Detecting Misclassification Errors in Neural Networks with a Gaussian Process Model

Xin Qiu,[1] **Risto Miikkulainen** [1, 2]
[1] Cognizant AI Labs
[2] The University of Texas at Austin
qiuxin.nju@gmail.com, risto@cognizant.com

Abstract

As neural network classifiers are deployed in real-world applications, it is crucial that their failures can be detected reliably. One practical solution is to assign confidence scores to each prediction, then use these scores to filter out possible misclassifications. However, existing confidence metrics are not yet sufficiently reliable for this role. This paper presents a new framework that produces a quantitative metric for detecting misclassification errors. This framework, RED, builds an error detector on top of the base classifier and estimates uncertainty of the detection scores using Gaussian Processes. Experimental comparisons with other error detection methods on 125 UCI datasets demonstrate that this approach is effective. Further implementations on two probabilistic base classifiers and two large deep learning architecture in vision tasks further confirm that the method is robust and scalable. Third, an empirical analysis of RED with out-of-distribution and adversarial samples shows that the method can be used not only to detect errors but also to understand where they come from. RED can thereby be used to improve trustworthiness of neural network classifiers more broadly in the future.

1 Introduction

Classifiers based on Neural Networks (NNs) are widely deployed in many real-world applications (LeCun, Bengio, and Hinton 2015; Anjos et al. 2015; Alghoul et al. 2018; Shahid, Rappon, and Berta 2019). Although good prediction accuracies are achieved, it is usually not clear whether a particular prediction can be trusted, which is a severe issue especially in safety-critical domains such as healthcare (Selişteanu et al. 2018; Gupta et al. 2007; Shahid, Rappon, and Berta 2019), finance (Dixon, Klabjan, and Bang 2017), and automated driving (Janai et al. 2017; Hecker, Dai, and Van Gool 2018).

One way to estimate trustworthiness of a classifier prediction is to use its inherent confidence-related score, e.g., the maximum class probability (Hendrycks and Gimpel 2017), entropy of the softmax outputs (Williams and Renals 1997), or difference between the highest and second highest activation outputs (Monteith and Martinez 2010). However, these scores are unreliable and may even be misleading: Predictions often have high-confidence but are nevertheless incorrect (Provost, Fawcett, and Kohavi 1998; Guo et al. 2017; Nguyen, Yosinski, and Clune 2015; Goodfellow, Shlens, and Szegedy 2014; Amodei et al. 2016). In a practical setting, it is beneficial to have a detector that can raise a red flag whenever the predictions are likely to be wrong. A human observer can then evaluate such predictions, making the classification system safer.

Such a detector can be constructed by first developing quantitative metrics for measuring the reliability of predictions under different circumstances, and then setting a warning threshold based on users' preferred precision-recall tradeoff. Existing such methods can be categorized into three types based on their focus: error detection, which aims to detect the natural misclassifications made by the classifier (Hendrycks and Gimpel 2017; Jiang et al. 2018; Corbière et al. 2019); out-of-distribution (OOD) detection, which identifies samples that are from different distributions compared to training data (Liang, Li, and Srikant 2018; Lee et al. 2018a; Devries and Taylor 2018); and adversarial sample detection, which filters out samples from adversarial attacks (Lee et al. 2018b; Wang et al. 2019; Aigrain and Detyniecki 2019).

Among these categories, error detection, also called misclassification detection (Jiang et al. 2018) or failure prediction (Corbière et al. 2019), is the most challenging (Aigrain and Detyniecki 2019) and most underexplored. For instance, Hendrycks and Gimpel (2017) defined a baseline error detection score using the maximum class probability after the softmax layer. Although this baseline is a good starting point, it is ineffective in some cases, indicating that there is room for improvement (Hendrycks and Gimpel 2017). Jiang et al. (2018) proposed Trust Score, which measures the similarity between the original classifier and a modified nearest-neighbor classifier. The main limitation of this method is scalability: the Trust Score may provide no or negative improvement over the baseline for high-dimensional data. ConfidNet (Corbière et al. 2019) builds a separate NN model to learn the true class probablity, i.e. softmax probability for the ground-truth class. However, ConfidNet itself is a standard NN, so its detection scores may be unreliable or misleading: A random input may generate a random detection score, and ConfidNet does not indicate uncertainty of these detection scores. Moreover, none of these methods can differentiate natural classifier errors from risks caused by OOD or adversarial samples; if a detector could do that, it would

be easier for practitioners to fix the problem, e.g., by retraining the original classifier or by applying better preprocessing techniques to filter out OOD or adversarial data.

To meet these challenges, a new framework is developed in this paper for error detection in NN classifiers. The main idea is to learn to predict the correctness of a classification result with a Gaussian Process (GP) model. The new system, referred to as RED (Residual-based Error Detection), not only produces an enhanced error detection score based on the original maximum class probability, but also provides a quantitative uncertainty estimation of that score. As a result, misclassification errors can be detected more reliably. Note that in this manner, RED is different from traditional confidence calibration methods (Platt 1999; Zadrozny and Elkan 2001, 2002; Guo et al. 2017), which do not improve misclassification detection.

The GP model in RED is constructed based on the RIO method for uncertainty estimation (Residual prediction with Input/Output kernel; Qiu, Meyerson, and Miikkulainen 2020). It is notable that the modified RIO is only a part of the RED framework, and RED has several fundamental differences compared to RIO. First, whereas the original RIO is only applicable to regression models, RED works with classification tasks, which are more common. Second, whereas RIO calibrates the outputs of base models (thus reducing prediction errors), RED generates a new detection score for error detection usage only; the original classifier outputs are unchanged, and the classification accuracy remains the same. Third, whereas RIO quantifies the predictive uncertainty of the base model, RED only quantifies the uncertainty of the detection score, which is separate from the base classifier.

RED is compared empirically to state-of-the-art error detection methods on 125 UCI datasets and two vision tasks, with implementations on one standard NN classifier, two probabilistic NN classifiers, and two deep NN architectures. The results demonstrate that the approach is effective and robust. A further empirical study with OOD and adversarial samples shows the potential of using RED to diagnose the sources of mistakes as well, thereby paving the way to a comprehensive approach for improving trustworthiness of neural network classifiers in the future.

2 Related Work

In the past two decades, a large volume of work was devoted to calibrating the confidence scores returned by classifiers. Early works include Platt Scaling (Platt 1999; Niculescu-Mizil and Caruana 2005), histogram binning (Zadrozny and Elkan 2001), isotonic regression (Zadrozny and Elkan 2002), with recent extensions like Temperature Scaling (Guo et al. 2017), Dirichlet calibration (Kull et al. 2019), and distance-based learning (Xing et al. 2020). These methods focus on reducing the difference between reported class probability and true accuracy, and generally the rankings of samples are preserved after calibration. As a result, the separability between correct and incorrect predictions is not improved. In contrast, RED aims at deriving a score that can differentiate incorrect predictions from correct ones better.

A related direction of work is the development of classifiers with rejection/abstention option. These approaches either introduce new training pipelines/loss functions (Bartlett and Wegkamp 2008; Yuan and Wegkamp 2010; Cortes, DeSalvo, and Mohri 2016), or define mechanisms for learning rejection thresholds under certain risk levels (Dubuisson and Masson 1993; Santos-Pereira and Pires 2005; Chow 2006; Geifman and El-Yaniv 2017). In contrast, RED assumes an existing pretrained NN classifier, and provides an additional metric for detecting potential errors made by this classifier, without specifying a rejection threshold.

