PREDICTING UNRELIABLE PREDICTIONS BY SHATTERING A NEURAL NETWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

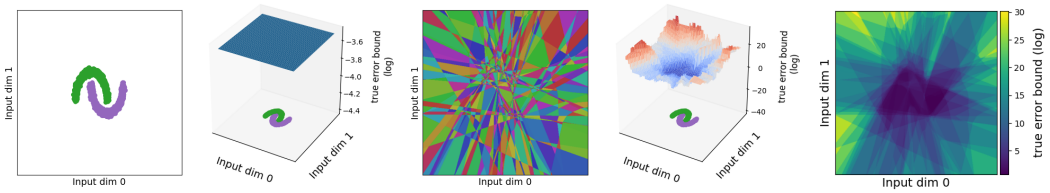
Generalization error bounds measure the deviation of performance on unseen test data from performance on training data. However, by providing one scalar per model, they are input-agnostic. What if one wants to predict error for a specific test sample? To answer this, we propose the novel paradigm of input-conditioned generalization error bounds. For piecewise linear neural networks, given a weighting function that relates the errors of different input activation regions together, we obtain a bound on each region's generalization error that scales inversely with the density of training samples. That is, more densely supported regions are more reliable. As the bound is input-conditioned, it is to our knowledge the first generalization error bound applicable to the problems of detecting out-of-distribution and misclassified in-distribution samples for neural networks; we find that it performs competitively in both cases when tested on image classification tasks.

1 INTRODUCTION

Maximizing generalization is the problem of quantifying and minimizing model error on unseen test data. This includes model selection, the focus of much of the classical work on generalization (Mohri et al., 2018), which involves bounds on expected test error that vary with model or model class, and sample selection (Lee et al., 2018), which involves test error metrics for a given model that vary with location in input space. Whilst model selection is an important part of the training process, sample selection is important for testing, because even the best selected model invariably makes mistakes and being able to discard erroneous predictions is critical for safe deployment. Unlike generalization bounds for model selection, the link to measuring generalization error for sample selection methods is often not explicit. This paper proposes a novel metric for the class of piecewise-linear neural networks that bridges both categories: a generalization error bound that is sample dependent.

What is the reason for targeting neural networks? In piecewise-linear neural networks, the linear function applied varies with location in input space. Specifically, for each linear activation region in input space there is a distinct linear function that uses a different parameterization or subset of network weights (Montúfar et al., 2014), which for convenience we call the subfunctions of the network. Generalization error bounds measure deviation in empirical error for a given function. Thus, one can construct an input-dependent generalization error bound by relying on the bound of the subfunction corresponding to the region the given input belongs to. The main challenge with this idea is that naively constructing a bound in the typical format of empirical training error plus bound on generalization gap only allows the generalization error of subfunctions with a well defined empirical training error (i.e. those seen in training) to be quantified. However, test inputs typically fall in activation regions that were unseen in training for deep networks. We resolve this by introducing a weighting function that smooths training error over activation regions, allowing it to be defined for activation regions without training sample coverage. Then we demonstrate how the bound on this sample-dependent error can be brought back to model-level by integrating over subfunctions, yielding new model-level bounds that scale with the number of linear activation regions (or equivalently, binarized activation patterns) of the network. The idea of Occam's Razor, or pick the simplest model all else equal, is prevalent in model selection. The novelty of our model-level bounds compared to existing work is that they use expressivity (Montúfar et al., 2014; Hanin & Rolnick, 2019) to measure model simplicity or complexity, rather than number of parameters or data mean and covariance metrics (Rivasplata et al., 2020). An intuitive interpretation is that one can expect generalization ability to scale with the degree of abstraction or information loss implemented by model inference.

Figure 1 illustrates input-dependent generalization error bounds for a model trained on the half moons dataset. The network is shattered into subfunctions corresponding to linear activation regions, and a generalization error bound is derived that uses smooth training sample density. Whereas model-level bounds produce one value for the model, considering the model as a composition of individual functions allows us to discriminate risk based on input region.



(a) (b) (c) (d) (e)

Figure 1: Shattering a neural network trained on the halfmoons dataset into co-dependent linear subfunctions to obtain a heatmap of unreliability across the input space. The model is a MLP with two hidden ReLU layers of 32 units each. (a) Training data. (b) Traditional model-level error bound: shown here is the single model bound from Mohri et al. (2018, eq. 2.17). (c) Linear activation regions. (d) Ours: subfunction error bound as unreliability. (e) Unreliability heatmap in 2D.

2 Input-dependent generalization error bounds

2.1 Notation

Let $f : \mathcal{X} \to \mathcal{Y}$ be a piecewise linear neural network comprised of linear functions and ReLU activation functions. Let $V = (v_j)_{1 \leq j \leq M}$, ordered according to any fixed ordering, be the set of M ReLUs in f where $\forall j : v_j : \mathcal{X} \to \{0, 1\}$ is 1 if the ReLU is positive valued when $f(x)$ is computed, and 0 otherwise. Let patterns set $P \subset \{0,1\}^M$, $P = (p_i)_{1 \leq i \leq C}$ be the set of all feasible ReLU activation decisions, meaning all possible binary patterns induced by traversing the input space $\mathcal{X}$. In general this is not equal to the full bit space $\{0, 1\}^M$, and computing it exactly is an open problem (Montúfar et al., 2014). There is a bijection between these patterns and the linear activation regions of the network, as each linear activation region is uniquely determined by the set of ReLU activation decisions (Raghu et al., 2017). Let activation region a_i correspond to pattern p_i:

$$a_i \triangleq \{x \in \mathcal{X} \quad | \quad \forall j \in [1, M] : p_i^j = v_j(x)\}, \tag{1}$$

Each pattern p_i defines a distinct linear function $h_i : \mathcal{X} \to \mathcal{Y}$, or *subfunction*, by the replacement of the ReLU corresponding to each v_j (non-linear) by the identity function if $p_i^j = 1$ or the zero function otherwise (both linear). Let the set of all subfunctions be $H = (h_i)_{1 \leq i \leq C}$, ordered identically to P. Since activation regions are non-overlapping polytopes in input space, each input sample belongs to a unique activation region, so inference with model f can be rewritten as $f(x) = h_i(x)$ where $x \in a_i$. For convenience we overload the definition of H so the subfunction induced for any input sample x is $H(x) \triangleq h_i$ where $x \in a_i$.

2.2 From model-level bounds to input-level bounds

Consider a training dataset S containing N sampled pairs, $S \in (\mathcal{X} \times \mathcal{Y})^N$, where samples are drawn iid from distribution D. Define the empirical error or risk of the full network on this dataset:

$$\widehat{R}_S(f) \triangleq \frac{1}{|S|} \sum_{n=1}^{|S|} r(S^n; f), \tag{2}$$

where r is a bounded error function on samples: $0 \leq r((x, y); f) \leq 1$ for all $x, y \in (\mathcal{X} \times \mathcal{Y})$ and S^n is the n-th element of S. Note empirical error, e.g. proportion of incorrect classifications, is not necessarily the objective function for training; it is fine for the latter to not have finite bounds.

Let us quantify the empirical error for a subfunction h_i using traditional empirical error (eq. (2)). The activation region dataset is $S_i \triangleq S \cap a_i$ with $N_i \triangleq |S_i|$ samples drawn iid from its data distribution D_i, defined as $\mathbb{P}_{D_i}(x, y) = \mathbb{P}_D(x, y | x \in a_i)$. Then:

$$\widehat{R}_{S_i}(h_i) = \frac{1}{N_i} \sum_{n=1}^{N_i} r(S_i^n; f), \tag{3}$$

and for any $\delta \in (0, 1]$, with probability $> 1 - \delta$: $$\mathop{\mathbb{E}}_{S_i}[\widehat{R}_{S_i}(h_i)] \leq \widehat{R}_{S_i}(h_i) + \sqrt{\frac{\log \frac{2}{\delta}}{2N_i}}. \tag{4}$$

This uses a Hoeffding-inequality based model-level bound (Mohri et al., 2018, eq. 2.17) on each subfunction. There are several issues with this formulation. First, it treats the empirical error of different subfunctions as independent when in general, they are not. Since different activation regions are bounded by shared hyperplanes, and hence the subfunctions share parameters, there exists useful evidence for a subfunction's performance outside its own activation region. Second, test samples overwhelmingly induce unseen activation patterns in deep networks, and the bound in eq. (4) is infinite if $N_i = 0$, making this quantity uninformative for the purposes of comparing subfunctions.

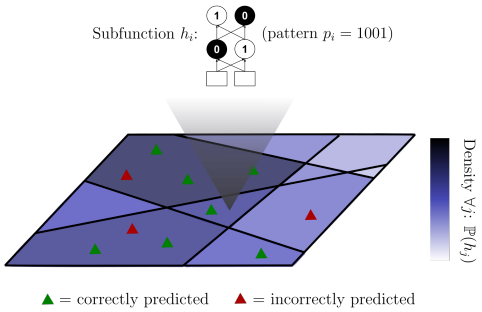


Figure 2: Each linear activation region in 2D input space (plane) is mapped to a unique subfunction, activation decision pattern, and set of training samples (triangles). A smooth density is defined given the number of samples in each region.

