Uncertainty Estimation Using a Single Deep Deterministic Neural Network

Joost van Amersfoort [1] **Lewis Smith** [1] **Yee Whye Teh** [2] **Yarin Gal** [1]

Abstract

We propose a method for training a deterministic deep model that can find and reject out of distribution data points at test time with a single forward pass. Our approach, deterministic uncertainty quantification (DUQ), builds upon ideas of RBF networks. We scale training in these with a novel loss function and centroid updating scheme and match the accuracy of softmax models. By enforcing detectability of changes in the input using a gradient penalty, we are able to reliably detect out of distribution data. Our uncertainty quantification scales well to large datasets, and using a single model, we improve upon or match Deep Ensembles in out of distribution detection on notable difficult dataset pairs such as Fashion-MNIST vs. MNIST, and CIFAR-10 vs. SVHN.

1. Introduction

Estimating uncertainty reliably and efficiently has remained an open problem with many important applications such as guiding exploration in Reinforcement Learning (Osband et al., 2016) or as a method for selecting data points for which to acquire labels in Active Learning (Houlsby et al., 2011). Until now, most approaches for estimating uncertainty in deep learning rely on ensembling (Lakshminarayanan et al., 2017) or Monte Carlo sampling (Gal & Ghahramani, 2016). In this paper, we introduce a deep model that is able to estimate uncertainty in a single forward pass. We call our model DUQ, Deterministic Uncertainty Quantification, and we construct it by re-examining ideas originally suggested in the 90s. We combine these with recent advances and make a number of improvements which enable scalable training of modern deep learning architectures. We evaluate our model against the current best approach for estimating uncertainty in Deep Learning, Deep
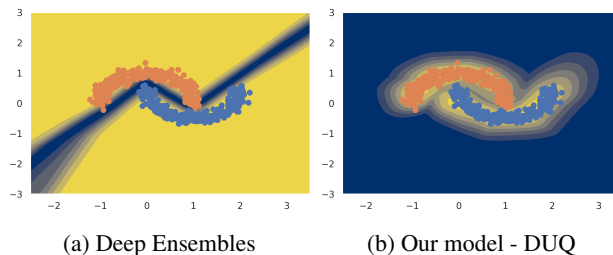

(a) Deep Ensembles   (b) Our model - DUQ

Figure 1. Uncertainty results on two moons dataset. Yellow indicates high certainty, while blue indicates uncertainty. DUQ is certain only on the data distribution, and uncertain away from it: the ideal result. Deep Ensembles is uncertain only along the decision boundary, and certain elsewhere.

Ensembles, and show that DUQ compares favourably on a number of evaluations, such as out of distribution (OoD) detection of FashionMNIST vs MNIST, and CIFAR vs. SVHN. We visualise how DUQ performs on the two moons dataset in Figure 1. We see that DUQ is only certain on the training data, and its certainty decreases away from it. Deep Ensembles are not able to obtain meaningful uncertainty on this dataset, because of a lack of diversity in the different models in the ensemble. We make our code publicly available[1].

DUQ consists of a deep model and a set of feature vectors corresponding to the different classes (*centroids*). A prediction is made by computing a kernel function, a distance function, between the feature vector computed by the model and the centroids. This type of model is called an *RBF network* (LeCun et al., 1998a) and uncertainty is measured as the distance between the model output and the closest centroid. A data point for which the feature vector is far away from all centroids does not belong to any class and can be considered out of distribution. In this paper, we define uncertainty to be *predictive* uncertainty.

The model is trained by minimising the distance to the correct centroid, while maximising it with respect to the others. This incentivises the model to put the features of training data close to a particular centroid, however there is no mechanism that dictates what should happen away from the training data. Therefore we need to enforce that DUQ is

---

[1]OATML, Department of Computer Science, University of Oxford [2]Department of Statistics, University of Oxford. Correspondence to: Joost van Amersfoort <joost.van.amersfoort@cs.ox.ac.uk>.

[1]https://github.com/y0ast/deterministic-uncertainty-quantification

sensitive to changes in the input, such that we can reliably detect out of distribution data and avoid mapping out of distribution data to in distribution feature representations — an effect we call *feature collapse.* The upper bound of this sensitivity can be quantified by the Lipschitz constant of the model. We are interested in models for which this sensitivity is not too low, but also not too high, because that could hurt generalisation and optimisation. DUQ achieves this result by regularising the Jacobian with respect to the input, as was first introduced by Drucker & Le Cun (1992).

In practice, RBF networks prove difficult to optimise, because of instability of the centroids and a saturating loss. We propose to make training stable by updating the centroids using an exponential moving average of the feature vectors of the data points assigned to them, as was introduced in van den Oord et al. (2017). We use a "one vs the rest" loss function minimising the distance to the correct centroid, while maximising the other distances. We find that these two changes stabilise training and lead to accuracies that are similar to the standard softmax and cross entropy set up on standard datasets such as FashionMNIST and CIFAR-10.

Uncertainty quantification in deep neural networks with a softmax output is generally done by measuring the entropy of the predictive distribution, so the maximally uncertain output is achieved by uniformly assigning probabilities over all the classes. The only way to achieve a uniform output for out of distribution data, is by training on additional data and hoping it generalises to out of distribution samples at test time. This does not happen in practice, and it is found that the only uncertainty that can reliably be captured by looking at the entropy of the softmax distribution is aleatoric uncertainty (Gal, 2016; Hein et al., 2019). In DUQ, it is possible to predict that none of the classes seen during training is a good fit, when the distance between the model output and all centroids is large.

The contributions of this paper are as follows:

- We stabilise training of RBF networks and show, for the first time, that these type of models can achieve competitive accuracy versus softmax models.

