Quantifying the Importance of Latent Features in Neural Networks

Amany Alshareef, Nicolas Berthier, Sven Schewe, Xiaowei Huang

Department of Computer Science
University of Liverpool, UK
{amany.alshareef, nicolas.berthier, sven.schewe, xiaowei.huang}@liverpool.ac.uk

Abstract

The susceptibility of deep learning models to adversarial examples raises serious concerns over their application in safety-critical contexts. In particular, the level of understanding of the underlying decision processes often lies far below what can reasonably be accepted for standard safety assurance. In this work, we provide insights into the high-level representations learned by neural network models. We specifically investigate how the distribution of features in their latent space changes in the presence of distortions. To achieve this, we first abstract a given neural network model into a Bayesian Network, where each random variable represents the value of a hidden feature. We then estimate the importance of each feature by analysing the sensitivity of the abstraction to targeted perturbations. An importance value indicates the role of the corresponding feature in underlying decision process. Our empirical results suggest that obtained feature importance measures provide valuable insights for validating and explaining neural network decisions.

Keywords— Neural network latent representation, Bayesian network, Feature importance, Sensitivity analysis.

1 Introduction

When neural networks are used in critical applications, the reliability of their decision making becomes a major concern. Various techniques have been developed to verify, falsify, enhance, or explain the neural networks, see (Huang et al. 2020) for a recent survey. In this paper, we focus on gaining insight on the decision mechanisms based on the main features of the network.

Deep neuronal networks (DNNs) learn their decision rule through training on a large dataset by gradually optimising parameters until they achieve the required accuracy. Therefore, they do not have a specific control-flow structure, which makes it difficult to precisely define suitable test criteria. Neuron activation (Pei et al. 2017) and other structural coverage techniques, such as MC/DC (Sun et al. 2019), that are defined based on the syntactic model components have proven to be less effective in validating the safety behaviour of the intelligent systems (Sun et al. 2018a). This paper analyses the internal representation of a DNN built from the training dataset, together with the training data itself, toward defining a testing approach that uses semantic aspects.

This is a contribution to a recent trend to exploring the internal logic of the learning model, such as eXplainable Artificial Intelligence (XAI) (Miller 2019), semantic-level robustness (Hamdi and Ghanem 2020; Xu et al. 2021), and exhibition of internal working mechanism through test cases (Huang et al. 2021).

We build on the work of Berthier et al. (2021), which elicited semantic assumptions by advancing an approach that relies on a Bayesian Network (BN) abstraction to examine whether latent features are adequately exercised by a set of inputs. The Bayesian view of statistics treats the latent parameters as random variables and seeks to learn a distribution of these parameters conditional on what is observed in the training data.

Contribution. Our key contribution is to propose a method that estimates the importance of a neural network's latent features by analysing an associated Bayesian network's sensitivity to distributional shifts. This allows us to define semantic testing metrics and to identify distributional shifts in the feature space through the effect they have on the random variables in BNs.

This provides us with a separation of concern: we can study the effect of a distributional shift in the latent feature space, which is typically low-dimensional, independent of potential shifts in the input distributions. This provides insight on the semantic mechanisms of decision making in the DNN as well as information for testing the sensitivity of features to distributional shifts.

A weighted scoring model is commonly applied in statistics when certain selected criteria are assigned more importance than others. The **feature importance (FI)** describes how much each feature influences the classifier's decision, and thus indicates the importance of the feature for the classification. This is to be contracted with the existing notion of feature importance in explanation models, which assigns the importance value to the features that belong to the input space, *e.g.,* age, sex, education. Instead, we investigate the learning models' latent feature space and examine how much their deep representation relies on a specific hidden feature to change their prediction.

We seek to evaluate the learning models' semantic robustness by developing a weight-based test metric that utilises the Bayesian Network model from Berthier et al. (2021). However, instead of directly using the extracted hidden features to measure some test coverage metric, we first compute a weight value, w_i, for the i-th latent feature, by analysing the BN's sensitivity to a controlled noise applied to this feature. In this paper, we develop several analyses that rely on the BN abstraction to estimate the relative impacts and sensitivity of the latent features. For example, we measure the relative impact one feature has on another for all feature pairs by estimating how a controlled noise impacts the BN's probability distributions. As an alternative approach, we also estimate the sensitivity to a given latent feature by comparing the probability distributions of training samples before and after the feature has been perturbed. Figure 1 outlines the proposed feature sensitivity analysis

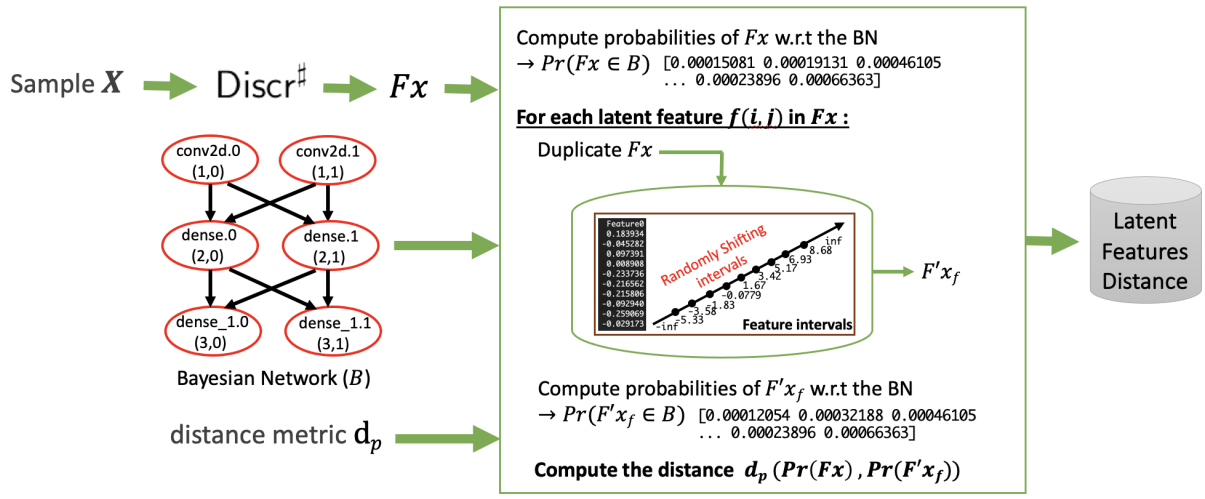

Figure 1: Illustration of the proposed BN analysis technique to compute the sensitivity of extracted latent features.

approach.

This allows us to monitor the behaviour of a DNN via its associated BN. The structure of this BN is built based on a *parameterisable abstraction scheme* that defines a series of DNN layers to consider (conv2d, dense and dense_1 in the Figure), a feature extraction technique to identify a given number of latent features for each one of these layers (2 in the example), and a discretisation strategy that determines the granularity at which values of latent features are aggregated into distinct intervals of indistinguishable values. Combined with the feed-forward nature of the DNNs we consider, this scheme allows us to derive the structure of a BN, as shown in Figure 1.

In addition, this scheme provides us with a *discretisation function* Discr$^\sharp$, that transforms a set of inputs X into a low-dimensional, discretised version F_X. In the Figure, the vector of inputs X is transformed into F_X, which associates each input $x \in X$ with six feature intervals, one for each latent feature represented by the BN. As the BN assigns a probability to an input sample that belongs to the distribution it represents, we can compare the probabilities of a sample under a given BN before and after a perturbation. To do so, we conduct the interior analysis on F_X by calculating the probability of each sample under the BN B. After that, we iterate over all considered latent feature f, and *shift* the associated intervals in F_X to produce a modified F'_{X_f} *w.r.t.* the feature f, and calculate its probability belonging to the BN B distribution. The term **intervals shifting** refers to a technique used to artificially simulate a controlled distribution shift by randomly shifting intervals in the selected feature space. To identify the impact of a perturbation, we compute a distance between the original probability vector and the probability vector obtained from the perturbed features.

2 Related Works

Robustness of Neural Networks latent features. Features play an essential role in the field of image processing and classification. They are considered as the basic conceptual components of the semantics of an image. Similar as this paper, there are recent studies concentrated on the features space to study the hidden semantic representation of intelligent models. Ilyas et al. (2019) categorised useful features in the input-space into robust and non-robust features. They demonstrate that adversarial perturbation can arise from flipping non-robust features in the data that are useful for the classification of regular inputs in the standard setting. They further argue that ML models are highly vulnerable to adversarial examples due to the presence of these useful non-robust features. This was further emphasised by Madaan, Shin, and Hwang (2020), who showed that the factor causing the adversarial vulnerability is the distortion in the latent feature space. Going beyond these works which justify the needs of considering features (instead of pixels or neurons), we study the causality relation between features by constructing a formal model – Bayesian network.

Bayesian Networks (BNs) and Neural Networks (NNs). Current trends towards using Bayesian modelling to solve challenging issues of neural networks have seen a growing recognition of the vital links between them. Daxberger et al. (2021) developed a framework for scaling Bayesian inference to NNs to be able to quantify the uncertainty in NN predictions. Furthermore, due to the scalability problem that arises when analysing neural networks, Berthier et al. (2021) uses a statistical analysis of activations at network layers, and abstracts the behaviours of the DNN using a Bayesian Network. They identify hidden features that have been learned by hidden layers of the DNN and associate each feature with a node of the BN. Their Bayesian network approximation model is therefore defined based on high-level features, rather than on low-level neurons. These extracted features are minimal semantic components that can be analysed to understand the behaviour of the feature space and the internal logic of the analysed DNN. This paper is, based on the BN in Berthier et al. (2021), to consider different methods to quantify the importance of latent features.