Designing metrics for detecting potential risks in NN classifiers has become popular recently. While most approaches focus on detecting OOD (Liang, Li, and Srikant 2018; Lee et al. 2018a; Devries and Taylor 2018) or adversarial examples (Lee et al. 2018b; Wang et al. 2019; Aigrain and Detyniecki 2019), work on detecting natural errors, i.e., regular misclassifications not caused by external sources, is more limited. Ortega (1995) and Koppel and Engelson (1996) conducted early work in predicting whether a classifier is going to make mistakes, and Seewald and Fürnkranz (2001) built a meta-grading classifier based on similar ideas. However, these early works did not consider NN classifiers. More recently, Hendrycks and Gimpel (2017) and Haldimann et al. (2019) demonstrated that raw maximum class probability is an effective baseline in error detection, although its performance was reduced in some scenarios.

More elaborate techniques for error detection have also been developed recently. Mandelbaum and Weinshall (2017) proposed a detection score based on the data embedding derived from the penultimate layer of a NN. However, their approach requires modifying the training procedure in order to achieve effective embeddings. Jiang et al. (2018) introduced Trust Score to measure the similarity between a base classifier and a modified nearest-neighbor classifier. Trust Score outperforms the maximum class probability baseline in many cases, but negative improvement over baseline can be observed in high-dimensional problems, implying poor scalability of local distance computations. ConfidNet (Corbière et al. 2019) learns to predict the class probability of true class with another NN, while IntrospectionNet (Aigrain and Detyniecki 2019) utilizes the logit activations of the original NN classifier to predict its correctness. Since both models themselves are standard NNs, the detection scores returned by them may be arbitrarily high without any uncertainty information. Moreover, existing approaches for error detection cannot differentiate natural misclassification error from OOD or adversarial samples, making it difficult to diagnose the sources of risks. In contrast, RED explicitly reports its uncertainty about the estimated detection score, providing more reliable error detection. The uncertainty information returned by RED may also be helpful in clarifying the cause of classifier mistakes, as will be demonstrated in this paper.

3 Methodology

This section gives the general problem statement, reviews the RIO method for uncertainty estimation, and describes the technical details of RED.

3.1 Problem Statement

Consider a training dataset $\mathcal{D} = (\mathcal{X}, \mathbf{y}) = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, and a pretrained NN classifier that outputs a predicted label $\hat{y}_i$ and class probabilities for each class $\sigma_i = [\hat{p}_{i,1}, \hat{p}_{i,2}, \ldots, \hat{p}_{i,K}]$ given $\mathbf{x}_i$, where N is the total number of training points and K is the total number of classes. The problem is to develop a metric that can serve as a quantitative indicator for detecting natural misclassification errors made by the pretrained NN classifier.

3.2 RIO

The RIO method (Qiu, Meyerson, and Miikkulainen 2020) was developed to quantify point-prediction uncertainty in regression models. More specifically, RIO fits a GP to predict the residuals, i.e. the differences between ground-truth and original model predictions. It utilizes an I/O kernel, i.e. a composite of an input kernel and an output kernel, thus taking into account both inputs and outputs of the original regression model. As a result, it measures the covariances between data points in both the original feature space and the original model output space. For each new data point, a trained RIO model takes the original input and output of the base regression model, and predicts a distribution of the residual, which can be added back to the original model prediction to obtain both a calibrated prediction and the corresponding predictive uncertainty.

SVGP (Hensman, Fusi, and Lawrence 2013; Hensman, Matthews, and Ghahramani 2015) was used in RIO as an approximate GP to significantly reduce the computational cost. Both empirical results and theoretical analysis showed that RIO is able to consistently improve the prediction accuracy of base models as well as provide reliable uncertainty estimations. It therefore forms a promising foundation for improving reliability of error detection metrics as well.

3.3 RED

Classification models generate class probabilities as their outputs. Therefore, the maximum class probability is an inherent baseline metric for error detection (Hendrycks and Gimpel 2017; Haldimann et al. 2019). RED derives a more reliable detection score from this baseline by incorporating a GP model based on RIO. Since detecting errors in classification models is beyond the scope of the original RIO, two modifications are made to adapt RIO to this new domain.

First, since RIO was designed for single-output regression problems, it contains an output kernel only for scalar outputs. In RED, this original output kernel is extended to multiple outputs, i.e. to vector outputs such as those of the final softmax layer of a NN classifer, representing estimated class probabilities for each class. This modification allows RIO to access more information from the classifier outputs. This new variant of RIO is hereinafter referred to as mRIO ("m" for multi-output).

Second, RIO estimates the point-prediction uncertainty by predicting the residuals between ground-truths and model outputs, both of which are limited to one-dimensional continuous space. However, ground-truth labels in classification problems are in a categorical space. Therefore, a different problem needs to be constructed: Instead of learning to reach the ground-truths directly, RED learns to predict whether the original prediction is correct or not. A target detection score is assigned to each training data point according to whether it is correctly classified by the base model. The residual between this target score and the original maximum class probability is calculated, and an mRIO model is trained to predict these residuals. Given a new data point, the trained mRIO model combined with the base classifier thus provides an aggregated score for detecting misclassification errors. Note that the outputs of base classifiers are not changed.

Figure 1 illustrates the RED training and deployment processes conceptually, and Algorithm 1 specifies them in detail. In the training phase, the first step is to define a target detection score c_i for each training sample $(\mathbf{x}_i, y_i, \hat{y}_i, \sigma_i)$. In nature, any functions that assign target values to correct and incorrect predictions differently can be used. For simplicity, the Kronecker delta $\delta_{y_i,\hat{y}_i}$ is used in this work: all training samples that are correctly predicted by the original NN classifier receive 1 as the target detection score, and those that are incorrectly predicted receive 0. The validation dataset during the original NN training is included

Algorithm 1: RED training and deployment procedures

Require:

$(\mathcal{X}, \mathbf{y}) = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$: training data

$\hat{\mathbf{y}} = \{\hat{y}_i\}_{i=1}^N$: labels predicted by original NN classifier on training data

$\boldsymbol{\sigma} = \{\sigma_i = [\hat{p}_{i,1}, \hat{p}_{i,2}, \ldots, \hat{p}_{i,K}]\}_{i=1}^N$: softmax outputs of original NN classifier on training data

$\hat{\mathbf{c}} = \{\hat{c}_i = \max(\sigma_i)\}_{i=1}^N$: maximum class probability returned by original NN classifier on training data

$\mathbf{x}_*$: data to be predicted

σ_*: softmax outputs of original NN classifier on $\mathbf{x}_*$

$\hat{c}_*$: maximum class probability returned by original NN classifier on $\mathbf{x}_*$

Ensure:

$\hat{c}'_* \sim \mathcal{N}(\hat{c}_* + \bar{\hat{r}}_*, \mathrm{var}(\hat{r}_*))$: $\hat{c}_* + \bar{\hat{r}}_*$ can be used as detection score for error detection, and $\mathrm{var}(\hat{r}_*)$ represents the uncertainty of returned detection score

Training Phase:

