Exploring the Limits of Epistemic Uncertainty Quantification in Low-Shot Settings

Matias Valdenegro-Toro

German Research Center for Artificial Intelligence, 28359 Bremen, Germany.

matias.valdenegro@dfki.de

Abstract

Uncertainty quantification in neural network promises to increase safety of AI systems, but it is not clear how performance might vary with the training set size. In this paper we evaluate seven uncertainty methods on Fashion MNIST and CIFAR10, as we sub-sample and produce varied training set sizes. We find that calibration error and out of distribution detection performance strongly depend on the training set size, with most methods being miscalibrated on the test set with small training sets. Gradient-based methods seem to poorly estimate epistemic uncertainty and are the most affected by training set size. We expect our results can guide future research into uncertainty quantification and help practitioners select methods based on their particular available data.

1 Introduction

Neural networks are now ubiquitous for many tasks in various fields like computer vision [9], natural language processing, and autonomous driving [3] [14]. Despite their success, these methods generally have problems quantifying their own uncertainty [6], which is necessary for safety, reliability, and overall trustworthiness, particularly when used in human environments.

Bayesian Deep Learning promises good uncertainty estimates [17], but methods often rely in approximations to the bayesian posterior, or quantify uncertainty in approximate [16] [1] or non-bayesian ways [10]. Evaluating the quality of output uncertainty is difficult as there are no labels. For specific details we refer the reader to [5].

Real-world datasets have multiple issues that are not present in academic benchmarks (like CIFAR10, Fashion MNIST, ImageNet, etc), such as low number of samples. There is clear interest on information how uncertainty methods perform under low shot scenarios, where the training set might have low number of samples (less than 1000). It is unclear how these methods might perform, and there is a relationship with model (also known as epistemic) uncertainty, where the lack of information in the training set produces increased uncertainty at the output [2] [8].

In this paper, we evaluate multiple uncertainty methods on sub-sampled versions of two toy datasets, and the Fashion MNIST and CIFAR10 training sets, over a variety of metrics for uncertainty quantification, including entropy, maximum probability, calibration error, and out of distribution performance over several combinations of datasets. An example of our evaluation on the Two Moons dataset (classification) is shown in Figure 1 and a toy example (regression) is shown in Figure 4, where it is clear that uncertainty varies considerably with the training set size.

Bayesian Deep Learning workshop, NeurIPS 2021.

