Evaluation of Uncertainty Quantification in Deep Learning

Niclas Ståhl(✉), Göran Falkman, Alexander Karlsson, and Gunnar Mathiason

School of Informatics, University of Skövde, Högskolevägen 28, 54145 Skövde, Sweden
niclas.stahl@his.se

Abstract. Artificial intelligence (AI) is nowadays included into an increasing number of critical systems. Inclusion of AI in such systems may, however, pose a risk, since it is, still, infeasible to build AI systems that know how to function well in situations that differ greatly from what the AI has seen before. Therefore, it is crucial that future AI systems have the ability to not only function well in known domains, but also understand and show when they are uncertain when facing something unknown. In this paper, we evaluate four different methods that have been proposed to correctly quantifying uncertainty when the AI model is faced with new samples. We investigate the behaviour of these models when they are applied to samples far from what these models have seen before, and if they correctly attribute those samples with high uncertainty. We also examine if incorrectly classified samples are attributed with an higher uncertainty than correctly classified samples. The major finding from this simple experiment is, surprisingly, that the evaluated methods capture the uncertainty differently and the correlation between the quantified uncertainty of the models is low. This inconsistency is something that needs to be further understood and solved before AI can be used in critical applications in a trustworthy and safe manner.

1 Introduction

Much of the great progress of AI in the last years is due to the development of *deep learning (DL)* [14]. However, one big problem with DL methods is that they are considered to be "black box" methods, which are difficult to interpret and understand. This becomes problematic when DL algorithms are taking more and more critical decisions, impacting the daily life of people and no explanation is given for why a certain decision is taken. There are some researchers, for example Samek et al. [21] and Montavon et al. [18], that currently address this problem and try to make DL models interpretable. This problem is far from solved and it is an important research direction since it is likely that many critical decisions taken by AI based algorithms in the near future will be in consensus with a human [8]. Examples of such decisions would, for example, be those of an autonomous car with a human driver and those of a doctor using an

M.-J. Lesot et al. (Eds.): IPMU 2020, CCIS 1237, pp. 556–568, 2020.
https://doi.org/10.1007/978-3-030-50146-4_41

image to determine if a patient has skin cancer or not [2,4,22]. In these cases, the AI will support and enhance the human decision maker. Here, the human can act in contradiction to what is suggested by the AI, and, thus, prevent erroneous decisions taken by the AI. In both of the named cases, such erroneous decisions could potentially cause a lot of human suffering and even fatalities. However, there are problems where the AI cannot be supervised by a human and, consequently, the AI itself needs to be able to determine when there is a risk of an incorrect decision.

While wrongly taken decisions can be decreased by better models and more and better training data, it is infeasible to cover all possible situations for all but the most trivial problems. Consequently, a system built with an ML model will always encounter situations that differ from all the previous samples used for training. In order to be trustworthy, in this case, it is crucial that the model shows that it encounters an unknown situation where it is forced to extrapolate its knowledge and emphasises that its outcome, therefore, is uncertain [12]. However, as pointed out by Gal and Ghahramani [3], Richter and Roy [20] and Lakshminarayanan et al. [13], it is a challenging and still open problem to quantify the uncertainty of deep learning models. Hendrycks and Gimpel [5] do, for example, show that deep learning models that use the softmax activation function in the last layer are bad at estimating prediction uncertainty and often produce overconfident predictions. It is not difficult to imagine that such overconfident predictions can lead to catastrophic outcomes in some of the previously mentioned cases, such as in the medical domain. Therefore, it is an important research direction to find methods that allow for the quantification of uncertainty in the provided predictions. In this paper, we do not propose such an approach, but do instead evaluate existing models that have been proposed to solve this problem. This is done in order to further understand their limitations and to highlight the differences that arise when different models are selected.

When quantifying the uncertainty, it is essential that methods consider both the epistemic uncertainty (the uncertainty that arises due to lack of observed data) and the aleatory uncertainty (the uncertainty that arises due to underlying random variations within the collected data) [7]. But, it is also important to differentiate between these two causes of uncertainty. In the latter case, there is an observed variation among the samples and, hence, the uncertainty can be well quantified and all risks can be assessed. It is therefore possible to take a well-informed decision, knowing the uncertainty. This is not the case for epistemic uncertainty, where the uncertainty arises due to lack of data and, hence, the model is forced to extrapolate its knowledge. When the model extrapolates it takes an uniformed decision, which can be far from optimal.

