

A Comparative Study of Uncertainty Estimation Methods in Deep Learning Based Classification Models

Iswariya Manivannan

Publisher: Dean Prof. Dr. Wolfgang Heiden

**Hochschule Bonn-Rhein-Sieg – University of Applied Sciences,
Department of Computer Science**

Sankt Augustin, Germany

September 2020

Technical Report 04-2020



**Hochschule
Bonn-Rhein-Sieg**
University of Applied Sciences

ISSN 1869-5272

ISBN 978-3-96043-085-8



We gratefully acknowledge
the support by
Bosch Center for
Artificial Intelligence (BCAI)

This work was supervised by

Laura Beggel
Deebul Nair
Paul G. Plöger

Copyright © 2020, by the author(s). All rights reserved. Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Das Urheberrecht des Autors bzw. der Autoren ist unveräußerlich. Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Das Werk kann innerhalb der engen Grenzen des Urheberrechtsgesetzes (UrhG), *German copyright law*, genutzt werden. Jede weitergehende Nutzung regelt obiger englischsprachiger Copyright-Vermerk. Die Nutzung des Werkes außerhalb des UrhG und des obigen Copyright-Vermerks ist unzulässig und strafbar.

Digital Object Identifier <https://doi.org/10.18418/978-3-96043-085-8>

Abstract

Deep learning models produce overconfident predictions even for misclassified data. This work aims to improve the safety guarantees of software-intensive systems that use deep learning based classification models for decision making by performing comparative evaluation of different uncertainty estimation methods to identify possible misclassifications.

In this work, uncertainty estimation methods applicable to deep learning models are reviewed and those which can be seamlessly integrated to existing deployed deep learning architectures are selected for evaluation. The different uncertainty estimation methods, deep ensembles, test-time data augmentation and Monte Carlo dropout with its variants, are empirically evaluated on two standard datasets (CIFAR-10 and CIFAR-100) and two custom classification datasets (optical inspection and RoboCup@Work dataset). A relative ranking between the methods is provided by evaluating the deep learning classifiers on various aspects such as uncertainty quality, classifier performance and calibration. Standard metrics like entropy, cross-entropy, mutual information, and variance, combined with a rank histogram based method to identify uncertain predictions by thresholding on these metrics, are used to evaluate uncertainty quality.

The results indicate that Monte Carlo dropout combined with test-time data augmentation outperforms all other methods by identifying more than 95% of the misclassifications and representing uncertainty in the highest number of samples in the test set. It also yields a better classifier performance and calibration in terms of higher accuracy and lower Expected Calibration Error (ECE), respectively. A python based uncertainty estimation library for training and real-time uncertainty estimation of deep learning based classification models is also developed.

Acknowledgements

I would like to thank my supervisors Prof. Dr. Paul Plöger, Deebul Nair and Laura Beggel from Bosch Center for Artificial Intelligence (BCAI) for providing the opportunity to work on this Research and Development project.

I would like to thank Laura Beggel and Deebul Nair for their continuous encouragement, guidance and support throughout this project. I would also like to thank William Beluch and other BCAI colleagues for their inputs and BCAI for this wonderful opportunity.

Finally, I extend my gratitude to my friends, especially Sathiya Ramesh and my family for their endless support, motivation and love.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Challenges and Difficulties	3
1.3	Problem Statement	4
2	Background	7
2.1	Bayesian Modeling	7
2.2	Variational Inference	9
2.2.1	Approximate inference	10
3	Related Work	13
3.1	Approximate Bayesian Methods for Uncertainty Estimation	14
3.1.1	Monte Carlo dropout	14
3.1.2	Stochastic batch normalization	17
3.1.3	Test-time data augmentation	19
3.2	Non-Bayesian Methods for Uncertainty Estimation	21
3.2.1	Deep ensembles	21
3.2.2	Softmax calibration	22
3.2.3	Selective classification	24
4	Methodology	27
4.1	Datasets	27
4.2	Models	29
5	Experimental Evaluation	35
5.1	Metrics	36
5.1.1	Sharpness Metrics	36

5.1.2	Classifier Performance Metrics	37
5.1.3	Calibration Metrics	39
5.2	Comparison Based on Uncertainty Quality	40
5.2.1	Metric identification	40
5.2.2	Method identification	45
5.2.3	Identification of misclassified samples	48
5.2.4	Performance on Out Of Distribution (OOD) data	52
5.3	Comparison Based on Classifier Performance	53
5.4	Comparison Based on Classifier Calibration	56
5.5	Discussions	56
6	Conclusion	61
6.1	Contributions	62
6.2	Lessons Learned	62
6.3	Future Work	63
Appendix A	Evaluation using Risk Coverage curves	65
Appendix B	Evaluation using Brier score	67
Appendix C	Uncertainty Estimation Work Flow Data	69
References		71

1

Introduction

The effectiveness of deep learning based image classification models over other traditional machine learning methods is undeniable. Despite their high accuracy, often an unrealistic high confidence value is attributed to wrong predictions [14], [34]. As a result it is important for users to be aware of the ramifications of the decisions made by the deep learning model in order to mitigate the adverse socio-economic impacts of unreliable predictions from models. One solution to address the problem of unreliable predictions is to produce a reliability score along with the model predictions. The uncertainty estimate also called the negative reliability score (i.e., the higher the uncertainty, the lesser is the reliability), helps the user understand what the classifier knows and what it does not know. This enables deep learning practitioners to make informed decisions in cases of high uncertainty, such as ignoring the model’s predictions and later re-training the model with those highly uncertain inputs to improve performance and so on. In this work we explore the power of uncertainty estimation to circumvent ”the problem of accidents in machine learning” [2] to a certain extent, where the unintended errors produced by the model can have potentially harmful consequences.

1.1 Motivation

Deep learning models do not provide reliable predictions. Their increasing usage in various domains such as medical diagnosis [37], autonomous driving [35], or drug discovery [44] can be attributed to the fact that they are able to quickly

analyze and make predictions on large diverse datasets with high accuracy. However, incorrect model predictions in safety-critical applications endanger human lives. One such example is when IBM’s Watson supercomputer often gave erroneous, unsafe treatment recommendations to patients [48]. Another instance is the accident caused by Uber’s self-driving car resulting in the death of a pedestrian, due to the fact that its perception model failed to detect the person on a bicycle most likely because of insufficient illumination [42]. In most of the cases, the exact reason remains unknown because deep learning models often act as a black-box [46]. Although several approaches try to uncover the actual operating mechanism under the hood, explainable AI is far from maturity [1]. As a result, we wish to take an alternate route by making the model speak for itself by providing uncertainty estimates when it is unsure of its predictions, as in the case of the above mentioned examples.

One of our applications of deep learning involves the use of a deep neural network based classification model for optical inspection at Robert Bosch GmbH. The main objective of the optical inspection process is to distinguish defective parts from non-defective ones. The defective parts are produced when the manufacturing process is improper/incomplete and are generally manually classified by a human inspecting the images of these parts. The optical inspection process is automated by deploying a trained deep learning based classification model that can distinguish between the defective and non-defective parts. It is necessary that the automation of this inspection process by means of a deep learning model guarantees that defective products are not being classified as non-defective ones (i.e., a low miss-rate). Hence to ensure that defective parts do not get classified as non-defective ones, uncertainty estimation techniques are applied to provide a statistics on how reliable the model’s predictions are.

Another similar application uses a deep neural network based perception module in the KUKA YouBot, an industrial service robot extensively used in the RoboCup@Work league [31]. Robots participating in the competitions of the RoboCup@Work league are required to execute efficient and intelligent solutions to the given scenarios, such as pickup and delivery of tools from different stations. The perception module uses a deep learning model to extract and classify the objects present in the station image from the robot’s camera. Once an object is classified as a tool, it is picked up by the robot’s arm, placed on the robot’s base and carried

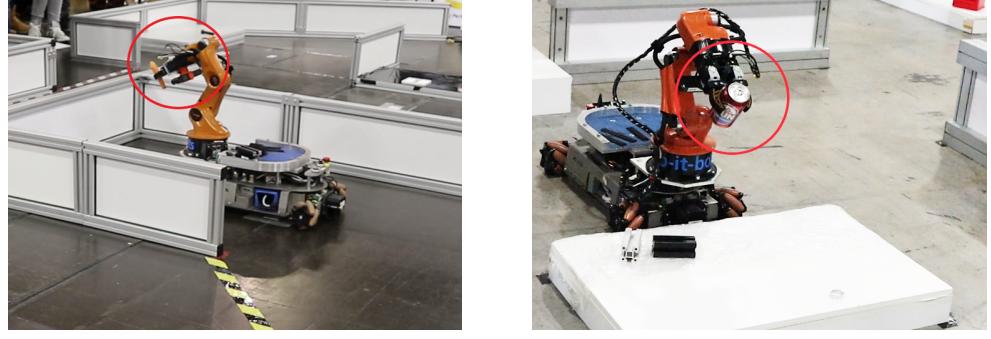


Figure 1.1: Images of KUKA YouBot picking up objects which are not tools from the station [38].

to the delivery station. Fig. 1.1 shows the YouBot accidentally picking up a carrot and a beer can because of confident misclassifications made by the deep learning based classification model. As a result, the robot receives a penalty for picking up a non-tool object. However, if the model is able to provide an uncertainty score along with its predictions (thereby indicating that it is unsure of its predictions), then alternative solutions can be devised, such as ignoring current object and moving on to the next.

1.2 Challenges and Difficulties

Uncertainty estimation methods need to address the following problems to yield reliable confidence estimates.

- Deep learning models produce point estimates as outputs, which do not yield any measure of uncertainty. Therefore, it is imperative that models must be adapted to produce distributional outputs from which uncertainty estimates can be extracted.
- The softmax function used in deep learning based classification models produce overconfident point estimates. The softmax activation is used in the last layer of the network to scale each of the logit values per class to a value between 0 and 1. The softmax outputs of all logits sum to 1, thereby giving the softmax an interpretation that it converts the predicted logit values into class confidence probabilities. The model’s prediction is given by the class to which

the maximum confidence value is attributed. However, the softmax is known to produce very high confidence values even when the model is given a input data which it has not seen during training [14].

- Deep learning models do not provide calibrated outputs [22]. Calibration measures the reliability of the model in producing consistent predictions. A consistent prediction for instance would refer to the case where the classifier is 90% accurate when it makes a prediction with 90% confidence. For a calibrated model, the predicted confidence estimates should reflect the probability of the prediction being correct, i.e., the predicted confidence and model accuracy should bear a linear relationship. This does not hold for deep learning models, as calibration is not taken into account during the training process.
- An important constraint posed by our optical inspection problem is that uncertainty estimation methods should be fast and packageable as an auxiliary module which can be integrated on top of existing deep learning models. This is because we aim to estimate the predictive uncertainty of an already deployed model. Thus, these methods should not introduce major architecture changes that would cause the re-training and hyper-parameter tuning of the deployed deep learning model.

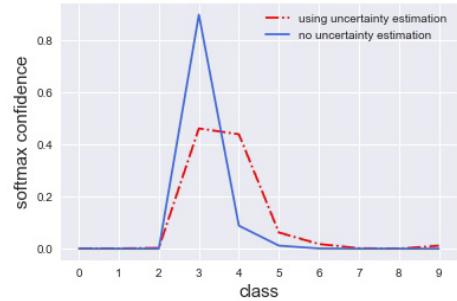
1.3 Problem Statement

The purpose of uncertainty estimation is to obtain reliable predictions from a classifier. The single point estimates predicted by existing deep learning classifiers need to be replaced by predictive distributions to quantify uncertainty (Fig. 1.2). It is vital for the predictive distributions to be sharp for highly certain predictions and flat for data-points where the classifier is not confident about its predictions. The sharper the individual predictive distributions are, the lesser is the uncertainty per data-point and vice-versa. Therefore, we first identify uncertainty estimation methods that satisfy the constraint posed by our optical inspection problem (section 1.2) and are able to obtain predictive distributions from which uncertainty estimates can be extracted.

In the case of deep learning models, sharpness is usually achieved during the training process by minimizing the Negative Log-Likelihood (NLL) objective. The



(a) Deer image (class 4) from CIFAR10 dataset



(b) Distribution of softmax confidence probabilities per class

Figure 1.2: Fig. 1.2a shows the image of a deer belonging to class 4 of the CIFAR-10 dataset classified using a VGG-16 model. The blue line depicts the over-confident wrong prediction made by the model when it is not using any uncertainty estimation method. The model makes a prediction that the image is a cat (class 3) with a high confidence probability of 0.89. Fig. 1.2b illustrates the reduction in sharpness of the softmax confidence when using uncertainty estimation methods (Monte Carlo dropout and test time augmentation).

result of this training process yields deep learning models which produce overconfident predictions (popularly referred to as the softmax overconfidence problem [14]). As a result, it is desirable to have flatter distributions (i.e., with greater uncertainty) to compensate for the unreliable over-confident predictions. Hence, we compare uncertainty estimation techniques applicable to deep learning models, which are not only able to produce predictive distributions but also reduce their excessive sharpness, leading to an increase in uncertainty. Appropriate metric(s) that are able to quantify the increase in model uncertainty on the application of different methods have to be selected from the vast collection available in the literature. The selected metric is used to measure the uncertainty of the misclassified samples in the dataset and also to identify the method which detects the highest number of the misclassified samples as uncertain.

While it is important to reduce the excessive sharpness of the predictions, it is also necessary to consider the notion of classifier calibration (more details in Chap. 3), which gives the statistical consistency between the model’s predictions and the observations [20]. In practice, deep learning models have the inherent property

1.3. Problem Statement

of being miscalibrated [22] and the improvement in classifier calibration after the application of different uncertainty estimation methods will be investigated. This is essential because sharpness or calibration by themselves alone cannot guarantee reliable predictions and hence, it is necessary to jointly optimize for both. As a result it is beneficial to have uncertainty estimation methods that are able to tackle both the softmax over-confidence and the calibration problem.

2

Background

This chapter presents an overview of the various techniques used to formalize an uncertainty estimation procedure applicable to models ranging from simple Bayesian models to neural networks (for more details refer Chap. 2 and 3 of Gal [13]). Although, these methods are fundamental and do not scale well to deep neural networks, they form the basis of many uncertainty estimation methods developed for deep neural networks discussed in the upcoming chapters. These techniques aim at obtaining output probability distributions, which are in general able to capture the statistical dispersion of the represented quantity -a necessary attribute for quantifying uncertainty- from the model at hand.

2.1 Bayesian Modeling

The Bayesian approach is used to quantify the uncertainty in our beliefs using probability distributions [40]. Let us consider the low dimensional input data-points given by $X = x_1, x_2, \dots, x_n$ and their corresponding output labels $Y = y_1, y_2, \dots, y_n$ generated from some unknown probability distribution. We are required to find a function which is a mapping between the input data-points X and the outputs Y . Suppose we use a Bayesian model parameterized by the model parameters ω which are drawn from the initial prior distribution $P(\omega)$ to learn this mapping function, then the prior signifies our initial belief on this mapping function. The goal of the learning process is to modify the prior distribution based on the training data in such a way that the model is transformed into an ideal or close to ideal mapping

function between X and Y . The below equation depicts how the prior distribution of the weights $P(\omega)$ is modified to give the posterior $P(\omega|X, Y)$ based on the data likelihood $P(Y|X, \omega)$.

$$P(\hat{\omega}|X, Y) = \frac{P(Y|X, \omega)P(\omega)}{P(Y|X)} \quad (2.1)$$

where,

$$P(Y|X) = \int P(Y|X, \omega)P(\omega)d\omega \quad (2.2)$$

The denominator in Eq. (2.1), which is central to the computation of the posterior distribution is called the model evidence. Since Eq. (2.2) performs a marginalization by computing the integral over all possible values of ω , it is also called the marginal likelihood. The use of conjugate priors facilitates the easy computation of the posterior in Eq. (2.1). This is because conjugate priors mimick the form of the likelihood distribution and yield a posterior which is in the same family as the prior.

Having computed the posterior of the weights $P(\hat{\omega}|X, Y)$, the next step in a Bayesian model would be to make an inference for a new test data-point. This essentially means that the model is expected to make a prediction y^* for a test input x^* given its learned weights $\hat{\omega}$. Given the test input, the output can be inferred by integrating over all values of the posterior probability distribution of $\hat{\omega}$.

$$P(y^*|x^*, x, y) = \int P(y^*|x^*, \hat{\omega})P(\hat{\omega}|X, Y) \quad (2.3)$$

From Eq. (2.2) and (2.3), it is seen that a simple Bayesian model is capable of providing a distributional output (weights and predictions), in contrast to a single point estimate which is generally being used/obtained from classical non-Bayesian neural networks. Distributions are very convenient because uncertainty estimates can be effortlessly extracted (for instance computing the variance) and point estimates are obtained using various statistics such as mean, median or mode. However, it is not possible to obtain such predictive posterior distributions in deep neural networks using such simple Bayesian modeling tools. This is because the computation of the posterior in Eq. (2.1) and the closed form computation of the integral in Eq. (2.3) is intractable due to the complex data likelihood function and the presence of millions of weight parameters with infinite combinations of values. As a result, we resort to

approximate inference techniques, when closed form computation is not possible.