1: obtain target detection score $\mathbf{c} = \{c_i = \delta_{y_i,\hat{y}_i}\}_{i=1}^N$, where $\delta_{y_i,\hat{y}_i}$ is the Kronecker delta ($\delta_{y_i,\hat{y}_i} = 1$ if $y_i = \hat{y}_i$, otherwise $\delta_{y_i,\hat{y}_i} = 0$)

2: calculate residuals $\mathbf{r} = \{r_i = c_i - \hat{c}_i\}_{i=1}^N$

3: **for** each optimizer step **do**

4: calculate covariance matrix $\mathbf{K}_c((\mathcal{X}, \boldsymbol{\sigma}), (\mathcal{X}, \boldsymbol{\sigma}))$, where each entry is given by $k_c((\mathbf{x}_i, \sigma_i), (\mathbf{x}_j, \sigma_j)) = k_{\mathrm{in}}(\mathbf{x}_i, \mathbf{x}_j) + k_{\mathrm{out}}(\sigma_i, \sigma_j)$, for $i, j = 1, 2, \ldots, N$

5: optimize GP hyperparameters by maximizing log marginal likelihood $\log p(\mathbf{r}|\mathcal{X}, \boldsymbol{\sigma}) = -\frac{1}{2}\mathbf{r}^\top(\mathbf{K}_c((\mathcal{X}, \boldsymbol{\sigma}), (\mathcal{X}, \boldsymbol{\sigma})) + \sigma_n^2\mathbf{I})^{-1}\mathbf{r} - \frac{1}{2}\log|\mathbf{K}_c((\mathcal{X}, \boldsymbol{\sigma}), (\mathcal{X}, \boldsymbol{\sigma})) + \sigma_n^2\mathbf{I}| - \frac{n}{2}\log 2\pi$

Deployment Phase:

6: calculate residual mean $\bar{\hat{r}}_* = \mathbf{k}_*^\top(\mathbf{K}_c((\mathcal{X}, \boldsymbol{\sigma}), (\mathcal{X}, \boldsymbol{\sigma})) + \sigma_n^2\mathbf{I})^{-1}\mathbf{r}$ and residual variance $\mathrm{var}(\hat{r}_*) = k_c((\mathbf{x}_*, \sigma_*), (\mathbf{x}_*, \sigma_*)) - \mathbf{k}_*^\top(\mathbf{K}_c((\mathcal{X}, \boldsymbol{\sigma}), (\mathcal{X}, \boldsymbol{\sigma})) + \sigma_n^2\mathbf{I})^{-1}\mathbf{k}_*$, where $\mathbf{k}_*$ denotes the vector of kernel-based covariances (i.e., $k_c(\mathbf{x}_*, \mathbf{x}_i)$) between $\mathbf{x}_*$ and all training data

7: return distribution of error detection score $\hat{c}'_* \sim \mathcal{N}(\hat{c}_* + \bar{\hat{r}}_*, \mathrm{var}(\hat{r}_*))$

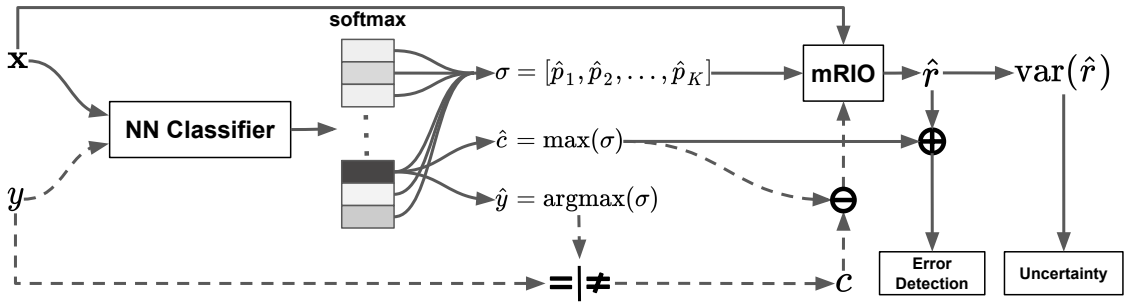

Figure 1: **The RED training and deployment processes.** The solid pathways are active in both training and deployment phases, while the dashed pathways are active only in the training phase. During the training phase, a target detection score c is assigned to each training sample according to whether it is correctly predicted by the original NN classifier or not. An mRIO model is then trained to predict the residual between the target detection score c and the original maximum class probability $\hat{c}$. The I/O kernel in mRIO utilizes both the raw feature $\mathbf{x}$ and softmax outputs σ to predict the residuals. In the deployment phase, given a new data point, the trained mRIO model provides a Gaussian distribution of estimated residual $\hat{r}$ defined by the mean $\bar{\hat{r}}$ and variance $\text{var}(\hat{r})$. Addition of $\hat{r}$ and $\hat{c}$ forms a score for error detection, and $\text{var}(\hat{r})$ indicates the corresponding uncertainty.

in the training dataset for RED. After the target detection scores are assigned, a regression problem is formulated for the mRIO model: Given the original raw features $\{\mathbf{x}_i\}_{i=1}^N$ and the corresponding softmax outputs of the original NN classifier $\{\sigma_i = [\hat{p}_{i,1}, \hat{p}_{i,2}, \ldots, \hat{p}_{i,K}]\}_{i=1}^N$, predict the residuals $\mathbf{r} = \{r_i = c_i - \hat{c}_i\}_{i=1}^n$ between target detection scores $\mathbf{c} = \{c_i\}_{i=1}^N$ and the original maximum class probabilities $\hat{\mathbf{c}} = \{\hat{c}_i = \max(\sigma_i)\}_{i=1}^N$.

The mRIO model relies on an I/O kernel consisting of two components: the input kernel $k_{\text{in}}(\mathbf{x}_i, \mathbf{x}_j)$, which measures covariances in the raw feature space, and the modified multi-output kernel $k_{\text{out}}(\sigma_i, \sigma_j)$, which calculates covariances in the softmax output space. The hyperparameters of the I/O kernel are optimized to maximize the log marginal likelihood $\log p(\mathbf{r}|\mathcal{X}, \boldsymbol{\sigma})$. In the deployment phase, given a new data point $\mathbf{x}_*$, the trained mRIO model provides a Gaussian distribution for the estimated residual $\hat{r}_* \sim \mathcal{N}(\bar{\hat{r}}_*, \text{var}(\hat{r}_*))$. By adding the estimated residual back to the original maximum class probability $\hat{c}_*$, a distribution of detection score is obtained as $\hat{c}'_* \sim \mathcal{N}(\hat{c}_* + \bar{\hat{r}}_*, \text{var}(\hat{r}_*))$. The mean $\hat{c}_* + \bar{\hat{r}}_*$ can be directly used as a quantitative metric for error detection, and the variance $\text{var}(\hat{r}_*)$ represents the corresponding uncertainty of the detection score.

4 Empirical Evaluation

In this section, the error detection performance of RED is evaluated comprehensively on 125 UCI datasets, comparing it to other related methods. Its generality is then evaluated by applying it to two other base models, and its scale-up properties measured in two larger deep learning architectures solving two vision tasks. Finally, RED's potential to improve robustness more broadly is demonstrated in an empirical study involving OOD and adversarial samples.