To further highlight this problem, this paper examine how well current methods for the quantification of uncertainty manage to show the uncertainty that arises from out of the distribution samples. Two experiments are therefore conducted. In the first experiment, deep learning models that has been proposed to support quantification of predictive uncertainty and that can be used for classification of data are evaluated. These models are: a *deep neural network* with

a *softmax* output layer, an *ensemble* of deep neural networks [13] and a *deep Bayesian neural network* [3], where two separate ways to quantify the uncertainty are used for the softmax model. The first treats the output as a probability while the second method considers the gradient information. The result from these models are compared to another deep learning approach for the detection of out of distribution samples, namely an *autoencoder*. Of these methods, there are two, the Bayesian neural network and the ensemble of neural networks, that are able to disentangle the epistemic and the aleatory uncertainties. A second experiment is therefore conducted, with these two methods, in order to see if the results can be further improved when the uncertainty is split into an aleatoric and an epistemic part.

With these experiments, we show that there is a clear difference in how the investigated methods quantify the uncertainty and what samples they considered to be uncertain. The correlations between the quantified uncertainty of the different models are also very low, showing that there is an inconsistency in the uncertainty quantification. Thus, there is a need for further study of uncertainty in deep learning methods before these can be applied in real world applications in an absolutely safe way.

2 Method

The different models and their setups are first described in this section. Since the targets of the different models differ, there is a need to quantify the uncertainty of these models differently. The second part of this section will therefore describe different ways to quantify the uncertainty. In this section, the motivation behind the selection of how to quantify the uncertainty in the experiments is also given. The last part of this section then describes the experimental setup for all experiments.

2.1 Models

Different deep learning models for classification and uncertainty quantification are used in the conducted experiments. They are all described below, together with the corresponding architecture and parameter settings that are used in the experiments. How the uncertainty is quantified is described in Sect. 2.2–2.3.

Softmax Deep Neural Network. The softmax function is often used in neural networks to fuzzily assign a sample to a given class [1]. Thus, the softmax will give the proportional belief of how much a sample belongs to a given class. This is often used as an approximation of the probability for how likely it is that a sample belongs to a given class. The softmax function is given by:

$$p(y = z|x, \omega) = \frac{f_z^\omega(x)}{\sum_{z' \in Z} f_{z'}^\omega(x)}, \tag{1}$$

where z is the given output class, which belongs to the set of all possible outcomes, Z. X is the input sample and $f_z^\omega(x)$ is an arbitrary function, parameterised by ω, giving the support that x belongs to class z. Equation (1) allows us to find the probability distribution of all possible outcome classes. This distribution can be used to quantify the uncertainty, as described in Sect. 2.2.

Recent studies, for example by Oberdiek et al. [19] and Vasudevan et al. [26], suggest that the uncertainty of the model is reflected by the stability of the model, where the stability can be measured by the gradients of the parameters. Hence, the stability of the model is given by:

$$\nabla_\omega \mathcal{L} = \nabla_\omega l\left(\hat{y}_i, f^\omega(x_i)\right), \tag{2}$$

where $\mathcal{L}$ is the loss of the model given by an arbitrary loss function l, $\hat{y}_i$ is the predicted class for the i:th sample and $f^\omega(x_i)$ is the prediction from the model that is parameterised by ω. We follow the same experimental setup as Oberdiek et al. [19] and use the negative log-likelihood for the predicted class as the loss function. In this case, Eq. 2 can be written as

$$\nabla_\omega \mathcal{L} = \nabla_\omega - \log\left(p(y_i = \hat{y}_i | x_i, \omega)\right). \tag{3}$$

Furthermore, we use a deep neural network as the underlying model, that is, $p(y_i = \hat{y}_i | x_i, \omega)$ in Eq. 3 is given by Eq. 1.