This motivates the following empirical risk metric for subfunctions. Fix non-negative weighting or closeness distance function between subfunctions, $k : H \times H \to \mathcal{R}^{\geq 0}$. For any subfunction $h_i \in H$, define its probability mass:

$$\mathbb{P}(h_i) \triangleq \frac{\sum_{j \in [1,C]} N_j \; k(h_i, h_j)}{\sum_{l \in [1,C]} \sum_{j \in [1,C]} N_j \; k(h_l, h_j)}, \qquad \text{so by construction} \sum_{i \in [1,C]} \mathbb{P}(h_i) = 1, \tag{5}$$

which quantifies how densely h_i is locally populated by training samples. Rewrite the empirical error of the network as an expectation over subfunction empirical error:

$$\widehat{R}_S(f) = \frac{1}{N} \sum_{(x,y) \in S} r((x, y); f) \tag{6}$$

$$= \frac{1}{N} \sum_{(x,y) \in S} \frac{1}{\sum_{j \in [1,C]} \mathbb{P}(h_j) \; k(H(x), h_j)} \sum_{i \in [1,C]} \mathbb{P}(h_i) \; k(H(x), h_i) \; r((x, y); f) \tag{7}$$

$$= \frac{1}{N} \sum_{i \in [1,C]} \mathbb{P}(h_i) \sum_{(x,y) \in S} \frac{k(H(x), h_i) \; r((x, y); f)}{\sum_{j \in [1,C]} \mathbb{P}(h_j) \; k(H(x), h_j)} \tag{8}$$

$$= \mathop{\mathbb{E}}_{h_i} \left[\frac{1}{N} \sum_{(x,y) \in S} \frac{k(H(x), h_i) \; r((x, y); f)}{\sum_{j \in [1,C]} \mathbb{P}(h_j) \; k(H(x), h_j)} \right] \tag{9}$$

$$= \mathop{\mathbb{E}}_{h_i} \left[\frac{1}{N} \sum_{l \in [1,C]} \frac{k(h_l, h_i) \; N_l \; \widehat{R}_{S_l}(h_l)}{\sum_{j \in [1,C]} \mathbb{P}(h_j) \; k(h_l, h_j)} \right] \triangleq \mathop{\mathbb{E}}_{h_i}[\widehat{R}^*_S(h_i)]. \tag{10}$$

Theorem 2.1 (Subfunction true error bound). *Assume that $\tilde{D}_S$ is a distribution over size-N datasets that have the same activation region data distributions and dataset sizes (D_i and N_i, $\forall i \in [1, C]$) as S. Consider S as a dataset drawn iid from $\tilde{D}_S$. Then $\forall i \in [1, C]$, for any given $\delta \in (0, 1]$,*

with probability $> 1 - \delta$*:*

$$\underbrace{R^*_{\tilde{D}_S}(h_i) \triangleq \mathop{\mathbb{E}}_{S}[\widehat{R}^*_S(h_i)]}_{\text{true or generalization error}} \leq \overbrace{\underbrace{\widehat{R}^*_S(h_i)}_{\text{empirical error}} + \underbrace{\frac{1}{\mathbb{P}(h_i)}\sqrt{\frac{\log\frac{2}{\delta}}{2N}}}_{\text{generalization gap bound}}}^{\text{subfunction true error bound (STEB)}}, \tag{11}$$

proof in appendix A. Intuitively this implies that the more a subfunction is surrounded by samples for which the model makes accurate predictions - both from its own region and regions of other subfunctions, weighted by the weighting function k - the lower its empirical error and bound on generalization gap, and thus the lower its bound on true error, given any fixed δ. The further that test samples fall from densely supported training regions or subfunctions, the less likely the model is to generalize. Unlike eq. (4), this bound is finite even for subfunctions without training samples because such subfunctions are assigned non-zero density (fig. 2), given a positive-valued weighting function.

2.3 Weighting function selection

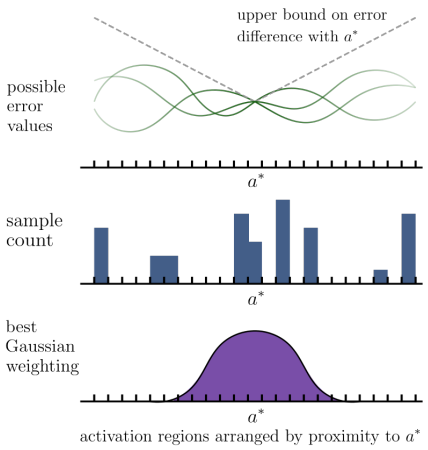


Figure 3: Given a family of Gaussian weighting functions, the best weighting function according to eq. (15) trades off precision of the smooth empirical error with sample density.

Although $\frac{1}{\mathbb{P}(h_i)}$ can be very large in general, note firstly that it is tempered by a factor of $\frac{1}{\sqrt{N}}$, and secondly that if one were to optimize the weighting function k by minimizing the distance between true error and STEB (eq. (11)), this necessarily involves finding a function that yields high densities for activation regions, otherwise the distance to true error would be large. Now we discuss the problem of selecting such a weighting function.

The ideal weighting function k and probability parameter δ produce a bound for each subfunction h_i that reflects the subfunction's true error accurately, i.e. minimizes the difference with the expected true error (without weighting function) if we had unlimited samples of its activation region a_i:

$$\min_{k,\delta} \left\| \mathop{\mathbb{E}}_{S_i}[\widehat{R}_{S_i}(h_i)] - \left(\widehat{R}^*_S(h_i) + \frac{1}{\mathbb{P}(h_i)}\sqrt{\frac{\log\frac{2}{\delta}}{2N}} \right) \right\| \tag{12}$$

Being limited to dataset instance S, we do not have access to subfunction h_i's true error $\mathbb{E}_{S_i}[\widehat{R}_{S_i}(h_i)]$. However, we can construct an estimate by taking the samples x, y that k includes in the subfunction's smooth empirical error $\widehat{R}^*_S(h_i)$ (that is, all the training samples, for positive valued k) and adjusting their error values to reflect what they would be if the sample did belong to a_i. This improves on $\widehat{R}^*_S(h_i)$, the weighted error of surrounding activation regions, by transforming it into a weighted error of the target activation region a_i itself. Let $g : \mathcal{X} \times \mathcal{Y} \to \mathcal{X} \times \mathcal{Y}$ be the shifting function that moves samples into region a_i, defined as $g(x, y) \triangleq \text{argmin}_{(x',y')\sim D} \|(x', y') - (x, y)\|$ s.t. $x' \in a_i$. Using the Taylor expansion, assuming an error function r with bounded first order gradients:

$$\forall (x, y) \in S : r(g(x, y); f) = r((x, y); f) + \mathcal{O}(\|(x, y) - g(x, y)\|) \tag{13}$$

Replacing the true error in eq. (12) with the estimate constructed using shifted samples:

$$\left\| \frac{1}{N} \sum_{(x,y)\in S} \frac{1}{w(x)} k(H(x), h_i)\, r(g(x, y); f) - \left(\widehat{R}^*_S(h_i) + \frac{1}{\mathbb{P}(h_i)}\sqrt{\frac{\log\frac{2}{\delta}}{2N}} \right) \right\| \tag{14}$$

$$\leq \left\| \frac{1}{N} \sum_{(x,y)\in S} \frac{1}{w(x)} k(H(x), h_i)\, \mathcal{O}(\|(x, y) - g(x, y)\|) \right\| + \frac{1}{\mathbb{P}(h_i)}\sqrt{\frac{\log\frac{2}{\delta}}{2N}} \tag{15}$$

where $w(x) \triangleq \sum_{j \in [1,C]} \mathbb{P}(h_j)\ k(H(x), h_j)$ is the weight normalization term. We draw 2 conclusions from this analysis. First, weighting value $k(h_i, h_j)$ for any h_i, h_j should decrease with the distance between their activation regions. This is evident from eq. (15) as x is in region $H(x)$ and $g(x, y)[0]$ is in region h_i, so larger distances should be penalized by smaller weights. However, $k(h_i, h_j)$ cannot be too low (for example 0 for all $h_j \neq h_i$ given any h_i) because the magnitude of $\frac{1}{\mathbb{P}(h_i)}$ would be large. Thus the best k combines error precision (upweight near regions and downweight far regions) with support (upweight near and far regions). The need to minimize weight with distance justifies restricting the search for k within function classes where output decreases with activation region distance, such as Gaussian functions of activation pattern distance. An example of this trade-off is illustrated in fig. 3.

Secondly, in practice there is no need to explicitly minimize eq. (15), which would involve estimating the Lipschitz constant for error function r. A simple alternative is to use a validation set metric that correlates with how well k, δ minimize eq. (15) (i.e. approximate the true error), such as the ability of the bound to discriminate between validation samples that are accurately or inaccurately predicted, and select k and δ such that they minimize the validation metric. This is the approach we take in our experiments.