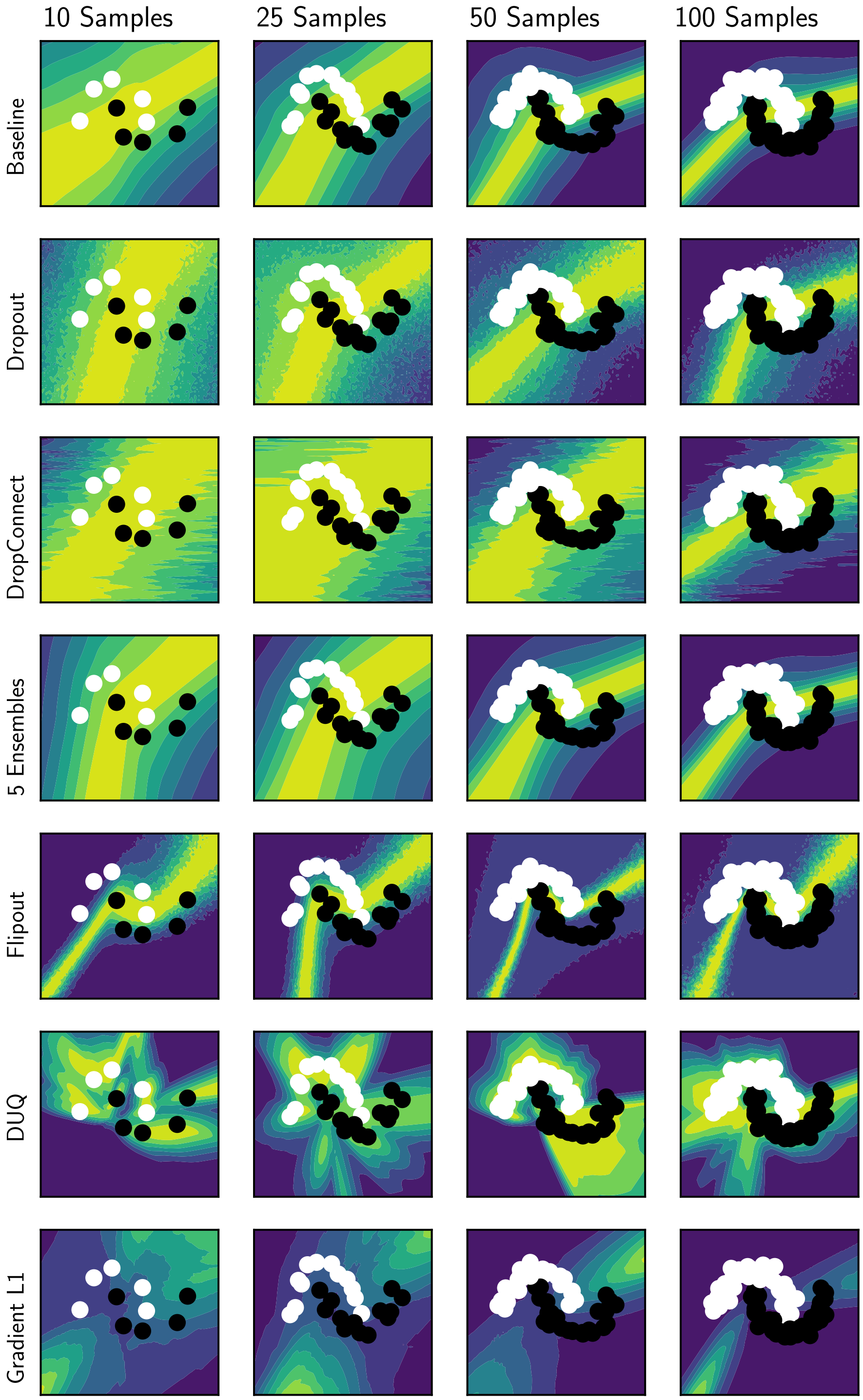


Figure 1: Comparison of uncertainty/confidence in the Two Moons dataset across multiple methods and training set samples per class. This is a synthetic dataset and is available in scikit-learn [13] at https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html. This example visually supports our main conclusion that most uncertainty methods produce miscalibrated predictions at small training set sizes, and that gradient uncertainty is counter-intuitive and overconfident.

2 Experimental Setup

Our principal setup is to sub-sample the training set and train a neural network on this dataset, evaluating a set of metrics related to uncertainty. We sub-sample training sets to a fixed number of samples per class (SPC), namely $S \in [1, 5, 10, 50, 100, 250, 500, 1000, 5000]$ Sub-sampling happens by randomly drawing a fixed number of samples for each class, without replacement. For each value of s, we perform $K = 5$ trials where one model is trained on the sub-sampled training set, and metrics are evaluated on each trial. We report the mean and standard deviation of each metric across the K trials.

Uncertainty Methods. We evaluate MC-Dropout (DO) [4], MC-DropConnect (DC) [11], Deep Ensembles (DE) [10], Direct Uncertainty Quantification (DUQ) [15], Variational Inference with Flipout (VI) [16], and Gradient-based uncertainty (GD) [12]. This selection covers scalable as well as approximate methods and recent advances. Detailed descriptions and hyper-parameters of each method is available in the appendix. We also use a standard CNN architecture without any uncertainty quantification as a comparison baseline (denoted as BL).

Metrics. We use a selection of metrics that measure different aspects of uncertainty quality. As basic metrics we evaluate accuracy, entropy, and maximum probability, all in the test set. For entropy and maximum probability, we are interested in measuring the confidence level of the classifier, and how it changes with the size of the training set. Some additional metrics are:

- **Expected Calibration Error**. We evaluate expected calibration error [6] in the train and test sets. We expect that classifiers should be calibrated independent of the training set size.
- **Out of Distribution Detection**. For each dataset, we define an OOD dataset, and then evaluate the area under the ROC curve (AUC) for different combinations of in distribution (ID) and out of distribution (OOD) datasets, based both on the predictive entropy and maximum probability. The dataset combinations are:
  - Test ID vs Test OOD. This is the standard OOD benchmark that is reported many times in the literature [10] [7].
  - Training ID vs Test OOD. This setting evaluates information in the training set uncertainty, as this dataset increases in size, against the out of distribution test set. This is similar to the below experiment but without evaluating for generalization that is usually done with ID test sets.
  - Training ID vs Test ID. Here we evaluate discrimination between the train and test sets in the in-distribution setting. We expect that as the training set size increases, the AUC should decrease and settle around 0.5, as the uncertainty of the train and test sets would be very similar. This setup validates this assumption experimentally.

  Note that for gradient uncertainty methods, only maximum probability is available, as only a single confidence value is produced.

Datasets. We use Fashion MNIST (with MNIST as OOD dataset), and CIFAR10 (with SVHN as OOD dataset).