4.1 Comparisons with Related Approaches

As a comprehensive evaluation of RED, an empirical comparison with seven related approaches on 125 UCI datasets (Dua and Graff 2017) was performed. These approaches include maximum class probability (MCP) baseline (Hendrycks and Gimpel 2017), three state-of-the-art approaches, namely Trust Score (T-Score; Jiang et al. 2018), ConfidNet (C-net; Corbière et al. 2019), and Introspection-Net (I-net; Aigrain and Detyniecki 2019), as well as three earlier approaches, i.e. entropy of the original softmax outputs (Steinhardt and Liang 2016), DNGO (Snoek et al. 2015), and the original SVGP (Hensman, Fusi, and Lawrence 2013; Hensman, Matthews, and Ghahramani 2015). The 125 UCI datasets include 121 datasets used by Klambauer et al. (2017) and four more recent ones. Preliminary tests indicate that RED is not sensitive to the choice of kernel functions, so the standard RBF (Radial Basis Function) kernel is used in the current RED implementation. Full details about the datasets and parametric setup of all tested algorithms, and source codes for reproducing the experimental results are provided in supplementary material.

Following the experimental setup of Hendrycks and Gimpel (2017); Corbière et al. (2019); Aigrain and Detyniecki (2019), the task for each algorithm is to provide a detection score for each testing point. An error detector can then use a predefined fixed threshold on this score to decide which points are probably misclassified by the original NN classifier. For RED, the mean $\hat{c}_* + \bar{\hat{r}}_*$ was used as the reported detection score. Five threshold-independent performance metrics were used to compare the methods: AUPR-Error, which computes the area under the Precision-Recall (AUPR) Curve when treating incorrect predictions as positive class during the detection; AUPR-Success, which is similar to AUPR-Error but uses correct predictions as positive class; AUROC, which computes the area under receiver operating characteristic (ROC) curve for the error detection task; AP-Error, which computes the average precision (AP) under different thresholds treating incorrect predictions as the positive class; and AP-Success, which is similar to AP-Error but uses correct predictions as the positive class. AUPR and AUROC are commonly used in prior work (Hendrycks and Gimpel 2017; Corbière et al. 2019; Aigrain and Detyniecki 2019), but as discussed by Davis and Goadrich (2006) and Flach and Kull

Method	AP-Error mean±std	AUPR-Error mean±std	AP-Success mean±std	AUPR-Success mean±std	AUROC mean±std
RED	**1.39±0.61***	**1.49±0.78***	**1.74±0.97***	**1.80±1.03***	**1.65±0.82***
MCP	2.93± 0.89	3.06±0.92	2.77±1.07	2.75±1.11	2.80±1.08
T-Score	3.92±2.45	3.86±2.50	3.64±2.25	3.61±2.25	3.76±2.31
C-Net	6.13±1.37	6.33±1.38	6.07±1.51	6.07±1.41	5.97±1.45
I-Net	5.34±1.65	5.38±1.65	5.83±1.46	5.89±1.51	5.71±1.50
Entropy	3.47±1.08	3.59±1.19	3.19±1.26	3.23±1.32	3.26±1.28
DNGO	6.19±1.51	5.46±1.82	6.84±1.33	6.80±1.44	6.57±1.47
SVGP	6.59±1.60	6.80±1.49	5.89±1.54	5.83±1.49	6.24±1.61

The symbol * indicates that the differences between the marked entry and all other counterparts are statistically significant at the 5% significance level for both paired t-test and Wilcoxon test. The best entries that are significantly better than all the others under both tests are in boldface.

Table 1: Mean rank on UCI datasets

RED vs.	AP-Error + / = / -	AUPR-Error + / = / -	AP-Success + / = / -	AUPR-Success + / = / -	AUROC + / = / -
MCP	87 / 35 / 0	90 / 32 / 0	58 / 63 / 1	56 / 65 / 1	61 / 60 / 1
T-Score	53 / 44 / 16	49 / 47 / 17	50 / 47 / 16	48 / 49 / 16	59 / 37 / 17
C-Net	100 / 22 / 0	100 / 22 / 0	106 / 16 / 0	106 / 16 / 0	109 / 13 / 0
I-Net	93 / 29 / 0	90 / 32 / 0	98 / 24 / 0	98 / 24 / 0	101 / 21 / 0
Entropy	74 / 47 / 1	75 / 46 / 1	53 / 68 / 1	53 / 68 / 1	52 / 69 / 1
DNGO	92 / 17 / 0	73 / 31 / 5	99 / 10 / 0	97 / 12 / 0	98 / 11 / 0
SVGP	98 / 23 / 1	98 / 23 / 1	97 / 25 / 0	97 / 25 / 0	102 / 19 / 1
BNN-M	102 / 20 / 0	104 / 18 / 0	95 / 26 / 1	88 / 33 / 1	95 / 26 / 1
BNN-E	67 / 53 / 2	68 / 52 / 2	48 / 66 / 8	48 / 66 / 8	53 / 64 / 5
MCD-M	87 / 35 / 0	88 / 34 / 0	70 / 52 / 0	67 / 55 / 0	71 / 51 / 0
MCD-E	54 / 68 / 0	55 / 67 / 0	38 / 77 / 7	38 / 76 / 8	42 / 74 / 6
BLR-res	77 / 43 / 0	76 / 44 / 0	92 / 28 / 0	90 / 30 / 0	88 / 32 / 0

The columns labeled + show the number of datasets on which RED performs significantly better at the 5% significance level in a paired t-test, Wilcoxon test, or both; those labeled - represent the contrary case; those labeled = represent no statistical significance.

Table 2: A pairwise comparison between RED and other methods on UCI datasets

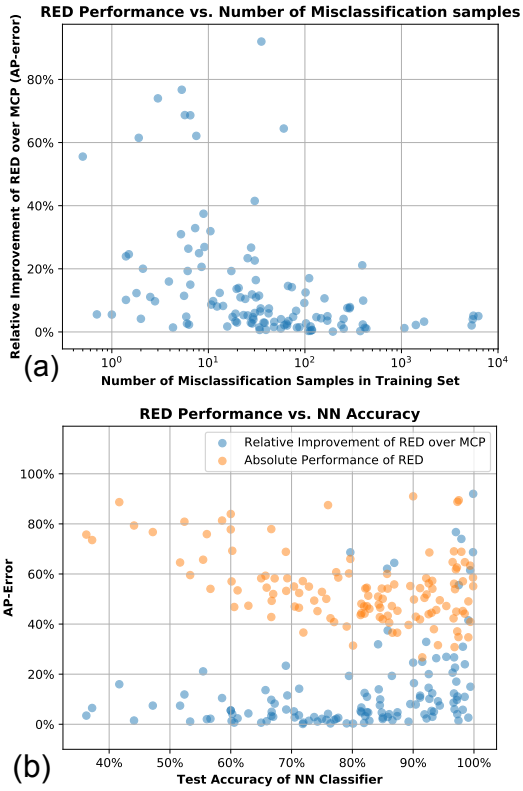

Figure 2: **Impact of Number of Misclassification Samples (a) and Impact of Classifier Accuracy (b)**. Each dot represents one dataset, averaged over 10 independent runs. In some runs of a dataset, there are no misclassified samples (100% accuracy), so the average number of misclassified samples may be less than 1. The improvement of RED over MCP baseline is more significant for situations where number of misclassification samples is small (few-shot learning) or accuracies of base NN classifiers are high (imbalanced training set), indicating that RED provides the largest advantage in extreme cases.