2.4 From input-level bounds back to model-level bounds

Corollary 2.2 (Expectation reconstructed true error bound). *Taking an expectation over subfunctions yields the following model-level bound. In expectation over subfunctions h_i, the following holds with probability $> 1 - \delta$:*

$$\mathop{\mathbb{E}}_{h_i}[R^*_{\tilde{D}_S}(h_i)] \leq \mathop{\mathbb{E}}_{h_i}\left[\widehat{R}^*_S(h_i) + \frac{1}{\mathbb{P}(h_i)}\sqrt{\frac{\log \frac{2}{\delta}}{2N}}\right] = \overbrace{\widehat{R}_S(f) + C\sqrt{\frac{\log \frac{2}{\delta}}{2N}}}^{\textit{expectation reconstructed true error bound (E-RTEB)}} \quad . \tag{16}$$

Corollary 2.3 (Union reconstructed true error bound). *Taking a union bound over subfunctions yields the following model-level bound. For any $\delta > 0$, the following holds for all subfunctions h_i with probability $> 1 - \delta$:*

$$\mathop{\mathbb{E}}_{h_i}[R^*_{\tilde{D}_S}(h_i)] < \overbrace{\widehat{R}^*_S(h_i) + \frac{1}{\mathbb{P}(h_i)}\sqrt{\frac{\log \frac{2C}{\delta}}{2N}}}^{\textit{union reconstructed true error bound (U-RTEB)}} \quad , \tag{17}$$

proof in appendix B. It is clear that eq. (17) is the tighter and thus more useful bound, since the generalization gap scales with $\sqrt{\log C}$ instead of linearly with C (eq. (16)). These bounds share some similar characteristics with classical model-level bounds; see Mohri et al. (2018) for a review. For instance, larger datasets are better, because the bounds scale inversely with N, and generalization ability corresponds to sample efficiency, because in order to attain the same bound with smaller N, one decreases model complexity or the number of subfunctions C.

A key difference with existing work is the metric for model complexity is not the number of parameters (capacity), but the number of subfunctions or feasible activation regions (expressivity) combined with training sample density (eq. (17)). Whereas expressivity and density are dependent on training data and training procedure, capacity is not. We conjecture that there is a potentially interesting avenue to the recent double descent phenomenon (Belkin et al., 2019; Nakkiran et al., 2019) where it was observed, counter to accepted wisdom, that generalization in deep networks can be improved by increasing the number of parameters. Whereas Belkin et al. (2019) proposed the norm of network weights as an alternative complexity measure to the number of parameters; eq. (17) suggests combining the number of subfunctions (equivalently, number of activation regions) with training sample density of the activation regions. An intuitive interpretation of this bound is in order to maximise generalization, one should minimize the number of distinct behaviours within the model, whilst ensuring the data support for each of those behaviours is above some threshold, because generalization is strongly penalized by low training sample density.

Minimizing the number of binary activation patterns across layers corresponds to minimizing worst-case entropy of the binary representation, since the worst-case distribution for entropy is uniform and

entropy of uniform distributions scales with the number of states (proof in appendix C). Therefore from an information-theoretic perspective, minimizing eq. (17) over models corresponds to maximizing information loss from input space to feature space, or maximising the abstractness of the model's data representation. This is the same principle used by the information bottleneck (Tishby et al., 2000; Ahuja et al., 2021), which minimizes an upper bound on the entropy of continuous network representations, and implicitly by sparse factor graphs (Goyal & Bengio, 2020) and feature discretization methods (Liu et al., 2021) including discrete output unsupervised learning methods (Ji et al., 2019), which are all methods that build low expressivity into the model as a prior for improved generalization. Information loss also increases invariance of the feature representation with respect to changes in input sample, which is similar to Ganin et al. (2016); Sun & Saenko (2016) that impose invariance of the feature distribution with respect to changes in input datasets. In short, this work supports the idea that information loss in model representations is key for robust generalization, by providing a generalization error bound that explicitly scales with model expressivity.

Data represenations that satisfy the criterion of low expressivity and high training sample density (eq. (17)) exhibit a high degree of *abstraction* and *usage* across training data, by definition. These two measures are co-dependent; for example decreasing the number of activation regions by removing a particular hyperplane edge in shattered input space monotonically increases training sample density across activation regions, since it monotonically increases in the newly enlarged region and stays constant elsewhere. This explicitly illustrates how increasing abstraction improves robustness of the activation pattern to low level data variations, by definition of enlarging the input region of the pattern. (However expressivity and density are not completely redundant measures since one can modify density across regions by moving hyperplanes without changing the total number of regions.) Meanwhile, information can only monotonically decrease with the application of successive non-stochastic functions (Ji et al., 2019). Furthermore, note that summing the bound in eq. (17) over subfunctions implies a uniform optimal distribution of densities across activation regions, all else equal, because $\forall P \in \mathcal{R}^C_{\geq 0}$:

$$\min_P \sum_{i\in[1,C]} \frac{1}{P_i} \text{ s.t. } \sum_{i\in[1,C]} P_i = 1 \text{ implies } \forall i \in [1,C] : P_i = \frac{1}{C}, \quad (18)$$

proof in appendix D, as one would expect from the shape of the reciprocal function since it penalizes small densities disproportionately to larger ones. In summary, this suggests that piecewise-linear neural networks that generalize optimally resemble a chain of maximal information "squeezes" or losses of sample information (C in eq. (17)), where samples are evenly partitioned by activation patterns (eq. (18)) and empirical training error is still minimized i.e. training data can still be fitted ($\widehat{R}^*_S(h_i)$ in eq. (17)). Interestingly, this is exactly the kind of efficient hierarchical representation implemented by ontologies such as balanced k-d trees (Cormen et al., 2009), which minimize the number of input regions given any fixed number of discriminative hyperplanes by ensuring each hyperplane only partitions one superset region, whilst maximizing search efficiency with balanced partitioning resulting in uniform density across input regions (fig. 4).

Figure 4: Input regions for a balanced k-d tree on 2D space. Color denotes unique decision set.

3 Related work

Input-dependent generalization error bounds brings together several areas of machine learning research that are not typically combined. The first is generalization error bounds intended for model selection (Mohri et al., 2018; Akaike, 1974). The second is sample-level metrics intended for out-of-distribution or unreliable in-distribution sample selection (Ovadia et al., 2019). The third is work on linear activation regions, motivated by characterizing neural network expressivity (Montúfar et al., 2014; Raghu et al., 2017; Hanin & Rolnick, 2019).

So far we have focused on generalization error bounds and linear activation regions; now we review existing ideas for sample selection, which is necessary to understand baselines in the practical experiments. Well known sample uncertainty or unreliability metrics include the maximum response of the final softmax prediction layer (Hendrycks & Gimpel, 2017; Geifman & El-Yaniv, 2017; Cordella et al., 1995; Chow, 1957), its entropy (Shannon, 1948), or its top-two margin (Scheffer et al., 2001),

all conditioned on the input sample. Liang et al. (2017) combines maximum response with temperature scaling and input perturbations. Jiang et al. (2018) combines the top-two margin idea with class distance. Some ideas use distance to prototypes in representation space, which is similar at high level to ours if one assumes prototypes are in high-density regions. Lee et al. (2018) trains a logistic regressor on layer-wise distances of a sample's features to its nearest class, with the idea that distance to features of the nearest class should scale with unreliability. This was shown to outperform Liang et al. (2017). Sehwag et al. (2021) is an unsupervised variant of Lee et al. (2018). Tack et al. (2020) clusters feature representations instead of using classes, using distance to nearest cluster as unreliability. In Bergman & Hoshen (2020) the clusters are defined by input transformations; we were unable to get this working in our setting as models appear to suffer from feature collapse across transformations when not trained explicitly for transformation disentanglement. Non-parametric kernel based methods such as Gaussian processes provide measures of uncertainty that also scale with distance from samples and can be appended to a neural network base (Liu et al., 2020). Zhang et al. (2020) assume density in latent space is correlated to reliability, using residual flows (Chen et al., 2019) for the density model. If multiple models trained on the same dataset are available (which we do not assume), one could use ensemble model metrics such as variance, max response or entropy (Lakshminarayanan et al., 2016); an ensemble can also be simulated in a single model with Monte Carlo dropout (Geifman & El-Yaniv, 2017; Gal & Ghahramani, 2016).

Many of these works seek to predict whether a sample is out-of-distribution (OOD) for its own sake, which is an interesting problem, but we care about 1) epistemic uncertainty in general, including in-distribution misclassification, not just OOD 2) in the context of the main model trained for a practical task, or in other words, exposing what the task model does not know using the task model itself, as opposed to training separate models on the data distribution specifically for outlier detection.

4 Experiments

We tested the ability of the input-conditioned bound in eq. (11) to predict out-of-distribution and misclassified in-distribution samples. Taking pre-trained VGG16 and ResNet50 models for CIFAR100 and CIFAR10, we computed area under false positive rate vs. true positive rate (AUROC) and area under coverage vs. effective accuracy (AUCEA) for each method. Definitions are given in appendix G. These metrics treat predicting unreliable samples as a binary classification problem, where for out-of-distribution, ground truth is old distribution/new distribution, and for misclassified in-distribution, ground truth is classified correct/incorrect. Method output is accept/reject. All methods produce a metric per sample that is assumed to scale with unreliability, so 1K thresholds for discretizing into accept/reject decisions were uniformly sampled across the maximum test set range, yielding the AUROC and AUCEA curves. In-distribution misclassification validation set was used to select hyperparameters, i.e. we use a realistic setting where OOD data is truly unseen for all parameters. For our method, we used Gaussian weighting with standard deviation ρ ($k(h_i, h_j) = e^{-\text{Hamming}(p_i,p_j)^2/(2\rho^2)}$) and took log of the bound to suppress large magnitudes.