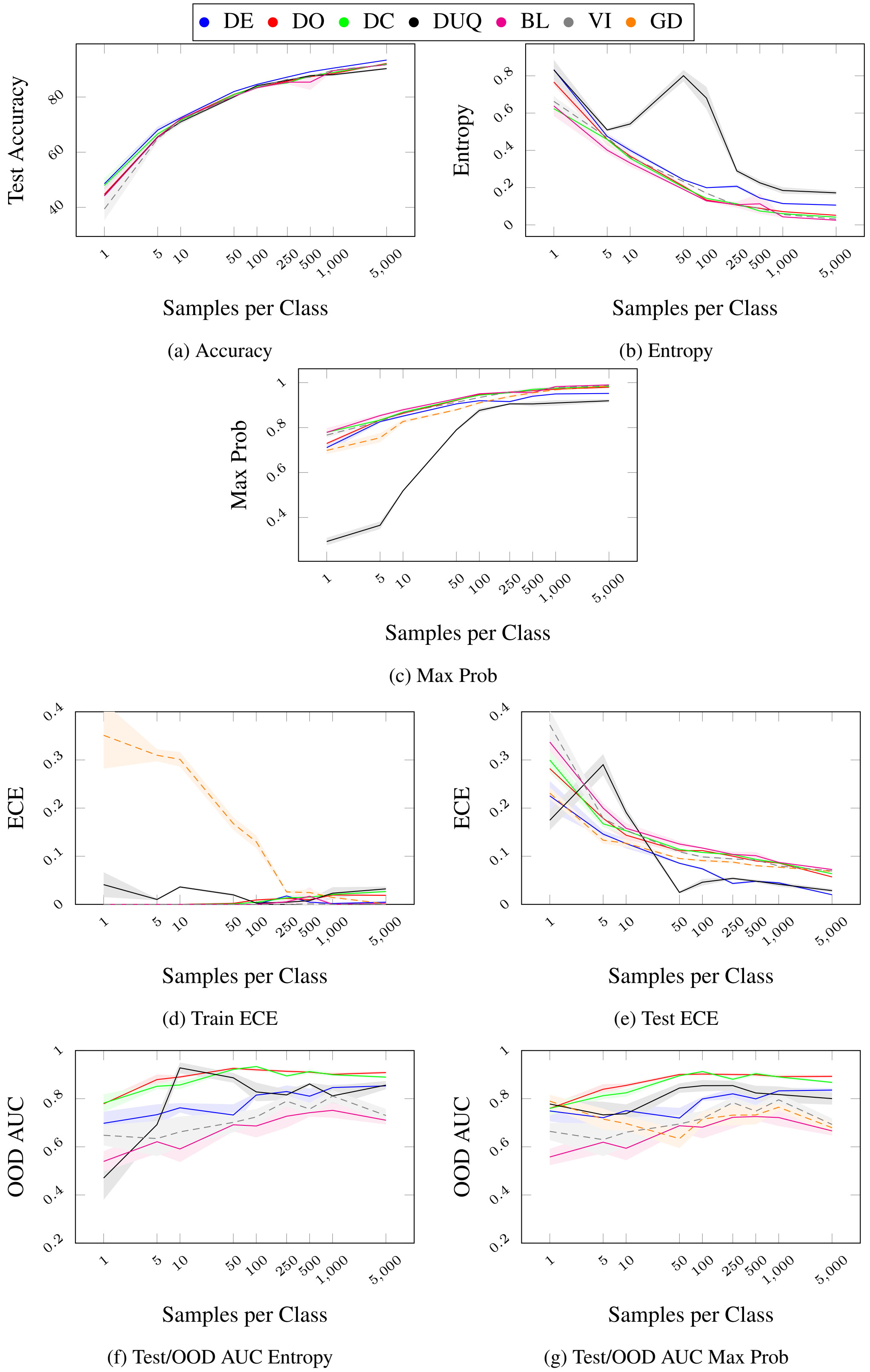
Training. We train each model for 100 epochs using Adam, with a batch size $B = 64$. All models converge in this setting at all training set sizes. A categorical cross-entropy loss is used for all models except for DUQ, which uses a binary cross-entropy loss for training. Information about the network architecture is available in the appendix.

3 Experimental Evaluation and Results

Our main results are presented in Figure 2 for Fashion MNIST, Figure 3 for CIFAR10, and additional out of distribution detection plots in Figures 5 and 6.

In terms of accuracy, all models perform similarly, with ensembles slightly outperforming other methods in most settings, but DropConnect and Dropout are competitive in low shot settings (less than 10 sample per class).

Looking at entropy and maximum probability distributions, DUQ clearly stands out as its the least confident method in terms of maximum probability, and with higher entropy than other methods, and