2.2 Variational Inference

When it is not possible to analytically compute the true posterior $P(\omega|X, Y)$, variational inference [8] is used. It involves using an approximate variational distribution $q(\omega)$ parameterized by θ instead of the true posterior $P(\omega|X, Y)$. The parameter θ is also called a latent variable θ , since it is not directly observed but inferred from the observed data-points X . It is estimated using an optimization procedure. The aim of the optimization process is to find the set of latent variables θ which can best describe the observed data. The Kullback-Leibler (KL) divergence (also called relative entropy) [33] which is used to measure the similarity between the two distributions is used as the optimization objective. Thus, the optimization process aims to minimize the KL divergence so that $q(\omega)$ is very close to $P(\omega|X, Y)$. The below equation gives the KL divergence between the two distributions

$$\begin{aligned} KL(q(\omega)||P(\omega|X, Y)) &= \int q(\omega) \log\left(\frac{q(\omega)}{P(\omega|X, Y)}\right) d\omega \\ &= \int q(\omega) \log(q(\omega)) d\omega - \int q(\omega) \log(P(\omega|X, Y)) d\omega \\ &= \int q(\omega) \log(q(\omega)) d\omega - \int q(\omega) \log(P(\omega, X, Y)) d\omega + \log(P(X, Y)) \end{aligned} \tag{2.4}$$

where,

$$ELBO = - \int q(\omega) \log(q(\omega)) d\omega + \int q(\omega) \log(P(\omega, X, Y)) d\omega \leq \log(P(X, Y)) \tag{2.5}$$

The ELBO as the name suggests is a lower bound on the probability of observed data (also called evidence). Substituting Eq. (2.5) in Eq. (2.4) gives the following equation.

$$KL(q(\omega)||P(\omega|X, Y)) = -ELBO + \log(P(X, Y)) \tag{2.6}$$

Eq. (2.6) shows that an alternative to minimizing the KL divergence objective is to maximize the ELBO. The log probability $\log(P(X, Y))$ of the data is an additive constant which can be ignored during the optimization process. Essentially this

means that increasing the lower bound on the data makes it closer to the true posterior $P(\omega|X, Y)$ distribution. This process of using an approximate distribution $q(\omega)$ to model the true posterior $P(\omega|X, Y)$ and optimizing the parameters of $q(\omega)$ by minimizing the KL divergence is called variational inference. Variational inference has two major advantages: Firstly, the marginalization process Eq. (2.2) required to compute the true posterior $P(\omega|X, Y)$ is not necessary. Secondly, it outputs a probability of weights $q(\omega)$ from which uncertainty estimates can be derived, in contrast to deep learning methods which provide point-estimates of the network weights. However, this also does not scale well to large datasets or deep learning models with several parameters because computing the integral in Eq. (2.4) is intractable. We shall discuss several methods in Chap. 3 that apply an approximate Bayesian inference procedure to deep neural networks to obtain output probability distributions instead of point estimates.

2.2.1 Approximate inference

Approximate inference in neural networks can be traced back to Hinton and Van Camp [26] who utilize the Minimum Description Length (MDL) principle [47] to restrict the amount of information on the weights by adding noise to reduce over-fitting. Here a neural network with a single hidden layer is trained with its weights drawn from an independent Gaussian distribution. During back-propagation the gradients for the mean and variance of the respective Gaussian distributions can be computed because it is possible to perform analytical integration over the Gaussian distribution. This technique is applicable only for single layer neural networks because of its computational complexity, and it relies on the simplifying assumption that the weights of the hidden layer are drawn from independent Gaussian distributions. Barber and Bishop [6] improved on the work of Hinton and Van Camp [26] by modeling the dependency between the weights using non-diagonal covariance matrices. However, this comes at an additional increased computational cost and it is impossible to perform this analytical computation when this method is applied to deep neural networks.

Graves [21] extends the work of Hinton and Van Camp [26] to a recurrent neural network by using Monte Carlo sampling for computing the posterior of the weights.

But Monte Carlo estimators are known for their high variance noisy estimates and when combined with Stochastic Gradient Descent (SGD) optimizer, perform poorly in terms of noisy, inaccurate results. Blundell et al. [9] uses the re-parametrization trick, which was developed by Kingma and Welling [30] in the context of latent variable models, to obtain uncertainty estimates of the weights of the neural network. The re-parametrization trick allows a continuous random weight parameter of a neural network drawn from a Gaussian distribution to be expressed as a sum of a deterministic variable and a noise parameter sampled from a Gaussian distribution. Applying this re-parametrization trick to the previous methods, where the weights are directly sampled from independent Gaussian distributions, enables the representation of each weight parameter by a deterministic mean μ and standard deviation σ parameter, with a small Gaussian noise added to σ . During back-propagation the deterministic gradients are computed with respect to μ and σ for each weight parameter. This method is more efficient as compared to the previous methods of representing the weights as a distribution. However, the number of parameters of the network are doubled since each weight is represented by its mean and standard deviation, hence training and inference would cost more in terms of time and memory.

3

Related Work

In the previous chapter, we have discussed several techniques to obtain output distributions from a neural network. However, most of these techniques are either analytically intractable or are slow, approximate and inaccurate due to the complexity of deep neural networks. This chapter presents the state-of-the-art uncertainty estimation methods which can be applied to deep neural networks. The uncertainty estimation methods can be broadly classified into two categories - Bayesian and non-Bayesian. These methods extend the techniques discussed previously to obtain approximate output distributions with fast and accurate uncertainty estimates.

The obtained predictive uncertainty is of two types epistemic uncertainty and aleatoric uncertainty [57], or a combination of both. Epistemic uncertainty refers to the uncertainty in model parameters, especially when the model has not seen enough data. This type of uncertainty is reducible as it decreases when more data is given to the model. Aleatoric uncertainty on the other hand, refers to noise in the data (image noise in the case of image datasets). This is further classified into two types - homoscedastic and heteroscedastic noise. The former denotes the noise that is prevalent in the entire dataset, while the latter represents the noise that is unique to each data-point.

3.1 Approximate Bayesian Methods for Uncertainty Estimation

The training of Bayesian models usually involves placing a prior distribution over the parameters of the model, computing the posterior given the data likelihood and performing inference over the test data by marginalizing over the posterior of the learned parameters (Section 2.1). Bayesian modeling and inference are intractable for deep learning models due to the reasons discussed in the previous chapter. For this reason we resort to approximation methods for posterior computation and marginalization. Our problem statement requires a deep learning based classification model which has practical time and resource constraints, and should also be able to handle high dimensional inputs such as images. Hence, we are seeking uncertainty estimation methods that are applicable to current deep learning architectures such as VGG [53], ResNet [24], Xception [10], MobileNet [27]. On the other hand, Bayesian uncertainty estimation methods developed in the context of Bayesian Neural Networks (BNN) developed by Depeweg et al. [11], Hernández-Lobato and Adams [25], Blundell et al. [9] are not discussed here. This is because it is not possible to deploy BNNs for real-world scenarios because training and inference is very slow.

3.1.1 Monte Carlo dropout

Dropout is a regularization technique used in deep learning models to reduce over-fitting [54]. Dropout is originally designed to be used during the training phase where the model is trained by randomly turning off or dropping a set of neurons with a certain probability, called the dropout probability (or dropout mask). This in turn gives a regularization effect because it indirectly combines many different network architectures approximately and efficiently. During test time, dropout is deactivated and predictions are made without dropping any weights.

A neural network with dropout applied to its weight layers can be cast as an approximate Gaussian Process (GP) [41]. It is a well known fact that making an inference from a GP is advantageous since the output is a distribution rather than a point estimate and uncertainty estimates can be drawn from this output distribution. Therefore, by approximating a deep neural network to a GP one can obtain uncertainty estimates for the predictions made by the network [14].

The below equation 3.1 describes a typical loss function used in deep neural networks where, the first term on the right hand side represents the squared loss or the softmax loss between the network predictions \hat{y}_i and the target y_i for N training samples, and the second term is the L_2 regularization used to reduce over-fitting. L_2 regularization contains the product of the weight decay constant λ and the squared L_2 norm of the weights W_i and biases b_i of the L-layered network.

$$L_{dropout} = \frac{1}{N} \sum_{i=1}^N E(y_i, \hat{y}_i) + \lambda \sum_{i=1}^L (\|W_i\|_2^2 + \|b_i\|_2^2) \quad (3.1)$$

The predictive probability for a test data-point (x^*, y^*) of the deep GP model with weights ω trained on the training data (X, Y) is given as follows,

$$P(y^*|x^*, X, Y) = \int P(y^*|x^*, \omega) P(\omega|X, Y) d\omega \quad (3.2)$$

The computation of the true posterior of the weights $P(\omega|X, Y)$ is intractable, hence an approximate posterior given by the dropout weights $q(\omega)$ is used. This approximate inference problem is solved by minimizing the KL divergence objective.

$$-\int q(\omega) \log(p(Y|X, \omega)) d\omega + KL(q(\omega)||p(\omega)) d\omega \quad (3.3)$$

The minimization objective in Eq. (3.3) is solved by approximating the integral with a Monte Carlo substituting a regularization term with prior-length scale l and model precision τ as shown below.

$$L_{GP-MC} \propto \frac{1}{N} \sum_{i=1}^N \frac{-\log(p(y_n|x_n, \hat{\omega}_n))}{\tau} + \sum_{i=1}^L \left(\frac{p_i l^2}{2\tau N} \|M_i\|_2^2 + \frac{l^2}{2\tau N} \|m_i\|_2^2 \right) \quad (3.4)$$

It can be seen that Eq. (3.1) is similar to Eq. (3.4) and proves that the minimization of the loss function in a dropout neural network is equivalent to minimizing the KL divergence between the approximate posterior obtained using the dropout weights and the true posterior of the deep GP. During inference the approximate predictive distribution $q(y^*|x^*)$ is obtained using the approximate posterior $q(\omega)$ (weights of

3.1. Approximate Bayesian Methods for Uncertainty Estimation

the dropout network).

$$q(y^*|x^*) = \int P(y^*|x^*, \omega) q(\omega) d\omega \quad (3.5)$$

The integral can be approximated using Monte Carlo estimation, which essentially means that the average of the prediction confidences over T forward passes is taken, where W_l^t is the dropout weight for layer l and forward pass t .

$$E_{q(y^*|x^*)}(y^*) = \frac{1}{T} \sum_{t=1}^T \hat{y}^*(x^*, W_1^t, \dots, W_L^t) \quad (3.6)$$

The expected prediction confidence $E_{q(y^*|x^*)}(y^*)$ for a test data-point x^* with label y^* is obtained by computing the average of the predictions y^* from T forward passes. The expected prediction confidence is a more reliable estimate of the model prediction confidence on a particular data-point as compared to the prediction from a single forward pass. Uncertainty estimates can be obtained by computing the variance or entropy of the predictions over several forward passes. The number of forward passes depends mostly on the model and dataset at hand but an approximate upper limit on the number of forward passes is 50.

The approach discussed so far yields uncertainty estimates which are a function of the model weights, more specifically the weights produced by dropout. Hence, it is more appropriate to classify the uncertainty estimates from Monte Carlo (MC) Dropout as epistemic uncertainty estimates. Kendall and Gal [29] introduces a slight modification to the network architecture and loss function of the deep learning model so as to produce an additional estimate of aleatoric uncertainty, which is a measure of the inherent noise in the dataset. A typical classification model produces as many logit values per image f_i as the number of classes c in its last layer, which is then passed to an activation function such as softmax to give per class confidence values. An additional logit is added to the last layer to provide an aleatoric uncertainty estimate per image. Thus, the network has two output nodes, one producing the softmax output $\text{softmax}(f_i)$ and the other producing an aleatoric uncertainty estimate σ . An extra term is added to the loss function of the network in order to optimize the

weights of σ as shown in Eq. (3.7).

$$L_{aleatoric} = \frac{1}{N} \sum_{i=1}^N \log \frac{1}{M} \sum_{m=1}^M \exp(\hat{f}_{i,m,c} - \log \sum_c \hat{f}_{i,m,c}) \quad (3.7)$$

where,

$$\hat{f}_{i,m} = f_i + \sigma_i \epsilon_m, \quad \epsilon_m \sim \mathcal{N}(0, 1) \quad (3.8)$$

The aleatoric component of the loss function models an approximation to the distribution of true noise per data-point by drawing samples from a Gaussian prior Eq. (3.8) and performing Monte Carlo integration of M such samples.

This method of aleatoric uncertainty estimation can be combined with MC dropout to obtain both the aleatoric and epistemic uncertainty estimates per data-point. The combined loss function is given in Eq. 3.9.

$$L_{combined} = \frac{1}{N} \sum_{i=1}^N E(y_i, \hat{y}_i) + \frac{1}{N} \sum_{i=1}^N \log \frac{1}{M} \sum_{m=1}^M \exp(\hat{f}_{i,m,c} - \log \sum_c \hat{f}_{i,m,c}) \quad (3.9)$$

These methods suffer from minor drawbacks depending on the real-time application they are used for. The quality of epistemic uncertainty is directly proportional to the number of MC samples but an increase in number of MC samples per image results in an increased inference time. In situations where it is necessary to obtain quick uncertainty estimates, a compromise on uncertainty quality must be made by reducing the number of forward passes when choosing MC dropout. Aleatoric uncertainty estimation on the other hand has no such trade-off issues between inference time and quality but comes with its own cost, as it introduces changes to the network architecture and results in model re-training. This might not be useful for already deployed models, however MC dropout can be used here for obtaining uncertainty estimates given the existence of dropout layers in the deployed model.

3.1.2 Stochastic batch normalization

Input data fed in batches to a neural network is non-linearly transformed (using relu, tanh, sigmoid, leaky relu and others) as it passes from layer to layer until it

3.1. Approximate Bayesian Methods for Uncertainty Estimation

reaches the output layer. This non-linear transformation in each layer causes arbitrary scaling of the data thereby causing a distributional change with respect to the original input data. Thus, the weights of each layer of the network are optimized with respect to the distribution of data from the previous layer and not with respect to the input distribution. This distributional change of the input data as it passes through the layers of the network is termed as covariate shift [52]. Batch normalization attempts to rectify this problem by normalizing the batch of data in each layer [28] before it is given to the non-linearity. The batch norm parameters per layer, which include the mean μ_B and the variance σ_B are computed. The normalization is done by shifting each data-point with the mean and scaling it with the variance in every dimension of the data. A small constant ϵ is added to the variance to avoid division by zero errors.

$$\mu_B = \frac{1}{m} \sum_{i=1}^m x_i \quad (3.10)$$

$$\sigma_B = \sqrt{\frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2} \quad (3.11)$$

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad (3.12)$$

During test time the batch norm parameters, i.e. the mean and variance per layer, saved during the training phase are used.

Similar to MC dropout uncertainty estimates can be obtained using batch normalization by introducing stochasticity in the network during test time. However, unlike dropout the stochasticity is in the batch norm parameters and not in network weights. A valid mathematical formulation equivalent to MC dropout in Section 3.1.1 is also obtained by replacing the weights w with the batch norm parameters μ_B and σ_B .

In order to obtain uncertainty estimates at test time, T different batch norm parameters are required for T forward passes. These batch norm parameters are obtained by passing T different batches of training data through the network. As a result, to compute the uncertainty for a single test data, T different batches of training data have to be sampled and T forward passes of these sampled batches are required to get the batch norm parameters per batch [56]. It is computationally

expensive when this process has to be repeated for each data-point in the test set. Hence, an alternate option would be to store the batch norm parameters of T different training batches during training itself, and then later use them when required at test-time. This method performs at par with MC Dropout, however deteriorates when the batch size is less than 16, as batch norm itself proves to be ineffective in this case.

Another efficient method would be to construct an approximate distribution of the batch norm parameters μ_B , σ_B of each layer of the trained model. During test time for multiple forward passes any number of these parameters can be drawn by sampling from this distribution. Atanov et al. [4] suggests the use of a normal and a log normal distribution to approximate the mean μ_B and the variance σ_B respectively. This choice of distributions is able to fit real distributions of these parameters more closely and can be verified by the similarity between these distributions and the empirical distribution of μ_B , σ_B obtained from the training batches. Although, uncertainty estimates are obtained from the output predicted posterior, this method provides over-confident predictions as compared to MC dropout.

3.1.3 Test-time data augmentation

Data augmentation is a method used to increase the size of the training set when there are insufficient training samples. It involves applying spatial transformations such as rotation, flip, random crop, shear, brightness and contrast variations and so on to the training images. This allows the network to efficiently explore samples in the neighborhood of a particular training image. Uncertainty estimates can be obtained by applying data augmentation at test time. This would mean that several augmented versions of a single image is fed to the model for evaluation and the model prediction confidences for each augmented version is recorded. The entropy/variance of the predicted confidence values yields uncertainty estimates. Since the variations introduced are image specific, the uncertainty estimates more accurately describe the heteroscedastic aleatoric uncertainty [5], [59]. Mathematically, uncertainty estimation using test-time augmentation can be formulated using approximate Bayesian inference, similar to MC dropout [59]. The difference here is that the inference is performed over the parameters of the spatial transformation unlike MC

3.1. Approximate Bayesian Methods for Uncertainty Estimation

dropout where inference is over the network weights. Let us consider an input image X_0 to which the transformation operator T with parameters α is applied. The noise in the transformation process (for example, due to down-sampling or approximation errors) is modeled as a Gaussian noise τ .

$$X = T_\alpha(X_0) + \tau \quad (3.13)$$

The transformed input image X is given to a classifier $H(w, X)$ which is parameterized by the weights w . Thus, the posterior over the transformation parameter α and the noise parameter τ is given as follows.

$$P(\alpha, \tau | X_0, Y) = \frac{P(Y | X_0, \alpha, \tau) P(\alpha, \tau)}{P(Y | X_0)} \quad (3.14)$$

Using the posterior, inference is made at test-time on a sample image x_0^* and the prediction of the model is denoted by y^* .

$$P(y^* | x_0^*, X_0, Y) = \int P(y^* | x_0^*, \alpha, \tau) P(\alpha, \tau | X_0, Y) d\alpha d\tau \quad (3.15)$$

This type of Bayesian inference is practically impossible, since neither the posterior (in Eq. 3.14) nor the integral in Eq. 3.15 can be computed. This is because there are innumerable transformations that can be applied to an image, thereby causing α to take on infinite values. For this reason, approximate Bayesian inference techniques are used, where the integral is approximated by the taking empirical mean of the predictions \hat{y} over certain chosen transformations such as rotation, scaling, flipping depending on the dataset.

$$E(\hat{y}^* | x_0^*) = \frac{1}{N} \sum_{n=1}^N y_n^* \quad (3.16)$$

Eq. 3.16 computes the average per class prediction confidence of an image by averaging over the predicted confidences of N different augmentations of the same image. The classification decision is made by assigning the class label to the image which has the highest average per class prediction confidence.

