optimal one for each loss function.

As qwk-loss function uses a normalization term in the denominator, we will use a batch-based gradient calculation. Presumably this loss function will be more sensible to small batches that the log-loss, this is the reason for considering the batch size (BS) an important hyper-parameter to study.

For every batch, the images are chosen randomly from the training set, with repetition. Data augmentation techniques are applied to augment the diversity of the classes (random rotations and brightness and contrast modifications). The epoch size is set to maintain fixed the total number of images per epoch to 100.000. This value is approximately the number of sample images required to ensure that all of them have been selected every epoch. Additionally, the number of updates per epoch of the parameters of the neural network is changed for different batch sizes. In the case of small batch sizes the number of updates per epoch is greater than the case of bigger batch sizes. Studying different batch sizes we would also explore different number of parameter updates. In all experiments training were run for 100 epochs.

5. Results

In this section we present the results of the experiments. For each case, we present a table with the detailed experiments done training the model with the two loss functions with different parameters. Additionally, a figure is presented with the results obtained in the best model achieved in the training stage, for every batch size and loss function.

5.1. Case 1. Search results relevance

For the selection of the best hyper-parameters a grid search is done over the learning rate (LR) and the batch size (BS) of the model. In Table 2 we show all the experiments. In every row of the table we show the results obtained from training the model either with the log-loss and with the qwk-loss functions.

In Fig. 2 we show the best results achieved for every batch size with the validation set for the best hyper-parameter selection. As

Table 2
List of all experiments for Case 1: search results relevance.

ID	BS	LR	$\kappa_{val}^{qwk-loss} 10^3$	$Epoch_{best}^{qwk-loss}$	$\kappa_{val}^{log-loss} 10^3$	$Epoch_{best}^{log-loss}$
1	4	1e−02	405	10	439	9
2		1e−03	485	6	476	6
3		5e−04	**506**	15	**483**	14
4		1e−04	503	19	467	3
5		1e−05	461	99	455	50
6	8	1e−02	450	13	444	6
7		1e−03	**544**	6	**482**	0
8		5e−04	524	9	475	18
9		1e−04	535	31	482	10
10		1e−05	445	99	479	50
11	16	1e−02	504	18	425	3
12		1e−03	535	10	480	1
13		5e−04	543	11	**480**	1
14		1e−04	**545**	50	483	9
15		1e−05	395	99	479	86
16	32	1e−02	515	12	436	2
17		1e−03	548	16	494	1
18		5e−04	**548**	17	**508**	4
19		1e−04	519	60	501	17
20		1e−05	385	99	485	99
21	64	1e−02	520	10	473	8
22		1e−03	531	12	488	4
23		5e−04	**545**	37	506	9
24		1e−04	514	61	**507**	25
25		1e−05	377	99	400	99
26	128	1e−02	513	4	479	5
27		1e−03	**544**	23	497	6
28		5e−04	527	31	499	8
29		1e−04	503	72	**509**	55
30		1e−05	317	99	0	0
31	256	1e−02	510	8	490	1
32		1e−03	**552**	32	**516**	14
33		5e−04	490	21	507	24
34		1e−04	421	45	469	48
35		1e−05	189	99	61	34
36	512	1e−02	512	18	488	7
37		1e−03	**531**	38	**516**	24
38		5e−04	509	64	493	27
39		1e−04	443	99	450	86
40		1e−05	15	4	41	43

