International Journal of Computer Vision (2026) 134:185
https://doi.org/10.1007/s11263-026-02794-3

GPify: Leveraging the Combined Strength of Normalizing Flow and Softmax For an Out-of-Distribution aware Confidence Score

Simon Kristoffersson Lind[1] · **Rudolph Triebel**[2,3] · **Volker Krüger**[1]

Received: 12 March 2025 / Accepted: 18 February 2026 / Published online: 9 March 2026

Abstract

In order for any learning-based model to be considered reliable, it needs a well-behaved uncertainty or confidence estimate. Most modern neural networks do produce a confidence estimate in the form of their softmax output probability. However, the softmax probability is invalid for out-of-distribution data. Gaussian processes are known to produce a well-behaved confidence estimate that is aware of out-of-distribution samples. Inspired by Gaussian processes, we propose GPify, which combines the softmax probability with a Normalizing Flow in order to add out-of-distribution awareness to the confidence estimate from a neural network. The resulting confidence from GPify is an uncertainty measure that is interpretable and intuitive, while also being probabilistically sound. We evaluate GPify in a selective classification framework, and conclude that it achieves comparable performance to state-of-the-art methods. In addition, we show that GPify has capabilities for detecting adversarial examples, which is a direct improvement over softmax confidence.

Keywords Uncertainty · Computer Vision · Normalizing Flows · Softmax

1 Introduction

It is well known that neural networks perform unreliably when an input is too dissimilar from the data it was trained on (Yang et al., 2024). In order to reason about reliability a model needs introspective capabilities, specifically a well-behaved confidence or uncertainty estimate (Grimmett et al., 2016). Confidence and uncertainty are fundamentally equivalent in this context, and therefore we discuss them interchangeably throughout this work.

Gaussian Processes (GP) (Rasmussen & Williams, 2005) are considered to be one of the most reliable models, in part due to their ability to produce a well-behaved and probabilistically sound confidence estimate. However, due to their non-parametric nature, GPs fail to scale to the data quantities in modern learning applications. With this work, we propose *GPify*: a simple method of combining the softmax confidence from a neural network with probability estimates from a Normalizing Flow (NF) in a way that mimics the confidence estimates from a GP. The resulting confidence score is probabilistically sound and interpretable like a GP, but it *a)* successfully scales to larger applications, and *b)* is easy to apply with any neural network.

Uncertainty is commonly discussed in the form of aleatoric and epistemic uncertainty (Kristoffersson Lind et al., 2024). In the context of classification problems, aleatoric uncertainty refers to the inherent ambiguity between classes. As such, the output confidence from a softmax operation reflects a model's aleatoric uncertainty. Epistemic uncertainty, however, refers to gaps in the training data, where a model cannot make accurate predictions because it was never trained on similar samples. Since most neural network architectures use a simple classifier with linear decision boundaries, they do not inherently reflect epistemic uncertainty. These two types of uncertainty are typically measured and evaluated in entirely separate problem settings. Aleatoric uncertainty is commonly evaluated in the context of calibration (Guo et al., 2017), and epistemic uncertainty is typically evaluated in the

Communicated by Gunhee Kim.

✉ Simon Kristoffersson Lind
simon.kristoffersson_lind@cs.lth.se

Rudolph Triebel
rudolph.triebel@dlr.de

Volker Krüger
volker.krueger@cs.lth.se

[1] Lund University, Lund, Sweden

[2] German Aerospace Center, Oberpfaffenhofer, Germany

[3] Karlsruhe Institute of Technology, Karlsruhe, Germany

context of out-of-distribution (OOD) detection (Yang et al., 2024).

Selective classification is a problem that extends classification with an option to reject a data sample (El-Yaniv & Wiener, 2010). Here, classification is the primary task, and rejection is presented as a mechanism to improve accuracy on the remaining samples. Kim et al. (2023) propose *unknown detection* as a selective classification problem where out-of-distribution samples are included among the inputs. The inclusion of both in- and out-of-distribution samples implies the presence of both aleatoric and epistemic uncertainty. *Softmax Information Retaining Combination* (SIRC) (Xia & Bouganis, 2024) is a recent approach that combines an aleatoric and epistemic confidence into a single score. SIRC is shown to perform well on the unknown detection problem. Xia and Bouganis (2024) coin the acronym SCOD for *Selective Classification in the presence of Out-of-distribution Data*, as a synonym to unknown classification. In this work we also adopt the term SCOD, since it better emphasizes classification as the primary task.

Our work is undoubtedly similar to SIRC (Xia & Bouganis, 2024). However, instead of focusing directly on maximizing performance within the SCOD framework, we focus primarily on designing GPify to be intuitive and probabilistically sound, which makes it more desirable for end-users in real-world applications. Nonetheless, we compare GPify against SIRC within the SCOD problem setting, to evaluate its performance.

In addition to our SCOD evaluation, we conduct a small-scale experiment to evaluate whether GPify has the capacity to detect adversarial examples.

To summarize, our contributions are as follows:

- We propose GPify, which combines the confidence scores from a softmax classifier with the likelihood from a Normalizing Flow, to create a single confidence score that captures both aleatoric and epistemic uncertainty.
- We evaluate GPify in the SCOD problem setting, and show that GPify offers highly competitive performance.
- We evaluate GPify with different neural network architectures, to verify that it generalizes to different architectures.
- In an additional, minor experiment, we show that GPify is capable of detecting adversarial examples, though a more thorough experiment is needed to evaluate how GPify compares to dedicated adversarial detection methods.

2 Related Work

Out-of-distribution detection is an active research area with the aim of classifying data samples as in- or out-of-distribution (Huang et al., 2024; Wang et al., 2022; Yang et al., 2024). The general problem of OOD detection is not necessarily concerned with the reliability of neural networks, but the task of discriminating between different data distributions is fundamentally the same. If we place OOD detection in the context of uncertainty estimation, it is only capable of representing aleatoric uncertainty. As such, OOD detection methods tend to underperform in the SCOD problem setting, as shown in (Kim et al., 2023).

Complementary to OOD detection, there exists a body of work focused on improving *calibration* (Guo et al., 2017; Wenger et al., 2020), which is a term used to describe how accurate a model's confidence score is. In this context, the accuracy of a confidence score is determined by how well it reflects the probability of a sample being correctly classified (Guo et al., 2017). As such it is only defined on in-distribution data, and only reflects epistemic uncertainty.

To the best of our knowledge, SIRC (Xia & Bouganis, 2024) is the first work to explicitly construct a combined confidence score that incorporates both aleatoric and epistemic uncertainty. However, as pointed out in (Xia & Bouganis, 2024), there exists other methods that can be interpreted as a combination score, most notably GradNorm (Huang et al., 2024) and ViM (Wang et al., 2022). While both of these methods are framed as OOD detection methods, GradNorm can be interpreted as a combination of softmax and the norm of a feature vector (Huang et al., 2024), and ViM can be interpreted as a combination of logits and the residual of a feature vector (Wang et al., 2022). Neither of these combination scores produce values that are easily interpretable or probabilistically sound.

There has been some work focused on modifying Gaussian Processes to improve their performance, and reduce their computational load on larger learning problems (Blomqvist et al., 2020; Wilk et al., 2017). Despite these efforts, GPs are still significantly more computationally demanding, and perform significantly worse than neural network-based solutions. While we use GPs as a reference point for their probabilistically sound confidence, we refrain from comparing against them empirically as we aim to construct a well-behaved confidence without degrading the performance of neural networks.