There are usually some caveats when porting theory to practical experiments, and in this case there were 2. First, for tractability on deep networks, we use a coarse partitioning of the network by taking activations from the last ReLU layer only, so the subfunctions are piecewise linear instead of purely linear. In this case activation regions are still disjoint, eq. (11) becomes a bound on piecewise linear

Table 1: Summary of out-of-distribution and misclassified in-distribution results, by difference to the top performing method in each architecture $\times$ dataset setting. Values are difference in AUROC and average $\pm$ standard deviation is shown over all architecture $\times$ dataset settings. Higher is better.

	Out-of-distr.	Misc. in-distr.	Average
Residual flows density (Chen et al., 2019)	-0.538 $\pm$ 2E-01	-0.356 $\pm$ 4E-02	-0.447 $\pm$ 1E-01
GP (Liu et al. (2020) w/ fixed features)	-0.204 $\pm$ 2E-01	-0.159 $\pm$ 1E-01	-0.181 $\pm$ 1E-01
Class distance (Lee et al., 2018)	-0.214 $\pm$ 1E-01	-0.334 $\pm$ 9E-02	-0.274 $\pm$ 1E-01
Margin (Scheffer et al., 2001)	-0.037 $\pm$ 2E-02	-0.007 $\pm$ 7E-03	-0.022 $\pm$ 1E-02
Entropy (Shannon, 1948)	-0.025 $\pm$ 2E-02	**-0.002 $\pm$ 2E-03**	-0.014 $\pm$ 1E-02
Max response (Cordella et al., 1995)	-0.034 $\pm$ 2E-02	-0.008 $\pm$ 8E-03	-0.021 $\pm$ 1E-02
MC dropout (Geifman & El-Yaniv, 2017)	-0.061 $\pm$ 3E-02	-0.048 $\pm$ 2E-02	-0.054 $\pm$ 2E-02
Cluster distance (Tack et al., 2020)	-0.052 $\pm$ 9E-02	-0.021 $\pm$ 7E-03	-0.036 $\pm$ 5E-02
Subfunctions (ours)	**-0.007 $\pm$ 1E-02**	-0.006 $\pm$ 4E-04	**-0.007 $\pm$ 6E-03**

Table 2: Results for models trained on CIFAR10 on out-of-distribution detection vs CIFAR100/SVHN. AUROC shown, higher is better. For equivalent table on CIFAR100, see table 5.

	→ CIFAR100		→ SVHN	
	VGG16	ResNet50	VGG16	ResNet50
Residual flows density	0.513 ± 2E-04	0.513 ± 1E-04	0.084 ± 1E-04	0.084 ± 1E-04
GP	0.810 ± 1E-02	0.575 ± 2E-02	0.844 ± 2E-02	0.473 ± 9E-02
Class distance	0.673 ± 7E-02	0.468 ± 3E-02	0.806 ± 6E-02	0.462 ± 1E-01
Margin	0.829 ± 2E-03	0.825 ± 5E-03	0.854 ± 4E-02	0.856 ± 2E-02
Entropy	0.853 ± 2E-03	0.822 ± 6E-03	0.869 ± 3E-02	0.858 ± 3E-02
Max response	0.829 ± 3E-03	0.827 ± 5E-03	0.850 ± 4E-02	0.858 ± 2E-02
MC dropout	0.776 ± 6E-03	0.807 ± 3E-03	0.778 ± 6E-02	0.838 ± 3E-02
Cluster distance	**0.862 ± 4E-03**	**0.867 ± 3E-03**	**0.870 ± 5E-02**	**0.892 ± 1E-02**
Subfunctions (ours)	**0.858 ± 4E-03**	**0.862 ± 2E-03**	**0.886 ± 2E-02**	**0.915 ± 2E-02**

Table 3: Results for models trained on CIFAR10. Predicting misclassification on in-distribution test. Higher is better. For equivalent table on CIFAR100, see table 6.

	VGG16		ResNet50	
	AUCEA	AUROC	AUCEA	AUROC
Residual flows density	0.442 ± 5E-02	0.520 ± 1E-02	0.492 ± 3E-02	0.577 ± 1E-02
GP	0.983 ± 1E-03	0.865 ± 8E-03	0.943 ± 5E-03	0.625 ± 2E-02
Class distance	0.948 ± 2E-02	0.669 ± 8E-02	0.900 ± 3E-02	0.471 ± 6E-02
Margin	0.982 ± 2E-03	0.900 ± 4E-03	0.848 ± 1E-02	**0.894 ± 4E-03**
Entropy	**0.989 ± 1E-04**	**0.914 ± 3E-03**	**0.984 ± 1E-03**	0.890 ± 5E-03
Max response	0.980 ± 3E-03	0.898 ± 4E-03	0.832 ± 1E-02	**0.895 ± 5E-03**
MC dropout	0.982 ± 6E-04	0.845 ± 6E-03	0.976 ± 2E-03	0.868 ± 1E-02
Cluster distance	0.988 ± 4E-04	0.901 ± 2E-03	0.981 ± 2E-03	0.867 ± 2E-03
Subfunctions (ours)	**0.988 ± 3E-04**	**0.907 ± 3E-03**	**0.983 ± 2E-03**	0.889 ± 6E-03

subfunction error and still holds. Second, computing the set of feasible activation regions (or even the size of this set) is currently intractable, so we use the full bit space, $P = \{0, 1\}^M$. This means the bound is computed in an altered subfunction space where some infeasible subfunctions are assigned a non-zero density, affecting the weight normalization. An upside of using the full bit space is it allows us to make some computational savings when computing the bound, detailed in appendix E.

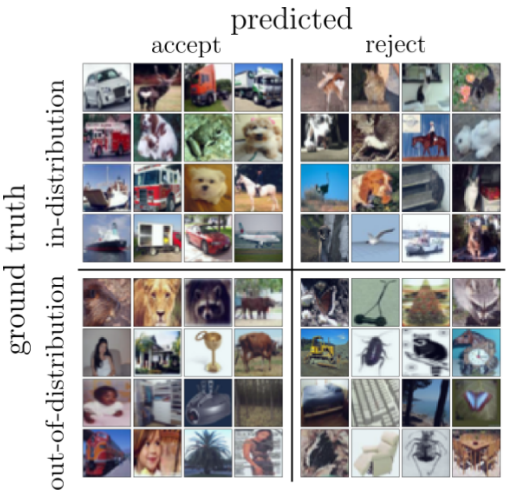


Figure 5: Sample confusion matrix, OOD for CIFAR10 → CIFAR100 on ResNet50. Random samples from top 20% in each quadrant shown.

These 2 limitations mean that the performance attained by the method in the experiments, already good, is actually a lower bound that is likely improvable if more tractable implementations are found.

Subfunctions and entropy were found to be the top 2 methods overall in each category (table 1), with subfunctions better on average. Cluster distance also performed well on CIFAR10, but was penalized by poor performance on CIFAR100 and particularly VGG16 (tables 5 and 6), which is a more difficult dataset for determining outlier status as the in-distribution classes are more finegrained. We conclude that subfunctions and entropy were good predictors of unreliability for both in-distribution and out-of-distribution scenarios. The AUCEA results for the in-distribution setting (tables 3 and 6) mean that using either to filter predictions would have raised effective model accuracy (accuracy of accepted samples) to $90 \sim 92\%$ from $70 \sim 73\%$ for the original CIFAR100 models, and to $98 \sim 99\%$ from

$91 \sim 92\%$ for the original CIFAR10 models (table 4), on average over thresholds. Entropy is simpler and computationally cheaper than subfunction error bound, but suffers from the drawback that it can only be computed exactly if model inference includes a discrete probabilistic variable. This is the case for these experiments but not in general (e.g. consider MSE objectives), whereas our method does not have this restriction as it bounds risk more generally.

Empirically, we observed for subfunctions that reliable in-distribution images tended to be prototypical images for their class, whilst OOD images erroneously characterized as reliable tended to resemble the in-distribution classes, as one would expect (fig. 5). To test this hypothesis further, we took a CIFAR10 model and plotted the rankings of samples from each CIFAR100 class in fig. 6, where each box denotes the median and first and third quartiles. The classes are ordered by median. We identified the superclasses in CIFAR100 with the highest semantic overlap with CIFAR10 classes (mostly vehicles and mammals), whose classes are coloured green. It is clear from the correlation between green and lower unreliability ranking that subfunction error bounds rate OOD classes semantically closer to the training classes as more reliable. Note that even the exceptions towards the right are justified, because the inclusion of e.g. bicycles, lawn mowers and rockets in "vehicles" is questionable; certainly these objects do not correspond visually to the vehicle classes in CIFAR10. In addition, we ran the same plot for the other methods and found that in every case the correlation between reliability and semantic familiarity was less strong (appendix I).

