Neural Computing and Applications (2025) 37:3505–3520
https://doi.org/10.1007/s00521-024-10721-1

ORIGINAL ARTICLE

Quantification using permutation-invariant networks based on histograms

Olaya Pérez-Mon[1] · Alejandro Moreo[2] · Juan José del Coz[1] · Pablo González[1]

Received: 27 March 2024 / Accepted: 14 October 2024 / Published online: 14 December 2024

Abstract
Quantification, also known as class prevalence estimation, is the supervised learning task in which a model is trained to predict the prevalence of each class in a given bag of examples. This paper investigates the application of deep neural networks for tasks of quantification in scenarios where it is possible to apply a symmetric supervised approach that eliminates the need for classification as an intermediate step, thus directly addressing the quantification problem. Additionally, it discusses existing permutation-invariant layers designed for set processing and assesses their suitability for quantification. Based on our analysis, we propose HistNetQ, a novel neural architecture that relies on a permutation-invariant representation based on histograms that is especially suited for quantification problems. Our experiments carried out in two standard competitions, which have become a reference in the quantification field, show that HistNetQ outperforms other deep neural network architectures designed for set processing, as well as the current state-of-the-art quantification methods. Furthermore, HistNetQ offers two significant advantages over traditional quantification methods: i) it does not require the labels of the training examples but only the prevalence values of a collection of training bags, making it applicable to new scenarios; and ii) it is able to optimize any custom quantification-oriented loss function.

Keywords Quantification · Prevalence estimation · Deep learning · Deep neural networks

1 Introduction

In many real-world applications [1–6], predicting the class of each individual example in a dataset is of little concern, since the real interest lies in the *aggregate* level, i.e., in estimating the prevalence of the classes in a bag of examples. Quantification, also known as "class prevalence estimation," is the supervised learning task that tackles this particular problem [7]. Quantification has already proven useful in a wide variety of fields, providing answers to questions such as: *what is the percentage of positive, neutral, and negative reviews for a specific product of a given company?* [5] or *what is the percentage of plankton organisms belonging to each of the phytoplankton species in this water sample?* [3].

This learning problem can be formalized as follows. Let $\mathcal{Y} = \{c_j\}_{j=1}^{l}$ be the classes of interest, the goal is to learn a quantifier: $q : \mathbb{N}^{\mathcal{X}} \rightarrow \Delta^{l-1}$, i.e., a functional $q \in \mathcal{Q}$ that, given a test bag $B = \{\mathbf{x}_i\}_{i=1}^{m}$ in which $\mathbf{x}_i \in \mathcal{X}$ is a vector of features representing a data example, returns a vector of class prevalence estimations $q(B) \in \Delta^{l-1}$. In this context, $\mathbb{N}^{\mathcal{X}}$ denotes the space of bags (aka "multisets") over the input space $\mathcal{X}$. The function q thus maps any bag B into the probability simplex $\Delta^{l-1} = \{(p_1, \ldots, p_l) \mid p_j \in [0, 1], \sum_{j=1}^{l} p_j = 1\}$, i.e., into the domain of all vectors representing probability distributions over $\mathcal{Y}$. We will use $\mathbf{p}_B \in \Delta^{l-1}$ to indicate the true prevalence values of a bag B, and

Alejandro Moreo, Juan José del Coz and Pablo González have contributed equally to this work.

✉ Olaya Pérez-Mon
perezolaya@uniovi.es

Alejandro Moreo
alejandro.moreo@isti.cnr.it

Juan José del Coz
juanjo@uniovi.es

Pablo González
gonzalezgpablo@uniovi.es

[1] Artificial Intelligence Center, University of Oviedo, C/ Blasco de Garay, Gijón 33204, Asturias, Spain

[2] Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa 56124, Italy

$\hat{\mathbf{p}}_B^A \in \Delta^{l-1}$ to indicate the estimated prevalence values predicted by the quantification algorithm A, so that $p_B(c_j)$ and $\hat{p}_B^A(c_j)$ are the true and the predicted class prevalence, for class c_j, respectively.

At first glance, quantification seems a task very similar to classification in spirit. Indeed, the most straightforward solution to the quantification problem, referred to as Classify & Count (CC) in the literature, comes down to first learning a hard classifier $h : \mathcal{X} \rightarrow \mathcal{Y}$ using a training dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ drawn from $\mathcal{X} \times \mathcal{Y}$, to then issue label predictions for all examples in the test bag B, and finally counting the number of times each class has been attributed. However, it has been observed that CC gives rise to biased estimators of class prevalence [8]. The reason is that h is biased toward the training prevalence and therefore tends to underestimate (resp. overestimate) the true prevalence of a class when this class becomes more prevalent (resp. less prevalent) in the test bag B than it was in the training set D. Noticeably, most quantification algorithms rely on the predictions of a classifier[1] which are subsequently post-processed using information from D and B. This post-processing is necessary since, in quantification, we assume to face a shift in the data distribution (i.e., that the prevalence of the classes may differ between D and B).

This particular shift is generally known as "label shift" or "prior probability shift" [11], according to which the prior distribution $P(Y)$ can change between training and deployment conditions, while the class-conditional densities $P(X|Y)$ are assumed stationary. The fact that CC is not suitable for quantification under prior probability shift conditions has led to the development of a myriad of methods designed specifically for quantification, which is by now recognized as a task in its own right (see, e.g., [12, 13] for an overview).

One of the main advantages of adopting deep neural network architectures (DNNs) for quantification is that DNNs allow the learning process to handle bags of examples (labeled by their class prevalence values) instead of individual examples (labeled by class). Following this intuition, a change in the learning paradigm with respect to the traditional one was first proposed in [14]. In this paper, we offer an in-depth exploration of the implications of this change of paradigm, by analyzing the main advantages and limitations with respect to traditional approaches to quantification. Conversely, traditional quantification methods adopt an *asymmetric* approach in which a classifier is trained to infer the class of the individual examples and in which the label predictions are used to estimate the prevalence of the classes in the bag. This way, the training labels (class labels attached to the example) and the labels to be predicted (class prevalence values attached to the bag) are not *homologous*. In contrast, following the approach proposed in [14], we can reframe the quantification problem as a *symmetric* supervised learning task in which the training set consists of a collection of bags with examples labeled at the aggregate level (i.e., without individual class labels). This formulation posits the quantification problem as a multivariate regression task, in which the labels provided for training and the labels we need to predict become homologous. Throughout this paper, we will demonstrate further advantages of this formulation. Among them, and in contrast to traditional quantification methods, the quantifier becomes capable of optimizing any specific loss function.

With this aim, our paper investigates the application of DNNs to the symmetric quantification problem. The paper begins by addressing a central issue that arises when making predictions for entire bags rather than for individual examples, namely how to represent bags in a permutation-invariant manner [15–17]. Two influential DNN architectures have been proposed for set processing: DeepSets [18] and SetTransformers [19]. The former employs a pooling layer like max, average, or median, to summarize each bag, while the latter uses a transformer architecture without positional encoding. These approaches were designed as universal approximation functions for set-based problems. Here, we propose a new architecture, called HistNetQ, that gains inspiration from histogram representations of empirical distributions.[2] The reason why histograms seem promising is twofold: histograms are naturally geared toward representing densities and convey more information than plain statistics (like the mean, or median). We will show that histogram-based layers can be seen as a generalization of the pooling layers proposed in [14, 18].

The contributions of this paper are threefold. First, we analyze the symmetric approach of [14] for quantification, discussing its strengths and limitations. Secondly, we empirically assess the suitability of previously proposed permutation-invariant layers to the quantification problem. Finally, we propose HistNetQ, a new permutation-invariant architecture based on differentiable histograms, specifically useful for quantification tasks.

Our experiments reveal two main findings: i) HistNetQ outperforms not only traditional quantification methods and previous general-purpose DNN architectures for set

[1] Other alternatives exist which instead rely directly on the features of the examples (the covariates) [9, 10]; however, the literature has shown that these approaches tend to be less competitive.

[2] The term "histogram" is typically used to refer to a visualization tool for representing empirical distributions. Although there is no visual representation within the model, we still use this term because our method is based on certain operators that, in previous works, have come to be known as "differentiable histograms".

processing but also state-of-the-art quantification-specific DNN methods [14, 20] in the two editions of the LeQua competition [21, 22], the only competitions entirely devoted to quantification held to date, ii) HistNetQ proves competitive not only in the symmetric approach, but also under the asymmetric approach too, that is, when a set of bags labeled by prevalence is not available and must be generated from D via sampling.

2 Related work

This section briefly describes the most important quantification methods based on the asymmetric approach as well as DNN architectures specifically designed to handle set-based data.

2.1 Quantification methods

The Adjusted Classify and Count (ACC) method (see [8, 23]), later renamed as Black Box Shift Estimation "hard" (BBSE-hard) in [24], learns a classifier h and then applies a correction relying on the law of total probability:

$$p(h(\mathbf{x}) = c_i) = \sum_{c_j \in \mathcal{Y}} p(h(\mathbf{x}) = c_i | c_j) \cdot p(c_j), \tag{1}$$

which corresponds to the following linear system:

$$\hat{\mathbf{p}}_B^{\mathrm{CC}} = \mathbf{M}_h \cdot \mathbf{p}, \tag{2}$$

where $\hat{\mathbf{p}}_B^{\mathrm{CC}}$ are the prevalence estimates returned by the CC method for the test bag B, and $\mathbf{M}_h$ is the misclassification matrix characterizing h, in which the entry m_{ij} is the probability that h predicts c_i if the true class is c_j. Note that $\mathbf{M}_h$ is unknown but can be estimated via cross-validation. The reason is that the probability shift assumption by which the class-conditional distribution of the covariates $P(X|Y)$ is assumed stationary across training and test conditions, and implies that the class-conditional distribution of the predictions $P(h(X)|Y)$ remains stationary too [25]. ACC comes down to solving (2) as $\hat{\mathbf{p}}_B^{\mathrm{ACC}} = \hat{\mathbf{M}}_h^{-1} \cdot \hat{\mathbf{p}}_B^{\mathrm{CC}}$ if $\mathbf{M}_h$ is invertible; otherwise, the Penrose pseudoinverse can be used [26].