Bayesian Neural Network. We consider Bayesian neural networks to be neural networks that have a posterior distribution of weights instead of a single point estimate. The same definition is, for example, used by Gal and Ghahramani [3]. Hence, the training of a Bayesian neural network consists of finding a good estimate to the probability distribution $p(\omega|X, Y)$, where ω is the network weights and X is the set of inputs and Y is the set of outputs. It is however, unfeasible to find the exact solution to $p(\omega|X, Y)$ and, hence, an estimate must be used instead. In this paper we approximate $p(\omega|X, Y)$ with a network that uses dropout [25] during both the training and the testing phase. This is the same approach as Gal and Ghahramani [3]. With an estimated posterior distribution, $p(\omega|X, Y)$, multiple network weights can be sampled. Hence, many likely network weights can be used for predictions, which would allow for a smaller risk of overfitting and a greater diversity in the output. The final classification of a sample x of a Bayesian neural network is given by:

$$p(y|x, \Omega) = \frac{1}{M} \sum_{i=0}^{M} f^{\omega_i}(x), \tag{4}$$

where M is the total number of samples, f is a neural network parameterized by ω_i that is the i:th sample from the posterior distribution, Ω, of network weights.

Ensemble of Neural Networks. Ensemble methods are learning algorithms that consist of a set of models. Each model makes its own prediction independently of the other models in the ensemble. The final prediction is then derived

from the composition of all models in the ensemble. We use an ensemble of neural networks, such as the one presented by Maqsood et al. [17]. Such ensembles have been shown to be good at quantifying the predictive uncertainty, as shown by Lakshminarayanan et al. [13]. The classification of a sample x by the ensemble is the average prediction over all classifiers. Hence, the prediction, y, is given by:

$$p(y|x, w_0, \ldots, w_M) = \frac{1}{M} \sum_{i=0}^{M} f^{w_i}(x), \tag{5}$$

where M is the number of networks in the ensemble and w_i is the parametrisation of the i:th classifier, f, in the ensemble. Note the similarity with the deep Bayesian neural network as given in Eq. 4.

Autoencoder. An autoencoder is a neural network that has the same number of input neurons as output neurons. This network consists of two parts: an encoder that compresses the input to a compressed representation of the sample, with an as low loss of information as possible, and a decoding part that decompresses the compressed representation to the original representation [6]. These parts are jointly trained and, hence, the encoder is forced to learn an encoding scheme that the decoder can decompress. Therefore, these two models will learn how to collaborate, but only on data that is similar to the data they see during training. This means that the encoder would not be able to encode novel data in such a way that it can be reconstructed by the decoder. This can be exploited to detect how much a new sample diverges from an initial distribution. Thus, the uncertainty of a prediction in a predictive model may be quantified by the reconstruction error of a sample given to the autoencoder. This is, for example, done by Leibig et al. [16], and the same approach will be used in our experiments.

2.2 Uncertainty Quantification

There are multiple ways that uncertainty can be quantified. Kendall and Gal [9], for example, quantifies the uncertainty as the variance in the predictions. We follow the same approach as Lakshminarayanan et al. [13] and use the Shannon entropy [23]:

$$H(y, X) = -\sum_{i} p(y = i|X) \log p(y = i|X) \tag{6}$$

as a measure of the uncertainty in the predictions of the models specified in Sect. 2.1. This design choice is mainly selected to enable the comparison between the uncertainties of the softmax network and the other models, since the softmax network does not have any variation in its predictions.

It is, however, not possible to use this uncertainty metric for the experiments that consider the gradient information. In this case, we use the approach suggested by Oberdiek et al. [19], namely to use the Euclidean norm of the gradients. The quantified uncertainty of a prediction is then given by $||\nabla_w L||_2$, where $\nabla_w L$ is described in Eq. 3.

It is also impossible to measure the entropy of predictions in the autoencoder (described in Sect. 2.1) since the autoencoder does not provide any predictions. Instead, we quantify the uncertainty by measuring the Euclidean distance between the original sample and the encoded and decoded sample:

$$||X, \hat{X}|| = \sqrt{\sum_{i=1}^{n} (x_i - \hat{x_i})^2}, \tag{7}$$

here x_i is the original value of the i:th feature of x and $\hat{x_i}$ is the reconstructed value for the same feature. Hence, Eq. 7 measures how well the autoencoder manages to encode the vector and then decode the sample vector X of length n.