Bayesian Networks Sensitivity Analysis. Sensitivity analysis in Bayesian networks is concerned with understanding how a small change in local network parameters may affect the global conclusions drawn based on the network (Castillo, Gutiérrez, and Hadi 1997). The key aspect of performing a sensitivity analysis on a Bayesian network can be listed as follows:

- Quantify the impact of different nodes on a target node;
- Discover important features that have significant influence on the classifier decision;
- Determine sensitive parts of the network that might cause network vulnerability.

However, we cannot directly apply the sensitivity analysis in the traditional sense with Bayesian Networks, where the sensitivity is

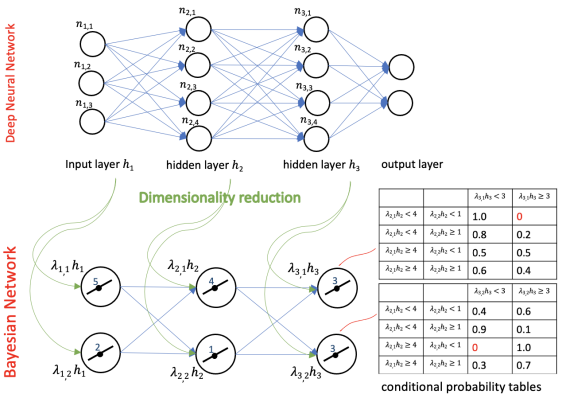

Figure 2: Structure of the Bayesian Network abstraction after reducing each h_1, h_2, h_3 into two features $\lambda_{i,1} \circ h_i$ and $\lambda_{i,2} \circ h_i$ with two intervals each. The conditional probability tables are shown for features $\lambda_{3,1}$ and $\lambda_{3,2}$.

performed by changing the BN parameter from the input space and observing how that influences the final decision. Instead, since our analysis targets the latent features in the low-dimensional space, we measure how sensitive are the BN probability distributions to changes in the values of hidden features. This process gives insight into how such perturbations impact the inner Bayesian network distribution and hence reflects the ground truth of the neural networks' behaviour in the presence of adversarial inputs.

Throughout this work, we use the Bayesian network probability distributions to study the neural networks latent features and analyse their deep representations.

3 Preliminaries

At the core of our approach lies the proposed BN-based latent feature analysis algorithms. Before that, we introduce the scheme and Bayesian Network that have been used by Berthier et al. (2021) as an explainable abstraction of DNNs' latent features.

Let $\mathcal{N}$ be a trained deep neural network with sequential layers $\mathcal{L} = (l_1, \ldots, l_K)$ and X a training dataset. As an abstract model of $\mathcal{N}$ and X, a Bayesian Network (BN) is a directed acyclic graph $\mathcal{B} = (V, E, P)$, where V are nodes, E are edges that indicate dependencies between features in successive layers, and P maps each node in V to a probability table representing the conditional probability of the current feature over its parent features *w.r.t.* X.

Example 1 *Figure 2 gives a simple neural network of 2 hidden layers and its Bayesian Network abstraction. h_i is a function that gives the neuron activations at layer l_i from any given input sample, and $\lambda_{i,j}$ is a feature mapping from the set $\Lambda_i = \{\lambda_{i,j}\}_{j \in \{1,\ldots,|\Lambda_i|\}}$. Each random variable $\lambda_{i,j} \circ h_i$ in the BN represents the j-th component of the value obtained after mapping h_i into the latent feature space. Since each function $\lambda_{i,j} \circ h_i$ ranges over a continuous space, the respective feature components—which are the codomains of the $\lambda_{i,j}$'s—are discretised into a finite set of feature intervals.*

Each node in BN abstractions represents an extracted feature, and we let $F^{\sharp}_{i,j} = \{f^{\sharp 1}_{i,j}, \ldots, f^{\sharp m}_{i,j}\}$, for the j-th extracted feature from layer l_i, be a finite set of m intervals that partition the value range of the feature. We formally define a **feature** as a pair (i, j), where i indexes a layer l_i in $\mathcal{L}$, and j identifies a component of the extracted feature space for layer l_i, *i.e.,* $j \in \{1, \ldots, |\Lambda_i|\}$. Each

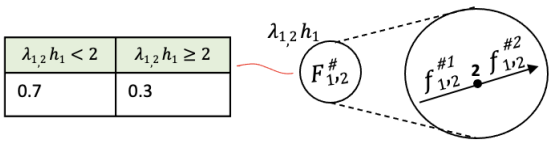

Figure 3: Illustration of probability tables and feature intervals with a Bayesian Network node.

node in the BN model is associated with either a marginal probability table for hidden features of layer l_1, or a conditional probability table for hidden or output layers. In Figure 2, the conditional probability table for the feature component $\lambda_{3,1}$ is defined for each feature interval $\{(-\infty, 3[, [3, +\infty)\}$ for layer l_3, *w.r.t.* each combination of the parent feature intervals from previous layer l_2.

Example 2 *Figure 3 illustrates an example node in a BN, which corresponds to the second extracted feature from the first NN layer, i.e., $i = 1, j = 2$. The set $F^{\sharp}_{1,2}$ contains two intervals, $f^{\sharp 1}_{1,2}$ and $f^{\sharp 2}_{1,2}$, which partition the real line. The node is denoted as a random variable named $\lambda_{1,2} \circ h_1$, which is associated with a probability table. The probability table is a marginal probability table because the features on the first layer do not have parent features. The table says that this feature has probability 0.7 to have a value smaller than 2 and probability 0.3 to have a value no less than 2.*

The previous demonstrating examples were simple, which illustrated a BN built using each layer of the NN. In practice, we can select specific NN layers to be abstracted, as we did in our analysis experiments.

Using Bayesian Network Abstractions. We can fit the set of probability tables in a BN abstraction $\mathcal{B}$ by using a training sample X. This process first transforms X by means of the discretisation function to obtain a vector of elements from the discretised latent feature space $F_X = \mathsf{Discr}^{\sharp}(X)$. It then updates the probability tables in $\mathcal{B}$ in such a way that the joint probability distribution it represents fits the distribution of F_X.

We can query the fitted BN $\mathcal{B}$ for the probabilities of the discretised input sample $F_{X'}$. We denote this query operation $\Pr(F_{X'} \in \mathcal{B})$, and may abuse this notation by defining $\Pr(X' \in \mathcal{B}) = \Pr\big(\mathsf{Discr}^{\sharp}(X') \in \mathcal{B}\big)$.

Perturbation of Latent Features. Our developments in the next Section rely on the application of a controlled change of a feature (i, j) in an element F_x of the latent feature space. This operation simulates a distortion in single targeted component of the latent feature space by substituting its associated interval with an adjacent one. (We assume that each latent feature component is partitioned into at least two intervals). When an interval has two neighbours, we chose uniformly at random between them. We denote this operation with the function $\mathrm{random\_shift}(F_x, i, j)$, which replaces the feature interval $f^{\sharp k}_{i,j}$ of F_x with either $f^{\sharp k-1}_{i,j}$ or $f^{\sharp k+1}_{i,j}$. For instance, assuming two hidden feature components extracted from activations at two layers of a NN, each component being discretised into small-enough intervals, *i.e.,* 10 intervals, $\mathrm{random\_shift}((f^{\sharp 4}_{0,0}, f^{\sharp 7}_{0,1}, f^{\sharp 1}_{1,0}, f^{\sharp 9}_{1,1}), 1, 0)$ returns either $(f^{\sharp 4}_{0,0}, f^{\sharp 7}_{0,1}, f^{\sharp 0}_{1,0}, f^{\sharp 9}_{1,1})$ or $(f^{\sharp 4}_{0,0}, f^{\sharp 7}_{0,1}, f^{\sharp 2}_{1,0}, f^{\sharp 9}_{1,1})$.

4 BN-based Latent Feature Analysis

In this section, we develop several BN-based analysis approaches we employ to gain insights on latent features. The first approach produces a pairwise comparison matrix that exhibits the relative

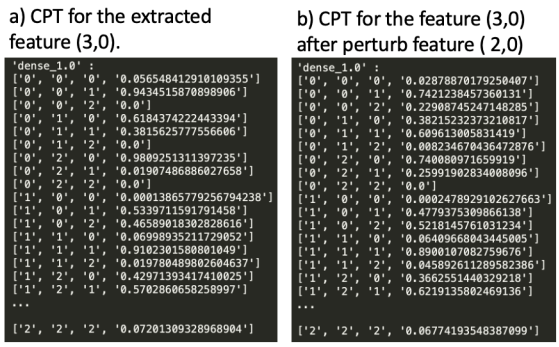

Figure 4: A toy example, with only three intervals for each feature, illustrates the conditional probability table for the first extracted feature from layer dense_1 before and after shifting intervals of feature $(2, 0)$ in the dataset used to fit the BN.

impact the latent features have on each other. Next, we leverage the BN to estimate the sensitivity of individual features to a controlled distribution shift. We then describe how the sensitivity analysis technique can be applied to define feature importance based on a generic definition of weights on features. Finally, we formalise a concrete definition of weights based on our BN-based feature sensitivity.