Thus, the increasing the number of augmentations will result in an increase in uncertainty estimation time per sample image. Since there are no fixed number

and type of data augmentations and as they vary from dataset to dataset, it is a hyper-parameter that has to be deliberated upon leading to a trade-off between uncertainty estimation time and the quality of uncertainty estimate.

3.2 Non-Bayesian Methods for Uncertainty Estimation

3.2.1 Deep ensembles

Dropout can be interpreted as an ensemble model combination [54]. It yields predictions and uncertainty estimates by computing averages over multiple forward passes [14]. Each forward pass is characterized by a different dropout rate, thereby yielding models with different weights per forward pass. As a result, multiple forward passes using dropout at test-time are equivalent to an ensemble of models with different weights. Thus, a more direct solution is to use an ensemble of models to perform uncertainty estimation. Randomized ensembles refers to a collection of models where each model has the same architecture but a different combination of randomly initialized parameters and is trained with randomly sub-sampled data-points [34]. This yields a better performance than Bayesian Model Averaging (BMA) or boosting based approaches [34].

Beluch et al. [7] used the ensemble-based uncertainty estimation method as an acquisition function for Active Learning (AL). This involves initially training a deep-learning based classification model on some images from the training set and then evaluating its uncertainty over the remaining pool of images in the training set using an acquisition function. The acquisition function uses different measures such as entropy, mutual-information, and variation ratio to quantify the estimated uncertainty. The images with maximum uncertainty as evaluated by the acquisition function are chosen and added to the set of images with which the model is re-trained again. This process is repeated for certain acquisition steps and during each step the accuracy on the validation and test set is measured. The ensemble based acquisition function yields models with greater test set accuracy as compared to its Bayesian counterpart Monte-Carlo dropout. Typically, an ensemble with 3 models yields uncertainty estimates comparable to dropout, however, in general 5 models are used to get better estimates. The better performance of ensembles can be attributed to

three main factors:

- Ensembles operate with multiple networks rather than a single one in the case of Monte-Carlo dropout
- More stochasticity in the model since each network is initialized with different random weights and the training data is fed in a different order.
- Using dropout at test-time reduces the model capacity, whereas ensembles do not suffer from this problem.

3.2.2 Softmax calibration

The notion of calibration is closely tied with the concept of model interpretability [22], [20]. Calibrated models are good because when a classifier makes a prediction with some confidence which falls in a certain prediction interval, then the accuracy of the classifier also falls in the same interval. For instance, when a classifier makes a prediction with 90% confidence, then the accuracy of the classifier for that prediction can be also inferred as 90%, if it is calibrated. This allows us to verify the empirical validity of the model’s predictions over a small subset of the dataset (such as validation set or test set). Let us consider a classifier with $X \in \mathcal{X}$ belonging to one of the k classes in $Y = 1, 2, \dots, k$ and the classifier makes a prediction $\hat{Y} = y$ on the input with a confidence $\hat{p} = p$. Thus, for a calibrated classifier the predicted model confidence p will be equal to probability of correctness of the prediction (i.e., accuracy). This sufficient condition for calibration is realized using the following equation,

$$P(\hat{Y} = y | \hat{p} = p) = p, \quad \forall p \in [0, 1] \quad (3.17)$$

Calibration can be treated as an additional piece of information, which makes it possible for a human user to decide how accurate the model is on a particular prediction. Model calibration can be visualized using calibration curves or reliability diagrams. For a well calibrated classifier, the relationship between prediction confidence and model accuracy is linear and is referred to as the ideal calibration line. The calibration error is the amount of deviation of the classifier’s calibration curve from the ideal calibration line. In a reliability diagram as displayed in Fig.

3.1, model prediction confidence on different input samples are grouped into interval bins and plotted along with the model accuracy per bin.

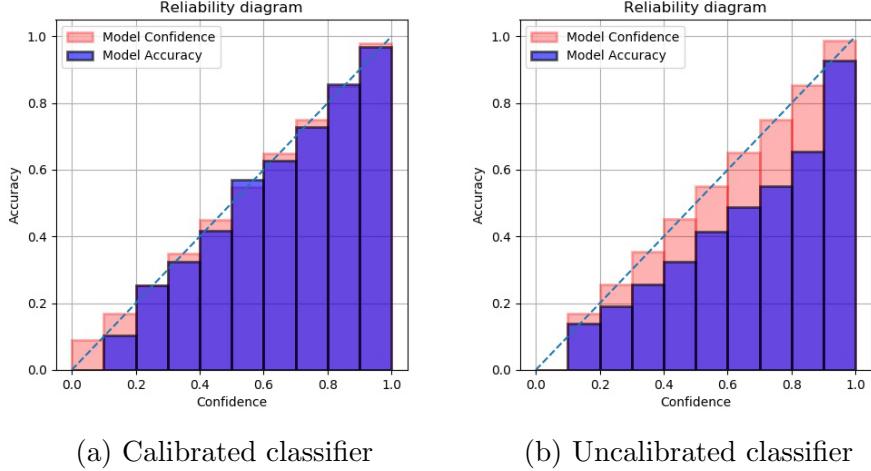


Figure 3.1: Reliability diagrams illustrating the difference between a calibrated classifier (Fig. 3.1a) and an uncalibrated classifier (Fig. 3.1b) with confidence and accuracy on the x and y axis respectively. Note the linear relationship between confidence and accuracy for a calibrated classifier (Fig. 3.1a).

There exists several classifier post-calibration techniques which make slight modifications to the predicted confidence values so that the classifier’s calibration curve is as close as possible to the ideal calibration line, i.e., resulting in minimal calibration error. Histogram binning involves separating the prediction confidence values into several bins of fixed size and assigning an optimal calibrated confidence value for each bin [61]. Isotonic regression involves optimizing both the bin size and the calibrated confidence value per bin. Thus, the calibrated confidence value per prediction is determined by the bin to which the prediction confidence value belongs [62]. Platt scaling is a calibration technique introduced in the context of SVM classifiers which employs a logistic regression model on top of the classifier [43]. Here, the input to the logistic regression model are the prediction probabilities from the classifier and the outputs are calibrated prediction probabilities. The parameters of the logistic regression model are optimized using the validation set.

In the context of deep learning based classification models, Guo et al. [22] introduced a simple temperature scaling method. These models are hierarchical

models where the output from one layer is fed as input to following layer. The last layer of the model has as many number of logits as the number of classes in the dataset, followed by a softmax linearity. The softmax function is responsible for rescaling the logit values into a number between 0 and 1, hence is often used to quantify per class model prediction confidence. Temperature scaling involves scaling the logit values from the last layer by a temperature constant T before they are fed to the softmax function. This has an effect of reducing the over-confident predictions previously made by softmax. Since all logit values are scaled by the same temperature constant, the peak of the softmax function is not affected. Hence, temperature scaling does not modify the model accuracy but prevents over-confident predictions.

3.2.3 Selective classification

Selective classification is the process in which a classifier has the ability to abstain from making predictions on certain instances of the dataset in order to reduce the risk of misclassification. The empirical risk of a selective classifier $f : \mathcal{X} \rightarrow \mathcal{Y}$ [17], which defines a mapping over the m data-points \mathcal{X} to the k class labels \mathcal{Y} is given as follows,

$$\hat{r}(f, g_\theta) = \frac{\frac{1}{m} \sum_{i=1}^m l(f(x_i), y_i) g_\theta(x_i)}{\frac{1}{m} \sum_{i=1}^m g_\theta(x_i)} \quad (3.18)$$

From Eq. (3.18) it is seen that the empirical risk of the selective classifier differs from the conventional classifier by the use of the selection function $g_\theta(x)$. The selection function is responsible for allowing the classifier to make a prediction on a particular data-point and therefore its value is either 0 or 1,

$$g_\theta(x) = \begin{cases} 1 & \text{if } \kappa_f(x) \geq \theta; \\ 0 & \text{otherwise.} \end{cases} \quad (3.19)$$

In the above equation, $\kappa_f(x)$ refers to the confidence function used to give the final prediction confidence on the input data-points. This allows to use different confidence functions from MC dropout, ensembles or any other method discussed above other than the softmax. The prediction threshold θ is used so that only those data-points

are considered for classification whose confidence score are above this threshold.

In order to obtain a low risk, it is important to have a good classifier f which does not have many misclassifications and a good threshold (θ) value so that the selection function $g_\theta(x)$ is able to reject the data-points that are misclassified by f . Geifman et al. [18] uses a combination of snapshots of the model from various training epochs rather than just the last model after training. This is because the last model does not provide reliable confidence estimates and is often over-confident in its predictions [13]. Geifman et al. [18] attributes this problem to the dynamics of the training process, where the model is able to produce high confidence scores for easy instances and low confidence scores for difficult instances during the early training epochs. However, as the training continues, the weights are further optimized so that the confidence estimates of the difficult instances are improved. At this point, the easy instances are ignored by the model since it only concentrates on the difficult instances, thereby making over-confident predictions on the easy instances. In order to address this issue, a fixed number of evenly spaced snapshots of intermediate models through the training epochs are taken and the confidence function κ_f is the average of these snapshots. An appropriate value for θ which gives the least risk for maximum coverage can be inferred from the Risk-Coverage (RC) curves where the risk and coverage are given by the numerator and denominator of Eq. (3.18).

Selective classification is able to improve the performance of other uncertainty estimation methods such as MC dropout [14], and deep ensemble [34]. However, it comes at an additional cost of requiring to save intermediate models during training as well as increased inference time because predictions from all these models are required for a test data-point.

3.2. Non-Bayesian Methods for Uncertainty Estimation

4

Methodology

This chapter describes the experimental setup used to assess some of the uncertainty estimation methods discussed in the previous chapter. The different model architectures and datasets used to evaluate the uncertainty estimates produced by these methods are described in detail here.

4.1 Datasets

The evaluation procedure consists of experimentation with standard image classification datasets such as CIFAR-10 and CIFAR-100 [32] followed by two other custom datasets. CIFAR-10 is a balanced dataset consisting of 60000 low resolution images belonging to 10 different classes. Each image is of size $32 \times 32 \times 3$ and there are 6000 images per class. The images are by default split into a training set with 50000 images and a test set with 10000 images. We further make a training-validation split with a validation set consisting of 5000 images and the training set reduced to 45000 images. CIFAR-100 is also a balanced dataset, however it has 60000 high resolution images belonging to 100 different classes, with each class consisting of 600 images. The image dimensions and the train-validation-test split are similar to CIFAR-10. Evaluation on the CIFAR-100 dataset is important because the images are very similar to many real world datasets and it is highly likely that an algorithm performing well on this dataset will produce similar results on other real world datasets. The CIFAR-10 dataset, on the other hand is more akin to a toy dataset which is used for preliminary quick evaluation of algorithms.



(a) CIFAR-10 truck (b) CIFAR-100 pickup van

Figure 4.1: Fig. 4.1a showing more pixelated image CIFAR-10 image than a CIFAR-100 image in Fig. 4.1b

For evaluating the performance of uncertainty estimation methods on real world problems, we use the Bosch optical inspection dataset which consists of 62814 gray-scale images belonging to 2 classes -OK and NOK- indicating the success/failure of the manufacturing process. The images in the dataset are scaled down to a size of 128×128 . The dataset is highly imbalanced as there are only 3232 images which belong to the NOK class. Class balancing is done while training the model. The training set consists of 50246 images, while the validation and test set consists of 6281 and 6287 images.

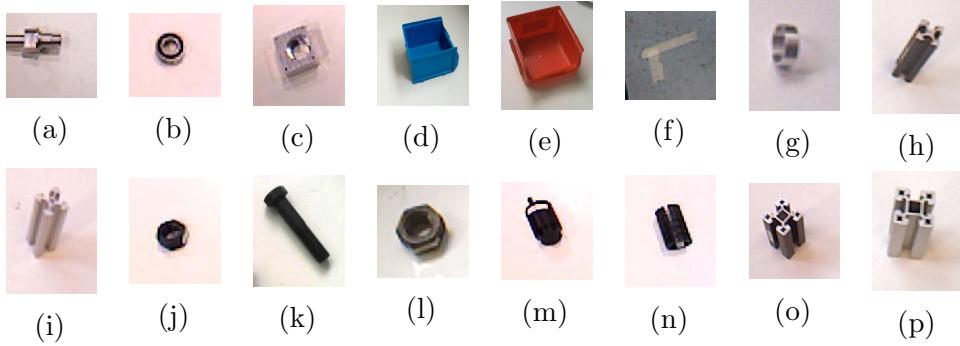


Figure 4.2: Sample image from each of the 16 classes in the RoboCup@Work dataset [38].

Additional evaluation is also performed on RoboCup@Work dataset consisting of images of tools used in the RoboCup@Work League [31]. The objects belong to 16 different classes as shown in Fig. 4.2, and out of the total 21706 images, 17358 are used for training, 2170 for validation, and 2178 for testing.

4.2 Models

Convolutional Neural Networks (CNN) have enjoyed great success after their superior performance in ImageNet Large Scale Visual Recognition Competition (ILSVRC) [49]. While a major factor for the success of CNNs is the availability of powerful hardware resources such as GPU's, it is also important to note that new model architectures and ideas have boosted their performance. The different uncertainty estimation methods are evaluated on three popular standard deep learning architectures - VGG-16 [53] model, Wide-ResNet model [63], and Xception model [10]. The first two are used for classifying the standard CIFAR-10 and CIFAR-100 datasets, while the Xception model is used for the optical inspection and RoboCup@Work dataset. The weights of all layers of all three models are initialized using Glorot initialization [19].

The VGG-16 model consists of 16 convolutional layers and 3 fully connected layers as shown in Fig. 4.3. The dropout layers are placed in between the convolutional layers (in short conv layers) and after the dense layers, having a dropout probability of 0.2 and 0.5, respectively. The model is trained for 160 epochs with a batch size of 128 using Stochastic Gradient Descent with Nesterov momentum as an optimizer with a learning rate and momentum of 0.01 and 0.9 respectively. The learning rate is decayed by a factor of 10 after 50, 80 and 110 epochs. Minor changes in those parameters are made in order to use the same architecture for evaluating the various uncertainty estimation.

The Wide-ResNet-28-8 model as shown in Fig. 4.4 is used for classification of the CIFAR-10 and CIFAR-100 datasets with the same parameters. It has 4 convolutional blocks, where each block consists of convolutional layers followed by relu non-linearity, dropout layers, and a residual connection connecting the input and output of the block. The architecture is more memory efficient as it has fewer but wider layers compared to other architectures such as Inception and ResNet, and it also retains the benefits of residual connections. The total number of convolutional layers in the model, including the average global pooling and output layer is 28, while the 3x3x16 kernels are expanded by a width factor of 8. The model is able to achieve a higher accuracy compared to the VGG-16 model because of the presence of wider kernels. It uses a dropout rate of 0.3 and weight decay of 0.0005. The initial convolutional layer

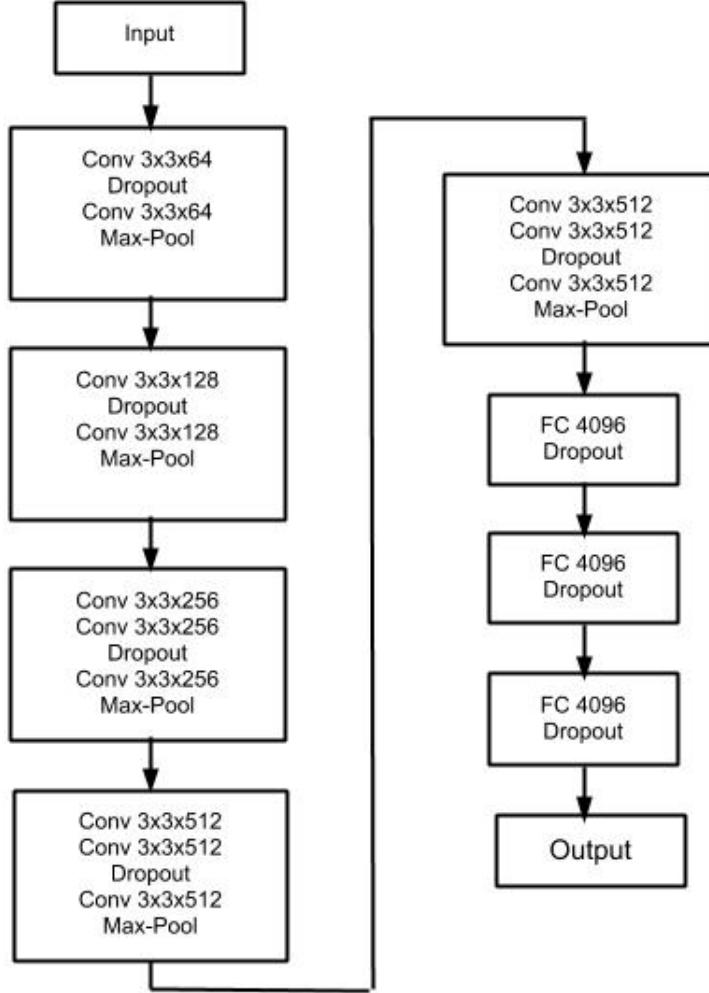
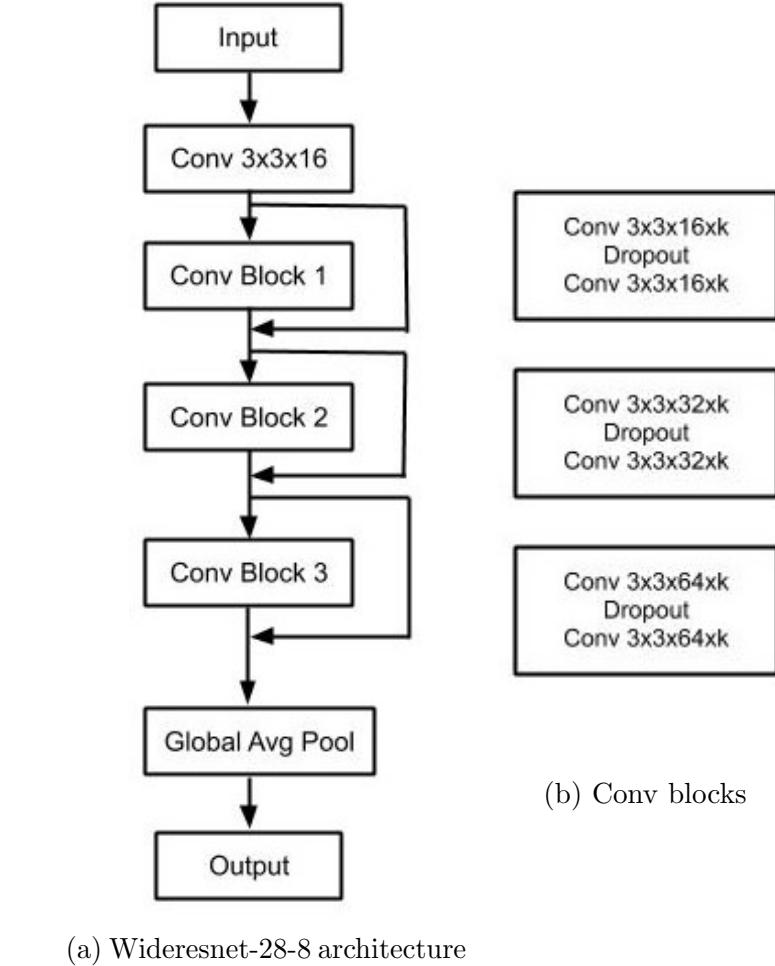


