Uncertainty Modeling in Deep Neural Networks for Image Classification

A. Khoshsirat

School of Electrical & Computer Engineering, Shiraz University, Shiraz, Iran

Abstract. Estimating how confident an AI system is in its predictions is important to improve the safety of such systems. Deep neural networks (NNs) are powerful black box predictors that have recently achieved impressive performance on a wide spectrum of tasks. Quantifying predictive uncertainty in NNs is a challenging and yet on-going problem. Although there has been many efforts to equip NNs with tools to estimate uncertainty, many of the previous methods only focus on one of the three sources of uncertainty: Model uncertainty, Data uncertainty or Distributional uncertainty. In this paper we propose a complete framework to capture and detect all of these three types of uncertainty for the task of image classification and demonstrate the efficiency of our method on popular image datasets.

Keywords: Uncertainty · Deep Neural Networks · Confidence.

1 Introduction

Deep neural networks (NNs) have achieved state-of-the-art performance on a wide variety of machine learning tasks and are becoming increasingly popular in domains such as computer vision. DNNs have proven to perform as good or even better than humans in certain scenarios and have come up with phenomenal results in areas like image analysis [1], voice recognition [2] or even game strategies [3]. Due to this ongoing success, Deep Learning methods are being incorporated into all kinds of applications. Research currently goes into using this technology for self-driving cars [4] or for ID checks at passport controls [5]. However, when it comes to critical systems like these, we want to make sure that our model does not make mistakes by any chance. We cannot expect it to be always accurate or one-hundred percent perfect at their task, however, we want it to let us know if it is not certain about a situation. This way, a second check can be performed or the task can be passed to a human specialist.

Despite impressive accuracy in supervised learning benchmarks, NNs are poor at quantifying predictive uncertainty, and tend to produce overconfident predictions. Classic Deep Learning networks do not incorporate uncertainty information but only return a point prediction. Classifiers failing to indicate when they are likely mistaken can limit their adoption or cause serious accidents. For classification problems, we sometimes consider a distribution over potential classes in order to express confidence for each class. However, when faced with an unclassifiable input, i.e. an input that does not correspond to either class, results can be deceiving [6]. For example, a medical diagnosis model may consistently classify with high confidence, even while it should flag difficult examples for human intervention. Estimating uncertainty in a model's predictions is important, as it enables, for example, the safety of an AI system to be increased by acting on the model's prediction in an informed manner. This is crucial to applications where the cost of an error is high, such as in autonomous vehicle control and medical, financial and legal fields.

Several research has been done in the field of uncertainty in machine learning. Different approaches have been published which enhance existing technologies by adding capabilities to networks which enable them to not only give predictions or classification solutions but to also incorporate a confidence measure. For modeling the uncertainty in these systems, one must consider different aspects of predictive uncertainty, which results from three separate factors - model uncertainty, data uncertainty and distributional uncertainty. Model uncertainty, or epistemic uncertainty , measures the uncertainty in estimating the model parameters given the training data - this measures how well the model is matched to the data. Data uncertainty, or aleatoric uncertainty is uncertainty which arises from the natural complexity of the data, such as class overlap and label noise. Distributional uncertainty arises due to mismatch between the training and test distributions (also called dataset shift). Taking all these three forms of uncertainties into account and being able to distinguish between them for high risk applications, can help us choose what action to take to avoid fatal errors in the predictions of the system.

Although recent studies on deep neural networks have resulted in good detection of one form of uncertainty, none have fully anticipated all types of the known uncertainties. We've proposed a complete framework for deep neural networks to take these three kinds of uncertainties into account for the task of image classification.

2 Related Works

Although there has been a lot of recent interest in adapting Deep Neural Networks to capture uncertainty, most of the methods focus on just one or two of the mentioned uncertainties and don't put any efforts into specification of the sources. One group of methods called Bayesian Neural Networks [7] concentrate on model uncertainty by specifying a prior distribution upon the parameters of a NN and then, given the training data, the posterior distribution over the parameters is computed.

In a traditional generic neural networks we have fixed weights and biases that determine how an input is transformed into an output. In a Bayesian neural network, all weights and biases have a probability distribution attached to them. When the training is finished and the distributions are learned, the test data is given to the network more than one time. To classify an image, you do multiple runs (forward passes) of the network, each time with a new set of sampled weights and biases. Instead of a single set of output values what you get is multiple sets, one for each of the multiple runs. The set of output values represent a probability distribution on output values and hence you can find out confidence and uncertainty in each of the outputs. The variance of the distribution is then calculated for quantifying the uncertainty of the network for a specific test data. The current limitation of doing this work in large scale or real time production environments is posterior computation. Variational inference techniques and/or efficient sampling methods to obtain posterior are computational demanding. This problem has lead the researchers to work on finding other methods with less computational and time cost [8].

One recent development on estimating uncertainty, has been the technique of Monte-Carlo Dropout [9]. Dropout is usually applied for deep neural networks in order to avoid over-fitting. Also originally, dropout is only used during training time and dismissed at prediction time. Monte-Carlo Dropout , however, uses this method to introduce randomness to the prediction process. This way, the network can be evaluated for the same input multiple times, resulting in a set of predictions that can be used to estimate a distribution in the target domain or to determine statistical values. At prediction time, this technique estimates predictive uncertainty using an ensemble of multiple stochastic forward passes and computing the mean and spread of the ensemble. The mean prediction can be estimated by averaging the forward passes while the uncertainty can be estimated in terms of the empirical variance. This method has been successfully applied to tasks in computer vision [10].

