Uncertainty-Aware Explanations Through Probabilistic Self-Explainable Neural Networks

Jon Vadillo[1], **Roberto Santana**[1], **Jose A. Lozano**[1,2], **Marta Kwiatkowska**[3]

[1]Department of Computer Science and Artificial Intelligence,
University of the Basque Country UPV/EHU
[2]Basque Center for Applied Mathematics (BCAM)
[3]Department of Computer Science, University of Oxford
{jon.vadillo, roberto.santana, ja.lozano}@ehu.eus,
marta.kwiatkowska@cs.ox.ac.uk

Abstract

The lack of transparency of Deep Neural Networks continues to be a limitation that severely undermines their reliability and usage in high-stakes applications. Promising approaches to overcome such limitations are Prototype-Based Self-Explainable Neural Networks (PSENNs), whose predictions rely on the similarity between the input at hand and a set of prototypical representations of the output classes, offering therefore a deep, yet transparent-by-design, architecture. So far, such models have been designed by considering pointwise estimates for the prototypes, which remain fixed after the learning phase of the model. In this paper, we introduce a probabilistic reformulation of PSENNs, called Prob-PSENN, which replaces point estimates for the prototypes with probability distributions over their values. This provides not only a more flexible framework for an end-to-end learning of prototypes, but can also capture the explanatory uncertainty of the model, which is a missing feature in previous approaches. In addition, since the prototypes determine both the explanation and the prediction, Prob-PSENNs allow us to detect when the model is making uninformed or uncertain predictions, and to obtain valid explanations for them. Our experiments demonstrate that Prob-PSENNs provide more meaningful and robust explanations than their non-probabilistic counterparts, thus enhancing the explainability and reliability of the models.

1 Introduction

The complexity in the architectures of state-of-the-art Deep Neural Networks (DNNs) largely accounts for their "black box" nature [34], which is in conflict with one of the basic requirements for trustworthiness [5]: to be able to explain and understand the decisions of the model. Although several strategies have been proposed in order to explain black box models [47], they often provide only partial information about their inner workings, such as which parts of the input at hand most condition the output [6, 35, 37, 39–41]. However, these approaches do not provide information about how the model processes that information or why they imply the output [2, 18, 29, 34]. In order to achieve a more transparent classification process, recent works advocate the use of Prototype-Based Self-Explainable Neural Networks (PSENNs, for simplicity) [4, 8, 14, 18, 28], which are trained to jointly maximize their prediction performance and their explainability. These approaches rely primarily on prototype-based models, in which the prototypes are learned during the training phase, aiming to capture discriminative and also semantically-meaningful features. In this way, the output classification is based on the similarity between the input and the learned prototypes, providing an intuitive and human-understandable classification process.

On the other hand, current DNNs models, including the PSENNs discussed above, are optimized with the aim of finding the point estimate values of the parameters that minimize a given loss term, leading to models with a deterministic inference process. An important drawback of these approaches is that, generally, it is not possible to measure to what extent the model is confident in its own prediction, known in the literature as the epistemic uncertainty of the model [13, 20, 25]. Indeed, it has been shown that DNNs tend to be overconfident in their predictions [17], classifying inputs with high probability even when random or out-of-distribution inputs are provided. Being unable to detect when the model is making such unreliable or uninformed decisions dramatically reduces the trustworthiness and reliability of the model, and is, therefore, of great concern in high-stakes applications. To address this issue, recent works proposed replacing point estimates for the parameters of the model with a probability distribution over their values [16]. This leads to a stochastic model, for which it is possible to measure the aforementioned types of uncertainty, and, thus, identify when the model is making uncertain predictions that should not be trusted.

In this paper, we aim to address the aforementioned drawbacks and challenges by leveraging recent tools from uncertainty estimation in DNNs, and combining them with PSENN architectures in order to develop Probabilistic Self-Explainable Neural Networks (Prob-PSENNs). The key feature of this model is that the prototypes are defined as random variables, for which suitable probability distributions over their values are learned during the training phase of the model – jointly with the rest of the parameters of the network. This probabilistic paradigm allows us not only to improve the explainability of the classification process, but also to capture different sources of uncertainty regarding the explanation. Furthermore, since the output of the PSENNs directly depends on the explanatory component [28], we can establish connections between their corresponding uncertainties, enabling a more thorough analysis of the prediction process, and allowing the models to self-explain their own uncertainties.

The main contributions of this work are summarized below:

1. First, we introduce Prob-PSENN, a paradigm shift for PSENNs, which relies on probabilistic modeling of the explanatory component, thus replacing point estimates for the prototypes with probability distributions over their values.
2. We demonstrate how the probabilistic reinterpretation of the prototypes substantially enhances the explanatory capabilities of the model, enabling more diverse, meaningful and robust explanations than those achievable with the non-probabilistic counterparts.
3. Furthermore, Prob-PSENNs allow us, for the first time, to model the uncertainty in the explanations and in the predictions, which is a missing feature in conventional PSENNs. More specifically, we formalize different explanatory uncertainty notions, and show how Prob-PSENNs enable not only the quantification of those uncertainties, but also their explanations, thus increasing reliability and trustworthiness.
4. The effectiveness of Prob-PSENN is evaluated in several classification tasks, demonstrating that it can effectively learn highly diverse and prototypical representations of the output classes, capture the model uncertainties and maintain competitive predictive performance.

2 Preliminaries: Prototype-Based Self-Explainable Neural Networks (PSENNs)

Let us consider a classification problem in which the goal is to classify d-dimensional inputs $x \in \mathbb{R}^d$ into one of the c possible classes in $Y = \{y_1, y_2, \dots, y_c\}$. Using as baseline the architecture introduced in [28], illustrated in Figure 1, the main components of the PSENNs are described as follows. We refer the reader to Appendix A for a justification for the choice of this baseline, as well as for a discussion on related works.

The first component of this architecture is an encoder module $e : \mathbb{R}^d \to \mathbb{R}^l$, where l denotes the dimensionality of the *latent* space to which the input is encoded. The latent representation computed by the encoder $e(\cdot)$ for the input x will be denoted as $e_x = e(x)$. Oppositely, let $g : \mathbb{R}^l \to \mathbb{R}^d$ denote a decoding module, capable of reconstructing the original input x from the corresponding encoded representation e_x. The key component of this architecture involves the generation of m prototypes in the latent space: $R = (r_1, r_2, \dots, r_m)$, with $r_i \in \mathbb{R}^l$, $1 \leq i \leq m$. The aim of generating these prototypes is to capture diverse, discriminative and semantically meaningful features capable of

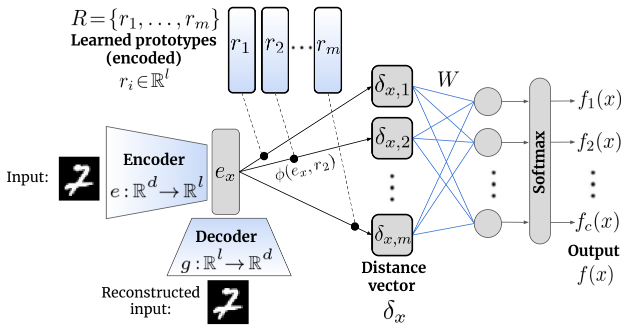


Figure 1: Architecture of the Prototype-Based Self-Explainable Neural Network [28] described in Section 2, illustrated for a handwritten digit classification task [27].

representing the classes of the problem. In this way, the similarity between an encoded input e_x and each of the m prototypes can be employed to determine the output class for x in an easily interpretable manner, based on the assumption that an input x belonging to the class y_i will be closer to those prototypes representing the class y_i than to the prototypes corresponding to the remaining classes. To formalize the aforementioned similarities, let the column-vector $\delta_x = (\delta_{x,1}, \delta_{x,2}, \ldots, \delta_{x,m})^\intercal$ represent the distance between an encoded input e_x and the m prototypes in R. More detailedly, the distance between e_x and the i-th prototype r_i will be computed as $\delta_{x,i} = \phi(e_x, r_i),\ 1 \leq i \leq m$, based on a distance metric $\phi : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R}$. The distance vector δ_x will condition both the output classification and the corresponding explanation, as it is detailed below.

Classification Taking δ_x as the input, the output provided by the model will be determined by a prototype classifier $h : \mathbb{R}^m \to [0,1]^c$. As in [28], a classification network $h(\delta_x) = s(W\delta_x)$ will be assumed, being W a $c \times m$ weight matrix and $s(\cdot)$ the *softmax function*. For simplicity, the end-to-end classification process will be denoted as $f(x) = h(\delta_x)$. Notice that $f(x)$ outputs a vector representing the probability with which the input belongs to each of the possible classes of the problem, according to the model. The probability assigned to each class y_i will be denoted as $f_i(x)$, $1 \leq i \leq c$, and thus we can define the output as: $f(x) = \big(f_1(x), f_2(x), \ldots, f_c(x)\big)$. Generally, the class with the highest probability is considered the output class of the prediction, what will be denoted as $f_*(x) = \arg\max_{y_i \in Y} f_i(x)$.

Explanation Since the output class is based on the distance-vector δ_x, the explanation ξ_x for the prediction $f(x)$ can be determined also by the similarity between the prototypes and the input at hand: $\xi_x = \big\{\{g(r_1), \delta_{x,1}\}, \{g(r_2), \delta_{x,2}\}, \ldots, \{g(r_m), \delta_{x,m}\}\big\}$, being $g(\cdot)$ the decoder defined above. Notice that the interpretability of the model relies, therefore, on to what extent the decoded prototypes are capable of capturing relevant and semantically-meaningful features representing each class of the problem, which is enforced during the training phase by means of including interpretability regularizers in the loss function [28].

3 Probabilistic Self-Explainable Neural Networks (Prob-PSENNs)

As mentioned before, PSENNs treat the prototypes as network parameters, for which point estimate values are computed and hence are fixed once the training is completed. Furthermore, the number of prototypes m needs to be finite and defined beforehand, even if, in practice, it might result cumbersome or infeasible to determine how many prototypes will be required to capture sufficiently diverse characterizations of the classes.