3 Background

3.1 Uncertainty, Calibration, and OOD Detection

There exists two, largely separate bodies of research in the domain of uncertainty estimation, namely those that focus on aleatoric and epistemic uncertainty. Aleatoric uncertainty refers to uncertainty that is inherent in the data, for example class ambiguities or measurement noise (Kristoffersson Lind et al., 2024). As such, aleatoric uncertainty cannot be reduced

or removed by adding more data. Epistemic uncertainty, on the other hand, refers to uncertainty stemming from missing data. As an example, consider an animal classifer that is trained on pictures of japanese spitz's, and arctic foxes. For such a classifier, aleatoric uncertainty stems from the fact that many Japanese spitz's look similar to arctic foxes, and vice versa. Conversely, a picture of a penguin would constitute epistemic uncertainty.

Uncertainty is divided into these two categories because they answer fundamentally different questions (Kristoffersson Lind et al., 2024). One way to view these two uncertainties is as probability distributions, where aleatoric uncertainty has the goal of modelling $p(class \mid \mathbf{x})$, as is a common interpretation of a softmax classifier (Xia & Bouganis, 2024). In this view, epistemic uncertainty aims to model $p(\mathbf{x})$, either directly or indirectly (Yang et al., 2024).

Aleatoric uncertainty is typically evaluated using the Expected Calibration Error (ECE) metric (Guo et al., 2017), which aims to directly evaluate the error in $p(class \mid \mathbf{x})$:

$$\text{ECE} = \sum_m \frac{|B_m|}{|n|} \left|\text{accuracy}(B_m) - \text{confidence}(B_m)\right| .$$

Here, B_m represents equidistant bins created from the testing data based on the predicted confidence for each sample, and n represents total number of testing samples.

We argue that ECE is entirely invalid in the presence of OOD samples. Since any prediction made on OOD data is invalid, the accuracy is 0, and any non-zero confidence contributes to increasing the ECE metric.

Epistemic uncertainty is commonly evaluated in the OOD detection setting, using an AUROC score based on the detector's ability to discriminate between in- and out-of-distribution data samples (Yang et al., 2024). In addition to AUROC, OOD detectors are often evaluated based on their successful OOD detection at a fixed false-negative-rate, where a detection threshold is set such that a fixed percentage of in-distribution samples are rejected as OOD. As OOD detection is a classification problem itself, which treats the in-distribution as a singular class, it fundamentally cannot evaluate aleatoric uncertainty.

3.2 Adversarial Examples

First coined by Szegedy et al. (2014), the term *Adversarial Example* refers to images that have been perturbed at a pixel-level to cause a misclassification. By constructing an optimization problem

$$\min ||\delta||_2 , \quad \text{s.t. } f(\mathbf{x} + \delta) \neq y_{\text{true}}$$

they are able to generate images that look unchanged to the human eye, but are misclassified by a neural network. Here, we use $f(x)$ to denote the predicted label from a classifier f, and y_{true} to denote the true class label. Since these examples cause a misclassification in the softmax output, they cannot be detected using softmax probabilities alone.

Goodfellow et al. (2015) improve the process of creating adversarial examples, resulting in the *Fast Gradient Sign Method* (FGSM):

$$\mathbf{x}_{\text{adversarial}} = \mathbf{x} + \epsilon \, \text{sign}(\nabla_x \mathcal{L}(f(\mathbf{x}), y_{\text{true}})).$$

Here $\mathbf{x}$ is directly modified in the direction of the gradient in order to maximize the loss function $\mathcal{L}$. FGSM is further modified by Kurakin et al. (2017), who perform the same update on $\mathbf{x}$ in several steps:

$$\begin{aligned} \mathbf{x}_{t+1} &= \mathbf{x}_t + \alpha \, \text{sign}(\nabla_x \mathcal{L}(f(\mathbf{x}_t), y_{\text{true}})) \\ \mathbf{x}_0 &= \mathbf{x} \ , \end{aligned}$$

which they call the *Basic Iterative Method*. In this method α refers to some small constant, similar to the learning rate in gradient descent.

3.3 Gaussian Processes

We derive GPify by imitating GPs, and therefore we cover the necessary prerequisites to understand their uncertainty estimation, and why GPs are intractable for learning at a larger scale. For a complete treatment, we refer to (Rasmussen & Williams, 2005).

A GP is a non-parametric learning method that can approximate arbitrary functions, given some basic assumptions of smoothness. Contrary to neural networks, and most other parametric methods, the function approximation learned by a GP is probabilistic, and thus non-deterministic (Rasmussen & Williams, 2005).

Predictions from a GP take the form of a Gaussian distribution:

$$y_{\text{pred}} \sim \mathcal{N}(\mu, \sigma) \ .$$

where the parameters μ and σ are defined using a mean function $m(\mathbf{x})$, and a covariance function $k(\mathbf{x}, \mathbf{x}')$. The mean function is most commonly defined to be a constant $\mathbf{0}$. While there are many different forms for the covariance function, they are all constructed around the idea that the covariance should be tied to the distance between points such that points that are closely clustered should have high covariance. To illustrate, one of the most common choices for k is the squared exponential:

$$k(\mathbf{x}, \mathbf{x}') = e^{-\frac{1}{2}|\mathbf{x}-\mathbf{x}'|^2} \ .$$

Let X and Y form the GP's training data, where $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ and $Y = [y_1, \ldots, y_n]^\intercal$. In order to model aleatoric uncertainty, Y is assumed to be subject to Gaussian noise:

$$y_i \sim \mathcal{N}\left(y_i^{\text{true}}, \epsilon\right) ,$$

where ϵ is usually found by maximum likelihood estimation. Then the mean and variance for a prediction at some unseen point $\mathbf{x}^*$ is:

$$\mu = m(\mathbf{x}^*) + k(\mathbf{x}^*, X)\left[k(X, X) + \epsilon^2\mathbf{I}\right]^{-1}(Y - m(X)),$$
$$\sigma = k(\mathbf{x}^*, \mathbf{x}^*) - k(\mathbf{x}^*, X)\left[k(X, X) + \epsilon^2\mathbf{I}\right]^{-1}k(X, \mathbf{x}^*) .$$

Where we take $m(X)$ and $k(X, X)$ to denote element-wise and pair-wise applications of the functions $m(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}')$.

Usually, the predicted function value is taken to be μ, and the uncertainty is taken to be σ. Importantly, σ explicitly models both epistemic and aleatoric uncertainty.

Here we also want to direct the reader's attention to the term $\left[k(X, X) + \epsilon^2\mathbf{I}\right]^{-1}$, which is the inverse of an $n \times n$ covariance matrix. Even with algorithmic optimizations, this inversion scales in the order of $O(n^2)$ (Gardner et al., 2018), which makes GPs intractable for the quantities of data used in modern learning problems.