2.3 Heteroscedastic Aleatoric Uncertainty

The aleatoric uncertainty can be divided into two sub-categories: *heteroscedastic* and *homoscedastic* uncertainty. Heteroscedastic uncertainty assumes that the aleatoric uncertainty is data dependent and, thus, that the uncertainty varies over different inputs. Hence, models that can capture the heteroscedastic uncertainty are useful when the uncertainty is greater in some areas of the input space. Such is the case in the MNIST dataset [15], where some of the digits are badly written and the output class is uncertain.

The heteroscedastic uncertainty in the models will be treated in the same way by Kendall and Gal [9] and furthermore described as in Kendall et al. [10]. Here, the expected variance of the noise in the output is modelled by the noise parameter σ. This parameter will be dependent on the input and the models will learn how to predict it, given some particular input. In the presented multiclass setting, the loss with included heteroscedastic uncertainty can be approximated with:

$$\mathcal{L}(\omega, x, y) = \frac{1}{\sigma^2}\mathcal{L}_{ce}(softmax(f^{\omega}(x)), y, \omega) + \log(\sigma), \tag{8}$$

where $\mathcal{L}_{ce}$ is the categorical cross entropy loss and f^{ω} is the model, parameterized by ω and predicting logits to the softmax function. The predicted logits are assumed to be drawn from a Gaussian distribution with a variance of σ, where σ is dependent on the input x. This loss function is used in the second part of the experiments, where it is examined how well the Bayesian neural network and the ensemble of neural networks can capture and separate epistemic and aleatoric uncertainty.

2.4 Experiment Setup

All previously described models are trained on the MNIST dataset, which is a dataset that contains 70,000 samples of hand-written digits [15]. The predefined and commonly used training and test split, which uses 10,000 samples in the test set, is used in our experiments as well. A randomly selected validation set, consisting of 10% of the training set, is also used to prevent overfitting of the

models when training. All models are then evaluated on the MNIST test set and the uncertainty of their predictions is quantified and split a set of correctly classified samples and a set of incorrectly classified samples. The main hypothesis is that the uncertainty would be much greater in the set of incorrectly classified samples. These models are then applied to the manually cleaned notMNIST[1] set that consist of 19,000 characters, set in different fonts. The objective of the models is to detect that these samples are very different from the original training data and attribute them with a high uncertainty. Since the autoencoder is not used to perform any classification, we decided to use a feed forward neural network that uses the latent encoding to predict the class of the output.

Parameter Settings. Both the Bayesian neural network and the softmax deep neural network have two layers with 800 neurons each. This is the same network architecture used by Simard et al. [24]. The inference in the Bayesian neural network is conducted in the same way as described by Gal and Ghahramani [3], with 60% dropout rate. The networks in the ensemble are each trained on bootstrap samples, which have 60% less samples than the original dataset. Since the amount of data is reduced, we also reduce the number of neurons in each layer to 40 % of the size of the Bayesian neural network. Hence, the ensemble will consist of 50 networks where each network has two layers of 320 neurons.

The autoencoder used in the experiment has 7 layers with the following number of neurons: 1000, 250, 50, 10, 50, 250, 1000. This is the same setup as used by Wang et al. [27]. All models are trained using the ADAM optimisation algorithm [11] with the commonly used learning rate of 0.001, to minimise the binary cross entropy error between the model predictions and the targeted classes.

3 Experimental Results

All presented deep learning methods are trained on the MNIST dataset and then evaluated on a smaller test set from MNIST as well as the notMNIST dataset. Some examples of samples from these datasets are shown in Fig. 1. The accuracy of all models are approximately the same and in line with what is expected from a two layered neural network model and the MNIST dataset [24]. The Bayesian neural network is, for example, the best performing model with an error rate of 1.3%, while softmax is the worst with an error rate of 1.6%.