To address these issues, we introduce Prob-PSENNs, generalizing the prototypes by means of defining a probability distribution over their values. The main motivation for this choice is twofold. On the one hand, for each class of the problem, there might exist several sets of prototypes R capable of representing the data with a comparable effectiveness. For illustration, let us consider a handwritten digit classification scenario, and the fact that there are several ways of representing each digit. This implies that there are several representative prototypes for each class. Consequently, instead of relying on a single set of prototypes, it might be more reasonable to consider all those possible variations

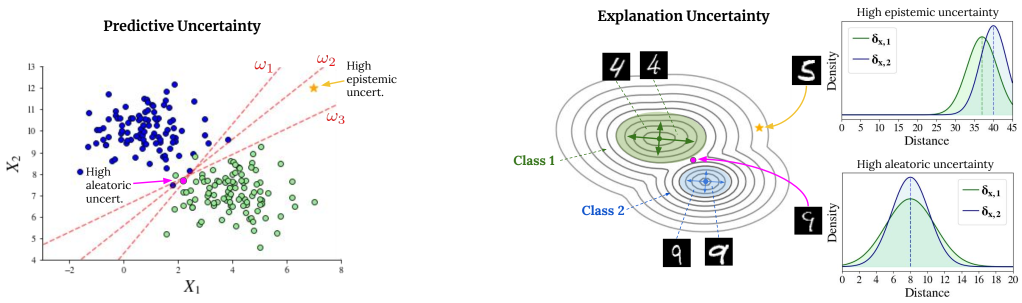


Figure 2: Comparison between the predictive and explanatory uncertainty, illustrated for a binary classification problem with a two-dimensional input space.

in order to classify an input. This probabilistic reformulation also enables a flexible way to decide the variety of the prototypes required to represent the classes, which will be determined, assuming one prototype per class, by the variance of the distribution over the prototypes. On the other hand, the choice of the prototypes in R conditions not only the output but also the explanation. Therefore, placing a probability distribution over the prototypes allows us to enable a probabilistic framework for the generation of the explanation as well, which can be employed to capture the uncertainty in the explanations of the model, and, thereby, produce more informative and useful explanations.

3.1 Predictive distribution

Instead of considering a point estimate for R, we will consider it a random variable, for which we will use the notation $\boldsymbol{R}$. Furthermore, we will assume $m = c$ prototypes, such that $\boldsymbol{r}_i$ represents the class y_i, $1 \leq i \leq c$. Due to the randomness in the network's prototypes, the inference of the model will be stochastic, and will be denoted as $f^{\boldsymbol{R}}(x)$. In this way, a predictive distribution for the output can be defined:

$$p(\boldsymbol{y}|x) = \mathbb{E}_{p(\boldsymbol{R})}[p(\boldsymbol{y}|x, R)] = \int p(\boldsymbol{y}|x, \boldsymbol{R}) \cdot p(\boldsymbol{R})\, d\boldsymbol{R} = \int f^{\boldsymbol{R}}(x) \cdot p(\boldsymbol{R})\, d\boldsymbol{R}. \tag{1}$$

In practice, $p(\boldsymbol{y}|x)$ can be approximated by considering a finite number of N samples from $p(\boldsymbol{R})$:

$$\tilde{p}(\boldsymbol{y}|x) = \bar{f}(x) = \frac{1}{N} \sum_{n=1}^{N} f^{R_n}(x), \;\; \text{with} \;\; R_n \sim p(\boldsymbol{R}),\; 1 \leq n \leq N. \tag{2}$$

3.2 Capturing the predictive uncertainty

The predictive distribution defined in Equation (1) allows us to better model the *predictive uncertainty* in comparison to its deterministic counterpart. First, the total predictive uncertainty can be measured by means of the entropy [36] of the prediction: $H[p(y|x)] = -\sum_{i=1}^{c} p(y = y_i|x) \cdot \log p(y = y_i|x) \cdot$ However, for a model with fixed (pointwise) values for the parameters, the source of this uncertainty will be only *aleatoric* uncertainty, which refers to the possibly unavoidable noise, randomness or ambiguity inherent in the data. Considering a probability distribution for the parameter values instead of point estimates allows us to capture also the epistemic uncertainty [20, 22, 25], which accounts for the model uncertainty about the true parameters modeling the data (e.g., as a consequence of insufficient training data). This enables a safer use of the model in critical tasks, since it allows us to identify when the model is making unreliable predictions. An illustrative comparison between aleatoric and epistemic uncertainty is provided in Figure 2 (left).

Thus, for the predictive distribution defined in Equation (1), $H[p(y|x)]$ encapsulates both aleatoric and epistemic uncertainty. To measure epistemic uncertainty only, the mutual information between the output $\boldsymbol{y}$ and the parameters $\boldsymbol{R}$ can be employed [11, 13, 19, 20], which, in our case, can be formalized as: $I(\boldsymbol{y}, \boldsymbol{R}|x) = H(\boldsymbol{y}|x) - \mathbb{E}_{p(\boldsymbol{R})}[H(\boldsymbol{y}|x, \boldsymbol{R})]$, while $\mathbb{E}_{p(\boldsymbol{R})}[H(\boldsymbol{y}|x, \boldsymbol{R})]$ represents a measure of aleatoric uncertainty. Notice that the values of all these metrics can be normalized to the range $[0, 1]$, where 1 would represents the maximum uncertainty, by dividing the resulting value by the entropy of the uniform categorical distribution of c categories. In practice, $H[p(y|x)]$ can be approximated by $H[\tilde{p}(y|x)]$ (see Equation 2).

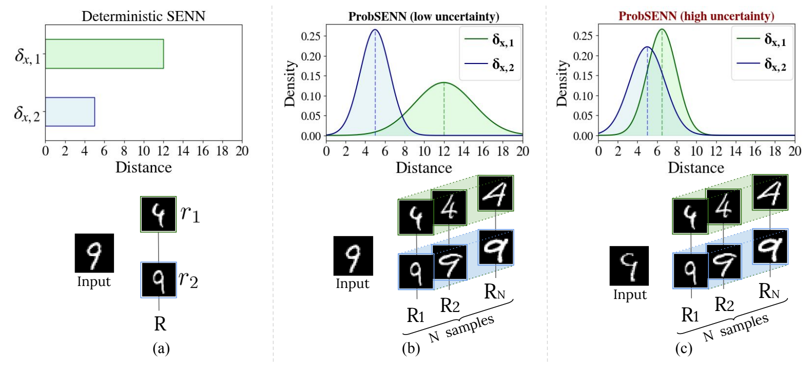


Figure 3: Illustrative comparison between a conventional PSENN (left column) and the proposed Probabilistic-PSENN (middle and right columns) regarding the computation of the distance-vector δ_x. Whereas a fixed set of prototypes R is used in (a), leading to a point estimate δ_x, placing a probability distribution over $\boldsymbol{R}$, as in (b) and (c), induces a probability distribution over the distance $\delta_{x,i}$ between the encoded input and the random prototypes $\boldsymbol{r}_i$ representing the class y_i, $1 \leq i \leq c$.

3.3 Capturing the explanation uncertainty

With a PSENN with fixed prototypes, only a point estimate δ_x can be provided for the input at hand, that is, a point estimate on how close the input is to the particular prototypes selected to represent each class, as illustrated in Figure 3-(a). In contrast, the probability distribution over the prototypes considered in Prob-PSENNs induces a probability distribution over each value of δ_x, $p(\boldsymbol{\delta}_{x,i}) = p\big(\phi(e_x, \boldsymbol{r}_i)\big)$, as exemplified in Figure 3-(b) and Figure 3-(c). This allows us to compute not only more robust and representative information for each value of δ_x, but also to compute the uncertainty in the explanation. Furthermore, this enables a direct and explicit connection between the explanation uncertainty and the output uncertainty, being therefore a key advantage of Prob-PSENNs.

Types of explanation uncertainty Similarly to the predictive uncertainty, different types of uncertainty can be attributed to the explanations of Prob-PSENNs. In the context of explanations, we will denominate *aleatoric* uncertainty to the uncertainty in determining which class-prototypes best represent the input at hand, that is, in determining which prototypes are closest to the encoded input, possibly due to ambiguity in the representation of the input. Therefore, high aleatoric uncertainty will imply highly overlapped distance-distributions $p(\boldsymbol{\delta}_{x,i})$. In contrast, *epistemic* explanatory uncertainty will be high for those inputs that lie far from the training distribution, and, consequently, also far from the distribution over the prototypes, resulting in uninformative and non-representative explanations for that input. Therefore, high epistemic uncertainty will imply large distances with respect to all the prototypes. An illustrative representation of these types of explanatory uncertainty is provided in Figure 2 (right).

Quantifying explanation uncertainty Given that, in the general case, computing $p(\boldsymbol{\delta}_{x,i})$ will be intractable both analytically and numerically, we will rely on sample-based estimators in order to compute meaningful information about those distributions. To begin with, being $\delta_x^{(1)}, \ldots, \delta_x^{(N)}$ the distance-vectors obtained for N inferences of the model (i.e., for N random sets of prototypes), the sample mean $\bar{\delta}_{x,i} = \frac{1}{N}\sum_{n=1}^{N} \delta_{x,i}^{(n)}$ can inform us about the average similarity between the input and the prototypes of the class y_i, while the sample variance $\mathrm{Var}\,\big(p(\boldsymbol{\delta}_{x,i})\big) \approx \frac{1}{N-1}\sum_{n=1}^{N}\big(\delta_{x,i}^{(n)} - \bar{\delta}_{x,i}\big)^2$ can be considered a measure of uncertainty about those similarities.