3.4 SIRC and the SCOD Problem Setting

In order to unify the evaluation of aleatoric and epistemic uncertainty measures, Kim et al. (2023) propose the unknown detection (or SCOD, in (Xia & Bouganis, 2024)) problem setting. Specifically, they do this by extending the selective classification setting (El-Yaniv & Wiener, 2010) with OOD data. For any trained classifier, the distribution made up by its training dataset constitutes its *in-distribution* (ID). Any data sample that comes from a different distribution is by definition out-of-distribution (OOD). Xia and Bouganis (2024) formalize the SCOD problem setting by constructing a two-way discrimination problem. They subdivide the ID dataset into disjoint sets $\text{ID}_\checkmark$ and $\text{ID}_\times$ containing the correctly and incorrectly classified examples respectively. AUROC scores are then calculated separately for the discriminative tasks $\text{ID}_\checkmark$ vs. $\text{ID}_\times$, and $\text{ID}_\checkmark$ vs. OOD. They argue that, in practical applications, an incorrectly classified sample incurs a similar cost as an OOD sample, and therefore it makes sense to detect both separately.

3.4.1 SIRC

Within the SCOD problem setting, Xia and Bouganis (2024) propose Softmax Information Retaining Combination (SIRC). SIRC is based on the idea that a softmax classifier is able to reflect aleatoric uncertainty, but not epistemic uncertainty. In other words, softmax is expected to perform well on $\text{ID}_\checkmark$ vs. $\text{ID}_\times$ discrimination, but not on $\text{ID}_\checkmark$ vs. OOD discrimination. Their idea with SIRC is then to combine softmax with another confidence score that performs better on $\text{ID}_\checkmark$ vs. OOD discrimination.

Mathematically, SIRC is defined on two scores s_1 and s_2, where s_1 is expected to perform well on $\text{ID}_\checkmark$ vs. $\text{ID}_\times$ discrimination, and s_2 is expected to perform well on $\text{ID}_\checkmark$ vs. OOD detection:

$$SIRC(s_1, s_2) = -(s_1^{max} - s_1)\left(1 + e^{-b(s_2 - a)}\right) .$$

Here, s_1^{max} denotes a known upper bound for the s_1 score. a and b are parameters that change the shape and location of the function surface. Default values are

$$a = \mu_{s_2} - 3\sigma_{s_2}, \text{ and } b = \frac{1}{\sigma_{s_2}},$$

where μ_{s_2} and σ_{s_2} are the empirical mean and standard deviation of s_2 measured on ID data, either from training or validation data.

3.4.2 Confidence Scores used in SIRC

In their experiments, Xia and Bouganis (2024) evaluate different score combinations for s_1 and s_2. For s_1, they use Maximum Softmax Probability (MSP) and negative softmax entropy ($\mathcal{H}$):

$$\text{MSP} = \max_i \pi_i, \text{ and } \mathcal{H} = \sum_i \pi_i \log \pi_i,$$

where π_i denotes the output softmax probability for class i.

All s_2 scores used in SIRC operate on feature vectors $\mathbf{z}$, where $\mathbf{z}$ is taken to be the input vector to the final hidden layer of the model. The specific scores used in SIRC are the L_1-norm $||\mathbf{z}||_1$, the Residual score $||\mathbf{z}^{P^\perp}||_2$ (Wang et al., 2022), and the L_2-normalized distance to the kth nearest neighbor of $\mathbf{z}$ from the training data.

3.5 Normalizing Flows

In our work we use NFs to supplement the softmax with epistemic uncertainty. NFs are a class of models, and we use a specific instance in this class, called *coupling layers* (Papamakarios et al., 2021). Here we give an introduction to NFs, focusing specifically on coupling layers. For a complete treatment of NFs, we refer to (Papamakarios et al., 2021).

When reasoning about epistemic uncertainty, we are interested in a model's in-distribution. Specifically, this in-

distribution is made up of the model's training data $p_{\text{train}}(\mathbf{x})$. Normalizing Flows allow us to directly learn $p_{\text{train}}(\mathbf{x})$ and provides us with ways to sample and evaluate it.

A Normalizing Flow is an invertible and differentiable bijective function, also called a *diffeomorphism* (Papamakarios et al., 2021). It is often interpreted as a transformation between two vector-spaces. Given a sample $\mathbf{x} \sim p_{\text{train}}(\mathbf{x})$, a Normalizing Flow is a transformation T such that:

$$\mathbf{u} = T^{-1}(\mathbf{x}) \sim p_u(\mathbf{u}) \ .$$

T being a diffeomorphism allows us to directly compute the probability density at $\mathbf{x}$ (Papamakarios et al., 2021):

$$\begin{aligned} p_{\text{train}}(\mathbf{x}) &= p_u(\mathbf{u}) \left|\det J_T(\mathbf{u})\right|^{-1} \\ &= p_u(T^{-1}(\mathbf{x})) \left|\det J_{T^{-1}}(\mathbf{x})\right| \end{aligned}$$

where J_T and $J_{T^{-1}}$ denote the Jacobians of T and T^{-1} respectively. Finally, making T a learnable function allows us to define $p_u(\mathbf{u})$ freely. The most common choice is a standard Gaussian:

$$p_u(\mathbf{u}) = \mathcal{N}(0, 1) \ .$$

While it may be tempting to construct T as a regular neural network, such models typically do not fulfill the requirements of being both invertible and differentiable. Even in models that do, it is generally intractable to calculate the determinant of the Jacobian matrix. As such, some care must be taken to ensure that T is invertible, differentiable, and has a tractable Jacobian determinant. Coupling layers solve all three problems by splitting the input vector $\mathbf{x}$ into two disjoint parts $\mathbf{x}_1$, $\mathbf{x}_2$. This split can be constructed arbitrarily, but is commonly performed by taking $\mathbf{x}_1$ and $\mathbf{x}_2$ to be the first and second halves of $\mathbf{x}$. One part of $\mathbf{x}$ is then transformed while leaving the other fixed:

$$\mathbf{x}'_1 = \mathbf{x}_1 \circ \alpha(\mathbf{x}_2) + \beta(\mathbf{x}_2)$$

where we use $\circ$ to denote element-wise multiplication. The final output $\mathbf{x}'$ is constructed by concatenating $\mathbf{x}'_1$ and the unmodified $\mathbf{x}_2$. Constructing T in this way results in a triangular Jacobian, for which the determinant is just a product along the diagonal:

$$\left|\det J_T(\mathbf{x})\right| = \prod_i \alpha_i$$

where α_i are the elements of $\alpha(\mathbf{x}_2)$. Powerful flow models can be constructed by applying several coupling layers in sequence, while alternating which partition of $\mathbf{x}$ is being modified.

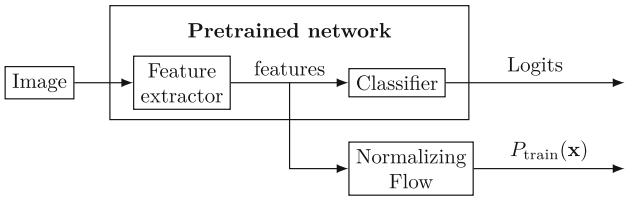

Fig. 1 Illustration of how we apply NFs to features from a pretrained network. When working with images, the feature extractor will typically be a Convolutional network or a Transformer architecture

3.5.1 Normalizing Flows Applied to Features

Kirichenko *et al.* (Kirichenko et al., 2020) observe that NFs underperform in terms of OOD detection when applied directly to images. Their results indicate that NFs tend to learn simple pixel-correspondences. To prevent such behaviour they propose to instead apply NFs to features extracted from a pretrained neural network as illustrated in Fig. 1 (Kirichenko et al., 2020).