4.1 Pairwise Comparison

This particular study is to assess the degree to which the extracted features can affect each other by comparing the parallelised Conditional Probability Tables (CPTs) of a sample, under a BN, with the CPTs of the same sample after perturbing the features intervals of the BN. The pairwise comparison method is used to make a recursive comparison. It begins by extracting a set of inputs X from training data, and computing its feature intervals with $\mathsf{Discr}^\sharp$. This produces a sample F_X of intervals w.r.t. X. To generate the probability tables, we fit the Bayesian network with F_X, which gives the clean reference probability tables $CPTs(F_X)$. Figure 4-(a) shows the CPT for feature $(3, 0)$, which is the first extracted feature from the third NN layer, named dense_1 in the BN from Figure 1.

To extract knowledge about a given feature's independence and robustness, we apply a controlled change to the targeted feature f, by using the $\mathrm{random\_shift}$ operation to shift f's intervals in F_X to obtain F'_X. We then re-fit the BN's probabilities with F'_X, which gives the modified probability tables $CPTs(F'_X)$ *w.r.t.* the perturbed feature f, examplified in Figure 4-(b). To identify the impact, we use the mean squared error (MSE) between each corresponding table in the reference $CPTs(F_X)$ and generated $CPTs(F'_X)$.

We illustrate and give an example of pairwise comparison in Section 5 below.

4.2 Feature Sensitivity Analysis

The core benefit of relying on a Bayesian Network is to have a model that exhibits the relevant theoretical aspects of Bayesian analysis. To estimate the sensitivity of the abstraction scheme on a given latent feature, we measure the impact of artificially perturbing the intervals representing the selected feature on the probability distribution represented by the BN. In this algorithm, the BN is already fitted using a training dataset, and the distribution it represents does not change.

Algorithm 1: BN-based Feature Sensitivity Analysis

Input: Bayesian network $\mathcal{B}$ and associated feature mapping & discretisation function $\mathsf{Discr}^\sharp$, training dataset X, distance metric d_p.

Output: Mapping associating a distance measure with each considered latent feature

1: Compute the feature intervals *w.r.t.* X: $F_X = \mathsf{Discr}^\sharp(X)$
2: Compute the reference probabilities of F_X *w.r.t.* $\mathcal{B}$: $P_{ref} = \Pr(F_X \in \mathcal{B})$
3: **for** each considered feature $f = (i, j)$ **do**
4: $\quad P'_f = \langle \Pr(\mathrm{random\_shift}(F_x, i, j) \in \mathcal{B}) \rangle_{F_x \in F_X}$
5: $\quad d_f = d_p\Big(P_{ref}, P'_f\Big)$
6: **end for**
7: **return** distances d_f, for all f

The feature sensitivity analysis is given in Algorithm 1. This procedure receives an input sample X, taken from the training dataset, and first performs the feature projection and discretisation step with $\mathsf{Discr}^\sharp$ to obtain the associated feature intervals F_X. It then calculates the probability of each element of F_X *w.r.t.* the BN $\mathcal{B}$; this gives the vector of reference probabilities P_{ref}, that associates a probability with each set of abstracted latent features that are elicited by each x in X. Then, for each extracted feature f, a random perturbation is performed in F_X via the $\mathrm{random\_shift}$ function introduced in the previous Section. This leads to a second vector, that holds the probabilities of the resulting F'_{X_f} *w.r.t.* the BN $\mathcal{B}$. The given distance d_p between these two probability vectors for the perturbed feature f is eventually computed.

We chose to make the feature sensitivity analysis algorithm parametric in the distance metric p for the purposes of easing further experimental use of the FI measure. The considered distances are:

- L_p's with different norms, typically 1, 2, or ∞;
- JS is the Jensen-Shannon distance, that is a metric that measures the similarity between two probability distributions based on entropy computations;
- $corr$ is the correlation distance;
- cos is the cosine distance;
- MSE is the *mean squared error*;
- $RMSE$ is the *root mean squared error*;
- MAE is the *mean absolute error*;
- AF is a special purpose *anti-fit* divergence, which we define based on the *coefficient of determination* R^2. R^2 is a score that is typically used as a "goodness-of-fit" measure for regression models, and we refer to it as score_{R^2}. While the maximal score is 1 (indicating a perfect fit), the score decreases with the amount of variance in P that is not in Q and can take negative values. With this we define $d_{AF}(P, Q) = 1 - \mathsf{score}_{R^2}(P, Q)$.

  The rationale of using score_{R^2} as a basis for measuring the divergence is that we can view the probability vectors for perturbed features as output by a model. Divergence will be large when the effect of the perturbation is significant, and small when the model is not (very) sensitive to the perturbation.

distance perturbed feature	d_{L_1}	d_{L_2}	d_{L_∞}	d_{JS}	d_{corr}	d_{cos}	d_{MSE}	d_{RMSE}	d_{MAE}	d_{AF}
$(1,0)$	150	0.726	0.009 56	0.224	0.142	0.114	0.000 000 879	0.000 937	0.000 249	0.278
$(1,1)$	340	1.18	0.009 89	0.353	0.448	0.361	0.000 002 32	0.001 52	0.000 567	0.735
$(2,0)$	325	1.09	0.009 46	0.365	0.332	0.267	0.000 001 98	0.001 41	0.000 541	0.625
$(2,1)$	360	1.16	0.0103	0.393	0.395	0.323	0.000 002 24	0.001 50	0.000 600	0.710
$(3,0)$	276	0.880	0.008 89	0.258	0.170	0.137	0.000 001 29	0.001 14	0.000 460	0.408
$(3,1)$	315	1.07	0.009 60	0.324	0.318	0.264	0.000 001 92	0.001 39	0.000 525	0.608

Table 1: Example distance measures.

4.3 Feature Importance

We associate each extracted feature f with a weight w_f based on the set of measured sensitivity distances as follows:

$$w_f = \frac{e^{d_f}}{\sum_{f \in T} e^{d_f}} \quad (1)$$

where T is the set of considered latent features. The soft-max weighting in Eq. (1) acts as a normalisation function, *i.e.,* it ensures the sum of the feature components' weights equals one. The normalised importance weight for each feature is usually positively correlated with the respective probabilities distances.

Example 3 *Table 1 shows selected distance measures computed based on one experiment detailed in the next Section. Assuming the d_{corr} distance is chosen to determine feature importance,*

feature $(1,1)$ is assigned the largest weight at 0.192, followed by feature $(2,1)$ at 0.182, etc.

The importance weight for an extracted latent feature of a DNN's layer may reflect some relevant amount of information/variance/ that the abstracted DNN uses at the considered layer. The current abstraction scheme, however, does not relate latent features with the DNNs' decisions. Still, perturbing a specific part of the latent space and observing the implicit changes of the learning models' distribution contributes to understanding their internal decisions.

5 Experiments

In this Section, we first illustrate the results of pairwise comparison of latent features, and then turn to an empirical evaluation of the sensitivity of BN abstractions at detecting distribution shift induced by adversarial examples.

5.1 Illustration of Pairwise Comparison

We first concentrate on the BN given in examples so far.

We report in Table 2 a pairwise comparison matrix for this example, where we arrange the perturbed features in the first column and compute their impact on each feature (i,j)'s probability tables. The numbers reported in this matrix represent the change in the probability values. For instance, our controlled perturbation of feature $(2,0)$ intervals has an impact on features $(3,0)$ and $(3,1)$ values. More specifically, the MSE between the $(3,0)$ probability tables for feature $(3,0)$, given in Figure 4 (a) before and (b) after perturbing feature $(2,0)$, is 0.0113.

Discussion. Suppose we set the diagonal line to zeros since the change is made from the feature itself. In that case, we can observe that the perturbations are not affecting the probability of features from the previous layer (parent features) or the same layer as expected. On the other hand, random shifting only influenced the immediate features in the next layer. The largest difference occurred on feature $(3,1)$ when perturbing feature $(2,1)$. Although this impact is relatively small, we can (as expected) observe the dependencies between latent feature values of the BN model. However, the perturbations do not change the features' probability for deeper layers, *e.g.,* features of Layer 3 are not affected by the perturbation made on features of Layer 1, which is surprising.

5.2 Sensitivity Analysis

Let us now turn to our empirical assessment of the effectiveness of the BN sensitivity analysis method in examining the behaviour of the latent features under perturbation.

Datasets and Experimental Setup. We have selected two trained CNN models for our experiments: the first one targets the MNIST classification problem with 99.38% validation accuracy, and the second model targets the CIFAR-10 dataset with 81.00% validation accuracy. The models are reasonably sized, with more than 15 layers including blocks of convolutional and max-pooling layers, followed by a series of dense layers. They have 312 000 and 890 000 trainable parameters, respectively.

The Bayesian Network abstraction scheme accepts a wide range of feature extraction techniques and discretisation strategies. To explore their impact on our approach, we use a wide set of BN abstractions. We have selected two linear feature extraction techniques: Principal Component Analysis (PCA) and Independent Component Analysis (ICA), and one non-linear technique: radial basis functions (RBF) kernel-PCA. We also decided to fix the number of extracted features at three features per layer; this choice of a relatively small number of hidden features enables us to use many intervals (5 or 10) for their discretisation while still obtaining reasonably-sized probability tables. We applied both uniform- and quantile-based discretisation strategies, with or without the addition of two left- and right-most intervals that do not contain any element of the training sample. Finally, we considered three hidden layers to construct the BN abstractions: for the two models, the first two selected layers directly follow a block of convolutions, while the last is a dense ReLU layer that is situated few layers before the NN's output layer. The layers chosen criteria is based on a belief that the activation values at these layers capture relevant patterns w.r.t the NN decisions.

