A Missing Details

A.1 Motivations for working with model latent space

In Section 3, we introduced the confusion density matrix that allows us to categorize suspicious examples at testing time. Crucially, this density matrix relies on kernel density estimations in the latent space $\mathcal{H}$ associated to the model f through Assumption 1. Why are we performing a kernel density estimation in latent space rather than in input space $\mathcal{X}$? The answer is fairly straightforward: we want our density estimation to be coupled to the model and its predictions.

Let us now make this point more rigorous. Consider two input examples $x_1, x_2 \in \mathcal{X}$. The model assigns a representations $g(x_1), g(x_2) \in \mathcal{H}$ and class probabilities $f(x_1), f(x_2) \in \mathcal{Y}$. If we define our kernel κ in latent space $\mathcal{H}$, this often means[1] that $\kappa[g(x_1), g(x_2)]$ grows as $\|g(x_1) - g(x_2)\|_{\mathcal{H}}$ decreases. Hence, examples that are assigned a similar latent representation by the model f are related by the kernel. Since our whole discussion revolves around model *predictions*, we would like to guarantee that two examples related by the kernel are given similar predictions by the model f. In this way, we would be able to interpret a large kernel density $\kappa[g(x_1), g(x_2)]$ as a hint that the predictions $f(x_1)$ and $f(x_2)$ are similar. We will now show that, under Assumption 1, such a guarantee exists. Similar to [36], we start by noting that

$$\begin{aligned}\|(l \circ g)(x_1) - (l \circ g)(x_2)\|_{\mathbb{R}^C} &= \|l\left[g(x_1) - g(x_2)\right]\|_{\mathbb{R}^C} \\ &\leq \|l\|_{\text{op}} \|g(x_1) - g(x_2)\|_{\mathcal{H}},\end{aligned}$$

where $\|\cdot\|_{\mathbb{R}^C}$ is a norm on $\mathbb{R}^C$ and $\|l\|_{\text{op}}$ is the operator norm of the linear map l. In order to extend this inequality to black-box predictions, we note that the normalizing map in Assumption 1 is often a Lipschitz function with Lipschitz constant $\lambda \in \mathbb{R}$. For instance, a Softmax function with inverse temperature constant λ^{-1} is λ-Lipschitz [37]. We use this fact to extend our inequality to predicted class probabilities:

$$\begin{aligned}\|f(x_1) - f(x_2)\|_{\mathcal{Y}} &= \|(\varphi \circ l \circ g)(x_1) - (\varphi \circ l \circ g)(x_2)\|_{\mathcal{Y}} \\ &\leq \lambda \|(l \circ g)(x_1) - (l \circ g)(x_2)\|_{\mathbb{R}^C} \\ &\leq \lambda \|l\|_{\text{op}} \|g(x_1) - g(x_2)\|_{\mathcal{H}}.\end{aligned}$$

This crucial inequality guarantees that examples $x_1, x_2 \in \mathcal{X}$ that are given a similar latent representation $g(x_1) \approx g(x_2)$ will also be given a similar prediction $f(x_1) \approx f(x_2)$. In short: two examples that are related according to a kernel density defined in the model latent space $\mathcal{H}$ are guaranteed to have similar predictions. This is the motivation we wanted to support the definition of the kernel κ in latent space.

An interesting question remains: is it possible to have similar guarantees if we define the kernel in input space? When we deal with deep models, the existence of adversarial examples indicates the opposite [38]. Indeed, if x_2 is an adversarial example with respect to x_1, we have $x_1 \approx x_2$ (and hence $\|x_1 - x_2\|_{\mathcal{X}}$ small) with two predictions $f(x_1)$ and $f(x_2)$ that are significantly different. Therefore, defining the kernel κ in input space might result in relating examples that are given a significantly different prediction by the model. For this reason, we believe that the latent space is more appropriate in our setting.