(2015), AUPR may provide overly-optimistic measurement of performance. Moreover, AUROC is sometimes less informative (Manning and Schütze 1999) and not ideal when the positive class and negative class have greatly differing base rates (Hendrycks and Gimpel 2017) (this happens when the base classifier has high prediction accuracy so there are only few misclassified examples). To compensate for these issues, AP-Error and AP-Success are included as additional metrics. Since the target of all tested approaches is to detect misclassification errors, the following discussion will focus more on AP-Error and AUPR-Error.

Ten independent runs were conducted for each dataset. During each run, the dataset was randomly split into training dataset and testing dataset, and a standard NN classifier trained and evaluated on them. The same dataset split and trained NN classifier were used to evaluate all methods. The testbed covers a diversified collection of dataset sizes and classifier accuracies, thereby enabling a comprehensive evaluation of the tested approaches. Full details of the experimental setup are provided in Appendix A.1.

Table 1 shows the ranks of each algorithm averaged over all 125 UCI datasets (see Appendix A.4 for detailed results). The rank of each algorithm on each dataset is based on the average performance over 10 independent runs. RED performs best on all metrics; the performance differences between RED and all other methods are statistically significant under paired t-test and Wilcoxon test. Trust Score has the highest standard deviation, suggesting that its performance varies significantly across different datasets.

Table 2 shows how often RED performs statistically significantly better, how often the performance is not significantly different, and how often it performs significantly worse than the other methods. RED is most often significantly better, and very rarely worse. In a handful of datasets Trust Score is better, but most often it is not.

To illustrate the impact of misclassification samples in training set, Figure 2 shows the performance of RED under different numbers of misclassification samples and base NN classifier accuracies. RED consistently improves over the MCP baseline, and this improvement is more significant for extreme cases, i.e. when only a limited number of misclassification samples is provided (i.e. in case of few-

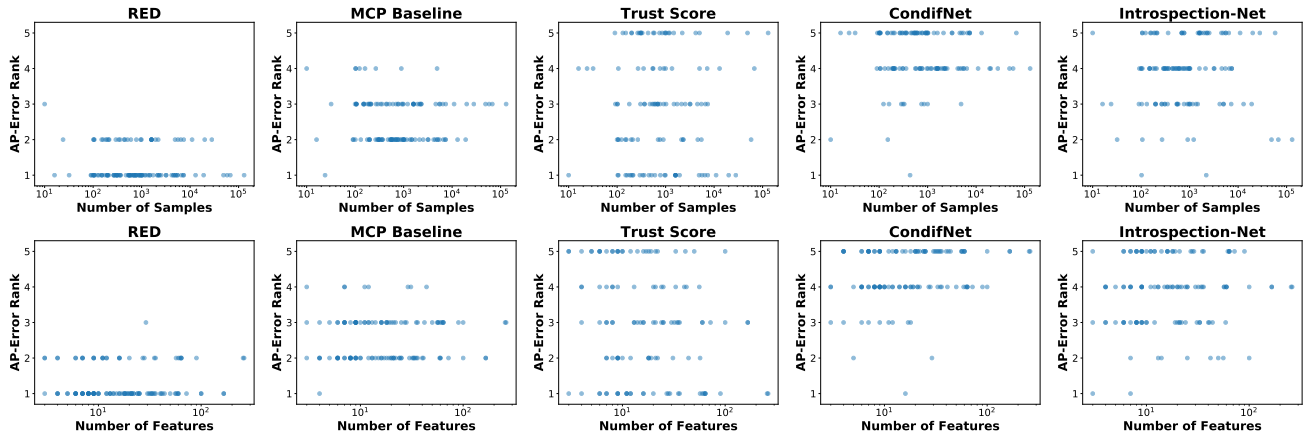

Figure 3: **Performance ranks across dataset sizes and feature dimensionalities on UCI datasets.** Each plot represents the distribution of relative ranks for one method (each column) as a function of the dataset size (top row) and the feature dimensionality (bottom row). Each dot in each plot represents the relative rank in one dataset. RED performs consistently well over datasets of different sizes and feature dimensionalities. Trust Score performs inconsistently, and ConfidNet performs poorly on larger datasets.

shot learning) or the base NN classifier has high accuracy (i.e. in case of an imbalanced training set), demonstrating the robustness of RED. RED's AP-Error is generally higher with low-accuracy classifiers. This is because higher ratio of misclassification samples naturally leads to higher detection precision, and an accuracy around 50% leads to a more balanced training set for the error detector.

To further study the robustness of RED compared to the baseline and the three state-of-the-art approaches, Figure 3 shows the distribution of the relative rank of RED, MCP baseline, Trust Score, ConfidNet and Introspection-Net as a function of the number of samples and the number of features in the dataset. These plots are based on the AP-Error metric; other metrics provide similar results. RED performs consistently well over different dataset sizes and feature dimensionalities. Trust Score performs best in several datasets, but occasionally also worst in both small and large datasets, making it a rather unreliable choice. ConfidNet generally exhibits worse performance on datasets with large dataset sizes and high feature dimensionalities, i.e. it does not scale well to larger problems.

To evaluate whether GP is indeed an appropriat model for the RED framework, it was replaced by a Bayesian linear regressor (BLR; Snoek et al. 2015), with all other components unchanged. This BLR-residual (BLR-res) variant was then compared with the original RED in all 125 UCI datasets (see Appendix A.1 for the setup). Results in Table 2 (last row) show that RED dominates BLR-res, indicating that GP is a good choice for error detection tasks.

4.2 Generality wrt. Base Models

To evaluate generality of RED, it was applied to two other base models: an NN classifier using Monte Carlo-dropout (MCD) technique (Gal and Ghahramani 2016) and a Bayesian Neural Network (BNN) classifier (Wen et al. 2018). They were each trained as base classifiers, and RED was then applied to each of them (implementation details are provided in Appendix A.1). Experiments analogous to those in Section 4.1 were performed on 125 UCI datasets in both cases. Table 2 (rows starting with "BNN" or "MCD") summarizes the pairwise comparisons between RED and the internal detection scores returned by the base models (see Appendix A.4 for detailed results). "-M" and "-E" represent the maximum class probability and entropy of softmax outputs, respectively, after averaging over 100 test-time samplings. RED significantly improves MCD and BNN classifier in most datasets, demonstrating that it is a general technique that can be applied to a variety of models.

4.3 Scaling up to Larger Architectures

To confirm that the RED approach scales up to large deep learning architectures, a VGG16 model (Simonyan and Zisserman 2015) and a Wide Residual Network (WRN) model (Zagoruyko and Komodakis 2016) was trained on the CIFAR-10 dataset (Krizhevsky 2009), and a VGG19 model (Simonyan and Zisserman 2015) was trained on the CIFAR-100 dataset (Krizhevsky 2009), using state-of-the-art training pipelines (Bingham and Miikkulainen 2020) (see Appendix A.2 for details). In order to remove the influence of feature extraction in image preprocessing and to make the comparison fair, all approaches used the same logit outputs of the trained VGG/WRN model as their input features. 10 independent runs are performed. During each run, a VGG/WRN model is trained, and all the methods are evaluated based on this VGG/WRN model.