(a) Accuracy

(b) Entropy

(c) Max Prob

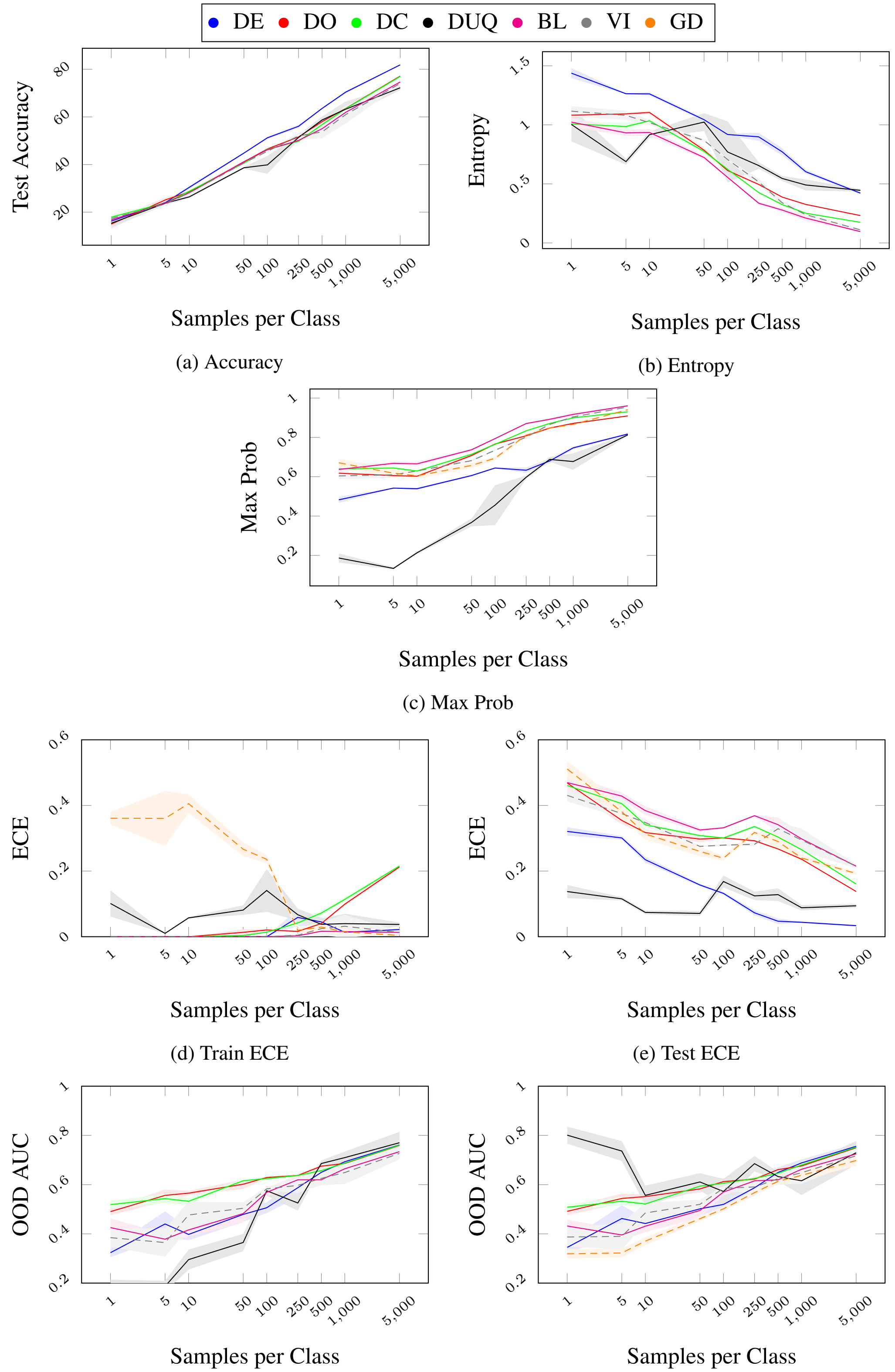
(d) Train ECE

(e) Test ECE

(f) Test/OOD AUC Entropy

(g) Test/OOD AUC Max Prob

Figure 2: Comparison of uncertainty as size of the training set is varied on Fashion MNIST.



(a) Accuracy

(b) Entropy

(c) Max Prob

(d) Train ECE

(e) Test ECE

(f) Test/OOD AUC Entropy

(g) Test/OOD AUC Max Prob

Figure 3: Comparison of uncertainty as size of the training set is varied on CIFAR10.

a peak in entropy around SPC $= 50$. All methods become more confident as the training set increases in size, which we believe indicates good estimation of epistemic uncertainty.

Calibration error shows that most methods are calibrated in the training set across all values of SPC, except for gradient uncertainty which produces uncalibrated probabilities in the low shot setting (SPC < 250). All methods have improved calibration on the test set as SPC increases. The best calibrated method seems to be Deep Ensembles and VI with Flipout, with DUQ producing unstable calibration error in some SPC settings. While there are no guarantees for test calibration error, overall we believe that our results show that many methods could be improved in terms of calibration with different training set sizes.

Out of distribution performance is more varied. In the Test/OOD experiment, all uncertainty methods improve their performance as SPC increases, and this is more evident with entropy than maximum probability. In the Train/Test experiment, all methods start with high AUC when SPC is low, and constantly decrease as SPC increases, signaling that uncertainty cannot be used to discriminate between train and test sets. We think this makes sense and validates the changes in uncertainty as the training set size is varied. With a small training set, it is easier to discriminate the test set since the classifier has not learned good class concepts and it is uncertain enough on the test set.

Finally, the Train/OOD experiment shows that with low SPC it is very easy to discriminate the OOD dataset, since it is very different than the training set knowledge, but this AUC performance decreases with most methods when SPC increases. Only DUQ and Gradient perform in the opposite way. Ensembles in this scenario has almost constant performance. The main takeaways from our experimental results are:

- All methods except gradient, across all training set sizes, are well calibrated in the training set, but miscalibrated on the test set, with improving calibration as the training set size increases (SPC $\uparrow$).
- DUQ is considerably less confident when SPC is low, which indicates that it correctly gauges its own uncertainty, but these results are mixed when looking at test calibration error when SPC < 50.
- Gradient-based methods seems to estimate a very different kind of uncertainty than other methods, with poor Test/OOD AUC performance, and the worse calibration error both in train and test sets. More research is needed in order to understand how these methods work and the effect of their aggregation metric (see Figure 7).
- Ensembles are competitive in terms of accuracy and calibration error, but do not perform as well in some out of distribution detection scenarios (Test/OOD).
- It is not clear if maximum probability or entropy is the best for out of distribution detection, performance varies considerably between the two. It should be chosen carefully [7].
- There is no method that clearly outperforms all others across varied SPC values. Some methods work very well for calibration, but are outperformed in different out of distribution detection settings.
- Selection of uncertainty methods should be done carefully, considering the training set size for a given task.

4 Conclusions and Future Work

In this paper we evaluated the performance of various uncertainty quantification methods as the training set size is varied. We perform a comprehensive evaluation in terms of metrics for uncertainty, including calibration error and out of distribution performance as measured by AUC in several settings. Our overall results show that uncertainty quantification performcance has a strong dependency on the training set size, which can be related to model uncertainty, but this relationship is not always intuitive or predictable. We make clear takeaways for the community to learn from our results.

We expect that our results can guide future research into these methods, in particular for DUQ and gradient-based methods. DUQ provides excellent epistemic uncertainty quantification, while we believe that gradient-based methods do not estimate epistemic uncertainty correctly.