A.2 Details: Flagging IDM and Bnd Examples with Thresholds

In order to understand uncertainty, it will be clearer to map those scores into binary classes with thresholds. In our experiments, we use empirical quantiles as thresholds. e.g., to label an example as IDM, we specify an empirical quantile number q, and calculate the corresponding threshold based on the order statistics of IDM Scores for test examples: $S_{\text{IDM}}^{(1)}, ..., S_{\text{IDM}}^{(|\mathcal{D}_{\text{test}}|)}$, where $S_{\text{IDM}}^{(n)}$ denotes the n-th smallest IDM score out of $|\mathcal{D}_{\text{test}}|$ testing-time examples. Then, the threshold given quantile number q is

$$\tau_{\text{IDM}}(q) \equiv S_{\text{IDM}}^{(\lfloor|\mathcal{D}_{\text{test}}|\cdot q\rfloor)}.$$

[1] This is the case for all the kernels that rely on a distance (e.g. the Radial Basis Function Kernel, the Matern kernel or even Polynomial kernels [32]).

Similarly, we can define quantile-based threshold in flagging Bnd examples based on the order statistics of Bnd Scores for test examples, such that for given quantile q,

$$\tau_{\mathrm{Bnd}}(q) \equiv S_{\mathrm{Bnd}}^{(\lfloor|\mathcal{D}_{\mathrm{test}}|\cdot q\rfloor)}.$$

Practically, a natural choice of q is to use the validation accuracy: when there are $1-q$ examples misclassified in the validation set, we also expect the testing-time in distribution examples with the highest $1-q$ to be marked as Bnd or IDM examples.

B Improving Predicting Performance of Uncertain Examples

Knowing the category that a suspicious example belongs to, can we improve its prediction? For ease of exposition, we focus on improving predictions for $\mathcal{S}_{\mathrm{B\&I}}$.

Let $p\left(x \mid \mathcal{S}_{\mathrm{B\&I}}\right)$ be the latent density be defined as in Definition 1. We can improve the prediction performance of the model on $\mathcal{S}_{\mathrm{B\&I}}$ examples by focusing on the part of examples in the training set that are closely related to those suspicious examples. We propose to refine the training dataset $\mathcal{D}_{\mathrm{train}}$ by only keeping the examples that resembles the latent representations for the specific type of test-time suspicious examples, and train another model on this subset of the training data:

$$\tilde{\mathcal{D}}_{\mathrm{train}} \equiv \{x \in \mathcal{D}_{\mathrm{train}} | p\left(x \mid \mathcal{S}_{\mathrm{B\&I}}\right) \geq \tau_{\mathrm{test}}\}, \tag{6}$$

where τ_{test} is a threshold that can be adjusted to keep a prespecified proportion q of the related training data. Subsequently, new prediction model $f_{\mathrm{B\&I}}$ is trained on $\tilde{\mathcal{D}}_{\mathrm{train}}$.

Orthogonal to ensemble methods that require multiple models trained independently, and improve *overall* prediction accuracy by bagging or boosting, our method is targeted at improving the model's performance on a *specified* subclass of test examples by finding the most relevant training examples. Our method is therefore more transparent and can be used in parallel with ensemble methods if needed.

Threshold $\tau_{\mathrm{test}}(q)$ For every training example $x \in \mathcal{D}_{\mathrm{train}}$, we have the latent density $p(x|\mathcal{D}_{\mathrm{B\&I}})$ over the B&I class of the test set. With their order statistics $p_{(1)}(x|\mathcal{D}_{\mathrm{B\&I}}), ..., p_{(|\mathcal{D}_{\mathrm{train}}|)}(x|\mathcal{D}_{\mathrm{B\&I}})$. Given quantile number q, our empirical quantile based threshold τ_{test} is chosen as

$$\tau_{\mathrm{test}}(q) \equiv p_{(\lfloor q\cdot|\mathcal{D}_{\mathrm{train}}|\rfloor)}(x|\mathcal{D}_{\mathrm{B\&I}}).$$

During the inverse training time, we train our model to predict those B&I class of test examples only with the training data with higher density than $\tau_{\mathrm{test}}(q)$. We experiment with different choices of q in the experiment (Figure 6 in Sec. 4.4).