While the aforementioned two metrics describe only each distribution $p(\boldsymbol{\delta}_{x,i})$ separately, considering all these distributions jointly allows us to estimate aleatoric and epistemic uncertainty in the explanations. As mentioned before, aleatoric uncertainty can be determined based on the overlap between the probability density functions describing $p(\boldsymbol{\delta}_{x,i})$, which will be denoted as $f_{\boldsymbol{\delta}_{x,i}}(\cdot)$. For a binary case, the overlap can be measured by means of (normalized) overlapping indices [32]: $\int_{\mathbb{R}} \min[f_{\boldsymbol{\delta}_{x,1}}(z),\ f_{\boldsymbol{\delta}_{x,2}}(z)]\,dz$. In order to address multi-class problems, we will generalize the metric as the maximum overlap between the density function $f_{\boldsymbol{\delta}_{x,i^*}}(\cdot)$ corresponding to the most-likely class

y_{i^*} and the functions corresponding to the remaining classes:

$$\mathscr{U}_A(x) = \max_{\substack{j=1,\dots,c\\ j\neq i^*}} \left[\int_{\mathbb{R}} \min[f_{\boldsymbol{\delta}_{x,i^*}}(z),\ f_{\boldsymbol{\delta}_{x,j}}(z)]\, dz \right]. \tag{3}$$

In practice, the density functions can be estimated by means of kernel-density estimation methods, based on the distances obtained for a finite number of inferences N.

Regarding epistemic explanatory uncertainty, as explained above, a high uncertainty will be achieved for those inputs for which the distances to all the prototypes are large. On that account, the minimum sample mean: $\bar{\delta}_{x,i*} = \min_{1\leq j\leq c} \bar{\delta}_{x,j}$ represents a suitable base metric for evaluating this uncertainty. Furthermore, in order to determine the severity of the uncertainty in a normalized and scale-independent manner, a quantile-based metric will be defined, close to the approach proposed in [33]. This metric will be based on comparing the value at hand with the values obtained for the data distribution $\mathscr{D}$, which can be estimated, in practice, using a training set D:

$$\mathscr{U}_E(x) = P_{\substack{(x',y)\sim\mathscr{D}\\ y=y_{i*}}}\big(\bar{\delta}_{x',i*} \leq \bar{\delta}_{x,i*}\big). \tag{4}$$

3.4 Creating more meaningful explanations

In practice, we expect to sample N different explanations, and, thereby, N different yet representative prototypes for each class $y_i \in Y$ (assuming $m = c$). Although presenting all these prototypes as part of the explanation might be impractical, different strategies can be considered to summarize or distill the results, for the sake of more meaningful and useful explanations. First, among the prototypes corresponding to the most likely class, the ones closest to the input at hand can be considered the best candidates to justify the prediction of the model. Furthermore, the closest prototypes corresponding to the remaining classes can be taken as *counterarguments* against the most likely class, which can be used, for instance, to identify and explain why an input can be considered ambiguous or challenging to classify. On the other hand, the farther prototypes corresponding to the most likely class can be used to exemplify alternative yet still valid representations for that class, which can provide useful complementary information, thus being valuable for *knowledge-acquisition* purposes.

4 Training procedure

Let $p_\lambda(\boldsymbol{R})$ represent a parametric probability distribution over the prototypes. The goal of the training procedure will be to optimize the parameters λ, jointly with the rest of parameters of the model, in order to maximize both the predictive performance and interpretability of the model. For simplicity, the prototypes $\boldsymbol{r}_1 \dots, \boldsymbol{r}_c$ will be assumed to be independent random variables, with each $\boldsymbol{r}_i$ distributed according to a distribution $p_{\lambda_i}(\boldsymbol{r}_i)$, $1 \leq i \leq c$, and, hence, $p_\lambda(\boldsymbol{R}) = p_\lambda(\boldsymbol{r}_1, \dots, \boldsymbol{r}_c) = \prod_{i=1}^{c} p_{\lambda_i}(\boldsymbol{r}_i)$. The following generalized loss function will be employed to train the entire architecture:

$$\mathscr{L} = \tau_1 \cdot \mathscr{L}_{NLL} + \tau_2 \cdot \mathscr{L}_{REC} + \tau_3 \cdot \mathscr{L}_{INT}$$

where,

$$\mathscr{L}_{\text{NLL}} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{(x,y)\in D}\sum_{i=1}^{c} \mathbb{1}(y=y_i)\cdot \log\big(f_i^{R_n}(x)\big), \quad R_n = [r_i]_{i=1}^{c} \text{ s.t. } r_i \sim p_{\lambda_i}(\boldsymbol{r}_i),\ \ 1\leq n\leq N,$$

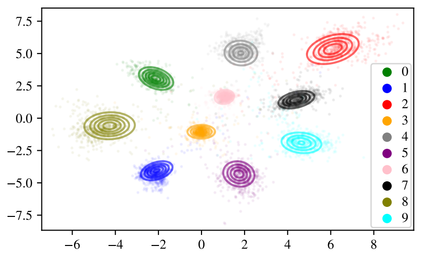
$$\mathscr{L}_{\text{REC}} = \sum_{(x,\cdot)\in D} ||x - g\big(e(x)\big)||_2^2, \qquad \mathscr{L}_{\text{INT}} = -\sum_{(x,y)\in D}\sum_{i=1}^{c} \mathbb{1}(y = y_i)\cdot \log p_{\lambda_i}\big(e(x)\big),$$

and $\tau_1, \tau_2, \tau_3 \in \mathbb{R}^+$ weight the contribution of each term. Notice that $\mathscr{L}_{\text{NLL}}$ represents the classification (cross-entropy) loss, averaged for N samples of prototypes, and $\mathscr{L}_{\text{REC}}$ the reconstruction loss. Finally, $\mathscr{L}_{\text{INT}}$ can be seen as an interpretability loss, as it will encourage the distribution over the prototypes to be similar to the distribution of the encoded data, which is a crucial requirement for interpretability [28]. Based on this methodology, the entire architecture can be jointly optimized by means of conventional gradient descent approaches.

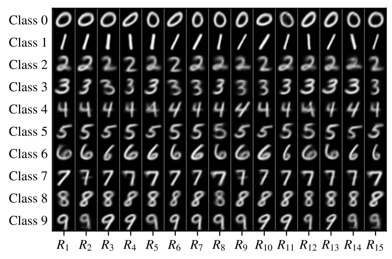
5 Experiments

For illustration purposes, the effectiveness and capabilities of the proposed model will be assessed on the MNIST dataset [27], while additional experiments on the Fashion-MNIST [45] and K-MNIST [9] datasets are reported in Appendix D and Appendix E, respectively. In all the experiments, the Euclidean distance will be used as the distance metric ϕ, and a Gaussian distribution will be assumed for each $p_{\lambda_i}(\boldsymbol{r}_i)$, $1 \leq i \leq c$. The full implementation details are reported in Appendix B.

Learned latent spaces and distributions over the prototypes For illustration, the results obtained for the case in which $l = 2$ (i.e., a two-dimensional latent space is considered), are shown in Figure 4. The left figure shows the learned latent space, visualizing the latent representations assigned to a random set of test data (dots), as well as the learned distribution over the prototypes $p_{\lambda_i}(\boldsymbol{r}_i)$, $1 \leq i \leq c$ (contour lines). The right figure represents 15 random sets of prototypes R sampled from the learned distributions, and decoded by the decoder g. As can be seen, even for a very constrained (low-dimensional) latent space, the model is capable of capturing and decoding prototypical and varied representations for each class. The model achieves a top-1 accuracy of 0.98 and 0.96 in the train and test set, respectively, computed for $N = 30$ inferences. Further results for $l = 5$ and $l = 10$ are shown in Appendix C, where it can be observed that the increase in the dimensionality of the latent space makes it possible to capture prototype distributions with a higher degree of complexity in their structure, as well as to obtain more detailed decodings. Further results for the Fashion-MNIST and K-MNIST datasets are provided in Appendix D and Appendix E, respectively. In addition, to show that our approach can scale to a larger number of classes, Appendix F includes results on the E-MNIST dataset [10], an extension of MNIST which contains handwritten representations of numbers and letters and is composed of 47 classes.



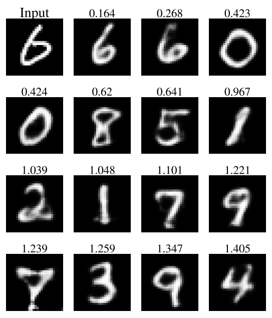
(a) Latent space.


Class 0
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
Class 8
Class 9
R_1 R_2 R_3 R_4 R_5 R_6 R_7 R_8 R_9 R_{10} R_{11} R_{12} R_{13} R_{14} R_{15}

(b) Randomly sampled sets of prototypes.

Figure 4: Results obtained with a Prob-PSENN with $l = 2$ for the MNIST dataset. (a) Learned latent space, where the contours represent the prototype distributions $p_{\lambda_i}(\boldsymbol{r}_i)$ and the dots represent encoded input samples. (b) 15 random sets of prototypes $R_n = (r_1, \ldots, r_c)$ s.t. $r_i \sim p_{\lambda_j}(\boldsymbol{r}_j), 1 \leq i \leq c$, $1 \leq n \leq 15$.

