Classification of ordinal data in deep learning: an experimental study

Víctor-Manuel Vargas, Pedro-Antonio Gutiérrez and César Hervás

***Abstract*—This paper proposes a deep convolutional neural network model for ordinal regression based on the use of a family of probabilistic ordinal link functions in the output layer called Cumulative Link Models. This kind of models projects each pattern into a 1-dimensional space. In this case, the projections are estimated by a non-linear deep neural network. A set of ordered thresholds splits this space into the different classes of the problem. Different link functions will be studied in this experimental study, and the results will be contrasted with statistical analysis. To further improve the results, we combine these ordinal models with a loss function that takes into account the distance between the categories, based on the weighted Kappa index. The experiments run over two different ordinal classification problems, and the statistical tests confirm that these models improve the results of a nominal model and outperform the proposals considered in the literature.**

***Index Terms*—Deep learning, Ordinal Regression, Link models.**

I. Introduction

DEEP LEARNING, introduced by Yann Lecun [1], combines multiple machine learning techniques and allows computational models that are composed of numerous processing layers to learn representations of data with various levels of abstraction. These methods have dramatically improved the state-of-the-art in many domains, such as image classification [2], [3], [4], speech recognition [5], control problems [6], object detection [7], [8], privacy protection [9], recovery of human pose [10], semantic segmentation [11] and image retrieval [12]. Convolutional Neural Networks (CNN) are one of the types of deep networks that are designed to process data that comes in the form of multiple arrays. CNNs are appropriate for images, video, speech and audio processing, and they have been used extensively in the last years for automatic classification tasks [13], [14], [15]. On image classification tasks, each colour channel is represented by a 2D array. In this case, convolutional layers extract the main features from the pixels of the images and, after that, a fully connected layer classify every sample based on its extracted features. At the output of the CNN, a softmax function provides the probabilities of the set of classes predefined in the model for classification tasks. These outputs are compared against the correct values.

Ordinal classification problems are those classification tasks where labels are ordered, and there are different inter-classes importances for each pair of classes. This kind of problem can be treated as a nominal classification problem, but doing this discards the ordinal information. A better approach is to use specific methods that take into account this kind of information to improve the performance of the classification model. The Proportional Odds Model (POM) [16] is an ordinal alternative to the softmax function that was designed for ordinal regression. It belongs to a wider family of models called Cumulative Link Models (CLM). It is inspired in the concept of a latent variable that is projected in an n-dimensional space and a set of thresholds that divides that space into the different ordinal levels. This kind of models uses a link function which can be of different types. The most common link function is the `logit` one, that is used in POM. However, there are other functions that can be explored. This model will be described in depth in Section III.

Ordinal models can also be applied to deep learning models. In the case of convolutional networks, the model projection used by the threshold model will be obtained from the last layer of the network. When working with a 1-dimensional space, the last layer will have only one neuron, and its value will be used to classify every sample in the corresponding class according to the thresholds. Some previous works have used the POM in traditional neural networks [17], but it has not been applied to convolutional networks yet. Also, there are some link functions that has not been explored.

In this paper, an experimental study regarding link functions for Cumulative Link Models in deep learning will be made. Also, other parameters that can affect the training process and the model performance, like the learning rate of the optimization algorithm, the batch size, and its interaction will be studied. The nominal version of this model will be used as a baseline for comparison. We will contrast the results obtained with statistical analysis to provide more robust conclusions. An approximated ANOVA III [18] test followed by a posthoc Tukey's test [19] will be performed because of the limitations of the computational time required to run a higher number of executions.

The experiments will be run using two different ordinal datasets: Diabetic Retinopathy [20], which contains high-resolution fundus images related with diabetes disease, and Adience [21], which includes human faces images associated with an age range.

This paper is organized as follows: in Section II, we take a look at previous works related to this paper. Section III present a formal description of an ordinal problem and the Cumulative Link Models. In Section IV, we describe the model, the experiments and the datasets used, while, in Section V, we present the results obtained and the statistical analysis. Finally, Section VI exposes the conclusions of this work.

II. Related works

There are many works related to the application and development of CNN models. However, few works are focused on ordinal classification problems.

J. de la Torre et al. [20] proposed the use of a continuous version of the QWK metric as loss function for the optimization algorithm. They compared this cost function against the traditional log-loss function across three different databases, including the Diabetic Retinopathy database as the most complex one. They proved that their function could improve the results as it reduces overfitting and training time. Also, they checked the importance of hyper-parameter tuning. First, they defined QWK metric as follows:

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

where ω is the penalization matrix (in this case, quadratic weights are considered), O is the confusion matrix and E is the normalized outer product between the prediction and the true vector.

Then, they provided a continuous version of the QWK metric (QWK_c) based on the probabilities of the predictions:

$$\mathrm{QWK}_c = 1 - \frac{\sum_{k=1}^{N} \sum_{q=1}^{Q} \omega_{t_k,q} P(y = \mathcal{C}_q|\mathbf{x}_k)}{\sum_{i=1}^{Q} \frac{N_i}{N} \sum_{j=1}^{Q} (\omega_{i,j} \sum_{k=1}^{N} P(y = \mathcal{C}_j|\mathbf{x}_k))}, \tag{2}$$

where $\mathrm{QWK}_c \in [-1, 1]$, $\mathbf{x}_k$ and t_k are the input data and the real class of the k-th sample, Q is the number of classes, N is the number of samples, N_i is the number of samples of the ith class, $P(y = \mathcal{C}_q|\mathbf{x}_k)$ is the probability that the kth sample belongs to class $\mathcal{C}_q$ and $\omega_{i,j}$ are the elements of the penalization matrix. Generally, $\omega_{i,j} = \frac{|i-j|^n}{(C-1)^n}$, where $\omega_{i,j} \in [0, 1]$.

Then, they defined the loss function based on the QWK as follows:

$$\mathscr{L} = \log(1 - \mathrm{QWK}_c), \tag{3}$$

where $\mathscr{L} \in [-\infty, \log 2]$. This loss is a function to be minimized while the QWK metric must be maximized.

Z. Niu et al. [22] proposed a learning approach to address ordinal regression problems using convolutional neural networks. They divided the problem into a series of binary classification sub-problems and proposed a multiple output CNN optimization algorithm to collectively resolve these classification sub-problems, taking into account the correlation between them.

C. Beckham and C. Pal [21] proposed a straightforward technique to constrain discrete ordinal probability distributions to be unimodal, via the use of the Poisson and binomial probability distributions. They evaluated this approach in the context of deep learning on two large ordinal image datasets, including the Adience dataset used in this paper, obtaining promising results. Also, they proposed a simple squared-error reformulation [23] that was sensitive to class ordering.

Adience dataset has been used in other works for human age estimation. E. Eidinger [24] presented an approach using support vector machines and neural networks. J.-C. Chen [25] proposed a coarse-to-fine strategy for deep convolutional networks. G. Levi [26] presented another convolutional network model for age estimation. Estos son ordinales? No son ordinales pero los cito aqui porque luego los uso para comparar con ellos ya que son los que he encontrado que utilicen las base de datos Adience para predecir edad.

H. Li et al. [27] applied deep learning techniques for solving the ordinal problem of Alzheimer's diagnosis and detecting the different levels of the disease.

Y. Liu et al. [28] proposed a new approach of which transforms the ordinal regression problem to binary classification problems and use triplets with instances from different categories to train deep neural networks. In this way, high-level features describing the ordinal relationship are extracted automatically.

A. Rios et al. [29] presented a CNN model designed to handle ordinal regression tasks on psychiatric notes. They combined an ordinal loss function, a CNN model and conventional feature engineering. Also, they applied a technique called Locally Interpretable Model-agnostic Explanation (LIME) to make the non-linear model more interpretable.

S. Chen et al. [30] proposed a deep method termed Ranking-CNN. This method combines multiple binary CNNs that are trained with ordinal age labels. The binary outputs are aggregated for the final age prediction and they achieved a tighter error bound for ranking-based age estimation.

H. Fu et al. [31] applied deep learning techniques to Monocular Depth Estimation. They introduced a spacing-increasing discretization strategy to treat the problem as an ordinal regression problem. They improved the performance when training the network with an ordinary regression loss. Also, they used a multi-scale network structure that avoids unnecessary spatial pooling.

Y. Liu et al. [32] proposed a constrained optimization formulation for the ordinal regression problem which minimizes the negative loglikelihood for multiple categories constrained by the order relationship between instances.