- We show how two-sided Jacobian regularisation makes it possible to obtain reliable uncertainty estimates for RBF networks.

- We obtain excellent uncertainty in a single forward pass, while maintaining competitive accuracy.

2. Methods

DUQ consists of a deep feature extractor, such as a ResNet (He et al., 2016), but without the softmax layer. Instead, we have one learnable weight matrix W_c per class, c. Using the
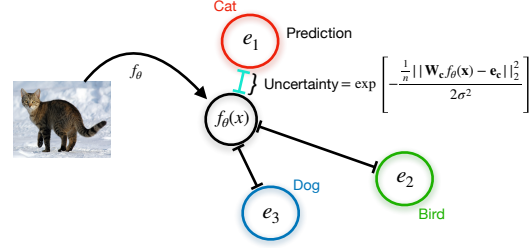

Figure 2. A depiction of the architecture of DUQ. The input is mapped to feature space, where it is assigned to the closest centroid. The distance to that centroid is the uncertainty.

output and the class centroids, we compute the exponentiated distance between the model output and the centroids:

$$K_c(f_\theta(\mathbf{x}), \mathbf{e}_c) = \exp\left[-\frac{\frac{1}{n}||\mathbf{W}_c f_\theta(\mathbf{x}) - \mathbf{e}_c||_2^2}{2\sigma^2}\right], \quad (1)$$

with $f_\theta : \mathbb{R}^m \to \mathbb{R}^d$ our model, m the input dimension, d the output dimension, and parameters θ. $\mathbf{e}_c$ is the centroid for class c, a vector of length n. $\mathbf{W}_c$ is a weight matrix of size n (centroid size) by d (feature extractor output size) and σ a hyper parameter sometimes called the length scale. This function is also referred to as a Radial Basis Function (RBF) kernel. The class dependent weight matrix allows features insensitivity on a class by class basis, minimising the potential for feature collapse. A prediction is made by taking the class c with the maximum correlation (minimum distance) between data point $\mathbf{x}$ and class centroids $\mathbf{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_C\}$:

$$\arg\max_c K_c(f_\theta(\mathbf{x}), \mathbf{e}_c). \quad (2)$$

we define the uncertainty in this model as the distance to the closest centroid, i.e. replacing the $\arg\max$ operator by a $\max$ in Equation (2).

The loss function is the sum of the binary cross entropy between each class' kernel value $K_c(\cdot, \mathbf{e}_c)$, and a one-hot (binary) encoding of the label. For a particular data point $\{\mathbf{x}, \mathbf{y}\}$ in our data set $\{X, Y\}$:

$$L(\mathbf{x}, \mathbf{y}) = -\sum_c y_c \log(K_c) + (1 - y_c) \log(1 - K_c) \quad (3)$$

where we shortened $K(f_\theta(\mathbf{x}), \mathbf{e}_c)$ as K_c. During training, we average the loss over a minibatch of data points, and perform stochastic gradient descent on θ and $\mathcal{W} = \{\mathbf{W}_1, \cdots, \mathbf{W}_c\}$. The class centroids, $\mathbf{E}$, are updated using an exponential moving average of the feature vectors of data points belonging to that class. If the model parameters, θ and $\mathcal{W}$, are held constant, then this update rule leads to the closed form solution for the centroids that minimises the

loss:

$$N_{c,t} = \gamma * N_{c,t-1} + (1 - \gamma) * n_{c,t} \tag{4}$$

$$\mathbf{m}_{c,t} = \gamma * \mathbf{m}_{c,t-1} + (1 - \gamma) \sum_i \mathbf{W}_c f_\theta(\mathbf{x}_{c,t,i}) \tag{5}$$

$$\mathbf{e}_{c,t} = \frac{\mathbf{m}_{c,t}}{N_{c,t}} \tag{6}$$

where $n_{c,t}$ is the number of data points assigned to class c in minibatch t, $\mathbf{x}_{c,t,i}$ is element i of a minibatch at time t, with class c. γ is the momentum, which we usually set between [0.99, 0.999]. This method of updating centroids was introduced in the Appendix of van den Oord et al. (2017) for updating quantised latent variables. The high momentum leads to stable optimisation that is robust to initialisation.

The proposed set up leads to the centroids being pushed further away at each minibatch, without converging to a stable point. We avoid this by regularising the l_2 norm of θ. This restricts the model to sensible solutions and aids optimisation.

2.1. Gradient Penalty

As discussed in the introduction, without further regularisation deep networks are prone to feature collapse. We find that it can be avoided by regularising the representation map using a gradient penalty. Gradient penalties were first introduced to aid generalisation in Drucker & Le Cun (1992), who named it "double backpropagation". Recently, this type of penalty has been used successfully in training Wasserstein GANs (Gulrajani et al., 2017) to regularise the Lipschitz constant.

In our set up, we consider the following two-sided penalty:

$$\lambda \cdot \left[||\nabla_x \sum_c K_c||_2^2 - 1 \right]^2, \tag{7}$$

where $||\cdot||_2$ is the l_2 norm and the targeted Lipschitz constant is 1. We found empirically that regularising the gradient of $\sum_c K_c$ works better than $f_\theta(x)$ or $\mathbf{K}_c(x)$ (which is the vector of kernel distances for input x). A similar approach was taken for softmax models by Ross & Doshi-Velez (2018).