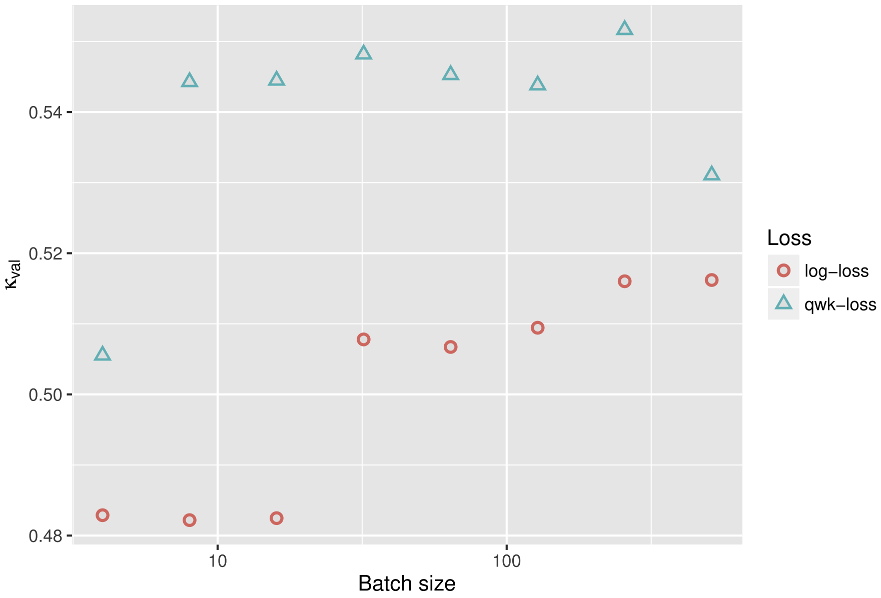

Fig. 2. κ_{val} of the best model for every batch size (case 1: search results relevance).

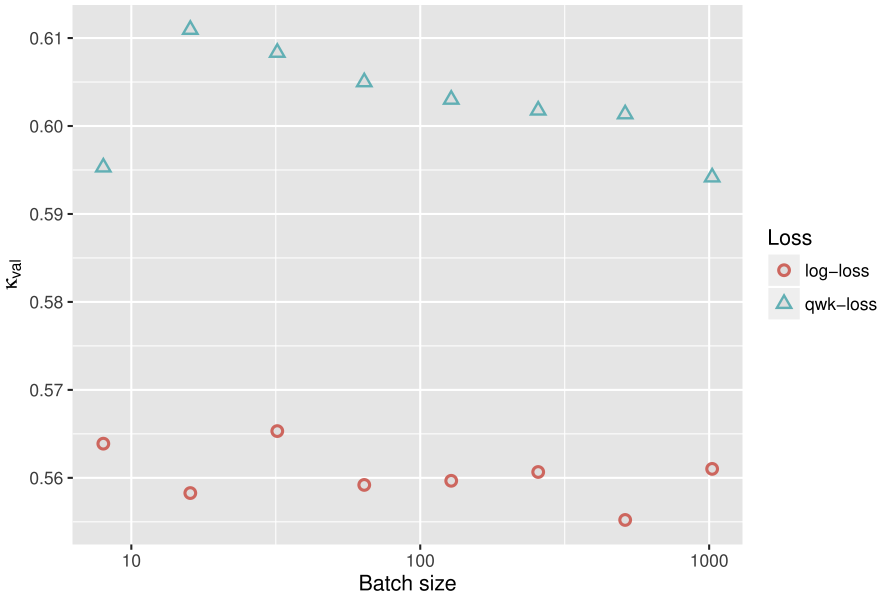

Fig. 3. κ_{val} of the best model for every batch size (case 2: life insurance assessment).

we can see, the models trained with qwk-loss function perform consistently better in all the configurations.

The κ_{val} indicator is always higher for the model that optimizes qwk-loss function. This indicates that the class order information has been correctly introduced into the model. Therefore, the classification mistakes are done within neighbor categories and not within separated categories (which is highly penalized with the quadratic weighted kappa index).

After the study, we check the performance of the best model trained with qwk-loss and log-loss, using the test set (i.e. against never seen before records) consisting on 256,735 records. The test of the best model trained with qwk-loss reports a $\kappa_{test} = 0.500 \pm 0.043$ (95% confidence). The best model trained with log-loss achieves a kappa in the test set of $\kappa_{test} = 0.468 \pm 0.050$ (95% confidence). The difference between the two models is about a 6% increase of the value of κ_{test} (see Fig. 6).

5.2. Case 2. Life insurance assessment

In Fig. 3, the best results achieved on the training stage for every batch size and learning rate are shown. For each batch size, 5 LR values are tested (Table 3). In bold we have the best κ_{val} of each batch size. This are graphically displayed in Fig. 3. Next, the best parameters are chosen for the two models and the performance is evaluated using the test set, consisting on 19,765 records. The test of the best model trained with qwk-loss reports a $\kappa_{test} = 0.618 \pm 0.016$ (95% confidence). In the case of the best model trained with log-loss the value of kappa in the test set is $\kappa_{test} = 0.562 \pm 0.018$ (95% confidence). A significant difference between the two best models in the test set is found, with about a 10% increase of the value of κ_{test} (see Fig. 6).

