Neural Processing Letters
https://doi.org/10.1007/s11063-022-10824-7

Error-Correcting Output Codes in the Framework of Deep Ordinal Classification

Javier Barbero-Gómez[1] · **Pedro Antonio Gutiérrez**[1] · **César Hervás-Martínez**[1]

Accepted: 4 April 2022

Abstract
Automatic classification tasks on structured data have been revolutionized by Convolutional Neural Networks (CNNs), but the focus has been on binary and nominal classification tasks. Only recently, ordinal classification (where class labels present a natural ordering) has been tackled through the framework of CNNs. Also, ordinal classification datasets commonly present a high imbalance in the number of samples of each class, making it an even harder problem. Focus should be shifted from classic classification metrics towards per-class metrics (like AUC or Sensitivity) and rank agreement metrics (like Cohen's Kappa or Spearman's rank correlation coefficient). We present a new CNN architecture based on the Ordinal Binary Decomposition (OBD) technique using Error-Correcting Output Codes (ECOC). We aim to show experimentally, using four different CNN architectures and two ordinal classification datasets, that the OBD+ECOC methodology significantly improves the mean results on the relevant ordinal and class-balancing metrics. The proposed method is able to outperform a nominal approach as well as already existing ordinal approaches, achieving a mean performance of $RMSE = 1.0797$ for the Retinopathy dataset and $RMSE = 1.1237$ for the Adience dataset averaged over 4 different architectures.

Keywords Ordinal classification · Convolutional neural networks · Cumulative link model · Ordinal binary decomposition

1 Introduction

There exists a large variety of classification tasks tackled in Machine Learning (ML) literature. It is natural to group them, for example, depending on the number of different class labels

✉ Javier Barbero-Gómez
jbarbero@uco.es

Pedro Antonio Gutiérrez
pagutierrez@uco.es

César Hervás-Martínez
chervas@uco.es

[1] Departamento de Informática y Análisis Numérico, Universidad de Córdoba, Campus de Rabanales, 14014 Córdoba, Córdoba, Spain

Published online: 12 May 2022

assigned to the classification samples. According to this, we differentiate between binary classification tasks (those where only two different labels are present, usually a "positive" class and a "negative" class) and multi-class classification tasks (those where more than two different labels exist).

Focusing on multi-class tasks, one could also pay attention to the relation between the class labels. Classic approaches assume all classes equally without relations between them and try to minimize simply the number of samples correctly assigned a label.

However, when an order relation between the class labels is present due to the nature of the problem itself, these tasks can be posed as "ordinal classification" (sometimes referred as "ordinal regression") tasks, which have gained popularity in the last decade. This family of problems, halfway between nominal classification and regression, presents extra information which can be exploited in order to improve performance, sometimes regarding different metrics than usual [1, 4, 16]. The benefits of this exploitation have been proven to outperform purely nominal methods in the context of unstructured data [10, 29, 30], and some methods have been proposed to search for ordinality in the class labels of apparently purely categorical datasets [24].

In this work, we propose and explore a novel general methodology for ordinal classification tasks of 2D images. This includes a generic structure for the final layers of a Convolutional Neural Network (CNN), adaptable to a wide range of already existing architectures, as well as a prediction scheme adapted to this structure and an ordinal target label encoding, both based on the Error-Correcting Output Code (ECOC) framework. Our hypothesis is that this exploitation of ordinal information in the context of image classification may improve performance, not only on ordinal metrics but also in nominal ones.

This work is structured as follows: in Sect. 2 a brief literature review on ordinal classification and CNNs is presented. In Sect. 3 a baseline nominal methodology for training CNNs to solve classification problems is posed. Then in Sect. 4 the ordinal classification framework is described and three different ordinal classification methodologies for CNNs (two already existing methods based on previous works and one novel method) are described. In Sect. 5, the experiments for the comparison of these four approaches are presented, including the datasets used for evaluation. Finally, in Sect. 6, the experiment results are shown, and Sect. 7 concludes with a discussion of these results.

2 Related Work

Early ordinal classification approaches were limited to unstructured input data, where no spatial or temporal relations exist between the inputs. Some basic approaches include using regular regression methods with rounding applied at the outputs [23] or using the label distance as a cost penalty [22]. The performance of such methods is limited because of the potentially unequal underlying distance between labels. Cumulative Link Models (CLMs) such as the Proportional Odds Model (POM) [27] or the `gologit` model [35], which not only learn a latent continuous variable but also a set of thresholds for each rank, are able to overcome this limitation. There are also adaptations of Support Vector Machines (SVMs) like SVORIM or SVOREX [7] which add ordinal constraints to the optimization of the model. Lastly, an approach known as Ordinal Binary Decomposition (OBD), where the original ordinal problem is split into a set of binary problems, has also proven to improve performance. Examples of this are the cascade linear utility model [36], where a different model solves each binary problem, or neural networks coupled with multiple outputs, one

for each binary subproblem [9, 20]. The main problem with OBD is the matter of combining the different outputs to produce a final decision.

These approaches are not suitable for structured information such as 2D images, where domain-specific feature extraction is still necessary. In this regard, CNNs provide an automatic method for extracting learned features from structured data in classification tasks. Unfortunately, due to their high number of parameters CNNs suffer easily from overfitting problems resulting in low generalization performance. In addition to classic techniques such as L_2 regularization and dropout, recent techniques include multi-stage implicit regularization [37] and network path pruning [38] to avoid this problem.

Adapting CNNs to work with ordinal information is a recent line of research, still needing extensive work. In [12, 33], a CLM has been adapted as the activation function of a single output of a CNN. In [28] a CNN architecture for solving the OBD version of an age estimation problem is proposed, with a very simple combination of the binary outputs for obtaining a rank. [25] proposes a different methodology for small datasets based on triplets of samples and majority voting. Finally, [6] proposes an improvement over [28] by bounding the maximum binary error of each output.

3 Base Nominal CNN Methodology

Nominal classification is the general framework for tasks where there is a need to assign a class label to a randomly sampled object from a specific distribution. More formally, we want to obtain a rule $r : \mathcal{X} \to \mathcal{Y}$ that associates an input vector $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^K$ to a class label $y \in \mathcal{Y} = \{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_Q\}$ in a finite set. In order to learn this relation, a dataset D is provided consisting on tuples of correctly classified samples $D = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathcal{X},\ y_i \in \mathcal{Y},\ i \in \{1, \ldots, N\}\}$.

Focusing on image classification tasks, CNNs are able to capture the spatial nature of image features, where nearby pixels have a stronger association between them than far away ones. We have considered four different well-known and competitive CNN architectures for image classification in order to have a good performance baseline: VGG11 [31], ResNet18 [17], MobileNetV3 [19] and ShuffleNetV2 [26]. We use these architectures as a baseline for traditional nominal classification.

While the specifics of each architecture varies wildly, their general design follows the following overall premise:

- First, several blocks of convolution and pooling operations are applied to the input image.
- Then, the mapped features are processed by one or more hidden fully-connected layers.
- Finally, an output layer with as many units as classes and softmax activation is used, whose value represent the probability of input sample $\mathbf{x}$ being assigned each class label $P(y = \mathcal{C}_q \mid \mathbf{x})$. These are compared to the ground truth labels of dataset D to compute a loss function ℓ and minimize it through some sort of gradient descent procedure.

3.1 Decision Rule

During evaluation of the model, the maximum probability class of $\mathbf{x}_i$ is selected as the predicted class label $\hat{y}_i$:

$$\hat{y}_i = \arg\max_{\mathcal{C}_q \in \mathcal{Y}} P(y_i = \mathcal{C}_q \mid \mathbf{x}_i), \tag{1}$$

where $P(y_i = \mathcal{C}_q \mid \mathbf{x}_i)$ is the probability of sample $\mathbf{x}_i$ being assigned label $\mathcal{C}_q$ predicted by the network.

3.2 Loss Function

For the baseline nominal methodology, categorical cross-entropy is used as the loss function ℓ during training:

$$\ell = -\frac{1}{N}\sum_{i=1}^{N}\sum_{q=1}^{Q} 1\{y_i = \mathcal{C}_q\}\log(P(y_i = \mathcal{C}_q \mid \mathbf{x}_i)), \tag{2}$$

where $1\{y_i = \mathcal{C}_q\}$ is the indicator function that is equal to 1 when $y_i = \mathcal{C}_q$ and 0 otherwise.

4 The Ordinal Classification Framework

As in a nominal classification framework, an ordinal classification task is characterised as the prediction process of assigning a label y to an input vector $\mathbf{x}$, where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^K$ and $y \in \mathcal{Y} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_Q\}$, i.e., $\mathbf{x}$ is a K-dimensional vector and y is a class label in a finite set. The goal is also to obtain some classification rule $r : \mathcal{X} \to \mathcal{Y}$ that predicts the categories of new patterns given a dataset D.

Where the ordinal framework differs from the nominal framework is in the presence of a natural ordering of the class labels: $\mathcal{C}_1 \prec \mathcal{C}_2 \prec \cdots \prec \mathcal{C}_Q$, where $\prec$ is an order relation. This is similar to regression, where $y \in \mathbb{R}$, and real values can be ordered by the $<$ operator but, in this case, the labels are discrete and include qualitative information instead of quantitative [16]. Throughout this work, the convention that $i < j \Rightarrow \mathcal{C}_i \prec \mathcal{C}_j$ always holds.

4.1 Adapting CNNs for ordinal classification

Without altering the architectures in a major way, several different options are available for introducing the ordinal information of the original dataset in the model and its training process:

(a) Using a loss function that incorporates ordinal information in the optimization procedure.
(b) Altering only the fully-connected layer phase of the architecture, maintaining all previous layers as-is.
(c) Furthermore, altering the decision rule that assigns a label to each sample when making a prediction.

In the following three sections, three different ordinal methodologies are described: two already present in the literature as well as our proposed method.

4.2 Using an ordinal loss function: Quadratic Weighted Kappa

A naive approach to integrate ordinal information in the learning process of the model consists on optimizing an order-sensitive loss function instead of the classic categorical cross-entropy.

A promising such function is the weighted Kappa metric [3] (described in Sect. 5.4), a relevant score for ordinal classifiers as it measures the rank agreement between two raters (in our case, the ground truth labels and the model outputs) based on a disagreement penalty.

This penalty is usually defined as the absolute (linear) or square (quadratic, used in the rest of this paper) difference between the rank labels. It is often used in medical diagnosis systems, where the severity of a disease presents naturally ordered stages. It is defined as:

$$(\kappa) = 1 - \frac{\sum_{i=1}^{Q}\sum_{j=1}^{Q} w_{\mathcal{C}_i,\mathcal{C}_j} p_{\mathcal{C}_i,\mathcal{C}_j}}{\sum_{i=1}^{Q}\sum_{j=1}^{Q} w_{\mathcal{C}_i,\mathcal{C}_j} e_{\mathcal{C}_i,\mathcal{C}_j}}, \tag{3}$$

where $w_{\mathcal{C}_i,\mathcal{C}_j}$ is the disagreement cost when $y = \mathcal{C}_i$ and $\hat{y} = \mathcal{C}_j$ ($w_{\mathcal{C}_i,\mathcal{C}_j} = (i-j)^2$ for the quadratic case), and $p_{\mathcal{C}_i,\mathcal{C}_j}$ and $e_{\mathcal{C}_i,\mathcal{C}_j}$ are the observed agreement and expected agreement due to chance for classes $\mathcal{C}_i$ and $\mathcal{C}_j$, respectively. A larger κ value corresponds with a better agreement and vice versa, and so it is a metric to be maximised.