A. Pal et al. [33] defined a loss function for CNN that is based on the Earth Mover's Distance and takes into account the ordinal class relationships.

M. ALALI et al. [34] proposed a complex CNN architecture for solving Twitter Sentiment Classification as an ordinal problem. They checked that using average pooling preserves significant features that provide more expresiveness to ordinal scale.

III. Cumulative Link Models (CLM)

An ordinal classification problem consists of predicting the label y of an input vector $\mathbf{x}$, where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^K$ and $y \in \mathcal{Y} = \{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_Q\}$, i.e. $\mathbf{x}$ is in a K-dimensional input space, and y is in a label space of Q different labels. The objective of the ordinal problem is to find a function $r : \mathcal{X} \to \mathcal{Y}$ to predict the labels or categories of new patterns, given a training set of N points, $D = \{(\mathbf{x}_i, y_i), i = 1, ..., N\}$. Labels have a natural ordering in ordinal problems: $\mathcal{C}_1 \prec \mathcal{C}_2 \prec ... \prec \mathcal{C}_Q$. The order between labels makes it possible to compare two different elements of $\mathcal{Y}$ by using the relation $\prec$. This is not possible under the nominal classification setting. In regression (where $y \in \mathbb{R}$), real values in $\mathbb{R}$ can be ordered by the standard $<$ operator, but labels in ordinal regression ($y \in \mathcal{Y}$) do not carry metric information, so the category serves as a qualitative indication of the pattern rather than a quantitative one.

The Proportional Odds Model (POM) arises from a statistical background and is one of the first models designed explicitly for ordinal regression [35]. It dated back to 1980 and is a member of a wider family of models recognised as Cumulative Link Models (CLM) [16]. CLMs predict probabilities of groups of contiguous categories, taking the ordinal scale into account. In this way, cumulative probabilities $P(y \prec C_q|\mathbf{x})$ are estimated, which can be directly related to standard probabilities:

$$P(y \preceq \mathcal{C}_q|\mathbf{x}) = P(y = \mathcal{C}_1|\mathbf{x}) + ... + P(y = \mathcal{C}_q|\mathbf{x}), \quad (4)$$

$$P(y = \mathcal{C}_q|\mathbf{x}) = P(y \preceq \mathcal{C}_q|\mathbf{x}) - P(y \preceq \mathcal{C}_{q-1}|\mathbf{x}), \quad (5)$$

with $q = 2, ..., Q-1$, and considering that $P(y = \mathcal{C}_1|\mathbf{x}) = P(y \preceq \mathcal{C}_1|\mathbf{x})$ and $P(y \preceq \mathcal{C}_Q|\mathbf{x}) = 1$.

The model is inspired by the notion of a latent variable, where $f(\mathbf{x})$ represents a one-dimensional mapping obtained from the output of the last layer, which has only one neuron. The decision rule $r : \mathcal{X} \to \mathcal{Y}$ is not fitted directly, but stochastic ordering of space $\mathcal{X}$ is satisfied by the following general model form [36]:

$$g^{-1}(P(y \preceq \mathcal{C}_q|\mathbf{x})) = b_q - f(\mathbf{x}), \quad q = 1, ..., Q-1, \quad (6)$$

where $g^{-1} : [0, 1] \to (-\infty, +\infty)$ is a monotonic function often termed as the inverse link function, and b_q is the threshold defined for class $\mathcal{C}_q$. Consider the latent variable $y^* = f(\mathbf{x})^* = f(\mathbf{x}) + \epsilon$, where ϵ is the random variable of the error. The most common choice for the probability distribution of ϵ is the logistic function (which is the default function for POM). Label $\mathcal{C}_q$ is predicted if and only if $f(\mathbf{x}) \in [b_{q-1}, b_q]$, where the function f and $\mathbf{b} = (b_0, b_1, ..., b_{Q-1}, b_Q)$ are to be determined from the data. It is assumed that $b_0 = -\infty$ and $b_Q = +\infty$, so the real line defined by $f(\mathbf{x}), \mathbf{x} \in \mathcal{X}$, is divided into Q consecutive intervals. Each interval corresponds to a category. The constraints $b_1 \leq b_2 \leq ... \leq b_{Q-1}$ ensures that $P(y \preceq \mathcal{C}_q|\mathbf{x})$ increases with q [35]. This order is achieved by defining the first threshold and calculating the rest of thresholds from the first in the following form:

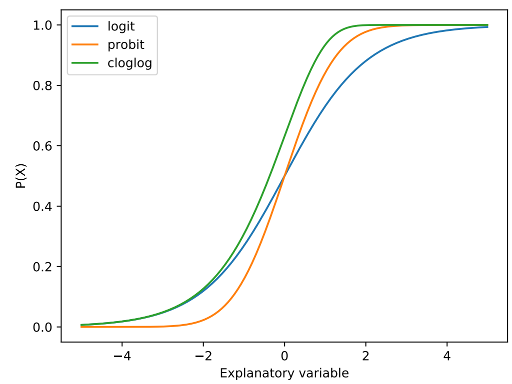

Fig. 1. Representation of link functions.

$$b_n = b_1 + \sum_{i=1}^{n-1} \alpha_i^2, \quad n = 2, ..., N, \quad (7)$$

where b_1 and α_i are learnable parameters, and N is the number of classes.

In this work, we use this ordinal model with different link functions for the probability distribution of ϵ, including `logit`, `probit` and complementary log-log (`clog-log`).

These three types of links are explained below and represented in Figure 1. They all follow the same form $P(y \preceq \mathcal{C}_q|\mathbf{x}) = \Phi(b_q - f(\mathbf{x}))$ for a continuous cdf Φ.

- `Logit`. `logit` link function is the function used for the Proportional Odds Model. The `logit` link is:

$$\begin{aligned}\text{logit}[P(y \preceq \mathcal{C}_q|\mathbf{x})] = \log \frac{P(y \preceq \mathcal{C}_q|\mathbf{x})}{1 - P(y \preceq \mathcal{C}_q|\mathbf{x})} = \\ = b_q - f(\mathbf{x}), \quad q = 1, ..., Q-1,\end{aligned} \quad (8)$$

  or the equivalent expression:

$$P(y \preceq \mathcal{C}_q|\mathbf{x}) = \frac{1}{1 + e^{-(b - f(\mathbf{x}))}}. \quad (9)$$

- `Probit`. `probit` link function is the inverse of the standard normal cumulative distribution function (cdf) Φ. Its expression is:

$$\begin{aligned}\Phi^{-1}[P(y \preceq \mathcal{C}_q|\mathbf{x})] = b_q - f(\mathbf{x}), \quad q = 1, ..., Q-1, \\ P(y \preceq \mathcal{C}_q|\mathbf{x}) = \Phi(b_q - f(\mathbf{x})), \quad q = 1, ..., Q-1,\end{aligned} \quad (10)$$

  which can also be expressed as:

$$P(y \preceq \mathcal{C}_q|\mathbf{x}) = \int_{-\infty}^{b_q - f(\mathbf{x})} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \mathrm{d}x. \quad (11)$$

- *Complementary log-log*. Like the `logit` and the `probit` transformation, the complementary log-log transformation takes a response that is restricted to the $(0, 1)$ interval and converts it into something in the $(-\infty, +\infty)$ interval. Complementary log-log expression is:

$$\begin{aligned}\log[-\log[1 - P(y \preceq \mathcal{C}_q|\mathbf{x})]] = \\ = b_q - f(\mathbf{x}), \quad q = 1, ..., Q-1,\end{aligned} \quad (12)$$

that is:

$$P(y \preceq \mathcal{C}_q|\mathbf{x}) = 1 - e^{-e^{b_q - f(\mathbf{x})}}, \quad q = 1, ..., Q-1. \quad (13)$$

`Logit` and `probit` links are symmetric:

$$\text{link}[P(y \preceq \mathcal{C}_q|\mathbf{x})] = -\text{link}[1 - P(y \preceq \mathcal{C}_q|\mathbf{x})], \quad (14)$$

which means that the response curve for $P(y \preceq \mathcal{C}_q|\mathbf{x})$ is symmetric around the point $P(y \preceq \mathcal{C}_q|\mathbf{x}) = 0.5$, i.e. $P(y \preceq \mathcal{C}_q|\mathbf{x})$ has the same rate when approaching 0 than when approaching 1. This symmetric property can be demonstrated as follows:

1) Let $P(y \preceq \mathcal{C}_q|\mathbf{x}) \equiv p$. For the `logit` function, we have:

$$\begin{aligned}\text{link}(p) = \text{logit}(p) = \log\left(\frac{p}{1-p}\right) = \\ = \log(p) - \log(1-p),\end{aligned} \quad (15)$$

while:

$$\begin{aligned}-\text{link}(1-p) = -\text{logit}(1-p) = \\ = -\log\left(\frac{1-p}{p}\right) = -\log(1-p) + \log(p).\end{aligned} \quad (16)$$

2) For the `probit`:

$$\begin{aligned}p \equiv P(y \preceq \mathcal{C}_q|\mathbf{x}) = \Phi(b_q - f(\mathbf{x})) = \\ = \int_{-\infty}^{b_q - f(\mathbf{x})} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \mathrm{d}x,\end{aligned} \quad (17)$$

which leads to:

$$\text{probit}(p) = \Phi^{-1}(p) = b_q - f(\mathbf{x}), \quad (18)$$

$$-\text{probit}(1-p) = \Phi^{-1}(1-p) = -b_q + f(\mathbf{x}), \quad (19)$$

where:

$$1 - p = \int_{-\infty}^{-b_q + f(\mathbf{x})} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \mathrm{d}x, \quad (20)$$

$$p = 1 - \int_{-\infty}^{-b_q + f(\mathbf{x})} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \mathrm{d}x. \quad (21)$$

Unlike `logit` and `probit`, the complementary log-log model is asymmetrical. It is frequently used when the probability of an event is very small or very large. When the given data is not symmetric in the $[0, 1]$ interval and increase slowly at small to moderate value but increases sharply near 1, the `logit` and `probit` models are inappropriate. However, in this situation, the complementary log-log model might give a satisfying answer.

IV. Experiments

A. Data

In order to evaluate the different models, we make use of two ordinal datasets:

- *Diabetic Retinopathy (DR)*[1]. DR is a dataset consisting of extremely high-resolution fundus image data. The training set consists of 17563 pairs of images (where a pair consists of a left and right eye image

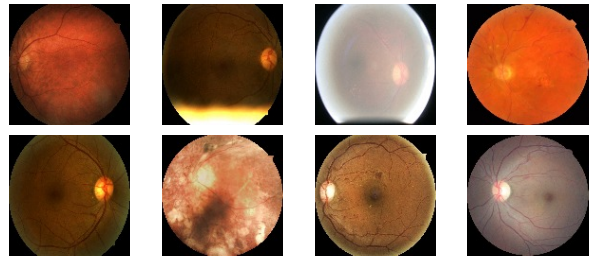

Fig. 2. Examples of the Diabetic Retinopathy test set.

Fig. 3. Examples of the Adience test set.

corresponding to a patient). In this dataset, we try to predict the correct category from five levels of diabetic retinopathy: no DR (25810 images), mild DR (2443 images), moderate DR (5292 images), severe DR (873 images), or proliferative DR (708 images). The test set contains 26788 pairs of images. These images are taken in variable conditions: by different cameras, varying conditions of illumination and different resolutions. These images come from the EyePACS dataset that was used in a Diabetic Retinopathy Detection competition hosted on the Kaggle platform. Also, this dataset was used in later works [20], [37], applying an ordinal QWK cost function in [20] to achieve better performance. A validation set is set aside, consisting of 10% of the patients in the training set. The images are resized to 128 by 128 pixels and rescaled to $[0, 1]$ range. Data augmentation techniques, described in Section IV-C, are applied to achieve a higher number of samples. A few images of this dataset are shown in Figure 2.
- *Adience*[2]. This dataset consists of 26580 faces belonging to 2284 subjects. We use the form of the dataset where faces have been pre-cropped and aligned. The dataset was preprocessed, using the methods described in a previous work [21], so that the images are 256 pixels in width and height, and pixels values follow a $(0; 1)$ normal distribution. The original dataset was split into five cross-validation folds. The training set consists of merging the first four folds which comprise a total of 15554 images. From this, 10% of the images are held out as part of a validation set. The last fold is used as test set. Some images of this dataset are shown in Figure 3.

[1]https://www.kaggle.com/c/diabetic-retinopathy-detection/data

[2]http://www.openu.ac.il/home/hassner/Adience/data.html

TABLE I
DESCRIPTION OF THE ARCHITECTURE USED IN THE DR EXPERIMENTS.

Layer	Output shape
2 x Conv3x3@32s1	252x252x32
MaxPool2x2s2	126x126x32
2 x Conv3x3@64s1	122x122x64
MaxPool2x2s2	61x61x64
2 x Conv3x3@128s1	57x57x128
MaxPool2x2s2	28x28x128
2 x Conv3x3@128s1	24x24x128
MaxPool2x2s2	12x12x128
Conv4x4@128s1	9x9x128

TABLE II
DESCRIPTION OF THE ARCHITECTURE USED IN THE ADIENCE EXPERIMENTS.

Layer	Output shape
Conv7x7@32s2	112x112x32
MaxPool3x3s2	55x55x32
2 x ResBlock3x3@64s1	55x55x32
1 x ResBlock3x3@128s2	28x28x64
2 x ResBlock3x3@128s1	28x28x64
1 x ResBlock3x3@256s2	14x14x128
2 x ResBlock3x3@256s1	14x14x128
1 x ResBlock3x3@512s2	7x7x256
2 x ResBlock3x3@512s1	7x7x256
AveragePool7x7s2	1x1x256

B. Model

CNNs have been used for both datasets. The different architectures of CNN used in these experiments are presented in tables I and II. The architecture for DR is the same that was used in [20] and the network for Adience is a small Residual Network (ResNet) [3] that was used in [21]. The most important parameters for convolutional layers are the number of filters that are used to make the convolution operation, the size of these filters and the stride, which is the number of pixels that the filter is moved for obtaining the next pixel. Pooling layers have got similar parameters: the pool size is the number of pixels that will be involved in the pooling operation, and the stride represents the same concept that in convolutional layers. For convolutional layers, ConvWxH@FsS = F filters of size WxH and stride S. For pooling layers, PoolWxHsS = pool size of WxH and stride S.

The Exponential Linear Unit (ELU) [38] has been used for activation function for all the convolutional and dense layers, instead of the ReLU [39] function, as it mitigates the effects of the vanishing gradients problem [40], [41] via the identity for positive values. Also, ELUs lead to faster training and better generalization performance than ReLU and Leaky ReLU (LReLU) [42] functions on networks with more than five layers.

After every ELU activation function of the convolutional layers, Batch Normalization [43] is applied. This method reduces the internal covariate shift by normalizing layer outputs. It allows us to use higher learning rates and be less careful about weights initialization. It also eliminates the need for using regularization techniques like Dropout.

At the output of the network, the CLM is used. Also, a learnable parameter has been used to rescale the projections used by the Cumulative Link Model to make it more stable and guarantee the convergence in most cases.

C. Experimental design

The model is optimized using a batch based first-order optimization algorithm called Adam [44]. We study different initial learning rates (η) in order to find the optimal one for each problem. We apply an exponential decay across training epochs to the initial learning rate (η_0) following the expression below:

$$\eta = \eta_0 \cdot e^{-0.025 \cdot \text{epoch}}. \tag{22}$$

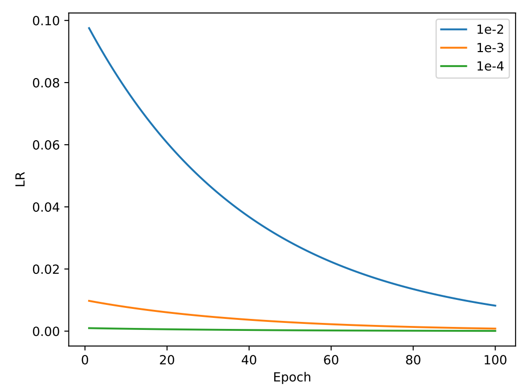

Fig. 4. Representation of the learning rate decay.

Fig. 4 represents the learning rate decay across 100 epochs for the different initial values considered in this work.

Both datasets are artificially equalized using data augmentation techniques [45]. However, different transformations are applied to each one. DR dataset augmentation is based on image cropping and zooming, horizontal and vertical flipping, brightness adjustment and random rotations. Horizontal flipping is the only transformation applied to Adience dataset. These transformations are applied every time a new batch is loaded, and the parameters of each one are randomly chosen from a defined range, providing a new set of transformed images for each batch. This technique reduces the overfitting risk and provides an important performance boost as we always work with different but similar images [4].