The two-sided penalty was introduced by Gulrajani et al. (2017), who mention that despite a one-sided penalty being sufficient to satisfy their requirements, the two sided penalty proved to be better in practice. The one-sided penalty is defined as:

$$\lambda \cdot \max(0, ||\nabla_x \sum_c K_c||_F^2 - 1). \tag{8}$$

In Section 4.1, we show the difference between the single and two sided penalties experimentally. We find the two-sided penalty to be ideal for enforcing sensitivity, while still allowing strong generalisation.

2.2. Intuition about Gradient Penalty

A gradient penalty enforces smoothness, limiting how quickly the output of a function changes as the input x changes. Smoothness is important for generalisation, especially if we are using a kernel which depends on distances in the representation space. It is simple to show that regularising the l_2 norm of the Jacobian, J, enforces a Lipschitz constraint at least locally, since for a small region around x we have $g(x + \epsilon) - g(x) \simeq J_g(x)\epsilon \leq ||J(x)||_2||\epsilon||_2$.

However, smoothness still leaves us vulnerable to the feature collapse problem outlined earlier, where multiple inputs are mapped to the same $g(x)$. Lipschitz smooth functions can collapse their inputs — the constant function $g(x) = c$ is Lipschitz for any Lipschitz constant L. Collapsing features can be beneficial for accuracy, but it hurts our ability to perform out of distribution detection, since it has the potential to make input points indistinguishable in the representation space. We find empirically in our work that the two sided penalty is extremely important: using the one sided penalty, i.e, enforcing only smoothness, is not sufficient to produce the sensitive behaviour we want in our representation. This can be seen in Figure 4b, in contrast to Figure 1b with the two-sided penalty.

By keeping the norm of the Jacobian above some value, intuitively we encourage sensitivity of the learnt function, by preventing it from collapsing to a locally constant function, ignoring all changes in the input space. This argument is speculative, as this regularisation scheme has no effect on sensitivity in directions orthogonal to the local Jacobian, and more work is needed to explain definitively exactly why this penalty seems to encourage sensitivity, as it would seem mathematically that collapsing the representation would still be possible. However, we find empirically that it is important for preserving out of distribution performance. In Appendix C, we evaluate a number of alternative approaches such as using a reversible model as feature extractor (guaranteed to be invertible) and computing the Jacobian with respect to the vector K_c and $f_\theta(x)$.

2.3. Epistemic and Aleatoric Uncertainty

When quantifying uncertainty, it can be useful to distinguish between "epistemic" and "aleatoric" uncertainty. Epistemic uncertainty comes from uncertainty in the parameters of the model. This uncertainty is high for out of distribution data, but also for example for informative data points in active learning (Houlsby et al., 2011). Aleatoric uncertainty is uncertainty inherent in the data such as an image of a 3 that is similar to an 8 (Smith & Gal, 2018). In this case, the true class cannot be determined.

In practice, DUQ captures both aleatoric and epistemic uncertainty. Informally, when a point is far from all cen-
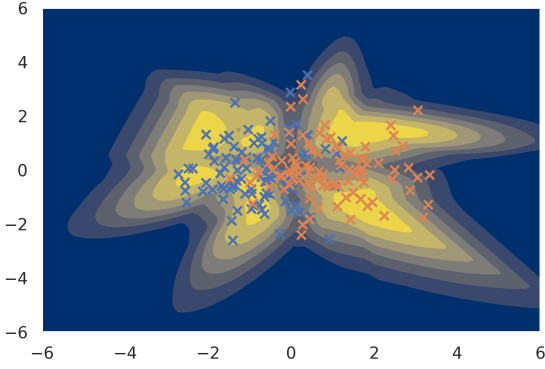
Figure 3. The uncertainty learned by DUQ on a simple problem of classifying samples from two overlapping Gaussian distributions. Yellow indicates certainty, while blue indicates uncertainty. There is significant aleatoric uncertainty due to the overlap between the classes. DUQ can express high aleatoric uncertainty by placing centroids close to each other in feature space, and is able to learn this in practice if the task needs it, as shown here by the higher uncertainty around the 0 mark on the x axis.

troids in feature space there is epistemic uncertainty. While aleatoric uncertainty is expressed by placing centroids so that they are close in feature space (see Figure 3) and mapping a data point close to both of them. It is important that the centroids are close in feature space, because otherwise the model would not map them in between as it incurs a large loss, following Equation 3. We do not currently have a formal way to distinguish between these two kinds of uncertainty in DUQ. Solving this problem is an interesting direction for future research.

2.4. Why Sensitivity can be at odds with Classification

In this section we analyse some of the trade-offs and assumptions encoded in detecting out-of-distribution inputs. We show in a toy experiment that standard classification losses can hurt out-of-distribution detection. Consider fitting a model on a problem with two features, x_1 and x_2, both sampled from a unit Gaussian, and output y, such that $y = \text{sign}(x_1) * \epsilon$, where ϵ is noise with a low probability of flipping the label. The optimal decision function in terms of the empirical risk, no matter the algorithm, is the function $f(x_1, x_2) = \text{sign}(x_1)$. But this says nothing about the out of distribution behaviour. What happens if we now see the input $x_1, x_2 = 1, 1000$? By our definition of the problem, this is out of distribution, as it lies many standard deviations away from the observed data. But should it be detected as out of distribution? The data does not define what could be given as the input, at least if we take a conventional empirical risk minimisation approach.

