Project Report for Deep Learning: Uncertainty Estimation Using a Single Deep Deterministic Neural Network

Wendong LIANG, Jingyi LI
Université Paris-Saclay - Institut Polytechnique de Paris
wendong.liang@ens-paris-saclay.fr, jingyi.li@ip-paris.fr

May 4, 2022

Abstract

Deterministic uncertainty quantification (DUQ)[12] was proposed to find and reject out of distribution (OoD) data points at test time with a single pass. DUQ is built on an adapted RBF network [6] with a novel loss function related to two-sided gradient penalty and the feature distance between the model output and the updated centroids. In this report, we plan to provide a critical analysis of DUQ through methodological analysis and experimental reproduction. Furthermore, we do some extended explorations such as changing the dataset, doing grid search for hyperparameters and comparing different methods. Our code is available here: `https://drive.google.com/file/d/1r_1bEI62Kbdnktpwx6lzuZncYHGp2oYQ/view?usp=sharing`.

1 Problem Statement

The paper introduces a deep model called DUQ that is able to estimate uncertainty in a single forward pass. DUQ consists of a deep model and a set of feature vectors corresponding to the different classes (centroids). The uncertainty is measured as the distance between the model output and the closest centroid. The model is trained by minimising the distance to the correct centroid and maximising it to others. But for the data points that are not in the training data and whose feature vector is far away from all centroids, called "out of distribution (**OoD**) data", there is no mechanism to dictate what should happen.

Therefore, DUQ should be enforced to be 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 called *feature collapse*. They are interested in models whose sensitivity is not too low, but also not too high, because that could decrease generalisation and optimisation.

In addition, DUQ is based on Radial Basis Function (RBF) networks, but in practice, RBF networks prove difficult to optimise because of instability of the centroids and a saturating loss. The author improved it with centroid updating scheme, as was introduced in [13].

2 DUQ Method

There are two key parts in the DUQ method. The first part is the adaptation of RBF network with binary cross entropy loss and centroid updating scheme to make the training stable. The second part is a two-sided gradient penalty which helps keep a balance between smoothness and sensitiveness of RBF network. A description of the architecture of DUQ is shown in Figure 1.

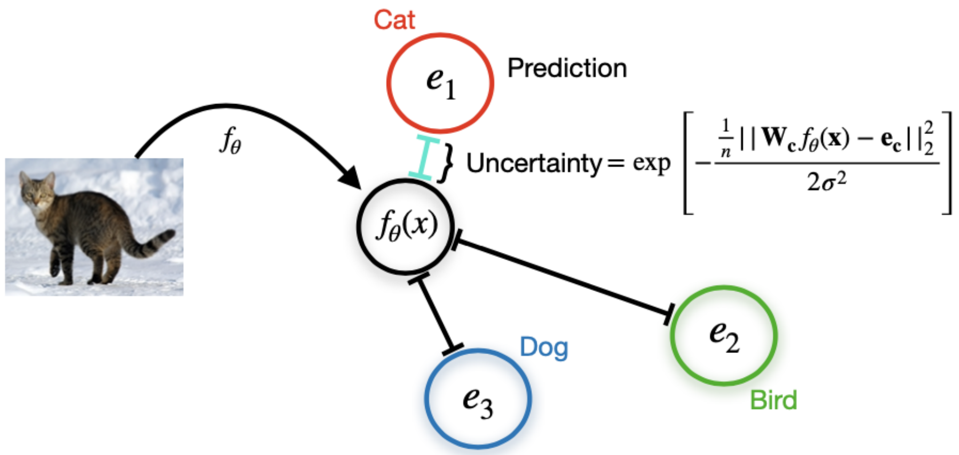

Cat
e_1
Prediction
f_θ
Uncertainty $= \exp\left[-\frac{\frac{1}{n}||\mathbf{W_c} f_\theta(\mathbf{x}) - \mathbf{e_c}||_2^2}{2\sigma^2}\right]$
$f_\theta(x)$
Dog
e_2
e_3
Bird

Figure 1: The structure of DUQ. The input image is first mapped to the feature space, where it is assigned to the closest centroid. The distance to that centroid represents the uncertainty.