Unfortunately, like is the case with accuracy, this metric is not continuous and is expressed in terms of discrete labels, preventing the application of gradient descent methods. In [8] a proposal is made to adapt this metric as a loss function to be used in CNN model training maintaining the architecture of the network as well as the decision rule.

4.2.1 Loss Function

First, κ is expressed in terms of probabilities instead of class labels, maintaining the penalty matrix $w_{\mathcal{C}_i,\mathcal{C}_j}$ but substituting $p_{\mathcal{C}_i,\mathcal{C}_j}$ and $e_{\mathcal{C}_i,\mathcal{C}_j}$ for the probability outputs of the model:

$$\hat{\kappa} = 1 - \frac{\sum_{i=1}^{N}\sum_{q=1}^{Q} w_{y_i,\mathcal{C}_q} P(y_i = \mathcal{C}_q \mid \mathbf{x}_i)}{\sum_{j=1}^{Q}\frac{N_j}{N}\sum_{k=1}^{Q}(w_{\mathcal{C}_j,\mathcal{C}_k}\sum_{i=1}^{N} P(y_i = \mathcal{C}_k \mid \mathbf{x}_i))}, \tag{4}$$

where N_j is the number of samples with class label $\mathcal{C}_j$ in the dataset D.

Then, in order to pose it as a minimization problem, loss ℓ is defined as:

$$\ell = \log(1 - \hat{\kappa}), \text{ where } \ell \in (-\infty, \log 2]. \tag{5}$$

Further derivation and a more in-depth discussion can be found in [8].

4.2.2 Decision Rule

In the same manner as the nominal approach of Sect. 3, the maximum probability class of $\mathbf{x}_i$ is selected as the predicted class label $\hat{y}_i$.

4.3 The Cumulative Link Model Approach

For the CLM framework (family of models which includes the POM [27]), only a small modification to the baseline nominal model is needed: the output is reduced to only a single unit in the last layer, and the `logit` cumulative link function is used as the activation function instead of softmax:

$$P(y \preceq \mathcal{C}_q \mid \mathbf{x}) = \sigma(b_q - f(\mathbf{x})) \ , \ 1 \leq q < Q, \tag{6}$$

where $P(y \preceq \mathcal{C}_q \mid \mathbf{x})$ is the probability of sample $\mathbf{x}_i$ being assigned label $\mathcal{C}_q$ or lower predicted by the network, $f(\mathbf{x})$ is the single output of the model, σ is the sigmoid function and b_q is one of the $Q-1$ thresholds learned as additional parameters. Note that cumulative probabilities $P(y \preceq \mathcal{C}_q \mid \mathbf{x})$ are predicted by this function instead of individual ones like $P(y = \mathcal{C}_q \mid \mathbf{x})$.

4.3.1 Decision Rule

During evaluation, elementary probability rules are used to combine the cumulative probabilities from Eq. (6) into individual probabilities [15]:

$$P(y_i = \mathcal{C}_q \mid \mathbf{x}_i) = \begin{cases} P(y_i \preceq \mathcal{C}_1 \mid \mathbf{x}_i), & \text{if } q = 1, \\ P(y_i \preceq \mathcal{C}_q \mid \mathbf{x}_i) - P(y_i \preceq \mathcal{C}_{q-1} \mid \mathbf{x}_i), & \text{if } 1 < q < Q, \\ 1 - P(y_i \preceq \mathcal{C}_{Q-1} \mid \mathbf{x}_i), & \text{if } q = Q, \end{cases} \tag{7}$$

and the maximum probability class is then selected as the predicted label $\hat{y}_i$:

$$\hat{y}_i = \underset{\mathcal{C}_1 \preceq \mathcal{C}_q \preceq \mathcal{C}_Q}{\arg\max} \; P(y_i = \mathcal{C}_q \mid \mathbf{x}_i). \tag{8}$$

4.3.2 Loss Function

Cross-entropy loss is used as the loss function in the same manner as in the nominal model.

4.4 Our approach: Ordinal Binary Decomposition

For our ordinal approach, we decompose the original Q-class ordinal problem into $Q - 1$ binary decision problems, what is known as Ordinal Binary Decomposition (OBD). Each q problem consists on deciding whether $y \succ \mathcal{C}_q$ conditioned to sample $\mathbf{x}$ ($1 \leq q < Q$) (this is referred to as the "Ordered partitions" scheme in [16]).

To adapt the outputs of the model to this, the final fully-connected block is substituted by $Q - 1$ fully-connected blocks, each one with a single output unit with sigmoid activation[1]. Each of the $Q - 1$ outputs of the model o_q is trying to predict the probability $P(y \succ \mathcal{C}_q \mid \mathbf{x})$. The result of this modification is obtaining $Q - 1$ different models, which share their convolutional feature extraction parameters and are trained simultaneously.

4.4.1 Decision Rule

In the case of the OBD models, because the outputs are not individual probabilities but cumulative ones ($o_k = P(y \succ \mathcal{C}_k \mid \mathbf{x})$), the decision rule requires combining several outputs. Moreover, these probabilities may be inconsistent: nothing forces them to fulfil basic probability properties like $P(y \succ \mathcal{C}_i) \geq P(y \succ \mathcal{C}_{i+1})$ and $\sum_{i=1}^{Q} P(y = \mathcal{C}_i) = 1$. For this reason, Eq. (7) cannot be applied as for the CLM.

In order to circumvent this problem, a stable approach based on the ECOC framework is used: the ideal output vector $\mathbf{v}(\mathcal{C}_i)$ for each class $\mathcal{C}_i$ is considered, $\mathbf{v}(\mathcal{C}_i) = (c_1, \ldots, c_{Q-1})$ where $c_j = 1\{\mathcal{C}_j \prec \mathcal{C}_i\}$, i.e. a vector with ones in all positions corresponding with classes which are lower than $\mathcal{C}_i$ in the ordinal scale. This makes the ideal output vector for a sample $\mathbf{x}_i$ with label $y_i = \mathcal{C}_k$ be:

$$\mathbf{v}(\mathcal{C}_k) = (c_1, ..., c_{k-1}, c_k, ..., c_{Q-1}) = (1, ..., 1, 0, ..., 0), \tag{9}$$

i.e., for a 4 class ordinal problem with labels $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$, and $\mathcal{C}_4$ the ideal outputs would be $\mathbf{v}(\mathcal{C}_1) = (0, 0, 0)$, $\mathbf{v}(\mathcal{C}_2) = (1, 0, 0)$, $\mathbf{v}(\mathcal{C}_3) = (1, 1, 0)$, and $\mathbf{v}(\mathcal{C}_4) = (1, 1, 1)$.

[1] For architectures where an extra hidden layer of size H is present (like VGG11 and MobileNetV3), these are reduced to $\lfloor H/(Q-1) \rfloor$ units in order to maintain a similar number of parameters.

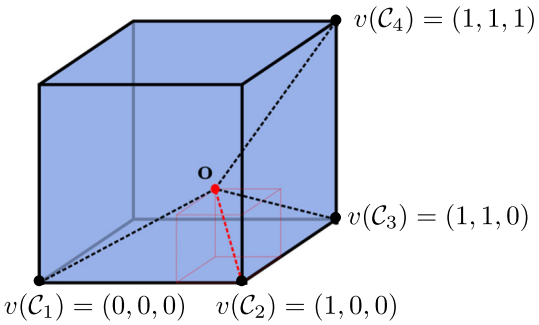

Fig. 1 The model output vector $\mathbf{o}$ (dot in red) for sample $\mathbf{x}$ and its distance to each of the ideal class vectors (dashed lines), illustrated as a 3D graphic where each dimension represents each of the three model outputs. The closest point is $v(\mathcal{C}_2)$ (marked in red), and thus $\mathbf{x}$ is assigned label $\mathcal{C}_2$

The decision rule is based on determining the ideal vector which minimizes the distance to the obtained output vector $\mathbf{o}$:

$$\hat{y}_i = \underset{\mathcal{C}_1 \preceq \mathcal{C}_q \preceq \mathcal{C}_Q}{\arg\min} \left\| \mathbf{o} - \mathbf{v}(\mathcal{C}_q) \right\|_2, \tag{10}$$

where $\|\cdot\|_2$ is the L_2 norm. This distance metric is selected in order to align it with the loss function of the optimization process.

As an example to illustrate this prediction criterion, assume a 4 class ordinal problem like the one previously mentioned. For sample $\mathbf{x}$, let the output of the model be the 3 dimensional vector $\mathbf{o} = (0.8, 0.3, 0.2)$. The distance to each ideal class vector would be computed as:

$$\begin{aligned}
\|\mathbf{o} - \mathbf{v}(\mathcal{C}_1)\|_2 &= \|(0.8 - 0, 0.3 - 0, 0.2 - 0)\|_2 = 0.77, \\
\|\mathbf{o} - \mathbf{v}(\mathcal{C}_2)\|_2 &= \|(0.8 - 1, 0.3 - 0, 0.2 - 0)\|_2 = 0.17, \\
\|\mathbf{o} - \mathbf{v}(\mathcal{C}_3)\|_2 &= \|(0.8 - 1, 0.3 - 1, 0.2 - 0)\|_2 = 0.57, \\
\|\mathbf{o} - \mathbf{v}(\mathcal{C}_4)\|_2 &= \|(0.8 - 1, 0.3 - 1, 0.2 - 1)\|_2 = 1.17.
\end{aligned} \tag{11}$$

This process is illustrated in Fig. 1. The vector closest to $\mathbf{o}$ is $\mathbf{v}(\mathcal{C}_2)$ and thus, sample $\mathbf{x}$ would be assigned the class label $\hat{y} = \mathcal{C}_2$.

4.4.2 Loss Function

For the OBD methodology, categorical cross-entropy has been substituted by the Mean Squared Error loss because it copes better with the distance function used for the ECOC decision [2]:

$$\ell = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{Q-1} (1\{y_i \succ \mathcal{C}_k\} - P(y_i \succ \mathcal{C}_k \mid \mathbf{x}_i))^2. \tag{12}$$

where $1\{y_i \succ \mathcal{C}_k\}$ is the indicator function that is equal to 1 when $y_i \succ \mathcal{C}_k$ and 0 otherwise, and $P(y_i \succ \mathcal{C}_k \mid \mathbf{x}_i)$ is the probability that $y_i \succ \mathcal{C}_k$ predicted by the network given a sample $\mathbf{x}_i$.

The four methodologies described in this section are illustrated in Fig. 2.

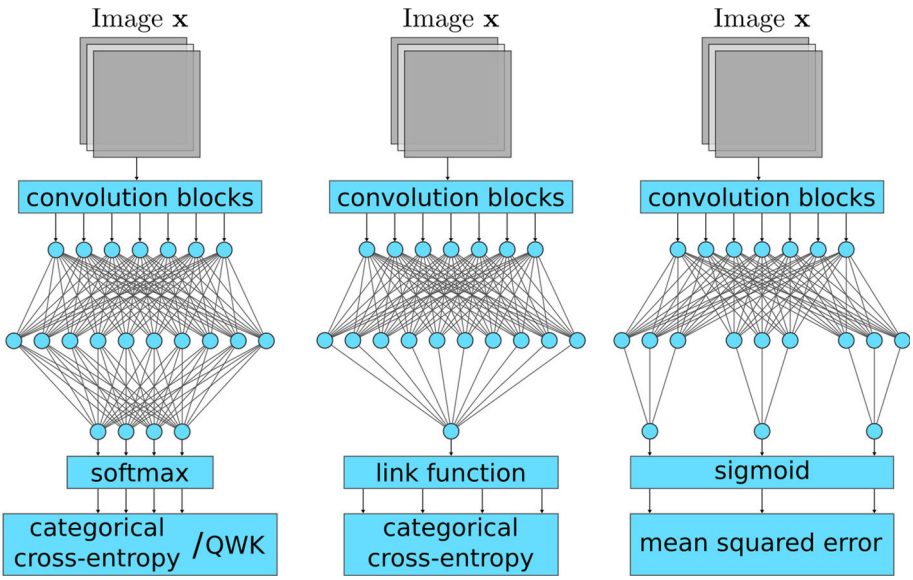

Fig. 2 The four compared methodologies, from left to right: the baseline nominal architecture (both using categorical cross-entropy as well as QWK as the loss function), CLM, and our proposal, OBD

5 Experiment Design

5.1 Datasets

The effects of the four described methodologies will be tested against the following two different datasets, chosen specifically for the ordinal nature of their class labels and an acute class imbalance.