In [27], the authors propose two probabilistic variants of CC and ACC, that consist of replacing the hard classifier h with a soft classifier $s : \mathcal{X} \to \Delta^{l-1}$, thus giving rise to Probabilistic Classify & Count (PCC):

$$\hat{\mathbf{p}}_B^{\mathrm{PCC}} = \frac{\sum_{\mathbf{x} \in \mathbf{B}} s(\mathbf{x})}{|B|}, \tag{3}$$

and Probabilistic Adjusted Classify and Count (PACC) (also known as BBSE-soft in [24]):

$$\hat{\mathbf{p}}_B^{\mathrm{PACC}} = \hat{\mathbf{M}}_s^{-1} \cdot \hat{\mathbf{p}}_B^{\mathrm{PCC}}. \tag{4}$$

The Expectation Maximization for Quantification (EMQ) [28] method applies the EM algorithm to adjust the posterior probabilities generated by a soft classifier s to the potential shift in the label distribution. It iterates between two steps: expectation, where the posteriors are updated, and maximization, where the priors are updated, continuing until convergence. The literature has convincingly shown that EMQ is a "hard to beat" quantification method [29, 30]. However, the performance of EMQ heavily relies on the quality of the posterior probabilities generated by s (i.e., on the fact that these posterior probabilities are well-calibrated). For this reason, different calibration strategies have been proposed in the literature; among these, the Bias-Corrected Temperature Scaling (BCTS) calibration proved the best of the lot [29]. In the experiments of Sect. 5, we will consider two variants of EMQ: one in which the posterior probabilities are not recalibrated and another in which BCTS calibration is applied.

The HDy method [9] uses a combination of histograms to represent the distributions of the training data D and the test bag B, using the Hellinger distance (HD) to compare them. HDy builds the histograms using the posterior probabilities returned by a soft classifier s. Figure 1 illustrates the inner workings of the HDy method in a binary quantification problem. In the training phase, the distributions of the posteriors returned by s for the positive and negative examples in the training set D are estimated using histograms D^+ and D^-, respectively. At test time, the posteriors of the test bag B are computed and represented using the same procedure. HDy will then return the prevalence value $\hat{p}$ that minimizes the HD between the mixture and the test bag distributions, solving the following optimization problem:

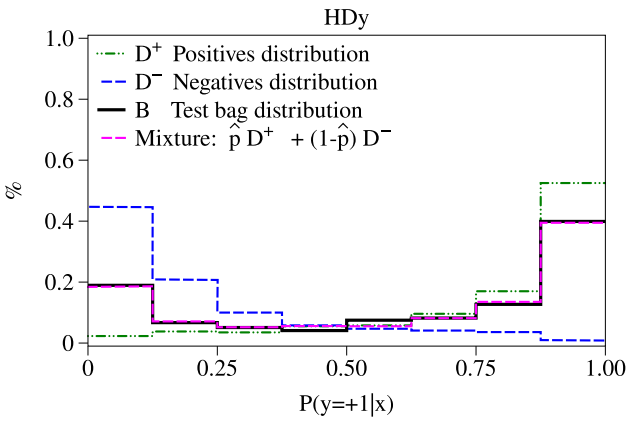

Fig. 1 In this example, we observe the distributions of positive cases (green) and negative cases (blue) within the training dataset D. Additionally, we can see the mixture distribution (magenta) that provides the best approximation of the test bag distribution (black)

$$\arg\min_{\hat{p}\in[0,1]} HD(\hat{p}\cdot D^{+}+(1-\hat{p})\cdot D^{-}\ \ B\ ). \tag{5}$$

QuaNet is a DNN architecture for binary quantification [20]. QuaNet sorts the inputs by their posterior probabilities and processes the sequence using a bidirectional LSTM that learns a predictor of class prevalence. The prevalence estimation is then combined with the estimates computed with some base quantification methods (CC, ACC, PCC, PACC, and EMQ). QuaNet then generates many bags out of the training data D to train the model. However, in contrast to the rest of the DNN architectures that this paper analyzes, QuaNet follows the asymmetric approach and requires (just like all quantification methods discussed in this section) the availability of a training set D with individual example labels.

A more exhaustive description of these (and other) quantification algorithms can be found in [12, 13].

2.2 DNN architectures for sets

In recent years, dedicated DNNs have been proposed to handle set-based data. Even though these architectures were not originally devised with class prevalence estimation in mind, they seem apt for the task since they all construct on top of permutation-invariant representations. Quantification requires permutation-invariant layers, because the prevalences of B do not change if the examples in B are shuffled.

The first of these architectures is called DeepSets [18]. DeepSets relies on different permutation-invariant pooling operators, like max, average, or median. Pooling operators are applied to the features representing the examples in a given bag B. An operator is said to be *permutation-invariant* when the output of the layer is not affected by the order in which the examples appear in the (serialized) input sequence S. More formally, a function f is permutation-invariant if $f(S) = f(\pi(S))$ for any permutation function π. In [14], the authors use the same architecture and pooling layers as in DeepSets, proposing its application to quantification problems. For the sake of clarity, we will refer to the use of simple pooling layers, as max, average, or median, as DeepSets.

In [19], one step forward was taken by replacing the simple pooling operators of DeepSets with transformers, i.e., with attention-based mechanisms that model complex interactions between the elements in the set. In this architecture, called SetTransformers, positional encoding is not included since the order of the examples in the bag is unimportant. Instead of modeling the interactions between every possible pair of examples, SetTransformers incorporates the concept of *inducing points*, learnable latent data points of the vector space that is given as input to the self-attention mechanism. In this way, the original $\mathcal{O}(n^2)$ complexity of SetTransfomer is reduced to $\mathcal{O}(nI)$, where n is the bag size and I (with $I \ll n$) the number of inducing points. To the best of our knowledge, SetTransformers have never been used in quantification.

3 Symmetric quantification: a case study analysis

While [14] pioneered the symmetric approach to the field of quantification learning, the authors did not delve deeper into the implications of the new approach. Among other things, this section aims to fill this gap by providing a comprehensive analysis of its main advantages and limitations.

Most previous quantification algorithms (as for example those described in Sect. 2.1) require a training dataset D, in which labels are attached to individual examples, in order to learn a quantifier $q \in \mathcal{Q}$, $q : \mathbb{N}^{\mathcal{X}} \to \Delta^{l-1}$ that, given a test bag B, computes estimates of class prevalence. Therefore, the learning device is of the form $L : (\mathcal{X} \times \mathcal{Y})^n \to \mathcal{Q}$, meaning that the quantification problem is posed as an asymmetric task: training labels are defined in $\mathcal{Y}$, while predictions are probability distributions from Δ^{l-1}. In order to reformulate quantification as a symmetric supervised task, the training set needs to be defined as $D' = \{(B_i, \mathbf{p}_i)\}_{i=1}^{n'}$, with $B_i \in \mathbb{N}^{\mathcal{X}}$ a training bag labeled according to its class prevalence values $\mathbf{p}_i \in \Delta^{l-1}$. The learning device is thus formalized as $L' : (\mathbb{N}^{\mathcal{X}} \times \Delta^{l-1})^{n'} \to \mathcal{Q}$, so that the labels provided for training and the labels we need to predict become

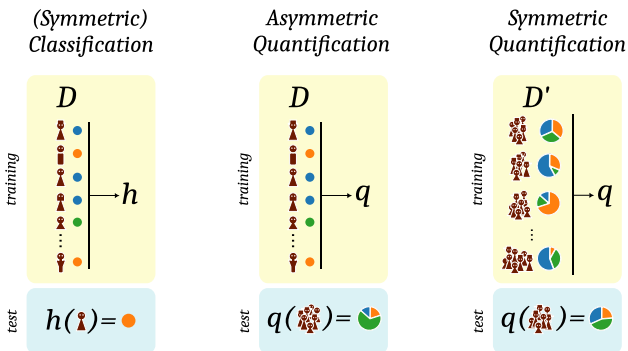

Fig. 2 Different learning paradigms with respect to the type of data used for training and testing. First column: a typical scenario of classification; the training data consist of individually labeled data instances and the test consist of predicting class labels for individual data instances. Middle column: traditional quantification methods are trained using individually labeled data instances; at test time, quantifiers are required to predict the distribution of a bag. Last column: in a symmetric quantification approach the model is trained on bags labeled by prevalence

homologous, i.e., are both probability distributions in Δ^{l-1}. See Fig. 2 for a graphical representation of this intuition.

This reformulation presents some advantages and disadvantages that were not discussed in [14]. The first advantage of the new approach is that the quantification method is no longer necessarily bound to prior probability shift. This is a major implication, since most previously proposed methods in the quantification literature assume to be in the presence of prior probability shift, and are specifically devised to counter it. In contrast, by adopting the symmetric approach, training examples can potentially exhibit any type of shift, to which the method at hand will try to develop resilience as part of the learning procedure. This characteristic is significant, as it considerably broadens the applicability of the quantification method to scenarios beyond prior probability shift.