Example Distributions and Distances. We plot in Figure 5 example distributions of probabilities in vectors obtained from a BN abstraction of the MNIST model. We have annotated each one of these plots with various measures of distances between the reference probabilities P_{ref} that is generated using a sample from the training data set, and the respective six perturbed features probabilities P'_f. The shown difference between these two probability distributions illustrates the internal change in the distribution represented by the BN. For instance, when applying the $\mathrm{random\_shift}$ on the first feature that is extracted from the first selected layer *i.e.,* perturbed feature (1,0), the calculated probability

perturbed feature	$(1,0)$	$(1,1)$	$(2,0)$	$(2,1)$	$(3,0)$	$(3,1)$
$(1,0)$	0.003 01	0.	0.008 58	0.008 08	0.	0.
$(1,1)$	0.	0.002 57	0.007 83	0.008 65	0.	0.
$(2,0)$	0.	0.	0.0143	0.	0.0113	0.008 07
$(2,1)$	0.	0.	0.	0.0102	0.008 89	0.0114
$(3,0)$	0.	0.	0.	0.	0.0229	0.
$(3,1)$	0.	0.	0.	0.	0.	0.0161

Table 2: Example pairwise comparison matrix for six extracted features. Each cell describes the extent to which a feature (rows) affects the others (columns).

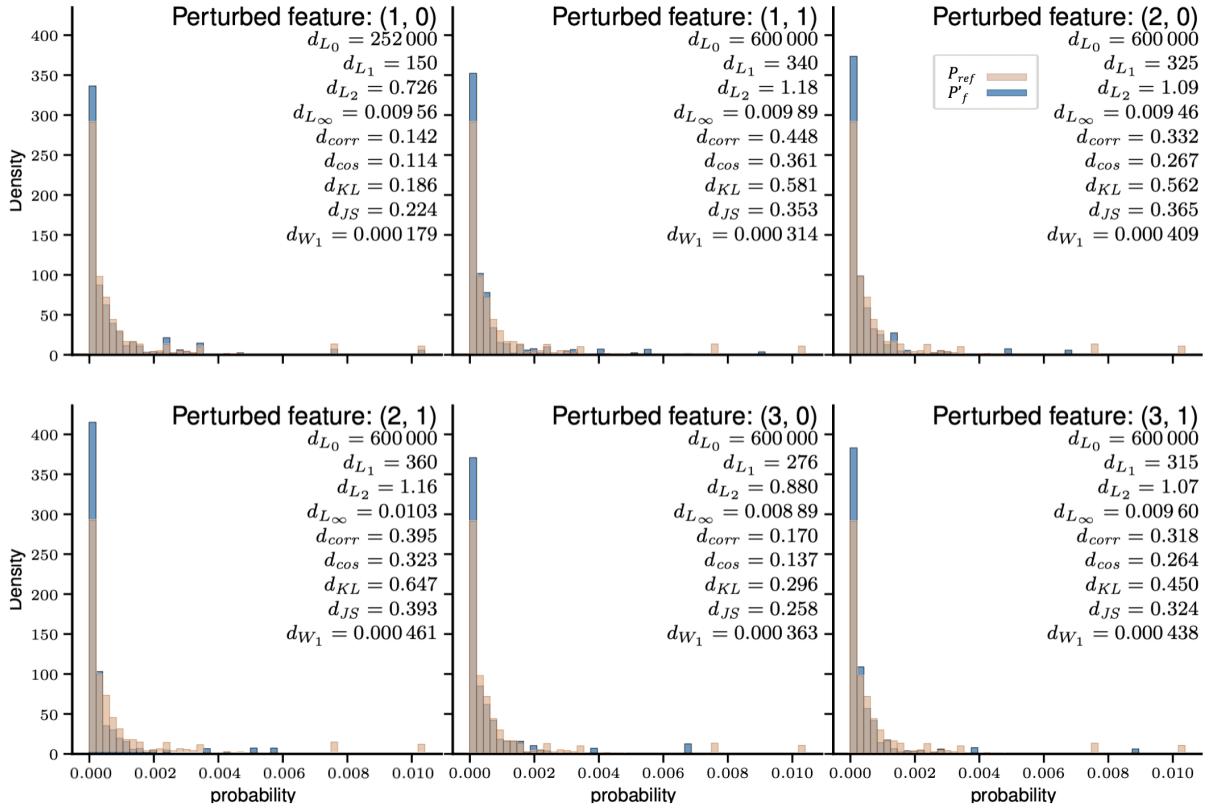

Figure 5: Distributions of probabilities obtained from one example BN abstraction for each perturbed feature, for the MNIST model. Each plot shows respective distance measures *w.r.t.* the probabilities obtained from the BN for the clean (unperturbed) features (P_{ref}).

distribution $P'(1,0)$, coloured with blue, shows a change on probabilistic causal relation that implies the change on the probability represented by the BN. Hence, we can determine the safety violation risk by comparing an input probability belonging to the BN probability distribution.

Sensitivity to Adversarial Distribution Shift. We have carried out a set of experiments to assess whether the set of three features extracted for each considered hidden layer allows us to capture relevant properties of the learnt representations. In particular, we wanted to check whether the BN abstraction allows us to detect the shift in the distribution of inputs that occurs when the NN is subject to adversarial examples. In other words, we want to discover whether some distance measures indicate that the BN abstractions capture relevant latent features (and their dependencies) with sufficient precision to associate diverging probabilities between "legitimate" inputs and adversarially perturbed ones. If such is the case, we shall conclude that our abstraction scheme and the associated BN are sufficiently precise to capture relevant dependencies in latent feature values that may not be matched (or matched too well, depending on the sign of the actual difference in probabilities) by some adversarial inputs.

To carry out these experiments, we have selected the following adversarial attacks:

fgsm is the Fast Gradient Sign Method of Goodfellow, Shlens, and Szegedy (2015);

pgdlinf and pgdl2 are the Projected Gradient Descent approach of Madry et al. (2017) with L_∞ and L_2 norm, respectively;

cwlinf and cwl2 are Carlini and Wagner (2017)'s attack with L_∞

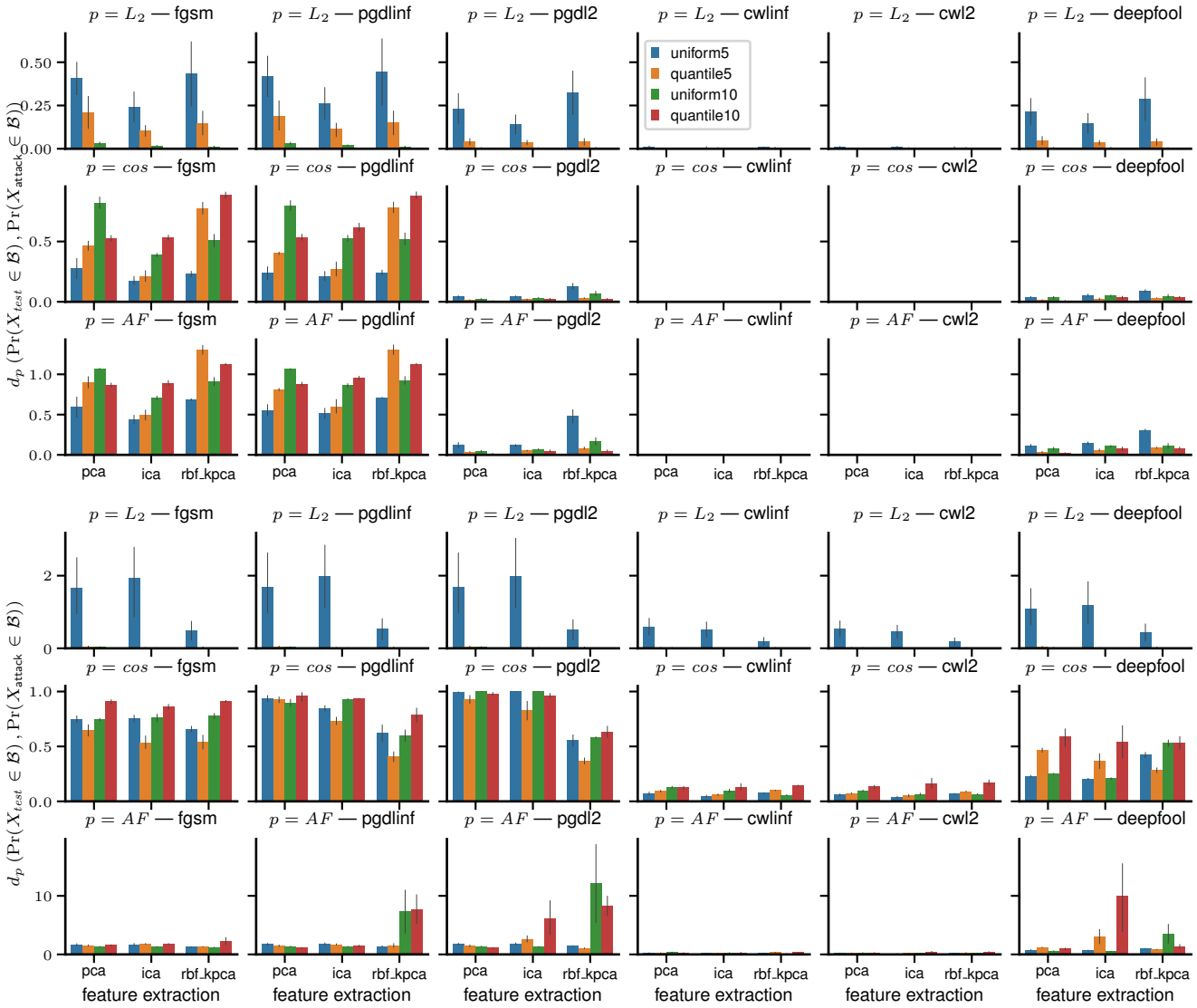

Figure 6: Selected distances (vertical axes) between probability vectors obtained for the validation dataset ($\Pr(X_{test} \in \mathcal{B})$) and probability vectors ($\Pr(X_{\text{attack}} \in \mathcal{B})$) obtained for datasets generated by selected adversarial attacks (shown on each column), for a range of BN abstractions $\mathcal{B}$. The top (resp. bottom) three rows show results for the MNIST (resp. CIFAR10) model. Hue indicates the discretisation strategy and the number of intervals. The grey vertical lines show confidence intervals.