The epoch size is equal to the number of images in the training set. It could be a higher number as we are using data augmentation, but instead of increasing the epoch size, we rather run the training for more epochs. In this case, we set the maximum number of epochs to 100. However, we always save the best model, that is evaluated when the training finishes.

The model is evaluated using the Quadratic Weighted Kappa metric (QWK) [46]. This evaluation measure gives a higher weight to the errors that are further from the correct class. A similar approach to the one described in [20] is used as the loss function as it gives better performance for ordinal classification problems. In this case, we define the QWK loss function (QWK_l) as follows:

$$\mathrm{QWK}_l = \frac{\sum_{k=1}^{N}\sum_{q=1}^{Q} \omega_{t_k,q} P(y=\mathcal{C}_q|\mathbf{x}_k)}{\sum_{i=1}^{Q} \frac{N_i}{N} \sum_{j=1}^{Q} (\omega_{i,j} \sum_{k=1}^{N} P(y=\mathcal{C}_j|\mathbf{x}_k))}, \quad (23)$$

where $\mathrm{QWK}_l \in [0,2]$, $\mathbf{x}_k$ and t_k are the input data and the real class of the k-th sample, Q is the number of classes, N is the number of samples, N_i is the number of samples of the ith class, $P(y=\mathcal{C}_q|\mathbf{x}_k)$ is the probability that the kth sample belongs to class $\mathcal{C}_q$ and $\omega_{i,j}$ are the elements of the penalization matrix. In this case, $\omega_{i,j} = \frac{(i-j)^2}{(C-1)^2}$, where $\omega_{i,j} \in [0,1]$.

Also, other evaluation metrics are used to ease the comparison with other works: Minimum Sensitivity (MS) [47], Mean Absolute Error (MAE) [47], Mean Squared Error (MSE) [48], Correct Classification Rate (CCR), Top-2 CCR [21], Top-3 CCR [21] and 1-off accuracy [25], [26], [24].

Experiments are run with the standard cross-entropy loss and the softmax function too in order to prove the performance improvement of considering the ordinality of the problem (QWK loss and the Cumulative Link Model). The results of these experiments are analysed in Section V-D.

D. Factors

Three different factors are considered: learning rate, batch size and link function for the final output layer.

- *Learning rate* (LR, η). Learning rate is one of the most critical hyper-parameters to tune for training deep neural networks. Optimal learning rate can vary depending on the dataset and the CNN architecture. Previous works have presented some techniques that adjust this parameter in order to achieve better performance [49], [50]. Within this work, we consider three different values for this parameter: 10^{-2}, 10^{-3} and 10^{-4}.
- *Batch size* (BS). Batch size is also an important parameter as it controls the number of weight updates that are made on every epoch. It can affect the training time and the model performance. In this paper, we try three separate batch sizes for each dataset. For DR dataset, we use 5, 10 and 15 while, for the Adience dataset, 64, 128 and 256 images are used. We took the batch sizes that were used in [20] and [21] as a reference, and we expanded the range on both sides.
- *Link function* (LF). Different link functions are used for the CLM at the last layer output: `logit`, `probit` and complementary log-log.

V. Results

In this section, we present the results of the experiments. For each dataset, we show a table with the detailed results of the experiments performed for training the model with each combination of parameters. Every parameter combination was run five times. These tables show the mean value and the standard deviation (SD) of each metric across these five executions for the test set.

A. Diabetic Retinopathy

Detailed test results for the Diabetic Retinopathy dataset are presented in Table III. The best result for each metric is marked in bold and the second best is in italic font.

The best mean QWK value was obtained with the complementary log-log link function using a batch size of 10 and a learning rate of 10^{-3}. However, the best CCR value was obtained with a batch size of 15, the `logit` link and a learning rate of 10^{-4}. The optimal configuration depends on the metric we are analysing. In this case, as we are working with an ordinal problem, the most reliable metric is the QWK. However, the rest of the metrics are also included to allow further comparisons with future works.

B. Adience

Test results for the experiments made with the Adience dataset are shown in Table IV. The best result for each metric is marked in bold and the second best is in italic font.

The best mean QWK value was obtained with the `logit` link function using a batch size of 64 and a learning rate of 10^{-4}. Also, this configuration obtained the best score for Top-2, Top-3 and 1-off accuracy, and the second best for MS, MAE and CCR. In this case, this configuration can be selected as the optimal for this problem.

C. Statistical analysis

In this subsection, a statistical analysis will be performed in order to obtain conclusions from the result.

The significance and relative importance of the parameters concerning the results obtained, as well as suitable values for each of them, were obtained using an ANalysis Of the VAriance (ANOVA).

The ANOVA test [18] is one of the most widely used statistical techniques. ANOVA is essentially a method of analysing the variance to which a response is subject into its various components, corresponding to the sources of variation which can be identified.

ANOVA, in this case, examines the effects of three quantitative variables (termed factors) on one quantitative response. Considered factors are the link function, the learning rate for the Adam optimization algorithm, and the batch size.

Following the setup of the previous study, we performed an ANOVA III analysis and multiple comparison tests. We assume that five executions are enough to do the statistical tests because of the computational time limitations.

We denote by $\mathrm{QWK}_{i,j,k,l}(i=1,...,3; j=1,...,3; k=1,...,3)$ the value observed when the first factor is at the

TABLE III
DIABETIC RETINOPATHY RESULTS. BS STANDS FOR BATCH SIZE, LF FOR LINK FUNCTION AND LR FOR LEARNING RATE. MEAN AND STANDARD DEVIATION MEAN_{SD}.