As an added benefit, applying NFs to pretrained features allows the NF itself to be significantly smaller, which reduces both training and inference times. This structure also makes NFs a prime candidate for adding epistemic uncertainty to an existing neural network.

4 GPIFY: derivation

Here we discuss the idea behind GPify through intuitive arguments with a series of increasingly complex examples. Since GPs compute an intuitive and probabilistically grounded confidence estimate, we aim to design GPify in a way that mimics the confidence estimate from a GP. We begin with a 1-dimensional 2-class toy problem, where we derive our GPify formula by comparison with a GP. Afterwards, we continue by comparing GPify to a GP on a 2-dimensional 3-class toy problem, and finally on MNIST (Lecun et al., 1998).

4.1 Turning NF Densities into Probabilities

By using a Normalizing Flow as our model for GPify, we already have a strong probabilistic foundation. However, by default our NF outputs a log-likelihood $\ln p_{\text{train}}(\mathbf{x})$, which is unbounded and difficult to interpret for an end-user. Even if we remove the logarithm, $p_{\text{train}}(\mathbf{x})$ represents a probability density, which is still unintuitive. Instead, we aim to convert the log-likelihood directly into a probability by the means of an empirical distribution. Given a set of ID samples $S = \{\mathbf{x}_i \mid i = 1 \ldots n\}$ we compute:

$$\Pr(\mathbf{x}) = \frac{|\{\mathbf{x}_i \mid \ln p_{\text{train}}(\mathbf{x}_i) \leq \ln p_{\text{train}}(\mathbf{x}),\ \mathbf{x}_i \in S\}|}{|S|}$$

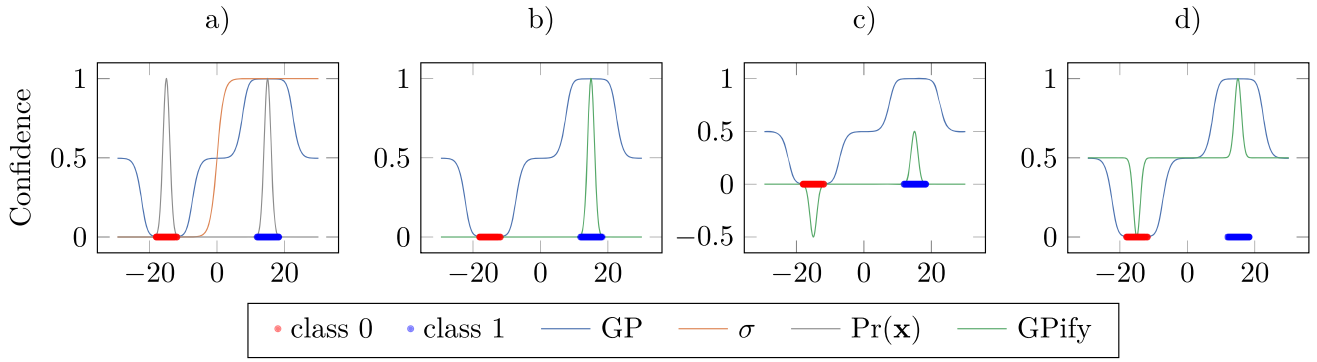

Fig. 2 Various stages of our derivation of GPify applied to our 1-dimensional 2-class toy problem. These plots display each respective model's confidence in class=1. **a)** Shows confidences from a GP along with an unmodified sigmoid, $\sigma(\mathbf{x})$, and unmodified $\Pr(\mathbf{x})$ from our NF. **b)** Shows how we can select the mode representing class=1 by $\sigma(\mathbf{x}) \cdot \Pr(\mathbf{x})$. **c)** Shows how we can represent lower confidence for class=0 by $\left(\sigma(\mathbf{x}) - \frac{1}{2}\right) \cdot \Pr(\mathbf{x})$. **d)** Shows our final GPify formula $\left(\sigma(\mathbf{x}) - \frac{1}{2}\right) \cdot \Pr(\mathbf{x}) + \frac{1}{2}$

where $|S|$ denotes the size of the dataset. While we use the notation $\Pr(\mathbf{x})$, we should note that this is not strictly the probability of the element $\mathbf{x}$, but rather the probability that $\mathbf{x} \sim p_{\text{train}}$. It is also noteworthy that $\Pr(\mathbf{x})$ exactly corresponds to the false-negative-rate as used in OOD detection literature (Yang et al., 2024). When constructing this empirical distribution, it is valid to choose S to be the training data for the NF. However, since the NF learns the data distribution directly, it is also possible to construct S by sampling from the NF, which is especially useful when training data is sparse. In our experiments we evaluate both methods of constructing S.

4.2 1D Toy Problem

Figure 2 illustrates the stages in our intuitive process when designing GPify. All 4 plots display each respective model's confidence $P(class = 1)$. Since the NF alone has no notion of classes, we plot the empirical probability $\Pr(\mathbf{x})$ in *a)*. We also use a sigmoid here instead of softmax, since it is a 2-class problem.

Figure 2 *a)* leads to the observation that we can effectively select a mode in $\Pr(\mathbf{x})$ by multiplying it with the sigmoid, as illustrated in Fig. 2 *b)*. By shifting the sigmoid, we recover the $class = 0$ mode with negative confidence:

$$\left(\sigma(\mathbf{x}) - \frac{1}{2}\right) \cdot \Pr(\mathbf{x}) \ ,$$

as illustrated in Fig. 2 *c)*. Shifting the resulting function by $\frac{1}{2}$ removes any negative confidence values:

$$\text{GPify}(\mathbf{x}) = \left(\sigma(\mathbf{x}) - \frac{1}{2}\right) \cdot \Pr(\mathbf{x}) + \frac{1}{2} \ .$$

As can be seen in Fig. 2 *d)*, this construction gives us a result that is similar to the GP. The primary difference between GPify and the GP is that GPify produces a narrower probability mass. GPify successfully reproduces both the base uncertainty-level of $\frac{1}{2}$, and the probability of $< \frac{1}{2}$ for the mode representing $class = 0$.

4.3 2D Toy Problem

In this section we extend GPify to multi-class problems by a simple change in the formula. For a problem with C classes, the base uncertainty-level should be $\frac{1}{C}$. As such we adjust GPify to the following:

$$\text{GPify}(\mathbf{x}) = \left(\text{MSP}(\mathbf{x}) - \frac{1}{C}\right) \cdot \Pr(\mathbf{x}) + \frac{1}{C} \ ,$$

Where MSP refers to the maximum softmax probability as defined in Sect. 3.4.2. Figure 3 shows contour plots of each model's confidence for $class = 0$. The leftmost plot shows softmax, and we observe that it can be used to select a mode in the NF, just like the sigmoid in a single dimension. It is immediately clear that the general structure of GPify is identical to the GP. However, as in the 1-dimensional case, GPify produces a narrower probability mass.

When sampling S from the NF in order to create our empirical distribution $\Pr(\mathbf{x})$, we sample from a standard normal $\mathcal{N}(0, 1)$, since that is the distribution we choose for $p_u(\mathbf{u})$ when we train our NF. If we instead sample S using a larger standard deviation, it will contain more low-probability samples, and thus produce a less narrow empirical probability mass. In the rightmost plot of Fig. 3 we plot GPify with an empirical distribution sampled from $\mathcal{N}(0, 5)$. We include this example to further highlight the similarity to the GP.