Another method that has been proposed recently, uses an ensemble of deep neural networks for uncertainty estimation. Deep Ensembles [11] take a different approach than previous methods: it assumes the data to have a given parametrized distribution where the parameters depend on the input. Finding these parameters is the aim of the training process, i.e. the prediction network will not output a single value but instead will output the distributional parameters for the given input. In other words, instead of training one network, an ensemble of identical networks are trained with parameter sharing and data set splitting. At prediction time, the individual distributions are then averaged resulting in the final estimate. The outputs distribution (its variance) can then be used for estimating the uncertainty of the system. This method has been tested on regression problems.

Although these methods have resulted in good estimates for predictive uncertainty, they can not specify the source of it. For instance, in a classification problem, a test data with high variance of outputs distribution in a Monte-Carlo Dropout network, can either be interpreted as, an intrinsically hard to classify sample of the known data distribution, that is near a decision boundary (data uncertainty), or an out-of-distribution sample which has not been introduced to the network in training time (distributional uncertainty).

On the other end of the spectrum, some methods have been proposed to specifically estimate one of the three kinds of uncertainty for a test data point in DNNs. A recent study, uses deep autoencoders for detecting out-of-distribution samples at test time on image data [12]. Autoencoders optimize the compression of input data to a latent space of a smaller dimension and attempt to accurately reconstruct the original input using the features that has been learned in the latent space. Since the latent vector is optimized to capture the salient features from the inlier class only, it is assumed that images of objects from outside of the training classes cannot effectively be compressed and reconstructed. The study proposes using deep autoencoders and then a clustering routine to detect OOD samples at test time. Although it hasn't directly stated, a form of distributional uncertainty is used for this detection which has been produced by computing the reconstruction error.

In another study, misclassified examples detection is done by using the softmax probabilities for classification deep neural networks [13]. Correctly classified examples tend to have greater maximum softmax probabilities than erroneously classified examples, allowing for their detection. This also can be considered a use of data uncertainty, since the softmax layer focuses on the inlier classes and their boundaries with each other.

3 Proposed Method

As mentioned in the previous section, different methods have been proposed to capture different types of uncertainty in DNNs. In this paper we propose a simple and functional framework which combines the previously proposed methods and also new methodologies to better capture the different predictive uncertainties for the task of image classification. This framework includes an ensemble of DNNs for model uncertainty, a supervised reconstruction auto-encoder for distributional uncertainty and using the softplus activation function in the last layer of the DNN for capturing the data uncertainties, working side by side to model the three types of uncertainties. Finally our framework is evaluated on detection of misclassified or out-of-distribution samples for classification of various image datasets.

In the sections below we describe our proposed methods for any of the three types of the mentioned uncertainties.

3.1 Model Uncertainty

Model or epistemic uncertainty is about how well a used model is matched to our training data for the problem at hand. Although Bayesian methods can model this uncertainty for NNs, Their computational cost and complexity can somehow affect the system performance, therefore using non-Bayesian methods are encouraged.

Here we use an ensemble of deep neural networks with the same architecture (suitable for the classification task) and random initialization to present the parameter uncertainties of the model. This method can show how uncertain we are for a deep neural network model on a specific data set.

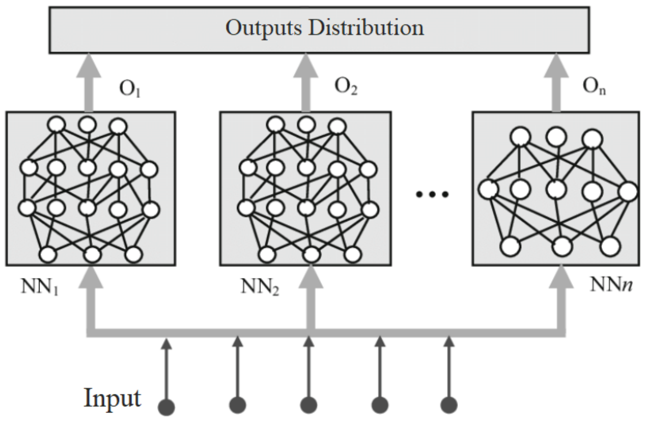


Fig. 1: Ensemble of DNNs, the proposed architecture for estimating model uncertainty.

Due to different initialization of the networks' parameters, an unsuitable network could result in more dissimilar output results in the ensemble. Hence the spread of the distribution on the outputs, can be used to estimate how well the model is fitting for the classification task. This method is simple and parallelizable compared to Bayesian methods and can very well be applied to different types of NNs for this task. In this scenario, we've used ten Convolutional Neural Networks (CNN) having the same architecture, but different initialization (random) for image classification.

3.2 Data Uncertainty

Data uncertainty or aleatoric uncertainty is the complexity of the data itself. Modeling and detecting this type of uncertainty can help us predict whether a test sample is classified correctly and how confident we are for any given prediction.

As mentioned in the previous section, some works have been done to use softmax probabilities for somehow capturing the data uncertainty specifically and detecting the probable misclassified samples. However, softmax activation func. limits the probabilities of the last layer of the network to always sum up to 1. Therefore, during training, the network is bound to decrease the other classes' probabilities in order to increase the maximum probable class for a specific training data. This limitation can lead to missing the beneficial information about the data for capturing the accurate data uncertainty.

To address this issue, instead of softmax, we use the Softplus activation function in the last layer of a DNN. The softplus func. has no such limitations on the last layer's neurons and can better show the data points probability information. For computing the data uncertainty metric at test time, We take the softplus probabilities and compute two terms. First we subtract the maximum prob. by the second maximum prob. for a test data point. This value is larger for the samples with less data uncertainty; confident network predictions. Secondly, we sum up all the other classes' probabilities. This value shows the spread of the class probabilities which can be interpreted as the uncertainty of the networks decision to classify a test data point to a specific class. Finally we divide these two terms to compute a suitable metric for networks data uncertainty for predictions. Using proper thresholds, we can very well detect the hard to classify or probably will-be-misclassified samples. The final formula for computing data uncertainty is shown below:

$$Data\ uncertainty = \frac{\sum_{i=3}^{n} p_{sp}(c_i)}{p_{sp}(c_1) - p_{sp}(c_2)} \tag{1}$$

In this equation, p_{sp} is the probability of softplus layer and the class probabilities are sorted from the maximum to minimum (c_1,c_2,...,c_n) for a specific data point.