We believe our results can guide practitioners into selecting proper methods to estimate uncertainty of machine learning models.

References

[1] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

[2] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112, 2009.

[3] Di Feng, Ali Harakeh, Steven L Waslander, and Klaus Dietmayer. A review and comparative study on probabilistic object detection in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

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

[5] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342*, 2021.

[6] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

[7] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2017.

[8] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

[9] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *arXiv preprint arXiv:1703.04977*, 2017.

[10] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

[11] Aryan Mobiny, Pengyu Yuan, Supratik K Moulik, Naveen Garg, Carol C Wu, and Hien Van Nguyen. Dropconnect is effective in modeling uncertainty of bayesian deep networks. *Scientific reports*, 11(1):1–14, 2021.

[12] Philipp Oberdiek, Matthias Rottmann, and Hanno Gottschalk. Classification uncertainty of deep neural networks based on gradient information. In *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, pages 113–125. Springer, 2018.

[13] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[14] Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, et al. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420, 2018.

[15] Joost Van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty estimation using a single deep deterministic neural network. In *International Conference on Machine Learning*, pages 9690–9700. PMLR, 2020.

[16] Yeming Wen, Paul Vicol, Jimmy Ba, Dustin Tran, and Roger Grosse. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv preprint arXiv:1803.04386*, 2018.

[17] Andrew Gordon Wilson. The case for bayesian deep learning. *arXiv preprint arXiv:2001.10995*, 2020.

A Classification Neural Network Architecture and Training Setup

All models over all datasets are trained using the same convolutional architecture. The purpose is to get comparable results, not to obtain performance that mimics the state of the art.

The convolutional architecture uses convolution with 64 3×3 filters, followed by 2×2 Max-Pooling, then 128 3×3 filters with 2×2 Max-Pooling, and finally 128 3×3 filters with 2×2 Max-Pooling. The network is complete with two fully connected layers, one with 256 units, and the output layer with C units equal to the number of classes, and a softmax activation. All layers except the output use a ReLU activation, and we insert Batch Normalization layers between Convolutional and Max-Pooling layers.

This network without any uncertainty quantification method applied is denoted as Baseline (BL) in our experiments.

B Details on Uncertainty Methods for Classification

In this section we briefly describe the uncertainty method we used, their hyper-parameters, and any modifications that we made to their training procedures.

MC-Dropout (DO) Dropout sets random activations in a layer to zero, and it is intended as a regularizer that is only applied during training. MC-Dropout [4] enables this activation drop during test/inference time, and the model becomes stochastic, where each forward pass produces one sample from the Bayesian posterior distribution [4]. We use one dropout layer before the last layer, with a drop probability $p = 0.25$. For evaluation we take $M = 50$ forward passes and take the mean over samples as the output prediction.

MC-DropConnect (DC) DropConnect is very similar to Dropout, with the difference being that DropConnect randomly sets weights to zero instead of activations, with the same regularizing effect. MC-DropConnect enables DropConnect at inference time, and it has also been shown to produce samples from the Bayesian posterior distribution [11]. We use a single DropConnect layer at the network output (replacing the standard Dense layer) with drop probability $p = 0.25$. For evaluation we take $M = 50$ forward passes and take the mean over samples as the output prediction.

Deep Ensembles (DE) Ensembling is a standard method in Machine Learning, where outputs of several models are combined, which usually produces a better model. It has been shown [10] that ensembles also have excellent uncertainty quantification properties. We use an ensemble of neural networks with the same architecture and $N = 5$ ensemble members.

Direct Uncertainty Quantification (DUQ) This method [15] replaces the standard softmax classifier with a radial basis function (RBF) classifier, where the output layer learns a weight matrix and a centroid for each class, using the minimum distance to a centroid to decide which class to output, and the centroid distance as an uncertainty measure. Centroids are updated using a running mean on input feature space, but we found that this method is unstable and does not converge in a low-show setting, so we learn the centroids using gradient descent.

VI with Flipout (VI) Variational inference is a popular method which models weights as approximate distributions, usually selecting a Gaussian distribution [1], where the components of the kernel and bias matrices are Gaussian distributions. This transforms the model into a stochastic one. Flipout [16] is used as an additional formulation on top of a stochastic perturbation model that reduces variance, greatly improving learning stability and performance. To effectively learn across different training set sizes, we disable the use of a prior, and only the weights are modeled as a Gaussian distribution, the bias being a fixed learnable scalar value. For evaluation we take $M = 50$ forward passes and take the mean over samples as the output prediction. Our network architecture uses Flipout VI only in the output layer.

Gradient Uncertainty (GD) This method [12] computes the gradient of the loss with respect to trainable parameters, using a virtual label that is the one-hot encoded version of the predicted label, and passes the gradient vector through an aggregation function that produces a scalar, which can be used as an uncertainty measure. This can only be done in a classification setting. We normalize the aggregated gradient g to the $[0, 1]$ range using min-max normalization and transform it to a pseudo probability as $p = 1 - g$, which can be used to evaluate calibration error and maximum probability metrics. We found that the aggregation metric has a large impact on performance across training set sizes, as shown in Figure 7, and overall we use the L1 metric as it performs the best on most metrics (as shown in Figure 7). Note that this method produces a single confidence value for a prediction.

C Code Implementation

Source code implementation is available at `https://github.com/mvaldenegro/paper-quality-epistemic-uncertainty-bayes`.