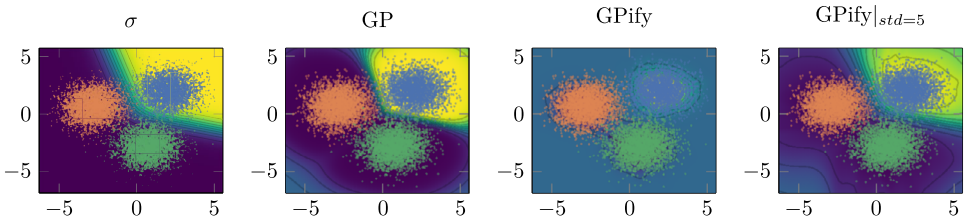

Fig. 3 Our 2-dimensional 3-class problem. Each contour plot shows the confidence of $class = 0$ for its respective model. The scatter points represent the 3 classes: blue=0, red=1, green=2. GPify$|_{std=5}$ refers to GPify where the empirical distribution is sampled from $\mathcal{N}(0, 5)$

4.4 Mnist

Our purpose with this section is to apply GPify to a larger classification problem to investigate if its properties scale to larger problems. Since MNIST is a significantly larger problem compared to our toy problems, it is not feasible to use the entire training set for the GP. As such, we randomly sample 8192 images from the training set to use for the GP. Using this sub-sampled training set, our GP gets $\sim 95\%$ test accuracy, and as such we consider it a fair comparison. However, for larger and more complex datasets GPs do not scale. Even for CIFAR-10 (Krizhevsky et al., 2009), which is a small dataset by modern standards, GPs perform significantly worse than neural network-based models (Blomqvist et al., 2020; Wilk et al., 2017).

Another reason we choose MNIST for this section is because it is small and simple enough to be visualized in 2D using UMAP (Mcinnes et al., 2020). In order to produce 2D contour plots, like Fig. 3, we train a parametric UMAP (Sainburg et al., 2021) autoencoder, which allows us to map MNIST to and from the 2D plane.

Figure 4 shows contour plots representing confidences in two different MNIST digit classes. We choose to plot confidences for $class = 4$ and $class = 1$ to highlight the fact that both models appear to produce sharp decision boundaries where data is tightly clustered. Again, we observe for $class = 1$ that GPify produces a narrower probability mass compared to the GP. However, in contrast to the previous two examples, here it looks like the GP's probability mass includes areas outside the data regions, which makes GPify's narrow distribution appear more conservative.

We would like to stress that UMAP is an approximation, and therefore it is impossible to draw definitive conclusions from it. In spite of that, we believe this UMAP projection serves as a good indication that our GPify confidence formula retains its desirable properties, and compares well to the GP even for larger, higher-dimensional problems.

4.5 Interpretation

In accordance with the view of uncertainties representing probability distributions, we interpret our resulting GPify score as a joint probability. While the softmax computes its confidence in the form of a probability $P(class = i \mid \mathbf{x})$, that probability is only valid if $\mathbf{x} \sim p_{\text{train}}(\mathbf{x})$. As such, we interpret GPify as a joint probability $P(class = 1,\ \mathbf{x} \sim p_{\text{train}}(\mathbf{x}) \mid \mathbf{x})$.

Our main field of study is robotics, where we tend to view behaviours as a sequence of operations. In this view, classifying an image is made up of two operations: capturing an image, and then classifying it. Thus, softmax gives us the probability that its classification is correct, and the NF gives us the probability that the captured image is valid. Finally GPify gives us the joint probability that both operations were executed successfully.

5 Scod Evaluation

As we have seen in previous sections, GPify is an intuitive confidence score that encompasses both aleatoric and epistemic uncertainty in a way that is similar to a GP. We have also seen that GPify appears to scale to larger problems beyond our toy examples. In this section, we complement our qualitative results with a quantitative evaluation within the SCOD (Xia & Bouganis, 2024) framework.

For our experiments we construct ID and OOD data by collecting subsets of ImageNet1K (Russakovsky et al., 2015). We select 150 random classes without replacement from ImageNet1K, and use 100 classes (127,478 train and 5000 val images) for our ID data, and the remaining 50 classes (2500 val images) as our OOD data. This partitioning follows Xia and Bouganis (2024) in that OOD data is disjoint from ID data, and as such any prediction on an OOD sample should be considered invalid by default.

In addition to evaluating $\text{ID}_{\checkmark}$ vs. $\text{ID}_{\times}$, and $\text{ID}_{\checkmark}$ vs. OOD discrimination, we also evaluate ID vs. OOD discrimination. We agree with Xia and Bouganis (2024) that it is desirable for the combined score to retain the softmax's capabilities in $\text{ID}_{\checkmark}$ vs. $\text{ID}_{\times}$ discrimination, and by the same reasoning we believe it is also important for the combined score to retain the NF's capabilities for ID vs. OOD discrimination.

5.1 Backbone Networks

Since the NF in our GPify score is applied to features extracted from a trained network, we are interested in its generalizability across different network architectures. Kirichenko et al. (2020), as well as our previous work (Lind et al., 2023, 2024), has shown that NFs work well

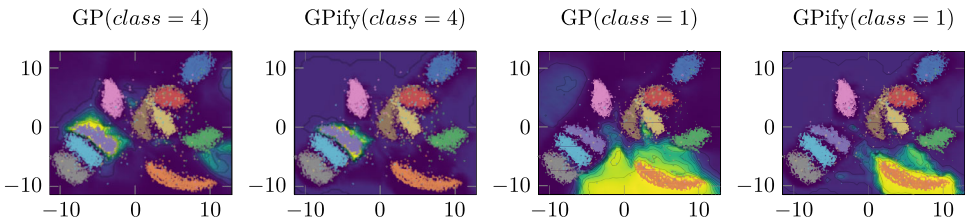

Fig. 4 Contour plots from MNIST, mapped into the 2D plane using UMAP. Each color of the scatter points indicate a digit class. The two leftmost plots display their respective model's confidence in $class = 4$, and the two rightmost display confidence for $class = 1$

for epistemic uncertainty when paired with convolutional architectures. However, while there are works to integrate attention-mechanisms into NFs (Ho et al., 2019), we are not aware of any work that applies an NF to features from an attention-based architecture. Therefore, we conduct our SCOD experiment using 4 different backbone networks, both convolutional and attention-based: ResNet34, EfficientNet-B0, ViT-B16, and Swin-B. We refrain from evaluating other backbone architectures, as these are the most commonly used in image processing.

Each backbone is trained for 300 epochs with MixUp ($\alpha = 0.2$) using the Adam (Kingma & Ba, 2014) optimizer. Both convolutional architectures, ResNet34 and EfficientNet-B0, are trained with a cosine annealing scheduler (Loshchilov & Hutter, 2017), a starting learning rate of 0.008, and a batch size of 256. For the transformer architectures, ViT-B16 and Swin-B, we found better results using a fixed learn rate of 0.0001, and a batch size of 128.