3.3 Distributional Uncertainty

This uncertainty arises when the samples given at test time are not from the distribution of the training data. For detecting this, we propose using a supervised reconstruction autoencoder. Reconstruction autoencoders are NNs that compress the data to a bottleneck layer (encoder) and then reconstruct the input data from that layer (decoder). This would force the NN to learn the distribution of the input data in the bottleneck layer. As previously mentioned, this tool can be used to detect out of distribution samples at test time.

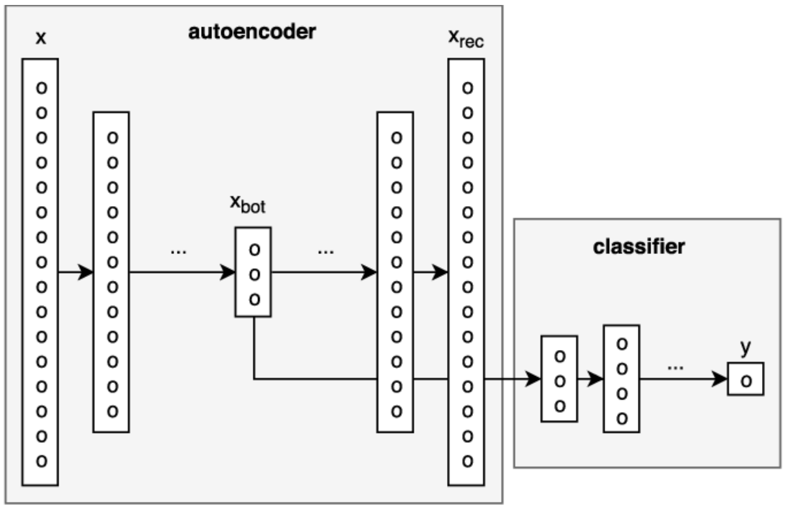


Fig. 2: Architecture used for supervised reconstruction autoencoder. This network is trained by minimizing both the classification and reconstruction loss of the labeled input data.

Using the supervised version of autoencoder can guarantee the distinction of the class distributions in the bottleneck layer as well and therefore can result in better detection of OOD samples from unknown classes at test time. OOD sample detection is done by computing the reconstruction loss for each test sample at test time, and comparing the loss to a threshold. Figure 1 shows the architecture used for the supervised autoencoder. Another method that we used to capture this uncertainty is equipping the NNs in the ensemble with an 'unknown' class and then adding various data from other distributions with 'unknown' label to the training data. Training each NN in the ensemble with different 'unknown' labeled data gives the best result for detecting OOD samples.

Figure 2 shows the complete diagram of the proposed framework. Using proper thresholds, the framework can be used for detecting misclassified or OOD samples for image classification.

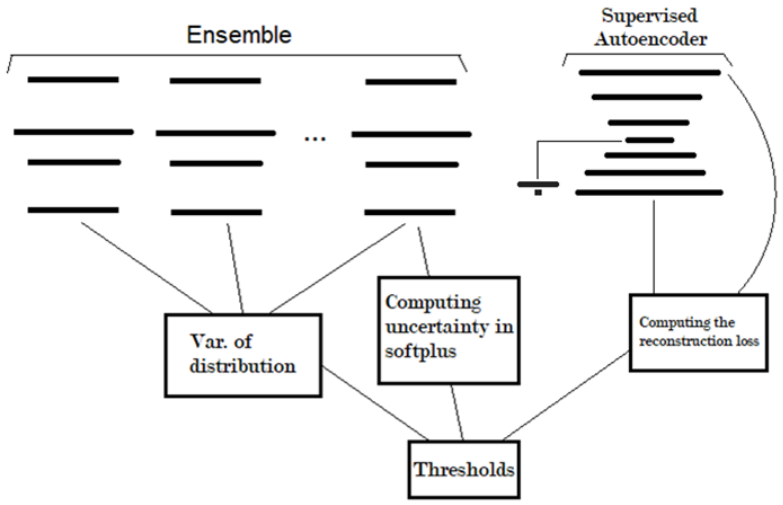

Ensemble
Supervised Autoencoder
Var. of distribution
Computing uncertainty in softplus
Computing the reconstruction loss
Thresholds

Fig. 3: The diagram of the complete proposed framework

4 Experiments

The proposed framework can capture all three kinds of uncertainty. For each one, we've implemented various tests to show the efficiency and higher accuracy of the proposed method compared to the previous works.

First the results of data uncertainty are presented. In figures 4 and 5, we can see the test data points of two datasets (Cifar10 and MNIST) that have large data uncertainty, detected with the proposed method. As one can see, these instances are hard to classify for the model because they look like they can originate from two or more classes. For example, we've detected the planes that are much like birds, which are two classes of data in Cifar10 data set.

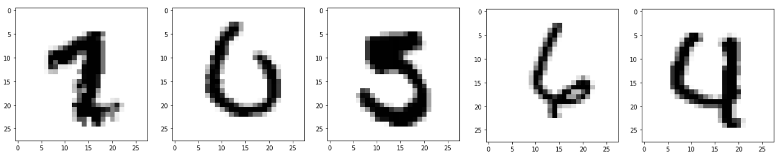


Fig. 4: Test data with high data uncertainty for MNIST dataset.

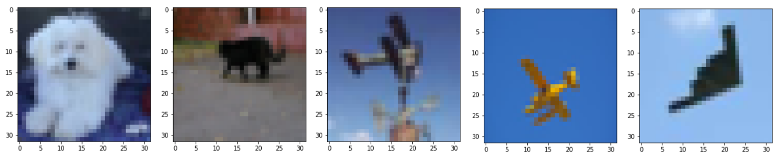


Fig. 5: Test data with high data uncertainty for Cifar10 dataset.