Figure 4.3: VGG-16 architecture used for classifying images in CIFAR-10 and CIFAR-100. The conv layers in the architecture diagram represent the combination of convolution layer, batch-norm layer and relu non-linearity.

is followed by three convolutional blocks, where each contains 4 pairs of convolutional layers. The model is trained with Adam and a learning rate of 0.001 with a batch size of 128 for 110 epochs. The learning rate is decayed to 0.0001 and 0.00001 after 50 and 80 epochs.

The Xception architecture consists of point-wise 1×1 convolutions followed by spatial 3×3 convolution. It is an extreme version of the Inception module [55] with the cross-channels correlations and the spatial correlations are entirely decoupled. A miniature version of the Xception architecture in contrast to the full architecture



(a) Wideresnet-28-8 architecture

(b) Conv blocks

Figure 4.4: Fig. 4.4b expands the conv blocks used in Fig. 4.4a. The width factor k is 8 and N is 4 in Fig. 4.4b. Each conv layer is a combination of convolution layer, batch-norm layer and relu non-linearity.

presented in Chollet [10] is used to solve our classification problem on the optical inspection dataset. The model consists of 3 Xception blocks with each Xception block containing 3 separable convolutional layers with intermediate relu non-linearities and dropout layers as shown in Fig. 4.5. The separable convolutional layers constitute the combination of point-wise 1×1 convolutions followed by depthwise 3×3 convolutions. The model uses a dropout rate of 0.4 and weight decay of 0.0005. The model is trained for 3 epochs with a batch size of 64 using Adam as an optimizer with a learning rate of 0.0001. The RoboCup@Work dataset uses the same model but with

minor changes in the following parameters. Stochastic Gradient Descent is used for training the model for 100 epochs with a learning rate of 0.01 and a Nesterov momentum of 0.9. Learning rate decay is used to reduce the learning rate to 0.0001 and 0.00001 after 50 and 80 epochs. All models are trained on one Nvidia Geforce GTX 1080 GPU except for the optical inspection dataset which requires two GPUs due to larger images of size 128×128 .

The choice of these architectures is primarily based on the presence of dropout layers in each of these models because uncertainty estimation using the MC dropout (MCD) method requires the presence of dropout layers in the model. The ensemble method (EN) uses the above described models with exactly the same parameters. On the other hand, the dropout layers are turned on during test time for MC dropout (MCD). Similarly for test time augmentation (TTA), the same model is evaluated but with several augmented test set images. Several meaningful augmentations are drawn from a set of image augmentations such as flipping, rotating, cropping, shifting, and brightness and contrast variations. The aleatoric uncertainty estimation method (MCDA) requires substantial changes to the model architecture and loss function. The last output layer of the model is split into two nodes - one for per class prediction confidence and the other for aleatoric uncertainty estimate as shown in the Fig. 4.6. The loss function used in [29] jointly optimizes the parameters of the model so as to produce proper uncertainty estimates along with class confidence scores.

A final note on this chapter is that all implementations of the above architectures and methods are done using a python Keras framework with Tensorflow backend.

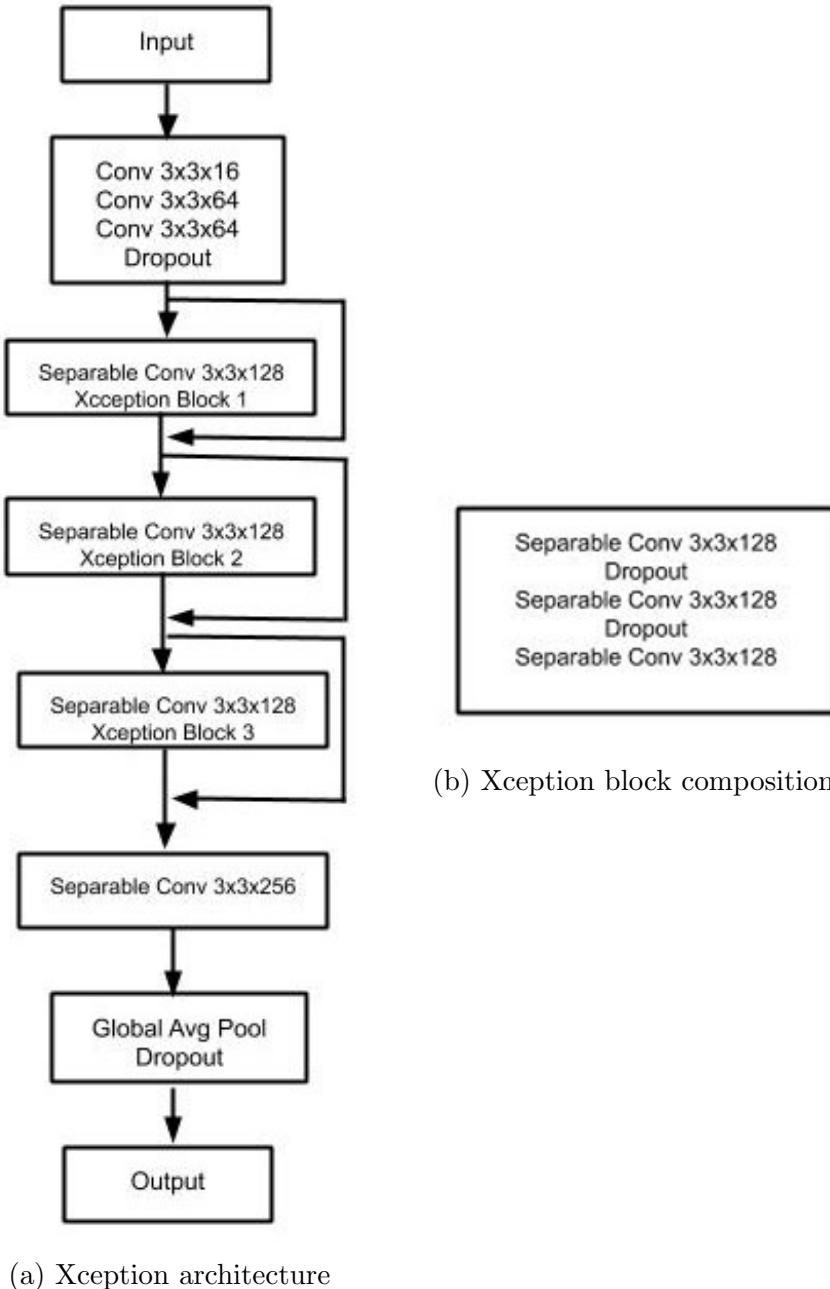


Figure 4.5: Fig. 4.5b expands the Xception blocks used in Fig. 4.4a. All Xception blocks have convolutional and dropout layers with the same parameters. Each conv layer is a combination of convolution layer, batch-norm layer and relu non-linearity.

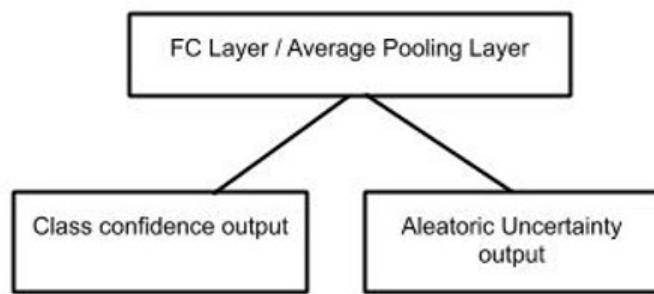


Figure 4.6: Second last layer of the models (Fully connected layer in case of VGG and average pooling layer in the case of Wideresnet and Xception) branches out into two nodes

5

Experimental Evaluation

This chapter focuses on the evaluation of different uncertainty estimation methods - MC dropout (MCD) [14], deep ensembles (EN)[34], test time data augmentation (TTA) [5], the combination of MC dropout and test time data augmentation (MCD+TTA) [59] and MC dropout with Aleatoric loss (MCDA) [29] - used to address the overconfidence problem. These methods are applied to the various models and datasets discussed in the previous chapter. A per class output distribution of the softmax confidence is produced by these methods rather than a single point estimate for each test image. The resulting classifiers are evaluated on three aspects,

- classifier performance
- calibration
- sharpness

We aim to find the best uncertainty estimation method that is able to improve or maintain the classifier performance, and capture predictive uncertainty in a significant number of test set samples, while maximizing calibration.

5.1 Metrics

5.1.1 Sharpness Metrics

Uncertainty estimation techniques are able to reduce the sharpness of the overconfident predictions by producing a predictive distribution instead of point estimates. A predictive distribution is obtained for each image in the test set from the N forward passes (in MCD or TTA) or the N members of the ensemble. The sharpness of the predictive distribution is inversely proportional to produced uncertainty. The mean softmax confidence per class is computed from the predictive distribution and is used in the metric computation. The following metrics are used to measure the spread of the predictive distributions (for more details on the following metrics refer Chap. 2 of Murphy [40]), thereby directly yielding the uncertainty estimates per prediction made by the classifier.

1. **Entropy** : Entropy in general is used to measure the disorder/uncertainty of a random variable with some probability distribution [51]. The entropy $H[\hat{y}|x, w]$ of the predicted softmax confidence \hat{y} with a probability distribution p from a model with weights w for input image x , over each of the N forward passes or N models in the ensemble [7] is represented as follows,

$$H[\hat{y}|x, D_{train}] = - \sum_c \left(\frac{1}{N} \sum_n p(\hat{y} = c|x, w) \log \left(\frac{1}{N} \sum_n p(\hat{y} = c|x, w) \right) \right) \quad (5.1)$$

Here, the number of realizations of the predictions \hat{y} is equal to the number of classes c .

2. **Cross-Entropy** : Another closely related measure is the cross-entropy. It measures the difference between the true distribution of the random variable and another distribution which approximates the true distribution. The approximating distribution is given by the predictive distribution consisting of predicted confidences \hat{y} for all N forward passes/models in the ensemble, while the true class labels y are represented using the indicator variable $\mathbb{1}(y == c)$

for class c [7].

$$H[y, \hat{y}|x, D_{train}] = - \sum_c \left(\frac{1}{N} \sum_n \mathbb{1}(y == c) \log \left(\frac{1}{N} \sum_n p(\hat{y} = c|x, w) \right) \right) \quad (5.2)$$

It is also popularly called as the negative log loss in machine learning literature and is used to measure the performance of a classifier.

3. **Mutual Information** : Mutual information measures how much one random variable depends on another random variable. The mutual information between the predicted labels \hat{y} and the stochastic variable s for each of the methods, such as the weights of the network for MCD [16], the data augmentations applied during test time for TTA or the models of the models with different weights for EN, gives the measure of uncertainty. The greater the mutual information between the predicted labels and the stochastic variable, the higher the uncertainty.

$$I[\hat{y}, s|x, D_{train}] = H[\hat{y}|x, D_{train}] - \frac{1}{N} \sum_n \sum_c (-p(\hat{y} = c|x, s) \cdot \log(p(\hat{y} = c|x, s))) \quad (5.3)$$

4. **Variance** : Variance is another measure which is used to estimate the spread of predicted confidences from each forward pass/models in the ensemble. The per class prediction variance is given by the squared absolute difference between each per class predicted confidence $p(\hat{y} = c|x, w)$ and the mean of all per class predictions from the N forward passes/models in the ensemble [16].

$$\sigma = \frac{1}{N} \sum_n ((p(\hat{y} = c|x, w)) - \frac{1}{N} \sum_n p(\hat{y} = c|x, w))^2 \quad (5.4)$$

5.1.2 Classifier Performance Metrics

Classifier performance metrics are used to evaluate the performance of the classifier in correctly classifying samples from the test set (according to their true labels), when using uncertainty estimation methods. They measure the ability of the classifier to discriminate between samples from various classes.

- **Accuracy** : Accuracy is defined as the ratio of number of correct predictions to the total number of predictions [40]. It denotes the level of agreement between the predictions and the true labels. A difference between the model predictions and the true label results in a misclassification. Accuracy is representative of the misclassification rate of the model, i.e., lower the misclassification rate, the greater the accuracy. In our context, accuracy can be used to select an uncertainty estimation method which yields the lowest misclassification rate.

$$Acc = \frac{\text{No. of correct predictions}}{\text{Total no. of predictions}} \quad (5.5)$$

- **Excess-AURC** : AURC represents the Area Under the Risk Coverage curves [18]. The risk $\hat{r}(f, g_\theta)$ of a classifier f for a given coverage can be calculated according to Eq. 3.18 in Chap. 3. Eq. 5.6 and Eq. 5.7 are used to calculate and compare the AURC curves for the classifier using an uncertainty estimation method $AURC(\kappa, f|V_n)$ and the baseline classifier $AURC(\kappa^*, f|V_n)$ respectively, by varying the coverage of the test set V_n with n data-points. The baseline classifier refers to a single trained deep learning model without any uncertainty estimation method applied to it.

$$AURC(\kappa, f|V_n) = \frac{\hat{r}(f, g_\theta)|V_n}{n} \quad (5.6)$$

$$AURC(\kappa^*, f|V_n) = \frac{1}{n} \sum_{i=1}^{\hat{r}n} \frac{i}{n(1 - \hat{r}(f, g_\theta) + i)} \quad (5.7)$$

$$E\text{-AURC} = AURC(\kappa^*, f|V_n) - AURC(\kappa, f|V_n) \quad (5.8)$$

Excess-AURC is the difference between the AURC of the baseline classifier and that of the classifier using the uncertainty estimation method [18]. This metric is a unit-less measure which gives a bounded value between 0 and 1.

5.1.3 Calibration Metrics

Calibration metrics assess the consistency of the classifier in producing reliable predictions. The degree of linearity between the predictive confidence and accuracy is directly proportional to calibration quality. Calibration metrics capture the deviations between accuracy and the predicted confidence, when they do not follow a linear relationship. The following metrics will be used to assess the calibration of the predictions produced by the deep learning classifier using the uncertainty estimation techniques.

1. **Expected Calibration Error (ECE)** : It is the expected difference between the model accuracy and the predictive confidence $p(\hat{y} = c|x, w)$ (for the true class c of the sample) [22]. This can also be visualized in the reliability diagrams as the gap between the model's calibration curve and the ideal calibration line (Fig. 5.8).

$$ECE = \mathbb{E}(P(\hat{y} = c|p(\hat{y} = c|x, w)) - p(\hat{y} = c|x, w)) \quad (5.9)$$

In the above equation, the first term denotes the frequency of correct predictions made by the model (accuracy).

2. **Maximal Calibration Error (MCE)** : Maximal Calibration Error represents the maximum deviation between the model's accuracy and the predictive confidence [22].

$$MCE = \max(P(\hat{y} = c|p(\hat{y} = c|x, w)) - p(\hat{y} = c|x, w)) \quad (5.10)$$

3. **Brier score** : Brier score measures the closeness of the model's predictive probability to true class probability (which is always 1). A good Brier score value is very close to zero as the squared difference between the true class probability ($\mathbb{1}(y == c)$) and the predicted confidence ($p(\hat{y} = c|x, w)$) is less. It

is a measure of both the classifier's discriminative power and calibration [3].

$$BS = \frac{1}{N} \sum_{t=1}^N (p(\hat{y} = c|x, w) - \mathbb{1}(y == c))^2 \quad (5.11)$$

5.2 Comparison Based on Uncertainty Quality

This experiment consists of four parts as illustrated in Fig. 5.1. Section 5.2.1 involves identifying the metric that is able to most distinctly represent the uncertainty in the predictions. Section 5.2.2, on the other hand consists of utilizing the identified metric to find out the best method that is able to reduce the sharpness of the deep learning classifier's overconfident predictions. Section 5.2.3 aims to find the best method that identifies the maximum number of misclassifications as uncertain. Finally, Section 5.2.4 provides the results of evaluating the models with uncertainty estimation methods on data which does not belong to training distribution, called Out Of Distribution (OOD) datasets.