5.1.1 Diabetic Retinopathy Dataset

The diabetic retinopathy dataset from Kaggle[2] (referred to as "Retinopathy" from now on) consists on a total of 88 702 retina images labelled by a clinician on a 0 to 4 scale evaluating the presence of Diabetic Retinopathy (DR), an eye disease present in a large proportion of diabetes patients. It contains 65 343 images labelled as No DR, 6205 images labelled as Mild DR, 13 153 images labelled as Moderate DR, 2087 images labelled as Severe DR, and 1914 images labelled as Proliferative DR. The task consists on predicting the clinician label using the colour image of the retina. Three sample images can be seen in Fig. 3. All images have been normalized down to a size of 128×128 pixels.

5.1.2 Adience Faces Dataset

The Adience faces dataset for age classification [11] (referred to simply as "Adience" from now on) is composed of 26 580 photos of 2284 different subjects extracted from real online albums and automatically cropped and aligned. 17 702 of these photos have an age label

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

Example from class 0 Example from class 2 Example from class 4

Fig. 3 Sample images from the Retinopathy dataset

Example from class 1 Example from class 4 Example from class 7

Fig. 4 Sample images from the Adience faces dataset

attached, referring to one of 8 different age groups of increasing value: 0–2 years, 4–6 years, 8–13 years, 15–20 years, 25–32 years, 38–43 years, 48–53 years, and 60 years and up. The task consists on assigning one of these 8 age labels to each photo. A sample of this images can be seen in Fig. 4. As a preprocessing step, all images have been resized down to 256×256 pixels.

5.2 Methodologies and Validation Scheme

Four different methodologies are tested against each other:

- The baseline nominal architecture, with categorical cross-entropy loss function.
- The same architecture, but substituting the loss function by the Quadratic Weighted Kappa (QWK) function described in Sect. 4.2.
- The CLM approach, as described in Sect. 4.3.
- The OBD approach with ECOC decision rule, as described in Sect. 4.4.

All of these are applied to all four of the previously mentioned architectures (VGG11, ResNet18, MobileNetV3 and ShuffleNetV2), yielding a total of sixteen different experiments for each of the two datasets.

In order to obtain a statistically significant result to test the hypotheses, each experiment is repeated 30 times on 30 different holdout splits of the original dataset, where 80% of samples are used for training and 20% are used for model evaluation. This split is performed in a stratified fashion, preserving the original proportion of the classes of the original dataset in the subsets. For the Retinopathy dataset this leaves 70 962 training samples (of which 7096

Table 1 Number of trainable parameters and total memory size of the trained models for each methodology and architecture

		Adience dataset	Retinopathy dataset
VGG11	Nominal/QWK	128 799 112 (6163 MB)	128 786 821 (1931 MB)
	CLM	128 770 440 (6163 MB)	128 770 437 (1931 MB)
	OBD	114 363 707 (6106 MB)	116 187 524 (1880 MB)
ResNet18	Nominal/QWK	11 180 616 (3838 MB)	11 179 077 (993 MB)
	CLM	11 177 032 (3838 MB)	11 177 029 (993 MB)
	OBD	11 180 103 (3838 MB)	11 178 564 (993 MB)
MobileNetV3	Nominal/QWK	4 212 280 (6697 MB)	4 208 437 (1690 MB)
	CLM	4 203 320 (6697 MB)	4 203 317 (1690 MB)
	OBD	4 197 547 (6697 MB)	4 203 316 (1690 MB)
ShuffleNetV2	Nominal/QWK	5 361 388 (5711 MB)	5 355 241 (1444 MB)
	CLM	5 347 052 (5711 MB)	5 347 049 (1444 MB)
	OBD	5 359 339 (5711 MB)	5 353 192 (1444 MB)

are reserved for validation) and 17 740 test samples in each split. In the case of the Adience dataset, 14 161 are used for training (of which 1416 are reserved for validation) and 3541 are used for evaluation.

5.3 Training Scheme

In all experiments, weights are initialized randomly using the He initialization scheme described in [18]. They are then adjusted using the Adam method [21] with a learning rate $\eta = 1 \times 10^{-4}$.

In the case of VGG11, both dropout ($p = 0.5$) and L_2 regularization (with a weight of 5×10^{-4}) are applied only in the fully-connected layers as in the original paper [31]. For ResNet18, batch normalization is applied after every convolution operation and L_2 penalty (with a weight of 1×10^{-4}) is added to all mappings [17]. The number of trainable parameters for each model is available on Table 1.

In order to help overcome the class imbalance, class weighting is applied to the loss function based on N_q (number of training samples for class $\mathcal{C}_q$):

$$w_q = \frac{e^{-CN_q}}{\sum_{i=1}^{Q} e^{-CN_i}}, \tag{13}$$

where C is a constant defined as $C = 3 \times 10^{-5}$. This weight w_q is multiplied by the loss contribution of each sample with a ground truth label of $\mathcal{C}_q$.

Before training, 10% of training samples are reserved for validation, again selected in a stratified fashion according to the class labels. Model weights are updated in batches of 72 training samples and loss performance is monitored on both training and validation. If validation performance does not increase for 5 full epochs, training is halted, and the best performing parameters over the validation set are restored.

The code used to perform the experiments can be accessed through GitHub[3].

[3] https://github.com/ayrna/ordinal-cnn-ecoc.

5.4 Performance Metrics

The classical performance metric in classification tasks is the Correct Classification Rate *(CCR)*. However, given that both datasets present a very high class imbalance, the traditional *CCR* is not a representative measure of model performance: for example, in the case of the Retinopathy dataset, a dummy classifier that always assign the majority class label (class 0) would obtain a *CCR* of 73%.

In order to monitor global per-class performance, metrics such as the Average Area Under the Receiver Operating Characteristic (ROC) curve (*AvAUC*), minimum sensitivity (*MS*) and geometric mean of the sensitivities (*GMS*) [10] will also be included.

Also, for ordinal classification problems, rank agreement metrics including the Root of Mean Squared Error *(RMSE)* (comparing actual and predicted labels, represented as consecutive integers in the ordinal scale), Spearman's rank correlation coefficient (r_s) [5] or the Quadratic Weighted Cohen's Kappa (κ) [3] (described in Eq. (3)) have been selected as well for evaluation:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(O(\hat{y}_i) - O(y_i))^2}, \tag{14}$$

$$r_s = \frac{\mathrm{Cov}(O(y), O(\hat{y}))}{\sigma_{O(y)}\sigma_{O(\hat{y})}} \tag{15}$$

where $O(\mathcal{C}_q) = q$ is the ordinal number of label $\mathcal{C}_q$, $\mathrm{Cov}(O(y), O(\hat{y}))$ is the covariance between the ground truth labels ordinal numbers and the predicted labels ordinal numbers, and $\sigma_{O(y)}$ and $\sigma_{O(\hat{y})}$ is their standard deviation.

An illustrated example of the global experimentation procedure can be found in Fig. 5.

In Sect. 6, mean results and standard deviation are reported for each methodology. Then, statistical hypothesis testing will be performed in order to discern the effects of the different factors and conclude whether the OBD methodology shows a significant improvement over the other two.

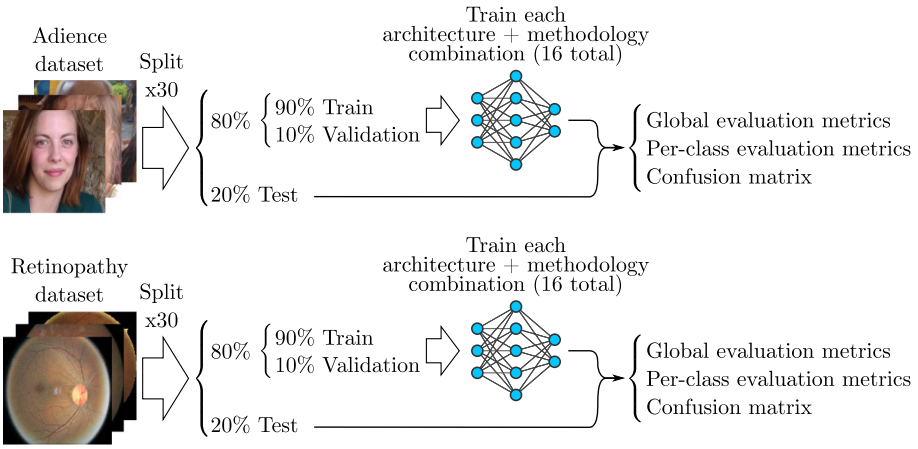

Fig. 5 The general experimentation procedure as described in Sect. 5

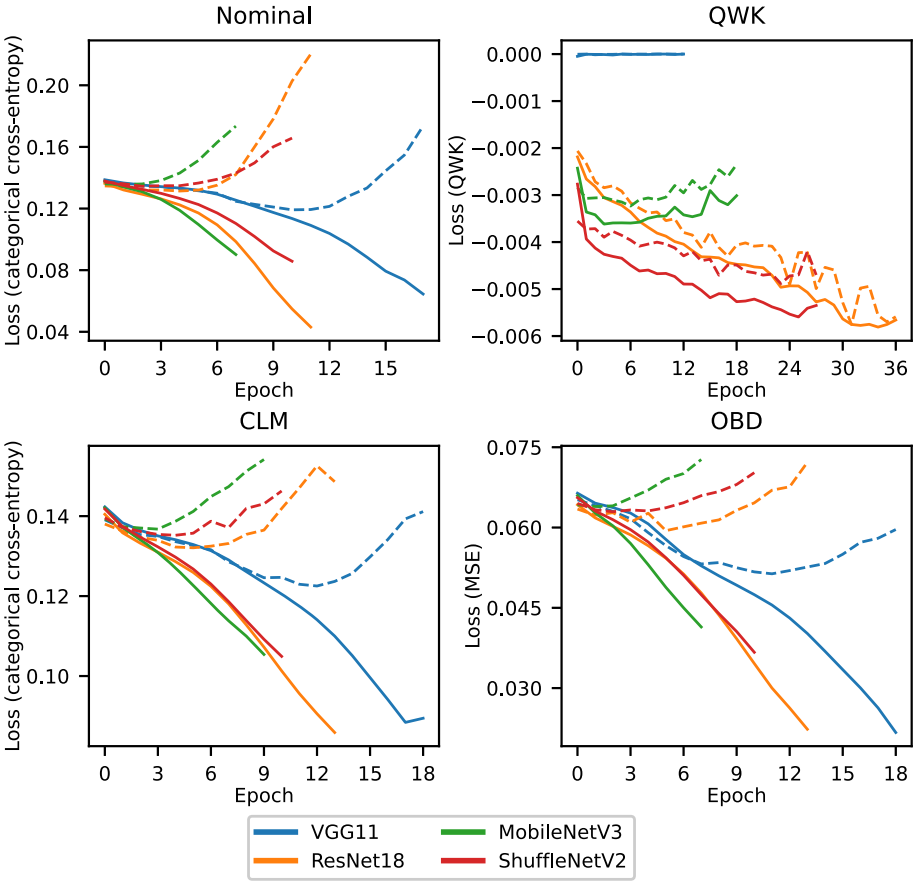

Fig. 6 Average training curves for each model and methodology when applied to the diabetic retinopathy dataset. Train loss is shown as solid lines and validation loss as dashed lines. The average is calculated over all executions which reach the corresponding iteration

6 Results

The average of the training curves over all 30 repetitions is shown in Figs. 6, 7.[4] Note how the QWK methodology fails to converge when used in conjunction with the VGG11 architecture: the high depth of this architecture makes the gradients disappear in the backpropagation phase of training, something known as the "vanishing gradients" problem. All the other architectures tested implement residual paths into the network, allowing them to avoid this problem [34]. Note how the OBD methodology does not alter the depth of the CNN model, so it will never cause this problem by itself.