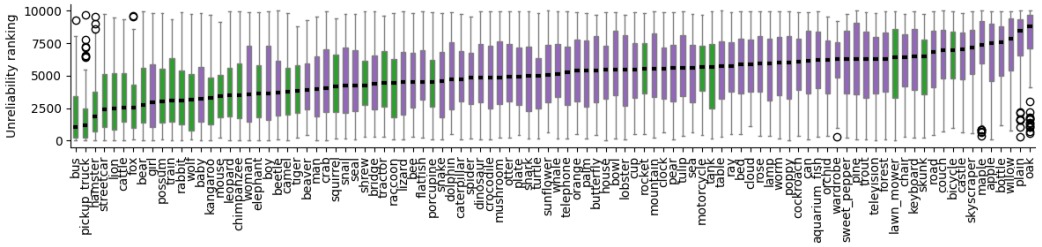


Figure 6: OOD for CIFAR10 $\rightarrow$ CIFAR100 on ResNet50. 10k CIFAR100 test samples were ranked by unreliability (log STEB). Boxplots summarize rankings per class (lower = less unreliable). Green denotes superclasses similar to CIFAR10: carnivores, omnivores, herbivores, mammals, vehicles.

5 Conclusion

We proposed a new form of input-conditioned generalization error bound for piecewise linear neural networks. This is done by shattering the network into activation regions and corresponding subfunctions before computing smoothed error according to a weighting function. The bound on this input-conditioned error indicates that dense regions are more reliable, whilst integrating over subfunctions suggests that fewer activation regions with higher training sample density are indicative of better generalization at model-level. In experiments we verified that the input-conditioned bound is empirically meaningful and can be used to detect misclassified in-distribution as well as OOD samples, but challenges remain in improving computational tractability.

ETHICS STATEMENT

Given that this work proposes a theoretical approach for assessing the generalization of neural networks, the authors do not foresee any specific negative social impacts.

REPRODUCIBILITY STATEMENT

The code for reproducing all results and figures has been released with this paper.

REFERENCES

Kartik Ahuja, Ethan Caballero, Dinghuai Zhang, Yoshua Bengio, Ioannis Mitliagkas, and Irina Rish. Invariance principle meets information bottleneck for out-of-distribution generalization. *arXiv preprint arXiv:2106.06607*, 2021.

Hirotugu Akaike. A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6):716–723, 1974.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Liron Bergman and Yedid Hoshen. Classification-based anomaly detection for general data. *arXiv preprint arXiv:2005.02359*, 2020.

Ricky TQ Chen, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. *arXiv preprint arXiv:1906.02735*, 2019.

Chi-Keung Chow. An optimum character recognition system using decision functions. *IRE Transactions on Electronic Computers*, pp. 247–254, 1957.

Luigi Pietro Cordella, Claudio De Stefano, Francesco Tortorella, and Mario Vento. A method for improving classification reliability of multilayer perceptrons. *IEEE Transactions on Neural Networks*, 6(5):1140–1147, 1995.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. *arXiv preprint arXiv:1705.08500*, 2017.

Anirudh Goyal and Yoshua Bengio. Inductive biases for deep learning of higher-level cognition. *arXiv preprint arXiv:2011.15091*, 2020.

Boris Hanin and David Rolnick. Deep ReLU networks have surprisingly few activation patterns. *Advances in Neural Information Processing Systems*, 32:361–370, 2019.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*, 2017.

Xu Ji, Joao F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9865–9874, 2019.

Heinrich Jiang, Been Kim, Melody Y Guan, and Maya R Gupta. To trust or not to trust a classifier. In *NeurIPS*, pp. 5546–5557, 2018.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *arXiv preprint arXiv:1807.03888*, 2018.

Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

Dianbo Liu, Alex Lamb, Kenji Kawaguchi, Anirudh Goyal, Chen Sun, Michael Curtis Mozer, and Yoshua Bengio. Discrete-valued neural communication. *arXiv preprint arXiv:2107.02367*, 2021.

Jeremiah Zhe Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax-Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *arXiv preprint arXiv:2006.10108*, 2020.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. *arXiv preprint arXiv:1402.1869*, 2014.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *arXiv preprint arXiv:1912.02292*, 2019.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua V Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. *arXiv preprint arXiv:1906.02530*, 2019.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *international conference on machine learning*, pp. 2847–2854. PMLR, 2017.

Omar Rivasplata, Ilja Kuzborskij, Csaba Szepesvári, and John Shawe-Taylor. Pac-bayes analysis beyond the usual bounds. *arXiv preprint arXiv:2006.13057*, 2020.

Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, pp. 309–318. Springer, 2001.

Vikash Sehwag, Mung Chiang, and Prateek Mittal. Ssd: A unified framework for self-supervised outlier detection. *arXiv preprint arXiv:2103.12051*, 2021.

Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pp. 443–450. Springer, 2016.

Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin. Csi: Novelty detection via contrastive learning on distributionally shifted instances. *arXiv preprint arXiv:2007.08176*, 2020.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Hongjie Zhang, Ang Li, Jie Guo, and Yanwen Guo. Hybrid models for open set recognition. In *European Conference on Computer Vision*, pp. 102–117. Springer, 2020.

A PROOF FOR THEOREM 2.1

This follows the Chernoff/Hoeffding bounding technique. Fix model f, patterns P and thus subfunctions set H. Let the true data distribution for each activation region a_i be D_i. Assume that $\tilde{D}_S$ is a distribution over size-N datasets that have the same activation region data distributions and dataset sizes (D_i and N_i, $\forall i \in [1, C]$) as S. Consider S as a dataset drawn iid from $\tilde{D}_S$. Without loss of generality, since dataset order is immaterial for the error metrics, assume the index n of sample S^n determines its activation region and distribution from which it is drawn independently of other samples. That is, $\forall n \in [1, N] : S^n \sim D_{J^n}$ iid for some fixed $J \in [1, C]^N$. Recall that D_i is constructed as $\mathbb{P}_{D_i}(x, y) = \mathbb{P}_D(x, y | x \in a_i)$. Call the inputs and targets X_S and Y_S, $\forall n \in [1, N] : (X_S^n, Y_S^n) \triangleq S^n$. Recall that empirical error function for samples is bounded: $\forall x, y \in (\mathcal{X} \times \mathcal{Y}) : 0 \leq r((x, y); f) \leq 1$ and weighting function k is non-negative valued.

Choose any subfunction $h_i \in H$, $i \in [1, C]$. Then $\forall \epsilon > 0, t > 0, \tilde{S} \sim \tilde{D}_S$ iid:

$$\mathop{\mathbb{P}}_{S}(\widehat{R}^*_S(h_i) - R^*_{\tilde{D}_S}(h_i) \geq \epsilon) = \mathop{\mathbb{P}}_{S}(\widehat{R}^*_S(h_i) - \mathop{\mathbb{E}}_{\tilde{S}}[\widehat{R}^*_{\tilde{S}}(h_i)] \geq \epsilon) \tag{19}$$

$$= \mathop{\mathbb{P}}_{S}(e^{t(\widehat{R}^*_S(h_i) - \mathbb{E}_{\tilde{S}}[\widehat{R}^*_{\tilde{S}}(h_i)])} \geq e^{t\epsilon}) \tag{20}$$

$$\leq e^{-t\epsilon} \mathop{\mathbb{E}}_{S}[e^{t(\widehat{R}^*_S(h_i) - \mathbb{E}_{\tilde{S}}[\widehat{R}^*_{\tilde{S}}(h_i)])}] \tag{21}$$

$$= e^{-t\epsilon} \mathop{\mathbb{E}}_{S}[e^{t \sum_{n=1}^{N} \left(\frac{k(H(X_S^n), h_i)\ r(S^n; f)}{N \sum_{j \in [1,C]} \mathbb{P}(h_j)\ k(H(X_S^n), h_j)} - \mathbb{E}_{\tilde{S}^n} \left[\frac{k(H(X_{\tilde{S}}^n), h_i)\ r(\tilde{S}^n; f)}{N \sum_{j \in [1,C]} \mathbb{P}(h_j)\ k(H(X_{\tilde{S}}^n), h_j)} \right] \right)}] \tag{22}$$

$$= e^{-t\epsilon} \prod_{n=1}^{N} \mathop{\mathbb{E}}_{S^n}[e^{t \left(\frac{k(H(X_S^n), h_i)\ r(S^n; f)}{N \sum_{j \in [1,C]} \mathbb{P}(h_j)\ k(H(X_S^n), h_j)} - \mathbb{E}_{\tilde{S}^n} \left[\frac{k(H(X_{\tilde{S}}^n), h_i)\ r(\tilde{S}^n; f)}{N \sum_{j \in [1,C]} \mathbb{P}(h_j)\ k(H(X_{\tilde{S}}^n), h_j)} \right] \right)}] \tag{23}$$

$$\leq e^{-t\epsilon} \prod_{n=1}^{N} e^{\frac{t^2 \left(\frac{1}{N\, \mathbb{P}(h_i)} - 0 \right)^2}{8}} \tag{24}$$

$$= e^{\frac{t^2}{8N\, \mathbb{P}(h_i)^2} - t\epsilon}, \tag{25}$$

by making use of the following:

1. eq. (20) $\rightarrow$ eq. (21) : Markov's inequality. For random variable $Z \geq 0$ and constant $a > 0$, $\mathbb{P}(Z \geq a) \leq \frac{\mathbb{E}[Z]}{a}$.
2. eq. (21) $\rightarrow$ eq. (22) : definition of $\widehat{R}^*_S(h_i)$ and linearity of expectation.
3. eq. (22) $\rightarrow$ eq. (23) : $\widehat{R}^*_S(h_i)$ is a sum over independently drawn samples. $\forall h_j \in H : \mathbb{P}(h_j)$ is constant because activation region dataset sizes are constant.
4. eq. (23) $\rightarrow$ eq. (24) : first bound the length of the range of the exponent. $\forall i \in [1, C], n \in [1, N] : 0 \leq \frac{k(H(X_S^n), h_i)\ r(S^n; f)}{N \sum_{j \in [1,C]} \mathbb{P}(h_j)\ k(H(X_S^n), h_j)} \leq \frac{k(H(X_S^n), h_i)}{N\, \mathbb{P}(h_i)\ k(H(X_S^n), h_i)} = \frac{1}{N\, \mathbb{P}(h_i)}$. Subtracting a constant does not change the length of the range of a random variable. Then apply Hoeffding's lemma: for random variable Z where $a \leq Z \leq b$ and $\mathbb{E}[Z] = 0$, then $\forall t > 0 : \mathbb{E}[e^{tZ}] \leq e^{\frac{t^2 (b-a)^2}{8}}$ holds.