BS	LF	LR	$\overline{\text{QWK}}_{(SD)}$	$\overline{\text{MS}}_{(SD)}$	$\overline{\text{MAE}}_{(SD)}$	$\overline{\text{MSE}}_{(SD)}$	$\overline{\text{CCR}}_{(SD)}$	$\overline{\text{Top-2}}_{(SD)}$	$\overline{\text{Top-3}}_{(SD)}$	$\overline{\text{1-off}}_{(SD)}$
5	`clog-log`	10^{-2}	$0.414_{(0.057)}$	$0.075_{(0.042)}$	$0.177_{(0.023)}$	$0.165_{(0.020)}$	$0.556_{(0.057)}$	$0.833_{(0.042)}$	$0.968_{(0.011)}$	$0.816_{(0.021)}$
5	`clog-log`	10^{-3}	$0.534_{(0.027)}$	$0.102_{(0.011)}$	$0.137_{(0.006)}$	$0.123_{(0.004)}$	$0.658_{(0.015)}$	$0.871_{(0.011)}$	$0.966_{(0.003)}$	$0.852_{(0.002)}$
5	`clog-log`	10^{-4}	$0.520_{(0.006)}$	$0.067_{(0.008)}$	$0.123_{(0.003)}$	$0.104_{(0.001)}$	$0.697_{(0.006)}$	$0.842_{(0.008)}$	$0.961_{(0.003)}$	$0.851_{(0.002)}$
5	`logit`	10^{-2}	$0.416_{(0.041)}$	$0.095_{(0.029)}$	$0.175_{(0.021)}$	$0.162_{(0.018)}$	$0.563_{(0.054)}$	$0.762_{(0.040)}$	$0.908_{(0.026)}$	$0.807_{(0.029)}$
5	`logit`	10^{-3}	$0.554_{(0.013)}$	$0.093_{(0.009)}$	$0.137_{(0.003)}$	$0.123_{(0.003)}$	$0.660_{(0.008)}$	$0.802_{(0.005)}$	$0.936_{(0.004)}$	$0.853_{(0.005)}$
5	`logit`	10^{-4}	$0.520_{(0.003)}$	$0.063_{(0.004)}$	$0.122_{(0.002)}$	$0.102_{(0.001)}$	$0.706_{(0.005)}$	$0.823_{(0.004)}$	$0.949_{(0.003)}$	$0.862_{(0.003)}$
5	`probit`	10^{-2}	$0.460_{(0.048)}$	$0.079_{(0.046)}$	$0.197_{(0.064)}$	$0.182_{(0.053)}$	$0.504_{(0.167)}$	$0.808_{(0.034)}$	$0.927_{(0.073)}$	$0.689_{(0.240)}$
5	`probit`	10^{-3}	$0.564_{(0.018)}$	$0.099_{(0.013)}$	$0.147_{(0.018)}$	$0.132_{(0.014)}$	$0.636_{(0.045)}$	$0.822_{(0.040)}$	$0.939_{(0.020)}$	$0.840_{(0.015)}$
5	`probit`	10^{-4}	$0.523_{(0.005)}$	$0.067_{(0.012)}$	$0.122_{(0.002)}$	$0.105_{(0.001)}$	$0.701_{(0.006)}$	$0.823_{(0.002)}$	$0.953_{(0.002)}$	$0.860_{(0.003)}$
10	`clog-log`	10^{-2}	$0.423_{(0.239)}$	$0.062_{(0.051)}$	$0.127_{(0.017)}$	$0.120_{(0.014)}$	$0.684_{(0.046)}$	$\mathbf{0.894}_{(\mathbf{0.062})}$	$\mathbf{0.986}_{(\mathbf{0.012})}$	$0.832_{(0.020)}$
10	`clog-log`	10^{-3}	$\mathbf{0.582}_{(\mathbf{0.016})}$	$0.102_{(0.006)}$	$0.128_{(0.003)}$	$0.115_{(0.002)}$	$0.680_{(0.007)}$	$\mathit{0.880}_{(\mathit{0.004})}$	$0.972_{(0.003)}$	$0.861_{(0.004)}$
10	`clog-log`	10^{-4}	$0.537_{(0.010)}$	$0.064_{(0.004)}$	$\mathit{0.116}_{(\mathit{0.001})}$	$0.096_{(0.001)}$	$0.717_{(0.003)}$	$0.837_{(0.002)}$	$0.971_{(0.001)}$	$0.860_{(0.002)}$
10	`logit`	10^{-2}	$0.531_{(0.031)}$	$0.107_{(0.008)}$	$0.151_{(0.010)}$	$0.140_{(0.008)}$	$0.623_{(0.025)}$	$0.802_{(0.022)}$	$0.934_{(0.013)}$	$0.838_{(0.014)}$
10	`logit`	10^{-3}	$0.579_{(0.009)}$	$0.096_{(0.012)}$	$0.127_{(0.005)}$	$0.113_{(0.004)}$	$0.686_{(0.013)}$	$0.817_{(0.006)}$	$0.954_{(0.005)}$	$0.861_{(0.002)}$
10	`logit`	10^{-4}	$0.539_{(0.007)}$	$0.074_{(0.013)}$	$0.126_{(0.005)}$	$0.095_{(0.002)}$	$0.707_{(0.010)}$	$0.823_{(0.007)}$	$0.957_{(0.005)}$	$0.858_{(0.004)}$
10	`probit`	10^{-2}	$0.508_{(0.037)}$	$0.088_{(0.044)}$	$0.145_{(0.018)}$	$0.137_{(0.014)}$	$0.639_{(0.045)}$	$0.835_{(0.015)}$	$0.960_{(0.008)}$	$0.829_{(0.020)}$
10	`probit`	10^{-3}	$0.558_{(0.034)}$	$\mathbf{0.111}_{(\mathbf{0.005})}$	$0.134_{(0.003)}$	$0.120_{(0.002)}$	$0.666_{(0.008)}$	$0.831_{(0.007)}$	$0.955_{(0.001)}$	$0.863_{(0.003)}$
10	`probit`	10^{-4}	$0.541_{(0.010)}$	$0.076_{(0.006)}$	$0.119_{(0.002)}$	$0.098_{(0.001)}$	$0.712_{(0.005)}$	$0.828_{(0.003)}$	$0.961_{(0.002)}$	$0.862_{(0.001)}$
15	`clog-log`	10^{-2}	$0.564_{(0.016)}$	$0.108_{(0.014)}$	$0.143_{(0.006)}$	$0.134_{(0.005)}$	$0.640_{(0.015)}$	$0.879_{(0.011)}$	$0.972_{(0.005)}$	$0.851_{(0.006)}$
15	`clog-log`	10^{-3}	$0.559_{(0.026)}$	$\mathit{0.111}_{(\mathit{0.008})}$	$0.127_{(0.004)}$	$0.113_{(0.003)}$	$0.682_{(0.010)}$	$0.871_{(0.008)}$	$\mathit{0.974}_{(\mathit{0.002})}$	$\mathbf{0.868}_{(\mathbf{0.002})}$
15	`clog-log`	10^{-4}	$0.538_{(0.009)}$	$0.054_{(0.003)}$	$\mathbf{0.115}_{(\mathbf{0.002})}$	$\mathit{0.093}_{(\mathit{0.001})}$	$0.720_{(0.006)}$	$0.835_{(0.007)}$	$0.970_{(0.003)}$	$0.860_{(0.006)}$
15	`logit`	10^{-2}	$0.551_{(0.020)}$	$0.104_{(0.008)}$	$0.139_{(0.011)}$	$0.129_{(0.009)}$	$0.654_{(0.027)}$	$0.815_{(0.017)}$	$0.948_{(0.016)}$	$0.856_{(0.015)}$
15	`logit`	10^{-3}	$0.551_{(0.010)}$	$0.106_{(0.016)}$	$0.129_{(0.008)}$	$0.114_{(0.005)}$	$0.680_{(0.019)}$	$0.818_{(0.008)}$	$0.952_{(0.007)}$	$\mathit{0.866}_{(\mathit{0.001})}$
15	`logit`	10^{-4}	$0.543_{(0.008)}$	$0.056_{(0.003)}$	$0.121_{(0.004)}$	$\mathbf{0.090}_{(\mathbf{0.001})}$	$\mathbf{0.723}_{(\mathbf{0.004})}$	$0.833_{(0.004)}$	$0.964_{(0.003)}$	$0.862_{(0.004)}$
15	`probit`	10^{-2}	$0.534_{(0.032)}$	$0.104_{(0.013)}$	$0.148_{(0.015)}$	$0.138_{(0.014)}$	$0.631_{(0.038)}$	$0.845_{(0.030)}$	$0.964_{(0.010)}$	$0.852_{(0.010)}$
15	`probit`	10^{-3}	$\mathit{0.580}_{(\mathit{0.021})}$	$0.104_{(0.016)}$	$0.129_{(0.008)}$	$0.116_{(0.005)}$	$0.680_{(0.018)}$	$0.832_{(0.010)}$	$0.959_{(0.007)}$	$0.866_{(0.003)}$
15	`probit`	10^{-4}	$0.533_{(0.004)}$	$0.065_{(0.005)}$	$0.117_{(0.002)}$	$0.094_{(0.001)}$	$\mathit{0.721}_{(\mathit{0.004})}$	$0.832_{(0.002)}$	$0.964_{(0.001)}$	$0.863_{(0.001)}$

TABLE IV
ADIENCE TEST RESULTS. BS STANDS FOR BATCH SIZE, LF FOR LINK FUNCTION AND LR FOR LEARNING RATE. MEAN AND STANDARD DEVIATION MEAN_{SD}.