Additionally, in Figs. 8, 9 the training time for each methodology and architecture is shown. In accordance to the number of parameters Table 1, the VGG11 architecture takes the longest time to train compared to the other three, which all take a similar time. Regarding

[4] The original experiment results can be checked in the GitHub repository: https://github.com/ayrna/ordinal-cnn-ecoc/blob/main/results.xlsx..

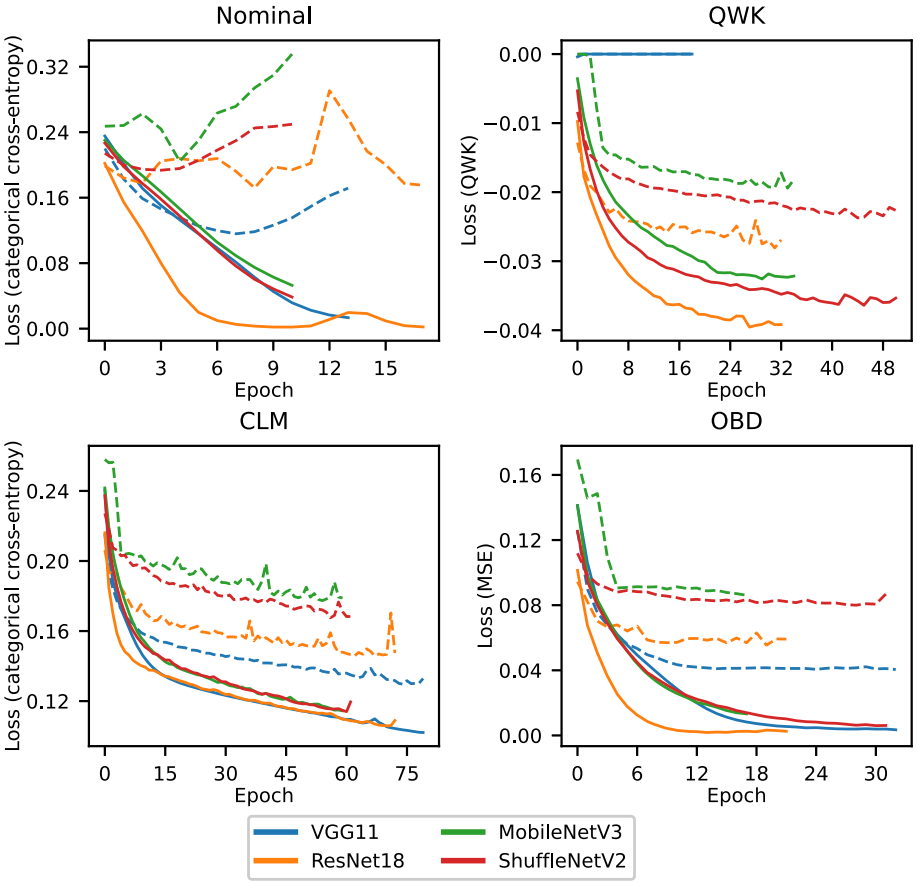

Fig. 7 Average training curves for each model and methodology when applied to the Adience dataset. Train loss is shown as solid lines and validation loss as dashed lines. The average is calculated over all executions which reach the corresponding iteration

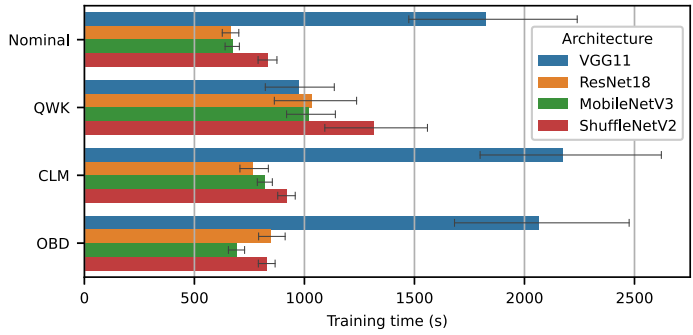

Fig. 8 Average training time of each methodology and architecture for the Retinopathy dataset. Error bars indicate ± the standard deviation

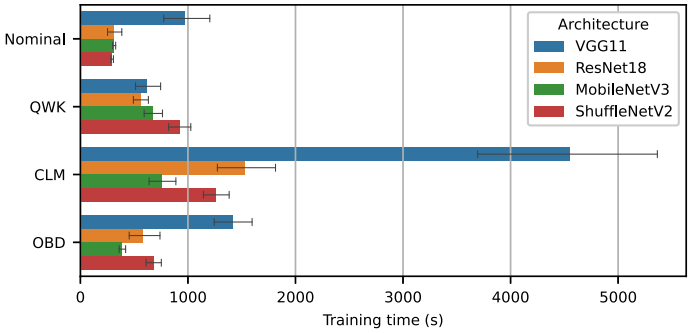

Fig. 9 Average training time of each methodology and architecture for the Adience dataset. Error bars indicate ± the standard deviation

Table 2 Average experimental results for each of the four methodologies on the test sets of both datasets. Metrics to maximize are marked with (↑) and metrics to minimize with (↓). Best results are highlighted in bold and second best in italics

Retinopathy	Nominal		QWK		CLM		OBD	
	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
CCR (↑)	*0.6668*	0.0344	0.3923	0.1778	**0.6913**	0.0219	0.4775	0.0986
AvAUC (↑)	*0.6768*	0.0560	0.5168	0.0166	0.6703	0.0519	**0.6806**	0.0560
MS (↑)	*0.0078*	0.0147	0.0000	0.0000	0.0000	0.0000	**0.0598**	0.0739
GMS (↑)	*0.0531*	0.0629	0.0000	0.0000	0.0000	0.0000	**0.1442**	0.1276
RMSE (↓)	1.1394	0.0624	1.1025	0.2966	*1.0932*	0.0610	**1.0797**	0.0748
r_s (↑)	0.2204	0.0949	0.0927	0.0543	*0.2357*	0.0938	**0.2557**	0.0898
ˇ (↑)	0.2432	0.1150	0.0766	0.0461	*0.2645*	0.1197	**0.2834**	0.1220
Adience	Nominal		QWK		CLM		OBD	
	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
CCR (↑)	0.5233	0.1090	0.3572	0.1556	*0.5264*	0.0848	**0.5573**	0.1137
AvAUC (↑)	0.8406	0.0705	0.6724	0.0997	*0.8502*	0.0646	**0.8562**	0.0622
MS (↑)	*0.1111*	0.1377	0.0000	0.0000	0.0000	0.0000	**0.2153**	0.1505
GMS (↑)	*0.2706*	0.2480	0.0000	0.0000	0.0000	0.0000	**0.4346**	0.1609
RMSE (↓)	1.3910	0.2885	1.7373	0.9177	*1.1477*	0.2482	**1.1237**	0.2372
r_s (↑)	0.7063	0.1190	0.5753	0.3352	**0.8009**	0.0800	*0.7980*	0.0850
ˇ (↑)	0.7101	0.1233	0.5860	0.3417	**0.8111**	0.0803	*0.8099*	0.0825

the methodologies, while the nominal approach usually takes less time than the ordinal ones, the OBD methodology is a close second in speed.

The average experimental results for each experiment are shown in Appendix A (Tables 4–11) and the mean over all four architectures is summarized in Table 2 for convenience. It can be noted that for the Retinopathy dataset the CLM models are able to improve ordinal metrics by a little, at the cost of worsening metrics related to the imbalance problem (*AvAUC*, *MS*, and *GMS*). Meanwhile, the OBD models improve the ordinal metrics further while also improving class balancing metrics. This is done at the cost of worsening *CCR*, but only because of the high class imbalance. In the case of the Adience dataset the OBD models still achieve a higher score in class-balancing metrics, although r_s and κ are worsened slightly.

From the confusion matrices it can be noted that, although Table 2 shows that the CLM improves on the ordinal metrics on the Adience dataset, it fails on every class balancing metric compared to the nominal model, as it ignores both classes 1 and 3 in the Retinopathy dataset, as well as classes 3, 5 and 6 in the Adience dataset. The OBD model, on the other hand, is able to improve both class balancing and ordinal metrics. This is achieved at the cost of losing some performance on the extreme classes, but note how sensitivity and precision never fall to zero when using the OBD model on any class, that is, no class is ignored systematically. This is easily seen on the confusion matrices for each methodology and architecture shown in Appendix A (Figs. 10–17).

6.1 Statistical Analysis

To determine the statistical significance of the mean differences observed for each classifier, each architecture and each dataset, we have carried out a parametric Analysis of Variance (ANOVA) test [13, 14] for each of the evaluated metrics. The three factors considered for the experimental design are: (i) the database (Adience and Retinopathy), (ii) the CNN network architecture (VGG11, ResNet18, MobileNetV3 and ShuffleNetV2) and (iii) the methodology (nominal, QWK, CLM and OBD).

For each combination of these three factors we have repeated the experiment 30 times with different data splits and weight initialization seeds. We have tested, using the Kolmogorov-Smirnov test for all metrics mentioned in Sect. 5.4, whether the null hypothesis stating that the results are drawn from a normal distribution cannot be rejected (for a significance level of $\alpha = 0.05$). This is true for all metrics except *MS* and *GMS*, namely the Quadratic Weighted Cohen's Kappa (κ), *AvAUC*, *RMSE*, Spearman's rank correlation coefficient (r_s) and *CCR*. Only these metrics will be considered for the subsequent analysis, given that ANOVA is a parametric test and can only be applied to normally distributed variables.

After this, ANOVA is performed for these five metrics. The ANOVA tables are available in Appendix B. According to this analysis, for all normally distributed metrics there exist significant differences in the mean value (for a significance level of $\alpha = 0.05$) concerning the three individual factors (Dataset, Architecture and Methodology, all p-values < 0.001). Then, we also found significant interactions between all the pairs of factors (p-values < 0.001) and between all three factors (p-value < 0.001). This shows that:

1. the impact of the architecture and the methodology varies across datasets,
2. the architecture significantly affects performance,
3. the effect of the methodology is affected by the CNN architecture (that is, some architectures are better suited for each methodology), and
4. the methodology alone affects the performance, OBD being in the lead according to the mean results of Table 2.