Table 3 shows the results on the two main error detection performance metrics (note that the table lists absolute values instead of rankings along each metric). Trust Score performs much better than in previous literatures (Corbière et al. 2019). This difference may be due to the fact that logit outputs are used as input features here, whereas Corbière et al. (2019) utilized a higher dimensional feature space for

Task	Metric	RED mean±std	MCP Baseline mean±std	Trust Score mean±std	ConfidNet mean±std	Introspection-Net mean±std	Entropy mean±std	DNGO mean±std	SVGP mean±std
VGG16 on	AP-Error(%)	**49.88±1.99***	47.09±2.19	48.76±2.28	45.80±2.85	42.11±1.98	47.91±2.17	33.91±2.94	40.71±2.33
CIFAR-10	AUPR-Error(%)	**49.79±2.00***	46.99±2.21	48.68±2.29	45.70±2.86	42.01±1.98	47.81±2.19	34.43±2.92	40.60±2.34
WRN-10-4 on	AP-Error(%)	**52.51±2.81***	48.76±1.47	51.15±3.29	48.34±3.05	40.52±3.64	48.12±1.51	28.96±3.54	6.43±0.32
CIFAR-10	AUPR-Error(%)	**52.45±2.82***	48.70±1.48	51.09±3.30	48.27±3.05	40.44±3.65	48.06±1.51	29.40±3.60	6.42±0.32
VGG19 on	AP-Error(%)	**73.40±1.05***	71.78±1.24	72.63±1.20	72.12±1.33	69.73±1.13	72.95±1.19	55.44±1.64	29.24±0.90
CIFAR-100	AUPR-Error(%)	**73.39±1.05***	71.77±1.25	72.62±1.20	72.10±1.13	69.72±1.13	72.94±1.20	55.44±1.64	33.64±1.24

The symbol * indicates that the differences between the marked entry and all other counterparts are statistically significant at the 5% significance level for both paired t-test and Wilcoxon test. The best entries that are significantly better than all the others under both tests are in boldface.

Table 3: Error detection performance with deep NN models on CIFAR-10 and CIFAR-100

RED-variance vs.	AP-OOD + / = / -	AUPR-OOD + / = / -	AP-Adversarial + / = / -	AUPR-Adversarial + / = / -
MCP baseline	101 / 15 / 9	101 / 13 / 11	122 / 3 / 0	124 / 1 / 0
RED-mean	100 / 14 / 11	100 / 13 / 12	122 / 3 / 0	122 / 3 / 0

The columns labeled + show the number of datasets on which RED performs significantly better at the 5% significance level in a paired t-test, Wilcoxon test, or both; those labeled - represent the contrary case; those labeled = represent no statistical significance.

Table 4: A pairwise comparison between RED-variance and other methods on detection of OOD and adversarial samples

AP-OOD(%)	AUPR-OOD(%)
RED-variance / MCP baseline	RED-variance / MCP baseline
86.282±2.212* / 82.964±1.850	**86.276±2.213*** / 82.958±1.851

The symbol * indicates that the difference between the RED-variance and the MCP baseline is statistically significant at the 5% significance level for both paired t-test and Wilcoxon test. The significantly better entries under both tests are in boldface.

Table 5: Performance on detecting OOD samples (SVHN data) from CIFAR-10 data (mean±std over 10 runs)

Trust Score. RED significantly outperforms all the counterparts in both metrics. This result demonstrates the advantages of RED in scaling up to larger architectures.

4.4 Distinguishing In-sample Errors from OOD and Adversarial Samples

In all experiments so far, the mean of detection score $\hat{c}_* + \bar{\hat{r}}_*$ was used as RED's detection metric. Although good performance was observed in error detection by only using the mean, the variance of detection score $\mathrm{var}(\hat{r}_*)$ may be helpful if the scenario is more complex, e.g., the dataset includes some OOD data, or even adversarial data.

RED was evaluated in such a scenario by manually adding OOD and adversarial data into the test set of all 125 UCI datasets. The synthetic OOD and adversarial samples were created to be highly deceptive, aiming to evaluate the performance of RED under difficult circumstances. The OOD data were sampled from a Gaussian distribution with mean 0 and variance 1. All samples from original dataset were normalized to have mean 0 and variance 1 for each feature dimension so that the OOD data and in-distribution data had similar scales. The adversarial data simulate situations where negligible modifications to training samples cause the original NN classifier to predict incorrectly with highest confidence (Goodfellow, Shlens, and Szegedy 2014). Full setup details are provided in Appendix A.3.

Figure 4 shows the distribution of mean and variance of detection scores for testing samples, including correctly and incorrectly labeled actual samples, as well as the synthetic OOD and adversarial samples. The mean is a good separator for correctly classified and incorrectly classified samples, which tend to cluster on the top and bottom half of the image, respectively. On the other hand, variance is a good indicator of OOD and adversarial samples. RED's detection scores of in-distribution samples have low variance because they covary with the training samples. The variance thus represents RED's confidence in its detection score. Samples with large variance indicate that RED is uncertain about its detection score, which can be used as a basis for detecting OOD and adversarial samples.

In order to quantify the potential of RED in detecting OOD and adversarial samples, the variance of detection scores $\mathrm{var}(\hat{r}_*)$ (RED-variance) was used as the detection metric, and detection performance compared with MCP baseline and stardard RED (RED-mean) in all 125 UCI datasets (10 independent runs each). The performance in detecting OOD samples was measured by AP-OOD and AUPR-OOD, which treat OOD samples as the positive class. Similarly, AP-Adversarial and AUPR-Adversarial were used as measures in detecting adversarial samples. The RED training pipeline was exactly the same as in Section 4.1. A summary of the experimental results is shown in Table 4 (see Appendix A.3 for setup details, and Appendix A.5 for detailed results). Intriguingly, RED-variance performs well in both OOD and adversarial sample detection even though it was not trained on any OOD/adversarial samples. In contrast, the original MCP baseline performs significantly worse in both scenarios. The original NN classifier always returns highest class probabilities on deceptive adversarial samples; as a result, MCP makes a purely random guess, resulting in a consistent AP-Adversarial/AUPR-Adversarial of 50%/25%. In addition, the comparison between RED-variance and RED-mean verifies that the variance $\mathrm{var}(\hat{r}_*)$ is a more discriminative metric than mean $\hat{c}_* + \bar{\hat{r}}_*$ in detecting OOD and adversarial samples.

The scalability of RED-variance was evaluated in a more complex OOD detection task: Images from the SVHN dataset (Netzer et al. 2011) were treated as OOD samples for VGG16 classifiers trained on CIFAR-10 dataset. The same RED and VGG16 models as in Section 4.3 were used without retraining. Experimental results in Table 5 show that RED-variance consistently outperforms the MCP baseline.