This implementation uses Keras 2.2.4, TensorFlow 1.14, and Keras-Uncertainty (available at `https://github.com/mvaldenegro/keras-uncertainty`).

Models were trained on a single RTX 2070 GPU. Each model instance takes from 1 to 15 minutes to train (depending on training set size), with all experiments taking less than 24 hours of GPU time.

D Additional Regression Toy Example

In this section we present a small regression toy example, corresponding to sampling the function $f(x) = \sin(x) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma(x))$ and $\sigma(x) = 0.15(1 + e^{-x})^{-1}$ where $x \in [-4, 4]$. The interval $[-4, 4]$ is sampled at $S \in [25, 50, 100, 200]$ equally spaced samples, and various models are trained using a mean squared error loss. Evaluation happens on a fixed sized set at ranges $[-7, 7]$. This result is presented in Figure 4.

For the regression setting, we use a Negative Log-Likelihood (NLL) loss to capture epistemic uncertainty [10] [9], which is formulated below:

$$L(\mathbf{x}, y) = 0.5N^{-1} \sum_i \left(\log \sigma^2(\mathbf{x}_i) + \frac{(\mu(\mathbf{x}_i) - y_i)^2}{\sigma^2(\mathbf{x}_i)} \right) \tag{1}$$

This loss requires that the model contains two output heads, one for the mean $\mu_i(\mathbf{x})$ and another for the variance $\sigma^2(\mathbf{x})$. The variance head uses a softplus activation function in order to always predict positive soft variances.

With this loss, the output variance $\sigma^2(x)$ can be interpreted as an estimate of the aleatoric uncertainty in the data. Epistemic uncertainty is computed through a specific uncertainty quantification method.

The network architecture is two fully connected layers with 32 units each, and a ReLU activation, and two output heads (mean and variance) with a single neuron each, and a linear activation for the mean head. No Batch Normalization layers are used in this example. Models that do not predict variance use only a single output head.

Figure 4 shows that Flipout without NLL (using mean squared error instead) cannot estimate aleatoric uncertainty, while using the NLL loss the model can estimate aleatoric uncertainty correctly. Classical NN, Ensembles, and Flipout + NLL do not correctly fit the training set with a low number of samples (less than 100 samples), while Flipout and MC Dropout and MC DropConnect do estimate the training function more closely. For Flipout + NLL and Ensembles, their large uncertainty indicates that it is an incorrect fit and shows that their uncertainty is useful.