In this situation, it seems natural to prefer the kind of decisions which would be made by a generative model, for

example. If x_1 and x_2 represent medical data, then presumably a highly abnormal value for x_2 is notable, and we would like to detect it. However, if x_2 is a truly irrelevant variable, say, the temperature on the surface of a distant planet, then presumably our model is correct to ignore its value, even if the value of the irrelevant variable is highly abnormal. When training using empirical risk minimisation, features not relevant to classification accuracy can simply be ignored by the feature extractors of a neural network. This makes out-of-distribution detection more difficult using feature space methods, even those that use a distance loss as we do. It is important to note that there is a potential tension here with classification accuracy. Enforcing sensitivity can make accurate classification harder because it forces the model to represent changes in input — as in the example above, these may be irrelevant to the causal structure of the problem. If we know about invariances that are appropriate for the problem at hand, we can enforce these by corresponding construction of the network. For example, we enforce translation invariance by using convolutional networks in this paper.

3. Related Work

The largest body of research on obtaining uncertainty in deep learning are Bayesian neural networks (MacKay, 1992; Neal, 2012). While exact inference in them is intractable, a range of approximate methods have been proposed. Mean-field variational inference methods, such as *Bayes by Backprop* (Blundell et al., 2015) and Radial BNNs (Farquhar et al., 2020) are a promising direction but have not yet lead to stable training on large image datasets. A more scalable alternative is MC Dropout (Gal & Ghahramani, 2016), which is very simple to implement and evaluate. In practice, these variational Bayesian methods are outperformed by Deep Ensembles (Lakshminarayanan et al., 2017). This is a simple, non-Bayesian, method that involves training multiple deep models from different initialisations and a different data set ordering. Snoek et al. (2019) showed that Deep Ensembles consistently outperform Bayesian neural networks that were trained using variational inference. This performance comes at the expense of computational cost, Deep Ensembles' memory and compute use scales linearly with the number of ensemble elements at both train and test time.

Aside from using discriminative models, there have also been attempts at finding out of distribution data using generative models. Nalisnick et al. (2019a) showed that simply measuring the likelihood under the data distribution does not work. Recently, a more advanced approach that involves separating the likelihood of the semantic foreground from the background did show promising results on selected datasets (Ren et al., 2019). While generative models are a

promising avenue for out of distribution detection, they are not able to assess predictive uncertainty; given that a data point is in distribution, can our discriminative model actually make a reliable prediction? Further, generative models are significantly more expensive to train than classification models.

Our approach is distinct from both ensembles/Monte Carlo methods, which aim to find different explanations for the data and increase uncertainty when these disagree, and generative models which model the data distribution directly. Instead our approach is more related to pre-deep learning kernel methods (Quinn & Sugiyama, 2014; Schölkopf et al., 2000), such as Gaussian processes which revert to a prior away from data, and Support Vector Machines, where the distance to the separating hyperplane is informative of the uncertainty. These approaches have never scaled to high dimensional data, because of a lack of well performing kernel functions.

The decision function based on kernel distances was first used in the context of convolutional neural networks by LeCun et al. (1998a). They were quickly abandoned for softmax models, because they were difficult to scale and optimise with gradient-based approaches due to saturating gradients and unstable centroids. Notable improvements in our work over the original are the updating mechanism of the centroids and the loss function that is based on a multivariate Bernoulli, solving the problems of unstable centroids and saturating gradients.

Regularising the Jacobian has a long history, starting with Drucker & Le Cun (1992) and more recently Ross & Doshi-Velez (2018). Both papers aim to regularise the l_2 norm of Jacobian down to zero. In the first case to obtain better generalisation, while the second paper aims to achieve adversarial robustness and interpretability. In neither case are the authors interested in *increasing* the Jacobian. Gulrajani et al. (2017) showed how a gradient penalty can be applied to training GANs with the Wasserstein distance, which was a more scalable and simpler alternative to weight clipping. They use the double sided penalty and mention it works better in practice. Follow up work has analysed the penalty in more detail and concluded that, contrary to our case, for training Wasserstein GANS the one-sided penalty is preferable theoretically and practically (Jolicoeur-Martineau & Mitliagkas, 2019; Petzka et al., 2017).

4. Experiments

We show the behaviour of DUQ in two dimensions, with the two moons dataset and show the effect of leaving out the gradient penalty and using a one sided penalty. We continue by looking at the out of distribution detection performance for some notable difficult data set pairs (Nalisnick et al.,
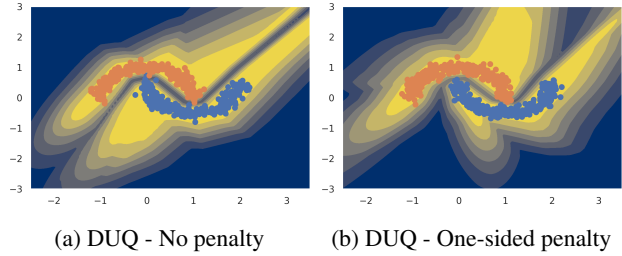

(a) DUQ - No penalty   (b) DUQ - One-sided penalty

Figure 4. Uncertainty results for two variations of DUQ: left without gradient penalty, and right with a one-sided gradient penalty ($\lambda = 1$). Yellow indicates certainty, while blue indicates uncertainty. Both results are significantly worse than DUQ with a two-sided penalty.

2019a), such as FashionMNIST vs MNIST, and CIFAR-10 vs SVHN. We further study sensitivity to two important hyper parameters the length scale σ and gradient penalty weight λ and propose how to tune them without relying on example OoD data.

4.1. Two Moons

We use the scikit-learn (Pedregosa et al., 2011) implementation of this dataset and describe the model architecture and optimisation details in Appendix A.1. For colouring the visualisations, we normalise the colour map within the figure.