C Additional Experiments

C.1 Categorization of Uncertainty under Different Thresholds

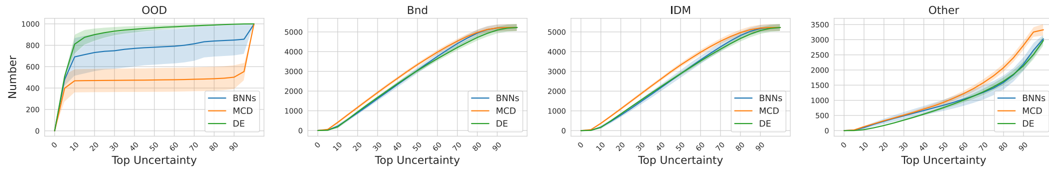
In the main text, we provide results with $\tau_{\mathrm{Bnd}} = \tau_{\mathrm{IDM}} = 0.8$, which approximates the accuracy on validation set—as a natural choice. In this section, we vary these thresholds and show in Figure 7 that changing those thresholds does not significantly alter the conclusions drawn above.

Figure 8 looks more closely into the top 25% uncertain examples for each method, and the accuracy on each of the uncertainty classes. As expected, the accuracy of the B&I examples is always lower than that of the trusted class, meaning that those examples are most challenging for the classifier. And the accuracy of flagged classes are always lower than the *other* class, verifying the proposed categorization of different classes.

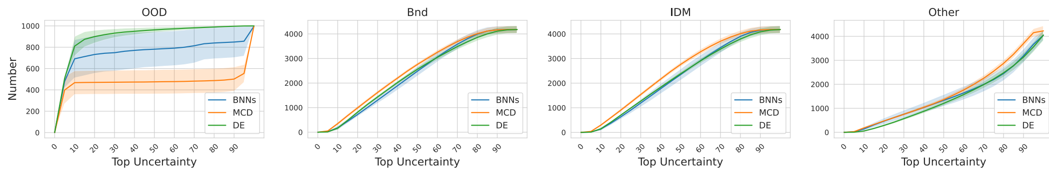
C.2 Inverse Direction: More Results

In the main text, we show the results on improving prediction performance on the B&I class with training example filtering (On the Covtype, Digits dataset). More results on other classes of examples are provided in this section.

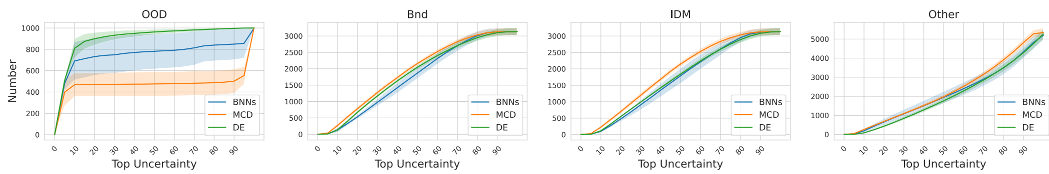
We experiment on three UCI datasets: **Covtype**, **Digits**, and **Spam**. And experiment with three classes we defined in this work:



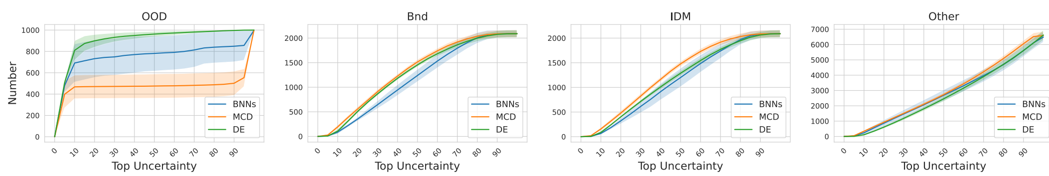
(a) $\tau_{\mathrm{BD}}(0.5), \tau_{\mathrm{IDM}}(0.5)$