2.1 RBF network

DUQ consists of a deep feature extractor but without a softmax layer. Instead, they have one learnable weight matrix W_c per class c. They compute the exponential distance between the model output and centroids:

$$K_c(f_\theta(x), e_c) = exp[-\frac{\frac{1}{n} \parallel W_c f_\theta(x) - e_c \parallel_2^2}{2\sigma^2}] \tag{1}$$

with our model $f_\theta : \mathbb{R}^m \to \mathbb{R}^d$, the input dimension m, the output dimension d, and parameters θ. e_c is the centroid for class c, a vector of length n. W_c is a weight matrix of size n (centroid size) by d (feature extractor output size) and σ is a hyper-parameter sometimes called the length scale. This function is also referred to as a Radial Basis Function (RBF) kernel. A prediction is made by taking the class c with the maximum correlation (minimum distance) between data point x and class centroids $E = \{e_1, ..., e_C\}$:

$$\arg\max_c K_c(f_\theta(x), e_c) \tag{2}$$

They define the uncertainty in this model as the distance to the closest centroid:

$$\max_c K_c(f_\theta(x), e_c) \tag{3}$$

As mentioned above, RBF networks are difficult to optimise because of instability of the centroids and a saturating loss. So the author improved it on the following two points.

2.1.1 Loss Function

For a particular data point {x,y} in the data set $\{\mathcal{X}, \mathcal{Y}\}$, the loss function is the sum of the binary cross entropy:

$$L(x,y) = -\sum_{x} y_c log(K_c) + (1-y_c) log(1-K_c) \tag{4}$$

where $K(f_\theta(x), e_c)$ is shortened as Kc. During training, they average the loss over a minibatch of data points and perform stochastic gradient descent on θ and $\mathcal{W} = \{W_1, ..., W_c\}$.

2.1.2 Centroid Updating Scheme

The class centroids, 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 fixed, then this update rule leads to the closed form solution for the centroids that minimises the loss:

$$N_{c,t} = \gamma * N_{c,t-1} + (1-\gamma) * n_{c,t} \tag{5}$$

$$m_{c,t} = \gamma * m_{c,t-1} + (1-\gamma) \sum_i W_c f_\theta(x_{c,t,i}) \tag{6}$$

$$e_{c,t} = \frac{m_{c,t}}{N_{c,t}} \tag{7}$$

where $n_{c,t}$ is the number of data points assigned to class c in minibatch t, $x_{c,t,i}$ is element i of a minibatch at time t with class c. γ is the momentum, which is usually set between $[0.99, 0.999]$. The high momentum leads to stable optimisation that is robust to initialisation.

2.2 Two-sided Gradient Penalty

As discussed in the problem statement part, deep networks are prone to feature collapse without feature regularisation and they find that feature collapses can be avoided by regularising the representation map using a gradient penalty, which has been used successfully in training Wasserstein GANs [1] to regularise the lipschitz constant.

In their set up, they consider the following two-sided penalty:

$$\lambda \cdot [\| \nabla_x \sum_c K_c \|_2^2 - 1]^2 \tag{8}$$

where $\| \cdot \|_2$ is the l_2 norm and the targeted Lipschitz constant is 1.

There is also a one-sided penalty defined as:

$$\lambda \cdot \max(0, \| \nabla_x \sum_c K_c \|_F^2 - 1) \tag{9}$$

In the experiment part, the authors showed the difference between the single and two sided penalties and found that the two-sided penalty is ideal for enforcing sensitivity, while still allowing strong generalisation.

The intuitive understanding of the two-sided gradient penalty is that the output of a function shouldn't change too quickly as the input x changes in order to maintain smoothness, but the output shouldn't change too slowly either to enforce sensitivity.

2.3 Epistemic and Aleatoric Uncertainty

The authors proposed 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 [3]. In DUQ method, there is epistemic uncertainty when a point is far from all centroids in feature space.

Aleatoric uncertainty is uncertainty inherent in the data such as an image of a 3 that is similar to an 8[10]. For DUQ, it happens when a data point is close to both centroids.

This paper currently does not have a formal way to distinguish between these two kinds of uncertainty in DUQ.

3 Experiments

In this section, we first conduct experiments mentioned in the paper, then we make some further improvements and explorations on this basis.