The second advantage is that the quantification problem is addressed *directly*, and not via classification as an intermediate step. This should be advantageous by virtue of Vapnik's principle, according to which *"If you possess a restricted amount of information for solving some problem, try to solve the problem directly and never solve a more general problem as an intermediate step. It is possible that the available information is sufficient for a direct solution but is insufficient for solving a more general intermediate problem."* Notice that all the methods described in Sect. 2.1 (with the sole exception of QuaNet) do not directly learn a model by minimizing a task-oriented loss (as is rather customary in other areas of supervised machine learning). The reason is that methods like ACC, PACC, EMQ, and HDy undertake an asymmetric training in which a classifier is learned, and then a predefined post-processing function is employed to yield prevalence estimates. As a result, most quantification methods proposed so far are agnostic to specific quantification loss functions.

In contrast, methods based on the symmetric approach (including HistNetQ) can be specifically tailored to minimize a quantification-oriented loss function. This is important as different applications may be characterized by different notions of criticality; well-designed loss functions play a crucial role in accurately reflecting these notions, thereby enabling a method to become accurate in terms of application-dependent requirements. For example (a) one may opt for adopting the absolute error (AE) as an easily interpretable metric in general cases; (b) in applications related to epidemiology, estimating the prevalence of rare diseases might be better served by the relative absolute error (RAE); (c) in different contexts, employing a cost-sensitive error measure could help weigh the relative importance of different classes. See [31] for a broader discussion on evaluation measures for quantification.

The third advantage is a widening of the range of problems to which quantification can be applied. Current quantification algorithms cannot be applied to problems in which labels are provided at the aggregate level (i.e., datasets of "type D'"). Problems in which the supervised training data naturally arise in the form of sets labeled by prevalence are many, and are the object of study of research areas like multi-instance learning [32], and learning from label proportions (LLP) [33, 34]. Examples of these problems include, for instance, post-electoral results by census tract,[3] demographic analysis in which sensible information (e.g., race, gender) is anonymized but provided at the aggregate level, or public records of proportions of diagnosed diseases per ZIP code. Notice that the symmetric approach allows these problems to be tackled directly.

However, the symmetric approach faces at least two important issues. The first one is that the number of available training bags in D' may be limited in some applications. This limitation arises because supervised learning requires a large amount of labeled data. While the previous approach requires labeling individuals (i.e., instances), the symmetric approach requires labeling populations (i.e., bags of instances); the latter is certainly more demanding to obtain. In Sect. 3.1, we present a method aimed at mitigating this issue that consists of generating new synthetic bags from existing ones.

The second aspect concerns the applicability of the symmetric approach to cases in which the only available training set is a traditional one, i.e., a dataset of "type D" with individual class labels. However, note that such a setup poses no real limitation to the symmetric approaches since a dataset of "type D'" can be easily obtained from a dataset of "type D" via sampling. Sect. 3.2 discusses one sampling generation protocol that fulfills this requirement; the protocol is well-known in the quantification literature, although it is more commonly employed for evaluation purposes, i.e., for generating, out of a collection of labeled individuals, many test bags exhibiting different class distributions that are used for testing quantification algorithms. Of course, while feasible in principle, it is yet to be determined whether a symmetric approach trained via a sampling protocol performs comparably in terms of quantification accuracy with respect to a traditional asymmetric approach trained on the original dataset. This aspect will be analyzed in the experiments of Sect. 5.

[3] See, for example, the PUMS (public use microdata sample) of the U.S. Census Bureau https://www.census.gov/data/datasets/2000/dec/microdata.html.

3.1 Bag mixer: data augmentation for quantification

Arguably, the most important issue the symmetric approach has to face concerns the potential limited size of D', which could easily result in overfitting, especially when using DNN methods. The reason why is that training bags are the equivalent counterparts of training examples from a classification problem. This means that even a relatively high number of training bags (e.g., the LeQua datasets we use in the experiments of Sect. 5 comprise 1000 bags each) remains quite low when compared to classification datasets customarily used in deep learning (that typically comprise tens or hundreds of thousands of instances).

One possible solution to this problem comes down to generating new bags out of the original ones via subsampling and mixing. Of course, although we are able to generate new bags out of the examples in our dataset, we do not know the (gold) true prevalence of the newly generated bags. However, we can guess it and label our new bags with (silver) prevalence values instead. We propose a heuristic called "Mixer" which operates as follows: given a dataset of type D', at each epoch, we generate new training bags $(B, \hat{\mathbf{p}})$, from the original ones, in which $B = B'_i \bigcup B'_j$, where B'_i (resp. B'_j) is a random subset containing half of the elements of B_i (resp. B_j) and $\hat{\mathbf{p}} = (|B'_i|\mathbf{p}_i + |B'_j|\mathbf{p}_j)/(|B'_i| + |B'_j|)$, and in which bags B_i and B_j are chosen randomly from our original dataset D'. Bags generated with the Mixer are fed into the network along with real bags from D'. A single hyperparameter controls the proportion of real bags used in each iteration. A conceptualization can be consulted in Fig. 4.

This heuristic certainly introduces some noise in the labels of the newly generated bags. However, we found that this noise is generally smaller than the error produced by other surrogate quantifiers would produce if employed in place of the heuristic for estimating the bag prevalence (see the experiment in Fig. 3). We use the Bag Mixer for training all DNN methods.

3.2 Generating a collection of bags from *D*

Many experiments in quantification papers use benchmark datasets borrowed from classification problems. In these datasets, testing bags are not naturally provided so we generate them artificially for testing quantification algorithms. This way, a sampling protocol is employed to generate a sufficiently large collection of testing bags, $D' = \{(B_i, \mathbf{p}_i)\}_{i=1}^{m'}$, with $B_i \in \mathbb{N}^{\mathcal{X}}$ and $\mathbf{p}_i \in \Delta^{l-1}$, from a labeled classification test set $T = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$. The most widely adopted sampling protocol is called artificial prevalence protocol (APP) [35] which is designed to simulate prior probability shift. APP consists of drawing a fixed number of bags in which the bag prevalence $\mathbf{p}_i$ is uniformly drawn at random from the probability simplex Δ^{l-1}, and the testing bag B_i for each class prevalence $\mathbf{p}_i$ is generated from T via random sampling with replacement, trying to maintain $P(X|Y)$ constant. In order to draw prevalence vectors uniformly at random, we use the Kraemer algorithm [36]

Note that the APP protocol is also useful for generating a training dataset of "type D'" from a training dataset of "type D"; that is, when there are no dedicated training bags available but we want to train DNN-based methods using the symmetric approach.

While APP is specialized in generating prior probability shift, notice that if we have some prior knowledge about the application at hand, other sampling protocols designed for reproducing the expected shift could be applied in place [37].

4 HistNetQ: differentiable histograms

In this paper, we propose a permutation-invariant layer for quantification that gains inspiration from histograms. Histograms represent powerful tools for summarizing and describing sets of values: they are directly aligned with the concept of counting, and they disregard the order in which the values are presented. However, histograms are not differentiable operators and hence cannot be directly employed as building blocks in a deep learning model. In order to overcome this impediment, histograms can be approximated by using common differentiable operations such as convolutions and pooling layers. Different realizations of this intuition have been reported in the literature of computer vision [38–42], but, to the best of our knowledge, no one before has investigated differentiable histograms in quantification.

Previous attempts for devising differentiable histograms differ in how these are implemented. On the one hand, [39, 40] proposed soft variants in which every value can potentially contribute to more than one bin, based on the distance of the value to the center of the bin and the width thereof. On the other hand, in [43] the authors propose a hard variant, that is, every value only contributes to the bin in which the value falls. Throughout preliminary experiments we carried out using all variants, we found that the differences in performance were rather small. The hard variant proved slightly better in such experiments (in terms of validation loss) and is our variant of choice for the experiments of Sect. 5. Other architectures and their results are discussed in the supplementary material.

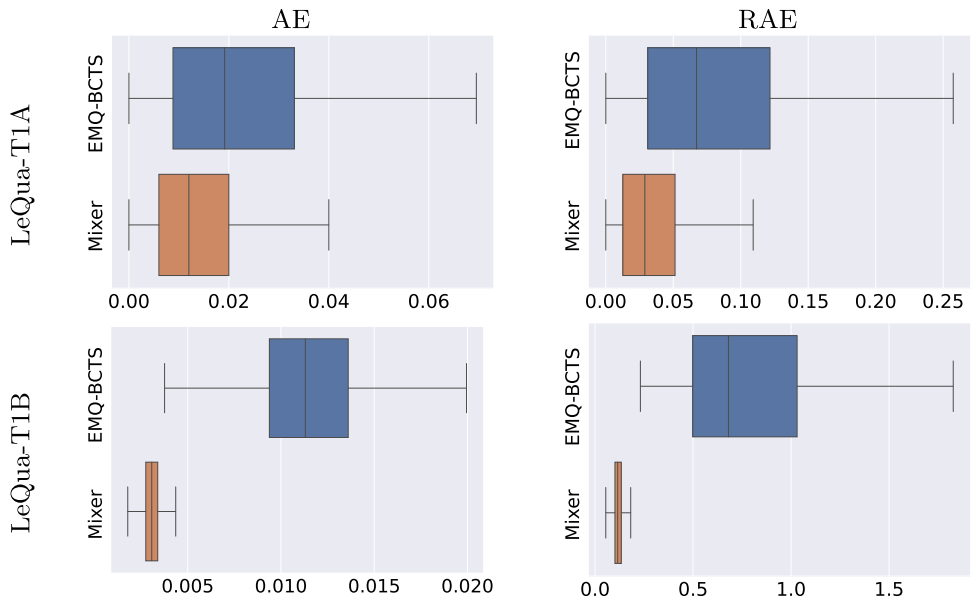

Fig. 3 Distribution of errors produced by EMQ-BCTS and "Mixer" heuristic in terms of Absolute Error (AE) and Relative AE (RAE) as evaluated in LeQua datasets T1A (top row) and T1B (bottom row) (see more details in Sect. 5). EMQ-BCTS was trained and optimized using, respectively, the training and validation sets, and evaluated in the corresponding test bags, while for Mixer we run Monte Carlo simulations generating bags out of the training examples of each task