Comparison between conventional PSENN and Prob-PSENN First, to thoroughly compare the predictive performance of both approaches, the accuracies obtained for the three datasets considered are compared in Table 1. For the sake of completeness, the performance of PSENNs is computed for different number of prototypes, while preserving its original architecture, consisting of a Convolutional Autoencoder, a latent space of $l = 40$ dimensions, and using elastic deformation as data augmentation. The performance of Prob-PSENNs has been evaluated for the following configurations, fully described in Appendix B: using a Multi-Layer Perceptron Autoencoder (MLP), a Convolutional Autoencoder (CNN), and including elastic deformation [38] as data augmentation (AUG). As can be seen, Prob-PSENNs achieve a very competitive performance in all the datasets, with an accuracy gain of 2% on the K-MNIST dataset and a loss of less than 1% on MNIST and F-MNIST, while having as benefits a greater robustness, trustworthiness and enhanced explanatory capabilities, as we showcase in Figure 5. In particular, this figure compares, for the MNIST task, the explanation computed by a conventional PSENN ($m = 15$) with the one computed by Prob-PSENN ($l = 5$, MLP). More comparisons will be provided in the supplementary material. As can be seen, the prototypes sampled by Prob-PSENN exhibit considerably more diversity than the fixed prototypes learned by a conventional PSENN, which enables to find more tailored prototypes for the input at hand, as well as to provide alternative forms that the class might take. In addition, while PSENNs only return point distances to the prototypes, Prob-PSENNs provide a distribution over those distances, enabling to assess the uncertainty of the prediction, and, hence, representing a more robust and trustworthy approach.



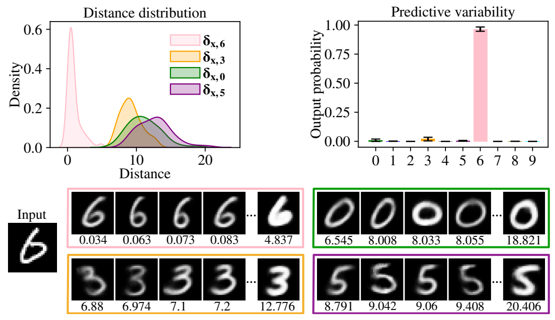


Figure 5: Comparison between the explanation provided by a PSENN with fixed prototypes (left) and a Prob-PSENN (right). For simplicity, the four most likely classes are visualized for Prob-PSENN, considering, for each class, the four closest prototypes and the most distant ones.

Table 1: Predictive accuracies (mean and standard deviation for 10 runs) obtained by Prob-PSENN and PSENN for the MNIST, Fashion-MNIST and K-MNIST datasets. The results are shown in percentages.

Configuration	MNIST	F-MNIST	K-MNIST
PSENN ($l = 40$, $m = 10$)	98.97 ± 0.16	90.15 ± 0.33	91.11 ± 0.47
PSENN ($l = 40$, $m = 15$)	98.99 ± 0.11	90.10 ± 0.61	90.93 ± 0.41
PSENN ($l = 40$, $m = 20$)	99.01 ± 0.12	90.22 ± 0.50	91.15 ± 0.53
PSENN ($l = 40$, $m = 30$)	99.05 ± 0.15	89.90 ± 0.33	91.04 ± 0.96
PSENN ($l = 40$, $m = 40$)	98.93 ± 0.17	90.17 ± 0.39	90.81 ± 0.96
PSENN ($l = 40$, $m = 50$)	99.02 ± 0.11	89.76 ± 0.44	91.17 ± 0.74
Prob-PSENN ($l = 5$, MLP)	97.55 ± 0.29	88.83 ± 0.25	88.69 ± 1.23
Prob-PSENN ($l = 10$, MLP)	97.59 ± 0.48	88.77 ± 0.50	88.92 ± 1.00
Prob-PSENN ($l = 10$, CNN)	98.28 ± 0.17	89.47 ± 0.34	89.06 ± 0.34
Prob-PSENN ($l = 10$, MLP + AUG)	98.53 ± 0.08	87.78 ± 0.50	93.17 ± 0.33
Prob-PSENN ($l = 10$, CNN + AUG)	98.75 ± 0.17	87.98 ± 0.46	89.40 ± 1.14

Evaluating the explanatory uncertainties Figure 6 compares, for a Prob-PSENN with $l = 2$, the areas of the latent space with high explanatory uncertainty. The aleatoric and epistemic explanatory uncertainties (second and third columns) have been evaluated using the metrics $\mathscr{U}_A(x)$ and $\mathscr{U}_E(x)$ defined in Equations (3) and (4), respectively. The fourth column shows the areas with high epistemic uncertainty exclusively (i.e., with no aleatoric uncertainty), which has been computed as $\mathscr{U}_E(x) - \mathscr{U}_A(x)$. As can be seen from the results, the regions with high aleatoric uncertainty are those that are at a similar distance from the two regions that the model associates to each class. On the other hand, the areas with high epistemic uncertainty are those far from the regions to which the prototype distributions assign a high density, and, therefore, the explanations are no longer representative of those regions.

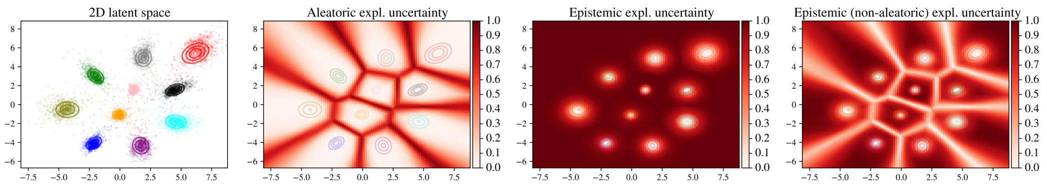


Figure 6: Evaluating the explanatory uncertainty in different regions of the learned latent space.

Epistemic uncertainty Figure 7 compares the epistemic uncertainty achieved by a Prob-PSENN, trained for the MNIST dataset, when classifying 1000 test samples of the MNIST and Fashion-MNIST datasets. As can be seen, when inputs from the Fashion-MNIST dataset are presented to the model, a very high explanatory epistemic is captured for almost all the inputs, while for the MNIST test data the uncertainty is uniformly distributed. For reference, Figure 7 also shows, for each dataset, the samples that achieve the lowest and highest uncertainty. As can be seen, even for the MNIST dataset, highly unusual inputs that could be considered out-of-distribution instances can be found, which are effectively detected by the uncertainty estimation capabilities of Prob-PSENN.

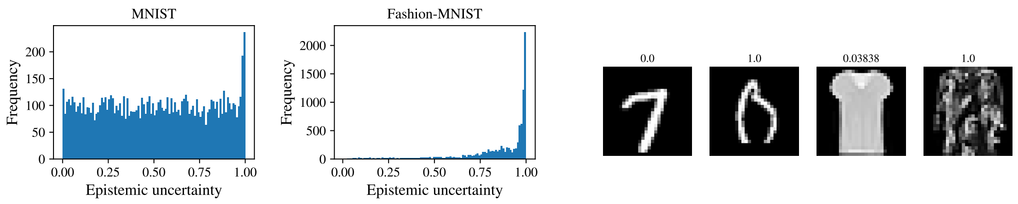


Figure 7: Histograms showing the explanatory epistemic uncertainty obtained by a Prob-PSENN for 1000 test samples from MNIST (1st figure) and Fashion-MNIST (2nd figure). Notice that the scales of the Y-axis differ in the two plots. The following four images display those inputs that achieve, for each dataset, the lowest and highest epistemic uncertainty.

Aleatoric uncertainty Finally, the prediction and explanation for a particular input with high aleatoric uncertainty is shown in Figure 8. The results are shown for a Prob-PSENN with $l = 5$ and $N = 50$ inferences. The top row includes the density function over the distances corresponding to the two most likely classes, $\boldsymbol{\delta}_{x,4}$ and $\boldsymbol{\delta}_{x,9}$, estimated by Gaussian-kernel density estimation, as well as a box plot showing the variability in the output probabilities assigned to each class, considering all the inferences. As it can be assessed, the distributions of both the distances and the output probabilities are highly overlapped, evidencing a high explanatory and predictive aleatoric uncertainty. The second row of the figure includes the input and a subset of the sampled prototypes, sorted by distance to the input in increasing order. As can be seen, both distributions over the prototypes capture and sample prototypical representations that could be taken as equally descriptive of the input at hand, thus explaining its ambiguity.

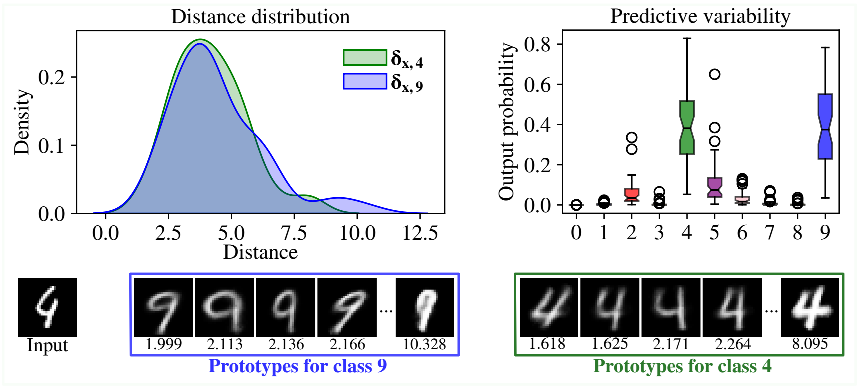


Figure 8: Prediction and explanation results obtained for a test input with high aleatoric uncertainty. The prototypes are sorted by distance to the input, in increasing order.

Harnessing uncertainty to improve predictive accuracy and reliability We experimentally validated that taking model uncertainty into consideration can provide substantial improvements in accuracy. Specifically, by discarding inputs with high explanatory epistemic uncertainty (e.g., greater than 95% or 99%), the accuracy of Prob-PSENN increases in all the datasets and configurations, as we show in Table 2 (Appendix G). To complement these results, Table 3 shows the percentage of inputs that are discarded in each case. At the same time, a large number of the discarded inputs are anomalous or challenging to classify, which would justify discarding them for the sake of higher reliability and robustness. As evidence, we show in Table 4 that the error rate of the deterministic baseline model is also higher in the inputs which receive high uncertainty than in the rest. Thus, this evaluation shows how capturing model uncertainty provides greater accuracy, robustness and safety. We justify the choice of the mentioned uncertainty metric, based on the fact that it is reasonable to consider a prediction unreliable if all the prototypes have low similarity to the input, given the distance-based classification pipeline of the model.