5.2 Normalizing Flow

In our preliminary testing, the NF consistently produced its best results when trained on the validation set for its corresponding backbone model. We believe that applying the backbone to its own training data produces a skewed distribution of features compared to examples that were not seen during training, perhaps due to some amount of overfitting. More tests are required to verify this hypothesis. Nonetheless, we exclude the backbone's training set when training our NF, and only use it for evaluation, to avoid overfitting.

Ideally, we would run our SCOD evaluation using the ImageNet1K test set while reserving the entire validation set for our NF training. However, we do not have access to labels for the ImageNet1K test set. Therefore, we further subdivide the ID validation set and use 4000 samples to train our NF, and reserve 1000 samples for the final SCOD evaluation.

Our NF architecture is made up of 10 sequential coupling layers, operating on alternating halves of the input vector. $\alpha(\mathbf{x})$ and $\beta(\mathbf{x})$ are each made up of a 2-layer neural network, with a hidden layer size of 512. The same architecture is used for all 4 backbones.

We train our NFs for 300 epochs using the Adam (Kingma & Ba, 2014) optimizer and a cosine annealing scheduler (Loshchilov & Hutter, 2017), with a starting learning rate of 0.001, a batch size of 256, and an L_2 weight decay factor of 0.01.

5.3 Constructing the Empirical Distribution

As discussed in Sect. 4.1, the set S used when creating our empirical distribution can either be taken directly from the NF's training data, or it can be sampled from the NF itself. In Sect. 4.3, we briefly mentioned that changing the standard deviation when sampling the NF changes the resulting empirical distribution. The first method we evaluate sets S to be equal to the NF's training set, and we denote this by $\text{GPify}|_{\text{train}}$. We also evaluate two strategies of sampling S from the NF itself, one where we use a fixed standard deviation of 1, and one where we optimize the standard deviation based on the $\text{ID}_{\checkmark}$ vs. $\text{ID}_{\times}$ discrimination performance on the NF's training set. These are denoted by $\text{GPify}|_{std=\ldots}$.

5.4 Other Scores

For comparison with GPify, we evaluate all the originally proposed variants of SIRC (Xia & Bouganis, 2024), along with its constituent components: Maximum Softmax Probability (MSP), entropy ($\mathcal{H}$), $||\mathbf{z}||_1$ (L_1), Residual $||\mathbf{z}^{P^\perp}||_2$ (Wang et al., 2022) (Res), and L_2-normalized distance to the kth nearest ID neighbor (KNN). Like Xia and Bouganis (2024), we set $k = 10$ for KNN. With the Residual score, we compute the principal subspace based on $\frac{1}{4}$ of the dimensions in the feature vector.

We also compare with GradNorm and ViM, as well as the Energy score over logits (Liu et al., 2020): $-\log \sum_i \exp(l_i)$, since it is a component in ViM (Wang et al., 2022).

Additionally, we compare with the plain likelihood from our NFs, to verify that GPify does not significantly reduce the efficacy for ID vs. OOD discrimination. We also evaluate a SIRC combination using the same components as GPify (s_1=MSP and s_2=NF).

5.5 Results and Discussion

Tables 1, 2, 3, and 4 show AUROC scores for each test in the SCOD framework. Each table corresponds to one backbone model.

Table 1 AUROC scores for the ResNet34 model. **Bold** indicates the best score, and underline indicates scores that are less than 1% below the best.

	$ID_{\checkmark}$ vs. $ID_{\times}$	$ID_{\checkmark}$ vs. OOD	ID vs. OOD
MSP	89.73%	86.53%	79.89%
$\mathcal{H}$	87.98%	85.50%	79.35%
L_1	83.67%	83.97%	78.46%
Res	81.76%	82.37%	77.30%
KNN	25.00%	21.19%	24.89%
NF	80.94%	86.99%	82.65%
Energy	86.88%	86.34%	80.39%
ViM	85.89%	85.71%	79.94%
GradNorm	26.07%	24.88%	28.89%
SIRC(MSP, L_1)	90.16%	87.19%	80.43%
SIRC(MSP, Res)	90.00%	87.12%	80.47%
SIRC(MSP, KNN)	89.61%	86.11%	79.47%
SIRC(-$\mathcal{H}$, L_1)	88.95%	86.63%	80.21%
SIRC(-$\mathcal{H}$, Res)	88.76%	86.58%	80.31%
SIRC(-$\mathcal{H}$, KNN)	87.64%	84.84%	78.72%
SIRC(MSP, NF)	89.92%	87.47%	81.04%
GPify$\|_{\text{train}}$	86.45%	89.11%	83.90%
GPify$\|_{std=1.0}$	90.37%	94.39%	93.66%
GPify$\|_{std=0.96}$	**90.49%**	**94.54%**	**93.87%**

Table 2 AUROC scores for the EfficientNet-B0 model. **Bold** indicates the best score, and underline indicates scores that are less than 1% below the best.

	$ID_{\checkmark}$ vs. $ID_{\times}$	$ID_{\checkmark}$ vs. OOD	ID vs. OOD
MSP	87.58%	83.82%	75.87%
$\mathcal{H}$	85.04%	83.27%	76.30%
L_1	67.61%	66.37%	66.70%
Res	45.65%	41.92%	42.76%
KNN	62.54%	67.27%	64.90%
NF	82.15%	82.21%	75.83%
Energy	72.76%	74.26%	69.62%
ViM	67.34%	68.91%	65.36%
GradNorm	81.71%	81.77%	75.61%
SIRC(MSP, L_1)	**87.89%**	83.88%	75.72%
SIRC(MSP, Res)	87.70%	83.82%	75.72%
SIRC(MSP, KNN)	87.48%	**84.30%**	76.67%
SIRC(-$\mathcal{H}$, L_1)	85.46%	83.42%	76.27%
SIRC(-$\mathcal{H}$, Res)	85.15%	83.27%	76.20%
SIRC(-$\mathcal{H}$, KNN)	85.10%	83.93%	**77.22%**
SIRC(MSP, NF)	87.83%	83.67%	75.53%
GPify$\|_{\text{train}}$	84.79%	83.24%	76.25%
GPify$\|_{std=1.0}$	87.71%	83.78%	75.80%
GPify$\|_{std=1.11}$	87.80%	83.80%	75.78%

Table 3 AUROC scores for the ViT-B16 model. **Bold** indicates the best score, and underline indicates scores that are less than 1% below the best.

	$ID_{\checkmark}$ vs. $ID_{\times}$	$ID_{\checkmark}$ vs. OOD	ID vs. OOD
MSP	83.27%	80.76%	72.53%
$\mathcal{H}$	83.01%	80.56%	72.53%
L_1	81.49%	81.23%	73.30%
Res	86.03%	85.38%	76.20%
KNN	72.66%	69.35%	63.75%
NF	85.90%	85.43%	76.41%
Energy	80.94%	77.87%	70.36%
ViM	86.58%	85.65%	76.35%
GradNorm	70.91%	67.73%	62.90%
SIRC(MSP, L_1)	83.94%	81.55%	73.12%
SIRC(MSP, Res)	83.87%	81.52%	73.08%
SIRC(MSP, KNN)	83.70%	81.05%	72.70%
SIRC(-$\mathcal{H}$, L_1)	83.79%	81.58%	73.26%
SIRC(-$\mathcal{H}$, Res)	83.79%	81.52%	73.17%
SIRC(-$\mathcal{H}$, KNN)	83.47%	80.88%	72.65%
SIRC(MSP, NF)	84.41%	82.32%	73.76%
GPify$\|_{\text{train}}$	**87.08%**	**86.35%**	**78.63%**
GPify$\|_{std=1.0}$	86.16%	84.85%	75.90%
GPify$\|_{std=1.01}$	86.16%	84.85%	75.90%

Table 4 AUROC scores for the Swin-B model. **Bold** indicates the best score, and underline indicates scores that are less than 1% below the best.

