Université de Montréal

Variational Aleatoric Uncertainty Calibration in Neural Regression

par

Dhaivat Bhatt

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Discipline

July 14, 2021

Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé

Variational Aleatoric Uncertainty Calibration in Neural Regression

présenté par

Dhaivat Bhatt

a été évalué par un jury composé des personnes suivantes :

Christopher Pal
(président-rapporteur)

Liam Paull
(directeur de recherche)

Devon Hjelm
(membre du jury)

Résumé

Des mesures de confiance calibrées et fiables sont un prérequis pour la plupart des systèmes de perception robotique car elles sont nécessaires aux modules de fusion de capteurs et de planification qui interviennent plus en aval. Cela est particulièrement vrai dans le cas d'applications où la sécurité est essentielle, comme les voitures à conduite autonome. Dans le contexte de l'apprentissage profond, l'incertitude prédictive est classée en incertitude épistémique et incertitude aléatoire. Il existe également une incertitude distributionnelle associée aux données hors distribution. L'incertitude aléatoire représente l'ambiguïté inhérente aux données d'entrée et est généralement irréductible par nature. Plusieurs méthodes existent pour estimer cette incertitude au moyen de structures de réseau modifiées ou de fonctions de perte. Cependant, en général, ces méthodes manquent de calibration, ce qui signifie que les incertitudes estimées ne représentent pas fidèlement l'incertitude des données empiriques. Les approches actuelles pour calibrer l'incertitude aléatoire nécessitent soit un "ensemble de données de calibration", soit de modifier les paramètres du modèle après l'apprentissage. De plus, de nombreuses approches ajoutent des opérations supplémentaires lors de l'inférence. Pour pallier à ces problèmes, nous proposons une méthode simple et efficace d'entraînement d'un régresseur neuronal calibré, conçue à partir des premiers principes de la calibration. Notre idée maîtresse est que la calibration ne peut être réalisée qu'en imposant des contraintes sur plusieurs exemples, comme ceux d'un mini-batch, contrairement aux approches existantes qui n'imposent des contraintes que sur la base d'un échantillon. En obligeant la distribution des sorties du régresseur neuronal (la distribution de la proposition) à ressembler à une distribution cible en minimisant une divergence f, nous obtenons des modèles nettement mieux calibrés par rapport aux approches précédentes. Notre approche, f-Cal, est simple à mettre en œuvre ou à ajouter aux modèles existants et surpasse les méthodes de calibration existantes dans les tâches réelles à grande échelle de détection d'objets et d'estimation de la profondeur. f-Cal peut être mise en œuvre en 10-15 lignes de code PyTorch et peut être intégrée à n'importe quel régresseur neuronal probabiliste, de façon peu invasive. Nous explorons également l'estimation de l'incertitude distributionnelle pour la détection d'objets, et employons des méthodes conçues pour les systèmes de classification. Nous établissons un problème d'arrière-plan hors distribution qui entrave l'applicabilité des méthodes d'incertitude distributionnelle dans la détection d'objets.

Mots clés: Calibration de l'incertitude aléatoire, apprentissage probabiliste, Vision robotique, estimation de l'incertitude.

Abstract

Calibrated and reliable confidence measures are a prerequisite for most robotics perception systems since they are needed by sensor fusion and planning components downstream. This is particularly true in the case of safety-critical applications such as self-driving cars. In the context of deep learning, the sources of predictive uncertainty are categorized into epistemic and aleatoric uncertainty. There is also distributional uncertainty associated with out of distribution data. Epistemic uncertainty, also known as knowledge uncertainty, arises because of noise in the model structure and parameters, and can be reduced with more labeled data. Aleatoric uncertainty represents the inherent ambiguity in the input data and is generally irreducible in nature. Several methods exist for estimating aleatoric uncertainty through modified network structures or loss functions. However, in general, these methods lack *calibration*, meaning that the estimated uncertainties do not represent the empirical data uncertainty accurately. Current approaches to calibrate aleatoric uncertainty either require a held out "calibration dataset" or to modify the model parameters post-training. Moreover, many approaches add extra computation during inference time. To alleviate these issues, this thesis proposes a simple and effective method for training a calibrated neural regressor, designed from the first principles of calibration. Our key insight is that calibration can be achieved by imposing constraints across multiple examples, such as those in a mini-batch, as opposed to existing approaches that only impose constraints on a per-sample basis. By enforcing the distribution of outputs of the neural regressor (the proposal distribution) to resemble a target distribution by minimizing an f-divergence, we obtain significantly better-calibrated models compared to prior approaches. Our approach, f-Cal, is simple to implement or add to existing models and outperforms existing calibration methods on the large-scale real-world tasks of object detection and depth estimation. f-Cal can be implemented in 10-15 lines of PyTorch code, and can be integrated with any probabilistic neural regressor in a minimally invasive way. This thesis also explores the estimation of distributional uncertainty for object detection, and employ methods designed for classification setups. In particular, we attempt to detect out of distribution (OOD) samples, examples which are not part of training data distribution. I establish a background-OOD problem which hampers applicability of distributional uncertainty methods in object detection specifically.

Keywords: Aleatoric uncertainty calibration, Probabilistic learning, Robotic vision, Uncertainty estimation

Contents

List of tables

List of figures

Liste des sigles et des abréviations

CNN	Convolutional neural networks
MCMC	Markov Chain Monte Carlo
MCE	Maximum calibration Error
ECE	Expected calibration error
BNN	Bayesian neural networks
i.i.d/IID	Independently and identically distributed
OOD	Out of distribution
DoF	Degrees of freedom

Remerciements

This thesis would not have been possible without many people. First and foremost, I am incredibly grateful to Prof. Liam Paull, he has been supportive throughout my masters. He always lets us freely explore various different areas of the field, and provided his valuable guidance at every stage of this journey. His emphasis to achieve perfection greatly motivates us to push ourselves beyond our limits. I had to travel to India in the middle of Covid-19 pandemic owing to certain family emergency, and he supported me throughout the period of crisis. For that, I will forever be thankful to him. I was fortunate enough to have talented roboticist Krishna Murthy with me, who has been with me throughout my time at MILA. He has provided me valuable inputs many times, and steered projects in the right direction when I lose the track.

I was fortunate enough to be able to collaborate with talented set of researchers from MIT, Teddy Ort, Rohan Banerjee, Igor Gilitschenski and others for MapLite. I'd like to express deep sense of gratitude for my talented labmates at REAL lab, including Krishna, Sai, Vincent, Manfred, Florian, Dishank, Bhairav, Nithin, Breandan and Gunshi. During many group meeting presentations, they provided me valuable comments to enhance quality of my work. I was lucky to be around a talented peer group at MILA, including Nikolaus, Mostafa, Rey, Phillipe, Soumye, Charan, Soroosh with whom I have had numerous interactions which shaped my views about academia and research. I am deeply thankful to tech and admin team at MILA, and Olexa, for saving many debugging hours. I would like to thank my co-authors for this work, Kaustubh, Krishna, Hanju Lee, Dishank and Liam, without whom this thesis would not have materialized. I am deeply grateful to Kaustubh, for passionately working with me on this project, from designing experiments, to proofreading papers and reviewing codes, he was by my side throughout this project.

I am deeply thankful to MITACS, for providing me with summer training scholarship, as a part of my masters. I am thankful to University of Montreal, for type-C scholarship, which exempted me from paying international tuition at the university.

Montreal wouldn't have been fun without my roommate Utsav Sadana, who had been very supportive throughout my masters, always cooked tea for me during conference deadlines. I am thankful to my friends Thibault, Valeriane, Aditya, without whom life would have been boring. I am thankful to Vincent, for being a very supportive friend during the time of crisis. Many games of table-tennis and foosball with Vincent always refreshed me after long hours of coding. I am

thankful to Tim Hortons at Cote-des-neiges/Av. Linton, for their French vanillas and free WiFi, which kept me caffeinated during late night paper reading sessions.

I am foremost grateful to my family, for supporting me throughout my masters. Here I take an opportunity to extend my gratitude to my parents, my sister, my brother-in-law for their unconditional love and moral support. Their enormous faith in my decisions and continuous encouragement kept me going during many moments of crisis. I cannot forget to thank my 3 years old niece Shlesha, who always brought a smile to my face with her gurgling laughter after a long day at the lab. I am deeply grateful to my closest friend and support system, Miloni, for believing in me, standing by me and motivating me, during many conference deadlines. At last, I would like to thank all my school and college friends, who kept checking on me throughout the Covid-19 pandemic, I can not list all the names here, but you are always on my mind.

Chapter 1

Introduction

The *performance* of deep neural network based computer vision perception systems has increased considerably in recent years. Convolutional neural networks (CNNs) have become the *de facto* choice for advanced perception systems in robotics and autonomous driving. Deep neural networks have become a standard choice for many computer vision applications such as object detection, semantic segmentation, keypoint estimation and object tracking. CNNs achieve state-of-the-art performance for variety of these tasks. However, for many *embodied* applications, performance alone is not sufficient. In order to ensure compliance of these models with other elements of the standard robotics stack, such as sensor fusion (e.g., [**10**]) and probabilistic planning (e.g., [**80**]), we also require that these models are able to provide reliable and *calibrated* measures of *the confidence* associated with their predictions.

Modern deep learning models act as *deterministic functions*. Existing deep neural regression models provide point estimate for each output prediction. *Reliably* measuring the confidence of such predictions benefits a variety of downstream robotics tasks, ranging from state estimation, to planning, to control. The operative word here is *reliable*: simple schemes that interpret softmax classifier scores as probabilities have shown to be grossly incorrect and overconfident [**22, 73, 5, 14**]. This has resulted in the development of *Bayesian deep learning* techniques, which have garnered plenty of attention [**25, 26, 69, 54**] in the community.

In a typical robotics stack, a *measurement model* is expressed as: $z = h(x,\nu)$, where z is the measurement, x is the underlying partially observable state, and ν is some noise term that accounts for the *inherent variability* of the measurement process. Here, measurement process refers to the action of measuring certain quantity from an input. In range-bearing sensor, we can consider the physical environment and robot position as inputs to the sensor, and it produces range and angle readings of the landmarks within an environment. In the case of traditional non-deep learning based sensor systems, the model h is derived from our understanding of the underlying operation of the sensor (e.g., a laser range finder reading should be modeled as a "range/bearing" measurement). A parametric model class is assumed for the noise and then the parameters are fit through a controlled

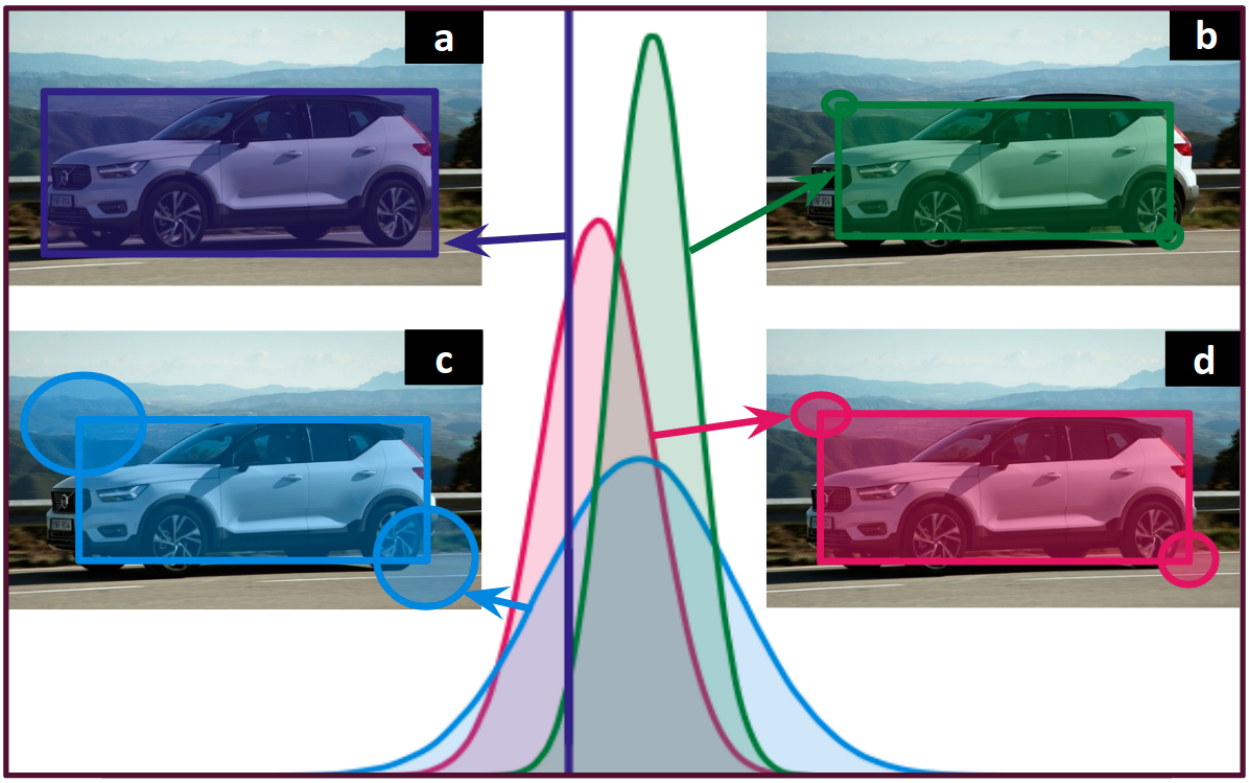


Fig. 1.1. f**-Cal** is a simple and effective approach to *calibrate* the uncertainty estimates from a neural network performing regression. In above figure, we represent uncertainty of each bounding box with the uncertainty ellipses. Each ellipse represent uncertainty for a definite confidence interval. (a) ground-truth bounding box, (b) overconfident and inconsistent estimate (c) under-confident but consistent estimate, (d) calibrated estimate - the error ellipses correspond to the true underlying uncertainties.

calibration experiment. For example, using GPS data to calibrate the noise characteristics of an acoustic-ranging based localization system [**101**]. This stochastic model is a requirement in order to be able to determine a *measurement likelihood*.

In deep learning, the model, h is given by the CNN, and the measurement z is given by the output of the model. We can draw analogy similar to range-bearing sensor here. If we take an example of image classification, input to the sensor(CNN in this case) is an image, and the measurement is categorical distribution over space of output classes. In this case, forward pass through the model can be interpreted as measurement process. Output of this process is categorical distribution, which we can interpret as measurement likelihood. In a typical deep learning setup, we have no way of evaluating the measurement likelihood without a calibrated estimate of the uncertainty. We need to use the *dataset itself* as a proxy for the traditional controlled sensor calibration experiment. For deep networks to be integrated into safety-critical embodied systems, they must be capable of estimating the predictive uncertainty in their outputs.

In the machine learning community, this is encompassed by the rapidly growing field of "Bayesian deep learning" and has resulted in the development of network models that estimate a posterior distribution over the output space [**14, 15, 46, 67, 66, 54, 3, 36**]. These techniques try to predict the posterior over the output instead of providing a point estimate. Central to this discussion is the characterization of the types of predictive uncertainty that may be present. I adopt the nomenclature of [**14, 46**], which distinguishes between uncertainty resulting from data distributional shift (distributional uncertainty), which arises due to an input from a completely different data-distribution. Such examples are known as out-of-distribution samples. Another major source of predictive uncertainty is a poorly trained or improperly chosen model (epistemic uncertainty), which results in lack of proper knowledge of the model, and can be reduced using sufficient data. Lastly, an irreducible error in the underlying process (aleatoric uncertainty) can introduce noise in the measurement process. This could happen because of disagreement in annotations of the same input. For example, in object detection, different human annotators could annotate the same object with slightly different bounding boxes, thus introducing noise in the system. In this work I assume that we are operating "in distribution" (unless otherwise specified), and that our model is well-chosen and trained (epistemic and distributional uncertainties are low) and are largely concerned with the irreducible uncertainty associated with the "measurement" process. This uncertainty could result from, for example, inherent ambiguity or inconsistency in the underlying data used for training. We also assume that this uncertainty is *input dependent* and therefore heteroscedastic. For example, different annotators might put a bounding box for an object at a slightly different location, particularly when the object is partially occluded.

Bayesian neural networks have become the *de facto* standard for uncertainty estimation in modern neural networks [**14, 15**]. However, most techniques to estimate aleatoric uncertainty suffer from a lack of calibration [**22**]. Uncalibrated uncertainty estimates do not provide an accurate representation of error in estimates and are therefore unreliable. These estimates either over or under estimate the confidence, and therefore cannot be interpreted as probability densities, nor can they be directly comparable across multiple samples and/or models. Calibrated uncertainties have the advantage of being directly interpreted as probability densities, and are directly comparable across multiple samples and/or models. In a typical classification setup, if the model is predicting something with 0.8 confidence, we expect the model to be right exactly 80 out of 100 times. A model is calibrated if its confidence measure is same as the accuracy of the model. If model has low confidence but high accuracy, such models can be interpreted as under-confident models. While models with extremely high confidence values, but lower accuracy are typically overconfident. The objective is to have a calibrated model, so we can use confidence of the model for informed decision making. Current classification models trained with cross-entropy objective do not provide well-calibrated confidence predictions.

To mitigate this lack of uncertainty calibration, several methods have been proposed in the context of Bayesian neural networks(BNNs), such as to apply isotonoic regression [**106**] or temperature scaling [**22**]. These methods are restricted to the case of classification. In this work we are strictly interested in regressor models whose outputs are continuously valued.

There have been some recent works to attempt to adopt some of these BNN methods to the regression setting [**22, 49, 13, 57**]. However, a large portion of the works perform a post-hoc analysis to learn a transformation on uncertainty estimates to produce better calibrated estimates. As a result, these methods are highly dependent on a recalibration dataset and tend to overfit on it. Additionally, these are post-processing methods which add significantly to the computation requirements *at inference time*.

To this end, I present a simple and effective procedure to directly train calibrated neural networks for regression problems. Inspired by the first principles of calibration, I derive a variational approach that directly enforces that the neural regressor is calibrated. Unlike prior approaches, this approach neither require a held-out calibration dataset, nor requires a post-hoc modification to the model parameters. Our approach is easily imposed as an additional loss term over standard aleatoric uncertainty estimation, and can be implemented in under $10-15$ lines of additional code. Our approach implicitly encourages the model to learn uncertainty aware representations.