That is why we now analyse the magnitude of those differences according to the Methodology factor. A post-hoc Tukey's HSD multiple comparison test [32] has been performed on each of the metrics shown to be affected by this factor. The purpose of this test is to group each of the methodologies into groups of significantly similar performance, where each group is significantly different than the rest. The results of this test are summarized in Table 3 by grouping the methodologies in subsets according to their performance on each metric. The first subset contains the worst methodology, while the last one includes the best methodologies.

Note that for κ, *AvAUC*, and r_s the OBD methodology has a significantly better mean performance than the other three methodologies. For the *RMSE* metric both CLM and OBD

Table 3 Results of the Tukey's HSD test for all tested metrics. Methodologies are grouped such that the elements within a subset are not significantly different, while the differences between each group are significant. The first subset contains the worst methodologies, while the last subset groups the best methodologies. The best performing subset is highlighted in bold

κ					*AvAUC*			
Methodology	Subsets				Methodology	Subsets		
	1	2	3	4		1	2	3
QWK	0.33127				QWK	0.5946207		
Nominal		0.4766773			Nominal		0.7587429	
CLM			0.5377609		CLM		0.7602492	
OBD				**0.5466245**	**OBD**			**0.7684145**
p-values	1.000	1.000	1.000	1.000	*p*-values	1.000	0.535	1.000

RMSE				r_s				
Methodology	Subsets			Methodology	Subsets			
	1	2	3		1	2	3	4
QWK	1.4199260			QWK	0.3340351			
Nominal		1.2651859		Nominal		0.4633777		
CLM			**1.1204655**	CLM			0.5182891	
OBD			**1.1017190**	**OBD**				**0.5268579**
p-values	1.000	1.000	0.583	*p*-values	1.000	1.000	1.000	1.000

CCR			
Methodology	Subsets		
	1	2	3
QWK			
OBD		0.5174322	
Nominal			**0.595**
CLM			**0.609**
p-values	1.000	1.000	0.083

exhibit similar performance, but significantly better than the other two methodologies. Finally, for *CCR* both CLM and the nominal methodology perform similarly and better than OBD and QWK.

7 Conclusions and Future Work

A new ordinal CNN architecture based on Ordinal Binary Decomposition has been proposed, as well as a decision scheme based on ECOC, showing that it is able to significantly outperform a purely nominal approach as well as already existing ordinal approaches, especially when considering highly imbalanced scenarios like medical datasets and web-scraped datasets. Specifically, the proposed OBD methodology is able to improve both class balancing and ordinal metrics such as *RMSE*, Spearman's rank correlation coefficient and Quadratic Weighted Cohen's Kappa. This methodology is easy to adapt to any other ordinal tasks.

While the tested architectures are widely established and overall well performing models, different and more novel architectures could also be adapted in the same manner. This adds

a new fairly generic tool for classification tasks where ordinal information can be exploited. Moreover, this modification does not increase the number of parameters or memory consumption of the network and does not significantly increase the running time for training, making it applicable in memory limited scenarios.

In a future work, more complex data structures like 3D images can be studied. This is possible because the needed modifications only alter the latter stages of the network, allowing for arbitrary input shapes. Also, even though we have been able to improve on class imbalance sensitive metrics, further work is necessary, as can be noted from the confusion matrices. Better class balancing approaches than loss weighting, such as a data augmentation scheme sensitive to ordinal information, can be applied in order to improve on this.

Acknowledgements This work has been partially subsidised by the "Agencia Estatal de Investigación" (Spain) [grant reference: PID2020-115454GB-C22/AEI/10.13039/501100011033], the "Consejería de Salud y Familias" (Junta de Andalucía) [grant reference: PS-2020-780] and the "Consejería de Transformación Económica, Industria, Conocimiento y Universidades" (Junta de Andalucía) y Programa Operativo "FEDER 2014-2020" [grant references: UCO-1261651 and PY20_00074]. Javier Barbero-Gómez research has been subsidised by the FPI Predoctoral Program of the "Ministerio de Ciencia, Innovación y Universidades" (Spain) [grant reference PRE2018-085659].

Author Contributions JB-G: Writing – original draft, Conceptualization, Methodology, Software, Validation, Investigation. P-AG: Writing – review & editing, Supervision. CH-M: Formal analysis, Writing – review & editing, Supervision.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Declarations

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

Appendix A: Experimentation Result Tables and Confusion Matrices

See Tables 4, 5, 6, 7, 8, 9, 10, 11. See Figs. 10, 11, 12, 13, 14, 15, 16, 17.

Table 4 Mean results using the VGG11 architecture for the Retinopathy dataset. Metrics to maximize are marked with (↑) and metrics to minimize with (↓). Best results are highlighted in bold and second best in italics

Retinopathy VGG11	Nominal		QWK		CLM		OBD	
	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
CCR (↑)	*0.6898*	0.0227	0.5397	0.3036	**0.7159**	0.0104	0.5956	0.0303
AvAUC (↑)	*0.7605*	0.0081	0.5119	0.0119	0.7482	0.0051	**0.7644**	0.0057
MS (↑)	*0.0008*	0.0019	0.0000	0.0000	0.0000	0.0000	**0.1723**	0.0475
GMS (↑)	*0.0385*	0.0626	0.0000	0.0000	0.0000	0.0000	**0.3141**	0.0219
RMSE (↓)	1.0604	0.0449	1.3039	0.5518	*0.9990*	0.0220	**0.9673**	0.0287
r_s (↑)	0.3679	0.0174	0.0030	0.0092	*0.3791*	0.0103	**0.3931**	0.0126
ˇ (↑)	0.4206	0.0196	0.0013	0.0041	*0.4437*	0.0138	**0.4709**	0.0163

Table 5 Mean results using the ResNet18 architecture for the Retinopathy dataset. Metrics to maximize are marked with (↑) and metrics to minimize with (↓). Best results are highlighted in bold and second best in italics

Retinopathy ResNet18	Nominal		QWK		CLM		OBD	
	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
CCR (↑)	*0.6567*	0.0301	0.3331	0.0561	**0.6861**	0.0140	0.5195	0.0429
AvAUC (↑)	*0.6911*	0.0078	0.5210	0.0211	0.6824	0.0060	**0.6953**	0.0063
MS (↑)	*0.0257*	0.0197	0.0000	0.0000	0.0000	0.0000	**0.0590**	0.0217
GMS (↑)	*0.1280*	0.0451	0.0000	0.0000	0.0000	0.0000	**0.2012**	0.0272
RMSE (↓)	1.1519	0.0418	**1.0351**	0.0107	1.1017	0.0262	*1.0833*	0.0286
r_s (↑)	0.2303	0.0168	0.1237	0.0204	*0.2495*	0.0127	**0.2721**	0.0108
ˇ (↑)	0.2574	0.0190	0.0994	0.0205	*0.2871*	0.0184	**0.3053**	0.0172

Table 6 Mean results using the MobileNetV3 architecture for the Retinopathy dataset. Metrics to maximize are marked with (↑) and metrics to minimize with (↓). Best results are highlighted in bold and second best in italics

Retinopathy MobileNetV3	Nominal		QWK		CLM		OBD	
	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
CCR (↑)	*0.6608*	0.0374	0.3549	0.0559	**0.6948**	0.0165	0.4250	0.0665
AvAUC (↑)	*0.6178*	0.0087	0.5179	0.0188	0.6131	0.0059	**0.6213**	0.0071
MS (↑)	**0.0005**	0.0015	*0.0000*	0.0000	0.0000	0.0000	0.0000	0.0000
GMS (↑)	**0.0059**	0.0129	*0.0000*	0.0000	0.0000	0.0000	0.0000	0.0000
RMSE (↓)	1.1673	0.0419	**1.0410**	0.0153	1.1279	0.0212	*1.1266*	0.0327
r_s (↑)	0.1277	0.0184	0.1145	0.0160	*0.1317*	0.0131	**0.1634**	0.0126
ˇ (↑)	*0.1307*	0.0272	0.0966	0.0170	0.1267	0.0225	**0.1663**	0.0174

Table 7 Mean results using the ShuffleNetV2 architecture for the Retinopathy dataset. Metrics to maximize are marked with (↑) and metrics to minimize with (↓). Best results are highlighted in bold and second best in italics

Retinopathy	Nominal		QWK		CLM		OBD	
ShuffleNetV2	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
CCR (↑)	*0.6597*	0.0360	0.3417	0.0321	**0.6685**	0.0140	0.3700	0.0403
AvAUC (↑)	*0.6381*	0.0074	0.5165	0.0123	0.6373	0.0051	**0.6413**	0.0063
MS (↑)	*0.0043*	0.0063	0.0000	0.0000	0.0000	0.0000	**0.0077**	0.0100
GMS (↑)	*0.0402*	0.0400	0.0000	0.0000	0.0000	0.0000	**0.0615**	0.0614
RMSE (↓)	1.1780	0.0384	**1.0300**	0.0049	1.1441	0.0221	*1.1418*	0.0305
r_s (↑)	0.1559	0.0150	0.1298	0.0099	*0.1823*	0.0076	**0.1942**	0.0105
ˇ (↑)	0.1642	0.0223	0.1089	0.0098	**0.2003**	0.0123	*0.1911*	0.0175

Table 8 Mean results using the VGG11 architecture for the Adience dataset. Metrics to maximize are marked with (↑) and metrics to minimize with (↓). Best results are highlighted in bold and second best in italics

Adience	Nominal		QWK		CLM		OBD	
VGG11	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
CCR (↑)	*0.6738*	0.0151	0.1179	0.0790	0.6403	0.0294	**0.7058**	0.0127
AvAUC (↑)	*0.9305*	0.0056	0.5070	0.0107	**0.9319**	0.0058	0.9298	0.0034
MS (↑)	*0.2734*	0.0769	0.0000	0.0000	0.0000	0.0000	**0.4223**	0.0645
GMS (↑)	*0.5676*	0.0314	0.0000	0.0000	0.0000	0.0000	**0.6401**	0.0163
RMSE (↓)	0.9802	0.0377	3.1913	0.6807	*0.8322*	0.0361	**0.8115**	0.0291
r_s (↑)	0.8629	0.0082	0.0060	0.0344	*0.8991*	0.0072	**0.9023**	0.0059
ˇ (↑)	0.8726	0.0098	0.0027	0.0148	*0.9107*	0.0068	**0.9121**	0.0063

Table 9 Mean results using the ResNet18 architecture for the Adience dataset. Metrics to maximize are marked with (↑) and metrics to minimize with (↓). Best results are highlighted in bold and second best in italics

Adience	Nominal		QWK		CLM		OBD	
ResNet18	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
CCR (↑)	*0.5643*	0.0598	0.5194	0.0273	0.5629	0.0201	**0.6177**	0.0332
AvAUC (↑)	0.8844	0.0215	0.7595	0.0209	*0.8889*	0.0080	**0.9008**	0.0125
MS (↑)	*0.1652*	0.1314	0.0000	0.0000	0.0000	0.0000	**0.2533**	0.0842
GMS (↑)	*0.4181*	0.1382	0.0000	0.0000	0.0000	0.0000	**0.5057**	0.0705
RMSE (↓)	1.3026	0.1469	1.0576	0.0407	*1.0161*	0.0328	**0.9909**	0.0475
r_s (↑)	0.7702	0.0404	0.8449	0.0139	*0.8488*	0.0075	**0.8549**	0.0137
ˇ (↑)	0.7746	0.0481	0.8486	0.0101	*0.8589*	0.0066	**0.8631**	0.0152

Table 10 Mean results using the MobileNetV3 architecture for the Adience dataset. Metrics to maximize are marked with (↑) and metrics to minimize with (↓). Best results are highlighted in bold and second best in italics