The result of our model trained with a two-sided gradient penalty is shown in Figure 1b. The uncertainty is exactly as one would expect for the two moons dataset: certain on the training data, uncertain away from it and in the heart within the two moons. The difference with Deep Ensembles is striking (Figure 1a). The uncertainty for DUQ is quantified as the distance to the closest centroid (max over the kernel distances), the uncertainty for Deep Ensembles is computed as the predictive entropy of the average output, see Appendix B. The ensemble elements were trained separately using the same model as described in Appendix A.1, but without L2 regularisation to encourage diverse solutions.

Discussion While Figure 1b is an impressive result in deep learning, it is worth highlighting that Gaussian processes are able to obtain such result too. A good visualisation can be found in Bradshaw et al. (2017). Interestingly, even though Deep Ensembles have been successfully applied to many large datasets (Snoek et al., 2019), they fail to estimate uncertainty well on the two moons dataset. This is due to the simplicity and low dimensionality of this dataset, the ensembles generalise in nearly the same way — with a diagonal line dividing the top left and the bottom right.

Gradient Penalty In Section 2.1, we introduced the two-sided gradient penalty. Figure 4 shows why it is important. In Figure 4a, we show the result of having no gradient

penalty, which shows that the model is certain every far away from the data. In Figure 4b, we see that the uncertainty does not improve when only a one-sided penalty is applied. In both cases, there are 'blobs' sticking out of the training data domain that are also classified with high certainty.

Hyper parameters We found classification performance on two moons to be insensitive to our setting of the gradient penalty weight λ, likely because of the simplicity of the two moons dataset. For the uncertainty visualisation, we found it important to set the length scale to be small (in the interval $[0.05, 0.5]$), despite accuracy not being affected by this hyper parameter. In the following experiments, we will discuss methods for picking the length scale and the weight of the gradient penalty.

4.2. FashionMNIST vs MNIST

In this experiment, we assess the quality of our uncertainty estimation by looking at how well we can separate the test set of FashionMNIST (Xiao et al., 2017) from the test set of MNIST (LeCun et al., 1998b) by looking only at the uncertainty predicted by the model. We train our model on FashionMNIST and we expect it to assign low uncertainty to the FashionMNIST test set, but high uncertainty to MNIST, since the model has never seen that dataset before and it is very different from FashionMNIST.

During evaluation we compute uncertainty scores on both test sets and measure for a range of thresholds how well the two are separated. As in previous work (Ren et al., 2019), we report the AUROC metric, where a higher value is better and 1 indicates that all FashionMNIST data points have a higher certainty than all MNIST data points. We picked FashionMNIST vs MNIST, because it is a notably difficult dataset pair (Nalisnick et al., 2019a), while MNIST vs NotMNIST (Bulatov, 2011) is much simpler.

Experimental set up Our model is a three layer convolutional network and we report all architectural and optimisation details in Appendix A.2. It is important to note that at test time we set Batch Normalization to evaluation mode, meaning that we use the mean and standard deviation of the feature activations computed from the training set (i.e. FashionMNIST). It is unlikely that in practice we would get an entire batch of (uncorrelated) OoD points, so we can not normalise using test time batch statistics[2]. Further, we use the same data normalisation for the out of distribution set as the in distribution set. Skipping either of these steps makes the problem artificially simple.

Length Scale Most hyper parameters, such as as the learning rate or weight decay parameter, can be set using the standard train/validation split. However there are two hyper
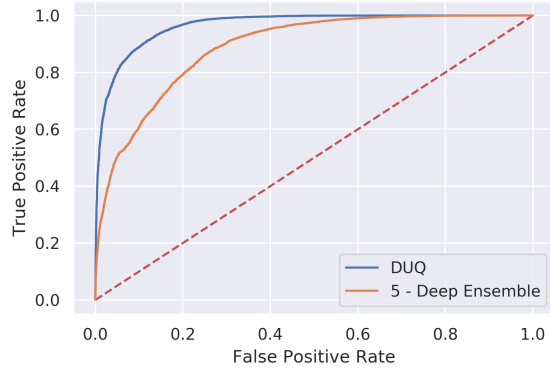

Figure 5. ROC curve for DUQ trained on FashionMNIST and evaluated on FashionMNIST and MNIST. The task is to separate these data sets based on uncertainty estimates.

parameters that are particularly important: the length scale σ and the gradient penalty weight λ. We set the length scale by doing a grid search over the interval $(0, 1]$ while keeping $\lambda = 0$. We pick the value that leads to the highest validation accuracy. Following this process, we found that a length scale of 0.1 leads to the highest accuracy, as measured over five runs. While this process might not result in a length scale that leads to the best OoD performance, it works well in practice.

Gradient Penalty Setting the λ parameter is more involved: from Section 2.4, we know that the accuracy can suffer as a result from gaining the ability to do out of distribution detection, so we cannot rely on it to select the best λ. We also cannot use the AUROC score on the MNIST dataset, because that would give the method an unfair advantage: we cannot assume access to the OoD set in advance in practice.[3] Instead we use a third dataset on which we evaluate the AUROC and select our λ values based on that. We follow previous work (Ren et al., 2019) and use NotMNIST as the third dataset for this pair. The results can be seen in Table 1. As expected, the accuracy goes down as λ increases, and we also observe that the best AUROC result for NotMNIST coincides with the best score for MNIST, which shows that the strategy of selecting a hyper parameter based on the NotMNIST data set is reasonable. We note that while NotMNIST generalises to MNIST, we cannot rely on this property in general. Therefore, we propose an alternative method for model selection based on predictive uncertainty in Section 4.3.

Comparison We show our results and compare with alternative methods in Table 2. Our proposed method, DUQ, outperforms all other classification based methods. The only method that is better is LL ratio (Ren et al., 2019), which is based on generative models. These type of models are more

---

[2]Just one datapoint needs to have significantly different activation statistics for the entire batch to be easily detectable.