Find the optimal t as the one that yields the tightest bound:

$$\nabla_t e^{\frac{t^2}{8N\, \mathbb{P}(h_i)^2} - t\epsilon} = 0, \qquad t = 4N\epsilon\, \mathbb{P}(h_i)^2, \tag{26}$$

which is a minimum because the second derivative is positive. Substitute into eq. (25) :

$$\mathop{\mathbb{P}}_{S}(\widehat{R}^*_S(h_i) - R^*_{\tilde{D}_S}(h_i) \geq \epsilon) \leq e^{-2N\epsilon^2\, \mathbb{P}(h_i)^2}. \tag{27}$$

The symmetric case can be proved in the same way:

$$\mathop{\mathbb{P}}_{S}(\widehat{R}^*_S(h_i) - R^*_{\tilde{D}_S}(h_i) \leq -\epsilon) = \mathop{\mathbb{P}}_{S}(R^*_{\tilde{D}_S}(h_i) - \widehat{R}^*_S(h_i) \geq \epsilon) \leq e^{-2N\epsilon^2\, \mathbb{P}(h_i)^2}, \tag{28}$$

specifically because swapping the order of the subtraction in eq. (23) does not change the value of the squared range used in eq. (24). Combining eq. (27) and eq. (28):

$$\mathop{\mathbb{P}}_{S}(|\widehat{R}^*_S(h_i) - R^*_{\tilde{D}_S}(h_i)| \geq \epsilon) \leq 2e^{-2N\epsilon^2\,\mathbb{P}(h_i)^2}. \tag{29}$$

Setting the right hand side to δ and solving for ϵ completes the derivation. Namely for any $\delta > 0$, the following hold with probability $\leq \delta$:

$$|\widehat{R}^*_S(h_i) - R^*_{\tilde{D}_S}(h_i)| \geq \frac{1}{\mathbb{P}(h_i)}\sqrt{\frac{\log\frac{2}{\delta}}{2N}} \tag{30}$$

$$R^*_{\tilde{D}_S}(h_i) - \widehat{R}^*_S(h_i) \geq \frac{1}{\mathbb{P}(h_i)}\sqrt{\frac{\log\frac{2}{\delta}}{2N}}. \tag{31}$$

Hence the following hold with probability $> 1 - \delta$:

$$R^*_{\tilde{D}_S}(h_i) < \widehat{R}^*_S(h_i) + \frac{1}{\mathbb{P}(h_i)}\sqrt{\frac{\log\frac{2}{\delta}}{2N}} \tag{32}$$

$$R^*_{\tilde{D}_S}(h_i) \leq \widehat{R}^*_S(h_i) + \frac{1}{\mathbb{P}(h_i)}\sqrt{\frac{\log\frac{2}{\delta}}{2N}}. \tag{33}$$

□.

B Proof for Corollary 2.3

Use shorthand $\hat{r}_i \triangleq \widehat{R}^*_S(h_i), r_i \triangleq R^*_{\tilde{D}_S}(h_i)$ to denote empirical error and true error. Start from eq. (30) which stated for any subfunction with index i and $\delta > 0$, the following holds with probability $\leq \delta$:

$$|\hat{r}_i - r_i| \geq \frac{1}{\mathbb{P}(h_i)}\sqrt{\frac{\log\frac{2}{\delta}}{2N}} \tag{34}$$

Take a union bound over all the subfunctions, i.e. with some probability $\leq C\delta$, where C is the number of subfunctions, these hold:

$$\mathrm{OR}_k\left[|\hat{r}_i - r_i| \geq \frac{1}{\mathbb{P}(h_i)}\sqrt{\frac{\log\frac{2}{\delta}}{2N}}\right] \tag{34}$$

$$\mathrm{OR}_k\left[r_i \geq \hat{r}_i + \frac{1}{\mathbb{P}(h_i)}\sqrt{\frac{\log\frac{2}{\delta}}{2N}}\right] \tag{35}$$

the latter since eq. (35) is less likely than eq. (34). So with probability $> 1 - C\delta$:

$$\mathrm{AND}_k\left[r_i < \hat{r}_i + \frac{1}{\mathbb{P}(h_i)}\sqrt{\frac{\log\frac{2}{\delta}}{2N}}\right] \tag{36}$$

Rewrite $\delta' = C\delta$. In summary, for any $\delta' > 0$, the following holds for all subfunctions h_i in the network, with probability $> 1 - \delta'$:

$$r_i < \hat{r}_i + \frac{1}{\mathbb{P}(h_i)}\sqrt{\frac{\log\frac{2C}{\delta'}}{2N}} \tag{37}$$

□.

C Proof for entropy of discrete uniform distribution

Let X by discrete random variable with n states and a uniform distribution. Then its entropy $\mathcal{H}(X)$ is given by:

$$\mathcal{H}(X) = -\sum_{x\in X}\frac{1}{n}\log\frac{1}{n} = -\log\frac{1}{n} = \log n \tag{38}$$

which increases monotonically with n □.

D PROOF FOR EQ. (18)

$\forall P \in \mathcal{R}^C_{\geq 0}$, write the Lagrangian for eq. (18) as:

$$\mathcal{L}(P, \lambda) = \sum_{i \in [1,C]} \frac{1}{P_i} - \lambda \Bigg(\Bigg(\sum_{i \in [1,C]} P_i \Bigg) - 1 \Bigg). \tag{39}$$

Setting the partial derivatives to 0:

$$\forall i : \frac{\partial \mathcal{L}}{\partial P_i} = \frac{1}{P_i^2} - \lambda = 0 \rightarrow P_i = \frac{1}{\sqrt{\lambda}}, \tag{40}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = -\Bigg(\sum_{i \in [1,C]} P_i \Bigg) + 1 = 0. \tag{41}$$

Equation (40) and eq. (41) combined imply the uniform distribution, $\forall i : P_i = \frac{1}{C}$. Further, this minimizes rather than maximizes $\sum_{i \in [1,C]} \frac{1}{P_i}$. Proof by contradiction: for $C = 2$, $\frac{1}{0.8} + \frac{1}{0.2} < 2\frac{1}{0.5} \rightarrow \perp$ $\square$.

E EFFICIENT COMPUTATION OF BOUND

For normalization in eq. (5), we reduce computational complexity from exponential $\mathcal{O}(N 2^M)$ to linear $\mathcal{O}(M)$ (derivations below):

$$\sum_{l \in [1,C]} \sum_{\substack{j \in [1,C] \\ \text{s.t. } N_j > 0}} N_j \, k(h_l, h_j) = N \sum_{b=0}^{M} d(b) \binom{M}{b} \triangleq u, \tag{42}$$

and for normalization in eq. (10), from exponential $\mathcal{O}(N^2 2^M)$ to polynomial $\mathcal{O}(N^2 + M^3)$:

$$\forall_{\substack{l \in [1,C] \\ \text{s.t. } N_l > 0}} : \sum_{j \in [1,C]} \mathbb{P}(h_j) \, k(h_l, h_j) = \frac{1}{u} \sum_{\substack{i \in [1,C] \\ \text{s.t. } N_i > 0}} N_i \, z(\text{Hamming}(p_l, p_i)), \tag{43}$$

$$\text{where } \forall a \in [0, M] : z(a) \triangleq \sum_{b=0}^{M} d(b) \sum_{c=0}^{b} \binom{a}{c} \binom{M-a}{b-c} d(a + (b-c) - c). \tag{44}$$

Normalization term in eq. (5). Computed once for the network.