and L_2 norm, respectively, both targeting 0.1 confidence;

deepfool is the DeepFool attack by Moosavi-Dezfooli, Fawzi, and Frossard (2016).

Attacks involving the L_∞ norm target a maximum of $\varepsilon = 0.1$ perturbation in the input images, whereas pgdl2 targets a maximum perturbation $\varepsilon = 10$.

For each attack, we have generated an adversarial dataset X_{attack} from the validation dataset X_{test} for both the MNIST and CIFAR10 models, where each dataset consists of 10 000 inputs. Then, for each attack and BN abstraction $\mathcal{B}$ built and fit using 20 000 elements drawn from the respective training datasets, we measured a set of distances p between the vectors of probabilities $\Pr(X_{test} \in \mathcal{B})$ and $\Pr(X_{\text{attack}} \in \mathcal{B})$, denoted $d_p(\Pr(X_{test} \in \mathcal{B}), \Pr(X_{\text{attack}} \in \mathcal{B}))$.

Results and Discussion. Figure 6 shows our results for three selected distances L_2, cos, and AF. We give more detailed results in Appendix A. Each chart in the figure illustrates the calculated distances with four colours according to the discretisation method and the number of intervals in the vertical axis, using three sets of feature extraction techniques (pca, ica, and rbf.kpca) in the horizontal axis. The used distance metric and attack type are shown at each chart's top. First of all, we can observe that some combinations of abstractions and distance measures exhibit notable differences between the validation dataset and the adversarial one for some attacks. For instance, every distance shown allows us to measure a shift in input distribution for every attack, except Carlini and Wagner (2017)'s in some cases. Next, although the feature extraction technique does not have a noticeable impact on any measured distance, the discretisation strategy certainly plays a role in the ability of the BN to model each abstracted latent feature and their de-

pendencies with sufficient precision. For example, in the first row of the CIFAR-10 experiment (L_2 distance), the distribution shift is detected when using the uniform-based discretisation method with five intervals (distance with blue color).

Overall, the experimental results show that computing distances between two BN probability distributions, clean and perturbed by intervals-shift or adversarial attacks, can detect the distribution shift where it exists. We emphasise that, in the case of adversarial shift, this is measured based on *the latent features only*. Given this empirically confirmed property, BN-based computation of feature importance appears to be one tool, which adds to the growing set of useful techniques for the detection of important features as well as of adversarial examples. What is more, it adds a semantic twist to this analysis and allows for explaining in which way the changes in the features contribute to the distribution shift.

6 Discussions

In this section, we discuss a few aspects related to either the method we take or the potential application of the method.

Hyper-parameters in BN Construction. The parametric nature of the scheme advanced by Berthier et al. (2021) enables the exploration of a wide range of DNN abstractions. For instance, in our experiments, the sensitivity to adversarial distribution shift is relied most on the *linear* dimensionality reduction techniques to extract latent features. We plan to conduct further experiments with more non-linear feature extraction techniques, like manifold learning (Lee 2000), to assess the properties of extracted features in extended cases. The effect of more advanced discretisation strategies can also be explored, for instance by relying on kernel density estimations to partition each latent feature component into intervals that span across ranges of the real line that are either densely or non-densely exercised by the training sample.

Hyper-parameters in Weight Quantification. There are a number of building blocks in the weight quantification method (Algorithm 1), including *e.g.,* the perturbation made to generate new CPTs, the random shifting function, and the distance metrics for probabilities (P_{ref}) and (P'_f). In this paper, we have explored several different options of the distance metrics for a comparison. It would also be useful to study if and how the other hyper-parameters may affect the overall results.

Utility of Feature Weights. Quantifying the importance of the hidden features provides three advantages. First, visualising the most important features provides insight into the model's internal decisions by highlighting dominating regions in the feature space.

Second, we can use the importance measurement to design high-level testing metrics that evaluate the robustness of the DNN. Some attempts have been made in Berthier et al. (2021), where no feature weight is taken into consideration.

Third, with FI as a defence, we can utilise the obtained importance in the training process and force the DNN to adjust its parameters according to the features that are most relevant for the prediction. This direction is the most widely adopted strategy. For example, Zhang et al. (2021) propose a hierarchical feature alignment method that computes the difference between clean and adversarial feature representations and utilises it as a loss function when optimising network parameters, while Bai et al. (2021) suggest that different channels of a DNN's intermediate layers contribute differently to a specific class prediction and propose a Channel-wise Activation Suppressing training technique that learns the channel importance, and leverages them to suppress the channel activation while training the network.

Utility of Bayesian Network. As suggested, BN can be seen as an abstraction of the original DNN. It is therefore imperative to understand how this abstraction may help in either analysing or enhancing the original DNN. In Berthier et al. (2021), test metrics are designed over the BN by extending the MC/DC metrics proposed by Sun et al. (2019). As the next step, it would be interesting to understand if test case generation methods (Sun et al. 2018b), in particular the one based on symbolic computation (Sun, Huang, and Kroening 2018), can also be extended to work with BNs. Moreover, it will be useful to see if the generated test cases can be more nature and diverse when comparing with those generated directly on DNNs, as done in (Huang et al. 2021).

In addition to testing, it would also be interesting to see if such abstraction may bring any benefit to e.g., verification (Huang et al. 2017), interpretation of DNN training (Jin et al. 2020), explainable AI (Zhao et al. 2021c), and safety case (Zhao et al. 2020). For example, scalability is the key obstacle of DNN verification due to its complexity (Ruan, Huang, and Kwiatkowska 2018). Considering that BN is significantly smaller than the original DNN, it will be interesting to understand if BN can be used to alleviate the problem without losing the provable guarantee. A potential difficulty may be whether and how the verification result on the BN can be transferred to the DNN.

Similar as the above discussion for testing and verification, the potential for the BN to be used as an intermediate step for the reliability assessment (Zhao et al. 2021a) and safety case (Zhao et al. 2021b) is worthy of exploration. This may probably require a quantification of the error, or the loss of information, when using BN as an abstraction of the DNN.

7 Conclusions

In this study, we have advanced a novel technique that employs a BN abstraction to investigate how to measure the importance of high level features when they are used by the neural network to make classification decisions. In addition to the observed ability of detecting the distribution shifts before and after perturbation, this will open many doors for future exploration. For example, it will certainly be interesting to understand if the generated importance values can support the explanation of the black-box learning model. It will also be useful if such importance values can be utilised to improve the training process.

References

Bai, Y.; Zeng, Y.; Jiang, Y.; Xia, S.-T.; Ma, X.; and Wang, Y. 2021. Improving Adversarial Robustness via Channel-wise Activation Suppressing. In *International Conference on Learning Representations*.

Berthier, N.; Alshareef, A.; Sharp, J.; Schewe, S.; and Huang, X. 2021. Abstraction and Symbolic Execution of Deep Neural Networks with Bayesian Approximation of Hidden Features. *arXiv preprint arXiv:2103.03704*.

Carlini, N.; and Wagner, D. 2017. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, 39–57. IEEE Computer Society.

Castillo, E.; Gutiérrez, J. M.; and Hadi, A. S. 1997. Sensitivity analysis in discrete Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 27(4): 412–423.

Daxberger, E.; Nalisnick, E.; Allingham, J. U.; Antorán, J.; and Hernández-Lobato, J. M. 2021. Bayesian deep learning via subnetwork inference. In *International Conference on Machine Learning*, 2510–2521. PMLR.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. arXiv:1412.6572.

Hamdi, A.; and Ghanem, B. 2020. Towards analyzing semantic robustness of deep neural networks. In *European Conference on Computer Vision*, 22–38. Springer.

Huang, W.; Sun, Y.; Zhao, X.; Sharp, J.; Ruan, W.; Meng, J.; and Huang, X. 2021. Coverage-Guided Testing for Recurrent Neural Networks. *IEEE Transactions on Reliability*, 1–16.

Huang, X.; Kroening, D.; Ruan, W.; Sharp, J.; Sun, Y.; Thamo, E.; Wu, M.; and Yi, X. 2020. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, 37.