Unlike the accuracy, there is a great difference in how the uncertainty of the models are quantified. This can be seen in Fig. 2, where the distributions of the quantified uncertainties are shown. The distributions over the uncertainty is split into three different distributions: the distribution over the uncertainty for correctly classified samples, the distribution over the uncertainty for incorrectly classified samples and the distribution over the uncertainty for samples from the notMNIST dataset. If a model acts as desirable, it should separate these three classes and thus, that the distributions in Fig. 2 are disjoint. This optimal case

[1] Available at: http://yaroslavvb.blogspot.co.uk/2011/09/notmnist-dataset.html.

corresponds to that the model correctly detects digits that are easy to classify and attribute them with a low uncertainty. Furthermore, the model detects odd looking digits and correctly attribute them with medium uncertainty, since the classification may be erroneous. Also, when a sample that is clearly not a number, the model should attribute it to an even higher uncertainty, since the prediction is extrapolated far from what the model knows. However, this implies that correct and possibly incorrect predictions can be identified by the quantified uncertainty and real digits can, thus, be filtered from non digits. This is not the case, the distributions do indeed overlap, as can be seen by studying the 95% quantiles for the distributions in Fig. 2.

The consensus of the uncertainties of the models are measured by their Pearson correlation (see Fig. 4). The measurements show a strong correlation between gradient information and the softmax predictions, but no strong correlation besides that. The quantified uncertainty of the softmax neural network is even negatively correlated to the autoencoder.

The uncertainty is furthermore divided into an epistemic and an aleatoric part, as shown in Fig. 3. The expected result would be that the notMNIST samples would have much greater epistemic uncertainty than all other samples, while the misclassified MNIST samples would have a greater aleatoric uncertainty. However, this can only partially be observed, since both the notMNIST and the misclassified MNIST samples show a high epistemic uncertainty.

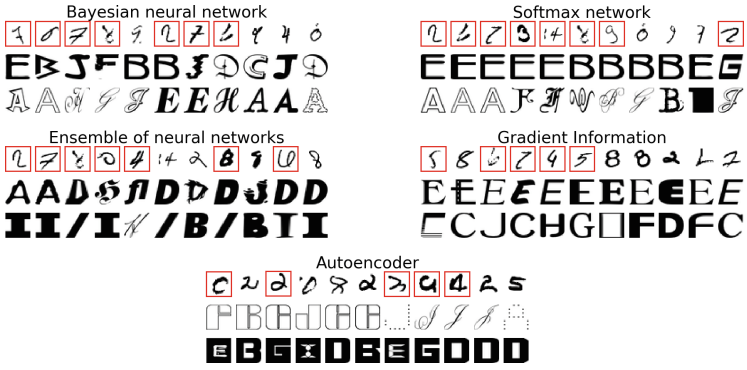

Fig. 1. Examples of the behaviour of the evaluated methods tested on the MNIST and the notMNIST datasets. The first row, for the given method, consists of the eleven most uncertain predictions from the MNIST dataset. Incorrectly classified examples are marked red. The second and the third row show the eleven most certain and uncertain examples from the notMNIST dataset, respectively. (Color figure online)

4 Discussion

The results show that all the evaluated methods quantify the uncertainty differently. The results, furthermore, support the previous observation by Hendrycks

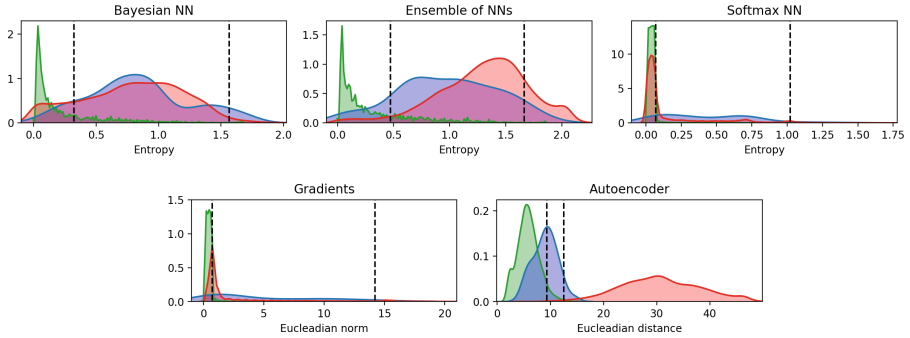

Fig. 2. The distribution of the quantified uncertainty for the different methods. In green is the distribution of correctly classified digits in the MNIST dataset. In blue is the distribution of incorrectly classified digits and in red is the distribution of the quantified uncertainty for samples from the notMNIST dataset. The 95% quantile of the quantified uncertainty of all samples from the MNIST dataset is marked with the dashed line to the left. The right dashed line is the 95% quantile of the quantified uncertainty when only the misclassified samples are considered. (Color figure online)