BS	LF	LR	$\overline{\text{QWK}}_{(SD)}$	$\overline{\text{MS}}_{(SD)}$	$\overline{\text{MAE}}_{(SD)}$	$\overline{\text{MSE}}_{(SD)}$	$\overline{\text{CCR}}_{(SD)}$	$\overline{\text{Top-2}}_{(SD)}$	$\overline{\text{Top-3}}_{(SD)}$	$\overline{\text{1-off}}_{(SD)}$
64	`clog-log`	10^{-2}	$0.808_{(0.025)}$	$0.086_{(0.041)}$	$0.147_{(0.008)}$	$0.129_{(0.007)}$	$0.415_{(0.031)}$	$0.677_{(0.024)}$	$0.798_{(0.036)}$	$0.804_{(0.015)}$
64	`clog-log`	10^{-3}	$0.873_{(0.006)}$	$0.144_{(0.057)}$	$\mathbf{0.124}_{(\mathbf{0.003})}$	$0.101_{(0.003)}$	$\mathbf{0.519}_{(\mathbf{0.014})}$	$\mathit{0.764}_{(\mathit{0.010})}$	$0.861_{(0.019)}$	$0.886_{(0.006)}$
64	`clog-log`	10^{-4}	$0.799_{(0.010)}$	$0.000_{(0.000)}$	$0.174_{(0.001)}$	$0.100_{(0.002)}$	$0.324_{(0.015)}$	$0.616_{(0.020)}$	$0.795_{(0.012)}$	$0.771_{(0.014)}$
64	`logit`	10^{-2}	$0.778_{(0.019)}$	$0.074_{(0.041)}$	$0.159_{(0.006)}$	$0.137_{(0.007)}$	$0.366_{(0.025)}$	$0.636_{(0.015)}$	$0.785_{(0.010)}$	$0.775_{(0.015)}$
64	`logit`	10^{-3}	$\mathbf{0.881}_{(\mathbf{0.005})}$	$\mathit{0.178}_{(\mathit{0.023})}$	$\mathit{0.126}_{(\mathit{0.001})}$	$0.098_{(0.003)}$	$\mathit{0.518}_{(\mathit{0.008})}$	$\mathbf{0.765}_{(\mathbf{0.015})}$	$\mathbf{0.902}_{(\mathbf{0.005})}$	$\mathbf{0.894}_{(\mathbf{0.005})}$
64	`logit`	10^{-4}	$0.784_{(0.011)}$	$0.000_{(0.000)}$	$0.180_{(0.001)}$	$0.108_{(0.004)}$	$0.318_{(0.026)}$	$0.621_{(0.034)}$	$0.772_{(0.024)}$	$0.731_{(0.030)}$
64	`probit`	10^{-2}	$0.836_{(0.005)}$	$0.135_{(0.021)}$	$0.134_{(0.002)}$	$0.121_{(0.002)}$	$0.468_{(0.011)}$	$0.720_{(0.009)}$	$0.861_{(0.009)}$	$0.829_{(0.005)}$
64	`probit`	10^{-3}	$\mathit{0.874}_{(\mathit{0.004})}$	$0.134_{(0.012)}$	$0.126_{(0.003)}$	$0.105_{(0.003)}$	$0.511_{(0.014)}$	$0.756_{(0.009)}$	$\mathit{0.895}_{(\mathit{0.003})}$	$\mathit{0.889}_{(\mathit{0.003})}$
64	`probit`	10^{-4}	$0.805_{(0.004)}$	$0.000_{(0.000)}$	$0.170_{(0.001)}$	$0.100_{(0.002)}$	$0.360_{(0.011)}$	$0.653_{(0.011)}$	$0.809_{(0.009)}$	$0.790_{(0.009)}$
128	`clog-log`	10^{-2}	$0.832_{(0.013)}$	$0.123_{(0.031)}$	$0.135_{(0.004)}$	$0.117_{(0.002)}$	$0.463_{(0.013)}$	$0.705_{(0.019)}$	$0.813_{(0.025)}$	$0.832_{(0.006)}$
128	`clog-log`	10^{-3}	$0.873_{(0.006)}$	$\mathbf{0.185}_{(\mathbf{0.029})}$	$0.128_{(0.002)}$	$0.100_{(0.001)}$	$0.513_{(0.007)}$	$0.758_{(0.008)}$	$0.870_{(0.011)}$	$0.880_{(0.009)}$
128	`clog-log`	10^{-4}	$0.659_{(0.025)}$	$0.000_{(0.000)}$	$0.190_{(0.002)}$	$0.125_{(0.004)}$	$0.235_{(0.026)}$	$0.466_{(0.031)}$	$0.640_{(0.030)}$	$0.536_{(0.041)}$
128	`logit`	10^{-2}	$0.781_{(0.041)}$	$0.096_{(0.059)}$	$0.153_{(0.007)}$	$0.125_{(0.007)}$	$0.398_{(0.031)}$	$0.638_{(0.033)}$	$0.790_{(0.025)}$	$0.779_{(0.020)}$
128	`logit`	10^{-3}	$0.865_{(0.005)}$	$0.127_{(0.026)}$	$0.134_{(0.001)}$	$0.099_{(0.003)}$	$0.497_{(0.009)}$	$0.754_{(0.008)}$	$0.882_{(0.009)}$	$0.874_{(0.008)}$
128	`logit`	10^{-4}	$0.586_{(0.008)}$	$0.000_{(0.000)}$	$0.196_{(0.001)}$	$0.151_{(0.005)}$	$0.192_{(0.001)}$	$0.364_{(0.060)}$	$0.581_{(0.034)}$	$0.396_{(0.002)}$
128	`probit`	10^{-2}	$0.849_{(0.005)}$	$0.132_{(0.010)}$	$0.131_{(0.001)}$	$0.115_{(0.001)}$	$0.479_{(0.004)}$	$0.728_{(0.007)}$	$0.854_{(0.009)}$	$0.847_{(0.007)}$
128	`probit`	10^{-3}	$0.866_{(0.002)}$	$0.124_{(0.043)}$	$0.130_{(0.002)}$	$0.100_{(0.003)}$	$0.505_{(0.006)}$	$0.750_{(0.010)}$	$0.882_{(0.004)}$	$0.873_{(0.006)}$
128	`probit`	10^{-4}	$0.718_{(0.015)}$	$0.000_{(0.000)}$	$0.185_{(0.001)}$	$0.110_{(0.002)}$	$0.300_{(0.031)}$	$0.575_{(0.015)}$	$0.733_{(0.010)}$	$0.640_{(0.033)}$
256	`clog-log`	10^{-2}	$0.853_{(0.004)}$	$0.157_{(0.024)}$	$0.130_{(0.002)}$	$0.110_{(0.001)}$	$0.485_{(0.009)}$	$0.744_{(0.006)}$	$0.842_{(0.016)}$	$0.858_{(0.004)}$
256	`clog-log`	10^{-3}	$0.840_{(0.017)}$	$0.095_{(0.017)}$	$0.144_{(0.005)}$	$0.097_{(0.004)}$	$0.456_{(0.021)}$	$0.720_{(0.022)}$	$0.840_{(0.018)}$	$0.842_{(0.018)}$
256	`clog-log`	10^{-4}	$0.552_{(0.010)}$	$0.000_{(0.000)}$	$0.199_{(0.001)}$	$0.165_{(0.004)}$	$0.187_{(0.001)}$	$0.368_{(0.022)}$	$0.475_{(0.025)}$	$0.387_{(0.001)}$
256	`logit`	10^{-2}	$0.764_{(0.102)}$	$0.077_{(0.067)}$	$0.155_{(0.020)}$	$0.125_{(0.015)}$	$0.387_{(0.083)}$	$0.632_{(0.103)}$	$0.790_{(0.077)}$	$0.783_{(0.065)}$
256	`logit`	10^{-3}	$0.851_{(0.008)}$	$0.100_{(0.030)}$	$0.147_{(0.003)}$	$\mathbf{0.094}_{(\mathbf{0.002})}$	$0.449_{(0.015)}$	$0.726_{(0.015)}$	$0.861_{(0.006)}$	$0.850_{(0.008)}$
256	`logit`	10^{-4}	$0.558_{(0.008)}$	$0.000_{(0.000)}$	$0.202_{(0.001)}$	$0.191_{(0.002)}$	$0.187_{(0.002)}$	$0.206_{(0.007)}$	$0.395_{(0.046)}$	$0.389_{(0.003)}$
256	`probit`	10^{-2}	$0.858_{(0.005)}$	$0.164_{(0.033)}$	$0.130_{(0.002)}$	$0.112_{(0.002)}$	$0.486_{(0.007)}$	$0.741_{(0.008)}$	$0.867_{(0.008)}$	$0.862_{(0.005)}$
256	`probit`	10^{-3}	$0.850_{(0.008)}$	$0.111_{(0.040)}$	$0.144_{(0.002)}$	$\mathit{0.095}_{(\mathit{0.001})}$	$0.460_{(0.011)}$	$0.732_{(0.006)}$	$0.865_{(0.006)}$	$0.853_{(0.007)}$
256	`probit`	10^{-4}	$0.565_{(0.010)}$	$0.000_{(0.000)}$	$0.196_{(0.001)}$	$0.150_{(0.005)}$	$0.189_{(0.001)}$	$0.409_{(0.014)}$	$0.602_{(0.022)}$	$0.392_{(0.002)}$

i-th level, the second at the j-th level and the third at the k-th level. We assume that the three factors do not act independently and therefore there exists an interaction between each pair of them and between the three factors. In this case, the observations fit:

$$\begin{aligned}\mathrm{QWK}_{i,j,k,l} = \mu + L_i + P_j + B_k + LP_{i,j} + LB_{i,k} \\ + PB_{j,k} + LPB_{i,j,k} + \epsilon_{i,j,k,l},\end{aligned} \tag{24}$$

where μ is the fixed effect that is common to all the populations; L_i is the effect associated with the *i-th* level of the link factor (`logit`, `probit`, complementary log-log); P_j is the effect associated with the *j-th* level of the learning rate factor and B_k is the effect associated with the *k-th* level of the batch size factor. The term $LP_{i,j}$ denotes the joint effect of the presence of level i of the first factor and level j of the second one; this, therefore, is denominated the interaction term between L and P factors. The same interaction effect is appreciated on $LB_{i,k}$, $PB_{j,k}$ and $LPB_{i,j,k}$. The term $\epsilon_{i,j,k,l}$ is the influence on the result of everything that could not be assigned or of random factors. $QWK_{i,j,k,l}$ is the quadratic weighted kappa measure, the response variable used to perform the statistical analysis.