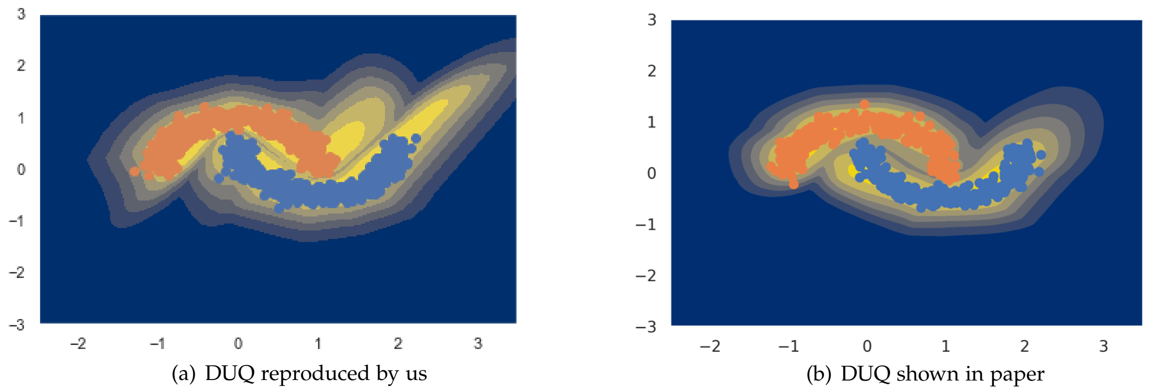
3.1 Reproduction

Specifically, we first validate the effectiveness of the two-sided gradient penalty on the toy dataset, two-moons. Then we verify the out of distribution detection performance for two dataset pairs: FashionMNIST vs MNIST, and CIFAR-10 vs SVHN.

3.1.1 Two Moons

The paper uses the scikit-learn [9] implementation of this dataset. We follow it to set the noise level to 0.1 and generate 1500 points for the training set and 200 points for the testing set. Other settings are kept the same with the paper. In figure 2, we compare the visualization reported in the paper and reproduced by us.

In Figure 2, yellow indicates certainty while blue indicates uncertainty. Overall, both of the visualization shows high uncertainty on the data distribution and demonstrate uncertainty away from the distribution and areas near the decision boundary. However, our result is worse in the upper right boundary of the image, because there are some out of distribution data with high certainty. We adjusted the parameter range but it didn't get any better, and it's worth further investigation.



(a) DUQ reproduced by us (b) DUQ shown in paper

Figure 2: Two Moons Experiment

3.1.2 Fashion MNIST vs MNIST

The purpose of this experiment is to assess the quality of the uncertainty estimation by looking at how well they can separate the test set of FashionMNIST [14] from the test set of MNIST [5]. They train DUQ on FashionMNIST and expect the FasionMNIST test set with low uncertainty and MNIST test set with high uncertainty because the model has nevel seen MNIST when training.

The authors quantify the uncertainty with 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.

For experiment setting, we reproduce the result on Google Colab. There are two important hyper parameters to be decided: the length scale σ and the gradient penalty weight λ. We follow the paper to set the length scale σ to 0.1 and the gradient penalty weight λ to 0.05. In Table 1, we compare our results with those in the paper and see that our values are very close to those in the original paper.

	Acc (FM)	AUROC(NM)	AUROC(N)
OUR RESULT	92.8%	0.97	0.95
IN PAPER	92.4%±.2	0.946±.018	0.955 ± .007

Table 1: Comparison of our results of FashionMNIST vs MNIST with those in the paper. FM stands for FashionMNIST, NM for NotMNIST, and M for MNIST.

3.1.3 CIFAR-10 vs SVHN

In this section, a more difficult dataset pair is considered: CIFAR-10 [4] with SVHN [7] as OoD set. CIFAR-10 is a more difficult dataset for out of distribution detection because there is a significant amount of data noise and the training set is small compared to its complexity.

For this task, ResNet-18 [2] is used as feature extractor, specifically the version provided by PyTorch [8]. But the paper makes minor modifications including using 64 filters in the first convolution layer and skipping the first pooling operation and last linear layer.