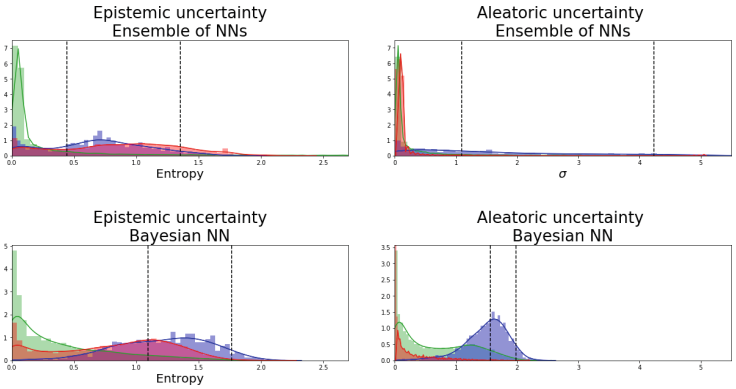

Fig. 3. The distribution of the quantified uncertainty for the different methods, split up into aleatoric and epistemic uncertainty for the Bayesian neural network and the ensemble of neural networks. In green is the distribution of correctly classified digits in the MNIST dataset. In blue, the incorrectly classified digits in MNIST and, in red is the distribution of the quantified uncertainty for samples from the notMNIST dataset. The 95% quantile of the quantified uncertainty of all samples from the MNIST dataset is marked with the dashed line to the left. The right dashed line is the 95% quantile of the quantified uncertainty when only the misclassified samples are considered. (Color figure online)

and Gimpel [5] that deep learning models that only use the softmax activation function to quantify the uncertainty are overconfident when faced with out of the distribution samples. The same holds true when the gradient information of the softmax neural network is used to quantify the uncertainty.

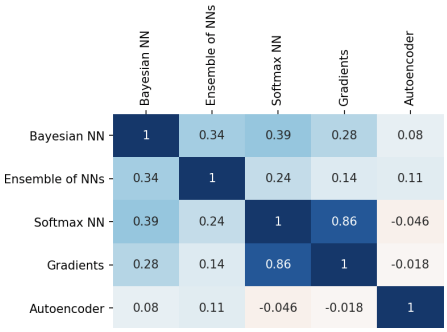

Fig. 4. The Pearson correlation between the quantified uncertainty of all tested methods. Many of the methods are very weakly correlated and the softmax neural network is even negatively correlated to the uncertainty of the autoencoder.

When the results in Fig. 1 were furthered studied, we identified several interesting behaviours of the models. The models that are used for classification extract knowledge about the shape of digits and apply it to the notMNIST data. Both the Bayesian neural network and the ensemble of neural networks do, for example, pick up the curvy shape of a "B" and interpret this as the digit "3" and, hence, the models are certain of the output in these cases. The round shape and the empty middle of the letter "D" being classified as the digit "0" is another example of the extrapolation of features into the new domain. The two methods that are based on a softmax neural network do an even cruder extrapolation and classify everything with a straight horizontal line at a certain height as a "7" (all the first eight samples shown in Fig. 1 for the softmax network and the gradient information is classified as the digit "7").

No model achieved the optimal goal of quantifying the uncertainty in such a way that it separates the three different cases of input: digits that could be classified correctly, digits that could not be classified correctly and characters from the notMNIST set. However, the autoencoder correctly uses the uncertainty quantification to separate all notMNIST samples from the MNIST samples, while the Bayesian neural network and the ensemble of neural networks can correctly separate classified MNIST samples from the other two cases. It can, consequently, be efficient to use an autoencoder as a first filtering step to remove all out of the distribution samples. Another method, such as a Bayesian neural network, can then be used to perform safer classifications, where the uncertainty quantification can be used to identify possibly misclassified samples. There is no downside of such a combination of models, besides the slightly higher computational cost. It is, therefore, an interesting future research question how different models can be combined in order to handle and distinguish between the different cases that may cause these models to be uncertain.