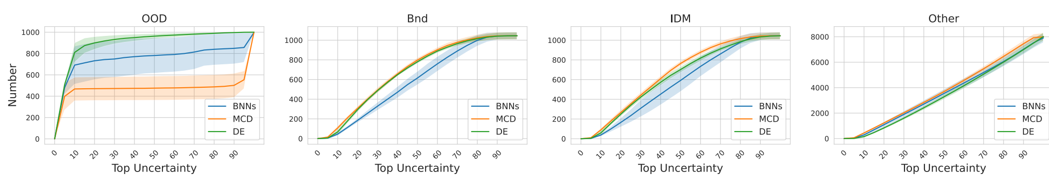
(b) $\tau_{\mathrm{BD}}(0.6), \tau_{\mathrm{IDM}}(0.6)$



(c) $\tau_{\mathrm{BD}}(0.7), \tau_{\mathrm{IDM}}(0.7)$



(d) $\tau_{\mathrm{BD}}(0.8), \tau_{\mathrm{IDM}}(0.8)$



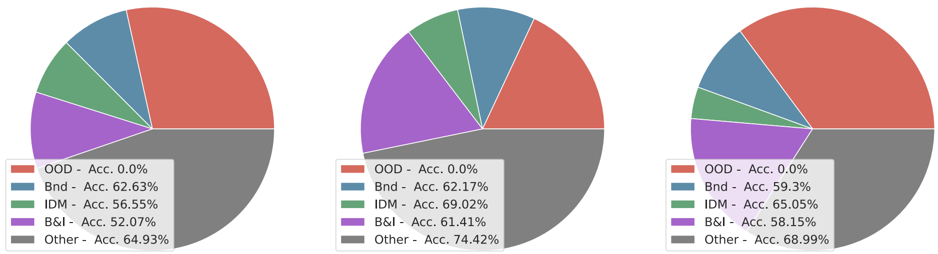
(e) $\tau_{\mathrm{BD}}(0.9), \tau_{\mathrm{IDM}}(0.9)$

Figure 7: Experiments on different choices of thresholds.

1. **B&I** class (Figure 9). As we have discussed in our main text, the prediction accuracy on the B&I class are always the lowest among all classes. By training with filtered examples in $\mathcal{D}_{\text{train}}$ rather than the entire training set, the B&I class of examples can be classified with a remarkably improved accuracy.

2. **Bnd** class (Figure 10). This class of examples are located at boundaries in the latent space of validation set, but not necessarily have been misclassified. Therefore, their performance baseline (training with the entire $\mathcal{D}_{\text{train}}$) is relatively high. The improvement is clear but not as much as on the other two classes.

3. **IDM** class (Figure 11). For this class of examples, similar mistakes have been make in the validation set, yet those examples are not necessarily located in the boundaries—the misclassification may be caused by ambiguity in decision boundary, imperfectness of either the model or the dataset. The primal prediction accuracy on this class of examples is lower than the Bnd class but higher than the B&I class, training with filtered $\mathcal{D}_{\text{train}}$ also clearly improve the performance on this class of examples.