6 Conclusions

In this paper we introduced Prob-PSENN, a novel Prototype-Based Self-Explainable Neural Network paradigm relying on a probabilistic generalization of the prototypes, which enhances the capabilities of the model in two key aspects. First, unlike previous approaches, our model is not restricted to a fixed number or set of prototypes. This enables Prob-PSENN to learn much more diverse prototypes, and to consider all of them when classifying an input, which represents a more flexible and robust approach. Second, our probabilistic approach equips Prob-PSENNs with tools to capture and harness the uncertainty in both the prediction and the explanation of the network. This novel feature makes it possible to detect when the model is providing uncertain outcomes, substantially increasing the reliability of the model. We experimentally demonstrate how Prob-PSENNs achieve more informative, robust and trustworthy explanations than their non-probabilistic counterparts, while keeping a competitive predictive accuracy.

7 Limitations and future work

As future work, inspired by recent advances in Bayesian Neural Networks [13, 16, 31], we plan to develop a Bayesian formulation for the distribution over the prototypes, which might enable a more theoretically-grounded approach for uncertainty quantification. Furthermore, we also believe that a fully-probabilistic model (i.e., placing a probability distribution for the rest of the network parameters as well) could enhance the quantification of the uncertainty in Prob-PSENNs, or even induce new notions of uncertainty, such as the *encoding* or *class-representation uncertainty*. Apart from that, while Prob-PSENNs have demonstrated high predictive and explanatory performance, we found it challenging to model probability distributions over the prototypes in high-dimensional spaces, and, therefore, resort to relatively low dimensional latent spaces in our experiments. We plan to address this limitation in the future, in order to scale the proposed approach to higher-dimensional datasets.

Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 834115), the Spanish Ministry of Science, Innovation and Universities (project PID2022-137442NB-I00, grant FPU19/03231, and BCAM Severo Ochoa accreditation CEX2021-001142-S), and the Basque Government (projects KK-2023/00012 and IT1504-22).

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv preprint arXiv:1603.04467*, 2015. doi: 10.48550/arXiv.1603.04467.

[2] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim. Sanity Checks for Saliency Maps. In *Advances in Neural Information Processing Systems*, volume 31, pages 9505–9515, 2018.

[3] U. Aivodji, H. Arai, O. Fortineau, S. Gambs, S. Hara, and A. Tapp. Fairwashing: The Risk of Rationalization. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pages 161–170, 2019.

[4] D. Alvarez-Melis and T. Jaakkola. Towards Robust Interpretability with Self-Explaining Neural Networks. In *Advances in Neural Information Processing Systems*, volume 31, pages 7775–7784, 2018.

[5] M. Ashoori and J. D. Weisz. In AI We Trust? Factors That Influence Trustworthiness of AI-infused Decision-Making Processes. *arXiv preprint arXiv:1912.02675*, 2019. doi: 10.48550/arXiv.1912.02675.

[6] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLOS ONE*, 10(7):e0130140, 2015. ISSN 1932-6203. doi: 10.1371/journal.pone.0130140.

[7] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli. Evasion Attacks against Machine Learning at Test Time. In *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, pages 387–402, 2013. ISBN 978-3-642-40994-3. doi: 10.1007/978-3-642-40994-3_25.

[8] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. K. Su. This Looks Like That: Deep Learning for Interpretable Image Recognition. In *Advances in Neural Information Processing Systems*, volume 32, pages 8930–8941, 2019.

[9] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha. Deep Learning for Classical Japanese Literature. In *Neural Information Processing Systems 2018 Workshop on Machine Learning for Creativity and Design*, 2018.

[10] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik. EMNIST: Extending MNIST to handwritten letters. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926, 2017. doi: 10.1109/IJCNN.2017.7966217.

[11] S. Depeweg, J.-M. Hernandez-Lobato, F. Doshi-Velez, and S. Udluft. Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-sensitive Learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pages 1184–1193, 2018.

[12] J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, and R. A. Saurous. TensorFlow Distributions. *arXiv preprint arXiv:1711.10604*, 2017. doi: 10.48550/arXiv.1711.10604.

[13] Y. Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[14] S. Gautam, A. Boubekki, S. Hansen, S. Salahuddin, R. Jenssen, M. Höhne, and M. Kampffmeyer. ProtoVAE: A Trustworthy Self-Explainable Prototypical Variational Model. In *Advances in Neural Information Processing Systems*, volume 35, pages 17940–17952, 2022.

[15] A. Ghorbani, A. Abid, and J. Zou. Interpretation of Neural Networks Is Fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3681–3688, 2019. doi: 10.1609/aaai.v33i01.33013681.

[16] E. Goan and C. Fookes. Bayesian Neural Networks: An Introduction and Survey. *Case Studies in Applied Bayesian Data Science*, 2259:45–87, 2020. doi: 10.1007/978-3-030-42553-1_3.

[17] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On Calibration of Modern Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pages 1321–1330, 2017.

[18] P. Hase, C. Chen, O. Li, and C. Rudin. Interpretable Image Recognition with Hierarchical Prototypes. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 7, pages 32–40, 2019. ISBN 978-1-57735-820-6.

[19] N. Houlsby, F. Huszár, Z. Ghahramani, and M. Lengyel. Bayesian Active Learning for Classification and Preference Learning. *arXiv preprint arXiv:1112.5745*, 2011. doi: 10.48550/arXiv.1112.5745.

[20] E. Hüllermeier and W. Waegeman. Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods. *Machine Learning*, 110(3):457–506, 2021. ISSN 1573-0565. doi: 10.1007/s10994-021-05946-3.

[21] H. Joo, J. Kim, H. Han, and J. Lee. Distributional Prototypical Methods for Reliable Explanation Space Construction. *IEEE Access*, 11:34821–34834, 2023. ISSN 2169-3536. doi: 10.1109/ACCESS.2023.3264794.

[22] A. Kendall and Y. Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In *Advances in Neural Information Processing Systems*, volume 30, pages 5574–5584, 2017.

[23] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, 2015. doi: 10.48550/arXiv.1412.6980.

[24] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[25] Y. Kwon, J.-H. Won, B. J. Kim, and M. C. Paik. Uncertainty Quantification Using Bayesian Neural Networks in Classification: Application to Ischemic Stroke Lesion Segmentation. In *Medical Imaging with Deep Learning (MIDL)*, 2022.

[26] H. Lakkaraju and O. Bastani. "How do I fool you?": Manipulating User Trust via Misleading Black Box Explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '20, pages 79–85, 2020. ISBN 978-1-4503-7110-0. doi: 10.1145/3375627.3375833.

[27] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. ISSN 1558-2256. doi: 10.1109/5.726791.

[28] O. Li, H. Liu, C. Chen, and C. Rudin. Deep Learning for Case-Based Reasoning Through Prototypes: A Neural Network That Explains Its Predictions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pages 3530–3537, 2018.

[29] Z. C. Lipton. The Mythos of Model Interpretability: In Machine Learning, the Concept of Interpretability Is Both Important and Slippery. *Queue*, 16(3):31–57, 2018. ISSN 1542-7730. doi: 10.1145/3236386.3241340.

[30] M. Nauta, R. Van Bree, and C. Seifert. Neural Prototype Trees for Interpretable Fine-grained Image Recognition. In *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14933–14943, 2021. ISBN 978-1-66544-509-2. doi: 10.1109/CVPR46437.2021.01469.

[31] R. M. Neal. *Bayesian Learning for Neural Networks*, volume 118. Springer Science & Business Media, 2012.

[32] M. Pastore and A. Calcagnì. Measuring Distribution Similarities Between Samples: A Distribution-Free Overlapping Index. *Frontiers in Psychology*, 10(1089), 2019. ISSN 1664-1078. doi: 10.3389/fpsyg.2019.01089.

[33] A. Petrov and M. Kwiatkowska. Robustness of Unsupervised Representation Learning without Labels. *arXiv preprint arXiv:2210.04076*, 2022. doi: 10.48550/arXiv.2210.04076.

[34] C. Rudin. Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. *Nature Machine Intelligence*, 1(5):206–215, 2019. ISSN 2522-5839. doi: 10.1038/s42256-019-0048-x.

[35] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017. doi: 10.1109/ICCV.2017.74.

[36] C. E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. ISSN 0005-8580. doi: 10.1002/j.1538-7305.1948.tb01338.x.

[37] A. Shrikumar, P. Greenside, and A. Kundaje. Learning Important Features Through Propagating Activation Differences. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pages 3145–3153, 2017.

[38] P. Simard, D. Steinkraus, and J. Platt. Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, pages 958–963, 2003. doi: 10.1109/ICDAR.2003.1227801.

[39] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *Workshop of the 2014 International Conference on Learning Representations (ICLR)*, 2014.

[40] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for Simplicity: The All Convolutional Net. In *Workshop of the 2015 International Conference on Learning Representations (ICLR)*, 2015. doi: 10.48550/arXiv.1412.6806.

[41] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic Attribution for Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pages 3319–3328, 2017.

[42] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[43] J. Vadillo, R. Santana, and J. A. Lozano. When and How to Fool Explainable Models (and Humans) with Adversarial Examples. *arXiv preprint arXiv:2107.01943*, 2021.

[44] C. Wang, Y. Chen, F. Liu, D. J. McCarthy, H. Frazer, and G. Carneiro. Mixture of Gaussian-distributed Prototypes with Generative Modelling for Interpretable Image Classification. *arXiv preprint arXiv:2312.00092*, 2023. doi: 10.48550/arXiv.2312.00092.

[45] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:2107.01943*, 2017. doi: 10.48550/arXiv.1708.07747.

[46] X. Yuan, P. He, Q. Zhu, and X. Li. Adversarial Examples: Attacks and Defenses for Deep Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9):2805–2824, 2019. ISSN 2162-2388. doi: 10.1109/TNNLS.2018.2886017.

[47] Y. Zhang, P. Tiňo, A. Leonardis, and K. Tang. A Survey on Neural Network Interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5):726–742, 2021. ISSN 2471-285X. doi: 10.1109/TETCI.2021.3100641.