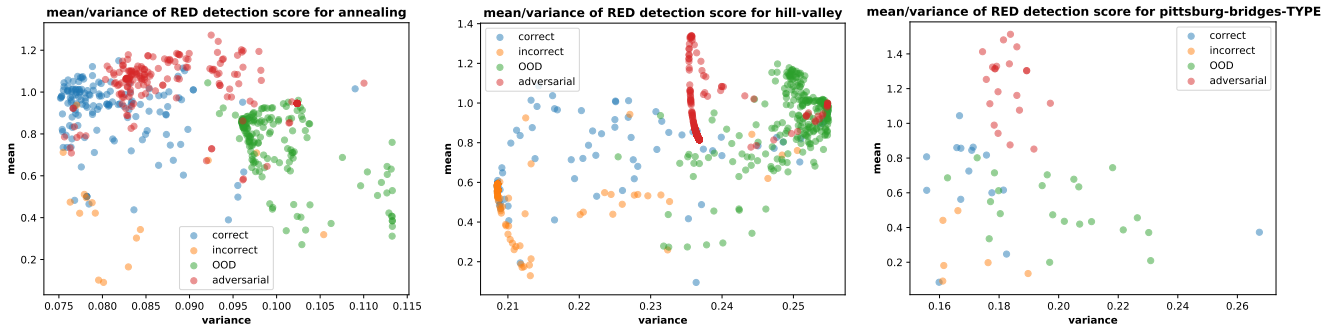

Figure 4: **Distinguishing in-sample errors from OOD and adversarial samples based on mean and variance of detection scores.** Each dot represents one sample in the testing set in the corresponding UCI task. The horizontal axis denotes the variance of RED-returned detection score, and the vertical axis denotes the mean. If an in-distribution sample is correctly classified by original NN classifier, it is marked as "correct", otherwise it is marked "incorrect". Mean is a good separator of correct and incorrect classifications. High variance, on the other hand, indicates that RED is uncertain about its detection score, which can be used to identify OOD and adversarial samples.

Thus, the empirical study in this subsection shows that RED provides a promising foundation not just for detecting misclassifications, but for distinguishing them from other error types as well. This is a new dimension in reliability and interpretability in machine learning systems. RED can therefore serve as a step to make deployments of such systems safer in the future.

5 Discussion and Future Work

One interesting observation from the experiments is that RED almost never performs worse than the MCP baseline. This result suggests that there is almost no risk in applying RED on top of an existing NN classifier. Since RED is based on a GP model, the estimated residual $\bar{\hat{r}}_*$ is close to zero if the predicted sample is far from the distribution of the original training samples, resulting in a negligible change to the original MCP. In other words, RED does not make random changes to original MCP if it is very uncertain about the predicted sample, and this uncertainty is explicitly represented in the variance of the estimated detection score. Moreover, the distance-awareness ability of GP also substantially improves the quality of uncertainty estimation (Liu et al. 2020). These properties make RED a particularly reliable technique for error detection.

Another inspiring observation is that the variance is also helpful in detecting OOD and adversarial samples. This result follows from the design of the GP model. Since mRIO in RED has an input kernel and a multi-output kernel, lower estimated variance requires that the predicted sample is close to training samples in both the input feature space and the classifier output space. This requirement is difficult for OOD and adversarial attacks to achieve, providing a basis for detecting them.

In a real-world deployment, it is necessary to define a threshold for triggering error warning based on RED's detection scores. A practical way is to use a validation dataset to determine how the precision/recall tradeoff changes over different thresholds. The user can then select a threshold based on their preference.

The main limitation of RED is that it is not applicable to classifiers that have 100% prediction accuracy in both training and validation datasets, as might happen in some cases of overfitting to small datasets. In this case, there are no misclassification samples for RED to learn. In practice, this situation is easy to identify by directly looking at the training and validation performance of the original NN classifier. One potential solution is to reserve a portion of data that include at least one misclassification sample for training RED.

The most compelling direction of future work is to extend the capability of RED in distinguishing errors further. Instead of using a single dimensional score for error detection, it is possible to use mean and variance simultaneously, leading to a two dimensional detection space. Further separation between OOD and adversarial samples may be possible by adding one more dimension, such as the ratio between input kernel output and multi-output kernel output. Also, instead of using a hard target detection score (i.e. either 0 or 1), it may be possible to define a soft target score that is more informative. Further, RED may be used on top of other existing detection metrics, such as the Trust Score, which may lead to a further improvement in detection performance.

6 Conclusion

This paper introduced a new framework, RED, for error detection in neural network classifiers that can produce a more reliable detection score than previous methods. RED is able to not only provide a detection score, but also report the uncertainty of that score. Experimental results show that RED's scores consistently outperform state-of-the-art methods in separating the misclassified samples from correctly classified samples. Further empirical studies also demonstrate that the approach is applicable to various types of base classifiers, scales up to large deep learning architectures, and can form a basis for detecting OOD and adversarial samples as well. It is therefore a promising foundation for improving robustness of neural network classifiers.

Acknowledgements

We thank Garrett Bingham for sharing source code and providing helpful suggestions in training WRN.

References

Aigrain, J.; and Detyniecki, M. 2019. Detecting Adversarial Examples and Other Misclassifications in Neural Networks by Introspection. *ArXiv*, abs/1905.09186.

Alghoul, A.; Ajrami, S. A.; Jarousha, G. A.; Harb, G.; and Abu-Naser, S. S. 2018. Email Classification Using Artificial Neural Network. *International Journal of Academic Engineering Research (IJAER)*, 2(11): 8–14.

Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P.; Schulman, J.; and Man, D. 2016. Concrete Problems in AI Safety.

Anjos, O.; Iglesias, C.; Peres, F.; Martínez, J.; García, A.; and Taboada, J. 2015. Neural networks applied to discriminate botanical origin of honeys. *Food Chemistry*, 175: 128 – 136.

Bartlett, P. L.; and Wegkamp, M. H. 2008. Classification with a Reject Option Using a Hinge Loss. *J. Mach. Learn. Res.*, 9: 1823–1840.

Bingham, G.; and Miikkulainen, R. 2020. Discovering Parametric Activation Functions. *ArXiv*, abs/2006.03179.

Chow, C. 2006. On Optimum Recognition Error and Reject Tradeoff. *IEEE Trans. Inf. Theor.*, 16(1): 41–46.

Corbière, C.; THOME, N.; Bar-Hen, A.; Cord, M.; and Pérez, P. 2019. Addressing Failure Prediction by Learning Model Confidence. In *Advances in Neural Information Processing Systems 32*, 2902–2913. Curran Associates, Inc.

Cortes, C.; DeSalvo, G.; and Mohri, M. 2016. Learning with Rejection. In *Algorithmic Learning Theory*, 67–82. Cham: Springer International Publishing.

Davis, J.; and Goadrich, M. 2006. The Relationship Between Precision-Recall and ROC Curves. volume 06.

Devries, T.; and Taylor, G. W. 2018. Learning Confidence for Out-of-Distribution Detection in Neural Networks. *ArXiv*, abs/1802.04865.

Dixon, M.; Klabjan, D.; and Bang, J. 2017. Classification-Based Financial Markets Prediction Using Deep Neural Networks. *Algorithmic Finance*.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository.

Dubuisson, B.; and Masson, M. 1993. A statistical decision rule with incomplete knowledge about classes. *Pattern Recognition*, 26(1): 155 – 165.

Flach, P.; and Kull, M. 2015. Precision-Recall-Gain Curves: PR Analysis Done Right. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28*, 838–846. Curran Associates, Inc.

Gal, Y.; and Ghahramani, Z. 2016. Dropout As a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, 1050–1059. JMLR.org.

Geifman, Y.; and El-Yaniv, R. 2017. Selective Classification for Deep Neural Networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 4885–4894. Red Hook, NY, USA.