Our insight is that distributional calibration *cannot be achieved by a loss function that operates over individual samples*. Calibration is achieved when distributional constraints are enforced upon uncertainty estimates across multiple (i.i.d.) samples. To achieve this we employ distribution matching over the error statistics at the *mini-batch level*. This can be enforced so long as we can compute the distance between the empirical distribution of model prediction errors, and some known calibrated distribution.

We formulate this approach first in the general case, and then show that this is particularly straightforward in the case that the errors are modeled as Gaussian. We leverage the fact that each sample can be scaled to a unit standard normal, and then that the sum of these follows a χ^2 distribution with K degrees of freedom, where K is the number of samples in the mini-batch. For large K, this distribution approaches a normal distribution with mean K and variance $2K$. We can therefore enforce a variational loss that encourages the normalized sum of the errors to follow $\mathcal{N}(K,2K)$, and this explicitly enforces that the error uncertainty estimates are calibrated.

In summary, we communicate the following contributions[1]:

[1]This work closely overlaps with the work submitted under the title "f-Cal: Variational calibration of aleatoric uncertainty in neural regression" to the fifth Conference on Robot learning (CoRL 2021) and is currently under review. Chapters 2, 3 and 4 largely overlap with the submitted work and have been augmented with explanations of background concepts and a more extensive literature review. The authors of the submitted paper are Dhaivat Bhatt, Kaustubh Mani, Dishank Bansal, Krishna Murthy, Hanju Lee and Liam Paull. Liam Paull came up with the intuition of distribution matching for calibration. Dhaivat Bhatt conceptualized f-Cal based on this idea, designed and prototyped the loss function, and tested it on object detection. Dhaivat did literature survey, formulated the problem, designed experimental setup, implemented and tested the Bayesian object detector and ran a series of experiments on object detection to robustly evaluate the idea. Kaustubh designed the toy experiments under controlled setup. Dishank and Kaustubh also

- I present a simple, principled, and effective approach to calibrating uncertainty estimates in neural regressors;
- Our approach—dubbed f-Cal—neither requires a held-out calibration dataset, nor requires additional inference time compute. Our loss function can be employed in most regression pipelines without any modification to the remainder of the pipeline.
- This method demonstrates superior performance to existing calibration approaches across several tasks, dataset sizes, and network architectures.

We demonstrate the effectiveness, scalability and widespread applicability of this approach on large-scale, real-world tasks such as object detection and depth estimation. In addition to aleatoric uncertainty, I also attempt to detect out-of-distribution samples for object detection. Through controlled experiments, I establish that the concept of a **background** class in object detection severely impacts the applicability of typical distributional uncertainty estimation methods designed for image classification and segmentation.

The rest of the thesis is divided into 4 different chapters. In Chapter 2, I introduce some of the background concepts. In particular, I review some key concepts of probability and statistics, followed by a brief overview of object detection. Then I talk about different types of uncertainties in deep learning in Section 2.1.3. I formally define problem of uncertainty calibration and go through related work in Section 2.1.4. In Chapter 3, I explain the f-Cal method, derive the f-Cal loss and show the algorithm for Gaussian uncertainties. In Section 4, I discuss about evaluation metrics for calibration, then I show results on a toy dataset in Section 4.4. In Section 4.5, I discuss consistency and calibration and extensively report results for the KITTI and Cityscapes datasets. In Section 4.6, I show results of f-Cal for the problem of depth estimation. This is followed by an ablation over the different components of the method. In Chapter 5, I show experimental results of out-of-distribution detection for the problem of object detection. Finally, in Chapter 6.2, I list certain limitations of the proposed method and conclude.

worked on the depth estimation task. Krishna helped in writing of the paper and Hanju provided his valuable inputs and feedback.

Chapter 2

Background

In this chapter, I review several key concepts required to understand the key contributions of this work. I start by reviewing some tools from linear algebra and probability theory in Section 2.1.1. In Section 2.1.2, I specifically delve into the problem of object detection, and provide overview of various methods. In Section 2.1.3, I describe predictive uncertainty and its common categorizations. In Section 3.1, I formalize the problem of calibrated uncertainty estimation. Finally in Section 2.1.4, I provide a detailed overview of related work.

2.1. Preliminaries

In this section, I will briefly cover several concepts necessary to understand the material in the rest of the thesis. I start by reviewing some important concepts of probability theory and statistics. In particular, I review the Gaussian distribution, central limit theorem, Mahalanobis distance, KL-divergence and Wasserstein distance. These concepts are heavily used in this work.

2.1.1. Distributions and Divergences

In this section, I will review some of the most important concepts of probability theory and statistics, which are used in work presented in this thesis.

Gaussian distribution:

The Gaussian distribution is one of the most widely used distributions in probability theory. The Gaussian (also known as Normal) distribution is a continuous probability distribution for a real-valued random variable.

The Gaussian distribution is characterized by two parameters, the mean (μ) and the standard deviation (σ). A Gaussian distribution is denoted by symbol $\mathcal{N}$. If a random variable x is distributed according to Gaussian distribution, then its density $p(x)$ can be expressed as below,

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

Central limit theorem:

If $x_1, x_2, ..., x_n$ are independently and identically distributed random variables with population mean μ and standard deviation σ, then according to the central limit theorem,

$$z = \frac{\sum_{i=1}^{n} x_i - n\mu}{\sqrt{n}\sigma} \sim \mathcal{N}(0,1) \text{ as } n \to \infty$$

Here, the distribution of x_i doesn't matter. The central limit theorem holds as long as the population mean and standard deviations can be estimated.

Chi-squared distribution:

A Chi-squared distribution with K degrees of freedom is the sum of squares of K IID standard normal random variables. If $z_1, z_2, ..., z_K$ are K IID standard normal random variables, then their sum of squares is Chi-squared distributed:

$$q_i = \sum_{i=1}^{K} z_i^2 \sim \chi^2(K)$$

Here, q_i is distributed according to the Chi-squared distribution with K degrees of freedom. If q_1 and q_2 are two chi-squared random variables with degrees of freedom K_1 and K_2, then,

$$q = q_1 + q_2 \sim \chi^2(K_1 + K_2)$$

Mahalanobis distance:

The Mahalanobis distance is a distance measure between a Gaussian distribution P and a point $\mathbf{x}$. It measures the number of standard deviations a point is away from the mean of the distribution. If P is a normal distribution with mean $\mu \in R^d$ and co-variance matrix $\mathbf{\Sigma} \in R^{d \times d}$, then for a point $\mathbf{x} \in R^d$, the mahalanobis distance between distribution P and point $\mathbf{x}$ is,

$$D_M(\mathbf{x}) = \sqrt{(\mathbf{x}-\mu)^T \mathbf{\Sigma^{-1}} (\mathbf{x}-\mu)}$$

The Mahalanobis distance is uniquely related to the confidence contour of a Gaussian distribution. For a univariate Gaussian, 68%, 95% and 99.7% values lie within one, two and three standard

deviations of the mean value. They correspond to a Mahalanobis distance of 1, 2 and 3, respectively. This unique mapping is characterized according to the inverse Chi-squared distribution. The squared Mahalanobis distance is distributed according to the Chi-squared distribution.

KL divergence:

In probability and statistics, the Kullback–Leibler divergence (also known as KL-divergence or relative entropy) is a measure of distance between two probability distributions. It is an expectation over the log difference between two probability densities. If $p(x)$ and $q(x)$ are two probability density functions over a random variable x, then the KL-divergence between the two distributions can be expressed as below,

$$\begin{aligned} D_{KL}(p||q) &= E_{x\sim p(x)}\Big[\log(p(x)) - \log(q(x))\Big] \\ &= \int_{x\sim p(x)} p(x)(\log(p(x)) - \log(q(x))) \\ &= \int_{x\sim p(x)} p(x)\log\left(\frac{p(x)}{q(x)}\right) \end{aligned}$$

It is important to note that $D_{KL}(p||q) \neq D_{KL}(q||p)$. $D_{KL}(p||q)$ is integrated over the support of $p(x)$, and is known as the forward KL. Whenever we employ a variational loss, we integrate over the support of the true distribution, from where a random variable is supposed to have been sampled. For two Gaussian distributions $p(x) = \mathcal{N}(\mu_1, \sigma_1^2)$ and $q(x) = \mathcal{N}(\mu_2, \sigma_2^2)$, there is closed form expression to compute the KL-divergence between p and q, which can be expressed as below,

$$\boxed{D_{KL}[\mathcal{N}(\mu_1, \sigma_1^2)||\mathcal{N}(\mu_2, \sigma_2^2)) = \log(\frac{\sigma_2}{\sigma_1}) + \frac{\sigma_1^2 - (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}}$$

Wasserstein distance:

The Wasserstein distance, also known as the Kantorovich–Rubinstein metric, is a distance function between two probability distributions. It is the minimum cost to turn one distribution into another. The closed form of the Wasserstein distance between two Gaussian distributions $p(x) = \mathcal{N}(\mu_1, \sigma_1^2)$ and $q(x) = \mathcal{N}(\mu_2, \sigma_2^2)$ can be expressed as below,

$$\boxed{W[\mathcal{N}(\mu_1, \sigma_1^2), \mathcal{N}(\mu_2, \sigma_2^2)] = (\mu_1 - \mu_2)^2 + \sigma_1^2 + \sigma_2^2 - 2\sigma_1\sigma_1}$$

2.1.2. Object detection

Object detection is one of the most fundamental and important problems in computer vision. In object detection, the goal is to localize and classify all the objects in a natural image, from a predefined set of categories. Deep learning has shown remarkable progress in learning visual

representations using labeled datasets. In this work, I evaluate proposed method extensively for object detection. In this section, I will briefly walk through modern deep learning based object detection architectures. Object detector architectures are mainly of two types. 1) Single stage object detection, 2) Two-stage object detection. In single stage object detection, features maps are directly mapped to final predictions, and there is no component for object proposals. In two stage object detection, potential regions with objects are identified, and then mapped to a fixed representation, and then we classify the object and regress the bounding box co-ordinates over that representation. Here, I briefly walk through some popular one-stage and two-stage architectures.

Backbones:

Backbones are convolutional neural networks, which can be used as feature extractors in computer vision tasks. In most cases they have been pre-trained on some large dataset such as ImageNet [**47**]. Their intermediate representation can provide rich feature representations of an input and there has been surge of interest in such CNN architectures. Most of these models can be used as feature extractors in object detection architectures. Here, I present a few popular ones below.

(1) **AlexNet** [**47**]: AlexNet was the first major CNN architecture, showing wide-scale success in image classification task on natural images. It employed ReLU non-linearity and max-pooling in the architecture.

(2) **VGG** [**91**]: VGGNet was proposed in 2014. It was deeper than AlexNet with standardized design rules for convolution and pooling layers. All the convolutions were 3x3 with stride and padding of 1, and a pooling layer, after which number of channels doubles. It proposed 2 variants with 16 and 19 layers including fully-connected layers.

(3) **GoogLeNet** [**93**]: GoogLeNet was proposed in 2015. The focus of this architecture was efficiency. It aggressively downsampled the input, and proposed the inception module where each layer was a neural network. It also proposed global average pooling which removed the necessity of fully connected layers, and reduced the number of parameters drastically. It also proposed an auxiliary classification head to effectively train the model.

(4) **ResNet** [**30**]: In Residual networks (ResNets), the idea was to have skip connections to make the gradient flow more easily. It enabled computation of the identity function and made training deeper networks easy. It came with two types of residual blocks corresponding to "Basic Block" and "Bottleneck block". It is still widely used in most of the computer vision tasks for deep learning.

(5) **ResNext** [**104**]: ResNext modified the Bottleneck residual block of the ResNet by replacing it with parallel pathways of the same computational complexity. It resulted in improvement of performance while maintaining same computational complexity.

(6) **S&E** [**39**]: The squeeze-and-excitation network modified the Residual block by adding an extra squeeze-and-excite branch to add global context to each residual block.

(7) **DenseNet** [**40**]: DenseNet constitutes dense blocks where each layer is connected to all previous layers within a dense block. It alleviates the vanishing gradient problem, and encourages feature propagation and reuse.

(8) **MobileNet** [**38**]: MobileNets were designed for efficiency. They replaced traditional convolution blocks with a much more efficient depth-wise separable convolution, which are computationally less expensive and enables easy deployment on mobile devices.

The above backbones are used as feature extractors for various computer vision tasks. The above list is non-exhaustive and there are many other architectures which are used in deep learning models. Next, I will briefly introduce one-stage and two-stage object detection architectures built using above backbones.

Single stage architectures:

(1) **YOLO** [**84**]: You only look once (YOLO), is a single-stage real time object detection architecture, which divides an input image into an $S \times S$ grid, with each grid cell responsible for predicting an object whose center falls within that grid. It directly regresses bounding boxes and predicts classes from the input image.

(2) **YOLOv2** [**85**]: YOLOv2 is an improved version of YOLO with convolutions with anchors, batch normalization [**42**] and multi-scale training. It also proposes a classification backbone which achieves higher accuracy at lower computation.

(3) **SSD** [**65**]: The single-shot detector is a one-stage architecture which predicts bounding box offsets for a set of predefined bounding boxes at various different scales within feature maps. These default bounding boxes have different scales and aspect ratios in the feature maps, and are placed at a regular interval over the feature map grid.

(4) **YOLOv3** [**86**]: YOLOv3 is an improved version of YOLOv2, which uses multi-label classification, a deeper feature extractor inspired from ResNet architecture.

(5) **RetinaNet** [**61**]: RetinaNet is a single-stage architecture with a "focal loss" for classification. RetinaNet fixes the problem of foreground-background class imbalance in object detection by proposing a modified loss function that would penalize foreground classification severely.

Two stage architectures:

(1) **R-CNN** [**19**]: R-CNN is a CNN based method for object detection where first region proposals are extracted using selective search [**98**], and a fixed length feature representation is extracted out of each region. This representation is used for classification and bounding box regression.

(2) **Fast R-CNN** [**18**]: Fast R-CNN is faster version of R-CNN where classification and bounding box regression are done over feature maps. It reduces computational load by having only a single forward pass over an image. It significantly reduces computation from R-CNN.

(3) **Faster R-CNN** [**87**]: Faster R-CNN improves over Fast R-CNN by incorporating a learning based region proposal pipeline. It learns to extract region proposals through anchors over feature maps from the backbone network. It removes the necessity of hand-engineered methods such as selective search for extracting region proposal. Faster RCNN is one of the most widely used object detection models.

(4) **Mask R-CNN** [**28**]: Mask R-CNN adds one more head in the second stage of Faster R-CNN to predict the mask of an object instance. It is one of the most popular methods for instance segmentation.

(5) **FPN** [**60**]: The feature pyramid network enables detecting objects at multiple different scales. It puts anchor boxes of multiple scales of the feature maps from the backbone networks, which enables accurate detection of objects with different sizes, scales and aspect ratios[1].

There are many other object detection methods using deep learning. The list above gives brief overview of various different architectures, and is not full.

mean average precision (mAP):

mAP is the most used object detector evaluation metric in the literature. In mAP, a detection is considered to be a true positive if its intersection over union (IoU) with the groundtruth object (annotated bounding box) is greater than a prescribed threshold and the probability of grountruth class is greater than a predefined threshold. In mAP, the precision of an object detector is estimated under this criteria, and this precision is computed for a range of IoU thresholds (from 0.5 to 0.95 at the interval of 0.05) and categories. The mean of these precision values across IoU thresholds and categories is the mean average precision of an object detector.

2.1.3. Uncertainty in Deep Learning

Reliably measuring the predictive uncertainty of blackbox models could be extremely useful in downstream robotics tasks. Previous studies demonstrate that softmax-based classification methods give highly brittle predictions and that they could be easy targets for adversarial attacks [**73, 5**]. This has resulted in a surge of interest in *bayesian deep learning* techniques [**25, 26, 69, 54**]. In the discussion of a *probabilistic* neural regression framework, various types/sources of uncertainties in the system are of central importance. Drawing from previous studies [**46, 69, 25**], the types

[1]Note that FPN is not strictly a two stage architecture. FPN is a modification in the way backbone feature maps are treated, and can be used in RCNN based methods such as Faster RCNN [**87**] or RetinaNet [**61**]

of uncertainties associated with a deep learning model are classified into two sub-categories. 1) In-distribution uncertainty and 2) Distributional uncertainty.

- ***In-distribution uncertainty*** or ***predictive uncertainty*** is categorized into *Aleatoric uncertainty* and *Epistemic uncertainty* [**14**]. Together, Epistemic and Aleatoric uncertainties induce *predictive uncertainty*.
  - **Aleatoric uncertainty** (data uncertainty) is the noise in the data presented to the system. It is "irreducible" (can only be estimated), and corresponds to the underlying entropy in the data distribution.
  - **Epistemic uncertainty** (knowledge uncertainty) is a result of noise in the model parameters and structure of the model(*structure uncertainty*). *Epistemic uncertainty* is considered to be reducible in nature, and can be reduced with availability of more data.
- ***Distributional uncertainty*** arises when models are presented with data outside of the training distribution. Many works have suggested that discriminative models could make overconfident predictions for an OOD input. Many attempts [**34, 54, 33, 59, 22, 35, 72, 69, 89**] have been made to understand and detect outlier data for neural networks.

For object detection, the data and model uncertainties can further be decomposed into **spatial** and **semantic** uncertainties, concerning the uncertainties in the location and label of objects respectively. In current probabilistic object detection setups, spatial uncertainty is characterized by a Gaussian distribution over the predicted bounding boxes. Semantic uncertainty is represented as a discrete probability distribution over a set of all possible labels. Classical object detection setups only deal with semantic uncertainty estimates while providing deterministic outputs for bounding box prediction ([**87, 28**]).

2.1.4. Existing Methods of Uncertainty Estimation and Calibration

Bayesian deep learning has seen significant work in recent years [**14, 15, 46, 67, 66, 54, 3, 36**]. Bayesian deep learning aims to design neural architectures capable of estimating the predictive uncertainty in the input data, as well as in the model parameters [**14**].