Appendix A Discussion on previous PSENN approaches

As introduced in Section 1, the development of PSENNs arises as an alternative to black-box DNNs and posthoc explanations, with the goal of ensuring a human-understandable decision process by construction, while keeping a competitive predictive performance. This is achieved by integrating a transparent case-based reasoning as the core element, so that the prediction of the model is determined based on the similarity between the input and a set of prototypical representations of the output classes (see Section 2 for further details).

The way in which such prototypes are defined has led to different types of PSENNs in the literature. In the seminal work of Li et al. [28], mirrored by recent follow-up works [14, 21], the prototypes represent complete characterizations of each class or concept (e.g., full representations of the digits in handwritten digit recognition). In contrast, other follow-up works aim to capture more abstract [4] or specific features (e.g., prototypical parts or patches) [8, 18, 30] as prototypes. However, an important drawback of these approaches is that the prototypes are restricted to being samples or sample parts of (encoded) training inputs [4, 8, 18, 30]. This implies a flexibility loss regarding the end-to-end optimization of the prototypes, as well as a transparency loss when *surrogate prototypes* (e.g., nearest training images) are selected to visualize the explanations [4, 14]. For these reasons, we followed the same spirit as in [28], and thus employed the corresponding architecture as a baseline, in order to guarantee a fully transparent and end-to-end trainable architecture. We also remark that, while [14, 21] also focus on learning a fixed set of complete, unconstrained, and decodable prototypes, the proposed architecture relies on a Variational Autoencoder [24] as the autoencoding module. In contrast, as in [28], our approach does not require enforcing such a regularized latent space, hence representing a more general approach.

Closer to our approach, concurrent works [21, 44] leverage the use of distributions over the prototypes to improve the consistency between image similarity and latent space location [21] and to increase the representation power of the model [44]. Contrary to our work, both approaches employ only the means of the learned distributions as prototypes, whereas our sampling-based inference process allows us to sample multiple sets of prototypes, avoiding the constraint of committing to a fixed set, and harnessing all of them for both the explanation and classification. Furthermore, the approach in [44] follows the strategy of [4, 8, 18] and rely on surrogate prototypes, replacing the distribution means with nearest training instances, with the corresponding flexibility and transparency loss discussed above. Another key difference between our approach and that in [21] is that, in their work, the similarities between the input's distributional embedding and the prototype distributions, which fed the final classification layer, is measured by considering the entire distributions, using distributional similarity metrics. However, as stated in [21], such similarities do not form nor rely on observable prototypes, which is contrary to the purpose of explaining and justifying the classification. Instead, we advocate for a more transparent approach, in which the model predictions are solely derived from the distances to the sampled sets of prototypes, ensuring a complete and transparent justification of model outputs.

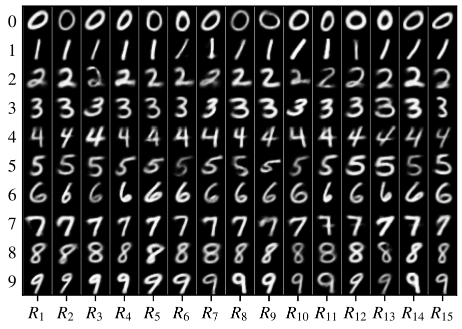
Appendix B Implementation details

Our models have been implemented in Python, using TensorFlow [1] and TensorFlow Probability [12] packages. The experiments have been conducted on a cluster with Nvidia RTX A5000 GPUs and AMD EPYC 7252 8-Core CPUs. However, we point out that our models are amenable to being trained on commodity hardware. The architecture of the MLP Autoencoder module is composed of two dense layers with 1000 hidden units and ReLU activation, followed by a dense layer with l output neurons. A linear activation is used for the last layer of the encoder. The architecture of the CNN Autoencoder is composed of four convolutional layers, followed by a dense layer with l output neurons and linear activation for the last layer of the encoder, in order to control the dimensionality of the latent space. The convolutional layers employ a 3×3 kernel size, a stride value of 2, same zero padding. We use 32 channels for the first three layers and 10 channels for the fourth layer.

The experiments have been run for $\tau_1 = 1.0, \tau_2 = 10.0, \tau_3 = 0.05$, a batch size of 128 and 30 epochs. The Adam optimization method [23] has been used to train the networks, using a learning rate of 0.001. The final classification layer W was set as $W = -I$, where I represents the identity matrix of dimension c, thus ensuring that the association between the prototype $\boldsymbol{r}_i$ and the class y_i, $1 \leq i \leq c$, is explicit and unequivocal. For all the datasets, the standard train and test splits have been used, with sizes 60000 and 10000, respectively, and a subset of 6000 training samples has been extracted as a validation set. The results on the baseline model [28] have been reproduced using the implementation released by the authors.

Appendix C Complementary results on the MNIST dataset

In order to complement the results reported in Section 5, Figure 9 shows, for the MNIST dataset, 15 randomly sampled sets of prototypes from a Prob-PSENN with $l = 5$ (left) and $l = 10$ (right). As it can be observed, the increase in the dimensionality of the latent space makes it possible to capture prototype distributions with a higher degree of complexity in their structure, as well as to obtain more detailed decodings.


0
1
2
3
4
5
6
7
8
9
R_1 R_2 R_3 R_4 R_5 R_6 R_7 R_8 R_9 R_{10} R_{11} R_{12} R_{13} R_{14} R_{15}

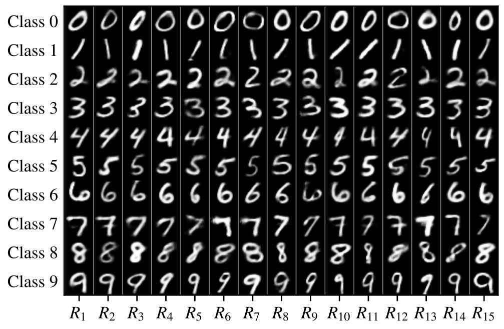
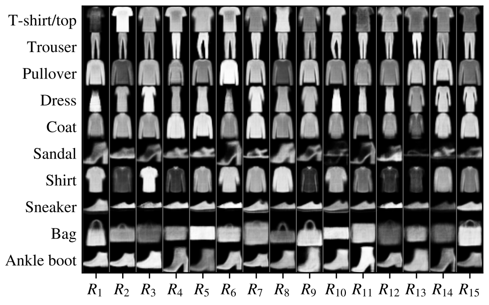

Class 0
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
Class 8
Class 9
R_1 R_2 R_3 R_4 R_5 R_6 R_7 R_8 R_9 R_{10} R_{11} R_{12} R_{13} R_{14} R_{15}

Figure 9: 15 random sets of prototypes, obtained with a Prob-PSENN with $l = 5$ (left) and $l = 10$ (right).

Appendix D Results on the Fashion-MNIST dataset

This section includes complementary results on the Fashion-MNIST dataset. First, Figure 10 shows randomly sampled prototypes for a Prob-PSENN with $l = 5$ (left) and $l = 10$ (right). These figures demonstrate that the probabilistic redefinition of the prototypes introduced in Prob-PSENNs enables the model to learn, even for relatively low-dimensional latent spaces, diverse but still prototypical representations of each class.


T-shirt/top
Trouser
Pullover
Dress
Coat
Sandal
Shirt
Sneaker
Bag
Ankle boot
R_1 R_2 R_3 R_4 R_5 R_6 R_7 R_8 R_9 R_{10} R_{11} R_{12} R_{13} R_{14} R_{15}

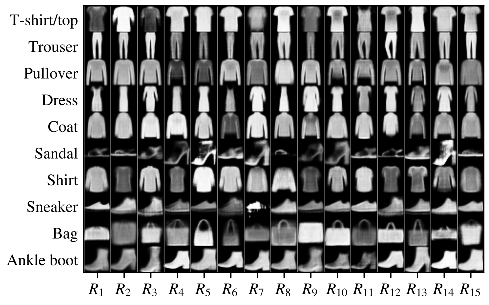

T-shirt/top
Trouser
Pullover
Dress
Coat
Sandal
Shirt
Sneaker
Bag
Ankle boot
R_1 R_2 R_3 R_4 R_5 R_6 R_7 R_8 R_9 R_{10} R_{11} R_{12} R_{13} R_{14} R_{15}

Figure 10: Learned prototypes for the Fashion-MNIST dataset, with $l = 5$ (left) and $l = 10$ (right).

Furthermore, a visual comparison between the explanations provided by a non-probabilistic PSENN and a Prob-PSENN is provided in Figures 11 and 12. For the sake of completeness, the former figure includes the explanations provided by a PSENN with $m = 15$ (left) and $m = 50$ (right) prototypes. In both cases, the original input is shown in the top-left corner, and the prototypes are colored according to the classes returned by the model when the prototypes are classified, in order to make it clearer which class each prototype is most associated with. Notice that this connection is explicit and unequivocal by construction in our Prob-PSENNs, being the class y_i represented by the prototype r_i, $1 \leq i \leq c$, resulting in more transparent decision processes and easier-to-interpret explanations. As can be observed in Figures 11 and 12, the prototypes learned by the PSENNs are, even for $m = 50$ prototypes, considerably less meaningful and varied than the ones captured by Prob-PSENN. Furthermore, the uncertainty metrics introduced in Section 3.3 enable us to quantify a low aleatoric and epistemic explanatory uncertainty for this input (0.004 and 0.16, respectively), which increases the trustworthiness and reliability of the outcomes of Prob-PSENN.