[3]If we do assume access, then we can trivially train a binary classifier on the original and OoD set.
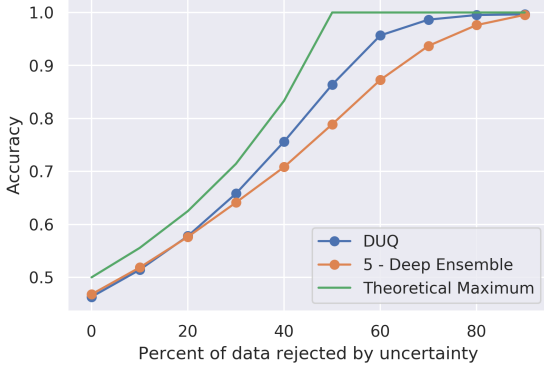
Figure 6. Rejection classification plot: accuracy on a combination of FashionMNIST and MNIST test sets. The x-axis indicates the proportion of data rejected based on the uncertainty score. The theoretical maximum is computed from a classifier with 100% accuracy on FashionMNIST and rejects all MNIST points first.

Figure 7. A histogram of uncertainty estimates as computed using DUQ ($\lambda = 0.5$). CIFAR-10 and SVHN are clearly separated. The counts are normalised, because the SVHN test set is significantly larger than CIFAR-10's.

computationally costly to train than DUQ. The PixelCNN++ (Salimans et al., 2017) used by LL ratio for FashionMNIST uses 2 blocks of 5 gated ResNet layers, while our model is a simple three layer convolutional network. An alternative, competitive approach is Mahalanobis Distance (Lee et al., 2018), which computes a distance in the feature space of a pretrained softmax/cross entropy model in combination with a number of dataset specific augmentations. The method relies on hyper parameter tuning using 1,000 samples of the out of distribution dataset.

The difference in AUROC between our Deep Ensemble result and Ren et al. (2019)'s is due to using different architectures. For a fair comparison, we use the same architecture for the ensemble elements as for DUQ (replacing the class dependent final layer by the usual single linear layer). In Figure 5, we show the complete ROC curve for our implementation of Deep Ensembles and DUQ. We see that DUQ outperforms Deep Ensembles at all chosen rates.

Accuracy and Gradient Penalty To confirm that training using DUQ's distance based output achieves competitive accuracy, we train two models using our architecture: the standard softmax and cross entropy set up and DUQ with $\lambda = 0$. We obtained $92.4\% \pm 0.1$ accuracy for the softmax model, and for our proposed set up $92.4\% \pm 0.2$, both averaged over five runs. The results show that we can obtain competitive accuracy using DUQ, resolving previous problems with RBF networks. In Table 1, we show how accuracy changes for an increasingly weighted gradient penalty. The accuracy only degrades slightly, while AUROC is improved.

Rejection Classification In Figure 6, we visualise how well these algorithms work in a more realistic scenario. We combine the FashionMNIST and MNIST test sets, then we reject
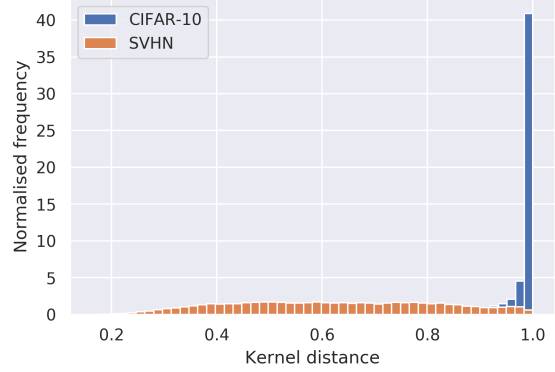
a certain portion of the combined dataset by uncertainty score. Next we compute the accuracy on the remaining data for each portion, considering all predictions on the OoD MNIST set to be incorrect. We expect the accuracy to go up as we reject more of the data points on which the model is uncertain. Ideally, we reject the incorrectly classified FashionMNIST points and all MNIST points. The *Theoretical Maximum* is computed by assuming a model that has perfect accuracy on the FashionMNIST test set **and** is able to reject all MNIST data before any FashionMNIST data. This experiment combines out of distribution detection, with detecting difficult to classify data points, which is closer to actual deployment scenarios than the AUROC metric, and also a suggested practically informed evaluation method by Filos et al. (2019). Note that the ensemble model has an accuracy of 93.6% on FashionMNIST, giving it a 1.2% head start on DUQ, which has an accuracy of 92.4%. We see that DUQ outperforms Deep Ensembles in this more realistic scenario.

λ	Acc (FM)	AUROC (NM)	AUROC (M)
0	$92.4\% \pm .2$	$0.933 \pm .009$	$0.948 \pm .004$
0.05	**$92.4\% \pm .2$**	**$0.946 \pm .018$**	**$0.955 \pm .007$**
0.1	$92.4\% \pm .1$	$0.938 \pm .0018$	$0.948 \pm .005$
0.2	$92.2\% \pm .1$	$0.945 \pm .019$	$0.944 \pm .011$
0.3	$92.3\% \pm .1$	$0.944 \pm .013$	$0.941 \pm .011$
0.5	$92.0\% \pm .1$	$0.946 \pm .014$	$0.932 \pm .009$
1.0	$91.9\% \pm .1$	$0.945 \pm .018$	$0.934 \pm .006$

Table 1. FM stands for FashionMNIST, NM for NotMNIST, and M for MNIST. The results are mean/std computed from 5 experiment repetitions. We show AUROC for separating FashionMNIST from NotMNIST and MNIST; higher is better. We see that the gradient penalty improves AUROC performance slightly, but performance on this dataset pair is already very strong.