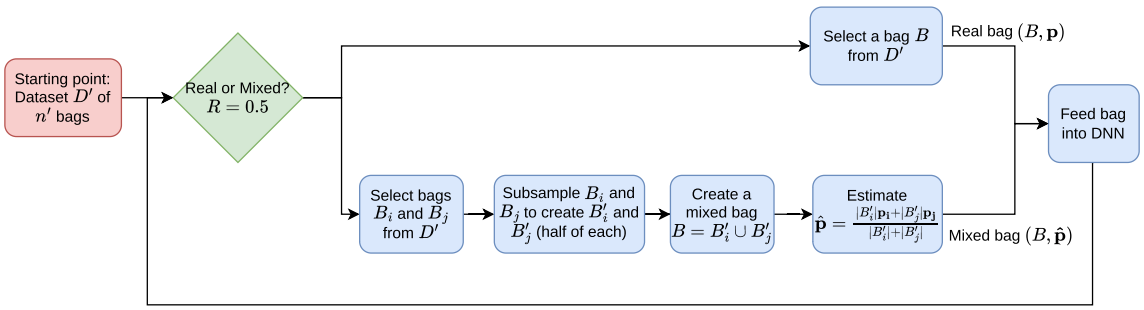

Fig. 4 Flowchart describing the bag mixer augmentation procedure

More formally, given a bag of n data examples $B = \{\mathbf{x}_i\}_{i=1}^{n}$, with $\mathbf{x}_i \in \mathcal{X}$, our goal is to compute a histogram for every feature vector $\{\mathbf{f}_k\}_{k=1}^{d}$, where $\mathbf{f}_k \in \mathbb{R}^n$ represents the values of the k-th feature across the n instances in the bag B, and where d is the number of features extracted. In other words, every histogram is computed along a different column from an $n \times d$ matrix representing B. The hard differentiable histogram layer proposed in [43] takes a user-defined hyperparameter N determining the (fixed) number of bins (we use the same number of bins for all feature vectors), and defines $\{(\mu_b^{(k)}, w_b^{(k)})\}_{b=1}^{N}$, the bin centers and widths, as independent learnable parameters for each feature vector $\mathbf{f}_k$. The value in the b-th bin of the k-th histogram is computed as:

$$H_b^{(k)}(B) = \frac{1}{n}\sum_{i=1}^{n} \phi(\mathbf{f}_k[i]; \mu_b^{(k)}, w_b^{(k)}), \tag{6}$$

where ϕ is defined by:

$$\phi(v; \mu, w) = \begin{cases} 0, & \text{if } 1.01^{w-|v-\mu|} \leq 1 \\ 1, & \text{otherwise.} \end{cases} \tag{7}$$

The value 1.01 in Eq. 7 is justified in [43] simply as a value that yields slightly smaller values than 1 when the exponent is <0 and slightly bigger values than 1 if the exponent is >0. This, in combination with a threshold operation, results in a (differentiable) mechanism to detect which values fall into which bin (see Fig. 5 for a graphical representation of the layer).

Note that we compute densities (by dividing the counts by n) and not plain counts, in order to factor out the effect of the bag size in the final representation. Note also that the total number of parameters of a differentiable histogram layer is $2Nd$. Since the bin centers and widths are learnable, the output can contain interval "gaps" (i.e., intervals in which values are not taken into account), interval overlaps (thus allowing one value to contribute to more than one overlapping bin at the same time), or even zero-width bins. This means that the output of the layer is not strictly a

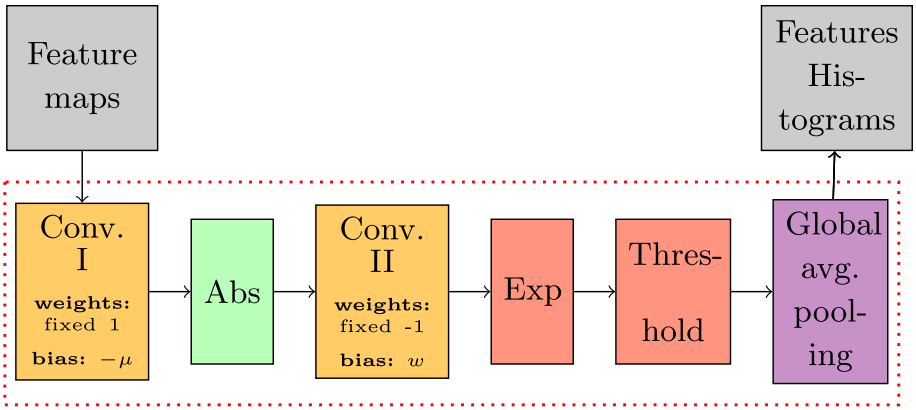

Fig. 5 Learnable histogram layer with hard binning and learnable bin centers and widths. The individual components are common operations used in DL frameworks that we use to compute Eq. 7

histogram, but this allows the model to control the complexity of the representation (should N be too high, the model can well learn to overlap bins or create zero-width ones).

It is worth noting that the quantification method HDy, described in Sect. 2.1, also relies on histograms. However, there are significant differences between HDy and HistNetQ. To begin with, HistNetQ models histograms on the latent representations of the (potentially high-dimensional) data, whereas HDy models histograms on the posterior probabilities returned by a soft classifier. Also, as HistNetQ uses a symmetric approach and learns directly from bags, it does not need to impose any learning assumption, whereas HDy relies on the prior probability shift assumptions. Lastly, HistNetQ enables the optimization of a specific loss function during the learning process, while this is not possible in HDy.

Lemma 4.1 *Hard differentiable histogram layers are permutation-invariant.*

Proof The proof is straightforward. The value $H_b^{(k)}(B)$ is computed by summing over the values returned by the ϕ function. Although $\pi(B)$ with π any permutation function alters the order of the values within the feature vectors $\mathbf{f}_k$, this ordering does not affect the final counts since:

$$\frac{1}{n}\sum_{i=1}^{n}\phi(\mathbf{f}_k[i];\mu_b^{(k)},w_b^{(k)}) = \frac{1}{n}\sum_{i=1}^{n}\phi(\pi(\mathbf{f}_k)[i];\mu_b^{(k)},w_b^{(k)}),$$

and hence $H_b^{(k)}(B) = H_b^{(k)}(\pi(B))$. □

One of the claims of the paper is that polling layers like average, median, or max proposed for set operations [14, 18] can be seen as simplified models (or ablations) of our proposal of using histogram layers (in other words, that a histogram subsumes the information conveyed by these statistics). In order to verify this, we designed a toy experiment where a small neural network is trained to learn each of the aggregation functions (average, median, and max). To this aim, we equip our network with a single histogram layer of 64 bins, followed by just two fully connected layers (sizes 32 and 16). The network is then trained on randomly generated vectors of 100 real values between [0, *max*], where *max* is a random number in the range [0, 1].

The absolute errors are pretty low: 0.0055 (average), 0.0090 (median), and 0.0219 (max) suggesting that histograms are richer representations than the average, median, or max.

As the histogram layer can capture the distribution of the data, it provides a more comprehensive view of the data beyond single summary statistics, something that makes them a promising approach for machine learning tasks that require a density estimation method over sets.

5 Experiments

In this section, we turn to describe the experiments we have carried out in order to assess the performance of our HistNetQ model. To this aim, we have performed two main experiments.[4] The most important one was based on the datasets provided[5] for the LeQua 2022 [21] and the datasets provided[6] for the LeQua 2024 [22], the follow-up edition, competitions, which have become the standard benchmarks for comparing modern quantification methods. These datasets allowed us to conduct a perfectly controlled comparison between asymmetric and symmetric methods.

The LeQua 2022 competition consists of four subtasks of product reviews quantification: two subtasks (T2A and T2B) having to do with raw text documents, and two subtasks (T1A and T1B) in which documents were already

[4] The source code for reproducing the experiments is available at https://github.com/pglez82/histnetq.

[5] https://zenodo.org/record/5734465, see also https://lequa2022.github.io/.

[6] https://zenodo.org/records/11661820, see also https://lequa2024.github.io/.

converted into numerical vectors ($\mathcal{X} \subset \mathbb{R}^{300}$) by the organizers. We focused on T1A and T1B subtasks since we are unconcerned with textual feature extraction in this paper. Both datasets contain real product reviews crawled from the Amazon website. LeQua-T1A is a binary task of estimating the prevalence of positive versus negative opinions, where positive sentiment corresponds to reviews with 4 or 5 stars, while negative sentiment encompasses 1-star and 2-stars reviews (3-stars reviews were filtered out). The organizers provided a training dataset D with 5,000 labeled opinions, a validation set D' with 1,000 bags of 250 unlabeled opinions annotated by prevalence, and 5,000 testing bags of 250 opinions each. T1B is the multiclass task of estimating the prevalence of 28 merchandise product categories, and consists of a training set D with 20,000 labeled opinions, a validation set D' with 1,000 bags of 1,000 unlabeled documents annotated by prevalence, and 5,000 testing bags of 1,000 documents.

A follow-up edition of the LeQua competition was recently published as the LeQua 2024 competition. In this case, tasks T1 and task T2 have the same characteristics as T1A and T1B respectively but using different data and a different feature extractor. In particular, in the LeQua 2022, the features were extracted by averaging 300-dimensional GloVe (static) embeddings, while in LeQua 2024, the features were extracted using the ELECTRA-Small model [44], which outputs 256-dimensional vectors.

We trained our DNN methods using the validation bags D', in line with the symmetric approach (Sect. 3), while traditional quantification methods (Sect. 2.1) were trained using the training set D. Notice that, as could be expected in most applicative domains, the size of the latter (i.e., the number of labeled instances) is larger than the size of the former (i.e., the number of labeled bags). In order to compensate for this shortage of training bags, we employ the Bag Mixer (Sect. 3.1) to train all DNN methods.