	$ID_{\checkmark}$ vs. $ID_{\times}$	$ID_{\checkmark}$ vs. OOD	ID vs. OOD
MSP	90.90%	86.96%	81.82%
$\mathcal{H}$	90.29%	86.68%	81.72%
L_1	80.61%	80.94%	76.96%
Res	86.95%	86.78%	82.16%
KNN	78.54%	75.38%	71.91%
NF	86.42%	86.05%	81.53%
Energy	88.00%	84.54%	79.92%
ViM	88.26%	87.76%	82.95%
GradNorm	65.44%	64.07%	62.24%
SIRC(MSP, L_1)	**91.29%**	87.63%	82.35%
SIRC(MSP, Res)	91.26%	87.73%	82.51%
SIRC(MSP, KNN)	91.16%	87.20%	82.02%
SIRC(-$\mathcal{H}$, L_1)	90.61%	87.47%	82.34%
SIRC(-$\mathcal{H}$, Res)	90.65%	87.63%	82.55%
SIRC(-$\mathcal{H}$, KNN)	90.58%	87.00%	81.96%
SIRC(MSP, NF)	91.27%	87.94%	82.72%
GPify$\|_{\text{train}}$	88.10%	87.10%	82.54%
GPify$\|_{std=1.0}$	88.47%	87.85%	**84.63%**
GPify$\|_{std=2.33}$	90.52%	**88.34%**	83.26%

Table 5 AUROC scores for adversarial detection. **Min Diff** refers to the adversarial image created using the minimum number of iterations for misclassification. $>$ **x%** refers to images where the Softmax confidence is $>$ x% for the adversarial class.

	ResNet	EffNet	ViT	Swin
Min Diff	96.04%	95.79%	89.45%	79.01%
$>$ 90%	97.75%	23.59%	79.64%	80.26%
$>$ 95%	97.81%	13.65%	74.84%	83.61%
$>$ 99%	98.81%	1.89%	80.43%	92.63%

Our first and most important observation is that GPify performs competitively with SIRC, even outperforming SIRC in 8 out of the 12 AUROC tests. Additionally, GPify appears to retain the discriminative performance of both the softmax and the NF. We observe no cases where $\text{ID}_{\checkmark}$ vs. OOD or ID vs. OOD detection performance is reduced by $> 1\%$ compared to the standalone softmax or NF. The only cases where we do observe a decrease in performance of $> 1\%$ are for $\text{ID}_{\checkmark}$ vs. $\text{ID}_{\times}$, where $\text{GPify}|_{\text{train}}$ reduces performance in 3 out of the 4 backbones, and $\text{GPify}|_{std=1}$ reduces performance for Swin. As such we recommend to optimize the sample standard deviation to maximize $\text{ID}_{\checkmark}$ vs. $\text{ID}_{\times}$ on the NF's training set, since that appears to result in the best performance overall. However, with 3 of our 4 backbones, the optimized standard deviation is close to 1, and thus it appears that sampling with $std = 1$ is a good default choice.

6 Adversarial Detection

Here we reuse the same models, and datasets from our SCOD evaluation, and we create adversarial examples using the basic iterative method.

For each classifier, we create adversarial examples based on all the ID images in our test set. In order to evaluate the performance at different levels of severity, we save multiple adversarial examples throughout the iterative method. First, we save one adversarial example as soon as the image is misclassified, which is similar to the original idea of finding the minimum perturbation required for misclassification. We also save adversarial examples when the misclassification confidence (MSP) reaches 90%, 95%, and 99%.

Table 5 summarizes AUROC scores for each model, and each misclassification confidence threshold.

Based on these results, we can quickly conclude that GPify does have the capability of detecting adversarial examples. Since these misclassifications are constructed based on the classifier's MSP, it follows that they cannot be detected using MSP alone. Hence, these results show that our $\Pr(\mathbf{x})$ from the NF is sensitive enough for GPify to detect these adversarial examples in spite of their high MSP values.

It should be noted, however, from this simple experiment, we cannot draw any conclusions as to how well GPify compares against dedicated adversarial detectors (Aldahdooh et al., 2022). Additionally, the efficacy seems to differ between the different backbone models, especially EfficientNet. As such, we call for future work to more closely investigate differences between architectures, and a thorough comparison to dedicated adversarial detection methods.

7 Future Work

As indicated in section 3.1, the notion of epistemic uncertainty is closely tied to the problem of OOD detection. In our work, we have have assumed only the availability of in-distribution data, which is a common assumption in OOD detection literature (Yang et al., 2024), mainly due to the inherent difficulty of data collection within some domains. Consider, for example, human intent recognition (Jain & Argall, 2019), where the inherent inconsistency in human behavior makes OOD data sampling impractical, or terrain traversability classification (Sevastopoulos & Konstantopoulos, 2022), where the sheer variability of outdoor terrain makes it difficult to define OOD. For domains where OOD data is available, however, supervised learning methods can be used to explicitly reject OOD samples, which often leads to improved performance (Yang et al., 2024). It would thus be an interesting research direction to evaluate supervised epistemic uncertainties within the SCOD setting, for example using a supervised NF (Izmailov et al., 2020).

Additionally, Since both SIRC and GPify rely heavily on the backbone model for estimating aleatoric uncertainty, it would be interesting to evaluate their performance on long-tailed datasets. For such datasets, the aleatoric uncertainty from softmax might be poorly calibrated due to data sparsity among tail classes. We hypothesize that the NF in our GPify score would recognize tail classes as low-density regions and thus lower the final confidence estimate, but this remains to be empirically confirmed.

Finally, in section 5.2 we posed a hypothesis that features extracted from the backbone's training set is not representative of the broader dataset. First and foremost, more experiments are needed to thoroughly investigate this hypothesis. Second, it is commonly accepted within learning literature that more training data leads to better results. As such, the backbone's training set is a source of untapped potential for improving the NF, and it would therefore be interesting to see if it can be effectively integrated into the NF's training.

8 Conclusion

In this paper, we have proposed GPify, which is a combined confidence score based on softmax and normalizing flows. We have derived GPify mostly based on intuitive reasoning, with the goal making it an intuitive and probabilistically founded confidence score similar to that of a Gaussian process. As a direct improvement over softmax, we show that GPify has capabilities for detecting adversarial examples. Our quantitative experiments show that GPify performs competitively with SIRC, and preserves the properties of both the softmax and normalizing flow. We believe that the main advantage to GPify over a Gaussian process is its scalability, where it can scale to arbitrarily large problems thanks to its neural network-based implementation. Its main advantage over SIRC is its probabilistic foundation, from the normalizing flow, as well as the fact that it is intuitive and interpretable by design.

Acknowledgements This research is funded by the Excellence Center at Linköping-Lund in Information Technology (ELLIIT), and the Wallenberg AI, Autonomous Systems and Software Program (WASP). Computations for this publication were enabled by the supercomputing resource Berzelius provided by the National Supercomputer Centre at Linköping University and the Knut and Alice Wallenberg foundation.