Epistemic uncertainty is typically estimated by either using ensembles of neural networks or by stochastic regularization at inference time (Monte-Carlo dropout) [**15, 54, 95**]. **Distributional** uncertainty is also being extensively studied, to detect out of training-distribution examples [**34, 54, 33, 59, 22, 35, 72, 69, 89**]. However, there is no direct approach to address distributional uncertainty for regression settings. Also closely related are the areas of zero-shot [**83, 1, 109, 81, 82, 23**] and few-shot learning [**100, 43, 105, 44, 12**]. Initial works to introduce uncertainty in neural networks used ensemble methods and dropout at test time [**15, 54**]. Many methods have been proposed to model distributional uncertainty for OOD [**69**]. Uncertainty estimation formulation can be broken down into: Epistemic and Aleatoric [**46**]. A loss attenuation

formulation to predict Aleatoric uncertainty was proposed. Generally, epistemic uncertainty is estimated using ensembles of networks whereas aleatoric uncertainty is predicted by the network. These **uncertainty estimation techniques** have since been applied to real-world problems such as object detection [**6, 26, 31**], semantic segmentation [**45, 41**], depth estimation [**45, 63, 2**], view synthesis [**71**], and medical imaging [**79, 53**]. These real-world problems even heightens the demands for calibrated uncertainty estimates.

In this work, the focus is specifically on **calibrating aleatoric uncertainty** estimates in neural regression. Several approaches have been proposed for calibrating neural classifiers [**22, 106, 107, 102, 70, 51, 64, 97, 108, 50, 68, 57**]. Typically, classifiers are calibrated using post-hoc techniques such as histogram binning [**106**], isotonic regression [**107**], or temperature scaling [**13**]. Such post-hoc techniques either require a large held-out calibration dataset [**13**] and/or add parameters to the model after training [**13, 107**]. [**68**] forms an approximate posterior distribution over the neural network parameters by fitting a Gaussian using stochastic average weighing over SGD iterates. It induces regularization which results in calibration of Bayesian deep models. [**102**] proposed a non-parametric approach using latent Gaussian process to calibrate multi-class classifier. [**70**] calibrates logits with decoupled Bayesian stage, where logits are fed to a Bayesian neural network to get calibrated logits. [**51**] proposed post-hoc scaling-binning calibrator to get calibrated output probabilities for classification problem. [**64**] proposed deep calibrator network to tackle generalized zero-shot learning, for calibrated scores of source as well as unseen data. Authors in [**97**] study calibration of mix-up training [**108**] for classification and discover that mixup training provides fairly calibrated classification probabilities. [**50**] proposed using Dirichlet distribution to calibrate multiclass classification model. [**52**] proposes RKHS kernel based calibration measure that trains model parameters in conjunction with negative log likelihood loss to achieve calibrated model.

Compared to classification, **neural regression** settings have received far less attention in terms of uncertainty estimation [**49, 92, 57, 13**]. Most techniques for calibrating regressors are applied post-hoc, and either require a large held-out calibration dataset [**13**] and/or add parameters to the model after training [**13, 107**]. The notion of quantile regression has been developed to quantifying the extent of calibration [**95, 37, 88, 96**]. Calibration is quantified by accuracy of quantile levels. Other methods, such as isotonic regression and temperature scaling have also been extended to the regression setting [**49, 13**]. In temperature scaling [**13**], a temperature parameter is introduced and trained on a validation dataset, to achieve calibrated confidence scores. In Isotonic regression, the predicted confidence scores are transformed using an isotonic function. This non-decreasing isotonic function is learned on a separate validation set to achieve calibrated transformed probabilities. Both isotonic regression and temperature scaling, require a separate held-out dataset for training, and also incur additional compute overhead. More recently, a *calibration loss* is proposed in [**13**]. This loss enforces the predicted variances to be equal to per-sample errors, thus grounding

each prediction. However, this takes on a *local view* of the calibration problem, and while individual samples might appear well calibrated, the overall distribution of the regressor errors exhibits a strong deviation from the expected target distribution.

A recent approach that is somewhat similar to ours in spirit is Gaussian process beta calibration (GP-beta) [**92**]. It is a post-hoc approach that employs a Gaussian process model (with a beta-link function prior) to calibrate uncertainties during inference. This requires computation of pairwise statistics, exacerbating inference time. f-Cal, on the other hand, is a simpler (and superior performing) loss function that enjoys the same inference time as typical Bayesian neural networks [**45**]. In [**8**] - distribution matching is performed using maximum mean discrepancy to achieve calibration - is also similar in spirit. This method was proposed for small datasets and didn't scale to our toy experiment, so I avoid comparing with it. I used original implementation of [**8**] but it failed to scale for the Bokeh dataset and Object detection. GP-beta, however, is a post-hoc approach that applies a Gaussian process model (assuming a beta-link function prior), while I derive a much simpler (and superior performing) loss function from first principles. Importantly, GP-beta incurs a significant inference time overhead since it computes pairwise statistics across test samples, while our proposed approach enjoys the same inference time as that of standard Bayesian neural nets [**45**]. Work in [**92**] relates the calibration from quantile regression perspective to distribution matching and shares same motivation as ours. However our approach significantly differ from them as they propose a post-hoc technique while f-Cal loss is designed based on definition of calibration. Authors in [**48**] tackle the problem of calibration in an online setting, where input can potentially be out of distribution. [**78**] regularize low entropic over-confident output distributions by penalizing it.

Chapter 3

f-Cal

In this section I present f-Cal, a simple, principled, and effective approach to obtaining calibrated aleatoric uncertainty estimates from neural regressors.

3.1. Problem Statement

We assume a regression problem over an i.i.d. labelled training dataset $\mathcal{D} \triangleq \{(x_i,y_i)\}_{i=1...|\mathcal{D}|}$ with $x_i \in \mathcal{X}$ where $\mathcal{X}$ is the (n-dimensional) input space and $y_i \in \mathcal{Y}$ where $\mathcal{Y} \subseteq \mathbb{R}^m$ is the output space. A *deterministic* model $f_d : \mathcal{X} \mapsto \mathcal{Y}$ [1] directly learns the mapping from the input to the output space by minimizing a loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ through empirical risk minimization where the empirical risk is given by

$$R_{emp}(f_d) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f_d(x_i),y_i). \tag{3.1.1}$$

Equation 3.1.1 is typically estimated over a mini-batch of size $N << |\mathcal{D}|$ during stochastic gradient descent (SGD). Following the notation in [**92**], we desire a *probabilistic* model $f_p : \mathcal{X} \mapsto \mathcal{S}_\mathcal{Y}$ where $\mathcal{S}_\mathcal{Y}$ is the space of all probability density functions $s(y)$ over $\mathcal{Y}$ ($s : \mathcal{Y} \mapsto [0,\infty)$ and $\int s(y)dy = 1$). The probability density function (PDF) is defined through its cumulative density function (CDF): $S(y) = \int_{-\infty}^{y} s(y')dy'$.

Uncertainty Calibration: Calibrated uncertainty estimates are those where the output uncertainties can be exactly interpreted as confidence intervals of the underlying target label distribution. This allows uncertainty estimates across multiple samples (and models) to be compared. Intuitively, we understand the notion of uncertainty calibration to mean that if we repeated a stochastic experiment a large number of times, that the empirical output distribution would approach the output density of our probabilistic model, f_p. However, in practice, it is impractical to run multiple forward passes over every possible input. We desire that the probabilistic model outputs this

[1] In practice these models are assumed to be neural networks with parameters θ but I omit the θ for clarity at this stage.

calibrated density directly from one pass of the model. Using our definitions and adapting from [**92**], we can define what we desire in terms of calibration in the case of a deep neural regressor as follows:

Definition 1 (**Uncertainty Calibration**). *A neural regressor f_p is calibrated if and only if :*

$$p(Y \leq y|s(y)) = \int_{-\infty}^{y} s(y')dy' \; \forall y \in \mathcal{Y} \tag{3.1.2}$$

If we can assume that the noise is sampled from a parametric distribution $s(y;\phi)$, then the probabilistic model need only output the parameters associated with each sample. In this case, we can consider the model to be calibrated if and only if the aggregated error statistics over multiple outputs of a model align with the parameters predicted by the model.

Definition 2 (**Consistency:**). *A neural regressor f_p is consistent for any arbitrary confidence bound c if*[2]*,*

$$p(Y \leq y|s(y)) \leq c \tag{3.1.3}$$

3.1.1. Aleatoric Uncertainty Estimation

Aleatoric uncertainty is the irreducible uncertainty in the underlying process and can only be estimated. This uncertainty is usually *heteroscedastic*, i.e., observation noise is not constant across all inputs. The most widely used technique for estimating heteroscedastic aleatoric uncertainty is *loss attenuation* [**46, 76**], which performs maximum likelihood estimation by minimizing the negative log-likelihood loss:

$$R_{emp}(f_p) = -\frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_{\text{attenuation}}(f_p(x_i),y_i) \tag{3.1.4}$$

$$= -\frac{1}{N}\sum_{i=1}^{N} \log s(y_i; f_p(x_i)) \tag{3.1.5}$$

Here $f_p(x_i)$ outputs the parameters of the distribution. For example, if the aleatoric uncertainty is characterized by a Gaussian random variable ($\phi \triangleq (\mu,\sigma)$), the above expression becomes,

$$R_{emp}(f_p) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{2}\left(\frac{(y_i-\mu_i)^2}{\sigma_i^2} + \log \sigma_i^2\right) \tag{3.1.6}$$

[2]Referring to Fig. 1.1, the requirement for calibration is more stringent than that of consistency, which is a one-way constraint at an arbitrary confidence bound c.

We refer to the loss derived in Eq. 3.1.6 as the **NLL** loss in the experiments. However, probabilistic neural regressors trained using this negative log-likelihood typically lack **calibration** according to Def. 1. The estimated uncertainties often over/under-estimate the true aleatoric uncertainty.

3.2. Calibration as distribution matching

Existing approaches that have demonstrated success on calibration have typically operated in a post-hoc manner [**92**]; once a neural regressor has been trained, these techniques require a separate *calibration* phase. Some techniques such as temperature scaling [**13**] also crucially require large held-out *calibration datasets*.

Existing calibration approaches lie on a broad spectrum. Calibration losses [**13**] are quite easily applied at train time (as an additional loss term), but still operate at a per-sample level and are unable to capture *global* properties of the label error distribution. At the other end of the spectrum lie post-hoc calibration approaches such as isotonic regression [**107**] and GP-beta models [**92**]. These end up being expensive at inference time, as they require additional computation to output calibrated uncertainty estimation. This work brings the best of both these worlds by proposing f-Cal, a principled approach to aleatoric uncertainty calibration that is extremely simple to implement. The idea is to approximate the empirical posterior error distribution of the neural regressor errors (residuals) $(f(x_i) - y_i)$ by a simpler, tractable target distribution Q.

Distributional calibration means that empirical distribution of target variable Y should agree with output of the model conditioned on modeling assumption, which follows from the core definition of calibration (Eq. 1). Following the definition of distributional calibration (Definition 1), f-Cal formulates a variational minimization objective to calibrate the uncertainty estimates from a neural regressor. This method seek to approximate the empirical posterior over some canonical transformation of the target variables Y by a simpler (tractable) target distribution Q (modeling choice). This enables us to leverage an abundant class of distribution matching metrics—f-divergences to formulate a loss function enforcing distributional calibration. This allows us to formulate a variational optimization problem involving the empirical (proposal) distribution and the desired calibration (target) distribution. For tractable inference, I assume i.i.d. minibatches of training data and instead impose distribution matching losses over empirical error residuals across each batch[3].

This work assumes that we can transform each training sample output distribution to some canonical element of the distribution family. For instance, Gaussian random variables are canonicalized by centering the distribution (subtracting the output label), followed by normalization (scaling the result by the variance). These canonical elements are used (in conjunction with the

[3]Note that this method does not strictly require minibatches. For tasks such as object detection and depth estimation, we can always leverage multiple regressed outputs from a single input for distribution matching.

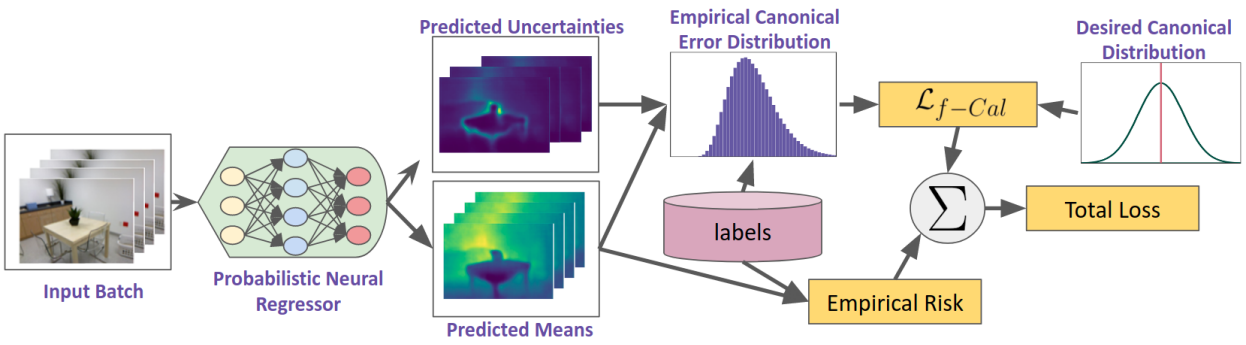


Fig. 3.1. Block diagram of f**-Cal**. In addition to deterministic regression loss terms (such as $L1$ and $L2$ errors), I impose a distribution matching constraint over the error residuals across a mini-batch. By enforcing the distribution of mini-batch errors to match a target calibrating distribution, I obtain significantly superior results compared to prior calibration approaches, most of which perform post-hoc calibration or assume a large held-out calibration dataset. f-Cal has no inference time overhead, even for dense regression tasks such as depth estimation.

ground-truth labels) to determine the *empirical* error distribution. f-Cal then performs distribution matching across this empirical and a target distribution (a modeling choice). I add an additional variational loss over the mini-batch errors that minimizes the distance between the empirical distribution generated by the errors in the output predictions and the ground truth canonical distribution that they are meant to be sampled from. Doing so will enforce calibration according to Def. 1. as long as our error statistics are well represented by the parametric class of distributions that f-Cal is minimizing the distance to.

I assume a neural regressor $f_\theta : \mathcal{D}^X \mapsto \mathcal{D}^Y$ parameterized by learnable weights θ and with real-valued domains and co-domains $\mathcal{D}^X$ and $\mathcal{D}^Y$ respectively. I desire a method for calibrating aleatoric uncertainty that is simple but principled, and fast but robust. I seek to train a probabilistic model such that the prediction errors of f_θ are distribution-calibrated (*cf.* Def. 1).

3.3. f-Cal Algorithm

Here, we use y_i to denote the ground-truth regression target corresponding to the ith prediction, and by ϕ_i the ith regressor output containing parameters of the modeling distribution. Given a mini-batch containing N inputs x_i, a probabilistic regressor predicts N sets of parameters $f_p(x_i) = \phi_i$ to the corresponding probability distribution $s(y_i; \phi_i)$. Define $h : \mathcal{Y} \times \Phi \mapsto \mathcal{Z}$ as the function that maps the target random variable y_i to a random variable z_i[4]. which follows a known canonical distribution. Since these residuals $\{z_1, z_2, \ldots, z_N\}$ must ideally follow a chosen calibrating (target) distribution Q, I compute the empirical statistics of the residuals to fit a proposal distribution P of the same family as Q. We define a variational loss function that minimizes the f-divergence

[4]For a Gaussian target distribution, $\phi_i \triangleq (\mu_i, \sigma_i)$, h can be chosen to be $\frac{y_i - \mu_i}{\sigma_i}$

between these two distributions. Examples for the Gaussian case are give in Sec. 3.3.1. Since modern deep learning frameworks (such as PyTorch [**77**]) provide out-of-the-box differentiable functions for distribution matching, and because the residuals are computed in a fully differentiable manner (e.g. by maintaining a differentiable computation graph all the way from the inputs), gradients with respect to this loss function can be computed using standard autodifferentiation tools [**77**].

In summary, I propose a distribution matching loss function that augments typical supervised regression losses, and results in the neural regressor being calibrated to the target distribution. Calibration by itself is not sufficient to predict accurate regression output [**49**], predicted mean and variance can be abruptly large and still they still can give calibrated results but won't be accurate as empirical risk is poor. Hence, the confidence bound is required to be as tight as possible. For this I also add term to minimize empirical risk($\mathcal{L}_{\text{supervised}}$). In regression setup, we try to minimize empirical risk through Maximum likelihood estimate. Empirical risk term bounds the mean of the prediction. In section 3.1, we defined training dataset as $\mathcal{D} \triangleq \{(x_i,y_i)\}_{i=1...|\mathcal{D}|}$, we can write *empirical distribution* of the dataset as below,

$$\hat{p}_{\mathcal{D}}(x) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \delta(x, x_i) \tag{3.3.1}$$

Here, $\delta(.)$ is a Kronecker delta function to represent a point mass. If we have a parameterized likelihood model with model parameters θ, then we can express the log likelihood over a minibatch of N examples(N $<<$ $|\mathcal{D}|$) as below,

$$\sum_x \hat{p}_{\mathcal{D}}(x) \log p(x|\theta) = \frac{1}{N} \sum_{i=1}^{N} \log p(x_i|\theta) \tag{3.3.2}$$

From [**74**], we know that maximizing likelihood is equivalent to minimizing KL-divergence between empirical data distribution and estimated distribution. Hence, empirical risk objective function can be expressed as below,

$$\mathcal{L}_{\text{supervised}} = -\frac{1}{N} \sum_{i=1}^{N} \log p(x_i|\theta) \tag{3.3.3}$$

$$= KL(\hat{p}_{\mathcal{D}}(x)|| \log p(x|\theta)) \tag{3.3.4}$$

$\mathcal{L}_{\text{supervised}}$ could be any maximum likelihood loss function. For popular losses such as L2 or L1, we can have interpretations in terms of priors. If $\mathcal{L}_{\text{supervised}}$ is L2 loss, it can be thought of as Maximizing log-likelihood of a Gaussian with unit variance(see 7.1.1 of [**21**]). If $\mathcal{L}_{\text{supervised}}$ is L1 loss, it can be interpreted as maximizing log-likelihood of a Laplace distribution with unit scale parameter(see 7.1.2 of [**21**]). In probabilistic regression setup, we do not assume homoscedastic

noise, and also learn variance/scale parameters, where maximum likelihood takes form of equation 3.1.4.