The target loss function of both LeQua competitions was the relative absolute error:

$$RAE(\mathbf{p}, \hat{\mathbf{p}}) = \frac{1}{|\mathcal{Y}|} \sum_{c_i \in \mathcal{Y}} \frac{|\delta(p(c_i)) - \delta(\hat{p}(c_i))|}{\delta(p(c_i))}, \tag{8}$$

in which $\delta(p_i) = \frac{p_i + \epsilon}{|\mathcal{Y}|\epsilon + 1}$ is the smoothing function, with ϵ the smoothing factor that we set to $(2|B|)^{-1}$ following [8], where $|B|$ corresponds to the number of instances in the bag B. We also report values of the absolute error,[7] given by

$$AE(\mathbf{p}, \hat{\mathbf{p}}) = \frac{1}{|\mathcal{Y}|} \sum_{c_i \in \mathcal{Y}} |p(c_i) - \hat{p}(c_i)|. \tag{9}$$

We have optimized all DNN methods to minimize the RAE loss, because this was the official evaluation measure. As recalled from Sect. 2, most traditional quantification methods rely on the predictions of an underlying classifier. We use logistic regression in all cases as it has demonstrated consistently good performance across many recent experimental studies [5, 45–47]. The hyperparameters of the classifier were optimized, independently for each quantification method, in terms of RAE in the validation bags, either by the LeQua organizers[8,9] (CC, PCC, ACC, PACC, HDy, QuaNet) or by ourselves (EMQ-BCTS, EMQ-NoCalib), using the QuaPy quantification library [46].

We also use the Fashion-MNIST dataset [48] (a more challenging variant of the well-known MNIST) for the second experiment. In this case, the goal was to analyze the performance of symmetric approaches when the training data consists of individual labeled examples. A side objective of this experiment is to use a different data type. While the LeQua datasets consists of text reviews, this dataset is composed of color images, which forces the network to be equipped with an image-specific feature extraction layer (see Fig. 6). Table 1 shows some statistics of the datasets we used in our experiments.

The training set D consists of 60,000 images, while the test set consists of 10,000 images of 28x28 pixels. Both sets are labeled according to 10 classes. We use the APP protocol (Sect. 3.2) for: i) generating the training bags D' for DNNs methods (500 bags with 500 examples for each epoch), and ii) evaluating all methods (5,000 test bags, of 500 examples each).

To ensure the fairness of the experiment, all the methods used the same feature extraction module (described in Sect. 5.1). The quantifiers learned by DNN methods were optimized for AE or RAE depending on the evaluation measure under consideration. The classifier employed with quantification methods consists of a classification head on top of the feature extraction module, with a softmax activation function, optimized to minimize the cross-entropy loss. We applied early stopping on a validation set in order to prevent overfitting. The validation set was then used to generate the posterior probabilities on which some methods (ACC, PACC) estimate the misclassification rates, and in which EMQ-BCTS optimizes the calibration function.

5.1 A common architecture

In order to guarantee a fair comparison, we used the exact same network architecture, depicted in Fig. 6, for all deep

[7] While many other evaluation metrics for quantification exists in the literature [31], RAE and AE are regarded as the best suited metrics in modern evaluations of quantification performance; see, for example, [5, 45, 46].

[8] https://github.com/HLT-ISTI/QuaPy/tree/lequa2022.

[9] https://github.com/HLT-ISTI/QuaPy/tree/lequa2024.

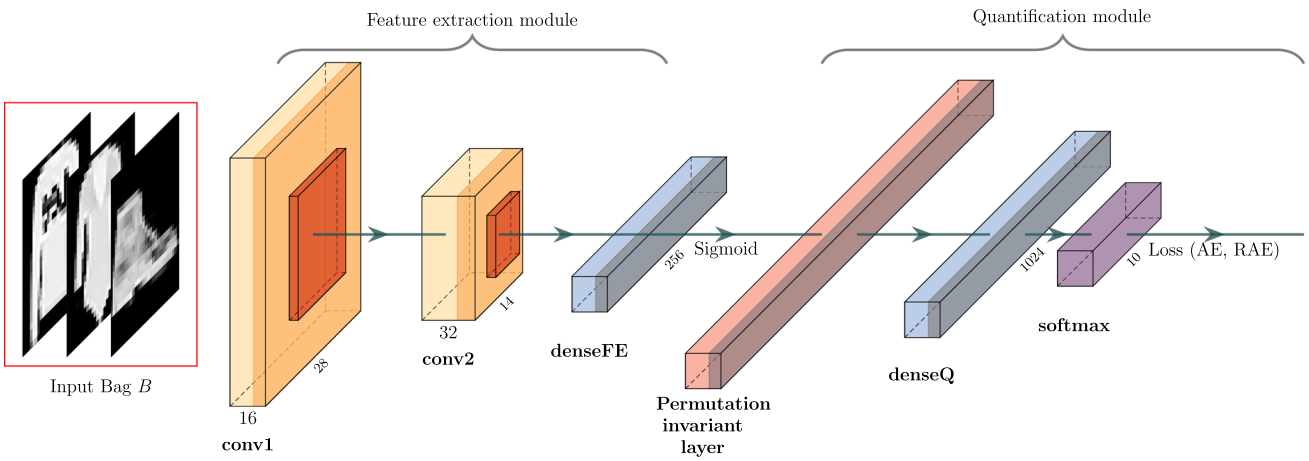

Fig. 6 An example of the common architecture used for DNNs methods. The feature extraction layer and the layer sizes correspond to a computer vision problem (Fashion-MNIST dataset). DenseFE and denseQ are sequences of fully connected layers used in the feature extraction module and in the quantification module, respectively

Table 1 Comparison across the datasets used, focusing on important characteristics like the number of classes, features, and instances

	T1A	T1	T1B	T2	Fashion-MNIST
Data type	Text reviews	Text reviews	Text reviews	Text reviews	Images
Kind of problem	Binary	Binary	Multiclass	Multiclass	Multiclass
Number of classes	2	2	28	28	10
Number of features	300	256	300	256	784
Number of training instances	5000	5000	20000	20000	60000
Number of training bags	1000	1000	1000	1000	–
Bag size	250	250	1000	1000	–
Number of test instances	1250000	1250000	5000000	5000000	10000

learning-based methods, for which we replace only the permutation-invariant layer. The architecture is very similar to the ones previously proposed for set-based problems [14, 18, 19]. The first part of the network is in charge of extracting features from the input examples. For the LeQua datasets, we used a series of fully connected layers each followed by a LeakyReLU activation function and dropout, while for Fashion-MNIST we used a simple CNN with two convolutional layers and one fully connected layer as output (Fig. 6). Note that, if the individual class labels are provided, then the features extracted from this layer might well be used in other downstream tasks as, for example, to train a classifier. However, it is worth noting that our network does not employ any underlying classifier, and therefore these features are used only as the input to a permutation-invariant layer that embeds the bag. The output of this layer is passed then through a feed-forward module followed by a softmax activation, which finally outputs a vector containing the estimated prevalence values $\hat{\mathbf{p}}$. As a final remark, note that one particularity of any neural network adopting the symmetric approach is that, in order to backpropagate the errors through the network, the complete bag must be feedforwarded through it. This is in contrast to other learning endeavors, like neural classification, in which stochastic gradient descent can applied to arbitrary batch sizes of individual data items.

All DNN methods were trained following the same exact procedure: the training set D' was split into an actual training set and a validation set used for monitoring the validation loss; we applied early stopping after 20 epochs without improvement in validation.

Hyperparameters (see supplementary material) were tuned with the aid of OPTUNA [49].

5.2 Results and discussion

Table 2 and Fig. 7 report the results of the multiclass experiments, while Table 3 and Fig. 8 report the results for the binary experiments. Arguably, the most important outcome is that HistNetQ outperforms EMQ in three out of

Table 2 Results for multiclass datasets LEQUA-T1B, LEQUA-T2, and FASHION-MNIST, in terms of AE and RAE. Methods that are not statistically significantly different from the best one (**bold**), according to a Wilcoxon signed-rank test, are marked with † if $0.001 \leq p\text{-value} \leq 0.05$ and with ‡ if $p\text{-value} > 0.05$

	LeQua-T1B		LeQua-T2		Fashion-MNIST	
	AE	RAE	AE	RAE	AE	RAE
CC	0.0141 ± 0.003	1.8936 ± 1.187	0.0166 ± 0.003	2.3096 ± 1.383	0.0163 ± 0.007	0.5828 ± 0.723
PCC	0.0171 ± 0.003	2.2646 ± 1.416	0.0193 ± 0.003	2.6751 ± 1.605	0.0204 ± 0.008	0.7817 ± 0.974
ACC	0.0184 ± 0.004	1.4213 ± 1.270	0.0164 ± 0.004	1.3479 ± 1.161	0.0082 ± 0.003	0.2226 ± 0.238
PACC	0.0158 ± 0.004	1.3054 ± 0.988	0.0155 ± 0.004	1.1942 ± 1.135	0.0067 ± 0.002	0.1831 ± 0.193
EMQ-BCTS	0.0117 ± 0.003	0.9372 ± 0.817	0.0138 ± 0.004	1.1500 ± 0.978	0.0065 ± 0.002	**0.1510 ± 0.152**
EMQ-NoCalib	0.0118 ± 0.003	0.8780 ± 0.751	**0.0134 ± 0.003**	1.1616 ± 0.991	0.0132 ± 0.005	0.2549 ± 0.222
DeepSets (avg)	0.0128 ± 0.004	0.9954 ± 0.658	0.0408 ± 0.010	1.6982 ± 2.263	0.0083 ± 0.003	0.3283 ± 0.233
DeepSets (med)	0.0143 ± 0.004	0.8443 ± 0.543	0.0209 ± 0.006	1.2353 ± 0.891	0.0094 ± 0.003	0.7195 ± 0.586
DeepSets (max)	0.0277 ± 0.005	1.4646 ± 1.026	0.0219 ± 0.004	2.4217 ± 1.879	0.0219 ± 0.007	0.3520 ± 0.323
SetTransformers	0.0385 ± 0.008	1.6748 ± 1.428	0.0384 ± 0.013	3.6275 ± 4.218	0.0104 ± 0.003	2.2017 ± 1.190
HistNetQ (ours)	**0.0107 ± 0.004**	**0.7574 ± 0.489**	0.0181 ± 0.006	**0.9508 ± 0.576**	**0.0060 ± 0.002**	‡0.1592 ± 0.171