A surprising observation is that the quantified uncertainties of most of the models are weakly correlated. All three models that are used for prediction are, for example, weakly correlated to the autoencoder, which is considered to

capture the initial distribution well. Since we only measure the linear correlation it is difficult to draw any major conclusions from this, but it still gives us some insights into the behaviour of the models. We, therefore, suggest that these methods do not capture the uncertainty that arises due to the extrapolation, but instead finds fuzzy decision boundaries between the different classes and, hence, are able to spot odd looking samples between the different classes. However, this implies that there is no guarantee that predictions on out of the distribution samples will be considered uncertain. This poses a potential risk when using these kind of models in critical real world applications.

The use of an autoencoder is a good way to approximate the distance between a new sample and its closest neighbour in the training set. This is a promising result since the autoencoder is more efficient, when considering the computational complexity, compared to finding the closest neighbour in the training set and calculating the Euclidean distance. The computational complexity of finding the closest neighbour in the training set grows linearly in terms of the cardinality of the training dataset, while the computational complexity of the autoencoder is constant. Hence, it appears that the autoencoder correctly discovers when a model is faced with a sample that is far from what the model has seen before and, hence, forces the model to extrapolate. Thus, an autoencoder could potentially be used to detect when a sample would force a predictive model to extrapolate, if trained with the same data.

Splitting up the uncertainty into an epistemic and an aleatoric part and then use the epistemic uncertainty to detect outliers is not a successful approach in the performed experiments. While we expect the epistemic uncertainty to be much higher for such samples, it is not the case, since both the badly written MNIST digits and the notMNIST samples are attributed with a high epistemic uncertainty. However, the notMNIST samples distinguish themselves from the rest by having a very low aleatoric uncertainty. Hence, the outlier samples distinguish themselves from the rest by having a low aleatoric uncertainty, rather than having a high epistemic uncertainty. The combined epistemic and aleatoric uncertainty can therefore be used to detect the notMNIST samples. The reason why outlier samples are attributed with a low uncertainty can be seen in Eq. 8. Since the models are good at predicting the outcome, the expected cross entropy loss would be rather small. Hence, it is more beneficial for the model to minimise the $\log(\sigma)$ term for new unknown samples than to expect a large cross entropy error.

5 Conclusion and Summary

In this paper, several models for the quantification of uncertainty are evaluated. Even though the experimental setup is rather basic, it is shown that there is no consensus in the uncertainty of the models and that they capture different dimensions of the uncertainty. This problem is likely to persist, and may even be worse, when more advanced models are used or when more complicated problems are tackled. It is shown that the uncertainty quantification of some models

(the Bayesian neural network and the ensemble of neural networks) can be used to distinguish between samples that are easy to classify and those that are difficult. Hence, these models quantify the uncertainties around the hyperplanes separating the different classes. The autoencoder, on the other hand, is good at quantifying the uncertainty that arises due to the extrapolation of points far from the training distribution. The performed experiments show that it can be beneficial to split up the uncertainty into an epistemic and an aleatoric part. However, the notMNIST samples did not differentiate themselves from the rest by having much higher epistemic uncertainty than the other samples, as was expected. Instead, the notMNIST samples stood out by having the combination of a high epistemic uncertainty and a low aleatoric uncertainty. However, none of the models managed to separate the three different cases of samples that were studied, namely correctly classified samples, incorrectly classified samples and samples that are far from the training distribution. On the others hand, as described above, some methods succeeded partially, and managed to separate one of the cases from the other. It can, therefore, be beneficial to use several models in real world applications to capture all uncertainties that may arise, in order to build safer AI systems.