5.3. Case 3. Diabetic retinopathy detection

Table 4 shows the collection of all the conducted experiments where *input* represents the size of the input layer, *BS* the batch size used in the experiment, *LR* the learning rate, κ_{train} the maximum value of κ achieved during training over the training set, κ_{val} the maximum value of κ achieved during training over the validation set, *gap* the difference between κ_{train} and κ_{val}, *epoch* the epoch where the maximum value of κ_{val} is achieved and finally *updates* the number of parameters updates required to achieve the maximum value of κ_{val}.

We can see that for every input size the maximum value achieved over the training set optimizing qwk-loss is always higher than the maximum value achieved optimizing log-loss (bold values). The gaps between both training and validation are also lower in the case of qwk-loss indicating a lower overfitting of the training data. We see that qwk-loss consistently reports better results than log-loss except for very low values of BS. In those cases, the qwk-loss performs worse than log-loss, but even in those cases the maximum value achieved by log-loss is lower than the best value of qwk-loss for the same model over the whole set of experiments.

In Fig. 4 we show a graphical representation of the maximum value of κ_{val} for the different configurations. We can see that directly optimizing qwk-loss gives consistently better results than optimizing log-loss. Only in the case of very small batch sizes (for this application case, $BS = 5$) log-loss performs better than qwk-loss. This is probably due to the fact that qwk-loss uses a normalization term in the denominator that with not big enough batch sizes could cause instabilities in the gradient that affect the performance. In any case, even in those cases the results obtained with the log-loss are worse that the best achieved using the qwk-loss configuration.

The optimal batch size for solving this diabetic retinopathy classification is of 5 in the case of using the log-loss and 15 in the case of the qwk-loss. For greater values of this hyper-parameter, the increase in precision of the calculation of the gradient does not report any advantage in the optimization. In the case of the log-loss a lower precision in the calculation of the gradient works better. It could probably be due to the fact that this imprecision in the calculation increases the stochasticity of the optimization and due to the fact that we are not optimizing directly the metrics index. This makes possible to explore adjacent zones that could work better for improving the metrics index than the ones that specifically improve the log-loss. Additionally, although smaller batch sizes give worse approximations of the gradient, the number of updates per epoch of the parameters of the model is greater. This seems to be an advantage in this case.

After all this study, we check the performance of the best model trained with qwk and log losses against the *never seen before* 53,576 image test set. The test of the best model trained with qwk-loss reports a $\kappa_{test} = 0.740 \pm 0.006$ (95% confidence). In

Table 3
List of experiments for Case 2: life insurance assessment.

ID	BS	LR	$\kappa_{val}^{qwk-loss}10^3$	$Epoch_{best}^{qwk-loss}$	$\kappa_{val}^{log-loss}10^3$	$Epoch_{best}^{log-loss}$
1	8	1e−02	412	0	564	5
2		1e−03	593	3	550	4
3		5e−04	592	8	556	2
4		1e−04	**595**	11	**560**	4
5		1e−05	588	20	556	27
6	16	1e−02	570	1	555	8
7		1e−03	605	5	550	4
8		5e−04	**611**	7	**558**	2
9		1e−04	605	7	558	9
10		1e−05	596	14	554	28
11	32	1e−02	587	0	560	1
12		1e−03	604	2	**565**	3
13		5e−04	**608**	2	552	4
14		1e−04	603	11	562	11
15		1e−05	598	24	545	44
16	64	1e−02	587	0	547	7
17		1e−03	**605**	3	557	5
18		5e−04	603	4	**559**	4
19		1e−04	600	5	557	8
20		1e−05	595	38	542	30
21	128	1e−02	592	2	554	4
22		1e−03	602	5	**560**	2
23		5e−04	**603**	4	556	8
24		1e−04	599	7	558	18
25		1e−05	567	48	537	45
26	256	1e−02	589	6	555	1
27		1e−03	**602**	3	555	2
28		5e−04	598	6	555	1
29		1e−04	592	6	**561**	20
30		1e−05	589	49	515	26
31	512	1e−02	**601**	16	**555**	1
32		1e−03	601	5	548	5
33		5e−04	598	3	554	9
34		1e−04	598	15	548	19
35		1e−05	563	49	523	48
36	1024	1e−02	594	4	549	3
37		1e−03	592	5	**561**	7
38		5e−04	**594**	5	554	12
39		1e−04	593	21	547	22
40		1e−05	472	49	470	49