A common use of data uncertainty is the detection of misclassified test samples beforehand. For this test, we've compared the proposed method to MC-Dropout on the ability to detect the hard to classify samples. Figure 6 shows the T-SNE results of the second last layer of a classification network trained on MNIST. As we can see, visualizing this layer can very well demonstrate the different classes' distributions and clusters of the test data. In figures 8, we see the results of misclassified samples detection for the test data using the proposed method and MC-Dropout. As we can see, the proposed method tends to be more accurate in finding the between class or wrongly placed test samples. Moreover, MC-Dropout have more false positives compared to the proposed methods for some classes. This can be confirmed in the numeric results later in this section.

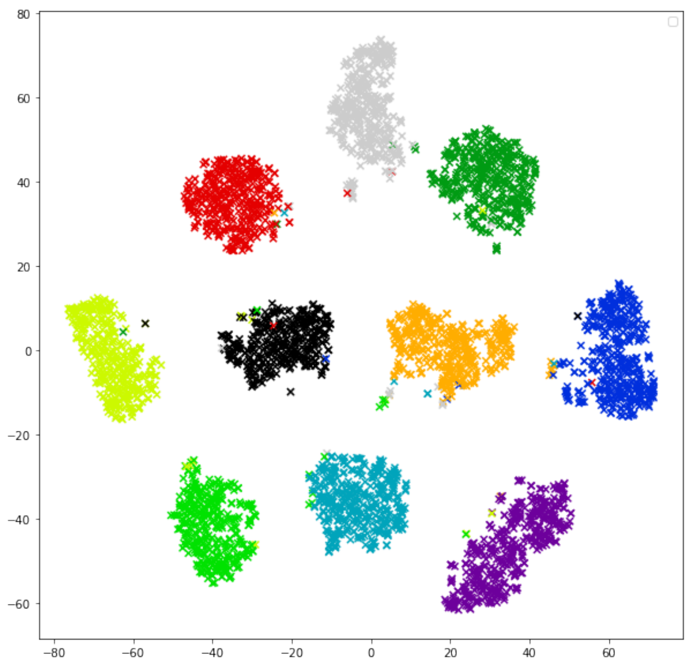
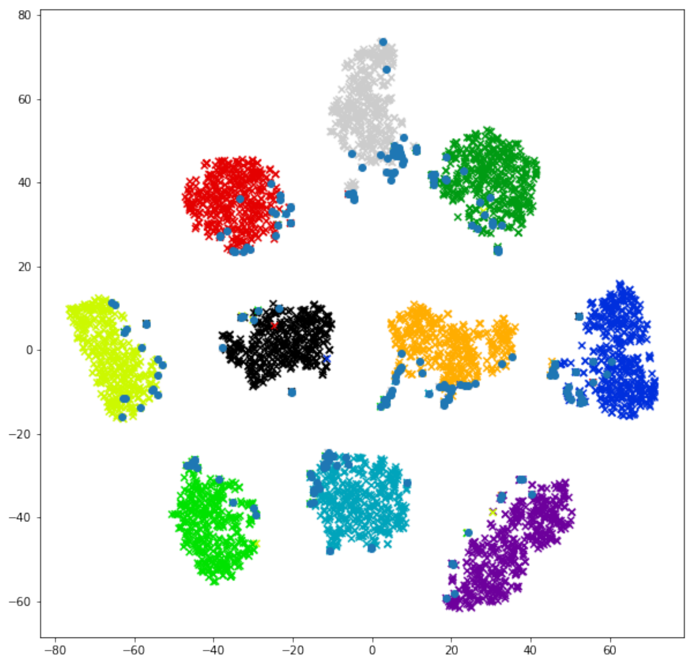
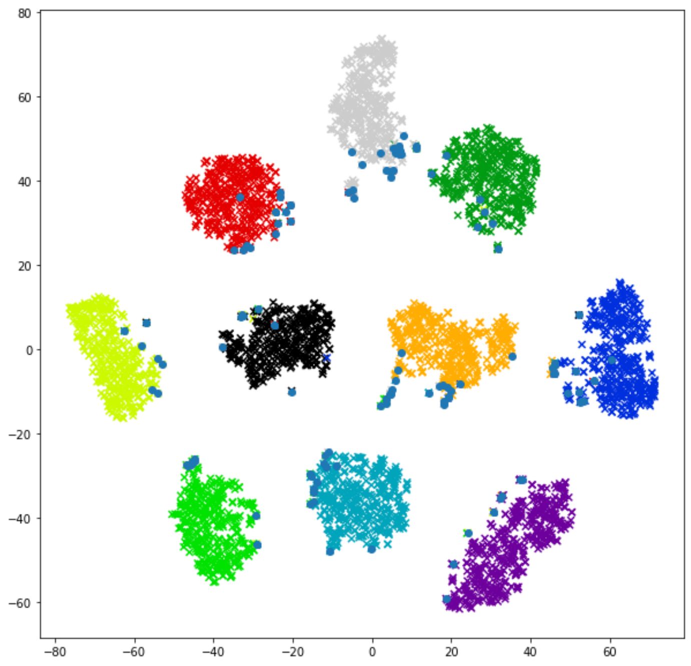


Fig. 6: The T-SNE visualization of the second last layer in a network trained on MNIST data set, for test samples.



(a) (a)



(b) (b)

Fig. 7: The T-SNE visualization of the second last layer in a network trained on MNIST data set, for test samples. The detected misclassified samples with MC-Droput method (a) and the proposed method (b) are painted blue.