Goodfellow, I.; Shlens, J.; and Szegedy, C. 2014. Explaining and Harnessing Adversarial Examples. *ArXiv*, abs/1412.6572.

Guo, C.; Pleiss, G.; Sun, Y.; and Weinberger, K. Q. 2017. On Calibration of Modern Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, 1321–1330. JMLR.org.

Gupta, C. N.; Palaniappan, R.; Swaminathan, S.; and Krishnan, S. M. 2007. Neural network classification of homomorphic segmented heart sounds. *Applied Soft Computing*, 7(1): 286 – 297.

Haldimann, D.; Blum, H.; Siegwart, R.; and Cadena, C. 2019. This is not what I imagined: Error Detection for Semantic Segmentation through Visual Dissimilarity. *ArXiv*, abs/1909.00676.

Hecker, S.; Dai, D.; and Van Gool, L. 2018. Failure Prediction for Autonomous Driving.

Hendrycks, D.; and Gimpel, K. 2017. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. *Proceedings of International Conference on Learning Representations*.

Hensman, J.; Fusi, N.; and Lawrence, N. D. 2013. Gaussian Processes for Big Data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, 282–290.

Hensman, J.; Matthews, A.; and Ghahramani, Z. 2015. Scalable Variational Gaussian Process Classification. In Lebanon, G.; and Vishwanathan, S. V. N., eds., *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, 351–360.

Janai, J.; Güney, F.; Behl, A.; and Geiger, A. 2017. Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-the-Art. *ArXiv*, abs/1704.05519.

Jiang, H.; Kim, B.; Guan, M. Y.; and Gupta, M. 2018. To Trust or Not to Trust a Classifier. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, 5546–5557.

Klambauer, G.; Unterthiner, T.; Mayr, A.; and Hochreiter, S. 2017. Self-Normalizing Neural Networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 972–981.

Koppel, M.; and Engelson, S. P. 1996. Integrating Multiple Classifiers By Finding Their Areas of Expertise. In *In: AAAI-96 Workshop On Integrating Multiple Learning Models*, 53–58.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images. *University of Toronto*.

Kull, M.; Perello Nieto, M.; Kängsepp, M.; Silva Filho, T.; Song, H.; and Flach, P. 2019. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with Dirichlet calibration. In *Advances in Neural Information Processing Systems 32*, 12316–12326.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *Nature*, 521: 436 EP –.

Lee, K.; Lee, H.; Lee, K.; and Shin, J. 2018a. Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples. In *International Conference on Learning Representations*.

Lee, K.; Lee, K.; Lee, H.; and Shin, J. 2018b. A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31*, 7167–7177. Curran Associates, Inc.

Liang, S.; Li, Y.; and Srikant, R. 2018. Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks. In *International Conference on Learning Representations*.

Liu, J.; Lin, Z.; Padhy, S.; Tran, D.; Bedrax Weiss, T.; and Lakshminarayanan, B. 2020. Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 7498–7512.

Mandelbaum, A.; and Weinshall, D. 2017. Distance-based Confidence Score for Neural Network Classifiers. *ArXiv*, abs/1709.09844.

Manning, C. D.; and Schütze, H. 1999. *Foundations of Statistical Natural Language Processing*.

Monteith, K.; and Martinez, T. R. 2010. Using multiple measures to predict confidence in instance classification. *The 2010 International Joint Conference on Neural Networks (IJCNN)*, 1–8.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading Digits in Natural Images with Unsupervised Feature Learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

Nguyen, A.; Yosinski, J.; and Clune, J. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. 427–436.

Niculescu-Mizil, A.; and Caruana, R. 2005. Predicting Good Probabilities with Supervised Learning. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, 625–632. New York, NY, USA.

Ortega, J. 1995. Exploiting Multiple Existing Models and Learning Algorithms. In *In AAAI 96 - Workshop in Induction of Multiple Learning Models*, 239–266.

Platt, J. C. 1999. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In *Advances in Large Margin Classifiers*, 61–74. MIT Press.

Provost, F. J.; Fawcett, T.; and Kohavi, R. 1998. The Case against Accuracy Estimation for Comparing Induction Algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, 445–453. San Francisco, CA, USA.

Qiu, X.; Meyerson, E.; and Miikkulainen, R. 2020. Quantifying Point-Prediction Uncertainty in Neural Networks via Residual Estimation with an I/O Kernel. In *International Conference on Learning Representations*.

Santos-Pereira, C. M.; and Pires, A. M. 2005. On Optimal Reject Rules and ROC Curves. *Pattern Recogn. Lett.*, 26(7): 943–952.

Seewald, A. K.; and Fürnkranz, J. 2001. An Evaluation of Grading Classifiers. In *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis*, IDA '01, 115–124. Berlin, Heidelberg.

Selişteanu, D.; Maksimenko, V. A.; Kurkin, S. A.; Pitsik, E. N.; Musatov, V. Y.; Runnova, A. E.; Efremova, T. Y.; Hramov, A. E.; and Pisarchik, A. N. 2018. Artificial Neural Network Classification of Motor-Related EEG: An Increase in Classification Accuracy by Reducing Signal Complexity. *Complexity*, 2018: 938–947.

Shahid, N.; Rappon, T.; and Berta, W. 2019. Applications of artificial neural networks in health care organizational decision-making: A scoping review. *PLOS ONE*, 14(2): 1–22.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

Snoek, J.; Rippel, O.; Swersky, K.; Kiros, R.; Satish, N.; Sundaram, N.; Patwary, M. M. A.; Prabhat, P.; and Adams, R. P. 2015. Scalable Bayesian Optimization Using Deep Neural Networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, 2171–2180. JMLR.org.

Steinhardt, J.; and Liang, P. 2016. Unsupervised Risk Estimation Using Only Conditional Independence Structure. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, 3664–3672. Red Hook, NY, USA.

Wang, J.; Dong, G.; Sun, J.; Wang, X.; and Zhang, P. 2019. Adversarial Sample Detection for Deep Neural Network through Model Mutation Testing. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, 1245–1256.

Wen, Y.; Vicol, P.; Ba, J.; Tran, D.; and Grosse, R. 2018. Flipout: Efficient Pseudo-Independent Weight Perturbations on Mini-Batches. In *International Conference on Learning Representations*.

Williams, G.; and Renals, S. 1997. Confidence Measures for Hybrid HMM/ANN Speech Recognition. In *Proceedings of EuroSpeech*, 1955–1958.

Xing, C.; Arik, S.; Zhang, Z.; and Pfister, T. 2020. Distance-Based Learning from Errors for Confidence Calibration. In *International Conference on Learning Representations*.

Yuan, M.; and Wegkamp, M. 2010. Classification Methods with Reject Option Based on Convex Risk Minimization. *Journal of Machine Learning Research*, 11(5): 111–130.

Zadrozny, B.; and Elkan, C. 2001. Obtaining Calibrated Probability Estimates from Decision Trees and Naive Bayesian Classifiers. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, 609–616. San Francisco, CA, USA.

Zadrozny, B.; and Elkan, C. 2002. Transforming Classifier Scores into Accurate Multiclass Probability Estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, 694–699. New York, NY, USA: Association for Computing Machinery. ISBN 158113567X.

Zagoruyko, S.; and Komodakis, N. 2016. Wide Residual Networks. *ArXiv*, abs/1605.07146.