Adience MobileNetV3	Nominal		QWK		CLM		OBD	
	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
CCR (↑)	0.4131	0.0161	0.3704	0.0253	**0.4320**	0.0221	*0.4284*	0.0210
AvAUC (↑)	0.7660	0.0110	0.7213	0.0180	*0.7732*	0.0159	**0.7823**	0.0115
MS (↑)	*0.0054*	0.0083	0.0000	0.0000	0.0000	0.0000	**0.0797**	0.0450
GMS (↑)	*0.0885*	0.1048	0.0000	0.0000	0.0000	0.0000	**0.2679**	0.0728
RMSE (↓)	1.6606	0.0577	*1.4196*	0.0615	1.4550	0.0780	**1.3816**	0.0323
r_s (↑)	0.5791	0.0289	0.6951	0.0240	*0.7001*	0.0256	**0.7002**	0.0156
ˇ (↑)	0.5795	0.0281	**0.7193**	0.0233	0.7104	0.0234	*0.7154*	0.0166

Table 11 Mean results using the ShuffleNetV2 architecture for the Adience dataset. Metrics to maximize are marked with (↑) and metrics to minimize with (↓). Best results are highlighted in bold and second best in italics

Adience ShuffleNetV2	Nominal		QWK		CLM		OBD	
	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
CCR (↑)	0.4419	0.0089	0.4210	0.0301	*0.4706*	0.0220	**0.4774**	0.0306
AvAUC (↑)	0.7817	0.0104	0.7019	0.0201	*0.8069*	0.0160	**0.8121**	0.0165
MS (↑)	*0.0002*	0.0011	0.0000	0.0000	0.0000	0.0000	**0.1059**	0.0493
GMS (↑)	*0.0084*	0.0458	0.0000	0.0000	0.0000	0.0000	**0.3246**	0.0738
RMSE (↓)	1.6205	0.0523	**1.2808**	0.0481	*1.2877*	0.0743	1.3108	0.0512
r_s (↑)	0.6131	0.0148	*0.7553*	0.0195	**0.7557**	0.0235	0.7345	0.0224
ˇ (↑)	0.6139	0.0198	**0.7734**	0.0162	*0.7642*	0.0214	0.7490	0.0246

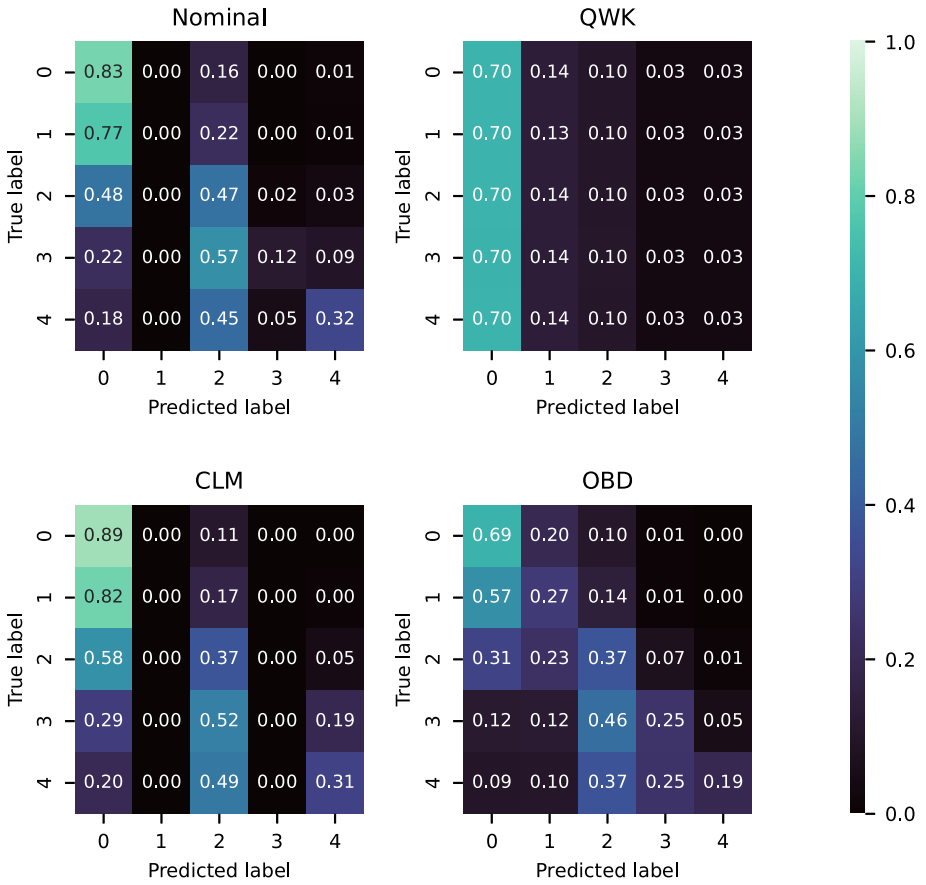

Fig. 10 Average confusion matrices for each methodology using the VGG11 architecture on the Retinopathy dataset. Rows are normalised according to the total number of samples in the test set for each class

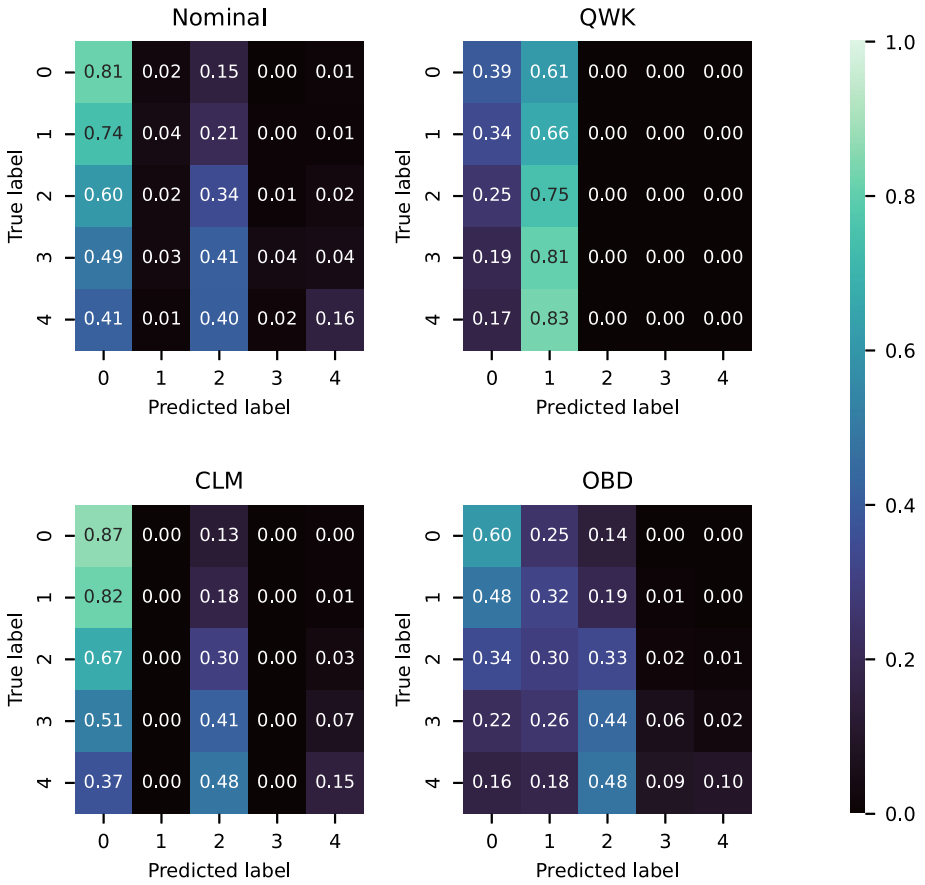

Fig. 11 Average confusion matrices for each methodology using the ResNet18 architecture on the Retinopathy dataset. Rows are normalised according to the total number of samples in the test set for each class

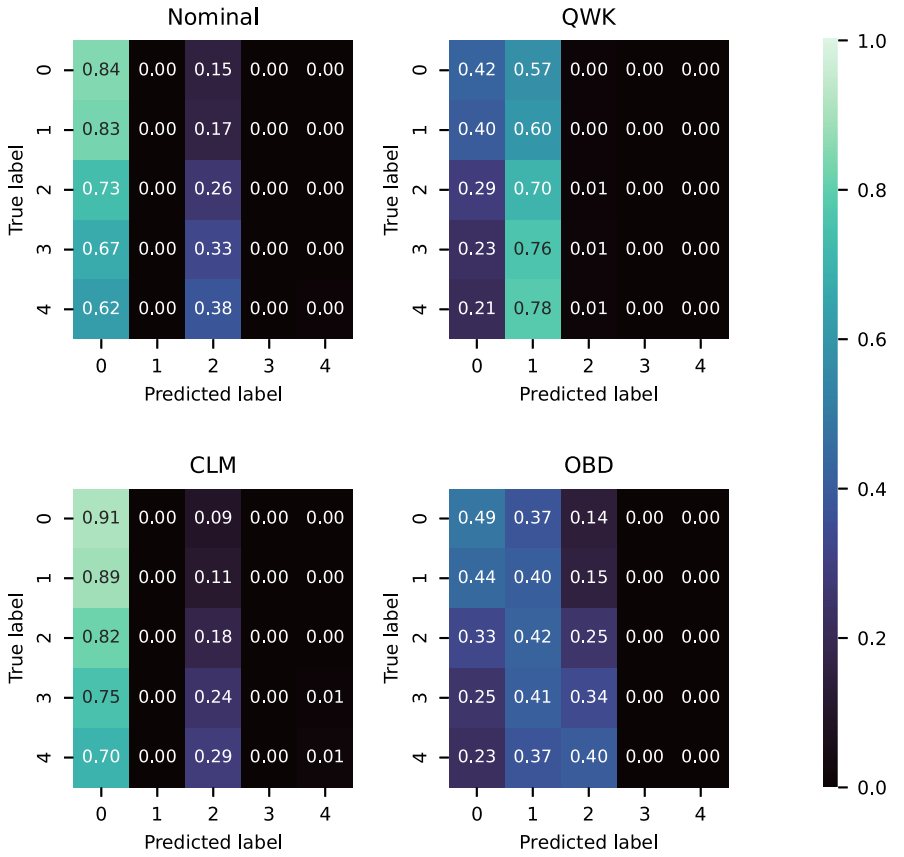

Fig. 12 Average confusion matrices for each methodology using the MobileNetV3 architecture on the Retinopathy dataset. Rows are normalised according to the total number of samples in the test set for each class