For Cifar10 data set, Figure 9 shows the T-SNE results of the second last layer. We've also showed the results of misclassified samples detection for the test data using the proposed method and MC-Dropout on Cifar10 in figure 10. In this task, the proposed method does a lot better in finding the between class or wrongly placed test samples than MC-Dropout, with less false positives and more true positives. The MC-Dropout method seems to act more strict and have more false positives in complex datasets.

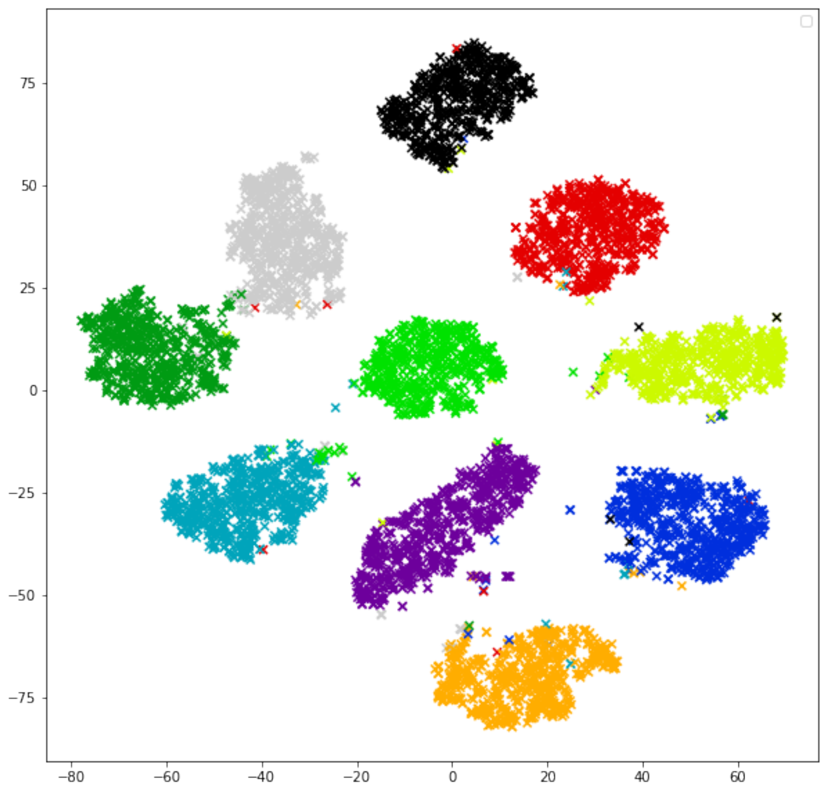
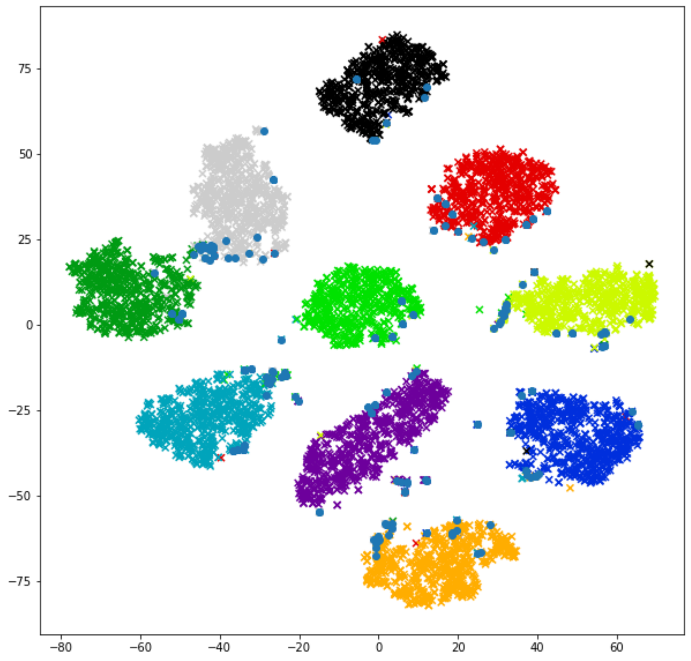
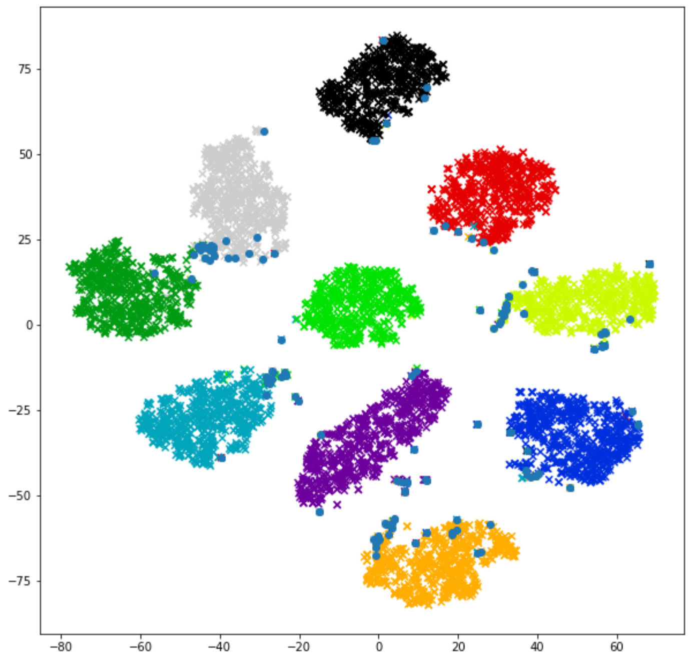


Fig. 8: The T-SNE visualization of the second last layer in a network trained on Cifar10 data set, for test samples.



(a) (a)



(b) (b)

Fig. 9: The T-SNE visualization of the second last layer in a network trained on Cifar10 data set, for test samples. The detected misclassified samples with MC-Droput method (a) and the proposed method (b) are painted blue.