Huang, X.; Kwiatkowska, M.; Wang, S.; and Wu, M. 2017. Safety Verification of Deep Neural Networks. In *International Conference on Computer Aided Verification*, 3–29. Springer.

Ilyas, A.; Santurkar, S.; Tsipras, D.; Engstrom, L.; Tran, B.; and Madry, A. 2019. Adversarial Examples Are Not Bugs, They Are Features. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Jin, G.; Yi, X.; Zhang, L.; Zhang, L.; Schewe, S.; and Huang, X. 2020. How does Weight Correlation Affect the Generalisation Ability of Deep Neural Networks. *Advances in Neural Information Processing Systems*.

Lee, J. 2000. A global geometric framework for non-linear dimensionality reduction. In *Proceedings of the 8th European symposium on artificial neural networks, 2000*, volume 1, 13–20.

Madaan, D.; Shin, J.; and Hwang, S. J. 2020. Adversarial neural pruning with latent vulnerability suppression. In *International Conference on Machine Learning*, 6575–6585. PMLR.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards Deep Learning Models Resistant to Adversarial Attacks. arXiv:1706.06083.

Miller, T. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267: 1–38.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; and Frossard, P. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2574–2582.

Pei, K.; Cao, Y.; Yang, J.; and Jana, S. 2017. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*, 1–18.

Ruan, W.; Huang, X.; and Kwiatkowska, M. 2018. Reachability Analysis of Deep Neural Networks with Provable Guarantees. In *IJCAI*, 2651–2659.

Sun, Y.; Huang, X.; and Kroening, D. 2018. Testing Deep Neural Networks. *CoRR*, abs/1803.04792.

Sun, Y.; Huang, X.; Kroening, D.; Sharp, J.; Hill, M.; and Ashmore, R. 2018a. Testing deep neural networks. *arXiv preprint arXiv:1803.04792*.

Sun, Y.; Huang, X.; Kroening, D.; Sharp, J.; Hill, M.; and Ashmore, R. 2019. Structural Test Coverage Criteria for Deep Neural Networks. *ACM Trans. Embed. Comput. Syst.*, 18(5s).

Sun, Y.; Wu, M.; Ruan, W.; Huang, X.; Kwiatkowska, M.; and Kroening, D. 2018b. Concolic Testing for Deep Neural Networks. In *ASE*, 109–119.

Xu, Q.; Tao, G.; Cheng, S.; and Zhang, X. 2021. Towards Feature Space Adversarial Attack by Style Perturbation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 10523–10531.

Zhang, X.; Wang, J.; Wang, T.; Jiang, R.; Xu, J.; and Zhao, L. 2021. Robust feature learning for adversarial defense via hierarchical feature alignment. *Information Sciences*, 560: 256–270.

Zhao, X.; Banks, A.; Sharp, J.; Robu, V.; Flynn, D.; Fisher, M.; and Huang, X. 2020. A Safety Framework for Critical Systems Utilising Deep Neural Networks. In *SafeComp2020*, 244–259.

Zhao, X.; Huang, W.; Banks, A.; Cox, V.; Flynn, D.; Schewe, S.; and Huang, X. 2021a. Assessing the Reliability of Deep Learning Classifiers Through Robustness Evaluation and Operational Profiles. In *AISafety*.

Zhao, X.; Huang, W.; Bharti, V.; Dong, Y.; Cox, V.; Banks, A.; Wang, S.; Schewe, S.; and Huang, X. 2021b. Reliability Assessment and Safety Arguments for Machine Learning Components in Assuring Learning-Enabled Autonomous Systems. arXiv:2112.00646.

Zhao, X.; Huang, X.; Robu, V.; and Flynn, D. 2021c. BayLIME: Bayesian Local Interpretable Model-Agnostic Explanations. In *UAI*.

A Detailed Results for Sensitivity to Adversarial Shift Experiments

We have plotted in Figure 6 some statistics for a subset of the distances we have considered for comparing probability vectors. Figures 7, 8, and 9 show the distances computed for the MNIST model, and Figures 10, 11, and 12 show the results for the CIFAR10 model. In these plots, hue still indicates the discretisation strategy. However, we have discriminated between extended and non-extended strategies: the prefix '-X' denotes that latent features are discretised in such a way that left- and right-most intervals do not contain any (projected) training sample.

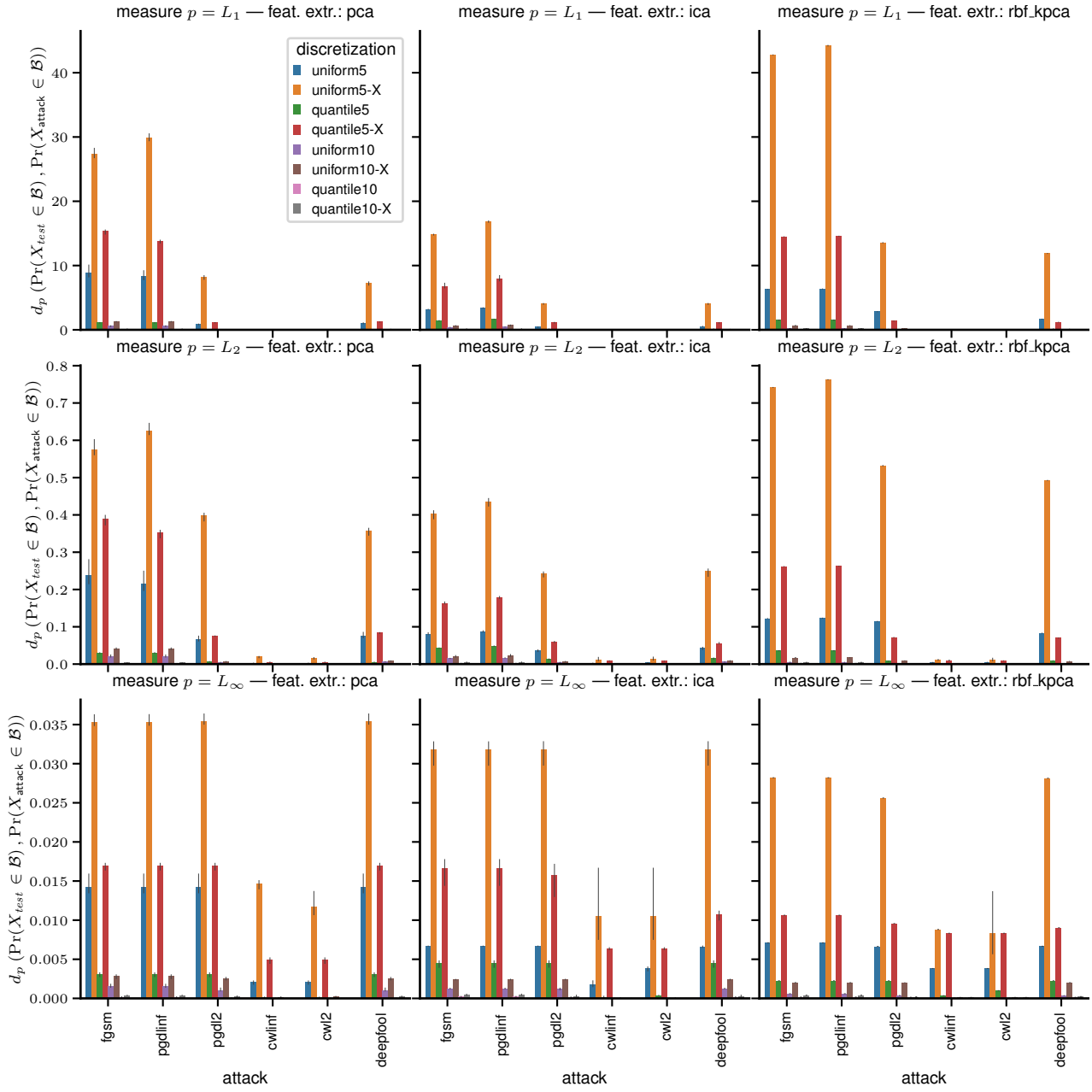

Figure 7: Distances (vertical axes) between probability vectors obtained for the MNIST validation dataset ($\Pr(X_{test} \in \mathcal{B})$) and probability vectors ($\Pr(X_{\mathsf{attack}} \in \mathcal{B})$) obtained for datasets generated by selected adversarial attacks (attack, shown on the horizontal axes), for a range of BN abstractions $\mathcal{B}$. Every abstraction involves 3 layers for which 3 features have been extracted using PCA (left-hand side column), ICA (middle), or radial basis functions (RBF) kernel-PCA (right). Plotted data aggregates five independent runs, and shows confidence intervals.