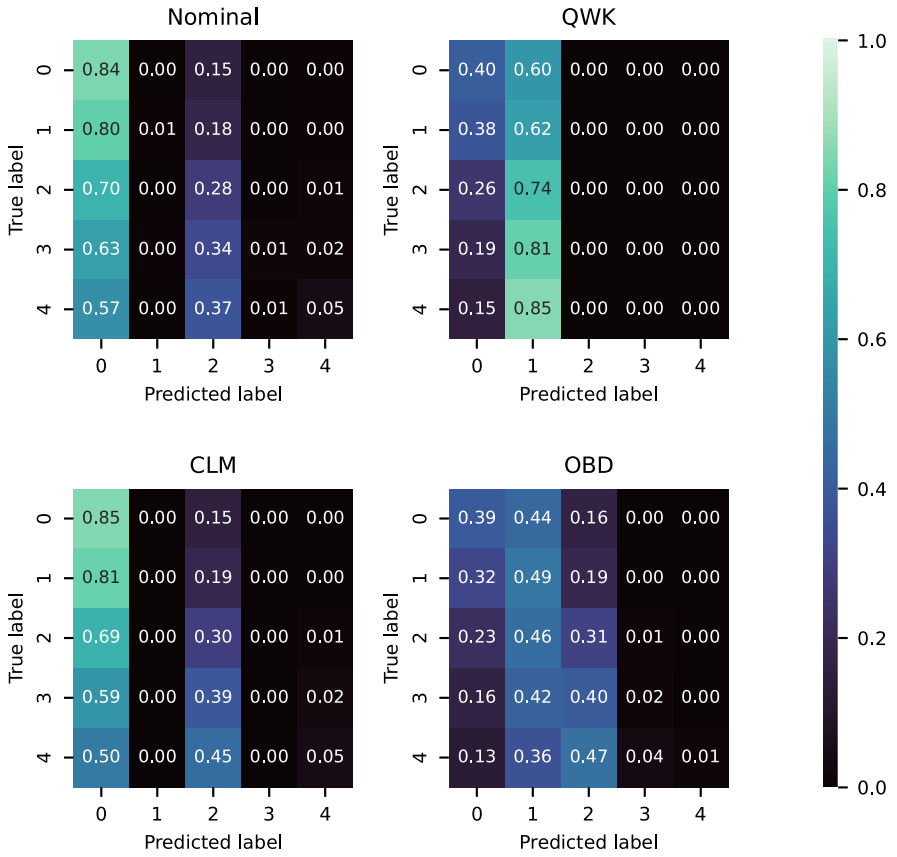

Fig. 13 Average confusion matrices for each methodology using the ShuffleNetV2 architecture on the Retinopathy dataset. Rows are normalised according to the total number of samples in the test set for each class

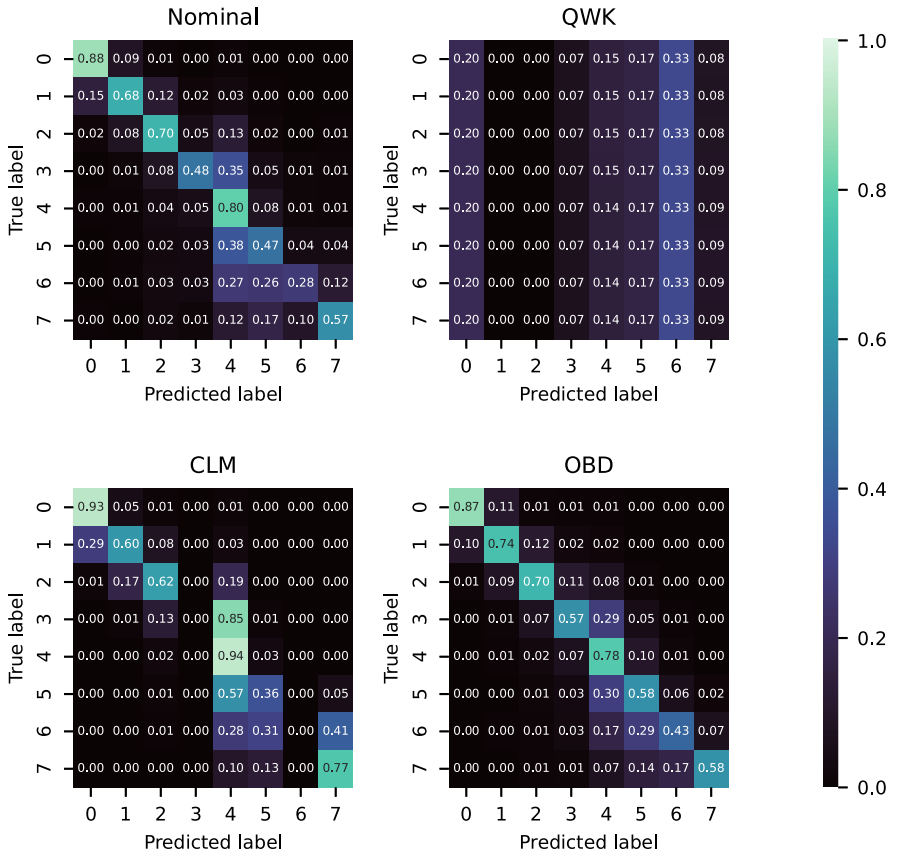

Fig. 14 Average confusion matrices for each methodology using the VGG11 architecture on the Adience dataset. Rows are normalised according to the total number of samples in the test set for each class

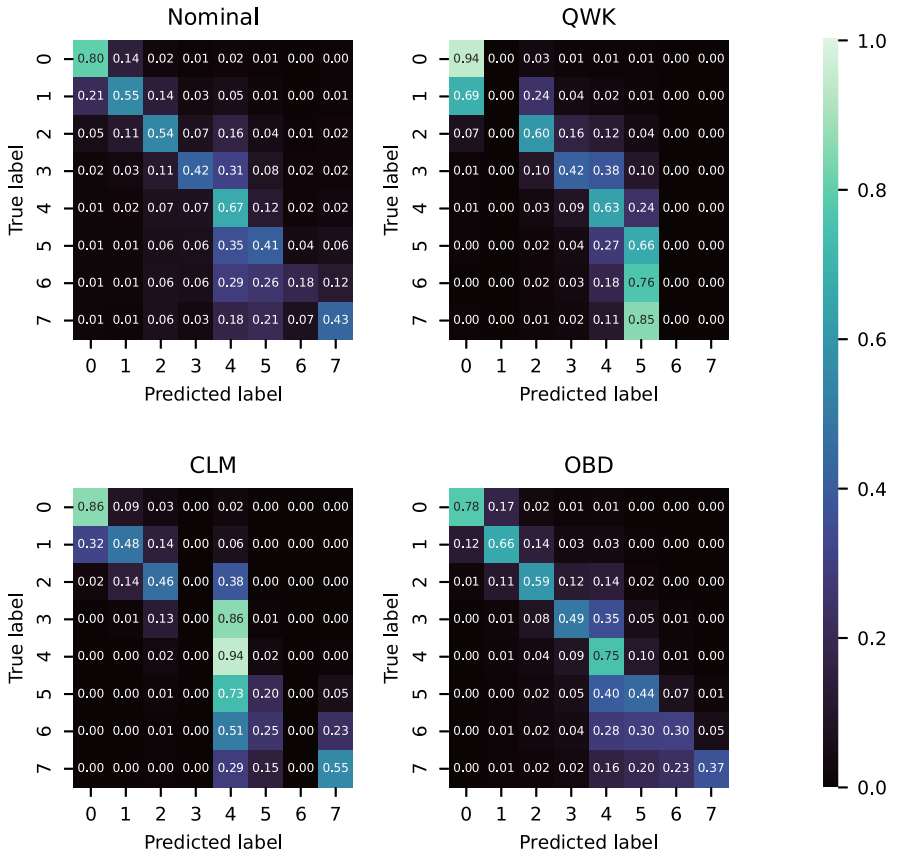

Fig. 15 Average confusion matrices for each methodology using the ResNet18 architecture on the Adience dataset. Rows are normalised according to the total number of samples in the test set for each class

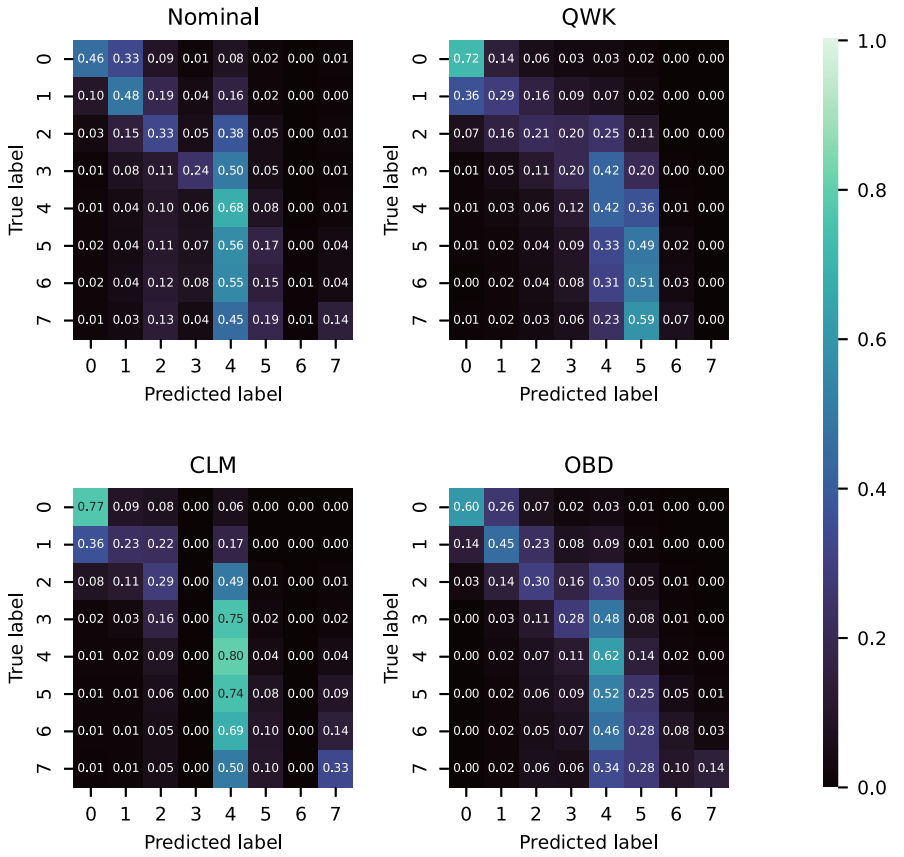

Fig. 16 Average confusion matrices for each methodology using the MobileNetV3 architecture on the Adience dataset. Rows are normalised according to the total number of samples in the test set for each class

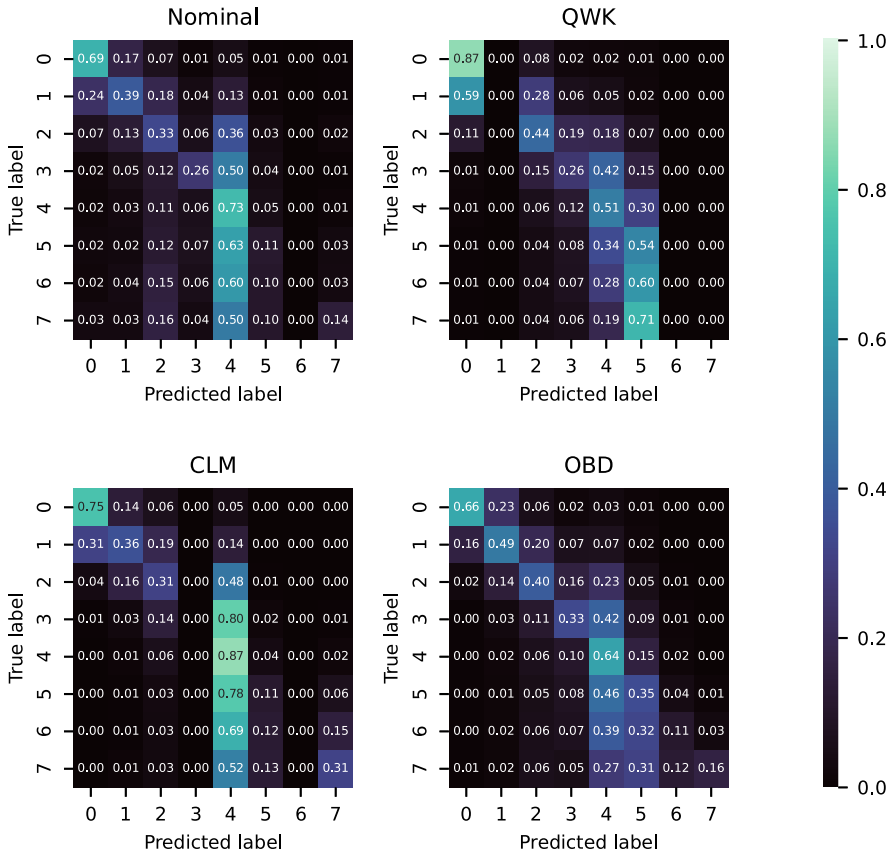

Fig. 17 Average confusion matrices for each methodology using the ShuffleNetV2 architecture on the Adience dataset. Rows are normalised according to the total number of samples in the test set for each class