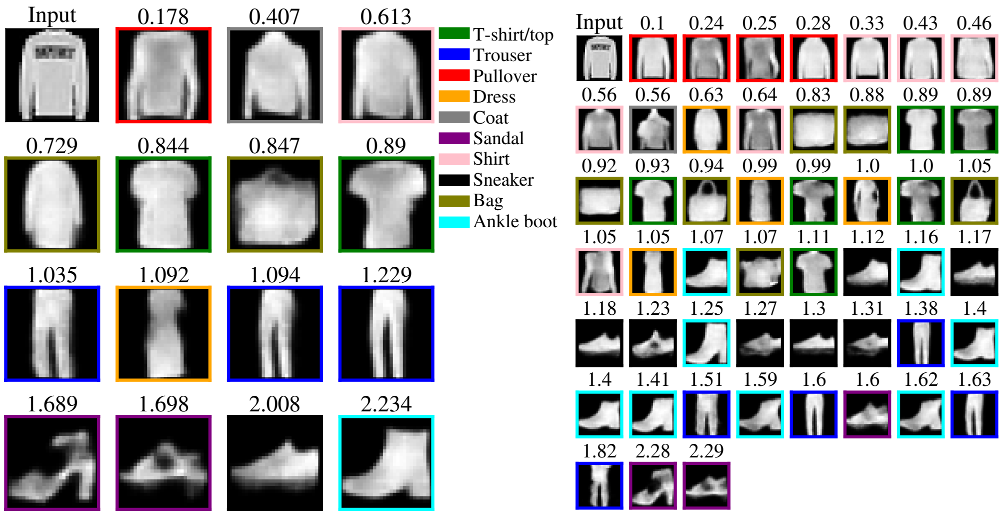


Figure 11: Explanations provided by a PSENN with $m = 15$ (left) and $m = 50$ (right) for the Fashion-MNIST dataset. Both models classify the input (shown in the top-left corner of each grid) as the class *pullover* with a probability of 0.99. Note that the prototypes are sorted based on their distances to the input (displayed on top of each figure), in increasing order.

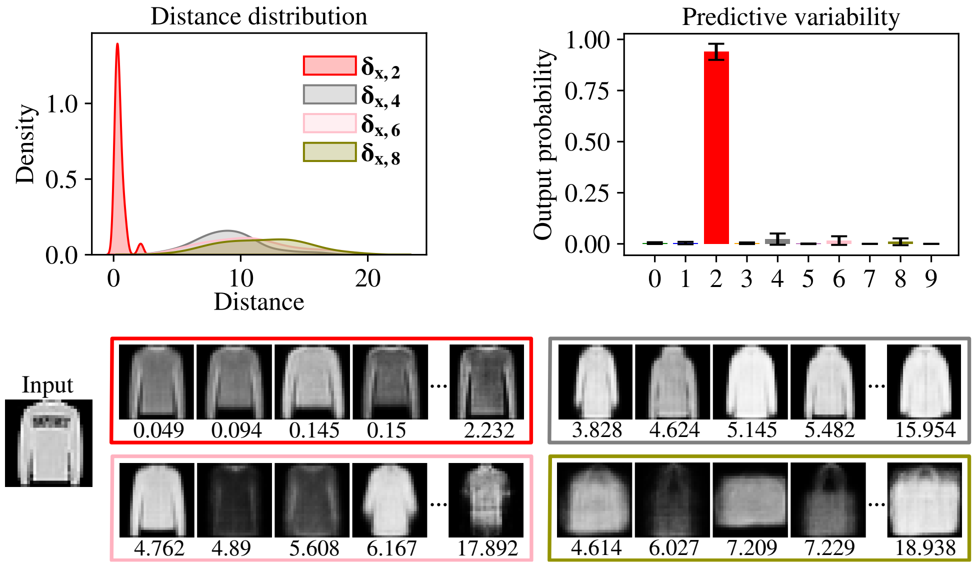


Figure 12: Explanation provided by a Prob-PSENN, for the Fashion-MNIST dataset. The top row shows the distance distribution to the four closest classes (left), and the variability in the output probabilities (right). The bottom part shows the prototypes corresponding to the four closest classes, and the corresponding distances to the input. Note that the prototypes are ordered based on their distance, in increasing order.

Appendix E Results on the K-MNIST dataset

This section contains additional results on the 10-class K-MNIST dataset [9], which includes handwritten representations of Japanese letters. Random images from this dataset are shown in Figure 13, from which it can be noticed a high variance in the representation of each letter. Randomly sampled prototypes from Prob-PSENNs with $l = 5$ and $l = 10$ are displayed in Figure 14. As can be assessed in the figure, despite the complexity and high variability of this dataset, Prob-PSENNs are capable of capturing the most frequent representations of each class, ensuring a sufficiently diverse and meaningful characterization of the classes.

As for the previous datasets, a comparison between the explanations of PSENNs and Prob-PSENN for the K-MNIST dataset is provided in Figures 15 and 16. Notice how, also due to the higher variability in the representations of each class, the prototypes learned by a PSENN with $m = 15$ prototypes (Figure 15-left) are not sufficiently representative of the classes, and, consequently, we need to resort to a large number of prototypes ($m = 50$, Figure 15-right). However, even for $m = 50$, the prototypes learned by PSENN are considerably less varied and informative than the ones sampled by Prob-PSENN, which evidences the higher flexibility of our approach, as well as its increased transparency and explanatory capabilities.

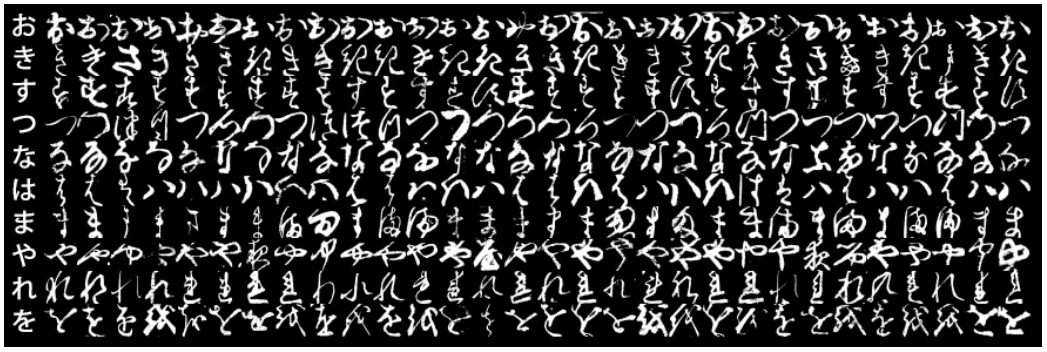
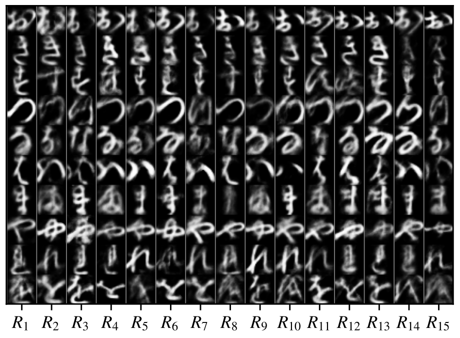


Figure 13: Examples of the K-MNIST dataset [9], as reported in the original repository `https://github.com/rois-codh/kmnist` (CC BY-SA 4.0 license). Each row corresponds to one class. The first column of the figure shows the modern representation of each letter, as a reference.


R_1 R_2 R_3 R_4 R_5 R_6 R_7 R_8 R_9 R_{10} R_{11} R_{12} R_{13} R_{14} R_{15}

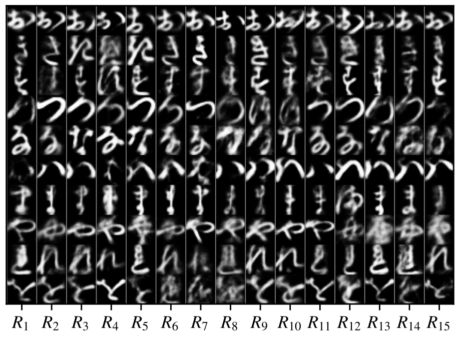

R_1 R_2 R_3 R_4 R_5 R_6 R_7 R_8 R_9 R_{10} R_{11} R_{12} R_{13} R_{14} R_{15}

Figure 14: Sampled prototypes for the K-MNIST dataset, using a Prob-PSENN with $l = 5$ (left) and $l = 10$ (right).

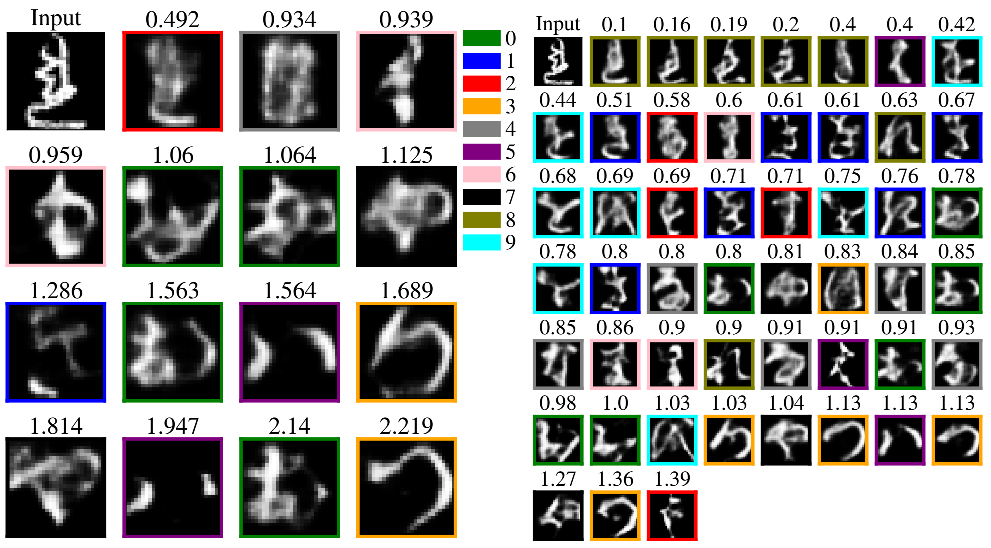


Figure 15: Explanations provided by a PSENN with $m = 15$ (left) and $m = 50$ (right) for the K-MNIST dataset. Both models classify the input (shown in the top-left corner of each grid) as the class *8* with a probability of 0.99. Note that the prototypes are sorted based on their distances to the input (displayed on top of each figure), in increasing order.