For experiment setting, we still follow the paper. We set the length scale to 0.1 and the gradient penalty λ to 0 and 0.05. We train the model 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. Results are shown in Table 2, we can see that our results are still comparable to those in the paper.

	λ	Acc	AUROC
OUR RESULT	0	0.944	0.881
IN PAPER	0	0.942 ± 0.2	0.861 ± 0.032
OUR RESULT	0.5	0.940	0.917
IN PAPER	0.5	0.932 ± 0.4	0.927 ± 0.013

Table 2: Comparison of our results of CIFAR-10 vs SVHN with those in the paper.

3.2 Exploration

3.2.1 SVHN vs LFWPeople

We look at the SVHN dataset, with LFWpeople as OOD set, which contains 13233 images of 5749 people. We keep the framework ResNet-18 as feature extractor with modifications in the article. To use the trained model, the input size must be equal for two datasets, which is not the case for SVHN (3*32*32) and LFWpeople (3*250*250). We have resized the input of LFWpeople into 3*32*32.

3.2.2 Grid Search and Early Stopping

We tune 3 hyperparameters by grid search: learning rate, λ for gradient penalty, and the length scale σ of the kernel distance. The score function is the accuracy on the validation set. To do grid search, we use Optuna package, which is a universal tool for optimizing parameters, and fits well with the Ignite package for Pytorch.

We made 20 random trials over the space of hyperparameters: lr chosen loguniformly in $[0.001, 0.2]$, σ chosen uniformly in $[0.1, 1]$, and λ chosen uniformly in $[0, 1]$. For every trial, we run 50 epochs maximum with batch size 128, and add an early stopping with patience of 4 epochs. Figure 3 shows the convergence of accuracy with the maximal value of trials. Also, we impose a pruning criterion that those trials with the objective value smaller than the best one in the beginning (4 epochs in our case) would be pruned. As is shown in Figure 4, all trials after the 2nd are pruned because their 4 first epochs have relatively poor accuracy. This may suggest that the pruning is decided too early to see the more long-term evolution of accuracy.

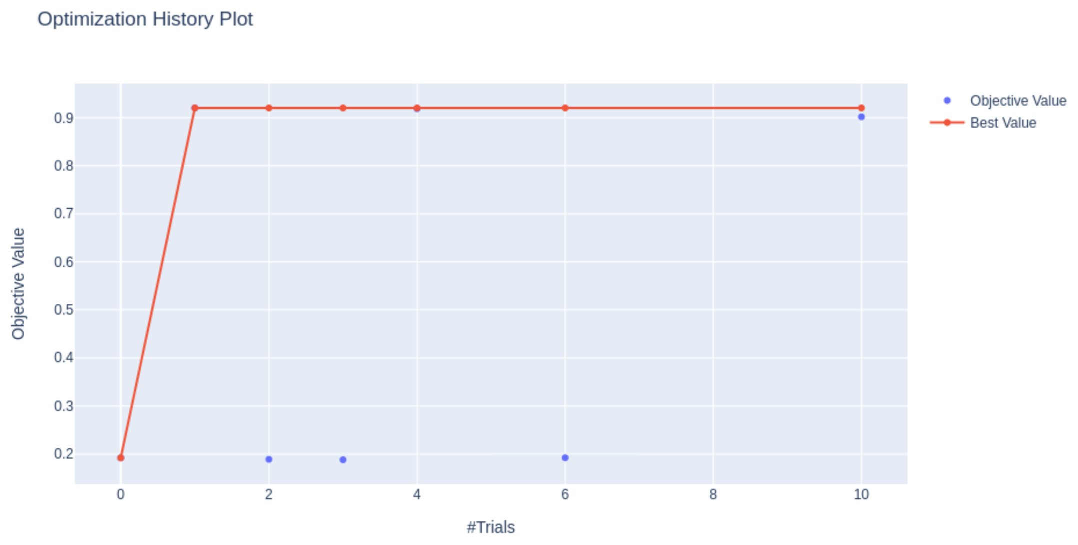


Figure 3: Convergence of accurary with respect to the choice of paramaters. Every value is the maximum of all previous trials.