(a) BNNs (b) MCD (c) DE

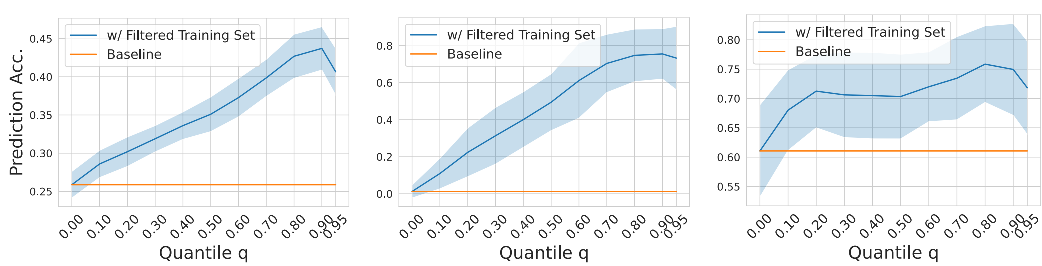
Figure 8: The top 25% uncertain examples identified by different methods. Legend of each figure provide the accuracy and proportion of each class. As the classifier can not make correct predictions on the OOD examples, it's always better for uncertainty estimators to flag more OOD examples.

Table 4: DAUC is not the only choice in identifying OOD examples. On the Dirty-MNIST dataset, DAUC, Outlier-AE and the IForest can identify most outliers in the test dataset. (Given threshold = 1.0 for those two benchmark methods).

Method	Precision	Recall	F1-Score
DAUC	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000
Outlier-AE	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000
IForest [40]	0.9998 ± 0.0004	1.0000 ± 0.0000	0.9999 ± 0.0002

C.3 Alternative Approach in Flagging OOD

As we have mentioned in the main text, although DAUC has a unified framework in understanding all three types of uncertainty the uncertain caused by OOD examples can also be identified by off-the-shelf algorithms. We compare DAUC to two existing outlier detection methods in Table 4, where all methods achieve good performance on the Dirty-MNIST dataset. Our implementation is based on Alibi Detect [39].

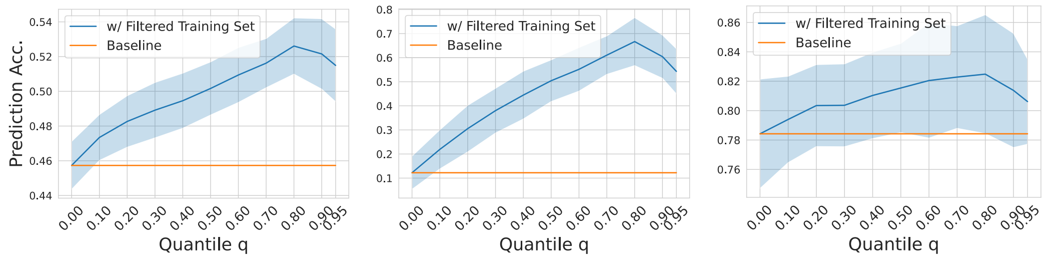


(a) Covtype (b) Digits (c) Spam

Figure 9: Experiments on the B&I class (reported in the main text).

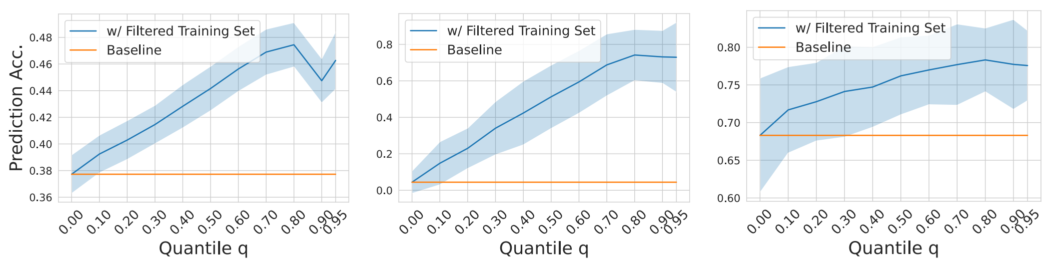
C.4 Experiments on Dirty-CIFAR-10

Dataset Discription In this experiment, we introduce a revised version of the CIFAR-10 dataset to test DAUC's scalability. Similar to the Dirty-MNIST datset [34], we use linear combinations of the latent representation to construct the "boundary" class. In the original CIFAR-10 Dataset, each of the 10 classes of objects has 6000 training examples. We split the training set into training set (40%), validation set (40%) and test set (20%). To verify the performance of DAUC in detecting OOD examples, we randomly remove one of those 10 classes (denoted with class-i) during training and manually concatenate OOD examples with the test dataset, with label i. In our experiment, we



(a) Covtype (b) Digits (c) Spam

Figure 10: Experiments on the Bnd class.