4.3. CIFAR-10 vs SVHN

In this section we look at the CIFAR-10 dataset (Krizhevsky et al., 2014), with SVHN (Netzer et al., 2019) as OoD set. We use a ResNet-18 (He et al., 2016) as feature extractor $f_\theta(\cdot)$, specifically the version provided by PyTorch (Paszke et al., 2017) with some minor modifications: we use 64 filters in the first convolutional layer, and skip the first pooling operation and last linear layer. CIFAR-10 is a difficult dataset for out of distribution detection for several reasons. There is a significant amount of data noise: some of the dog and cat examples are not distinguishable using only 32 by 32 pixels. The training set is small compared to its complexity, making it easy to overfit without data augmentation.

Experimental set up As in the previous section, we tune the length scale using the accuracy on the validation set, and find that 0.1 works best from a range of $[0.05, 1]$. We train for a fixed 75 epochs and reduce the learning rate by a factor of 0.2 at 25 and 50 epochs. We use random horizontal flips and random crops as data augmentation and find that this is enough regularisation to prevent the model from overfitting. All architectural and optimisation details are described in Appendix A.3. We obtain an accuracy of $94.1\% \pm 0.2$ using the standard softmax/cross entropy loss. A Deep Ensemble of several softmax models obtains an accuracy of 95.2%. DUQ without a gradient penalty ($\lambda = 0$) obtains $94.2\% \pm 0.2$ accuracy, while accuracy of DUQ with $\lambda = 0.5$ is $93.2\% \pm 0.4$.

Gradient Penalty For CIFAR-10, we do not use a third dataset to set λ. Instead, we avoid using more data and use in-distribution uncertainty. We measure this using the AUROC of detecting correctly and incorrectly classified validation set data points using the predicted uncertainty. We found that optimising λ using this procedure also transfer to λ values that lead to strong out of distribution detection performance. In general, this approach is preferable over using a third dataset, because it is difficult to find an appropriate out of distribution dataset, which will have the same characteristics as those encountered during deployment. Imagine a particular difficult traffic situation or an

Method	AUROC
DUQ	0.955
LL ratio (generative model)	0.994
Single model	0.843
5 - Deep Ensembles (ours)	0.861
5 - Deep Ensembles (ll)	0.839
Mahalanobis Distance (ll)	0.942

Table 2. Results on FashionMNIST, with MNIST as OoD set. Deep Ensembles is by Lakshminarayanan et al. (2017), Mahalanobis Distance by Lee et al. (2018), LL ratio by Ren et al. (2019). Results marked by (ll) are obtained from Ren et al. (2019), (ours) is implemented using our architecture. Single model is our architecture, but trained with softmax/cross entropy.
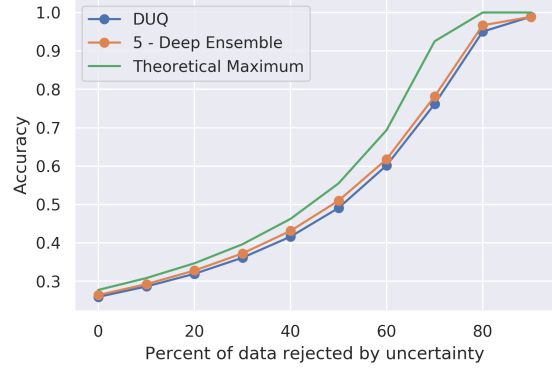

Figure 8. Rejection classification plot, which shows model performance on a mix of CIFAR-10 and SVHN, while rejecting uncertain points. The theoretical maximum is achieved when a hypothetical classifier obtains 100% accuracy on CIFAR-10 and rejects all SVHN data points first. We see that DUQ and a 5 element Deep Ensemble perform very similar.

MRI scan which shows a new type of disease, these scenarios have no reasonable out of distribution set available. Generative models are not able to take this approach, because they do not have predictive uncertainty. Even if we use a hybrid model (Nalisnick et al., 2019b), then the discriminative part, a softmax/cross entropy model, does not have reliable predictive uncertainty.

Results In Figure 7, we show a a normalised histogram for the kernel distances of CIFAR-10 and SVHN. We see that most of CIFAR-10 is very close to 1, while SVHN is uniformly spread out over the range of distances. This shows that DUQ works as expected and that out of distribution data ends up away from all of the centroids in feature space.

The rejection classification plot, Figure 8, is created similar to the previous experiment in the last section. Note that this time the *Theoretical Maximum* line is significantly lower, because the SVHN test set contains close to $26{,}000$ elements, while CIFAR-10's only contains $10{,}000$. This means that the best possible accuracy when 100% of the data is considered is about 28%. We see that DUQ and Deep Ensembles perform similarly.

In Table 3, we compare DUQ with several alternative methods. We see that DUQ performs competitively with a number of recent approaches. Interestingly, on these more complicated data sets Deep Ensembles performs the best. We suspect this is because the complexity of the data set allows the ensemble elements to be more diverse while still explaining the data well.

We further see a significant gap between DUQ with and without a gradient penalty: there is a big improvement going from $\lambda = 0$ to $\lambda = 0.5$. We suspect this is because there is a lot of within class variation, which incentivises the model to collapse more diverse data points to the class centroids.