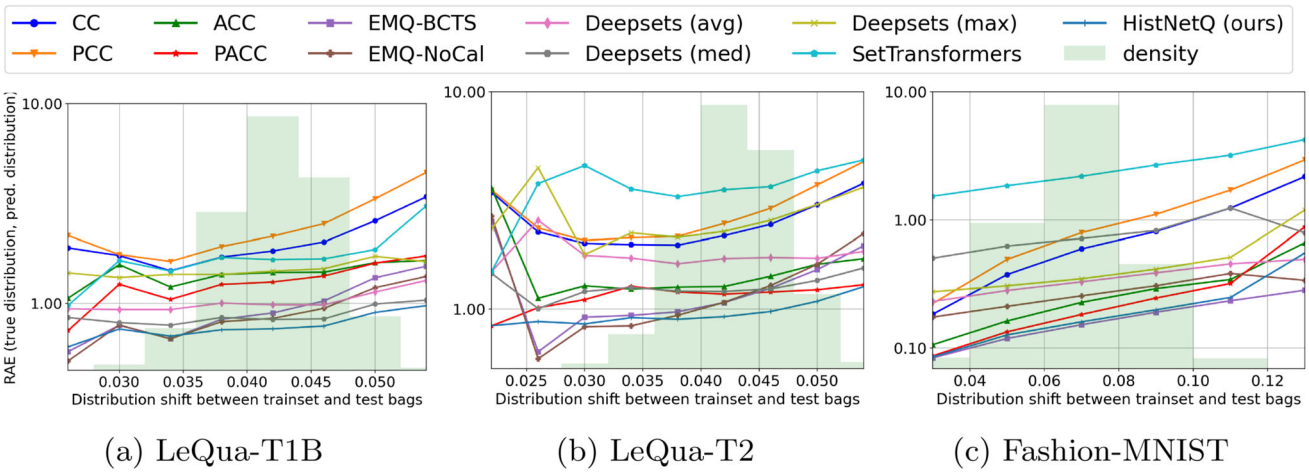

Fig. 7 Error distribution (measured in terms of RAE on a logarithmic scale) binned by the amount of prior probability shift between the training set and each test bag (here measured in terms of $|p_D - p_B|$), for the multiclass datasets. The green bars in the background represent the distribution of bags per bin

four LeQua datasets (namely, in the multiclass datasets LeQua-T1B and LeQua-T2, and in the binary dataset LeQua-T1A), and performs comparably in the binary dataset LeQua-T1. This result is significant because EMQ is considered one of the best quantification methods in the literature [29]. The improvement obtained by HistNetQ is substantial in both multiclass problems in LeQua-T1B (more than 13%) and LeQua-T2 (more than 21%). We think that this is due to two factors: (i) LeQua-T1B and LeQua-T2 are multiclass problems in which it is difficult to accurately estimate the posterior probabilities (a crucial requirement for EMQ), and (ii) the performance of EMQ suffers as the shift between the training set and the test bags increases (see Fig. 7a).

Regarding the comparison of DNN methods, the results show that representing bags using histograms (HistNetQ) brings about better quantification performance than when using SetTransformers or simple aggregation functions (DeepSets) [14] across the five datasets, and the difference in performance seems to correlate with the complexity of the problem, with the multiclass datasets LeQua-T1B and LeQua-T2 standing out as the most challenging datasets among them. We conjecture that this improvement comes from the fact that histogram-based representations are naturally geared toward "counting," and this turns beneficial for quantification. Interestingly, DeepSets(median) obtains the second-best RAE score in T1B. This may be surprising because it uses an apparently simplistic pooling

Table 3 Results for binary datasets LEQUA-T1A and LEQUA-T1 in terms of AE and RAE. Methods that are not statistically significantly different from the best one (**bold**), according to a Wilcoxon signed-rank test, are marked with † if $0.001 \leq p\text{-value} \leq 0.05$ and with ‡ if $p\text{-value} > 0.05$

	LeQua-T1A		LeQua-T1	
	AE	RAE	AE	RAE
CC	0.0916 ± 0.055	1.0840 ± 4.311	0.0796 ± 0.048	0.9774 ± 3.919
PCC	0.1166 ± 0.070	1.3940 ± 5.621	0.1017 ± 0.060	1.2656 ± 5.113
ACC	0.0372 ± 0.029	0.1702 ± 0.508	0.0264 ± 0.020	0.1644 ± 0.603
PACC	0.0298 ± 0.023	0.1522 ± 0.464	0.0240 ± 0.018	0.1339 ± 0.463
HDy	0.0281 ± 0.022	0.1451 ± 0.456	0.0221 ± 0.017	**0.1067 ± 0.290**
QuaNet	0.0342 ± 0.025	0.3176 ± 1.352	0.0243 ± 0.018	0.2640 ± 1.284
EMQ-BCTS	0.0269 ± 0.021	0.1183 ± 0.251	0.0221 ± 0.017	‡0.1097 ± 0.324
EMQ-NoCalib	0.0236 ± 0.018	0.1088 ± 0.267	0.0211 ± 0.017	0.1110 ± 0.367
DeepSets (avg)	0.0278 ± 0.021	0.1269 ± 0.228	0.0208 ± 0.016	0.1096 ± 0.331
DeepSets (med)	0.0292 ± 0.023	0.1389 ± 0.256	0.0237 ± 0.018	0.1235 ± 0.324
DeepSets (max)	0.0499 ± 0.042	0.2183 ± 0.488	0.0447 ± 0.037	0.2191 ± 0.575
SetTransformers	‡0.0225 ± 0.017	‡0.1096 ± 0.262	**0.0202 ± 0.016**	0.1114 ± 0.374
HistNetQ (ours)	**0.0224 ± 0.017**	**0.1071 ± 0.233**	‡0.0204 ± 0.016	0.1069 ± 0.312

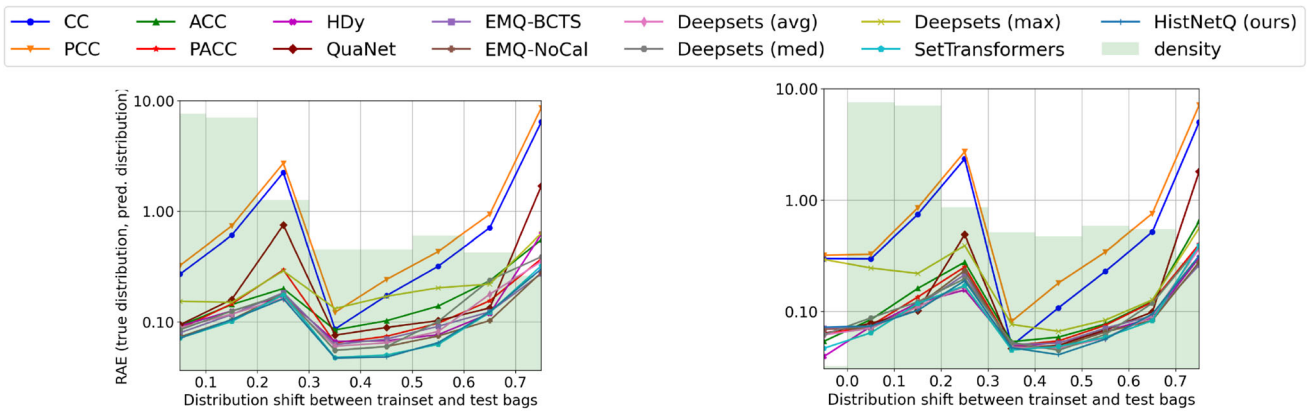

Fig. 8 Error distribution (measured in terms of RAE on a logarithmic scale) binned by the amount of prior probability shift between the training set and each test bag (here measured in terms of $|p_D - p_B|$), for the binary datasets. The green bars in the background represent the distribution of bags per bin

layer. The performance of SetTransformers is erratic: it performs very well in the binary datasets (LeQua-T1A and LeQua-T1), getting a statistically significant result, to HistNetQ in LeQua-T1A, but it obtains the worst results from the deep learning lot in the multiclass datasets LeQua-T1B, LeQua-T2, and Fashion-MNIST. LeQua-T1B and LeQua-T2 are undeniably harder than LeQua-T1A and LeQua-T1 and SetTransformers' inducing points likely struggled to capture the interactions between all the classes. We were unable to make SetTranformers converge to better results in this case, despite trying many combinations of its hyperparameters (number of inducing points, number of heads, etc.). Although transformers are powerful tools in many contexts, they seem not to be the most adequate solution for quantification tasks where the order and the relation between examples in a bag are less important. (This is in contrast to other types of data, such as in natural language processing, where transformers excel in learning from the order and relations between words.)