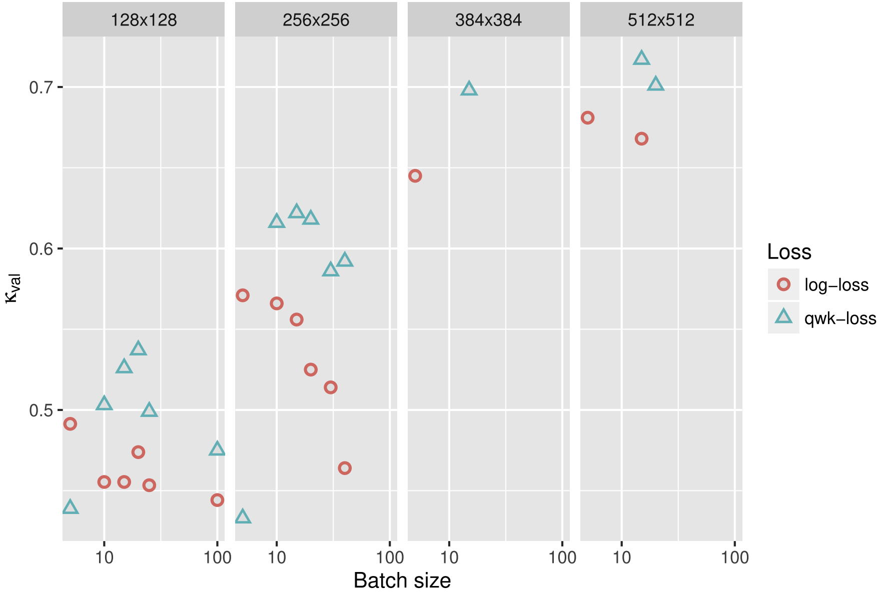

Fig. 4. κ_{val} achieved for the diabetic retinopathy detection use case in every experiment in function of the batch size and the loss function used.

Table 4
List of experiments for Case 3: diabetic retinopathy detection.

Input	BS	Loss	LR	$\kappa_{train} 10^3$	$\kappa_{val} 10^3$	Gap	Epoch	Updates 10^{-3}
128	5	log	10^{-5}	771	418	353	78	1560
			10^{-4}	851	**491**	360	73	1460
			10^{-3}	676	418	258	29	580
		qwk	5×10^{-5}	545	402	143	50	1000
			10^{-5}	646	439	207	70	1400
			10^{-4}	497	326	171	31	620
	10	log	10^{-5}	797	397	400	82	820
			10^{-4}	874	455	419	81	810
			10^{-3}	514	336	178	57	570
		qwk	10^{-5}	774	476	298	82	820
			10^{-4}	755	503	252	84	840
			10^{-3}	596	289	307	95	950
	15	log	10^{-5}	803	368	435	79	527
			10^{-4}	899	458	441	95	633
			10^{-3}	868	447	421	80	533
		qwk	5×10^{-5}	715	491	224	77	513
			10^{-4}	800	526	274	77	513
			5×10^{-4}	823	523	300	72	480
	20	log	10^{-4}	896	474	422	79	395
		qwk	10^{-4}	835	**537**	298	93	465
	25	log	10^{-5}	821	315	506	96	384
			10^{-4}	913	453	460	93	372
			10^{-3}	849	382	467	70	280
		qwk	10^{-5}	808	423	385	95	380
			10^{-4}	824	499	325	65	260
			10^{-3}	655	447	208	80	320
	100	log	10^{-4}	929	377	552	98	98
			10^{-3}	947	444	503	99	99
			10^{-2}	842	412	430	67	67
		qwk	10^{-4}	879	450	429	93	93
			10^{-3}	798	455	343	71	713
			10^{-2}	–	–	–	–	–
256	5	log	10^{-4}	871	**571**	300	52	1040
		qwk	10^{-4}	605	433	172	15	300
	10	log	10^{-4}	903	566	337	75	750
		qwk	10^{-4}	832	616	216	70	700
	15	log	10^{-4}	925	556	369	98	653
		qwk	10^{-4}	878	**622**	256	93	620
	20	log	10^{-4}	923	525	398	97	485
		qwk	10^{-4}	891	618	273	97	485
	30	log	10^{-4}	925	514	411	93	310
		qwk	10^{-4}	900	586	314	98	327
	40	log	10^{-4}	922	464	458	93	233
		qwk	10^{-4}	894	592	302	78	195
384	5	log	10^{-4}	863	**641**	222	38	760
	15	qwk	10^{-4}	889	**698**	191	93	620
512	5	log	10^{-4}	980	**681**	299	88	1760
	15	log	10^{-4}	978	668	310	94	626
		qwk	10^{-4}	884	**717**	167	86	573
	20	qwk	10^{-4}	903	701	202	89	445