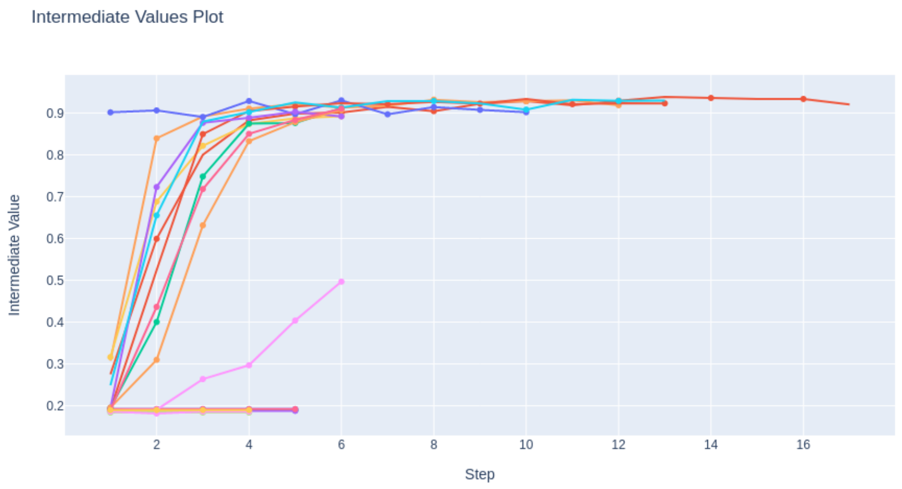


Figure 4: The intermediate score (accuracy) for 20 trials hyperparameters. The best trial is achieved at the trial 1, with lr= 0.02385234757844707, λ= 0.15601864044243652, σ =0.2403950683025824

.

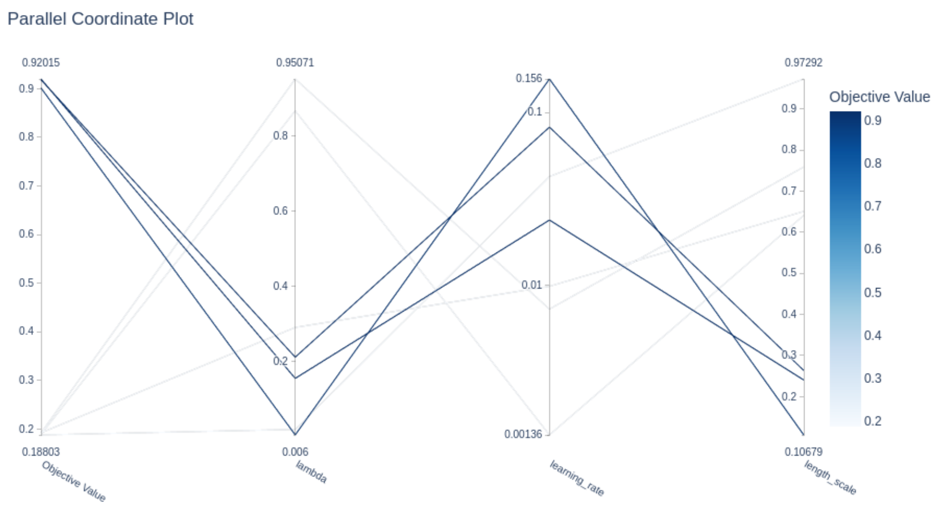


Figure 5: The parallel coordinate plot for 20 trials. Deep color corresponds to better accuracy.