For misclassified detection test, we've compared our proposed method to MC-Dropout and also the baseline method which uses softmax to compute the data uncertainty on various image classification datasets. The datasets we've used include: Mnist ,Fashion-Mnist [17], Cifar10, Cifar100 [18] and TinyImageNet [19]. The metrics used in this paper are True Positive Rate and False Positive Rate [20]. These metrics are commonly used for tasks that include a threshold for detection which is the case in misclassified samples detection.

Mnist		
	TPR	FPR
Baseline (Softmax)	77.6 %	2.9 %
MC-Dropout	80.2 %	7.4 %
Proposed	**84.1** %	**2.3** %

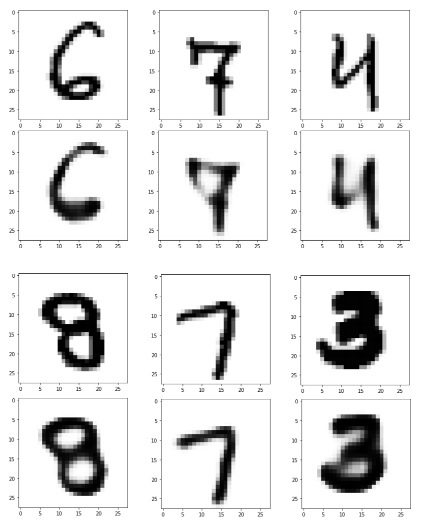
Fashion-Mnist		
	TPR	FPR
Baseline (Softmax)	90.32 %	**3.6** %
MC-Dropout	88.3 %	4.3 %
Proposed	**92.61** %	3.8 %

Cifar10		
	TPR	FPR
Baseline (Softmax)	85.8 %	9.5 %
MC-Dropout	82.1 %	10.1 %
Proposed	**87.6** %	**8.2** %

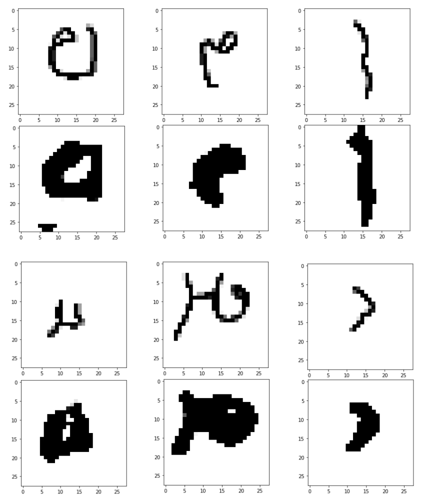
Cifar100		
	TPR	FPR
Baseline (Softmax)	55.68 %	**26.0** %
MC-Dropout	60.54 %	32.7 %
Proposed	**64.21** %	29.5 %

TinyImageNet		
	TPR	FPR
Baseline (Softmax)	53.81 %	42.9 %
MC-Dropout	58.74 %	**28.1** %
Proposed	**59.01** %	30.4 %

For comparing the ability to capture distributional uncertainty, we have tested our framework on the datasets mentioned above for OOD samples detection. The OMNIGLOT dataset [21], scaled down to 28x28 pixels, was used as real 'OOD' data for MNIST. For OOD of Cifar10, we've used TinyImageNet data set which is identical to Cifar10. As shown in Figure 3, the autoencoder can reconstruct the numbers found in the training data much more accurately than the OOD samples. These examples shows the efficiency of the model in OOD detection.

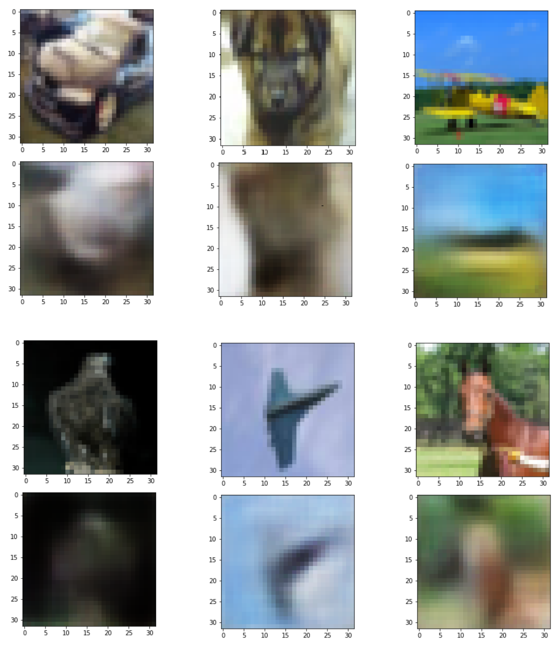


(a) (a)



(b) (b)

Fig. 10: The input and output of the supervised reconstruction autoencoder (trained on MNIST) for some MNIST (a) and OMNIGLOT (b) samples. The OMNIGLOT (OOD) samples are reconstructed poorly, indicating they're from a different distribution than training data.


(a) (a)

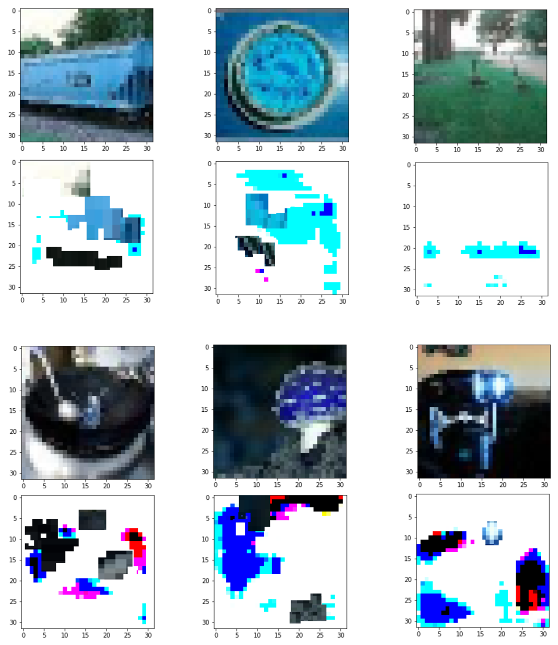

(b) (b)

Fig. 11: The input and output of the supervised reconstruction autoencoder (trained on Cifar10) for a few Cifar10 (a) and TinyImageNet (b) samples. The TinyImageNet (OOD) samples are reconstructed poorly, indicating they're from a different distribution than training data.