$$\sum_{l \in [1,C]} \sum_{\substack{j \in [1,C] \\ \text{s.t. } N_j > 0}} N_j \, k(h_l, h_j) = \sum_{\substack{j \in [1,C] \\ \text{s.t. } N_j > 0}} N_j \sum_{l \in [1,C]} k(h_l, h_j) = N \sum_{b=0}^{M} d(b) \binom{M}{b} \triangleq u, \tag{45}$$

where the trick is that $\sum_{l \in [1,C]} k(h_l, h_j) = \sum_{l \in [1,C]} d(\text{Hamming}(p_l, p_j))$ is the same for all h_j due to completeness of the bit space P, and the specific value can be found by looping through all possible Hamming distances b. This reduces the computational complexity from $\mathcal{O}(N 2^M)$ to $\mathcal{O}(M)$, because the number of populated activation regions, i.e. length of loop over j, is upper bounded by the number of samples N, and $\binom{M}{b}$ can be iteratively updated in $\mathcal{O}(1)$ in the loop over b:

$$\binom{M}{b} = \binom{M}{b-1} \frac{M - (b-1)}{b}. \tag{46}$$

Normalization term in eq. (10). Computed $\forall l \in [1, C]$ s.t. $N_l > 0$:

$$\sum_{j\in[1,C]} \mathbb{P}(h_j)\; k(h_l, h_j) \tag{47}$$

$$= \frac{1}{u} \sum_{j\in[1,C]} \sum_{\substack{i\in[1,C] \\ \text{s.t. } N_i>0}} k(h_l, h_j)\; k(h_j, h_i)\; N_i \tag{48}$$

$$= \frac{1}{u} \sum_{\substack{i\in[1,C] \\ \text{s.t. } N_i>0}} N_i \sum_{j\in[1,C]} k(h_l, h_j)\; k(h_j, h_i) \tag{49}$$

$$= \frac{1}{u} \sum_{\substack{i\in[1,C] \\ \text{s.t. } N_i>0}} N_i \overbrace{\sum_{b=0}^{M} \overbrace{d(b)}^{\text{from } k(h_l,h_j)} \sum_{c=0}^{b} \binom{\text{Hamming}(p_l,p_i)}{c} \binom{M-\text{Hamming}(p_l,p_i)}{b-c}}^{=z(\text{Hamming}(p_l,p_i))\text{ (eq. (44))}} \cdot \overbrace{d(\text{Hamming}(p_l,p_i)+(b-c)-c)}^{\text{from } k(h_j,h_i)}. \tag{50}$$

Note the loops over i and l only need to range over populated activation regions due to the inclusion of their sample counts (N_i and N_l) as multiplicative factors in their loop contents, which means the iterations are bounded by $\mathcal{O}(N)$. The problematic loop is over j, i.e. all subfunctions/activation regions whether populated by samples or not, which is $\mathcal{O}(2^M)$ when P is the full bit space $\{0,1\}^M$. However, similar to eq. (45) above, it is also this fullness that we exploit.

Now we explain eq. (50) in detail. The value $z(\text{Hamming}(p_l, p_i))$ corresponds to the sum over j in eq. (49): loop over every bit pattern p_j in P, multiply its closeness to p_l with its closeness to p_i (both fixed outside this loop), and sum. Now group p_j by bit distance to p_l, b. Consider one instance of b (fig. 7). Within this group for b, the closeness of every p_j to p_l is constant, namely equal to $d(b)$. But they have different distances to p_i. However we know the number of bits that are different between p_l and p_i: this is $\text{Hamming}(p_l, p_i)$. Consider a specific instance of p_j. Visualize p_l turning into p_j by flipping bit one at a time; each will either bring the pattern closer to or further away from p_i. There is a budget of b different bits to flip to reach p_j. Say c of those flips brought it closer (i.e. turned the bit value at that position into the same as p_i's). Then we know $b-c$ brought it further. So the number of different bits between p_j and p_i is exactly $\text{Hamming}(p_l, p_i) + (b-c) - c$. And the number of p_j that fit this description is the number of ways of choosing c from the different bits between p_l and p_i multiplied by the number of ways of choosing $b-c$ from the shared bits between p_l and p_i: $\binom{\text{Hamming}(p_l,p_i)}{c}\binom{M-\text{Hamming}(p_l,p_i)}{b-c}$.

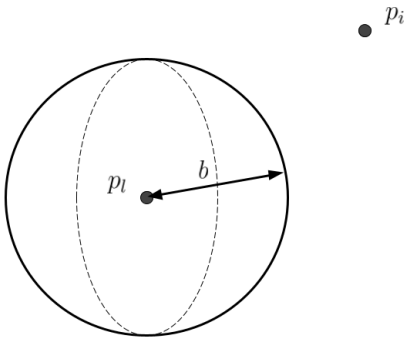


Figure 7: Sphere of p_j that are b bits away from p_l. For illustration only.

Conveniently, $\binom{q}{g} = 0$ for $g > q$, so there is no need to add special cases to exclude the cases where the number we seek to choose is greater than the number available.

Equation (50) enables computational complexity reduction of eq. (48), including the outer loop over l, from exponential to polynomial time: $\mathcal{O}(N^2 2^M)$ to $\mathcal{O}(N^2 + M^3)$ (excluding computation of u). The M^3 comes from constructing lookup table z, with the cubed power coming from looping over all possible values for $\text{Hamming}(p_l, p_i)$, b and c. This subsumes M^2 to compute $\forall q \in [0, M], \forall g \in [0, M] : \binom{q}{g}$ in the same iterative manner as eq. (46).

F ADDITIONAL RESULTS TABLES

Table 4: Test accuracy of original models.

Dataset	Model	Accuracy
CIFAR100	VGG16	0.704 ± 1E-03
CIFAR100	ResNet50	0.729 ± 8E-03
CIFAR10	VGG16	0.920 ± 2E-03
CIFAR10	ResNet50	0.908 ± 5E-03

Table 5: Model trained on CIFAR100. Out-of-distribution detection (AUROC) vs CIFAR10/SVHN.

	→ CIFAR10		→ SVHN	
	VGG16	ResNet50	VGG16	ResNet50
Residual flows density	0.495 ± 2E-04	0.495 ± 2E-04	0.090 ± 2E-04	0.090 ± 2E-04
GP	0.708 ± 6E-03	0.404 ± 3E-02	**0.830 ± 3E-02**	0.396 ± 8E-02
Class distance	0.627 ± 4E-02	0.513 ± 4E-02	**0.833 ± 3E-02**	0.579 ± 1E-01
Margin	0.716 ± 2E-03	0.736 ± 3E-03	0.771 ± 5E-03	0.790 ± 3E-02
Entropy	**0.725 ± 3E-03**	0.745 ± 3E-03	0.786 ± 6E-03	**0.814 ± 4E-02**
Max response	0.719 ± 3E-03	0.740 ± 3E-03	0.776 ± 6E-03	0.799 ± 3E-02
MC dropout	0.693 ± 3E-03	0.725 ± 4E-03	0.771 ± 1E-02	0.795 ± 3E-02
Cluster distance	0.639 ± 1E-02	**0.754 ± 4E-03**	0.561 ± 2E-02	**0.810 ± 5E-02**
Subfunctions (ours)	**0.738 ± 2E-03**	**0.750 ± 7E-03**	0.797 ± 8E-03	0.807 ± 3E-02

Table 6: Model trained on CIFAR100. Predicting misclassification on in-distribution test (AUCEA and AUROC).

	VGG16		ResNet50	
	AUCEA	AUROC	AUCEA	AUROC
Residual flows density	0.293 ± 1E-02	0.622 ± 1E-02	0.293 ± 2E-02	0.630 ± 1E-02
GP	0.882 ± 1E-03	0.803 ± 2E-03	0.670 ± 2E-02	0.384 ± 2E-02
Class distance	0.838 ± 3E-02	0.739 ± 6E-02	0.627 ± 7E-02	0.476 ± 3E-02
Margin	0.899 ± 2E-03	0.852 ± 4E-03	0.870 ± 6E-03	0.855 ± 4E-03
Entropy	0.895 ± 2E-03	**0.856 ± 5E-03**	**0.916 ± 4E-03**	**0.859 ± 4E-03**
Max response	**0.899 ± 2E-03**	0.853 ± 4E-03	0.864 ± 7E-03	**0.857 ± 4E-03**
MC dropout	0.894 ± 1E-03	0.828 ± 8E-03	0.898 ± 5E-03	0.841 ± 2E-03
Cluster distance	0.833 ± 9E-03	0.722 ± 2E-02	0.900 ± 5E-03	0.824 ± 7E-03
Subfunctions (ours)	**0.904 ± 1E-03**	**0.864 ± 3E-03**	**0.902 ± 4E-03**	0.827 ± 4E-03

G METRICS

Evaluation metrics were area under the graphs of coverage vs. effective accuracy (AUCEA) and false positive rate vs. true positive rate (AUROC), with the latter as standard for OOD experiments.

$$\text{coverage} = \frac{TP + FP}{TP + FP + TN + FN} \qquad \text{effective accuracy} = \text{precision} = \frac{TP}{TP + FP} \tag{51}$$

$$\text{false positive rate} = \frac{FP}{FP + TN} \qquad \text{true positive rate} = \frac{TP}{TP + FN}. \tag{52}$$

H ADDITIONAL EXPERIMENTAL DETAILS

Experiments were averaged over 5 random seeds. The classification models were trained with SGD optimization with learning rate 0.1, momentum 0.9, weight decay 5e-4 and standard schedules: 100 epochs with learning rate $^*0.1$ every 30 epochs (CIFAR10) and 200 epochs with learning rate $^*0.2$ every 60 epochs (CIFAR100). For each unreliability metric, each threshold yielded one set of accept/reject decisions for the test set which yielded one point in each evaluation graph, and area under graph was computed using the trapezoidal rule. For the in-distribution setting, AUCEA corresponds to effective model accuracy (i.e. of accepted samples) averaged over different thresholds, and thus can be compared against original model accuracy.