In addition to empirical loss of form $\mathcal{L}_{\text{supervised}}$, we also introduce distribution matching loss to enforce calibration. We denote estimated distribution of our mini-batch residuals with P, and our objective is to make it resemble the target distribution Q. In subsequent sections, with a specific case of Gaussian modeling assumption, forms of P and Q will become clearer. Here, the distribution matching objective is,

$$\mathcal{L}_{f\text{-Cal}} = D_f(P||Q) \tag{3.3.5}$$

Here, D_f is a differentiable f-divergence – a measure of overlap between distributions P and Q. We take the weighted combination of empirical objective and calibration objective to construct our final loss function. Our final loss function takes the shape of the following form,

$$\begin{aligned} \mathcal{L} &= (1-\lambda)\mathcal{L}_{\text{supervised}} + \lambda\mathcal{L}_{f\text{-Cal}} \\ &= (1-\lambda)KL(\hat{p}_D(x)||\log p(x|\theta)) + \lambda D_f(P||Q) \end{aligned} \tag{3.3.6}$$

$\mathcal{L}$ is the final loss objective and λ is a hyper-parameter to weigh the two loss terms. Our estimates are calibrated due to the distribution matching loss terms in Eq. 3.3.6. We use KL-divergence and Wasserstein distance as f-divergences in this work. f-Cal can be applied to any probabilistic neural regressor, in just about 10-15 lines of PyTorch code. The training process is virtually unchanged, enabling easy integration into almost any regression task.

3.3.1. f-Cal for Gaussian calibration

The f-Cal framework is generic and can be applied to arbitrary—even empirical—distributions. In this section I consider the case when the distribution $s(y_i;\phi_i)$ is Gaussian with $\phi_i \triangleq (\mu_i,\sigma_i)$. The variance σ_i^2 denotes the aleatoric uncertainty in this case. The error residuals are computed as $z_i = \frac{y_i-\mu_i}{\sigma_i}$, where μ_i and σ_i are predicted mean and the standard deviation of the ith Gaussian output from the neural network for each input x_i.

$$\begin{aligned} &y_i \sim \mathcal{N}(\mu_i,\sigma_i^2) \\ &\implies z_i \sim \mathcal{N}(0,1) \text{ (known canonical distribution)} \end{aligned}$$

3.3.2. Chi-squared Hyper-constraints

One may apply several transforms to the random variables y_i and impose distributional *hyper*-constraints over the transformed variables. In practice, I find that this can improve the stability of the training process and enforces stronger calibration constraints. The variable $z_i = \frac{y_i - \mu_i}{\sigma_i}$ follows standard normal distribution. We can construct a chi-squared random variable with degrees of freedom of K, by fusing K such z_i's in the following way,

$$\begin{aligned} & z_i \sim \mathcal{N}(0,1) \\ \implies & q = \sum_{i=1}^{K} z_i^2 \sim \chi_k^2 \end{aligned} \tag{3.3.7}$$

Algorithm 1: f-Cal for Gaussian uncertainties

Input: Dataset D, probabilistic neural regressor, f_p, degrees of freedom K, batch size N, number of samples for hyper-constraint H

for $i = 1 \dots N$ **do**
  $(\mu_i, \sigma_i) \leftarrow f_p(x_i)$
  $z_i \leftarrow \frac{y_i - \mu_i}{\sigma_i}$
end
$C = \emptyset$ // Samples from Chi-squared distribution
for $i = 1 \dots H$ **do**
  // Create a chi-squared hyper-constraint
  $q_i \leftarrow \sum_{j=1}^{K} z_{ij}^2, z_{ij} \sim \{z_1 \cdots z_N\}$
  C.append(q_i)
end
$P \leftarrow$ Fit-Chi-Squared-Distribution(C)
$\mathcal{L}_{f\text{-Cal}} \leftarrow D_f(P||\chi_K^2)$
return $\mathcal{L}_{f\text{-Cal}}$

In context of regression, I am fitting a univariate Gaussian for each prediction. Hence, over a batch of predictions, we expect the model to output many standard normal random variables. In theory, we can construct a chi-squared random variable from many standard normal random variables as shown in equation 3.3.7. In this work, we have a batch of outputs, each following a univariate Gaussian distribution, we normalize the output prediction to construct a standard normal random variable according to $h(\mu_i, \sigma_i, y_i) = \frac{y_i - \mu_i}{\sigma_i}$. For higher degrees of freedom, the chi-squared distribution follows another interesting property as well. When degrees of freedom K is large(>50), the chi-squared distribution converges on a Gaussian distribution with mean K and variance 2K [**4**], according to central limit theorem.

$$\lim_{K\to\infty} \frac{\chi^2_K - K}{\sqrt{2K}} \to \mathcal{N}(0,1) \tag{3.3.8}$$

$$\implies \lim_{K\to\infty} \chi^2_K \to \mathcal{N}(K,2K) \tag{3.3.9}$$

I exploit this property to construct the target distribution. I construct multiple chi-squared random variables with degrees of freedom K from canonicalized random variables(z_i) predicted over the batch of inputs. These chi-squared random variables are treated as samples of univariate Gaussian distribution with mean K and variance 2K. I construct proposed distribution from this chi-squared random variables, and perform distributional matching between our empirical distribution and target distribution($\mathcal{N}(K, 2K)$). Subsequently, I impose hyper-constraints that compute the sum-of-squared error residuals q, and enforce the resulting distribution to be chi-squared with parameter K i.e $q \sim \chi^2_K$

So, instead of employing distribution matching over z I use distribution matching over variable q. So I use our loss function as distribution matching on variable q which can be expressed as below.

$$\mathcal{L}_{f\text{-Cal}} = D_f(P \parallel \chi^2_k) = D_f(P \parallel \mathcal{N}(K, 2K)) \tag{3.3.10}$$

where $f \in F$ and F is the family of different distance/divergence function on probability distribution. We did our experiment with F = *{KL-Divergence, Wasserstein Distance}* .

In practice, this variation of the central limit theorem for Chi-squared random variables holds for moderate values of K (i.e., $K > 50$). This is practical to ensure, particularly in dense regression tasks such as bounding box object detection (where hundreds of proposals have to be scored per image) and per-pixel regression. We summarize the Algorithm for generating the $\mathcal{L}_{f\text{-Cal}}$ in Algorithm 1. This loss is then combined with the typical empirical risk as given by Equation 3.3.6.

3.4. Derivation - f-Cal loss:

In this section, I derive the f-Cal loss for KL-divergence and Wasserstein distance. Let's say that the neural regressor is predicting N regression variables over an entire batch of inputs. I use the following notations for our predictions and ground-truth.

- predicted means: $\mu_1, \mu_2, ..., \mu_N$
- predicted variance: $\sigma^2_1, \sigma^2_2, ..., \sigma^2_N$
- Ground truth: $y_1, y_2, ..., y_N$
- K = degrees of freedom of a chi-squared random variable, generally, K > 50.

Here, N is assumed to be larger, generally N > 1000. Here K is a hyper-parameter.

- $z^2_i = \frac{(y_i - \mu_i)^2}{\sigma^2_i} \sim \chi^2_1$; i = {1, 2, ..., N}, are Mahalanobis distances with DoF 1.

$$q_i = \sum_{j=1}^{K} z_{ij}^2 = \sum_{j=1}^{K} \frac{(y_{ij} - \mu_{ij})^2}{\sigma_{ij}^2}$$

$$y_{ij} \sim \{y_1, y_2, ..., y_N\}$$

$$\boxed{q_i = \sum_{j=1}^{K} \frac{(y_{ij} - \mu_{ij})^2}{\sigma_{ij}^2} \sim \chi_K^2}$$

Here, μ_{ij} and σ_{ij} are predictions corresponding to y_{ij}. y_{ij} is uniformly sampled without replacement. We get H such q_i, each distributed as chi-squared random variable with DoF K. H is a hyper-parameter, which is number of chi-squared samples.

The distribution resulting out of these H random variables is a chi-squared distribution. For K > 50, $\chi_K^2 \rightarrow \mathcal{N}(K, 2K)$. The empirical mean ($\mu_{\chi_K^2}$) and variance ($\sigma_{\chi_K^2}^2$) of the Chi-squared distribution can be written as below,

$$\mu_{\chi_K^2} = \frac{1}{H} \sum_{i=1}^{H} q_i = \sum_{i=1}^{H} \sum_{j=1}^{K} \frac{(y_{ij} - \mu_{ij})^2}{\sigma_{ij}^2}$$

$$\sigma_{\chi_K^2}^2 = \frac{1}{H-1} \sum_{i=1}^{H} (q_i - \mu_{\chi_K^2})^2$$

$$\implies \sigma_{\chi_K^2}^2 = \frac{1}{H-1} \sum_{i=1}^{H} \left(\sum_{j=1}^{K} \frac{(y_{ij} - \mu_{ij})^2}{\sigma_{ij}^2} - \frac{1}{K} \sum_{i=1}^{H} \sum_{j=1}^{K} \frac{(y_{ij} - \mu_{ij})^2}{\sigma_{ij}^2} \right)^2$$

In the above equation, we get empirical means and variance sof our Chi-squared distribution over a batch of predictions. According to the central limit theorem, the Chi-squared distribution with degrees of freedom K(K $\geq$ 50) follows a Gaussian distribution mean K and variance 2K, hence the target mean($\hat{\mu}_{\chi_K^2}$) and target variance($\hat{\sigma}_{\chi_K^2}^2$) are:

$$\hat{\mu}_{\chi_K^2} = K, \hat{\sigma}_{\chi_K^2}^2 = 2K$$

Proposal distribution: $\boxed{p(x) = \mathcal{N}(\hat{\mu}_{\chi_K^2}, \hat{\sigma}_{\chi_K^2}^2)}$

Target distribution: $\boxed{q(x) = \mathcal{N}(\mu_{\chi_K^2}, \sigma_{\chi_K^2}^2)}$

Here, we have statistics of the proposal distribution($p(x)$) and target distribution($q(x)$). The closed form KL-divergence and Wasserstein distance between two univariate normal distributions can be expressed as below

$$\text{KLD} = KL(p||q) = \frac{1}{2} \log \left(\frac{\hat{\sigma}_{\chi_K^2}^2}{\sigma_{\chi_K^2}^2} \right) + \frac{\sigma_{\chi_K^2}^2 + (\mu_{\chi_K^2} - \hat{\mu}_{\chi_K^2})^2}{2\hat{\sigma}_{\chi_K^2}^2} - \frac{1}{2}$$

$$\text{KLD} = KL(p||q) = \frac{1}{2}\log\left(\frac{2K}{\frac{1}{H-1}\sum_{i=1}^{H}\left(\sum_{j=1}^{K}\frac{(y_{ij}-\mu_{ij})^2}{\sigma_{ij}^2} - \frac{1}{K}\sum_{i=1}^{H}\sum_{j=1}^{K}\frac{(y_{ij}-\mu_{ij})^2}{\sigma_{ij}^2}\right)^2}\right) + \left(\frac{2K + (\sum_{i=1}^{H}\sum_{j=1}^{K}\frac{(y_{ij}-\mu_{ij})^2}{\sigma_{ij}^2} - K)^2}{4K}\right) - \frac{1}{2}$$

$$\text{W-dist} = W(p,q) = (\mu_{\chi^2_K} - \hat{\mu}_{\chi^2_K})^2 + (\hat{\sigma}^2_{\chi^2_K} + \sigma^2_{\chi^2_K} - 2\sigma_{\chi^2_K}\hat{\sigma}_{\chi^2_K})$$

$$\text{W-dist} = W(p,q) = (\sum_{i=1}^{H}\sum_{j=1}^{K}\frac{(y_{ij}-\mu_{ij})^2}{\sigma_{ij}^2} - K)^2 + (2K + \frac{1}{H-1}\sum_{i=1}^{H}\left(\sum_{j=1}^{K}\frac{(y_{ij}-\mu_{ij})^2}{\sigma_{ij}^2} - \frac{1}{K}\sum_{i=1}^{H}\sum_{j=1}^{K}\frac{(y_{ij}-\mu_{ij})^2}{\sigma_{ij}^2}\right)^2 - 2\sqrt{\frac{1}{H-1}\sum_{i=1}^{H}\left(\sum_{j=1}^{K}\frac{(y_{ij}-\mu_{ij})^2}{\sigma_{ij}^2} - \frac{1}{K}\sum_{i=1}^{H}\sum_{j=1}^{K}\frac{(y_{ij}-\mu_{ij})^2}{\sigma_{ij}^2}\right)^2}\sqrt{2K})$$

Chapter 4

Experiments

We conduct a number of large-scale experiments on both synthetic and real-world datasets. I report the following key findings which we elaborate on in the remainder of this section.

- f-Cal achieves significantly superior calibration compared to existing methods for calibrating aleatoric uncertainty estimates.
- These performance trends are consistently observed across multiple regression setups, neural network architectures, and dataset sizes.
- I show the trade-off between deterministic and calibration performance by varying the λ hyper-parameter. This trade-off has been established in previous literature [**13, 22**]
- I demonstrate that consistency is a by product of calibration. I explicitly design metrics to estimate consistency, and show high consistency for f-Cal.

4.1. Regression tasks

We consider 3 regression tasks to evaluate f-Cal.

- A synthetic disc tracking dataset (Bokeh), where we track location of one disk amidst a set of distractor disks.
- Object detection: KITTI [**17**] and Cityscapes [**7**]
- Depth estimation: KITTI [**17**]

These tasks are chosen to span the gamut of regression tasks relevant for robotics applications: sparse (one output per image in disc tracking), semi-dense (object detection), and pixelwise (fully) dense (depth estimation). Unless otherwise specified, we model aleatoric uncertainty using heteroscedastic Gaussian distribution.

4.2. Baselines

We compare f-Cal models with the following baselines[1]:

[1]It is important to note that GP-beta [**92**] and isotonic regression [**107**] scaled to our synthetic dataset task, but didn't scale to large, real world datasets.

- **Loss attenuation** [**76, 14**]: The *de facto* method for aleatoric uncertainty estimation. Here, we train the model with NLL Loss(eq 3.1.6), to regress over mean and standard deviation estimates.
- **Temperature scaling** [**13**]: A post-hoc calibration technique that requires a *large, held-out* calibration dataset. In this method, we train the model using NLL Loss, then introduce a temperature parameter, which scales the standard deviation. This parameter is learned using a held-out validation set.
- **Isotonic regression** [**107**]: A post-hoc calibration technique that requires expensive inference time (quadratic in the number of test examples).
- **Calibration loss** [**13**]: The loss function proposed in [**13**]; induces calibration at a *local* (per-sample) level by making variance close to per-sample error.
- **GP-beta** [**92**]: Most recent method in calibrated aleatoric uncertainty estimation (inference time is quadratic in the number of test examples).[2]
- **MMD** [**8**]: Current state of the art art for small scale regression problems. It uses maximum mean discrepancy based distribution matching scheme to achieve calibration[3].
- f**-Cal-KL**: Our approach trained using KL-divergence as the distance metric for distribution matching.
- f**-Cal-Wass**: Our approach trained using Wasserstein distance as the metric for distribution matching.

4.3. Evaluation metrics

There are several existing metrics to evaluate the calibration of the model. In this work, we extensively evaluate f-Cal models against multiple existing calibration metrics. In particular, we benchmark it with expected calibration error (ECE) and maximum calibration error (MSE) [**75**]. In addition to these metrics, we also report negative log likelihood (NLL) [**27**]. We report KL-divergence (KLD) and Wasserstein distance (W-dist) between output probability distribution and target probability distribution ($\mathcal{N}(K, 2K)$) for object detection. We also propose modifications of MSE and ECE to quantify miscalibration between the target and output distributions. For ECE/MSE, we group output Chi-squared random variables (q) into S intervals, each of size $\frac{1}{S}$, and compare the empirical frequency of samples with the true frequency for each bin. Let B_s be the set of indices for interval s. If we have total of P Chi-squared samples over our test set, ECE and MSE can be expressed as below,

[2]GP-beta is not state of the art in our toy setup because of complexity of the task at hand. Original work was evaluated for relatively simpler tasks and small datasets, owing to complexity of Gaussian processes. While our toy experiment is relatively complex task. GP-beta didn't scale for any real-world tasks of object detection or depth estimation.

[3]Being designed for very low data regimes, it failed to solve any of our tasks considered. We communicated with the authors and used their original implementation, but it failed to scale for any of our tasks.

$$ECE_{\chi^2_K} = \sum_{s=1}^{S} \frac{|B_s|}{P} \left\| \frac{1}{S} - \frac{|B_s|}{P} \right\| \tag{4.3.1}$$

$$MCE_{\chi^2_K} = \max_{s \in \{1,2,..,S\}} \left\| \frac{1}{S} - \frac{|B_s|}{P} \right\| \tag{4.3.2}$$

For large values of K, when equation 3.3.9 holds, the fraction of samples falling in a particular bin can be found using the inverse CDF of a Gaussian distribution. In addition to these metrics, we also present a reliability diagram for our experiments, through which we can qualitatively see the amount of miscalibration. Similar to our proposed ECE for Chi-squared random variables, we also present reliability diagram over the output Chi-squared distribution.

4.4. Bokeh: synthetic disc-tracking benchmark

Since ground truth estimates of aleatoric uncertainty are extremely challenging to obtain in practice, we first validate our proposed approach in simulation.

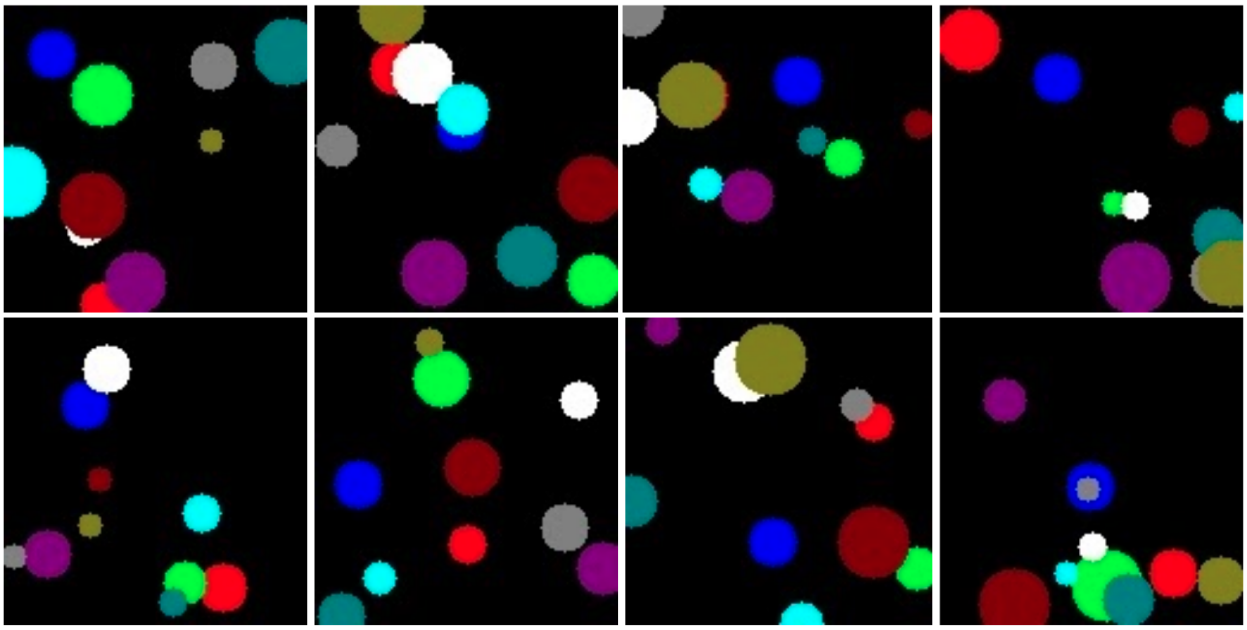


Fig. 4.1. Bokeh dataset: A synthetic dataset where task is to localize red disk among other distractor disks.