We consider some hypotheses testing where the null hypothesis is proposed that each term of the above equation is independent of the levels involved. The hypotheses for the levels of the L factor are $H_0 \equiv L_1 = L_2 = L_3$, and $H_1 \equiv$ some L_i is different.

The same hypotheses are made for the other factors. In this way, we test in the null hypothesis that all of the population means are equal against an alternative hypothesis that there is at least one mean that is not equal to the others.

The hypothesis associated with the interaction between L and P is $H_0 \equiv LP_{i,j} = 0, \forall i,j$, and $H_1 \equiv \exists LP_{i,j} \neq 0$. Similar hypotheses can be assumed for the interaction between the other factors.

The analysis of variance table represents the initial study in a compact form, containing the sum of squares, degrees of freedom, mean square, test statistics and significance level, where non-significative factors and interactions have been removed (p-value > 0.05). These factors and interactions take part of the error component now. In this way, the results of the ANOVA III test for the Diabetic Retinopathy dataset are summarised in Table V. There are significant differences in average QWK depending on the link function and also depending on the learning rate for $\alpha = 0.05$ (p-value $= 0.000$). Moreover, an interaction between the link function and the learning rate can be recognised (p-value $= 0.001$). It means that the learning rate and the link functions have a significant impact on the optimization algorithm results.

Given that there exist significant differences between the means, we analyse now those differences. A post-hoc multiple comparison test has been performed on the mean QWK obtained. An HSD Tukey's test [19] was made under the null hypothesis that the variance of the error of the dependent variable is the same between the groups.

TABLE V
ANOVA III for the analysis of the main factors in the design of a Convolutional Ordinal Neural Network for the Retinopathy dataset.

Response variable QWK					
Source	S.S.	D.F.	M.S.	F-ratio	Sig.
Model	37.860	9	4.207	1562.840	0.000
L factor	0.057	2	0.029	10.646	0.000
P factor	0.121	2	0.060	22.468	0.000
LP factors	0.057	4	0.014	5.261	0.001
Error	0.339	126	0.003		
Total	38.199	135			

TABLE VI
Tukey's test results for the Diabetic Retinopathy dataset.

LF	LF	Mean diff.	Sig.
`logit`	`probit`	-0.002	0.011
	`clog-log`	0.012	0.000
`probit`	`logit`	0.002	0.011
	`clog-log`	0.014	0.248
`clog-log`	`logit`	-0.012	0.000
	`probit`	-0.014	0.248
LR	**LR**	**Mean diff.**	**Sig.**
10^{-2}	10^{-3}	-0.073	0.000
	10^{-4}	-0.044	0.000
10^{-3}	10^{-2}	0.073	0.000
	10^{-4}	0.029	0.023
10^{-4}	10^{-2}	0.044	0.000
	10^{-3}	-0.029	0.023

TABLE VII
ANOVA III for the analysis of the main factors in the design of a Convolutional Ordinal Neural Network for the Adience dataset.

Response variable QWK					
Source	S.S.	D.F.	M.S.	F-ratio	Sig.
Model	84.372	27	3.125	4414.006	0.000
L factor	0.156	2	0.078	110.103	0.000
P factor	0.925	2	0.462	653.163	0.000
B factor	0.040	2	0.020	28.218	0.000
LP factors	0.284	4	0.071	100.118	0.000
LB factors	0.008	4	0.002	2.837	0.028
PB factors	0.026	4	0.007	9.267	0.000
LPB factors	0.021	8	0.003	3.728	0.001
Error	0.076	108	0.001		
Total	84.449	135			

The results of this test over the test set are shown in Table VI. They show that the best link function is the complementary log-log but the `probit` link performance is close to it. Also, the best value for the learning rate parameter is 10^{-3}. The batch size is not relevant for this dataset with the values considered.

The results of the ANOVA III test for the Adience dataset are shown in Table VII. First, we observe that there exist significant differences in average QWK concerning the three factors (p-value $= 0.000$). Secondly, we found interactions between all the pairs of factors and between all the three factors together (p-values 0.000, 0.000, 0.000 and 0.001). It means that the joint action of two or three factors significantly affects the results obtained by the

TABLE VIII
Tukey's test results for the Adience dataset.

LF	LF	Mean diff.	Sig.
logit	probit	0.046	0.000
	clog-log	0.084	0.000
probit	logit	−0.046	0.000
	clog-log	0.038	0.000
clog-log	logit	−0.084	0.000
	probit	−0.038	0.000
LR	**LR**	**Mean diff.**	**Sig.**
10^{-2}	10^{-3}	−0.046	0.000
	10^{-4}	0.148	0.000
10^{-3}	10^{-2}	0.046	0.000
	10^{-4}	0.194	0.000
10^{-4}	10^{-2}	−0.148	0.000
	10^{-3}	−0.194	0.000
BS	**BS**	**Mean diff.**	**Sig.**
64	128	−0.041	0.026
	256	−0.027	0.000
128	64	0.041	0.026
	256	0.014	0.000
256	64	0.027	0.000
	128	−0.014	0.000

TABLE IX
Comparison between the best results of nominal, ordinal and previous works for the Diabetic Retinopathy dataset.

Method	$\overline{QWK}_{(SD)}$	$\overline{CCR}_{(SD)}$	$\overline{\text{1-off}}_{(SD)}$
Ordinal network	$0.582_{(0.016)}$	$0.723_{(0.004)}$	$0.868_{(0.002)}$
Nominal network	$0.498_{(0.011)}$	$0.692_{(0.012)}$	$0.854_{(0.006)}$
J. Torre et al. [20]	$0.537_{(-)}$	-	-
À. Nebot et al. [37]	$0.555_{(-)}$	-	-

TABLE X
Comparison between the best results of nominal, ordinal and previous works for the Adience dataset.

Method	$\overline{QWK}_{(SD)}$	$\overline{CCR}_{(SD)}$	$\overline{\text{1-off}}_{(SD)}$
Ordinal network	$0.881_{(0.005)}$	$0.519_{(0.013)}$	$0.894_{(0.005)}$
Nominal network	$0.787_{(0.004)}$	$0.458_{(0.008)}$	$0.800_{(0.007)}$
E. Eidinger et al. [24]	-	$0.451_{(0.026)}$	$0.807_{(0.011)}$
J.-C. Chen et al. [25]	-	$0.529_{(0.060)}$	$0.885_{(0.022)}$
G. Levi et al. [26]	-	$0.507_{(0.051)}$	$0.847_{(0.022)}$

algorithm.

As we did for the DR dataset, a post-hoc multiple comparison test has been performed on the average QWK obtained for Adience. Under the null hypothesis that the variance of the error of the dependent variable is the same between groups, an HSD Tukey's test has been applied. The results of this test over the test set are shown in Table VIII.

The results over the test set show that the best link function is the `logit` one, the best learning rate is 10^{-3} and the best batch size is 128. However, the interactions between these factors made the configuration that uses a `logit` link, $\eta = 10^{-3}$ and batch size of 64, the best configuration. It obtained a mean QWK value of 0.940 for validation and 0.881 for test. The same parameters, but using the `probit` link, achieves the second best result (0.874). The standard deviation is very low for both cases.

To sum up, the results showed that the best parameter configuration depends on the problem that is being solved. The optimal value for the batch size and the optimal link function are not the same for Retinopathy and Adience datasets. These results highlight the importance of adjusting the hyper-parameters for each problem instead of trying to find an optimal configuration for all the datasets. However, the best learning rate for both datasets were 10^{-3}. It is recommended to use this value for future datasets. The best batch size for DR was 10 while the best value for Adience was 128 (intermediate values considered). Finally, there are more interactions between the three factors for the Adience dataset than for the Diabetic Retinopathy one. This highlights the importance of making experimental designs associated with each dataset to determine the best value for each factor.

D. Comparison with nominal method and previous works

The same experiments described in Section IV were repeated using the cross-entropy instead of the QWK as loss function for the optimizer and the softmax function instead of the Cumulative Link Model for the output of the network. The evaluation metrics remains the same in order to be able to compare. There are some parameter configurations where the training process gets stagnated and a very low QWK is obtained. As we saw in Sections V-A and V-B, this problem is not found when using the ordinal method.