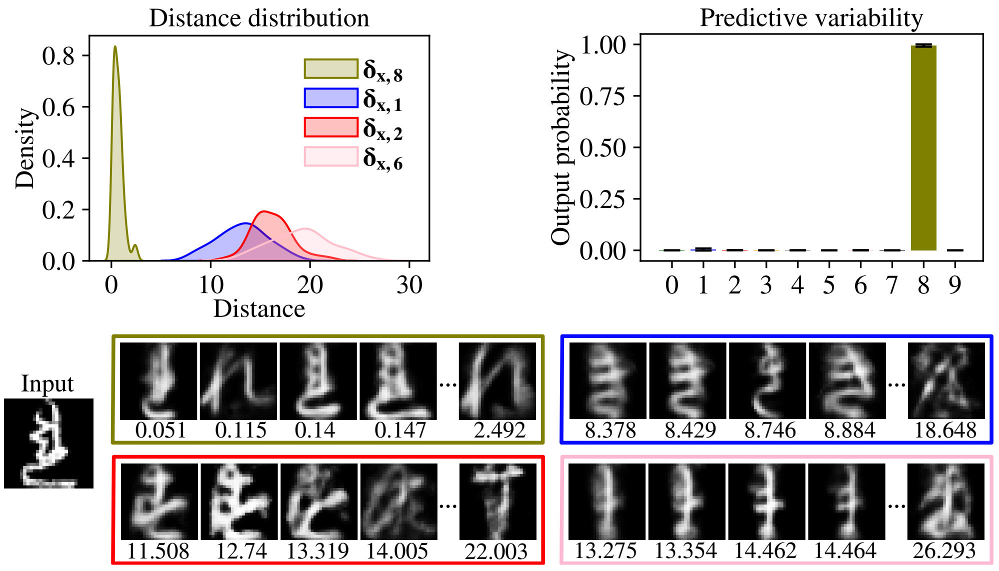


Figure 16: Explanation provided by a Prob-PSENN, for the K-MNIST dataset. The top row shows the distance distribution to the four closest classes (left), and the variability in the output probabilities (right). The bottom part shows the prototypes corresponding to the four closest classes, and the corresponding distances to the input. Note that the prototypes are ordered based on their distance, in increasing order.

Appendix F Results on the E-MNIST dataset

In order to analyze the scalability of ProbSENN in tasks consisting of larger number of classes, we evaluated its performance in the E-MNIST dataset [10] (47 classes), an extension of MNIST which includes both numeric and alphabetic characters. As it can be seen in Figure 17, ProbSENN is still capable of learning suitable distribution over the prototypes for the 47 classes, yielding realistic and diverse prototypes, while achieving a test accuracy of 0.85.

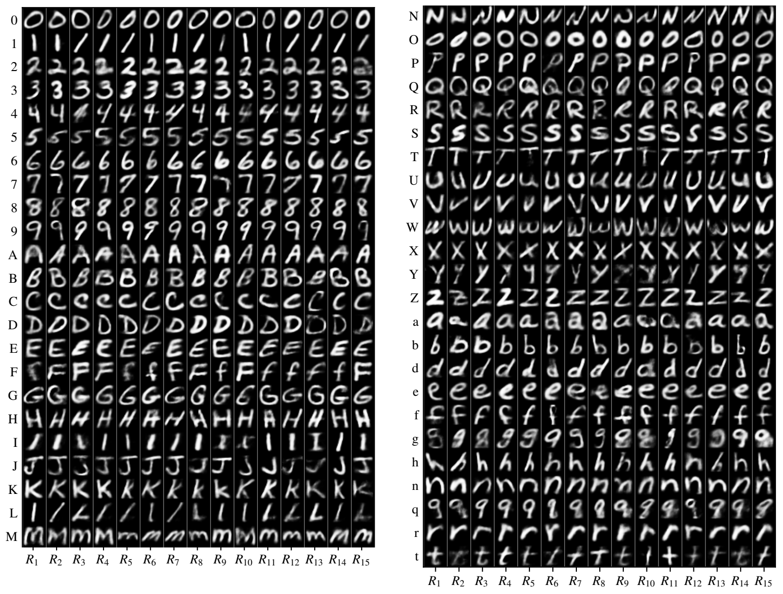

0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
a
b
d
e
f
g
h
n
q
r
t
R_1 R_2 R_3 R_4 R_5 R_6 R_7 R_8 R_9 R_{10} R_{11} R_{12} R_{13} R_{14} R_{15}

Figure 17: Prototypes sampled for the E-MNIST dataset (47 classes), consisting of handwritten representations of numbers and letters.

Appendix G Harnessing uncertainty to improve predictive accuracy and reliability: Results

Table 2: **Predictive accuracy percentages** (mean and standard deviation for 10 runs) obtained by Prob-PSENN for the following configurations: using the MLP-Autoencoder, using the CNN-Autoencoder, including elastic deformation as data augmentation (AUG), and discarding the inputs with an uncertainty rate $\mathscr{U}_E(x)$ (see Equation 4 in the paper) higher than a threshold α (UNC).

Configuration	MNIST	F-MNIST	K-MNIST
Prob-PSENN ($l = 10$, MLP)	97.59 ± 0.48	88.77 ± 0.50	88.92 ± 1.00
+ UNC ($\alpha = 0.99$)	98.99 ± 0.42	89.16 ± 0.52	96.19 ± 1.42
+ UNC ($\alpha = 0.95$)	99.68 ± 0.13	90.52 ± 0.67	98.77 ± 0.39
Prob-PSENN ($l = 10$, CNN)	98.28 ± 0.17	89.47 ± 0.34	89.06 ± 0.34
+ UNC ($\alpha = 0.99$)	98.83 ± 0.25	89.52 ± 0.41	89.58 ± 0.50
+ UNC ($\alpha = 0.95$)	99.68 ± 0.09	90.22 ± 0.59	94.58 ± 0.74
Prob-PSENN ($l = 10$, MLP + AUG)	98.53 ± 0.08	87.78 ± 0.50	93.17 ± 0.33
+ UNC ($\alpha = 0.99$)	98.97 ± 0.14	87.81 ± 0.49	95.29 ± 0.65
+ UNC ($\alpha = 0.95$)	99.67 ± 0.05	88.29 ± 0.57	98.44 ± 0.33
Prob-PSENN ($l = 10$, CNN + AUG)	98.75 ± 0.17	87.98 ± 0.46	89.40 ± 1.14
+ UNC ($\alpha = 0.99$)	99.00 ± 0.18	88.00 ± 0.46	89.56 ± 1.09
+ UNC ($\alpha = 0.95$)	99.74 ± 0.09	88.45 ± 0.52	93.53 ± 1.28

Table 3: **Percentage of discarded inputs** when filtering by uncertainty (mean and standard deviation for 10 runs).

Threshold	Configuration	MNIST	F-MNIST	K-MNIST
$\alpha = 0.99$	Prob-PSENN ($l = 10$, MLP)	2.77 ± 0.53	0.79 ± 0.26	11.49 ± 2.31
	Prob-PSENN ($l = 10$, CNN)	1.01 ± 0.26	0.13 ± 0.10	0.70 ± 0.40
	Prob-PSENN ($l = 10$, MLP + AUG)	0.90 ± 0.14	0.05 ± 0.03	4.25 ± 0.97
	Prob-PSENN ($l = 10$, CNN + AUG)	0.52 ± 0.11	0.04 ± 0.04	0.17 ± 0.11
$\alpha = 0.95$	Prob-PSENN ($l = 10$, MLP)	7.69 ± 0.50	5.16 ± 0.86	24.89 ± 1.33
	Prob-PSENN ($l = 10$, CNN)	5.27 ± 0.21	2.57 ± 0.45	12.38 ± 1.10
	Prob-PSENN ($l = 10$, MLP + AUG)	5.58 ± 0.23	1.84 ± 0.29	17.02 ± 1.01
	Prob-PSENN ($l = 10$, CNN + AUG)	4.67 ± 0.21	1.75 ± 0.30	9.59 ± 1.17

Table 4: **Error rate** (%) of the deterministic baseline model on the inputs receiving high/low uncertainty by Prob-PSENNs (with $l = 10$ and an MLP-Autoencoders).

Threshold	Configuration	MNIST	F-MNIST	K-MNIST
$\alpha = 0.99$	High uncert. ($\mathscr{U}_E(x) > \alpha$)	13.10 ± 1.87	25.56 ± 6.72	37.12 ± 2.69
	Low uncert. ($\mathscr{U}_E(x) \leq \alpha$)	0.73 ± 0.13	9.73 ± 0.32	5.36 ± 0.51
$\alpha = 0.95$	High uncert. ($\mathscr{U}_E(x) > \alpha$)	8.75 ± 1.24	25.55 ± 3.29	27.12 ± 1.38
	Low uncert. ($\mathscr{U}_E(x) \leq \alpha$)	0.41 ± 0.09	8.99 ± 0.35	2.92 ± 0.33

Appendix H Broader Impact

As specified above, the goal of Prob-PSENNs is to promote a transparent-by-design DNN architecture while maintaining competitive predictive performance. In addition to transparency, Prob-PSENNs provide tools to quantify the uncertainty in the model's predictions and explanations, which further enhances its safety and reliability. Therefore, our work is aligned with three important needs of deep learning models: interpretability, transparency, and reliability. However, a current concern in deep learning models is their lack of robustness to adversarial attacks [7, 42, 46], a vulnerability that also threatens explanation methods and self-explainable models [3, 15, 26, 43]. Therefore, in order to ensure a safe and responsible deployment of Prob-PSENNs in practice, it remains crucial to carefully assess the implications of adversarial attacks, and, consequently, to evaluate the robustness of the model against them.