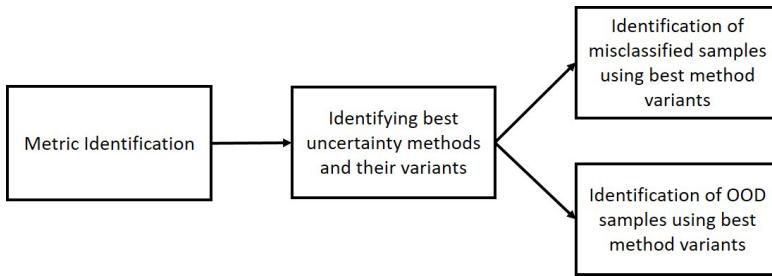


Figure 5.1: Workflow for comparative evaluation of different uncertainty estimation methods based on uncertainty quality

5.2.1 Metric identification

The uncertainty in the predictions is estimated using the various sharpness metrics described in Section 5.1.1. The aim of this experiment is to identify the best metric that is able to represent a greater spread given a particular uncertainty estimation method, dataset, and model. In order to evaluate the quality of these metrics, we utilize the concept of rank histograms, a diagnostic tool which is used to assess the

spread of ensemble predictions in weather forecasting [45]. Originally, rank histograms consists of intervals/bins of sorted predictions from different ensemble members along the x-axis and the relative frequency of an observation falling into the bins on the y-axis. The reliability of the ensemble forecast is verified by repeatedly conforming if the real observations fall into one of the bins. If all bins contain approximately an equal number of predictions then the rank histogram is flat, thereby indicating the reliability of the ensemble forecast. On the other hand, deviations from a flat rank histogram are indicative of too much bias or variance exhibited by the ensemble [23].

Rank histograms are adapted to our context of identifying uncertainty metrics by retaining their structure while giving them a new interpretation. We use the actual structure of rank histograms by ordering the model’s predictions into 100 bins with the x-axis scaled by the uncertainty metrics in ascending order, and the number of samples in the test set that fall into these bins on the y-axis. The rank histograms as shown in Fig. 5.2 are highly right skewed with most of the samples having close to zero values for all the uncertainty metrics. This can be explained by the fact that deep learning models produce highly over-confident predictions which provide very small values of uncertainty. Since our aim is to identify the uncertain predictions made by the model, a particular threshold value is chosen and samples that fall beyond this threshold are considered to be uncertain. The choice of 99 percentile area is made as compared to 97.5 or 95 percentile here because the distribution is highly right skewed and most of the samples are concentrated close to zero for all four metrics. However, it is important to note that the shape of the rank histograms is highly dependent on the model and dataset and does not exhibit the right skewness always.

Rank histograms can be plotted for all methods evaluated on all datasets with one of the four sharpness metrics on the x-axis and the number of test set samples on the y-axis. Fig. 5.2 shows the rank histograms plotted for the predictions from the VGG-16 used to classify images in CIFAR-10 test set with TTA as the uncertainty estimation method. The number of test samples that fall in the last 99 percentile area (i.e., the shaded region in Fig. 5.2) are counted for different metrics. The greater the number of samples that fall in the shaded region of the histogram, the better is the metric in capturing the spread of the distribution.

The following observations are made from Fig. 5.2.

5.2. Comparison Based on Uncertainty Quality

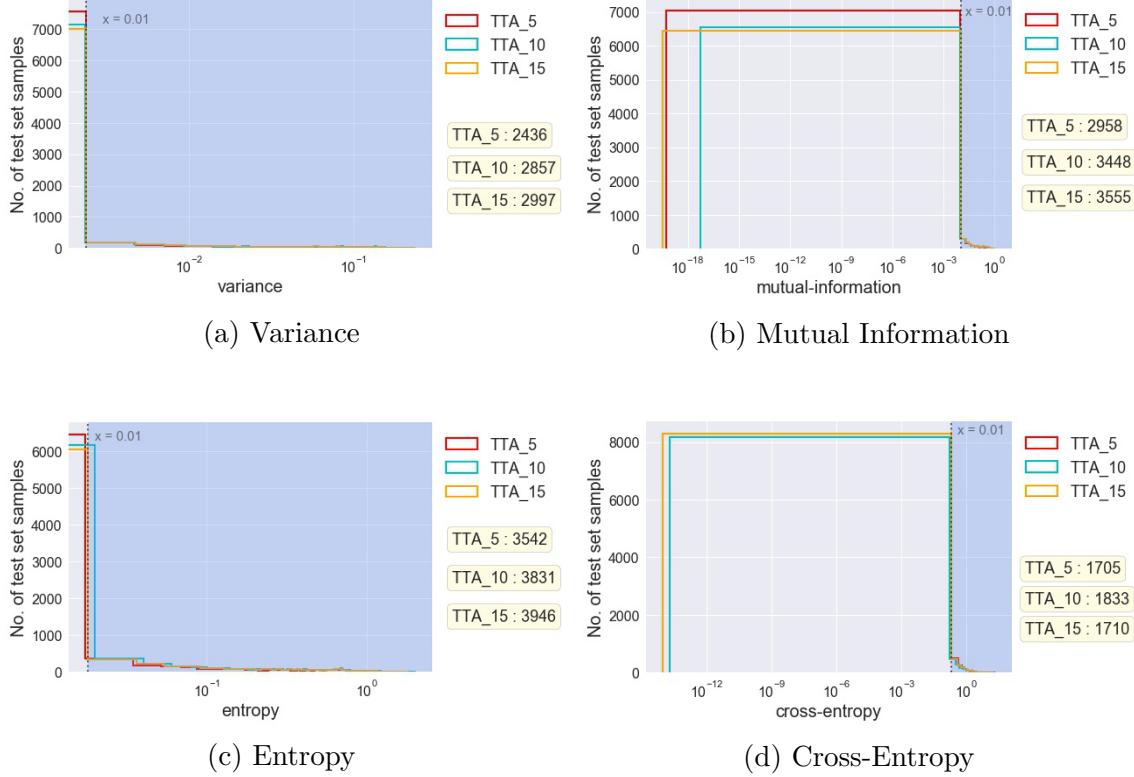


Figure 5.2: Rank histograms plotted for different sharpness metrics using the predictions from TTA evaluated on CIFAR-10 dataset and VGG-16 classifier. The dotted line denotes the 1 percentile line, beyond which lies in the 99 percentile area (shaded region) of the distribution. The number of samples that fall in the shaded region for different TTA variants are given beside the plot. The x-axis is plotted in log-scale to better observe the difference in the peaks.

- TTA with 15 forward passes has a greater spread characterized by the lowest peak around 0 for all metrics except cross-entropy as compared to its other variants.
- Cross-entropy does not provide a consistent measure of the spread as the number of samples do not follow any incremental or decremental pattern.
- Entropy has the highest number of samples in the shaded region for CIFAR-10 dataset classified by the VGG-16 model using different variants of TTA.

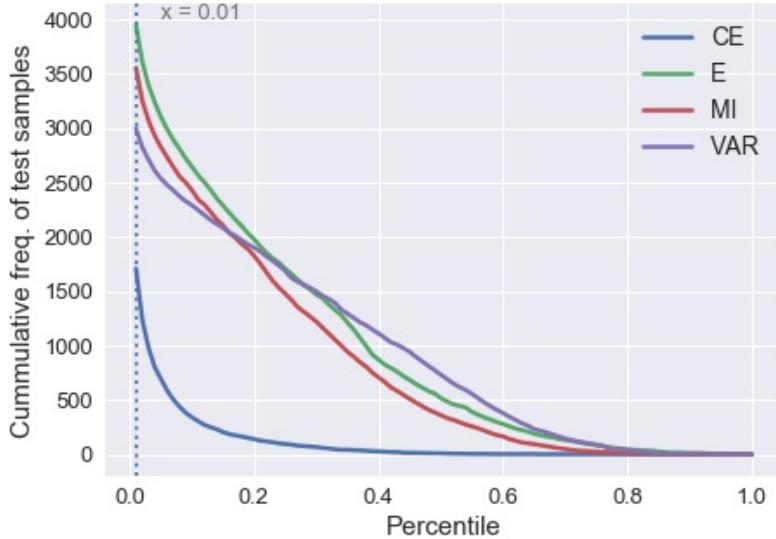


Figure 5.3: Cumulative frequency plot for different metrics showing the number of samples that fall beyond the given percentiles in the rank histogram for TTA_15 on CIFAR-10 with VGG-16 classifier. The values at the intersection of the curves and the 0.01 percentile line in this figure are the same as those listed in Fig. 5.2 for TTA_15. (The legends are abbreviated as follows - CE:Cross Entropy, E:Entropy, MI:Mutual Information, VAR:Variance)

A more distinct representation is given by the cumulative plot in Fig. 5.3, which directly provides a count of the number of samples that fall beyond different percentiles in the rank histogram. It provides a more lucid discrimination between the various metrics as compared to the rank histograms, while representing the same values listed in the Fig. 5.2.

5.2. Comparison Based on Uncertainty Quality

	Entropy	Cross-Entropy	Mutual Information	Variance
MCD-5	2628	1368	2198	1705
MCD-10	2736	1343	2486	1943
MCD-15	2811	1326	2622	2070
EN-2	2549	1317	1820	1363
EN-3	2740	1364	2148	1743
EN-4	2860	1371	2363	1897
EN-5	2942	1346	2514	2027
MCDA-5	2736	1340	2251	1727
MCDA-10	2831	1371	2426	1941
MCDA-15	2898	1392	2637	2094
TTA-5	3542	1705	2958	2436
TTA-10	3831	1833	3448	2857
TTA-15	3946	1710	3555	2997
MCD+TTA-5	3738	2010	3191	2686
MCD+TTA-10	4106	1944	3699	3195
MCD+TTA-15	4323	2109	3941	3428

Table 5.1: Number of test set samples in the last 99 percentile for CIFAR-10 + VGG-16 model

Similar histograms and cumulative plots can be drawn for other methods and are indeed very insightful in depicting the entire distribution of the samples for different metrics. However, our objective is to find the metric which yields the greatest spread, i.e., the greatest number of samples that fall in the last 99 percentile area for different combinations of uncertainty estimation methods, datasets and models. These numbers are more compactly represented in Tables 5.1, 5.2, 5.3, 5.4, 5.5, 5.6 from which conclusions can be drawn.

The highlighted values in the table denote the metric and the method variant which has the highest number of samples that fall in the last 99 percentile of their rank histogram when plotted. Cross-entropy exhibits a similar inconsistent behavior for all datasets and models. In all cases, entropy displays a greater spread for all methods and their variants. Therefore, from this experiment it can be concluded that entropy is better able to capture the dispersion in the predictions as compared to all other metrics.

	Entropy	Cross-Entropy	Mutual Information	Variance
MCD-5	1866	821	1473	1106
MCD-10	1971	829	1701	1263
MCD-15	1983	843	1755	1345
EN-2	1819	849	1274	901
EN-3	1970	894	1460	1172
EN-4	2047	828	1667	1290
EN-5	2142	836	1819	1426
MCDA-5	3780	1493	2900	2153
MCDA-10	3862	1468	3291	2424
MCDA-15	3892	1520	3390	2580
TTA-5	2565	1073	2141	1727
TTA-10	2813	1056	2490	2004
TTA-15	2924	1076	2678	2194
MCD+TTA-5	2772	1138	2379	1915
MCD+TTA-10	3099	1119	2710	2281
MCD+TTA-15	3254	1177	2960	2489

Table 5.2: Number of test set samples in the last 99 percentile for CIFAR-10 + Wide-Resnet model

5.2.2 Method identification

The identification of the best uncertainty estimation method is accomplished by comparing the different methods and their variants using entropy (the best performing metric from the previous section) to evaluate the uncertainty quality. The first column of the Tables 5.1, 5.2, 5.3, 5.4, 5.5, 5.6 provides the number of samples that fall in the last 99 percentile area when using entropy as the uncertainty metric. The cumulative frequency plots introduced in the previous section are used here to compare the different methods. The best variants of each method as highlighted in the table are plotted in Fig. 5.4.

The following conclusions are drawn from Fig. 5.4.

- The combination of MC dropout and test time augmentation (MCD+TTA) indicates uncertainty in a greater number of samples for all cases except CIFAR-10 with Wide-ResNet (Fig. 5.4b), where it is less than MCDA by 638 samples (as seen from Table 5.2). This result can be explained by the

5.2. Comparison Based on Uncertainty Quality

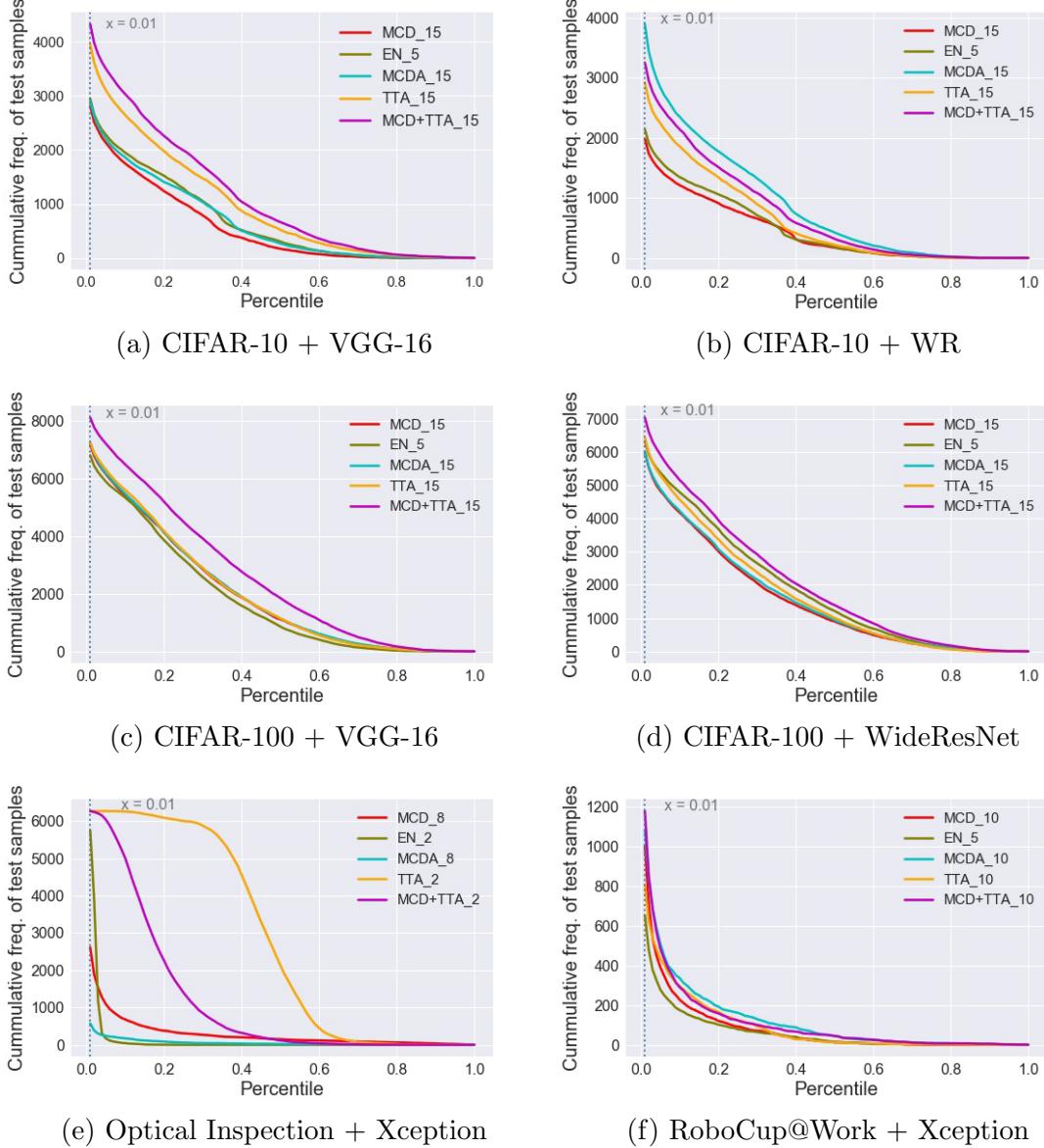


Figure 5.4: Comparison of the best variants of different uncertainty estimation methods on different datasets using cumulative frequency plots with entropy as the metric. The best variants of the different uncertainty estimation methods are obtained from the entropy scores values in Tables 5.1, 5.2, 5.3, 5.4, 5.5, 5.6.

	Entropy	Cross-Entropy	Mutual Information	Variance
MCD-5	6928	5032	6424	5688
MCD-10	7072	5010	6812	6118
MCD-15	7155	5033	6952	6303
EN-2	6231	4460	5515	4470
EN-3	6590	4807	6019	5392
EN-4	6725	4870	6199	5682
EN-5	6794	4785	6365	5871
MCDA-5	6924	4916	6504	5737
MCDA-10	7163	5055	6841	6176
MCDA-15	7241	4961	7055	6357
TTA-5	6937	4858	6478	5901
TTA-10	7121	4938	6813	6295
TTA-15	7219	4899	6950	6508
MCD+TTA-5	7720	5542	7237	6730
MCD+TTA-10	7960	5869	7716	7230
MCD+TTA-15	8101	5886	7866	7456

Table 5.3: Number of test set samples in the last 99 percentile for CIFAR-100 + VGG-16 model

fact that MCD+TTA and MCDA combine both the epistemic and aleatoric uncertainty. The epistemic uncertainty estimate is obtained from dropout, while the aleatoric uncertainty estimate is obtained from TTA (in the case of MCD+TTA) or the aleatoric loss (in the case of MCDA).