Mnist-Omniglot			
	TPR	FPR	False-Neg
Baseline (Autoencoder)	96.6 %	5.25 %	188
MC-Dropout	98.9 %	4.12 %	154
Proposed	**99.2** %	**4.01** %	132

Fashion-Mnist-Omniglot			
	TPR	FPR	False-Neg
Baseline (Autoencoder)	84.1 %	**7.21** %	234
MC-Dropout	87.3 %	9.01 %	198
Proposed	**92.2** %	9.62 %	173

Cifar10-TinyImageNet			
	TPR	FPR	False-Neg
Baseline (Autoencoder)	91.5 %	17.01 %	357
MC-Dropout	90.2 %	**15.12** %	401
Proposed	**95.0** %	15.40 %	273

Cifar100-TinyImageNet			
	TPR	FPR	False-Neg
Baseline (Autoencoder)	56.5 %	29.0 %	427
MC-Dropout	63.7 %	37.4 %	304
Proposed	**66.8** %	**26.2** %	282

Cifar50-Cifar50			
	TPR	FPR	False-Neg
Baseline (Autoencoder)	57.3 %	39.4 %	562
MC-Dropout	**64.2** %	41.9 %	445
Proposed	63.7 %	**37.6** %	483

Finally we've tested our proposed method in case of computing the model uncertainty. For this experiment, we've created different scenarios of changing a model for a classification problem to observe the behaviour of the computed model uncertainty. For each specific model, we've implemented an ensemble of the same networks to capture the distribution on the outputs and interpret the spread of the distribution as model uncertainty.

At first, we've changed the number of layers of a CNN for MNIST classification in each step and then computed the distribution on the ensemble's outputs. In figure 11, we can see that for larger number of layers the distribution is sharper, so the model uncertainty decreases, which can be interpreted as if the model has better capacity for the classification problem at hand.

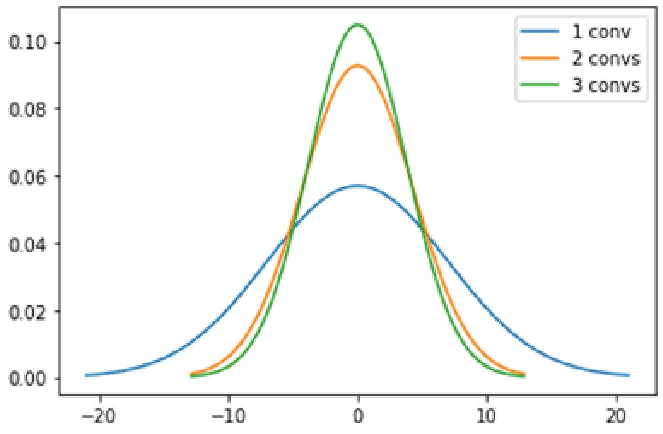


Fig. 12: Distribution of test data outputs on MNIST for different number of layers. Note that the variance of the distribution is the model uncertainty for every scenario

In the next experiment, we've changed the network's loss function in each step. In figure 12, we can see the different model uncertainties (variance of the distribution) for models trained with different loss functions. The two loss functions specified for a multi-class classification problem gives us the least model uncertainty, which is logically accurate. The one with the least model uncertainty, categorical cross-entropy function, is commonly used for classification of MNIST [16].

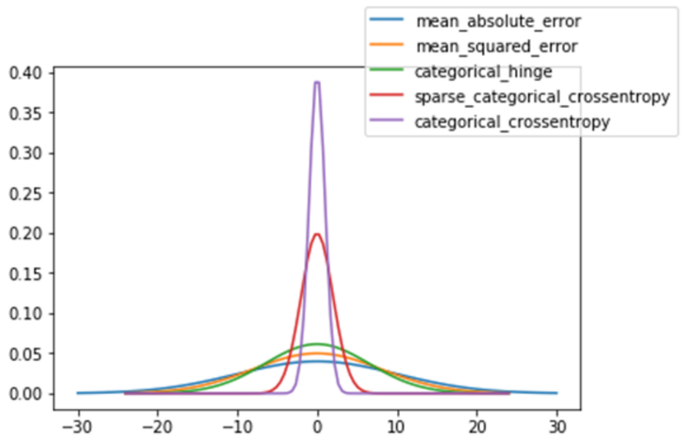


Fig. 13: Distribution of test data outputs on MNIST for different loss functions. Note that the variance of the distributions are the model uncertainty for every loss function used.

Next, we've tested various number of hidden layer neurons of the networks. The model uncertainty seems to be decreasing rapidly, up until the number of neurons reach the satisfying number of 16 for classification of MNIST data set. This also shows the accuracy of the computed model uncertainties.