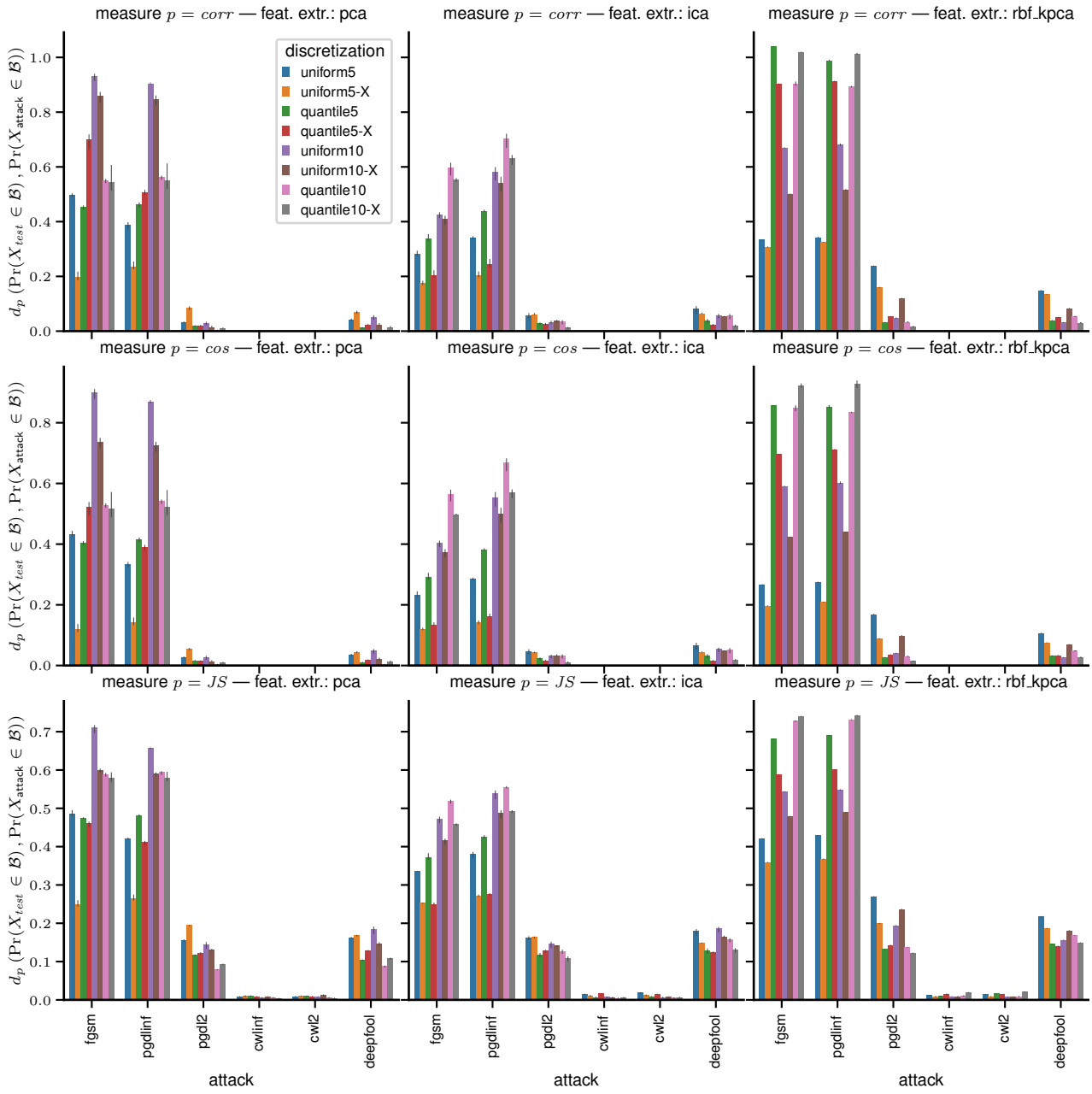

Figure 8: See Figure 7.

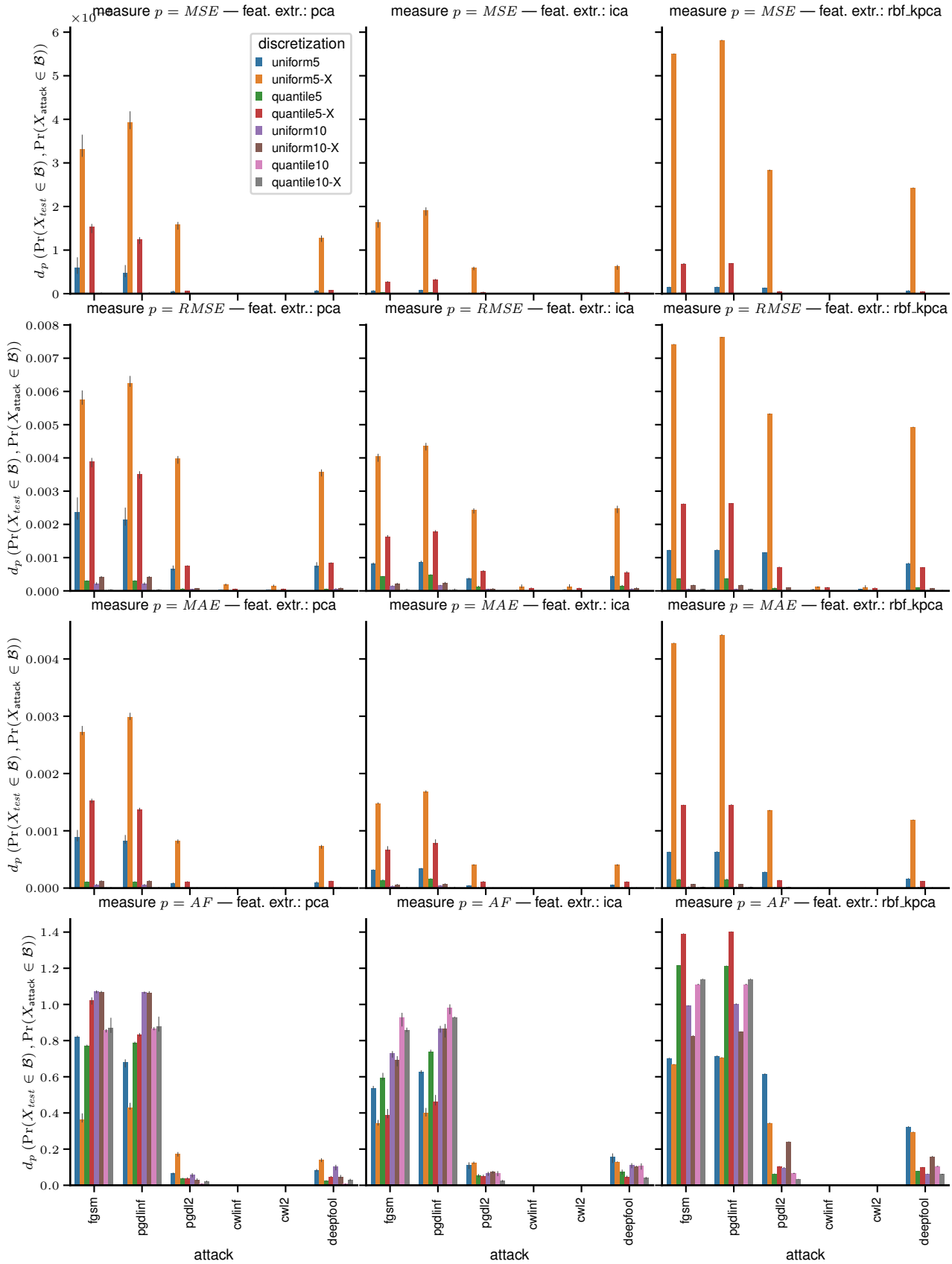

Figure 9: See Figure 7.