- The second best method yielding a higher number of uncertain samples following MCD+TTA is Test time data augmentation (TTA). TTA provides an estimate of only the aleatoric uncertainty in the dataset (Section 3.1.3) and not the epistemic uncertainty. This signifies that even in the absence of an epistemic uncertainty estimate, aleatoric uncertainty alone is able to identify the uncertainty in greater than 80% of the total uncertain classifier predictions obtained from either MCD+TTA or MCDA. However, the reverse is not true as MCD mostly performs worse than TTA.

	Entropy	Cross-Entropy	Mutual Information	Variance
MCD-5	5813	4050	5284	4396
MCD-10	5943	4037	5605	4717
MCD-15	5979	4085	5708	4863
EN-2	5766	4155	5073	3894
EN-3	6004	4275	5420	4659
EN-4	6208	4344	5681	4981
EN-5	6308	4393	5833	5226
MCDA-5	5877	3932	5314	4431
MCDA-10	5942	3931	5614	4722
MCDA-15	6015	3951	5713	4844
TTA-5	6235	4333	5631	4827
TTA-10	6326	4388	5900	5225
TTA-15	6439	4430	6026	5337
MCD+TTA-5	7720	5542	7237	6730
MCD+TTA-10	7960	5869	7716	7230
MCD+TTA-15	8101	5886	7866	7456

Table 5.4: Number of test set samples in the last 99 percentile for CIFAR-100 + Wide-Resnet model

5.2.3 Identification of misclassified samples

The number of misclassified samples that fall inside the last 99 percentile area of the rank histograms of the different variants of uncertainty estimation methods are calculated. The main objective of this experiment is to investigate whether the uncertainty estimation methods and the metrics discussed in the previous sections are able to identify misclassifications made by the classifier. Hence, we ignore the correct predictions which fall under the 99 percentile area in contrast to the previous section. The number of misclassified samples identified is proportional to the number of uncertain samples identified in Tables 5.1, 5.2, 5.3, 5.4, 5.5, 5.6. Therefore, the best variants from these tables are also able to find the maximum number of misclassified samples. Table 5.7 shows the percentage of misclassified samples identified by the best variants (as per Fig. 5.4) of different uncertainty estimation methods. From Table 5.7 it is seen that MCD+TTA is also able to identify the maximum number of misclassified samples apart from indicating uncertainty in maximum number of

	Entropy	Cross-Entropy	Mutual Information	Variance
MCD-2	2480	168	414	173
MCD-4	2567	200	722	285
MCD-6	2613	179	707	194
MCD-8	2614	185	801	314
MCD-10	2608	181	941	364
EN-2	5820	328	185	3
EN-3	4972	342	1	33
EN-4	3863	357	8	68
EN-5	4501	654	737	392
MCDA-2	1078	632	185	56
MCDA-4	1055	634	263	85
MCDA-6	1056	633	337	100
MCDA-8	6141	1067	5497	3032
MCDA-10	6139	988	5609	3082
TTA-2	6286	326	3679	3683
TTA-4	6286	326	5940	5941
TTA-6	6285	326	6211	6211
TTA-8	6285	326	6257	6257
TTA-10	6285	326	6282	6282
MCD+TTA-2	6286	326	3679	3683
MCD+TTA-4	6286	326	5940	5941
MCD+TTA-6	6285	326	6211	6211
MCD+TTA-8	6285	326	6257	6257
MCD+TTA-10	6285	326	6282	6282

Table 5.5: Number of test set samples in the last 99 percentile for the optical inspection dataset

5.2. Comparison Based on Uncertainty Quality

	Entropy	Cross-Entropy	Mutual Information	Variance
MCD-2	908	132	338	112
MCD-4	910	110	553	185
MCD-6	983	104	684	240
MCD-8	940	116	666	244
MCD-10	999	101	784	256
EN-2	584	79	228	84
EN-3	606	96	333	135
EN-4	637	102	384	155
EN-5	652	110	423	180
MCDA-2	966	151	320	145
MCDA-4	1064	136	543	279
MCDA-6	1048	145	599	307
MCDA-8	1096	133	708	348
MCDA-10	1080	128	699	320
TTA-2	758	178	305	161
TTA-4	791	178	458	248
TTA-6	792	193	484	249
TTA-8	804	199	534	280
TTA-10	805	196	535	298
MCD+TTA-2	930	100	339	132
MCD+TTA-4	979	111	548	190
MCD+TTA-6	1090	104	599	253
MCD+TTA-8	1149	102	627	278
MCD+TTA-10	1176	139	718	249

Table 5.6: Number of test set samples in the last 99 percentile for the RoboCup@Work dataset

predictions.

	MCD	EN	MCDA	TTA	MCD+TTA
CIFAR-10 (VGG-16)	90.59	94.09	92.5	97.15	97.77
CIFAR-10 (Wide-ResNet)	90.13	92.02	96.60	95.11	96.40
CIFAR-100 (VGG-16)	98.49	98.61	98.84	98.93	99.54
CIFAR-100 (Wide-ResNet)	95.84	97.38	98.84	98.93	99.54
Optical Inspection (Xception)	67.86	97.54	41.06	100	99.69
RoboCup@Work (Xception)	87.5	100	100	100	100

Table 5.7: Percentage of misclassified samples identified by the best variants of different uncertainty estimation methods.

Uncertainty estimation workflow for identification of misclassified data

The workflow of a deep learning model with an uncertainty estimation module is presented in Fig. 5.5. A deep learning model with a desired uncertainty estimation method is chosen from our pool of five methods described in the beginning of Chap. 5. For instance, Fig. 5.5 uses a combination of MC dropout and test time augmentation (MCD_TTA) to estimate the uncertainty of a VGG-16 classifier trained on the CIFAR 10 dataset. An image from the test set is given to the classifier which predicts confidence probabilities for the ten different classes. The uncertainty estimation module stores predicted confidences from different forward passes of MCD_TTA (displayed in appendix C Table C.1) and the average softmax confidence per class is compute. The class having the highest average confidence value is the model’s predicted class. Since the uncertainty estimation module uses entropy as the metric to measure the uncertainty quality, the entropy of only the predicted class confidences across the several forward passes is computed. The entropy value is compared against the previously computed threshold using the rank histograms as discussed in Section 5.2. In Fig. 5.5, the threshold is given by entropy value at the 1 percentile line and anything that falls beyond this line, i.e., in the remaining 99 percentile region (shaded area in the plot) is identified as an uncertain prediction. Since the computed entropy value of the deer image in the figure falls in the shaded 99 percentile region, the classifier prediction is flagged as uncertain. This threshold can be varied by user according to the requirements of the application. Similarly, it is left to the user to

5.2. Comparison Based on Uncertainty Quality

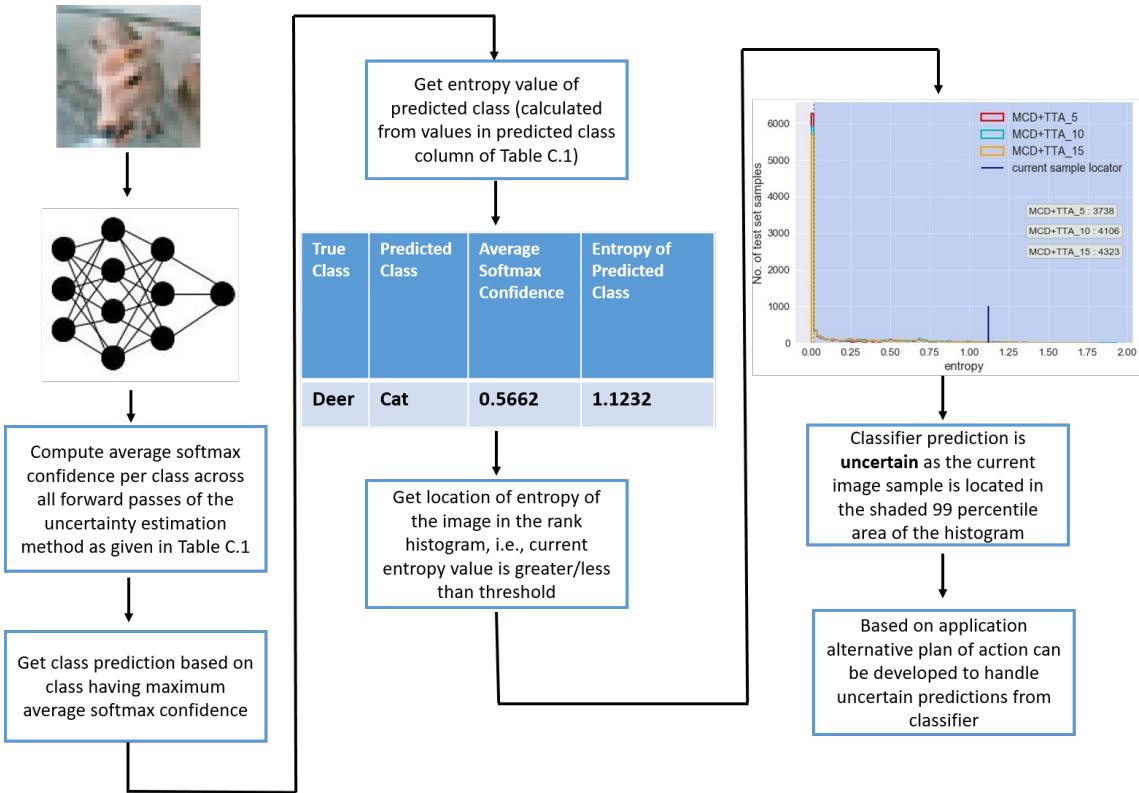


Figure 5.5: Uncertainty estimation workflow diagram depicting how uncertainty estimates are produced for a sample image of a deer from CIFAR 10 dataset with a VGG-16 classifier using MCD_TTA with 15 forward passes.

decide on an alternative course of action when the classifier prediction is identified as uncertain. For instance, the model predictions can be ignored and a human is brought inside the loop to make a classification decision when the deep learning model is uncertain, or the images causing uncertain predictions are collected and the model is re-trained on these images.

5.2.4 Performance on Out Of Distribution (OOD) data

The performance of various models along with the best variants of the uncertainty estimation methods is evaluated on three OOD datasets - TinyImageNet, Gaussian Noise and Uniform noise (Fig. 5.6). 1000 random samples from each of these datasets are collected and resized. The objective is to find the method(s) that are able to

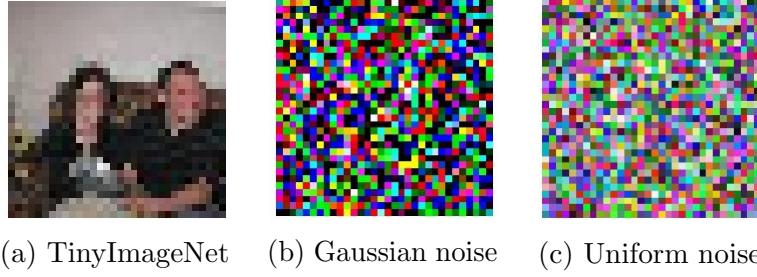


Figure 5.6: Sample images from TinyImageNet, Gaussian noise and uniform noise dataset

distinguish the OOD samples based on the number samples fall in the 99 percentile area of the entropy rank histograms based on the thresholds calculated from the experiments in Section 5.2.2.

Based on the percentage of identified OOD samples recorded in Table 5.8, the following inferences are made.

- The average percentage of samples identified by all models and methods is 89.4 for TinyImageNet, whereas it is only 63.11 and 58.98 for Gaussian noise and uniform noise dataset respectively. This shows that all methods are able to mostly identify images from another dataset such as TinyImageNet, but are not always able to identify noise.
- The ensemble method (EN) performs consistently well in identifying both the noise samples (except for Xception on RoboCup@Work dataset). Therefore, it can be concluded to be more robust to noise as compared to other methods.

5.3 Comparison Based on Classifier Performance

This section focuses on the comparison of classifiers using different uncertainty estimation methods based on the classifier performance metrics described in Section 5.1.2. This comparison is essential because it is important for uncertainty estimation methods to not degrade the performance of the baseline classifier, while the baseline model refers to a single model with a single forward pass through the test data (i.e., the red curves in Fig. 5.7 with 1 model on the top x-axis). In this regard, a comparison is made between the variants of different uncertainty methods MCD,

		TinyImagNet	Gaussian Noise	Uniform Noise
VGG-16 (CIFAR-10)	MCD	100	100	100
	EN	94.5	100	100
	MCDA	100	10	100
	TTA	100	10	100
	MCD+TTA	100	10	100
Wide-ResNet (CIFAR-10)	MCD	100	0	9.2
	EN	94.6	100	100
	MCDA	100	0	10
	TTA	100	0	10
	MCD+TTA	100	0	1.9
VGG-16 (CIFAR-100)	MCD	100	100	79.6
	EN	98.7	100	99.8
	MCDA	100	100	99.7
	TTA	100	100	100
	MCD+TTA	100	100	33.9
Wide-ResNet (CIFAR-100)	MCD	100	100	100
	EN	97.5	100	100
	MCDA	100	60.1	97.5
	TTA	100	100	100
	MCD+TTA	100	100	100
Xception (Optical Inspection dataset)	MCD	40.6	0	0
	EN	36.8	100	0
	MCDA	69.3	0	100
	TTA	44	0	0
	MCD+TTA	47.8	0	0
Xception (Robocup dataset)	MCD	93.4	3.1	0
	EN	65.8	100	27.8
	MCDA	99.8	100	100
	TTA	99.9	0	0
	MCD+TTA	99.3	0.2	0

Table 5.8: Percentage of OOD samples identified by different models and best variants of uncertainty estimation methods.

Chapter 5. Experimental Evaluation

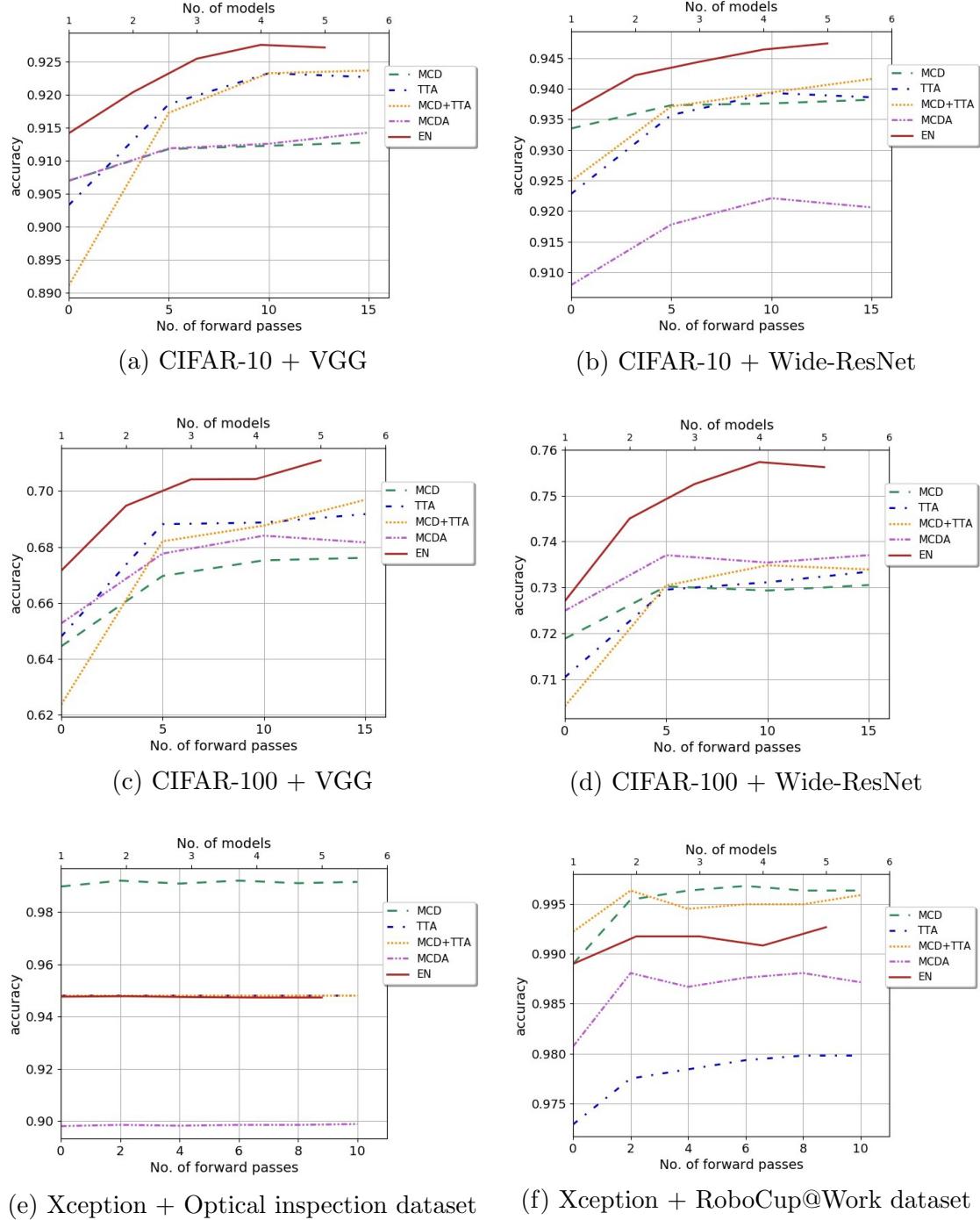


Figure 5.7: Figures depicting the accuracies of different classifiers on various datasets using different uncertainty estimation method. There are two x-axes (above and below the plot), one indicating the number of forward passes for MCD, TTA, MCD+TTA and MCDA the other indicating the number of models in the ensemble EN.