Table 5
Summary of the difference in performance between qwk-loss and log-loss trained models in function of input size for the Diabetic retinopathy detection case.

Input	Total layers	Feature layers	Classific. layers	Params 10^{-6}	$\kappa_{val}^{qwk-loss}$	$\kappa_{val}^{log-loss}$	Δ
128 × 128	12	10	1	1.16	0.537	0.491	9.3%
256 × 256	14	12	1	1.44	0.622	0.571	8.9%
384 × 384	14	12	1	1.77	0.698	0.663	5.3%
512 × 512	14	12	1	11.3	0.717	0.681	5.3%

the case of the same model trained with log-loss the value of $\kappa_{test} = 0.686 \pm 0.008$ (95% confidence). The difference between the two best models in the test set is of more than a 7% increase of the value of κ. (see case 3 in Fig. 6).

Fig. 5 helps to understand the difference between the performance of the optimized loss functions. This figure displays the histograms of the predicted classes for every true class. It can be seen how the predicted histograms of the log-loss trained model are more scattered than the ones of the qwk-loss trained model. In those cases where there is a discrepancy between the real value and the predicted one, the model trained with the qwk-loss assign a category that are closer to the true category than the ones predicted by the log-loss trained model. This can be seen specially in the central category T2, where the distribution of qwk-loss model is concentrated in categories 1, 2 and 3 (with around 30% each), while in the log-loss model, there are 25% observations classified into class 0 as well as 10% into class 4. This is of great importance when the classification is related with medical diagnosis. For a patient having severe retinopathy (class 4) it is better to be classified having a moderate or a proliferative one (closer classes) than to be classified as having a mild or as not having any retinopathy at all.

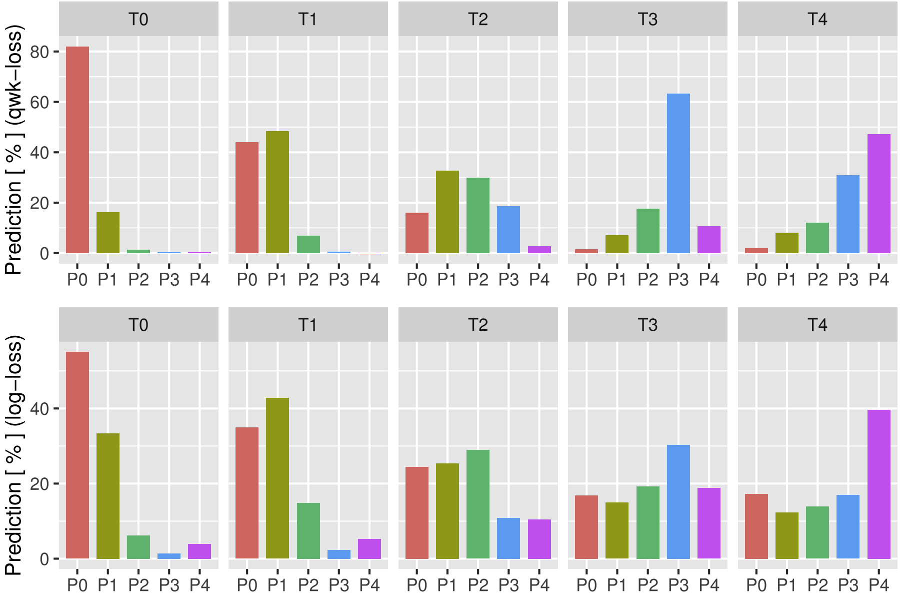

Fig. 5. Histograms of the predicted classes for every real class over the test set for the best qwk-loss (above) and log-loss (below) trained models in the diabetic retinopathy multi-class classification use case.