Funding Open access funding provided by Lund University.

Data Availability All data used in this work is publicly available under the ImageNet dataset (Russakovsky et al., 2015).

Declarations

Competing interests The authors declare no competing interests.

References

Aldahdooh, A., Hamidouche, W., Fezza, S. A., & Déforges, O. (2022). Adversarial example detection for dnn models: a review and experimental comparison. *Artificial Intelligence Review, 55*(6), 4403–4462. https://doi.org/10.1007/s10462-021-10125-w

Blomqvist, K., Kaski, S., & Heinonen, M. (2020). Deep convolutional gaussian processes. In: Brefeld, U., Fromont, E., Hotho, A., Knobbe, A., Maathuis, M., Robardet, C. (eds.) Machine Learning and Knowledge Discovery in Databases, pp. 582–597. Springer, Cham. https://doi.org/10.1007/978-3-030-46147-8_35.

El-Yaniv, R., & Wiener, Y. (2010). On the foundations of noise-free selective classification. *Journal of Machine Learning Research, 11*(53), 1605–1641.

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K.Q. (2017). On calibration of modern neural networks. In: Proceedings of the 34th International Conference on Machine Learning - Volume 70. ICML'17, pp. 1321–1330. JMLR.org, Sydney, NSW, Australia.

Gardner, J., Pleiss, G., Weinberger, K.Q., Bindel, D., & Wilson, A.G. (2018). Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. Advances in neural information processing systems 31.

Goodfellow, I.J., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples arXiv:1412.6572 [stat].

Grimmett, H., Triebel, R., Paul, R., & Posner, I. (2016). Introspective classification for robot perception. *The International Journal of Robotics Research, 35*(7), 743–762. https://doi.org/10.1177/0278364915587924

Ho, J., Chen, X., Srinivas, A., Duan, Y., & Abbeel, P. (2019). Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design. arxiv:org/abs/1902.00275.

Huang, R., Geng, A., & Li, Y. (2024). On the importance of gradients for detecting distributional shifts in the wild. In: Proceedings of the 35th International Conference on Neural Information Processing Systems. NIPS '21, pp. 677–689. Curran Associates Inc., Red Hook, NY, USA.

Izmailov, P., Kirichenko, P., Finzi, M., & Wilson, A.G. (2020). Semi-supervised learning with normalizing flows. In: III, H.D., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 4615–4630. PMLR, https://proceedings.mlr.press/v119/izmailov20a.html.

Jain, S., & Argall, B. (2019).Probabilistic human intent recognition for shared autonomy in assistive robotics. J. Hum.-Robot Interact. 9(1), 2–1223 https://doi.org/10.1145/3359614.

Kingma, D.P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. arXiv. arxiv:org/abs/1412.6980.

Kurakin, A., Goodfellow, I., & Bengio, S. (2017). Adversarial examples in the physical world arXiv:1607.02533 [cs].

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Kirichenko, P., Izmailov, P., & Wilson, A.G. (2020). Why Normalizing Flows Fail to Detect Out-of-Distribution Data. In: Advances in Neural Information Processing Systems, vol. 33, pp. 20578–20589. Curran Associates Inc.

Kim, J., Koo, J., & Hwang, S. (2023). A unified benchmark for the unknown detection capability of deep neural networks. Expert Systems with Applications 229, 120461 https://doi.org/10.1016/j.eswa.2023.120461. arXiv:2112.00337 [cs].

Kristoffersson Lind, S., Triebel, R., Nardi, L., & Krueger, V. (2023). Out-of-distribution detection for adaptive computer vision. In: Gade, R., Felsberg, M., Kämäräinen, J.-K. (eds.) Image Analysis. Lecture Notes in Computer Science, pp. 311–325. Springer, Cham. https://doi.org/10.1007/978-3-031-31438-4_21.

Kristoffersson Lind, S., Xiong, Z., Forssén, P.-E., & Krüger, V. (2024). Uncertainty quantification metrics for deep regression. *Pattern Recognition Letters, 186*, 91–97. https://doi.org/10.1016/j.patrec.2024.09.011

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324. https://doi.org/10.1109/5.726791

Loshchilov, I., & Hutter, F. (2017). Sgdr: Stochastic gradient descent with warm restarts arXiv:1608.03983 [cs].

Lind, S.K., Triebel, R., & Krüger, V. (2024) . Making the flow glow - robot perception under severe lighting conditions using normalizing flow gradients arXiv:2412.07565 [cs].

Liu, W., Wang, X., Owens, J., & Li, Y. (2020). Energy-based out-of-distribution detection. In: Advances in Neural Information Processing Systems, vol. 33, pp. 21464–21475. Curran Associates Inc, https://proceedings.neurips.cc/paper/2020/hash/f5496252609c43eb8a3d147ab9b9c006-Abstract.html.

McInnes, L., Healy, J., & Melville, J. (2020). Umap: Uniform manifold approximation and projection for dimension reduction arXiv:1802.03426 [stat].

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., & Lakshminarayanan, B. (2021). Normalizing Flows for Probabilistic Modeling and Inference. *Journal of Machine Learning Research, 22*(57), 1–64.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV), 115*(3), 211–252. https://doi.org/10.1007/s11263-015-0816-y

Rasmussen, C.E., & Williams, C.K.I. (2005). Gaussian Processes for Machine Learning. The MIT Press, Cambridge. https://doi.org/10.7551/mitpress/3206.001.0001.

Sevastopoulos, C., & Konstantopoulos, S. (2022). A survey of traversability estimation for mobile robots. *IEEE Access, 10*, 96331–96347. https://doi.org/10.1109/ACCESS.2022.3202545

Sainburg, T., McInnes, L., & Gentner, T. Q. (2021). Parametric umap embeddings for representation and semisupervised learning. *Neural Computation, 33*(11), 2881–2907.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks arXiv:1312.6199 [cs].

Wilk, M., Rasmussen, C.E., & Hensman, J. (2017). Convolutional gaussian processes. In: Advances in Neural Information Processing Systems, vol. 30. Curran Associates Inc, https://papers.nips.cc/paper_files/paper/2017/hash/1c54985e4f95b7819ca0357c0cb9a09f-Abstract.html.

Wenger, J., Kjellström, H., & Triebel, R. (2020) Non-parametric calibration for classification. In: Chiappa, S., Calandra, R. (eds.) Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 108, pp. 178–190. PMLR, https://proceedings.mlr.press/v108/wenger20a.html.

Wang, H., Li, Z., Feng, L., & Zhang, W. (2022). Vim: Out-of-distribution with virtual-logit matching. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4911–4920. https://doi.org/10.1109/CVPR52688.2022.00487.https://ieeexplore.ieee.org/document/9879414.

Xia, G., & Bouganis, C.-S. (2024). Augmenting the softmax with additional confidence scores for improved selective classification with out-of-distribution data. *International Journal of Computer Vision, 132*(9), 3714–3752. https://doi.org/10.1007/s11263-024-02029-3

Yang, J., Zhou, K., Li, Y., & Liu, Z. (2024). Generalized out-of-distribution detection: A survey. *International Journal of Computer Vision, 132*(12), 5635–5662. https://doi.org/10.1007/s11263-024-02117-4