MCDA, EN, TTA, and MCD+TTA. We aim to find the method which is able to provide a better performance in terms of accuracy as compared to the baseline model.

The plots in Fig. 5.7 depict a minor increase in accuracy when using uncertainty estimation methods as compared to the baseline model in all cases except for MCDA in Fig. 5.7b, 5.7e and 5.7f, and TTA in Fig. 5.7f. The ensemble method EN achieves the highest accuracy in all cases and is closely followed or seconded by MCD+TTA.

We also find that the Area Under the Risk Coverage (AURC) and Excess - Area Under the Risk Coverage (EURC) metrics are not able to differentiate between different uncertainty estimation methods in all cases (more details in appendix A).

5.4 Comparison Based on Classifier Calibration

While uncertainty estimation methods provide reliable predictions, calibration measures the consistency of the classifier in producing such reliable predictions. Calibration of the classifier using different uncertainty estimation techniques is measured using the calibration metrics discussed in Section 5.1.3. The results using Brier score are not presented here because it evaluates both the discriminative power and the calibration of a classifier [3]. Since the main goal of this experiment is to solely evaluate classifier calibration, the results based on Expected Calibration Error (ECE) are presented below. However, more details on the results based on Brier score can be found in appendix B. The plots in Fig. 5.8 compare the variation in ECE for different methods on different datasets. Reliability diagrams for the method variants which have the least calibration error are given in Fig. 5.9.

Inferences from the plots in Fig. 5.8 are given as follows.

- MCD+TTA, TTA and EN produces more calibrated results as compared to other methods on the CIFAR-10 and CIFAR-100 datasets.
- TTA and MCD provide results with the least calibration error for the optical inspection dataset.
- TTA and EN have the least calibration error for the RoboCup@Work dataset.

5.5 Discussions

A summary of the observations and inferences is presented below.

- Theoretically developed metrics in Section 5.2 do not meet the requirements of practical applications (for e.g., optical inspection in industries) as the metric values need to be within constant bounds for all methods, datasets and models for easy interpretation by the user. As a result, the metrics are adapted by using scaled values that lie between 0 and 1, thereby facilitating easier comparison across different datasets and models. The predicted metric values for all samples in test set are scaled as follows,

$$scaled\ value = \min\left(\frac{orig\ value - min\ value}{max\ value - min\ value}, 1\right) \quad (5.12)$$

The maximum and minimum values are taken from the calculated maximum and minimum metric values for the classifier’s predictions among all the samples in the test set. Note that Eq. 5.12 also clips the value of the metric to 1 when an unseen image is fed to classifier and has a value greater than any of the images in the test set.

- The different metrics used of measuring the uncertainty quality are evaluated using the rank histogram based method of counting the number of test set samples that fall in the last 99 percentile area. Entropy yields the highest number of samples and therefore, is able to represent the uncertainty of the classifier in a larger portion of the dataset.
- Based on entropy, different variants of the uncertainty estimation methods are compared on various datasets. MCD+TTA augmentation performs better than the other methods in terms of uncertainty quality (Fig. 5.4), as it produces a combined estimate of the epistemic and aleatoric uncertainty. It is also able to identify the maximum number of misclassified samples as per table 5.7. MCD and EN have poor uncertainty quality, thereby yielding over-confident predictions as compared to MCD+TTA and TTA. MCDA does not exhibit consistent behavior in all datasets and models.
- All methods are able to identify OOD data from TinyImageNet dataset but do not exhibit consistent performance on Gaussian and uniform noise dataset. Analysis of this behavior on noise images is left for discussions in the future work Section 6.3.

- There is an improvement in model accuracy when uncertainty estimation methods are used in most of the cases. The ensemble method (EN) has the highest accuracy and followed by MCD+TTA.
- It is observed from the calibration plots in Fig. 5.8 that TTA, MCD+TTA and EN produce calibrated results as compared to other methods, thereby eliminating the need of using post calibration techniques such as softmax scaling [22]. However, the reason for this calibrated behavior is not analyzed here and is deferred for future analysis.
- Based on all the comparisons, TTA and its variant MCD+TTA provide good uncertainty estimates, model accuracy and calibrated predictions. This method poses a time constraint as multiple forward passes are required to obtain a prediction. The prediction time highly depends on how optimized the implementation is, for instance the data generators and the iterators used. Our implementation using the Keras data generator yields TTA_15 predictions in 5 milliseconds for a single image. This time constraint is a bit relaxed upon using the ensemble method (EN). However, this method has a very serious limiting factor as it requires additional GPU memory to store multiple models.

Chapter 5. Experimental Evaluation

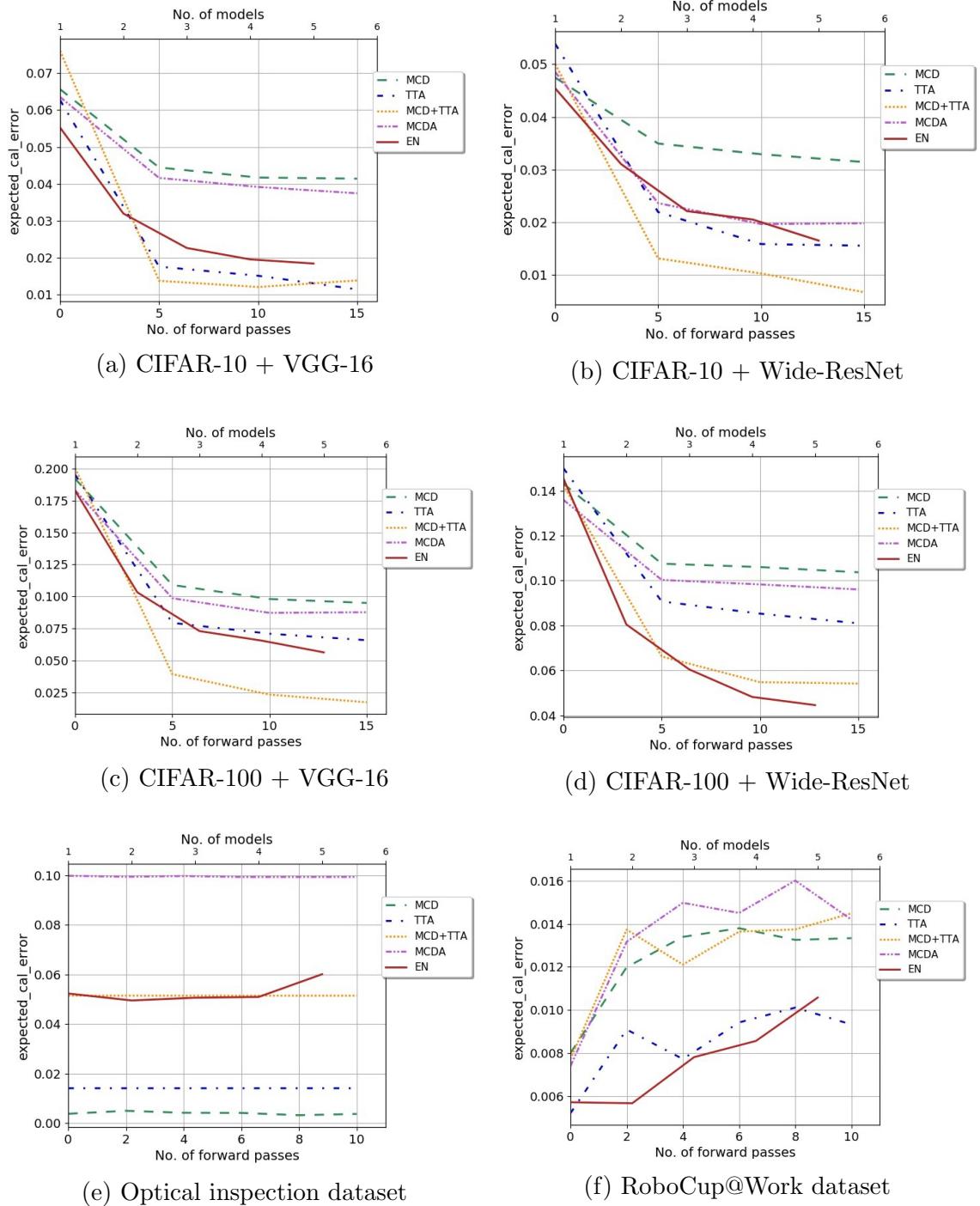
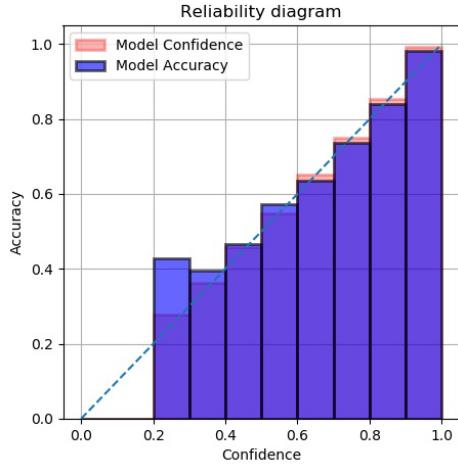
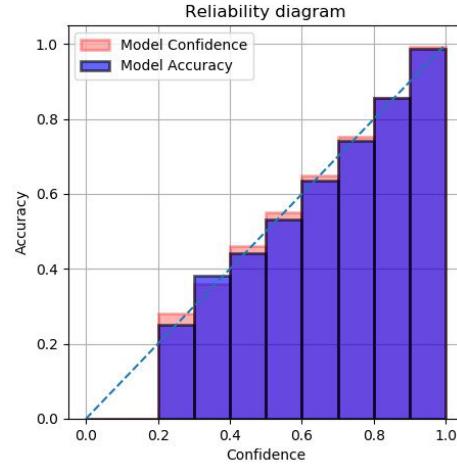


Figure 5.8: Figures depicting the Expected Calibration Error (ECE) of different uncertainty estimation methods on different datasets. There are two x-axes, one indicating the number of forward passes for MCD, TTA, MCD+TTA and MCDA, while the other indicating the number of models in the ensemble EN.

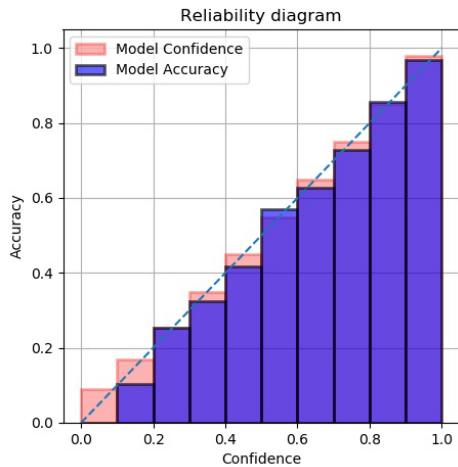
5.5. Discussions



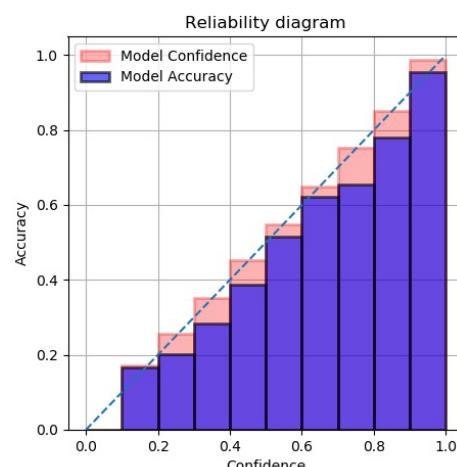
(a) TTA₁₅ on CIFAR-10 + VGG-16



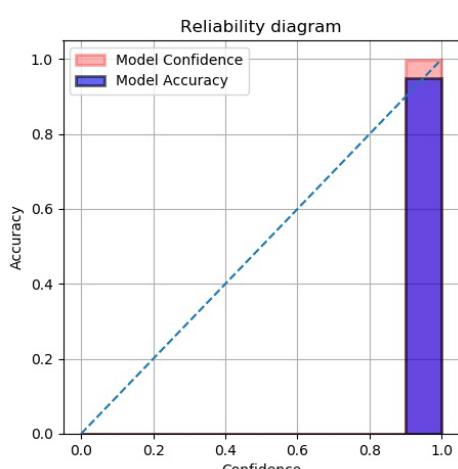
(b) MCD+TTA₁₅ on CIFAR-10 + WR



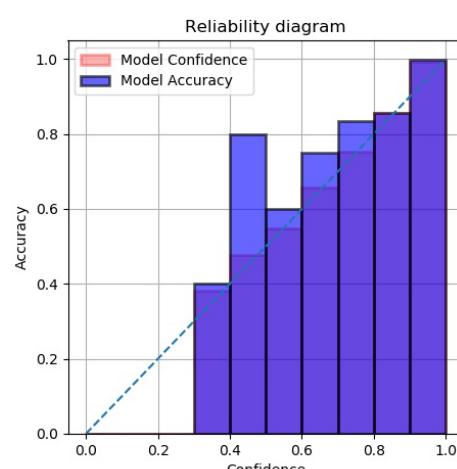
(c) MCD+TTA₁₅ on CIFAR-100 + VGG-16



(d) EN₅ on CIFAR-100 + Wide-ResNet



(e) MCD+TTA₈ on Optical inspection dataset + Xception



(f) TTA₁₀ on RoboCup@Work + Xception

Figure 5.9: Reliability diagrams for TTA variants which achieve least Expected Calibration Error (ECE) in Fig. 5.8

6

Conclusion

This work focuses on the use of uncertainty estimation methods to obtain reliable predictions from deep learning based classification models. A deep learning model using these methods produces uncertainty estimates along with the class predictions. The uncertainty estimates are high when the model is not sure about its predictions, for instance, when it makes a prediction on an image different from the training set. The state-of-the-art uncertainty estimation methods which can be applied to deep learning models are reviewed in this work. The two main constraints of the optical inspection and Robocup problem are that the uncertainty estimation methods should be fast enough in producing real-time uncertainty estimates and should not introduce major architecture changes to the existing deployed model. Five methods - MC dropout (MCD), deep ensembles (EN), test time data augmentation (TTA), combination of MC dropout and test time data augmentation (MCD+TTA) and MC dropout with Aleatoric loss (MCDA) - which satisfy the constraints are selected, and are evaluated on two standard datasets (CIFAR-10 and CIFAR-100) and two real world datasets (optical inspection and RoboCup@Work datasets).

Uncertainty estimation methods are evaluated on the following aspects,

- Uncertainty metrics which measure uncertainty quality are chosen. Based on the selected metric, the best uncertainty estimation method that is able to capture the model uncertainty, distinguish the misclassified samples and the Out Of Distribution (OOD) data is determined.

- Accuracy of the classifier using the uncertainty estimation methods is checked and compared, to ensure that the uncertainty estimation methods do not cause any degradation in classifier performance.
- Classifier calibration is evaluated to find out whether the use of uncertainty estimation methods is able to ensure a statistical consistency between the predicted confidence and accuracy.

6.1 Contributions

The major contributions of this work include,

1. Survey of different uncertainty estimation methods that can be applied to deep learning based classification models.
2. Implementation and evaluation of uncertainty estimation methods on open source and real world datasets.
3. Comparison and selection of uncertainty metrics that are used to evaluate the selected uncertainty estimation methods.
4. Comparison of uncertainty estimation methods in terms of number of identified misclassifications and detection of OOD.
5. Identification of misclassified samples in all selected datasets using uncertainty estimation.
6. Python (Keras/Tensorflow) based uncertainty estimation library which can be used to train models and obtain real-time uncertainty estimates by plugging in trained models and any dataset (more details on workflow in Appendix C).

6.2 Lessons Learned

The following lessons were learned during the comparison and evaluation of different uncertainty estimation methods,

- Uncertainty estimates can be harnessed from deep learning models by introducing stochasticity during prediction time. Stochasticity obtained by varying

the weights of the model yields epistemic uncertainty. On the other hand stochasticity in the input image by introducing noise in it yields aleatoric uncertainty.

- Aleatoric uncertainty constitutes a major portion of the predictive uncertainty. Even in the absence of an epistemic uncertainty estimate, the results from aleatoric uncertainty alone is very close to the combination of both.
- A classifier using test time data augmentation produces calibrated results, along with good uncertainty estimates. This is advantageous as it eliminates the use of additional post calibration techniques such as softmax calibration [22].
- All the selected uncertainty estimation methods are able to distinguish meaningful images from other datasets (such as TinyImageNet) but do not perform well on noise images.

6.3 Future Work

MCD provides an estimate of the epistemic uncertainty by using a different set of model weights during each forward pass at test time. The dropout layers in the model induce the stochasticity in the weights, causing the model weights to change during each forward pass. However, the number of dropout layers and the dropout rates have to be decided and set by user while it is impossible to manually identify the values that result in an optimal model. Concrete dropout [15] and variational dropout [39], on the other hand, provide optimal dropout rates which can automatically be learned during training. Further experiments need to be carried out to observe whether the use of these dropout variants for uncertainty estimation provides an improvement in epistemic uncertainty of the model.

Guo et al. [22] showed that deep learning models are poorly calibrated. However, our experiments show that test time data augmentation (TTA) causes the deep learning model to produce calibrated confidence estimates, thereby eliminating the need for any post calibration techniques. The exact reason for the well calibrated behavior of models using test time data augmentation requires further analysis and reasoning.

6.3. Future Work

A more detailed analysis on the performance of different uncertainty estimation methods on other OOD datasets such as LSUN [60], iSUN and adversarial perturbations is required. The failure of uncertainty estimation methods in not being able to consistently detect images from the Gaussian noise or uniform noise dataset needs further investigation. Also, alternate methods of OOD detection as in [36], [12], [58], [50] can be used in such cases. A comparative evaluation on the performance of the uncertainty estimation methods and OOD detection methods, and their combinations is worth investigating.