Table 4 Results by number of bins in LeQua-T1B

	AE	RAE
HistNetQ (8 bins)	0.0297 ± 0.008	1.2878 ± 1.000
HistNetQ (16 bins)	0.0212 ± 0.007	1.0572 ± 0.738
HistNetQ (32 bins)	0.0121 ± 0.005	0.7851 ± 0.520
HistNetQ (64 bins)	**0.0107 ± 0.004**	**0.7574 ± 0.489**

The best results are highlighted in bold

Yet another aspect that proved essential for avoiding overfitting in all DNN methods is the Bag Mixer heuristic. We analyze this in more detail in Sect. 5.3. Concerning HistNetQ, we also analyzed the extent to which the number of bins affects performance (see Table 4). We observe that in complex problems, like LeQua-T1B and LeQua-T2, the performance of HistNetQ tends to improve as the number of bins increases, leading to networks with a higher number of parameters and a richer bag representation. However, this is not necessarily a rule of thumb, because it is reasonable to believe that, in simpler problems, having too many bins might lead to overfitting. We would therefore prescribe treating the number of bins just as any other hyperparameter to be tuned for each specific problem.

The results on Fashion-MNIST show that EMQ with calibration is the best approach, even while requiring less computational resources than DNN methods. According to the literature, these results were to be expected, but the performance of HistNetQ is rather similar and not significantly worse in terms of RAE, while it even fares slightly better in terms of AE. This is important, since it is reasonable to think that the leap in performance HistNetQ delivers in the LeQua competitions with respect to EMQ primarily comes from the way the former exploits the available bags labeled by prevalence. Note both methods use these bags one way or another during the training process; however, while HistNetQ is capable of directly learning from them, EMQ can only use them for optimizing the hyperparameters of the classifier (i.e., for validating different configurations of it). The Fashion-MNIST experiment thus settles down a comparison in which both methods have access to the very same amount of information: a collection of individually labeled instances. Despite the fact that this setup may seem favorable to EMQ, HistNetQ performs on par in terms of RAE, in a statistically significant sense, and even better in terms of AE. However, it is also fair to mention that in this dataset, HistNetQ performs worse when the amount of shift is large (see Fig. 7c). On the other hand, HistNetQ outperforms the rest of the quantification algorithms (only PACC fares similarly) as well as the DNN methods also in this case. As witnessed by the first experiment, DeepSets using median or average polling layers prove more stable than SetTransformers, especially for RAE. Altogether, these results seem to suggest that HistNetQ is competitive in this scenario too and should be considered even for problems in which the supervised information at one's disposal comes in the form of individually labeled examples.

5.3 Ablation study

As discussed in Sect. 3.1, the Bag Mixer is a data augmentation technique meant to enhance the training data of DNN symmetric quantifiers in order to avoid overfitting. In this section, we analyze the extent to which the Bag Mixer impacts the performance of each network. To do so, we carry out additional experiments in which the Bag Mixer is not used (that is, using only the training bags provided in D'), and we compare the results with those previously reported in Tables 2 and 3. For this experiment, we concentrate on two of the datasets (LeQua-T1A and LeQua-T1B) where the training bags with their corresponding prevalence values are provided and thus limited.

For LeQua-T1A, Fig. 9 and Table 5 show that the networks exhibit lower training errors when using only 1000 training bags. However, this reduction in training error leads to a significant drawback, as these models tend to perform notably worse on the validation data as a clear consequence of overfitting.

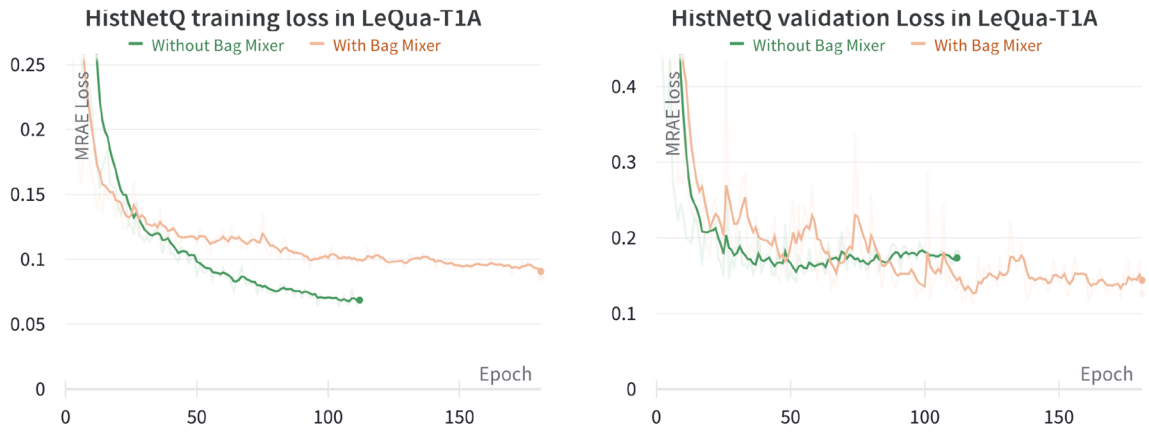

Fig. 9 Training and validation loss trends of HistNetQ in LeQua-T1A, with and without the Bag Mixer using a patience criterion (i.e. stopping the training process after a number of consecutive epochs in which the validation loss does not improve). The training loss decreases faster without the Bag Mixer (left figure); however, the validation loss keeps improving with the Bag Mixer (right figure). This is an indication that the Bag Mixer helps counter overfitting

Table 5 LeQua-T1A results without Bag Mixer. Relative error variation with respect to when using Bag Mixer (Table 3) is shown in parenthesis

	AE	RAE
Deepsets (avg)	0.0326 (+17.3%)	0.1469 (+15.8%)
Deepsets (median)	0.0416 (+42.5%)	0.1810 (+30.3%)
Deepsets (max)	0.0570 (+14.2%)	0.2287 (+4.8%)
SetTransformers	0.0368 (+63.6%)	0.1553 (+41.1%)
HistNetQ (ours)	0.0279 (+24.6%)	0.1265 (+18.1%)

Table 6 LeQua-T1B results without Bag Mixer. Relative error variation with respect to when using the Bag Mixer (Table 2) is shown in parenthesis

	AE	RAE
Deepsets (avg)	0.0449 (+250.8%)	1.5029 (+51%)
Deepsets (median)	0.0215 (+50.4%)	1.0991 (+30.2%)
Deepsets (max)	0.0200 (−27.8%)	1.5740 (+7.5%)
SetTransformers	0.0311 (−19.2%)	4.2416 (+153.3%)
HistNetQ (ours)	0.0445 (+315.9%)	1.5108 (+99.5%)

In the case of LeQua-T1B, Table 6, the adverse effects of overfitting are amplified. This was to be expected, since the number of classes is much higher in this dataset (up to 28), while the number of training bags stays the same (i.e., 1000). In this scenario, the networks, especially those with more complex architectures containing a greater number of parameters, such as SetTranformers or HistNetQ, prove more prone to overfit.

6 Conclusions

This paper introduces HistNetQ, a DNN for quantification that relies on a permutation-invariant layer based on differentiable histograms. We carried out experiments using different quantification problems (from computer vision and text analysis) in which we compared the performance of HistNetQ against previously proposed networks for set processing and also against the most important algorithms from the quantification literature. The results show that HistNetQ achieved state-of-the-art performance in both problems.

From a qualitative point of view, HistNetQ also offers several noteworthy properties like i) the ability to directly learn from bags labeled by prevalence, which allows HistNetQ to be applied to scenarios in which traditional methods cannot; and ii) the possibility to directly optimize for specific loss functions.

This research may hopefully offer a new viewpoint in quantification learning, since our results suggest that exploiting data labeled at the aggregate level might be more effective, in terms of quantification performance than exploiting data labeled at the individual level. Overall, this study suggests that HistNetQ is a promising alternative for implementing the symmetric approach in real applications, obtaining state-of-the-art results that surpass previous approaches.

Future work may include i) studying the capabilities of HistNetQ when confronted with types of dataset shift other than prior probability shift [37, 50], ii) exploring potential applications of this architecture to other problems that, like quantification, require learning a model from density estimates over sets of examples and iii) investigating new representations alternative to histograms, along the lines of [47], that may be able to capture the interrelationships between features that may exist in the data

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00521-024-10721-1.

Acknowledgement The work by the first, third and fourth authors has been funded by MINECO (the Spanish Ministerio de Economía y Competitividad) and FEDER (Fondo Europeo de Desarrollo Regional), grant PID2019-110742RB-I00 (MINECO/FEDER). The work of the second author has been supported by the SoBig- Data.it and FAIR projects funded by the Italian Ministry of University and Research under NextGenerationEU program, and the QuaDash project "Finanziato dall'Unione europea-Next Generation EU, Missione 4 Componente 1 CUP B53D23026250001".

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

Data Availability Statement The code discussed and developed in this paper is available in a GitHub open repository, accessible at https://github.com/pglez82/histnetq. Detailed instructions for executing this code can be found in the repository's README file. For the experiments analyzed in Sect. 5, the datasets are available from two different sources. Firstly, the datasets from the LeQua Competition 2022 [21], where T1A and T1B are included, consist in product reviews converted into numerical vectors. They are accessible at https://zenodo.org/record/5734465. Secondly, the datasets from the LeQua Competition 2024 [22], including T1 and T2, consist in product reviews that are converted into numerical vectors. These datasets can be accessed at https://zenodo.org/records/11661820. Lastly, the Fashion-MNIST dataset [48], a more challenging variant of the well-known MNIST, can be obtained from https://github.com/zalandoresearch/fashion-mnist.

Declarations

Conflict of interest No potential conflict of interest was reported by the authors.