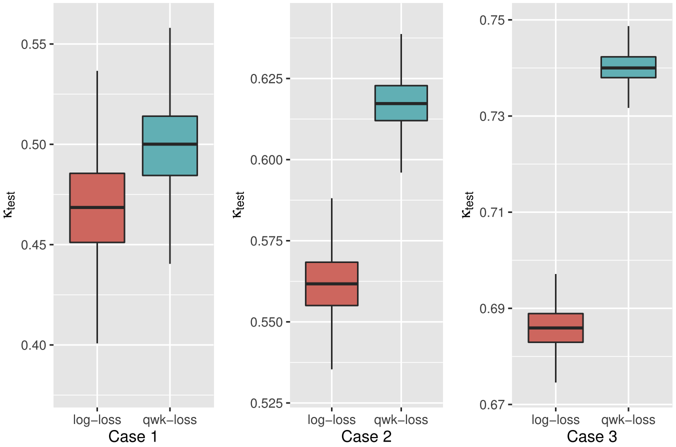

Fig. 6. κ_{test} confidence intervals for the log and qwk best models for the three studied cases.

5.4. Overall discussion on the performance improvement

In this section, the quality of the classification with the testing datasets is analyzed, although a brief comment has been made before on each case study. Fig. 6 shows the κ index obtained on the testing data for the three case studies with the best model of every loss function. With this figure we can check if the difference between the two optimization techniques is statistically significant or not. Improvement is clear for Case 2 and Case 3. Case 1 also has almost non-overlapping boxplots but the κ_{test} value achieved is more similar in the two models. However, in this case study the neural network used was the simplest one, thus the influence of the loss function in the final model is less. Improvement achieved with qwk-loss optimization is, thus, clear and supports the initial hypothesis of this research work.

6. Conclusions

We presented a new loss function for training deep learning models in ordinal classification problems based on the optimization of the weighted kappa index. In contrast to the logarithmic loss that uses a uniform prior over the set of classes, this new loss function defines a penalization over the discrepancy that is proportional to a power of the distance (quadratically in the case of κ) that allows to encode the prior known information about the predefined ordering of the classes.

We checked the performance of this new loss function with three different real-world case studies with diverse types of input data: textual in the first, categorical and numerical in the second and images in the third. Moreover, each case study was solved using decision models of increasing level of complexity: a linear classifier in the first, a multilayer perceptron in the second and a deep convolutional neural network in the third.

The results presented in this paper show that with the direct optimization of the κ index consistently better generalization results can be achieved than with the standard use of the logarithmic loss. Log-loss has to learn the predefined ordering of the classes from data and this seems to be a disadvantage. Results showed that, depending on the use case, between 6% and 10% of improvement of κ scores can be obtained from the direct optimization of the function.

This is a significant improvement that may be worth in many domains, such as the one of medical diagnosis, since an accurate detection of the level of severity of a disease usually has great influence on the treatment prescription and the possibility of minimizing bad consequences of the illness.

One minor drawback of the new loss is its low performance with very small batch sizes in the image classification study. The experiments show that for the retinopathy classification problem with batch sizes of 5, the performance of the function is lower than using the logarithmic loss. This parameter is for sure problem dependent and has to be taken into account as an important parameter to check in other image classification tasks that use a deep neural network.

Acknowledgments

This work is supported by the URV grants 2014PFR-URV-B2-60 and 2015PFR-URV-B2-60, as well as, for the Spanish research projects PI15/01150 and PI12/01535 (Instituto de Salud Carlos III). The authors would like to thank to the Kaggle, Crowdflower, Prudential and EyePACS for providing the data used in this paper.