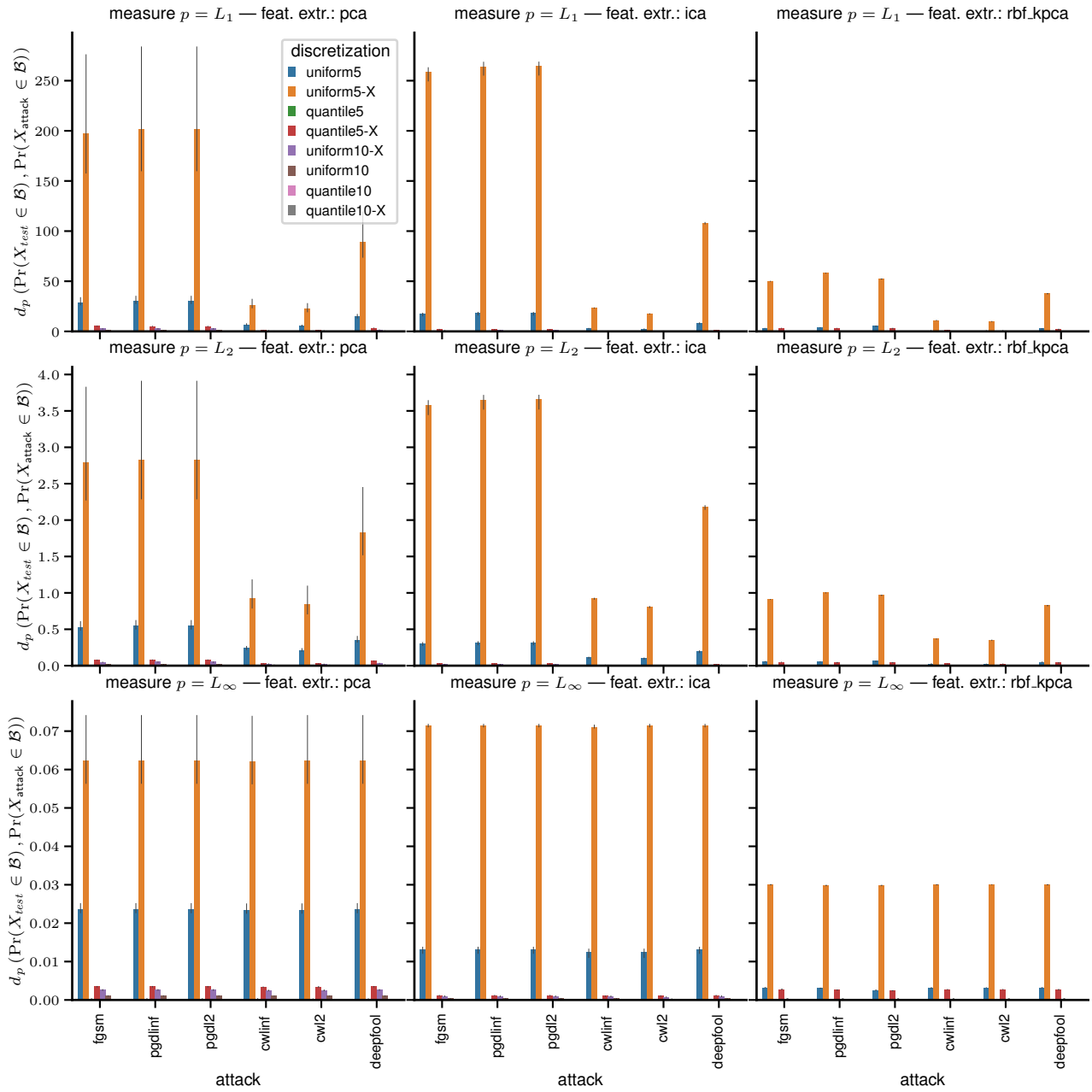

Figure 10: Distances (vertical axes) between probabilities obtained for the CIFAR10 validation dataset and datasets generated by selected adversarial attacks (horizontal axes). See Figure 7 for further details.

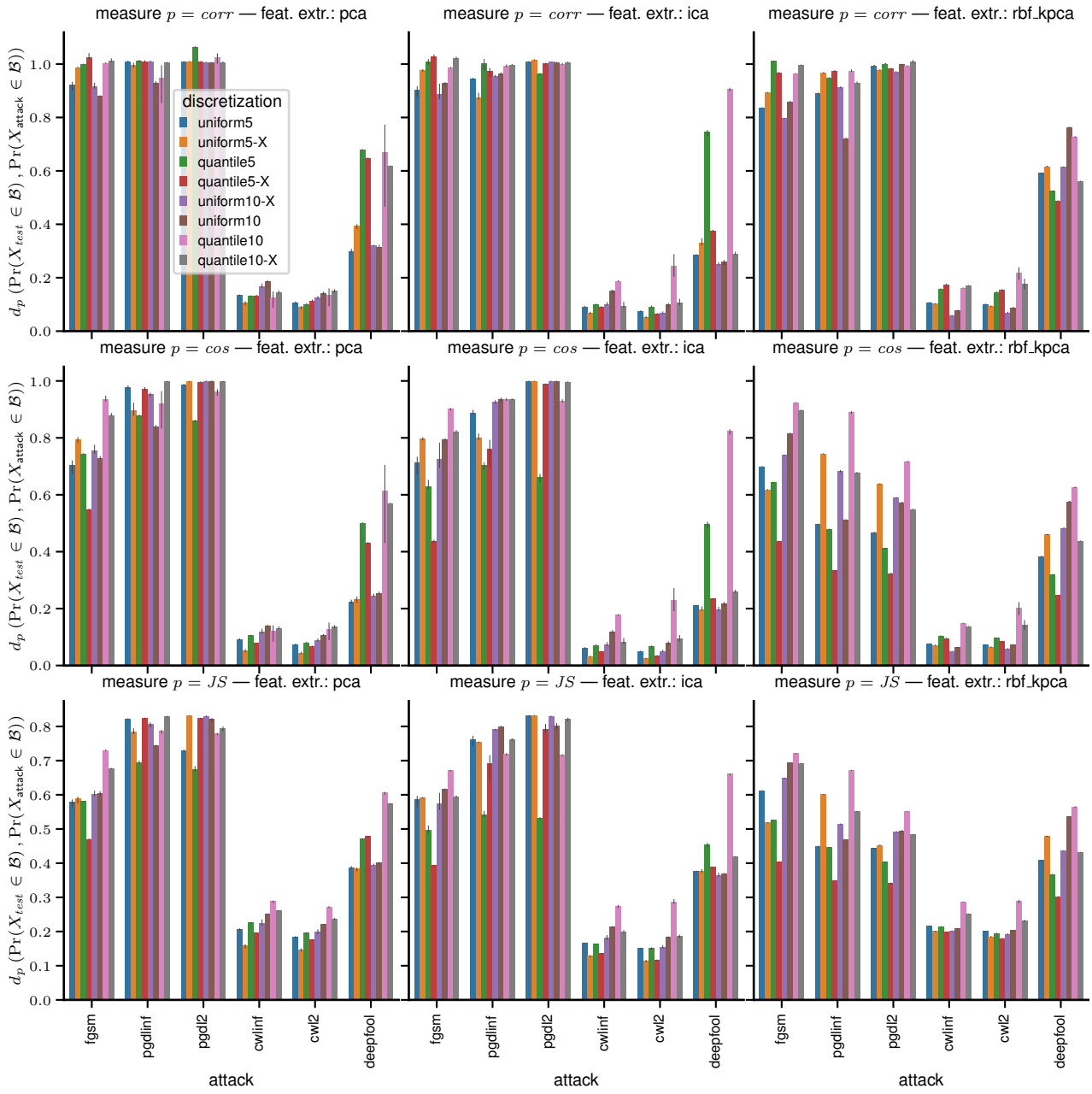

Figure 11: See Figure 10.

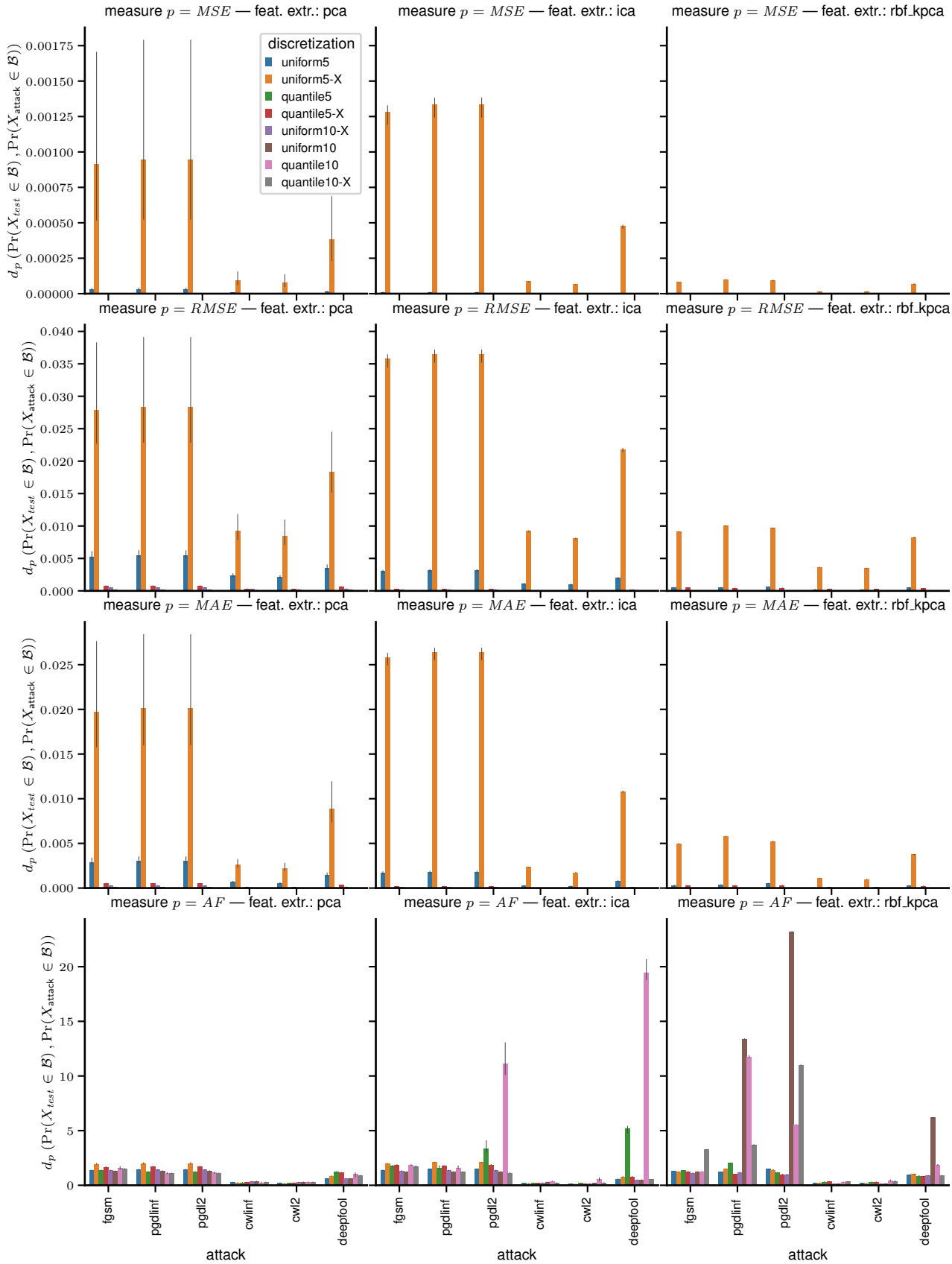

Figure 12: See Figure 10.