References

1. Bridle, J.S.: Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In: Touretzky, D.S. (ed.) Advances in Neural Information Processing Systems 2, pp. 211–217. Morgan-Kaufmann (1990)
2. Esteva, A., et al.: Dermatologist-level classification of skin cancer with deep neural networks. Nature **542**(7639), 115 (2017)
3. Gal, Y., Ghahramani, Z.: Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In: Balcan, M.F., Weinberger, K.Q. (eds.) Proceedings of The 33rd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 48, pp. 1050–1059. PMLR, New York (2016)
4. Gerdes, J.C., Thornton, S.M.: Implementable ethics for autonomous vehicles. In: Maurer, M., Gerdes, J.C., Lenz, B., Winner, H. (eds.) Autonomes Fahren, pp. 87–102. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-45854-9_5
5. Hendrycks, D., Gimpel, K.: A baseline for detecting misclassified and out-of-distribution examples in neural networks. arXiv preprint arXiv:1610.02136 (2016)
6. Hinton, G.E., Zemel, R.S.: Autoencoders, minimum description length and Helmholtz free energy. In: Cowan, J.D., Tesauro, G., Alspector, J. (eds.) Advances in Neural Information Processing Systems 6, pp. 3–10. Morgan-Kaufmann (1994)
7. Hüllermeier, E., Waegeman, W.: Aleatoric and epistemic uncertainty in machine learning: a tutorial introduction. arXiv preprint arXiv:1910.09457 (2019)
8. Jarrahi, M.H.: Artificial intelligence and the future of work: human-AI symbiosis in organizational decision making. Bus. Horiz. **61**(4), 577–586 (2018)
9. Kendall, A., Gal, Y.: What uncertainties do we need in Bayesian deep learning for computer vision? In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems 30, pp. 5574–5584. Curran Associates, Inc. (2017)

10. Kendall, A., Gal, Y., Cipolla, R.: Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7482–7491 (2018)
11. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
12. Kuleshov, V., Fenner, N., Ermon, S.: Accurate uncertainties for deep learning using calibrated regression. arXiv preprint arXiv:1807.00263 (2018)
13. Lakshminarayanan, B., Pritzel, A., Blundell, C.: Simple and scalable predictive uncertainty estimation using deep ensembles. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems 30, pp. 6402–6413. Curran Associates, Inc. (2017)
14. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
15. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
16. Leibig, C., Allken, V., Ayhan, M.S., Berens, P., Wahl, S.: Leveraging uncertainty information from deep neural networks for disease detection. Sci. Rep.-UK **7**(1), 17816 (2017)
17. Maqsood, I., Khan, M.R., Abraham, A.: An ensemble of neural networks for weather forecasting. Neural Comput. Appl. **13**(2), 112–122 (2004). https://doi.org/10.1007/s00521-004-0413-4
18. Montavon, G., Samek, W., Müller, K.R.: Methods for interpreting and understanding deep neural networks. Digit. Sig. Process. **73**, 1–15 (2018)
19. Oberdiek, P., Rottmann, M., Gottschalk, H.: Classification uncertainty of deep neural networks based on gradient information. In: Pancioni, L., Schwenker, F., Trentin, E. (eds.) ANNPR 2018. LNCS (LNAI), vol. 11081, pp. 113–125. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99978-4_9
20. Richter, C., Roy, N.: Safe visual navigation via deep learning and novelty detection. In: Robotics: Science and Systems Conference. Robotics: Science and Systems Foundation, July 2017
21. Samek, W., Wiegand, T., Müller, K.R.: Explainable artificial intelligence: understanding, visualizing and interpreting deep learning models. arXiv preprint arXiv:1708.08296 (2017)
22. Shalev-Shwartz, S., Shammah, S., Shashua, A.: Safe, multi-agent, reinforcement learning for autonomous driving. arXiv preprint arXiv:1610.03295 (2016)
23. Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. J. **27**(3), 379–423 (1948)
24. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. In: Proceedings of the Seventh International Conference on Document Analysis and Recognition, pp. 958–963 (2003)
25. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
26. Vasudevan, V.T., Sethy, A., Ghias, A.R.: Towards better confidence estimation for neural models. In: ICASSP 2019–2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 7335–7339. IEEE (2019)
27. Wang, Y., Yao, H., Zhao, S.: Dropout: a simple way to prevent neural networks from overfitting. Neurocomputing **184**, 232–242 (2016)