(a) Covtype (b) Digits (c) Spam

Figure 11: Experiments on the IDM class.

use 1000 MNIST digits as the OOD examples, with zero-padding to make those digits share the same input shape as the CIFAR-10 images. Combining those boundary examples, OOD examples and the vanilla CIFAR-10 examples, we get a new benchmark, dubbed as Dirty-CIFAR-10, for quantitative evaluation of DAUC.

Quantify the performance of DAUC on Dirty-CIFAR-10 Quantitatively, we evaluate the performance of DAUC in categorizing all three classes of uncertain examples. Results of averaged performance and standard deviations based on 8 repeated runs are provided in Table 5.

Table 5: Quantitative results on the Dirty-CIFAR-10 dataset. DAUC scales well and is able to categorize all three classes of uncertain examples.

Category	Precision	Recall	F1-Score
OOD	0.986 ± 0.003	0.959 ± 0.052	0.972 ± 0.027
Bnd	0.813 ± 0.002	0.975 ± 0.000	0.887 ± 0.001
IDM	0.688 ± 0.041	0.724 ± 0.017	0.705 ± 0.027

Categorize Uncertain Predictions on Dirty-CIFAR-10 Similar to Sec. 4.3 and Figure 5, we can categorize uncertain examples flagged by BNNs, MCD and DE using DAUC—see Figure 12. We find that in the experiment with CIFAR-10, DE tends to discover more OOD examples as top uncertain examples. Differently, although BNNs flags less OOD examples as top-uncertain, it continuously discover those OOD examples and is able to find most of them for the top 50% uncertainty. On the contrary, MCD performs the worst among all three methods, similar to the result drawn from the DMNIST experiment. On the other hand, while BNN is good at identifying OOD examples, it flags less uncertain examples in the Bnd and IDM classes. DE is the most apt at flagging both Bnd and IDM examples, and categorizes far less examples into the *Other* class. **These observations are well aligned with the experiment results we had with DMNIST in Sec. 4.3, showing the scalability of DAUC to large-scale image dataset.**

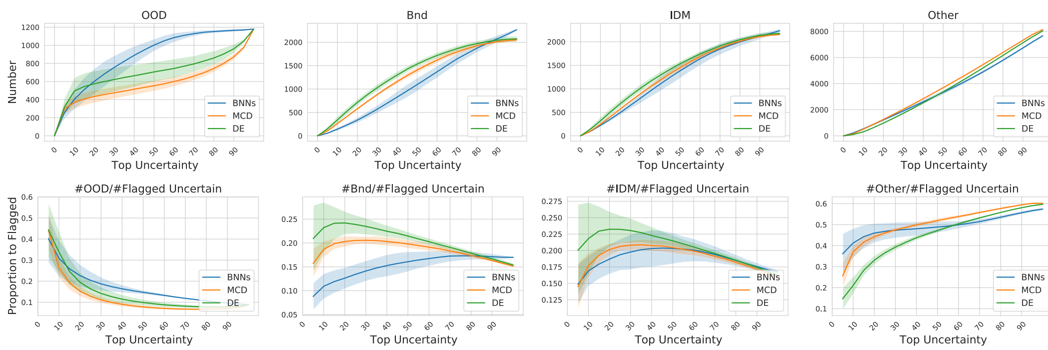


Figure 12: Experiments on the CIFAR-10 dataset. Results of applying DAUC in categorizing different uncertainty estimation methods. First row: comparisons on the numbers in different classes of examples. Second row: comparisons on the proportion of different classes of flagged examples to the total number of identified uncertain examples. Different methods tend to identify different certain type of uncertain examples.