Dataset: We design a synthetic dataset (Fig. 4.1) akin to [**24**] for a *disc-tracking* task. We design Bokeh with the complexities of a typical regression problem for computer vision in mind. The goal is to predict the 2D location of the centre of a red disc from an input image containing other distractor discs. All discs are sampled from a known data-generating distribution[4]. Randomly coloured

[4]This is important, as it eliminates the usual handicaps with real data, where one may not have access to the label-generating distribution.

Approach	SmoothL1 (GT)	SmoothL1	ECE(z)	ECE(q)	NLL
NLL Loss[76]	1.67 ± 0.35	2.16 ± 0.36	0.016 ± 0.004	0.854 ± 0.109	-1.31 ± 0.14
Calibration Loss[13]	1.68 ± 0.32	2.19 ± 0.31	0.013 ± 0.004	0.72 ± 0.232	-1.36 ± 0.20
Temperature Scaling[13]	1.67 ± 0.35	2.16 ± 0.36	0.007 ± 0.001	0.128 ± 0.098	-1.38 ± 0.12
Isotonic Regression[107]	1.60 ± 0.32	2.05 ± 0.31	0.019 ± 0.005	0.909 ± 0.049	-1.31 ± 0.16
GP-Beta[92]	1.60 ± 0.31	2.06 ± 0.31	0.018 ± 0.009	0.837 ± 0.147	-1.31 ± 0.20
f-Cal-KL (ours)	1.67 ± 0.32	2.16 ± 0.32	0.005 ± 0.001	0.068 ± 0.007	-1.43 ± 0.08
f-Cal-Wass (ours)	1.59 ± 0.28	2.08 ± 0.28	0.006 ± 0.001	0.037 ± 0.0022	-1.45 ± 0.05
NLL Loss[76]	1.44 ± 0.34	1.54 ± 0.34	0.0173 ± 0.0027	0.9183 ± 0.0238	-1.60 ± 0.21
Calibration Loss[13]	1.46 ± 0.29	1.57 ± 0.31	0.0113 ± 0.0022	0.7611 ± 0.0129	-1.68 ± 0.19
Temperature Scaling[13]	1.44 ± 0.34	1.54 ± 0.34	0.0082 ± 0.0019	0.0922 ± 0.0235	-1.70 ± 0.15
Isotonic Regression[107]	1.38 ± 0.27	1.49 ± 0.27	0.0205 ± 0.0045	0.9378 ± 0.0124	-1.57 ± 0.24
GP-Beta[92]	1.39 ± 0.26	1.49 ± 0.27	0.0221 ± 0.0082	0.9348 ± 0.0208	-1.54 ± 0.28
f-Cal-KL (ours)	1.42 ± 0.33	1.52 ± 0.33	0.0056 ± 0.0011	0.0921 ± 0.0135	-1.76 ± 0.15
f-Cal-Wass (ours)	1.43 ± 0.34	1.54 ± 0.34	0.0079 ± 0.0016	0.0799 ± 0.0095	-1.75 ± 0.15

Table 4.1. Bokeh - disc-tracking: We evaluate several baselines under homoscedastic (top-half) and heteroscedastic noise (bottom-half). f-Cal is consistently better calibrated (lower ECE) compared to all considered baselines outperforms all considered baselines. Notably, this improved calibration comes without any in terms of calibration, without sacrificing regression performance. SmoothL1 and SmoothL1 (GT) scores have been scaled by 1000 and ECE scores by 100.

discs are added to the image to occlude the red disc and act as distractors. The locations and radii of these distractor discs are sampled from a uniform distribution. We generate homoscedastic and heteroscedastic variants of the dataset.

We introduce noise to our ground-truth labels and create two separate synthetic datasets: one where noise is momoscedastic and the other where it is heteroscedastic. Noise in x and y are sampled independently from a Gaussian distribution. In the case of homoscedastic noise, the noise generating distribution is $\mathcal{N}(0, \sigma)$, where σ is a fixed value. On the other hand, heteroscedastic noise is generated from the distribution $\mathcal{N}(0, \sigma(x))$, where σ is a function of the input image x. $\sigma(x)$ depends on the proximity of the distractor discs in relation to the red disc. Simply put, if the distractor discs are nearby or occluding the red disc, the $\sigma(x)$ value will be high and conversly it will be low when they are far away. We split the dataset into training, validation and test sets in proportion of 3:1:1. All disc locations are sampled from a known data-generating distribution.

Models: We use a 3-layer ConvNet architecture extended to predict uncertainty.

Training: First, We train a purely deterministic model with SmoothL1 loss for both the homoscedastic and heteroscedastic cases. This becomes a good baseline to compare the deterministic performance of the calibration methods. Next, We train a loss attenuation model which acts as our primary baseline and also is used as a good initialization for all f-Cal training. We train our model using both KL divergence and Wasserstein distance as the distance measures between the ground truth Chi-squared distribution and the predicted Chi-squared empirical distribution. All the baseline calibration methods (Table 4.1) were initialized with NLL loss trained weights.

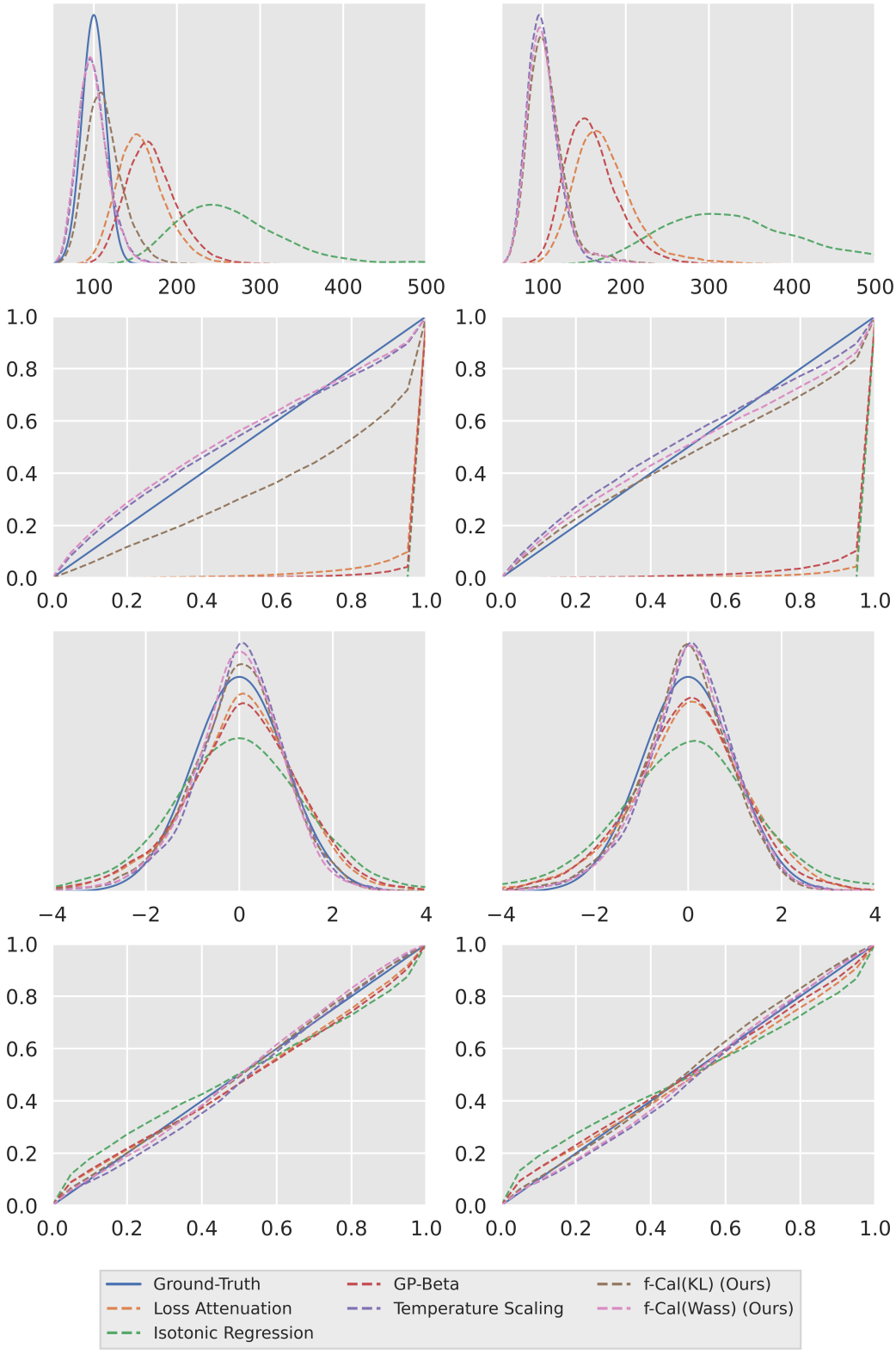


Fig. 4.2. Distributional and Reliability Diagrams (Bokeh): (Col 1) with homoscedastic Noise (Col 2) with heteroscedastic noise. (Row 1) shows the Chi-squared distributional comparison of the predicted outputs with the target. (Row 2) shows the reliability diagram with Chi-squared distribution. (Row 3) shows the standard-normal distributional comparison. (Row 4) shows the reliability curve with standard-normal variables.

Results: Table 4.1 compares f-Cal to the aforementioned baselines, evaluating *performance* (i.e., the accuracy of the estimated mean) and *calibration* quality. We report the performance (SmoothL1 error) for both the *noise-free* ground-truth[5], and the *noisy* ground-truth (accounting for label generation error). We ran each algorithm over 5 seeds for statistical significance. The deterministic performance with a SmoothL1 error (denoted as L1 in short in Table 4.1) for both the actual ground-truth (L1(GT)) and the noisy ground-truth (L1). We see in Table 4.1 that f-Cal outperforms all baselines considered by a significant margin. From Fig. 4.2, we can see qualitatively that the output distributions from f-Cal trained models are much closer to the ground truth distribution than the baselines. We can also see from the reliability curves, that f-Cal models are much closer to the diagonal line representing perfect calibration than the baselines. Apart from the baseline methods shown in Table4.1, we also trained MMD [**8**], but the model fails to converge on Bokeh which is the simplest of the three datasets used in this work.

Calibration scores like ECE standard normal (ECE(z)) and ECE Chi-squared (ECE(q)) show that f-Cal is able to outperform all the baseline methods for both the homoscedastic and heteroscedastic cases. Overall, Table 4.1 shows that f-Cal is able to predict calibrated uncertainties without sacrificing significant deterministic performance. It is worth noting that we perform better than temperature scaling [**13**] despite this being an unfair comparison (temperature scaling leverages a large held-out calibration dataset, while we do not use any additional data).

4.5. Object detection

Setup: We use the popular Faster R-CNN [**87**] with a feature pyramid network [**60**] and a Resnet-101 backbone [**30**]. We use the publicly available Detectron2 [**103**] and PyTorch [**77**] implementation and extend the model to regress uncertainty estimates. For uncertainty estimation in object detectors, we add an uncertainty head in stage-2 of the network. We employ *xyxy* bounding box parameterization as used in [**32**] as opposed to *xywh* used in [**87**]. Using *xyxy* bounding box parameterization ensures that we have all linear transformations over our predictions to get final the bounding box. The reason for employing *xyxy* bounding box parameterization is to ensure that our final prediction over the bounding box is a Gaussian distribution. A Gaussian uncertainty going through a non-linear transformation may no longer be Gaussian. In the uncertainty head, we have a fully connected layer followed by a Generalized Sigmoid non-linearity, $g(x) = \alpha + \frac{\beta - \alpha}{1 + \exp(-\eta x)}$. Here, β is an upper asymptote, α is a lower asymptote and η is the sharpness. For all the experiments in this section, we have $\alpha = 0, \beta = 50$ and $\eta = 0.15$. These hyperparameters are chosen to have a wide range of uncertainty estimates. Using the generalized sigmoid function bounds our variance predictions as well as provides stable training dynamics. The training is divided into 3 stages. Let's say that we have C classes in our dataset. Here, we represent our region

[5] In typical ML settings, we never have access to this variable. We only ever access the noisy ground-truth that includes aleatoric uncertainty.

proposal's prediceted anchor offsets as $t_i = [t^i_{x_1}, t^i_{y_1}, t^i_{x_2}, t^i_{y_2}]$ and ground-truth anchor offsets as $t_{*i} = [t^{*i}_{x_1}, t^{*i}_{y_1}, t^{*i}_{x_2}, t^{*i}_{y_2}]$. If p_i is the probability of positive anchor and p_{*i} is the anchor label, then a multi-objective loss function to train the Region proposal network(RPN) can be expressed as below,

$$\mathcal{L}_{RPN} = \beta_1 \mathcal{L}_{BCE}(p_i, p_{*i}) + \beta_2 \mathcal{L}_{loc}(t_i, t_{*i}) \tag{4.5.1}$$

Here, $\mathcal{L}_{BCE}$ is a binary cross-entropy loss and $\mathcal{L}_{loc}$ is a smoothL1 loss. $\mathcal{L}_{loc}$ is evaluated only for positive anchors. We take the weighted combination of these losses, to form $\mathcal{L}_{RPN}$. $\mathcal{L}_{RPN}$ is then used to train the region proposal network. In stage-2, let's say $p = (p_1, p_2, ..., p_C)$ is output probability distribution over $C + 1$ classes, and u is the actual class of the ROI region. We have ground-truth ROI region offsets $B_g = [x^g_1, y^g_1, x^g_2, y^g_2]$. Our predicted ROI region offsets are represented by $\mu_d = [\mu^d_{x_1}, \mu^d_{y_1}, \mu^d_{x_2}, \mu^d_{y_2}]$. In probabilistic object detection, we actually predict $B_d = \mathcal{N}(\mu_d, \Sigma_d)$, where Σ_d is 4 x 4 matrix representing co-variance matrix for bounding box. In this work, we assume a diagonal co-variance matrix. Given this, we train stage-2 using the following loss function,

$$\mathcal{L}_{RCNN} = \alpha_1 \mathcal{L}_{cls}(p, u) + \alpha_2 \mathcal{L}_{loc}(B_g, \mu_d) \tag{4.5.2}$$

Here, $\mathcal{L}_{cls}$ is a multi-class cross entropy loss function and $\mathcal{L}_{loc}$ is a smoothL1 loss, same as one used to train RPN. Here, $\mathcal{L}_{loc}$ is only trained for foreground regions. $\mathcal{L}_{RCNN}$ is the weighted sum of classification and regression loss, used to train second stage of the 2-stage network. Given this, the training is divided into 3 stages.

(1) Train a deterministic model without uncertainty head with 4.5.1 and 4.5.2.

(2) Train an uncertainty head with NLL loss(3.1.6).

(3) Train stage-2 of the model with empirical risk as NLL loss(3.1.6) and $\mathcal{L}_{f\text{-Cal}}$.

All the baseline calibration models were initialized with NLL loss trained weights, and trained with a learning rate of 1e-4. Each model is trained, until optimal validation performance is achieved.

Dataset For **KITTI** [**17**], there are 7481 images with corresponding annotations. This data is divided intro train/test/val splits. 4500 images are used for training, 500 images are in the validation set and the remaining 2481 are in the test dataset.

We have exactly the same procedure for the **Cityscapes** [**7**] dataset as well. In Cityscapes, we have 3475 annotated images, out of which 2500 are used for training, 475 are used for validation, and 500 are used for testing the models. We perform similar holdout cross validation for Cityscapes also, and choose the best performing model for testing. For temperature scaling [**13**], we use the validation dataset to learn the temperature parameter. We train the temperature parameter until its value is stabilized.

Fig. 4.3. KITTI dataset [**17**] - sample images

Fig. 4.4. Cityscapes dataset [**7**] - sample images

4.5.1. Consistency metrics

In this section, we present two new metrics to evaluate consistency of an object detector. We report results for Expected calibration error (ECE) and Negative log likelihood (NLL), maximum calibration error (MCE), KLD, Wasserstein distance between proposed and target distributions. The above metrics reflect calibration quality of the models. In addition, for object detection, we also modify existing popular metrics such as mAP [**62**] and PDQ [**25**], according to definition 2, to report consistency of the models. Calibration generally implies consistency (vice versa is not true), and we wish to validate that.

From the definition of consistency (Definition 2), we can observe that the requirement for calibration is more stringent than that of consistency. We can have consistent probabilistic detection if we predict arbitrarily high uncertainty (Fig. 1.1(c)), though the uncertainties can not be interpreted as confidence scores. Through these new metrics, we show that consistency is the byproduct of calibration. We do not enforce any explicit constraints for consistency, yet we show in our results that we end up achieving highly consistent prediction. It is important to note that the consistency metrics we report do not automatically imply calibration. So these metrics should be interpreted in conjunction with calibration metrics.

Towards this end, we modify two popular object detection evaluation metrics, mAP and PDQ, for consistency estimation. The new metrics are a minor modification of mAP and PDQ, designed to evaluate consistency of the object detector. We use Definition 2 to build these metrics.