Appendix B Statistical Analysis Tables

See Tables 12, 13, 14, 15, 16

Table 12 ANOVA III table for the *CCR* test results

Source	Sum of Squares	Degrees of freedom	F-ratio	Sig. level
Intercept	263.617	1	64535.256	<0.001
Dataset (D)	1.043	1	255.386	<0.001
Architecture (A)	2.232	3	182.175	<0.001
Methodology (L)	8.295	3	676.924	<0.001
D.A Interaction	0.688	3	56.110	<0.001
D.L Interaction	2.279	3	185.996	<0.001
A.L Interaction	1.614	9	43.913	<0.001
D.A.L Interaction	3.478	9	94.614	<0.001
Error	3.791	928		
Total	287.038	960		

Table 13 ANOVA III table for the *AvAUC* test results

Source	Sum of Squares	Degrees of freedom	F-ratio	Sig. level
Intercept	498.365	1	3308978.403	<0.001
Dataset (D)	6.836	1	45386.169	<0.001
Architecture (A)	1.295	3	2865.268	<0.001
Methodology (L)	5.084	3	11252.321	<0.001
D.A Interaction	0.204	3	451.423	<0.001
D.L Interaction	0.022	3	49.183	<0.001
A.L Interaction	1.802	9	1329.763	<0.001
D.A.L Interaction	0.393	9	289.951	<0.001
Error	0.140	928		
Total	514.141	960		

Table 14 ANOVA III table for the *RMSE* test results

Source	Sum of Squares	Degrees of freedom	F-ratio	Sig. level
Intercept	1444.893	1	55143.615	<0.001
Dataset (D)	14.551	1	555.317	<0.001
Architecture (A)	6.064	3	77.146	<0.001
Methodology (L)	15.774	3	200.663	<0.001
D.A Interaction	5.435	3	69.142	<0.001
D.L Interaction	13.720	3	174.541	<0.001
A.L Interaction	66.181	9	280.642	<0.001
D.A.L Interaction	34.190	9	144.982	<0.001
Error	24.316	928		
Total	1625.124	960		

Table 15 ANOVA III table for the r_s test results

Source	Sum of Squares	Degrees of freedom	F-ratio	Sig. level
Intercept	203.702	1	626011.422	<0.001
Dataset (D)	64.644	1	198663.249	<0.001
Architecture (A)	1.975	3	2022.995	<0.001
Methodology (L)	5.699	3	5837.678	<0.001
D.A Interaction	1.703	3	1744.819	<0.001
D.L Interaction	0.306	3	313.698	<0.001
A.L Interaction	12.911	9	4408.478	<0.001
D.A.L Interaction	3.217	9	1098.343	<0.001
Error	0.302	928		
Total	294.458	960		

Table 16 ANOVA III table for the κ test results

Source	Sum of Squares	Degrees of freedom	F-ratio	Sig. level
Intercept	214.856	1	567385.301	<0.001
Dataset (D)	63.004	1	166379.208	<0.001
Architecture (A)	2.482	3	2184.378	<0.001
Methodology (L)	7.131	3	6277.503	<0.001
D.A Interaction	2.445	3	2152.196	<0.001
D.L Interaction	0.207	3	182.132	<0.001
A.L Interaction	14.140	9	4149.026	<0.001
D.A.L Interaction	3.169	9	929.813	<0.001
Error	0.351	928		
Total	307.786	960		

References

1. Agresti A (2010) Analysis of ordinal categorical data. Wiley Series in Probability and Statistics, Wiley, Hoboken, NJ
2. Allwein EL, Schapire RE, Singer Y (2000) Reducing multiclass to binary: a unifying approach for margin classifiers. J Mach Learn Res 1:113–141
3. Ben-David A (2008) Comparison of classification accuracy using Cohen's Weighted Kappa. Expert Syst Appl 34(2):825–832. https://doi.org/10.1016/j.eswa.2006.10.022
4. Cardoso JS, Pinto da Costa JF (2007) Learning to classify ordinal data: the data replication method. J Mach Learn Res 8:1393–1429
5. Cardoso JS, Sousa R (2011) Measuring the performance of ordinal classification. Int J Pattern Recognit Artif Intell 25(08):1173–1195. https://doi.org/10.1142/S0218001411009093
6. Chen S, Zhang C, Dong M, et al (2017) Using ranking-CNN for age estimation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 5183–5192
7. Chu W, Keerthi SS (2007) Support vector ordinal regression. Neural Comput 19(3):792–815. https://doi.org/10.1162/neco.2007.19.3.792
8. de la Torre J, Puig D, Valls A (2018) Weighted kappa loss function for multi-class classification of ordinal data in deep learning. Pattern Recognit Lett 105:144–154. https://doi.org/10.1016/j.patrec.2017.05.018
9. Deng WY, Zheng QH, Lian S et al (2010) Ordinal extreme learning machine. Neurocomputing 74(1):447–456. https://doi.org/10.1016/j.neucom.2010.08.022

10. Dorado-Moreno M, Pérez-Ortiz M, Gutiérrez PA et al (2017) Dynamically weighted evolutionary ordinal neural network for solving an imbalanced liver transplantation problem. Artif Intell Med 77:1–11. https://doi.org/10.1016/j.artmed.2017.02.004
11. Eidinger E, Enbar R, Hassner T (2014) Age and gender estimation of unfiltered faces. IEEE Trans Inform Forensics Secur 9(12):2170–2179. https://doi.org/10.1109/TIFS.2014.2359646
12. Fernández-Navarro F (2017) A generalized logistic link function for cumulative link models in ordinal regression. Neural Process Lett 46(1):251–269. https://doi.org/10.1007/s11063-017-9589-3
13. Fisher RA (1925) Theory of statistical estimation. Math Proc Camb Philos Soc 22(5):700–725. https://doi.org/10.1017/S0305004100009580
14. Fisher RA (1954) Statistical methods for research workers, twentieth. Oliver and Boyd, Edinburgh
15. Frank E, Hall M (2001) A simple approach to ordinal classification. In: European Conference on Machine Learning. Springer, Berlin, Heidelberg, Freiburg, Germany, 145–156, https://doi.org/10.1007/3-540-44795-4_13
16. Gutiérrez PA, Pérez-Ortiz M, Sánchez-Monedero J et al (2016) Ordinal regression methods: survey and experimental study. IEEE Trans Knowl Data Eng 28(1):127–146. https://doi.org/10.1109/TKDE.2015.2457911
17. He K, Zhang X, Ren S, et al (2015a) Deep residual learning for image recognition. arXiv:1512.03385
18. He K, Zhang X, Ren S, et al (2015b) Delving deep into rectifiers: surpassing human-level performance on ImageNet classification. arXiv:1502.01852
19. Howard A, Sandler M, Chu G, et al (2019) Searching for MobileNetV3. arXiv:1905.02244
20. Jianlin Cheng, Zheng Wang, Pollastri G (2008) A neural network approach to ordinal regression. In: IEEE international joint conference on neural networks (IEEE world congress on computational intelligence), 1279–1284, https://doi.org/10.1109/IJCNN.2008.4633963
21. Kingma DP, Ba J (2017) Adam: a method for stochastic optimization. arXiv:1412.6980
22. Kotsiantis SB, Pintelas PE (2004) A cost sensitive technique for ordinal classification problems. In: Vouros GA, Panayiotopoulos T (eds.) Methods and applications of artificial intelligence, lecture notes in computer science, https://doi.org/10.1007/978-3-540-24674-9_24
23. Kramer S, Widmer G, Pfahringer B, et al (2010) Prediction of ordinal classes using regression trees. In: Raś ZW, Ohsuga S (eds) Foundations of intelligent systems. Springer, Berlin, Heidelberg, Lecture notes in computer science, 426–434, https://doi.org/10.1007/3-540-39963-1_45
24. Lausser L, Schäfer LM, Kühlwein SD et al (2020) Detecting ordinal subcascades. Neural Process Lett 52(3):2583–2605. https://doi.org/10.1007/s11063-020-10362-0
25. Liu Y, Kong A, Goh C (2017) Deep ordinal regression based on data relationship for small datasets. In: IJCAI international joint conference on artificial intelligence, https://doi.org/10.24963/ijcai.2017/330
26. Ma N, Zhang X, Zheng HT, et al (2018) ShuffleNet V2: Practical guidelines for efficient cnn architecture design. arXiv:1807.11164
27. McCullagh P (1980) Regression models for ordinal data. J Royal Stat Soc: Series B (Methodol) 42(2):109–127. https://doi.org/10.1111/j.2517-6161.1980.tb01109.x
28. Niu Z, Zhou M, Wang L, et al (2016) Ordinal regression with multiple output CNN for age estimation. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR), 4920–4928, https://doi.org/10.1109/CVPR.2016.532
29. Pérez-Ortiz M, Fernández-Delgado M, Cernadas E et al (2016) On the use of nominal and ordinal classifiers for the discrimination of states of development in fish oocytes. Neural Process Lett 44(2):555–570. https://doi.org/10.1007/s11063-015-9476-8
30. Sánchez-Monedero J, Pérez-Ortiz M, Sáez A et al (2018) Partial order label decomposition approaches for melanoma diagnosis. Appl Soft Comput 64:341–355. https://doi.org/10.1016/j.asoc.2017.11.042
31. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556
32. Tukey JW (1949) Comparing individual means in the analysis of variance. Biometrics 5(2):99–114. https://doi.org/10.2307/3001913
33. Vargas VM, Gutiérrez PA, Hervás-Martínez C (2020) Cumulative link models for deep ordinal classification. Neurocomputing 401:48–58. https://doi.org/10.1016/j.neucom.2020.03.034
34. Veit A, Wilber M, Belongie S (2016) Residual networks behave like ensembles of relatively shallow networks. arXiv:1605.06431
35. Williams R (2006) Generalized ordered logit/partial proportional odds models for ordinal dependent variables. Stata J 6:58–82. https://doi.org/10.1177/1536867X0600600104
36. Wu H, Lu H, Ma S (2003) A practical SVM-based algorithm for ordinal regression in image retrieval. In: 11th ACM international conference on multimedia. Association for computing machinery, Berkeley, 612–621, https://doi.org/10.1145/957013.957144

37. Zheng Q, Yang M, Yang J et al (2018) Improvement of generalization ability of deep CNN via implicit regularization in two-stage training process. IEEE Access 6:15844–15869. https://doi.org/10.1109/ACCESS.2018.2810849
38. Zheng Q, Tian X, Yang M et al (2020) PAC-Bayesian framework based drop-path method for 2D discriminative convolutional network pruning. Multidimens Syst Signal Process 31(3):793–827. https://doi.org/10.1007/s11045-019-00686-z