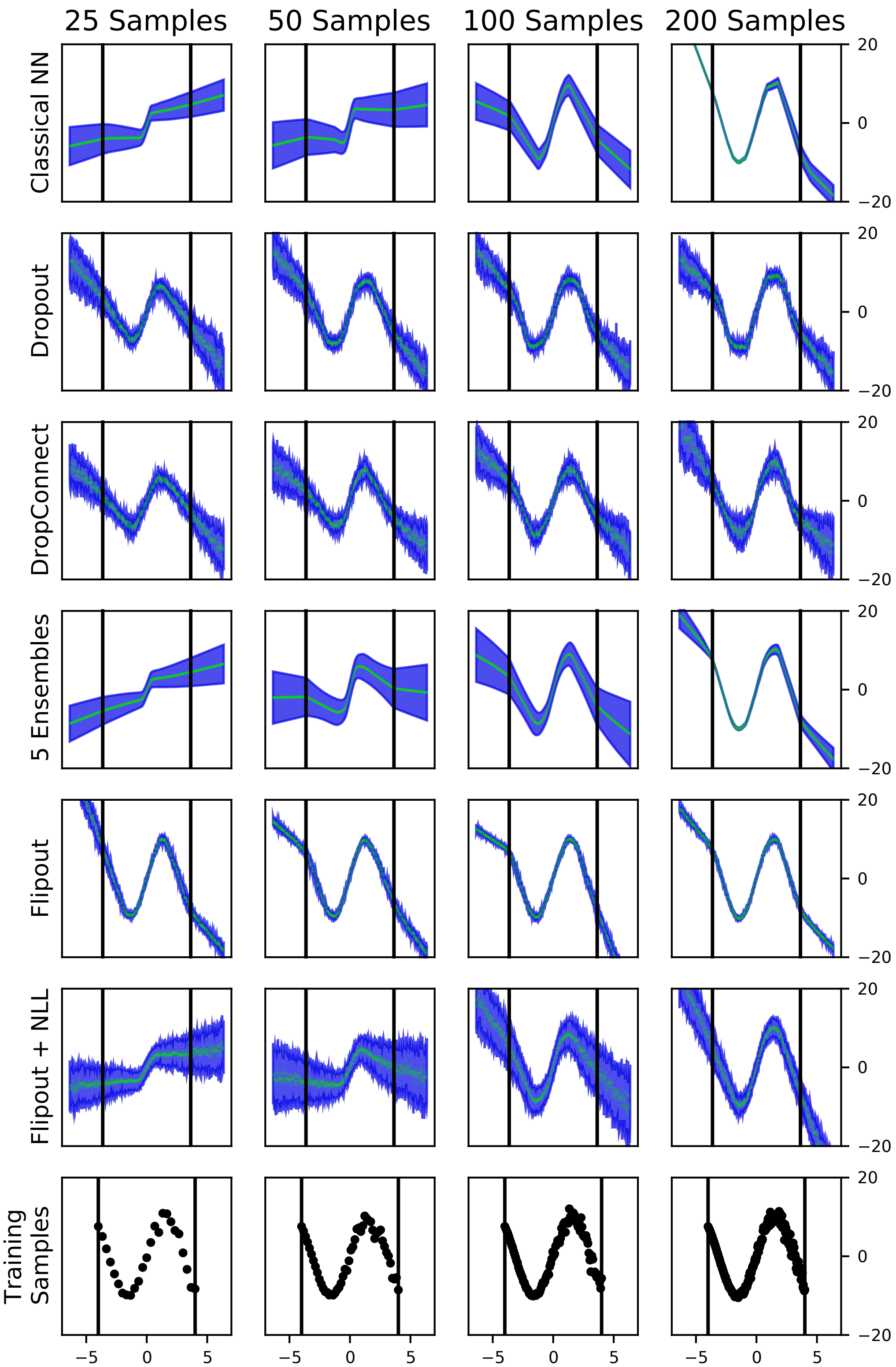
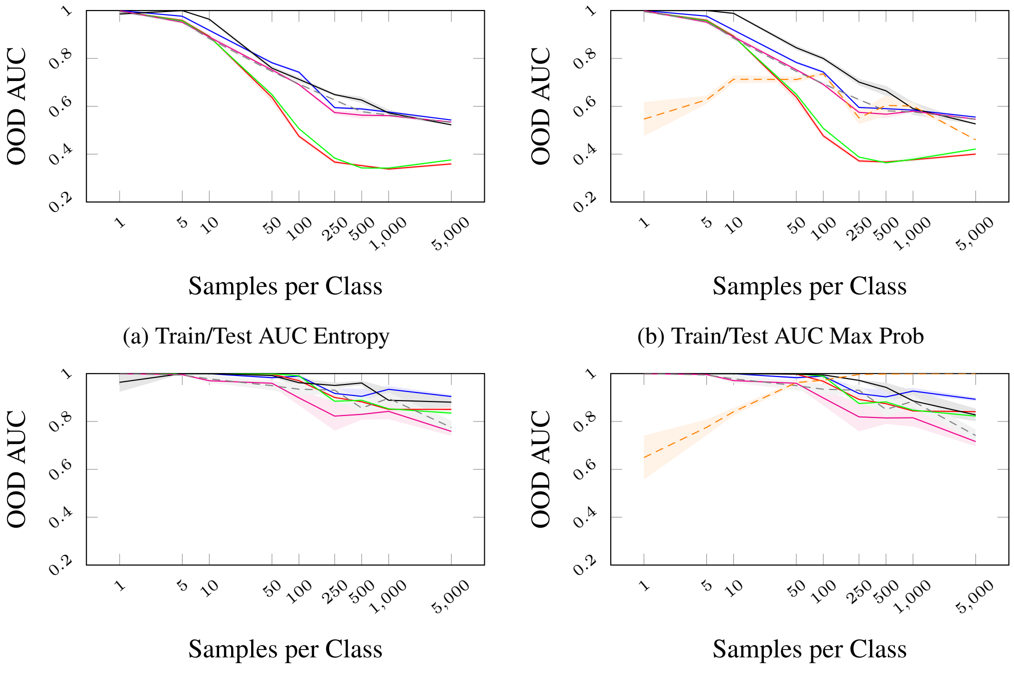


Figure 4: Comparison of uncertainty in the toy regression example, as number of training samples is varied. The last row shows the samples used for training. The two black lines indicate the limits of the training set, while the out of distribution test set ranges at $[-7, -4] \cup [4, 7]$. Green represents the mean, while the blue shaded areas are one standard deviation uncertainty. These plots clearly show that uncertainty degrades with small training sets.