Formally, let's say our ground truth bounding box is $B_g = [x_1^g, y_1^g, x_2^g, y_2^g]$, represented by top left and bottom right corners of the bounding box. Our predicted box is represented by $B_d =$

$\mathcal{N}(\mu_d, \Sigma_d)$. Here, $\mu_d = [\mu^d_{x_1}, \mu^d_{y_1}, \mu^d_{x_2}, \mu^d_{y_2}]$. Σ_d is 4 x 4 matrix representing co-variance matrix for bounding box. For the purpose of this thesis, Σ_d is a diagonal co-variance matrix, however, it could be a full co-variance too and this formulation will still hold. The loss attenuation formulation looks as below,

$$L_{la} = (B_g - \mu_d)^T \Sigma_d^{-1} (B_g - \mu_d) + \log(det(\Sigma_d)) \tag{4.5.3}$$

In equation 4.5.3, the first term represents the squared Mahalanobis distance, which characterizes the number of standard deviations a point is away from the mean of a distribution. The squared Mahalanobis distance followed a Chi-squared distribution with K degrees of freedom. We obtain a Mahalanobis distance threshold $\mathcal{M}_{thresh}$ when we evaluate a Chi-squared distribution with K degrees of freedom and confidence interval α. In this case, it will denote the probability of the ground truth being in the hyper-ellipse defined by the squared Mahalanobis distance.

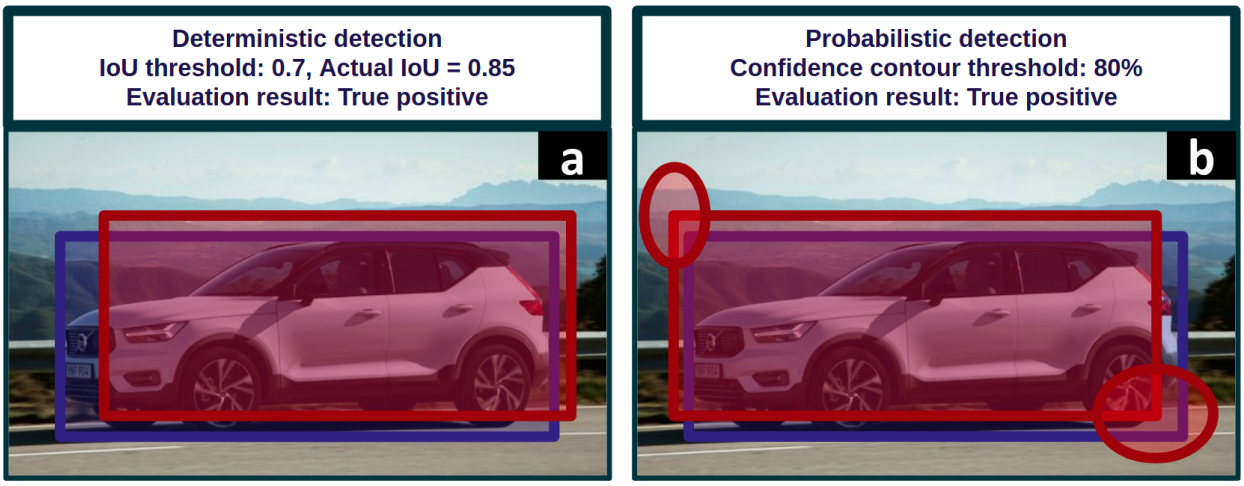


Fig. 4.5. The definition of true positive changes when we evaluate a probabilistic object detector. To evaluate consistency of probabilistic object detector, I propose a novel criteria based on confidence interval of the prediction, the ground-truth sample lies on. The blue box represents the ground truth object in both images. In the deterministic detection (a), we evaluate a proposal as a true positive if the IoU with the actual annotation is greater than a certain threshold. In probabilistic detection (b), we represent the uncertainty with ellipses. The ellipses visualized correspond to the 80% confidence contour, and we observe that the ground truth falls within 80% confidence contour, hence we classify that probabilistic detection as a true positive. We could perceive confidence contour threshold as similar to IoU threshold used in deterministic object detection, and just like mAP, where we evaluate the model for wide range of IoU thresholds, here also we evaluate consistency for a wide range of confidence contour thresholds.

In this work, we propose to use probability confidence between the ground truth and the predicted distribution as a quality measure for detection. The lower the Mahalanobis distance, the closer the ground-truth and the distribution are, the smaller the confidence contour would be.

Now we formally define a confidence contour and corresponding Mahalanobis distance. Let's consider the confidence contour of a Gaussian distribution's volume of α. It means that probability of a random variable X falling inside this confidence contour is α. In this case, the probability of random variable leaving this volume (also known as critical value) is $\beta = 1 - \alpha$. Then, the squared Mahalanobis distance threshold is $\mathcal{M}_{thresh} = \tilde{\chi}^2(K, \beta)$. This means that for a normal distribution $\mathcal{N}(\mu, \Sigma)$, if X falls in confidence contour α, then $(X-\mu)^T\Sigma^{-1}(X-\mu) \leq \tilde{\chi}^2(K, 1-\alpha)$.

Mean Mahalanobis average precision(mMAP)

Mean average precision(mAP) [**62**] has been the most popular metric to evaluate object detector. For consistency estimation, we modify this metric to incorporate probabilistic bounding box predictions. In mAP, precision is calculated for IoUs of 0.5 to 0.95 at the interval of 0.05. In mMAP, we replace the IoU threshold with the confidence contour thresholds. We keep the confidence contour as a threshold and determine true positives based on the Mahalanobis distance as explained in Fig. 4.5.

In this work, we have thresholds of [0.999, 0.995, 0.99 , 0.95, 0.9, 0.85, 0.8, 0.7]. We observe that mMAP is a more informative metric to analyse probabilistic consistency. The definition of consistency (Definition 2) requires the prediction to be in a certain confidence contour bound, c. This metric just evaluates that over multiple thresholds and averages across thresholds and categories just like mAP.

Probability-based detection quality(PDQ)

PDQ (Probability-based Detection Quality) [**25**] is a recently proposed metric to evaluate probabilistic object detectors. It incorporates spatial and label quality into the evaluation criteria, and explicitly rewards probabilistically accurate detections. Both spatial and label quality measures are calculated between all possible pairs of detections and the annotation. The geometric mean of these two measures is calculated and used to find the optimal assignment between all detections and annotations.

In PDQ, spatial quality is calculated by fusing background and foreground loss, which are computed using the ground truth segmentation mask and the probabilistic detection. This requires the availability of segmentation masks during test time which may not be possible for a bounding box based object detection dataset. In addition to that, the evaluation objective of spatial quality estimation is different compared to the training objective of probabilistic object detectors, which are trained with NLL loss. In contrast, the modified evaluation criteria (Fig. 4.5) evaluates spatial quality without the need of any segmentation mask. Incorporating the Mahalanobis distance based criteria enables the modified metric to evaluate consistency. Mahalanobis distance is a reflection of how many standard deviations away your mean prediction is compared to the sample (ground truth in this case). Less Mahalanobis distance could be interpreted as good spatial quality of the

Approach	mAP	AP50	AP75	mMAP	PDQ	PDQ (spatial)	PDQ (label)	NLL
NLL Loss (base)	54.451	78.476	62.876	76.76	0.601	0.725	0.976	1.022
NLL Loss (full)	51.764	76.245	58.893	35.066	0.443	0.483	0.975	0.932
Calibration loss	50.404	70.442	57.963	49.162	0.449	0.511	0.968	0.773
fCal-WS (ours)	48.04	77.768	53.107	73.525	0.565	0.703	0.968	0.914
fCal-KL (ours)	51.874	76.377	59.181	70.503	0.535	0.665	0.96	0.846
Temperature scaling	54.451	78.476	62.876	77.433	0.608	0.737	0.976	1.021

Table 4.2. Object detection - KITTI [**17**]: Consistency and deterministic results. This table shows results against various evaluation metrics for consistency and deterministic performance.

prediction. We redefine the spatial quality as below,

$$Q_s(B_g, B_d) = \exp(\frac{-(B_g - \mu_d)^T \Sigma_d^{-1} (B_g - \mu_d)}{T}), \tag{4.5.4}$$

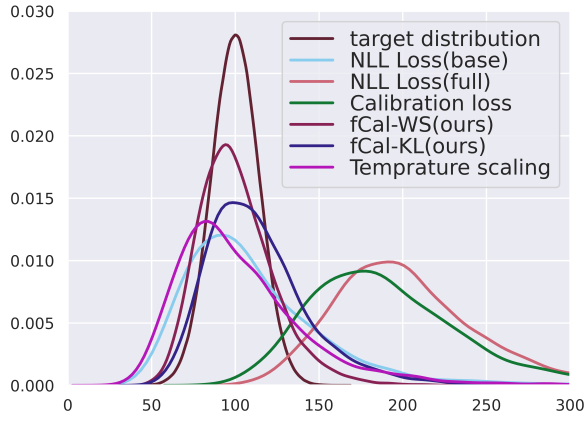
where T is the temperature parameter (Not to be confused with temperature parameter of the temperature scaling method). It determines how much the Mahalanobis distance should penalize spatial quality. The higher the temperature value, the lower the penalty. In our experiments, we keep the value of T to be 10.

Both these metrics complement mAP, which gives us estimates of how accurate our bounding boxes are. On a contrary, mMAP and modified PDQ will tell us how consistent our uncertainty values are. We can infer more about consistency of our models by analysing these metrics.

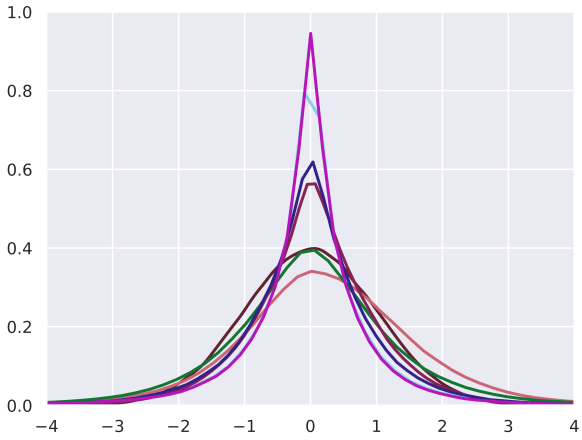
4.5.2. KITTI results

In table 4.2 and 4.3, we extensively report results for the KITTI [**17**] object detection dataset. We evaluate our model for various metrics and baselines. We see that f-Cal is able to obtain highly consistent and calibrated results. We observe that when we the train entire model with NLL loss[6] [**76**], due to its mean seeking nature, it is trying to maximize the likelihood, and predicting very low uncertainty values, resulting in overconfident uncertainty estimates. This results in inconsistent predictions as evident from the mMAP, PDQ and PDQ (spatial) values. Note that many baselines have high consistency values but poorer calibration, which is a result of highly inflated

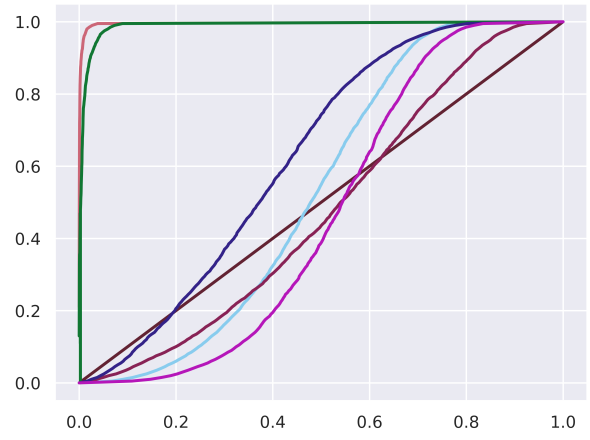
[6]In practice, NLL Loss is trained without freezing any part of the model. In this work, we just train the uncertainty head with NLL loss to have base model, which is used as weight initializer for other methods. NLL Loss (base) works better than NLL loss (full) for consistency and calibration.



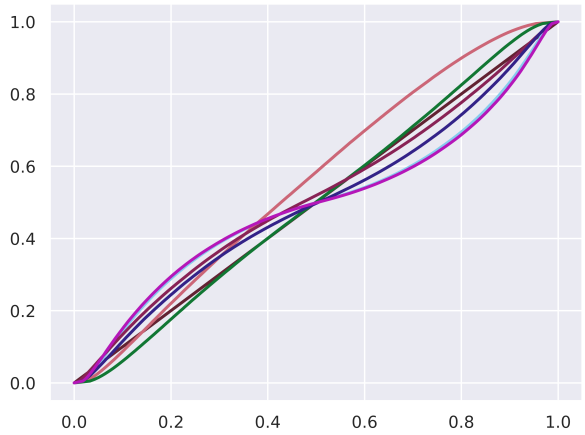
(a) Object detection - KITTI [**17**] - Chi-squared distribution plots of different baselines. To quantitatively understand the results, see **W-dist(q)** and **KLD(q)** columns of Table 4.3. Lower values of **W-dist(q)** and **KLD(q)** correspond to better curves.



(b) Object detection - KITTI [**17**] - standard normal distribution plots of different baselines. To quantitatively understand the results, see **W-dist(z)** and **KLD(z)** columns of Table 4.3.



(c) Object detection - KITTI [**17**] - Chi-squared reliability diagrams of different baselines. Closer to the diagonal is better. To quantitatively understand the results, see **MCE(q)** and **ECE(q)** columns of Table 4.3. Lower values of **MCE(q)** and **ECE(q)** reflect better curves.



(d) Object detection - KITTI [**17**] - standard normal reliability diagrams of different baselines. Closer to the diagonal is better. To quantitatively understand the results, see **MCE(z)** and **ECE(z)** columns of Table 4.3.

Fig. 4.6. Reliability diagrams and distribution plots for object detection - KITTI[**17**] dataset.

uncertainty estimates. High consistency won't be very useful if we do not have good calibration. So consistency metrics must be interpreted in conjunction with calibration metrics to make accurate conclusions. Currently, f-Cal achieves state-of-the-art calibration, while having competitive consistency.

Approach	ECE(z)	MCE(z)	ECE(q)	MCE(q)	W-dist(z)	KLD(z)	W-dist(q)	KLD(q)
NLL Loss (base)	0.00304	0.01396	0.0537	0.21358	0.005	0.004	1183.549	0.768
NLL Loss (full)	0.00345	0.0406	0.93312	0.96343	0.227	0.11	12190.394	3.052
Calibration loss	0.00233	0.02759	0.8261	0.90619	0.151	0.088	10065.516	2.435
fCal-WS (ours)	0.00115	0.00697	0.00697	0.06596	0.003	0.002	78.076	0.174
fCal-KL (ours)	0.00162	0.01175	0.0393	0.18911	0.005	0.004	528.606	0.517
Temperature scaling	0.00315	0.01268	0.04126	0.16199	0.002	0.001	742.99	0.631

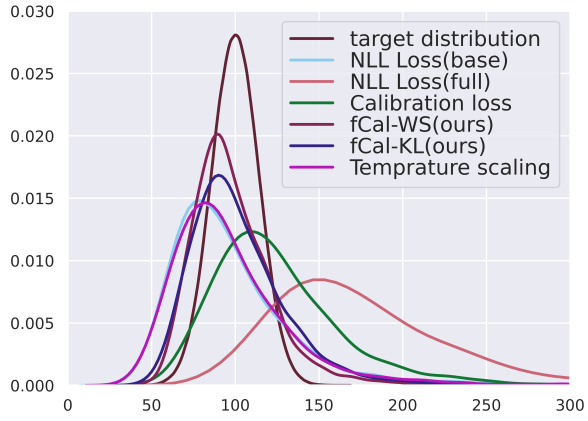
Table 4.3. Object detection - KITTI [**17**]: Results of f-Cal and other baselines for various calibration metrics.

Appoach	ECE(z)	MCE(z)	ECE(q)	MCE(q)	W-dist(z)	KLD(z)	W-dist(q)	KLD(q)
NLL Loss (base)	0.00224	0.00886	0.03503	0.12766	0.00225	0.00130	681.220	0.607
NLL Loss (full)	0.00146	0.02318	0.59936	0.77085	0.12145	0.07374	10953.997	1.768
Calibration loss	0.00163	0.01464	0.09681	0.30432	0.01529	0.01258	1501.167	0.848
fCal-WS (ours)	0.00104	0.00656	0.00832	0.06299	0.00022	0.00015	97.245	0.201
fCal-KL (ours)	0.00126	0.00880	0.01686	0.11125	0.00037	0.00025	304.647	0.403
Temperature scaling	0.00226	0.00928	0.02705	0.10635	0.00206	0.00110	754.356	0.637

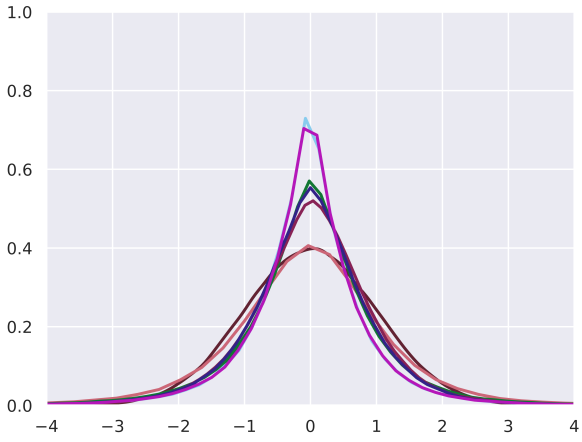
Table 4.4. Object detection - Cityscapes [**7**]: Results of f-Cal and other baselines for various calibration metrics.