A

Evaluation using Risk Coverage curves

The Risk Coverage (RC) curves [18] measure the risk incurred by the classifier in misclassifying samples as the number of samples in the dataset (coverage) increase. The lesser the Area Under the Risk Coverage (AURC) curve, the lesser the risk incurred by the classifier for an increased coverage value. Fig. A.1 shows the RC curve plotted for a baseline classifier, which does not use any uncertainty estimation method and a classifier which uses different uncertainty estimation methods. The difference between the baseline curve and the curve of the classifier using an uncertainty estimation method gives the value of Expected Area Under the Risk Coverage (E-AURC) curve. The values of AURC and E-AURC are computed as per Section 5.1.2. From the plots in Fig. A.1, it is seen that a classifier using an uncertainty estimation method has a lesser AURC than the baseline classifier. Hence, uncertainty estimation methods reduce the risk incurred by a classifier in misclassifying samples, which is also evident from the increased accuracy in the accuracy plots in Fig. 5.7. However, the RC curves are not able to discriminate and identify the best uncertainty estimation, because the E-AURC values are more or less the same for all methods.

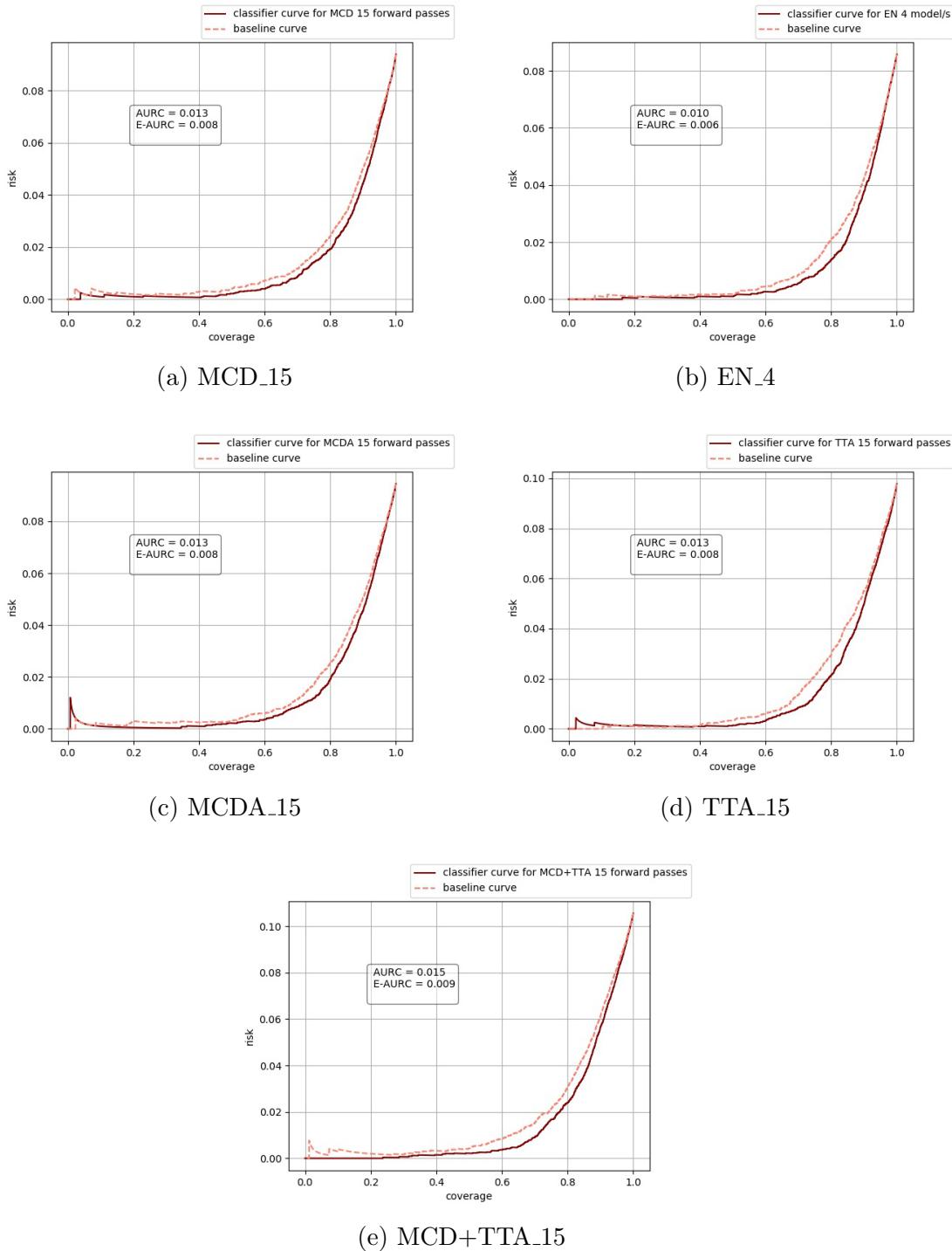


Figure A.1: Figures depicting the RC curves along with the E-AURC values for different uncertainty estimation methods on VGG-16 model used for classifying images from CIFAR-10 dataset.

B

Evaluation using Brier score

The results of evaluating the classifier predictions using Brier score is presented here. The Brier score per classifier prediction is computed as described in Section 5.1.3. Since, Brier score evaluates both the calibration and the discriminative power of the classifier, the plots in Fig. B.1 are approximately the inverse of the accuracy plots in Fig. 5.7. The plots in Fig. B.1 represent the average Brier score values for different variants of the uncertainty estimation methods on various datasets and models.

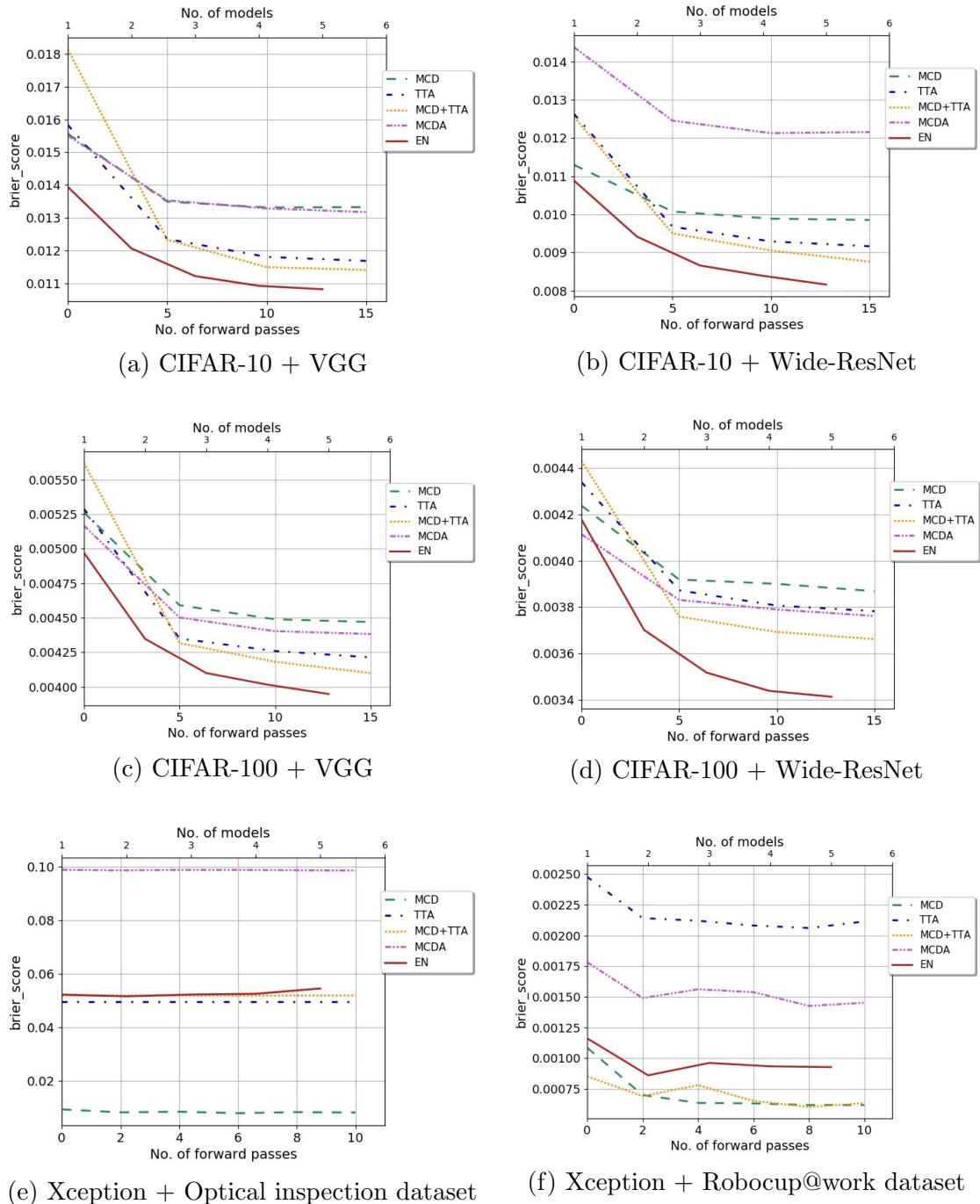


Figure B.1: Figures depicting the average Brier score values of different classifiers on various datasets using different uncertainty estimation method. There are two x-axes, one indicating the number of forward passes for MCD, TTA, MCD+TTA and MCDA, while the other indicating the number of models in the ensemble EN.

C

Uncertainty Estimation Work Flow Data

No. of forward passes	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
1	0.0005	0.0002	0.0011	0.5875	0.0059	0.2633	0.0003	0.1394	0.0002	0.0014
2	0.0002	0.0001	0.0348	0.0194	0.001	0.9421	0.0006	0.0014	0.0002	0.0001
3	0.0004	0.0001	0.001	0.1542	0.0238	0.0272	0.0003	0.792	0.0001	0.0009
4	0.0001	0	0.002	0.4241	0.0247	0.2156	0.0001	0.3328	0.0002	0.0003
5	0.0003	0	0.0035	0.771	0.0675	0.003	0.0001	0.1538	0.0003	0.0005
6	0	0	0.0001	0.8893	0.0003	0.1096	0.0001	0.0005	0	0
7	0.0007	0.0005	0.0009	0.1623	0.3332	0.0845	0.0025	0.4109	0.0004	0.0041
8	0	0	0.0003	0.958	0.0002	0.0392	0	0.0022	0	0
9	0	0	0	0.9999	0	0.0001	0	0	0	0
10	0.0003	0.0001	0.0013	0.1609	0.1768	0.5718	0.0003	0.0881	0.0001	0.0002
11	0.0016	0.0001	0.0019	0.9466	0.0035	0.035	0.0002	0.0086	0.0007	0.0017
12	0.0004	0.0002	0.0031	0.2275	0.0723	0.5188	0.0004	0.176	0.0006	0.0008
13	0.0005	0	0.0011	0.6426	0.0002	0.3492	0	0.0059	0.0003	0.0002
14	0.0006	0.0005	0.0016	0.5905	0.0077	0.3509	0.0015	0.0441	0.0003	0.0023
15	0.0002	0	0.0002	0.9597	0.0013	0.0369	0.0004	0.001	0.0001	0.0002
Average confidence	0.0004	0.0001	0.0035	0.5662	0.0479	0.2365	0.0005	0.1438	0.0002	0.0009

Table C.1: Softmax confidence per class for 15 forward passes of MCD_TTA with the deer image (in Fig. 5.5) as input to the VGG-16 classifier.

References

- [1] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160, 2018.
- [2] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.
- [3] Melissa Assel, Daniel D Sjoberg, and Andrew J Vickers. The brier score does not evaluate the clinical utility of diagnostic tests or prediction models. *Diagnostic and prognostic research*, 1(1):19, 2017.
- [4] Andrei Atanov, Arsenii Ashukha, Dmitry Molchanov, Kirill Neklyudov, and Dmitry Vetrov. Uncertainty estimation via stochastic batch normalization. *arXiv preprint arXiv:1802.04893*, 2018.
- [5] Murat Seckin Ayhan and Philipp Berens. Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks. *International conference on Medical Imaging with Deep Learning*, 2018.
- [6] David Barber and Christopher M Bishop. Ensemble learning in bayesian neural networks. *Nato ASI Series F Computer and Systems Sciences*, 168:215–238, 1998.
- [7] William H Beluch, Tim Genewein, Andreas Nürnberger, and Jan M Köhler. The power of ensembles for active learning in image classification. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9368–9377, 2018.
- [8] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

-
- [9] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.
 - [10] François Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint*, pages 1610–02357, 2017.
 - [11] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning. *arXiv preprint arXiv:1710.07283*, 2017.
 - [12] Terrance DeVries and Graham W Taylor. Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865*, 2018.
 - [13] Yarin Gal. *Uncertainty in deep learning*. PhD thesis, PhD thesis, University of Cambridge, 2016.
 - [14] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *International Conference on Machine Learning*, pages 1050–1059, 2016.
 - [15] Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. *Advances in Neural Information Processing Systems*, pages 3581–3590, 2017.
 - [16] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. *International Conference on Machine Learning*, pages 1183–1192, 2017.
 - [17] Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. *Advances in Neural Information Processing Systems*, pages 4878–4887, 2017.
 - [18] Yonatan Geifman, Guy Uziel, and Ran El-Yaniv. Boosting uncertainty estimation for deep neural classifiers. *arXiv preprint arXiv:1805.08206*, 2018.
 - [19] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. *International Conference on Artificial Intelligence and statistics*, pages 249–256, 2010.

References

- [20] Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):243–268, 2007.
- [21] Alex Graves. Practical variational inference for neural networks. *Advances in Neural Information Processing Systems*, pages 2348–2356, 2011.
- [22] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017.
- [23] Thomas M Hamill. Interpretation of rank histograms for verifying ensemble forecasts. *Monthly Weather Review*, 129(3):550–560, 2001.
- [24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.
- [25] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. *International Conference on Machine Learning*, pages 1861–1869, 2015.
- [26] Geoffrey Hinton and Drew Van Camp. Keeping neural networks simple by minimizing the description length of the weights. *Proceedings of the 6th Ann. ACM Conf. on Computational Learning Theory*, 1993.
- [27] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- [28] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- [29] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in Neural Information Processing Systems*, pages 5574–5584, 2017.

-
- [30] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
 - [31] Gerhard K. Kraetzschmar, Nico Hochgeschwender, Walter Nowak, Frederik Hegger, Sven Schneider, Rhama Dwiputra, Jakob Berghofer, and Rainer Bischoff. Robocup@work: Competing for the factory of the future. In Reinaldo A. C. Bianchi, H. Levent Akin, Subramanian Ramamoorthy, and Komei Sugiura, editors, *RoboCup 2014: Robot World Cup XVIII*, volume 8992 of *Lecture Notes in Computer Science*, pages 171–182. Springer International Publishing, 2015.
 - [32] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.
 - [33] Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1997.
 - [34] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.
 - [35] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, et al. Towards fully autonomous driving: Systems and algorithms. pages 163–168, 2011.
 - [36] Shiyu Liang, Yixuan Li, and R Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.
 - [37] Alexander Selvikvåg Lundervold and Arvid Lundervold. An overview of deep learning in medical imaging focusing on mri. *Zeitschrift für Medizinische Physik*, 29(2):102–127, 2019.
 - [38] MAS-Group. Robocup@work, 2019. URL https://mas-group.inf.h-brs.de/?page_id=23.
 - [39] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. pages 2498–2507, 2017.

References

- [40] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. MIT Press, 2013.
- [41] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.
- [42] National Transportation Safety Board (NTSB). *Preliminary Report Highway HWY18MH010*, 2007.
- [43] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.
- [44] Bharath Ramsundar, Steven Kearnes, Patrick Riley, Dale Webster, David Konerding, and Vijay Pande. Massively multitask networks for drug discovery. *arXiv preprint arXiv:1502.02072*, 2015.
- [45] Jorge A Revelli, Miguel A Rodríguez, and Horacio S Wio. The use of rank histograms and mvl diagrams to characterize ensemble evolution in weather forecasting. *Advances in Atmospheric Sciences*, 27(6):1425–1437, 2010.
- [46] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*, 2016.
- [47] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.
- [48] Casey Ross and Ike Swetlitz. Ibms watson supercomputer recommended unsafe and incorrectcancer treatments, internal documents show. *Stat News*. URL <https://www.statnews.com/wp-content/uploads/2018/09/IBMs-Watson-recommended-unsafe-and-incorrect-cancer-treatments-STAT.pdf>. [Online; accessed 14-August-2019].
- [49] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

- [50] Gabi Shalev, Yossi Adi, and Joseph Keshet. Out-of-distribution detection using multiple semantic label representations. *Advances in Neural Information Processing Systems*, pages 7375–7385, 2018.
- [51] Claude E Shannon and Warren Weaver. *The mathematical theory of communication*. University of Illinois press, 1998.
- [52] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.
- [53] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [54] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [55] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *IEEE conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.
- [56] Mattias Teye, Hossein Azizpour, and Kevin Smith. Bayesian uncertainty estimation for batch normalized deep networks. *arXiv preprint arXiv:1802.06455*, 2018.
- [57] Uncertainty quantification. Uncertainty quantification — Wikipedia, the free encyclopedia, 2010. URL https://en.wikipedia.org/wiki/Uncertainty_quantification. [Online; accessed 23-July-2019].
- [58] Apoorv Vyas, Nataraj Jammalamadaka, Xia Zhu, Dipankar Das, Bharat Kaul, and Theodore L Willke. Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. *European Conference on Computer Vision (ECCV)*, pages 550–564, 2018.

References

- [59] Guotai Wang, Wenqi Li, Michael Aertsen, Jan Deprest, Sébastien Ourselin, and Tom Vercauteren. Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks. *Neurocomputing*, 2019.
- [60] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.
- [61] Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. *International Conference on Machine Learning*, 1:609–616, 2001.
- [62] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699, 2002.
- [63] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.