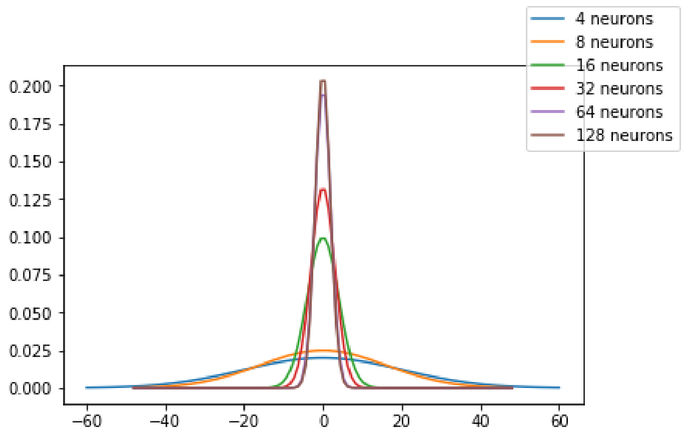


Fig. 14: Distribution of test data outputs on MNIST for different Number of hidden layers' neurons.The variance of the distributions are the model uncertainty for every number of hidden neurons used.

Finally, we experimented different number of training epochs. For a few epochs, the model tend to have high uncertainty due to the fact that it hasn't learned much from the data and the random first initialization are not changed much. For 4 epochs, which is enough for networks' training, and higher numbers the model uncertainty decreases to a minimum.

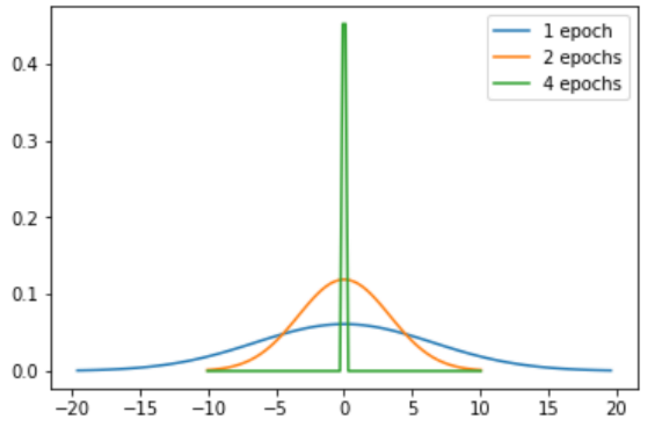


Fig. 15: Distribution of test data outputs on MNIST for different Number of training epochs.The variance of the distributions are the model uncertainty for every scenario.

4.1 Conclusion

This work describes the limitations of previous works on uncertainty estimations as a matter of sources of uncertainty and proposes to treat the three known types of uncertainty in a complete framework. We have presented a novel method, which allows data, distributional and model uncertainty to be treated separately within a consistent interpretable framework. The proposed method is shown to yield more accurate estimates of data uncertainty and distributional uncertainty than MC Dropout and standard baseline softmax probabilities and autoencoders on the task of OOD and misclassified samples detection for various image datasets. Uncertainty measures can be analytically calculated at test time in one pass in the proposed framework, reducing computational cost relative to previous Bayesian or dropout approaches. Having investigated the framework for image classification, it is interesting to apply them to other tasks in computer vision.

References

1. A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: Advances in Neural Information Processing

Systems 25. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097—1105.

2. A. Graves, A. rahman Mohamed, and G. Hinton. “Speech Recognition with Deep Recurrent Neural Networks.” In: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing 2013. IEEE, 2013, pp. 6645—6649.

3. D. Silver et al. “Mastering the game of Go without human knowledge.” In: Nature 550 (2017), pp. 354-–359.

4. M. Bojarski et al. End to End Learning for Self-Driving Cars. 2016. arXiv: 1604.07316 [cs.CV].

5. Y. Shi and A. K. Jain. DocFace: Matching ID Document Photos to Selfies. 2018. arXiv: 1805.02283 [cs.CV].

6. C. Szegedy et al. Intriguing properties of neural networks. 2014. arXiv: 1312. 6199 [cs.CV].

7. MacKay, D.J., 1995. Bayesian neural networks and density networks. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 354(1), pp.73–80.

8. Yarin Gal, Uncertainty in Deep Learning, Ph.D. thesis, University of Cambridge, 2016.

9. Y. Gal and Z. Ghahramani. “Dropout as a Bayesian Approximation: Rep- resenting Model Uncertainty in Deep Learning.” In: Proceedings of the 33rd International Conference on Machine Learning. Ed. by M. F. Balcan and K. Q. Weinberger. PMLR, 2016, pp. 1050-–1059.

10. A. Kendall and Y. Gal, “What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision,” in Proc. Conference on Neural Information Processing Systems (NIPS), 2017.

11. B. Lakshminarayanan, A. Pritzel, and C. Blundell. “Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles.” In: Advances in Neural Information Processing Systems 30. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 6402-–6413.

12. T. Amarbayasgalan, B. Jargalsaikhan, K. Ryu, T. Amarbayasgalan, B. Jargalsaikhan, and K. H. Ryu, “Unsupervised Novelty Detection Using Deep Autoencoders with Density Based Clustering,” Applied Sciences, vol. 8, no. 9, p. 1468, 8 2018.

13. Dan Hendrycks, Kevin Gimpel, “A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks,” ICLR 2017.

14. Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin, “Training confidence-calibrated classifiers for detecting out-of-distribution samples,” International Conference on Learning Representations, 2018.

15. Maaten LV, Hinton G. Visualizing data using t-SNE. Journal of machine learning research. 2008;9(Nov):2579-605.

16. Baldominos, A., Saez, Y. and Isasi, P., 2019. A survey of handwritten character recognition with mnist and emnist. Applied Sciences, 9(15), p.3169.

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

18. Krizhevsky, A. and Hinton, G., 2009. Learning multiple layers of features from tiny images.

19. Le, Y. and Yang, X., 2015. Tiny imagenet visual recognition challenge. CS 231N.

20. Davis, J. and Goadrich, M., 2006, June. The relationship between Precision-Recall and ROC curves. In Proceedings of the 23rd international conference on Machine learning (pp. 233-240).

21. Lake, B.M., Salakhutdinov, R. and Tenenbaum, J.B., 2019. The Omniglot challenge: a 3-year progress report. Current Opinion in Behavioral Sciences, 29, pp.97-104.