For the Diabetic Retinopathy dataset, the best mean value of QWK was 0.497 and was obtained when using a batch size of 10 and a learning rate of 10^{-4}.

In the case of Adience dataset, the highest QWK was 0.787 and was achieved with a batch size of 64 and a learning rate of 10^{-3}.

Finally, a comparison of the best results for each dataset for ordinal and nominal cases and previous works is shown in tables IX and X. All the results are given for the test set, except those from [20] (DR dataset), because the authors only provided validation results for 128×128 images (however, validation results are usually better than test results). The proposed ordinal model outperforms all the other alternatives in terms of QWK. The performance gain of CLM over the softmax reaches 16.8% for Diabetic Retinopathy and 11.9% for Adience dataset. The improvement of the ordinal method for Retinopathy dataset is higher than for Adience dataset. It seems that the method proposed in this work offers a more significant improvement as the given problem complexity increases.

VI. Conclusions

The first conclusion obtained from the results of our experiments is that the optimal values for the different parameters considered are problem-dependant.

The complementary log-log function offers the best results in Diabetic Retinopathy dataset while the `logit` link is the best option for the Adience dataset. These results provide an opportunity for exploring new generalised link functions that could be dynamically adapted to any problem.

The best value for the learning rate parameter for both datasets is $\eta = 10^{-3}$. It can be considered a good value for this parameter when training the model with new datasets.

Both datasets have obtained the best performance with an intermediate batch size: 10 for Diabetic Retinopathy and 128 for Adience.

Also, the statistical tests reported that there are relevant interactions between the three factors that we have take into account. The results highlight the importance of making an experimental design where all of these parameters are adjusted for each problem.

The proposed CLM has improved the performance of the deep network compared to the model that uses the softmax function and the models proposed in previous works. Also, it reduces the chance that the model gets stuck when training with some parameter configurations. So, the most significant improvements of these link functions are the performance increase, the reduction of the number of parameters configurations that should be tried to find the best one and the prevention of the over-fitting and the stagnation.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] D. Cireşan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," *arXiv preprint arXiv:1202.2745*, 2012.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE conf. computer vision and pat. recog.*, 2016, pp. 770–778.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Adv. in neural inf. proc. sys.*, 2012, pp. 1097–1105.

[5] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[7] X. Jiang, Y. Pang, X. Li, and J. Pan, "Speed up deep neural network based pedestrian detection by sharing features across multi-scale models," *Neurocomputing*, vol. 185, pp. 163–170, 2016.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[9] J. Yu, B. Zhang, Z. Kuang, D. Lin, and J. Fan, "iprivacy: image privacy protection by identifying sensitive objects via deep multi-task learning," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 5, pp. 1005–1016, 2017.

[10] C. Hong, J. Yu, J. Wan, D. Tao, and M. Wang, "Multimodal deep autoencoder for human pose recovery," *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5659–5670, 2015.

[11] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[12] Z. Li and J. Tang, "Weakly supervised deep metric learning for community-contributed image retrieval," *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 1989–1999, 2015.

[13] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European conference on computer vision*. Springer, 2014, pp. 184–199.

[14] Y. Sun, X. Wang, and X. Tang, "Deep convolutional network cascade for facial point detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 3476–3483.

[15] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[16] A. Agresti, *Analysis of ordinal categorical data*. J. Wiley & Sons, 2010, vol. 656.

[17] P. A. Gutierrez, M. Perez-Ortiz, J. Sanchez-Monedero, F. Fernandez-Navarro, and C. Hervas-Martinez, "Ordinal regression methods: survey and experimental study," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 1, pp. 127–146, 2016.

[18] R. G. Miller Jr, *Beyond ANOVA: basics of applied statistics*. Chapman and Hall/CRC, 1997.

[19] J. W. Tukey, "Comparing individual means in the analysis of variance," *Biometrics*, pp. 99–114, 1949.

[20] J. de la Torre, D. Puig, and A. Valls, "Weighted kappa loss function for multi-class classification of ordinal data in deep learning," *Pattern Recognition Letters*, vol. 105, pp. 144–154, 2018.

[21] C. Beckham and C. Pal, "Unimodal probability distributions for deep ordinal classification," *arXiv preprint arXiv:1705.05278*, 2017.

[22] Z. Niu, M. Zhou, L. Wang, X. Gao, and G. Hua, "Ordinal regression with multiple output cnn for age estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4920–4928.

[23] C. Beckham and C. Pal, "A simple squared-error reformulation for ordinal classification," *arXiv preprint arXiv:1612.00775*, 2016.

[24] E. Eidinger, R. Enbar, and T. Hassner, "Age and gender estimation of unfiltered faces," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 12, pp. 2170–2179, 2014.

[25] J.-C. Chen, A. Kumar, R. Ranjan, V. M. Patel, A. Alavi, and R. Chellappa, "A cascaded convolutional neural network for age estimation of unconstrained faces," in *Proc. of 8th IEEE Conference on Biometrics Theory, Applications and Systems (BTAS)*. IEEE, 2016, pp. 1–8.

[26] G. Levi and T. Hassner, "Age and gender classification using convolutional neural networks," in *Proc. of the IEEE conf. comp. vision and pat. rec.*, 2015, pp. 34–42.

[27] H. Li, M. Habes, and Y. Fan, "Deep ordinal ranking for multi-category diagnosis of alzheimer's disease using hippocampal mri data," *arXiv preprint arXiv:1709.01599*, 2017.

[28] Y. Liu, A. W.-K. Kong, and C. K. Goh, "Deep ordinal regression based on data relationship for small datasets," in *IJCAI*, 2017, pp. 2372–2378.

[29] A. Rios and R. Kavuluru, "Ordinal convolutional neural networks for predicting rdoc positive valence psychiatric symptom severity scores," *Journal of biomedical informatics*, vol. 75, pp. S85–S93, 2017.

[30] S. Chen, C. Zhang, M. Dong, J. Le, and M. Rao, "Using ranking-cnn for age estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5183–5192.

[31] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2002–2011.

[32] Y. Liu, A. Wai Kin Kong, and C. Keong Goh, "A constrained deep neural network for ordinal regression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 831–839.

[33] A. Pal, A. Chaturvedi, U. Garain, A. Chandra, R. Chatterjee, and S. Senapati, "Severity assessment of psoriatic plaques using deep cnn based ordinal classification," in *OR 2.0 Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis*. Springer, 2018, pp. 252–259.

[34] M. ALALI, N. M. Sharef, H. Hamdan, M. A. A. Murad, and N. A. Husin, "Multi-layers convolutional neural network for

twitter sentiment ordinal scale classification," in *International Conference on Soft Computing and Data Mining.* Springer, 2018, pp. 446–454.
[35] P. McCullagh, "Regression models for ordinal data," *Journal of the royal statistical society. Series B (Methodological)*, pp. 109–142, 1980.
[36] R. Herbrich, "Large margin rank boundaries for ordinal regression," *Advances in large margin classifiers*, pp. 115–132, 2000.
[37] À. Nebot *et al.*, "Diabetic retinopathy detection through image analysis using deep convolutional neural networks," in *A.I. Research and Development: Proc. of the 19th Int. Conf. of the Catalan Association for A.I.* IOS press, 2016, p. 58.
[38] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.
[39] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.
[40] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.
[41] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, 2013, pp. 1310–1318.
[42] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1, 2013, p. 3.
[43] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.
[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[45] D. A. Van Dyk and X.-L. Meng, "The art of data augmentation," *Journal of Computational and Graphical Statistics*, vol. 10, no. 1, pp. 1–50, 2001.
[46] A. Ben-David, "Comparison of classification accuracy using cohen's weighted kappa," *Expert Systems with Applications*, vol. 34, no. 2, pp. 825–832, 2008.
[47] M. Cruz-Ramírez, C. Hervás-Martínez, J. Sánchez-Monedero, and P. A. Gutiérrez, "Metrics to guide a multi-objective evolutionary algorithm for ordinal classification," *Neurocomputing*, vol. 135, pp. 21–31, 2014.
[48] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? a new look at signal fidelity measures," *IEEE signal processing magazine*, vol. 26, no. 1, pp. 98–117, 2009.
[49] L. N. Smith, "Cyclical learning rates for training neural networks," in *IEEE Winter Conference on Applications of Computer Vision (WACV).* IEEE, 2017, pp. 464–472.
[50] A. Senior, G. Heigold, K. Yang *et al.*, "An empirical study of learning rates in deep neural networks for speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2013, pp. 6724–6728.