E Out of Distribution Detection Performance



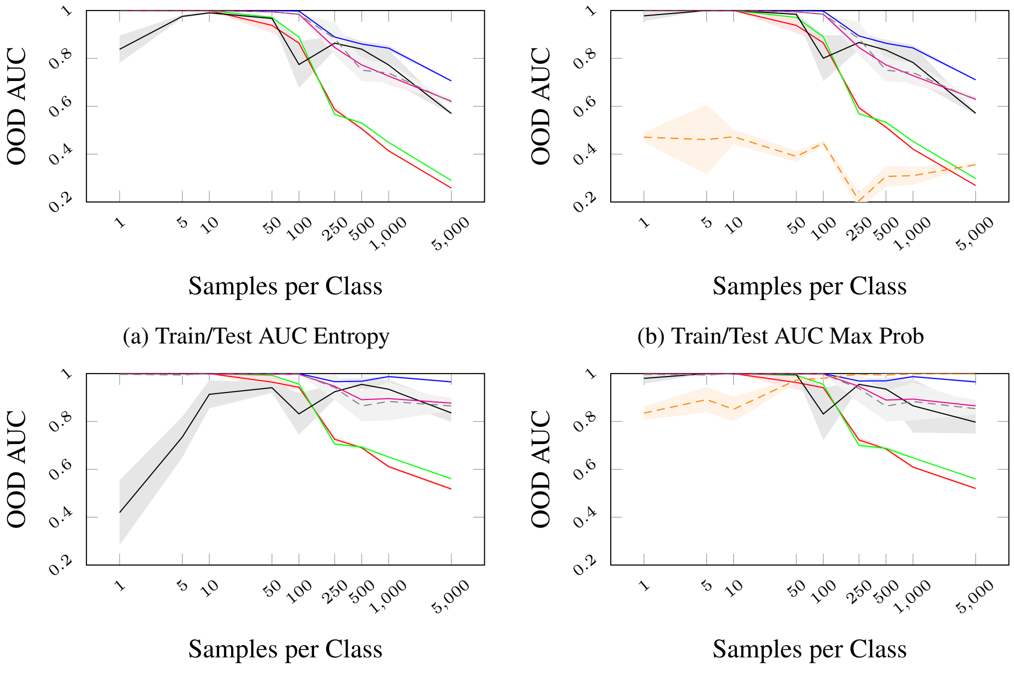
(a) Train/Test AUC Entropy

(b) Train/Test AUC Max Prob

(c) Train/OOD AUC Entropy

(d) Train/OOD AUC Max Prob

Figure 5: Comparison of Out of Distribution Performance on Fashion MNIST



(a) Train/Test AUC Entropy

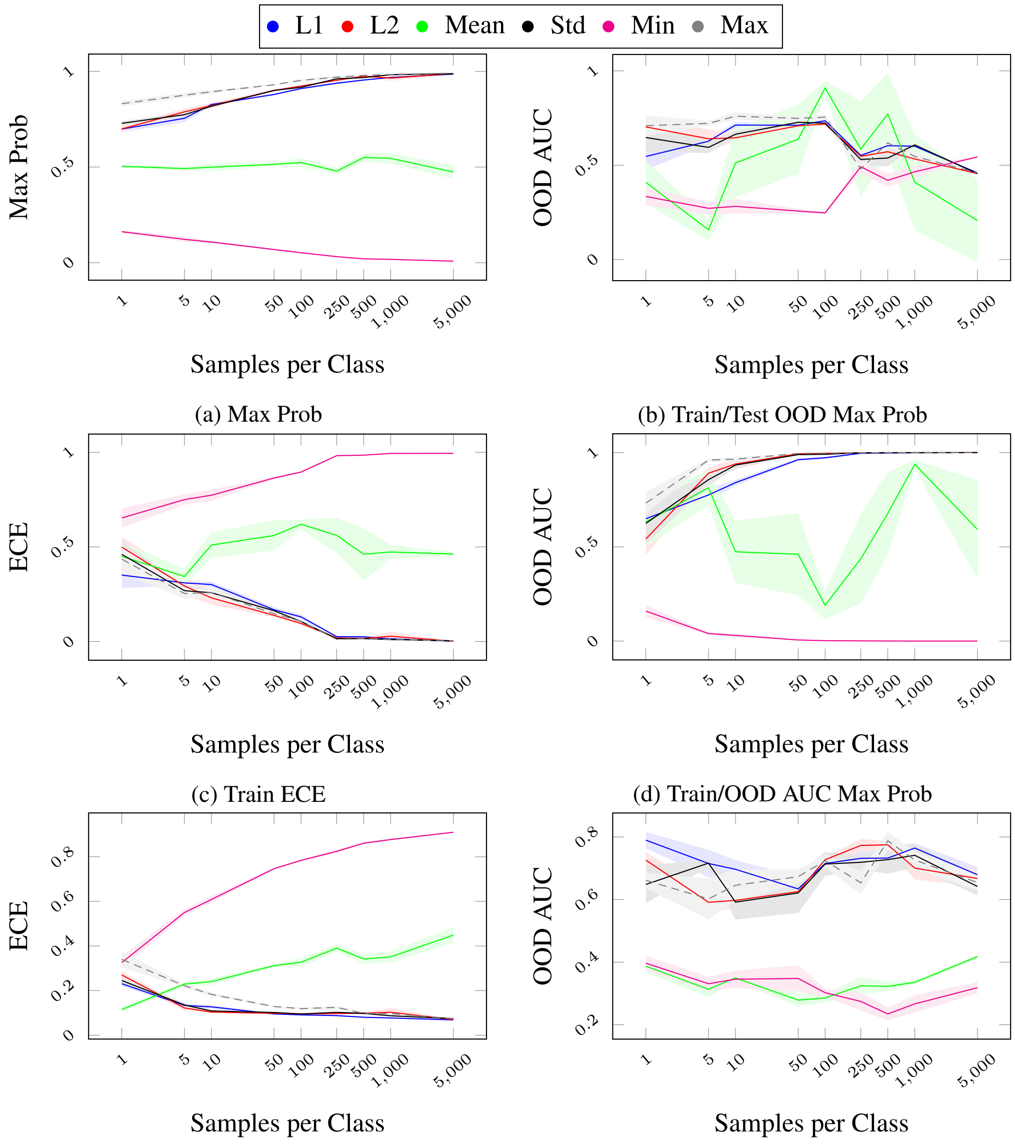
(b) Train/Test AUC Max Prob

(c) Train/OOD AUC Entropy

(d) Train/OOD AUC Max Prob

Figure 6: Comparison of Out of Distribution Performance on CIFAR10

F Gradient Uncertainty Performance



(a) Max Prob

(b) Train/Test OOD Max Prob

(c) Train ECE

(d) Train/OOD AUC Max Prob

(e) Test ECE

(f) Test/OOD AUC Max Prob

Figure 7: Comparison of Gradient uncertainty on Fashion MNIST across training set sizes, with different aggregation metrics.