References

[1] Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (2009) 1–127. Also published as a book. Now Publishers, 2009.
[2] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (2013) 1798–1828.
[3] D. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (elus), CoRR abs/1511.07289(2015).
[4] J. Cohen, A coefficient of agreement for nominal scales, Educ. Psychol. Meas. 20 (1960) 37–46.
[5] J. Cohen, Weighted kappa: nominal scale agreement provision for scaled disagreement or partial credit., Psychol. Bull. 70 (1968) 213.
[6] G.E. Dahl, T.N. Sainath, G.E. Hinton, Improving deep neural networks for LVCSR using rectified linear units and dropout, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2013, pp. 8609–8613.
[7] A. Dundar, J. Jin, E. Culurciello, Convolutional clustering for unsupervised learning, CoRR abs/1511.06241(2015).
[8] J. Escorcia-Gutierrez, J. Torrents-Barrena, P. Romero-Aroca, A. Valls, D. Puig, Interactive optic disk segmentation via discrete convexity shape knowledge using high-order functionals, in: Artificial Intelligence Research and Development - Proceedings of the 19th International Conference of the Catalan Association for Artificial Intelligence, Barcelona, Catalonia, Spain, October 19–21, 2016, 2016, pp. 39–44.
[9] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[10] B. Graham, Fractional max-pooling, CoRR abs/1412.6071(2014).
[11] K.P. Günther, Y. Sun, Reliability of radiographic assessment in hip and knee osteoarthritis, Osteoarthritis Cartilage 7 (1999) 239–246.
[12] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: surpassing human-level performance on imagenet classification, in: 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1026–1034.
[13] S.R. Hintz, T. Slovis, D. Bulas, K.P. Van Meurs, R. Perritt, D.K. Stevenson, W.K. Poole, A. Das, R.D. Higgins, N.N.R. Network, et al., Interobserver reliability and accuracy of cranial ultrasound scanning interpretation in premature infants, J. Pediatr. 150 (2007) 592–596.
[14] G. Hripcsak, D.F. Heitjan, Measuring agreement in medical informatics reliability studies, J. Biomed. Inform. 35 (2002) 99–110.
[15] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, CoRR abs/1502.03167(2015).
[16] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, CoRR abs/1412.6980(2014).
[17] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: F. Pereira, C. Burges, L. Bottou, K. Weinberger (Eds.), Advances in Neural Information Processing Systems 25, Curran Associates, Inc., 2012a, pp. 1097–1105.
[18] J.R. Landis, G.G. Koch, The measurement of observer agreement for categorical data, Biometrics 33 (1) (1977) 159–174.
[19] Y. Lecun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.
[20] Z. Liao, G. Carneiro, Competitive multi-scale convolution, CoRR abs/1511.05635(2015).
[21] N. Mehdiyev, D. Enke, P. Fettke, P. Loos, Evaluating forecasting methods by considering different accuracy measures, Procedia Comput. Sci. 95 (2016) 264–271.
[22] K.P. Murphy, Machine Learning: A Probabilistic Perspective, The MIT Press, 2012.
[23] S. Patra, E. Gomm, M. Macipe, C. Bailey, Interobserver agreement between primary graders and an expert grader in the bristol and weston diabetic retinopathy screening programme: a quality assurance audit, Diabetic Med. 26 (2009) 820–823.
[24] J. Ramos, Using TF-IDF to Determine Word Relevance in Document Queries, in: Technical Report, Department of Computer Science, Rutgers University, 23515 BPO Way, Piscataway, NJ, 2003, p. 08855e.
[25] P. Romero-Aroca, J. Fernández-Ballart, M. Almena-Garcia, I. Méndez-Marín, M. Salvat-Serra, J.A. Buil-Calvo, Nonproliferative diabetic retinopathy and macular edema progression after phacoemulsification: prospective study, J. Cataract Refractive Surg. 32 (2006) 1438–1444.
[26] D. Saad, Online algorithms and stochastic approximations, Online Learn. 5 (1998).
[27] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Networks 61 (2015) 85–117.
[28] J. de la Torre, A. Valls, D. Puig, Diabetic retinopathy detection through image analysis using deep convolutional neural networks, in: Artificial Intelligence Research and Development - Proceedings of the 19th International Conference of the Catalan Association for Artificial Intelligence, Barcelona, Catalonia, Spain, October 19–21, 2016, 2016, pp. 58–63.
[29] R. Varma, W.C. Steinmann, I.U. Scott, Expert agreement in evaluating the optic disc for glaucoma, Ophthalmology 99 (1992) 215–221.
[30] A.J. Viera, J.M. Garrett, et al., Understanding interobserver agreement: the kappa statistic, Fam. Med. 37 (2005) 360–363.
[31] C. Wilkinson, F. Ferris 3rd, R.E. Klein, P.P. Lee, C.D. Agardh, M. Davis, D. Dills, A. Kampik, R. Pararajasegaram, J.T. Verdaguer, Global diabetic retinopathy project group. proposed international clinical diabetic retinopathy and diabetic macular edema disease severity scales, Ophthalmology 110 (2003) 1677–1682.