4.5.3. Cityscapes results

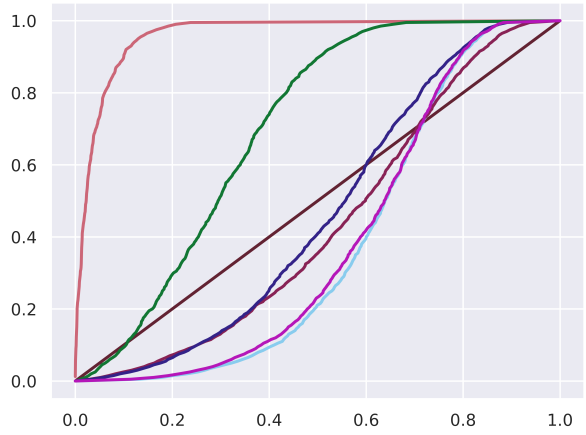
In Tables 4.4 and 4.5, we report results for the Cityscapes [**7**] dataset. We observe that f-Cal is able to obtain highly consistent and calibrated uncertainty estimates. We observe that the NLL loss (full) is providing overconfident uncertainty estimates, resulting in poor consistency. We observe that other baselines such as NLL loss (base) and calibration loss are able to obtain high consistency, but poorer calibration, which is a result of inflated uncertainty estimates. f-Cal and temperature scaling are able to yield calibrated and consistent uncertainty estimates, while retaining deterministic performance. However, temperature scaling had a held out validation set for tuning the temperature parameter, while f-Cal results are directly obtained using training data only. We also note that f-Cal enables the model to learn uncertainty-aware representations, as



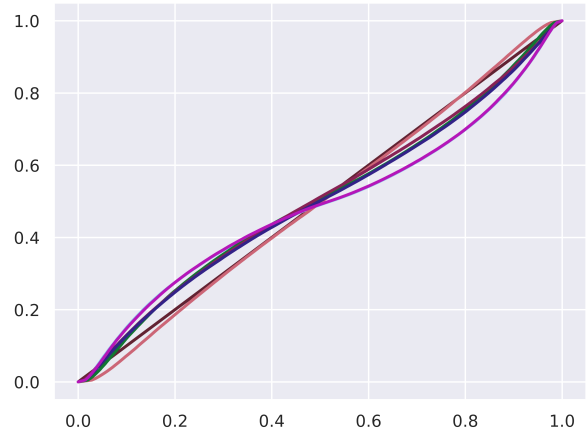
(a) Object detection - Cityscapes [7] - Chi-squared distribution plots of different baselines. To quantitatively understand the results, see **W-dist(q)** and **KLD(q)** columns of Table 4.4. Lower values of **W-dist(q)** and **KLD(q)** correspond to better curves, which can be visually understood in the figure.



(b) Object detection - Cityscapes [7] - standard normal distribution plots of different baselines. To quantitatively understand the results, see **W-dist(z)** and **KLD(z)** columns of Table 4.4.



(c) Object detection - Cityscapes [7] - Chi-squared reliability diagrams of different baselines. To quantitatively understand the results, see **MCE(q)** and **ECE(q)** columns of Table 4.4. Lower values of **MCE(q)** and **ECE(q)** reflect better curves.



(d) Object detection - Cityscapes [7] - standard normal reliability diagrams of different baselines. To quantitatively understand the results, see **MCE(z)** and **ECE(z)** columns of Table 4.4.

Fig. 4.7. Reliability diagrams and distribution plots for Object detection - Cityscapes [7] dataset.

opposed to temperature scaling, where representations learned are same as those of uncalibrated models. In Fig. 4.7, we show distributions and reliability diagrams of all the different baselines and f-Cal.

4.6. KITTI Depth Estimation

Setup: To test the scalability of f-Cal, we also evaluate it on the the task of depth estimation. We train f-Cal and several baseline calibration techniques on the KITTI depth estimation benchmark

method	mAP	AP50	AP75	mMAP	PDQ	PDQ (spatial)	PDQ (label)	NLL
NLL Loss (base)	38.309	61.548	39.142	49.380	0.454	0.613	0.910	1.069
NLL Loss (full)	36.199	55.878	39.283	23.763	0.361	0.453	0.934	1.029
Calibration loss	39.218	61.922	40.220	42.302	0.424	0.560	0.920	0.999
fCal-WS (ours)	37.220	61.486	38.469	46.202	0.442	0.593	0.911	1.007
fCal-KL (ours)	38.481	61.924	40.210	43.982	0.442	0.584	0.915	0.929
Temperature scaling	38.309	61.548	39.142	48.965	0.452	0.610	0.911	1.065

Table 4.5. Object detection - Cityscapes[**7**]: Consistency and deterministic results. This table shows results against various evaluation metrics for consistency and deterministic performance.

dataset [**17**]. We modify the Big2small model [**55**] for supervised depth estimation into a Bayesian Neural Network by adding a variance decoder. Big2small model's encoder is a standard deep feature extractor, while decoder uses dense atrous spatial pyramid pooling layer, which along with local planar guidance module give predicted depth map. We evaluate the deterministic performance using SiLog and RMSE metrics and calibration using ECE(z), ECE(q) and NLL. Let y_i where $i \in \{1,2,3,...,N\}$ be the predicted depth and y_i^* be the groundtruth depth. Then RMSE and SiLog can be expressed as below,

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{n} ||y_i - y_i^*||^2}$$

$$SiLog = \frac{1}{N}\sum_{i=1}^{N} 2\log\left(\frac{y_i}{y_i^*}\right) - \frac{1}{N^2}\left(\sum_{i=1}^{N}\log\left(\frac{y_i}{y_i^*}\right)\right)^2$$

The loss function used for training these models was $\mathcal{L} = \mathcal{L}_{reg} + \lambda * \mathcal{L}_{cal}$, where $\mathcal{L}_{cal}$ can be NLL, calibration loss or f-Cal.

4.6.1. Accuracy v/s calibration trade-off:

Discussion: Through our experiments, we conclude that there is a trade-off between determinisitic and calibration performance (also established in [**22, 13**]). We can control this trade-off by varying the λ in Eq. 3.3.6. By plotting SiLog and ECE(z) for different values of λ we can analyze

Approach	SiLog	RMSE	ECE(z)	ECE(q)	NLL
NLL Loss[76]	9.213 ± 0.092	2.850 ± 0.035	2.39 ± 0.224	99.9 ± 0.001	3.403 ± 0.258
Calibration Loss[13]	9.604 ± 0.165	2.918 ± 0.015	1.71 ± 0.412	99.9 ± 0.000	2.878 ± 0.262
Temperature Scaling[13]	9.213 ± 0.092	2.850 ± 0.035	2.36 ± 0.214	99.9 ± 0.004	3.362 ± 0.221
f-Cal-KL (ours)	9.679 ± 0.091	2.911 ± 0.293	0.074 ± 0.021	22.5 ± 13.684	2.004 ± 0.143
f-Cal-Wass (ours)	9.509 ± 0.098	3.202 ± 0.247	0.156 ± 0.044	67.9 ± 9.616	2.157 ± 0.159

Table 4.6. Depth Regression - KITTI [17]: f-Cal on average gives better calibration performance in comparison with the baselines. ECE scores have been scaled by 100 to enhance readability

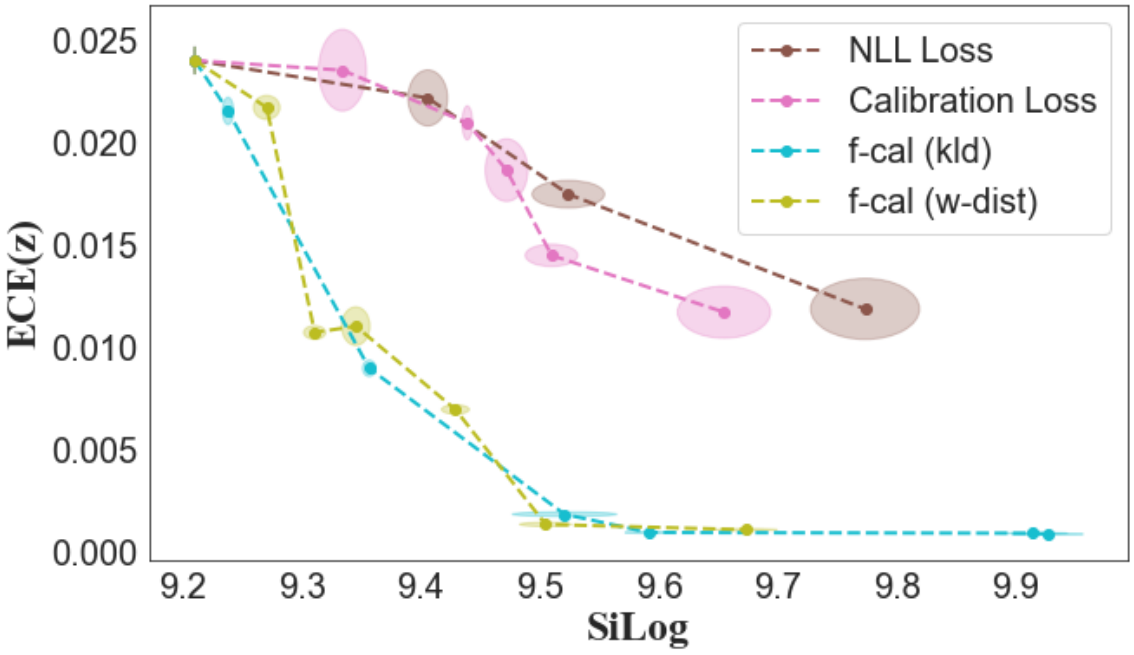


Fig. 4.8. Calibration vs. deterministic performance trade-off: We see that this trade-off is observed for all the three calibration techniques. For similar deterministic performance f-Cal models are able to achieve smaller ECE values (i.e., better calibration).

this trade-off for the baseline calibration techniques. We run each experiment over 5 seeds to capture the variability in deterministic and calibration performances represented by ellipses in Fig. 4.8. f-Cal models result in better calibration performance than the baselines for similar deterministic scores.

In Table. 4.6, for every method we select a λ which best balances between deterministic performance and calibration. For this fixed λ we run the experiment over multiple seeds and report mean scores (full results are given in Table. 4.6). We see that f-Cal outperforms all baselines on all calibration metrics. We also observe that unlike Bokeh (Table. 4.1), temperature scaling struggles

to calibrate uncertainties by tuning a single temperature parameter on such a large and complex dataset.

4.7. Ablation

4.7.1. Impact of modeling assumption

We postulate that for real-world datasets such as KITTI [**17**], the trade-off in calibration and deterministic performance occurs due to poor modeling assumptions (i.e., modeling uncertainty using a distribution family that is quite different from the underlying label error distribution). We carry out an analysis on the impact of the modeling assumption needed to perform distribution matching. Through controlled experiments on the Bokeh dataset, we observe that a reduction in deterministic performance could be due to the true label error distribution not matching the assumed label error distribution. To investigate this, we introduce a mismatch between the true distribution, a Gamma distribution parameterized by the shape parameter γ[7], and the model distribution, a Gaussian distribution, on the synthetic (Bokeh) dataset (Fig. 4.9 (left)). For lower distributional mismatch (higher γ), the performance gap between the calibrated and deterministic models is reduced. The hypothesis is that KITTI doesn't align well with the modeling assumption, while Bokeh (synthetic) dataset does (by construction), resulting in a trade-off in the former.

Note that this facet of our approach allows us to empirically estimate the underlying distribution that is producing the aleatoric uncertainty. In the case that the *modeling* assumptions about the noise distribution is not correct, we can detect it by observing a larger drop in deterministic performance in order to achieve calibration. If there is no degradation in deterministic performance, then it is more likely that we will be able to jointly minimize the two terms in Eq. 3.3.6. An interesting avenue for future work could be to automate this process of determining the true underlying noise-generating distribution family.

4.7.2. Effect of degrees of freedom (K)

We analyze how the number of degrees of freedom in the Chi-squared distribution (K) would impact calibration performance. We train models with different values of K and measure the degree of calibration. In Fig. 4.9 (right), we can observe that for $K > 50$, the central limit theorem holds and we see superior calibration when compared with models trained for $K \leq 50$, when our approximation of a Gaussian distribution breaks, resulting in poor calibration.

[7] note that the γ distribution converges to a Gaussian for large values of the shape parameter

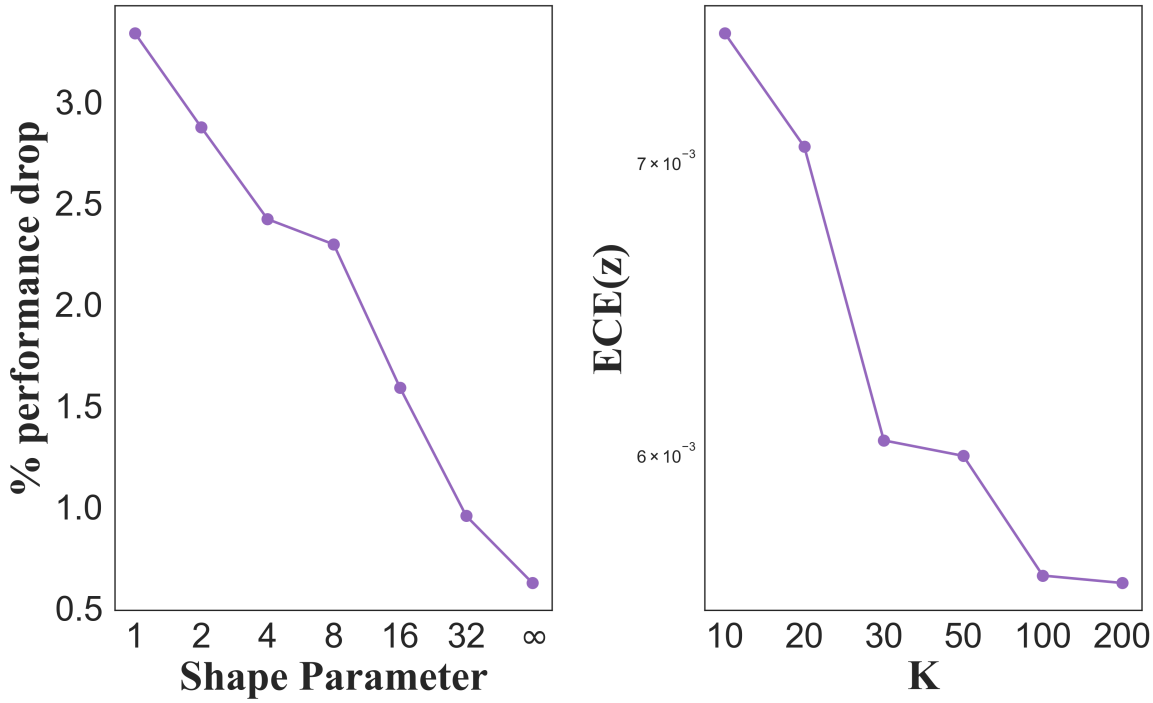


Fig. 4.9. Ablation: (left) We plot the % drop in deterministic performance compared to a deterministic model for different noise distributions. For large shape parameter, the Gamma distribution converges to a Gaussian, resulting in nearly identical performance to a deterministic model. (right) Effect of K on the performance of f-Cal, we see that as long as $K > 50$, the central limit theorem holds and we get good calibration.

Chapter 5

Distributional uncertainty in Object detection

In this chapter, we will discuss the out-of-distribution (OOD) detection problem for object detection. Predominant architectures for object detection include: single-stage methods [**94, 90, 84, 85, 65**] and two-stage methods [**20, 29, 18, 87, 9, 28, 58, 61, 60**]. Most of these use "anchor boxes" to handle variation in the number, size, and position of objects in an image. Each anchor box is classified as either an in-distribution class or the background class. Unlike classification, in object detection, if we have n in-distribution classes, the network learns to predict $n + 1$ classes to incorporate background anchors. Current state of the art 1-stage architecture([**61**]) and 2-stage architecture([**87, 60**]) are both anchor-based with a background class.

So far, OOD detection methods [**34, 54, 33, 59, 22, 35, 72, 69, 89**] have been designed exclusively for image classification. Object detection methods, on the other hand, bear a different design philosophy. The notion of a *background* class in object detection obfuscates the distinction between novel objects and the background class. This necessitates discovery of background-aware OOD detection methods to detect novel objects in an image. While some zero- and few-shot learning methods recognize novel objects in an image, zero-shot methods can only recognize OOD objects that are close to the training distribution [**83, 1, 109, 81, 82, 23**]. These methods employ usage of visual-semantic embeddings. This enables the detector to recognize novel objects which are semantically close to in-distribution objects. This assumes some semantic relation between seen and unseen classes. However, these methods would fail to recognize a novel object which is semantically different from in-distribution objects. Some few shot learning methods [**100, 43, 105, 44, 12**] port over the few-shot meta-learning setup for the specific setting of object detection and develop algorithms to enable object detectors to detect and localise a set of new target objects given limited samples for them.

We observe that it is arduously challenging to distinguish between the background class and a novel object, owing to design decisions baked into modern object detectors. We now formalize the definition of OOD for object detection. There are two types of OOD objects in this setup:

- *Seen* OOD objects: In object detection, Seen OOD objects are unannotated objects that are present in training images. We call such objects "seen OOD objects" because they are part of the training distribution, but due to absence of any label, they are automatically labeled as background.
- *Unseen* OOD objects: Unseen OOD objects are completely novel objects which have not been encountered during training in any form. Such objects are seen during test time only. For instance, a cow on the road is a novel occurrence, when the detection models have been trained on standard driving datasets [**16**] including only constrained urban scenes.

We conducted an experiment to understand the adverse impact of the background class for the "seen OOD objects" case. In this experiment, we use a clustering based method to classify OOD objects. The overall objective in this method is to view intermediate representations of an object in the trained model (input to softmax layer[1]) as residing on a high-dimensional manifold, and then, assuming that in-distribution samples should be "close" to each other on this manifold [**99**]. We employ class conditional Gaussian clustering inspired from [**56**]. The embedding representation in the layer immediately preceding the softmax layer is a suitable choice for perception/classification tasks based on the intuition that the network is trying to find a representation at this layer that is easily separable by the softmax.

We train a model on l classes of the dataset and hold out k classes. We treat these k classes as *seen OOD objects*. We use the KITTI Object dataset [**17**] with $l = 4$ and $k = 3$. We follow the procedure below,

(1) Train the probabilistic detector to convergence.
(2) Get embeddings of in-distribution objects and background.
(3) For each class c calculate a class-conditional Gaussian distribution (μ_c,Σ_c) based on the cluster embeddings
(4) For a test datum x, calculate the closest class cluster in the embedding space, where "closeness" is determined by Mahalanobis distance:

$$M(x) = \max_c -(f(x) - \mu_x)\Sigma_c^{-1}(f(x) - \mu_x) \tag{5.0.1}$$

(5) If the closest class is above a threshold Mahalanobis distance, then the input is designated as OOD since it does not correspond well to any of the known classes.

We treat an acceptable accuracy of the in-distribution samples as the control variable. This control variable automatically presents a threshold on the Mahalanobis distance. Following the procedure above, for each class, we obtain several data points in the validation set and compute a Gaussian mean and covariance using the training data. We then use the validation data to find the Mahalanobis distance threshold that will achieve the acceptable accuracy, across all object classes. Next, using this threshold, we determine whether a new embedding belongs to any of these classes

[1]Note that getting intermediate representations is possible in 2-stage object detection only, we get intermediate representation from stage-2 of the Faster RCNN

or not. If it doesn't we label it OOD. We perform this experiment for the two problem setups as explained below,

- *Setup 1 ("Easy" version without background)*: In this setup, we only keep in-distribution objects and exclude the background class. If the Mahalanobis distance of an OOD embedding with a class conditional Gaussian is greater than a Mahalanobis threshold of the corresponding class, it is considered as a true positive (classified as OOD). Otherwise, it is deemed a misclassification.
- *Setup 2 ("Hard" version with background):* In this setup, the Mahalanobis distance is also calculated with background Gaussian statistics. Giving OOD embedding chance to be classified as background.