Method	AUROC
DUQ ($\lambda = 0.5$)	0.927 ± 0.013
DUQ ($\lambda = 0$)	0.861 ± 0.032
LL ratio (generative model)	0.930
Single model	0.906 ± 0.007
3 - Deep Ensembles	0.926 ± 0.010
5 - Deep Ensembles	0.933 ± 0.008
10 - Deep Ensembles	0.941
15 - Deep Ensembles	0.942

Table 3. Deep Ensembles is by Lakshminarayanan et al. (2017), but re-implemented and evaluated using our architecture. LL ratio is as reported in Ren et al. (2019). Single model is our architecture, but trained with softmax/cross entropy. We show the AUROC for separating CIFAR-10 from SVHN.

Runtime One of the main advantages of DUQ over Deep Ensembles is computational cost. For Deep Ensembles, both computation and memory cost scale linearly in the number of ensemble components, during both train and test time. DUQ has to compute the Jacobian at training time, which is expensive, but at test time there is only a marginal overhead over a softmax based model. Training for one epoch on a modern 1080 Ti GPU, takes 21 seconds for a softmax/cross entropy model, which leads to 105 seconds for a Deep Ensemble with 5 components. DUQ with gradient penalty needs 103 seconds for one epoch at training time, but only 27 seconds without gradient penalty. DUQ is 25% slower at test time than single softmax/cross entropy model, but about 4 times faster than a Deep Ensemble with 5 components.

5. Conclusion

We introduced DUQ, Deterministic Uncertainty Quantification, a simple method for obtaining uncertainty using a deep neural network in a single forward pass. Evaluations show that our method is better in some scenarios and competitive in others with the more computationally expensive Deep Ensembles.

Interesting future work would be to place DUQ in a probabilistic framework, enabling a calibrated notion of uncertainty and a rigorous way of separating out epistemic and aleatoric uncertainty.

6. Acknowledgements

References

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pp. 1613–1622, 2015.

Bradshaw, J., Matthews, A. G. d. G., and Ghahramani, Z. Adversarial examples, uncertainty, and transfer testing robustness in gaussian process hybrid deep networks. *arXiv preprint arXiv:1707.02476*, 2017.

Bulatov, Y. notmnist dataset. *Tech. Rep.[Online]. Available: http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html*, 2011.

Dinh, L., Krueger, D., and Bengio, Y. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Drucker, H. and Le Cun, Y. Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks*, 3(6):991–997, 1992.

Farquhar, S., Osborne, M., and Gal, Y. Radial Bayesian neural networks: Beyond discrete support in large-scale bayesian deep learning. *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, 2020.

Filos, A., Farquhar, S., Gomez, A. N., Rudner, T. G., Kenton, Z., Smith, L., Alizadeh, M., de Kroon, A., and Gal, Y. A systematic comparison of bayesian deep learning robustness in diabetic retinopathy tasks. *arXiv preprint arXiv:1912.10481*, 2019.

Gal, Y. Uncertainty in deep learning. *University of Cambridge*, 1:3, 2016.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pp. 1050–1059, 2016.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pp. 5767–5777, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hein, M., Andriushchenko, M., and Bitterwolf, J. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 41–50, 2019.

Hoffman, J., Roberts, D. A., and Yaida, S. Robust learning with jacobian regularization. *arXiv preprint arXiv:1908.02729*, 2019.

Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

Hutchinson, M. F. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

Jacobsen, J.-H., Smeulders, A., and Oyallon, E. i-revnet: Deep invertible networks. In *ICLR 2018-International Conference on Learning Representations*, 2018.

Jolicoeur-Martineau, A. and Mitliagkas, I. Connections between support vector machines, wasserstein distance and gradient-penalty gans. *arXiv preprint arXiv:1910.06922*, 2019.

Krizhevsky, A., Nair, V., and Hinton, G. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55, 2014.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pp. 6402–6413, 2017.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998a.

LeCun, Y., Cortes, C., and Burges, C. J. The mnist database of handwritten digits, 1998. *URL http://yann.lecun.com/exdb/mnist*, 10:34, 1998b.

Lee, K., Lee, K., Lee, H., and Shin, J. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, pp. 7167–7177, 2018.

MacKay, D. J. *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology, 1992.

Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D., and Lakshminarayanan, B. Do deep generative models know what they don't know? In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=H1xwNhCcYm.

Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D., and Lakshminarayanan, B. Hybrid models with deep and invertible features. *arXiv preprint arXiv:1902.02767*, 2019b.

Neal, R. M. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. The street view house numbers (svhn) dataset, 2019.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pp. 4026–4034, 2016.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

Petzka, H., Fischer, A., and Lukovnicov, D. On the regularization of wasserstein gans. *arXiv preprint arXiv:1709.08894*, 2017.

Quinn, J. A. and Sugiyama, M. A least-squares approach to anomaly detection in static and sequential data. *Pattern Recognition Letters*, 40:36–40, 2014.

Ren, J., Liu, P. J., Fertig, E., Snoek, J., Poplin, R., Depristo, M., Dillon, J., and Lakshminarayanan, B. Likelihood ratios for out-of-distribution detection. In *Advances in Neural Information Processing Systems 32*, pp. 14680–14691, 2019.

Ross, A. S. and Doshi-Velez, F. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

Schölkopf, B., Williamson, R. C., Smola, A. J., Shawe-Taylor, J., and Platt, J. C. Support vector method for novelty detection. In *Advances in neural information processing systems*, pp. 582–588, 2000.

Smith, L. and Gal, Y. Understanding measures of uncertainty for adversarial example detection. *arXiv preprint arXiv:1803.08533*, 2018.

Snoek, J., Ovadia, Y., Fertig, E., Lakshminarayanan, B., Nowozin, S., Sculley, D., Dillon, J., Ren, J., and Nado, Z. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems 32*, pp. 13969–13980. Curran Associates, Inc., 2019.

van den Oord, A., Vinyals, O., et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pp. 6306–6315, 2017.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.