References

1. Beijbom O, Hoffman J, Yao E, Darrell T, Rodriguez-Ramirez A, Gonzalez-Rivero M, Guldberg OH- (2015) Quantification in-the-wild: data-sets and baselines. arXiv:1510.04811 [cs] (2015). arXiv: 1510.04811
2. Forman G (2006) Quantifying trends accurately despite classifier error and class imbalance. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining (KDD 2006), Philadelphia, US, pp. 157–166. https://doi.org/10.1145/1150402.1150423
3. González P, Castaño A, Peacock EE, Díez J, Del Coz JJ, Sosik HM (2019) Automatic plankton quantification using deep features. J Plankton Res 41(4):449–463
4. Hopkins D, King G (2010) A method of automated nonparametric content analysis for social science. Am J Polit Sci 54(1):229–247
5. Moreo A, Sebastiani F (2022) Tweet sentiment quantification: an experimental re-evaluation. PLOS ONE 17(9):1–23. https://doi.org/10.1371/journal.pone.0263449
6. Dias FF, Ponti MA, Minghim R (2022) A classification and quantification approach to generate features in soundscape ecology using neural networks. Neural Comput Appl 34(3):1923–1937
7. González P, Díez J, Chawla N, Coz JJ (2017) Why is quantification an interesting learning problem? Prog Artif Intell 6(1):53–58. https://doi.org/10.1007/s13748-016-0103-3
8. Forman G (2008) Quantifying counts and costs via classification. Data Min Knowl Discov 17(2):164–206. https://doi.org/10.1007/s10618-008-0097-y
9. González-Castro V, Alaiz-Rodríguez R, Alegre E (2013) Class distribution estimation based on the Hellinger distance. Inf Sci 218:146–164
10. Kawakubo H, Du Plessis MC, Sugiyama M (2016) Computationally efficient class-prior estimation under class balance change using energy distance. IEICE TRANS Inf Syst 99(1):176–186
11. Quionero-Candela J, Sugiyama M, Schwaighofer A, Lawrence ND (2009) Dataset shift in machine learning. The MIT Press, Cambridge, MA
12. González P, Castaño A, Chawla NV, Coz JJD (2017) A review on quantification learning. ACM Comput Surv (CSUR) 50(5):1–40
13. Esuli A, Fabris A, Moreo A, Sebastiani F (2023) Learning to quantify. Springer, Cham, CH. https://doi.org/10.1007/978-3-031-20467-8
14. Qi L, Khaleel M, Tavanapong W, Sukul A, Peterson D (2021) A framework for deep quantification learning. In: Machine learning and knowledge discovery in Databases: European conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part I, pp. 232–248. Springer
15. Edwards H, Storkey AJ (2017) Towards a neural statistician. In: 5th International conference on learning representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings
16. Murphy RL, Srinivasan B, Rao VA, Ribeiro B (2019) Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. In: 7th international conference on learning representations, ICLR 2019, May 6-9, 2019. OpenReview.net, New Orleans, LA, USA
17. Wagstaff E, Fuchs F, Engelcke M, Posner I, Osborne MA (2019) On the limitations of representing functions on sets. In: International conference on machine learning, pp. 6487–6494. PMLR
18. Zaheer M, Kottur S, Ravanbakhsh S, Poczos B, Salakhutdinov RR, Smola AJ (2017) Deep sets. Adv Neural Inf Process Syst. https://doi.org/10.48550/arXiv.1703.06114
19. Lee J, Lee Y, Kim J, Kosiorek A, Choi S, Teh YW (2019) Set transformer: A framework for attention-based permutation-invariant neural networks. In: International conference on machine learning, pp. 3744–3753. PMLR
20. Esuli A, Moreo A, Sebastiani F (2018) A recurrent neural network for sentiment quantification. In: Proceedings of the 27th ACM international conference on information and knowledge management (CIKM 2018), Torino, IT, pp. 1775–1778. https://doi.org/10.1145/3269206.3269287
21. Esuli A, Moreo A, Sebastiani F, Sperduti G (2022) A detailed overview of LeQua@CLEF 2022: Learning to quantify. In: Proceedings of the working notes of CLEF 2022 - conference and Labs of the evaluation Forum, Bologna, Italy, September 5th-8th, 2022. CEUR Workshop Proceedings, vol. 3180, pp. 1849–1868. CEUR-WS.org, Bologna, Italy
22. Esuli A, Moreo A, Sebastiani F, Sperduti G (2024) A detailed overview of LeQua@LQ 2024: Learning to quantify. In: Proceedings of the workshop learning to quantify: methods and applications (LQ 2024) Vilnius, Lithuania, September 13, 2024, Vilnius, Lithuania
23. Fernandes Vaz A, Izbicki R, Bassi Stern R (2019) Quantification under prior probability shift: the ratio estimator and its extensions. J Mach Learn Res 20:79–17933
24. Lipton Z, Wang Y-X, Smola A (2018) Detecting and correcting for label shift with black box predictors. In: International conference on machine learning, pp. 3122–3130. PMLR
25. Tasche D (2024) Comments on Friedman's method for class distribution estimation. arXiv:2405.16666 [cs.LG]
26. Bunse M (2022) On multi-class extensions of adjusted classify and count. In: Proceedings of the 2nd international workshop on learning to quantify (LQ 2022), Grenoble, IT, pp. 43–50
27. Bella A, Ferri C, Hernández-Orallo J, Ramírez-Quintana MJ (2010) Quantification via Probability Estimators. In: 2010 IEEE International conference on data mining, pp. 737–742. https://doi.org/10.1109/ICDM.2010.75 . ISSN: 2374-8486
28. Saerens M, Latinne P, Decaestecker C (2002) Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. Neural Comput 14(1):21–41. https://doi.org/10.1162/089976602753284446
29. Alexandari A, Kundaje A, Shrikumar A (2020) Maximum likelihood with bias-corrected calibration is hard-to-beat at label shift adaptation. In: International conference on Machine learning, pp. 222–232. PMLR
30. Esuli A, Molinari A, Sebastiani F (2020) A critical reassessment of the Saerens-Latinne-Decaestecker algorithm for posterior probability adjustment. ACM Trans Inf Syst(TOIS) 39(2):1–34
31. Sebastiani F (2020) Evaluation measures for quantification: an axiomatic approach. Inf Retr J 23(3):255–288. https://doi.org/10.1007/s10791-019-09363-y

32. Foulds JR, Frank E (2010) A review of multi-instance learning assumptions. Knowl Eng Rev 25(1):1–25. https://doi.org/10.1017/S026988890999035X
33. Freitas N, Kück H (2005) Learning about individuals from group statistics. In: Proceedings of the 21st conference in uncertainty in artificial intelligence (UAI 2005), Edimburgh, UK, pp. 332–339
34. Quadrianto N, Smola AJ, Caetano TS, Le QV (2009) Estimating labels from label proportions. J Mach Learn Res 10:2349–2374
35. Forman G (2005) Counting positives accurately despite inaccurate classification. In: Proceedings of the 16th European conference on machine learning (ECML 2005), Porto, PT, pp. 564–575. https://doi.org/10.1007/11564096_55
36. Smith NA, Tromble RW (2004) Sampling uniformly from the unit simplex. Johns Hopkins University, Tech. Rep 29
37. Zhang K, Schölkopf B, Muandet K, Wang Z (2013) Domain adaptation under target and conditional shift. In: ICML, pp. 819–827
38. Avi-Aharon M, Arbelle A, Raviv TR (2020) Deephist: Differentiable joint and color histogram layers for image-to-image translation. arXiv preprint arXiv:2005.03995
39. Peeples J, Xu W, Zare A (2022) Histogram layers for texture analysis. IEEE Trans Artif Intell 3(4):541–552. https://doi.org/10.1109/TAI.2021.3135804
40. Wang Z, Li H, Ouyang W, Wang X (2016) Learnable histogram: Statistical context features for deep neural networks. In: European conference on computer vision, pp. 246–262. Springer
41. Hussain MA, Hamarneh G, Garbi R (2019) Renal cell carcinoma staging with learnable image histogram-based deep neural network. In: Suk H-I, Liu M, Yan P, Lian C (eds) Machine learning in medical imaging. Springer, Cham, pp 533–540
42. Hussain MA, Hamarneh G, Garbi R (2019) Imhistnet: learnable image histogram based dnn with application to noninvasive determination of carcinoma grades in ct scans. In: Shen D, Liu T, Peters TM, Staib LH, Essert C, Zhou S, Yap P-T, Khan A (eds) Medical image computing and computer assisted intervention - MICCAI 2019. Springer, Cham, pp 130–138
43. Yusuf I, Igwegbe G, Azeez O (2020) Differentiable histogram with hard-binning. arXiv preprint arXiv:2012.06311
44. Clark K, Luong M-T, Le QV, Manning CD (2020) ELECTRA: Pre-training text encoders as discriminators rather than generators. In: Proceedings of the 8th international conference on learning representations (ICLR 2020), Addis Ababa, ET. https://openreview.net/pdf?id=r1xMH1BtvB
45. Schumacher T, Strohmaier M, Lemmerich F (2021) A comparative evaluation of quantification methods. arXiv:2103.03223v1 [cs.LG]
46. Moreo A, Esuli A, Sebastiani F (2021) QuaPy: a python-based framework for quantification. In: Proceedings of the 30th ACM international conference on information & knowledge management, pp. 4534–4543
47. Moreo A, González P, Coz JJ (2024) Kernel density estimation for multiclass quantification. arXiv preprint arXiv:2401.00490
48. Xiao H, Rasul K, Vollgraf R (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747
49. Akiba T, Sano S, Yanase T, Ohta T, Koyama M (2019) Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25rd ACM SIGKDD international conference on knowledge discovery and Data Mining
50. Tasche D (2022) Class prior estimation under covariate shift: No problem? In: Proceedings of the 2nd international workshop on learning to quantify: methods and applications (LQ 2022), ECML/PKDD. arXiv:2206.02449 [stat.ML], Grenoble (France)