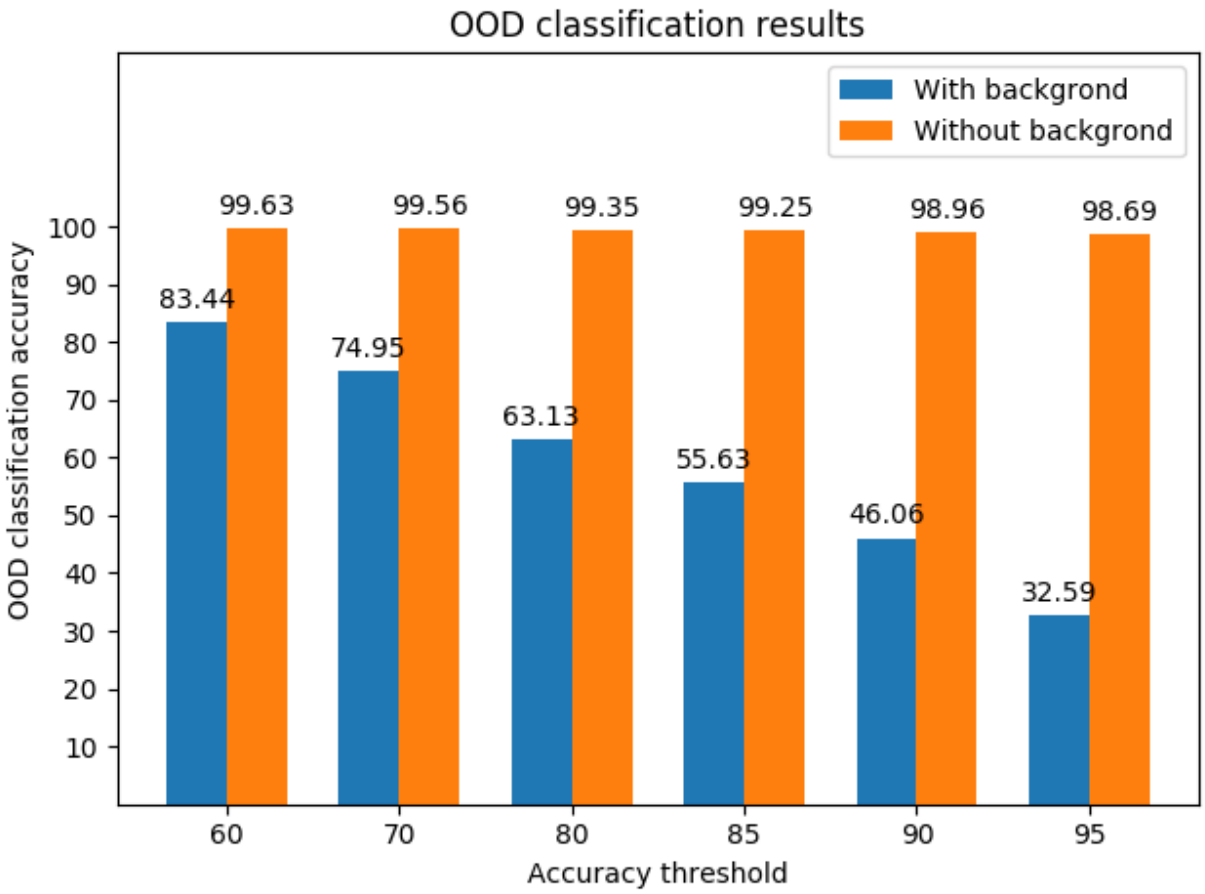


Fig. 5.1. Background class hampers OOD detection performance as the cross-entropy objective forces an OOD object to be very confidently classified as background. We demonstrate that removing the notion of a background class boosts performance. Our results are also corroborated by [**11**].

We run experiments for both the above setups, for various acceptable accuracy thresholds, and then we calculate the accuracy of OOD classes explicitly. To effectively understand the role of

the background class, we follow the procedure outlined in 5 for both the setups (with and without background class). Results are shown in Fig 5.1.

This demonstrates that *redesigning object detectors without allowing a catch-all "background" class can boost OOD detection performance*.

Chapter 6

Conclusion

6.1. Limitations

f-Cal, while being simple and effective, has a few limitations that warrant deeper investigation and follow-up work.

(1) *Need large values for K*: The choice of K (Fig. 4.9(right)) has an impact on calibration quality in that lower values ($K < 30$) incur larger calibration errors. This limits the applicability of scenarios to very sparse regression tasks (e.g., when there is only a single scalar predicted per input image). A workaround would be to use a much larger batch size during training, which is compute-intensive.

(2) *Performance relies on choice of modeling distribution*: As with any Bayesian deep learning approach, the success of f-Cal relies crucially on the choice of the modeling error distribution. If the modeled error distribution does not match the underlying true distribution, we observe a substantial performance drop (Fig. 4.9(left)). However, f-Cal can guide a practitioner towards a good modeling choice (by tracking the drop in model performance).

(3) *Limited distribution families*: The f-Cal loss can only be applied to distribution families that admit an analytical $f-$divergence computation, and allow for a canonical element to be computed (such as $z = \frac{y-\mu}{\sigma}$ for a Gaussian).

Despite these limitations, f-Cal is of high practical relevance, as demonstrated by our large-scale real-world experiments on object detection and depth estimation.

6.2. Conclusion

In this thesis, I presented f-Cal, a simple and efficient technique based on variational inference for calibrating aleatoric uncertainty estimates from a neural regressor. Different from previous work, f-Cal does not require a large held-out calibration dataset or post-hoc computation. Unlike previous methods, which mostly operated on per-sample basis, this method operates on a batch of samples, to enforce calibration across the samples. The key insight was that calibration is typically

achieved by looking at multiple data samples at the same time. This key insight motivated us to enforce a distribution matching loss, as opposed to per-sample loss, to achieve calibrated uncertainty estimates. This distribution matching loss function is applied in conjunction with typical supervised learning losses. This allows us to implement f-Cal as an easy add-on to a typical aleatoric uncertainty regression training loop, in about $10-15$ lines of PyTorch code. Introduction of hyper-constraint during training enforced stronger calibration during training. The stochastic nature of this hyper-constraint enabled our model to capture the true underlying aleatoric uncertainty of the data. I believe that this method opens up several avenues for employing well-calibrated uncertainties in downstream modules like sensor fusion, planning in a typical robotics stack. For example, uncertainties associated with object detectors can be employed in object-based state estimation or in model-predictive control loops. I also characterize the limitations of our technique that I believe will lead to interesting follow-up work. Another interesting avenue for future work could be to investigate non-iid settings – common in sequential and online learning scenarios. I also presented a method of distributional uncertainty estimation in context of object detection. Out-of-distribution detection has received much attention in classification setups. I employed a standard classification OOD detection method for object detection. A controlled experimental setup demonstrated that typical distributional uncertainty estimation methods are not applicable to object detection setup, owing to background class. The design of state of the art object detector forces presence of background class as a catch-all class. This presence massively blurs the distinction between OOD and background class. Background class, without any semantic structure or meaning, can have arbitrarily high entropy. This may be consuming OOD objects into the background class. Interesting future work could be to design object detectors which can capture generic object properties in the bounding box candidate, which can definitely help in overcoming this challenge. Another interesting avenue to explore would be designing an object detector without background class, which may eliminate this problem.

References

[1] Ankan Bansal, Karan Sikka, Gaurav Sharma, Rama Chellappa, and Ajay Divakaran. Zero-shot object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 384–400, 2018.

[2] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. CodeSLAM—learning a compact, optimisable representation for dense visual SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[3] C Blundell, J Cornebise, K Kavukcuoglu, and D Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

[4] George EP Box, William H Hunter, Stuart Hunter, et al. *Statistics for experimenters*, volume 664. John Wiley and sons New York, 1978.

[5] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, 2017.

[6] Jiwoong Choi, Dayoung Chun, Hyun Kim, and Hyuk Jae Lee. Gaussian YOLOv3: An accurate and fast object detector using localization uncertainty for autonomous driving. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:502–511, 2019.

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[8] Peng Cui, Wenbo Hu, and Jun Zhu. Calibrated reliable regression using maximum mean discrepancy. *arXiv preprint arXiv:2006.10255*, 2020.

[9] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[10] Frank Dellaert and Michael Kaess. Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[11] Taylor Denouden. An application of out-of-distribution detection for two-stage object detection networks. Master's thesis, University of Waterloo, 2020.

[12] Qi Fan, Wei Zhuo, and Yu-Wing Tai. Few-Shot Object Detection with Attention-RPN and Multi-Relation Detector. *ArXiv*, abs/1908.01998, 2019.

[13] Di Feng, Lars Rosenbaum, Claudius Glaeser, Fabian Timm, and Klaus Dietmayer. Can We Trust You? On Calibration of a Probabilistic Object Detector for Autonomous Driving. 2, 2019.

[14] Yarin Gal. Uncertainty in deep learning. *University of Cambridge*, 1(3), 2016.

[15] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *ArXiv*, abs/1506.02142, 2016.

[16] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[17] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[18] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[19] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[20] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2015.

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[22] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017.

[23] D. Gupta, A. Anantharaman, N. Mamgain, S. K. S, V. N. Balasubramanian, and C. V. Jawahar. A Multi-Space Approach to Zero-Shot Object Detection. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1198–1206, 2020.

[24] Tuomas Haarnoja, Anurag Ajay, Sergey Levine, and Pieter Abbeel. Backprop kf: Learning discriminative deterministic state estimators. In *Advances in Neural Information Processing Systems*, pages 4376–4384, 2016.

[25] David Hall, Feras Dayoub, John Skinner, Haoyang Zhang, Dimity Miller, Peter Corke, Gustavo Carneiro, Anelia Angelova, and Niko Sünderhauf. Probabilistic object detection: Definition and evaluation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1031–1040, 2020.

[26] Ali Harakeh, Michael Smart, and Steven L. Waslander. BayesOD: A Bayesian Approach for Uncertainty Estimation in Deep Object Detectors.

[27] Tibshirani Hastie and Robert Tibshirani. Friedman. *The Elements of Statistical Learning," Springer*, page 52, 2001.

[28] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[31] Yihui He, Chenchen Zhu, Jianren Wang, Marios Savvides, and Xiangyu Zhang. Bounding box regression with uncertainty for accurate object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:2883–2892, 2019.

[32] Yihui He, Chenchen Zhu, Jianren Wang, Marios Savvides, and Xiangyu Zhang. Bounding box regression with uncertainty for accurate object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2888–2897, 2019.

[33] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 41–50, 2019.

[34] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[35] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018.

[36] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

[37] Yvonne HS Ho and Stephen MS Lee. Calibrated interpolated confidence intervals for population quantiles. *Biometrika*, 92(1):234–241, 2005.

[38] 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.

[39] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[40] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[41] Po-Yu Huang, Wan-Ting Hsu, Chun-Yueh Chiu, Ting-Fan Wu, and Min Sun. Efficient uncertainty estimation for semantic segmentation in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 520–535, 2018.

[42] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[43] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-Shot Object Detection via Feature Reweighting. pages 8419–8428, 10 2019.

[44] Leonid Karlinsky, Joseph Shtok, Sivan Harary, Eli Schwartz, Amit Aides, Rogerio Feris, Raja Giryes, and Alex Bronstein. RepMet: Representative-Based Metric Learning for Classification and Few-Shot Object Detection. pages 5192–5201, 06 2019.

[45] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015.

[46] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017.

[47] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[48] Volodymyr Kuleshov and Stefano Ermon. Reliable confidence estimation via online learning. *arXiv preprint arXiv:1607.03594*, pages 2586–2594, 2016.

[49] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. *arXiv preprint arXiv:1807.00263*, 2018.

[50] Meelis Kull, Miquel Perello Nieto, Markus Kängsepp, Telmo Silva Filho, Hao Song, and Peter Flach. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. *Advances in neural information processing systems*, 32:12316–12326, 2019.

[51] Ananya Kumar, Percy S Liang, and Tengyu Ma. Verified uncertainty calibration. In *Advances in Neural Information Processing Systems*, pages 3792–3803, 2019.

[52] Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. Trainable calibration measures for neural networks from kernel mean embeddings. In *International Conference on Machine Learning*, pages 2805–2814, 2018.

[53] Yongchan Kwon, Joong-Ho Won, Beom Joon Kim, and Myunghee Cho Paik. Uncertainty quantification using bayesian neural networks in classification: Application to ischemic stroke lesion segmentation. 2018.

[54] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles, 2017.

[55] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*, 2019.

[56] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018.

[57] Dan Levi, Liran Gispan, Niv Giladi, and Ethan Fetaya. Evaluating and calibrating uncertainty prediction in regression tasks. *arXiv preprint arXiv:1905.11659*, 2019.

[58] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. Light-head r-cnn: In defense of two-stage object detector. *arXiv preprint arXiv:1711.07264*, 2017.

[59] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

[60] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[61] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[62] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[63] Chao Liu, Jinwei Gu, Kihwan Kim, Srinivasa G Narasimhan, and Jan Kautz. Neural rgb (r) d sensing: Depth and uncertainty from a video camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[64] Shichen Liu, Mingsheng Long, Jianmin Wang, and Michael I Jordan. Generalized zero-shot learning with deep calibration network. In *Advances in Neural Information Processing Systems*, pages 2005–2015, 2018.

[65] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[66] Christos Louizos and Max Welling. Structured and efficient variational deep learning with matrix gaussian posteriors. In *International Conference on Machine Learning*, pages 1708–1716, 2016.

[67] Christos Louizos and Max Welling. Multiplicative Normalizing Flows for Variational Bayesian Neural Networks, 2017.

[68] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, pages 13153–13164, 2019.

[69] Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems*, pages 7047–7058, 2018.

[70] Juan Maronas, Roberto Paredes, and Daniel Ramos. Calibration of deep probabilistic models with decoupled bayesian neural networks. *Neurocomputing*, 407:194–205, 2020.

[71] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *arXiv*, 2020.

[72] Sina Mohseni, Mandar Pitale, JBS Yadawa, and Zhangyang Wang. Self-Supervised Learning for Generalizable Out-of-Distribution Detection.

[73] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, 2016.

[74] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[75] Mahdi Pakdaman Naeini, Gregory F Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, volume 2015, page 2901. NIH Public Access, 2015.

[76] David A Nix and Andreas S Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, volume 1, pages 55–60. IEEE, 1994.

[77] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[78] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

[79] Janis Postels, Francesco Ferroni, Huseyin Coskun, Nassir Navab, and Federico Tombari. Sampling-free epistemic uncertainty estimation using approximated variance propagation. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[80] Samuel Prentice and Nicholas Roy. The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance. *The International Journal of Robotics Research*, 28(11-12):1448–1465, 2009.

[81] Shafin Rahman, Salman Khan, and Nick Barnes. Polarity Loss for Zero-shot Object Detection. *arXiv e-prints*, page arXiv:1811.08982, November 2018.

[82] Shafin Rahman, Salman Khan, and Nick Barnes. Transductive Learning for Zero-Shot Object Detection. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[83] Shafin Rahman, Salman Khan, and Fatih Porikli. Zero-Shot Object Detection: Learning to Simultaneously Recognize and Localize Novel Concepts. In C. V. Jawahar, Hongdong Li, Greg Mori, and Konrad Schindler, editors, *Computer Vision – ACCV 2018*, pages 547–563, Cham, 2019. Springer International Publishing.

[84] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[85] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[86] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[87] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[88] M Rueda, S Martínez-Puertas, H Martínez-Puertas, and A Arcos. Calibration methods for estimating quantiles. *Metrika*, 66(3):355–371, 2007.

[89] Vikash Sehwag, Arjun Nitin Bhagoji, Liwei Song, Chawin Sitawarin, Daniel Cullina, Mung Chiang, and Prateek Mittal. Analyzing the robustness of open-world machine learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 105–116, 2019.

[90] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[91] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[92] Hao Song, Tom Diethe, Meelis Kull, and Peter Flach. Distribution calibration for regression. *arXiv preprint arXiv:1905.06023*, 2019.

[93] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[94] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *Advances in neural information processing systems*, pages 2553–2561, 2013.

[95] Natasa Tagasovska and David Lopez-Paz. Single-model uncertainties for deep learning. In *Advances in Neural Information Processing Systems*, pages 6417–6428, 2019.

[96] Maxime Taillardat, Olivier Mestre, Michaël Zamo, and Philippe Naveau. Calibrated ensemble forecasts using quantile regression forests and ensemble model output statistics. *Monthly Weather Review*, 144(6):2375–2393, 2016.

[97] Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 13888–13899, 2019.

[98] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[99] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, Aaron Courville, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states, 2018.

[100] Xin Wang, Thomas E. Huang, Trevor Darrell, Joseph E. Gonzalez, and Fisher Yu. Frustratingly Simple Few-Shot Object Detection. *arXiv e-prints*, page arXiv:2003.06957, March 2020.

[101] Sarah E Webster, Ryan M Eustice, Hanumant Singh, and Louis L Whitcomb. Advances in single-beacon one-way-travel-time acoustic navigation for underwater vehicles. *The International Journal of Robotics Research*, 31(8):935–950, 2012.

[102] Jonathan Wenger, Hedvig Kjellström, and Rudolph Triebel. Non-parametric calibration for classification. In *International Conference on Artificial Intelligence and Statistics*, pages 178–190. PMLR, 2020.

[103] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[104] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[105] Xiaopeng Yan, Ziliang Chen, Anni Xu, Xiaoxi Wang, Xiaodan Liang, and Liang Lin. Meta R-CNN: Towards General Solver for Instance-Level Low-Shot Learning. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9576–9585, 2019.

[106] Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In *Icml*, volume 1, pages 609–616. Citeseer, 2001.

[107] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699, 2002.

[108] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[109] Pengkai Zhu, Hanxiao Wang, and Venkatesh Saligrama. Zero shot detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(4):998–1010, 2019.