D Implementation Details

D.1 Code

Our code is anonymously available at https://anonymous.4open.science/r/DAUC-B234/.

D.2 Hyperparameters

D.2.1 Bandwidth

In our experiments, we use (z-score) normalized latent representations and bandwidth 1.0. In the inverse direction, as the sample sizes are much smaller, a bandwidth of 0.01 is used as the recommended setting. There is a vast body of research on selecting a good bandwidth for Kernel Density Estimation models [41–43] and using these to adjust DAUC's bandwidth to a more informed choice may further improve performance.

D.3 Inverse Direction: Quantile Threshold q

As depicted in Appendix A.2, a natural choice of q is to use the validation accuracy. We use this heuristic approach in our experiments for the inverse direction.

D.4 Model Structure

In our experiments, we implement **MCD** and **DE** with 3-layer-CNNs with ReLU activation. Our experiments on **BNNs** are based on the IBM UQ360 software [44]. More details of the convolutional network structure are provided in Table 6.

Table 6: Network Structure

Layer	Unit	Activation	Pooling
Conv 1	$(1, 32, 3, 1, 1)$	ReLU()	MaxPool2d(2)
Conv 2	$(32, 64, 3, 1, 1)$	ReLU()	MaxPool2d(2)
Conv 3	$(64, 64, 3, 1, 1)$	ReLU()	MaxPool2d(2)
FC	$(64 \times 3 \times 3, 40)$	ReLU()	-
Out	$(40, N_{\text{Class}})$	SoftMax()	-

D.5 Implementation of Kernel Density Estimation and Repeat Runs

Our implementation of KDE models are based on the sklearn's KDE package [45]. Gaussian kernels are used as default settings. In all experiments, we run with 10 random seeds and report the averaged results. In our experiments, we find using different kernels in density estimation provides highly correlated scores. We calculate the Spearman's ρ correlation between scores DAUC gets over 5 runs with Gaussian, Tophat, Exponential kernels under the same bandwidth. Changing the kernel brings highly correlated scores (all above $\mathbf{0.86}$) for DAUC and, hence, has minor impact on DAUC's performance. We preferred KDE since the latent representation is relatively low-dimensional. We found that a low-dim latent space (e.g., 10) works well for all experiments (including CIFAR-100).

D.6 Hardware

All results reported in our paper are conducted with a machine with 8 Tesla K80 GPUs and 32 Intel(R) E5-2640 CPUs. The computational cost is mainly in density estimation, and for low-dim representation space, such an estimation can be efficient: running time for DAUC on the Dirty-MNIST dataset with KDE is approximately 2 hours.

Assumptions and Limitations

In this work, we introduced the confusion density matrix that allows us to categorize suspicious examples at testing time. Crucially, this density matrix relies on kernel density estimations in the latent space $\mathcal{H}$ associated to the model f through Assumption 1. We note this assumption generally holds for most modern uncertainty estimation methods.

While the core contribution of this work is to introduce the concept of confusion density matrix for uncertainty categorization, the density estimators leveraged in the latent space can be further improved. We leave this to the future work.

Broader Impact

While previous works on uncertainty quantification (UQ) focused on the discovery of uncertain examples, in this work, we propose a practical framework for categorizing uncertain examples that are flagged by UQ methods. We demonstrated that such a categorization can be used for UQ method selection — different UQ methods are good at figuring out different uncertainty sources. Moreover, we show that for the inverse direction, uncertainty categorization can improve model performance.

With our proposed framework, many real-world application scenarios can be potentially benefited. e.g., in Healthcare, a patient marked as uncertain that categorized as OOD — preferably identified by Deep Ensemble, as we have shown — should be treated carefully when applying regular medical experience; and an uncertain case marked as IDM — preferably identified by MCD — can be carefully compared with previous failure cases for a tailored and individualized medication.