H.1 SUBFUNCTION ERROR BOUND HYPERPARAMETERS

The searched values for likelihood parameter δ were [0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9] and for weighting function standard deviation parameter ρ were [32, 48, 64, 98, 128, 192, 256, 384, 512] (VGG16) and [4, 6, 8, 12, 16, 24, 32, 48, 64] (ResNet50). Validation set AUCEA was used for selection. The selected values are in table 7.

Table 7: Hyperparameters used for subfunction error. Brackets denote number of seeds.

Dataset	Model	ρ	δ
CIFAR100	VGG16	128.0 (#: 5)	0.001 (#: 5)
CIFAR100	ResNet50	16.0 (#: 5)	0.001 (#: 4), 0.1 (#: 1)
CIFAR10	VGG16	48.0 (#: 3), 98.0 (#: 1), 64.0 (#: 1)	0.001 (#: 4), 0.9 (#: 1)
CIFAR10	ResNet50	16.0 (#: 3), 24.0 (#: 2)	0.001 (#: 4), 0.3 (#: 1)

H.2 DATASET STATISTICS

All results infer unreliability of the test sets. The datasets are publicly available.

Dataset	Train	Validation	Test
CIFAR10 (Krizhevsky, 2009)	42500	7500	10000
CIFAR100 (Krizhevsky, 2009)	42500	7500	10000
SVHN (Netzer et al., 2011)	62269	10988	26032

H.3 COMPUTATIONAL RESOURCES

Experiments were run given a shared cluster of machines with approximately 140 GPUs. Jobs required less than 24GB GPU memory. For subfunction error, normalization constants were computed first in a pre-computation phase. This took up most of the runtime and was parallelized by splitting jobs by architecture, dataset, seed and ρ; each job took approximately 20 minutes. Subsequent inference on the test sets (i.e. computing unreliability of unseen samples) took approximately 2 - 10 minutes per combination of architecture, dataset and seed.

I ADDITIONAL BOXPLOTS FOR OTHER METHODS

Settings apart from the method are the same as fig. 6. We measured the Spearman correlation coefficient between semantic similarity (green=1, purple=0) and reliability (- median unreliability rank for each class) and found it was lower for all baselines compared to subfunctions, which had a correlation coefficient of 0.511. The closest baseline method was MC dropout (0.433).

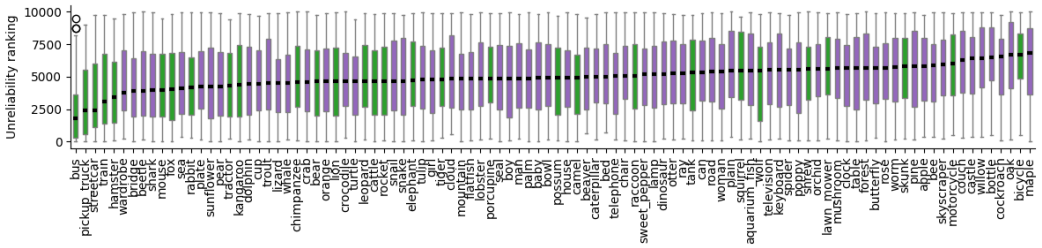


Figure 8: Entropy.

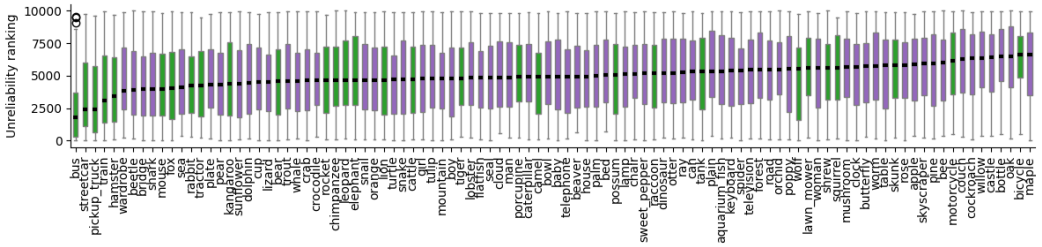


Figure 9: Max response.

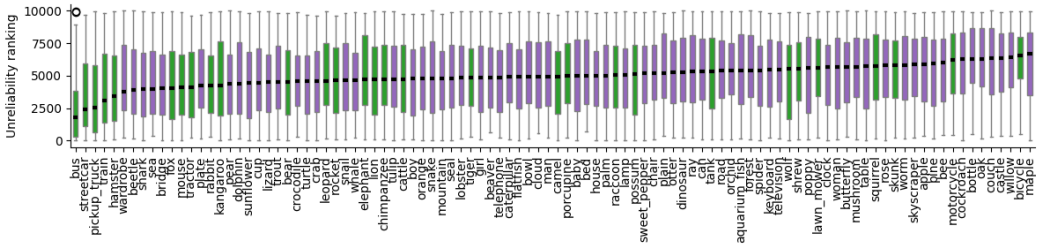


Figure 10: Margin.

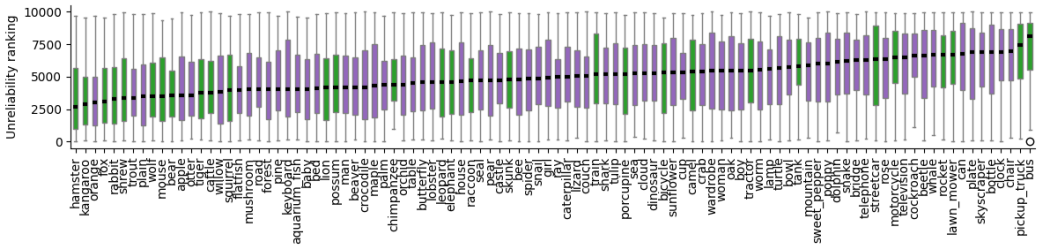


Figure 11: Class distance.

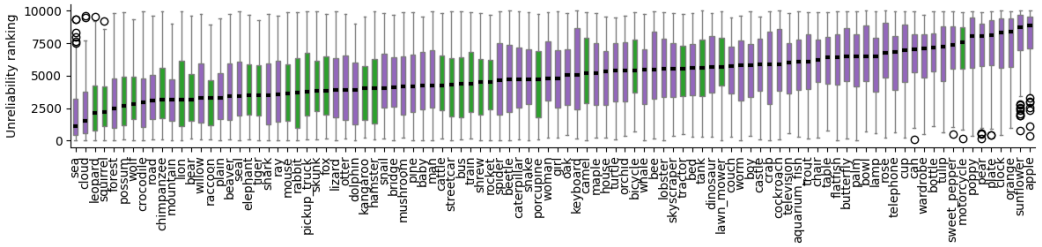


Figure 12: GP.

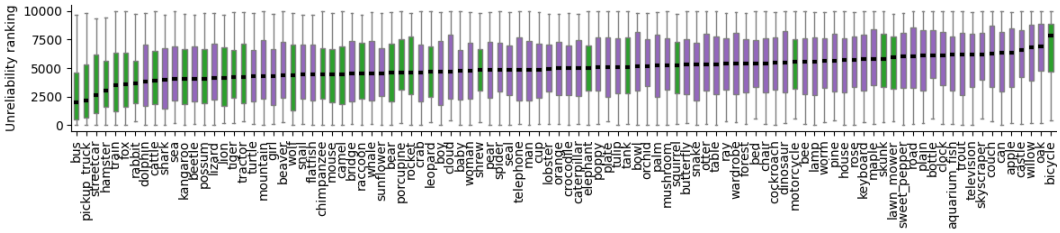


Figure 13: MC dropout.

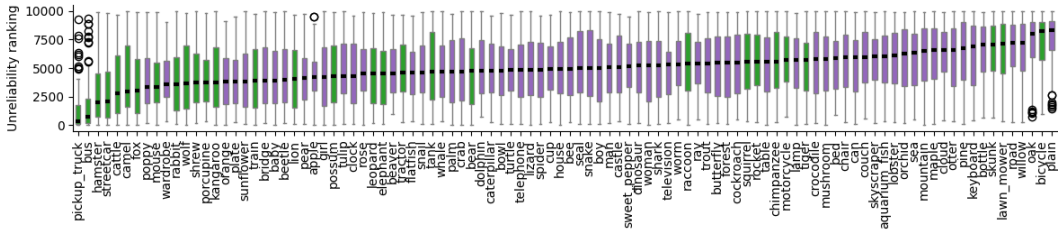


Figure 14: Cluster distance.

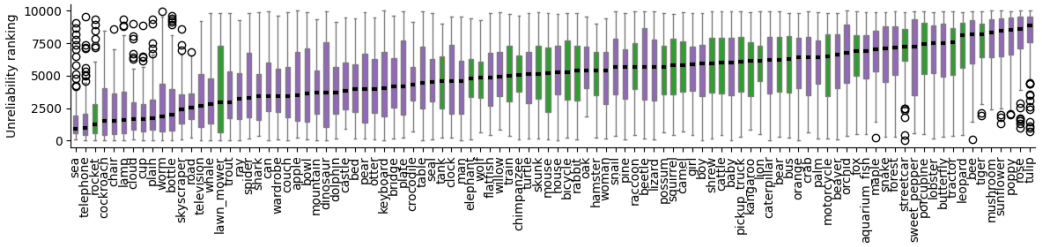


Figure 15: Residual flows density.