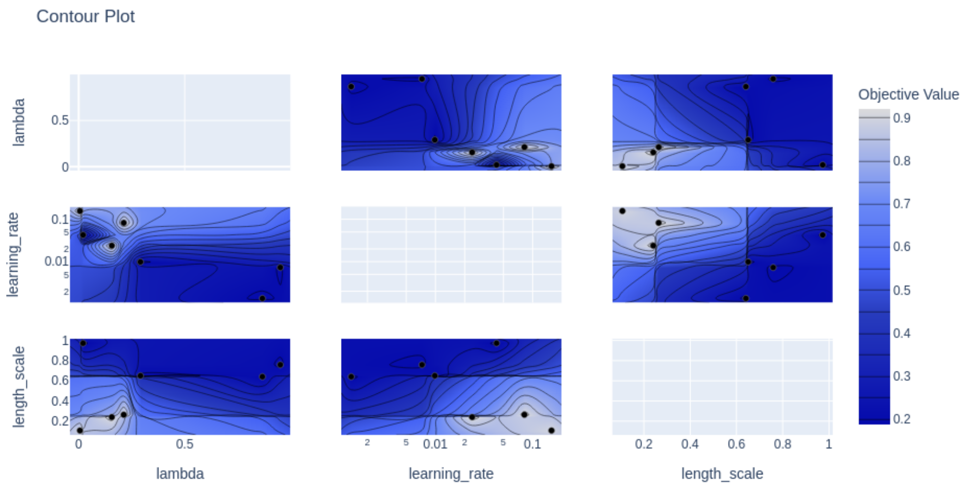


Figure 6: 2d level plot of objective values with respect to every two parameters of three.

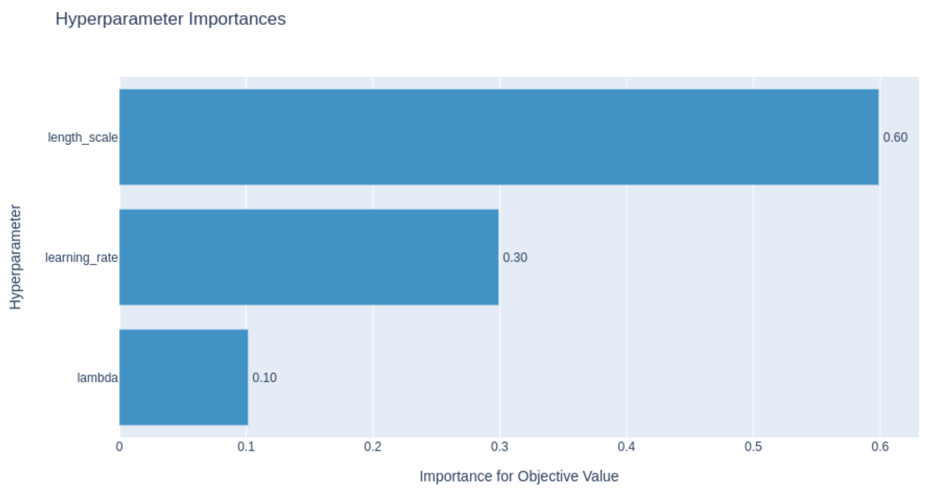


Figure 7: The proportion of importance of three hyperparameters in the training.

3.2.3 Comparing Different Methods

DUQ: After grid search, our optimal choice of hyperparameters is lr= 0.02385, λ= 0.156, σ =0.24. we obtain an accuracy of 94.2% $\pm$ 0.5 on validation set. The accuracy on test set of SVHN is 95.6%, with AUROC 0.96.
DUE: we compare DUQ with the latest work DUE [11] from the same group. In this work, they use direct spectral normalization and residual connections, as they find it to be more stable than a direct gradient penalty and significantly more computationally efficient than reversible models. They also train their model on CIFAR-10 and test it on SVHN. The accuracy on the test set of SVHN is 95.6% $\pm$ 0.04 with AUROC 0.958 $\pm$ 0.005.

References

[1] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. *Advances in neural information processing systems 30* (2017).

[2] 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* (2016), pp. 770–778.

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

[4] Krizhevsky, A., Nair, V., and Hinton, G. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html 55*, 5 (2014).

[5] LeCun, Y. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/* (1998).

[6] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324.

[7] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. The street view house numbers (svhn) dataset. Tech. rep., Technical report, Accessed 2016-08-01.[Online], 2018.

[8] 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.

[9] 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. *the Journal of machine Learning research 12* (2011), 2825–2830.

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

[11] van Amersfoort, J., Smith, L., Jesson, A., Key, O., and Gal, Y. On feature collapse and deep kernel learning for single forward pass uncertainty. *arXiv preprint arXiv:2102.11409* (2021).

[12] Van Amersfoort, J., Smith, L., Teh, Y. W., and Gal, Y. Uncertainty estimation using a single deep deterministic neural network. In *International conference on machine learning* (2020), PMLR, pp. 9690–9700.

[13] Van Den Oord, A., Vinyals, O., et al. Neural discrete representation learning. *Advances in neural information processing systems 30* (2017).

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